```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os
os.chdir('Calisma_Dizniniz')
dataset = pd.read_csv('SosyalMedyaReklamKampanyası.csv')
```

Spyder’ın variable explorer penceresinden veri setimizi görelim:

dataset - DataFrame

Index	KullaniciID	Cinsiyet	Yas	TahminiMaas	SatinAldiMi
0	15624510	Erkek	19	19000	0
1	15810944	Erkek	35	20000	0
2	15668575	Kadın	26	43000	0
3	15603246	Kadın	27	57000	0
4	15804002	Erkek	19	76000	0
5	15728773	Erkek	27	58000	0
6	15598044	Kadın	27	84000	0
7	15694829	Kadın	32	150000	1
8	15600575	Erkek	25	33000	0
9	15727311	Kadın	35	65000	0
10	15570769	Kadın	26	80000	0
11	15606274	Kadın	26	52000	0
12	15746139	Erkek	20	86000	0
13	15704987	Erkek	32	18000	0
14	15628972	Erkek	18	82000	0
15	15697686	Erkek	29	80000	0
16	15733883	Erkek	47	25000	1
17	15617482	Erkek	45	26000	1
18	15704583	Erkek	46	28000	1
19	15621083	Kadın	48	29000	1
20	15649487	Erkek	45	22000	1
21	15736760	Kadın	47	49000	1

Format Resize ☑ Background color ☑ Column min/max OK Cancel

Veriyi Anlamak

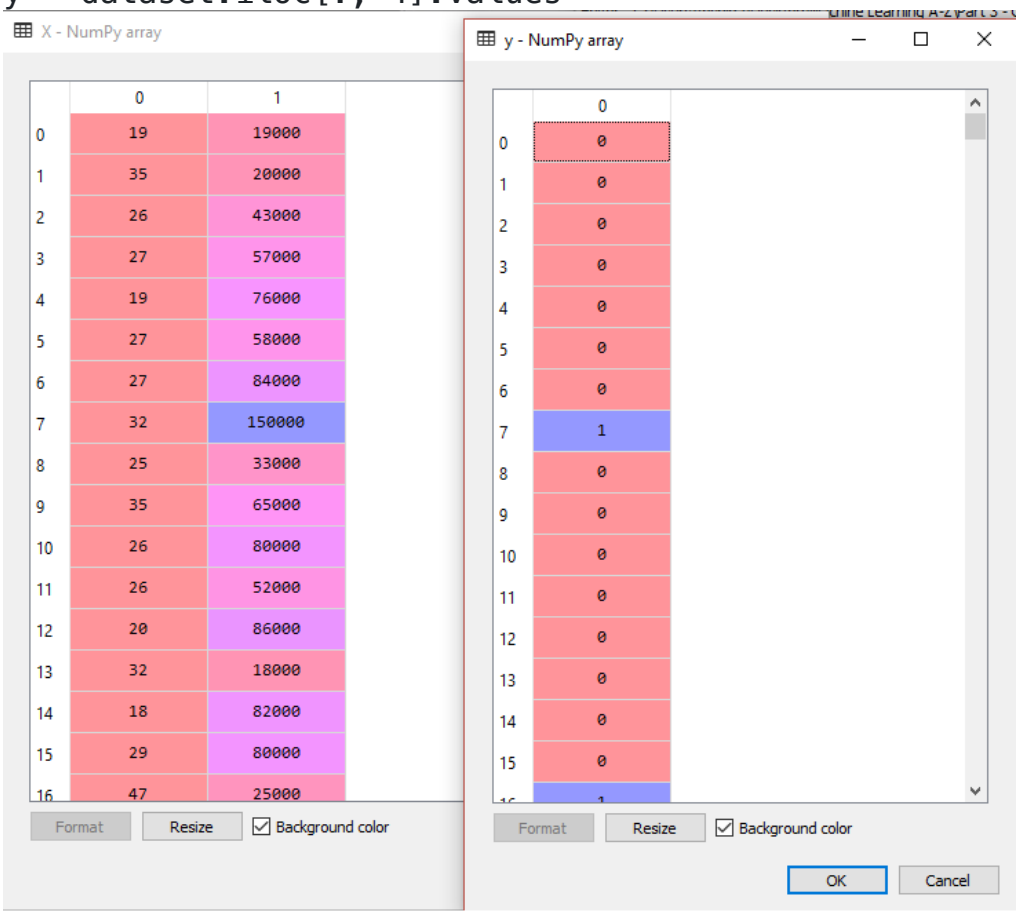
Yukarıda gördüğümüz veri seti beş nitelikten oluşuyor. Veri seti bir sosyal medya kayıtlarından derlenmiş durumda. KullaniciID müşteriyi belirleyen eşsiz rakam, Cinsiyet, Yaş, Tahmini Gelir yıllık tahmin edilen gelir, SatinAldiMi ise belirli bir ürünü satın almış olup olmadığı, hadi lüks araba diyelim. Bu veri setinde kolayca anlaşılabileceği gibi hedef değişkenimiz SatinAldiMi'dir. Diğer

dört nitelik ise bağımsız niteliklerdir. Bu bağımsız niteliklerle bağımlı nitelik (satın alma davranışının gerçekleşip gerçekleşmeyeceği) tahmin edilecek.

Veri Setini Bağımlı ve Bağımsız Niteliklere Ayırmak

Yukarıda gördüğümüz niteliklerden bağımsız değişken olarak sadece yaş ve tahmini maaşı kullanacağız.

```
X = dataset.iloc[:, [2,3]].values
y = dataset.iloc[:, 4].values
```



	0	1
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
5	27	58000
6	27	84000
7	32	150000
8	25	33000
9	35	65000
10	26	80000
11	26	52000
12	20	86000
13	32	18000
14	18	82000
15	29	80000
16	47	25000

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0

Veriyi Eğitim ve Test Olarak Ayırmak

Veri setinde 400 kayıt var bunun 300'ünü eğitim, 100'ünü test için ayıralım.

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.25, random_state = 0)
```

Feature Scaling

Bağımsız değişkenlerden yaş ile tahmini gelir aynı birimde olmadığı için feature scaling uygulayacağız.

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

K En Yakın Komşu Modeli Oluşturmak ve Eğitmek

Şimdi scikit-learn kütüphanesi neighbors modülü KNeighborClassifier sınıfından oluşturacağımız classifier nesnesi modelimiz oluşturalım.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5,
metric='minkowski', p = 2)
classifier.fit(X_train, y_train)
```

Sınıf parametrelerinden biraz bahsedelim. n_neighbors kullanılacak komşu sayısı. metric ise komşuların yakınlığını belirlemede hangi yöntemi kullanacağımız. mesafeye dayalı yöntem kullanacak isek minkowski seçiyoruz. p ise hangi mesafe yöntemini k kullanacağımız, 2 öklid mesafesini kullan demektir.

Test Seti ile Tahmin Yapmak

Ayırdığımız test setimizi (X_test) kullanarak oluşturduğumuz model ile tahmin yapalım ve elde ettiğimiz set (y_pred) ile hedef değişken (y_test) test setimizi karşılaştıralım.

```
y_pred = classifier.predict(X_test)
```

y_test - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

y_pred - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	1
10	0
11	0
12	0

Yukarıda y_test (gerçek veri) ile modelin tahmin ettiği y_pred bir görüntü bulunuyor. Örneğin 9. indekste bulunan müşteriye baktığımızda gerçekte satın alma gerçekleşmemiş iken model satın alır demiş, yani yanlış sınıflandırma yapmış. Şimdi kaç tane doğru kaç tane yanlış sınıflandırma

olmuş bir bakalım.

Hata Matrisini Oluşturma

Yaptığımız sınıflandırmanın doğruluğunu kontrol etme yöntemlerinden birisi de hata matrisi oluşturmaktır. Hata matrisi için scikit-learn kütüphanesi metrics modülü confusion_matrix fonksiyonunu kullanıyoruz.

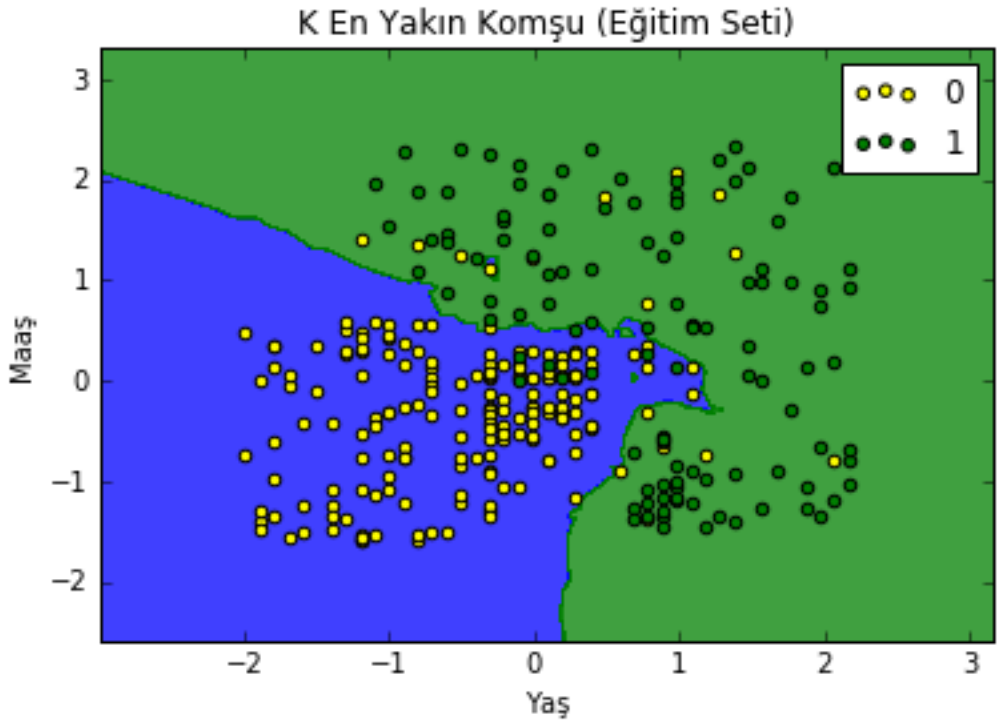
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[64 4]

[ 3 29]]
```

Bildiğiniz gibi 100 kayıtlık test verisi ayırmıştık. Yukarıda gördüğümüz hata matrisine göre 7 kayıt yanlış sınıflandırılmış, 93 kayıt doğru sınıflandırılmış. Grafiğimizi görelim:

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('blue',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
     plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                 c = ListedColormap(('yellow', 'green'))(i),
label = j)
plt.title('K En Yakın Komşu (Eğitim Seti)')
plt.xlabel('Yaş')
plt.ylabel('Maaş')
plt.legend()
plt.show()
```



Şimdi grafiğimizi test setleri için çizelim. Bunun için yukarıdaki kodda veri setlerini ve etiket bilgilerini değiştirmek yeterli olur.

Yanlış sınıflandırılan 7 noktayı buradan sayabiliriz.