Büyük Veri Proje Ödevi

2022-2023

Güz Dönemi

Grup üyeleri:

Vedat, Arslan, 2/A, B181210030, vedat.arslan@ogr.sakarya.edu.tr

İbrahim, Şahin, 2/A, B181210027, ibrahim.sahin5@ogr.sakarya.edu.tr

Grup üyelerinin projeye katkısı:

VEDAT ARSLAN PROJE KATKI:

. Yolcu sınıflarına göre hayatını kaybeden/hayatta kalan erkek ve kadın yolcu sayıları görselleştirme.

. Yolcu sınıflarına göre hayatını kaybeden (0) ve hayatta kalan (1) yolcu sayıları görselleştirme

. Yolcu sınıfına göre; yalnız seyahat eden, ailesiyle birlikte seyahat eden yolcuların hayatta kalma oranları elde edilmesi.

. Ödenen bilet ücretlerine göre hayatta kalma oranlarının bulunmuştur. Bunun için bilet ücretleri (0'dan 512'ye) büyük farklarla çeşitlilik gösterdiğinden bilet ücreti değerleri eşit aralıklara bölünerek hayatta kalma oranları elde edilmiştir.

. Oluşturulan ücret bantlarına göre hayatta kalma oranları gösterilmiştir. Bu sonuca göre yolcular arasında ödenen bilet ücretleri arttıkça hayatta kalma oranları da artmıştır.

. Makine öğrenimi modelinin, devasa gemi enkazı sırasında kimin hayatta kaldığını tahmin etmesi gerekiyor.

İBRAHIM SAHİNİN PROJE KATKI:

. Cinsiyete göre yolcuların dağılımı ve yuvarlak diyagram ile bu dağılımın görselleştirilmesi

. Farklı yaş aralıklarına göre yolcuların dağılımı

. Yolcu sınıflarına göre yolcu sayılarının dağılımı yatay çubuk grafik ile görselleştirilmesi.

. Yolcuların gemiye biniş limanlarına göre dağılımları çubuk grafik ile görselleştirilmesi

. Cinsiyete göre hayatını kaybeden (0) ve kurtulan yolcuların (1) sayıları çubuk bar ile görselleştirilmesi.

. İki sütuna ('Sex', 'Survived') .groupby() fonksiyonu ile size() ve unstack() fonksiyonlarının uygulanması sonucunda hayatta kalma ('Survived') değişekeninin tek çubuk üzerinde "stacked=True" komutu eklenerek ile daha iyi bir görsele dönüştürme.

. Cinsiyete göre; yalnız seyahat eden, ailesiyle birlikte seyahat eden yolcuların hayatta kalma oranları ve analizi

Projenin adı: Buzdağına Çarpan Titanic Gemisinin Veri Analizi ve Tahmini

Projenin amacı: Yolcuların cinsiyet, yaş ve yolcu sınıflarına göre dağılımları ve bu sınıflandırmalara göre hayatta kalma oranları analiz etme ve Makine öğrenimi modelinin, devasa gemi enkazı sırasında kimin hayatta kaldığını tahmin etmesi gerekiyor.

Verinin açıklanması:

Survival Hayatta kalma 0 = Hayır, 1 = Evet

Pclass Bilet sınıfı 1 = Birinci, 2 = İkinci, 3 = Üçüncü sınıf

Sex Cinsiyet

Age Yaş

Sibsp # yolcunun gemide bulunan kardeş / eş sayısı

Parch # yolcunun gemide bulunan ebeveyn / çocuğu sayısı

Ticket Bilet numarası

Fare Bilet ücreti

Cabin Kamara numarası

Embarked Gemiye biniş limanı C = Cherbourg, Q = Queenstown, S = Southampton

Kullanılan büyük veri platformları:

Kaggle ‘dan Titanic Veri Seti

Projenin açıklanması:

Pyhton diliyle titanic.csv datasetini analiz ettik

Kullandıgımız Kütüphaneler:

import numpy as np

import pandas as pd (Veri manipülasyonu)

import matplotlib.pyplot as plt(veri görsellestirme)

import seaborn as sns(veri görsellestirme)

from sklearn.preprocessing import MinMaxScaler(ham verinin temiz bir veri setine dönüştürülmesi ve veri setinin sayısal verilere dönüştürülmesi gerekmektedir. tahmin, Makine öğrenimi modelinin, devasa gemi enkazı sırasında kimin hayatta kaldığını tahmin etmesi gerekiyor.)

Projenin çıktıları:

read_csv('file.csv')=Csv veri setindeki verileri okur.

drop()=Csv veri setindeki parametrede verilen sütunları siler.

len()=Veri seti uzunluğu döndürür.

print()=Ekrana yazdırma fonksiyonu

```
passenger_df=pd.read_csv('/kaggle/input/titanic/titanic.csv')
passenger_df = passenger_df.drop(['PassengerId', 'Name', 'Cabin', 'Ticket'], axis=1)
print ('Kadın Yolcu Sayısı: ', len(passenger_df.groupby('Sex').groups['male']))
print ('Erkek Yolcu Sayısı: ', len(passenger_df.groupby('Sex').groups['female']))
male_passenger = passenger_df[passenger_df['Sex']== 'male']
female_passenger = passenger_df[passenger_df['Sex']== 'female']
kid_passenger = passenger_df[passenger_df['Age'] < 16]
male_kid_passenger = kid_passenger[kid_passenger['Sex'] == 'male']
female_kid_passenger = kid_passenger[kid_passenger['Sex'] == 'female']
print(male_passenger)
adult_male_passenger = male_passenger.drop(male_kid_passenger.index[:])
adult_female_passenger = female_passenger.drop(female_kid_passenger.index[:])
print ('Tüm Yolcu Sayısı:', len(passenger_df))
print ('Tüm Erkek Yolcu Sayısı:', len(male_passenger))
print ('Tüm Kadın Yolcu Sayısı:', len(female_passenger))
print ('Tüm Yetiskin Kadın Yolcu Sayısı:', len(adult_male_passenger))
print ('Tüm Yetiskin Erkek Yolcu Sayısı:', len(adult_female_passenger))
print ('Tüm Erkek Cocuk Sayısı:', len(kid_passenger))
```

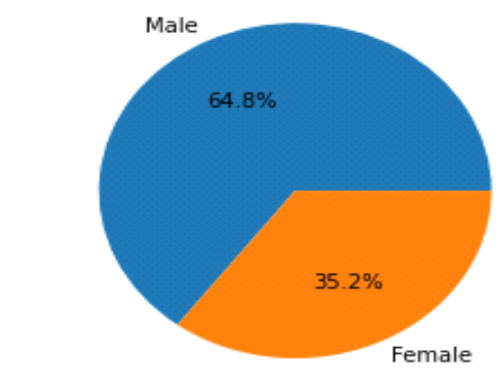
```
Kadın Yolcu Sayısı:  577
Erkek Yolcu Sayısı:  314
     Survived  Pclass   Sex   Age  SibSp  Parch     Fare Embarked
0           0       3  male  22.0      1      0   7.2500        S
4           0       3  male  35.0      0      0   8.0500        S
5           0       3  male   NaN      0      0   8.4583        Q
6           0       1  male  54.0      0      0  51.8625        S
7           0       3  male   2.0      3      1  21.0750        S
..        ...     ...   ...   ...    ...    ...      ...      ...
883         0       2  male  28.0      0      0  10.5000        S
884         0       3  male  25.0      0      0   7.0500        S
886         0       2  male  27.0      0      0  13.0000        S
889         1       1  male  26.0      0      0  30.0000        C
890         0       3  male  32.0      0      0   7.7500        Q

[577 rows x 8 columns]
Tüm Yolcu Sayısı: 891
Tüm Erkek Yolcu Sayısı: 577
Tüm Kadın Yolcu Sayısı: 314
Tüm Yetiskin Kadın Yolcu Sayısı: 537
Tüm Yetiskin Erkek Yolcu Sayısı: 271
Tüm Erkek Cocuk Sayısı: 83
```

plt.pie()=dairesel grafik için veri seti isimleri yüzdelik oran

plt.title()=grafik başlığı

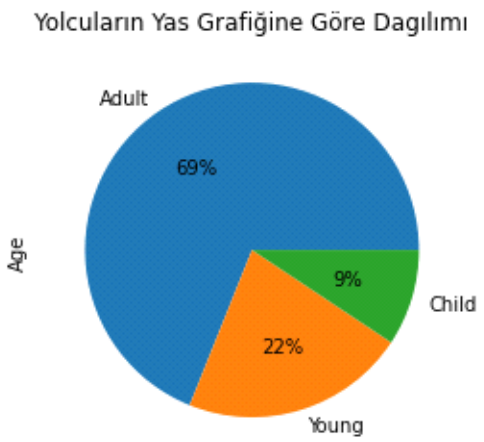
plt.show()=grafik göster.

```
x = [len(male_passenger), len(female_passenger)]
label = ['Male', 'Female']
plt.pie(x, labels = label, autopct = '%1.01f%%')
plt.title('Yolcuların Cinsiyet Grafiğine Göre Dagılımı')
plt.show()
```



Yasa göre dağılım yapan fonksiyon Yazdım. Çocuk , Genç ve Yetişkin

```python
def age_distribution(x):
    if x>=0 and x <16:
        return 'Child'
    elif x>=16 and x<=24:
        return 'Young'
    else:
        return 'Adult'
passenger_df['Age'].apply(age_distribution).value_counts().plot(kind='pie', autopct='%1.0f%
plt.title('Yolcuların Yas Grafiğine Göre Dagılımı')
plt.show()
```



mean()= Ortalamı alan değer.

plt.xlabel()= Barh grafiği için x ekseni ismi

plt.ylabel()= Barh grafiği için y ekseni ismi

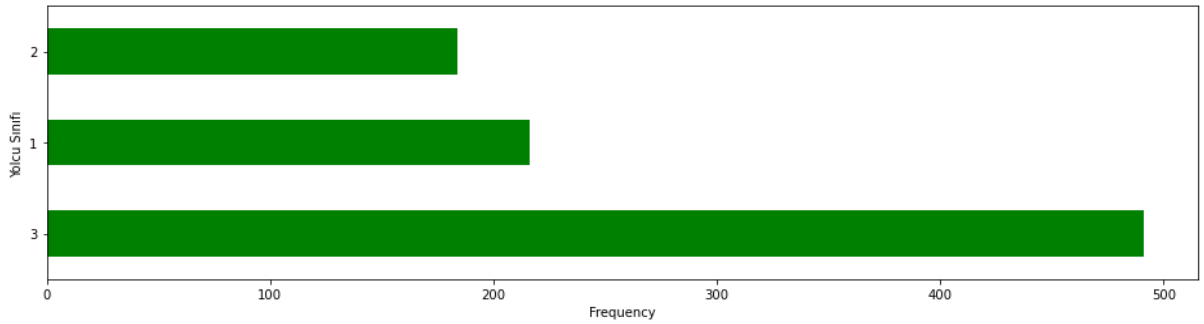
figsize()= Grafik ölçeklendirme

```python
print ('Yetişkin Erkek Yolcuların Yas Ortalaması:', adult_male_passenger['Age'].mean())
print ('Yetişkin Kadın Yolcuların Yas Ortalaması:', adult_female_passenger['Age'].mean())
print (' Cocuk Yolcuların Yas Ortalaması:', kid_passenger['Age'].mean())
passenger_df['Pclass'].value_counts()
passenger_df['Pclass'].value_counts().plot(kind='barh', color='green', figsize=[16,4])
plt.xlabel('Frequency')
plt.ylabel('Yolcu Sınıfı')
plt.show()
```

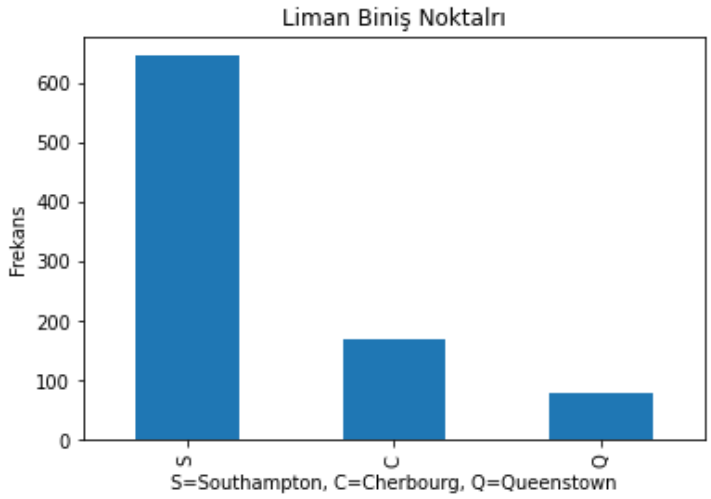
Yetişkin Erkek Yolcuların Yas Ortalaması: 33.17312348668281

Yetişkin Kadın Yolcuların Yas Ortalaması: 32.0

Cocuk Yolcuların Yas Ortalaması: 6.369518072289157

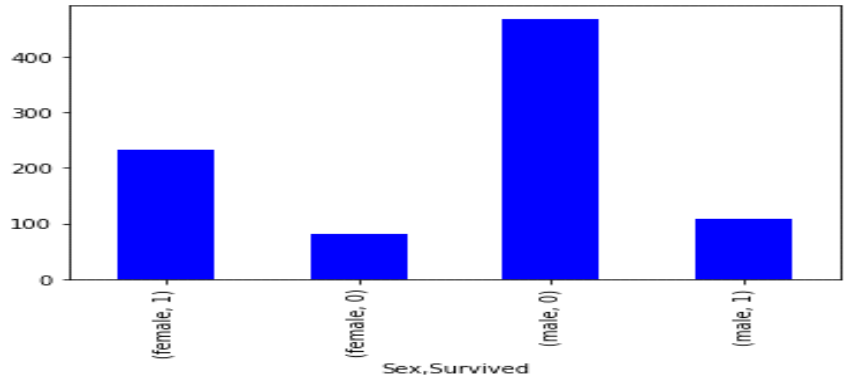



```
first_class_passenger = passenger_df[passenger_df['Pclass'] == 1]
second_class_passenger = passenger_df[passenger_df['Pclass'] == 2]
third_class_passenger = passenger_df[passenger_df['Pclass'] == 3]
print(passenger_df['Embarked'].describe())
passenger_df['Embarked'] = passenger_df['Embarked'].fillna('S')
passenger_df['Embarked'].value_counts().plot(kind='bar')
plt.title('Liman Biniş Noktalrı')
plt.ylabel('Frekans')
plt.xlabel('S=Southampton, C=Cherbourg, Q=Queenstown')
plt.show()
```

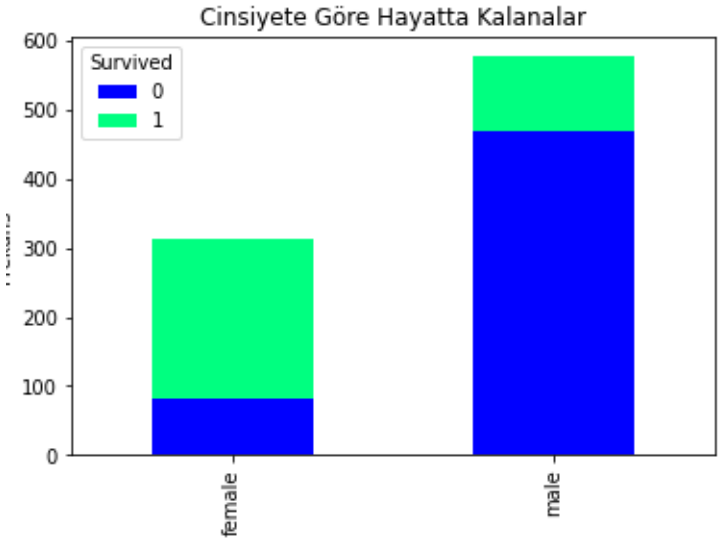


```
print(passenger_df['Survived'].value_counts())
print(passenger_df.groupby('Sex')['Survived'].value_counts())
passenger_df.groupby('Sex')['Survived'].value_counts().plot(kind='bar', stacked=True, colormap='winter')
plt.show()
##################################################
sex_survived = passenger_df.groupby(['Sex', 'Survived'])
sex_survived.size().unstack().plot(kind='bar', stacked=True, colormap='winter')
plt.ylabel('Frekans')
plt.title('Cinsiyete Göre Hayatta Kalanalar')
plt.show()
```

```
0    549
1    342
Name: Survived, dtype: int64
Sex      Survived
female   1           233
         0            81
male     0           468
         1           109
Name: Survived, dtype: int64
```

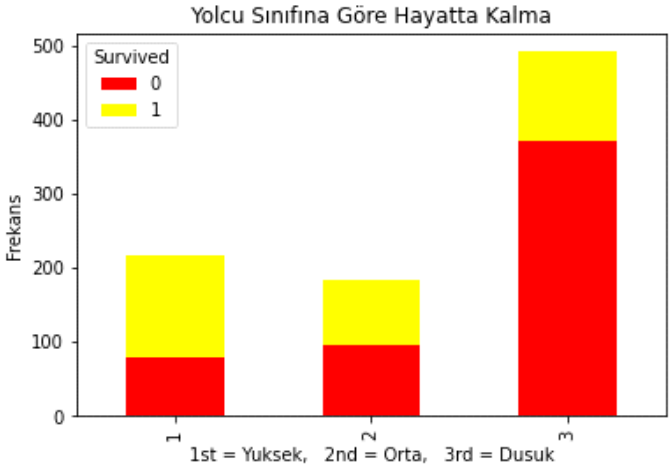


unstack()= En içteki düzeyi özetlenmiş dizin etiketlerinden oluşan yeni bir sütun etiketleri düzeyine sahip bir DataFrame döndürür. Dizin bir MultiIndex değilse,



```
print ('Kadın Yolcuların Hayatta Kalma Ortalaması:', adult_female_passenger['Survived'].mean())
print ('Erkek Yolcuların Hayatta Kalma Ortalaması:', adult_male_passenger['Survived'].mean())
class_survived = passenger_df.groupby(['Pclass', 'Survived'])
print(class_survived.size().unstack())
class_survived.size().unstack().plot(kind='bar', stacked=True, colormap='autumn')
plt.xlabel('1st = Yuksek,   2nd = Orta,   3rd = Dusuk')
plt.ylabel('Frekans')
plt.title('Yolcu Sınıfına Göre Hayatta Kalma')
plt.show()
```

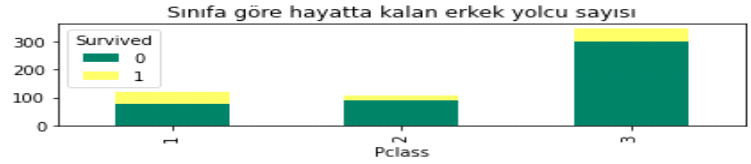
```
Kadın Yolcuların Hayatta Kalma Ortalaması: 0.7564575645756457
Erkek Yolcuların Hayatta Kalma Ortalaması: 0.16387337057728119
Survived    0    1
Pclass
1          80  136
2          97   87
3         372  119
```

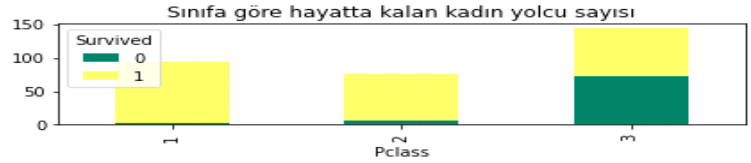


Groupby()= DataFrame'i bir eşleyici kullanarak veya bir dizi sütuna göre gruplandırın. Bir groupby işlemi, nesneyi bölme, bir işlev uygulama gibi bazı kombinasyonları içerir.

```
print ('Yolcu Sınıfında Kalan Erkek Yolcu Sayısına Göre Hayatta Kalma: ',
male_passenger.groupby(['Pclass', 'Survived']).size().unstack())
print ('Yolcu Sınıfında Kalan Kadın Yolcu Sayısına Göre Hayatta Kalma:',
female_passenger.groupby(['Pclass', 'Survived']).size().unstack())
fig, axes = plt.subplots(nrows=2, ncols=1)
male_passenger.groupby(['Pclass','Survived']).size().unstack().plot(kind='bar', title='Sınıfa göre hayatta kalan erkek yolcu sayısı',
                                                                    stacked=True, colormap='summer', ax=axes[0])
female_passenger.groupby(['Pclass','Survived']).size().unstack().plot(kind='bar', title='Sınıfa göre hayatta kalan kadın yolcu sayısı',
                                                                      stacked=True, colormap='summer', ax=axes[1])
plt.tight_layout()
plt.show()
```

```
Yolcu Sınıfında Kalan Erkek Yolcu Sayısına Göre Hayatta Kalma:  Survived    0   1
Pclass
1            77  45
2            91  17
3           300  47
Yolcu Sınıfında Kalan Kadın Yolcu Sayısına Göre Hayatta Kalma: Survived   0   1
Pclass
1             3  91
2             6  70
3            72  72
```






```
plt.show()
without_sibsp_passenger = passenger_df[passenger_df['SibSp']==0]
alone_passenger = without_sibsp_passenger[without_sibsp_passenger['Parch']==0]
print(alone_passenger.head(7))
family_passenger = passenger_df.drop(alone_passenger.index[:])
print(family_passenger.tail(6))
print ('Yolculardan Tek Olanların Ortalaması:', alone_passenger.groupby('Sex')['Survived'].mean())
print ('Yolculardan Aileyle Olanların Ortalaması:', family_passenger.groupby('Sex')['Survived'].mean())
print ('')
print ('')
#Mean of survived alone passengers and passengers with family by passenger class
print ('Hayatta kalan tek başına yolcuların ortalaması:', alone_passenger.groupby('Pclass')['Survived'].mean())
print ('Hayatta kalan Aile  yolcuların ortalaması:', family_passenger.groupby('Pclass')['Survived'].mean())
passenger_df['Fare'].fillna(passenger_df['Fare'].dropna().median(), inplace=True)
passenger_df['FareBand'] = pd.qcut(passenger_df['Fare'], 4)
passenger_df['FareBand'].value_counts().sort_values(ascending= False)
passenger_df[['FareBand', 'Survived']].groupby(['FareBand'],
                        as_index=False).mean().sort_values(by='FareBand',
                        ascending=True)
```

```
     Survived  Pclass     Sex   Age  SibSp  Parch     Fare Embarked
2           1       3  female  26.0      0      0   7.9250        S
4           0       3    male  35.0      0      0   8.0500        S
5           0       3    male   NaN      0      0   8.4583        Q
6           0       1    male  54.0      0      0  51.8625        S
11          1       1  female  58.0      0      0  26.5500        S
12          0       3    male  20.0      0      0   8.0500        S
14          0       3  female  14.0      0      0   7.8542        S
     Survived  Pclass     Sex   Age  SibSp  Parch     Fare Embarked
871         1       1  female  47.0      1      1  52.5542        S
874         1       2  female  28.0      1      0  24.0000        C
879         1       1  female  56.0      0      1  83.1583        C
880         1       2  female  25.0      0      1  26.0000        S
885         0       3  female  39.0      0      5  29.1250        Q
888         0       3  female   NaN      1      2  23.4500        S
Yolculardan Tek Olanların Ortalaması: Sex
female    0.785714
male      0.155718
Name: Survived, dtype: float64
Yolculardan Aileyle Olanların Ortalaması: Sex
female    0.712766
male      0.271084
Name: Survived, dtype: float64


Hayatta kalan tek başına yolcuların ortalaması: Pclass
1    0.532110
2    0.346154
3    0.212963
Name: Survived, dtype: float64
Hayatta kalan Aile  yolcuların ortalaması: Pclass
1    0.728972
2    0.637500
3    0.299401
Name: Survived, dtype: float64
```

y_test hayatta kalma 0 hayata değil 1 hayatta X veri setinine bağlı y_pred değerleri asıl sonuçları tahmin etmeye çalısıyoruz.

```
df = pd.read_csv('/kaggle/input/titanic/titanic.csv')
cols = ['Name', 'Ticket', 'Cabin']
df = df.drop(cols, axis=1)

df=df.dropna()

dummies = []
cols = ['Pclass', 'Sex', 'Embarked']
for col in cols:
    dummies.append(pd.get_dummies(df[col]))

titanic_dummies = pd.concat(dummies, axis=1)
df = pd.concat((df,titanic_dummies), axis=1)
df = df.drop(['Pclass', 'Sex', 'Embarked'], axis=1)


df['Age'] = df['Age'].interpolate()

X = df.values
y = df['Survived'].values
X = np.delete(X, 1, axis=1)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)



from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train)

y_pred = regressor.predict(X_test)
print(y_pred)
```

İlk 10 değeri aşağıdaki tabloda gösterdim.

y_pred (Formatlanmış %0g) değerleri

0.5	**0.3**	**0.7**	**0.3**	**-0.01**	**-0.002**	**0.7**	**1**	**0.6**	**1**

y_test değerleri

0	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**