Binary Classification of Formula 1 World Championship Dataset using Random Forest and Neural Networks

Baanee Singh & Shaan Sidhu

Task & Dataset: Binary Classification

This project focuses on predicting whether a Formula 1 driver would finish on the podium (top 3) in a race using structured historical data from the Formula 1 World Championship, spanning 1950 to 2020. The core objective is to treat this as a binary classification task, where the model learns from historical race features to determine the likelihood of a podium finish. Such predictions can have real-world applications in race strategy development, broadcast analytics, and fan engagement tools.

To build the dataset, we used the comprehensive Formula 1 dataset available on Kaggle, which includes detailed data table for race results, drivers, constructor teams, qualifying outcomes, circuits, lap times, pit stops, and more. Key information was compiled by merging several datasets using consistent identifiers: raceId, driverId, and constructorId. This merging process connected race-level metadata (year, round, circuit), driver details (name, nationality), and constructor information (team name), and performance metrics like qualifying position and finishing position.

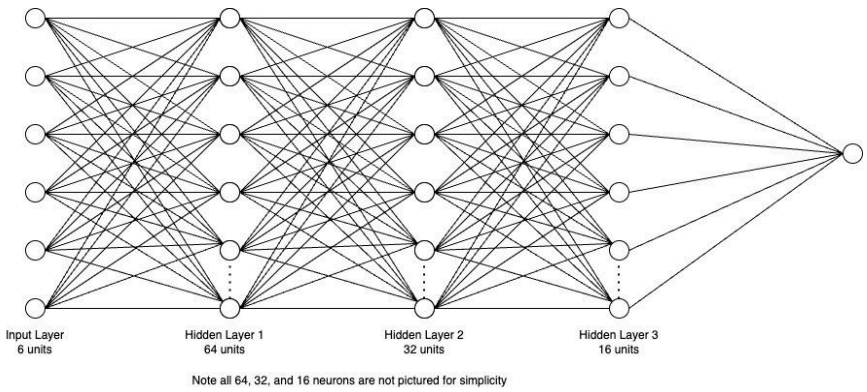
After consolidating the data, we performed several cleaning and preprocessing steps. We removed duplicates and filtered out entries missing crucial fields such as finishing and qualifying positions to ensure data quality. The positionOrder was renamed to finish_position, and qualifying data was merged using suffixes to distinguish it. We engineered a binary target variable, podium, defined as 1 if a driver finished in position 1-3 and 0 otherwise. We then selected key features likely to influence performance based on a correlation graph: driverId, constructorId, year, round, qualifying_position, and grid (starting position). Categorical variables like driver and constructor IDs were encoded numerically using category codes to make them more compatible with machine learning models.

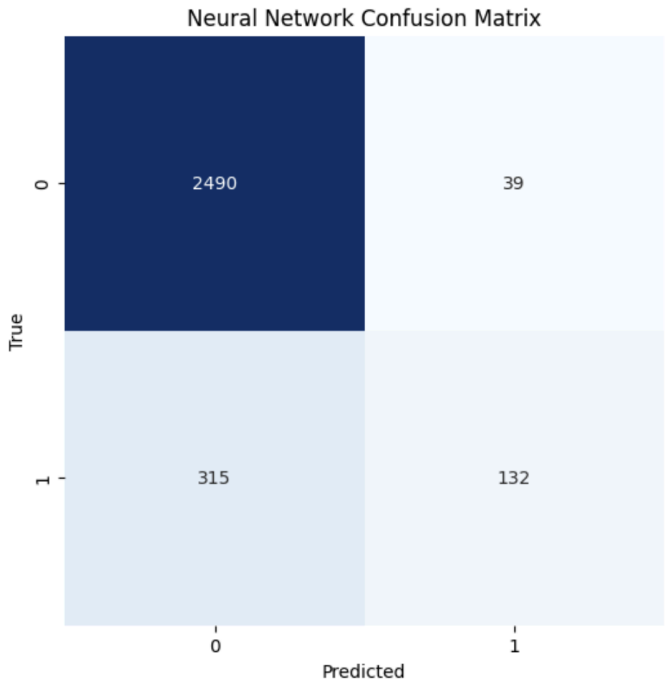
We implemented a temporal split for training and testing to maintain a realistic evaluation setting. All race data before 2018 was used for model training and validation, while data from 2018 onward was held out for testing. This simulates a production-like environment where the model is trained on past races and evaluated on unseen, future events. The resulting dataset forms a structured and meaningful foundation for building predictive models that capture trends in driver performance and race outcomes over time.

Multi-Layer Neural Network

The dataset was divided into training, validation, and test sets. Stratified sampling ensured consistent class distribution: training set (60%) used to fit the model, validation set (40%) used to tune hyperparameters, and test set included unseen data (post-2018) to evaluate final performance. The target variable is the podium, a binary label indicating a top 3 finish.

For our first model, we did a multi-layer neural network model. We created a feedforward, fully connected neural network using PyTorch. We extended the built-in PyTorch nn.Module to customize it for our data. Our model consists of 64 neurons in the first layer, 32 neurons in the second layer, 16 neurons in the third layer, and 1 neuron in the output layer (see image). The architecture of the neural network was chosen to start with more neurons and then compress down to one, as this is a binary classification problem, so a binary output is expected from our neural network. It is a common deep learning method to reduce the number of neurons by half, going through each layer. The model has a dropout rate of 0.3, which means 30% of the neurons are set to 0, for randomization and to make sure the model doesn't overfit. We hyperparameter-tuned the learning rate of the neural network. Deciding that a rate of 0.001 was best. Outside of the project code, we manually experimented with different num_epochs and batch sizes, but they didn't make any difference in performance, so we decided not to do any further tuning on those hyperparameters or include them.





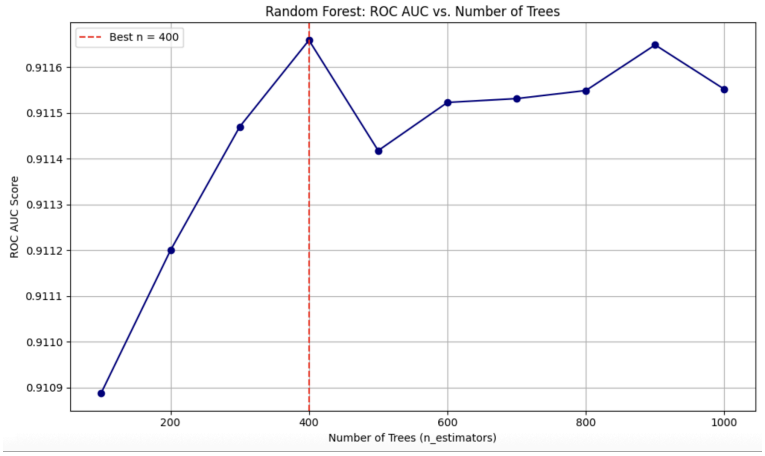
The neural network achieved an accuracy of 89.65%, with a precision of 0.7125, a recall of 0.5213, an F1 score of 0.6021, and an AUC score of 0.9172. All of these are very strong performance metrics, indicating an accurate and well-trained model. The confusion matrix is provided, with the ROC curve being in the results section.

Random Forest Classifier

For our second model, we implemented a Random Forest Classifier due to its robustness to overfitting, capability to handle class imbalance, and interpretability through feature importance scores.

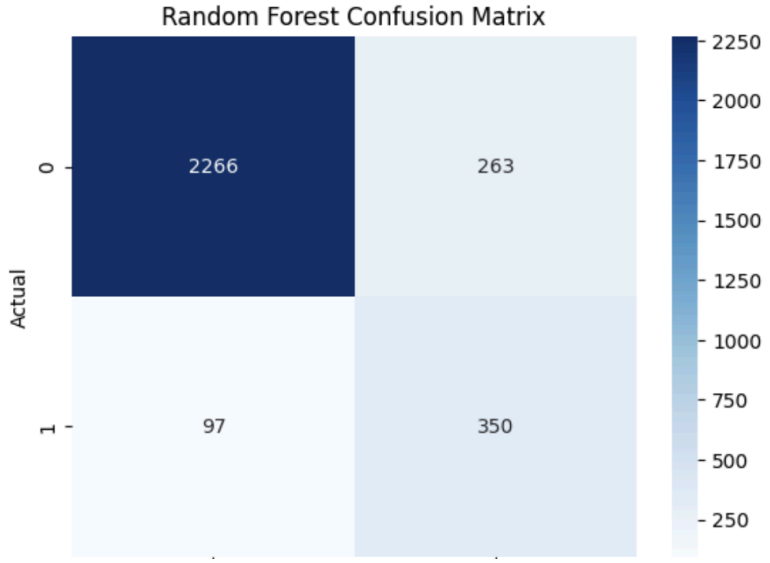
The dataset for this model was also divided into training, validation, and test sets. Stratified sampling ensured consistent class distribution: training set (60%) used to fit the model,

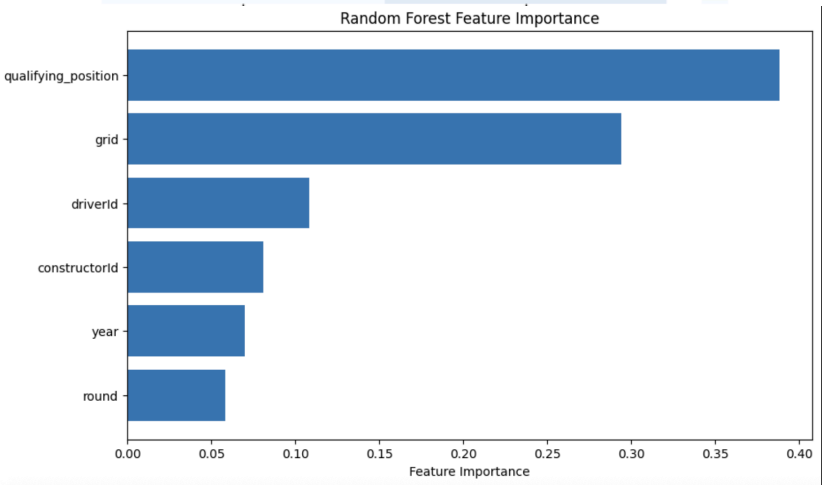
validation set (40%) used to tune hyperparameters, and test set included unseen data (post-2018) to evaluate final performance. The target variable is the podium, a binary label indicating a top 3 finish.



Hyperparameter tuning focused on finding the optimal number of decision trees (n_estimators). Models were trained with tree counts ranging from 100 to 1000, increasing in steps of 100. Each model's performance was measured using the ROC AUC score on the validation set. Results showed that validation AUC steadily improved up to 400 trees, peaking at 0.9117. Beyond that, the score changes were marginal, so 400 trees were selected as the optimal configuration.

Using this setup, a final model was trained and evaluated on the test set. The model achieved an accuracy of 87.9%, a ROC AUC score of 0.916, and an F1-score of 0.66 for the podium class. While precision for predicting podium finishes was moderate at 0.57, the model achieved a high recall of 0.78, suggesting that it was effective at identifying most true podium finishes, albeit with some false positives. The ROC AUC curve for Random Forest and the neural network can be found in the results section.





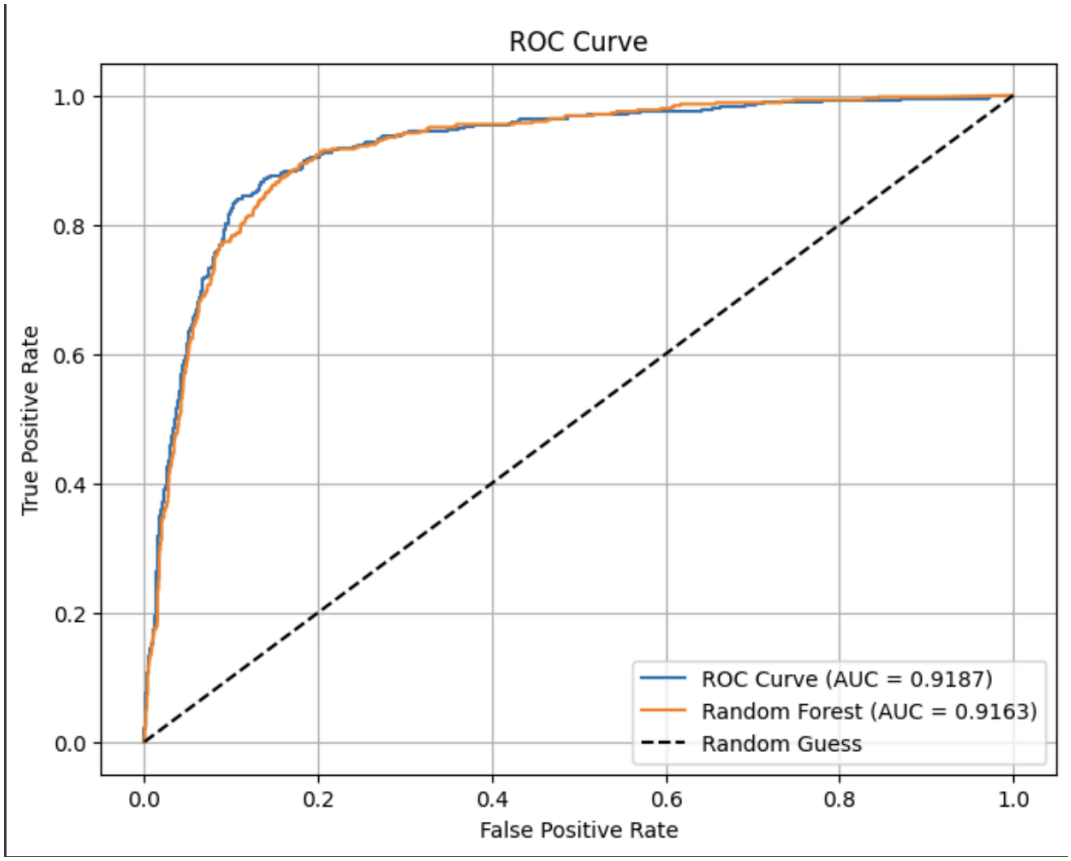
A confusion matrix visualization confirmed this trade-off, showing a greater number of false positives than false negatives for the minority class (podium finishes). The ROC curves further supported the model's strength, illustrating a clear separation between classes with an AUC of 0.916. Additionally, feature importance analysis provided insights

into what influenced the model's decisions. The most important features were qualifying position and grid position, which align with real-world race dynamics where starting position strongly impacts outcomes. Other influential features included constructor ID and driver ID, capturing consistent performance traits of teams and drivers, as well as year and round number, which helped the model account for season-long trends and variations across race stages.

Overall, the Random Forest Classifier demonstrated strong predictive performance and interpretability. Its ability to generalize well on unseen data and highlight relevant features makes it a suitable choice for this classification task. Although precision could be improved, especially for the podium class, the model offers a solid foundation that could be worked on further.

Results & Further Discussion

Both ROC curves were put on the same graph for easy comparison and are provided below.



The Multi-Layer Neural Network and the Random Forest Classifier demonstrated strong performance in predicting podium finishes, but each model offered distinct advantages. The neural network achieved a slightly higher accuracy (89.65% vs. 87.9%) and a marginally better ROC AUC score (0.9172 vs. 0.916), indicating strong overall classification capability. However, the Random Forest classifier outperformed the neural network in terms of recall (0.78 vs. 0.5213), making it more effective at identifying actual podium finishes, a crucial metric when the

goal is to capture as many positive cases as possible. The neural network, on the other hand, achieved higher precision (0.7125 vs. 0.57), suggesting it made fewer false positive predictions.

Referencing the metrics and graphs of each model, we can conclude that Random Forest and neural network models can be used to accurately predict podium placements in an F1 race. The high accuracy, recall, F1, precision, and AUC scores all provide support for this claim. These results highlight the potential of data-driven approaches to support strategic decision-making in motorsport, driver sponsorship valuation techniques, and strategies for increasing fan engagement. Future work could explore real-time data integration, feature enrichment (e.g., weather, tire strategies), and model explainability to further improve performance and usability.

Note: All metrics based on one or two runs of the Google Colab so metrics may change on re-run.