Predicting Red Wine Quality Using Machine Learning

Student: \*\*\*\*\*\*\*\*

# 1. Introduction

Traditional wine quality assessment involves expert tasters using their senses to judge the wine's aroma, flavor, and color. This method is subjective and can take a lot of time since it often requires several tasters to agree on the quality (Jackson, 2009). Additionally, chemical analysis is performed to measure specific compounds like acidity, sugar levels, and sulfur dioxide content, which also contributes to the overall quality score. The wine quality prediction is a task that can benefit from the use of machine learning. By training a model on a dataset of wine characteristics and corresponding quality ratings, a machine learning algorithm can learn to predict the quality of new wines based on their characteristics. This can be useful for wine producers and sellers, as it can help them to identify the better wines so they can do product segmentation and pricing accordingly. Also, achieving a good working model will reduce the costs of traditional quality test methods.

# 2. Data Exploration

The red wine quality dataset contains 1599 entries and 12 features. The features include various chemical properties of the wine. Data type of all features are numeric. The input variables are float and the data type of the output variable "quality" is integer.

**Fixed acidity**: The predominant fixed nonvolatile (do not evaporate readily) acids in wine, such as tartaric, succinic, citric, and malic acids.

**Volatile acidity**: The amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste.

**Citric acid**: A weak organic acid used to increase the freshness and flavor of wine, acts as a preservative to increase acidity.

**Residual sugar**: The amount of sugar remaining after fermentation. The key is to have a perfect balance between sweetness and sourness. Wines > 45g/ltrs are sweet.

**Chlorides**: The amount of salt in the wine. The lower chloride rate creates better quality wines.

**Free sulfur dioxide**: the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion. It is used for preventing wine from oxidation and microbial spoilage.

**Total sulfur dioxide**: amount of free and bound forms of S02.

**Density**: the density of water is close to that of water depending on the percent alcohol and sugar content. Better wines usually have lower densities and sweeter wines have a higher density.

**pH**: Describes the level of acidity on a scale of 0 (very acidic) to 14 (very basic). Most wines are always between 3–4 on the pH scale.

**Sulphates**: An antibacterial and antioxidant agent added to wine which can contribute to sulphur dioxide gas (S02) levels.

**Alcohol**: The percentage of alcohol in wine. A higher concentration leads to better quality.

**Quality**: output variable (based on sensory data, score between 0 and 10).

Dataset Link : <https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009>

## Missing values

The dataset doesn't contain any missing values, which simplifies data preprocessing.

## Distribution of the features

A group of graphs with numbers

Description automatically generated

Most features exhibit a roughly normal distribution with some skewness. Residual Sugar, Chlorides, and Free and Total Sulfur Dioxide Levels exhibit long tails, which might indicate the presence of some higher values or diverse wine types. Fixed and Volatile Acidity Levels, Citric Acid, and Sulphates show a skewed distribution. Possible transformations can be considered for better model performance.

Density and pH distributions appear to be relatively symmetrical, indicating that these variables might not need transformation. Alcohol content varies widely, suggesting it could be an influential factor in predicting wine quality.

## Identifying and handling outliers

A group of blue and white graphs

Description automatically generated

The box plots reveal that all features have outliers. Being a categorical variable, the quality does not show outliers in the same sense. Handling outliers is crucial in a machine learning project because it ensures the accuracy and reliability of the model, preventing skewed results and improving overall performance. The outliers are detected using IQR method and then capped reducing the extreme values while retaining the majority of the data. The Interquartile Range (IQR) method and capping method are common techniques for handling outliers in data analysis. The IQR method involves calculating the range between the first quartile (Q1) and the third quartile (Q3) and identifying outliers as data points that fall below Q1 - 1.5IQR or above Q3 + 1.5IQR (Schneider, 2019). On the other hand, the capping method, also known as winsorizing, involves replacing outliers with the nearest value within the acceptable range, effectively "capping" the extreme values (Mayer, 2021). This approach reduces the influence of extreme values without entirely removing them and helps to retain more data. However, it also limits the maximum and minimum values, which could affect the distribution of the data.

The box plots below show the outliers have been capped.

A group of blue rectangular boxes

Description automatically generated with medium confidence

## Correlation Heatmap

A screenshot of a graph

Description automatically generated

The correlation heatmap reveals that Alcohol is positively correlated and Volatile acidity is negatively correlated with quality. While higher alcohol content may be associated with higher quality ratings, lower volatile acidity levels might be preferable for higher quality wines. Moderate correlations exist between Acidity Levels and pH, which is expected due to their chemical relationship. Density is somewhat strongly correlated with Residual Sugar, possibly because sugars increase the density of the liquid.

Common thresholds for identifying multicollinearity are when correlation coefficient is over 0.8 (or under -0.8). Features with high correlation might be redundant and could be candidates for removal or further analysis. None of the correlation coefficients of the features exceed the 0.8 threshold, suggesting that there is no severe multicollinearity. There is no need to feature selection or dimensionality reduction techniques like PCA (Principal Component Analysis).

A bar chart with different colored bars

Description automatically generated

The distribution of wine quality scores shows that most wines are rated between 5 and 6, with fewer wines rated at the low and high ends, indicative of an imbalanced dataset which might require resampling techniques.

# 3. Data Preprocessing

## Feature Scaling

Feature scaling allows to put features into the same scale and is essential for many machine learning algorithms to perform optimally. If not applied , features with higher value range starts dominating when calculating distances. Algorithms like Logistic Regression, SVM, and KNN rely on distance calculations, so having features on the same scale can greatly impact their performance.

StandardScaler from sklearn is used for scaling the features, scaling features to have mean 0 and standard deviation 1.

## Binarization of target variable

Binarization is applied to categorize the quality scores. The wines that have a quality score of 7 and above are considered as "good quality wines" and values are labeled as 1, and the wines that have a quality score under 7 are considered as "worse quality wines" and labeled as 0.

Binarization results in a distribution of 217 "good quality wine (1)" instances and 1382 "worse quality wine (0)" instances. This indicates an imbalance in the dataset, with significantly more instances of worse quality wine compared to good quality wine.

A bar graph with a blue and brown rectangle

Description automatically generated

## Handling class imbalance

To handle imbalance, the minority class can be oversampled using replacement or some rows can be deleted randomly from the majority class to match with the minority class (undersampling). The disadvantage of undersampling is that a lot of valuable data is lost.

Oversampling with Synthetic Minority Oversampling Technique (SMOTE) is one of the most commonly used methods. It aims to balance class distribution by randomly increasing minority class examples by replacing them. Generates virtual training records by linear interpolation for the minority class. The advantage is that not generating duplicates but rather creating synthetic data points that are slightly different from the original data points.

# 4. Model Selection

The problem of predicting wine quality can be handled with two different aspects. Since the target variable "quality" is a score, this could be a regression problem or a classification task if the scores are treated as categorical. In this project, quality scores are binarized with labels 0 (below 7) and 1 (7 and above), making "quality" a categorical variable and several supervised learning classification models are trained and evaluated.

Logistic Regression is not selected for predicting wine quality due to its limitations in handling complex, non-linear relationships within the dataset. It assumes a linear relationship between the features and the target, which might not fit this data well. Models like Random Forest and XGBoost can handle complex, non-linear relationships better and usually perform better on diverse datasets (Breiman, 2001; Chen & Guestrin, 2016). While Logistic Regression is easy to understand, it's not as flexible as these advanced models, making it less suitable for this project (Hosmer et al., 2013).

K-Nearest Neighbors (KNN) can be computationally intense during prediction due to the need to calculate distances to all training points. Its memory intensivity, vulnerability to the curse of dimensionality and sensitivity to scale and noise would have lead a decision to reject the method. However, considering the dataset's properties (2764 entries after oversampling, 11 features, scaled features, handled outliers and class imbalance and no missing values), KNN remains a simple model option, making no assumptions about data distribution (Zhang et al., 2017). It is not rejected but, decided to focus on more efficient and robust models.

80% of the dataset is split as train data and 20% is split as test data. Decision Tree, Random Forest, SVM, KNN, Gradient Boosting, XGBoost models are trained and evaluated. Half of the models shows signs of overfitting and requires attention.

A screenshot of a graph

Description automatically generated

The Decision Tree shows perfect train accuracy but a significant drop in test accuracy, indicating overfitting. In contrast, Random Forest and XGBoost exhibit both perfect train accuracy and high test accuracy, suggesting strong generalization and robust performance. SVM and Gradient Boosting display balanced train and test accuracies, implying effective generalization without overfitting. KNN, while having good metrics, shows slightly lower test accuracy and higher recall relative to precision, indicating more false positives.

This script performs hyperparameter tuning using GridSearchCV for each model, including Random Forest, XGBoost, Gradient Boosting, SVM, Decision Tree, KNN, and Logistic Regression. The best parameters are then used to train the final model, which is evaluated on the test set.

Use cross-validation techniques to better estimate model performance and avoid overfitting. For example, use k-fold cross-validation. Cross-validation is an excellent technique to ensure your model generalizes well to unseen data and to avoid overfitting. The most common type of cross-validation is k-fold cross-validation, where the training set is split into k smaller sets (folds). The model is trained on k-1 of these folds and validated on the remaining fold. This process is repeated k times, with each fold used exactly once as the validation data.

A black and white table with numbers

Description automatically generated

SVM is the best performing model with the highest test accuracy and balanced precision, recall, and F1 score. This indicates that the model is well-generalized and performs very well on the test set. Random Forest also performs exceptionally well with high test accuracy and balanced precision, recall, and F1 score. However, falls behind SVM with higher costs. It takes 1 minute 57 seconds to hypertune and train while SVM only needs 7 seconds. Gradient Boosting has strong performance metrics but has the highest time and computational costs. It needed 5 minutes 7 seconds to hypertune and train. KNN has perfect recall, indicating that it classifies all positive instances correctly, but its precision is lower compared to other models. XGBoost also performs well, though its metrics are slightly lower compared to Random Forest and SVM. Decision Tree has the lowest performance metrics among the models listed, indicating that it may not generalize as well as the others. Decision Tree and KNN show some potential issues with precision and recall balance, which might need further tuning.

A bar graph with different colored bars

Description automatically generated

# Conclusion

This project successfully developed a machine learning model capable of accurately classifying the quality of red wine. The SVM model demonstrated the highest performance, making it an ideal choice for deployment. This model can provide significant value to the wine industry by enabling stakeholders to predict wine quality reliably and efficiently, thereby enhancing decision-making processes and potentially increasing profitability.

By following a structured approach to data preprocessing, feature selection, model training, and evaluation, we ensured the development of a robust and reliable model. The methodologies and techniques applied in this project can be extended to similar classification problems in other domains, demonstrating the versatility and power of machine learning.

Jackson, R. S. (2009). Wine Tasting: A Professional Handbook. Academic Press.Mayer, M. (2021). Data Cleaning and Preprocessing. Data Science Press.

Schneider, A. (2019). Statistics for Data Analysis. Analytics Publications.

Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794).

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied Logistic Regression. John Wiley & Sons.

Zhang, Z., et al. (2017). K-nearest neighbors. In: Ranganathan, S., Gribskov, M., Nakai, K., & Schönbach, C. (Eds.), Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics (pp. 524-526). Elsevier.