CHURN MODELLING -YAPAY SİNİR AĞLARI

EMEL NUR KARAMAN 180101041

ENGİN KAHEN ÖZTÜRK 180101019

1-VERİ BİLGİSİ

Bu veri setini ‘Deep Learning A-Z: Hands-on Artificial Neural Networks’ udemy kursundaki veri setlerinin arasından seçtik.

Churn Modelling veri seti bir bankadaki 10.000 tane müşteriye aittir. Bu müşterilerin belirli özellikleri vardır: yaş, cinsiyet ve maaş gibi ve bu bağlamda da müşterilerin hala bankaya kayıtlı olup olmadığını belirten ‘exited’ sütunu mevcuttur ve 0-1 değerleri ile ifade edilmiştir. Kısacası Churn Modelling müşterilerin bankada hangi durumlarda kalıp hangi durumlarda ayrıldığının yorumlanmasını sağlayan bir veri setidir.

Veri setinde herhangi bir NaN değer olup olmadığı ‘isna()’ fonksiyonu ile kontrol edildi.

```
1 dataset.isna()#herhangi bir satır veya sütunda veri kaybı yok.
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimate
0	False	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	False	False	False	False	False	False	False	False	False	False	False	False	
9996	False	False	False	False	False	False	False	False	False	False	False	False	
9997	False	False	False	False	False	False	False	False	False	False	False	False	
9998	False	False	False	False	False	False	False	False	False	False	False	False	
9999	False	False	False	False	False	False	False	False	False	False	False	False	

10000 rows × 14 columns

Ardından ‘isna().any()’ fonksiyonu ile de daha anlaşılır bir çıktı alındı.

```
dataset.isna().any()#sütunlarda herhangi bir veri kaybı yok
```

```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

Veri kaybı olmadığı anlaşıldıktan sonra anlamlı değer verecek olan sayısal sütunların seçimi ile ‘describe()’ fonksiyonu kullanılarak istatistiki şekilde sütunlar yorumlandı.

```
dataset[['CreditScore',
         'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
         'IsActiveMember', 'EstimatedSalary']].describe() #belirtilen sütunların istatistiki açıklaması
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000
mean	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881
std	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818
min	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000
25%	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500
max	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000

Veri setinde kaç tane unique ‘surname’ olduğunu belirleyebilmek amacı ile ‘unique()’ fonksiyonu kullanıldı. Bunun sonucunda veri setinde unique olarak bulunan surname’ler döndü ve 2932 tane unique surname olduğu görüldü.

```
1 print(dataset['Surname'].unique())#Unique surname'leri döndürür
2 print()
3 print(len(dataset['Surname'].unique()))#Unique surname'lerin sayısını döndürür
```

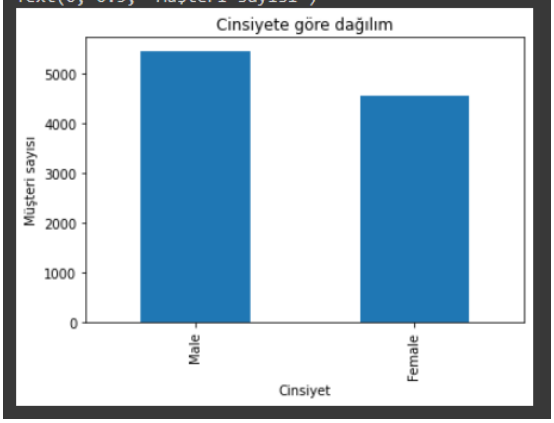
```
['Hargrave' 'Hill' 'Onio' ... 'Kashiwagi' 'Aldridge' 'Burbidge']

2932
```

Gender ve Geography sütunlarının ‘plot()’ fonksiyonu ile veri görsellemesi yapıldı.

```
1 dataset['Gender'].value_counts().plot(kind="bar")
2 plt.title('Cinsiyete göre dağılım')
3 plt.xlabel('Cinsiyet')
4 plt.ylabel('Müşteri sayısı')
5
```

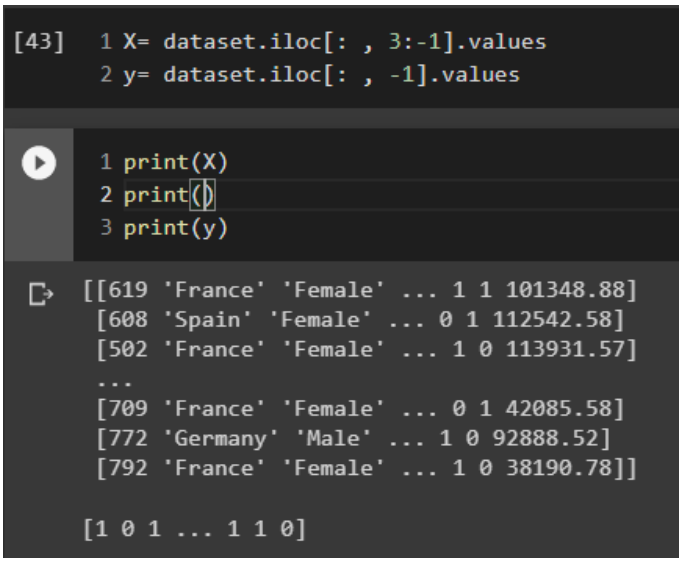
```
Text(0, 0.5, 'Müşteri sayısı')
```




```
1 dataset['Geography'].value_counts().plot(kind="bar")
2 plt.title('Ülkelere göre dağılım')
3 plt.xlabel('Ülkeler')
4 plt.ylabel('Müşteri sayısı')
```

```
Text(0, 0.5, 'Müşteri sayısı')
```

Yapay sinir ağına vereceğimiz Feature’ları seçtik ve X matrisine verdik. Bunlar: ‘Credit Score’, ‘Geography’, ‘Gender’, ‘Age’, ‘Tenure’, ‘Num Of Products’, ‘HasCrCard’, ‘IsActiveMember’, ‘EstimatedSalary’. Daha sonra ‘Exited’ sütununu da y matrisine verdik.



X matrisindeki nominal verileri YSA'nın anlayabileceği nümerik verilere çevirmek için 'Gender' sütununda Label Encoding metodu, 'Geography' sütununda da One Hot Encoding metodu kullanıldı. Ardından datanın %80'i Train , %20'si Test için ayrıldı. Feature Calling işlemi verinin standardize edilmesi yani standart sapmanın 1 ortalamanın 0 değerlerine indirgenmesi için yapıldı.

2-MODEL

Yapay sinir ağını oluştururken başlangıçta hidden layer ve nöron sayılarına karar verirken 2n+1, 2n-1, n+1, n-1 formüllerini kullanıldı(n=10) ve aktivasyon fonksiyonu olarak da 'Sigmoid','Relu', 'TanH' fonksiyonlarından yararlanıldı.

ANN:

```
[103]  1 ann=tf.keras.models.Sequential()

[104]  1 #ilk hidden layer 2n+1 nöron
       2 ann.add(tf.keras.layers.Dense(units=21, activation='sigmoid'))

[105]  1 #ikinci hidden layer 2n-1
       2 ann.add(tf.keras.layers.Dense(units=19, activation= 'sigmoid'))

[106]  1 #output layer
       2 ann.add(tf.keras.layers.Dense(units= 1, activation='sigmoid'))
```

Sonuç:

```
[[1527   68]
 [ 276  129]]
Accuracy Score: 82.8
```

ANN:

```
[127]  1 ann=tf.keras.models.Sequential()

       1 #ilk hidden layer n+1 nöron
       2 ann.add(tf.keras.layers.Dense(units=11, activation='tanh'))

       1 #ikinci hidden layer n-1
       2 ann.add(tf.keras.layers.Dense(units=9, activation= 'tanh'))

[130]  1 #output layer
       2 ann.add(tf.keras.layers.Dense(units= 1, activation='sigmoid'))
```

Sonuç:

```
[[1500   95]
 [ 259  146]]
Accuracy Score: 82.3
```

ANN:

```
[138] 1 ann=tf.keras.models.Sequential()

      1 #ilk hidden layer 2n-1 nöron
      2 ann.add(tf.keras.layers.Dense(units=19, activation='tanh'))

[140] 1 #ikinci hidden layer 2n+1
      2 ann.add(tf.keras.layers.Dense(units=21, activation= 'tanh'))

[141] 1 #output layer
      2 ann.add(tf.keras.layers.Dense(units= 1, activation='sigmoid'))
```

Sonuç:

```
[[1512   83]
 [ 225  180]]
Accuracy Score: 84.6
```

ANN:

```
[149] 1 ann=tf.keras.models.Sequential()

[150] 1 #ilk hidden layer 2n+1 nöron
      2 ann.add(tf.keras.layers.Dense(units=21, activation='relu'))

      1 #output layer
      2 ann.add(tf.keras.layers.Dense(units= 1, activation='sigmoid'))
```

Sonuç:

```
[[1528   67]
 [ 217  188]]
Accuracy Score: 85.8
```

ANN:

```
[159]  1 ann=tf.keras.models.Sequential()

[160]  1 #ilk hidden layer 2n+1 nöron
       2 ann.add(tf.keras.layers.Dense(units=21, activation='relu'))

       1 #output layer
       2 ann.add(tf.keras.layers.Dense(units= 1, activation='relu'))
```

Sonuç:

```
[[1446  149]
 [ 294  111]]
Accuracy Score: 77.85
```