Formula One 2022 Predictions
11/19/2022

Fundamentals of Data Science
Rubber Casino

Marc Ebersberger Jr. Logan W. Gillum

Abstract

The goal behind this project is to predict the winner of the 2022 Grand Prix by analyzing drivers on a per circuit basis. Drivers will be assigned some form of score based on their history at each circuit over the span of sample years used in this study. These scores will be combined with the hopes of producing a score that can predict the outcome of the race.

Three factors will be used to build this prediction score:

Performance Score: How a drivers lap time compares to other driver lap times lap over lap.

Consistency Score: How a drivers lap time varies lap over lap.

Team Score: How the race team affects the driver.

1 Observations

There are many factors that can affect a drivers performance, so it is unlikely that the data captured can tell the entire story. Without additional data, we cannot account for weather, strategy, tire wear, and car condition to name a few. This missing data does have an effect on a driver's performance, but we feel we can come to a reasonable score for a driver with the data available.

1.1 Driver experience

We are using 10 years of racing data to support our model, but not all drivers have been racing for that long. Our range of experience goes from 1 year to 10 years. We believe factoring in years of experience into their total score will be the best way to compare drivers of varying years.

1.2 DNF

Drivers commonly DNF during races which this leads to missing values throughout the race data. We must understand how to best handle this missing data so that our model does not punish a driver for a single mistake.

1.3 Median vs Mean Lap time

Calculating an accurate lap time is imperative for the driver performance score. We had two options when it came to defining a base lap time. The accuracy of Mean lap time is questionable due to outliers in the data. Figure 1 shows the heavy impact of outliers on the mean lap time. Median lap time appears to be affected less by these outliers, which makes it the best fit for the base lap time.

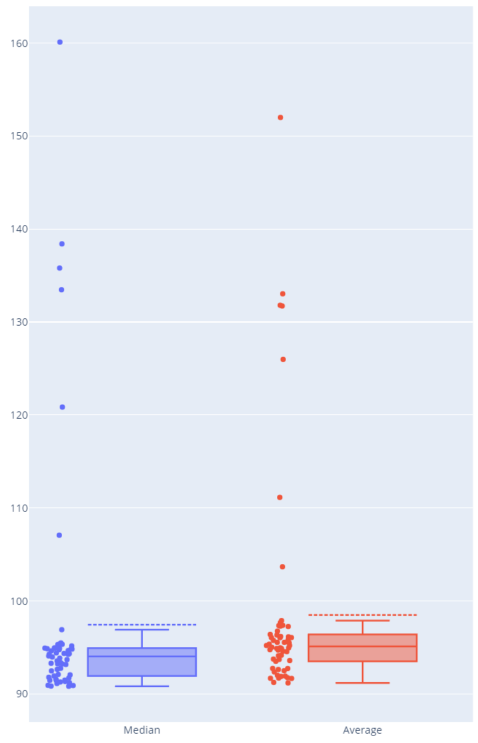


Figure 1: Box-and-Whisker/Scatter for Average vs. Median

2 Assumptions for Outcome

Drivers will more experience will generally have a higher total score.

Drivers with less experience will generally have a lower total score.

Teams will have a significant effect on the outcome, both positive and negative.

The best drivers will typically always show up in a top 5 position.

Drivers that perform better relative to their current partner to be placed higher than them.

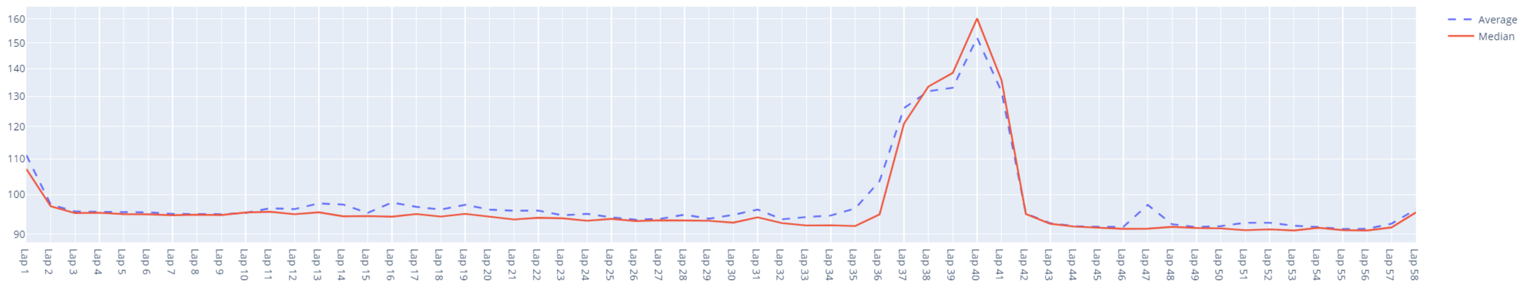


Figure 2: Average vs. Median

3 Outliers

F1 data is notorious for having outliers in the lap time data. This is reflected heavily in our data set.

The Average line in Figure 2 shows a these outliers in the form of short spikes and large spikes. Short spikes are indicative of an individual driver mistakes, or pit stops, whereas large spikes indicate a change in track conditions such as safety cars and flags that drastically reduce lap times across the board. Our lap over lap scoring is not negatively affected by these larger spikes as they typically impact all racers equally.

```
            "number": "20",
            "Timings": [
                {
                    "driverId": "button",
                    "position": "1",
                    "time": "1:32.880"
                },
                {
                    "driverId": "perez",
                    "position": "2",
                    "time": "1:34.423"
                },
                {
                    "driverId": "hamilton",
                    "position": "3",
                    "time": "1:34.233"
                }
```

```
            "number": "40",
            "Timings": [
                {
                    "driverId": "button",
                    "position": "1",
                    "time": "2:39.748"
                },
                {
                    "driverId": "vettel",
                    "position": "2",
                    "time": "2:40.235"
                },
                {
                    "driverId": "hamilton",
                    "position": "3",
                    "time": "2:40.660"
                }
```

4 Cleaning

4.1 Noisy Data

The majority of our data frames are built from the Race Data. Instead of removing information, to handle the noisy data we built dataframes that only include what we need, thus cutting out irrelevant information. This included circuitID as the title, unique drivers as rows and laps as columns.

4.2 Missing Data

We have situations of missing data. Drivers who DNF, we have nan's that fill in the remainder laps. Drivers who don't have 10 years of track data. Currently we do not fill this in, but rather factor the number of available years into a drivers total score

4.3 Grouping

We are only interested in drivers and circuits that are active in the 2022 Grand Prix. The data set we are using includes all drivers and all circuits over the past 10 years. Grouping the drivers and circuits by active/inactive allows us to filter out extraneous data, which results in less computation time and expended resources.

5 Raw Data

5.1 Driver

This data holds basic information on a unique driver. Mainly used for the ‘driverId‘ which is used within the Race Data to identify a driver.

```
{
    "driverId"
    "permanentNumber"
    "code"
    "url"
    "givenName"
    "familyName"
    "dateOfBirth"
    "nationality"
}
```

5.2 Circuit

This data holds all the information pertaining to a race given a specific season (year) and round (circuit). This is where the bulk of our predictive model gets its information from. This tells us lap by lap what drivers are active and their lap time.

```
{
    "season"
    "round"
    "url"
    "raceName"
    "Circuit": {
        "circuitId"
        "url"
        "circuitName"
        "Location": {
            "lat"
            "long"
            "locality"
            "Country"
        }
    },
    "date"
    "time"
    "Laps": [
        "number": "1",
        "Timings": [
            {
                "driverId"
                "position"
                "time"
            },
        ]
    ]
}
```

5.3 Schedule

Schedule data is used to describe all the circuits within a given year for every year in the data set. Each year comes with its own set of the following variables:

```
{
    "season"
    "round"
    "url"
    "raceName"
    "Circuit": {
        "circuitId"
        "url"
        "circuitName"
        "Location": {
            "lat"
            "long"
            "locality"
            "country"
        }
    },
    "date"
    "time"
}
```

6 Pre-Processing

6.1 Wrangling

The raw data we pull is formatted in JSON. In order to use this data, we built data frames using Pandas to convert the data to CSV format. We started by initializing an empty data frame where all rows are unique drivers within the race. We then set the number of columns equal to the number of laps, and then filled in each cell with the respective drivers lap time. The code snippet below shows our process for accomplishing these steps.

```
for lap in jsondata['Laps']:
    lap_data = []
    for _ in range(0, len(drivers)):
      try:
         driver = next(item for item in lap['Timings']
                  if item['driverId']==drivers[_])

         # Create a tuple of driver and lap time
         lap_data.append([driver['driverId'],
            driver['time']])
      except:
         lap_data.append([drivers[_],None])

# Sort it so it matches the rows
lap_data.sort(key = lambda x: x[0])

# Add new column of lap time
time_list = list(list(zip(*lap_data))[1])

df[f"Lap {lap['number']}"] = time_list
```

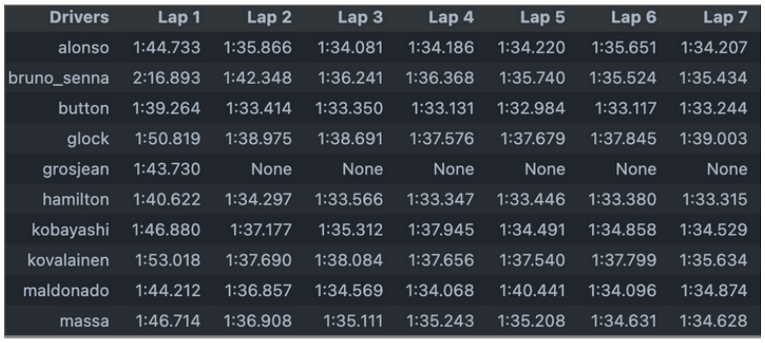
6.2 Cleaning

The source of this F1 race data did lots of cleaning on the data prior to publishing it to their database. We decided to check to make sure this was the case with our data. The only inconsistencies that were present pertained to how lap times were recorded.
The standard format for a lap time is **Min:Sec.Milli**, but we found a few rare cases where the lap times would have a prepended character or symbol. This was easy for us to clean from the data while still maintaining the lap time data.

```
# Removes + in front of the times
    if '+' in time:

        dirty_time = time[1:]

        # Rare case 's' or 'sec' is in the time
        # or
        # Rare case there is a space infront
        # of the hour time EX: ' 1.06.7'

        if 'sec' in dirty_time:
            dirty_time = dirty_time[:-4]

        elif 's' in dirty_time:
            dirty_time = dirty_time[:-1]

        elif dirty_time[:1] == ' ':
            dirty_time = dirty_time[1:]
```

6.3 Transformation

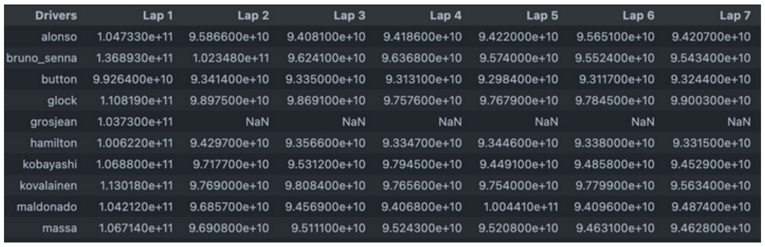
The lap times we pull from the source are read in as strings. We use the laptime as the source of our calculations, so we had to convert this into a workable data type. We broke this down into two phases; Phase 1 converts the string time into a timedelta representation allowing us to perform time calculation, and Phase 2 takes our lap time and converts it into a nanosecond representation.



Drivers	Lap 1	Lap 2	Lap 3	Lap 4	Lap 5	Lap 6	Lap 7
alonso	1:44.733	1:35.866	1:34.081	1:34.186	1:34.220	1:35.651	1:34.207
bruno_senna	2:16.893	1:42.348	1:36.241	1:36.368	1:35.740	1:35.524	1:35.434
button	1:39.264	1:33.414	1:33.350	1:33.131	1:32.984	1:33.117	1:33.244
glock	1:50.819	1:38.975	1:38.691	1:37.576	1:37.679	1:37.845	1:39.003
grosjean	1:43.730	None	None	None	None	None	None
hamilton	1:40.622	1:34.297	1:33.566	1:33.347	1:33.446	1:33.380	1:33.315
kobayashi	1:46.880	1:37.177	1:35.312	1:37.945	1:34.491	1:34.858	1:34.529
kovalainen	1:53.018	1:37.690	1:38.084	1:37.656	1:37.540	1:37.799	1:35.634
maldonado	1:44.212	1:36.857	1:34.569	1:34.068	1:40.441	1:34.096	1:34.874
massa	1:46.714	1:36.908	1:35.111	1:35.243	1:35.208	1:34.631	1:34.628

Figure 3: Pre-Transformation Dataframe

```
def time_to_ns(raw_time):
    # Catch NaaN
    try:
        dirty = datetime.strptime(
            raw_time, '%M:%S.%f').time()
        nanoseconds = (dirty.minute*6e10)
                        +(dirty.second*1e9)
                        +(dirty.microsecond*1e3)
        return nanoseconds
    except:
        return raw_time

# Copy original dataframe to work on
test_df = df.copy()

# Convert all times in a column to nanosecond
for col in test_df.columns[1:]:
    test_df[col] = test_df[col].apply(
        lambda x : time_to_ns(x))
```



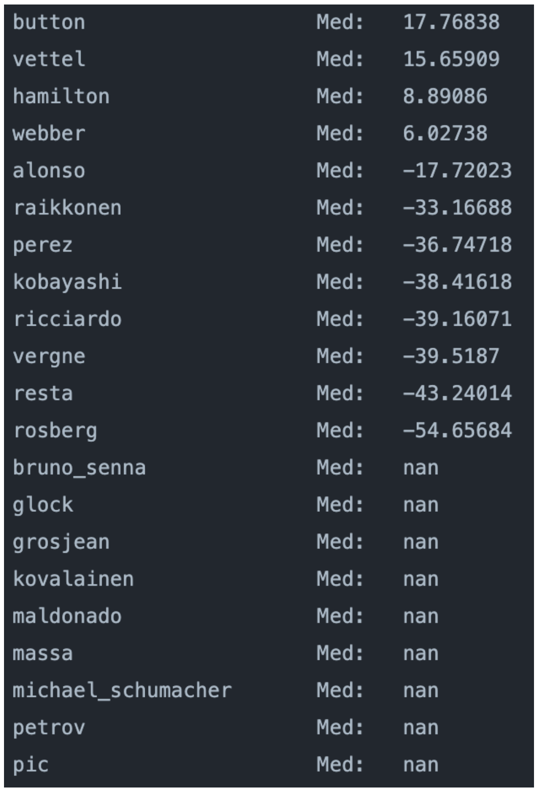
Drivers	Lap 1	Lap 2	Lap 3	Lap 4	Lap 5	Lap 6	Lap 7
alonso	1.047330e+11	9.586600e+10	9.408100e+10	9.418600e+10	9.422000e+10	9.565100e+10	9.420700e+10
bruno_senna	1.368930e+11	1.023480e+11	9.624100e+10	9.636800e+10	9.574000e+10	9.552400e+10	9.543400e+10
button	9.926400e+10	9.341400e+10	9.335000e+10	9.313100e+10	9.298400e+10	9.311700e+10	9.324400e+10
glock	1.108190e+11	9.897500e+10	9.869100e+10	9.757600e+10	9.767900e+10	9.784500e+10	9.900300e+10
grosjean	1.037300e+11	NaN	NaN	NaN	NaN	NaN	NaN
hamilton	1.006220e+11	9.429700e+10	9.356600e+10	9.334700e+10	9.344600e+10	9.338000e+10	9.331500e+10
kobayashi	1.068800e+11	9.717700e+10	9.531200e+10	9.794500e+10	9.449100e+10	9.485800e+10	9.452900e+10
kovalainen	1.130180e+11	9.769000e+10	9.808400e+10	9.765600e+10	9.754000e+10	9.779900e+10	9.563400e+10
maldonado	1.042120e+11	9.685700e+10	9.456900e+10	9.406800e+10	1.004410e+11	9.409600e+10	9.487400e+10
massa	1.067140e+11	9.690800e+10	9.511100e+10	9.524300e+10	9.520800e+10	9.463100e+10	9.462800e+10

Figure 4: Post-Transformation Dataframe

7 Calculating Performance Score

Performance score is one of the scores we use for our predictive model. This score is given to a driver on a per lap basis, depending upon their lap time percent different from the median lap time. Drivers who were faster than the median were given a positive score, while those who were slower were given a negative score. This produces a performance score for a circuit and year for a driver, that when combined with all the other years, gives us a total score for a driver and circuit. This score is purely a reflection of how a driver is compared to other drivers, and does not factor in team, consistency, strategy, etc.

```
# Percent difference for driver time and average time
def percent_diff(driver_time,average_time):

    diff = abs((driver_time - average_time)
            /((driver_time + average_time)/2))*100

    if driver_time > average_time:
        return -abs(diff)
    return diff

# Stores Tuples of driver and score
total = []

# Iterate through each drivers times
for driver in drivers:
    # Base score
    score = 0
    # Filters dataframe to just a drivers row
    filter = test_df.loc[test_df['Drivers'] == driver]
    # Iterate through each lap
    for lap in filter.columns[1:]:
        score += percent_diff(filter[lap].values[0],
                              test_df[lap].median())

    total.append((score,driver))
```



```
button              Med:   17.76838
vettel              Med:   15.65909
hamilton            Med:   8.89086
webber              Med:   6.02738
alonso              Med:   -17.72023
raikkonen           Med:   -33.16688
perez               Med:   -36.74718
kobayashi           Med:   -38.41618
ricciardo           Med:   -39.16071
vergne              Med:   -39.5187
resta               Med:   -43.24014
rosberg             Med:   -54.65684
bruno_senna         Med:   nan
glock               Med:   nan
grosjean            Med:   nan
kovalainen          Med:   nan
maldonado           Med:   nan
massa               Med:   nan
michael_schumacher  Med:   nan
petrov              Med:   nan
pic                 Med:   nan
```

Figure 5: Output of Total

The outcome of the Performance Score calculation appears to be a relationship between Performance Score and the Position variable found in Circuit data.

If we compare performance score placements in Figure 5 with the actual positions held by drivers, we can see this relationship clearly. The cause behind our performance score relating to final positions is the fact that we are comparing a driver to all other drivers on a lap over lap basis. If a driver is performing consistently better than other drivers lap over lap, they will earn a higher Performance Score and also place higher in positions granted no outliers occur.

```
{
    "driverId": "button",
    "position": "1",
    "time": "1:30.846"
},
{
    "driverId": "vettel",
    "position": "2",
    "time": "1:29.682"
},
{
    "driverId": "hamilton",
    "position": "3",
    "time": "1:29.867"
},
{
    "driverId": "webber",
    "position": "4",
    "time": "1:29.622"
},
{
    "driverId": "alonso",
    "position": "5",
    "time": "1:33.838"
}
```

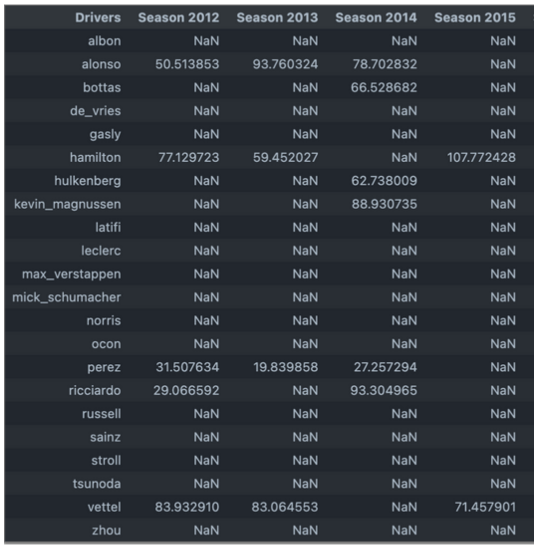
2012 Australian Grand Prix - Top 5 Drivers

8 Data Integration

The data frames for each race for each year need to be integrated into one presentable data frame. We achieve this by creating unique data frames for all the circuits in the 2022 schedule. These data frames are built with 2022 drivers as rows, and each column being their Performance score for each year. This allows us to visualize a drivers performance through 2012-2022.

We used the following code snippets to integrate the data.

```
def get_round(year,circuitId):
    # Returns the round
    # within a given year the circuitId was found

    filepath = Path(f'data/scheduled/{year}.json')
    jsondata = dict()
    with open(filepath, 'r', encoding='utf-8') as file:
        jsondata = json.load(file)

    for circuit in jsondata:
        if circuit['Circuit']['circuitId'] == circuitId:
            return circuit['round']
    return None


# Main Loop
df = pd.DataFrame()
df['Drivers'] = current_drivers

# Get All the seasons data for selected circuit
for year in range(2012,2022):
    round = get_round(year,track_id)
    if round != None:
        for driver in current_drivers:
            score_list = get_score(year, round,
                current_drivers)
    else: continue
    df[f'Season {year}'] = score_list
```



Drivers	Season 2012	Season 2013	Season 2014	Season 2015
albon	NaN	NaN	NaN	NaN
alonso	50.513853	93.760324	78.702832	NaN
bottas	NaN	NaN	66.528682	NaN
de_vries	NaN	NaN	NaN	NaN
gasly	NaN	NaN	NaN	NaN
hamilton	77.129723	59.452027	NaN	107.772428
hulkenberg	NaN	NaN	62.738009	NaN
kevin_magnussen	NaN	NaN	88.930735	NaN
latifi	NaN	NaN	NaN	NaN
leclerc	NaN	NaN	NaN	NaN
max_verstappen	NaN	NaN	NaN	NaN
mick_schumacher	NaN	NaN	NaN	NaN
norris	NaN	NaN	NaN	NaN
ocon	NaN	NaN	NaN	NaN
perez	31.507634	19.839858	27.257294	NaN
ricciardo	29.066592	NaN	93.304965	NaN
russell	NaN	NaN	NaN	NaN
sainz	NaN	NaN	NaN	NaN
stroll	NaN	NaN	NaN	NaN
tsunoda	NaN	NaN	NaN	NaN
vettel	83.932910	83.064553	NaN	71.457901
zhou	NaN	NaN	NaN	NaN

Figure 6: Results of Data Integration

9 Consistency Score & Team Score

This is still a work in progress, as we are not sure just yet how we want to factor this in and the weight of it.

10 References

http://ergast.com/mrd/