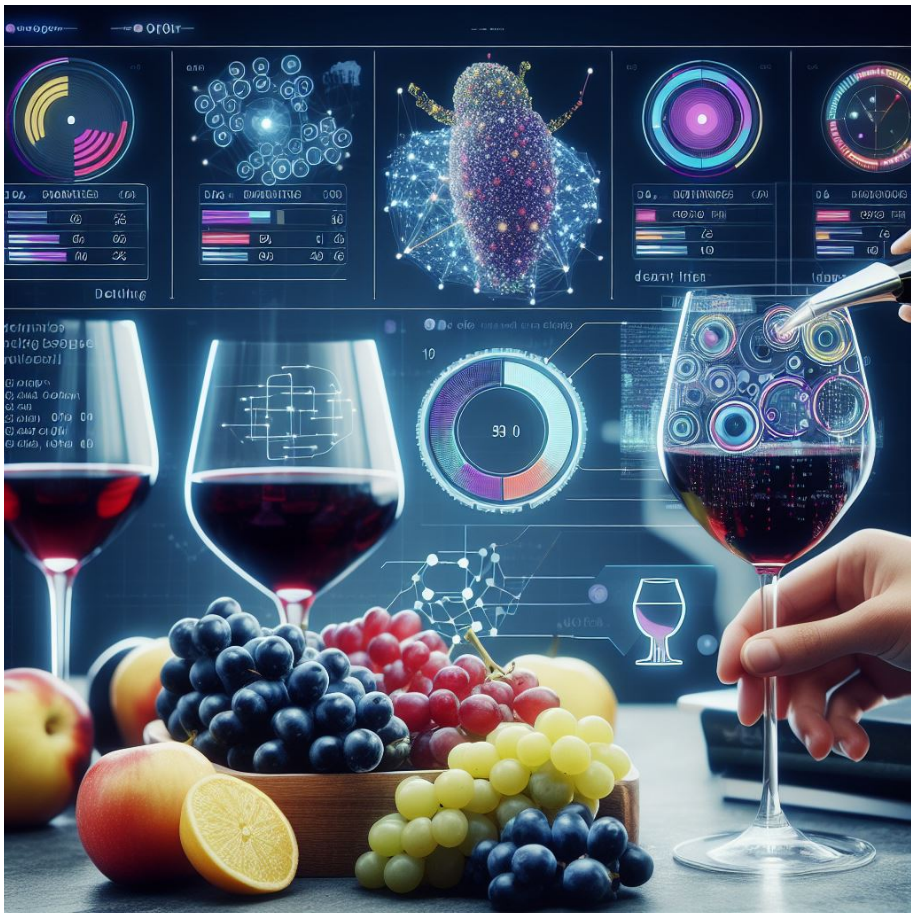
Wine Quality Prediction Using Random Forest

Artificial Intelligence L2-L3 project 2024



Submitted by:

Rita Gayitmazova

Humay Yusifli

24.03.2024

Contents

Objective

The objective of this project is to understand the concept of random forests and apply them to the Wine Quality Dataset. This dataset consists of samples of red and white wines, with various features and quality. The project aims to answer the following questions:

- What features are important, and which ones are redundant and can be removed?
- How does each feature value affect our target metric?
- What are the factors for each prediction?
- How to estimate the confidence of each prediction?

0.1 Introduction

We have provided with Wine Quality Dataset containing features and quality of both red and white wine.
Dataset Description

- **Number of Samples:**
  - Red wine: 1599
  - White wine: 4898
- **Attribute Information:**

1. Fixed acidity
2. Volatile acidity
3. Citric acid
4. Residual sugar
5. Chlorides
6. Free sulfur dioxide
7. Total sulfur dioxide
8. Density
9. pH
10. Sulfates
11. Quality (score between 0 and 10)

0.2 Main body

0.2.1 Preprocessing

The first step in the analysis is to combine the provided red and white wine datasets. A new column named 'type' is added to distinguish between wine varieties:

```
df = pd.concat([red_df, white_df])
```

Next, a check is performed for categorical values and null values, as these data types need to be addressed before being used with scikit-learn models – because scikit-learn can't work with these values. Fortunately, this check reveals no null values are present:

```
df.isna().sum().sum()    Output: 0
```

However, the 'type' column, currently containing strings, is converted into a numerical format using categorical encoding. This process assigns integer values (0 and 1 in this case) to the different string categories ("red" and "white").

```
categories = {}
for p in ['type']:
    df[p] = pd.Categorical(df[p])
    categories[p] = df[p].cat.categories

df['type'] = df[['type']].apply(lambda x: x.cat.codes)
categories
{'type': Index(['red', 'white'], dtype='object')}
```

Finally, the dataset is divided into training and testing sets using a 80:20 ratio, ensuring adequate data for both model development and evaluation:

```
train_df, test_df = sklearn.model_selection.train_test_split(df,
   test_size=0.2)
train_X.shape, test_X.shape = (5197, 11), (1300, 11)
```

0.2.2 Random Forest

In simple terms, a random forest can be understood as an ensemble of decision trees trained on random subsets of data samples and features. The core concept is to combine multiple independent models and use their averaged predictions.

Training a Basic Random Forest: A basic random forest model was trained with 100 trees and a minimal leaf node size of 100:

```
model =sklearn.ensemble.RandomForestRegressor(100,min_samples_leaf
   =100,oob_score=True)
model.fit(train_X, train_y)
```

Mean Absolute Error of testing dataset = 0.5834052244481425
Mean Absolute Error of trainig dataset = 0.5592789385682757
Out of Bag Error of training dataset = 0.5754998508451644

Out-of-bag Error: An Alternative for Validation

Random forests employ a unique validation method by utilizing data points excluded from the training of certain decision trees. These ”unseen” data points enable error estimation through predictions generated by the trees that didn’t include them. This out-of-bag (OOB) error offers insights into how the model might perform on new data.

Influence of number of trees and minimum leaf samples

A significant advantage of random forests is their resistance to overfitting as the number of trees increases. This is due to the averaging of uncorrelated errors, which mitigates the risk of overfitting. As the number of trees grows, model quality will generally improve, approaching an asymptote. This relationship between model performance, the number of trees, and minimum leaf samples is illustrated in Figure 1.

```
px.line(final_df.pivot(index = 'num_trees', columns = '
   min_samples_leaf', values = 'test_ma
```

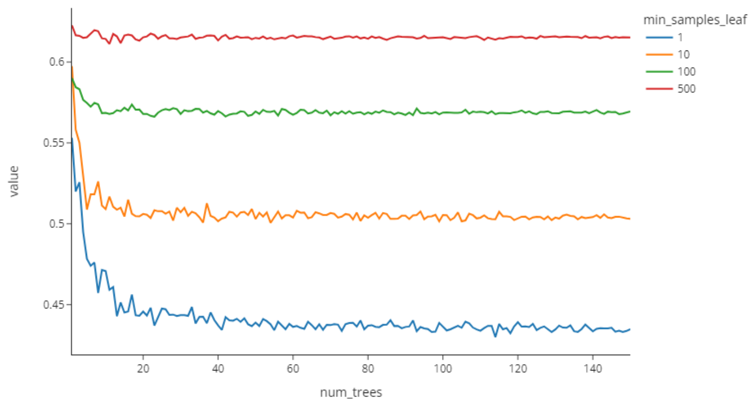


Figure 1:

Figure 1 analysis shows that a smaller min_samples_leaf value tends to give a lower mean absolute error (MAE) for a fixed number of trees. However, longer execution times are a cost for this. On the other hand, quicker execution is achieved with bigger min_samples_leaf values, but the MAE is not much improved. Additionally, increasing the number of trees generally reduces MAE, but this improvement stops beyond a certain point (between 80 and 100 trees in this instance). Therefore, the selection of optimal values for both the number of trees and min_samples_leaf requires careful consideration to balance accuracy and computational efficiency.

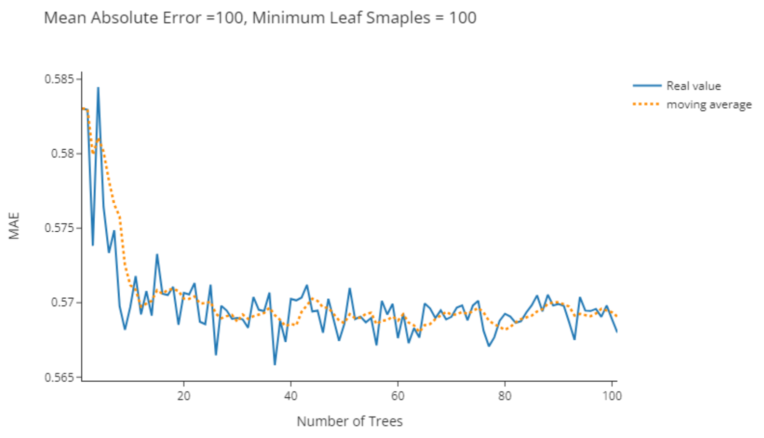


Figure 2:

Maximum depth vs. MAE

We also tested how maximum depth of decision trees can influence MAE:

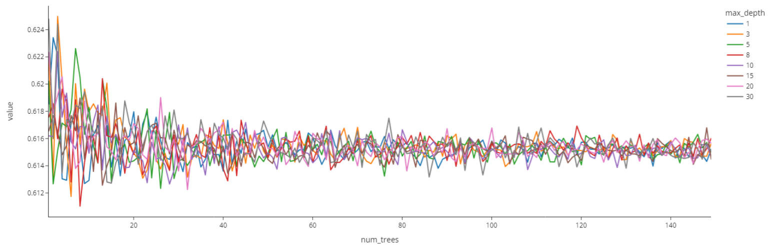


Figure 3:

We can conclude that depth of decision trees doesn't have much influence on mean absolute error.

0.2.3 Feature Importances

What features are important, and which ones are redundant and can be removed? *model.feature_importances_* have been used to visualize the feature importance:

```
    def visualize_feature_importance(model, feature_names,
       importance_threshold=None):
        data = list(zip(feature_names, model.feature_importances_))
        importance_df = pd.DataFrame(data, columns=['feature', '
           importance'])
        importance_df = importance_df.set_index('feature').sort_values
           ('importance', ascending=False)
```

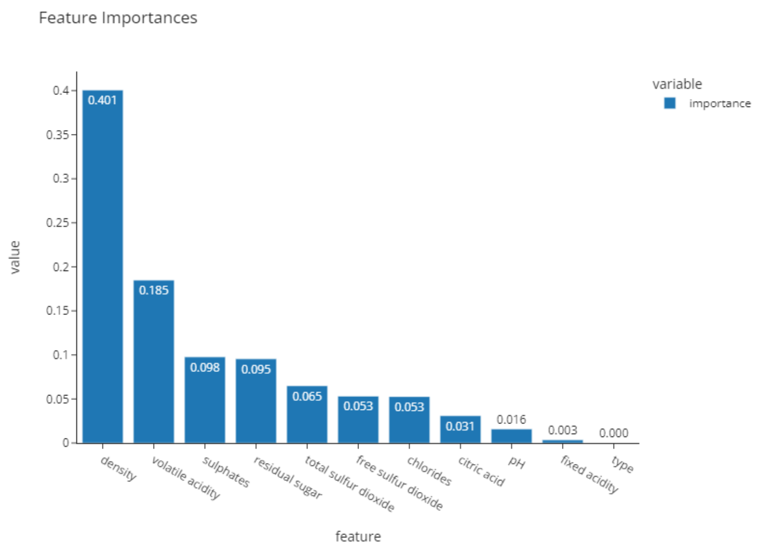


Figure 4:

Figure 4 indicates 'density' and 'volatile acidity' as the features demonstrating the highest importance. Conversely, 'fixed acidity' exhibits minimal influence, suggesting its redundancy and potential removal from the dataset. While the 'type' feature has zero importance, it is retained due to its deliberate introduction for dataset merging and differentiation between red and white wine varieties.

Redundancy and Correlation Analysis

Initial analysis identified the least important features. To further investigate potential redundancy, a correlation heatmap was generated (Figure 5) to examine relationships between the remaining features.

```
sns.heatmap(df.corr()>0.7, annot=True, cbar=False)
```

Addressing Correlated Features

Figure 5 demonstrates a strong correlation (0.7) between 'total sulfur dioxide' and 'free sulfur dioxide'. To address this redundancy, the decision has been made to remove one of these features from the dataset:

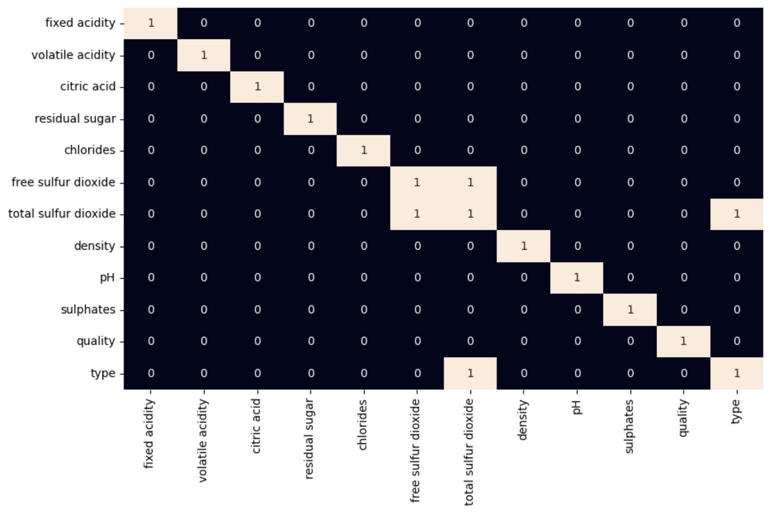
```
df = df.drop('total sulfur dioxide', axis=1)
```

Partial Dependence

How does each feature value affect our target metric? We can use sklearn.inspection module to easily plot this relations.

```
sklearn.inspection.PartialDependenceDisplay.from_estimator(clf,
    train_X, range(10))
```

Figure 5: Figure 5



A variety of conclusions can be derived from Figure 6:

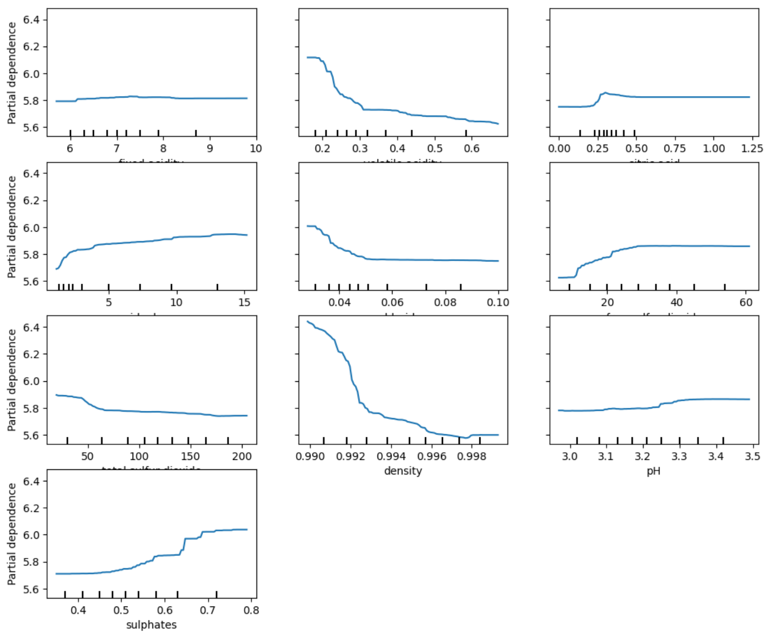
- wine quality decreases with growth of density, the lower the level of density (the better the quality). The same conclusion with volatile acidity, but with a little bit less impact.
- Wine quality increases with the growth of free sulfur dioxide up to 30, but then it is almost constantly stable after this point;
- Wine quality decreases with the growth of chlorides up to 0.06, after this, it is almost constant fixed acidity that does not change almost anything. That's why we got it as the least important feature.
- with cidric acid quality change only at in the small range of values(0.25, 0.5), except this points it's stable.
- with residual sugar quality increases logarithmically.
- with sulphates quality increases as well, whereas pH does not have much impact on quality

What are the factors for each prediction? The main factors are feature importance and partial dependencies. The importance of a feature is measured by looking at how much the tree nodes that use that feature reduce impurity on average (across all trees in the forest). Features that are used at the top of the trees are usually more important. We have already discussed how each particular feature affects output value.

Confidence of predictions

How to estimate the confidence of each prediction? For that, we could calculate predictions from each tree in the ensemble and look at variance or standard deviation:

Figure 6: Figure 6

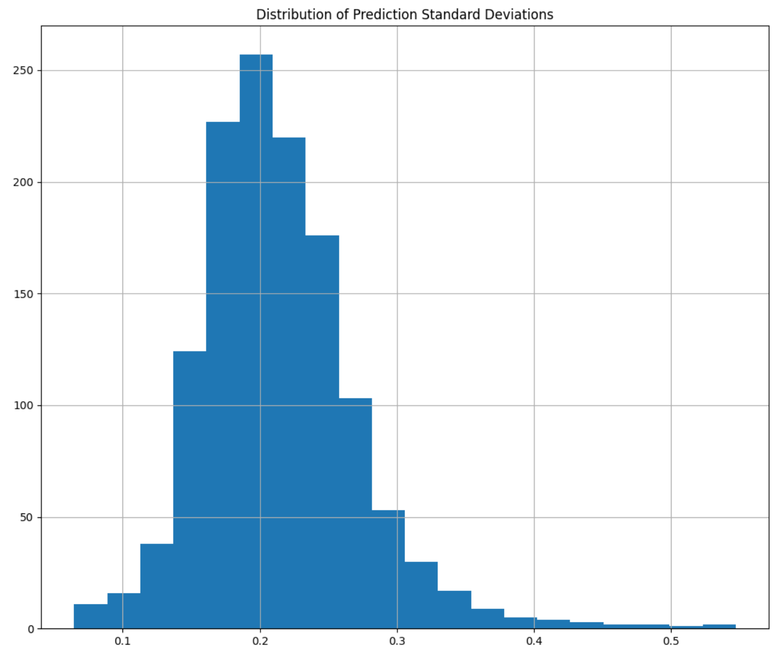


```python

predictions_test = model.predict(test_X)
tree_predictions = [dt.predict(test_X.values) for dt in model.
    estimators_]

predictions_mean = np.mean(tree_predictions, axis=0)
predictions_std_dev = np.std(tree_predictions, axis=0)
test_df['predicted_values'] = predictions_test
test_df['mean_prediction'] = predictions_mean
test_df['std_dev_prediction'] = predictions_std_dev
ax = test_df['std_dev_prediction'].hist(bins=20)
ax.set_title('Distribution of Prediction Standard Deviations')
```

Figure 7: Figure 7



- Consistency of Predictions: The majority of predictions have a standard deviation between 0.1 and 0.3. This suggests that there is a moderate level of consistency in the predictions made by the individual trees within the random forest for most data points.
- Confidence Estimation: A lower standard deviation indicates higher confidence in the prediction, as the individual trees are in closer agreement. Conversely, a higher standard deviation suggests less confidence, as the trees' predictions vary more widely.

0.3 Summary

In this project, we have learned how to interpret the Random Forest model on the Wine Quality Dataset. Also learned to determine which features are more important than the others and which ones are redundant and can be removed from the dataset. Apart from that, we learned how to define how each feature affects the output target value.

0.4 References

- https://towardsdatascience.com/interpreting-random-forests-638bca8b49ea
- P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236. Available at: [@Elsevier] http://dx.doi.org/10.1016/j.dss.2009.05.016 [Pre-press (pdf)] http://www3.dsi.uminho.pt/pcortez/winequality09.pdf [bib] http://www3.dsi.uminho.pt/pcortez/dss09.bib

  https://github.com/miptgirl/miptgirl_medium/blob/main/random_forests_101/random_forests_101.ipynb