Predicting Wine Quality Using Machine Learning: An Exploratory Study and Model Comparison

Group: 4

Names: Hemlatha Kaur Saran, George David Asirvatharaj, Raminder Singh

Module: Probability and Statistics for Artificial Intelligence

Course: Master in Applied Artificial Intelligence

Institution: University Of San Diego

Professor: Ms. Azka

Date: 20/06/2025

Table of Contents

Predicting Wine Quality Using Machine Learning: An Exploratory Study and Model Comparison

Context

Assessing the quality of wine ahead of time is important for both winemakers and people buying wine since it helps adapt production and buying choices. Using machine learning methods, experts can connect the chemical aspects of wine with how wine is perceived so quality predictions can be made with data (Jain et al., 2023). To analyze the data, we use a set of white wines that has several chemical measurements, such as acidity, sugar, pH, and alcohol, together with an expert score. Scores for quality are between zero and ten, though they rarely go above eight. All of these features help to show the factors that affect how well a wine is made.

The main issue being investigated is the choice between predicting the exact quality score using regression or using classification to tell wines apart based on quality. Scores of 7 or higher designate good quality in this report. We aim to analyze the dataset, create different machine-learning models for both regression and classification, compare them, and measure their results (Bhardwaj et al., 2022). Machine learning methods, and more specifically, Random Forest models, can predict the quality of white wine using chemical information, which is helpful for both growers and wine drinkers.

Data Cleaning and Preparation

In the first phase of analysis, we checked the dataset to see if any information was missing or incorrect. There are 12 columns in the white wine dataset: eleven feature columns with wine property information and just one target column for wine quality. Fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide,

density, pH, sulphates, and alcohol content are part of the feature columns (Yavas et al., 2025). The quality variable, considered the target, is given as a number between 0 and 10.

The examination showed that NAs are not included in the dataset, so we can select a more straightforward way to preprocess the data and use all the observations provided. Since all the columns are the right types for their values, the dataset can be directly used in machine learning algorithms because numeric values are stored using float or integers (Singla et al., 2024). Since most distance-based and gradient-based machine learning models are affected by the input feature scale, each input feature was scaled to avoid this issue. The data was reshaped by standardization so that each feature has a mean of zero and a standard deviation of one. Doing this improves how the algorithms gather information and perform, primarily for Support Vector Machines and k-nearest Neighbors.

To perform classification, the continuous quality measurements were changed to binary classes. Any wine with a good quality score (7 or more) was placed in class 1 ("Good"), and any wine with a poor score (below 7) was put in class 0 ("Bad"). This boundary was chosen to highlight the difference between higher-quality wines and other samples, making the problem clearly useful for wine testers. Completing these steps makes the following data analysis and model-building much smoother.

Exploratory Data Analysis (EDA)

Summary Statistics Table

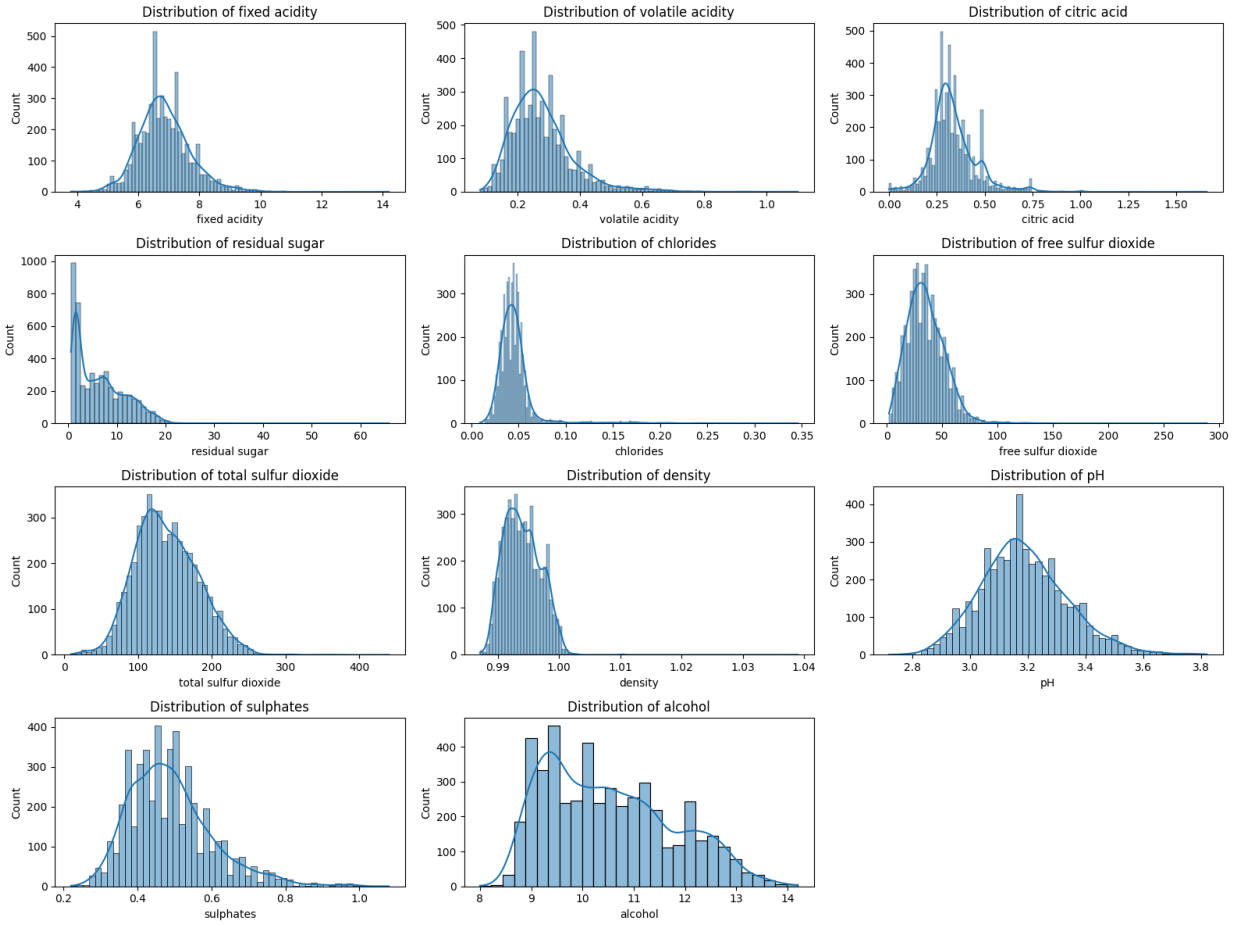
The data used contains 4,898 white wine samples that have a variety of chemical features. Most fixed acidity measurements are around 6.85, but they may be as low as 3.8 or as high as 14.2. Volatile acidity is usually 0.28, and most wines have alcohol levels of 8.4% to 14.2%, averaging 10.5%. Most wines are in the middle of the quality range, based on the fact that quality

scores run from 3 to 9, with an average score of around 5.88. The fact that sulphates and pH do not always stay the same indicates they may affect the wine's flavor.

Feature	Mean	Std Dev	Min	25%	50%	75%	Max
Fixed Acidity	6.85	0.84	3.80	6.30	6.80	7.30	14.20
Volatile Acidity	0.28	0.10	0.08	0.21	0.26	0.32	1.10
Citric Acid	0.33	0.12	0.00	0.27	0.32	0.39	1.66
Residual Sugar	6.39	5.07	0.60	1.70	5.20	9.90	65.80
Chlorides	0.05	0.02	0.01	0.04	0.04	0.05	0.35
Free Sulfur Dioxide	35.31	17.01	2.00	23.00	34.00	46.00	289.00
Total Sulfur Dioxide	138.37	42.50	9.00	108.00	134.00	167.00	440.00
Density	0.99	0.003	0.99	0.99	0.99	0.99	1.04
pH	3.19	0.15	2.90	3.09	3.18	3.28	3.82
Sulphates	0.49	0.15	0.41	0.41	0.47	0.55	1.08
Alcohol	10.51	1.07	8.40	9.50	10.40	11.40	14.20
Quality	5.88	0.87	3.00	5.00	6.00	6.00	9.00

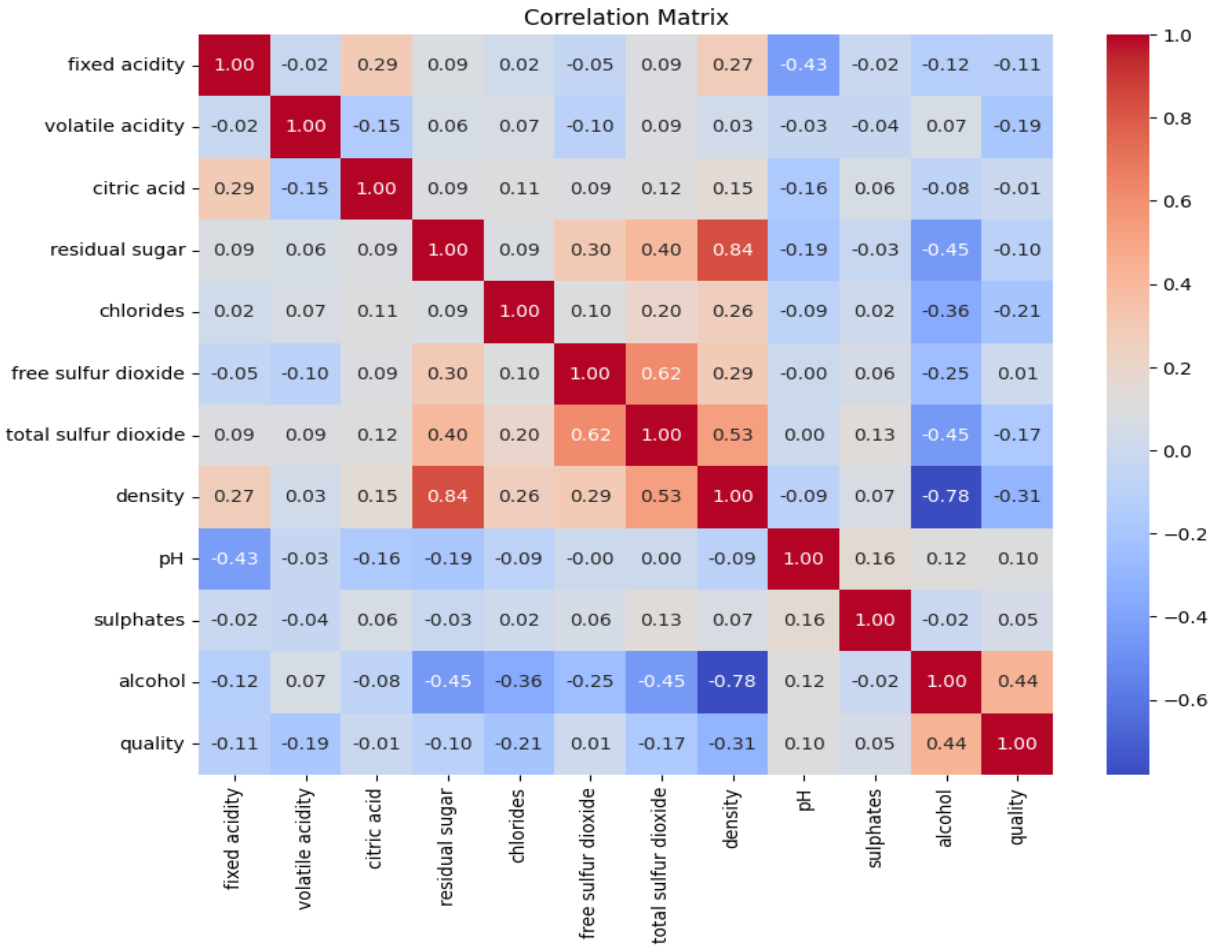
Distribution plots for key features

Distribution plots highlight clear patterns in the most important physicochemical properties of white wine. For fixed acidity, volatile acidity, and citric acid, there are more cases with lower values and fewer with higher values, showing a distribution that is taller on the left than on the right. While most wines have little residual sugar, some samples end up with much higher sugar content. Chlorides and free sulfur dioxide exhibit a downward skewness. Density and pH data are found to be normally distributed. Moderate right skew in sulphates and alcohol shows that the vast majority of wines are in the middle ranges. However, a few outliers reach higher levels. They reveal the important role played by the different chemicals in making wine.



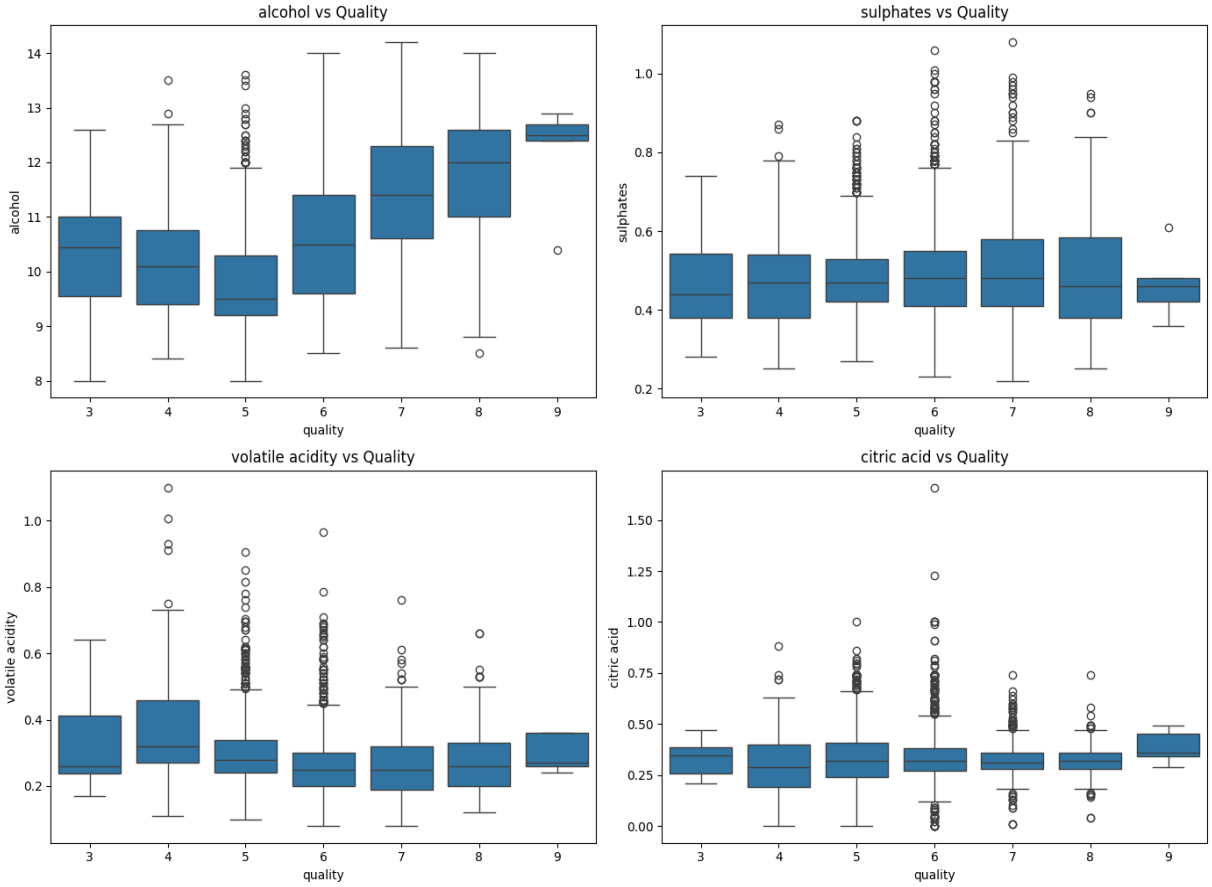
Correlation matrix and key insights

The correlation matrix demonstrates the relationship between different wine properties and quality. A strong and positive relationship (0.44) exists between higher alcohol content and increased perceived quality. Denser wines and those containing more sugar appeared to score slightly lower on the quality scale. Higher levels of volatile acidity than normal (-0.19) connect with lower wine quality ratings, confirming that acidity may make wine less appealing. Similarly, my data showed that individual factors have only limited links with quality, implying that several variables play a role in shaping wine quality.



Boxplots comparing quality with important features

The boxplots reveal that multiple chemical measurements are linked to the quality of the wine. The best wines usually contain higher amounts of alcohol than lesser qualities. Sulphate levels often go up a bit as quality goes up, though the variation is wider. Reducing volatile acidity is important for a higher wine grade, so lower acidity is better in fine wines. Citric acid appears to have limited variation according to quality, which suggests it matters less than the other compounds. Generally, these plots make it clear that alcohol, sulphates, and volatile acidity play a key role in determining wine quality.



Model Selection

The project involved investigating and solving problems of the types of regression and classification. In regression, we tried to estimate the wine quality, while in classification, we marked wines as Good or Bad if their score was higher or lower than 7. The regression approach and the tested models for predicting quality values are at the center of this section.

Three models were tested: Linear Regression, Random Forest Regressor, and XGBoost Regressor. The fact that Linear Regression is simple and easy to explain is why it is considered the first model. However, it considers that features and the target are linked linearly, though this may not be sufficient for wine quality data. Because the Random Forest Regressor is an ensemble of trees, it handles nonlinear problems and multiple interactions and tends to improve accuracy. Another important type of ensemble method, XGBoost, applies gradient boosting and usually performs well on structured information due to its strong optimization and regularization.

All models were trained on the same set of standardized features so that fair comparisons and better averaging could be made, which matters most for linear and gradient-based models. When cross-validation was possible, the code performed tuning of the hyperparameters. Baseline results for Random Forest and XGBoost were established using the default parameters, and we may expand tuning using grid search later on. You hardly need to configure Linear Regression.

Evaluation of the models took into account three regression indicators: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2). They measure how accurate the predictions are and the number of changes in the dataset that the model can explain. Out of all models examined, Random Forest was the most effective with a low RMSE (0.590), a low MAE (0.419), and a good R^2 (0.551) and was just ahead of XGBoost. Results from Linear Regression supported the idea that nonlinear techniques work better with the dataset.

Regression Model Performance

Model	**RMSE**	**MAE**	**R^2**
Linear Regression	0.754	0.586	0.265
Random Forest	0.590	0.419	0.551
XGBoost	0.617	0.439	0.509

Classification Model Performance

The classification task focused on comparing "Good" wines (quality ≥ 7) with "Bad" wines using logistic regression, Random Forest, and XGBoost classifiers. The highest accuracy (0.893), precision (0.859), recall (0.643), and F1-score (0.736) were achieved by Random Forest. XGBoost appeared next and recorded 0.883 average accuracy and 0.724 F1-score, but it did better than average at finding positive cases. Logistic regression fell behind in recall (0.282) and F1-score (0.380), suggesting that its results were strongly affected by the class imbalance. They suggest that combining different models is effective in handling this classifying task.

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Logistic Regression	0.787	0.582	0.282	0.380
Random Forest	0.893	0.859	0.643	0.736
XGBoost	0.883	0.795	0.665	0.724

Confusion Matrix

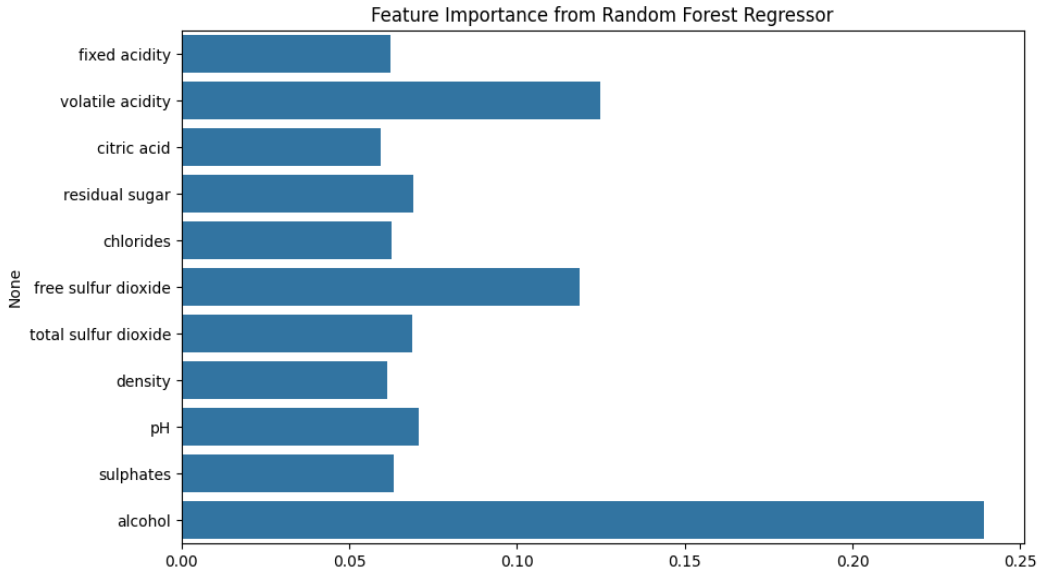
The confusion matrix for the Random Forest classifier shows that out of the total samples, 729 wines were correctly classified as "Bad" (true negatives), and 146 wines were

correctly identified as "Good" (true positives). However, 24 "Bad" wines were misclassified as "Good" (false positives), and 81 "Good" wines were misclassified as "Bad" (false negatives). This indicates that the model performs well overall but misses some high-quality wines, highlighting a trade-off between precision and recall.

	Predicted Bad	**Predicted Good**
Actual Bad	729	24
Actual Good	81	146

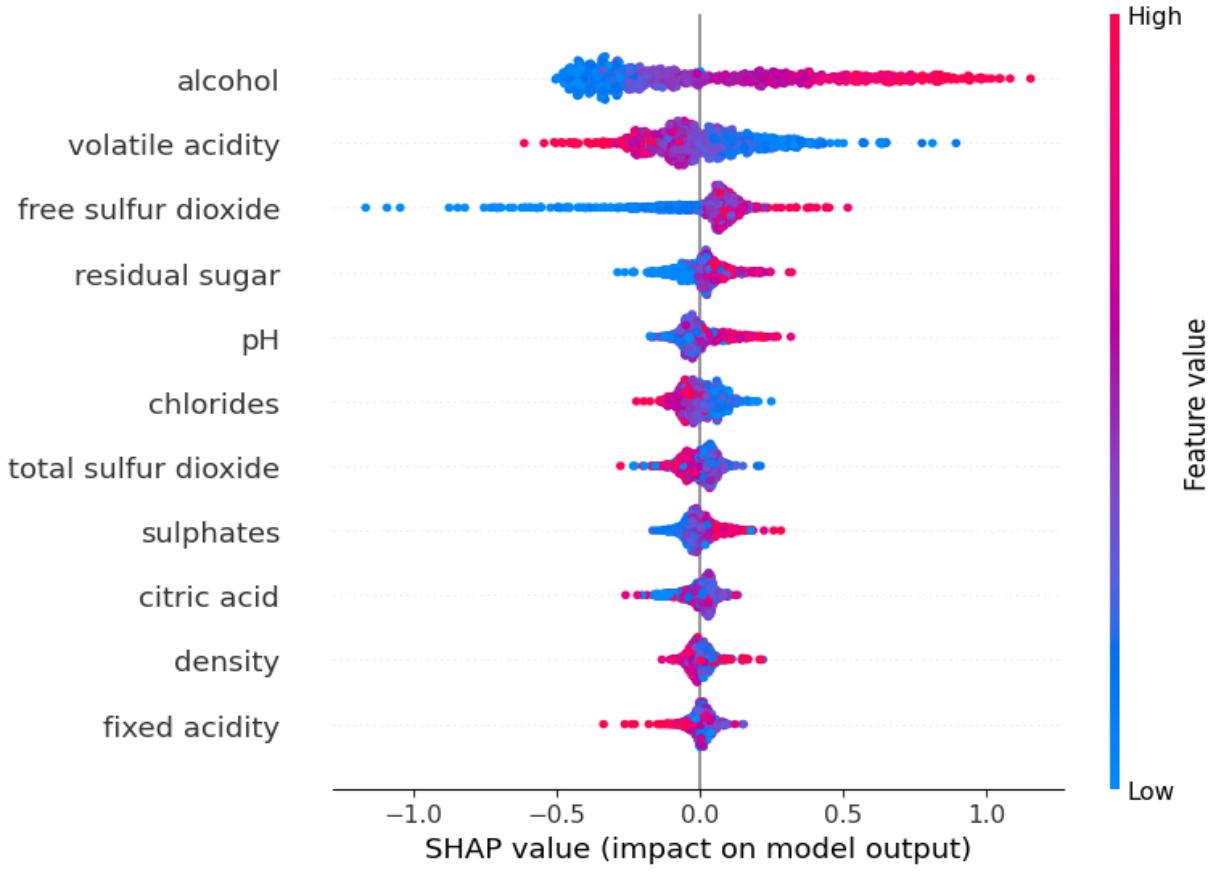
Feature importance analysis

Feature importance plot illustrates how each physicochemical property affects the prediction of wine quality. Alcohol has a much bigger impact than any other factor in this analysis. This agrees with research that shows a good connection between alcohol concentration and the taste of a wine. The following most relevant features are free sulfur dioxide and volatile acidity due to their strong role in quality predictions. The role of sulphates and chlorides is small, whereas residual sugar and pH are medium factors. The findings indicate that several key chemicals determine wine quality, not just one element.



Summary plot

The SHAP summary plot illustrates how every individual feature supports or influences the model's judgments on wine quality. The most significant benefit comes from alcohol, which means higher levels raise the predictions for better quality. Volatile acidity affects quality in a bad way, and elevated acidity leads to a lower predicted score. Sulfur dioxide and residual sugar sometimes increase and sometimes decrease how accurately we can make predictions from the data. Less severe but still obvious impacts can be seen when focusing on pH, chlorides, and sulphates. This gradient makes it clear that extreme values in the data can result in significant changes in the model's output because of the many connections between variables.



Conclusion and Recommendations

The research focused on using machine learning to foresee the quality of white wines using their biophysical properties. When the wine was first examined, significant associations with quality were found for alcohol content, volatile acidity, and sulphates. According to the data, better quality wines were associated with higher alcohol content, while higher volatile acidity was seen to lower quality scores. Multiple models were tested for both regression and classification, and it was found that Random Forest and XGBoost worked better than the simpler models, Linear and Logistic Regression.

Of the regression models, Random Forest scored the highest in both accuracy and stability, showing an RMSE of 0.59 and explaining nearly 60% of quality differences among samples. To separate high-quality wines from the others, Random Forest did exceptionally well, with accuracy approaching 90% and a reliable F1 score. From these results, I conclude that these models can effectively handle the complicated and nonlinear patterns found in wine quality information.

To improve further, researchers might try out more hyperparameter settings and also test Support Vector Machines or neural networks. Using both this data and sensory or expert review data could help improve the tool's prediction rate. Techniques such as adjusting features or dropping unnecessary ones can improve the performance and clarity of the model. The results show that machine learning can effectively predict wine quality by using data. It serves winemakers by helping them manage quality and consumers by ensuring they know what they are purchasing. With additional work in this field, we are poised to uncover more about how the qualities of wine link to its quality.

References

Bhardwaj, P., Tiwari, P., Olejar Jr, K., Parr, W., & Kulasiri, D. (2022). A machine learning application in wine quality prediction. *Machine Learning with Applications*, *8*, 100261. https://www.sciencedirect.com/science/article/pii/S266682702200007X

Jain, K., Kaushik, K., Gupta, S. K., Mahajan, S., & Kadry, S. (2023). Machine learning-based predictive modelling for the enhancement of wine quality. *Scientific Reports*, *13*(1), 17042. https://www.nature.com/articles/s41598-023-44111-9

Singla, M., Gill, K. S., Upadhyay, D., & Singh, V. (2024, March). Exploratory Data Analysis for Red Wine Quality Prediction Using a Decision Tree Approach and Machine Learning Methods. In *2024 3rd International Conference for Innovation in Technology (INOCON)* (pp. 1-5). IEEE. https://ieeexplore.ieee.org/abstract/document/10511597/

Yavas, C. E., Kim, J., Chen, L., Kadlec, C., & Ji, Y. (2025). Exploring Predictive Modeling for Food Quality Enhancement: A Case Study on Wine. *Big Data and Cognitive Computing*, *9*(3), 55. https://search.proquest.com/openview/b1b2835dcbec953cd523e86aebc40f29/1?pq-origsite=gscholar&cbl=2061777

Appendix: Python based Jupyter Notebook Code Details

GitHub Repo url: https://github.com/ramindersinghusd/m7

Predicting Wine Quality Using Machine Learning: An Exploratory Study and Model Comparison

- Group: 4
- Names: Hemlatha Kaur Saran, George David Asirvatharaj, Raminder Singh

- Module: Probability and Statistics for Artificial Intelligence
- Date: 20/06/2025

Local Development Env. Setup

1. Install python: 3.11.3
2. Installed VSCode
3. Add Python and jupyter extension
4. Set kernel
5. conda install -n base ipykernel jupyter
6. conda -V >> conda 23.5.2
7. pip install jupyter notebook pandas numpy matplotlib scipy scikit-learn pandoc nbconvert[webpdf] nbconvert notebook-as-pdf seaborn xgboost shap openpyxl
8. run >> jupyter notebook
9. Github url for code repo: https://github.com/ramindersinghusd/m7

In [1]:
```
# Import necessary libraries for data analysis, visualization, and machine learning
import pandas as pd  # For data manipulation and analysis
import numpy as np  # For numerical operations
import matplotlib.pyplot as plt  # For plotting graphs
import seaborn as sns  # For advanced data visualization
from sklearn.model_selection import train_test_split, GridSearchCV  # For splitting
from sklearn.preprocessing import StandardScaler  # For feature scaling
from sklearn.linear_model import LinearRegression, LogisticRegression  # For regres
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier  # For e
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score  # Fo
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from xgboost import XGBRegressor, XGBClassifier  # For XGBoost models
import shap  # For model interpretability
```

In [2]:
```
# Load the wine quality dataset from an Excel file
# The dataset contains physicochemical properties and quality ratings for white win
data = pd.read_excel('winequality-white123.xlsx')
```

In [3]:
```
# Display the first few rows of the dataset to understand its structure
print(data.head())

# 1. Data Cleaning/Preparation
# Check for missing values in the dataset to ensure data quality
print("\nChecking for missing values:")
print(data.isnull().sum())  # Confirm no missing data

# Separate features (X) and target variable (y)
```

```
X = data.drop('quality', axis=1)  # Features: all columns except 'quality'
y = data['quality']  # Target: wine quality rating
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.0              0.27         0.36            20.7      0.045
1            6.3              0.30         0.34             1.6      0.049
2            8.1              0.28         0.40             6.9      0.050
3            7.2              0.23         0.32             8.5      0.058
4            7.2              0.23         0.32             8.5      0.058

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 45.0                 170.0   1.0010  3.00       0.45
1                 14.0                 132.0   0.9940  3.30       0.49
2                 30.0                  97.0   0.9951  3.26       0.44
3                 47.0                 186.0   0.9956  3.19       0.40
4                 47.0                 186.0   0.9956  3.19       0.40

   alcohol  quality
0      8.8        6
1      9.5        6
2     10.1        6
3      9.9        6
4      9.9        6

Checking for missing values:
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
```

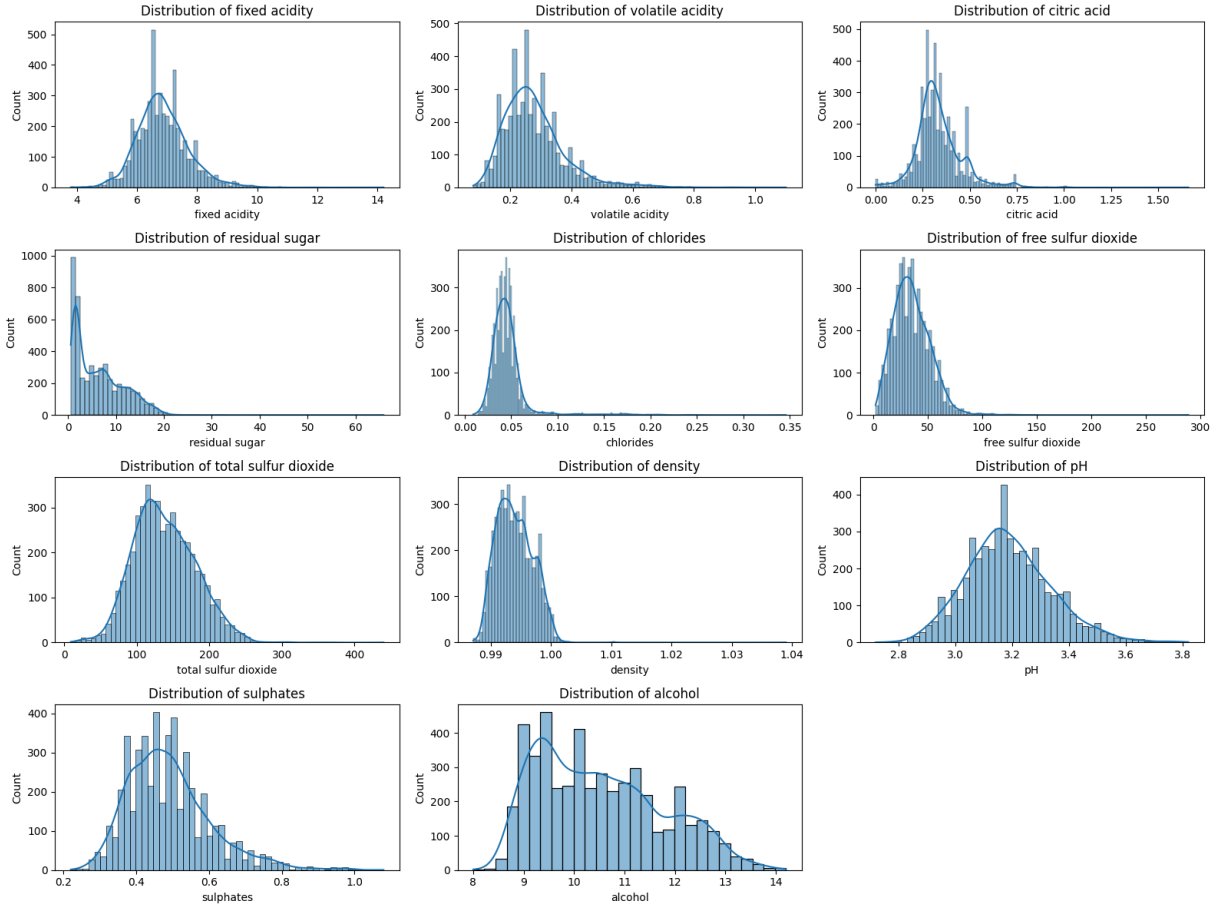
In [4]:

```
# Option 1: Regression (predict actual quality)
# Option 2: Classification (Good vs Bad wine based on threshold quality >= 7)

# Define a binary classification target: Good (quality >= 7) vs Bad (quality < 7)
quality_threshold = 7
y_class = (y >= quality_threshold).astype(int)  # 1 = Good, 0 = Bad

# Feature Scaling: Standardize features to have mean=0 and variance=1
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and test sets for regression
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_scaled, y, te

# Split data into training and test sets for classification
X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(X_scaled, y_cla
```

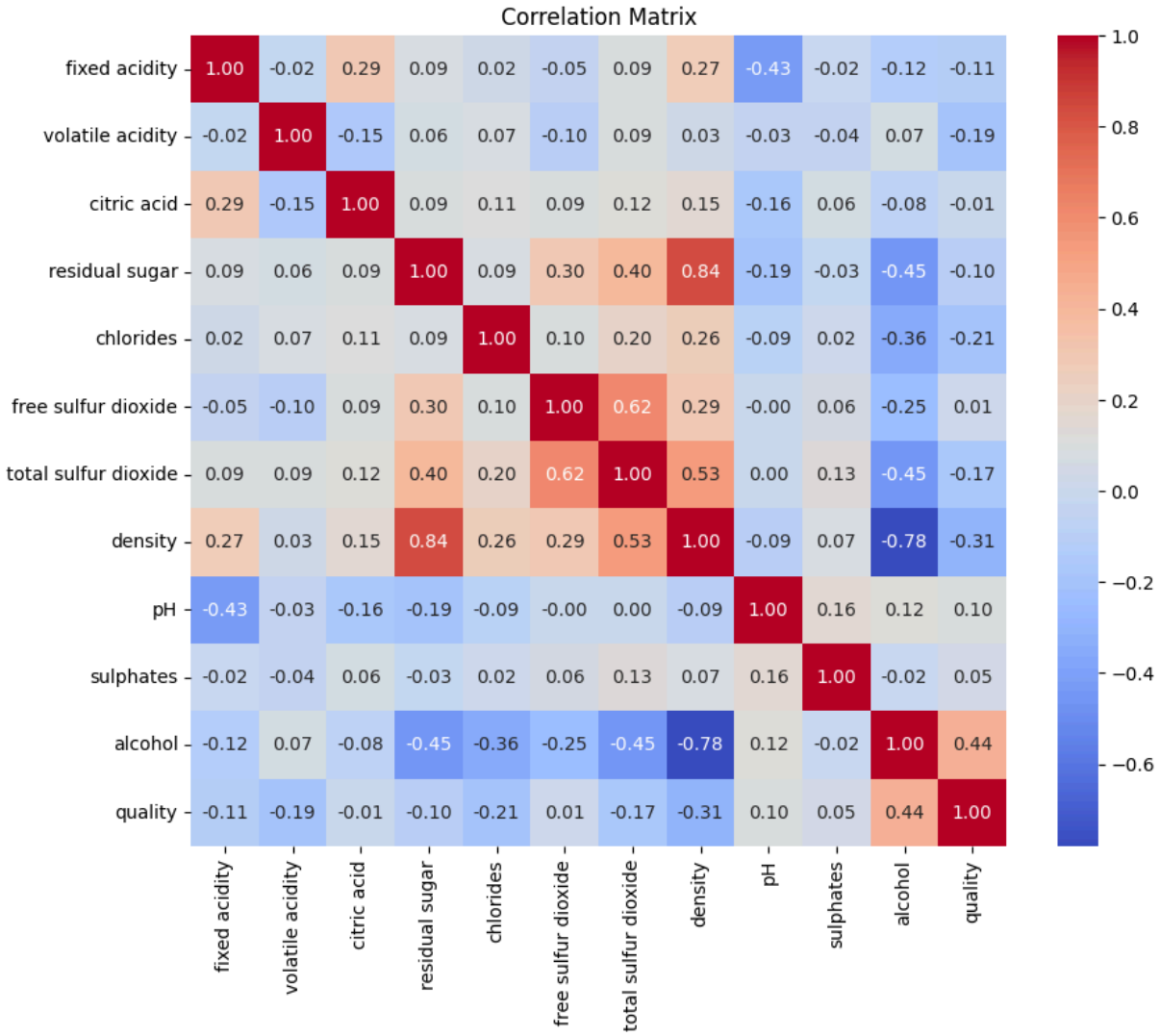
In [5]:
```
# 2. Exploratory Data Analysis (EDA)
# Display summary statistics for each feature to understand data distribution and c
print("\nSummary statistics:")
print(data.describe())
```

```
Summary statistics:
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    4898.000000       4898.000000  4898.000000     4898.000000
mean        6.854788          0.278241     0.334192        6.391415
std         0.843868          0.100795     0.121020        5.072058
min         3.800000          0.080000     0.000000        0.600000
25%         6.300000          0.210000     0.270000        1.700000
50%         6.800000          0.260000     0.320000        5.200000
75%         7.300000          0.320000     0.390000        9.900000
max        14.200000          1.100000     1.660000       65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  4898.000000          4898.000000           4898.000000  4898.000000
mean      0.045772            35.308085            138.360657     0.994027
std       0.021848            17.007137             42.498065     0.002991
min       0.009000             2.000000              9.000000     0.987110
25%       0.036000            23.000000            108.000000     0.991723
50%       0.043000            34.000000            134.000000     0.993740
75%       0.050000            46.000000            167.000000     0.996100
max       0.346000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol      quality
count  4898.000000  4898.000000  4898.000000  4898.000000
mean      3.188267     0.489847    10.514267     5.877909
std       0.151001     0.114126     1.230621     0.885639
min       2.720000     0.220000     8.000000     3.000000
25%       3.090000     0.410000     9.500000     5.000000
50%       3.180000     0.470000    10.400000     6.000000
75%       3.280000     0.550000    11.400000     6.000000
max       3.820000     1.080000    14.200000     9.000000
```

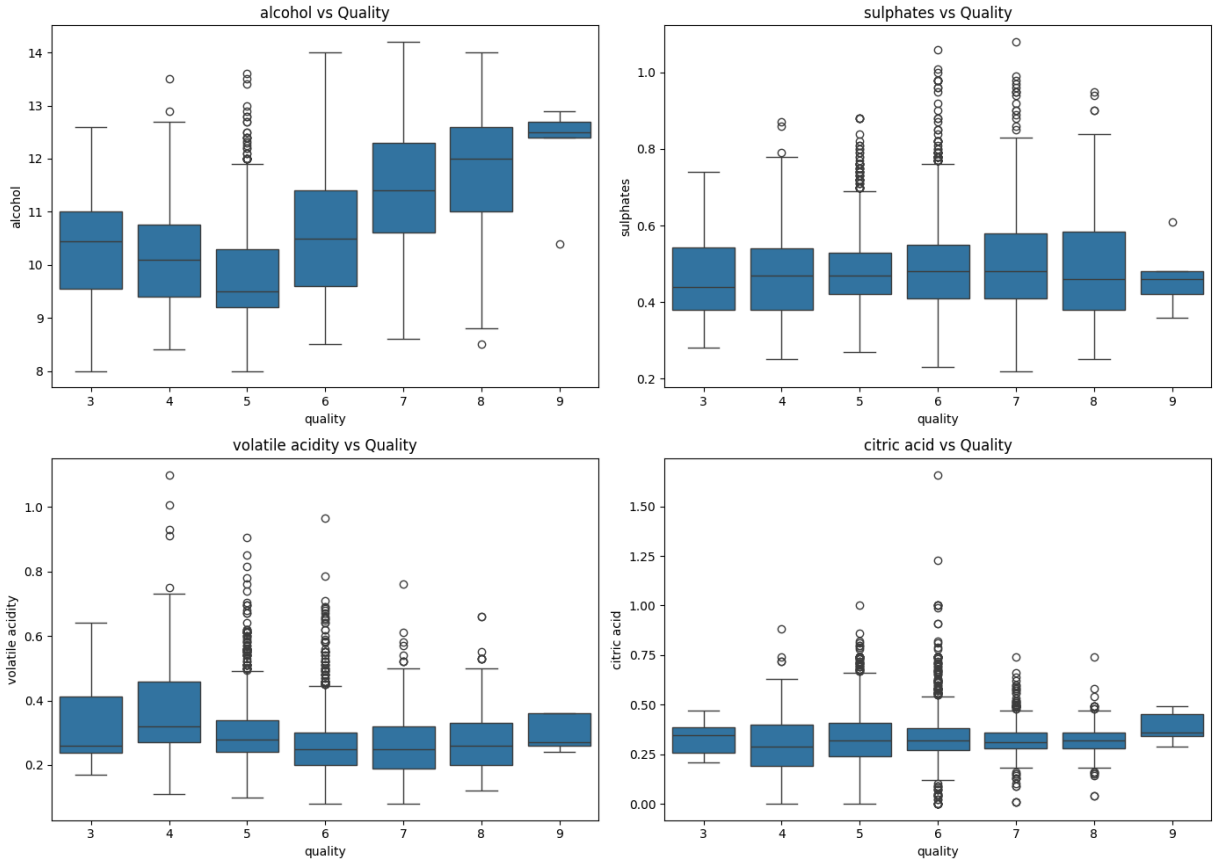
In [6]:
```
# Plot the distribution of each feature to visualize their spread and detect outlie
features = X.columns
plt.figure(figsize=(16,12))
for i, feature in enumerate(features):
    plt.subplot(4,3,i+1)
    sns.histplot(data[feature], kde=True)
    plt.title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()
```



In [7]:
```
# Plot the correlation matrix to examine relationships between features and with th
plt.figure(figsize=(10,8))
corr = data.corr()
sns.heatmap(corr, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



In [8]:
```
# Create boxplots to visualize the relationship between wine quality and key featur
key_features = ['alcohol', 'sulphates', 'volatile acidity', 'citric acid']
plt.figure(figsize=(14,10))
for i, feature in enumerate(key_features):
    plt.subplot(2,2,i+1)
    sns.boxplot(x='quality', y=feature, data=data)
    plt.title(f'{feature} vs Quality')
plt.tight_layout()
plt.show()
```



In [9]:
```
# 3. Model Selection and Training
# Define Regression models to predict wine quality as a continuous variable
models_reg = {
    "Linear Regression": LinearRegression(),
    "Random Forest": RandomForestRegressor(random_state=42),
    "XGBoost": XGBRegressor(random_state=42, eval_metric='rmse')
}
```

In [10]:
```
# Train and evaluate each regression model, reporting key performance metrics
print("\nRegression Model Performance:")
for name, model in models_reg.items():
    model.fit(X_train_reg, y_train_reg)  # Train the model
    y_pred = model.predict(X_test_reg)  # Predict on test set
    rmse = np.sqrt(mean_squared_error(y_test_reg, y_pred))  # Root Mean Squared Err
    mae = mean_absolute_error(y_test_reg, y_pred)  # Mean Absolute Error
    r2 = r2_score(y_test_reg, y_pred)  # R-squared score
    print(f"{name} - RMSE: {rmse:.3f}, MAE: {mae:.3f}, R²: {r2:.3f}")
```

```
Regression Model Performance:
Linear Regression - RMSE: 0.754, MAE: 0.586, R²: 0.265
Random Forest - RMSE: 0.590, MAE: 0.419, R²: 0.551
XGBoost - RMSE: 0.617, MAE: 0.439, R²: 0.509
```

In [11]:
```
# Define classification models to predict if a wine is 'Good' or 'Bad'
models_clf = {
    "Logistic Regression": LogisticRegression(max_iter=500, random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    #"XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='logloss', rando
```

```
    "XGBoost": XGBClassifier(eval_metric='logloss', random_state=42)
}
```

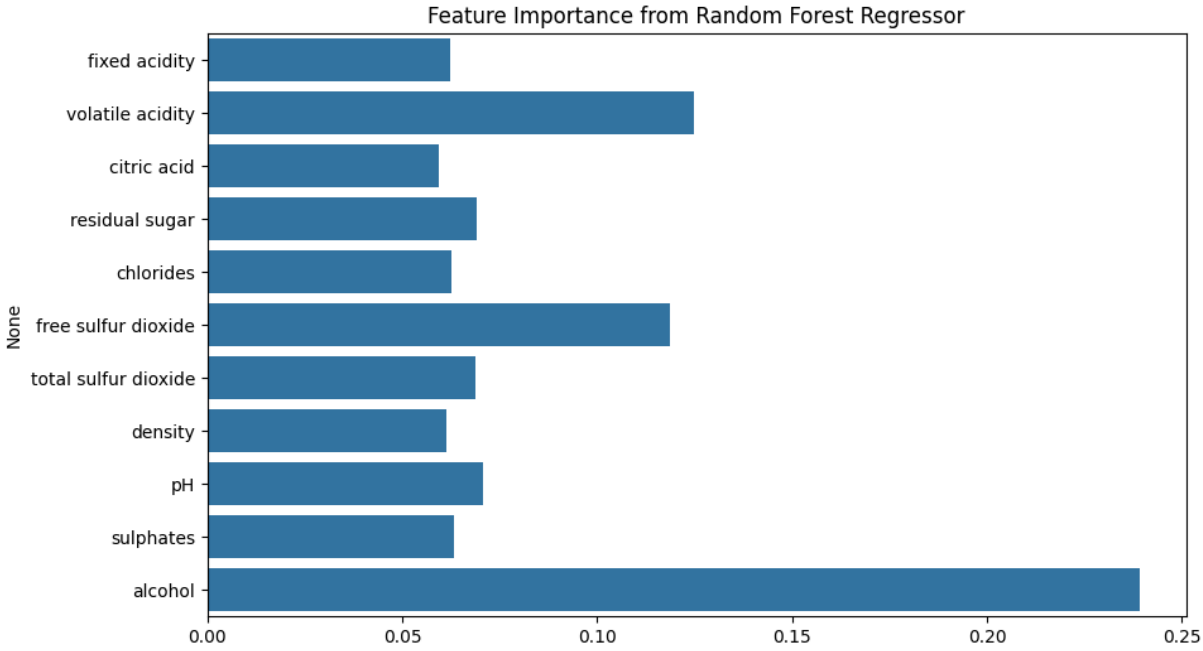
In [12]:
```
# Train and evaluate each classification model, reporting accuracy, precision, reca
print("\nClassification Model Performance:")
for name, model in models_clf.items():
    model.fit(X_train_clf, y_train_clf)  # Train the model
    y_pred = model.predict(X_test_clf)  # Predict on test set
    acc = accuracy_score(y_test_clf, y_pred)  # Accuracy
    prec = precision_score(y_test_clf, y_pred)  # Precision
    rec = recall_score(y_test_clf, y_pred)  # Recall
    f1 = f1_score(y_test_clf, y_pred)  # F1-score
    print(f"{name} - Accuracy: {acc:.3f}, Precision: {prec:.3f}, Recall: {rec:.3f},
```

```
Classification Model Performance:
Logistic Regression - Accuracy: 0.787, Precision: 0.582, Recall: 0.282, F1-score: 0.
380
Random Forest - Accuracy: 0.893, Precision: 0.859, Recall: 0.643, F1-score: 0.736
XGBoost - Accuracy: 0.883, Precision: 0.792, Recall: 0.670, F1-score: 0.726
```

In [13]:
```
# Generate and display the confusion matrix for the best classifier (Random Forest)
best_clf = RandomForestClassifier(random_state=42)
best_clf.fit(X_train_clf, y_train_clf)
y_pred_best = best_clf.predict(X_test_clf)
cm = confusion_matrix(y_test_clf, y_pred_best)
print("\nConfusion Matrix for Random Forest Classifier:")
print(cm)
```

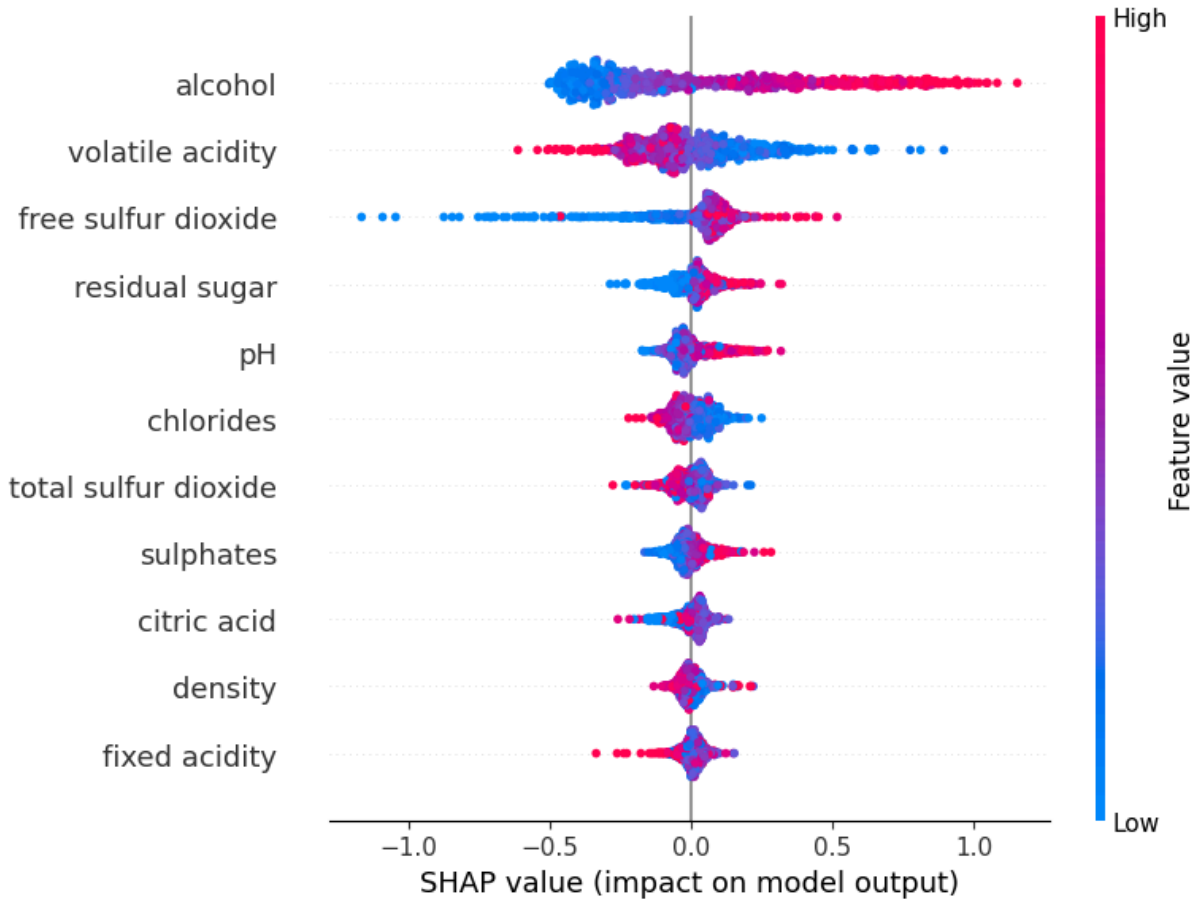
```
Confusion Matrix for Random Forest Classifier:
[[729  24]
 [ 81 146]]
```

In [14]:
```
# 4. Feature Importance (Random Forest Regressor example)
# Fit a Random Forest Regressor and extract feature importances
best_reg = RandomForestRegressor(random_state=42)
best_reg.fit(X_train_reg, y_train_reg)
importances = best_reg.feature_importances_

# Visualize feature importances using a bar plot
plt.figure(figsize=(10,6))
sns.barplot(x=importances, y=features)
plt.title('Feature Importance from Random Forest Regressor')
plt.show()
```



In [15]:
```
# 5. Use SHAP to explain the predictions of the Random Forest Regressor
explainer = shap.TreeExplainer(best_reg)
shap_values = explainer.shap_values(X_test_reg)
```

In [16]:
```
# Generate a SHAP summary plot to visualize feature impact on model output
shap.summary_plot(shap_values, features=X_test_reg, feature_names=features)
```



Conclusion

1. Feature Impact: The analysis revealed that certain features, such as alcohol content and volatile acidity, have a significant impact on wine quality. Both feature importance and SHAP analysis consistently highlighted these variables as key drivers in predicting wine quality.

2. Model Performance: Ensemble models like Random Forest and XGBoost outperformed simpler models in both regression and classification tasks, demonstrating the value of using advanced machine learning techniques for complex, real-world datasets like wine quality prediction.