Demand of Football in European Leagues

DATA 450 Capstone

Matthew Wilcox

# 1. Introduction

Football (Soccer) is the most populat sport in the world. With an estimated 3.5 billion fans world wide it is the most viewed sport in the word [https://sportforbusiness.com/the-worlds-most-watched-sports/]. Being such a popular sport, there is a large business and economy surrounding the sport. Football teams need to be profitable to succeed. These teams have 5 primary revenue sources. They are television money, prize money, player transfers, sponsorships, and matchday revenues[https://www.football-stadiums.co.uk/articles/how-do-football-clubs-make-money/]. Out of all of these one of the most universial is matchday revenues. Television, prize money, player transfers, and sponsorships can all vary based on what level the team is at. Matchday revenues are much more applicable at all levels of the game. Match day revenues are the profits a team makes from people attending the game. This is from ticket, consessions, and merchandise sales from attending a game. For many teams this is the lifeblood of the club and what allows for the club to survie.

The most important factor within match day revenues is the attendnace. The amount of people that will attend a given match will greatly affect the match day revenues. So understanding factors and being able to predict the attendance for a given match is so important. If a club was able to predict the amount of people that will attend a given match, they could be better prepared for an individual match. Additionally if a predicted match was predicted to have lower attendnace then desired from the club, the club could market it differently or have special promotions in an attempt to increase the attendance for that match.

What has been done here is an evaluation of factors that may impact the attendance for an individual match. These factors are the day/time of the match, betting odds for a match, and whom the away team is for any given match. Additionally, in the end a random forest model was produced to predict attendnace of matches based on these factors.

import seaborn as sns  
from sklearn.ensemble import RandomForestRegressor  
import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
from sklearn.preprocessing import LabelEncoder  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import mean\_squared\_error  
from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report  
from sklearn import metrics  
from sklearn.linear\_model import LinearRegression  
import july  
from datetime import datetime as dt  
from jupyter\_dash import JupyterDash  
from dash import html, dcc, Input, Output  
import plotly.graph\_objects as go  
import plotly.express as px

# 2. Data Collection

The data used in this project was collected from two sources: worldfootball.net and Football-data.co.uk. The data that was collected from worldfootball.net was information on the match such as the names of the teams, the time and date of the match, and most importantly the attendnace for an individual match. This data was not already tabulated so it was scraped from the website. This scrape occured on January 31st 2023. The data that was collected from Football-data.co.uk was primarily betting information for each game. This data was already tabulated into csv files however they were divided based upon the year and league. All the files were downloaded on Febuary 3rd 2023.

The data that was collected was between 2010 and 2023. It consisted of leagues from 11 countries being England, Scotland, Germany, Italy, Spain, France, Netherlands, Belgium, Portugal, Turkey, and Greece. From these Countries 21 leagues of data was collected.

# 3. Data Processing

All of the csv file’s from Football-data.co.uk were combined togther resulting in two datasets. One with all of the betting data and the other with the attendance. These two datasets were to be combined on the home team name, away team name, and the date/time of the game. However, the two data sets had differnet naming structures for team names. For example, the team Manchester City in one dataset would be identified as “man\_city” and in the other as “manchester\_city”. With this differenting structures and spellings of team names, a list of team names was created for both datasets. These lists were than put through a script that took a team from one list and compared the characters to values in the other list. This was starting with one entire team and slowly decreasing its size and observign that through the other team list.

| Itteration | Results |
| --- | --- |
| 1 | man\_city |
| 2 | man\_cit, an\_city |
| 3 | man\_ci, an\_cit, n\_city |
| 4 | man\_c, an\_ci, n\_cit, \_city |

This is an example of how it would divide the name of one list up. It would do this until the team list was broken up into single characters. Then It would start with the largest length of a team name and look through the other list for any results, It would then proceed through the remaining potential names. What resulted is a list of potential matching teams with the team with the most similar name at the top. Then I would determine from the suggestion what was the matching team name. Now that the two data sets had a matching naming of home and away teams, the data sets were able to combine. The resulting data set had 79673 rows and 172 columns where each row was an individual match.

However more processing was needed. Although the intial dataset collected data all the way to 2023, the range of the data was filterd to only 2010 to 2019. This was to attribute to the COVID-19 pandemic. During the pandemic attendance basically ceased to occur for matches. Additionally some leagues cancelled the remaining matches for the season. For those reasons the dataset is focused up untill that time.

Certain leagues were removed from the dataset for analysis . The Scotish Divsiion 2 and Division 3 leagues as well as the Ethniki Katigoria, which is the Greek top league, were removed. This is due to them having several missing values for many variables. Some matches from a variety of leagues had missing values for only betting variables. These matches within leagues were used during the analysis of day/time and the impact of the away team, however, were dropped from the dataset for analysis betting data and in the modeling.

Lastly a few new variables were added. The first variable added was the Season the match occured. Although leagues end on different dates on different years the date selected for the season to switch was July 14th. Most leagues conclude in the begining of June and start back in the begining fo August. July is predominantly used for international games. Although there were a couple of matches that occured in July from 2010-2019, July 14th was the only date with zero matches played. So it was used as the cutoff point.

The other variables were the mean and standard deviaiton of the home team for that season and the z-score of that individual match. The mean and standard deviation were just used to create to create the z-score variable. The z-score is the standardization of the mathces attendance in relation to the home team’s average attendance for that particular season.

## 3.1 Final Dataset

The resulting dataset that was used consisted fo 53,224 rowss and 34 columns. Here is a list of the variables used as well as their description:

| Variable | Description |
| --- | --- |
| home\_team | Name of Home team for na individual match |
| away\_team | Name of Away team for an individual match |
| raw\_attendance | Total number of people who attended an individual match |
| division | The league the match took place in |
| B365H | Bet365 home team win odds |
| B365D | Bet 365 draw odds |
| B365A | Bet365 away team win odds |
| BWH | Bet&Win home team win odds |
| BWD | Bet&Win draw odds |
| BWA | Bet&Win away team odds |
| WHH | William Hilll Home win Odds |
| WHD | William Hill Draw odds |
| WHA | William Hill Away win odds |
| VCH | VC Bet Home team win odds |
| VCD | VC bet draw odds |
| VCA | VC Bet away win odds |
| BbAv>2.5 | Bet Brain Average over 2.5 goals |
| BbAV<2.5 | Bet Brain Average under 2.5 goals |
| date\_time | Date and time of when a match occured |
| season | The season a match occured |
| mean\_attend | Average home team attednace for that season |
| std\_attend | Standard deviation of the home team attendance that season |
| Standard\_attendance | The z-score of attendance for a match in relation to the home team attednacne that season |

total\_data = pd.read\_pickle('../data/final\_datasets/data\_standardized.pkl')  
total\_data.head()

|  | home\_team | away\_team | home\_score | away\_score | date | time | day\_of\_week | raw\_attendance | stadium | city | ... | BbMx>2.5 | BbAv>2.5 | BbMx<2.5 | BbAv<2.5 | capacity\_filled | date\_time | season | mean\_attend | std\_attend | standard\_attend |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | tottenham\_hotspur | manchester\_city | 0 | 0 | 2010-08-14 | 12:45 | Saturday | 35928 | Stadium News | London | ... | 2.03 | 1.91 | 1.95 | 1.84 | 0.990189 | 2010-08-14 12:45:00 | 2011 | 35892.894737 | 269.766338 | 0.130132 |
| 1 | tottenham\_hotspur | wigan\_athletic | 0 | 1 | 2010-08-28 | 15:00 | Saturday | 35101 | Stadium News | London | ... | 1.55 | 1.50 | 2.63 | 2.48 | 0.967396 | 2010-08-28 15:00:00 | 2011 | 35892.894737 | 269.766338 | -2.935484 |
| 2 | tottenham\_hotspur | wolverhampton\_wanderers | 3 | 1 | 2010-09-18 | 15:00 | Saturday | 35940 | Stadium News | London | ... | 1.85 | 1.75 | 2.11 | 2.02 | 0.990519 | 2010-09-18 15:00:00 | 2011 | 35892.894737 | 269.766338 | 0.174615 |
| 3 | tottenham\_hotspur | everton\_fc | 1 | 1 | 2010-10-23 | 12:45 | Saturday | 35967 | Stadium News | London | ... | 2.07 | 1.99 | 1.87 | 1.79 | 0.991263 | 2010-10-23 12:45:00 | 2011 | 35892.894737 | 269.766338 | 0.274702 |
| 4 | tottenham\_hotspur | sunderland\_afc | 1 | 1 | 2010-09-11 | 20:00 | Tuesday | 35843 | Stadium News | London | ... | 1.90 | 1.80 | 2.06 | 1.97 | 0.987846 | 2010-09-11 20:00:00 | 2011 | 35892.894737 | 269.766338 | -0.184955 |

# 4. Date & Time

The first factor that will be evaluated is the date and time of individual matches. There are multiple attributes to this that will be viewed, from the day of the week, calendar date, and time of the match.

time\_df = total\_data[[  
 'date', 'time', 'day\_of\_week', 'date\_time', 'raw\_attendance', 'capacity\_filled', 'standard\_attend', 'division'  
]]  
  
  
df\_grouped\_mean = time\_df.groupby('day\_of\_week')['raw\_attendance', 'capacity\_filled', 'standard\_attend'].mean().reset\_index()  
df\_grouped\_median = time\_df.groupby('day\_of\_week')['raw\_attendance', 'capacity\_filled', 'standard\_attend'].median().reset\_index()  
  
day\_categories = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']  
df\_grouped\_median['day\_of\_week'] = pd.Categorical(df\_grouped\_median['day\_of\_week'], categories= day\_categories)  
df\_grouped\_median.sort\_values(by = 'day\_of\_week', inplace = True)

## 4.1 Day of Week

First in [Figure 1](#fig-day_median_attend), the average attendance for matches are being viewed by the day of the week. What is significant here is that it appears that wednesday and sunday has the highest average attendance while Tuesday has the lowest. Wednesday being having the highest average attendance is striking. Most would expect games on the weekday games may have struggle to have high attendance

sns.barplot(data=df\_grouped\_median, x = 'day\_of\_week', y = 'raw\_attendance').set(title ='Median Attendance by Day of Week')  
plt.xticks(rotation=90)  
plt.xlabel('Day of the Week')  
plt.ylabel('Median Attendance')  
plt.show()

|  |
| --- |
| Figure 1: Median Attendnace by day of the week |

[Figure 1](#fig-day_median_attend) graph had saturday lower than expected average attendnace with wednesday havign the most games attended on average due to the distribution of number of games seen in [Figure 2](#fig-count_day). Saturday has by far the most amount of games in comparjision to any day of the week. This results in Saturday having more lower attend games from lower leagues. This is confirmed by [Figure 3](#fig-day_median_attend_div). Looking ath the amount of games by leagues on saturday, it had most of the lower leagues hosting matches on Saturday then other days of the week. Wednesday has the second fewest games played however looking again at [Figure 3](#fig-day_median_attend_div), it was predominantly composed of the top leagues in England, France, and Italy. The top leagues on average have greater attended games hence Wednesday on average has the greatest attendance on average.

grouped\_week\_count = time\_df.groupby('day\_of\_week').count().reset\_index()  
  
grouped\_week\_count['day\_of\_week'] = pd.Categorical(grouped\_week\_count['day\_of\_week'], categories= day\_categories)  
grouped\_week\_count.sort\_values(by = 'day\_of\_week', inplace = True)  
  
  
sns.barplot(data = grouped\_week\_count, x = 'day\_of\_week', y = 'date')  
plt.xlabel('Day of the Week')  
plt.ylabel('Count')  
plt.title('Number of games per day of the Week')  
plt.show()

|  |
| --- |
| Figure 2: Number of games per day of the week |

|  |  |  |
| --- | --- | --- |
| grouped\_week\_count\_division = time\_df.groupby(['day\_of\_week', 'division']).count().reset\_index()  grouped\_week\_count\_division['day\_of\_week'] = pd.Categorical(grouped\_week\_count\_division['day\_of\_week'], categories= day\_categories) grouped\_week\_count\_division.sort\_values(by = 'day\_of\_week', inplace = True)  fig = px.bar(grouped\_week\_count\_division, y = 'day\_of\_week', x = 'date', color = 'division', barmode = 'group') fig.show()   |  | | --- | | Unable to display output for mime type(s): text/html  (a) Number of games per day of week divided by League |  |  | | --- | | Unable to display output for mime type(s): text/html  (b) **?(caption)** |   Figure 3: **?(caption)** |

## 4.2 Time of Day

df\_grouped\_mean\_tod= time\_df.groupby(time\_df['date\_time'].dt.hour).mean()  
df\_grouped\_median\_tod= time\_df.groupby(time\_df['date\_time'].dt.hour).median()  
  
  
# sns.lineplot(data = df\_grouped\_median\_tod, x = 'date\_time', y = 'raw\_attendance', markers = True, marker = "o" )  
# plt.title('Attendance by Time of Day')  
# plt.xlabel('Hour of the Day')  
# plt.ylabel('Attendance')  
# plt.show()

Time and day of a match resulted in a similar situation as with day of the week. In [Figure 4](#fig-attendance_time) there was sdips in attendance at 3pm and 9pm however these were also the most attended times for matches. 12pm and 11pm matches had the highest attendance on average however had lower amount of games.

df\_grouped\_count = time\_df.groupby(time\_df['date\_time'].dt.hour).count()  
# print(df\_grouped\_count)  
df\_grouped\_count = df\_grouped\_count['raw\_attendance'].reset\_index()  
df\_grouped\_count['count'] = df\_grouped\_count['raw\_attendance']  
df\_grouped\_count = df\_grouped\_count[['date\_time', 'count']]  
# df\_grouped\_count= df\_grouped\_count.rename(columns = {'date':'count'})  
# print(df\_grouped\_count)  
  
df\_count\_atted = pd.merge(df\_grouped\_count, df\_grouped\_median\_tod, on = 'date\_time')  
df\_count\_atted = df\_count\_atted.drop(columns= ['capacity\_filled'])  
df\_count\_atted.rename( columns = {'raw\_attendance': 'Attendance', 'count': "Number of Games"}, inplace= True)  
# print(df\_count\_atted)  
melted\_count\_attend = pd.melt(df\_count\_atted, value\_vars=['Number of Games', 'Attendance'], id\_vars= 'date\_time')  
# print(melted\_count\_attend)  
  
sns.lineplot(data = melted\_count\_attend, x = 'date\_time', y = 'value', hue = 'variable')  
plt.title('Attendance and Game Count by Time of Day')  
plt.xlabel('Hour of the Day')  
  
plt.show()

|  |
| --- |
| Figure 4: Attendance and Match count by time of day |

## 4.3 Calendar Date

The last aspect of when the match occured evaulated was what day of the year a match occured. This is viewing trends during the calendar year to see if there is any insight.

calendar\_plot\_data = total\_data  
calendar\_plot\_data['month\_day'] = calendar\_plot\_data['date\_time'].dt.strftime('%m-%d')

## 4.4 Trends in Quanity of Matches

First before looking at the attendance its important to look and view trends in when matches actually occur. In [Figure 5](#fig-cal_count), there is a significantly low amount of games played from mid June to the end of July. This can most likely be attributed to Europeans leagues predominantly being on break during these months in relation to FIFA internation break. On this break players typically return to their national teams to play in international competitions and every four years the world cup. Another significant period of time for number of games is the Christmas holiday. Christmas has one of the lowest total number of games occurring. However, December 26th or Boxing Day had the most amount of games played for any day.

date\_of\_year = calendar\_plot\_data.groupby('month\_day').count()  
date\_of\_year['count'] = date\_of\_year['standard\_attend']  
date\_of\_year = date\_of\_year['count'].sort\_values().reset\_index()  
# print(date\_of\_year)  
date\_of\_year['total\_date'] = '2024-' + date\_of\_year['month\_day']  
date\_of\_year['total\_date']= pd.to\_datetime(date\_of\_year['total\_date'], format = "%Y-%m-%d")  
events = pd.Series(date\_of\_year['count'].values.tolist(), index = date\_of\_year['total\_date'].values.tolist())  
july.heatmap(dates = date\_of\_year['total\_date'], data = date\_of\_year['count'], date\_label = True, cmap = 'RdYlBu', fontsize =10, weekday\_label=False, year\_label= False, title = '# of Games per day 2010-2019', colorbar= True, dpi =1200)  
plt.show()

|  |
| --- |
| Figure 5: **?(caption)** |

## 4.5 Trends in Attedance of matches.

Now looking at the attendance for these matches based on their calendar date some trends appear. First their is a grouping of higher than average attendad matches in May. This can most likely be attributed to the seasons across Europe are finishing. In turn these matches have higher weight to them due to the ramifications of them. For teams at the top and bottom of the standings, these matches have massive implications for the club. They can result in the team being promoted (moved up a league) or relegated (moved down a league) and access to European competition. Additionally with holidays there is an affect on attendance. First November 6th which is All Saint’s day saw a decrease in attendance on average. Christmas had lower than average attedance while Boxing Day to New years eve had greater Attendance then what is seen in December and January.

attend\_date = calendar\_plot\_data.groupby('month\_day').mean()  
attend\_date['count'] = attend\_date['standard\_attend']  
attend\_date = attend\_date['count'].sort\_values().reset\_index()  
attend\_date['total\_date'] = '2024-' + attend\_date['month\_day']  
  
attend\_date['total\_date'] = pd.to\_datetime(attend\_date['total\_date'], format = "%Y-%m-%d")  
events = pd.Series(attend\_date['count'].values.tolist(), index = attend\_date['total\_date'].values.tolist())  
july.heatmap(dates = attend\_date['total\_date'], data = attend\_date['count'], cmap='RdYlBu', date\_label = True, fontsize =10, weekday\_label=False, year\_label= False, title = 'Avg Attendance per day Standardized 2010-2019', colorbar= True, dpi =1200)  
plt.show()

|  |
| --- |
| Figure 6: **?(caption)** |

# 5. Away Team Impact

The next factor viewed is the impact of the away team. Here we are looking at the awayteam

away\_team\_impact = total\_data.groupby(['away\_team', 'division'])['standard\_attend'].mean().reset\_index()  
  
  
# away\_team\_impact = away\_team\_impact[away\_team\_impact['division'].isin(['E3', 'E1', 'E2'])]  
div\_dict = {'D1':'Bundesliga', 'D2': '2. Bundesliga', 'E0':'Premier League', 'E1':'Championship',   
 'E2':'League 1', 'E3':'Leauge 2','SP1':'La Liga Primera', 'SP2':'La Liga Segunda',  
 'B1':'Jupiler League', 'F1':'Ligue 1','F2':'Ligue 2','I1':'Serie A','I2':'Seire B',   
 'SC0':'Scotish Premier League', 'SC1':'Scotish Division 1', 'T1':'Fubol Ligi 1', 'P1': 'Liga 1'}  
divisions\_list =['D1', 'D2', 'E0', 'E1', 'E2', 'E3', 'SP1' ,'SP2', 'B1', 'F1', 'F2', 'I1', 'I2', 'SC0', 'SC1', 'T1', 'P1']  
  
away\_team\_impact = away\_team\_impact[['away\_team', 'division', 'standard\_attend']]  
# print(away\_team\_impact)  
initial\_graph\_df = pd.DataFrame(columns = ['away\_team', 'division', 'standard\_attend'])  
  
for i in divisions\_list:  
 temp\_impact\_df = away\_team\_impact[away\_team\_impact['division'] == i].sort\_values('standard\_attend',ascending = False).head(3)  
 initial\_graph\_df = pd.concat([initial\_graph\_df, temp\_impact\_df], axis = 0)  
  
  
# fig = go.Figure(px.bar(away\_team\_impact, y= 'away\_team', x = 'standard\_attend', color = 'division'))  
  
app = JupyterDash(\_\_name\_\_)  
app.layout = html.Div(id = 'parent', children = [  
 html.H1(id = 'H1', children = 'Away Team Impact'),  
 dcc.Slider(0,20,1, value =3,id = 'slider'),  
 dcc.Dropdown(id = 'dropdown',   
 options = [  
 {'label': 'Bundesliga', 'value':'D1'},  
 {'label': '2. Bundesliga', 'value':'D2'},  
 {'label': 'Premier League', 'value':'E0'},  
 {'label': 'Championship', 'value':'E1'},  
 {'label': 'League 1', 'value':'E2'},  
 {'label': 'Leauge 2', 'value':'E3'},  
 {'label': 'La Liga Primera', 'value':'SP1'},  
 {'label': 'La Liga Segunda', 'value':'SP2'},  
 {'label': 'Jupiler League', 'value':'B1'},  
 {'label': 'Ligue 1', 'value':'F1'},  
 {'label': 'Ligue 2', 'value':'F2'},  
 {'label': 'Serie A', 'value':'I1'},  
 {'label': 'Serie B', 'value':'I2'},  
 {'label': 'Scotish Premier League', 'value':'SC0'},  
 {'label': 'Scotish Division 1', 'value':'SC1'},  
 {'label': 'Fubol Ligi 1', 'value':'T1'},  
 {'label': 'Liga 1', 'value':'P1'}  
  
  
 ], value = ['D1', 'D2', 'E0', 'E1', 'E2', 'E3', 'SP1' ,'SP2', 'B1', 'F1', 'F2', 'I1', 'I2', 'SC0', 'SC1', 'T1', 'P1'],  
 multi = True),  
 dcc.Graph(id = 'bar\_plot', figure=px.bar(initial\_graph\_df, x='away\_team', y='standard\_attend', color='division'))  
])  
  
@app.callback(  
 Output("bar\_plot", "figure"),   
 [Input("dropdown", "value"),  
 Input('slider', 'value')]  
 )  
def update\_graph(drop\_value, slider\_value):  
 # print(value)  
 df = away\_team\_impact  
  
   
  
 df = df[df['division'].isin(list(drop\_value))]  
 graph\_df = pd.DataFrame(columns = ['away\_team', 'division', 'standard\_attend'])  
 for i in drop\_value:  
 temp\_impact\_df = df[df['division'] == i].sort\_values('standard\_attend',ascending = False).head(slider\_value)  
 graph\_df = pd.concat([graph\_df, temp\_impact\_df], axis = 0)  
 graph\_df = graph\_df.reset\_index().drop(columns = ['index'])  
 fig = px.bar(graph\_df, x= 'away\_team', y= 'standard\_attend', color = 'division')  
 return fig  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run\_server(mode='inline')

Dash is running on http://127.0.0.1:8050/

<IPython.lib.display.IFrame at 0x1736418f490>