Internship project

Title: Wine Quality Prediction

Subtitle: Ensuring Data Quality for Reliable Insights

Presented by: Faleye Doyin Opeyemi

Date: 18-8-2025

1. Data Loading & Data collection ¶

In [48]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import accuracy_score
```

In [49]:
```
df = pd.read_csv("C:\\Users\\FALEYE DOYINSOLA\\Project 7 WineQT dataset excel.csv
```

In [50]:
```
# preveiw the data
df.head()
```

Out[50]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	q
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	

2. Inspect Dataset

In [51]:
```
# checking for the information of the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

In [52]:
```
# Summary Statistics of the data
df.describe()
```

Out[52]:

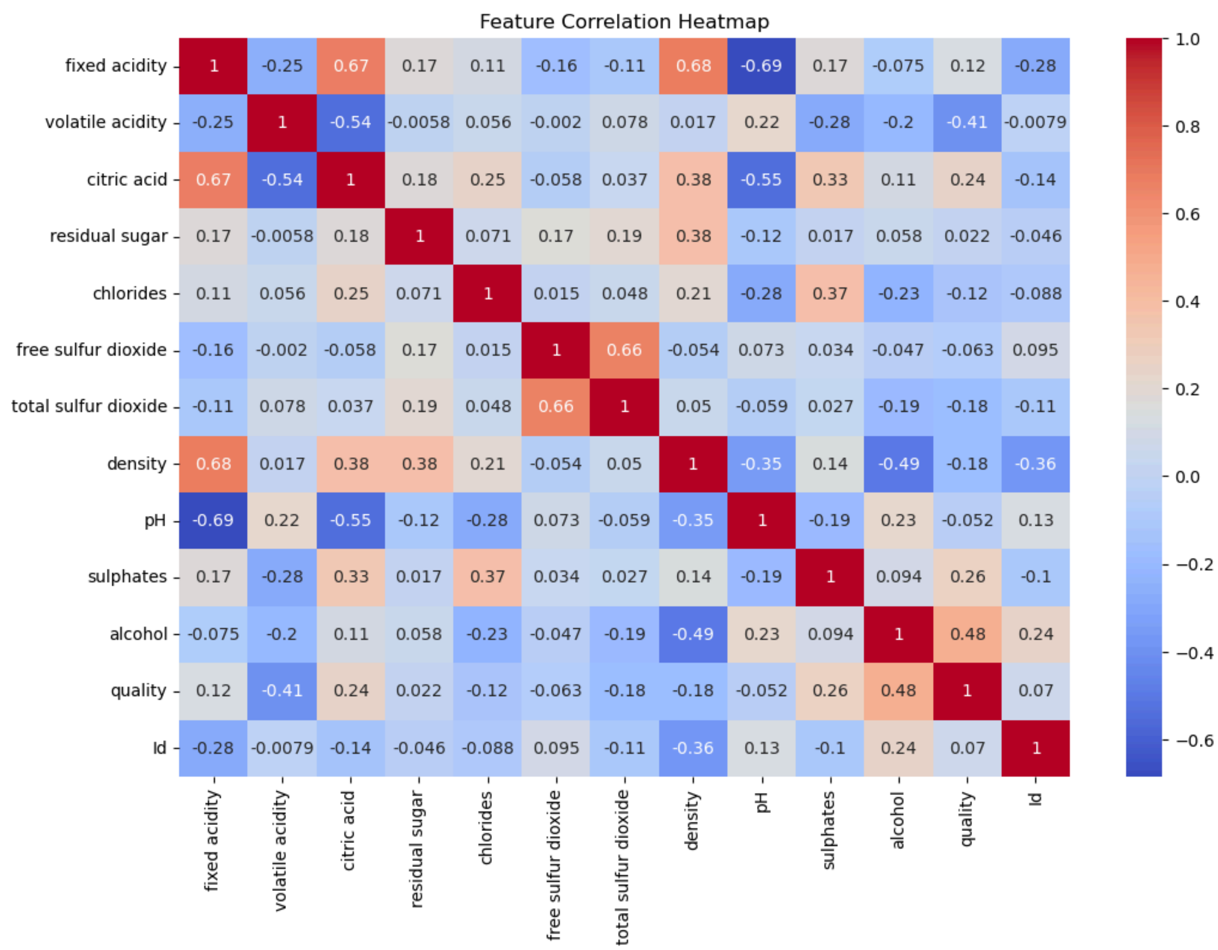
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000
mean	8.311111	0.531339	0.268364	2.532152	0.086933	15.615486	45.914698
std	1.747595	0.179633	0.196686	1.355917	0.047267	10.250486	32.782130
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000
25%	7.100000	0.392500	0.090000	1.900000	0.070000	7.000000	21.000000
50%	7.900000	0.520000	0.250000	2.200000	0.079000	13.000000	37.000000
75%	9.100000	0.640000	0.420000	2.600000	0.090000	21.000000	61.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	68.000000	289.000000

3. Exploratory Data Analysis (EDA)

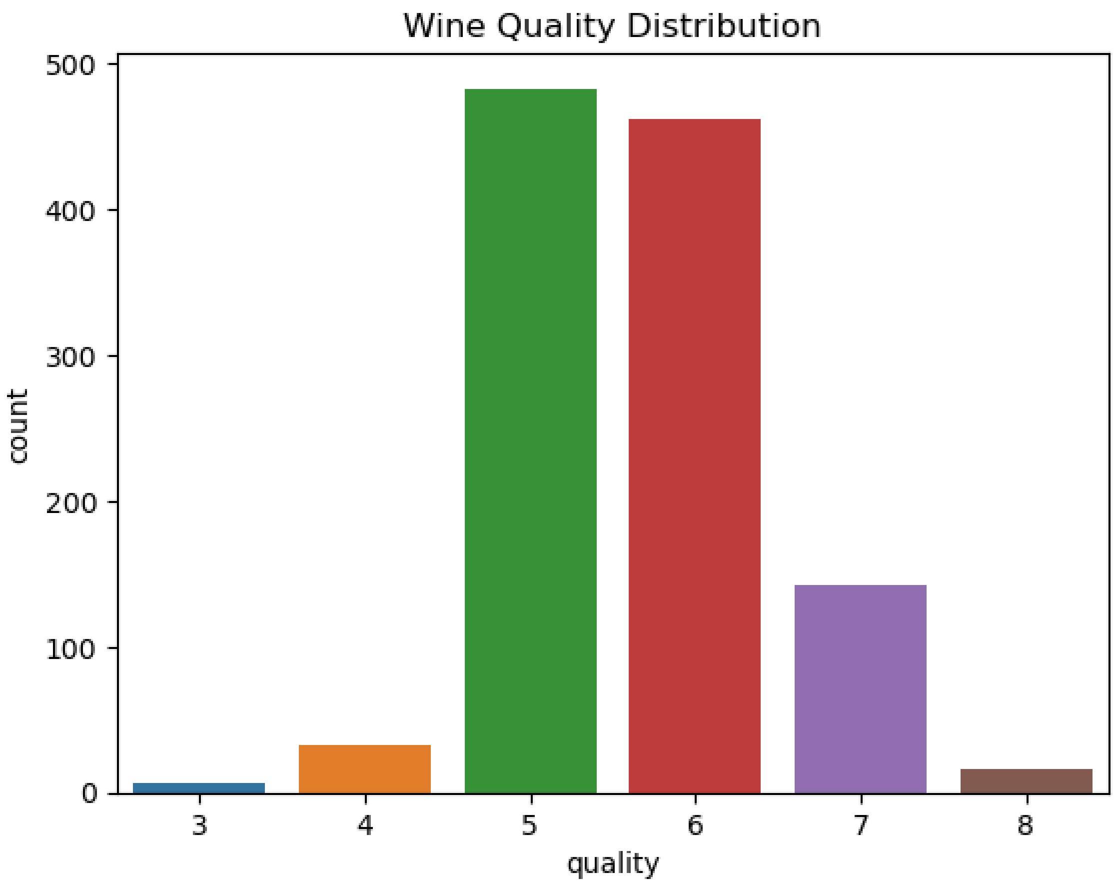
In [53]:
```
# Check for missing values
df.isnull().sum()
```

Out[53]:
```
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
Id                      0
dtype: int64
```

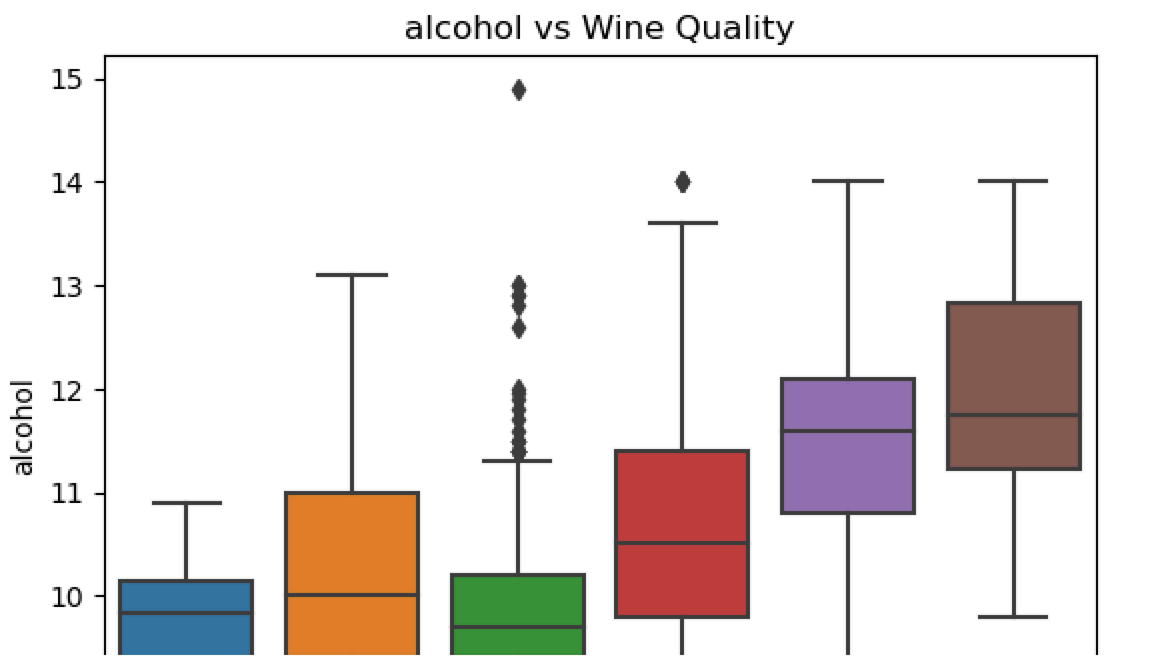
In [54]:
```
#Correlation heatmap
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Feature Correlation Heatmap')
plt.show()
```



In [55]:
```
# let see the countplot distribution of the wine quality
sns.countplot(df,x='quality')
plt.title('Wine Quality Distribution')
plt.show()
```



In [56]:
```python
# boxplot for key features
features = ['alcohol', 'volatile acidity', 'density', 'pH', 'sulphates']
for feature in features:
    sns.boxplot(x='quality', y=feature, data=df)
    plt.title(f'{feature} vs Wine Quality')
    plt.show()
```



4.Data Preprocessing

#a. Feature and target split

In [57]:
```python
# let drop id from the Dataset
df.drop('Id', axis=1, inplace=True)
```

In [58]:
```python
# preveiw the data
df.head()
```

Out[58]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	q
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 [59]:
```
# checking for how many rows and columns we have in the dataset
df.shape
```

Out[59]:
```
(1143, 12)
```

In [60]:
```
# to check for missing values i.e empty cells
df.isnull().sum()
```

Out[60]:
```
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 [61]:
```
# let create the wine quqlity from quality
df['Wine_Quality'] = df['quality'].apply(lambda q: 0 if q <= 4 else (1 if q <= 6
```

In [62]:
```
df['Wine_Quality'].unique()
```

Out[62]:
```
array([1, 2, 0], dtype=int64)
```

In [63]:
```
df.head()
```

Out[63]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	q
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 [64]:
```
df.drop('quality', axis=1, inplace=True)
```

In [65]:
```
#from sklearn.preprocessing import StandardScaler
# Scaling the dataset
```

In [66]:
```
x = df.drop('Wine_Quality', axis=1)
y = df['Wine_Quality']
```

In [67]:
```
standard = StandardScaler()
```

In [68]:
```
x = standard.fit_transform(x)
```

In [69]:
```
x
```

Out[69]:
```
array([[-0.52157961,  0.93933222, -1.36502663, ...,  1.27069495,
        -0.57365783, -0.96338181],
       [-0.29259344,  1.94181282, -1.36502663, ..., -0.70892755,
         0.1308811 , -0.59360107],
       [-0.29259344,  1.27349242, -1.16156762, ..., -0.32577481,
        -0.04525363, -0.59360107],
       ...,
       [-1.20853813,  0.38239855, -0.9581086 , ...,  0.88754221,
        -0.45623467,  0.05351522],
       [-1.38027776,  0.10393172, -0.8563791 , ...,  1.33455374,
         0.60057372,  0.70063152],
       [-1.38027776,  0.6330187 , -0.75464959, ...,  1.65384769,
         0.30701583, -0.22382033]])
```

In [70]:
```
# Train_Test_Split
from sklearn.model_selection import train_test_split
```

In [71]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,random_state=42)
```

5. Model Training and Evaluation

In [72]:
```
#a. Random Forest Classifier
rfc = RandomForestClassifier()
```

In [73]:
```
rfc.fit(xtrain,ytrain)
```

Out[73]:
```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [74]:
```
# prediction
rfcpred = rfc.predict(xtest)
```

In [75]:
```
rfcpred[:10]
```

Out[75]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [76]:
```
# checking for accuracy_score
rfc_acc = accuracy_score(rfcpred,ytest)*100
```

In [77]:
```
rfc_acc
```

Out[77]:
```
89.08296943231441
```

In [78]:
```
#b. Support Vector Classifier
svm = SVC()
```

In [79]:
```
svm.fit(xtrain,ytrain)
```

Out[79]:
```
SVC()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [80]:
```
# prediction
svmpred = svm.predict(xtest)
```

In [81]:
```
svmpred[:10]
```

Out[81]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [82]:
```
# checking for accuracy_score
svm_acc = accuracy_score(svmpred,ytest)*100
```

In [83]:
```
svm_acc
```

Out[83]:
```
86.46288209606988
```

In [84]:
```
#c. Stochastic Gradient Descent
sgd = SGDClassifier()
```

In [85]:
```
sgd.fit(xtrain,ytrain)
```

Out[85]:
```
SGDClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [86]:
```
# Prediction
sgdpred = sgd.predict(xtest)
```

In [87]:
```
sgdpred[:10]
```

Out[87]:
```
array([1, 1, 1, 1, 2, 2, 1, 1, 1, 1], dtype=int64)
```

In [88]:
```
# checking for accoracy_score
sgd_acc = accuracy_score(sgdpred,ytest)*100
```

In [89]:
```
sgd_acc
```

Out[89]:
```
83.4061135371179
```

```
# 📊 Interpretation
Three machine learning models were employed to classify wine quality:
•   Random Forest Classifier (RFC): Achieved the highest accuracy at 89%
•   Support Vector Classifier (SVC): Delivered an accuracy of 86%
•   Stochastic Gradient Descent (SGD): Reached an accuracy of 83%
Among these, the Random Forest Classifier demonstrated superior performance
in terms of predictive accuracy and robustness. Given its ensemble nature and
ability to handle feature interactions effectively,
RFC is recommended as the primary model for wine quality prediction.

✅ Recommendation
To further enhance model performance and potentially achieve even higher
accuracy, consider exploring advanced ensemble and hybrid models, such as:
```

- XGBoost: Known for its speed and performance in structured data tasks
- CatBoost: Handles categorical features efficiently and reduces overfitting
- LightGBM: Optimized for large datasets and faster training

These models often outperform traditional classifiers by leveraging gradient boosting techniques and can be fine-tuned for optimal results in wine quality prediction tasks.

In [ ]: