*Formula 1, also referred to as Formula One and F1, cars are the fastest road-course racing cars in the world – often exceeding 200 mph seconds after breaking to below 60 mph. There are ten teams, each with two drivers, competing for two concurrent awards – best driver and best constructor.*

*The driver with the most points at the end of a season is awarded the Drivers' Championship Award.  Teams who have accumulated the most points from their drivers receive the Constructors’ Championship Award. The point breakdown is below:*

| *Position* | *Points Scored* |
| --- | --- |
| *1* | *25* |
| *2* | *18* |
| *3* | *15* |
| *4* | *12* |
| *5* | *10* |
| *6* | *8* |
| *7* | *6* |
| *8* | *4* |
| *9* | *2* |
| *10* | *1* |
| *11-20* | *0* |

*In 2021, Red Bull drivers finished 1st and 4th overall, with the two drivers scoring a combined 585 points. That same season, Mercedes drivers finished 2nd and 3rd overall with the two drivers scoring a combined 613 points. Red Bull had the top driver while Mercedes was deemed the top constructor.*

*Success in Formula1 comes down to the build of the car. Bell et al. (2016) used a multi-level (random-coefficients) linear model to determine driver vs constructor effects. They found 86% of the variance in points scored came from the constructor and only 14% of the points came due to driver skill.*

*Kesteren and Bergkamp (2022) used a Bayesian multilevel Beta regression method to disentangle driver skill and constructor advantage. Their findings also found an 86% variance coming from constructors with 14% of variance due to driver skill.*

*The disparity between teams can be attributed to budget. Until 2021, there was no salary, or cost, cap. Asher (2022) breaks down what the cost cap does and does not cover. Notably, driver salaries as well as the wages of the three highest staff members and travel costs are not included in the salary cap. Engines are covered by a separate cap entirely.*

*Another difference that surely plays a role, but has not been studied extensively, is the race course and season length. In the 5 seasons we are studying, there have been 3 seasons of different lengths (2023 adds a 4th different length with 23 total races). In addition to the different lengths of seasons, there are different and duplicate courses raced. Italy gets two visits/year on average. On the other end of the spectrum, Qatar has only been visited once in the same five-year period. Continuing with the courses, there are different weather conditions, different times of the day, different rest times between race weekends, and perhaps most importantly, different road conditions. Some courses compete on a race track (similar to NASCAR) whereas others compete on normal street surfaces such as in Monaco.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Course*** | ***2018*** | ***2019*** | ***2020*** | ***2021*** | ***2022*** | ***Sub-Total*** |
| *Australia* | *1* | *1* | *0* | *0* | *1* | ***3*** |
| *Bahrain* | *1* | *1* | *2* | *1* | *1* | ***6*** |
| *China* | *1* | *1* | *0* | *0* | *0* | ***2*** |
| *Azerbaijan* | *1* | *1* | *0* | *1* | *1* | ***4*** |
| *Spain* | *1* | *1* | *1* | *1* | *1* | ***5*** |
| *Monaco* | *1* | *1* | *0* | *1* | *1* | ***4*** |
| *Canada* | *1* | *1* | *0* | *0* | *1* | ***3*** |
| *France* | *1* | *1* | *0* | *1* | *1* | ***4*** |
| *Austria* | *1* | *1* | *2* | *2* | *1* | ***7*** |
| *Great Britain* | *1* | *1* | *2* | *1* | *1* | ***6*** |
| *Germany* | *1* | *1* | *1* | *0* | *0* | ***3*** |
| *Hungary* | *1* | *1* | *1* | *1* | *1* | ***5*** |
| *Belgium* | *1* | *1* | *1* | *1* | *1* | ***5*** |
| *Italy* | *1* | *1* | *3* | *2* | *2* | ***9*** |
| *Singapore* | *1* | *1* | *0* | *0* | *1* | ***3*** |
| *Russia* | *1* | *1* | *1* | *1* | *0* | ***4*** |
| *Japan* | *1* | *1* | *0* | *0* | *1* | ***3*** |
| *United States* | *1* | *1* | *0* | *1* | *2* | ***5*** |
| *Mexico* | *1* | *1* | *0* | *1* | *1* | ***4*** |
| *Brazil* | *1* | *1* | *0* | *1* | *1* | ***4*** |
| *United Arab Emirates* | *1* | *0* | *0* | *0* | *0* | ***1*** |
| *Abu Dhabi* | *0* | *1* | *1* | *1* | *1* | ***4*** |
| *Portugal* | *0* | *0* | *1* | *1* | *0* | ***2*** |
| *Turkey* | *0* | *0* | *1* | *1* | *0* | ***2*** |
| *Netherlands* | *0* | *0* | *0* | *1* | *1* | ***2*** |
| *Qatar* | *0* | *0* | *0* | *1* | *0* | ***1*** |
| *Saudi Arabia* | *0* | *0* | *0* | *1* | *1* | ***2*** |
| ***Sub-Total*** | ***21*** | ***21*** | ***17*** | ***22*** | ***22*** | ***103*** |

*Over the course of the race weekend, 2 to 3 terabytes of data are generated per the Mercedes-AMG Petronas Formula One Team (2022). This is accumulated through three practice sessions, a qualifying race (which determines starting order), and the final race.*

*We do not have access to the full data. Through the Ergast Developer API and the Formula 1 Official Data Stream we can obtain much information. Luckily, there is a python package called FastF1 for accessing and analyzing the results, schedules, timing data, and telemetry. It was created by a German engineer (*[*https://github.com/theOehrly/Fast-F1*](https://github.com/theOehrly/Fast-F1)*). The background work was used extensively by copying and re-purposing examples using the documentation (*[*https://theoehrly.github.io/Fast-F1/*](https://theoehrly.github.io/Fast-F1/)*).*

*A shortcoming with the data is simply the lack of availability. Much of the timing data only dates back to 2018 as the information was not publicly available for prior years, or was not technically capable of producing and retaining. As such, we focused on the years beginning in 2018 and ending in 2022. This five year period provided 103 races from at least 27 different courses. The uniqueness of courses can be hidden by country name as some countries can provide multiple courses whereas others provide one course that is raced repeatedly.*

*Initially the FastF1 package was accessed via JupyterLab. After utilizing Jupyter Lab via Anaconda for the initial setup, I transferred the workstation to Spyder IDE via Anaconda. Python is where the majority of the programming and analytical work is done. Visualization was done in Python and R. With R, we used Shiny for exploratory data analysis.*

As can be seeon on the following two graphs, lap number when a pit stop is made is skewed left, as is the tyre life for when a pit stop is made. (<https://aje3887.shinyapps.io/thesis_test/>)

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

*Unfortunately there was no cumulative dataframe for each year. Instead, each race has it’s own group of dataframes (one for results, one for the lap-by-lap information, one for weather, etc.). This is due to how the data gets collected into the FastF1 package.*

*Our first step was to put each race into a singular dataframe for the year. The first thing we had to do was to merge the “laps” dataframe with the “results” dataframe for each race. The “laps” dataframe had 27 columns and between 60 and 1533 rows.*

*The “results” dataframe has 17 columns and 20 rows. In the results dataframe, we had to rename “Abbreviation” into “Driver” to complete the merge as there were no commonly-named columns. The results dataframe had multiple duplicate variables as the laps dataframe, so we only grabbed 'Driver', 'GridPosition', 'Position', 'Points'. This helped keep the merged dataframe to a manageable 31 columns. The issue with Results was a programming error that wouldn’t let us view the dataframe in the window pane like normal. We had to write the Results output to CSV and view it in Excel to understand what we were looking at.*

*After merging the laps and results dataframe, we created a new column called ‘TotalTime’ that calculated the total time a driver had driven. This let us figure out how long it took each driver to reach a certain lap in a race and therefore we were able to sort the race by their total time. The issue we ran across was drivers who did not complete the race, or who were lapped, became intermingled in the final results. The only way to account for those drivers would be to remove them from the dataframe – which would reduce the total number of rows from 110,124 to 3,415, a 97% loss of data.*

*With our goal of predicting pit stops, we determined that knowing the positioning of each driver at the end of each lap was important. We used the Saudi Arabia race in 2021 as our test data set. We transformed all the timing data into date-time type in seconds and performed column-wise addition to find the total lap time.*

*The second step was to calculate the pit stop time. The pit stop location on all race courses is at the start/finish line. This means that whenever a pit stop happens, the “pit stop in” time happens on lap “i” and the “pit stop out” time happens on lap “i + 1”.  In order to determine the true pit stop time, we used the ‘shift’ function in python to perform the diagonal subtraction of “pit in time” from “pit out time” with a lower bound of 0.*

*There were quite a number of issues that populated with pit stops. To begin with, everyone’s first lap included a pit stop exit time. This obviously is not true to the race as the drivers do not begin in the pit stop lane. This was overcome with the initial setup of the pit stop calculation by forcing the pit stop time to be 0.*

*The second issue was every single driver began the race with a time of pit stop out. The first record showed a pit out time of 24 minutes 6 seconds and 51 miliseconds.*

*This helped lead to the 3rd issue. Whenever there was a pit stop out time, there was no sector 1 time.  The race is broken into 3 sectors so we can see how quickly each driver makes it through each 1/third of the race. When a pit stop in time is recorded, there is no sector 1 time.*

*Pit stops also had a slight funkiness to it in that a pit stop in occurred on lap X, but the pit stop out occurred on lap X + 1. This is accurate because of the location of the pit stop on the track is right beside the start/finish line on the course.*

*Sector time of a lap per driver is based upon their actual timing in that sector. For example, sector 2 time in the Brazil 2021 race in the first observation was 29 seconds and 642 miliseconds. However, there is a sector time per session variable that showed the sector 2 time of 1 hour 2 minutes 43 seconds and 773 milliseconds.  That corresponds to the session time. The measurement is taken at the end of the sector.*

*We then created a cumulative time per lap by driver. We want to know how much time has elapsed for each driver at the end of each lap to determine their positioning. Pandas has a built in cumulative summing function that was used in conjunction with the groupby method.*

*We sorted that output by time. We are in the position to know the results of the race. Based off the cumulative time, the ending order should result with the point structure: 26, 18, 15, 12, 10, 8, 6, 4, 2, 1, 0, 0. However, our output was well off: 12, 26, 15, 18, 10, 2, 1, 4, 0, 6, 8, 0 (only 3rd and 7th are accurate)*

*The issue was mainly with the pit stop times. Due to red flags and potentially other unknown features, the length of pit stop times became inaccurate as the race was re-started and drivers in first place spent longer in the pit stop than others, which didn’t really impact the race as everybody was at a standstill.*

*We then removed pit stops and just sorted by race time, which resulted in closer to accurate results: 18, 26, 12, 15, 10, 8, 6, 4, 1, 2, 0, 0. There are three positions that are mistaken, and they are all off by exactly one spot (1st vs 2nd, 3rd vs 4th, and 8th vs 9th).*

*Another attempt was made by removing all the rows with red flags. This did not work as the resulted with a final sorting of 12, 26, 15, 18, 10, 2, 1, 4, 0, 6, 8, 0.*

*The timing data is a known issue such that there is an entire column dedicated to determining if the start lap time and end lap time are synced up correctly (CALCULATE PERCENTAGE).*

*Since our other attempts left us worse off than the original calculation, we stuck with it with the caveat that sometimes non-finishers are included in the final lap’s data. It was not prudent to continue attempting to manipulate the data to obtain the most accurate results. And determining the number of non-finishing or lapped drivers who were interspersed in the final results was not feasible to calculate as it was a manual adjustment for 103 races, which would be a comparison of 2060 rows to real-world results and then further attempts to adjust or fix the issue.*

*In addition to the dataframes from the laps and results, there were separate dataframes for weather and messages from race control, and the results. Those were also formed in a similar manner.  The reason for including the non-timing data was the impact this data has on the decision to make a pit stop or to stay out. Weather affects tire life and going from a torrential downpour to a blazing sun would require a change of tires. Race control messages can be sent out when there is a yellow flag or other issue that requires drivers to reduce their speed by a certain percentage. Under normal conditions, with the drivers driving at full speed, it is normal for a pit stop to have a driver lose 20 seconds. Under yellow flag conditions, that loss is reduced to 10 seconds and they get fresher tyres.*

*The race control messages dataframe would be useful for highly specific analysis to see how a specific team/driver reacts to a change in the field or a penalty. The laps dataframe already has a column called “TrackStatus” that incorporates flags, so we did not look at Race Control Messages closely.*

*The weather dataframe was also, unfortunately, skipped over. The weather data was accumulated on the minute and had a completely different initialized time compared to the laps dataframe. As such it was impossible to merge the data as the measurements were just on different scales. A future project, figuring out how to incorporate them, would definitely be worthy of exploring.*

*An F1 team involves somewhere between 300 and 1200 personnel. There are around 20 people in the pit stop alone.*



<https://www.caranddriver.com/news/a28567721/fastest-pit-stop-german-grand-prix/>

*The person who makes the decision for a pit stop is called the Strategy Engineer. The Strategy Engineer uses data to determine pit stop strategy – when to pit and what tyre compound to use.*

*There are many different types of tyre compounds depending on the weather and our data is not 100% accurate for this either. There are six dry tyre types, ranging in hardness from C0 (very hard) to C5 (very soft). We only get told if it is soft, medium, or hard. There are also intermediate tyres and wet tyres.*

*In addition to the compound, the track status, the location of competing drivers plays a role in pit stop decision making. It is not uncommon to have a strategy of “do the opposite” of the competing driver. Since pit stops are one of the few things that can be changed during the race, and seeing as strategists do care about when another team makes a pit stop, it seems to be a good challenge to try and predict when that pit stop will occur.*

*An important random event, or shock, that can alter the face of the race is a flag. There are ten distinct flags which have their own meaning. A yellow flag, for example, can be the result of a minor car crash. In a yellow flag all drivers must slow their speed and passing another car is not allowed. As a result, yellow flags are an ideal time to make a pit stop for fresh tyres.*

*Pretorious (2022) has a chart that shows the breakdown of flags and their meaning:*

|  |  |
| --- | --- |
| ***Flag Colour/Type*** | ***Meaning*** |
| *Yellow* | *Hazard on track.* |
| *Green* | *Normal racing conditions apply.* |
| *Red* | *Session is suspended.* |
| *Blue* | *A faster car is approaching. Move aside (when a driver is being lapped)* |
| *Yellow and Red Stripes* | *Track is slippery.* |
| *Black with Orange Circle* | *A driver has a mechanical issue and must return to the pits.* |
| *Black and White* | *Warning for unsportsmanlike behavior.* |
| *Black* | *Disqualification.* |
| *White* | *Slower moving vehicles ahead, or miscellaneous vehicles on the track.* |
| *Chequered* | *The session is finished (no new laps may be started).* |

*The flag data can also be found in the “Laps” dataframes under track status. Below is a breakdown of the variables on the Laps dataframe, and underneath that is the breakdown of the various TrackStatus definitions.*

*‘1’: Track clear (beginning of session ot to indicate the end of another status)*

*‘2’: Yellow flag (sectors are unknown)*

*‘3’: ??? Never seen so far, does not exist?*

*‘4’: Safety Car*

*‘5’: Red Flag*

*‘6’: Virtual Safety Car deployed*

*‘7’: Virtual Safety Car ending (As indicated on the drivers steering wheel, on tv and so on; status ‘1’ will mark the actual end)*

*With 86% of the results being determined by car construction, there are only certain variables that can affect the outcome of a race. Once the race starts there is no ability to change the driver or make modifications to the engine. The most significant impact once the race starts concerns pit stops. When to make a pit stop and what tyres to put on the car.  Deciding what tyres to put on the car is an optimization problem that has not yet been solved in certain situations and has been fully solved in other instances, such as when all 20 drivers use the same tyre type. We are not trying to optimize the tyre decision ahead of time; we want to solve the question if a team will make a pit stop or not using the theory of Bayesian Equilibrium.*

*Solving for Bayesian Equilibrium requires the use of Python and R to extract, manipulate, and visualize the data. We have global variables, local variables, and random shocks. The choice to utilize Bayesian statistics can be answered using the list provided by Santos et al. (2018) such as the incorporation of prior beliefs, probabilistic estimates, and updating the likelihood lap-after-lap.*

*The idea stemmed from Bruce Bueno de Mesquita’s book The Predictioneer’s Game (2010) where he discusses his prediction of political policies individuals will institute.  Prior to utilizing Bayesian statistics, we created a Poisson Regression similar to the one by Chimka and Talafuse (2016). It is comparable discrete data comparing laps driven before making a pit stop to golf strokes hit before making it in the hole.*

**When incorporating the Bayesian model, we drew inspiration from Dolmeta et al. (2021) who utilized a Bayesian GARCH approach with a smooth functional component to measure intra-season variability. We bring it from an intra-seasonal component to an intra-race component.**

**Another consideration was Spiliopoulos (2016) finding that underdogs introduce more randomness than the favorites in tennis. Essentially, a top rated player would not alter his game based upon who he is playing whereas a lower rated player would introduce more variation.**

*Tulabadin and Rudin (2014) performed a similar analysis on NASCAR races. They found that machine learning algorithms such as ridge regression, SVR, LASSO, and random forests are significantly more predictive. Their question was more of tyre optimization - in NASCAR it was whether to change 4 tyres or 2. In Formula1, all 4 tyres must be changed at the same time but there is variability as to the type of tyre used as there are six variations (Pirelli).*

**Another useful paper was Wunderlich (2021) who found that randomness in scoring goals in soccer (or football) decreased as time passes. That decrease in randomness, or variability, is something we would expect to see in the probabilities of making a pitstop - as a driver continues not to make a pitstop, it becomes more likely that they will make a pitstop in the next lap.**

**Issues with accumulating and cleaning the data prevented much of the exploration we had originally planned given the timeline to turnaround the thesis.**

**The first 2400 lines of code involved obtaining all the data and performing ETL methods for future analysis***. After getting the data into an imperfect but acceptable format, we performed a poisson regression on the data.  We want to predict when a pit stop is being made so we pared the dataframe down into just those laps involving a pit stop. We also removed duplicate and unnecessary column values. This reduced the columns to 16 and rows to 3415. The total time variable was changed into a total seconds rather than a dtype(‘<M8[ns]’) type.*

*We were left with five string variables. To get them in the proper form, we had to transform them into categories and then codify them. We can finally start with statistical analysis. The first thing we did was aggregate the pit stop only data by laps, calculating the 'mean', 'min', 'max', 'count'. We had to transform the count into a float. We were then able to visualize the number of pit stops both from a total number and from a normalized view point. (INSERT STATISTICS)*

*We then looked specifically at the tyre life when the pit stop was made, using the same parameters in the aggregated calculations for pit stops by lap. (INSERT STATISTICS)*

*Our next step was to perform statistics. Sci-kit Learning provides a Python framework for calculating the Poisson regression. It lead to a value error in our data using Lap Number as the Y-variable and Tyre Life as the Predictor-variable due to array dimensionality. We could reshape the array, but we found another package called statModels that allows for Poisson Regression. Around half of the data was able to be summarized without further transformation. For Fresh Tyres, we replaced with dummy variables. The remaining data that provided issues were all strings (Team, Driver, DriverNumber, Compound, and TrackStatus) that were transformed into categories and then codified.*

*Looking solely at the Saudi Arabia 2021 race data for only the pit stops, the average lap number a pit stop was made was 14.833. The variance was 46.567 (over three times as much). This breaks the poisson assumption of an equal mean-variance value, or equally dispersed values. To counteract the overdispersion, we started building Consul’s Generalized Poison regression model (*[*https://www.jstor.org/stable/1267389*](https://www.jstor.org/stable/1267389)*) , know as GP-1. Some additional errors occurred running this regression. For the “Compound”, “TrackStatus”, “Gap”, “Cumulative”, “CumulativeLap”, “CumulativePit”, “PitTime” factors there was an error where the maximum likelihood optimization failed to converge. The factor “PitInTime” had a divide by zero error.  “The factor “PitOutTime” had a linear algebra singular matrix error.*

*We then created a “gap” column to indicate the lag each driver has. This is something they know in the race and take account of when making a pit stop decision - for example, if a pit stop takes 20 seconds on average, if a driver is ahead by 25 seconds then a pit stop will be made; whereas if a driver is ahead by 15 seconds, a pit stop will not be made.*

*Sorting the data by cumulative race time proved troublesome as the pit stop time played a large role. In a red flag situation, for example, the race stops and restarts many minutes later. Every driver has to come into the pit stop lane during this stoppage so a driver more in the lead would sit longer than someone behind him.  The first lap also had a different pit out time for each driver on the same course.*

*Another potential build upon is Famoye's Restricted Generalized Poison regression model, known as GP-2.* [*https://www.tandfonline.com/doi/abs/10.1080/03610929308831089*](https://www.tandfonline.com/doi/abs/10.1080/03610929308831089)*We tested half of the variables, including those which did not have an issue with the normal Poisson model or Consul’s model, and all resulted in a convergence error.*

*There are only two options left: a quasi-Poisson process or a negative binomial regression. With the underlying criterion of mean-variance equality being violated, we built a negative binomial regression analysis. We could have attempted to fit an additional dispersion parameter using a quasi-poisson model, but then we would be unable to directly use model selection or other diagnostic techniques which require an underlying likelihood.*

*There are two common negative binomial formulas. In Joseph Hilbe’s Negative Binomial Regression book (*[*https://www.cambridge.org/core/books/negative-binomial-regression/12D6281A46B9A980DC6021080C9419E7*](https://www.cambridge.org/core/books/negative-binomial-regression/12D6281A46B9A980DC6021080C9419E7)*), he  states that "NB2 is the standard form of negative binomial used to estimate data that are Poisson-overdispersed, and is the form of the model which most statisticians understand by negative binomial. NB2 is typically the first model we turn to when we discover that a Poisson model is overdispersed."*

*The statmodels package has an entire method for the NB2 model: (*[*https://timeseriesreasoning.com/contents/negative-binomial-regression-model/*](https://timeseriesreasoning.com/contents/negative-binomial-regression-model/)*)*

|  |
| --- |
| ***class*** *statsmodels.genmod.families.family.NegativeBinomial(link****=****None, alpha****=****1.0)* |

*The default value of alpha is 1 which is not necessarily correct. There is a way to estimate the alpha. In Cameron and Trivedi’s book, they propose calculating alpha by using a technique they call auxiliary OLS regression without a constant (*[*https://faculty.econ.ucdavis.edu/faculty/cameron/racd2/*](https://faculty.econ.ucdavis.edu/faculty/cameron/racd2/)*)*

*We processed the negative binomial algorithm using a training and test set. #Set up the X and y matrices for the training and testing data sets#Add the λ vector as a new column called 'BB\_LAMBDA' to the dataframe of the training data set#add a derived column called 'AUX\_OLS\_DEP' to the pandas DataFrame. This new column will store the values of the dependent variable of the OLS regression#use patsy to form the model specification for the OLSR#Configure and fit the OLSR model. The result for the alpha was a -0.06. The actual alpha should be between 0 and 1. We did take the absolute value in and plugged it into a t-test online calculator to determine the significance. It was significant. It also lead to a decent predictive result. It cannot be trusted due to the manual adjustment of the alpha value though:*

*Since negative binomial did not work due to the negative alpha, we took a brief glance at a random forest regression. The accuracy score for the random forest was 0.8. There was a zero division error for the classification report so results such as the F-score were not available. There was a parameter to account for the error but*

*The difference in variance and mean can be explained by the compound type. The average number of laps a tyre undergoes before making a pit stop is dependent upon the type of tyre.*

*Since we tested the Poisson Regression, Consul’s Poisson, Famoye’s Poisson, Random Forest, and a Negative Binomial algorithms on a singular dataframe and it appears to work, we moved to apply it to the full dataset.*

*Looking at a normal Poisson Regression, the output showed everything was statistically significant. What initially seems interesting is that the DriverNumber and Driver do not have matching results, however driver numbers can change year-over-year so that could lead to the variance seen in the results.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Variable*** | ***Coef*** | ***std err*** | ***z*** | ***P>|z|*** | ***[0.025*** | ***0.975]*** |
| *CompoundCat* | *0.6502* | *0.001* | *781.636* | *0* | *0.649* | *0.652* |
| *TyreLife* | *0.1093* | *0* | *980.031* | *0* | *0.109* | *0.11* |
| *TotalTime* | *0.0007* | *6.60E-07* | *1008.049* | *0* | *0.001* | *0.001* |
| *Stint* | *1.0956* | *0.001* | *958.047* | *0* | *1.093* | *1.098* |
| *GapSeconds* | *0.0028* | *2.62E-05* | *106.938* | *0* | *0.003* | *0.003* |
| *Fresh Tyre False* | *3.1201* | *0.006* | *502.872* | *0* | *3.108* | *3.132* |
| *Fresh Tyre True* | *3.3663* | *0.004* | *862.888* | *0* | *3.359* | *3.374* |
| *TeamCat* | *0.2797* | *0* | *861.445* | *0* | *0.279* | *0.28* |
| *DriverNumberCat* | *0.1321* | *0* | *827.976* | *0* | *0.132* | *0.132* |
| *DriverCat* | *0.1283* | *0* | *869.119* | *0* | *0.128* | *0.129* |
| *TrackStatusCat* | *0.3362* | *0.001* | *446.083* | *0* | *0.335* | *0.338* |

*To perform the random forest, we had to remove the RACE category. We also created a train/test set for prediction purposes. The accuracy score was 42.6% (significantly worse than the RF accuracy score of 80% when looking only at one race) The results of the RF show the variable importance as shown below:*

|  |  |
| --- | --- |
| ***Variable*** | ***Importance*** |
| *TotalTime'* | *17%* |
| *TyreLife'* | *17%* |
| *GapSeconds'* | *10%* |
| *GridPosition'* | *7%* |
| *Position'* | *7%* |
| *DriverCat'* | *6%* |
| *DriverNumberCat'* | *6%* |
| *TeamCat'* | *5%* |
| *CompoundCat'* | *5%* |
| *Year'* | *5%* |
| *TrackStatusCat'* | *5%* |
| *Points'* | *4%* |
| *Stint'* | *3%* |
| *FreshTyre'* | *2%* |

*We also re-ran the mean and variane of the lap number. For Saudi Arabia, the variance was 46.567 with a mean of 14.833. For the full dataset the variance was 225.903 with a mean of 26.847. This has even more over-dispersion than the original analysis.*

*Chart, bar chart

Description automatically generated*

*We then looked at the mean and variance of the tyre compounds. In Saudi Arabia, the mean was .479 with a variance of .297, a difference of 60%. Looking at the full dataset, the mean was 2.904 with a variance of 3.01, a difference of -4%. There is still overdispersion although it’s relatively miniscule.*

*As with the Saudi Arabia specific analysis, Consul’s Generalized Poison regression model elicited very little feedback.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Variable*** | ***coef*** | ***std err*** | ***z*** | ***P>|z|*** | ***[0.025*** | ***0.975]*** |
| *Fresh Tyre False* | *3.1503* | *0.019* | *169.118* | *0* | *3.114* | *3.187* |
| *Fresh Tyre True* | *3.3541* | *0.013* | *263.023* | *0* | *3.329* | *3.379* |
| *alpha* | *2.3828* | *0.054* | *44.238* | *0* | *2.277* | *2.488* |
| ***Variable*** | ***coef*** | ***std err*** | ***z*** | ***P>|z|*** | ***[0.025*** | ***0.975]*** |
| *TeamCat* | *0.272* | *0.001* | *215.62* | *0* | *0.27* | *0.274* |
| *alpha* | *3.7426* | *0.065* | *57.343* | *0* | *3.615* | *3.87* |
| ***Variable*** | ***coef*** | ***std err*** | ***z*** | ***P>|z|*** | ***[0.025*** | ***0.975]*** |
| *DriverCat* | *0.1267* | *0.001* | *221.616* | *0* | *0.126* | *0.128* |
| *alpha* | *3.6547* | *0.065* | *55.802* | *0* | *3.526* | *3.783* |
| ***Variable*** | ***coef*** | ***std err*** | ***z*** | ***P>|z|*** | ***[0.025*** | ***0.975]*** |
| *DriverNumberCat* | *0.1297* | *0.001* | *211.178* | *0* | *0.128* | *0.131* |
| *alpha* | *3.7518* | *0.065* | *58.095* | *0* | *3.625* | *3.878* |

*We have now done regular Poissoin and Consul’s Poisson. Famoye’s Poissoin was rife with convergence issues so our next step is Negative Binomial. The first step is to run a GLM Poisson again using test/training data. And this time we found two variables that were not statistically significant (Driver Number and Driver).*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Variable*** | ***coef*** | ***std err*** | ***z*** | ***P>|z|*** | ***[0.025*** | ***0.975]*** |
| *Intercept* | *1.8562* | *0.022* | *85.897* | *0* | *1.814* | *1.899* |
| *DriverNumberCat* | *-0.0001* | *0* | *-0.379* | *0.704* | *-0.001* | *0.001* |
| *CompoundCat* | *-0.0065* | *0.002* | *-2.872* | *0.004* | *-0.011* | *-0.002* |
| *TyreLife* | *0.0252* | *0* | *63.039* | *0* | *0.024* | *0.026* |
| *FreshTyreCat* | *0.1804* | *0.009* | *20.299* | *0* | *0.163* | *0.198* |
| *Stint* | *0.1979* | *0.005* | *37.643* | *0* | *0.188* | *0.208* |
| *GapSeconds* | *-0.0002* | *5.24E-05* | *-4.21* | *0* | *0* | *0* |
| *TeamCat* | *0.0044* | *0.001* | *4.934* | *0* | *0.003* | *0.006* |
| *DriverCat* | *-0.0001* | *0* | *-0.317* | *0.752* | *-0.001* | *0.001* |
| *TrackStatusCat* | *0.0031* | *0.001* | *2.506* | *0.012* | *0.001* | *0.005* |
| *Points* | *0.0026* | *0.001* | *5.137* | *0* | *0.002* | *0.004* |
| *TotalTime* | *0.0001* | *3.07E-06* | *40.396* | *0* | *0* | *0* |

*The next step is to add the lambda column to the dataframe of the training step as it is an essential part to calculate the alpha in the equation*

*Diagram

Description automatically generated*

*Unfortunately, as in our Saudi Arabia data, the alpha calculated was a negative 0.029378. Alpha values cannot be negative so nothing further could be done with the negative binomial.*

*Hypothetically, if the alpha was a positive 0.029378, we could have created a predicted vs actual analysis with the following graph as output:*

*Chart

Description automatically generated*

*It’s not valid though, it’s just included to show that I can do some predictive work even though the data didn’t work with me.*

*My next attempt was working with a Quasi-Poisson process. There is no set feature in the Python framework that allows for a Quasi-Poisson model to be ran. The one model online which used a qausi-poisson process essentially backdoored R into the Python script, transforming the dataframe into an r dataframe and then running the R script on it. When this was done to our data, we were able to get it into the R format but “glm” was not recognized as an R command.*

*At this point, we decided to simply write the dataframe into a csv and work on it from RStudio. Unfortunately, this meant that data needed replicated manipulation – such as making the Driver or Team name a factor. After cleaning the data, we installed the PSCL package to allow us to run the statistical tests. In R, the normal Poisson AIC was 20872, the Negative Binomial AIC was similar at 20857. All GLMs use the same log-linear mean function (log(µ) = x >β) but make different assumptions about the remaining likelihood.*

*We then tried some further statistical tests, such as Generalized Estimating Equations through the R Package, GeePack. This did not work due ot the model matrix being rank deficient. We then tried the FlexMix package -* [*https://www.jstatsoft.org/article/view/v011i08*](https://www.jstatsoft.org/article/view/v011i08)

[*https://ro.uow.edu.au/cgi/viewcontent.cgi?referer=&httpsredir=1&article=3410&context=commpapers*](https://ro.uow.edu.au/cgi/viewcontent.cgi?referer=&httpsredir=1&article=3410&context=commpapers)

*FlexMix implements a general framework for fitting discrete mixtures of regression models in the R statistical computing environment: three variants of the EM algorithm can be used for parameter estimation, regressors and responses may be multivariate with arbitrary dimension, data may be grouped, e.g., to account for multiple observations per individual, the usual formula interface of the S language is used for convenient model specification, and a modular concept of driver functions allows to interface many different types of regression models. Existing drivers implement mixtures of standard linear models, generalized linear models and model-based clustering. FlexMix provides the E-step and all data handling, while the M-step can be supplied by the user to easily define new models.*

*Flex mix did not work due to NAN Log Likelihood Error which can can appear for numerous reasons.*

*The NLME package was considered as well but removed from consideration due to the Guassian distribution requirement (*[*https://cran.r-project.org/web/packages/nlme/nlme.pdf*](https://cran.r-project.org/web/packages/nlme/nlme.pdf)*).*

*Our next, and final attempt, is a Bayesian Analysis of Poisson data.* [*https://bookdown.org/kevin\_davisross/bayesian-reasoning-and-methods/poisson.html*](https://bookdown.org/kevin_davisross/bayesian-reasoning-and-methods/poisson.html)