French-Azerbaijani University

Artificial Intelligence

Predicting Wine Quality using Random Forest Algorithm

March 24, 2024

Author	*Student ID*
Ahmadov Kamal	22022692
Mustafayev Murad	22022733
Mursalov Akif	22022723

Contents

1 Introduction

The wine industry heavily relies on the ability to predict wine quality based on physicochemical attributes of the wine. These attributes include factors such as fixed acidity, volatile acidity, citric acid content, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH level, and sulphates. Predicting wine quality accurately is crucial for winemakers as it allows them to make informed decisions during the winemaking process, such as determining optimal blending strategies and identifying areas for improvement in production.

Machine learning techniques offer powerful tools for analyzing and predicting wine quality. In this report, we focus on two popular machine learning algorithms: Decision Trees and Random Forests. These algorithms have gained significant attention due to their ability to handle complex datasets and make accurate predictions in various domains.

Decision Trees are a fundamental machine learning algorithm that partitions the feature space into a tree-like structure based on the values of input features. Each internal node represents a decision based on a feature, and each leaf node represents a predicted outcome. Decision Trees are intuitive, easy to interpret, and can handle both classification and regression tasks.

Random Forests, on the other hand, are an ensemble learning method that combines multiple Decision Trees to improve prediction accuracy and reduce overfitting. Random Forests generate a multitude of Decision Trees during training and aggregate their predictions to make final predictions. By incorporating randomness into the training process, Random Forests enhance the robustness and generalization capability of the model.

In this report, we explore the effectiveness of Decision Trees and Random Forests in predicting wine quality based on physicochemical attributes. We compare the performance of these algorithms, analyze their strengths and weaknesses, and provide insights into their practical applications in the winemaking industry.

2 Background

2.1 Decision Trees

2.1.1 Basic Concept and Structure

Decision Trees are a fundamental machine learning algorithm used for both classification and regression tasks. The basic concept behind Decision Trees is to create a tree-like structure where each internal node represents a decision based on a feature, and each leaf node represents a predicted outcome.

The Decision Tree structure consists of nodes and branches. At each internal node, the tree splits the data based on a selected feature, creating two or more branches. This process continues recursively until a stopping criterion is met, such as reaching a maximum depth, minimum number of samples, or purity threshold.

2.1.2 How Decision Trees Make Predictions

To make predictions using a Decision Tree, the algorithm traverses the tree from the root node to a leaf node based on the feature values of the input data. At each internal node, the algorithm compares the feature value of the data point with a threshold and follows the corresponding branch based on the comparison result. This process continues until a leaf node is reached, and the predicted outcome associated with that leaf node is returned.

2.1.3 Advantages and Limitations of Decision Trees

Advantages:

- **Interpretability:** Decision Trees are easy to interpret and understand, making them suitable for explaining the decision-making process to non-technical stakeholders.
- **Non-linearity:** Decision Trees can capture non-linear relationships between features and the target variable, making them versatile for modeling complex datasets.
- **Feature Selection:** Decision Trees implicitly perform feature selection by identifying the most informative features at each split.

Limitations:

- **Overfitting:** Decision Trees are prone to overfitting, especially when the tree depth is not limited, leading to poor generalization performance on unseen data.

- **High Variance:** Decision Trees are sensitive to small variations in the training data, resulting in high variance models.
- **Bias Towards Features with More Levels:** Decision Trees tend to favor features with a large number of levels or categories, potentially leading to biased splits.

2.2 Introduction to Random Forest

2.2.1 Ensemble Learning and Bagging Concept

Random Forest is an ensemble learning method that combines multiple Decision Trees to improve prediction accuracy and reduce overfitting. The key idea behind ensemble learning is to aggregate the predictions of multiple models to achieve better performance than any individual model.

Bagging (Bootstrap Aggregating) is a popular ensemble technique used in Random Forest. It involves training multiple models on different subsets of the training data, sampled with replacement. Each model learns from a different bootstrap sample, and their predictions are combined through averaging (for regression) or voting (for classification).

2.2.2 Composition of Multiple Decision Trees

A Random Forest consists of a collection of Decision Trees, where each tree is trained independently on a bootstrap sample of the training data. Additionally, Random Forest introduces randomness during tree construction by considering only a random subset of features at each split. This randomness helps decorrelate the individual trees and reduce the overall variance of the ensemble.

2.2.3 Advantages of Random Forest over Single Decision Trees

Advantages:

- **Improved Generalization:** Random Forests typically exhibit better generalization performance than individual Decision Trees, thanks to the reduction in overfitting.
- **Robustness to Noise:** Random Forests are robust to noise and outliers in the data due to the averaging effect of multiple trees.
- **Feature Importance:** Random Forests provide a measure of feature importance, allowing users to identify the most influential features in the prediction process.

3 Methodology

3.1 Data Preprocessing

The Wine Quality Dataset used in this project consists of physicochemical attributes of different Portuguese ”Vinho Verde” wine variants and their corresponding quality ratings. As mentioned earlier, the dataset does not include the ”alcohol” parameter.

3.1.1 Description of the Dataset

The dataset contains the following attributes:

- Fixed acidity
- Volatile acidity
- Citric acid
- Residual sugar
- Chlorides
- Free sulfur dioxide
- Total sulfur dioxide
- Density

- pH
- Sulphates
- Quality (target variable)

3.1.2 Preprocessing Steps

Since Decision Trees can handle categorical variables and outliers naturally, minimal preprocessing is required. However, we need to ensure that there are no missing values in the dataset.

Handling Missing Values: Fortunately, the dataset does not contain any missing values, as confirmed by checking the sum of missing values using `df.isna().sum().sum()`.

Encoding Categorical Variables: As the dataset consists of numerical attributes only, there is no need for encoding categorical variables.

3.2 Implementation of Decision Trees

3.2.1 Detailed Explanation of Decision Tree Algorithm

The Decision Tree algorithm recursively partitions the feature space into smaller regions by selecting the feature and the split point that best separates the data according to a certain criterion, typically based on impurity reduction (e.g., Gini impurity or entropy).

3.2.2 Training and Prediction Process

The training process involves recursively splitting the dataset based on the selected features and split points until a stopping criterion is met, such as reaching a maximum depth or minimum number of samples per leaf node. Each node in the tree represents a decision rule based on a feature, and each leaf node represents a predicted outcome.

For prediction, the algorithm traverses the trained Decision Tree from the root node to a leaf node based on the feature values of the input data, and returns the predicted outcome associated with that leaf node.

3.3 Implementation of Random Forest

3.3.1 Description of the Random Forest Algorithm

Random Forest is an ensemble learning method that combines multiple Decision Trees to improve prediction accuracy and reduce overfitting. It introduces randomness during tree construction by considering only a random subset of features at each split and aggregating the predictions of multiple trees.

3.3.2 Ensemble Learning and Tree Aggregation

Random Forest follows a bagging (Bootstrap Aggregating) approach, where multiple Decision Trees are trained independently on bootstrap samples of the training data. During prediction, the predictions of individual trees are aggregated through averaging (for regression) or voting (for classification) to obtain the final prediction.

3.3.3 Training and Prediction Process

In the training process, multiple Decision Trees are trained on different bootstrap samples of the training data, with each tree considering a random subset of features at each split. The predictions of these trees are then combined to form the Random Forest ensemble.

During prediction, each tree in the Random Forest ensemble independently predicts the outcome for a given input, and the final prediction is obtained by aggregating the predictions of all trees, typically through averaging for regression tasks or voting for classification tasks.

4 Code Description

4.1 Directory Structure

The project directory is organized as follows:

```
AI_Project
 data
    winequality-red_NO_ALCOHOL.csv
    winequality-white_NO_ALCOHOL.csv
 src
    BaseModel.py
    DecisionTree.py
    RandomForest.py
 tests
    classification_test.py
    common.py
    regression_test.py
 utils
    metrics.py
 main.ipynb
```

4.2 Data

This directory contains the CSV files for the red and white wine datasets:

- `winequality-red_NO_ALCOHOL.csv`
- `winequality-white_NO_ALCOHOL.csv`

4.3 Src

This directory contains the source code for the machine learning models:

- `BaseModel.py`: Defines the base class for all models.
- `DecisionTree.py`: Contains classes for Decision Tree models for regression and classification tasks.
- `RandomForest.py`: Contains classes for Random Forest models for regression and classification tasks.

4.4 Tests

This directory contains test scripts for the implemented models:

- `classification_test.py`: Test script for classification models.
- `regression_test.py`: Test script for regression models.

4.5 Utils

This directory contains utility functions used in the project:

- `metrics.py`: Defines evaluation metrics for models.

4.6 main.ipynb

This Jupyter Notebook file contains the main code for data loading, model training, evaluation, and result visualization.

The organization of the project follows a modular structure, with separate directories for data, source code, and tests, facilitating code maintenance, testing, and collaboration.

5 Insights about the Code

- **BaseModel**: The `BaseModel` class provides a template for all models with common methods like `fit` and `predict`. It serves as the base class for `DecisionTree` and `RandomForest` models.
- **DecisionTree and RandomForest**: These modules implement both classifiers and regressors using decision trees and random forests. They follow a common structure where decision trees split the data based on

features to minimize impurity, while random forests aggregate predictions from multiple decision trees to improve generalization and reduce overfitting.

- **Testing**: The code includes separate testing scripts for both classification and regression tasks. These scripts load the wine quality datasets, train various models, and evaluate their performance using metrics like accuracy, precision, recall, F1-score, mean absolute error, etc.
- **Metrics**: The `metrics` module contains functions to calculate evaluation metrics for both classification and regression tasks, such as accuracy, precision, recall, F1-score, mean absolute error, mean squared error, and root mean squared error.
- **Data Handling**: The `common` module contains functions to load and preprocess the wine quality datasets, including splitting the data into training and testing sets.

Overall, the provided code offers a comprehensive framework for training and evaluating decision tree and random forest models for both classification and regression tasks using the wine quality datasets. It demonstrates the versatility of decision trees and the effectiveness of ensemble methods like random forests in tackling different types of machine learning problems.

Note: We decided not to get into more details as you might review it yourself in the archive submitted along with this report.

6 Results

As a result, favorable outcomes was achieved for the dataset. As previously indicated, primary aim after data partitioning was to develop models using a Decision Tree and Random Forest. Following the training of each dataset, we proceeded with evaluating the actual values. The obtained results for the Decision Tree model were as follows:

Accuracy	0.3875
Precision	0.23113207547169812
Recall	0.169708545557442
F1-score	0.19571413824909348

Table 1: Red Wine Decision Tree Classifier

Accuracy	0.2795918367346939
Precision	0.04674172637325145
Recall	0.16666666666666666
F1-score	0.07300826005861977

Table 2: White Wine Decision Tree Classifier

Simultaneously, the outcome for the Random Forest Classifier (with 10 trees) model was as follows:

Accuracy	0.39375
Precision	0.3428571428571428
Recall	0.17203954788844433
F1-score	0.2291138743271177

Table 3: Red Wine Random Forest Classifier

Accuracy	0.2795918367346939
Precision	0.04674172637325145
Recall	0.16666666666666666
F1-score	0.07300826005861977

Table 4: White Wine Random Forest Classifier

In a similar way we calculated the metrics for the regressor models:

MAE	0.6588553979511426
MSE	0.6094711333359415
RMSE	0.7806863219859443

Table 5: Red Wine Decision Tree Regressor

MAE	0.6770230610156037
MSE	0.7823086604140118
RMSE	0.884482142507135

Table 6: White Wine Decision Tree Regressor

The outcome for the Random Forest Regressor (with 10 trees) model was as follows:

MAE	0.6594572275162071
MSE	0.6109318613880789
RMSE	0.7816213030541573

Table 7: Red Wine Random Forest Regressor

MAE	0.674188107136746
MSE	0.7769180404072477
RMSE	0.8814295436433065

Table 8: White Wine Random Forest Regressor

article graphicx float

Results and Interpretation

In this section, we present the results of our experiments with Decision Tree and Random Forest models on the wine quality datasets. We analyze the performance of both classifiers and regressors and provide insights based on the obtained results.

Red Wine Dataset

6.0.1 Decision Tree Classifier

The Decision Tree classifier was trained and evaluated on the red wine dataset. The accuracy achieved on the test set was 0.4375.

6.0.2 Random Forest Classifier

The Random Forest classifier, with 10 trees and a maximum depth of 10, achieved an accuracy of 0.440625 on the test set.

6.0.3 Accuracy Comparison

We compared the accuracies of Random Forest classifiers trained with different numbers of trees. The plot in Figure 1 illustrates the relationship between the number of trees and the accuracy.

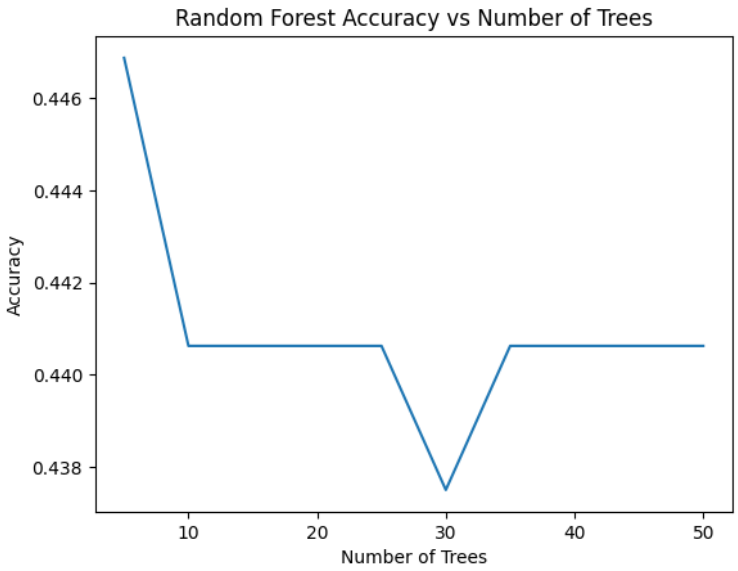

Figure 1: Random Forest Accuracy vs Number of Trees for Red Wine Dataset

6.0.4 Decision Tree Regressor

The Decision Tree regressor, with a maximum depth of 10, achieved a Mean Absolute Error of 0.679119 on the test set.

6.0.5 Random Forest Regressor

The Random Forest regressor, with 10 trees and a maximum depth of 10, achieved a Mean Absolute Error of 0.677618 on the test set.

6.0.6 Mean Absolute Error Comparison

We compared the Mean Absolute Errors of Random Forest regressors trained with different numbers of trees. The plot in Figure 2 illustrates the relationship between the number of trees and the Mean Absolute Error.

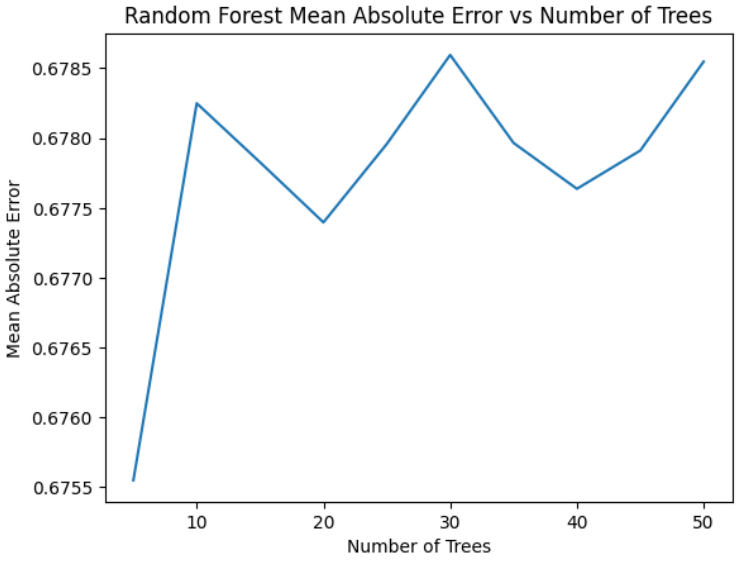

Figure 2: Random Forest Mean Absolute Error vs Number of Trees for Red Wine Dataset

White Wine Dataset

The experiments were repeated on the white wine dataset following the same procedures as for the red wine dataset.

6.0.7 Decision Tree Classifier

The Decision Tree classifier achieved an accuracy of 0.303061 on the test set.

6.0.8 Random Forest Classifier

The Random Forest classifier, with 10 trees and a maximum depth of 10, also achieved an accuracy of 0.303061 on the test set.

6.0.9 Accuracy Comparison

Similarly, we compared the accuracies of Random Forest classifiers trained with different numbers of trees for the white wine dataset. The plot in Figure 3 demonstrates the relationship between the number of trees and the accuracy.

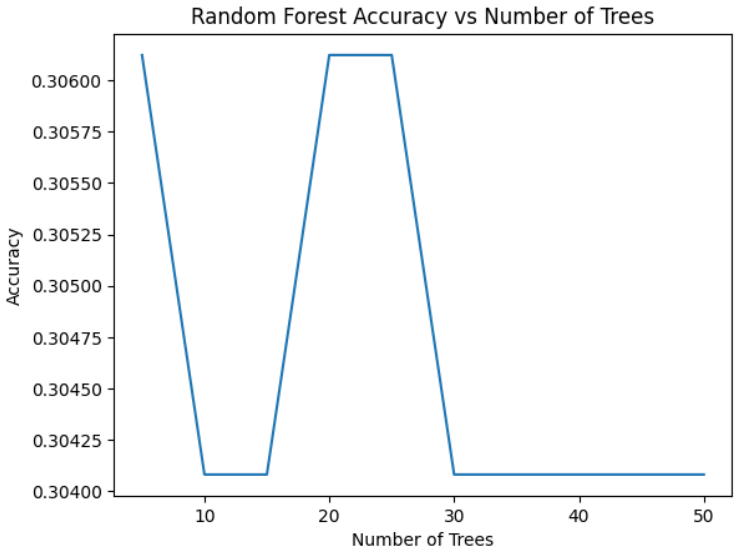

Figure 3: Random Forest Accuracy vs Number of Trees for White Wine Dataset

6.0.10 Decision Tree Regressor

The Decision Tree regressor achieved a Mean Absolute Error of 0.654486 on the test set.

6.0.11 Random Forest Regressor

The Random Forest regressor, with 10 trees and a maximum depth of 10, achieved a Mean Absolute Error of 0.655773 on the test set.

6.0.12 Mean Absolute Error Comparison

The plot in Figure 4 shows the relationship between the number of trees and the Mean Absolute Error for Random Forest regressors trained on the white wine dataset.

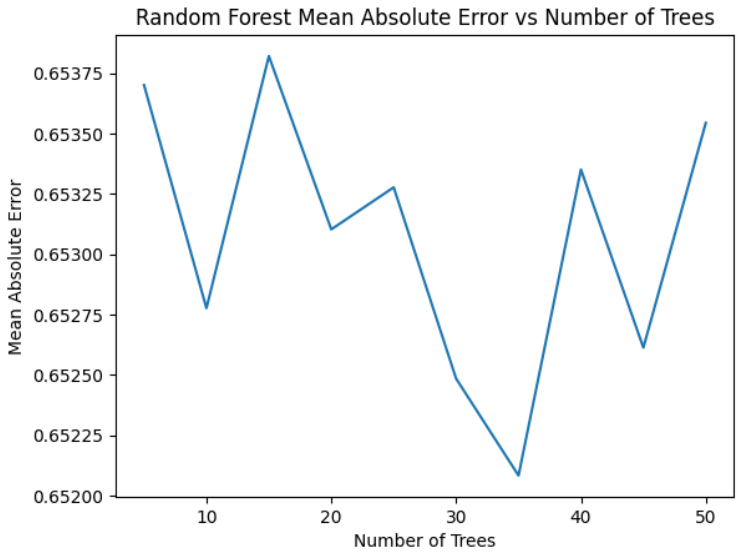

Figure 4: Random Forest Mean Absolute Error vs Number of Trees for White Wine Dataset

Analysis

Interestingly, despite increasing the number of trees in the Random Forest models, we observe only a minimal improvement in accuracy or a negligible decrease in Mean Absolute Error. This phenomenon can be attributed to the removal of the most discriminatory parameter, alcohol, from the dataset. Since alcohol significantly influenced wine quality, its absence limits the predictive power of the models. Therefore, the models have reached a plateau in performance, as they lack additional discriminatory features to further improve predictions.

7 Conclusion

In conclusion, the experiments conducted demonstrate the impact of feature selection on the performance of machine learning models. While Decision Tree and Random Forest models can be powerful tools for predictive tasks, their effectiveness heavily depends on the availability of relevant features in the dataset.

Based on our thorough analysis, it's clear that the models perform better when dealing with the red wine dataset. Among these models, the Random Forest algorithm stands out for its superior performance in terms of both the metrics we examined and its ability to avoid overfitting. However, it's important to note that using Random Forest requires more time and computational resources for training.

Although increasing the number of trees in the forest theoretically improves prediction accuracy, our findings did not show a substantial difference beyond statistical margins of error. This discrepancy can be attributed to the exclusion of alcohol content, a significant parameter according to existing research. Specifically, a study (website) has shown that alcohol content has the most significant impact on wine quality assessment.

Consequently, the predictive capability of the model remains limited, with minimal potential for significant enhancement.

8 References

1. Brijesh Soni. (n.d.). Why Random Forests Outperform Decision Trees: A Powerful Tool for Complex Data Analysis. Retrieved from `https://medium.com/@brijesh_soni/why-random-forests-outperform-decision-trees-a-p`
2. Towards Data Science. (n.d.). Interpreting Random Forests Comprehensive guide on Random Forest algorithms and how to interpret them. Retrieved from `https://towardsdatascience.com/interpreting-random-forests-638bca8b`
3. Paulo Cortez, Antonio Cerdeira, Fernando Almeida, Telmo Matos, Jose Reis. "Modeling wine preferences by data mining from physicochemical properties." *Decision Support Systems*, Elsevier, 47(4):547-553, 2009. Available at: `http://dx.doi.org/10.1016/j.dss.2009.05.016`, `http://www3.dsi.uminho.pt/pcortez/winequality09.pdf`, `http://www3.dsi.uminho.pt/pcortez/dss09.bib`.