REPORT

From Chemistry to Quality:Predicting Wine Quality Using past customer ratings and wine chemical Properties

Introduction

The aim of this research is to predict the quality of wine based on past customer ratings and using the following wine chemical properties:

- *Fixed Acidity*: The stable acids present in wine, such as tartaric acid, which do not change significantly during fermentation; they contribute to the wine's structure and freshness.

- *Volatile Acidity:* The measure of acetic acid and other volatile acids in wine, which can evaporate easily; high levels may indicate spoilage and can affect the wine's aroma and taste.

- *Citric Acid:* An organic acid commonly found in citrus fruits; in wine, it adds a fresh, zesty flavour and can enhance the wine’s overall acidity.

- *Residual Suga*r: The amount of sugar left in the wine after fermentation, influencing its sweetness and balance; it can range from bone dry to very sweet.

- *Chlorides*: The level of chloride ions in wine, often derived from salts; it can affect the wine’s flavour and mouthfeel, contributing to a saline quality.

- *Free Sulphur Dioxide*: The portion of sulphur dioxide that is not bound to other compounds and is effective in protecting wine from oxidation and microbial spoilage.

- *Total Sulphur Dioxide:* The total amount of sulphur dioxide in the wine, including both free and bound forms; it is used as a preservative and antioxidant.

- *Density*: The mass of wine per unit volume, which can provide insights into its composition and richness, influencing the mouthfeel and texture.

- *pH*: A measure of the acidity or alkalinity of the wine, with a scale from 0 to 14; a lower pH indicates higher acidity, affecting flavour and stability.

- *Sulphates*: The concentration of sulphate ions in wine, which can enhance dryness and contribute to a crisp, refreshing finish.

- *Alcohol*: The percentage of ethanol in wine, affecting its body, flavour intensity, and overall mouthfeel; higher alcohol levels can impart warmth and richness.

- Quality: It’s the score between 0 and 10 given by wine drinkers.

The Dataset

The dataset was obtained from the UCI Machine Learning Repository.

The two datasets pertain to the red and white varieties of the Portuguese "Vinho Verde" wine, as referenced in [Cortez et al., 2009]. The red wine dataset contains a total of 1,599 instances, whereas the white wine dataset comprises 4,898 instances. Both datasets feature 12 columns, each based on physicochemical tests.

Loading the Dataset

An essential first step in our process is to load the datasets into Jupyter Notebook and harness the capabilities of Pandas for data manipulation. Both datasets are available in CSV format. After loading the datasets, I began my exploration by using the .info() and .shape methods.

Data Cleaning

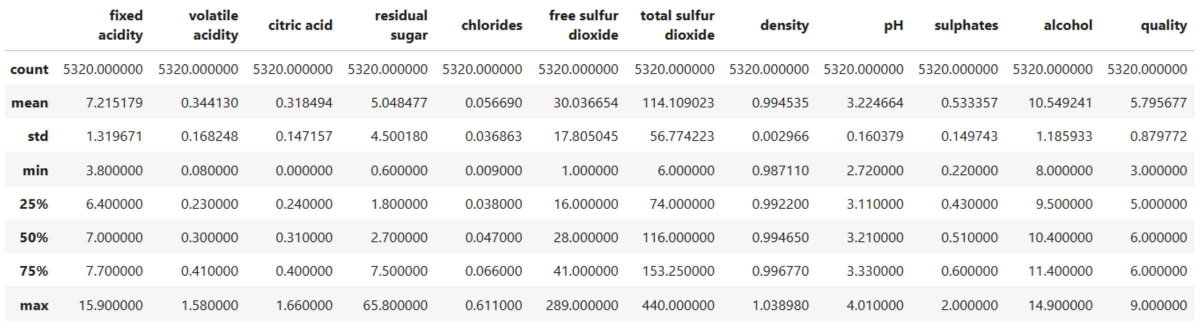
I examined the datasets to check for and address any missing values, discovering that both datasets have zero missing entries. Additionally, I searched for duplicates to identify and resolve any redundant entries, as duplicate rows can distort analysis and modelling results by disproportionately weighting repeated observations, which may lead to biased or inaccurate outcomes.
The duplicates were removed in order to have more reliable analysis and to improve model accuracy and prevent overfitting.

Merged the datasets

To perform the analysis and train machine learning models, I merged the red and white wine datasets. Before merging, I created a new column to indicate whether the wine is red or white. The two datasets were combined into a single one using the .concat method. Once the dataset was finalised, I saved it to a new CSV file.

Exploratory Data Analysis (EDA)

I first explored the dataset using descriptive statistics using .describe method.



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	5320.000000	5320.000000	5320.000000	5320.000000	5320.000000	5320.000000	5320.000000	5320.000000	5320.000000	5320.000000	5320.000000	5320.000000
mean	7.215179	0.344130	0.318494	5.048477	0.056690	30.036654	114.109023	0.994535	3.224664	0.533357	10.549241	5.795677
std	1.319671	0.168248	0.147157	4.500180	0.036863	17.805045	56.774223	0.002966	0.160379	0.149743	1.185933	0.879772
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000	3.000000
25%	6.400000	0.230000	0.240000	1.800000	0.038000	16.000000	74.000000	0.992200	3.110000	0.430000	9.500000	5.000000
50%	7.000000	0.300000	0.310000	2.700000	0.047000	28.000000	116.000000	0.994650	3.210000	0.510000	10.400000	6.000000
75%	7.700000	0.410000	0.400000	7.500000	0.066000	41.000000	153.250000	0.996770	3.330000	0.600000	11.400000	6.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000	9.000000

I then checked for outliers across all the numerical variables using a boxplot, and once I identified the outliers, I decided to remove them using the Z-score method. I used the Z-score to remove outliers because it quantifies how many standard deviations a data point is from the mean, allowing for a clear identification of extreme values. This method standardise the data, making it easier to compare across different datasets and ensuring that outliers can be systematically detected and addressed.

I subsequently examined the normal distribution of the numerical variables using a histogram with a density curve. By doing so, I gained a visual assessment of the data distribution, making it easier to see how closely the data conforms to a normal distribution. The density curve superimposed on the histogram helps to highlight deviations from normality, such as skewness or kurtosis, providing a clearer understanding of the data's characteristics. Furthermore, I used a statistical test called the Shapiro-Wilk test to further evaluate normality. Based on the plots and the results of the statistical test, it is evident that the numerical variables do not follow a normal distribution.

Using a countplot, I evaluated the distribution of the ratings. I then executed a correlation matrix of all numerical variables to summarise the relationships between multiple variables, providing a clear view of how each variable correlates with the others. This helps identify patterns, trends, and potential multicollinearity, guiding further analysis and model selection. It is important to note that, since our data were not normally distributed, I used Spearman's rank correlation coefficient, a non-parametric measure that is particularly useful when dealing with ordinal data or when the data contains outliers that could skew results in traditional Pearson correlation.

According to the correlation matrix, alcohol and quality have the strongest positive relationship, suggesting that higher alcohol content tends to be associated with better quality wines. Volatile acidity seems to negatively impact wine quality, although the relationship is not very strong. Density is inversely related to alcohol but positively related to residual sugar and chlorides. The correlation coefficient for density and alcohol is -0.68, indicating a strong negative correlation, meaning higher alcohol content is associated with lower density.

A regplot was used to visually depict the relationship between the numerical variables and the target variable (quality), providing insights into correlations and identifying which factors may significantly influence overall wine quality.

Data Transformation

<u>One-hot encoding</u>

I applied One-hot encoding to the ‘wine colour’ column, which is a column created previously in order to identify the wine type. It was decided to apply one-hot encoding because we are dealing with a categorical variable, and it is essential to transform it into a numerical format suitable for machine learning algorithms. This method creates binary columns for each category, allowing the model to interpret the presence or absence of each category without implying any ordinal relationship between them.

<u>Creating bins in the target variable</u>

I decided to divide the ‘quality’ variable into the following bins using .cut method.

- Low: 0 - 3
- Medium: 4 -7
- High: 8 - 10

This decision was taken in order to simplifies the data by grouping continuous ratings into discrete categories, which can help reduce noise and improve interpretability. This binning process can enhance the performance of machine learning classification models by making patterns more apparent, facilitating easier decision boundaries, and allowing the model to better handle non-linear relationships in the data.

Label encoding of the target variable

Using Label encoder, I encoded the quality variable, that was previously binned in low, medium, high. The label encoding effectively captures the order, which can be useful for machine learning models that can leverage this information.

Selecting relevant features

Based on the EDA, several features have been identified to be highly correlated with wine quality. These features are:

- 'citric acid'
- 'residual sugar'
- 'free sulfur dioxide',
- 'total sulfur dioxide'
- 'pH',
- 'sulphates',
- ‘alcohol’

In order to avoid data leakage, and to build the model using the relevant features we first select the relevant features and then we proceed with Train-test-split

Train Test Split

Split the dataset, our Y (the target variable) is ‘quality’. Using the train_test_split function from scikit-learn, where the test_size=0.2 indicates that 20% of the data will be allocated for testing, while 80% will be used for training.

Feature Scaling

Feature scaling was executed as a crucial process to standardise the range of independent variables in the dataset, ensuring that they contribute equally to model performance and preventing certain features from dominating others. Given that the data do not follow a normal distribution, I employed MinMaxScaler, which scales numerical features to a specified range (typically between 0 and 1) by transforming them based on their minimum and maximum values.

Oversampling

Since there is a strong class imbalance, where the Medium class has more observations,compared to the ‘Low’ and ‘High’, it is important to balance all the classes.

I decided to do oversampling to increase the number of instances in the minority classes using RandomOverSampler.

Model experimentation

We evaluated the performance of following classification models:

- Logistic Regression
- SVC
- Decision Tree
- KNN
- Random Forest

Logistic Regression: It is my baseline model, which is useful for multiclass classification and offers interpretability by showing how each chemical property contributes to predicting wine quality.

SVC (Support Vector Classifier): It’s effective in handling complex, non-linear decision boundaries, which might exist in the relationship between wine properties and quality.

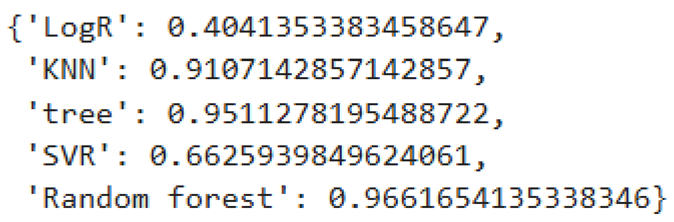
Decision Tree: This model provides an easily interpretable model by visualising the decision-making process, showing how individual features influence wine quality classification.

KNN (K-Nearest Neighbors): This model considers similarity between wines based on their chemical properties, assuming that wines with similar properties will have similar quality.

Random Forest: It combines multiple decision trees to improve prediction accuracy and reduce overfitting, making it a robust choice for handling complex datasets like this one.

In order to test and evaluate the models, a function was used to fit the models in the training data and to evaluate them on the test data using the sklearn .score() method.

The output of the function shows that Logistic Regression, performed the worst andRandom Forest is the best performer.



```
{'LogR': 0.4041353383458647,
 'KNN': 0.9107142857142857,
 'tree': 0.9511278195488722,
 'SVR': 0.6625939849624061,
 'Random forest': 0.9661654135338346}
```

I subsequently aimed to explore the logistic regression (the worst performer) and random forest model, the model that performed better,

Hyperparameter tuning

In order to optimise models performance I executed hyperparameter tuning of the logistic regression and the Random forest model . Random Search was used as this technique is generally more efficient, especially in larger search spaces, as it explores a variety of combinations without evaluating every possibility.

Models evaluation

Logistic regression:

Accuracy

Before tuning 40% of the predictions made by the model on the test set were correct, while after the tuning, there has been a slight decrease in accuracy (36%), suggesting that the model had a decrease in accuracy after tuning.

Precision

Precision indicates how many of the predicted positive instances were actually positive. A precision of 94% means that the model predicts a positive class, it is correct 94% of the time. This is very high, indicating that the model is good at identifying the positive class among the predicted positives.

Recall

However, the recall (measures the model's ability to capture all the actual positive instances) shows that the model only identified 36% of the actual positives in the test set. This indicates a significant number of actual positives were missed (false negatives).

F1 Score

The F1 score is the harmonic mean of precision and recall. It provides a single score that balances. The score of 0.50 indicates a moderate balance between precision and recall, but it suggests that the model might be struggling to capture the positive class effectively.

Random Forest

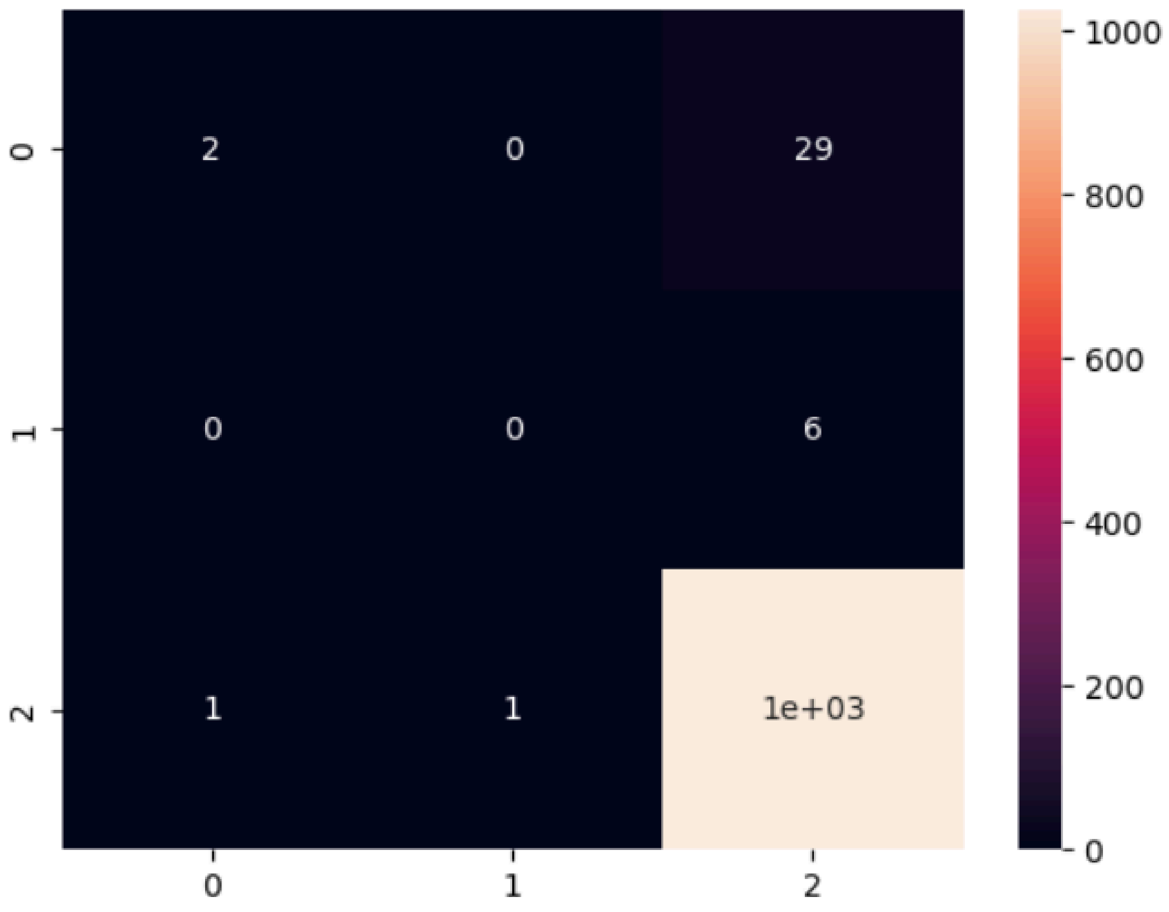
The performance of the Random Forest is shown below:

```
Classification Report:
              precision    recall  f1-score   support

        high       0.67      0.06      0.12        31
         low       0.00      0.00      0.00         6
      medium       0.97      1.00      0.98      1027

    accuracy                           0.97      1064
   macro avg       0.54      0.35      0.37      1064
weighted avg       0.95      0.97      0.95      1064

Confusion Matrix:
[[   2    0   29]
 [   0    0    6]
 [   1    1 1025]]
```



It was used the classification report from Sklearn to evaluate the model metrics: (https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.classification_report.html)

The classification report shows that the model has a strong bias toward the “medium”class,and fails to identify instances of the low and high classes. This suggests that the model might need further improvement for balancing the classes.

Conclusion

The Random Forest model emerged as the most effective in addressing our research question of predicting wine quality based on historical customer ratings and specific wine chemical properties; however, improvements are necessary to mitigate bias towards certain classes. This can be achieved by further investigating oversampling or undersampling techniques and refining the hyperparameter tuning process.