Project_40000797

June 18, 2019

0.1 Project for COMP 499 - Data Analytics

0.2 Steven Zanga - 40000797

Introduction: My Data Analytics project will be focused around the sport of Formula 1. At the beginning of every F1 season, the F1 league decides on specific set of rules that would likely influence some sort of change to the cars. These new rules may have an effect on the engine or the aerodynamics of the car.

Questions: I will be examining the changes to the car in order to determine if these changes have a direct impact on faster laptimes the following year, along with increased number of accidents. I will also be using driver performance data to try and predict a drivers final position based on their position in qualifying.

0.3 Data Retrieval and Standard Descriptive Analysis

My Dataset that I found consists of 10 CSV files containing racing data from 1950 to 2017. This data includes:

1. Circuits: **Description about the location of each track that has been included in the Formula 1 circuit since 1950.**
2. Constructor Results: **Results based on Constructor (Team)**
3. Drivers: **Description about every driver that has every raced in the Formula 1 circuit since 1950**
4. Driver Standings **Driver Standings in each year since 1950**
5. Lap Times **Laptimes of every driver, from every race**
6. Pit Stops **Amount of pitstop and the time the pitstop took**
7. Qualifying **Qualifying position of each racer, from every race**
8. Races **Each race, at every location, winner, fastest lap, winning team**
9. Results **Results from each race**
10. Seasons **Results at the end of each season**

This Dataset was found at: https://www.kaggle.com/cjgdev/formula-1-race-data-19502017

As I mentioned previously, I want to see if there is a correlation between fastest lap times and changes made to the car. A historical list of all changes made to Formula 1 cars can be found here: https://en.wikipedia.org/wiki/History_of_Formula_One_regulations

0.4 Data Wrangling - Cleaning the data

The main dataset that I will be using is the Results.CSV, however currently it contains an abundance of NaN values so we need to clean it up to make it actually usable.

[15]:
```
import pandas as pd
results = pd.read_csv("./data/results.csv")
results.head()
```

[15]:
```
   resultId  raceId  driverId  constructorId  number  grid  position  \
0         1      18         1              1    22.0     1       1.0
1         2      18         2              2     3.0     5       2.0
2         3      18         3              3     7.0     7       3.0
3         4      18         4              4     5.0    11       4.0
4         5      18         5              1    23.0     3       5.0

  positionText  positionOrder  points  laps     time  milliseconds  \
0            1              1    10.0    58  34:50.6     5690616.0
1            2              2     8.0    58   5.478     5696094.0
2            3              3     6.0    58   8.163     5698779.0
3            4              4     5.0    58  17.181     5707797.0
4            5              5     4.0    58  18.014     5708630.0

   fastestLap  rank fastestLapTime fastestLapSpeed  statusId
0        39.0   2.0        01:27.5           218.3         1
1        41.0   3.0        01:27.7         217.586         1
2        41.0   5.0        01:28.1         216.719         1
3        58.0   7.0        01:28.6         215.464         1
4        43.0   1.0        01:27.4         218.385         1
```

[16]:
```
results['fastestLapTime']
```

[16]:
```
0        01:27.5
1        01:27.7
2        01:28.1
3        01:28.6
4        01:27.4
5        01:29.6
6        01:29.5
7        01:27.9
8        01:28.8
9        01:29.6
10       01:30.9
11       01:31.4
12       01:28.2
13       01:29.5
14       01:29.3
15       01:32.0
16           NaN
17           NaN
```

```
18           NaN
19           NaN
20           NaN
21       01:28.7
22       01:35.4
23       01:35.9
24       01:35.9
25       01:36.1
26       01:35.5
27       01:35.4
28       01:36.7
29       01:36.3
          ...
23747    01:13.6
23748    01:13.3
23749    01:13.7
23750    01:14.8
23751    01:13.5
23752    01:11.9
23753    01:14.7
23754        NaN
23755        NaN
23756        NaN
23757    01:40.7
23758    01:41.5
23759    01:40.8
23760    01:42.3
23761    01:42.0
23762    01:42.4
23763    01:42.7
23764    01:42.6
23765    01:43.4
23766    01:43.0
23767    01:42.4
23768    01:44.0
23769    01:43.9
23770    01:43.9
23771    01:43.9
23772    01:43.8
23773    01:43.6
23774    01:42.3
23775    01:43.4
23776    01:42.8
Name: fastestLapTime, Length: 23777, dtype: object
```

Notice the NaN values, there are a lot more where that came from because of the lack of recording devices in the 50s and 60s.

[17]:
```
results.describe()
```

[17]:
```
           resultId        raceId      driverId  constructorId        number  \
count  23777.000000  23777.000000  23777.000000   23777.000000  23771.000000
mean   11889.481053    487.203937    226.515961      46.281785     16.965462
std     6864.691322    269.904857    231.386102      56.174091     13.644798
min        1.000000      1.000000      1.000000       1.000000      0.000000
25%     5945.000000    273.000000     55.000000       6.000000      7.000000
50%    11889.000000    478.000000    154.000000      25.000000     15.000000
75%    17833.000000    718.000000    314.000000      57.000000     23.000000
max    23781.000000    988.000000    843.000000     210.000000    208.000000

                grid      position  positionOrder        points          laps  \
count  23777.000000  13227.000000   23777.000000  23777.000000  23777.000000
mean      11.270303      7.782264      13.081591      1.601403     45.270598
std        7.346436      4.745105       7.824711      3.665154     30.525404
min        0.000000      1.000000       1.000000      0.000000      0.000000
25%        5.000000      4.000000       7.000000      0.000000     20.000000
50%       11.000000      7.000000      13.000000      0.000000     52.000000
75%       17.000000     11.000000      19.000000      1.000000     66.000000
max       34.000000     33.000000      39.000000     50.000000    200.000000

       milliseconds   fastestLap         rank      statusId
count  6.003000e+03  5383.000000  5531.000000  23777.000000
mean   6.303313e+06    41.061676    10.598807     18.242293
std    1.721748e+06    17.156435     6.272457     26.380824
min    1.474899e+06     2.000000     0.000000      1.000000
25%    5.442948e+06    29.000000     5.000000      1.000000
50%    5.859428e+06    44.000000    11.000000     11.000000
75%    6.495440e+06    53.000000    16.000000     16.000000
max    1.509054e+07    78.000000    24.000000    136.000000
```

[18]:
```
results.count()
```

[18]:
```
resultId          23777
raceId            23777
driverId          23777
constructorId     23777
number            23771
grid              23777
position          13227
positionText      23777
positionOrder     23777
points            23777
laps              23777
time               6004
milliseconds       6003
fastestLap         5383
```

```
rank                5531
fastestLapTime      5383
fastestLapSpeed     5383
statusId           23777
dtype: int64
```

[19]:
```
# We can check how many null values we have by using the following:
results.isnull().sum()
```

[19]:
```
resultId               0
raceId                 0
driverId               0
constructorId          0
number                 6
grid                   0
position           10550
positionText           0
positionOrder          0
points                 0
laps                   0
time               17773
milliseconds       17774
fastestLap         18394
rank               18246
fastestLapTime     18394
fastestLapSpeed    18394
statusId               0
dtype: int64
```

[20]:
```
results.dropna(inplace=True) # Drop all data where there exists a NaN in the␣
 ↪row.
results.isnull().sum()
```

[20]:
```
resultId           0
raceId             0
driverId           0
constructorId      0
number             0
grid               0
position           0
positionText       0
positionOrder      0
points             0
laps               0
time               0
milliseconds       0
fastestLap         0
rank               0
fastestLapTime     0
```

```
fastestLapSpeed    0
statusId           0
dtype: int64
```

[21]:
```
results['fastestLapTime']
```

[21]:
```
0         01:27.5
1         01:27.7
2         01:28.1
3         01:28.6
4         01:27.4
22        01:35.4
23        01:35.9
24        01:35.9
25        01:36.1
26        01:35.5
27        01:35.4
28        01:36.7
29        01:36.3
30        01:36.2
31        01:35.7
32        01:37.0
44        01:33.6
45        01:33.7
46        01:33.8
47        01:33.6
48        01:33.2
49        01:34.2
50        01:34.3
51        01:34.1
52        01:34.8
53        01:35.2
54        01:34.9
66        01:21.7
67        01:21.8
68        01:22.0
           ...
23684     01:35.3
23685     01:35.3
23697     01:38.8
23698     01:38.8
23699     01:37.8
23700     01:38.1
23701     01:37.8
23702     01:40.5
23703     01:40.5
23717     01:18.9
23718     01:19.4
```

```
23719    01:20.1
23720    01:18.8
23737    01:12.5
23738    01:12.5
23739    01:12.5
23740    01:11.8
23741    01:11.0
23742    01:12.0
23743    01:13.5
23744    01:13.5
23745    01:13.1
23757    01:40.7
23758    01:41.5
23759    01:40.8
23760    01:42.3
23761    01:42.0
23762    01:42.4
23763    01:42.7
23764    01:42.6
Name: fastestLapTime, Length: 2604, dtype: object
```

[22]:
```
results.count()
```

[22]:
```
resultId           2604
raceId             2604
driverId           2604
constructorId      2604
number             2604
grid               2604
position           2604
positionText       2604
positionOrder      2604
points             2604
laps               2604
time               2604
milliseconds       2604
fastestLap         2604
rank               2604
fastestLapTime     2604
fastestLapSpeed    2604
statusId           2604
dtype: int64
```

Our Results are all cleaned and ready to be used!

[24]:
```
# Our Data Types
results.dtypes
```

[24]:
```
resultId             int64
raceId               int64
```

```
driverId              int64
constructorId         int64
number              float64
grid                  int64
position            float64
positionText         object
positionOrder         int64
points              float64
laps                  int64
time                 object
milliseconds        float64
fastestLap          float64
rank                float64
fastestLapTime       object
fastestLapSpeed      object
statusId              int64
dtype: object
```

0.5 Data Integration & Data Enrichment

Combining the Ids from each CSV file

So if you look below, you can see we have a bunch of labels such as: 'RaceId' and 'ConstructorId' ..etc. These IDs correspond to data linked on the other CSV files, so in order to prepare my data for machine learning, I need to merge all these files together and then delete the columns of the data I don't need!

My approach to integrating data

Now my screen is only 13 inches so it will be a bit difficult to manage my data in a table with more than 30 columns. So my approach will be to merge each table individually, then delete the redundant columns, and repeat until all files are merged together and we have all the data we need. Now in order to save myself from making mistakes, I saved every merge to a new variable, I will use each variable as a checkpoint so that if I screw up, I don't have to restart from the beginning. The Final Variable will be called **FinalData**.

[25]:
```
results.head()
```

[25]:

	resultId	raceId	driverId	constructorId	number	grid	position	positionText	positionOrder	points	laps	time	milliseconds
0	1	18	1	1	22.0	1	1.0	1	1	10.0	58	34:50.6	5690616.0
1	2	18	2	2	3.0	5	2.0	2	2	8.0	58	5.478	5696094.0
2	3	18	3	3	7.0	7	3.0	3	3	6.0	58	8.163	5698779.0
3	4	18	4	4	5.0	11	4.0	4	4	5.0	58	17.181	5707797.0
4	5	18	5	1	23.0	3	5.0	5	5	4.0	58	18.014	5708630.0

```
   fastestLap  rank fastestLapTime fastestLapSpeed  statusId
0        39.0   2.0        01:27.5           218.3         1
1        41.0   3.0        01:27.7         217.586         1
2        41.0   5.0        01:28.1         216.719         1
3        58.0   7.0        01:28.6         215.464         1
4        43.0   1.0        01:27.4         218.385         1
```

[27]:
```
# Reading the Drivers csv File
drivers = pd.read_csv("./data/drivers.csv", encoding = "ISO-8859-1")
drivers.head()
```

[27]:
```
   driverId   driverRef  number code  forename     surname         dob  \
0         1    hamilton    44.0  HAM     Lewis    Hamilton  07/01/1985
1         2    heidfeld     NaN  HEI      Nick    Heidfeld  10/05/1977
2         3     rosberg     6.0  ROS      Nico     Rosberg  27/06/1985
3         4      alonso    14.0  ALO  Fernando      Alonso  29/07/1981
4         5  kovalainen     NaN  KOV    Heikki  Kovalainen  19/10/1981

  nationality                                          url
0     British     http://en.wikipedia.org/wiki/Lewis_Hamilton
1      German      http://en.wikipedia.org/wiki/Nick_Heidfeld
2      German       http://en.wikipedia.org/wiki/Nico_Rosberg
3     Spanish    http://en.wikipedia.org/wiki/Fernando_Alonso
4     Finnish  http://en.wikipedia.org/wiki/Heikki_Kovalainen
```

[29]:
```
#Merging the Drivers CSV file and saving it to merge
merge = pd.merge(results, drivers, how="inner", on="driverId")
merge
```

[29]:
```
      resultId  raceId  driverId  constructorId  number_x  grid  position  \
0            1      18         1              1      22.0     1       1.0
1           27      19         1              1      22.0     9       5.0
2           69      21         1              1      22.0     5       3.0
3           90      22         1              1      22.0     3       2.0
4          109      23         1              1      22.0     3       1.0
5          158      25         1              1      22.0    13      10.0
6          169      26         1              1      22.0     4       1.0
7          189      27         1              1      22.0     1       1.0
8          213      28         1              1      22.0     1       5.0
9          230      29         1              1      22.0     2       2.0
10         251      30         1              1      22.0     1       3.0
11         275      31         1              1      22.0    15       7.0
12         291      32         1              1      22.0     2       3.0
13         320      33         1              1      22.0     1      12.0
14         329      34         1              1      22.0     1       1.0
15         353      35         1              1      22.0     4       5.0
16         371      36         1              1       2.0     4       3.0
17         392      37         1              1       2.0     4       2.0
```

18	414	38	1	1	2.0	2	2.0
19	436	39	1	1	2.0	4	2.0
20	458	40	1	1	2.0	2	2.0
21	479	41	1	1	2.0	1	1.0
22	501	42	1	1	2.0	1	1.0
23	525	43	1	1	2.0	2	3.0
24	547	44	1	1	2.0	1	3.0
25	589	46	1	1	2.0	1	1.0
26	615	47	1	1	2.0	2	5.0
27	634	48	1	1	2.0	2	2.0
28	658	49	1	1	2.0	4	4.0
29	677	50	1	1	2.0	1	1.0
...	...	...	...	...	...	...	...
2574	23645	982	832	5	55.0	10	4.0
2575	23708	985	832	4	55.0	7	7.0
2576	22848	942	834	209	53.0	17	12.0
2577	22927	948	835	4	30.0	13	11.0
2578	23195	960	835	4	30.0	13	15.0
2579	23256	963	835	4	30.0	19	10.0
2580	23489	974	835	4	30.0	16	11.0
2581	23614	980	835	4	30.0	14	13.0
2582	23647	982	835	4	30.0	11	6.0
2583	23346	967	839	209	31.0	22	12.0
2584	23428	971	839	10	31.0	14	10.0
2585	23445	972	839	10	31.0	10	7.0
2586	23490	974	839	10	31.0	15	12.0
2587	23504	975	839	10	31.0	9	6.0
2588	23524	976	839	10	31.0	7	6.0
2589	23610	980	839	10	31.0	9	9.0
2590	23627	981	839	10	31.0	3	6.0
2591	23651	982	839	10	31.0	14	10.0
2592	23687	984	839	10	31.0	5	6.0
2593	23707	985	839	10	31.0	6	6.0
2594	23769	988	839	10	31.0	9	8.0
2595	23349	967	836	209	94.0	19	15.0
2596	23528	976	836	15	94.0	14	10.0
2597	23521	976	840	3	18.0	8	3.0
2598	23612	980	840	3	18.0	15	11.0
2599	23628	981	840	3	18.0	2	7.0
2600	23649	982	840	3	18.0	18	8.0
2601	23530	976	838	1	2.0	18	12.0
2602	23615	980	838	1	2.0	20	14.0
2603	23648	982	838	1	2.0	9	7.0

	positionText	positionOrder	points	...	fastestLapSpeed	statusId	\
0	1	1	10.0	...	218.3	1	
1	5	5	4.0	...	209.033	1	

2	3	3	6.0	...	204.323	1
3	2	2	8.0	...	222.085	1
4	1	1	10.0	...	153.152	1
5	10	10	0.0	...	205.022	1
6	1	1	10.0	...	199.398	1
7	1	1	10.0	...	216.552	1
8	5	5	4.0	...	193.533	1
9	2	2	8.0	...	197.285	1
10	3	3	6.0	...	233.175	1
11	7	7	2.0	...	232.44	1
12	3	3	6.0	...	171.969	1
13	12	12	0.0	...	206.47	1
14	1	1	10.0	...	203.722	1
15	5	5	4.0	...	209.177	1
16	3	3	6.0	...	221.083	1
17	2	2	8.0	...	206.355	1
18	2	2	8.0	...	206.674	1
19	2	2	8.0	...	202.205	1
20	2	2	8.0	...	159.528	1
21	1	1	10.0	...	205.239	1
22	1	1	10.0	...	206.101	1
23	3	3	6.0	...	207.34	1
24	3	3	6.0	...	226.6	1
25	1	1	10.0	...	196.724	1
26	5	5	4.0	...	218.464	1
27	2	2	8.0	...	251.456	1
28	4	4	5.0	...	233.002	1
29	1	1	10.0	...	186.259	1
...	...	...	...	...	...	...
2574	4	4	12.0	...	171.151	1
2575	7	7	6.0	...	197.555	1
2576	12	12	0.0	...	183.472	1
2577	11	11	0.0	...	205.376	1
2578	15	15	0.0	...	222.641	1
2579	10	10	1.0	...	200.853	1
2580	11	11	0.0	...	156.801	1
2581	13	13	0.0	...	230.725	1
2582	6	6	8.0	...	170.855	1
2583	12	12	0.0	...	176.686	1
2584	10	10	1.0	...	204.7	1
2585	7	7	6.0	...	213.203	1
2586	12	12	0.0	...	157.072	1
2587	6	6	8.0	...	205.904	1
2588	6	6	8.0	...	204.581	1
2589	9	9	2.0	...	229.804	1
2590	6	6	8.0	...	243.482	1
2591	10	10	1.0	...	169.339	1

2592	6	6	8.0	...	220.419	1
2593	6	6	8.0	...	197.482	1
2594	8	8	4.0	...	01:42.6	1
2595	15	15	0.0	...	176.439	1
2596	10	10	1.0	...	201.743	1
2597	3	3	15.0	...	205.605	1
2598	11	11	0.0	...	228.095	1
2599	7	7	6.0	...	243.559	1
2600	8	8	4.0	...	169.599	1
2601	12	12	0.0	...	202.636	1
2602	14	14	0.0	...	229.415	1
2603	7	7	6.0	...	170.855	1

	driverRef	number_y	code	forename	surname	dob	\
0	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
1	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
2	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
3	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
4	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
5	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
6	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
7	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
8	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
9	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
10	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
11	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
12	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
13	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
14	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
15	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
16	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
17	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
18	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
19	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
20	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
21	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
22	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
23	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
24	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
25	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
26	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
27	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
28	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
29	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	
...	...	...	...	...	...	...	
2574	sainz	55.0	SAI	Carlos	Sainz	01/09/1994	
2575	sainz	55.0	SAI	Carlos	Sainz	01/09/1994	

2576	rossi	53.0	RSS	Alexander	Rossi	25/09/1991
2577	jolyon_palmer	30.0	PAL	Jolyon	Palmer	20/01/1991
2578	jolyon_palmer	30.0	PAL	Jolyon	Palmer	20/01/1991
2579	jolyon_palmer	30.0	PAL	Jolyon	Palmer	20/01/1991
2580	jolyon_palmer	30.0	PAL	Jolyon	Palmer	20/01/1991
2581	jolyon_palmer	30.0	PAL	Jolyon	Palmer	20/01/1991
2582	jolyon_palmer	30.0	PAL	Jolyon	Palmer	20/01/1991
2583	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2584	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2585	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2586	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2587	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2588	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2589	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2590	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2591	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2592	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2593	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2594	ocon	31.0	OCO	Esteban	Ocon	17/09/1996
2595	wehrlein	94.0	WEH	Pascal	Wehrlein	18/10/1994
2596	wehrlein	94.0	WEH	Pascal	Wehrlein	18/10/1994
2597	stroll	18.0	STR	Lance	Stroll	29/10/1998
2598	stroll	18.0	STR	Lance	Stroll	29/10/1998
2599	stroll	18.0	STR	Lance	Stroll	29/10/1998
2600	stroll	18.0	STR	Lance	Stroll	29/10/1998
2601	vandoorne	2.0	VAN	Stoffel	Vandoorne	26/03/1992
2602	vandoorne	2.0	VAN	Stoffel	Vandoorne	26/03/1992
2603	vandoorne	2.0	VAN	Stoffel	Vandoorne	26/03/1992

	nationality	url
0	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
1	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
2	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
3	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
4	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
5	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
6	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
7	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
8	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
9	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
10	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
11	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
12	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
13	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
14	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
15	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
16	British	http://en.wikipedia.org/wiki/Lewis_Hamilton

```
17       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
18       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
19       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
20       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
21       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
22       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
23       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
24       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
25       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
26       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
27       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
28       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
29       British        http://en.wikipedia.org/wiki/Lewis_Hamilton
...          ...                                                ...
2574     Spanish      http://en.wikipedia.org/wiki/Carlos_Sainz_Jr.
2575     Spanish      http://en.wikipedia.org/wiki/Carlos_Sainz_Jr.
2576    American  http://en.wikipedia.org/wiki/Alexander_Rossi_%...
2577     British         http://en.wikipedia.org/wiki/Jolyon_Palmer
2578     British         http://en.wikipedia.org/wiki/Jolyon_Palmer
2579     British         http://en.wikipedia.org/wiki/Jolyon_Palmer
2580     British         http://en.wikipedia.org/wiki/Jolyon_Palmer
2581     British         http://en.wikipedia.org/wiki/Jolyon_Palmer
2582     British         http://en.wikipedia.org/wiki/Jolyon_Palmer
2583      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2584      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2585      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2586      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2587      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2588      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2589      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2590      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2591      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2592      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2593      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2594      French          http://en.wikipedia.org/wiki/Esteban_Ocon
2595      German       http://en.wikipedia.org/wiki/Pascal_Wehrlein
2596      German       http://en.wikipedia.org/wiki/Pascal_Wehrlein
2597    Canadian          http://en.wikipedia.org/wiki/Lance_Stroll
2598    Canadian          http://en.wikipedia.org/wiki/Lance_Stroll
2599    Canadian          http://en.wikipedia.org/wiki/Lance_Stroll
2600    Canadian          http://en.wikipedia.org/wiki/Lance_Stroll
2601     Belgian     http://en.wikipedia.org/wiki/Stoffel_Vandoorne
2602     Belgian     http://en.wikipedia.org/wiki/Stoffel_Vandoorne
2603     Belgian     http://en.wikipedia.org/wiki/Stoffel_Vandoorne

[2604 rows x 26 columns]
```

[31]:
```
# Let's drop columns we don't need to clean it up a little bit. This is me␣
 ↪testing dropping 'points' so that I dont
# screw up
merge = merge.drop('points', axis = 1)
merge.head()
```

[31]:
```
   resultId  raceId  driverId  constructorId  number_x  grid  position  \
0         1      18         1              1      22.0     1       1.0
1        27      19         1              1      22.0     9       5.0
2        69      21         1              1      22.0     5       3.0
3        90      22         1              1      22.0     3       2.0
4       109      23         1              1      22.0     3       1.0

  positionText  positionOrder  laps  ... fastestLapSpeed  statusId  driverRef  \
0            1              1    58  ...           218.3         1   hamilton
1            5              5    56  ...         209.033         1   hamilton
2            3              3    66  ...         204.323         1   hamilton
3            2              2    58  ...         222.085         1   hamilton
4            1              1    76  ...         153.152         1   hamilton

   number_y code forename   surname         dob  nationality  \
0      44.0  HAM    Lewis  Hamilton  07/01/1985      British
1      44.0  HAM    Lewis  Hamilton  07/01/1985      British
2      44.0  HAM    Lewis  Hamilton  07/01/1985      British
3      44.0  HAM    Lewis  Hamilton  07/01/1985      British
4      44.0  HAM    Lewis  Hamilton  07/01/1985      British

                                        url
0  http://en.wikipedia.org/wiki/Lewis_Hamilton
1  http://en.wikipedia.org/wiki/Lewis_Hamilton
2  http://en.wikipedia.org/wiki/Lewis_Hamilton
3  http://en.wikipedia.org/wiki/Lewis_Hamilton
4  http://en.wikipedia.org/wiki/Lewis_Hamilton

[5 rows x 25 columns]
```

[32]:
```
# Dropping columns: 'position, laps, url, nationality, dob, forename, code,␣
 ↪number_y'
merge = merge.drop(['position', 'laps', 'url', 'nationality', 'dob',␣
 ↪'forename', 'code', 'number_y'], axis = 1)
merge.head()
```

[32]:
```
   resultId  raceId  driverId  constructorId  number_x  grid positionText  \
0         1      18         1              1      22.0     1            1
1        27      19         1              1      22.0     9            5
2        69      21         1              1      22.0     5            3
3        90      22         1              1      22.0     3            2
4       109      23         1              1      22.0     3            1
```

```
   positionOrder     time  milliseconds  fastestLap  rank fastestLapTime  \
0              1  34:50.6     5690616.0        39.0   2.0        01:27.5
1              5   46.548     5525103.0        53.0   3.0        01:35.5
2              3    4.187     5903238.0        20.0   3.0        01:22.0
3              2    3.779     5213230.0        31.0   2.0        01:26.5
4              1  00:42.7     7242742.0        71.0   6.0        01:18.5

  fastestLapSpeed  statusId driverRef   surname
0           218.3         1  hamilton  Hamilton
1         209.033         1  hamilton  Hamilton
2         204.323         1  hamilton  Hamilton
3         222.085         1  hamilton  Hamilton
4         153.152         1  hamilton  Hamilton
```

[33]:
```
# reading the status csv file
status = pd.read_csv("./data/status.csv", encoding = "ISO-8859-1")
status.head()
```

[33]:
```
   statusId        status
0         1      Finished
1         2  Disqualified
2         3      Accident
3         4     Collision
4         5        Engine
```

[36]:
```
# merging merge and status, saving it to d1
d1 = pd.merge(merge, status, how="inner", on='statusId')
d1.head()
```

[36]:
```
   resultId  raceId  driverId  constructorId  number_x  grid positionText  \
0         1      18         1              1      22.0     1            1
1        27      19         1              1      22.0     9            5
2        69      21         1              1      22.0     5            3
3        90      22         1              1      22.0     3            2
4       109      23         1              1      22.0     3            1

   positionOrder     time  milliseconds  fastestLap  rank fastestLapTime  \
0              1  34:50.6     5690616.0        39.0   2.0        01:27.5
1              5   46.548     5525103.0        53.0   3.0        01:35.5
2              3    4.187     5903238.0        20.0   3.0        01:22.0
3              2    3.779     5213230.0        31.0   2.0        01:26.5
4              1  00:42.7     7242742.0        71.0   6.0        01:18.5

  fastestLapSpeed  statusId driverRef   surname    status
0           218.3         1  hamilton  Hamilton  Finished
1         209.033         1  hamilton  Hamilton  Finished
2         204.323         1  hamilton  Hamilton  Finished
3         222.085         1  hamilton  Hamilton  Finished
```

```
4         153.152         1  hamilton  Hamilton  Finished
```

[38]:
```
# droping columns: 'driverId, statusId, driverRef' and saving it to data.
data = d1.drop(['driverId', 'statusId', 'driverRef'], axis = 1)
data.head()
```

[38]:
```
   resultId  raceId  constructorId  number_x  grid positionText  \
0         1      18              1      22.0     1            1
1        27      19              1      22.0     9            5
2        69      21              1      22.0     5            3
3        90      22              1      22.0     3            2
4       109      23              1      22.0     3            1

   positionOrder     time  milliseconds  fastestLap  rank fastestLapTime  \
0              1  34:50.6     5690616.0        39.0   2.0        01:27.5
1              5   46.548     5525103.0        53.0   3.0        01:35.5
2              3    4.187     5903238.0        20.0   3.0        01:22.0
3              2    3.779     5213230.0        31.0   2.0        01:26.5
4              1  00:42.7     7242742.0        71.0   6.0        01:18.5

  fastestLapSpeed   surname    status
0           218.3  Hamilton  Finished
1         209.033  Hamilton  Finished
2         204.323  Hamilton  Finished
3         222.085  Hamilton  Finished
4         153.152  Hamilton  Finished
```

[40]:
```
# reading races csv, droping columns: 'round, circuitId, date, time, url'
races = pd.read_csv("./data/races.csv", encoding = "ISO-8859-1")
races = races.drop(['round', 'circuitId', 'date', 'time', 'url'], axis = 1)
races.head()
```

[40]:
```
   raceId  year                  name
0       1  2009  Australian Grand Prix
1       2  2009   Malaysian Grand Prix
2       3  2009     Chinese Grand Prix
3       4  2009     Bahrain Grand Prix
4       5  2009     Spanish Grand Prix
```

[41]:
```
# merging data and races, saving it to cleanedData
cleanedData = pd.merge(data, races, how='inner', on='raceId')
cleanedData.head()
```

[41]:
```
   resultId  raceId  constructorId  number_x  grid positionText  \
0         1      18              1      22.0     1            1
1         2      18              2       3.0     5            2
2         3      18              3       7.0     7            3
3         4      18              4       5.0    11            4
4         5      18              1      23.0     3            5
```

```
   positionOrder     time  milliseconds  fastestLap  rank fastestLapTime  \
0              1  34:50.6     5690616.0        39.0   2.0        01:27.5
1              2    5.478     5696094.0        41.0   3.0        01:27.7
2              3    8.163     5698779.0        41.0   5.0        01:28.1
3              4   17.181     5707797.0        58.0   7.0        01:28.6
4              5   18.014     5708630.0        43.0   1.0        01:27.4

  fastestLapSpeed     surname    status  year                  name
0           218.3    Hamilton  Finished  2008  Australian Grand Prix
1         217.586    Heidfeld  Finished  2008  Australian Grand Prix
2         216.719     Rosberg  Finished  2008  Australian Grand Prix
3         215.464      Alonso  Finished  2008  Australian Grand Prix
4         218.385  Kovalainen  Finished  2008  Australian Grand Prix
```

[47]:
```
# reading constructor csv, droping columns: constructorRef, nationality, url,␣
 ↪saving it to d2.
constructors = pd.read_csv("./data/constructors.csv", encoding = "ISO-8859-1")
constructors = constructors.drop(['constructorRef', 'nationality', 'url',␣
 ↪'Unnamed: 5'], axis = 1)
d2 = pd.merge(cleanedData, constructors, how='inner', on = 'constructorId')
d2.head()
```

[47]:
```
   resultId  raceId  constructorId  number_x  grid positionText  \
0         1      18              1      22.0     1            1
1         5      18              1      23.0     3            5
2        27      19              1      22.0     9            5
3        25      19              1      23.0     8            3
4        69      21              1      22.0     5            3

   positionOrder     time  milliseconds  fastestLap  rank fastestLapTime  \
0              1  34:50.6     5690616.0        39.0   2.0        01:27.5
1              5   18.014     5708630.0        43.0   1.0        01:27.4
2              5   46.548     5525103.0        53.0   3.0        01:35.5
3              3    38.45     5517005.0        19.0   7.0        01:35.9
4              3    4.187     5903238.0        20.0   3.0        01:22.0

  fastestLapSpeed     surname    status  year                name_x   name_y
0           218.3    Hamilton  Finished  2008  Australian Grand Prix  McLaren
1         218.385  Kovalainen  Finished  2008  Australian Grand Prix  McLaren
2         209.033    Hamilton  Finished  2008   Malaysian Grand Prix  McLaren
3         208.031  Kovalainen  Finished  2008   Malaysian Grand Prix  McLaren
4         204.323    Hamilton  Finished  2008     Spanish Grand Prix  McLaren
```

[51]:
```
# Dropping some Data and we are done! Saved to FinalData.
FinalData = d2.drop(['constructorId','raceId','number_x', 'positionText',␣
 ↪'positionText', 'milliseconds', 'time', 'rank'], axis = 1)
FinalData # WE ARE DONE CLEANING, This data is now ready for machine learning.
```

[51]:

	resultId	grid	positionOrder	fastestLap	fastestLapTime	\
0	1	1	1	39.0	01:27.5	
1	5	3	5	43.0	01:27.4	
2	27	9	5	53.0	01:35.5	
3	25	8	3	19.0	01:35.9	
4	69	5	3	20.0	01:22.0	
5	90	3	2	31.0	01:26.5	
6	109	3	1	71.0	01:18.5	
7	116	4	8	74.0	01:17.3	
8	158	13	10	40.0	01:17.5	
9	152	10	4	46.0	01:17.1	
10	169	4	1	16.0	01:32.8	
11	189	1	1	17.0	01:16.0	
12	193	3	5	63.0	01:16.5	
13	213	1	5	15.0	01:21.5	
14	209	2	1	19.0	01:21.8	
15	230	2	2	16.0	01:38.9	
16	232	5	4	19.0	01:39.1	
17	251	1	3	20.0	01:48.1	
18	275	15	7	52.0	01:29.7	
19	270	2	2	53.0	01:30.3	
20	291	2	3	14.0	01:46.1	
21	298	5	10	14.0	01:47.3	
22	320	1	12	65.0	01:19.6	
23	329	1	1	13.0	01:36.3	
24	353	4	5	31.0	01:14.2	
25	355	5	7	36.0	01:14.2	
26	371	4	3	20.0	01:26.4	
27	370	2	2	20.0	01:26.3	
28	392	4	2	22.0	01:36.7	
29	391	2	1	42.0	01:36.9	
...	...	...	...	...	...	
2574	1557	6	3	24.0	01:31.0	
2575	1559	5	5	55.0	01:31.1	
2576	1576	1	2	28.0	01:21.2	
2577	1599	3	5	47.0	01:17.7	
2578	1637	5	3	13.0	01:30.5	
2579	1699	4	5	50.0	01:16.0	
2580	1718	3	4	10.0	01:19.5	
2581	1725	8	11	12.0	01:20.8	
2582	1736	13	2	11.0	01:14.1	
2583	1742	8	8	29.0	01:14.6	
2584	1759	4	5	47.0	01:20.4	
2585	1816	3	2	33.0	01:32.9	
2586	1820	18	6	36.0	01:33.5	
2587	1837	5	3	33.0	01:33.8	
2588	1838	4	4	28.0	01:33.7	

2589	1860	6	6	51.0	01:11.9	
2590	1616	2	2	40.0	01:15.2	
2591	1677	3	3	47.0	01:10.7	
2592	1797	6	3	13.0	01:22.7	
2593	1798	5	4	32.0	01:22.7	
2594	1657	2	3	68.0	01:14.2	
2595	1505	19	11	13.0	01:36.6	
2596	1641	14	7	37.0	01:31.9	
2597	1703	12	9	53.0	01:16.0	
2598	1722	9	8	11.0	01:20.8	
2599	1740	11	6	65.0	01:14.9	
2600	1744	12	10	33.0	01:15.0	
2601	1780	13	6	11.0	01:47.5	
2602	1803	12	9	53.0	01:23.1	
2603	21656	22	17	47.0	01:48.6	

	fastestLapSpeed	surname	status	year	name_x	\
0	218.3	Hamilton	Finished	2008	Australian Grand Prix	
1	218.385	Kovalainen	Finished	2008	Australian Grand Prix	
2	209.033	Hamilton	Finished	2008	Malaysian Grand Prix	
3	208.031	Kovalainen	Finished	2008	Malaysian Grand Prix	
4	204.323	Hamilton	Finished	2008	Spanish Grand Prix	
5	222.085	Hamilton	Finished	2008	Turkish Grand Prix	
6	153.152	Hamilton	Finished	2008	Monaco Grand Prix	
7	155.586	Kovalainen	Finished	2008	Monaco Grand Prix	
8	205.022	Hamilton	Finished	2008	French Grand Prix	
9	205.87	Kovalainen	Finished	2008	French Grand Prix	
10	199.398	Hamilton	Finished	2008	British Grand Prix	
11	216.552	Hamilton	Finished	2008	German Grand Prix	
12	215.261	Kovalainen	Finished	2008	German Grand Prix	
13	193.533	Hamilton	Finished	2008	Hungarian Grand Prix	
14	192.917	Kovalainen	Finished	2008	Hungarian Grand Prix	
15	197.285	Hamilton	Finished	2008	European Grand Prix	
16	196.831	Kovalainen	Finished	2008	European Grand Prix	
17	233.175	Hamilton	Finished	2008	Belgian Grand Prix	
18	232.44	Hamilton	Finished	2008	Italian Grand Prix	
19	230.95	Kovalainen	Finished	2008	Italian Grand Prix	
20	171.969	Hamilton	Finished	2008	Singapore Grand Prix	
21	169.943	Kovalainen	Finished	2008	Singapore Grand Prix	
22	206.47	Hamilton	Finished	2008	Japanese Grand Prix	
23	203.722	Hamilton	Finished	2008	Chinese Grand Prix	
24	209.177	Hamilton	Finished	2008	Brazilian Grand Prix	
25	209.042	Kovalainen	Finished	2008	Brazilian Grand Prix	
26	221.083	Hamilton	Finished	2007	Australian Grand Prix	
27	221.178	Alonso	Finished	2007	Australian Grand Prix	
28	206.355	Hamilton	Finished	2007	Malaysian Grand Prix	
29	206.014	Alonso	Finished	2007	Malaysian Grand Prix	

...	...	...	...	...	...
2574	214.393	Button	Finished	2004	Bahrain Grand Prix
2575	214.061	Sato	Finished	2004	Bahrain Grand Prix
2576	218.701	Button	Finished	2004	San Marino Grand Prix
2577	214.439	Sato	Finished	2004	Spanish Grand Prix
2578	204.879	Button	Finished	2004	European Grand Prix
2579	209.021	Button	Finished	2004	French Grand Prix
2580	232.835	Button	Finished	2004	British Grand Prix
2581	229.082	Sato	Finished	2004	British Grand Prix
2582	222.167	Button	Finished	2004	German Grand Prix
2583	220.773	Sato	Finished	2004	German Grand Prix
2584	196.103	Button	Finished	2004	Hungarian Grand Prix
2585	211.154	Button	Finished	2004	Chinese Grand Prix
2586	209.804	Sato	Finished	2004	Chinese Grand Prix
2587	222.824	Button	Finished	2004	Japanese Grand Prix
2588	223.007	Sato	Finished	2004	Japanese Grand Prix
2589	215.626	Sato	Finished	2004	Brazilian Grand Prix
2590	159.851	Button	Finished	2004	Monaco Grand Prix
2591	213.372	Sato	Finished	2004	United States Grand Prix
2592	252.262	Button	Finished	2004	Italian Grand Prix
2593	252.296	Sato	Finished	2004	Italian Grand Prix
2594	211.453	Button	Finished	2004	Canadian Grand Prix
2595	203.22	Monteiro	Finished	2005	Chinese Grand Prix
2596	201.678	Webber	Finished	2004	European Grand Prix
2597	209.063	Webber	Finished	2004	French Grand Prix
2598	229.145	Webber	Finished	2004	British Grand Prix
2599	219.895	Webber	Finished	2004	German Grand Prix
2600	219.42	Klien	Finished	2004	German Grand Prix
2601	233.595	Klien	Finished	2004	Belgian Grand Prix
2602	250.99	Webber	Finished	2004	Italian Grand Prix
2603	184.078	de la Rosa	Finished	2012	Abu Dhabi Grand Prix

	name_y
0	McLaren
1	McLaren
2	McLaren
3	McLaren
4	McLaren
5	McLaren
6	McLaren
7	McLaren
8	McLaren
9	McLaren
10	McLaren
11	McLaren
12	McLaren
13	McLaren

```
14    McLaren
15    McLaren
16    McLaren
17    McLaren
18    McLaren
19    McLaren
20    McLaren
21    McLaren
22    McLaren
23    McLaren
24    McLaren
25    McLaren
26    McLaren
27    McLaren
28    McLaren
29    McLaren
...       ...
2574      BAR
2575      BAR
2576      BAR
2577      BAR
2578      BAR
2579      BAR
2580      BAR
2581      BAR
2582      BAR
2583      BAR
2584      BAR
2585      BAR
2586      BAR
2587      BAR
2588      BAR
2589      BAR
2590      BAR
2591      BAR
2592      BAR
2593      BAR
2594      BAR
2595   Jordan
2596   Jaguar
2597   Jaguar
2598   Jaguar
2599   Jaguar
2600   Jaguar
2601   Jaguar
2602   Jaguar
2603      HRT
```

```
[2604 rows x 11 columns]
```

[54]:
```
FinalData.to_csv('FinalData.csv', sep=',') # Saving the DF incase something␣
↪happens.
```

[55]:
```
# No Nulls Found, so all my data is cleaned.
FinalData.isnull().sum()
```

[55]:
```
resultId           0
grid               0
positionOrder      0
fastestLap         0
fastestLapTime     0
fastestLapSpeed    0
surname            0
status             0
year               0
name_x             0
name_y             0
dtype: int64
```

[3]:
```
import pandas as pd
FinalData = pd.read_csv('./FinalData.csv')
FinalData.sample(20)
```

[3]:

	Unnamed: 0	resultId	grid	positionOrder	fastestLap	fastestLapTime	\
946	946	21256	8	1	53.0	01:41.7	
1198	1198	1424	6	10	5.0	01:23.6	
1273	1273	354	10	6	32.0	01:14.1	
1843	1843	22269	8	8	56.0	01:19.4	
2187	2187	22501	2	2	62.0	01:13.6	
2104	2104	21068	8	5	46.0	01:27.4	
102	102	20854	6	6	47.0	01:31.2	
2542	2542	23346	22	12	47.0	01:27.8	
1119	1119	722	1	2	71.0	01:12.6	
2480	2480	21964	11	11	45.0	01:26.9	
2067	2067	7795	6	2	46.0	01:24.9	
950	950	21310	9	7	44.0	01:38.2	
1754	1754	21296	17	17	40.0	01:41.3	
1756	1756	21317	17	14	43.0	01:37.1	
720	720	22928	14	12	45.0	01:32.5	
1975	1975	23276	9	8	39.0	01:37.4	
854	854	315	5	7	55.0	01:18.4	
432	432	7737	5	4	65.0	01:22.5	
1631	1631	21554	7	11	53.0	01:53.7	
1607	1607	7715	4	2	42.0	01:34.1	

	fastestLapSpeed	surname	status	year	name_x	\

946	196.25	Alonso	Finished	2012	Malaysian Grand Prix
1198	249.507	Schumacher	Finished	2005	Italian Grand Prix
1273	209.465	Glock	Finished	2008	Brazilian Grand Prix
1843	197.73	Vergne	Finished	2014	Canadian Grand Prix
2187	210.895	Hamilton	Finished	2014	Brazilian Grand Prix
2104	238.607	Schumacher	Finished	2011	Italian Grand Prix
102	210.786	Button	Finished	2011	Turkish Grand Prix
2542	176.686	Ocon	Finished	2016	Brazilian Grand Prix
1119	213.716	Massa	Finished	2007	Brazilian Grand Prix
2480	239.853	RÌ_ikkÌęnen	Finished	2013	Italian Grand Prix
2067	245.538	Button	Finished	2009	Italian Grand Prix
950	198.397	Alonso	Finished	2012	Bahrain Grand Prix
1754	193.811	Ricciardo	Finished	2012	Chinese Grand Prix
1756	200.737	Vergne	Finished	2012	Bahrain Grand Prix
720	206.494	Magnussen	Finished	2016	Australian Grand Prix
1975	214.74	HÌ_lkenberg	Finished	2016	Japanese Grand Prix
854	209.456	Massa	Finished	2008	Japanese Grand Prix
432	191.245	Rosberg	Finished	2009	Hungarian Grand Prix
1631	160.585	Webber	Finished	2012	Singapore Grand Prix
1607	196.97	Vettel	Finished	2009	German Grand Prix

	name_y
946	Ferrari
1198	Ferrari
1273	Toyota
1843	Toro Rosso
2187	Mercedes
2104	Mercedes
102	McLaren
2542	Manor Marussia
1119	Ferrari
2480	Lotus F1
2067	Brawn
950	Ferrari
1754	Toro Rosso
1756	Toro Rosso
720	Renault
1975	Force India
854	Ferrari
432	Williams
1631	Red Bull
1607	Red Bull

0.6 Question 1: What is the likelyness that a driver will finish in a desired position if they finish 1st in qualifying?

Podium positions

[61]:
```
# 1 grid, 1 Position = 140 / 230 = 60.86% chance of finishing 1
(FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 1)].
 ↪shape[0]/230) * 100
```

[61]: 60.86956521739131

[62]:
```
# 1 grid, 2 Position = 45 / 230 = 19.56% chance of finishing 2
(FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 2)].
 ↪shape[0]/230) * 100
```

[62]: 19.565217391304348

[63]:
```
# 1 grid, 3 Position = 23 / 230 = 10% chance of finishing 3
FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 3)].
 ↪shape[0]/230*100
```

[63]: 10.0

Non Podiums - In points

[64]:
```
# 1 grid, 4 Position = 8 / 230 = 3.47% chance of finishing 4
FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 4)].
 ↪shape[0]/230*100
```

[64]: 3.4782608695652173

[65]:
```
# 1 grid, 5 Position = 4 / 230 = 1.74% chance of finishing 5
FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 5)].
 ↪shape[0]/230*100
```

[65]: 1.7391304347826086

[66]:
```
# 1 grid, 6 Position = 3 / 230 = 1.30% chance of finishing 6
FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 6)].
 ↪shape[0]/230*100
```

[66]: 1.3043478260869565

[67]:
```
# 1 grid, 7 Position = 1 / 230 = .43% chance of finishing 7
FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 7)].
 ↪shape[0]/230*100
```

[67]: 0.43478260869565216

[68]:
```
# 1 grid, 8 Position = 2 / 230 = .86% chance of finishing 8
FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 8)].
 ↪shape[0]/230*100
```

[68]: 0.8695652173913043

[69]:
```
# 1 grid, 9 Position = 2 / 230 = .86% chance of finishing 9
FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 9)].
 ↪shape[0]/230*100
```

[69]: 0.8695652173913043

Given a first place qualification:

99.08% chance that a driver will finish within points range [1, 10]
< 1% chance that a driver will finish out of points range
90% chance of finishing on the podium

[59]:
```
FinalData[(FinalData['grid'] == 1)].shape[0]
```

[59]: 230

[97]:
```
# Simple automation for finding data for my bar chart
# First Place on Grid:
for i in range(1,15):
    print(FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] ==␣
 ↪i)].shape[0])
```

```
140
45
23
8
4
3
1
2
2
0
0
1
1
0
```

[96]:
```
# Simple automation for finding data for my bar chart
# Second Place on Grid:
for i in range(1,15):
    print(FinalData[(FinalData['grid'] == 2) & (FinalData['positionOrder'] ==␣
 ↪i)].shape[0])
```

```
65
62
38
19
15
3
7
5
6
0
1
0
1
1
```

[95]:
```
# Simple automation for finding data for my bar chart
# Third Place on Grid:
for i in range(1,15):
    print(FinalData[(FinalData['grid'] == 3) & (FinalData['positionOrder'] ==␣
 ↪i)].shape[0])
```

```
21
49
56
31
25
9
6
7
2
5
1
1
0
1
```

[99]:
```
# Simple automation for finding data for my bar chart
# Fourth Place on Grid:
for i in range(1,15):
    print(FinalData[(FinalData['grid'] == 4) & (FinalData['positionOrder'] ==␣
 ↪i)].shape[0])
```

```
12
39
32
42
26
24
15
10
5
0
1
0
0
1
```

[120]:
```
# Simple automation for finding data for my bar chart
# Fifth Place on Grid:
for i in range(1,15):
    print(FinalData[(FinalData['grid'] == 5) & (FinalData['positionOrder'] ==␣
 ↪i)].shape[0])
```

```
7
21
31
45
30
25
13
6
2
10
4
0
0
1
```

[138]:
```
# Simple automation for finding data for my bar chart
# Sixth Place on Grid:
for i in range(1,11):
    print(FinalData[(FinalData['grid'] == 6) & (FinalData['positionOrder'] ==␣
 ↪i)].shape[0])
```

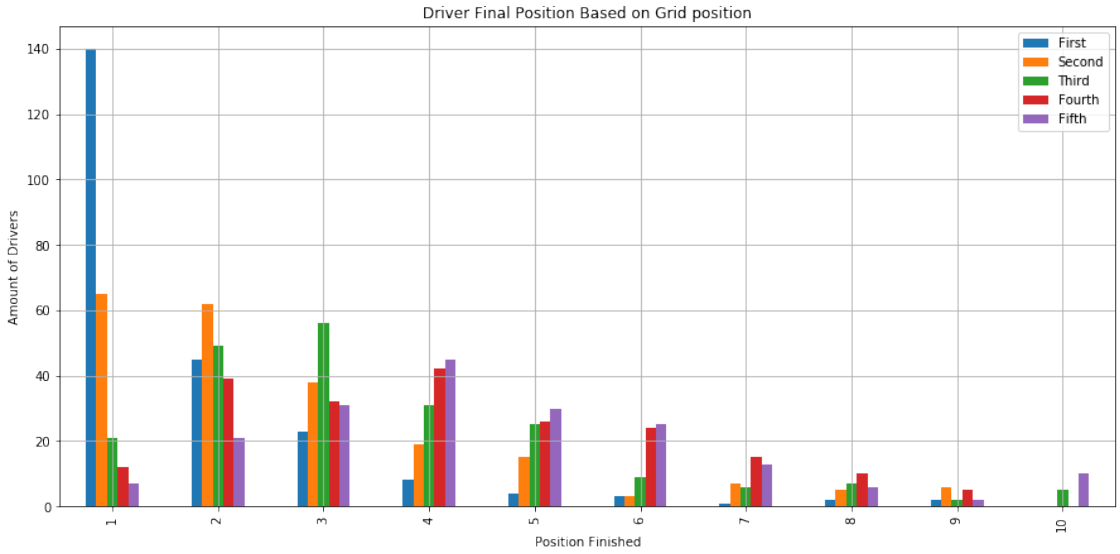
```
6
11
24
31
31
29
17
9
7
3
```

[146]:
```
# Simple automation for finding data for my bar chart
# Seventh Place on Grid:
for i in range(1,15):
    print(FinalData[(FinalData['grid'] == 7) & (FinalData['positionOrder'] ==␣
 ↪i)].shape[0])
```

```
3
9
13
10
28
29
19
18
11
```

```
7
3
6
5
0
```

[151]:
```
import matplotlib.pyplot as plt
First = [140,45,23,8,4,3,1,2,2,0]
Second =[65,62,38,19,15,3,7,5,6,0]
Third = [21,49,56,31,25,9,6,7,2,5]
Fourth =[12,39,32,42,26,24,15,10,5,0]
Fifth = [7, 21, 31, 45, 30, 25, 13, 6, 2, 10]
Sixth = [6,11,24,31,31,29,17,9,7,3]
Seventh = [3,9,13,10,28,29,19,18,11]

#Not used for bar chat


index = ['1','2','3','4','5','6','7','8','9','10']

barPlot = pd.DataFrame({
    'First': First,
    'Second': Second,
    'Third':Third,
    'Fourth':Fourth,
    'Fifth': Fifth

}, index = index)

barPlot.plot.bar(title='Driver Final Position Based on Grid position',␣
 ↪figsize=(15,7), grid = True)
plt.xlabel('Position Finished')
plt.ylabel('Amount of Drivers')
plt.show()
```



[133]:
```
# Checking if Data is Normal Distribution..
FourthDF = pd.DataFrame({
    'Fourth' : Fourth
})

FourthDF.skew(axis = 0) # Which gives us Symmetric Data! This Distribution is␣
 ↪Symmetrix as well...
```

[133]:
```
Fourth    0.1751
dtype: float64
```

[134]:
```
FifthDF = pd.DataFrame({
    'Fifth' : Fifth
})

FifthDF.skew(axis = 0) # Which gives us Symmetric Data!
```

[134]:
```
Fifth    0.592572
dtype: float64
```

[135]:
```
ThirdDF = pd.DataFrame({
    'Third' : Third
})

ThirdDF.skew(axis = 0) # Which gives us Moderately Skewed!
```

[135]:
```
Third    0.909016
dtype: float64
```

[142]:
```
SixthDF = pd.DataFrame({
    'Sixth' : Sixth
})
```

```
SixthDF.skew(axis = 0) # Which gives us Symmetric!
```

[142]:
```
Sixth    0.247902
dtype: float64
```

[152]:
```
SeventhDF = pd.DataFrame({
    'Seventh' : Seventh
})

SeventhDF.skew(axis = 0) # Which gives us Symmetric!
```

[152]:
```
Seventh    0.448699
dtype: float64
```

0.7 Question 2: If you qualify 1, 2 or 3: what is the chance you will finish on the podium?

[76]:
```
FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 1)].
↪shape[0]/230*100
```

[76]:
```
60.86956521739131
```

[78]:
```
FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 2)].
↪shape[0]/230*100
```

[78]:
```
19.565217391304348
```

[77]:
```
FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] == 3)].
↪shape[0]/230*100
```

[77]:
```
10.0
```

[86]:
```
FinalData[(FinalData['grid'] == 1) & (FinalData['positionOrder'] > 3)].shape[0]/
↪230*100
```

[86]:
```
9.565217391304348
```

Given a first place qualification:
drivers have a 90.45% chance of finishing on the podium and a 9.55% chance of missing podium

[70]:
```
FinalData[(FinalData['grid'] == 2) & (FinalData['positionOrder'] == 1)].
↪shape[0]/230*100
```

[70]:
```
28.26086956521739
```

[71]:
```
FinalData[(FinalData['grid'] == 2) & (FinalData['positionOrder'] == 2)].
↪shape[0]/230*100
```

[71]:
```
26.956521739130434
```

[72]:
```
FinalData[(FinalData['grid'] == 2) & (FinalData['positionOrder'] == 3)].
↪shape[0]/230*100
```

[72]:
```
16.52173913043478
```

[85]:
```
FinalData[(FinalData['grid'] == 2) & (FinalData['positionOrder'] > 3)].shape[0]/
↪230*100
```

[85]:
```
25.217391304347824
```

Given a second place qualification:
drivers have a 71.73% chance of finishing on the podium and a 25.21% chance of missing podium

[73]:
```
FinalData[(FinalData['grid'] == 3) & (FinalData['positionOrder'] == 1)].
↪shape[0]/230*100
```

[73]:
```
9.130434782608695
```

[74]:
```
FinalData[(FinalData['grid'] == 3) & (FinalData['positionOrder'] == 2)].
↪shape[0]/230*100
```

[74]:
```
21.304347826086957
```

[75]:
```
FinalData[(FinalData['grid'] == 3) & (FinalData['positionOrder'] == 3)].
↪shape[0]/230*100
```

[75]:
```
24.347826086956523
```

[84]:
```
FinalData[(FinalData['grid'] == 3) & (FinalData['positionOrder'] > 3)].shape[0]/
↪230*100
```

[84]:
```
38.26086956521739
```

Given a third place qualification:
drivers have a 54.77% chance of finishing on the podium and a 38.23% chance of missing podium

0.8 Next Section: Between 2013 and 2014, a decision was made to manipulate the formula of the cars. Starting in 2014, there would be a new V6 engine with 1600cc / 8 gearbox.

CC is the displacement volume of the engine, so it means that that the engine has more cylinders and a higher swept volume which directly translates into horse power and torque of the vehicle.

https://bleacherreport.com/articles/2003467-are-2014-formula-1-cars-slower-analysing-lap-times-at-australian-grand-prix

Between 2013 and 14 there was a 1.77 second drop in fastest average laptime between racers that stayed on the same team.

Sebastian Vettel: Difference in Race times between 2013 and 2014 average in each race: -2.29 seconds

Fernando Alonso: Difference in Race times between 2013 and 2014 average in each race: -2.07 seconds

Nico Rosberg: Difference in Race times between 2013 and 2014 average in each race: -1.78 seconds

Felipe Massa: Difference in Race times between 2013 and 2014 average in each race: -0.92 seconds

[39]:
```
# Checking the Driver's average lap time Below
```

[41]:
```
FinalData[((FinalData['year'] == 2013) | (FinalData['year'] == 2014)) &␣
↪((FinalData['surname'] == 'Vettel'))].sort_values(by=['surname','name_x']).
↪head(6)
```

[41]:

	Unnamed: 0	resultId	grid	positionOrder	fastestLap	fastestLapTime	\
1487	1487	22064	2	1	51.0	01:43.9	
1518	1518	22525	19	8	51.0	01:45.6	
1464	1464	21714	1	3	42.0	01:30.4	
1469	1469	21778	2	1	55.0	01:37.0	
1493	1493	22179	10	6	18.0	01:39.3	
1480	1480	21932	2	1	40.0	01:50.8	

	fastestLapSpeed	surname	status	year	name_x	name_y
1487	192.451	Vettel	Finished	2013	Abu Dhabi Grand Prix	Red Bull
1518	189.427	Vettel	Finished	2014	Abu Dhabi Grand Prix	Red Bull
1464	211.16	Vettel	Finished	2013	Australian Grand Prix	Red Bull
1469	200.938	Vettel	Finished	2013	Bahrain Grand Prix	Red Bull
1493	196.181	Vettel	Finished	2014	Bahrain Grand Prix	Red Bull
1480	227.657	Vettel	Finished	2013	Belgian Grand Prix	Red Bull

[43]:
```
FinalData[((FinalData['year'] == 2013) | (FinalData['year'] == 2014)) &␣
↪((FinalData['surname'] == 'Alonso'))].sort_values(by=['surname','name_x']).
↪head(6)
```

[43]:

	Unnamed: 0	resultId	grid	positionOrder	fastestLap	fastestLapTime	\
994	994	22068	10	5	55.0	01:43.4	
1021	1021	22526	8	9	45.0	01:47.4	
970	970	21713	5	2	53.0	01:29.6	
1167	1167	22133	5	4	57.0	01:33.2	
1007	1007	22288	4	5	58.0	01:12.6	
975	975	21785	3	8	41.0	01:37.2	

	fastestLapSpeed	surname	status	year	name_x	name_y
994	193.305	Alonso	Finished	2013	Abu Dhabi Grand Prix	Ferrari
1021	186.126	Alonso	Finished	2014	Abu Dhabi Grand Prix	Ferrari
970	213.162	Alonso	Finished	2013	Australian Grand Prix	Ferrari
1167	204.867	Alonso	Finished	2014	Australian Grand Prix	Ferrari
1007	214.527	Alonso	Finished	2014	Austrian Grand Prix	Ferrari
975	200.436	Alonso	Finished	2013	Bahrain Grand Prix	Ferrari

[38]:
```
FinalData[((FinalData['year'] == 2013) | (FinalData['year'] == 2014)) &␣
↪((FinalData['surname'] == 'Rosberg'))].sort_values(by=['surname','name_x']).
↪head(6)
```

[38]:

	Unnamed: 0	resultId	grid	positionOrder	fastestLap	fastestLapTime	\
2156	2156	22066	3	3	51.0	01:44.5	
2313	2313	22130	3	1	19.0	01:32.5	
2172	2172	22284	3	1	50.0	01:12.6	

```
2135        2135     21786     1               9        48.0        01:37.6
2164        2164     22175     1               2        49.0        01:37.0
2146        2146     21935     4               4        39.0        01:51.6

     fastestLapSpeed  surname    status  year                name_x    name_y
2156          191.41  Rosberg  Finished  2013   Abu Dhabi Grand Prix  Mercedes
2313         206.436  Rosberg  Finished  2014  Australian Grand Prix  Mercedes
2172         214.518  Rosberg  Finished  2014    Austrian Grand Prix  Mercedes
2135         199.647  Rosberg  Finished  2013     Bahrain Grand Prix  Mercedes
2164         200.816  Rosberg  Finished  2014     Bahrain Grand Prix  Mercedes
2146         225.971  Rosberg  Finished  2013     Belgian Grand Prix  Mercedes
```

[49]:
```
FinalData[((FinalData['year'] == 2013) | (FinalData['year'] == 2014)) &␣
↪((FinalData['surname'] == 'Massa'))].sort_values(by=['surname','name_x']).
↪head(6)
```

[49]:
```
     Unnamed: 0  resultId  grid  positionOrder  fastestLap fastestLapTime  \
514         514     22519     4              2        47.0        01:44.8
995         995     22071     7              8        52.0        01:45.4
971         971     21715     4              4        38.0        01:30.2
497         497     22287     1              4        63.0        01:12.6
493         493     22180     7              7        40.0        01:39.3
976         976     21792     4             15        42.0        01:38.8

    fastestLapSpeed surname    status  year                name_x    name_y
514         190.738   Massa  Finished  2014   Abu Dhabi Grand Prix  Williams
995         189.615   Massa  Finished  2013   Abu Dhabi Grand Prix   Ferrari
971         211.558   Massa  Finished  2013  Australian Grand Prix   Ferrari
497         214.553   Massa  Finished  2014    Austrian Grand Prix  Williams
493          196.26   Massa  Finished  2014     Bahrain Grand Prix  Williams
976          197.12   Massa  Finished  2013     Bahrain Grand Prix   Ferrari
```

[62]:
```
FinalData[((FinalData['year'] == 2013) | (FinalData['year'] == 2014)) &␣
↪((FinalData['surname'] == 'Button'))].sort_values(by=['surname','name_x']).
↪head(6)
```

[62]:
```
     Unnamed: 0  resultId  grid  positionOrder  fastestLap fastestLapTime  \
169         169     22075    12             12        43.0        01:46.3
192         192     22522     6              5        47.0        01:46.7
148         148     21720    10              9        41.0        01:30.2
258         258     22132    10              3        39.0        01:32.9
176         176     22294    11             11        60.0        01:12.9
153         153     21787    10             10        49.0        01:37.7

    fastestLapSpeed surname    status  year                name_x   name_y
169          188.03  Button  Finished  2013   Abu Dhabi Grand Prix  McLaren
192          187.32  Button  Finished  2014   Abu Dhabi Grand Prix  McLaren
148         211.654  Button  Finished  2013  Australian Grand Prix  McLaren
258          205.46  Button  Finished  2014  Australian Grand Prix  McLaren
```

```
176         213.752  Button  Finished  2014    Austrian Grand Prix  McLaren
153          199.33  Button  Finished  2013     Bahrain Grand Prix  McLaren
```

[65]:
```
FinalData[((FinalData['year'] == 2013) | (FinalData['year'] == 2014)) &␣
 ↪((FinalData['surname'] == 'Ricciardo'))].
 ↪sort_values(by=['surname','name_x']).head(6)
```

[65]:
```
      Unnamed: 0  resultId  grid  positionOrder  fastestLap fastestLapTime  \
1519        1519     22521    20              4        50.0        01:44.5
1500        1500     22291     5              8        55.0        01:13.1
1494        1494     22177    13              4        38.0        01:39.3
1640        1640     22373     5              1        44.0        01:53.0
1774        1774     21941    19             10        38.0        01:51.0
1502        1502     22308     8              3        34.0        01:38.5

     fastestLapSpeed    surname    status  year                name_x  \
1519         191.341  Ricciardo  Finished  2014  Abu Dhabi Grand Prix
1500         213.161  Ricciardo  Finished  2014   Austrian Grand Prix
1494         196.266  Ricciardo  Finished  2014    Bahrain Grand Prix
1640         223.187  Ricciardo  Finished  2014    Belgian Grand Prix
1774         227.224  Ricciardo  Finished  2013    Belgian Grand Prix
1502         215.395  Ricciardo  Finished  2014    British Grand Prix

          name_y
1519    Red Bull
1500    Red Bull
1494    Red Bull
1640    Red Bull
1774  Toro Rosso
1502    Red Bull
```

[71]:
```
FinalData[((FinalData['year'] == 2013) | (FinalData['year'] == 2014)) &␣
 ↪((FinalData['surname'] == 'Bottas'))].sort_values(by=['surname','name_x']).
 ↪head(6)
```

[71]:
```
      Unnamed: 0  resultId  grid  positionOrder  fastestLap fastestLapTime  \
515          515     22520     3              3        54.0        01:45.7
611          611     22134    15              5        56.0        01:32.6
498          498     22286     2              3        63.0        01:12.6
476          476     21791    15             14        57.0        01:38.2
494          494     22181     3              8        50.0        01:39.8
481          481     21946    20             15        31.0        01:52.7

     fastestLapSpeed surname    status  year                 name_x    name_y
515         189.113  Bottas  Finished  2014   Abu Dhabi Grand Prix  Williams
611         206.128  Bottas  Finished  2014  Australian Grand Prix  Williams
498         214.568  Bottas  Finished  2014    Austrian Grand Prix  Williams
476         198.419  Bottas  Finished  2013     Bahrain Grand Prix  Williams
494         195.296  Bottas  Finished  2014     Bahrain Grand Prix  Williams
```

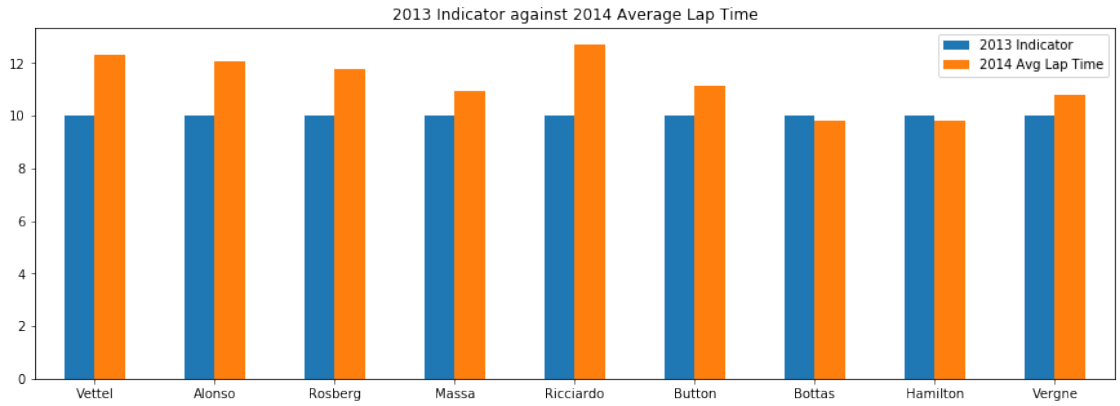
```
481         223.754  Bottas  Finished  2013     Belgian Grand Prix  Williams
```

[73]:
```
FinalData[((FinalData['year'] == 2013) | (FinalData['year'] == 2014)) &␣
↪((FinalData['surname'] == 'Hamilton'))].sort_values(by=['surname','name_x']).
↪head(6)
```

[73]:
```
      Unnamed: 0  resultId  grid  positionOrder  fastestLap fastestLapTime  \
2155        2155     22070     4              7        47.0        01:45.5
2189        2189     22518     2              1        49.0        01:45.6
2130        2130     21716     3              5        45.0        01:29.8
2171        2171     22285     9              2        41.0        01:12.2
2134        2134     21782     9              5        48.0        01:38.2
2163        2163     22174     2              1        49.0        01:37.1

     fastestLapSpeed   surname    status  year                name_x  \
2155         189.586  Hamilton  Finished  2013   Abu Dhabi Grand Prix
2189         189.342  Hamilton  Finished  2014   Abu Dhabi Grand Prix
2130         212.689  Hamilton  Finished  2013  Australian Grand Prix
2171          215.65  Hamilton  Finished  2014    Austrian Grand Prix
2134         198.395  Hamilton  Finished  2013     Bahrain Grand Prix
2163         200.634  Hamilton  Finished  2014     Bahrain Grand Prix

       name_y
2155  Mercedes
2189  Mercedes
2130  Mercedes
2171  Mercedes
2134  Mercedes
2163  Mercedes
```

[157]:
```
# The chart below signifies a 2014 average lap time versus a 2013 indicator␣
↪time.
Initial = [10,10,10,10, 10, 10, 10, 10, 10]
SpeedIn2014 = [12.29, 12.07, 11.78, 10.92, 12.71, 11.12, 9.79, 9.82, 10.77]
index = ['Vettel', 'Alonso', 'Rosberg', 'Massa', 'Ricciardo', 'Button',␣
↪'Bottas', 'Hamilton', 'Vergne']
df = pd.DataFrame({'2013 Indicator': Initial, '2014 Avg Lap Time':␣
↪SpeedIn2014}, index=index)
df.plot.bar(rot=0, figsize=(15,5), title = '2013 Indicator against 2014 Average␣
↪Lap Time')
```

[157]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c225a6198>
```



The reason for the cars being slower than last year: https://bleacherreport.com/articles/2003467-are-2014-formula-1-cars-slower-analysing-lap-times-at-australian-grand-prix

0.9 Linear Regression

using linear regression to figure out the correlation between Grid Position and Final Position.

[182]:
```
# Trying to check the correlation between Final Position and Grid Position.

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

# Using grid as X and position as y
X = FinalData['grid'].values.reshape(-1,1)
y = FinalData['positionOrder'].values.reshape(-1,1)

# Splitting the Data set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 0)

regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

[182]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[165]:
```
print(regressor.intercept_)
print(regressor.coef_)
```

```
[2.49616411]
[[0.45764567]]
```

[186]:
```
y_pred = regressor.predict(X_test)
```

```
predictions = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted' : y_pred.
 ↪flatten()})
predictions

# Actual Position vs Predicted.
```

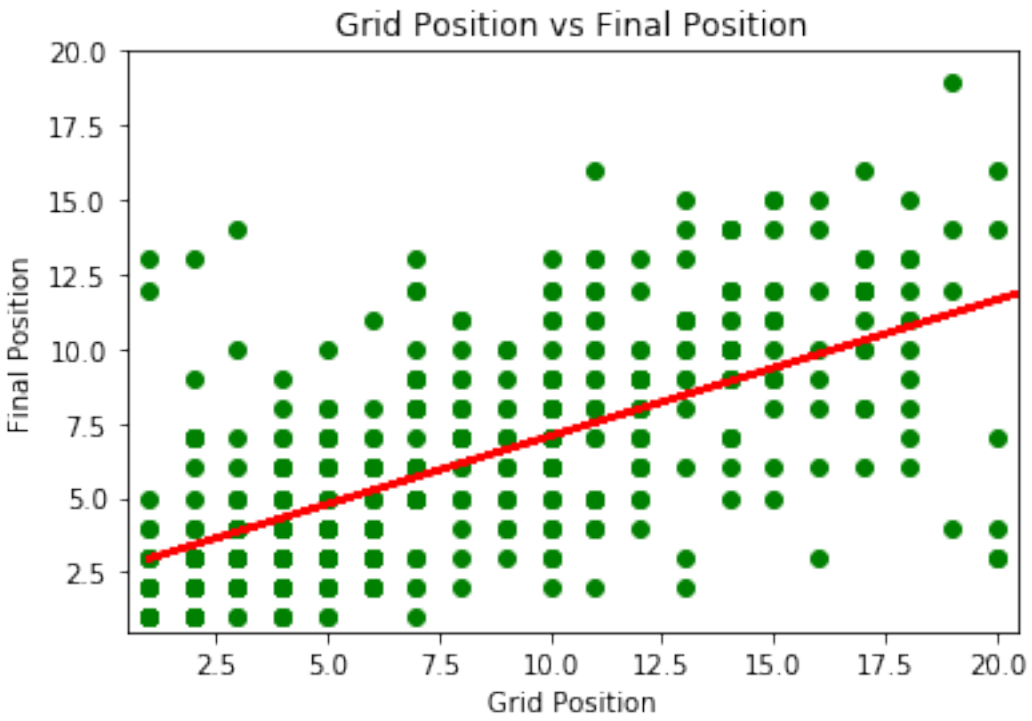
[186]:

	Actual	Predicted
0	2	2.953810
1	8	8.445558
2	5	5.242038
3	5	5.699684
4	4	5.242038
5	4	4.326747
6	1	3.411455
7	8	7.987912
8	9	6.157329
9	10	6.614975
10	5	4.784392
11	5	4.326747
12	6	7.072621
13	4	4.326747
14	2	4.326747
15	8	7.072621
16	2	2.953810
17	3	3.869101
18	1	3.411455
19	7	7.987912
20	2	5.242038
21	3	3.411455
22	2	4.326747
23	10	7.987912
24	8	7.530266
25	8	7.072621
26	4	6.614975
27	5	7.530266
28	5	9.360849
29	1	4.326747
..	...	...
491	9	7.987912
492	19	11.191432
493	3	5.242038
494	5	5.242038
495	6	5.242038
496	4	5.242038
497	2	4.784392
498	2	3.869101
499	1	2.953810

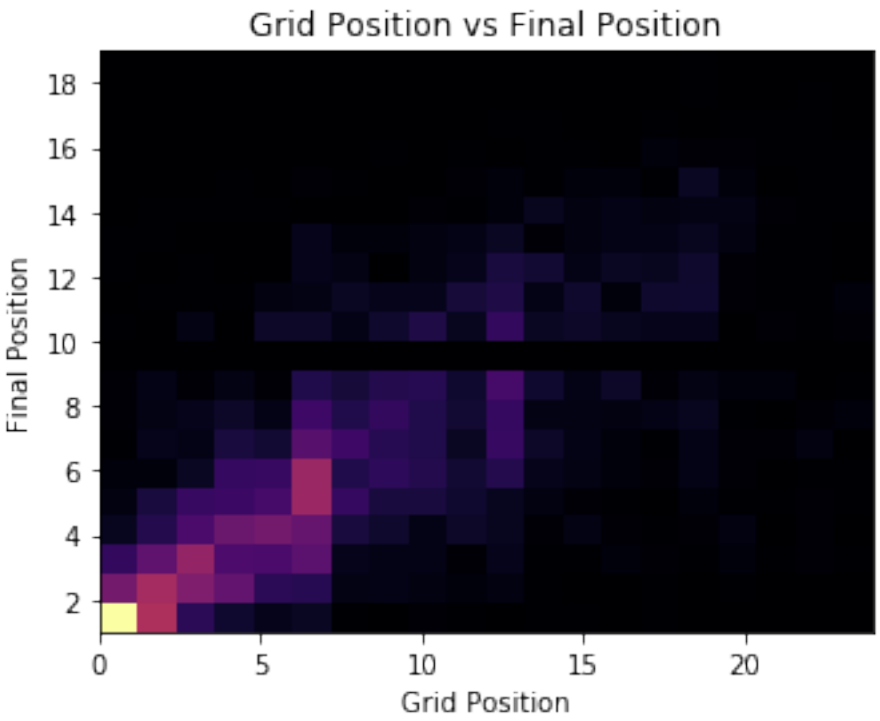
```
500       3   5.699684
501       2   3.411455
502       3   5.242038
503       8   5.699684
504       9   8.903203
505       1   2.953810
506       5   4.326747
507      12   7.530266
508      11  10.733786
509       6   8.445558
510       7   3.411455
511       7   3.411455
512       4   4.326747
513      14   8.903203
514       8   7.072621
515       2   3.411455
516       2   3.411455
517       6   5.242038
518       1   2.953810
519       5   3.411455
520       8   6.157329

[521 rows x 2 columns]
```

[245]:
```
# The problem with using a scatter plot to visualize this data is that there␣
 ↪are only a certain amount of positions
# that drivers can finish.. Between 1 and 20. That means that the graph will be␣
 ↪very uniform.
plt.scatter(X_test, y_test, color='green')
plt.plot(X_test, y_pred, color = 'red', linewidth = 2)
plt.title('Grid Position vs Final Position')
plt.xlabel('Grid Position')
plt.ylabel('Final Position')
plt.ylim(0.5, 20)
plt.xlim(0.5, 20.5)
plt.show
```

[245]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



[247]:
```
# Instead I decided to use a heatmap in order to show each block a little more␣
 ↪clear. Allows us to see the amount of
# drivers instead of just a green dot.

import numpy as np
import numpy.random
import matplotlib.pyplot as plt

heatmap, xedges, yedges = np.histogram2d(FinalData['grid'],␣
 ↪FinalData['positionOrder'], bins=(20))
extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]

plt.clf()
plt.imshow(heatmap.T, extent=extent, origin='lower', cmap='inferno')
plt.title('Grid Position vs Final Position')
plt.xlabel('Grid Position')
plt.ylabel('Final Position')
plt.ylim(1)
plt.xlim(0)
plt.show()
```


Grid Position vs Final Position
Final Position
Grid Position
18
16
14
12
10
8
6
4
2
0
5
10
15
20

[ ]: