12 March 2021

PREDICTING FOOTBALL RESULTS WITH DEEP LEARNING

LIAM BYRNE

Contents

Analysis

Introduction

Football is a simple game where two teams, composed of eleven players compete against each other to kick the ball in the opponent's goal in order to gain a point. A game can conclude with a win, draw or loss for each team. This simplicity gives anyone in the world access to play in the sport, so it is no surprise it's the world's most popular sport. Russia hosted the last World Cup which held over 3.572 billion viewer interactions worldwide, around half of the world's population [1].

Since the millennium, the production of datasets relating to sport has soared, for example the sports analysis company Opta Sports now cover over a thousand leagues and competitions in football [2]. The ever-continuing Moore's Law has meant computer processing power will only continue to rise in the near future. The combination of these two trends has promoted the development of a new field of study, Data Science in the context of football. Data Science defines an area of study which unifies statistics, data analysis, machine learning and related methods to understand and analyse actual phenomena [3]. A new set of techniques developed to handle the vast quantities of data produced and this was named Big Data. Older techniques become overwhelmed when the volume, velocity and variety of a dataset is too great requiring a specific set of methods and technology that must be utilised to extract understanding and identifying trends [4].

On the subject of datasets, football has an almost endless expanse of data relating to all professional footballers. The demand for this data has arisen through commercial purposes such as match betting or video games such as the EA Sports FIFA franchise. The access to this data could directly impact the revenue of a match-betting company as less detailed or outdated data may be a detriment to setting appropriate odds potentially reducing their gross gambling yield. Similarly, the EA Sports FIFA players data set is updated multiple times per season for preparation and maintenance for the next release of the game. This competition between companies makes football the almost perfect example in sport to demonstrate data-science. This project will include a method to predict the outcome of football matches with data science techniques.

Motivation

The reason I have decided to pursue this project is my personal experience of being a Fulham FC fan. Every season our performance frantically switches between periods of domination to periods of chaos, this has become a regular occurrence. The term *'Fulhamish'* was created to describe the volatile nature. I hope to be able to convert this chaos to data which can be analysed in order to create a prediction system.

The field of Data Science is fascinating; the prospect of understanding real life phenomena using statistical analysis. Additionally, it is very topical and there is still significant development taking place. Techniques using machine learning approaches supply an entirely different dynamic when creating a model. In recent years, machine learning is no longer theoretical approach, but it is now accessible to the masses, has been made possible due to the increasing power in microprocessors and volume of memory.

Football as a test subject has features which make it favourable:

Football matches are of a fixed length in time (After a period after 90 minutes, the game ends)
A goal scored is worth exactly and only 1 point/goal
Premier League football has 2.65 goals scored per game on average (since 1992 establishment of PL) [5] this reduces all the possible outcomes making prediction more reasonable.
Vast expanse of data surrounding players and matches.

Development with User

Methodology

User stories will be used as a framework to identify the system goals and requirements. This methodology is a key feature of Agile software development. A user story is used to describe the functionality that will be valuable to the purchaser or user of the system [6]. These stories will be generated through dialogue with the user in order to outline his needs and other requests. A story is set in the perspective of the end-user in order to bridge the gap and encourage communication between the development team and the user.

User stories have become a key aspect of the Agile development model. In recent years Agile has been recurrently viewed as the quintessential model which a software cooperation will strive to achieve. The importance of user stories is that it brings a user's perspective and requirements into the forefront of the operation. Furthermore, a story is minimalistic and can easily be transformed into project goals this gives a straightforward analysis phase.

The User

The primary end user I must develop this project with must have a strong interest in football and requires the project for a relevent application. For those reasons considered I have selected my brother, Ryan; he is a dedicated player of Fantasy Premier League Football and an avid follower of the sport.

Fantasy football operates by a participant selecting a team of players (usually eleven) before every game week of the Premier League. Each player selected will gain points based on their performance in their Premier League game in reality. Generally, players who are a member of the winning club gain more points than their losing counterparts, directly impacting the performance of the fantasy football team. At the end of every game week the points in the selected fantasy team is totalled and added to the running total of the entire season. Participants compete in a league against other competitors and their league position is governed by their season points total.

Ryan appears to have a relevant application where a football prediction software could directly benefit his performance in Fantasy Premier League football.

Dialogue with the User

The user stories will be built from the following dialogue with Ryan in order to crystalize the key user requirements of the project.

"Would you favour an outcome (win, draw or loss) or an exact score-based prediction system?"	
-Liam Byrne	
	"As I will use this to improve my team selection in fantasy football, only the outcome is important"
	-Ryan
"Which football leagues are important to make predictions for?"	
-Liam Byrne	
	"I participate in the official Fantasy Premier League game so only the Premier League is needed."
	-Ryan
"What time period before the game would you like the system to make accurate predictions prior to kick-off?"	
-Liam Byrne	
	"Only the upcoming fixtures for the next game week are relevent, the deadline is 1 hour before kick-off just after the line-ups are publicly released."
	-Ryan
"As you said the line-ups are available to you, would entering the exact line-ups so the system can make a more accurate prediction be viable?"	
-Liam Byrne	
	"That would definitely be possible, I think it would be an important inclusion as I want these predictions to be very specific to that exact match."
	-Ryan
"What would be an acceptable runtime for a prediction to be made?"	
-Liam Byrne	
	"After every game week, I have usually 7 days to select my next team, ideally I could receive a prediction immediately as I have to usually filter through all 20 teams in the premier league"

“Would recent team form be an important metric to influence predictions?” *-Liam Byrne*
“Yes, I think recent form would be a vital metric as my selections must be time sensitive which consider how teams are playing recently not just their theoretical ability.” -Ryan
“How accurate would you require the predictions to be?” *-Liam Byrne*
“Obviously accuracy is a fundamental quality of any prediction system however, I often have little time to do any extensive research on the best players and teams to select, resulting in me not changing my line-ups or resorting to near random selections. For that reason, I would benefit from a prediction system that would be more accurate than random selections but ideally closer to an extensively informed decision.” -Ryan
“What would be the ideal platform for you to generate and receive these predictions?” *-Liam Byrne*
“I would like to interact with the system through an intuitive user interface which lets me enter all the required inputs and displays the most likely outcome. It would be important that the program does not require an internet connection as I usually decide my team when commuting. -Ryan

User Stories

1. As a user I would like to receive a match prediction within minutes after inputting key match data so that I can request multiple matches being played within a reasonable duration of time.

2. As a user I can enter the line-ups and teams playing using a user interface, so I can receive a prediction specific for that match.

3. As a user I would like to be presented with predictions related to a football club’s recent form so that I would obtain predictions with awareness if the team is playing to their theoretical capability.

4. As a user I can access the system and acquire a forecast without internet connection, so that I can receive predictions at any location.

Current System Research

Rationale for Research

The research area is predicting football match outcomes with statistics. A wide literature research will help determine the feasibility of the project and the most appropriate methods for the development of the system. This project relates to popular topics within the data-science field – sports and prediction - meaning that there is a plethora of scientific investigation papers available. Subsequently, this is a rapidly advancing area of research. Currently there are a range of algorithms which can be applied to various datasets.

Some examples of data sets include:

- In-game team statistics
- Player ratings dataset

This data can be analysed using appropriate algorithms. Some of these models include:

- *Artificial Neural Networks* (deep-learning) [7]
    - Machine learning model based on the structure of the human brain to allow modelling of various inputs and their relationships to each other. This can be used to classify results with various input variables or predict scores with regression techniques.
- *Naïve Bayes* [8]
    - Application of *Bayes Theorem* to produce a machine learning classifier which can predict an outcome based on the appearance of different events.
- *Random Forests* [9]
    - Another machine learning model where a many *decision trees* are concatenated to produce a regression or classification model. The modal path in the forest is classified class, the mean path is the regression value.
- *ELO ratings* [10]
    - See Literature review below – *Corentin Herbinet* paper
- *xGoals* (expected models) model [11]
    - See Literature review below – *Corentin Herbinet* paper

The list above is created by samples taken from the scientific papers collected. In recent years there has been an influx of machine learning approaches due to the increasing processing power of microprocessors and a high volume of memory being readily available, thus allowing machine learning methods to be performed locally across a vast population.
Two particular references will be examined in more detail below to understand the feasibility and approach for this project.

Literature Review

1) Predicting Football Results Using Machine Learning Techniques - Corentin Herbinet [11]

A report conducted by an Imperial College London student details how various machine learning techniques can be applied for football prediction. His primary goal was to benchmark different techniques which utilise in-game statistics for their respective accuracy for prediction.

The data used is sourced from an online dataset on Kaggle (online platform containing datasets for open use) called 'European soccer database' which holds many players and matches played. Although he stated that he intended to use 'WhoScored.com' as they have more interesting data however, they restrict web-scraping (extracting content of webpages) for data protection.

xGoals is short for 'Expected Goals' which is a metric which states the number of goals which by probability were likely to have been scored based off in-game statistics. He calculates match xGoals and shot xGoals by analysing shots taken within a game. *'Elo'* ratings are a rating system first introduced into chess which not only gives the relative strength of a player but additionally allows probabilities to be mathematically produced about a player beating another player. The competing football teams have a normal distribution curve plotted around their skill rating, the probability of outcomes is proposed by the two curves proximity and intersections. These statistics are very data intensive requiring a vast amount of data to calculate with a level of accuracy.

By combining these statistics, a model is created which can be applied to machine learning algorithms. A classification model is built to predict the outcome of the match as there is only a maximum of 3 qualitative outcomes (win, draw or loss). A regression model is used to predict the exact score of the game, this is a more quantitative measure as the score of the game can have a potentially infinite range.

Finally, he details the success of different algorithms. For the match outcome classification; the different accuracies are shown in figure 1:

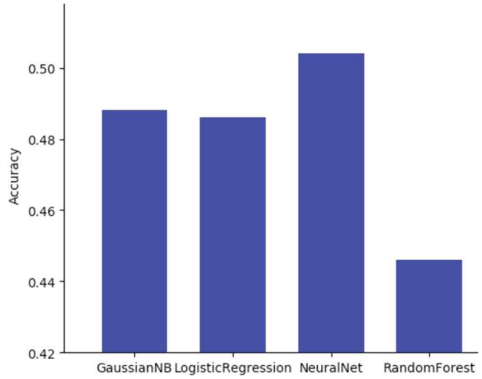


FIGURE 1

Figure 1 shows the accuracies calculated by:

EQUATION 1

$$Accuracy = \frac{n_{correct}}{n_{total}}$$

Where $n_{correct}$ is the number of times the model has correctly classified the outcome and n_{total} is the total number of attempts given to the model.

Conclusions drawn from this chart are that Artificial Neural Networks (ANN of 2 hidden layers of 100 neurons) have the greatest accuracy in this instance. Gaussian Naïve Bayes and Logistic regression have similar accuracies but fall behind ANNs. Random forests appear the least suitable in this instance will a significant drop in accuracy, perhaps they overfit the model.

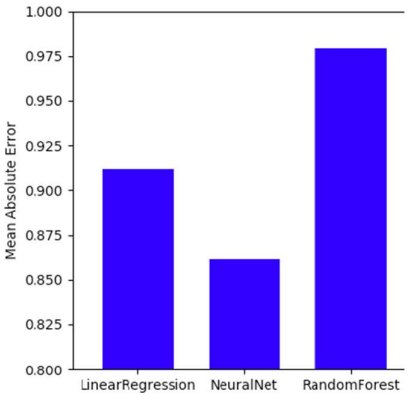


FIGURE 2

For exact score prediction the regression model is tested using mean squared error (lower error is more accurate). Once again Neural Networks are the most accurate, significantly ahead of linear regression and Random Forest algorithms.

2)Predicting Football Matches using EA Player Ratings and TensorFlow [12]

This source is an example of an implementation posted on medium.com by a data-scientist at Channel 4. Similarly, a player dataset has been used, but in this case, it is the popular *EA Sports FIFA dataset*. Deep-Learning with neural-networks was chosen and implemented with Google's TensorFlow Python libraries. The neural-network is composed of 2 hidden layers with 16 and 8 neurons respectively, the composition of the size of neural networks appears as largely arbitrary. Line-ups of each game were web-scraped and matched up with their player ratings in the other dataset to build a team with ratings. Web-scraping or scraping is the process of applying various techniques in order to extract information and data stored on a web-site, this data is then saved locally. This implementation uses the Python library: *'Scrapy'* which acts as a web-crawler, another common method is using the *'BeautifulSoup'* library in Python which inspects the HTML of a given page. The model was trained with the outcome of the games across multiple seasons of football, as reason to EA Sports releasing yearly ratings releases. A focus of his project was to predominantly use player data rather than the cliché xGoals model which has been used very heavily in football analysis. His approach allows an entire season to be simulated to build a league table by placing their most common line-up into the model. The betting industry was used as a way to measure the success of the model by measuring the return-on-investment.

Research Conclusions

The first source details the more theoretical use of various methods while the second source focusses on an implementation. After researching both, conclusions can be drawn which will guide the development of this project:

Artificial Neural-Networks are most appropriate for football data outcome classification. Outcome prediction and not score prediction will be sufficient as this is what our primary end-user - Ryan - asked for. Furthermore, ANNs can be produced in Python with well supported libraries.
Player ratings appear to be more accessible than in-game statistics, additionally they may allow predictions which are relevent further into the future.
These references show the general idea is feasible and possible to produce in Python.

Project Goals

This project will produce two artefacts: software to predict football matches and a documented analysis of the performance of the solution.

The first goal is ***"to produce a system where a user can receive a realistic prediction of the outcome of a specified Premier League football match"***. This system will include an interactive graphical user-interface which allows a user to enter the line-ups or predicted line-ups and the clubs competing. The system should take this data and process it and return the most likely outcome of the game being a home* or away**: win, draw or loss.

The second goal is ***"to document the analysis of the performance of the solution".*** The documentation will include the techniques and algorithms used to produce the most accurate predictions over a test set. Additionally, a documented method of testing the accuracy and the used input data which give the most accurate predictions.

* 'home' : where the team is playing at the stadium which belongs to the club.
** 'away': where the team is playing at a stadium which belongs to a different club.

Project Objectives

The goals have been broken down into several objectives which define the steps required to achieve a solution:

"To produce a system where a user can receive a realistic prediction of the outcome of a specified Premier League football match".

1. ***Gather player data from EA sports FIFA databases across available previous seasons.***
   - *1.1. Request the HTML data of the web-page holding players with the required player attributes (www.sofifa.com/players chosen in design phase).*
   - *1.2. Parse the HTML tags to locate the name of a player and their football attributes.*
   - *1.3. Navigate through different pages on the domain to iteratively collect the entirety of the player data.*
   - *1.4. Write player data to a CSV file.*

2. ***Gather Premier League results and line-ups across available previous seasons.***
   - *2.1. Request the HTML data of the web-page holding Premier League results and respective line-ups (www.soccerway.com chosen in design phase).*
   - *2.2. Parse HTML tags to locate and extract specific data about the match and line-ups.*
   - *2.3. Navigate through different pages on the domain to iteratively collect each game across multiple seasons.*
   - *2.4. Write results and line-ups to a CSV file.*

3. ***Store player data, results and line-ups persistently in a database.***
   - *3.1. Create a normalised SQL database with interlinked tables.*
     - *3.1.1. Create each table with columns for all data required for later processes in the system.*
     - *3.1.2. Interlink each table with the required relationship (see ERD diagram in design phase).*
   - *3.2. Insert scraped data into the corresponding table.*
     - *3.2.1. Assign each entry into the table a unique ID number to act as primary and foreign keys.*
     - *3.2.2. Link each player held on the player dataset to the same player found on the line-up dataset (Player data in different formats).*

4. ***Create an interactive GUI (Graphical User Interface) which allows the user to enter clubs and line-ups.***
   - *4.1. Deploy an appropriate interface for the user to enter a home and an away club*
     - *4.1.1. The user-interface must have a drop-down menu of all Premier League clubs.*
   - *4.2. Deploy an appropriate interface for the user to enter the line-up of each team.*
     - *4.2.1. The user-interface must suggest players in the selected team after the club is chosen.*
     - *4.2.2. The user-interface must have a drop-down menu must allow the user to select the exact player they are looking for.*
     - *4.2.3. The user-interface must only allow a line-up of 11 players to be entered.*
     - *4.2.4. The user-interface must allow specific players to be omitted or all currently entered players to be removed after being chosen.*

5. ***Calculate a club's recent form using the data stored in the database using relevant statistics.***
   5.1. Pull values the database which are within the necessary time period and required to calculate form.
   5.2. Calculate the recent form of the respective club for a specific game-week by accumulating the points achieved in the Premier League across the previous month.

6. ***Calculate relevent metrics from the database (e.g. defensive rating, offensive ratings)***
   6.1. Pull the line-up of the team playing in the specific match
   6.2. Identify if the fixture is being played at their home stadium or at an away stadium
   6.3. Produce various metrics which analyse the player ratings held in a line-up at different playing positions in the field-of-play.

7. ***Create a model with the metrics being mapped to the result using a machine learning technique.***

8. ***Train the model with metrics and the results in the database and store persistently.***
   8.1. Divide all input and output data into training and testing sets of data.
   8.2. *Input the various metrics and form calculated of a particular game into the model and train against the known historical result.*

9. ***Display the calculated prediction as a discrete outcome to the user.***

"To document the analysis of the performance of the solution"

1. ***Determine the most appropriate machine learning approach (e.g. Classification or Regression)***

2. ***Determine the specific machine learning algorithm***

3. ***Identify the most appropriate combinations of input data to use***

4. ***Recognise the pre-processing required for the specific model***

5. ***Test the accuracy of model using the testing partition of the original training data.***

System Overview

Data Sources

Being a data-science oriented project, access to high quality datasets is crucial. This project relies on two major datasets: FIFA Players and Premier League historical results/line-ups. Both of these datasets are presented on various webpages; this data will need to be extracted and built into a dataset locally. Extracting from web-pages requires a web-scraper, various websites prohibit web-scraping as data on their website may be protected by copyright, this was seen when Corentin Herbinet attempted to scrape 'www.WhoScored.com' but was rejected [11]. The web-pages selected must not restrict scraping or crawling to avoid IP blacklisting. The 'robots.txt' page of a domain details how a crawler should interact with the web-page and folders it has access rights to.

For the players dataset 'www.sofifa.com/players' appears a viable candidate as it appears to be the primary source of player data used by many other published investigations. The 'sofifa.com/robots.txt' page allows all access and does not restrict a scraping on the players database. Additionally, the HTML of the website is very regular and predictable allowing straight-forward scraping. Surprisingly Electronic Arts (EA) permits the use of all their player data despite it being used on to generate revenue with the annual release of FIFA football game.

If possible, another webpage will hold both Premier League results and their respective line-ups of the home and away clubs. One candidate is 'soccerway.com' which involve both data and permits scraping. Similarly, the HTML is very simple to parse but navigating the page is somewhat irregular where the page contains iterative elements which must be used to change page data. One issue predicted is marrying up the same player held on the FIFA dataset and these line-ups with different name formats.

Data Quality

This project is reliant on external data sources which means the accuracy and maintenance of the data are entirely governed by a third-party source. The FIFA dataset is built of mostly subjective measures where EA employs 'data reviewers' around the world to watch specific players and award ratings over specific attributes. All metrics measure from an arbitrary 0-100 rating system. EA updates this database multiple times in a season, but no major changes are made until imminently before EA Sports FIFA game is released at the end of September.

Specific Algorithms

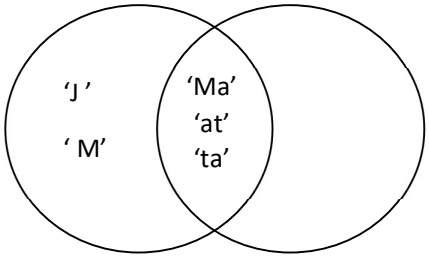
In objective 3.2.2, each player held on the player dataset must be matched to the same player on the line-up dataset. For example, a player is referred to as 'Jonny' and 'Jonathon Castro Otto' which will appear differently in each dataset meaning they simply can't be matched based on their names. Furthermore, no part of each name appears in the other name, this calls for a string similarity algorithm.

There are many choices over string similarity algorithms, most notably: Levenshtein edit distance, Jaro-winkler and n-grams. Levenshtein determines the minimum number of transpositions and character changes required to reach a target string from the original. Jaro-winkler works by calculating the relative similarity to produce a score between 0 and 1 where 1 is an exact match. Additionally, Jaro-Winkler favours characters near the beginning of a word to match names of people. However, many names in these datasets are inconsistent with their structure. For example, 'J. Mata' and 'Mata' are the same person however the first initial is included, as this is at the beginning of the word Jaro-Winkler will incorrectly determine a large distance. N-grams works by breaking down a string into a sequence of smaller chunks.

The example below breaks down 'J Mata' and 'Mata' into bigrams. The intersection of common bigrams is taken and divided by the total number of unique bigrams to determine the Jaccard co-efficient, representing similarity.

'J Mata' → ['J ', ' M', 'Ma', 'at', 'ta']

'Mata' → ['Ma', 'at', 'ta']



EQUATION 2

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \rightarrow \frac{|\{'Ma','at','ta'\}|}{|\{'J ',' M','Ma','at','ta'\}|} \rightarrow \frac{3}{5}$$

Data Flow Diagram (DFD)

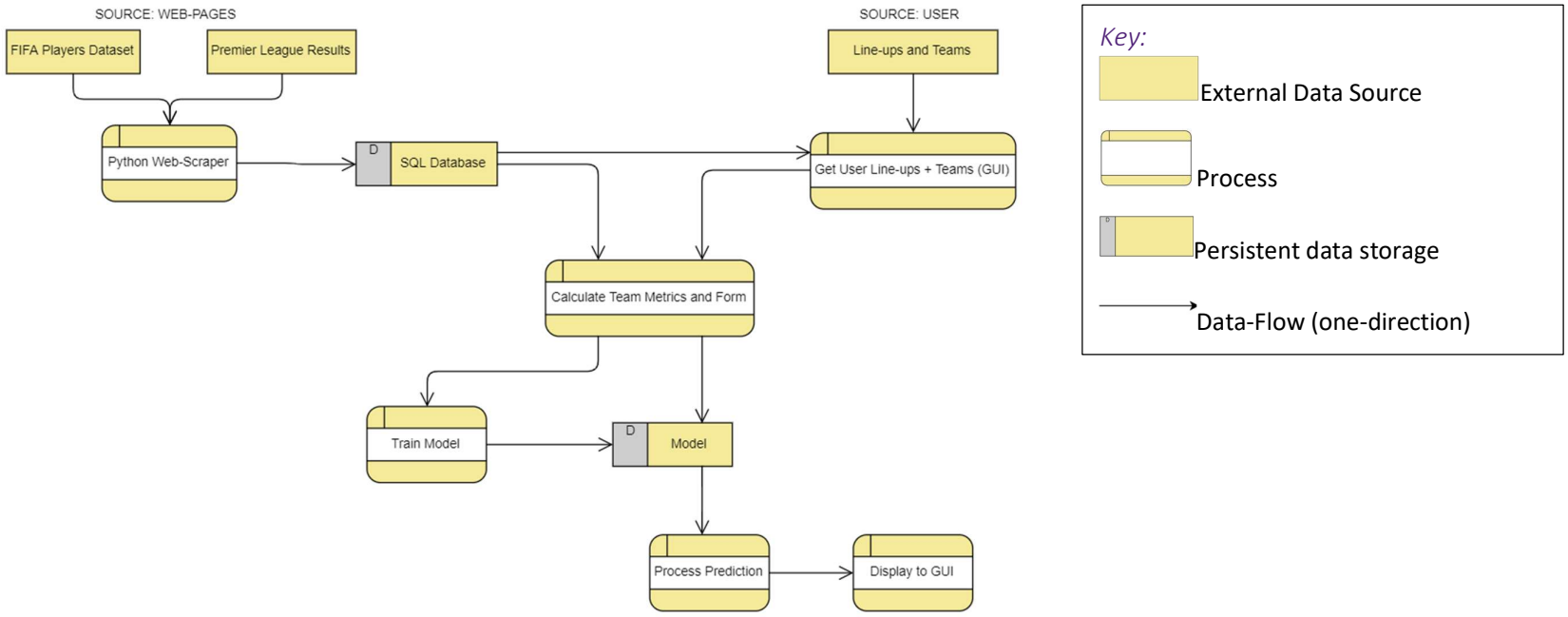


FIGURE 3

The diagram above has been assembled using the project objectives. A data flow diagram shows the data moves and is processed through the system.

The flows begin with the external data sources:

FIFA players dataset
Premier League results
Line-ups and Players (user)
} Held on online web-pages

Model Training Flow

The FIFA Players and Premier League results data is extracted from webpages using a web-scraper implemented in Python. This data is cleaned and placed in a normalised SQL Database holding both players attributes and results with the exact historical line-ups from multiple seasons. This persistent data is fed into a process which calculates different metrics of the exact line-ups to determine the theoretical strength of the team, recent team form is calculated using recent results to determine whether the team is playing to their potential or greater. These metrics are trained against the final result of the game to construct a model which can be used for prediction. The data set will be split into a training and testing sectors to determine the accuracy to identify if the user's specification has been reached. The volume of the data will depend on how far back in history the datasets have records of. Once trained this model is stored persistently so training does not have to take place every time the system is accessed.

Prediction Flow

Forecasting a game is the foremost feature of this project. This flow requires a fully trained model to generate a prediction. The user interacts with system through the graphical user interface (GUI) by entering the home and away line-ups prior to the game. This GUI has access to the players database to allow players to be suggested to allow players to be intuitively selected. Similar to the training flow, metrics of the inputted team are calculated and the recent form of the selected teams. These various metrics are inputs to this model. The model will determine the most likely outcome of the method and return a value. This output may have to be processed and converted to a simple discrete outcome, for example if regression is used a large decimal value will likely be returned. This discrete outcome is revealed to the user on the GUI.

Limitations

Prediction by definition has limitations as it is merely the analysis of probabilities. From an overall perspective football is a good subject for prediction; generally, the better team will dominate the underdog and it will end as expected. Nonetheless football is renowned for shock results where the underdog defies all the odds and rises to beat the powerhouse such as Leicester City FC winning the Premier League with odds of 5000/1 in August 2015. From a narrow perspective football contains anomalies where the preceding occurs, fortunately for the subject of prediction – this is purely an anomaly.

For my proposed implementation: the user must enter the line-up of a team. It is normal practise for Premier League clubs to release the line-ups of the upcoming match an hour prior to kick-off. This means that predictions which are entirely informed can only be made at earliest an hour before. Consequently, matches which take place beyond an hour must resort to a predicted, or the most common line-ups. Evidently this reduces the accuracy as it is likely the match will be predicted with a different set of players compared to reality.

Additionally, the solution will act in a pessimistic manner towards the underdog team as a result of the stronger team having greater player ratings, subsequently the weaker team is unlikely to be favoured. As described earlier, football performs in this predictable way for the majority of matches excluding the irregular unexpected result. With this considered, alternative methods which ignore the team rating or reputation may be favourable as they start with a level playing field.

Finally, substitutes during the match will not be accounted for in the training of the model and for prediction as both are not accessible prior to the line-up release (1 hour before kick-off). This will cause a greater inaccuracy as a substitute may drastically change the rating of the team, effecting the training of the model and predictions prior to a match. The majority of Premier League substitutions are made in the second half of the game, with this assumption - the substitute is less likely to have an impact on the game.

Documented Design

Overall System Design

As discussed earlier in a data-flow diagram (Figure 3), the system will be broken down into 2 major flows: Training Model flow (Figure 4) and Prediction flow (Figure 5). The processes revealed earlier can be further decomposed into additional sub-processes. The diagrams below show all the sub-processes in order to inform the development of prototypes and the final implementation. The flows are displayed on separate diagrams to ensure the structure is unambiguous, this means that some processes and entities are seen on both diagrams, please note that these will be implemented as single entities and integrated as documented previously in Figure 3.

As some elements of the system are more complex; they cannot be fully explained with a DFD. Therefore, their structure and the various processes required for set-up and operation are documented in the following sections.

Training Model Flow

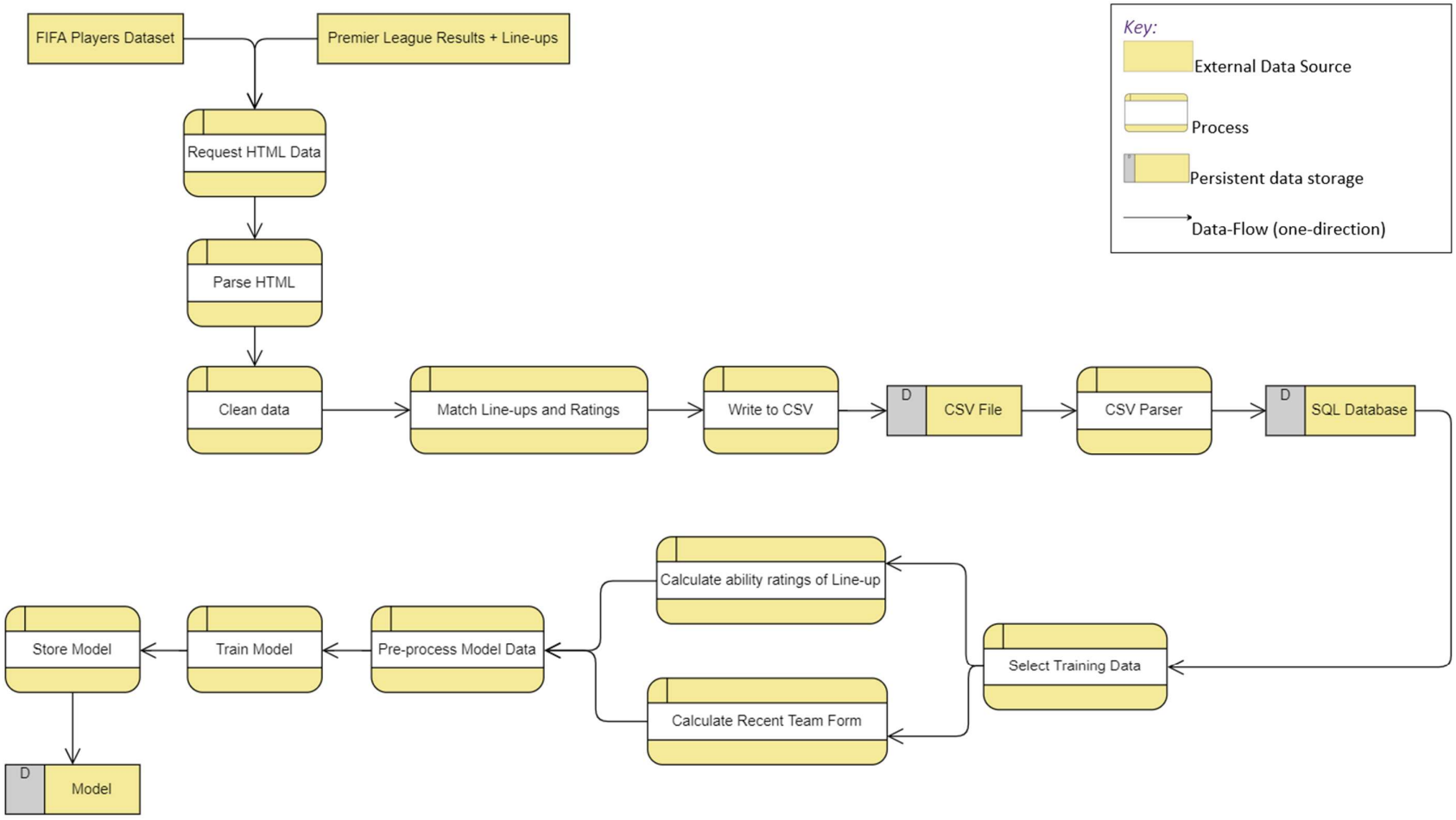
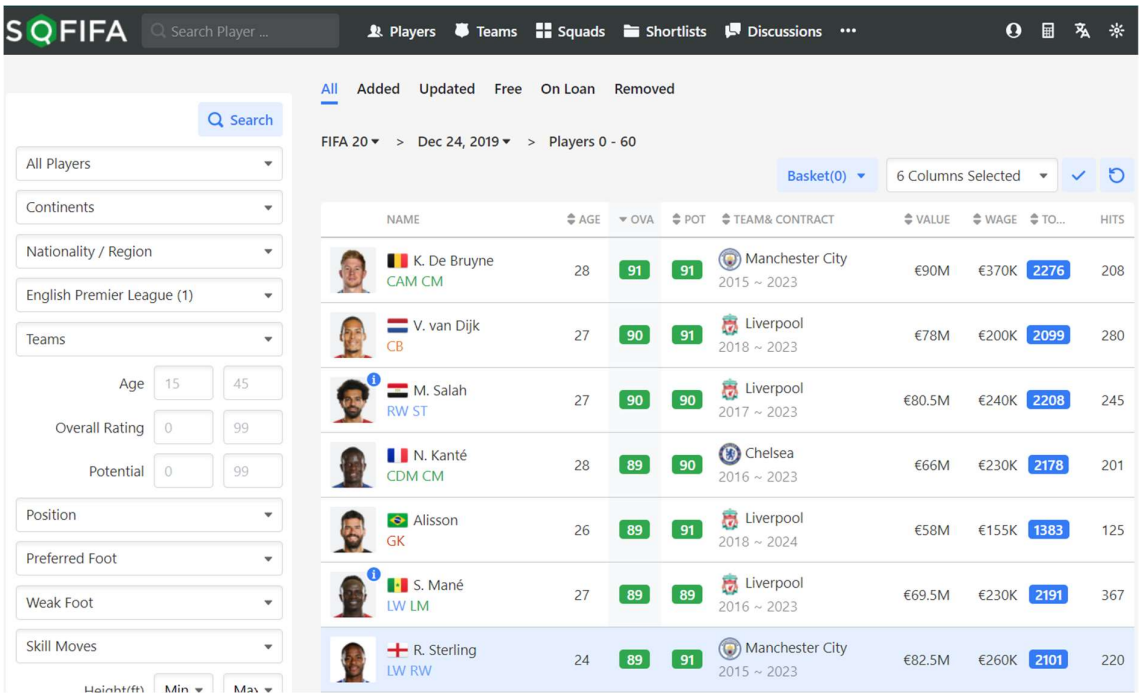


FIGURE 4

Collecting Data

The web site 'www.sofifa.com/players' was selected as a data source which provides the data on all Premier League players and their attributes as judged by EA Sports (Figure 5). Similarly, 'www.soccerway.com' was selected as a navigable page to provide Premier League results and the exact starting line-ups playing in that game. The data stored on these webpages must be extracted with web-scraping techniques. An implementation of scraping the web-pages would include the program making a request to download the HTML data stored on each web-page. The HTML of a webpage details all the content present on a page in a barebones structure and acts as a mark-up language. The HTML is then parsed which navigates through the tags and content stored on the page to extract the required information. Each web-page of the selected web-sites contains a uniform structure throughout, for example, the data of each player is stored in rows marked by the tag <tr> on a large table with the tags: <table class=" table table-hover persist-area">. The uniformity of each page allows a more automated solution which means that the program does not need to be hard-coded for each piece of content on the page, this would take an unreasonable duration of time to collect thousands of datapoints on each player and fixture. Errors and poor formatting usually come as detriment to a more automated solution. Data cleaning must take place to ensure the data follows a uniform format and doesn't contain erroneous data points.

FIGURE 5



As mentioned in the analysis section, 'soccerway.com' appears to have a more complicated structure. Figure 6 outlines how each page is connected on the domain. Our access point will be the 'Premier League Page' which contains all information about the competition, by editing the URL we can get each 'Season Page' (Figure 7). Figure 7 contains a table which lists every match played this season. Each listed match has a URL which can be extracted from its HTML HREF tag, this is the 'Match Page' (Figures 8, 9).

The table on each 'Season Page' is a dynamic JavaScript table rather than a static HTML table found on 'sofifa.com'. The HTML content of each page is inaccessible when the page is not in view. This means the 'next' button must be clicked to cycle through each page. The can be solved with a webdriver using the Selenium library. A webdriver opens a web-browser and uses executes JavaScript to interact with page elements.

FIGURE 6

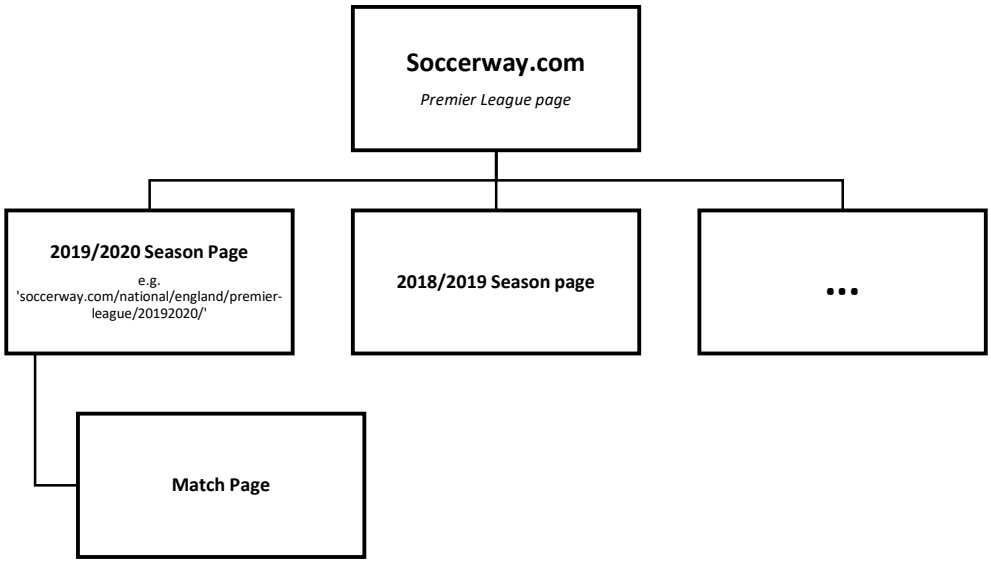


FIGURE 7

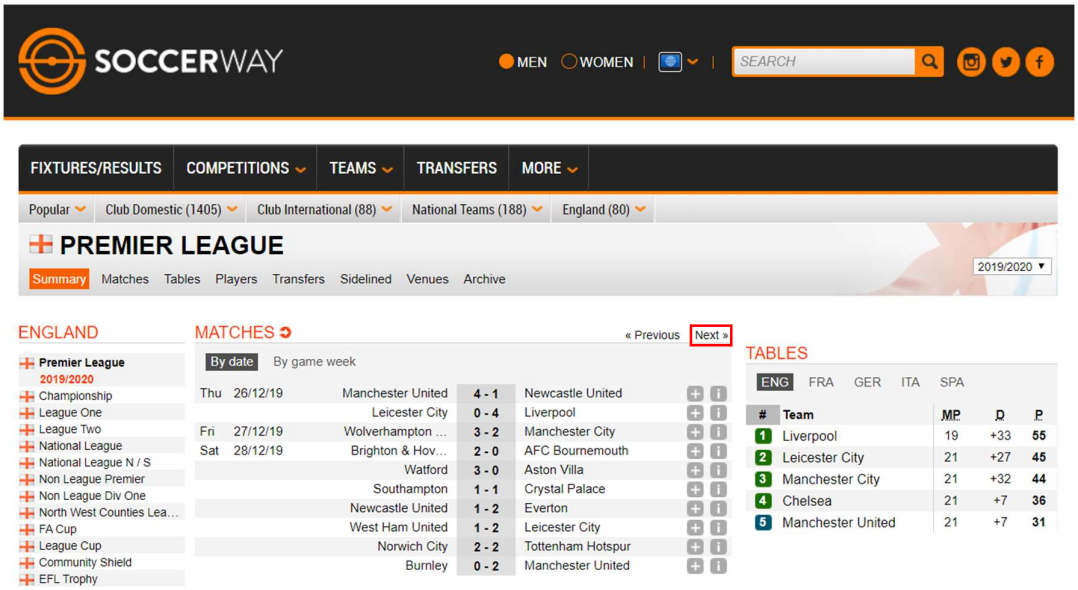


FIGURE 8

INFO

WATFORD	3 - 0	ASTON VILLA
W D W L D		L W L W L

Competition	Premier League
Date	28 December 2019
Game week	20
Kick-off	15:00
Half-time	1 - 0
Full-time	3 - 0
Venue	Vicarage Road (Watford)
Attendance	21348

GOALS

T. Deeney 42'	1 - 0
T. Deeney (PG) 67'	2 - 0
I. Sarr 71' (assist by E. Capoue)	3 - 0

FIGURE 9

26	B. Foster		1	T. Heaton	
6	A. Mariappa	51' 57'	27	Ahmed El Mohamady	
15	C. Cathcart		30	K. Hause	
27	C. Kabasele		18	M. Targett	
21	Kiko Femenía		15	E. Konsa	
29	E. Capoue	74'	14	C. Hourihane	
16	A. Doucouré		8	H. Lansbury	
19	W. Hughes		23	Jota Peleteiro	
9	T. Deeney	42' 67' 90+2'	10	J. Grealish	90+6'
7	Deulofeu		6	Douglas Luiz	
23	I. Sarr	71' 84'	9	Wesley	
Coach:	N. Pearson		Coach:	D. Smith	

Storing Data

The data stored in the system does not have any need to be changed once the system has been set-up. For this reason, a csv file will act as an intermediary data storage point between the raw data being pulled in and the SQL database. This allows the database to be restored without access to the internet, a feature the user said he would find useful. Another reason to include a csv file is that collecting all the data from the web-scraper and then inserting the values into the table will act as a safeguard to the database to ensure all values are entered correctly avoiding the database to crash. Some examples of a lines from the csv files is seen below:

FIGURE 10

Player:

```
T.Cairney, Tom Cairney, CAM, 78, Fulham, 2018/2019
```

FIGURE 11

Result:

```
Fulham, Southampton, 1 - 1, 26 December 2012, M. Schwarzer,…
```

A CSV file has the advantage that it has a structure which can easily be translated to a table within a database; the structure of the database is detailed in the section: Database Design. This data stored will inform the training, prediction and entering players into the GUI.

Calculating Metrics

A fundamental part of this system is the ability to identify how strong a football team is from their line-up alone and whether they are playing to that ability or beyond. The databases detail historical Premier League results meaning that a recent form statistic form statistic can be generated. Recent form is a metric which concerns the most recent games being played. Some examples of form measures would be accumulating the total points gained in the last 30 days or individually looking at player data or accumulated goals scored in the last 30 days. Regarding the data in the tables simply details the score with its date, accumulating the total points achieved in a timeframe would be an appropriate operation which doesn't require any additional data being brought into the tables.

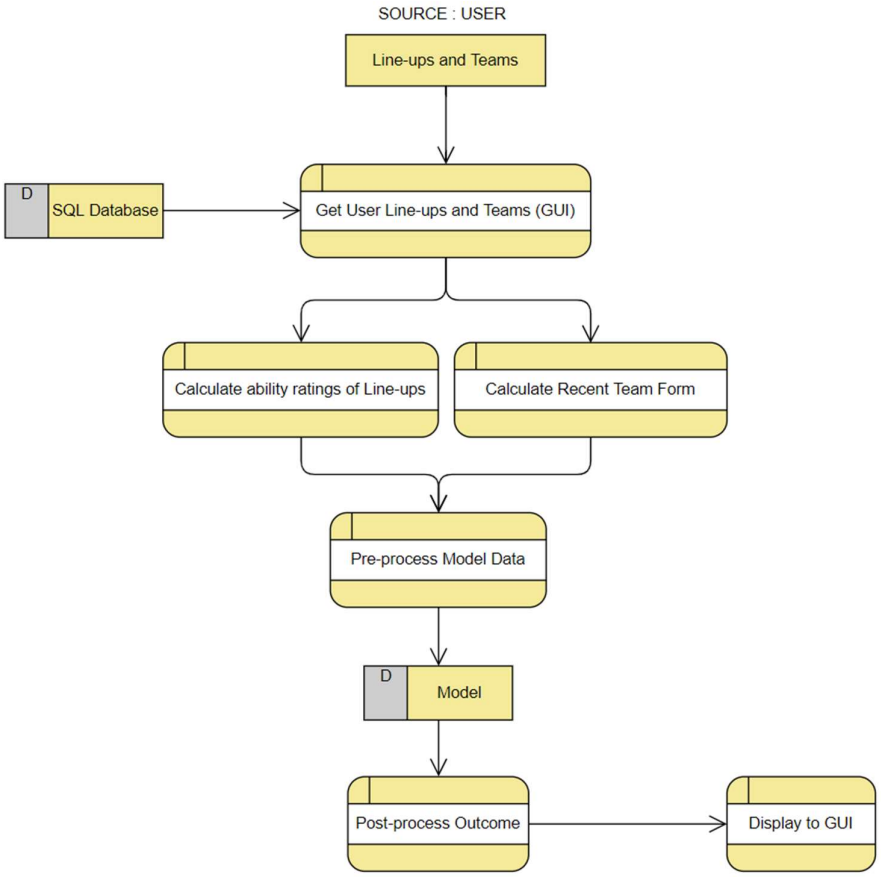
Calculating the strength of a line-up also uses the data already within the tables. Each player has a rating associated with them, each rating details a rating from 0-100 of their relative 'quality' as a player. The rating system is entirely based on secondary data produced by EA Sports. Each player in the team can have their rating associated with them to produce a set of 11 players. As the model will require a range of data as inputs, breaking the team down into positions on the field could increase the range of metrics while producing more in-depth analysis of a line-up. An example of a line-up is seen below as this vector, with each position is split up based on their relative field position. This example is from a Fulham FC line-up from the 2018/19 season:

Goalkeeper			*Defensive*				*Midfield*				*Offensive*	
80		**72**	**75**	**73**	**75**		**78**	**82**	**79**	**79**	**76**	**76**

In conclusion, the metrics used will be; four different line-up ratings based on the player's primary position of play and a single recent form metric. Across the football field of play, there are four general positions: goalkeeper, defence, midfield and forwards. The average of each position will be taken. The recent form metric will be the mean points a team achieves per game in the league across the last 30 days.

The choice of metrics has some limitations. Such as matches played within the first 30 days of the season will have a limited recent form scope. However, the players in Premier League teams can differ drastically through a summer transfer window. This means it is more informed to ignore last season's results. The other limitation may rise from the goalkeeper rating as this is only based off a single player, although a goalkeeper can be very influential into the result of a match.

Prediction Flow



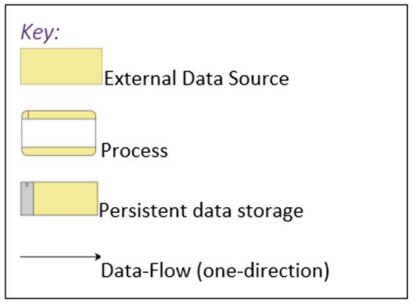


FIGURE 12

Collecting Line-ups and Teams (GUI)

The system is created with an end-user in mind. The user will be able to enter line-ups to receive a specific prediction, as stated in User Story 2. The exact design of the GUI will be illustrated in a later design section. A GUI is important as it supplies the trained model with data to produce a prediction. This GUI has a link to the players database to suggest and validate player entry.

Calculating Metrics and Model Interaction

Methods which calculate strength of a line-up in the training flow will be utilised to analyse the inputted line-ups. This will form the various metrics which the model has been trained to classify. The recent form is calculated by the most recent Premier League results of the selected teams which are present in the table at that time. These metrics are processed by the model pre-processing mapping identically to the training flow. The model is already trained so once these metrics are entered the model should return a classified result.

Displaying the Result

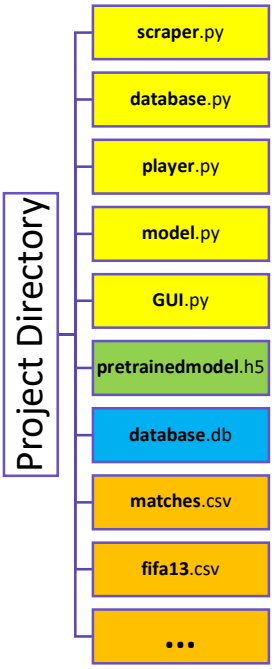
Once pre-processing is complete the GUI will display to the user the classified result. This will use Model-View-Controller (MVC) architecture to ensure the solution is robust and the user has no direct contact to the model and databases.

File Organisation

As this is a data science project there are many datasets. All raw data will be held in CSV (comma-separated value) files. This data is transferred to a relational SQLite database in a .db file. A single csv contains all the scraped match and line-up data (matches.csv). Each release of the FIFA player ratings is scraped from sofifa.com and stored in its own CSV file (fifa13.csv, fifa14.csv, etc).

The inclusion of a user-interface means that a pre-trained model must be stored to allow consistent and instant predictions. A model will be saved to a file after being trained and tested. This will be a .hdf5 file to store a checkpoint of models such as neural networks with TensorFlow.

All data files are held in the same directory as the 5 program files to allow direct accessibility.



Use of Libraries

As this project includes some complex processes, third-party libraries are used to enable the solution. The following libraries will be used in the implementation:

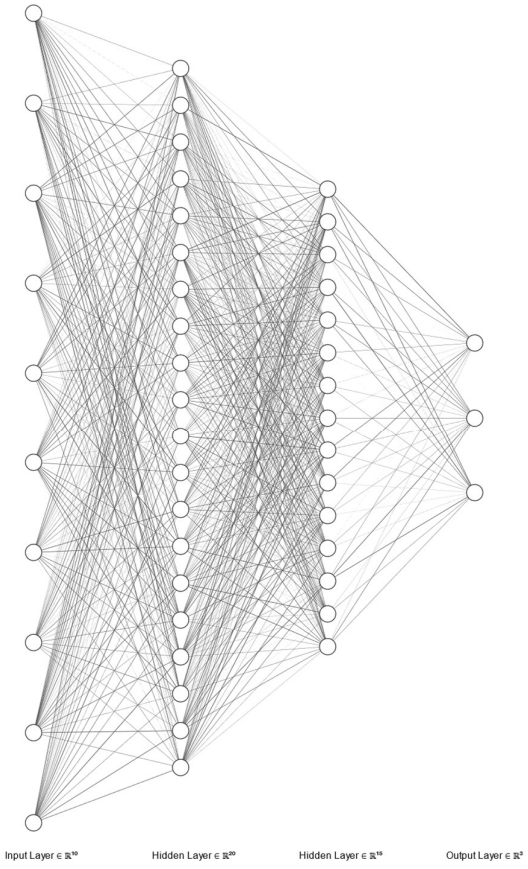
Requests (v2.22.0) – HTTP library using Apache2 to serve the HTTP content of webpages.
BeautifulSoup4 (v4.7.1) – HTTP library used to parse and navigate tags to extract content.
Selenium (v3.141.0) – Webdriver library used to interact with webpage elements to change content present on the page.
re (v2.2.1) – Regular expression library can be applied to strings to for validation or generation.
SQLite3 (v2.6.0) – SQLite (v3.21.0) library to create relational databases and perform queries.
Jellyfish (v0.7.2) – String similarity algorithms, includes implementation of Levenshtein distance.
NLTK (v3.4.4) – Natural Language Toolkit, used for word tokenising.
NumPy (v1.16.4) – Multidimensional array support with mathematical operations.
TensorFlow (v1.1.4) – Deep Learning library, used to create multi-layer perceptrons (Neural-networks)
Keras (v2.3.1) – Wrapper for TensorFlow, more intuitive syntax to interact with TensorFlow
Matplotlib (v3.1.1) – Library to create graphs for benchmarking.
appJar (v0.94) – Graphical User Interface library with widgets and a platform for the user to interact with.

Model Design

Objective 7 requires a prediction model which is given numerical metrics and makes an informed prediction through a machine learning technique. In this instance, machine learning is used to produce a generalised prediction model based on historical data. This project will use a feature vector of the line-up and recent form metrics from the home and away team as training data, each vector is labelled with the result of the game (e.g. home win) thus making the training process – supervised. There will be a 70-30 split between the testing data and training data.

In the analysis of current systems, both favoured the use of artificial neural networks (ANN) or Deep Learning. This model will use a feedforward multilayer perceptron to transform the inputs into a classified outcome. A perceptron is a model which takes numerical inputs and applies a weight to each input through multiplication, then the weighted inputs are summed, this value is inputted into an activation function. An activation function is a non-linear function to produce a normalised output.

In a neural network, a perceptron is a single hidden layer and a neuron is a single unit. 'feedforward' means there are no loops in the network, therefore the data is always heading towards the output. When multiple perceptrons are concatenated to form a feedforward ANN with one hidden layer; the non-linear nature of the activation functions allows any mathematical function to be approximated, this is called the 'universal approximation theorem'. This is what gives neural networks their flexibility. The number of hidden layers and number of neuron units should be in proportion with the number of inputs, however this is often an arbitrary amount. The following network takes the ten metrics as inputs and could be used to classify a result.



Weights and bias can be thought of as the importance of an input. When an ANN is created, all weights are a pseudorandom value. The 'training' of a network is simply the alteration of weights in backpropagation to minimise loss. Cost is a value given by a cost function which quantitatively deduces how far the current model is from the optimum. The process to find these optima is called gradient descent. To explain backpropagation, the process is mathematically derived over a single neuron with a single input:

The weighted input - Z, is the product of the input - x and a weight – ω.

EQUATION 3

$$Z = x\omega$$

$$Z'(\omega) = x$$

An activation function – R, is applied to the weighted input. The ReLU function is used as an example.

EQUATION 4

$$R = \max(0, Z)$$

$$R'(Z) = \begin{cases} 0 & Z < 0 \\ 1 & Z > 0 \end{cases}$$

A cost function – C in this example is Mean Squared Error (MSE).

EQUATION 5

$$C = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$C'(\hat{y}) = (y_i - \hat{y}_i)$$

All the previous functions are applied in the following order.

EQUATION 6

$$Cost\ value = C(R(Z(x\omega)))$$

The derivative of this composite function is given by the chain rule. In large networks, these calculations become very large, thus 'memoization' is adopted to cache repeated calculations.

EQUATION 7

$$C'(\omega) = C'(R) \cdot R'(Z) \cdot Z'(\omega)$$

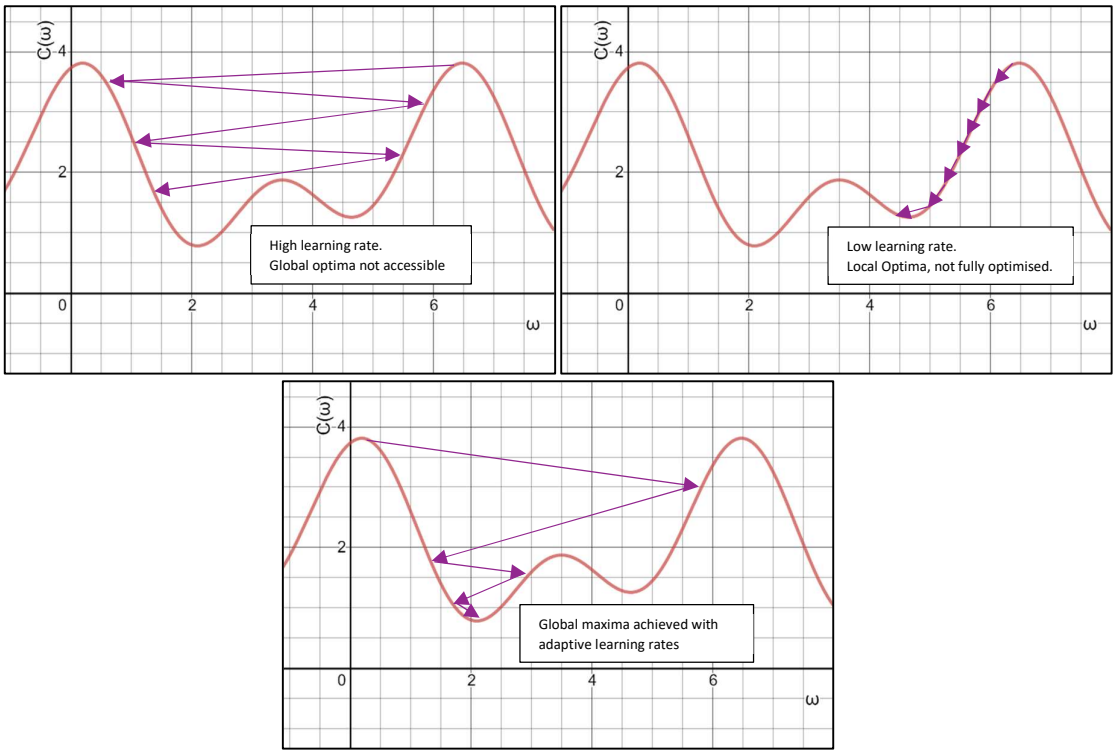
$$= (y_i - \hat{y}_i) \cdot R'(Z) \cdot X$$

The derivative of the cost function in respect to a single weight enables gradient descent. A weight is altered, and the gradient of the cost function is observed. The goal is to be as close to the minimum value of the cost function, where the shape of the function is unknown. The gradient descent process takes a trial and error approach. A step is a movement on the x-axis, in this case – weight. A step is a vector meaning it has a direction (determined by gradient of cost function) and a magnitude (learning rate). During gradient descent the learning rate is altered using an optimiser to efficiently reach the minimum cost.

Convergence is where the weights in the network become appropriately set to be treated as a 'trained' network. An optimiser adapts the learning rate to precisely reach this point. Different optimisers have different properties meaning they reach convergence at different rates, some will fail to reach the global optimum of the function due to local optima. A learning rate too large can cause oscillation around the ravine of the optima, meaning the optima is not accessible by this learning rate. Alternatively, a learning rate too low can take extreme amounts of steps to converge. Additionally, low learning rates are susceptible to being trapped in local optima.

The following graph shows how different learning rates effect the rate of convergence.

FIGURE 13



A stochastic gradient-descent optimisation process can be mathematically presented in the following equation. The magnitude of the next weight is calcualted based, this rule is applied until convergence.

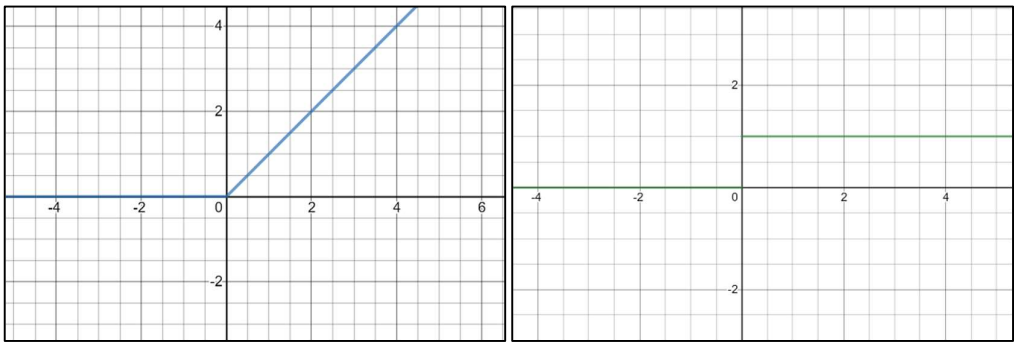
EQUATION 8

$$\omega_{n+1} = \omega_n - \alpha * \nabla_\omega \sum_{1}^{m} C_m(\omega)$$

Essentially, this rule takes the current weight and subtracts the product of the learning rate – α ($\alpha \in \mathbb{R}$) and the gradient of the cost function at that weight. This indicates the purpose of the learning rate as a large value of α will dramatically alter the weight.

To allow a network to be optimised, several steps must be made. A training set of raw data is fed through the network several times. The number of times the entire training set passes through the network is called the epoch count. During training, samples of data are batched together, where the size of the batch is the 'batch size'. The loss is calculated for each sample and combined for a single batch, then the weights are adjusted in a step. The number of steps is given by integer division of the total number of samples by the batch size. For example, if a training set consists of 2000 matches and the batch size is 32; the number of steps is 62.

An activation function can take many forms; ReLU, sigmoid , tanh, etc. Each function has its benefits. Generally, a function should be selected based on how well it optimises accuracy. The ReLU (Rectified Linear Unit) function is graphed below, with its derivative. ReLU is the most popular function as its simplicity means the gradient does not need to be calculated as its derivative is a constant. This predictable gradient means backpropagation can be performed in less computational time. Furthermore, ReLU solves the 'vanishing gradient problem' where the derivative of graphs such as sigmoid is small meaning that in large networks 'dead neurons' appear as the gradient used in back propagation tends to zero. Despite this, ReLU has a known issue called 'dying ReLU' where if a weighted sum of the inputs is negative, 0 is the normalised value. This means no propagation occurs after, thus making it a dead neuron. The solution is called 'Leaky ReLU' where the negative-x values are given a slight positive gradient to allow propagation of negative values.



Database Design

Database Overview

The system has a great demand on data, verging on Big Data. Large volumes of data require a medium of organised storage which allow efficient retrieval of data stored in the program. The datasets which are handled in this project have thousands of data entries; the football dataset contains around 17,000 players in total. The two subjects that the data covers are matches with results and starting line-ups and the players dataset, these have common themes which allow them to be interlinked with their data. The vast amount of data associated with a single player or match result means normalisation is a key method to simplify and decompose the large datasets.

Using Excel, example data has been manually entered from 'sofifa.com/players' into different tables in order to encourage normalisation and identifying relationships between interlinked tables. Please note that columns and format of the tables may not be identical to what is found inside the database.

PlayerID	ShortName	Season	Rating	ClubID	Position
1	S.Rico	18/19	80	1	Goalkeeper
2	C.Christie	18/19	72	1	Defensive
3	D.Odoi	18/19	69	1	Defensive
4	A.Mawson	18/19	75	1	Defensive
5	M. Le Marchand	18/19	75	1	Defensive
6	C.Chambers	18/19	75	1	Midfield
7	J.Seri	18/19	82	1	Midfield
8	A.Schürrle	18/19	78	1	Midfield
9	T.Cairney	18/19	79	1	Midfield
10	R.Sessegnon	18/19	75	1	Midfield
11	A.Mitrović	18/19	76	1	Offensive
12	A.McCarthy	18/19	77	2	Goalkeeper
13	C.Soares	18/19	77	2	Defensive
14	M.Yoshida	18/19	75	2	Defensive
15	W.Hoedt	18/19	76	2	Defensive
16	M.Targett	18/19	71	2	Defensive
17	M.Lemina	18/19	78	2	Midfield
18	P.Højbjerg	18/19	77	2	Midfield
19	M.Gabbiadini	18/19	77	2	Midfield
20	S.Armstrong	18/19	75	2	Midfield
21	N.Redmond	18/19	76	2	Midfield
22	C.Austin	18/19	76	2	Offensive

TABLE 1 – PLAYERS TABLE

ClubID	ClubName	ClubSeason
1	Fulham FC	18/19
2	Southampton FC	18/19

TABLE 2 – CLUBS TABLE

MatchID	Season	HomeClubID	AwayClubID	GoalDifference	HomeLineUpID	AwayLineUpID
1	18/19	1	2	1	1	2

TABLE 3 – MATCHES TABLE

LineUpID	PlayerID_H1	PlayerID_H2	PlayerID_H3	PlayerID_H4	PlayerID_H5	PlayerID_H6
1	1	2	3	4	5	6

TABLE 4 – LINE-UPS TABLE

NOTE: Columns omitted from line-ups table due to being 23 columns wide. Contains Player IDs of every home and away player.

Table 1 – Players Table

This table contains all of the players selected from the scraping process. A unique and referenceable identifier is assigned to each player. They have their season defined, this will be vital as there will be the same player from multiple seasons as their rating changes across iterations of EA Sports FIFA releases. They each have a rating which is a key measure for the team strength analysis. Each player has their current club listed as a Club ID which interlink with the clubs table. The positions listed in the table have been manually classified into a more general group to aid metric calculations. For example, Calum Chambers ('C.Chambers' in table) is listed as a CDM (centre-defensive-midfield) on the EA sports dataset but has been classified as a midfield player to have a fewer range of discrete groups allowing ratings to be calculated across the line-up not just as a single object. However, some players can be argued to fall into different positions; take André Schürrle who is listed as a LM (left-midfield) and grouped as a midfielder but during games he is undoubtably a forward as he moves forward to a left-wing position.

Table 2 – Clubs Table

Each player is part of a club, so this is a natural division to make. Players belong to different seasons so a solution to this is having a club defined by their season to prevent duplication or incorrectly selecting a player from the wrong season. This will also inform the user when entering the line-up into the GUI as they can be suggested players.

Table 3 – Matches Table

A method to combine match data and player data is to have table which contains matches. The match table and clubs table complement each other as a club participates in a match. Each row has the clubs involved and the goal difference in the perspective of the home club.

Table 4 – Line-ups Table

A key feature of this system is to train a model with metrics based on historical line-up ratings. Line-ups are a feature of a match so through normalisation a separate table is necessary. A line-up is composed of eleven players, this directly references the players table. Instead of having a home and away line-up, 22 columns (Home team starting 11 and away team starting 11) are dedicated to the home and away starting line-ups.

Entity Relationship Diagram (ERD)

The ERD diagrams describe the relationships between the tables introduced in the last section. Figure 6 displays the physical relationships between each table while Figure 7 displays how the table will be implemented, specifically handling the many-many relationships present by introducing intermediary tables.

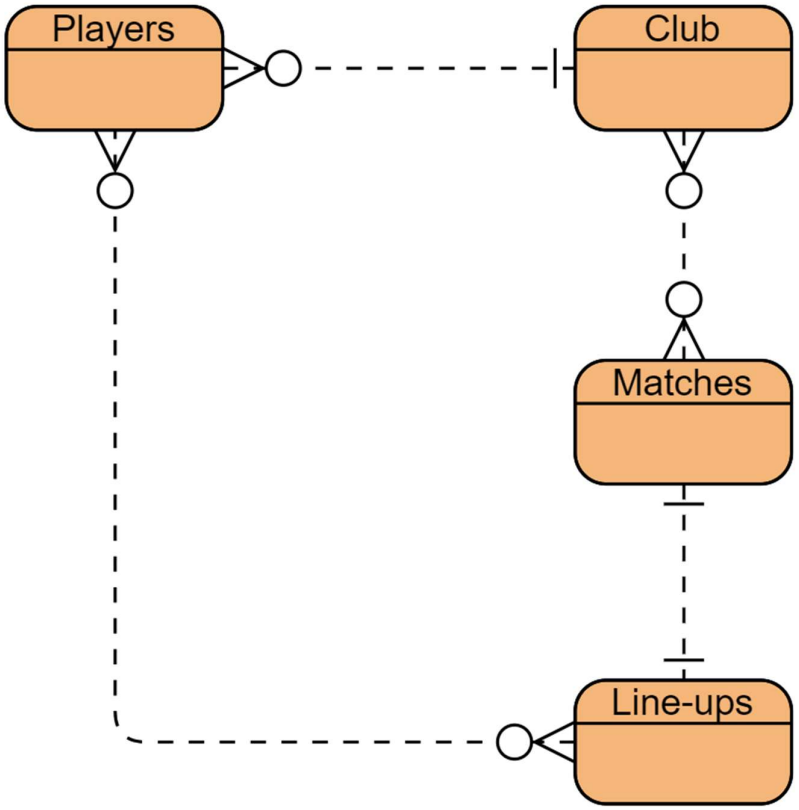


FIGURE 6

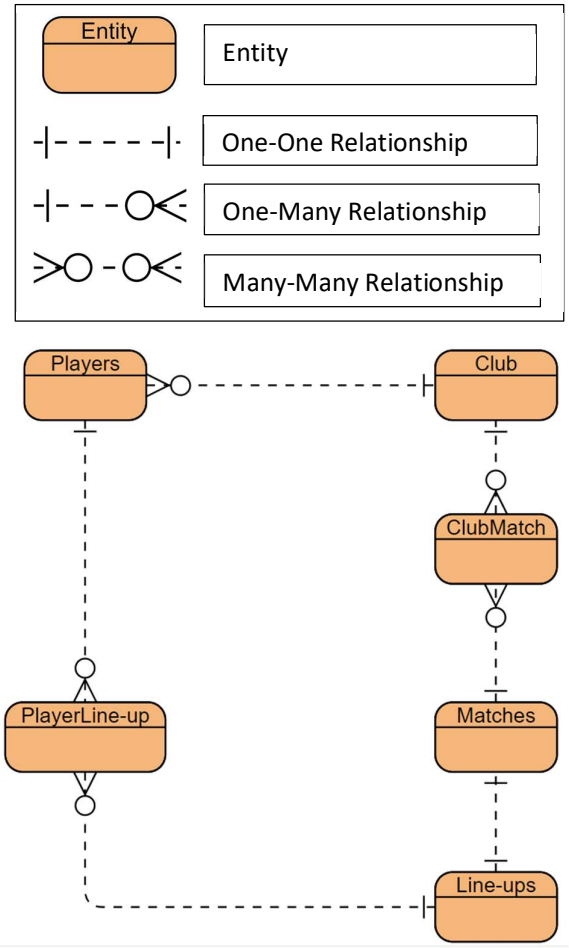


FIGURE 7

Graphical User Interface

The end-user asked for an offline graphical user interface (GUI) to enter all required information to receive predictions. Furthermore, the User emphasised the interface to be intuitive and interactive. Additionally, the User asked for entries to be edited or removed once entered.

Graphical User Interface Design

This section uses mock-ups to illustrate how the GUI will interact with the end-user. Drop-down selection boxes are an important tool to use as they provide the user with only relevant and available options. The interface makes use of pagination to allow subsequent drop-down boxes to be updated with only the relevant options. After the submit button is clicked, the values held in the input widgets are stored and the elements of the next page are produced.

All option boxes are filled with values which are fetched from the SQLite table using SELECT queries and the values are transferred across the MVC (Model View Controller) architecture. MVC is a suitable architecture because of its emphasis on separation of concerns between the end-user and the data processing. Additionally, MVC encourages encapsulation of the functions related to each part of the system which removes ambiguity in the code.

The first page asks the user to select the season from a drop-down box. The values are fetched from the database using an executed query which selects all available season in the database.

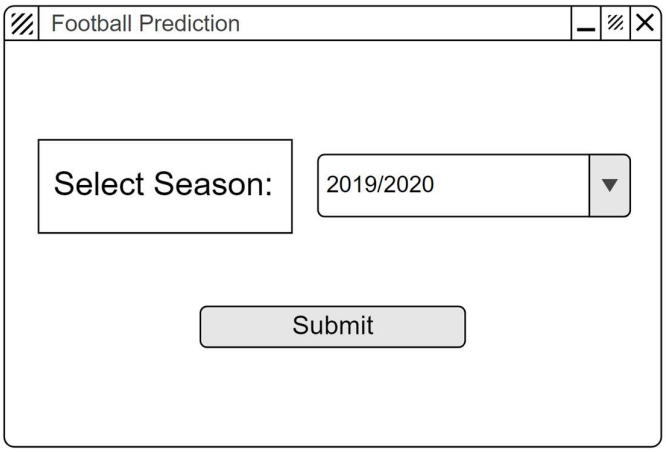


FIGURE 14

As recent form is a key metric which the User requests, the date is requested from the user. A date-picker widget can be used to allow the user to intuitively select the date of the match between the bounds of the season. Usually the match will be played the same day, thus the default value will be the current date. Additionally, this page allows matches to be predicted days in advance and previous matches to be tested.

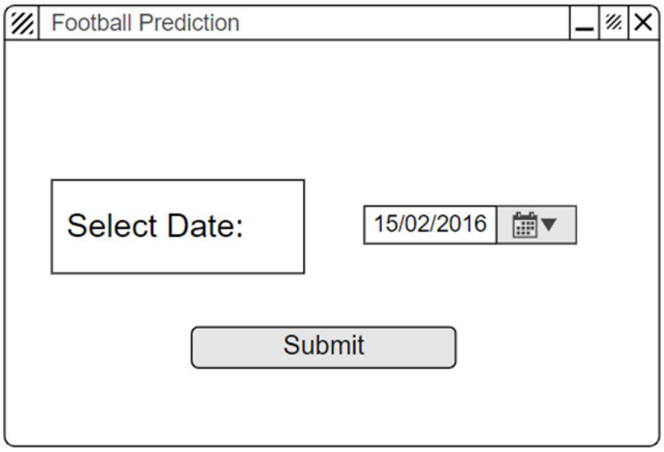


FIGURE 15

After the season and date is submitted, the name of every club in the Premier League from that season is fetched from the database. All 20 club names are fed into 2 dropdown boxes for the user to select a home and an away team.

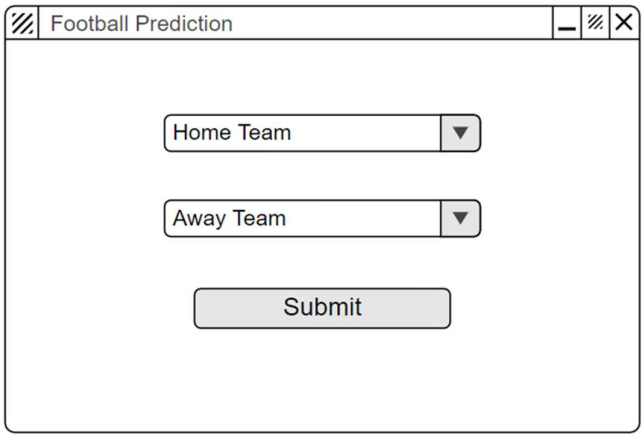
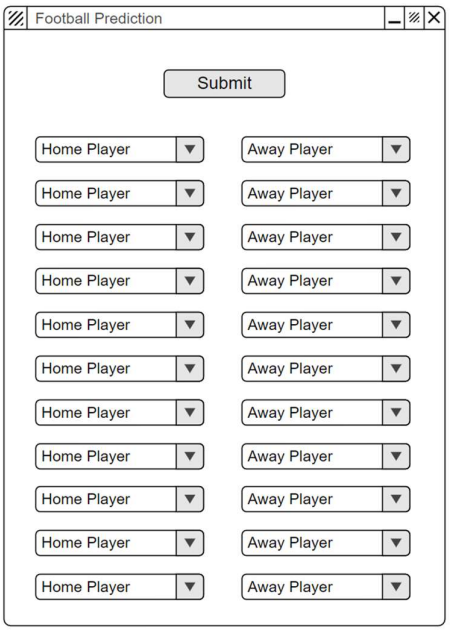
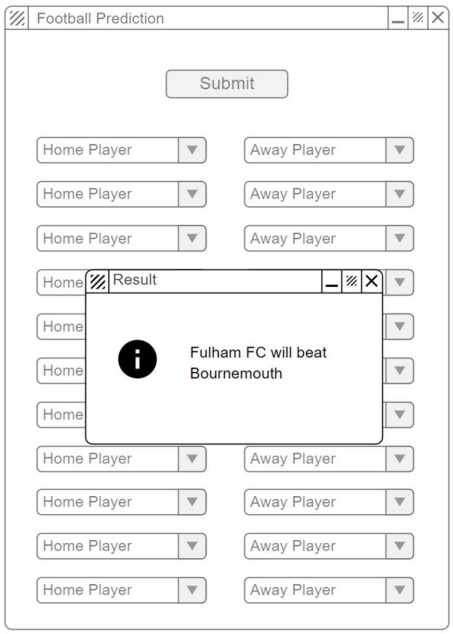


FIGURE 16

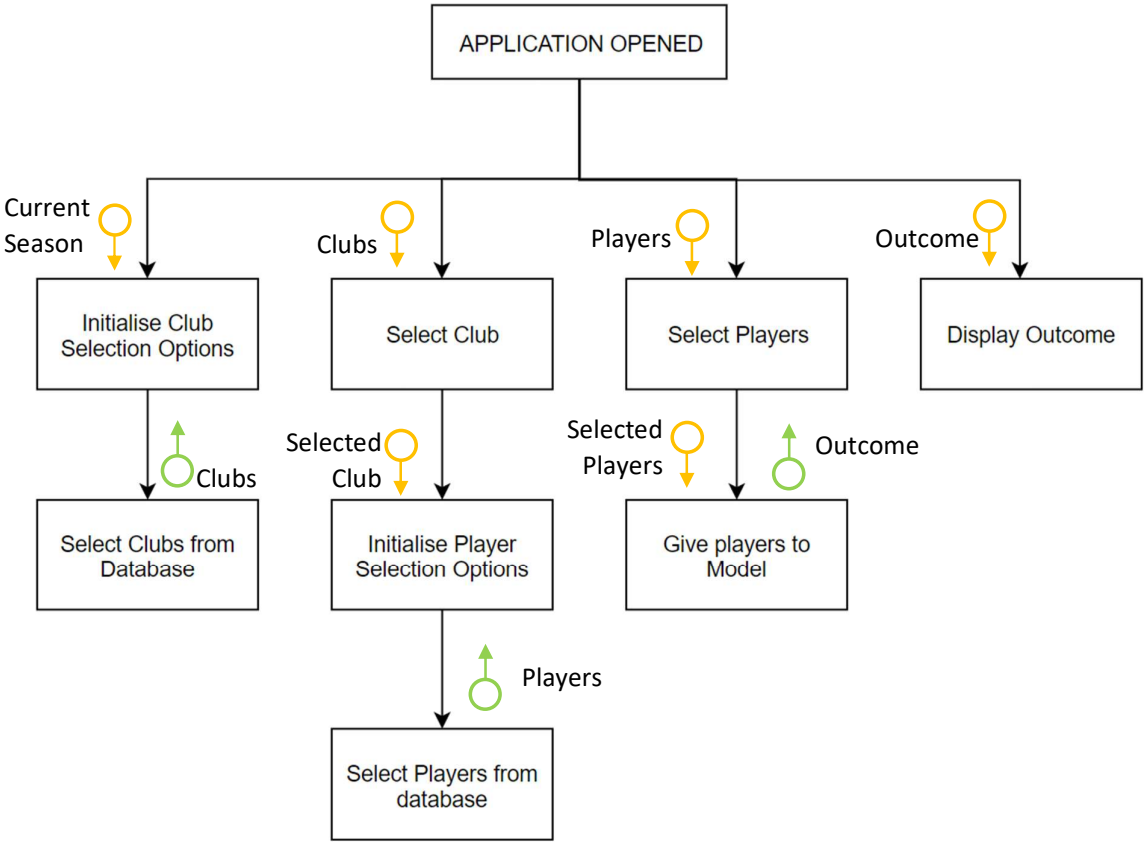
After the name of each club is submitted, the name is used in conjunction with the season to fetch all players from that season. The next page includes 2 sets of 11 drop-down boxes for the user to enter all 11 players for each team.



After 22 players have been entered correctly, the Player, Team and Match classes are used to instantiate different objects which handle the manipulation of the data. The Match parent class collates the player ratings and recent form calculated in the respective Team class. A feature vector is produced which is then fed to the trained TensorFlow model which will quickly produce a prediction. The model will output a home win, draw or away win. The view attaches the team names to the result and presents the outcome to the user in a dialogue box.



The following hierarchy chart is used to understand the processes which occur following user interaction.



Class / UML Diagrams

The system will be implemented entirely using object-oriented design. To illustrate each part of the system, UML class diagrams in this section detail the attributes and methods used in the implementation. The system is broken down into 5 separate parts: Scraper, Database, Player, Model and GUI. Each part has its own collection of classes.

To illustrate how the 5 collection of classes interact with each other, the UML diagram below shows the different associations in an abstracted format.

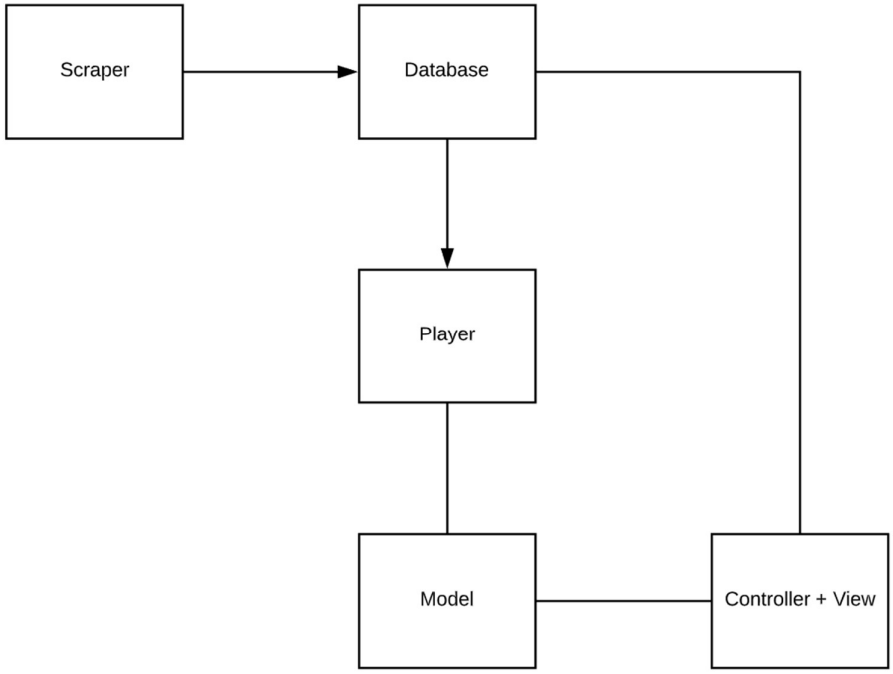


FIGURE 17

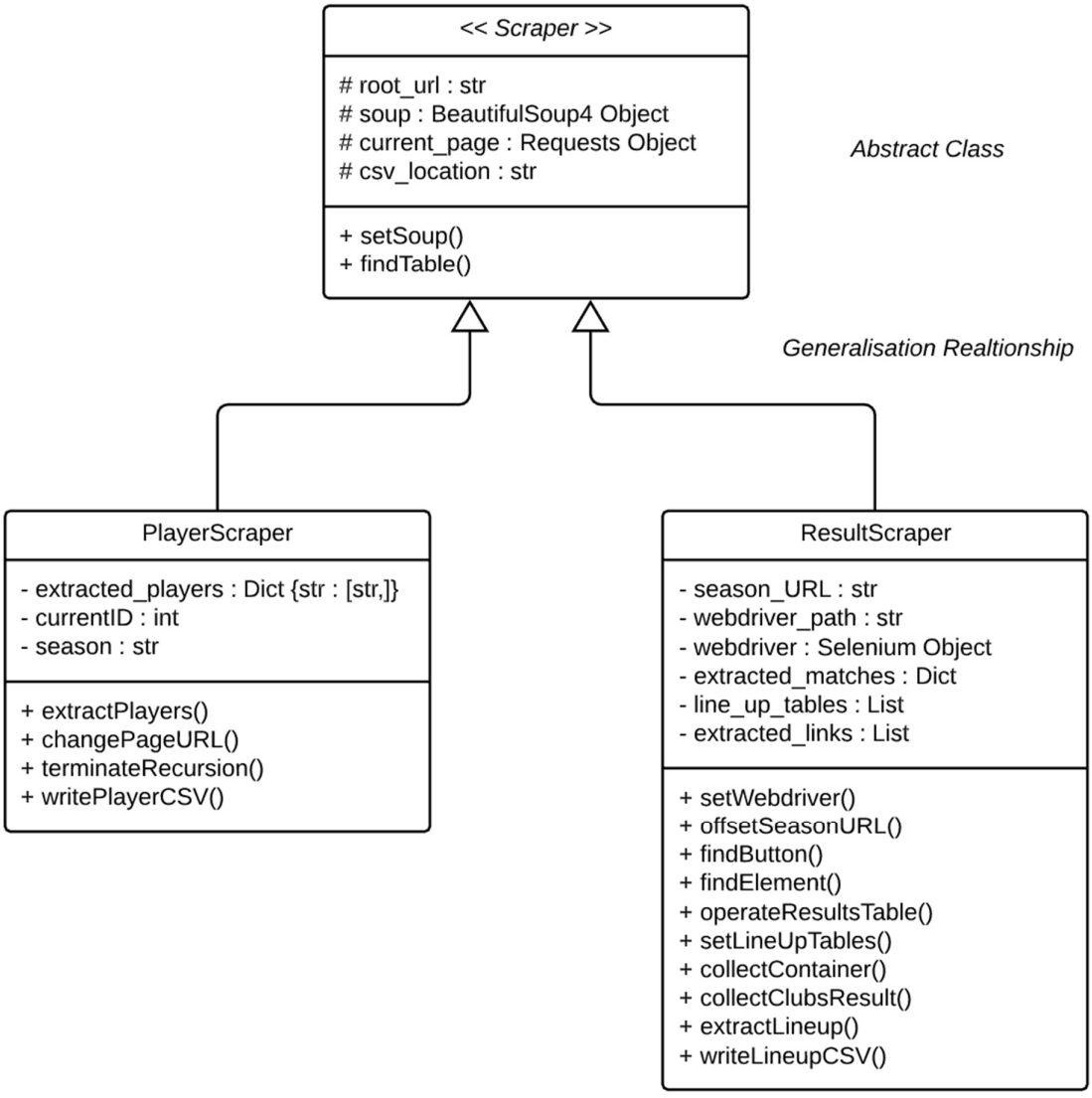
Scraper

The data source of this project is entirely from webpages. The collection of classes revealed below encapsulate all the necessary methods and data required to scrape both FIFA player data and recent Premier League matches.

The parent class Scraper is an abstract class meaning it is never instantiated. It contains methods and attributes which are utilised by both sub-classes.

The child class, PlayerScraper inherits the Scraper class which is used to set-up the BeautifulSoup library. This class encapsulates all the methods needed to web-scrape the HTML tables containing player information on 'www.sofifa.com'.

The other child class, ResultScraper also inherits all the Scraper class. ResultScraper utilises all of the methods to interact and web-scrape with 'www.soccerway.com' to collect Premier League line-ups and results.



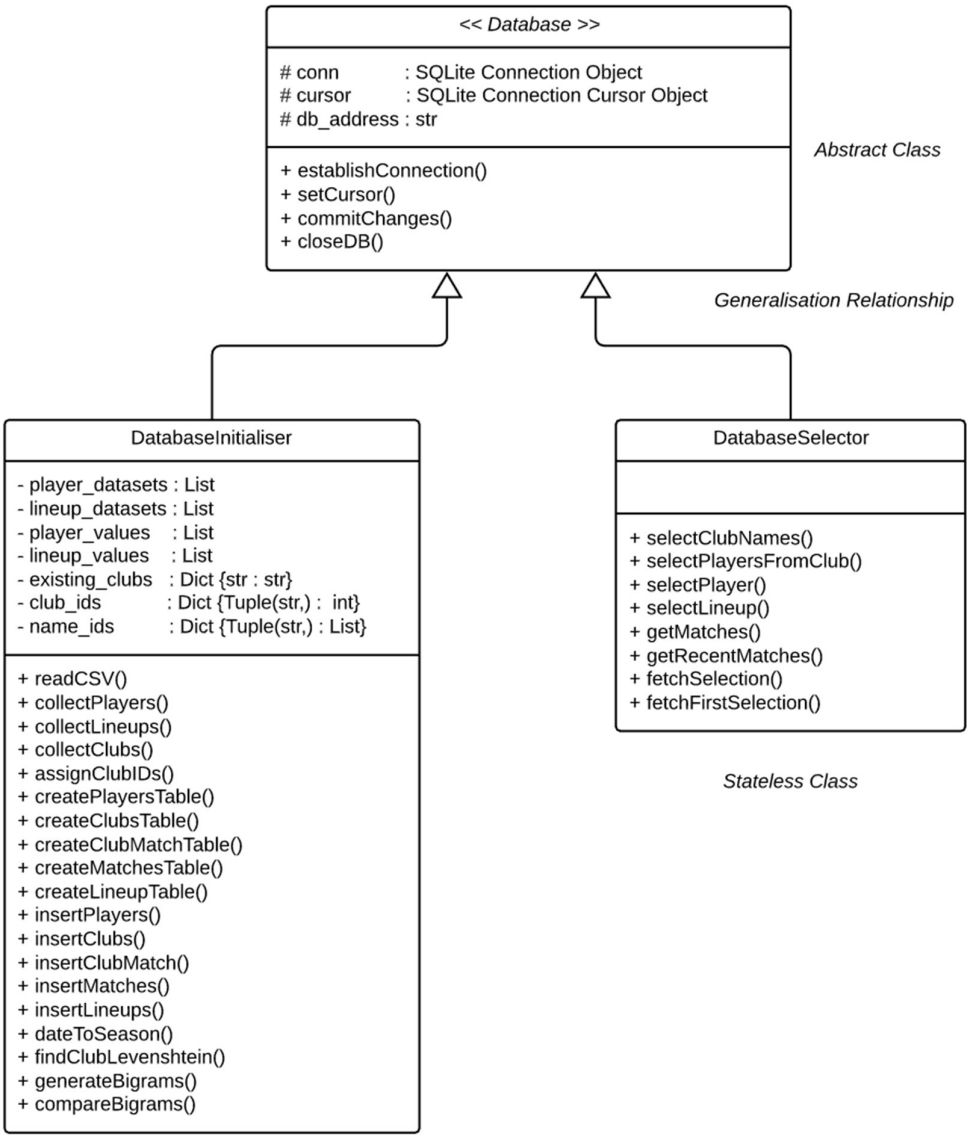
Database

An relational database will be used to store all of the player and match data persistently. The following classes are used to set-up the tables and their relationships, then fill the tables with values and provide an interface to access the stored data.

The class – Database – is an abstract class which contains the methods and attributes used to establish a database connection, resources common to both sub-classes.

The child - DatabaseInitialiser – encapsulates all the methods to create the SQLite relational database. Additionally, the class will read the CSV files created by the Scraper class. These values will be formatted and added to the database.

The other sub-class – DatabaseSelector – is used as an interface to extract values within the SQLite database. This is vital for subsequent functions including: model training/testing and user-interface.



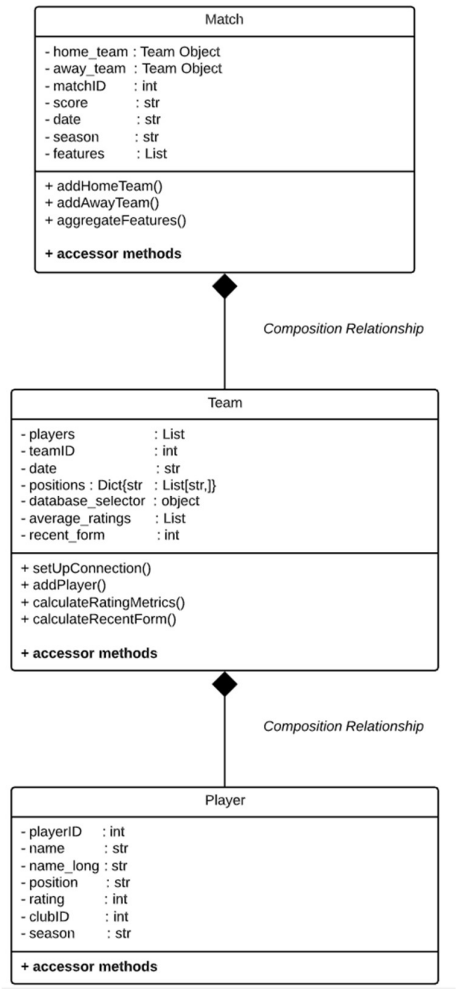
Player Instantiation

A match, team and a player are all physical objects which have associated data and functions. This makes it natural to create three separate objects. A single match is made up of two teams, each team is made up of eleven players. A Team cannot exist independent of a Match, and a lineup Player is independent of a Team. Thus, a composition relationship is used.

The – Match - class contains attributes related to the specific Premier League match. Additionally, both a home and an away team object is held. The method 'add[Home/Away]Team( )' is used to link each team to a match, satisfying the composition relationship. As this class is the highest level on the hierarchy, 'aggregateFeatures( )' is used to create a NumPy feature vector which is compatible with a TensorFlow model.

The – Team – class is composed of a collection of eleven Player objects. Also, this class is used to calculate the recent form and player rating metrics. As metrics require lots of data, this is the only class with a connection to the database. A composition relationship is formed with the – Player – class through the 'addPlayer( )' method.

Finally, the – Player – class contains all the data associated with a specific player.

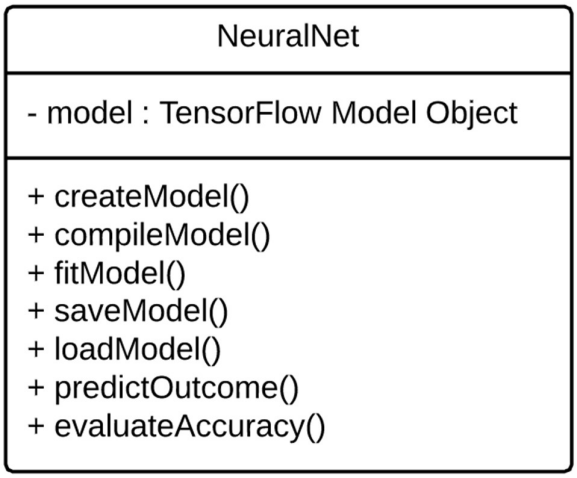


Model

Through analysis, a neural network model is most appropriate for the data. Furthermore, the Keras library will be used to implement the TensorFlow model. The following class acts as a wrapper on the Keras Library while providing a more optimised approach for a football prediction model.

The collection of feature vectors created by the – Match – class is divided to form a training set and a testing set. The training and testing set is given to the methods: 'fitModel( )' and 'evaluateAccuracy( )' respectively.

Once a model has been trained, the model can be saved to a file. The model can be loaded with all same weightings thus, ready to make predictions.



Example Data Structures

Feature vector

A feature vector can be thought of as an array (of multiple dimensions) containing data about an object. TensorFlow takes a feature vector in an input layer for neural network models; the vector must be a NumPy feature vector.

H – Home
A – Away
OVR – Overall average rating
DEF – Defensive average rating
MID – Midfield average rating
FOR – Forwards average rating

[	**H-OVR,**	**H-DEF,**	**H-MID,**	**H-FOR,**	**H-Recent form,**	**A-OVR,**	**A-DEF,**	**A-MID,**	**A-FOR,**	**A-Recent Form,**	**Result**	**]**

DATA STRUCTURE 1

The result column is only included during training to enable backpropagation as it is the parameter of the cost function, the column is omitted for testing.

Pseudocode

To directly inform the technical solution, specific algorithms and subroutines are expressed in pseudocode.

Recursive Scraping Solution

The first subroutine modelled is a method in the PlayerScraper class. The method is called extractPlayers() and is used to recursively web-scrape each page of the website 'sofifa.com'. The following pseudocode relies on both the Requests library to serve the HTML content of the page and BeautifulSoup to parse the various HTML tags present. Also, regular expressions are used to find and substitute elements in the URL to move to the next page. After analysing the HTML structure of the data table on the webpage, the implementation should follow very similarly.

To understand the pseudocode, knowledge of the following tags and regular expressions is a pre-requisite:

The tag – 'tr' is short for 'table row'
The tag – 'td' relates to a table header, or column
The tag – 'a' relates to an anchor element
The classes – 'col-name', 'col-oa', 'col-sort' relates to the properties of the table column
The class – 'nowrap' determines whether the content should wrap to a resolution

The presence of various tags and classes is used to navigate the HTML data before scraping specific content.

Regular expressions:
Pattern – A sequence of special characters used as a search term for pattern matching.
    In the pseudocode the following patterns are used:
        \d – relates to any digit character (0-9)
        \Z – relates to the end of a string
    Thus, \d+\Z relates to more than one (+ character) digit followed by the end of the string. This is used to adjust the page offset in the URL.
MATCH – Boolean operation which is True when the pattern can be matched in a string
FIND – creates a list of instances where the pattern is matched within the string
SUBSTITUTE – replaces instance of a pattern match with string

```
1.  root_url    : STRING <- "https://sofifa.com/players?offset=60"
2.  all_players : LIST   <- []
3.  METHOD extractPlayers() -> BOOLEAN:
4.      current_page <- REQUESTS GET PAGE(root_url)
5.      page_html    <- BEAUTIFULSOUP(current_page)
6.      IF REGEX MATCH https://sofifa.com/players\Z WITH root_url:
7.          RETURN TRUE
8.      table_tags <- 'table' tag IN page_html
9.      FOR EACH player_tag IN [FOR EACH tag CONTAINING 'tr' IN table_tags]:
10.         current_player : LIST = []
11.         FOR EACH attribute_tag IN [FOR EACH tag CONTAINING 'td' IN player_tag]:
12.             IF (attribute_tag HAS CLASS 'colname') AND (tag 'a' WITH CLASS 'nowrap'
                                                            EXISTS IN attribute_tag):
13.                 ADD attribute_tag TO END OF current_player
14.             ELSE IF attribute_tag HAS CLASS ["col", "col-oa", "col-sort"]:
15.                 ADD attribute_tag TO END OF current_player
16.          ADD current_player TO END OF all_players
17.     page_offset : INTEGER <- (REGEX FIND \d+\Z IN root_url) + 60
18.     root_url    : STRING  <- REGEX SUBSTUTUTE page_offset WHERE \d+\Z IN root_url
19.     extractPlayers()
```

Comments by line

3. Method of class PlayerScraper.
4. Uses Requests library to serve HTML data of the webpage given by root_url
5. Instantiates a BeautifulSoup object enabling parsing of HTML tags
6. Recursive base case uses regex to match the final URL (redirected to home page) to end recursion.
8. Assigns the HTML table tag to a variable, with all descendant tags. All tag parsing operations use BeautifulSoup methods similar to this.
9. Definite iteration through each instance of ‘tr’ tag in table_tags, each ‘tr’ tag relates to a player hence player_tag is the identifier.
11. Nested definite iteration through each instance of the ‘td’ tag in the current player_tag, each ‘td’ tag relates to each attribute on the table hence attribute_tag is the identifier.
12. Uses Boolean operators to identify the content of the current tag based on the presence of different tags and their classes.
17. Uses a regular expression to extract the current page offset in the URL, then 60 is added to move to the next page.
18. Uses a regular expression to substitute the current offset with the evaluated offset in line 17
19. A recursive tail-call to repeat the entire method for the next page of players.

n-gram String Similarity with Jaccard Co-efficient

As detailed in the analysis as a specific algorithm, string similarity is used as a error correction process. As the CSV datasets are from two distinct data sources, there are some differences in how a player name is formatted. This is a significant issue when entering line-up values into the database, where a player in the Player table cannot be matched with a player in the Line-ups table. The usual process is where a dictionary of players can be searched based on the player name as a key. This makes up about 98.5% of the 58,520 player entries into the Line-ups table. The remaining 878 entries will return a KeyError, hence the requirement for a heuristic approach to find the most likely matching player.

The following solution uses Jaccard co-efficient to assess the n-gram similarity of each name.

```
1.  METHOD generateBigrams(player : str) -> List [str,]:
2.      player_names   : LIST <- [FOR EACH word IN player]
3.      player_bigrams : LIST <- []
4.      FOR EACH name IN player_names:
5.          ADD [FOR EACH bigram IN name] TO END OF player_bigrams
6.              RETURN player_bigrams
7.  METHOD compareBigrams(players : LIST, team_ids : LIST) -> LIST:
8.      similar_players : LIST <- []
9.      FOR EACH player IN players:
10.         current_player_bigram <- generateBigrams(player)
11.         greatest_similarity : (INTEGER, INTEGER) <- (0, 0)
12.         FOR EACH club_player IN team_ids:
13.             player_bigrams : LIST <- generateBigrams(club_player[1])
14.             common_bigrams : INTEGER <- LENGTH([INTERSECTION OF player_bigrams AND
                                                    current_player_bigram])
15.             jaccard : REAL <- common_bigrams /
                                  LENGTH([UNION OF player_bigrams AND current_player_bigram])
16.             IF jaccard > greatest_similarity[0]:
17.                 greatest_similarity <- (jaccard, club_player[0])
18.         ADD greatest_similarity[1] TO THE END OF similar_players
19.         team_ids : LIST <-FILTER(team_ids WHERE player ≠ greatest_similarity[1])
20.     RETURN similar_players
```

Comments by line

1. Method of class – DatabaseInitialiser, the parameter – ‘player’ is a string of the full name of the player.
2. Generates a List of each name within player. E.g. ‘Luca de la Torre’ forms a list – [‘Luca’, ‘de’, ‘la’, ‘Torre’]. This is a necessary step as bigrams will be formed from each name without the inclusion of whitespace.
4. Definite iteration through separated name in the List – ‘player_names’
5. A list of bigrams is formed, and the elements are added to the List – ‘player_bigrams’
7. Method of class – DatabaseInitialiser, the first parameter ‘players’ is a List of players as strings who were returned a KeyError during the player dictionary lookup within a single line-up. The second parameter, ‘team_ids’ is a List of Tuples in the form [(#ID, #PLAYER NAME)], this contains the entirety of players in the current team.
9. Definite iteration through the list – players.
10. ‘current_player_bigram’ is given the value returned by ‘generateBigrams’ with the argument of the current player in ‘players’.

11. The tuple 'greatest_similarity' is used to recall the most similar player ID and its Jaccard co-efficient. (#PlayerID, #Jaccard)
12. Nested definite iteration is adopted to compare every player in every permutation of the two bigram sets (Cartesian Product: $-\{`players' \times `team\_ids'\}$)
14. A list is formed of each common value in each list of bigrams: ($player\_bigrams \cap curent\_player\_bigrams$). Finally, an integer value of the length is found.
15. Finally, the Jaccard co-efficient is calculated by dividing the number – 'common_bigrams' by the length of the union set of the two list of bigrams: ($player\_bigrams \cup current\_player\_bigrams$).
16. If the calculated Jaccard co-efficient is greater than the value held in 'greatest_similarity', the player ID and Jaccard is stored.
19. The player held within team_ids is removed to prevent multiple players matching to the same player.
20. Finally, the list of similar player IDs is returned from the method.

Example Queries

All data is stored in an SQLite database; data must be retrieved dynamically throughout the program. This section will outline the specific SELECT queries required and where they will be used.

QUERY 1

```
SELECT DISTINCT club_season
    FROM Clubs
    ORDER BY club_season DESC;
```

Query 1 is a simple query which makes use of the built in **DISTINCT** which truncates each instance of the same season to only the first instance. This will be used by the GUI to show the user a list of available seasons.

QUERY 2

```
SELECT home_clubID, score, away_clubID
    FROM Matches
    WHERE (home_clubID = clubID OR away_clubID = clubID) AND
(match_date BETWEEN
DATE(match_date, '-1 month') AND DATE(match_date, '-1 day'));
```

Query 2 is used to get match data from matches played over a previous month. As *SQLite* has no built-in data type for dates, we use the **TEXT** type with a date in the *ISO8601* format. The **BETWEEN** function allows dates to be selected between the date a month before and the day before the match. It is vitally important that the date range does not include the day of the match as that would make the testing phase unfair and would falsely improve the accuracy as the points per game would rise if the match was won.

Testing

The testing of this project requires both feature testing and benchmarking. This is because the quality of the solution must be assessed with the use of a heuristic prediction model. The model will be benchmarked with different metrics to analyse how it compares to existing models. Each objective created in the analysis phase will be evaluated to how well the technical solution achieves.

Video

The following video has been created to illustrate how the final production of the technical solution achieves each objective. The objective testing table has timestamps of where in the video the video is shown to be achieved.

- https://www.youtube.com/watch?v=8TZ96hMd8fM&feature=youtu.be

*The video is **only accessible** on YouTube through the **link above**.*

Benchmarking

As the neural network was created with TensorFlow, there are many built-in visualisation tools and metrics. The following metrics will be evaluated:

- Prediction accuracy
- Loss/Cost
- Validation accuracy by epoch
- Prediction accuracy by epoch
- Theoretical Pay-out from betting

Model Hyperparameters

The properties of the model used in the technical solution is a feed-forward neural network implemented in TensorFlow. The network receives 10 inputs and is fed to two hidden layers, with 20 and 15 units respectively. During testing it was clear that the number of units in each layer had minimal effect on the rate of convergence or prediction accuracy. A range of formations were experimented, two dense layers of 20 and 15 have 693 trainable parameters meaning the training process is not too intensive on memory and CPU usage.

Other hyperparameters:

- Training Testing set ratio: 70% training (1932 training matches and 828 testing matches)
- Activation Function: Leaky ReLU (used on both hidden layers), SoftMax (Used on the final output layer for classification)
- Cost Function: Sparse Categorical Cross Entropy
- Batch size: Step every 32 datapoints
- Optimiser: Adam
- Metrics: Accuracy, Loss (Evaluated after each epoch, used as validation)
- Epoch count: ≈15-40

Training Loss/Cost

Cost gives you the relative distance to convergence where the model is optimised. The cost function can be graphed to view the rate of convergence. Using TensorBoard, the cost is plotted against the epoch.

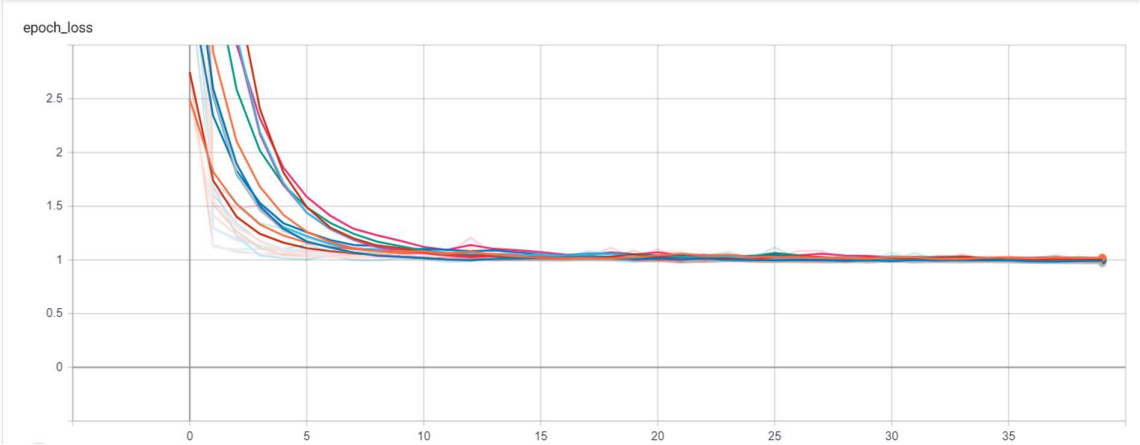


FIGURE 18

In figure 18, twelve different model iterations are plotted up to 40 epochs. It is instantly clear that the model plateaus very quickly at approximately 15 epochs with a loss of 1.0. This fast rate of convergence makes the training process meaningless after 15 epochs. This suggests that the Adam optimiser is suitable for the data.

Accuracy

Validation Accuracy

A validation set is a subset of the training set reserved for guiding the choice of hyperparameters before the training phase. Additionally, the set is used to avoid overfitting where the model becomes too specific to a training set rather than generalised.

The following graph produced in TensorBoard plots validation accuracy against epoch:

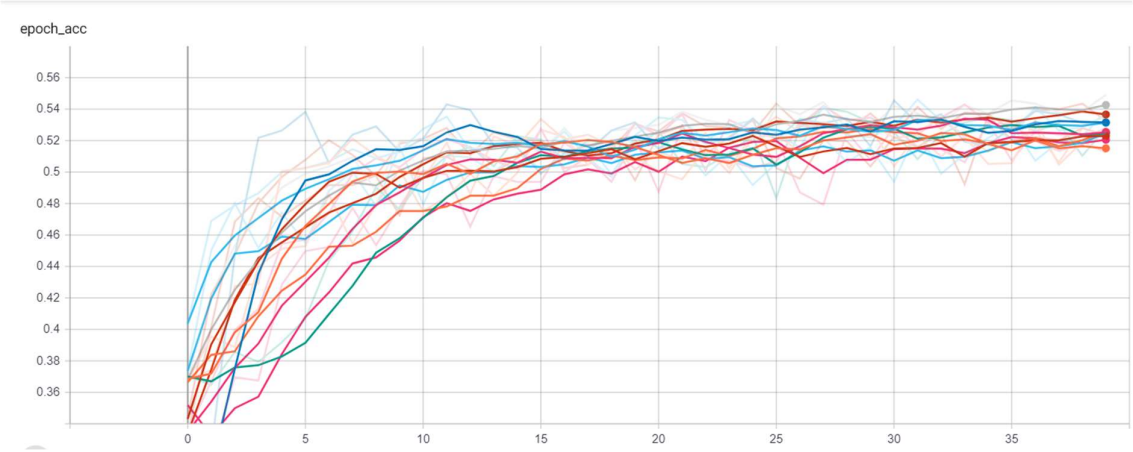


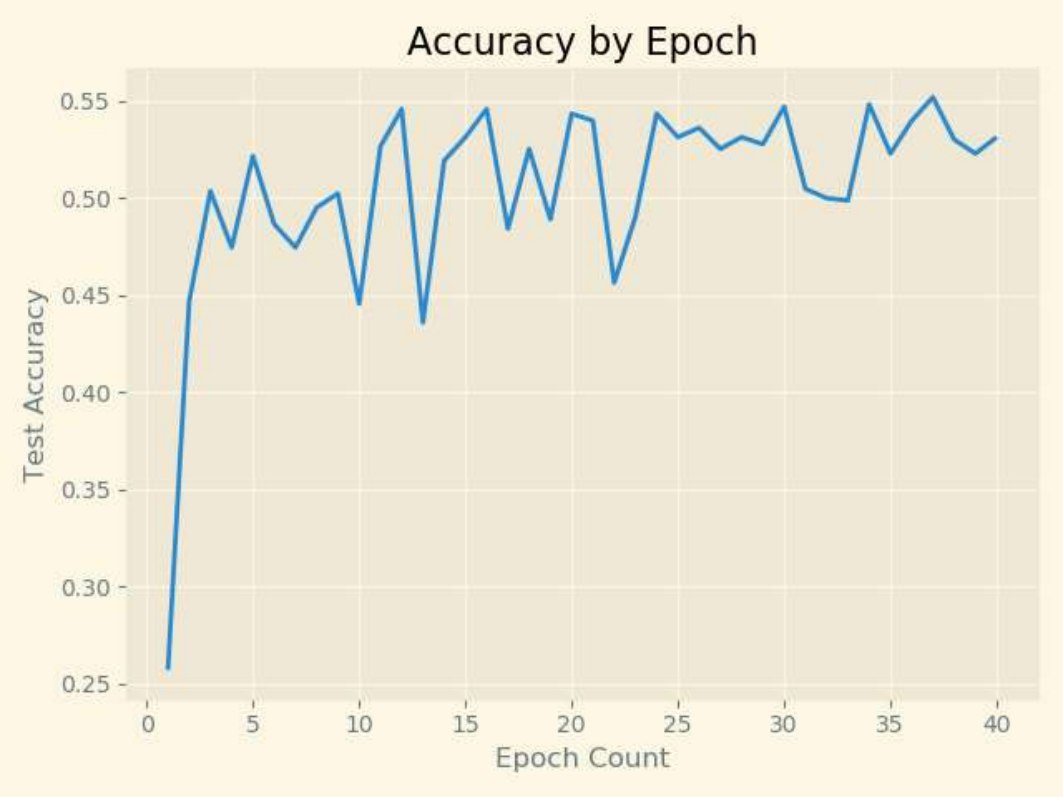
FIGURE 19

The graph shows very similar properties to the cost graph. Again, the accuracy plateaus at approximately 15 epochs. There is no need to continue the training process after, as overfitting may be a risk. However, the validation accuracy is similar to the prediction accuracy (≈58%) meaning it is unlikely that overfitting has occurred.

Prediction Accuracy

The prediction accuracy is evaluated using 828 randomly selected Premier League matches $(testing\ match\ \notin Training\ Set)$. After configuring hyperparameters the peak prediction accuracy was 60.2%. As the model is trained with a randomly shuffled set the accuracy will change on each run; the mean accuracy was 56%. This is a very respectable prediction accuracy, higher than Corentin Herbinet's model from the analysis phase. This is actually greater than the proportion of BET365 lowest outcome odds which are correct (52%).

Once again, a graph is plotted with the prediction accuracy against the epoch. Using Matplotlib, the following graph was constructed.



Pay-out (Pay-out test in APPENDIX A)

Another popular metric to assess the accuracy of models while giving context to the performance is pay-out. It involves making a series of £100 bets with betting companies over the predicted outcome. Despite pay-out being very subjective, it is still very popular.

To test pay-out, the historical odds from six different match betting companies offering match outcome were recorded in a CSV file (provided by 'football-data.co.uk' [13]). The model is given the line-up of that match and will make a prediction. The betting company with the highest odds for that prediction is selected. If the prediction is correct, the £100 bet is multiplied by odds and the initial stake is added. This was applied to 60 different Premier League matches from the 2018/2019 season. Bear in mind that the model was not trained on any match that season to prevent the model making bias predictions.

Results

Game-week	Correct Predictions	Pay-out	Return on Investment (ROI)
1	7/10	+ £91.00	+9.10%
6	5/10	- £118.00	-11.8%
12	7/10	+ £416.00	+41.6%
16	6/10	+ £145.00	+14.5%
24	6/10	+ £218.00	+ 21.8%
38	3/10	- £561.00	- 56.1%
OVERALL	34/60 (56.7%)	+ £192.00	+ 3.18%

Objective Testing

Each objective will be assessed and evaluated based on how well it achieves it.

"To produce a system where a user can receive a realistic prediction of the outcome of a specified Premier League football match".

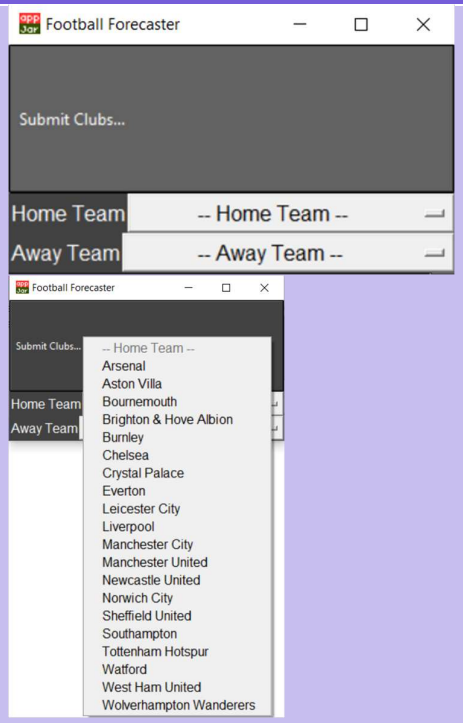
Objective number	Objective	Input data	Expected outcome	Actual outcome	Evaluation	Timestamp
1	"Gather player data from EA sports FIFA databases across available previous seasons."				All achieved	**0:10**
1.1	"Request the HTML data of the web-page holding players with the required player attributes"	"www.sofifa.com/players" Is given as the URL attribute in the constructor for the Scraper class.	Requests library will fetch HTTP data. The webpage data is received with status code 200 (OK).	Status code 200, page data stored in Requests object	Objective achieved	0:35
1.2	"Parse the HTML tags to locate the name of a player and their football attributes"	Page data from objective 1.1, the page contains player data	The page data becomes a BeautifulSoup object which can be parsed AND the tags can be located	BeautifulSoup object returns no exceptions. A Dictionary is formed with player name as key and a value of the list of player attributes	Objective achieved	0:45

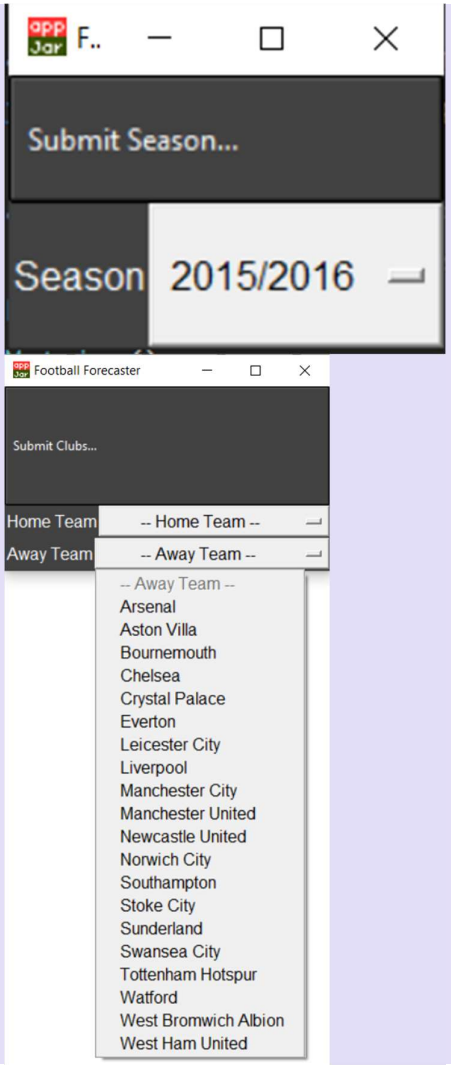
1.3	"Navigate through different pages on the domain to iteratively collect the entirety of the player data."	Navigable page URL from objective 1.1	The program moves through each page of the table and extracts all each player	Recursively moves through each page and adds to the existing dictionary holding player data.	Objective achieved, uses recursion instead of iteration.	0:40
1.4	"Write player data to a CSV file."	Dictionary containing each player and their attribute values	A correctly formatted CSV file with all players	8 different CSV files from FIFA13-FIFA20 players. Correctly formatted.	Objective achieved	1:09
2	"Gather Premier League results and line-ups across available previous seasons."				All achieved	**1:26**
2.1	"Request the HTML data of the web-page holding Premier League results and respective line-ups"	"www.soccerway.co.uk" URL of specific page containing results table.	Requests library will fetch HTTP data from each match page. Each webpage is received with status code 200 (OK).	List of 2660 URLs relating to each match from 8 seasons. Status code 200 on all 2660 , page data stored in Requests object.	Objective achieved	2:25
2.2	"Parse HTML tags to locate	URL relating to match page	The page data becomes a	BeautifulSoup object returns no exceptions. Each tag containing line-ups and match data is	Objective achieved	2:25

	and extract specific data about the match and line-ups.”		BeautifulSoup object which can be parsed AND the tags containing matches can be located.	located and extracted. All stored in a dictionary.		
2.3	“Navigate through different pages on the domain to iteratively collect each game across multiple seasons.”	The root URL of the results page.	Changes the content of the navigable URL to change season, then extracts matches (2.1, 2.2)	Seasons are navigated through by changing URL	Objective achieved	2:40
2.4	“Write results and line-ups to a CSV file.”	Data structure containing clubs, results, date and line-ups.	A CSV file is created with each row containing the all information about a single match.	A single CSV file was created aggregating all match data from every available season	Objective achieved	2:59

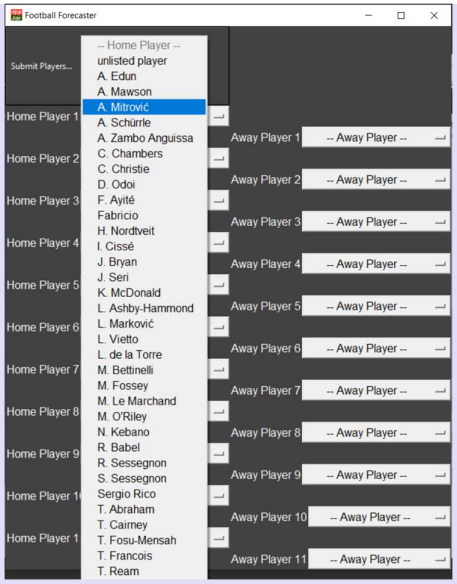
3	**Store player data, results and line-ups persistently in a database.**				**All achieved**	
3.1	"Create a normalised SQL database with interlinked tables."					**3:11**
3.1.1	"Create each table with columns for all data required for later processes in the system."	N/A	SQLite command executed to create table with suitable number of columns to hold CSV data	Command executed and creates 5 data tables.	Objective achieved	3:15
3.1.2	"Interlink each table with the required relationship (see ERD diagram in design phase)."	N/A	Foreign Keys are identified and reference other tables. Including composite table to handle man-many relationship.	Implemented exactly as ERD states.	Objective achieved	3:15

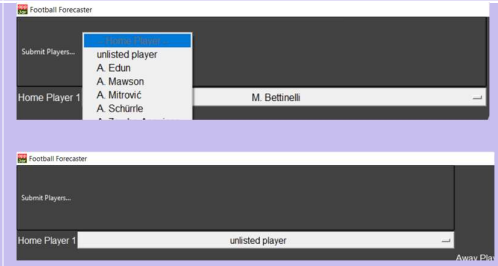
3.2	"Insert scraped data into the corresponding table."	CSV player and match data	Every value from CSV files are stored in database	All 5 tables filled with appropriate data.	Objective achieved	3:15
3.2.1	"Assign each entry into the table a unique ID number to act as primary and foreign keys."	Dictionaries containing CSV player and match values	A unique primary key is assigned	Each value given a unique primary key. No SQLite unique constraint errors.	Objective achieved	3:15
3.2.2	"Link each player held on the player dataset to the same player found on the line-up dataset (Player data in different formats)."	Player name strings; one sourced from "sofifa" and the other sourced from "soccerway".	The two strings are compared, and a normalised form is reached.	98.5% of player names use dictionary lookup. Remaining 1.5% use string similarity.	Objective achieved	3:15

4	**“Create an interactive GUI (Graphical User Interface) which allows the user to enter clubs and line-ups.”**					**4:18**
4.1	“Deploy an appropriate interface for the user to enter a home and an away club”	User opens user interface, enters season and date	Two drop down box to enter each club from that season		Objective achieved	4:53

4.1.1	"The user-interface must have a drop-down menu of all Premier League clubs."	User opens user interface, enters season and date	All 20 Premier League teams from that season are presented in the drop-down entry box.		Objective Achieved	4:58

4.2	"Deploy an appropriate interface for the user to enter the line-up of each team."	User opens user interface, enters season, date and clubs	22 dropdown boxes are presented to the user	 All 22 dropdown boxes are presented and operational	Objective Achieved	5:06

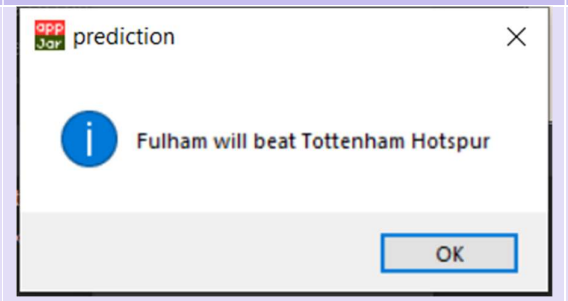
4.2.1	"The user-interface must suggest players in the selected team after the club is chosen."	User clicks on player entry box	A List of players available from that club are presented	 A List of selectable players are presented. Each player is from that club and season.	Objective Achieved	5:10

4.2.2	"The user-interface must have a drop-down menu must allow the user to select the exact player they are looking for."	The user clicks on a suggested player	The player is selected	 Each player can be entered	Objective achieved	5:10
4.2.3	"The user-interface must only allow a line-up of 11 players to be entered."	The user attempts to enter less than 11 players OR duplicated players	An error message is presented, and the user is asked to amend line-up		Objective achieved	5:42
4.2.4	"The user-interface must allow specific players to be omitted or all currently entered players to be removed after being chosen."	The user unselects a player	The player will be omitted	 The player is replaced with a place holder	Objective achieved	5:47
5	"Calculate a club's recent				All Achieved	**6:09**

	form using the data stored in the database using relevant statistics."					
5.1	"Pull values the database which are within the necessary time period and required to calculate form."	The date of the current game	A list of matches is retrieved within the last 30 days	Uses SQLite SELECT query to select match values from the previous 30 days, this is stored as a List.	Objective achieved	6:12
5.2	"Calculate the recent form of the respective club for a specific game-week by accumulating the points achieved in the Premier League across the previous month."	List of previous matches	A real value is calculated	Aggregates recent form by summing the total points and dividing by how many games played in that period.	Objective achieved	6:12

6	**"Calculate relevent metrics from the database (e.g. defensive rating, offensive ratings)"**				**Objective achieved**	**6:20**
6.1	"Pull the line-up of the team playing in the specific match"	Details about the match	List of players who are in the starting line-up	A List of objects called Player which holds all associated data	Objective achieved	6:21
6.2	"Identify if the fixture is being played at their home stadium or at an away stadium"	Information about the match being played.	A flag is created to signify if the match is being played at home or away.	During development it became apparent that there was no need to create a specific metric related to home/away. This was because the home team will always use the same input placeholder into the neural network therefore it is recognised implicitly.	Objective achieved	N/A
6.3	"Produce various metrics which analyse the player ratings held in a line-up at different playing positions in the field-of-play."	Player objects related to each player in a line-up	A series of averages are taken of the ratings of each position of play.	A NumPy array of 4 different averages are created: overall defensive, midfield, forward.	Objective achieved	6:21
7	"Create a model with	N/A	N/A	With TensorFlow, a neural network classifier with 2 hidden layers is made. Suitable	Objective achieved	6:38

	the metrics being mapped to the result using a machine learning technique."			Hyperparameters are chosen. Takes 10 metrics in a NumPy array and returns a discrete match outcome.		
8	"Train the model with metrics and the results in the database and store persistently."	2660 total matches with all associated data	A training set is generated, and the model is trained.	A training set of 1932 matches is made, and the training process is executed. The model is stored persistently with all weights and hyperparameters in a .H5 file.	All achieved	6:51
8.1	"Divide all input and output data into training and testing sets of data."	2660 total match objects	A suitable partition is made, and 2 sets are made with no overlap	70:30 training to testing ratio created. The sets only contain the metrics	Objective achieved	6:51
8.2	"Input the various metrics and form calculated of a	NumPy feature vectors	The NumPy feature vectors are labelled with actual	Each feature vector contains the final attribute relating to the actual outcome, this is used as a label for the training process.	Objective achieved	6:51

	particular game into the model and train against the known historical result."		outcome and this is used to train model.			
9	"Display the calculated prediction as a discrete outcome to the user."	User enters line-up into GUI	The user is informed with the predicted outcome	 The user is presented with a correct an informative message.	Objective achieved	5:58

Feature Testing

Graphical User Interface

As the GUI relies on drop down boxes, erroneous data can only be tested when there isn't a default value.

Test Number	Details	Test Data	Expected Outcome	Type	Result
1.0	User enters a season from the drop-down form.	User selection	The GUI should accept the input and proceed to next input forms.	Normal	Pass
1.1	User enters a valid date using the 3 drop-down boxes.	3 user selections	The GUI should accept the input and proceed to next input forms.	Normal	Pass
1.2.1	User enters 2 valid and distinct teams into the drop-down boxes.	2 user selections	The GUI should accept the input and proceed to next input forms.	Normal	Pass
1.2.2	User enters the same team into both drop-down boxes.	2 (duplicated) user selections	The GUI should recognise the invalid entry and inform the user to re-enter the teams	Erroneous	Pass
1.2.3	User does not enter a team into one or both of the input boxes	1 or 0 user selections	The GUI should recognise that a team hasn't been entered and inform the user to enter a team	Erroneous	Pass
1.3.1	User enters all players correctly	22 user selections	The GUI should accept all 22 player entries and give a prediction	Normal	Pass

1.3.2	User fails to enter 22 players	<22 user selections	The GUI should recognise a player entry form was not filled then the user must be informed to re-enter	Erroneous	Pass
1.3.3	User selects players with duplicated	22 user selections	The GUI should recognise that there is a duplicated entry and inform the user to re-enter.	Erroneous	Pass
1.3.4	The User enters multiple unlisted players.	22 (with multiple 'unlisted') user selections	The GUI should accept the entered players despite the duplicated 'unlisted' selection.	Normal	Pass

sofifa.com Scraper

Test Number	Details	Test Data	Expected Outcome	Type	Result
2.0.1	Scraper for 'sofifa.com' is given a URL	1 valid URL	The requests library opens a connection with API status code 200.	Normal	Pass
2.0.2	~	1 invalid URL of a different or non-existent page.	Exception caught, program exits.	Erroneous	Pass
2.1.1	Connection attempted with different Wi-Fi connectivity	Strong Wi-Fi	Program continues with status code API 200.	Normal	Pass

2.1.2	~	Intermittent Wi-Fi through runtime	If Wi-Fi connection drops during runtime, program should wait and attempt to reconnect until timeout.	Boundary	Pass
2.1.3	~	No Wi-Fi connection	Exception caught before player extraction is attempted.	Erroneous	Pass
2.2.1	Different formats of the initial season.	Season argument in correct format : (digits/digits)	PlayerScraper class is instantiated without exception.	Normal	Pass
2.2.2	~	Season in incorrect format: “XXXX/XXXX”	Exception is caught during initial instantiation of PlayerScraper.	Erroneous	Pass

soccerway.com Scraper

Test Number	Details	Test Data	Expected Outcome	Type	Result
3.0.1	Connection attempted with different Wi-Fi connectivity	Strong Wi-Fi	Webdriver opens and connects to domain	Normal	Pass
3.0.2	~	Intermittent Wi-Fi through runtime	If Wi-Fi connection drops during runtime, program should wait and attempt to reconnect until timeout.	Boundary	Pass
3.0.3	~	No Wi-Fi connection	Exception caught before player extraction is attempted. Webdriver does not open	Erroneous	Pass
3.1.1	The root URL of the season page is entered	1 correct URL	Webdriver opens and program proceeds	Normal	Pass
3.1.2	~	1 invalid URL	Webdriver Exception is caught, and program exits	Erroneous	Pass
3.1.3	~	1 URL of a page which exists but isn't soccerway.com	Webdriver Exception, program exists	Boundary	Pass
3.2	Path for the Google Chrome Webdriver is not found	1 invalid path to WebDriver	Webdriver Path exception	Erroneous	Pass

Database

Test Number	Details	Test Data	Expected Outcome	Type	Result
4.0.1	CSV file names are entered	Correctly formatted and valid CSV name	The program reads the CSV files and proceeds	Normal	Pass
4.0.2	~	Non-existent CSV files	The program should instantly catch the exception and exit.	Erroneous	Pass
4.0.3	~	Existent CSV, bad formatting	The program should raise exception due to an unexpected number of values	Boundary	Pass
4.1.1	Database address entered	Non-existent database	Exception caught and program exits	Erroneous	Pass
4.2.1	Name of Player can't be matched	Erroneous player name who has different name format due to different data sources	n-gram similarity is applied to find most similar player from the club	Erroneous	Pass
4.2.2	Name of Club can't be matched	Erroneous club name which has different name format due to different data sources	Levenshtein edit distance is used to find most similar club	Erroneous	Pass
4.3.1	Database already created	Path of a constructed database is entered	Only tables which don't exist will be generated	Erroneous	Pass

Player Instantiation

Test Number	Details	Test Data	Expected Outcome	Type	Result

5.0.1	Recent Form calculations	Valid match date in middle of season	Points-per-game is correctly calculated	Normal	Pass
5.0.2	~	Valid match date on the first day of a season (no matches over last 30 days)	Points-per-game set to zero.	Boundary	Pass
5.1.1	Rating metric calculations	A line-up with all valid ratings across the field-of-play	Average rating of each group is correctly calculated	Normal	Pass
5.1.2	~	Line-up does not contain players in a part of the field of play (e.g. no goalkeeper)	The average rating of that group is set to zero	Boundary	Pass
5.2.1	Match-club-player composition	Separate related objects	Match object composes of 2 club objects (composed of 11 player objects respectively)	Normal	Pass

Model

Test Number	Details	Test Data	Expected Outcome	Type	Result
6.0.1	Creating Model	Valid Hyperparameters	The schema of the model is created	Normal	Pass
6.1.1	Training model	Valid training/testing set	Model is trained and can be saved	Normal	Pass

6.1.2	~	Training set has invalid feature vector shape	Exception caught and program exits	Erroneous	Pass
6.1.3	~	Given as Python List rather than NumPy array	Exception caught and program exits	Erroneous	Pass
6.2.1	Saving a model	Valid .hdf5 name entered	Checkpoint of model is saved	Normal	Pass
6.3.1	Loading a model	Valid .hdf5 name entered	Checkpoint of model is loaded and ready for predictions	Normal	Pass
6.3.2	~	File doesn't exist	Exception is caught and program exits	Erroneous	Pass

"To document the analysis of the performance of the solution" [video; 7:13]

1. Determine the most appropriate machine learning approach (e.g. Classification or Regression)

The analysis phase, regression and classification approaches were evaluated to understand if the model would fit the raw data and be appropriate for predictions. After concluding that classification would be suitable for outcome predictions, this informed the design phase.

Objective achieved ✔

2. Determine the specific machine learning algorithm

The analysis phase also specified the specific algorithm used. The paper by Corentin Herbinet reviews a range of algorithms with football data. It appeared that the use of artificial neural networks is most appropriate. The underlying mathematics was reviewed in the design phase before using TensorFlow in the implementation.

Objective achieved ✔

3. Identify the most appropriate combinations of input data to use

The objectives set in the analysis section state the use of player data from the EA Sports FIFA franchise and match data from Premier League matches. In the design section a set of proposed quantitative metrics to be used as an input into the neural network.

Objective achieved ✔

4. Recognise the pre-processing required for the specific model

The proposed metrics were already in a correct format, meaning each rating metric uses the same relative numerical system. Although, the rating metric was altered to find the points per game rather than total. A NumPy array was chosen as a 1-dimensional feature vector format.

Objective achieved ✔

5. Test the accuracy of model using the testing partition of the original training data.

In the benchmarking phase the performance of the model was evaluated. Test statistics used were: Loss, prediction accuracy, validation accuracy and pay-out.

Objective achieved ✔

Evaluation

This section aims to evaluate the success of the technical solution. The objectives set during analysis will be used a criterion. The end-user – Ryan – will then give feedback.

My personal evaluation of the project is that the technical solution surpasses my initial expectation. The accuracy of the model is exceptional, this was surprising regarding only 10 basic statistics were used. The solution also appears to be robust and efficient despite large amounts of data processing. The user-interface is intuitive and allows a prediction to be received very quickly. This should satisfy Ryan's original demands. Although, through developing the program I identified limitations and potential improvements.

Objectives

The testing section evaluates the extent each objective was achieved in the technical solution. The technical solution achieves all objectives set and this report covers the objectives set regarding documenting the performance of the model.

Evaluation by objective group:

Objectives 1

All objectives achieved. The solution has a recursive solution for web scraping FIFA player data from 'sofifa.com'. Use of Requests library for retrieving page data before being parsed by BeautifulSoup.

Objectives 2

All objectives achieved. Due to interactive page elements, a webdriver was used which opens a browser and simulates a user clicking on page elements. This enabled the data from the results table to be iteratively extracted. Each result page was retrieved with Requests and parsed with BeautifulSoup.

Objectives 3

All objectives achieved. An SQLite database was implemented with 5 separate tables. CSV data was distributed appropriately. Also, string similarity algorithms: n-gram similarity and Levenshtein was used to match player and club entries from both data sources ('sofifa.com' and 'soccerway.com'). Also, various SELECT queries were implemented including those in the 'example queries' section.

Objectives 4

All objectives achieved. A GUI with an MVC architecture was produced and passes all testing. AppJar was used to create an interactive view for the user. The controller acts as the intermediary between the model (database and Neural Network Model) and the front-end.

Objectives 5

All objectives achieved. Using a query to get all matches over the last month, the points-per-game was calculated.

Objectives 6

All objectives achieved. Players in the line-up were split up into different groups based on field of play. The mean of each group was taken.

Objectives 7

Objective achieved. Using TensorFlow a neural network model was created with the correct hyperparameters for the data.

Objectives 8

All objectives achieved. A training and testing set were created, and the model was trained through propagation and backpropagation.

Objectives 9

Objective achieved. The most probable result is passed through the MVC architecture and presented to the user.

Completeness of Solution

Objectives alone will not outline whether the program is a good solution. Prediction is based upon optimisation hence the benchmarking section. The model achieved comparable accuracy to other recent research papers including proposed solutions.

Additionally, the program is flexible to add more up to date data as long as the webpages of data sources continue to permit it and don't alter their HTML structure too drastically.

Finally, the user interface allows the user to receive predictions almost instantly after entering data thanks to storing a pre-trained model.

User Feedback

Verbal feedback from Ryan:

"I have been using this program for a few weeks to aid my fantasy football selection. The features of the project meet all of my original demands. I can receive outcome predictions very quickly allowing me to get predictions for each match in a given game week within minutes. I find the user interface very simple with the use of drop-down boxes. The program has never failed to give me a prediction and does not require connection to the internet.

I have found the predictions very useful and they are correct more often than not. I have been informed that it uses recent form and finds the strength of a line-up; both of which I demanded earlier. Also, its quoted at between 55-60% accuracy which far exceeds my original demands of it being better than random (33%) - this was exceptional.

The only improvement I would suggest is that the model very rarely predicts a draw as an outcome."

Response to feedback

It is clear that Ryan liked the final version of the program. The goals and objectives set were appropriate for meeting his demands. He did mention that very few draws were predicted, this was also observed in testing where zero draws were predicted (see APPENDIX A, full listing of Premier League predictions).

Improvements

If I was to create another similar football prediction program, I would make the following changes:

1. Accurate predictions before 1-hour prior
2. Include more parameters for the model
3. Better data sources
4. Cloud-based version
5. Social media sentiment analysis

Accurate predictions before 1-hour prior

Currently, the model is set-up so the user can make a prediction once they have access to the match line-ups online (e.g. BBC Sport releases line-ups 1 hour before match). An improvement would be where the model could use a predicted line-up based on previous matches to enable accurate predictions to be made far before kick-off. This would be a simple operation as the previous line-ups are already stored in the database.

Include more parameters for the model

An obvious improvement would be to increase the range of inputs into the prediction model. This could be to include a weather metric as some teams play better than others in poor conditions.

Better data sources

Currently 'sofifa.com' is being used as a data source. The FIFA ratings they present are only updated at the beginning of each season. A more appropriate resource would update the rating of player based on their recent performances. The problem arises where teams sign players they fail to perform to their potential rating.

Cloud-based version

Instead of a local version of the program, which Ryan wanted. A more flexible approach would be to host the prediction software online allowing various internet connected devices to make predictions. Using REST APIs, the contents of the database could be changed centrally with CRUD (create-read-update-delete) methods. The architecture and platform of the whole system is dramatically altered meaning this would take lots of development.

Social media sentiment analysis

Another example of a different parameter would be to measure the polarity (positivity) of text posted on social media from fans. This has had varying levels of success in football prediction but is an established way of predicting election results.
Currently, the prediction model often favours the stronger team because they have higher player ratings. By combining a new metric based on social media polarity perhaps this could encourage the model to predict upsets better.

Bibliography

[1] A. Reed, “Half the world’s population tuned in to this year’s soccer World Cup,” CNBC, 21st December 2018. [Online]. Available: https://www.cnbc.com/2018/12/21/world-cup-2018-half-the-worlds-population-tuned-in-to-this-years-soccer-tournament.html. [Accessed 17th May 2019].

[2] OptaSports, “We cover the world of football in greater depth and detail than anyone else,” Opta, [Online]. Available: https://www.optasports.com/sports/football/. [Accessed 19th May 2019].

[3] C. Hayashi, What is Data Science? Fundamental Concepts and a Heuristic Example, Springer Japan, 1998.

[4] M. G. M. G. Andrea De Mauro, “What is Big Data? A Consensual Definition and a Review,” *Research Gate,* p. 8, 2015.

[5] My Football Facts, “Premier League Goal Records & Statistics Season 1992-93 to 2017-18,” My Football Facts, 2018. [Online]. Available: http://www.myfootballfacts.com/Premier_League_Goal_Statistics_1992-93_to_2009-10.html. [Accessed 17th May 2019].

[6] M. Cohn, User Stories Applied, Lafayette, Colorado: Addison-Wesley Professional, 2004.

[7] R. N. DANIEL PETTERSSON, “Football Match Prediction using Deep Learning,” Department of Electrical Engineering, Chalmers University of Technology , Gothenburg, 2017.

[8] N. E. F. M. N. Anthony Costa Constantinou, “Profiting from an inefficient association football gambling market,” *Knowledge-based Systems,* vol. 50, pp. 60-86, 2013.

[9] J. K. Johannes Stubinger, “Beat the Bookmaker – Winning Football Bets with Machine Learning,” *LNAI,* vol. 11311, pp. 219-233, 2018.

[10] H. A. Lars Magnus Hvattum, “Using ELO ratings for match result prediction in association football,” *International Journal of Forecasting,* vol. 26, pp. 460-470, 2010.

[11] C. Herbinet, “Predicting Football Results Using,” Imperial College London, London, 2018.

[12] B. Grantham, “Predicting Football Matches using EA Player Ratings and Tensorflow,” Medium : Towards Data-Science, 22nd July 2018. [Online]. Available: https://towardsdatascience.com/predicting-premier-league-odds-from-ea-player-bfdb52597392. [Accessed 2nd June 2019].

[13] football-data, “Football Results, Statistics & Soccer Betting Odds Data,” 20 October 2019. [Online]. Available: European Football Results and Betting Odds - Football-Data. [Accessed 22 October 2019].

Appendix A

Full Premier League Pay-out table (with odds from betting companies)

HomeTeam	AwayTeam	FTResult	Prediction	correct?	PROBABILITY	B365H	B365D	B365A	BWH	BWD	BWA	MAX HOME	MAX DRAW	MAX AWAY	PAY-OUT
Man United	Leicester	H	H	Y	0.63055086	1.57	3.9	7.5	1.53	4	7.5	1.58	4	7.5	£ 58.00
Bournemouth	Cardiff	H	H	Y	0.6293024	1.9	3.6	4.5	1.9	3.4	4.4	1.91	3.63	4.75	£ 91.00
Fulham	Crystal Palace	A	H	N	0.4264135	2.5	3.4	3	2.45	3.3	2.95	2.5	3.46	3	-£ 100.00
Huddersfield	Chelsea	A	A	Y	0.50311404	6.5	4	1.61	6.25	3.9	1.57	6.5	4.02	1.62	£ 62.00
Newcastle	Tottenham	A	A	Y	0.44062313	3.9	3.5	2.04	3.8	3.5	2	3.9	3.57	2.1	£ 110.00
Watford	Brighton	H	H	Y	0.5306842	2.37	3.2	3.4	2.35	3.1	3.3	2.43	3.3	3.4	£ 143.00
Wolves	Everton	D	H	N	0.47289756	2.37	3.3	3.3	2.35	3.2	3.2	2.38	3.4	3.3	-£ 100.00
Arsenal	Man City	A	A	Y	0.3879641	4	3.8	1.95	3.7	3.75	1.95	4	4	2	-£ 100.00
Liverpool	West Ham	H	H	Y	0.6903751	1.25	6.5	14	1.2	6.75	14	1.27	6.75	14	£ 27.00
Southampton	Burnley	D	H	N	0.4683811	1.85	3.5	5	1.8	3.5	4.75	1.86	3.6	5.2	-£ 100.00
Brighton	West Ham	H	H	Y	0.4884786	2.7	3.3	2.87	2.6	3.25	2.8	2.72	3.3	2.88	£ 172.00
Burnley	Huddersfield	D	H	N	0.6185444	2.14	3.1	4.2	2.05	3.1	4.1	2.16	3.1	4.33	-£ 100.00
Crystal Palace	Wolves	A	H	N	0.43971637	3	3.2	2.62	2.8	3.2	2.65	3	3.25	2.65	-£ 100.00
Leicester	Everton	A	H	N	0.48700723	2.2	3.4	3.6	2.2	3.4	3.3	2.2	3.52	3.6	-£ 100.00
Man United	Newcastle	H	H	Y	0.70232815	1.44	4.5	9	1.44	4.5	7.5	1.5	4.54	9	£ 50.00
Tottenham	Cardiff	H	H	Y	0.77323693	1.22	7	15	1.22	6.5	12.5	1.25	7	15.07	£ 25.00
Watford	Bournemouth	A	H	N	0.45737377	2.14	3.6	3.5	2.1	3.6	3.4	2.15	3.6	3.58	-£ 100.00
Fulham	Arsenal	A	A	Y	0.45725784	5.25	4.33	1.66	5	4.33	1.62	5.25	4.4	1.7	£ 70.00
Liverpool	Man City	D	H	N	0.4426582	2.54	3.6	2.8	2.5	3.6	2.7	2.56	3.6	2.85	-£ 100.00
Southampton	Chelsea	A	A	Y	0.4719868	6	4.1	1.61	6	3.9	1.6	6.02	4.2	1.65	£ 65.00
Cardiff	Wolves	H	H	Y	0.4100665	3.8	3.25	2.2	3.5	3.2	2.2	3.8	3.32	2.25	£ 280.00
Crystal Palace	Burnley	H	H	Y	0.5553737	1.61	3.9	6.5	1.62	3.75	6	1.67	3.92	6.65	£ 67.00
Huddersfield	Brighton	A	H	N	0.49260518	2.5	3	3.4	2.35	3	3.4	2.55	3.06	3.42	-£ 100.00
Leicester	Watford	H	H	Y	0.59725535	2.25	3.3	3.6	2.2	3.3	3.4	2.3	3.35	3.6	£ 130.00
Man City	Bournemouth	H	H	Y	0.76205677	1.11	11	26	1.1	10	23	1.13	11	26	£ 13.00

Newcastle	West Ham	A	H	N	0.48682997	2.5	3.3	3.1	2.45	3.25	3	2.5	3.38	3.13	-£	100.00
Southampton	Man United	D	A	N	0.56453323	4.33	3.6	1.95	4.2	3.6	1.87	4.33	3.6	1.99	-£	100.00
Arsenal	Tottenham	H	H	Y	0.43979022	2.62	3.6	2.75	2.5	3.6	2.7	2.65	3.73	2.75	£	165.00
Chelsea	Fulham	H	H	Y	0.7736188	1.18	8	17	1.16	8	16	1.2	8	17	£	20.00
Liverpool	Everton	H	H	Y	0.65387815	1.4	5	9	1.36	5	8.5	1.41	5.05	9	£	41.00
Wolves	Liverpool	A	A	Y	0.41546997	7	4	1.57	6.5	3.9	1.57	7	4.1	1.6	£	60.00
Arsenal	Burnley	H	H	Y	0.69578624	1.25	6.5	15	1.22	6.25	13.5	1.25	6.5	15	£	25.00
Bournemouth	Brighton	H	H	Y	0.48652515	1.9	3.7	4.33	1.85	3.6	4.33	1.95	3.7	4.4	£	95.00
Cardiff	Man United	A	A	Y	0.48815218	6.5	4.2	1.57	5.75	4	1.6	6.5	4.2	1.62	£	62.00
Chelsea	Leicester	A	H	N	0.6992126	1.33	5.5	11	1.33	5	10	1.35	5.5	11.37	-£	100.00
Huddersfield	Southampton	A	A	Y	0.4515654	3.1	3.1	2.6	3.1	3.1	2.5	3.1	3.15	2.68	£	168.00
Man City	Crystal Palace	A	H	N	0.7238591	1.14	10	21	1.15	8	17.5	1.15	10	23	-£	100.00
Newcastle	Fulham	D	H	N	0.58941126	2	3.6	4	1.95	3.5	3.9	2.03	3.6	4.1	-£	100.00
West Ham	Watford	A	H	N	0.55893064	2.37	3.5	3.1	2.25	3.5	3.1	2.38	3.57	3.16	-£	100.00
Everton	Tottenham	A	A	Y	0.39062122	3.25	3.5	2.3	3.2	3.4	2.25	3.27	3.5	2.35	£	135.00
Brighton	Burnley	A	H	N	0.4824237	2.05	3.3	4.2	2	3.3	4.1	2.06	3.41	4.33	-£	100.00
Crystal Palace	West Ham	D	H	N	0.4936752	2.1	3.5	3.75	2.05	3.4	3.75	2.15	3.5	3.8	-£	100.00
Fulham	Man United	A	A	Y	0.45889425	5.5	4.4	1.61	5.5	4	1.62	5.57	4.4	1.65	£	65.00
Huddersfield	Arsenal	A	A	Y	0.43178853	5.75	4.4	1.6	5.5	4.1	1.6	6	4.4	1.63	£	63.00
Liverpool	Bournemouth	H	H	Y	0.7531354	1.22	7.5	14	1.19	7.25	13.5	1.22	7.5	15.64	£	22.00
Southampton	Cardiff	A	H	N	0.5959641	1.75	3.9	5	1.75	3.7	4.75	1.8	3.9	5.2	-£	100.00
Watford	Everton	H	H	Y	0.4318293	2.35	3.5	3.2	2.25	3.4	3.2	2.41	3.5	3.24	£	141.00
Man City	Chelsea	H	H	Y	0.587394	1.5	4.75	7	1.48	4.5	6.5	1.55	4.75	7	£	55.00
Tottenham	Leicester	H	H	Y	0.6301321	1.7	3.9	5.5	1.65	3.9	5.25	1.72	3.9	5.7	£	72.00
Wolves	Newcastle	D	H	N	0.61896336	1.7	3.75	5.75	1.7	3.6	5.5	1.7	3.82	6	-£	100.00
Brighton	Man City	A	A	Y	0.536844	19	8.5	1.16	17.5	8.75	1.14	21	8.75	1.18	£	18.00
Burnley	Arsenal	A	H	N	0.39414206	3.25	3.8	2.2	3.1	3.75	2.15	3.25	3.8	2.25	-£	100.00
Crystal Palace	Bournemouth	H	H	Y	0.623286	1.9	4.2	3.8	1.85	3.9	3.9	1.9	4.2	4.1	£	90.00
Fulham	Newcastle	A	H	N	0.42681295	2.5	3.6	2.9	2.55	3.6	2.65	2.55	3.6	2.9	-£	100.00
Leicester	Chelsea	D	H	N	0.42856705	2.4	3.75	2.9	2.35	3.6	2.85	2.44	3.75	2.93	-£	100.00

Liverpool	Wolves	H	H	Y	0.6992229	1.3	6	11	1.3	5.75	9.5	1.31	6	11.5	£	31.00
Man United	Cardiff	A	H	N	0.7256925	1.28	6.5	11	1.25	6.25	11	1.29	6.5	11	-£	100.00
Southampton	Huddersfield	D	H	N	0.61093646	1.44	4.75	8.5	1.42	4.75	7.25	1.44	4.83	8.5	-£	100.00
Tottenham	Everton	D	H	N	0.56744236	2.2	3.5	3.5	2.1	3.5	3.5	2.2	3.64	3.7	-£	100.00
Watford	West Ham	A	H	N	0.4798237	2.25	3.75	3.2	2.2	3.7	3.1	2.25	3.85	3.25	-£	100.00

GAMEWEEK	*PAYOUT*	*ROI % (RETURN-ON-INVESTMENT)*
1	£ 91.00	9.10%
2	-£ 118.00	-11.80%
3	£ 416.00	41.60%
4	£ 145.00	14.50%
5	£ 218.00	21.80%
6	-£ 561.00	-56.10%
OVERALL	£ 191.00	3.18%

Standard deviation pay-out through 6 game weeks:	£ 308.882619