Idea: Wine Quality Prediction

Description: The focus is on predicting the quality of wine based on its chemical characteristics, offering a real-world application of machine learning in the context of viticulture. The dataset encompasses diverse chemical attributes, including density and acidity, which serve as the features for three distinct classifier models.

```
#Importing necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# Access the CSV file with the  path STEP1 LOAD DATA
path="/content/drive/MyDrive/OASIS/WineQT.csv"
df = pd.read_csv(path, na_values=["NA", "NaN", "", "?","Not Available"])
```

```
df.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	2
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	3

Next steps: View recommended plots

```
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
```

```
# Check for missing values
print(df.isnull().sum())
```

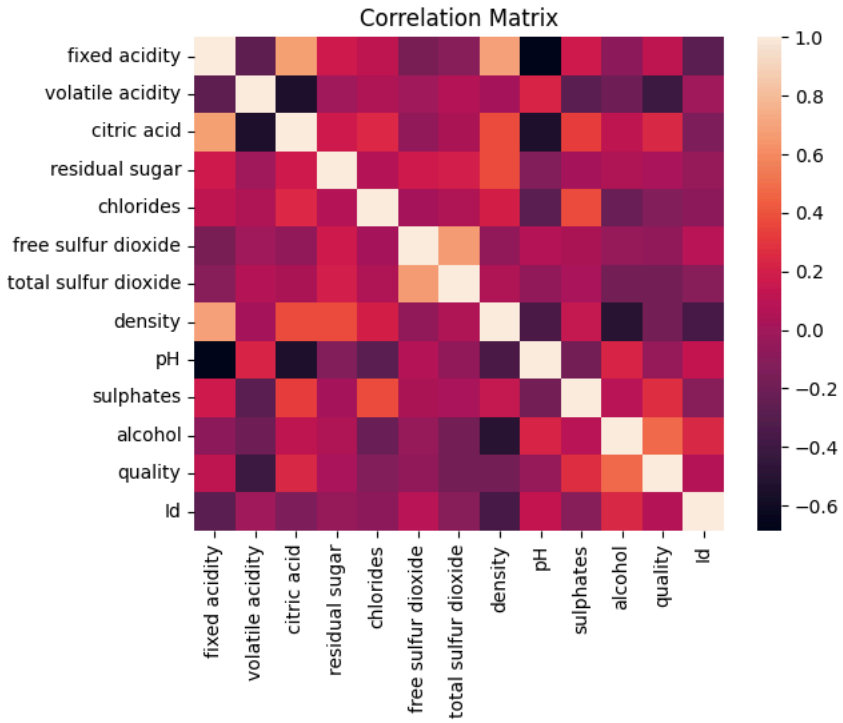
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
```

```
density               0
pH                    0
sulphates             0
alcohol               0
quality               0
Id                    0
dtype: int64
```

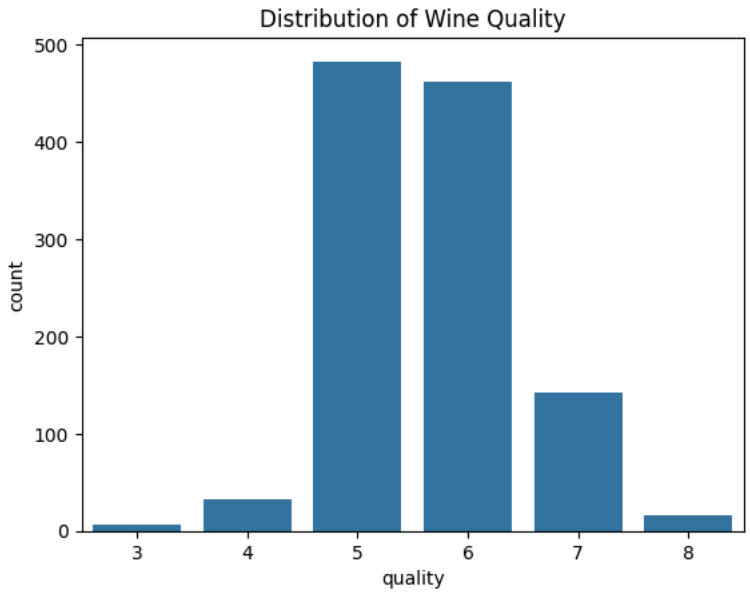
```
pip install pandas numpy seaborn matplotlib scikit-learn
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.25.2)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.49.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.3.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

```
correlation_matrix = df.corr()

#plt.figure(figsize=(10, 8))  # Adjust the figure size as needed
sns.heatmap(correlation_matrix)
plt.title('Correlation Matrix')
plt.show()
```




```
# Visualize the distribution of wine quality
sns.countplot(x='quality', data=df)
plt.title('Distribution of Wine Quality')
plt.show()
```




```
# Define features and target variable
X = df.drop('quality', axis=1)
y = df['quality']


# Splitting 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)


# Model 1: Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100)
rf_classifier.fit(X_train, y_train)
rf_pred = rf_classifier.predict(X_test)


print("Random Forest Classifier:")
print(classification_report(y_test, rf_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, rf_pred))
```

```
     Random Forest Classifier:
                   precision    recall  f1-score   support

                4       0.00      0.00      0.00         6
                5       0.71      0.80      0.75        96
                6       0.66      0.64      0.65        99
                7       0.65      0.65      0.65        26
                8       0.00      0.00      0.00         2

         accuracy                           0.69       229
        macro avg       0.41      0.42      0.41       229
     weighted avg       0.66      0.69      0.67       229

     Confusion Matrix:
     [[ 0  3  3  0  0]
      [ 0 77 18  1  0]
      [ 0 28 63  8  0]
      [ 0  0  9 17  0]
      [ 0  0  2  0  0]]
```

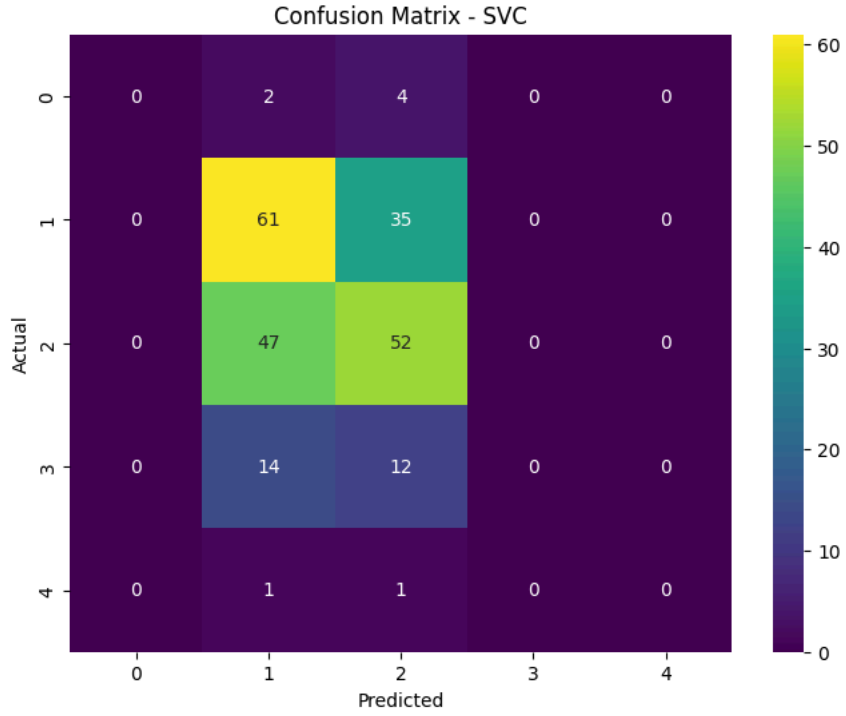
```
# Model 2: Stochastic Gradient Descent (SGD) Classifier
sgd_classifier = SGDClassifier()
sgd_classifier.fit(X_train, y_train)
sgd_pred = sgd_classifier.predict(X_test)


print("\nStochastic Gradient Descent (SGD) Classifier:")
print(classification_report(y_test, sgd_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, sgd_pred))
```

```
     Stochastic Gradient Descent (SGD) Classifier:
                   precision    recall  f1-score   support
```

```
              3       0.00      0.00      0.00         0
              4       0.00      0.00      0.00         6
              5       0.44      0.91      0.59        96
              6       0.00      0.00      0.00        99
              7       0.05      0.04      0.04        26
              8       0.00      0.00      0.00         2

       accuracy                           0.38       229
      macro avg       0.08      0.16      0.11       229
   weighted avg       0.19      0.38      0.25       229

   Confusion Matrix:
   [[ 0  0  0  0  0  0]
    [ 1  0  5  0  0  0]
    [ 1  0 87  0  8  0]
    [ 4  0 83  0 12  0]
    [ 3  0 22  0  1  0]
    [ 0  0  1  0  1  0]]
```

```
# Model 3: Support Vector Classifier (SVC)
svc_classifier = SVC()
svc_classifier.fit(X_train, y_train)
svc_pred = svc_classifier.predict(X_test)


print("\nSupport Vector Classifier (SVC):")
print(classification_report(y_test, svc_pred))
plt.figure(figsize=(8, 6))
cm = confusion_matrix(y_test, svc_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='viridis')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - SVC')
plt.show()
```

```
   Support Vector Classifier (SVC):
                 precision    recall  f1-score   support

              4       0.00      0.00      0.00         6
              5       0.49      0.64      0.55        96
              6       0.50      0.53      0.51        99
              7       0.00      0.00      0.00        26
              8       0.00      0.00      0.00         2

       accuracy                           0.49       229
      macro avg       0.20      0.23      0.21       229
   weighted avg       0.42      0.49      0.45       229
```

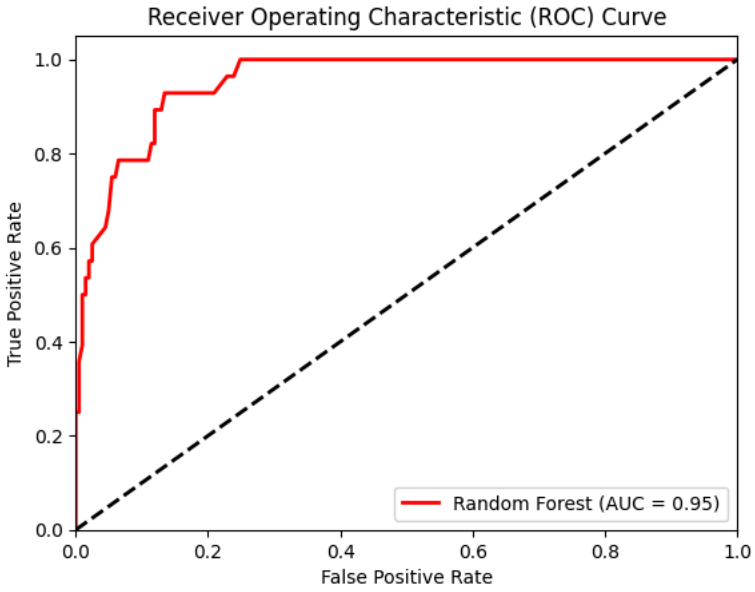


R Square Of The Model

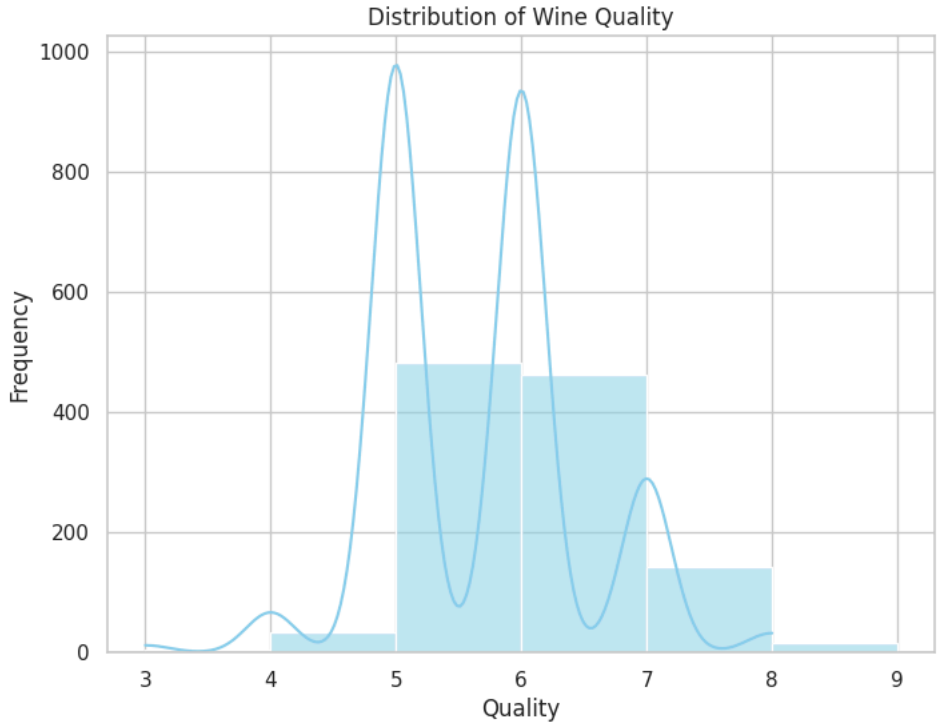
```
from sklearn.metrics import r2_score

# Step 8: Evaluate the models using R-squared
rf_r2 = r2_score(y_test, rf_classifier.predict(X_test))
sgd_r2 = r2_score(y_test, sgd_classifier.predict(X_test))
svc_r2 = r2_score(y_test, svc_classifier.predict(X_test))

print("Random Forest Classifier R-squared:", rf_r2)
print("Stochastic Gradient Descent Classifier R-squared:", sgd_r2)
print("Support Vector Classifier R-squared:", svc_r2)
```

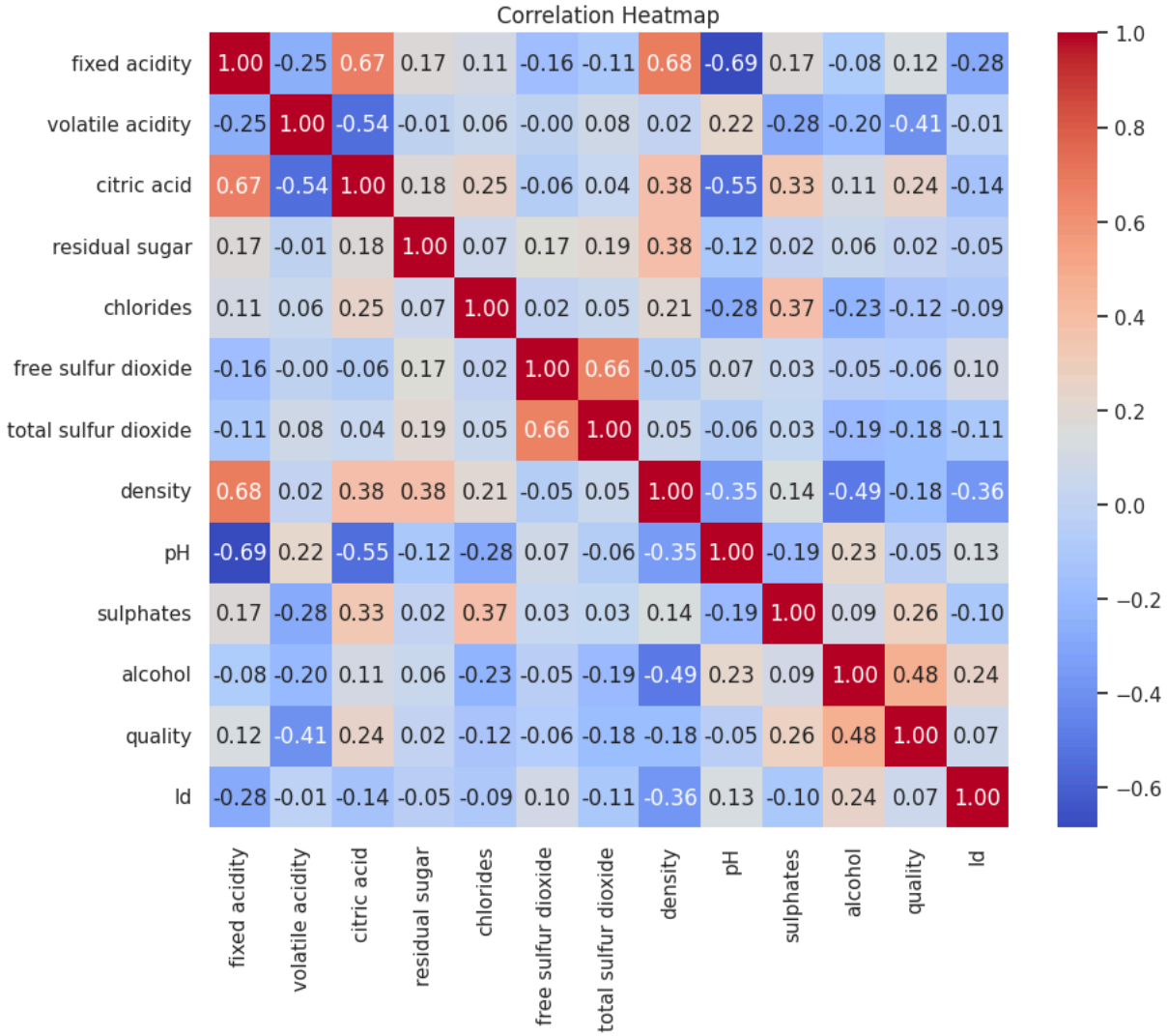
```
Random Forest Classifier R-squared: 0.2937427181139062
Stochastic Gradient Descent Classifier R-squared: -1.5032896991295992
Support Vector Classifier R-squared: -0.42036186690425525
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc, precision_recall_curve, average_precision_score, confusion_matrix



# Convert wine quality into binary classification (e.g., good quality vs. not good quality)
y_binary = y.apply(lambda x: 1 if x >= 7 else 0)

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y_binary, test_size=0.2, random_state=42)

# Train the Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)

# Predict probabilities for positive class
rf_probs = rf_classifier.predict_proba(X_test)[:, 1]

# Calculate false positive rate and true positive rate
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_probs)

# Calculate Area Under the Curve (AUC)
rf_roc_auc = auc(rf_fpr, rf_tpr)

# Plot ROC curve
plt.figure()
plt.plot(rf_fpr, rf_tpr, color='red', lw=2, label='Random Forest (AUC = %0.2f)' % rf_roc_auc)
plt.plot([0, 1], [0, 1], color='black', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```




```
import seaborn as sns
import matplotlib.pyplot as plt

# Set Seaborn style
sns.set(style="whitegrid")

# Plot histogram of wine quality
plt.figure(figsize=(8, 6))
sns.histplot(df['quality'], bins=range(3, 10), kde=True, color='skyblue')
plt.title('Distribution of Wine Quality')
plt.xlabel('Quality')
plt.ylabel('Frequency')
plt.show()

# Plot correlation heatmap
plt.figure(figsize=(10, 8))
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()

# Plot pairplot
plt.figure(figsize=(10, 8))
sns.pairplot(df, diag_kind='kde', hue='quality')
plt.suptitle('Pairplot of Wine Quality Dataset', y=1.02)
plt.show()

# Plot boxplot of chemical features by wine quality
plt.figure(figsize=(12, 8))
sns.boxplot(x='quality', y='alcohol', data=df, palette='muted')
plt.title('Boxplot of Alcohol Content by Wine Quality')
plt.xlabel('Quality')
plt.ylabel('Alcohol Content')
plt.show()
```






```
<Figure size 1000x800 with 0 Axes>
```