

1. Veri Ön İşleme

S1: Ön-işleme nedir?

C1: Üzerinde çalışılacak veri setinde,

- Eksik veri: Verilerin girilmemiş olması, veri içerisinde null değerlerin olması
- Yanlış veri türü: Maaş değerinin -10 olması, yaşı 200 girilmesi
- Farklı girilen veri: Tarih için 1995 veya 02.12.1995 gibi değişik yazımlar (standart olmalı! VT mantığında da bunu **constrain/kısıt**'lerle sağlarız)

bulunabilir. Bunları ele almamız **eğitilen algoritmanın başarımını** etkileyecektir. Bu durumlarda eldeki veriler kullanılmadan önce veri ön işleme çalışması yapılmalıdır. Bu çalışmalardan “eksik veri” adımını inceleyelim:

Eksik Verilerin Temizlenmesi: Bu durumda kullanılacak yöntemler şu şekildedir:

- Sütundaki diğer verilerin ortalama değerini alarak eksik verilerin yerine yazmak
- Sütunda en çok tekrar eden değerleri eksik verilerin yerine yazmak

Bunun dışında ön-işleme adımlarında karşımıza çıkan önemli bir iki adım aşağıdaki gibidir

Kategorik Verilerin Kodlanması: Bazı algoritmalar kategorik verilerle çalışamaz. Bunların sayısal hale çevrilmesi/kodlanması gerekir.

Öznitelik Ölçekleme: Verilerin algoritma eğitimde kullanılırken normalize/standardize edilmeli yani aynı değer aralıklarına getirilmelidir. Normalizasyon bir kolonda bütün değerlerin en büyük değere bölünmesi gibi düşünülebilir (tüm değerler 0-1 aralığına girer).

2. Veri Ön İşleme Uygulaması

a. Anaconda Kurulumu

1. <https://www.anaconda.com/download/> adresinden Python 3.5 üstü versiyonu kurunuz.

b. **Veri Seti İndirme İşlemleri:** Bu derste kullanacağımız veri setleri <https://www.superdatascience.com/machine-learning/> adresinde yer almaktadır.

c. Kullanılacak Temel Kütüphaneler:

- scikit-learn: Veri analitiği ve makine öğrenmesi uygulamalarında temel yöntemleri içeren python kütüphanesi.
- Numpy: Python'da matematiksel işlemlerin temel paketidir.
- Matplotlib : Veri görselleştirme işlemleri için kullanılır.
- Pandas: Veri analizi ve veri ön işleme işlemlerinde kullanılan python kütüphanesidir.

d. Veri Seti Seçme ve Görüntüleme

Iloc: Veri seçme işlemi için kullanılır. İlk parametre seçilecek satırı, 2. Parametre seçilecek sütunu ifade eder. “ : ” şeklindeki gösterim tüm satırın seçileceği anlamına gelir. Son sütunu seçmek için -1 kullanılabilir.

.iloc selections - position based selection

`data.iloc[<row selection>, <column selection>]`

Integer list of rows: [0,1,2]

Slice of rows: [4:7]

Single values: 1

Integer list of columns: [0,1,2]

Slice of columns: [4:7]

Single column selections: 1

Not1: iloc, loc kullanımları için aşağıdaki kaynağı okuyunuz

<https://medium.com/@cemayan/pandas-ile-veri-analizi-2-data-selection-missing-values-4ecec921be87>

Not2: Python’da liste indisleri **0’dan başlar**. Örneğin [0:1,1:3] 0’ıncı satırı (ilk satır dahil 1 hariç!) ve birinci ve ikinci sütunları (gerçekte 0 dan başladığı için 2 ve 3 nolu sütunlar da çalışıyor oluruz indislere dikkat!)

Normalde seçtiklerine bir ekle..

0	1	2	3
Sıfır			
birinci			

Dataframe : Listeleri satır ve sütunlardan oluşan veri yapılarına dönüştürür

```

8 """ilgili dizindeki veri seti okunur.File explorerda veri setinin bulunduğu dizin
9 seçilmelidir"""
10
11 dataset = pd.read_csv('Data.csv')
12
13 x = dataset.iloc[:, :-1].values # son sütundaki verilerin seçilmesi
14 y= dataset.iloc[:, 3].values #3. sütundaki verilerin seçilmesi
15
16 """numpy tipindeki listeleri dataframe tipine dönüştürülür.Verit setinin
17 varibla explorerda görüntülenebilmesi için bu dönüşüm yapıldı."""
18
19 dfx = pd.DataFrame(x)
20 dfy = pd.DataFrame(y)

```

pd → pandas'tan pd olarak import edilmiş

[:, :-1] → son sütun hariç (son sütunda genellikle etiketler ve tahmin edilecek değerler oluyor onu hariç tutuyoruz!) hepsinin seçilmesi (x ve y eğitim/test için ayrılıyor aşağıda gelecek!)

e. Eksik Verilerin Temizlenmesi/Yerine Değer Atanması

Imputer: Eksik verileri gidermek için kullanılır.

Imputer.fit() :Eğitim verileri üzerinde istatistikleri (ortalama, median..) hesaplar.

Imputer.transform(): Hesaplanan ortalama vs. gibi “değeri” boş alana uygular.

Fit ve transformun ayrı ayrı yapılması veri kayıplarını engellemek içindir. Eğitim ve test verilerinin aynı olduğu biliniyorsa fit_transform() metodu da kullanılabilir.

```

22 #Eksik verilerin temizlenmesi
23
24 from sklearn.preprocessing import Imputer
25 #eksik değerlerin sütundaki diğer değerlerin ortalaması ile değiştirilir.
26 imputer = Imputer(missing_values = "NaN",strategy = "mean",axis = 0)
27 imputer = imputer.fit(x[:,1:3])
28 x[:, 1:3] = imputer.transform(x[:,1:3]) #1.kolon dahil, 3. dahil değil

```

Notlar:

- axis= 0 sütun boyunca değiştirme/güncelleme, 1 satır boyunca değiştirme/güncelleme
- imputer = Imputer(missing_values='NaN', strategy='mean',axis=0)
- imputer isimli örnek oluştur, boş alanlara NaN yaz, boş alanlara yazılacak değeri 'mean' ile hesapla ve sütun-kolon boyunca çalış.
- imputer = imputer.fit(x[:, 1:3]) → Şimdi bu işlemin X değerine fit edilmesi (tüm bu anlatılan işlemin x üzerinde yapılması, NaN ve mean işlemini yap)
- X[:, 1:3]=imputer.transform(x[:, 1:3]) → transform ile hesaplanan ortalama değerleri ilgili kolona yansıt ve x'i yeni değerleri ile elde et. **NOT:** fit hesap kitabı yaparken gerçek değiştirmeyi transform yapıyor.
- fit_transform metodu : Bu işlemleri tek seferde yapar → fit_transform(X[, y])

f. Kategorik Verilerin Kodlanması (algoritmalar kategorik veri istemiyor!)

```


30 #kategorik verilerin kodlanması
31 from sklearn.preprocessing import LabelEncoder, OneHotEncoder
32 #sayısal verilere dönüştürür
33 labelEncoder_x = LabelEncoder()
34 x[:,0] = labelEncoder_x.fit_transform(x[:,0])
35 #sayısal verilere binarizasyon yapar.
36 onehotencoder = OneHotEncoder(categorical_features = [0])
37 x= onehotencoder.fit_transform(x).toarray()
38 #sayısal verilere dönüştürür
39 labelEncoder_y = LabelEncoder()
40 y = labelEncoder_y.fit_transform(y)
41

```

labelEncoder: sklearn.preprocessing kütüphanesinde bulunur. Veriyi birebir sayısallaştırmaya yarar. Kategorik her veriye sayısal bir değer atar (kaç grup varsa o kadar sayı kullanır!). Örneğin aşağıdaki ülkeler örneği için 1, 2, 3 gibi değerler kullanır

ama pratik olarak bunun anlamını çözmek daha zor olduğu için genellikle OneHotEncoder kullanılır.

OneHotEncoder: sklearn.preprocessing kütüphanesinden bulunan encoder bu kategorik verilerin binarizasyonunu gerçekleştirmemizi sağlar. Şekil 1deki gibi x matrisindeki 1. Kolon olan country kolonunda var olan değere “1” verip olmayanlara “0” vermektedir.



Country		France	Germany	Spain
France		1	0	0
Spain		0	0	1
Germany		0	1	0
Spain		0	0	1
Germany		0	1	0
France		1	0	0
Spain		0	0	1
France		1	0	0
Germany		0	1	0
France		1	0	0

Burada örneğin fransanın kodu 100, almanyanın ki 010 şeklinde düşünülebilir.

g. Veri Setinin Test ve Eğitim Olarak Bölümlenmesi

Sklearn.modelselection.train_test_split: Liste veya matrisleri rastgele eğitim ve test verilerine ayırır. Her seferinde aynı örnekleri aynı grupta tutmak istiyorsak (deneyin aynı şekilde tekrarlanabilmesi için!) random_state =0 veya 1 veya 42 veya başka ‘integer’ kullanılabilir. Tamamen random olması isteniyorsa o zaman random_state parametresi kullanılmaz.

test_size = Test için ayrılacak verilerin oranını gösterir. 0-1 arasında değer alır.

train_size = Eğitim için a ayrılacak verilerin oranını gösterir. 0-1 arasında değer alır.

test_size + train_size = 1 olacağı için birinin yazılması yeterlidir.

```
42 #veri setini eğitim ve test olarak bölümlene
43 from sklearn.model_selection import train_test_split
44 x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random_state = 0)
```

Not: python da tahmin yapılacak kolon (y) ayrılıyor..

Yukarıda gördüklerimizden kullanılacak olanlar ve varılacak hedefi inceleyelim!

Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40		Yes
France	35	58000	Yes
Spain		52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes
Germany	50	60000	???
MODEL CANLIYA GEÇECEK			

