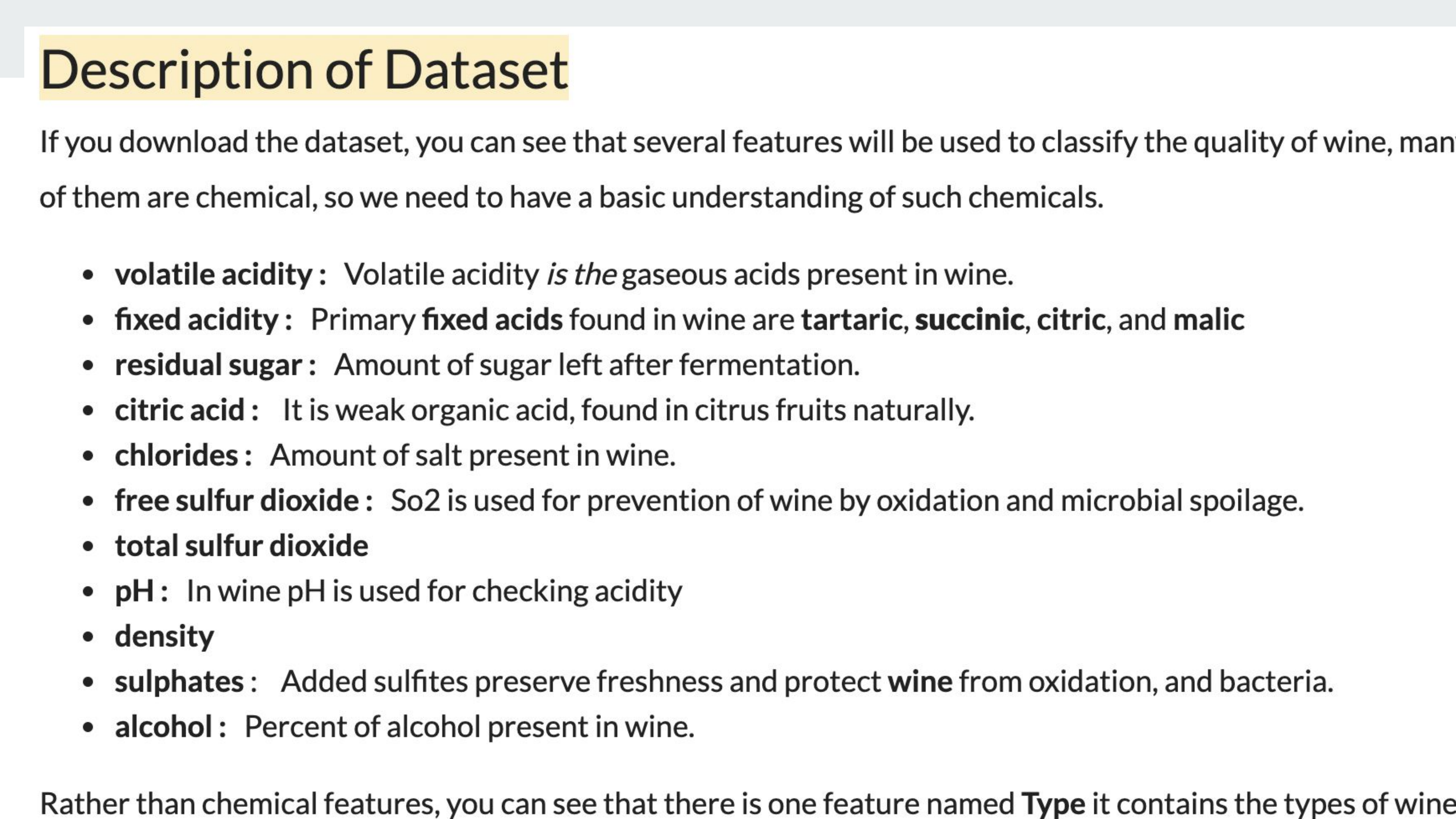
Wine Quality Prediction Using Machine Learning

" Wine is the most healthful and most hygienic of beverages "

Overview

- Basics understanding of Wine.
- Data description
- Importing modules
- Study dataset
- Visualization
- Handle null values
- Split dataset
- Normalization
- Applying model



Description of Dataset

If you download the dataset, you can see that several features will be used to classify the quality of wine, man

of them are chemical, so we need to have a basic understanding of such chemicals.

- **volatile acidity :** Volatile acidity *is the* gaseous acids present in wine.
- **fixed acidity :** Primary **fixed acids** found in wine are **tartaric**, **succinic**, **citric**, and **malic**
- **residual sugar :** Amount of sugar left after fermentation.
- **citric acid :** It is weak organic acid, found in citrus fruits naturally.
- **chlorides :** Amount of salt present in wine.
- **free sulfur dioxide :** So2 is used for prevention of wine by oxidation and microbial spoilage.
- **total sulfur dioxide**
- **pH :** In wine pH is used for checking acidity
- **density**
- **sulphates** : Added sulfites preserve freshness and protect **wine** from oxidation, and bacteria.
- **alcohol :** Percent of alcohol present in wine.

Rather than chemical features, you can see that there is one feature named **Type** it contains the types of wine

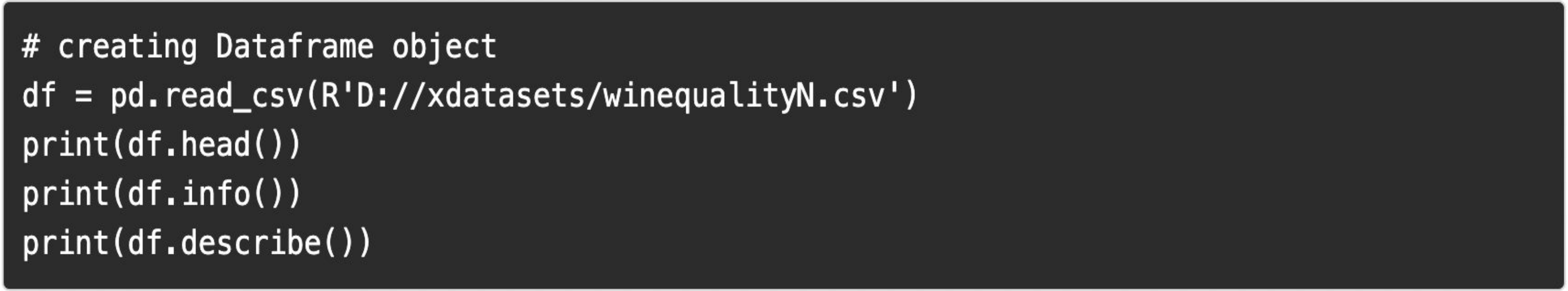
Importing modules

Let’s import,

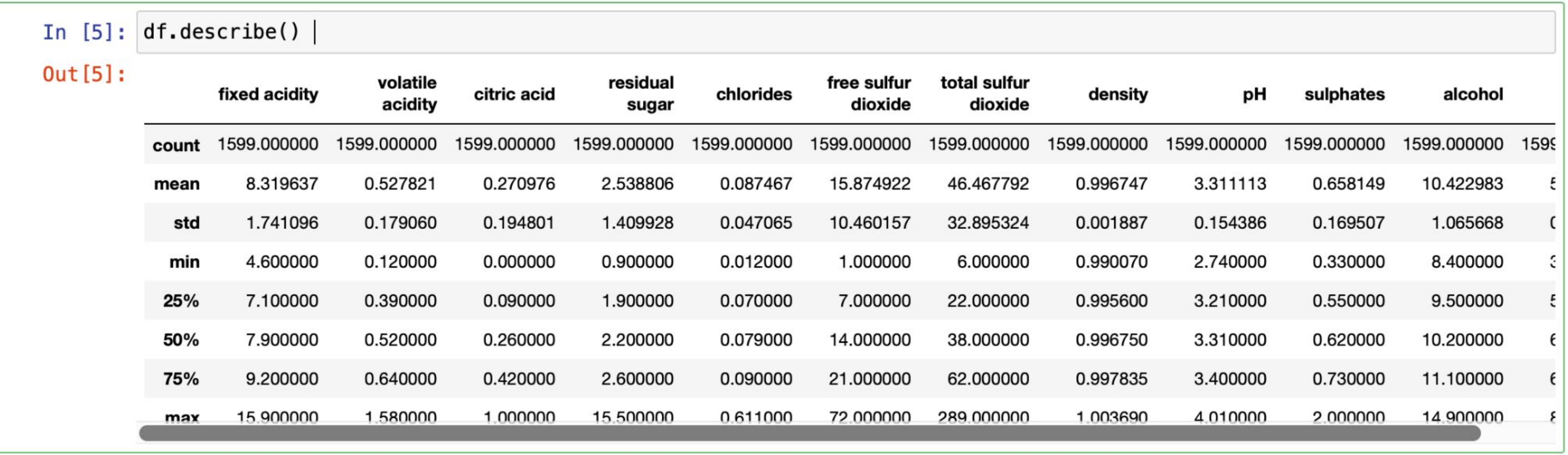
```
# import pandas
import pandas as pd

# import numpy
import numpy as np

# import seaborn
import seaborn as sb

# import matplotlib
import matplotlib.pyplot as plt
```

Study dataset

For the next step, we have to check what technical information contained in the data,



```
# creating Dataframe object
df = pd.read_csv(R'D://xdatasets/winequalityN.csv')
print(df.head())
print(df.info())
print(df.describe())
```



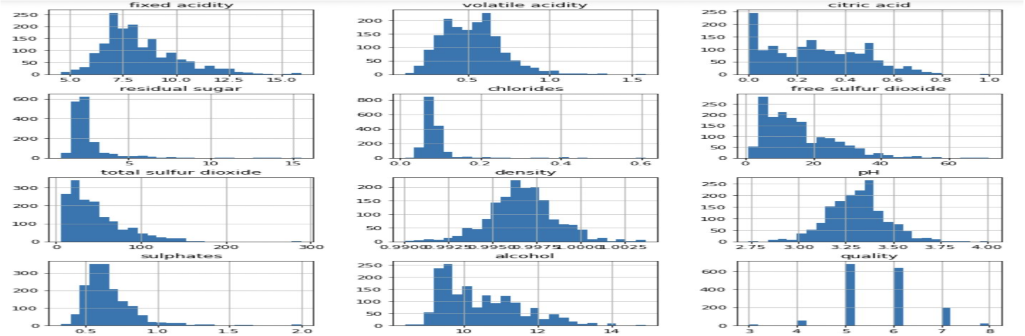
```
In [5]: df.describe()
```

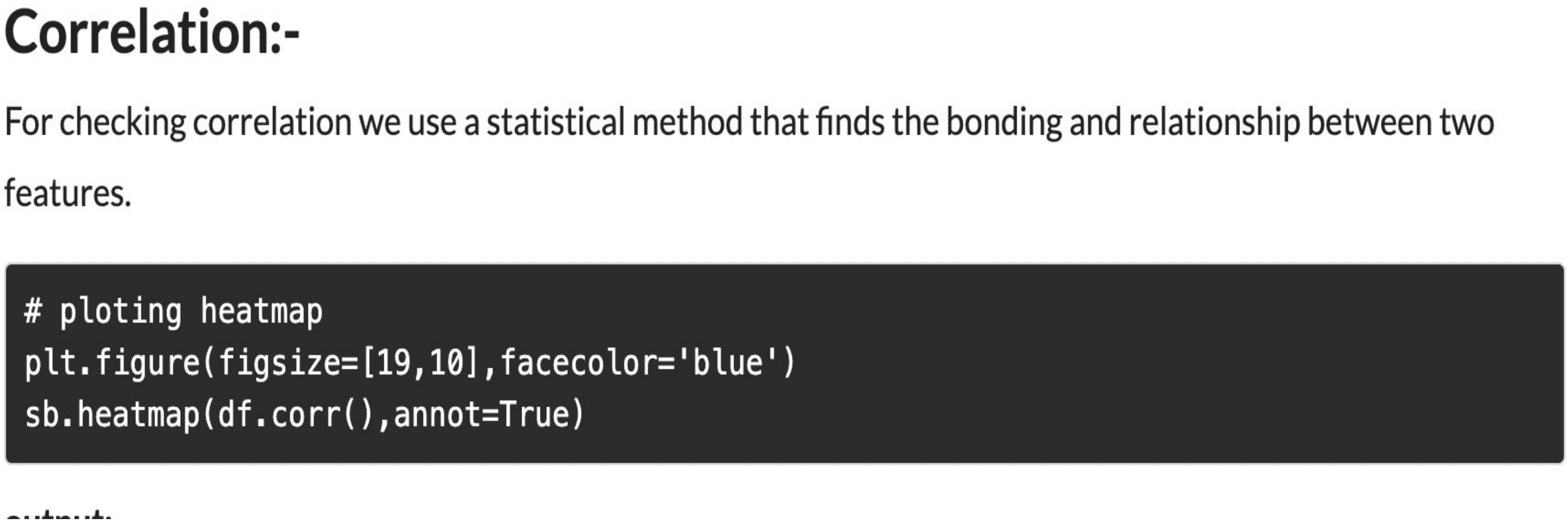
```
Out[5]:
       fixed acidity     volatile  citric acid     residual    chlorides  free sulfur  total sulfur      density           pH    sulphates      alcohol
                          acidity                     sugar                   dioxide       dioxide
count    1599.000000  1599.000000  1599.000000  1599.000000  1599.000000  1599.000000   1599.000000  1599.000000  1599.000000  1599.000000  1599.000000  1599
 mean       8.319637     0.527821     0.270976     2.538806     0.087467    15.874922     46.467792     0.996747     3.311113     0.658149    10.422983     5
  std       1.741096     0.179060     0.194801     1.409928     0.047065    10.460157     32.895324     0.001887     0.154386     0.169507     1.065668     0
  min       4.600000     0.120000     0.000000     0.900000     0.012000     1.000000      6.000000     0.990070     2.740000     0.330000     8.400000     3
  25%       7.100000     0.390000     0.090000     1.900000     0.070000     7.000000     22.000000     0.995600     3.210000     0.550000     9.500000     5
  50%       7.900000     0.520000     0.260000     2.200000     0.079000    14.000000     38.000000     0.996750     3.310000     0.620000    10.200000     6
  75%       9.200000     0.640000     0.420000     2.600000     0.090000    21.000000     62.000000     0.997835     3.400000     0.730000    11.100000     6
  max      15.900000     1.580000     1.000000    15.500000     0.611000    72.000000    289.000000     1.003690     4.010000     2.000000    14.900000     8
```

Visualization

We know that the “image speaks everything” here the visualization came into the work, we use visualization for explaining the data. In other words, we can say that it is a graphic representation of data that is used to find useful information.

```
df.hist(bins=25,figsize=(10,10))
# display histogram
plt.show()
```





Correlation:-

For checking correlation we use a statistical method that finds the bonding and relationship between two features.

```
# ploting heatmap
plt.figure(figsize=[19,10],facecolor='blue')
sb.heatmap(df.corr(),annot=True)
```

output:

Splitting dataset

Now we perform a split operation on our dataset:

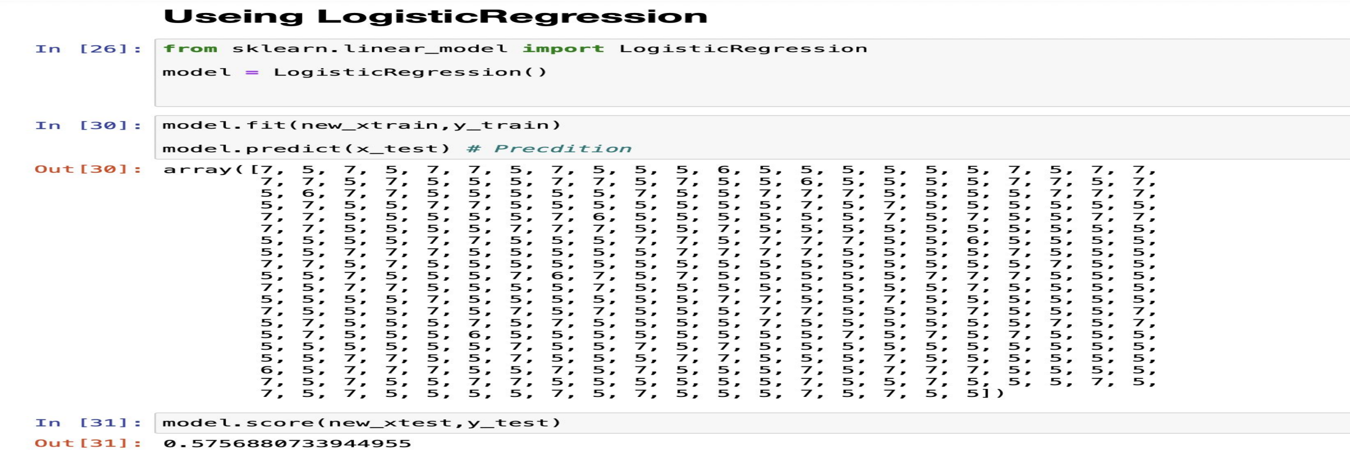
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=40)
```

Normalization

We do normalization on numerical data because our data is unbalanced it means the difference between the variable values is high so we convert them into 1 and 0.

```
#importing module
from sklearn.preprocessing import MinMaxScaler
# creating normalization object
norm = MinMaxScaler()
# fit data
norm_fit = norm.fit(x_train)
new_xtrain = norm_fit.transform(x_train)
new_xtest = norm_fit.transform(x_test)
# display values
print(new_xtrain)
```

Applying model



Useing LogisticRegression

```
In [26]: from sklearn.linear_model import LogisticRegression
         model = LogisticRegression()

In [30]: model.fit(new_xtrain,y_train)
         model.predict(x_test) # Precdition
Out[30]: array([7, 5, 7, 5, 7, 7, 5, 7, 5, 5, 5, 6, 5, 5, 5, 5, 5, 5, 7, 5, 7, 7,
                7, 7, 5, 7, 5, 5, 5, 7, 7, 5, 7, 5, 5, 6, 5, 5, 5, 5, 7, 7, 5, 7,
                5, 6, 7, 7, 5, 5, 5, 5, 5, 7, 5, 5, 7, 7, 7, 5, 5, 5, 5, 7, 7, 7,
                5, 7, 5, 5, 7, 7, 5, 5, 5, 5, 5, 5, 5, 7, 5, 7, 5, 5, 5, 5, 5, 5,
                7, 7, 5, 5, 5, 5, 5, 7, 6, 5, 5, 5, 5, 5, 5, 7, 5, 7, 5, 5, 7, 7,
                7, 7, 5, 5, 5, 5, 7, 7, 7, 5, 5, 7, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                5, 5, 5, 5, 7, 7, 5, 5, 5, 7, 7, 5, 7, 7, 7, 5, 5, 6, 5, 5, 5, 5,
                5, 5, 7, 7, 7, 5, 5, 5, 5, 5, 7, 7, 7, 7, 5, 5, 5, 5, 7, 5, 5, 5,
                7, 7, 5, 7, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 7, 5, 5,
                5, 5, 7, 5, 5, 5, 7, 6, 7, 5, 7, 5, 5, 5, 5, 5, 7, 7, 7, 5, 5, 5,
                7, 5, 7, 7, 5, 5, 5, 5, 7, 5, 5, 5, 5, 5, 5, 5, 5, 7, 5, 5, 5, 5,
                5, 5, 5, 5, 7, 5, 5, 5, 5, 5, 5, 7, 7, 5, 5, 7, 5, 5, 5, 5, 5, 5,
                7, 5, 5, 5, 7, 5, 7, 5, 7, 5, 5, 5, 7, 7, 5, 5, 5, 7, 5, 5, 5, 7,
                5, 7, 5, 5, 5, 7, 5, 7, 5, 5, 5, 5, 7, 5, 5, 5, 5, 5, 5, 7, 5, 7,
                5, 7, 5, 5, 5, 6, 5, 5, 5, 5, 5, 5, 5, 5, 7, 5, 7, 5, 7, 5, 5, 5,
                5, 5, 5, 5, 5, 5, 7, 5, 5, 7, 5, 7, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                5, 5, 7, 7, 5, 5, 7, 5, 5, 5, 7, 7, 5, 5, 5, 7, 5, 5, 5, 5, 5, 5,
                6, 5, 7, 7, 7, 5, 5, 7, 5, 7, 5, 5, 5, 7, 5, 7, 7, 7, 5, 5, 5, 5,
                7, 5, 7, 5, 5, 7, 7, 5, 5, 5, 5, 5, 5, 7, 5, 5, 7, 5, 5, 5, 7, 5,
                7, 5, 7, 5, 5, 5, 5, 7, 5, 7, 5, 5, 5, 7, 5, 7, 5, 5])

In [31]: model.score(new_xtest,y_test)
Out[31]: 0.5756880733944955
```

References :

The media shown in this article are not owned by Analytics Vidhya and is used at the Author’s discretion.

Thanks you !!