Python İle Makine Öğrenmesi

PYTHON İLE MAKINE ÖĞRENMESI

First edition. October 12, 2020.

ISBN: 979-8201940805

Written by Sadi Evren SEKER and Furkan GÜRÇAY.


Bilgisayar Kavramları
Bilkav Eğitim Danışmanlık A.Ş.

Önsöz

Bu bölümde makine öğrenmesine giriş yapılacaktır.

Bu kitaptaki hedeflenen temel amaçlar; makine öğrenmesinin ne olduğu, gerçek hayatta makine öğrenmesinin hangi konularda kullanıldığı ve gerçek hayatta öğrendiğimiz bilgileri hangi alanlarda uygulayabileceğimizi anlamaya çalışmaktır. Makine öğrenmesi, bilgisayarların ve verinin olduğu her alanda kullanılabilir. Verinin olduğu her alan makine öğrenmesi için uygundur. Bilgisayar ve makineler ile neler yapılabilir, bu kitap kapsamında anlatılan bilgiler nerelerde kullanılabilir gibi sorulara değinilecektir.

İlk olarak bilgisayar ile görü (computer vision), bilgisayarın gerçek hayattaki nesneleri tanıması ve bu nesnelere göre aksiyon alması makine

öğrenmesinin ilk ele aldığı konulardandır. Makine dediğimiz bilgisayarlar ve bilgisayarlar üzerinden çalışan yapay zekanın kullanıldığı yazılım iki şekilde düşünülebilir. İlk olarak, bilgisayardan gerçek dünyaya giden bir aksiyon vardır ve buna takiben ikinci olarak ise gerçek dünyadan algılanan, sensörler vasıtası ile hissedilen, toplanan bilgi vardır. Makinenin görevi alınan ve geri verilen bilgiler üzerine çalışmaktır. Literatürde bu durum Man-Machine Boundary ismi ile anılmaktadır. Bu nokta, insan ile makinenin ayrıldığı sınır noktasıdır. Bilgisayar ile görü, bizim gözle gördüğümüz şeyleri bilgisayarın görüp algılaması ile ilgili bir dünyayı konu almaktadır. Bu alan da hareketlerin algılanması, üzerinde yeni çalışılan konulardan biridir. Örneğin, hava alanında bir insan çantasını bırakıp koşmaya başlarsa, koşan bir insanın algılanması zamana yayılı bir süreçtir. Video dediğimiz anlık olarak çekilen birçok fotoğraf karesinin arka arkaya oynatılmasından oluşmak-

tadır. Bu süreçte koşmak gibi bir olayın tekrarlanıyor olması, koşma olayının çantayı bırakmak olayının arkasından geliyor olması ve bu durumların algılanarak bunun bir tehdit olacağının anlaşılması üzerine bu olay hakkında bir aksiyon alması günümüzde üzerinde çalışılan bilgisayar ile görü konularına örnek olarak gösterilebilir. Bir başka örnek olarak ise çekilmiş bir fotoğrafın üzerinden veya bir robotun gözünden görülen görüntünün objeleri tanımak için kullanılması gösterilebilir. Bir süpermarket, kamerası aracılığı ile insanları tanıyarak bu objeler hakkında cinsiyet, yaş, mutlu veya mutsuz olmaları gibi durumlarda yorum üretebilir. Bunun sonucunda ise, elde edilen çıkarımlar ile süpermarket, hangi tür ürünleri daha çok bulundurması gerektiğini veya müşterinin memnuniyet durumu gibi birçok bilgiyi ortaya çıkarır. Görüntü işleme de onlardan biridir.

Görüntü işleme yapılırken bir filtre uygulanır. Bunun sebebi ise, öznitelik çıkarımının kolaylaştırılmasıdır. Öznitelik çıkartıldığında makinenin tanıması ile görüntü işleme yapılır. Nesne tanıma, görüntü işleme, bilgisayar ile görü olaylarında derin öğrenme ağırlıklı olarak kullanılmaktadır.

Bu kitapta derin öğrenme, neural network, yapay sinir ağlarından bahsedilecektir. Bir aracın görüntü işleme kullanarak kontrol edilmesinde arabanın diğer araçlara göre kendini konumlandırması, şerit takibi veya araçların ani durmasını önceden tespit edip bunlara göre aksiyon alınması hedeflenmektedir. Görüntü işlemenin en çok kullanılan alanlarından biri olan yüz tanıma(face recognition) dır. Günümüzde bu özellik, cep telefonlarında ve bilgisayarlarda kimlik tanıma için parmak izinin yerini almıştır. Yüz tanıma, yüzün tanınmasının yanı sıra makine

öğrenmesi çalışmaları adı altında yüz üzerinden çeşitli verilerin okunmasına da olanak sağlamaktadır.

Sanal Gerçeklik (virtual reality) , makinenin tamamen sanal bir ortamda kullanıcı hareketlerinize göre aksiyon alması veya kullanıcıya bazı hisler yaşatabiliyor olmasını hedeflese de bu konuda yine makine öğrenmesi kullanılmaktadır. Sanal gerçeklik, birden fazla kişinin ortakça çalıştığı veya birlikte hareket ettiği ortamlar oluşturmak için de kullanılmaktadır. Makine öğrenmesi, oyun sektöründe de fazla kullanılan alanlardan biridir. Örnek olarak, kullanıcı bir oyun oynamakta ve bu oyun sanal bir dünyada geçiyor olmasından dolayı bilgisayar, bu sanal dünyada daha önce tasarlanmamış oyun seviyelerini makine öğrenmesi yardımıyla otomatik olarak üretebilmektedir. Oyun sektöründe yapay zekanın dünya satranç şampiyonunu yenilgiye uğratması ile bu alandaki ilk

zafer 1990'lı yıllarda satranç alanında yaşanmıştır.

Artırılmış gerçeklik (augmented reality) ise, cep telefonu kamerası bir sokağa tutulduğu zaman kullanıcıya en yakın eczaneyi saptayıp 3 boyutlu olarak yönlendirebilmesi örnek olarak verilebilir. Gelecekte artırılmış gerçeklik ile beklenen nokta, ev içerisinde herhangi bir eşyaya ihtiyaç duyulmadan mobilyaların ve duvarların artırılmış gerçeklik ile bir tıkla değişebileceği yönündedir.

Ses tanıma da makine öğrenmesi için kullanılan alanlarından biridir.

Robotik alanında kullanılan pekiştirmeli öğrenim(reinforcement learning) ile geçen kavram, robotların gerçek dünyadan tecrübeler doğrultusunda bilgiler edinerek uzun vadede kendi bilgi hazinesini geliştirmesidir. Makine öğrenme algoritmalarına kitabımızın ilerleyen kısımlarında

değineceğiz fakat şu bilinmelidir ki, makine öğrenme algoritmalarının başarılarını ölçülebilir. Dünyanın en başarısız algoritmasını bile doğru şekilde tasarlanıp pekiştirmeli öğrenime tabii tutulduğunda %0 ile başlayan süreç makine kendi kendine öğrettikçe kendini daha iyi bir duruma güncelleyerek geliştirmektedir. Robotlar da gerçek dünyadan yeni deneyimler edinerek ve öğrenerek denge kabiliyetlerini ve yürüme kabiliyetlerini bunun üzerine inşa edebilimektedirler.

Pazarlama alanı makine öğrenmesinin en çok kullanıldığı alanlardan birisi. Örnek olarak sosyal medya üzerinden bir reklamınız çıkacağı zaman, o reklam nasıl ve nerede gözükmeli ki müşteri potansiyeli olan kitle o reklama tıklama eğilimi göstersin gibi incelemeler için de makine öğrenmesi kullanılabilir. Bir başka örnek olarak, bir web sitesi ele alınsın. Web sitesine giren müş-

terilerin hangi gruplardan geldiği, hangi segmentlerden geldiği, müşteriye özel reklam (targeted marketing) içeriklerinin hazırlanması, müşteriye bir öneride bulunma(second best offer) gibi durumlarla birlikte bir kurum için müşterinin o kurumdan ayrılmadan önce ayrılabileceğini algılayabilir miyiz(customer churn analysis) gibi sorulara cevap aranmakta ve sonuç olarak, o müşteriyi kaybetmemek için mücadele edilip edilmeyeceğine karar vermek gibi ayrıntılı konular irdelenmektedir. İnsan kaynakları yönetimi (HRM) ve müşteri ilişkileri yönetimi(CRM) birbirlerine fazlasıyla benzeyen algoritmalar ile çalışmaktadırlar.

Bu alanda, makine öğrenmesi algoritmalarından söz edilecektir. Tavsiye algoritmaları(recommender algorithms), popülerliği gittikçe artan algoritmalardan birisidir.. Amazonun gelirinin %35'i , Netflix'in gelirinin %75'ine yakını tavsiye algoritmaları üzerinden sağlanmaktadır.

Müşteriye hangi ürün veya hangi film tavsiye edilirse müşteri ürün satın alabilir gibi bir işleyişi arka planda bir tavsiye algoritması yürütmektedir. Bu algoritmaların ne olduğu ve nasıl işlediğin hakkında detaylı bilgiler, kitabın ilerleyen kısımlarında bulunmaktadır.

Sağlık alanında tomografi, MR, röntgen filmlerinin ya da tahliller üzerinden teşhis ve tanı konulması, makine öğrenmesi algoritmalarının kullanımına bir örnek olarak gözlemlenebilir.

Uydu görüntüleri üzerinde çalışarak, tarım yapılması için hangi alanın uygun olacağının bulunması veya hangi bölgede kullanılması muhtemel tohumlar ile ne düzeyde tarımsal sonuçlar elde edilebileceği gibi bilgileri otomatik

olarak yapay zeka ve bilgisayar sistemleri sayesinde ulaşılabilinmektedir.

Telekom ve kredi kartı gibi alanlarda, sahtekarlık saptamak için de makine öğrenmesi çokça kullanılan alanlardan birisidir. Veri güvenliği, sistem sızmaları ve sistem tehditleri için bunları tespit edebilen otomatik algoritmaların yazılabiliyor olması makine öğrenmesi alanına örnek teşkil etmektedir.

Arama motorları bazında verebileceğimiz örneklerden biri ise, Web 1.0 statik web siteleriydi ve web 2.0 dinamik web sitelerine geçiş oldu. Bunlar ile birlikte hayatımıza giren WİKİ'ler, Black'lar ve sosyal ağlar ortaya çıktı. Arama motorları ise bu durumu takip ettiler. Web 3.0 ile hayatımıza giren semantik web ve anlam bilimsel ağlar giriş yaptı. Peki nedir semantik web?

Google'da arama motoruna Napolyon'un bisikletinin seri numarası nedir diye sorduğumuzda sonuç olarak Napolyon ile ilgili sonuçlar olarak, motosikletlerin seri numaraları ile ilgili sonuçların harmanlanıp gelmesini bekleriz çünkü şu anda Napolyon zamanında motosiklet yoktu tarzında bir sonucu Google'ın dönmesi mümkün değildir. Bunun sebebi, Google üzerinde semantik web ile ilgili bu kadar gelişmiş zamansal varlık bilimi yoktur. Web 3.0'ın yaygınlaşması ile bunları ileriki zamanlarda görmek mümkün olacaktır. Şimdilik İstanbul-Ankara arası kaç km veya Bosna'da saat kaç dediğimizde Google bu sorulara cevap verebiliyor. Bunlar gelişmiş arama çeşitleri olarak görülse de gelecekte web 3.0 ile çok daha gelişmiş sonuçlar üretecektir. Web 3.0 hayatımıza entegre olurken sonraki aşama olarak sırada web 4.0 bulunmaktadır. Web 4.0'da ana konu olarak kişisel ajanlardan bahsedilmektedir. Örnek olarak, Google bütün dünyadaki web sitelerini dolaşıp

bizim için harmanlanmış bazı sonuçlar getiriyor olması ve bu sonuçlar üzerinden cevabımızı buluyor olmamız Web 3.0'ın getirisi iken Web 4.0'da, Google yerine bütün dünyayı dolaşabilen kişisel ajanlarınız olduğunu, bütün interneti dolaşarak ve bize özel bilgiler getirdiğini düşünebiliriz. Örnek olarak buzdolabı satan bir işiniz var ve kişisel ajanınıza dediniz ki "anlık olarak dünya geneli bütün buzdolabı hareketlerini getir". Bu durumda kişisel ajanlarınız bütün buzdolabı satan yerlerdeki fiyat bilgilerini toplayarak bir yıl boyunca herhangi bir değişiklik olduğunda size haber vermesi gösterilebilir. Bununla ilgili uygulamalar günümüzde de bulunmaktadır. Günümüzde bu konu üzerine çeşitli siber savaşlar da görülebilmektedir. Birileri bu bilgiye ulaşmaya çalışıyor fakat bu duruma karşın birileri de bu bilgileri gizlemeye çalışıyor. Netice olarak bu konu ile ilgili makine öğrenmesi, yapay zeka ve veri bilimi uygulamalarının hayatımıza yavaş yavaş girdiği bilin-

mekle beraber bu kitapta yer alan bilgilerin bu alanlar temel taşı niteliği göstermektedir.

Önemli konulardan birisi endüstri 4.0'dır. Endüstri 4.0'dan bahsederken genelde aydınlatması olmayan fabrikalar ile özetleme yapmak mümkündür. Endüstri 4.0 temelinde, fabrikalarda insan gücüne ihtiyaç duyulmadan, kendi kendine yeterli olması yatar. Bir problem çıktığında kendi kendine sorunu bulup düzeltebilen fabrikalardan bahsedilmektedir. Tamamen otomatize olabilen, bilgisayarlar ve yapay zekalar tarafından kontrol edilebilen ve yazılımlarımızın vermiş olduğu talimatlara göre istediğimiz üretimi yapabilen fabrikalardan bahsedilmektedir. Endüstri 1.0 ilk buhar makineleri üretimi, Endüstri 2.0 endüstri devrimi, Endüstri 3.0 CNC makinelerini temel almakla birlikte Endüstri 4.0 neticesinde şu an ulaştığımız noktada, sensörler gibi birçok alandan veri toplayıp oluşturulan ve bu büyük veriler üzerinden belirli analitikler yapılarak, çeşitli yapay

zeka uygulamaları çalıştırılması hedeflenmektedir.

Gelecekte bu olayların daha çok hayatımıza girdiğini gözlemleyeceğiz ve bu gelecekten bahsetmek gerekirsek örnek olarak üç boyutlu yazıcı teknolojileri gösterilebilmektedir. Bu teknolojiye örnek olarak, kullanıcıların evinde çatal bittiği zamanda veya istenilen başka bir model ürünü basımı talep edildiğinde bu teknoloji sayesinde kolayca üretim sağlanabilen bir dünyadan bahsedilebilir. Endüstri 4.0 ile ilgili makine öğrenmesi algoritmalarının yoğun olarak kullanıldığı karar vermek süreçlerinin büyük miktarda makine öğrenmesi algortimaları ile sağlandığı ve veri bilimi çalışmaları ile desteklendiği bilinmektedir. Bir diğer uygulama olan ve günlük

yaşama büyük katkılar sağlayan IoT(internet of things) nesnelerin interneti teknolojisidir. Günlük hayattan örnek verirsek akıllı buzdolabı, içinde az kalan ürünleri saptayarak otomatik olarak sipariş verebilmektedir. Türkiye'de şu an da bulunmasa da Amerika'da SmartGrind uygulaması vardır. SmartGrind, akıllı elektrik dağıtım şebekesidir. Örnek vermek gerekirse, buzdolapları 10-15 dakika çalışma ve 30 dakika durmak gibi belli periyotlar ile çalışmaktadır. Bir elektrik dağıtım şebekesinde bütün buzdolaplarının aynı anda çalıştığını düşünülür ise bu durum, şebeke üzerinde büyük bir yüklenme oluşturacaktır ve hepsinin aynı anda durması ise sistemde büyük bir boşluğa sebep olacaktır. Bu olay elektrik dağıtımı için ciddi bir problem oluşturmaktadır. Eğer bu sistem düzene sokulursa, elektrik dağıtımı çok daha düzgün bir şekilde sağlanabilir. Elektrik dağıtım firması, elektrik dağıtım şebekesi aracılığı ile buzdolaplarına müdahale edilmesine imkan vermektedir. Bu

uygulamanın kullanıldığı alanlarda faturada ufak bir indirim yapılarak bu kullanım şekli kullanıcıya cazip hale getirilmeye çalışılmaktadır. Bu sisteme dahil olunduğu zaman buzdolabının çalışma frekansı şebeke merkezinden belirlenmektedir. Bu şekilde sistemdeki dağıtımı ve sistemde yaşanabilecek problem ve aksaklıkları azaltmak amaçlanmaktadır. Buzdolapların birbirleri ile senkronize olarak konuşmaları ve konuşarak birbirlerine göre sırayla çalışıyor olmaları nesnelerin internetine örnektir. Nesnelerin ve eşyaların birbirleri ile konuştukları bir internet devrine geçildi ve bu durum gelecekte daha da yaygın hale gelecektir.

Sonuç olarak, makine öğrenmesi çok farklı alanlarda kullanılabilir. Verinin olduğu her yerde makine öğrenmesi, veri bilimi, yapay zeka ile ilgili bir iş olduğu söylenebilir. Bu kitap boyunca makine öğrenmesi algoritmalarını

öğreneceğiz. Gerçek hayattan problemleri inceleyip, çözüm gerçekleştireceğiz.Online ve dijital dünya her geçen gün gelişiyor ve buna bağlı olarak veriler de gelişiyor. Gelecekte, günümüzdekinden daha büyük veri hacimlerinin hayatımıza girecek, daha fazla işlem kapasiteleri artacak ve insanlığın başa çıkamayacağı kadar büyük veriye doğru yol alacağız. Bu, veriler ile insanlığın değil, makinelerin başa çıkması gerektiğini ön plana çıkarmaktadır.

Birkaç veri örneğine bakacak olursak geçen yıl Amerika'da evlenen 8 çiften birisi online tanışmıştır. Kültür çizgilerimiz günün şartlarına göre değişiyor ve makinelerin daha fazla etkisi altına giriyoruz. Örnek olarak, hepimizin elinde cep telefonları , masamızda ise bilgisayarlar var. Bir cep telefonu kullanarak sürekli yaşantımızdan kesitler paylaşıyoruz. Cep telefonumuz ile

olan etkileşimimiz, insanlar ile etkileşimlerimizden çok daha yüksek düzeyde bulunmakta. Bu duruma istinaden makineler, bizim için insanlardan daha yakın hale gelmiş durumda. Geçen yıl yazılı olarak üretilen doküman miktarı, yazının icadından sonraki 2000 yıl boyunca oluşan toplam üretimden daha fazla olmuştu. İnsanlık yazıyı icat ettikten sonra yazıyor, bunları bir şekilde okuyor, depoluyor, kopyalıyor ve transfer ediyor. Bu verilerin toplam miktarından daha fazlasını sadece bir yıl içerisinde üretiyoruz. Mesajlarımız, e-maillerimiz, gazete haberleri bunlara ayrıca dahil olmaktadır. Üretilen veri durmadan artıyor. Facebook aktif kullanıcı sayısı aylık olarak 1.7 milyara ulaştı. Üniversite hayatımız, eğitim hayatımız değişiyor. Tercih döneminde tüm öğrenciler en gözde meslekler nedir diye araştırıyor, mezun olduğunda ise bu meslekler yerini başka mesleklere bırakıyor. Hızla değişen ve çokta öngörülemeyen bir dünyaya doğru yol alıyoruz. Bunun çözümü için bilin-

meyen meslekler için insanları yetiştirmemiz ve henüz icat edilmemiş teknolojiler için uzmanlar yetiştirmemiz gerekiyor. Güncel örneklerden Youtube, çok büyük bir veri kaynağına sahip.

Aktif olarak 1.3 milyar kullanıcısı bulunuyor ve günlük 30 milyon ziyaret alıyor. Son yıllarda Youtube'un izlenme süreleri %60 oranında arttı ve bu oran giderek artmaya devam etmekte. İnternette "Glassdoor" adında bir site mevcut ve bu sitenin internette ortaya çıkışı ile firmalar, şirketler hakkında içerden bilgi almak mümkün hale geldi. Herkes gizli olarak siteye nerede çalıştığını, aldığı maaşı, firma ile ilgili yaşadığı deneyimleri, iş görüşmelerinde yaşadığı deneyimleri paylaşmakta. Şu an sitede iş ilanları ve çeşitli araştırmalara da yer verilmektedir. Bu sitede 2018 yılında yapılan en gözde meslekler adlı bir araştırmada ilk sırada veri bilimci (data scientist) yer almakta. Sadece Amerika pazarında

4.524 açık pozisyon bulunmakta. Bu durum nedenlerinden biri ise veri bilimine ve özünde makine öğrenmesine bağlı çok sayıda meslek türü bulunmasıdır. Amerika'da neredeyse bütün üniversitelerde Veri Bilimi(Data Science) bölümü bulunmakta ve neredeyse herkesin veri ile alakalı bir işi bulunmakta. Bu duruma başlı olarak, işlerin nasıl daha kaliteli yapılacağı üzerine bir iş gücü pazarı ortaya çıktı ve bu alanda insan yetiştirmek öncelik haline geldi. Hayatımız değişiyor, veri bilimi anlamında, eğitim anlamında ve kültür anlamında değişimlerden bahsedebilmek mümkündür. Veri biliminin özündeki makine öğrenmesini bu kadar önemli kılan şeylerden biri de başa çıkabileceğimiz verinin bir limiti olması ve verinin de sürekli artıyor olmasıdır. Yazının icadından 2005 yılına kadar insanların toplam ürettiği 130 Exabyte'lık bir veridir. Örnek vermek gerekirse, insan genomu projesi 1 GB boyutunda bir veridir. Halbuki bir insanın doğduğu andan 80 yaşına kadar hay-

atının her saniyesini yüksek çözünürlüklü bir kamera ile hd kalitesinde çekebilirsek bu veri ancak 1TB boyutunda bir veri yapıyor, fakat bir jet motoru 7 saat içerisinde 30 TB veri üretebiliyor. Büyük veri için başka bir örnek vermek gerekirse, Amazon yağmur ormanlarında tahminen 700 milyar ağaç bulunmakta ve hepsini kağıda çevirip üzerine yazı yazsak bile bu bilgi 2 TB veri olarak bilgisayarlarda tutulabilir. Bu örneklere bakılırsa verinin kapasitesi, biz insanlar olarak hayal edebileceğimizin çok üstüne çıkmış durumda bulunmaktadır. Bir zaman çizgisi örnek vermek gerekirse:

-2005'e kadar toplam 130 EB veri üreten insanlık,

-2010'a kadar 1299 EB veri üretti.

-2015'e kadar 7900 EB veri üretti.

-2020'de tahminen 40900 EB veri üreteceğiz.

Türkiye ortalamasına bakarsak bir insan dakikada ortalama 200 kelime okuyor. Bu durumda yaklaşık

saatte 72KB veri elde edilecek demektir. İlk çıkan bilgisayardan komodor 64, Apollo programı kapsamında aya götürüp getiren navigasyon bilgisayarı gibi bilgisayar çeşitlerinden ikisinin ortak özelliği, ikisinin de

raminin 64 KB olması. Bir anda hafızasına alıp işlemek için tutabileceği veri 64 kb'lık bir veridir. Bunu

boyut olarak karşılaştıracak olursak bu, bir insanın bir saatte okuyabileceği bir veriye denktir. Bir cep

telefonu ortalama 3 GB ram ile fabrikadan çıkmaktadır. Her insanın mevcut bir işleme kapasitesi var fakat bu durum organik varlıklar için sınırlıdır. Çeşitli yöntemler ile işleme kapasitesini artırabilmek

mümkündür. Bilgisayarın da sınırlı bir işlem kapasitesi vardır ve Bilgisayarında ram kapasitesinin zamanla arttığını görmekteyiz . Fakat, bunların yanında verinin de hız kesmeden arttığını göz ardı etmemeliyiz.

Verinin artış hızı insanlarında, makinelerinden artış hızından çok daha fazla durumdadır. Bu kısımda yapay zeka devreye giriyor. Yapay zeka, problemlerimizi insan gibi çözmemize yarayan çözümler ortaya koymaktadır. İnsanlar, zaman harcamak istemediğimiz problemlerimizi yapay

zekaya öğreterek onun bizim yerimizi çözmesini sağlıyoruz, ama onunda belli bir kapasitesi var ve bu kapasite belli bir hızla artmakla beraber belli bir limite sahip olmaktadır. Bunun daha ötesine çıkmak istediğimizde gene yapay zekanın bir alt çalışma alanım olarak görebileceğimiz makine öğrenmesi devreye sokulmaktadır. Makine öğrenmesi, her seferinde insanın yapay zeka gibi algoritmasını çıkardığı, öğrettiği, verileri üzerinde test yaptığı, iyileştirdiği bir sürecin ötesine geçmiş durumdadır. Makine öğrenmesi, şu an da makinelerin veri üzerinden, dış dünyadan öğrenim artırımlı öğrenim (reinforced learning) gibi düşünürseniz kendilerini geliştirdikleri ve bu gelişme sürecinde kendilerini yargılayarak kendilerini eğittikleri bir dünyaya doğru insanların kırınımı anlamına geliyor. Bu kadar veri ile baş etmek için makine öğrenmesi kaçınılmaz bir son olarak görünmektedir. İnsan ırkı olarak hem nüfusumuz limitli, hem de yapabileceğimiz işler limitli ama problemlerimiz ve ürettiğimiz veri

bunun çok çok üzerinde bir sayıya ulaşmaktadır . Makine öğrenmesi, verinin olduğu her yerde bizim için iş vardır ve veri artık her yerde ve giderek çok daha büyük kapasitelere ve çok daha büyük problemlere dönüşmektedir. Bahsedilen bu bilgiler ışığında Makine öğrenmesi, geleceğin önemli mesleklerinde ve teknolojilerinden birisi olması kaçınılmazdır.

Ders 3: Python ve Anaconda'nın kurulması

Bu bölümde Anaconda ve Python`nın nasıl indirileceği ve bilgisayara kurulumu anlatılacaktır. İlk olarak Anaconda kuracağız. Bunun için Anaconda.com web sitesine giriyoruz.Sitede görüldüğü üzere “what is anaconda, products, support, resources ve about” kısımları bulunmakta. Bir sonraki adım olarak “downloads” yazan kısma tıklıyoruz.

Önümüze gelen ekrandan işletim sistemimizi seçiyoruz.

Ardından Python 3.6 version ve 2.7 version seçeneği ile karşılaşıyoruz. Bu iki Python sürümü arasında ciddi bir fark bulunmakta. Bu kitap kapsamında biz 3.6 versiyonu ile ilgileneceğiz. Bu aşamadaki önemli olan kısım, Python bilgisayarınızda yüklü değilse kurmanız.

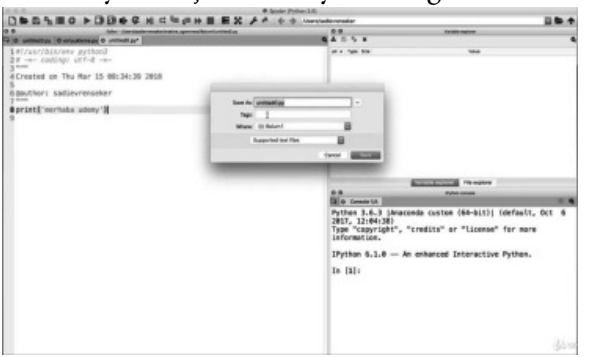
Kurulduktan sonra açılan navigator sayfasında üstünde birçok araç görünmekte. Anaconda, o

araçlara ulaşmamız için bir menüdür.Burada; R, Python ve Orange gibi birçok konsol bulunuyor. Bu platformda projeler yapılabilmektedir. Bu kitap kapmasında Spyder kullanacağız. Spyder, Python dilinde kod yazmak için kullanılan bir platformdur. Bu kodlama ortamının bize sağladığı başlıca avantajlar, birden fazla dosyadan oluşan projelere olanak sağlaması ve veri kütüphanelerine erişim imkânı sağlıyor olmasıdır.

Spyder'i açtığımız da bizleri bu ekran karşılamakta. Sağ taraf üst kısımda dosya explorer mevcut olmakla beraber sağ taraf alt kısımda ise konsol mevcuttur. Sol tarafa yazdığımız kodumuz çalıştıkça sonuçları sağ alt tarafta konsolumuzdan görebilmekteyiz.

"merhaba udemy" kodumuzu yazıp ardından yeşil "play" butonu yani "run" tuşuna tıkladığımızda önümüze aşağıdaki resimdeki sayfa gelecektir.

Bu kısımda bize dosyayı nasıl kaydetmek istediğimizi sorulacaktır. “İlk.py” diye kaydedip çalıştırdığımız zaman konsola “merhaba udemy” de yazılmış haliyle görülecektir.



“Print” komutu ekrana verilen string(harf dizini)`i yazdırmamıza yaramakta. String’ler ise tek tırnak veya çift tırnak kullanılarak yazılabilmektedir. Üç adet tırnak işareti ile başlatılarak ve üç adet tırnak işareti ile sonlandırarak yorum(comment) yazılabilmektedir. Bir başka seçenek ise, `#` kullanılarak yorum satırı oluşturmaktır. Yorum olarak koda eklediğimiz satırlar konsola dahil edilmemektedir.

Değişken tanımlamak için herhangi bir tanım cümlesine ihtiyacımız bulunmamaktadır. Basit bir şekilde değişkeni atamak mümkündür.

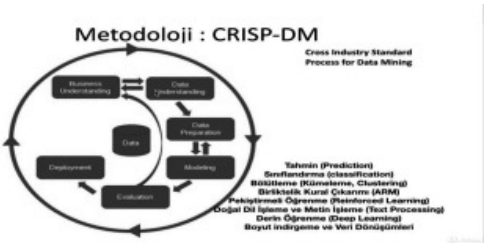
Liste oluşturmak için köşeli parantez kullanılır.

X değişkenine tanımladığımız değeri konsola yazdırmak için "print(x)" ibaresini kullanabili-

riz. Sağ alt kısımda gördüğümüz üzere "10" olarak ekrana yazdırılmıştır.

Ders 4: Derslerle ilgili önemli notlar ve Derslerin İzleyeceği Sıra

Bu kısım ile birlikte artık içeriğe giriyoruz. İçerik ile ilgili takip edeceğimiz bir metodoloji var. Bu metodoloji veri bilimi projesi veya makine öğrenmesi projesinin nasıl yönetileceği üzerine olacak. Bir projeye nereden başlanır, nerede biter, hangi aşamalar izlenir gibi sorulara bu kısımda cevap verilecektir. Bütün bu sorular için geliştirilmiş birçok metodoloji var. Kitabın hangi sıra ile konuları ele alacağını öğrenmek için CRISP-DM metodolojisinden yararlanılacaktır.



CRISP-DM metodolojisi problemin tanınması ve ne yapılmak istendiğinin anlaşılması ile başlar. Ardından veriyi tanıma ve veri hazırlama işlemlerine geçilir. Problem ve verinin var olacağının kabul ederek makine öğrenmesine başlıyoruz. 'Hangi problemler', 'nasıl yaklaşılır', 'veri ile ilgili özellikler nelerdir' ve 'hangi tip problemler vardır' bunlar veri biliminin konusudur. Makine öğrenmesinde ilk olarak veri ön işleme ve veri hazırlama aşamasından başlıyoruz. Sonrasında ise, modelleme ve değerlendirme kısmı ile makine öğrenmesine devam edeceğiz. Veri ön işleme kısmı kitabımızın ilk kısmını oluşturacaktır.

Kitapta değineceğimiz konuları buradan toplu olarak görebilirsiniz.

"www.bilkav.com/makine-ogrenmesi-egitimi/" adlı web sitemizden kursumuz için gerekli olan kaynakları bulmak mümkündür. Bir başka seçenek olarak www.bilkav.com [1]adresinden eğitimlerimiz bölümünden de bu sayfaya erişim sağlayabilirsiniz.

Ders 5: Verinin Yüklenmesi

Bölüm 2 ders 5'de yer alan veriler.csv'ye tıklayarak kaynaklara ulaşılabilir.

1. http://www.bilkav.com/

Dosyayı indirmek için sağ fare tuşunu tıklayarak "download linked file" ile bilgisayarınıza indirebilirsiniz.

İlk uygulamamız için gerekli olan dosya budur ve bilgisayarınıza indirmeniz gerekmektedir.

Ders 6: Kütüphanelerin Yüklenmesi: NumPY, Pandas ve MathPlot Yüklenmesi

Bu kısımda python ortamına kütüphanelerin yüklenmesini inceleyeceğiz.

İmport komutu bizim kütüphaneleri eklememize yarıyor. “import pandas as pd” olarak yazdığımız zaman pandas kütüphanesine pd olarak bir isim verilmektedir. Bu isimler, kütüphanelerdeki fonksiyonlara erişmek için

kullanılacaktır. Pandas kütüphanesi, veri çerçeveleri oluşturmak ve verileri düzgün bir şekilde muhafaza etmek için kullanılan bir kütüphanedir. “pd” dediğimiz zaman “insert” edilmiş olan pandas kütüphanesinin altındaki fonksiyonlara ve özelliklere erişmekteyiz. “pd” kısmının çalışabilmesi için pandasın doğru bir şekilde import(alınması) edilmesi gerekiyor.

Numpy kütüphanesini ise np olarak import etmekteyiz. Numpy kütüphanesi, büyük sayılar ve hesaplama işlemleri için kullanılacaktır.

Matplotlib.pyplot kütüphanesi ise, çizim için kullanacağımız kütüphanedir.

Python kodunun okunabilirliği açısından import komutlarını konsolun yukarı kısımlarına koymak daha kolaylık sağlayacaktır.

Derslerin takibinin daha iyi yapılabilmesi için "comment(yorum)" olarak kodun en üst kısmına eklemeler yapılmalıdır. Örnek olarak, "ders 6: kütüphanelerin yüklenmesi" gibi satılardır. Eğer herhangi bir hata alırsanız kodlarınızda yazımdan kaynaklanıp, kaynaklanmadığını kontrol etmek amaçlı www.bilkav.com[2] üzerinden kodlara ulaşabilirsiniz. Dosyalar genel olarak zipli olarak indirip, açabilirsiniz.

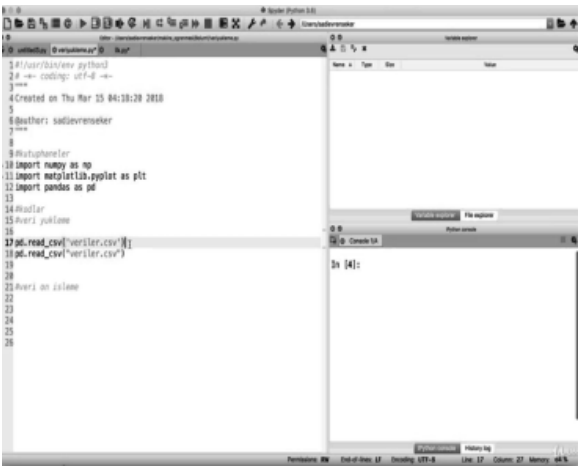
Ders 7 : Verinin içeri alınması (data import)

2. http://www.bilkav.com/

Bu kısımda ilk olarak Pandas kütüphanemizi kullanacağız. Bunun için "pd." ile kullanabileceğimiz özelliklerimiz gelmektedir. Bu aşamada "read(okuma)" özelliği kullanılacaktır. Okumak istediğimiz dosya turu csv olduğundan dolayı "read_csv" seçiyoruz. Csv ise "comma separeted value" yani verilerin birbirleri arasında virgül olması durumu demektir.

Pandas kütüphanemizi çağırdıktan ve read komutunu verdikten sonra tırnak içinde bir string

tanımlıyoruz. Bu tırnak, tek tırnak ile de olabilir çift tırnak ile de olabilir. Python'da çift tırnak veya tek tırnak için de ikisi de string ifade eder diyebiliriz. Tırnağın içine okumak istediğimiz dosyamızın ismini yazmamız gerekmektedir. Dosyamızın ismi "veriler.csv" dosyasıdır.



Yüklenmiş olan bilgiyi, veriler adında bir değişkene atıyoruz. Yazdığımız kod ile alakalı bir problem mevcuttur ve bu problem ise veriler.csv dosyamızın bilgisayardaki konumu ile alakalıdır. Python'nun dosyayı kaydettiğimiz yeri bulması olası değildir. Python'da kodu çalıştırdığımız yer ile dosyanın yüklü olduğu yer aynı olmak zorun-

dadır. Bundan dolayı bu, bizi çalışma dizini kavramına götürmektedir.

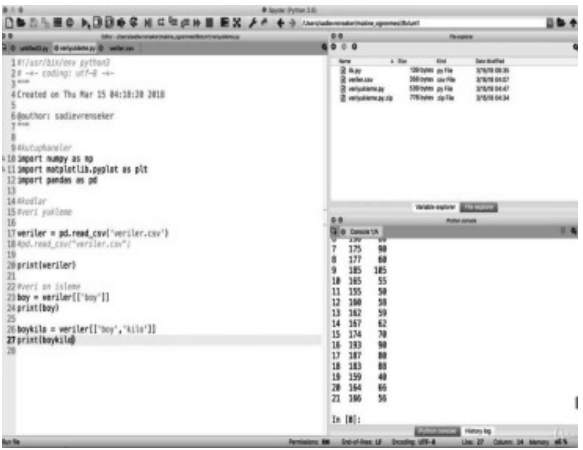
Sağ üst köşede mevcut olan yerde “variable Explorer” ve “file Explorer” mevcut. File explorer’a tıklarsak onun altında makine öğrenmesi adında bir dizin olduğunu göreceğiz. Burada Csv dosyası da bölüm 1 in içine kayıtlı bulunmaktadır . “Save/kaydet” seçeneğini seçerek bu dosyayı veri yukleme.py olarak bölüm 1’in içine kaydedebiliriz. Buradaki asıl amacımız, Python

kodu ile csv dosyalarını aynı dizinde olmasını sağlamaktır. Başka bir yol olarak "absolute path" vermek de kullanılabilir, fakat Kolay olması açısından relative path kullanılmaktadır. Daha sonrasında ise dosyayı okuduktan sonra üzerinde işlem yapabilmek olası hale gelecektir.

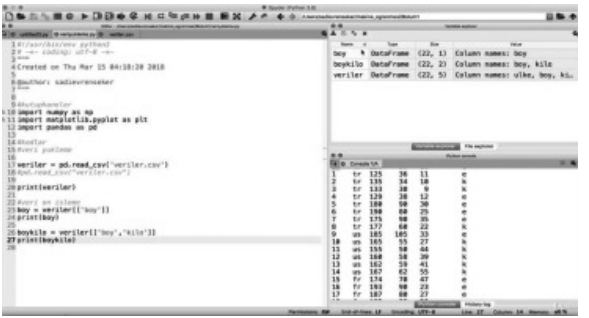
File explorer'da "veriler.csv" dosyasına tıkladığımızda dosyayı açabilmek mümkündür. Bu bir text dosyası olduğu için Python bunu sol tarafta açıyor. Bu durumda, Pandas kütüphanesinin getirdiği avantajları kullanıyoruz.

Verilerin herhangi bir değerine erişmek için 23. Satırda olan kodu yazıyoruz. Çift köşeli parantez açmamız gerekiyor çünkü bu kısımda "list of list" özelliğini kullanıyoruz. Sonraki adım olarak "Print" komutunu kullanarak atadığımız boy verilerin kolonunu konsola yazdırıyoruz.


ulke,boy,kilo,yas,cinsiyet
tr,138,38,38,e
tr,125,36,11,e
tr,135,34,38,k
tr,133,38,9,k
tr,129,38,12,e
tr,188,90,38,e
tr,198,88,25,e
tr,175,98,35,e
tr,177,68,22,k
us,185,105,33,e
us,165,55,27,k
us,155,58,44,k
us,168,58,39,k
us,162,58,41,k
us,167,62,55,k
fr,174,78,47,e
fr,193,98,23,e
fr,187,88,27,e
fr,183,88,28,e
fr,159,48,29,k
fr,164,66,32,k
fr,166,56,42,k
In [4]:



Boy-kilo değişken ismimiz olarak belirleyip veriden bu sefer boy ve kilo kolonlarını okuyoruz. Sağ alt karede görüldüğü üzere print komutu ile okuduğumuz verileri konsola yazdırıyoruz.



Sağ üst kısımda file explorer ile dosyaları görüntülüyoruz. Variable explorer'a geçiş yapılırsa

tanımlanmış olunan her değişken görülebilinmektedir. "Type" kısmında değişkenimizin çeşidini görebilimekteyiz. "size" kısmında ise, içinde bulunulan değişkenin içinde ne kadar veri barındırdığını görüntüleyebiliyoruz. "value" kısmında ise, değişkenimizin içindeki değeri görüntüleyebiliyoruz.

Verileri görmek istesek çift tıklayabiliriz. Pandas kütüphanesi sayesinde data frame olarak açabiliyoruz. Spider ide 'de python kodunu yazarken kullandığınız bazı artı özellikler mevcut. Kodu herhangi bir ortamda yazıp, herhangi bir python editöründe yazıp kullanılabilir. Dosyayı yanlış dizine kaydettiyseniz save copy as veya save as şeklinde dosyanın dizinini değiştirebilirsiniz. Önemli olan verile ile python kodunun aynı dizinde olması önemli bir kısım.

Ders 8: Python: Nesne Yönelimli programlama

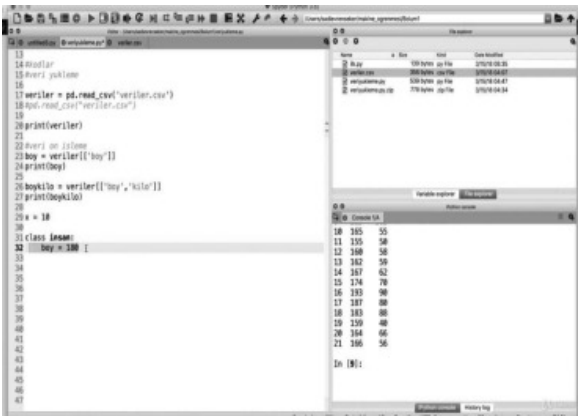
Bu bölümde Python dilini araç olarak kullandığımız için bu kısımda ona yer vereceğiz. Python ile iligili detaylı bir eğitim ihtiyacınız varsa bilkav.com adresinde eğitimler seçeneğinden Python eğitimlerine ulaşabilirsiniz.

Bu kitap boyunca Python dili ile ilgili herhangi bir başlangıç eğitimi sunulmayacaktır. Makine öğrenmesinde biz Python'u sadece bir araç olarak kullanacağız. Bu kısıma kadar kullandığımız kodlar her biri şablon şeklindelerdi. Her biri belirli yerlerini değiştirerek yapılabilecek ve yüksek düzeyde kodlama bilgisi gerektirmemekteydi. Bilinmesi gereken temel bazı durumlar vardır. Programın yukarıdan aşağıya doğru bir akışı olduğu, değişkenin ne olduğu, bir değişkenin aslında içinde geçici olarak veri barındırdığı ve bazı fonksiyonların veya kütüphanelerin hangi işe yaradığı ve nasıl kullanıldığı gibi durumlar örnek gösterilebilir. Dersi anlatırken kullandığımız kodları kopyala-yapıştır yaparak kullanabilirsiniz veya inceleyerek yapıları öğrenip kendiniz de yazabilirsiniz.

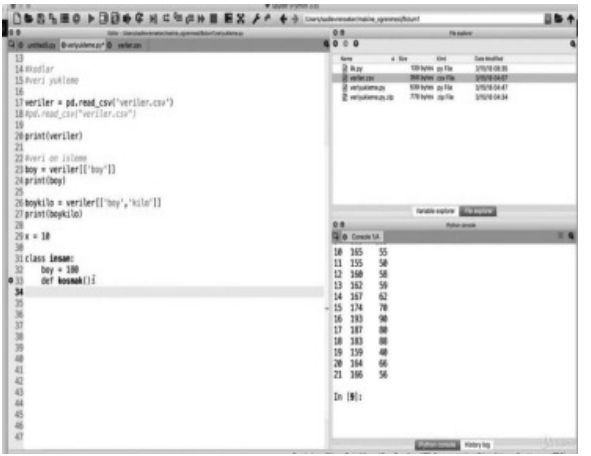
Kitabın bu kısmında nesne yönelimli programlamanın felsefesinden bahsedilecektir. Nesne yönelimli programlama, bu kitap boyunca işimize yarayacaktır. Bundan dolayı bu kısımda bu kavrama değineceğiz.

Class ve object kavramı:

Class komutunu kullanarak herhangi bir tanım yapmamız mümkündür. İki nokta, Python'da bir blogun başladığını göstermektedir. Tab kullanarak içerden başlayıp yazdığımız satırlar bu class(sınıf)'ın içinde yer alacaktır. Bu kısımdan sonra herhangi bir şey tanımlayabiliriz.

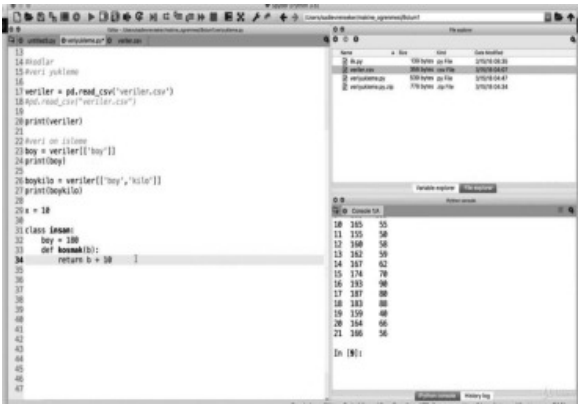


Örnek olarak insan isminde bir classımız olacaktır. Bu classın içinde boy isminde bir özellik olacaktır.

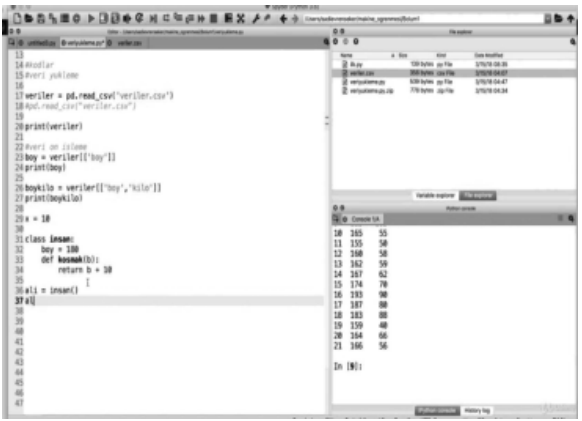


Bu aşamada definition(def) yani fonksiyon tanımlayabilirsiniz. İsimlendirme mantığı olarak insan koşabileceği için "koşmak" adlı bir fonksiyon tanımlamak mantıklı bir seçim olacaktır. Fonksiyonu tanımladıktan sonra parantez ve iki nokta koyulma kuralları unutulmamalıdır. İki nokta üst üsteden sonra kodun tab'lenmiş haliyle satıra başlıyoruz. Bu demek

oluyor ki bu fonksiyonun içinde hangi işlemleri yapılacağını söylemeye başlıyoruz.

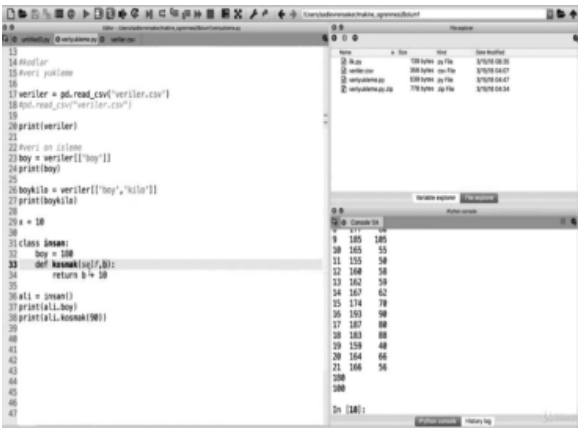


Öncelikle fonksiyonumuzun içine “b” adlı bir değişken atayalım. Fonksiyonumuzdan verdiğimiz b değerinin 10 eklenmiş halini çıktı olarak almak istiyoruz. Bu sebeple b+10’u fonksiyonumuzdan “return” ettirebiliriz. Class insan artık bir class haline geldi.



Kolonun baş kısmına tekrar gelip “ali = insan()” ifadesini kullanırsak bu ali insan tipinde bir obje haline gelmiş oluyor. Bu insan herhangi bir şey tanımlamamız için mümkün kılınmaktadır.

Örneğin, “ali.boy” ile Ali’nin boy değişkenine ulaşmak mümkündür. Print komutunu kullanıp ali.boy dediğimizde boy değerine ulaşabiliriz. Kullandığımız kodu çalıştırdığımızda ise çıktımız, 180 olarak konsola basılmaktadır.



Python için önemli olan almış olduğu parametrelerdir. Bir classın içinde bir metot tanımlandığı zaman self isminde özel bir parametre alınıyor ve bunun yanında ilave parametreleri verebiliyor. Bunun amacı, Python'un kendisine geçirilen parametreyi ilave olarak alıyor olmasıdır. Gördüğünüz üzere aldığı değişken b ve onu "90" olarak tanımladık. b = 90 oldu ve bize çıktı olarak konsolda "100"verdi.

Bilinmesi gereken temel özellikler bir class'ın nasıl tanımlandığı, bu class'dan objenin tanımlanma süreci, bir classın içinde değişkenlerin nasıl tanımlandığı ve classın içinde metodun nasıl tanımlandığıdır.

Metotların mantığı, matematikten de bildiğimiz fonksiyonların mantığı ile aynıdır. Bu metotlar almış olduğu parametreye herhangi bir işlem yaparak geri döndürebilinmektedir. Koşmak fonksiyonunu "y = f(x)" olarak tanımlarsak sonuç döndürdüğümüz işlemi "f(x)=x+10" olarak tanımlayabiliriz. Lise matematiğinden bildiğiniz fonksiyon ile de ifade edebiliriz.

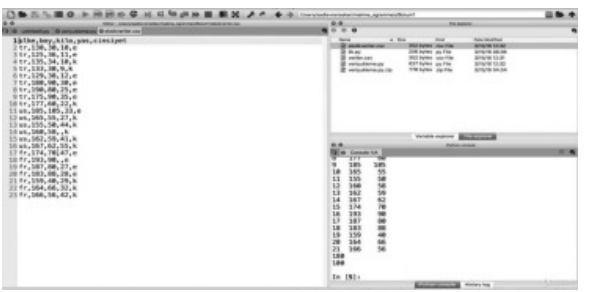
Python'la ilgili bilinmesi gereken bir başka konu ise listelerdir. Listeler, lisede gördüğünüz diziler konusuna benzerlik göstermektedir. Köşeli

parantezler ile listeleri oluşturabiliyoruz. Python'da listelerin elemanlarına erişmek, kopyalamak, taşımak mümkün. Bu işlemler için metotlar mevcuttur. Yeri geldikçe bunları kullanacağız. Eğer size çok karışık geldiyse bunun sadece bir giriş kısmı olduğunu bilmeniz yararınıza olacaktır. İleri bölümlerde bu metotları kullanacağız ve uygulama üzerinde daha anlaşılabilir hale gelmeye başlayacaklardır. Makine öğrenmesinde kendi kodunu yazmayı düşünenler için önerim, Python eğitim serisini tamamlayıp kitaba kaldıkları yerden devam etmeleridir. Python'un güzel yanı ise kolay öğrenilebilen bir dil ve esnek bir yapısının olmasıdır.

Ders 9: Eksik Veriler

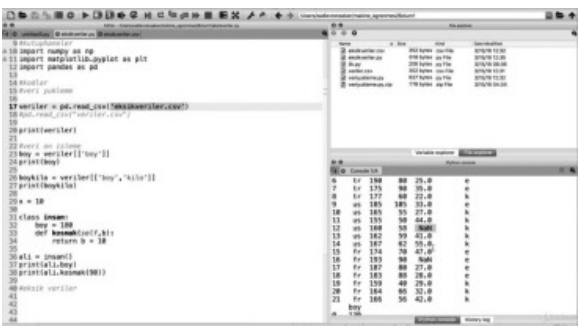
Kitabın bu kısmında bilkav.com adresinden "ders 9 Eksik Veriler" kısmından csv dosyasını indirin. Bulunduğumuz yer "Veri ön işleme" aşamasındır. Bu kısımda verileri, makine öğrenme algoritmalarına hazırlamaya çalışıyoruz. Bunun nedeni ise, bazı makine öğrenme algoritmaları eksik veri ile çalışamamaktadır. Bundan dolayı algoritmayı kullanmadan önce bu problemi çözmemiz gereklidir ve veriyi işlemlere hazırlamamız gerekmektedir. Eksik verilerin çözülebilmesi için çok geniş bir literatür mevcuttur. Günümüzde eksik verilerin çözülmesi ancak makine öğrenmesi algoritamaları ile yapılabilmektedir. Bu detayları kitabın ilerleyen bölümlerinde inceleyeceğiz.

Eksik veriler.csv adlı dosyayı indirdiniz. İçine göz atmak gerekirse:



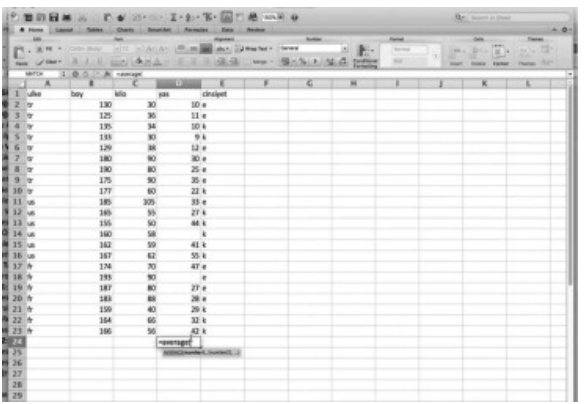
Bazı verilerin eksik olduğunu gözlemliyoruz. Veri yükleme adlı dosyanın adını değiştiriyoruz. Kodunuzda bir hata olmasına karşın siteden “eksikveriler.py” adlı dosyadan kontrol edebilirsiniz.

Üst kısımda yer alan " veriler.csv" dosyasını okuduğumuz kod satırını eksikveriler.csv şeklinde çevirdiğimizde eksik veriler dosyamızı okumuş oluyoruz. Adım adım bir şablon oluşturmaya doğru gidiyoruz.



Kodu çalıştırdığımız zaman eksik verilerin yerinde "not available (NaN)(uygun değildir)" yazısını görüyoruz. Çözüm yolu olarak, eksik verilerle elde ettiğimiz belirsizlik probleminin nasıl çözüleceğini inceleyelim. Üstünde uğraştığımız veri tipi sayısal bir veridir ve sayısal veriler farklı şekillerde çözümlenmek-

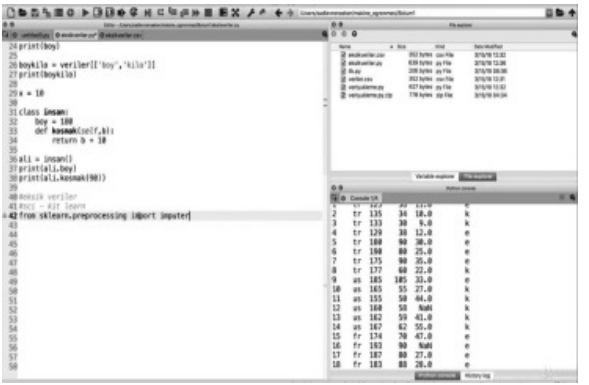
tedir. Nominal veriler, yani sayısal olmayan veriler içinde farklı yöntemler mevcuttur. Sayısal veriler için en kolay kullanılabilecek yöntemlerden birisi, her kolonun ortalamasını alarak bu ortalama değerin eksik değerlere yazılmasıdır.



	A	B	C	D	E
1	ülke	boy	kilo	yas	cinsiyet
2	tr		130	30	10 e
3	tr		125	36	11 e
4	tr		135	34	10 k
5	tr		133	30	9 k
6	tr		129	38	12 e
7	tr		180	90	80 e
8	tr		190	80	25 e
9	tr		175	90	35 e
10	tr		177	60	22 k
11	us		185	105	33 e
12	us		165	55	27 k
13	us		155	50	44 k
14	us		160	58	k
15	us		162	59	41 k
16	us		167	62	55 k
17	fr		174	70	47 e
18	fr		193	90	e
19	fr		187	80	27 e
20	fr		183	88	28 e
21	fr		159	40	29 k
22	fr		164	66	32 k
23	fr		166	56	42 k
24				=average(	
25					
26					
27					
28					
29					

Eksik veriler.csv dosyamızı açtığımızda verilerin eksik olduğu kolunun ortalamasını average(ortalama) fonksiyonunu kullanarak alıyoruz. Ek-

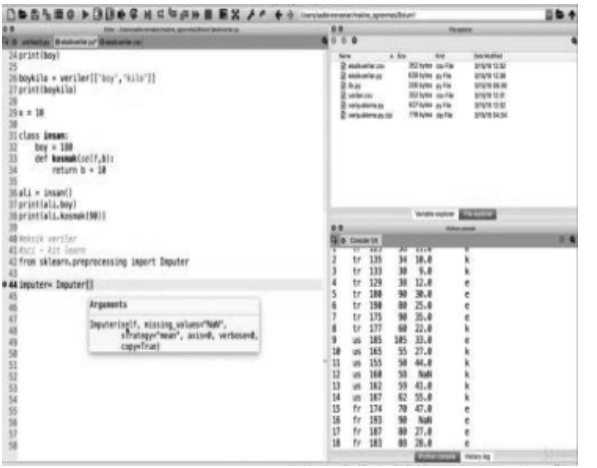
sik veriler dışında kalan verilerin ortalaması 28.45'i elde ediyoruz. Ortalama veriyi eksik kısımlara yazdırma işini Python'u kullanarak gerçekleştireceğiz.

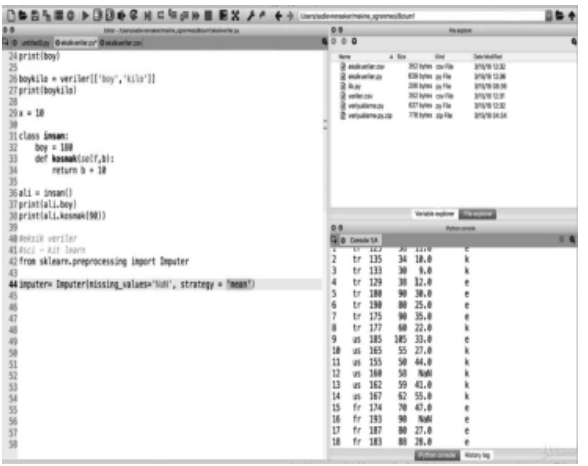


Veri kümesine erişmek için data seti ile oynayacağız. Bu uygulama için Sklearn'ü kullanacağız. Bu kütüphane Makine öğrenmesi ile ilgilidir. Sci-kit learn'ün sci kısmı science'ın (bilim) kısaltması, kit ise bir alet çantası ve learn ise öğrenmek anlamına gelmektedir. Yani bir makine öğrenmesi kütüphanesi olan Sklearn kütüphanesinin içinde preprocessing adlı alt kütüphanesi vardır. Preprocessing'in alt kütüphanesi altında olan bir başka kütüphane

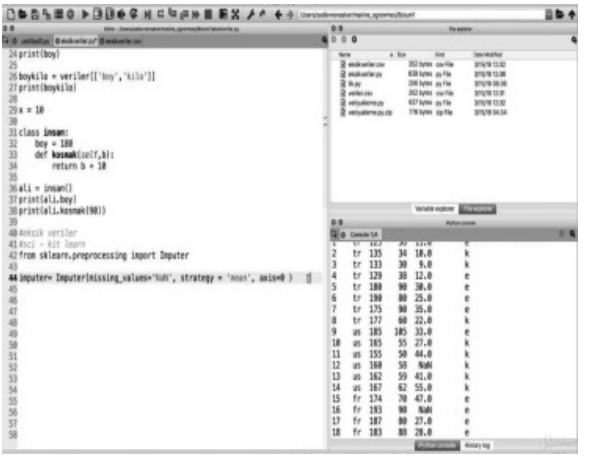
olan imputer'ı importluyoruz. Imput, Türkçe'ye töhmet olarak çevrilebilir. YBS ansiklopedi altında imputation ile alakalı bir yazı mevcuttur.

İlk amacımız data seti imput etmektir. Bunun için Imputer yapısını kullanılacaktır. İnsan tanımladığımız gibi imputer objesini de tanımlayacağız. Missing valuelar(eksik veriler) veri biliminde birçok yerde soru işareti olarak geçmektedir. Parantez açtığımızda parametreleri görmekteyiz. Missing valueları "NaN" olarak olarak tanıtıp, kodumuzun onları okumasını sağlıyoruz.

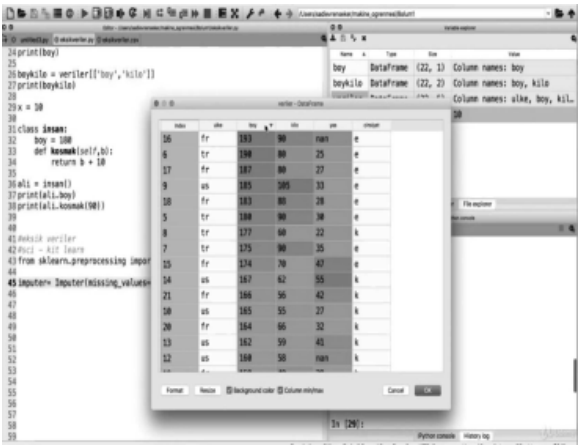

print(boy)
print(boykilo)
x = 10
class insan:
boy = 180
def kosmak(self,b):
return b + 10
ali = insan()
print(ali.boy)
print(ali.kosmak(90))
#eksik veriler
#sci - kit learn
from sklearn.preprocessing import Imputer
imputer= Imputer()
Arguments
Imputer(self, missing_values="NaN",
strategy="mean", axis=0, verbose=0,
copy=True)



Ardından "mean"(ortalama) olarak strateji belirliyoruz. Sonraki aşamadaki amacımız eksik verilerin olduğu yeri ortalamaları ile doldurmaktır.

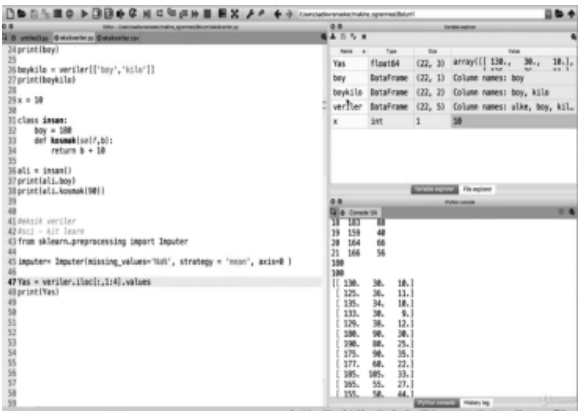


İmputer açtığımızda bu da ortalamanın nasıl alınacağı: satır bazında veya sütün bazında ortalama alınabileceği eklenebilir. Axis bu anlamda kullanılmaktadır. Kod satırımıza axis'i ekliyoruz. Sıfır axisinde bunu kullanması için 0'a eşitliyoruz. Kodu yazdıktan sonra imputer ile bu işlemi yaparak problemi çözüyoruz.



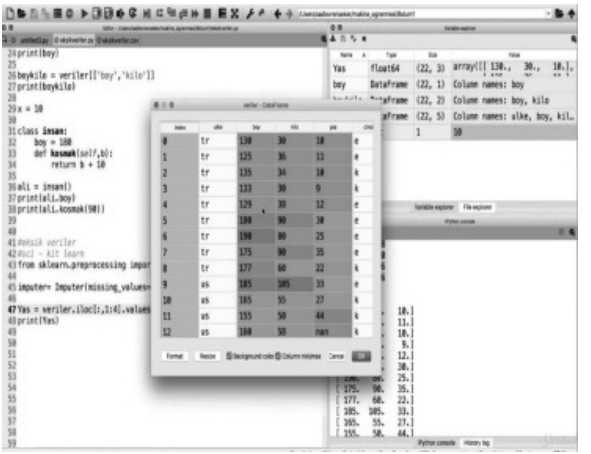
İmput etmeden önce dikkat etmemiz gereken konulardan birisi, imputation stratejisini sayısal bir değer üzerinden kurmuş olmamız. Verileri inceleyecek olursak verilerin ülke bilgisi, cinsiyet bilgisi gibi sayısal olmayan veriler mevcuttur. Sayısal olmayan veriler üzerinde imputerın çalışma şansı yoktur. Bu noktada yapılacak olan

sayısal olan verileri, sayısal olmayan verilerden ayırmak.

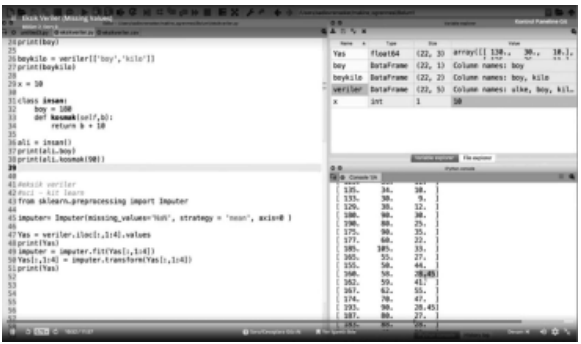


Yaşı imput etmek için yaş ile ilgili kısmı alıyoruz. Verilerin hangi kolonlarını almak istediğimizi belirtmek için iloc alt fonksiyonunu kullanıyoruz. İloc fonksiyonunun altına alan veriyoruz. Belirtiğimiz kolonların değerini val-

ues ile alıyoruz. Ardından print fonksiyonunu kullanarak kolonlarımızı yazdırıyoruz.



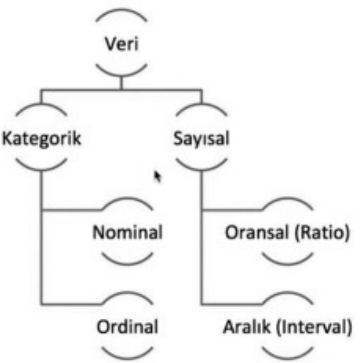
İlk gördüğümüz index kolonu kendiliğinden eklenen sayaçtır. 0. olarak sayacağımız kolon ülke kolonu 1:4 ifadesini kullanarak 1.kolon dahil 4.kolon dahil değil olarak kolonlarımızı çekiyoruz. Böylelikle sayısal verileri almış oluyoruz.



Sayısal verileri aldıktan sonra imputerı kullanabilmeye hazır duruma geliyoruz. “Fit” fonksiyonunu kullanarak belirlediğimiz stratejimizi veri üzerinde uyguluyoruz. Öncelikle her kolon için ortalama değer alınıyor. Sonrasında ise ,yaş kolonunda bulunan sütün kısmında 1’den başlayarak 4’e kadar bütün değerleri alıyoruz. Her 3 kolon için de uygulanabilmek için imputer, ayrı ayrı değerleri hesaplayacaktır. Yaşın değerini imputer ile imput ederek atıyoruz. İlk olarak 1’den 4 ‘e kadar aldığımız kolonlarda transformu kulla-

narak veriyi değiştiriyoruz. Ardından print fonksiyonunu kullanarak sonucu görüntülüyoruz. Sonuç olarak yaş kısmında eksik verilerin yerine 28.45 yazılmış oldu. İnternet sayfasından "eksik veriler.py" olarak kodu bulabilirsiniz.

Bölüm 2: 10.ders Kategorik Veriler



Bu bölümde, veri ön işleme için kritik olan veri tipleri arasında dönüşümden bahsedeceğiz . Takip ettiğimiz metodoloji: crisp-dm yöntemine göre verinin tanınmasından veriyi şuanda modeller için hazırlıyoruz. Çok farklı modeller,çok farklı tiplerde veri isteyebiliyor. Yapay sinir ağları gibi modellemeler sadece sayısal veri

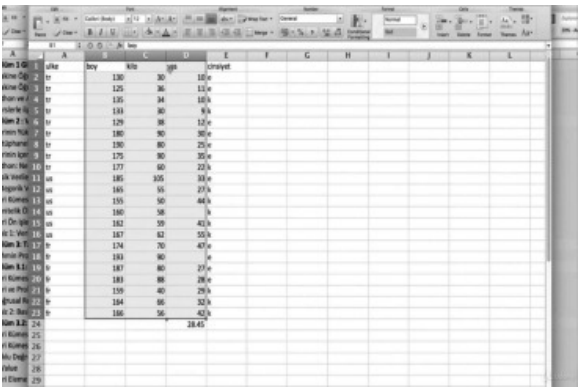
üzerinde çalışıyor ve sadece belli normalize olmuş veriler üzerinde çalışıyor. Bazı veri tipleri ise nominal veriler üzerinde çalışıyor. Bu durumun nedeni ise değişik veri tiplerinin mevcut olmasıdır. Veri, iki ana şekilde incelenebilir: Kategorik veriler ve sayısal veriler. Kategorik veriyle kastettiğimiz örnek olarak kadın veya erkek olması, eğitim düzeyi tarzında verilerdir. Kategorik verilerin aralarında büyüktür, küçüktür ilişkisi yoktur. Sayısal veriler ise bir kişinin yaşı, boyu, ayakkabı numarası tarzında elde ettiğimiz sayısal verilerdir. Kategorik veriler nominal ve ordinal olarak ikiye ayrılırken, sayısal veriler oransal ve aralık olarak ikiye ayrılır.

İlk olarak Kategorik verilerin altında bulunan ordinal verileri inceleyelim. Ordinal veriler kısmı orderdan geliyor ve bunlar bir sıraya sokulan aralarında büyüktür,küçüktür ilişkisine girebilen ama ölçülemeyen şeylerdir. Nominal veriler ise

ne sıralanabilen ne de ölçülebilen verilerdir. Örnek vermek gerekirse bir sokaktaki evlerin kapı numaraları 3 numaralı ev,5 numaralı ev gibidir. Bunlar arasında bir sıralama yapılabilir ama bu sıra herhangi bir ölçü belirtmez. Bunlar üzerinde bir operatör tanımlanamaz. Bu da bir kategorik veridir. Ordinal verilerde sıralama imkanı olduğu gibi nominal verilerde bir sıralama imkanı da yoktur ve önce, sonra şeklinde bir durum belirlenemez. Sayısal veriler arasında oransal(ratio) veriler birbirleri aralarında orantılanıp, çarpılıp bölünebilen verilerdir. Aralık(interval) veriler ise herhangi bir çarpma işlemini kabul etmeyen, toplama gibi işlemleri kabul edebilen değerlerdir. Aralık ile belirtmek istediğimiz elimizde olan verilerin belli bir aralığa hitap etmesi gerekmektedir.

Örnek vermek gerekirse bir odanın ısısının belirli bir değeri vardır ve bu verilerin kendi içinde

birbiri ile büyüktür, küçüktür ilişkisine girmesi mümkündür. Bir değerin zaman içinde artması ve azalması mümkündür fakat çarpmanın herhangi bir anlamı yoktur. Yaş verisine bakacak olursak bu, oransal değerlere girebilecek bir örnektir. Orandan bahsederken verilerin birbirine orantılanabilecek anlamında kullanılmaktadır. Makine öğrenme algoritmalarını değişik sebeplerle bunlar arasında geçiş yapması gerekmektedir.



	ülke	boy	kilo	yaş	cinsiyet
2	tr	130	30	10	e
3	tr	125	36	11	e
4	tr	135	34	10	k
5	tr	133	30	9	k
6	tr	129	38	12	e
7	tr	180	90	30	e
8	tr	190	80	25	e
9	tr	175	80	35	e
10	tr	177	60	22	k
11	us	185	105	33	e
12	us	165	55	27	k
13	us	155	50	44	k
14	us	160	58		k
15	us	162	59	42	k
16	us	167	62	55	k
17	tr	174	70	47	e
18	tr	193	90		e
19	tr	187	80	27	e
20	tr	183	88	28	e
21	tr	159	40	29	k
22	tr	164	86	32	k
23	tr	166	56	42	k
24				38.45	

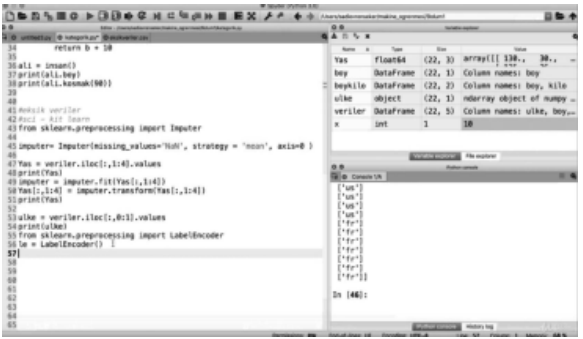
Örnek olarak elimizdeki verileri incelersek, bir makine öğrenmesi modeli geliştireceğiz diyelim ve boy, kilo ve yaş verilerini kullanarak cinsiyetini tahmin edebilir miyiz bunu inceleyelim. Makine öğrenmesi algoritmamızın sayı dışında veri kullanamadığını düşünürsek bu denklemde sayı harici verilerden yararlanamayız demektir. Bunun bir çözümü mevcut ve bu da, bu verileri sayıya çevirmektir. Örneğin Türkiye'ye 1 diye-

lim, Amerika'ya 2 ve Fransa'ya 3 diyelim. Bu çevirim numerik bir sonuç ile bitiyor ve bu numerik sonuç birbirleri ile büyüktür, küçüktür ilişkisine girebilen aynı boy, kilo ve yaş gibi bir sayıya bizi götürüyor. Türkiye'ye 1, Amerika'ya 2 dediğimiz zaman Amerika Türkiye'nin 2 katıdır durumunu kastetmiyoruz. Nominal verinin altında iki çeşit alt veri tipi daha mevcut. Bunlar Bionomial veya Polinomial veri tipleridir. Bionomial, iki ihtimalden biri olması sigara içip veya içmemek,erkek-kadın olma durumu gibidir. Polinomial ise, çok ihtimalden biri olması örnek olarak araba markaları gibidir.

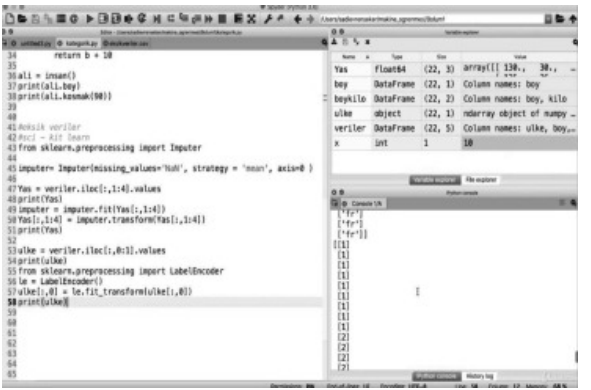
	TR	FR	US
TR	1	0	0
FR	0	1	0
US	0	0	1
TR	1	0	0
TR	1	0	0
FR	0	1	0
FR	0	1	0
US	0	0	1
US	0	0	1
TR	1	0	0
US	0	0	1
TR	1	0	0

Elimizdeki ülke isimlerini yani nominal değerleri numerik değere çevirirken çözüm olarak her bir etiketi kolon başlığına taşıyoruz. O kolonun başlığını taşıyana "1" koyuyoruz. Örneklendirmek gerekirse, Türkiye'de yaşayan birisi 1 diğer ülkelerdeki değeri 0'dır. Verilerin kendi aralarında da kutupsallık mevcuttur.

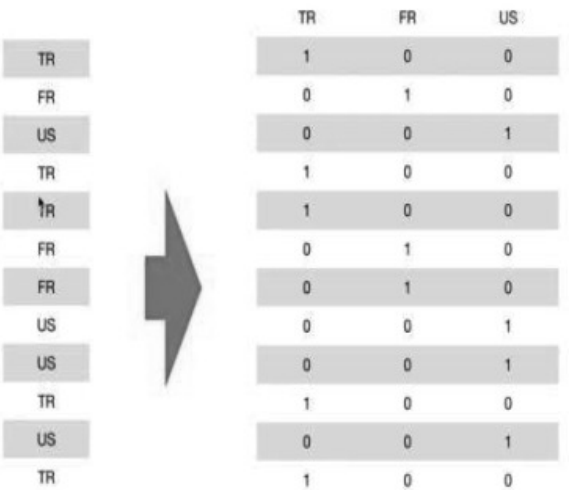
Bu anlattığımız olayın python kodunu yazmakla işe başlayalım. Dosyamızın ismini kategorik veriler olarak değiştiriyoruz. Önceden bildiğimiz üzere veriler dataframemizde kolonlardan hepsi kategorik değildir. Ülke ve cinsiyet bilgilerimiz nominal değerken boy, kilo ve yaş bilgilerimiz numerik değer. İlk olarak ülke kolonunu alıyoruz.



İloc kullanarak veriyi alıyoruz ve print fonksiyonu ile yazdırıyoruz. İlk olarak, Nominal olan bu değeri numerik değere çevireceğiz. Çevirebilmek için ilk olarak encoder kullanıyoruz. Sklearn kütüphanesinin altından ön işleme için preprocessing yapıyoruz ve ardından LabelEncoder'ı import ediyoruz. Sonrasında ise kullanmak üzere LabelEncoder objesi oluşturuyoruz.

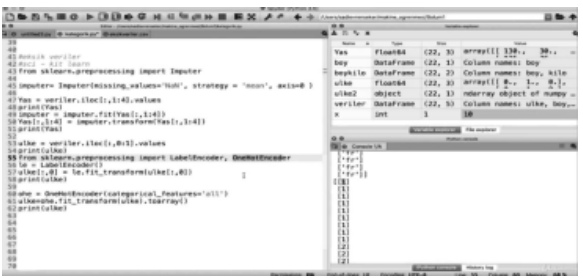

return b + 10
ali = insan()
print(ali.boy)
print(ali.kosmak(90))
#eksik veriler
#sci - kit learn
from sklearn.preprocessing import Imputer
imputer= Imputer(missing_values='NaN', strategy = 'mean', axis=0)
Yas = veriler.iloc[:,1:4].values
print(Yas)
imputer = imputer.fit(Yas[:,1:4])
Yas[:,1:4] = imputer.transform(Yas[:,1:4])
print(Yas)
ulke = veriler.iloc[:,0:1].values
print(ulke)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
ulke[:,0] = le.fit_transform(ulke[:,0])
print(ulke)
Yas float64 (22, 3) array([[130.,
boy DataFrame (22, 1) Column names: boy
boykilo DataFrame (22, 2) Column names: boy, kilo
ulke object (22, 1) ndarray object of numpy
veriler DataFrame (22, 5) Column names: ulke, boy,
x int 1 10
['fr']
['fr']
['fr']]
[[1]
[1]
[1]
[1]
[1]
[1]
[1]
[1]
[2]
[2]
[2]
[2]

Ülke değişkenimize sıfırıncı yani başlangıç kolonunu alarak bunu labelencoder'a koyuyoruz. Kodumuzun Fit fonksiyonunu uygulamasını istiyoruz ama aynı zamanda sonucu veri içine yazmasını istiyoruz. Ülkenin değerini alacağız bu nedenle sıfırıncı kolonu alıyoruz. Printi' kullanarak bu kodumuzu yazdırıyoruz. Türkiye'ye 1,Amerika 2 ve Fransa'ya 0 olarak atadık.



Önceden bahsettiğimiz bu dönüşümü nasıl yapacağımızdan bahsedelim. Sklearn'den preprocessing'i alıp içinden OneHotEncoder'ı seçiyoruz. Bir obje tanımlıyoruz. Parametresini açtığımız zaman kategorik featurları "all" olarak seçiyoruz. OneHotEncoder'ın içinde fit trans-

form'u kullanıyoruz. Fit Transform'da ülke isimlerini array'e çevirtiyoruz. Son olarak ise, Print kullanarak kodumuzu bastırıyoruz.

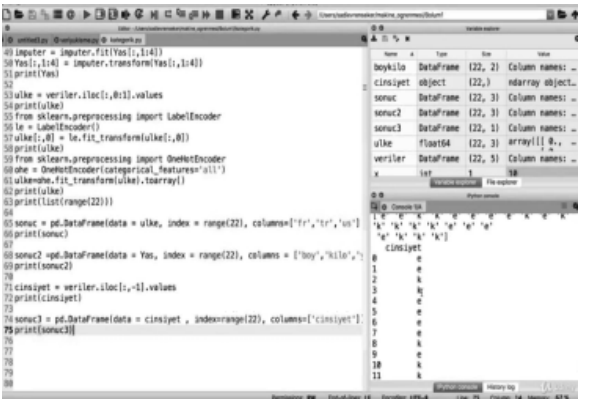


Kodu sadeleştirmek açısından Sklearn preprocessing içinden labelencoderı importladığımız kod satırında OneHotEncoder'ı ekleyerek fazladan bir satırdan kurtulabiliriz.

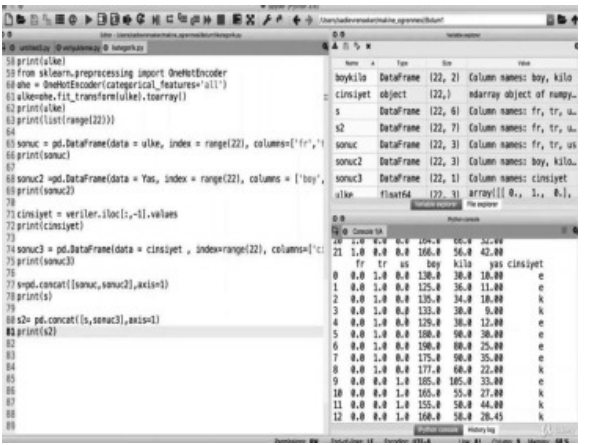
Kitabın bu kısmında verinin, kategorik verilerin nasıl sayısal verilere çevrildiğini değerlerini gördük.

Ders 11: Veri Kümelerinin Birleştirilmesi ve DataFrame Oluşturulması

Variable explorer kısmından elimizdeki verilerimizi görelim. Elimizde ülkeler ,yaş, boy, kilo ve cinsiyet vardır. Ayrıca hiç dokunulmamış orijinal veriler vardır. Ülkenin içindeki değerler kategorik verilerdi ardından bunları sayısal verilere çevirdik. Veriyi kategorik ve sayısal veriler olarak ikiye bölüp kategorik verileri ayrıca ele aldık bunlardan başka bir dataframe çıktı ve sonrasında sayısal verileri ayrı olarak ele aldık ve onlardan da başka bir dataframe çıkmış oldu.


imputer = imputer.fit(Yas[:,1:4])
Yas[:,1:4] = imputer.transform(Yas[:,1:4])
print(Yas)
ulke = veriler.iloc[:,0:1].values
print(ulke)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
ulke[:,0] = le.fit_transform(ulke[:,0])
print(ulke)
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(categorical_features='all')
ulke=ohe.fit_transform(ulke).toarray()
print(ulke)
print(list(range(22)))
sonuc = pd.DataFrame(data = ulke, index = range(22), columns=['fr','tr','us']
print(sonuc)
sonuc2 =pd.DataFrame(data = Yas, index = range(22), columns = ['boy','kilo',
print(sonuc2)
cinsiyet = veriler.iloc[:,-1].values
print(cinsiyet)
sonuc3 = pd.DataFrame(data = cinsiyet , index=range(22), columns=['cinsiyet'].
print(sonuc3)
boykilo DataFrame (22, 2) Column names:
cinsiyet object (22,) ndarray object
sonuc DataFrame (22, 3) Column names:
sonuc2 DataFrame (22, 3) Column names:
sonuc3 DataFrame (22, 1) Column names:
ulke float64 (22, 3) array[[0.,
veriler DataFrame (22, 5) Column names:
x int 1 10
cinsiyet
0 e
1 e
2 k
3 k
4 e
5 e
6 e
7 e
8 k
9 e
10 k
11 k

Ayrı oluşturduğumuz ülke dataframe ile sayısal veriler için oluşturduğumuz dataframleri birleştirmek olacaktır.

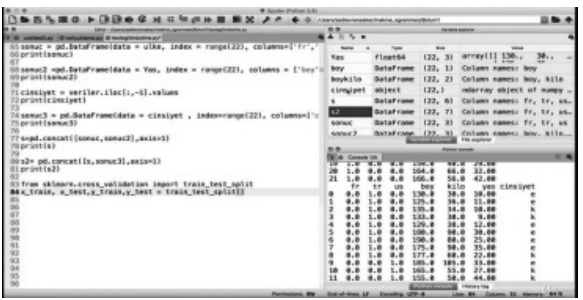


Birleştirmenin ardından veri kümemiz, eğitim ve test kümesi olarak ikiye bölünecektir. Eğitim ve test kümesi makine öğrenmesi algoritmalarının çalışabileceği iki farklı kümeye bölünmesi anlamına geliyor.

Ders 12: Veri Kümesinin Eğitim ve Test olarak bölünmesi

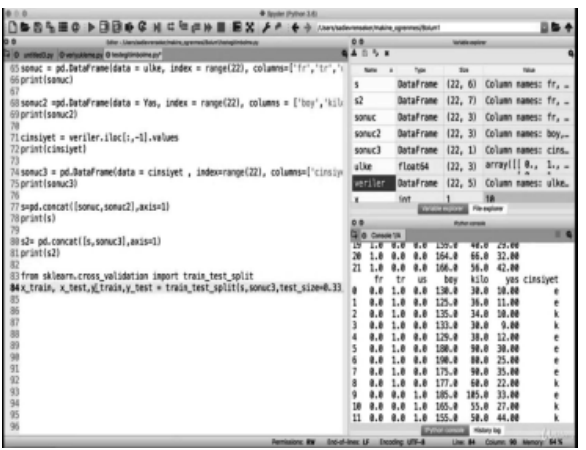
Veriyi bölmekteki amacımız , makine öğrenmesi algoritmasının başarısını rakamsal olarak ölçebilmek. Makinenin veri üzerinden öğrenim sağlaması ve bunun ne kadar başarılı olduğunu görebilmek için Percentage Split kullanıyoruz. Verimizi eğitim verisi ve test verisi olarak ikiye ayırıyoruz. Bu işlemi uygulamaktaki amacımız, makine öğrenmesi algoritmasına eğitim verisini vererek algoritmanın bu verilerden çıkarım yapmasını sağlamaktır. Makineye ilk olarak verinin 2/3'lük kısmını yükleyeceğiz. Makinenin boya, kiloya, yaşa bakarak bunlardan öğrenmesini ardından cinsiyet tahmini yapmasını isteyeceğiz. Bu tahminleri alarak gerçek sonuçlar ile karşılaştıracağız. Bu işlemin sonucunda makinenin başarı kriteri ortaya çıkması beklenir. Bu işlemin ardından algoritmanın yüzdelik dilimde

kaçlık bir başarı oranı olduğunu belirleyeceğiz. Kitabın bu bölümündeki amacımız, verinin işlenmesi aşamasında son adım olarak veriyi test ve eğitim kümesi olarak bölebilmektir.



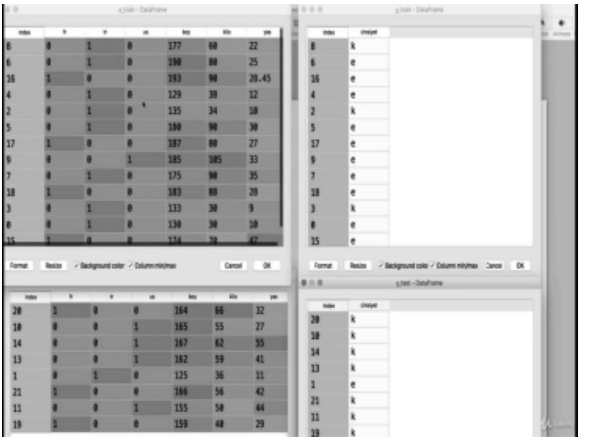
Sklearn kullanarak cross validation uyguluyoruz. Cross validation içinden de veriyi train_test_split olarak bölüyoruz. Makineyi eğitmek için kullanacağımız kısma değişken ismi olarak x ve y diyoruz. Train(x_train) ve test(x_test) şeklinde oluşturuyoruz. Bölme işlemi için sklearn içinden train_test_split'i kul-

lanıyoruz. Dataframe olan sonuc3, istediğimiz kolonu içeriyor.



İzleyeceğimiz yol sonuç 3'ü kendi içinde iki parçaya böleceğiz ve s'i kendi içinde iki parçaya böleceğiz. Buna bağlı olarak öncelikli trainin alı-

nacağı yerler s ve sonuç3 bunları ekledikten sonra 0.33 olarak test boyutu ekliyoruz. Literatürde split validation'un 1/3 test verisi,2/3 eğitim verisi olarak ayrılır. Değişik durumlar için farklı oranlarda veriler ayrılabilir. Random state'i sıfır ,bunun anlamı da veriyi nasıl böleceğidir. Bölme yöntemleri de mevcut örnek vermek gerekirse, veriyi bölerken ilk 14 satırı alırsak elimizdeki örnekler sadece Türkiye ve Amerika'dan toplanan örnekler olacak ve test kısmında ise sadece Fransa'dan gelen örnekler olacaktır.



Sonraki aşama olarak verileri bölündü, sol üst köşede x_train verimizi görebiliyoruz. Kolonlarımızda ülke bilgisi, boy, kilo ve yaş mevcut y_train verimizde ise cinsiyet ayrı bir değişken olarak duruyor. Makine öğrenmelerinde bir sonraki hedefimiz ise sayılar verip, makinenin tahmin etmeyi öğrenmesi ve ardından tahminler ile verilerin karşılaştırılmasıdır. Gözlemleyebileceğiniz üzere index değerleri ardışık değildir çünkü öncesinde random'luk özelliği eklemiştik. Bu bağlamda, makine öğrenmesinin paremetrelerini değiştirmediğimiz halde aldığımız başarı oran sonuçları farklılık gösterecektir. Bunun sebebi ise random(rastgele) olarak ayarlanan eğitim ve test verilerinde her zaman farklı veri kümeleri alması ve bu kümelere ile işlem yapıp farklı sonuçlar ortaya koymasıdır.

Ders 13: Öznitelik ölçekleme

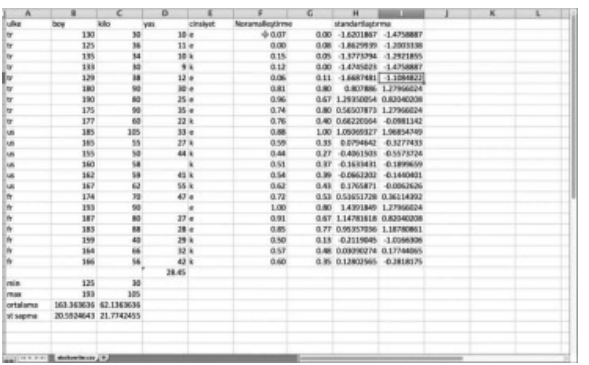
Orijinal verimizi inceleyecek olursak verimiz boy, kilo gibi değişkenlere sahipti. Bu değişkenler üzerinden işlemler yaparak sonuc2'deki halini çıkarttık. Bu verilerimiz numerik verilerdir.

Bu veriler üzerinde makine öğrenmesi algoritmasını kullanacağız. Üç kolonunda veri hareketliliği, istatiksel özellikleri (ortalaması, dağılımı, standart sapması) birbirinden farklı. Boy faktöründe en küçük değer 125'ken en büyük değer 193. Kiloya baktığımızda ise 30 ile 105 arasında bir hareket gözlemleyebiliyoruz.

Yaş değeri ise 9-55 arasında seyretmektedir. Verilerin birbirinden farklı ölçeklerden toplanması durumu veri bilimi için problem teşkil etmektedir. Veri biliminde, bir makine öğrenmesi algoritması oluşturmasından önce önişlem olarak, verileri ortak bir düzlemde birleştirmeliyiz. Bu işlem için klasik iki yöntem bulunmaktadır. Bunlar, Standartlaştırma (Standardisation) ve Normalleştirme (Normalisation)'dir.

Standartlaştırma, herhangi bir sayının toplam sayı kümesindeki ortalama ile farkının alınıp standart sapmaya bölünmesidir. Stadartlaştırmada, aradaki farkın negative çıkabilecek olmasından dolayı ortalama değeri 0'a yakınlaştırmaya

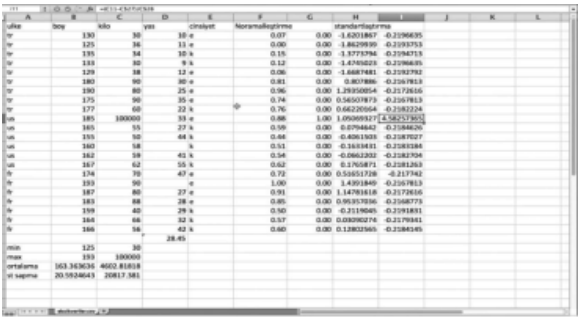
çalışırız. Normalleştirmede ise herhangi bir sayının minimumdan farkının alınıp, maksimum ve minimum arasındaki farkına bölünür. Bu işlemde ise payda pozitif olacağından dolayı veri 0 ile 1 arasındaki değerlere tanınmış olmaktadır. Excel üzerindeki veri tablomuzdaki değişkenler için minimum, maksimum, ortalama ve standart sapma değerlerini tanımlayıp Normalleştirme ve Standartlaştırma denklemi ile sonucu buluyoruz.

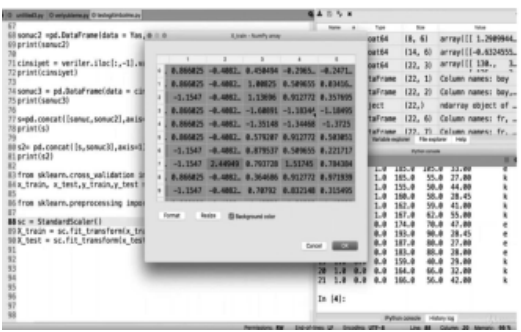


	A	B	C	D	E	F	G	H	I	J	K	L
	ulke	boy	kilo	yas	cinsiyet	Normalleştirme		standartlaştırma				
	tr	130	30	10	e	0.07	0.00	-1.6201867	-1.4758887			
	tr	125	36	11	e	0.00	0.08	-1.8629939	-1.2003338			
	tr	135	34	10	k	0.15	0.05	-1.3773794	-1.2921855			
	tr	133	30	9	k	0.12	0.00	-1.4745823	-1.4758887			
	tr	129	38	12	e	0.06	0.11	-1.6687481	-1.1084822			
	tr	180	90	30	e	0.81	0.80	0.807886	1.27966024			
	tr	190	80	25	e	0.96	0.67	1.29350054	0.83040208			
	tr	175	90	35	e	0.74	0.80	0.56307873	1.27966024			
	tr	177	60	22	k	0.76	0.40	0.66220164	-0.0981342			
	us	185	105	33	e	0.88	1.00	1.05969327	1.96854749			
	us	165	55	27	k	0.59	0.33	0.0794642	-0.3277433			
	us	155	50	44	k	0.44	0.27	-0.4861503	-0.5575724			
	us	160	58		k	0.51	0.37	-0.1619431	-0.1899609			
	us	162	59	41	k	0.54	0.39	-0.0662202	-0.1440401			
	us	167	62	55	k	0.62	0.43	0.1765871	-0.0062626			
	fr	174	70	47	e	0.72	0.53	0.53651728	0.36114392			
	fr	193	90		e	1.00	0.80	1.4391849	1.27966024			
	fr	187	80	27	e	0.91	0.67	1.14781618	0.83040208			
	fr	183	88	28	e	0.85	0.77	0.95357936	1.18780861			
	fr	159	40	29	k	0.50	0.13	-0.2119045	-1.0266926			
	fr	164	66	32	k	0.57	0.48	0.08390274	0.17744065			
	fr	166	56	42	k	0.60	0.35	0.12802965	-0.2818175			
				28.45								
	min	125	30									
	max	193	105									
	ortalama	163.363636	62.1363636									
	st sapma	20.5524643	21.7742455									

F ve H sütünlarında boy değişkenine bağlı olan sonuçlar görülmektedir. Örnek olarak, F sütünunda 193 boyundaki bir değer normalleştirme sırasında maksimum değere eşit olduğundan dolayı sonuç 1.00 olarak görülmektedir. Benzer bir bağlantıyla, H sütunundaki 164 boya sahip biri için boy ortalaması 163 olduğundan dolayı standartlaştırma değeri 0'a çok yakın ve pozitif (0.03090274) kalmıştır. Aynı durum kilo değişkenine bağlı olarak G ve I sütunlarında benzer sonuçlar vermektedir

Gürültülü veri kullanımındaki durumundan hareketle Standartlaştırma, Normalleştirme'ye göre daha stabil sonuçlar verdiğinden dolayı sektörde öncelikli olarak tercih edilmektedir.





Sklearn kullanarak preprocessing uyguluyoruz ve StandardScaler'i import ediyoruz. Standard-

Scaler ile train ve test kümelerini fit_transform ile ölçeklendiriyoruz. Bu kısımdaki genel amacımız, eksi sonsuzdan artı sonsuza dağılan veriler arasından ortalama değeri 0'a getirmek için Standartlaştırma kullanmak ve farklı düzlemlerdeki verileri aynı düzleme çekilebileceğini göstermektir.

Ders 14: Veri Ön işleme Şablonu

Bu kısımdaki amacımız, gelecekteki projelerde kullanılabilecek veri ön işleme ile ilgili bir şablon oluşturmaktır. Çoğu veri bilimcisinin kendi kod kütüphaneleri (Repository) vardır ve ihtiyaç durumunda bu kodları gerçek projelerinde orijinal olarak ya da küçük değişikliklerle kullanırlar. Veri ön işlemenin ilk adımı veri yüklemedir. Bunun için veri kümemiz direk olarak dosya yö-

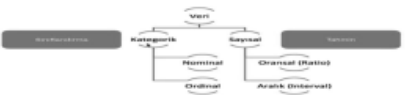
neticisinden ya da makineye dosya için direk bir yol göstererek (file:///C:/Users ..) belirlemeliyizdir.

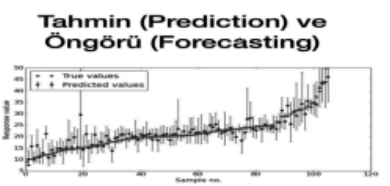
Veri ön işlerken verinin hangi kısmına ulaşmak istediğimizi doğru seçmek önemlidir. Örnek olarak, eğer imputer kullanacaksak ilgili imput edilecek kısmı, imputer'ın transformuna vermek gereklidir. Bu şekilde ihtiyaç duyulan kodu veri içerisinden kısa bir yol ile ulaşabilmekteyiz.

Bölüm 3: Ders 15 Tahmin Problemlerine Giriş

Tahmin için kullanılan algoritmaların büyük bir kısmı istatistik modellerinden gelmektedir. En çok kullanılan problem tipleri Sınıflandırma ve Tahmin problemleridir. Önceki bölümlerde verileri iki ana grupta incelemiştik. Bunlar, Kategorik ve Sayısal verilerdi. Genel olarak, kategorik veriler (cinsiyet, eğitim düzeyi vb.) üzerinde bir tahmin yapıldığında bunlar Sınıflandırma (Classification) , aynı şekilde sayısal(maaş, yaş

vb.) veriler üzerinde bir tahmin yapıldığında bunlar Tahmin (Prediction) olarak kabul edilmektedir.





Belirli bir zaman serisi üzerinde bir tahminde bulunulmuşsa ve bu zamanın daha ilerisi için örnekler üzerinden bir tahmin yapılmasına öngörü (Forecasting) denir. Örnek olarak, geçmiş finansal veriler ile geleceğe dair bir çıkarım (Örn: gelecek yıl dolar kuru) yapmayı öngörü olarak kabul edebiliriz. Prediction'da ise geçmişe dair tahmin yapabilip, olası eksik verilerin tahmini için kullanılabilmektedir.

Bölüm 3: Doğrusal Regresyon

Ders 16: Veri Kümesinin İndirilmesi

http://www.bilkav.com/makine-ogrenmesi-egitimi/ [3]adresinden veri kümelerine ulaşılabilir. Önümüzdeki ders için kullanacağımız veri kümesi "Aylara göre Satış Verileri" dosyasının verileridir, lütfen dosyayı indiriniz.

3. http://www.bilkav.com/makine-ogrenmesi-egitimi/

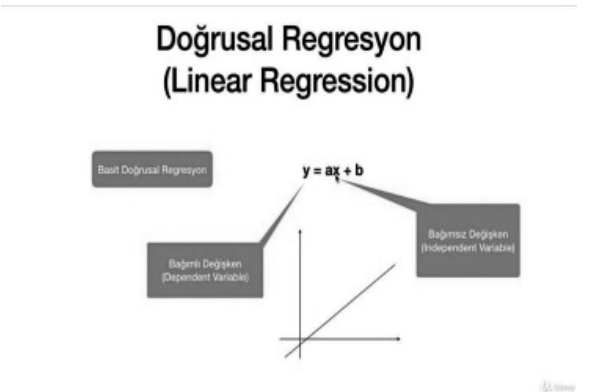
PYTHON İLE MAKINE ÖĞRENMESI117

Ders 17: Veri ve Problem Tanımı

Bu dersen sonrasına devam edebilmek için veri ön işleme bölümünde (Bölüm 2) yazılan kodlara ihtiyacınız olacaktır. Kodları dersin web sitesindeki bölüm 1 sonunda bulunan şablon kısmından indirebilirsiniz.

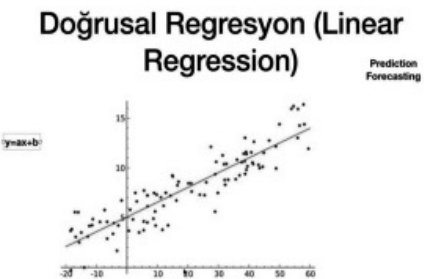
Ders 18: Basit Doğrusal Regresyon

Bu kısımda bir tahmin algoritması kullanacağız ve ilk kez deneyimleyeceğimiz makine öğrenme algoritmasına giriş yapacağız. Öncelikli olarak, doğrunun formülünü hatırlayalım "y = ax + b".

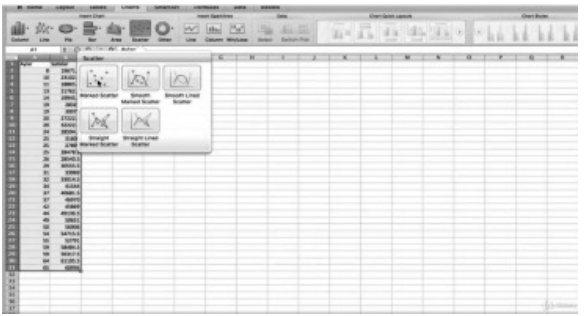


Buradaki "a" katsayı olarak kullanılabilse de aslında temelde doğrunun eğimini, b ise doğrunun iki boyutlu uzayda ne kadar öteleneceğini

göstermektedir. Bu doğrultuda çalışmamızın temel amacı, veri kümemizi en iyi modelleyebilen doğrunun formülünü çıkarmaktır. Bu doğru, tahmin bilgilerinin geneline en yakın doğru olması önemlidir.

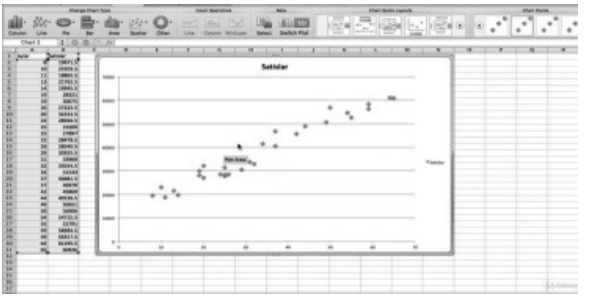


Bu tabloda, doğru ile nokta arasındaki uzaklık ne kadar fazla olursa hata oranı o denli yükselmektedir. Bu ölçümdeki amacımız ise her tahminin hata oranını ölçüp minimize etmeye çalışmaktır.



Veri kümemizi Marked Scatter ile grafiksel bir görsele dönüştürdüğümüzde "Ay/Satış" verilerine ait veri grafiği elde etmekteyiz. CRISP-DM metodolojisinde verinin anlaşılmasının büyük önemi olduğundan dolayı bu grafikteki doğrusallığın sezilmesi ve bu grafik için uygun doğrunun konumlandırılması, hataları minimize etmek için bize yardımcı olacaktır. Bu hata azaltımı ile yüksek doğrulukta modelleme yapabilmek mümkün olacaktır.

PYTHON İLE MAKINE ÖĞRENMESI121



Sonuç olarak, Basit Doğrusal Regresyon bize basit bir doğru aracılığı ile yapılmış tahminlerin hatalarını minimize etmemizi ve daha sağlıklı modeller oluşturup daha başarılı öngörüler sunabilmeyi imkan haline getirmektedir.

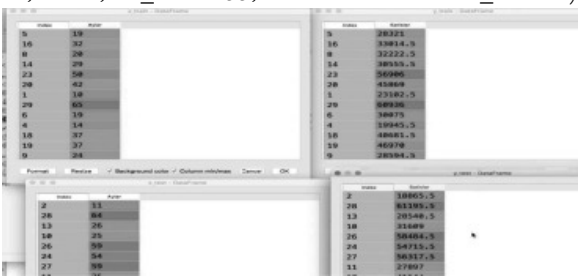
Ders 19: Veri Yükleme ve Ön İşleme Şablonunun Kullanılması ve Regresyona Hazırlık

Bu derste doğrusal regresyona giriş yapacağız. Bu kısımda, önceden hazırladığımız (Ders14) örnek şablonu kullanacağız.Öncelikle şablonu indiriniz. File Explorer ile indirdiğiniz dosyayı açınız. Eksik veriler, kategorik verilerin nümeriğe çevirilmesi kısımlarına artık ihtiyacımız kalmadığı için siliyoruz. Veri önişleme kısmındaki parametreleri "aylar-satışlar" olarak değiştiriyoruz.

```
aylar = veriler[['Aylar']]


print(aylar)


satislar = veriler[['Satislar']]
```

Eğitim-test kümesi oluşturuken “satıslar” bağlı, “aylar” bağımsız değişken olduğunu göz önünde bulundurup x_train ve x_test değerlerini çıkartıyoruz

x_train, x_test,y_train,y_test = train_test_split(aylar,satislar,test_size=0.33, random_state=0)



Buraya kadar; verinin nasıl yükleneceği, veriyi bölmeyi ve ilgili datasetlerle ilgili gerekli ayarlamaları yaptıktan sonra eğitim-test kümesindekiveriyi scale edip standartlaştırılmış hale dönüştürdük.

Ders 20: Python ile Basit Doğrusal Regresyon Model İnşaası

Bu kısımda, train kümesinden gelen eğitim sonucunu test kümesi üzerinde uyguluyacağız, fakat burada dikkat etmemiz gerekeen durumlardan birisi de train kümemizin bağlı ve bağımsız değişkenlerden oluştuğudur. Buraya kadar olan kısımda veri önişleme işlemimizi yaptık ve artık Modelleme kısmına hazırız. Öncelikler scikit learn kütüphanesindeki linear_model ile doğrusal regresyonu import ediyoruz.

Eksiksiz bir model için , Verilerin ölçeklendirilmesi kısmına Y'ye ait değerleri de ekliyoruz.

```
Y_train = sc.fit_transform(y_train)
```

```
Y_test = sc.fit_transform(y_test)
```

Doğrusal regresyonu çağırabilmek için bir obje oluşturuyoruz (“lr”). “.fit” fonsiyonu ile objemizin bir model inşaa etmesini sağlıyoruz.

```
lr.fit(X_train,Y_train)
```

Dipnot: Kısayol olarak, (Mac) için "cmd + i" veya "CTRL + I" (Windows) ile seçilen kod hakkında bilgiye ulaşbiliyoruz.

Ders 21: Python ile Basit Doğrusal Regresyon Uygulaması

Bu derste ilk regresyon uygulamasını yapacağız. "lr" objesi ile tahmin fonskiyonunu çağırıyoruz ve tahmin etmesini istediğimizi parametreyi (X_test) giriyoruz ve bu sonucuda tahmin değişkenine atıyoruz.

```
tahmin = lr.predict(X_test)
```

Burada toplamda dört adet veri kümemiz var X_train'den Y_train'i öğrendi, X_test'den de kendi tahmin sonuçlarını çıkardı.

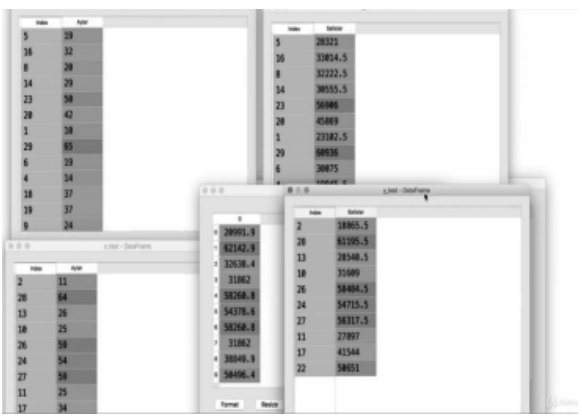
Aradaki farkı görmek için verinin ölçeklendirilmesi kısmını yorum satırına dönüştürüp, kodu yeniden x_train, y_train ,x_test ile deniyoruz.

```
from sklearn.linear_model import LinearRegression
```

```
lr = LinearRegression()
```

```
lr.fit(x_train,y_train)
```

```
tahmin = lr.predict(x_test)
```



Gerçek veriler ile tahmin bölümü arasındaki sapma sonucunu sonraki derslerde python ile hesaplayacağız fakat bu ders özelinde esas amacımız oluşturduğumuz modeli tahmin için kullanmak idi. Ayrıca önümüzdeki derslerde regresyon modelleri için başarı kriteri hakkında değerlendirme yapacağız.

Ders 22: Basit Doğrusal Regresyon Görselleştirmesi

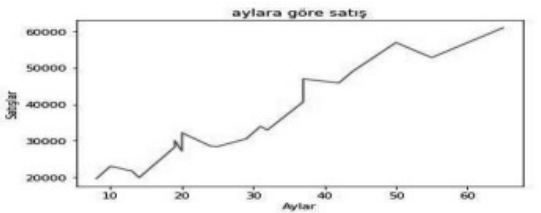
Görselleştirme yöntemi için geniş bir seçenek hazuvu olsa da bu derste basit bir şekilde veriyi keşfetme amaçlı görselleştirme yaparak konu bazlı görsel sonuçlar elde etmeye çalışacağız.

İlk olarak hatalı bir kod olan "plt.plot(x_train,y_train)" çalıştırılıyor, bunun nedeni ise değerlerin rastegele olarak eğitilmesidir. Bu hatanın çözümüsü ise bu değerleri sıralamaktır. Ancak dikkat edilmesi nokta, datasetlerin indexlere göre sıralanmasının en doğru çözüm olması çünkü verilere göre sıralamada en küçük ay için en küçük satış değeri tanımlanacağından dolayı yapmak istediğimiz tahmin için engel teşkil edecektir.

```
x_train = x_train.sort_index()
```

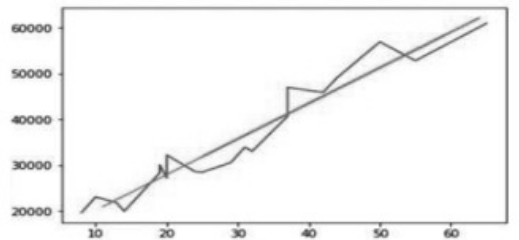
```
y_train = y_train.sort_index()
```

Bu sıralama sonucunda grafiğimizde doğrusallık görmekle berabere artış tarafında bir süreklilik tespit edememekteyiz.



Dipnot: Buradaki verilerin train kümelerinden gelen %66'lık kısım olduğunu ve değişkenik gösterebileceğini unutmamak gerekmektedir.

Doğrusal Regresyon kullanmak için "plt.plot(x_test,lr.predict(x_test))" kodunu kullanıyoruz.



Bu tabloda ise test değerleimizi ve o test değerlerine karşılık gelen tahmin değerlerini görmekteyiz.

Amacımız olan gerçek değerlere en yakın doğruyu çizmeyi turuncu doğru ile yapmış bulunmaktayız.

Ayrıca grafiğin doğruluğu için grafik, x ve y değerlerine başlık atamalıyız.

```
plt.title("aylara göre satış")
```

```
plt.xlabel("Aylar")
```

```
plt.ylabel("Satışlar")
```

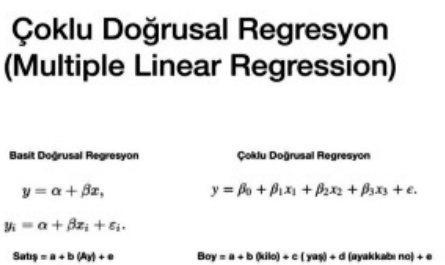
3.2 Çoklu Doğrusal Regresyon

Bu kısımdaki genel amacımız, birden fazla değişkenden oluşan durumlar için farklı problem çözümlerini irdelemektir.

Ders 23: Veri Kümesi ve Problemin Tanımı

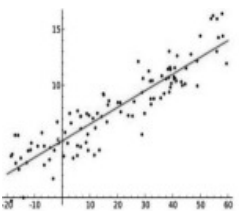
Basit doğrusal regresyonda bahsedilen durum, bir bağımlı değişkenin başka bir bağımsız değişkene bağlı olma durumuydu (Ders 16). "veriler.csv" datasetimizdeki(Ders 5) veriler ile yüksek düzey doğruluk oranına sahip bir tahmin yapmak istediğimizde birden fazla bağımsız değişken(boy,kilo,cinsiyet,ülke,yaş) durumundayız. Bu nedenle bu değerlendirmede Çoklu Doğrusal Regresyon kullanmalıyız.

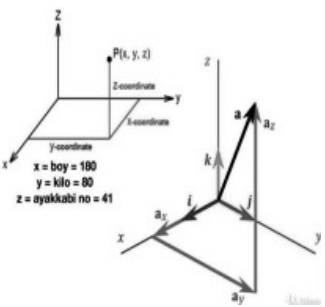
Ders 24: Çoklu Değişkenlerdeki Problemler ve Çözümleri



“” değeri “ i ” örneği için bir hata miktarını temsil etmektedir ve bu değer farklı örnekler için farklılık gösterebilmektedir. Çoklu doğrusal regresyon formülü ise birden fazla değişken ile çalışmak için tasarlanmıştır ve her bir değişken için farklı bir çarpan olacağı kabul edilmektedir. “Bo” ise herhangi bir değere bağlı olmayan “ a ”

gibi bir sabit sayıyı temsil etmektedir. Ayrıca her örnek için bir hata miktarı da oluşturulmalıdır. Çoklu regresyonda birinci dereceden bir denklem kullanıldığından dolayı bir doğru ve buna bağlı olarak hata miktarı ortaya çıkacaktır. Üç değişkenli problemi modellemek için üç boyutlu uzay konsepti kullandık fakat daha fazla değişken için modelleme yapmak da mümkün. Bunun için her bir değişken farklı bir eksene atanmaktadır.





Ders 25: Kukla Değişken(Dummy Variable) ve Kukla Tuzağı

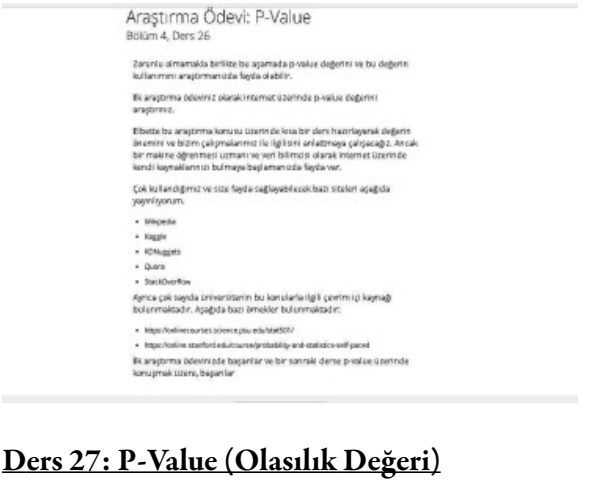
Kukla değişkeni bir değişkeni ifade eden başka bir değişken olarak tanımlayabiliriz. Kukla değişken, önişleme aşamasında değişkenler üzerinde kullandığımız değişik lik yapmamıza olanak sağlayan önemli özelliklerden biridir.

Burada bir kukla ve bu kuklayı yönlendiren bir kuklacıdan bahsedebiliriz dolayısıyla aynı anda iki bilginin birlikte alınması bizim için risk arz etmektedir. Bu riskin sebebi ise her kolunun sonuç makine öğrenme algoritmasına bir etkisi olması ve bu etkilerin eşit olarak dağılmasıdır. Ancak bazı algoritmaların bu örneğimizdeki gibi bir riske karşı dirençli olabileceği de unutulmamalıdır. Bu nedenle Kukla değişkeni ortaya çıkartmak istediğimizde dikkatli ve hassas seçimler yapmalıyız. Verinin üzerinde değişikler yaparken veya Kukla değişkeni ortaya çıkartırken Kukla değişkeni ile orijinal veriyi aynı anda çekmemek yönünde insiyatif kullanmalıyız.Eğer ki aynı anda bu veriler alınırsa sistem otomatik olarak birbirinden etkilenmeye başlayabilir.

Bazı durumlarda Kukla değişkeni örnekteki gibi iki veya daha fazla kolon üretebilir fakat bu kolonlar arasında da doğrudan bir bağlantı olabilir. Bu gibi durumlarda da Kukla değişkenlerin de kendi içinde seçilmesi gerekmektedir. Örnek vermek gerekirse, il bilgilerinin olduğu kolondaki şehir verileri kolon başlığına çevirdiğimizde bütün kolonları incelememiz gerekiyor çünkü bir bilgiden (Örn: İstanbul olmaması) ile başka bir bilgi(Örn. İzmir olması) çıkarımını kesin olarak yapmak mümkün değil. Ancak binominal değişkenlerin durumlarında yukarıdaki örneğimizdeki gibi (Erkek,Kadın olma durumu)

birbirlerine göre çıkarım yaparak değişkenleri saptamaya çalışmak bizim için tuzak oluşturuyor ve çözüm olarak verimizi tekrar incelememiz gerekmektedir.

Ders 26: Araştırma ödevi : P-Value



Araştırma Ödevi: P-Value

Bölüm 4, Ders 26

Zaman(u olmamakla birlikte bu aşamada p-value değerini ve bu değerin kullanımını araştırmanızda fayda olabilir.

Bir araştırma ödevi olarak internet üzerinde p-value değerini araştırınız.

Elbette bu araştırma konusu üzerinde kısa bir ders hazırlayarak değerin önemini ve bizim çalışmalarımız ile ilgisini anlatmaya çalışacağız. Ancak bir makine öğrenmesi uzmanı ve veri bilimcisi olarak internet üzerinde kendi kaynaklarınızı bulmaya başlamanızda fayda var.

Çok kullandığımız ve size fayda sağlayabilecek bazı siteleri aşağıda yayınlıyorum.

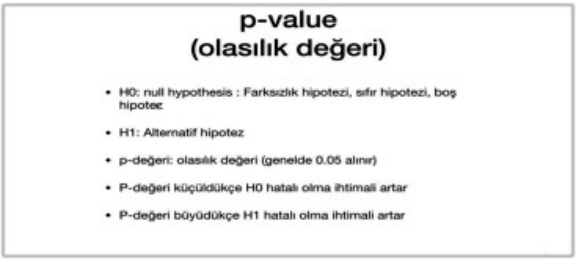
- Wikipedia
- Kaggle
- KDNuggets
- Quora
- StackOverflow

Ayrıca çok sayıda üniversitenin bu konularla ilgili çevrimiçi kaynağı bulunmaktadır. Aşağıda bazı örnekler bulunmaktadır:

- https://online.stanford.edu/course/probability-and-statistics-self-paced

Bir araştırma ödevinizde başarılar ve bir sonraki derste p-value üzerinde konuşmak üzere, başarılar

Ders 27: P-Value (Olasılık Değeri)



Olasılık değeri, hesaplamalarımızda oldukça faydalı olan değerlerden biridir ve genel olarak bir hipotezin değerlendirmesi sürecinde kullanılmaktadır. H0 olarak düşünülen hipotez, temel hipotezimizden oluşmaktadır (Örn."Kurabiyelerimizin hepsi 30 gramın üzerindedir"), H1 ise bu hipoteze karşılık gelen ters hipotez olarak tanımlanabilir. Yalnızca bir örnek ile hipotezimizi çürütmek mümkün değildir çünkü sistem içerisinde marjinal değerler olabilmekle beraber asıl hipotezi tehdit etmemektedir. P-değeri bize "ne kadar durumda örnek bulursam bu hipotezi çürütebilirim?" sorusunun cevabını vermeyi hedeflemektedir. Bu değer bulunurken

örnekleme ve asıl önemli kısım budur.Bu yöntem, rastgele seçilen örnekleri hipotezimizle karşılaştırıp sistem hakkında bir öngörü yaratmamızı sağlamaktadır fakat bunun öncesinde p-değerini hesaplayıp bu değer hakkında bir sınır(Significance Level) koymamız gerekmektedir.

Ders 28: Çok Değişkenli Modellerde Değişken Seçimi

Birden fazla değişken olma durumunda bütün değişkenlerin aynı ya da birbirinden farklı oranlarda sistemi etkileyip etkilemediğini ve hatta

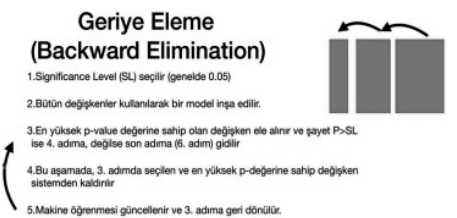
hiç etkilemediğini saptamak durumundayız. Bu durum bizi değişkenler arasında seçim yapmaya zorlamaktadır çünkü dahil olan yeni değişkenler sistemin başarı yüzdesini olumlu veya olumsuz şekillerde etkilemektedir.

-Bütün Değişkenleri Dahil Etmek

Bu yaklaşım bazı durumlarda, örneğin daha önceden değişken seçimi yapılmış,zorunluluk bulunduran(banka kredi skoru modelindeki

başarı ölçümü) veya diğer yaklaşımları kullanmadan önce keşif için bir önfikir oluşturulma amacıyla kullanılabilmektedir.

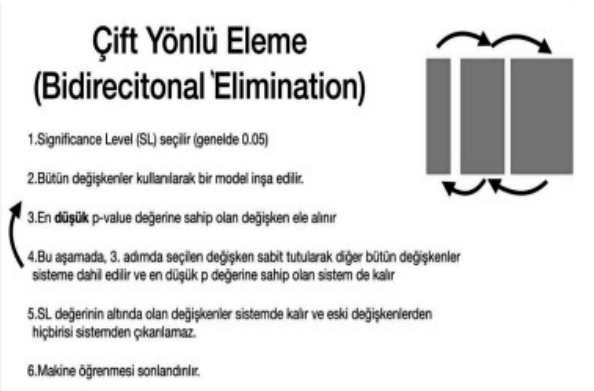
- **Geriye doğru eleme (Backward Eliminaton)**



İleriye seçim (Forward Eliminaton)

- Çift Yönlü Eleme (Bidirectional Elimination)

Bu yaklaşımda birden fazla SL seçilebilir. Bunun nedeni ise ileri ve geri yönlerde kullanacağımız SL değerleridir.



-Skor Karşılaştırması (Score Comparison)

İlk olarak başarı kriteri belirlenir ve olası bütün makine öğrenmesi modelleri inşaa edilir(ikili seçim). Bunun devamında ilk adımdaki başarı kriterini sağlayan en iyi yöntem bulunurak makine öğrenmesi sonlandırılır.

Ders 29: Çoklu Değişken için Veri Hazırlama (Python Kodu)

Bu kısımda ders 14 ve ders 5' de kullandığımız kaynakları kullanacağız (http://www.bilkav.com/makine-ogrenmesi-egitimi/)[4].

Öncelikle bir regresyon oluşturabilmek için "veriler.csv" dosyamızdaki cinsiyet kolonunu nümerik değerlere çevirmemiz gerekmektedir. Bunun nedeni ise, daha önceki derslerimizde de hatırlayacağımız gibi Regresyon algoritmalarının matematiksel denklemler üzerinde çalışmakta olduğu ve doğrunun denklemini çizebilmek için üç boyutlu uzayda noktalara ihtiyacımız olduğu idi.

4. http://www.bilkav.com/makine-ogrenmesi-egitimi/

Bu dönüşüm için Encoding kullanacağız. Önceki derslerimizde ulke kullandığımız encoding kodunu cinsiyet(c) değişken için yeniden dizayn ediyoruz.

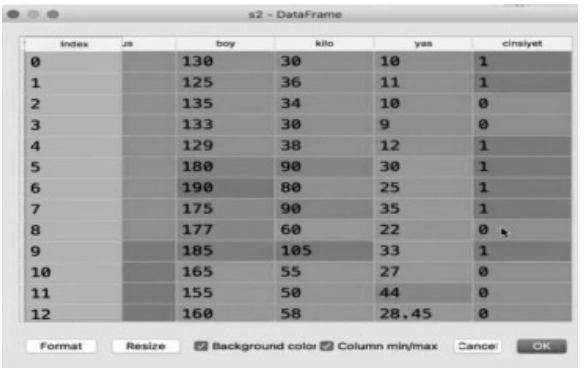
```
c = veriler.iloc[:,-1:].values

print(c)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

c[:,0] = le.fit_transform(c[:,0])

print(c)

from sklearn.preprocessing import OneHotEncoder
```

```
ohe = OneHotEncoder(categorical_features='all')

c=ohe.fit_transform(c).toarray()

print(c)
```

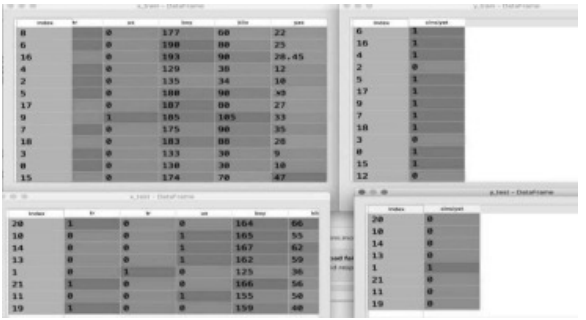
Bu tabloda da görebileceğimiz gibi 0 ve 1' ler bibirini tamamlayan değerler olduğundan dolayı akıllara Kukla Değişkeni Tuzağı olduğunu getirmektedir. İki kolonun birlikte olması risk arz etmekle birlikte değişkenleri tek kolona indirgeyerek bu tuzaktan korunmaya çalışılacaktır.

```
cinsiyet = veriler.iloc[:,-1].values


print(cinsiyet)


sonuc3 = pd.DataFrame(data = c[:,:1] , index=range(22),
columns=['cinsiyet'])


print(sonuc3)
```

Bu kod ile 0 ve 1'leri tek kolona indirgeyebiliriz.



Bu indirgeme ile y_train ve y_test kısımlarımız güncellenmektedir ve bu sayede Çoklu Doğrusal Regresyon'dan (birden fazla kolon arasında bağlantı kurmak) bahsedebiliriz. Buradaki genel arayışımız, bu değerler üzerinden yüksek doğrulukta cinsiyet tahmini yapmaktır.



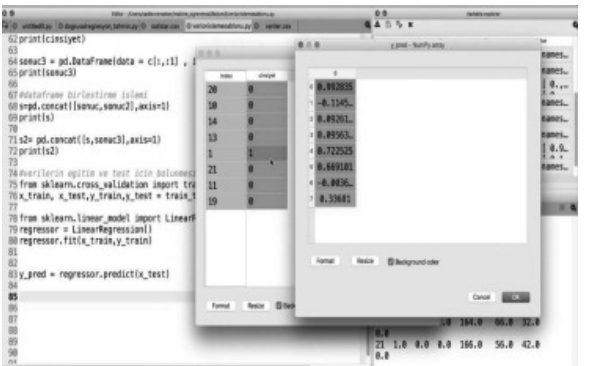
<u>Ders 30:</u> Çoklu Değişken Linear Model Oluşturma Python Kodlaması ve Model

Cinsiyet tahimini için doğrusal regresyona benzer bir model kullanacağız. Basit doğrusal regresyon yerine

birden fazla bağımsız değişken kullanabileceğimiz doğrusal regresyon modeli import ediyoruz.

“regressor” adlı obje tanımlıyor ve fit fonksiyonunu ile x_train ve y_train arasında doğrusal bir istatiksel

model oluşturuyoruz . “y_pred” ile verinin %67 sini oluşturan (x_train ve y_train) kısmından gelen verileri tahmin edeceğiz. “regressor.predict(x_test)” ile tahmin işlemimizi yapıyoruz.



```
print(cinsiyet)

sonuc3 = pd.DataFrame(data = c[:,:1] , i
print(sonuc3)

#dataframe birleştirme işlemi
s=pd.concat([sonuc,sonuc2],axis=1)
print(s)

s2= pd.concat([s,sonuc3],axis=1)
print(s2)

#verilerin eğitim ve test için bölünmesi
from sklearn.cross_validation import tra
x_train, x_test,y_train,y_test = train_t

from sklearn.linear_model import LinearR
regressor = LinearRegression()
regressor.fit(x_train,y_train)

y_pred = regressor.predict(x_test)
```

from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

regressor.fit(x_train,y_train)

y_pred=regressor.predict(x_test)

Aynı işlemi boy tahmini için yapacağız. “S2” datasetindekindeki boy kolonunun solundaki ve sağındaki değerleri alıp bir dataframe olarak birleştireceğiz ve bu x’i oluşturuken boy ise tek başına y’yi oluşturacak. Bu veri manipülasyonları için “.iloc” kullanıyoruz. Boy kolonunun solundaki ve sağındaki değerler için “sol” ve “sag” değişkenlerini oluştururken kendisi için “boy” değişkenini oluşturuyoruz. Sol ve sağ verileri birleştirmek için “concat” kullanıyoruz. “veri” değişkenini yaratıp sol ve sağ verilerini birleştiriyoruz.Ara sonuç olarak kodumuzda bağımsız değişkenleri içeren veri kümesi ve bağımsız değişkeni(boy) içeren boy kolonumuz var. Sonraki adım olarak boy değişkenine göre veri kümemizi bölüyoruz. Yeniden doğrusal regresyon modelini kullanıyoruz.

```
from sklearn.linear_model import Linear-
Regression


boy=s2.iloc[:,3:4].values


print(boy)


sol=s2.iloc[:,:3]


sag=s2.iloc[:,4:]


veri=pd.concat([sol,sag],axis=1)
```

```
x_train, x_test,y_train,y_test = train_test_split(veri,boy,test_size=0.33, random_state=0)


r2 = LinearRegression()


r2.fit(x_train,y_train)


y_pred=r2.predict(x_test)
```


#verilerin egitim ve test için bolunmesi
from sklearn.cross_validation import train_test_split
x_train, x_test,y_train,y_test = train_test_splitis,sonuc3,test_size=0.33, r
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train,y_train)
y_pred = regressor.predict(x_test)
boy = s2.iloc[:,3:4].values
print(boy)
sol = s2.iloc[:,:3]
sag = s2.iloc[:,4:]
veri = pd.concat([sol,sag],axis=1)
x_train, x_test,y_train,y_test = train_test_split(v
r2 = LinearRegression()
r2.fit(x_train,y_train)
y_pred = r2.predict(x_test)
ulke float64 (22, 3) array([[0.
veri DataFrame (22, 6) Column names..
veriler DataFrame (22, 5) Column names..
x_test DataFrame (8, 6) Column names..
x_train DataFrame (14, 6) Column names..
y_pred float64 (8, 1) array([[182.
y_test float64 (8, 1) array([[164.
182.254
153.6
163.844
158.797
130.887
173.764
150.555
157.278
164
165
167
162
125
166
155
159

Ders 31: Python ile Geri Eleme (Backward Elimination)

Bu dersimizde, modelimizin ve modelimizdeki değişkenlerin başarısını ölçmek ve p-değerini hesaplamaya çalışacağız. Öncelikle statsmodels kütüphanesi altındaki formula kısmındaki api çağırıyoruz ve sm değişkenine tanımlıyoruz. İlk olarak değişkenleri sisteme ekliyoruz. İlk olarak yapacağımız işlem,veri kümemizdeki değişkenlerden hangilerinin sisteme daha fazla etkisinin olduğunu görebilmek için bu değişkenleri içeren bir diziyi değerlendirmek olacaktır. Bu diziyi oluşturduktan sonraki izleyeceğimiz yol sırasıyla değişkenleri sistemden çıkartıp sonucun değişim durumunu gözlemlemek olacaktır. İşlem olarak yapacağımız ilk şey veri kümesine bir dizi ekleme olacaktır. Bunun nedeni ise çoklu doğrusal regresyon modelindeki ihtiyacımız olan for-

mülden de hatırlayacağımız gibi B0 sabit değerini yaratmak.

import statsmodels.formula.api as sm

X = np.append(arr = np.ones((**22,1**))**.astype(int), values=veri, axis=1**)

Bütün kolonların p-değerini hesaplamak için liste oluşturuyoruz bu işlem geri eleme yaparken bize kolaylık sağlayacak.

X_l = veri.iloc[:,[0,1,2,3,4,5]].value

Bir regresyon sonucu r yaratıp “sm.OLS” ile istatiksel değerlerimizi çıkarıyoruz.

r_ols = sm.OLS(endog = boy, exog =X_l)

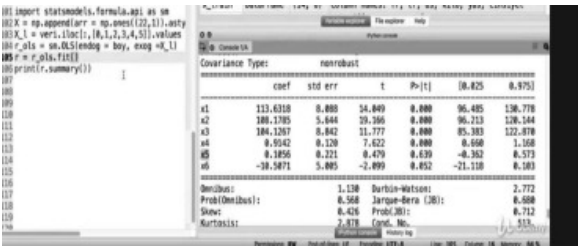
İhtiyacımız olan OLS Regresyon Sonuçlarını çıkarmak için bağımlı ve bağımız değişkenler arasındaki bağıntıyı çıkarmamız gerekiyor.

r_ols = sm.OLS(endog = boy, exog =X_l)

r = r_ols.fit()

print(r.summary())

OLS sonuçlarından p-değeri büyük olan örnek olara x5 (4.eleman) geri eleme algoritmasına göre elenmelidir.



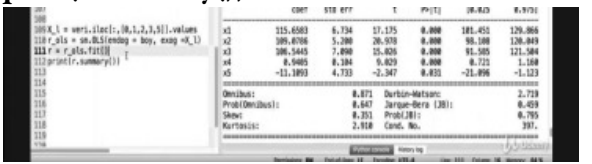
Bu işlemden sonra bu elemanı çıkartıp tekrar modeleme yapmalıyız.

X_l = veri.iloc[:,[0,1,2,3,5]].values

r_ols = sm.OLS(endog = boy, exog =X_l)

r = r_ols.fit()

print(r.summary())



0.031 kabul edilebilir bir değer(<0.05) olmasınının yanında isteğe bağlı olarak bu elemanıda çıkartmak mümkündür.

```
X_l = veri.iloc[:,[0,1,2,3]].values
```

```
r_ols = sm.OLS(endog = boy, exog =X_l)
```

```
r = r_ols.fit()
```

```
print(r.summary())
```

Geri işleme kısmımız buraya kadardı fakat gelecek derslerimizde hangi sistemin daha optimal

olduğuna R-squared ve F-statistic gibi önemli değerlerini incelediğimizde ulaşacağız.

Ders 32 : Ödev 1: Veri Kümesi ve Ödevin Açıklaması

http://www.bilkav.com/makine-ogrenmesi-egitimi/ [5]adresinden kullanacağımız veri kümesini indirdik.

Veri kümemizde hava durumu, sıcaklık,nemlilik,rüzgar durumu ve oyun değişkenleri içeriyor. Bu ödevde istenilen şey herhangi bir kolon için çoklu regresyon modeli geliştirmenizdir.

5. http://www.bilkav.com/makine-ogrenmesi-egitimi/

Ders 33 : Ödev 1: Çözüm 1.Parça: Verinin hazırlanması ve Çoklu Doğrusal Regresyon

Ödev çözümünde eğer rüzgarlılık veya oyun değişkenlerini Onehotencoding ile çevirirsek her bir değişken için ekstra kolon eklenecek ve bu da kukla değişkeni tuzağına sebep olacak. Bu nedenle bu değişkenleri Labelencoding yöntemiyle çevirmemiz yeterli olacaktır. Fakat hava durumu kolonundaki değişkenler sıraya sokulabilecek türden olmadıklarından dolayı bir problem yaratacaklardır. Bu nedenle ilk kolon Onehotencoding ile çevirirken son iki kolonu Labelencoding ile çeviriceğiz. Öncelikle play (oyun) kolonunu çeviriyoruz.

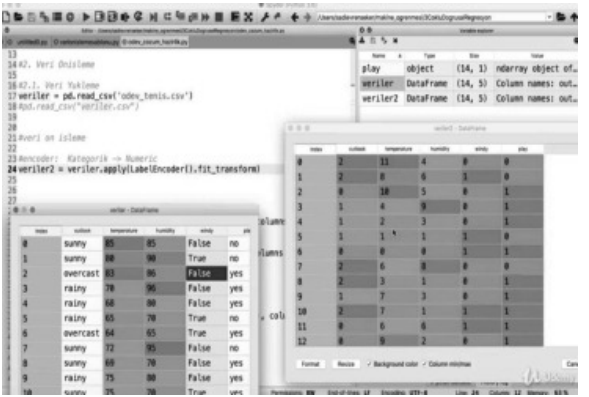
```
veriler = pd.read_csv('odev_tenis.csv')


#encoder: Kategorik -> Numeric
```

```
play = veriler.iloc[:,0:1].values

print(play)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

ulke[:,0] = le.fit_transform(play[:,0])

print(play)
```

Sonraki adım olarak rüzgarlılık kolonunu çeviriyoruz ve bunu aynı yöntemle yapmamız gerekmektedir .

Basitçe, play yerine windy koymak ve iloc için kolonu [:,-2:-1] olarak değiştirmemiz gerekmektedir.

Kısayol olarak “veriler2 = veriler.apply(LabelEncoder().fit_transform)” ile bütün kolonlar için encoding uygulayabiliriz.


#2. Veri Önişleme
#2.1. Veri Yükleme
veriler = pd.read_csv('odev_tenis.csv')
#pd.read_csv("veriler.csv")
#veri on isleme
#encoder: Kategorik -> Numeric
veriler2 = veriler.apply(LabelEncoder().fit_transform)
play object (14, 1) ndarray object of…
veriler DataFrame (14, 5) Column names: out…
veriler2 DataFrame (14, 5) Column names: out…

Ödevin devamı ve çoklu doğrusal regresyon modelinin oluşturulması öğrencilere bırakılmıştır.

Ders 34:Ödev 1: Çözüm 2.Parça: Geri Eleme (Backward Elimination)

Bağımlı değişkeni nemlilik(humidity) olarak belirliyoruz. Diğer kolonları encoding ile çevirip önceki dersimizde gördüğümüz tahmin yöntemiyle prediction yapmaya çalışıyoruz.

```
veriler = pd.read_csv('odev_tenis.csv')


#pd.read_csv("veriler.csv")
```

```
veriler2 = veriler.apply(LabelEncoder().fit_transform)


c = veriler2.iloc[:,:1]


from sklearn.preprocessing import OneHotEncoder


ohe = OneHotEncoder(categorical_features='all')


c=ohe.fit_transform(c).toarray()


print(c)
```

```
havadurumu = pd.DataFrame(data = c, index =
range(14), columns=['o','r','s'])


sonveriler = pd.concat([havadurumu,veril-
er.iloc[:,1:3]],axis = 1)

sonveriler = pd.concat([veriler2.iloc[:,-2:],son-
veriler], axis = 1)


from sklearn.cross_validation import
train_test_split


x_train, x_test,y_train,y_test =
train_test_split(sonveriler.iloc[:,:-1],sonveril-
er.iloc[:,-1:],test_size=0.33, random_state=0)
```

```
from sklearn.linear_model import Linear-
Regression

regressor = LinearRegression()

regressor.fit(x_train,y_train)

y_pred = regressor.predict(x_test)

print(y_pred)

#backward elimination
```

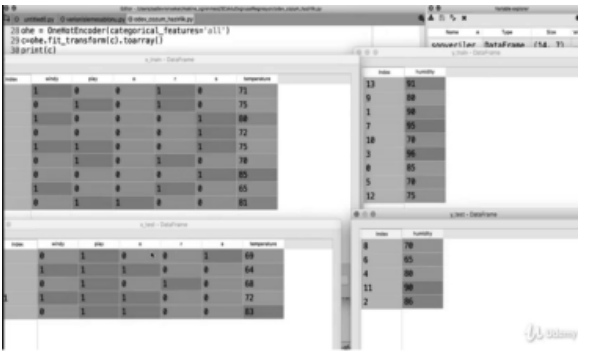
```
import statsmodels.formula.api as sm


X = np.append(arr = np.ones((14,1)).astype(int), values=sonveriler.iloc[:,:-1], axis=1 ) X_l = sonveriler.iloc[:,[0,1,2,3,4,5]].values


r_ols = sm.OLS(endog = sonveriler.iloc[:,-1:], exog =X_l)


r = r_ols.fit()


print(r.summary())


sonveriler = sonveriler.iloc[:,1:]
```

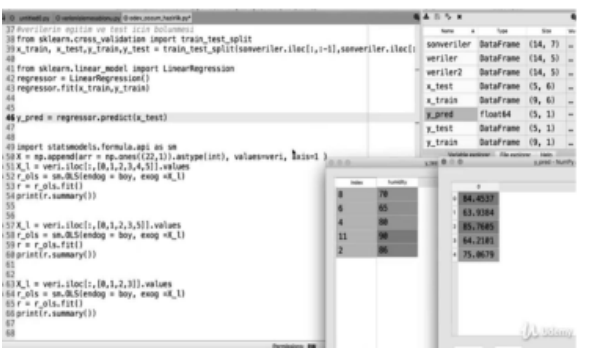
```
import statsmodels.formula.api as sm


X = np.append(arr = np.ones((14,1)).astype(int), values=sonveriler.iloc[:,:-1], axis=1 ) X_l = sonveriler.iloc[:,[0,1,2,3,4]].values


r_ols = sm.OLS(endog = sonveriler.iloc[:,-1:], exog =X_l)


r = r_ols.fit()


print(r.summary())
```

```
x_train = x_train.iloc[:,1:]


x_test = x_test.iloc[:,1:]


regressor.fit(x_train,y_train)


y_pred = regressor.predict(x_test)
```

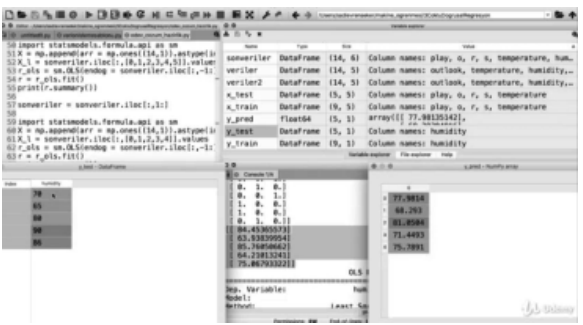
SADI EVREN SEKER AND FURKAN

180 GÜRÇAY





Backward Elimination(Geri Eleme) ile p-değeri yüksek bazı değişkenleri sistemden çıkaracağız. Böylece sistem daha hızlı çalışması dışında olasılık dahilinde tahmin doğruluğunda iyileşmeyi de mümkün kılacağız. Örneğin, geri eleme yaptıktan sonra tahminimiz gerçek veriye daha fazla yakınlaştığı görsellerden görülmekte.



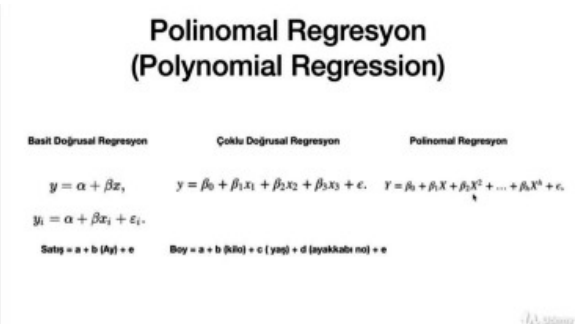
Bölüm 5:

Ders 35: Veri Kümesi , Kavramının ve Probleminin Tanımı (Polinomal Regresyona Giriş)

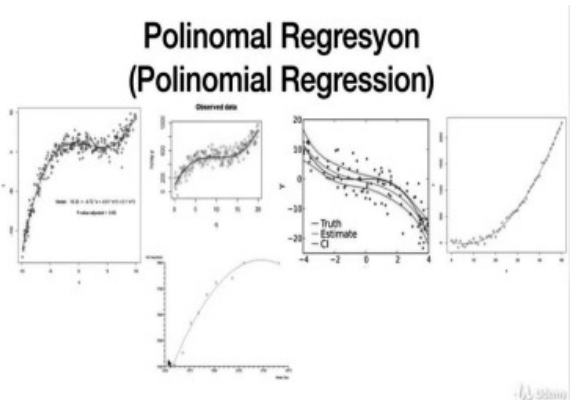
http://www.bilkav.com/makine-ogrenmesi-egitimi/ [6]adresinden maaslar.csv dosyasını indiriniz.

Polinomal regresyonun diğer regresyonlardan farkı, burada regresyonun derecesinin değişebilir olmasından dolayı hesaplamalarda regresyonun derecesinin de bizim için önemli olmasıdır.

6. http://www.bilkav.com/makine-ogrenmesi-egitimi/


Polinomal Regresyon
(Polynomial Regression)
Basit Doğrusal Regresyon
Çoklu Doğrusal Regresyon
Polinomal Regresyon
$y = \alpha + \beta x,$
$y_i = \alpha + \beta x_i + \varepsilon_i.$
$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon.$
$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \ldots + \beta_n X^n + \epsilon.$
Satış = a + b (Ay) + e
Boy = a + b (kilo) + c (yaş) + d (ayakkabı no) + e

Temelde bakıldığında çoklu doğrusal regresyon da birinci dereceden polinomal regresyona örnek gösterilebilir. Polinolmal Regresyon, sadece bir değişken belirlenip bu değişkenin üstlerinin alınıp değişik çarpanlarının bulunması değil birden fazla değişkenin üstlerinin ve hatta değişkenlerin kendi içinde çarpılıp değişik çarpanlarda çarpılması gibi bir denkleme sahiptir.



Polinomal Regresyon doğrusal regresyona göre verilerin oluşturduğu eğri (doğru olmayan yapı) üzerinde daha başarılı sonuçlar üretecektir.

Ders 36: Polinomal Regresyon'un Python ile Uygulama Kodu

Bu veri kümemizde oluşturmak istediğimiz regresyon, eğitim seviyesi ve maaş arasında olacaktır. Bu veri kümesindeki unvanları encode edersek her bir unvan için ayrı bir numara belirlenecek ve bu numaralar eğitim seviyesi kolonundan farklı olmayacaktı. Oluşturmak istediğimiz regresyon modelini daha iyi oluşturabilmek için elimizdeki bütün veriyi eğitim için kullanacağız. Sonrasında modelin başarısını göz ile gözlemleyeceğiz ve herhangi bir test sistemi geliştirmeyeceğiz. ".iloc" ile verimizdeki eğitim seviyesi kolonunu x, maaşları ise y değişkenine atıyoruz.

```
veriler = pd.read_csv('maaslar.csv')
```

```
x = veriler.iloc[:,1:2]
```

```
y = veriler.iloc[:,2:]
```

sklearn içerisinde dataframe'den kaynaklı çizim problemi olduğundan dolayı dataframe'i verilere çeviriyoruz.

```
X = x.values
```

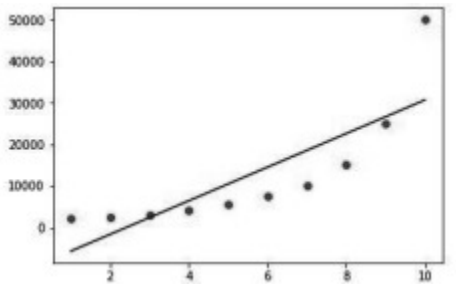
```
Y = y.values
```

Bildiğimiz gibi bu veriler düzgün bir doğru oluşturmadığından dolayı polinomal ilişkiye sokulabilecek veri kümesine örnek olarak göstermiştik. Polinomal ve doğrusal regresyon arasındaki farkı daha iyi kavramak adına oluşturacağımız koda doğrusal regresyonu da ekliyeceğiz.

```
#linear regression

from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()

lin_reg.fit(X,Y)
```

Doğrusal Regresyon Modelimizi daha iyi kavramak adına görselleştirme işlemi yapmamızda mümkün.

```
plt.scatter(X,Y,color='red')

plt.plot(x,lin_reg.predict(X), color = 'blue')

plt.show()
```


50000
40000
30000
20000
10000
0
2
4
6
8
10

Polinomal Regresyon oluşturmak için sci-kit learning'in içinden "PolynomialFeatures" import edeceğiz. Doğrusal Regresyondaki gibi obje oluşturup bir derece atıyoruz. Bu objenin özelliklerini kullanarak yine doğrusal regresyonu kullanabiliriz fakat öncesinde bu verileri polinomal dünyaya dönüştürüp sonrasında doğrusal regresyon için kullanabiliriz.

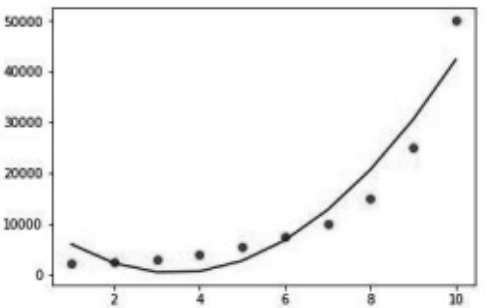
```
from sklearn.preprocessing import PolynomialFeatures

poly_reg = PolynomialFeatures(degree = 2)

x_poly = poly_reg.fit_transform(X)

print(x_poly)

lin_reg2 = LinearRegression()

lin_reg2.fit(x_poly,y)
```

Sonraki aşama olarak Regresyonumuzun çizimini yapacağız. Burada karşılaşacağımız önemli nokta ise doğrusal regresyonlar tahmin yapılmadan önce polinomal dönüşüm yapılma gereksinimidir.

```
plt.scatter(X,Y,color = 'red')
```

```
plt.plot(X,lin_reg2.predict(poly_reg.fit_transform(X)), color = 'blue')
```

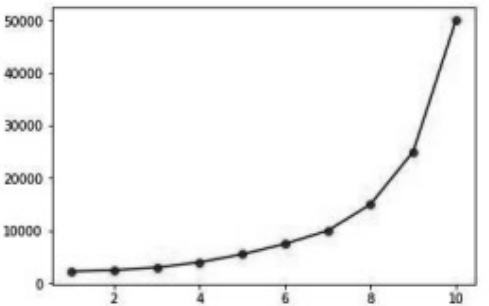
```
plt.show()
```


50000
40000
30000
20000
10000
0
2
4
6
8
10

Polinomal Regresyondaki atadığımız derecenin değişmini gözlemleyebilmek için farklı derece olan aynı kodu tekrar yazıyoruz.

```
from sklearn.preprocessing import PolynomialFeatures

poly_reg = PolynomialFeatures(degree = 10)

x_poly = poly_reg.fit_transform(X)

print(x_poly)

lin_reg2 = LinearRegression()

lin_reg2.fit(x_poly,y)

plt.scatter(X,Y,color = 'red')

plt.plot(X,lin_reg2.predict(poly_reg.fit_transform(X)), color = 'blue')
```

plt.show()



Görüldüğü üzere 4.dereceden olan regresyon modelimiz 2.dereceden olana göre daha başarılı sonuç vermiştir. Bu durumda, polinom derecesinin artışı daha başarılı sonuç alınmasında yardımcı olmuştur diyebiliriz fakat bu her veri için geçerli değildir bunun nedeni verilerin polinomal regresyon öğrenimi için özel seçilmiş olmasıdır. Bu verilerin özel olmasına rağmen gerçek hayatta da veriler gözlemlendiğinde verilerde doğrusallık ya da polinomallık olup olmadığını saptamak, kullanılacak olan model ile ilgili bize ipucu vermektedir. Herhangi bir tahmin yaparken ise "lin_reg.predict(birsayı(eğitim seviyesi))" kodunu çalıştırıyoruz.

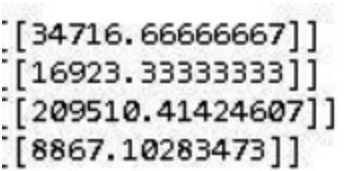
Örneğin:

```
print(lin_reg.predict(11))
```

```
print(lin_reg.predict(6.6))


print(lin_reg2.predict(poly_reg.fit_trans-
form(11)))


print(lin_reg2.predict(poly_reg.fit_trans-
form(6.6)))
```

İlk iki kod satırı doğrusal regresyon için yapıldığından dolayı 7.500-10.000 tl arası maaş alması gereken eğitim seviyesine sahip kişi için 16.923 gibi bir maaş belirlenmekte bunun nedeni ise ilk grafikteki gibi doğrunun normal maaşlardan daha yüksek seviyelerde bulunmasıdır.

Son iki kod satırında ise doğrusal regresyon tahmini yapmadan önce verileri polinomal dünyada dönüştürdüğümüzden dolayı daha makul sonuçlar elde etmekteyiz. Örneğin, bir önceki örnekteki eğitim seviyesine sahip bir kişi için 8867 tl gibi daha makul olan bir maaş belirlenmektedir.



Sonuç olarak yapılan işlem,verilerin durumuna bakılarak hangi regresyon modelinin kullanılması gerektiğini saptamak ve polinomal regresyon kullanmak için doğrusal regresyon bazında polinomal dünyada dönüştürülmüş veriler ile tahmin yapmaktır.

Ders 37: Python ile Doğrusal Olmayan Regresyon Şablon (Polinomal Regression Python Template)

Bu derste, önceki derste yazdığımız kodların daha stabil bir şekilde anlaşılabilmesi için doğrusal olmayan bir şablon oluşturacağız.

#Veri yükleme

veriler = pd.read_csv('maaslar.csv')

```
#data frame dilimlenmesi


x = veriler.iloc[:,1:2]


y = veriler.iloc[:,2:]
```

```
#NumPY dizi (array) dönüşümü


X = x.values


Y = y.values
```

```
#doğrusal regresyon

from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()

lin_reg.fit(X,Y)

#2.dereceden polinomal regresyon

from sklearn.preprocessing import PolynomialFeatures
```

```
poly_reg = PolynomialFeatures(degree = 2)

x_poly = poly_reg.fit_transform(X)

lin_reg2 = LinearRegression()

lin_reg2.fit(x_poly,y)

# 4.derecen polinomal regresyon

poly_reg 3= PolynomialFeatures(degree = 4)

x_poly 3= poly_reg3.fit_transform(X)
```

```
print(x_poly3)

lin_reg3 = LinearRegression()

lin_reg3.fit(x_poly,y)

# görselleştirme

plt.scatter(X,Y,color='red')

plt.plot(x,lin_reg.predict(X), color = 'blue')

plt.show()
```

```
plt.scatter(X,Y,color = 'red')


plt.plot(X,lin_reg2.predict(poly_reg.fit_trans-
form(X)), color = 'blue')


plt.show()


plt.scatter(X,Y,color = 'red')


plt.plot(X,lin_reg3.predict(poly_reg.fit3_transform(X)),
color = 'blue')


plt.show()
```

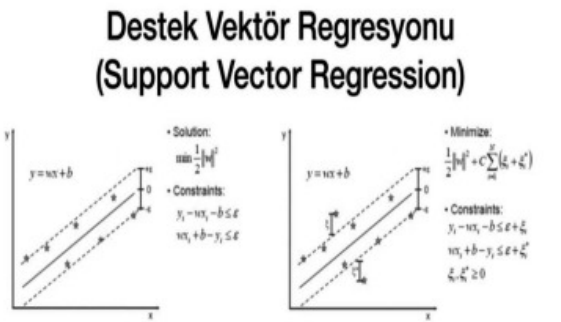
```
#tahminler


print(lin_reg.predict(11))


print(lin_reg.predict(6.6))


print(lin_reg2.predict(poly_reg.fit_trans-
form(11)))


print(lin_reg2.predict(poly_reg.fit_trans-
form(6.6)))
```

Bölüm 6:

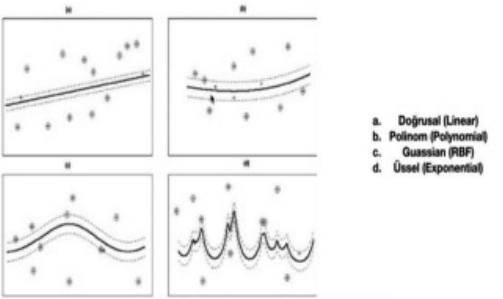
Ders 38: SVR (Destek Vektör Regresyonu) Tanıtım ve Problem

Tarihsel olarak baktığımızda, destek vektörleri regresyon olarak kullanılmadan önce 4.bölümde bahsedeceğimiz sınıflandırma için ortaya çıkmıştır. Asıl görevi doğrusal olarak birbirinden ayırılabilecek iki sınıfı birbirinden ayırmak için kullanılmıştır. Sınıflandırmak için kullanılacak olan doğrunun her iki tarafınıda bölüp maksimum marjini elde etmeye çalışmaktır. Bu işlem için birden fazla seçenek mevcuttur ancak hangi doğru en verimli şekilde ayırır sorusunun cevabı ise "destek vektör makinalarında maksimum marjini olan doğruyu seç" olarak cevaplanıyor. Regresyon modeli tabanındaki yaklaşımımız ise maksimum noktayı alabilen marjin aralığına sahip doğruyu seçmek olacaktır. Buradaki kilit noktalardan biri ise marjin değerini olabildiğince küçültmek olacaktır ve bunun için

marjini minimize eden doğruyu bulmak bize yardımcı olacaktır.



Kullandığımız fonksiyonları görselleştirmek gerekirse ortaya farklı eğimlerde doğrular çıkması kaçınılmazdır.



Dersi özetleyecek olursak, destek vektör regresyonu aslında marjin değerlerini tanımlıyor ve bu marjin değerlerine giren maksimum noktayı elde edebileceği minimum marjin değerine sahip marjini içeren fonksiyonu almayı amaçlamaktadır.

Ders 39: Python ile Destek Vektör Tahmini Uygulaması

İlk olarak yapmamız gerek şey verileri düzgün bir şekilde ölçeklemek olacaktır.

```
from sklearn.preprocessing import StandardScaler


sc1 = StandardScaler()


x_olcekli = sc1.fit_transform(X)


sc2 = StandardScaler()


y_olcekli = sc2.fit_transform(Y)
```

Bir sonraki görevimiz ise regresyon algoritmamızın objesini oluşturmak olacaktır.

```
svr_reg = SVR(kernel = 'rbf')
```

Bu kısmında SVR'ın parametresi olan kernel kısmı önemlidir. Burada varsayılan parametre olan 'rbf' (Radial Basis Function) tanımlı ve bu derste bu parametre kullanılacaktır, isteğe göre polinomal veya diğer parametreler kullanılabilir. Bir sonraki yapmamamız gereken işlem ise x_olcekli ile y_olcekli arasındaki bağlantıyı kurmak olacaktır.

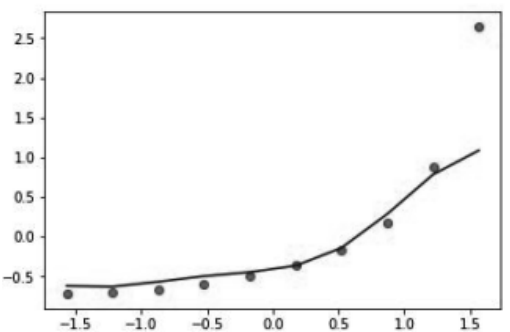
```
svr_reg.fit(x_olcekli,y_olcekli)
```

Bu bağlantıyı görselleştirmek mümkündür. Öncelikle x ve y boyutlarını oluşturmalıyız.

```
plt.scatter(x_olcekli,y_olcekli,color='red')
```

Sonrasında ise çizgimizi oluştururken “x_olcekli” için tahmin algoritmamızı çalıştırarak istenilen değer için tahminler yapılmasını sağlıyoruz.

```
plt.plot(x_olcekli,svr_reg.predict(x_olcekli),color='blue')
```



Öncesinde ölçekleme yapıldığı için -1.5 ile +1.5 arasında tahmin sonucumuzu görüyoruz.

Örnek tahminler olarak:

print(svr_reg.predict(11))

print(svr_reg.predict(6.6))

```
[0.01150915]
[0.01150915]
```

Dersimizi daha iyi kavramak adına 'rbf' yerine diğer kernel fonksiyonlar ile değiştirip farklı denemeler deneyebilirsiniz.

Bölüm 7:

Ders 40: Karar Ağacı Kullanarak Tahmin Yönetimi

Önceki derslerde yüklediğimiz veri kümemizi kullanacağız. Bu veri kümesinden ülke ve cinsiyet kolonlarını devre dışı bırakıyoruz. Karar ağacı, çoğunlukla sınıflandırma için kullanılan bir algoritma olmakla birlikte tahmin için de kullanılması mümkündür fakat en çok tercih edilen algoritma olarak adlandırmamız doğru olmaz.

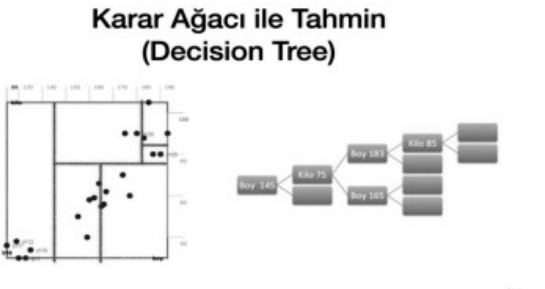
Elimizdeki problem ise boy ve kilo değişkenini kullanarak yaş değişkenini tahmin etmeye çalışmak olacaktır. Boy ve kilo değişkenini iki boyutlu uzaya dağıttığımızda aşağıdaki tablo ortaya çıkmaktadır.

Üçüncü boyutta da yaş değişkeni bulunacaktır.

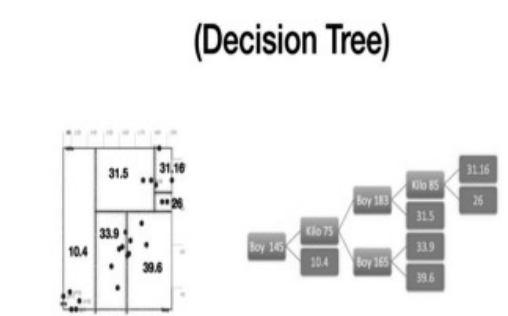
Bu metot ilk olarak uzayı ikiye bölerek bir karar ağacı oluşturmaktadır. Örnek olarak boyu 145'ten küçük ya da büyük olanlar olacaktır fakat bu bölme işleminin nereden bölüneceği sorusu önemlidir. Bu sorunun cevabı entropi ile ilgili ve termodinamiğe dayanıyor. Basit olarak baz alınan etken verilerin dağılımıdır. Bu konuya gelecek derslerdeki Sınıflandırma konusunda değineceğiz.

Boy bazında böldükten sonra bölmeye kilo bazında bölerek devam edebiliriz. Tablomuza göre sağ tarafımızı 75 kilo sınırından bölüyoruz fakat sol tarafı bölünemeyecek kadar düzenli olduğunu varsayıyoruz. Karar ağacımızı incelediğimizde boş yani label olmayan kutucuk-

ların karar verme sürecini etkilemediği ve bu kutucukların sayısı ile ağaçtaki bölgelerin eşit olduğu açıkça görülmektedir.



Asıl yapılmaya çalışılan işlem ise üçüncü boyuttaki veriyi ikinci boyuta ortalayarak yerleştirmek olacaktır. Bir diğer adım olarak her bölgenin yaş ortalamasını o bölgeye yazıyoruz ve karar ağacımıza son şeklini veriyoruz.



Bu ağaç bize öğrenme sürecimizin sona erdiğini göstermektedir. Bilindiği üzerine algoritmalarımızda üç aşama bulunmaktadır ve bunlar verilen veriler üzerinden öğrenme süreci, öğrenilen veriler üzerinden test edilmesi ile son aşama olan tahmin kısmıdır. Örneğin tabloya bakarak boyu 180 kilosu 80 olan kişi için karar ağacımız yaşı 31.5 olarak belirlemektedir.

Karar ağacı algoritması sadeliği ve amacına uygun formu nedeniyle yorumlamayı çoğunlukla tercih ettiği algoritmalardır. Bu nedenle Karar ağacı algoritmaları sadece sayısal tahmin ya da sınıflandırma için değil, bazen sadece veriyi görselleştirme amacıyla kullanılabilmektedir. Bu görselleştirme ile verinin uzayda dağılımı, hassas noktaları veya bölünme noktaları hakkında bilgilere ulaşmak mümkündür.

Ders 41: Python ile Karar Ağacı Kullanılarak Tahmin

İlk olarak sci-kit learn içinden "tree" kütüphanesindeki DecisionTreeRegressor'u import ediyoruz.

Sonrasında ise karar ağacı regresyonunu 'r_dt' objesi için atıyoruz.

```
r_dt = DecisionTreeRegressor(random_state=0)
```

random state kısmını varsayılan 0 değerine atamamızın karar ağacı oluşturmakta görevi vardır.

Sonrasında ise önceden kullandığımız unvan ve maaş değişkenlerini temsil eden X ve Y'yi eğitim için kullanacağız.

```
r_dt.fit(X,Y)
```

Kodumuzun devamında ise görselleştirme yöntemine başvuruyoruz.

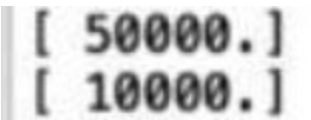
```
plt.scatter(X,Y, color='red')
```

```
plt.plot(x,r_dt.predict(X), color='blue')
```

Bir diğer alışılagelmiş adım olan tahmin işlemimizi de karar ağacımız yardımıyla gerçekleştiriyoruz.

```
print(r_dt.predict(11))
```

```
print(r_dt.predict(6.6))
```



Eğer ki X değerimizi 0.5 arttırır ya da 0.4 azaltırsak göreceğimiz değerler aynı olacaktır. Örneğin 0.6 , 1 ve 1.5 değeri için 2250 değeri döndürülecektir ve orijinal eğrimiz ile örnek veri eğrimiz üstüste olacaktır.

Z=X+0.5

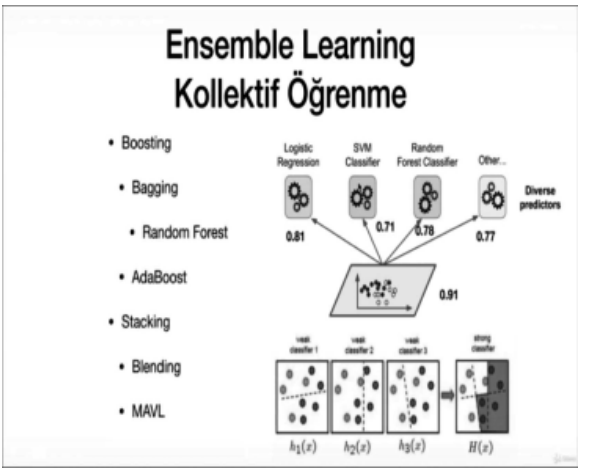
K=X-0.4

```
from sklearn.tree import DecisionTreeRegressor
r_dt = DecisionTreeRegressor(random_state=0)
r_dt.fit(X,Y)
Z = X + 0.5
K = X - 0.4
plt.scatter(X,Y, color='red')
plt.plot(x,r_dt.predict(X), color='blue')
plt.plot(x,r_dt.predict(Z),color='green')
plt.plot(x,r_dt.predict(K), color = 'yellow')
print(r_dt.predict(11))
print(r_dt.predict(6.6))
```

Bölüm 8:

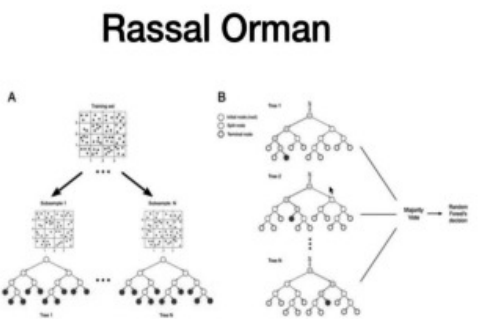
Ders 42: Rassal Ağaç (Random Forest) Algoritması ve Tahmin

Öncelikle kollektif öğrenme(Ensemble Learning) kavramından bahsedeceğiz. Kollektif öğrenme, birden fazla tahmin algoritması aynı anda kullanılarak daha başarılı bir sonuç çıkartılmasını sağlayan metot için kullanılmaktadır. Bu yöntem sınıflandırma veya tahmin algoritmalarında kullanılmaktadır. Bu metodun içindeki ilk göreceğimiz yöntemlerden birisi ise Rassal Ağaç algoritması olacaktır.



Rassal Ağaçlar(Orman) algoritması aslında birden fazla karar ağacının aynı problem için aynı veri kümesi üzerinde çizilmesi ve problem çözümünde hep birlikte kullanılması durumudur. Bu yöntem bazı kaynaklarda "çoğunluğun oyu (majority vote)" olarak da geçmektedir. Normal koşullarda veri kümemiz eğitim ve test olarak bölünmektedir. Önceki eğitimlerde görülen üzerine Rassal Ağaçlar da eğitilebilmek-

te fakat eğitilirken eğitim kümesi, alt küçük parçalara bölünerek veri kümesindeki bütün verileri bir algoritma için kullanmıyor. Bu sayede Rassal Ağaçlar birden fazla karar ağacını farklı veri kümelerinden öğrenmektedir. Aksi halde aynı karar ağaçları ortaya çıkacaktır. Rassal Orman ise bu farklı karar ağaçlarını topluyor ve Rassal Ağaç Algoritmasına gönderiyor. Burada Çoğunluğun Oy'u denilen yönteme tabi tutuyor.



Bu yöntem farklı şekillerde kullanılmaktadır. Sınıflandırma durumunda, örneğin kişinin boyundan ve kilosundan cinsiyetini tahmin ederken sonuç için çoğunluğun oyuna bakılmaktadır fakat eğer tahmin söz konusu ise bu durumda ortalama değer göz önünde bulundurulmaktadır.

Sanılanın aksine karar ağaçlarının veri kümesinin parçalarından beslenmesi onların başarısına olumlu yönde katkı sağlamaktadır, aksi halde aşırı öğrenme tehlikesi veya ağacın fazla büyümesi halinde hesaplama zamanının artması gibi tehlikeli durumlar ortaya çıkmaktadır. Rassal Orman'ın altında yatan genel mantık ise, daha az veri kullanılarak karar ağaçları içerisinde çeşitlilik elde etmek ve birden fazla bakış açısına ulaşmaktır.

Ders 43: Python ile Rassal Ağaç Kullanılarak Tahmin

İlk olarak Rassal Ağaç Regresyon'unu sci-kit learn'in ensemble kütüphanesinden import ediyoruz. Önceki regresyon modelleriyle benzer bir yol izleyerek regresyon için obje oluşturuyoruz.

```
from sklearn.ensemble import RandomForestRegressor

rf_reg = RandomForestRegressor(n_estimators = 10, random_state=0)
```

Burada estimators, oluşturmak istediğimiz karar ağaçlarının sayısını temsil etmek ve random_state parametresi ise genel olarak ağaç oluşturulmasında görev almaktadır.

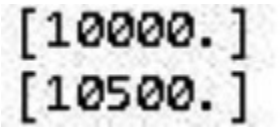
Sonraki adım olarak eğitim işlemi ile veriler arasındaki bağlantı oluşturulmaktadır.

```
rf_reg.fit(X,Y)
```

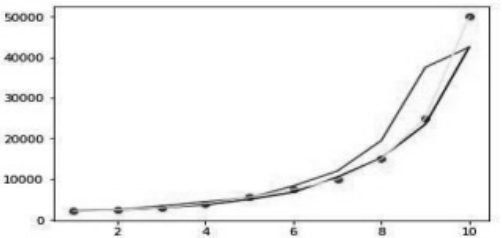
Son olarak tahmin işlemimizi yapıyoruz. Aradaki farkı anlamak için önceki regresyondaki tahminin aynısını kullanıyoruz.

```
print(rf_reg.predict(6.6))
```

Sonuç olarak karar ağacı regresyonunda sonuçlar önceden tanımlanan değerlerin dışına çıkmazken Rassal orman regresyonunda bu değer değişmekte ve önceki sonuç (10000),bu regresyonda 10500 olarak bulunmaktadır.



Görselleştirme işlemi için önceki yöntemimizi kullanıyoruz fakat burada sadece X tabanında kalmak yerine Z ve K değerleri için de tahmin yaptırılıp grafik üzerindeki farklılıklar açıkça gözlemlenmektedir.


50000
40000
30000
20000
10000
0
2
4
6
8
10

Bilinmeyen veriler söz konusu olduğunda Karar ağaçları bildiği veriler ile aynı karar ağaçlarını döndürme eğilimindedir. Rassal Orman ise her bir veri için yeni ve orijinal bir değer döndürme kapasitesine sahip olduğundan dolayı gerçek hayat problemlerinde daha fazla tercih edilen yöntemdir.

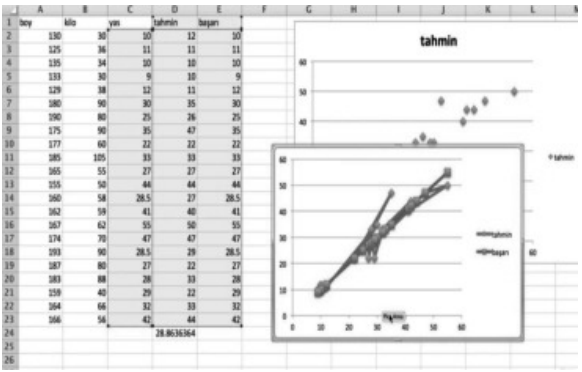
Bölüm 9:

Ders 44: R2 Hesaplanması

R2 (R-Square, R-Kare) yöntemi, daha önce geliştirilmiş tahmin metotlarının ölçümlenmesi için sık kullanılan bir yöntemdir. Ayrıca, farklı regresyon ile oluşturulmuş metotlar arasında seçim yapmamıza yardımcı olmaktadır. R-Kare

yönetimi temelde iki değer üzerine kurulmuştur. Bu değerler, Hata Kareleri Toplamı ve Ortalama Farkların Toplamı'dır. HKT, orijinal veriler ile tahmin değerleri arasındaki farkın karesini temsil etmektedir. OFT ise orijinal veriler ile ortalama tahmin değerlerinin farkının karesini incelemek için kullanılmaktadır. Bu miktarların değerlerinin pozitif çıkması için kare alma işlemini kullanmaktayız çünkü tahmin ve orijinal veri arasındaki fark pozitif oluşabileceği gibi negatifte çıkabilir.

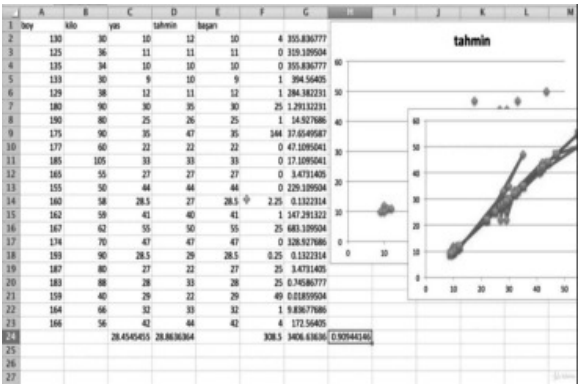
Örnek olarak, örnek excel dosyasındaki yaş ve tahmin değerlerinden türeyen başarı durumunu "Straight Marked Scatter" ile çiziyoruz.



	boy	kilo	yas	tahmin	basari
2		130	30	10	12	10
3		125	36	11	11	11
4		135	34	10	10	10
5		133	30	9	10	9
6		129	38	12	11	12
7		180	90	30	35	30
8		190	80	25	26	25
9		175	90	35	47	35
10		177	60	22	22	22
11		185	105	33	33	33
12		165	55	27	27	27
13		155	50	44	44	44
14		160	58	28.5	27	28.5
15		162	59	41	40	41
16		167	62	55	50	55
17		174	70	47	47	47
18		193	90	28.5	29	28.5
19		187	80	27	22	27
20		183	88	28	33	28
21		159	40	29	22	29
22		164	66	32	33	32
23		166	56	42	44	42
24				28.8636364		

Tabloda gördüğümüz kırmızı renkteki noktalar başarılı tahmini, mavi olanlar ise başarısız tahminleri temsil etmektedir. Kırmızı noktalar başarıyı temsil ettiğinden dolayı aynı doğru üzerinde görüntülenmektedir. Buradaki asıl amacımız başarı durumundaki sapmayı ölçümlemektir. Farklı tahmin modellerinde farklı tah-

min değerleri oluşacağından dolayı bu modellerdeki hata oranını ölçebilecek metodu R-Kare yöntemiyle oluşturmamız gerekmektedir. R-Kare için kabul edilebilen taban ve tavan değerler 0 v 1'dir. Örneğin hiç hata oluşmama durumunda HKT 0 olacağından dolayı R-Kare 1 olurken, eğer ortalama yaş değerine göre bir tahmin yapılırsa R-Kare 0 olarak gözlenmektedir. Eksi değerde R-Kare oluşması algoritmamızın kabul edilebilir durumdan bile daha işlevsiz olduğunu göstermektedir. Genel olarak R-Kare değerinin 1'e yakın bir değer olarak elde edilmesi, algoritmamızın doğruluğunun geçerliliğini arttırmaktadır.



boy	kilo	yas	tahmin	basarı			
130	30	10	12	10	4	355.836777	
125	36	11	11	11	0	319.109504	
135	34	10	10	10	0	355.836777	
133	30	9	10	9	1	394.56405	
129	38	12	11	12	1	284.382231	
180	90	30	35	30	25	1.29132231	
190	80	25	26	25	1	14.927686	
175	90	35	47	35	144	37.6549587	
177	60	22	22	22	0	47.1095041	
185	105	33	33	33	0	17.1095041	
165	55	27	27	27	0	3.4731405	
155	50	44	44	44	0	229.109504	
160	58	28.5	27	28.5	2.25	0.1322314	
162	59	41	40	41	1	147.291322	
167	62	55	50	55	25	683.109504	
174	70	47	47	47	0	328.927686	
193	90	28.5	29	28.5	0.25	0.1322314	
187	80	27	22	27	25	3.4731405	
183	88	28	33	28	25	0.74586777	
159	40	29	22	29	49	0.01859504	
164	66	32	33	32	1	9.83677686	
166	56	42	44	42	4	172.56405	
		28.4545455	28.8636364		308.5	3406.63636	0.90944146

Tabloda basit excel işlemleri yaparak ortalama tahmin, ortalama yaş, hata kare farkı toplamı ve ortalama fark karelerinin elde edebiliyoruz. Ayrıca formülümüzden yola çıkacak olursak R-Kare değerini 0.90944146 gibi bir değer olarak bulmaktayız.

Bu değer 1'e yakın olduğundan dolayı bu tahmin metodu için tatmin edici diyebiliriz. Sonuç olarak farklı tahmin metotlarının R-Kare değerlerini birbiriyle karşılaştırıp en az hata değerine

sahip metodu saptamak R-Kare yöntemiyle mümkündür.

Ders 45: Düzeltilmiş R2 Hesaplanması (Adjusted R2)

Önceki derste bahsettiğimiz R2-Kare yöntemi bazı problemler barındırmaktadır. Bu derste bu problemlerin çözümünden ve hangi durumlarda ortaya çıkabileceklerinden bahsedilmektedir. R-Kare yöntemi bazı durumlarda modelimiz üzerindeki yarattığımız pozitif etkiyi görmemizi engellediği ve bununla birlikte olumsuz bir etkiye sahip olduğundan dolayı alternatifi olan Düzeltilmiş R2 yöntemine ihtiyaç duymamıza neden olmaktadır.

Bilindiği üzere doğrusal regresyon modelinde istenildiği kadar değişken ekleme olanağına sahiptik. Bu model ile R-Kare değerini hesaplamamız da mümkündür. Bilindiği üzere doğrusal regresyonda değişkenlerin farklı çarpanlarla çarpılıp sabit bir değerlere toplanmaktadır. Örneğin R-Kare değerimizi iyileştirmek için maaş ve eğitim seviyesinin yanı sıra kişinin performans kriterini de denklemimize ekliyoruz. Bu kısımdaki problem, eklenen yeni değişkenin R-Kare değeri üzerinde olumlu etki yaratabilirken buna karşın olumsuz etki yaratamama durumudur.

Yöntemleri Karşılaştırmak

- R^2 (R- Square , R-Kare) Yöntemi
- Düzeltilmiş R^2 (Adjusted R^2) Yöntemi

Doğrusal Regresyon

$$y=a_0x_0+a_1x_1+b$$

$$y=a_0x_0+a_1x_1+a_2x_2+b$$

Hata Karaleri Toplamı = $\text{Topla}(y_i - y')^2$

Ortalama Farkların Toplamı = $\text{Topla}(y_i - y_{ort})^2$

$$R^2 = 1 - \frac{HKT}{OFT}$$

Düzeltilmiş $R^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$

Örneğin tablodaki gibi yeni değişkenin olumsuz etki yaratması durumunda a2 çarpanı sıfır olacaktır ve ilk durum arasındaki fark ortadan kalkacaktır. Bir diğer problem ise gerçek hayat özelinde ve nadir durumlar dışında çarpanların sıfır olması mümkün değildir. Bu nedenlerden dolayı istatiksel modelleme oluşturmadan önce veriyi önişlemeden geçiriyoruz. Tablodaki gibi Düzeltilmiş R-Kare değerini formüle ederken n değeri eleman sayısını p ise değişken sayısını sembolize etmektedir. Bu formül ile değişken sayısı arttıkça R-Kare'nin olumsuz etkisinin azaldığı bir yöntem oluşturmaktayız. Sonuç olarak, R-Kare ve Düzeltilmiş R-Kare yöntemleri kendi özellerinde farklı dezavantajlara sahip olsalar da doğru yerde kullanılmaları sistemin yararına olmaktadır.

Ders 46: Python ile R2 Hesaplama ve Algoritma Karşılaştırılması

Kodlama aşamasında ki kullanacağımız kütüphane sci-kit learn içindeki r2_score kütüphanesidir.

```
from sklearn.metrics import r2_score
```

R-Kare hesaplanması için ihtiyacımız olan değerler orijinal veri ve tahmin değeridir. Önceki derslerimizde bulduğumuz maaş lar ile eğitim seviyesine göre yaptığımız tahminleri kıyaslıyoruz.

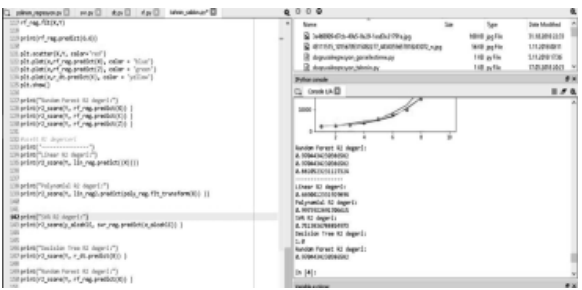
```
print("Random Forest R2 degeri:")

print(r2_score(Y, rf_reg.predict(X)) )

print(r2_score(Y, rf_reg.predict(K)) )

print(r2_score(Y, rf_reg.predict(Z)) )

print("Linear R2 degeri:")
```

```
print(r2_score(Y, lin_reg.predict((X))))

print("Polynomial R2 degeri:")

print(r2_score(Y, lin_reg2.predict(poly_reg.fit_transform(X)) ))

print("SVR R2 degeri:")

print(r2_score(y_olcekli, svr_reg.predict(x_olcekli)) )

print("Decision Tree R2 degeri:")
```

```
print(r2_score(Y, r_dt.predict(X)) )
```

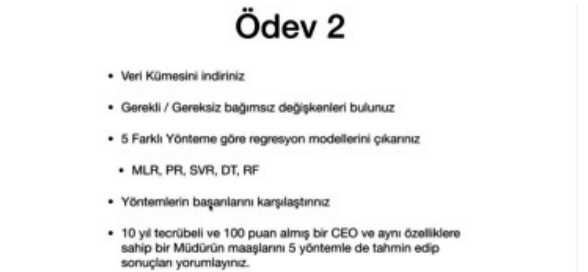
```
print("Random Forest R2 degeri:")
```

```
print(r2_score(Y, rf_reg.predict(X)) )
```



Sonuçlardan da görülebileceği gibi ilk bakışta Karar Ağacı Regresyonu'nun R-Kare değeri en iyi ve mükemmel olarak görülebilse de kendisi, algoritmanın mantığı bilinmeden sadece tek bir değere bakarak regresyon seçiminin ne denli hatalı bir işlem olduğunu bize ilk elden göstermektedir. Çünkü karar ağacı her değer için sabit bir değer aralığını döndürmekteydi. Eğer bir makine, makine öğrenmesi yapabilseydi bu hatayı göremeyecekti. Sonuç olarak, Karar Ağacı Regresyonu'nun sonucunu ihmal ettiğimizden dolayı Polinomal Regresyon ve Rassal Ağaca Regresyon'u ilk öne çıkan sonuçlar olmaktadır. SVR'da "rbf" yerine diğer değişkenleri kullanmamız farklı sonuçlar üretirken beklendiği üzerine Doğrusal regresyon en kötü sonucu oluşturmaktadır. Fakat bu sonuçlar bizi yanlış yönlendirmemeli çünkü farklı durumlarda farklı modeller bazında avantajlı olma durumu değişkenlik gösterir.

Ders 47: Ödev 2 : Ödev Tanımı ve Veri Kümesi



Veri kümesini(maaslar_yeni.csv) http://www.bilkav.com/makine-ogrenmesi-egitimi/ [7]adresinden indirebilirsiniz

7. http://www.bilkav.com/makine-ogrenmesi-egitimi/

Gerekli/Gereksiz bağımsız değişkenleri p-value ile bulmak daha hızlı bir yöntem olacaktır.

Daha önce öğrendiğimiz Regresyonların modellerini yeni veri kümemiz ile çıkartıyoruz.

Yöntem başarılarını R-kare yada Düzeltilmiş A-Kare yöntemi ile bulmak mümkünür.

Son olarak verilen örnek ile oluşturduğunuz modeller ile tahmin yapmanız ve bu tahmin sonuçlarını karşılaştırıp yorumlamanız beklenmektedir.

Ders 48: Ödev 2: Çözümü

Veri kümemizi indirdikten sonra veriyi inceliyoruz. İlk olarak "maaslar.csv" yerine

“maaslar_yeni.csv” yazıyoruz. Görülebildiği üzerine bu veri kümesinde daha fazla kolon bulunmakta.

```
veriler = pd.read_csv('maaslar_yeni.csv')
```

Yeni verimiz için ilk öğrenmemiz gerek şey ID kolonunu ihmal etmemiz olduğudur çünkü bu değişken ezberlemeye sebep olabilmektedir. Bir başka bilinmesi gereken şey ise unvan kolonu da ihmal edilemesi gerektiğidir çünkü bu veri setinde "unvan" kukla değişkeni gibi hareket etmektedir. Eğer ki bu veride unvan seviyesi değişkeni olmasaydı labelencoding ile unvan değişkeni encode edilmesi gerekebilirdi.

Buradaki amacımız bağımlı değişken olan maaşı, bağımsız değişkenlerden olan unvan seviyesi,kıdem ve puan ile tahmin etmektir. Bunun için önce bağımlı ve bağımsız değişkenleri tanımlamamız gerekmektedir.

```
x = veriler.iloc[:,2:5]
```

```
y = veriler.iloc[:,5:]

X = x.values

Y = y.values
```

İlk olarak P-değerini bulmak için Doğrusal Regresyon üzerinden bir model oluşturup p-değerini çıkartıyoruz.

```
#Doğrusal Regresyon OLS

import statsmodels.api as sm
```

model = sm.OLS(lin_reg.predict(X),X)

print(model.fit().summary())

	Calisan ID	UnvanSeviyesi	Kidem	Puan	maas
Calisan ID	1.000000	0.331847	0.206278	-0.251278	0.226287
UnvanSeviyesi	0.331847	1.000000	-0.125200	0.034948	0.727036
Kidem	0.206278	-0.125200	1.000000	0.322796	0.117964
Puan	-0.251278	0.034948	0.322796	1.000000	0.201474
maas	0.226287	0.727036	0.117964	0.201474	1.000000

OLS Regression Results

Dep. Variable:	y	R-squared:	0.903
Model:	OLS	Adj. R-squared:	0.892
Method:	Least Squares	F-statistic:	83.89
Date:	Mon, 10 Dec 2018	Prob (F-statistic):	8.38e-14
Time:	14:11:32	Log-Likelihood:	-295.74
No. Observations:	30	AIC:	597.5
Df Residuals:	27	BIC:	601.7
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	2494.8107	256.145	9.740	0.000	1969.244	3020.377
x2	1.3531	318.990	0.004	0.997	-653.161	655.867
x3	-26.5687	33.657	-0.789	0.437	-95.626	42.489

Sonuçta da görebileceğiniz gibi x2 ve x3'ün p-değerleri fazlasıyla yüksek (>0.05) çıkmıştır.

Sadece x1 ile doğrusal regresyon oluşturmak mümkündür.

x = veriler.iloc[:,2:3]

OLS Regression Results

Dep. Variable:	y	R-squared:	0.942
Model:	OLS	Adj. R-squared:	0.940
Method:	Least Squares	F-statistic:	468.1
Date:	Sun, 01 Apr 2018	Prob (F-statistic):	1.93e-19
Time:	19:01:55	Log-Likelihood:	-287.43
No. Observations:	30	AIC:	576.9
Df Residuals:	29	BIC:	578.3
Df Model:	1		
Covariance Type:	nonrobust		

Görüldüğü üzere yüksek P-değerine sahip bağımsız değişkenleri sistemden çıkartarak R-Kare değerinde bir artış sağlamış olduk. Aynı işlemi diğer Regresyon modellemelerinde de kullanabilirsiniz. Öncelikle 3 değişkenle sonrasında ise bu değişkenleri P-değerlerinin durumuna göre azaltarak OLS sonuçlarına bakabilirsiniz.

```
#Polinomal Regresyon OLS

print('poly ols')

model2 = sm.OLS(lin_reg2.predict(poly_reg.fit_transform(X)),X)

print(model2.fit().summary())

#Destek Vektör Regresyon OLS

print('svr ols')

model3 = sm.OLS(svr_reg.predict(x_olcekli),x_olcekli)
```

```
print(model3.fit().summary())
```

```
#Karar Ağacı Regresyonu OLS

print('dt ols')

model4 = sm.OLS(r_dt.predict(X),X)

print(model4.fit().summary())

#Rassal Orman Regresyonu OLS

print('rf ols')

model5 = sm.OLS(rf_reg.predict(X),X)
```

```
print(model5.fit().summary())
```

Daha sağlıklı sonuçlar için önceki tahmin satırlarımızı ve Polinomal Regresyon'daki 2.dereceden polinomu kaldırıyoruz. Örneğin çoklu parametreleri kullandığımız PR'da x2 ve x3 p-değerleri yüksek olsa da sisteme olumlu etki yaptıklarını görebiliyoruz. SVR bazında çoklu parametre ile R-kare değerinin yükseldiğini görmekteyiz,bunun nedeni ise diğer regresyonların aksine bu regresyon modellemesinde ölçekleme yapmamızın bir getirisidir. Karar Ağacı ve Rassal Orman'a bakınca R-Kare değerinde düşüklük

gözlemlenebilmektedir.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Apr  1 19:19:32 2018

@author: sadievrenseker
"""

Tek Parametreli olarak
linear
R-squared:                      0.942

Polynomial
R-squared:                      0.759

SVR
R-squared:                      0.770

DT
R-squared:                      0.751


RF
R-squared:                      0.719

```

```
Tek Parametreli olarak
linear
R-squared:                    0.942

Polynomial
R-squared:                    0.759

SVR
R-squared:                    0.770

DT
R-squared:                    0.751

RF
R-squared:                    0.719


3 parametreli
linear
R-squared:                    0.903

poly
R-squared:                    0.680

svr
R-squared:                    0.782

dt
R-squared:                    0.679


rf
R-squared:                    0.713
```

Ödeve ilave olarak pandas kütüphanesinden üretilmiş olana "veriler" dataframe'in ".corr" fonksiyonu ile değişkenler arasındaki korelasyonu(bağıntıyı) üretebilmemiz mümkündür.

```
x = x.values
Y = y.values

print(veriler.corr())

#Linear regression
from sklearn.linear_model import Lin
lin_reg = LinearRegression()
lin_reg.fit(X,Y)
```

```
In [156]: runfile('/Users/sadievrenseker/makine_ogrenmesi/BTahminSablonOdev/
tahmin_sablon.py', wdir='/Users/sadievrenseker/makine_ogrenmesi/
BTahminSablonOdev')
              Calisan ID  UnvanSeviyesi     Kiden      Puan      maas
Calisan ID      1.000000       0.331847  0.206278 -0.251278  0.226287
UnvanSeviyesi   0.331847       1.000000 -0.125200  0.034948  0.727836
Kiden           0.206278      -0.125200  1.000000  0.322796  0.117964
Puan           -0.251278       0.034948  0.322796  1.000000  0.201474
maas            0.226287       0.727836  0.117964  0.201474  1.000000
                                OLS Regression Results
```

Bu tabloya bakarak hangi değişkenlerin diğer değişkenlerin tahmini üzerindeki etkisini kolayca görebilmekteyiz.

Ders 49: Tahmin Metotlarının Karşılaştırılması

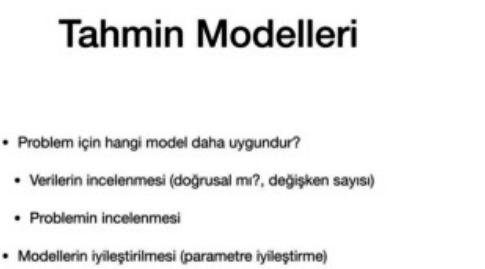
Bu dersimize kadar farklı Regresyon modellerini inceledik. Her bir model kendi özünde bazı avantaj ve dezavantajlara sahip olduğunu öğrendik. Bu artı ve eksi değerleri tek bir görselde toplayarak öğrenimimizi daha iyi pekiştirmeliyiz.

Tahmin Modelleri

Model	Artılar	Eksileri
Linear Regression	Veri boyutundan bağımsız olarak doğrusal ilişki üzerine kuruludur	Doğrusallık kabulü aynı zamanda hatadır
Polynomial Regression	Doğrusal olmayan problemleri adresler	Başarı için doğru polinom derecesi önemlidir
SVR	Doğrusal olmayan modellerde çalışır, marjinal değerlere karşı ölçekleme ile dayanıklı olur	Ölçekleme önemlidir, anlaşılması nispeten karışıktır, doğru kernel fonksiyonu seçimi önemlidir
Decision Tree Regression	anlaşılabilirdir, ölçeklemeye ihtiyaç duymaz, doğrusal veya doğrusal olmayan problemlerde çalışır	Sonuçlar sabitlenmiştir, küçük veri kümelerinde ezberleme olması yüksek ihtimaldir
Random Forest	ölçeklemeye ihtiyaç duymaz, doğrusal veya doğrusal olmayan problemlerde çalışır, ezber ve sabit sonuç riski düşüktür	Çıktıların yorumu ve görsellemesi nispeten zordur

Modellerin karşılaştırılmasından sonra üzerinde düşünmemiz gereken faktör ise modellerin, halihazırdaki problemimiz için bu modellerin gerektirdiği uygunluk durumudur. Bu durumu verilerin incelenmesi ve sorasında problemin incelenmesi ile giderebiliriz. Bir sonraki işlem olarak modellerin iyileştirilmesi aşaması yer almaktadır.

Örnek olarak Polinom Regresyonu'nda kaçıncı dereceden polinom alınacağı, Çoklu Doğrusal Regresyon'da hangi parametrelerin sisteme dahil edileceği ya da Rassal Orman'da kaç karar ağacı olacağı önemli kriterlerden birkaçıdır.



Bu ders ile birlikte tahmin problemleri için uçtan uca bir çözüm gerçekleştirebilecek bir seviyeye geldiğimizi söyleyebiliriz. Tahmin ile ilgili bölümlerimizin sonuna gelmiş bulunmaktayız bir sonraki ders olarak Sınıflandırma yöntemlerine giriş yapacağız.

Ders 50: Tahmin Metotları için Kullanılabilecek Python Şablonu

Destek Vektör Regresyonu ve Ölçekleme (Support Vector Regression , Scaling)

Kendi Notlarınız Ek

```
from sklearn.preprocessing import StandardScaler
sc1 = StandardScaler()
x_olcekli = sc1.fit_transform(x)
sc2 = StandardScaler()
y_olcekli = sc2.fit_transform(y)
from sklearn.svm import SVR
svr_reg = SVR(kernel = 'rbf')
svr_reg.fit(x_olcekli,y_olcekli)
model5 = sc2.inverse_transform(svr_reg.predict(x_olcekli))
print(r2_score(y_olcekli, svr_reg.predict(x_olcekli)))
```

Destek vektör regresyonunun en önemli özelliği, marjinal verilere karşı hassas olmasıdır. Bu yüzden ve verilerin daha iyi tahminin sağlamak için, öncelikle standartlaştırma yapılması gerekir. Yukarıdaki kodun ilk parçasında, StandardScaler sınıfından türetilmiş sc1 ve sc2 nesneleri sayesinde hem özellik hem de hedef değişken ölçeklenmektedir. Ardından SVR sınıfından rbf çekirdeği (kernel) ile üretilen svr_reg isimli nesne ile tahmin modeli oluşturulmaktadır, bunun için de ölçeklenmiş eğitim verileri fit metodu kullanılmaktadır.

Karar Ağacı ile Tahmin (Decision Tree)

```
from sklearn.tree import DecisionTreeRegressor
r_dt = DecisionTreeRegressor(random_state=0)
r_dt.fit(X,Y)
print(r_dt.predict(K))
model2 = sc2.inverse_transform(r_dt.predict(K))
print(model2.fit(X,r_dt.predict(K)))
print("Decision Tree R2 değeri: ")
print(r2_score(Y, r_dt.predict(K)))
```

Yapı olarak buraya kadar kullanılan sınıflandırma algoritmalarından pek de farklı olmayan karar ağacı sınıflandırması, DecisionTreeRegressor sınıfından türetilmiş ve yine X ve Y düzleri üzerinde fit() metodu ile bir model inşası için kullanılmıştır.

Rassal Orman (Random Forest)

Kendi Notlarınız Ek

```
from sklearn.ensemble import RandomForestRegressor
rf_reg = RandomForestRegressor(n_estimators = 10, random_state=0)
rf_reg.fit(X,Y)
print(rf_reg.predict(K))
model3 = sc2.inverse_transform(rf_reg.predict(K))
print("Random Forest R2 değeri: ")
print(r2_score(Y, rf_reg.predict(K)))
```

Ait yapısında karar ağacı kullanan rassal ormanlar (random forest), şimdiye kadar elde ettiğimiz puanya çok benzer şekilde, sklearn kütüphanesi içerisinden bir sınıf alarak RandomForestRegressor eklemiş (import) ve bu sınıftan da ürettiğimiz nesne ile fit() ve predict() metotlarını kullanarak makine öğrenmesini gerçekleştirmiştir.

Neler Öğrendik:

1. Sci-Kit Learn kütüphanesinin genel akışını ve fit(), predict()gibi metotlarını genel kullanımını

2. Doğrusal regresyon, Polinomla regresyon, destek vektör regresyonu, karar ağacı regresyonu ve rassal orman regresyonu metotlarının kullanımını

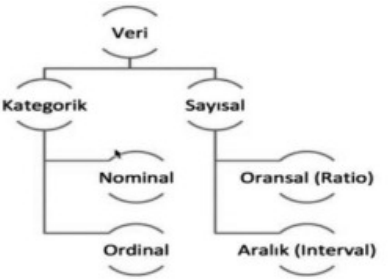
3. Korrelasyon matrisini ve önemini

4. r2_scoru ve OLS metotları ile yöntemler arasında karşılaştırma yapmaya

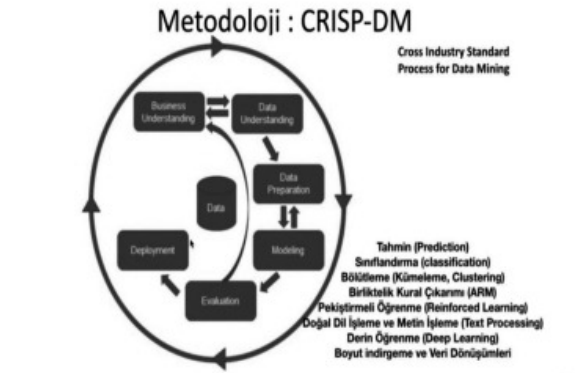
Bölüm 10:

Ders 51 : Sınıflandırma ve Temel Kavramlar, Problemin Algısı

Bu dersten itibaren Sınıflandırma problemlerine giriş yapacağız. İlk olarak önceki derslerimizde gördüğümüz veri tiplerini tekrar hatırlamalıyız.


Veri
Kategorik
Sayısal
Nominal
Oransal (Ratio)
Ordinal
Aralık (Interval)

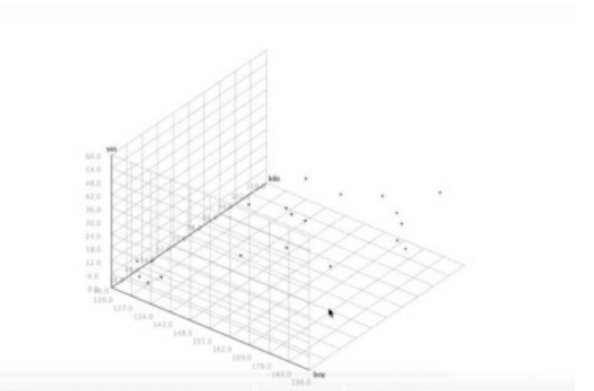
İlk derslerimizde verimizi ana iki kategoriye ayırmıştık. Bu dersimize kadar sayısal verilerimizin tahmini için birden fazla işlem yapmış bulunmaktayız. Bu işlemleri Regresyon yani Tahmin Algoritmaları başlığı altında inceledik.



Sınıflandırma problemlerinde ise sayısal olmayan i yani kategorik veriler ile tahmin gerçekleştirme yapmayı hedeflemekteyiz. Önceden de öğrendiğimiz gibi sayısal verilerin tahminini Re-

gresyon(tahmin), kategorik veriler ile yapılan tahminini de Sınıflandırma başlığı altında incelemekteyiz.

“veriler.csv” üzerinde RapidMiner kullanarak 3 Boyutlu görselleştirme yapmak da mümkündür.



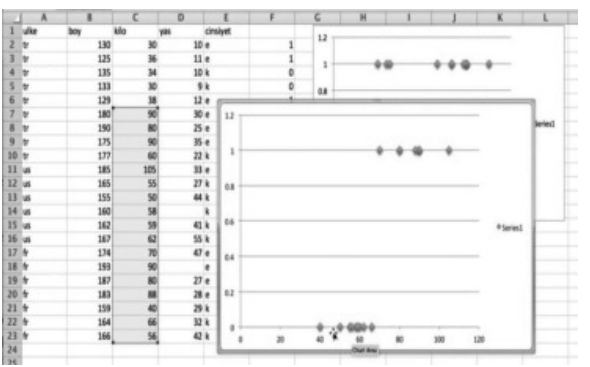
Buradaki genel amacımız cinsiyet değişkeni bazında kadınları ve erkekleri birbirinden ayırabilecek bir model bulmak olacaktır. Bu model ile

boy,kilo ve yaş değişkenlerinin verildiği sistemden bir tahmin yapılması beklenilecektir.

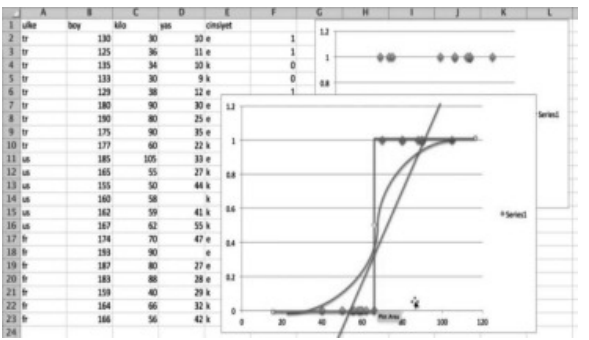
Bölüm 11:

Ders 52 : Lojistik Regresyon Kavramı ve Algoritmanın Teorisi

İlk olarak cinsiyet kolonunu labelencoding gibi dönüşümünü sağlamalıyız. Bu doğrultuda erkek cinsiyetine 1,kadına ise 0 değerini atıyoruz. Bu dersteki genel amacımız bu değerlerin sayısal değere dönüşümünü sağlamak ve regresyon ile bir bağlantı oluşturmak olacaktır. İlk örnek olarak, çocuk örnekleri hariç kilo ve dönüştürülmüş cinsiyet değerlerini Excel'deki Marked Scatter ile görselleştiriyoruz.



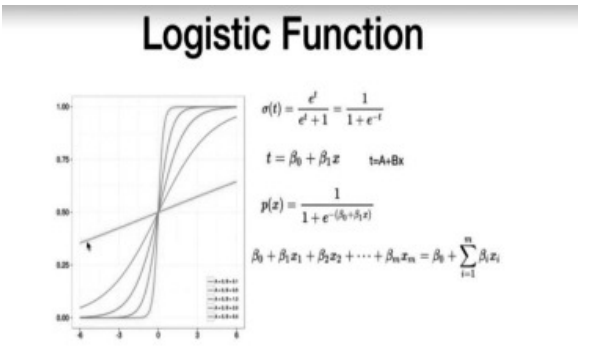
İlk ayırma yöntemleri olarak Doğrusal Regresyon'daki gibi "/" şeklindeki doğru ya da daha verimli olması için Lojistik Regresyon "S" şeklinde bir ayırma yöntemi kullanabliriz.



	A	B	C	D	E	F
1	ülke	boy	kilo	yas	cinsiyet	
2	tr		130	30	10 e	1
3	tr		125	36	11 e	1
4	tr		135	34	10 k	0
5	tr		133	30	9 k	0
6	tr		129	38	12 e	1
7	tr		180	90	30 e	
8	tr		190	80	25 e	
9	tr		175	90	35 e	
10	tr		177	60	22 k	
11	us		185	105	33 e	
12	us		165	55	27 k	
13	us		155	50	44 k	
14	us		160	58	k	
15	us		162	59	41 k	
16	us		167	62	55 k	
17	fr		174	70	47 e	
18	fr		193	90	e	
19	fr		187	80	27 e	
20	fr		183	88	28 e	
21	fr		159	40	29 k	
22	fr		164	66	32 k	
23	fr		166	56	42 k	
24						

Bu ayırma yöntemleri dışında Basamak fonkisyonu adını verdiğimiz "ters Z" şekli ile ayırma yöntemi bize daha iyi bir modelleme sunmaktadır çünkü diğer iki modelde veriye karşılık gelen cinsiyet değeri bulunmamaktadır ve bu durum bizim için problem oluşturmaktadır.

Lojistik Regresyon ile basamak fonksiyonundaki şekil veya doğrusal regresyondaki doğruyu yaratmak da mümkündür.



Lojistik Fonksiyon yardımıyla, istediğimiz kadar değişkenli sınıflandırma yapabilmemiz mümkündür. Böylece Lojistik Regresyon'da daha kapsamlı bir fonksiyon elde etmemiz ve bu fonksiyonlarımız arasında bağlantı kurmamız mümkün olmaktadır.

Özet olarak Lojistik Regresyon, sayısal veriler üzerinden sınıflandırma yapılmasına ve verilen farklı etiketlerin bu sınıflandırma ile elde edilebilmesine olanak sağlamaktadır. Ayrıca bu sayısal değerler üzerinde sınıflandırma yapılırken bir lojistik fonksiyon oluşturulmakta ve bu fonksiyon vasıtasıyla da hata miktarını minimize eden parametreler kolayca belirlenebilinmektedir .

Ders 53 : Python ile Lojistik Regresyon Uygulaması

Bu dersimizde, 14.derste gördüğümüz "verion-islemesablonu.py" kullanılacaktır. Eksik kaynak durumunda http://www.bilkav.com/makine-ogrenmesi-egitimi/ [8]adresinden şablonu indirilebilir. İlk olarak örnek kodları, eksik veriler için olan Imputer'ı , döüştürücü Encoder'ı , dataframe oluşturacak dönüşüm kısmını siliyoruz

8. http://www.bilkav.com/makine-ogrenmesi-egitimi/

ve elimizde sadece verilerin eğitim ve test için bölünmesi ve verilerin ölçeklenmesi kısmı kalıyor.

"eksikveriler.csv" veri kümemizi "veriler.csv" olarak değiştiriyoruz.

Buradaki önemli nokta verilerin hangi kısmının kullanılacağıdır. Eğitim ve test kümesi kısmındaki "s" ve "sonuc3" üzerinden alınan verileri düzenlemeli ve hangi kolondaki verilerin kullanılacağını belirtmeliyiz. Şimdilik ülke kolonunu kullanmayacağız isteğe göre LabelEncoding kullanılabilir. Bu modellememizde boy,kilo ve yaş kolonlarını kullanıp cinsiyet tahmini yapmaya çalışacağız.

Verileri ".iloc" ile bölüyoruz.

```
x = veriler.iloc[:,1:4].values #bağımsız değişken-
ler


y = veriler.iloc[:,4:].values #bağımlı değişken


print(y)
```

“s” ve “sonuc” kısımlarını güncelleyip yerlerine “x” ve “y” koyuyoruz

```
#verilerin egitim ve test icin bolunmesi

from sklearn.cross_validation import train_test_split

x_train, x_test,y_train,y_test = train_test_split(x,y,test_size=0.33, random_state=0)
```

Sci-kit learn kütüphanesi altından kullandığımız StandardScaler ile verilerimizi ölçeklendiriyoruz. Bilindiği üzere, sci-kit learn ile kullanılan

ve çoğu obje için benzer şekilde çalışan iki metot mevcuttur. Bu metotları ".fit_transform()" ve ".transform" olarak tanımaktayız. ".fit_transform" eğitim için kullanılırken ".transform" ise bu eğitimin uygulanması için kullanılmaktadır. Bu nedenle sırasıyla "x_train" ve "x_test" verilerini kullanıyoruz.

```
sc = StandardScaler()

X_train = sc.fit_transform(x_train)

X_test = sc.transform(x_test)
```

Bir sonraki adım olarak "sklearn.linear_model" kütüphanesindeki "LogisticRegression" kul-

lanılacak ve bir obje oluşturulacaktır. (Ayrıntılı bilgi için "scikit-learn.org" adresi kaynak olarak kullanılabilir.)

```
from sklearn.linear_model import LogisticRe-
gression


logr = LogisticRegression(random_state=0)


logr.fit(X_train,y_train)
```

“logr” objesi, “LogisticRegression” ile bilgiyi elde ediyor ve kendisini “X_train” ve “y_train” ile eğitiyor. Sonuçta elimizde eğitilmiş bir Lojistik Regresyon modeli oluşturuluyor. Oluşturduğumuz model, tahmin yapılmasına hazırdır.

```
y_pred = logr.predict(X_test)
```

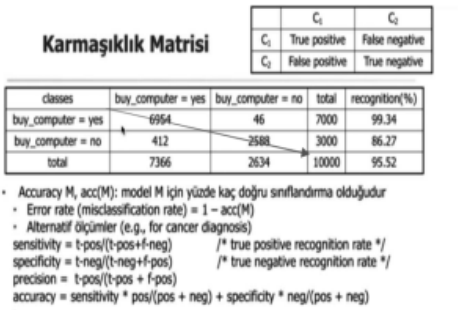
```
print(y_pred)


print(y_test)
```

Sonuç olarak, sadece son kişinin cinsiyeti doğru olarak saptandığı görülmektedir. Bu durum akıllarda soru işaretlerin gelmesine neden olsa da Lojistik Regresyon modelimizin ters tahmin yapma olasılığı da düşünülmelidir.

```
['e' 'e' 'e' 'e' 'k' 'e' 'e' 'k']
[['k']
 ['k']
 ['k']
 ['k']
 ['e']
 ['k']
 ['k']
 ['k']]
```

Ders 54 : Karmaşıklık Matrisi (Confusion Matrix) ve Sınıflandırma Şablonu

Bu dersimizde sınıflandırma problemleri için oldukça önemli bir ölçüm,raporlama yöntemi olan Karmaşıklık Matrisi işlenecektir. Tıpta kullanılan laboratuvar sonuçları terminolojisinden yararlanılmaktadır. Örnek olarak, bir tahlilin pozitif veya negatif çıkması örnek gösterilebilir. Karmaşıklık matrisi ise, bu bağlamı kullanarak gerçek ile tahmin verileri arasında bir bağlantı bulunmasını amaçlamaktadır.



Örnekte bir kişinin bilgisayar satın alıp almaması için tahmin yapılırken doğru ve yanlış tahminlerin ölçülmesi yapılmaktadır. Karmaşıklık Matrisimizdeki köşegen(örnekte mavi okla gösterilmiştir) , her zaman doğru sınıflandırılmış örnekleri göstermektedir. Bu köşegendeki değerler toplanıp toplam örnek sayısına bölünmesi, tahminimizin doğruluk değerini göstermektedir. Bulunan bu değerin 1'den çıkarılması ise ha-

ta miktarını göstermektedir. Fakat bazı durumlar için farklı yöntemler geliştirilmiştir.

Bu farklı durumlara kanser teşhisi örnek gösterilebilir. Örneğin, içerisindeki bir hastanın bulunması için kullanlan yüz kişilik denek grubumuz için geliştirdiğimiz model, %99'luk bir başarı tahminine sahip olduğu söylendiğinde aslında herkesin sağlıklı olduğu tahmini yapan bir modelin %99'luk bir başarısı olduğu da söylenebilmektedir.Bu durumlar için kullanılan diğer yöntemler sensitivity(duyarlılık), specificity(özgüllük), precision(kesinlik)ve accuracy(doğruluk) gösterilebilmektedir.

- Accuracy M, acc(M): model M için yüzde kaç doğru sınıflandırma olduğudur
- Error rate (misclassification rate) = 1 – acc(M)
- Alternatif ölçümler (e.g., for cancer diagnosis)

```
sensitivity = t-pos/(t-pos+f-neg)          /* true positive recognition rate */
specificity = t-neg/(t-neg+f-pos)          /* true negative recognition rate */
precision =  t-pos/(t-pos + f-pos)
accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)
```

Kod kısmı olarak ilk yapacağımız şey "sklearn.metrics" içinden "confusion_matrix" sınıfını alıyoruz. Her zaman yapıldığı üzerine bu sınıf için bir obje("cm") oluşturuyoruz. Karmaşıklık matrisi oluşturmak için aynı verinin tahmini(y_pred) ve orjinalini(y_test) alıyoruz.

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,y_pred)

print(cm)
```

```
[[0 1]
 [6 1]]
```

Karmaşıklık Matrisi'nin bize sağladığı katkı, büyük çaptaki verilerin nasıl sınıflandırıldığını bir matris içerisinden göstermesidir. Örneğin genel bir sınfılandırma algoritmasının başarı için accuracy(doğruluk) kulanılır. Doğruluğu hesaplarken başarılı örnek sayımızı toplam örnek sayımıza bölüyoruz ve 7/8 hata oranı buluyoruz. Algoritmanın başarısının verilerin durumuna bağlı olduğunu göz önünde bulundurduğumuzda, veri kümemizdeki çocuklara ait değerlerin yoksayılma durumunda bu başarının daha da artacağı bilinmektedir. Bunun için ilk beş veriyi silinerek tekrar deneme yapılabilir çünkü söylendiği üzere bu veriler sistemi bozan(outlayer) verilerdir.

Bölüm 12 :

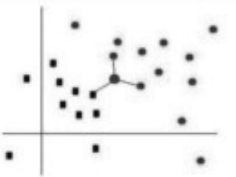
Ders 55 : K En Yakın Komşu (K Nearest Neighborhood "K-NN") Algoritmasına Giriş

http://bilgisayarkavramlari.sadievrenseker.com/?s=knn [9]adresinden ayrıntılı incelemeye ulaşılabilinmektedir.

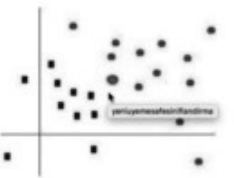
Algoritmanın temel çalışma mantığı, veriler uzaya dağıtıldığında uzaydaki sınıflandırmaya göre yeni bir komşunun sınıflandırılması için en yakın üç komşuya bakılmaktadır.

9. http://bilgisayarkavramlari.sadievrenseker.com/?s=knn

Yukarıdaki bu yeni gelen üyenin en yakın olduğu 3 üyeyi (3 nearest neighbors) tespit edelim.

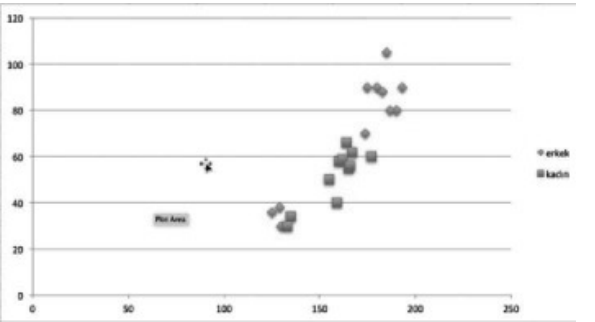


En yakın 3 üyenin iki tanesi kırmızı yuvarlak üyeler olduğuna göre yeni üyemizi bu şekilde sınıflandırabiliriz:



Sonuç olarak iki komşusu kırmızı olan yeni komşu, kırmızı olarak sınıflandırılmaktadır.

Kendi örnek kümemizde alt kısımdaki istisnai veriler çıkarıldığında üst kısım için oluşturulan yeni bir komşu için "erkek" sınıflandırılması yapılırken alt kısım için bu değer sınıflandırma "kadın" olacaktır.



K-NN algoritmasındaki yakınlık için kullanılan ölçüm, basit bir formüle sahip olan öklit mesafesi ile hesaplanmaktadır fakat boy ve kilo bazındaki 20 birimlik değişim birbirinden farklı etkilere sahip olduğundan dolayı mesafe ölçüm algoritmaları kendi içinde değişiklik göstermektedir.

K-NN algoriması çok stabil ve kullanım skalası geniş bir algoritma olmasının yanında basit bir algoritmadır. Bu algoritmanın da kendi içinde varyasyonları bulunmaktadır. Örnek olarak Lazy veya Eager verilebilir. Lazy, örneklediğimiz üzere yeni komşu ile ona en yakın noktalar ile bir komşuluk ilişkisi kurmaktadır. Eager ise yeni komşu yaratılmadan önce verileri bölgelere ayırarak yeni gelecek komşunun hangi sınıfta yer alabileceğini belirlemektedir.

Ders 56 : Python ile KNN Kodlaması

Kodlama kısmına geçmeden önce kaynaklarımızdan bahsedeceğiz.

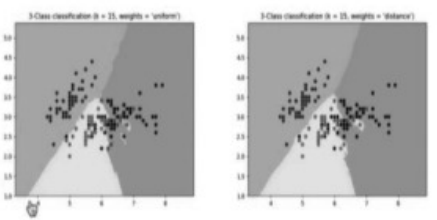
https://scikit-learn.org/stable/modules/neighbors.html[10]

Kaynağımızı incelediğimizde En Yakın Komşu Sınıflandırması ve En Yakın Komşu Regresyon'u görmekteyiz.

10. https://scikit-learn.org/stable/modules/neighbors.html

Saysısal tahmin yapak için ise En Yakın Komşu Regresyon'u kullanılabilinmektedir.

Sınıflandırmamızdaki genel amaç, veri tabolmuzdaki farklı bölgelerin sağlıklı bir şekilde saptanmasını sağlamaktır. Bu işlemi makine öğrenmesi adı altında değerlendirebiliriz. Farklı bölgelerin içinde diğer bölgelere ait alt bölgelere oluştuğu için "doğrusal olarak ayrışma olmayan durumlar" için de KNN algoritması kullanılabilmektedir.



Sınıflandırma yapılırken KNeighborsClassifier sınıfı içinde bazı metotlar bulunmaktadır. Bu metotlar etki değişkeni kullanılır, algoritmalar kullanılır. Örnek algoritmalar kullanılır ve bu algoritmalar örnekler arasındaki hesaplama yöntemlerinde kullanılır. Daha detaylı örnek olarak brute-force, yeni komşu geldiğinde bütün örnekleri inceler ve aralarındaki mesafeleri ölçmeye yaramaktadır ancak bu yöntem bütün verilerin incelenmesini kooşul gördüğünden dolayı en verimli yöntem olmayabilir.

Bir başka metot olarak ise distance metric kullanılabilir. Bu metric için minkowski algoritması kullanılır ve bu metric bazında p=2 olduğu durumda varsayılan olarak yani öklit mesafesi olarak görülmektedir. Bu metodu detaylandıracak olursak minkowski harici distance metric'ler de kullanılabilir.

Metrics intended for real-valued vector spaces:

identifier	class name	args	distance function
"euclidean"	EuclideanDistance	•	`sqrt(sum((x - y)^2))`
"manhattan"	ManhattanDistance	•	`sum(|x - y|)`
"chebyshev"	ChebyshevDistance	•	`max(|x - y|)`
"minkowski"	MinkowskiDistance	p	`sum(|x - y|^p)^(1/p)`
"wminkowski"	WMinkowskiDistance	p, w	`sum(w * |x - y|^p)^(1/p)`
"seuclidean"	SEuclideanDistance	V	`sqrt(sum((x - y)^2 / V))`
"mahalanobis"	MahalanobisDistance	V or VI	`sqrt((x - y)' V^-1 (x - y))`

Örneğin "euclidean", cetvelle ölçüme benzer basit bir ölçümü barındırmaktadır. "manhattan" ise iki nokta arasındaki mutlak değeri hesaplamaktadır.

Kod kısmı olarak önceki koda devam edilmektedir.

Kullanacağımız sınıfı kütüphanemizden alıyoruz ve bir objeye atıyoruz.

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=1, metric='minkowski')

Objemizi ".fit" ile eğitiyoruz.

knn.fit(X_train,y_train)

Eğitilmiş verilerimiz ile tahmin(predict) yapıyoruz

```
y_pred = knn.predict(X_test)
```

Tahminlerimizle Karmaşılık Matrisi'mizi güncelleyoruz.

```
cm = confusion_matrix(y_test,y_pred)
```

Son olarak matrisimizi görüntülüyoruz.

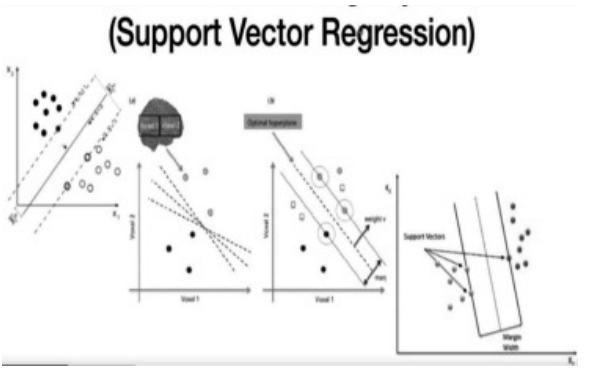
```
print(cm)
```

Özet olarak, farklı komşu sayısı girildiğinde başarı oranı değişmektedir. Örneğin n=5 iken başarı 1/8 iken n=1 durumda başarı oranımız 7/8'e kadar çıkabilmektedir. Farklı durumlarda

n sayısınının artışı olumlu katkılar da sağlayabilmektedir. Bu nedenle, doğru veriler ile doğru parametreleri kullanmak önemlidir ve bu öngörüyü elde edebilmek için ise detaylı bir şekilde algoritmaları kavramak ve mantığını özümsemek gerekmektedir. "metric" değişkenliği de "n" değişkenindeki gibi değişkenlik gösterdiği durumlarda farklı sonuçlar üretmektedir.

Bölüm 13:

Ders 57 : SVM algoritmasının Çalışması ve Detayları



Destek vektör makinaları orijinal olarak sınıflandırma için çıkmış ve farklı sınıfları

doğrusal olarak ayırmak için geliştirilmiş makinal olarak düşünülünebilir. Önceki derslerden hatırlacağımız üzere ana amacımız iki sınıfı ayıran en optimal çizgiyi bulmaktı. Bunun için iki sınıf arasındaki mesafeyi maksimize eden aralığı ve bu aralık yardımıyla orjin noktasından geçen doğruyu bulmamız gerekmektedir. Regresyon kısmında bu aralık içerisine düşen noktaları maksimize ederken sınıflandırma durumunda bu aralığa hiç bir noktanın düşmemesi amaçlanmaktadır.

Standartlaştırma, Destek Vektör Makinaları için önemli aşamalardan birisidir ve şayet doğru bir standartlaştırma yapılamaz ise problemler ortaya çıkmaktadır.

DVM kullandıktan sonra elde edilen sistemin diğer verileri dikkate almayarak bir marjin oluşturması söz konusudur dolayısıyla bir ilkellemeden(Regresyon) de bahsedilebilir. KNN'den de hatırlayabileceğimiz gibi KNN her noktanın çevresinde sınır öğrenebilmektedir ve dolayısıyla çok komplike bir model öğrenebilmektedir.

DSV ise , örnekteki fonksiyonlar cinsinden bu sınırın elde edilebileceği ihtimali üzerinde durmaktadır. Bu kısımdaki önemli ayrım ise gelecek derslerde görebileceğimiz Neural Network (Yapay Sinir Ağları) veya karar ağaçları değişik durumlar için sınıflandırma oluşturmaktadır ve bir bölgenin içinde çok farklı alt bölgeleri çok farklı alt sınıflara dahil edebilirken, DSV ise sınır ayrımlarını bir fonskiyon ile ifade edilebileceği kabulu üzerinden durmaktadır. Fakat bu ayrım her zaman basit bir yapıda olmak zorunda

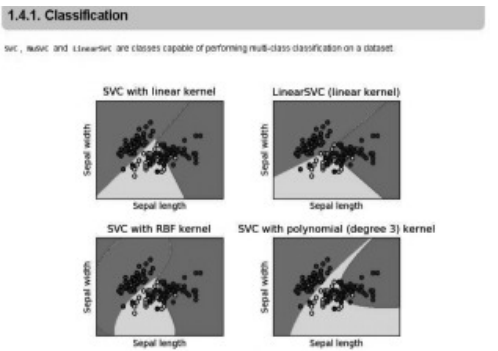
değildir. Diğer bazı algoritmalar bu farkları yakalayabilirken DSM bu konuda az da olsa zaafiyete sahiptir. Bu zaafiyetin getirdiği avantajlar ise hız ve verinin saklanabilmesi(işlem gücü) açısından önemli bir yapıya sahip olmaktır. DVM'nın günümüzdeki problemlerinden birisi de büyük veri konusundaki paralelleştirme ile ilgilidir. DVM çoğu makine öğrenme algoritmasında(paralelleştirilen) yer almamaktadır .

Ders 58 : SVM algoritmasının Python ile Kodlanması

https://scikit-learn.org/stable/modules/svm [11]adresinden gerekli literatür taramasını yapılabilmektedir.

11. https://scikit-learn.org/stable/modules/svm

Bu kısımda sınıflandırmadan bahsedilecektir. Sınıflandırma için birden fazla sınıf mevcuttur bunlar SVC, NuSVC ve LinearSVC'dir.



İris veri kümesi, veri biliminde veya makine öğrenmesinde sıklıkla kullanılan örnek veri kümesidir. Bu kümedeki ayrım sonucundaki yaşanılan en büyük sıkıntı ise birbirine girişken olan verilerdir. Örnekteki üç sınıfı birbirinden

ayırmak için algoritmamız birebir eşleştirme yaparak veriyi sınıflandırmaktadır. Bunun için algoritmamız, problemimizi üç farklı sınıflandırma problemi gibi ele alıp her sınıfı teker teker diğer sınıflarla karşılaştırarak aralarındaki çizgileri belirlemeye çalışmaktadır.

Kod kısmı olarak öncelikle SVM sınıfımız kütüphanemizden çağırıyoruz.

```
from sklearn.svm import SVC
```

Obje oluşturup SVC 'yi atıyoruz ve kernel için "poly" fonksiyonunu kullanıyoruz. Önceki derslerden de hatırlayabileceğimiz gibi kernel için doğrusal("linear"), polinomal("poly"), logit

(“sigmoid”), “precomputed” ve rbf(“rbf”) gibi değişik alternatiflerimiz de mevcuttur.

```
svc = SVC(kernel='poly')
```

Veri kümemizin eğitilmesi için “.fit” kullanılmaktadır. Eğitilirken modelimiz polinomal bir ayrım noktası bulmaya çalışmaktadır.

```
svc.fit(X_train,y_train)
```

Tahminimizi objemiz vasıtasıyla gerçekleştiriyoruz.

```
y_pred = svc.predict(X_test)
```

Karmaşıklık Matrisi'mizi tekrar yaratıyoruz.

```
cm = confusion_matrix(y_test,y_pred)
```

```
print('SVC')
```

Sonuçlarımızı görüntülüyoruz.

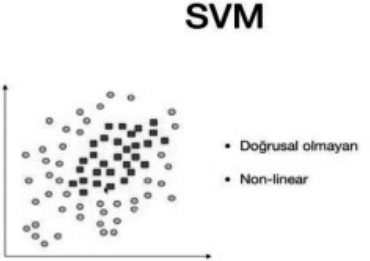
```
print(cm)
```

```
[[0 1]
 [6 1]]
[[1 0]
 [1 6]]
SVC
[[1 0]
 [7 0]]
```

Farklı kernel seçeneklerinin farklı veri kümeleri ile verdiği sonuçları gözlemlenerek(parametre optimizasyonu) tecrübe kazanmak, makine öğrenmesi alanında uzmanlaşmak isteyen kişiler için artı bir değer oluşturmaktadır. Özet olarak ele alınan problemlerin yeterince tanınması ve kazanılmış tecrübe ile olası çözümlerin değerlendirilmesi önemlidir.

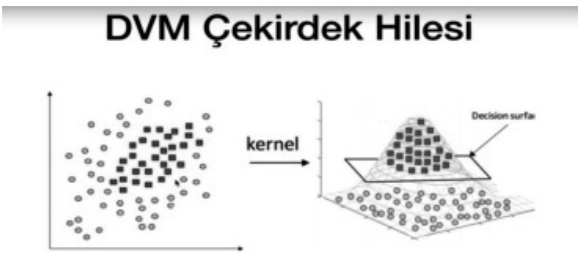
Bölüm 14:

Ders 59 : Kernel Trick (Çekirdek Hilesi)

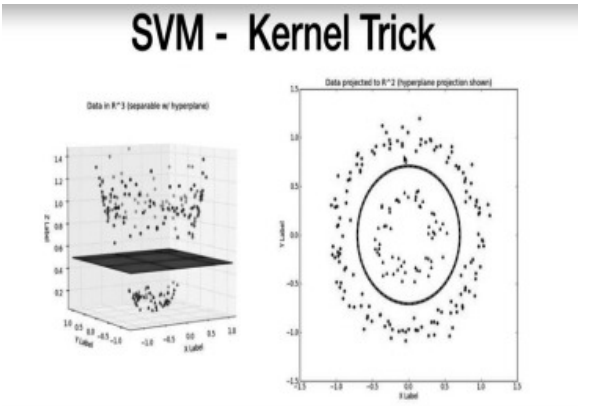


Bu veri kümemizi marjin aralığında istisnaları kabul edebilen "soft marjin" ya da bu istisnaları bile kabul etmeyen "hard marjin" ile ayırma şansımız yoktur. Şayet örnekteki kırmızı veri kümesinin ortasındaki gibi bir doğru nokta belirleyebilirsek bu kümeyi kendi içinde ayrı bir gruba paylaştırabiliriz. Doğrusal olmayan problemleri çözmek için KNN algoritması oldukça kullanışlıdır fakat DVM gibi doğrusal olarak ayrıştırmaya çalışan bir algoritmayı bu örneğe nasıl adapte edeceğimizi öğrenmemiz gerekmektedir.Bu sorunun cevabı ise üçüncü boyut elde etmektir. Bu boyut, örneğimizden yola çıkacak olursak kırmızı veri kümemizin orta noktasını(çekirdek noktası) çevresindeki veriler arasındaki mesafeyi çarpar ya da bölersek daha uza-

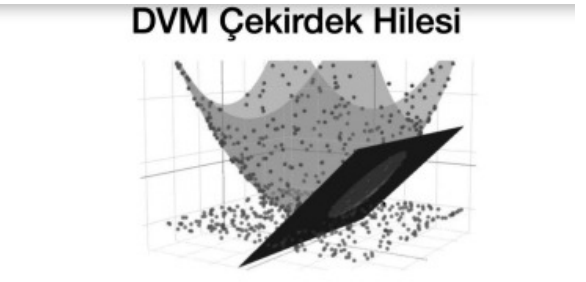
ktakiler daha aşağı kısımda yer alırken daha yakındakiler ise daha yukarıda yer almaktadır.



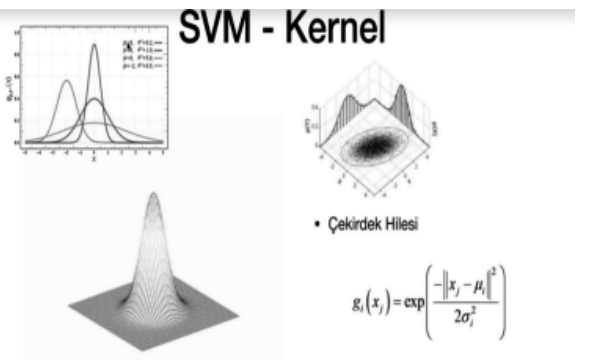
Temel şekilde düşünecek olursak bu yöntemle kendi sınıfından olan veriler sınıfdaşlarınını kendilerine yakın bir mesafeye konumlandırmaktadır. Bu sınıfları birbirinden ayırmayı üçüncü boyut olmasından dolayı uzayda düzlem olarak elde etsek bile üçüncü boyutta bir noktaya da karşılık gelmektedir.


SVM - Kernel Trick
Data in R^3 (separable w/ hyperplane)
Data projected to R^2 (hyperplane projection shown)

Birden fazla boyuta bağlı olan düzlemler çizerek bu gibi hilelerle verileri ayrıştırmamız da mümkündür.


DVM Çekirdek Hilesi

Yukarıdaki örneklerde de bahsettiğimiz çekirdek noktası belirlenmesi sürecinde Kernel fonksiyonları kullanılmaktadır. Örneğin "rbf" fonksiyonunun doğru sigma değerleri ile doğru dikliğe sahip dağılım bulunarak veriler bu dağılama göre dağıtılacak ve çarpanlar elde edilecektir. Bu çarpanlar ile verileri çarparak yakındaki çarpılan verilerin yakında kalması uzaktakilerin ise daha aşağıda kalması amaçlanmaktadır..

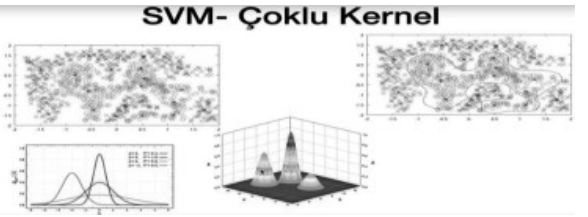


Sonuç olarak elde edeceğimiz koni şekli ile sınıflar arası ayrımı sağlayan bir düzlem ile düzgün bir nokta elde etmek mümkündür. "rbf" özelinde konuşmak gerekirse sigmanın değerini azaltarak konimiz üstünden bakıldığında elde edeceğimiz çemberin çapını daraltabiliriz. Bu ve bunun gibi kernel hileleri veya kernel fonksiyonları kullanımı ile elde edilen başarıyı arttırmak, günümüzde üzerinde fazlaca düşünülen ve çoğu tez konusuna ilham veren bir yaklaşımdır. Bu gibi iyileştirmeler basit bir algoritma bazında olsa bile çok geniş bir eylem dünyası kullanıcılara sunmaktadır. Bu dünyanın sadece bir kısmı python ile sklearn kütüphanesi ile kullanıma sunulmuş durumdadır.Örneğin, bu kütüphanede bulunmayan ama bilimsel literatürde mevcut olan ve bazı durumlarda çok daha başarılı çalışan çok sayıda algoritma olduğunun bilinmesinde fayda vardır.

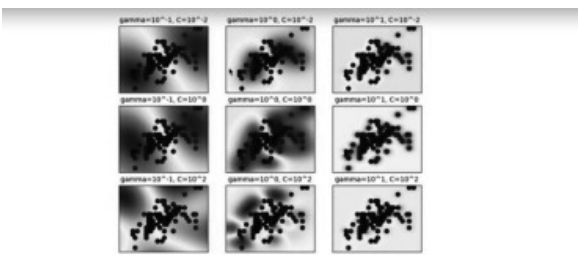
Özet olarak, DVM'nın doğrusal ayrıştırma özelliği üçünü boyuta taşınarak kernel hilesi yapılmakta ve doğrusal olarak ayrılamayacak verilerin de ayırabilmesi hedeflenmektedir.

Ders 60 : Python ile Çekirdek Hilesi Kodlama

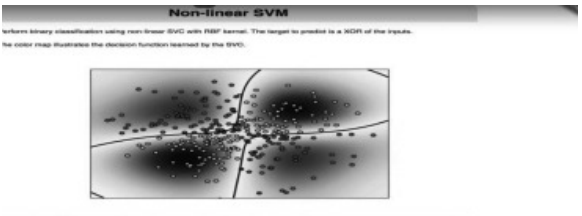
Destek vektör kullanımında çoklu kernel kullanımı da mümkündür. Birden fazla kernel noktası belirlenmesi farklı şekillerde dağılan verileri sınıflandırmak için kullanılan yöntemlerdendir.


SVM- Çoklu Kernel

Birden fazla "rbf" fonksiyonunu birbirine ekledikten sonra bu noktaları üçüncü boyutta yukarıya çektikten sonra hepsini aynı düzlemden kesip ayırarak farklı boyutlardaki dairelere yakın kesik şekiller de oluşturmak mümkündür. Bu şekilde doğrusal olmayan bir sınıflandırma yapmak mümkündür.



Örneğimizde iki sınıf için farklı dağılımlar kullanıldığındaki sonuçlar görüntülenmektedir. Görüldüğü üzere doğrusal bir ayrım şeklinden heatmap'e evrilmektedir. Bu dağılımları incelediğimizde ise doğrusal olmayan durumları da sınıflandırma şansımız vardır.



Bu örneğimiz ise fazlasıyla literatürde kullanılan XOR(Mantık Operatörü) örneğidir. Bu örnek te farklı kernel çekirdek noktaları belirlenmiş olduğu gözlenmektedir.

Kod kısmı olarak DVM'deki aynı kodu kullanıyoruz ve öncelikle SVM sınıfımız kütüphanemizden çağırıyoruz.

```
from sklearn.svm import SVC
```

Obje oluşturup SVC ‘yi atıyoruz ve kernel için “poly” fonksiyonunu kullanıyoruz. Önceki derslerden de hatırlayabileceğimiz gibi kernel için doğrusal(“linear”), polinomal(“poly”), logit (“sigmoid”), “precomputed” ve rbf(“rbf”) gibi değişik alternatiflerimiz de mevcuttur.

```
svc = SVC(kernel='poly')
```

Veri kümemizin eğitilmesi için “.fit” kullanılmaktadır. Eğitilirken modelimiz polinomal bir ayrım noktası bulmaya çalışmaktadır.

```
svc.fit(X_train,y_train)
```

Tahminimizi objemiz vasıtasıyla gerçekleştiriyoruz.

```
y_pred = svc.predict(X_test)
```

Karmaşıklık Matrisi'mizi tekrar yaratıyoruz.

```
cm = confusion_matrix(y_test,y_pred)
```

```
print('SVC')
```

Sonuçlarımızı görüntülüyoruz.

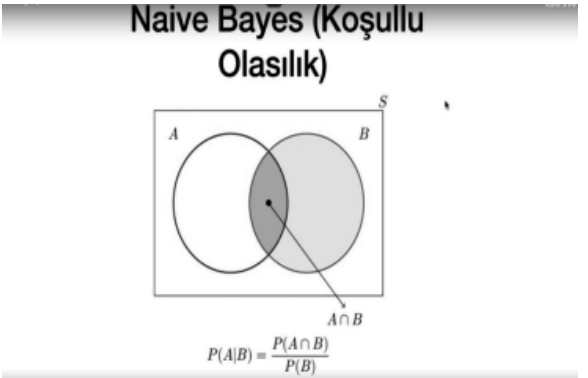
```
print(cm)
```

```
[[0 1]
 [6 1]]
[[1 0]
 [1 6]]
SVC
[[1 0]
 [7 0]]
```

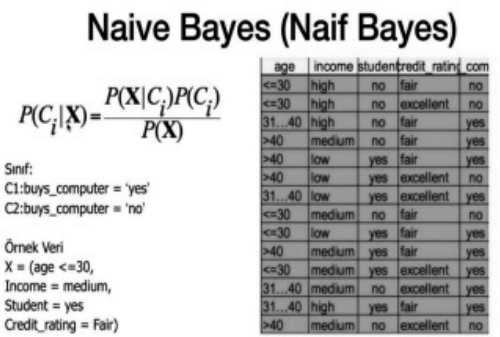
Bölüm 15:

Ders 61 : Naif Bayes (Naive Bayes)

Bu dersimizde Naive Bayes sınıflandırması incelenecektir. İlk olarak Bayes teoremi ve olası problemlerin tanımını yapılmalıdır. Teoremin net bir şekilde anlaşılabilmesi için öncelikle "koşullu olasılık" kavramının bilinmesi gerekmektedir.



Bu olasılık gerçekleşen B olayına bağlı olarak A olayının gerçekleşme ihtimalini incelemektedir. Örnek olarak yağmur yağma ihtimali ve buna bağlı olarak şemsiyemizi yanımıza alma ihtimali gösterilebilir.



Bayes Teoremi'nin sağladığı en büyük avantaj, koşuldaki olayların yerlerinin değiştirebilir olmasıdır. Örnekte görüldüğü üzere farklı gelir düzeyi, yaş, öğrenci olma durumu, kredi skoru ve bilgisayar satın alma gibi kriterler görülebilmektedir. Bu verilere bağlı olarak olası müşterilerin bilgileri kullanılarak bilgisayar alıp almayacağına dair bir tahmin yapılması mümkündür.

Örnek

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357

- Compute $P(X|C_i)$ for each class
  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Öncelikle koşul durumuna bağlı olarak kriterlerinin teker teker oranlanması gerekmektedir. Örnek olarak bilgisayar alan 9 müşteri içerisinden sadece 2 müşterinin 30 yaşının altında olduğu görülmektedir ve bu oran 2/9 işlemi sonucu olarak 0.222 bulunmaktadır.

Örnek verimiz için oranlarımız hesaplandıktan sonra değerlendirme kriterine sahip müşteriler için bilgisayar alanlar ve almayanlar kendi aralarında çarpılmaktadır. Son adım olarak hesaplanan sonucu normalleştirmek için kriterlere bağlı olmaksızın bilgisayar alınıp alınmama oranı ile çarpılmaktadır (Bilgisayar alanlar için 0.044* 0.643, almayanlar için 0.019*0.357 olarak hesaplanmaktadır). Böylece bilgisayar alınma durumunun oranını daha da yükseltirken alınmama durumunu tersi şekilde daha da düşürüyoruz ve sonuç olarak örnek verideki

kriterlere sahip müşterinin bilgisayar alma oranı almama oranının 0.028/0.035 olarak %80 olarak hesaplamaktayız.

Naive Bayes, bir veri kümesi üzerinde verilen bir örneğin etiketleme (Örn. bilgisayar alınma veya alınmama) ihtimalini tahmin etmeyi amaçlamaktadır. Bu nedenle Naive Bayes'i bir sınıflandırma algoritması olarak incelemekle beraber bu algoritma sınıflandırma işlemi sırasında olasılıkları kullanmaktadır.

Örneğin KNN yöntemi uzaklık metriği kullanırken Naive Bayes, bütün dağılıma bakarak her bir bireyin dağılım üzerindeki etkisini ölçerek olasılıksal bir model ile ilkelleyebilmeyi mümkün hale getirmektedir. Bu doğrultuda

Naive Bayes olasıksal yöntemler kullanması açısından avantaja sahip olduğu söylenebilinmektedir.

Geçmiş derslerden hatırlanacağı üzere Lazy learning(Tembel Öğrenme) sırasında veri kümemiz yeni gelecek kriterin etkisini hesaplamak için sürekli aktif tutulmak zorundadır fakat Eager Learning(İstekli Öğrenme) için bu durum geçerli değildir. Eager için herhangi bir kriter veri kümesinden istenilmesine gerek duyulmadan bütün olasılıklar veri kümesinden çıkartılarak veri kümesine olan ihtiyaç sıfırlanmaktadır. Örnek olarak, veri kümesinin karmaşık olma durumlarında Lazy Learning'in kullanılmasından dolayı "çeşitli durumlarda farklı öğrenme yöntemi kullanılır" denilmektedir.

Ders 62 : Python Kodu: Naive Bayes ve GaussianNB, MultinominaLNB, BernolliNB

Sci-kit Learn kütüphanesi Naive Bayes için iç farklı yöntemden bahsedilmektedir ve bunlar GaussianNB, MultinominalNB ve BernolliNB'dir.

Bu yöntemlerin özelliği farklı dağılımlar üzerinden çalışmasıdır.

GaussianNB'de sürekliliğe sahip olan değerleri(reel sayı, ondalık sayı vs.), MultinominalNB'de sayı atanan nominal değerleri(mezun olunana okul, araba markaları vs.) ve Bernolli'de ikili dağıtım olan değerleri(Kadın-Erkek, 0-1, sigara içme-içmeme vs.) içeren durumlarda kullanılmaktadır.

GaussianNB, ayrılabilir bir uzay seçeneğine sahip olduğundan dolayı ve büyük uzaydan küçük uzaya geçebilme olanağı tanıdığından dolayı bu ders için ağırlıklı olanak kullanılacaktır.

Kodlama kısmı olarak ilk olarak GaussianNB'yi kütüphanemizden çağırıyoruz.

```
from sklearn.naive_bayes import GaussianNB
```

Obje oluşturuyoruz.

```
gnb = GaussianNB()
```

Eğitim işlemini tamamlıyoruz.

```
gnb.fit(X_train, y_train)
```

Tahminimizi gerçekleştiriyoruz.

```
y_pred = gnb.predict(X_test)
```

Sonucumuzu görmek için sonucumuzu Karmaşıklık Matrisi'nde görüntülüyoruz.

```
cm = confusion_matrix(y_test,y_pred)

print('GNB')

print(cm)
```

```
GNB
[[0 1]
 [6 1]]
```

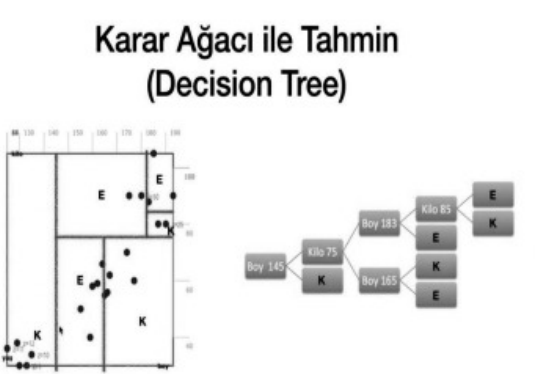
Sonuç olarak 8 tahminin sadece 1 tanesi doğru olarak tahmin edilmektedir. Bu sonucun nedeni aynı veri kümemizi ona uygun olmayan algoritmalarda kullandığımızdan kaynaklanmaktadır.

Bu derslerdeki amacımız optimizasyon olmamakla beraber gelecek derslerde bu konu üzerinde de çalışılacaktır.

Bölüm 16:

Ders 63 : Karar Ağacı ile Sınıflandırma (Decision Tree)

Regresyon konusundan bilindiği üzere Karar ağaçları, veri uzayını bölgelere ayırmakta ve bu ayrımlara göre karar ağaçları oluşturmaktaydı.



Bu ders özelinde, olası bir problem olarak örnekteki bir bölgede verilerin homojen olarak dağılmama durumunda ne gibi çözümler üretilebileceği görülecektir.

İlk çözüm olarak bu bölgenin daha fazla alt bölgeye bölünememesi ve hangi veri sınıfı çoğunlukta ise o bölgenin tamamının o sınıf ile

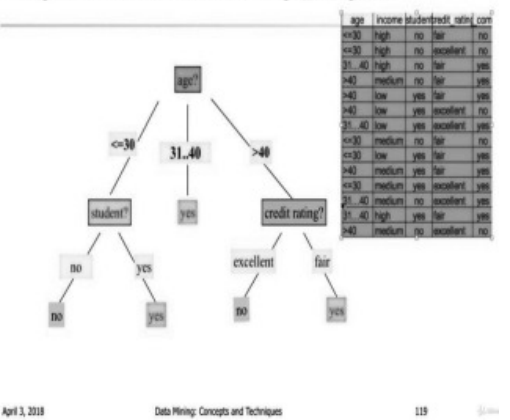
işaretlenmesidir(majority voting(çoğunluğun oyu)).

İkinci yaklaşımda ise homojen dağılım elde edilinceye kadar o bölgenin kendi alt bölgelerine bölünmesidir. İki yaklaşımında kendi özelinde avantaj ve dezavantajları vardır.

İlk yaklaşım için büyük bir veri kümesi üzerinde ihmal edilecek detayların fazlalığı sorun teşkil etmektedir. Örnek olarak 10 bin kişilik bir veri setinde 5001 kişinin kadın olma durumunda tüm veriyi kadın olarak sınıflamak problem yaratacaktır.

İkinci yaklaşım olarak da bölgeleri fazlaca bölgelemek overfitting riski yaratmaktadır. Böylece

her örneğin kendine ait bölgesi oluşucağından dolayı karar ağacımız öğrenmek yerine ezberleme yapmaya başlayacaktır ve bu durum tehlike yaratmaktadır.


Output: A Decision Tree for "buys_computer"
age?
<=30
31..40
>40
student?
yes
credit rating?
no
yes
excellent
fair
no
yes
no
yes
April 3, 2018
Data Mining: Concepts and Techniques
119

Örneğimizin Sınıflandırma bazından değerlendiğinden dolayı önceki derslerimizde görülenin aksine kategorik verilerin ayrım noktalarının belirlendiği görülmektedir.

Söz konusu olan karar ağacının çizimi asıl üzerinde düşünülmesi gereken konudur. Örneğimizde bağlı değişkeni karar ağacımızın uç kısımlarında görsek de kök olarak belirlediğimiz bağımsız değişkenimizi hangi kriterlere bağlı olarak saptadığımızın cevabı ise entropi kavramı karşımıza çıkmaktadır.

Bu kavaramı Information olarak ifade etmekler birlikte Information genel olarak bir sınıfın olasılıksal dağılımını temsil etmektedir.

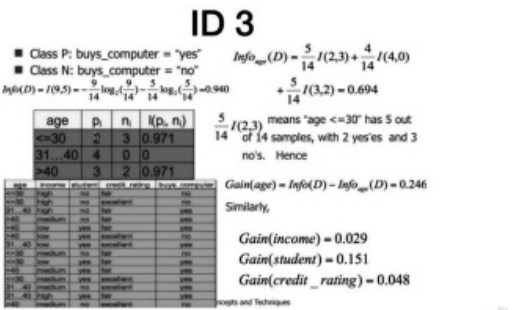
$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

$$Gain(A) = Info(D) - Info_A(D)$$

Bu formülde Pi olasılığı (sigara içme-içmeme) ifade etmektedir. Bu olaslığın, kendisinin 2 logaritmik tabanındaki haliyle çarpılması sonucu Info(D) elde edilmektedir. Her bir örneğin(yaş, öğrenci olma durumu) gösterdiği değer ile Information çarpılmakta ve sonrasında ise Gain(Kazanç) hesaplanmaktadır.

Bu hesaplama algoritması “ID 3” olarak literatürde bulunmaktadır.



Bu hesaplamamızdaki amaç, karar ağacının kök kısmına yerleştireceğimiz olan değerin (yaş, öğrencilik durumu vs.) karar ağacımız üzerindeki dağılımını hesaplamaktır. Örnek olarak sorulacak sorunun sol kısmında tüm veri kümesi olurken sağ kısmının boş olması sorulan bu soruyu anlamsızlaştırmaktadır. Bu dağıtım yapılırken en fazla devam yolu olan ve dolayısıyla veri kümesiniz en fazla dağıtan yani en küçük problem kümelerini bırak devam yolu tercih edilmektedir.

Değerlerin kendi özelinde kazanç (Gain) değerleri hesaplandıktan sonra kök kısmını belirleyebiliyoruz. Kazançları hesaplandıktan sonra yaş değerinin (0.246); gelir (0.029), öğrencilik durumu (0.151) ve kredi skoru (0.048) değerlerinden daha büyük kazanç sağladığı görülmekte olduğundan dolayı yaş değeri kök kısmına yazılmaktadır.

İlk olarak yaş bazında ayırım yapıldıktan sonraki ayrım net bir sonuç vermiyorsa soru sorulmasına devam edilmektedir. Eğer sorulabilecek sorular tükenmiş se ve alt ayrım sonucumuzda hala farklı son olasılıkları var ise total başarının (100 kişinin 80'i bilgisayar aldı yani %80) oranı bu karar ağacı için kabul edilmektedir.

Ders 64 : Python ile Karar Ağacı Sınıflandırma Kodu

Literatür kısmında sci-kit learn kütüphanesinin içerisindeki "DecisionTreeClassifier" yönteminde bazı parametreler bulunmaktadır. Bu ders için "criterion" yöntemi özelinde kalınacaktır ve bu yöntem "gini" veya "entropy" kullanılabilmesini sağlamaktadır. Varsayılan olarak "gini" parametresi belirlenmiştir. Önceki derste hesaplanan Info(D) değerinden farklı olarak logaritma kullanılmazken sonuç olarak olasılığın karesi olan Pi^2 elde edilmektedir.

Kod olarak,

Kütüphanemizden kullanılancak Sınıflandırma yöntemini çağırılıyor.

```
from sklearn.tree import DecisionTreeClassifier
```

Yöntemimiz objeye atanır ve parametremiz belirlenir.

```
dtc = DecisionTreeClassifier(criterion = 'entropy')
```

Veriler eğitilir.

```
dtc.fit(X_train,y_train)
```

Veriler ile tahmin yapılır.

```
y_pred = dtc.predict(X_test)
```

Karmaşıklık matrisi üzerinde sonuçlar değerlendirilir.

```
cm = confusion_matrix(y_test,y_pred)

print('DTC')

print(cm)
```

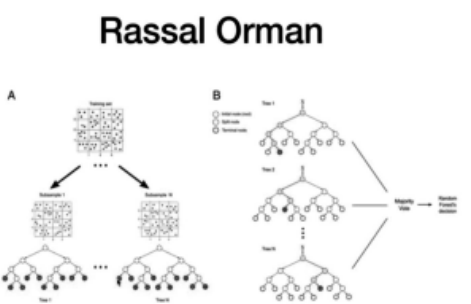
```
DTC
[[1 0]
 [1 6]]
```

Bölüm 17:

Ders 65: Rassal Orman ile Sınıflandırma

Regresyon modelleri bölümlerinden öğrenildiği gibi Rassal Orman, veri kümemizdeki sayısal değerleri farklı parçalara bölümleyip bu parçalar üzerinden farklı karar ağaçları oluşturmaya ve bu karar ağaçlarının ortak kararı olan "çoğunluğun oyu" ile tahmin işlemi gerçekleştirmekteydi. Bu regresyon modelinden farklı olarak elimizdeki verilerin sayısal değerleriyle işlem yapmak yerine

direk olarak kategorik verilerimizin oy dağılımları incelenmektedir.


Rassal Orman
A
B

Konumuzla ilgili olarak Microsoft firmasının Xbox cihazında hareketleri algılayıp sınıflandırmaya yönelik bir yöntem kullanılmaktadır. (https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf [12]).

12. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf

Bu makalede bahsedilen konulardan bazıları veri bilimi konsepti için önemlidir. Örnek olarak, ölçülmüş veriler üzerinden bir formül ya da bir simülasyona göre sentetik veri üretimi yapılabileceği ve bu verilerin test amacıyla

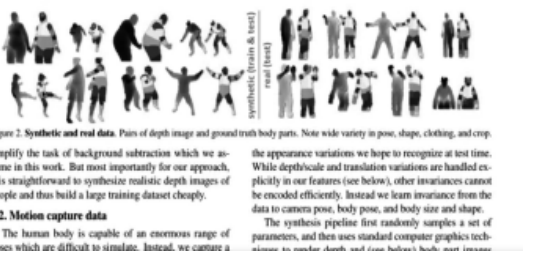


Figure 2. **Synthetic and real data**. Pairs of depth image and ground truth body parts. Note wide variety in pose, shape, clothing, and crop.

simplify the task of background subtraction which we assume in this work. But most importantly for our approach, it is straightforward to synthesize realistic depth images of people and thus build a large training dataset cheaply.

2.2. Motion capture data

The human body is capable of an enormous range of poses which are difficult to simulate. Instead, we capture a

the appearance variations we hope to recognize at test time. While depth/scale and translation variations are handled explicitly in our features (see below), other invariances cannot be encoded efficiently. Instead we learn invariance from the data to camera pose, body pose, and body size and shape.

The synthesis pipeline first randomly samples a set of parameters, and then uses standard computer graphics tech-

kullanılabildiği belirtilmektedir.

Bu makale özelinde, saniyede çekilen 200 kare fotoğraf içerisinden vücut bölgelerinin saptanması üzerinde çalışıldığından bahsedilmektedir ve bu şekilde cihaza komut verilerek kullanıcının vücut hareketleri belirlenmektedir.

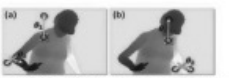


Figure 3. **Depth image features.** The yellow crosses indicates the pixel x being classified. The red circles indicate the offset pixels as defined in Eq. 1. In (a), the two example features give a large depth difference response. In (b), the same two features at new image locations give a much smaller response.

parts for left and right allow the classifier to disambiguate the left and right sides of the body.

Of course, the precise definition of these parts could be changed to suit a particular application. For example, in an upper body tracking scenario, all the lower body parts could be merged. Parts should be sufficiently small to accurately localize body joints, but not too numerous as to waste ca-

Figure 4. **Randomized Decision Forests.** A forest is an ensemble of trees. Each tree consists of split nodes (blue) and leaf nodes (green). The red arrows indicate the different paths that might be taken by different trees for a particular input.

3.3. Randomized decision forests

Randomized decision trees and forests [15, 30, 7, 9] have proven fast and effective multi-class classifiers for many tasks [20, 23, 36], and can be implemented efficiently on the GPU [34]. As illustrated in Fig. 4, a forest is an ensemble of T decision trees, each consisting of split and leaf nodes. Each split node consists of a feature f_θ and a threshold τ. To classify pixel x in image I, one starts at the root and re-

Ders 66: Python Kodu: Rassal Orman ile Sınıflandırma

Literatürde sınıflandırma modelinde kullanılan parametreler incelenmektedir. (https://scikit-learn.org/0.20/modules/generated/sklearn.ensemble.RandomForestClassifier.html [13])

Kod kısmı olarak regresyon yerine sınıflandırma yöntemini sci-kit learn kütüphanemizden çağırıyoruz.

from sklearn.ensemble import RandomForestClassifier

13. https://scikit-learn.org/0.20/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Yöntemimiz objeye tanımlanır.

```
rfc = RandomForestClassifier(n_estimators=10, criterion = 'entropy')
```

Veriler eğitilir.

```
rfc.fit(X_train,y_train)
```

Tahmin yapılır.

```
y_pred = rfc.predict(X_test)
```

Sonuçlar karmaşıklık matrisinden gözlemlenir.

```
cm = confusion_matrix(y_test,y_pred)


print('RFC')

print(cm)
```

```
RFC
[[1 0]
 [2 5]]
```

Bölüm 18:

Ders 67: Karmaşıklık Matrisi

Karmaşıklık Matrisi
Confusion Matrix

	Tahmin C_1	Tahmin C_2
Gerçek C_1	True positive	False negative
Gerçek C_2	False positive	True negative

Karmaşıklık Matrisi'nde gerçek değerler ile tahmin değerleri arasındaki doğruluk incelenmektedir.

Karmaşıklık Matrisi

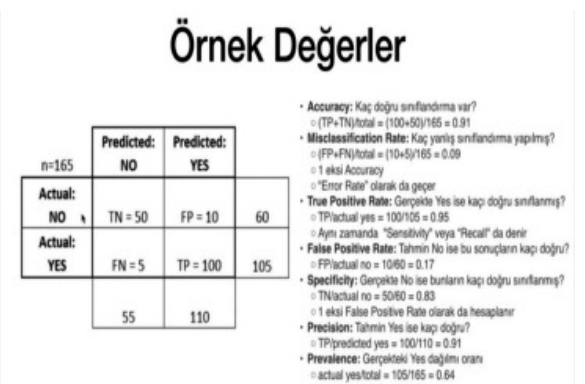
	C_1	C_2
C_1	True positive	False negative
C_2	False positive	True negative

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.52

- Accuracy M, acc(M): model M için yüzde kaç doğru sınıflandırma olduğudur
  - Error rate (misclassification rate) = 1 – acc(M)
  - Alternatif ölçümler (e.g., for cancer diagnosis)

```
sensitivity = t-pos/(t-pos+f-neg)          /* true positive recognition rate */
specificity = t-neg/(t-neg+f-pos)          /* true negative recognition rate */
precision =  t-pos/(t-pos + f-pos)
accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)
```

Üzerinde düşünülmesi gerekilen asıl nokta ise başarı kriterinin ne olması gerektiğidir. Örneğin 100 değer içerisindeki hatalı bir değeri bulmak isterken eğer modelimiz her değere hatasız etiketi ataması yaparsa sonuç olarak %99 oranında bir başarı saptanabilir fakat bu durum bize modelimizin öğrenme yapmadığını ve bu başarı oranına rağmen işlevsiz olabileceğini göstermektedir. Bu nedenle verilerin dağılımı başarı kriterimiz üzerinde önemli bir değere sahiptir.

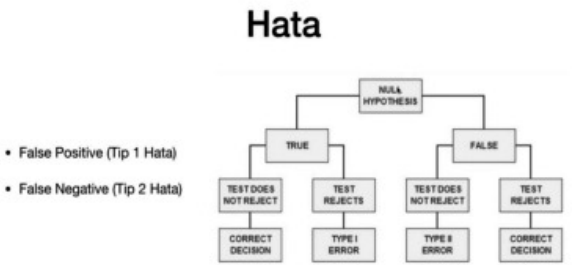


Değerlendirme

Sensitivity – True Positive Rate	$\frac{TP}{P} = \frac{TP}{TP+FN}$
Specificity – True Negative Rate	$\frac{TN}{N} = \frac{TN}{TN+FP}$
Precision – Positive Predicted Value	$\frac{TP}{TP+FP}$
Negative Predicted Value	$\frac{TN}{TN+FN}$
Fall-out – False Positive Rate	$\frac{FP}{N} = \frac{FP}{TN+FP}$
False Discovery Rate	$\frac{FP}{TP+FP}$
Miss Rate – False Negative Rate	$\frac{FN}{P} = \frac{FN}{TP+FN}$
Accuracy	$\frac{TP+TN}{Total}$
F1 Score	$\frac{2TP}{2TP+FP+FN}$

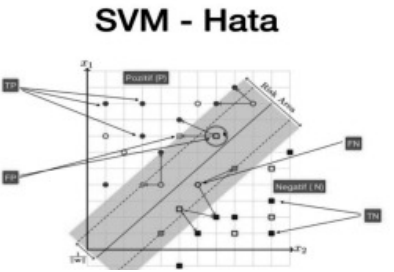
Ders 68 : False Positive ve False Negative Kavramları

Öncelikle Null Hipoteziyle dersimize giriş yapılacaktır. Null Hipotezine göre veri biliminde her şey bir kabul ile başlamaktadır ve bu kabulün doğru veya yanlış olmasıyla devam etmektedir. Örnek olarak makinenin satranç oyununda yaptığı her hamlenin oyunu galibiyete götüreceği kabul edilmektedir. Bu iki karar(doğru ve yanlış olma durumu), bizim seçimimizin doğru ya da yanlış sonuçlanmasıyla son erer. Bu bağlamda iki tip hata bulunmaktadır. Tip 1 hata doğru karar verilip yanlış sonuçlanması durumu(False Positive) iken, Tip 2 hata yanlış karar verilip yanlış sonuçlanmasını(False Negative) temsil etmektedir.



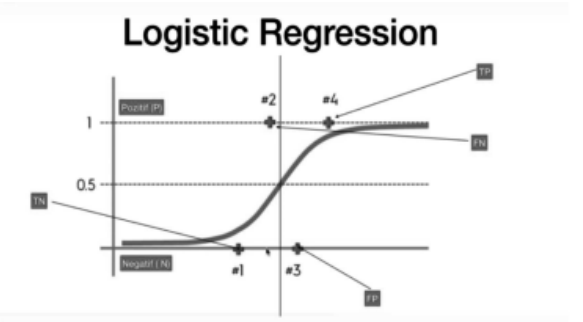
Destek Vektör Makinaları'ndan da hatırlanacağı üzere bir marjin oluşturmaya ve bu marin içerisindeki alanı maksimize etmeye çalışıyordu. Öğrenme verileri ile makinemizin öğrenme yaptığı varsayılıyor. Örnek olarak, şekildeki verilerin içi dolu olanlarının eğitim için kullanılan ve içi boş olanlar test için kullanıldığı varsayılıyor ve karelerin negatif yuvarlakların pozitif değerleri temsil ettiği

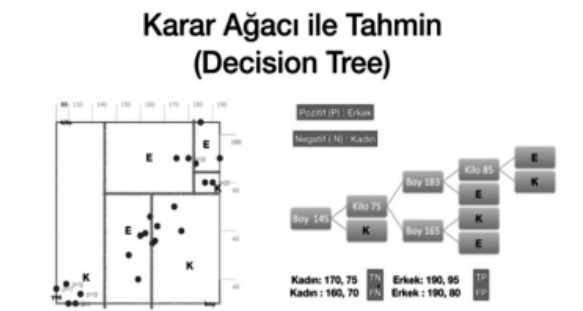
düşünülüyor.



Eğitim işlemi yapıldıktan sonra DVM'sinin doğrusal bir çizgiyle ayırdığı noktanın üst kısmında negatif bir değer bulunması(FP) ya da alt kısmında pozitif bir değer bulunması(FN) hata yapıldığını göstermektedir.

Lojistik Regresyon modlinden hatırlanacağı gibi değere karşılık gelen çizgi noktasına bakılarak en yakın değere ataması yapılmaktadır. Örneğimizde görüleceği gibi 1 olması gereken #2 değeri 0'a atanırken #3 değeri ise tam tersi bir durumla karşılaşmaktadır.


Logistic Regression
Pozitif (P)
1
0.5
Negatif (N)
#1
#2
#3
#4
TP
FN
TN
FP


Karar Ağacı ile Tahmin
(Decision Tree)
E
K
Pozitif (P) : Erkek
Negatif (N) : Kadın
Boy 183
Kilo 85
Boy 145
Kilo 75
Boy 165
Kadın: 170, 75
Kadın : 160, 70
TN
FN
Erkek: 190, 95
Erkek : 190, 80
TP
FP

Ders 69: Netlik/Doğruluk (Accuracy) Paradoksu

Bu paradoksta özelinde ZeroR algoritması incelenmektedir. Bu algoritmaya göre veri kümemizdeki sınıflara bakarak istatiksel olarak çoğunluğa sahip sınıfı sınıflandırma sonucu olarak kullanmayı hedeflemektedir. Bu değerlendirme sürecinde hiçbir özniteliğe bakılmayarak hiçbir öğrenme yapılmayarak sadece sonuçlandırma yapılmaktadır.

Örnek olarak normalde %95.5 kesinliğe sahipken ZeroR ile %99 kesinliğe sahip olunmaktadır.

Accuracy Paradox
Doğruluk Paradoksu

	Tahmin C_1	Tahmin C_2
Gerçek C_1	9900	0
Gerçek C_2	100	0

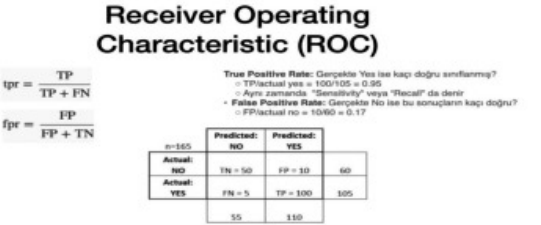
ZeroR : algoritması

	Tahmin C_1	Tahmin C_2
Gerçek C_1	9500	400
Gerçek C_2	50	50

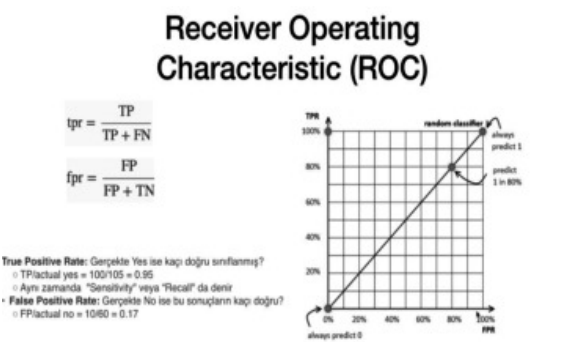
Bu örnek bağlamında bazı çıkarımlar yapılmaktadır. İlk olarak, sadece kesinliğe bakılarak sınıflandırma algoritmalarını değerlendirmenin doğru olmayabileceğidir. İkinci olarak ise, bir sınıflandırma algoritmasını başarılı olarak kabul edebilmemiz için başlangıç değeri ya da bir başka değişle alt çizgiye ihtiyacımız var olmasıdır. Algoritmamızın bu çizgiye göre konumuna bakılarak hakkında yorum yapılabilir. Örnek olarak algoritmanın bu çizginin altında kalması durumunda algoritma hakkında öğrenim yapabilen bir algoritma olmadığı sonucuna ulaşılabilir. Bu durumda algoritmanın tekrar değerlendirilmesi gerekmektedir.

Ders 70: ROC Eğrisi

Bu derste sınıflandırma algoritmalarının nasıl karşılaştırılabileceğinden bahsedilecektir.

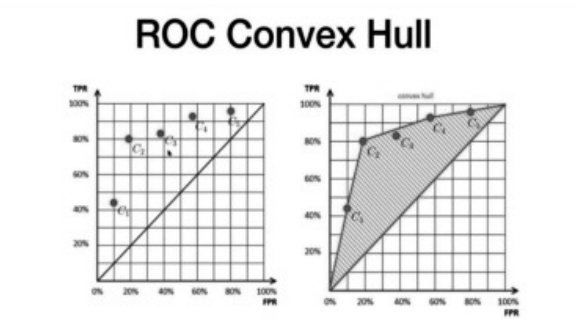


Bu örnek bağlamanında FPR'in %0 olma durumda ve TPR'in %100 olma durumda her değer için %100 doğruluk oranında tahmin yapılmakta olduğu bulunmaktadır.

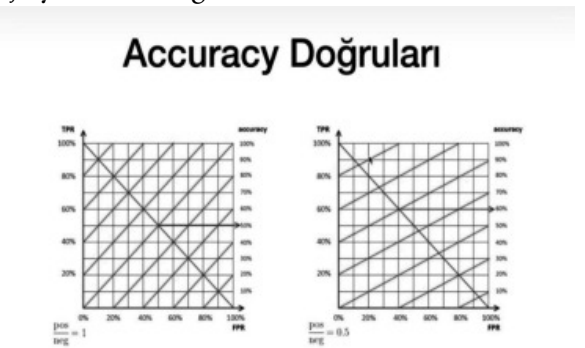


Roc ile sınıflandırma algoritmalarını karşılaştırmak mümkündür. Örnek olarak C1, C2, C3, C4, C5 değerlerinin sınıflandırma algoritma türlerini temsil ettiği varsayılmakla birlikte şekildeki gibi TPR ve FPR değerlerine göre konumlandırılmıştır. Bu koordinat noktalarının doğrusal olarak birleştirilmesi durumunda bazı algoritmaların oluşturulan şekil içerisinde kaldığı görülmektedir. Örneğimizde bu değer C3 olmakla birlikte, C2 ve C4 ile domine

edilmesinden dolayı kullanımı için herhangi bir neden kalmamaktadır.

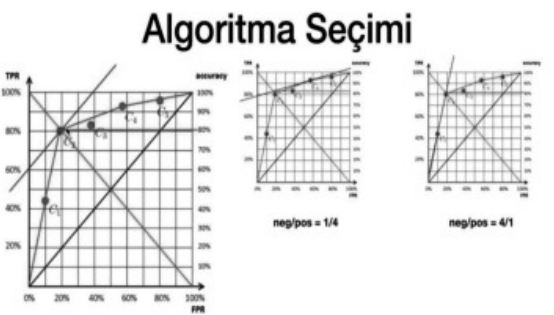


Kesinlik doğruları TPR VE FPR arasındaki ilişkiye göre belirlenmektedir.



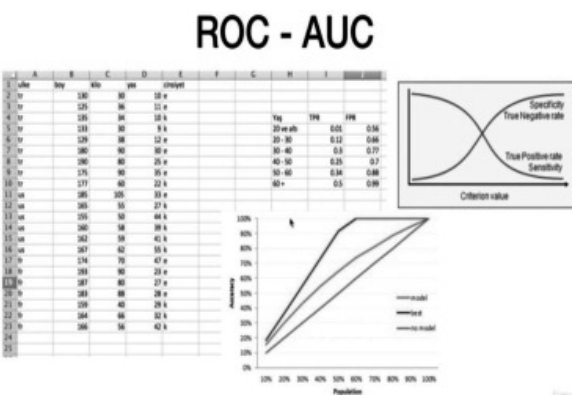
Algoritma seçimi yapılırken kesinlik doğrusuna paralel bir doğru çizilerek her algoritma için doğruluk değeri bulunulabilir. Örnekte bu doğruluk, C2 için %80 olurken C4 için %70 gibi bir değer bulunmaktadır. Ayrıca farklı negatif/pozitif değerleri için kesinlik değişebilmektedir. İlk şart altında C4 en iyi algortima olarak bulunurken tam tersi durumunda ise C1

ve C2 birbirine yakın ve en iyi sonuçları vermektedirler.



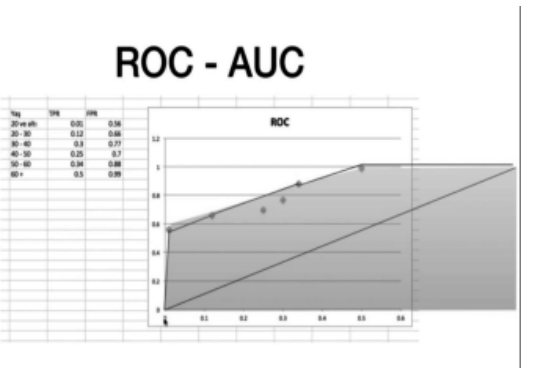
Bu bilgilere ek olarak, kullanılan değerlerin türü ile kullanılan algoritma arasındaki ilişki bulunulabilinmektedir. Örneğin, pozitif örnek sayısı arttıkça C5'in kesinlik durumu C2'ye göre daha hızlı artacaktır. Tam tersi durumda ise C1'in kesinlik durumu C2'ye göre daha hızı artacaktır.

Ders 71: AUC Değeri



Verilerimizi ROC ile doğrusal olarak birleştirdikten sonra bu eğrinin altında kalan alanı bulmak algoritmamızın ne kadar hızlı tepki verebildiğini ölçmek için kullanılmaktadır. Min-

imum değer olarak mavi bölgenin alt tarafında kalan üçgen gösterilebilirken maksimum olarak 1x1 alanındaki kare göterilmektedir. Modelimizdeki değerler üçgen ve kare arasında değişkenlik göstermektedir. Kapsanılan alan arttıkça başarı değeri de artış göstermektedir çünkü yukarıdaki "1" değerine yaklaşmak hedeflenen sonuçlardan birisidir.



Bölüm 19:

Ders 72: Sınıflandırma Şablonu

```
#1. kutuphaneler

import numpy as np


import matplotlib.pyplot as plt


import pandas as pd


from sklearn.metrics import confusion_matrix
```

#2. Veri Onisleme

```
#2.1. Veri Yukleme


veriler = pd.read_csv('veriler.csv')


#pd.read_csv("veriler.csv")


x = veriler.iloc[:,1:4].values #bağımsız değişkenler


y = veriler.iloc[:,4:].values #bağımlı değişken


print(y)
```

#verilerin egitim ve test icin bolunmesi

from sklearn.cross_validation import train_test_split

x_train, x_test,y_train,y_test = train_test_split(x,y,test_size=0.33, random_state=0)

```
#verilerin olceklenmesi


from sklearn.preprocessing import Standard-
Scaler
```

```
sc = StandardScaler()


X_train = sc.fit_transform(x_train)


X_test = sc.transform(x_test)
```

- **Buradan itibaren sınıflandırma algoritmaları başlar**

- **1. Logistic Regression**

```
from sklearn.linear_model import LogisticRegression


logr = LogisticRegression(random_state=0)


logr.fit(X_train,y_train) #egitim


y_pred = logr.predict(X_test) #tahmin


print(y_pred)


print(y_test)


#karmasiklik matrisi
```

```
cm = confusion_matrix(y_test,y_pred)

print(cm)

# 2. KNN

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=1,
metric='minkowski')

knn.fit(X_train,y_train)

y_pred = knn.predict(X_test)
```

```
cm = confusion_matrix(y_test,y_pred)

print(cm)

# 3. SVC (SVM classifier)
from sklearn.svm import SVC

svc = SVC(kernel='poly')

svc.fit(X_train,y_train)

y_pred = svc.predict(X_test)
```

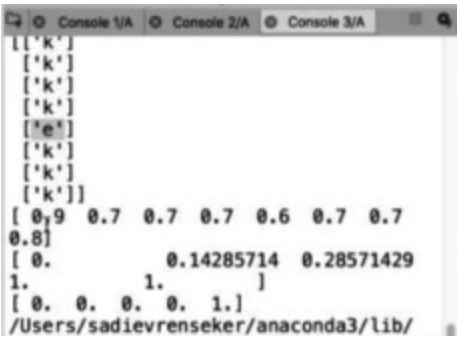
```
cm = confusion_matrix(y_test,y_pred)

print('SVC')

print(cm)

# 4. NAive Bayes

from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X_train, y_train)
```

```
y_pred = gnb.predict(X_test)


cm = confusion_matrix(y_test,y_pred)
print('GNB')


print(cm)


# 5. Decision tree


from sklearn.tree import DecisionTreeClassifier


dtc = DecisionTreeClassifier(criterion = 'entropy')
dtc.fit(X_train,y_train)
```

```
y_pred = dtc.predict(X_test)


cm = confusion_matrix(y_test,y_pred)


print('DTC')
print(cm)


# 6. Random Forest


from sklearn.ensemble import RandomForestClassifier


rfc = RandomForestClassifier(n_estimators=10, criterion =
'entropy')
```

```
rfc.fit(X_train,y_train)


y_pred = rfc.predict(X_test)


cm = confusion_matrix(y_test,y_pred)


print('RFC')


print(cm)
# 7. ROC , TPR, FPR değerleri


y_proba = rfc.predict_proba(X_test)
print(y_test)
```

```
print(y_proba[:,0])


from sklearn import metrics


fpr , tpr , thold =
metrics.roc_curve(y_test,y_proba[:,0],pos_label='e')
print(fpr)


print(tpr)
```


Console 1/A
Console 2/A
Console 3/A
[['k']
['k']
['k']
['k']
['e']
['k']
['k']
['k']]
[0.9 0.7 0.7 0.7 0.6 0.7 0.7
0.8]
[0. 0.14285714 0.28571429
1. 1.]
[0. 0. 0. 0. 1.]
/Users/sadievrenseker/anaconda3/lib/

Sırasıyla test için seçilen değerler, değerlerin verilen duruma göre bulunabilme oranları ve false positive değeri görülmektedir.

Ders 73: Ödev 3: İris veri kümesi ve sınıflandırma algoritmaları

İlk olarak, şablonumuz içerisindeki veri yüklemedeki " read_csv" kısmını ".xls" uzantısı için excel formatı için hazırlıyoruz.

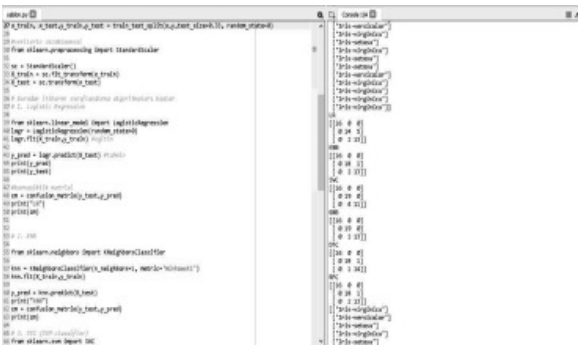
```
veriler = pd.read_excel('iris.xls')
```

Bu veri kümesi yapraklar üzerinden çalıştırılmış bir veri kümesidir. Veri kümesinde 150 adet satır

bulunmaktadır. Verilen satırdaki ilk 4 değer üzerinden yaprağın hangi sınıfa ait olduğu saptanmak istenmektedir.

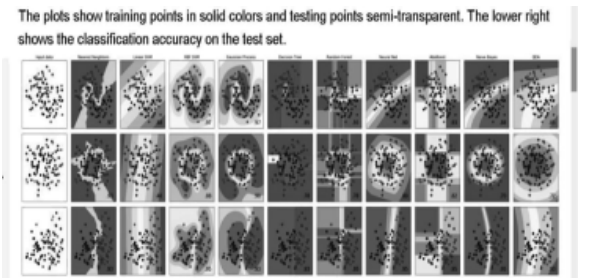
Bu ödevde verilen görev, sınıflandırma yapılarak hangi sınıflandırma algoritmasının daha iyi olduğunu bulmaktır. Bunun için ekstra konfigürasyon yapılabilir. Konfigürasyon örnekleri için KNN'deki k değeri veya DVM'ndaki kernel fonksiyonu olarak hangi parametrenin kullanılacağı gibi durumlar gösterilebilir.

Ders 74: Ödev 3: Çözümü



Sonuçlarda görüldüğü üzere DTC ve GNB 2 hata ile en başarılı sonucu döndürülmektedir. Örnek olarak test ve eğitim kümesinin 0.24 olarak dağıtılması bir çok yöntemdeki hata miktarını %0'a indirildiği görülmüştür. Bir başka konfigürasyon örneği olarak RFC'deki criterion= "gini" durumunda hata miktarı azalmaktadır. Bu ve buna benzer konfigürasyonlar ile

verilen eğitim kümesi için hangi sınıflandırma algoritmasının kullanılabileceğine dair tecrübe kazanılanılabilir.



(https://scikit-learn.org/0.20/auto_examples/classification/plot_classifier_comparison.html [14])

14. https://scikit-learn.org/0.20/auto_examples/classification/plot_classifier_comparison.html

Ders 75: Sınıflandırma Algoritmaları Karşılaştırma ve Özet

Sınıflandırma Algoritmaları Karşılaştırma ve Özet

Bölüm 19, Ders 75

Bölüm 19'a 7 sınıflandırma modelini öğrendiniz. Daha önce, 2. Bölümde Regresyon için sorduğumuz aşağıdaki soruları bu bölüm sonunda da sorup yanıtlıyoruz:

1. Her modelin artıları ve eksileri nelerdir?
2. Problemim için hangi modeli seçeceğimi nasıl bileceğim?
3. Bu modellerin her birini nasıl geliştirebilirim?

Yeniden bu soruların her birine tek tek cevap verelim:

1. Her modelin artıları ve eksileri nelerdir? Burada her sınıflandırma modelinin tüm artılarını ve eksilerini veren bir özet sayfası bulabilirsiniz (bunaya bilayım).

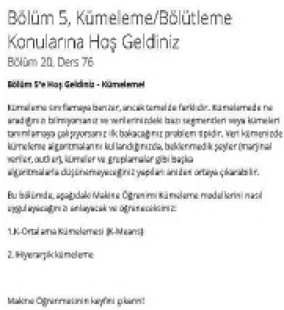
2. Problemim için hangi modeli seçeceğimi nasıl bileceğim? Regresyon modelleri için olduğu gibi, problemimizin doğrusal mi yoksa doğrusal olmayan özellikte mi olduğunu anlamamız gerekir. Bunu Bölüm 15 - Model Seçimi dersinde öğrenmiştiniz. Sonraç Eğer problemimiz doğrusal ise, Lojistik Regresyon veya SVM'yi kullanabilirsiniz. Eğer problemimiz lineer değil ise, K-NN, Naif Bayes, Karar Ağacı veya Rastgele Ormanı'nı kullanabilirsiniz. Peki hangisini her durumda seçmeliyim? Bunu Bölüm 17'de Model Seçimi k-Katlama Çapraz Doğrulama ile daha detaylı anlatıyor olacağız.

Algoritmalara farklı problemler açısından bakıldığında şunları kullanmayı tercih edebilirsiniz: - Tahminlerinizi olasılıklarınız göre sıralamak istediğinizde Lojistik Regresyon veya Naive Bayes. Örneğin, müşterilerinizi belirli bir ürünü satın alma olasılığı en yüksek olanından en düşük olanına doğru sıralamak istiyorsanız. Bu tapsilla, pazarlama kampanyalarınızı hedeflemiş müşterilere yönlendirebilirsiniz. İki tabi ki bu tür bir iş problemi için, eğer problemimiz doğrusal ise Lojistik Regresyonunu, ve problemimiz doğrusal değilse Naive Bayesi kullanabilirsiniz. - Müşterilerinizin hangi segmente ait olduğunu tahmin etmek istediğinizde SVM. Segmentler, örneğin, her türlü segment olabilir. Örneğin daha önce kimlerine ilk tanımladığınız (nasıl pazar segmentleri. - Model sonuçlarınızı net bir şekilde yorumlamak istediğinizde Karar Ağacı - Rastgele Orman gibi yöntemleri yine örneğin sadece daha az parametreye ihtiyacınız sahip yüksek performanslı algoritmalar anyonsanız kullanabilirsiniz.

3. Bu modellerin her birini nasıl geliştirebilirim? 2. Bölümdeki ile aynı cevabı vermemiz gerekecek: Bölüm 10 - Model Seçiminde, ayarlarınızı yaparak, modelinizin performansını iyileştirmenize bakacağız, yine bu bölümün altında, Parametre Ayarlamaya ayrılmış ikinci bölümü bulacaksınız. Büyük ölçüde bu, aslında her modelin iki tür parametreden oluştuğunu fark etmeyenizdir: öğrenilen parametreler, örneğin Doğrusal Regresyondaki katsayılar ve hiperparametreler. Hiperparametreler öğrenilmeyen ve model üzerinizdeki sabit değerleri olan parametrelerdir. Örneğin, lambda regülarizasyon parametresi veya panalti parametresi C hiperparametrelerdir. Şimdiye kadar bu hiperparametrelerin varsayılan değerini kullandık ve modelimiz daha yüksek performansı gösterirdi diye bunlarla optimum değerleri için araştırma yapmadık. En uygun değerlerini bulmak, Parametre Ayarının tam olarak ne olduğuyla ilgilidir. Bu nedenle, model performansınızı geliştirmek ve bazı parametreler ayarlamayı yapmakla ilgilenenler için, doğrudan Bölüm 10 - Model Seçimine atlamak iyi bir fikir olabilir.

Bölüm 20:

Ders 76: Kümeleme/Bölütleme Konularına Hoş Geldiniz



Bölüm 5, Kümeleme/Bölütleme Konularına Hoş Geldiniz
Bölüm 20, Ders 76

Bölüm 5'e Hoş Geldiniz - Kümeleme!

Kümeleme sınıflamaya benzer, ancak temelde farklıdır. Kümelemede ne aradığınızı bilmiyorsanız ve verilerinizdeki bazı segmentleri veya kümeleri tanımlamaya çalışıyorsanız ilk bakacağınız problem tipidir. Veri kümenizde kümeleme algoritmalarını kullandığınızda, beklenmedik şeyler (marjinal veriler, outlier), kümeler ve gruplamalar gibi başka algoritmalarla düşünemeyeceğiniz yapıları aniden ortaya çıkarabilir.

Bu bölümde, aşağıdaki Makine Öğrenimi Kümeleme modellerini nasıl uygulayacağınızı anlayacak ve öğreneceksiniz:

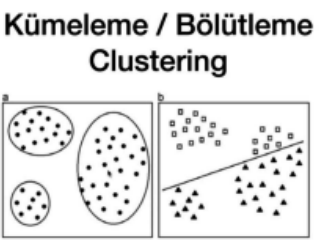
1.K-Ortalama Kümelemesi (K-Means)

2. Hiyerarşik kümeleme

Makine Öğrenmesinin keyfini çıkarın!

Ders 77: Gözetimsiz Öğrenmeye Giriş (Unsupervised Learning)

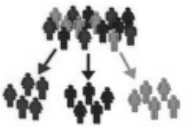
Bölütleme, Sınıflandırma ve Tahmin problemlerinden farklıdır. Bu problemler veri madenciliğinin fazla gelişmemiş olduğu problemlerdir fakat günümüzde görüntü işleme, müşteri segmentasyonu, Pazar segmentasyonu gibi konularda etkinlik göstermektedir.



Şekil a' da görüldüğü gibi aynı tür sembolleri birbirine yakınlığına göre parçalara bölünmeye çalışılınmaktadır. Şekil b'deki Sınıflandırma problemlerinde ise farklı tip semboller bulunmakla beraber yapılmak istenilen işlem sınıfların belirlenmesi ve yeni gelecek olan değerlerin hangi sınıfa dahil olması gerektiğinin saptanmasıdır. Bu nedenden dolayı Şekil a'da gözetimsiz eğitimden bahsedilmektedir. Bu eğitim türünde kaç sınıf bulunduğu veya gösterilen şekillerin hangi sınıfa ait olduğu bilinmemektedir. Sınıflandırma problemlerinde ise ön sınıf tanıma ve sonrasında ise makinenin halihazırdaki sınıfları öğrenilmesi görülmektedir. Bu tür eğitim ise gözetimli olarak adlandırılabilir.

Kümeleme / Bölütleme
Clustering

- Müşteri Segmentayonu
- Pazar Segmentasyonu
- Sağlık ve Görüntü işleme



Müşteri Segmentasyonu kısmında aynı demografik sınıfa dahil olan kişilerin benzer satın alımlar yapabileceği düşünülmektedir. Örnek olarak metro kullanan kişilere özel bir ürünü metro duraklarında reklamlar ile pazarlama yapmak gösterilebilir.

Tehdit kısmı olarak diğer bölütlerin dışında kalan "outlayer" veri gösterilebilir. Bu veriler bölütleme ile kolayca saptanabilmektedir.

Özet olarak, verinin alt kümesi üzerinde yapılan bütün işlemlerin uzmanlaşılmış bir şekilde segment üzerinden yapılması mümüküdür. Örnek olarak çalışan segmentasyonu bazında mutlu-mutsuz çalışanlar saptanabilinmektedir.

Davranışsal Segmentasyon'a örnek olarak android işletim sistemi nedeniyle müşterinin

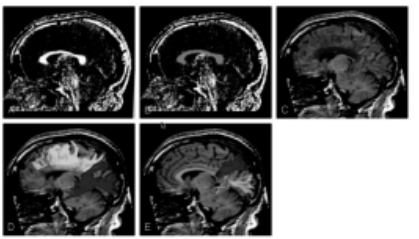
ücretsiz ürünler, özgür yazılım gibi isteklerinden dolayı tercih ederken, ios kullanıcıları ise uygulama için ücret ödeyen premium kullanıcılardan oluşmaktadır.

Demografik Segmentasyon için ise müşterinin yaşı, cinsiyeti veya eğitim durumu gibi veriler gösterilebilir.

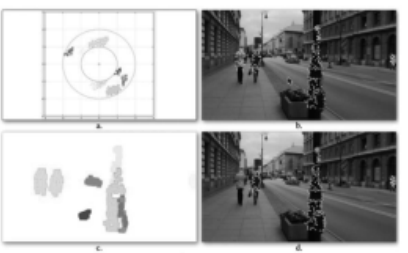
Psikolojik Segmentasyon ise müşterilerin beklentileriyle ilgili değerler içermektedir ve örnek olarak giyim tarzı gösterilebilir. Örnek olarak retro veya son moda vb. tekstil ürünleri satan mağazaların tercihi gösterilebilir.

Coğrafi Segmentasyon için verilerin ülkelere ve şehirlere göre dağılımı gösterilebilir.

Sağlıkta Kümeleme



Bilgisayar ile Görü / Kümeleme

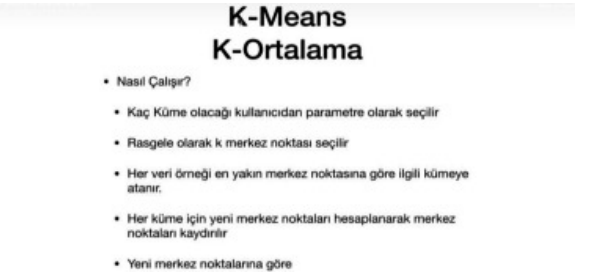


PYTHON İLE MAKINE ÖĞRENMESI405

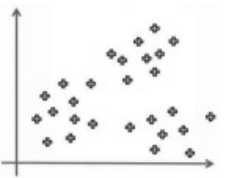
Bilgisayar ile Görü /
Kümeleme

Bölüm 21:

Ders 78: Algoritmaya ve Kavramlara Giriş

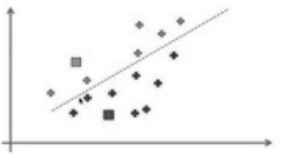


Veri kümemizin gözetimsiz olarak dağıldığını düşünelim.

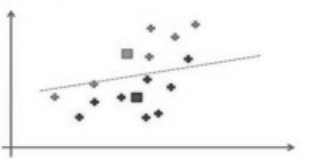


Uzayda rastgele iki nokta belirleniyor.

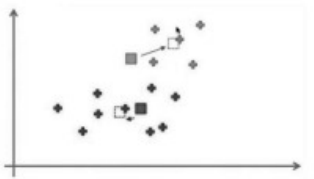
Bu iki nokta arasından doğru geçiriliyor.



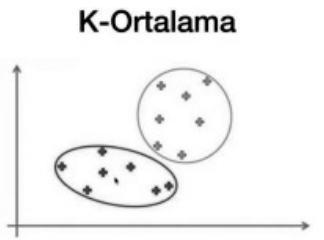
Bölütlemeden sonra kümelerin ağırlık noktaları belirleniyor ve daha önce belirlediğimiz noktalar bu bu koordinatlara kaydırılıyor.



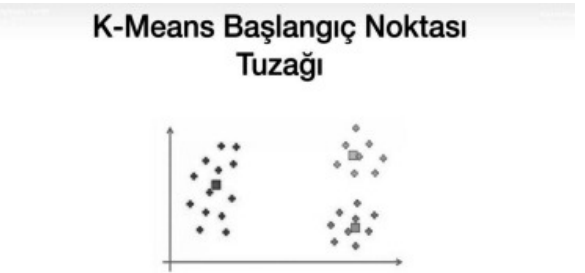
Sonrasında ise aynı işleme devam ediliyor.



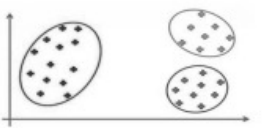
Kararlı bir uzay oluşuncaya işlem sona eriyor.


K-Ortalama

Ders 79: Rassal Başlangıç Tuzağı

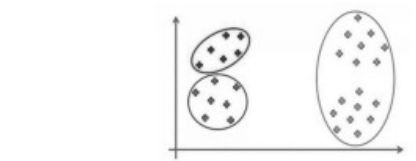


Bölütleme sonunda aşağıdaki durum gözlenmektedir.



Bu derste inceleyeceğimiz konu ise rastgele konumlanan başlangıç noktalarının istenilen gibi dağınılmadığı durum göz önüne alınılacaktır.

Bölütleme sonunda aşağıdaki durum gözlenmektedir.



Elde edilen iki bölütleme sonucundan ilk olan bölütleme daha başarılıdır ve bunun nedeni bölütlerin iki ucu arasındaki mesafe gösterilmektedir. Bu durum "saflık" olarak adlandırılımaktadır.

Herhangi bir algoritma üzerinde rastgelelik var ise bir problem oluşması yüksek bir ihtimaldir. Bu konu özelindeki üretilen problem çözümlerinden biri ise, her veri noktasının teker teker merkez noktası olarak seçilmesidir. K-ortalamaya göre hata yapmama ihtimali yüksek olmakla beraber ortaya çıkardığı işlem gücü basit bir durum değildir. Örnek olarak 1.3 milyar kullanıcıya sahip bir sosyal medya platformu kullanıcılarını segmente etmek istediğinde işlem gücü standartın çok üstünde olmaktadır(3 bölütleme için = 3^1.3M). Bu durum için K-ortalama++ algoritması geliştiriilmiştir.

K-Means ++
1. Rasgele seçilen noktalardan en yakınına her noktadan uzaklığı hesapla (buna D(x) diyelim)
2. Yeni noktaları mesafenin karesini olasılık olarak ($D(x)^2$ ile bul)

Bu durumda oluşabilecek yeni merkez noktaları halihazırdaki veri noktalarından biri olması düşünülmektedir. Rastgele seçilen merkez noktanın kendisine en yakın potansiyel noktaya olan uzaklığına D(x) denilmektedir. Rastgele merkez noktaya daha uzak noktada bulunan veri noktasının yakın olana göre yeni merkez noktası olma ihtimali daha fazladır(D(x)^2 nedeniyle).

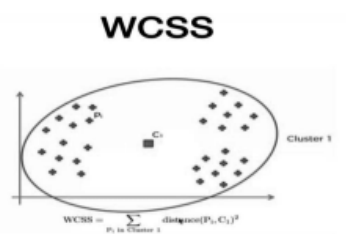
Ders 80: Küme/Bölüt Sayısına Karar Vermek

Bilindiği üzere bölütleme uygulamaları gözetimsiz olmakla beraber veri için herhangi bir ön bilgi bulunmamaktadır ve bölütleme işlemi makine tarafından yapılması beklenmektedir. X-ortalama olarak bilinen algoritmada ise verilen k değeri değişkenlik gösterilmekte ve en optimum bölütleme modelindeki k değerinin(bölüt sayısı) bulunması amaçlanmaktadır fakat buna rağmen

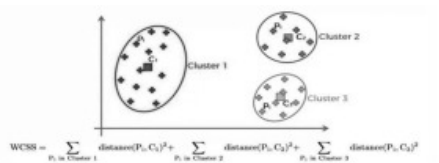
veri bilimcisinin gözlem yaparak bu modelin başarısına ilk elden karar vermesi gerekmektedir. Literatürde bazı çözümler vardır.

$$WCSS = \sum_{P_i \text{ in Cluster 1}} distance(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} distance(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} distance(P_i, C_3)^2$$

Bu formülde her bir bölüt içindeki tüm değerlerin, ait olduğu bölütün merkezine olan uzaklığının kareleri alınmakta ve toplanmaktadır. Daha basit bir görünüm için tek bir bölüt ele alınabilir.

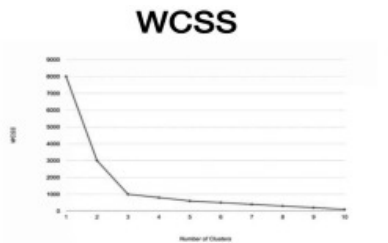


Çoklu bölütleme durumu ise örnekteki gibidir.



Bölüt sayısı arttıkça WCSS değeri azalmaktadır. Eğer makineye bu değeri optimize etmesi emredilirse makine sürekli olarak bölüt sayısını arttıracaktır ve son olarak her bir veri noktası bir bölüt oluşturacak ve WCSS değeri sıfırlanacaktır. Bu durum aşırı öğrenmeye sebep olacak ve işlemimiz öğrenme yerine ezberleme ile sonuçlanacaktır. Fakat gözlemci olan bir veri bil-

imcisi bu bölütlemenin grafiğine bakarak eğimin aniden değiştiği, kabaca "dirsek noktası" olarak adlandırılan noktayı seçecektir.



Şayet k değeri hassas veya daha önceden teknik nedenlerden dolayı herhangi bir ön tanımlı standart değer belirlemesi yapılmadıysa optimum durum için k değeri 3 olarak seçilmektedir.

Ders 81: Python ile K-Means Kodlaması

İlk olarak veri tablosu internet adresinden temin edilmelidir.

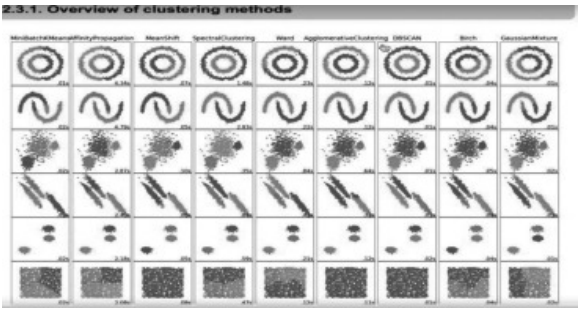
(https://www.udemy.com/makine-ogrenmesi/learn/v4/t/lecture/10142914?start=15 [15])

Bölütleme algortimalarının daha iyi anlaşılabilmesi için diğer yöntemleri de incelemeli ve araştırmalıyız.

(https://scikit-learn.org/stable/modules/clustering [16])

15. https://www.udemy.com/makine-ogrenmesi/learn/v4/t/lecture/10142914?start=15

16. https://scikit-learn.org/stable/modules/clustering



Şekildeki ilk kolon K-ortalama olarak düşünülebilir.

K-ortalama özelinde de literatür taraması yapılmalıdır ve bu yötemin düzgün çalışabilmesi için gerekli parametreler tanınmalıdır.

(https://scikit-learn.org/0.20/modules/generated/sklearn.cluster.KMeans.html [17])

Kod kısmı olarak;

Kütüphanelerimizi çağırıyoruz.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
```

17. https://scikit-learn.org/0.20/modules/generated/sklearn.cluster.KMeans.html

Veriler okunuyor.

```
veriler = pd.read_csv('musteriler.csv')
```

Verilerin içinde cinsiyet, id no , yaş, iş hacmi ve maaş gibi değerleri görmekteyiz.

Veri kümesi oluşturuluyor ve kişilere bağlı olarak maaş ve hacim değerleri alınarak örneğimiz için iki boyutlu uzayda konumlandırılıyor.

```
X = veriler.iloc[:,3:].values
```

Kütüphanemizdeki K-ortalama çağrılıyor.

```
from sklearn.cluster import KMeans
```

Önceki şablonlarımızda kullandığımız obje atama yöntemini kullanıyoruz.

Dökümantasyon kısmında da incelediğimiz üzere bölüt sayısı ile hangi yöntemin kullanılacağı gibi parametreleri tanımlıyoruz.

```
kmeans = KMeans ( n_clusters = 3, init = 'k-means++')
```

Veriler eğitiliyor.

kmeans.fit(X)

Algoritmadaki bölütleme merkezlerini görüntülüyoruz.

```
print(kmeans.cluster_centers_)
```

Bölütleme merkezlerindeki ilk noktalara işlem hacmini belirtirken ikinci nokta ise maaş değerini göstermektedir.

Bölütleme sayımızı değiştirmek için 1 ile 10 arasında bir döngü yaratıyoruz.

```
for i in range(1,11):
```

```
kmeans = KMeans (n_clusters = i, init='k-means++', random_state= 123)
```

```
kmeans.fit(X)


sonuclar.append(kmeans.inertia_)
```

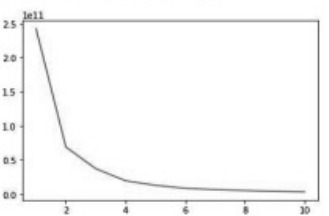
Döngü her başladığında aynı değer ile başlaması için random_state=123 tanımlanıyor.

WCSS değerlerimizi toplamak için ".inertia_" kullanıyoruz ve bu değerleri sonuçlar kısmında topluyoruz.

Bu değerlerimizi grafiksel olarak görüntülüyoruz.

plt.plot(range(1,11),sonuclar)

```
[[ 23539.13043478   5272.01086957]
 [ 63915.27777778   6140.625     ]
 [109905.55555556   7325.69444444]]
```

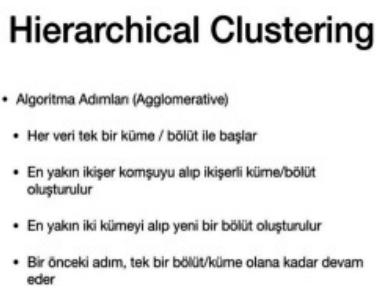


Bir gözlemciye göre dirsek noktası saptanma potansiyeli yüksek olarak belirlenen değerler 2, 3 veya 4'tür.

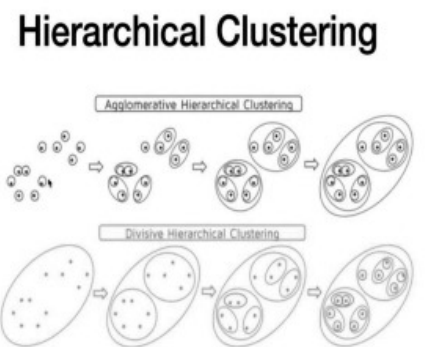
Bölüm 22:

Ders 82: Kavrama ve Algoritmaya Giriş

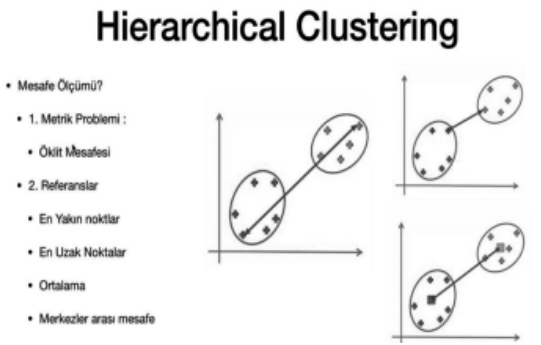
Hiyerarşik Bölütleme yöntemi iki farkı yaklaşıma sahiptir. Bu yaklaşımlar Agglomerative (aşağıdan yukarıya çalışan) ve Divisve (yukarıdan aşağıya çalışan) sistemlerdir.



İki yaklaşımın aralarındaki farkı incelemek üzere aşağıdaki görsel incelenebilir.



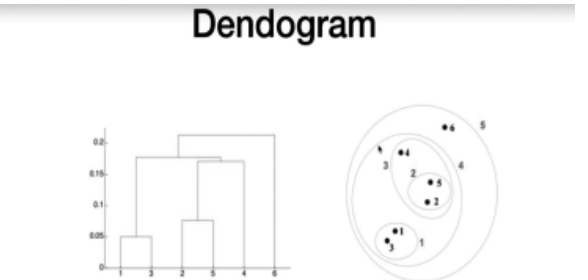
Bu yöntemler bazında karşılaşabileceğimiz bazı problemler bulunmaktadır. Bu problemler, veri noktaları arasındaki ve bölütlerin kendi arasındaki mesafelerin hangi uzaklık metrikleri ile ölçüleceği ile ilgilidir.



Ders 83: Hiyerarşik Bölütleme ve Dendogram Kavramı

Dersimize öncelikle Dendogram kavramının tanım ile başlıyoruz.

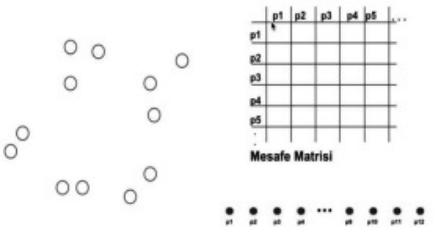
Şekilimizdeki siyah noktaları veri noktaları olarak kabul edelim. Agglomerative yaklaşımını ve öklid mesafesi değerlerini kullanalım.



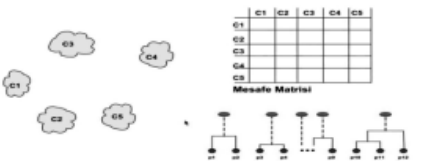
Şekildeki grafiğimizde veri noktalarını ve aralarındaki mesafelerin bağlantısını görmekteyiz. Mesafeleri bir ayar çubuğu olarak düşündüğümüzde ise yukarı hareket ile daha fazla bölütleme yapılacağı ve sonuç olarak bir bölüt elde edileceği görülmektedir. "Hiyerarşik bölütleme" kavramı bu mesafelerin etkilerine bağlı olarak ortaya çıkmaktadır.

Önceki örneğimizin aksine bu mesafelerin saptanması makine için mesafe matrisi ile

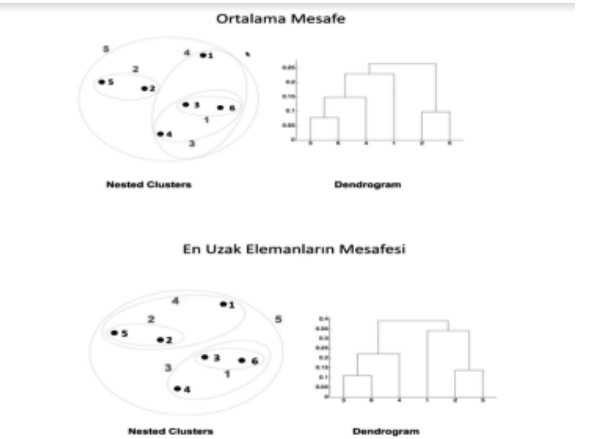
mümkündür. Her veri noktası ile diğer noktalar arasındaki mesafeler hesaplanarak simetrik bir biçimde matrise yerleştiriliyor.



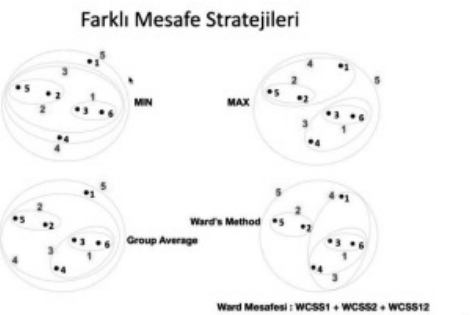
Sonrasında ise daha zor bir işlem olan bölütler arası mesafe hesaplaması yapılmalıdır. Önceki dersimizde öğrendiğimiz "referanslar" kısmı bu işlem için anahtar değer taşımaktadır.


C1
C2
C3
C4
C5
Mesafe Matrisi

Bölütleme işlemi için farklı referanslar ile işlem yapılıyor. Örneğin, ortalama mesafe ve en uzak elemanların mesafeleri gibi değişkenler ile bölütleme yapılabilir.



Farklı referanslar ile bölütleme yapılması durumunda aşağıdaki sonuçlar ortaya çıkmaktadır.

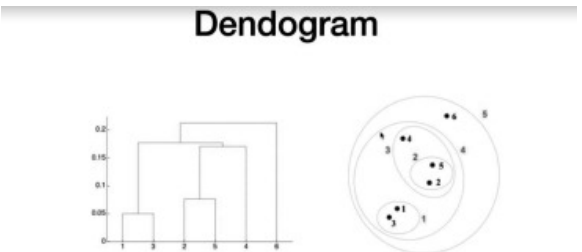


Farklı durumlarda farklı referanslar kullanılması gerekmekle birlikte her zaman geçerli olan en iyi bölütleme yöntemine varılamamaktadır.

Sonuçlar arasındaki Ward yöntemi ise sadece bölütler arasındaki mesafeyi ölçmek için kullanılmaktadır. Örneğin iki bölüt arasındaki Ward mesafesi hesaplanırken: ilk kümenin WCSS'i ,ikinci kümenin WCSS'i ve bu kümelerin birleşimi olan kümenin WCSS'i toplanmaktadır.

Dendogram kullanımının ana fikri, kullanıcının tanımladığı bölüt sayısına göre bu bölütlemenin nereden yapılacağını saptamaktır.

Hiyerarşik bölütlemede de K-ortalama yöntemindeki gibi bir dirsek noktası bulmak mümkündür. Örnek olarak şeklimizde yatay bir çizgi çizebileceğimiz en uzun mesafe 0.15 noktasının biraz üzeridir.



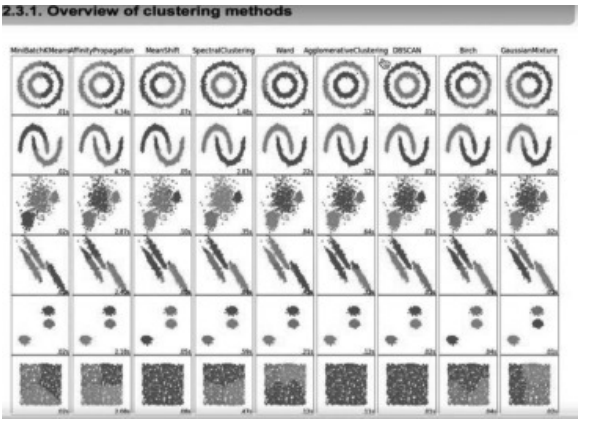
Sonuç olarak örneği inceleyecek olursak, en optimal bölütleme sayısı 4'e eşittir.

Ders 84: Hiyerarşik Bölütleme ve Dendogramların Kullanılması

İlk olarak literatür taraması yapılmalıdır.

(https://scikit-learn.org/stable/modules/clustering [18])

18. https://scikit-learn.org/stable/modules/clustering


2.3.1. Overview of clustering methods
MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
Birch
GaussianMixture

(https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering [19])

Sonrasında ise kodlama bölümüne geçiliyor.

Kütüphanemizden yöntemimizi çağırıyoruz.

```
from sklearn.cluster import AgglomerativeClustering
```

Objeye tanımlanması yapılıyor. Bölüt sayısı, mesafe ölçüm için doğrusallık değeri ve iki küme

19. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering

arasındaki farkın hangi yöntem ile bulunacağını belirten parametrelerimiz bölütleme için belirleniyor.

```
ac = AgglomerativeClustering(n_clusters=4, affinity='euclidean', linkage='ward')
```

Veriler eğitiliyor ve tahmin yapılarak objeye atanıyor.

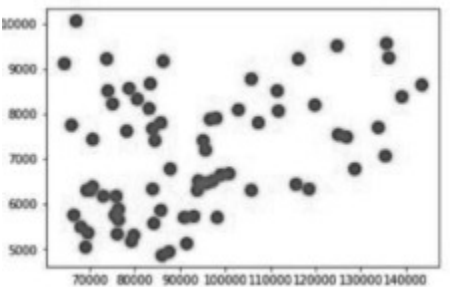
```
Y_tahmin = ac.fit_predict(X)
```

Tahmin sonuçlarını görüntülüyoruz.

```
print(Y_tahmin)
```

Detaylı görselliştime için tahmin ve sonuç değerleri alınıyor sonrasında ise renklendirme yapılarak grafik oluşturuluyor.

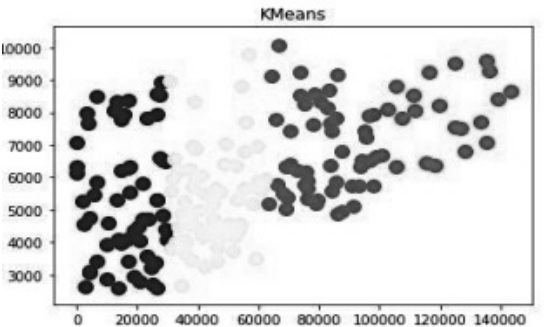
```
plt.scatter(X[Y_tahmin==0,0],X[Y_tah-
min==0,1],s=100, c='red')
```

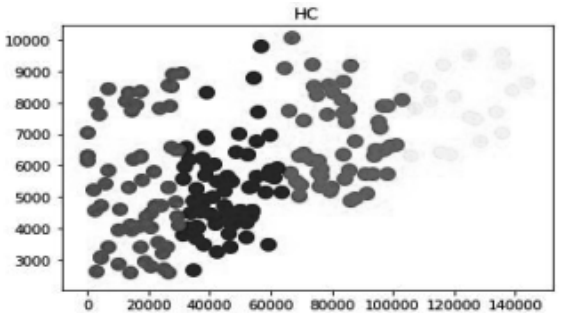


Farklı değerler için farklı renklendirme yapılıyor.

```
plt.scatter(X[Y_tahmin==1,0],X[Y_tah-
min==1,1],s=100, c='blue')

plt.scatter(X[Y_tahmin==2,0],X[Y_tah-
min==2,1],s=100, c='green')

plt.scatter(X[Y_tahmin==3,0],X[Y_tah-
min==3,1],s=100, c='yellow')

plt.title('HC')

plt.show()
```

Bu görselleştirmeyi K-ortalama değeri için de aynı kodlar ile uyguluyoruz.





Dendogram'ın görselleştirilmesi için ise farklı bir kütüphane olan scipy kullanılıyor.

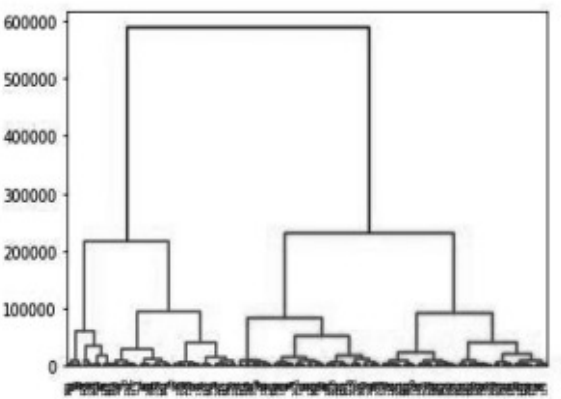
import scipy.cluster.hierarchy as sch

Dendogram tanımlanıyor ve yöntemimizdeki parametreler belirleniyor.

dendrogram = sch.dendrogram(sch.linkage(X, method='ward'))

Son olarak görselleştirme tamamlanıyor.

plt.show()



Çıkarım yapmak gerekirse, bölüt sayısının 2 alınması en uzun mesafeyi oluşturmakta ve 4

değeri ile birlikte bu değerler optimal potansiyele sahip bölütleme sayılarını oluşturmaktadır fakat bu değerin 3 alınması durumu ise hatalı bir sonuca neden olacaktır.

Ders 85: Bölüm Sonu ve Karşılaştırma

Bölüm Sonu ve Karşılaştırma

Bölüm 22, Ders 85

Bu bölümde, bölümleme / Kümeleme kavramına giriş yaparak, çok sık kullanılan ve önemli iki bölümleme / kümeleme algoritmasını python üzerinden kodlayarak örnek veri kümesi üzerinden uyguladık.

Bu bağlantıya tıklayarak, iki yöntemin karşılaştırmasını (artı ve eksi yönlerini) görebilirsiniz.

Tebrikler, bu bölümü tamamladınız, artık Birliktelik Kural Çıkarımı (Association Rule Mining) problemleri ve algoritmaları ile yolculuğunuza devam edebilirsiniz.

Bir sonraki bölümde başarılar.

Bağlantı adresi: http://www.bilkav.com/wp-content/uploads/2018/03/kumeleme_ozet.pdf

Bölüm 23:

Ders 86: Kavramlara Giriş

Bölüm 24:

Ders 87: Birliktelik Kural Çıkarımı Problemlerine Giriş

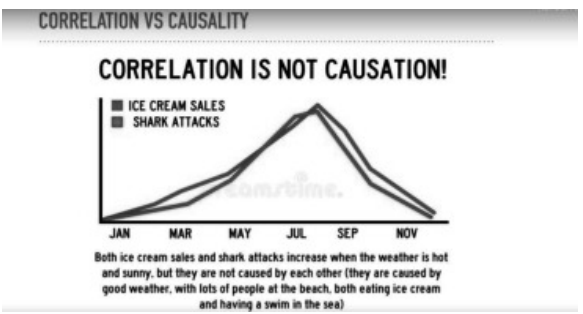
Bu bölümde problemlerimizi iki alt algoritma ile inceleyeceğiz. Bu algoritmalar literatürde Apriori ile Eclat isimleriyle bulunmaktadır.

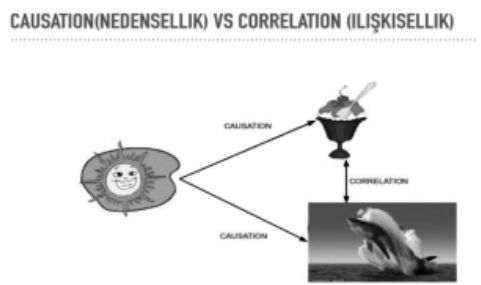
Bu kavramdaki amacımız, herhangi bir eylemin bir ürün üzerinden tekrarı durumunda bu eyle-

mi doğru bir şekilde belirlemektir. Bu şekilde hedef kitlesi ve ürün arasında bir bağlantı saptanarak kişiye özel tavsiye yapılması mümkün olmaktadır.

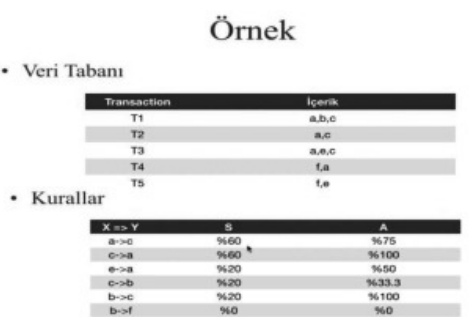
Bu kavram bağlamındaki tartışma konularından birisi ise "nedensellik mi yoksa ilişkisellik mi" sorusuna cevap aramaktadır. Makinalar için eylemler arasındaki ilişki nedenselliğe ihtiyaç duymadan ortaya çıkarılarak aksiyona dönüştürülebilirken insanlar için bu durum tam tersidir.

Örneğimizde köpek balığı saldırısı ile dondurma satışı sayılarının grafiği bulunmaktadır.





Veri tabanı üzerinden kuralların belirlenmesi sonucunda Destek ve Alfa denilen değerler ortaya çıkmaktadır.



Tek bir eylemden bahsedilirken Destek değeri yeterli olurken iki farklı eylemi birbirine bağlarken confidence(güvenilirlik) değerine bakılmaktadır.

Destek (Support)

Yüz kişiye sorduk

a ürününü aan 100 kişiye sorduk

B ürününü amak ne kadar artış sağlar?

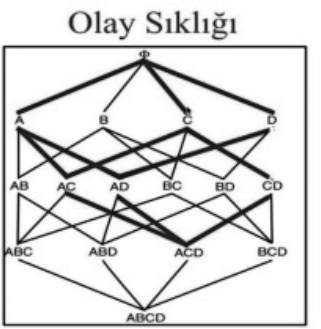
$$\text{Support(a)} = \frac{\text{a varlığını içeren eylemler}}{\text{Toplam Eylem Sayısı}}$$

$$\text{Confidence(a->b)} = \frac{\text{a ve b varlığını içeren eylemler}}{\text{a varlığını içeren eylemler}}$$

$$\text{Lift(a->b)} = \frac{\text{Confidence(a->b)}}{\text{support(b)}}$$

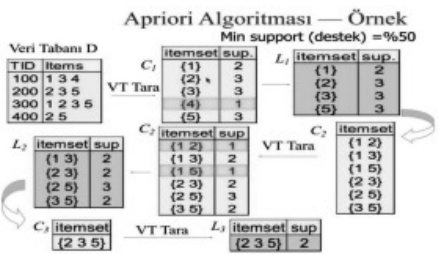
Lift durumunda ise eylemlerin birbiri üzerindeki olumlu etkileri saptanmaya çalışılmaktadır ve bu değerin 1'den büyük olması o eylemin diğer eylemi pozitif etkilediği anlamına gelmektedir.

Olay sıklığı tespit edilirken belirlenen destek değerin altında bulunan eylemler elenmektedir ve bunun nedeni ise,o eylemin gelecek destek değerlerini de düşük olmaya devam edecek olmasıdır.



Apriori algoritması basitçe frekans sayar olarak çalışmaktadır. Sistemdeki frekansları sayarken eylemlerin yüzdelik değerlerine göre eleme işlemi yapar.

Örnek olarak maksimum frekansa sahip olan 3 değerinin %50 si 1.5 olacağından dolayı 4 numaralı ürün elenmektedir. Sonrasında ise ürünleri birlikte değerlendirerek işleme devam ediliyor.



Sonuç olarak en yüksek talep potansiyeline sahip ürün paketi istendiği durumda 2, 3 ve 5

ürünlerinin bulunduğu paket doğru bir tercih olacaktır.

Son dönemlerde bu kural çıkarımı durumu , artan tüketici eylemlerinden dolayı bu eylemler arasında bağlantı kurmak isteyenlerin gözdesi haline gelmiş ve yatırımsal olarak ilgi çekici bir konu olmaya başlamıştır.

Apriori algoritması yönsüz çalışırken zaman serisi analizi durumda ise bir eylemin diğer eylemi tetikleme durumundan dolayı bu kural

çıkarımına yönlendirilmiş birliktelik kural çıkarımı denilmektedir. Örnek olarak borsa hareketleri veya dış siyaset hareketleri örnek olabilir.

Ders 88: Python ile Apriori Algoritmasının Kodlanması

Bu dersimizde sci-kit learn kütüphanesi kullanılamayacağından github üzerindeki kaynaklar kullanılmaya çalışılacaktır. Örnek olarak tarayıcıda " apriori python" aratıldığında çeşitli github yazarlarının kütüphanelerine erişilmektedir.

Bu kodlara ve kullanacağımız veri dosyasına sitemizden ulaşabilirsiniz.

Bütün dosyaların aynı dizinde olmasına dikkat edilmelidir.

(http://www.bilkav.com/makine-ogrenmesi-egitimi/ [20])

20. http://www.bilkav.com/makine-ogrenmesi-egitimi/

Kod kısmında ise alışılagelmiş şablonumuzu takip ediyoruz.

Kütüphanelerin yüklenmesi.

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd
```

Verilerimizin okunması. Hatırlayacağımız üzere “read_csv” kullanımı durumunda verilerimizde herhangi bir kolon başlığı bulunmamasından dolayı kolon başlığı kısmını None olarak tanımlıyoruz.

```
veriler = pd.read_csv('sepet.csv', header = None)
```

Algoritma incelenerek istenilen şekilde olan veri setimize ulaşmak için dönüştürme işlemi yapılıyor ve veriler liste içindeki listelere dönüştürülüyor..

```
t = []

for i in range (0,7501):

t.append([str(veriler.values[i,j]) for j in range (0,20)])
```

Github üzerinden elde edilen kütüphane çağırılıyor.

```
from apyori import apriori
```

Obje oluşturuluyor ve parametreler belirleniyor.

```
kurallar = apriori(t,min_support=0.01, min_confidence=0.2, min_lift = 3, min_length=2)
```

Kurallar elde edildikten sonra görüntüleniyor.

```
print(list(kurallar))
```

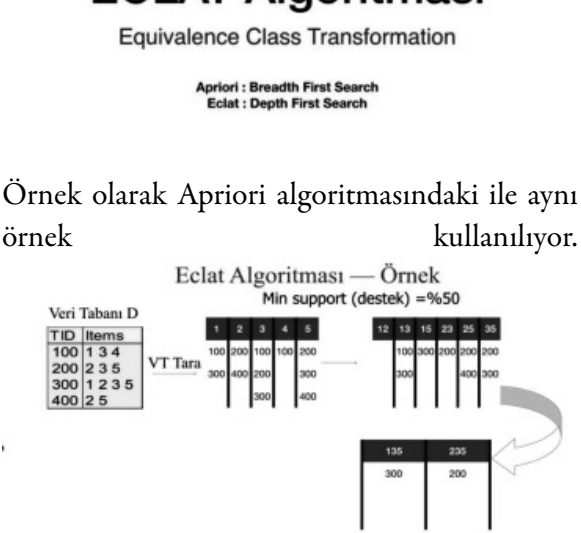
Sonuç olarak bazı ürünler arasındaki durumlar belirlediğimiz değerlere göre ortaya çıkmaktadır.

```
[RelationRecord(items=frozenset({'herb & pepper', 'ground beef'}), support=0.015997866951073192,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'herb & pepper'}),
items_add=frozenset({'ground beef'}), confidence=0.3234501347708895, lift=3.2919938411349285)]),
RelationRecord(items=frozenset({'herb & pepper', 'nan', 'ground beef'}),
support=0.015997866951073192, ordered_statistics=[OrderedStatistic(items_base=frozenset({'herb &
pepper', 'nan'}), items_add=frozenset({'ground beef'}), confidence=0.3234501347708895,
lift=3.2919938411349285)])]
```

Bölüm 25:

Ders 89: Eclat Algoritmasının Çalışması

Bu dersimizde Birliktelik Kural Çıkarımı için kullanılan diğer bir algoritma olan Eclat algoritması incelenecektir.



Örnek olarak Apriori algoritmasındaki ile aynı örnek kullanılıyor.

Görülebildiği üzere Eclat algoritmasının işlemi daha kısa sürmesinden dolayı en hızlı olan algoritma olarak adlandırmak yanlış olmaz. Eclat, derin öncelikli bir algoritma olduğundan dolayı

gösterilecek bir diğer eylem sisteme görünmeden önce daha önceki eylemler arasındaki birliktelik kurallarına ait çıkarımların yapılması mümkündür. Apriori algoritması için bütün itemleri incelemek gerekirken Eclat'da ise, bir ürün seçilerek bu ürünün geçtiği tüm Transactions(İşlemler) listelenerek bütün ürünler taranana kadar algoritmamız çalışmaya devam etmektedir. Sonrasında ise paketlemeler yapılarak en optimal ürün paketleri ortaya çıkartılıyor.

Bölüm 26:

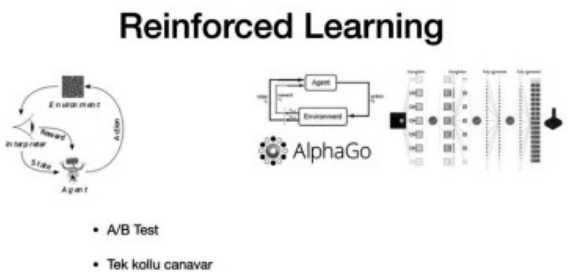
Ders 90: Reinforced Learning Kavramına Giriş

Bu dersimizde, Türkçe literatürde Takviyeli veya Pekiştirmeli Öğrenme olarak kullanılan kavram-

dan bahsedilecektir. Reinforced Learning, yüksek önem arz etmekle birlikte günümüzde fazlaca irdelenen makine öğrenmesi konularından birisidir.

Bu kavram, genelde tanımlanmış bir hedefe yönelik olarak makinanın kendini iyileştirmesidir. Örnek olarak, ayakta bile duramakta zorluk çeken bir robotun deneme yanılma sonucunda elde ettiği tecrübeler ile kendini bulunduğu duruma optimize etmesi gösterilebilir.

Fakat bu yöntem makinaların yeni bir şey öğrenmesi ile ilgili değildir. Bu yöntemin temel amacı, daha önce öğretilen durumun öğretilenden daha iyi yapılmasını sağlamaktır.



Çalışma mantığı oldukça basittir. Öncelikle bir makine(ajan) yazılıyor ve bu makine ortam içerisinde aksiyon alarak bir gözlem çıkarımı yapılıyor. Sonraki adım olarak aksiyon değerlendiriliyor ve duruma bağlı olarak aksiyon ödüllendiriliyor ya da cezalandırılıyor.

Bu çalışma mantığı çerçevesinde makineler, kendi kendine gelişebileceği ve daha iyi bir makineye evrimleşebileceği bir fonksiyona (fitness function) ihtiyaç duymaktadır.

Ders 91: A/B Testi, Tek Kollu Canavar ve Reinforced Learning Kullanım Alanları

Tek kollu canavar makinesi kavramı kumar bazlı işlemlerden ortaya çıkan bir makinedir.

Örnek olarak bir kumarhane ziyaretinde örnekteki makinelerden olduğu ve kişinin amacının ise para kazanmak olduğu düşünülsün. Bu amaç için en iyi kombinasyonun saptanmasına ihtiyaç vardır. Bu makineler önceden belirlenmiş para geri verme oranına sahip olmakla birlikte büyük ikramiyeye ulaşılsa bile total bakımda makinenin her zaman kazançlı çıkacağı bilinmektedir. Makinelerin sahip olduğu kombinasyon dağılımı için öncelikle gözlem yapılmalıdır ve en iyi seçenek seçilmelidir.

A/B Test kavramında ise örnek olarak internet ortamında sunulan reklamların ya da farklı kullanıcı arayüzlerinin başarı oranları gibi objektif olmayan ve bilimsel yaklaşıma sahip olmayan durumlar değerlendirilmektedir.

A/B Testi'nin ana çalışma mantığına örnek vermek gerekirsek, birden fazla reklamın kullanıcılara gösterilerek kullanıcı tepkilerine bağlı olarak bir karara ulaşmak gösterilebilir.

Saha uygulamalarından bahsedecek olursak Amazon şirket gelirinin %35'i ile Netflix şirket gelirinin %75'i bu ders çerçevesinde bahsedilen tavsiye algoritmalarından sağlanmaktadır.

Özet olarak Takviyeli Öğrenme kavramı çok farklı alanlarda uygulanabilir ve bilinen bütün aksiyon içeren veri tabanları bu öğrenme biçimi için uygundur.

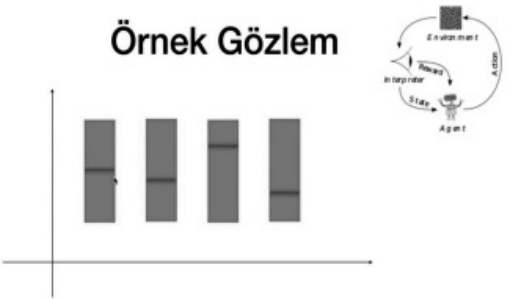
Bölüm 27:

Ders 92: UCB (Upper Confidence Bound, Üst Güven Sınırı)

Üst güven sınırı kavramı basitçe, olayların içerisindeki dağılımın gözlemlenerek istenilen optimal duruma ulaşılmaya çalışılma sırasında değerlendirilen bir değerdir.

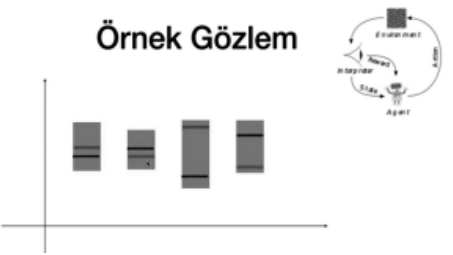
Örnek olarak, 4 farklı reklam incelendiği ve şekildeki gibi bir dağılıma sahip olduğu düşünül-

sün. Kırmızı ile gösterilen çizgiler ise tıklanma potansiyellerini temsil etmektedir.



Örnekteki güvenilirlik(confidence) aralığı tıklanma durumlarına göre güncellenmektedir.


Örnek Gözlem
Environment
Agent
Sensors
Interpreter
State
Action


Örnek Gözlem
Environment
Agent
Sensors
Interpreter
State
Action

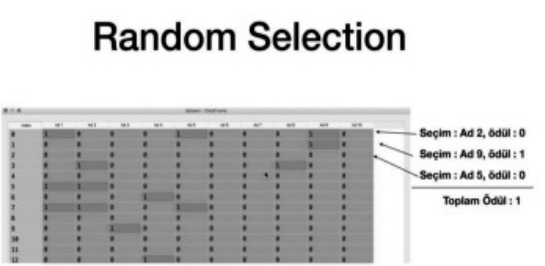
Sonuç olarak makinenin analizlerinden sonra öğrenme gerçekleşmiş oluyor.

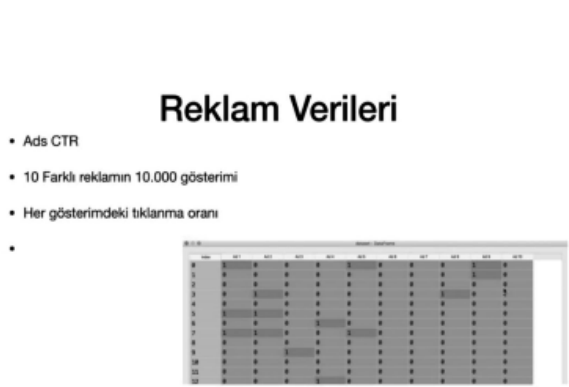
Asıl ulaşılması istenen sonuç ise sonsuz deneme yerine bir veya birkaç güncellemeden sonra optimal sonucun tahmin edilmesidir. Bu tahmin için ise üst güven sınırı kullanılmaktadır. Örneğin aşağıdaki örneğimizde 3 numaralı reklamın en iyi seçenek olduğu görülmektedir.

Ders 93: Python ile Rastgele Yaklaşımın Kodlanması

Bu derste UCB kodlanması için kullanılacak olan yollar incelenecektir. Öncelikle Rastgele Seçim kavramından bahsetmek gerekmektedir.

Saha kullanımına örnek olarak reklam tıklanma oranları incelenebilir. Rastgele Seçim'de ise rastgele reklam tahmini yapılarak ödül mekanizması devreye sokulur.Bu şekilde en iyi ilanı bulunarak yatırımımızın verimli bir şekilde gerçekleşmesi sağlanabilir.

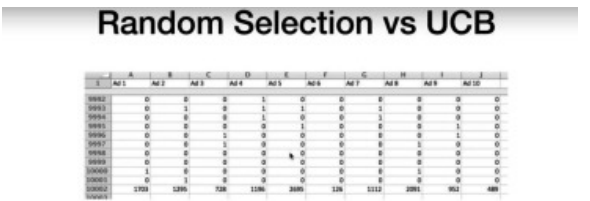




Sitemizden ulaşabileceğiniz veri dosyasında tıklama geçmişine dair 10 bin tıklama verisi olmak-

la birlikte bu veri kümesi zaman serilerine örnek gösterilebilir.

Rastgele seçim sırasında reklamların total tıklanma sayısı önemsizken UCB'de ise bu değerlere göre pekiştirmeli öğrenim yapmak mümkün kılınmıştır.



Rastgele seçim için kodlama kısmı olarak;

Kütüphanelerin çağırılması.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
```

Verilerin okunması.

```
veriler = pd.read_csv('Ads_CTR_Optimisation.csv')
```

Rastgele sayı üretimi için kütüphane çağırılıyor.

```
import random
```

Parametrelerin değerleri veri dosyamıza göre düzenlendikten sonra döngü yazılarak rastgele değerler üretiliyor ve ödül mekanizması duruma dahil ediliyor.

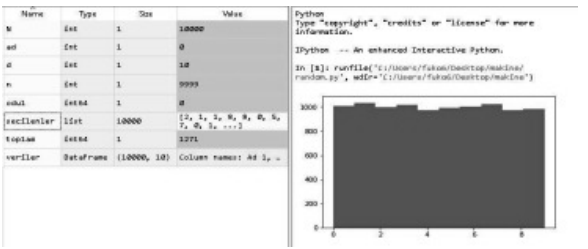
```
N = 10000

d = 10

toplam = 0

secilenler = []
```

```
for n in range(0,N):


ad = random.randrange(d)


secilenler.append(ad)


odul = veriler.values[n,ad] # verilerdeki n. satır =
1 ise odul 1 toplam = toplam + odul
```

Histogram oluşturularak ile görüntüleme yapılıyor.

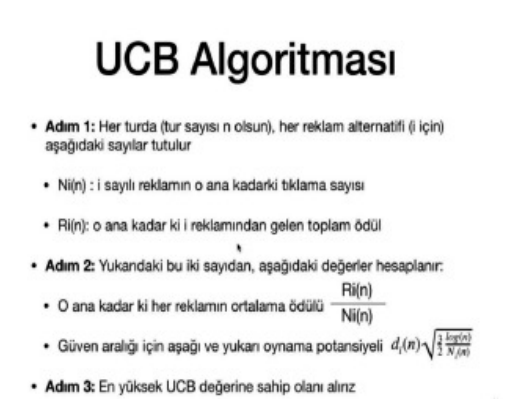
```
plt.hist(secilenler)
```

```
plt.show()
```



Ders 94: Python ile UCB Kodlama

Hatırlanacağı üzere bu algoritmanın özelliği, rastgele seçim yapmayarak önceden kazanılan tecrübelerin sonraki yapılacak seçimlere yansıtmasıydı.



Görseldeki bilgiler doğrultusunda rastgele seçim yapmanın ötesinde daha zeki ve öğrenme işlemi yapabilen bir algoritma yazılması mümkün hale gelmektedir.

Kodlama bölümü olarak ise;

Kullanılan ana kütüphanelere ek olarak matematiksel işlem yapılmasını mümkün kılan kütüphane ekleniyor ve veriler okunuyor.

```
import math

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

veriler = pd.read_csv('Ads_CTR_Optimisation.csv')
```

Parametreler için döngü öncesi değerler tanımlanıyor.

```
N = 10000 # 10.000 tıklama


d = 10 # toplam 10 ilan var


#Ri(n)


oduller = [0] * d #ilk basta butun ilanların odulu
0


#Ni(n)


tiklamalar = [0] * d #o ana kadarki tıklamalar
```

```
toplam = 0 # toplam odul


secilenler = []
```

Görseldeki veriler doğrultusunda döngüler yazılmakta ve ödül mekanizması devreye sokulmaktadır.

```
for n in range(1,N):


ad = 0 #seçilen ilan


max_ucb = 0


for i in range(0,d):
```

```
if(tiklamalar[i] > 0):

ortalama = oduller[i] / tiklamalar[i]

delta = math.sqrt(3/2* math.log(n)/tiklamalar[i])

ucb = ortalama + delta

else:
```

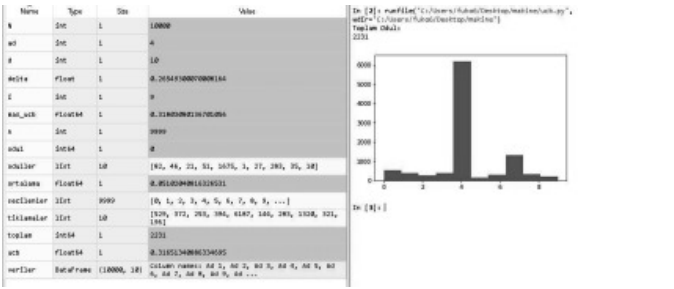
```
ucb = N*10 #Problemden kurtulabilmek adına ucb değeri yüksek bir değer olarak tanımlanıyor
if max_ucb < ucb: #max'tan büyük bir ucb çıktı


max_ucb = ucb


ad = i
```

```
secilenler.append(ad)


tiklamalar[ad] = tiklamalar[ad]+ 1


odul = veriler.values[n,ad] # verilerdeki n. satır
= 1 ise odul 1 oduller[ad] = oduller[ad]+ odul
toplam = toplam + odul
```

Son olarak toplam ödül ve histogramla desteklenen görüntüleme yapılmaktadır.

```
print('Toplam Odul:')


print(toplam)
```

```
plt.hist(secilenler)


plt.show()
```



Sonuç olarak 2231 olarak bulunan 5.reklam pekiştirmeli öğrenimini tamamlamış ve rastgele seçime göre daha yüksek bir başarı yüzdesi elde etmiştir.

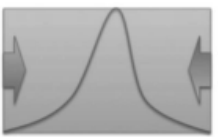
Bölüm 28:

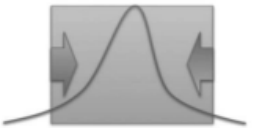
Ders 95: Thompson Yaklaşımı ve Çalışması

Thompson örnekleme algoritması üç adımdan oluşmakta ve önceki dersimizde yazmış olduğumuz UCB algoritmasına benzer özellikleri taşımakta olan bir algoritmadır. Bu algoritmaya örnek olarak bir üretim bandında gerçekleşen hata payı değerlerinin total üretim içinden seçilen numuneler ile bulunması gösterilebilir.

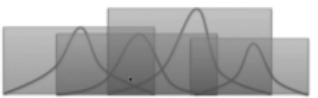
Detaylı kaynak olarak: https://web.stanford.edu/~bvr/pubs/TS_Tutorial.pdf

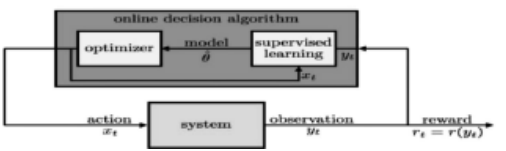
Tek kollu canavar kısmında anlatıdığı üzere her makinenin arkasında bir dağılım vardı. Bu dersteki konumuzun amacı bu dağılımların saptanmasına yöneliktir. Bu dağılımın UCB ile bulunması sırasında doğrusal bir yaklaşım gösterilmekteydi.





Bu algortimamızın diğer durumlardan farklı olarak yaptığı işlem, arkaplandaki dağılıma yakın bir uygulama yapaya çalışmaktır. Bu dağılımın ismi ise formülde de görülebilen Beta dağılımıdır.



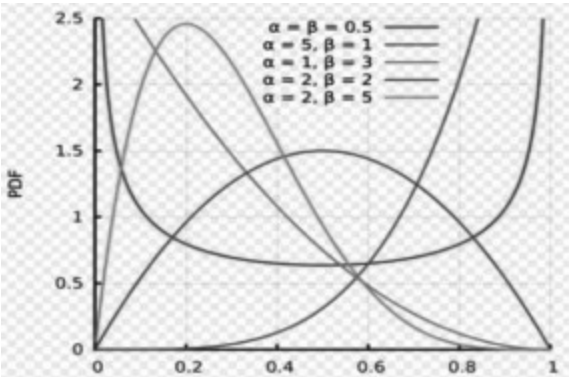


Konumuzun ana fikri ise, pekiştirmeli öğrenim için uygun bir dağılım elde etmektir.

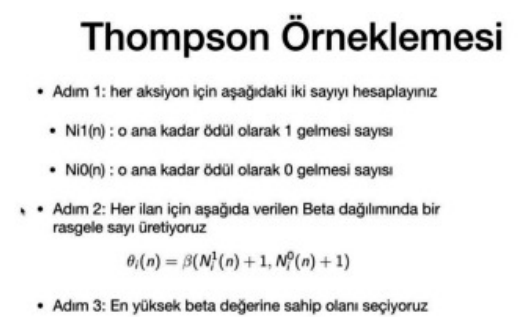
Bilgisayarlar rasyonel çalışan makineler olduğundan dolayı rastgele bir sayı elde etmek bazı durumlar ile mümkündür ve bu durumlar için kullanılan dağılımlar vardır. Bu dağılımlar içerisinde Beta dağılımı da kullanılmaktadır

(https://docs.python.org/2.6/library/random.html [21])

21. https://docs.python.org/2.6/library/random.html


α = β = 0.5
α = 5, β = 1
α = 1, β = 3
α = 2, β = 2
α = 2, β = 5
PDF
2.5
2
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1

Thompson Örnekleme algoritmasının çalışma prensibi ise aşağıdaki görselde görülebilmektedir.



Ders 96: Python Kodlaması : Thompson Sampling

Thompson Sampling, basit bir algoritma olmasıyla birlikte UCB kodlamasına da benzemektedir. UCB kodu üzerinde küçük düzenlemeler ile konumuza ait kodun yazımı mümkündür. Kodumuza ek olarak birler ve sıfırlar gibi değerler eklenerek reklamların tıklanma değerlerine göre güncellemeler yapılmak adına oluşturulacak döngüler ile kod yazımı için gerekli adımlar takip ediliyor.

Kod kısmı ise;

```
import random


import numpy as np


import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
veriler = pd.read_csv('Ads_CTR_Optimisation.csv')
```

```
N = 10000 # 10.000 tıklama

d = 10 # toplam 10 ilan var

#Ri(n)

#Ni(n)

toplam = 0 # toplam odul

secilenler = []

birler = [0] * d
```

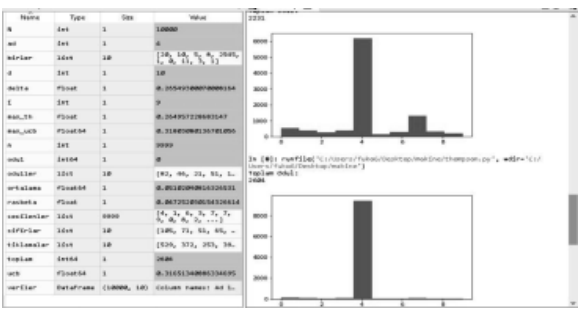
```
sifirlar = [0] * d
```

```
for n in range(1,N):

ad = 0 #seçilen ilan

max_th = 0

for i in range(0,d):

rasbeta = random.betavariate ( birler[i] + 1 , sifirlar[i] +1)

if rasbeta > max_th:
```

```
max_th = rasbeta


ad = i


secilenler.append(ad)


odul = veriler.values[n,ad] # verilerdeki n. satır =
1 ise odul 1 if odul == 1:


birler[ad] = birler[ad]+1


else :


sifirlar[ad] = sifirlar[ad] + 1
```

```
toplam = toplam + odul

print('Toplam Odul:')

print(toplam)
```

```
plt.hist(secilenler)


plt.show()
```



Verilen örneğe bağlı olarak, Thompson algoritması 5.reklam için gösterdiği 2604 değeri ile 2231 değerine sahip UCB algoritmasından daha iyi bir sonuç çıkarttığı görülebilmektedir.

Bölüm 29:

Ders 97: Kavramlara ve DDİ Dünyasına Giriş

Bu dersimizle birlikte Doğal Dil İşleme konusuna giriş yapılmış olunacaktır. Bu konudaki amacımız, herhangi bir doğal üzerinden yazılmış metin üzerinden nasıl işlem yapılacağının anlatılmasıdır.

Örnek olarak bir dilin diğer bir dile anlık çevirisi sürecinde NLU(Doğal dil anlama) ve NLG

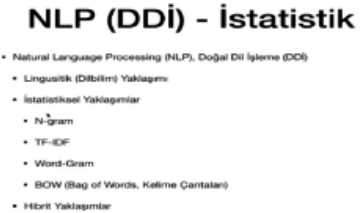
(Doğal dil üretme) kullanılmaktadır ve bu durum konumuza örnek teşkil etmektedir.

Linguistik yaklaşımlar dilbilimi kuralları uyguladığından dolay daha yavaş olan yaklaşım olmasına rağmen kelimlerin istatistiki dağılımını inceleyen yaklaşımdan daha iyi sonuç vermektedir. Bu yaklaşımların da alt alanları vardır.

Şekilbilim ve sözdizim'le birlikte kelimenin ek veya kök ihtimalleri inceleniyor. Bu inceleme sonunda Anlambilim'de cümlenin anlamı kelimeler düzeyinden başlanarak çıkarılıyor ve kullanımbilim'i için daha anlaşılır bir hale getiriliyor. Kullanımbilim'e örnek olarak "saatiniz var mı" sorusuna "yok" cevabı verilmemesi gösterilebilir.

Doğal Dil İşleme'nin günümüzde stabil ve eksiksiz bir değerlendirme sürecine sahip olduğu söylenememektedir. Bunun nedeni ise doğal dillerin yapılarına bağlıdır. Özellikle Türkçe'nin çok fazla ek alabilen yapısından dolayı daha ilk adımda kompleks problemler ortaya çıkmaktadır.

Makinelere daha uygun olan yaklaşımlar ise daha çok istatistikidir.



Bu yaklaşımlar için http://bilgisayarkavramlari.sadievrenseker.com/category/dogal-dil-isleme-nlp/ [22]adresi ziyaret edilebilir.

N-gram yaklaşımı için karakter sayımı kullanılırken BOW yaklaşımında belirli kelimeleri kullanarak kelime çantaları ile metni sınıflandırmak gerekmektedir.

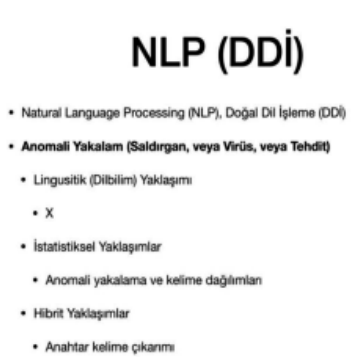
22. http://bilgisayarkavramlari.sadievrenseker.com/category/dogal-dil-isleme-nlp/

Hibrit yaklaşımda ise, metindeki kelimelere göre cümleler etiketlendikten sonra istatistiki bir şekilde sayılması gerekmektedir.

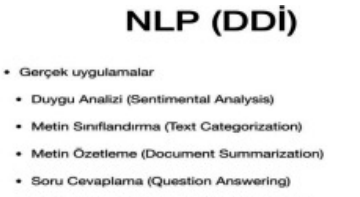
Örneğimiz özelinde yazarın yazdığı metni inceleyerek yazara özel çıkarımlar/noktalama hatası, kullanılan cümle türleri vs.) yapmak mümkündür.

Bir başka örnek olarak metin üzerinden fikir çıkarımı için de bu yöntem kullanılmaktadır.

Anormal davranışların tespit edilmesi konusunda da bu yöntem ile çalışmalar bulunmaktadır.



Diğer uygulama alanları ise aşağıdaki gibidir.



Ders 98: Doğal Dil İşleme Kaynakları

Bazı Doğal Dil İşleme Kaynakları:

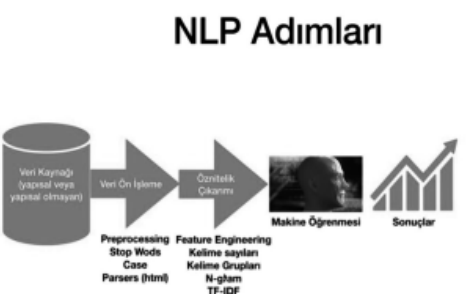
- NLTK : nltk.org
- SpaCy : spacy.io
- Stanford NLP
- OpenNLP: Apache : opennlp.apache.org
- Rapid Automatic Keyword Extraction (RAKE)
- Amueller Word Cloud
- Tensor Flow : Word2Vec

Türkçe Doğal Dil İşleme Kaynaklarından bazıları

- Zemberek (http://zembereknlp.blogspot.com)
- İTÜ : http://tools.nlp.itu.edu.tr
- Tspell (http://tspell.sourceforge.net/hakkinda.html)
- Yıldız Teknik Üniversitesi: Kemik (http://www.kemik.yildiz.edu.tr
- Wordnet (Balkanet)
- TrMorph (http://coltekin.net/cagri/trmorph/)
- TSCorpus (https://tscorpus.com)
- Metu- Sabancı Tree Bank ve ITU Doğrulama Kümesi (https://web.itu.edu.tr/gulsenc/treebanks.html)

Ders 99: Kodlama Öncesi Genel DDİ Problemleri ve Yaklaşımları

Bu derste NLP kavramına girilecek ve ilk uygulama yazılacaktır. Bunun öncesinde ise bazı adımlardan bahsetmekte fayda vardır.



Öznitelik çıkarımı, makinenin öğrenim sırasında kullanacağı sayısal verileri bazı yöntemler ile gerçek dünyadan çıkartmasının sonucudur. Bu

nedenler makine öğrenimi öncesi öznitelik çıkarımları ile verilerimizi hazırlamamız gerekmektedir fakat bunun öncesinde uygunsuz verilerin saptanması ve buna uygun işlemler yapılması gerekmektedir. Örnek olarak cümle ortasındaki kelimerin harflerinin büyük ya da küçük olması gibi etkenler gösterilebilir.

NLP'de önemli olan kısımlar çoğunlukla ön işleme ve öznitelik çıkarımı olmakla birlikte çoğunlukla yapısal olmayan veri kaynakları kullanılmaktadır. Bu kısımlar ile verilerimiz DDİ dünyasına optimize hale getirilmektedir.

Ders 100: Veri Kümesi ve Bazı Kaynaklar

Bu dersimizde kullanacağımız ve bilkav.com sitemizdeen elde edilebilecek veri kaynakları tanıtılacaktır.

Veri setimiz iyi bilinen bir veri kümesi olan Yelp(Restron Yorumları)'dır.

Bu veri setiyle ilgili daha önce yapılmış araştırmalar için literatür taraması yapılmalıdır.

(https://scholar.google.com.tr/scholar?hl=tr&as_sdt=0%2C5&q=yelp+reviews+&btnG= [23])

Identifying Restaurant Features via Sentiment Analysis on Yelp Reviews

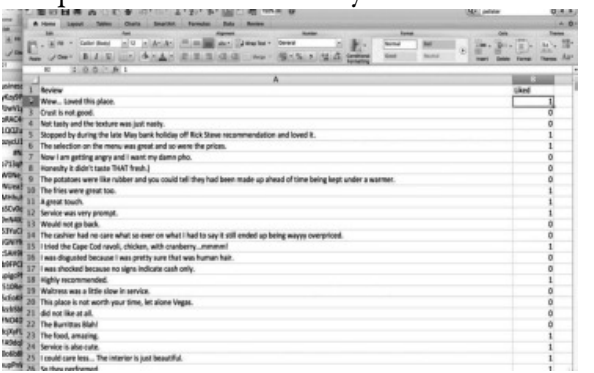
Boya Yu, Jiaxu Zhou, Yi Zhang, Yunong Cao
Center for Urban Science & Progress
New York University
New York, NY, the United States
by671@nyu.edu

***Abstract*—Many people use Yelp to find a good restaurant. Nonetheless, with only an overall rating for each restaurant, Yelp offers not enough information for independently judging its various aspects such as environment, service or flavor. In this paper, we introduced a machine learning based method to characterize such aspects for particular types of restaurants. The main approach used in this paper is to use a support vector machine (SVM) model to decipher the sentiment tendency of each review from word frequency. Word scores generated from the SVM models are further processed into a polarity index indicating the significance of each word for special types of restaurant. Customers overall tend to express more sentiment regarding service. As for the distinction between different cuisines, results that match the common sense are obtained: Japanese cuisines are usually fresh, some French cuisines are overpriced while Italian Restaurants are often famous for their pizzas.**

***Keywords*— Yelp, Sentiment Analysis, Support Vector Machine**

23. https://scholar.google.com.tr/scholar?hl=tr&as_sdt=0%2C5&q=yelp+reviews+&btnG=

Veri setimizde iş yeri ile ilgili farklı parametreler, iş yeri yorumları, verilen yıldız puanları gibi değerler bulunmaktadır. Makinamızı eğitimi yorumlar üzerinden gerçekleştirmek istenmekte ve bunun için yorumların olumlu-olumsuz olarak çıkarımlanması için bir makine algoritmasına ihtiyaç duyulmaktadır. Bu durum için duygusal kutupsallıktan yararlanılmaktadır.



	A	B
1	Review	Liked
2	Wow... Loved this place.	1
3	Crust is not good.	0
4	Not tasty and the texture was just nasty.	0
5	Stopped by during the late May bank holiday off Rick Steve recommendation and loved it.	1
6	The selection on the menu was great and so were the prices.	1
7	Now I am getting angry and I want my damn pho.	0
8	Honestly it didn't taste THAT fresh.)	0
9	The potatoes were like rubber and you could tell they had been made up ahead of time being kept under a warmer.	0
10	The fries were great too.	1
11	A great touch.	1
12	Service was very prompt.	1
13	Would not go back.	0
14	The cashier had no care what so ever on what I had to say it still ended up being wayyy overpriced.	0
15	I tried the Cape Cod ravoli, chicken, with cranberry...mmmm!	1
16	I was disgusted because I was pretty sure that was human hair.	0
17	I was shocked because no signs indicate cash only.	0
18	Highly recommended.	1
19	Waitress was a little slow in service.	0
20	This place is not worth your time, let alone Vegas.	0
21	did not like at all.	0
22	The Burritos Blah!	0
23	The food, amazing.	1
24	Service is also cute.	1
25	I could care less... The interior is just beautiful.	1
26	So there performed	1

Daha fazla dökümantasyon için bazı adresler aşağıda sıralanmıştır.

https://www.nltk.org[24]

https://www.kaggle.com[25]

Ders 101: Python İlk Problemler: Space Matrix ve İmla İşaretleri

Space matrix kavramı kullanım amacı, cümlelerdeki kelimelerin sayımı için kullanılmakta ve gelecek yorumlarda kullanılan kelimler ile eşleşme yakalanmaya çalışmaktadır. Bu şekilde kelime vektörü gittikçe büyüyecektir. Bu neden dolayı matrisin büyük bir kısmı boşluktan oluşa-

24. https://www.nltk.org/

25. https://www.kaggle.com/

cağından dolayı “uzay matrisi” ismini almaktadır.

Veri setimizde çözmek istediğimiz ilk sorun noktalama işaretleri ile ilgili olan olacaktır. Çözüm için düzenli ifadeleri içeren “regularexpression (re) ” kütüphanesinden yararlanılmaktadır.

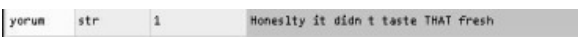
Kodlama işlemimize sürekli kullanılan şablon üzerinden başlanılmakta ve kütüphaneler çağırılarak veriler okunmaktadır.

```
import numpy as np

import pandas as pd
```

```
import re

import nltk


yorumlar = pd.read_csv('Restaurant_Reviews.csv')
```

Noktalama İşaretlerinin filtrelenmesi için alfabedeki karakterler gösterilerek bu karakter harici itemler ".sub" ile boşluğa dönüştürülüyor. Örnek olarak 6.satırdaki yorum kullanılabilir.

```
yorum = re.sub('[^a-zA-Z]',' ',yorumlar['Review'][6])
```



Ders 102: Python, 2.Problem: Büyük/Küçük Harfler

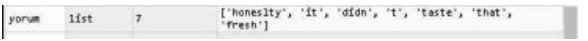
Bu kodlama işleminde bütün harflerimizi büyük/küçük standartına dönüştürmeyi hedeflemekteyiz.

Bu işlem için iki yöntem bulunmaktadır. Bu yöntemler, bütün harflerin küçük harflere dönüştürme ya da tam tersi şekilde büyük harflere dönüştürmektir. Duygusal kutupsallık örnekleri üzerinde çalışma yapıldığından dolayı bu yöntemler kullanılmakla birlikte farklı veri setilerinde büyük küçük harf farklılıkları önem taşıyabilmektedir.

Kod kısmı için iaşağıdaki kodlar kullanılmakta ve kelimeler küçük harflere dönüştürülerek birbirinden ".split" ile ayrılmaktadır.

```
yorum = yorum.lower()
```

```
yorum = yorum.split()
```



Ders 103: Python 3.Problem: Stop Words

Bu derste ise, metnimiz içerisindeki duygusal kutupsallık işlemlerimiz için anlam taşımayan kelimeleri filtreleme ve temizleme amacımız bulunmaktadır.

Bu filtreleme için ise "nltk" kütüphanesi kullanılmaktadır. Türkçe stop words için github kaynakları kullanılabilir.

Kütüphane kullanımı için stop words listesi indirilmeli ve çağırılmalıdır.

from nltk.corpus import stopwords

```
nltk.download('stopwords')
```

Kelime köklerinin bulunarak çekim eklerinden temizlenmesi için ise aşağıdaki kütüphane kullanılacaktır.

```
from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()
```

Ders 104: Python 4. Kelime Gövdeleri (Stemmer)

Bu dersimizde ise, kelime köklerini elde etmeye çalışılacak ve gereksiz çekim eklerinden temizlenme işlemi için yazılması gereken python kodumuz gösterilecektir.

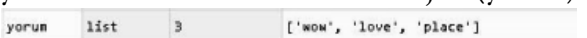
Kod kısmı olarak öncelikle cümlemizdeki bütün kelimeler bir "kelime" değişkeni ile gözlemleniyor ve bu kelimenin ekleri atılarak ingilizce stop words içinde olma durumu kontrol ediliyor. Bu kontrolden sonra çıkan liste köşeli parentezler ile ayrı bir listeye dönüştürülüyor. Ayrıca 'english' yerine ' turkish' yazarak Türkçe stop words listesine ulaşılabilinmektedir.

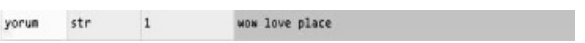
```
yorum = [ps.stem(kelime) for kelime in yorum
if not kelime in set(stopwords.words('english'))]
```

İlk cümle için sonucumuz aşağıdaki gibidir.

Listedeki kelimeleri otomatik bir şekilde sayabilmek için "CountVectorizer" yapısı kullanılması gerekmektedir fakat bu yapı için listemizin tekrar string formatına dönüşümü yapılmalıdır. Bu nedenle, kelimeler arasına boşluk konularak "."join" komutu ile tekrar string formatına dönüşümü sağlanıyor.

```
yorum = ' '.join(yorum)
```





Aşağıdaki kodumuz ile son derslerimizdeki kodlama kısımları birleştiriliyor ve bir döngü içerisinde bütün cümleler taranıyor. Son hali verilmesi için ise “derlem” listesi yardımıyla filtrelenmiş veriler toplanıyor.

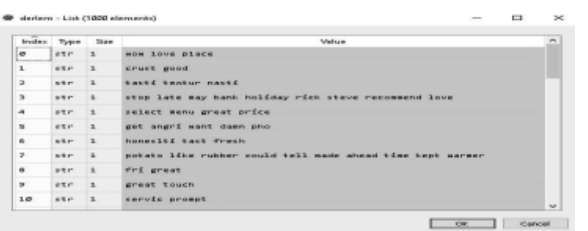
```
derlem = []

for i in range(1000):

yorum = re.sub('[^a-zA-Z]', ',yorumlar['Review'][i])

yorum = yorum.lower()
```

```
yorum = yorum.split()
```

```
yorum = [ps.stem(kelime) for kelime in yorum
if not kelime in set(stopwords.words('english'))]
```

```
yorum = ' '.join(yorum)
```

```
derlem.append(yorum)
```



Bu aşamaya kadarki kodlama kısımları ile verilerin uygun hale getirilmesini amaçlayan Preprocessing kısmı gösterilmiştir.

Ders 105: Python 5. Kelime Vektörü Sayaç Kullanımı (CountVectorizer)

Veri ön işleme kısmının geçilmesi üzerine bu dersimizde öz nitelik çıkarımı konusuna giriş yapılacaktır.

Bu çıkarımlar kelime sayıları üzerinden tespit edilecektir.

Bu çıkarım için ilk olarak gerekli olan Count Vector çağırılıyor ve objeye atanıyor.

#Feautre Extraction (Öznitelik Çıkarımı)

#Bag of Words (BOW)

from sklearn.feature_extraction.text import CountVectorizer

max_features parametresi ile en fazla kullanılan kelimeler için sınır koyulmaktadır.

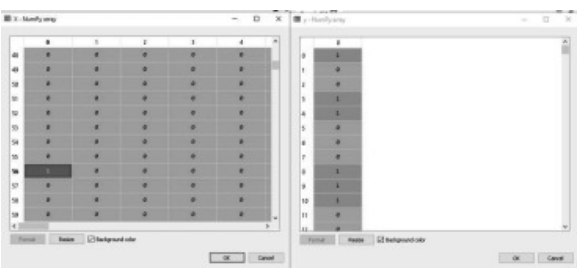
cv = CountVectorizer(max_features = 2000)

Daha önce ön işleme aşamasını geçen verilerimiz "cv" ile eğitilip dönüştürüldükten sonra dizi formatına çeviriliyor.

X = cv.fit_transform(derlem).toarray() # bağımsız değişken

Sonraki adım olarak, dugusal kutupsallıktan yararlanmak için ilk veri setimizin ikinci kolonu bağımlı değişken olarak alınmaktadır.

```
y = yorumlar.iloc[:,1].values # bağımlı değişken
```



Ders 106: Python 6. Makine Öğrenmesi ve Sınıflandırma Kısmı

Bu dersimizdeki önceki derslerimizden de tanıdık olduğumuz Sınıflandırma algoritmaları ile makine öğrenimi yapılacaktır.

Kodlamaya ilk olarak verilerin bölünmesi ile başlanmaktadır.

```
from sklearn.cross_validation import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

Bu veri setimiz için rastgele olarak Gaussian Naive Bayes sınıflandırma algoritması kullanılıyor ve veri eğitim işlemi yapılıyor.

```
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X_train,y_train)
```

Son kodlama kısmı olarak ise, tahminler yapıldıktan sonra sonuçlarımız karmaşıklık matrisi yardımıyla görüntülenmektedir.

```
y_pred = gnb.predict(X_test)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,y_pred)

print(cm) # %72.5 kesinlik sonucu elde edilmiştir

from sklearn.metrics import confusion_matrix
```

cm = confusion_matrix(y_test,y_pred)

print(cm) # %72.5 kesinlik sonucu elde edilmiştir

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\fuko6\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[[54 43]
 [12 91]]
```

Bölüm 30:

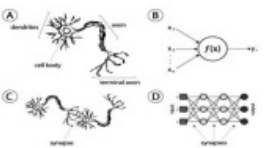
Ders 107: Yapay Sinir Ağlarına ve Nöron Yapısına Giriş

Bu kavram son yıllarda tekrar popüler hale gelmiş ve derin öğrenme konusunun temelini oluşturmaktadır. Sinirbilimi kavramı nörobilimden gelen ve insanların karar verme süreçlerini

için yönetim görevinde bulunan sistem için kullanılmaktadır. Bu kavramın incelenmesindeki temel amaç, şayet insan beynindeki gibi bir nöron yapısı modellenebilirse bu model ile makineye insan düşüncesinin aktarımını incelemektir.

Yapay Sinir Ağları (YSA)
Artificial Neural Network (ANN)

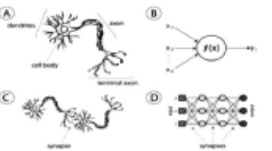
- Sinirbilim ve Bilgisayar Bilim (A)
- Nöron (Neuron) (B)
- Sinapsis (Synapsis) (C)
- YSA Çalışma Mantığı (örnek üzerinden) (D)
-



Yapay Sinir Ağları'nın diğer makine öğrenim metotlarına göre dezavantajları, yavaş hızda çalışması ve yüksek sayıda kaynak talep etmesidir. Son dönemlerde bilgisayarların hızlanması ve GPU verimliliğinden dolayı bu kavram tekrar popüler hale gelmiştir.

Yapay Sinir Ağları (YSA)
Artificial Neural Network (ANN)

- Sinirbilim ve Bilgisayar Bilim (A)
- Nöron (Neuron) (B)
- Sinapsis (Synapsis) (C)
- YSA Çalışma Mantığı (Örnek üzerinden) (D)
- YSA ile öğrenmek ne demektir ve bir örnek üzerinden öğrenme
- Gradient Descent (Gradyan iniş)
- Stochastic Gradient Descent
- Backpropagation

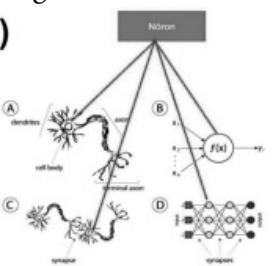


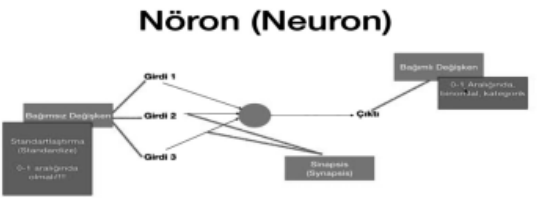
Yapay Sinir Ağları'na ait girdiler ve çıktılar olmakla beraber bu değerler sistemde sinapsisler yardımıyla taşınmaktadır. Nöronlar ise karar verme süreçlerinde görev almaktadır.

Yapay Sinir Ağları bazı hassasiyetler barındırmaktadır. Bu hassasiyeti ise Standartlaştırma yapılması zorunluluğudur. İşlemler sırasında arkaplanda Standartlaştırma yapıldığından dolayı sistemde yavaşlama gözlemlenmektedir.

Nöron (Neuron)

- Sinirbilim ve Bilgisayar Bilim (A)
- **Nöron (Neuron) (B)**
- Sinapsis (Synapsis) (C)
- YSA Çalışma Mantığı (örnek üzerinden) (D)
- YSA ile öğrenmek ne demektir ve bir örnek üzerinden öğrenme
- Gradient Descent (Gradyan iniş)
- Stochastic Gradient Descent
- Backpropagation




Nöron (Neuron)
Bağımsız Değişken
Standartlaştırma
(Standardize)
0-1 aralığında
olmalı!!!
Girdi 1
Girdi 2
Girdi 3
Çıktı
Sinapsis
(Synapsis)
Bağımlı Değişken
0-1 Aralığında
binomial kategorik

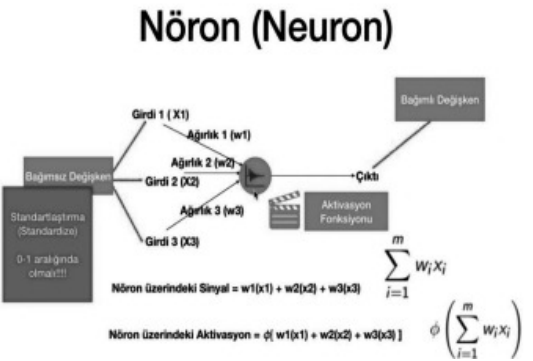
YSA için çoklu çıktı bulunması da mümkündür.

Sinapsisler özelinde ise, girdi taşınması sırasında katsayı kullanılmakta ve katsayıya bağlı olarak girdi sinyalinde zayıflama veya güçlenme görülebilmektedir. Bu durum da YSA'nın karar verme sürecine etki etmektedir.

Bu katsayıya örnek olarak hızlı bir şekilde hareket ettirilen elin havayı durduğu hale göre daha iyi hissetmesi gösterilebilir.

Nöron, üzerindeki sinyal yüküne göre karar verme işlemi gerçekleştirmekte ve aksiyon al-

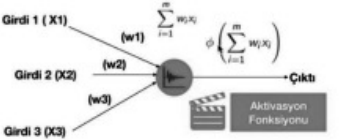
maktadır.



Ders 108: Aktivasyon Fonksiyonları (Activation Functions)

Önceki dersimizde nörona ait girdi, çıktı ve işlem katmanlarından bahsedilmişti. Bu ders bağlamında daha çok aktivasyon kavramı üzerinde durulacaktır.

Aktivasyon (Activation)

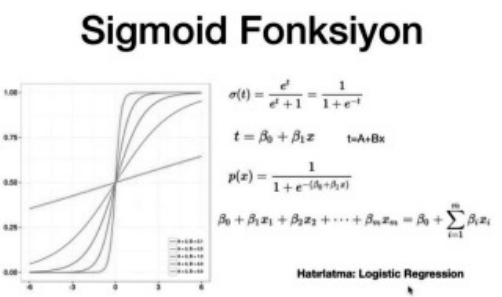


Nöron üzerindeki Sinyal = w1(x1) + w2(x2) + w3(x3)

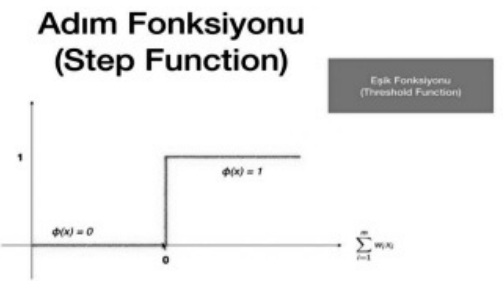
Nöron üzerindeki Aktivasyon = ϕ(w1(x1) + w2(x2) + w3(x3))

Bazı fonksiyon türleri, sağladığı avantajlar nedeniyle daha fazla kullanıma sahip olduğundan dolayı diğerlerine göre öne çıkmaktadır.

Lojistik dersinden de hatırlanabilecek Sigmoid Fonksiyonu, aktivasyon fonksiyonları içerisinde kendisine önemli bir yer sağlamaktadır.

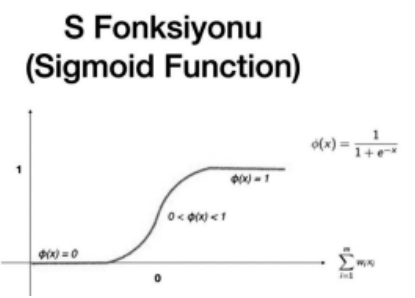


İlk olarak,Sigmoid fonksiyonun daha ilkel modeli olan Adım fonksiyonundan bahsedelim.

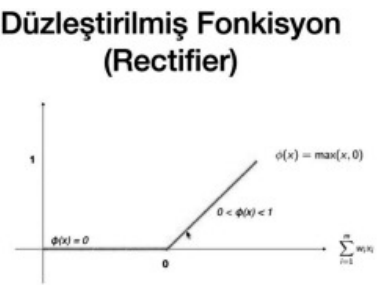


Adım Fonksiyon'u bazında eşik değeri bulununcaya kadar herhangi bir işlem yapılmamakta ve eşik değerine ulaşılması durumunda ise aksiyon alınmaktadır. Bu fonksiyon, nöronun üzerindeki toplam yük ve nöronun aksiyon alması bağıntısında kullanılmaktadır.

Sigmoid Fonksiyonu'nu Adım Fonksiyonu'ndan ayıran en önemli özellik, 0 ve 1 arasındaki değerlerin de süreçte rol oynamasıdır.

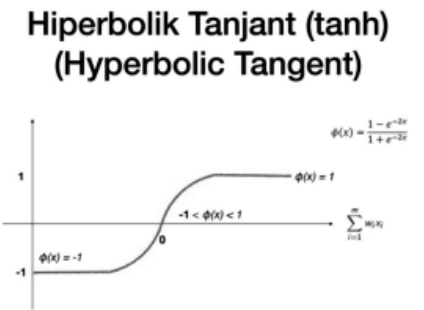


Düzleştirilmiş Fonksiyon da Sigmoid ile benzer aksiyonlar almakla birlikte daha farklı bir yol izlemektedir.



Bu fonksiyonda 0'ın altındaki negatif değerler 0 değerine atanırken pozitif değerler aynı şekilde grafikteki yerini almaktadır.

Hiperbolik Tanjant, Sigmoid Fonskiyonu'na benzemekle birlikte sadece -1 ile 1 arasındaki değerleri kullanmaktadır.



Bahsedilen aktivasyon fonksiyonları günümüzde fazlasıyla kullanılan fonksiyonlar olmakla beraber herhangi bir matematiksel fonksiyonun değiştirilerek aktivasyon fonksiyonu gibi kullanılması da mümkündür.

Aktivasyon fonksiyonunun sonuca etkileme payı büyük olduğundan dolayı verilere uygun fonksiyon seçimi yüksek önem arz etmektedir.

Önümüzdeki derin öğrenme derslerimizde gösterileceği üzere, yüksek nöron sayısına sahip YSA'nda bazı nöronlar farklı aktivasyon fonksiyonları ile yönetilecek ve bu fonksiyonlara ait parametreler tanıtılacaktır.

Ders 109: Katman Kavramı ve Çalışan Bir Yapay Sinir Ağı

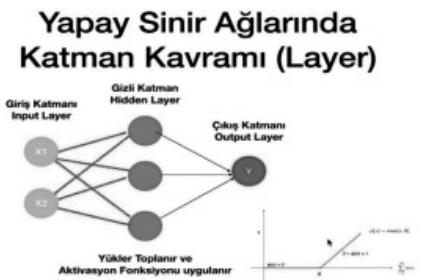
Bu derste YSA'nda katman(layer) olarak adlandırılan kavram incelenecektir.

Gizli katman sayısının arttırılması mümkün olduğu gibi bu katmanlardaki nöron sayısının artışı da mümkündür.

İlk olarak Giriş katmanındaki girdiler gizli katman için ağırlıklandırılıyor.

Sonraki adım olarak, birbirinden bağımsız çalışan ve sadece kendi üzerindeki sinyal yükü ile

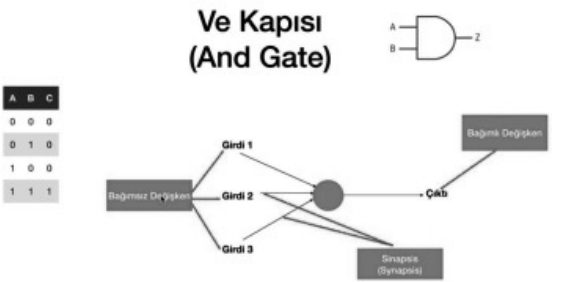
ilgilenen nöronlar, bir aktivasyon fonksiyonuna tabi tutularak aksiyon alma anlamında karar verme mekanizmalarını devreye sokuluyor.



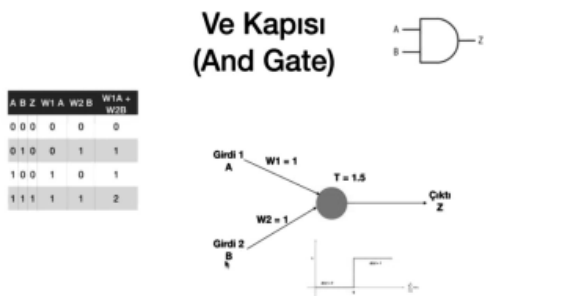
Çıkış Katmanı'nda ise Gizli Katman'dan gelen veriler yük olarak toplanmakta ve başka bir aktivasyon fonksiyonu kullanılarak çıktı elde edilmektedir.

Önceki dersimizde bahsedildiği üzere Gizli Katman'daki her nöron birbirinden farklı olduğundan dolayı bu nöronlar birbirinden farklı aktivasyon fonksiyonları ile de çalışabilmektedir.

Bu sisteme örnek olarak lise matematiğnden hatırlanabilecek mantık devreleri gösterilebilir.



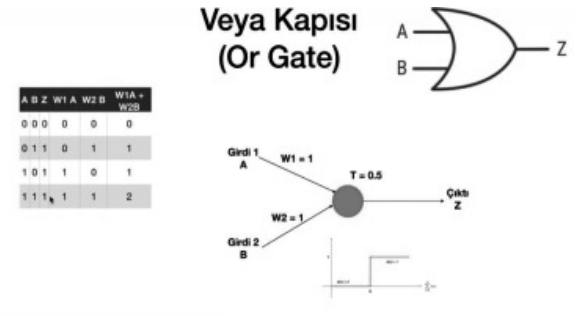
Tablodaki verilere göre katmanlar oluşturuluyor ve Adım Fonksiyon'u için eşik değeri belirleniyor.



Çıktı için girdilerin katsayılar ile kullanımı inceleniyor ve sonuçlar toplanarak eşik değeri ile karşılaştırılıyor.

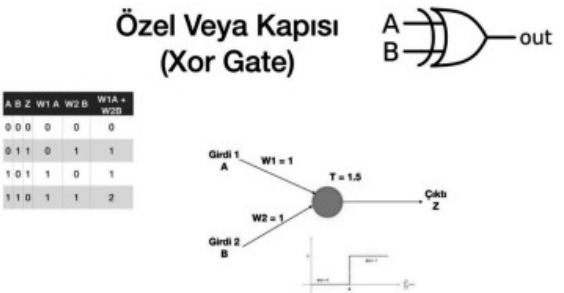
Ders 110: XOR Problemi ve Çözüm Olacak YSA Tasarımı

Bu dersimizde ise önceki derste çözümlenen problemin bir benzeri olan "Veya Kapısı" ve farklı bir durum olan "Özel Veya Kapısı" için YSA kullanımları incelenecektir. Önceki örneğimizin aksine Veya Kapısı'ndaki eşik değerinin daha düşük olmasından dolayı ağımızdaki nöron daha fazla aksiyon almaktadır.

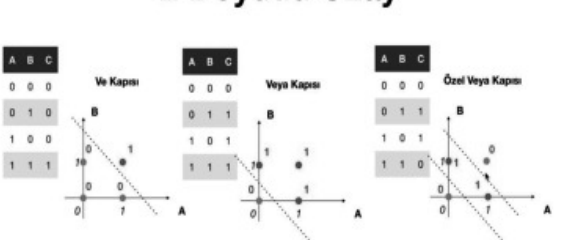


Örneğimizden de görüldüğü üzere tek bir nöron ile "Veya Kapısı" problemi çözülebilmektedir.

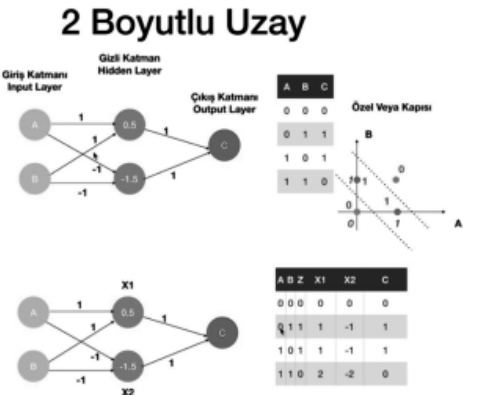
Bir başka önemli kapı örneği olan "Özel Veya Kapısı (XOR Gate) " farklı bir orijinal Veya Kapısı'ndan farklı sonuçlar göstermektedir. Örneğin eşik veya katsayı değerlerinin değişimi ile bu mantık bağlacı çözülemektedir.



Bu durumun nedeni ise 2 boyutlu uzayda 0 ve 1 değerlerinin diğer mantık kapılarına göre birbirinden doğrusallık ile ayrılamama durumundan kaynaklanmaktadır.



Bu problemin çözümü için birden fazla nöron kullanımı gerekmektedir ve bunun nedeni ise tek nöron ile 2 boyutlu uzay sadece bir defa bölünebilmesidir.

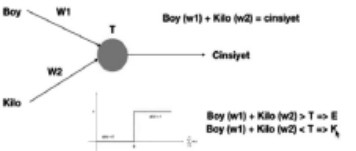


YSA'nın saha çalışmalarında kullanılacağı katsayı ve diğer değerler makine öğrenmesi ile saptanmakta ve önümüzdeki derslerin konularını oluşturmaktadır.

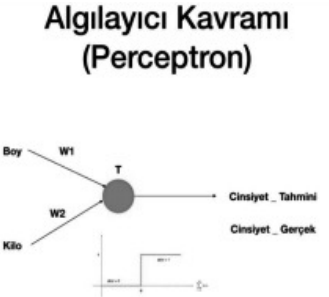
Ders 111: YSA Nasıl Öğrenir : Perceptron (Algılayıcı) Kavramı

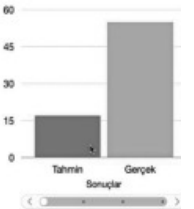
Bu derste YSA'nın öğrenme sürecinden ve Algılayıcı kavramının detaylarından bahsedilecektir.

Örneğimizde eşik değerlerine göre girdilerin sınıflara ayrılarak farklı çıktılar(Erkek,Kadın) elde edilmesi süreci gösterilmektedir. YSA'ı doğrusal olarak ayrılamayan veri setleri için kullanılabildiğinden dolayı önceki derslerimizde işlediğimiz Çekirdek Hilesi yönteminin kullanımına uygundur.

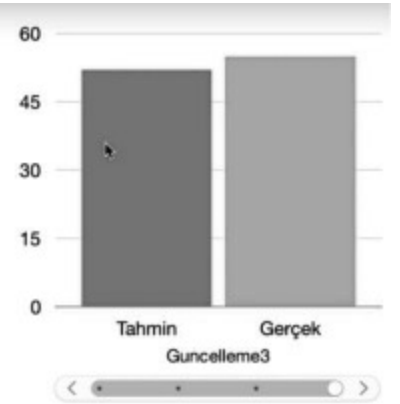


YSA'nın öğtenme sürecinde değişkenlerin (W1, W2, T) sonuca etkisi olduğundan dolayı doğru bir şekilde saptanması önemlidir. Bu süreç için Algılayıcı kavramı kullanılmaktadır. İlk derslerimizden aşina olduğumuz verilerin eğitim ve test olarak ayrılmasından sonraki eğitilme süreci bu kavrama örnek olabilir. Buradaki tahmin değerleri ile gerçek değerler arasındaki fark gözetilerek gerekli geribelemelerle güncellemeler yapılarak YSA'nın öğrenimi gerçekleşiyor.

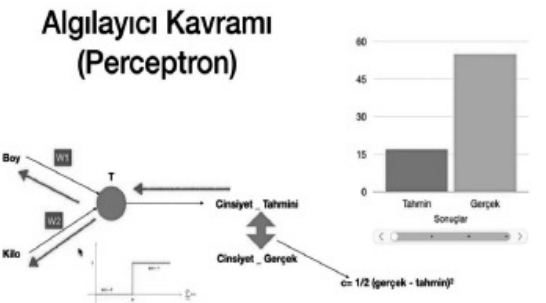




Son güncelleme ile Tahmin değerlerimizin Gerçek değerlere yakınlaştığı görülmektedir.



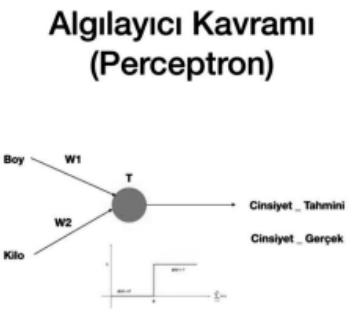
Değikenlerin güncellenmesi için kullanılan “c” hata değerine bağlı olarak değerlerimizde değişiklik gözlenmektedir.



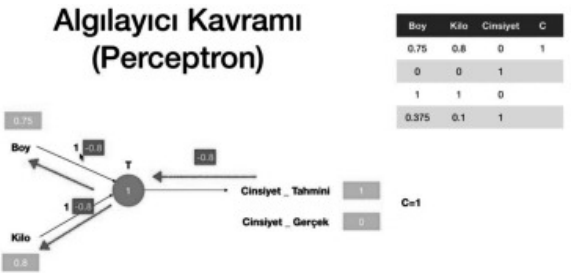
Öğrenme oranı kavramında ise, hata değerinin geri yansıltma durumunda bazı katsayılar devreye girebildiğinden dolayı sistem ani tepkiler verebilmekte ve o zamana kadar YSA'nın öğrendiği çıkarımları değiştirebileceğinden dolayı bu durum sistemde tehlike arz etmektedir.

Örnek üzerinde bu öğrenme sürecini inceleyelim. Öncelikle verilernormalize edilerek cinsiyet

değerlerinin encoding sürecinden geçmesi gerekmektedir.

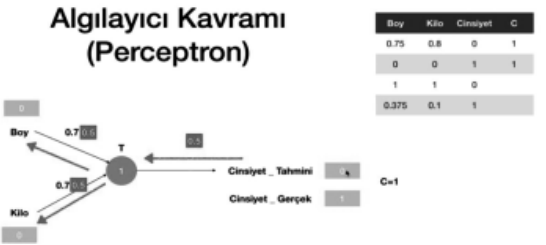


Sonrasında ise takviyeli öğrenme yapılacağından dolayı değişkenlere rastgele değerler verilerek YSA'nın öğrenme süreci başlatılmış olacaktır.



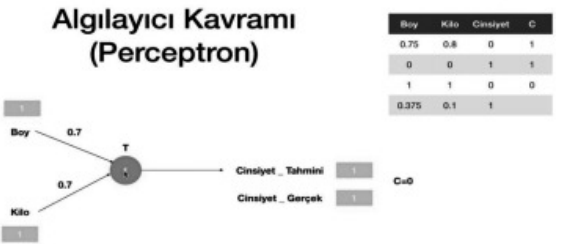
Boy	Kilo	Cinsiyet	C
0.75	0.8	0	1
0	0	1	
1	1	0	
0.375	0.1	1	

İlk öğrenme süreci hatalı olduğundan dolayı hata yansıtma uygulanacaktır. Bunun için öğrenme oranı önceden belirlenmiş katsayı değerinden küçük olarak verilerek süreç içerisindeki öğrenme durumu kontrol edilmektedir. Daha küçük öğrenme oranı sistemde daha küçük oynamalara sebep olacaktır. Gösterilen bu öğrenme oranları maliyet olarak geri dönmekte ve böylece sinapsların kendilerini güncellemeleri sağlanmaktadır.

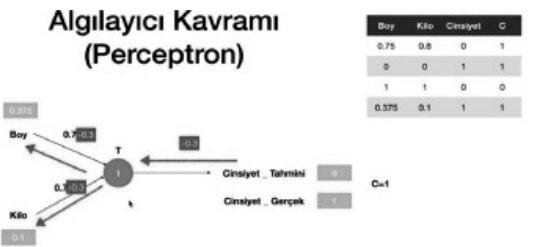


Hesaplanma sonucu yeniden yanlış değer elde ediliyor ve 0.2 olan katsayı değerleri öğrenme oranın 0.5'e düşürülmesinden dolayı 0.7 olarak güncellenerek bir sonraki deneme için hazırlanıyor.

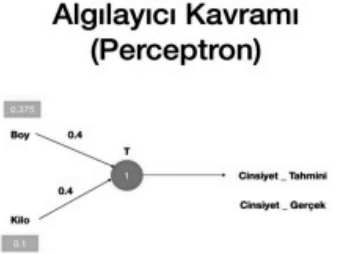
Sonraki örneğimizde ise doğru tahmin yapıldığından dolayı herhangi bir güncelleme gereği bulunmamaktadır.



Öğrenme süreci sonundaki son örnek ile tahmin yapılıyor ve yanlış tahmin sonucunda daha küçük bir öğrenme oranı ile sinapslar güncelleniyor.



Son adım olarak daha fazla örnek kalmadığından dolayı öğrenme süreci bitirilmektedir.



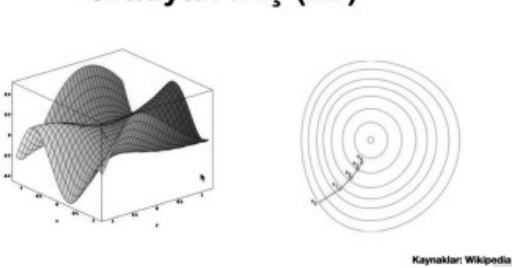
Boy	Kilo	Cinsiyet	C
0.75	0.8	0	1
0	0	1	1
1	1	0	0
0.375	0.1	1	1

Takviyeli öğrenme sürecini bitirmiş YSA'nın tekrar ilk örneği değerlendirilmesi istendiğinde, YSA önceki sonucun tam tersi olan doğru bir çıkarım elde etmektedir.

Ders 112: Gradyan Alçalış (Gredient Descendent)

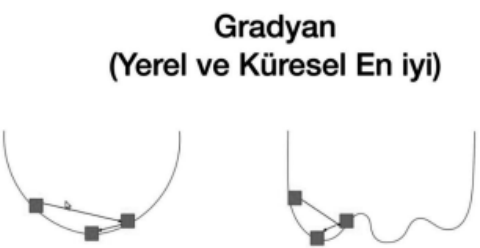
YSA'ı optimum çıkarım elde edene kadar öğrenme işlemini sürdürmektedir. Fakat karmaşık sinir ağalarında bu süreç oldukça yavaş işlemektedir. Bu süreçteki kilit noktalaradan biri ise öğrenme sırasında denemeler arasında nasıl atlamalar yapılacağıdır. Bu nedenle öğrenme oranı yüksek önem arz etmekle birlikte simetrik atlamalara sebep vermemek adına gittikçe küçülen bir hata yansıtma değerine sahip olmaktadır.

Yanlış belirlenen öğrenme oranları YSA'ını yanlış yönlendireceğinden dolayı YSA özelinde ulaşılmak istenilen optimal öğrenme noktasına erişmek için Gradyan Alçalış yöntemi kullanılmaktadır.



Ders 113: Stokastik(Stochastic), Yığın(Batch) ve Mini Yığın Gradyan Alçalışlar

Bu derste Gradyan İniş yöteminin alt türleri incelenecektir. Bu alt türler Stokastik(Stochastic), Yığın(Batch) ve Mini Yığın Gradyan Alçalışlar'dır.



Sağ şekilde görülebileceği üzere gradyan alçalışıyla birlikte yerel optimum noktaya ulaşıl-

ma durumunda sistem, global tanımda olan ve asıl ulaşılmak istenen optimal noktayı bulamayacaktır.

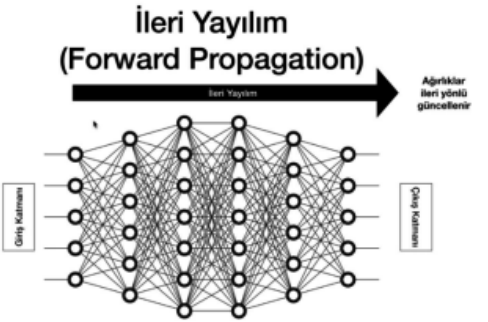
Yaklaşımlar bazında değerlendirme yapılacak olursa her veri değerlendirmenin sonunda çıkarım yapılan durum Stokastik yaklaşıma örnek olmakla birlikte bütün bir sistemin değerlendirilmesinden sonra yapılan çıkarım Yığın yaklaşımına örnektir.

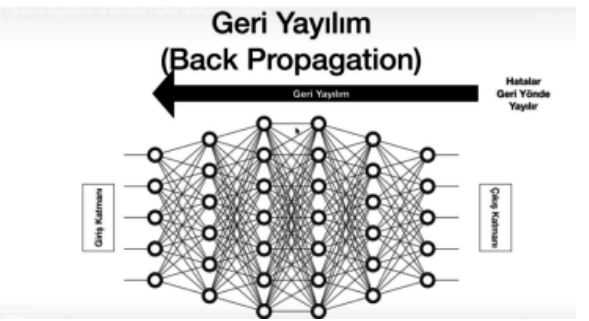
Bu iki yaklaşım mantığının kesişiminde ise küçük sistemlere ayrılmış verilerin bulunduğu mini yığın yaklaşımı bulunmaktadır.

Ders 114: YSA Öğrenme Algoritması ve İleri/Geri Yayılım (Backpropagation/Forwardpropagation)

Derin Öğrenme bölümümüzün en önemli dersi olarak YSA algoritmalarından ve sonrasında ise bu algoritmalar için kullanılan kütüphaneler ile problem tanımlarından bahsedilecektir.

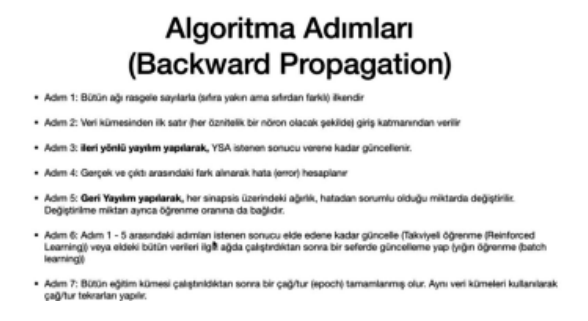
Bir YSA incelenirken İleri ve Geri yayılıma sahip olunmasından bahsedebiliriz.





Önceki derslerimizde görüldüğü üzere giriş katmanı kısmında kaç adet nöron olacağını öz nitelik sayımız belirlemektedir. YSA'ı doğaları gereği öz niteliklerin arasında oluşabilecek kukla değişkeni tuzağına karşı dirençlidir.

YSA algoritması, yayılımlara bağlı olarak 7 adımda saptanmaktadır.



Bu algoritmanın belirlenme sürecinde bazı parametreler veri bilimcisinin tecrübesi doğrultusunda optimum olarak belirnmeye çalışılmaktadır. Bu parametrelere öğrenme oranı ve öğrenme sürecinde kaç tur veri kullanılacağı gibi sayılar örnek olabilir. Tur sayısının büyük olarak sisteme verilmesi öğrenme bittikten sonra sistemin aynı öğrenmeyi gerçekleştirmek için boşa zaman harcayacağından dolayı verimsiz olabilirken tam tersi durumda öğrenme işlemi bitmeden süreç tamamlanacağından dolayı tekrar verimsiz bir sistem gözlenecektir.

Ders 115: Derin Öğrenme Kütüphaneleri

Kütüphanelerin erişim adresleri aşağıda bulunmaktadır.

Keras: https://keras.io[26]

TensorFlow: https://www.tensorflow.org[27]

Caffe : http://caffe.berkeleyvision.org[28]

26. https://keras.io/

27. https://www.tensorflow.org/

28. http://caffe.berkeleyvision.org/

DeepLearning4J : https://deeplearning4j.org[29]

PyTorch : https://pytorch.org[30]

Kütüphaneler incelenecek olursa DeepLearning4J'in Python desteği olmadığı ve Keras kütüphanesinin diğerlerine göre daha yüksek seviye işlemler için kullanıldığı sonucuna ulaşılabilir.

29. https://deeplearning4j.org/

30. https://pytorch.org/

Derin Öğrenme için GPU'nun seçilme nedeni, bir bilgisayara yüksek sayıda GPU konumlandırabilme ve bunlar üzerine işlemleri paralel olarak dağıtabilme fırsatıdır. Bu nedenle yüksek nöron sayısına sahip YSA için GPU bazlı sistemler tercih edilmektedir.

Google şirketinin geliştirmiş olduğu Tensorflow kütüphanesi bu alandaki ilk kütüphane

olduğundan dolayı temelden başlamak ya da detaylarla ilgilenmek isteyen kullanıcılar için cazip bir seçenek olmaktadır.

Theano ise sayısal işlemleri birden fazla GPU'ya paralel olarak dağıtabildiğinden dolayı tercih edilmektedir.

Derin Öğrenme mimarisinden kullanacağımız kütüphane üst seviye mimameriye sahip olmasından dolayı Keras olacaktır.

Ders 116: Kütüphane Kurulum Komutları

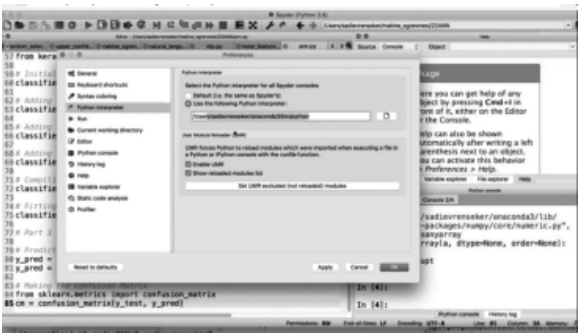
Ders 117: Derin Öğrenme Kütüphanelerinin Kurulması

http://www.bilkav.com/makine-ogrenmesi-egitimi/ [31]adresinden gerekli olan kodlar Anaco-

31. http://www.bilkav.com/makine-ogrenmesi-egitimi/

da'nın prompt kısmına girilerek kütüphanelerin kurulumu gerçekleşiyor. Python'un sadece 3. Sürümünü kulanan kullanıcılar için kod kısmındaki "pip" kodu "pip3" olarak değiştirilmelidir.

Kurulum tamamlandıktan sonra Spyder'ı yeniden başlatmak gerekmektedir.



Linux ve OS işletim sistemlerine sahip kullanıcılar için preferences üzerindeki python interpreter için doğru kaynak belirtilmelidir.

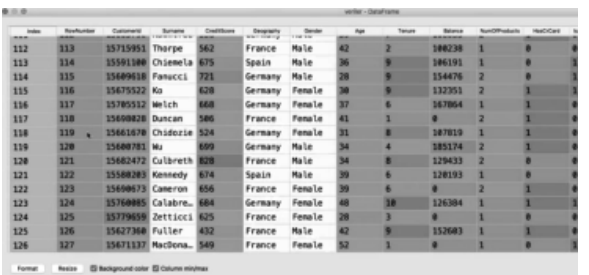
Ders 118: Problemin Tanımı ve Veri Kümesi

Bu dersimizde çoğu kurumsal şirketin de karşı karşıya kaldığı bir problem olan Müşteri Kayıp Analizi'nden bahsedilecektir. Bu problemdeki amaç, müşterinin gelecekte ne gibi davranışlar içerisinde olabileceğini önceden saptamaktır.

Yeni bir müşterinin kazanılması için harcanacak bütçe genellikle halihazırdaki müşterinin kaybedilmemesi için kullanılacak olan bütçenin 3

katıdır. Bu durumdan dolayı MKA , kurumsal firmalar için yüksek önem arz etmektedir.

Kodlama kısmı için ilk derslerimizde kullandığımız verionislemesablonu.py dosyasının gerekli bölümleri alınıyor ve üzerinde değişiklikler yapılarak veri kümemiz uygun hale getiriliyor.



Index	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	I
112	113	15719951	Thorpe	562	France	Male	42	2	100238	1	0	0
113	114	15591100	Chiemela	675	Spain	Male	36	9	106191	1	0	1
114	115	15609618	Fanucci	721	Germany	Male	28	9	154476	2	0	1
115	116	15675522	Ko	628	Germany	Female	30	9	132351	2	1	1
116	117	15705512	Welch	668	Germany	Female	37	6	167864	1	1	0
117	118	15600828	Duncan	506	France	Female	41	1	0	2	1	0
118	119	15661670	Chidozie	524	Germany	Female	31	8	107819	1	1	0
119	120	15600781	Wu	699	Germany	Male	34	4	185174	2	1	0
120	121	15682472	Culbreth	828	France	Male	34	8	129433	2	0	0
121	122	15588203	Kennedy	674	Spain	Male	39	6	120193	1	0	0
122	123	15690673	Cameron	656	France	Female	39	6	0	2	1	0
123	124	15760085	Calabre…	684	Germany	Female	48	10	126304	1	1	1
124	125	15779659	Zetticci	625	France	Female	28	3	0	1	0	0
125	126	15627360	Fuller	432	France	Male	42	9	152603	1	1	0
126	127	15671137	MacDona…	549	France	Female	52	1	0	1	0	1

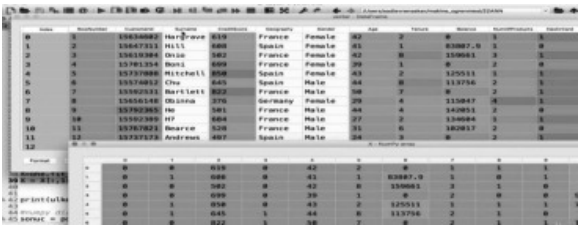
Format Resize Background color Column min/max

Kullanılacak olan veri kümesi içerisin deki benzersiz değerler (Rowid, surname) çıkartılarak

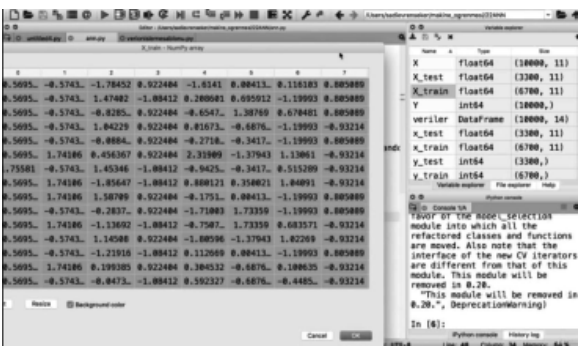
bağımsız değişken olarak kullanılabilecek kolonlar ayrıştırılıyor.

Bu problem, daha önceki modellerimize benzer şekilde olabilecek sınıflandırma ya da tahmin problemi olarak da düşünebilir fakat bu problem konumuzun gereği olarak YSA ile çözülecektir.

YSA'ı 0 ile 1 arasındaki değerleri kullandığından dolayı coğrafya ve cinsiyet gibi kolonların encode işlemine tabi tutulması gerekmektedir.



Son olarak verilerin test ve eğitim kümesi olarak bölünmesi, sonrasında ise bu verilerin ölçeklendirilmesi gerekmektedir.



Bu aşamaya kadar kullanılan kodlar aşağıdaki kısımda verilmektedir.

```
#1. kutuphaneler


import numpy as np

import matplotlib.pyplot as plt

import pandas as pd


#2.1. Veri Yukleme

veriler = pd.read_csv('Churn_Modelling.csv')

#pd.read_csv("veriler.csv")
```

```
#veri on isleme

X= veriler.iloc[:,3:13].values


Y = veriler.iloc[:,13].values

#encoder: Kategorik -> Numeric


from sklearn.preprocessing import LabelEn-
coder
```

```
le = LabelEncoder()


X[:,1] = le.fit_transform(X[:,1])
```

```
le2 = LabelEncoder()


X[:,2] = le2.fit_transform(X[:,2])
```

```
from sklearn.preprocessing import OneHotEncoder


ohe = OneHotEncoder(categorical_features=[1])


X=ohe.fit_transform(X).toarray()


X = X[:,1:]
```

```
#verilerin egitim ve test icin bolunmesi


from sklearn.cross_validation import train_test_split


x_train, x_test,y_train,y_test = train_test_split(X,Y,test_size=0.33, random_state=0)
```

```
#verilerin olceklenmesi

from sklearn.preprocessing import Standard-
Scaler

sc = StandardScaler()

X_train = sc.fit_transform(x_train)

X_test = sc.fit_transform(x_test)
```

Ders 119: YSA Kodlamasına Giriş ve Keras ile Tanışma

Bu dersimizde YSA oluşturulması üzerinde durulacaktır. Bir önceki dersimizde veri kümemiz üzerinde yapılmış olan düzenlemeler neticesinde verilerimiz YSA inşaasına hazır hale getirilmiştir.

Kodlama işlemine düzgün bir şekilde kurulduğu varsayılan keras kütüphanesinin yüklenmesiyle başlanacaktır.

```
import keras
```

Sonraki adım olarak keras kütüphane modelleri içerisinden Sequential kullanılarak bir YSA oluşturuluyor.

```
from keras.models import Sequential
```

Kodlamamıza nöron yapısını oluşturabileceğimiz Dense objesinin çağırılması sağlanıyor.

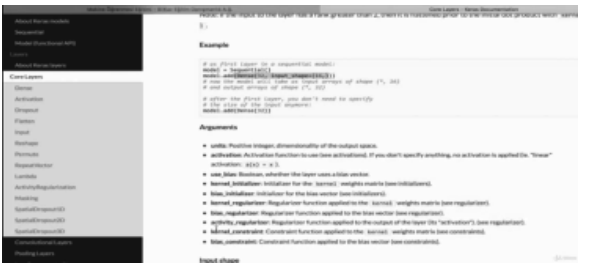
```
from keras.layers import Dense
```

Sonrasında ise YSA'nı bir sınıflandırma algoritması gibi kullanmak için Sequential ile boş bir YSA oluşturuluyor ve "classifier" nesnesine atanıyor.

```
classifier = Sequential()
```

Keras'ın mantığında YSA oluşturma işlemi, belirlenen bu obje üzerinde çeşitli parametreler(katman boyutu, katman sayısı) üzerinde

değişiklikler yapılarak gerçekleşmektedir. Bu değişiklikler ile YSA'na gizli katmanlar eklenmektedir.



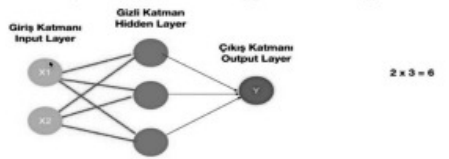
Bu parametrele ait öznel yaklaşımlar çoğunlukta olduğundan dolayı veri biliminde kullanılan algoritmaların parametre optimizasyonu o algoritmayı kullanan veri bilimcisinin insiyatifindedir. Bununla birlikte ön işleme kısmında görüldüğü üzere bazı nesnel yaklaşımlar da mevcuttur. Örneğin ön işleme kısmında bütün kolonlardan sadece sonuncu kolonun bağımlı

değişken olarak alınması ya da ilk 3 kolonun benzersiz değerlere sahip olmasından dolayı değerlendirme dışında bıraklıması bu yaklaşımlara örnek teşkil etmektedir.

Kodlama kısmına geri dönmek gerekirse öncelikle YSA'mızın ilklendirilmesi gerekmektedir.

Görselimizdeki X1 ve X2 değerleri ile işe başlıyor ve 0'a yakın bir değer ile ilklendirme işlemimiz yapılıyor.

Sonrasında ise aktivasyon fonksiyonu belirleniyor.



Son olarak ise bağımsız değişkenlerimizin sayısı kadar(11) boyut hazırlanıyor ve nöron sayıları giriş ve çıkış katmanlarındaki değişkenlerinin toplamının yarısı(6) kadar olarak belirleniyor.

classifier.add(Dense(6, init = 'uniform', activation = 'relu' , input_dim = 11))

Sonraki derste yaratılan bu yeni classifier ile problem çözülme istemi gerçekleştirilecektir.

Ders 120: Keras'a Gizli Katman Eklemek ve Bir Keras Bug'ı

Önceki dersten hatırlanacağı üzere 6 nörondan oluşan gizli katman ve 11 nöron girişine sahip bir giriş katmanını barındıran bir YSA yaratılmıştı.

İkinci gizli katman ekleniyor .

```
classifier.add(Dense(6, init = 'uniform', activation = 'relu'))
```

Tavsiye edildiği üzere giriş katmanında ve gizli katmanlarda doğrusal fonksiyonlar kullanılmakla birlikte çıkış katmanında ise Sigmoid fonksiyon kullanımalıdır.

Bir Keras bug'ı olarak buraya kadar yazılmış ve çalıştırılmış kodun tekrar çalıştırılması durumunda bir hata ile karşılaşılmaktadır. Bu durumun nedeni kütüphanelerin tekrar yüklenmesiyle alakalıdır. Bu nedenle Enable UMR kısmındaki tikin kaldırıldığından emin olmak gerekmektedir.

Ders 121: Yapay Sinir Ağını Derlemek, Eğitmek, Çalıştırmak ve Başarısı

Kodlamamıza Çıkış katmanımızın eklenmesi ile devam ediliyor. Önceki derste belirtildiği üzere bu katmanda lojistik bir şekilde çalışan Sigmoid fonksiyonu kullanılmaktadır.

Bağımlı değişkenimiz sadece bir adet olduğundan dolayı çıkış katmanımızda bir nöron hazır olarak bulunacaktır.

```
classifier.add(Dense(1, init = 'uniform', activation = 'sigmoid'))
```

Sıradaki adım olarak, YSA'nın Keras üzerinden çalışacağı prensipler belirlenmelidir. Bu nedenle ilk olarak derleme işlemimizi yapmamız ve sinapsisler üzerindeki verilerin optimizasyonu için Keras kütüphanesinde de bulunabilecek optimizer yöntemlerinden birisi seçilmelidir. Bu örneğimiz için "Adam" yötemi seçiliyor. Bu yöntem, Stokastik Alçalış yönteminin farklı bir versiyonudur. "loss" fonksiyonu olarak bağımlı değişkenimiz binomial değer içerdiğinden dolayı "binary_crossentropy" yöntemi kullanılacaktır. Son olarak ise, optimizasyonu yapılması istenen metriklerden en önemlisi olan kesinlik(accuracy) değeri kullanılıyor.

```
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy' , metrics = ['accuracy'] )
```

Bir diğer adım olarak verilerimizin eğitilmesi isteniyor ve YSA'nın öğrenim süreci boyunca kaç aşamada veya başka bir değişle turda öğrenmesini istediğimiz epochs değeri kullanıcı tarafından belirleniyor.

```
classifier.fit(X_train, y_train, epochs=50)
```

Elde edilen veriler ile tahmin işlemi yapılıyor.

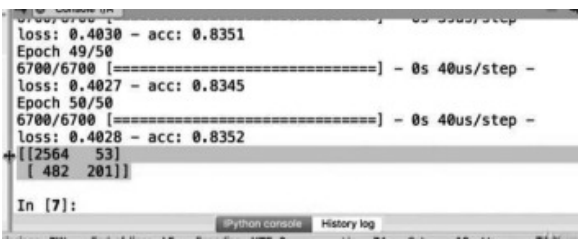
```
y_pred = classifier.predict(X_test)
```

Tahminler doğrultusunda Müşteri kayıp analizi yapılıyor

```
y_pred = (y_pred > 0.5)
```

Son işlem olarak sonuç değerleri karmaşıklık matrisinde görüntüleniyor.

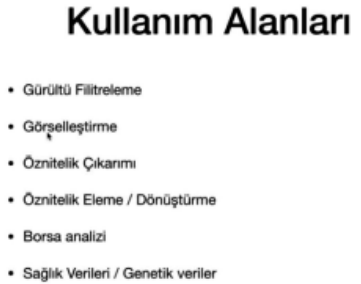
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,y_pred)

print(cm)
```



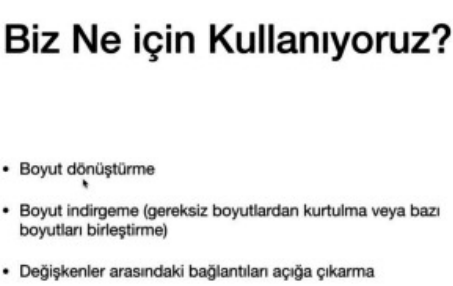
Bölüm 31:

Ders 122: PCA (Principal Component Analysis) Birincil Bileşen Analizi

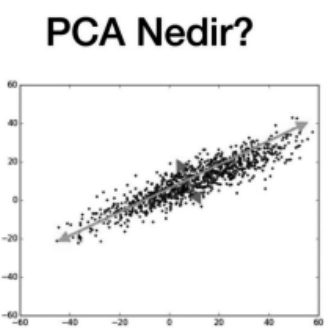
Bu derste boyut indirgemenin hangi durumlarda ve nasıl kullanıldığından söz edilecektir. Bu anlatım için PCA kullanılacaktır. PCA, çeşitli alanlarda kullanılmaktadır.



Bir veri bilimcisinin PCA'yı kullanım amacı diğer kullanıcılardan biraz daha farklıdır.

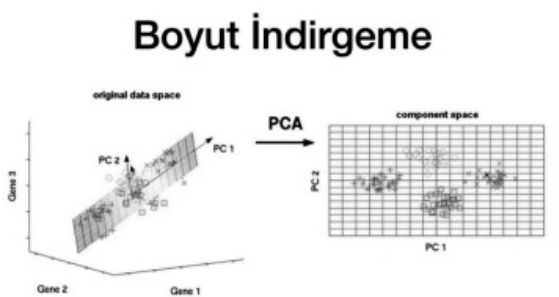


Örnek olarak iki boyutlu uzayda dağılmış verilerimizi göz önüne getirelim.



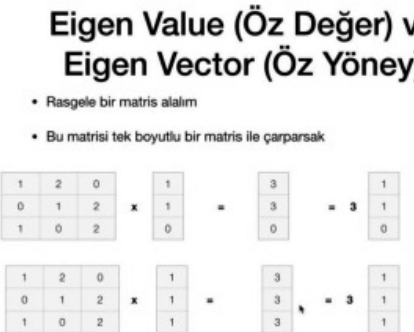
PCA'nın amacı, yeni bir boyut ile aynı x ekseni ya da y ekseni değerlerine sahip verileri birbirinden maksimum ayrışkanlıkla ayırmaya çalışmaktır. Örneğimizde görülebileceği üzere mavi renkte yeni bir boyut elde edilme yapılarak merkez noktası kırmız renk ile işaretlenmiştir. Önceki duruma göre daha verimli olduğu durumlara örnek olarak KNN algoritması verilebilir. Bir başka üzerinde düşünülmesi gereken şey ise boyut indirgeme yapıldığı takdirde veri kaybı yaşanabilme olasılığıdır. Veri kaybına örnek olarak,3 boyutlu bir uzaydaki verinin 2

boyutlu bir uzaya dönüştürülmesi sırasında birbirinden farklı iki noktanın yeni uzayda tek bir noktayla ifade edilmesi durumunda veri kaybı yaşanmaktadır. Bu durumdan kaçınmak için aynı boyut sayısına göre dönüştürme yapılmalıdır.

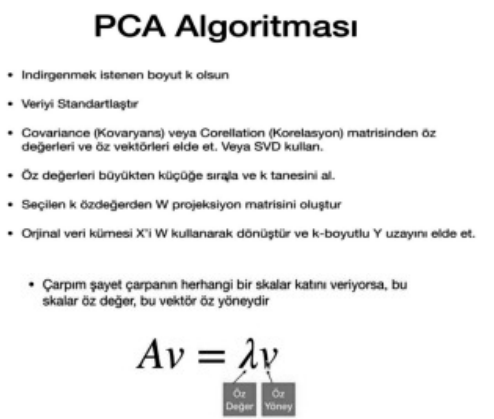


PCA algorimasının düzgün bir şekilde çalışabilmesi için öncelikle öz değer ve öz yöney değerlerinin bilinmesi gerekmektedir. Bu değerler sinyal işleme ve görsel işleme

gibi alanlarda da kullanılmaktadır.



İki farklı matrisin çarpımından sonra bir katsayı yardımıyla tekrar kendilerine dönüşebilen vektörlere Türkçe'de Öz Yöney ve bu katsayılara da Öz Değer denilmektedir.



PCA algoritması için çeşitli adımlar takip edilmelidir.

Ders 123: PCA, Python ile Kodlanması

Bu derste, web sitemizden indireilebilecek veri kümesi ile şarap ürünü hakkında müşteri segmentasyonu için kodlama yapılacaktır.

Kodlama kısmında ise daha önceden kullandığımız veri şablonumuz takip edilmektedir.

```
#1. kutuphaneler


import numpy as np


import matplotlib.pyplot as plt


import pandas as pd


# veri kümesi
```

```
veriler = pd.read_csv('Wine.csv')


X = veriler.iloc[:, 0:13].values


y = veriler.iloc[:, 13].values


# eğitim ve test kümelerinin bölünmesi

from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

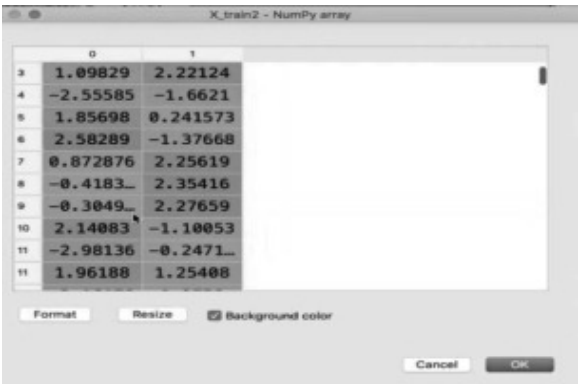
```
# Ölçekleme

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)
```

Bu işlemler sonrasında ise, toplam kolon sayısı PCA ile indirgenerek doğru tahmin yapılması istenecektir.

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)
```

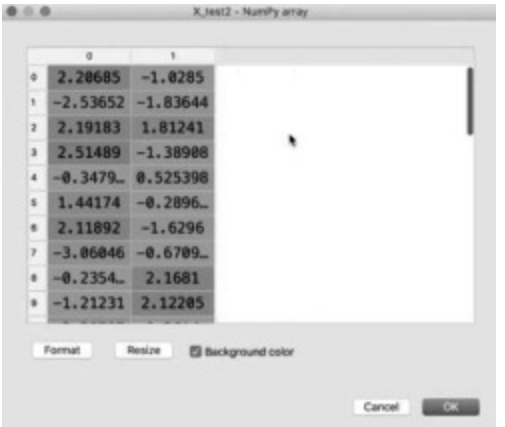
Sonraki adım olarak X_train kümesi PCA ile eğitilerek veri kümesi üzerinde uygulama yapılıyor. Bu şekilde veri kümemizin belirlediğimiz boyut sayısına indirgemesi gerçekleşmiş oldu.

```
X_train2 = pca.fit_transform(X_train)
```



Eğitim kümesinden farklı bir boyutlandırma işlemine sebep olmamak için test kümemiz için eğitim kullanılmayarak sadece dönüştürme işlemi yapılmaktadır.

X_test2 = pca.transform(X_test)



Elde edilen test ve eğitim kümelerimiz lojistik regresyon için kullanılacaktır. Boyutlandırma öncesi ve sonrası durumda oluşabilecek regresyon sonuçlarımız için iki farklı regresyon modeli oluşturuluyor.

#pca dönüşümünden önce gelen LR

```
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state=0)

classifier.fit(X_train,y_train)

#pca dönüşümünden sonra gelen LR

classifier2 = LogisticRegression(random_state=0)

classifier2.fit(X_train2,y_train)
```

```
#tahminler


y_pred = classifier.predict(X_test)


y_pred2 = classifier2.predict(X_test2)
```

Sonrasında ise bu regresyon modellerinini üç farklı karmaşıklık matrisi yardımıyla aşağıdaki görselde görüleceği gibi kesinlik değerleri üzerinden karşılaştırılıyor.

```
from sklearn.metrics import confusion_matrix

#actual / PCA olmadan çıkan sonuç

print('gercek / PCAsiz')

cm = confusion_matrix(y_test,y_pred)

print(cm)
```

```
#actual / PCA sonrası çıkan sonuç


print("gercek / pca ile")


cm2 = confusion_matrix(y_test,y_pred2)


print(cm2)
```

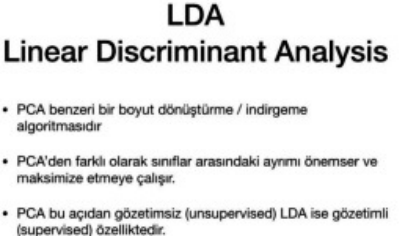
```
#PCA sonrası / PCA öncesi


print('pcasiz ve pcali')


cm3 = confusion_matrix(y_pred,y_pred2)


print(cm3)
```

```
In [2]: runfile('C:/Users/f
gercek / PCAsiz
[[14  0  0]
 [ 0 16  0]
 [ 0  0  6]]
gercek / pca ile
[[14  0  0]
 [ 1 15  0]
 [ 0  0  6]]
pcasiz ve pcali
[[14  0  0]
 [ 1 15  0]
 [ 0  0  6]]
```

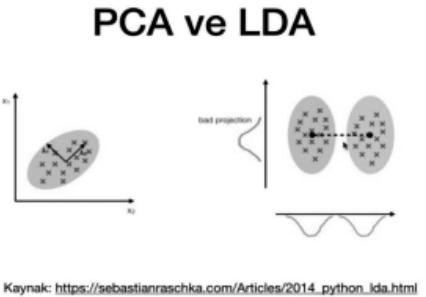
Sonuç olarak boyut indirgeme işlemi sonrasında hata miktarı artmasına rağmen kullanılan veri boyutunda ciddi düzeyde bir azalma görülmektedir.

Ders 124: LDA: Linear Discriminant Analysis (Doğrusal Ayrışım Analizi)

LDA algoritması, gerek boyut indirgeme için kullanımı gerek ise diğer boyut algortima özelliklerine sahip olmasından dolayı PCA'ya benzetilebilmektedir fakat bu algoritmalar arasında bazı farklılıklar bulunmaktadır.



En önemli fark ise gözetimli gözetimsiz yapı olma durumu gösterilebilir. Bu nedenden dolayı LDA, sınıflar arasındaki farkı maksimize etmeye çalışmaktadır.



Ders 125: LDA: Python ile Kodlaması

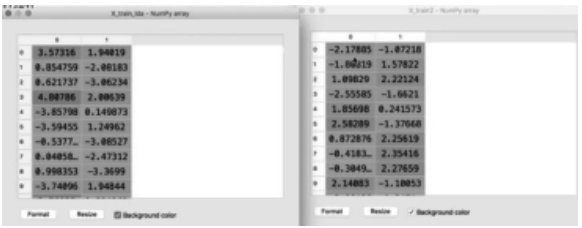
Kodlama kısmında, PCA için kullanılan veri kümesi üzerinden işlem yapılabilmesi için PCA koduna devam ediliyor. Aynı şablon kullanılarak karmaşıklık matrisi vasıtasıyla LDA ile boyut indirgendikten sonra yapılan tahminlerin kesinlik değerleri ile orijinal tahminlerin kesinlik değerleri kıyaslanıyor.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n_components = 2)
```

LDA gözetimli bir algoritma olmasından dolayı sınıfları öğrenmesi gerekmekte ve bunun için "y_train" değeri de kodumuza eklenmektedir.

```
X_train_lda = lda.fit_transform(X_train,y_train)

X_test_lda = lda.transform(X_test)
```

Görselimizde, PCA ve LDA işlemleri sonrası ortaya çıkan eğitim kümeleri görüntülenmektedir.



#LDA donusumunden sonra

classifier_lda = LogisticRegression(random_state=0)

classifier_lda.fit(X_train_lda,y_train)

#LDA verisini tahmin et

```
y_pred_lda = classifier_lda.predict(X_test_lda)
```

```
#LDA sonrası / orijinal

print('lda ve orijinal')

cm4 = confusion_matrix(y_pred,y_pred_lda)

print(cm4)
```

```
In [3]: runfile('C:/Users/fuko6/Desktop/makine/lda.py', wdir='C:/Users/fuko6/
Desktop/makine')
gercek / PCAsız
[[14  0  0]
 [ 0 16  0]
 [ 0  0  6]]
gercek / pca ile
[[14  0  0]
 [ 1 15  0]
 [ 0  0  6]]
pcasız ve pcalı
[[14  0  0]
 [ 1 15  0]
 [ 0  0  6]]
lda ve orijinal
[[14  0  0]
 [ 0 16  0]
 [ 0  0  6]]
```

Sonuç olarak, bu veri kümesi bazında,gözetim farkından dolayı LDA, PCA algoritmasından daha başarılı sonuç üretmektedir.

Bölüm 32:

Ders 126: Model Semi: Giriş

Model Semi: Giriş

Bölüm 32, Ders 126

Model Seçimi ve Arttırımı Yöntemleri Bölümüne Hoş Geldiniz

Çok sık gelen ve doğal olarak her veri bilimi uzmanının bilmesi gereken önemli konulardan birisi model seçiminin nasıl yapılacağıdır. Bu bölümde, çok sayıda alternatif modelden nasıl seçim yapılacağını öğrenirken aşağıdaki soruları cevaplamaya çalışacağız.

1. bağlantı / varyans (bias / variance) ikileminde bir model nasıl seçilir ve performansı nasıl değerlendirilir
2. Hiper parametreler için değer seçimi nasıl yapılır (hiper parametre öğrenme sürecinde yer almayan parametrelerdir)
3. iş süreçleri analiz edilen bir problem için en uygun modeli nasıl buluruz?

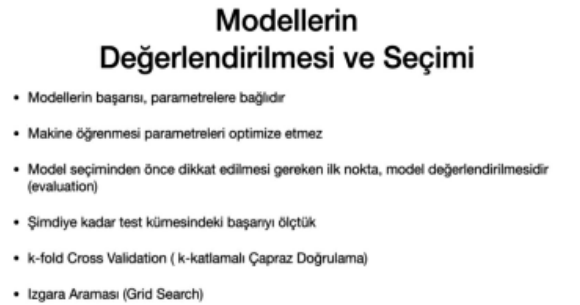
Yukarıdaki soruların cevaplanabilmesi için aşağıdaki iki yöntemi bu bölümde anlatacağız:

1. k-katlamalı çapraz doğrulama (k-fold Cross validation)
2. Izgara Araması (Grid Search)

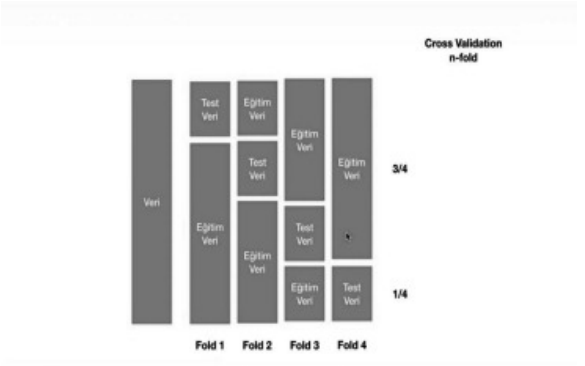
Dersin sonunda, Bonus kısmında anlattığımız XGBoost algoritması ile bu derste öğrendiklerimizi birleştireceğiz.

Ders 127: Model Değerlendirilmesi: k-fold Cross Validation

Bu dersimizde makine öğrenimi konusunda önemli bir yere sahip olan model seçimi incelenecektir. Aynı problemi çözen birden fazla model bulunması durumunda hangi modelin seçilmesi gerektiğine karar verilmesi veya hangi problem için hangi modelin seçilmesi gerektiği gibi konular özü bakımından olası sonuçlar için kritik öneme sahiptir.

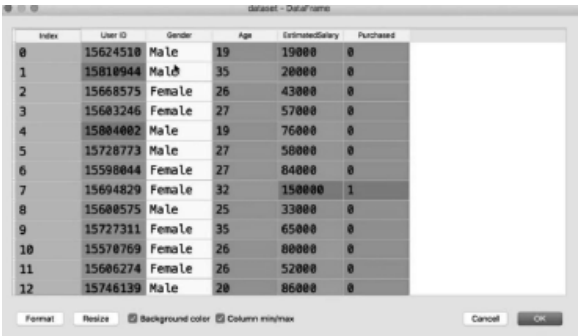


k-fold yönteminde belirlenen sayıya göre eğitim ve test kümesi bölünerek bu sayı kadar değerlendirme yapılıyor. Bu şekilde verilere eğitim ya da test kümesi olma fırsatı veriliyor.



k-fold'un kodlama kısmında ise yeni veri kümemiz ile eski veri şablonumuz kullanılıyor.

Bu veri kümemizde değerler kullanılarak envanterimizdeki bir ürünün alınıp-alınmama durumu gözlemlenmek isteniyor.



Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0

Format Resize Background color Column min/max Cancel OK

İlk kısım olarak SVM algoritması kullanılıyor.

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd
```

```
# veri kümesi

dataset = pd.read_csv('Social_Network_Ads.csv')

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values
```

```
# eğitim ve test kümelerinin bölünmesi

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Ölçekleme

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)


X_test = sc.transform(X_test)
```

```
# SVM

from sklearn.svm import SVC

classifier = SVC(kernel = 'rbf', random_state =
0)

classifier.fit(X_train, y_train)
```

```
# Tahminler


y_pred = classifier.predict(X_test)
```

```
# Confusion Matrix

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print(cm)
```

Sonraki adım olarak k-fold yöntemine bağlı olarak gerekli parametreler girilerek başarı ortalama değeri ve standart sapma değerleri görüntüleniyor. Standart sapma değerinin düşük gözlemlenmesi pozitif bir sonuç doğurmakla birlikte SVM yerine KNN gibi farklı algoritmalar kullanılması sistemdeki başarı oranının değiştirmektedir.

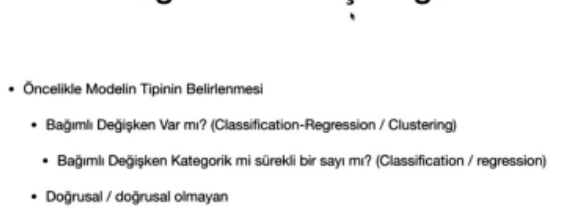
```
#k-katlamali capraz dogrulama


from sklearn.model_selection import cross_val_score

'''

    1. estimator : classifier (bizim durum)
    2. X
    3. Y
    4. cv : kaç katlamalı

'''


basari = cross_val_score(estimator = classifier, X=X_train, y=y_train , cv = 4)
```

```
print(basari.mean())


print(basari.std())
```

```
[[64  4]
 [ 3 29]]
0.900303461356093
0.03527298302303932
```

Ders 128: Model Seçimi : GridSearchCV

GridSearchCV, optimal model seçimi konusunda kullanılan yöntemlerden birisidir. Bu yöntem için öncelikle veriler üzerinden değişkenlerin durumları değerlendirilmelidir. Bir sonraki aşamada ise seçilmiş olan algoritmanın parametrelerinin optimizasyonu gerekmektedir.



Önceki kod kısmımıza ek olarak SVM algoritmamızın parametrelerinin optimize edilmesi için çalışılacaktır. Bunun için optimize edilmesi istenen değerler ve bazı vakalar için olası eşitlik durumları belirlenmelidir. Daha detaylı bir optimizayson için ise bazı metriklerin içerisindeki parametreler için de değişimler yapılabilir.

parametremetre optimizasyonu ve algoritma seçimi from

```
sklearn.model_selection import GridSearchCV
p =


[{'C':[1,2,3,4,5],'kernel':['linear']},
{'C':[1,2,3,4,5]


,'kernel':['rbf'], 'gamma':[1,0.5,0.1,0.01,0.001]} ]
```

Öncelikle eldeki değerler ile deneme yapılarak optimizasyon için daha küçük değer aralıkları bulunması gibi teknikler de saha çalışmalarında mevcuttur.

Tanımların detaylandırılması yapılıyor.

'''

GSCV parametreleri

estimator : sınıflandırma algoritması (neyi optimize etmek istediğimiz)

param_grid : parametreler/ denenecekler

scoring: neye göre skorlanacak : örn : accuracy

cv : kaç katlamalı olacağı

n_jobs : aynı anda çalışacak iş

'''

gs = GridSearchCV(estimator= classifier, #SVM algoritması

param_grid = p,

scoring = 'accuracy',

cv = 10,

n_jobs = -1)

GridSearchCV, SVM algoritmasının üzerinde konumlanmış bir akıl gibi çalışmakta ve eldeki veriler eğitilmektedir.

grid_search = gs.fit(X_train,y_train)

En optimal değerlerin saptanması içi ise ".best_score_" ve "best_params_" kullanılıyor.

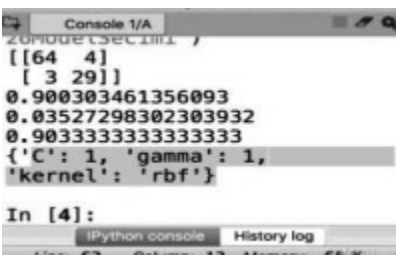
eniyisonuc = grid_search.best_score_

eniyiparametreler = grid_search.best_params_

Son adım olarak elde edilen değerler görüntüleniyor.

print(eniyisonuc)

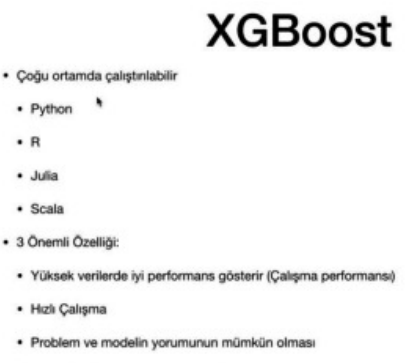
print(eniyiparametreler)



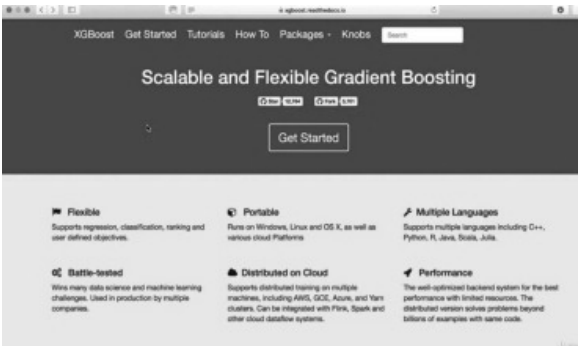
Bölüm 33:

Ders 129 : XGBoost: Giriş , Özellikleri ve Kaynaklar

Bu derste, veri bilimi dünyasındaki çeşitli yarışmalarda başarı sahibi algoritma olan XGBoost'dan bahsedilecektir. Bu nedenle, veri bilimcisinin rahatça kullanılabilecek algoritmalardan bir tanesidir.



Ders 130 : XGBoost: Kurulum ve Python Kodlaması

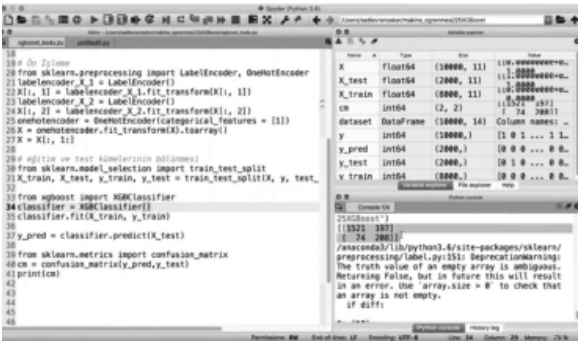


Kurulum için öncelikle kurulum sayfasına giriş yapılıyor ve Get started kısmından kurulum için gerekli olan Python kodu elde ediliyor.

Başka bir kurulum seçeneği olarak da github üzerinden kaynak kodların indirilmesi gösterilebilmektedir.

Kodlama kısmında “Churn_Modeling.csv” veri seti kullanılmıştır.

XGBoost için yazılmış olan kodlama kısmı ise aşağıdaki gibidir.



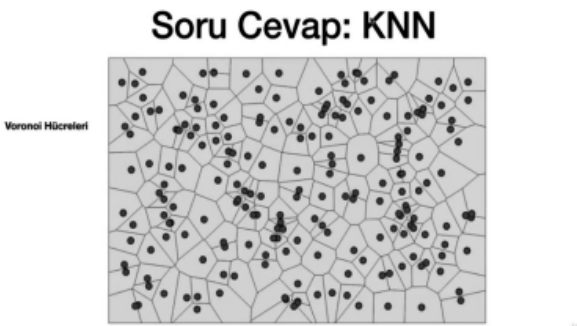
Sonuç olarak, XGBoost ile %85 oranında kesinlik sağlanmaktadır.

Bölüm 34:

Ders 131 : KNN'deki K değeri neye göre alınır?

Bu sorunun net bir cevabı olmamakla birlikte dersin sonunda bu değerin hesaplanmasına yönelik bir formül okurlara sunulacaktır.

Öncelikle KNN algoritmasının yapısı detaylı olarak anlaşılmalıdır. Bu nedenle K değeri 1'e eşit olan voronoi hücre kolonisi incelenmektedir. Voronoi hücreleri kendi arasında birleşerek daha büyük bir hücre yapısına da dönüşebilmektedir.



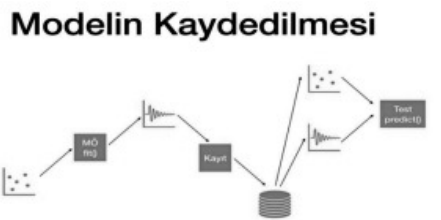
Örneğimizi incelediğimizde farklı renkteki hücreler arasında bazı farklı renkte adalar görmekteyiz. Bu görüntünün nedeni K değeridir. Şayet K değeri 3 olsaydı bu hücreler diğer adalara baskın gelerek görülmelerini olanaksız kılacaktı.

K değerinin 1 olma durumunda ezberleme riski oluşabilirken 2 olma durumunda eşit dağılım tehlikesiyle karşı karşıya kalınabilinmektedir. Bu nedenle veri bilimcinin tecrübesi doğrultusunda verdiği kararlar sonuçları etkileme bağlamında elde edilebilecek sonuç değerleri için yüksek önem teşkil etmektedir.

Ders 132 : Modelinin Kaydedilmesi ve Tekrar Kullanılması (Pickle Kütüphanesi)

Bu dersimizde makine öğrenmesi sonucunda elde edilen eğitilmiş bir modelin, ona her erişim istendiğinde tekrar öğrenme aşamasından geçmemesi için kaydedilerek kullanılması kullanıcıya zaman kazandırmaktadır. Bu işlem için 3 farklı kütüphane bulunmaktadır.

Bu kütüphanelerden biri olan ve farklı platformda eğitilebilmesi nedeniyle Pmml diğerlerine göre üstünlük sağlamaktadır.



Modelin kaydedilme sürecinde öncelikle veriler üzerinden makine öğrenimi gerçekleşerek bir model ortaya çıkmaktadır. Sonraki adım olarak, elde edilen bu model diske kaydedilecek ve bu sayede tekrar kullanımı sırasında herhangi bir eğitim sürecine tabi tutulmadan model okunabilecek ve yeni veriler üzerindeki tahmin işlemi gerçekleştirebilecektir.

Doğrusal regresyonumuz sonucunda eğitilmiş obje pickle yardımıyla belirlediğimiz dosya konumuna kaydediliyor.

Sonraki kullanımlar için hazır hale getirilen model tekrar yüklenerek tahmin yapılması isteniyor ve aynı değerler bulunuyor.

Ders 133: Verilerin Ölçeklendirilmesi(Standart/Normalleştirilmesi)

Verilerin Ölçeklenmesi (Standart/Normalleştirilmesi)

Bölüm 34, Ders 133

Gelen bir soru üzerine, farklı istatistiksel özelliklere sahip verilerin nasıl bir işlemeye tabi tutulacağını anlatmaya çalışacağız (bu içerik daha sonra video haline de getirilecektir, sorunun cevaplanması için önceden yazılı içerik halinde sunulmuştur).

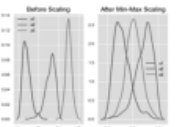
Python ile makine öğrenmesi dünyasında (scikit-learn) temel olarak 4 farklı sınıftan bu işlemler yapılabilir.

MinMaxScaler

Bu sınıfın amacı, verilerin 0 - 1 aralığına indirilmesi ve bu sırada verilerin birbirine göre görece olarak mesafe oranlarını korumaktır. Örneğin 100 - 200 aralığında dağılan verilerden iki tane sayıyı ele alalım. (bu sayılar arasındaki mesafe 30 ise, 0-1 aralığına inince 0,3 olacaktır (100 - 200 aralığı 100 birimlik min - Max aralığına sahiptir ve yeni min 0 ve yeni maks 1 olacak şekilde yeniden ölçeklendirilirler).

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Yukarıdaki formülde de gösterildiği gibi, herhangi bir verinin ölçeklenmiş hali, o verinin bulunduğu serideki minimum değere uzaklığının, o serideki maksimum ve minimum değerler arasındaki farka bölünmesidir.



Yukarıda soldaki dağılık 3 farklı veri kolonu için farklı aralıklarda dağılan verileri ifade etmektedir (x eksenindeki değerler ilk kolon (x1) için 0 - 30 aralığında, ikinci kolon (x2) için 5 - 45 aralığında ve son kolon (x3) için 35-65 aralığındadır). (Bu verilerin tamamı 0 - 1 aralığına normalleştirilmiştir (sağdaki resim) ve artık birbiri ile karşılaştırılabilir bir değer kazanmıştır.

Python ile kodlarken aşağıdaki kod satırları kullanılabilir (df isimli bir dataframe'in daha önceden tanımlı olduğu kabul edilmiştir).

```
from sklearn import preprocessing

scaler = preprocessing.MinMaxScaler()
scaled_df = scaler.fit_transform(df)
```

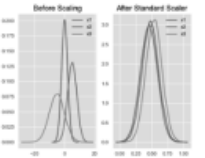
scikit-learn kütüphanesi linki : preprocessing.MinMaxScaler

Standard Scaler

Bu ölçekleme (scaler) yönteminı için verilerin normal dağılıma saygın dağılığı kabulü yapılır (aksi durumlarda kullanımı risk içermektedir). Verilerin ortalama değeri 0 olacak ve standart sapma 1 olacak şekilde yeniden verileri ölçeklenir.

$$z = \frac{X - \mu}{\sigma}$$

Yukarıda da formülü verildiği üzere, herhangi bir verinin ölçeklenmiş hali, o kolonun ortalama değerinin standart sapmasına bölümü ile bulunur.



Yukarıdaki şekilde de gösterildiği gibi, bütün veriler aynı x aralığına getirilmenin yanında ortalama değerleri ve standart sapmaları da sabitlenmiştir (solda ölçekleme öncesi ve sağda ölçekleme sonrası gösterilmektedir).

sciklt-learn kütüphanesi ile kodlama için aşağıdaki komutlar kullanılabilir.

```
from sklearn import preprocessing

scaler = preprocessing.StandardScaler()
scaled_df = scaler.fit_transform(df)
```

sciklt-learn kütüphanesine link : http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

RobustScaler

bu ölçeklemede ise bütün veriler, orijinal veri kümesindeki sırası korunarak ölçeklenir. Min-Max ölçeklemesinden en büyük farkı, aykırı / marjinal (outlier) verilere karşı dayanıklı (robust) olmasıdır.

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

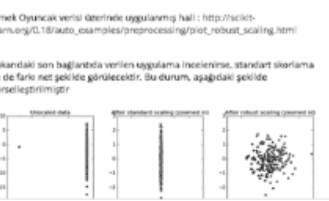
Python kodu aşağıdaki şekilde yazılabilir:

```
scaler = preprocessing.RobustScaler()
robust_scaled_df = scaler.fit_transform(x)
```

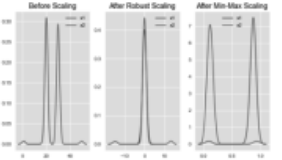
sciklt-learn kütüphanesi linki : http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

Örnek Oyuncak verisi üzerinde uygulanmış hali : http://scikit-learn.org/0.18/auto_examples/preprocessing/plot_robust_scaling.html

Yukarıdaki son bağlantıda verilen uygulama incelenirse, standart skorlama ile de farkı net şekilde görülecektir. Bu durum, aşağıdaki şekilde görselleştirilmiştir



Solda verinin ölçeklenmemiş hali, ortada standard skor ile ölçeklenmiş hali ve en sağda ise robustscaling ile ölçeklenmiş hali gösterilmektedir. Görüleceği üzere ölçekleme sonrası iki boyutta da verilerin marjinal verilerden etkilenmemiş hali ölçeklendirilmiştir (verilerin ölçeklenmemiş halindeki tek başına duran mavi noktaya dikkat ediniz ve standart ölçeklemenin bu basit tek bir nokta yüzünden bir doğru üzerinde ölçekleme sonucu çıkardığını gözlemleyiniz).



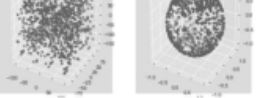
Orijinal verilere (solda) Robust Scaling uygulandıktan sonra (ortada) min-max ölçeklemesine göre (en sağda) marjinal verilerin etkilerinden arındırılmış sonuçlar üretildiği görülebilir.

Normalizer

Veriler her zaman iki boyutlu veya tek boyutlu uzayda ifade edilmez. Verilerin n boyutlu bir uzayda (n tane farklı kolon / öznitelik içermesi durumu) ifade edilmesi halinde, her boyut için farklı bir işleme tabi tutulması yerine, bütün boyutların tek bir formülde ölçekleme için kullanıldığı yöntemdir.

$$\frac{x_i}{\sqrt{x_i^2 + y_i^2 + z_i^2}}$$

Örneğin yukarıdaki formülde verilen 3 boyutlu (boyutların x, y ve z olduğu kabul edilmiştir) uzaydaki bir veri noktasının normalleştirilme sonrası alacağı değer gösterilmiştir. Buradaki normalizer yöntemi aslında bir noktanın her üç boyutta da aldığı uzaklık değerlerine kartezyen mesafe ile ölçülmektedir.



Bölüm 35:

Ders 134: CNN (Convolutional Neural Networks) Giriş

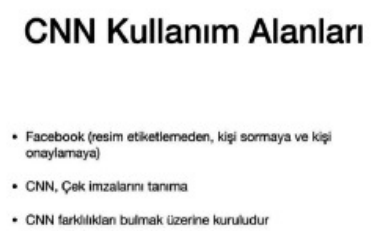
Bu derste görsel işleme ve sinyal işleme gibi son dönemdeki popüler konular için kullanılan yöntem olan CNN tanıtılacaktır.

Örneğin Görsel işleme kısmını ele alıcak olursak görsellerin makine öğrenimi için insanlar tarafından etiketlenmesi gerekmektedir fakat bazı veriler insanlar için bile farklı etiketlere sahip olabilmektedir.Örnek olarak arkaplan rengi bu farklılarla sahip olabilmektedir.

Makineler için etiketleme ihtiyacının sebebi ise, görsellerin makineler tarafından renklerine bağlı olarak ve sadece sayı formatı altında görülebilmesidir.


Evrişimsel Ağlar

Filtreleme kısmında bir pikselin çevresindeki pikseller bir operatöre bağlanıyor ve işlem bitiminden sonra o pikselin sağına kayma işlemi yapılarak aynı işlem yapılımaya devam ediyor.

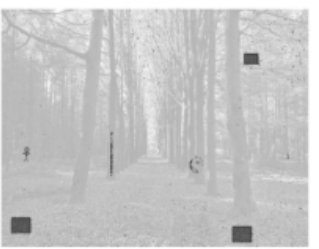


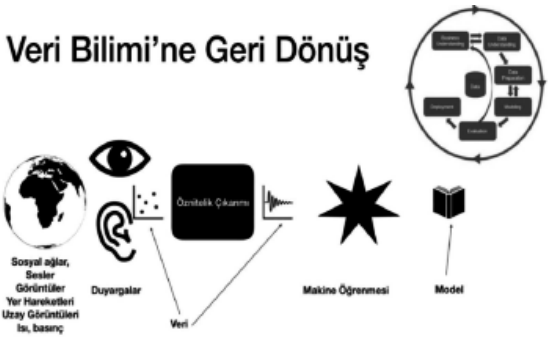
Görsel işleme kısmında, etiketleme işlemine örnek olarak aşağıdaki örnekler gösterilebilir.

Örnek olarak, iki benzer resim arasındaki ve insan gözü için zor seçime sahip farklılar görsel işleme yöntemi için kullanılan YSAlar ile makineler için saptanması oldukça kolay bir hale gelmektedir.

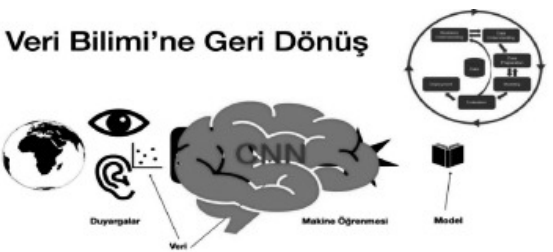
CNN

CNN



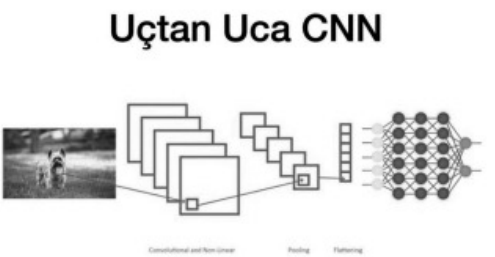


Veri bilimi aşamaları incelendiği zaman CNN'in yeri kolayca belirlenebilmektedir.



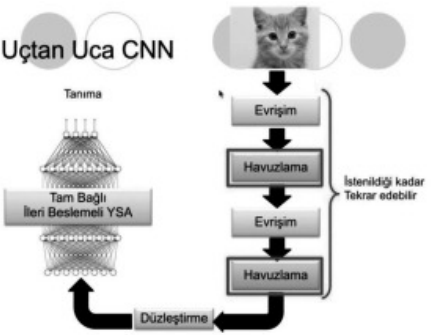
Ders 135: Uçtan Uca CNN Çalışması

Bu dersimizde CNN uygulama alanlarından biri olan Resim İşleme kısmında kullanılan filtreleme yöntemi üzerinde durulacaktır.

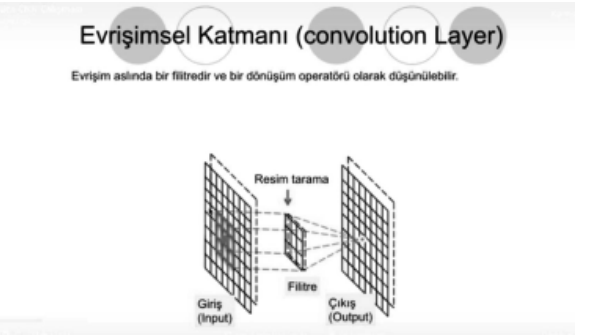


Filtreleme kısmı için ilk olarak bir girdiye ihtiyaç vardır. Bu konu özelinde, CNN'nin çalışma adımları incelendiğinde üç temel aşama gözlenmektedir.

Bu aşamalar Evrişim, Havuzlama ve Düzleştirme'dir.

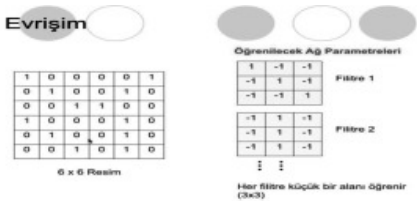


İlk aşamamız, kullanılan Yapay Sinir Ağları'na da ismini veren Convolution(Evrişim)'dur.

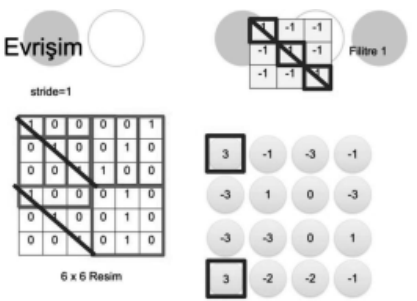


Bu aşamada çerçeve içindeki piksellere filtreleme uygulanarak elde edilen sonuç çıkış katmanına yansıtılıyor. Bu tür örneklere bağlı olarak pencere boyutları, sistemin başarısını etkilediğinden dolayı kritik bir öneme sahiptir.

Örnek olarak büyük bir resim içerisindeki küçük alanlar incelenerek çok sayıda filtre çıkarımı yapılıyor.

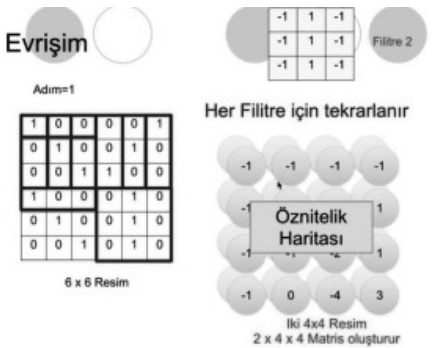


Örnek olarak belirlenen bir filtre bütün resim üzerinde işleme tabi tutuluyor ve son olarak Convolution Matrisi oluşturuluyor. Bu matris sonuç çıktımızı oluşturmaktadır.



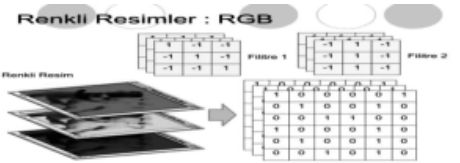
Bu işlem sonrasında ise aranan filtrenin, elde

edilen matrisimiz ile resmin hangi noktalarında olacağı saptanabilmektedir. Sonrasında ise bu işlem bütün filtrelere için kullanılmakta ve Öz Nitelik Haritası çıkartılmaktadır.

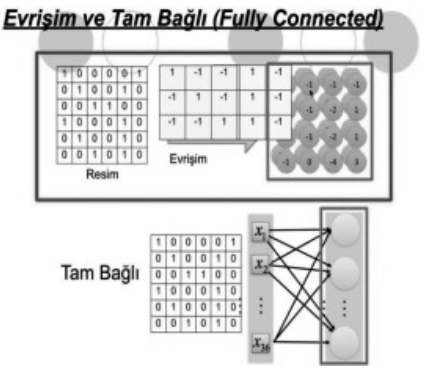


Renkli resimlerde ise, her renk için ayrı ayrı renk filtreleri uygulanacak ve renk farklılıkların saptanması olanak dahiline indirgenecektir.

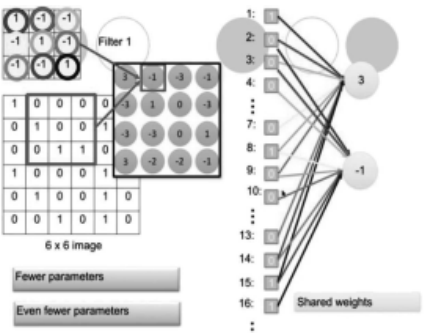
PYTHON İLE MAKINE ÖĞRENMESI689



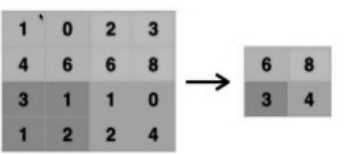
Convolution , sadece anlamlı olan noktaların belirlenmesinde kullanılırken tam bağlı matriste ise resimdeki bütün pikseller için girdi katmanı ve bunların bağlantısının oluşturulması söz konusudur. Bu durumun sonucu olarak bağlantı ve nöron sayısı yüksek düzeyde artış göstermektedir.



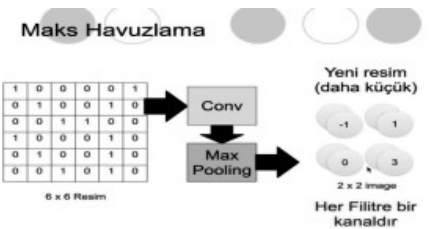
İkinci aşama olan pooling (havuzlama) da resmin alt alanlarındaki maksimum değerler çekilerek boyut indirgeme işlemi yapılmaktadır.



Havuzlama (Pooling) : Maks Havuzlama / Ortalama Havuzlama

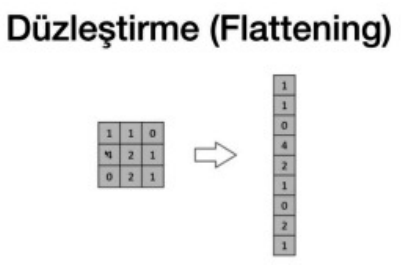


Bu işlem öncesinde bulunan maksimum değere sahip olmayan diğer değerler, CNN için değersiz kabul edilmektedir.



Ders 136: CNN Görsel Uygulama ve Düzleştirme (Flattening)

Bu dersimizde CNN'in son aşaması olan Flattening(Düzleştirme) incelenecektir. Düzleştirme işlemi, Evrişim aşaması sonucunda ortaya çıkan matrisin tek boyutlu bir dizi veya başka bir değişle girdi nöronlarına çevrilmesi için kullanılmaktadır.



http://www.cs.cmu.edu/~aharley/vis/conv/flat.html [32]adresinden bu aşamaların örneklerine ulaşmak mümkündür.

32. http://www.cs.cmu.edu/~aharley/vis/conv/flat.html

Ders 137: CNN: Python ile Kodlama

Bu dersimizde diğer kodlama derslerimizden farklı olarak hazır kod üzerinden gidilmiştir. Hazır olan bu kodlara http://www.bilkav.com/makine-ogrenmesi-egitimi/ [33]adresi üzerinden erişilebilir. Bu istisnai durumun neden veri kümemizin büyüklüğünden kaynaklı olarak işlem süremizin yüksek olmasıdır. Bu veri setimizde kadın ve erkekler ait resimler bulunmakta ve algorit-

33. http://www.bilkav.com/makine-ogrenmesi-egitimi/

mamıza cinsiyet tahmini yaptırılmaya çalışılmaktadır.

Bu veri setleri ve onlara ait etiketler, kullanıcılar tarafından değiştirilebilir.

Kodlama kısmında ise CNN aşamaları uygulandıktan sonra Yapay Sinir Ağları oluşturuluyor. Sonrasında ise ImageDataGenerator ile görseller bilgisayar donanımı için verimli bir şekilde çalışıyor. Son olarak ise kadın-erkek ayrımı yapılabilmesi için bir tetikleme mekanizması oluşturuluyor ve verilerin eğitilmesi sonrasında ortaya çıkan değerler ile tahminler yapılıyor. Bu tahmin değerleriyle gerçek değerler arasındaki başarı oranı(kesinlik(accuracy)) karmaşıklık matrisi yardımıyla görüntüleniyor.

```
from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense
```

ilkleme

classifier = Sequential()

- **Adım 1 - Convolution**

```
classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
```

Adım 2 - Pooling

classifier.add(MaxPooling2D(pool_size = (2, 2)))

- **2. convolution katmanı**

classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))

classifier.add(MaxPooling2D(pool_size = (2, 2)))

Adım 3 - Flattening

classifier.add(Flatten())

Adım 4 - YSA

classifier.add(Dense(output_dim = 128, activation = 'relu'))

classifier.add(Dense(output_dim = 1, activation = 'sigmoid'))

CNN

```
classifier.compile(optimizer = 'adam', loss =
'binary_crossentropy', metrics = ['accuracy'])
```

```
# CNN ve resimler

from keras.preprocessing.image import ImageDataGener-
ator

train_datagen = ImageDataGenerator(rescale = 1./255,

shear_range = 0.2,

zoom_range = 0.2,

horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255,
```

```
shear_range = 0.2,
zoom_range = 0.2,

horizontal_flip = True)

training_set =
train_datagen.flow_from_directory('veriler/training_set',

target_size = (64, 64),

batch_size = 1,

class_mode = 'binary')
```

```
test_set = test_datagen.flow_from_directory('veriler/test_set',

target_size = (64, 64),

batch_size = 1,

class_mode = 'binary')

classifier.fit_generator(training_set,

samples_per_epoch = 8000,

nb_epoch = 1,
validation_data = test_set,
```

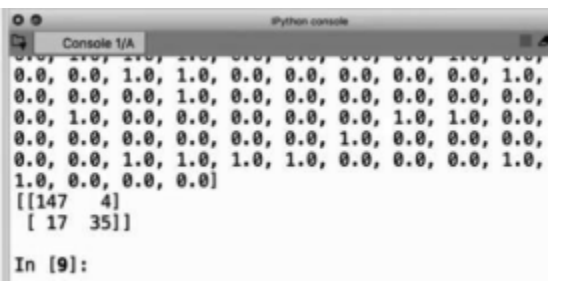
```
nb_val_samples = 2000)

import numpy as np

import pandas as pd

test_set.reset()

pred=classifier.predict_generator(test_set,verbose=1)
#pred = list(map(round,pred))

pred[pred > .5] = 1
```

```
pred[pred <= .5] = 0


print('prediction gecti')


#labels = (training_set.class_indices)
test_labels = []


for i in range(0,int(203)):


test_labels.extend(np.array(test_set[i][1]))


print('test_labels')
print(test_labels)
```

```
#labels = (training_set.class_indices)


'''


idx = []


for i in test_set:

ixx = (test_set.batch_index - 1) * test_set.batch_size


ixx = test_set.filenames[ixx : ixx + test_set.batch_size]


idx.append(ixx)


print(i)
```

```
print(idx)
'''
dosyaisimleri = test_set.filenames

#abc = test_set.

#print(idx)
#test_labels = test_set.
sonuc = pd.DataFrame()
sonuc['dosyaisimleri']= dosyaisimleri

sonuc['tahminler'] = pred

sonuc['test'] = test_labels
```

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(test_labels, pred)

print (cm)
```



İyi Çalışmalar Dileriz...