Python ile Uçtan Uca Veri Bilimi

Engin Bozaba

Değerli okurlarım kitamıza başlamadan önce bir ricam olacak.

Kitapta hata gördüğünüz de hemen kızmayın, sizlerin de desteğiyle hatalarımızı ve eksikliklerimizi tarafıma bildirmeniz durumunda bir sonraki baskıda düzeltilecektir. :)

Engin BOZABA

muhendis.github.io

facebook.com/EnginBozabaOfficial

twitter.com/enginbozaba

instagram.com/enginbozaba

linkedin.com/in/enginbozaba

udemy.com/user/engin-bozaba-2

youtube.com/c/EnginBozaba

github.com/muhendis

enginbozaba@gmail.com

İçindekiler

Bölüm 1: Giriş ve Kurulumlar 1

Veri Bilimi 2

Anaconda ile Proje Ortamı Kurulumu 3

Windows İşletim Sistemi için Kurulum 4

macOS İşletim Sistemi için Kurulum 5

Linux İşletim Sistemi için Kurulum 6

Anaconda Ortamını Tanıma 6

Anaconda Navigator 7

Jupyter QtConsole ile Konuları Öğrenme 8

Jupyter Lab ile Proje Geliştirme 9

Amazon Web Services Hesabı Oluşturma 12

Neler Öğrendik? 14

Bölüm 2: İş Problemini Anlamak 17

Sorunu Çerçevelendirme 18

İş Problemlerini Tanımlama 19

Veri Bilimi Dışı Çözümlerin Belirlenmesi 19

Çıktı Kullanımının Tanımlanması 20

Yanlış Sonuçların Yönetilmesi 21

Veri Kaynaklarını Tanımlama 21

Verileri İşlemek için Kütüphanelerin Belirlenmesi 21

Ürünün Nereden Hizmet Vereceği Yeri Belirleme 22

Neler Öğrendik? 23

BÖLÜM 3: KULLANILACAK KÜTÜPHANELERİ VE TEKNOLOJİLERİNİ TANIMAK 25

NumPy ile Matematiksel Hesaplama 26

NumPy Veri Tipleri 28

Tamsayılar 28

İşaretsiz Tamsayılar 29

Mantıksal İfadeler 29

Ondalıklı Sayılar 30

Karmaşık sayılar 31

NumPy Dizileri Oluşturma 32

NumPy Kütüphanesini Kullanarak NumPy dizilerini oluşturma 32

Var Olan Python list veya tuple Dosyalarını Kullanarak Numpy Dizisi Oluşturma 41

NumPy Dizilerini İndeksleme ve Dilimleme 43

NumPy Dizilerinin Manipülasyonu 46

np.append 46

np.delete 47

np.ndarray.transpose 48

np.ndarray.reshape 49

np.ndarray.flatten 51

np.concatenate 52

np.vstack 53

np.hstack 54

NumPy Dizilerinde Temel Aritmetik Operasyonlar 55

NumPy Dizilerinde Eleman Bazlı Fonksiyonlar 57

np.exp 57

np.sqrt 58

np.log 58

np.cos 58

np.sin 59

NumPy Dizilerinde Toplama Fonksiyonları 59
np.min 59
np.argmin 60
np.max 60
np.argmax 60
np.mean 61
np.std 61
np.var 61
NumPy Dizilerinde Koşullu Fonksiyonlar 62
np.argwhere 62
np.where 62
np.select 63
Pandas ile Veri İşleme 64
Pandas Veri Yapıları 65
Seriler 65
Veri Çerçeveleri 66
Pandas Veri Çerçeveleri Oluşturma 67
NumPy dizilerinden Pandas Veri Çerçevesi oluşturma 67
Sözlük Yapılarından Pandas Veri Çerçevesi Oluşturma 68
Pandas ile Veri Okuma 69
.txt Dosyasından Veri Okuma 69
Pandas Veri Çerçevelerine Genel Bir Bakış 70
.csv Dosyasından Veri Okuma 70
Pandas Veri Çerçevelerinin Manipülasyonu 76
Pandas ile Veri Yazma/Kaydetme 92

Matplotlib ve Seaborn ile Veriyi Görselleştirerek Keşfetme 94
Kendi Yardımcı Kütüphanemizi Kurmaya Hızlı Bir Giriş 94
1. İskelet kurma 95
2. Tablolu Dairesel Grafik Oluşturma Metodunu Ekleme 104
3. Veri Setindeki Verilerin Tipini Gösteren Grafiğin Metodunu Ekleme 106
4. Veri Setindeki Kayıp Hücre Miktarını Gösteren Grafiğin Metodunu Ekleme 107
5. Veri Setindeki Tekrarlanan Satır Miktarını Gösteren Grafiğinin Metodunu Ekleme 108
6. Veri Setindeki Değişkenlerinin Dağılım Grafiğinin Metodunu Ekleme 109
7. Veri Setindeki Bir Özelliğin Dağılım Ölçülerinin Grafiğinin Metodunu Ekleme 110
8. Veri Setindeki Bir Değişkenin Merkezi Dağılım Grafiğinin Metodunu Ekleme 115
9. Veri Setindeki Sürekli Değişkenlerinin Kovaryans Grafiğinin Metodunu Ekleme 117
10. Veri Setindeki Sürekli Değişkenlerinin Korelasyon Grafiğinin Metodunu Ekleme 119
11. Veri Setini Temel Bileşenler Analizi ile İki Boyutta İnceleme Grafiğinin Metodunu Ekleme 121
12. Veri Setindeki İki Değişkenin İlişkisini Gösteren Grafiğin Metodunu Ekleme 123
ProfillıngReport Sınıfını Kullanarak Örnek Bir Çalışma 124
Neler Öğrendik? 134

BÖLÜM 4: MAKİNE ÖĞRENMESİ 137

Makine Öğrenmesi Nedir? 138
Denetimli Öğrenme (Supervised Learning) 139
Denetimsiz Öğrenme (Unsupervised Learning) 140
Makine Öğrenimi Projesinin Temel Adımları 141
Adım 1: Veri Toplama 141
Adım 2: Veri Hazırlama 142
Adım 3: Model Seçme ve Değerlendirme 143
Adım 4: Modeli Hizmete Sunma 144
Adım 5: Modeli Yeniden Eğitme 144

Detaylı İnceleme: Veri Hazırlama 145
Veri Temizleme 145
Eksik Değer Tedavisi 146
Aykırı Değer Tespiti 148
PyKasif Kütüphanesine DataCleaning Adında Veri Temizleme Bölümü Ekleme 150
DataCleaning Sınıfını Kullanarak Örnek Bir Çalışma 161
Öznitelik Seçimi 172
Yöntemler 172
1. Filtreleme (Filter) Yöntemleri 172
2. Sarmal (Wrapper) Yöntemler 173
3. Gömülü (Embedded) Yöntemler 173
PyKasif Kütüphanesine FeatureSelection Adında Öznitelik Seçme Bölümü Ekleme 174
1. İskelet Kurma 174
FeatureSelection Sınıfını Kullanarak Örnek Bir Çalışma 188
Detaylı İnceleme: Model Seçme ve Değerlendirme 195
Model Seçme 195
2. Karar Ağaçları (Decision Trees) 199
Model Değerlendirme 202
Neler Öğrendik? 205

BÖLÜM 5: PROJELER 207

Uçtan Uca Sınıflandırma Veri Bilimi Projesi: Titanic Sağkalım Tahmini 208
1. Veriyi Yükleme ve Anlama 210
2. Veriyi Temizleme 221
3. Öznitelik Seçme 229
4. Model Kurmadan Önceki Son Adım: Kategorik Değişkenleri Sayısala Dökme 231
5. Model Kurma 233

Uçtan Uca Regresyon Veri Bilimi Projesi: Boston Ev Fiyat Tahmini 240

1. Veriyi Yükleme ve Anlama 242

2. Veriyi Temizleme 251

3. Öznitelik Seçme 257

4. Model Kurma 258

Sonsöz 265

1

Giriş ve Kurulumlar

Bu Bölümde

Veri Bilimi	2
Anaconda ile Proje Ortamı Kurulumu	3
Anaconda Ortamını Tanıma	6
Anaconda Navigator	7
Amazon Web Services Hesabı Oluşturma	12
Neler Öğrendik?	14

Bu bölümde beklenen hedefler:

» Veri Biliminin tanımı ve hedeflerini kavramak,

» Kitapta kullanılacak araçların işletim sistemlerine özgü nasıl kurulmasını ve kullanılması aktarmak,

olacaktır.

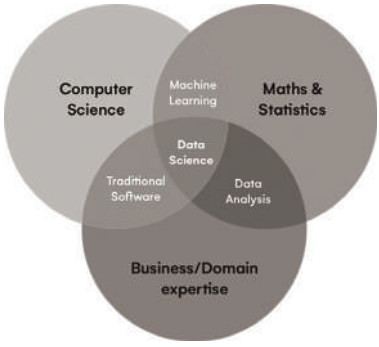
VERİ BİLİMİ

Veri bilimi, büyük miktarda karmaşık veriler ile yararlı bilgiler sağlayan bir uygulamalı matematik ve istatistik alanıdır. Temel olarak matematik, bilgisayar bilimi ve endüstriye özgü uzman bilgisi alanları arasındaki kesişimdir. Veri bilimi bölümünde çalışan kişiler, veri bilimcileri olarak adlandırılır .

Veri biliminin hedefleri:

» Verilerden bilgi üretmek,
» Verilerden eylem önerileri türetmek,
» Karar alma desteği,
» İş süreçlerini optimize etmek ve otomatikleştirmek,
» Ve bunlarla birlikte kurumsal yönetim desteği de yer alır.

Veri bilimcileri, genellikle veri biliminde daha fazla eğitim almış bilgisayar bilimcileri, matematikçiler, istatistikçiler, işletme ekonomistleri, programcılar, veritabanı uzmanları veya fizikçilerdir.



Konuya özel ek olarak, bir veri bilimcisi verilerden üretilen bilgiyi net bir şekilde sunabilmeli ve çeşitli hedef gruplara yaklaştırabilmelidir. Uygun iletişim ve sunum becerileri gereklidir. Kurumsal ortamda veri bilimcisi, iş analisti veya veri analisti terimleri genellikle karıştırılır. Görevleri ve faaliyet alanları zaman zaman çakışabilir.

Veri analisti klasik, pratik veri analizi gerçekleştirirken, veri bilimcisi karmaşık yöntemlerle, örneğin; yapay zeka veya makine öğrenimi ve gelişmiş analiz ve tahmin teknikleri ile bir yaklaşım benimsiyor. Veri bilimcisi, asıl odak noktasının yalnızca iş modellerinin ve iş süreçlerinin analizi olmadığı için kendisini iş analistinden ayırır. İş analistleri, veri bilimcileri tarafından hazırlanan verileri ve kendileri tarafından sağlanan etkileşimli dashboard veya veri araçları analizlerinde kullanırlar.

Günümüzde, endüstrilerdeki şirketler için veri bilimi araçları ve yöntemleri kullanılarak büyük miktarda veri değerlendirilmektedir. Aşağıda örnek senaryolara bakarak daha iyi kavrayabiliriz.

» Perakende ve ticaret şirketleri, müşterilerin satın alma davranışlarını analiz ederek veri biliminden yararlanır. Olası iade nedenlerinin araştırılması, iade sayısının azaltılmasına yardımcı olacaktır.
» Sağlık sektöründe (tıp ve eczacılık), hastaların verileri ile bireyselleştirilmiş tedavi ve ilaç optimizasyonu için analizler oluşturulmasını sağlar.
» Lojistik şirketleri, iş süreçlerini ve nakliye hizmetlerinin kalitesini iyileştirmek için veri bilimini kullanır.
» Endüstriyel şirketler, veri bilimini kullanarak üretim süreçlerini kontrol eder ve optimize eder.

Anaconda ile Proje Ortamı Kurulumu

Anaconda, bilimsel Python paketleri, araçları, kaynakları ve geliştirme ortamlarından oluşan harika bir koleksiyondur. Bu paket, bir veri bilimcinin Python'ın inanılmaz gücünden yararlanmak için kullanabileceği birçok temel aracı içerir. Anaconda Individual Edition ücretsiz ve açık kaynaklıdır. Ve bizde bu ürünü kullanacağız. Ücretsiz ve açık kaynaklı olması, Anaconda ile çalışmayı erişilebilir ve kolay hale getirir. Bu ürün Windows, macOS ve Linux makinelerde çalışabilmektedir.

235 bölgede 20 milyondan fazla kullanıcı ve 2,4 Milyardan fazla paket indirme ile; Anaconda, son derece büyük bir topluluk oluşturdu. Anaconda, çeşitli bilimsel makine ve veri bilimi paketlerine erişmeyi kolaylaştırır.

Veri biliminde yeniyseniz ve tam anlamıyla Python deneyimini yaşamak istiyorsanız veya daha fazla işlevsellik ve verimlilik arayan deneyimli bir veri bilimcisiyseniz bu harika dağıtımı incelemenizi gerçekten tavsiye ederim. Ve kitapta biz de bu ekosistemden faydalanacağız. Anaconda, Paket yönetimini ve dağıtımını hızlı ve kolay hale getirir. Araçlar, geliştirme ortamları, paketler ve kitaplıklarla dolu Anaconda, veri bilimi için gerçekten orjinal bir seçimdir. Anaconda'nın bu tür gelişmiş özelliklerinin olduğu pek çok sektörde ve alanda popülaritesi giderek artarken bu sürekli genişleyen araç ve kaynak paketiyle başlamak için daha iyi bir zaman olmamıştır.

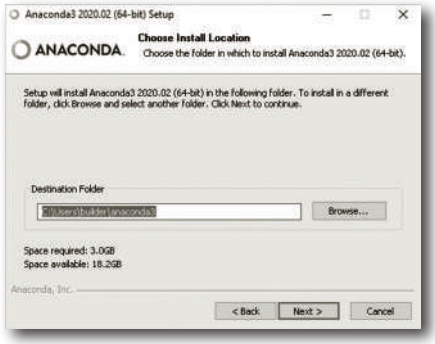
Biz, yukarıda da bahsedildiği gibi **Anaconda Individual Edition** ürününü kullanacağız. Kurulum için aşağıdaki işlemleri takip ediniz eğer güncellik veya diğer konular ile alakalı bir sorun yaşarsanız ***https://docs.anaconda.com/*** adresini ziyaret ediniz.

Kurulum için; ***www.anaconda.com/products/individual*** adlı siteyi ziyaret edin.

Aşağıda görüldüğü üzere işletim sisteminize göre ilgi bağlantıya tıklayın. İnenen dosyaya iki kere tıkladıktan sonra ilerleyen sayfalarda işletim sisteminize göre olan adımları takip edin.

WINDOWS İŞLETİM SİSTEMİ İÇİN KURULUM

Anaconda'yı kurmak için bir hedef klasör seçin ve **Next** butonuna tıklayın.



2

İş Problemini Anlamak

Bu Bölümde

Sorunu Çerçevelendirme 18
İş Problemlerini Tanımlama 19
Veri Bilimi Dışı Çözümlerin Belirlenmesi 19
Çıktı Kullanımının Tanımlanması 20
Yanlış Sonuçların Yönetilmesi 21
Veri Kaynaklarını Tanımlama 21
Verileri İşlemek için Kütüphanelerin Belirlenmesi 21
Ürünün Nereden Hizmet Vereceği Yeri Belirleme 22
Neler Öğrendik? 23

Bu bölümde, veri bilimi iş sorununu nasıl ele alınmalı ve bu sürecin nasıl yürütülmesi üzerine bilgi verilmiştir.

Bu süreç, sorunu çerçevelemekten başlayıp, sorundan hareketle oluşturulan modelin nerede nasıl hizmet verme senaryosuna kadar bu bölümde değinilmiştir.

SORUNU ÇERÇEVELENDİRME

Her bir sorunun durumu farklıdır ve kendisine özel birçok özellik barındırır. Her türlü soruna uyan tek bir yaklaşım yoktur. Bir iş problemini, istenen sonucu ve durumun güçlü ve zayıf yönlerini anlama yeteneği bu işin kilit noktasıdır. Esasında, madalyonun her iki tarafını, beklentileri ve yetenekleri anlamanız gerekir. Bu bilgileri göz önüne alarak daha sonra çözümü çerçevelemeye devam edebilirsiniz.

Bu adımlar günümüzde iş geliştirmede sıklıkla kullanılır. Ve bu adımlar sadece veri biliminde değil günlük hayatta da birçok yerde kullanılır.

Yukarıda bahsettiğim olay **SWOT** (Strengths, Weaknesses, Opportunities, Threats) Türkçe karşılığı; Güçlü Yönler, Zayıf Yönler, Fırsatlar, Tehditle Analiz olarak yorumlanabilir.

SWOT analizi, bir organizasyonun, tekniğin, sürecin, durumun bir proje veya iş girişimindeki kişinin güçlü ve zayıf yönlerini belirlemek iç ve dış ortamdan gelen fırsatları, tehditleri belirlemek için kullanılan stratejik bir tekniktir.

Bu teknik, projenin veya iş girişiminin hedeflerinin belirlenmesini ve hedefe ulaşmak için olumlu/olumsuz iç ve dış faktörlerin tanımlanmasını gerektirir.

Bu yöntem 1960'larda Harvard Üniversitesi, Learned, Christensen, Andrews ve Guth profesörleri tarafından geliştirilmiştir.

Veri bilimi projesinde SWOT analizi kullandığımızda yararlı olabilir. Başlıca yararları:

» Fırsatlardan yararlanmak için gücümüzü kullanabiliriz.
» Zayıf yönlerimizin farkında olarak, onları güçlü yönlere dönüştürmek için stratejiler uygulayabiliriz.
» Çevremizdeki tehditleri güçlü yönlerimize entegre ederek, fırsatlara dönüştürebiliriz.

İş Problemlerini Tanımlama

Bir amaca hizmet etmek için bir organizasyon kurulur. Bu organizasyonun iş birimleri, ortak bir istenen sonucu elde etmek için bağımsız olarak planlanmalı ancak ilgili faaliyetleri odakta tutacak şekilde olmalıdır. İş probleminin ölçülebilir bir şekilde tanımlanması gerekir.

Ölçülebilir metrikler, projenin erken bir aşamasında tanımlanmalıdır. Bu, ekibin ulaşılabilir hedefe odaklanmasına yardımcı olur.

Veri Bilimi Dışı Çözümlerin Belirlenmesi

Tüm iş sorunları bir veri bilimi çözümüne ihtiyaç duymaz. Başlıca aşağıdaki noktalar göz önüne alınmalı:

» Eğitim Verilerin Yetersiz Miktarı

Veri bilimi çözümlerini ilk seferde kullanılmamasının nedeni, doğru modellerin eğitimini engelleyen yetersiz veri miktarıdır.

Daha da kötüsü, bazı durumlarda yanlış modeller tamamen rastgele tahmin veya sınıflandırma sonuçları üretir ve tüm iş işlevselliği sorgulanabilir.

» Önemli Veri Noktalarının Eksikliği

Büyük bir veri kümesine sahip olmak, veri bilimini bir çözüm olarak kullanmak için bir sonraki husus, modele girdi olarak kullanılacak kadar önemli olup olmadıklarına karar vermek için veri noktalarını (öznitelikler) keşfetmektir. Özellik seçimi, doğruluk ve güvenilirlik açısından modelin genel performansını doğrudan etkilediği için çok önemli bir parçadır.

» Veri Bilimi Modelini Güncellemenin Pratik Olmaması

Veri bilimi çözümlerini kullanmanın önündeki üçüncü olası engel, sık zaman aralıklarında bir güncelleme modelinin uygulanmasının zor olması olabilir. Başlangıçta doğru bir modele sahip olmak, kısa bir süre için güvenilir tahminler elde etmek için yeterli olabilir. Ancak daha uzun süreler boyunca performans, kullanıcıların olaylarındaki değişiklikler nedeniyle olumsuz etkilenecektir.

Tüm bunlar, güncel verileri kullanarak kendini sürekli olarak yeniden eğiten planlanmış bir güncelleme modelinin yerleştirilmesini gerektirir. Sorun, güncel verilere erişimin her zaman mümkün olmamasıdır. Çünkü, kaynak bir veya daha fazla farklı kaynaktan elde edilen geçmiş veriler olabilir.

» Yorumlanabilirlik Eksikliği

Bazen, son derece doğru eğitilmiş bir modelinin sonuçları bile son kullanıcılar tarafından kolayca anlaşılamayabilir ve bu da benimsenmesini çok zor veya imkansız hale getirir. Örneğin; karar ağaçları söz konusu olduğunda ortaya çıkan alt gruplar rahatlıkla yorumlanması zordur.

» Canlı Ortamda Yavaşlık

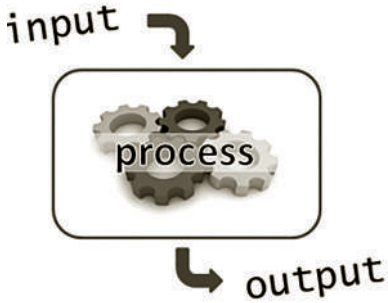
Geleneksel kural tabanlı yaklaşımların aksine veri bilimi tabanlı yaklaşımların uygulanmasında her zaman birkaç ekstra aşama vardır: veri ön işleme ve/veya temizleme, özellik seçimi ve/veya dönüştürme, model seçimi ve tasarımı, eğitim ve test, vb. Bütün bunlar genel yürütme süresine eklenir ve derin öğrenme modelleri söz konusu olduğunda tasarımlarındaki milyon ölçekli parametre sayısı nedeniyle daha da kötüleşir.

Bazı durumlarda, tahmin zamanında birkaç saniyelik bir gecikme bile son kullanıcılar için çok istenmeyen bir durumdur ve bu da onları uygulamanın kullanımından hemen vazgeçirebilir. Örneğin, gerçek zamanlı ürün/hizmet öneri sistemlerinin çoğu durumunda tüm kullanıcılar hızlı bir yanıt süresi bekler ve bu onları uygulamayı kullanmaya motive etmenin en önemli faktörlerinden biri olabilir.

Sonuç olarak, veri bilimi çözümleri gelecek vaat eden performanslarıyla işin tüm yönlerinde devrim yarattı. İş sorunlarını ele almak için veri bilimi tabanlı çözümleri kullanmayı önermeden önce olası tüm zorlu senaryoları göz önünde bulundurmak önemlidir. Çünkü, erken değerlendirme kuruluşların önemli miktarda zaman ve kaynaklarından tasarruf etmesini sağlayacaktır. Yukarıdaki sorunları çözüme ulaştırdığınız zaman elinizde çok sağlam bir veri bilimi çalışması mevcut olur. Özetle mümkün olduğu kadar basit ve açıklanabilir bir yöntem ile sonuç elde edilmesi önerilir.

Çıktı Kullanımının Tanımlanması

Projenizin çıktısının mutlaka son ürün olması gerekmeyebilir. Büyük resmin ve çabanızın bütüne nasıl uyduğunun farkında olmalısınız. Örneğin, çıktınız müşteri kaybı için bir notu verebilir ve bu daha sonra başvuru sahibi hakkındaki diğer özelliklerle birlikte bir karar verme sürecinde kullanılabilir.



Yanlış Sonuçların Yönetilmesi

Başarısızlık yönetimi, proje yönetiminin önemli bir bileşenidir. Bunun için çıkan sonuçların iyi yorumlanması gerekir ve analizler ile hatalı bölgeler tespit edilmelidir. Çünkü, modelin nerede eksik olduğunu anladığınız vakit bu süreci kolaylıkla yönetebilirsiniz.

Veri Kaynaklarını Tanımlama

Bir veri bilimi çözümünün temeli büyük veya küçük veridir. Verilerin gerçekliği, izlenebilirliği ve kalitesi çözümünüze olan güveni doğrudan etkileyecektir. Bu nedenle, bir veri kaynağı belirttiğinizde, kaynağını bilmeniz ve gelecekte tanımlanmış açık bir plana sahip olmanız önemlidir. Verilerinizle ilgili herhangi bir kalite sorunu çözümünüzün güvenilirliğine ciddi şekilde zarar verecektir.

Verileri İşlemek için Kütüphanelerin Belirlenmesi

Veri bilimi projesine başlanırken günümüzde hangi kütüphanelerin kullanılacağı neredeyse artık sabit, önerilen bir yol vardır. Veri biliminde kütüphane seçimi projelerin aşamalarına bağlıdır.

Bölümlerden bir tanesi Veri İşleme ve Modelleme, bunun için kitabımızda NumPy, Pandas ve SciKit-Learn kütüphanelerini kullanacağız. Diğer bölüm ise Veri Görselleştirme bu aşama için ise Matplotlib ve Seaborn kütüphaneleri kullanacağız.

3

Kullanılacak Kütüphaneleri ve Teknolojilerini Tanımak

Bu Bölümde

NumPy ile Matematiksel Hesaplama 26
NumPy Veri Tipleri 28
NumPy Dizileri Oluşturma 32
NumPy Dizilerini İndeksleme ve Dilimleme 43
NumPy Dizilerinin Manipülasyonu 46
NumPy Dizilerinde Temel Aritmetik Operasyonlar 55
NumPy Dizilerinde Eleman Bazlı Fonksiyonlar 57
NumPy Dizilerinde Toplama Fonksiyonları 59
NumPy Dizilerinde Koşullu Fonksiyonlar 62
Pandas ile Veri İşleme 64
Pandas Veri Yapıları 65
Pandas Veri Çerçeveleri Oluşturma 67
Pandas ile Veri Okuma 69
Pandas Veri Çerçevelerine Genel Bir Bakış 70
Pandas Veri Çerçevelerinin Manipülasyonu 76
Pandas ile Veri Yazma/Kaydetme 92
Matplotlib ve Seaborn ile Veriyi Görselleştirerek Keşfetme 94
ProfillıngReport Sınıfını Kullanarak Örnek Bir Çalışma 124
Neler Öğrendik? 134

Bu bölümde, veri bilimi projelerinde en sık kullanılan ve sorunlarımızın önemli bir kısmını çözen NumPy, Pandas, Matplotlib ve Seaborn kütüphanelerine değineceğiz.

NumPy kütüphanesi ile matematiksel hesaplama üzerine yoğunlaşılacaktır.

Pandas kütüphanesi ile temelde veri kümesi manipülasyonu konusu ile uğraşacağız.

Matplotlib ve Seaborn kütüphanesi ile de veriyi görselleştirerek keşfetme yeteneği kazanacağız.

Ve bu bölüm itibariyle PyKasif adında kendi yardımcı kütüphanemizi kurmaya adım atacağız. Kodlama ve programlama yönünden yoğun bir bölüme başlıyoruz.

NumPy ile Matematiksel Hesaplama

Bu bölümde NumPy kütüphanesini inceleyeceğiz.

NumPy, Python ile kolay bir şekilde sayısal hesaplamayı amaçlayan açık kaynaklı bir projedir. NumPy her zaman %100 açık kaynaklı yazılım ve herkesin kullanımına açık olacağını kendi web sitesinde belirtmiştir.

NumPy çok boyutlu bir dizi ve matris veri yapıları içerir. Trigonometrik, istatistiksel ve cebirsel rutinler gibi diziler üzerinde bir dizi matematiksel işlem gerçekleştirmek için kullanılabilir. Bu nedenle, kütüphane çok sayıda matematiksel, cebirsel ve dönüşüm fonksiyonları içerir.

NumPy kütüphanesini kullanmak için programımıza çağırmamız gerekiyor. Bunun için `import` anahtar kelimesini kullanarak çağırdıktan sonra `as` anahtar kelimesi ile `np` takma adı altında kullanacağız. Aşağıda kullanımı gösterilmektedir:

Örnek Kod

```
In [1]: import numpy as np
```

Ve artık NumPy kütüphanesinin fonksiyonlarına ve sınıflarına erişebileceksiniz. NumPy sürümünü ve yapılandırmasını ekrana yazdıralım.

Örnek Kod

```
In [2]: np.__version__
```

Ekran Çıktısı

```
Out[2]: '1.19.5'
```

Görüldüğü üzere NumPy kütüphanesinin `1.19.5` versiyonu yüklü. Bu şuan en güncel sürümdür.

> **NOT** Siz kitabı aldığınızda belki NumPy kütüphanesinin daha güncel bir versiyonu çıkmış olabilir, siz onu görebilirsiniz. Fakat kitapta anlatılacak konular ile uyumlu olacaktır.

Şimdi bir `ndarray` sınıfından bir dizi oluşturalım.

Örnek Kod

```
In [3]: veri = np.array([ [1, 2,3], [4,5, 6] ])
```

Oluşan diziyi ekrana bastırma yapalım.

Ekran Çıktısı

```
In [3]: veri
```

Ekran Çıktısı

```
Out[3]: array([[1, 2, 3],
               [4, 5, 6]])
```

Artık elimizde iki boyutlu bir dizi vardır. Şimdi bu nesnenin özelliklerine sırasıyla bakalım:

Oluşturduğumuz nesnenin tipi öğrenmek için `type()` fonksiyonunu kullanalım.

Örnek Kod

```
In [4]: type(veri)
```

Ekran Çıktısı

```
Out[4]: numpy.ndarray
```

Nesnenin kaç boyutlu olduğuna bakmak için `ndim` özelliğini çağıralım.

Örnek Kod

```
In [5]: veri.ndim
```

Ekran Çıktısı

```
Out[5]: 2
```

Bu çıktı 2 boyutlu bir nesne olduğunu belirtir.

Nesnenin boyutlarının uzunluğuna bakmak için `shape` özelliğini çağıralım.

Örnek Kod

```
In [6]: veri.shape
```

Ekran Çıktısı

```
Out[6]: (2, 3)
```

Bu çıktı nesnede 2 satır 3 sütun olduğunu belirtir.

Nesnenin içerdiği elementlerin tipini öğrenmek için `dtype` özelliğini çağıralım.

Örnek Kod

```
In [7]: veri.dtype
```

Ekran Çıktısı

```
Out[7]: dtype('int64')
```

Buraya kadar beraber bir nesne oluşturduk ve belli başlı konulara değindik.

Şimdi biraz daha derine inerek bir sonraki başlığımızda NumPy dizisinin veri tiplerine değinelim.

NumPy Veri Tipleri

Bir NumPy dizisi homojen olmak zorundadır. Homojen demek dizi içindeki elementlerin veri tiplerinin aynı olması durumudur. Bu veri tipleri tamsayılar, işaretsiz tamsayılar, mantıksal ifadeler, ondalıklı sayılar ve karmaşık sayılardır. Şimdi bu veri tiplerini tanıyalım.

Tamsayılar

NumPy kütüphanesinde tam sayı verileri bu tipte tutarak kulllanabiliriz. En sık kullanılan tamsayı formları int8, int16, int32 ve int64'dür. Bu bahsedilen formlar verinizin değeri büyüdükçe kullanılır. Aşağıdaki örnekte `np.array` kullanarak örnek bir tamsayı verisi ürettik.

Bir `np.array` fonksiyonu ile 3 elemanlı dizi oluşturuldu.

Örnek Kod

```
In [1]: a=np.array([100, 200, 300], dtype=np.int)
```

a adlı değişkenini görüntüleyelim.

Örnek Kod

```
In [2]: a
```

Ekran Çıktısı

```
Out[2]: array([100, 200, 300])
```

a adlı değişkenin veri tipine bakalım.

Örnek Kod

```
In [3]: a.dtype
```

Ekran Çıktısı

```
Out[3: dtype('int64')
```

İşaretsiz Tamsayılar

Eğer veri setinizde negatif tamsayı içeren bir veri yoksa bu formatı kullanarak bilgisayardaki belleği daha etkili kullanabilirsiniz. En sık kullanılan işaretsiz tamsayı formları **uint8**, **uint16**, **uint32**, **uint64**. Bu formatı çok sıklıkla kullanmayacaksınız çünkü günlük hayatta verilerimiz için uygun olmayabilir özellikle görselleştirme için.

Mantıksal İfadeler

Eğer veri setiniz mantıksal ifadeler içeriyorsa yani **True** veya **False** ifadeleri varsa bu formatı kullanarak bilgisayarınızın belleğini daha etkili kullanabilirsiniz.

Tamsayı değerler ile `np.array` fonksiyonun `dtype` parametresinde belirtmiş olduğumuz `np.bool` ile mantıksal diziyi oluşturalım. Bu diziyi oluştururken -1, 0 ve 1 değerlerini kullanalım. Çünkü sizlere bu sayısal değerlerden nasıl bir mantıksal dizi olabileceğini göstermek istiyorum. Bu dönüşüm için de `dtype` parametresine `np.bool` ifadesini aşağıda belirtmek zorundayız.

Örnek Kod

```
In [1]: a=np.array([-1,0,1], dtype=np.bool)
```

Şimdi `a` dizisini ekrana basalım. Dizi incelendiğinde True ve False değerleri içeriyor. Bu mantıksal atamanın oluşma şekli `:` `0` rakamın karşılığına False verilirken, diğer rakamların karşılığı True verilerek oluşturulur.

Örnek Kod

```
In [2]: a
```

Ekran Çıktısı

```
Out[2]: array([ True, False, True])
```

Ve `a` adlı mantıksal değerleri içeren dizinin veri tipine bakalım. (In [1]'de belirtildiği gibi sonuç elde ediyoruz)

Örnek Kod

```
In [3]: a.dtype
```

Ekran Çıktısı

```
Out[3]: dtype('bool')
```

NumPy Dizilerini İndeksleme ve Dilimleme

NumPy dizilerinin öğelerine ve alt dizilerine, Python listelerinde de kullanılan standart köşeli ayraç gösterimi kullanılarak erişilir.

Bu konuyu anlamak için 2 temel örnek üzerinden ileriyeceğiz.

1. Tek boyutlu dizilerde indeksleme ve dilimle operasyonları

Burada `np.linspace` fonksiyonunu kullanarak ürettiğimiz veri üzerinden uygulamalı inceleyeceğiz.

Örnek olarak, 9 değeriyle başlayan 99 değeri aralığında 10 sayı oluşturalım.

Örnek Kod

```
In [1]: veri = np.linspace(start=9,stop=99,num=10)
```

Oluşan diziyi inceleyelim.

Örnek Kod

```
In [2]: veri
```

Ekran Çıktısı

```
Out[2]: array([ 9., 19., 29., 39., 49., 59., 69., 79., 89., 99.])
```

Şimdi verimizin ilk elemanına erişelim.

Örnek Kod

```
In [3]: veri[0]
```

Ekran Çıktısı

```
Out[3]: 9.0
```

Verimizin son elemanına erişelim.

Örnek Kod

```
In [4]: veri[-1]
```

Ekran Çıktısı

```
Out[4]: 99.0
```

Verimizin ikinici indexteki elemanına yani üçüncü elemanına erişelim.

Örnek Kod

```
In [5]: veri[2]
```

Ekran Çıktısı

```
Out[5]: 29.0
```

Verimizdeki ikinci indeks ile dördüncü indeks aralığında bulunan değerlerine erişelim. Demek istenilen ikinci, üçüncü ve dördüncü indeksteki değerler nelerdir.

Örnek Kod

```
In [6]: veri[2:5]
```

Ekran Çıktısı

```
Out[6]: array([29., 39., 49.])
```

Verimizde belli bir element aralığını seçmek için `[ başla:bitir:adım_uzunluğu ]` yapıyı kullanmamız gerekir, örneğin sıfırıncı indeksten başlayarak iki adım ile yedinci indekse kadar olan değerlere erişim için `[0:7:2]` yapısı kullanılır.

Örnek Kod

```
In [7]: veri[0:7:2]
```

Ekran Çıktısı

```
Out[7]: array([ 9., 29., 49., 69.])
```

2. Çok boyutlu dizilerde indeksleme ve dilimle operasyonları

Burada `np.array` fonksiyonunu kullanarak ürettiğimiz veri üzerinden uygulamalı inceleyeceğiz. Örnek olarak, İlk başta 1'den 12'e kadar değerleri içeren 3 satır 4 sütunlu bir dizi oluşturalım.

Örnek Kod

```
In [1]: veri = np.array([[1, 2,3,4], [5,6,7,8],[9,10,11,12]])
```

Diziye bir bakalım.

4

Makine Öğrenmesi

Bu Bölümde

Makine Öğrenmesi Nedir? 138
Makine Öğrenimi Projesinin Temel Adımları 141
Detaylı İnceleme: Veri Hazırlama 145
Öznitelik Seçimi 172
Detaylı İnceleme: Model Seçme ve Değerlendirme 195
Neler Öğrendik? 205

Bu bölümde, makine öğrenmesi kavramının ne olduğunu, hangi tip öğrenme şekillerinin olduğunu ve hangi öğrenme tipinin hangi projelerde nasıl kullanılmasına dair bilgiler içeriyor iken, aynı zamanda bir makine öğrenmesinin hangi temeller üzerine kurulduğunu ve bu temellerin detayları yer almaktadır.

MAKİNE ÖĞRENMESİ NEDİR?

Makine öğrenimi, bilgisayarların programlanmadan çalışmasını sağlamaya yarayan önemli bir alandır. Yapay zekanın bir dalı olan makine öğrenmesi, sistemlerdeki veri kalıplarını tanımlamasını, kararlar almasını ve gelecekteki sonuçları tahmin etmesini sağlayabilir. Aslında, makine öğrenimi, bilgisayarlara insanlar gibi düşünmeyi, öğrenmeyi ve davranmayı öğretme girişimidir. Artan internet hızları, depolama teknolojisindeki gelişmeler ve bilgi işlem gücü sayesinde, makine öğrenimi katlanarak gelişti ve neredeyse her endüstrinin ayrılmaz bir parçası haline geldi.

Makine öğrenmesinin geçmişinde yer alan bazı önemli anlar aşağıdaki gibidir;

» **1920'ler öncesi:** Thomas Bayes, Andrey Markov, Adrien-Marie Legendre ve diğer ünlü matematikçiler temel makine öğrenimi teknikleri için gerekli zemini hazırladı.

» **1943'lü yıllarda:** Sinir ağlarının ilk matematiksel modeli Walter Pitts ve Warren McCulloch tarafından bilimsel bir makalede sunuldu.

» **1950 yılında:** Alan Turing, yapay zekayı tanımlamaya çalışır ve makinelerin öğrenme yeteneklerine sahip olup olmadığını sorgular.

» **1951 yılında:** Marvin Minsky ve Dean Edmonds, ilk yapay sinir ağını kurdu.

» **1965:** Alexey (Oleksii) Ivakhnenko ve Valentin Lapa ilk çok katmanlı algılayıcıyı geliştirdi. Ivakhnenko genellikle derin öğrenmenin babası olarak kabul edilir. *(Derin öğrenme, makine öğreniminin bir alt kümesi)*

» **1997:** IBM satranç bilgisayarı Deep Blue, dünya satranç şampiyonu Garry Kasparov'u yendi.

» **2009:** Görsel nesne tanıma araştırmaları için yaygın olarak kullanılan büyük bir görüntü veritabanı olan ImageNet, Fei-Fei Li tarafından piyasaya sürüldü.

» **2014:** Lan Goodfellow ve meslektaşları, Çekişmeli Üretici Ağı geliştirdi. Aynı yıl, Facebook DeepFace'i geliştirdi. *(DeepFace, görüntülerdeki insan yüzlerini yaklaşık %97,25 doğrulukla tespit edebilen derin öğrenmeli bir yüz tanıma sistemidir)*

» **2015:** AlphaGo, Go'da profesyonel bir oyuncuyu yenen ilk yapay zeka oldu.

» **2020:** OpenAI, insan benzeri metin oluşturma yeteneğine sahip güçlü bir doğal dil işleme algoritması olan GPT-3'ü duyurdu.

Makine öğrenimi yöntemlerinin veri bilimi uygulamasında veri modelleri oluşturmanın bir aracı olarak düşünülebilinir. Bu araçları etkili kullanmak için öncelikle sorunu çok iyi anlamak gerekir. Bundan dolayı ilk olarak yaklaşım türlerine kategori bazlı bakalım.

Makine öğrenmesi iki temel alt kategoriye sahiptir. Bu başlıklardan biri **Denetimli Öğrenme** (Supervised Learning), diğeri ise **Denetimsiz Öğrenme** (Unsupervised learning) denilebilinir.

Denetimli Öğrenme (Supervised Learning)

Denetimli öğrenme, bir veri bilimcinin öğretmen gibi davrandığı temel kuralları ve etiketli veri kümelerini besleyerek yapay zeka sistemini eğittiği bir makine öğrenimi yaklaşımıdır. Veri kümeleri, etiketli girdi verilerini ve beklenen çıktı sonuçlarını içerecektir. Bu makine öğrenimi yönteminde sisteme giriş verilerinde nelere bakması gerektiği açıkça söylenir.

Daha basit bir ifadeyle, denetimli öğrenme algoritmaları örnek olarak öğrenir. Bu tür örnekler topluca eğitim verileri olarak adlandırılır. Bir makine öğrenimi modeli, eğitim veri kümesi kullanılarak eğitildikten sonra modelin doğruluğunu belirlemek için ona test verileri verilir.

Denetimli öğrenme ayrıca iki türe ayrılabilir: Sınıflandırma ve Regresyon.

1. Sınıflandırma

Etiketlenmemiş örneklere etiket atamak için sınıflandırma algoritmaları kullanılır. Özelliklerle birlikte kullanılacak etiketleri içeren eğitim verilerinden öğrenerek çalışırlar ve ardından yeni bir örneğin hangi sınıfa girmesi gerektiğini tahmin etmek için verilerde buldukları kalıpları kullanırlar.

Örneğin, bir evin belirli bir fiyattan daha fazla satıp satmayacağı veya bir e-postanın spam olup olmadığını tahmin etmek olabilir.

Sınıflandırma konusuna bir kaç örnek vermek gerekirse;

» Bir evin belirli bir fiyattan daha fazla satıp satmayacağının sınıflandırması,
» Bir e-postanın spam olup olmadığının sınıflandırması,
» Bir fotoğrafta hayvan içerip içermemesinin sınıflandırılması gibi düşünebiliriz.

2. Regresyon

Makine öğreniminde regresyon, bir dizi özelliğe dayalı olarak sürekli bir çıktıyı tahmin etmek için bir algoritmayı eğittiğiniz yerdir.

Regresyon konusuna örnekler olarak:

» Bölgedeki okulların kalitesi, evdeki yatak odası sayısı ve evin konumu gibi verilere dayalı olarak ev fiyatlarının tahmin edilmesi.
» Bir şirketin önceki satışlarının verilerine dayanarak şirketin satış gelirini tahmin etmek.
» Bir müşterinin, önceki müşterilerden gelen verilere dayanarak bir şirkete ne kadara mal olacağı gibi örnekleri düşünebiliriz.

5

Projeler

Bu Bölümde

Uçtan Uca Sınıflandırma Veri Bilimi Projesi: Titanic Sağkalım Tahmini 208

Uçtan Uca Regresyon Veri Bilimi Projesi: Boston Ev Fiyat Tahmini 240

Sonsöz 265

Öncelikle sizleri tebrik etmek isterim arkadaşlar.

Artık bu bölümde veri biliminde önemli olan NumPy, Pandas, Seaborn, Matplotlib, Scikit-Learn ve SciPy kütüphaneleri ile veri setlerini birleştirerek bir çözüm üreteceksiniz.

Burada çözeceğimiz problemlerin türü sınıflandırma ve regresyondur. Titanic Veri Seti üzerinde sınıflandırma ile sağkalım projesi yapılırken, Boston ev Fiyatı Veri Seti üzerinde de regresyon ile ev fiyat tahmini projesi yapılacaktır.

Siz bu projeleri yaptığınızda kafanızda herşey şekillenecektir diye umuyorum.

Hadi başlayalım artık... :)

UÇTAN UCA SINIFLANDIRMA VERİ BİLİMİ PROJESİ: TİTANİC SAĞKALIM TAHMİNİ

Bu bölümde Titanic veri setinde sınıflandırma projesi yapılacaktır. Titanic gemisi, tarihin en kötü kazalarından biridir. 15 Nisan 1912'de Titanic ilk seferi sırasında bir buzdağıyla çarpıştıktan sonra battı ve 2224 yolcu ve mürettebattan 1502'si öldü. Bu sansasyonel trajedi uluslararası toplumu şok etti ve gemiler için daha iyi güvenlik düzenlemelerine yol açtı. Hayatta kalmak için bir miktar şans unsuru olsa da, bazı insan gruplarının hayatta kalma olasılığı diğerlerinden daha yüksekmiş gibi görünüyor.

Biz ise bu gemide yer alan kişilerin bilgilerini içeren veri setini kullanarak kimlerin hayatta kaldığını tahmin etmeye çalışan bir **Makine Öğrenmesi** modeli kuracağız.

Projemiz temelde 5 adımda oluşuyor. Adımlar:

1. Veriyi Yükleme ve Anlama,
2. Veriyi Temizleme,
3. Öznitelik Seçme,
4. Model Kurmadan Önceki Son Adım: Kategorik Değişkenleri Sayısala Dökme,
5. Model Kurma,

Bu projeyi de Jupyter Lab aracında Jupyter Notebook açarak devam edeceğiz. Bu proje dosyasında lütfen diğer Jupyter Notebook dosyalarının yanına koyarak çalıştırınız. Burada yer alan kodları ***https://github.com/muhendis/pykasif*** adresinde bulabilirsiniz. (Bu arada pykasif kod deposuna bir yıldız bırakırsanız, sevinirim :-))

Eğer kodlara ulaşamazsanız, **enginbozaba@gmail.com** adresine mail atabilirsiniz. Proje dosyamızın ismi **titanic.ipynb**'dir.

Veriyi yüklemeden önce projemizde kullanılacak olan kütüphaneleri projemize dahil edelim. Aşağıda görüldüğü üzere ilk başta kendi yazdığımız yardımcı kütüphanelerimizi (ProfillingReport, DataCleaning,FeatureSelection) dahil ettik. Ardından kitabımızda en sık kullandığımız kütüphaneler olan NumPy, Pandas, Matplotlib ve Seaborn kütüphanelerini dahil ettik. Hemen sonra ise kitapta yer yer kullandığımız sklearn (Scikit-learn) kütüphanesinin çeşitli sınıflarını dahil ediyoruz. Çünkü bu bölümde makine öğrenmesi modelerini kurarken ve diğer eğitim için gerekli olan işlemler için dahil ettik. Biraz detaya inersek:

» `from sklearn.linear_model import LogisticRegression`, Lojistik Regresyon modeli kurulması için dahil edildi.
» `from sklearn.svm import SVC`, Destek Vektör Makineleri modeli kurulması için dahil edildi.
» `from sklearn.ensemble import RandomForestClassifier`, Rassal Orman modeli kurulması için dahil edildi.

» `from sklearn.neighbors import KNeighborsClassifier`, K-En Yakın Komşular modeli kurulması için dahil edildi.

» `from sklearn.tree import DecisionTreeClassifier`, Karar Ağacı modeli kurulması için dahil edildi.

» `from sklearn.model_selection import train_test_split`, veri seti eğitim ve test diye ayırması için dahil edildi.

» `from sklearn.model_selection import StratifiedKFold`, modelin doğruluğu ve modelin optimizasyonu için dahil edildi. Bu konuyu ilerleye sayfalarda tekrar geleceğiz. Orada detaylı olarak işlenecektir.

» `from sklearn.model_selection import GridSearchCV`, modellerin en uygun parametre değerlerini bulmaya yarayan GridSearchCV aracı için dahil edildi.

» `from sklearn.metrics import accuracy_score`, modelli değerlendirme kullanacağımız doğruluk metriği için dahil edildi.

Ve en son olarakta proje dosyasında çıkan uyarı bastırmak için **warnings** kütüphanesi dahil edildi.

(pykasif/titanic.ipynb)

```
1.  # kendi yazdığımız yardımcı kütüphanemiz
    from assistant.eda import ProfillingReport
    from assistant.datacleaning import DataCleaning
    from assistant.featureselection import FeatureSelection

    # temel kütüphaneler
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns

    # eğitim için
    from sklearn.linear_model import LogisticRegression
    from sklearn.svm import SVC
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.model_selection import StratifiedKFold
    from sklearn.model_selection import GridSearchCV
```

```
from sklearn.metrics import accuracy_score

# proje dosyasındaki uyarılar için
import warnings
warnings.simplefilter(action='ignore')
```

1. VERİYİ YÜKLEME VE ANLAMA

Bütün gerekli kütüphaneleri yüklediğimize göre veri setini yükleyip hikayesini anlamaya çalışalım. İlk olarak veri setini seaborn kütüphanesinden `load_dataset()` fonksiyonu ile yükleyelim. Ve akabinde 8. kod bloğundan sonra bu veri setinin hikayesine göz atalım.

(pykasif/titanic.ipynb)

```
2. df = sns.load_dataset("titanic")
```

Veri çerçevesine genel olarak bakmak istediğimizde df değişkenini çağırarak görüntüleyebiliriz.

(pykasif/titanic.ipynb)

```
3. df
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

Yukarıdaki veri setinde her bir satır bir yolcuya ait bilgileri içeriyor. Bu bilgiler kişinin ekonomik statüsden cinsiyetine kadar bilgileri içeriyor. Veri setinde yer alan özniteliklerin ne anlam içeriklerine bakalım.

» **survived:** Titanic gemisinde hayatta kalanlar 1, hayatta kalmayanlar 0

» **pclass:** Ekonomi statüsüdür

->1 = Üst

-> 2 = Orta

-> 3 = Düşük

» **sex:** Male/Female

» **age:** Yolcuların Yaşları

» **sibsp:** Gemideki kardeş ve eş sayısını verir

» **parch:** Gemideki ebeveyn ve çocuk sayısını söyler

» **fare:** Bilet fiyatı

» **embark:** Liman isimleri (C, Q, S)

-> C = Cherbourg,

-> Q = Queenstown,

-> S = Southampton

» **class:** Ekonomi statüsüdür

-> First = Üst

-> Second = Orta

-> Third = Düşük

» **who:** Yaşa göre sınıflama

-> man (18+)

-> woman (18+)

-> child (18 küçük)

» **adult_male:** 18 yaşında veya daha büyük bir erkek (0 = Hayır, 1=Evet)

» **deck:** geminin güvertesi

» **embark_town:** Liman isimleri (C, Q, S)

-> C = Cherbourg,

-> Q = Queenstown,

-> S = Southampton

» **alive:** yaşıyor mu?

-> Yes = Evet

-> No = Hayır

» **alone:** yalnız mı?

-> 1= yalnız,

-> 0= yalnız değil (gemide en az 1 kardeşiniz, eşiniz, ebeveyniniz veya çocuğunuz var)

Engin BOZABA

muhendis.github.io 

facebook.com/EnginBozabaOfficial 

twitter.com/enginbozaba 

instagram.com/enginbozaba 

linkedin.com/in/enginbozaba 

udemy.com/user/engin-bozaba-2 

youtube.com/c/EnginBozaba 

github.com/muhendis 

enginbozaba@gmail.com