Untitled1

June 9, 2021

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC #support vector classifier
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
```

[2]:
```
# loading dataset
wine = pd.read_csv('winequality-red.csv', sep=',')
wine.head()
```

[2]:

	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

[3]:
```
wine.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
```

[4]:
```
wine.isnull().sum()
```

[4]:
```
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
```

[5]:
```
# Preprocessing Data
bins = (1, 6.5, 10)  # 2 variants, mean, max
group_names = ['bad', 'good']
wine['quality'] = pd.cut(wine['quality'], bins= bins, labels= group_names)
wine['quality'].unique()
```

[5]:
```
['bad', 'good']
Categories (2, object): ['bad' < 'good']
```

[6]:
```
label_quality = LabelEncoder()
```

[7]:
```
wine['quality'] = label_quality.fit_transform(wine['quality'])
```

[8]:
```
wine.head(10)
```

[8]:

	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	0
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	0
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	0
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	0
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	1
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	1
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	0

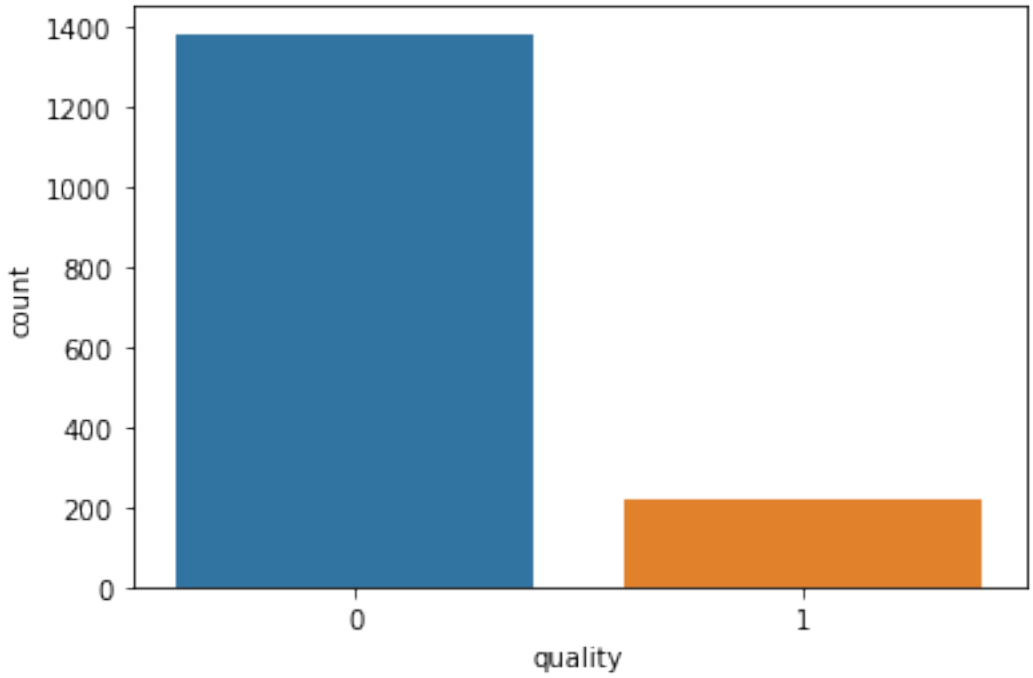
[9]:
```
wine['quality'].value_counts()
```

[9]:
```
0    1382
1     217
Name: quality, dtype: int64
```

[10]:
```
#fig, ax = plt.subplots()
#ax.hist(wine['quality'])
#ax.set_xticks([0, 1])
sns.countplot(wine['quality'])
```

```
C:\Users\Harald\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[10]:
```
<AxesSubplot:xlabel='quality', ylabel='count'>
```



[11]:
```
# Now separate dataset as response variable and feature variables
X = wine.drop('quality', axis=1)
y = wine['quality']
```

[18]:
```
# Train and Test splitting of data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[19]:
```
# Appliing Standard scaling to get optimized result

sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

1 Random Forest Classifier

[20]:
```
rfc = RandomForestClassifier(n_estimators=200)  # n_estimator = number of trees
↪in model
rfc.fit(X_train, y_train)
pred_rfc = rfc.predict(X_test)
```

[21]:
```
print (classification_report(y_test, pred_rfc))
print (confusion_matrix(y_test, pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.95       273
           1       0.77      0.51      0.62        47

    accuracy                           0.91       320
   macro avg       0.85      0.74      0.78       320
weighted avg       0.90      0.91      0.90       320

[[266   7]
 [ 23  24]]
```

2 SVM Classifier

[22]:
```
clf = svm.SVC()
clf.fit(X_train, y_train)
pred_clf = clf.predict(X_test)
```

[23]:
```
print (classification_report(y_test, pred_clf))
print (confusion_matrix(y_test, pred_clf))
```

```
              precision    recall  f1-score   support

           0       0.88      0.98      0.93       273
           1       0.71      0.26      0.37        47

    accuracy                           0.88       320
   macro avg       0.80      0.62      0.65       320
weighted avg       0.86      0.88      0.85       320

[[268   5]
 [ 35  12]]
```

3 Neural Wokrs

[24]:
```
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\Harald\anaconda3\lib\site-
packages\sklearn\neural_network\_multilayer_perceptron.py:582:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

[25]:
```
print (classification_report(y_test, pred_clf))
print (confusion_matrix(y_test, pred_clf))
```

```
              precision    recall  f1-score   support

           0       0.88      0.98      0.93       273
           1       0.71      0.26      0.37        47

    accuracy                           0.88       320
   macro avg       0.80      0.62      0.65       320
weighted avg       0.86      0.88      0.85       320

[[268   5]
 [ 35  12]]
```

[26]:
```
from sklearn.metrics import accuracy_score
cm = accuracy_score(y_test, pred_rfc)
cm
```

[26]:
```
0.90625
```

[27]:
```
wine.head(10)
```

[27]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
0	7.4	0.70	0.00	1.9	0.076	
1	7.8	0.88	0.00	2.6	0.098	
2	7.8	0.76	0.04	2.3	0.092	
3	11.2	0.28	0.56	1.9	0.075	
4	7.4	0.70	0.00	1.9	0.076	
5	7.4	0.66	0.00	1.8	0.075	
6	7.9	0.60	0.06	1.6	0.069	
7	7.3	0.65	0.00	1.2	0.065	
8	7.8	0.58	0.02	2.0	0.073	
9	7.5	0.50	0.36	6.1	0.071	

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	\
0	11.0	34.0	0.9978	3.51	0.56	

```
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56
5                 13.0                  40.0   0.9978  3.51       0.56
6                 15.0                  59.0   0.9964  3.30       0.46
7                 15.0                  21.0   0.9946  3.39       0.47
8                  9.0                  18.0   0.9968  3.36       0.57
9                 17.0                 102.0   0.9978  3.35       0.80

   alcohol  quality
0      9.4        0
1      9.8        0
2      9.8        0
3      9.8        0
4      9.4        0
5      9.4        0
6      9.4        0
7     10.0        1
8      9.5        1
9     10.5        0
```

[30]:
```
X_new = [[7.3, 0.65, 0.00, 1.2, 0.065,15.0,21.0,0.9946,3.39,0.47, 10.0]]
X_new = sc.transform(X_new)
ynew = rfc.predict(X_new)
ynew
```

[30]:
```
array([1])
```

[ ]:
```

```