5. Hafta-Sklearn Kütüphanesi

Scikit-learn kütüphanesi;

1. Veri analizi için kullanılan kolay ve etkili bir araçtır.
2. NumPy, SciPy ve matplotlib kütüphaneleri üzerine kurulmuştur.
3. Açık Kaynaklı BSD lisansı izinlerine sahiptir.

https://scikit-learn.org/stable/ (scikit-learn)

In [ ]:
```
# Scikit-learn kütüphanesi sayesinde makine öğrenmesi uygulamaları yapabiliriz.
# Önce kütüphaneyi notebook'a alalım ve birkaç veri seti yükleyelim.
```

Tüm verisetlerine ulaş (https://scikit-learn.org/stable/datasets/index.html)

Veri seti, üç Iris türünün (Iris setosa, Iris virginica ve Iris versicolor) her birinden 50 örnekten oluşur. Her örnekten dört özellik vardır: sepal uzunluğu, genişliği, petal uzunluğu ve genişliği. Bu dört özelliğin kombinasyonuna dayanarak, türleri ayırt etmek için doğrusal bir model geliştirilmiştir.

In [1]:
```
from sklearn import datasets
iris = datasets.load_iris()
# veri setini görelim
iris
```

Out[1]:
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
                [4.9, 3. , 1.4, 0.2],
                [4.7, 3.2, 1.3, 0.2],
                [4.6, 3.1, 1.5, 0.2],
                [5. , 3.6, 1.4, 0.2],
                [5.4, 3.9, 1.7, 0.4],
                [4.6, 3.4, 1.4, 0.3],
                [5. , 3.4, 1.5, 0.2],
                [4.4, 2.9, 1.4, 0.2],
                [4.9, 3.1, 1.5, 0.1],
                [5.4, 3.7, 1.5, 0.2],
                [4.8, 3.4, 1.6, 0.2],
                [4.8, 3. , 1.4, 0.1],
                [4.3, 3. , 1.1, 0.1],
                [5.8, 4. , 1.2, 0.2],
                [5.7, 4.4, 1.5, 0.4],
                [5.4, 3.9, 1.3, 0.4],
                [5.1, 3.5, 1.4, 0.3],
                [5.7, 3.8, 1.7, 0.3],
                [5.1, 3.8, 1.5, 0.3],
                [5.4, 3.4, 1.7, 0.2],
                [5.1, 3.7, 1.5, 0.4],
                [4.6, 3.6, 1. , 0.2],
                [5.1, 3.3, 1.7, 0.5],
                [4.8, 3.4, 1.9, 0.2],
                [5. , 3. , 1.6, 0.2],
                [5. , 3.4, 1.6, 0.4],
                [5.2, 3.5, 1.5, 0.2],
                [5.2, 3.4, 1.4, 0.2],
                [4.7, 3.2, 1.6, 0.2],
                [4.8, 3.1, 1.6, 0.2],
                [5.4, 3.4, 1.5, 0.4],
                [5.2, 4.1, 1.5, 0.1],
                [5.5, 4.2, 1.4, 0.2],
                [4.9, 3.1, 1.5, 0.2],
                [5. , 3.2, 1.2, 0.2],
                [5.5, 3.5, 1.3, 0.2],
                [4.9, 3.6, 1.4, 0.1],
                [4.4, 3. , 1.3, 0.2],
                [5.1, 3.4, 1.5, 0.2],
                [5. , 3.5, 1.3, 0.3],
                [4.5, 2.3, 1.3, 0.3],
                [4.4, 3.2, 1.3, 0.2],
                [5. , 3.5, 1.6, 0.6],
                [5.1, 3.8, 1.9, 0.4],
                [4.8, 3. , 1.4, 0.3],
                [5.1, 3.8, 1.6, 0.2],
                [4.6, 3.2, 1.4, 0.2],
                [5.3, 3.7, 1.5, 0.2],
                [5. , 3.3, 1.4, 0.2],
                [7. , 3.2, 4.7, 1.4],
                [6.4, 3.2, 4.5, 1.5],
                [6.9, 3.1, 4.9, 1.5],
                [5.5, 2.3, 4. , 1.3],
                [6.5, 2.8, 4.6, 1.5],
                [5.7, 2.8, 4.5, 1.3],
                [6.3, 3.3, 4.7, 1.6],
                [4.9, 2.4, 3.3, 1. ],
                [6.6, 2.9, 4.6, 1.3],
                [5.2, 2.7, 3.9, 1.4],
                [5. , 2. , 3.5, 1. ],
                [5.9, 3. , 4.2, 1.5],
                [6. , 2.2, 4. , 1. ],
                [6.1, 2.9, 4.7, 1.4],
                [5.6, 2.9, 3.6, 1.3],
                [6.7, 3.1, 4.4, 1.4],
                [5.6, 3. , 4.5, 1.5],
                [5.8, 2.7, 4.1, 1. ],
                [6.2, 2.2, 4.5, 1.5],
                [5.6, 2.5, 3.9, 1.1],
                [5.9, 3.2, 4.8, 1.8],
                [6.1, 2.8, 4. , 1.3],
                [6.3, 2.5, 4.9, 1.5],
                [6.1, 2.8, 4.7, 1.2],
                [6.4, 2.9, 4.3, 1.3],
```

In [2]:
```
# özellik isimlerini görelim
iris.feature_names
```

Out[2]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [3]:
```
type(iris)
```

Out[3]:
```
sklearn.utils.Bunch
```

In [4]:
```
iris.target
```

Out[4]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [5]:
```
iris.target_names
# 0-setosa, 1-versicolor, 2-virginica
```

Out[5]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [6]:
```
import pandas as pd
data = pd.DataFrame(iris.data)
```

In [7]:
```
data.head()
```

Out[7]:

	0	1	2	3
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [8]:
```
data.columns = iris.feature_names
```

In [9]:
```
data.head()
```

Out[9]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Bu eğitim [adresinden (https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcfb9)](https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcfb9) alınmıştır

Eksik verilerin Sklearn ile Çözülmesi

In [10]:
```
import numpy as np
import pandas as pd
X = pd.DataFrame(np.array([5,7,8, np.NaN, np.NaN, np.NaN, -5, 0,25,999,1,-1, np.NaN, 0, n
p.NaN]).reshape((5,3)))
X.columns = ['f1', 'f2', 'f3']
```

In [11]:
```
X
```

Out[11]:

	f1	f2	f3
0	5.0	7.0	8.0
1	NaN	NaN	NaN
2	-5.0	0.0	25.0
3	999.0	1.0	-1.0
4	NaN	0.0	NaN

axis: 0 satır için 1 sütun için
tresh: Kaç verinin tutulacağı
inplace: veri setinin güncellenmesi

In [12]:
```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values=np.nan, strategy='mean')
```

In [13]:
```
X_imputed = pd.DataFrame(imp.fit_transform(X), columns=["f1","f2","f3"])
```

In [14]:
```
X_imputed
```

Out[14]:

	f1	f2	f3
0	5.0	7.0	8.000000
1	333.0	2.0	10.666667
2	-5.0	0.0	25.000000
3	999.0	1.0	-1.000000
4	333.0	0.0	10.666667

Ordinal Verilerin Etiketlenmesi

In [15]:
```
X = pd.DataFrame(
    np.array([
        'M', 'O-', 'medium', 'M', 'O-', 'high', 'F', 'O+', 'high', 'F', 'AB',
        'low', 'F', 'B+', 'low'
    ]).reshape((5, 3)))
X.columns = ['gender', 'blood_type', 'edu_level']
```

In [16]:
```
X
```

Out[16]:

	gender	blood_type	edu_level
0	M	O-	medium
1	M	O-	high
2	F	O+	high
3	F	AB	low
4	F	B+	low

In [17]:
```
from sklearn.preprocessing import OrdinalEncoder
encoder = OrdinalEncoder() # Kategorileri otomatik bıraktık
X.edu_level = encoder.fit_transform(X.edu_level.values.reshape(-1, 1))
```

In [18]:

Out[18]:

	gender	blood_type	edu_level
0	M	O-	2.0
1	M	O-	0.0
2	F	O+	0.0
3	F	AB	1.0
4	F	B+	1.0

In [19]:
```
encoder.categories_
```

Out[19]:
```
[array(['high', 'low', 'medium'], dtype=object)]
```

Burada low: 1, medium: 2, high: 3 yazmasını istiyoruz.

In [20]:
```
X = pd.DataFrame(np.array(['M', 'O-', 'medium','M', 'O-', 'high','F', 'O+', 'high','F', '
AB', 'low','F', 'B+', 'low']).reshape((5,3)))
X.columns = ['gender', 'blood_type', 'edu_level']
```

In [21]:
```
edu_levels = {
            'low': 1,
            'medium': 2,
            'high': 3}
X['edu_level']=X['edu_level'].apply(lambda x:edu_levels[x])
```

In [22]:
```
X
```

Out[22]:

	gender	blood_type	edu_level
0	M	O-	2
1	M	O-	3
2	F	O+	3
3	F	AB	1
4	F	B+	1

Nominal verilerin encode edilmesi

one-hot-encoding

n adet kategorik özellikler n adet ikili özelliğe çevrilir.

In [23]:
```
from sklearn.preprocessing import OneHotEncoder

onehot = OneHotEncoder(dtype=np.int, sparse=True)
nominals = ['gender', 'blood_type']
```

In [24]:
```
X[nominals]
```

Out[24]:

	gender	blood_type
0	M	O-
1	M	O-
2	F	O+
3	F	AB
4	F	B+

In [25]:
```
onehot.fit_transform(X[nominals]).toarray() # to array ile dönüşümü diziye çeviriyoruz.
```

Out[25]:
```
array([[0, 1, 0, 0, 0, 1],
       [0, 1, 0, 0, 0, 1],
       [1, 0, 0, 0, 1, 0],
       [1, 0, 1, 0, 0, 0],
       [1, 0, 0, 1, 0, 0]])
```

In [26]:
```
onehot_X = pd.DataFrame(onehot.fit_transform(X[nominals]).toarray(),columns=['F', 'M', 'A
B', 'B+','O+', 'O-'])
```

In [27]:
```
onehot_X
```

Out[27]:

	F	M	AB	B+	O+	O-
0	0	1	0	0	0	1
1	0	1	0	0	0	1
2	1	0	0	0	1	0
3	1	0	1	0	0	0
4	1	0	0	1	0	0

In [31]:
```
X_imputed.f3
```

Out[31]:
```
0     8.000000
1    10.666667
2    25.000000
3    -1.000000
4    10.666667
Name: f3, dtype: float64
```

In [42]:
```
# Nümerik Verilerin ölçeklendirilmesi hale getirilmesi
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit_transform(X_imputed.f3.values.reshape(-1, 1))
```

Out[42]:
```
array([[-0.31933647],
       [ 0.        ],
       [ 1.71643352],
       [-1.39709705],
       [ 0.        ]])
```

In [40]:
```
ortalama=X_imputed.f3.mean()
standart_sapma = X_imputed.f3.std()
sigma = standart_sapma/(5**0.5)
```

In [48]:
```
scaler.mean_
```

Out[48]:
```
array([10.66666667])
```

In [41]:
```
(X_imputed.f3[0]-ortalama)/sigma
```

Out[41]:
```
-0.638672938997955
```

$$x_{scaled} = (x — u)/s$$

In [29]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(-3,3))
scaler.fit_transform(X_imputed.f3.values.reshape(-1, 1))
```

Out[29]:
```
array([[-0.92307692],
       [-0.30769231],
       [ 3.        ],
       [-3.        ],
       [-0.30769231]])
```

$$x_{scaled} = (x - min(x))/(max(x) - min(x))$$

In [30]:
```
from sklearn.preprocessing import MaxAbsScaler
scaler = MaxAbsScaler()
scaler.fit_transform(X_imputed.f3.values.reshape(-1, 1))
```

Out[30]:
```
array([[ 0.32      ],
       [ 0.42666667],
       [ 1.        ],
       [-0.04      ],
       [ 0.42666667]])
```

$$x_{scaled} = x/max(abs(x))$$

Kaynak (https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcfb9)

In [ ]: