Ödev Sahibi

Büşra Boyacı

1.Problemin Tanımlanması (Business Understanding)

Veriler Portekizli bir bankacılık kurumunun pazarlama kampanyalarıyla ilgilidir. Bankanın pazarlama kampanyaları doğrudan müşterilerin telefon görüşmelerine dayanıyordu. Genellikle ürünün (Bankanın sağladığı kredi vs fonlar) kabul edilip edilemeyeceğini anlamak (cevabın 'evet' veya 'hayır' mı olduğunu anlamak) için müşteriyle birden fazla telefon görüşmesi yapılması gerektiğine inanılıyordu. Bu nedenle müşterilerin kişisel özelliklerinden hangilerinin bankaya cevabında etkili olduğunu Çoklu Regresyon ve Lojistik Regresyon makine öğrenmesi modellerini kullanarak tahminlerde bulunmaya çalışacağız.

2.Veriyi Anlama (Data Understanding)

Verilerimizi ve bunun için gerekli kütüphaneleri çalışma ortamımıza yüklüyoruz.

```
#Kütüphaneler
import pandas as pd
#veri yükleme
veriler = pd.read_csv("bank.csv")
print(veriler)
```

```
       age           job   marital  education  ... pdays  previous poutcome    y
0       58    management   married   tertiary  ...    -1         0  unknown   no
1       44    technician    single  secondary  ...    -1         0  unknown   no
2       33  entrepreneur   married  secondary  ...    -1         0  unknown   no
3       47   blue-collar   married    unknown  ...    -1         0  unknown   no
4       33       unknown    single    unknown  ...    -1         0  unknown   no
...    ...           ...       ...        ...  ...   ...       ...      ...  ...
45206   51    technician   married   tertiary  ...    -1         0  unknown  yes
45207   71       retired  divorced    primary  ...    -1         0  unknown  yes
45208   72       retired   married  secondary  ...   184         3  success  yes
45209   57   blue-collar   married  secondary  ...    -1         0  unknown   no
45210   37  entrepreneur   married  secondary  ...   188        11    other   no
```

Verilerin işlem görmeden önceki halleri

1. Age (sayısal)
2. Job: iş türlerini içeren kategoriler
   ('admin', 'bluecollar', 'entrepreneur', 'jousemaid', 'management', 'retired', 'selfemployed', 'servises', 'student', 'technician', 'unemployed', 'unknown')
3. marital : medeni durumların kategorileri
   ('divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4. education: eğitim kategorileri
   ('basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5. default: Kredisi var mı? Kategoriler ('no','yes','unknown')
6. balance: average yearly balance, in euros (numeric)
7. housing: konut kredisi var mı? kategoriler ('no','yes','unknown')
8. loan: Kişisel kredisi var mı? Kategoriler ('no','yes','unknown')
9. contact: İletişim türü? Kategoriler ('cellular','telephone')
10. day: last contact day of the month (numeric)
11. month: Yıl içindeki en son iletişim kurulan ay? ('jan', 'feb', 'mar', ..., 'nov', 'dec')
12. duration: Son görüşmenin saniye cinsinden süresi? Bu özellik çıktıyı çok etkiler. (e.g., if duration=0 then y='no').

13. campaign: Bu kampanya sırasında iletişim kurulan müşteri sayısı? (numeric)
14. pdays: müşteriyle en son yapılan görüşmeden sonraki gün sayısı ? (numeric; 999 means client was not previously contacted)
15. previous: Bu kampanyadan önce bu müşteriyle yapılan görüşme sayıları? (numeric)
16. poutcome: Önceki pazarlama kampanyasının sonucu? Kategoriler ('failure','nonexistent', 'success')
17. Output variable (desired target): y - has the client subscribed a term deposit? (binary: "yes","no")

2.1 Keşifsel Veri Analizi

Bu aşamada verilerimizde eksik veri olup olmadığını, çeşitli istatistiksel değerleri nasıl görüntüleyebileceğimizi ve hangi kolonlar arasında nasıl bir ilişki olduğunu görüntüleyebildiğimiz bazı tabloları ekranımıza bastırıyoruz.

İsnull özelliğiyle eksik verileri kontrol ediyoruz.

```
#veri keşif
eksikveri= veriler.isnull().sum()
print(eksikveri)
```

```
[45211 rows x 17 columns]
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

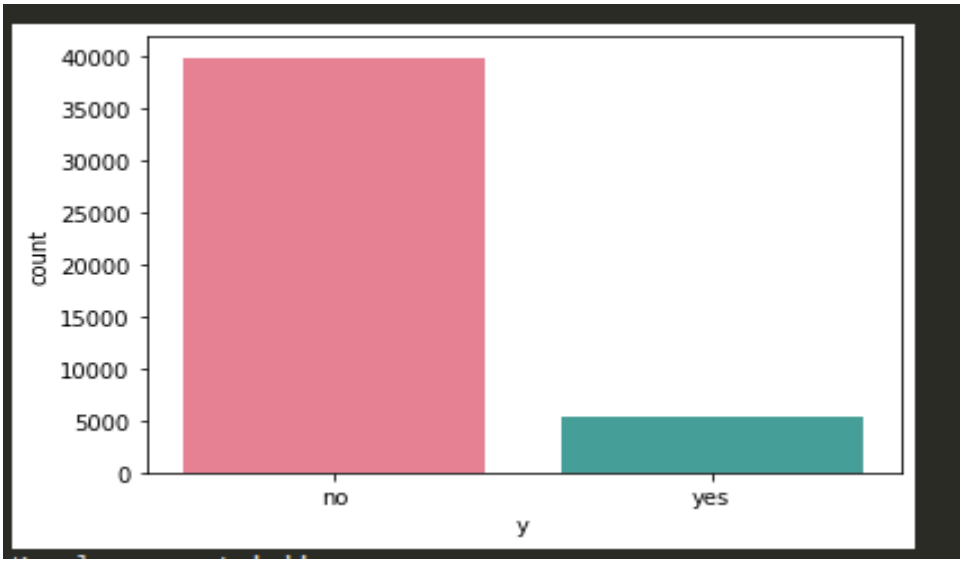
Verilerimizde eksik veri olmadığını görmekteyiz.

Describe özelliğiyle istatistiksel değerlere ulaştık. **Value Count** yöntemiyle verilerde bir dengesiz dağılım olup olmadığını gözlemledik.

```
# istatistik tablosu
istatistikler = veriler.describe()
print(istatistikler)
# verilerin sayılarının grafikleştirilmesi
print(veriler['y'].value_counts())
import seaborn as sn
import matplotlib.pyplot as plt
# palette ve husl ifadeleri grafiğimizin rengini belirlediğimiz ifadelerdir.
sn.countplot(x='y', data=veriler, palette='husl')
plt.show()
```

	age	balance	...	pdays	previous
count	45211.000000	45211.000000	...	45211.000000	45211.000000
mean	40.936210	1362.272058	...	40.197828	0.580323
std	10.618762	3044.765829	...	100.128746	2.303441
min	18.000000	-8019.000000	...	-1.000000	0.000000
25%	33.000000	72.000000	...	-1.000000	0.000000
50%	39.000000	448.000000	...	-1.000000	0.000000
75%	48.000000	1428.000000	...	-1.000000	0.000000
max	95.000000	102127.000000	...	871.000000	275.000000

Veri setindeki istatistiksel değerleri bulunduran bir çıktı bastırtırdık. Bu değerler ortalama, standart sapma, veri setindeki en büyük ve en küçük değer gibi değerleri tek seferde görebilmemizi sağlar. Altındaki grafikte de çıktı kolonumuz olan y'nin dağılımını görmekteyiz. 40000 değer 'no', 5000 değer 'yes' ifadesine sahip olduğumuzu görüyoruz.

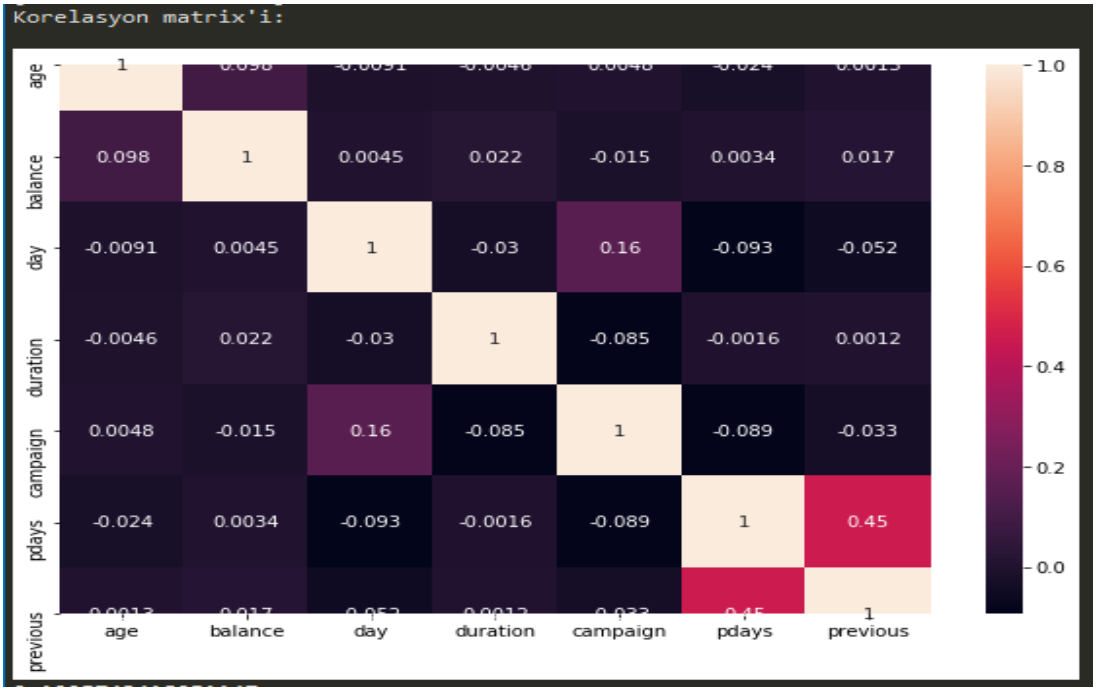


Keşfimize kolonlar arasındaki ilişkileri görebileceğimiz renkli bir grafikle devam ediyoruz.

Seaborn kütüphanesini korelasyon matrisini hesaplamak, **Matplotlib.pyplot'u** da görselleştirebilmek amacıyla kuruyor, kolonlar arasındaki ilişkiyi veren Korelasyon Matrix'ini ekrana getiriyoruz.

```
#korelasyon matrisi (keşif devam)

import seaborn as sn
import matplotlib.pyplot as plt
print("Korelasyon matrix'i: ")
corrmatrix = veriler.corr()
# Matrisin boyutu
plt.figure(figsize=(10,7))
sn.heatmap(corrmatrix, annot=True)
# grafiği göster
plt.show()
```



Şekildeki renk dalgalanmaları hangi iki kolonun birbiriyle ne kadar ilişkili olduğunu gösterir. Renk koyulaşması iki kolonun birbiriyle negatif ilişkili, açılması ise pozitif ilişkili olduğunu gösterir.

3. Veri Hazırlama (Data Preparation)

Problem ve veri seti hakkında yeterli bilgiyi edindikten sonra verileri seçtiğimiz makine öğrenmesi algoritmalarına karşı hazır hale getirdiğimiz aşamadır. Verileri değiştirmek, düzenlemek, temizlemek ve hazırlamak için gerçekleştirilen işlemlerden oluşur.

3.1 Veri Dönüştürme

İlk Kullanacağımız makine öğrenme modeli **Çoklu regresyon (Multiple linear regression**) modeli olduğundan Kategorik olan verilerimizin formlarını sayısal değerlere dönüştürmemiz gerekiyor.

```
#ordinal encoder kullanımı (ver ön işleme)
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder()
oe.fit(veriler[["job","marital", "education", "default", "housing", "loan", "contact", "month",
"poutcome", "y"]])
veriler[["job","marital", "education", "default", "housing",
         "loan", "contact", "month", "poutcome", "y"]] = oe.transform(veriler[["job","marital",
"education", "default", "housing",
                                                "loan", "contact", "month", "poutcome",
"y"]])
```

Kategorik olarak geçen bütün kolanları **Sklearn Kütüphanesi** ve **Ordinal encoder** yardımıyla dönüştürme işlemini gerçekleştirdik.

```
#girdi(x) ve çıktı(y) kolonlarının birbirinden ayrılması
x = veriler.iloc[:, :-1]
y = veriler.iloc[:, 16:17]
```

3.2.1 Öznitelik Üretme veya Seçme

Veri setinde çıktı değişkeni dahil 17 kolonumuz bulunmakta. İlk denememizde 16 girdi değişkenin 16'sını modelimize ekleyerek nasıl bir başarı elde ettiğimizi görmek istiyoruz. O yüzden değişkenlerle ilgili özel bir işlem uygulamıyoruz.

4.1 Çoklu Regresyon Modeli

```
#eğitim ve test verisinin ayrıştırılması
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =0.2, random_state=0)
#Multiple regression (Çoklu regresyon)
from sklearn.linear_model import LinearRegression
model1 = LinearRegression()
model1.fit(x_train, y_train)
#tahmin sonuçları
y_tahmin = model1.predict(x_test)
# modelin başarı scorlaması r2
from sklearn.metrics import r2_score
model1_basari = r2_score(y_test, y_tahmin)
print(model1_basari)
```

5.1 Değerlendirme

R^2 score:

model1_basari	float64	1	0.1995743418251147

Modelin başarı oranı olarak değerlendirdiğimiz **R^2 score** değeri olması gerekenden çok düşük. Bunu iyileştirmek için **crisp-dm** sürecinin **öznitelik** aşamasına geri dönüyor öznitelikler arasından en uygun seçimleri yaparak modelin başarısını arttırmaya çalışıyoruz.

3.2.2 Öznitelik Üretme veya Seçme

Bu aşamaya geri dönüyor ve çıktıyı en iyi tahmin etmemizi sağlayacak olan 5 değişkeni **feature selection** özelliğini kullanarak bulup modelimize dahil ediyoruz (K=5). Regresyon modelini kullandığımız için seçilecek değişkenlerin de **f_regression** parametresine göre seçilmesini istiyoruz.

```
#Feature selection (Öznitelik seçme)
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
# feature extraction
X = veriler.iloc[:, :-1]
Y = veriler.iloc[:, 16:17]
test = SelectKBest(score_func=f_regression, k=5)
fit = test.fit(X, Y)
# Skorları özetle
print(fit.scores_)
features = fit.transform(X)
```

features - NumPy array

	0	1	2	3	4
0	1	2	261	-1	0
1	1	2	151	-1	0
2	1	2	76	-1	0
3	1	2	92	-1	0
4	0	2	198	-1	0

4. Modelleme

Seçtiğimiz öznitelik kolonlarına tanımlanan index değerlerini değiştirmek için kolonlara karşılık gelen isimlerini veriyoruz.

```
#öznitelikleri modele ekleme
ftr = pd.DataFrame(data=features, columns=["housing", "contact", "duration", "p-day",
"previous"])
```

ftr - DataFrame

Index	housing	contact	duration	p-day	previous
0	1	2	261	-1	0
1	1	2	151	-1	0
2	1	2	76	-1	0

4.2 Çoklu Regresyon Modelini Oluşturma

Regresyon modelimizi bu kez de ayrıştırdığımız öznitelikleri dahil ederek kuruyoruz.

```
# Test ve Eğitim verileri olarak ayırıp eğitime hazır hale getirme işlemi
X_train, X_test, Y_train, Y_test = train_test_split(ftr, Y, test_size =0.3, random_state=5)
# Regresyon modeli
from sklearn.linear_model import LinearRegression
model= LinearRegression()
model.fit(X_train, Y_train)
# Regresyon tahmin sonuçları
Y_tahmin = model.predict(X_test)
# r2 score
model_basari = r2_score(Y_test, Y_tahmin)
print(model_basari)
```

5.2 Değerlendirme

R^2 Başarı Skoru:

model_basari	float64	1	0.2110716218414176

Score sonucumuz bir önceki modelimizden daha iyi olsa da hala yeterli doğru tahmin oranına ulaşamadık. Bu yüzden bir kez de ayrıştırılmış öznitelikler üzerinden verileri normalize ederek sonucu nasıl etkilediğini görmek için **3. veri dönüştürme** adımına geri dönüyoruz.

3.2.3 Veri Dönüştürme

Normalize Etme işlemi

```
#verileri normalize ederek skor sonucunu iyileştirmeye çalışma
from sklearn import preprocessing
normalized_X = preprocessing.normalize(ftr)
normalized_Y = preprocessing.normalize(Y)
X_train, X_test, Y_train, Y_test = train_test_split(normalized_X, normalized_Y, test_size =0.3,
random_state=5)
```

normalized_X - NumPy array

	0	1	2	3	4
0	0.00383125	0.0076625	0.999956	-0.00383125	0
1	0.00662165	0.0132433	0.999868	-0.00662165	0
2	0.0131511	0.0263021	0.999481	-0.0131511	0

normalized_Y - NumPy array

	0
0	0
1	0
2	0

Normalizasyon işleminden sonra Regresyon modelimizi bu kez de normalize edilmiş veriler için kuruyor, modelleme aşamasına geri dönüyoruz.

4.3 Çoklu Regresyon Modeli

```
# regresyon
model2 = LinearRegression()
model2.fit(X_train, Y_train)
# tahmin sonuçları
Y_normalized_tahmin = model2.predict(X_test)
# r2 score
model2_basari = r2_score(Y_test, Y_normalized_tahmin)
print(model2_basari)
```

5.3 Değerlendirme

R^2 Skoru:

model2_basari	float64	1	0.05367500581509177

Normalizasyon işlemi modelin başarısında olumsuz sonuç verdi.

Kullanılan bütün modellerin ve İyileştirici Yöntemlerin Genel Değerlendirmesi

Çoklu Regresyon modelimiz için en başarılı denememiz **0,21** sonucuyla **4.2 Regresyon Modeli'ydi**. Regresyon modelinde iyileştirmek için denediğimiz yöntemler işe yaramış olsa da tam istediğimiz sonucu vermediğinden modelimizi değiştirmeye karar veriyoruz.

4.4 Lojistik Regresyon Modeli

Lojistik regresyon bir veya daha fazla değişkenden oluşan modellerin çıktılarını tahmin etmek için kullanılan yöntemdir. Genellikle kesikli veriler dediğimiz verilerin tahminlerinde başarılı bir makine öğrenmesi modelidir. Modelimizi çıkardığımız en iyi 5 özniteliğimizi dahil ederek kuruyoruz.

Modelde öznitelik oluşturma aşamasından sonra oluşturulmuş olan test ve eğitim verilerimizi aynen kullandığımız için tekrar bir eğitim ve test verisi oluşturmuyoruz. Bu da iki model arasındaki farkı daha homojen bir şekilde aynı veriler üzerinden görmemizi sağlamaktadır.

```
#lojistik regresyon modeli
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
# log_reg tahmin sonuçları
Y_log_pred = logreg.predict(X_test)
```

Modelimizi oluşturduk ve tahmin değerlerini elde ettik.

5.4 Değerlendirme

```
# Tahmin doğruluk oranının ekrana yazdırılması
log_score = logreg.score(X_test, Y_test)
print('test veri setinin tahmin doğruluk oranı:', log_score)
```

log_score	float64	1	0.8918460631082277

```
test veri setinin tahmin doğruluk oranı: 0.8918460631082277
```

```
# confusion matrix görünümü ve ekrana bastırılması
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(Y_test, Y_log_pred)
print('Confusion matrix:', confusion_matrix)
```

```
Confusion matrix:
[[11775   190]
 [ 1277   322]]
```

confusion_matrix - NumPy array

	0	1
0	11775	190
1	1277	322

Sonuç

Uygulanan modellerden lojistik regresyon en başarılı tahmin sonuçlarını veren model olmuştur. Confusion matrix'ini yorumladığımızda 11965 tane '0' (no) değerinden 11775 tanesini doğru tahmin etmiştir. 1599 tane '1' (yes) değerinden 1277 tanesini doğru tahmin etmiştir. Bu durumda banka veya bu tür veri analizi gerektiren herhangi bir sektör, müşterilerinin telefon kampanyalarına verecekleri cevabı, kayıt esnasında sahip oldukları veriler veya müşterilerin kişisel özelliklerine dayanarak tahmin edebilmektedir.

Kaynakça

https://archive.ics.uci.edu/ml/datasets/Bank+Marketing#

https://oys.akdeniz.edu.tr/

https://medium.com/@hasan.amanet/crisp-dm-nedir-767b697caeae

https://seaborn.pydata.org/generated/seaborn.countplot.html

https://datatofish.com/multiple-linear-regression-python/

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html