Predicting F1 results using Machine Learning algorithms

Kotoula Artemis[1], Evangelinou Maria[1]

[1] School of Medicine, University of Crete, Heraklion, Greece

Formula One (F1) is regarded as the pinnacle of international motorsport for single-seater formula racing cars, where F1 teams rely heavily on the massive amounts of data produced during each session to gain a competitive advantage. This assignment aims to explore the application of machine learning algorithms for driver performance prediction in F1 races. The training dataset for the algorithms contained various forms of race data, including race results, qualifying positions, and other factors that may significantly impact performance, such as constructor's standings. By analyzing these data points, the goal was to develop a predictive model for race outcomes. Several methods used in machine learning were employed in the analysis, including Principal Component Analysis (PCA), Logistic Regression for binary classification, Random Forest to improve prediction accuracy and overcome overfitting problems, and Cross Validation to verify the results. By applying these techniques, models were produced that identify data patterns which significantly influence race outcomes, providing insights into critical factors affecting driver performance and enhancing predictive capabilities in Formula One racing.

Introduction

Formula One (F1) is the highest class of international racing for open-wheel single-seater formula racing cars sanctioned by the Fédération Internationale de l'Automobile (FIA). The FIA Formula One World Championship has been one of the world's premier forms of racing since its inaugural running in 1950. The championship consists of two parts: the drivers' and the constructors' where currently twenty drivers compete for ten teams and make up points according to their finishing position at the end of each race. For the races specific cars are assembled by the constructors which must be in compliance with the regulations sanctioned by the FIA[1].

In order to gain a competitive advantage in every season, F1 teams rely on the massive amounts of data produced during each race and for this reason, in recent years, machine learning (ML) and the use of it in F1 has transformed the way predictions happened[2].

This project aims to make use of data produced in F1 races and apply machine learning algorithms in order to predict driver performance in future F1 races. By analyzing these data, ML algorithms can identify patterns and trends that may influence race outcomes and develop a predictive model for those outcomes. Several machine learning methods were employed in this analysis including, Principal Component Analysis (PCA) for feature exploration, Logistic Regression and Random Forest for binary classification and Cross Validation (CV) that provided a more reliable estimate of the model's performance versus a single train-test split.

Materials and Methods

Data collection

The first step in the methodology involved the data collection. The initial approach was using the python library FastF1[3], which offers a wide range of data regarding all F1 sessions through the ERGAST API (http://ergast.com/mrd/), a web service that provides a historical record of F1 races since 1950. This approach was unsuccessful due to their server's often shutdowns. This issue was overcome by downloading available data from past F1 championships from Kaggle[4], an online data science community platform where users can share their datasets. In this analysis, the Formula 1 World Championship (1950 - 2024) dataset by Kaggle user Vopani was used (https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020). This dataset contained most information on the Formula 1 races, drivers, constructors, qualifying, circuits, lap times, pit stops, championships from 1950 till the latest 2024 season.

Data Preprocessing

After several datasets were downloaded from Kaggle containing results and data for racing and qualifying sessions, drivers, constructors and circuits, several parameters were selected to be used as features for the models' training. These parameters and their meanings can be seen in Table 1.

A few changes were made to the acquired data. The *q3* parameter was converted to seconds and the error rates for the drivers and the constructors were calculated by dividing the number of did not finish (DNF) statuses that were due to drivers' or constructors' fault respectively, by the total of the races they competed in. In order to account for several constructors which were renamed, their ids were merged into one. Finally, before training the model, the finishing position of the drivers was excluded from the training datasets and used for prediction.

Principal Component Analysis

Principal Component Analysis (PCA) was used in order to make an initial exploration of the chosen features and to locate the most important factors influencing race outcomes. PCA was implemented using the scikit-learn[5] Python module using the default parameters while only changing the number of components.

Random Forest Classifier

The random forest classifier was applied to the preprocessed data, again using the skit-learn module. This classifier creates multiple decision trees by using different subsets of the original dataframe and makes predictions based on the majority result of these trees. It is worth mentioning that for reproducibility purposes the random state parameter with the value 2 was used wherever it was applicable.

Table 1: Features selected for model training and their meanings.

Feature	Explanation
raceId	An id number assigned to each race
driverId	An id number assigned to each driver
constructorId	An id number assigned to each constructor
grid	The driver's starting grid position at the race
race_position	The driver's finishing race position
standings_points	The driver's points in the championship
standings_position	The driver's position in the championship
standings_wins	The driver's season wins at the time of the race
constructor_points	The constructor's points in the championship
constructor_position	The constructor's position in the championship
constructor_wins	The constructor's season wins at the time of the race
year	The year that the race took place
circuitId	An id number assigned to each circuit
qualifying_position	The driver's qualifying position for the race
q3	The top 10's drivers' qualifying time
DriverErrorRate	The error rate of the driver
ConstrErrorRate	The error rate of the constructor
home_advantage	If the driver competes in their country

Logistic Regression

The next algorithm used for prediction was that of logistic regression. This analysis, alongside random forest, was used to predict the odds of a driver winning the race, based on the features of the dataset provided, after the appropriate hyperparameter tuning and feature selection was implemented.

Cross Validation

As a last step to ensure the validity and the reliability of the results that the generated model had provided, a 5-fold cross-validation analysis was applied, using the *cross_val_score* function of the sklearn module and setting the parameters *cv* and *scoring* to *5* and *'accuracy'* respectively. This allows for the evaluation of the performance of each algorithm and the detection of over-/under-fitting of the model by comparing the cross-validation score to the accuracy of the already computated models.

Hyperparameter Tuning and Feature Selection

For both random forest and linear regression, hyperparameter tuning and feature selection were used in order to assure the best performance and accuracy. For the random forest, hyperparameter tuning was applied first, using the *RandomizedSearchCV* of sklearn with the parameters: *scoring = "neg_mean_absolute_error", n_iter = 10, random_state = 2, n_jobs = 2*, and the *param_distributions as: "max_features": [1, 2, 3, 5, None], "max_leaf_nodes": [10, 100, 1000, None] and "min_samples_leaf": [1, 2, 5, 10, 20, 50, 100]*. For the logistic regression model, *LogisticRegressionCV* was used with default values to create a model with the best parameters.

After the best parameters were calculated, they were used for feature selection, where the feature importance was calculated using permutation and implementing *permutation_importance* for both models, again form the sklearn python module. This allowed for the estimation of the effect that each feature has on the model and the creation of a new dataset by excluding features of negative importance. On this new dataset, once again hyperparameter tuning was performed and the final parameters of the model were calculated.

Results

PCA

As a first initiative, PCA was implemented in order and to have a first look at the features used. The data were loaded and converted into a dataframe, the features were standardized and PCA was applied. The explained variance ratio for each component was also extracted and the cumulative variance was calculated and plotted (Figure 1) to assess how many components are required to capture most of the variance. As shown in Figure 1, at least ten principal components (PCs) were needed to retain over 90 percent of the variance.

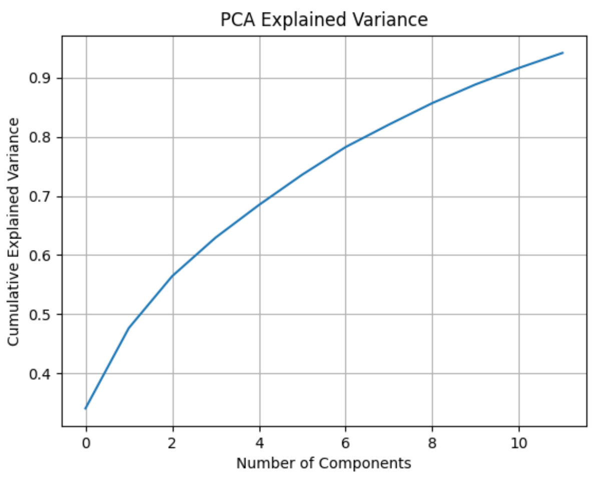
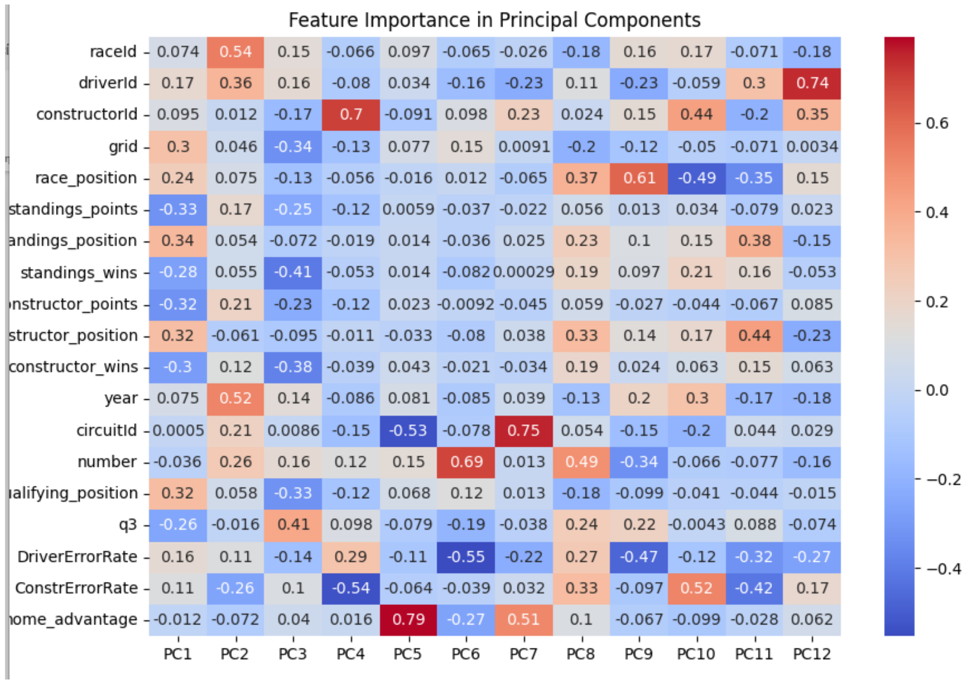


Figure 1: A plot showing how the explained variance increases by increasing the number of principal components (PCs). The x-axis shows the number of PCs, and the y-axis shows the total amount of variation explained by those PCs.

The importance of each feature had on each PC was calculated and visualized in a heatmap, as shown in Figure 2. The color scale of the heatmap ranges from blue to red and indicate the relative importance of the features. Blue indicates a feature's lower value while red a higher importance and each color suggest a smaller or larger role to the variance of the dataset. The columns represent the PCs and the rows correspond to the original feature list. The values which were either positive or negative indicate the direction of the relationship between the feature and the PC. A positive number suggests that the feature increases as the PC increases, while a negative value suggests a decreasing relationship.



	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12
raceId	0.074	0.54	0.15	-0.066	0.097	-0.065	-0.026	-0.18	0.16	0.17	-0.071	-0.18
driverId	0.17	0.36	0.16	-0.08	0.034	-0.16	-0.23	0.11	-0.23	-0.059	0.3	0.74
constructorId	0.095	0.012	-0.17	0.7	-0.091	0.098	0.23	0.024	0.15	0.44	-0.2	0.35
grid	0.3	0.046	-0.34	-0.13	0.077	0.15	0.0091	-0.2	-0.12	-0.05	-0.071	0.0034
race_position	0.24	0.075	-0.13	-0.056	-0.016	0.012	-0.065	0.37	0.61	-0.49	-0.35	0.15
standings_points	-0.33	0.17	-0.25	-0.12	0.0059	-0.037	-0.022	0.056	0.013	0.034	-0.079	0.023
andings_position	0.34	0.054	-0.072	-0.019	0.014	-0.036	0.025	0.23	0.1	0.15	0.38	-0.15
standings_wins	-0.28	0.055	-0.41	-0.053	0.014	-0.082	0.00029	0.19	0.097	0.21	0.16	-0.053
nstructor_points	-0.32	0.21	-0.23	-0.12	0.023	-0.0092	-0.045	0.059	-0.027	-0.044	-0.067	0.085
structor_position	0.32	-0.061	-0.095	-0.011	-0.033	-0.08	0.038	0.33	0.14	0.17	0.44	-0.23
constructor_wins	-0.3	0.12	-0.38	-0.039	0.043	-0.021	-0.034	0.19	0.024	0.063	0.15	0.063
year	0.075	0.52	0.14	-0.086	0.081	-0.085	0.039	-0.13	0.2	0.3	-0.17	-0.18
circuitId	0.0005	0.21	0.0086	-0.15	-0.53	-0.078	0.75	0.054	-0.15	-0.2	0.044	0.029
number	-0.036	0.26	0.16	0.12	0.15	0.69	0.013	0.49	-0.34	-0.066	-0.077	-0.16
ialifying_position	0.32	0.058	-0.33	-0.12	0.068	0.12	0.013	-0.18	-0.099	-0.041	-0.044	-0.015
q3	-0.26	-0.016	0.41	0.098	-0.079	-0.19	-0.038	0.24	0.22	-0.0043	0.088	-0.074
DriverErrorRate	0.16	0.11	-0.14	0.29	-0.11	-0.55	-0.22	0.27	-0.47	-0.12	-0.32	-0.27
ConstrErrorRate	0.11	-0.26	0.1	-0.54	-0.064	-0.039	0.032	0.33	-0.097	0.52	-0.42	0.17
ome_advantage	-0.012	-0.072	0.04	0.016	0.79	-0.27	0.51	0.1	-0.067	-0.099	-0.028	0.062

Figure 2: A heatmap showing the importance of each feature in each PC. The red color indicates a higher value and the blue a lower one.

The heatmap shows that certain features were crucial to the PCA outcome. Specifically, *driverId* parameter had an importance of 0.74 in the PC12, the *constructorId* had 0.7 in PC4, the *circuitId* 0.75 in PC7, the *home_advantage* 0.79 in PC5. There were also several parameters with negative effect, such as the *DriverErrorRate* in PC6, *ConstErrorRate* in PC4, and *circuitId* in PC5.

Random Forest

Random forest was implemented in two ways, both for binary classification. Firstly, a model was created in order to classify podium winners, meaning the top three finishers and second another model to classify the winner and non-winners of the race.

Podium Prediction

The first step in this methodology was to import the preprocessed data into the algorithm and split them into training and testing datasets. These datasets were then fitted in a random forest model with default parameters, which showed an accuracy of 0.892. The next step was hyperparameter tuning where several different parameters were tested in order to find the best set of parameters that maximize the model's accuracy. After the implementation, the best parameters were found to be the following: *min_samples_leaf = 5, max_leaf_nodes = 10* and *max_features* = *None*, which improved the accuracy to 0.899. After that, the feature importance was explored through random forest and each feature was investigated for their possible effect to the model, either positive or negative. The importance was calculated using the *permutation_importance* tool of the sklearn module using permutation for each feature. Figure 3 plots the importance of each feature on the model. In total, 10 features had positive importance, five had negative and two were of zero importance, which can be seen extensively in Table 2.

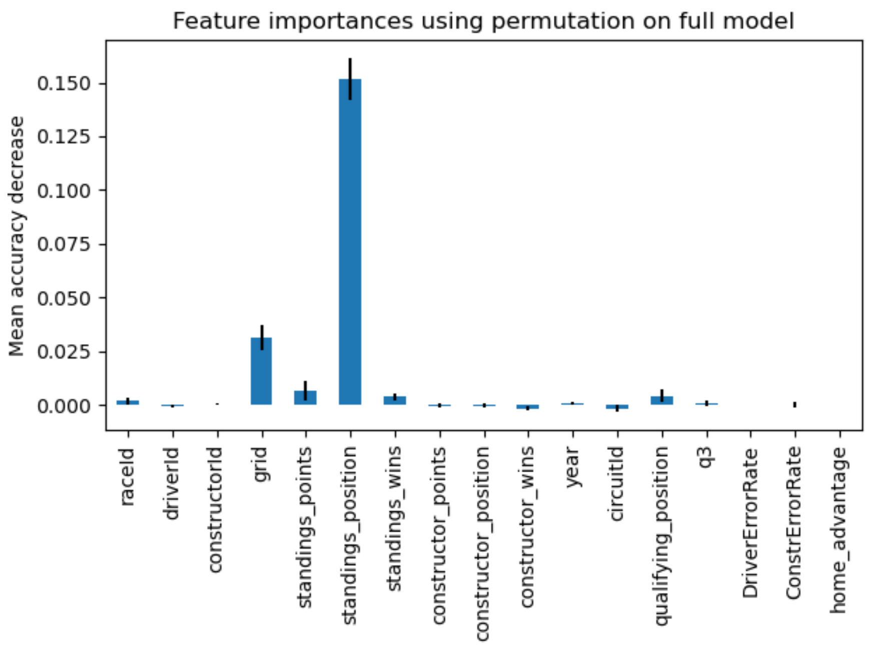


Figure 3: Bar plot showing the significance of each initial feature for the podium prediction random forest classifier model.

After the feature exploration, all features of negative importance except the driverId were removed from the dataset in an effort to improve the accuracy and the new dataframe with the reduced 13 features, was again put through hyperparameter tuning. This time around, the best parameters were *'min_samples_leaf': 5, 'max_leaf_nodes': 100, 'max_features': 3* and the accuracy of the model was found to be 0.904.

Table 2: Table showing the feature importance scores of the podium random forest prediction model.

Positive importance		**Negative Importance**	
standings_position	0.151574	constructor_position	-0.000336
grid	0.031319	driverId	-0.000521
standings_points	0.006597	constructor_points	-0.000556
qualifying_position	0.004225	circuitId	-0.001620
standings_wins	0.003854	constructor_wins	-0.001632
raceId	0.001736		
q3	0.000764	**Zero Importance**	
year	0.000394	DriverErrorRate	0.0
constructorId	0.000174	home_advantage	0.0
ConstrErrorRate	0.000069		

Winner prediction

A random forest classifier model was also computed for the prediction of the race winner. The method followed was the same as the one followed for the podium prediction. A model with default parameters was first created with an accuracy of 0.942. The hyperparameter tuning showed *min_samples_leaf = 5, max_leaf_nodes = 10* and *max_features = None* as the best parameters and increased the accuracy to 0.946. Figure 4 as well as Table 3 showcase the importance of each feature. The features chosen to be removed were *constructor_wins, circuitId, standings_wins* and *DriverErrorRate* and after their exclusion, the new best parameters for the model were found to be *min_samples_leaf = 5, max_leaf_nodes = 1000, max_features = 2*, which lead the accuracy to be 0.949.

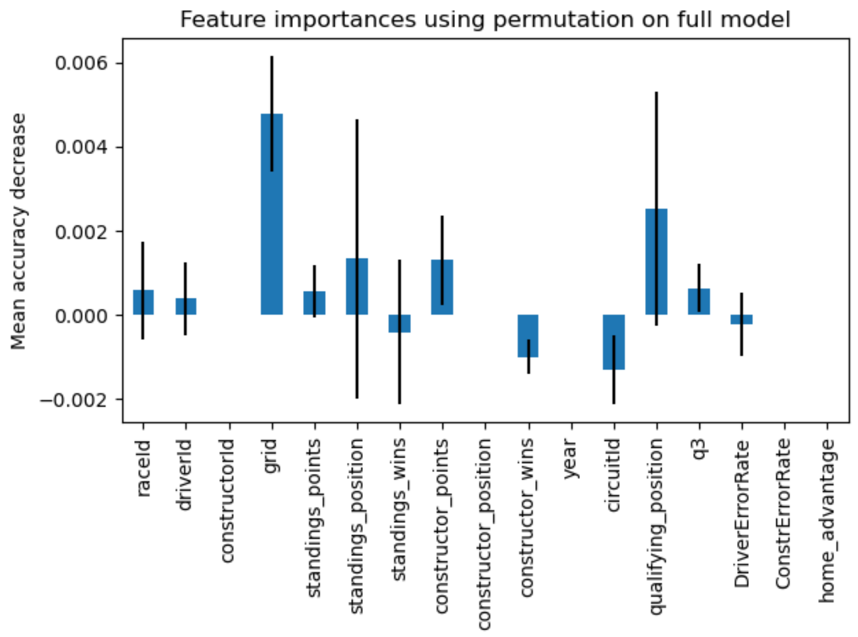


Figure 4: Bar plot showing the significance of each initial feature for the winner prediction random forest classifier model

Table 3: Table showing the feature importance scores of the winner random forest prediction model.

Positive importance		**Negative Importance**	
grid	0.004780	DriverErrorRate	-0.000231
qualifying_position	0.002523	standings_wins	-0.000405
standings_position	0.001343	constructor_wins	-0.000995
constructor_points	0.001308	circuitId	-0.001308
		Zero Importance	
q3	0.000637		
raceId	0.000590	constructorId	0.0
standings_points	0.000567	home_advantage	0.0
driverId	0.000394	constructor_position	0.0
		year	0.0
		ConstrErrorRate	0.0

Logistic Regression Model

The next model chosen to be used for predictions was a logistic regression model. This model's main goal is to describe and estimate the relationship between one dependent binary variable and independent variables.

The methodology used followed the same logic as the one mentioned above. After splitting the initial dataset to training and test data, a first model was created with default parameters, which had an accuracy of 0.894 for the podium prediction and 0.94 for the winner. After hyperparameter tuning, the accuracy was improved slightly to 0.902 for the top three prediction, while for the winner, there was no significant improvement. Finally, at the feature selection stage, the *q3* feature was removed for both models, as it was the only feature with a negative impact. Figure 5 shows the feature significance and Table 4 shows extensively each value. After *q3* was removed, the accuracy of the podium model was about the same, at 0.902 but the winners model accuracy was slightly decreased at 0.939.

Table 4: Table showing the feature importance scores of both the podium and winner prediction logistic regression models.

Podium		Winner	
Feature	**Importance**	**Feature**	**Importance**
standings_position	0.077431	standings_points	0.028086
grid	0.032716	standings_wins	0.016127
qualifying_position	0.029977	constructor_wins	0.006443
standings_points	0.018480	qualifying_position	0.005478
constructor_points	0.015934	grid	0.005170
constructor_position	0.007755	constructor_points	0.002739
raceId	0.003935	constructor_position	0.000502
standings_wins	0.003627	circuitId	0.000347
circuitId	0.001929	constructorId	0.000270
constructorId	0.000849	standings_position	0.000231
constructor_wins	0.000424	raceId	0.000077
driverId	0.000309	year	0.000000
year	0.000270	driverId	0.000000
DriverErrorRate	0.0	DriverErrorRate	0.000000
ConstrErrorRate	0.0	ConstrErrorRate	0.000000
home_advantage	0.0	home_advantage	0.000000
q3	-0.000694	q3	-0.000772

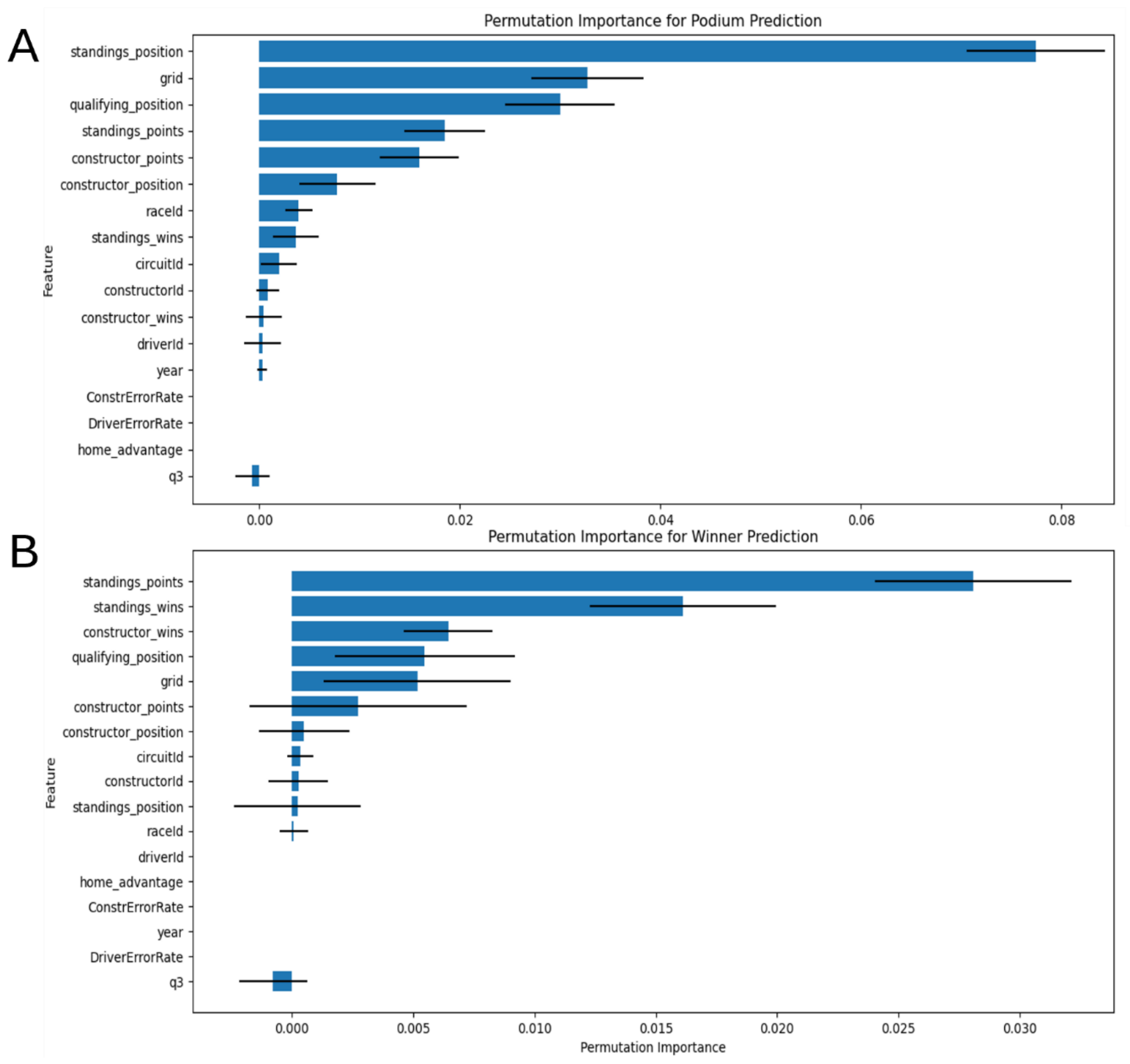


Figure 5: Bar plots showing the significance of each initial feature for the logistic regression models. A. The feature importance for the podium prediction. B. The feature importance for the podium prediction.

Cross Validation

Cross validation (CV) was implemented from the sklearn library and was used to evaluate the models' performance in a more reliable way. The dataset was split in five different subsets, and the model was trained five times, each time "hiding" one of the subsets from the model and using it as the test set while the four remaining subsets were used for the training set. The score that was calculated from each of these subsets represented the performance of the model on each different subset of the data.

Random Forest CV

CV was first implemented for the random forest model that was previously produced and the scores can be seen at Table 5. The mean accuracy for the podium using the Random Forest model based on the CV analysis was 0.89 across all folds, and for the winner was 0.94, while the original accuracies of the models were 0.904 and 0.949 respectively. The fact that the differences between the cross validation and the models are small, indicate that the random forest is stable for the dataset and there is no evidence for overfitting.

Logistic Regression CV

CV was also applied to the logistic regression models, where the mean accuracy for the podium produced by CV was 0.885 and for the winner was 0.947 across all folds, with the original accuracy scores produced by the models were 0.902 and 0.94 respectively, indicating that there was no overfitting and that the model was reliable.

Overall, the results demonstrated that the Random Forest predictions were slightly more accurate.

Table 5: Cross validation accuracy scores for both podium and winner models in random forest and linear regression.

Random Forest CV		**Logistic Regression CV**	
Podium	**Winner**	**Podium**	**Winner**
0.89788054	0.94412331	0.89980732	0.9344894
0.87668593	0.94412331	0.87090559	0.94219653
0.89382239	0.94980695	0.88996139	0.95945946
0.88996139	0.95366795	0.88803089	0.95559846
0.89382239	0.95366795	0.87837838	0.94401544

Discussion

This project's main objective was the use of machine learning algorithms for accurate prediction of Formula One race results, and the identification of the algorithm which would be able to generate the best predictions between random forest and linear regression.

In order to achieve this, PCA was used as an initial feature exploration where it was evident that the variance explained by each feature was generally low and many features, relative to their total number would need to be retained to prevent the loss of significant variance, which is why the decision was made to not use PCA for dimensionality reduction.

For both models created, the hyperparameter tuning and feature selection preformed, seemed to lead to an increase in accuracy and comparing the resulted accuracies of the random forest and logistic regression models, the first one shows a slightly higher score with both of them being adequately high.

Finally, cross validation confirmed the validity of those models with a small but expected difference in the accuracy scores, leading to the assumption that there was no overfitting responsible for those high numbers and that the models created can predict with high precision both the winners, and the top podium finishers in F1 races.

Future Steps

While the results presented sound promising, there are future steps that could be taken in F1 prediction, and a more exhaustive analysis could be conducted by taking into account more features. While this analysis used only 17, there are many more that could influence race outcomes such as the weather, driver age and experience, performance in the three practices and many more. By taking into consideration factors like the aforementioned, the models could make more precise predictions.

The models could be also extended to predict where a certain driver could finish, the probability of the driver to achieve the fastest lap or to not finish the race, even train the model in real time during a race, employing real-time data to influence predictions. In general, F1 is a sport that relies on precision, producing vast amounts of data on each race weekend which can all be used for the development of new models and more complicated models.

References

1. Formula 1. (2024, February 24). Everything you need to know about F1 – Drivers, teams, cars, circuits and more. Formula 1® - the Official F1® Website. https://www.formula1.com/en/latest/article/drivers-teams-cars-circuits-and-more-everything-you-need-to-know-about.7iQfL3Rivf1comzdqV5jwc
2. Miller, R. (2021, October 21). How Cloud Data-Crunching Power Accelerates the F1 Racing Experience. *Data Center Frontier*. https://www.datacenterfrontier.com/cloud/article/11427867/how-cloud-data-crunching-power-accelerates-the-f1-racing-experience
3. The FastF1 Development Team FastF1 (3.4.0), FastF1, (2024), GitHub repository, https://github.com/theOehrly/Fast-F1
4. Kaggle (2021). *Formula 1 World Championship Data*. Retrieved from https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020
5. Pedregosa et al. (2011) *Scikit-learn: Machine Learning in Python*. JMLR 12, pp. 2825-2830.