introduction with Value Based Problem Statement

For this assignment, I did it from the perspective of a data scientist in a sports beverage company (like Monster or Red Bull). Due to the popularity and extremeness of Formula 1, the company is currently looking to sponsor an F1 team and expand its sports roster. From this sponsorship, The company intends to increase its brand awareness and potentially grow its market share.

The company’s management wants to choose a top-performing Formula 1 team as the team’s success will bring better brand recognition and association for the company.

Hence, the company has assigned the data scientists to come up with a model to predict the top teams in future races so that the company is able to decide based on the model, the team with great potential in future races to invest in.

roblem Formulation

• Load and Explore the Data

• Understand the Data

Since I was coding a model that was predicting teams, I decided to use constructor-related tables.

When going through the tables and data dictionary and researching more about F1, I found some information that may affect the data-wrangling process. For example, the time stated in the race circuit was not in the local circuit location’s time but instead was in UTC, which is the local time in England. This means that I could not actually determine what time the race started at the circuit location. For example, the time states that the race happened at 12 pm for the Singapore Grand Prix, but the race actually happened at 8 pm in Singapore time.

Another one was that constructors can only employ 2 drivers currently, but in the early 1950s and 60s, there were no limits on the number of drivers. This could make it difficult to determine the number of drivers the team has for each race and find information about them.

After looking through data dictionary and the different tables, I was interested in the following tables and columns:

‘circuits\_mod’: circuitRef, Location, country, alt

* These features help provide further details about the circuit. They display the different circuit locations and altitudes which may play a part in determining how a team performs.

‘constructors\_mod’: constructorRef, nationality

* These features provide details of the constructors. The nationality of the constructor might matter as it is possible that some constructors perform better at home compared to foreign circuits.

‘races\_mod’: date

* The date of the race might affect the environmental factors. For example, months later in the year typically see colder weather and possibly snow. This affects the teams and drivers’ situation and performance.

‘constructors\_standing\_mod’: position, wins

* Position can be our target variable and used to create the target column. Wins help us how successful a team is overall, which might help us determine the top teams in future races.

I also wanted to use qualifying, driver and status tables however drivers were not grouped together by the constructor in each race which made it hard to extract the status and timings of the drivers in each team.

• Formulate a Prediction Problem

After going through the data, I decided to predict the top 6 constructors based on position. For this problem, I used classification where we set the top 6 to be indicated by 1 and the rest to be indicated by 0. I eventually chose my machine learning model to be binary logistic regression.

Data Wrangling on multiple tables

• Concatenate, Merge or Join the tables

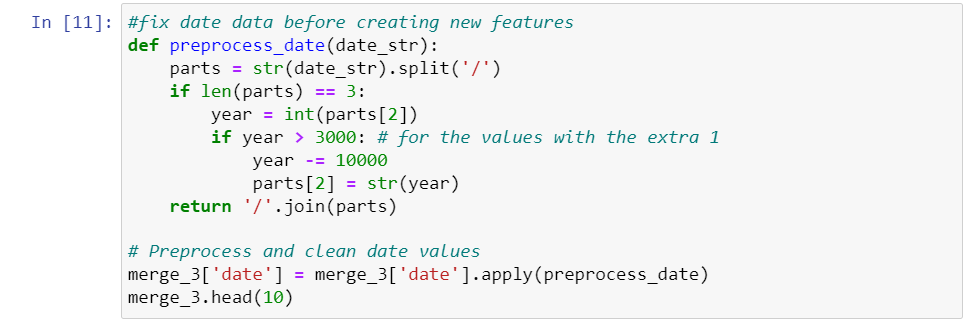
So, the tables I used were ‘circuits\_mod’, ‘constructors\_data’, ‘races\_data’ and ‘constructors\_standings\_mod’.

I merged all these tables by using an inner merge. Afterwards, I dropped columns that I decided cannot be used or were repeats of other columns. The unusable columns were URL and time. URL was unusable as it was just a Wikipedia link and time could not be used as it was in UTC as explained above.

Next, I dropped repeated columns such as positionText, lat, lng and name as all these columns could already be represented by other columns in the dataset.

• Extract and Create features from different tables

Firstly, I had to clean the data in the date column before I could extract any features from it. The date column had some values where there was an additional 1 in front of the year. For example, ’4/5/12009’ has an extra 1 in front of 2009. Because of this, I had to preprocess the data first. I created a function which splits the date into three parts, before taking the third part (the year) and checking if it was greater than 3000 (which would show it had an extra 1), I then deducted 10000 from the year to remove the 1 and returned the whole date.



Since the date column was now clean, I converted the column to Datetime and extracted the year, month, day of the month and day of the week.

Next, I aggregated the average points for each race. I grouped the races by raceID and found the average points. I then converted it into a data frame and merged it with my current one, forming a new mean\_points\_of\_race column. Using the new column, I compared the points of each row to the corresponding average of the race. If the points were greater than the mean, a 1 will be indicated else it would be a 0. If a team performs better than the average points in the race, it may indicate that the team is doing better than at least half of the competition, which can help predict if they will be the top team.

Next, I summed the number of wins for each constructor. I grouped the constructors by constructorID and summed the number of wins through the ‘wins’ column. I converted it into a dataframe and merged it with my current one, forming a new sum\_of\_wins column. By summing the wins of each constructor, we can see how successful a team is overall. A successful team may have a higher probability of being in the top 6 teams for future races which helps the model predict.

Next, I created the target variable column. If the teams’ position was <= 6th, a 1 would be indicated else 0. The dataset was quite balanced, with the ratio of 0s to 1s being 6755 : 5961

A graph with blue bars

Description automatically generated

Lastly, I dropped the position and ID columns. Position was used to create my target column hence it should not be added to the model. ID columns such as circuitId and raceId were dropped because they could either be replaced by the unique identifier Ref columns or were not relevant to the model as they are unique identifiers. This helps reduce noise and unnecessary complexity in the dataset.

Data Cleansing and Transformation

• Missing Value and Outliers

For missing values, a very small portion of data was missing from the alt and date variables columns. For the date variables, I decided to drop the rows. There was less than 0.85% of the data missing hence I just dropped it. Dropping was also the only way I could deal with the missing date values. Datetime is typically not used in the same way as numerical values hence mean and median imputation was not recommended. Mode date could also not be used as each race has a unique date. Forward and backward interpolation also does not work as the dates were not in chronological order.

For the alt column, it had ‘\N’ values which meant unclassified, which is technically null. So I converted all the ‘\N’ values to np.nan first before cleaning. For alt I tried median and random sampling, both methods seem have the same model performance. I just chose to do median imputation as it was more consistent.

A screenshot of a computer

Description automatically generated

(Before outliers)

I did outlier cleaning on variables: points, alt, sum\_of\_wins

For example in the diagnostic plot on points, we can see that points had a lot of outliers and an extremely positively skewed distribution. We can see from the QQ plot that the distribution is very far from a normal distribution. Outliers can affect the performance of the machine learning models hence it is important to clean them.

For outliers, I tried trimming, windsorization and capping. Trimming was not suitable due to the number of rows it removed. Trimming would have removed 1425 rows out of 12609 which was losing around 10% of the total rows. Due to the amount of data that would be lost, I decided not to use trimming.

For windsorization and capping, since the distribution is skewed for all the features, both methods used the IQR capping methods. When comparing model performance, capping sees a slightly better LogReg model performance than windsorization hence I decided to use capping.

A screenshot of a computer

Description automatically generated

(After outlier)

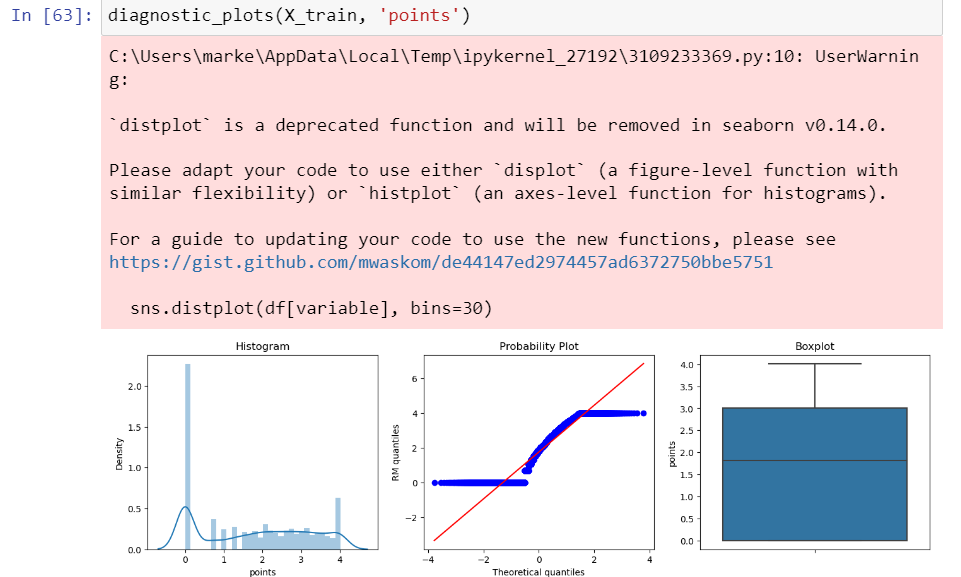
• Categorical Data

Firstly, I created rare categories for categorical features with more than 40 unique values. For the features I grouped, I set the threshold percentage to 1%. Next, I tried encoding methods to transform the strings into numbers before entering the model. I tried Ordered Ordinal and Target Mean encoding. Target mean encoding performed slightly better than Ordered Ordinal Encoding in terms of the LogReg Model accuracy score.

• Numerical Data

First, I tried to transform the data using Power and Yeo-Johnson transformers. Logarithm, Reciprocal and Box-Cox transformers could not be used as there were negative and zero values in ‘alt’. When trying the Power transformer, the transformer kept returning null values to me due to the negative values in alt. I tried different exponential values, but all returned null values. Hence, I could only use Yeo-Johnson.

Next, I did variable binning and discretization on points, alt and sum\_of\_wins. The features had skewed distributions as seen in Figure () hence I performed binning and discretization. I tried Equal Frequency, Discretization plus Encoding and Equal Width Frequency. Equal Width Frequency was by far the best out of the 3 in terms of LogReg model score.



(After transformation for points)

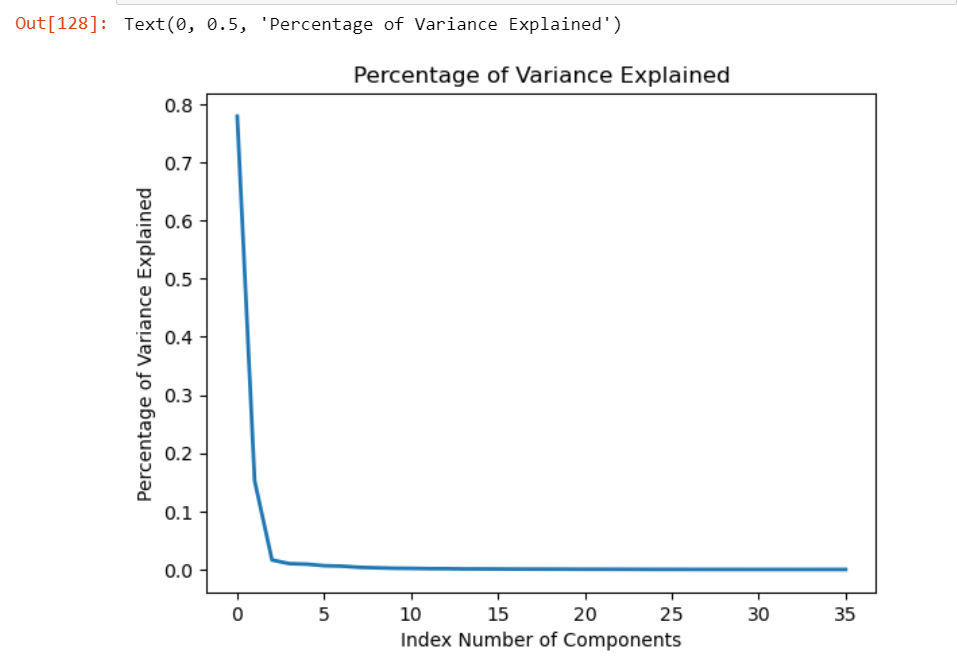
QQ plot already showed a normal distribution already, transformation ensured that the distribution was normal and removed outliers.

• Others

I also did feature scaling and engineering. Firstly, I scaled the data with different methods. I used Standardization, Min-Max Normalisation and Robust Scaling. Robust Scaling performed the best out of the three in terms of LogReg performance.

Next, I did Polynomial Expansion on points, alt and sum\_of\_wins. Polynomial Expansion seemed to increase the predictive power of the algorithm as the LogReg model performance improved slightly.

Lastly, I did PCA. I scaled once more before doing PCA. The analysis showed that the first component made up 77.9% of the variance, with the first 5 components making up 95% of the variance. The remaining components each made up less than 0.8% of the variance. I found that reducing components seemed to negatively impact LogReg model performance, hence I chose not to reduce the components.



6. Machine Learning Model

• Show Count of Rows and Columns

The final dataset had 8826 rows and 36 columns

• Build and Evaluate the model against a Naïve Baseline Model

As my target variable was between 1 and 0, I used a binary logistic regression model.

What is Naïve Baseline Model?

Naïve models are usually fitted on a training dataset and evaluated on a test dataset, the performance of the model is usually reported as a percentage of the number of correct predictions compared to the total number of predictions made, also known as accuracy. The models are naïve because they do not use knowledge about the domain or any learning to make predictions hence, they lack predictive power. However, due to their simplicity, they are often used as a benchmark for trained models. If the model does not surpass the baseline model’s performance on the data, the entire process must be reevaluated.

Comparing LogReg and Naïve Baseline

The accuracy score for the Naïve Baseline was 52.83% for train data and 54.06% for test data. The logistic regression model saw much better results, with accuracy scores being 91.30% on training data and 91.25% on test data. Since the Logistic Regression model performed better than the naïve baseline model, the Logistic Regression model demonstrated that it has skill on the prediction problem. Hence, the model can be used to predict the top 6 teams in future races.

Summary and Further Improvements

My prediction problem: Predicting the Top 6 Constructors in terms of position for future races.

After data exploration, I decided to use the circuits\_mod, constructors\_mod, races\_mod and constructor\_standings\_mod tables. I used inner merge to combine the tables and dropped columns that were either unusable or could be represented by other columns. I also created additional features from the cleaned Date values, as well as a higher\_than\_mean indicator and a sum\_of\_wins column.

I then created a target variable indicating position <= 6 to be 1 while the rest were 0. The dataset had a relatively good balance of 0s to 1s. Lastly, I dropped ID columns that could either be replaced by Ref columns or were not relevant to the model.

For data cleansing and transformation, I dropped rows for missing Date data and did median imputation for missing alt values. For outlier cleaning, I used Capping as it performed slightly better than Windsorization. For numerical variables, I used Yeo-Johnson as the power transformer and Equal Width Frequency for binning and discretization as it performed the best out of the 3 I tested.

For categorical data, I grouped the rare categories for features with more than 40 values and used Target Mean Encoding as it performed better than Ordered Ordinal Encoding. For feature scaling, Robust Scaling performed the best of the 3 I tried. Polynomial Expansion improved my model performance slightly while reducing principal components in PCA seemed to negatively impact my model score.

When evaluating the model, the Logistic Regression scores were significantly better than the Naïve Baseline model scores, showing that the LogReg model could be used to effectively predict the Top 6 Constructors.

For further improvements, I could try to use more include more datasets. Datasets that include weather data, car model data and team staff data for example will provide me with much more information to increase the predictive ability of the model and improve model accuracy scores. I could also try and compare with other Machine Learning models such as Probit Regression and Decision Trees to evaluate my data wrangling process.