



# Python Project: Football Score Prediction through Machine Learning



Gabriel Oppermann (Gp04397) and Alejandro Häfliger (Alejandro)



# Structure

1. General description of the Project
2. How to use the App
3. The Programming behind the App

# 1. General description of the Project

Our application is based on linear regression models on historic data. The goal is to analyze correlations between the odds, that are released by Betting Houses before the game, to predict the outcome of the match.

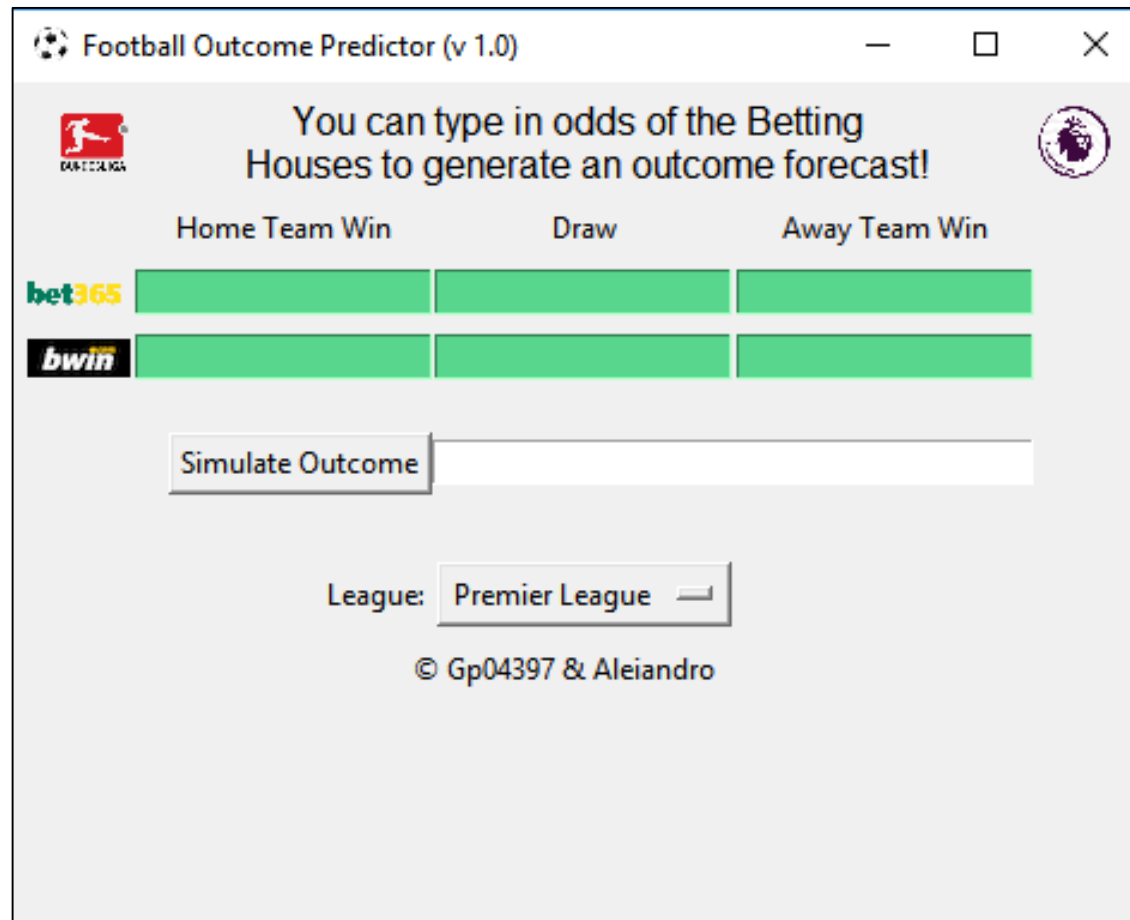
To achieve that, our application **downloads** the relevant data from **the three most recent seasons** (this is done dynamically: always current + two past seasons) automatically from an online datasource, prepares it for data analysis and performs linear regression models on the data.

Afterwards, the user can interact with the app, by typing in the odds that are currently available for future games, to receive a prediction of the game outcome. This predicted outcome is based on the analysis of odds and game outcomes of the last three seasons.

Because the online data base, that our application utilizes, is updated after every matchday, the automated prediction model also gets updated with the newest data.

**Important Note:** For the Code to run successfully, open the .exe file or the .ipyng file only if it WITHIN the same folder as the icons (.png and .ico files are stored in same .zip location)

## 2. How to use the App



The screenshot shows a Windows application titled "Football Outcome Predictor (v 1.0)". The interface includes a header with a soccer ball icon and the text "You can type in odds of the Betting Houses to generate an outcome forecast!". Below this, there are three columns labeled "Home Team Win", "Draw", and "Away Team Win". Two rows of green input fields are provided, with the first row labeled "bet365" and the second row labeled "bwin". A "Simulate Outcome" button is located below the input fields. At the bottom, there is a "League:" label followed by a dropdown menu currently set to "Premier League". The copyright notice "© Gp04397 & Aleiandro" is displayed at the very bottom.

	Home Team Win	Draw	Away Team Win
bet365			
bwin			

Simulate Outcome

League: Premier League

© Gp04397 & Aleiandro

### Explanatory notes:

In the app, you can insert the odds into the green cells, according to the home win/draw/away win odds displayed by both featured betting houses. Additionally, on the bottom one has to select if the game is taking place in the German Bundesliga or in the English Premier League.

The next slide will show a concrete example.

**Important:** Please keep in mind that the app might take a few seconds to open since it is a large .exe file.

## 2. How to use the App

The screenshot shows the 'Football Outcome Predictor (v 1.0)' application window. It features a table for inputting odds from different betting houses. The table has three columns: 'Home Team Win', 'Draw', and 'Away Team Win'. Two rows are shown, one for 'bet365' and one for 'bwin'. The odds for bet365 are 3.8, 3.5, and 1.95 respectively. The odds for bwin are 4.0, 3.5, and 1.95. Below the table is a 'Simulate Outcome' button and a text box displaying 'Draw is expected'. At the bottom, there is a 'League' dropdown menu set to 'Premier League' and a copyright notice '© Gp04397 & Aleiandro'.

	Home Team Win	Draw	Away Team Win
bet365	3.8	3.5	1.95
bwin	4.0	3.5	1.95

Simulate Outcome: Draw is expected

League: Premier League

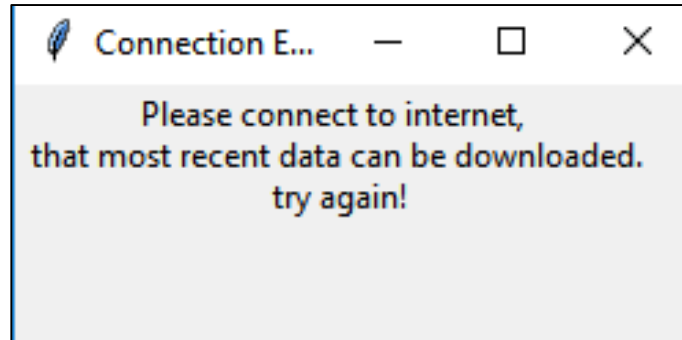
© Gp04397 & Aleiandro

### Explanatory notes:

First we browse the web to **get the odds displayed by the betting houses Bet365 and bwin.**

We then simply **insert them into the app** and select „Premier League“ in this case. After **clicking „Simulate Outcome“**, we see that the expected outcome is a draw between both teams.

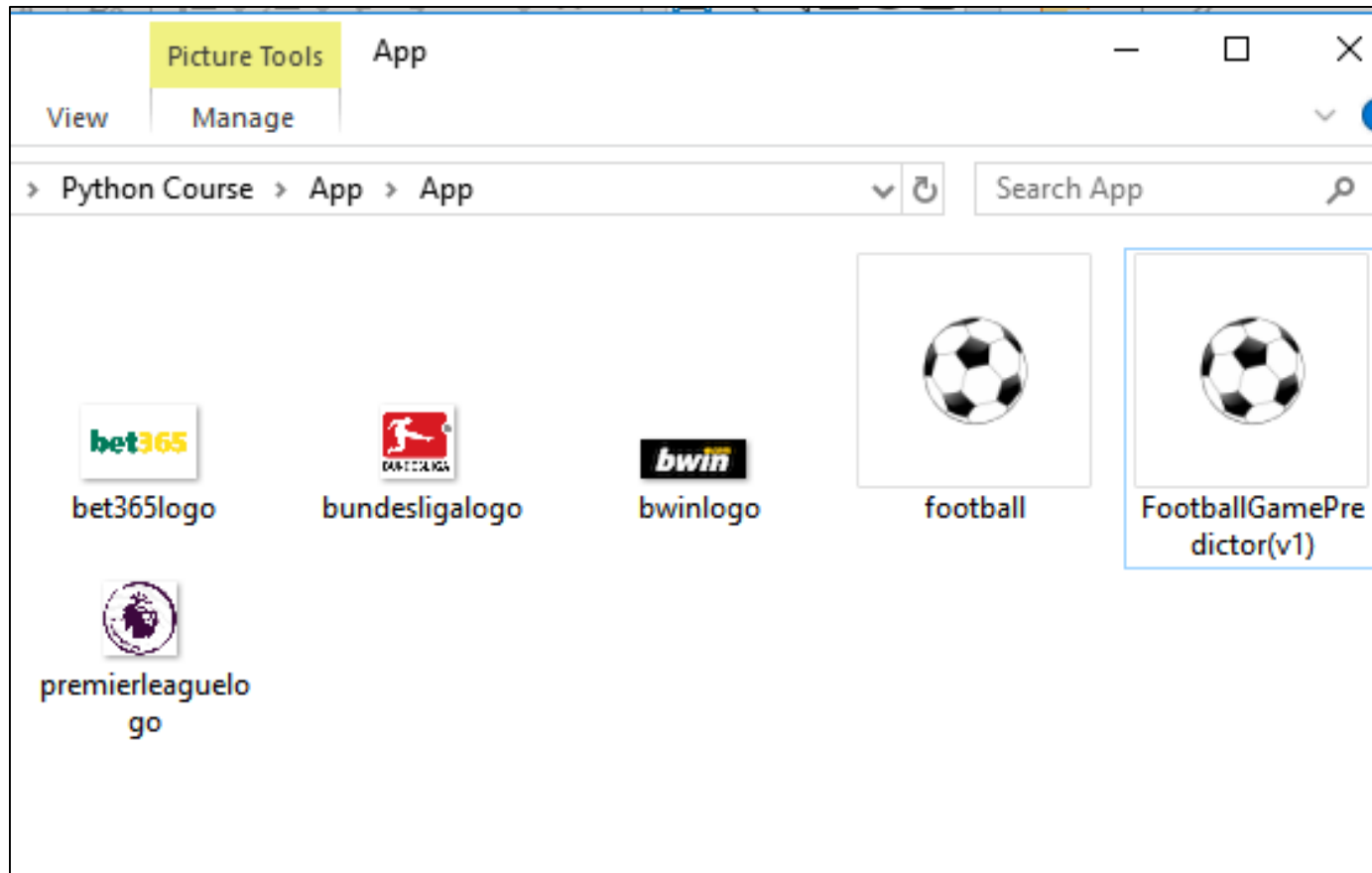
## 2. How to use the App



### **Explanatory notes:**

In case the app has connectivity issues, it will display this message. Once you have established a connection to the internet, you may try again and the app should run as expected.

## 2. How to use the App



### Explanatory notes:

It's important to notice that for the app to correctly function, the .exe file or the .py file must be in the **same file** as the icons and logos (as seen in the screenshot). This should happen automatically since the compressed .zip files should decompress all in the same file.

# 3. The Programming behind the App

## Phase A: Searching for appropriate data sources to analyse

After some research, we found a free online database, that is **weekly updated** with the latest football data. It contains historic data on the match outcomes and the odds of major international Betting Houses.

Because of the **regular updates**, it also makes sense to **automate data downloads** and thereby regularly **adapt the machine learning models** to the new data. A screenshot of the website can be seen on the right side.

You can access the website for the Premier League data via <http://www.football-data.co.uk/englandm.php>

And the website for the Bundesliga data via <http://www.football-data.co.uk/germanym.php>

**Football-Data.co.uk**  
Results / Odds / Tipsters

Network Sites

Updated: 21/05/19 SHARE BeGambleAware.org Follow

Home Free Bets Livescores Betting Advice Betting System Odds Casino Poker Tennis Books

**bet365** JOIN NOW

Claim your Open Account Offer

18+ Gamble Responsibly Terms and Conditions apply

**FREEBIES & PROMOS**

- £100 Free Bets
- £50 Free Bets
- £30 Free Bets
- £25 Free Bets
- £20 Free Bets
- bet365 Promos
- William Hill Promos

**WORLD'S FAVOURITE BOOKMAKER**

bet365

**TOP RATED**

- bet365
- betfair
- William Hill
- PADDYPOWER
- Boylesports
- 888sport
- BETFRED
- betway.com
- 000000

**ACCA INSURANCE**

Paddy Power

**PINNACLE SPORTS**

**Data Files: England**  
Last updated: 12/05/19

Registering with any of the advertised bookmakers on Football-Data will help keep access to the historical results & betting odds data files FREE.

New customers only, max refund for this offer is €€20. First Sportsbook bet only. Territorial restrictions and T&Cs apply. **PADDYPOWER**

Below you will find download links to all available CSV data files to use for quantitative testing of betting systems in spreadsheet applications like Excel. League tables, head2head statistics and information on goalscorers, first scorers and top scorers can now be accessed through the [Livescore](#) service. Latest betting odds are available through the [Odds Comparison](#).

You are free experiment with the data yourselves, but if you are looking for a bespoke Excel application that has been designed specifically to work with Football-Data's files, visit [BetGPS](#) for an exceptional data analysis workbook. Like all of Football-Data's files, it free to download.

[Notes.txt](#)  
(text file key to the data files and data source acknowledgements)

**Season 2018/2019**

- [Premier League](#) (FT & HT results; match stats; match, total goals & AH odds)
- [Championship](#) (FT & HT results; match stats; match, total goals & AH odds)
- [League 1](#) (FT & HT results; match stats; match, total goals & AH odds)
- [League 2](#) (FT & HT results; match stats; match, total goals & AH odds)
- [Conference](#) (FT & HT results; match, total goals & AH odds)

**Season 2017/2018**

- [Premier League](#) (FT & HT results; match stats; match, total goals & AH odds)
- [Championship](#) (FT & HT results; match stats; match, total goals & AH odds)
- [League 1](#) (FT & HT results; match stats; match, total goals & AH odds)
- [League 2](#) (FT & HT results; match stats; match, total goals & AH odds)
- [Conference](#) (FT & HT results; match, total goals & AH odds)

**Season 2016/2017**

**SITE RESOURCES**

- [Livescores](#)
- [Historical Data](#)
- [Betting Advice](#)
- [Football News](#)
- [Free Bets](#)
- [Odds Comparison](#)
- [Rating Systems](#)
- [Football Ratings](#)
- [In-Play Betting](#)
- [Wisdom of Crowds](#)
- [True Odds](#)
- [Test your Bets](#)
- [Test your Bets 2](#)
- [Forum](#)
- [Betting Articles](#)
- [Betting Problem](#)
- [Other Sites](#)
- [Like this site?](#)
- [Contact](#)

**ODDS & RESULTS: MAIN LEAGUES**

**Latest Matches**

- [England](#)
- [Scotland](#)
- [Germany](#)
- [Italy](#)



# Phase B: Coding

## 1. First, we want to import the necessary modules for the following three steps

```
# Modules required for Webscraping : BeautifulSoup4, lxml, requests
from bs4 import BeautifulSoup
import urllib.request
import requests

# Modules required for Data-Analysis and Machine Learning
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

# Modules required for App Interface
from tkinter import *
from tkinter.messagebox import *

# Module required to end script if error occurs
import sys
```

### Explanatory notes:

For the following **functionalities**, that we want to code, we need the **modules**, that are presented in the code.

Specifically for:

- Webscraping,
- Data-Analysis,
- Machine Learning and
- App design.

Meaning that we needed a number of modules.

## 2. Secondly, we want to webscrape and download the data for the english "Premier League" and for the german "Bundesliga"

```
try:
    # 1. Websrape data for Premier League
    # 1.1. get source code for England Football leagues of website using requests librarys
    source_pl = requests.get('http://www.football-data.co.uk/englandm.php').text

    # 1.2. pass it into beautiful soup to work with it, specifying the parser as lxml: standard parser
    soup = BeautifulSoup(source_pl, 'lxml')

    # 1.3. find all "a" tags in html code to generate download links
    code = soup.find_all('a')

    # 1.4. Loop through list of all "a" tags starting from link 73 to 83 in steps of 5 --> create download-
    #       url for each link --> download file
    x = 1

    for i in range(73,84,5):
        code2 = code[i]
        download_url = 'http://www.football-data.co.uk/' + code2.get('href') # concatenate url
        urllib.request.urlretrieve(download_url, f'PL{x}.csv')                #download url, name file

        x=x+1 #to name files accordingly by increasing numbers
```

### Explanatory notes:

To **webscrape** the relevant data, we analyzed the html code of the website. We found, that the downloadable links were all neatly sorted. Therefore, looping through the relevant **<a> tags** could get us the relevant URLs. Followingly, we could download the datasets by using the concatenated URL.

It was important for us to **fully automate the data download**, for the user's convenience.

For prediction purposes, we decided to download the data from **the latest season as well as two preceding seasons** (three datasets for three seasons in total)

```

# 2. Websrape data for Bundesliga
# 2.1. get source code of "Bundenliga" website using requests librarys
source_bl = requests.get('http://www.football-data.co.uk/germanym.php').text

# 2.2. pass it into beautiful soup to work with it, specifying the parser as lxml: standard parser
soup = BeautifulSoup(source_bl, 'lxml')

# 1.3. find all "a" tags in html code to generate download links
code = soup.find_all('a')

# 1.4. Loop through list of all "a" tags starting from link 73 to 77 in steps of 2 --> create download-
#       url for each link --> download file
x = 1

for i in range(73,78,2):
    code2 = code[i]
    download_url = 'http://www.football-data.co.uk/' + code2.get('href') # concatenate url
    urllib.request.urlretrieve(download_url, f'BL{x}.csv')                #download url, name file

    x=x+1 #to name files accordingly by increasing numbers

# 3. if error: no internet connection: display error notification and terminate script
except:
    #messagebox.showinfo("Connection Error", "Please connect to internet, that most recent data can be downloaded.")
    root = Tk ()
    root.title("Connection Error")
    root.geometry('250x100')
    Label(root, text="Please connect to internet, \n that most recent data can be downloaded. \n try again!").grid(row=
0, column = 0)
    mainloop ()
    sys.exit()

```

## Explanatory notes:

Applying the same logic to the „Bundesliga“ data allowed us to fully automate data downloads for this league as well.

In case, the user has **no internet connection**, which is required for the functionality, the user gets an **error notification**. Furthermore, the **script is interrupted** in this case.

### 3. Thirdly, we want to Read data into pandas dataframe, prepare data for modelling and and train model on data

#### 3.1. Prediction Model for Premier League

```
try:
    # PREPARE DATA FOR MODELLING
    # 1. Import csv data; we only want to use the Goals and the Odds of Bet365 (B365) and BWin (BW)
    pl_df1 = pd.read_csv('PL1.csv', usecols=['FTHG', 'FTAG', 'B365H', 'B365D', 'B365A', 'BWH', 'BWD', 'BWA'])
    pl_df2 = pd.read_csv('PL2.csv', usecols=['FTHG', 'FTAG', 'B365H', 'B365D', 'B365A', 'BWH', 'BWD', 'BWA'])
    pl_df3 = pd.read_csv('PL3.csv', usecols=['FTHG', 'FTAG', 'B365H', 'B365D', 'B365A', 'BWH', 'BWD', 'BWA'])

    # 2. Concatenate Dataframes
    pl_df = pd.concat([pl_df1, pl_df2, pl_df3], ignore_index=True)

    # 3. Add Column "Goal Difference Home-Away"
    pl_df['Goal Difference Home-Away'] = pl_df.FTHG - pl_df.FTAG

    #Erase Columns: "Full-Time Goals Home Team" and "Full Time Goals Away Team"
    pl_df = pl_df.drop(['FTHG', 'FTAG'], axis = 1)

    # TRAIN MODEL ON DATA
    # 1. Select column of Dataframe that shall be forecasted (y) and columns that are used as predictors (X)
    X = pl_df[['B365H', 'B365D', 'B365A', 'BWH', 'BWD', 'BWA']]
    y = pl_df['Goal Difference Home-Away']

    # 2. Create linear regression model
    lm_pl = LinearRegression()

    # 3. Fit Model on Data
    lm_pl.fit(X, y)

# ERROR HANDLING: If error in data: display message and end script
except:
    root = Tk ()
    root.title("Data Error")
    root.geometry('250x100')
    Label(root, text="Premier League data cannot be worked with!").grid(row=0, column = 0)
    mainloop ()
    sys.exit()
```

#### Explanatory notes:

The dataset was highly comprehensive, therefore we **only imported the columns** of the data in our Pandas dataframe, that we wanted to **perform linear regressions on**.

Our linear regression model analyzes the **correlation of the goal difference**, between the Home and the Away Team **with the Odds that the Betting Houses** „Bet365“ and „Bwin“ release before the match.

Therefore, we created a column **„goal difference“** in the dataframe and deleted the columns that this computation based on (FTHG: Full Time Home-Team Goals; FTAG: Full Time Away-Team Goals)

**Error handling:** In case that data is not usable (does not happen), user gets notified and the script ends.

### 3.2. Prediction Model for Bundesliga

```
try:
    # PREPARE DATA FOR MODELLING
    # 1. Import csv data; we only want to use the Goals and the Odds of Bet365 (B365) and BWin (BW)
    bl_df1 = pd.read_csv('BL1.csv',usecols=['FTHG','FTAG','B365H','B365D','B365A','BWH','BWD','BWA'])
    bl_df2 = pd.read_csv('BL2.csv',usecols=['FTHG','FTAG','B365H','B365D','B365A','BWH','BWD','BWA'])
    bl_df3 = pd.read_csv('BL3.csv',usecols=['FTHG','FTAG','B365H','B365D','B365A','BWH','BWD','BWA'])

    # 2. Concatinate Dataframes
    bl_df = pd.concat([bl_df1,bl_df2,bl_df3], ignore_index=True)

    # 3. Add Column "Goal Difference Home-Away"
    bl_df['Goal Difference Home-Away'] = bl_df.FTHG - bl_df.FTAG

    # 4. Erase Columns: "Full-Time Goals Home Team" and "Full Time Goals Away Team"
    bl_df = bl_df.drop(['FTHG','FTAG'], axis = 1)

    # TRAIN MODEL ON DATA
    # 1. Select column of Dataframe that shall be forecasted (y) and columns that are used as predictors (X)
    A = bl_df[['B365H','B365D','B365A','BWH','BWD','BWA']]
    b = bl_df['Goal Difference Home-Away']

    # 2. Create linear regression model
    lm_bl = LinearRegression()

    # 3. Fit Model on Data
    lm_bl.fit(A,b)

except:
    # ERROR HANDLING: If error in data: display message
    root = Tk ()
    root.title("Data Error")
    root.geometry('250x100')
    Label(root, text="Bundesliga data cannot be worked with!").grid(row=0, column = 0)
    mainloop ()
    sys.exit()
```

#### Explanatory notes:

Please relate to the preceding page. This is just the same code applied on the „Bundesliga“ dataframe.

In the end of step 3, we wanted to have **two statistical prediction models**

- **lm\_bl** model for Bundesliga data
- **lm\_pl** model for Premier League data,

to predict game outcomes based on the odds of the aforementioned Betting Houses.

## 4. App Interface to interact with user and apply Bundesliga or Premier League prediction model

```
# 1. Define Function
def show_simulation():

    # 1.1 Delete previous content
    blank.delete(0, END)

    # 1.2. Get user input from Entry fields
    try:
        B365H = float(num1.get())
        B365D = float(num2.get())
        B365A = float(num3.get())

        BWH = float(num4.get())
        BWD = float(num5.get())
        BWA = float(num6.get())

        # 1.3. If Input is non numerical: Error notification
    except:
        messagebox.showinfo("Error!", "Error! Fill out all fields with numerical values. Syntax: 1 or 1.5!")

    # 1.4. if input is appropriate: Form input to NumPy Array --> apply previously generated model on input,
    #       depending on OptionMenu selection (Bundesliga or Premier league) --> generate answer, depending on
    #       expected Home Team Goal Lead: if number negative, Away Team wins --> round results to get full goals
    #       --> Output answer in "blank" field

    else:
        I = np.array([[B365H, B365D, B365A, BWH, BWD, BWA]])

        if league.get() == "Bundesliga":
            Goals = int(round(lm_bl.predict(I)[0]))
        elif league.get() == "Premier League":
            Goals = int(round(lm_pl.predict(I)[0]))

        if Goals > 0:
            Ans = (f'Home Team Wins with {Goals} goals difference')
        elif Goals == 0:
            Ans = ('Draw is expected')
        else:
            Ans = (f'Away Team Wins with {Goals*-1} goals difference')

        blank.insert(0, Ans)
```

### Explanatory notes:

Now, we want to **apply the prediction models on the user-input**. Therefore, first, we define a function that is triggered, if the user presses the „Simulate Outcome“ Button:

1. User input from Entry widgets get **collected**
2. If input is non-numerical, user gets notified
3. Input gets passed into appropriately formatted **numpy array**, so that the prediction models can work with it
4. Depending on the selected league in the OptionMenu, „lm\_bl“ or „lm\_pl“ is **applied on the inputted Odds**. The resulting predicted goal lead of the Home Team is rounded
5. The last step is **inserting the predicted** Home Team Goal lead into the field „blank“ (If predicted Home Team lead is -3, the Away Team is expected to win the match with a 3 goal lead).

```

# 2. Defining Tkinter app interface

# 2.1. define Tkinter object --> give it title, icon and determine window size
main = Tk()
main.title("Football Outcome Predictor (v 1.0)")
main.iconbitmap("football.ico")
main.geometry('475x350')

# 2.2. To make it visually appealing, first column must have minimum size
main.grid_rowconfigure(0, minsize=50)

# 2.3. Defining fields in the grid

# 2.3.1. Option Menu: Assign stringvariable to selected option from OptionMenu (later needed) & set "Premier League"
# as default
league=StringVar(main)
league.set('Premier League')
optionmenu = OptionMenu(main, league, "Bundesliga", "Premier League").grid(row=8, column = 2, sticky = W)#added own val
ues to appear in spinbox and adjusted width

# 2.3.2 Title
Label(main, text = "You can type in odds of the Betting \n Houses to generate an outcome forecast!", font=(11)).grid(ro
w=0, column = 1, columnspan = 3, sticky = W+E)

# 2.3.3. Labels of the grid: Bet365, BetWin, Home Team Win, Away Team win, Draw
photoB365 = PhotoImage(file="bet365logo.png")
photoBWin = PhotoImage(file="bwinlogo.png")
Label(main, image = photoB365).grid(row=3, column = 0, sticky = E)
Label(main, image = photoBWin).grid(row=4, column = 0, sticky = E)
Label(main, text = "").grid(row=5)
Label(main, text = "Home Team Win").grid(row=2, column = 1)
Label(main, text = "Draw").grid(row=2, column = 2)
Label(main, text = "Away Team Win").grid(row=2, column = 3)

```

## Explanatory notes:

This section of the code organizes the visual aspects of the UI. We used the **grid design of Tkinter**. Therefore we defined our UI as a grid with

- Labels,
- Label widget cells where pictures are displayed,
- Entry widgets to receive user's input and
- a button to run the function that is defined in the preceding slide



```
# 2.3.4. Entry Fields: Defition of fields --> configuration (green background) --> Location
```

```
num1 = Entry(main)
num2 = Entry(main)
num3 = Entry(main)
num4 = Entry(main)
num5 = Entry(main)
num6 = Entry(main)
```

```
num1.configure(background="#58d68d")
num2.configure(background="#58d68d")
num3.configure(background="#58d68d")
num4.configure(background="#58d68d")
num5.configure(background="#58d68d")
num6.configure(background="#58d68d")
```

```
num1.grid(row=3, column=1)
num2.grid(row=3, column=2)
num3.grid(row=3, column=3)
```

```
num4.grid(row=4, column=1)
num5.grid(row=4, column=2)
num6.grid(row=4, column=3)
```

```
# 2.3.5. League Icon Fields
```

```
photoBL = PhotoImage(file="bundesligalogo.png")
photoPL = PhotoImage(file="premierleaguelogo.png")
Label(main, image = photoBL).grid(row = 0, column = 0, sticky = E)
Label(main, image=photoPL).grid(row=0,column=4,stick = W)
```

```
# 2.3.6. Output field of estimated result
```

```
blank = Entry(main)
blank.grid(row=6, column=2, columnspan =2, sticky = W + E)
```

### Explanatory notes:

Please refer to the previous slide, this code simply contains further definition of cells, that was explained previously.



```
# 2.3.7. Placeholder cell, "League" Label and "Simulate Outcome" - Button
Label(main, text = "").grid(row=7, column = 1)

Label(main, text = "League:").grid(row=8, column = 1, pady=10, sticky = E)

Button(main, text='Simulate Outcome', command=show_simulation).grid(row=6, column=1, sticky=E)

# 2.3.8. Copyright row
Label(main, text = "© Gp04397 & Aleiandro").grid(row=9, column =0, columnspan = 6 , sticky = W+E)


# 3. Mainloop: to let it wait for events and update the GUI
mainloop()
```

### Explanatory notes:

The code ends with a „**mainloop()**“ to ensure that the GUI waits for the events to happen and update the fields graphically.

# Phase C: Converting our scrip to a .exe file

After finishing the application coding, we wanted to **convert** the resulting .py file **to a stand-alone executable file** (.exe), because we wanted to increase the **ease-of-use**, so that e.g. no further programm would have to be installed to run the code.

We used **Pyinstaller** in the command line for conversion. The problem, that occured was, that Pyinstaller does **not support the Sklearn module** (necessary for performing the linear regression machine learning). Consequently, the core back-end functionality would not work in that case.

Therefore we had to programm a **hook file** additionally to pass in the Sklearn module manually. The code of the hook file looks like this:

```
from PyInstaller.utils.hooks import collect_submodules
hiddenimports = collect_submodules('sklearn')
```

We named the respective file „hook-sklearn.py“.

Then, we **adjusted the Pyinstaller command** in the command prompt as follows (we named or python script „2.py“)

```
(base) C:\Users\goppermann\Documents\2>Pyinstaller --onefile --icon=football.ico --noconsole --additional-hooks-dir=. 2.py
```

The output was, the desired .exe file.

Name	Änderungsdatum	Typ	Größe
 2.exe	21.05.2019 19:44	Anwendung	347.677 KB

We renamed the file and our project was **finally finished!!**