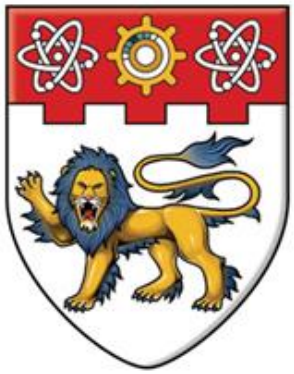


NANYANG TECHNOLOGICAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CZ4032 Data Analytics and Mining

Team Project: Formula 1 Data Set

Group 2

Siek Ming Jun (U1820369F)

Poh Ying Xuan (U1821489B)

Hoo Bing Yuan (U1823670H)

Shearman Chua (U1820058D)

Vivian Siow (U1823920J)

Saw Jia Yi (U1822525F)

Supervising Teaching Assistance: Emadeldeen Eldele

Submission Date:

1 Contents

2 Abstract

Formula One (F1) is a motorsport with an extensive global viewership. Multiple Constructors (teams) and their drivers compete annually for the Constructor's Championship and the Driver's Championship respectively. By winning the Constructor's Championship, the Constructor will be allocated a larger budget to be used in the next season of F1. As such, Constructors should come up with a race strategy that would provide them the best advantage to win the championship.

Therefore, our project aims to make use of publicly available F1 data and perform real world Data Analytics and Mining tasks to help derive new insights that can help Constructors devise better race strategies. The project will utilize data from Formula 1 seasons from the year 1950 to 2017, available on Kaggle.

In the first part of the project, we will be studying various classification models for binary classification to predict if a Driver is able to be in a Point Winning Position (PWP) for a particular race. By predicting if a driver will be in PWP, we could determine the race pace and strategy the Constructor could execute for that race.

In the second part of the project, we will be utilizing a neural network to perform regression to predict what will be the fastest lap time for a particular race. This is to allow teams to try to hit that timing early on in the race and focus on getting into a PWP afterwards. An additional point is gained by achieving the fastest lap time in a race, which is an added advantage towards winning the championships.

In the third part of the project, we will be utilizing a neural network to perform regression to predict the final Driver's Standing of a Driver-Constructor pair. After each season, Constructors could bid for their driver of choice. By predicting which driver performs better with the Constructor, Constructors could bid for that driver which could improve the chance of winning the championships.

In the fourth part of the project, we will be utilizing a neural network to perform a multiclass classification on which race classification a Driver will achieve for a particular race, the classes namely, Did Not Finish (DNF), podium, Point Winning Position and Finish. This classification model will serve to be an extension of the first part of the project to help improve race strategy for Constructors.

This report will also highlight the various results obtained by the various prediction models for the various Data Mining tasks that we have defined. From the results, we were able to gain valuable insights into the intricacies of F1 and some of the prediction models we have developed are of decent test accuracy and can be used by the Constructors in F1 to help devise better race strategies which will aid them in winning the Constructor's Championship.

3 Introduction

Formula One (F1) is one of the most well-known international motorsports. An F1 season consists of a series of races, known as Grand Prix (GP), which take place worldwide on purpose-built circuits and public roads. The results of each race are evaluated using a points system to determine two annual World Championships: Driver Championship (DC) and Constructor Championship (CC). Every season, F1 teams, also known as Constructors, compete using cars with chassis built by themselves. Some examples are major car companies like Ferrari. Every Constructor in F1 must run two cars in every session in a GP weekend, and every Constructor may use up to four drivers in a season.

3.1 Grand Prix

3.1.1 Practice Session

Practice sessions are conducted on Fridays where drivers could get accustomed to the circuit.

3.1.2 Qualifying Session

Qualifying sessions are conducted on Saturday. This will determine the starting grid for the actual race. Qualifying sessions consist of three periods, known as Q1, Q2, and Q3. In each period, drivers run laps with the slowest drivers being knocked out of that period. The rest of the drivers continue on to the next period. The drivers that were knocked out solidifies their grid position for the race based on their lap time and will not be participating in the next period. Q1 determines the bottom 5 placing, Q2 determines the next 15 while Q3 determines the final 5. Drivers are allowed to run as many laps as they wish within each period and only the fastest timing will be considered. In the end, the drivers starting grid positions are determined.

3.1.3 Actual Race

The actual race will be conducted on Sunday. Drivers start at the starting line based on starting grid position determined by their results in Qualifying. The drivers then raced for a predetermined number of laps on the track with the final race standings determined when the drivers crossed the finishing line after completing all laps. In the current format, points are awarded to the first 10 drivers that finished the races, therefore the top 10 positions in a race are also known as Point Winning Positions (PWPs) . These points contribute to the DC and CC where the Constructor points for each race are just the points obtained by the Constructor's two drivers for that particular race.

4 Problem Description

4.1 Motivation

The main goal of a Constructor is to win the CC in order to be allocated a bigger budget in the next F1 season. The main goal of a Driver is to win points for each race in order to gain points towards winning the DC. These points earned will also contribute towards the driver's own Constructor points tally, which is used for the CC. In order to win, the Constructor will require the best race strategy for the season in order to increase the chances of winning.

To win the DC, drivers must be placed in Point Winning Position (PWP) for each race. Points are allocated based on the position. The driver points are accumulated for each race and after

the final race, the driver with the highest points in the Final Drivers' Standings will win the DC.

To win the CC, Constructors must accumulate points. The points earned by the Constructor's drivers in each race are added to the Constructor points. The Constructor points are accumulated for each race and after the final race, the Constructor with the highest Constructor points in the Final Constructor Standing will win the championship.

Therefore, if a Constructor is able to predict if a driver will be in PWP for each race, they could determine the type of strategy to execute for a particular race.

In F1, a way to earn additional points is that a driver achieves the fastest lap time for the race and finishes in PWP. The driver will be awarded one additional point which could affect the overall Drivers' Standings. However, it is hard to determine what will be the fastest lap time for a particular race. If this could be predicted, the Constructor could notify the driver of the predicted time before the race and he could try to hit the time early on, instead of waiting until the end of the race.

At the end of each season, Constructors are able to bid for new drivers. In order to prepare for the next season of F1, one of the strategies that the Constructor could execute is to bid for better drivers to join them. By determining how a specific driver will perform when racing with a particular constructor, the Constructor is able to know which driver to bid for in order to perform better in the next season.

4.2 Problem Definition

Objective: Use Data Analytics and Mining concepts to devise a race strategy that will allow the Constructor to win the CC.

4.2.1 Problem 1

To earn driver points, drivers must finish in a PWP in a race. However, it is not easy to predict whether a driver will finish in PWP. In addition, the amount of F1 power unit elements that teams are allowed during the season is limited. During the 2019 F1 season, drivers are allowed to use only 3 ICE, 3 TC, 3 MGU-H, 2 MGU-K, 2 ES and 2 CE units. Drivers that use more power unit elements will receive grid penalties in the next F1 race. Therefore, if a Constructor is able to predict that they are not able to finish in a PWP for a particular race, they can limit the usage and performance of the power unit, in order to preserve the life span of power unit elements for future races, avoiding grid penalties which could be a better strategy in the long run.

4.2.2 Problem 2

By finishing the race in PWP and achieving the fastest lap time, the driver is awarded an additional point towards his driver points. However, drivers normally achieve their fastest lap time during the last few laps of the race. If we could predict the fastest lap time for a particular race in the season, the Constructor could inform the driver of what time to hit before the race. The driver could try to hit this timing early on and focus on finishing in PWP afterwards.

4.2.3 Problem 3

When bidding for new drivers, it is hard to determine whether a driver is able to achieve a better performance using a new constructor. If we could predict how a driver performs when they switch to a new constructor, the Constructor could select the driver that performs the best and bid on them after the season ends. This would be one step towards the team's goal of winning the CC.

4.2.4 Problem 4

As an extension to Problem 1, instead of performing a binary classification to predict if a driver will finish in a PWP in a race, we will perform a multiclass classification to predict the results of a driver in a race into more specific categories. This will also allow us to better understand which category of race classification is more predictable and which categories are harder to predict. We will also be utilizing another dataset for Formula One, retrieved from Kaggle.

5 Approach

5.1 Methodology

5.1.1 Software Requirements

For this assignment, we will primarily be using Jupyter for our EDA, Data Cleaning and Data Mining.

5.1.2 Project Components

There are 3 components in our project:

1. Report Documentation: Documents our journey of this assignment
2. Source Code: Contains the datasets which are in csv format and Jupyter files to execute the codes
3. README.md file: Provides instructions on how to execute the Jupyter files

5.1.3 Sources and Datasets

The chosen datasets contain data of F1 seasons from the year 1950 to 2017. The data consists of tables describing constructors, race drivers, lap times, pit stops and more. This original dataset is provided by Chris G and is made publicly available on Kaggle.

In addition, for Problem 4, we will be utilizing an additional dataset provided by Diana and is made publicly available on Kaggle.

6 Data Pre-processing

6.1 Datasets Used

After analysing the datasets, we decided to drop csv files that do not provide useful information. These are constructors, constructorResults, constructorStandings, lapTimes, pitStops and seasons.

The reasons are as follows:

- constructors, constructorResults and constructorStandings: Since Constructor point is equivalent to the cumulative driver points and constructorId exists in other datasets such as results, these three datasets were classified as duplicate information.

- lapTimes: This was considered as real-time race data and will cause data leakage if we were to utilize them.
- pitStops: Pit stops strategy depends on the compound of the tires and since we do not have tire data, it is difficult to use pit stop information.
- seasons: This dataset consists of redundant information.

We kept the following csv files and the reasons are as follows:

- results: Contains the results of each race for each GP for each season.
- circuits: Contains circuit information for circuits used during GP.
- drivers: Contains all drivers information such as name and driver id.
- driverStandings: Contains the cumulative points for each driver in each race in each season. It also includes whether the driver won the race and their final position.
- qualifying: Contains the qualifying times for all 3 periods during the qualifying round.
- races: Contains race information for each season
- status: Contains the status that a driver and Constructor could have at each race.

6.2 Data Pre-Cleaning

Before we start, we conducted a pre-cleaning of the datasets. This would make it easier for us to conduct our experiments. To summarise, we did the following:

1. Removed redundant attributes from most dataset, e.g: driverRef, url.
2. Convert time string from minute-second format to milliseconds.
3. Merge driver's name into 1 attribute instead of 3.

Once this is completed, we export the datasets into a new csv file with "_cleaned" appended to the end of the name. These files are stored in a folder called Cleaned Data.

6.3 Data Insertion

6.3.1 Adding of Additional Information in circuit.csv

While studying the datasets, we discovered that the circuits.csv does not provide important information about the circuits. The original file only has basic information of where the circuits are located and not important information such as lap length and race length. This additional information could be easily obtained from trustworthy websites such as https://www.racefans.net/. As we believed that this information is needed for our Data Mining tasks, we manually inserted the following information into the circuit_cleaned.csv file based on information from the website.

- Turns, Lap Length, Race Laps, Race Distance, Max. Speed, DRS Zone, Full Throttle Percentage, Longest Flatout Section, Downforce Level, Gear Changes per Lap.

6.3.2 Categorizing Values in status.csv

In the status.csv file, we have 136 different status that a driver could have for each race in each season. Most of the status means the same thing and thus, could be binned into one category. As such, we have decided to manually bin the values inside status.csv into these 4 categories:

- Finished, Team, Track, Constructor

6.4 Data Integration

In order to obtain the input features combinations that are required for each of the defined data mining tasks, we have to integrate the various datasets, as not all required input features are in one dataset.

6.4.1 Problem 1

For this problem, we will require information that is available to the Constructors before the start of the race such as the starting Grid Position of the Drivers, the ID of the Constructor the Drivers are driving for, as well as information regarding the circuit used for that particular race.

Therefore, these files are imported into Jupyter as dataframes and are merged together to give us the input features we require for Problem 1:

- results_clean.csv (Drivers race results for each race)
- status_clean.csv (Status code and their meanings)
- races_clean.csv (Year and GP information)
- circuits_clean.csv (Information regarding the circuits used for racing)
- drivers_clean.csv (Drivers information)

6.4.2 Problem 2

For this problem, we will require information that is available to the Constructors before the start of the race. We will require a dataset that consists of the circuit's information for each race, fastest lap time set on that circuit from the previous years, as well as the fastest qualifying time set for that race.

Therefore, these files are imported into Jupyter as dataframes and are merged together to give us the input features we require for Problem 2:

- qualifying_clean.csv (Qualifying results of Drivers before each race)
- races_clean.csv (Year and GP information)
- circuits_clean.csv (Information regarding the circuits used for racing)
- results_clean.csv (Drivers race results for each race)

6.4.3 Problem 3

For this problem, we will be reusing the dataframe we created in Problem 1 with the added new features (Driver error rate, Constructor error rate and Circuit error rate).

6.4.4 Problem 4

For this problem, we wanted to see if other information regarding the races can help improve race classification accuracy. Therefore, we decided to include another dataset from Kaggle regarding F1 and name it as dataset.csv. We will be using the dataframe we created in Problem 1 with the added new features (Driver error rate, Constructor error rate and Circuit error rate), and new features from the new dataset from Kaggle. For the new dataset from Kaggle, we have the following new race information:

- statusId : 1 is Finished, 0 is Did Not Finish
- Did not finish, Podium, Pos 4 to 10: Probability of a driver in each of the race classification based on past 3 years' race results for that circuit

- Sc: Probability of a Safety Car
- Wet: 1 is wet weather forecast, 0 is dry weather forecast
- pitStop timing (avg): average pitstop timing of driver
- pitStop timing prop(driver): proportion of pitstop time as compared to race time

6.5 Data Splitting

For question 3, we have to split the data in order to create our train and test set. We split the dataframe by grouping the items by the 'driverId'. With this GroupDataFrame, we created a train and test set with the following conditions:

- For each item in a group:
    1. If a driver only drives 1 Construction Throughout 2004 to 2017, append the whole group into the train set.
    2. If a driver drives multiple Constructor Throughout 2004 to 2017:
        a. If it is not the last item:
            i. Append item into train set
        b. If it is the last item:
            i. Append item into test set

After splitting, we exported the train and test set to train_q3.csv and test_q3.csv respectively.

6.6 Data Transformation

Some of the data mining tasks require us to transform the data rows into a different form to fit the task at hand. This includes data aggregation to find the mean values of data columns by race, as well as mean values of data columns by Constructors and Drivers.

We normalized the data by changing the values of numeric columns in the dataset to a common scale without distorting differences in the ranges of values. This will bring all the variables to the same range so that no input feature is more important as a predictor than the others to the classification model.

6.6.1 Problem 2

Data Transformation is required for Problem 2 as we require a new data column which contains the fastest qualifying time set by each driver out of the 3 qualifying sessions in a particular race.
Data transformation for Problem 2 is executed in this order:

- Generate "fastest_q" column from the fastest timing within q1, q2 and q3.
    - Insert fastest timing between q1, q2 and q3 into column.

6.6.2 Problem 3

We group both train and test set by 'driverId', 'constructorId' and 'year' and we aggregate the rows by using mean.

6.7 Data Cleaning (Post Integration and Transformation)

In order to prevent noise in the data from affecting the experiment results, we perform data cleaning to ensure that the prediction models can effectively learn the data. This also includes dropping data columns we do not need in order to reduce data dimensionality.

6.8 Data Exporting

Once we are done with pre-processing, we can export all of our finalized dataframes into csv files for ease of access.

Problem 1: question1_initial.csv

Problem 2: question2.csv

Problem 3: train_q3 and test_q3.csv

Problem 4: question4.csv

7 Implementation and Results of Experiments

7.1 Problem 1: Predicting whether a driver finishes in PWP

7.1.1 Experiment Setup

7.1.1.1 Initial Dataset

For this problem, we used the combined-initial.csv which contains the required input features for the binary classification task.

These features contain information that is available to the Constructors before the start of the race such as Track Features and Starting Grid Position. Using this information available to Constructors pre-race, we will like to develop a classification model that is able to predict whether a Driver will end up in PWP based on the input features.

For this classification problem, we first have to create the labels, or targets for the classification model. The labels are binary classes where '0' represents that the Driver will not be in a PWP and '1' represents that the Driver will be in a PWP.

7.1.1.2 Dataset with Added Input Features

Adding new features could improve the model's accuracy. One of the features we thought of is to calculate various error rates based on the 'error_category' attribute. From this, we can derive 3 features, Driver error rate, Constructor error rate and Circuit error rate. These features are added to provide information on the percentage of races where Drivers, Constructors and Circuit incidents, which prevented the Drivers from completing the race and dropped out of the PWP.

7.1.1.3 Dataset with Reduced Number of Features

A large number of input features may not necessarily result in a better fitted classification model and having a smaller number of features would allow us to avoid the Curse of Dimensionality.

Therefore, for the last set of experiments, we used the sklearn function called SelectKBest() to select the k best input features based on chi-squared stats, where chi-squared stats between each non-negative feature and class are computed. This score can be used to select the n_features with the highest values for the test chi-squared statistic from X, which must contain only non-negative features such as booleans or frequencies, relative to the classes. Chi-square test measures dependence between stochastic variables, so using this function "weeds out" the features that are the most likely to be independent of class and therefore irrelevant for classification.

Using the above method, these are the selected input features used for the classifiers:

```
(4611, 5)
Selected features:
driverId
constructorId
grid
driver_error_rate
constructor_error_rate
```

7.1.2 Classifiers Used

For this Classification problem, we fitted the dataset on 4 different classification models. The 4 classification models used are:

- Support Vector Machine(SVM)
- Naives Bayes Classifier
- Decision Tree Classifier
- Random Forest Classifier

7.1.3 Comparison Schemes

In order to determine the best classification model to use, we split the data into train and test sets using the dataset obtained from 'combined-initial.csv'. This is done using the *sklearn* function called ***ShuffleSplit()***. The test size of each cross validation split is 20 percent of the dataset.

```
cv = ShuffleSplit(n_splits=100, test_size=0.2, random_state=0)
```

We fitted the 4 different classifiers using the cross validation dataset obtained from the ***ShuffleSplit()*** function. We fit the 4 different classifiers using the *sklearn* function called ***learning_curve()***. The function determines cross-validated train and test scores for different train set sizes. A cross-validation generator splits the whole dataset, k times in train and test data. Subsets of the train set with varying sizes will be used to train the estimator and a score for each train subset size and the test set will be computed. Afterwards, the scores will be averaged over all k runs for each training subset size.

```
train_sizes, train_scores, test_scores, fit_times, _ = \
    learning_curve(estimator, X, y, cv=cv, n_jobs=n_jobs,
                   train_sizes=train_sizes,
                   return_times=True,random_state = 0)
```

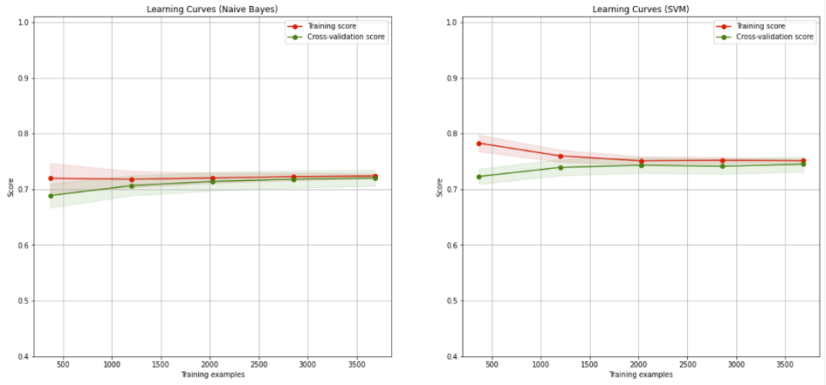
The best classification model is selected based on which model has the highest test accuracy, F1 score, as well as ROC AUC score obtained after fitting all the training examples to the 4 classification models.

- The test accuracy tells us how accurately the classification model predicts the target label correctly.
- F1 score conveys the balance between the precision and the recall of the model, where precision is the number of True Positives divided by the number of True Positives and False Positives and recall is the number of True Positives divided by the number of True Positives and the number of False Negatives.
- ROC AUC score computes the Area Under the Receiver Operating Characteristic Curve to tell how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting the true positives and true negatives in the data.

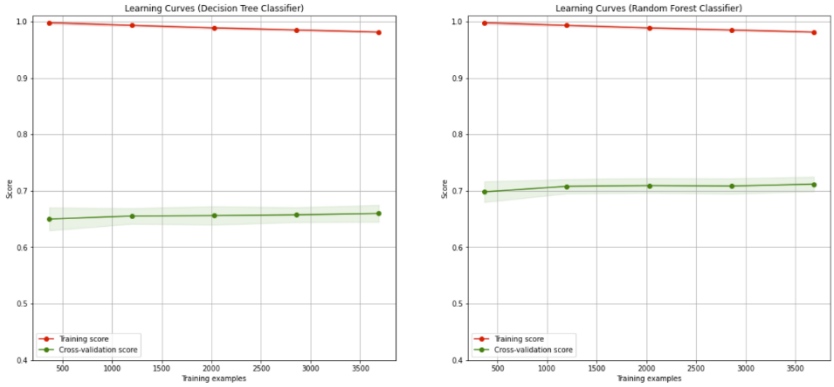
7.1.4 Result and Analysis

7.1.4.1 Initial Dataset

The train and test scores obtained after fitting the dataset to the 4 classifiers are plotted against the number of training examples. The results of the experiments for the initial dataset is shown below:




```
Naive Bayes Validation Accuracy: 71.70596205962062%
Support Vector Machine Validation Accuracy: 74.09214092140924%
```




```
Decision Tree Accuracy: 66.15040650406505%
Random Forest Accuracy: 71.54878048780492%
```

7.1.4.2 Test Accuracies Comparison of All Datasets

	Initial Dataset	Dataset with Added Features	Reduced Dataset
SVM	75.623%	76.056%	76.273%
Naives Bayes	74.000%	74.431%	74.864%
Decision Tree	68.797%	68.689%	73.564%
Random Forest	72.156%	72.589%	74.431%

* Tables for F1 Score, ROC AUC and Cross Validation can be found in Appendix: 10.1

7.1.5 Conclusion

From the results obtained, we can observe that the SVM classifier performed the best with the highest test accuracy, F1 Score and ROC AUC Score. From the experiments, we can observe that the added features helped improve the accuracy of 3 of the classifiers except for the Decision Tree Classifier. Even when we created the reduced features dataset based on chi-squared stats, 2 of the 3 added features were selected as input features for the classification models and those features are more strongly correlated to the target variables as compared to

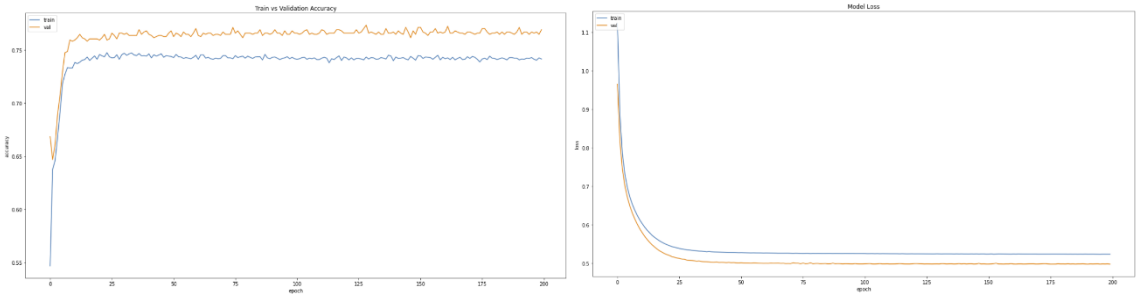
most of the input features in the initial dataset. This shows that the added features are relevant features to the classifiers in predicting whether a driver will finish in a PWP. From the results, the classifiers perform the best when using a reduced dataset, which implies that we should always eliminate redundant features in order to achieve better results for any prediction model.

7.1.6 Using Neural Network for Prediction

After obtaining the results with the various classifiers above, we would like to see if using a neural network would enable us to achieve a higher prediction accuracy.
For this, we utilized a 3-layered Neural Network to perform the classification. The neural network consists of:

- Input layer
- Dense Layer of 20 neurons
- Dense Layer with 'Softmax' activation as output layer

We then used the SGD optimizer for the neural network with a learning rate of 0.001 and a learning decay rate of 1e-6. The model is trained for 200 epochs on 80 percent of the original dataset and then validated on 20 percent of the dataset for each epoch.
The input dataset that we used for this classification neural network is the reduced features dataset which consists of 5 input features.



From the first graph, the neural network manages to learn the classification rather quickly, with the validation accuracies converging even before 25 epochs. The validation accuracies converge around 76% to 77%. We saved the model with the highest validation accuracy and made predictions on the test set. We compare the predictions against the actual class labels to give us the test accuracy of the model. Using the neural network classifier, we were able to achieve 77.36% test accuracy.

7.2 Problem 2: Predicting the Fastest Lap Time for a Race

7.2.1 Experiment Setup

7.2.1.1 Initial Dataset

For this problem, we will be using the dataframe imported from question2_new.csv. The dataframe includes features such as circuit information and history of past 4 years' fastest lap times of the circuit. This allows us to design a regression model to try to predict the fastest lap time for a particular circuit for a particular year.

7.2.1.2 Reduced Dataset

A large number of input features may not necessarily result in a better fitted regression model and having smaller input features would allow us to avoid the Curse of Dimensionality. Therefore, for the last set of experiments, we used the *sklearn* function called ***SelectKBest()*** to select the k best input features based on the f_regression function in *sklearn*. The function uses a Linear model for testing the individual effect of each of many regressors. This is a

scoring function to be used in a feature selection procedure, not a free standing feature selection procedure.
This is done in 2 steps:

1. The correlation between each regressor and the target is computed, that is, ((X[:, i] - mean(X[:, i])) * (y - mean_y)) / (std(X[:, i]) * std(y)).
2. It is converted to an F score then to a p-value.

Using the above method, these are the selected input features used for the model:

- turns, lap_length, race_laps, fastest_q, prev_year_1_lap, prev_year_2_lap, prev_year_3_lap, prev_year_4_lap

7.2.2 Regression Model

For this regression problem, we utilized a 4-layered Neural Network to perform the regression. The neural network consists of:

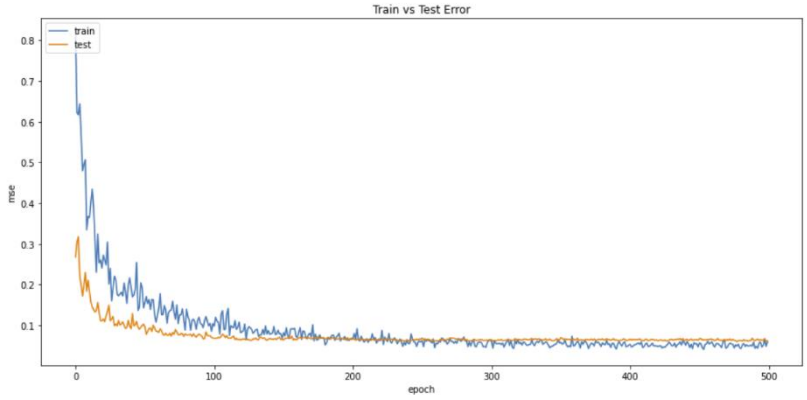
- Input layer
- Dense Layer of 10 neurons
- Dropout Layer of 0.2 Dropout Rate
- Dense Layer of 10 neurons
- Dropout Layer of 0.2 Dropout Rate
- Output Layer

We then used the Adam optimizer for the neural network with a learning rate of 0.001 and a learning decay rate of 1e-6. The model is trained for 500 epochs on 75 percent of the original dataset and then validated on 25 percent of the dataset for each epoch.

7.2.3 Results and Analysis

7.2.3.1 Initial Dataset

As this is a regression problem, we monitor the Mean Squared Error (MSE) of the regression model and the model is trained to give us the lowest M.S.E possible which reduces the error between the predicted fastest lap time value and the actual fastest lap time value. Below shows the plot of M.S.E against the number of training epochs for the model:



Towards the last 100 epochs, the M.S.E of the regression model converges to around 0.06 for each epoch.

```
Epoch 498/500
24/24 - 0s - loss: 0.0721 - mse: 0.0667 - val_loss: 0.0720 - val_mse: 0.0666
Epoch 499/500
24/24 - 0s - loss: 0.0534 - mse: 0.0481 - val_loss: 0.0667 - val_mse: 0.0613
Epoch 500/500
24/24 - 0s - loss: 0.0645 - mse: 0.0592 - val_loss: 0.0674 - val_mse: 0.0621
```

We also compared the predicted fastest lap times against the actual fastest lap times and observed the number of predicted fastest lap times that are within 0.2 threshold of the actual fastest lap times.

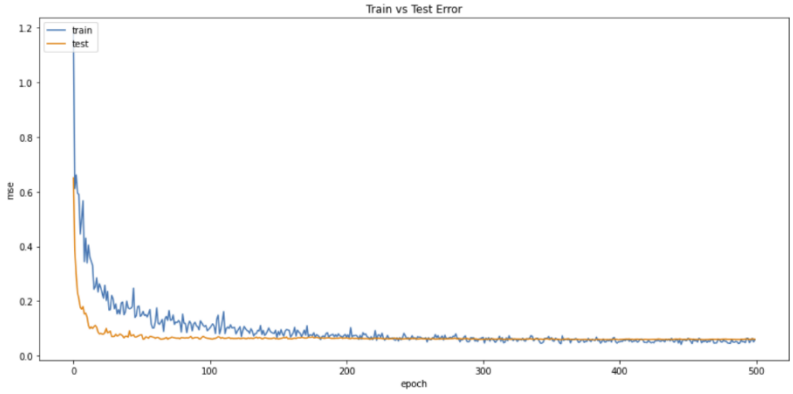
```
count = 0

for i in range(0,len(predictions)):
  if(abs(predictions[i]-y_test[i])<=0.2):
    count += 1

print("number of predictions within acceptable range: " + str(count))
print("percentage of predictions within acceptable range: " + str((count/32)*100) + "%")
```

```
number of predictions within acceptable range: 24
percentage of predictions within acceptable range: 75.0%
```

The regression model was able to give us 75 percent of predictions within acceptable range as can be seen from the figure above.

7.2.3.2 Reduced Dataset

Below shows the plot of M.S.E against the number of training epochs for the model for the reduced dataset:



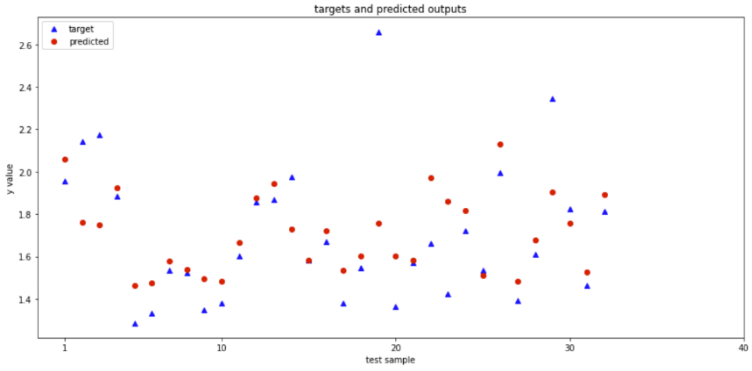
Similar to the model that used the full dataset, the M.S.E of the model converges to about 0.06 for the last 100 epochs. However, the M.S.E are in the lower 0.06 values which shows that the model does improve slightly with a reduced dataset.

```
Epoch 498/500
24/24 - 0s - loss: 0.0680 - mse: 0.0633 - val_loss: 0.0667 - val_mse: 0.0620
Epoch 499/500
24/24 - 0s - loss: 0.0556 - mse: 0.0510 - val_loss: 0.0661 - val_mse: 0.0614
Epoch 500/500
24/24 - 0s - loss: 0.0628 - mse: 0.0582 - val_loss: 0.0649 - val_mse: 0.0602
```

```
for i in range(0,len(predictions)):
  if(abs(predictions[i]-y_test[i])<=0.2):
    count += 1

print("number of predictions within acceptable range: " + str(count))
print("percentage of predictions within acceptable range: " + str((count/32)*100) + "%")
```

```
number of predictions within acceptable range: 24
percentage of predictions within acceptable range: 75.0%
```

The percentage of predictions within acceptable range did not increase even with the slightly better M.S.E values achieved by the model. Below shows the plot of the predicted fastest lap time values and the actual fastest lap time values for the test set obtained from the dataset that we used.



7.2.4 Conclusion

From the results, we can observe that the regression model is not able to obtain the best test accuracy in predicting the fastest lap times for each of the circuits for each year. This can be partially attributed to the fact that we do not have enough data to train and validate the regression model on.

Also, the input features that we used for the prediction of the fastest lap times may not be the best features for the predictor to learn a pattern for fastest lap time. This is due to the fact that in reality, many other factors affect the fastest lap time set by the drivers during the race, these include the type of tyres used on the car, amount of fuel left in the car when attempting the fastest lap amongst other things and we do not have access to this information. Therefore, we are unable to improve the loss or the accuracy of the regression model for the prediction of fastest lap times.

7.3 Problem 3: Predict Final Driver Standing for Driver-Constructor Pair

7.3.1 Experimental Setup

7.3.1.1 Initial Dataset

For this problem, we will be importing train_q3.csv and test_q3.csv as dataframes.

7.3.2 Implementation of Regression Model

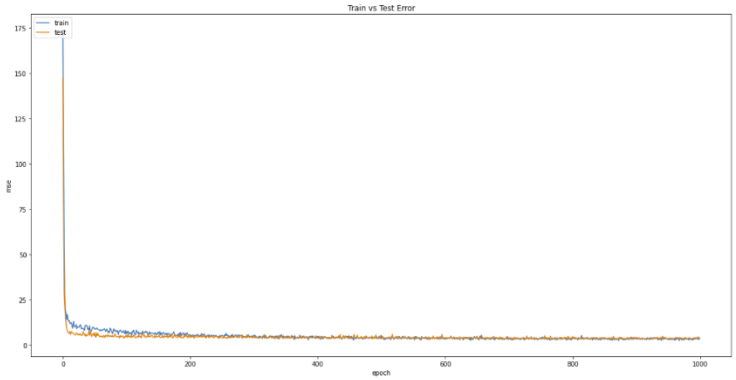
For this task, the objective is to train a prediction model that can predict the Final Drivers' Standing for a season for a Driver-Constructor pair. Thus, we decided to build a regression model where the output value will be a real number which will be rounded off to a whole number to give us the final predicted Driver Standing for a given driver driving for a particular constructor.
For this regression problem, we utilized a 4-layered Neural Network to perform the regression. The neural network consists of:

- Input layer
- Dense Layer of 32 neurons
- Dropout Layer of 0.2 Dropout Rate
- Dense Layer of 32 neurons
- Dropout Layer of 0.2 Dropout Rate
- Output Layer

We then used the Adam optimizer for the neural network with a learning rate of 0.001 and a learning decay rate of 1e-6. The regression model is then trained for 1000 epochs.

7.3.3 Result and Analysis

As this is a regression problem, we monitor the Mean Squared Error (MSE) of the regression model and the model is trained to give us the lowest M.S.E possible which reduces the error between the predicted Driver's Standing and the actual Driver's Standing. Below shows the plot of M.S.E against the number of training epochs for the model:

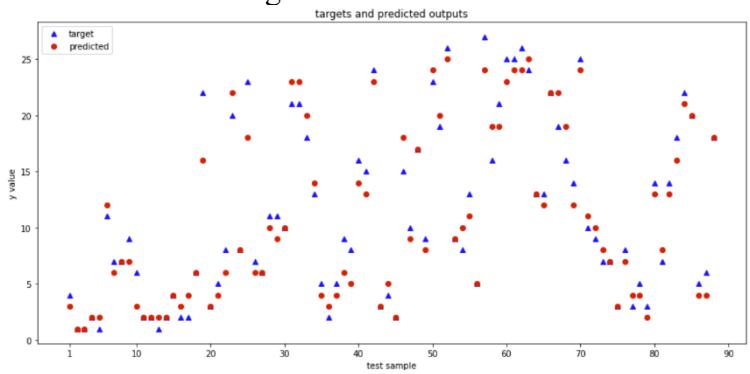


We then rounded the predicted Driver's Standings to whole numbers and compared them against the actual Driver's Standings. We then calculated the percentages of predictions that are within 1 position difference between the predicted Driver's Standings and actual Driver's Standings. The results are shown below:

```
for i in range(0,len(predictions)):
  if(abs(predictions[i]-y_test[i])<=1):
    count += 1

print("number of predictions within acceptable range: " + str(count))
print("percentage of predictions within acceptable range: " + str((count/88)*100) + "%")
```

```
number of predictions within acceptable range: 60
percentage of predictions within acceptable range: 68.18181818181817%
```

From the results, we can see that 68 percent of the predictions are within 1 position difference between the predicted Driver's Standings and actual Driver's Standings. The graph below shows the plot for the predicted Driver's Standings and the actual Driver's Standing. We can see that there are quite a handful of predictions where the predictions are the same as the actual Driver's Standings as shown by the red dots covering the blue triangles. Also, 88.63% of the predictions are within a 2 positions difference between the predicted Driver's Standings and actual Driver's Standings



7.3.4 Conclusion

From the results, we can observe that the regression model is not able to provide the best possible accuracy in predicting the final Driver Standing for a Driver-Constructor pair. This can be partially attributed to the fact that we do not have enough data to train and validate the regression model on.

In addition, the input features that we used for the prediction of the Driver Standing may not be the best features for the predictor to learn a pattern for Driver Standing for a Driver-Constructor pair. This is due to the fact that in reality, many other factors affect the final Driver's standing as other teams may develop better cars in the season, the car developed by the given Constructor is not as good as the previous seasons, and many other contributing

factors. Therefore, we are unable to improve the loss or the accuracy of the regression model for the prediction of Driver Standing for a Driver-Constructor pair.

7.4 Problem 4: Multiclass Classification of F1 Races

For this problem, we wanted to see if other information regarding the races can help improve race classification accuracy. Therefore, we decided to include another dataset from Kaggle regarding F1. After obtaining the results from Problem 1, we wanted to understand the reason why we were only able to achieve a maximum of 77.36% accuracy for the prediction of whether a driver will be in a PWP. With that in mind, we decided to extend Problem 1 into a multilabel classification. The new labels for classification are:

- 0: DNF (Did Not Finish)
- 1: Podium (1st to 3rd Positions)
- 2: Point Winning Positions (4th to 10th)
- 3: Finished (11th and above race finishes)

7.4.1 Initial Dataset

- dataset.csv, races_cleaned.csv, drivers_cleaned.csv, results_clean.csv, combined - final.csv,
- For the new dataset from Kaggle, we have the following new race information:
    - statusId : 1 is Finished, 0 is Did Not Finish
    - Did not finish, Podium, Pos 4 to 10: probability of a driver in each of the race classification based on past 3 years' race results for that circuit
    - Sc: Probability of a Safety Car
    - Wet: 1 is wet weather forecast, 0 is dry weather forecast
    - pitStop timing (avg): average pitstop timing of driver
    - pitStop timing prop(driver): proportion of pitstop time as compared to race time

7.4.2 Implementation of Classification Model

For this classification problem, we utilized a 4-layered Neural Network to perform the classification. The neural network consists of:

- Input layer
- Dense Layer of 32 neurons
- Dropout Layer of 0.2 Dropout Rate
- Dense Layer of 32 neurons
- Dropout Layer of 0.2 Dropout Rate
- Dense Output Layer of 4 neurons with "softmax" activation

We then used the SGD optimizer for the neural network with a learning rate of 0.001 and a learning decay rate of 1e-6. The classification model is then trained for 1000 epochs.

7.4.3 Result and Analysis

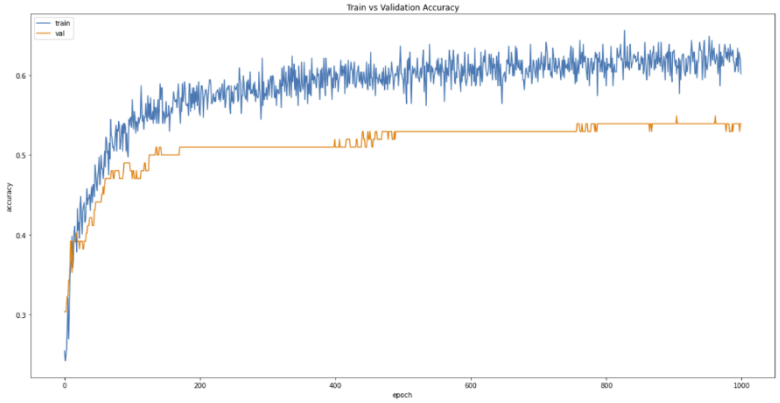


Fig 6.4.1.1: Train and validation Accuracy vs. Epochs

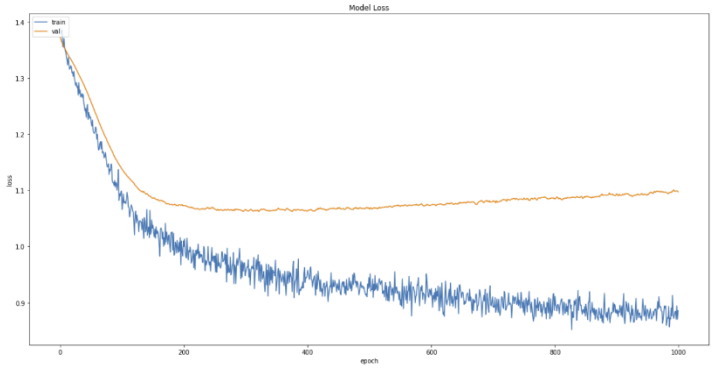


Fig 6.4.1.2: Train and validation Loss vs. Epochs

From the results above, we can see that the test accuracy obtained by the model is not very high, with the test accuracy converging at about 53.9% accuracy.

7.4.4 Further Analysis of Results Obtained

After observing that we are getting unsatisfactory results for our classification model, we went on to do further analysis on the results obtained.

First, we create a list of classes predicted by the model by using the **keras** model.predict() function using the input data from the test set. We then also get the target values, the driver and constructor we are predicting the classes for and put them together into a dataframe.

	race_class	predicted	drivers	constructors
329	3	2	4.0	1.0
371	0	2	18.0	1.0
219	2	1	817.0	9.0
403	3	3	828.0	15.0
78	1	1	20.0	6.0
15	1	1	1.0	131.0
487	3	3	154.0	210.0
340	2	2	4.0	1.0
310	3	2	815.0	10.0
102	0	2	8.0	6.0
418	3	3	828.0	15.0
411	3	3	828.0	15.0
446	3	3	836.0	15.0
386	0	3	828.0	15.0

Fig 6.4.4.1: Head of Dataframe

Next, for each of the classes, we loop through the dataframe to find out the percentage of correct predictions for each of the classes by comparing the predicted classes with the actual target classes. These are the results for each of the classes:

```
percentage correct DNF: 0.0%
percentage correct podiums: 72.72727272727273%
percentage correct 4th to 10th: 73.33333333333333%
percentage correct finished more than 10th: 60.0%
Number of DNFs: 20
percentage of correct predictions: 54.90196078431373%
percentage of correct predictions if exclude DNFs: 68.29268292682927%
Percentage of DNFs: 19.607843137254903%
```

From the results, we can see that for the DNF class, we are getting 0% correct predictions, while for the other 3 classes, we are getting decent correct prediction accuracies with the accuracy of the 3 classes higher than that of the overall test accuracy of the model.

Not only that, we found out that DNFs took up 19% of the test set and if we do not consider the DNF rows in the test set, we are able to achieve an overall test accuracy of up to 68.29% for the other 3 classes. This explains the poor performance of the model in predicting the race classes. If we were to take out predicting DNFs, and only predict 3 classes, and consider data that do not have DNF, these are the results obtained:

```
percentage correct podiums: 80.76923076923077%
percentage correct 4th to 10th: 87.09677419354838%
percentage correct finished more than 10th: 73.07692307692307%
percentage of correct predictions: 80.72289156626506%
```

8 Conclusions

8.1 Summary of Project Achievements

From this project, we have gained valuable insight into the use of Data Analytics and Mining Concepts for the purpose of improving race strategies in F1.

For Problem 1, after experimenting on various classification models, we were able to achieve a respectable prediction accuracy of 77.36% using an Artificial Neural Network for predicting whether a driver will finish in a PWP. Therefore, a Constructor can make use of this prediction model to determine the race strategy to be implemented for each of its Driver for a particular race.

For Problem 2, by using an Artificial Neural Network for regression, we performed the prediction of the fastest lap time for a given race. Using the regression model, we are able to achieve a prediction accuracy of 75.0% for the prediction of fastest lap time for a race, given a 0.2 threshold. This model can also aid the Constructors in estimating the time to hit if they want to obtain the fastest lap time for a particular race.

For Problem 3, we performed regression to predict the Final Drivers' Standings for Drivers if they are to race with a particular Constructor. For this problem, we are able to achieve a decent prediction accuracy of 68.18%. Therefore, using the predicted values a Constructor is better able to decide which Driver to sign for in the next season, to give them the biggest advantage.

For Problem 4, we performed a multi-label classification on the race outcome for a particular driver, for a particular race. We utilized an artificial neural network to attempt to predict the 4 classes of race outcomes: 0 : DNF (Did Not Finish); 1 : Podium (1st to 3rd Positions); 2: Point Winning Positions (4th to 10th); 3: Finished (11th and above race finishes). However, we were only able to achieve a test accuracy of 54.9% for the classification task. After further analysis, we realized that the model is not able to predict DNF (Did Not Finish) outcomes as these outcomes are due to collisions and accidents which are random by nature. Therefore, by removing the DNF data rows and only predicting the 1 : Podium (1st to 3rd Positions); 2: Point Winning Positions (4th to 10th); 3: Finished (11th and above race finishes) race outcomes, we were able to achieve a high accuracy of 80.72%. This prediction model can help Constructors better predict race outcomes of their Drivers and thus, help them determine the race strategy to use for the race.

8.2 Directions for Improvement

From the various data mining tasks performed, we realized that there are a lot of factors and information that we did not consider when performing the predictions. For example, for Problem 1, 2 and 3, they are prediction problems regarding races, and we did not consider other factors such as the tyres the drivers are using, track temperature, wind speed of the track, etc. These factors can affect the performance of the cars and drivers during races which will be helpful towards race related predictions. For Problem 4, for the prediction of the Final Driver's Standings, we did not take into account that new Drivers may join F1 in the next season and their results can affect the next season Driver's Standings.

There is also a lack of data that is available publicly regarding the races, which would have been useful to us in performing the data mining tasks. This includes data like the telemetry

data of the various F1 cars, which will tell us how various cars built by the constructors perform around different track conditions, which can aid in our analysis of F1.

To achieve even higher prediction accuracies, an extension of the project can be to tune the parameters of the various prediction models using techniques such as GridSearch. This ensure that we are getting the best possible results for each of the prediction models.

9 References

Kaggle Dataset 1: https://www.kaggle.com/cjgdev/formula-1-race-data-19502017

Kaggle Dataset 2: https://www.kaggle.com/coolcat/f1-binary-classification-of-race-finish-status/notebook?select=Pirelli_Tyre_Categories.csv

10 Appendix

10.1 Problem 1

10.1.1 Cross Validation Accuracies Comparison of All Datasets

	Initial Dataset	Dataset with Added Features	Reduced Dataset
SVM	74.092%	74.112%	74.741%
Naives Bayes	71.706%	70.730%	71.236%
Decision Tree	66.150%	66.644%	71.244%
Random Forest	71.549%	71.806%	72.533%

10.1.2 F1 Score Comparison of All Datasets

	Initial Dataset	Dataset with Added Features	Reduced Dataset
SVM	0.75936	0.76514	0.76776
Naives Bayes	0.74948	0.76447	0.76892
Decision Tree	0.68831	0.68207	0.73362
Random Forest	0.72747	0.73171	0.75210

10.1.3 ROC AUC Score Comparison of All Datasets

	Initial Dataset	Dataset with Added Features	Reduced Dataset
SVM	0.75617	0.76045	0.76259
Naives Bayes	0.73968	0.74355	0.74786
Decision Tree	0.68803	0.68710	0.73579
Random Forest	0.72142	0.72575	0.75611