Experiment 3: Classifier

Aim:
To use multi-layered perceptron for performing a classification task on a dataset

Problem Statement:
Choose a classification dataset of your choice from any of the following Repository Links, download it:

1. Kaggle: https://www.kaggle.com/
2. UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/index.php

Perform Linear Regression on the chosen dataset.

Your notebook should contain:
1. Basic EDA
2. Creation of MultiLayered Perceptron Model using sklearn and subsequent training on Training set
3. Test the model on test set by printing on the classification metrics

[***Hint***: Follow the steps in Titanic notebook uploaded on moodle under Expt 3 reference material]

Tool/Language:
Programming language: Python
Libraries: numpy, pandas, sklearn, matplotlib, seaborn

Code with visualisation graphs:

1) **Dataset Chosen:** Red Wine Quality
2) **Dataset Description:** This dataset is related to red variants of the Portuguese "Vinho Verde" wine.
   a) Input variables (based on physicochemical tests):
      - fixed acidity (most acids involved with wine or fixed or non-volatile)
      - volatile acidity (the amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste)
      - citric acid (found in small quantities, citric acid can add 'freshness' and flavour to wines)
      - residual sugar (the amount of sugar remaining after fermentation stops, it's rare to find wines with less than 1 gram/litre)
      - chlorides (the amount of salt in the wine)
      - free Sulphur dioxide (the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulphite ion)
      - total Sulphur dioxide (amount of free and bound forms of S02)
      - density (the density of water is close to that of water depending on the percent alcohol and sugar content)
      - pH (describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic))

- sulphates (a wine additive which can contribute to Sulphur dioxide gas (S02) levels, which acts as an antimicrobial)
- alcohol

b) Output variable (based on sensory data):

- quality (score between 0 and 10)

3) Code:

```
from google.colab import files
uploaded = files.upload()

# Basic Pre-processing
import numpy as np
import pandas as pd
import io

# For Model Selection and Training
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_sco
re

df = pd.read_csv(io.BytesIO(uploaded['winequality-red.csv']))
df.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	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

```
df.shape
```

```
(1599, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

```
# Target Class
df['quality'].unique()
```

```
array([5, 6, 7, 4, 8, 3])
```

```
# Seperating target classes and feature classes
target_classes = df['quality']
target_classes
```

```
0       5
1       5
2       5
3       6
4       5
       ..
1594    5
1595    6
1596    6
1597    5
1598    6
Name: quality, Length: 1599, dtype: int64
```

```
features = df.drop(['quality'], axis=1)
features
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

1599 rows × 11 columns

```
# Split into training and testing

X_train, X_test, Y_train, Y_test = train_test_split(features, target_classes, test_size=0.4, random_state=101)
X_train.shape
```

```
(959, 11)
```

```
# Neural Network Testing

mlp = MLPClassifier(hidden_layer_sizes=(7,7,7), activation='relu', solver='adam',
 max_iter=15000)
mlp
```

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(7, 7, 7), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=15000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

```
mlp.fit(X_train, Y_train)

predict_test = mlp.predict(X_test)
print('Accuracy Score - ')
accuracy_score(Y_test, predict_test)
```

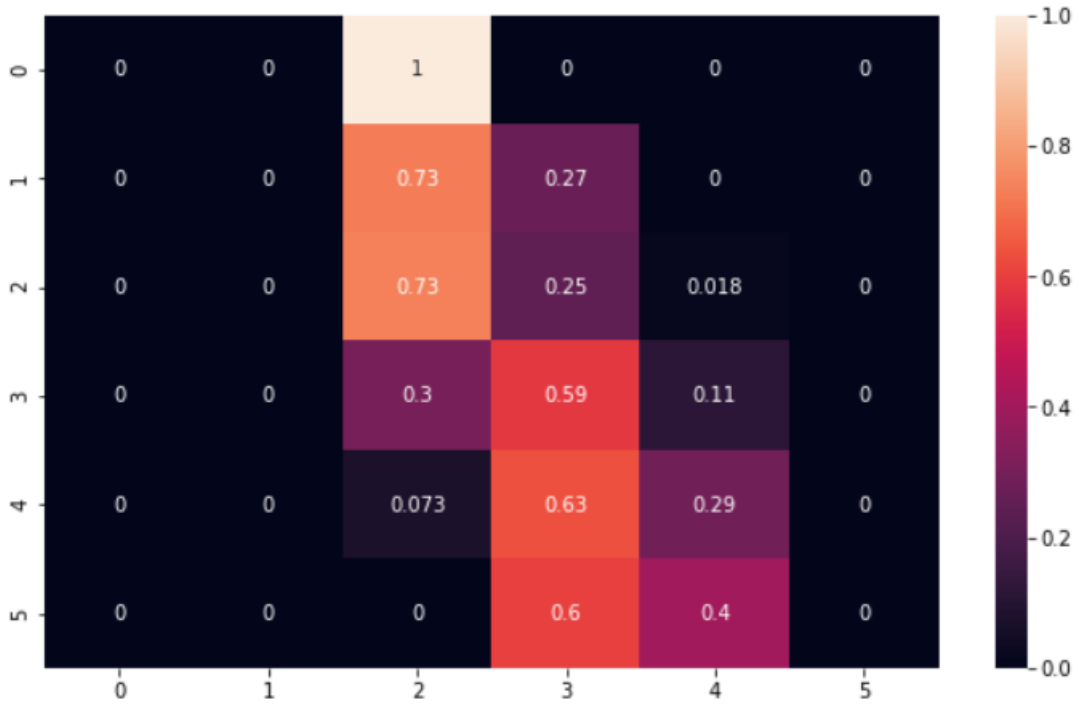
```
Accuracy Score -
0.578125
```

```
Observation: The model is 57.8% accurate.

cm = confusion_matrix(Y_test, predict_test, normalize='true')
cm
```

```
array([[0.        , 0.        , 1.        , 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.72727273, 0.27272727, 0.        ,
        0.        ],
       [0.        , 0.        , 0.73454545, 0.24727273, 0.01818182,
        0.        ],
       [0.        , 0.        , 0.30081301, 0.58536585, 0.11382114,
        0.        ],
       [0.        , 0.        , 0.07317073, 0.63414634, 0.29268293,
        0.        ],
       [0.        , 0.        , 0.        , 0.6       , 0.4       ,
        0.        ]])
```

```
# For Visualizations
import matplotlib.pyplot as plt
import seaborn as sns

fig = plt.figure(figsize=(10,6))
sns.heatmap(cm,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f05e07a4110>
```




```
Observation: The above heatmap shows the normalized confusion matrix

classification_report(Y_test, predict_test)
```

	precision	recall	f1-score	support
3	0.00	0.00	0.00	5
4	0.00	0.00	0.00	22
5	0.67	0.73	0.70	275
6	0.52	0.59	0.55	246
7	0.39	0.29	0.34	82
8	0.00	0.00	0.00	10
accuracy			0.58	640
macro avg	0.26	0.27	0.26	640
weighted avg	0.54	0.58	0.56	640

Conclusion:

The model works pretty good for quality index of 5,6 and 7. It doesn't work good enough for quality index 3,4 and 8. This is because the dataset is very small and does not hold good number of values for quality index 3,4 and 8. This reduces the overall accuracy of the model.