Supervised Learning

AUTHOR
Nandini Kodali

Introduction

Supervised Learning is a branch of Machine Learning where models are trained on labeled data. The primary goal of supervised learning is to learn a mapping function that can predict outputs for unseen data accurately. This approach relies on historical data to identify patterns and relationships, enabling it to generalize well on new, unseen data.

Process:

- The model is provided with input-output pairs from historical data. Each data point consists of:
  - Features (X): Independent variables or predictors.
  - Labels (Y): Target outputs or dependent variables.
- The model learns the relationship between the input features and corresponding outputs using a loss function that measures prediction error. The goal is to minimize this error.
- Once trained, the model can predict outputs for new inputs.
- Performance is evaluated using metrics like accuracy, precision, recall, F1-score for classification tasks, or RMSE, MAE for regression tasks.

Types of Supervised Learning

1. Classification:

- The goal is to categorize inputs into discrete classes or categories.
- Algorithms: Logistic Regression, Support Vector Machines (SVM), Decision Trees, Random Forest, Naive Bayes, K-Nearest Neighbors (KNN).

2. Regression:

- The goal is to predict a continuous output based on input features.
- Algorithms: Linear Regression, Polynomial Regression, Ridge and Lasso Regression, Support Vector Regression (SVR).

Challenges:

- Supervised learning requires large volumes of labeled data, which can be expensive and time-consuming to obtain.
- Models may memorize training data instead of generalizing to new inputs. Regularization and cross-validation techniques help mitigate this.

- In classification problems, imbalanced classes can affect performance.

In this project:

Supervised Learning Algorithms are applied to classify race results and predict pit stop duration.

Required Libraries

▶ Code

Feature Selection

Feature Selection is the process of identifying the most relevant features from a dataset that contribute significantly to predicting the target variable. By reducing the number of features, it improves computational efficiency during training, minimizes the risk of overfitting, and enhances model performance. Additionally, reducing feature complexity simplifies models, making them easier to interpret and explain, which is crucial for understanding the underlying data relationships.

▶ Code

▶ Code

	Year	Round	RaceName	DriverID	Lap1	Lap2	Lap3	Lap4	La
0	2011	1	Australian Grand Prix	alguersuari	-1.492522	-0.208201	1.339923	-0.281099	-0
1	2011	1	Australian Grand Prix	alonso	-0.414215	0.355923	1.723412	-0.281099	-0
2	2011	1	Australian Grand Prix	ambrosio	-0.218159	0.976458	-0.577525	-0.281099	-0
3	2011	1	Australian Grand Prix	barrichello	-0.316187	0.130273	0.956433	3.039918	-0
4	2011	1	Australian Grand Prix	buemi	-0.120131	0.468747	-0.577525	-0.281099	-0

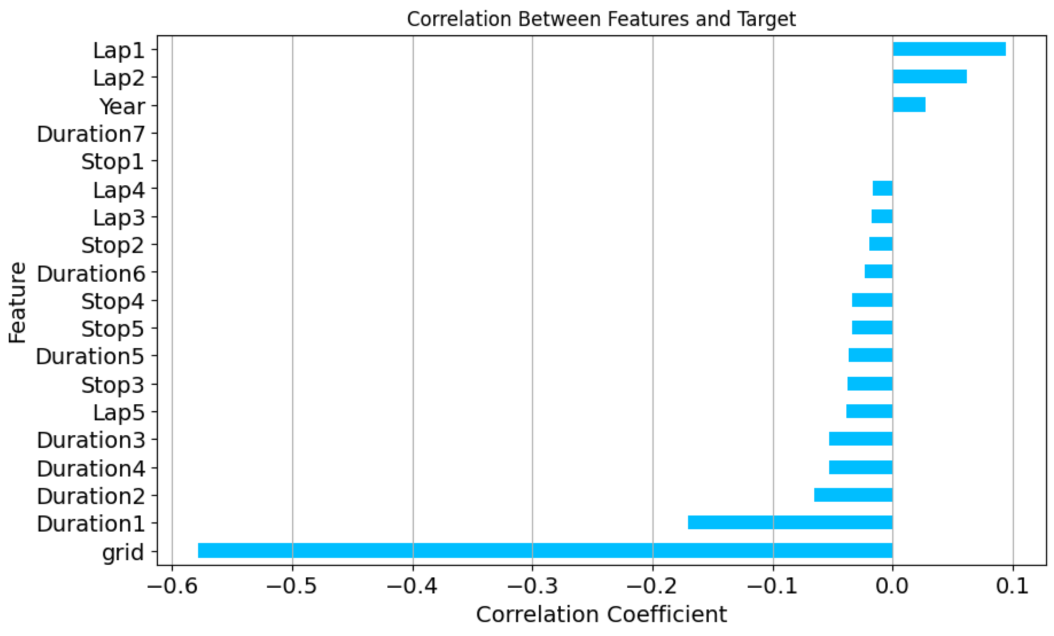
5 rows × 36 columns

▶ Code

Correlation Analysis

Correlation analysis helps identify features that are strongly correlated with the target variable.

▶ Code



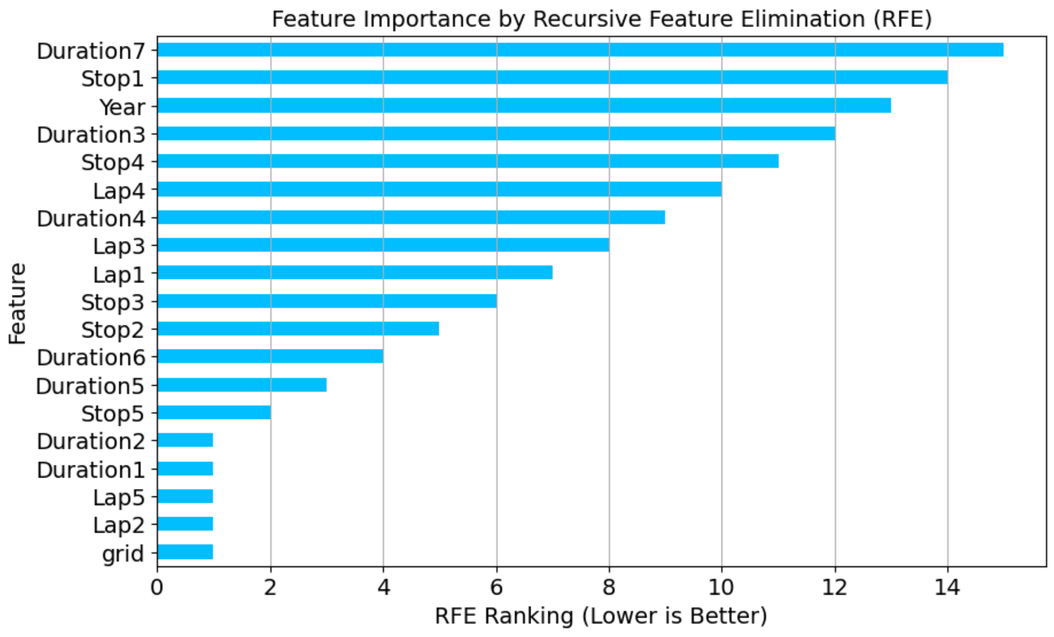
Recursive Feature Elimination

RFE is a wrapper method that iteratively removes the least important features based on a model's performance.

Process:

1. Train a machine learning model.
2. Rank features based on their importance or weights.
3. Recursively eliminate the least significant feature(s) and retrain the model.
4. Stop when the desired number of features is selected.

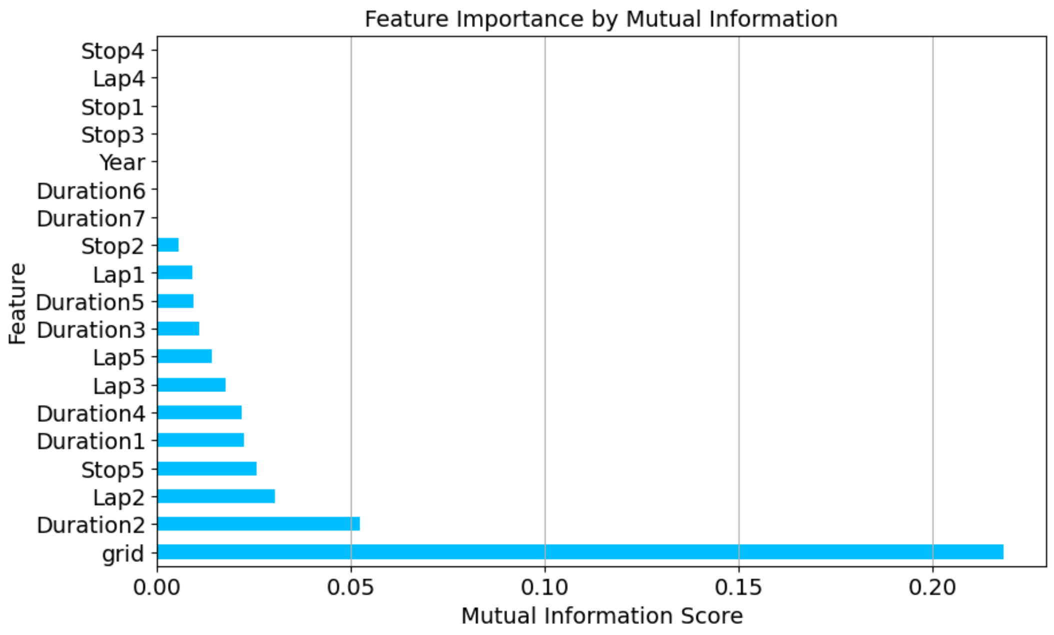
▶ Code



Mutual Information

Mutual information measures the dependency between two variables. It identifies the most informative features by quantifying how much knowing one variable reduces uncertainty about the other

▶ Code



Selected Features: `grid`,`Duration1`, `Duration2`, `Lap1`, `Lap2`, `Lap3`,`Stop2`, `Stop3`

These features have been consistenly selected as important by the Feature Selection Algorithms.

Binary Classification

Decision Trees

A Decision Tree is a supervised machine learning algorithm used for both classification and regression tasks. It operates by recursively splitting the dataset based on the most informative features, represented as a tree-like structure of decisions. Each node in the tree represents a feature, branches represent decisions or splits, and the leaves represent the final outcome.

Process:

- Decision Trees identify the best feature for splitting the data at each step. Splits are made to maximize homogeneity within the resulting subsets.
  - For classification, splitting is often based on measures like Gini Impurity or Entropy (Information Gain)
  - For regression, the split minimizes the Mean Squared Error (MSE).
- Recursive Partitioning: The algorithm continues to split the data until a stopping condition is met, such as reaching a maximum depth, minimum samples in a node, or no further gain from splitting.
- Prediction:

- In classification, predictions are based on the majority class in the leaf node.
- In regression, predictions are based on the average value in the leaf node.

Hyperparameters:

1. `max_depth`: Limits the depth of the tree. A smaller depth reduces overfitting but may underfit the data.
2. `min_samples_split`: The minimum number of samples required to split a node. Increasing this value reduces overfitting by preventing small, irrelevant splits.
3. `min_samples_leaf`: Specifies the minimum number of samples a leaf node must contain. Larger values result in smoother decision boundaries.
4. `criterion`: Determines the measure of impurity.
5. `max_features`: Limits the number of features considered at each split, reducing overfitting and computational cost.
6. `max_leaf_nodes`: Limits the number of leaf nodes in the tree

Evaluation Metrics:

- Accuracy: Measures the percentage of correctly predicted instances.
- Precision: Fraction of true positive predictions out of all positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

- Recall (Sensitivity): Fraction of actual positives correctly identified.

$$Recall = \frac{TP}{TP + FN}$$

- F1-Score: Harmonic mean of precision and recall, useful when dealing with imbalanced classes.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- Confusion Matrix: Provides a matrix of predicted vs. actual class labels, offering insights into false positives, false negatives, etc.
- ROC-AUC Score: Evaluates the trade-off between true positive rate and false positive rate.

Challenges

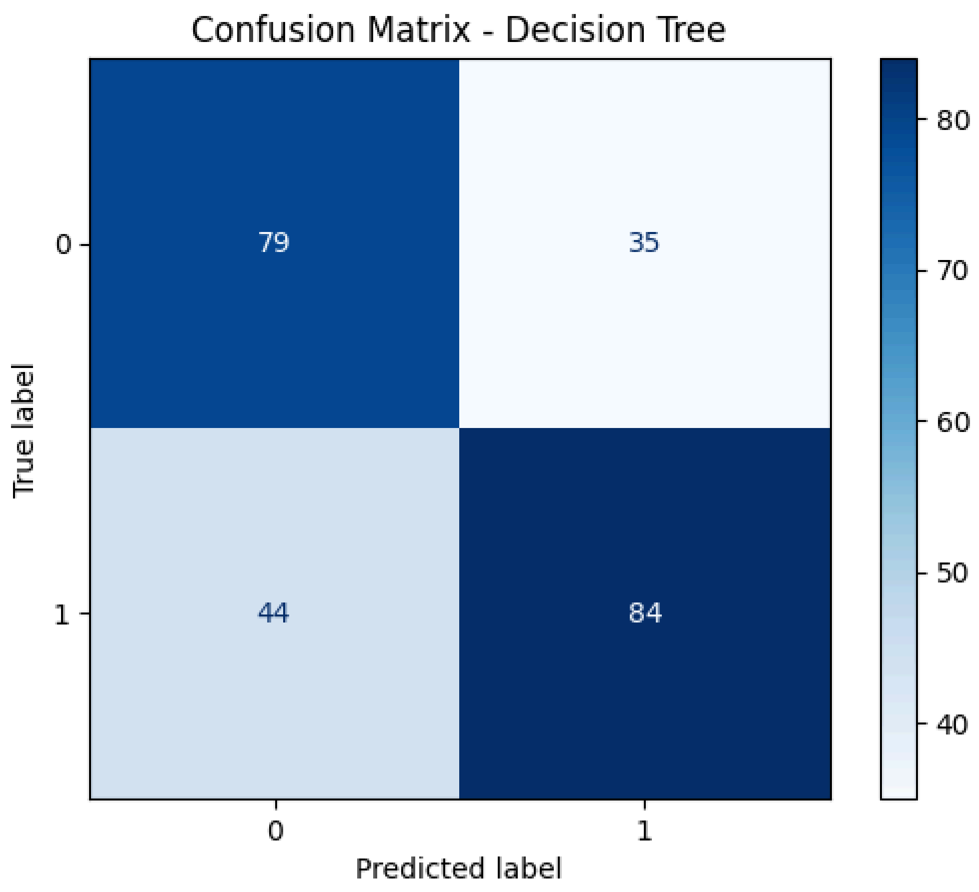
- Trees tend to overfit when they grow too deep, capturing noise in the data.
- Bias to Greedy Splits: Decision Trees use a greedy approach, which may lead to suboptimal splits.
- Instability: Small changes in the data can result in significant changes in the tree structure

▶ Code

▶ Code

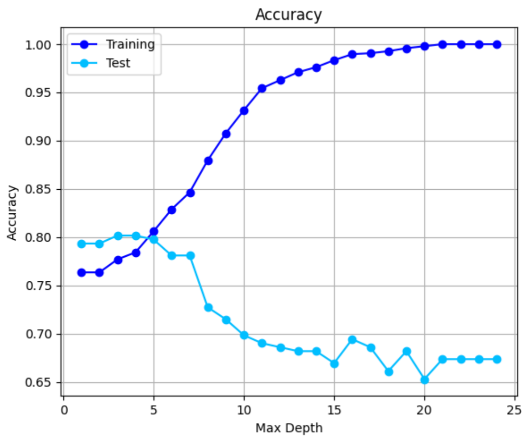
▶ Code

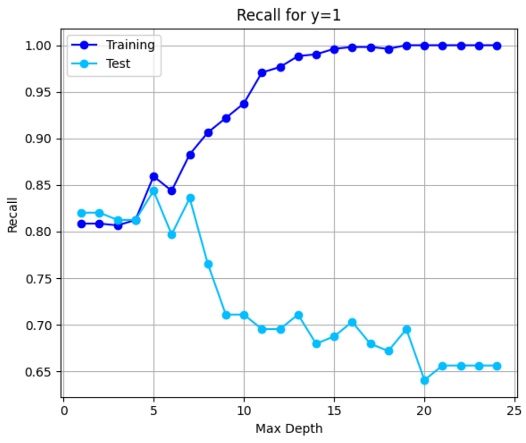
```
Classification Report - Decision Tree:
              precision    recall  f1-score   support

           0       0.64      0.69      0.67       114
           1       0.71      0.66      0.68       128

    accuracy                           0.67       242
   macro avg       0.67      0.67      0.67       242
weighted avg       0.68      0.67      0.67       242
```

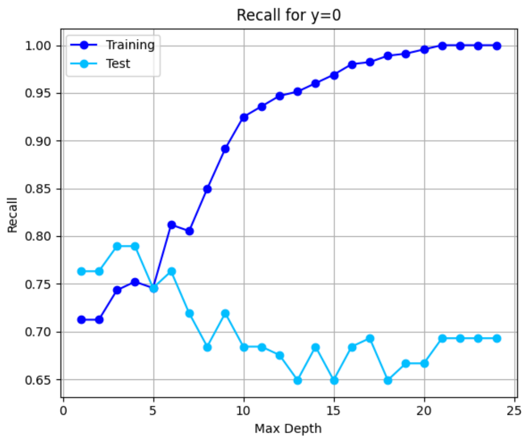


- The model performs similarly for both classes, indicated by similar recall, and f-1 scores.
- 67% accuracy indcates that the model can be imporoved further with hyperparameter tuning.
- There is no significant class imbalance.

▶ Code



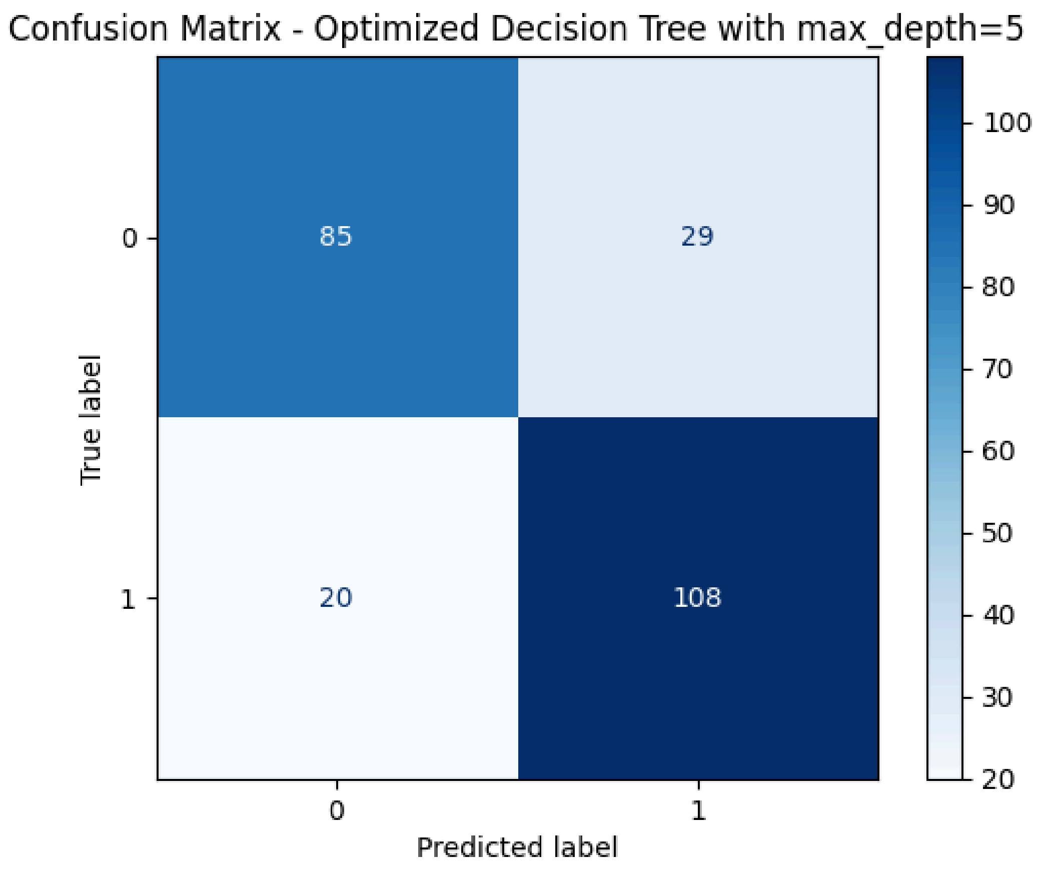




As the depth of the tree increases, the training accuracy of the model increases and the test accuracy of the model decreases. This is because the model starts to overfit the training data. To avoid overfitting of the model, we select the optimal number of layers when the test and training accuracy are high. In this case, we select the **max depth to be 5**. Beyond 5, the test accruacy significantly drops.

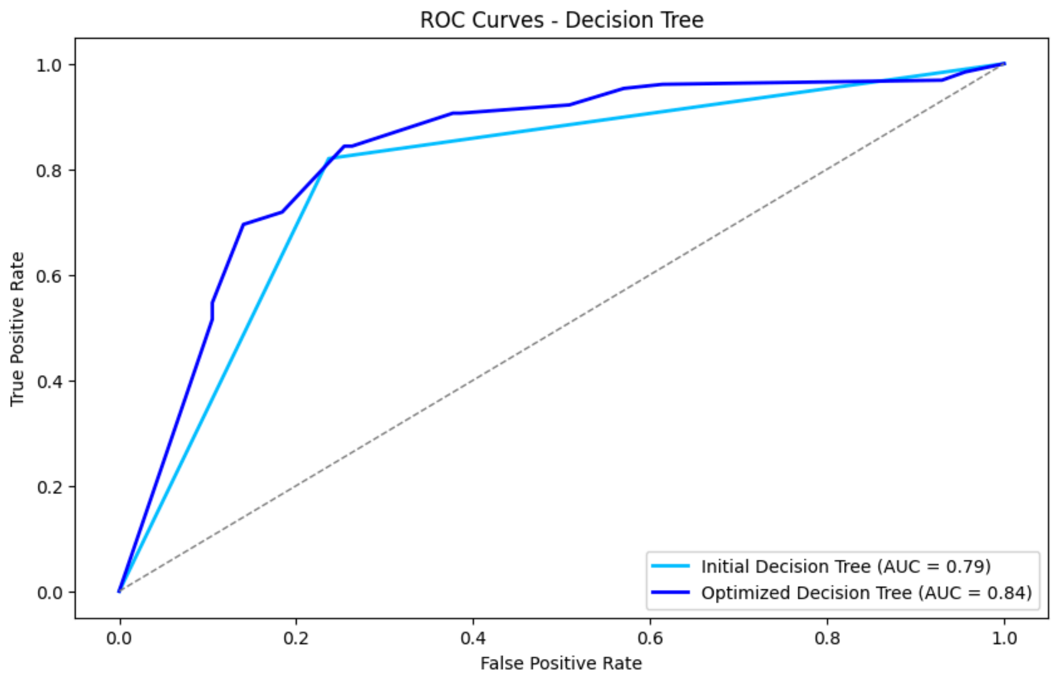
▶ Code

```
Best Parameters: {'min_samples_leaf': 2, 'min_samples_split': 5}
Classification Report - Optimized Decision Tree with max_depth=5:
              precision    recall  f1-score   support

           0       0.81      0.75      0.78       114
           1       0.79      0.84      0.82       128

    accuracy                           0.80       242
   macro avg       0.80      0.79      0.80       242
weighted avg       0.80      0.80      0.80       242
```



- After hyperparameter tuning, we observe a significant increase in accuracy.
- The precision for the turned model is also more balanced than the original model.

▶ Code



- **The Receiver Operating Characteristic (ROC)** curve is used to evaluate the trade-off between True Positive Rate (TPR) (sensitivity/recall) and False Positive Rate (FPR) across different threshold values.
- **Area Under The Curve (AUC)** represents the measure of separability. Higher the AUC, the better is the model at predicting the classes.

- The optimized Decision Tree outperforms the initial model, as evidenced by the higher AUC (0.84 vs. 0.79). Hyperparameter tuning has led to improvements in the model's performance.

Random Forest

Random Forests combine the strengths of multiple decision trees to deliver a robust, versatile, and high-performing model for classification and regression tasks.

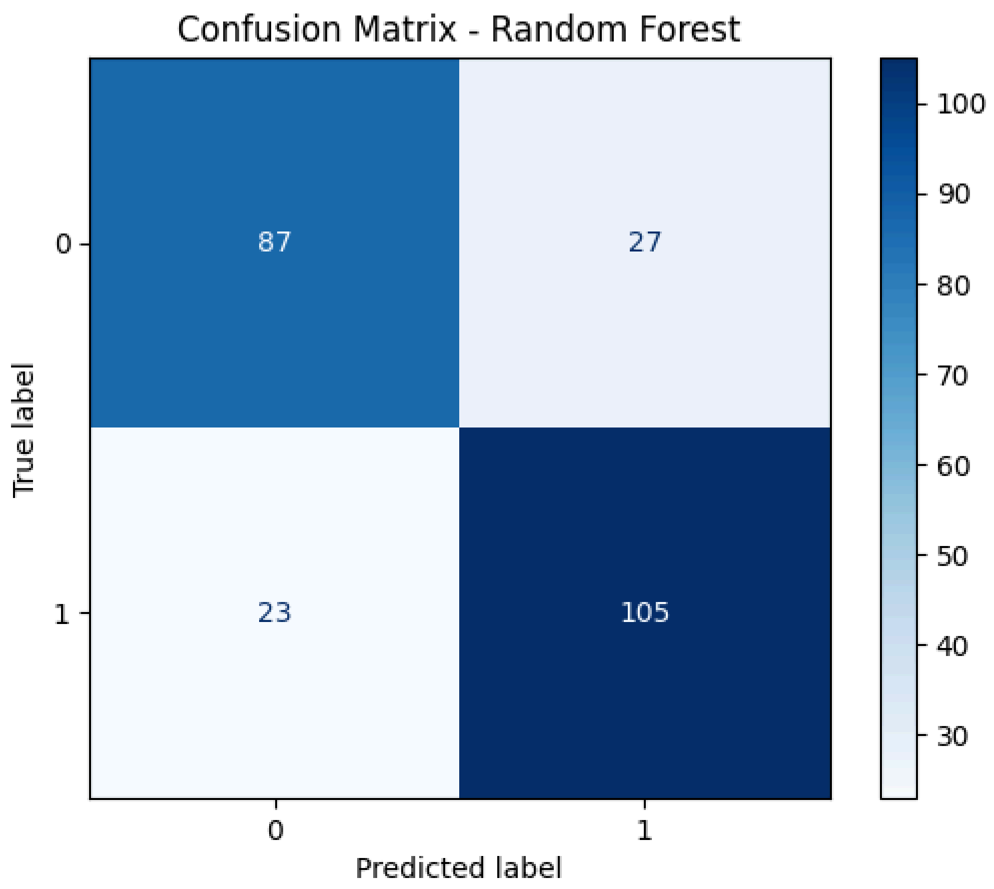
Process:

- Random Forest builds multiple decision trees using bootstrap aggregation, where each tree is trained on a bootstrapped subset of the data.
- At each node, a random subset of features is considered for splitting, introducing diversity among the trees and reducing overfitting.
- Each tree is grown independently to its maximum depth or based on stopping criteria like max_depth or min_samples_split.
- For classification tasks, the final prediction is determined through a majority vote from all trees. For regression tasks, the final output is obtained by averaging the predictions of all trees.
- By combining outputs from multiple decision trees, Random Forest reduces variance, improves generalization, and enhances model robustness.

▶ Code

▶ Code

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

           0       0.79      0.76      0.78       114
           1       0.80      0.82      0.81       128

    accuracy                           0.79       242
   macro avg       0.79      0.79      0.79       242
weighted avg       0.79      0.79      0.79       242
```



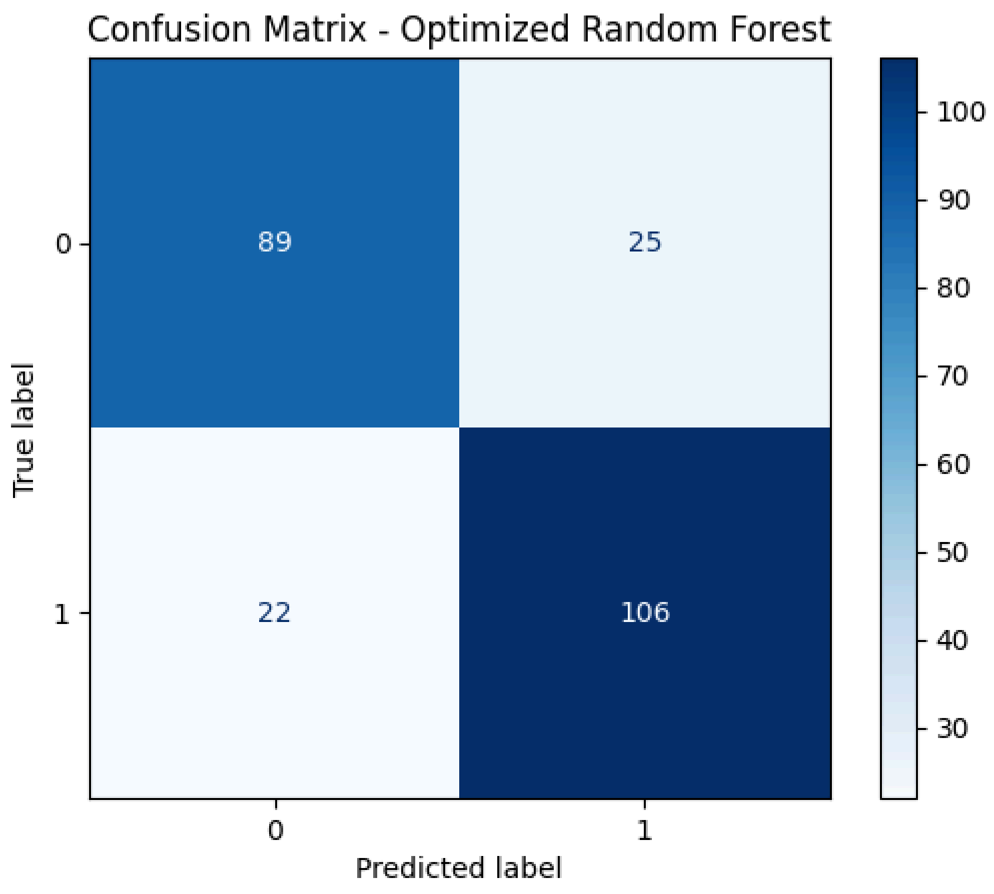
```
Accuracy: 0.7933884297520661
```

The Random Forest Classifier, using its default hyperparameters, achieved an accuracy of approximately 0.80. Comparable to a Decision Tree after hyperparameter tuning, this highlights the superior performance and robustness of Random Forests, especially when handling large datasets, as they effectively reduce overfitting compared to a single decision tree.

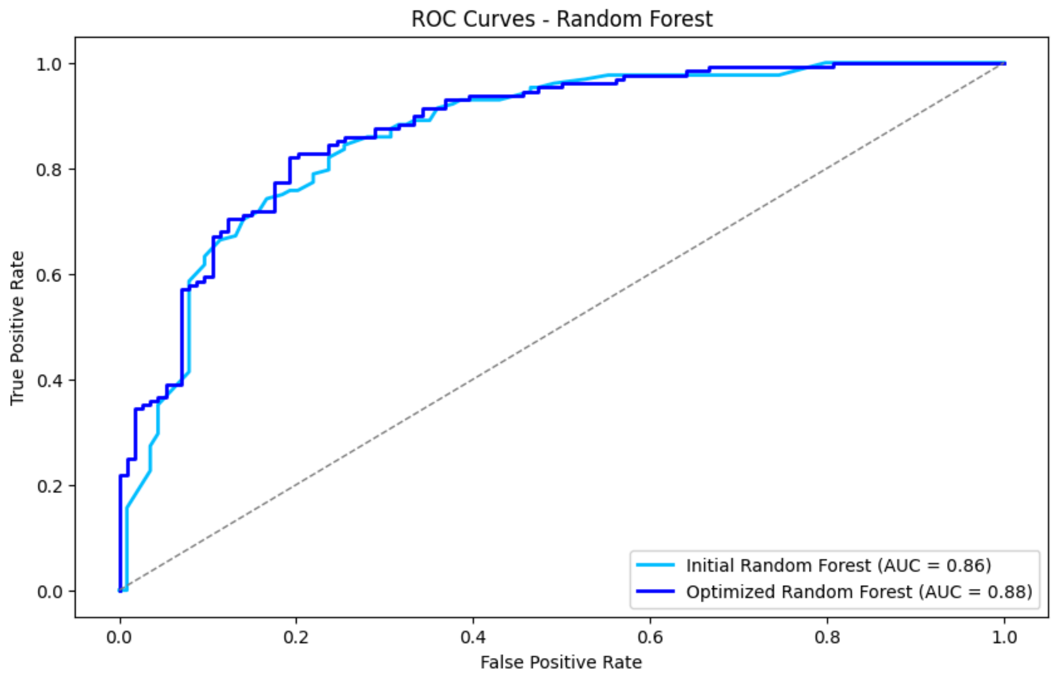
▶ Code

```
Best Parameters: {'max_depth': 10, 'min_samples_split': 10,
'n_estimators': 200}
Classification Report - Optimized Random Forest:
              precision    recall  f1-score   support

           0       0.80      0.78      0.79       114
           1       0.81      0.83      0.82       128

    accuracy                           0.81       242
   macro avg       0.81      0.80      0.80       242
weighted avg       0.81      0.81      0.81       242
```

▶ Code



▶ Code



- The Random Forest Classifier has outperformed the Decision Tree in terms of accuracy, demonstrating a higher capability to classify race outcomes more effectively using the pit stop data.
- This also suggests that race outcomes are significantly influenced by pit stop strategies, highlighting an important insight that can help predict potential race winners.

Multiclass Classification

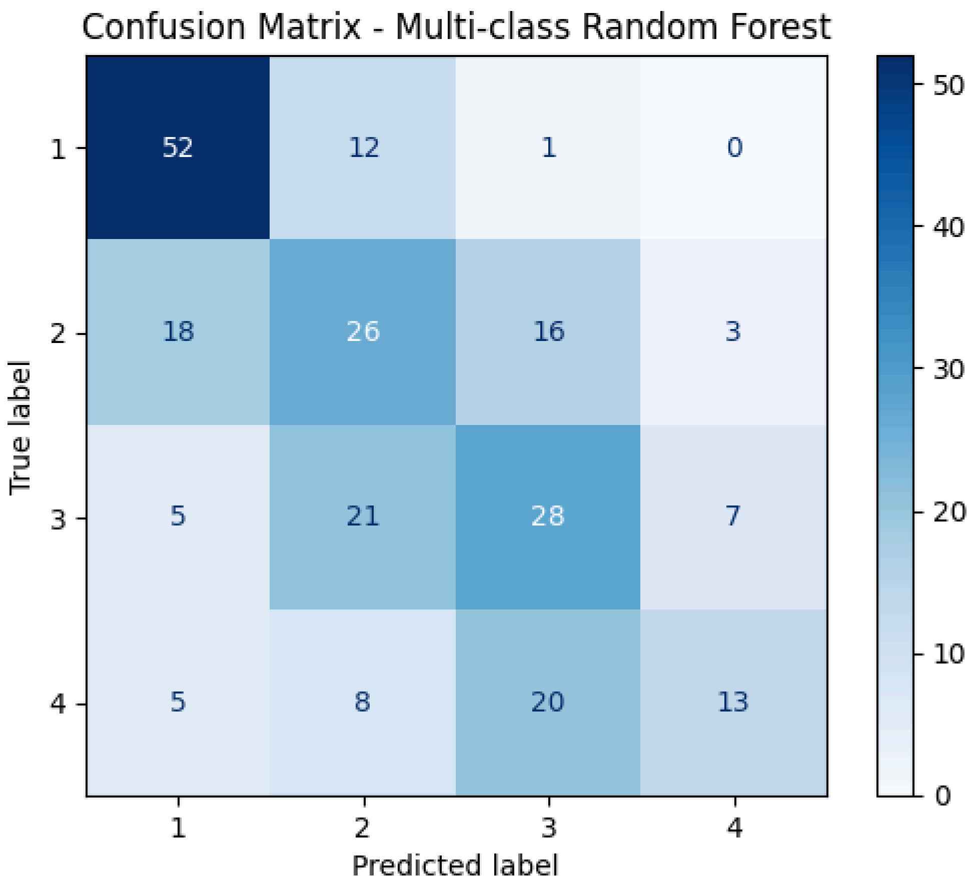
Random Forest Classifier

▶ Code

▶ Code

▶ Code

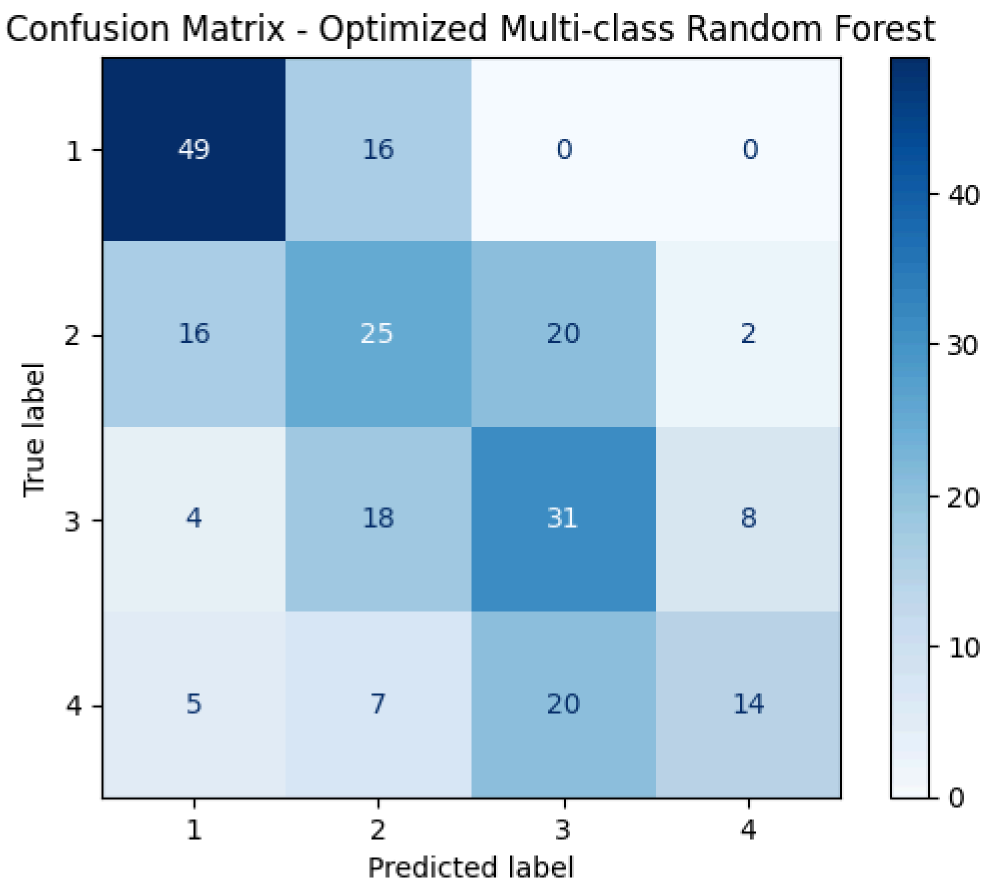
```
Classification Report - Multi-class Random Forest:
              precision    recall  f1-score   support

           1       0.65      0.80      0.72        65
           2       0.39      0.41      0.40        63
           3       0.43      0.46      0.44        61
           4       0.57      0.28      0.38        46

    accuracy                           0.51       235
   macro avg       0.51      0.49      0.48       235
weighted avg       0.51      0.51      0.49       235
```



▶ Code

```
Best Parameters: {'max_depth': 10, 'min_samples_split': 5,
'n_estimators': 200}
Classification Report - Optimized Multi-class Random Forest:
              precision    recall  f1-score   support

           1       0.66      0.75      0.71        65
           2       0.38      0.40      0.39        63
           3       0.44      0.51      0.47        61
           4       0.58      0.30      0.40        46
```

```
    accuracy                           0.51       235
   macro avg       0.52      0.49      0.49       235
weighted avg       0.51      0.51      0.50       235
```



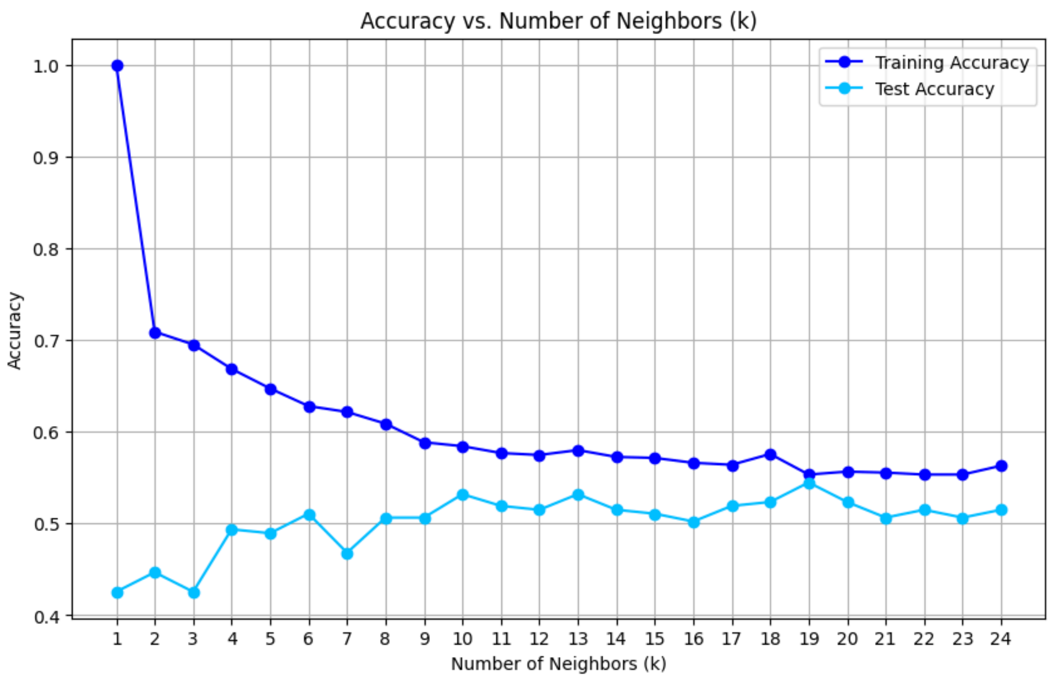
K-Nearest Neighbors (KNN)

KNN is a non-parametric and instance-based learning algorithm, meaning it makes no assumptions about the underlying data distribution and makes predictions based on the similarity of the input data to its neighbors.

Process:

- The algorithm calculates the distance between the input (test) data point and all training data points.
- Based on the distance metric, KNN identifies the k-nearest neighbors to the test data point.
- Prediction:
  - Classification: The algorithm assigns the class label that is the majority among the k-nearest neighbors (majority voting).
  - Regression: The algorithm averages the values of the k-nearest neighbors to predict the outcome.
  - **Hyperparameters**: K – number of neighbors.

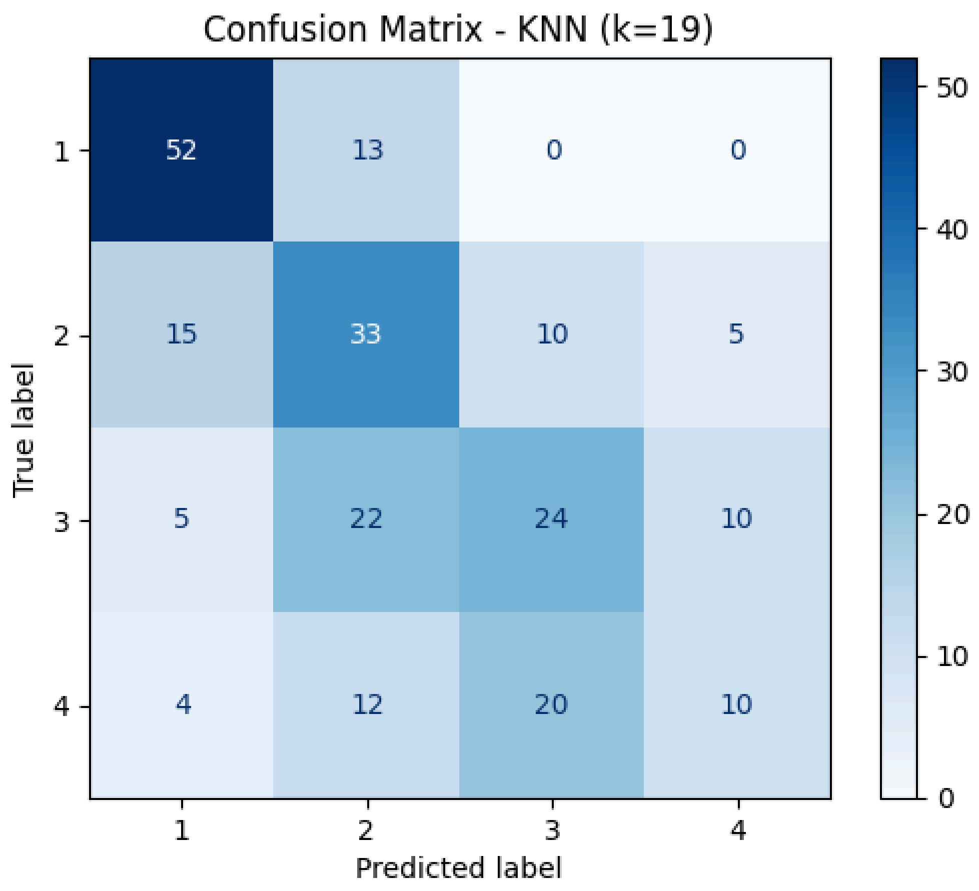
▶ Code



Optimal number for k is 19. Beyond 19, the test accuracy starts to drop.

▶ Code

```
Classification Report - KNN (k=19):
              precision    recall  f1-score   support

           1       0.68      0.80      0.74        65
           2       0.41      0.52      0.46        63
           3       0.44      0.39      0.42        61
           4       0.40      0.22      0.28        46

    accuracy                           0.51       235
   macro avg       0.49      0.48      0.47       235
weighted avg       0.49      0.51      0.49       235
```



Across models, the first class is consistently classified with the highest accuracy. This indicates that as the granularity of race results increases, classifying exact outcomes based solely on pit stop data becomes more challenging. While it is relatively easier to determine whether a driver will score points or not, accurately predicting the precise finishing position proves to be much more difficult.

Regression

Regression is a class of supervised learning algorithms used to predict continuous target variables by modelling the relationship between the input variables and the target variable.

Algorithms used in this project

1. Linear Regression: It is one of the most fundamental and widely used regression algorithms. It models the relationships between the dependent variables and the independent variables using a straight line.
2. Support Vector Regression (SVR): SVR is an extension of support vector machines for regression tasks. It tries to find a hyperplane that fits the data within a specified margin of tolerance.
3. Polynomial Regression: Polynomial regression extends linear regression by including polynomial terms (e.g., x2,x3x^2, x^3x2,x3) to capture non-linear relationships.
4. Random Forest Regressor: Random forest is an ensemble learning method that aggregates predictions from multiple decision trees to improve accuracy and robustness.

Reasons for choosing these algorithms

- Linear Regression: Provided a baseline model to understand the data's linear trends and evaluate other methods against.
- SVR: Captured non-linear relationships between features and outcomes, offering insights into more complex patterns.
- Polynomial Regression: Enhanced the ability to model curvature and non-linear interactions while maintaining interpretability.
- Random Forest Regressor: Leveraged its robustness and flexibility to handle complex interactions between features, offering strong predictive performance.

Regularization

To prevent overfitting, regularization techniques are applied.

- Ridge Regression: This technique adds an L2 penalty (squared magnitude of coefficients) to the loss function. It shrinks coefficients, effectively preventing overfitting by reducing the model's complexity while retaining all features.
- Lasso Regression: This method incorporates an L1 penalty (absolute magnitude of coefficients), which not only prevents overfitting but also performs feature selection by forcing some coefficients to become exactly zero. This helps in identifying the most relevant features for the prediction task.

Evaluation Metrics:

- Mean Absolute Error (MAE): Measures the average absolute difference between predicted and actual values, offering a straightforward interpretation of prediction errors.
- Mean Squared Error (MSE): Computes the average of squared differences between predictions and actual values, penalizing larger errors more heavily than MAE.
- Root Mean Squared Error (RMSE): The square root of MSE, providing error estimates in the same units as the target variable for easier interpretability.
- R-squared (R^2): Represents the proportion of variance in the target variable explained by the model, with higher values indicating better performance.
- Adjusted R-squared: Enhances R^2 by accounting for the number of predictors, discouraging the inclusion of irrelevant features.
- Mean Absolute Percentage Error (MAPE): Expresses prediction errors as a percentage of actual values, making it useful for comparing across datasets with varying scales.

▶ Code

	Year	Round	RaceName	DriverID	Lap	Stop	Time	Duration
0	0.0	0.0	Australian Grand Prix	alguersuari	0.000000	0.0	0.0	0.451650

	Year	Round	RaceName	DriverID	Lap	Stop	Time	Duration
1	0.0	0.0	Australian Grand Prix	michael_schumacher	0.000000	0.0	0.0	0.420133
2	0.0	0.0	Australian Grand Prix	webber	0.129870	0.0	0.0	0.393351
3	0.0	0.0	Australian Grand Prix	alonso	0.142857	0.0	0.0	0.390412
4	0.0	0.0	Australian Grand Prix	massa	0.155844	0.0	0.0	0.400336

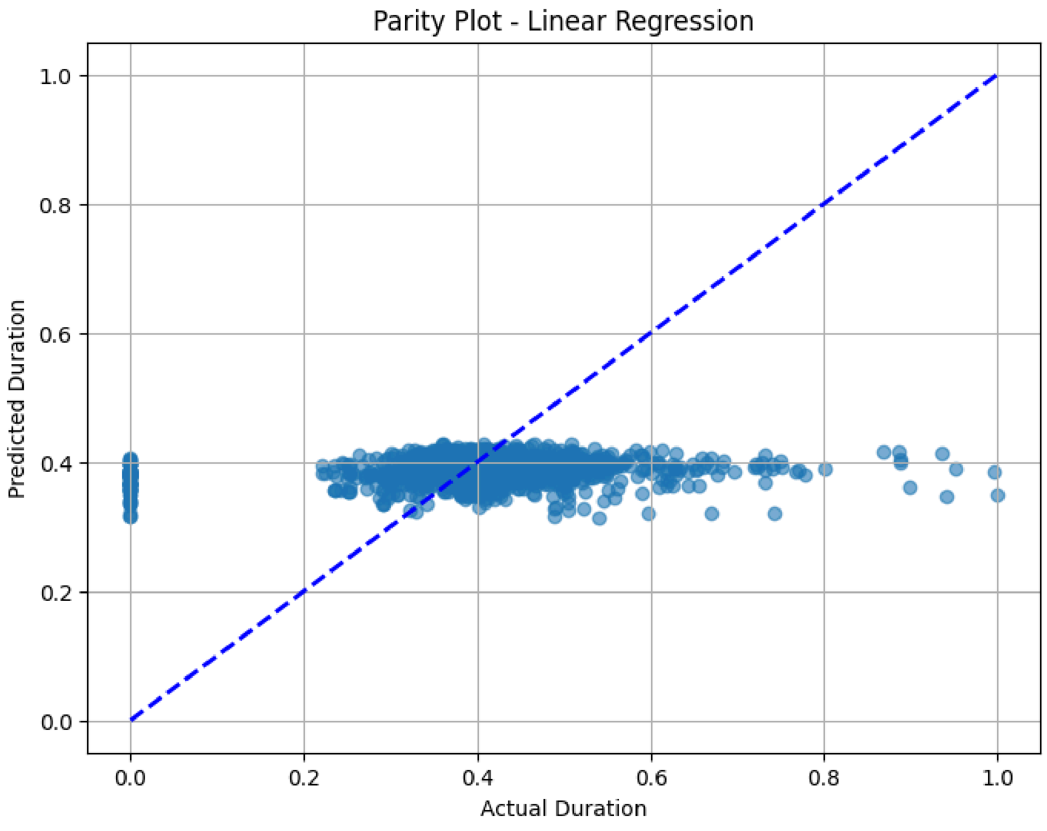
▶ Code

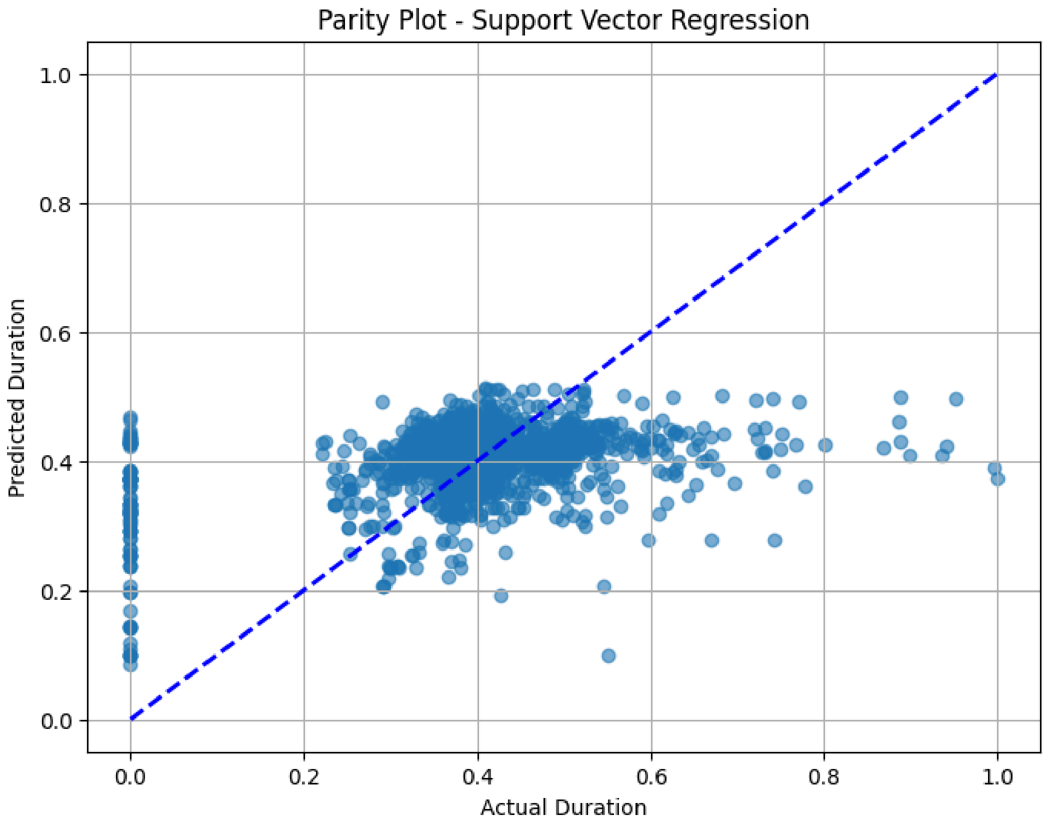
```
Linear Regression Metrics:
Mean Absolute Error: 0.0654
Mean Squared Error: 0.0118
R-squared: 0.0268

Support Vector Regression Metrics:
Mean Absolute Error: 0.0683
Mean Squared Error: 0.0104
R-squared: 0.1385

Polynomial Regression Metrics:
Mean Absolute Error: 0.0654
Mean Squared Error: 0.0116
R-squared: 0.0444

Random Forest Regressor Metrics:
Mean Absolute Error: 0.0341
Mean Squared Error: 0.0043
R-squared: 0.6408
```

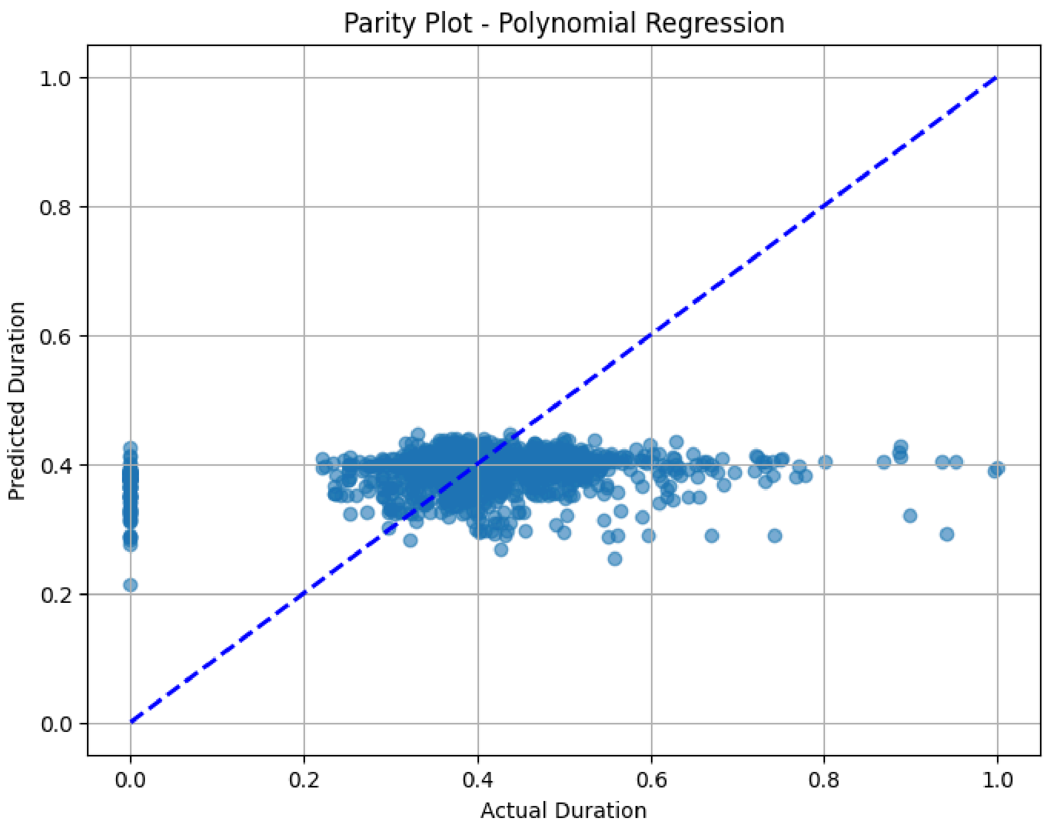
- Random Forest Regressor significantly outperforms other models in terms of MAE, MSE, and R-Squared metrics.
- Linear and Polynomial Regression models seem to struggle due to inherent complexity and non-linearity in the data.
- SVR performs better than Linear Regression but still underperforms compared to Random Forest Regressor.
- The results show that non-linear models are better suited for out data.

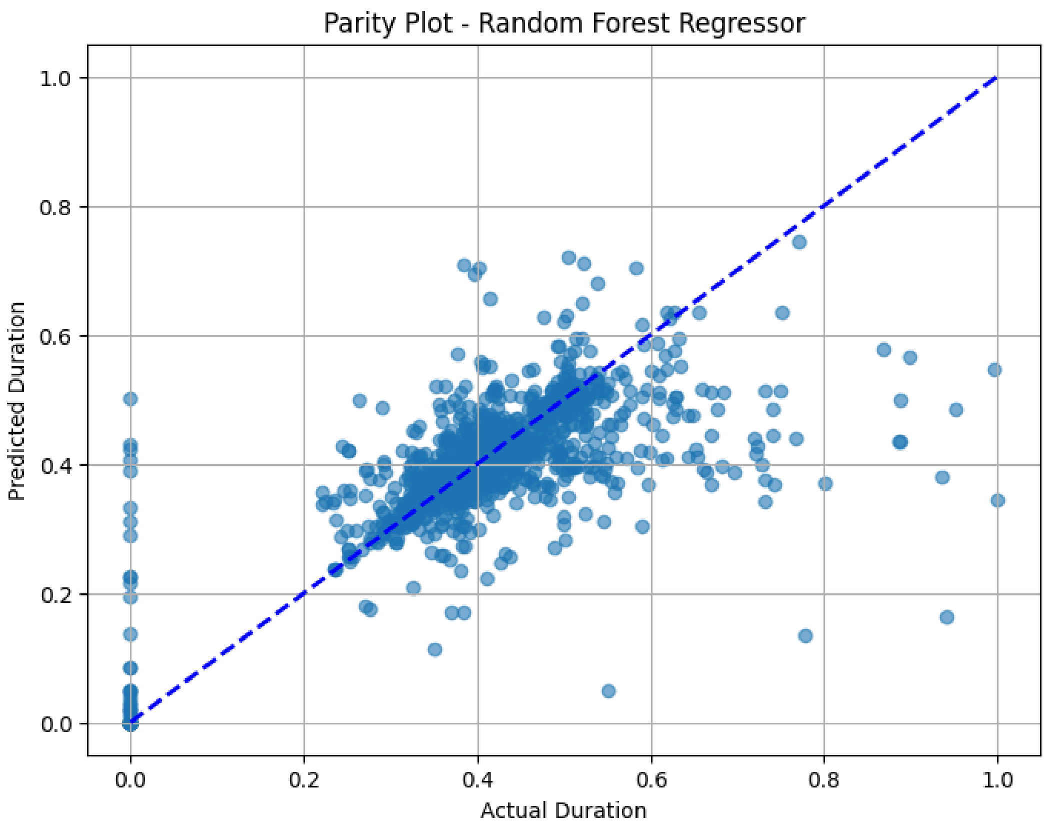
Well be tune the Random Forest Regressor to imporve its accuracy further.

▶ Code


Parity Plot - Linear Regression
Predicted Duration
Actual Duration
0.0
0.2
0.4
0.6
0.8
1.0


Parity Plot - Support Vector Regression
Predicted Duration
Actual Duration
0.0
0.2
0.4
0.6
0.8
1.0





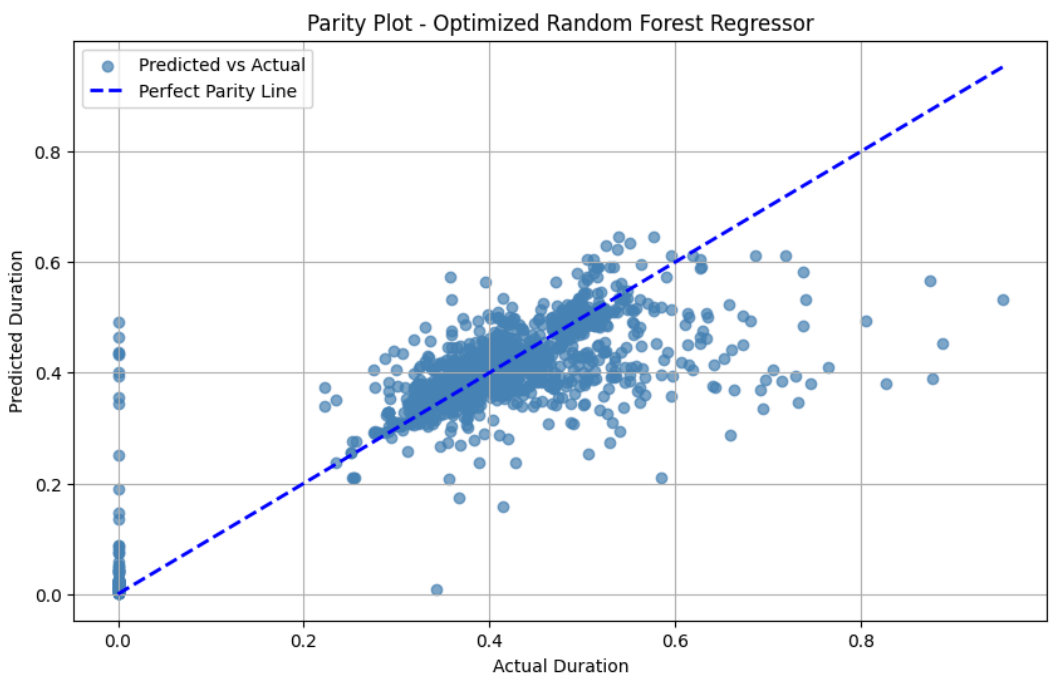
▶ Code

▶ Code

▶ Code

```
Optimized Random Forest Regressor Metrics:
Mean Absolute Error: 0.0343
Mean Squared Error: 0.0043
R-squared: 0.6439
```

▶ Code



The Parity Plot compares the predicted values of “Duration” (on the y-axis) with the actual values (on the x-axis) for the optimized Random Forest Regressor. The blue dashed line represents the perfect parity line, where predicted values are equal to actual values.

- Most of the points are concentrated near the parity line, indicating that the Random Forest model performs well in predicting the “Duration”, this suggests a good agreement between predicted and actual values.
- There are some outliers where the predicted durations deviate significantly from the actual values, especially for very low or very high durations.
- While the model performs well overall, further refinements, such as handling outliers or incorporating additional features, could improve accuracy, especially for extreme values.