Matt Nummy

CS-110

12/13/2020

Final Project Report

For my final project, I decided to take a look at the injury statistics in the five most popular European soccer leagues over the last five seasons. The 2019/20 season was a strange one in Europe because of the unexpected 2-3-month break in the middle of the season due to the Coronavirus pandemic. When games resumed sans-supporters after the break, they had to play until July to finish the remaining matches. Because the various national football associations did not want to start the 2020/21 season too late, they opted for a dramatically reduced offseason and preseason. Less time off to prepare for the current season has led to an almost universal uptick in the frequency of player injuries. Since this is an individual project and I was gathering all the data from the internet manually, I chose to stick with only the top six placed teams from last season in their respective leagues. Even still, this left me with 600 datapoints to use. The only exception to this rule is the Spanish League (La Liga). The second placed Spanish team last season was FC Barcelona, but the website that I used for the statistics did not have a properly functioning page for them. So, I replaced second placed FC Barcelona with seventh placed Granada. At the moment, this program does not have a clean, forward-facing ‘user interface’ but it’s very easy to change function call arguments in the source code to get any output you want. For example, the following code is two functions designed to find which team had the least/most of a certain injury type, how many of that injury type they had, and what season that stat is from. I’ve used the total number of injuries as the focus for this example:

[15]:
```
def who_has_the_most_of(df, inj_type, year=None):
    if year:
        res = df[['Club', year]].loc[(df['Type'] == inj_type)]
        return res.iloc[res[year].argmax()]
    else:
        res = df[
            ['Club', "2016/17", "2017/18", "2018/19",  "2019/20",  "2020/21"]
        ].loc[(df['Type'] == inj_type)]
        rr = pd.concat([res.set_index('Club').idxmax(), res.set_index('Club').max()], axis=1)
        return rr[0].loc[rr[1].idxmax()], rr[1].max(), rr[1].idxmax()

def who_has_the_least_of(df, inj_type, year=None):
    if year:
        res = df[['Club', year]].loc[(df['Type'] == inj_type)]
        return res.iloc[res[year].argmin()]
    else:
        res = df[
            ['Club', "2016/17", "2017/18", "2018/19",  "2019/20",  "2020/21"]
        ].loc[(df['Type'] == inj_type)]
        rr = pd.concat([res.set_index('Club').idxmax(), res.set_index('Club').min()], axis=1)
        return rr[0].loc[rr[1].idxmin()], rr[1].min(), rr[1].idxmin()
```

[30]:
```
who_has_the_most_of(df, 'Injuries', year=None)
```

[30]:
```
('Borussia Dortmund', 59, '2016/17')
```

I expect that the users of my program would be mostly soccer fans that have an interest in statistics or are looking to discover patterns in the injury records of certain teams and leagues for the purposes of sports betting. Additionally, the graphs and charts created by this program turn raw data into a much more digestible format for the average reader. Because of this, I could see these graphs being used by sports journalists in an article discussing the issue of over-training in Europe's top five leagues this season or something similar.

For the collection of my data, I went to transfermarkt.us and found a list of absences for each team in each competition going back a long time. This site told you who was out, for how long and whether the reason for their absence was an injury, a suspension or something else. I went through the data manually and counted the number of injuries in each season for every team. I also counted the number of long-term injuries (ruled out for more than four consecutive

matches), the number of repeat injuries (players injured on more than one occasion in a single season), and the highest number of concurrent injuries (highest number of players from that team that were out injured on a single matchday). I stored all of this data in a spreadsheet. When I was finished collecting all of my data, I had five leagues with six teams from each and four stats that I had tracked for each team over five separate seasons. That left me with exactly 600 datapoints to use in my program. I imported this spreadsheet (tsv) and stored the table as follows:

```
[1]: import pandas as pd
     import matplotlib.pyplot as plt
```

```
[4]: df = pd.read_csv('injury_data.tsv', sep='\t')
     df['2020/21 proj'] = df['2020/21'] * 3
```

```
[5]: df
```

[5]:

	Country	Club	Type	2020/21	2019/20	2018/19	2017/18	2016/17	2020/21 proj
0	Germany	RB Leipzig	Injuries	14	35	26	30	19	42
1	Germany	RB Leipzig	Long-Term	5	11	10	6	5	15
2	Germany	RB Leipzig	Concurrent	5	6	5	4	4	15
3	Germany	RB Leipzig	Repeat	1	7	8	9	5	3
4	Germany	Borussia Dortmund	Injuries	22	40	58	45	59	66
...	...	...	...	...	...	...	...	...	...
115	France	Reims	Repeat	3	1	0	0	0	9
116	France	Rennes	Injuries	13	12	14	4	16	39
117	France	Rennes	Long-Term	8	5	10	3	8	24
118	France	Rennes	Concurrent	5	5	9	3	4	15
119	France	Rennes	Repeat	0	3	2	0	6	0

120 rows × 9 columns

As you can see in the above screenshot, I added a 2020/21 projection column to the table by multiplying the current 2020/21 season statistics by three. I did this because at this point in time, all five of these leagues are just under 1/3 of the way finished with the current season. This is a fairly naïve approach to projecting the 2020/21 end-of-season stats, but under the time constraint, this was much more reasonable than a complicated projection algorithm. After I imported my

table, I wrote individual functions and lines of code that would return interesting subsets of the data from which useful conclusions could be drawn. There are four actual functions in this program. The first, called vs_time_plot, is used to display a line graph that represents all of a certain stat type (injuries, long-terms, concurrents, repeats) for a group of clubs or countries over the five-year time period that I have data for. The code for this function is as follows:

Injuries vs Time

[9]:
```
def vs_time_plot(df, inj_type, clubs=None, countries=None):
    clubs = clubs if clubs else []
    countries = countries if countries else []

    if clubs:
        fltr = df['Club'].isin(clubs)
    elif countries:
        fltr = df['Country'].isin(countries)
    else:
        fltr = ~df.index.isna()

    res = df[
        ['Club', "2016/17", "2017/18", "2018/19",  "2019/20",  "2020/21"]
    ].loc[
        (df['Type'] == inj_type) & fltr
    ].set_index('Club')
    res.T.plot(figsize=(12,8), ylabel=f'{inj_type} vs Time')
```

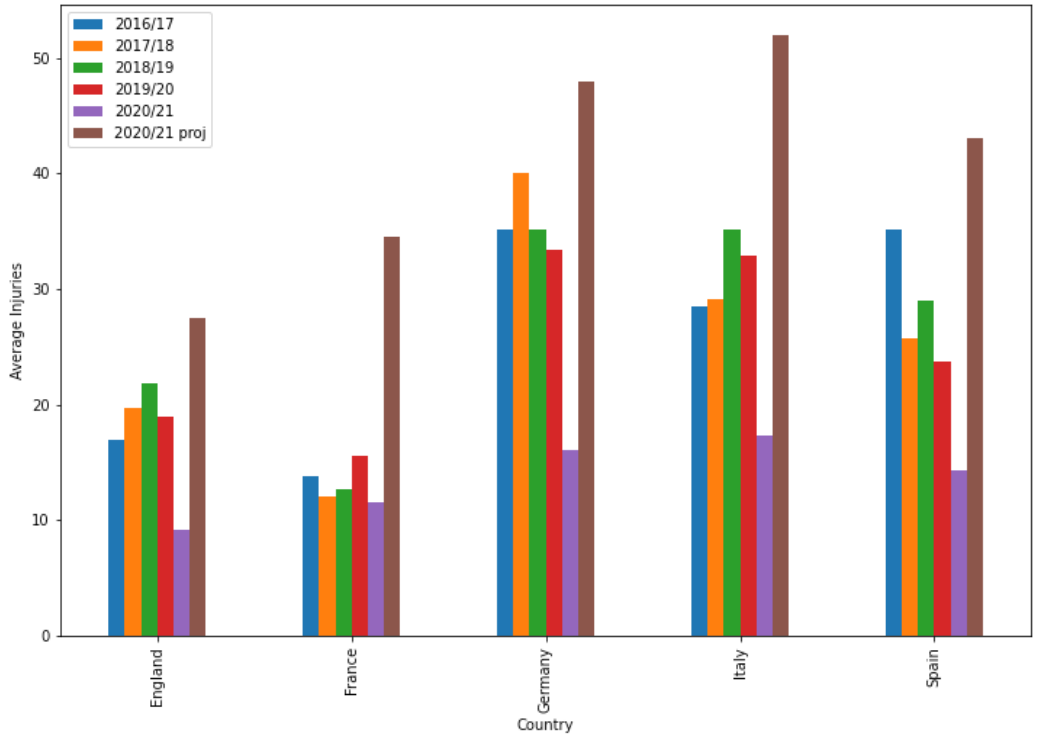
The second function that I wrote was called who_has_the_most_of and it's followed immediately by its counterpart, who_has_the_least_of. These functions will take in a stat type and a year and then return the team with the most/least of that stat type and the number of that stat type they had. If no year is given, they will find the max/min of all five years and return the year that stat was achieved along with the team and stat type. The code for these functions can be seen on page two of this report. My fourth and final function was one called compare_countries and it's really the meat and potatoes of this project. When I had the idea for this project, I thought: "Is there a way to compare the response to the reduced offseason by different European leagues and see which country has handled that reduced pre-season rest and training the most

effectively?" This final function does just that. It can compare the average of any of the four stat categories in all five of the leagues in question across all five seasons for which I've collected data. The code for compare_countries is as follows:

```
[27]: def compare_countries(df, inj_type):
          res = df.loc[(df['Type'] == inj_type)].groupby('Country')[COLS + ['2020/21 proj']].mean()
          res.plot(kind='bar', ylabel=f'Average {inj_type}', figsize=(12,8))
```

When you run this function with the arguments (df, 'Injuries'), you get this graph displaying the average number of injuries in Europe's top five leagues over the last five seasons and it also includes the projection for this year.

```
[18]: compare_countries(df, 'Injuries')
```



With any of these functions, if you change the arguments to a different team, country or stat type, the graph is able to update to reflect that change.

The biggest challenge that I encountered was learning how to use pandas and Jupyter to actually code this project. I knew that I wanted to be doing some sort of data manipulation on a larger scale than I've done in the past, so I started researching the best ways to display data sets as graphs and charts when that data is coming from a tsv. I settled on pandas, but it took me about a week of watching and reading tutorials to finally have a good enough understanding of the new tools at my disposal to make use of them. Once I had cleared that hurdle, it was pretty much smooth sailing. Obviously, the data collection was a long and arduous task, but not very difficult in the grand scheme of things.

The first thing I would do to extend this project would be to expand the dataset to include all of the teams in my five selected leagues instead of only the top 6 teams. I initially wanted to have all of these teams available for this project, but the data collection just took too much time to do on my own. Also, I could collect more historical data for seasons preceding the 2016/17 season. Additionally, a real user interface with a menu of options to choose from would be an improvement that I'd like to make. It would be tricky to do because there are a lot of different datasets to show and in different ways. I'd have to consider the best ways to create a menu that is concise and covers all the possible graph displays but doesn't require previous knowledge of soccer team names and such. The last thing that I'd like to update is the data collection method. It's extremely tedious to fill a spreadsheet manually with hundreds of data points so writing code to scrape the data I need from the web would certainly be a worthwhile upgrade on this program. This would also allow me to keep the spreadsheet permanently up to date. Overall, I had a lot of fun working on this project. It's rare that I get to do a computer science assignment where I'm working with data that I'm genuinely interested in.