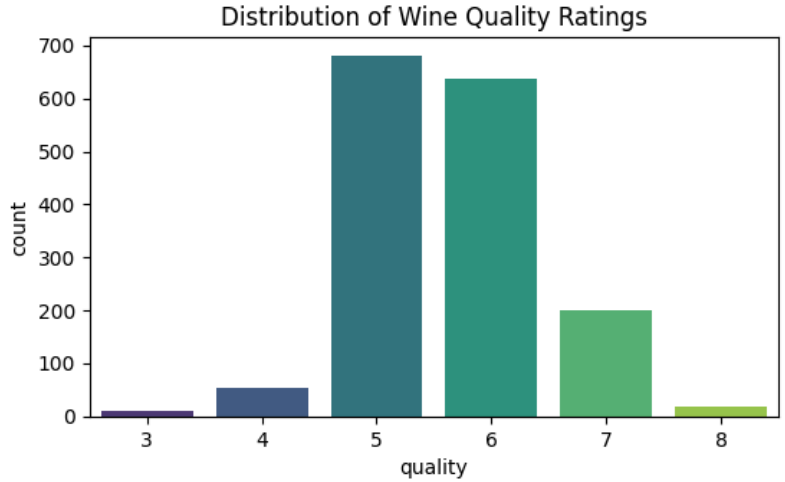
```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import warnings
warnings.filterwarnings('ignore')

# Load the dataset (using red wine data)
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
wine_df = pd.read_csv(url, sep=';')

# Display basic information about the dataset
print("Dataset Shape:", wine_df.shape)
print("\nFirst 5 rows of the dataset:")
print(wine_df.head())
print("\nDataset Information:")
print(wine_df.info())
print("\nSummary Statistics:")
print(wine_df.describe())
print("\nQuality Distribution:")
print(wine_df['quality'].value_counts().sort_index())

# Visualize the dataset
plt.figure(figsize=(12, 8))

# Distribution of wine quality
plt.subplot(2, 2, 1)
sns.countplot(x='quality', data=wine_df, palette='viridis')
plt.title('Distribution of Wine Quality Ratings')

# Correlation heatmap
plt.subplot(2, 2, 2)
correlation_matrix = wine_df.corr()
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap of Wine Features')

# Relationship between alcohol content and quality
plt.subplot(2, 2, 3)
sns.boxplot(x='quality', y='alcohol', data=wine_df, palette='viridis')
plt.title('Alcohol Content by Wine Quality')

# Relationship between volatile acidity and quality
plt.subplot(2, 2, 4)
sns.boxplot(x='quality', y='volatile acidity', data=wine_df, palette='viridis')
plt.title('Volatile Acidity by Wine Quality')
```

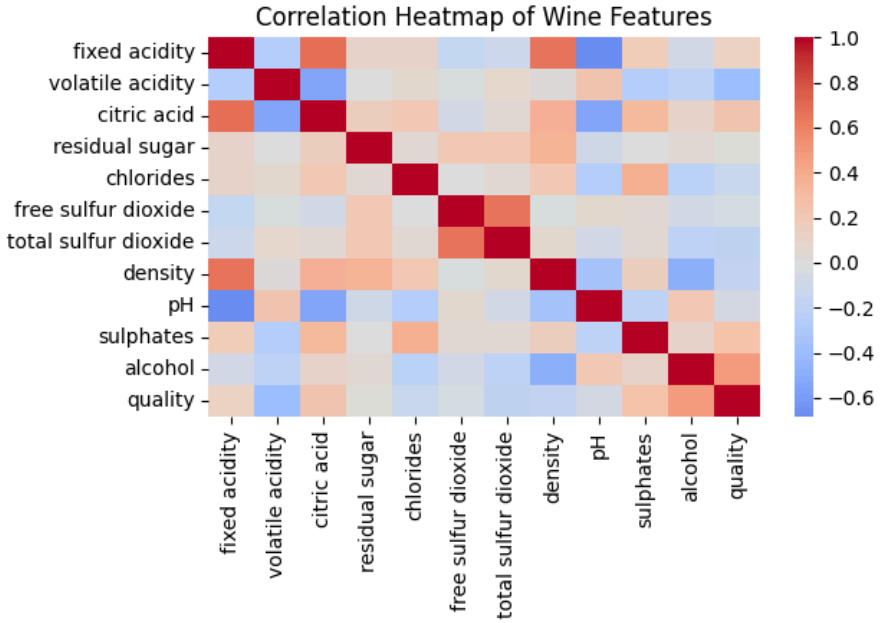
```
plt.tight_layout()
plt.savefig('wine_visualizations.png')
plt.show()

# Preprocess the data
# Convert quality into binary classification (good wine vs bad wine)
# We'll consider wine with quality >= 7 as good (1), and < 7 as bad (0)
wine_df['quality_binary'] = wine_df['quality'].apply(lambda x: 1 if x >= 7 else 0)

# Check class distribution
print("\nBinary Quality Distribution:")
print(wine_df['quality_binary'].value_counts())

# Separate features and target
X = wine_df.drop(['quality', 'quality_binary'], axis=1)
y = wine_df['quality_binary']

# 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, stratify=y)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create and train the logistic regression model
log_reg = LogisticRegression(random_state=42, max_iter=1000)
log_reg.fit(X_train_scaled, y_train)

# Make predictions
y_pred = log_reg.predict(X_test_scaled)
y_pred_proba = log_reg.predict_proba(X_test_scaled)[:, 1]

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel Accuracy: {accuracy:.4f}")

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion matrix
plt.figure(figsize=(8, 6))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Bad Wine', 'Good Wine'],
            yticklabels=['Bad Wine', 'Good Wine'])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix for Wine Quality Prediction')
plt.savefig('confusion_matrix.png')
plt.show()

# Feature importance
```

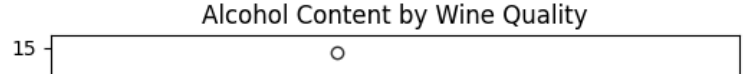
```
feature_importance = pd.DataFrame({
    'feature': X.columns,
    'importance': log_reg.coef_[0]
}).sort_values('importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x='importance', y='feature', data=feature_importance, palette='viridis')
plt.title('Feature Importance for Predicting Wine Quality')
plt.tight_layout()
plt.savefig('feature_importance.png')
plt.show()

print("\nFeature Importance:")
print(feature_importance)

# Test with a sample prediction
sample_wine = X.iloc[0:1]  # Take the first wine as a sample
sample_wine_scaled = scaler.transform(sample_wine)
prediction = log_reg.predict(sample_wine_scaled)
prediction_proba = log_reg.predict_proba(sample_wine_scaled)

print(f"\nSample Wine Prediction: {'Good Wine' if prediction[0] == 1 else 'Bad Wine'}")
print(f"Prediction Probability: {prediction_proba[0][1]:.4f}")
print(f"Actual Quality: {wine_df.iloc[0]['quality']} ({'Good Wine' if wine_df.iloc[0]['quality_binary'] == 1 else 'Bad Wine'})")
```

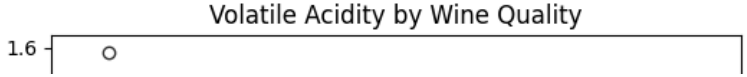
```
Dataset Shape: (1599, 12)

First 5 rows of the dataset:
   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
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5

Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
None

Summary Statistics:
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000
mean        8.319637          0.527821     0.270976        2.538806
std         1.741096          0.179060     0.194801        1.409928
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.390000     0.090000        1.900000
50%         7.900000          0.520000     0.260000        2.200000
75%         9.200000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000
```

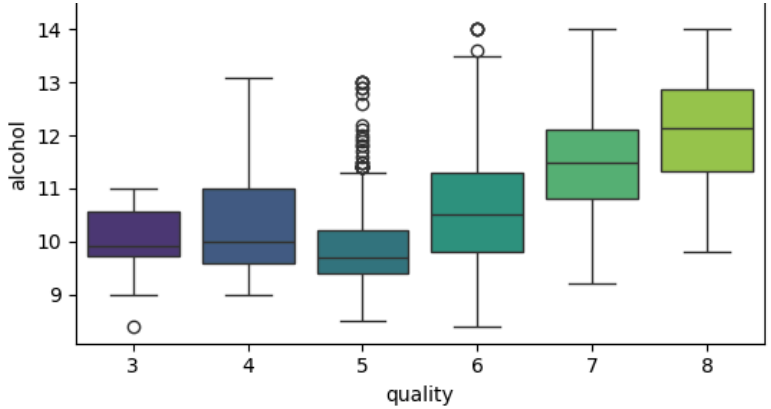
```
         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  1599.000000          1599.000000           1599.000000  1599.000000
mean      0.087467            15.874922             46.467792     0.996747
std       0.047065            10.460157             32.895324     0.001887
min       0.012000             1.000000              6.000000     0.990070
25%       0.070000             7.000000             22.000000     0.995600
50%       0.079000            14.000000             38.000000     0.996750
75%       0.090000            21.000000             62.000000     0.997835
max       0.611000            72.000000            289.000000     1.003690

                pH    sulphates      alcohol      quality
count  1599.000000  1599.000000  1599.000000  1599.000000
mean      3.311113     0.658149    10.422983     5.636023
std       0.154386     0.169507     1.065668     0.807569
min       2.740000     0.330000     8.400000     3.000000
25%       3.210000     0.550000     9.500000     5.000000
50%       3.310000     0.620000    10.200000     6.000000
75%       3.400000     0.730000    11.100000     6.000000
max       4.010000     2.000000    14.900000     8.000000

Quality Distribution:
quality
3     10
4     53
5    681
6    638
7    199
8     18
Name: count, dtype: int64
```

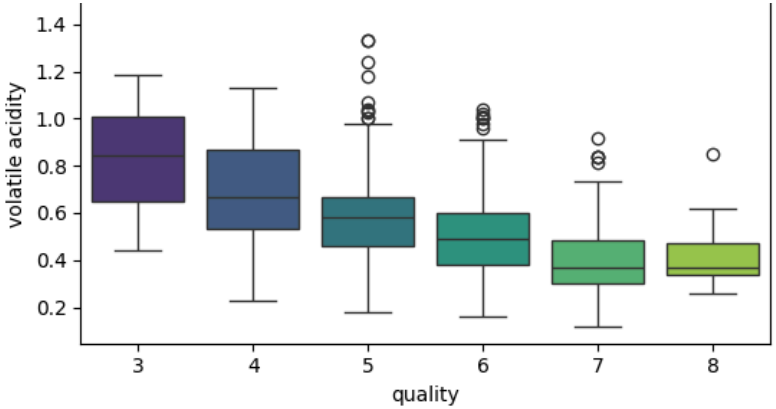













```
Binary Quality Distribution:
quality_binary
0    1382
1     217
Name: count, dtype: int64

Model Accuracy: 0.8938

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.97      0.94       277
           1       0.70      0.37      0.48        43

    accuracy                           0.89       320
   macro avg       0.80      0.67      0.71       320
weighted avg       0.88      0.89      0.88       320
```