SI 330 Final Project: PGA Tour vs Weather

Motivation

Golf has been a sport of fascination for decades, primarily because of how difficult and how varied performance can be. A professional golfer can go out one day and have the round of his life, while the next day could play as if he had never played golf before. There is no reasoning for this besides that it is the nature of the game. With the many variables affecting golfers, I was interested in the effect weather had on golfers, mainly PGA Tour Players. Specifically, I wanted to see if the temperature for a given day of a tournament would have an effect on players' scores for that day. If it did, this could help predict performance of players based on the weather temperatures they face. My hypothesis is that players would perform better (have lower scores) as the temperature gets warmer/higher.

Data Sources

This project used 3 sources in order to collect all the relevant data; 2 API's to retrieve the golf data as well as a weather website to scrape weather information from. The first part of the project involved deciding which tournaments to use and which years to pull from. I chose tournaments that had the most competitive fields (most popular tournaments) where the location didn't change from year to year. The tournaments were as follows: *The Masters, The Memorial, PLAYERS Championship, Arnold Palmer Invitational. Bridgestone Invitational, Waste Management Open, The Honda Classic, Farmers Insurance Open, Greenbrier Classic, and The John Deere Classic.* This helped create consistency in holding as many variables that could potentially impact a player's score as possible, such as type of course, the players, and weather of the location. I pulled data from these 10 tournaments over the 4 years 2014, 2015, 2016, and 2017. This additionally helped with consistency to have mostly the same players playing across the years. In order to retrieve player data from the API, I had to use a different API to collect the tournament ID for that specific year, which would be used in the following API request to get the corresponding leaderboard data for the tournament of interest.

To create this index, I used API's from Sportradar. I created an account in order to retrieve an API key, which I then used each time I requested the API's. The first API I made use of was their *Tournament Schedule API* (documentation here). Each query would return a large amount of data in JSON format of all tournaments played in a given year, including the name and ID of each tournament, the winning purse, the start data, as well as other information. This API was called 4 times, once for each year to pull the 10 tournament ID's for each year. The second API I used was the *Tournament Leaderboard API* (documentation here). This would return a large amount of data in JSON format. The data was extremely nested containing information about the tournament as well as each player in the tournament such as their: position, total score, round score, birdies, pars, bogeys, and other less meaningful information. This API was called 39 times, one for each tournament each year of analysis. 1 tournament was missing because that tournament was cancelled since the course was being used for another tournament. The data was appended into a list in string format. I then saved all of that data into a JSON file, putting all of the data back into JSON format as it would be too difficult to put straight into a

dataframe with the way each tournament's meta-data was structured. This also allowed for me to stay under my API request limit as that was an issue I had to deal with.

The third source of data I had was from a weather website (http://www.wunderground.com/) where I was able to have the 10 website addresses, one for each tournament, plug in the date I needed the temperature for, call the website, and scrape the mean day temperature using beautifulSoup to get the temperature. The format of that link was as follows: https://www.wunderground.com/history/airport/KDNL/2017/4/12/DailyHistory.html?req_city=&req_state=&req_statename=&reqdb.zip=&reqdb.magic=&reqdb.wmo
I called these addresses a total of 156 times (once for each of the 4 days of each of the 39 tournaments played). In total, I retrieved roughly 15,000 records of data for all the tournaments.

Data Manipulation Methods

In order to properly collect all the data I needed, I had to go through multiple data manipulation steps. In order to retrieve data about a tournament, I needed the unique ID for that tournament which changed every year. I iterated over the years and tournaments I was interested in and pulled the various ID's I needed, applying them to the tournament leaderboard API in order to retrieve data on each tournament. As previously stated, I appended each tournament's meta data to a list, then saving it as a JSON file because of the complex nesting of the data.

After all the data was collected and saved, I opened up the JSON file, reformatted the raw data properly as the way that would be most useful for my analysis and be able to put into a data frame. I would pull the information I needed for each player for each tournament as a dictionary, and then append that dictionary to a list. The resulting data structure was a list of dictionaries, where each dictionary represented one line of data for the csv file/data frame. I formatted the csv file and data frame to have each row contain the players first name, last name, tournament, start date, position, round number, round score, strokes, pars, birdies, bogeys, double bogeys and eagles. Once I formatted the data correctly, I uploaded it to a csv file titled 'tournament_data.csv', which I then used to create my data frame. I used the pd.read_csv() function to create my data frame from the csv file. A portion of the data frame for references is shown below:

	Unnamed: 0	birdies	bogeys	double_bogeys	eagles	first_name	last_name	pars	position	round	round_score	start_date	strokes	tournament
0	0	1	0	0	0	Sergio	Garcia	17	1.0	1	-1	2017-04-06	71	Masters Tournament
1	1	6	3	0	0	Sergio	Garcia	9	1.0	2	-3	2017-04-06	69	Masters Tournament
2	2	4	2	0	0	Sergio	Garcia	12	1.0	3	-2	2017-04-06	70	Masters Tournament
3	3	3	2	0	1	Sergio	Garcia	12	1.0	4	-3	2017-04-06	69	Masters Tournament
4	4	5	4	0	0	Justin	Rose	9	2.0	1	-1	2017-04-06	71	Masters Tournament

In order to get the weather data for each tournament, I created a function titled 'populate_Temp'. The purpose of this function is to take a row of data from the data frame and return the corresponding weather temperature depending on what information was given in

the row. I was able to merge the data into the data frame using a .apply() function. By design, .apply() runs a function that will be applied to each row of the data frame, allowing for this function I derived to properly work. The function would take a row in the data, look up the round, tournament and start date of each row. It would use the round number to decide how to collect the information. Since the start date of each tournament was the first round, I was able to extract the proper format of that date to pass into the weather website for that tournament to pull the weather data from. I used a dictionary where the keys were the names of tournaments and the values were the corresponding websites to be used to find the weather data for. This allowed me to compare the tournament in the given row with the correct website to pull data from.

Since there are 4 days in a given tournament, I had an else statement if the round number wasn't 1, which would calculate the day of the tournament based on the round number and the start date of the tournament using a date function in python. For each day analyzed, I would also see if data for that day was already pulled, as pinging these websites 15,000 times would take hours to do. I would cache all the data into a dictionary called "temps_pulled", where the keys would be the current date being analyzed, and values were the temperature for that day. If the day was not in the dictionary, I would then call the corresponding weather website based on the tournament location and input the date that was necessary, scrape the page for the weather, and ultimately return the average temperature for that day. After running the .apply() function on my data frame, my resulting data frame had an additional column for each row titled "temperature" with the temperature for that given day. The final data frame with weather temperatures was as follows:

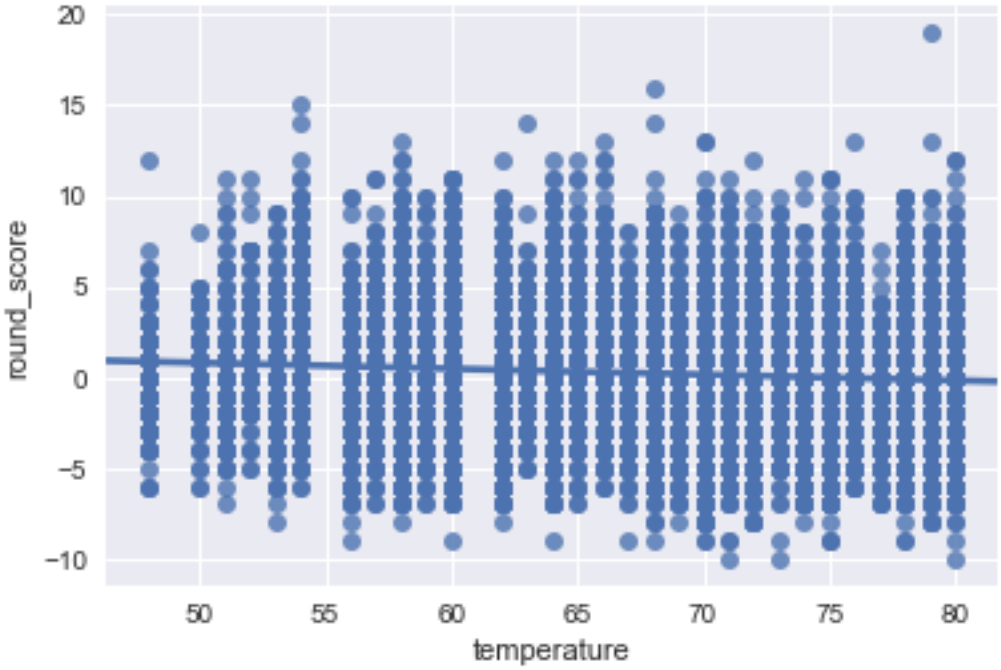
	birdies	bogeys	double_bogeys	eagles	first_name	last_name	pars	position	round	round_score	start_date	strokes	tournament	temperature
0	1	0	0	0	Sergio	Garcia	17	1.0	1	-1	2017-04-06	71	The Masters	59
1	6	3	0	0	Sergio	Garcia	9	1.0	2	-3	2017-04-06	69	The Masters	57
2	4	2	0	0	Sergio	Garcia	12	1.0	3	-2	2017-04-06	70	The Masters	58
3	3	2	0	1	Sergio	Garcia	12	1.0	4	-3	2017-04-06	69	The Masters	63
4	5	4	0	0	Justin	Rose	9	2.0	1	-1	2017-04-06	71	The Masters	59
5	4	4	0	0	Justin	Rose	10	2.0	2	0	2017-04-06	72	The Masters	57
6	7	2	0	0	Justin	Rose	9	2.0	3	-5	2017-04-06	67	The Masters	58

After this, there were many instances where tournament names had changed over the years, so I had to normalize the names of the tournaments by creating a dictionary titled "normalizes_tournament_names" with keys as the desired tournament name and values as potential names the tournament could have been written as. Going through the data frame, I used .replace() to change the names if they were not the desired one, to make sure I could group the data by each tournament. I then dropped columns that were not needed. I uploaded the data into a CSV file named "tournament_weather.csv" which I then called for the rest of the project, removing a column that had been created from uploading the data into the CSV file.

Some of the challenges I encountered was how to get around the API limit, which I was able to do when I figured out how to save the data correctly and format it properly for it to be useful. Another issue I had was to make sure that each API call was spaced out by one second since the API also had a limit of one call per second. I solved this by using a sleep function to have the program stop for a second in between each iteration of the loop. Also, I had to figure out an efficient way to get the weather data. I found 10 locations, 1 for each location and left the date parameters empty in the URL. This allowed me to calculate the date of the tournament, choose the URL for the tournament being analyzed, and attach the date to retrieve the correct weather data, and store that data so I wouldn't have to do that for each of the 15,000 rows of data in my dataset.

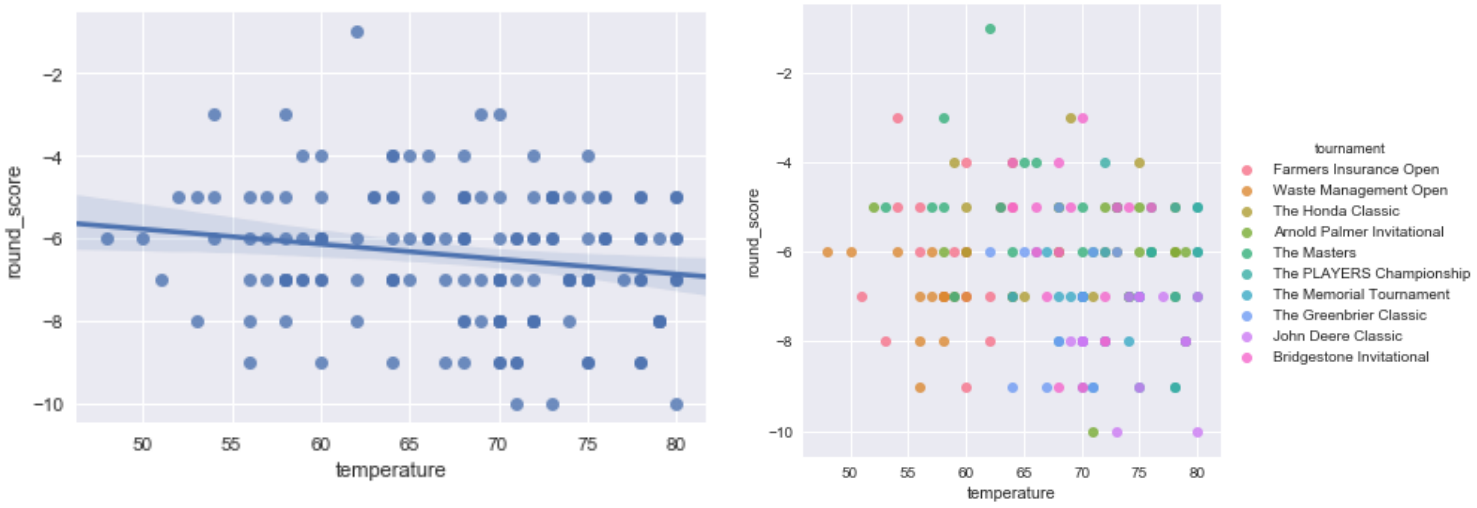
Analysis and Visualization

At first, I created a visualization based on each score in the data set versus the temperature for that day. 15,000 plotted points was too many to show any meaningful information, as shown below:



I then decided to group the data frame by the start date and the round number for each tournament and analyze the minimum round score for that day of a tournament and the weather for that day. Since each tournament has its own start date, I could use this to group by each tournament. I used a groupBy and min function. I used the min function to get the best score for that given day because the minimum score could show the highest potential a player can have for that given temperature. It didn't matter that I was also getting the minimum

temperature for a given day because the weather temperatures for any given day for a tournament were the same throughout the dataset. By having this data, I could see the lowest score (best score) for a day of a tournament versus the temperature for that day of the tournament. I then created 2 seaborne regression plots of those scores versus the temperature. One plot has a regression line highlighting the correlation whereas the other is color coded by each tournament that was being analyzed. Both graphs are shown below:



The graph on the left shows a downward sloping regression between round score and temperature. This means that as the temperature increases, the round score on average decreases. A decreasing round score in golf is better as you want to have the lowest score possible. Although all potential variables aren't being controlled for and the regression has some variation, this did confirm my original thesis that a warmer temperature would allow for players to play better, shooting lower scores. It was evident that weather has an effect on scoring, and also supports golf tournaments being held in warmer locations since fans enjoy watching professional players shoot very low scores.

I will be interested in working with this data in the future, segmenting it in different ways such as by players, and seeing specific player performance compared to the temperature for the given day. I also would like to add in other variables, such as wind, to see what effects that would have on the round scores players shoot.

Overall, this project was extremely fun to do and allowed me to collect information I was generally curious about. I was able to use a lot of new data manipulation techniques I learned from this class, which allowed me to do things I would have previously not had the knowledge of knowing how to do.