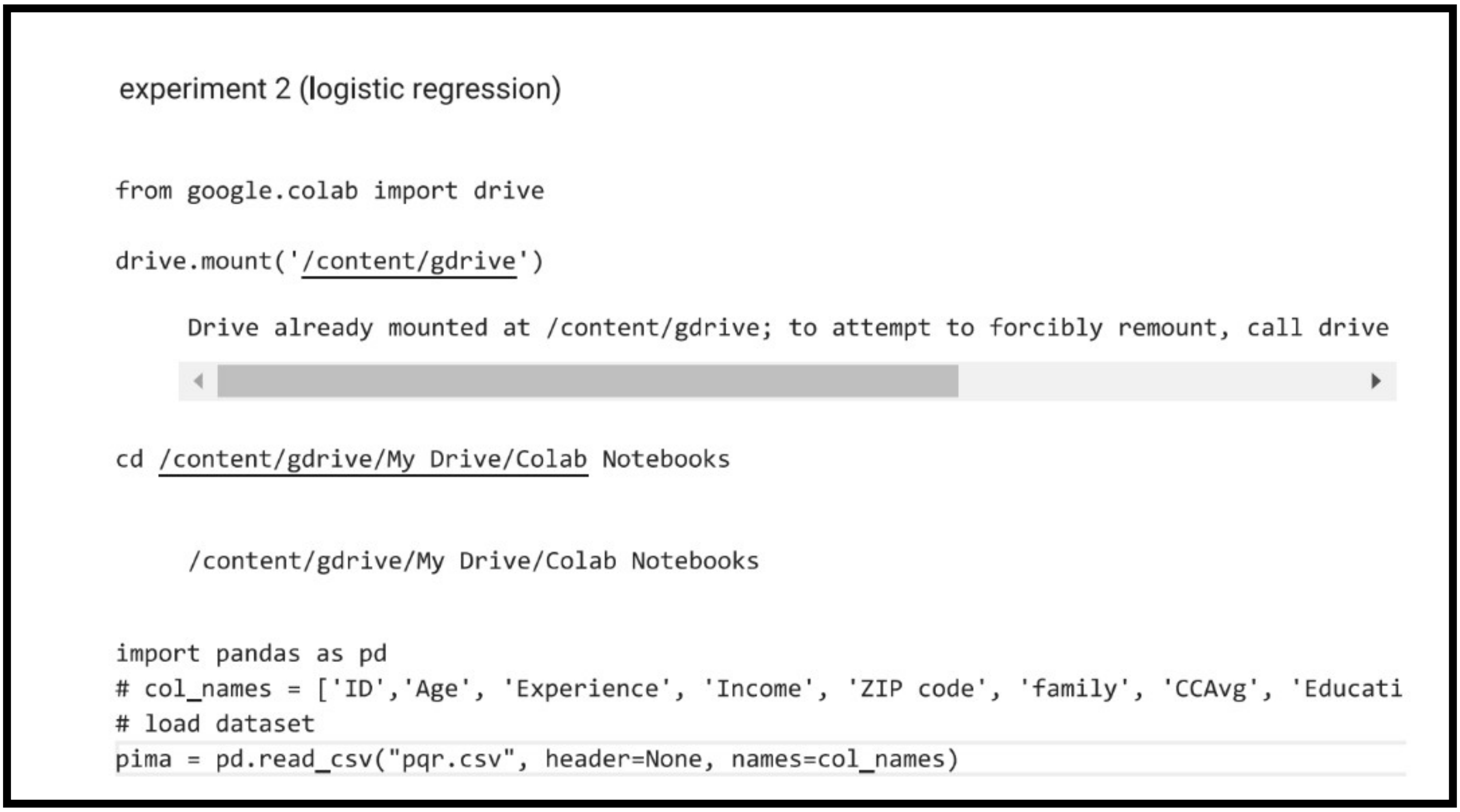
Experiment number:2

Aim: To perform various types of Logistic Regression:

Theory: Logistic regression **estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables**. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.

- In this logistic Regression Experiment I have used dataset of Bank Personal Loan Modelling .
- It has 5000 row and 14 columns.

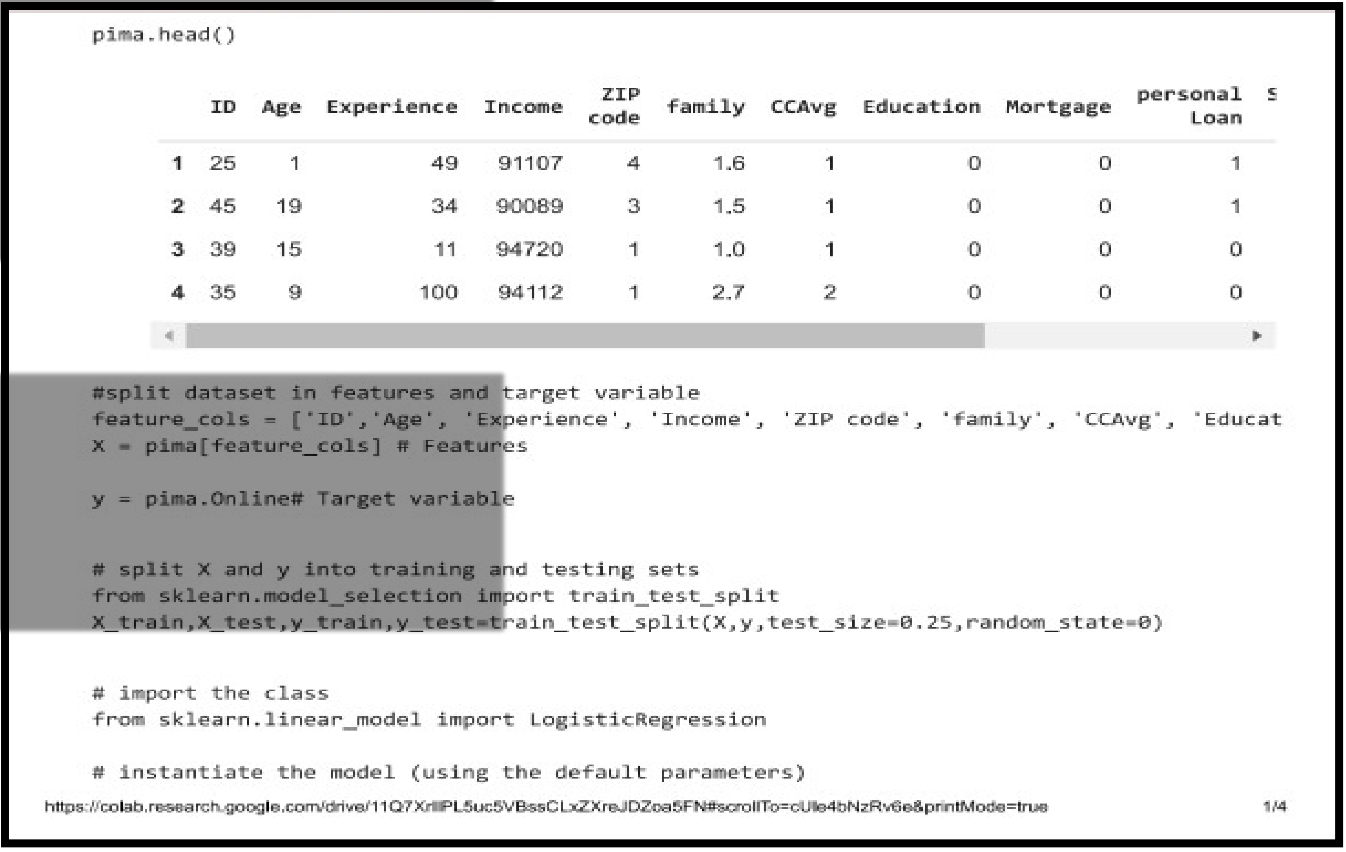


experiment 2 (logistic regression)

```
from google.colab import drive

drive.mount('/content/gdrive')

     Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive
```

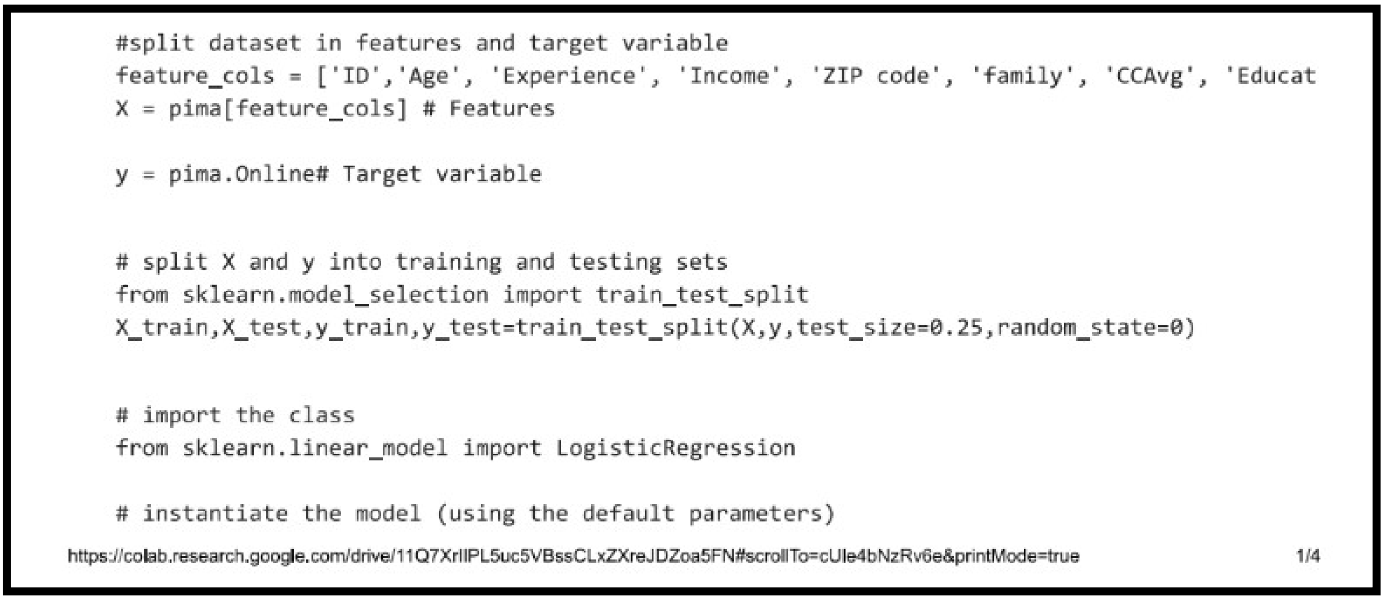
```
cd /content/gdrive/My Drive/Colab Notebooks


     /content/gdrive/My Drive/Colab Notebooks
```

```
import pandas as pd
# col_names = ['ID','Age', 'Experience', 'Income', 'ZIP code', 'family', 'CCAvg', 'Educati
# load dataset
pima = pd.read_csv("pqr.csv", header=None, names=col_names)
```



```
pima.head()

     ID  Age  Experience  Income   ZIP   family  CCAvg  Education  Mortgage  personal  S
                                   code                                        Loan

  1  25    1          49   91107     4      1.6      1          0         0         1
  2  45   19          34   90089     3      1.5      1          0         0         1
  3  39   15          11   94720     1      1.0      1          0         0         0
  4  35    9         100   94112     1      2.7      2          0         0         0
```

```
#split dataset in features and target variable
feature_cols = ['ID','Age', 'Experience', 'Income', 'ZIP code', 'family', 'CCAvg', 'Educat
X = pima[feature_cols] # Features

y = pima.Online# Target variable


# split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)


# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
```



#split dataset in features and target variable
feature_cols = ['ID','Age', 'Experience', 'Income', 'ZIP code', 'family', 'CCAvg', 'Educat
X = pima[feature_cols] # Features

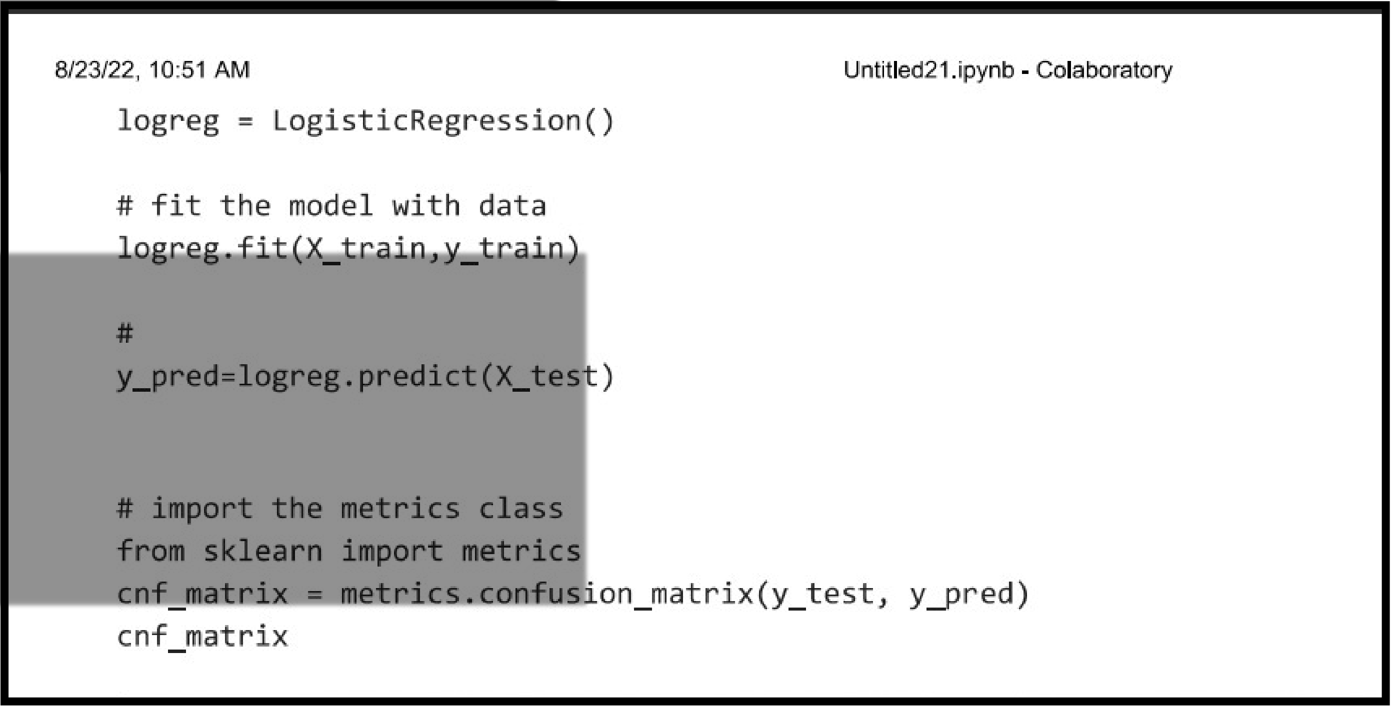
y = pima.Online# Target variable

# split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)

# import the class
from sklearn.linear_model import LogisticRegression

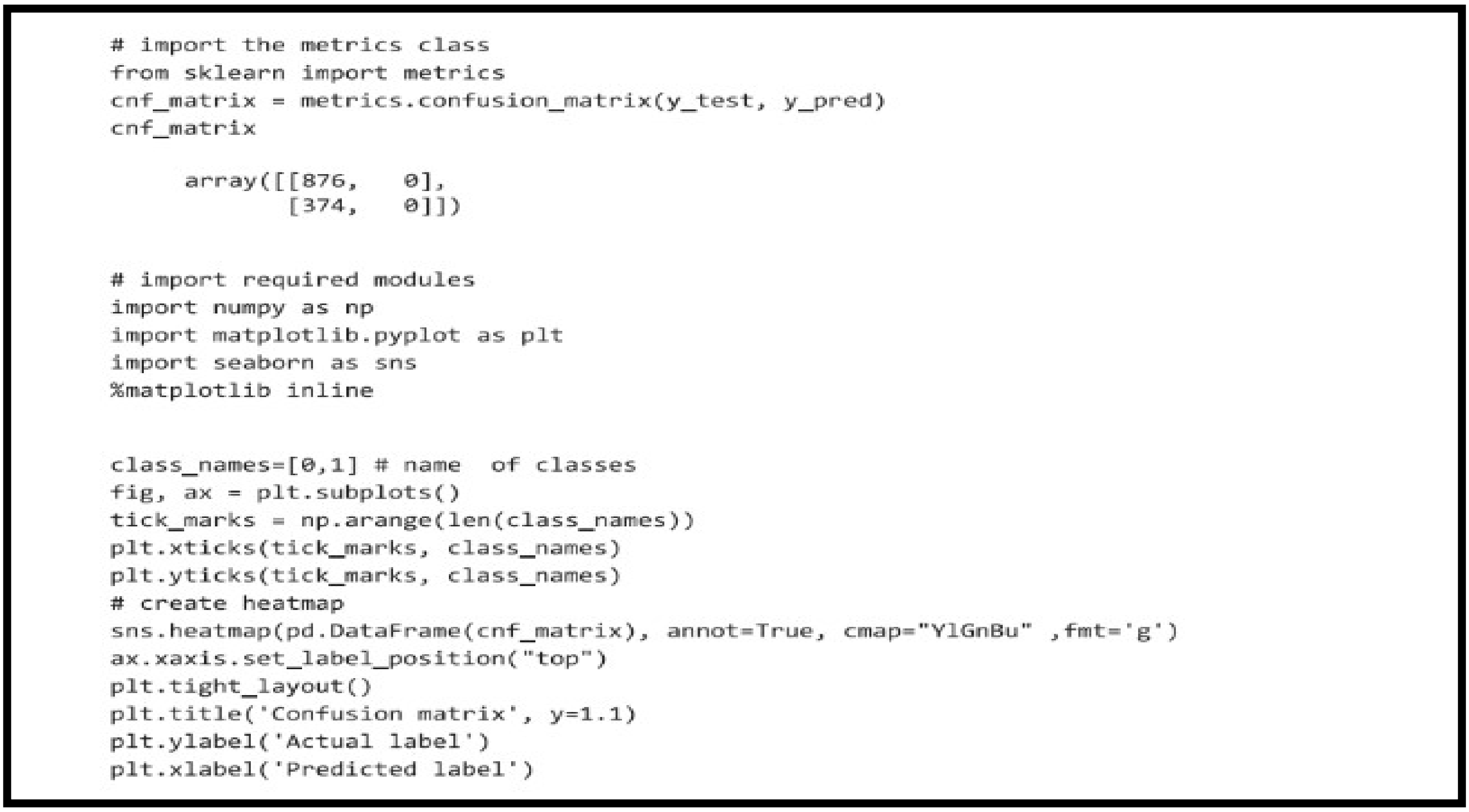
instantiate the model (using the default parameters)

https://colab.research.google.com/drive/11Q7XrlIPL5uc5VBssCLxZXreJDZoa5FN#scrollTo=cUle4bNzRv6e&printMode=true 1/4

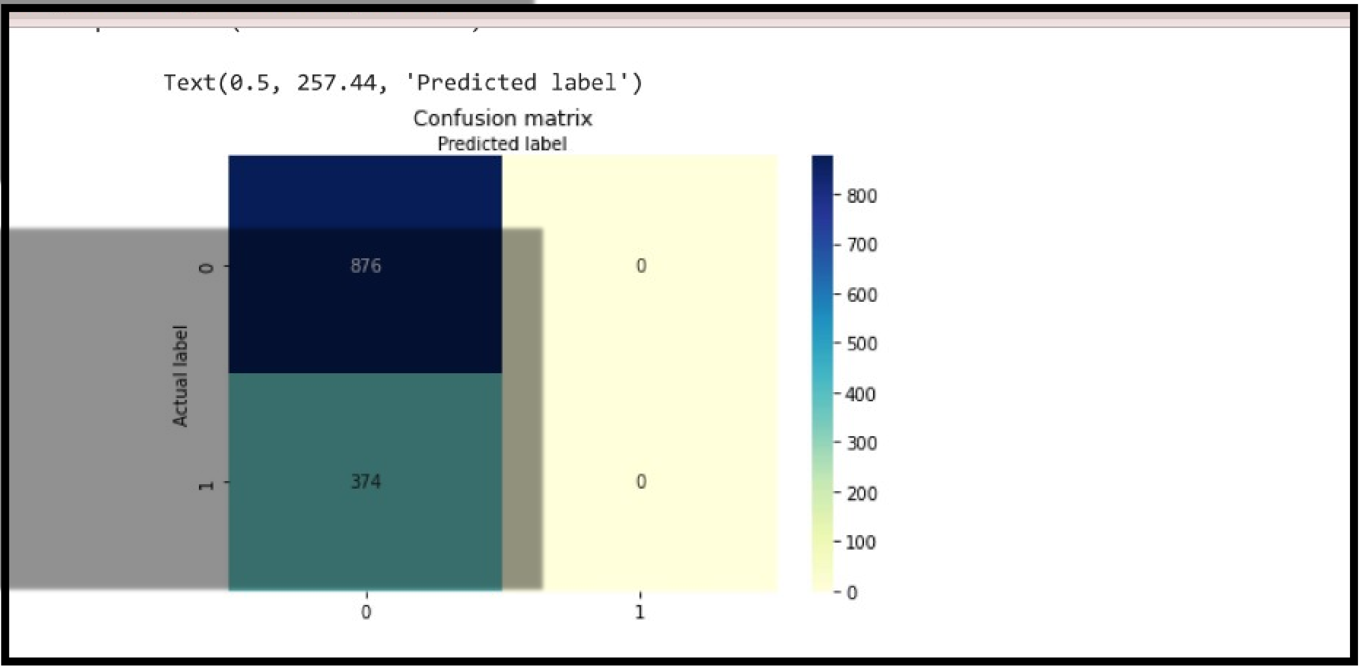


8/23/22, 10:51 AM Untitled21.ipynb - Colaboratory

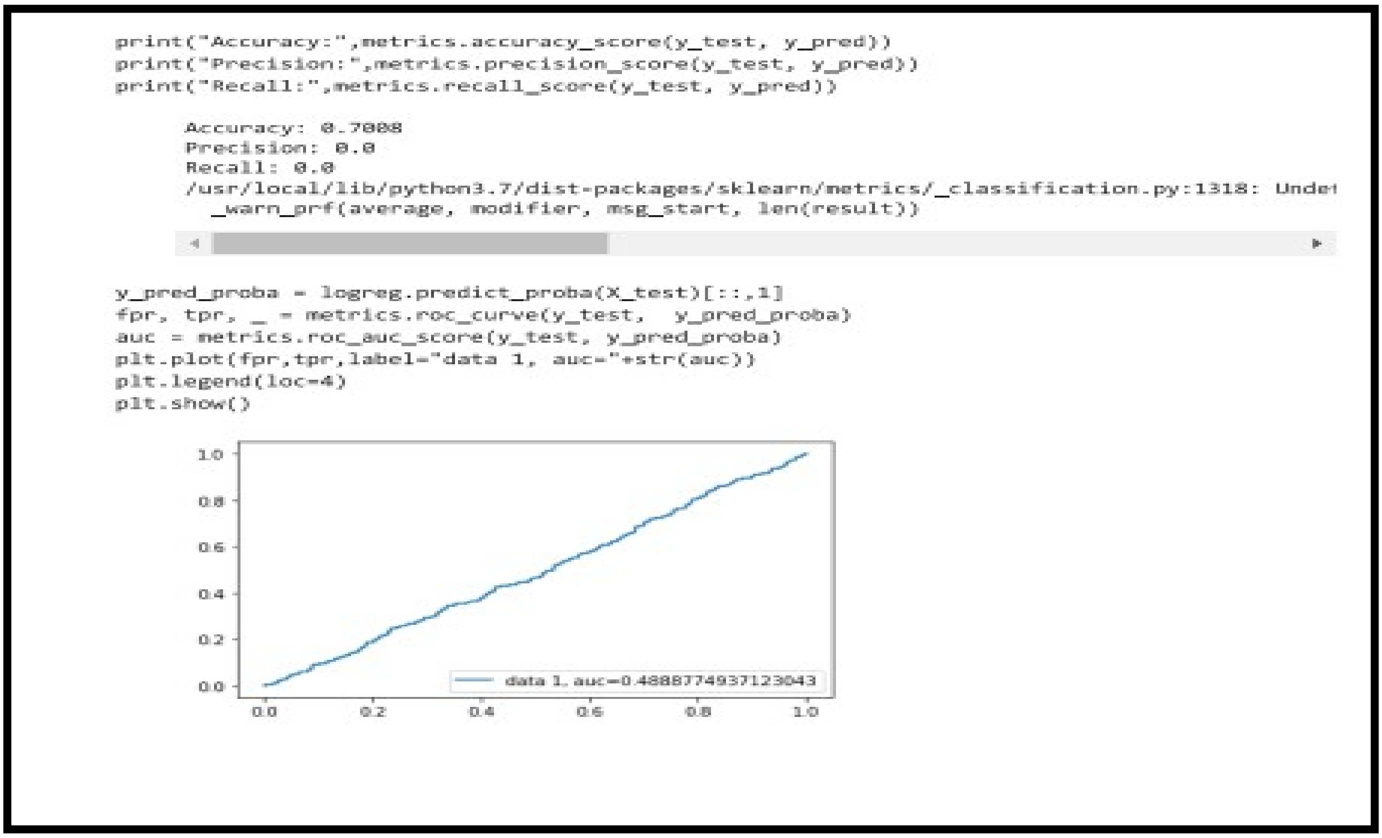
```
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X_train,y_train)

#
y_pred=logreg.predict(X_test)


# import the metrics class
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```



```
# import the metrics class
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix

     array([[876,   0],
            [374,   0]])


# import required modules
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline


class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```



```
Text(0.5, 257.44, 'Predicted label')
```



```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.7008
Precision: 0.0
Recall: 0.0
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Unde
  _warn_prf(average, modifier, msg_start, len(result))
```

```
y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```