# Assignment 2: Logistic Regression

*Shiva Chakravarty Gollapudi*
*Student ID: 11468697*

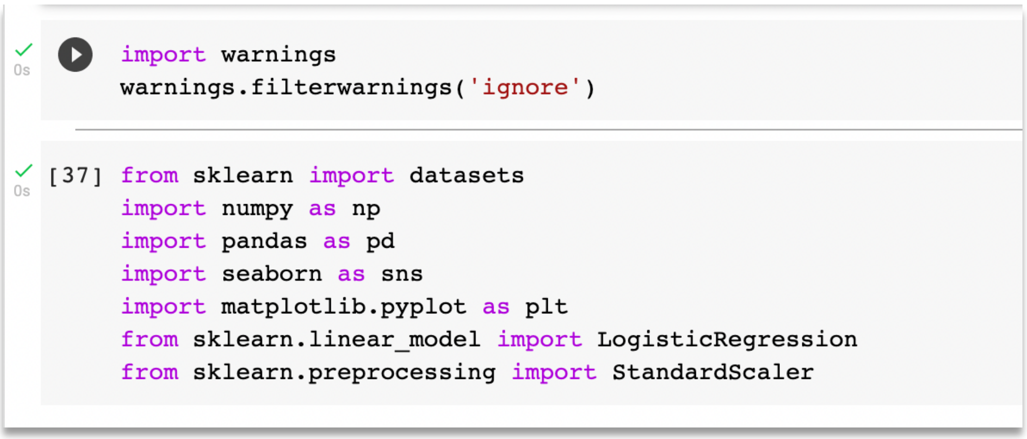
Logistic regression is used to when the target variable is categorical.

Using the data provided created Logistic Regression model and evaluated the output.

Data: Predict the quality of red wines

Tools and Environment: Using python programming and Google Colab, I have explored the data and created Linear Regression models.

## 1. Load Libraries:

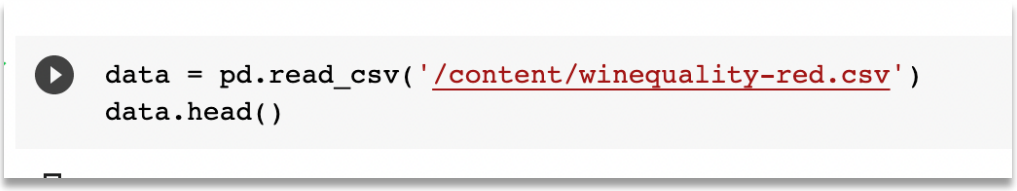


```
import warnings
warnings.filterwarnings('ignore')
```

```
[37] from sklearn import datasets
     import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.linear_model import LogisticRegression
     from sklearn.preprocessing import StandardScaler
```

Imported the libraries which will useful to build the model and data analysis.

## 2. Load Data:



```
data = pd.read_csv('/content/winequality-red.csv')
data.head()
```

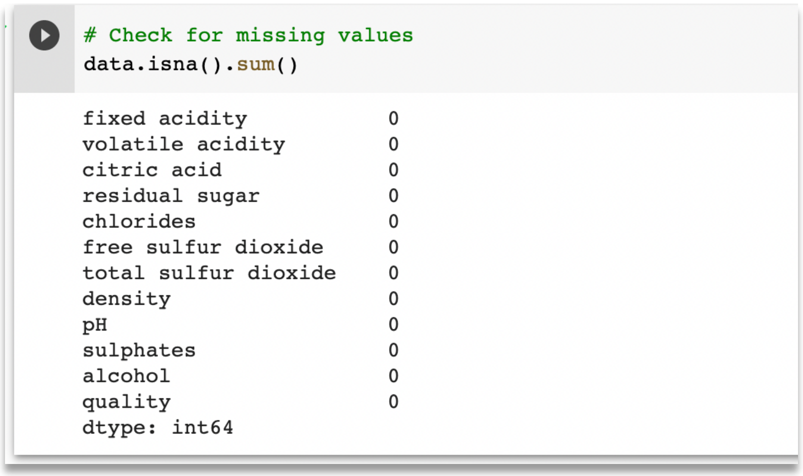
Data Description: 'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol' are independent variables. 'Quality' is the dependent variable.

## 3. Data Frame Analysis:

Investigate the data

1. Checking for missing values.
2. Checking for correlation, plot heat map and properties.

### 1. Checking for missing values:



```
# Check for missing values
data.isna().sum()
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

There are no missing values.

### 2. Checking for correlation, plot heat map and properties:

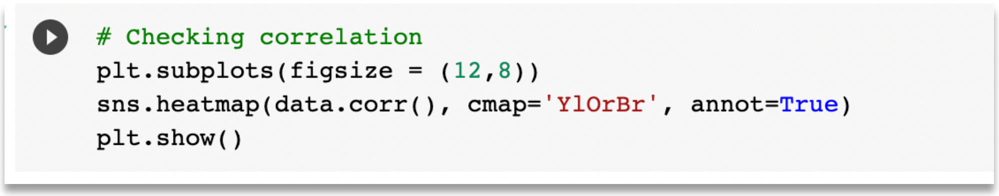
Correlation, which assesses the strength of the relationship between two variables.

Coefficient correlation range:

- If the coefficient of the correlation is greater than zero, then there is a positive relation among the variables.
- If the coefficient of the correlation is less than zero, then there is a negative relation among the variables.
- If the coefficient of the correlation equal to zero, then there is no relation among the variables.

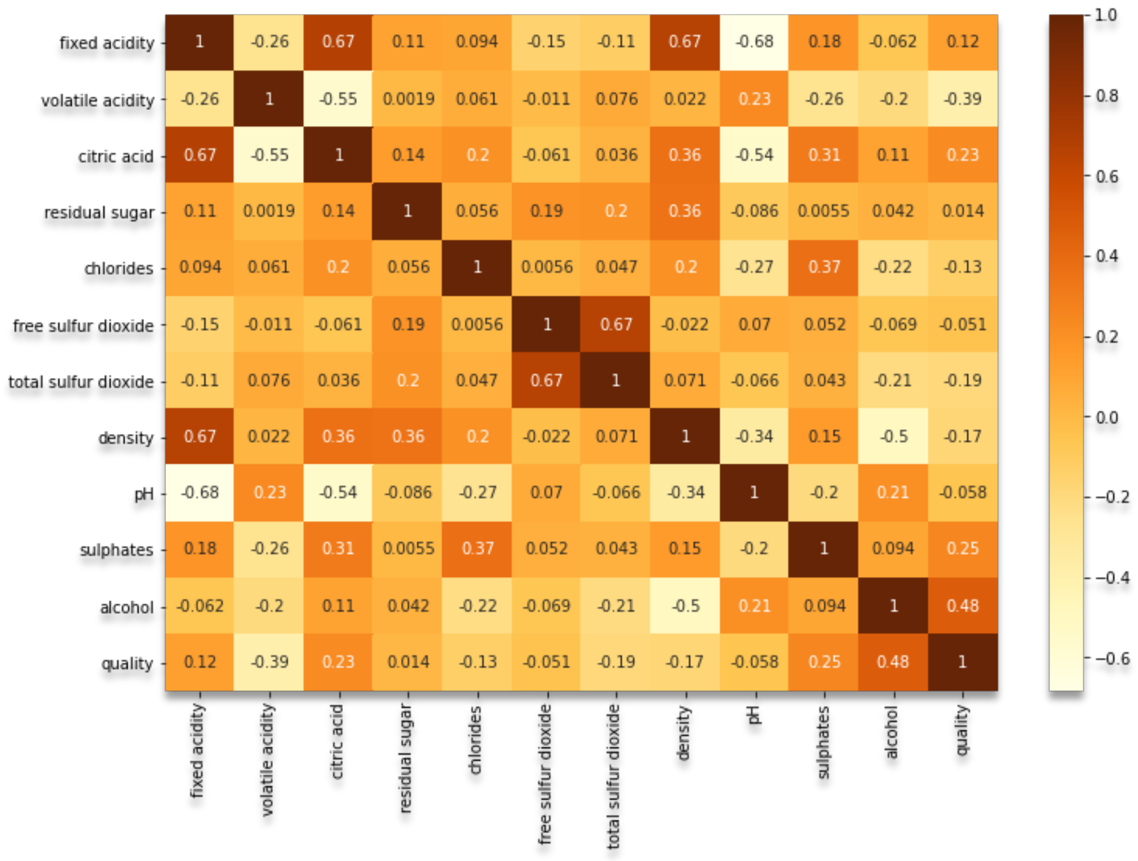
  Using Heatmap with Seaborn library, we have identified the correlation between the variables.

CODE:



```
# Checking correlation
plt.subplots(figsize = (12,8))
sns.heatmap(data.corr(), cmap='YlOrBr', annot=True)
plt.show()
```

OUTPUT:



| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| fixed acidity | 1 | -0.26 | 0.67 | 0.11 | 0.094 | -0.15 | -0.11 | 0.67 | -0.68 | 0.18 | -0.062 | 0.12 |
| volatile acidity | -0.26 | 1 | -0.55 | 0.0019 | 0.061 | -0.011 | 0.076 | 0.022 | 0.23 | -0.26 | -0.2 | -0.39 |
| citric acid | 0.67 | -0.55 | 1 | 0.14 | 0.2 | -0.061 | 0.036 | 0.36 | -0.54 | 0.31 | 0.11 | 0.23 |
| residual sugar | 0.11 | 0.0019 | 0.14 | 1 | 0.056 | 0.19 | 0.2 | 0.36 | -0.086 | 0.0055 | 0.042 | 0.014 |
| chlorides | 0.094 | 0.061 | 0.2 | 0.056 | 1 | 0.0056 | 0.047 | 0.2 | -0.27 | 0.37 | -0.22 | -0.13 |
| free sulfur dioxide | -0.15 | -0.011 | -0.061 | 0.19 | 0.0056 | 1 | 0.67 | -0.022 | 0.07 | 0.052 | -0.069 | -0.051 |
| total sulfur dioxide | -0.11 | 0.076 | 0.036 | 0.2 | 0.047 | 0.67 | 1 | 0.071 | -0.066 | 0.043 | -0.21 | -0.19 |
| density | 0.67 | 0.022 | 0.36 | 0.36 | 0.2 | -0.022 | 0.071 | 1 | -0.34 | 0.15 | -0.5 | -0.17 |
| pH | -0.68 | 0.23 | -0.54 | -0.086 | -0.27 | 0.07 | -0.066 | -0.34 | 1 | -0.2 | 0.21 | -0.058 |
| sulphates | 0.18 | -0.26 | 0.31 | 0.0055 | 0.37 | 0.052 | 0.043 | 0.15 | -0.2 | 1 | 0.094 | 0.25 |
| alcohol | -0.062 | -0.2 | 0.11 | 0.042 | -0.22 | -0.069 | -0.21 | -0.5 | 0.21 | 0.094 | 1 | 0.48 |
| quality | 0.12 | -0.39 | 0.23 | 0.014 | -0.13 | -0.051 | -0.19 | -0.17 | -0.058 | 0.25 | 0.48 | 1 |

From the above heatmap, we see that 'volatile acidity', 'citric acid', 'sulphates', 'alcohol' are most correlated with 'quality', the target variable. 'fixed acidity', 'chlorides', 'total sulfur dioxide', 'density' is less correlated.

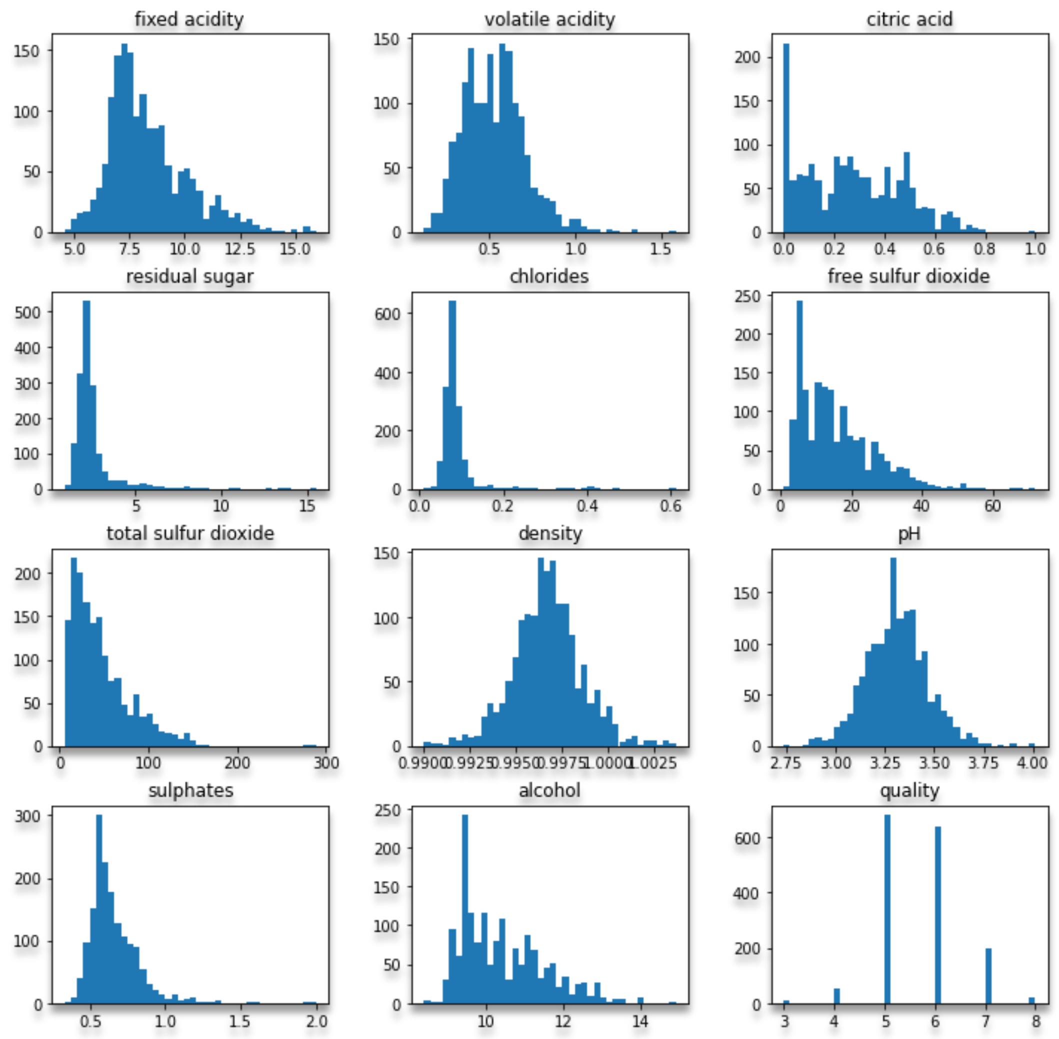
Few independent variables are likely multicollinear to each other that can be shown with the high correlation scores. Pairs of multicollinear independent variables are

1. 'total sulfur dioxide' and 'free sulfur dioxide',
2. 'fixed acidity' and 'citric acid',
3. 'fixed acidity' and 'density',
4. 'fixed acidity' and 'pH'.

These may affect the model, so we can exclude the variables which are having the high correlation score to generate significant model.

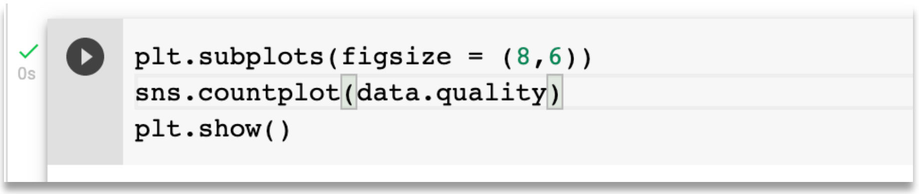
## 4. Numerical Variable Analysis:

Numerical variable is done to check the data distribution of the numerical variables.



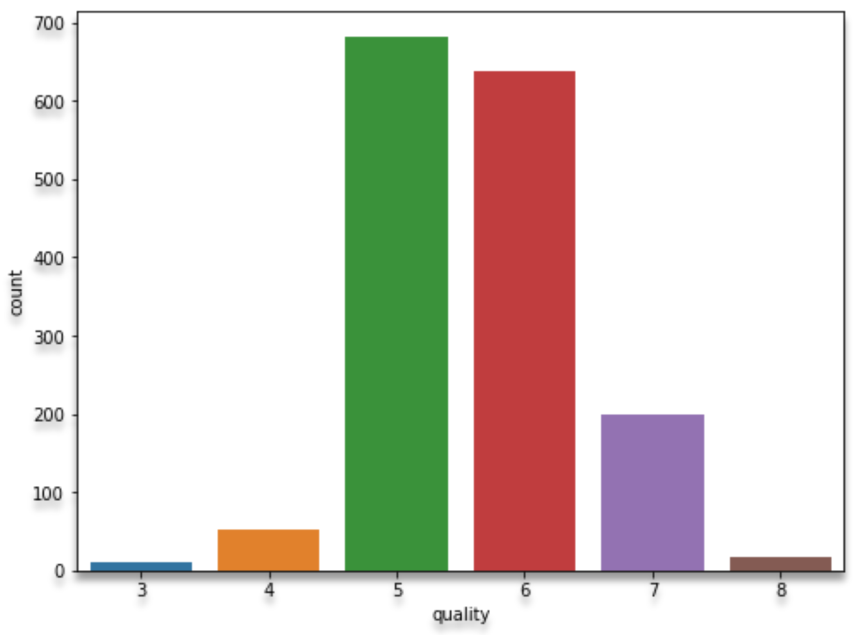
From above histograms, we can observe that density and PH values are distributed normally rest all variables are skewed. Quality is a categorical data.

## 5. Categorical Variable Analysis:



```
plt.subplots(figsize = (8,6))
sns.countplot(data.quality)
plt.show()
```

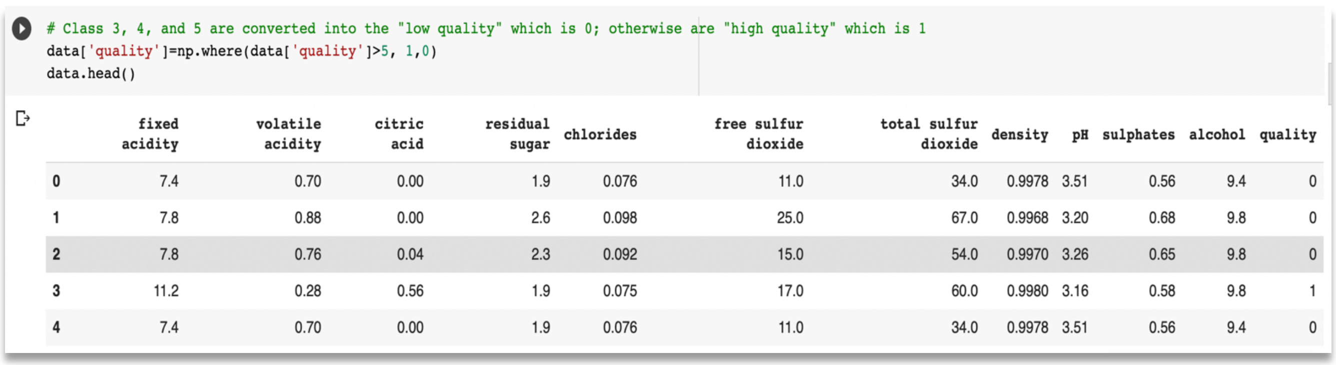
OUTPUT:



From the obtained data, the wine quality is more in 5 and 6.

## Encoding:

Convert the Target variable (categorical data) into binary. Class 3, 4, and 5 are converted into the "low quality" which is 0; otherwise, are "high quality" which is 1.



```
# Class 3, 4, and 5 are converted into the "low quality" which is 0; otherwise are "high quality" which is 1
data['quality']=np.where(data['quality']>5, 1,0)
data.head()
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | 0.56 | 9.4 | 0 |
| **1** | 7.8 | 0.88 | 0.00 | 2.6 | 0.098 | 25.0 | 67.0 | 0.9968 | 3.20 | 0.68 | 9.8 | 0 |
| **2** | 7.8 | 0.76 | 0.04 | 2.3 | 0.092 | 15.0 | 54.0 | 0.9970 | 3.26 | 0.65 | 9.8 | 0 |
| **3** | 11.2 | 0.28 | 0.56 | 1.9 | 0.075 | 17.0 | 60.0 | 0.9980 | 3.16 | 0.58 | 9.8 | 1 |
| **4** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | 0.56 | 9.4 | 0 |

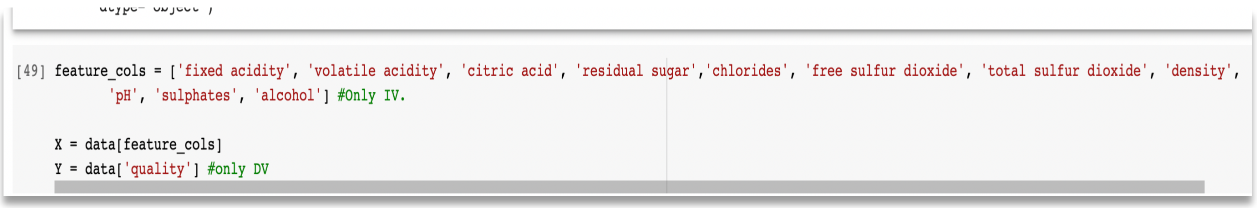
**6. Model Building:**

1. Separate independent variable and dependent variable.
2. split the data
3. import the libraries for the algorithm.
4. invoke the function.
5. Fit on Train dataset.
6. Generate predictions using IV.
7. Check accuracy metrics.

   I. Classification - Accuracy, ROC and AUC
8. Plot the accuracy metrics

1. Separate independent variable and dependent variable:

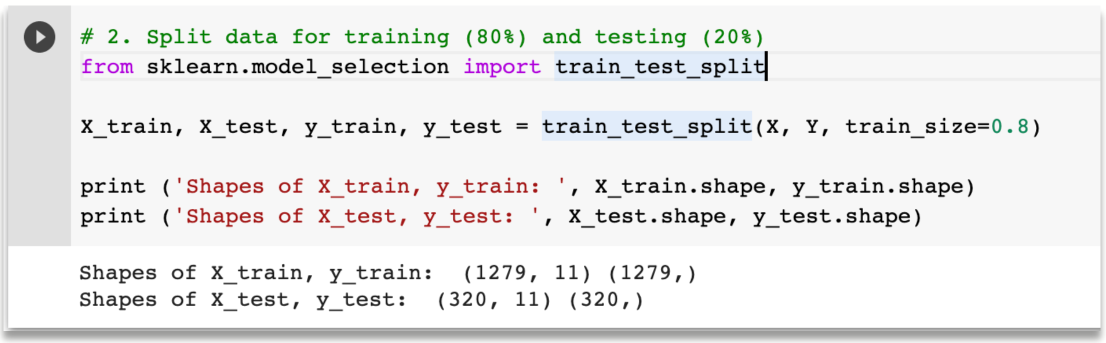
Before creating the model, separated the dependent variable (DV) and independent variable (IV).



```
[49] feature_cols = ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar','chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
            'pH', 'sulphates', 'alcohol'] #Only IV.

     X = data[feature_cols]
     Y = data['quality'] #only DV
```

2. Split the data:

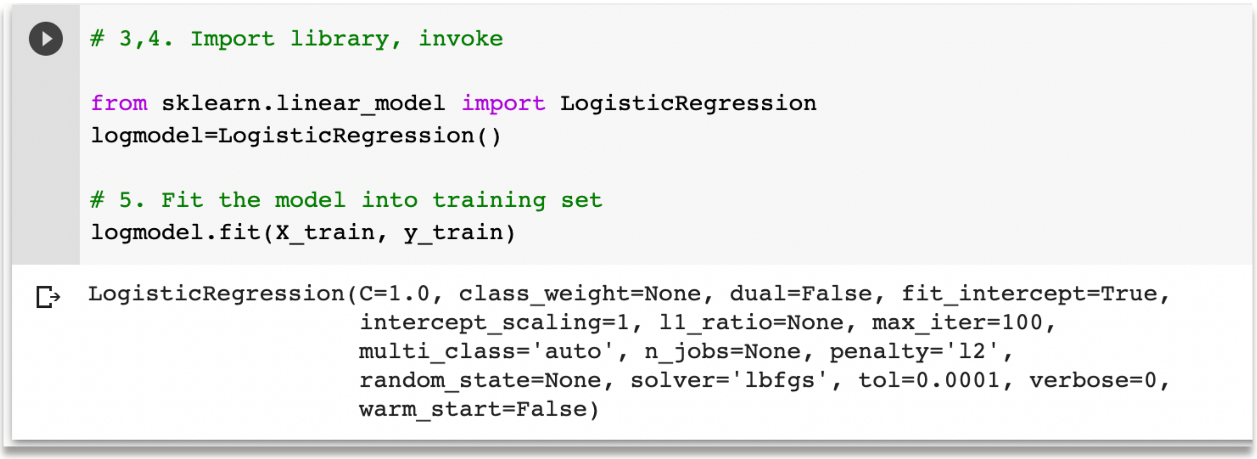
Split data for training (80%) and testing (20%)



```
# 2. Split data for training (80%) and testing (20%)
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, Y, train_size=0.8)

print ('Shapes of X_train, y_train: ', X_train.shape, y_train.shape)
print ('Shapes of X_test, y_test: ', X_test.shape, y_test.shape)
```

```
Shapes of X_train, y_train:  (1279, 11) (1279,)
Shapes of X_test, y_test:  (320, 11) (320,)
```

3, 4 and 5: Import the libraries, invoke and fit data to the model:

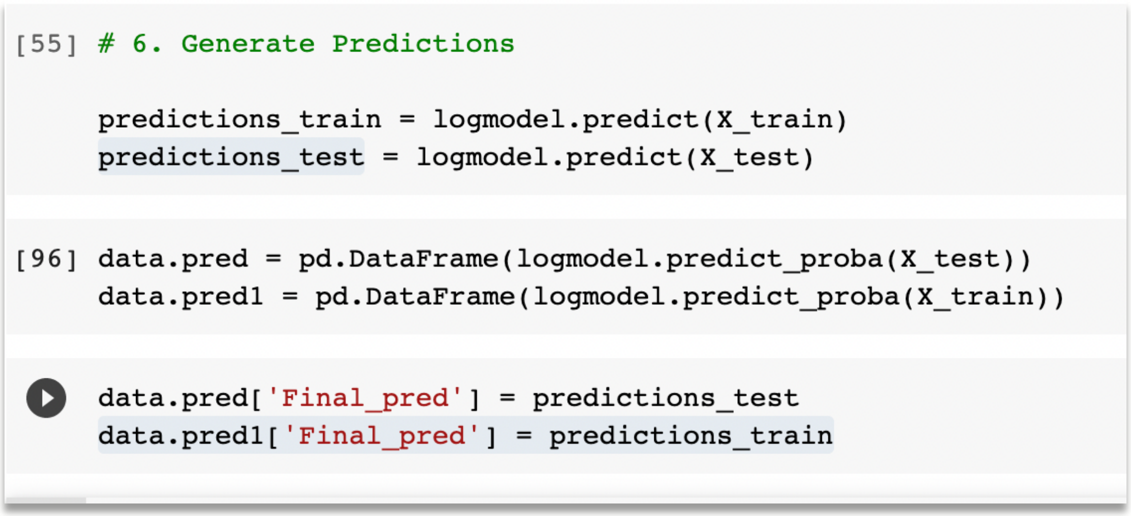


```
# 3,4. Import library, invoke

from sklearn.linear_model import LogisticRegression
logmodel=LogisticRegression()

# 5. Fit the model into training set
logmodel.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

6. Generate Predictions:

Generated the predictions for both test and train data.



```
[55] # 6. Generate Predictions

     predictions_train = logmodel.predict(X_train)
     predictions_test = logmodel.predict(X_test)
```

```
[96] data.pred = pd.DataFrame(logmodel.predict_proba(X_test))
     data.pred1 = pd.DataFrame(logmodel.predict_proba(X_train))
```

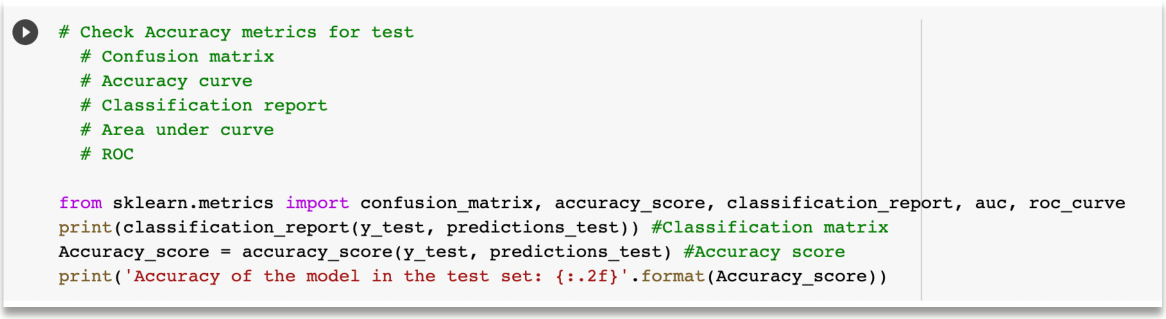
```
data.pred['Final_pred'] = predictions_test
data.pred1['Final_pred'] = predictions_train
```

7. Check accuracy metrics.

Classification - Accuracy, ROC and AUC.

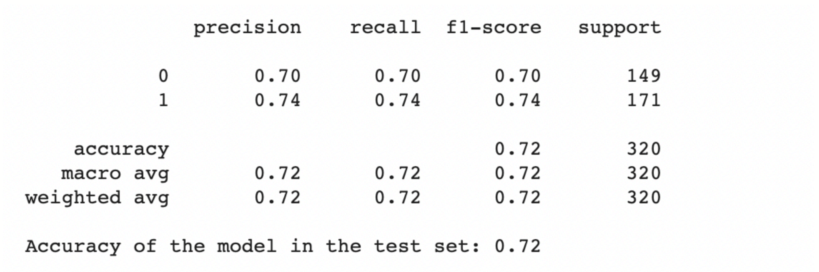
I have checked accuracy, ROC and AUC for both train and test data.

CODE for test data:



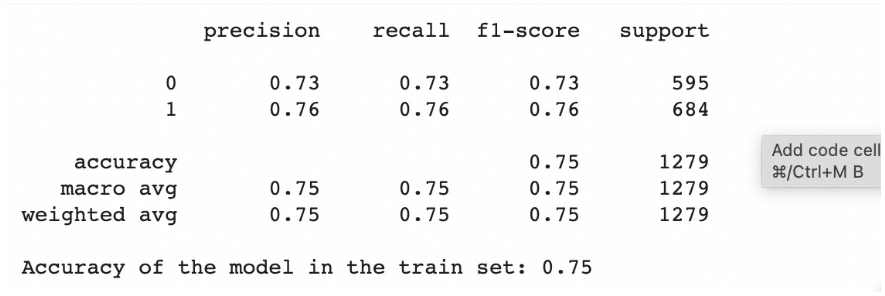
```
# Check Accuracy metrics for test
  # Confusion matrix
  # Accuracy curve
  # Classification report
  # Area under curve
  # ROC

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, auc, roc_curve
print(classification_report(y_test, predictions_test)) #Classification matrix
Accuracy_score = accuracy_score(y_test, predictions_test) #Accuracy score
print('Accuracy of the model in the test set: {:.2f}'.format(Accuracy_score))
```

OUTPUT: Accuracy for test data is 72%.



```
              precision    recall  f1-score   support

           0       0.70      0.70      0.70       149
           1       0.74      0.74      0.74       171

    accuracy                           0.72       320
   macro avg       0.72      0.72      0.72       320
weighted avg       0.72      0.72      0.72       320

Accuracy of the model in the test set: 0.72
```

OUTPUT: Accuracy for train data is 75%.

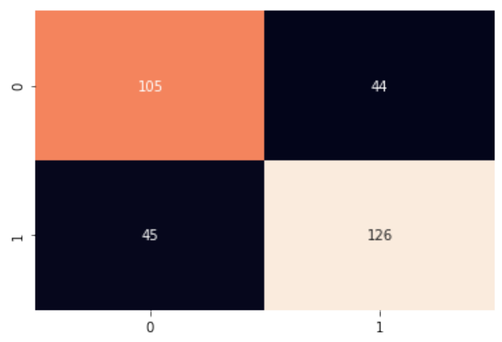


```
              precision    recall  f1-score   support

           0       0.73      0.73      0.73       595
           1       0.76      0.76      0.76       684

    accuracy                           0.75      1279
   macro avg       0.75      0.75      0.75      1279
weighted avg       0.75      0.75      0.75      1279

Accuracy of the model in the train set: 0.75
```

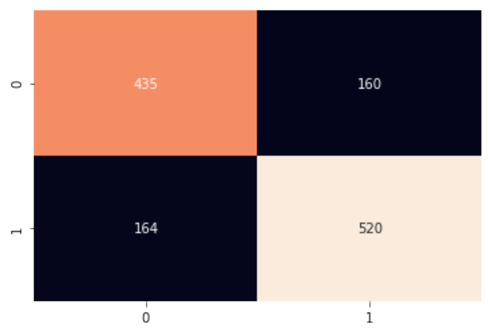
Confusion matrix: Confusion matrix is used to describe the performance of a classification model.

For test confusion matrix: TN = 105, FP = 44, FN= 45 and TP = 126.

For test confusion matrix: TN = 435, FP = 160 FN= 164 and TP = 520.



*Confusion matrix for test data*



*Confusion matrix for train data*

TPR: TPR is the probability that an actual positive will test positive.

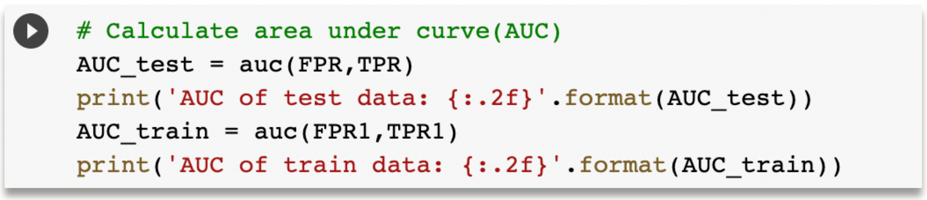
$$\text{TPR} = \frac{TP}{TP+FN}$$

FPR: FPR is the model mistakenly predicted the positive class

$$\text{FPR} = \frac{FP}{TN+FP}$$

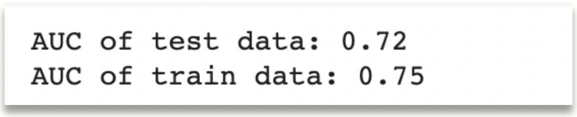
Area Under Curve (AUC):

CODE:



```
# Calculate area under curve(AUC)
AUC_test = auc(FPR,TPR)
print('AUC of test data: {:.2f}'.format(AUC_test))
AUC_train = auc(FPR1,TPR1)
print('AUC of train data: {:.2f}'.format(AUC_train))
```

OUTPUT:



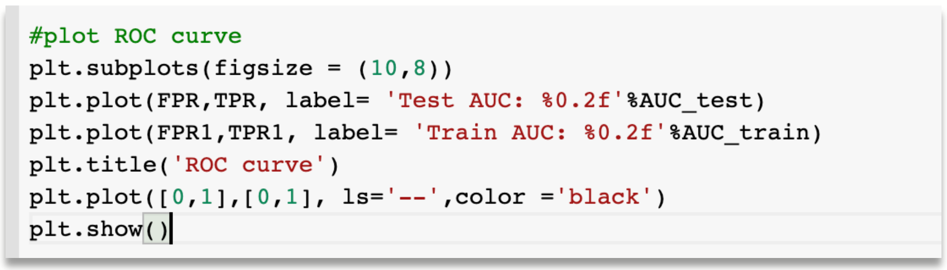
```
AUC of test data: 0.72
AUC of train data: 0.75
```

Area under curve for test data is 0.72, and for train data in 0.75.

ROC Curve:

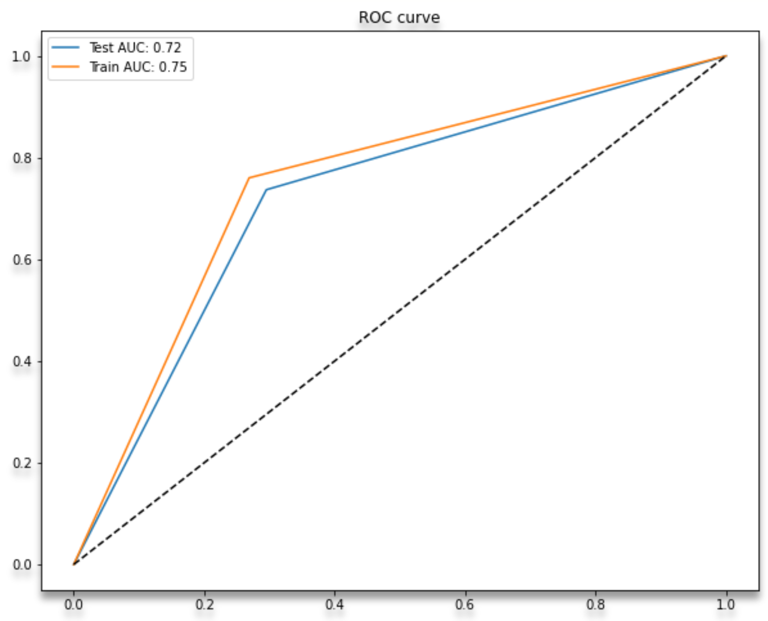
An ROC (Receiver Operating Characteristic Curve) is a graph showing the performance of a classification model at all classification thresholds.

CODE:



```
#plot ROC curve
plt.subplots(figsize = (10,8))
plt.plot(FPR,TPR, label= 'Test AUC: %0.2f'%AUC_test)
plt.plot(FPR1,TPR1, label= 'Train AUC: %0.2f'%AUC_train)
plt.title('ROC curve')
plt.plot([0,1],[0,1], ls='--',color ='black')
plt.show()
```

OUTPUT:



Classification report:

F1 score will be the best value at 1 and the worst value at 0. For the test data F1 score is 0.72 and for the training data is 0.75.