Introduction

Project: Final Team Project Group 1 **Team**: Lokesh Upputri, Safwan Syed

Title: Predicting Wine Quality Using Machine Learning Techniques

This project focuses on developing a robust predictive model to assess wine quality based on physicochemical properties, utilizing the UCI Wine Quality dataset. The primary objectives include meticulous data preprocessing, comprehensive exploratory data analysis, strategic feature selection, and the application of diverse machine learning algorithms to classify wine quality. By evaluating the performance of these models, we aim to identify the most effective approach for accurately predicting wine quality.

Data Source: The dataset is sourced from the UCI Machine Learning Repository and can be accessed here.

Dataset Details:

- **Number of Variables**: 11
- **Size of Dataset**: 89.2 KB
- **Citation**: Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Wine Quality [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C56S3T.

Through this project, we seek to leverage machine learning techniques to uncover insights into wine quality prediction, ultimately contributing to the field of data-driven decision-making in the wine industry.

Data Cleaning/Preparation

In [5]:
```
import pandas as pd

# Read the red wine quality data
red_wine = pd.read_csv('winequality-red.csv', sep=';')

# Read the white wine quality data
white_wine = pd.read_csv('winequality-white.csv', sep=';')

# Combine the two DataFrames
combined_wine = pd.concat([red_wine, white_wine], ignore_index=True)

# Save the combined data to a new CSV file
combined_wine.to_csv('combined_winequality.csv', index=False, sep=';')
```

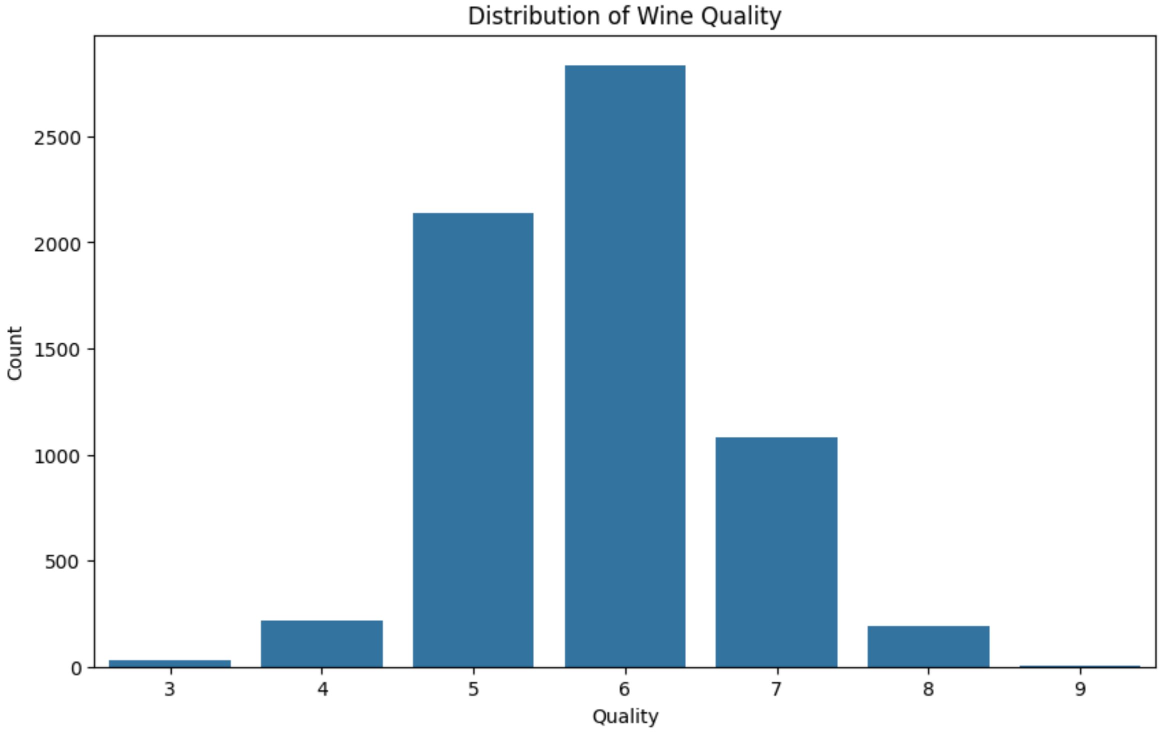
Exploratory Data Analysis

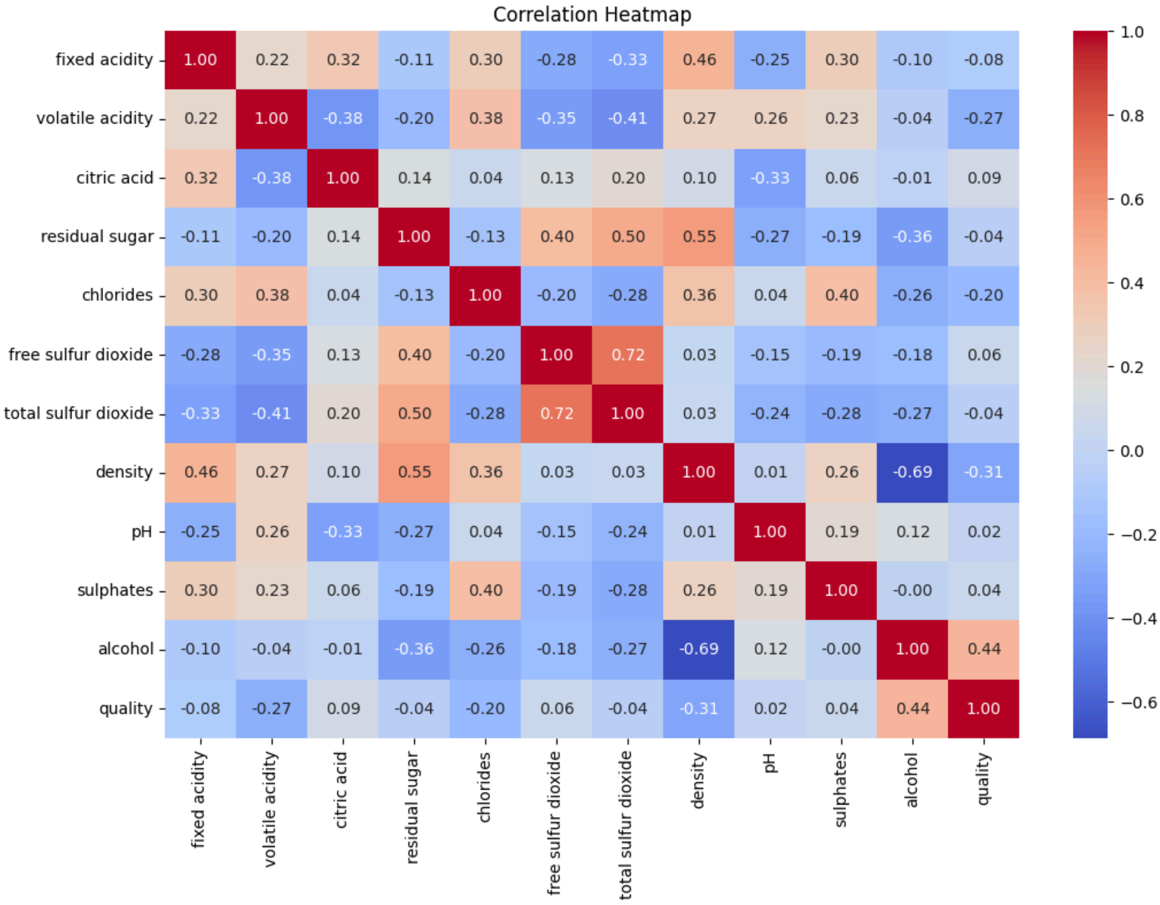
In [11]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

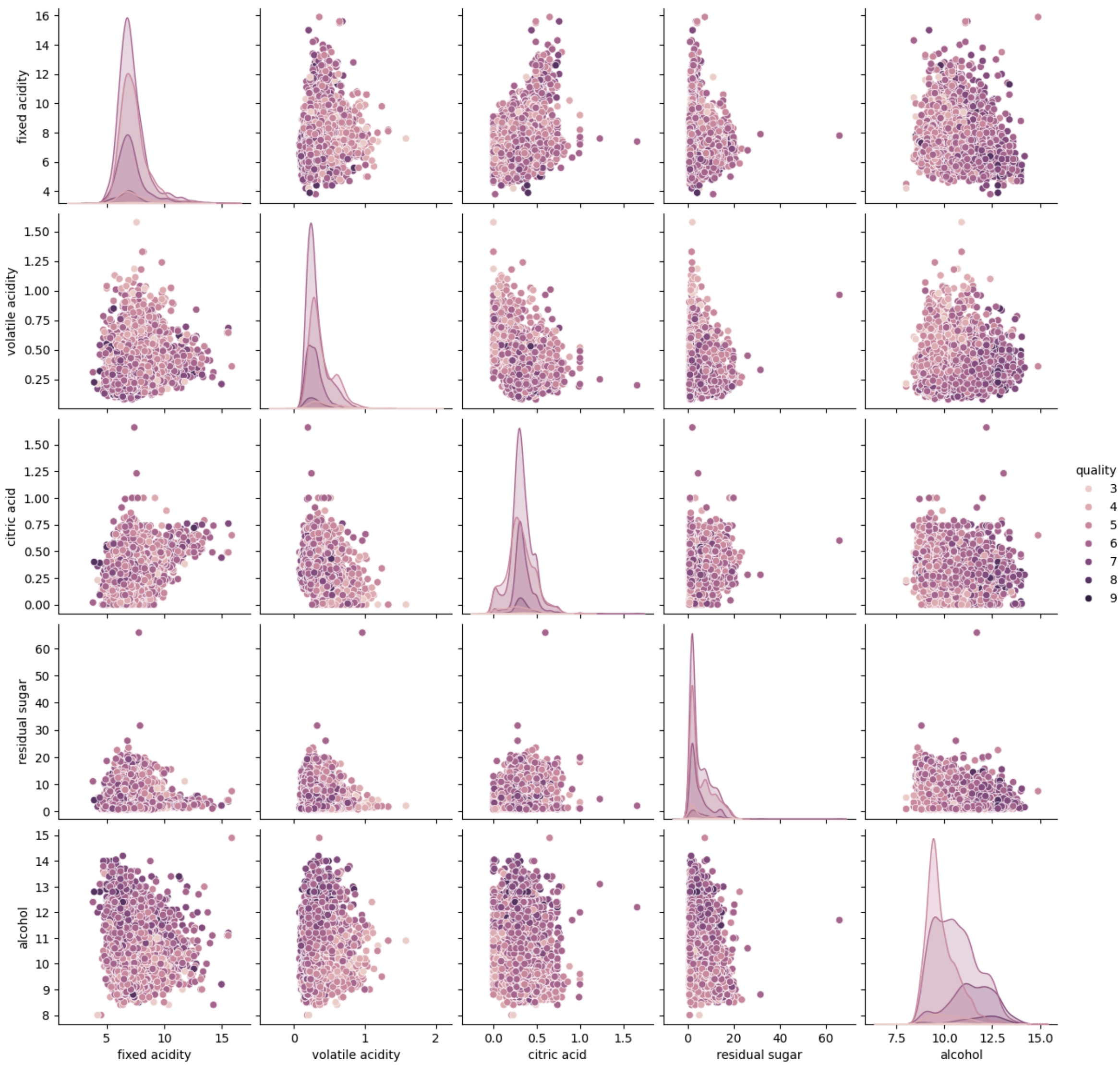
```python
# Load the dataset
data = pd.read_csv('combined_winequality.csv', sep=';')

# Display the first few rows of the dataset
print(data.head())

# Display the shape of the dataset
print("Shape of the dataset:", data.shape)

# Display data types of each column
print(data.dtypes)

# Summary statistics
print(data.describe())

# Check for missing values
print(data.isnull().sum())

# Distribution of wine quality
plt.figure(figsize=(10, 6))
sns.countplot(x='quality', data=data)
plt.title('Distribution of Wine Quality')
plt.xlabel('Quality')
plt.ylabel('Count')
plt.show()

# Correlation heatmap
plt.figure(figsize=(12, 8))
correlation = data.corr()
sns.heatmap(correlation, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# Pairplot for selected features
sns.pairplot(data, hue='quality', vars=['fixed acidity', 'volatile acidit
plt.show()

# Alcohol vs Quality
plt.figure(figsize=(10, 6))
sns.boxplot(x='quality', y='alcohol', data=data)
plt.title('Alcohol Content vs Quality')
plt.xlabel('Quality')
plt.ylabel('Alcohol Content')
plt.show()

# Fixed Acidity vs Quality
plt.figure(figsize=(10, 6))
sns.boxplot(x='quality', y='fixed acidity', data=data)
plt.title('Fixed Acidity vs Quality')
plt.xlabel('Quality')
plt.ylabel('Fixed Acidity')
plt.show()
```

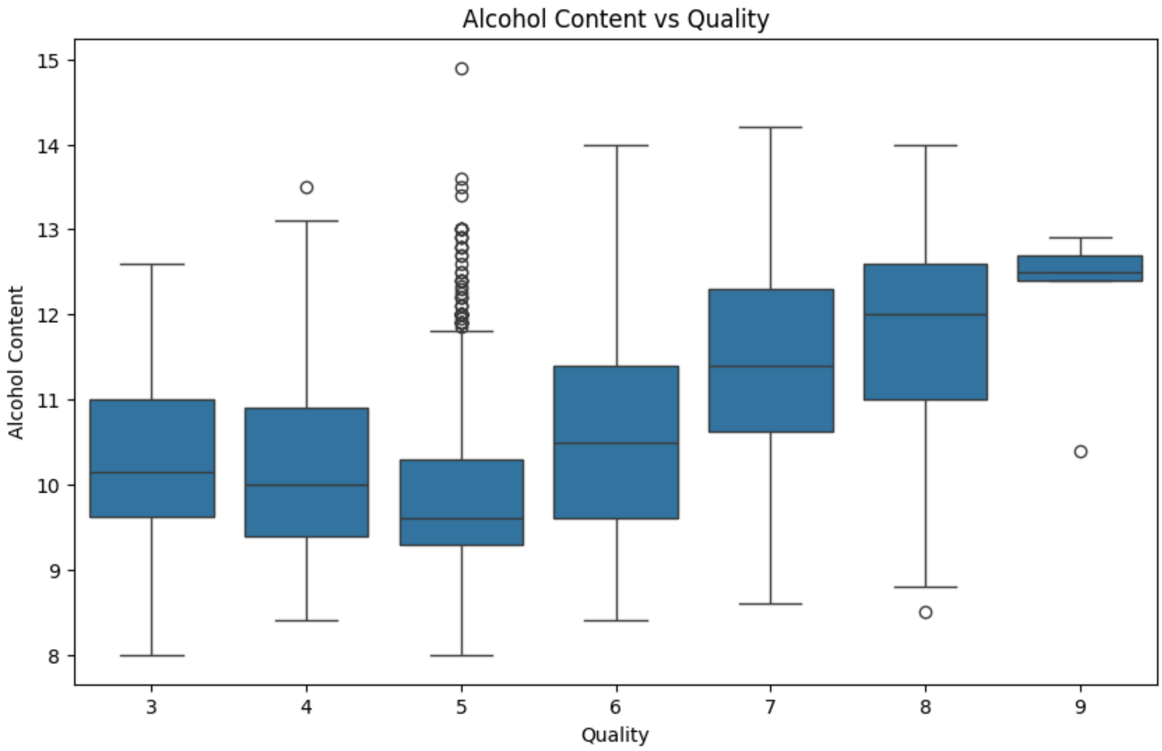
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      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
Shape of the dataset: (6497, 12)
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    6497.000000       6497.000000  6497.000000     6497.000000
mean        7.215307          0.339666     0.318633        5.443235
std         1.296434          0.164636     0.145318        4.757804
min         3.800000          0.080000     0.000000        0.600000
25%         6.400000          0.230000     0.250000        1.800000
50%         7.000000          0.290000     0.310000        3.000000
75%         7.700000          0.400000     0.390000        8.100000
max        15.900000          1.580000     1.660000       65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  6497.000000          6497.000000           6497.000000  6497.000000
mean      0.056034            30.525319            115.744574     0.994697
std       0.035034            17.749400             56.521855     0.002999
min       0.009000             1.000000              6.000000     0.987110
25%       0.038000            17.000000             77.000000     0.992340
50%       0.047000            29.000000            118.000000     0.994890
75%       0.065000            41.000000            156.000000     0.996990
max       0.611000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol      quality
count  6497.000000  6497.000000  6497.000000  6497.000000
mean      3.218501     0.531268    10.491801     5.818378
std       0.160787     0.148806     1.192712     0.873255
```

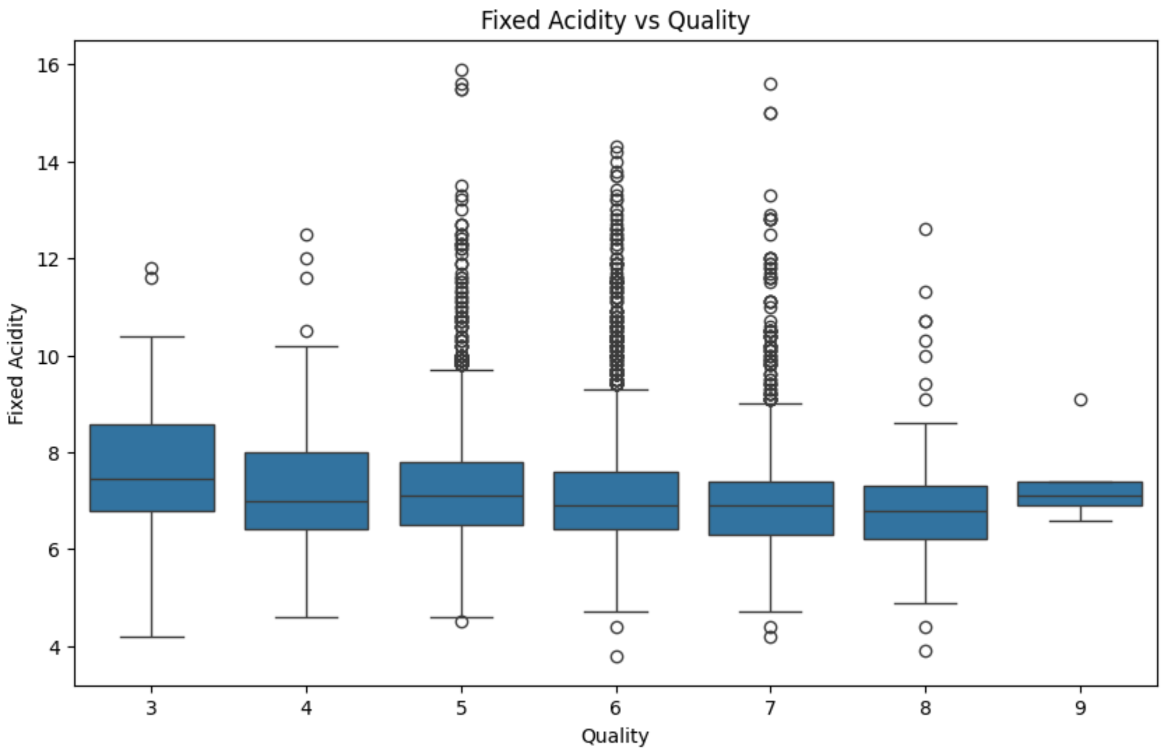
```
min       2.720000     0.220000     8.000000     3.000000
25%       3.110000     0.430000     9.500000     5.000000
50%       3.210000     0.510000    10.300000     6.000000
75%       3.320000     0.600000    11.300000     6.000000
max       4.010000     2.000000    14.900000     9.000000
fixed acidity           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
```




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


fixed acidity
volatile acidity
citric acid
residual sugar
alcohol
quality
3
4
5
6
7
8
9





Finding correlation

In [7]:
```
# Calculate the correlation matrix
correlation_matrix = data.corr()

# Extract the correlation of each feature with the target variable 'quali
quality_correlation = correlation_matrix['quality'].sort_values(ascending

# Display the correlation values
print(quality_correlation)
```

```
quality                 1.000000
alcohol                 0.444319
citric acid             0.085532
free sulfur dioxide     0.055463
sulphates               0.038485
pH                      0.019506
residual sugar         -0.036980
total sulfur dioxide   -0.041385
fixed acidity          -0.076743
chlorides              -0.200666
volatile acidity       -0.265699
density                -0.305858
Name: quality, dtype: float64
```

Random Forest Regressor model

In [13]:

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

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

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

# Create a Random Forest Regressor model
model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse:.3f}')  # Format to 3 decimal places
print(f'R^2 Score: {r2:.3f}')  # Format to 3 decimal places
```

```
Mean Squared Error: 0.371
R^2 Score: 0.497
```

Summary

The model's performance metrics are as follows:

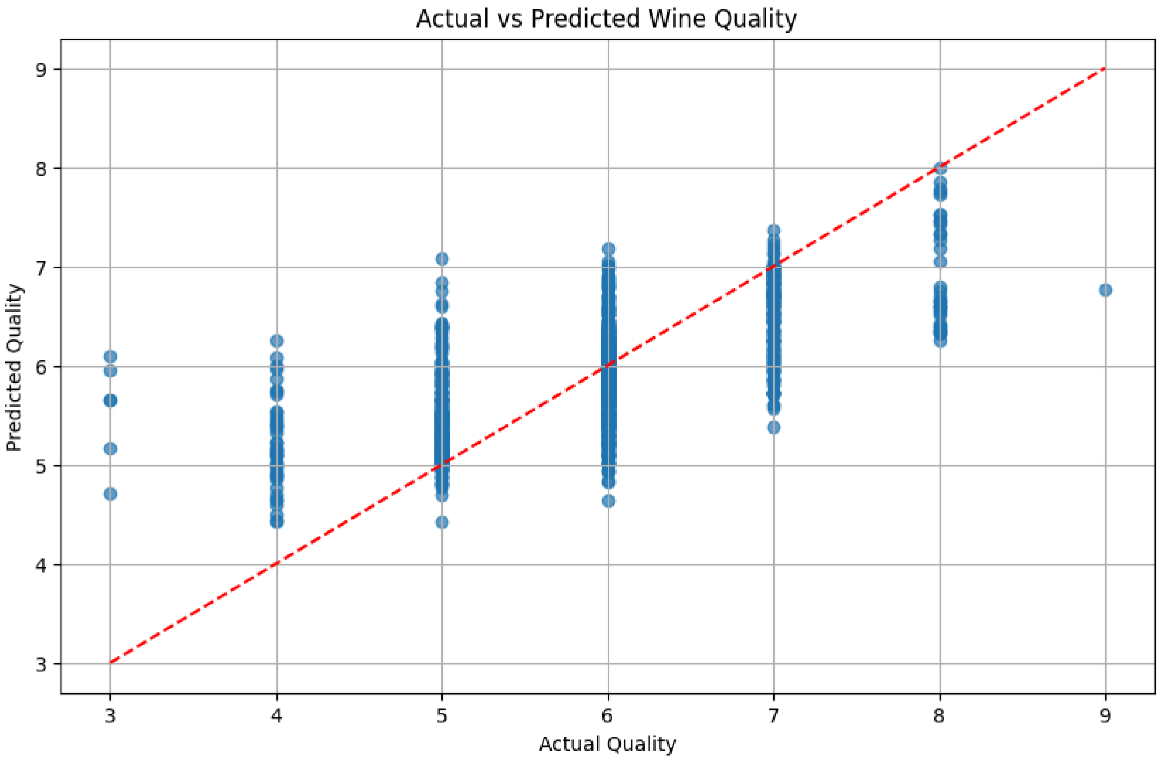
- **Mean Squared Error (MSE)**: 0.371
- **R^2 Score**: 0.497

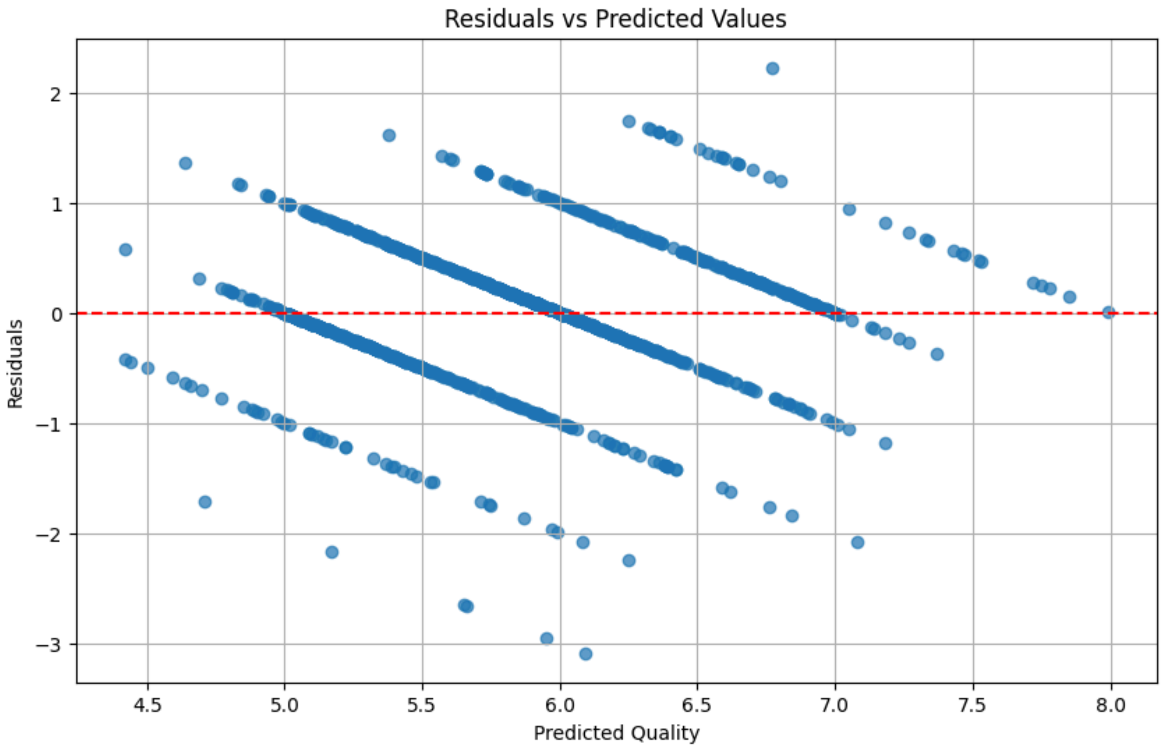
These metrics indicate that the model has a moderate predictive accuracy, with the R^2 score suggesting that approximately 49.7% of the variance in the data is captured by the model.

Scatter plot of actual vs predicted values, Plot residuals, Perform cross-validation

In [9]:

```
import matplotlib.pyplot as plt

# Scatter plot of actual vs predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')  # Line for perfe
plt.title('Actual vs Predicted Wine Quality')
plt.xlabel('Actual Quality')
plt.ylabel('Predicted Quality')
plt.grid()
plt.show()

# Calculate residuals
residuals = y_test - y_pred

# Plot residuals
plt.figure(figsize=(10, 6))
plt.scatter(y_pred, residuals, alpha=0.7)
plt.axhline(0, color='red', linestyle='--')
plt.title('Residuals vs Predicted Values')
plt.xlabel('Predicted Quality')
plt.ylabel('Residuals')
plt.grid()
plt.show()

from sklearn.model_selection import cross_val_score

# Perform cross-validation
cv_scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_
cv_rmse = (-cv_scores)**0.5  # Convert to RMSE
print(f'Cross-Validated RMSE: {cv_rmse.mean()} ± {cv_rmse.std()}')
```






```
Cross-Validated RMSE: 0.739209314436778 ± 0.03556426457587208
```

Summary

The cross-validated Root Mean Squared Error (RMSE) for the model is approximately 0.739, with a standard deviation of ±0.036. This indicates that the model's prediction error is relatively consistent across different subsets of the data, suggesting robust performance.

Evaluate 3 models

In [10]:

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor

# Initialize models
models = {
    'Random Forest': RandomForestRegressor(n_estimators=100, random_state
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42)
}

# Evaluate each model
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f'{name} - Mean Squared Error: {mse}, R^2 Score: {r2}')
```

```
Random Forest - Mean Squared Error: 0.3713522307692308, R^2 Score: 0.49718
520459177784
Linear Regression - Mean Squared Error: 0.5466964419580572, R^2 Score: 0.2
59767312979018
Decision Tree - Mean Squared Error: 0.7284615384615385, R^2 Score: 0.01365
5475650245497
```

Model Performance Evaluation

In this analysis, three different machine learning models were evaluated based on their predictive performance using Mean Squared Error (MSE) and the coefficient of determination (R^2 score). The models assessed were Random Forest, Linear Regression, and Decision Tree.

Random Forest:

Mean Squared Error (MSE): 0.371 R^2 Score: 0.497 The Random Forest model demonstrated the lowest MSE and the highest R^2 score among the three models, indicating that it provides the best fit to the data and has the highest predictive accuracy.

Linear Regression:

Mean Squared Error (MSE): 0.547 R^2 Score: 0.260 The Linear Regression model showed moderate performance with a higher MSE and a lower R^2 score compared to the Random Forest model, suggesting it captures less variance in the data.

Decision Tree:

Mean Squared Error (MSE): 0.728 R^2 Score: 0.014 The Decision Tree model had the highest MSE and the lowest R^2 score, indicating poor predictive performance and a weak fit to the data.

Conclusion

Based on the evaluation metrics, the Random Forest model outperforms both Linear Regression and Decision Tree models in terms of predictive accuracy and fit to the data. It is recommended to use the Random Forest model for this particular dataset and predictive task. Further tuning and validation may be necessary to optimize model performance.