Wine Quality Prediction using Support Vector Machine

Get Understanding about Data set

Import Library

```
import pandas as pd
```

```
import numpy as np
```

Import CSV as DataFrame

Use URL of file directly

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

or use local file path in jupyter Notebook

```
# df = pd.read_csv(r'C:\Users\YBI Foundation\Desktop\WhiteQuality.csv')
```

or use file after uplaoding file in Google Colab Notebook

```
# df = pd.read_csv(r'/content/WhiteWineQuality.csv')
```

Get the First Five Rows of Dataframe

```
df.head()
```

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

Next steps: Generate code with df | View recommended plots | New interactive sheet

Get Information of DataFrame

```
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 the Summary Statistics

```
df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	to sul diox
count	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000
mean	6.854788	0.278241	0.334192	6.391415	0.045772	35.308085	138.360
std	0.843868	0.100795	0.121020	5.072058	0.021848	17.007137	42.498
min	3.800000	0.080000	0.000000	0.600000	0.009000	2.000000	9.000
25%	6.300000	0.210000	0.270000	1.700000	0.036000	23.000000	108.000
50%	6.800000	0.260000	0.320000	5.200000	0.043000	34.000000	134.000
75%	7.300000	0.320000	0.390000	9.900000	0.050000	46.000000	167.000
max	14.200000	1.100000	1.660000	65.800000	0.346000	289.000000	440.000

Get Columns Names

```
df.columns
```

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

Get Columns Names

```
df.shape
```

```
(4898, 12)
```

Get Shape of DataFrame

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

quality	count
6	2198
5	1457
7	880
8	175
4	163
3	20
9	5

dtype: int64

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

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
4	7.129448	0.381227	0.304233	4.628221	0.050098	23.358896	125.279141	0.994277
5	6.933974	0.302011	0.337653	7.334969	0.051546	36.432052	150.904598	0.995263
6	6.837671	0.260564	0.338025	6.441606	0.045217	35.650591	137.047316	0.993961
7	6.734716	0.262767	0.325625	5.186477	0.038191	34.125568	125.114773	0.992452
8	6.657143	0.277400	0.326514	5.671429	0.038314	36.720000	126.165714	0.992236
9	7.420000	0.298000	0.386000	4.120000	0.027400	33.400000	116.000000	0.991460

Define y (dependent or label or target variable) and X (independent or features or attribute Variable)

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

```
y.shape
```

```
(4898,)
```

```
y
```

	quality
0	6
1	6
2	6
3	6
4	6
...	...
4893	6
4894	5
4895	6
4896	7
4897	6

4898 rows × 1 columns

dtype: int64

```
lfur dioxide', 'total sulfer dioxide', 'density', 'pH', 'sulphates', 'alcohol']]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-21-b88da4b1b7fe> in <cell line: 1>()
----> 1 x = df[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
'chlorides', 'free sulfur dioxide', 'total sulfer dioxide', 'density', 'pH', 'sulphates',
'alcohol']]

                                 2 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in
_raise_if_missing(self, key, indexer, axis_name)
   6177
   6178             not_found = list(ensure_index(key)[missing_mask.nonzero()
[0]].unique())
-> 6179             raise KeyError(f"{not_found} not in index")
   6180
   6181     @overload

KeyError: "['total sulfer dioxide'] not in index"
```

Next steps: Explain error

or use.drop function to define X

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

```
x.shape
```

```
(4898, 11)
```

x

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

4898 rows × 11 columns

Next steps: Generate code with x | View recommended plots | New interactive sheet

Get X Variable Standardized

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

x = ss.fit_transform(x)

x
```

```
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

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

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

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

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

Get Model Train

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

```
svc = SVC()
```

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

```
SVC()
```

Get Model Prediction

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

```
y_pred.shape
```

```
(1470,)
```

```
y_pred
```

```
array([5, 7, 5, ..., 5, 5, 5])
```

Get ModelEvaluation

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

```
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]]
```

```
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

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: Undefinec
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: Undefinec
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: Undefinec
  _warn_prf(average, modifier, msg_start, len(result))
```

Get Model Re-run with Two Class Created for Wine Quality

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

```
  File "<ipython-input-46-40e7759fa3bd>", line 1
    y = df['quality'].apply(lamda y_value: 1 if y_value>=6 else 0)
                            ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

Next steps: Fix error

```
y.value_counts()
```

quality	count
6	2198
5	1457
7	880
8	175
4	163
3	20
9	5

dtype: int64

Get Train Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, stratify= y, random_
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-54-1508bbec45f8> in <cell line: 1>()
----> 1 X_train, X_test, y_train , y_test = train_test_split(X,y, test_size = 0.3,
stratify= y, random_state=2529)

NameError: name 'X' is not defined
```

Next steps: Explain error

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

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

Get Model Train

```
from sklearn.svm import SVC

svc = SVC()

svc.fit(x_train, y_train)
```

```
SVC
SVC()
```

Get Model Prediction

```
y_pred = svc.predict(x_test)

y_pred.shape
```

```
(1470,)
```

```
y_pred
```

```
array([5, 7, 5, ..., 5, 5, 5])
```

Get Model Evalution

```
from sklearn.metrics import confusion_matrix, classification_report

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]]
```

```
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

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: Undefinec
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: Undefinec
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: Undefinec
  _warn_prf(average, modifier, msg_start, len(result))
```

Get Future Pediction

Let select a random sample from existing dataset as new value

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

```
df_new
```

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.