COMP257 - Data Science Project

Friday 1pm Group D

Predicting Formula 1 Drivers Salary based on pervious years performance

Project Summary

Formula 1 also know as F1, is one of the most elite sports in the world. The first F1 season was held in 1950 which means the sport has a legacy of over half a century. In 1950 there were a total of 7 races that made up the season and now in 21 the competition has expanded globally to 21 races.

We will be closely monitoring how factors such as driver age, number of world championships, pole positions and race wins will ultimately affect the drivers salary in the coming year.

Project Goal

- Find race factors that most impact a drivers salary.
- Model previous F1 driver salaries.
- Predict 2019 driver salaries based on drivers' previous performance.

Data Sources Summary

Driver Salary Data

Our data has been collected from various sources including outlets such as Forbes, the BBC, Crash.net (one of the oldest motorsport website in the world) and other sources. Most annual wages are estimated as drivers will received a higher bonus – if they perform above expectations.

Ergast Developer API

> The Ergast Developer API is an experimental web service which provides a historical record of motor racing data for non-commercial purposes.
>
> The API provides data for the Formula One series, from the beginning of the world championships in 1950.

Ergast Developer API (http://ergast.com/mrd/?fbclid=IwAR1giD7DxhujDLdjom8lL1WDPRCjpK2LVBegBVZ8NldsKuaTjnY1ndtZI_I)

We will be using this data as our main source for each drivers season and race data to compare against their salaries for each season. The website also provides a direct dump of the data in CSV or MySQL format. We will be downloading this data as a CSV and then using pandas to transform it into a format we like.

Data Manipulation Processes

Ergast Data Manipulation

As this data came as a database in CSV format that was heavily normalised, it needed to be transformed into a format where it was more accessible and usable for us.

To start I had to filter the data down so that we only had the 2013-2018 relevant data, This was done by:

1. filtering down to the races that happened in this time period.
2. Finding the results, lap times, pit stops, etc for those races.
3. Finding the drivers and constructors that competed in those races from the results of those races.
4. Finding which circuits *(tracks)* that those races took place on.

We will then need to match the data from the Ergast DB to our salary data that we have scraped from various websites, this will be done by matching the salary data onto the driverId's provided by Ergast.

Finally once we have the data we need, we can use it to get a range of predictors and features about each driver and their performance through the years.

If you would like to see this process check out [Data Manipulation Process.ipynb (https://github.com/MQCOMP257/data-science-project-comp_pract_02-fri-1pm-_group-d/blob/master/Data%20Manipulation%20Process.ipynb)](https://github.com/MQCOMP257/data-science-project-comp_pract_02-fri-1pm-_group-d/blob/master/Data%20Manipulation%20Process.ipynb)

Data Analysis Techniques

- Exploratary Data Analysis (EDA)
- Correlation Matrix
- Recursive Feature Elimination
- Linear and multiple regression
- Kmeans Clustering
- K Nearest Neighbor

Exploratary Data Analysis (EDA)

In [1]:
```
# import all necessary libraries
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
import seaborn as sns
import numpy as np
```

Load the data

In [2]:
```
data_2018 = pd.read_csv('data/2018_Data.csv', encoding='latin-1')
data_2018.head()
```

Out[2]:

	Driver Name	Salary	Team	Driver Status	Age	Number of World Championships	Number of Pole Positions	Number of Race Wins	Number of Podiums	Number of DNF
0	Sebastian Vettel	60000000	Ferrari	0	31	4	5	5	12	1
1	Lewis Hamiton	50000000	Mercedes	0	33	4	11	11	17	1
2	Kimi Räikkönen	40000000	Ferrari	1	38	1	1	1	12	4
3	Fernando Alonso	30000000	McLaren F1 Team	0	37	2	0	0	0	8
4	Valtteri Bottas	12000000	Mercedes	1	29	0	2	0	8	2

Remove spaces within the dataset.

In [3]:
```
headers = data_2018.columns
new_headers = []

for header in headers:
    new_headers.append(header.strip())

data_2018.columns = new_headers
```

Data Dictionary

- Driver Name: Name of the driver. There were 20 drivers in the F1 2018 season.
- Salary - Driver's Salary in US Dollars (USD).
    - Note: Some drivers are paid drivers (which means that, instead of being paid by the owner of his car, drives for free and brings with him either personal sponsorship or personal or family funding to finance the team's operations).
- Team: Driver's Team. There were 10 teams competing in 2018.
- Driver Status. Each driver will have a driver status within his team.
    - Note: The first driver (coded as 0) will usually have the latesst resources (such as new parts), technical development and he will be the priority of the team. Nonetheless, some teams do not implement such system.
- Age: The age of the driver.
- Number of World Championships: Each year the driver with the most championship points will be winning the 'World Champtionship'.
- Number of Pole Positions: Pole position is the position at the inside of the front row at the start of a racing event. This position is typically given to the car and driver with the best qualifying time in the trials before the race (the leader in the starting grid). This number-one qualifying driver is referred to as the pole sitter.
- Number of Race Wins: The number of times the driver has won the race. There were 21 races in 2018 season.
- Number of Podiums: Podiums refers to whether the driver has finished within the top 3 positions in the race.
- Number of DNF (Did Not Finish): The number of times that driver does not finish a race. This can due mechnical failture, the driver crashed his cars or other reasons.
- Number of DSQ (Disqualified): The number of times a driver gets disqualified as he (or his team) breach the rules.
- Number of DNS (Did Not Start): The number of times that a driver did not start a race, this can due illness or the mechanics are not able to fix the car due to eariler crash.
- WD (Withdrawn): The number of times a driver withdraw from a race.
- Average Grid Position: The average starting position of each of the drivers.
- Average Finish Position: The average finish position of each of the drivers.
- Lead Lap Finish.
- Points: Number of points that a driver scored in the 2018 F1 season.
- Number of Fastest Laps: The number of times a driver sets the quickest lap run during a race.
- Rookie Status ('Yes' is coded 1, while 'No'is coded 0). Rookie driver refers to the fact a driver is starting his first Formula 1 season.
- Pay Driver: As mentioned above.

Check Missing Data

Firstly, we should have a look whether the data is completed or not. Because the missing value will have an adverse impact on the building of regression model.

In [4]:
```
null_values_col = data_2018.isnull().sum()
null_values_col = null_values_col[null_values_col != 0].sort_values(ascending = False
).reset_index()
null_values_col.columns = ["variable", "number of missing"]
null_values_col
```

Out[4]:

variable	number of missing

There are no missing values.

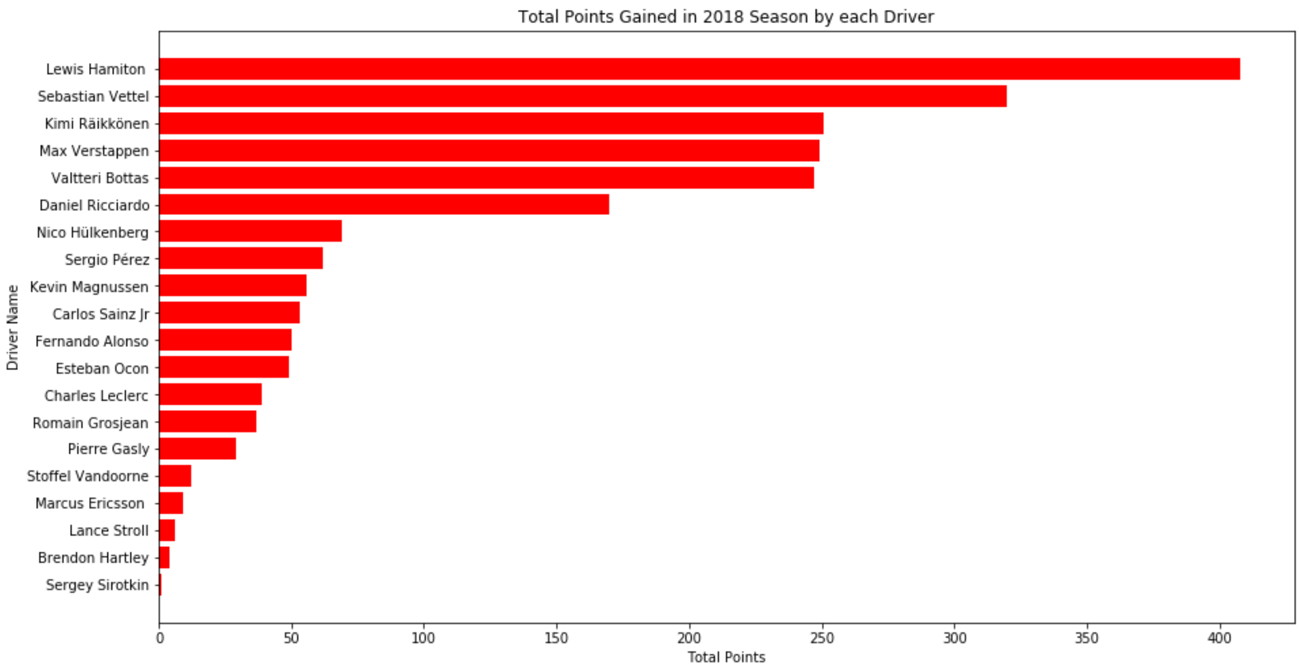
Descriptive Statistics

In [5]:
```
data_2018.describe()
```

Out[5]:

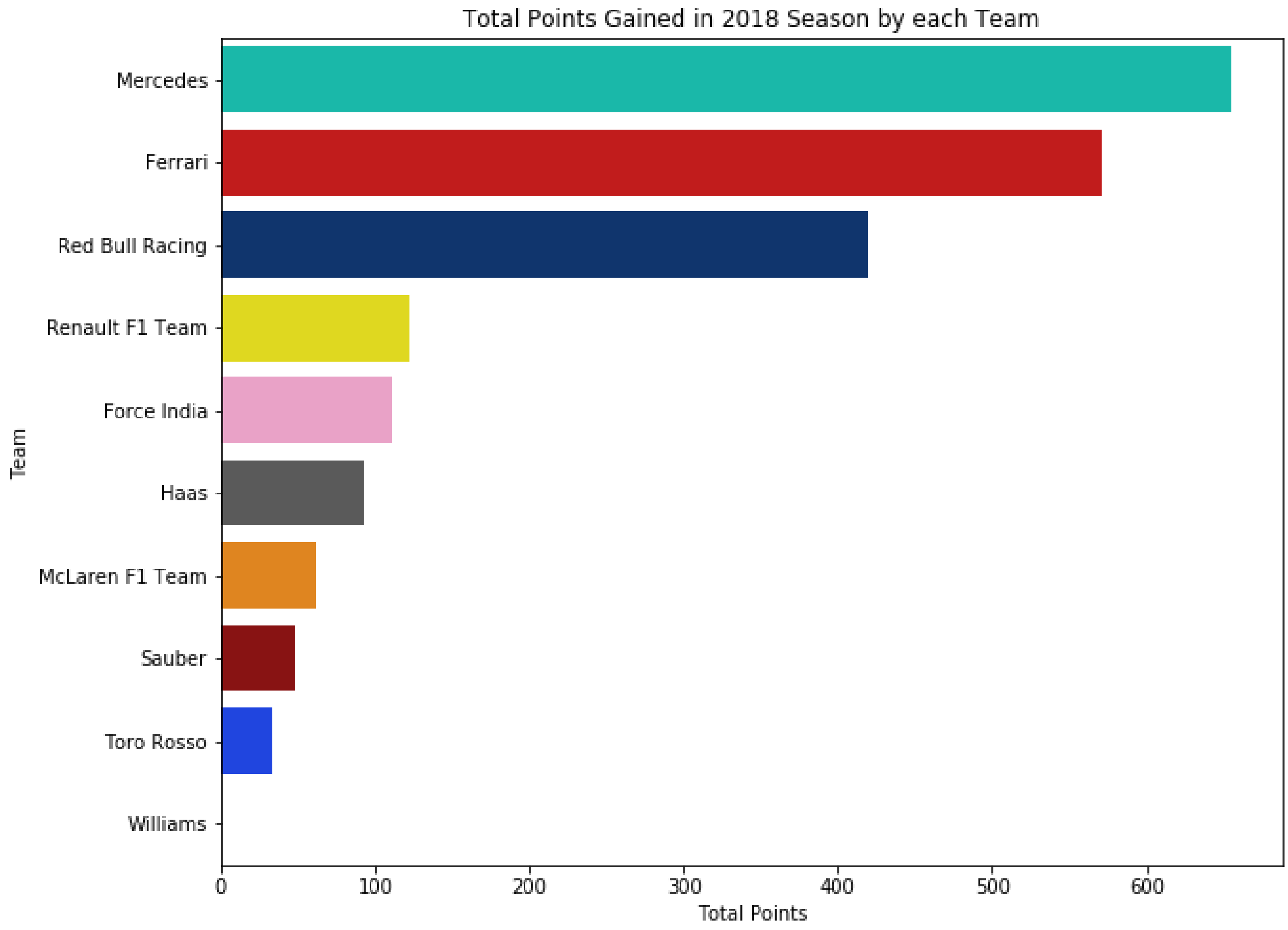
	Salary	Driver Status	Age	Number of World Championships	Number of Pole Positions	Number of Race Wins	Number of Podiums	Number of DNF
count	2.000000e+01	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000
mean	1.202000e+07	0.400000	27.100000	0.550000	1.100000	1.050000	3.150000	4.250000
std	1.796935e+07	0.502625	5.543132	1.276302	2.633789	2.645254	5.470254	2.244877
min	1.500000e+05	0.000000	19.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	9.375000e+05	0.000000	22.750000	0.000000	0.000000	0.000000	0.000000	2.750000
50%	4.725000e+06	0.000000	27.500000	0.000000	0.000000	0.000000	0.000000	4.000000
75%	1.050000e+07	1.000000	31.000000	0.000000	1.000000	0.250000	3.500000	6.000000
max	6.000000e+07	1.000000	38.000000	4.000000	11.000000	11.000000	17.000000	8.000000

In [6]:
```
# order points in an ascending order
driver_points = data_2018[['Driver Name', 'Points']].sort_values(['Points'], ascendin
g=True)

# bar chart of total points gained by each driver
plt.figure(figsize=(15,8))
plt.barh(driver_points['Driver Name'], driver_points['Points'], color='red')
plt.title('Total Points Gained in 2018 Season by each Driver')
plt.xlabel('Total Points')
plt.ylabel("Driver Name")
plt.show()
```



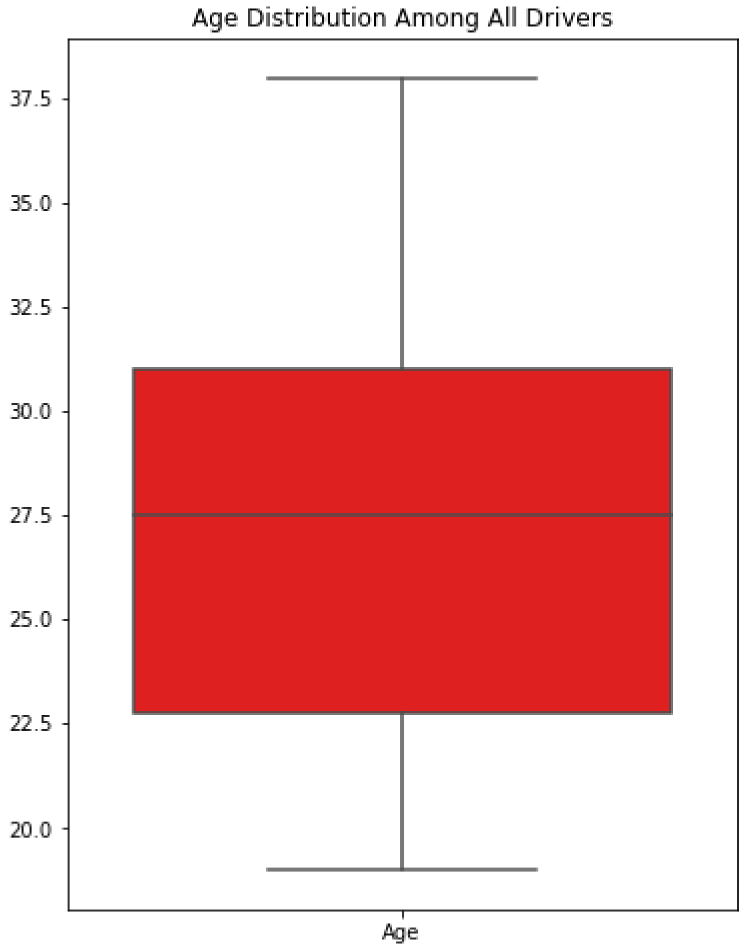
The top 6 drivers gained significantly more point in 2018 season compared to the other drivers. It seems these top drivers dominated this season.

In [7]:
```python
# points are grouped and sorted
team_points = data_2018[['Team', 'Points']].sort_values(['Points'], ascending=True)
team_points = team_points.groupby(['Team']).sum()
team_points = team_points.sort_values(['Points'], ascending=False)

# bar chart of total points gained by each team
team_colours = ["#00D2BE",
                "#DC0000",
                "#00327D",
                "#FFF500",
                "#F596C8",
                "#5A5A5A",
                "#FF8700",
                "#9B0000",
                "#0032FF",
                "#FFFFFF"]

plt.figure(figsize=(10,8))
ax = sns.barplot(y = team_points.index,
                 x = team_points["Points"],
                 palette=sns.color_palette(team_colours))
ax.set_title('Total Points Gained in 2018 Season by each Team')
ax.set_xlabel('Total Points')
plt.show()
```



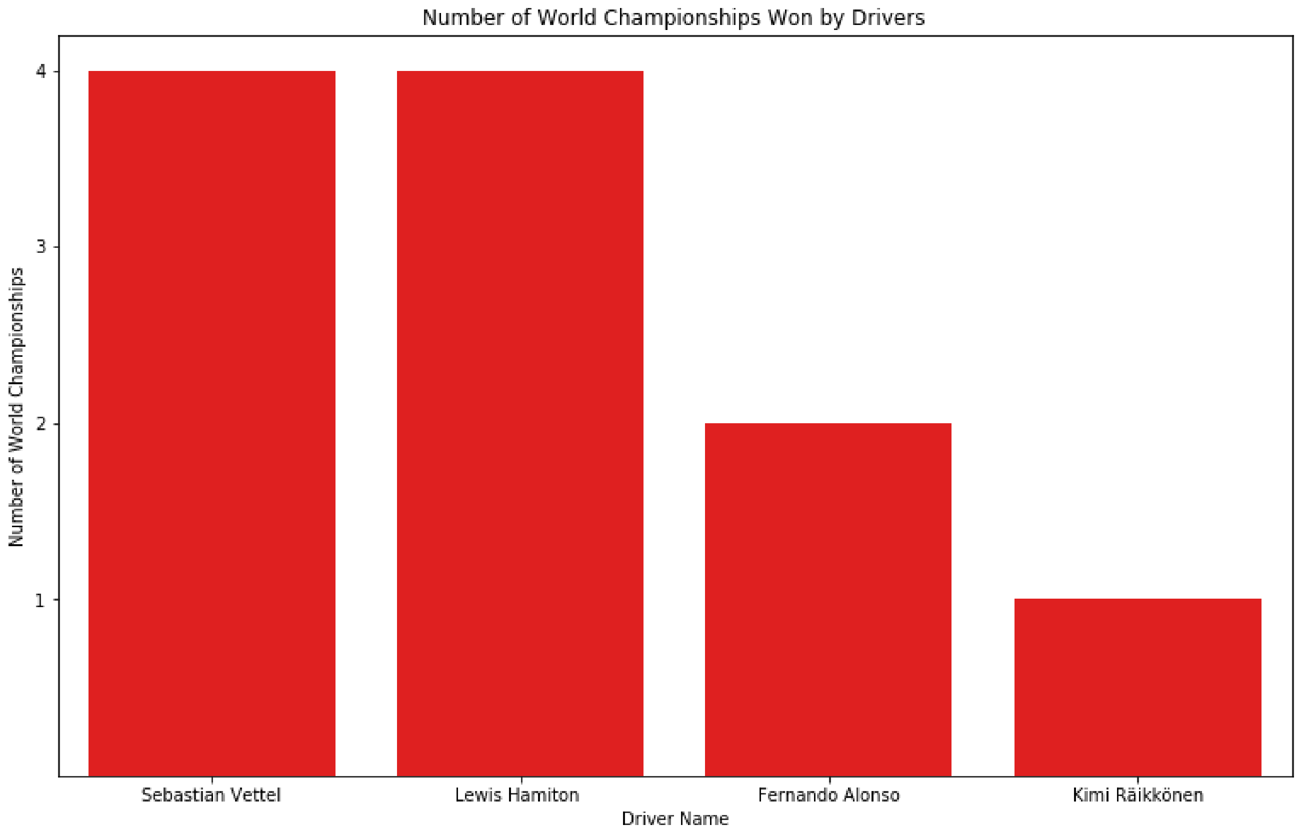
Mercedes team gained the most amount of points, over 400 points, followed by Ferrari and Red Bull Racing. Rest of the teams did not perform as well as the top 3.

In [8]:
```
# boxplot of age
plt.figure(figsize=(6, 8))
sns.boxplot(data=data_2018[['Age']], color = 'red')
plt.title('Age Distribution Among All Drivers')
plt.show()
```



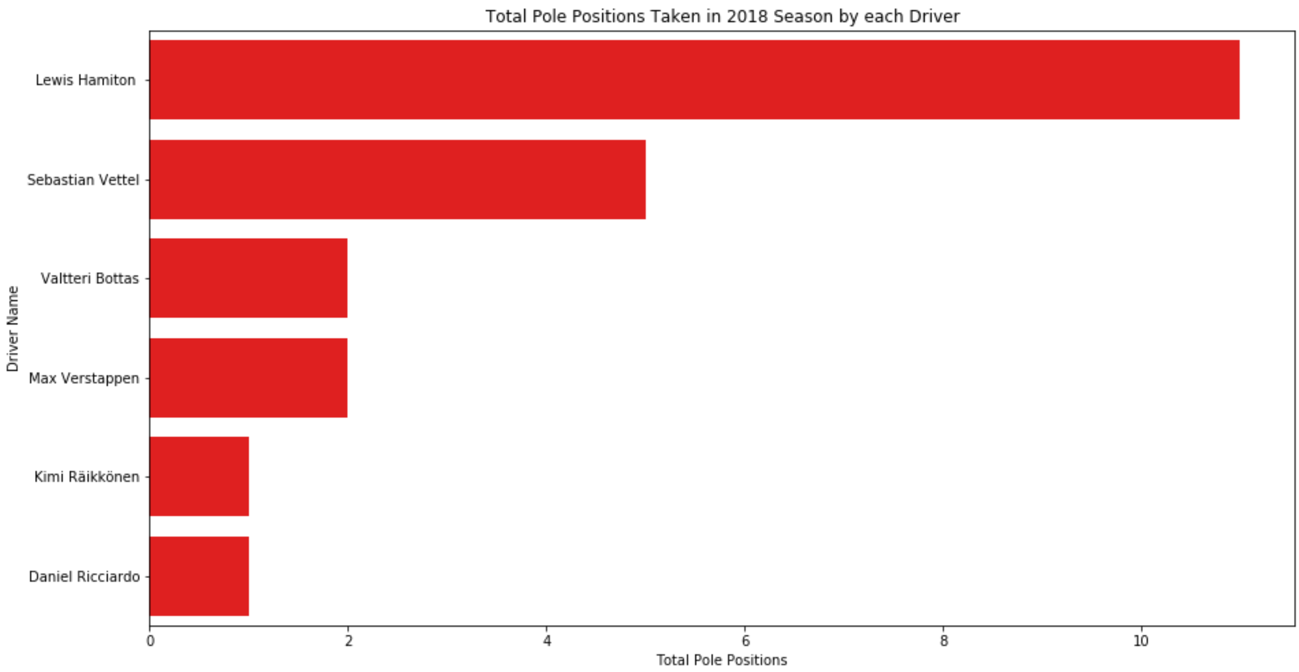
The median age among all drivers is 27.5 years. Age ranges from 19 to 38 years which suggests that drivers only race up to certain age.

In [9]:
```python
# select drivers who won at least one world championship & order them
top_drivers = data_2018[data_2018['Number of World Championships'] > 0].sort_values([
'Number of World Championships'], ascending=False)

# visualise top drivers in a bar chart
plt.figure(figsize=(13,8))
ax = sns.barplot(top_drivers['Driver Name'], top_drivers['Number of World Championshi
ps'], color='red')
ax.set_title('Number of World Championships Won by Drivers')
ax.set_yticks(list(range(1,5)))
plt.show()
```



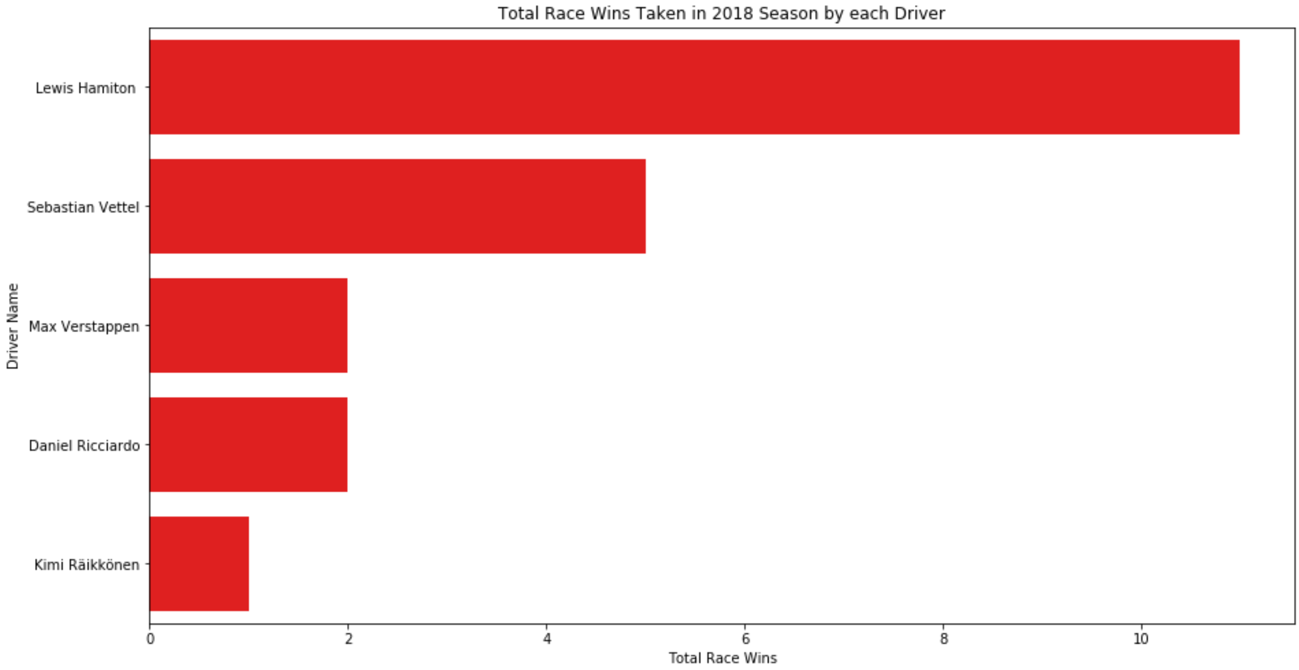
Vettel and Hamiton won 4 world championships, Alonso won 2, and Raikkonen won only one in their career. While, other drivers did not win any world championships.

In [10]:
```
driver_poles = data_2018[['Driver Name', 'Number of Pole Positions']].sort_values(['Number of Pole Positions'], ascending=False)
driver_poles = driver_poles[driver_poles["Number of Pole Positions"] > 0]

# bar chart of total points gained by each driver
plt.figure(figsize=(15,8))
ax = sns.barplot(y = driver_poles['Driver Name'], x = driver_poles['Number of Pole Positions'], color='RED')
ax.set_title('Total Pole Positions Taken in 2018 Season by each Driver')
ax.set_xlabel('Total Pole Positions')
plt.show()
```



It appears that there are only 6 drivers managed to take pole positions in F1 2018. Namely, these drivers are from Mercedes, Ferrari and Red Bull. Given the fact that, these are the most dominant teams in F1 therfore it is valid.

In [11]:
```
driver_wins = data_2018[['Driver Name', 'Number of Race Wins']].sort_values(['Number of Race Wins'], ascending=False)
driver_wins = driver_wins[driver_wins["Number of Race Wins"] > 0]

# bar chart of total points gained by each driver
plt.figure(figsize=(15,8))
ax = sns.barplot(y = driver_wins['Driver Name'], x = driver_wins['Number of Race Wins'], color='RED')
plt.title('Total Race Wins Taken in 2018 Season by each Driver')
plt.xlabel('Total Race Wins')
plt.show()
```



From the above graph, it shows that Hamiton has won the most races, which demostrates how dominant the Mercedas an car was in 2018. On the other hand, Vettel has won 5 and Räikkönen has won 1. Max Verstappen is the only non Mercedas or Ferrari driver which has won the race.

Correlation between variables

In [12]:
```
data_2018.corr()
```

Out[12]:

	Salary	Driver Status	Age	Number of World Championships	Number of Pole Positions	Number of Race Wins	Number of Podiums	Numb of DN
Salary	1.000000	-0.156814	0.643115	0.935807	0.751272	0.741394	0.811580	-0.3402(
Driver Status	-0.156814	1.000000	-0.052894	-0.278951	-0.190837	-0.213761	-0.061255	0.1865
Age	0.643115	-0.052894	1.000000	0.520014	0.312918	0.304741	0.381341	0.1205
Number of World Championships	0.935807	-0.278951	0.520014	1.000000	0.828262	0.833246	0.703720	-0.3444
Number of Pole Positions	0.751272	-0.190837	0.312918	0.828262	1.000000	0.981312	0.839110	-0.4762
Number of Race Wins	0.741394	-0.213761	0.304741	0.833246	0.981312	1.000000	0.796009	-0.4010
Number of Podiums	0.811580	-0.061255	0.381341	0.703720	0.839110	0.796009	1.000000	-0.4832
Number of DNF	-0.340209	0.186582	0.120543	-0.344430	-0.476241	-0.401056	-0.483240	1.0000(
Number of DSQ	-0.208751	-0.057166	-0.085529	-0.185731	-0.180006	-0.171079	-0.248186	0.1439
Number of DNS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
WD	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Average Grid Position	-0.711697	-0.095138	-0.446270	-0.574791	-0.675750	-0.625645	-0.847718	0.3423
Average Finish Position	-0.748174	-0.039316	-0.368375	-0.633121	-0.768949	-0.715033	-0.921809	0.4818
Lead Lap Finish	0.705150	0.089546	0.329935	0.570670	0.704248	0.656533	0.890732	-0.4000
Points	0.803036	-0.020102	0.412447	0.709245	0.847225	0.807977	0.964864	-0.4106
Number of Fastest Laps	0.387086	0.200820	0.227111	0.295472	0.509921	0.412846	0.609171	-0.2778
Rookie Status	-0.217821	0.068041	-0.345501	-0.147376	-0.142833	-0.135750	-0.196934	0.0380
Pay Driver?	-0.339204	-0.235702	-0.224410	-0.255262	-0.247394	-0.235125	-0.319442	-0.1187

Comments: Relatively high correlation can be observed between 'Salary in USD' and the following variables:

- Age
- Number of World Championships
- Number of Pole Positions
- Number of Race Wins
- Number of Podiums
- Average Grid Position
- Average Finish Position
- Points

Read/Prepare 2014 - 2018 data

In [13]:
```
f1 = []

for i, year in enumerate(list(range(2014,2019))):
    f1.append(pd.read_csv("data/%s_Data.csv"%str(year), encoding='latin-1'))
    f1[i]["year"] = year
    f1[i].columns = [x.strip() for x in f1[i].columns]

f1_df = pd.concat(f1, sort=True)
f1_df = f1_df.reset_index()
```

Import Data

In [14]:
```
notPayDriver = f1_df["Pay Driver?"] == 0

f1_df = f1_df[notPayDriver].copy()
```

Remove Pay drivers from our data as they do not recieve a salary for driving

In [15]:
```
team_converter = {'caterham': "caterham",
                  'ferrari': "ferrari",
                  'force india': 'force india',
                  'lotus': 'lotus',
                  'marussia': 'marussia',
                  'mclaren': 'mclaren',
                  'mercedes': 'mercedes',
                  'red bull': 'red bull',
                  'sauber': 'sauber',
                  'toro rosso': 'toro rosso',
                  'williams': 'williams',
                  'manor': 'manor',
                  'mercerdes': 'mercedes',
                  'hass': 'haas',
                  'red bull/toro rosso': 'red bull/toro rosso',
                  'renault': 'renault',
                  'red bull racing': 'red bull',
                  'mclaren f1 team': 'mclaren',
                  'renault f1 team': 'renault',
                  'haas': 'haas'}

f1_df["Team"] = f1_df["Team"].apply(lambda x: team_converter[x.strip().lower()])

team_list = list(f1_df["Team"].unique())

f1_df["team_num"] = f1_df["Team"].apply(lambda x: team_list.index(x))
```

Convert Teams to Numbers

In [16]:
```
for col in f1_df.columns:
    print(col, f1_df[col].isnull().sum())

f1_df = f1_df.fillna(0)

print("\nNumber of null Values now:", f1_df.isnull().sum().sum())
```

```
index 0
Age 0
Average Finish Position 0
Average Grid Position 0
Driver Name 0
Driver Status 0
Lead Lap Finish 0
Number of DNF 0
Number of DNS 0
Number of DSQ 0
Number of Fastest Laps 0
Number of Podiums 15
Number of Pole Positions 0
Number of Race Wins 0
Number of World Championships 0
Pay Driver? 0
Points 0
Rookie Status 0
Salary 0
Team 0
WD 0
year 0
team_num 0

Number of null Values now: 0
```

Remove null values

- we filled all null values with 0 as they were all numeric values in the Number of Podiums column

Linear and Multiple Regression

In [17]:
```
#qqplot library for seaborn
import seaborn_qqplot as sqp

#regression libraries
from sklearn.metrics import mean_squared_error, r2_score
from sklearn import linear_model
```

Importing Libraries needed for this section

In [18]:
```
reg = linear_model.LinearRegression()
x = f1_df[['Age']]
y = f1_df['Salary']
reg.fit(x,y)
print("y = x *", reg.coef_, "+", reg.intercept_)
```

```
y = x * [1421802.13534676] + -28904446.83087248
```

In [19]:
```
predicted = reg.predict(x)
print("MSE:", mean_squared_error(y, predicted))
print("R^2:", r2_score(y, predicted))
```

```
MSE: 126671840199013.45
R^2: 0.3083638550862182
```

In [20]:
```
MSEs = []
r_squareds = []
cont_params = []
for param in list(f1_df.columns):
    if param != "Salary" and f1_df[param].dtypes == 'int64':
        reg = linear_model.LinearRegression()
        x = f1_df[[param]]
        y = f1_df['Salary']
        reg.fit(x,y)

        # score model
        cont_params.append(param)
        predicted = reg.predict(x)
        mse = mean_squared_error(y, predicted)
        MSEs.append(mse)
        r2 = r2_score(y, predicted)
        r_squareds.append(r2)

model_scores = pd.DataFrame({"param": cont_params, "MSE":MSEs, "rsquared":r_squareds
})
top_features = model_scores.nlargest(5, 'rsquared')
top_features.reset_index()[["param", "MSE", "rsquared"]]
```

Out[20]:

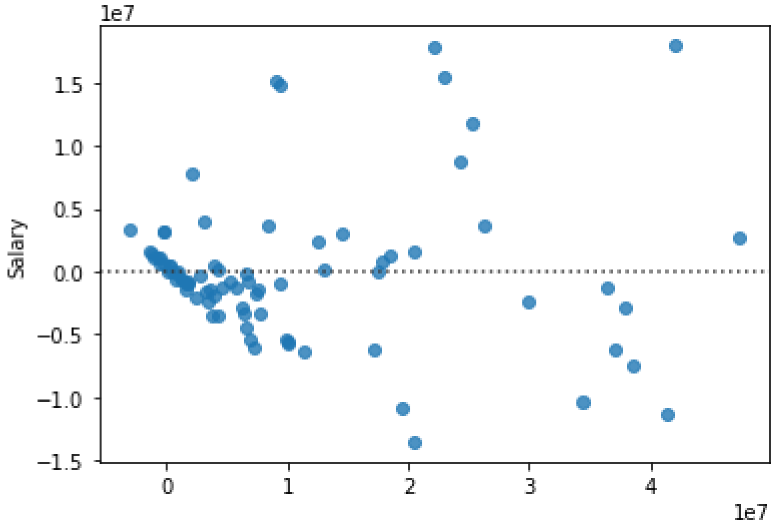
	param	MSE	rsquared
0	Number of World Championships	5.242878e+13	0.713736
1	Points	1.252612e+14	0.316066
2	Age	1.266718e+14	0.308364
3	Number of Race Wins	1.316678e+14	0.281086
4	Number of Pole Positions	1.342991e+14	0.266719

In [21]:
```
reg2 = linear_model.LinearRegression()
x = f1_df[list(top_features['param'])]
y = f1_df['Salary']
reg2.fit(x,y)
predicted = reg2.predict(x)

print("y = x * %s + %s" % ( list(reg2.coef_), reg2.intercept_))
print("MSE: {}".format(mean_squared_error(y, predicted)))
print("R^2: {}".format(r2_score(y, predicted)))
```

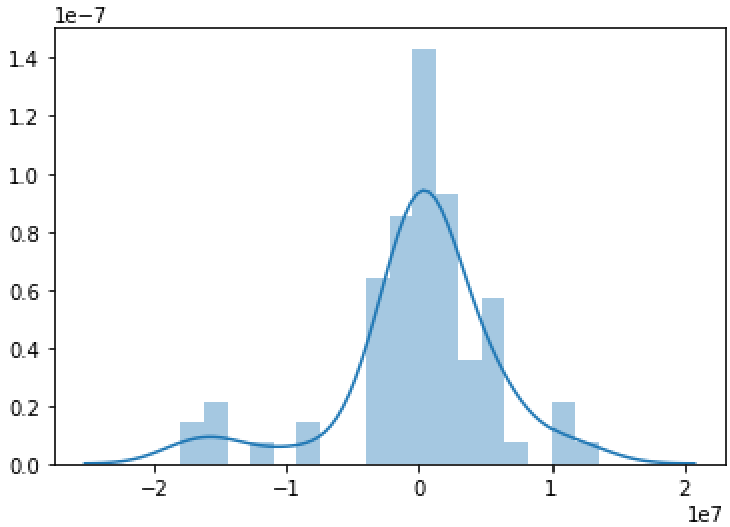
```
y = x * [7436085.3963623755, 10192.451595025297, 727254.8225489911, 118124.477907816
37, 371016.38992398104] + -15933782.000029188
MSE: 33735183037347.508
R^2: 0.81580379737711
```

In [22]:
```
sns.residplot(x=predicted,y=y)
```

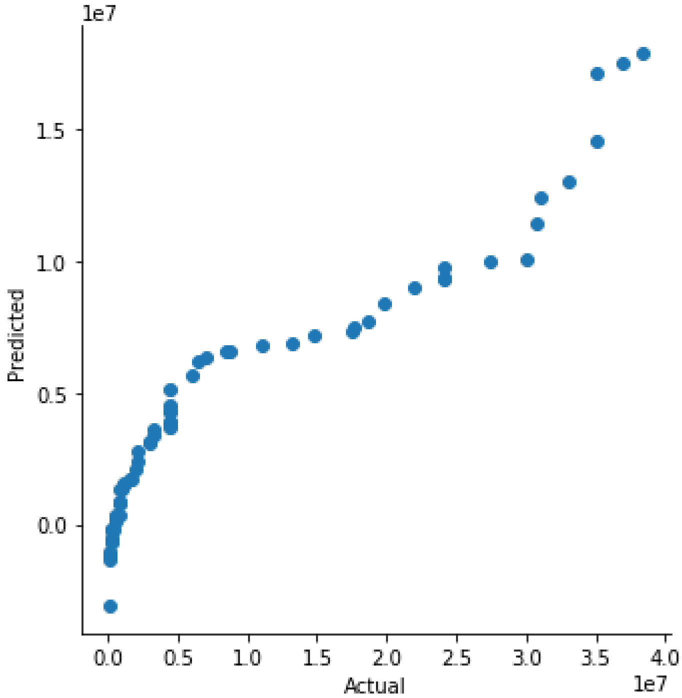
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x1427ada1248>



In [23]:
```
diff = np.array(predicted-y)
sns.distplot(diff)
plt.show()
```



In [24]:
```
results = pd.DataFrame()
results["Predicted"] = predicted
results["Actual"] = y

ax = sqp.qqplot(y="Predicted", x="Actual", data=results, height=5)
plt.show()
```



Kmeans Clutering and K-Nearest Neighbor Classification

Preparation

In [25]:
```
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
```

- Import needed packages

Kmeans Clustering

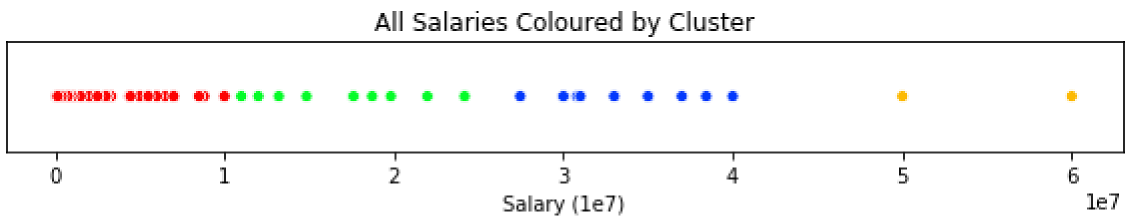
In [26]:
```
X = f1_df[["Salary"]]
kmeans = KMeans(n_clusters = 4, random_state=0).fit(X)
f1_df["grouping"] = kmeans.labels_
```

Perform Clustering based on salary into 4 clusters

In [27]:
```
ax = sns.boxplot(data=f1_df[["Salary", "grouping"]].sort_values("Salary"),
                 y="Salary",
                 x="grouping",
                 palette = sns.color_palette(['#fcba03', "#fc0303", "#0339fc", "#03fc
28"]))
ax.set_title("Box Plot of Salary Clusters")
ax.set_ylabel("Salary (1e7)")
plt.show()
```

Used box-plots to check the spread of various differnet clustering spreads and found four to to have the most even spreads of Salaries

In [28]:
```
f1_df[["Salary", "grouping"]]
fig = plt.figure(figsize=(10,1))
ax = sns.scatterplot(x="Salary",
                     y=np.zeros_like(f1_df["Salary"]),
                     data=f1_df[["Salary", "grouping"]],
                     hue="grouping",
                     palette=sns.color_palette(['#fcba03', "#fc0303", "#0339fc", "#03
fc28"]),
                     legend=False)
ax.set_yticks([])
ax.set_title("All Salaries Coloured by Cluster")
ax.set_xlabel("Salary (1e7)")
plt.show()
```



Visualise Clusters on a number line colouring each cluster with a different colour

K-Nearest Neightbour Classification

In [29]:
```
train = f1_df[f1_df["year"] < 2018]
test = f1_df[f1_df["year"] == 2018]

X_headers = ['Age',
             'Average Finish Position',
             'Average Grid Position',
             'Driver Status',
             'Lead Lap Finish',
             'Number of DNF',
             'Number of DNS',
             'Number of DSQ',
             'Number of Fastest Laps',
             'Number of Podiums',
             'Number of Pole Positions',
             'Number of Race Wins',
             'Number of World Championships',
             'Points',
             'Rookie Status',
             'team_num',
             'WD']

X_train = train[X_headers]
y_train = train["grouping"]

X_test = test[X_headers]
y_test = test["grouping"]
```

Create Train and test data.

- Our goal is to predict 2018 salaries so that will become out test data, therefore 2014 to 2017 will be our train data

In [30]:
```
knn = KNeighborsClassifier(n_neighbors = 1)
knn.fit(X_train, y_train)
print("Accuracy Score:", knn.score(X_test, y_test))
```

```
Accuracy Score: 0.6666666666666666
```

Results of K-Nearest Neighbour Classification using all Features

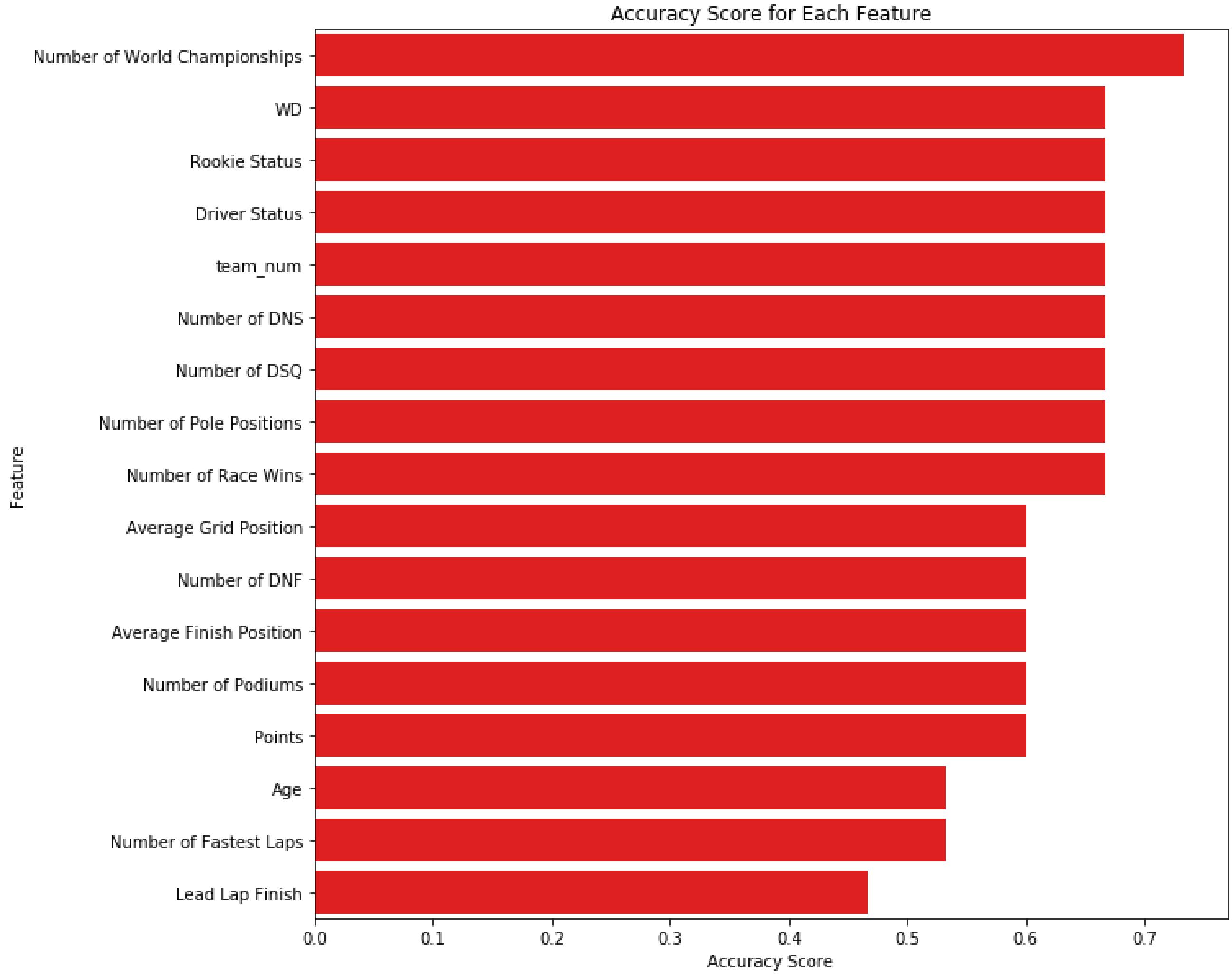
- From these results we can see that using all features we can estimate what catagory of pay a driver will fall into with a moderate degree of accuracy

In [31]:
```
feature_score = pd.DataFrame(columns=["Feature", "Accuracy Score"])
for feature in X_train.columns:
    knn = KNeighborsClassifier(n_neighbors = 1)
    knn.fit(X_train[[feature]], y_train)
    knn.score(X_test[[feature]], y_test)
    feature_score = feature_score.append({"Feature": feature, "Accuracy Score": knn.s
core(X_test[[feature]], y_test)}, ignore_index=True)
```

Accuracy of K-Nearest Neighbour for each feature

- From our lacklustre accuracy score from the previous KNN results we decided to try to filter out some features that did not perform as well to see if this would help our accuracy score

In [32]:
```
feature_score = feature_score.sort_values("Accuracy Score", ascending = False)
plt.figure(figsize=(10,10))
ax = sns.barplot(data=feature_score,
                 x="Accuracy Score",
                 y="Feature",
                 color = "RED")
ax.set_title("Accuracy Score for Each Feature")
plt.show()
```



Graphing the accuracy of each variable in K-Nearest Neighbors

- From the graph below we can see that some features such as "Age", "Number of Fastest Laps" and "Lead Lap Finish" were underperfoming in our K-Nearest Neighbour model.

In [33]:
```
bestFeats = list(feature_score[feature_score["Accuracy Score"] > 0.61].copy()["Featur
e"])

knn = KNeighborsClassifier(n_neighbors = 1)
knn.fit(X_train[bestFeats], y_train)
print("Accuracy Score:", knn.score(X_test[bestFeats], y_test))
```

```
Accuracy Score: 0.7333333333333333
```

K-Nearest Neighbor based on our 9 best features

- Taking our best performing features from the graph above we again perfromed our K-Nearest Neighbour Classification and found that is made an improvement on the Accuracy Score