Formula 1 Grand Prix Pole to Win Classification

Dakota Mellish
Universidad de Madrid Politécnica
Madrid, España 28660
Email: dmellish99@gmail.com

Abstract

Since 2009, The qualifying winner of Formula 1 races has won the grand prix roughly fifty-percent of the time. The researcher desires to understand if it is possible to predict this event given information on the drivers, car constructors, qualification results, race circuit, well as weather conditions. In part 1 of the study, five non-probabilistic classifiers are utilized, with 4 different methods of feature subset selection. In part 2, eight probabilistic classifiers are also considered with different methods of feature subset selection Each classification model undergoes a rigorous process of grid-search and cross-validation to ensure robust results. The researcher concludes that it is indeed possible to classify a pole-to-win conversion with approximately eighty-two percent accuracy.

I. Introduction

A. Motivation

Formula 1 races have been around since 1950 and have evolved considerably since their inception. Compared to other motor sports such as NASCAR and Indy car series in the USA, the qualifying part of the race weekend has considerably more weight, as passing other racers is more challenging in Formula 1. This is primarily due to the complexity of the race circuits as well as the heavy downforce that F1 cars possess. An interesting question poses itself, "Is it possible to predict if the winner of qualifying (pole position) will win the race?" Such a model could have interesting applications in the areas of sports betting, where casual and avid fans alike are eager to have an edge prior to placing their bets.

B. Research Goal

The goal of this study is three-fold and will be identified separately and combined for both non-probabilistic and probabilistic classifiers:
a) Understand if classification methods can classify F1 pole to win better than random guessing (above 50% accuracy).
b) Identify which of the models utilized provide the best overall classification (considering various metrics).
c) Identify which of the four subset selection methods provide the best overall classification.

II. Methodology

A. Dataset

The primary source of data is an open-source Kaggle dataset [1], which is derived from official F1 statistics and records. The dataset goes back as far as 1950, however, only 2009-2024 races will be considered due to the many rule changes and design changes to the F1 cars that have occurred. A secondary dataset (for weather observations) was sourced from Visual Crossing [2], a historical weather data API that was called for each race event and retrieves aggregate information from t-2 hours of the race start until t+3 hours. One row is considered for each pole leader of each race, and other racers are not considered directly (though there are columns which address this). In total, 293 observations (races) for 293 pole leaders are considered beginning in 2009 and ending September 2024.

It is important to explain the format of a typical Formula 1 race weekend. Each weekend typically has 1-3 *free practice sessions*, which are then followed by a *qualifying session* of *3 rounds* to determine initial racing grid starting position, and finally *the race*. Attributes are primarily derived off of known driver and car constructor attributes, but *qualifying results* and *race* statistics are also considered.

B. Features

In Table 1.1 a short summary of the features implemented are provided with a description of each feature.

Table 1.1: Features Used in Classification

Feature	Description	Variable Type
finished_first_q1	Binary variable that assesses if the pole winner won the first round of qualifying.	categorical
finished_first_q2	Binary variable that assesses if the pole winner won the second round of qualifying.	categorical
dif_seconds_quali_first_second	Seconds difference in the final round of qualifying between the pole (first) and second position.	continuous
dif_seconds_quali_second_third	Seconds difference in the final round of qualifying between the second and third position.	continuous
dif_seconds_quali_third_fourth	Seconds difference in the final round of qualifying between the third and fourth position.	continuous
dif_seconds_qualif_fourth_fifth	Seconds difference in the final round of qualifying between the fourth and fifth position.	continuous
is_americas	Binary variable that assesses if the race course is in the Americas.	categorical
is_australia	Binary variable that assesses if the race course is in Australia.	categorical
is_europe	Binary variable that assesses if the race course is in Europe.	categorical
is_reigning_constructor_champion	Binary variable that assesses if the pole winner races for the latest constructor's champion.	categorical
is_reigning_driver_champion	Binary variable that assesses if the pole winner is the latest driver's champion.	categorical
is_leader_25_percent_through	Binary variable that assesses if the pole winner is in first place at the 25% mark of the race.	categorical
is_leader_50_percent_through	Binary variable that assesses if the pole winner is in first place at the 50% mark of the race.	categorical
driver_age	Driver's age in days	discrete
had_rain	Binary variable that assesses if rain occurred during t-2 hours up until t+3 hours of the race time (t).	categorical
avg_temp	Average air temperature from t-2 hours up until t+3 hours of the race time (t).	continuous
avg_humidity	Average air humidity percentage from t-2 hours up until t+3 hours of the race time (t).	continuous
avg_pressure	Average air pressure from t-2 hours up until t+3 hours of the race time (t).	continuous
avg_wind_speed	Average wind speed from t-2 hours up until t+3 hours of the race time (t).	continuous
constructor_position_pre_race	Ranking of the pole leader's constructor (team) position prior to the race.	discrete
driver_position_pre_race	Ranking of the pole leader's driver position prior to the race.	discrete

Many of the features were derived using feature engineering using Pandas in a Jupyter notebook. This includes the *dif_seconds* family pertaining to the final qualification session, as well as the *driver_position_pre_race* and *constructor_position_pre_race* fields. The features *finished_first_q1* and *finished_first_q2* refer to if the pole winner also finished first in the 1st and 2nd rounds of qualifying, respectively. Other fields were simply joined to primary as the F1 Kaggle dataset is comprised of many separate tables such as qualification, races, constructors standings, driver standings, driver info, circuit info and more. It is worth mentioning that although many races do occur in Asia, this binary variable is excluded, as it is explained by is_americas, is_europe and is_australia. The features draw upon known attributes prior to the day of the race (such as driver and team characteristics), track specific elements, qualifying performance and day-of-race variables such as weather, along with in-race variables such as *is_race_leader_25_percent_through*.

Data Preprocessing

Because of the large difference in the scale of data between features, it is imperative to scale all numerical features prior to fitting the classification models to the features, so that the model doesn't inadvertently place higher priority to the features with larger differences. The StandardScaler() method from scikit-learn [3] proves to be quite helpful in this case. It scales all variables based on their deviation from the mean for the given feature.

Consideration of Class Balance

It is important to check if the target output class is sufficiently balanced prior to fitting, that is to say that the amount of qualifying winners who win the race is equal to the amount of qualifying winners who do not win the race. Upon quick inspection, it is easy to see in table 1.2 that the dataset is well-balanced.

Table 1.2: Class Label Balance

Class	Count	%
1	151	52%
0	142	48%

C. Models and Feature Subset Selection (FSS) Methods

The following non-probabilistic classification models will be considered: *k nearest neighbors*, *RIPPER (repeated incremental pruning produce error reduction)*, *support vector machine*, *decision tree*, and a *neural network multilayer perceptron with two hidden layers, the first layer having 16 neurons and the second layer having 8 neurons*. The following probabilistic classifiers will also be considered: *logistic regression*, *linear discriminant analysis*, *naive bayes bernoulli*, *tree augmented naive (TAN) bayes*, *random forest, bagging, stacking, and xgboost classifier*. The stacking classifier includes a mixture of weak-learners, such as bagging, boosting, decision tree, k nearest neighbors, logistic regression, random forest, support vector machine, and

XGBoost classifier.

The following feature subset selection methods are considered: *all features, univariate filter, multivariate filter and wrapper filter*. For the univariate filter, 5 categorical features are selected using a chi square test in relation to the class label (won race or not), and 4 numerical features are selected using an f regression in relation to the class label. The multivariate subset selection method uses a correlation feature subset selection filter using Weka [4] , and the wrapper method selects 7 features are using the sequential feature selection technique in Sci-kit learn [5]. For the neural network and tree augmented naive bayes, the wrapper method in Weka using the greedy stepwise algorithm is employed [6].

Table 1.3: Non-Probabilistic Classifiers and Feature Subset Selection (FSS) Methods

Classifier/Feature Subset Selection	All Variables	Univariate Filter	Multivariate Filter	Wrapper Method
K Nearest Neighbors	No Extra Steps	Categorical Variables: Chi Square Test (Top 5 Scorers Selected) Numeric Variables: Anova F Regression (Top 4 Scorers Selected)	Correlation Subset Feature Select with GreedyStepwise (Weka)	Sequential Feature Selection with 7 variables selected
RIPPER				Sequential Feature Selection with 7 variables selected
Support Vector Machine				Sequential Feature Selection with 7 variables selected
Decision Tree				Sequential Feature Selection with 7 variables selected
Neural Network (Feed Forward 2 hidden layers)				GreedyStepwise (Weka)

Table 1.4: Probabilistic Classifiers and Feature Subset Selection Methods

Classifier/Feature Subset Selection	All Variables	Univariate Filter	Multivariate Filter	Wrapper Method
Logistic Regression	No Extra Steps	Categorical Variables: Chi Square Test (Top 5 Scorers Selected) Numeric Variables: Anova F Regression (Top 4 Scorers Selected)	Correlation Subset Feature Select with GreedyStepwise (Weka)	Sequential Feature Selection with 7 variables selected
Linear Discriminant Analysis				
Naive Bayes Bernoulli				
TAN Bayes*				GreedyStepwise algorithm (Weka)
Random Forest Classifier				
Bagging Classifier				Sequential Feature Selection with 7 variables selected
Stacking Classifier				
XGBoost Classifier				

In general, all objects are implemented in Python using Scikit-learn [7], and Tensorflow [8] for the neural network. The RIPPER rules induction algorithm is an exception, as it utilizes both scikit-learn and a custom library called Wittgenstein [9]. Tree augmented naive bayes is also an exception; it uses the Weka open-source software library [10], as well as XGBoost, which has its own open source python library [11]. Apart from TAN bayes due to its Weka-only implementation, all models will utilize a process of both grid search cross validation in python [12] and then evaluated based on a KFold [13] cross validation with 5 folds using the best hyperparameters selected from the grid search. The hyperparameters explored and selected results can be located in Table 5.1 of the appendix.

Table 1.5: Univariate Filter (Left) and Multivariate Filter (Right) Variables

Feature
finished_first_q2
avg_wind_speed
is_leader_25_percent_through
is_leader_50_percent_through
is_reigning_constructors_champion

Feature
finished_first_q1
finished_first_q2
is_reigning_constructor_champion
is_leader_25_percent_through
'is_leader_50_percent_through
dif_seconds_quali_first_second
dif_seconds_qualif_fourth_fifth
driverAge
driverPositionPreRace

Due to the nature of the wrapper method, there are too many different variable combinations to show for each individual model.

D. Evaluation Criteria

Mean KFold cross-validation accuracy, precision, recall, F1-score, as well as AUC-ROC curve characteristics (area under curve receiver operating characteristic) will be analyzed and discussed across model selection and feature subset selection. The mean was chosen in favor of the max due to the small number of samples in the dataset (n=293) and the desire for robust results.

III. Results for Non-Probabilistic Classifiers

Below are the results for mean precision, recall and F1 for each non-probabilistic classifier and feature subset selection method.

Table 2.1: Cross Validation Mean Precision, Recall, F1 Score, Accuracy and AUC

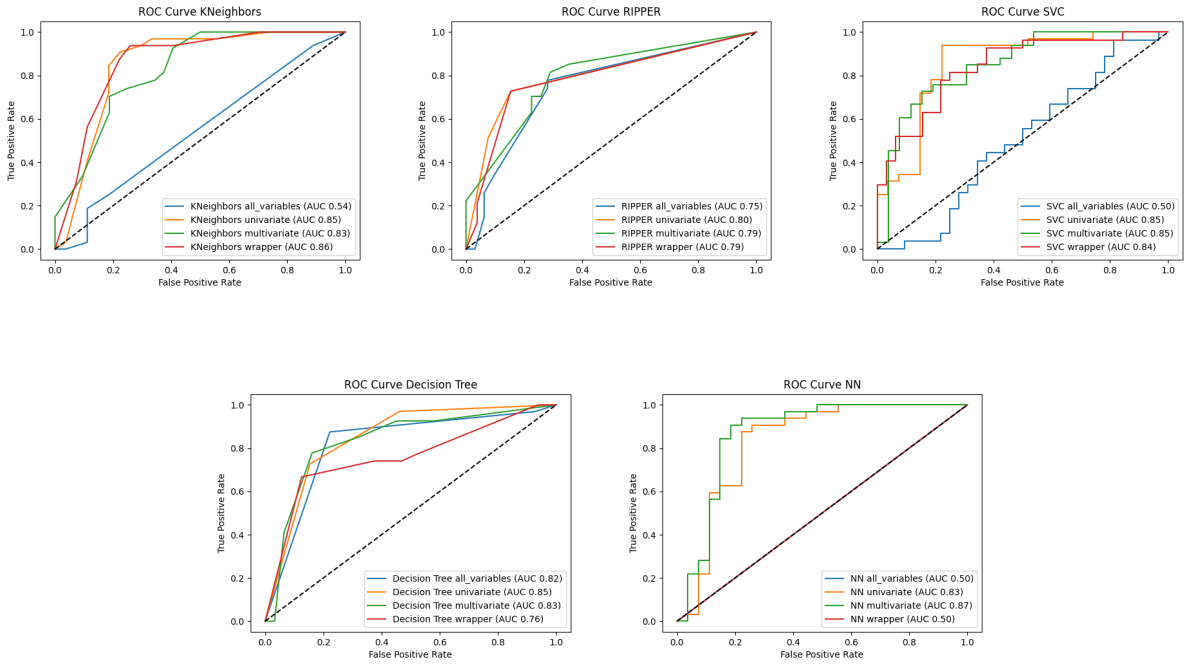
Metric	Precision				Recall				F1 Score				Accuracy				AUC			
Classifier/Feature Subset Selection	All Features	Univar Filter	Multivar Filter	Wrapper Method	All Features	Univar Filter	Multivar Filter	Wrapper Method	All Features	Univar Filter	Multivar Filter	Wrapper Method	All Features	Univar Filter	Multivar Filter	Wrapper Method	All Features	Univar Filter	Multivar Filter	Wrapper Method
K Nearest Neighbors	50.2%	79.8%	79.3%	**85.6%**	48.3%	75.9%	76%	**78%**	.435	.776	.773	**.814**	**47.1%**	76.1%	78.2%	**79.5%**	49.6%	**84.9%**	84.4%	82.8%
RIPPER	73.9%	80.8%	**82.6%**	82.4%	75.8%	**75.4%**	76.5%	80%	.731	.775	.791	**.81**	72.7%	74.1%	79.5%	**81.2%**	75.8%	78.7%	**82.1%**	80.5%
Support Vector Machine	53.6%	81.2%	78.3%	**81.9%**	55.2%	80.8%	81.4%	**84.3%**	.467	.808	.796	**.823**	51.2%	80.9%	81.3%	**82.6%**	52.2%	**86.1%**	**86.1%**	84.8%
Decision Tree	76.5%	**81.2%**	80.1%	70.8%	81.4%	80.8%	**82.8%**	76.1%	.785	.808	**.812**	.714	77.5%	77.8%	**80.6%**	**80.6%**	75%	**84.6%**	84.3%	75.9%
Neural Network (Feed Forward 2 hidden layers)	54.7%	81.5%	**81.7%**	**44.8%**	50.9%	76.3%	**77.6%**	**20%**	.512	.787	**.793**	**.276**	48.1%	76.5%	**80.6%**	47.8%	**48.9%**	85.2%	**85.4%**	51.4%

In Table 2.1, the k nearest neighbors classifier using the wrapper method provided the highest degree of precision. In this context, this means it correctly identify the true pole-to-win conversions out of all pole-to-win-conversions predicted **85.6% of the time**. On the flip side, this means that it incorrectly classifies non pole-to-win conversions **14.4% of the time** by classifying them as pole-to-win conversions. There does not appear to be an FSS method that performs the best for precision. Looking at recall, the support vector machine when using the wrapper method performs the best, meaning that out of all

true pole-to-win conversion, it correctly predicted the outcome **84.3% of the time**, also indicating that **15.7% of predictions** were not correctly picked up by the classifier. Notably, the neural network with a wrapper method scores quite low in the area of recall (**20%**), suggesting a high rate of false negatives (pole-to-win conversions that are classified as non pole-to-win conversions). The doesn't appear to be a clear leader in performance with regard to recall and feature subset selection methods. For F1 score, the support vector classifier provides the highest mean between precision and recall at **.823**. The wrapper method appears to perform the best in 3/5 models for F1 score. With regard to mean overall accuracy within cross-validation, the support vector machine with the wrapper method has the highest (**82.6%**). With regard to feature subset selection, the wrapper method provided the highest accuracy in the majority of the models. For AUC, the results are expressed as a percentage of total area (with 100% being an area of 1). Considering the area under the curve for the receiver operating characteristics, it is shown that the support vector machine yields the best results using the univariate and multivariate filter. The univariate and multivariate filter are tied with 3/5 models each showing the best performance with regard to this category.

Figure 2.1: Receiver Operating Characteristics Curves by Model and FSS

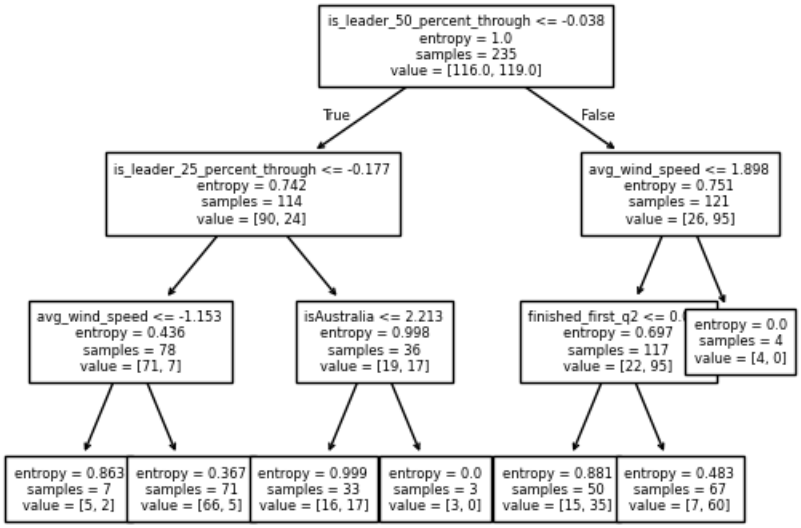
Below the median receiver operating characteristics are displayed for each classifier and FSS.



For the ROC curves, some notable distinctions are that the curve trajectory varies substantially depending on the feature subset selection method utilized. In the case of the neural network, the wrapper method and all variables leads to performance that effectively amounts to guessing (close to the line y=x). For other models such as k nearest neighbors and support vector machine, the models provide classifications that are almost just as bad as random guessing without any feature subset selection. For models such as RIPPER and the decision tree, the curve shows a propensity towards favoring sensitivity (correctly identifying a pole-to-win conversion) than specificity, which is correctly identifying a non pole-to-win conversion.

Figure 2.2: Decision Tree (univariate filter) Graph

The best performing decision tree model's graph is displayed below.



The variable *is_leader_50_percent_through* is at the top the decision tree (giving it the highest amount of predictive power), with true instances then going to the left and false instances to the right. From there, the features *is_leader_25_percent_through* is evaluated for true instances of *is_leader_50_percent_through* and *avg_wind_speed* for false instances of *is_leader_50_percent_through*. The feature *avg_wind_speed* is also assessed, though later on for positive instances of *is_leader_50_percent_through*.

Figure 2.3: Rules for RIPPER Algorithm (multivariate filter) The RIPPER algorithm provides an intuitive set of rules as part of its output. The best performing

```
[[is_leader_50_percent_through=1.0]
                OR
[is_reigning_constructors_champion=1.0 AND is_leader_25_percent_through=1.0 AND finished_first_q2=1.0]]
```

As shown, only two distinct rules exist for the set (given there are only five variables for the multivariate filter). The model checks if the pole qualifier is in first place 50% through the race, and if not, the model checks for a case where the given pole position qualifier drives for the constructors champion, is the leader of the race 25% through, and finished first in part 2 of the qualifying. Overall, it is helpful to understand which variables the models give the highest importance to.

IV. Discussion of Non-Probabilistic Classifiers

Circling back to the 3 mentioned objectives for the study, a discussion is made for each topic.

A. Understand if classification methods can classify F1 pole to win conversions better than random guessing (50% accuracy)

It is clear based on Table 2.2 regarding accuracy that there is an improvement in predictive power for pole-to-win conversions, as all models are able to achieve ¿80% accuracy. To be sure, there is much left to be desired with these results from a functional perspective (such as 90% accuracy), however given the vast amount of uncertainty and variation that comes with any sporting event such as Formula 1, such a goal may ultimately be unattainable. The most realistic approach would be to acknowledge that there a portion of both explainable variation (which can be explained using the given features), and unexplainable variation.

B. Identify which of the five classification models utilized provide the best overall classification (considering various metrics)

When looking at precision, the k nearest neighbors classifier when fitted with the wrapper method filter set of features classifier provides the highest rate of true pole-to-win conversions out of all predictions made. However, the support vector machine when using the wrapper method scores the highest in terms of recall. This is also evidenced in the F1 score, where the support vector machine also scores the highest. Taking into account that the support vector machine also has the highest accuracy and area under the curve (though not for the wrapper method), a strong argument can be made for this classifier. Based on recall, F1 score, accuracy, and auc, the support vector machine appears to be the best overall choice for classifying F1 pole-to-win conversions.

C. Identify which of the four subset selection methods (FSS) provide the best overall classification

As stated before, there is no clear winner with regard to FSS method for precision and recall, however F1 score and accuracy suggest that the wrapper method is best suited for this task (best outcome in 3/5 models). Although it doesn't provide the highest area under the curve for the receiver operating characteristic, F1 score and overall accuracy are metrics of higher importance, as accuracy considers all cases and F1 weighs both precision and recall equally.

D. Consideration of All Classifiers

Upon inspection, it is clear that some models are more sensitive in overall performance based on the variables chosen. Models such as k nearest neighbors, support vector machine, and the neural network particularly struggle with accuracy, as they provide results that are at times worse than random guessing. These classifiers appear to be sensitive to noisy variables, or perhaps having extra dimensions makes it challenging for the algorithms, when considering they rely heavily on the geometric distance between observations when fitting. It is worth remembering that these are non-probabilistic classifiers, and hard cuts are made within the features. In addition, the spread in accuracy for the best performing classifier of each model only spans 2-3%, suggesting that the model methodology only provides a small degree of performance boost. It is clear that with feature subset selection, all models did see an increase in all categories when subset selection was employed and that it is a useful technique to be used for classifying formula 1 pole-to-win-conversion.

E. Conclusion

In summary, non-probabilistic classifiers can predict Formula 1 pole-to-win conversions above 50% (random guessing), although they fall short of providing extremely reliable predictions (such as 95% accuracy) The support vector machine provides the highest overall degree of performance when taking into account its recall, accuracy, F1 score as well as AUC. It is also clear that in general, feature subset selection provides a boost in performance, and that the wrapper method is a good choice for this task.

V. Results for Probabilistic Classifiers

Below are the results for mean precision, recall, F1 score and accuracy for each probabilistic classifier and feature subset selection method

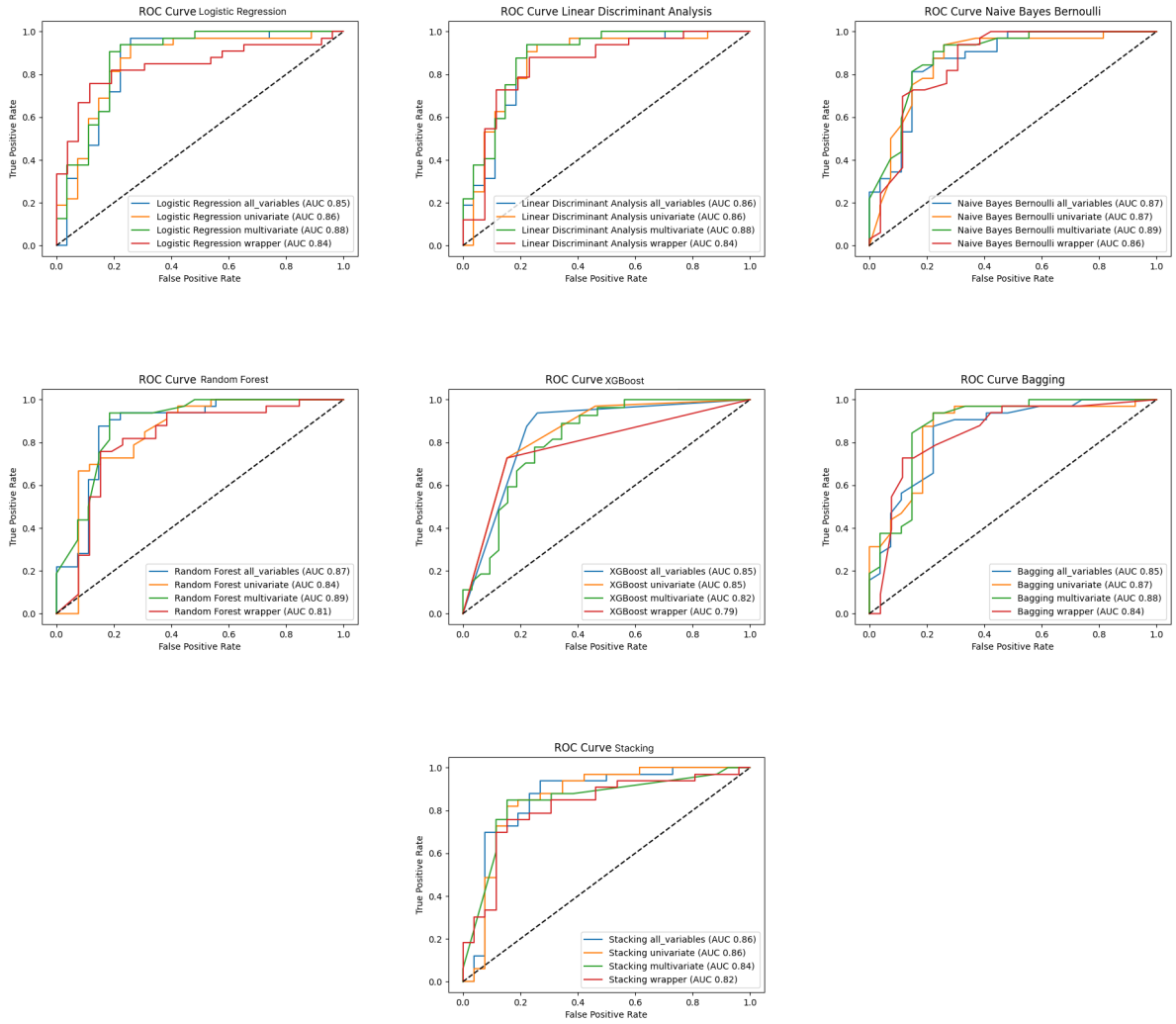
Table 5.1: Cross Validation Mean Precision, Recall, F1 Score, Accuracy and AUC

Metric	Precision				Recall				F1				Accuracy				AUC			
Classifier/Feature Subset Selection	All Features	Univar Filter	Multivar Filter	Wrapper Method	All Features	Univar Filter	Multivar Filter	Wrapper Method	All Features	Univar Filter	Multivar Filter	Wrapper Method	All Features	Univar Filter	Multivar Filter	Wrapper Method	All Features	Univar Filter	Multivar Filter	Wrapper Method
Logistic Regression	79.7%	80%	**82.2%**	81%	81.7%	80.3%	80.2%	**82.2%**	.805	.798	.81	**.814**	80.6%	81.2%	80.9%	**82.6%**	84.3%	85.2%	**87.1%**	84.3%
Linear Discriminant Analysis	78.7%	81.1%	**81.7%**	81.2%	80.3%	78.9%	**80.9%**	80.8%	.792	.795	**.811**	.808	79.5%	81.6%	81.6%	**81.9%**	86.0%	85.9%	**87.5%**	84.9%
Naive Bayes Bernoulli	81.4%	**82.4%**	82%	80.2%	76.7%	78.1%	**79.5%**	79.4%	.789	.798	**.806**	.796	78.5%	80.9%	**81.9%**	**81.9%**	85.4%	85.9%	**87.3%**	86.1%
TAN Bayes	79%	79.7%	79.7%	**81.6%**	82.8%	**83.2%**	**83.2%**	81.8%	.808	.814	.814	**.818**	79.7%	80.4%	80.4%	**81.2%**	84.4%	**85%**	**85%**	78.5%
Random Forest Classifier	82.7%	**76.9%**	**85.5%**	78.1%	**80.2%**	77.5%	75.5%	73.5%	**.813**	.768	.799	.754	77.2%	79.9%	80.9%	**81.2%**	87%	82.7%	**87.6%**	83.2%
Bagging Classifier	79.9%	**82.3%**	80%	79.2%	75.1%	80%	**81%**	79.4%	.767	**.809**	.802	.791	78.2%	**81.6%**	80.6%	79.5%	84.7%	86%	**86.8%**	85%
Stacking Classifier	79.1%	79.9%	**82.2%**	80.5%	75.5%	77.3%	**81.5%**	79.7%	.770	.783	**.816**	.797	77.8%	79.2%	**81.9%**	80.2%	85.3%	85.1%	**87.6%**	83.2%
XGBoost Classifier	81.3%	**84.7%**	80.8%	81.2%	68.9%	**68.2%**	78.9%	**80.8%**	**.736**	.745	.795	**.808**	79.5%	80.2%	80.2%	**80.9%**	82.5%	**85.4%**	85.3%	81.4%

In Table 5.1, the random forest classifier using the multivariate filter provided the highest degree of precision. In this context, this means it provides the the rate of true pole-to-win conversions out of all pole-to-win-conversions predicted **85.5% of the time**. The univariate and multivariate filter method also performed the best for 4 out of 8 models each. Looking at recall, the

TAN bayes classifier provides when using the multivariate filter performs the best, meaning that out of all true pole-to-wins, it correctly predicted the outcome **83.2% of the time**. The tree augmented naive bayes model when fitted with the wrapper method provides the highest F1 score at **.818**. The multivariate filter appears to be the best subset selection method for recall. There doesn't appear to be a clear winner for feature subset selection methods and F1 score.
With overall accuracy, the logistic regression classifier leads the way with the wrapper method at **82.6%**. In addition, 6/8 models scored the highest when using the wrapper method. The area under the curve is expressed as a percentage of maximum area (1=100%). The random forest and stacking classifiers when using the multivariate filter maximize the area under the curve. The multivariate filter appears to be the strongest selection method when considering AUC.

Figure 5.1: Receiver Operating Characteristics Curves by Model and FSS (except for TAN Bayes)
Below the median receiver operating characteristics are displayed for each classifier and FSS.



Unfortunately, due to the model creation of the TAN Bayes in Weka it was not possible to construct an ROC curve. For all other classifier models, they possess curves that yield a higher area under the curve than random guessing (y=x). The trajectory of the curves appears to vary more greatly the random forest, logistic regression and XGBoost classifiers. For some classifiers such as bagging and random forest, the curves have a sharp inclination up until x=.2 (where y=.8), suggesting a propensity towards sensitivity over specificity. In this context it means the classifiers perform better at identifying pole-to-win conversions than identifying non pole-to-win conversions. No curve exceeds .9 in total area.

Figure 5.2: Logistic Regression Coefficients (univariate filter)

feature	log odds	odds ratio	% Change in Odds	interpretation
is_leader_50_percent_through	0.98	2.66	166%	Increases Odds
is_leader_25_percent_through	0.59	1.80	80%	Increases Odds
finished_first_q2	0.45	1.58	58%	Increases Odds
is_reigning_constructor_champio	0.40	1.48	48%	Increases Odds
driverAge	0.13	1.14	14%	Increases Odds
finished_first_q1	0.11	1.12	12%	Increases Odds
driverPositionPreRace	0.09	1.09	9%	Increases Odds
dif_seconds_qualif_fourth_fifth	0.00	1.00	0%	Doesn't change Odds
isAustralia	-0.25	0.78	-22%	Decreases Odds

When exponentiation is used, the odds ratio can be interpreted for each feature. Understandably, *is_leader_50_percent_through* and *is_leader_25_percent_through* provide the greatest degree of change, as these are in-race metrics. A unit change in these variables (since they are binary variables this means if they are true) leads to an approximate 166% and 80% increase in odds of conversion for each feature, respectively. For the variable *isAustralia*, the race circuit location being in Australia leads to a 22% decrease in odds of winning approximately.

VI. Discussion for Probabilistic Classifiers

Circling back to the 3 mentioned objectives for the study, a discussion is made for each topic.

A. Understand if classification methods can classify F1 pole to win conversions better than random guessing (50% accuracy)

It is clear based on Table 2.2 that the variables used offer predictive power for pole-to-win conversions, as all models are able to achieve above 80% accuracy. To be sure, there is much left to be desired with these results from a functional perspective (such as 95+% accuracy), however given the vast amount of uncertainty and variation that comes with any sporting event such as Formula 1, such a goal may ultimately be unattainable. The most realistic approach would be to acknowledge that there a portion of both explainable variation (which can be explained using the given features), and unexplainable variation.

B. Identify which of the five classification models utilized provide the best overall classification (considering various metrics)

When looking at precision (total true predicted conversions out of all predicted pole-to-win conversions), the random forest classifier scores the highest, though perhaps at the expense of recall, where it performs less well. Similarly, the tree augmented naive bayes possesses the highest recall but with a lower degree of precision. It then inferable from Table 2.1 that no single model will possess the highest scores of both precision and recall for all categories of analysis. When considering F1 score, the tree augmented model is a tempting choice, however its 81.2% overall accuracy is a bit below the pack. The stacking classifier with the multivariate filter warrants some consideration as it has the next highest F1 score at .816 and is tied for first in overall accuracy, however it scores a lower accuracy than the logistic regression model with a wrapper method. The logistic regression model possesses the highest accuracy, the second highest recall, the third highest F1, and a respectable area under the curve at 84.6%. Overall, based on accuracy, recall and F1 score, the logistic regression classifier appears to be the strongest choice.

C. Identify which of the four subset selection methods (FSS) provide the best overall classification

It would be easy to simply select the feature subset selection that provided the best overall performance for the best overall classifier (logistic regression). However, it is important to consider all models and evaluate where performance improvement was made. It appears the multivariate filter models were the best performing for most classifiers in the areas of recall and area under curve. However, it is tied with the wrapper method for accuracy and F1 score. Furthermore, unlike the other models presented it is more interpretable in that the features' relationships can be understood with respect to the class. The two also tend not differ from one another by more than a percentage point or so, and with such small differences seems fair to suggest that both the multivariate filter and wrapper method offer a strong approach.

D. Consideration of All Classifiers

Unlike non-probabilistic classifiers, all of the probabilistic classifiers did not appear to be as sensitive to the features used. Part of this lies in the fact that probabilistic classifiers inherently depend on a probability which has flexibility from 0-1. This is a useful feature in that more features could potentially be considered without creating problems for feature auto-correlation or noisy variables. Some interesting notes can be made with respect to the TAN bayes and the naive bayes bernoulli classifiers.

Despite accounting for conditional dependence, the naive bayes bernoulli actually outperforms the TAN bayes with respect to accuracy, perhaps the fact that a bernoulli distribution is considered unlike the TAN bayes which isn't distribution specific. With regard to ensemble methods, it is intriguing how the random forest can achieve its highest F1 score using all features, a distinction not seen in any other model. Its overall accuracy however is not the highest when using all features.

E. Conclusion

In summary, probabilistic classifiers can predict Formula 1 pole-to-win conversions above 50% (random guessing), although they fall short of providing reliable predictions (such as 95% accuracy) The logistic regression model provides the highest overall degree of classifier performance when taking into account accuracy (82.6%), F1 score (.814), as well as AUC (.843). It is also clear that in general, feature subset selection provides a boost in performance, and that the multivariate filter and wrapper method are good choices for this task.

VII. Interpretation of All Classifier Results

As shown earlier for non-probabilistic methods, support vector machine using the wrapper method was shown to be the best overall model, with an F1 score of .823 an accuracy rate of 82.6%, and an AUC of .861. In comparison with the best model of the probabilistic classifiers, the support vector machine slightly outperforms the logistic regression classifier in F1 score and AUC, though with additional model tweaks such as further tuning of the L1/L2 regularization parameter, (which was utilized in the hyperparameter grid search), it is possible that the logistic regression model could outperform the support vector classifier. The support vector head-to-head appears to be the stronger choice based on having a higher F1 score and area under the curve.

Unfortunately, unlike models such as a decision tree or logistic regression, the support vector machine classifier lacks an intuitive interpretation, it is known that from the hyperparameter grid search, it selected the radial basis function kernel and a C parameter of 3, meaning it placed more emphasis on correctly identifying training examples. The radial basis function proved to be a better choice in the end than the linear kernel, which could suggest there are more non-linear relationships associated with the predictor variables and the outcome of the F1 race. Another important consideration is that while 21 features are available for the models, in the end techniques such as the multivariate filter only need 5 features to provide accuracy that is above 80%, suggesting potential autocorrelation between the features and/or noisy variables. It is worth considering which other features exist that might add predictive power not already found with the multivariate feature variables. Above all, it cannot be forgotten that only 293 observations are available unfortunately, all which span across nearly 15 seasons of Formula 1 racing. Performance would likely improve if more observations existed for this problem, such as having more races in a season or using more historical races, but given the limitations of existing data and how much the F1 cars have changed since their inception, it is best to consider more homogeneous races in nature.

References

[1] Rohan Rao. Formula 1 World Championship (1950 - 2024), 2020. Accessed: 2024-11-04.
[2] Visual Crossing Corporation. Weather api, 2024. Accessed: 2024-11-04.
[3] Scikit-Learn Developers. Standardscaler, 2024. Accessed: 2024-11-04.
[4] Weka Developers. Cfssubseteval, 2024. Accessed: 2024-11-04.
[5] Scikit-Learn Developers. Sequentialfeatureselection, 2024. Accessed: 2024-11-04.
[6] Weka Developers. wrappersubseteval, 2024. Accessed: 2024-11-04.
[7] Scikit-Learn Developers. Scikit-learn: Machine learning in python, 2024. Accessed: 2024-11-04.
[8] Tensorflow Developers. Tensorflow, 2024. Accessed: 2024-11-04.
[9] Jake Coltman. Wittgenstein, 2024. Accessed: 2024-11-04.
[10] Weka Developers. The weka workbench, 2024. Accessed: 2024-11-04.
[11] XGBoost Developers. Xgboost classifier, 2024. Accessed: 2024-11-04.
[12] Scikit-Learn Developers. Gridsearchcv, 2024. Accessed: 2024-11-04.
[13] Scikit-Learn Developers. Kfold, 2024. Accessed: 2024-11-04.

VIII. Appendix

Table 8.1: Hyperparameter Grids and Selected Hyperparameters for Non-Probabilistic Classifiers

Metric	Hyper Parameter Grid Used	Hyperparameters Selected			
Classifier/Feature Subset Selection		**All Variables**	**Univariate Filter**	**Multivariate Filter**	**Wrapper Method**
K Nearest Neighbors	n_neighbors: [list from 1-20]	n_neighbors=19	n_neighbors=9	n_neighbors=11	n_neighbors=7
RIPPER	prune_size:[.2,.33,.5], k=[1,2,3] n_discretize_bins=[5,10,15]	k=8, n_discretize_bins=5 prune_size=.33	k=5, n_discretize_bins=11 prune_size=.5	k=5, n_discretize_bins=6 prune_size=.2	k=4, n_discretize_bins=7 prune_size=.4
Support Vector Machine	kernel:["rbf","sigmoid"], C:[0.1,.2,.5,.7,1,2,3,4,5] gamma: ['scale','auto']	kernel=sigmoid, C=.5 gamma=scale	kernel=sigmoid, C=.1 gamma=scale	kernel=rbf, C=.7 gamma=auto	kernel=rbf, C=3 gamma=auto
Decision Tree	criterion: ['gini','entropy','log_loss'] max_features:[list from 1,15] max_depth:[list from 1-15] min_samples_split:[list from 1-10]	criterion=log_loss max_depth=2 max_features=14 min_samples_split=8	criterion=gini max_depth=2 max_features=7 min_samples_split=7	criterion=entropy max_depth=4 max_features=4 min_samples_split=8	criterion=gini max_depth=3 max_features=4 min_samples_split=9
Neural Network (Feed Forward 2 hidden layers)	optimizer=[Adam(learn_rate=.001), Adam(learn_rate=.01)] epochs=[10,50,100,200,300] batch_sizes=[4,16,32,64]	optimizer= Adam(learn_rate=.01) epochs=10 batch_size=16	optimizer= Adam(learn_rate=.001) epochs=10 batch_size=4	optimizer= Adam(learn_rate=.001) epochs=300 batch_size=64	optimizer= Adam(learn_rate=.001) epochs=10 batch_size=4

Table 8.2: Hyperparameter Grids and Selected Hyperparameters for Probabilistic Classifiers

Metric	Hyper Parameter Grid Used	Hyperparameters Selected			
Classifier/Feature Subset Selection		**All Variables**	**Univariate Filter**	**Multivariate Filter**	**Wrapper Method**
Logistic Regression	C: [.5, .75, 1, 1.25, 1.5, 1.75, 2] penalty: ['L1', 'L2']	c=.5 penalty=L1	c=.5 penalty=L1	c=1 penalty=L2	c=.5 penalty=L1
Linear Discriminant Analysis	shrinkage:['auto', .01,.1, .3, .5] solver:['lsqr','eigen']	shrinkage=auto solver=lsqr	shrinkage=.3 solver=lsqr	shrinkage=auto solver=lsqr	shrinkage=auto solver=lsqr
Naive Bayes Bernoulli	alpha: [.2, .4, .6, .8, 1, 1.2, 1.4, 1.6, 1.8, 2]	alpha=2	alpha=2	alpha=2	alpha=2
TAN Bayes*	alpha=[.1,.5,1], search method=[K2, HillClimb, SimulatedAnnealing]	alpha=.5 search method=Simulated Annealing	alpha=1 search method=HillClimb	alpha=.5 search method=K2	All Results are Identical
Random Forest Classifier	n_estimators: [100,150,200] max_features:[1,3,6,9,12,15] max_depth:[1,3,6,9,12,15] min_samples_split: [1,3,5,7,10]	n_estimators=100 max_features=6 max_depth=1 min_samples_split=3	n_estimators=100 max_features=12 max_depth=3 min_samples_split=3	n_estimators=150 max_features=1 max_depth=1 min_samples_split=3	n_estimators=150 max_features=15 max_depth=3 min_samples_split=5
Bagging Classifier	n_estimators:[10,20,30,40,50,60,70,80.90,100] max_samples:[1-15]	max_samples=9 n_estimators=60	max_samples=8 n_estimators=90	max_samples=9 n_estimators=80	max_samples=12 n_estimators=30
Boosting Classifier	stack_method: [auto,predictproba,decision_function,predict]	stack_method=auto	stack_method=auto	stack_method=predict	stack_method=auto
XGBoost Classifier	n_estimators: [100,110,120 130 150,160, 170, 180, 190, 200] max_leaves:[1-15] max_depth:[1-15] learning_rate: [.001,.01,.02]	n_estimators=150 max_leaves=4 max_depth=2 learning_rate=.001	n_estimators=140 max_leaves=4 max_depth=2 learning_rate=.001	n_estimators=190 max_leaves=7 max_depth=3 learning_rate=.001	n_estimators=100 max_leaves=2 max_depth=1 learning_rate=.001