Time Series Analysis to Predict Player Points in Fantasy Premier League

By: Joakim Martin Torsvik
Candidate Number: 244848

MSc in Data Science

Date of submission deadline: 25th of August 2022

Abstract

The challenge of predicting the performances of athletes before they perform has become very appealing to fans and spectators, and as such Fantasy sports has become a huge past-time hobby for fans with this desire. The biggest sport in the world, football (soccer), is naturally also the biggest in Fantasy sports (1) with Fantasy Premier League being the biggest platform today, having more than 9.1 million people (2) around the world play the game in the 2021-2022 season. The aim of this research is to find a suitable machine learning algorithm that is capable of performing a time series analysis on each player in the Premier League in order to predict their performances in the next gameweek(s). The machine learning models in question will be a univariate ARIMA-model, an Ensemble Gradient Boosting-model (XGBoost), a Support Vector Machine-model (SVM) and finally a Deep Learning, Recurrent Neural Network that will be state-of-the-art for predicting time series analyses. These models will be well suited for finding the degree of complexity that is required to make a valid prediction.

Table of Content

Fantasy Football Terminology

Manager - A player of Fantasy Football

Player - The real-life players that the game is based on

Gameweek - Every team will play 38 games in the season. A gameweek represents a week where all the teams are scheduled to play. The scheduling is set before the season starts and alterations can be made so that a given team does not play in one gameweek but plays two games in the next gameweek.

Transfers - A transfer in Fantasy Football terms refers to selling one player and replacing them with another.

Points - Points are obtained by the players in the team. The goal of Fantasy Football is to get as many points as possible. This will be the dependent variable (predicted output) for this project.

Assist – In Fantasy Premier League, an assist is considered the player who makes the final pass to the goalscorer. The assist must come from a player in the same team as the goalscorer and the opposing team players cannot have touched the ball in a manner that significantly alters the route of the ball. Minor touches that do not affect the intended destination of the ball will still count as an assist.

Own Goal – If a player shoots or passes the ball into the goal of their own team, it is described as an own goal. Own goals also include touching the ball and significantly altering the route of the ball, accidentally, into their own goal.

FDR – Fixture Difficulty Rating. A classification algorithm that depicts the difficulty of the team and their opponents.

1.0.0 Introduction

In the 21st century, Fantasy Sports has risen to become all sports fans favourite hobbies when games are not being played. The game is being played in all different types of sports, has participances of more than 500 million people on a yearly basis (3) and despite being a free game, it grossed over 21.39 billion USD in 2021 alone.

Football (soccer) is the most popular out of all Fantasy Sports platforms. Each football league comes with their own Fantasy platform, with the most popular coming from the English Premier League with more than 9 million players in 2021. Many players try every week and every year, to predict which team will win and which players will score a goal or two. The game of FPL is about finding the players that will perform the best in the next weeks game and throughout the season. As many different factors affect the players performances, it is very difficult to filter out the noise and make accurate predictions to which player one should pick. The idea for the dissertation is to use Machine Learning to try and predict which players that will perform in the future, and if successful, become a tool for Fantasy managers to apply when selecting players.

1.1.0 Aim

The dissertation aims to analyse if it is possible to predict a player’s performance for an upcoming game using historic information about previous matches. To do so, the challenge will be to find a model that will fit to the complexity of the dataset and be able to predict an accurate output. The models in question will range from simple statistical models to a state-of-the-art deep learning neural network-model. The performance of a player is measure from the virtual, open-play game, Fantasy Premier League where the goal is to select different football players that the fantasy player believes will play well in the next games and thus return high points in the game.

1.2.0 Limitations

The dataset is an open-source retrieval of a player’s performance in Fantasy Premier League and contains data where a player is able to gain points or loose points. The dataset of choice contains only data on successful happenings from the player in question and tells little on their attempts, their positioning on the pitch, or any other data that could be useful to have. This kind of data is usually proprietary and can be bought from Sports Analytics companies such as Opta Sports and Statsbomb.

The data is also limited to the English Premier League, so when a team buys a player from another national league or when a team is promoted from the EFL Championship, their players and team will not have any data available for testing. In some cases, this will limit the datasets quite a bit.

1.3.0 Literature Review

The articles (4)- (5) is a two-part article about how to improve your game with use of Machine Learning. They recognized, as stated in the introduction, that the most difficult part of playing FPL is picking players and so they created a model to select a team that they could start their 2020-21 season with, and a simple linear regression as their model. The first part mostly focused on obtaining and analysing the data and the second part of the article focused on how the model predicted team players to perform. The results weren't fantastic as some of the suggested player choices were questionable, e.g. selecting Nemanja Matic as a midfield option and the author did recognize this in their conclusion stating that it was a fairly simplistic approach to solving the problem, but expressed that this would be a good starting-point for others to carry on the research with more complex solutions. Another article made by Dhaneswara Mandrasa (6) tried to make a forecast-model to predict player performances similar to the forementioned author, but this time with a Random Forest and Artificial Neural Network [ANN] as predictor models. Dhaneswara made a very thorough exploratory data analysis and found that players like Harry Kane, which tends to be one of the favourites by FPL participants, tends not to deliver as many points as his value might suggest. Both of the articles did fairly basic forms of model predictions and used algorithms that doesn't account for local trends especially well.

FPL is definitely a game that favours strategic thinking. The grandmaster of chess, Magnus Carlson, plays FPL regularly and at a point in the 2019-20 season, he was even at the top of the global leader board (7). At the end of the season Carlson didn't win in the overall rankings. The winner was instead an Oxford University mathematician, Joshua Bull, who later published a presentation (8) he made of how strategic thinking can help to improve one's performance in FPL. He proposed multiple strategies such as picking players based on whether it was a home-or away fixture, because of past performances or due to the simplicities of their upcoming fixtures. He went on to explain that he had built a model that simulated a lot of random teams from a pool of the most popular players in the game where each team had a one of the proposed strategies that he wanted to analyse. He then simulated the teams playing an entire season of FPL where the captain of each team was picked based on the individual selection strategy assigned to the teams and repeated the simulation 10,000 times. From the simulations he concluded that players that have a positive trend over the past weeks should be preferred over players that have "easy" upcoming gameweeks. He did, however, specify that following this advice, one should take heed of The Gambler's fallacy, which states that an event that is proven independent of past occurrences shouldn't be less or more likely to occur in the future when the event has happened. In FPL terms, this mean that a player that could be translated to, just because a player scored in the previous game, doesn't mean they are more or less likely to score in the next game. Since the presentation wasn't about the model, but rather the results, Bull didn't focus a lot on explaining in detail how the model worked, but in a paper by Jim Joseph et al. (9) they built a Monte Carlo simulator and made a player selection model based of a simple ANN. The author acknowledged that their recruiter model was a "lazy" and generalized solution to player selection, but the goal of this paper was to find which strategy was the best to take when recruiting players, rather than building an optimized recruiter model. The Monte Carlo simulator predicted the outcomes of what would happen when they substituted a player from the current team with one from the predicted outcomes of the ANN. To swap out a player, they calculated the "return on investment" of each player in the team and transferred out the weakest player, then the recruiter ANN selected the best player to substitute in, and the transfer is then simulated for 10,000 times. They found by looking at the simulation that the most favourable players to transfer in was player that tended to be very influential in the games. FPL receives statistics from Opta which tells how involved players are in the game, their creativity and goal-threat (ICT index) (10). This is an interesting observation seeing as ICT isn’t one of the most talked about statistics most participants use when selecting players and so, this could be good to take note of for the research.

From the forementioned works the authors have mostly looked to predict player performances based of machine learning models without looking at them like time-series problems. However, since FPL is a game played over a span of 10 months it could be more useful to look at the problem as a time-series issue. Delano Ramdas (11) did this in his dissertation when he looked to answer how Convolution Neural Networks fared as a Deep Learning time-series forecaster by implementing it on FPL data and comparing it to a Recurrent Neural Network [RNN LSTM] and a regular ANN that didn’t sequence the data as time-series. He focused on looking at the individual performances of each player without including what team they played for and the opponents they faced and built his simulator to predict the next game week result based on the previous performances of the players in the last 3, 6 and 9 game weeks and found that the CNN did indeed regularly outperformed the ANN and the LSTM RNN model with lower residual error with the Mean Squared Error [MSE], and stated that the model successfully predicted player performances with a low error. However, it would be interesting to see whether the model actually could build a team of high performing players and compare the results of this team against the FPL overall rankings.

Two Swedish students, Adrian Lindberg and David Söderberg wrote a similar thesis (12), where their goal was to answer whether time-series based machine learning models would be better at predicting player performances than non-time-series models. They compared a similar RNN LSTM-model to that of Delano Ramdas (13), with a Support Vector Machine [SVM] and an ANN. During their research, they had an interesting idea making a regression model that predicts the expected maximum for every player, this has been standard with most problems as of now, they would also make a classifier that was going to generalize the problem statement a lot more and finding multiple players who are expected to do well, but not specify which player will do the best and the worst. The reasoning for this was that since the uncertainties of the game are so great, picking a singular best performing player could lead to wrong predictions more times than not. Their models would then generate a team of players to select for a team and compare the results from the models with that of the average game week scores by the world-wide participants. They found that both the classifier and the regression methods for the chosen models were effective to selecting players for upcoming game weeks, with the regression-method giving the best returns for every model. The model with the highest returns and the lowest losses was the time-series LSTM-model. It was capable of making good decisions that for most game weeks returned a higher score than the average, but still showed limitations as it made some questionable decisions at times as they exemplified with its decision to pick Chelsea defender Antonio Rüdiger who at the time didn’t receive a lot of regular playing-time.

1.4.0 Rules of Fantasy Premier League

The rules of Fantasy Premier League are listed on their website (10)

1.4.1 Team Structure

The rules of different Fantasy leagues tend to be slightly different depending on wat type of league-system that is in place. In Fantasy Premier League, the fantasy-player (known as the "manager" in Fantasy-terms) initially buys 15 players in each respective position. The rules state that the team must consist of 2 goalkeepers, 5 defenders, 5 midfielders and 3 attackers, and in order to prevent every manager overloading on players from the best teams there is a cap of maximum 3 players from the same team.

After purchasing the 15 players for the team, the manager must decide on 11 players to play the next gameweek and 4 players to put on the bench. Only one of the two goalkeepers can be played, and the team must play a minimum of three defenders in the same gameweek. Only the players that are playing will have their points recorded for the team, but if a player that was set up to play, doesn't play in real life, one of the substitutes can come in as long as it doesn't break the rules of how the team is supposed to be structured. It is possible to prioritize which player on the bench will come on and play by sorting which substitute will be first priority, second and third. The substitute goalkeeper can only switch with the playing goalkeeper, so they will not count in the prioritization.

Finally, the manager must assign a captain and a vice-captain. A player that becomes captain will double their points for the gameweek(s) they hold the captaincy, even if the player receives negative points. The vice-captain will become captain if the forementioned captain doesn't play, otherwise vice-captains receive no benefits.

1.4.2 Budget

Managers are given a budget of £100 million to buy players to fill in the team and each player's price is determined before every season, so a manager can't simply buy each of the most recognized "best player" in every position. It is also worth noting that the price of the players rises and falls depending on their performances and the popularity of owning said player. However, in order to remove the financial benefits of a massively increased budget for teams with multiple players with increased values, the game has set a selling fee of 50% of the profit, rounded down to the nearest decimal point (0.1). This rule only applies to profits and not losses, so managers are not protected by any decrease in a player's value from their initial purchasing price.

E.g.:
If a player was purchased for £7.5M and they increased in price to £7.6M, the selling price of the player would still be £7.5M. If the price further increased to £7.7M, the selling price would

be £7.6M. If their value decreased to £7.4M, the selling price for the player would still be £7.4M

1.4.3 Transfers

Before the start of the season, the managers are given unlimited free transfers of players within the budget of £100 million. When the season starts, after every gameweek, the managers are given the option to transfer an existing player for a different player in the same position as long as they are within the budget of the team. If the manager does not make a transfer, the free transfer will be stored as one extra free transfer for future gameweeks. The number of free transfers to own are capped at 2 free transfers, meaning that if a manager chooses not to make a transfer for two or more weeks, only one of the free transfers are stored for future gameweeks. It is possible to make transfers outside initial 1 or 2 free transfers, but that comes at a cost of 4 points per player transfer outside of the free transfers.

Buying one player does not prevent other managers from buying the same player. Each manager is able to buy any player in the game and benefit from their performances as long as they are a part of a Premier League team. This could be viewed upon as something similar to that of a financial stock market, where companies issue millions, billions or maybe even trillions of publicly tradable stocks and apart from the biggest banks, hedge funds and sovereign wealth funds, have a seemingly endless supply of stocks.

1.4.4 Point system

During the season, each player will be allocated points based of their actual performances in the games in the Premier League. Table 1 shows how the points are allocated to each player per game:

Action	**Points**
Playing more than 60 minutes (excluding stoppage time)	2 points
Goal scored by a goalkeeper or defender	6 points
Goal scored by a midfielder	5 points
Goal scored by a forward	4 points
Assisting a goal	3 points
Clean Sheet by a goalkeeper or defender	4 points
Clean Sheet by a midfielder	1 point
For saving every three shots by a goalkeeper	1 point
Penalty saves by a goalkeeper	5 points
Penalty missed or saved by opposition	-2 points
Bonus points for the best players in a match*	1-3 points
For every 2 goals conceded by a goalkeeper or defender	-1 point

Yellow Card	-1 point
Red Card	-3 points
Scoring an Own Goal	-2 points

Table 1: In-game actions that rewards points

Clean Sheets are awarded for not conceding a goal whilst on the pitch and playing at least 60 minutes (excluding stoppage time). If a player is substituted out of the game in real-life, any goals conceded after that will not affect the players clean sheet points, except if the player has to leave the game due to a red card.

The three best players in the game will be awarded bonus points for their performances. The best player receives 3 bonus points, the second best receives 2 points, and the third best receives 1 point. The best players on the pitch are calculated by a Bonus Point System (BPS), where points are distributed based on statistics from the game and the best players are the ones with the highest BPS-score.

Action	**BPS**
Playing 1 to 60 minutes	3
Playing over 60 minutes	6
Goalkeepers and defenders scoring a goal	12
Midfielders scoring a goal	18
Forwards scoring a goal	24
Assists	9
Goalkeepers and defenders keeping a clean sheet	12
Goalkeeper saving a penalty	15
Goalkeeper saving a shot on goal	2
Successful cross from open play	1
Creating a big chance to score	3
Every 2nd block, clearance and/or interception	1
For every 3 ball recoveries	1
Key pass	1
Successful tackle	2
Successful dribble	1
Scoring a goal that wins the game	3
Pass completion of 70-79% (with minimum 30 passes attempted)	2
Pass completion of 80-89% (with minimum 30 passes attempted)	4
Pass completion of +90% (with minimum 30 passes attempted)	6
Conceding a penalty	-3
Missing a penalty	-6
Yellow card	-3
Red card	-9
Own goal	-6
Missing a big chance	-3
Making an error which leads to a goal	-3
Making an error which leads to an attempt at goal	-1

Being tackled	-1
Conceding a foul	-1
Being caught offside	-1
Shot off target	-1

Table 2: In-game actions that determines bonus points

2.0.0 Data Description and Pre-Processing

The following section will cover what kind of data that is used in the report, how it was pre-processed and manipulated, and finally the in-depth analysis made on the data.

2.1.0 Data

The Fantasy Premier League platform comes with a lot of free and available data for managers to use in order to help them make good decisions. It can also be downloaded through their open-sourced API, but this comes with a lot of cleaning. The *Fantasy-Premier-League* GitHub-repository by username *vaastav* (14) has addressed this issue and delivers a cleaned version of the Fantasy Premier League data. The data in question for this project comes from the last three seasons of 2019-20, 2020-21 and 2021-22. The total amount of data then consists of 989 players from 23 teams, playing 1140 games. The dataset details each players performance in every gameweek. The relevant features from these datasets are as follows:

Column name	Description	Data Type
Name	Player name	String
Position	Position of the player	String
Team	Team of the player	String
Assists	Number of assists	Integer
Bonus	Bonus points	Integer [0, 1, 2, 3]
BPS (Bonus Point Score)	Bonus point score	Float
Clean Sheets	If the team conceded a goal	Boolean [0, 1]
Creativity	A metric that measures a player's performance in terms of producing goal scoring opportunities for other players (15).	Float
Fixture	The fixture-number of each season	Integer
Goals Conceded	Number of goals conceded in a game	Integer
Goals Scored	Number of goals scored in a game	Integer
ICT Index	A combinator of Influence, Creativity and Threat (ICT) that measures overall performance of a player in a game (15).	Float
Influence	A metric that measures if a player has made an impact on a game (something that directly, or in-directly, affects the outcome of the game) (15).	Float

Minutes	Number of minutes a player has played the game.	Integer
Own Goals	Number of own goals.	Integer
Penalties Missed	Number of penalties missed by a player.	Integer
Penalties Saved	Number of penalties saved by the goalkeeper.	Integer
Red Cards	If a player received a red card or not	Boolean [0, 1]
Saves	Number of saves made by a goalkeeper	Integer
Selected	Number of managers that has bought the player	Integer
Threat	A metric that measures the threat of a player scoring a goal. The metric looks at the location of the player and their attempts at scoring a goal.	Integer
Total Points	The total points a player has received after a match, including bonus points.	Integer
Value	The price of a player in the game.	Integer
Was Home	Whether or not the team is playing at home or not.	Boolean [0, 1]
Yellow Cards	Number of yellow cards for a player	Integer [0, 1, 2]
Team FDR*	The difficulty rating of the team of the player	Integer
Opponent FDR	The difficulty rating of the opponent of the team	Integer

Table 3: Important features in the dataset

The theory in the project is to capture the performances of the players through these features to get a perspective of the players' form. Combining players' form and the difficulty of the opponent with the fixture difficulty rating will then hopefully give a good indication of how well the player will perform in upcoming gameweeks.

2.2.0 Importation and Pre-Processing

To get the detailed player statistics, the masterfiles of the vaastav datasets was downloaded from their GitHub repository. It was then later merged with the vaastav fixtures data to also contain the Fixture Difficulty Rating. The data from the 2019-20 season lacked two features that was included into the later seasons, namely the position-feature and the "team" feature. The name of the players was also structured differently. This posed a problem for both the later analysis and for the appending of FDR-data due to one of two teams being missing from the fixtures. To circumvent this issue, a forward filling [ffil] was applied to each player. This method fills the empty cells [NaN] with the earliest recorded data after the NaN-values. I.e. the team and the position from the 2020-21 season was applied to the earlier season.

Name	Season	Team	Position
Marcus Rashford	2020-21	Manchester United	MID
Marcus_Rashford_0321	2019-20	NaN	NaN

Table 4.1: Missing values and different name type

```
pd.DataFrame[df['name'] == 'Marcus Rashford'].fillna(method='ffill')
```

Name	Season	Team	Position
Marcus Rashford	2020-21	Manchester United	MID
Marcus Rashford	2019-20	Manchester United	MID

Table 4.2: After dataframe Pandas filling missing values and changing name type

The name is altered by replacing the underscore with space-string and removing the numeric values and the last space.

The forward-fill method will not be 100% accurate seeing as a few of the players switched teams from one Premier League team to another between these seasons, and so they will be recorded with the wrong team. However, since this only affects a few players, this problem is not considered important.

2.2.1 COVID-19 outbreak

In December of 2019, the first known case of the Coronavirus Disease 2019 (COVID-19) was discovered and quickly spread around the world, resulting in a pandemic (16). This resulted in, amongst other things, the temporary suspension of all matches from March 13th (17) until they could be restarted on June 17th. This poses a problem in terms of reading the fixture numbers seeing as they decided to start from gameweek 39 to 48 rather than keep on using the same fixture numbers from where they left off on March 13th. Further problems occurred in the following two seasons when games would be cancelled if there were registered COVID-19 outbreaks in one of the teams until the team became free of COVID. To circumvent this problem, new fixture-numbers are created to give each player a clean timeline from 1 to n, where n is the number of games where the player is registered.

2.2.2 Fixture Difficulty Rating [FDR]

Each team is given their own fixture difficulty rating from 1 to 5 and depicts the difficulty rating of their opponents and of themselves. A fixture difficulty rating of 5 means that the team is one of the best teams in the league and a rating of 1 means that it is one of the worst teams in the league. The rating is a confidential algorithm made by Opta Analytics (18), a third-party company that supplies data and statistics in Sports Analytics. The only description of the rating system is:

"The Fixture Difficulty Rating (FDR) is based on a complex algorithm developed by Fantasy Premier League experts. A set of formulas process key Opta data variables, along with each team's home and away form (past six matches), in order to generate a rank for the perceived difficulty of each Gameweek opponent. The FDR is designed to help FPL managers plan their transfer and team selection strategy and is reviewed on a weekly basis" (15).

To extract the rating, the original player player-detailed dataset was merged with the vaastav fdr-score of the players team and the opponents' team from the "fixtures" datasets. Due to using ffill to fill the players teams from the 2019 season, can't be traced which fixture they played. To remedy this issue, those players will be given an FDR of both players- and opponents team with the median FDR-score of 3.

2.3.0 Exploratory Data Analysis

After collecting the data and pre-processing it, the dataset was analysed for its quality and feature importance.

2.3.1 Player Analysis

Figure 1 is an analysis of how much players tend to play, and which players tend to have the most goal involvements. This shows that most players that play pretty much the entire game are also the ones that tend to score the most goals. However, the graph on the right shows that most players doesn't really get to play a lot of minutes.

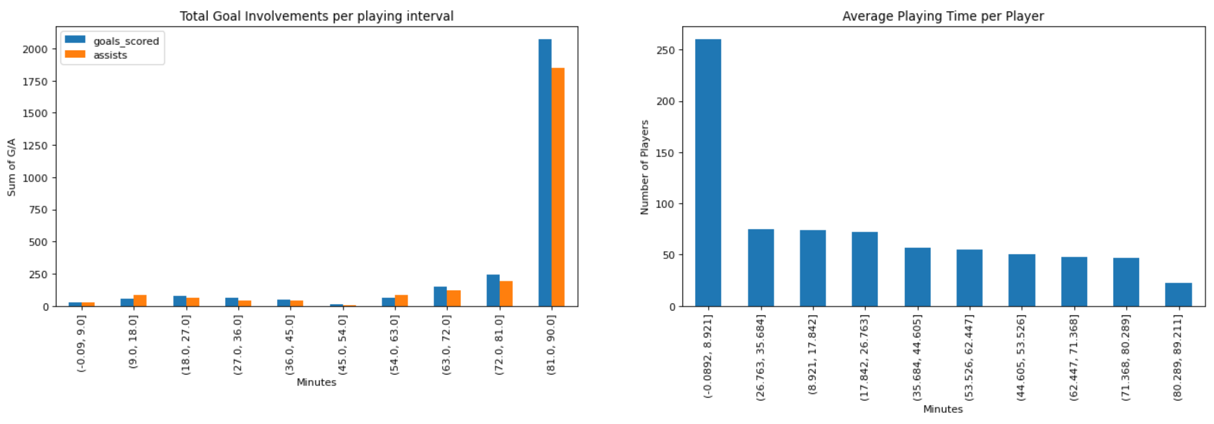


Figure 1: Left graph: Goal involvements for intervals of average playing time
Right graph: Number of players with average playing time of the intervals

The take-away from this is that most players will not be very useful to predict their performances, because they don't get enough game time. By reducing the number of players in the initial dataset, it will help improve the computational run-time of many of the models to come. A threshold will be set of any player with an average playing time of less than 45 minutes (one half) will be removed from the dataset. This leaves 214 players to still be predicted, but this figure could be reduced even further to improve the computational run-time.

Figure 2 tries to show which positions receive the most points. It shows that midfielders and attackers earn the most points on average and the right graph shows that midfielders are also the most represented groups along with defenders. The position with the least points is the goalkeeper position.

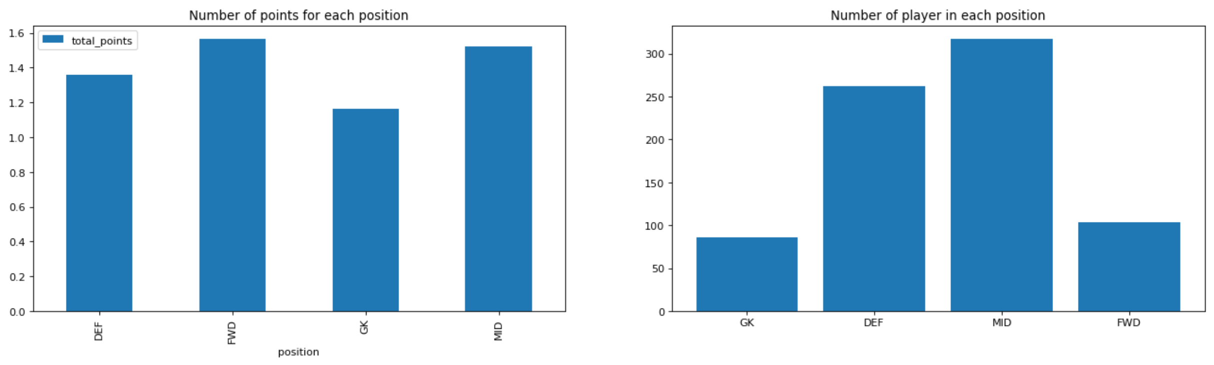


Figure 2: Left graph: Shows the number of points recieved from each position.
Right graph: Shows the number of players in each position

To complement figure 2.1, figure 3 shows the average value of players in each position. The bar-chart is almost identical to that of figure 2.1 which tells that the prices are aligned with the points return of the players.

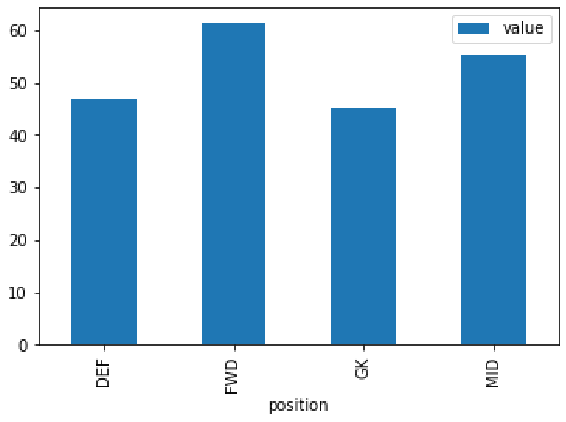


Figure 3: Shows the average price of each position

Figure 4 shows a linear correlation between players point return and an increase in their value.

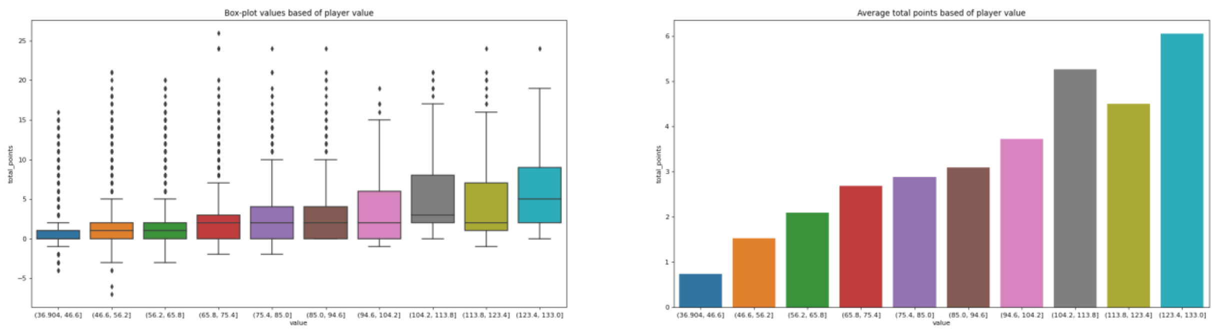


Figure 4: Left graph: Boxplot of point return based on player value.
Right graph: Bar chart of point return based on player value.

2.3.2 Fixture Difficulty Rating Analysis

The FDR is a tool made by experts with little detailed description about it, so one could be excused if they did not put much faith in it. Figure 4 shows four sub-plots that depicts the average goals scored for the home- and away team, when the home team has a fixed FDR-rating with the away team having a varied rating.

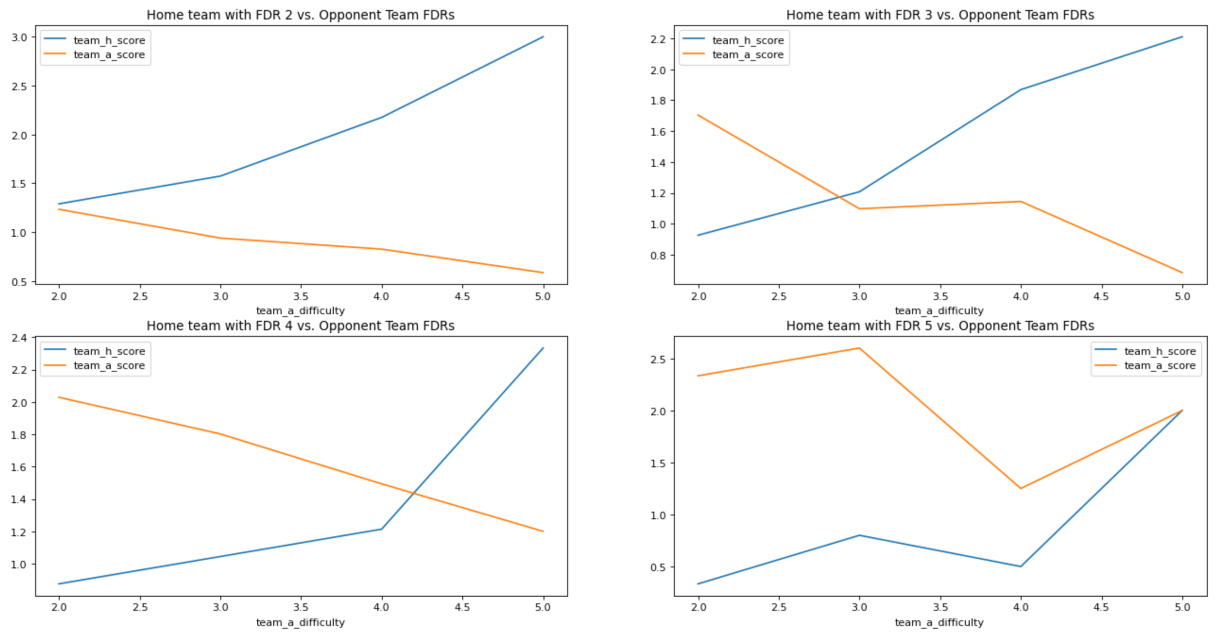


Figure 5: Fixture difficulty rating for each home-team FDR rating

The figure 5 sub-plots show that the FDR-rating works pretty well seeing as weaker teams (low FDR) tend to score less than stronger team (high FDR). Optimally, the most desirable players should then have a high of an FDR-rating as possible, and their opponents should have a low FDR-rating. However, when looking at the FDR in the FPL application, one team could have a higher FDR if they play home than if they play the same team away. This means that the FDR do give a home team advantage when making their predictions. Figure 6 depicts the number of wins for home teams, away teams and draws. It shows that home teams win approximately 20% more than away teams, which gives reason to setting the FDR rating slightly higher for home teams.

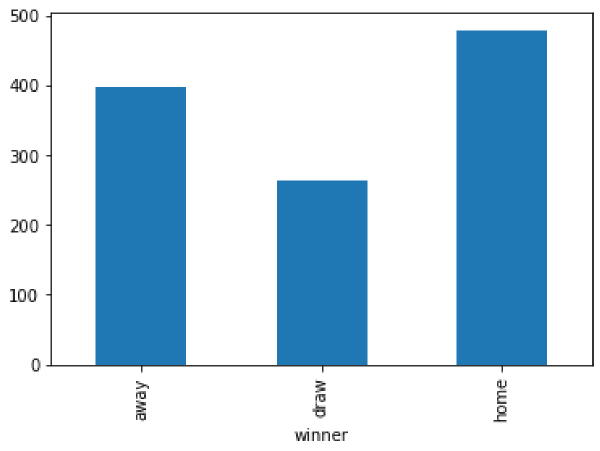
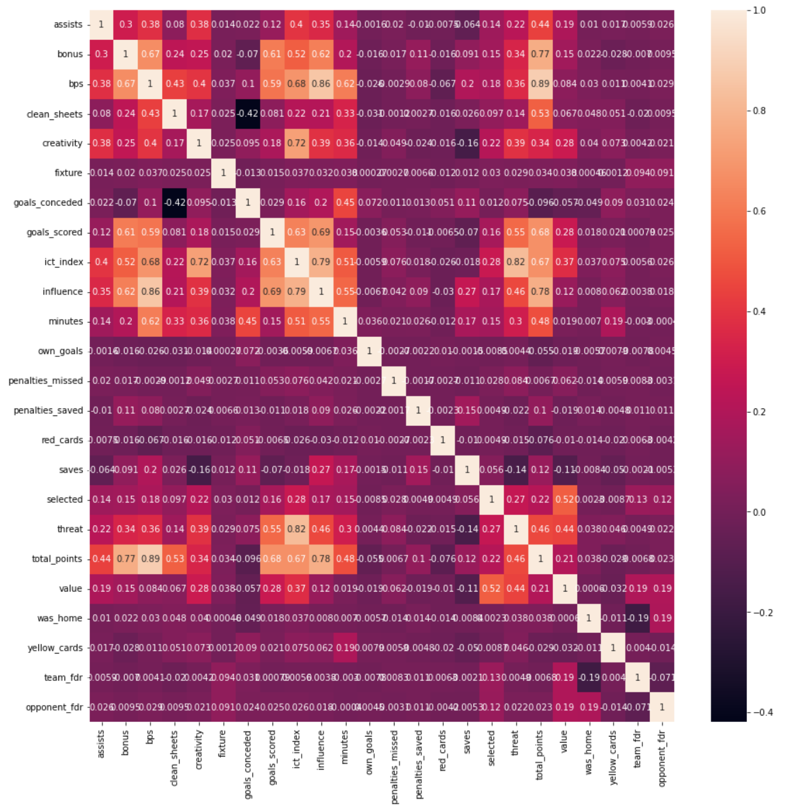


Figure 6: Number of winners for home- and away games and draws.

2.3.3 Correlation Analysis



The correlation matrix is a tool to prove the linear correlation between features, and thus the overall quality of the data. The target data is directly dependent on most of the features seeing as an occurrence in those features return points to the player. However, since the features don't always occur, the linear correlation between them and the total points aren't highly dependent. The correlation matrix does show, however, that total points is highly correlated to goals, bonus points, assists and the ict measurements which could be interpreted that the game heavily favours attacking players.

2.3.4 Stationarity Analysis

In the ARIMA-model, stationarity is an advantage to predicting the future. The more stationary the data is, the easier it will be to predict a value. Stationarity refers to a non-changing distribution of the data over time, and variations such as trends and seasonality are two factors that disrupts stationarity. It won't be constructive to estimate stationarity of every player in the dataset, however, aggregating the data will not help either, as it will naturally make a more stationary distribution. A solution that is applied in this research is to estimate the stationarity of 5 desirable players in different positions. If they all show stationarity (or not) then it could be assumed that other players are the same.

The 5 players in question are : [Mohamed Salah, Kevin De Bruyne, Virgil van Dijk, Harry Kane, Ederson Santana de Moraes]

Figure 7 depicts one of the 5 players in question, Kevin De Bruyne (KDB). The rolling average and std on the left graph, with a window size of 5 time steps, shows that the total points of the player is not stationary in any other game other than the times the player was injured. The right graph shows the autocorrelation of the same total points. It should be smooth, but it is definitely not which is another sign of non-stationarity. This was the same case for all of the other players as well.

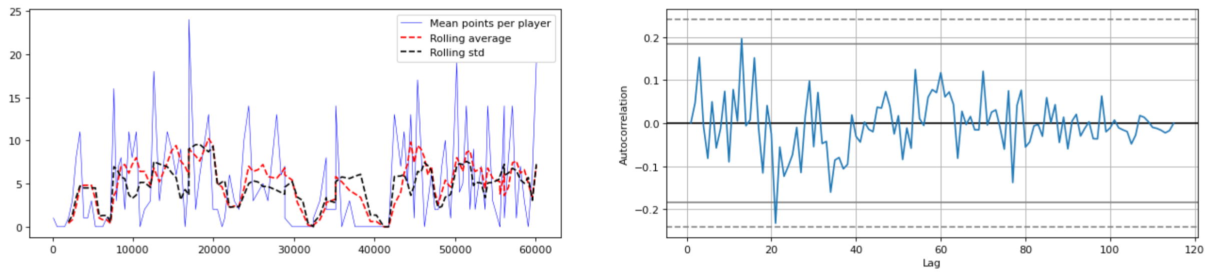


Figure 7: Left graph: Depicts a Rolling Average and Std. and Kevin De Bruyne.
Right Graph: Depicts the autocorrelation of KDBs Total Points

Another method, with a more definite conclusion to understanding if a feature is stationary or not is the Augmented Dickey-Fuller test. This method is a hypothesis test where the null hypothesis states that the data is non-stationary and the alternative hypothesis states that it is stationary:

$$H_0 : \mu \geq 0.05 \rightarrow Data\ is\ non-stationary$$

$$H_A : \mu < 0.05 \rightarrow Data\ is\ stationary$$

The ADF-function returns the p-value that gives the probability of obtaining test results at least as extreme as the observation. Most of the players failed the test seeing as the ADF-test returned an optimal lag value of 0, which automatically gives p-value of 0 (because it doesn’t run on previous values). The only player who didn’t fail the test was Virgil van Dijk who was injured for a long period of time and thus had a full season of 0 value returns.

2.4.0 Log Transformation

Log Transformation is a data transformation method where the data is replaced by the logarithm of the variable. A logarithm, or simply *log* for short, is defined to be the inverse function to an exponentiation, i.e.: from a given feature variable (x) and a base value of the logarithm, the log of x is then the exponential of the base that returns x. In mathematical terms it is written as:

$$Y = \log_b(X)$$

$$\downarrow$$

$$b^Y = x$$

Log transformations is a very useful tool in statistical analysis because it transforms normalizes non-normal (skewed) feature distributions. It does so because a skewed feature as the one depicted in the left graph in Figure 8 is exponential, and as stated above, the log is the inverse function of exponentiation (19). The right graph in Figure 8 depicts the application of the log transformation to the skewed feature. Using the log on a variable improves the fit of the model by transforming the distribution of the features into a distribution closer to that of a bell-shaped curve.

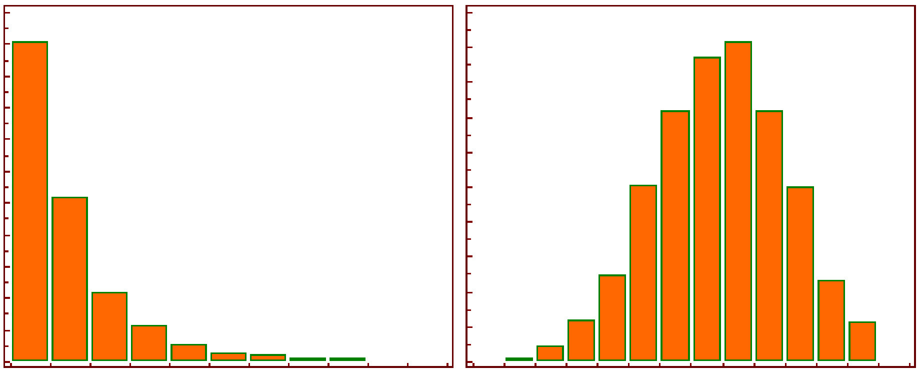


Figure 8: Left graph depicts a skewed distribution, and the right graph depicts the same feature with log transformation applied.

There is however a problem with applying a log transformation to certain datasets, as the log of 0 is undefined. This is easily explained by remembering that no exponential value equals zero, so the log function will not work on zero-values. A simple way of circumventing this issue is to add each zero-value with a number significantly closer to zero than any other value. Another problem is handling the log of a negative number. In the case of this research, negative values are uncommon and usually close to zero, so to treat them and the zero values together, all values equal, or lower than zero will be treated with the log of a value close to zero.

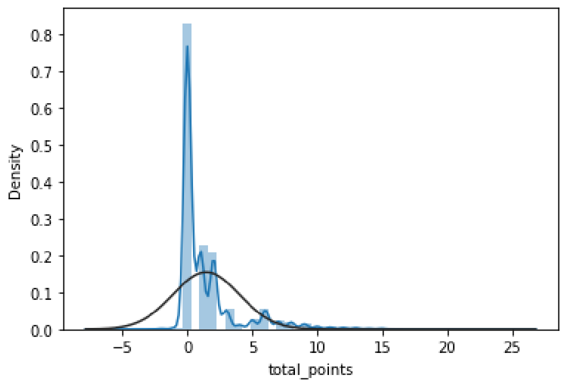


Figure 9: The distribution of the target variable, with a comparison of the bell curve

Figure 9 depicts the distribution of the target variable. It shows that the distribution is skewed to the left. After applying the log transformation, the distribution is flatter, and when ignoring the zero values on the very end, the point distribution looks to be more normally distributed.

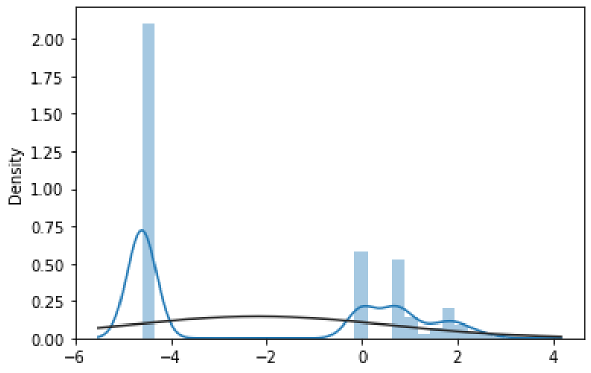


Figure 10: The same feature as in fig 9, but after log transformation

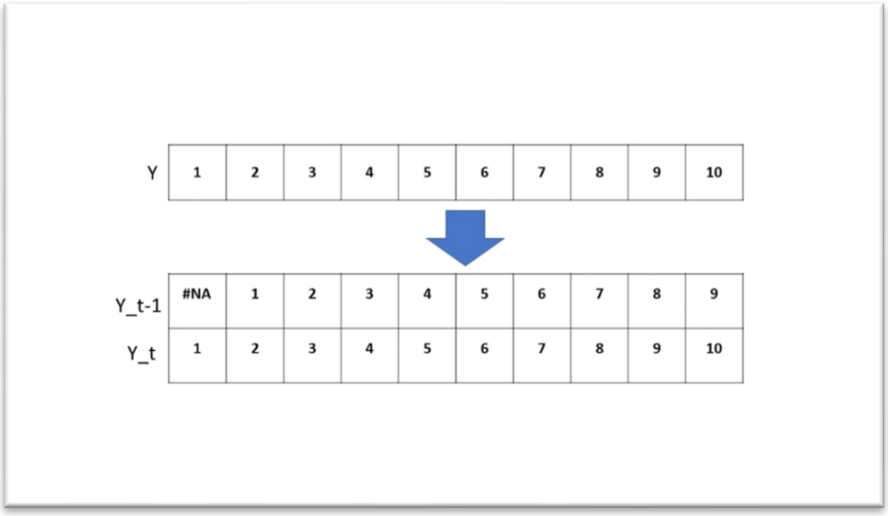
2.5.0 Transforming Time Series data into Supervised Learning data

After an initial pre-processing of the dataset, the data is still not ready to be fully processed by the Machine Learning models. For the ARIMA model, the input data is quite skewed and non-stationary, so as stated in section 2.4.0, the data is run through a log transformation. However, with the multivariate models, they require some more work. For starters, the predictor features before importation are in-game statistics, meaning that they tell the story of what is happening in the game that is currently trying to be predicted. Leaving them unattended will make the model overfit with the data and return overly optimistic results in terms of low error rates. The point of using those statistics isn’t to predict what the total points are by what happened in the game, it is to tell the story of how good a player has been for the past few games. To do that, each feature containing in-game statistics will be turned into averages of their previous games in a given window, meaning that for a window size of 6, the player statistics for gameweek t will be the average of their previous 6 games.

$$player\_stat[feature]_t = \frac{\sum_{i=1}^{w} player\_stat[feature]_{t-i}}{w} \leftarrow given\ that\ t > i$$

Where w is the window size. For the initial games before there is collected more data than the window size parameter, the window size is reduced to the number of games with data collected for the player.

The second pre-processing step to make is to introduce a re-organization algorithm that uses a sliding window from the time series to stack previous time steps together to create a sort of “memory” in the input data. This technique is illustrated below in Figure 11 where variable Y is slided back one time step. This could be done multiple times in order to create a longer memory of the previous values of Y.



Y	1	2	3	4	5	6	7	8	9	10

Y_t-1	#NA	1	2	3	4	5	6	7	8	9
Y_t	1	2	3	4	5	6	7	8	9	10

Figure 11: Backwards sliding window

3.0.0 Model Selection

“The simplest solution is almost always the best.”
Occam’s Razor (20)

The famous quote from William of Ockham advocates that when presented with competing hypotheses about the same prediction, one should select the solution with the fewest assumptions. This principle has since, largely been applied to scientific research as a reminder not to overcomplicate their proposed solutions to a problem where higher complexities are not needed.

Machine Learning is a branch of Artificial Intelligence where a model is presented with data and from this, is capable of learning, identifying and making decisions without humans having to manually tweak and set conditions of the model as a mathematician would do. There are numerous different types of Machine Learning approaches that classifies the different models, with Supervised Learning, Unsupervised Learning and Reinforcement Learning being the three most normal approaches. This dissertation will mainly focus on Supervised Learning for solving the problem.

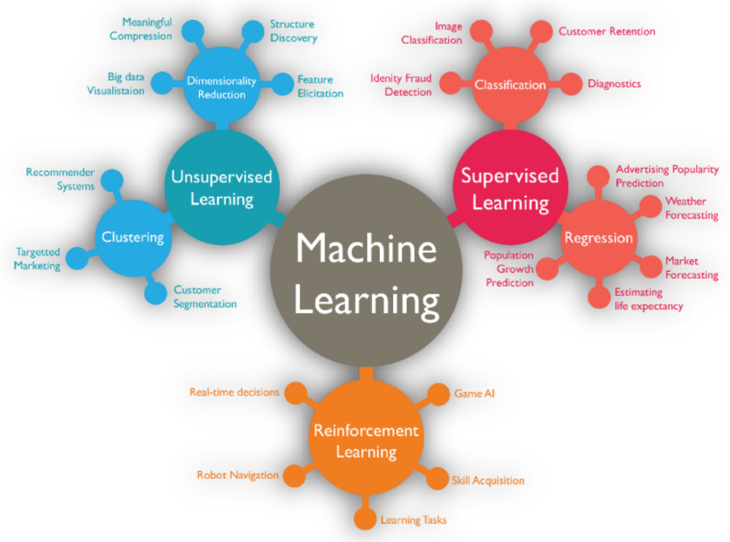


Figure 12: Depicts the most commonly used Machine Learning approaches and their user cases (source: (21)

Supervised Learning uses labelled data to either accurately predict or classify outcomes in the future. When input data is fed into the model, it will automatically adjust its weights until the model best fits to the datapoints or accurately classifies each category. In order for the model to best fit to a dataset, it uses a loss function to calculate the error rate, then it will adjust the weights until the errors have been minimized. In order to calculate the error rate of the model it is implied that the ground truth is known for the data and without it the model will not work.

This research work will only take use of regression algorithms in order to predict the players performances. This means that the model will attempt to fit a regression line (or a hyperplane for the SVM) that most accurately fits with the target variable with the least loss. With a *normal* regression problem, the data tends to be randomized in order for a fair result that prevents biases, however, with time-series problems this will not be suitable as their current values may be affected by previous values. E.g., when predicting what the weather will be in an hour it is important to know what the weather is now. If it is pouring rain now, it is less likely that the sky will be clear of clouds in the next hour.

3.1.0 Autoregressive Integrated Moving Average (ARIMA)

The ARMA/ARIMA model is a statistical analysis model used for time-series data to predict the future. It was originally applied as an econometric model but has later been applied in other areas of science as well (22). Because of the wide-spread popularity of this model in econometrics and the strong resemblance of this problem of that of an econometric problem such as valuation of a commodity, this model has been selected as a suitable candidate to predict player performances.

The ARMA model is a combination of two other models, i.e. the Autoregression model (AR) and the Moving Average model (MA). An additional differencing step (I) can be applied, but in the case of this research, after tuning the parameters, the integration proved not to be useful.

3.1.1 Moving Average – MA(q)

The Moving Average is defined as: $MA(q)\colon y_t = \mu + \epsilon_t + \sum_{i=0}^{q} \varphi_i \epsilon_{t-i}$

Where:

ε_t = random noise
μ = The expected average of y
φ = Weight parameter of the Moving Average
q = A lag parameter to set the size of the window average

The moving average is used to smooth out the autoregressor and thus remove noise from the full model.

3.1.2 Autoregression – AR(p)

An autoregressive model is a model where the current value depends on the values that the current variable took in its previous periods (plus an error term) (23).

The Autoregression is defined as: $AR(p)\colon y_t = c + \epsilon_t + \sum_{i=0}^{p} \theta_i y_{t-i}$

Where:
θ_i = a model parameter
c = A constant
p = A measure of the autocorrelation between successive values of the time-series.

Setting the parameter p to 0, the model would simply output the noise of the model. When p is increased to 1 and upwards, the model starts to output a variable that is dependent on its previous values for the period (p), plus the error term.

In order for the AR model to accurately estimate the next value in the time-series, it is highly dependent on the stationarity of the input data. One of the reasons for this is due to the fact that the model bases itself on predicting variables based only on past information, major changes to this will then lead to inaccurate results.

3.1.3 Differencing Integration – I(d)

The final differencing parameter (I) is a transformation that is applied to a non-stationary time-series to make it more stationary. The integration function is a pre-processing for the ARIMA model that replaces the current time step t by subtracting it with a previous time step (t-d). The function is defined as:

$$I(d): y_t' = y_t - y_{t-d}$$

Where d = the duration of the period.

Differencing removes the changes in the level of the time series, by stabilizing the mean of the time series (24).

3.1.4 Putting it all together to get the AR(I)MA

After defining each individual component of the ARIMA it is simple to put the model together. As explained in section 3.1.3, the integration function is a simple pre-processing algorithm to make the input variable more stationary by subtracting it by an earlier time step. When this is done and the input variable is processed, it will pass through the ARMA function, which is a combination of the Moving Average function and the Autoregression function. The input variable is first passed through each function and their outputs are added together. In its entirety, the function looks like this:

$$I(d): y_t' = y_t - y_{t-d}$$

$$ARMA(p,q): y_t' = \mu + \epsilon_t + \sum_{i=0}^{p} \theta_i y_{t-i} + c + \sum_{i=0}^{q} \varphi_i \epsilon_{t-i}$$

In conclusion, the ARIMA tries to replicate the previous time steps, with an added measure of instability that comes from the autocorrelation, then tries to smooth it with the moving average. This is the reason why the ARIMA demands stationarity.

3.2.0 Support Vector Machine (SVM)

Support Vector Machine is a robust Machine Learning algorithm that is very commonly used in performing regressions, classifications and outlier detection of both a linear-, and non-linear fashion. The fundamental idea behind the SVM algorithm came from Vladimir Vapnik and Corinna Cortes (25) in 1995 where they presented a support vector regressor that was capable of handling polynomial (non-linear) data, commonly referred to as the *kernel method*. In terms of the support vector regression algorithm, it outputs a hyperplane that fits the closest to the inputted datapoints. In contrast, the classification term is completely opposite of the regression function by the SVM outputting a hyperplane that separates each classification category the most. There are different types of kernels to use depending on the data, but this dissertation will focus on the Gaussian RBF kernel, which is available in the sci-kit learn library (26). A linear kernel was also considered for this problem setting, but in order to capture trends (i.e. increasing performances), the Gaussian RBF is ultimately the kernel that is chosen for this problem. The RBF kernel function is defined by:

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\left\|\vec{x}-\vec{l}^i\right\|^2}{2\epsilon^2}}$$

Where:
K = Gaussian RBF Kernel function
X = input vector x
l = Landmark vector
i = number of landmarks
ε = a fixed parameter

This function uses a landmark vector to find the most optimal placement of the regression line by eliminating outliers that create noise for the dataset. In the function, outliers in vector x will create a large deviation from the landmark, then this large value will be turned negative. This large negative value is exponentiated by eulers number, which turns the large outlier-number close to zero, thereby eliminating its importance to the regressor. The fixed parameter (ε) determines how large the outlier boundaries will be set. A large epsilon decreases the likelihood of value x to be determined as an outlier and a small epsilon increases the likelihood, this is exemplified in the image below using a linear Kernel model (but the intuition is the same with the Gaussian RBF).

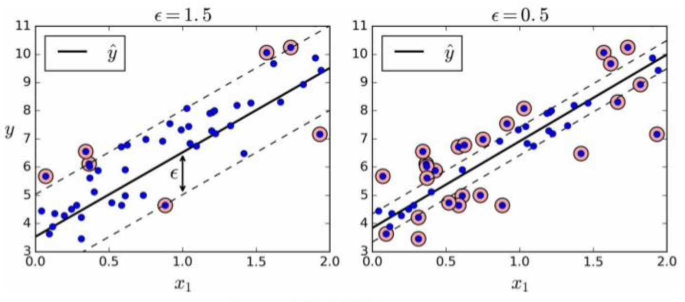


Figure 5-10. SVM Regression

Figure 13: Image gathered from the blog of Yang Xiao Ling *(27)*

The Support Vector Machine is chosen in this dissertation for its capabilities of detecting outliers and reducing their significance in fitting the regression line. Seeing as outliers are extremely common in football (players doing well for single games) this could be a useful Machine Learning model to predict performances of football players in FPL.

3.3.0 Gradient Boosting Regression (XGBoost)

The Gradient Boosting technique known as XGBoost is one of many different types of Ensemble Learning algorithms that uses multiple Decision Trees to predict Supervised Learning problems. The intuition of the Ensemble methods comes from a theory called *Wisdom of the Crowd* (28), which states that the aggregated answers of many is better than that of a single predictor. The very popular Random Forest is exactly one of these Ensemble methods that combine Decision Trees, but the difference between an algorithm such as the Random Forest and the XGBoost lies in how they optimize the decision trees. In this dissertation, the focus will be on the XGBoost and how it uses the Gradient Boosting method to tweak the weights of the next decision tree predictor by the residual errors that was made by the previous predictor.

3.3.1 Decision Tree Algorithm

A Decision Tree is a Machine Learning algorithm that uses many if-else statements to approximate a regression line or a classification separator that fits best with the data at hand. Each decision is represented as nodes in a data structure, usually binary trees, where the nodes diverge towards two new nodes. The decision tree starts by sampling the attributes of the training data and asks whether the predicted value should be larger or smaller than the sampled value. By reviewing the input features, it makes a decision based on the *impurity* (randomness) of the data. E.g.: When predicting the height of an individual and the individual has a shoe size of 46 (European size), with the training data showing that only individuals taller than 1.85 metres has a shoe size this big, then a prediction that the man is taller than 1.85 metres would be much *purer* than a prediction that he is smaller than 1.85 metres. Thus, in the decision tree, the decision would be more likely to moving up than down. These if-statements are repeated at different values until the model is capable of making a good approximation of the data. To visualize how a decision tree regressor operates, look at Figure 12. The starting point is referred to as the root node and depending on the impurity, moves down to depth 1 by either yes or no. And so it goes until it gets to the bottom of the tree where it determines the most likely value of y.

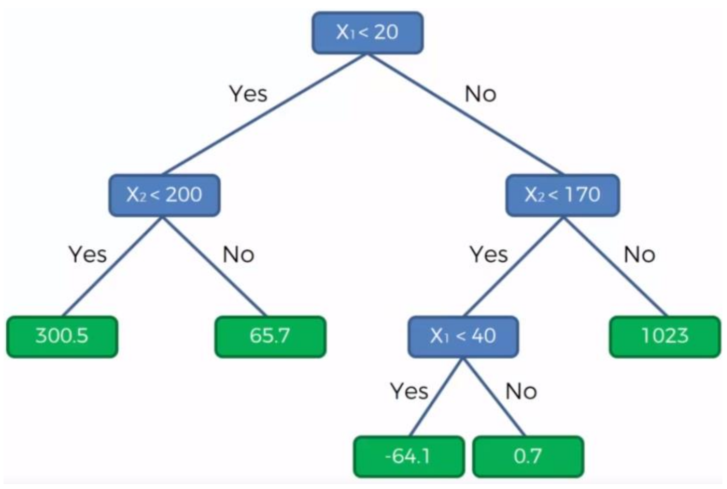


Figure 14: Decision Tree Regressor of target value y, based on input values x1 and x2 *(29)*.

The SciKit Learn library uses an algorithm named *CART* to train its Decision Trees (30) which produces only binary trees, such as the depicted image in Figure 14. Other algorithms, such as the ID3 does produce decision trees with multiple branches, but since the XGBoost is a library function based on the SKlearn library, it uses the CART-version as well (31).

Purity is measured by one of two different purity measures. One of those measurement is Entropy:

$$H_i = -\sum_{k=1}^{n} p_{i,k} * \log_2(p_{i,k})$$

And the other one is Gini:

$$Gini_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

Where $p_{i,k}$ = The probability of event *k* occurring in the training data, in node *i*.

The differences between choosing entropy or Gini are usually not that dissimilar and tends to result in very similar decision trees. The biggest differences between the two trees are that Gini tends to isolate the most frequent class of the tree whereas entropy creates more balanced trees. Therefore, in this dissertation work, the chosen impurity measure for the algorithm was the entropy-function, though, this was not based on performances since they were very similar.

3.3.2 Ensemble Learning and Optimized Gradient Boosting

Ensembles are in simple terms a collection of weak learners (i.e. singular weak predictors) with different ways of aggregating the results of each learner to make for one strong learner. XGboost is an Ensemble method that uses a version of Gradient Boosting called optimized Gradient Boosting which was developed by Tianqi Chen in 2014 (32) and has quickly become one of the most popular open-sourced Supervised Learning algorithms today. XGBoost uses an ensemble of decision trees to predict the outcome of a dataset. This means that it splits the training set into as many subsets as the number of estimators. This is depicted in Figure 15.

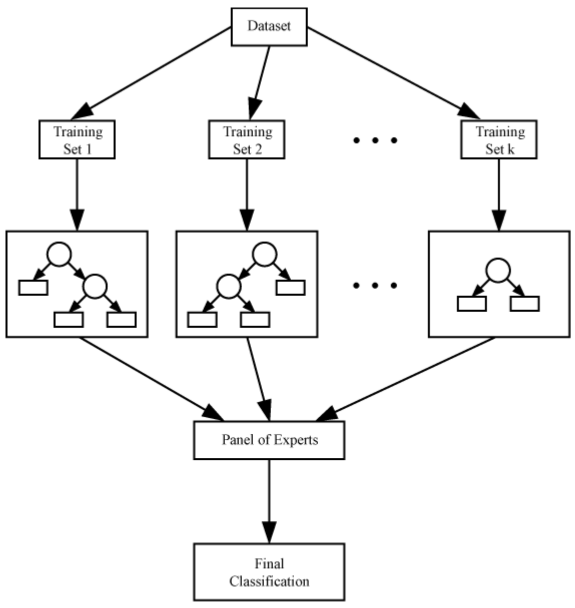


Figure 15: Ensemble method

The Gradient Boosting algorithm works by fitting the next decision tree in the ensemble by the residuals from the previous regression tree. This can be written in algorithmically terms as:

$$tree_1 = Decision\_tree\_regressor.fit(x, y_1)$$

$$y_2 = y_1 - tree_1.predict(\,x)$$

$$tree_2 = Decision\_tree\_regressor.fit(x, y_2)$$

$$y_3 = y_2 - tree_2.predict(\,x)$$

$$\ldots$$

$$tree_k = Decision\_tree\_regressor.fit(x, y_k)$$

$$y_{k+1} = y_k - tree_k.predict(\,x)$$

Where k is the number of estimators. In order to scale the contribution of each estimator in the ensemble, a learning rate is applied. However, if the learning rate is too low, the model will require more estimators to be able to predict with a high accuracy.

Since the data used in the project is lacking explainability, as discussed in section 1.1.0, decision trees tend to be good candidates for generalizing data with these problems. As explained above, the XGBoost is an Ensemble of multiple decision trees, so the hope for this model is that it will be able to make better predictions than using a single Decision Tree. The XGBoost is a Machine Learning algorithm that works best with larger sets of training data. This could be a good contrast to the Support Vector Machine algorithm that is well suited for smaller amounts of training data (though, not suggesting that the SVM is any worse with large training sets).

3.4.0 Recurrent Neural Network

Recurrent Neural Networks (RNN) is a class of neural nets that aim to predict an event using past (and sometimes also future) data. As the topic of RNNs can be quite long and dense, this explanation will take some short-cuts in describing the model in question. It was originally built to handle Natural Language Processing (NLP) problems but has become a very popular model for time series analysis in later years.

With standard artificial neural networks (ANN), the data flows forward in one direction, from the input to an activation function (neuron) and then outputted from said function. With RNNs however, the connections between layers flow backwards as well, by the activation function receiving an input, producing the output and then passing the output both forward to the next layer just as the ANN, and also back to itself again to be processed again with the next sequence data (30). Figure 16 shows this comparison where the left-side model is a recurrent neural network and the right-side model is a (artificial) feed-forward neural network.

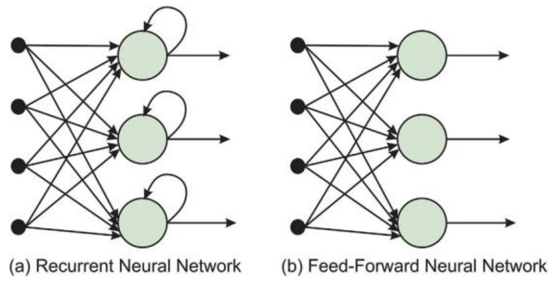


Figure 16: Comparison of RNN (a) and ANN (b) (source (33))

In the initial time step, since there are no other time steps before it the recurrent layer is at zero, so the neuron only handles the input data from the current time step and outputs it. At the next step, the neuron will then receive both the input data from the current time step, and the output from the previous time step. This is depicted in figure 17 which, on the left, depicts the recurrent neuron and on the right, it shows how each time step is receives information from the previous time step as well as new information from the input.

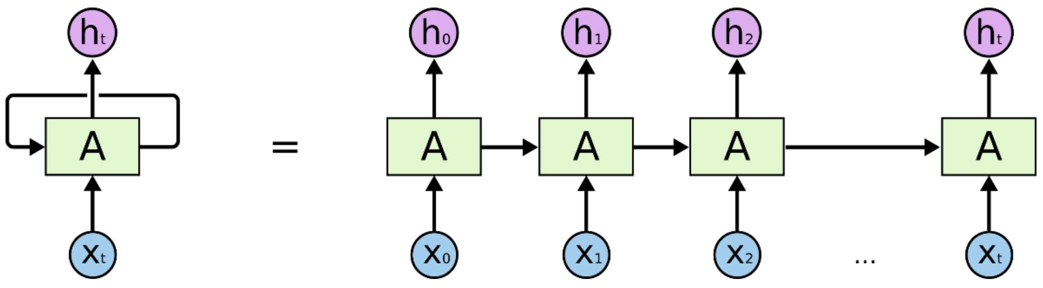


Figure 17: Depicts the recurrent neuron (source: *(33)*)

In mathematical terms, with a regular feed-forward network, a single instance would be described as:

$$y = w^T x_t + b$$

With:

W = weight
X = Input data
Y = Output data
b = Bias term

The recurrent neural network, however, requires an additional set of weights to manage the recurrent data from the previous time step:

$$y_t = w_x^T x_t + w_y^T y_{t-1} + b$$

Because the output of a neuron is a function of the previous time step, and the previous time step is a function of the time step before, it is possible to look at this function as a form of decision making based on memory of previous events.

3.4.1 Backpropagation

Just like a regular ANN, the RNN first feeds forward the input and passes it through the network, then the output sequence (in this instance, a single scalar of time step t) is evaluated with the use of a performance measure, in this research it is the Mean Squared Error. The gradients of the performance measure are then backpropagated through the network using and the network weights are updated using the gradient function.

For this research, the Adam Optimization function is used to calculate the gradient descent of the loss function (MSE). It works by combining the idea of two other optimization method, AdaGrad and RMSProp (34), and it works by tracking the decaying average of past gradients.

3.4.2 Recurrent Neural Network with Long Short-Term Memory

In theory, the Recurrent Neural Network is capable of holding the memory of very long sequences, however this has been disproven in practice, due to the *Unstable Gradients Problem* (33). The Unstable gradient problem is a problem often occurs in deep neural networks and states that the gradient tends to either explode or vanish in the early layers. To elaborate, supposing that the gradient descent updates the weights at the first time step slightly. Then, since the same weights are used at the second time step, those weights are also slightly increased, and this goes on for the entire sequence until the gradients explode (35). The vanishing gradient works the same way, apart from that the weights are decreased until the gradient vanishes. This problem means that the model will either give too much dependence on past time steps, or too little. In order to counter this problem, some form of normalization has to be applied, but this solution has its own problems in that it loses information at every time step and at a certain step, it looses virtually all information about the initial time steps (30).

Long Short-Term Memory networks (LSTMs) is a kind of RNN that doesn't have this problem. Figure 18 depicts a single neuron/cell of the LSTM. In this report the LSTM will be considered as a black box seeing as unpacking the LSTM is quite a job.

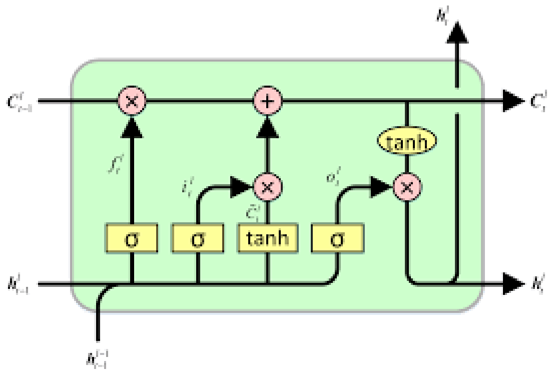


Figure 18: Inside the LSTM cell (source: *(36)*)

3.4.3 Dropout

Dropout is a technique used in neural networks to reduce overfitting by dropping a fraction of neurons in a given layer to make the neurons less dependent of each other. For every epoch, the neurons in the given layer are either dropped with a probability of *p* or kept with a probability of *1 – p*.

3.4.4 The Structure of the RNN-LSTM Research Model

The model that will predict player performances will have the following structure:

```
Model: "model_103"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_104 (InputLayer)      [(None, 72, 1)]           0

 sequential_103 (Sequential)  multiple                 0

 lstm_206 (LSTM)             (None, 72, 64)            16896

 dropout (Dropout)           (None, 72, 64)            0

 lstm_207 (LSTM)             (None, 64)                33024
 dense_103 (Dense)           (None, 1)                65

=================================================================
Total params: 49,985
Trainable params: 49,985
Non-trainable params: 0
_________________________________________________________________
```

- The first layer is simply an input layer to feed the data into the next layers. The output shape shows the number of features to be considered in each time step.

- The second layer, “Sequential_103” isn’t really a layer, but a direction that tells the model to traverse ascendingly with the time steps.

- The third layer is the first RNN-LSTM layer that the data is processed through, as described in section 3.4.2.

- The fourth layer, like the second layer isn’t really a layer, but an information drop-out function as described in section 3.4.3.

- The fifth layer is an additional LSTM layer. It is quite common to stack multiple layers in an RNN model in order to increase the performance of the model. Most state-of-the-art models these days are versions similar to RNN with multiple layers, such as the WaveNet and ConvNet.

- The final layer is the output layer where the information that has been passed through the Deep Network is translated back into the total point sequence.

In total, the model traverses through 49.985 parameters with each layer of the deep RNN, containing 64 neurons in each layer. In order to optimize the model, the model should be tuned with different number of layers, neurons in each layer, dropout functions and much more. But unfortunately, this is very time consuming and quite computationally heavy so because of time constraints in the project, optimization is not performed for this model.

3.5.0 Performance Measure

There are more performance measures that can be are capable of telling how well a model fits with the real data, such as the R^2, Adjusted R^2 and the Median- Squared Error and Absolute but for this research, the Mean Squared Error (MSE), the Root Mean Squared Error (RSME) and Mean Absolute Error (MAE) are the ones to be considered. When selecting a measurement to describe the accuracy of a model, the most commonly applied metric tends to be the RMSE. The function for each algorithm is written as:

$$MSE = \frac{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}{n}$$

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}{n}}$$

The error metric is well suited for visualizing the error of a model because it reduces the significance of accurate predictions and increases the significance of worse predictions. The idea of squaring (real – pred) is because e.g., being off by 10 should be more significant than being off by 5. However, when the difference between the target variable and the predicted variable is low, removing the root will more easily visualize the differences between each performance (e.g.: it is easier to see that an error rate of 6 is worse than 5, rather than $\sqrt{6} = 2.44948$ is worse than $\sqrt{5} = 2.23607$.

Another performance measurement to be considered is the Mean Absolute Error metric, which calculates the average absolute distance between the predicted value and its target and is written as:

$$MAE = \frac{\sum_{i=0}^{n}|y_i - \hat{y}_i|}{n}$$

Both MSE and MAE do practically the same by subtracting the real value with the predicted and dividing the sum of them by the number of instances, so the only difference between the two is that the MAE are being considered in a linear fashion, whereas in the MSE they are considered exponentially. In short, because the MSE weigh highly inaccurate predictions more heavily than predictions closer to the true value, then this metric will be used to measure the error rate of the models in this dissertation.

3.6.0 Validation Method

Validation is a very important part of understanding the true accuracy of Supervised Learning models. Due to the fact that the model is trained on a dataset with known target values, the only way of actually knowing if the model generalizes well on unseen data is to actually test it on unseen data. K-Fold Cross Validation does this by splitting the dataset into *k* sets (/folds), e.g. 5 sets, and then uses 4 folds to train the model and 1 fold to test it, then calculates the accuracy. This process is repeated for as many iterations as folds and the test set changed to a currently unused test set, and the previously used testing set becomes a part of the training set. When the process is finished, the accuracy is aggregated based on the results of each iteration. This whole process is visualized in Figure 19 below.

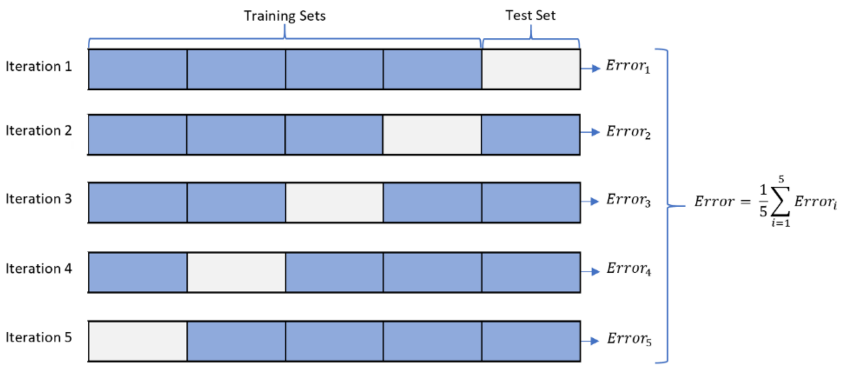


Figure 19: Depicts a 5-fold cross validation

There is however a flaw with this method when the goal is to predict time-series: For one, cross validation assumes that the order of the data does not make any difference and so the dataset will be randomized in order to prevent biases and unfair results. As stated in section 3.0.0, in Time Series Analysis, order is extremely important and when predicting time step t, the model needs to know t-1. When performing time series analysis with the aforementioned randomization, the outcome can often lead to inaccurate and overconfidence (37).

One validation method that handles time dependent data is the Walk-Forward Validation method. This method is a commonly used validation method for time series analysis because it trains the data on past data (t – n) and tests it on newer data (t). This will remove the overoptimism that is caused by cross-validation and give a more realistic performance estimate. The method works by supplying the Machine Learning model with a small subset of data to train the model on, then testing it on the immediately following datapoint(s). This action is repeated until the model has iterated through the entire dataset. The process is depicted in Figure 20 below.

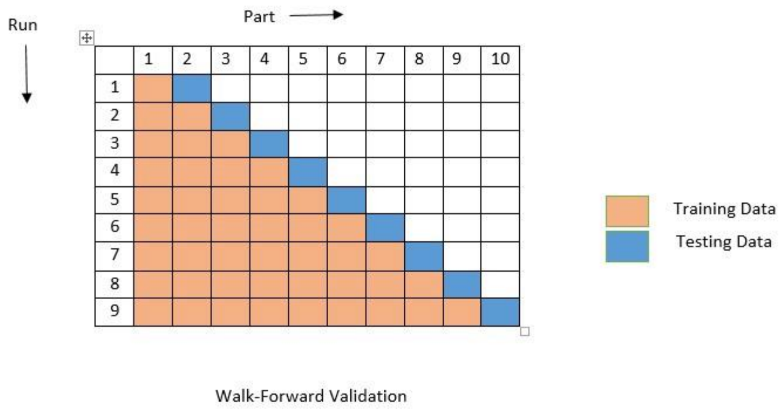


Figure 20: An illustration of Walk Forward Validation where orange represent training data, blue represent testing data and white represent a left out portion of the dataset. In this illustration, the dataset is split in 10 parts, which could either be single time steps or sets of multiple time steps. (source: *(38)*)

4.0.0 Results

To find the model that best predicts players performances, the model will run through each of the players and then calculate the average MSE of every player’s results. To tune the hyper-parameters of the model, a simple approach is taken by tuning the model on a single player, in this case, Mohamed Salah, and using the same hyper-parameters for all players. However, a more effective approach would be aggregate the results of each player, but this would be extremely computationally heavy, so it was not done.

4.1.0 ARIMA Results

After tuning and validating the ARIMA, the results showed that the parameters best suited to predict the results of the time series, were also the lowest values:

Parameters	Parameter setting
Autoregression (p)	1
Integration (d)	0
Moving Average (q)	1

Table 5: Optimized parameter settings for the ARIMA

From this parameter setting it is clear to see that the model struggles to find any accurate way of predicting the players' performance, and by increasing the parameters (i.e. increasing the time dependencies), only adds more noise to an already inaccurate model as depicted in figure 21.

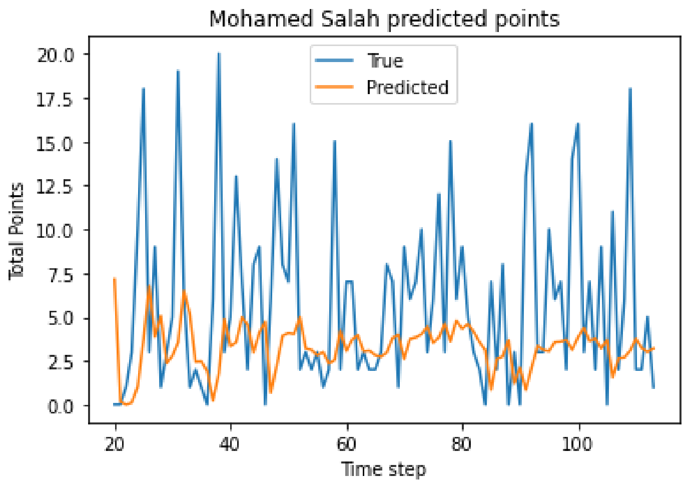


Figure 21: Predicting player performances with the ARMA model

Figure 21 shows that the ARIMA model-, because the hyper-parameters were set so low, the autoregression of the model is set very similar to that of the previous gameweek. The Moving Average tries to reduce the noise from the autoregression, but since the data is so non-stationary, it struggles to make any accurate predictions.

4.2.0 Support Vector Machine Results

When tuning the hyperplane (epsilon) of the model, the results show that a narrow hyperplane is a better estimator. Similar to the ARIMA, this could be due to the noise and non-stationarity of the target data across the time series that makes the SVM struggle to accurately predict the next performances.

Epsilon	MSE
0.1	28.047
1.0	35.641
5.0	51.835
10.0	51.835

Table 6: Error rate for a changing hyperplane (epsilon)

Figure 22 show that the model quite clearly underfits with the predictions. Underfitting can be caused by a model that is too simplistic or input features that fail in explaining the output in a good enough way. Indeed, the data do fail to accurately explain the target variable, as stated in section 1.2.0, but the model as well seem to be a bit too simplistic to be able to accurately predict performances.

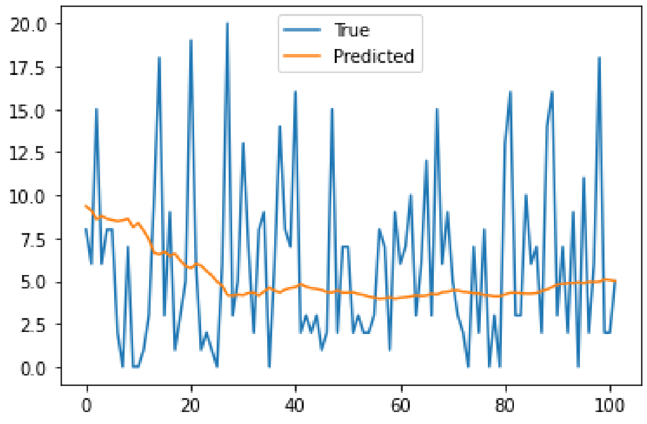


Figure 22: Predicting player performances with SVM model

4.3.0 XGBoost Results

The XGBoost is the model with the lowest error rate, beating the SVM by 1. As figure 23 shows, the model still underfits, just like the SVM, but in terms of the model being too simple to predict the data, this probably is not the case this time. The XGBoost may actually be the model that is the most capable of predicting player performances due to the way each decision tree in the ensemble makes their predictions. As stated in section 3.3.1, the decision tree attempts to find an accurate prediction of the output based on the purity of the input. With this being done 100 times in the ensemble, the prediction tends to be strong when the input data is capable of explaining the target variable.

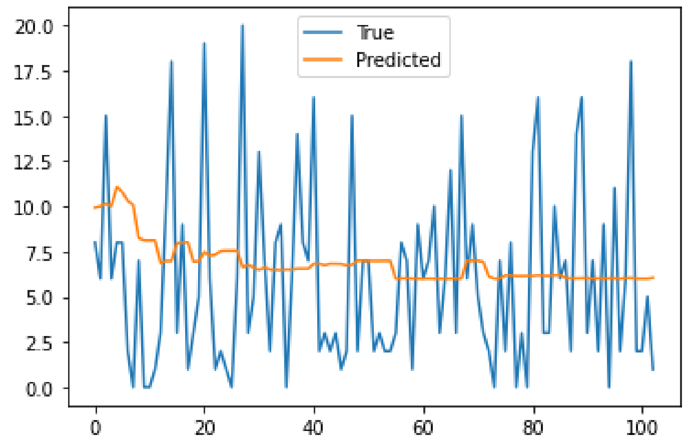


Figure 23: Predicting player performances with the XGBoost model

The model does struggle more to predict the data initially with more noise being added in the first 15-20 predictions, but when enough data is added to the training data the model stabilizes at a point where the player most is the most likely to receive points, but as mentioned above, still underfits with the data.

4.4.0 RNN-LSTM Results

The RNN is the model that struggles the most with too low of a sample size in the beginning of the dataset, with a lot of noise being added to the model at all predictions until about the 30th - 40th prediction, as seen in figure 24. Then, like the rest of the models it stabilizes at the end and underfits with the data.

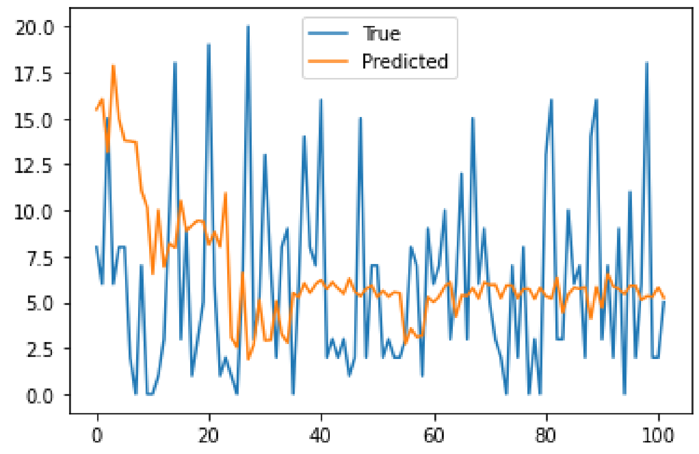


Figure 24: Predicting player performances with Deep RNN model

In terms of the reasons for this model to underfit, the reason is the same as for the XGBoost; due to a lack of explanatory input data the model struggles to accurately predict the target, stabilizing at a point which underfits the model heavily.

Regarding the suitability of the model for a task such as this, it could probably make very accurate predictions given an improved dataset and an optimized model, but it would be a very computationally heavy process, demanding a lot more time than other, less complex models such as the XGBoost. It would probably also struggle to outperform ensemble models such as XGBoost and Random Forest due to their explanatory power, so although the RNN could definitely be considered for tasks such as this, it wouldn't necessarily be the best model to solve the problem and would demand a lot more computational power than other models.

4.5.0 Evaluation and Conclusion

Model	Parameter settings	Error rate (MSE)
ARIMA	p:1, d:0, q:1	35.067
SVM	Epsilon: 0.1	28.047
XGBoost	Default	27.178
RNN-LSTM	See section 3.4.4	37.449

Table 7: Error rate of each model when tested on Mohamed Salah

All models, apart from the ARIMA, quite apparently output a lot of noise in their initial predictions at the start before they stabilize with an expected points distribution that is similar to each other throughout the dataset, which is a sign that the models struggle to predict with low training sets. This shows that for the selected Machine Learning algorithms to predict a player's performance, the models require more data than what they are fed in the beginning of their time steps, which means new players to the Premier League will have to play for at least a full season before the models are capable of making a prediction with less noise.

The ARIMA model doesn't have this problem as the dependencies of the model can be varied by changing the hyper-parameters, but because it struggles with longer time dependencies, it too is not capable of making accurate predictions, and it is quite clear to see that the autoregression of the model looks to have a "lagged overfitting" by predicting a rise in a player's performance similar to that of the week before. The model struggles with the non-stationarity of the dataset and thus it results are not accurate.

5.0.0 Future works

The data used to predict the performances in the research lacks explainability, and this is shown in the performances of the Machine Learning models. In order to improve the quality of the predictive power of the models, different datasets with more explanatory power could strengthen the models. Propriatary data from Sports Analytics companies such as Opta Sports and Statsbomb could be the solution to reduce the error rates and noise from the models. These companies hold much more information about how a player plays in the games with features such as number of touches on the ball, passes made and which direction the passes went, where the player tends to move (heatmap), and much more. This data would then be capable of telling not only what a player manages to do within the set limits of the current dataset, but also what a player attempts to do, and what style of football they play.

Another way of improving upon the current work is by tuning the RNN-model to best predict the data. Testing the accuracy of the model with different number of layers and neurons could improve the performance enough to make the model actually understand what the data is trying to tell it.

Bibliografi

1. ReportLinker. ReportLinker.com. [Online] 01 07 2022. [Cited: 27 07 2022.] https://www.reportlinker.com/p06309369/Fantasy-Sports-Market-Growth-Trends-COVID-19-Impact-and-Forecasts.html?utm_source=GNW.

2. Premier League. PremierLeague.com. [Online] 01 06 2022. [Cited: 27 07 2022.] https://www.premierleague.com/news/2173986.

3. Vig, Himanshu and Deshmukh, Roshan. Allied Market Research. *https://www.alliedmarketresearch.com/fantasy-sports-market-A06468.* [Online] 01 09 2020. [Cited: 23 07 2022.] https://www.alliedmarketresearch.com/fantasy-sports-market-A06468.

4. Truman, Rupert. Splunk.com. *Blog.* [Online] 04 09 2020. [Cited: 08 06 2022.] https://www.splunk.com/en_us/blog/tips-and-tricks/how-to-win-at-fantasy-football-with-splunk-and-machine-learning-part-1.html.

5. —. Splunk.com. *Blog.* [Online] 18 09 2020. [Cited: 08 06 2022.] https://www.splunk.com/en_us/blog/platform/how-to-win-at-fantasy-football-with-splunk-and-machine-learning-part-2.html.

6. T., Dhaneswara Mandrasa. Rstudio.com. *Publications.* [Online] RStudio Publications, 2020. [Cited: 08 06 2022.] https://rstudio-pubs-static.s3.amazonaws.com/577042_d4492e2e511848fb97ef839be077e9b8.html.

7. The Guardian Sport. The Guardian Sport. [Online] The Guardian, 16 12 2019. [Cited: 08 06 2022.] https://www.theguardian.com/sport/2019/dec/16/chess-champion-magnus-carlsen-top-of-world-fantasy-football-rankings-premier-league.

8. Bull, Joshua. YouTube. [Online] Oxford University, 2020. [Cited: 08 06 2022.] https://www.youtube.com/watch?v=LzEuweGrHvc&t=1403s.

9. Joseph, Jim. *Suboptimal Player Transfer Strategy.* s.l. : Stony Brook University, 2021.

10. The English Football Association. fantasy.premierleague.com. *Rules.* [Online] Oracle, 01 07 2021. [Cited: 26 07 2022.] https://fantasy.premierleague.com/help/rules.

11. Ramdas, Delano. Researchgate.net. *Publications.* [Online] [Cited: 08 06 2022.] https://www.researchgate.net/publication/360009648_Using_Convolution_Neural_Networks_to_Predict_the_Performance_of_Footballers_in_the_Fantasy_Premier_League, 2022. DOI: 10.13140/RG.2.2.10010.72645/2.

12. Lindberg, Adrian and Söderberg, David. *Comparison of Machine Learning Approaches Applied to Predicting Football Player Performances.* Gotenburg, Sweden : University of Gotenburg, 2020.

13. Ramdas, Delano. *Using Convolution Neural Networks to Predict the Performance of Footballers.* s.l. : https://www.researchgate.net/publication/360009648_Using_Convolution_Neural_Networks_to_Predict_the_Performance_of_Footballers_in_the_Fantasy_Premier_League, 2022. DOI: 10.13140/RG.2.2.10010.72645/2.

14. GitHub. *vaastav/Fantasy-Premier-League.* [Online] 2022. [Cited: 01 06 2022.] https://github.com/vaastav/Fantasy-Premier-League.

15. Fantasy Premier League. [Online] [Cited: 27 07 2022.] https://fantasy.premierleague.com/help.

16. Moore, Sarah. News Medical. [Online] 28 09 2021. [Cited: 27 07 2022.] https://www.news-medical.net/health/History-of-COVID-19.aspx.

17. Premier League. [Online] 15 06 2020. [Cited: 27 07 2022.] https://www.premierleague.com/news/1682374.

18. Opta Sports. Opta Analytics. [Online] [Cited: 26 07 2022.] https://www.statsperform.com/opta-analytics/.

19. Frost, Jim. Statisticsbyjim.com. [Online] [Cited: 01 08 2022.] https://statisticsbyjim.com/basics/normal-distribution/.

20. Occam, William of. [Online] [Cited: 01 08 2022.] https://en.wikipedia.org/wiki/Occam%27s_razor#Formulations_before_William_of_Ockham.

21. Wordstream.com. [Online] [Cited: 01 08 2022.] https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications.

22. Brooks, Chris. Univariate Time Series Modelling and Forecasting. *Introductory Econometrics for Finance, 3rd Edition.* Cambridge : Cambridge University Press, 2014.

23. Fernando, Jason. Investopedia.com. [Online] 06 04 2022. [Cited: 01 08 2022.] https://www.investopedia.com/terms/a/autoregressive.asp.

24. Wikipedia.org. Wikipedia.org. *Wikipedia.org.* [Online] [Cited: 01 08 2022.] https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average#Differencing.

25. Vapnik, Vladimir and Cortes, Corinna. *Support-Vector Networks.* s.l. : Springer.com, 1995. https://doi.org/10.1007/BF00994018.

26. sci-kit learn. sklearn.org. *Support Vector Machine.* [Online] [Cited: 01 08 2022.] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html.

27. cnblogs.com. *Yang Xiao Ling blog.* [Online] [Cited: 01 08 2022.] https://www.cnblogs.com/yangxiaoling/p/10276424.html.

28. Halton, Clay. Investopedia.com. [Online] 31 07 2022. [Cited: 01 08 2022.] https://www.investopedia.com/terms/w/wisdom-crowds.asp#:~:text=Wisdom%20of%20the%20crowd%20is,and%20innovating%20than%20an%20individual..

29. [Online] [Cited: 01 08 2022.] https://medium.com/@sametgirgin/decision-tree-regression-in-6-steps-with-python-c1564b153b74.

30. Géron, Aurélien. *Hands-On Machine Learning with SciKit Learn, Keras and Tensorflow, 2nd Edition.* Canada : O'Rielly Media Inc., 2019. 978-1-492-03264-9.

31. XGBoost library. XGBoost. *Read The Docs.* [Online] [Cited: 01 08 2022.] https://xgboost.readthedocs.io/en/stable/python/python_api.html.

32. Chen, Tianqi. XGBoost: A Scalable Tree Boosting System. 2016.

33. Eliasy, Ashkan. Research Gate. [Online] 01 07 2020. [Cited: 01 08 2022.] https://www.researchgate.net/figure/The-comparison-between-Recurrent-Neural-Network-RNN-and-Feed-Forward-Neural-Network_fig1_338672883.

34. Olah, Cristopher. Github.com. *Colah's Blog.* [Online] 27 08 2015. [Cited: 01 08 2022.] http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

35. Brownlee, Jason. Machine Learning Mastery. [Online] 03 07 2017. [Cited: 01 08 2022.] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

36. Bohra, Yash. Analytics Vidhya. [Online] 21 06 2021. [Cited: 01 08 2022.] https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/.

37. [Online] [Cited: 01 08 2022.] https://stackoverflow.com/questions/50488427/what-is-the-architecture-behind-the-keras-lstm-cell.

38. Auffarth, Ben. *Machine Learning for Time-Series with Python.* Birmingham : Packt Publishing, 2021. 978-1-80181-962-6.

39. Vikash. [Online] [Cited: 01 08 2022.] https://miro.medium.com/max/1350/1*yE6-RsMd0D2XIGgld_o_KA.jpeg.

40. Splunk.com. *splunk.com/blog.* [Online] 04 09 2020. [Cited: 08 06 2022.] https://www.splunk.com/en_us/blog/tips-and-tricks/how-to-win-at-fantasy-football-with-splunk-and-machine-learning-part-1.html.

41. Truman, Rupert. Splunk.com. *Splunk.com/blog.* [Online] 18 09 2020. [Cited: 08 06 2022.] https://www.splunk.com/en_us/blog/platform/how-to-win-at-fantasy-football-with-splunk-and-machine-learning-part-2.html.

42. T., Dhaneswara Mandrasa. Rstudio Publications. [Online] 2020. [Cited: 08 06 2022.] https://rstudio-pubs-static.s3.amazonaws.com/577042_d4492e2e511848fb97ef839be077e9b8.html.

43. The Guardian Sport. The Guardian Sport. [Online] 16 12 2019. [Cited: 08 06 2022.] https://www.theguardian.com/sport/2019/dec/16/chess-champion-magnus-carlsen-top-of-world-fantasy-football-rankings-premier-league.

44. Bull, Joshua. YouTube.com. *YouTube.com.* [Online] Oxford University, 2020. [Cited: 08 06 2022.] https://www.youtube.com/watch?v=LzEuweGrHvc.

45. Joseph, Jim. *Suboptimal Player Transfer Strategy.* s.l. : Stony Brook University, 2021.

46. fantasy.premierleague.com. fantasy.premierleague.com. [Online] 2022. [Cited: 07 06 2022.] https://fantasy.premierleague.com/entry/642722/history.

47. Anand, Vaastand, Dhillon, Ravgeet and Ilaro, Daniel. Github.com. [Online] 08 06 2022. [Cited: 08 06 2022.] https://github.com/vaastav/Fantasy-Premier-League.

48. Maier, Marc. Github.com. [Online] 24 07 2021. [Cited: 08 06 2022.] https://github.com/177arc/fpl-data.