# Machine Learning - Group 70 - Premier League Match Predictions

## Introduction

The outcomes of football games are notoriously hard to predict, with games won by the finest of margins and the deemed ‘favourite’ never being a sure thing. Given this seemingly unpredictable nature and our interest in the sport, we decided to focus our project on the prediction of match results in the English Premier League 2018/19. We knew that finding statistics on each season on the internet would not be difficult as it is one of the most popular and media-covered competitions in the world. We trained Logistic Regression, Lasso Regression, Ridge Regression, and K-nearest neighbours classifier models, hoping to analyse, and to gauge the performance of each model with predicting matches. We will use our models to predict the outcome of matches through each gameweek, using the previous gameweeks as training data. As the training data grows with each gameweek, we hope to see better accuracy with our predictions. We also wish to investigate which features carry the highest weights, once the model has been trained i.e. which metrics have the greatest impact on the outcome of these games. We will then use a random baseline classifier to compare all of these models against. We used the following features as inputs for each of our models:

- Attendance
- Home win streak
- Home total points
- Home total goals scored
- Home total goals conceded
- Home previous season points
- Home team value
- Away win streak
- Away total points
- Away total goals scored
- Away total goals conceded
- Away previous season points
- Away team value

Each season consists of 38 gameweeks, with 10 matches played each gameweek. These input features were obtained for each match in each gameweek (380 matches), along with the result of the match which was ‘H’ for a home win, ‘A’ for an away win and ‘D’ for a draw.
Next, starting at gameweek 2, we trained our models using the features for all games up to the current gameweek. So for gameweek 2 the input features to predict are all the matches in gameweek 2, and the training data is all the matches in gameweek 1. For gameweek 5, the training data would be all of the matches from gameweek 1 to gameweek 4 and the test data is the matches from gameweek 5, etc. Our models would then output their prediction for each match in the current gameweek.

Initially we planned to compare two seasons; one during the pandemic and one before and investigate any differences in the models trained. We decided to scrap the idea and focus on the analysis of our models predictions for a single season only.

## Dataset and Features

For each season of interest, the dataset generation process begins with the scraping of three web pages for statistics on season fixtures, wage bills and previous season standings. The scraping is performed using a LXML HTML parser, and the data from these web pages is stored in respective CSV files that are generated when the scraping is complete. The largest of these files is the season fixtures with 380 entries of matches across 38 gameweeks, gathered from FBRef [1] containing the date and time of the fixture, the home and away teams with their expected goals, the final score of the game, the attendance, venue and the referee in charge. The next file is the standings of each team in the previous season with 20 entries, gathered from Sky Sports [2] which shows the final standings of the teams playing in the previous season. The last file is the wage bill for each team in the current season also with 20 entries, gathered from spotrac [3] which shows each team along with the number of active players and their expenditure on wages of players in each position.

While our original plan was to build and train a model for each team to hold this information, it was realised that there would be no way to inform each model of the strength of their opponent and it would also likely result in many more lines of code. Ultimately we decided it would be best to design a model which uses a combination of all of the team statistics related to an individual fixture as training data, and use the fixture result as test data.

The aim was to use the gathered statistics to build a profile for each team that somehow captured their performance against other teams throughout the season, their current form and some other important metrics about the club, for example the value of their roster. To construct this profile for each team, the following features were designed; number of goals scored in the current fixture, current win streak, number of season wins, number of season points, number of season goals scored, number of season goals conceded, previous season position and team value. These features are passed to the model for each team playing in a fixture, along with the number of fans in attendance at the match.

These features are generated by iterating through the fixtures CSV file and appending the statistics from the current fixture to the statistics for the home/away team in their last fixture. For example, the number of goals that the home team has scored coming into any fixture is equal to the number of goals that they had scored going into their previous fixture added to the number of goals that they scored in their previous fixture. This required iterating backwards towards the top of the CSV file, until the home team was found as in either the ‘Home’ or ‘Away’ column, and extracting the relevant column values at that row position. When each fixture has been processed, a new CSV is created with the ‘cleaned’ statistics for each season. An example of this processed dataset is shown below for a 2018 fixture between Manchester City F.C. and A.F.C. Bournemouth, where the columns that are used as model features are highlighted in orange.

| Index | WeekNumber | Attendance | FTHomeGoals | HomeTeam | HomeTeamWinStreak | HomeTeamSeasonWinsSoFar | HomeTeamSeasonPointsSoFar |
|---|---|---|---|---|---|---|---|
| 131 | 14 | 54409 | 3 | Manchester City | 5 | 11 | 35 |

| HomeTeamSeasonGoalsScoredSoFar | HomeTeamSeasonGoalsConcededSoFar | HomeTeamPreviousSeasonPos | HomeTeamValue | FTAwayGoals | AwayTeam | AwayTeamWinStreak |
|---|---|---|---|---|---|---|
| 40 | 5 | 1 | 145206000 | 1 | Bournemouth | 0 |

| AwayTeamSeasonWinsSoFar | AwayTeamSeasonPointsSoFar | AwayTeamSeasonGoalsScoredSoFar | AwayTeamSeasonGoalsConcededSoFar | AwayTeamPreviousSeasonPos | AwayTeamValue | MatchResult |
|---|---|---|---|---|---|---|
| 6 | 20 | 22 | 18 | 12 | 17940000 | H |

This also required some processing of the data that was scraped, for example manipulation of the fixture score which is stored as a string, reformatting of attendances and currency values to remove commas and pound symbols, and some consolidation of team names that were sometimes represented differently in the three datasets. Furthermore, some teams are present in a season that were in a lower league in the previous season meaning they have no valid previous season position in the Premier League, and are therefore left with a position of -1. Normalisation of these values before using them as features was considered but ultimately deemed unnecessary as the iterative weighting process would scale the feature values appropriately.

## Methods

We decided to use several different models, in order to compare and contrast the outputs of the models and analyse the differences and strengths of each. Each model received the same parameters as features and was evaluated using the same metrics in order to ensure a fair and

representative comparison. In addition to this all of our models were compared to a dummy classifier which predicted random outcomes, as a baseline.
We carried out hyperparameter tuning for each of our models by using a 5-fold cross-validation on the training data. Cross-validation allows us to estimate how our models will perform when used on data not used during the training of the model. Using cross-validation results in a less biased estimate of the model skill than other methods, such as a simple train/test split. It involves splitting the dataset into groups, taking each group one by one as a hold out data set, the remaining groups as a training data set, fitting a model on the training set and evaluating it on the test set. We chose the hyperparameters for each model which optimised the performance metrics measured when carrying out this cross-validation.

## Logistic Regression

A logistic regression model is used for parameter classification. It models the probability of an observation belonging to a certain class, in our case the classes are the win/loss/draw outcome of a football game. It uses a logistic function to fit a S shaped curve, called Sigmoid, to find a linear separation between two of these classes using training data. The model takes a weighted combination of features $\theta^T x$, with the signs of these weights indicating which class each feature is associated with. These weights are learned by the models using the training data. The decision boundary is 0 which means that $\theta^T x > 0$ is classified as +1, while $\theta^T x < 0$ is classified as -1.
As opposed to linear regression, where the line is fit to the model using the "least squares" method, with logistic regression we use a logarithmic cost function $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} log(1 + e^{-y^{(i)}\theta^T x^{(i)}})$ which penalizes prediction errors and encourages the predictions to be well away from the prediction boundary, employing a standard gradient descent algorithm. Overly complex models are penalised using regularization in order to avoid over-fitting, with this regularization strength determined by the 'C' parameter. This value for 'C' was chosen using cross-validation as outlined above. When working with more than 2 classes, the 'one vs rest' strategy is used, this involves training a classifier per class, with that class set as positive samples and all other classes as negatives.

## K-nearest neighbour

A K-nearest neighbours (KNN) is a classification method that calculates predictions by comparing an observation to the nearest 'k' training examples in the feature space. The observation is then assigned to the class most common among its k nearest neighbors, adjusted for the weighting assigned to each. The 'k' nearest training examples are located using a standard Euclidean distance $d(x^{(i)}, x)$ which gives the distance between feature vector $x^{(i)}$ and $x$ for each training point i. The weighting of each neighbouring point is a function of their distance from the input point so that the nearer neighbors contribute more to the average than the more distant ones. One simple example is to give each neighbor a weight of 1/d, where d is the distance to the neighbor. A more advanced example is to use a Gaussian weighting function $x(i) = e^{-\gamma d(x^{(i)}, x)^2}$. When making a prediction, we get the sign of the weighted combination of the k nearest neighbours, or simply a majority vote when no weights are given. The choice of the size of 'k' can lead to over/under-fitting, our value for 'k' was chosen using cross-validation as outlined above. One downside to the KNN model is that we must search through all training data in order to find the nearest neighbours, which can be computationally intense and slow.

## Lasso/Ridge Regression Models:

Lasso and Ridge regression are forms of regularized linear regressions. The difference between the two models is the regularization penalty which they use. For both models the regularisation strength is determined by the parameter α = 1/2C.

For both models the regularisation strength is determined by the parameter α = 1/2C.
Ridge regression uses an L2 penalty which doesn't result in elimination of coefficients or models with few coefficients. It adds the square of each coefficient to the cost function, which incentivises the model to keep the coefficient values low in order to minimise the cost function. The weight of the added penalty is calculated as the inverse of the hyperparameter 'C', meaning the larger 'C' is, the smaller the penalty will be and the more like a standard linear regression model it will behave. The smaller 'C' is, the larger the penalty will be and the coefficient magnitudes are considerably less compared to the linear regression case. Lasso regression on the other hand uses an L1 penalty which is equal to the absolute value of the magnitude of coefficients. This can result in models with few coefficients as unimportant coefficients become zero and be eliminated from the model. A larger L1 penalty results in coefficient values closer to zero, which is the ideal for producing simpler models. The 'C' values for both the Lasso and Ridge regression models were once again chosen using cross-validation as outlined above.

## Experiments / Results / Discussion

### Random Baseline Classifier

A random classifier was used to compare against the performance of other models, where the predicted outcome of any match is generated uniformly at random. This model is trained with all of the same features as other models, but it does not actually learn from this data and makes predictions regardless of either team's strength. The F1 score and accuracy of the predictions are shown in Table A below for a sample number of gameweeks.

| **Gameweek** | 2 | 3 | 4 | 5 | ... | 34 | 35 | 36 | 37 |
|---|---|---|---|---|---|---|---|---|---|
| **F1 Score** | 0.074 | 0.095 | 0.074 | 0.357 | ... | 0.194 | 0.185 | 0.311 | 0.333 |
| **Accuracy** | 0.4 | 0.4 | 0.4 | 0.6 | ... | 0.467 | 0.467 | 0.6 | 0.6 |

Table A : Random classifier performance metrics

As expected, the random classifier shows no improvement on performance as the season progresses and the training dataset increases, as it consistently picks one of three possible outcomes for a game at random. Although this baseline classifier is not suitable for any machine learning classification application, it is useful for gauging the performance of the other models. However in the context of trying to predict the outcome of football games, the random classifier may have an accuracy close to other models due to the erratic nature of real world sports events.

Using gameweek 1 to 30 as training data, we predicted gameweek 30 to 38 to get a more accurate measure of our performance metrics by using a larger test data set. The precision for the random baseline classifier was 0.3542, the recall was 0.3460, the F1 accuracy was 0.3468 and the accuracy was 0.5864. The accuracy score here is quite high. We can use these figures to compare our models against.

### Logistic Regression

A logistic regression model was then used to predict the outcome of each fixture, having been trained on all fixtures of the season until that point. A number of C values were used in the implementation of the model, ranging from 0.001 to 100, along with the default L2 penalty. The confusion matrix is obtained for the model via the sklearn package, comparing the predicted outcome for a fixture with the actual result from test data, which allows the accuracy and F1 score to be obtained. Plotting these metrics alongside the selection of C values should inform of an optimal C value for use in the model. These plots for a gameweek at the beginning, midway through and at the end of the season are shown below in Table 2.

| Gameweek | 2 | 2 | 2 | 2 | 2 | 2 | 15 | 15 |
|---|---|---|---|---|---|---|---|---|
| C | 0.001 | 0.01 | 0.1 | 1 | 10 | 100 | 0.001 | 0.01 |
| F1 Score | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.296 | 0.296 |
| Accuracy | 0.735 | 0.735 | 0.735 | 0.735 | 0.735 | 0.735 | 0.6 | 0.6 |

| Gameweek | 15 | 15 | 15 | 15 | 37 | 37 | 37 | 37 |
|---|---|---|---|---|---|---|---|---|
| C | 0.1 | 1 | 10 | 100 | 0.001 | 0.01 | 0.1 | 1, 10, 100 |
| F1 Score | 0.296 | 0.296 | 0.296 | 0.296 | 0.432 | 0.432 | 0.432 | 0.432 |
| Accuracy | 0.6 | 0.6 | 0.6 | 0.6 | 0.735 | 0.735 | 0.735 | 0.735 |

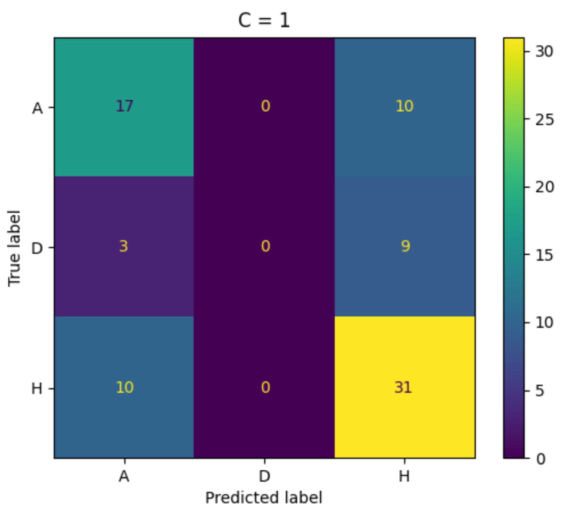
Table 2: Logistic regression model performance metrics

Table 2 allows two conclusions to be drawn about the Logistic Regression model;

1. The model does not seem to be 'learning', in that the predictions do not improve over time as the model is trained with more data. This is likely due to the stochastic nature of the training data, meaning that the multinomial logistic regression model struggles with making accurate predictions on the test data.
2. Changing values of C does not seem to have any impact on the performance of the model. This means that the L2 penalty term has little impact on the predictions made by the model, regardless of its magnitude.

From these conclusions, it can be said that a logistic regression model is not a suitable choice for this application, as it does not respond well to training and does not have a reasonable or consistent performance.

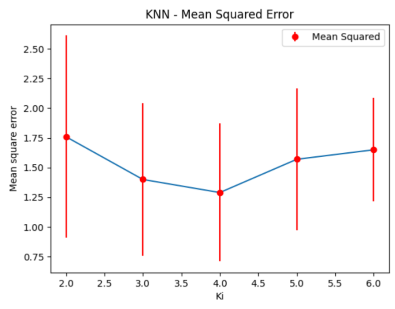
**Gameweek 30 to 38**



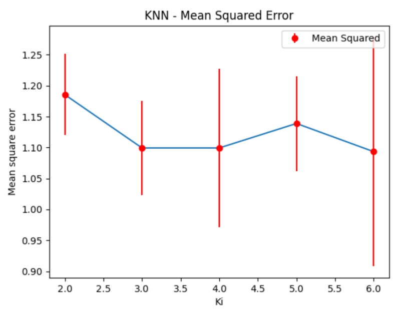
## K-Nearest Neighbours

KNN is an algorithm which attempts to predict the right class by measuring distance between test data points and training data points. It assumes similar things exist close to one another. For our KNN model, we performed 5-fold cross validation to determine the optimum K value to use for the number of neighbours. For lower values of K, we expect some overtfitting to occur, with not enough neighbours being used. Likewise for higher values of, we expect underfitting to occur. We plotted the mean and standard deviation of the prediction error against K to help us determine an optimal K value. The graphs for Gameweek 2, 15 and 37 are included below. For the earliest gameweek the error is higher than the two later ones at around 1.6, however it seems that for gameweek 15 it is around 1.1 which is lower than the latest gameweek which is around 1.2. This is slightly unexpected as with more training data you would expect the model to perform better, however the test data stays the same size, so seeing impactful results may prove difficult. For the latest test dataset gameweek 37, We can see underfitting occurring for values of K below 4. Overfitting can be seen then with values of K above 6 it seems. The mean squared error is not good to begin with, but we can optimise it by choosing a value for K of 5 between 4 and 6 based on the reasons above.

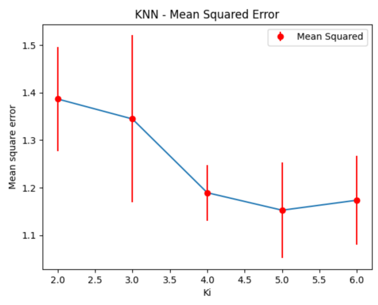
**Gameweek 2**



**Gameweek 15**

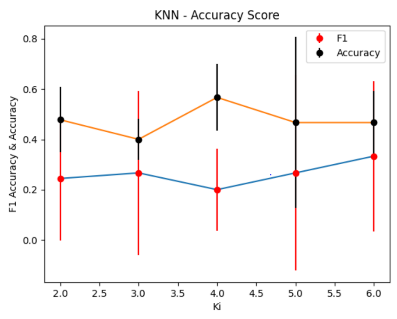


**Gameweek 37**

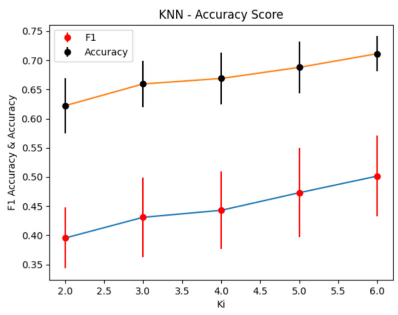


We then analysed the performance of our model by calculating the accuracy and F1 scores before plotting them. This was done by first determining the 3x3 confusion matrix from our predictions compared to the actual results in the gameweek. We can see in these graphs, especially for gameweek 2, the values of K that are even tend to have higher performance scores due to the split vote problem which causes some data points not to contribute to the model. It's clear that the value of K must be higher than 3 and no greater than 6 to avoid overfitting and underfitting. This backs up with our optimal K value of 5 obtained using the mean squared error method.

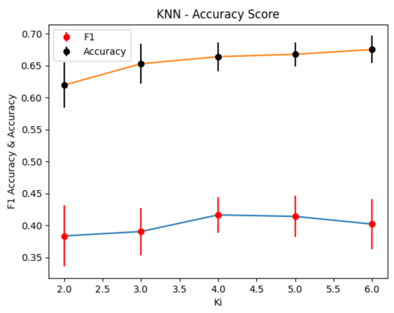
**Gameweek 2**



**Gameweek 15**

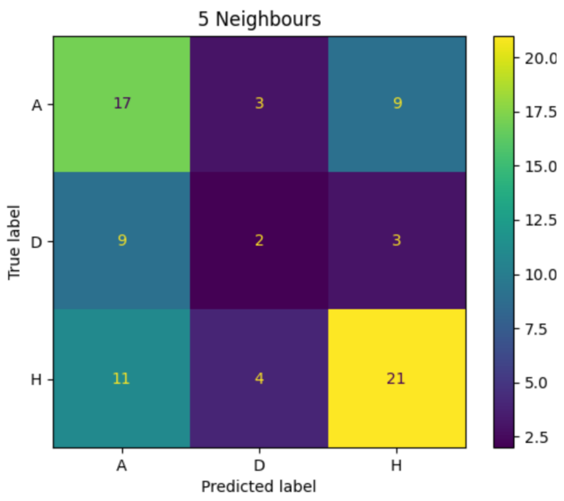


**Gameweek 37**



We decided that determining performance metrics on such small test datasets may not be accurate, as some gameweeks may have lots of unexpected results compared to others. To test the model more accurately, we decided to use gameweek 1 to 30 as training data, and use gameweek 30 to 38 as test data to obtain performance values. This resulted in a more accurate value of 0.4393 for the precision and 0.4374 for the recall. The precision and recall respectively scores are around 0.08 and 0.09 higher than the baseline classifiers precision and recall scores. The accuracy score was respectably 0.6709 and the F1 accuracy was 0.4325. Both are around 0.1 higher than that of the random classifier. These comparisons are positive as all performance scores have increased. The model is not performing astoundingly, but it is learning nonetheless. The confusion matrix used for calculating the performance using the optimal value of 5 for K can be seen below.

**Gameweek 30 to 38**

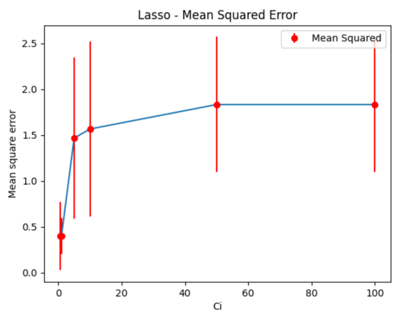


Based on these results, we can see that the KNN classifier does perform better than the baseline classifier, although not by a huge amount. It also performs a lot better than the logistic model as the predictions change with different K values. This is the best performing model so far.

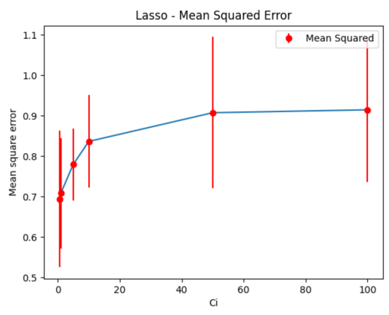
## Lasso Regression

For the Lasso Regression model, we performed 5-fold cross validation to determine the optimum C value which specifies how much of a regularization penalty we wish to use. We can expect that for lower values of C, less of the coefficient features will be affected. Looking at the mean and standard deviation of the prediction error against C plots will help us determine our optimum C value. For the earliest gameweek the error is higher than the two later ones at around 1.5 to 1.8. For gameweek 15 it is around 0.8 to 0.9. The latest gameweek has the lowest mean squared error which makes sense as it has the most training data to learn from. We will use the latest test week to pick a good value for C. The error rate seems to rise most weeks with an increase in C, as more coefficient features are affected. Some overfitting seems to be occurring in the latest gameweek with values of C higher than 50. For values close to 0 underfitting seems to occur. Based on our analysis, picking a C value between 10 and 30 may be optimal

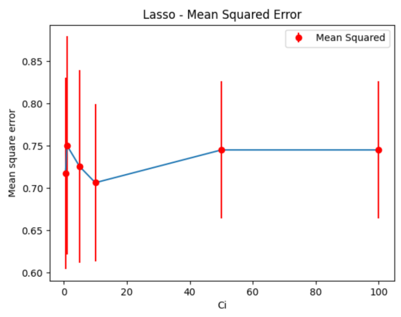
**Gameweek 2**



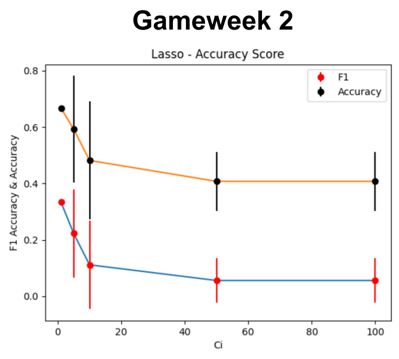
**Gameweek 15**

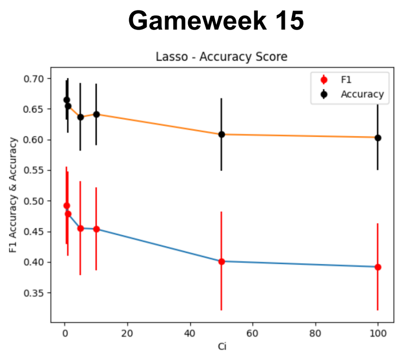


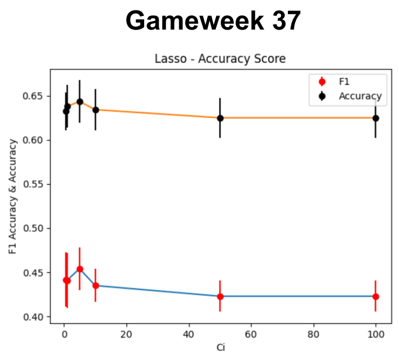
**Gameweek 37**



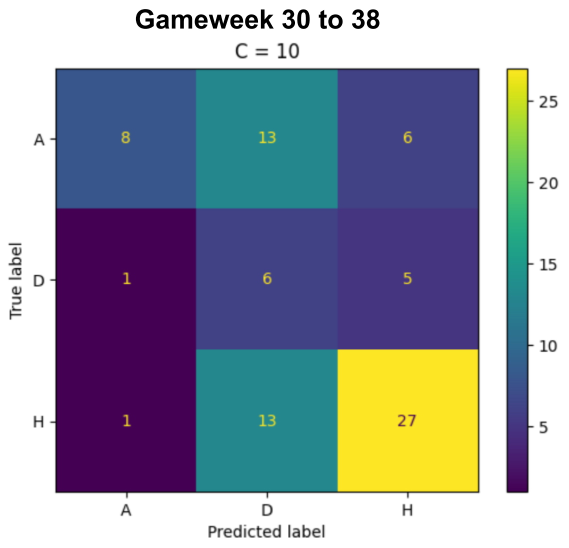
Analysing the performance of the model at each gameweek will further help us to determine an optimal C value. We will look at the accuracy and F1 score plots of the model. We can see that for the later gameweeks, both the F1 accuracy and accuracy scores are higher, as the model has more training data at later weeks. However, both performance scores seem to lower with sharper increases. This leads us to believe we do not want to choose a high value for C above say 40. We also don't want to underfit the model however and choose too low a value for C, so taking the mean squared error optimal C value range into account (10 to 30), we can choose a value of around 10. The model seems to perform best around this value for both cross validation techniques.



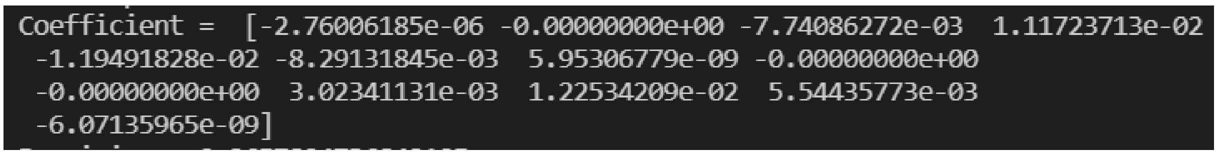




Next we determined the performance metrics on a larger test data set, in order to test our model more accurately. Using the same method as for KNN, we used gameweek 1 to 30 as training data, and used gameweek 30 to 38 as test data to obtain performance values. We had a value of 0.2658 for the precision and 0.3444 for the recall. The precision score was about 0.1 higher than the baseline classifiers precision indicating this model is not very good in determining relative data points. The recall score is around the same, which is again not a good sign. The F1 accuracy score was very low at 0.2852, lower than the random baseline classifiers score by around 0.15. This is as a result of the low precision and recall scores. The accuracy was around 0.5500 which is also below the random classifiers score. These comparisons are not hopeful as all performance scores have decreased. This regression model does not seem to suit this topic. The confusion matrix used for calculating the performance using the optimal value of 10 for C can be seen below.



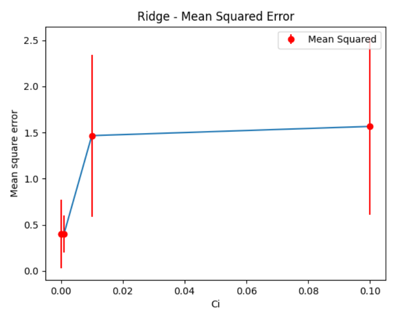
Based on these results, we can see that the Lasso model underperforms compared to the baseline classifier. It also significantly underperforms compared to the KNN model. This is the worst performing model used. This is because L1 regularization can reduce the coefficient values of some features to 0, meaning they are not being used. The model ended up 3 features to 0. These features were home win streak, away win streak and away total points.


```
Coefficient =  [-2.76006185e-06 -0.00000000e+00 -7.74086272e-03  1.11723713e-02
 -1.19491828e-02 -8.29131845e-03  5.95306779e-09 -0.00000000e+00
 -0.00000000e+00  3.02341131e-03  1.22534209e-02  5.54435773e-03
 -6.07135965e-09]
```

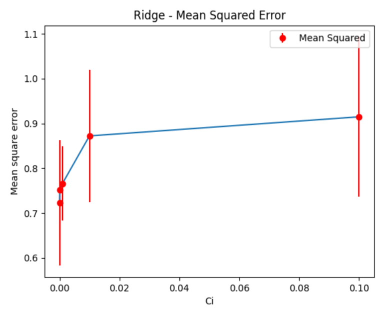
## Ridge Regression

For the Ridge Regression model, we again performed 5-fold cross validation to determine the optimum C value which specifies how much of a regularization penalty we wish to use. The weight of the added penalty is calculated as the inverse of the hyperparameter ‘C’, meaning the larger ‘C’ is, the smaller the penalty will be and the more like a standard linear regression model it will behave. Again, looking at the mean and standard deviation of the prediction error against C plots will help us determine our optimum C value. For the earliest gameweek the varies much more than the two later ones from around 0.4 to 1.5. This variance can be attributed to the lack of training data, and the low error can be disregarded. For gameweek 15 it is around 0.72 to 0.9. The latest gameweek has the lowest reliable mean squared error(.7) which makes sense as it has the most training data to learn from. We will use the latest test week to pick a good value for C. The error rate seems to rise most weeks with an increase in C, as our model begins to behave more like a standard linear regression model . For our lower values of C however there is an increase in standard deviation of the error as the coefficient magnitudes are considerably less and underfitting seems to occur. Based on our analysis, picking a C value between .001 and .01 may be optimal.

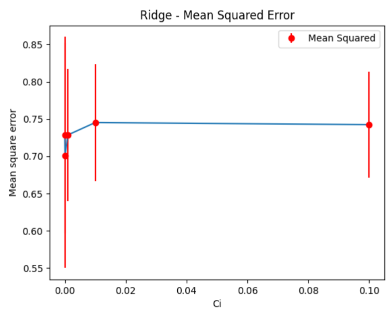
**Gameweek 2**



**Gameweek 15**

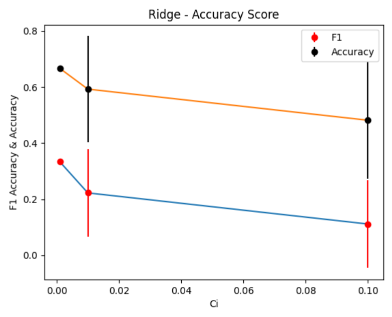


**Gameweek 37**

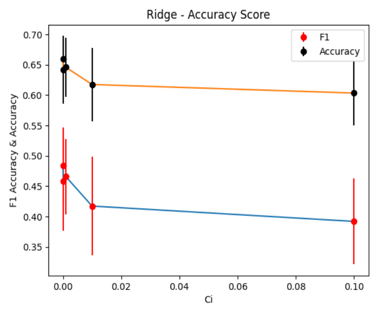


As we did with our Lasso Regression model; analysing the performance of the model at each gameweek will further help us to determine an optimal C value. We will look at the accuracy and F1 score plots of the model. Again we can see that for the later gameweeks, both the F1 accuracy and accuracy scores are slightly higher, as the model has more training data at later weeks. Performance can be seen to peak at about C=.001 consistently through the gameweeks. With over and under-fitting seen to be occurring on either side of this point. The model seems to perform best around this value for judging by both F1 score & accuracy metrics.

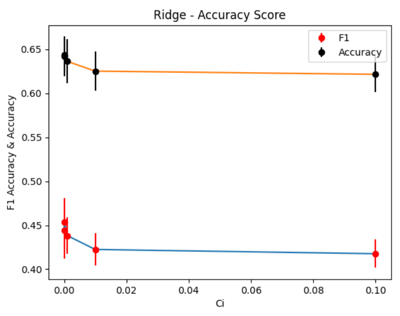
**Gameweek 2**



**Gameweek 15**

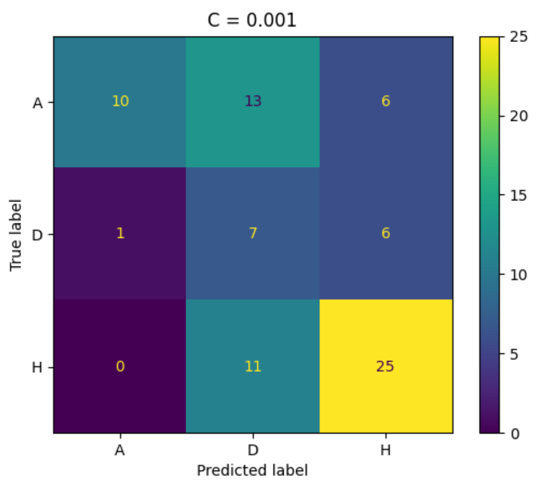


**Gameweek 37**



Again, as we did with our Lasso Regression model; we determined the performance metrics on a larger test data set, in order to test our model more accurately. Using the same method as for KNN, we used gameweek 1 to 30 as training data, and used gameweek 30 to 38 as test data to obtain performance values. Our Ridge Regression model had a value of 0.3905 for the precision and 0.3569 for the recall. The precision and recall scores are around the same as the baseline classifiers precision indicating this model is not very good in determining relative data points. Similarly to our Lasso model, the F1 accuracy score was very low at 0.3348, lower than the random baseline classifiers score by around 0.1. This is as a result of the low precision and recall scores. The accuracy was around 0.5696 which is also below the random classifiers score. The confusion matrix used for calculating the performance using the optimal value of .001 for C can be seen below.

**Gameweek 30 to 38**



By all of the used metrics, this model performed worse than our baseline random classifier and so it is safe to say that this regression model does not work with this data. It also significantly underperforms compared to the KNN model. This model performed similarly to our Lasso Regression model, our worst performing model.

## Summary

In this project, the problem of predicting the outcome of Premier League football games was tackled. In tackling this problem, multiple supervised machine learning algorithms were built and assessed. The model which we found to work the best was the K-Nearest Neighbours model, which outperformed our baseline and all other models in all of our chosen performance metrics. KNN's use of feature similarity is probably why it performs better than the other algorithms. Using previous matchups with teams of similar strength to predict match results is a logical and seemingly effective technique.
Variants of linear models such as Ridge and Lasso regression that work with feature vectors consisting of different statistics about the teams and season were found to have a worse performance than even our baseline random classifier, with our Logistic regression model performing to a similar standard.

We can conclude that this problem is inherently complex, with football games being won by such fine margins. There is a touch of randomness to the outcomes, and a plethora of possible input features adds to this complexity. Our resulting algorithms are unlikely to be of much use for real world applications due to low performance, however the idea of exploring the viability of deep learning approaches on a much larger dataset in the future is exciting.

## Contributions

There were several stages involved in the creation of this project. The first step of analysing the assignment specification and coming up with a viable project idea took longer than expected, involving several long meetings to decide what to do and how to go about it.
We all kept in touch throughout the project with regular meetings and communication through our group chat. We divided up the work accordingly at each stage and used our github repo(linked below) for version control and collaboration of our code.

Below is a summary of which tasks were assigned to and carried out by each team member:

Killian KR

**Code:**

The primary focus I had was on code maintenance for the project.

- Setting up of project and code base.
- webReader function within the scraper, contributing to the data collection. This parsed tables from the HTML of the passed web address. We used this to generate our uncleaned CSV data files.
- Set up the random baseline, KNN and Logistic classifiers.
- Determined performance scores for each model. The two functions calculate3x3Metrics and calculate2x2Metrics both take a confusion matrix and return or print the precision, recall, accuracy and F1 score of the predictions. The confusion matrix associated with each gameweek had to be obtained with 3 possible results being ‘A’, ‘D’ and ‘H’.
- K-fold cross validation for each model, which included the graphing of the mean squared error and accuracy score used for choosing optimal C or K values.

**Report:**

- Contributed to the introduction along with Thomas.
- Wrote the analysis for K-Nearest Neighbours and Lasso Regression in the “Experiments / Results / Discussion” section.

Thomas Thomas Dixon

**Code:**

- Lasso+Ridge Confusion Matrices
- Lasso+Ridge Performance Functions
- Added C-range for ridge and Lasso
- F1 Score Graphs

**Report:**

- Wrote the Introduction w/ Killian
- Wrote the methods section
- Wrote the Ridge Regression section
- Wrote the summary section

Dylan DS

**Code:**

- Wrote the code for the data cleaner and made some modifications to Killian’s web reader function and how it was called, which gave us the dataset to use in this application.
- Wrote the code for obtaining and plotting the weights of the features of various models

**Report:**

- Wrote the Dataset and Features section
- Wrote the Random Classifier baseline and Logistic Regression sections of results

## References

[1] FBref.com. 2021. *Football Statistics and History | FBref.com*. [online] Available at: <https://fbref.com/en/>.

[2] Sky Sports. 2021. *Sky Sports - Sports News, Transfers, Scores | Watch Live Sport*. [online] Available at: <https://www.skysports.com/roi>.

[3] Sports Contracts, Salaries, Caps, Bonuses, & Transactions. 2021. *Spotrac.com*. [online] Available at: <https://www.spotrac.com/>.

## GitHub

https://github.com/killianronan/Premier-League-ML-Analysis