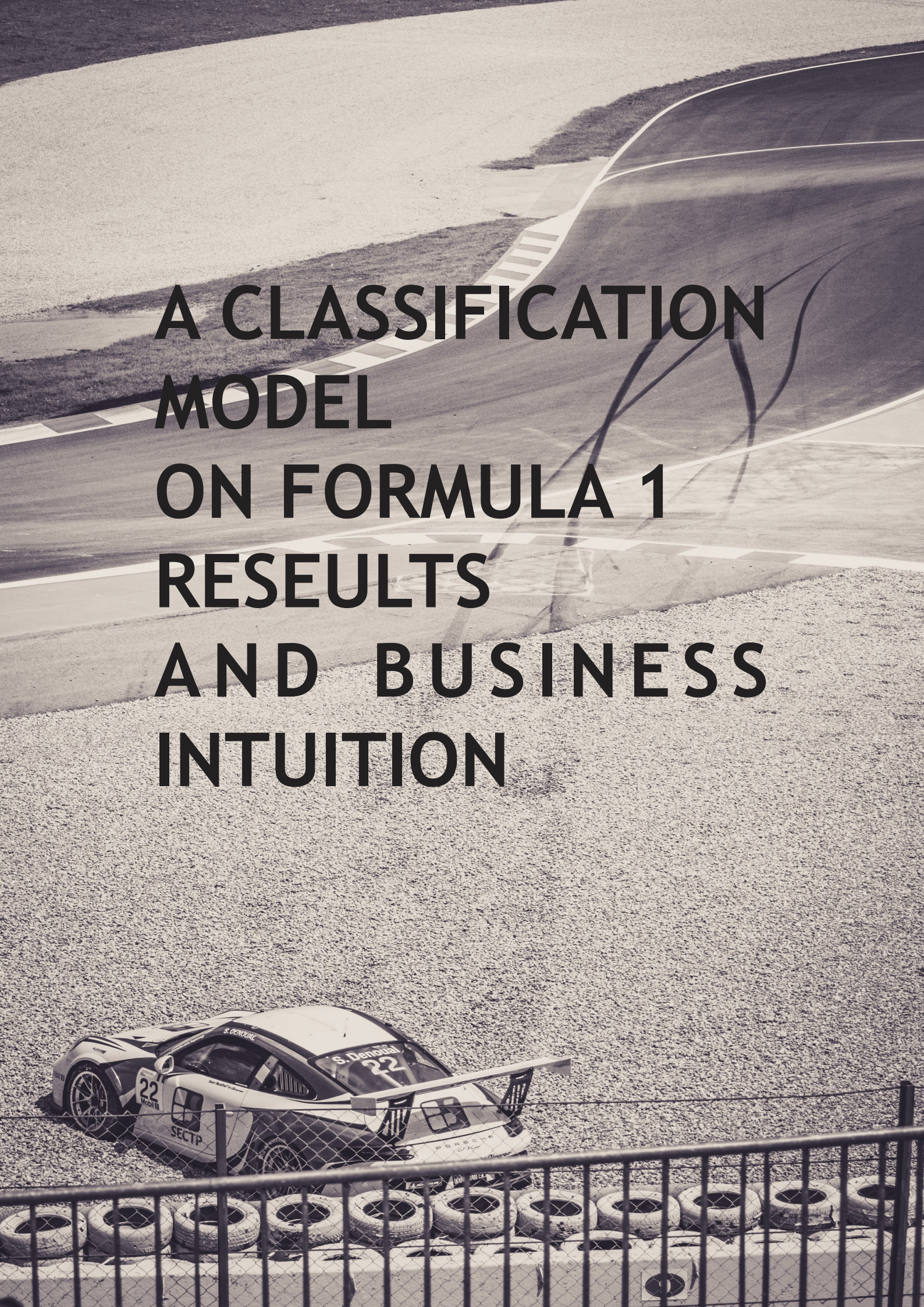
A CLASSIFICATION MODEL ON FORMULA 1 RESEULTS AND BUSINESS INTUITION



01 INTRODUCTION

Formula 1 presents itself as a sport game that is expensive while less attractive to outsiders. Millions of dollars are spent by each constructor on paying the racers, maintaining the research and development of racing cars, operating the team and so on. One may ask what is the point to watch a few machines with wheels running on the track? Unlike other sport games that the players are exposed and one can read their expressions and understand the tensions, what makes Formula 1 popular anyway for the racers are hidden behind the steering wheels? Last but not the least, why the racers would ever risk their lives for there is always the possibility that collision or other accidents happen?

The documentary “Drive to Survive” produced by Netflix unveiled Formula 1 from the angle of middle-sized constructors like Haas and make Formula 1 to be less untouchable to the general public. There are wins and there are losses; there are competitions on the track and there are tensions off the track; there are hopes and there are depressions; there are those rich constructors and there are poor constructors; there are all or nothing and there are dreams. Formula 1 is more than just machines of wheels racing without any traffic jam. To give an example, as Formula 1 starts to use biofuel, it can “showed what a hybrid could be and demonstrate another viable alternative energy source is possible” commented by F1 chief technical officer Pat Symonds.

In this report, we narrow down our scope to the very basic question asking by the audience as well as the constructors, what makes a racer or a car outperform the others and win? By saying “win”, we focus on the top 10 racers of each race instead of only on the top 1 or top 3 racers. There are two reasons for that. First, in recent years, racers like Lewis Hamilton always top the game and there is not much variance of the top 1 or 3 racers. Another reason is that top 10 racers can earn points which is highly relevant to their career and the survival of their teams. Just like there is rank for super models or billiards players, higher points mean higher rank for racers and constructors.

The Formula 1 datasets available provide rough information from 1950 to 2017 about the drivers, constructors, seasons, circuits, races, laps performance, duration of pit stops, standings of drivers and constructors and results of each race. Appendix 6-10 offers a dictionary to the dataset. Based on the datasets, we want to answer the following two questions:

1. What are those factors that make a racer to enter into the top 10 (and so earn points)?
2. What can a middle-sized team do with those factors in mind to survive in Formula 1?

In late 2009, Formula 1 officially banned refueling for the purpose of reducing costs and improving safety. Before refueling is taken with great care by the constructors for it is part of strategy towards victory for reasons like the weight of fuel also influences the speed of the car. Therefore, only data from 2010 to 2017 will be studied. Out of the considerations above, we want to look for a classification model to support in answering the questions.

02 DATA PROCESSING

In general, the dataset is clean and the usability is acceptable. Through the way of data processing, the following steps are followed.

Data Exploration

Appendix 1 to 3a show the distribution of the potentially relevant variables. A few highlights can be drawn from the plots:
1. Normal distribution is not followed by most of the variables. However, since a classification model is built out, there might not be necessary to normalize the data.
2. Cars at any grid position can make to the top 10.
3. Constructor Ferrari, McLaren and Red Bull dominate in having the greatest number of wins. And some constructor has never made to top 10 like Virgin.

Correlation Check Among Numerical Variables

To study the correlation between pairs of numerical variables: year of the game, month, grid, round, latitude, longitude, average day temperature, maximum day wind speed, historical average lap speed (i.e. up_to_date) and the outcome of the racing (i.e. target). Appendix 3.b shows that there is a strong positive relationship between round and month. One possible explanation for that is that the time for the game taken place at a particular location is mostly fixed. For example, Shanghai Grand Prix is normally held in March or April. Another pair that shows a rather close relationship is grid and outcome of the racing. It suggests a negative relationship which the classification model would explore later on.

Historical Speed of Each Racer

We also want to see if the average speed of each racer previously on a certain circuit would influence his performance on that circuit; for instance, whether the average lap speed of Carlos Sainz between 2010 and 2013 on Albert Park Grand Prix Circuit will influence his performance on that circuit in 2014. Data from 2009 is included for the purpose of calculation of average speed for the year of 2010. With the information provided by the dataset “laps”, historical average speed of each racer on a particular circuit can be denoted generally as:

$driver\ id:\ i = 1, 2, 3, \ldots\ldots, 842$
$year:\ j = 2009,\ 2010,\ 2011,\ \ldots\ldots,\ 2017$
$number\ of\ laps\ :\ n = 1, 2, 3, \ldots\ldots, n$
$circuit\ id:\ k = 1, 2, 3, \ldots\ldots, 73$
$y = speed\ of\ each\ lap$
$m = number\ of\ years\ that\ the\ racer\ participated\ in\ the\ game\ of\ a\ particular\ circuit$

$$Average\ lap\ speed\ of\ a\ racer\ on\ a\ particular\ circuit: x_{i,j,k} = \frac{\sum_{n=1}^{n} y_{i,n}}{n}$$

$$Historical\ Average\ Lap\ Speed:\ \overline{x}_{ijk} = \frac{\sum_{j=2009}^{j-1} x_{i,j,k}}{m}$$

If a racer does not attend a game held in a particular circuit in consecutive years, then $x_{i,j,k} = 0$. For example, if racer 20 only went to the game of Suzuka Circuit (circuit id: 22) in 2012 and 2014 but not 2013, then

$x_{20,2013,22} = 0$

However, there exists an absence of historical lap speed if a racer goes to a circuit for only once across 2009 to 2013. For example, Nico Rosberg raced in Baku City Circuit only in 2016 but no other years. In this case, instead of averaging out previous records, the historical average speed of Nico Rosberg in Baku City Circuit is set equal to the average speed he achieved in 2016 in Baku. The historical average speed, in this case, is denoted as:

$$\overline{x}_{i,j,k} = x_{i,j,k}$$

Add Wind and Temperature Data

During research, temperature and wind may have significant impact on the performance of Formula 1. Temperature may influence whether a racer can find and master the limit of a car. In addition to temperature, wind can be another factor behind the performance of racers. In 2017, driver Valtteri Bottas from Mercedes commented that “Formula 1’s cars can be more difficult to drive in windy conditions for which may increase the number of spins” and “the cars in general felt quite snappy”. Upon these considerations, we sorted out all races from 2010 to 2017 and their dates and locations from dataset “races” and then add average day temperature and maximum wind speed of each race. There are a total number of 177 races and the wind and temperature statistics are imported from the website Weather Underground.

Set Target Variable

The position order of each racer suggests the final result of each racer of a race. The smaller the number, the better the performance of the racer. If the position is smaller than or equal to 10, classify as 1; otherwise 0 for not making it to top 10.

Towards A Dataset Ready for Modeling

To prepare the dataset for later analysis, different datasets from original datasets and the derived ones like “Historical Average Speed of Each Racer” and “Temperature and Wind” are joined together. Appendix 10 gives more details about the columns in the final dataset and their interpretation.

There are 120 observations containing “NULL” values under the column of historical average speed (i.e. “up_to_date” in the dataset). The reason for that is that if a racer did not finish the race for reasons like collision or accident which are common in Formula 1, his laps speed may not be recorded. Since it does not comprise a proportionally large part in the datasets, these 120 observations are removed.

03 MODEL SELECTION

Feature Selection

Feature selection is a crucial step before we build the classification mode. We do feature selection in this project because:
In our final table which has already well processed, we have 26 variables. There are too many features in terms of the analysis and it is also possible that some of the features are irrelevant to our analysis.
Reducing some features will help us build classification model efficiently and also increase accuracy of the model since it can prevent overfitting problem.

In this case, we use the method of Principal Component Analysis (PCA), Random Forest, Logistic Regression, as well as intuition to check which features in our final table (Appendix 10) are most relevant to our analysis.

Intuition: Intuition refers that we can exclude any features are irrelevant to the analysis without using any technique tools. More specifically, ID numbers are nothing but a symbol to give each observation and identification; therefore, they are useless to the models, and we exclude them. Predictors only know after event happens should also be excluded from the table. For example, position text and position order describes the outcome of the race, laps describe how many laps does player run, status ID refers that what is the status of the player after the race, and points describe how many points does the player gains. Apparently, those predictors are only available right after players finish their race; hens, we should remove them from our final features.

PCA: PCA is a good way to begin to check with the relationship among all numerical predictors with target variable. It will give us a better visual way to determine how variables relate to each other and, thus, helps us choosing predictors. In this case, we include all numerical predictors as well as dummied target variable in the PCA plot and we have the results in Appendix 4.a. As we can see, target is our target variable, and year, latitude, and max wind speed of the day seem close to target; therefore, they might be useful when we predict the target. Historical average lap speed (up_to_date) and grid are in the exact opposite direction to target, hens, we may say these predictors are also highly correlated with target and may be quite useful.

However, longitude, average day temperature, as well as month and round are rather orthogonal to target, thus, we consider these predictors may not helpful. Especially for month and round, similar to the result from collinearity matrix shown in the above section, they are extremely close to each other indicating that these two predictors are significantly correlated, and for this fact, we will exclude them in our model.

Logistic regression: This classification regression method helps us better understand the correlation between categorical predictors. When categorical predictors such as constructor name, nationality, circuit name, location, and country are tested out on logistic regression at an early stage, we find that some of the constructor names are found to be linearly dependent with each other. The parameter of constructor Renault, for instance, can be fairly collinear with the other constructor names. Based on this finding, constructors are removed from the set of predictors.

Random forest: random forest is not only a classification algorithm for building models, but also a method for testing the importance of each predictor in models. After we exclude some predictors by using intuition and PCA, we put all other predictors in random forest model and get the result showing in Appendix 4.b. Accuracy score and gini score both indicate that grid is the most important feature; while in contrast, latitude is the least important feature. The importance plot makes us clearly interpret the ranking of importance of each feature.

By combining the outputs of the above methods, selected features are grid, nationality, historical average lap speed (up_to_date), year, circuit name and max wind speed of the day.

Classification Model Selection

Before we tune models, we split training and test dataset from final table. The split rate is 70% for training dataset and 30% for test dataset. We use training data to feed the following models.

Firstly, we choose decision tree algorithm with predictors chosen above to predict the target which is whether the player can get his/her points in the race (in other words, to check whether the player can enter the top 10). We start from building an overfitting tree which is a tree that contains large amounts of terminal nodes. Then we prune this complex tree by studying its out-of-sample (OOS) mean squared error (MSE) performance. The table as well as the plot in Appendix 5.b reveal how OOS MSE changes as the increase of the size of tree. The plot displays that at split of 9, this tree gives us an OOS MSE of 44.48%.

We also produce a random forest classification To select the best number of trees built in random forest, we compare the out-of-bag (OOB) MSE. We set 100000 trees and calculate the OOB MSE at every 1000 trees. The partial results show in Appendix 5.a. As the number of trees increases from 1000 to 5000, the random forest iterations improve the OOB MSE from 21.80% to 21.67%. After 5000 trees, the OOB MSE fluctuated around 22.00%. Therefore, we choose 5000 trees in our model.

04 RESULTS

The Best Model

Which model will be the best model for our analysis? The measurement of the performance of classification models is accuracy score. If a model has a higher accuracy score, we will choose it as our final model. To compare with it, we will use well-built decision tree and random forest from section 3 to predict the value of target variable in test dataset and then calculate the accuracy score of each model. We try different test dataset for 10 times and figure 1 are the results:

#	Accuracy Score - Decision Tree	Accuracy Score - Random Forest
1	77.94%	77.05%
2	79.54%	77.84%
3	77.15%	76.85%
4	76.95%	78.14%
5	76.45%	75.05%
6	76.75%	74.65%
7	75.85%	74.45%
8	76.65%	74.95%
9	76.65%	76.35%
10	76.75%	76.95%
Average	**77.07%**	**76.23%**
Standard Deviation	**1.02%**	**1.36%**

figure 1

The average accuracy score of decision tree is 77.07%, while the average accuracy score of random forest is 76.23%. In terms of the variance, the standard deviation of the results of decision tree is 1.02% while that of random forest is 1.36%. Since our dataset is kind of well structured and small, the results for both methods are not significantly different, and the average accuracy score of decision tree is even 0.84% more than that of random forest; however, when there is a huge and not well structured dataset to be analyzed, decision tree may have the problem of high variance of prediction results. Therefore, we still choose random forest algorithm as our best practical model.

Final Model Description

The description data of the final random forest model (at the 10th iteration) shows in figure 2:

Call			
randomForest(formula=target~grid+nationality+up_to_date+year+circuit_name+max_day_wind_speed,data=train,ntree=5000)			
Type of random forest	classification		
Number of trees	5000		
No. of variables tried at each split	2		
OOB estimate of error rate	23.47%		
Confusion matrix			
	0	1	class.error
0	940	308	0.2467949
1	240	847	0.2207912

figure 2

Which model will be the best model for our analysis? The measurement of The final model consists of 5000 different trees. Each tree has a randomly bootstrapped sample of observations and random subset of predictors. There are binary split for each node. OOB test indicates that our model is about 23.47% chances wrongly predict whether a player can be raking in top 10. Confusion matrix presents that when we use training dataset for this model, out of 2335 players (observations), 940 players are correctly predicted not entering the top 10, similarly, 847 players are correctly predicted entering the top 10. Therefore, we can calculate that under this training set, the prediction accuracy is around 76.53% which is close to the OOB estimate of error rate as well as the average accuracy score of random forest that we discussed above.

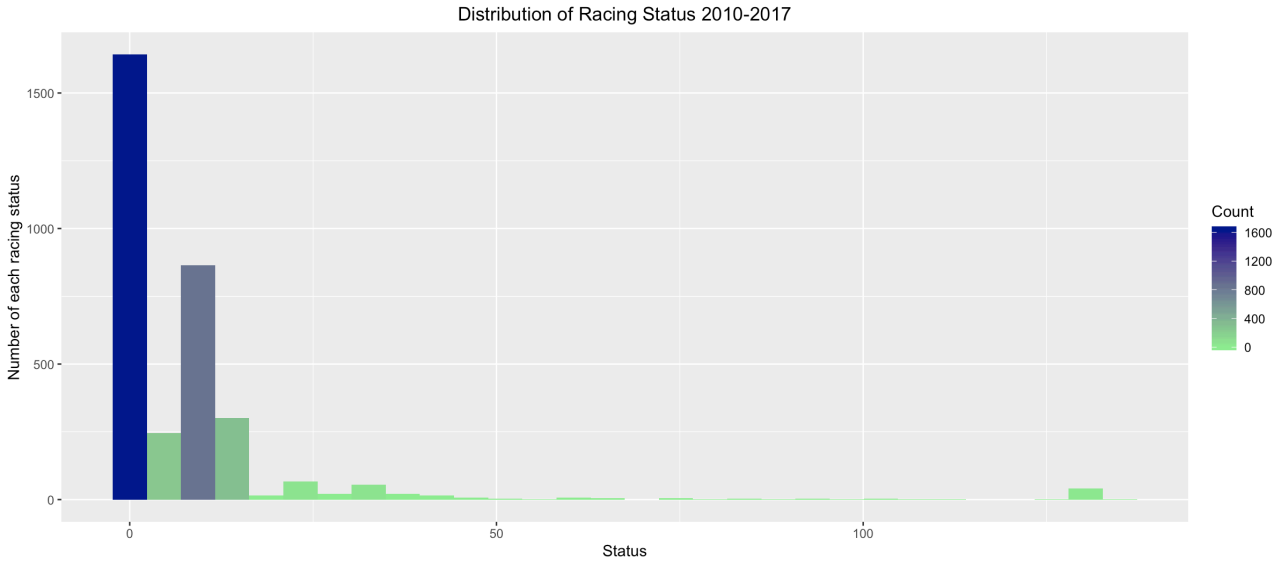
05 CONCLUSIONS

Currently, rich teams like McLaren, Ferrari and Red Bulls still dominate Formula 1. This makes Formula 1 offers less surprise to the general public as well as potential business partners. For middle-sized team like Haas with less budget, it would better to put more resources in performing well in qualifying games and get a bigger possibility in winning the grand prix. To show constant efforts in making progress would bring them with potential advertising collaborators. This will generate extra revenue source for them and in turn fuel their development in cars and growing competitive racers.

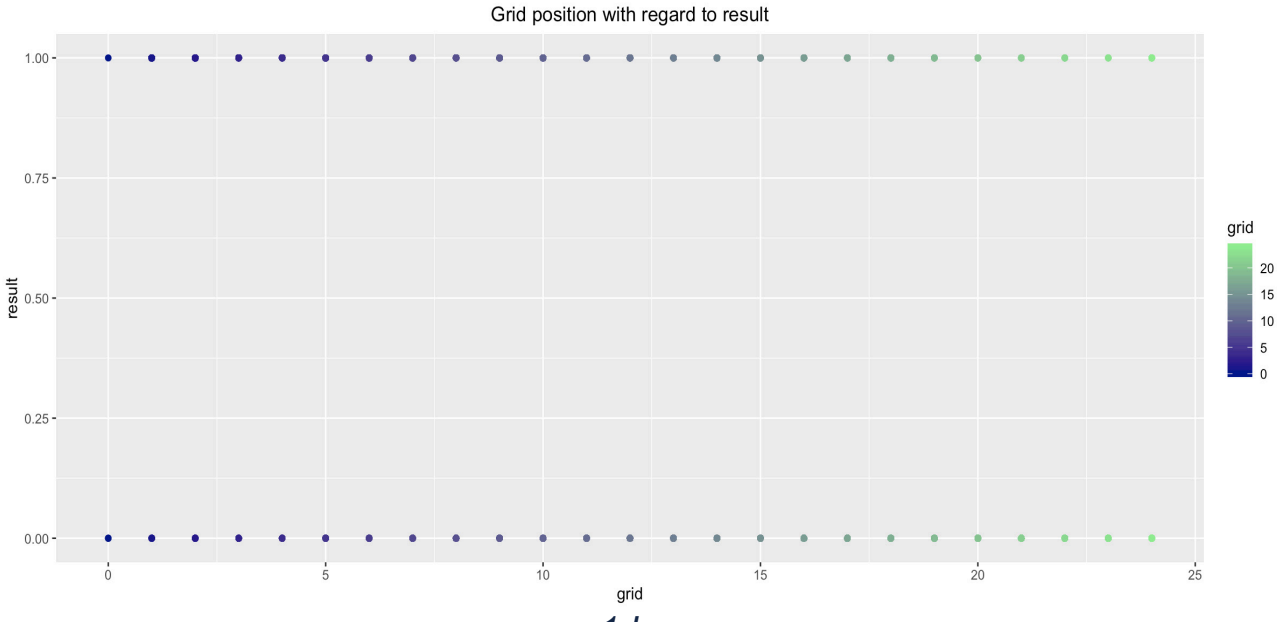
There are multiple limitations for this research. First, there would be better if detailed data about the drivers and circuits should be collected; for instance, how many DRS track available with regard to each circuit and how much is the slope of the turning corner. Second, it would be better to calculate the accident rate for each circuit so that to analyze if one circuit tend to be more dangerous than the other. Finally, with more data, it would offer an opportunity to clustering circuits and drivers so analysis over their features can be better summarized. Eventually, this will benefit the constructors in better analyzing their racers and tune strategy for different races.

06 APPENDIX

Appendix 1

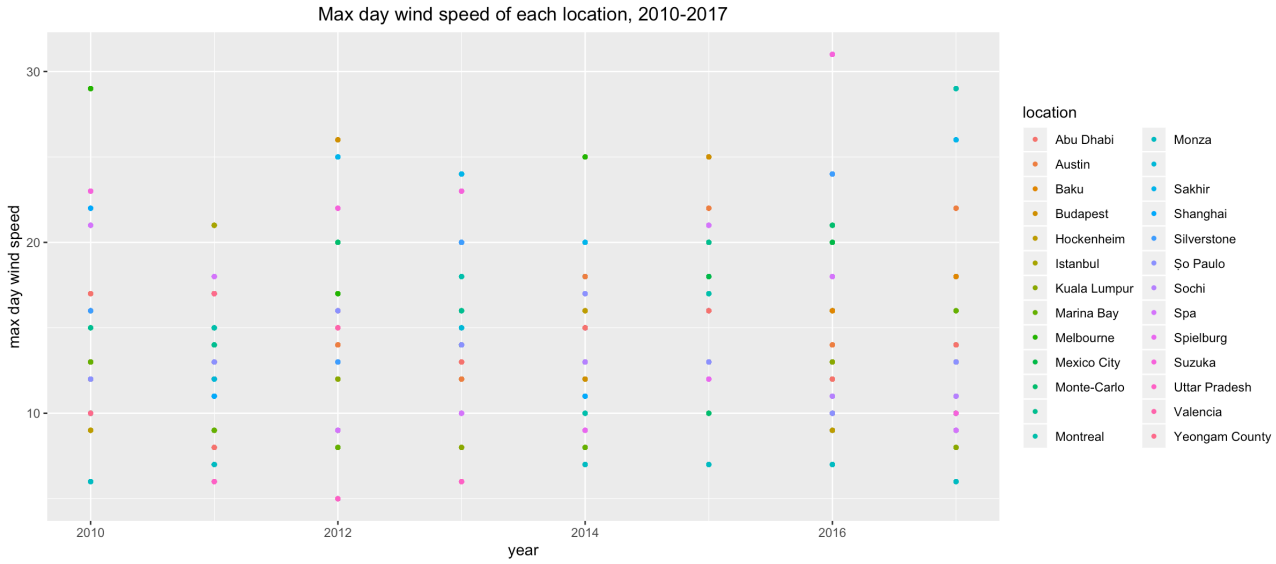


1.a

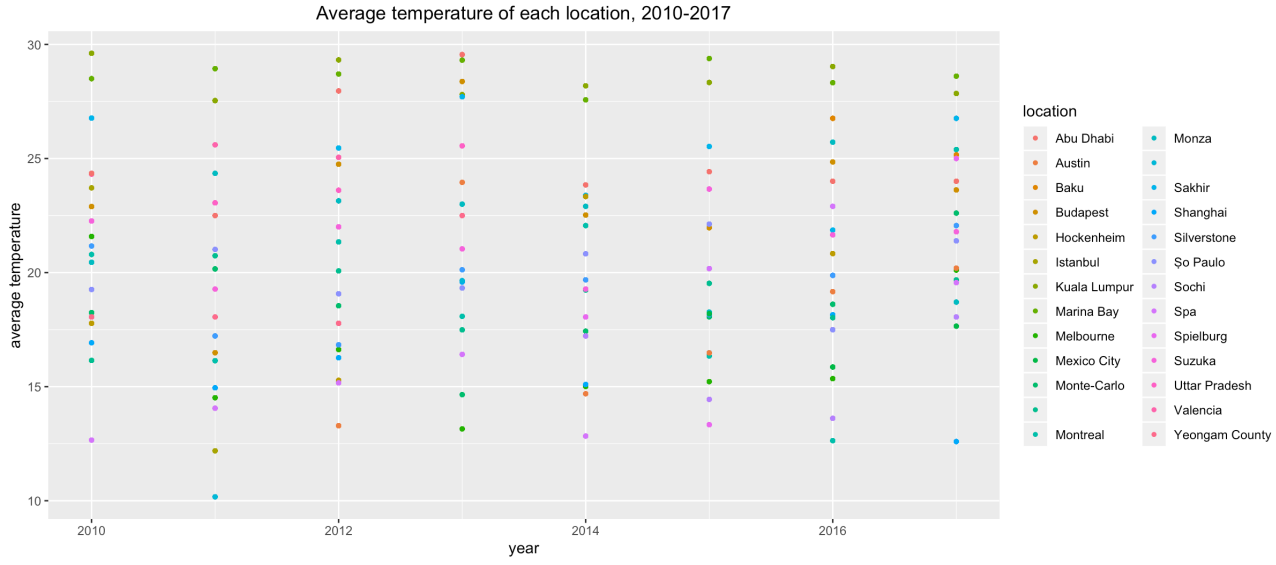


1.b

Appendix 2

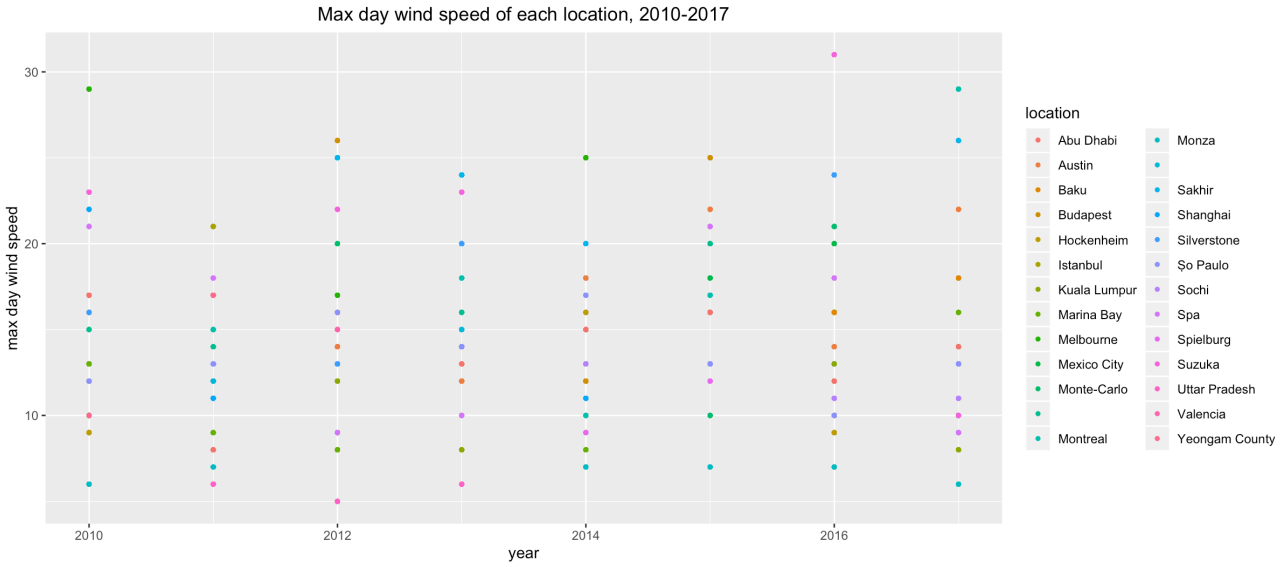


2.a

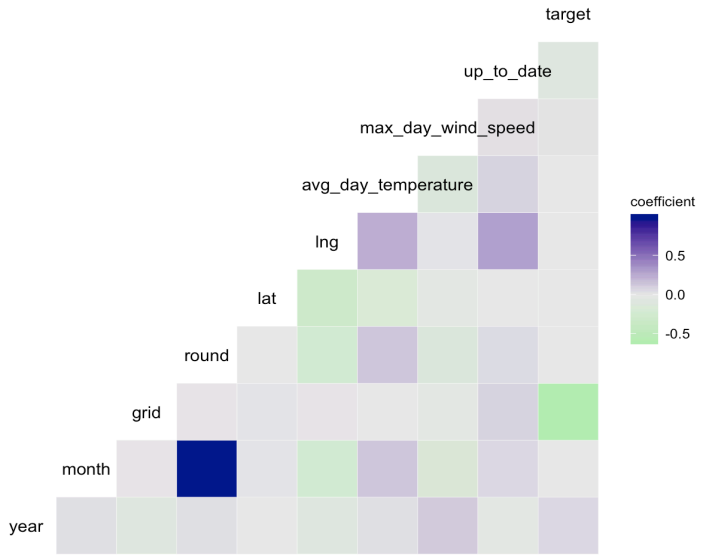


2.b

Appendix 3

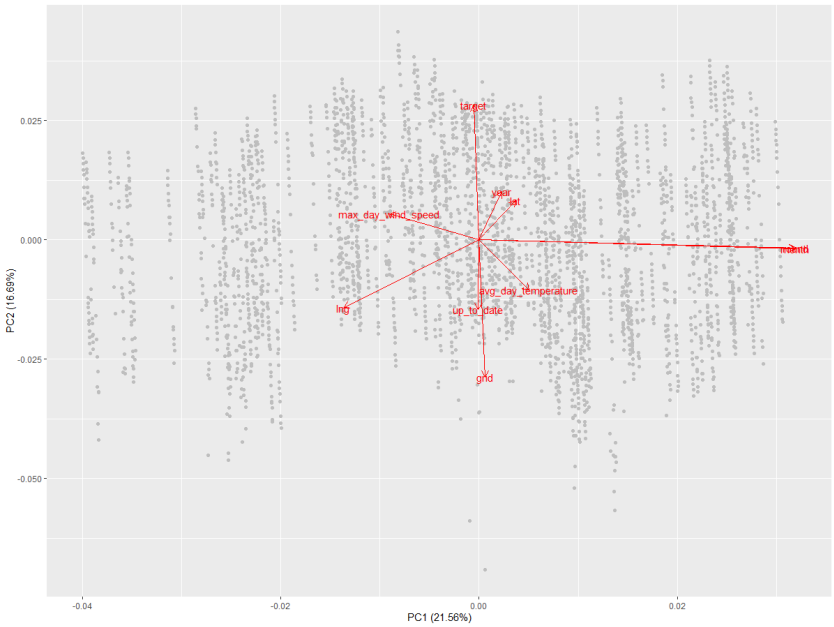


3.a

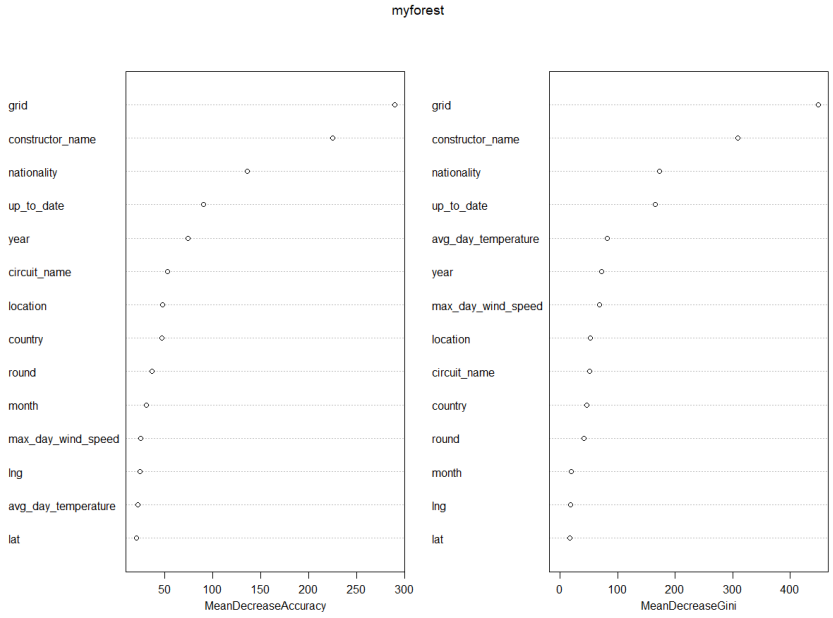


3.b

Appendix 4



4.a

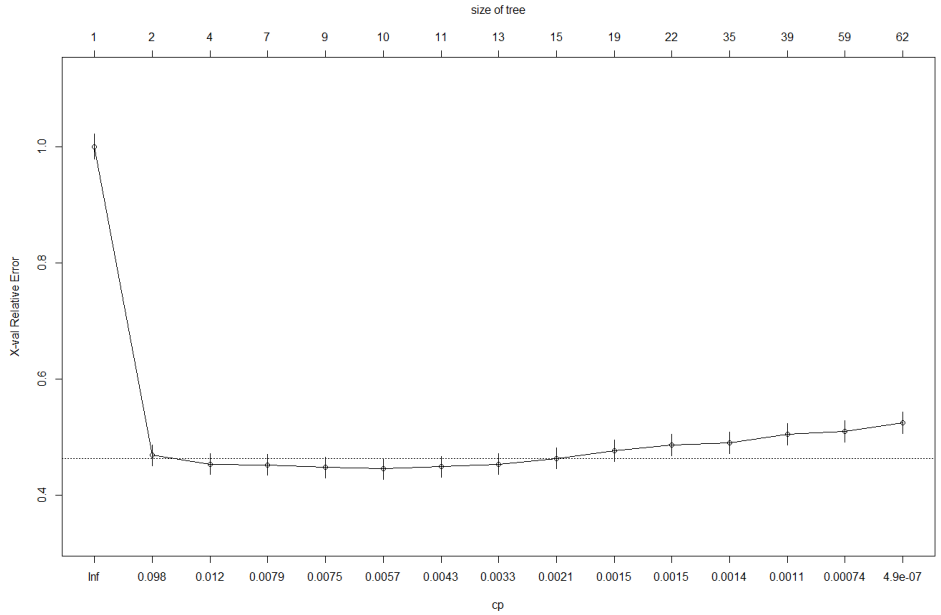


4.b

Appendix 5

#	CP	nsplit	rel error	xerror	xstd
1	5.39E-01	0.00%	1	100.00%	2.19%
2	1.77E-02	1	46.11%	46.83%	1.82%
3	8.15E-03	300.00%	0.42572	45.29%	1.80%
4	7.70E-03	6	40.13%	45.20%	1.79%
5	7.25E-03	800.00%	0.38587	44.75%	1.79%
6	**4.53E-03**	**9**	**37.86%**	**44.48%**	**1.78%**
7	4.08E-03	1000.00%	0.37409	44.84%	1.79%
8	2.72E-03	12	36.59%	45.29%	1.80%
9	1.59E-03	14	0.36051	0.46286	0.018097
10	1.51E-03	18	35.42%	47.65%	1.83%
11	1.45E-03	21	0.34964	0.48551	0.018407
12	1.36E-03	34	32.52%	48.91%	1.85%
13	9.06E-04	38	0.31975	0.50453	0.018654
14	6.04E-04	58	29.80%	50.91%	1.87%
15	4.00E-10	61	0.2962	0.52446	0.018901

5.a



5.b

Appendix 6

ntree	OOB	1	2
1000	21.80%	24.94%	18.30%
2000	21.88%	24.86%	18.57%
3000	21.93%	24.86%	18.66%
4000	21.93%	24.86%	18.66%
5000	**21.67%**	**24.78%**	**18.21%**
6000	21.76%	24.78%	18.39%
7000	21.71%	24.86%	18.21%
8000	21.67%	24.70%	18.30%
9000	21.80%	24.86%	18.39%
...	...	...	...
91000	21.97%	25.10%	18.48%
92000	21.88%	25.02%	18.39%
93000	21.84%	25.02%	18.30%
94000	21.84%	25.02%	18.30%
95000	21.88%	25.02%	18.39%
96000	21.93%	25.10%	18.39%
97000	22.01%	25.18%	18.48%
98000	21.97%	25.10%	18.48%
99000	21.97%	25.10%	18.48%
100000	2197.00%	25.10%	% 18.48%

Appendix 7

circuits		
Table description	Contains a list of every formula 1 circuit, including name, location and geographic data	
Feature description	circuitId	ID for each circuit. Primary key of the table
	circuitRef	Abbreviation of the circuit name
	name	Full name of the circuit name
	location	Location of the circuit
	country	Country of the circuit
	lat	Latitude of the location
	lng	Longitude of the location
	alt	Altitude of the location
	url	Website of the circuit

7.a

constructors		
Table description	Contains a list of every constructor team including name and nationality	
Feature description	constructorId	ID for each construtor. Primary key of the table
	constructorRef	Abbreviation of the constructor name
	name	Full name of the constructor name
	nationality	Country the constructor belonging to
	url	Website of the constructor

7.b

Appendix 8

drivers		
Table description	Contains a list of every Formula 1 driver, including full name, dob, and nationality	
Feature description	driverId	ID for each driver. Primary key of the table
	driverRef	Abbreviation of the driver name
	number	Number of the racing car of that driver
	code	Abbreviation of the driver name
	forename	First name of the driver
	surname	Last name of the driver
	dob	Birthday of the driver
	nationality	Country the driver belonging to
	url	Website of the driver

8.a

lapTimes		
Table description	Details the lap times for every race, including driver, lap number, position and time	
Feature description	raceId	ID for each race. Primary key of the table
	driverId	ID for each driver. Primary key of the table
	lap	Laps of each driver of each race
	position	Result of each race of each driver
	time	Total time of each of each driver
	milliseconds	Transform time into milliseconds

8.b

Appendix 9

races		
Table description	Details of every race, including year, date, time, circuit and round	
Feature description	raceId	ID for each race. Primary key of the table
	year	The race takes place in
	round	The nth race of each year
	circuitId	ID for each circuit. Primary key of the table
	name	Race name
	date	The day the race takes place
	time	The daytime the race takes place
	url	Website of the race

9.a

results		
Table description	Details of results of each race	
Feature description	resultId	ID for each result. Primary key of the table
	raceId	ID for each race. Primary key of the table
	driverId	ID for each driver. Primary key of the table
	constructorId	ID for each construtor. Primary key of the table
	number	Number of the racing car of that driver
	grid	Starting position of each driver (depends on qualifying game results)
	position	Result of each race of each driver
	positionText	Result of each race of each driver in text form
	positionOrder	Result of each race of each driver in number form

9.b

Appendix 10

	final table				
Table description	Contains joined features from other tables. Features in red are used in final model				
Predictors	resultId	raceId	driverId	circuitId	constructorId
	year	month	number	**grid**	positionText
	positionOrder	statusId	points	laps	round
	location	country	lat	lng	avg_data_temperature
	max_day_wind_speed	circuit_name	**up_to_date**	constructor_name	**nationality**
Target Variable (Response Variable)	**target**				

07 CODE

```
pit=read.csv("C:\\...\\pitStops.csv")

qualifying=read.csv("C:\\...\\qualifying.csv")

races=read.csv("C:\\...\\races.csv")

results=read.csv("C:\\...\\results.csv")

status=read.csv("C:\\...\\status.csv")

drivers=read.csv("C:\\...\\drivers.csv")

status=read.csv("C:\\...\\status.csv")

constructors=read.csv("C:\\...\\constructors.csv")

circuits=read.csv("C:\\...\\circuits.csv")

cons_result=read.csv("C:\\...\\constructorResults.csv")

driver_standings=read.csv("C:\\...\\driverStandings.csv")

seasons=read.csv("C:\\...\\seasons.csv")

laptimes=read.csv("C:\\...\\lapTimes.csv")


library(dplyr)

attach(races)

###join results and races

df1 = left_join(results,races,by="raceId")
```

```
###filter out results from 2010 and onwards

df1 = subset(df1,year >= 2010)

####extract month from races and add a new column

df2 = subset(races,year >= 2010)

df2$month = format(as.Date(df2$date),"%m")


########## Calculate historical lapTime of each racer regarding to each race ##########

###join dataset "laptimes" and "races"

df3 = left_join(laptimes,races,by="raceId")

df3 = subset(df3,year >= 2009)

###drop unnecessary data

df3 = df3 %>% select(-time.x,-date,-time.y)

###change to seconds

df3$milliseconds = (df3$milliseconds)/1000

###create a new table for the purpose of calculating historical data

circuit_annual_speed1 = df3 %>% group_by(year=df3$year,driverId=df3$driverId,circuitId=df3$circ
uitId,raceId=df3$raceId) %>% summarize(avg=mean(milliseconds))

circuit_annual_speed2 = circuit_annual_speed1 %>% arrange(circuit_annual_speed1$driverId,
circuit_annual_speed1$circuitId)
```

```
detach(races)

attach(circuit_annual_speed2)

count = 0

update = 0

for (i in 1:nrow(circuit_annual_speed2)){

  if (i==1){

    circuit_annual_speed2$up_to_date[i]=circuit_annual_speed2$avg[1]

  }else if (circuit_annual_speed2[i,3] == circuit_annual_speed2[i-1,3]){

    count = count + 1

    a = (update+(circuit_annual_speed2[i-1,5]))/count

    update = a*count

    circuit_annual_speed2$up_to_date[i]=a

   }else if (circuit_annual_speed2[i,3] != circuit_annual_speed2[i-1,3] & i != nrow(circuit_annual_
speed2)){

    count = 0

    update = 0

    if (circuit_annual_speed2[i,3] != circuit_annual_speed2[i+1,3]){

      circuit_annual_speed2$up_to_date[i]=circuit_annual_speed2[i,5]

    }else if (circuit_annual_speed2[i,3] == circuit_annual_speed2[i+1,3]){

      circuit_annual_speed2$up_to_date[i]=circuit_annual_speed2[i,5]

    }
```

```
###join with circuits to get the city name

df4 = left_join(df2, circuits, by='circuitId')

###add temperature and max wind into df4

Tempt_wind = read.csv("C:\\Users\\zewen\\Desktop\\MGSC661\\Final Project\\formula-1-race-data-19502017\\Tempt_wind.csv")

###set temperature to celsius

df4$avg_day_temperature = (5/9)*(Tempt_wind$temperature-32)

df4$max_day_wind_speed = Tempt_wind$wind.speed



################################### Create final tables ###################################

results_df4 = left_join(results, df4, by='raceId')

results_df4_races = left_join(results_df4, races, by='raceId')

###filter out results from 2010 and onwards

detach(circuit_annual_speed2)

attach(results_df4_races)

results_df4_races_2010_2017 = subset(results_df4_races,year.y >= 2010)

results_df4_races_2010_2017_clean = results_df4_races_2010_2017 %>% select(-time.x,-milliseconds,-date.x,-time.y,-url.x,-url.y

                                                                           ,-circuitRef,-name.y,-alt,-year.y,-round.y,-circuitId.y

                                                                           ,-name.x,-date.y,-time,-url)

detach(results_df4_races)

attach(results_df4_races_2010_2017_clean)

names(results_df4_races_2010_2017_clean)[names(results_df4_races_2010_2017_clean)=="year.x"] = "year"

names(results_df4_races_2010_2017_clean)[names(results_df4_races_2010_2017_clean)=="round.x"] = "round"

names(results_df4_races_2010_2017_clean)[names(results_df4_races_2010_2017_clean)=="circuitId.x"] = "circuitId"
```

```
###join with avg_speed for each curcuit for each driver

final_result_draft1 = left_join(results_df4_races_2010_2017_clean, circuit_annual_speed2, by=c('d
riverId','raceId','circuitId'))

###clean constructors table

detach(results_df4_races_2010_2017_clean)

attach(constructors)

constructors = constructors%>% select(-url,-X)

###join with constructor

final_table = left_join(final_result_draft1, constructors, by='constructorId')

###rename and rearrange final_table

detach(constructors)

attach(final_table)

final_table = final_table %>% select(-year.y,-avg,-constructorRef,-position,-fastestLap,-rank,-
fastestLapTime,-fastestLapSpeed)

names(final_table)[names(final_table)=="year.x"] = "year"

names(final_table)[names(final_table)=="name.x"] = "circuit_name"

names(final_table)[names(final_table)=="name.y"] = "constructor_name"

col_order = c('resultId','raceId','driverId','circuitId','constructorId','year','month','number','grid','positio
nText','positionOrder'

              ,'statusId','points','laps','round','location','country','lat','lng','avg_day_temperature','max_
day_wind_speed'

            ,'circuit_name','up_to_date','constructor_name','nationality')

final_table = final_table[, col_order]

###drop null value

final_table = final_table[up_to_date != "NULL",]

###add target variable

final_table$target = ifelse(final_table$positionOrder <= 10, 1, 0)
```

```
###change variable class

final_table$up_to_date = unlist(final_table$up_to_date)

final_table$target = as.factor(final_table$target)

final_table$month = as.integer(final_table$month)

final_table$location = as.character(final_table$location)

final_table$constructor_name = as.character(final_table$constructor_name)

final_table$location = as.factor(final_table$location)

final_table$constructor_name = as.factor(final_table$constructor_name)

################################## Plot distribution of the variables ##################
####################

library(ggplot2)

library(ggfortify)

###histogram of month

ggplot(data=final_table,aes(x=month))+geom_histogram(aes(fill=..count..))+

  scale_fill_gradient("Count",low="light green",high="dark blue")+

  scale_x_continuous("month", labels = as.character(month), breaks = month,)+

  labs(y="number of races held", x = "month")+ggtitle("Distribution of Games 2010-2017")+

  theme(plot.title = element_text(hjust = 0.5))

###histogram of statusId

ggplot(data=final_table,aes(x=statusId))+geom_histogram(aes(fill=..count..))+

  scale_fill_gradient("Count",low="light green",high="dark blue")+

   labs(y="Number of each racing status", x = "Status")+ggtitle("Distribution of Racing Status 2010-
2017")+

  theme(plot.title = element_text(hjust = 0.5))

## plot of average day temperature

ggplot(data=final_table,aes(x=year,y=avg_day_temperature,color=location))+

  geom_point(size=1)+labs(y="average temperature", x = "year")+

  ggtitle("Average temperature of each location, 2010-2017")+

  theme(plot.title = element_text(hjust = 0.5))
```

```
###plot of max wind speed
ggplot(data=final_table,aes(x=year,y=max_day_wind_speed,color=location))+
  geom_point(size=1)+labs(y="max day wind speed", x = "year")+
  ggtitle("Max day wind speed of each location, 2010-2017")+
  theme(plot.title = element_text(hjust = 0.5))
###histogram of laps
ggplot(data=final_table,aes(x=laps))+geom_histogram(aes(fill=..count..))+
  scale_fill_gradient("Count",low="light green",high="dark blue")
###histogram of up_to_date
ggplot(data=final_table,aes(x=up_to_date))+geom_histogram(aes(fill=..count..))+
  scale_fill_gradient("Count",low="light green",high="dark blue")+
  labs(y="Number of each historical speed", x = "Historical Average Lap Speed")+
  ggtitle("Distribution of Historical Average Lap Speed 2010-2017")+
  theme(plot.title = element_text(hjust = 0.5))
###plot of grid with respect to target
ggplot(data =final_table, mapping = aes(x=grid, y=target,color=grid))+
  geom_point()+scale_color_gradient(low="dark blue", high="light green")+
  labs(y="result", x = "grid")+ggtitle("Grid position with regard to result")+
  theme(plot.title = element_text(hjust = 0.5))
###plot of constructor with respect to target
final_table$target = ifelse(final_table$positionOrder <= 10, 1, 0)
test01= final_table %>% group_by(constructors=final_table$constructor_name) %>% summarize(n
umber=sum(target==1))
ggplot(data=test01,aes(x=constructors,fill=number))+geom_col(mapping=aes(y=number))+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  scale_fill_gradient("number",low="light green",high="dark blue")+
  ggtitle("Number of wins achieved by each constructor, 2010-2017")+
  labs(y="number of wins")+theme(plot.title = element_text(hjust = 0.5))
```

```
###correlation Matrix

numerical_vars_with_target = final_table[,c(6,7,9,15,18,19,20,21,23,26)]

numerical_vars_with_target$target = as.integer(numerical_vars_with_target$target)

library(GGally)

ggcorr(numerical_vars_with_target)+scale_fill_gradient2(low="light green",high="dark blue",mid="gray91")



#################################### feature selection ##############################
#########

###logistic Regression

detach(final_table)

attach(final_table)



final_table$target = as.factor(final_table$target)

mlogit=glm(final_table$target~grid+up_to_date+year+circuit_name+nationality+max_day_wind_speed,

          family="binomial")

summary(mlogit)



test_data=subset(final_table,select=c("grid","up_to_date","year","circuit_name",

                                      "nationality","max_day_wind_speed"))

test_logit= predict(mlogit,test_data,type="response")

test_logit = ifelse(test_logit>0.5,1,0)

(nrow(final_table)-sum(test_logit==final_table$target))/nrow(final_table)

library(randomForest)

###random forest to check importance of predictors

myforest=randomForest(target~year+month+grid+round+location+country+lat+lng+avg_day_temperature+max_day_wind_speed
```

```
+circuit_name+up_to_date+constructor_name+nationality,ntree=10000,data=final_
table,importance=TRUE)

importance(myforest)

varImpPlot(myforest)

###numerical feature selection

###PCA for numerical correlation

numerical_vars_no_target = final_table[,c(6,7,9,15,18,19,20,21,23)]

numerical_vars_with_target = final_table[,c(6,7,9,15,18,19,20,21,23,26)]

numerical_vars_with_target$target = as.integer(numerical_vars_with_target$target)

pca1 = prcomp(numerical_vars_with_target, scale=TRUE)

autoplot(pca1,data=numerical_vars_with_target,loadings=TRUE,col='grey',loadings.label=TRUE)

pca1

pca2 = prcomp(numerical_vars_no_target, scale=TRUE)

autoplot(pca2,data=numerical_vars_no_target,loadings=TRUE,col=ifelse(numerical_vars_with_
target$target==2,'green','light blue'),loadings.label=TRUE)

pca2

###then we found that round and month extremly correlated

###from PCA, lodistic regression,and Random Forest

###we choose grid,constructor_name,nationality,up_to_date,year,circuit_name,max_day_wind_
speed

###we find constructor_name and nationality has higher correlation from logistic reression ######
############################## model selection ####################################
###############

###split train/test in 70%/30%

trainIndex = sample(1:nrow(final_table), 0.7 * nrow(final_table))

train = final_table[trainIndex,]

test = final_table[-trainIndex,]
```

```
 ###decision tree

detach(final_table)

attach(train)

library(tree)

library(rpart)

library(rpart.plot)

overfittedtree = rpart(target~grid+nationality+up_to_date+year+circuit_name+max_day_wind_
speed

          ,control = rpart.control(cp = 0.0000000004))

printcp(overfittedtree)

plotcp(overfittedtree)

min(overfittedtree$cptable[,"xerror"])

overfittedtree$cptable[which.min(overfittedtree$cptable[,"xerror"]),"CP"]

###random forest

randomforest = randomForest(target~grid+nationality+up_to_date+year+circuit_name+max_day_
wind_speed

                    ,ntree = 100000,data = train, do.trace = 1000, importance = TRUE)

randomforest
############################################## Results ############################
################################

###compare by testing accuracy score

detach(train)

attach(test)

###accuracy score for decision tree

decisiontree_final = rpart(target~grid+nationality+up_to_date+year+circuit_name+max_day_wind_
speed,control = rpart.control(cp = 0.004528986),data = train)

t_pred_dt = predict(decisiontree_final,test,type = 'class')

error_rate_dt = (nrow(test)-sum(t_pred_dt==test$target))/nrow(test)

accuracy_score_dt = 1 - error_rate_dt

accuracy_score_dt
```

```
###accuracy score for random forest

randomforest_final = randomForest(target~grid+nationality+up_to_date+year+circuit_name+max_
day_wind_speed,ntree = 5000,data = train)

t_pred_rf = predict(randomforest_final,test,type='class')

error_rate_rf = (nrow(test)-sum(t_pred_rf==test$target))/nrow(test)

accuracy_score_rf = 1 - error_rate_rf

accuracy_score_rf

###random forest stat

randomforest_final
```