REPORT : Wine Quality Prediction using Random Forest Technique

1.)Description of the supervised learning technique - Random Forest Classifier

Random Forest is a supervised machine learning algorithm which is based on ensemble learning(A technique which combines giving a more accurate prediction by combining the prediction of several techniques). It is mainly used for classification and regression problems by combining the prediction of several decision tree and making a more accurate prediction. It is suitable for large data sets with complex correlation between features and target variables .

For both classification and regression problems, the subset of predictor variables selected to split an internal node depends on predetermined splitting criteria that are formulated as an optimization problem. - Resource

Underlying Principles

Random forest works on two underlying principles - Bagging & Random Subspaces

- Bagging(Bootstrap Aggregating) - Random forest uses bootstrapping to produce several subset of the original data set. Each decision Tree in random forest is trained on a different subset. This guarantees that the model is resistant towards overfitting

- Random Subspaces - Only a random subset of features is employed at each decision node of a tree, as opposed to taking into account all of the features that are accessible. By increasing the diversity among the trees, this reduces their correlation and enhances the model's generalizability.

Training Algorithm working

1. - several bootstrapped subset of data set are created
2. - for every subset a decision tree is created using random features
3.- every tree makes a prediction and all trees are combined to provide the final prediction
4. - to generate predictions fresh data can be sent to the forest. Each tree will then vote on the result and the class with most votes is chosen as final prediction

2. Data set description

The data set used is wine.csv which is taken from kaggle containing 12 columns and 1599 rows.

Data set details -

The basic details about column data types, names and how much memory they use \ are found out by using pandas .info() function and result is given below-

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   object 
dtypes: float64(11), object(1)
memory usage: 150.0+ KB
```

There are two feature types present in this data set

- Continuous variable
- Categorical variable(quality: 0–10 integer numbers, however we'll reduce it to a "good" or "bad" binary classification.)

Target variable chosen-

Wine quality(is chosen) which we categorize into two classes(quality < 5) bad and good(quality > 5)

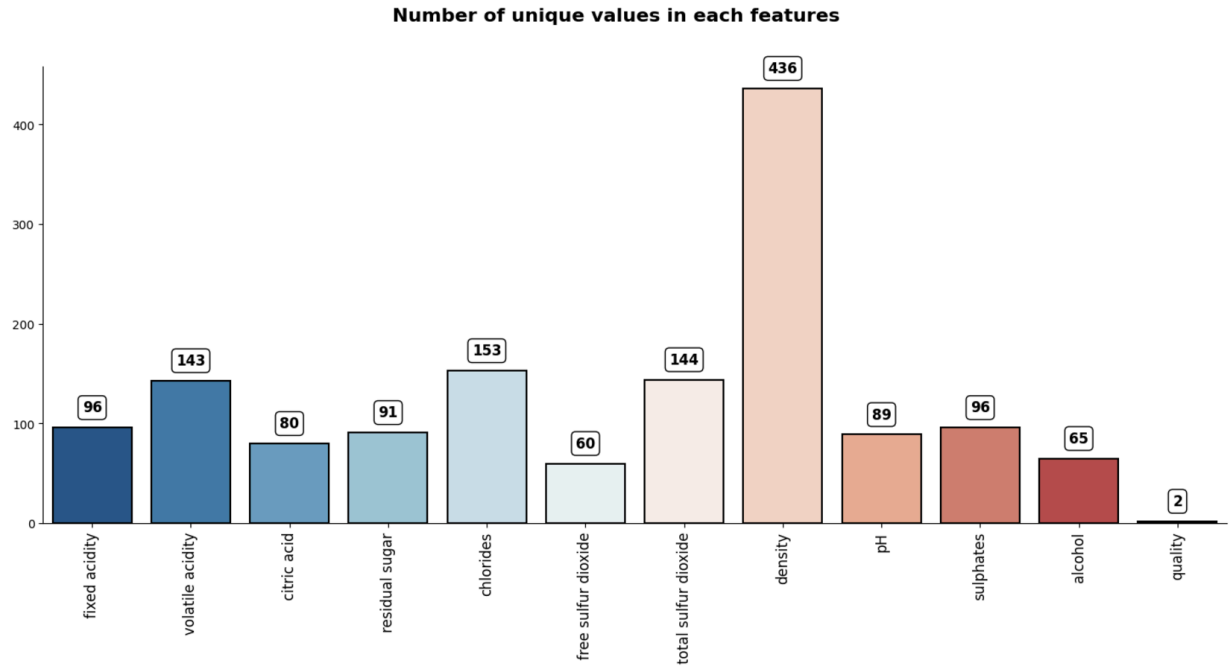
PROBLEM STATEMENT

The goal is to predict the quality of wine based on its internal composition like the amount of alcohol or phosphates present in wine, its density and also rest of the column

3.) Exploratory Data Analysis

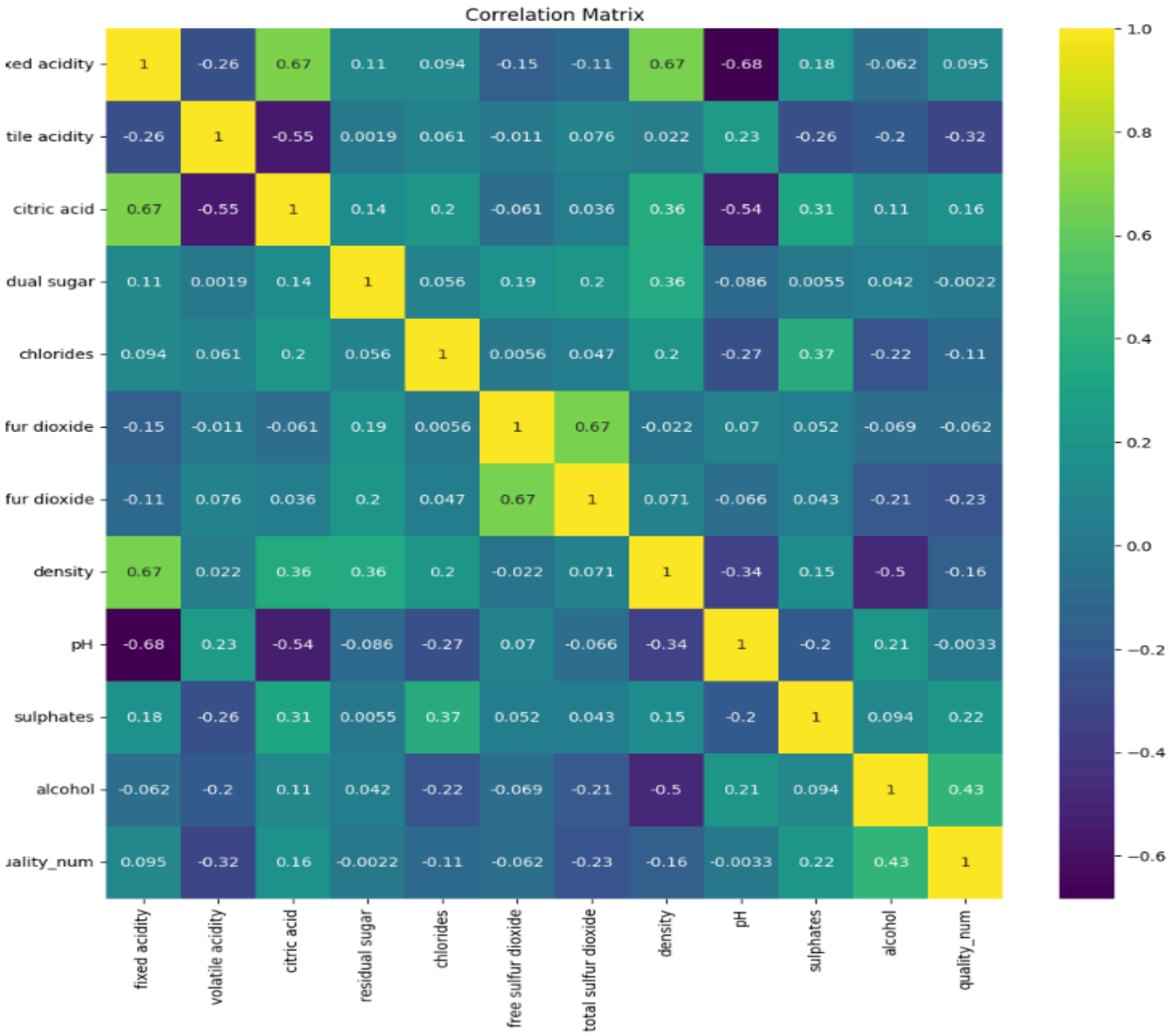
After data analysis it was found that there are no missing values present in any row and the data set is balanced is balanced so no need to drop any columns for now.

Now we need to check frequency distribution of values in data set to see if we will have to drop or change any columns



As quality only contains two variables and is of object type we change it into a numeric column to easily perform analysis on data set

```
# classifying quality as a numeric column
df['quality_num'] = df.quality.map({'bad':0,'good':1})
```



Correlation Matrix Analysis

The correlation matrix was analyzed to learn more about features and how they affect the quality of alcohol . The above graph reveals that volatile acidity has a negative correlation with quality while alcohol has a positive effect on the quality . Also the above matrix clarifies that each data is correlated to each other so random forest classification should be the preferred method for solving the dataset. The above matrix clarifies which factors affect the quality the most and should be used in feature selection in random forest ensuring random forest models robustness and interoperability

4. Data preprocessing

In this phase no imputation was needed as there was no missing values present in the data set.

```
df.isnull().sum() # searching 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
```

At first quality was a categorical variable with values like good and bad. We used label encoding to make it appropriate for model training assigning them as 0 or 1 respectively

To guarantee accurate model evaluation the data set was split into training , test and validate

```
# split data into training and test
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

# Further spliting the test set into validation and test sets
X_val, X_test1, y_val, y_test1 = train_test_split(X_test, y_test, test_size=0.5, random_state=42)
```

Scaling DataSet

StandardScaler was used to standardize the features through feature scaling. To guarantee that the test data was converted consistently with the training data, the StandardScaler was first applied to the training set and then the same transformation was applied to the test set. While not strictly required for Random Forests, this step guarantees that the features have a mean of 0 and a standard deviation of 1, which can improve the performance of some models. Crucially, in order to prevent data leaking, the scaler was only fitted to the training set.

scaling the dataset

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
X_temp = sc.fit_transform(X_temp)
X_val = sc.fit_transform(X_val)
```

Searching for best tunable parameters

The GridSearchCV module from scikit-learn was used to do a grid search across a range of hyperparameter settings in order to maximize the Random Forest Classifier's performance. Finding the optimal set of hyperparameters to maximize accuracy on the validation set was the aim of this procedure.

And the best parameter can be used for future performance testing.

```
Best Hyperparameters: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Validation Accuracy: 0.7956787802840434
```

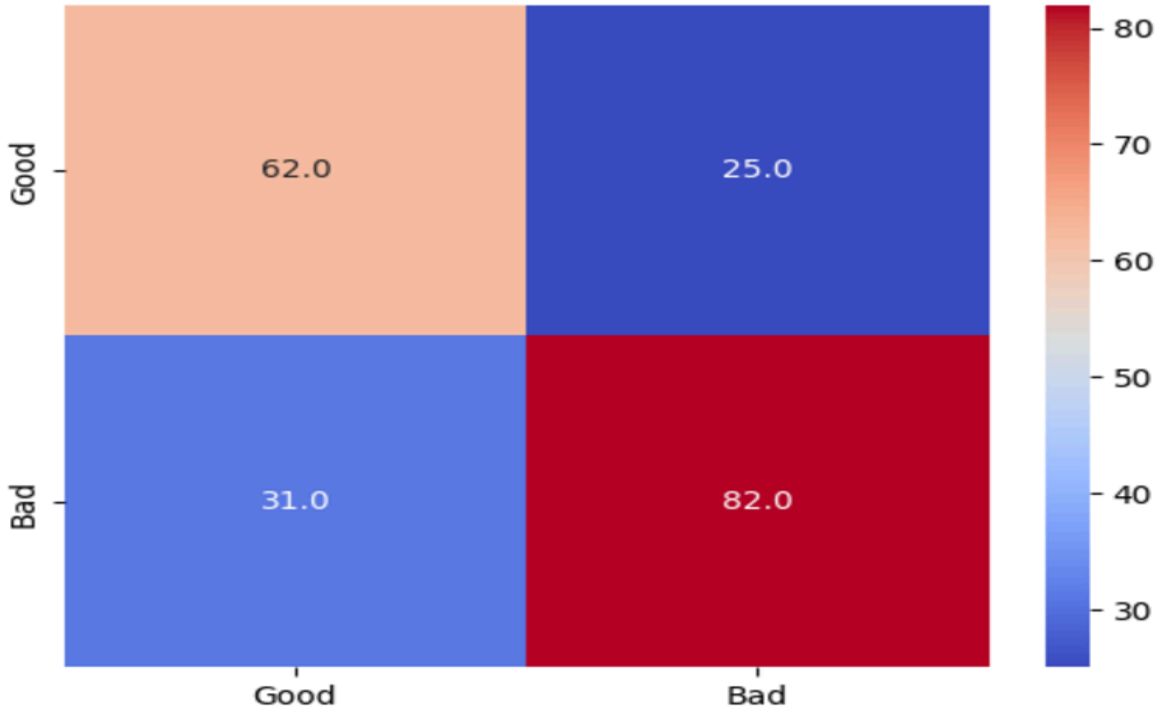
5.) Model Training and Evaluation

Ten decision trees (n_estimators=10) were used to train a Random Forest Classifier to predict the target variable. The performance of the model was assessed using the test set. The findings displayed:

Accuracy: 0.7200

ROC-AUC score : 0.8234

These outcomes show how well the Random Forest model identifies the underlying patterns in the data.

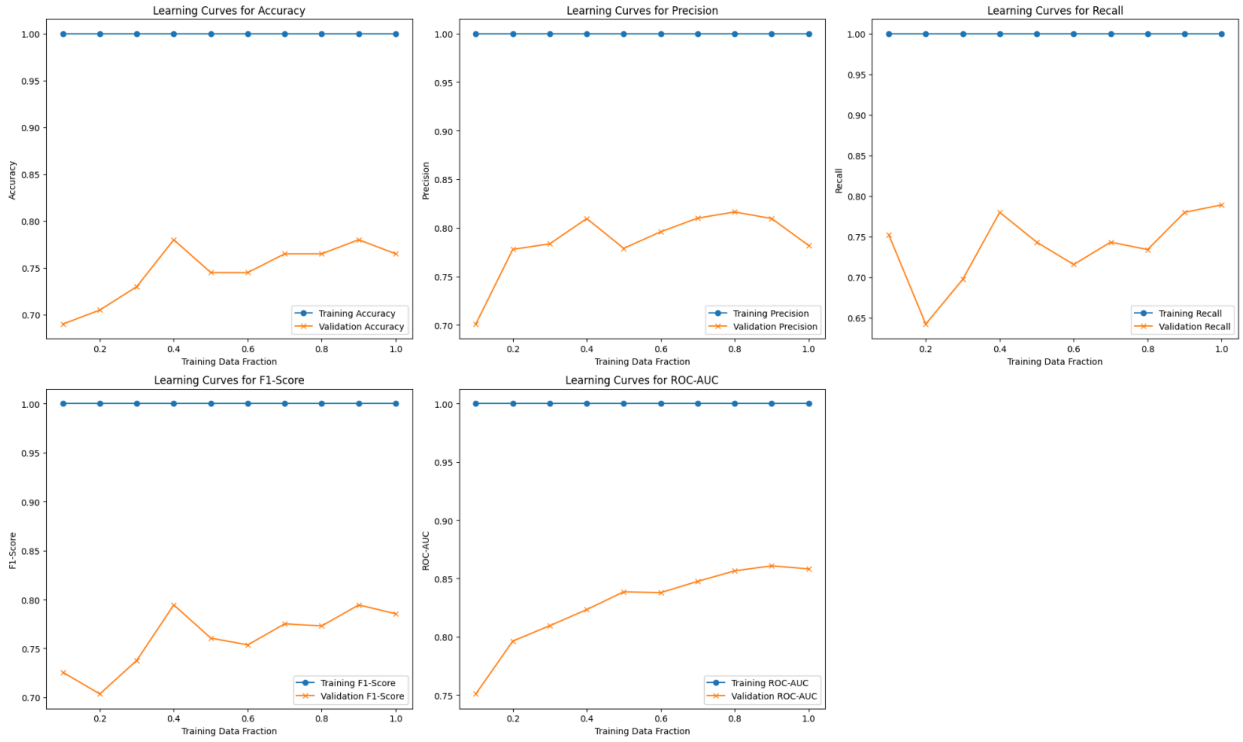


Analysis of Confusion Matrix

A thorough analysis of the model's classification performance on the test set is given by the confusion matrix. The counts of false positives, false negatives, true positives, and true negatives are shown as follows:

62 True Positives (Good projected as Good)
25 False Positives (Bad projected as Good)
82 True Negatives (Bad predicted as Bad).
31 false negatives (Good predicted as bad).

The matrix gives insight into possible areas for improvement by highlighting regions where the model misclassified and where it did well. For instance, even if the model did rather well, the number of false positives and false negatives suggests that it may be improved, maybe by adjusting the hyperparameters or adding more features.



6)Performance for varying training data sizes

The performance of a Random Forest Classifier across various percentages of training data is assessed by the line graph above. For both the training and validation datasets, these metrics—accuracy, precision, recall, F1-score, and ROC-AUC—are shown.

Metrics for training:

The training score is constant at 1.0 for all training data fractions for all performance metrics (accuracy, precision, recall, F1-score, and ROC-AUC). This shows that the model matches the training data perfectly, which may indicate overfitting because it probably captures all of the patterns in the training set.

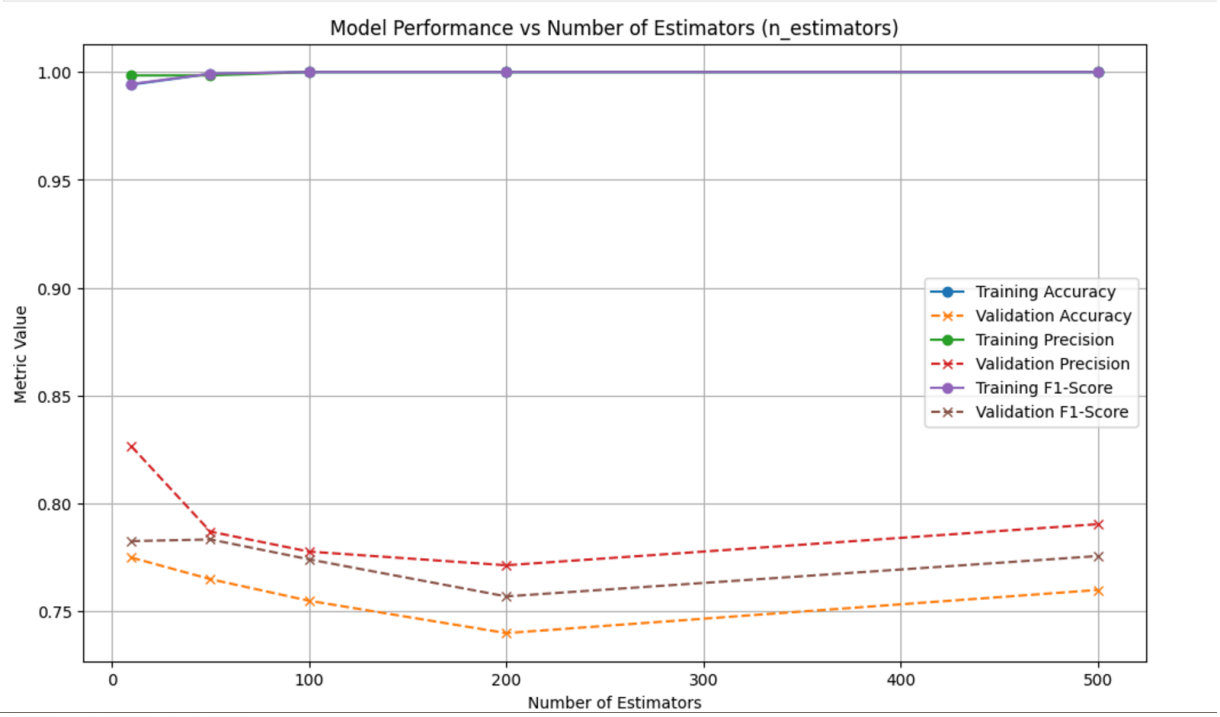
Metrics for Validation:

Validation measures, in contrast to training metrics, exhibit variability that rises with the amount of training data. At first, there is an upward trend in the validation accuracy, precision, recall, and F1-score, indicating that the model gets better with additional data. These indicators do, however, eventually plateau or even modestly decline, highlighting the limitations of the model's ability to generalize to new data. However, as training data increases, the ROC-AUC score exhibits a more steady upward trend, indicating better class discrimination in the model.

Insights:

Potential overfitting is indicated by the discrepancy between training and validation measures. The disparity between training and validation metrics shows how poorly the model generalizes to the validation data, even while it performs flawlessly on the training set.
Validation performance stabilizes as training data volume rises, indicating that increasing data may somewhat reduce overfitting.

Overfitting: This situation where any given model is performing too well on the training data but the performance drops significantly over the test set is called an overfitting model.



7)Model Performance vs Number of estimators

Observations-

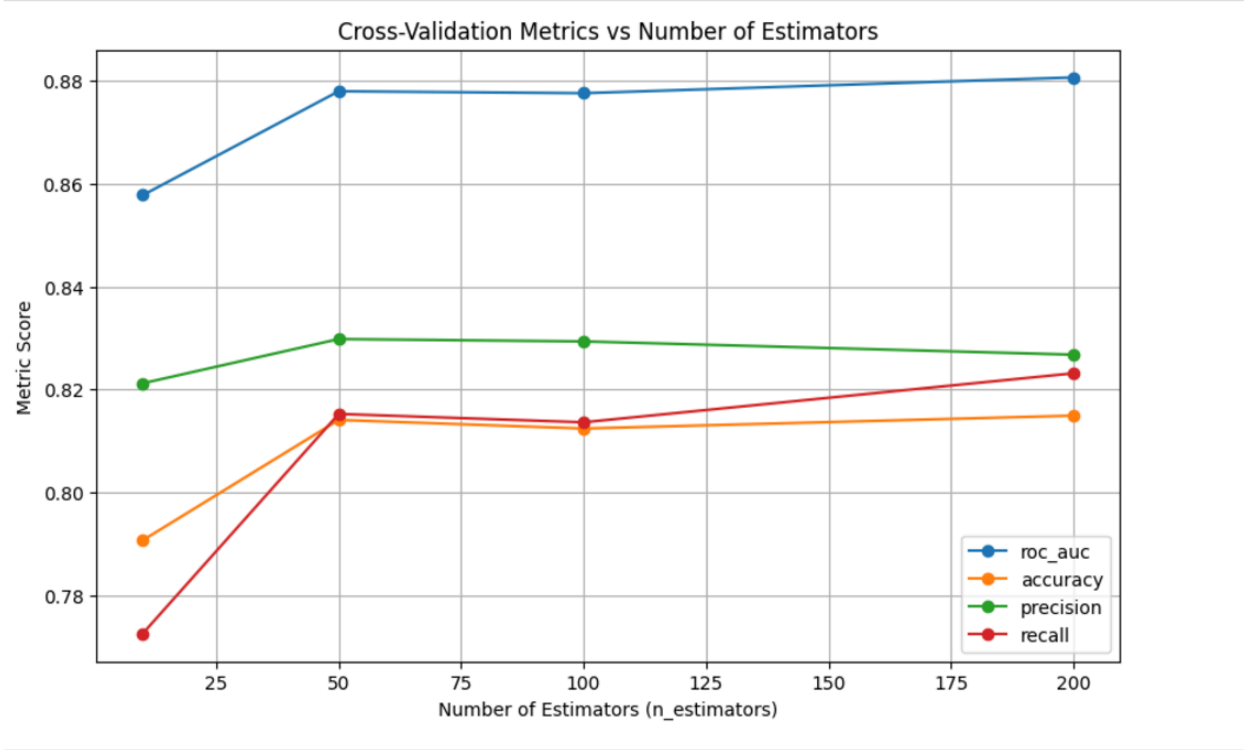
Training Data-

Training Accuracy : high and around 1.0 all the time, showing that the model does a great job of fitting the training data for all estimator counts.
Training F1-Score & Precision :Very high as well, with a trend that is comparable to the training accuracy

Validation Data-

Validation Accuracy : reduced accuracy compared to training, indicating Overfitting. With a minor upward trend after about 200 estimators, it stays comparatively constant as the number of estimators increases.
Validation F1-Score & Precision :Compared to their training counterparts, both display lower values, which first decline as the number of estimators rises and then stabilize at about 200 estimators. These measurements show that the model's generalization performance is lacking.

This implies that although the model exhibits remarkable performance on training data, it has limited generalization to unobserved data. In order to enhance the balance between training and validation performance, methods such as regularization, cross-validation, or hyperparameter tuning could be investigated.



8)Cross validation metric vs Number of Estimators

Strong performance is shown by the Random Forest model, especially when it comes to ROC AUC, which continuously surpasses 0.86 and stabilizes at 0.88 as the number of estimators increases. Precision stays constant at about 0.83, and accuracy and recall likewise gradually increase to about 0.82 and 0.81,

respectively. These findings suggest that adding more estimators improves the model's recall (the ability to recognize true positives) without appreciably lowering the accuracy (the ability to identify false positives). The majority of measures peak after 100 estimators, indicating a decline in the ratio of computational expenses to performance gains. As a result, 100 estimators might be the best ratio for efficiency and performance. By investigating feature importance and adjusting other hyperparameters, more improvements could be made.

9) Future direction- feature engineering

Feature engineering approaches can be used to significantly increase the precision and resilience of wine quality predictions. The predictive power of machine learning models, such as Random Forest classifiers, can be greatly increased by identifying new features and improving those that already exist. Potential future paths for applying feature engineering to wine quality assessments are listed below:

Features that Interact: like alcohol content and residual sugar or pH and fixed acidity, may reveal non-linear interactions that affect wine quality. Models may be able to learn more intricate relationships if these interaction terms are included in the dataset.

Feature Selection and Dimensionality Reduction: The most influential characteristics can be found by employing sophisticated feature selection methods like model-based importance ranking or recursive feature elimination (RFE). This method would reduce data noise while increasing computing efficiency and model interpretability.

Explainability and Interpretability: Another goal of feature engineering is to produce features that are easy to understand, such as how particular chemical characteristics affect the quality of wine. Those involved in the sector would especially benefit from this.

10)Conclusion

The Random Forest Classifier demonstrated its efficacy in managing intricate data interactions by correctly predicting wine quality with a 72% accuracy rate and a ROC-AUC score of 0.8234. Important characteristics like alcohol and volatile acidity were found through exploratory data analysis, which affected quality forecasts. A little discrepancy in validation measures suggested overfitting, even if the model operated flawlessly on training data, underscoring the need for additional improvement. Performance and computational efficiency were balanced via hyperparameter adjustment, especially with 100 estimators.

To overcome constraints, future developments can examine ensemble models like Gradient Boosting, regularization strategies, or sophisticated feature engineering. Accuracy and generalization could be improved by using automated hyperparameter tuning or by growing the dataset. All things considered, the Random Forest model offered a solid basis with room for improvement to optimize predictive performance.

11.) References

- https://journals.sagepub.com/doi/10.1177/1536867X20909688
- https://medium.com/analytics-vidhya/correlation-matrix-5e764bcee34
- https://aws.amazon.com/what-is/overfitting/#:~:text=Overfitting%20occurs%20when%20the%20model,to%20several%20reasons%2C%20such%20as%3A&text=The%20training%20data%20size%20is,all%20possible%20input%20data%20values.

BY - Hamza Jameel Hashmi
2649918