Data Science

Does Hitting It Far Give Us As Big An Advantage As We Think?

Lucas Steinmetz

Table of Contents

Intro

Being an enthusiastic golf player and supporter, I have always admired the best players on Tour. Especially the ones who bomb the ball. Due to this, I decided to explore the statistics of long hitters and see if hitting it far actually gives you as big an advantage as we think. All the data in this project is from the PGA Tour Website. To find an answer to my question, we will graph and explore various aspects of the game.

Description of Data

An understanding of terms used in the data set we are analyzing is essential. We need to have complete mastery of the vocabulary to find an answer to our question accurately.

- Rounds: The number of games that a player played
- Fairway Percentage: The percentage of time a tee shot lands on the fairway
- Year: The year in which the statistic was collected
- Avg Distance: The average distance of the tee-shot
- gir: (Green in Regulation) is met if any part of the ball is touching the putting surface while the number of strokes taken is at least two fewer than par

- Average Putts: The average number of strokes taken on the green
- Average Scrambling: Scrambling is when a player misses the green in regulation but still makes par or better on a hole
- Average Score: Average Score is the average of all the scores a player has played in that year
- Points: The number of FedExCup points a player earned in that year. These points can be earned by competing in tournaments.
- Wins: The number of competitions a player has won in that year
- Top 10: The number of competitions where a player has placed in the Top 10
- Average SG Putts: Strokes gained: putting measures how many strokes a player gains (or loses) on the greens.
- Average SG Total: The Off-the-tee + approach-the-green + around-the-green + putting statistics combined
- SG:OTT: Strokes gained: off-the-tee measures player performance off the tee on all par-4s and par-5s.
- SG:APR: Strokes gained: approach-the-green measures player performance on approach shots. Approach shots include all shots that are not from the tee on par-4 and par-5 holes and are not included in strokes gained: around-the-green and strokes gained: putting. Approach shots include tee shots on par-3s.

- SG:ARG: Strokes gained: around-the-green measures player performance on any shot within 30 yards of the edge of the green. This statistic does not include any shots taken on the putting green.

Now that we got the basic vocabulary down, we can move onto some code.

Importing Packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing Data Sets

```
df2020=pd.read_csv(r'/Users/lucas/OneDrive - The Hill School/Coding Projects/data sets/pga_tour_stats_2020.csv')
df=pd.read_csv(r'/Users/lucas/OneDrive - The Hill School/Coding Projects/data sets/pgatourstatdata.csv')
```

Examining the Data

We are going to be using two data sets. The main one we will be using (df) comprises most PGA Tour stats from 2010 to 2018. The second one (df2020) includes more stats for each player. However, it is only for the 2020 season. The info for both data sets is below.

```
df2020.head(10)
```

	PLAYER NAME	AVG_APPROACH_BIRDIE_OR_BETTER	AVG_APPROACH_BOGEY_OR_WORSE	AVG_APPROACH_PAR	AVG_APPROACH_SHOT_DISTANCE	FAIRWAY_AI
0	Zac Blair	152.9	158.9	158.5	157.5	
1	Jimmy Walker	156.9	166.8	160.3	161.0	
2	Cameron Percy	161.3	164.0	161.6	161.9	
3	Sebastian Cappelen	161.7	161.1	153.6	156.4	
4	Brian Gay	162.3	159.8	156.1	158.0	
5	Ted Potter, Jr.	162.7	166.2	156.2	159.2	
6	Arjun Atwal	162.9	163.0	156.3	158.7	
7	Vaughn Taylor	163.0	167.7	160.3	162.1	
8	Jason Dufner	164.2	161.4	162.8	162.9	
9	Roger Sloan	164.2	164.2	156.2	159.0	

```
df.head(10)
```

	player_name	Rounds	Fairway Percentage	Year	avg_distance	gir	Average Putts	Average Scrambling	Average Score	Points	Wins	Top 10	Average SG Putts	Average SG Total	SG:OTT	SG:APR	SG:A
0	Henrik Stenson	60	75.19	2018	291.5	73.51	29.93	60.67	69.617	868	NaN	5.0	-0.207	1.153	0.427	0.960	-0
1	Ryan Armour	109	73.58	2018	283.5	68.22	29.31	60.13	70.758	1,006	1.0	3.0	-0.058	0.337	-0.012	0.213	0
2	Chez Reavie	93	72.24	2018	286.5	68.67	29.12	62.27	70.432	1,020	NaN	3.0	0.192	0.674	0.183	0.437	-0
3	Ryan Moore	78	71.94	2018	289.2	68.80	29.17	64.16	70.015	795	NaN	5.0	-0.271	0.941	0.406	0.532	0
4	Brian Stuard	103	71.44	2018	278.9	67.12	29.11	59.23	71.038	421	NaN	3.0	0.164	0.062	-0.227	0.099	0
5	Brian Gay	103	71.37	2018	282.9	64.52	28.25	63.26	70.280	880	NaN	6.0	0.442	0.565	-0.166	0.036	0
6	Kyle Stanley	93	71.29	2018	295.7	71.09	29.89	54.80	70.404	1,198	NaN	5.0	0.037	0.686	0.378	0.298	-0
7	Emiliano Grillo	94	70.16	2018	295.2	68.84	29.04	61.05	70.152	901	NaN	5.0	0.546	1.133	0.364	0.345	-0
8	Russell Henley	77	70.03	2018	293.0	68.77	29.80	54.33	70.489	569	NaN	3.0	0.167	0.541	0.093	0.467	-0
9	Jim Furyk	50	69.91	2018	280.5	63.19	28.73	62.58	70.342	291	NaN	2.0	0.389	0.412	-0.392	0.179	0

```
df2020.shape
```

```
(342, 73)
```

```
df.shape
```

```
(1678, 18)
```

Data Cleaning

In the data set “df,” the column names are not configured in the format I like. For example, instead of “Average Putts,” it would be ideal for it to be “average_putts.” Let’s change that.

```
df.head(10)
```

	player_name	rounds	fairway_percentage	year	avg_distance	gir	average_putts	average_scrambling	average_score	points	wins	top_10	average_sg_p
0	Henrik Stenson	60	75.19	2018	291.5	73.51	29.93	60.67	69.617	868	NaN	5.0	-0
1	Ryan Armour	109	73.58	2018	283.5	68.22	29.31	60.13	70.758	1,006	1.0	3.0	-0
2	Chez Reavie	93	72.24	2018	286.5	68.67	29.12	62.27	70.432	1,020	NaN	3.0	0
3	Ryan Moore	78	71.94	2018	289.2	68.80	29.17	64.16	70.015	795	NaN	5.0	-0
4	Brian Stuard	103	71.44	2018	278.9	67.12	29.11	59.23	71.038	421	NaN	3.0	0
5	Brian Gay	103	71.37	2018	282.9	64.52	28.25	63.26	70.280	880	NaN	6.0	0
6	Kyle Stanley	93	71.29	2018	295.7	71.09	29.89	54.80	70.404	1,198	NaN	5.0	0
7	Emiliano Grillo	94	70.16	2018	295.2	68.84	29.04	61.05	70.152	901	NaN	5.0	0
8	Russell Henley	77	70.03	2018	293.0	68.77	29.80	54.33	70.489	569	NaN	3.0	0
9	Jim Furyk	50	69.91	2018	280.5	63.19	28.73	62.58	70.342	291	NaN	2.0	0

Much better. After briefly glancing at the data, we need to...

- Convert NaN values into 0.
- Change Top 10, Wins, Points, and Round columns into integers.
- Remove dollar signs and commas in the Money column.

```
#converts NaN values to 0
df['wins'].fillna(0, inplace=True)

df['top_10'].fillna(0, inplace=True)

df2020['FINISHES_TOP10'].fillna(0, inplace=True)

df.dropna(axis = 0, inplace=True)
```

```
#converts into integers
df['points'] = df['points'].apply(lambda x: x.replace(',',''))
df['points'] = df['points'].astype(int)

df['top_10'] = df['top_10'].astype(int)

df['rounds'] = df['rounds'].astype(int)

df['wins'] = df['wins'].astype(int)

#removes commans and money signs
df['money'] = df['money'].apply(lambda x: x.replace('$',''))
df['money'] = df['money'].apply(lambda x: x.replace(',',''))
df['money'] = df['money'].astype(float)
```

That should fix it. Let's double-check.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1674 entries, 0 to 1677
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   player_name          1674 non-null   object 
 1   rounds               1674 non-null   int64  
 2   fairway_percentage   1674 non-null   float64
 3   year                 1674 non-null   int64  
 4   avg_distance         1674 non-null   float64
 5   gir                  1674 non-null   float64
 6   average_putts        1674 non-null   float64
 7   average_scrambling   1674 non-null   float64
 8   average_score        1674 non-null   float64
 9   points               1674 non-null   int64  
 10  wins                 1674 non-null   int64  
 11  top_10               1674 non-null   int64  
 12  average_sg_putts     1674 non-null   float64
 13  average_sg_total     1674 non-null   float64
 14  sg_ott               1674 non-null   float64
 15  sg_apr               1674 non-null   float64
 16  sg_arg               1674 non-null   float64
 17  money                1674 non-null   float64
dtypes: float64(12), int64(5), object(1)
memory usage: 248.5+ KB
```

Looks good here. Let's see if the money column looks good.

```
df.head()
```

distance	gir	average_putts	average_scrambling	average_score	points	wins	top_10	average_sg_putts	average_sg_total	sg_ott	sg_apr	sg_arg	money
291.5	73.51	29.93	60.67	69.617	868	0	5	-0.207	1.153	0.427	0.960	-0.027	2680487.0
283.5	68.22	29.31	60.13	70.758	1006	1	3	-0.058	0.337	-0.012	0.213	0.194	2485203.0
286.5	68.67	29.12	62.27	70.432	1020	0	3	0.192	0.674	0.183	0.437	-0.137	2700018.0
289.2	68.80	29.17	64.16	70.015	795	0	5	-0.271	0.941	0.406	0.532	0.273	1986608.0
278.9	67.12	29.11	59.23	71.038	421	0	3	0.164	0.062	-0.227	0.099	0.026	1089763.0

Basic Analysis

To find out if hitting the ball far gives you an advantage, we should look at the correlation between wins and the average distance. The closer to 0 the value is, the less correlation it has to the number of wins. For example, just because average_score is negative, it does not mean that average_score has the most negligible correlation to wins. The closer a number is to -1 or 1, the more correlation it possesses. As we can see, driving distance has a below-average correlation to the number of wins.

Correlation

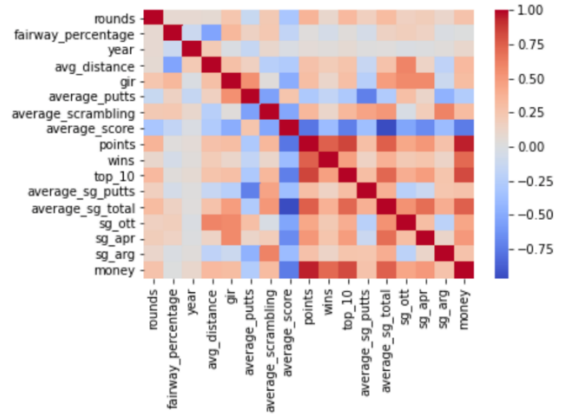
```
df.corr()['wins']
```

```
rounds                0.103162
fairway_percentage   -0.047949
year                  0.039006
avg_distance          0.206294
gir                   0.120340
average_putts        -0.168764
average_scrambling    0.125193
average_score        -0.390254
points                0.750110
wins                  1.000000
top_10                0.473453
average_sg_putts      0.149155
average_sg_total      0.384932
sg_ott                0.232414
sg_apr                0.259363
sg_arg                0.134948
money                 0.721665
Name: wins, dtype: float64
```

Here is a more visual representation of the correlation.

```
corr = df.corr()
sns.heatmap(corr, xticklabels=corr.columns.values, yticklabels=corr.columns.values, cmap="coolwarm")
```

```
<AxesSubplot:>
```



Average distance is most correlated to strokes-gained-off-the-tee and fairway percentage. Wins are most correlated to money and points. Now let's approach our question with a more individual approach. Maybe we will get more definite answers. We will find the average distance of players who haven't won, then find the average distance of winners.

Comparing Losers and winners (non-visual)

```
losers=df[df['wins'] == 0]
losers.mean()['avg_distance']
```

```
290.07965492451467
```

The average driving distance of losers on the PGA Tour is about 290 yards. Now let's find the average distance of winners.

```
winners=df[df['wins'] > 0]
winners.mean()['avg_distance']
```

```
294.2583038869257
```

The average driving distance of people who have won on Tour is about 4 yards longer than those who haven't won. Let's look at our 2020 data set to see if it's similar.

```
losers2020=df2020[df2020['FINISHES_TOP10'] == 0]
losers2020.mean()['TEE_AVG_DRIVING_DISTANCE']
```

```
295.03953488372093
```

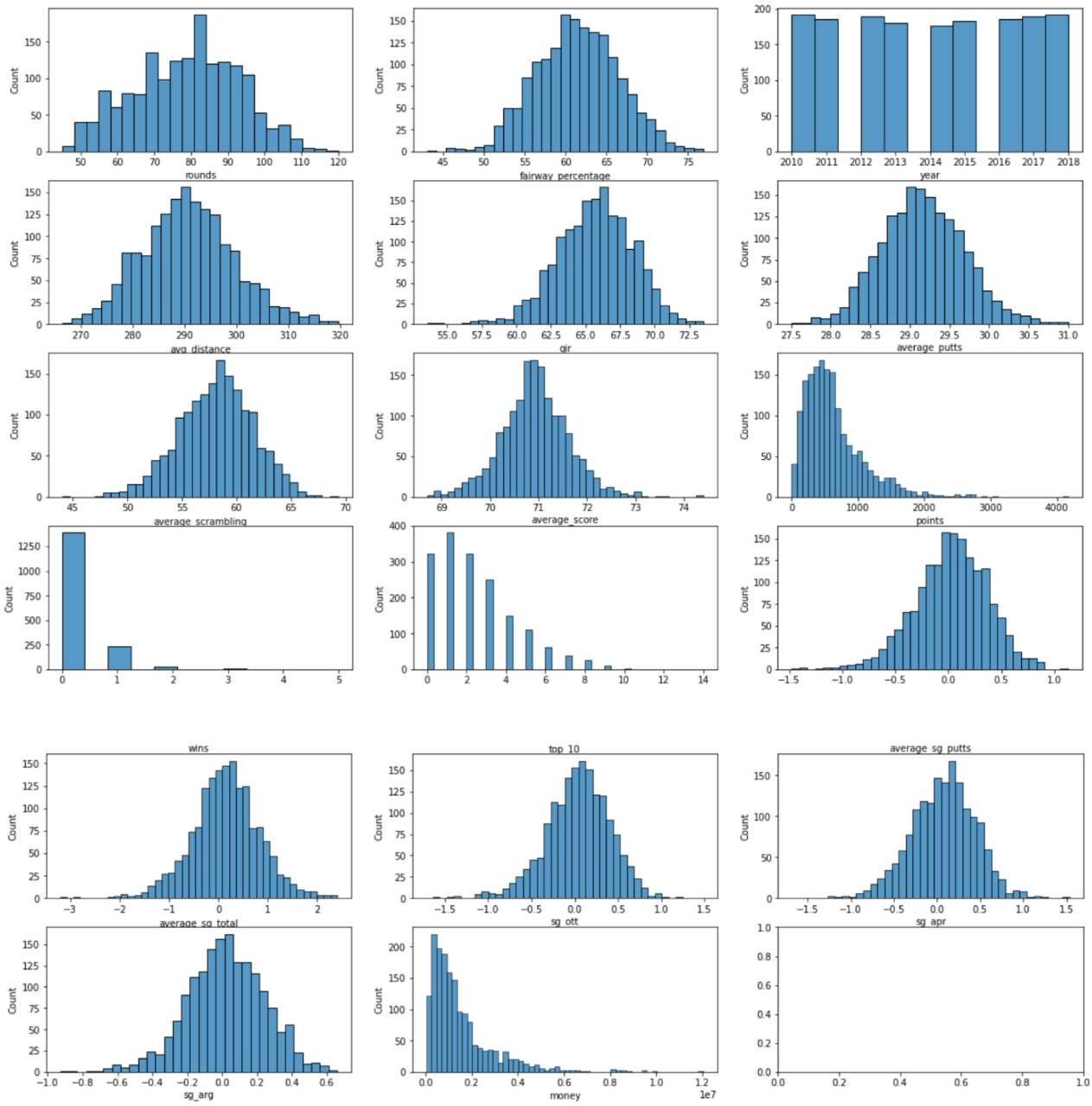
```
winners2020=df2020[df2020['FINISHES_TOP10'] > 0]
winners2020.mean()['TEE_AVG_DRIVING_DISTANCE']
```

```
297.7966666666666
```

Graphing

Data Distribution

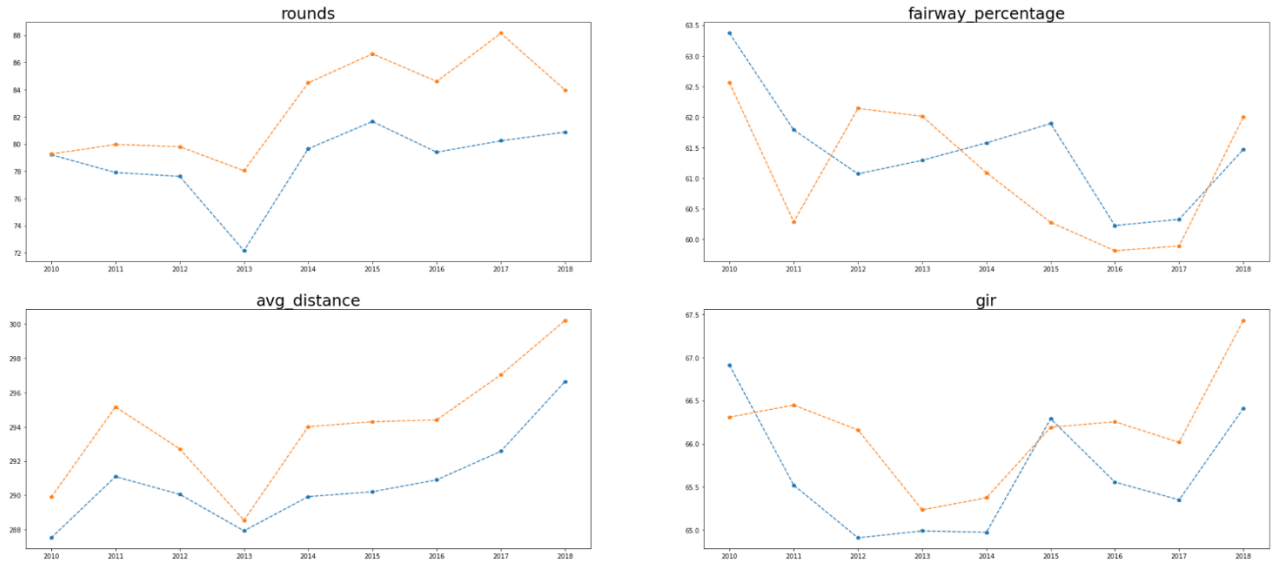
```
f, ax = plt.subplots(nrows = 6, ncols = 3, figsize=(20,20))
distribution = df.loc[:,df.columns!='player_name'].columns
rows = 0
cols = 0
for i, column in enumerate(distribution):
    p = sns.histplot(df[column], ax=ax[rows][cols])
    cols += 1
    if cols == 3:
        cols = 0
        rows += 1
```

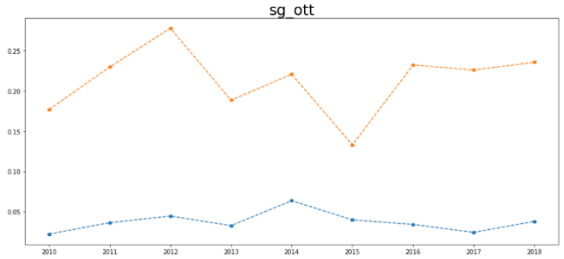


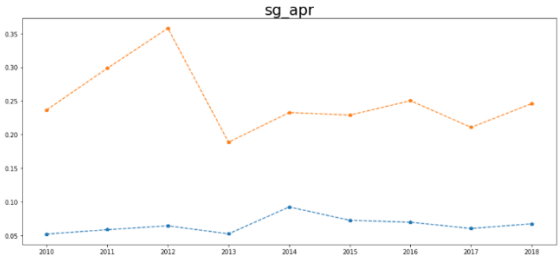
As we can see, most of the graphs are relatively centered, including average distance. Let's approach the concept of "comparing winners to average" with a more visual perspective. Here are graphs that compare the Winners to the Average in each category. The Winners are orange, and the Average is blue.

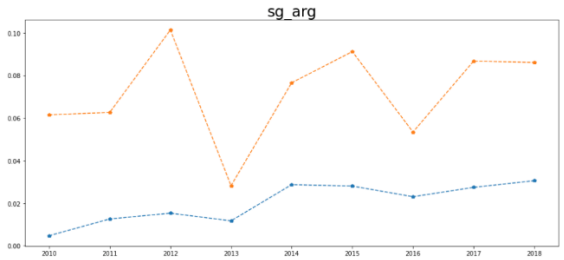
Comparing Average and Winners

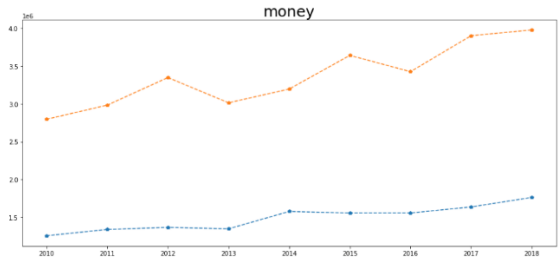
```
winner = df.loc[df['wins'] >= 1, :]

fig, ax = plt.subplots(nrows = 8, ncols = 2, figsize=(35,65))
distribution = df.loc[:,df.columns!='player_name'].columns
distribution = distribution[distribution != 'year']

rows = 0
cols = 0
lower_better = ['average_putts', 'average_score']
for i, column in enumerate(distribution):
    avg = df.groupby('year')[column].mean()
    best = winner.groupby('year')[column].mean()
    ax[rows,cols].plot(avg, 'p--',)
    ax[rows,cols].plot(best, 'p--',)
    ax[rows,cols].set_title(column, fontsize = 25)

    cols += 1
    if cols == 2:
        cols = 0
        rows += 1
```

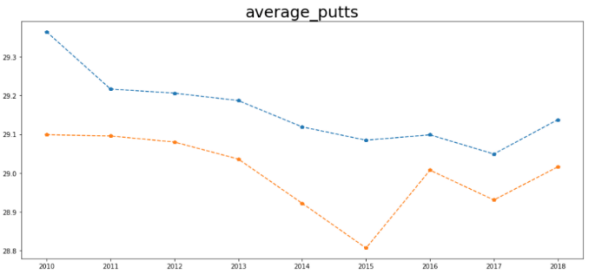


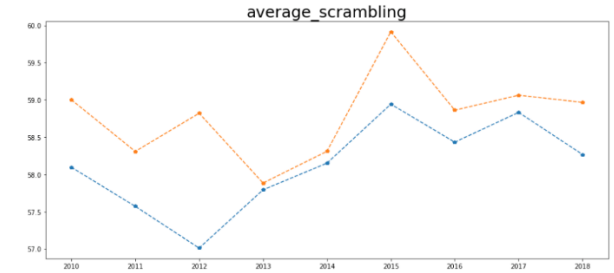

sg_ott

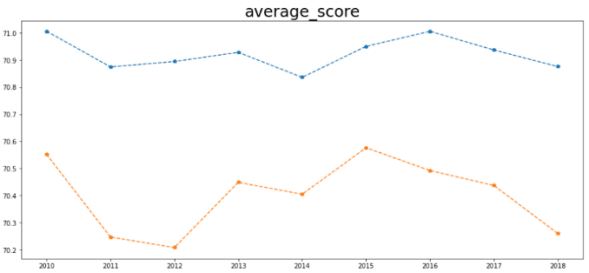

sg_apr

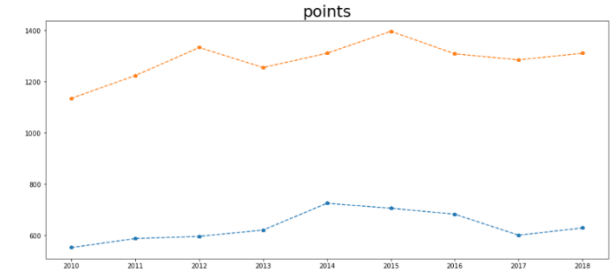

sg_arg

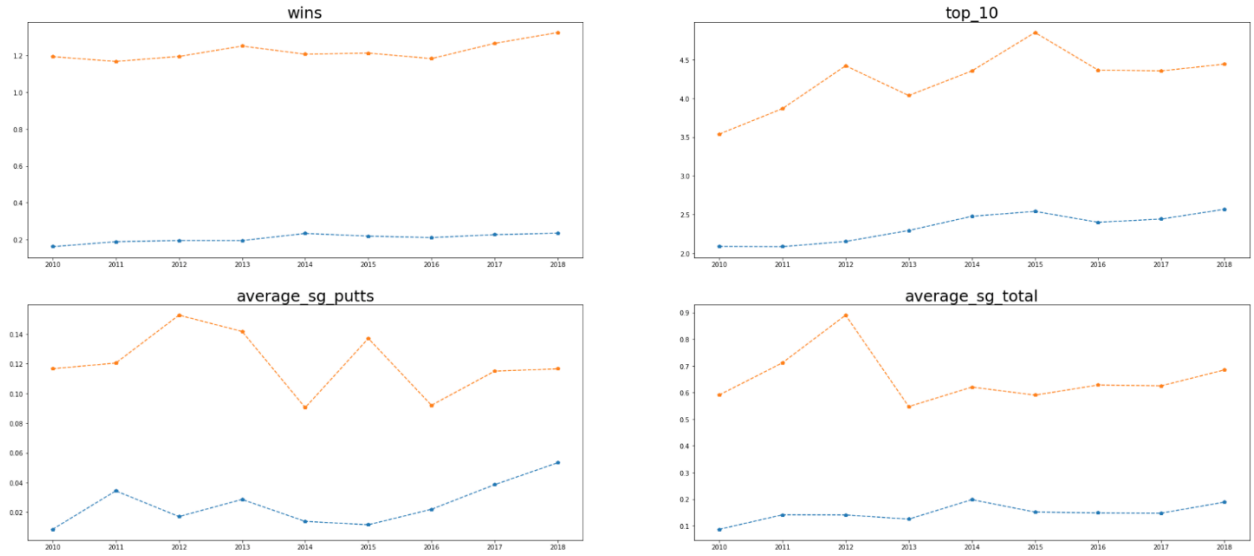

money


average_putts


average_scrambling


average_score


points



Like we saw before, the average distance of winners is higher than the average. Is it by a significant amount? No, I wouldn't say so.

Conclusion

What we learned

From this notebook, I concluded that driving distance does not have as big of a correlation to winnings/earnings as I thought. Based on the various graphs and correlation tables we created, we can see that putting and SG putting are the stats that separate the winners from the losers.