Scikit learn

In [3]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import svm
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split
%matplotlib inline
```

In [69]:
```python
df=pd.read_csv(r"C:\Users\user\Downloads\winequality-red.csv")
```

In [70]:
```python
df.head(10)
```

Out[70]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5

In [71]:
```
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
```

In [72]:
```
df.isnull().sum()
```

Out[72]:
```
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
```

In [73]:
```
df.head()
```

Out[73]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

In [86]:
```
#preprocessing data
bins=(2,6.5,8)
group_names=["good","bad"]
df["quality"]=pd.cut(df["quality"],bins=bins,labels=group_names)
df["quality"].unique()
```

Out[86]:
```
[NaN, 'good']
Categories (2, object): ['good' < 'bad']
```

In [87]:
```
label_quality=LabelEncoder()
```

In [88]:
```
df["quality"]=label_quality.fit_transform(df["quality"])
```

In [89]:
```
df.head(10)
```

Out[89]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5

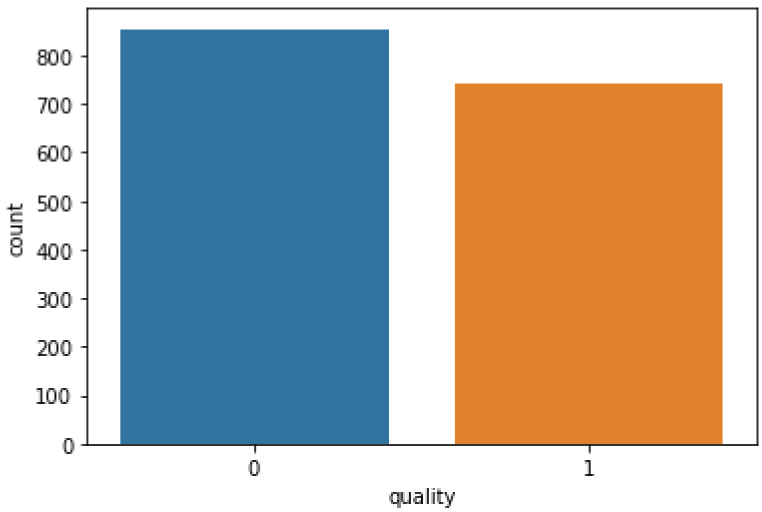
In [90]:
```
df["quality"].value_counts()
```

Out[90]:
```
0    855
1    744
Name: quality, dtype: int64
```

In [91]:
```
sns.countplot(df["quality"])
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments withou
t an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[91]: <AxesSubplot:xlabel='quality', ylabel='count'>



In [97]:
```
#seperating dataset as response variable and feature variable
x=df.drop("quality",axis=1)
y=df["quality"]
```

In [102]:
```
#train and testing the spliting of data
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random
```

In [103]:
```
#applying standard scaler to get optimized results
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In [106]:
```
x_train[:10]
```

Out[106]:
```
array([[ 0.21833164,  0.88971201,  0.19209222,  0.30972563, -0.04964208,
         0.69100692,  1.04293362,  1.84669643,  1.09349989,  0.45822284,
         1.12317723],
       [-1.29016623, -1.78878251,  0.65275338, -0.80507963, -0.45521361,
         2.38847304,  3.59387025, -3.00449133, -0.40043872, -0.40119696,
         1.40827174],
       [ 1.49475291, -0.78434707,  1.01104539, -0.52637831,  0.59927236,
        -0.95796016, -0.99174203,  0.76865471, -0.07566946,  0.51551749,
        -0.58738978],
       [ 0.27635078,  0.86181102, -0.06383064, -0.66572897, -0.00908493,
         0.01202048, -0.71842739,  0.08948842,  0.05423824, -1.08873281,
        -0.96751578],
       [ 0.04427419,  2.81487994, -0.62686095,  2.39998549, -0.31326357,
        -0.47296984,  0.2229897 ,  1.1998714 ,  0.37900751, -0.9741435 ,
        -0.49235828],
       [-0.07176411, -0.78434707,  1.11341454, -0.17800167,  0.21397941,
         3.01896045,  2.62208486,  0.60694845,  0.44396136,  1.89058918,
        -0.58738978],
       [-1.17412793,  0.10848444, -0.62686095, -0.52637831, -0.23214927,
         0.98200112, -0.35400787, -1.95879086,  0.05423824,  0.91658007,
         1.12317723],
       [-0.1878024 , -0.17052541,  0.60156881,  0.03102432, -0.13075639,
        -0.37597178, -0.01995665,  0.93036097,  0.76873063, -0.229313  ,
         0.26789373],
       [-0.07176411,  0.61070216, -0.01264607, -0.38702766,  0.13286511,
        -1.05495822,  0.92146044,  0.37516948, -1.17988496, -0.229313  ,
        -1.25261029],
       [ 1.8428678 , -1.95618842,  1.21578369,  1.00647892,  0.31537229,
        -1.15195628, -0.71842739,  1.52328391, -0.20557717,  1.77599987,
        -0.30229528]])
```

Random forest classifier

In [109]:
```
rfc=RandomForestClassifier(n_estimators=200)
rfc.fit(x_train,y_train)
pred_rfc = rfc.predict(x_test)
```

In [112]:
```
pred_rfc[:20]
```

Out[112]:
```
array([1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1])
```

In [116]:
```
#model performing
print(classification_report(y_test,pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.81      0.81      0.81       179
           1       0.76      0.76      0.76       141

    accuracy                           0.79       320
   macro avg       0.78      0.78      0.78       320
weighted avg       0.79      0.79      0.79       320
```

Svm classifier

In [117]:
```
clf=svm.SVC()
clf.fit(x_train,y_train)
pred_clf=clf.predict(x_test)
```

In [118]:
```
print(classification_report(y_test,pred_clf))
print(confusion_matrix(y_test,pred_clf))
```

```
              precision    recall  f1-score   support

           0       0.81      0.77      0.79       179
           1       0.73      0.77      0.75       141

    accuracy                           0.77       320
   macro avg       0.77      0.77      0.77       320
weighted avg       0.77      0.77      0.77       320

[[138  41]
 [ 32 109]]
```

Neural network

In [121]:
```
mlpc=MLPClassifier(hidden_layer_sizes=(11,11,11),max_iter=500)
mlpc.fit(x_train,y_train)
pred_mlpc=mlpc.predict(x_test)
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_pe
rceptron.py:692: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(500) reached and the optimization hasn't converged yet.
  warnings.warn(
```

In [122]:

```
print(classification_report(y_test,pred_mlpc))
print(confusion_matrix(y_test,pred_mlpc))
```

```
              precision    recall  f1-score   support

           0       0.78      0.78      0.78       179
           1       0.72      0.72      0.72       141

    accuracy                           0.75       320
   macro avg       0.75      0.75      0.75       320
weighted avg       0.75      0.75      0.75       320

[[139  40]
 [ 40 101]]
```

In [123]:

```
from sklearn.metrics import accuracy_score
cm=accuracy_score(y_test,pred_rfc)
cm
```

Out[123]: 0.7875

In [124]:

```
df.head()
```

Out[124]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4