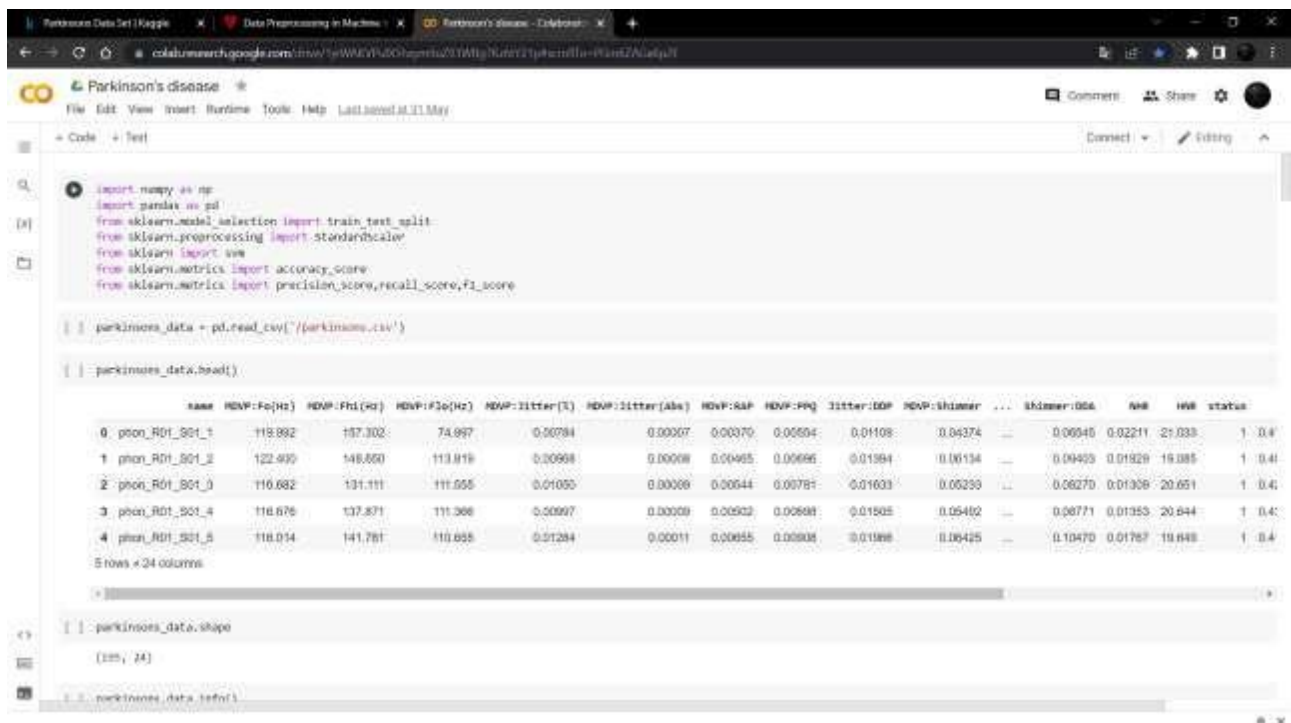


CHAPTER 6

CODING & SCREENSHOTS

The following code has been implemented in python using GOOGLE COLAB

6.1 SAMPLE CODING FOR PARKINSON DISEASE DETECTION



```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score

parkinsons_data = pd.read_csv('parkinsons.csv')

parkinsons_data.head()
```

	name	MDVP:F0(Hz)	MDVP:F1(Hz)	MDVP:F2(Hz)	MDVP:Dttr(L)	MDVP:Dttr(Abs)	MDVP:RAP	MDVP:RPQ	Dttr:DDP	MDVP:Shimmer	...	Shimmer:DDA	Age	HW	status	
0	phon_R01_S01_1	113.892	137.302	74.997	0.06784	0.00007	0.00070	0.00504	0.01108	0.04374	...	0.06940	0.02211	21.033	1	0.4
1	phon_R01_S01_2	122.490	148.660	113.819	0.00998	0.00008	0.00465	0.00696	0.01994	0.00134	...	0.09403	0.01929	19.085	1	0.4
2	phon_R01_S01_3	110.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.00293	...	0.06270	0.01306	20.651	1	0.4
3	phon_R01_S01_4	118.676	137.871	111.368	0.00997	0.00009	0.00932	0.00608	0.01505	0.05402	...	0.06771	0.01355	20.644	1	0.4
4	phon_R01_S01_5	118.014	141.781	110.655	0.01284	0.00011	0.00655	0.00808	0.01986	0.06425	...	0.10470	0.01767	19.848	1	0.4

```
parkinsons_data.shape
(195, 24)

parkinsons_data.info()
```

Figure 6.1.1 Importing Dataset

```

parkinsons_data.shape

(195, 24)

parkinsons_data.info()
Out[1]:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 195 entries, 0 to 194
Data columns (total 24 columns):
 #   Column              Non-Null Count  Dtype  
---  -
 0   name                 195 non-null    object  
 1   PDVP: Fo(Hz)         195 non-null    float64  
 2   PDVP: Fb1(Hz)        195 non-null    float64  
 3   PDVP: Fb2(Hz)        195 non-null    float64  
 4   PDVP: Jitter(X)      195 non-null    float64  
 5   PDVP: Jitter(Abs)    195 non-null    float64  
 6   PDVP: RAP            195 non-null    float64  
 7   PDVP: PPQ            195 non-null    float64  
 8   Jitter: DDP          195 non-null    float64  
 9   PDVP: Shimmer        195 non-null    float64  
10  PDVP: Shimmer(dB)    195 non-null    float64  
11  Shimmer: APQ2        195 non-null    float64  
12  Shimmer: APQ5        195 non-null    float64  
13  PDVP: APQ            195 non-null    float64  
14  Shimmer: DDA         195 non-null    float64  
15  HNR                  195 non-null    float64  
16  HNR                  195 non-null    float64  
17  status              195 non-null    int64  
18  RPSE                195 non-null    float64  
19  DFA                 195 non-null    float64  
20  spread1             195 non-null    float64  
21  spread2             195 non-null    float64  
22  D2                  195 non-null    float64  
23  PPE                 195 non-null    float64  
dtypes: float64(22), int64(1), object(1)

```

Figure 6.1.2 Data Collection

```

parkinsons_data.isnull().sum()
Out[1]:
name                0
PDVP: Fo(Hz)        0
PDVP: Fb1(Hz)       0
PDVP: Fb2(Hz)       0
PDVP: Jitter(X)     0
PDVP: Jitter(Abs)   0
PDVP: RAP           0
PDVP: PPQ           0
Jitter: DDP         0
PDVP: Shimmer       0
PDVP: Shimmer(dB)   0
Shimmer: APQ2       0
Shimmer: APQ5       0
PDVP: APQ           0
Shimmer: DDA        0
HNR                 0
HNR                 0
status              0
RPSE                0
DFA                 0
spread1             0
spread2             0
D2                  0
PPE                 0
dtypes: int64

parkinsons_data.describe()
Out[1]:
      PDVP: Fo(Hz)  PDVP: Fb1(Hz)  PDVP: Fb2(Hz)  PDVP: Jitter(X)  PDVP: Jitter(Abs)  PDVP: RAP  PDVP: PPQ  Jitter: DDP  PDVP: Shimmer  PDVP: Shimmer(dB)  Shimmer: APQ2  Shimmer: APQ5  HNR  RPSE  DFA  spread1  spread2  D2  PPE
count  195.000000  195.000000    195.000000    195.000000    195.000000    195.000000  195.000000  195.000000  195.000000  195.000000    195.000000    195.000000  195.000000  195.000000  195.000000  195.000000  195.000000  195.000000  195.000000

```

Figure 6.1.3 Analyzing data

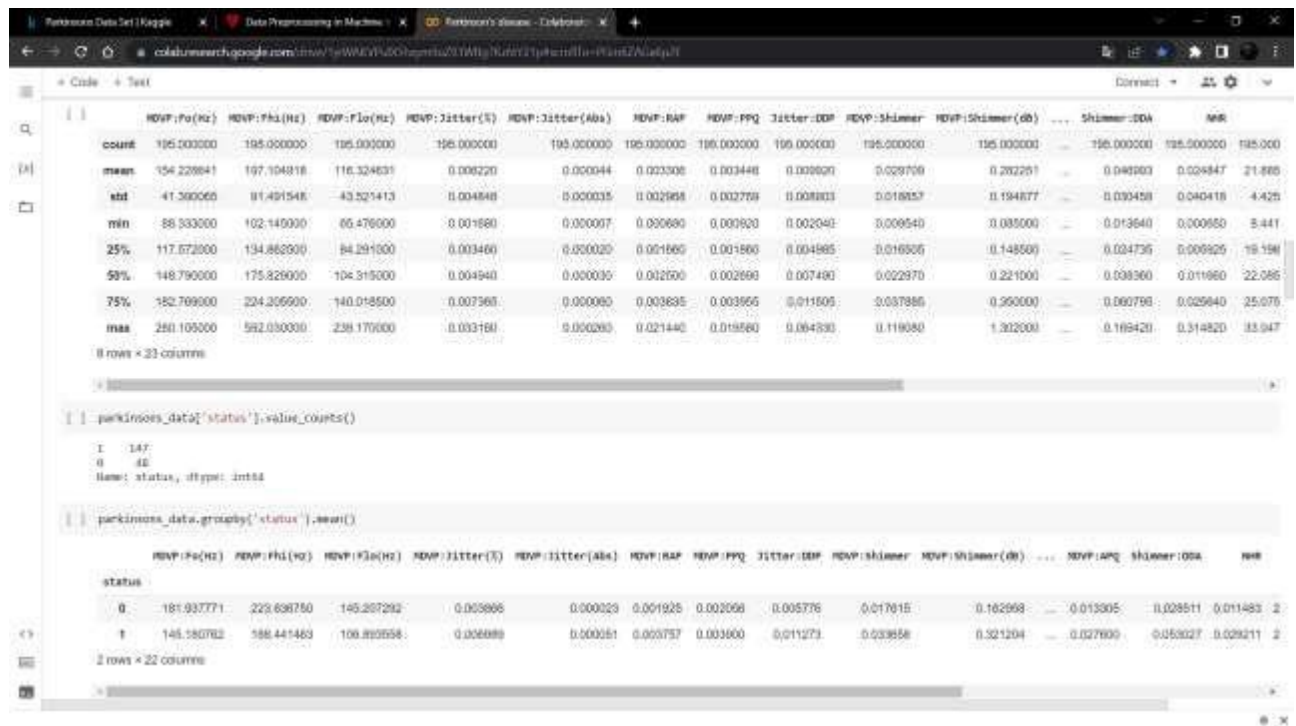


Figure 6.1.4 Describing Data

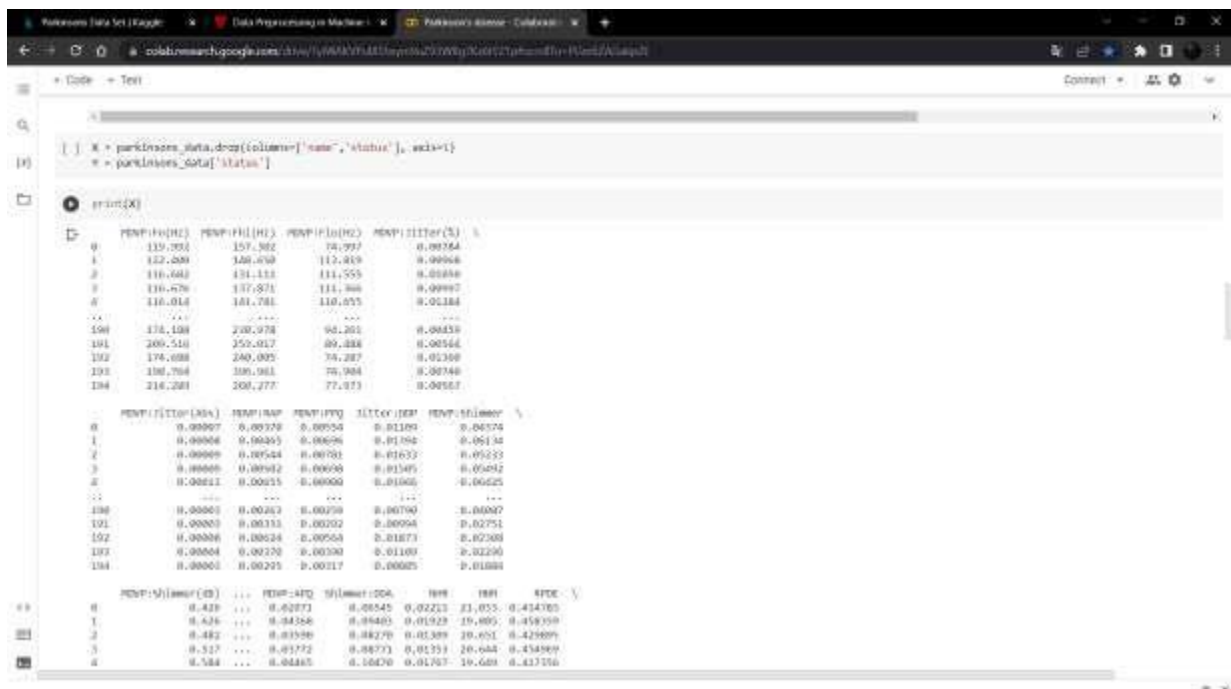


Figure 6.1.5 Data Preprocessing

```

Rankinson Data Set | Kaggle | Data Processing in Machine | Rankinson's disease - ColabNotebook
colab.research.google.com/notebooks/WWVWV50Zgmrtu2YTAh3Kutv2tp4mndfUu-PrintZNAuq0/

Code | Text
[ ] 2 0.483 ... 0.07500 0.08270 0.01300 38.653 0.429895
[ ] 3 0.537 ... 0.03772 0.00771 0.01353 20.644 0.438669
[ ] 4 0.584 ... 0.04465 0.10430 0.03767 19.649 0.417356
[ ] ...
[ ] 190 0.805 ... 0.02345 0.07000 0.02764 19.527 0.440439
[ ] 191 0.263 ... 0.03879 0.00812 0.00810 19.147 0.431674
[ ] 192 0.250 ... 0.03067 0.03904 0.10735 17.083 0.407569
[ ] 193 0.283 ... 0.03180 0.03794 0.07211 19.020 0.351221
[ ] 194 0.100 ... 0.01373 0.03050 0.03310 11.200 0.662801

DFA spread1 spread2 B3 PPE
0 0.815285 -4.811011 0.366402 2.301042 0.200014
1 0.819521 -4.075192 0.325500 2.486055 0.360074
2 0.825288 -4.643179 0.311173 2.342299 0.132614
3 0.819235 -4.117501 0.334147 2.405534 0.360975
4 0.825484 -3.907707 0.234513 2.332180 0.430335
...
190 0.857980 -4.598180 0.121052 2.857476 0.133050
191 0.885344 -4.193125 0.128361 2.784312 0.168316
192 0.850803 -4.707107 0.150453 2.670772 0.131736
193 0.843856 -4.704577 0.203454 2.136886 0.123306
194 0.668357 -5.724056 0.190067 2.555477 0.148588

[195 rows x 22 columns]

[ ] print(Y)
0 1
1 1
2 1
3 1
4 1
...
190 0
191 0
192 0
193 0
194 0
Name: status, length: 195, dtype: int8

```

```

Rankinson Data Set | Kaggle | Data Processing in Machine | Rankinson's disease - ColabNotebook
colab.research.google.com/notebooks/WWVWV50Zgmrtu2YTAh3Kutv2tp4mndfUu-PrintZNAuq0/

Code | Text
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)
[ ] print(X.shape, X_train.shape, X_test.shape)
(195, 22) (156, 22) (39, 22)

[ ] scaler = StandardScaler()
[ ] scaler.fit(X_train)
StandardScaler()

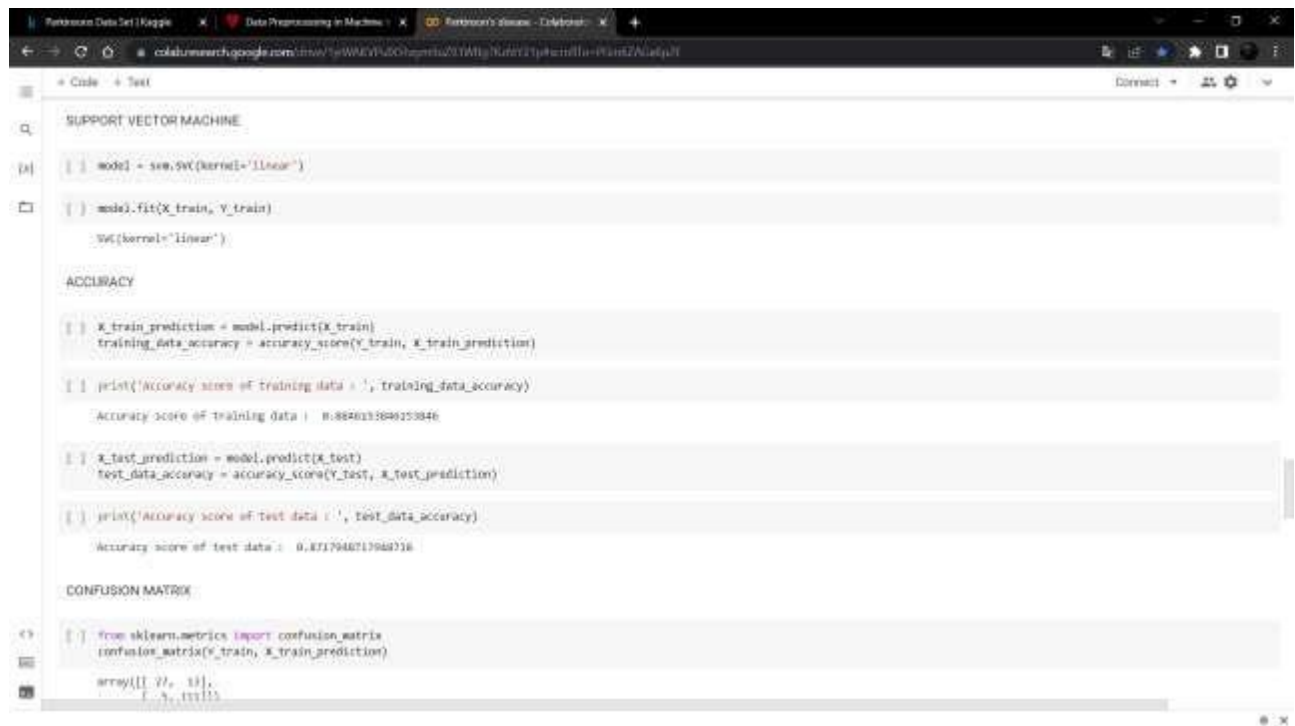
[ ] X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

[ ] print(X_train)
[[ 0.62239611 -0.02733083 -0.87945849 ... -0.07586547 -0.50160318
  0.07760404]
 [-1.05512729 -0.83337043 -0.0264770 ... 0.3081098 -0.81824073
  0.20201782]
 [ 0.02896167 -0.29531068 -1.12211307 ... -0.43937044 -0.62849605
 -0.50846406]
 ...
 [-0.0906785 -0.6637302 -0.340038 ... 1.23001037 -0.47806029
 -0.23504022]
 [-0.35577689 0.19731022 -0.70003679 ... -0.17096019 -0.47271035
  0.28381321]
 [ 1.03067008 0.19922313 -0.81044072 ... -0.710232 1.23032006
 -0.04629346]]

SUPPORT VECTOR MACHINE

```

Figure 6.1.6 Splitting data (training and testing data)



The screenshot shows a Jupyter Notebook interface with a browser window at the top displaying the URL `colab.research.google.com`. The notebook has three tabs: "Parkinson's Data Set | Kaggle", "Data Preprocessing in Machine", and "Parkinson's disease - ColabNote". The active tab is "Parkinson's disease - ColabNote". The notebook contains the following code and output:

```
SUPPORT VECTOR MACHINE

[ ] model = svm.SVC(kernel='linear')

[ ] model.fit(X_train, y_train)

SVC(kernel='linear')

ACCURACY

[ ] X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(y_train, X_train_prediction)

[ ] print('Accuracy score of training data : ', training_data_accuracy)

Accuracy score of training data : 0.88203584253846

[ ] X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(y_test, X_test_prediction)

[ ] print('Accuracy score of test data : ', test_data_accuracy)

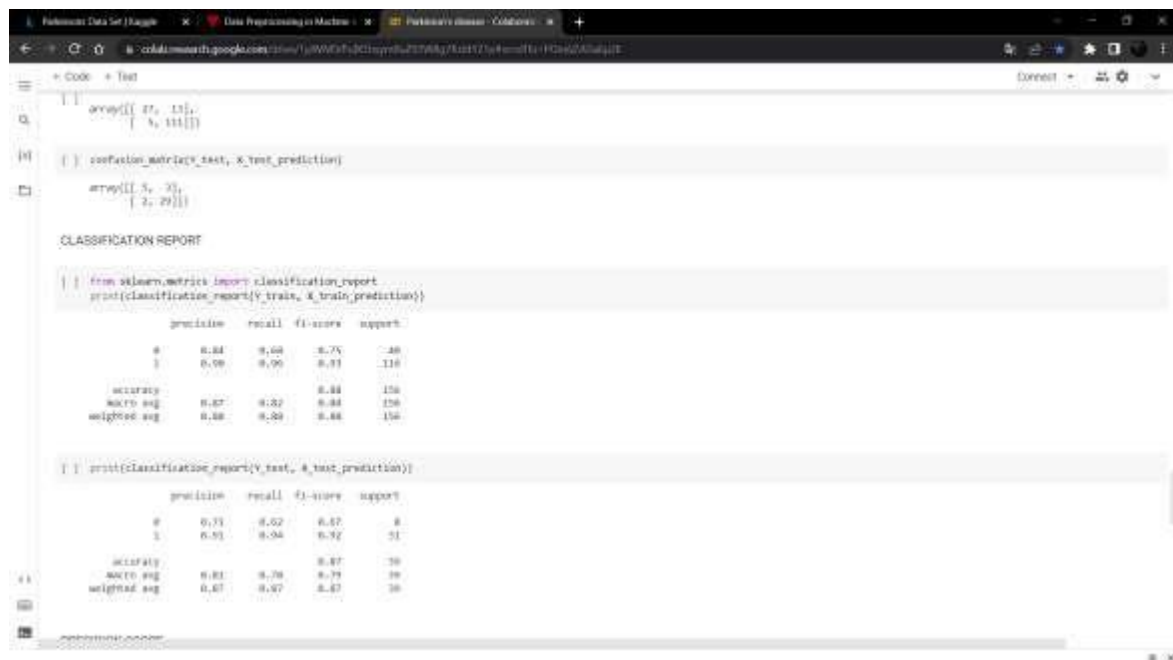
Accuracy score of test data : 0.837946717049718

CONFUSION MATRIX

[ ] from sklearn.metrics import confusion_matrix
confusion_matrix(y_train, X_train_prediction)

array([[ 77, 13],
       [ 3, 111]])
```

Figure 6.1.7 Accuracy



The screenshot shows a Jupyter Notebook interface with a browser window at the top displaying the URL `colab.research.google.com`. The notebook has three tabs: "Parkinson's Data Set | Kaggle", "Data Preprocessing in Machine", and "Parkinson's disease - ColabNote". The active tab is "Parkinson's disease - ColabNote". The notebook contains the following code and output:

```
array([[ 77, 13],
       [ 3, 111]])

[ ] confusion_matrix(y_test, X_test_prediction)

array([[ 3, 3],
       [ 2, 29]])

CLASSIFICATION REPORT

[ ] from sklearn.metrics import classification_report
print(classification_report(y_train, X_train_prediction))

      precision    recall  f1-score   support

     0       0.84      0.68      0.75        88
     1       0.59      0.90      0.73       118

 accuracy: 0.87
 macro avg: 0.82
 weighted avg: 0.88

[ ] print(classification_report(y_test, X_test_prediction))

      precision    recall  f1-score   support

     0       0.73      0.67      0.70         3
     1       0.93      0.94      0.93        31

 accuracy: 0.81
 macro avg: 0.79
 weighted avg: 0.87
```

Figure 6.1.8 Classification Report

The screenshot shows a Google Colab notebook with the following content:

```
PRECISION SCORE

[ ] precision_score(y_train, x_train_prediction)
0.9951612003229880

[ ] precision_score(y_test, x_test_prediction)
0.99025

RECALL SCORE

[ ] recall_score(y_train, x_train_prediction)
0.99990031741371

[ ] recall_score(y_test, x_test_prediction)
0.9754836700677419

F1 SCORE

[ ] f1_score(y_train, x_train_prediction)
0.995

[ ] f1_score(y_test, x_test_prediction)
0.9806389200380086
```

Below the code, there is a section labeled "PREDICTION" which is currently empty.

Figure 6.1.9 Precision score, Recall F1 score

[illegible]

Sample Output