Final Report for Formula One Predictions

Fundamentals of Data Science
Rubber Casino

Marc Ebersberger Jr.
Logan Gillum

12/6/2022

1 Purpose

The goal of this project was to be able to predict the winner of the 2022 Grand Prix. After our Exploratory Analysis, we concluded that the best approach would be scoring each driver per circuit within the year. We derived the variable 'Performance Score' to evaluate a driver's lap time compared to the median lap time for that lap.

Using the past 9 years of driver race data, We were able to go into a drivers history to assign them a performance score on every circuit they raced. This performance score is then used to help us predict possible point outcomes for that circuit in 2022. This process would result in an ordered list of drivers and their positions based on their overall score for that circuit. We found that our predicted order was pretty far off of reality; however, most of our drivers were only a handful of positions off of their true position.

We concluded this was due to the fact that there were many factors we did not include into our prediction. The main factor seemed to be the team of the driver. Our predictions were geared towards the assumption the driver stayed on the team that they received their performance score on, but this resulted in our worst predictions were due to a driver being on a better team this year. This can be seen by looking at our prediction for George Russel, who was predicted to place 17th, but he actually placed 4th.

2 Assumptions for Outcome

We made the following assumptions during our Exploratory Analysis. These assumptions held up after we finished our analysis. We explained how each assumption affected the outcome of the project below:

A1. Drivers with more experience will generally have a higher total score.

Experience benefited the amount of points accumulated by a driver.

A2. Drivers with less experience will generally have a lower total score.

Experience harmed the amount of points accumulated by a driver.

A3. Teams will have a significant effect on the outcome, both positive and negative.

The quality of the team greater affects the driver's ability to race their best.

A4. The best drivers will typically always show up in a top 5 position.

Performance Score typically reflected the outcome of the race.

A5. Drivers that perform better relative to their current partner to be placed higher than them

Drivers who performed better gained more performance points than those who performed worse.

3 Procedure

3.1 Exploratory Analysis

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.

```
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
```

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 pre-pended 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:]
```

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 1: 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 2: Post-Transformation Dataframe

3.2 Full Analysis

Performance score is a derived variable we used for our predictive model. 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 compares a driver to other drivers, but does not factor in team, consistency, strategy, etc.

Initialize an empty data frame so that we can populate the data frame with all current drivers as of 2022. The data frame will contain columns for each current circuit, and the performance score for each driver at that circuit for the last 10 years.

```
pp_df = pd.DataFrame()
pp_df['Driver'] = current_drivers
for circuitId in current_circuits:
    # Blank score per circuit
    score_list = [0]*len(current_drivers)
    for year in range(2012,2022):
        round = get_round(year, circuitId)
        # Not all circuits are in a year
        if round != None:
            scores = get_performance_score(year,round,current_drivers)
            #combine scores
            score_list = [score_list[i] + scores[i] for i in range(len(score_list))]
    pp_df[circuitId] = score_list
pp_df
```

	Driver	bahrain	jeddah	albert_park	imola	miami	catalunya	monaco	baku	villeneuve	...
0	albon	89.103159	0.000000	0.000000	7.598544	0	71.043959	27.213295	0.000000	0.000000	...
1	alonso	71.676867	0.000000	190.424411	-15.106234	0	215.308146	260.309117	73.410707	231.554612	...
2	bottas	453.832656	85.697144	364.190539	80.799936	0	685.263162	277.871271	277.052720	529.965931	...
3	gasly	134.499444	63.277383	0.000000	-28.781454	0	88.510150	229.811209	46.224608	0.000000	...
4	hamilton	778.088462	112.600267	654.949432	192.534699	0	1088.833158	793.841292	363.696561	915.717513	...
5	kevin_magnussen	33.591365	0.000000	-51.403464	0.000000	0	67.514199	65.668659	44.472498	11.073528	...
6	latifi	0.000000	12.421981	0.000000	34.046139	0	0.000000	0.000000	30.353808	0.000000	...
7	leclerc	94.418262	70.316228	64.867981	224.191050	0	178.835802	0.000000	128.583367	156.261442	...
8	max_verstappen	272.363704	105.659382	312.203597	234.621749	0	614.520330	402.019513	143.379764	305.800506	...
9	mick_schumacher	0.000000	0.000000	0.000000	0.000000	0	0.000000	0.000000	-0.154640	0.000000	...
10	norris	135.217000	56.754542	0.000000	173.311253	0	0.000000	125.976924	68.658162	0.000000	...
11	ocon	65.363710	69.838618	8.873232	-14.542553	0	0.000000	118.284370	51.207646	68.947395	...
12	perez	150.581707	0.000000	19.024409	131.935577	0	115.236850	324.205783	224.010183	137.459389	...
13	ricciardo	268.586612	61.589899	241.269273	135.267724	0	474.892110	531.171107	151.205665	303.535331	...
14	russell	0.000000	0.000000	0.000000	0.000000	0	0.000000	0.000000	0.000000	0.000000	...
15	sainz	95.029329	61.006991	71.293997	165.988932	0	143.039619	262.033546	136.509228	0.000000	...
16	stroll	1.636321	17.765431	-10.879295	50.800160	0	0.000000	0.000000	135.266918	0.000000	...
17	tsunoda	4.159269	0.000000	0.000000	-2.877308	0	0.000000	0.000000	40.285636	0.000000	...
18	vettel	482.067826	0.000000	649.514450	41.058802	0	760.945778	692.440824	388.447288	894.590177	...
19	zhou	0.000000	0.000000	0.000000	0.000000	0	0.000000	0.000000	0.000000	0.000000	...

	Driver	hungaroring	spa	zandvoort	monza	marina_bay	suzuka	americas	rodriguez	interlagos	yas_marina
0	albon	35.792931	103.396781	0.000000	77.586709	33.741853	79.542212	55.691723	102.021110	16.987207	207.156396
1	alonso	329.283765	193.660083	0.000000	152.253192	284.760054	40.550670	228.861667	0.000000	237.095370	159.528233
2	bottas	349.570060	516.095875	126.956188	619.542243	285.312233	493.460558	490.834636	389.897135	340.759251	594.480013
3	gasly	103.262744	152.624251	0.000000	69.488936	23.466647	0.000000	0.000000	74.705417	14.429883	113.271328
4	hamilton	1129.776850	703.027946	175.536347	789.788483	528.132249	778.165133	1012.221355	640.161554	723.226151	943.357389
5	kevin_magnussen	-6.827716	93.144069	0.000000	36.353996	-8.312951	19.764040	32.606955	0.000000	24.995591	7.510592
6	latifi	0.867753	69.256529	0.000000	50.957697	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
7	leclerc	113.007413	197.069461	0.000000	179.862385	52.833605	0.000000	161.911496	173.791183	127.071910	177.035386
8	max_verstappen	474.647510	307.054515	200.409732	165.539630	309.779620	355.007048	549.377170	773.061070	446.874338	675.814010
9	mick_schumacher	0.000000	45.182533	0.000000	-17.334905	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
10	norris	0.000000	98.849276	0.000000	128.942912	28.550134	0.000000	91.897306	0.000000	-0.839117	106.794821
11	ocon	97.405527	195.446928	0.000000	174.284057	23.905880	76.582589	64.154007	0.000000	30.950291	98.362893
12	perez	13.636818	331.973945	0.000000	417.783513	160.380998	147.918348	258.418504	243.397779	223.607733	219.122578
13	ricciardo	336.599071	440.063440	0.000000	424.463869	473.662420	432.283916	367.526366	202.742267	224.292839	389.388029
14	russell	1.416090	57.540821	0.000000	61.424893	0.000000	0.000000	0.000000	0.000000	-22.492943	0.000000
15	sainz	79.526898	80.181959	0.000000	149.588190	84.800424	67.984724	212.555887	15.692497	126.877695	219.927969
16	stroll	61.888279	131.210766	0.000000	154.031066	25.346802	0.000000	0.000000	0.000000	0.000000	37.775020
17	tsunoda	4.863520	45.681449	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	66.948998
18	vettel	876.665061	645.902466	0.000000	465.062069	619.908867	662.730085	615.320618	565.001344	566.299472	681.020575
19	zhou	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Function that takes a year and round. For each driver goes through all there years of racing and gets a total score based upon there performance each year. Returns a list of all current drivers and there scores.

```
def get_performance_score(year,round,current_drivers):
    filepath = Path(f'../data/races/{year}/{round}.csv')
    if path_exist(filepath):
        original_df = pd.read_csv(filepath)
    else:
        raise
    # Copy Original DF
    w_df = original_df.copy()

    # Convert all time columns to a nanosecond representation (actually seconds)
    for col in w_df.columns[1:]: w_df[col] = w_df[col].apply(lambda x : time_to_ns(x))
    # Track scores for current drivers
    score_list = []
    for driver in current_drivers:
        score_avg = 50
        driver_row = w_df.loc[w_df['Drivers'] == driver]
        # Skips drivers who did not participate in this race
        if not driver_row.empty:
            # Skips driver column
            for lap in driver_row.columns[1:]:
                score_avg+=percent_difference(driver_row[lap].values[0],w_df[lap].median())
                # Handle Nan times
                if np.isnan(score_avg):
```

```
                score_avg = 0
        else:
            score_avg = 0
        score_list.append(score_avg)
    return score_list
```

```
norm_df = pp_df.copy()
# Take a drivers score - mean of everybody's / STD of all
for circuit in norm_df.columns[1:]:
    mean = norm_df[circuit].mean()
    std = norm_df[circuit].std()
    norm_df[circuit] = (norm_df[circuit]-mean)/std
norm_df
```

	Driver	bahrain	jeddah	albert_park	imola	miami	catalunya	monaco	baku	villeneuve	...	hungaroring	spa	zandvoort	monza	marina_bay	suzuka	americas	rodriguez	interlagos	yas_marina
0	albon	-0.329396	-0.894404	-0.586893	-0.709745	NaN	-0.480867	-0.736162	-0.981789	-0.617300	...	-0.525295	-0.567377	-0.401698	-0.574852	-0.563838	-0.318448	-0.551640	-0.234812	-0.637925	-0.100402
1	alonso	-0.414551	-0.894404	0.302082	-0.965761	NaN	-0.030848	0.226083	-0.355770	0.186877	...	0.413181	-0.129549	-0.401698	-0.237928	0.693438	-0.477317	0.079416	-0.655072	0.386837	-0.272923
2	bottas	1.452900	1.243828	1.113290	0.115663	NaN	1.435133	0.298582	1.380811	1.223243	...	0.478049	1.434447	1.626452	1.870660	0.696204	1.368041	1.034082	0.951046	0.869467	1.302574
3	gasly	-0.107561	0.684431	-0.586893	-1.119960	NaN	-0.426382	0.100185	-0.587603	-0.617300	...	-0.309551	-0.328596	-0.401698	-0.611392	-0.615303	-0.642538	-0.754588	-0.347335	-0.649831	-0.440476
4	hamilton	3.037416	1.915088	2.470665	1.375567	NaN	2.694032	2.428562	2.119676	2.562937	...	2.972864	2.341173	2.402529	2.638876	1.912417	2.528055	2.934088	1.981970	2.650126	2.566289
5	kevin_magnussen	-0.600661	-0.894404	-0.826864	-0.795425	NaN	-0.491877	-0.577414	-0.602544	-0.578842	...	-0.661580	-0.617109	-0.401698	-0.760910	-0.774478	-0.562011	-0.635764	-0.655072	-0.600640	-0.823566
6	latifi	-0.764809	-0.584463	-0.586893	-0.411526	NaN	-0.702482	-0.848501	-0.722943	-0.617300	...	-0.636973	-0.732977	-0.401698	-0.695012	-0.732841	-0.642538	-0.754588	-0.655072	-0.717013	-0.850771
7	leclerc	-0.303423	0.860058	-0.284064	1.732520	NaN	-0.144620	-0.848501	0.114721	-0.074612	...	-0.278391	-0.113012	-0.401698	-0.113345	-0.468213	-0.642538	-0.164560	0.060833	-0.125401	-0.209508
8	max_verstappen	0.566130	1.741906	0.870595	1.850135	NaN	1.214457	0.811079	0.240899	0.444729	...	0.878001	0.420478	2.799886	-0.177974	0.818754	0.803919	1.247419	2.529428	1.363510	1.597185
9	mick_schumacher	-0.764809	-0.894404	-0.586893	-0.795425	NaN	-0.702482	-0.848501	-0.983108	-0.617300	...	-0.639748	-0.849749	-0.401698	-1.003175	-0.732841	-0.642538	-0.754588	-0.655072	-0.717013	-0.850771
10	norris	-0.104055	0.521680	-0.586893	1.158807	NaN	-0.702482	-0.328455	-0.396298	-0.617300	...	-0.639748	-0.589435	-0.401698	-0.343113	-0.589842	-0.642538	-0.419701	-0.655072	-0.720919	-0.463935
11	ocon	-0.445401	0.848141	-0.545469	-0.959405	NaN	-0.702482	-0.360210	-0.545110	-0.377849	...	-0.328280	-0.120882	-0.401698	-0.138516	-0.613103	-0.330506	-0.520802	-0.655072	-0.572917	-0.494478
12	perez	-0.028973	-0.894404	-0.498080	0.692261	NaN	-0.343011	0.489856	0.928484	-0.139911	...	-0.596142	0.541351	-0.401698	0.960247	0.070460	-0.039853	0.187125	0.347566	0.324043	-0.057058
13	ricciardo	0.547672	0.642327	0.539446	0.729834	NaN	0.778900	1.344231	0.307635	0.436862	...	0.436573	1.065646	-0.401698	0.990391	1.639594	1.118780	0.584729	0.180092	0.327232	0.559683
14	russell	-0.764809	-0.894404	-0.586893	-0.795425	NaN	-0.702482	-0.848501	-0.981789	-0.617300	...	-0.635219	-0.789805	-0.401698	-0.647780	-0.732841	-0.642538	-0.754588	-0.655072	-0.821733	-0.850771
15	sainz	-0.300437	0.627783	-0.254065	1.076241	NaN	-0.256283	0.233202	0.182310	-0.617300	...	-0.385450	-0.679982	-0.401698	-0.249953	-0.308100	-0.365538	0.019995	-0.590429	-0.126305	-0.054141
16	stroll	-0.756813	-0.451139	-0.637682	-0.222610	NaN	-0.702482	-0.848501	0.171716	-0.617300	...	-0.441852	-0.432464	-0.401698	-0.229905	-0.605886	-0.642538	-0.754588	-0.655072	-0.717013	-0.713941
17	tsunoda	-0.744485	-0.894404	-0.586893	-0.827869	NaN	-0.702482	-0.848501	-0.638248	-0.617300	...	-0.624196	-0.847329	-0.401698	-0.924953	-0.732841	-0.642538	-0.754588	-0.655072	-0.717013	-0.608266
18	vettel	1.590875	-0.894404	2.445293	-0.332452	NaN	1.671218	2.009970	2.330741	2.489563	...	2.163505	2.064082	-0.401698	1.173586	2.372099	2.057721	1.487726	1.672360	1.919520	1.616044
19	zhou	-0.764809	-0.894404	-0.586893	-0.795425	NaN	-0.702482	-0.848501	-0.981789	-0.617300	...	-0.639748	-1.068910	-0.401698	-0.924953	-0.732841	-0.642538	-0.754588	-0.655072	-0.717013	-0.850771

Make a copy of the original dataframe and convert all performance scores to Z-score representations.

```
champ_df = norm_df.copy()
# for circuit in champ_df['miami']:
for circuit in champ_df.columns[1:]:
    order = champ_df[circuit].sort_values(ascending=False)
    if str(order.values[0]) == 'nan':
        champ_df[circuit] = [0]*len(champ_df['Driver'])
        continue
    points = [25,18,15,12,10,8,6,4,2,1,0,0,0,0,0,0,0,0,0,0]
    champ_points = merge(list(order.index),points)
    champ_points.sort(key = lambda x: x[0])
    temp = []
    for i in champ_points:
        temp.append(i[1])
    champ_df[circuit] = temp

champ_df
```

	Driver	bahrain	jeddah	albert_park	imola	miami	catalunya	monaco	baku	villeneuve	...	hungaroring	spa	zandvoort	monza	marina_bay	suzuka	americas	rodriguez	interlagos	yas_marina
0	albon	0	0	0	0	0	0	0	0	0	...	0	0	12	0	1	6	0	4	0	4
1	alonso	0	0	8	0	0	8	4	1	8	...	8	2	0	1	8	1	6	0	10	1
2	bottas	15	15	15	4	0	15	8	15	15	...	12	15	15	18	10	15	12	12	12	12
3	gasly	4	8	0	0	0	1	2	0	0	...	4	1	0	0	0	0	0	2	0	0
4	hamilton	25	25	25	15	0	25	25	18	25	...	25	25	18	25	18	25	25	18	25	25
5	kevin_magnussen	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
6	latifi	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7	leclerc	1	12	4	18	0	6	0	2	6	...	6	6	0	8	2	0	2	6	4	2
8	max_verstappen	12	18	12	25	0	12	12	8	12	...	15	8	25	4	12	10	15	25	15	15
9	mick_schumacher	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
10	norris	6	2	0	12	0	0	1	0	0	...	0	0	0	0	0	0	1	0	0	0
11	ocon	0	10	1	0	0	0	0	0	2	...	2	4	0	6	0	4	0	0	1	0
12	perez	8	0	2	6	0	2	10	12	4	...	0	10	10	10	6	8	8	10	6	6
13	ricciardo	10	6	10	8	0	10	15	10	10	...	10	12	0	12	15	12	10	8	8	10
14	russell	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0
15	sainz	2	4	6	10	0	4	6	6	0	...	1	0	2	0	4	2	4	1	2	8
16	stroll	0	1	0	2	0	0	0	4	0	...	0	0	4	2	0	0	0	0	0	0
17	tsunoda	0	0	0	0	0	0	0	0	0	...	0	0	6	0	0	0	0	0	0	0
18	vettel	18	0	18	1	0	18	18	25	18	...	18	18	8	15	25	18	18	15	18	18
19	zhou	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Sort each column from highest to lowest Z-score and depending on there position give them respective championships points. Create a new column called "total" and get the sum of all the scores for a driver. Then sort the dataframe from highest to lowest scores. This is our predicted outcome for the 2022 Grand Prix.

```
champ_df['Total'] = champ_df.sum(axis=1)
champ_df.sort_values('Total',ascending=False)
```

	Driver	bahrain	jeddah	albert_park	imola	miami	catalunya	monaco	baku	villeneuve	...	spa	zandvoort	monza	marina_bay	suzuka	americas	rodriguez	interlagos	yas_marina	Total
4	hamilton	25	25	25	15	0	25	25	18	25	...	25	18	25	18	25	25	18	25	25	487
18	vettel	18	0	18	1	0	18	18	25	18	...	18	8	15	25	18	18	15	18	18	326
8	max_verstappen	12	18	12	25	0	12	12	8	12	...	8	25	4	12	10	15	25	15	15	295
2	bottas	15	15	15	4	0	15	8	15	15	...	15	15	18	10	15	12	12	12	12	286
13	ricciardo	10	6	10	8	0	10	15	10	10	...	12	0	12	15	12	10	8	8	10	205
12	perez	8	0	2	6	0	2	10	12	4	...	10	10	10	6	8	8	10	6	6	134
7	leclerc	1	12	4	18	0	6	0	2	6	...	6	0	8	2	0	2	6	4	2	111
15	sainz	2	4	6	10	0	4	6	6	0	...	0	2	0	4	2	4	1	2	8	72
1	alonso	0	0	8	0	0	8	4	1	8	...	2	0	1	8	1	6	0	10	1	72
11	ocon	0	10	1	0	0	0	0	0	2	...	4	0	6	0	4	0	0	1	0	30
0	albon	0	0	0	0	0	0	0	0	0	...	0	12	0	1	6	0	4	0	4	27
10	norris	6	2	0	12	0	0	1	0	0	...	0	0	0	0	0	1	0	0	0	27
3	gasly	4	8	0	0	0	1	2	0	0	...	1	0	0	0	0	0	2	0	0	27
16	stroll	0	1	0	2	0	0	0	4	0	...	0	4	2	0	0	0	0	0	0	13
17	tsunoda	0	0	0	0	0	0	0	0	0	...	0	6	0	0	0	0	0	0	0	6
5	kevin_magnussen	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	2
14	russell	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	1
9	mick_schumacher	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6	latifi	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
19	zhou	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

4 Conclusions

4.1 Performance Score

What we noticed about the way we gave performance scores was that at the end of a race, the order from high to low of performance score almost mirrored the outcome of the race. Those with the most points were typically in 1st, 2nd, 3rd, and those with the least points were 20th, 19th, 18th etc. Because of this, we used this relationship to predict the outcome of a race. Then based off the ordered outcome, we can give respective points to a driver and their predicted position.

4.2 Experience

Experience is one of the missing data groups. We knew at the beginning that drivers with more history were going to have higher performance scores just due to the fact that they have more years of data to get points. This is a issue because its not a true representation of the drivers. An old driver is not necessarily a good driver, but their performance score is inflated due to their experience. Our only solution to this was to replace a drivers performance score with a Z-score representation with the idea to normalize the range of score. This really did not affect the outcome, but made the scoring visually easier to distinguish. One idea we had to attempt to solve this issue was to build a model that took a driver and a circuit and used random lap times from their history to simulate their performance on a track to predict there position, but this was a late idea and still does not tackle the problem of team data.

4.3 Consistency Score and Team Score

Unfortunately, Consistency Score and Team Score did not factor into our final product. We wanted to solve some bias in the data by creating two other scoring methods that weigh on the performance score, but we were unable to fully implement these ideas.

Consistency score is given to a driver on a individual level. The idea is to compare a driver to themselves lap over lap. Drivers that put up consistent lap times would receive a high score. This would benefit good drivers who happen to be on a bad team.

Team score represents how good or bad a team is overall, and how this affects drivers who race for them. This would allow us to factor teams into the drivers score, which would paint a better prediction for this year. If a good driver is on a bad team, that will most definitely affect the outcome of the Grand Prix.

5 References

5.1 Formula One Ergast Data API

http://ergast.com/mrd/

5.2 Formula One Official Website

https://www.formula1.com/