Predicting Red Wine Quality with Random Forests Regressor

Name: Dharanee Dharan Kanagaraj
GitHub Link: https://github.com/Dharaneephort/Machine-Learning-Dharanee

1. Introduction to the Model and Dataset

Overview of the Dataset:

Imagine the perfect glass of red wine, its unique blend of acidity, alcohol, and sugar content. Now, picture a powerful machine learning model that can predict the quality of this wine on a scale from 0 to 10 based on these very features! This dataset is like a winemaker's dream come true, filled with essential physicochemical properties of red wine, paired with a quality rating. It's more than just data it's a journey into the world of wine, where the goal is to predict its quality with precision and expertise. Cheers to the science of wine and machine learning coming together to make every sip count.

Illustration of Random Forest:

Random Forest is a robust and versatile machine learning algorithm that belongs to the family of ensemble methods. At its core, it operates by combining multiple decision trees to improve prediction accuracy and reduce the risk of overfitting. The power of Random Forest lies in its ability to aggregate predictions from several models, each trained on different subsets of the data, which leads to stronger generalization and more reliable predictions.

How Random Forest Works:

1. **Ensemble of Decision Trees:**

    - A Random Forest consists of many decision trees, where each tree operates independently. Think of each tree as a separate expert, with its own way of looking at the data.
    - While a single decision tree can be prone to overfitting, especially when it's deep and complex, Random Forest combats this by averaging or voting on the results from many trees, which typically results in better accuracy and stability.

2. **Bootstrapped Sampling (Bagging):**

    - To create diversity among the trees, Random Forest uses a technique called **bootstrapping** (or **bagging**), where each tree is trained on a different random subset of the data.
    - This means that some data points are repeated across trees, and others might not appear at all. The variation in the training sets helps the model generalize better and avoid overfitting to any particular subset of data.

3. **Random Feature Selection at Each Split:**

    - Another important feature of Random Forest is that at each node (decision point) in the tree, it evaluates a **random subset** of features rather than considering all features at once. This randomness encourages diversity among the trees and prevents them from becoming too similar to each other, thus reducing the model's tendency to overfit.
    - This process makes Random Forest especially powerful, as it can capture non-linear relationships and interactions between features that might be missed by a single decision tree.

4. **Prediction Aggregation:**

    After each tree in the forest makes its prediction, the Random Forest algorithm aggregates the results:

    - **For regression tasks** (like predicting wine quality), the model averages the predictions from all the individual trees to give a final result. This reduces the variance of the prediction and typically improves accuracy.
    - **For classification tasks** (e.g., determining whether a wine is good or bad), the Random Forest uses majority voting to determine the final class. The class predicted by the most trees becomes the final output.

2. Data Pre-processing

Before training a machine learning model, it's crucial to pre-process the data. Here are the key steps taken in this project:

Handling Missing Data:

Missing values in the dataset are handled by filling them with the median value of the respective column. This ensures that no rows or features are lost during model training due to missing values.

```
# Load the dataset
wine_data = pd.read_csv('winequality-red.csv', sep=';')
wine_data = wine_data.fillna(wine_data.median())
```

Scaling Features with Standard Scaler:

Wine properties like pH, alcohol, and sugar vary across different ranges. Standard Scaler ensures that all features have a mean of 0 and standard deviation of 1, making the data suitable for machine learning models.

```
# Standardize the features using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

This ensures that all features are on the same scale, making them comparable when using models like logistic regression or neural networks, which require normalized input.

Data Splitting:

The dataset is split into **training** and **testing** sets. The training set is used to train the model, and the testing set is used to evaluate the model's performance.

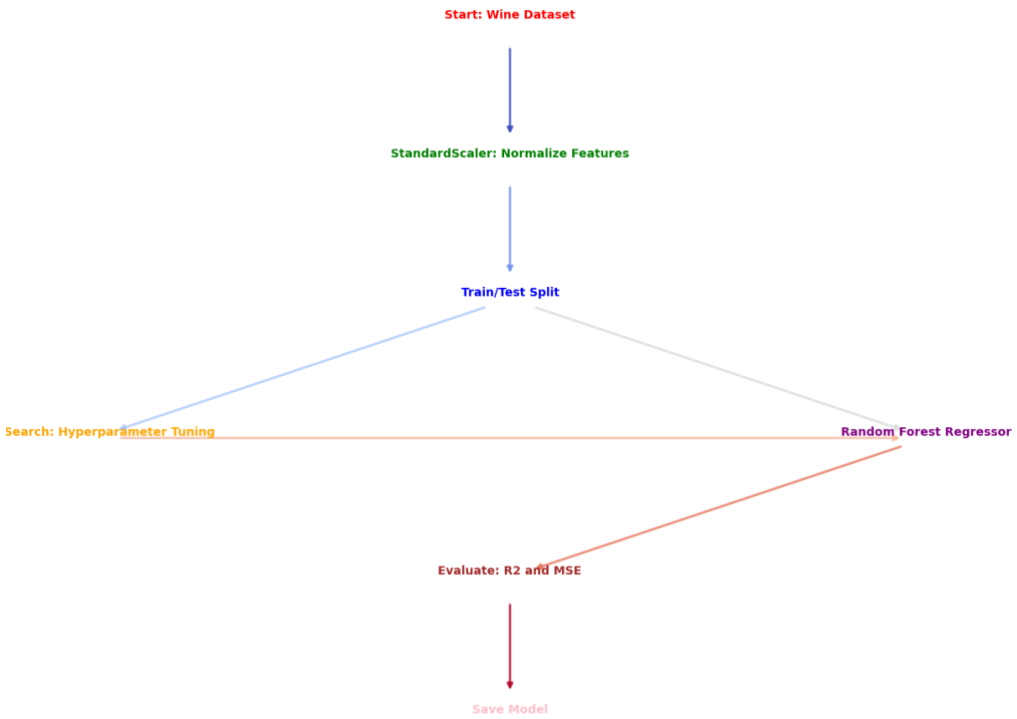
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.1, random_state=5)
```

3. Model Selection: Random Forest Regressor

The core of this project is based on the Random Forest Regressor. Random Forest is an ensemble learning method where multiple decision trees are trained on different random subsets of the data. The final prediction is made by averaging the predictions from all individual trees.

Why Random Forest?

- **Non-linear Relationships**: Random Forest handles non-linear relationships between features effectively.
- **Robust to Overfitting**: Since the model aggregates predictions from multiple trees, it is less likely to overfit than a single decision tree.
- **Feature Importance**: Random Forest automatically calculates feature importance, which can help identify which features have the most impact on the predicted quality.



4. Hyperparameter Tuning: Grid Search CV

To enhance the performance of the Random Forest model, Grid Search CV is employed to explore various hyperparameter combinations and identify the optimal set. The hyperparameters being tuned include:

- **n_estimators**: The number of trees in the forest. Increasing the number of trees generally improves performance, though it also increases computational complexity.
- **max_depth**: The maximum depth of each tree. Restricting the depth helps to prevent overfitting.
- **min_samples_split**: The minimum number of samples required to split an internal node. Larger values prevent the creation of excessively deep trees.
- **min_samples_leaf**: The minimum number of samples required at a leaf node. Higher values help in reducing overfitting.

The grid search is carried out with 3-fold cross-validation (CV), which divides the training data into three subsets to assess model performance and reduce the risk of overfitting.

```
# Define a parameter grid for hyperparameter tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Perform Grid Search with Cross Validation
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3, scoring='r2', verbose=1, n_jobs=-1)
grid_search.fit(X_train, y_train)
```

The best combination of hyperparameters is selected based on the highest **R-squared (R^2)** value, which indicates how well the model explains the variance in the target variable.

5. Model Evaluation

R-squared (R^2):

R-squared is a metric that assesses how well the model fits the data by comparing its predictions to those of a baseline model, which simply predicts the average value of the target variable. A value of R^2 closer to 1 indicates better model performance.

Mean Squared Error (MSE):

MSE measures the average squared difference between the predicted and actual values. A lower MSE indicates better model performance.

```python
# Evaluate the model on the test set
y_pred = best_rf.predict(X_test)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

# Print results
print("Best Hyperparameters:", grid_search.best_params_)
print("R-squared on Test Set:", r2)
print("Mean Squared Error on Test Set:", mse)
```

For this project, the model achieved an **R^2** of approximately 0.53 and an **MSE** of around 0.25, which means it does a reasonable job of predicting wine quality but could be further optimized.

6. Visualizing the Results

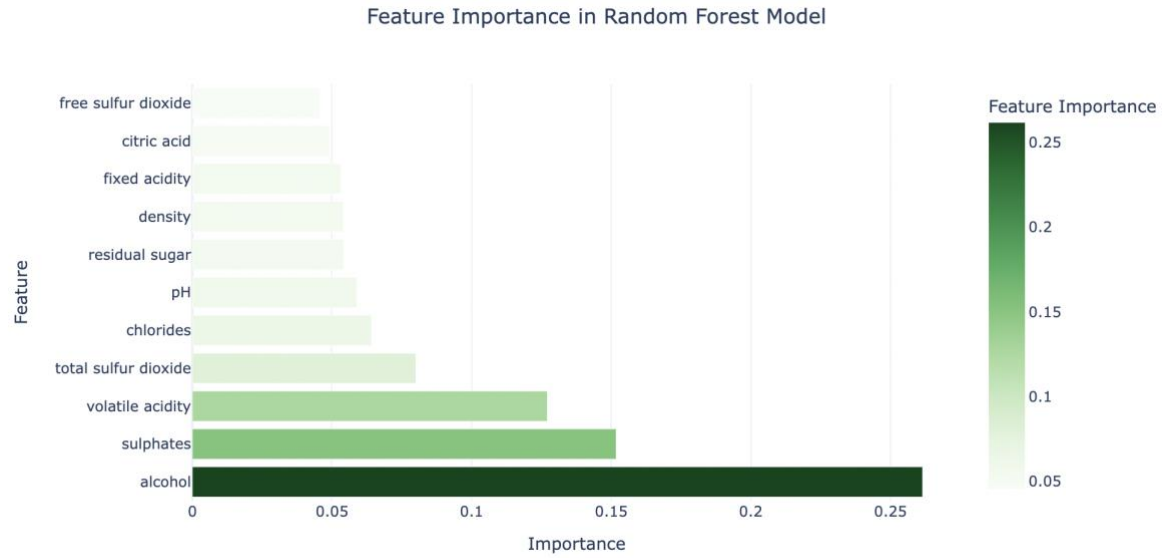
A unique advantage of Random Forest is its ability to evaluate feature importance. This is done by calculating how much each feature contributes to reducing impurity at each split across all trees in the forest.

Visualizing Feature Importance:

Using Plotly, we can visualize which features have the most impact on wine quality prediction. This helps to understand what the model has learned and which variables are most influential in determining the wine's quality.

```python
# Feature importance visualization
feature_importances = best_rf.feature_importances_
features = X.columns

# Create a DataFrame for better handling
importance_df = pd.DataFrame({
    'Feature': features,
    'Importance': feature_importances
}).sort_values(by='Importance', ascending=False)

# Create an interactive horizontal bar chart
fig = px.bar(
    importance_df,
    x='Importance',
    y='Feature',
    orientation='h',
    title='Feature Importance in Random Forest Model',
    labels={'Importance': 'Feature Importance', 'Feature': 'Feature'},
    color='Importance',
    color_continuous_scale='Greens'
)

fig.update_layout(
    xaxis=dict(title='Importance'),
    yaxis=dict(title='Feature'),
    template='plotly_white',
    title=dict(x=0.5, xanchor='center')  # Center the title
)

fig.show()
```

To better interpret the model's output, we can create visualizations of the feature importance and the results of the model's predictions. This allows stakeholders, such as winemakers or consumers, to understand the factors that most affect wine quality.



Understanding the Feature Importance Graph:

This graph visualizes the **feature importance** in the Random Forest regression model used to predict wine quality. Feature importance quantifies how much each feature contributes to the prediction model. Higher values indicate that a feature has a greater impact on the target variable (in this case, wine quality).

Key Elements of the Graph

1. **X-axis (Importance)**:

   The **X-axis** of the graph represents the importance score of each feature, which is derived from how effectively a feature reduces impurity (variance) in the decision trees of the Random Forest model. Features with higher values on this axis play a more significant role in predicting wine quality

2. **Y-axis (Feature)**:

   The **Y-axis,** on the other hand, lists the 11 physicochemical properties of wine used as input features in the model. Among these, **alcohol** stands out as the most important feature with the highest importance score, followed by **sulphates.** Less significant features include properties like chlorides, pH, and citric acid.

3. **Colour Scale**:

   Additionally, the graph employs a **colour scale** to visually represent the magnitude of feature importance. The green colour gradient used in the graph varies in

intensity, where darker shades correspond to features with higher importance, and lighter shades represent features with lesser importance.

Interpretation of the Graph

Alcohol (Highest Importance):

Alcohol content has the greatest impact on wine quality. This is likely because alcohol strongly influences the sensory perception and body of the wine, which are key factors in determining quality.

Sulphates and Volatile Acidity:

- **Sulphates**: Contribute significantly to preservation and taste, making them highly relevant.
- **Volatile Acidity**: Affects the sourness of wine. Too much can negatively impact the quality.

Middle-Importance Features:

- **Total Sulphur Dioxide**: Important for wine preservation, but less impactful than alcohol or sulphates.
- **Density and Residual Sugar**: Correlate with the mouthfeel and sweetness of the wine, but their effect on quality is moderate.

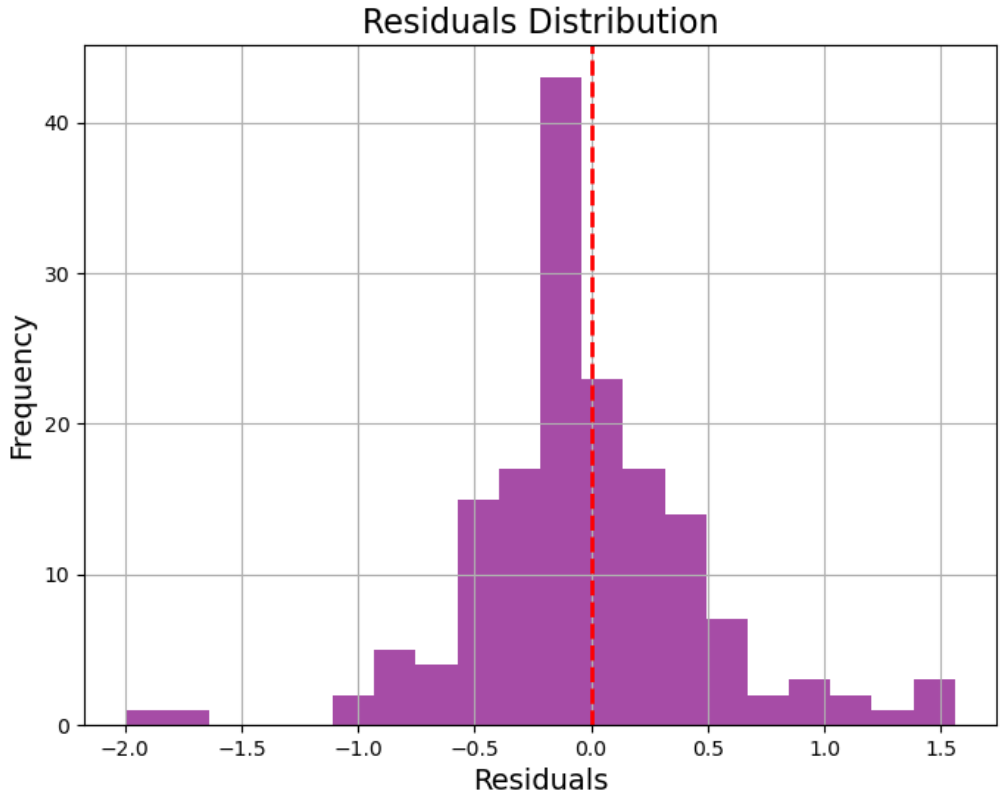
Low-Importance Features:

Free Sulphur Dioxide, **Citric Acid**, **and Fixed Acidity**: Have minor contributions to predicting wine quality, as their variations might not significantly affect taste or stability.

Why This Matters?

- **Optimizing Production**: Winemakers can focus on controlling key factors like alcohol and sulphates during production.
- **Feature Selection**: For future models, less important features can be removed to simplify the model without losing much predictive power.
- **Insights into Wine Chemistry**: This analysis provides a data-driven understanding of which chemical properties most influence quality.

Residuals Distribution:



What Does the Residuals Histogram Show?

- **X-axis (Residuals):** The difference between the actual wine quality and the predicted wine quality, calculated as:

  Residual=Actual Quality−Predicted Quality

- **Y-axis (Frequency):** The number of predictions that fall within each bin (range) of residual values.
- **Negative values:** Indicate that the model over-predicted the wine quality.
- **Positive values:** Indicate that the model under-predicted the wine quality.
- **Zero residual:** Indicates a perfect prediction.

Observations:

1. **Centred Around Zero:**

   - The majority of residuals are concentrated around 0, which indicates that the model generally predicts wine quality with minimal bias.
   - This is a good sign because it shows the model isn't systematically over-predicting or under-predicting.

2. **Symmetrical Shape:**
   - The histogram has a roughly symmetric distribution around 0, which suggests that the errors are evenly distributed across the dataset.
   - This symmetry is desirable in a regression model because it indicates that predictions are not skewed toward over-prediction or under-prediction.

3. **Narrow Spread:**

   - Most residuals are within the range of −1-1−1 to 111, with only a few outliers beyond these values.
   - This narrow spread indicates that the model is generally accurate, with small prediction errors for most instances.

4. **Red Dashed Line at Zero:**

   - This vertical line at Residual=0 serves as a reference to show where predictions would perfectly match the actual wine quality.
   - The fact that most residuals are close to this line confirms that the model performs well overall.

7. Model Deployment and Prediction on New Data

Once the model is trained, it can be deployed for predicting wine quality based on new input data. In this case, the features provided for the prediction must match the ones used in the training data.

```
# Load the saved model
model = joblib.load('random_forest_wine_quality_scaled.pkl')

# Load the scaler
scaler = StandardScaler()

# Example input data for prediction
new_data = pd.DataFrame({
    'fixed acidity': [7.4],
    'volatile acidity': [0.7],
    'citric acid': [0.0],
    'residual sugar': [1.9],
    'chlorides': [0.076],
    'free sulfur dioxide': [11.0],
    'total sulfur dioxide': [34.0],
    'density': [0.9978],
    'pH': [3.51],
    'sulphates': [0.56],
    'alcohol': [9.4]
})

# Scale the input data
scaled_data = scaler.fit_transform(new_data)  # Use `scaler.transform` if the same scaler was saved

# Make predictions
predictions = model.predict(scaled_data)

# Output the prediction
print("Predicted Wine Quality:", predictions[0])
```

This data is then scaled using the same scaler used during training, and the model makes a prediction. The model might predict a wine quality of **5.9** for this input, which can be rounded to **6**, representing the expected quality of the wine.

8. Conclusion and Future Work

In conclusion, this project demonstrates the application of Random Forest in predicting the quality of red wines based on various chemical features. The model provides valuable insights into which factors are most important in determining wine quality. Although the model performs reasonably well, there is room for improvement. Future work might include:

- **Feature Engineering**: Adding new features such as geographic region or grape variety.
- **Model Assembling**: Combining the Random Forest model with other machine learning models for enhanced performance.
- **Tuning the Model Further**: Trying different approaches for hyperparameter tuning, such as Randomized Search CV.

The application of machine learning models to predict wine quality not only helps automate the process but can also provide useful insights to winemakers in improving their products.

Final Thoughts

This project exemplifies how machine learning, particularly ensemble methods like Random Forest, can be applied to real-world tasks such as predicting wine quality. The project combines essential elements of data pre-processing, model training, hyperparameter tuning, evaluation, and deployment, making it a comprehensive example of using Random Forest in predictive analytics.

References :

Random Forest Regression in Python
https://www.geeksforgeeks.org/random-forest-regression-in-python/

Random Forest Regression — How it Helps in Predictive Analytics?
https://medium.com/@byanalytixlabs/random-forest-regression-how-it-helps-in-predictive-analytics-01c31897c1d4