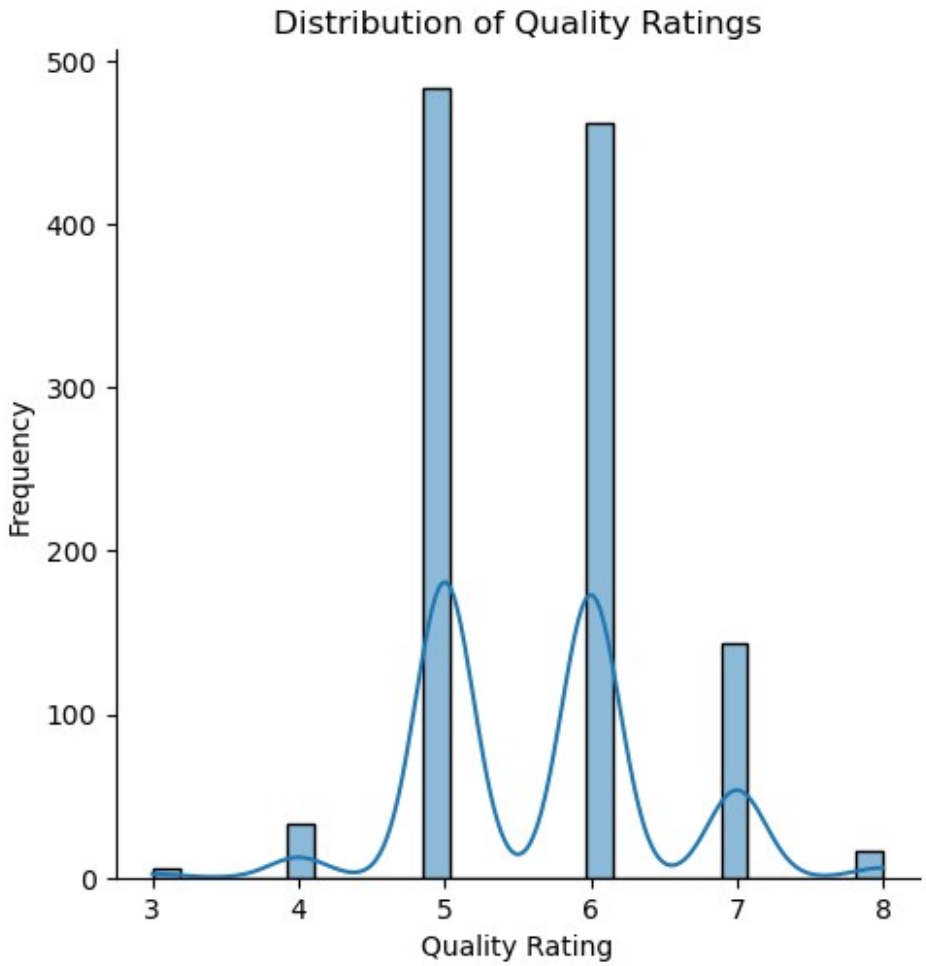
```python
# Import Neccessery Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import warnings

# Load the dataset
df = pd.read_csv('D:\Internship data\WineQT.csv')

# To rmove Warnings
warnings.filterwarnings("ignore")

# Analyze the dataset
print(df.head())
print(df.info())
print(df.describe())
```

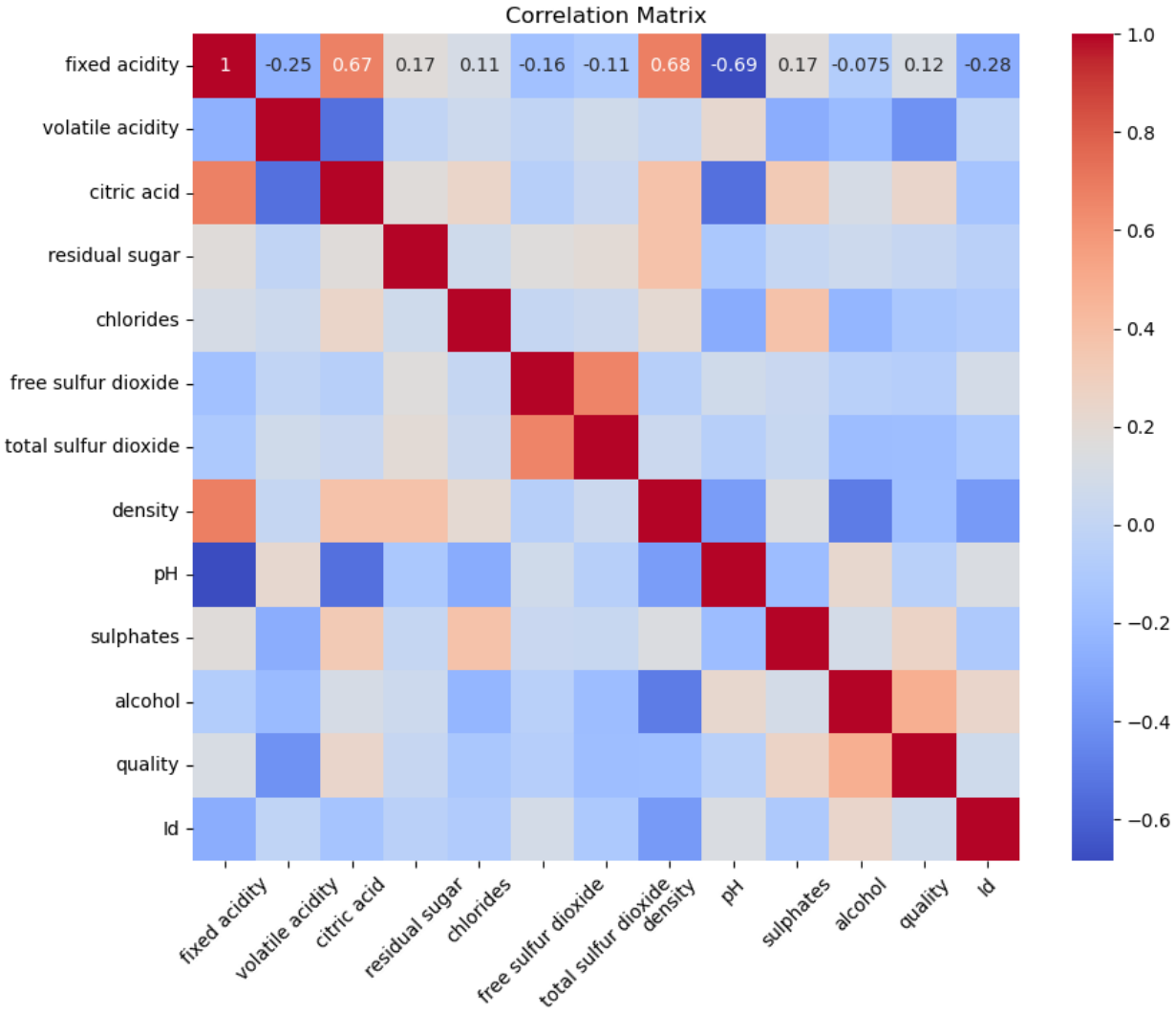
```
   fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0            7.4              0.70         0.00             1.9
0.076
1            7.8              0.88         0.00             2.6
0.098
2            7.8              0.76         0.04             2.3
0.092
3           11.2              0.28         0.56             1.9
0.075
4            7.4              0.70         0.00             1.9
0.076

   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


   alcohol  quality  Id
```

```
0      9.4        5   0
1      9.8        5   1
2      9.8        5   2
3      9.8        6   3
4      9.4        5   4
<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
None
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1143.000000       1143.000000  1143.000000     1143.000000
mean        8.311111          0.531339     0.268364        2.532152
std         1.747595          0.179633     0.196686        1.355917
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.392500     0.090000        1.900000
50%         7.900000          0.520000     0.250000        2.200000
75%         9.100000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000

         chlorides  free sulfur dioxide  total sulfur dioxide
density  \
count  1143.000000          1143.000000           1143.000000
1143.000000
mean      0.086933            15.615486             45.914698
0.996730
std       0.047267            10.250486             32.782130
0.001925
min       0.012000             1.000000              6.000000
0.990070
25%       0.070000             7.000000             21.000000
0.995570
50%       0.079000            13.000000             37.000000
```

```
0.996680
75%       0.090000            21.000000             61.000000
0.997845
max       0.611000            68.000000            289.000000
1.003690

                pH    sulphates      alcohol      quality           Id

count  1143.000000  1143.000000  1143.000000  1143.000000  1143.000000

mean      3.311015     0.657708    10.442111     5.657043   804.969379

std       0.156664     0.170399     1.082196     0.805824   463.997116

min       2.740000     0.330000     8.400000     3.000000     0.000000

25%       3.205000     0.550000     9.500000     5.000000   411.000000

50%       3.310000     0.620000    10.200000     6.000000   794.000000

75%       3.400000     0.730000    11.100000     6.000000  1209.500000

max       4.010000     2.000000    14.900000     8.000000  1597.000000


# Visualize the distribution of quality ratings
sns.displot(data=df, x="quality", kind="hist", kde=True)
plt.title('Distribution of Quality Ratings')
plt.xlabel('Quality Rating')
plt.ylabel('Frequency')
plt.show()
```




```
# Visualize the correlation between features
corr_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', square=True)
plt.title('Correlation Matrix')
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()
```




```
# Preprocess the data
X = df.drop(['quality'], axis=1)
y = df['quality']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Normalize the features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Construct the models
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
sgd_model = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42)
svc_model = SVC(kernel='rbf', C=1, gamma=0.1, random_state=42)
```

```
# Train the models
rf_model.fit(X_train, y_train)
sgd_model.fit(X_train, y_train)
svc_model.fit(X_train, y_train)
```

```
SVC(C=1, gamma=0.1, random_state=42)
```

```
# Evaluate the models
y_pred_rf = rf_model.predict(X_test)
y_pred_sgd = sgd_model.predict(X_test)
y_pred_svc = svc_model.predict(X_test)

print('Random Forest Classifier:')
print('Accuracy:', accuracy_score(y_test, y_pred_rf))
print('Classification Report:')
print(classification_report(y_test, y_pred_rf))
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred_rf))

print('Stochastic Gradient Descent Classifier:')
print('Accuracy:', accuracy_score(y_test, y_pred_sgd))
print('Classification Report:')
print(classification_report(y_test, y_pred_sgd))
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred_sgd))

print('Support Vector Classifier:')
print('Accuracy:', accuracy_score(y_test, y_pred_svc))
print('Classification Report:')
print(classification_report(y_test, y_pred_svc))
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred_svc))
```

```
Random Forest Classifier:
Accuracy: 0.6899563318777293
Classification Report:
              precision    recall  f1-score   support

           4       0.00      0.00      0.00         6
           5       0.73      0.75      0.74        96
           6       0.64      0.71      0.67        99
           7       0.76      0.62      0.68        26
           8       0.00      0.00      0.00         2

    accuracy                           0.69       229
   macro avg       0.43      0.41      0.42       229
weighted avg       0.67      0.69      0.68       229

Confusion Matrix:
[[ 0  3  3  0  0]
 [ 0 72 24  0  0]
```

```
 [ 0 24 70  5  0]
 [ 0  0 10 16  0]
 [ 0  0  2  0  0]]
Stochastic Gradient Descent Classifier:
Accuracy: 0.5851528384279476
Classification Report:
              precision    recall  f1-score   support

           4       0.00      0.00      0.00         6
           5       0.68      0.78      0.73        96
           6       0.66      0.39      0.49        99
           7       0.34      0.77      0.47        26
           8       0.00      0.00      0.00         2

    accuracy                           0.59       229
   macro avg       0.34      0.39      0.34       229
weighted avg       0.61      0.59      0.57       229

Confusion Matrix:
[[ 0  2  4  0  0]
 [ 0 75 12  8  1]
 [ 0 31 39 29  0]
 [ 0  2  4 20  0]
 [ 0  0  0  2  0]]
Support Vector Classifier:
Accuracy: 0.6593886462882096
Classification Report:
              precision    recall  f1-score   support

           4       0.00      0.00      0.00         6
           5       0.69      0.75      0.72        96
           6       0.62      0.70      0.66        99
           7       0.71      0.38      0.50        26
           8       0.00      0.00      0.00         2

    accuracy                           0.66       229
   macro avg       0.41      0.37      0.38       229
weighted avg       0.64      0.66      0.64       229

Confusion Matrix:
[[ 0  4  2  0  0]
 [ 0 72 24  0  0]
 [ 0 27 69  3  0]
 [ 0  1 15 10  0]
 [ 0  0  1  1  0]]
```

```python
# Analyze feature importance
importance = rf_model.feature_importances_
print('Feature Importance:')
print(importance)
```

```
Feature Importance:
[0.06740713 0.09987361 0.06891662 0.05907279 0.07717214 0.05892458
 0.0866716  0.08145545 0.06913166 0.11407127 0.13596273 0.08134042]

# Visualize feature importance
importance = rf_model.feature_importances_
sns.barplot(x=importance, y=X.columns.tolist())
plt.title('Feature Importance')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```