Hazırlayan: Özgür Durak

Öğrenci No: 170101012

Proje Türü: Classfication

Proje Amacı: Bir kişinin yılda 50.000 'den fazla kazanıp kazanmadığını tahmin eden sınıflama

1.Proje hakkında

Proje insanların gelirlerini , eğitimleri ,meslekleri, ülkeleri gibi kriterlerin bulunduğu veri setini kullanarak bir kişinin yıllık kazancının 50.000 'in üzerinde olup olmadığı sınıflandırma problemidir.

2. Makine Öğrenmesi ve Projeye Giriş

Makine öğrenmesi, insanların öğrenme şekillerini taklit etmek için veri ve algoritmaların kullanımına odaklanıp doğruluğunu kademeli olarak artıran bir yapay zeka(AI) ve bilgisayar bilimi dalıdır.Genel olarak, makine öğrenmesi algoritmaları bir tahmin ya da sınıflandırma yapmak için kullanılır.

Bir makine öğrenmesi metodu tahminde bulunmak için bir çıktı üretir. Bu çıktı kategorik ise **sınıflandırma** (classification) ve eğer nümerik ise **regresyon** (regression) denir. Açıklayıcı bir modelleme olan **kümeleme** (Clustering) ise benzer gözlemleri aynı kümelere atama işlemidir.

Gözetimli Öğrenme

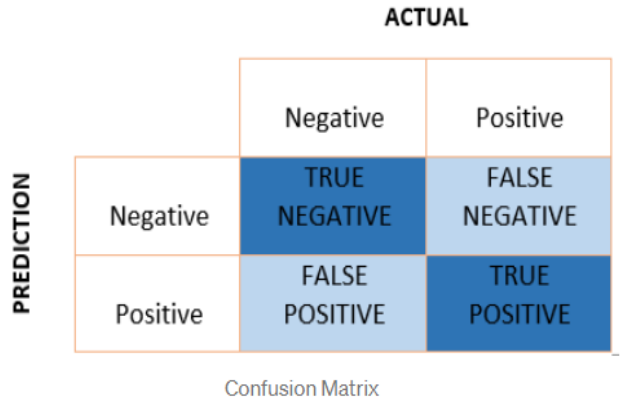
Eğitim verileri üzerinden bir fonksiyon üreten bir makine öğrenmesi tekniğidir. Başka bir deyişle, bu öğrenme tekniğinde girdilerle (işaretlenmiş veri – labelled data) ile istenen çıktılar arasında eşleme yapan bir fonksiyon üretir. Eğitim verisi hem girdilerden

hem çıktılardan oluşur. Fonksiyon, sınıflandırma (classifiction) veya eğri uydurma (regression) algoritmaları ile belirlenebilir.

Bu yöntemde işaretlenmemiş (unlabelled) veri üzerinden bilinmeyen bir yapıyı tahmin etmek için bir fonksiyon kullanan makine öğrenmesi tekniğidir. Burada girdi verisinin hangi sınıfa ait olduğu belirsizdir.

Karmaşıklık matrisi

Bir sınıflama algoritmasının performasını özetlemek için kullanılan bir tekniktir. Karmaşıklık matrisi hesaplamak Sınıflama Modelinizin neleri doğru ve hatalı yaptığını anlamada yardımcı olur.



Confusion Matrix

$$TPR = \frac{TP}{Actual\ Positive} = \frac{TP}{TP + FN}$$

$$FNR = \frac{FN}{Actual\ Positive} = \frac{FN}{TP + FN}$$

$$TNR = \frac{TN}{Actual\ Negative} = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{Actual\ Negative} = \frac{FP}{TN + FP}$$

Accuracy = Doğru Tahminlerin Sayısı / Tüm Tahminlerin Sayısı

Precision = Pozitif olarak tahmin edilenlerin gerçekte kaçta kaçı doğru.

Specifiy = Negatif olarak tahmin edilenlerin gerçekte kaçta kaçı negatif olduğu tahmin ediliyor.

Recall = Toplam pozitifin yüzde kaçının pozitif olduğu tahmin ediliyor.

F1 Score,Precision ve Recall değerlerinin ağırlıklı (harmonik) ortalamasıdır.

Performance Metrics

Confusion Matrix

		Predicted Class	
		Positive	Negative
Actual Class	Positive	**True positive (TP)**	**False negative (FN)**
	Negative	**False positive (FP)**	**True negative (TN)**

- **Accuracy** = (Σ True positive + Σ True negative) / total
- **Recall (Sensitivity, True positive rate)** = Σ True positive / Σ condition positive
- **Specificity (True negative rate)** = Σ True negative / Σ condition negative
- **Precision** = Σ True positive / Σ Classifier outcome positive
- **F1-score** = 2 x Precision x Recall / (Precision + Recall)
- **Matthews correlation coefficient**
- **Receiver operating characteristic (ROC) curve**
- **Area under ROC curve (AUC)**

3. Veri Seti

```
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1    workclass      32561 non-null  object
 2    fnlwgt         32561 non-null  int64
 3    education      32561 non-null  object
 4    education-num  32561 non-null  int64
 5    marital-status 32561 non-null  object
 6    occupation     32561 non-null  object
 7    relationship   32561 non-null  object
 8    race           32561 non-null  object
 9    sex            32561 non-null  object
 10   capital-gain   32561 non-null  int64
 11   capital-loss   32561 non-null  int64
 12   hours-per-week 32561 non-null  int64
 13   native-country 32561 non-null  object
 14   income         32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

age: continuous.
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
fnlwgt: continuous.
education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
education-num: continuous.
marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex: Female, Male.
capital-gain: continuous.
capital-loss: continuous.
hours-per-week: continuous.
native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

4. Veri Seti Ön İşleme(Preprocessing)

Veri setimizdeki numerik olmayan yani string değerleri Sklearn kütüphanesinden Label Encoder fonksiyonu ile numerik(sayısal)değerleri dönüştürdük. Bu sayede veri setimizdeki bütün değerler integer olarak tutuluyor.

Makine öğrenmesi modelleri kategorik değişkenleri algılayamadığı için ‘object’ tipindeki değişkenleri numeric(int) tipindeki değişkenlere dönüştürdük

```
#Convert string values to integer
le = LabelEncoder()
data["workclass"] = le.fit_transform(data["workclass"])
data["education"] = le.fit_transform(data["education"])
data["marital-status"] = le.fit_transform(data["marital-status"])
data["occupation"] = le.fit_transform(data["occupation"])
data["relationship"] = le.fit_transform(data["relationship"])
data["race"] = le.fit_transform(data["race"])
data["sex"] = le.fit_transform(data["sex"])
data["native-country"] = le.fit_transform(data["native-country"])
data["class"] = le.fit_transform(data["class"])

data.head()
```

✓ 0.1s

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	class
0	39	7	77516	9	13	4	1	1	4	1	2174	0	40	39	0
1	50	6	83311	9	13	2	4	0	4	1	0	0	13	39	0
2	38	4	215646	11	9	0	6	1	4	1	0	0	40	39	0
3	53	4	234721	1	7	2	6	0	2	1	0	0	40	39	0
4	28	4	338409	9	13	2	10	5	2	0	0	0	40	5	0

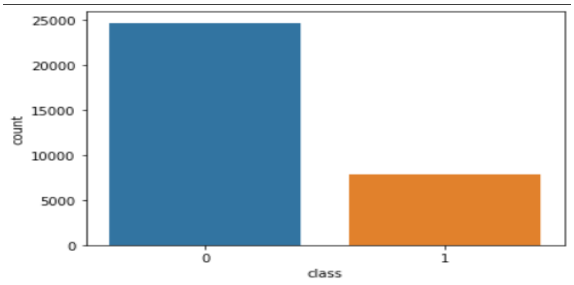
Veri setimizdeki boş(null) olan değerleri kontrol ettim ve toplamını ekrana yazdırdım.

```
#Checking for missing values
print(data.isnull().sum())

#Data Visualization of Target(class) Variables
sns.countplot(x='class',data =data)
```

✓ 0.1s

```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
class             0
dtype: int64
<AxesSubplot:xlabel='class', ylabel='count'>
```

Hedef değişkenlerinin verisinin görselleştirilmesi .Veri setindeki 50.000 üzeri maaş alanlar için '1' , alamayanlar için '0' seklinde grafik çıktısı.



Veri setinin string değerleri ile işlemlerin ardından verinin daha okunabilir hale gelmesi için normalizyon ile özellik aralığı(feature_range) (1,0) aralığına sınırlandırılmıştır.

```
#Normalize
from sklearn.preprocessing import LabelEncoder, minmax_scale

for column in data.columns.values:
    data[column] = minmax_scale(data[column], feature_range=(0,1), axis=0)

data.head()
```

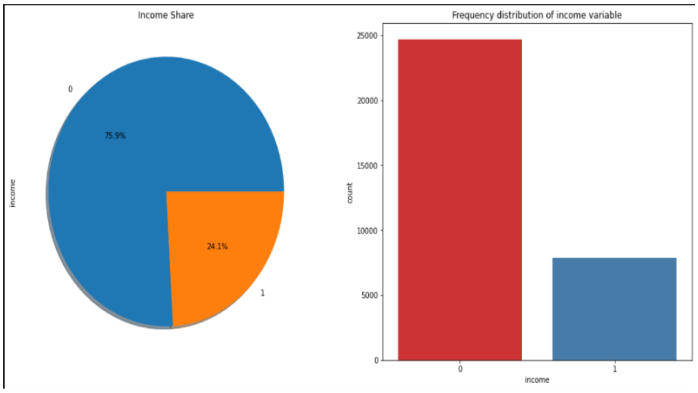
✓ 0.1s

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	0.301370	0.875	0.044302	0.600000	0.800000	0.666667	0.071429	0.2	1.0	1.0	0.02174	0.0	0.397959	0.951220	0.0
1	0.452055	0.750	0.048238	0.600000	0.800000	0.333333	0.285714	0.0	1.0	1.0	0.00000	0.0	0.122449	0.951220	0.0
2	0.287671	0.500	0.138113	0.733333	0.533333	0.000000	0.428571	0.2	1.0	1.0	0.00000	0.0	0.397959	0.951220	0.0
3	0.493151	0.500	0.151068	0.066667	0.400000	0.333333	0.428571	0.0	0.5	1.0	0.00000	0.0	0.397959	0.951220	0.0
4	0.150685	0.500	0.221488	0.600000	0.800000	0.333333	0.714286	1.0	0.5	0.0	0.00000	0.0	0.397959	0.121951	0.0

Burada income(gelir) verimizin frekans dağılımını tablolaştırdık.

```
f,ax=plt.subplots(1,2,figsize=(18,8))

ax[0] = data['income'].value_counts().plot.pie(explode=[0,0],autopct='%1.1f%%',ax=ax[0],shadow=True)
ax[0].set_title('Income Share')

#f, ax = plt.subplots(figsize=(6, 8))
ax[1] = sns.countplot(x="income", data=data, palette="Set1")
ax[1].set_title("Frequency distribution of income variable")
```

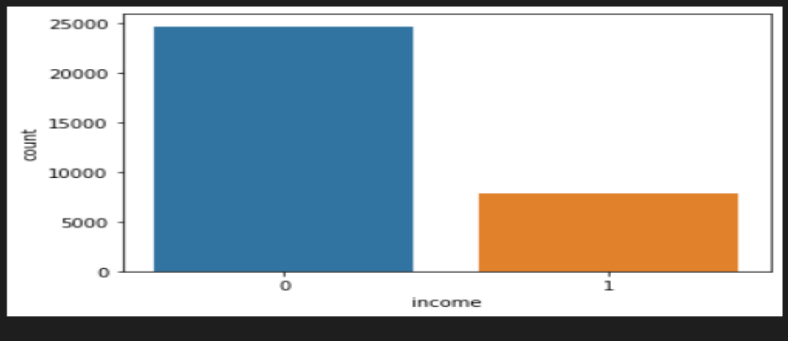
```
Text(0.5, 1.0, 'Frequency distribution of income variable')
```



İncome 0 ve 1 değeleri için kontrol edilmiştir. bu grafikte görüldüğü gibi iki income classı arasında büyük veri farkı bulunmaktadır. Böyle bir durumda model 0 income yönelimli olacağından bu fark düşürülmektedir. Bundan dolayı verilerin eşit sayıya getirilmesi 24720 adet bulunan 0 income 7841 adete rastgele olarak düşürülmüştür.(random under sampling)

```
#Data Visualization of Target(income) Variables
sns.countplot(x="income",data=data)

data.income.value_counts()
```

```
0    24720
1     7841
Name: income, dtype: int64
```



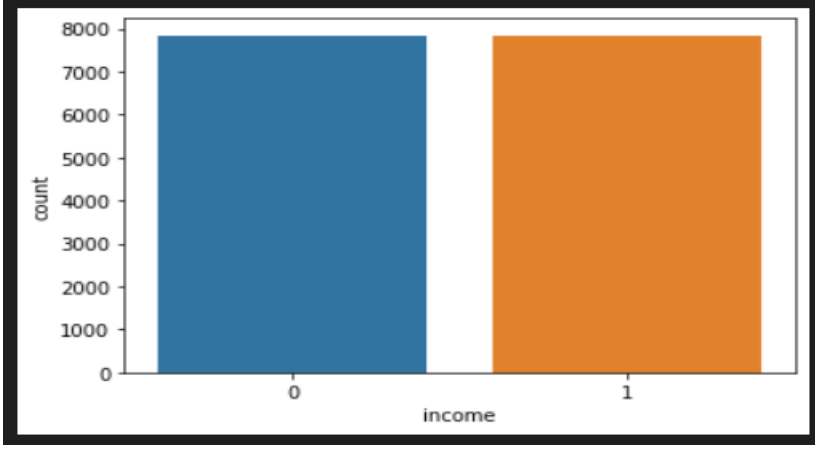
```
#Random Under Sampling

data = data.sample(frac=1)
income_1 = data.loc[data['income'] == 1]
income_0 = data.loc[data["income"] == 0][:7841]
dist= pd.concat([income_1, income_0])
```

Burada shuffle ile verilerimizi karıştırdık

```
from sklearn.utils import shuffle

#Shuffle
data_frame = shuffle(dist,random_state=42)
data = data_frame

#Data Visualization of Target(income) Variables
sns.countplot(x="income",data=data)
data.income.value_counts()
```

Under Sampling sonucu

```
0    7841
1    7841
Name: income, dtype: int64
```



Train ve Test Verilerinin Hazırlanması

Model içerisinde kullanmak amacıyla verinin 2 ayrı katmana bölünmesi gerekmektedir.

Train: Veri setindeki bu katman modelin eğitilmesi amacıyla kullanılır.

Test: Oluşturulmuş olan modelin performansının tespit edilmesi için ayrılan alandır. Bu bölümde modele verilerek tahmin edilen değerlerin, olması gereken değerler ile karşılaştırılması yapılmaktadır.

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# bağımlı ve bağımsız değişkenlerimizi oluşturduk
X = data.drop(["income"],axis=1)
y = data["income"]

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=0)
X_train.shape, X_test.shape, y_train.shape, y_test.shape

print("Train size: ",len(X_train))
print("Test size:", len(X_test))
```

```
Train size:  22792
Test size: 9769
```

Bu model içerisinde kullanmak amacıyla veri seti %30 test ve %70 train için bölünmüştür. Train ve test boyutlarının çıktısını yukarıda görüldüğü gibidir.

4. Makine Öğrenmesi Sınıflama Algoritmaları ve Sonuçlar

K-NN *(K-Nearest Neighbor)* algoritması en basit ve en çok kullanılan sınıflandırma algoritmasından biridir. K-NN non-parametric (parametrik olmayan), lazy (tembel) bir öğrenme algoritmasıdır. *lazy* kavramını anlamaya çalışırsak *eager learning* aksine lazy learning'in bir eğitim aşaması yoktur. Eğitim verilerini öğrenmez, bunun yerine eğitim veri kümesini *"ezberler"*. Bir tahmin yapmak istediğimizde, tüm veri setinde en yakın komşuları arar.

Algoritmanın çalışmasında bir *K* değeri belirlenir. Bu K değerinin anlamı bakılacak eleman sayısıdır. Bir değer geldiğinde en yakın K kadar eleman alınarak gelen değer arasındaki uzaklık hesaplanır. Uzaklık hesaplama işleminde genelde Öklid fonksiyonu kullanılır. Öklid fonksiyonuna alternatif olarak Manhattan, Minkowski ve Hamming fonksiyonlarıda kullanılabilir. Uzaklık hesaplandıktan sonra sıralanır ve gelen değer uygun olan sınıfa atanır.

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

print("Accuracy:\n", accuracy_score(y_test, y_pred_rf))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
print("\nClassification Report:\n", classification_report(y_test, y_pred_rf))
```

```
Accuracy:
 0.8520831200737026

Confusion Matrix:
 [[6934  473]
 [ 972 1390]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.94      0.91      7407
           1       0.75      0.59      0.66      2362

    accuracy                           0.85      9769
   macro avg       0.81      0.76      0.78      9769
weighted avg       0.85      0.85      0.85      9769
```

Naive Bayes Classifier

Naive Bayes sınıflandırıcısı Bayes teoreminin bağımsızlık önermesiyle basitleştirilmiş halidir. 1812 yılında Thomas Bayes tarafından bulunan koşullu olasılık hesaplama formülüdür. **Bayes teoremi**, olasılık kuramı içinde incelenen önemli bir konudur. Bu **teorem** bir rassal değişken için olasılık dağılımı içinde koşullu olasılıklar ile marjinal olasılıklar arasındaki ilişkiyi gösterir.

Bayes teoremi aşağıdaki denklemle ifade edilir;

P(*A*|*B*) ; B olayı gerçekleştiği durumda A olayının meydana gelme olasılığıdır (bakınız koşullu olasılık)
P(*B*|*A*) ; A olayı gerçekleştiği durumda B olayının meydana gelme olasılığıdır
P(A) ve P(B) ; A ve B olaylarının önsel olasılıklarıdır.

Naive Bayes sınıflandırıcısının temeli Bayes teoremine dayanır. **lazy** (tembel) bir öğrenme algoritmasıdır aynı zamanda dengesiz veri kümelerinde de çalışabilir. Algoritmanın çalışma şekli bir eleman için her durumun olasılığını hesaplar ve olasılık değeri en yüksek olana göre sınıflandırır. Az bir eğitim verisiyle çok başarılı işler çıkartabilir. Test kümesindeki bir değerin eğitim kümesinde gözlemlenemeyen bir değeri varsa olasılık değeri olarak 0 verir yani tahmin yapamaz.

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
gnb=GaussianNB()
y_pred = gnb.fit(X_train,y_train).predict(X_test)

print("Number of mislabeled points out of a total %d points : %d"% (X_test.shape[0], (y_test != y_pred).sum()))
print("Accuracy:\n", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Number of mislabeled points out of a total 9769 points : 1980
Accuracy:
 0.7973180468829972

Confusion Matrix:
 [[7039  368]
 [1612  750]]

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.95      0.88      7407
           1       0.67      0.32      0.43      2362

    accuracy                           0.80      9769
   macro avg       0.74      0.63      0.65      9769
weighted avg       0.78      0.80      0.77      9769
```

Random Forest Classfier

Rastgele Orman algoritması denetimli bir sınıflandırma algoritmasıdır. (Supervised classification algorithm).Fakat geleneksel yöntemlerden biri olan karar ağaçlarının en büyük problemlerinden biri aşırı öğrenme-veriyi ezberlemedir (overfitting). Rassal orman modeli bu problemi çözmek için hem veri setinden hem de öznitelik setinden rassal olarak 10'larca 100'lerce farklı alt-setler seçiyor ve bunları eğitiyor. Bu yöntemle 100'lerce karar ağacı oluşturuluyor ve her bir karar ağacı bireysel olarak tahminde bulunuyor.
Random forest modelinin diğer bir özelliği bize özniteliklerin ne kadar önemli olduğunu vermesi. (Bir özniteliğin önemli olması demek o özniteliğin bağımlı değişkendeki varyansın açıklanmasına ne kadar katkı yaptığıyla alakalı.)

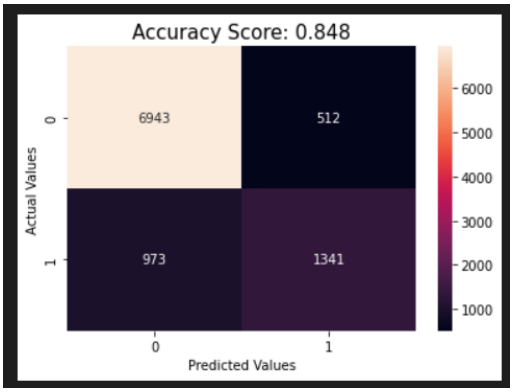
n_estimators : Bu, maksimum oylama veya tahmin ortalamalarını almadan önce oluşturmak

istediğiniz ağaç sayısıdır. Daha fazla sayıda ağaç size daha iyi performans sağlar ancak

kodunuzu yavaşlatır.

```
    rf = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
    rf.fit(X_train, y_train)
    y_pred_rf = rf.predict(X_test)
    print("Accuracy:\n", accuracy_score(y_test, y_pred_rf))
    print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
    print("\nClassification Report:\n", classification_report(y_test, y_pred_rf))
```

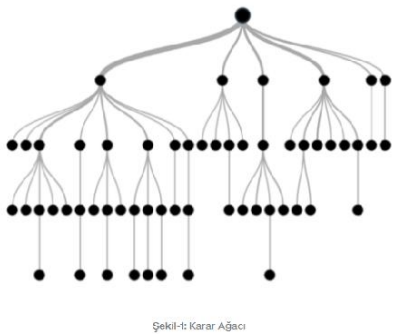
```
Accuracy:
 0.8520831200737026

Confusion Matrix:
 [[6934  473]
 [ 972 1390]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.94      0.91      7407
           1       0.75      0.59      0.66      2362

    accuracy                           0.85      9769
   macro avg       0.81      0.76      0.78      9769
weighted avg       0.85      0.85      0.85      9769
```



Karar Ağaçları(Desicion Tree)

Ağaç tabanlı öğrenme algoritmaları, en çok kullanılan gözetimli öğrenme algorimalarındandır. Genel itibariyle ele alınan bütün problemlerin (sınıflandırma ve regression) çözümüne uyarlanabilirler. Karar agaçları, tesadüfi orman, gradyen güçlendirme (gradient boosting) gibi yöntemler, her türlü veri bilimi problemlerinde yaygın bir şekilde kullanılmaktadırlar. Bu nedenle veri analistleri için bu algoritmaları öğrenmek ve kullanmak çok önemlidir.

Bir karar ağacı, çok sayıda kayıt içeren bir veri kümesini, bir dizi karar kuralları uygulayarak daha küçük kümelere bölmek için kullanılan bir yapıdır. Yani basit karar verme adımları uygulanarak, büyük miktarlardaki kayıtları, çok küçük kayıt gruplarına bölerek kullanılan bir yapıdır. Karar ağaçları, parametrik olmayan bir yöntem olarak düşünülebilir.
Ezbere öğrenme yaşanabilir (“over-fitting”). Bu problemin çözümü için model parametrelere kısıtlamalar ve budama gibi yöntemler kullanılabilir. Budama işlemi, az sayıda nesneyi barındıran yaprak düğümlerin karar ağacı grafiğinden atılmasını ifade etmektedir



Şekil-1: Karar Ağacı

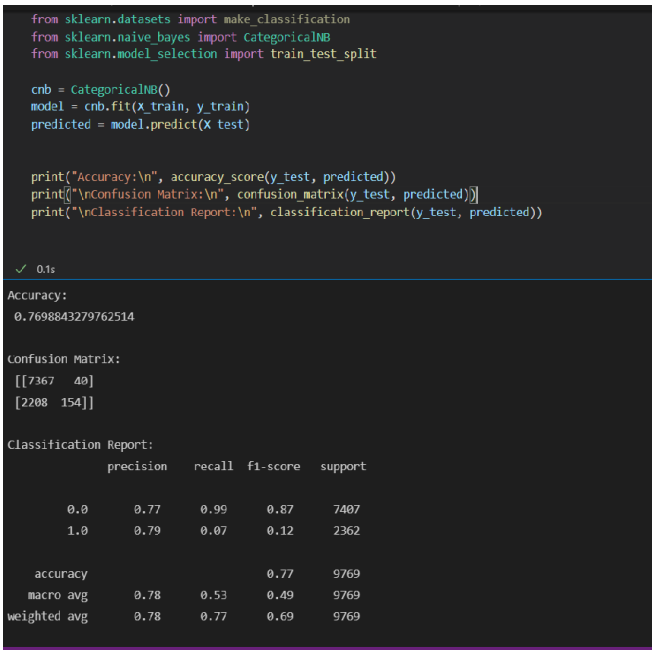
```
rfc = DecisionTreeClassifier()
rfc.fit(X_train, y_train)
rfc_results = result = rfc.predict(X_test)

print("Accuracy:\n", accuracy_score(y_test, y_pred_rf))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
print("\nClassification Report:\n", classification_report(y_test, y_pred_rf))

rfc_cm = confusion_matrix(y_test, rfc_results)
rfc_cm = pd.DataFrame(rfc_cm)
plt.figure(figsize = (10,7))
sns.heatmap(rfc_cm, annot=True)
```

✓ 0.3s

```
Accuracy:
 0.8515712969597707

Confusion Matrix:
 [[6930  477]
 [ 973 1389]]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.88      0.94      0.91      7407
         1.0       0.74      0.59      0.66      2362

    accuracy                           0.85      9769
   macro avg       0.81      0.76      0.78      9769
weighted avg       0.84      0.85      0.85      9769
```

Categorical Naïve Bayes

Her bir özellik için kategorik dağılım kabul eden ayrık özniteliklerle sınıflandırmaya uygundur. Özellikler, her kategori benzersiz bir sayı ile eşlenecek şekilde etiket kodlama teknikleri kullanılarak kodlanmalıdır.



```
from sklearn.datasets import make_classification
from sklearn.naive_bayes import CategoricalNB
from sklearn.model_selection import train_test_split

cnb = CategoricalNB()
model = cnb.fit(X_train, y_train)
predicted = model.predict(X test)


print("Accuracy:\n", accuracy_score(y_test, predicted))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, predicted))
print("\nClassification Report:\n", classification_report(y_test, predicted))
```

✓ 0.1s

```
Accuracy:
 0.7698843279762514

Confusion Matrix:
 [[7367   40]
 [2208  154]]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.77      0.99      0.87      7407
         1.0       0.79      0.07      0.12      2362

    accuracy                           0.77      9769
   macro avg       0.78      0.53      0.49      9769
weighted avg       0.78      0.77      0.69      9769
```

AdaBoost

Adaptive Boosting'in kısaltması olan AdaBoost algoritması, Machine Learning'de Ensemble Method olarak kullanılan bir Boosting tekniğidir. Ağırlıklar her bir örneğe yeniden atandığından ve yanlış sınıflandırılan örneklere daha yüksek ağırlıklar atandığından buna Uyarlamalı Yükseltme denir. Güçlendirme, denetimli öğrenme için sapmanın yanı sıra varyansı azaltmak için kullanılır. Öğrencilerin sırayla büyümesi ilkesine göre çalışır. İlki dışında, sonraki her öğrenci daha önce yetiştirilen öğrencilerden yetiştirilir. Basit bir deyişle, zayıf öğrenenler güçlü olanlara dönüştürülür. AdaBoost algoritması, küçük bir farkla artırma ile aynı prensipte çalışır.

```python
from sklearn.ensemble import AdaBoostClassifier
aBc = AdaBoostClassifier()
model = aBc.fit(X_train, y_train)
predicted = model.predict(X_test)


print("Accuracy:\n", accuracy_score(y_test, predicted))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, predicted))
print("\nClassification Report:\n", classification_report(y_test, predicted))
```

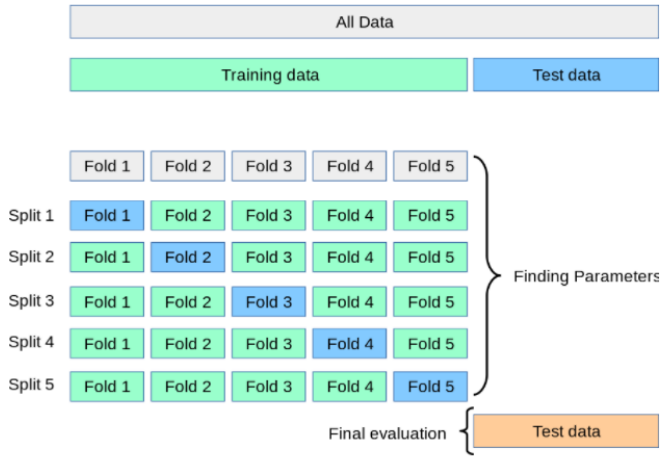
✓ 0.9s

```
Accuracy:
 0.8595557375371071

Confusion Matrix:
 [[6920  487]
 [ 885 1477]]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.89      0.93      0.91      7407
         1.0       0.75      0.63      0.68      2362

    accuracy                           0.86      9769
   macro avg       0.82      0.78      0.80      9769
weighted avg       0.85      0.86      0.85      9769
```

Veri setimize uyguladığımız Makine öğrenmesi algoritmaların en yüksek sonucu AdaBoost algoritması veriyor.

Cross-Validation

Cross-validation, makine öğrenmesi modelinin görmediği veriler üzerindeki performansını mümkün olduğunca objektif ve doğru bir şekilde değerlendirmek için kullanılan istatistiksel bir yeniden örnekleme(resampling) yöntemidir.



AdaBoost modelimizde 10 cross validation sonucunda Accuarcy 0.86 çıkmıştır

```python
from sklearn.model_selection import cross_val_score,cross_val_predict
scores = cross_val_score(model, X, y, cv=10)
scores
print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(), scores.std()))
```

✓ 22.5s

```
0.86 accuracy with a standard deviation of 0.01
```

Referanslar

https://www.kaggle.com/lodetomasi1995/income-classification

https://inblog.in/Categorical-Naive-Bayes-Classifier-implementation-in-Python-dAVqLWkf7E

https://machinelearningmastery.com/confusion-matrix-machine-learning/

https://towardsdatascience.com/performance-metrics-confusion-matrix-precision-recall-and-f1-score-a8fe076a2262

https://medium.com/data-science-tr/makine-%C3%B6%C4%9Frenmesi-dersleri-5-bagging-ve-random-forest-2f803cf21e07

https://medium.com/@ekrem.hatipoglu/machine-learning-classification-naive-bayes-part-11-4a10cd3452b4

https://medium.com/@ekrem.hatipoglu/machine-learning-classification-k-nn-k-en-yak%C4%B1n-kom%C5%9Fu-part-9-6f18cd6185d

https://www.mygreatlearning.com/blog/adaboost-algorithm/

https://medium.com/@k.ulgen90/makine-%C3%B6%C4%9Frenimi-b%C3%B6l%C3%BCm-5-karar-a%C4%9Fa%C3%A7lar%C4%B1-c90bd7593010

https://medium.com/@sertacozker/boosting-algoritmalar%C4%B1-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-edac1174e971