Krish Sukhani

TE IT

Batch : D

2018140059

ML - Exp 3 - Wine Quality Classification Daataset

```
#importing necessary libraries
import numpy as np
import pandas as pd
import warnings
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount("/content/gdrive")
```

```
    Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
wine = pd.read_csv('/content/gdrive/My Drive/datasets/wine.csv',encoding= 'unicode_escape')
```

```
wine.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	bad
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	bad
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	bad
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	good
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	bad

```
wine.isnull().sum()
```

```
    fixed acidity           0
    volatile acidity        0
    citric acid             0
    residual sugar          0
    chlorides               0
    free sulfur dioxide     0
    total sulfur dioxide    0
    density                 0
    pH                      0
    sulphates               0
    alcohol                 0
    quality                 0
    dtype: int64
```

```
plt.figure(figsize=(40,25))
plt.subplots_adjust(left=0, bottom=0.5, right=0.9, top=0.9, wspace=0.5, hspace=0.8)
plt.subplot(141)
plt.title('Percentage of players from each country',fontsize = 20)
wine['quality'].value_counts().plot.pie(autopct="%1.1f%%")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f342a1b5710>
```

```
wine['quality'].replace({'bad': 0 , 'good': 1}, inplace=True)
```

```
wine.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	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	0
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	0
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	1
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	0

```
Y = wine['quality']
```

```
Y
```

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

```
X = wine.drop(['quality'],axis = 1)
```

```
X
```

	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

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
training_set_scaled = sc.fit_transform(X)
```

```
num_val = 0.2
X_train, X_test, y_train, y_test = train_test_split(training_set_scaled, Y, test_size=num_val, random_state=23)
```

```
mlp = MLPClassifier(hidden_layer_sizes=(15,15,15), activation='relu', solver='adam', max_iter=1000)
mlp.fit(X_train,y_train)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
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=(15, 15, 15), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=1000,
              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)
```

```
# y_pred = mlp.predict(X_test)
# print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
predict_train = mlp.predict(X_train)
predict_test = mlp.predict(X_test)
```

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
cf_matrix = confusion_matrix(y_train,predict_train)
```

```
print(confusion_matrix(y_train,predict_train))
print(classification_report(y_train,predict_train))
```

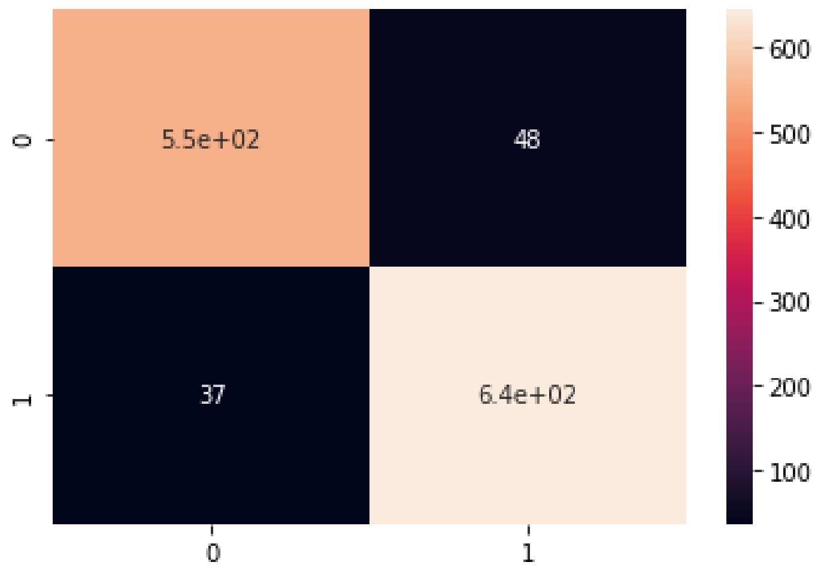
```
[[550  48]
 [ 37 644]]
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       598
           1       0.93      0.95      0.94       681

    accuracy                           0.93      1279
   macro avg       0.93      0.93      0.93      1279
weighted avg       0.93      0.93      0.93      1279
```

```
mlp.predict([[7.4,  0.700,  0.00, 1.9,  0.076,  11.0, 34.0, 0.99780,  3.51, 0.56, 9.4]])
```

```
array([0])
```

```
sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3429696908>
```



5.5e+02
48
37
6.4e+02
0
1
0
1
600
500
400
300
200
100