İSTANBUL TİCARET ÜNİVERSİTESİ

İstanbul Ticaret Üniversitesi

2023-2024 Güz Dönemi

ENM 341

İŞ ANALİTİĞİ

Proje Ara Raporu 3

Hazırlayan;
Melih Sağırlı
Ömercan Korkmaz
İrem Uçmak

Karar Ağacı

```
: import numpy as np
  import matplotlib.pyplot as plt
  import pandas as pd
  import os
  os.chdir('/Users/omercankorkmaz/Desktop/Endüstri')
  dataset = pd.read_csv('SosyalMedyaReklamKampanyası.csv')
```

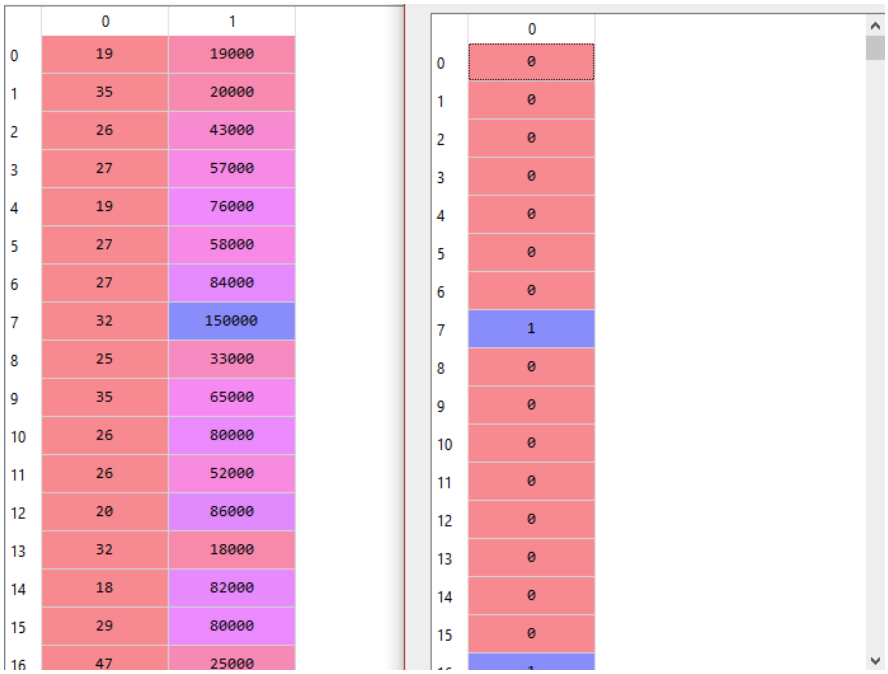
Index	KullaniciID	Cinsiyet	Yas	TahminiMaas	SatinAldiMi
0	15624510	Erkek	19	19000	0
1	15810944	Erkek	35	20000	0
2	15668575	Kadın	26	43000	0
3	15603246	Kadın	27	57000	0
4	15804002	Erkek	19	76000	0
5	15728773	Erkek	27	58000	0
6	15598044	Kadın	27	84000	0
7	15694829	Kadın	32	150000	1
8	15600575	Erkek	25	33000	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

Veri seti beş nitelikten oluşuyor. Veri seti bir sosyal medya kayıtlarından derlenmiş durumda. KullaniciID, Cinsiyet, Yaş, Tahmini Gelir yıllık tahmin edilen gelir, SatinAldiMi ise belirli bir ürünü satın almış olup olmadığı. Bu veri setinde kolayca anlaşılabileceği gibi hedef değişkenimiz SatinAldiMi'dir. Diğer dört nitelik ise bağımsız niteliklerdir. Bu bağımsız niteliklerle bağımlı nitelik tahmin edilecek.

Veri Setini Bağımlı ve Bağımsız Niteliklere Ayırmak

Gördüğümüz niteliklerden bağımsız değişken olarak sadece yaş ve tahmini maaşı kullandık.

```
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 setindeki 400 kayıtın 300'ünü eğitim, 100'ünü test için ayırdık.

```
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)
```

Normalizasyon – Feature Scaling

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

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

Karar Ağacı Modeli Oluşturmak ve Eğitmek

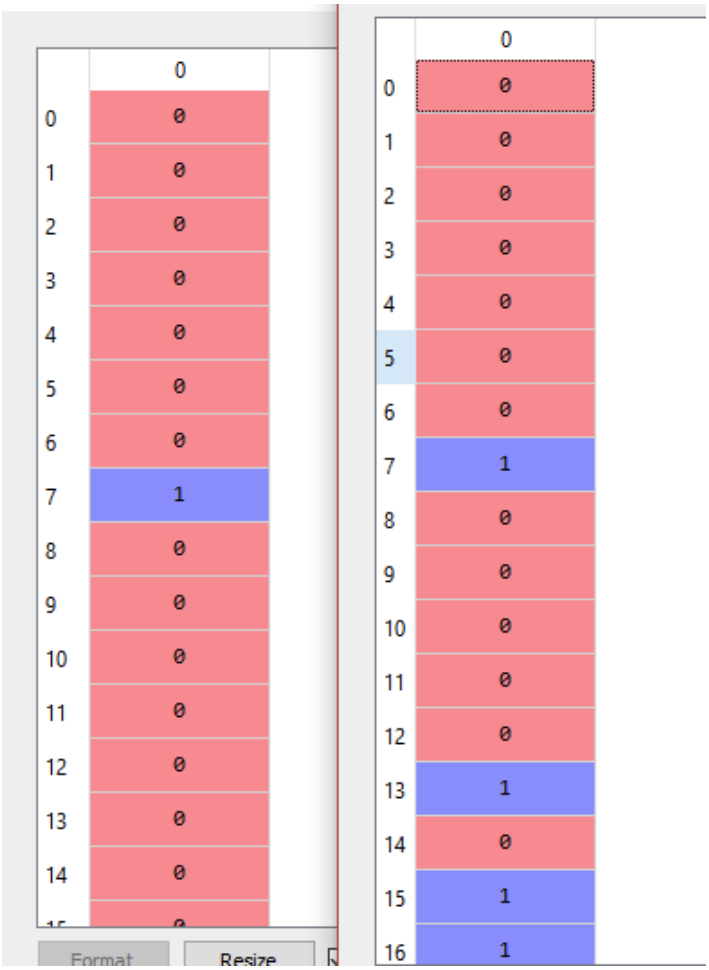
Şimdi scikit-learn kütüphanesi tree modülü DecisionTreeClassifier sınıfından yaratacağımız classifier nesnesi ile modelimizi oluşturduk. İlk parametrede ağaç oluşturma kriteri olarak entropi seçtik.

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state=0)
classifier.fit(X_train, y_train)
```

Test Seti ile Tahmin Yapmak

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

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



Solda gerçek, sağda ise tahmin değerleri bulunuyor. 13 indeksli kayıt satın almamış iken satın aldı diye sınıflandırılmış. Yani yanlışa doğru demiş, false positive.

Hata Matrisini Oluşturma

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

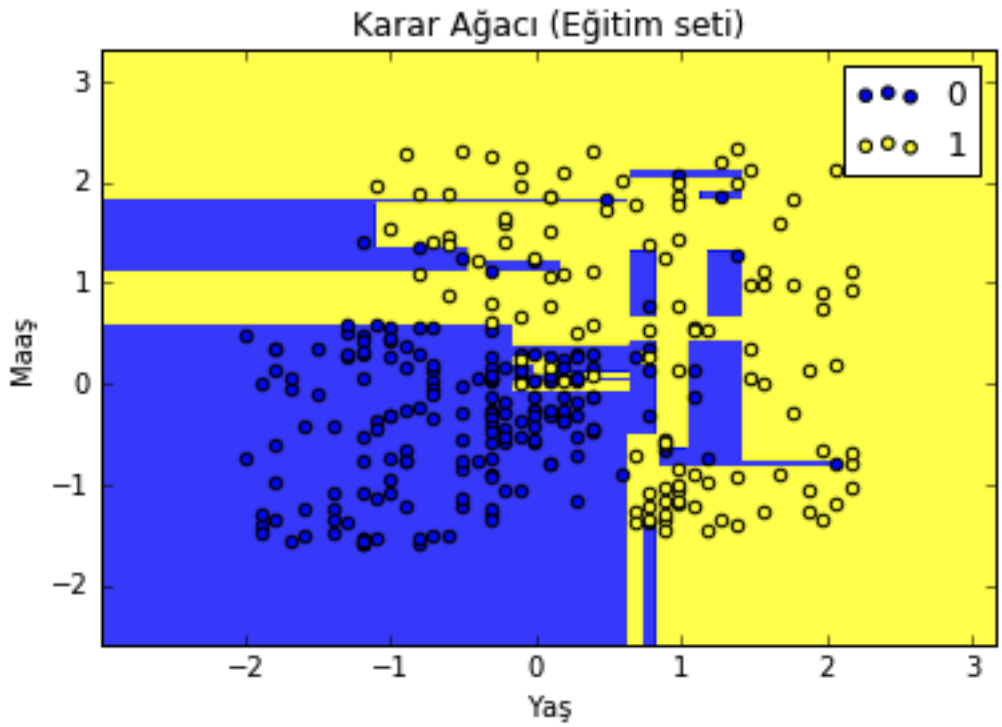
Çıktı:

```
[[62  6]
 [ 3 29]]
```

Matriste gördüğümüz gibi 9 adet hatalı sınıflandırma var.

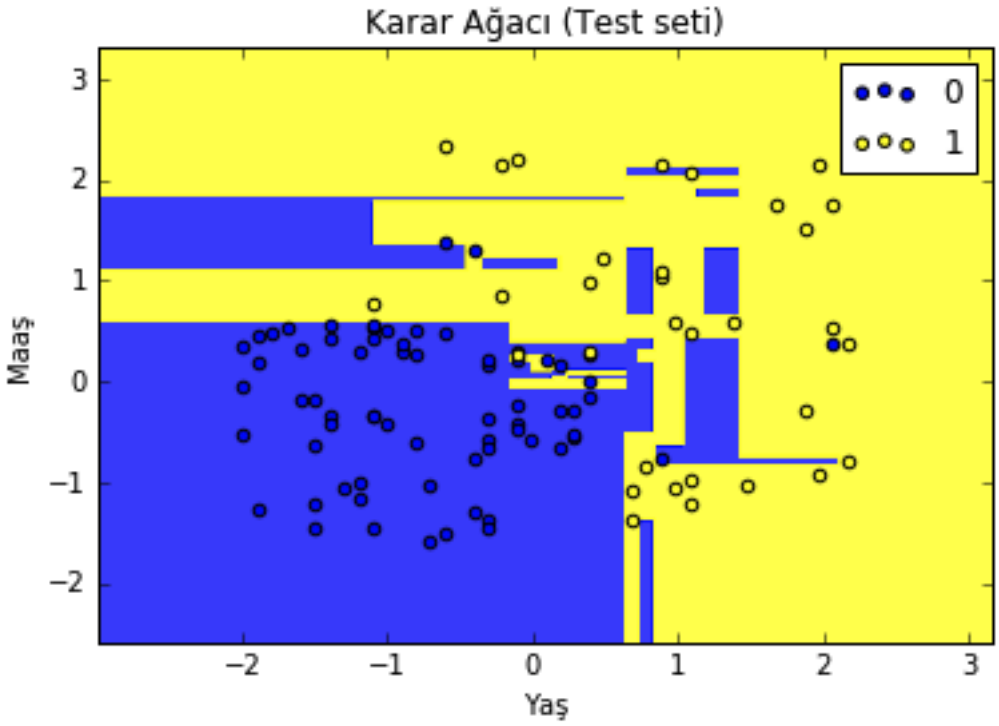
Eğitim Seti İçin Grafik

```
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', 'yellow')))
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(('blue', 'yellow'))(i), label = j)
plt.title('Karar Ağacı (Eğitim seti)')
plt.xlabel('Yaş')
plt.ylabel('Maaş')
plt.legend()
plt.show()
```



Test Seti İçin Grafik

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
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', 'yellow')))
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(('blue', 'yellow'))(i), label = j)
plt.title('Karar Ağacı (Test seti)')
plt.xlabel('Yaş')
plt.ylabel('Maaş')
plt.legend()
plt.show()
```



9 tane hatalı sınıflandırma vardı. Bunlar: Mavi bölgede 3 tane sarı, sarı bölgede 7 tane mavi var.

```
: import numpy as np
  import matplotlib.pyplot as plt
  import pandas as pd
  import os
  os.chdir('/Users/omercankorkmaz/Desktop/Endüstri')
  dataset = pd.read_csv('SosyalMedyaReklamKampanyası.csv')
```

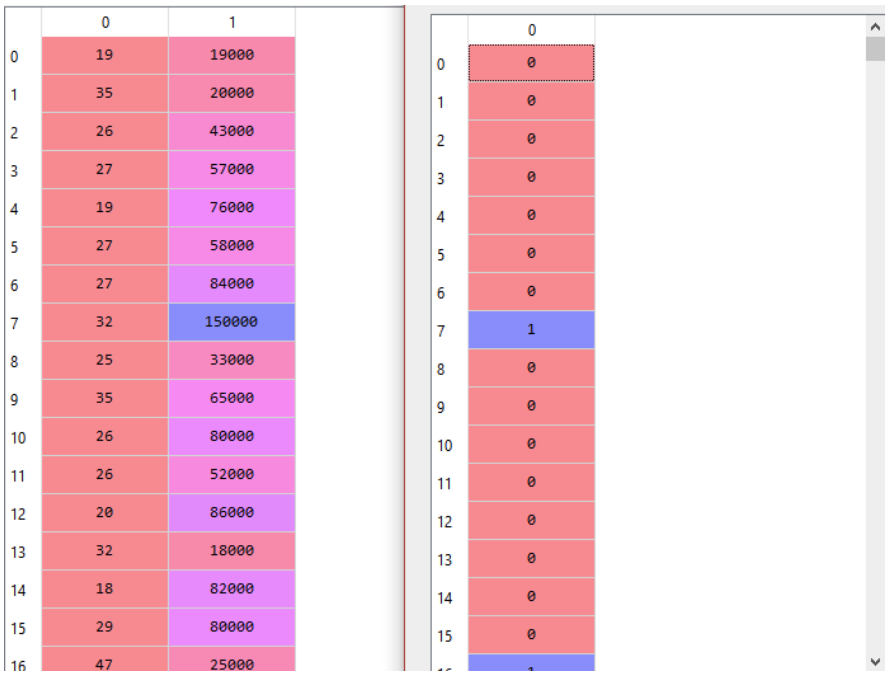
Index	KullaniciID	Cinsiyet	Yas	TahminiMaas	SatinAldiMi
0	15624510	Erkek	19	19000	0
1	15810944	Erkek	35	20000	0
2	15668575	Kadın	26	43000	0
3	15603246	Kadın	27	57000	0
4	15804002	Erkek	19	76000	0
5	15728773	Erkek	27	58000	0
6	15598044	Kadın	27	84000	0
7	15694829	Kadın	32	150000	1
8	15600575	Erkek	25	33000	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

Veri seti beş nitelikten oluşuyor. Veri seti bir sosyal medya kayıtlarından derlenmiş durumda. KullaniciID, Cinsiyet, Yaş, Tahmini Gelir yıllık tahmin edilen gelir, SatinAldiMi ise belirli bir ürünü satın almış olup olmadığı. Bu veri setinde kolayca anlaşılabileceği gibi hedef değişkenimiz SatinAldiMi'dir. Diğer dört nitelik ise bağımsız niteliklerdir. Bu bağımsız niteliklerle bağımlı nitelik tahmin edilecek.

Veri Setini Bağımlı ve Bağımsız Niteliklere Ayırmak

Gördüğümüz niteliklerden bağımsız değişken olarak sadece yaş ve tahmini maaşı kullandık.

```
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

Veri Setini Test ve Eğitim Olarak Ayırma

Veriyi eğitim ve test olarak ayırmak için scikit-learn kütüphanesi `cross_validation` modülü içindeki `train_test_split` fonksiyonunu dahil ettik.

```
from sklearn.cross_validation import train_test_split
```

Yukarıda veri setimizi X ve y olarak niteliklerden (sütun bazlı bölme) ayırdık. Şimdi hem X hem de y'i satır bazlı ikiye böldük. Satırların % 80'i eğitim, % 20'si test için ayırdık. Aşağıdaki kodlarla X'ten rastgele seçilecek %80 satırı `X_train`'e; kalan %20 ise `X_test`'e atandı. Aynı şekilde y'den rastgele seçilecek %80 satırı `y_train`'e; kalan %20 ise `y_test`'e atandı. Bu seçim işlemi hepsinde aynı satırlar seçilecek şekilde yapılır.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

Karar Ağacı Modelini Eğitme

Sınıflandırma için karar ağacı kullandık. Karar Ağacı modeli için scikit-learn kütüphanesi `tree` modülünden `DecisionTreeClassifier` sınıfını dahil ettik.

```
from sklearn.tree import DecisionTreeClassifier
```

Dahil ettiğimiz sınıftan bir nesne oluşturduk, sınıfın bir metodu olan `fit()` ile nesnemizi eğittik.

```
decisionTreeObject = DecisionTreeClassifier()
decisionTreeObject.fit(X_train,y_train)
```

Karar Ağacı Modelini Test Etme

Karar Ağacı modelimizi eğitim için ayırdığımız `X_train` ve `y_train` setleriyle eğittik. Test için ayırdığımız veriler ile doğruluk skorlarını hesapladık. Bunun için `score()` metodunu kullandık.

```
dt_test_sonuc = decisionTreeObject.score(X_test, y_test)
print("Karar Ağacı Doğruluk (test_seti): ",round(dt_test_sonuc,2))
```

Yukarıdaki kodun çıktısı:

```
Karar Ağacı Doğruluk (test_seti):  0.91
```

Doğruluk 0.91 gibi yüksek bir değer.

Random Forest Modeli Oluşturma

Scikit-learn kütüphanesi, `ensemble` modülünden `RandomForestClassifier` sınıfını dahil ettik.

```
from sklearn.ensemble import RandomForestClassifier
```

Sınıf nesnemizi oluşturup, fit metodu ile nesnemizi eğittik.

```
randomForestObject = RandomForestClassifier(n_estimators=10)
randomForestObject.fit(X_train, y_train)
```

Test sonuçlarını hesapladık:

```
df_test_sonuc = randomForestObject.score(X_test, y_test)
print("Random Forest Doğruluk (test_seti): ",round(df_test_sonuc,2))
```

Yukarıdaki kodların çıktısı:

Random Forest Doğruluk (test_seti): 0.94

Random Forest daha yüksek doğruluk değerini yakaladı.

Bagging

Scikit-learn kütüphanesi, `ensemble` modülünden `BaggingClassifier` sınıfını dahil ettik:

```
from sklearn.ensemble import BaggingClassifier
```

`BaggingClassifier` sınıfından yarattığımız modeli eğittik.

```
baggingObject = BaggingClassifier(DecisionTreeClassifier(), max_samples=0.5, max_features=1.0, n_estimators=20)
baggingObject.fit(X_train, y_train)
```

Bagging modelinin test sonuçları:

```
baggingObject_sonuc = baggingObject.score(X_test, y_test)
print("Bagging Doğruluk (test_seti): ", round(baggingObject_sonuc,2))
```

Bagging Doğruluk (test_seti): 0.92

Bagging sonucu(0.92) da yüksek de olsa tek bir karar ağacından(0.91) daha yüksek çıktı.