MACHINE LEARNING LAB

EXERCISE :: 5

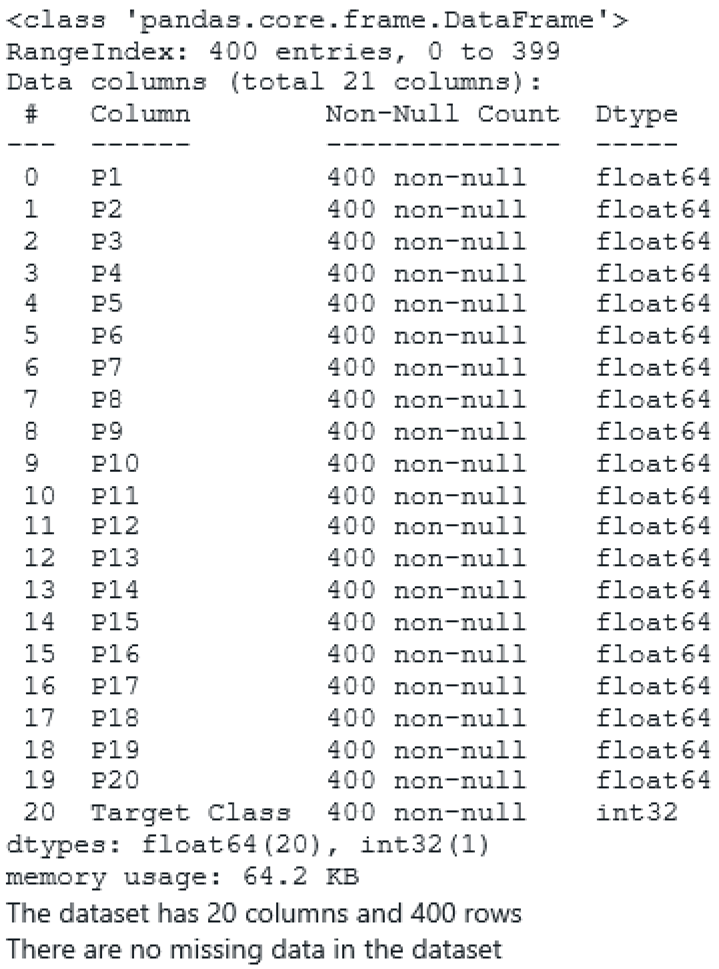
NAME:: Saptarshi Datta

REG NO.:: 19BAI1041

Single Layer Perceptron ::

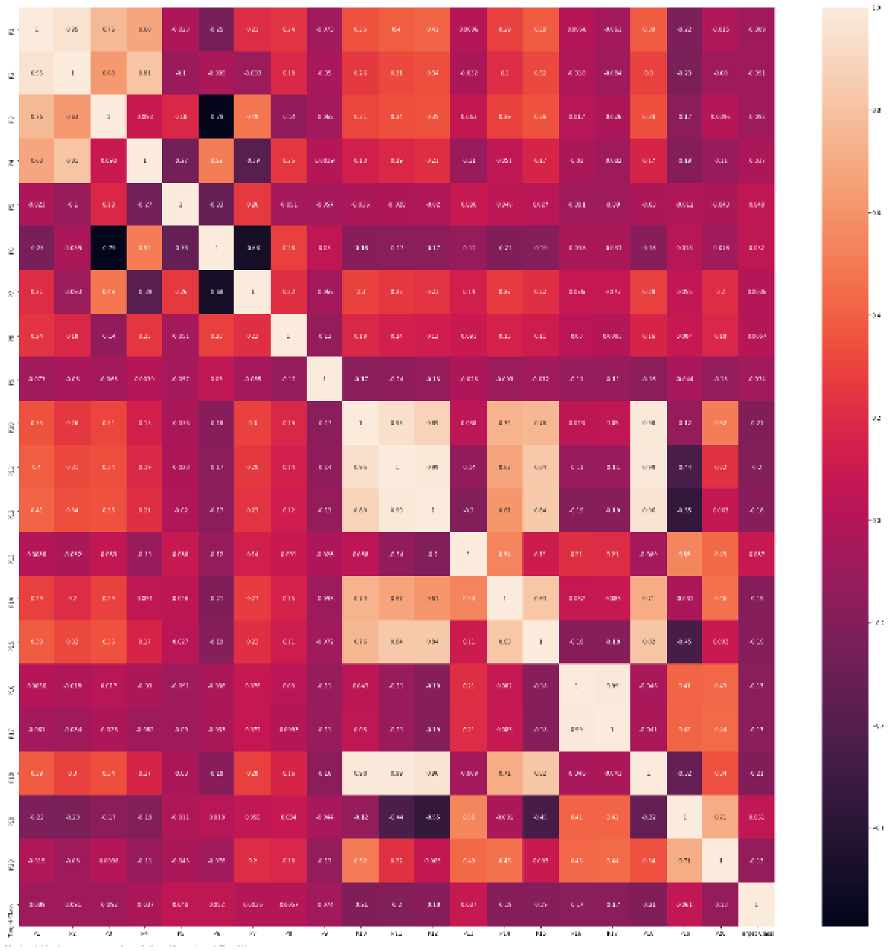
Code ::

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.preprocessing import normalize as NLZ

from sklearn.model_selection import train_test_split as tts
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix

import tensorflow
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout
from tensorflow.keras.optimizers import RMSprop

from collections import Counter
from imblearn.over_sampling import SMOTE

read_data_train=pd.read_excel('Multiclass-dataset-train.xlsx')
read_data_train.to_csv('Dataset_train.csv',index=None)
df_train=pd.DataFrame(pd.read_csv('Dataset_train.csv'))

read_data_test=pd.read_excel('Multiclass-dataset-test.xlsx')
read_data_test.to_csv('Dataset_test.csv',index=None)
df_test=pd.DataFrame(pd.read_csv('Dataset_test.csv'))
                                                                  In [3]:
df_train['Target Class']=np.where(df_train['Target Label']=='V4', 1, 0)
df_test['Target Class']=np.where(df_test['Target Label']=='V4', 1, 0)
                                                                  In [4]:
df_train=df_train.drop('Target Label',axis=1)
df_test=df_test.drop('Target Label',axis=1)
                                                                  In [5]:
df_train.info()
```



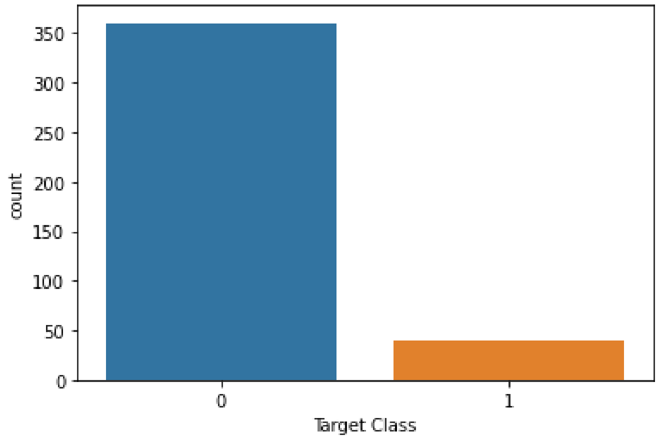
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 21 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   P1            400 non-null    float64
 1   P2            400 non-null    float64
 2   P3            400 non-null    float64
 3   P4            400 non-null    float64
 4   P5            400 non-null    float64
 5   P6            400 non-null    float64
 6   P7            400 non-null    float64
 7   P8            400 non-null    float64
 8   P9            400 non-null    float64
 9   P10           400 non-null    float64
 10  P11           400 non-null    float64
 11  P12           400 non-null    float64
 12  P13           400 non-null    float64
 13  P14           400 non-null    float64
 14  P15           400 non-null    float64
 15  P16           400 non-null    float64
 16  P17           400 non-null    float64
 17  P18           400 non-null    float64
 18  P19           400 non-null    float64
 19  P20           400 non-null    float64
 20  Target Class  400 non-null    int32  
dtypes: float64(20), int32(1)
memory usage: 64.2 KB
The dataset has 20 columns and 400 rows
There are no missing data in the dataset
```

```
plt.figure(figsize=(30,30))
sns.heatmap(df_train.corr(),annot=True)
```



```
sns.countplot(x='Target Class',data=df_train)
```

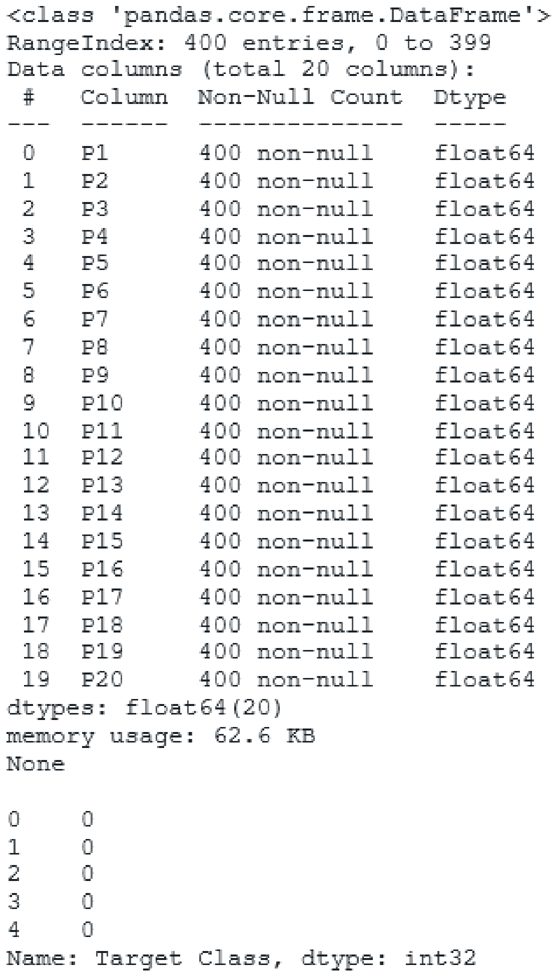
```
Out[7]:<AxesSubplot:xlabel='Target Class', ylabel='count'>
```



Only around 50 intances of our class exist in this Dataset, indicating a major imbalance

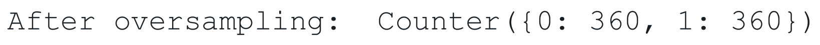
```
x_train = df_train[df_train.columns[0:20]]
y_train = df_train[df_train.columns[20]]
x_test = df_test[df_train.columns[0:20]]
y_test = df_test[df_train.columns[20]]
```

```
print(x_train.info(),'\n')
print(y_train.head())
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 20 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   P1      400 non-null    float64
 1   P2      400 non-null    float64
 2   P3      400 non-null    float64
 3   P4      400 non-null    float64
 4   P5      400 non-null    float64
 5   P6      400 non-null    float64
 6   P7      400 non-null    float64
 7   P8      400 non-null    float64
 8   P9      400 non-null    float64
 9   P10     400 non-null    float64
 10  P11     400 non-null    float64
 11  P12     400 non-null    float64
 12  P13     400 non-null    float64
 13  P14     400 non-null    float64
 14  P15     400 non-null    float64
 15  P16     400 non-null    float64
 16  P17     400 non-null    float64
 17  P18     400 non-null    float64
 18  P19     400 non-null    float64
 19  P20     400 non-null    float64
dtypes: float64(20)
memory usage: 62.6 KB
None

0    0
1    0
2    0
3    0
4    0
Name: Target Class, dtype: int32
```

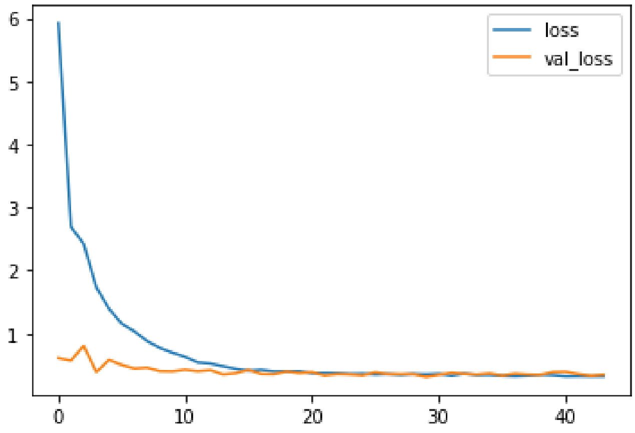
```
SMOTE = SMOTE()
x_train, y_train = SMOTE.fit_resample(x_train, y_train)
print("After oversampling: ",Counter(y_train))
```



```
After oversampling:  Counter({0: 360, 1: 360})
```

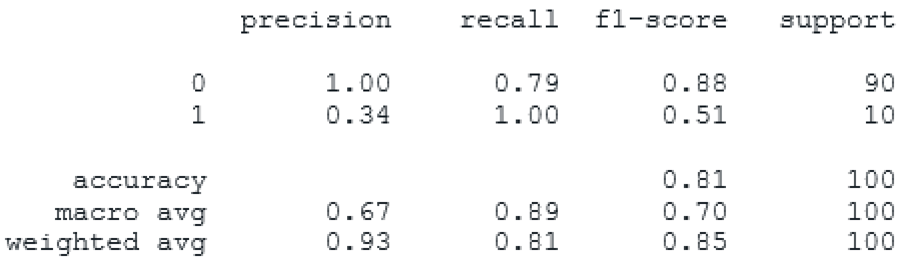
```
model_sl=Sequential()
model_sl.add(Dense(20,activation='relu'))
model_sl.add(Dropout(0.2))
model_sl.add(Dense(1, activation='sigmoid'))
model_sl.compile(optimizer='adam', loss='binary_crossentropy')
model_sl.fit(x_train,y_train,epochs=44,validation_data=(x_test,y_test),verbose=0)

loss_model_sl=pd.DataFrame(model_sl.history.history)
loss_model_sl.plot()
```

Out[12]:<AxesSubplot:>



At epochs=44, loss is minimum

prediction_model_sl=model_sl.predict(x_test)
prediction_model_sl=np.where(prediction_model_sl>=0.5,1,0)
prediction_model_sl=pd.Series(prediction_model_sl.reshape(100,))
print(classification_report(y_test,prediction_model_sl,labels=np.unique(prediction_model_sl)))

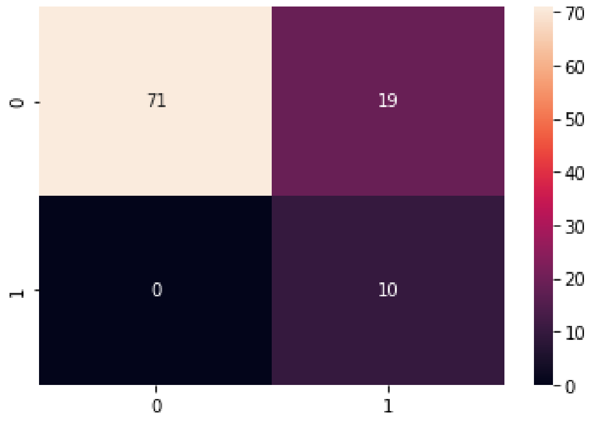


```
              precision    recall  f1-score   support

           0       1.00      0.79      0.88        90
           1       0.34      1.00      0.51        10

    accuracy                           0.81       100
   macro avg       0.67      0.89      0.70       100
weighted avg       0.93      0.81      0.85       100
```

Accuracy is 81%

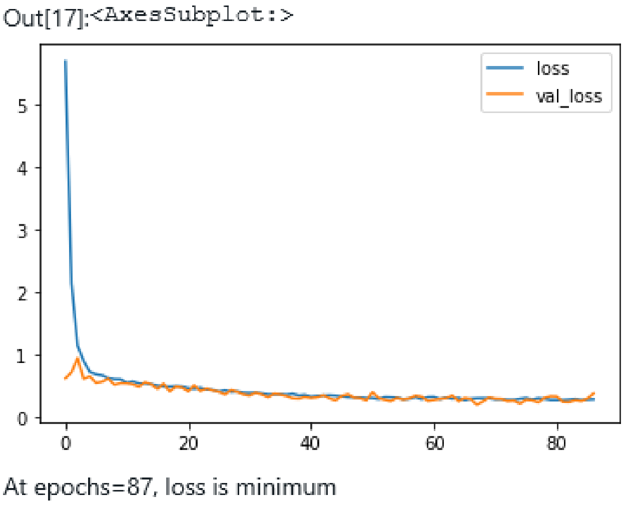
sns.heatmap(confusion_matrix(y_test,prediction_model_sl),annot=True)

Out[14]:<AxesSubplot:>



```
model_ml=Sequential()
model_ml.add(Dense(20,activation='relu'))
model_ml.add(Dropout(0.2))
model_ml.add(Dense(10,activation='relu'))
model_ml.add(Dropout(0.2))
model_ml.add(Dense(1, activation='sigmoid'))
model_ml.compile(optimizer='adam',loss='binary_crossentropy')
```

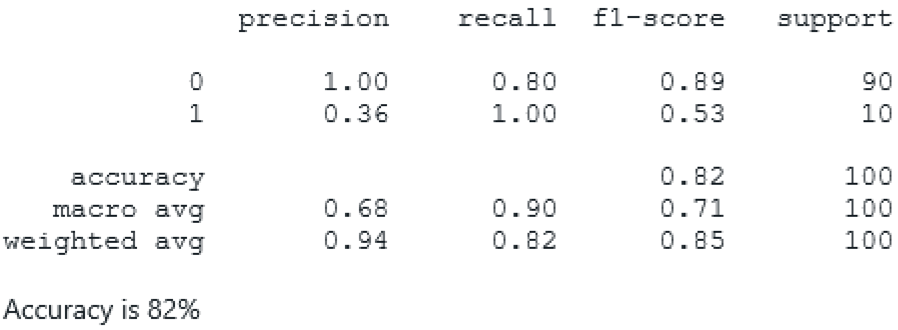
```
model_ml.fit(x_train,y_train,epochs=87,validation_data=(x_test,y_test),verbose=0)
loss_model_ml=pd.DataFrame(model_ml.history.history)
loss_model_ml.plot()
```



```
Out[17]:<AxesSubplot:>
```

At epochs=87, loss is minimum

```
prediction_model_ml=model_ml.predict(x_test)
prediction_model_ml=np.where(prediction_model_ml>=0.5,1,0)
prediction_model_ml=pd.Series(prediction_model_ml.reshape(100,))
print(classification_report(y_test,prediction_model_ml,labels=np.unique(prediction_mode
l_ml)))
```

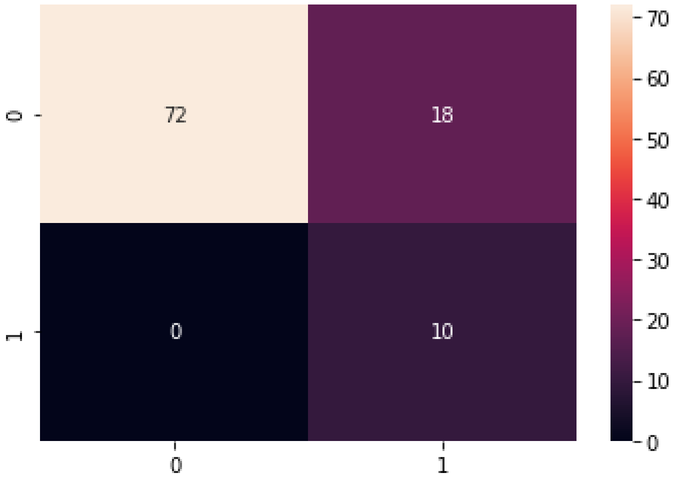


```
              precision    recall  f1-score   support

           0       1.00      0.80      0.89        90
           1       0.36      1.00      0.53        10

    accuracy                           0.82       100
   macro avg       0.68      0.90      0.71       100
weighted avg       0.94      0.82      0.85       100
```

Accuracy is 82%

```
sns.heatmap(confusion_matrix(y_test,prediction_model_ml),annot=True)
```

```
Out[19]:<AxesSubplot:>
```



0
1
72
18
0
10
0
1
70
60
50
40
30
20
10
0