

# 1.0 INTRODUCTION

Every year there are over 2,000+ matches played in Europe. Many factors determine the outcome of these matches. Some of these factors include players, teams, infrastructure, strategy etc. Making a prediction on a match outcome is quite difficult due to many factors to be put into consideration.

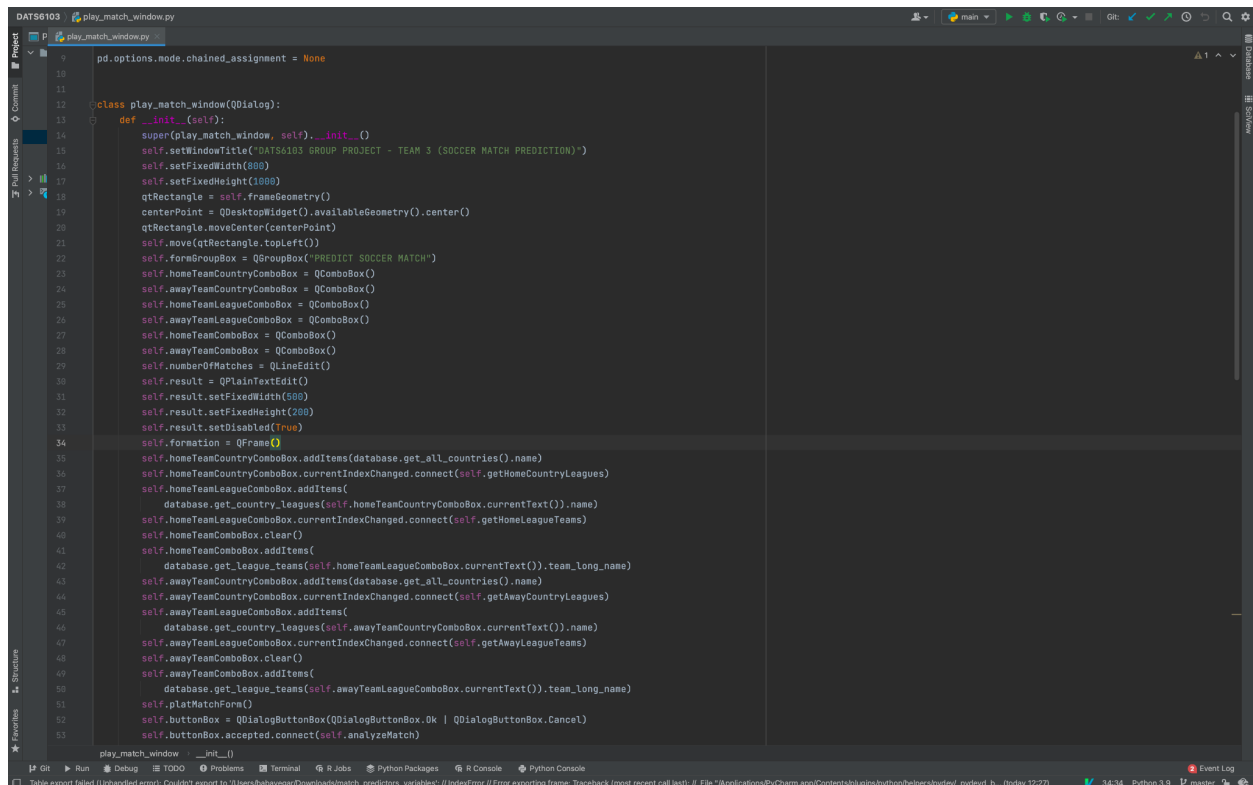
The project is supposed to analyze previous matches played and predict future matches between two teams. Since the teams are the target, we will build our predictors based on the teams statistics and features. We will design a GUI that will enable a user select both the home and away team and generate the predictors based on the selected teams to predict the match outcome.

I will be designing the GUI with the required controls we need to select the home and away teams for prediction. Also I will be adding controls to display the results of each match outcome. The GUI will be designed using PyQt5 and the GUI will be populated using pandas. Some additional results like the team formations will be displayed to give the soccer fans a more related and genuine experience.

Lastly the decision tree will be used to predict the match outcome based on analysis and training of past matches. The match outcome will either be a Home Team Win, Draw or an Away Team Win.

## 2.0 WORK DESCRIPTION

### 2.1 GUI Code 1



```
1 pd.options.mode.chained_assignment = None
2
3
4
5
6
7
8
9
10
11
12 class play_match_window(QDialog):
13     def __init__(self):
14         super(play_match_window, self).__init__()
15         self.setWindowTitle('DAT56103 GROUP PROJECT - TEAM 3 (SOCCER MATCH PREDICTION)')
16         self.setFixedWidth(800)
17         self.setFixedHeight(1000)
18         qtRectangle = self.frameGeometry()
19         centerPoint = QDesktopWidget().availableGeometry().center()
20         qtRectangle.moveCenter(centerPoint)
21         self.move(qtRectangle.topLeft())
22         self.formGroupBox = QGroupBox("PREDICT SOCCER MATCH")
23         self.homeTeamCountryComboBox = QComboBox()
24         self.awayTeamCountryComboBox = QComboBox()
25         self.homeTeamLeagueComboBox = QComboBox()
26         self.awayTeamLeagueComboBox = QComboBox()
27         self.homeTeamComboBox = QComboBox()
28         self.awayTeamComboBox = QComboBox()
29         self.numberOfMatches = QLineEdit()
30         self.result = QTextEdit()
31         self.result.setFixedWidth(500)
32         self.result.setFixedHeight(200)
33         self.result.setEnabled(True)
34         self.formation = QTextEdit()
35         self.homeTeamCountryComboBox.addItem(database.get_all_countries().name)
36         self.homeTeamCountryComboBox.currentIndexChanged.connect(self.getHomeCountryLeagues)
37         self.homeTeamLeagueComboBox.addItem(
38             database.get_country_leagues(self.homeTeamCountryComboBox.currentText()).name)
39         self.homeTeamLeagueComboBox.currentIndexChanged.connect(self.getHomeLeagueTeams)
40         self.homeTeamComboBox.clear()
41         self.homeTeamComboBox.addItem(
42             database.get_league_teams(self.homeTeamLeagueComboBox.currentText()).team_long_name)
43         self.awayTeamCountryComboBox.addItem(database.get_all_countries().name)
44         self.awayTeamCountryComboBox.currentIndexChanged.connect(self.getAwayCountryLeagues)
45         self.awayTeamLeagueComboBox.addItem(
46             database.get_country_leagues(self.awayTeamCountryComboBox.currentText()).name)
47         self.awayTeamLeagueComboBox.currentIndexChanged.connect(self.getAwayLeagueTeams)
48         self.awayTeamComboBox.clear()
49         self.awayTeamComboBox.addItem(
50             database.get_league_teams(self.awayTeamLeagueComboBox.currentText()).team_long_name)
51         self.plotMatchForm()
52         self.buttonBox = QDialogButtonBox(QDialogButtonBox.Ok | QDialogButtonBox.Cancel)
53         self.buttonBox.accepted.connect(self.analyzeMatch)
```

- a. I first declared the class and its constructor. I also defined the window title, fixed height and fixed width to accommodate the form controls appropriately. I used the frame geometry to centralize the form when executed on any computer. I declared a group box control to embed all other controls inside it.
- b. I used the combo box control to create the following dropdowns
  - Home Team Country Combo
  - Home Team League Combo
  - Home Team Combo
  - Away Team Country Combo
  - Away Team League Combo

- Away Team Combo

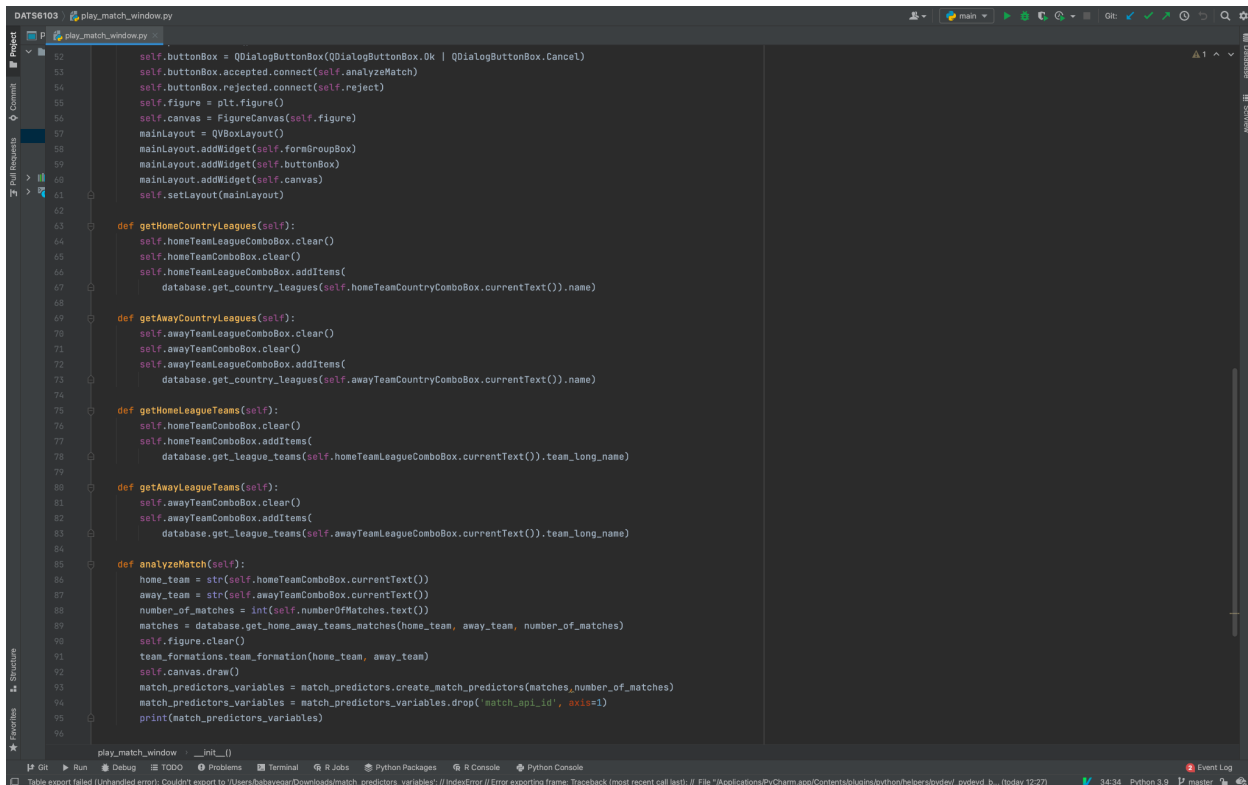
I also created a plan edit field to display the results of the predicted model and I also sized it appropriately and disabled it to make it readonly.

I added a frame control to display the team's formation plot.

c. After creating all the controls, populated all the country, league and team combo with data from the dataset using sql queries in the database.py file. When a country is selected it triggers and populates the league combo and when a league is selected it triggers and populates the team combo

d. I added a button group of default pyqt5 buttons which consists of the cancel and ok buttons. The cancel button closes the form and the ok button submits the form to perform the model analysis and prediction.

## 2.2 GUI Code 2

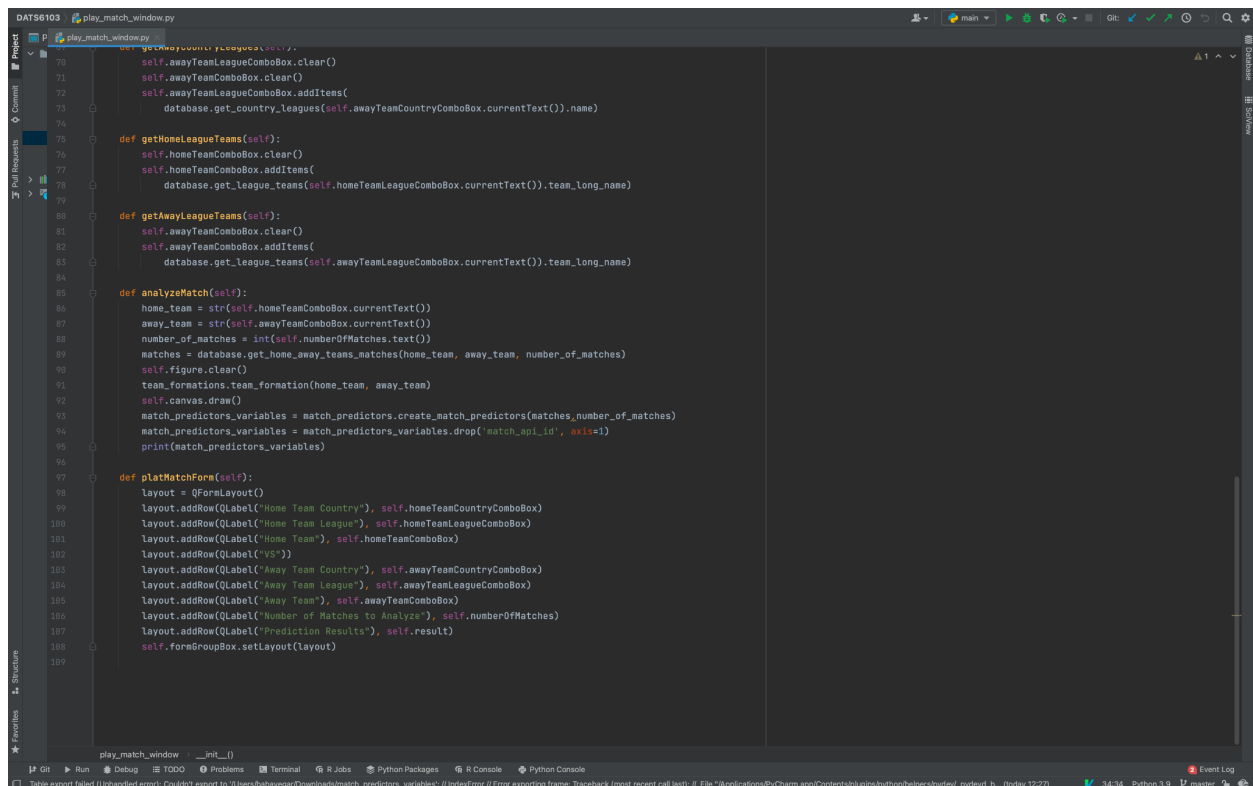


```
52 self.buttonBox = QDialogButtonBox(QDialogButtonBox.Ok | QDialogButtonBox.Cancel)
53 self.buttonBox.accepted.connect(self.analyzeMatch)
54 self.buttonBox.rejected.connect(self.reject)
55 self.figure = plt.figure()
56 self.canvas = FigureCanvas(self.figure)
57 mainLayout = QVBoxLayout()
58 mainLayout.addWidget(self.formGroupBox)
59 mainLayout.addWidget(self.buttonBox)
60 mainLayout.addWidget(self.canvas)
61 self.setLayout(mainLayout)
62
63 def getHomeCountryLeagues(self):
64     self.homeTeamLeagueComboBox.clear()
65     self.homeTeamComboBox.clear()
66     self.homeTeamLeagueComboBox.addItem(
67         database.get_country_leagues(self.homeTeamCountryComboBox.currentText()).name)
68
69 def getAwayCountryLeagues(self):
70     self.awayTeamLeagueComboBox.clear()
71     self.awayTeamComboBox.clear()
72     self.awayTeamLeagueComboBox.addItem(
73         database.get_country_leagues(self.awayTeamCountryComboBox.currentText()).name)
74
75 def getHomeLeagueTeams(self):
76     self.homeTeamComboBox.clear()
77     self.homeTeamComboBox.addItem(
78         database.get_league_teams(self.homeTeamLeagueComboBox.currentText()).team_long_name)
79
80 def getAwayLeagueTeams(self):
81     self.awayTeamComboBox.clear()
82     self.awayTeamComboBox.addItem(
83         database.get_league_teams(self.awayTeamLeagueComboBox.currentText()).team_long_name)
84
85 def analyzeMatch(self):
86     home_team = str(self.homeTeamComboBox.currentText())
87     away_team = str(self.awayTeamComboBox.currentText())
88     number_of_matches = int(self.numberMatches.text())
89     matches = database.get_home_away_teams_matches(home_team, away_team, number_of_matches)
90     self.figure.clear()
91     team_formation.team_formation(home_team, away_team)
92     self.canvas.draw()
93     match_predictors_variables = match_predictors.create_match_predictors(matches, number_of_matches)
94     match_predictors_variables = match_predictors_variables.drop('match_api_id', axis=1)
95     print(match_predictors_variables)
96
97 play_match_window = _init_()
```

- I created box layout as the main form layout and i added the form group containing the controls, the button box containing the ok and cancel buttons and the canvases to display the team formation as widgets to the main layout
- `getHomeCountryLeagues()`: This method is triggered whenever a home team country is changed. It returns a dataframe of the list of leagues for the home team.
- `getAwayCountryLeagues()`: This method is triggered whenever a away team country is changed. It returns a dataframe of the list of leagues for the away team.
- `getHomeLeagueTeams`: This method is triggered whenever a home team league is changed. It returns a dataframe of the list of teams in that league for the home team.
- `getAwayLeagueTeams()`: This method is triggered whenever a away team league is changed. It returns a dataframe of the list of teams in that league for the away team.

f. `analyzeMatch()` : This method is triggered when the ok button is clicked. In this method the home and away team selected are collected from the form and passed to the `get_home_away_team_matches` which is also a method that gets the last n number of matches of both the home and away team played in their respective leagues. Next, the team formations plot is displayed in the canvas created earlier. And based on the teams selected the target variables and predictor variables are generated and passed to the prediction model.

## 2.3 GUI Code 3



```
70 self.awayTeamLeagueComboBox.clear()
71 self.awayTeamComboBox.clear()
72 self.awayTeamLeagueComboBox.addItem(
73     database.get_country_leagues(self.awayTeamCountryComboBox.currentText()).name)
74
75 def getHomeLeagueTeams(self):
76     self.homeTeamComboBox.clear()
77     self.homeTeamLeagueComboBox.addItem(
78         database.get_league_teams(self.homeTeamLeagueComboBox.currentText()).team_long_name)
79
80 def getAwayLeagueTeams(self):
81     self.awayTeamComboBox.clear()
82     self.awayTeamLeagueComboBox.addItem(
83         database.get_league_teams(self.awayTeamLeagueComboBox.currentText()).team_long_name)
84
85 def analyzeMatch(self):
86     home_team = str(self.homeTeamComboBox.currentText())
87     away_team = str(self.awayTeamComboBox.currentText())
88     number_of_matches = int(self.numberOfMatches.text())
89     matches = database.get_home_away_teams_matches(home_team, away_team, number_of_matches)
90     self.figure.clear()
91     team_formation.team_formation(home_team, away_team)
92     self.canvas.draw()
93     match_predictors_variables = match_predictors.create_match_predictors(matches, number_of_matches)
94     match_predictors_variables = match_predictors_variables.drop('match_id', axis=1)
95     print(match_predictors_variables)
96
97 def playMatchForm(self):
98     layout = QFormLayout()
99     layout.addRow(QLabel("Home Team Country"), self.homeTeamCountryComboBox)
100    layout.addRow(QLabel("Home Team League"), self.homeTeamLeagueComboBox)
101    layout.addRow(QLabel("Home Team"), self.homeTeamComboBox)
102    layout.addRow(QLabel("VS"))
103    layout.addRow(QLabel("Away Team Country"), self.awayTeamCountryComboBox)
104    layout.addRow(QLabel("Away Team League"), self.awayTeamLeagueComboBox)
105    layout.addRow(QLabel("Away Team"), self.awayTeamComboBox)
106    layout.addRow(QLabel("Number of Matches to Analyze"), self.numberOfMatches)
107    layout.addRow(QLabel("Prediction Results"), self.result)
108    self.formGroupBox.setLayout(layout)
109
```

a. The `play_match_form()` is the method that finally displays the form(box layout).To make the GUI more readable I added labels for each control. I used the add row also to make the controls uniform and easy to navigate. When the labels are called, it also calls its respective text edit or combo defined earlier.

## 2.4 Data Preprocessing

```
4 con = sqlite3.connect("soccer.sqlite")
5 cur = con.cursor()
6
7
8 def get_all_countries():
9     return pd.read_sql('SELECT * FROM Country;', con)
10
11 def get_country_leagues(country_name):
12     query = f'SELECT DISTINCT l.name FROM League l ' \
13             f'JOIN Country c ON l.country_id = c.id ' \
14             f'WHERE c.name = "{country_name}" ORDER BY l.name ASC'
15     return pd.read_sql(query, con)
16
17 def get_league_teams(league_name):
18     query = f'SELECT DISTINCT l.name , t.team_long_name FROM Match m JOIN League l ' \
19             f'ON m.league_id = l.id JOIN Team t ON m.home_team_api_id = t.team_api_id WHERE ' \
20             f'l.name = "{league_name}" ' \
21             f'ORDER BY t.team_long_name ASC;'
22     return pd.read_sql(query, con)
23
24
```

- a. `get_all_countries()` : This fetches the list of countries from the sqlite dataset in a dataframe.No parameter is passed to this method.
- b. `get_country_leagues()` : This fetches all the leagues in a country. The country name parameter is passed.
- c. `get_league_teams()` : This fetches all the teams in a league by joining with the match table and removing duplicates.

## 3.0 RESULTS

### 3.1 Play Match Window

PREDICT SOCCER MATCH

Home Team Country

Belgium

Home Team League

Belgium Jupiler League

Home Team

Beerschot AC

VS

Away Team Country

Belgium

Away Team League

Belgium Jupiler League

Away Team

Beerschot AC

Prediction Results

Cancel

OK

The figure 3.1 above, shows the main GUI of the project. This is the first page that pops up when the program is executed. The view is a window where all controls are defined. The window has a title to make it more descriptive.

Since we are playing a soccer match i decided to have both team selections for the home and away team in different combo boxes. When a country is selected, triggers a method to populate the leagues for that country. When a league is selected it also triggers a method to populate the teams under the selected league. The data used to populate the country, league and teams are called from the methods in the database.py file as described in the code subroutines above. The controls were added using the addItem() function and calling the variable to be displayed.

Lastly i added two action buttons, which are the cancel button and the Ok button. The cancel button is used to quit the whole form and the ok button is used to proceed with the model training and generating the results. Based on the team selection, the matches for both teams are generated while limiting the records to the number of matches specified.



### 3.2 Match Played Results

PREDICT SOCCER MATCH

Home Team Country

England

Home Team League

England Premier League

Home Team

Arsenal

VS

Away Team Country

Germany

Away Team League

Germany 1.Bundesliga

Away Team

1.FC Kaiserslautern

Prediction Results

Draw

	precision	recall	f1-score	support
0.0	1.00	0.25	0.41	23382
1.0	0.00	0.00	0.00	0
2.0	0.00	0.00	0.00	0

accuracy

0.25

23382

macro avg

0.33

0.08

0.14

23382

weighted avg

1.00

0.25

0.41

23382

Cancel

OK

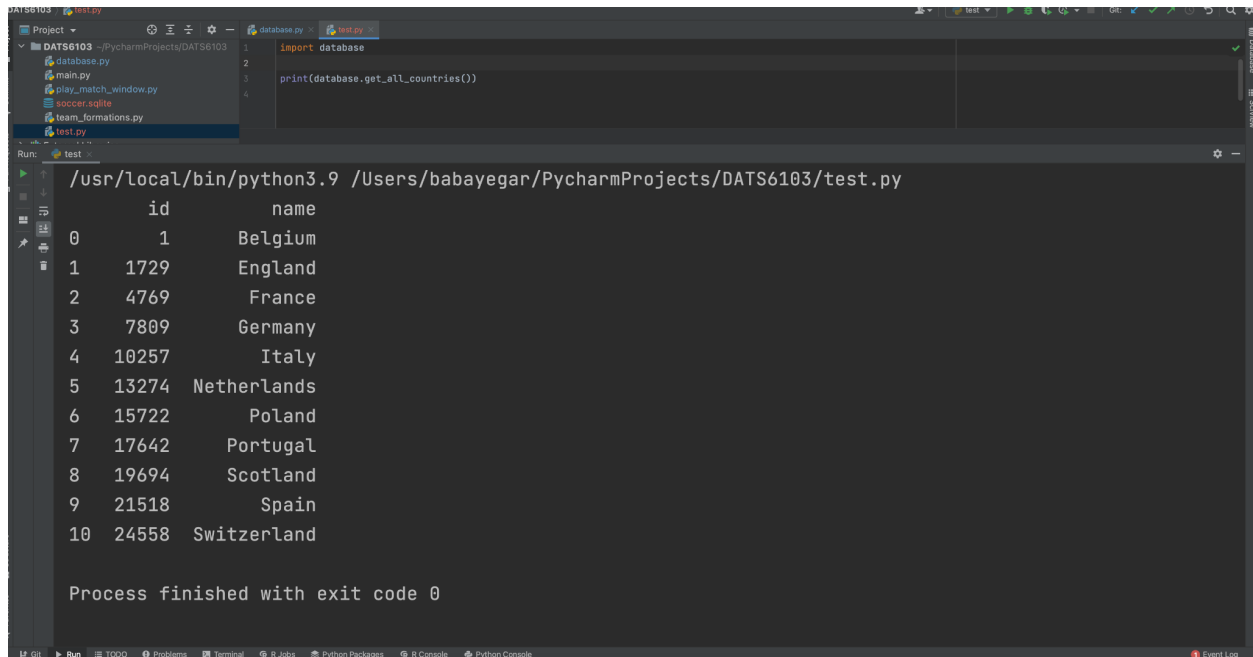
The diagram illustrates the predicted lineups for a match between Arsenal (Home Team) and 1.FC Kaiserslautern (Away Team). The field is divided into two halves by a center line. Arsenal players are represented by blue dots, and 1.FC Kaiserslautern players are represented by red dots. The Arsenal lineup includes Bellerin, Mertesacker, Cech, Paulista, Monreal, Cazorla, Ramsey, Oezil, Coquelin, Sanchez, Walcott, and Shechter. The 1.FC Kaiserslautern lineup includes Tiffert, Bugera, Wit, Rodnei, Trapp, Kouemaha, Kirch, Amedick, Sahan, and Dick. The diagram also shows the positions of the goalkeepers, Cech for Arsenal and Trapp for 1.FC Kaiserslautern.

From the figure 3.2, when the home and away team is selected , the generated predictors are passed to the decision tree classifier for prediction. When the result is returned, I first added the match outcome label to the plain text edit using the set text function and I further appended the results from the classification report to the plain text edit using the append function so that it will not overwrite the match outcome displayed.

I also went ahead to display the team formation returned as an axis on the canvas we created earlier.It has to be an axis rather than a plot.

## 3.3 Data Preprocessing Results

### 3.3.1 Country Data



The screenshot shows a PyCharm IDE window with a project named 'DATS6103'. The file explorer on the left lists several files: 'database.py', 'main.py', 'play\_match\_window.py', 'soccer.sqlite', 'team\_formation.py', and 'test.py'. The 'test.py' file is open in the editor, showing the following code:

```
1 import database
2
3 print(database.get_all_countries())
4
```

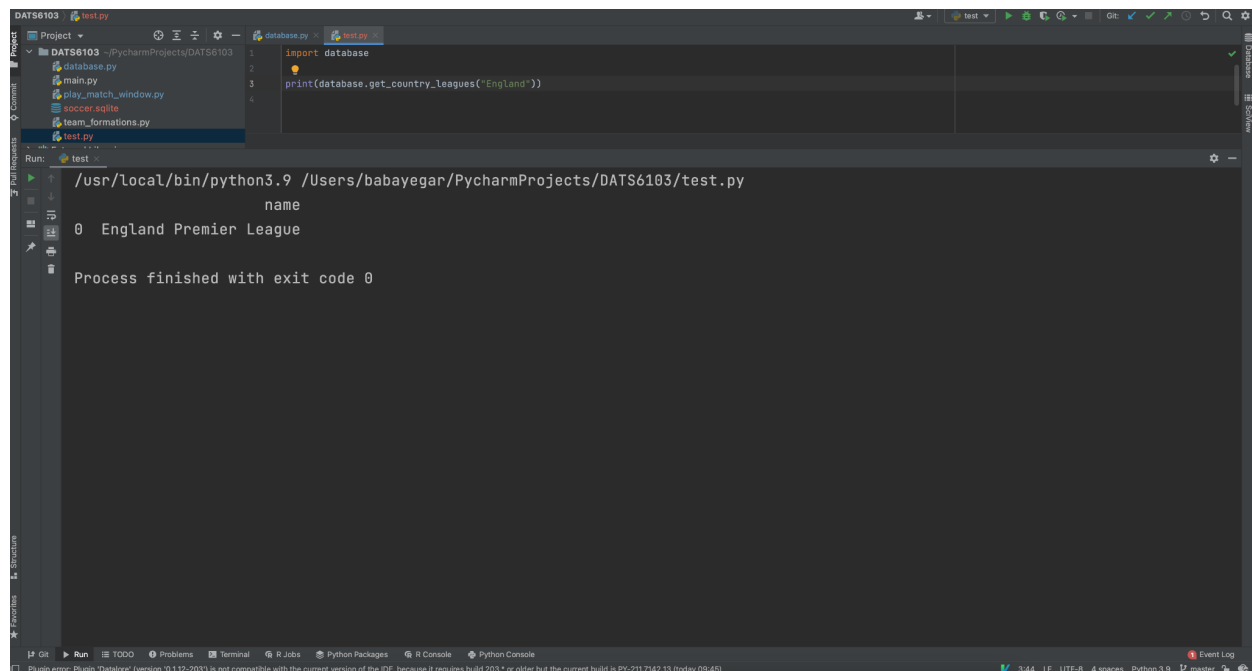
The Run console at the bottom shows the output of the script, which is a pandas DataFrame with two columns: 'id' and 'name'. The data is as follows:

	id	name
0	1	Belgium
1	1729	England
2	4769	France
3	7809	Germany
4	10257	Italy
5	13274	Netherlands
6	15722	Poland
7	17642	Portugal
8	19694	Scotland
9	21518	Spain
10	24558	Switzerland

Below the DataFrame, the console shows the message: 'Process finished with exit code 0'.

Figure 3.3.1 shows the list of countries returned from the `get_all_countries()` method in the database file. It returns both the country's id and name. When the results are obtained, it is returned in the form of a pandas dataframe. This requires no parameter to be passed.

### 3.3.2 League Data



The screenshot shows a PyCharm IDE window for a project named 'DATS6103'. The left sidebar displays a file explorer with the following files: 'database.py', 'main.py', 'play\_match\_window.py', 'soccer.sqlite', 'team\_formation.py', and 'test.py'. The 'test.py' file is open in the editor, showing the following code:

```
1 import database
2
3 print(database.get_country_leagues("England"))
4
```

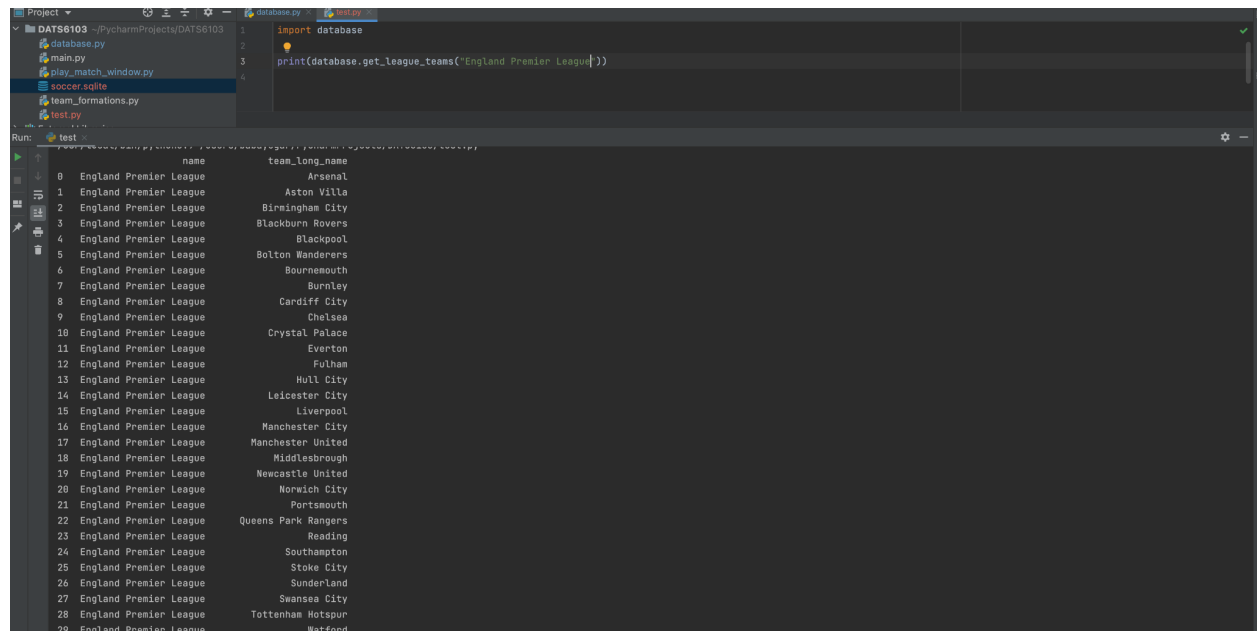
The 'Run' tab at the bottom shows the execution output for the 'test' configuration. The output is as follows:

```
/usr/local/bin/python3.9 /Users/babayegar/PycharmProjects/DATS6103/test.py
name
0 England Premier League
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8, the Python version is 3.9, and the current branch is 'master'. A small error message is visible in the bottom left corner: 'Plugin error: Plugin "Database" (version 0.112-203) is not compatible with the current version of the IDE, because it requires build 203.\* or older but the current build is PY-211.7142.13 (today 09:46)'.

Figure 3.3.2 shows the list of leagues in a country returned from the `get_country_leagues()` method is the database file. It returns both the league name. When the results are obtained, it is returned in the form of a pandas dataframe. This requires the country name parameter to be passed.

### 3.3.3 Team Data



The screenshot shows a PyCharm IDE with a project named 'DATS6103'. The file explorer on the left lists several files: 'database.py', 'main.py', 'play\_match\_window.py', 'soccer.sqlite', 'team\_formation.py', and 'test.py'. The 'test.py' file is open in the editor, showing the following code:

```
1 import database
2
3 print(database.get_league_teams("England Premier League"))
4
```

The 'Run' console at the bottom displays the output of the script, which is a list of 29 rows. Each row contains the league name and the team name:

	name	team_long_name
0	England Premier League	Arsenal
1	England Premier League	Aston Villa
2	England Premier League	Birmingham City
3	England Premier League	Blackburn Rovers
4	England Premier League	Blackpool
5	England Premier League	Bolton Wanderers
6	England Premier League	Bournemouth
7	England Premier League	Burnley
8	England Premier League	Cardiff City
9	England Premier League	Chelsea
10	England Premier League	Crystal Palace
11	England Premier League	Everton
12	England Premier League	Fulham
13	England Premier League	Hull City
14	England Premier League	Leicester City
15	England Premier League	Liverpool
16	England Premier League	Manchester City
17	England Premier League	Manchester United
18	England Premier League	Middlesbrough
19	England Premier League	Newcastle United
20	England Premier League	Norwich City
21	England Premier League	Portsmouth
22	England Premier League	Queens Park Rangers
23	England Premier League	Reading
24	England Premier League	Southampton
25	England Premier League	Stoke City
26	England Premier League	Sunderland
27	England Premier League	Swansea City
28	England Premier League	Tottenham Hotspur
29	England Premier League	Watford

Figure 3.3.3 shows the list of leagues in a country returned from the `get_league_teams()` method is the database file. It returns both the league name and the team name. As said earlier, because there was no direct relationship on the team and league table, a join with the match table had to be used and duplicates dropped. When the results are obtained, it is returned in the form of a panda dataframe. This requires the league name parameter to be passed.

## 4.0 SUMMARY & CONCLUSIONS

This project was set out to analyze previous matches and predict a new match. Moreover it was designed to be interactive, where a user's input determines the outcome of the match which makes the result change on each game played.

My work on this project was building up the entire GUI and trying to preprocess data from the sqlite dataset file. From my results, that has been very successful. From the GUI section, using PyQt5 has been a very good learning experience. One thing I noticed in PyQt5 is that when designing layouts is that the code must be written properly or else controls might not even show up. You have to size controls properly and call them at the right time. On the data preprocessing section, it was to see that SQL results can be converted to a pandas dataframe. This made populating the GUI controls possible.

## 5.0 CODE STATISTICS

The codes i wrote are both from the database.py and the play\_match\_window.py.

I wrote a total of 113 lines of code.

Line of code copied from the Internet : 31

Lines of code modified : 13

Lines of added : 82

% of code from the internet is : 15.9%

## 6.0 REFERENCE

1. <https://www.geeksforgeeks.org/pyqt5-set-fix-window-size-for-height-or-width/>
2. <https://pythonprogramminglanguage.com/pyqt5-center-window/#:~:text=To%20center%20a%20Python%20PyQt,the%20center%20of%20the%20screen.>
3. [https://www.tutorialspoint.com/pyqt5/pyqt5\\_tutorial.pdf](https://www.tutorialspoint.com/pyqt5/pyqt5_tutorial.pdf)
4. <https://datacarpentry.org/python-ecology-lesson/09-working-with-sql/index.html>
5. <https://github.com/BABAYEGAR/Data-Mining/blob/master/Demo/PyQt5/Demo/Main.py>