MACHINE LEARNING LAB

EXERCISE :: 6

NAME:: Saptarshi Datta

REG NO.:: 19BAI1041

Multi-Layer Perceptron ::

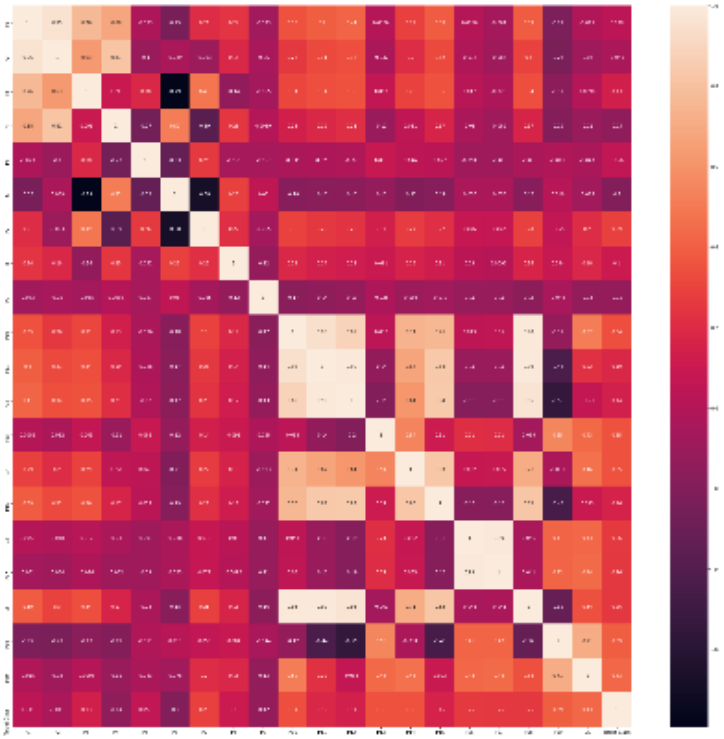
Code ::

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

from sklearn.preprocessing import normalize as NLZ
from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

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

from sklearn.neural_network import MLPClassifier

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'))

replace_class={'V1':1,'V2':2,'V3':3,'V4':4,'V5':5,'V6':6,'V7':7,'V8':8,'V9':9,'V10':10}
df_train['Target Class']=df_train['Target Label'].apply(lambda x:replace_class[x])
df_test['Target Class']=df_test['Target Label'].apply(lambda x:replace_class[x])

df_train=df_train.drop('Target Label',axis=1)
df_test=df_test.drop('Target Label',axis=1)
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    int64
dtypes: float64(20), int64(1)
memory usage: 65.8 KB
```

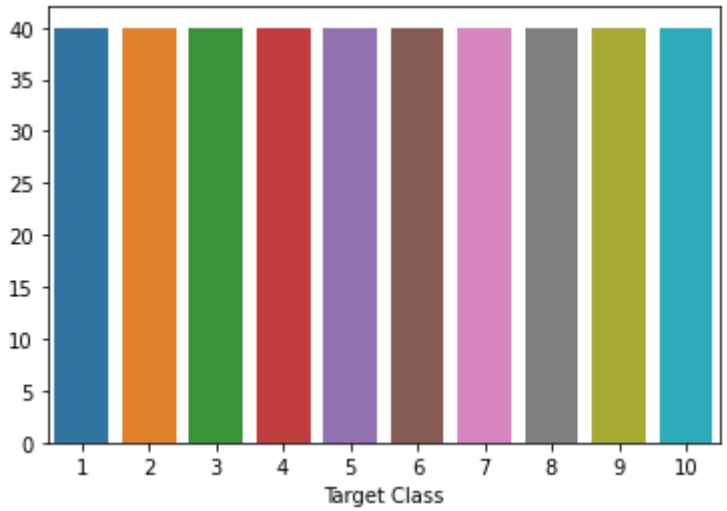
The dataset has 20 columns and 400 rows
There are no missing data in the dataset

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



Most variable show case a normal correlation with our target
But: P20, P19, P14, P13, P10, P7, P16, P17, P18 show a strong positive correlation.

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



There exists a fair distribution among the classes

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())

```
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    1
1    1
2    1
3    1
4    1
Name: Target Class, dtype: int64
```

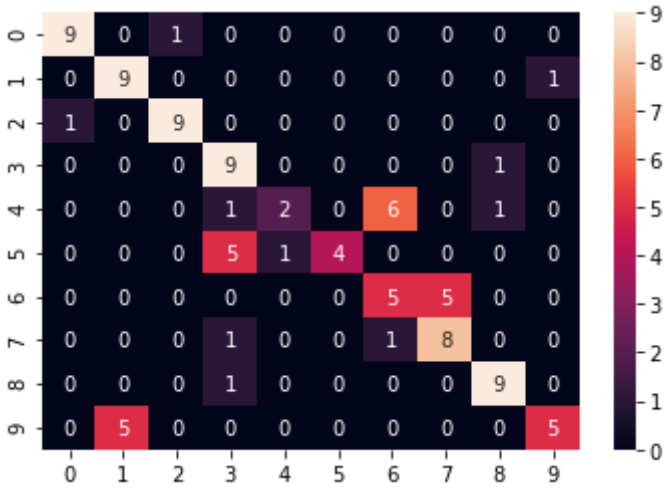
```
mlp=MLPClassifier(max_iter=500,activation='relu')
mlp.fit(x_train,y_train)
```

```
MLPClassifier(max_iter=500)
```

```
prediction=mlp.predict(x_test)
prediction
```

```
array([ 1,  1,  3,  1,  1,  1,  1,  1,  1,  1,  2,  2,  2,  2,  2,  2,  2,
       10,  2,  2,  3,  3,  3,  3,  3,  1,  3,  3,  3,  3,  9,  4,  4,  4,
        4,  4,  4,  4,  4,  4,  7,  5,  7,  7,  4,  7,  9,  7,  5,  7,  5,
        6,  6,  4,  6,  4,  4,  4,  4,  6,  8,  7,  7,  8,  8,  7,  8,  7,
        8,  7,  8,  8,  8,  7,  4,  8,  8,  8,  8,  8,  9,  9,  9,  4,  9,
        9,  9,  9,  9,  9,  2, 10, 10,  2, 10,  2, 10, 10,  2,  2],
```

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




```
print(classification_report(y_test,prediction))
```

	precision	recall	f1-score	support
1	0.90	0.90	0.90	10
2	0.64	0.90	0.75	10
3	0.90	0.90	0.90	10
4	0.53	0.90	0.67	10
5	0.67	0.20	0.31	10
6	1.00	0.40	0.57	10
7	0.42	0.50	0.45	10
8	0.62	0.80	0.70	10
9	0.82	0.90	0.86	10
10	0.83	0.50	0.62	10
accuracy			0.69	100
macro avg	0.73	0.69	0.67	100
weighted avg	0.73	0.69	0.67	100

Accuracy: 69%