Wine Quality Prediction using Support Vector Machine

Import Library

In [72]:

```
import pandas as pd
```

In [73]:

```
import numpy as np
```

Import CSV as DataFrame

In [74]:

```
df = pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/WhiteWineQuality.csv',
```

Get the First rows of Dataframe

In [75]:

```
df.head()
```

Out[75]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoh
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9

Get Information of DataFrame

In [76]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         4898 non-null   float64
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
```

Get Summary Statistics

In [77]:

```
df.describe()
```

Out[77]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total su dio
count	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.00(
mean	6.854788	0.278241	0.334192	6.391415	0.045772	35.308085	138.36(
std	0.843868	0.100795	0.121020	5.072058	0.021848	17.007137	42.49{
min	3.800000	0.080000	0.000000	0.600000	0.009000	2.000000	9.00(
25%	6.300000	0.210000	0.270000	1.700000	0.036000	23.000000	108.00(
50%	6.800000	0.260000	0.320000	5.200000	0.043000	34.000000	134.00(
75%	7.300000	0.320000	0.390000	9.900000	0.050000	46.000000	167.00(
max	14.200000	1.100000	1.660000	65.800000	0.346000	289.000000	440.00(

Get Column Names

In [78]:

```
df.columns
```

Out[78]:

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'densit
y',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

Get Shape of DataFrame

In [79]:

```
df.shape
```

Out[79]:

```
(4898, 12)
```

Get Unique Values (Class or Label) in y Variable

In [80]:

```
df['quality'].value_counts()
```

Out[80]:

```
6    2198
5    1457
7     880
8     175
4     163
3      20
9       5
Name: quality, dtype: int64
```

In [81]:

```
df.groupby('quality').mean()
```

Out[81]:

quality	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
3	7.600000	0.333250	0.336000	6.392500	0.054300	53.325000	170.600000	0.994884	3.1
4	7.129448	0.381227	0.304233	4.628221	0.050098	23.358896	125.279141	0.994277	3.1
5	6.933974	0.302011	0.337653	7.334969	0.051546	36.432052	150.904598	0.995263	3.1
6	6.837671	0.260564	0.338025	6.441606	0.045217	35.650591	137.047316	0.993961	3.1
7	6.734716	0.262767	0.325625	5.186477	0.038191	34.125568	125.114773	0.992452	3.2
8	6.657143	0.277400	0.326514	5.671429	0.038314	36.720000	126.165714	0.992236	3.2
9	7.420000	0.298000	0.386000	4.120000	0.027400	33.400000	116.000000	0.991460	3.3

Define y and X

In [82]:

```
y = df['quality']
```

In [83]:

```
y.shape
```

Out[83]:

```
(4898,)
```

In [84]:

```
y
```

Out[84]:

```
0       6
1       6
2       6
3       6
4       6
       ..
4893    6
4894    5
4895    6
4896    7
4897    6
Name: quality, Length: 4898, dtype: int64
```

In [85]:

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

In [86]:

```
x.shape
```

Out[86]:

```
(4898, 11)
```

In [87]:

```
x
```

Out[87]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	al
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.00100	3.00	0.45	
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.99400	3.30	0.49	
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.99510	3.26	0.44	
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	
...	...	...	...	...	...	...	...	...	...	...	
4893	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	
4894	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	
4895	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	
4896	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	
4897	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	

4898 rows × 11 columns

Get X Variables Standardized

In [88]:

```
from sklearn.preprocessing import StandardScaler
```

In [89]:

```
ss = StandardScaler()
```

In [90]:

```
x = ss.fit_transform(x)
```

In [91]:

```
x
```

Out[91]:

```
array([[ 1.72096961e-01, -8.17699008e-02,  2.13280202e-01, ...,
        -1.24692128e+00, -3.49184257e-01, -1.39315246e+00],
       [-6.57501128e-01,  2.15895632e-01,  4.80011213e-02, ...,
         7.40028640e-01,  1.34184656e-03, -8.24275678e-01],
       [ 1.47575110e+00,  1.74519434e-02,  5.43838363e-01, ...,
         4.75101984e-01, -4.36815783e-01, -3.36667007e-01],
       ...,
       [-4.20473102e-01, -3.79435433e-01, -1.19159198e+00, ...,
        -1.31315295e+00, -2.61552731e-01, -9.05543789e-01],
       [-1.60561323e+00,  1.16673788e-01, -2.82557040e-01, ...,
         1.00495530e+00, -9.62604939e-01,  1.85757201e+00],
       [-1.01304317e+00, -6.77100966e-01,  3.78559282e-01, ...,
         4.75101984e-01, -1.48839409e+00,  1.04489089e+00]])
```

Get Train Test Split

In [92]:

```
from sklearn.model_selection import train_test_split
```

In [93]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3, stratify=y, rando
```

In [94]:

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[94]:

```
((3428, 11), (1470, 11), (3428,), (1470,))
```

Get Model Train

In [95]:

```
from sklearn.svm import SVC
```

In [96]:

```
svc = SVC()
```

In [97]:

```
svc.fit(x_train, y_train)
```

Out[97]:

```
SVC()
```

Model Prediction

In [98]:

```
y_pred = svc.predict(x_test)
```

In [99]:

```
y_pred.shape
```

Out[99]:

```
(1470,)
```

In [100]:

```
y_pred
```

Out[100]:

```
array([5, 7, 5, ..., 5, 5, 5], dtype=int64)
```

Model Evaluation

In [101]:

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

In [102]:

```
print(confusion_matrix(y_test, y_pred))
```

```
[[  0   0   1   5   0   0   0]
 [  0   2  25  22   0   0   0]
 [  0   3 273 160   1   0   0]
 [  0   0 122 515  23   0   0]
 [  0   0   6 191  67   0   0]
 [  0   0   0  39  14   0   0]
 [  0   0   0   0   1   0   0]]
```

In [104]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         6
           4       0.40      0.04      0.07        49
           5       0.64      0.62      0.63       437
           6       0.55      0.78      0.65       660
           7       0.63      0.25      0.36       264
           8       0.00      0.00      0.00        53
           9       0.00      0.00      0.00         1

    accuracy                           0.58      1470
   macro avg       0.32      0.24      0.25      1470
weighted avg       0.57      0.58      0.55      1470
```

```
C:\Users\91814\New folder\lib\site-packages\sklearn\metrics\_classification.
py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` par
ameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\91814\New folder\lib\site-packages\sklearn\metrics\_classification.
py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` par
ameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\91814\New folder\lib\site-packages\sklearn\metrics\_classification.
py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` par
ameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Model Re-run with two class created for wine quality

In [105]:

```
y = df['quality'].apply(lambda y_value: 1 if y_value>=6 else 0)
```

In [106]:

```
y.value_counts()
```

Out[106]:

```
1    3258
0    1640
Name: quality, dtype: int64
```

Train test split

In [107]:

```
from sklearn.model_selection import train_test_split
```

In [108]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y ,test_size = 0.3, stratify= y, rand
```

In [109]:

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[109]:

```
((3428, 11), (1470, 11), (3428,), (1470,))
```

Model Train

In [110]:

```
from sklearn.svm import SVC
```

In [111]:

```
svc = SVC()
```

In [112]:

```
svc.fit(x_train, y_train)
```

Out[112]:

```
SVC()
```

Model Prediction

In [113]:

```
y_pred = svc.predict(x_test)
```

In [114]:

```
y_pred.shape
```

Out[114]:

```
(1470,)
```

In [115]:

```
y_pred
```

Out[115]:

```
array([0, 1, 1, ..., 1, 1, 1], dtype=int64)
```

Model Evaluation

In [116]:

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

In [117]:

```
print(confusion_matrix(y_test, y_pred))
```

```
[[289 203]
 [124 854]]
```

In [118]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.59      0.64       492
           1       0.81      0.87      0.84       978

    accuracy                           0.78      1470
   macro avg       0.75      0.73      0.74      1470
weighted avg       0.77      0.78      0.77      1470
```

Future Predictions

In [119]:

```
df_new = df.sample(1)
```

In [120]:

```
df_new
```

Out[120]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	al
4611	5.9	0.42	0.36	2.4	0.034	19.0	77.0	0.99184	3.25	0.48	

In [121]:

```
df_new.shape
```

Out[121]:

```
(1, 12)
```

In [122]:

```
x_new = df_new.drop(['quality'], axis = 1)
```

In [123]:

```
x_new = ss.fit_transform(x_new)
```

In [124]:

```
y_pred_new = svc.predict(x_new)
```

In [125]:

```
y_pred_new
```

Out[125]:

```
array([1], dtype=int64)
```