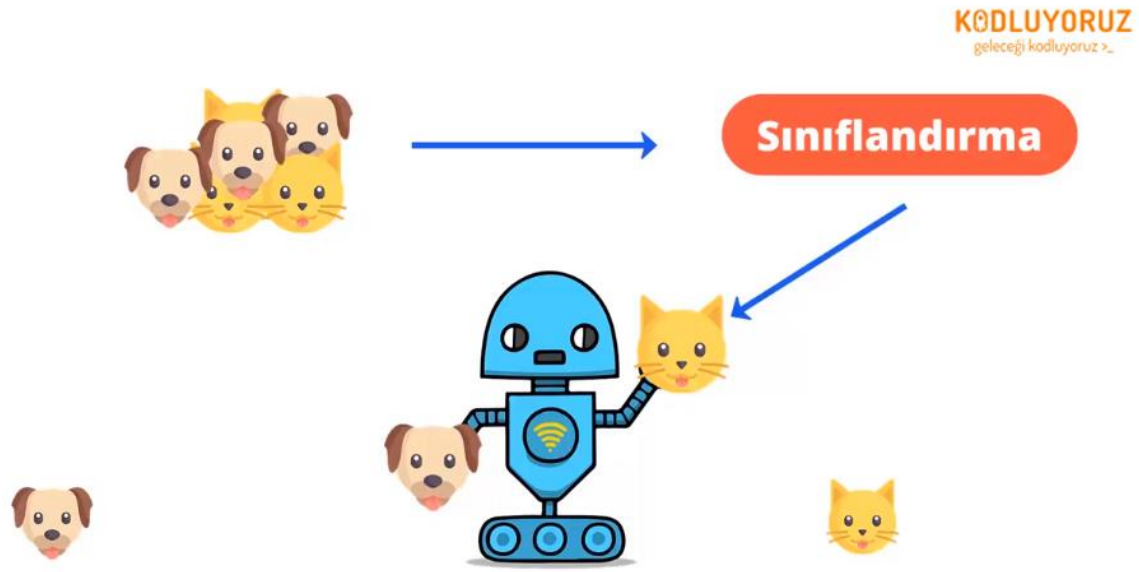
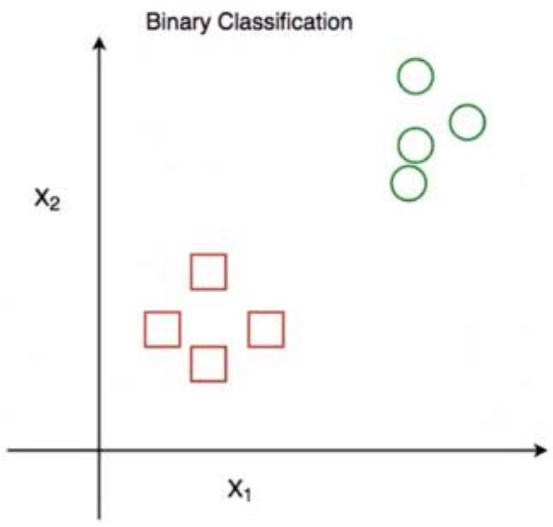
SINIFLANDIRMA

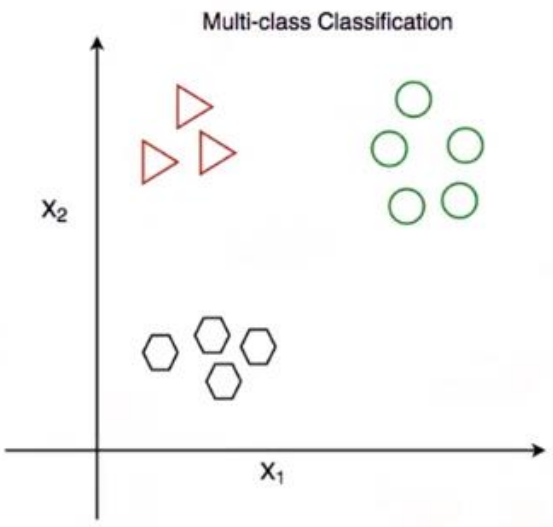
Sınıflandırma, alt kategorilere ayırma işlemidir. Makine öğreniminde sınıflandırma, kategorilerinin ne olduğunu bildiğimiz bir eğitim veri setine bakarak yeni bir verinin hangi kategoriye ait olduğunu belirleme sorunudur.



Makine öğrenmesinde sınıflandırmayı genellikle iki farklı türde ele alırız.

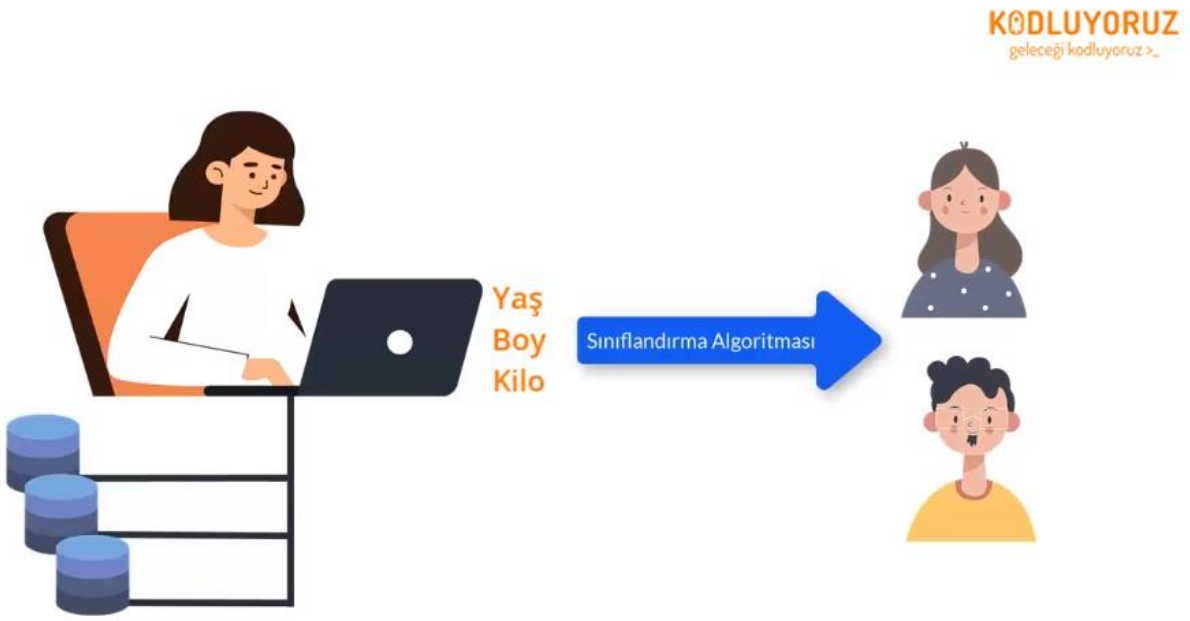
İlki Binary Classification olarak adlandırılan ve tam olarak iki sınıf arasında ayrım yapmamızı sağlayan sınıflandırma türü, diğeri ise Multiclass Classification olarak adlandırılan ve ikiden fazla sınıflı bir durum olduğunda kullandığımız bir sınıflandırmadır.





Örnek yapma:

Elimizde olan yaş, boy, kilo değerlerine göre bir yolcumuzun cinsiyetini öğrenecek olsun. Bu veri setimiz bizim eğitim verilerimiz olmuş olacak. Daha sonra algoritmamız bu bilgilere sahip ve yen gelen bir yolcunun cinsiyetini otomatik olarak tahmin edecek.



Kadın ve erkek etiketleri için ayrı ayrı her bir sütun değerlerinin ortalamasını alalım:

EMpower

yas	boy	kilo	cinsiyet
19	162	50	Kadın
22	169	55	Kadın
21	171	57	Kadın
23	161	55	Kadın

Yaş ortalaması = 21.25

Boy ortalaması = 165.75

Kilo ortalaması = 54.25

yas	boy	kilo	cinsiyet
20	175	70	Erkek
21	179	80	Erkek
20	183	90	Erkek
19	181	75	Erkek

Yaş ortalaması = 20

Boy ortalaması = 179.5

Kilo ortalaması = 78.75

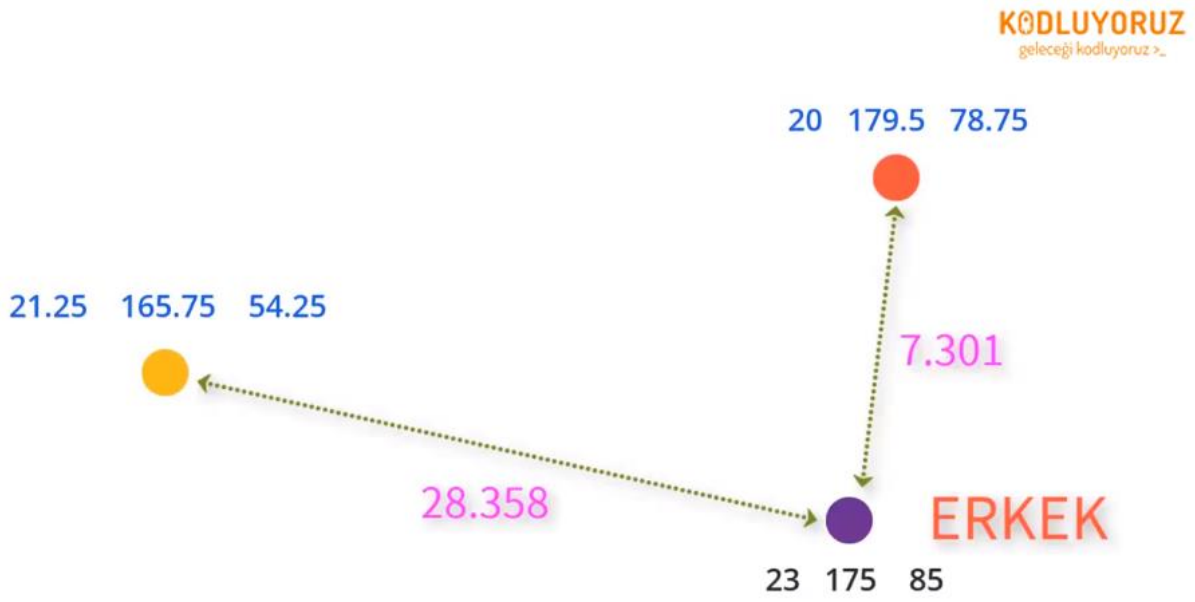
Bulmuş olduğumuz değer bizim algoritmamızın öğrenmiş olduğu değer olacak.

Yeni gelen bir yolcumuzun bilgileri:

yas	boy	kilo
23	185	75

Daha önce algoritmamızın öğrenmiş olduğu değerlerimize göre bu kişinin cinsiyetini tahmin edelim:

Öklid uzaklık hesaplamasını kullanarak yeni gelen yolcumuzun değerlere uzaklığını hesaplama:



Hesaplanan sonuçlara göre algoritmamız yeni gelen kişinin cinsiyetini Erkek sınıfında tanımladı.

Makine öğreniminde en yaygın algoritma olan Lojistik Regresyon Algoritması'nı ele alalım.

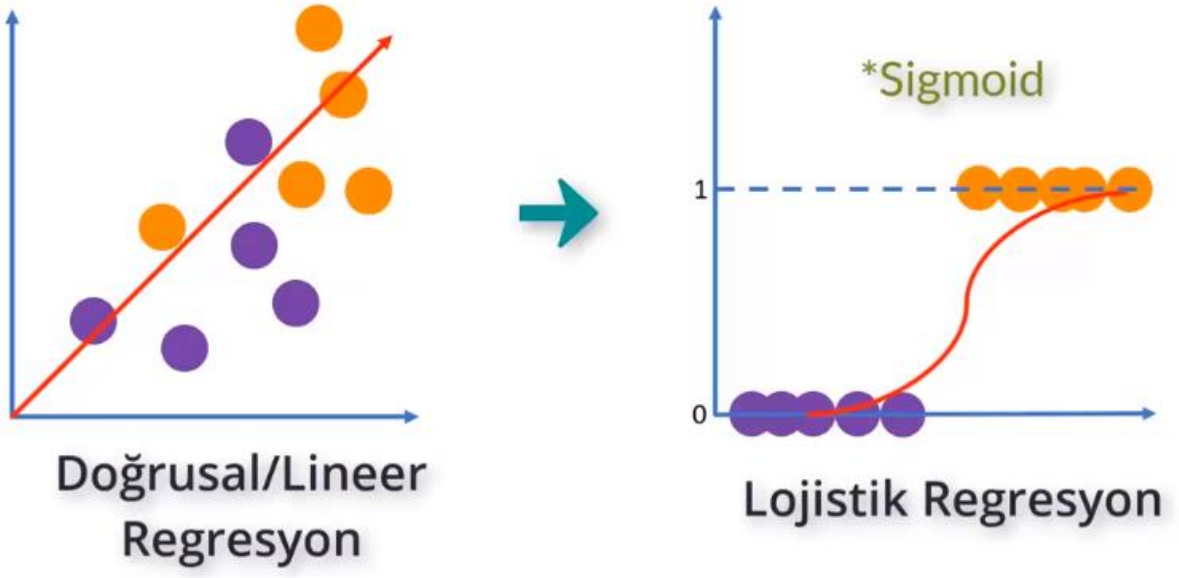
Makine öğrenimi dünyasında Lojistik Regresyon, adında "regresyon" kelimesi bulunmasına rağmen bir tür parametrik sınıflandırma modelidir. Hedef değişkenimizin kategorik olduğu durumlarda kullanılan bir algoritmadır.

Ders Çalışma Süreleri	Ders Durumu	
X	Y	
72 saat	1	Geçti
45 saat	0	Kaldı
103 saat	1	Geçti
36 saat	0	Kaldı
13 saat	0	Kaldı

Geçti olma olasılığı $P(Y=1 \mid X)$

Kaldı olma olasılığı $P(Y=0 \mid X)$

Lojistik Regresyonda verilerimizi doğrusal regresyonda olduğu gibi doğrudan düz bir çizgi üzerine çizdiremeyiz. Bunun yerine gözlemlerimize grafik üzerinde gördüğümüz gibi Sigmoid adı verilen S şekilli bir eğri uyduruyoruz.



Y ekseni 0'dan 1'e gider. Bunun nedeni sigmoid fonksiyonunun her zaman maksimum ve minimum bu iki değeri almasıdır.

Sınıflandırma Uygulaması

Kütüphane import etme ve verimizi okuma:

```
[1] import pandas as pd
```

```
df = pd.read_csv('/content/cardio_train.csv', sep=';')
```

```
df.head()
```

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

İçerisindeki değerler gün olarak verilmiş bir age kolonu, cinsiyet bilgisini 1 ve 2 şeklinde kategorik olarak tutan gender kolonu, height ve weight kolonumuz var. Bir de hastalık bulgusunu anlamamıza yarayacak olan muayene edilerek ölçülmüş veriler var. Bunlar tansiyon, kan basıncı, kolesterol ve glikoz değerleri. Diğer bilgiler ise sigara içilip içilmediği, alkol kullanımı ve fiziksel aktivite yapılıp yapılmadığına dair bilgiler içeren kolonlardır. Bu bilgiler de kategorik olarak 0 ve 1 şeklinde ifade edilmiştir. En son kolonumuz ise bu kişilerin kardiyovasküler hastalığının olup olmadığını gösteren cardio kolonudur. Yani target değerler.

Burada bulunan id kolonunu kullanmayacağımız için onu silebiliriz.

```
df.drop('id', axis = 1, inplace = True)
```

```
[6] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   age          70000 non-null  int64  
 1   gender       70000 non-null  int64  
 2   height       70000 non-null  int64  
 3   weight       70000 non-null  float64
 4   ap_hi        70000 non-null  int64  
 5   ap_lo        70000 non-null  int64  
 6   cholesterol  70000 non-null  int64  
 7   gluc         70000 non-null  int64  
 8   smoke        70000 non-null  int64  
 9   alco         70000 non-null  int64  
 10  active       70000 non-null  int64  
 11  cardio       70000 non-null  int64  
dtypes: float64(1), int64(11)
memory usage: 6.4 MB
```

Veri seti üzerindeki işlemlerimizden önce tekrar eden değerleri ele almak çok önemlidir. Tekrar eden değerlerin model eğitimine herhangi bir etkisi yoktur. Sadece eğitim boyutunu arttırırlar. Tekrarlı verileri bulmak için duplicated() fonksiyonunu kullanıyoruz.

```
print("{} tane var".format(df.duplicated().sum()))
```

```
24 tane var
```

```
dp = df[df.duplicated(keep=False)]
dp = dp.sort_values(by=['age', "gender", "height"], ascending= False)

dp.head(2)
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
2677	22077	1	175	69.0	120	80	1	1	0	0	1	1
45748	22077	1	175	69.0	120	80	1	1	0	0	1	1

Tekrarlı verilerimizin olup olmadığını kontrol edelim:

```
df.drop_duplicates(inplace= True)
print("{} tane var".format(df.duplicated().sum()))
```

```
0 tane var
```

Aykırı değerlerimizi tespit etmek:

Aykırı değerlerimizi tespit etmek ve bunları ele almak doğruluk değerimizi artırabilir.

```
df.describe().T
```

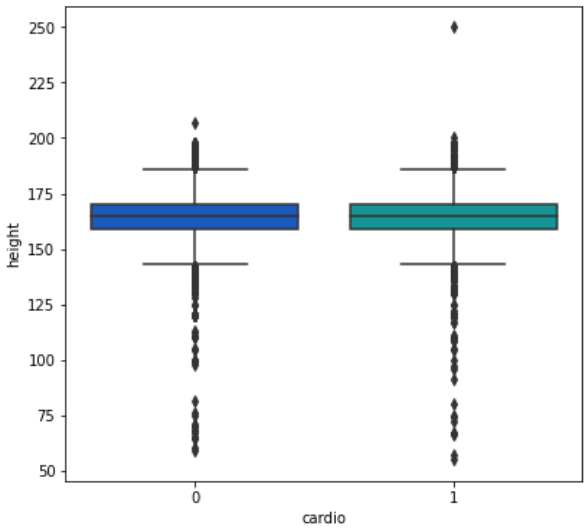
	count	mean	std	min	25%	50%	75%	max
age	69976.0	19468.950126	2467.374620	10798.0	17664.0	19703.0	21327.0	23713.0
gender	69976.0	1.349648	0.476862	1.0	1.0	1.0	2.0	2.0
height	69976.0	164.359152	8.211218	55.0	159.0	165.0	170.0	250.0
weight	69976.0	74.208519	14.397211	10.0	65.0	72.0	82.0	200.0
ap_hi	69976.0	128.820453	154.037729	-150.0	120.0	120.0	140.0	16020.0
ap_lo	69976.0	96.636261	188.504581	-70.0	80.0	80.0	90.0	11000.0
cholesterol	69976.0	1.366997	0.680333	1.0	1.0	1.0	2.0	3.0
gluc	69976.0	1.226535	0.572353	1.0	1.0	1.0	1.0	3.0
smoke	69976.0	0.088159	0.283528	0.0	0.0	0.0	0.0	1.0
alco	69976.0	0.053790	0.225604	0.0	0.0	0.0	0.0	1.0
active	69976.0	0.803718	0.397187	0.0	1.0	1.0	1.0	1.0
cardio	69976.0	0.499771	0.500004	0.0	0.0	0.0	1.0	1.0

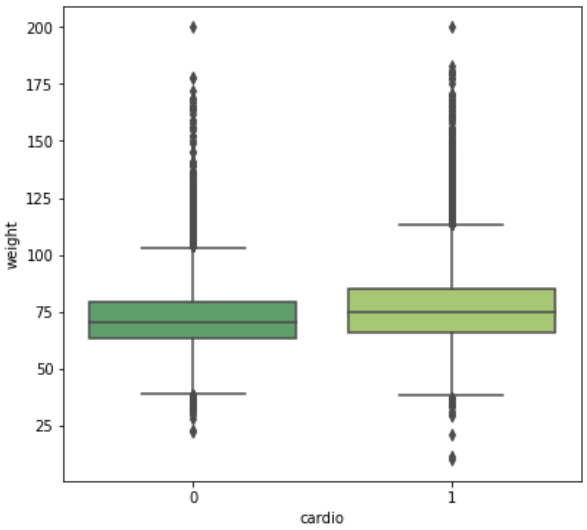
Burada T, transpozunu alarak daha güzel görüntülememizi sağlıyor. Aykırı değer, diğer gözlemlerden önemli ölçüde farklı olan bir veri noktası. Ölçümdeki değişkenlikten kaynaklanıyor olabilir veya deneysel hatayı gösterebilir.

Boxplot kullanarak height ve weight kolonları için aykırı verilerimizi gösterelim. Seaborn ve matplotlib kütüphanelerini ekliyoruz ve grafiğini çizdiriyoruz.

```
import seaborn as sns
import matplotlib.pyplot as plt


plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
sns.boxplot(x='cardio',y='height',data=df,palette='winter')
plt.subplot(1,2,2)
sns.boxplot(x='cardio',y='weight',data=df,palette='summer')
```





Height ve weight bir arada değerlendirmek için vücut kitle endeksi dediğimiz BMI'yı kullanalım. Vücut kitle indeksi, tıbbi değerlendirme ve kalp sağlığı için kullanılan yaygın bir ölçümdür.

```
df["bmi"] = (df["weight"] / (df["height"] / 100)**2).round(1)
```

```
df.head()
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	bmi
0	18393	2	168	62.0	110	80	1	1	0	0	1	0	22.0
1	20228	1	156	85.0	140	90	3	1	0	0	1	1	34.9
2	18857	1	165	64.0	130	70	3	1	0	0	0	1	23.5
3	17623	2	169	82.0	150	100	1	1	0	0	1	1	28.7
4	17474	1	156	56.0	100	60	1	1	0	0	0	0	23.0

Aşırı zayıf, obez, uzun ve kısa insanları çıkarma:

Böylelikle height ve weight değişkenlerine ihtiyacımız olmadığı için onları silebiliriz.

```
[14] df = df[(df["bmi"]>10) & (df["bmi"]<100)]
```

```
df.drop(["weight","height"],axis = 1,inplace = True)
```

Şimdi sırada tansiyon ve kan basıncı değerleri var. Bu değerlerin negatif olamayacağını, sırasıyla 180 ve 120'den fazla değerlerin acil durumu gösterdiğini biliyoruz. Bu nedenle 250 ve 200'den yukarıdaki değerleri bu uygulamamızda almıyoruz. Age kolonu gün olarak tutulmuş yıla çevirmeliyiz.

```
df = df[(df['ap_hi'] < 250) & (df['ap_lo'] < 200)]
df = df[(df['ap_hi'] > 20) & (df['ap_lo'] > 20)]

df['age'] =  df['age'] / 365
```

```
df.head()
```

	age	gender	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	bmi
0	18393	2	110	80	1	1	0	0	1	0	22.0
1	20228	1	140	90	3	1	0	0	1	1	34.9
2	18857	1	130	70	3	1	0	0	0	1	23.5
3	17623	2	150	100	1	1	0	0	1	1	28.7
4	17474	1	100	60	1	1	0	0	0	0	23.0

Kolesterol ve glikoz değerlerimiz 1,2 ve 3 rakamlarıyla ifade edilen kategorilerden oluşuyor. Modelimize bu şekilde verirsek bu 3'ü 2'den üstün görebilir. Bu nedenle encoding işlemi uygulayacağız. (one-hot encoding)

```
[17] df['cholesterol'].unique()

array([1, 3, 2])
```

```
df['cholesterol'] = df['cholesterol'].map({ 1: 'normal', 2: 'aboveNormal', 3: 'wellAboveNormal'})
df['gluc']=df['gluc'].map({ 1: 'normal', 2: 'aboveNormal', 3: 'wellAboveNormal'})

dummies = pd.get_dummies(df[['cholesterol','gluc']])


final_df = pd.concat([df,dummies],axis=1)
final_df.drop(['cholesterol','gluc'],axis=1,inplace=True)
final_df.head()
```

```
final_df["gender"] = final_df["gender"] % 2
```

Modelleme işlemi:

```
from sklearn.model_selection import train_test_split

y = final_df["cardio"]
X = final_df.drop("cardio", axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=17)
```

```
[21] X_train.shape

(48107, 14)
```

```
X_test.shape

(20618, 14)
```

Burada fit, elimizde train olarak ayırdığımız verilerden öğrenme gerçekleştiriyor. Daha sonra predict kullanarak eğitim yaparken hiç görmediğimiz x_test verilerini kullanarak hastalık olup olmadığına dair tahminler üretiyoruz ve bunu y_pred adlı değişkenimize aktarıyoruz. Bu y_pred değişkenimizi kullanarak daha önceden ayırmış olduğumuz y_test verileri ile karşılaştıracağız. Bu şekilde üretmiş olduğumuz modelin ne kadar doğrulukla çalıştığını görebiliriz.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

logreg = LogisticRegression()

logreg.fit(X_train,y_train)

y_pred = pd.Series(logreg.predict(X_test))

accuracy_score(y_test,y_pred)
```

```
0.7212629741003007
```

0.72 oranında bir doğruluk ile kardiyovasküler bir hastalığı olup olmadığını söyleyebilir.

KAYNAKÇA

Bilgeiş “Herkes için Yapay Zekâ II” eğitimi.

KODLUYORUZ
geleceği kodluyoruz >_

EMpower
Enriching young lives in emerging markets