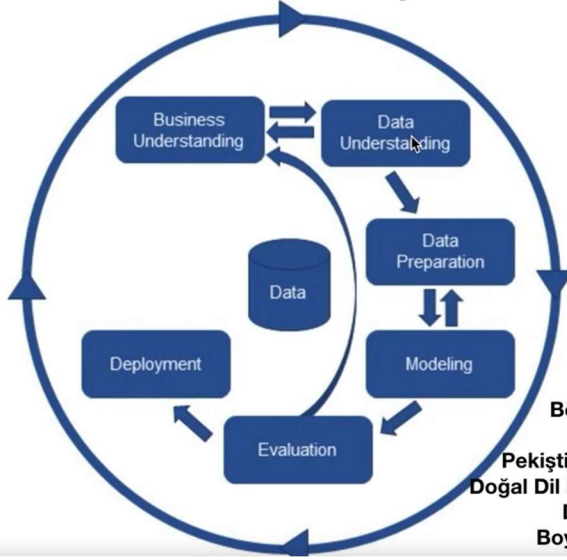


## VERİ ÖNİŞLEME[DATA PREPROCESSING]

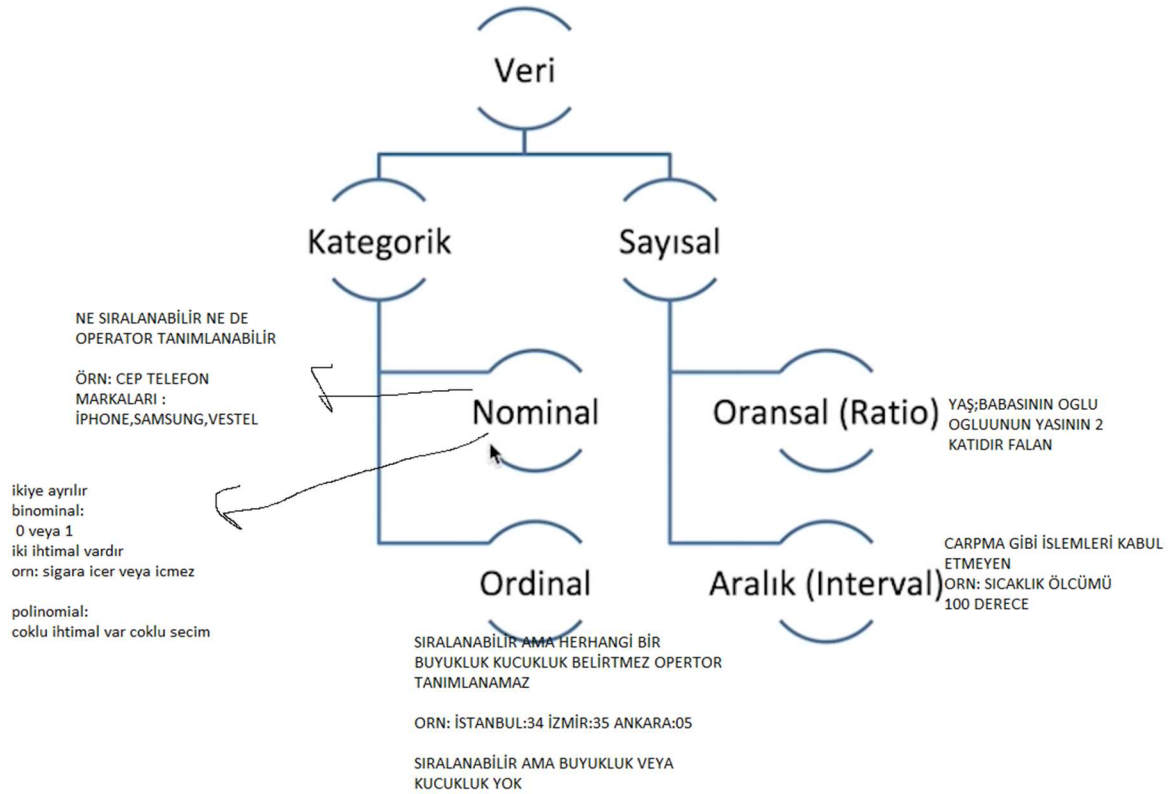
### Metodoloji : CRISP-DM

Cross Industry Standard  
Process for Data Mining



Tahmin (Prediction)  
Sınıflandırma (classification)  
Bölütleme (Kümeleme, Clustering)  
Birliktelik Kural Çıkarımı (ARM)  
Pekiştirmeli Öğrenme (Reinforced Learning)  
Doğal Dil İşleme ve Metin İşleme (Text Processing)  
Derin Öğrenme (Deep Learning)  
Boyut indirgeme ve Veri Dönüşümleri

öğrenilen kısım : data preparation



A	B	C	D	E
ulke	boy	kilo	yas	cinsiyet
tr	130	30	10	e
tr	125	36	11	e
tr	135	34	10	k
tr	133	30	9	k
tr	129	38	12	e
tr	180	90	30	e
tr	190	80	25	e
tr	175	90	35	e
tr	177	60	22	k
us	185	105	33	e
us	165	55	27	k
us	155	50	44	k
us	160	58		k
us	162	59	41	k
us	167	62	55	k
fr	174	70	47	e
fr	193	90		e
fr	187	80	27	e
fr	183	88	28	e
fr	159	40	29	k
fr	164	66	32	k
fr	166	56	42	k
			28.45	

Buradaki ülke bilgilerini makinenin anlayacağı sayısal değerlere çevirmemiz gerekiyor; encode işlemi;

	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

Yukarıda yapılan işlem one-hot encoding (encode-> **kategorik verilerin sayısal**la dönüştürmek)

"Encode" işlemi, kategorik verilerin sayısal değerlere dönüştürülmesini sağlayan bir dönüşüm işlemidir. Örneğin, cinsiyet gibi bir kategorik değişkeni kodlamak için "One-Hot Encoding" veya "Label Encoding" gibi teknikler kullanılabilir.

- ```
ulke = veriler.iloc[:,0:1].values
print(ulke)

from sklearn import preprocessing # -> ön işleme

labelEnCode = preprocessing.LabelEncoder()

ulke[:,0]= labelEnCode.fit_transform(veriler.iloc[:,0]) #->fit_transform yöntemi, veriyi hem eğitir
# hem de dönüştürür bu nedenle genellikle
#eğitim verisi üzerinde kullanılır

print(ulke)

oneHotEnCode = preprocessing.OneHotEncoder()
ulke= oneHotEnCode.fit_transform(ulke).toarray()
print(ulke)
```

The figure displays two side-by-side screenshots of the 'veriler' application, which is used for visualizing data frames with background color coding.

**Left Screenshot (sonuc - DataFrame):** This window shows a data frame with 14 rows and 8 columns. The columns are 'Index', 'fr', 'tr', 'us', 'boy', 'kilo', 'yas', and 'cinsiyet'. The data is color-coded based on the 'cinsiyet' (gender) column. Rows with 'e' (female) have a red background, while rows with 'k' (male) have a blue background. The data is as follows:

| Index | fr | tr | us | boy | kilo | yas   | cinsiyet |
|-------|----|----|----|-----|------|-------|----------|
| 0     | 0  | 1  | 0  | 130 | 30   | 10    | e        |
| 1     | 0  | 1  | 0  | 125 | 36   | 11    | e        |
| 2     | 0  | 1  | 0  | 135 | 34   | 10    | k        |
| 3     | 0  | 1  | 0  | 133 | 30   | 9     | k        |
| 4     | 0  | 1  | 0  | 129 | 38   | 12    | e        |
| 5     | 0  | 1  | 0  | 180 | 90   | 30    | e        |
| 6     | 0  | 1  | 0  | 190 | 80   | 25    | e        |
| 7     | 0  | 1  | 0  | 175 | 90   | 35    | e        |
| 8     | 0  | 1  | 0  | 177 | 60   | 22    | k        |
| 9     | 0  | 0  | 1  | 185 | 105  | 33    | e        |
| 10    | 0  | 0  | 1  | 165 | 55   | 27    | k        |
| 11    | 0  | 0  | 1  | 155 | 50   | 44    | k        |
| 12    | 0  | 0  | 1  | 160 | 58   | 28.45 | k        |
| 13    | 0  | 0  | 1  | 162 | 59   | 41    | k        |

**Right Screenshot (veriler - DataFrame):** This window shows the same data frame, but with a different color-coding scheme. The background color is now determined by the 'us' (USA) column. Rows with '1' (USA) have a blue background, while rows with '0' (not USA) have a red background. The data is as follows:

| Index | ulke | boy | kilo | yas | cinsiyet |
|-------|------|-----|------|-----|----------|
| 0     | tr   | 130 | 30   | 10  | e        |
| 1     | tr   | 125 | 36   | 11  | e        |
| 2     | tr   | 135 | 34   | 10  | k        |
| 3     | tr   | 133 | 30   | 9   | k        |
| 4     | tr   | 129 | 38   | 12  | e        |
| 5     | tr   | 180 | 90   | 30  | e        |
| 6     | tr   | 190 | 80   | 25  | e        |
| 7     | tr   | 175 | 90   | 35  | e        |
| 8     | tr   | 177 | 60   | 22  | k        |
| 9     | us   | 185 | 105  | 33  | e        |
| 10    | us   | 165 | 55   | 27  | k        |
| 11    | us   | 155 | 50   | 44  | k        |
| 12    | us   | 160 | 58   | nan | k        |

Both screenshots show a toolbar at the bottom with options: 'Format', 'Resize', 'Background color' (checked), 'Column min/max' (checked), 'Save and Close', and 'Close'.

Verilerin test ve train olarak bolunmesi ;

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(s, cinsiyetDF, test_size=0.33, random_state=0)
# x bağımsız y bağımlı degisken
# test_size -> %67 train %33 test
# hedef = bağımlı degisken
# x train ve x testleri kullanarak y[bağımlı degiskeni] tahmin ettircez
```

x\_test - DataFrame

| Index | fr | tr | us | boy | kilo | yas |
|-------|----|----|----|-----|------|-----|
| 20    | 1  | 0  | 0  | 164 | 66   | 32  |
| 10    | 0  | 0  | 1  | 165 | 55   | 27  |
| 14    | 0  | 0  | 1  | 167 | 62   | 55  |
| 13    | 0  | 0  | 1  | 162 | 59   | 41  |
| 1     | 0  | 1  | 0  | 125 | 36   | 11  |
| 21    | 1  | 0  | 0  | 166 | 56   | 42  |
| 11    | 0  | 0  | 1  | 155 | 50   | 44  |
| 19    | 1  | 0  | 0  | 159 | 40   | 29  |

y\_test - DataFrame

| Index | cinsiyet |
|-------|----------|
| 20    | k        |
| 10    | k        |
| 14    | k        |
| 13    | k        |
| 1     | e        |
| 21    | k        |
| 11    | k        |
| 19    | k        |

x\_train - DataFrame

| Index | fr | tr | us | boy | kilo | yas   |
|-------|----|----|----|-----|------|-------|
| 8     | 0  | 1  | 0  | 177 | 60   | 22    |
| 6     | 0  | 1  | 0  | 190 | 80   | 25    |
| 16    | 1  | 0  | 0  | 193 | 90   | 28.45 |
| 4     | 0  | 1  | 0  | 129 | 38   | 12    |
| 2     | 0  | 1  | 0  | 135 | 34   | 10    |
| 5     | 0  | 1  | 0  | 180 | 90   | 30    |
| 17    | 1  | 0  | 0  | 187 | 80   | 27    |
| 9     | 0  | 0  | 1  | 185 | 105  | 33    |
| 7     | 0  | 1  | 0  | 175 | 90   | 35    |
| 18    | 1  | 0  | 0  | 183 | 88   | 28    |
| 3     | 0  | 1  | 0  | 133 | 30   | 9     |
| 0     | 0  | 1  | 0  | 130 | 30   | 10    |
| 15    | 1  | 0  | 0  | 174 | 70   | 47    |
| 12    | 0  | 0  | 1  | 160 | 58   | 28.45 |

y\_train - DataFrame

| Index | cinsiyet |
|-------|----------|
| 8     | k        |
| 6     | e        |
| 16    | e        |
| 4     | e        |
| 2     | k        |
| 5     | e        |
| 17    | e        |
| 9     | e        |
| 7     | e        |
| 18    | e        |
| 3     | k        |
| 0     | e        |
| 15    | e        |
| 12    | k        |



Evet, `'train_test_split'` fonksiyonunun genel kullanımı bu şekildedir. Bu fonksiyon, veri setini eğitim ve test setlerine bölmek için oldukça yaygın bir araçtır ve makine öğrenimi modelinin eğitilmesi ve değerlendirilmesi için veri bölme işlemini gerçekleştirmek için sıklıkla kullanılır.

Genel kullanım şu şekildedir:

python

Copy code

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(s, cinsiyetDF, test_size=0.
```

Burada:

- `'s'`: Bağımsız değişkenleri (özellikleri) içeren veri setidir.
- `'cinsiyetDF'`: Bağımlı değişkeni (hedef değişkeni) içeren veri setidir.
- `'test_size'`: Veri setinin ne kadarının test seti olarak ayrılacağını belirten orandır. Genellikle 0 ile 1 arasında bir değerdir ve bu oran, test setinin toplam veri setine oranını ifade eder.
- `'random_state'`: Veri setinin rastgele bölünmesinde kullanılacak rastgele durumun belirlenmesini sağlayan bir değerdir. Bu, işlemi tekrarlandığında aynı bölünmenin elde edilmesini sağlar.

Bu satır, veri setini eğitim ve test setlerine böler ve sonuç olarak dört değişkene atanır: `'x_train'`, `'x_test'`, `'y_train'`, ve `'y_test'`. Bu değişkenler, sırasıyla eğitim setinin özellikleri, test setinin özellikleri, eğitim setinin hedefleri ve test setinin hedeflerini içerir. Bu şekilde, model eğitimi ve performans değerlendirmesi için gerekli veriler elde edilmiş olur.

Öznitelik ölçekleme; [verileri aynı dünyaya çevirmek birbirine yakın sayılar elde etmek][birbirine göre ölçeklenmiş oldu standart scaler kullanarak]

X\_test - NumPy object array

|   | 0         | 1         | 2  | 3         | 4         | 5         |
|---|-----------|-----------|----|-----------|-----------|-----------|
| 0 | 1.29099   | -0.377964 | -1 | 0.4724    | 1.32854   | -0.249913 |
| 1 | -0.774597 | -0.377964 | 1  | 0.549527  | 0.20439   | -0.649773 |
| 2 | -0.774597 | -0.377964 | 1  | 0.70378   | 0.919757  | 1.58944   |
| 3 | -0.774597 | -0.377964 | 1  | 0.318147  | 0.613171  | 0.469836  |
| 4 | -0.774597 | 2.64575   | -1 | -2.53554  | -1.73732  | -1.92932  |
| 5 | 1.29099   | -0.377964 | -1 | 0.626653  | 0.306586  | 0.549808  |
| 6 | -0.774597 | -0.377964 | 1  | -0.221739 | -0.306586 | 0.709752  |
| 7 | 1.29099   | -0.377964 | -1 | 0.0867674 | -1.32854  | -0.489829 |

X\_train - NumPy object array

|    | 0         | 1        | 2         | 3         | 4         | 5         |
|----|-----------|----------|-----------|-----------|-----------|-----------|
| 0  | -0.632456 | 0.866025 | -0.408248 | 0.450494  | -0.296579 | -0.247171 |
| 1  | -0.632456 | 0.866025 | -0.408248 | 1.00825   | 0.509655  | 0.0341619 |
| 2  | 1.58114   | -1.1547  | -0.408248 | 1.13696   | 0.912772  | 0.357695  |
| 3  | -0.632456 | 0.866025 | -0.408248 | -1.60891  | -1.18344  | -1.18495  |
| 4  | -0.632456 | 0.866025 | -0.408248 | -1.35148  | -1.34468  | -1.3725   |
| 5  | -0.632456 | 0.866025 | -0.408248 | 0.579207  | 0.912772  | 0.503051  |
| 6  | 1.58114   | -1.1547  | -0.408248 | 0.879537  | 0.509655  | 0.221717  |
| 7  | -0.632456 | -1.1547  | 2.44949   | 0.793728  | 1.51745   | 0.784384  |
| 8  | -0.632456 | 0.866025 | -0.408248 | 0.364686  | 0.912772  | 0.971939  |
| 9  | 1.58114   | -1.1547  | -0.408248 | 0.70792   | 0.832148  | 0.315495  |
| 10 | -0.632456 | 0.866025 | -0.408248 | -1.43729  | -1.50593  | -1.46628  |
| 11 | -0.632456 | 0.866025 | -0.408248 | -1.566    | -1.50593  | -1.3725   |
| 12 | 1.58114   | -1.1547  | -0.408248 | 0.321782  | 0.106538  | 2.09727   |
| 13 | -0.632456 | -1.1547  | 2.44949   | -0.278878 | -0.377202 | 0.357695  |

```
from sklearn.preprocessing import StandardScaler # -> verileri standartlaştırmaya yarar

sc=StandardScaler()

X_train = sc.fit_transform(x_train)
X_test = sc.fit_transform(x_test)
#Öznitelik ölçekleme; [verileri aynı dünyaya çevirmek birbirine yakın sayılar elde etmek]
#[birbirine göre ölçeklenmiş oldu]
# VERİLER STANDARTLAŞTIRILDI [StandardScaler]
```

## Veri Önleme - ADIMLAR:

- 1- Veri yükleme: [pd.read\_csv('eksikveriler.csv')]
- 2- Eksik veri tamamlama: [imputer= SimpleImputer(missing\_values=np.nan, strategy='mean')]
- 3- Kategorik[nominal] veri donuşumu: [preprocessing.OneHotEncoder()]
- 4- Veri kümelerini birleştirme: [pd.concat([s,cinsiyetDF],axis=1)]
- 5- Verileri bölme: [train and test]  
[x\_train, x\_test,y\_train ,y\_test =train\_test\_split(s,cinsiyetDF,test\_size=0.33,random\_state=0)]
- 6- Öznitelik ölçekleme: [StandardScaler().fit\_transform(x\_train)]



### ChatGPT

Bu iki kod parçası arasındaki fark, "age" değişkeninin nasıl atanmış olduğudur.

İlk kod parçasında:

python

Copy code

```
age = data.iloc[:,3:4].values
```

Burada `iloc` metodu kullanılarak `data` veri çerçevesinin tüm satırlarını ve 3. sütununu (indeks numarasıyla 3) içeren bir alt veri çerçevesi seçilir. `.values` metodu bu alt veri çerçevesini bir NumPy dizisine dönüştürür. Bu şekilde `age` değişkeni 2 boyutlu bir NumPy dizisi olacaktır.

İkinci kod parçasında ise:

python

Copy code

```
age = data.iloc[:,3].values
```

Burada yine `iloc` metodu kullanılarak `data` veri çerçevesinin tüm satırlarını ve 3. sütununu seçeriz. Ancak bu sefer sadece bir sütun seçtiğimiz için çıktı 1 boyutlu bir NumPy dizisi olur.

Yani temel fark, ilk kod parçasında bir sütun seçilirken, ikinci kod parçasında bir sütun dilimleme işlemi yapılarak seçilmiştir.



One-hot encoding : ülke gibi herhangi bir sıralaması olmayan - cinsiyet

Label encoding : sıralaması olacak