Football goal prediction with Linear Regression and Football match classification with K-Nearest Neighbors

Abstract

Traditionally, the outcome of professional football matches is predicted using the number of goals scored or conceded. However, the predictions are usually incorrect as there are many different elements that affect a team's performance and the result of a football game. The main objective of this project is to use data from the datasets to predict the full-time goals and determine whether the match is interesting or not. We can achieve that using different techniques to analyze statistics from the datasets and come up with a model prediction, then cross-validate it with the real values. In this project, we used PCA and some strong indicators to build a regression model for predicting the goals. For classification, we applied K-nearest neighbor algorithm to determine a match is interesting or not. Our results for goals prediction were not useful, while our results for match classification were acceptable.

1. Introduction

A particularly important element of Data Science in football is the ability to evaluate a team's performance in games and use that information to attempt to predict the result of future games based on this data. In this project, our biggest question is how to maximize the accuracy of our goals forecast and also label if a match is interesting or not.

Outcomes from football matches can be difficult to predict, as each game is different from another. Additionally, there are a lot of random elements that can affect the game scores. One way to make the predictions more accurate is to dive deeper into match using in-game statistics.

Previously, betting agents used only the number of goals scored of conceded to predict a future game's outcome. Now, with all the statistics at hand, we can leverage them to create an "expected goals" metrics or match classification.

The main approach we will take is to build a model for expected goal statistics in order to generate better predictions in the future.

There are many challenges we have to face to achieve the tasks mentioned. In order to design our models and classification, we will need to have a thorough understanding of prediction or classification techniques used. Then, we will have to come up with different experiments and different models to compare in order to find out which one is the best. This would require using quick and easy approaches to answer our questions (Herbinet, 2018).

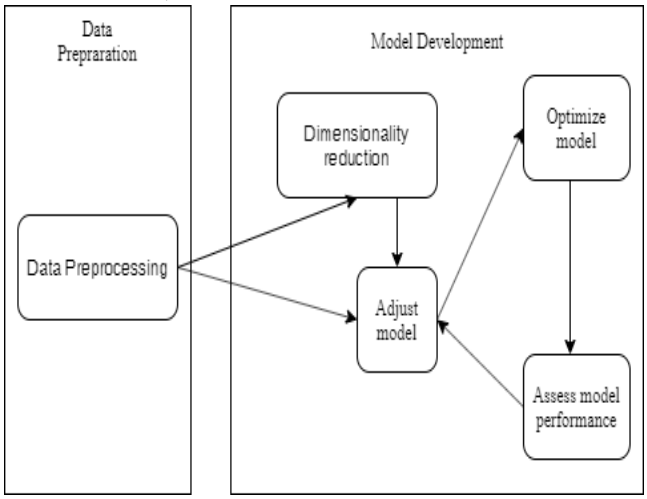

Figure 1. Project workflow.

2. Data Analysis

2.1. Data preprocessing

Datasets used:

Results of The Premier League in seasons 2016, 2017 and 2018 distributed by Football-Data.co.uk. There are a total of 1140 matches, and 15 features from each match are collected. The samples are split into train set (798 matches) and test set (342 matches).

We first imported all 4 files, which contained no NaN. We then concatenated them, resulting in df_train (train_x + train_y) and df_test (test_x + test_y). These two will then be concatenated into df_full_full. df_train will be used to predict full time goals and classification in df_test.

	HomeTeam	AwayTeam	HTHG	HTAG	HTR	HS	AS	HST	AST	HF	AF	HC	AC	Interest	FTG
0	19	8	2	1	1	18	6	7	2	8	10	5	4	1	5
1	24	16	2	0	1	12	8	6	3	12	13	5	1	0	3
2	3	12	0	0	0	13	8	4	4	8	8	0	11	1	4
3	2	20	0	0	0	7	6	2	2	10	10	2	6	0	0
4	24	9	0	1	-1	14	11	6	3	13	16	4	5	0	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1135	16	6	0	0	0	14	12	4	3	14	8	6	3	0	0
1136	5	2	0	1	-1	11	14	4	9	9	11	7	8	0	6
1137	24	0	0	1	-1	21	13	11	8	8	8	7	4	0	1
1138	9	0	0	0	0	8	13	2	6	9	13	2	1	1	2
1139	0	2	2	0	1	5	14	2	2	10	16	1	13	0	3

1140 rows × 15 columns

Figure 2. df_full_full.

Some interesting numbers during data exploration:

For data exploration, we use the full set df_full_full to explore the dataset and find out some interesting numbers:

- Average goals per match = 2.7
- Average half time goal = 1.197368
- Average shots per game: 25
- Half time results: Home Team Wins: 385
    - Draw: 463
    - Home Team Loses: 292
- 28584 shots in total, of which 15848 are Home's,12736 are Away's.
- Away teams make slightly more fouls than the home team (555>491)
- Home Team has more shots in 57.5% of the matches.
- Total shots on target: 9785, which accounts for 34% of total shots.
- The average full-time goals for all of the interesting match: 3.52
- Out of 1140 matches, 481 are interesting, which is 42% of the matches.
- Of all the matches that Home Team has more shots (or equal) (721 matches), 593 of them results in the Home Team has more (or equal) shots on target, which is 82%.

Some small conclusions:

- The Home Team has a slight edge over the Away Team, having 10% more shots. This leads to having better stats on other elements too (Shots on Target, Corners, Half Time Goals, etc.). Ultimately, the home team has an 8% better chance at winning.
- More goals are scored during the second half.
- Out of 10 matches, there are only 4 interesting matches.

2.2. *Correlation*

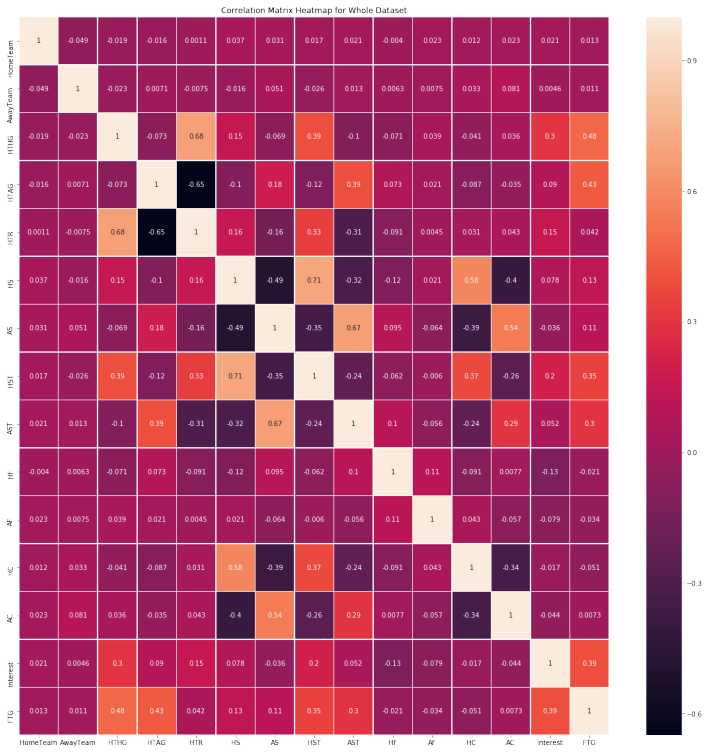

Figure 3. Heatmap for the whole dataset.

Most correlated features:

- HS and HST
- AS and AST
- HS and HC
- AS and AC
- HTHG and HTR
- HTHG and FTG

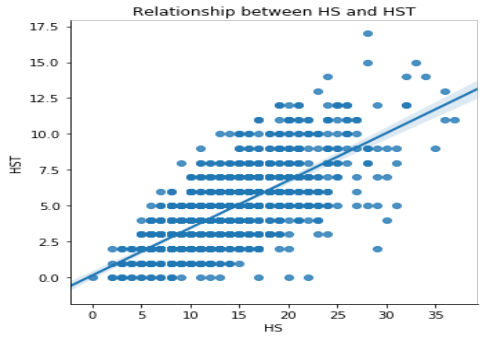

Figure 4. Relationship between HS and HST

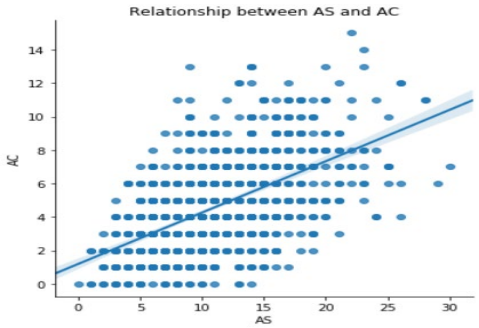

Figure 5. Relationship between AS and AC

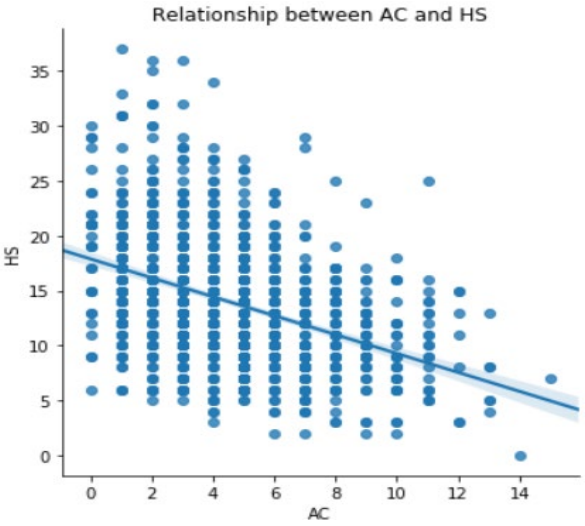

Figure 6. Relationship between AC and HS.

Inference:

It is easy to see that HS/AS has an impact on a lot of other features in the dataset. For example, if a match has a high HS, other features regarding the Home Team will probably have high statistics too. Inversely, if the Home Team has a high number of shots, the Away Team's figures will suffer a decrease in numbers and vice versa. This is easily seen in a real football match as if a team is controlling the pace of the game, its number of shots will be higher, hence all of the other features' values will be higher too.

2.3. *Class distribution and dimensionality*

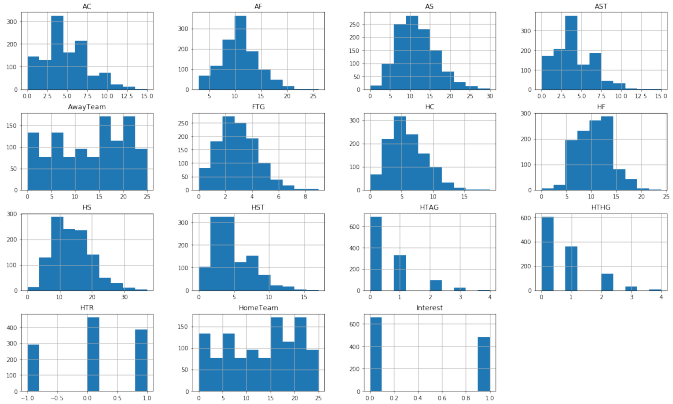

Figure 7. Class distribution of the whole dataset

The dataset has 15 features in total, 2 of which ('FTG', 'Interest') are labels for machine learning tasks. Since there are quite a few dimensions, with all of them being on different scales, it is predicted that the classifier will perform badly if dimensionality is not reduced or if data is not scaled. Across all features, it can be seen that the data is neither normally nor uniformly distributed.

The 'Interest' attribute will be used for classification task. The 0 class (not interesting) is a majority class, while the 1 class (interesting) is a minority class. According to conventional wisdom, and Weiss & Provost (2001), classifiers usually perform worse on the minority class mainly because the minority class contains fewer samples than does the majority class.

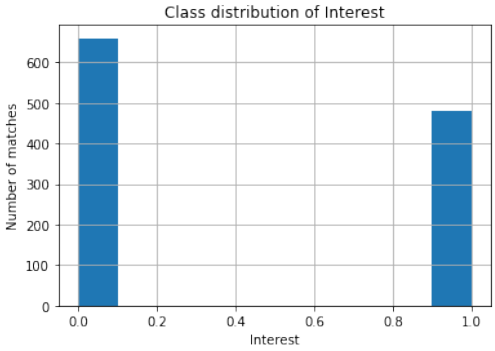

Figure 8. Class distribution of 'Interest' attribute.

2.4. *Principal Component Analysis*

After preprocessing (standardization or normalization), PCA was performed on the data. PCA is a linear transformation that aims to reduce the dimensionality of the data. Each principal component is a linear combination of the original features of the data, and the weights of each term in the linear combination is a component of the corresponding eigenvector, whose corresponding eigenvalues are determined by the covariance matrix of the original data. The projected PCA data has no correlation; the variance is maximized; the construction error is also minimized (by Pythagoras' theorem).

For any task that required data to be transformed with PCA, the train data was first fitted and transformed with a PCA object. After that, the test data was transformed using the same PCA object. By doing this, we ensured that both the train and test data were transformed in the same way. Below are plots describing information about that PCA object which was fitted on the train set.

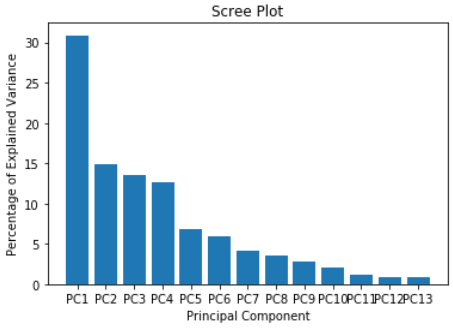

Figure 9. Percentage of Explained Variance for each principal component.

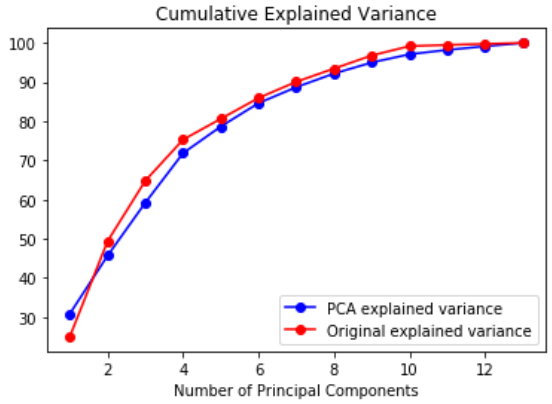

Figure 10. Cumulative Explained Variance of principal components and original variables

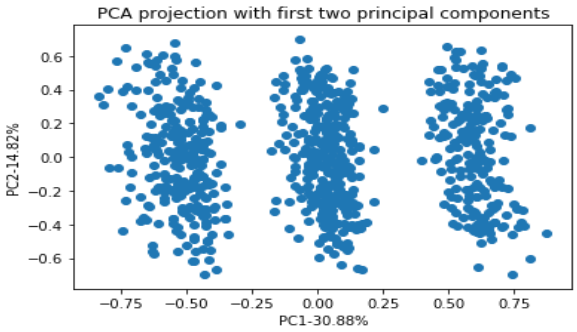

Figure 11. PCA projection of train data using the first 2 principal components.

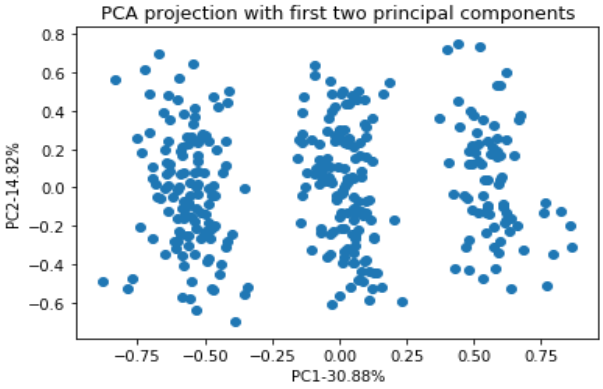

Figure 12. PCA projection of test data using the first 2 principal components.

It took the first 4 principal components to account for 71.86% of the data, with the first component accounting for only 30.88% of the data.

3. Methods

3.1. Regression

Linear Regression is a simple and basic technique for predictive analysis. It uses a set of variables which are significant predictors to impact the outcome variable. These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. Linear Regression is a good approach to predict goals. Thanks to its simplicity, we can compare different methods and experiments easily.

In this case, we can easily do Linear Regression with the help of sklearn.linear_model.Regression(). We first fit the regression model with data from the train set, then predict the full-time goals for test set. We can then cross-validate the predicted values with the true values and calculate accuracy.

For the first experiment, we picked out each of the strongest indicators to see its correlation to full time goals. From that, we plotted the predictions of the test set using trained data from that indicator and the full-time goals.

We again applied the same method for the second experiment, but this time using multiple strongest indicators to train the regression model. We expected that this will yield a more accurate result compared to the first experiment.

For the third experiment, we adopted the projected data from PCA processing earlier to train the regressor. Because the projected PCA has 13 components, we developed a method to pick out a combination of components for which the predictions had the least mean squared error. We then trained the model with that combination to yield the results.

3.2. Classification

For classification, k-nearest neighbor (KNN) algorithm was used. This method classifies a fresh dataset based on the training data. KNN achieves this by computing the Euclidean between each of the new data points in the new set and the points in the training set. Each point in the new data is classified based on the number (k) of its closest points in the training set. The Euclidean distance between two data vectors is calculated as follows:

$$D(x, y) = \|\boldsymbol{x} - \boldsymbol{y}\| = \sum_{i=1}^{n} (x_i - y_i)^2$$

where x, y are the two data vectors, and n is the number of dimensions.

If an unclassified vector data is equidistant from an even number of classified vectors, either a random class is assigned the new vector or k needs to be changed. KNN was chosen because of its interpretability.

Before PCA and/or applying KNN, the data were preprocessed. One method was data normalization with sklearn.preprocessing.minmaxScaler(). This scaling method (the Min-Max scaler) is given by

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}}$$
$$X_{scaled} = X_{std}(max - min) + min$$

where min, max = feature_range. (scikit-learn)

The transformation is calculated as:

$$X_{scaled} = scale \cdot X + min - X_{min} \cdot scale$$

where $scale = \frac{max-min}{X_{max}-X_{min}}$.

Another preprocessing method which was used alongside the previous one to compare performance was sklearn.preprocessing.StandardScaler(), equivalent to preprocessing.scale(). This method (the standard scaler) standardizes features by removing the mean and scaling to unit variance. In other words, the standardized features have zero mean and standard deviation of 1.

The standard score of a sample x is calculated as:

$$z = \frac{x - \mu}{\sigma}$$

where μ is the mean of the training samples or zero if with_mean=False, and σ is the standard deviation of the training samples or one if with_std=False (Pedregosa et al., 2011).

After preprocessing and PCA, we plotted the newly projected data using the first two principal components. After that, a new KNN model was trained on the PCA-projected train set. Another model was trained on the original train set to compare performance. The model was used to classify whether a football match is interesting (labelled 1) or not (labelled 0). The k parameter was optimized by the accuracy score of each k from 1 to 50 given the number of features used. The accuracy was given as a ratio between the number of correct predictions and the total number of matches. The number of principal components used (pc_count) was optimized based on the optimized number of nearest neighbors. In other words, among the accuracy scores given by the optimal k in each case of using a number of features, the case with the highest accuracy score was considered the best. This method of determining the best combination of principal components to produce the best result is similar to the method used in the regression part. If the criterion was mean error instead of accuracy, the lowest mean error was considered the best. Mean error was given by the ratio between the number of incorrect predictions and the total number of matches. It can be seen that mean error equaled 1 minus accuracy. Therefore, only accuracy scores were considered in the results of the experiments.

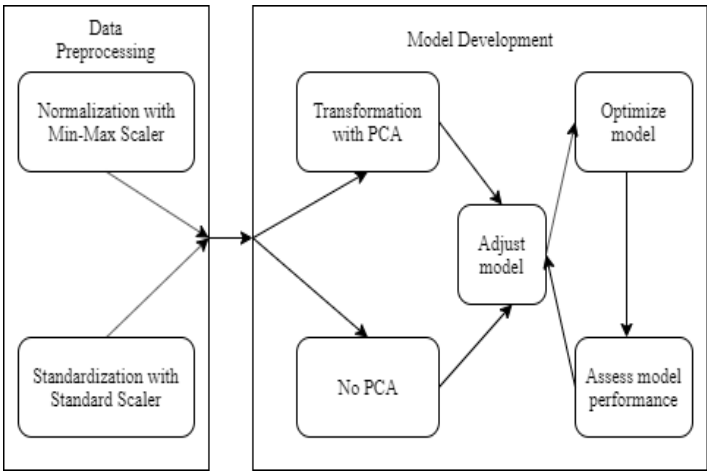

Figure 13. Project network chart

4. Experiments and Results

4.1. Regression

The strongest indicators for full time goals used in these experiments are shown below.

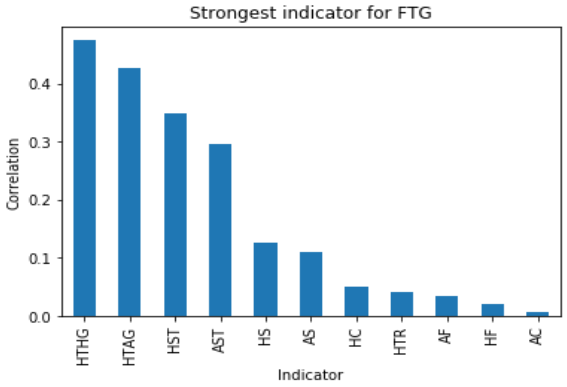

Figure 14. Strongest indicators for FTG.

The plots in our first experiment has an accurate rate and mean squared error ranging from 23-26%, 2.7-2.9, respectively. Both the accuracy and mean squared error are far from ideal.

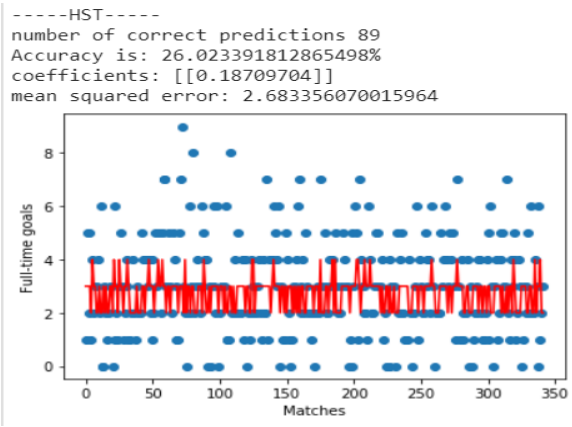

Figure 15. Using HST to predict FTG.

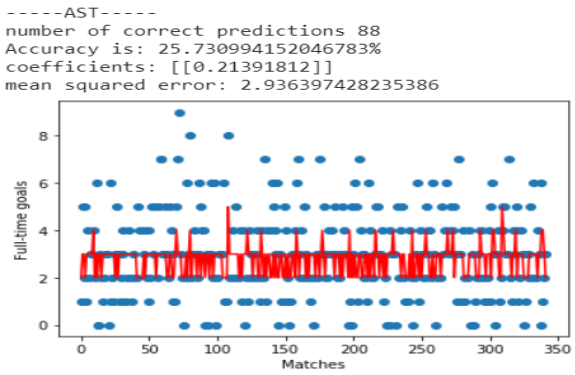

Figure 16. Using AST to predict FTG.

The plot from the second experiment showed a slight increase in accuracy as the precision was 33.33% and mean squared error was 1.485. This was obtained using the strongest 7 indicators from Figure 10.

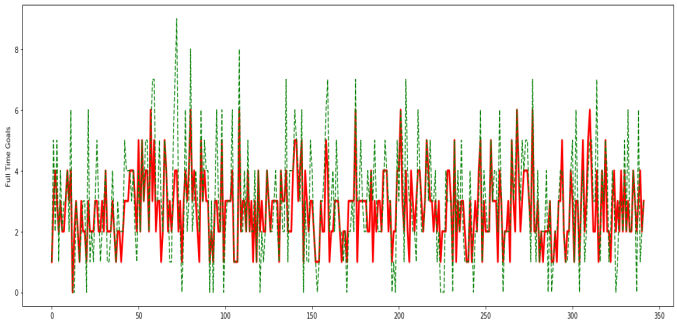

Figure 17. Using multiple predictors to predict FTG.

The third experiment used all of the PCA components, which gave somewhat disappointing results with just 32.45% accuracy and 1.494 mean squared error.

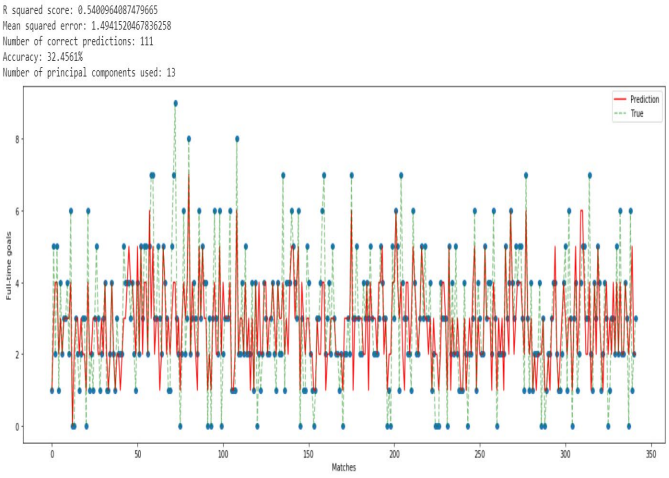

Figure 18. Adopting PCA to build Linear Regression for FTG predictions.

Mean squared error for each number of components used are displayed in the plot below with the lowest mean squared error being 1.494. From this plot we learned that just by using 5 principal components we could obtain the results similar to that of using 13 principal components.

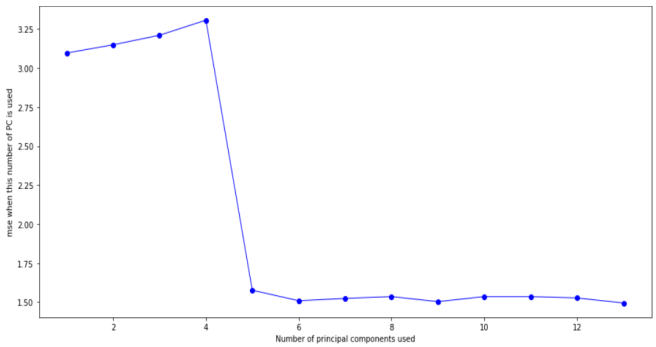

Figure 19. Mean squared error for each number of principal components used.

4.2. Classification

4.2.1. Min-Max scaler

The method used to preprocess the data prior to applying PCA had a noticeable effect on the accuracy of the KNN model. When the Min-Max scaler was used, the best accuracy was 75.44% by using the first principal component and 8 nearest neighbors. There was a noticeable decrease in accuracy between pc_count = 2 and pc_count = 3.

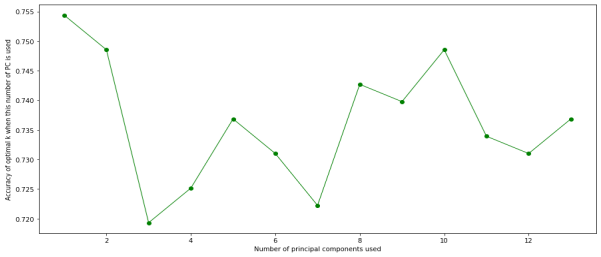

Figure 20. How accuracy of each optimal k changed when the number of principal components used increased, with Min-Max scaler.

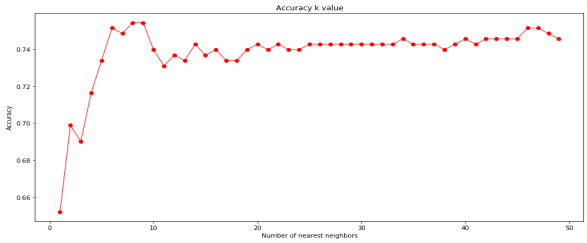

Figure 21. How accuracy changed as k increased when pc_count = 1.

	precision	recall	f1-score	support
0	0.75	0.86	0.80	194
1	0.77	0.62	0.69	148
accuracy			0.75	342
macro avg	0.76	0.74	0.74	342
weighted avg	0.76	0.75	0.75	342

Figure 22. Classification report when k=8 and pc_count = 1.

There were 166 correctly classified uninteresting matches and 92 correctly classified interesting matches.

On the other hand, 28 uninteresting matches were erroneously classified as interesting, and 56 interesting matches were wrongly identified as uninteresting.

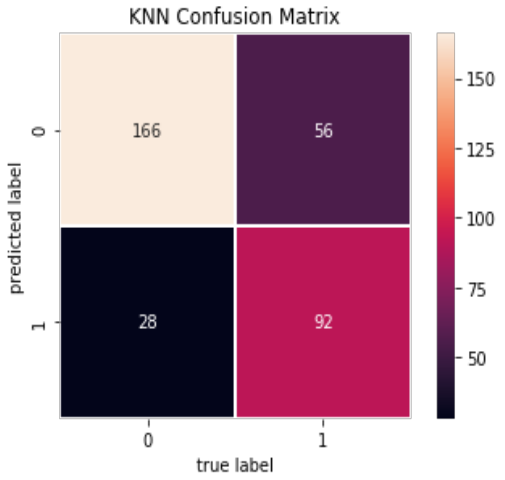

Figure 23. Confusion Matrix when k=8 and pc_count=1.

In each combination of principal components, the accuracy increased sharply when k < 10. When k > 10, the accuracy varied, but there was no noticeable change.

4.2.2. Standard scaler

When the standard scaler was used, the best accuracy was 75.44% by using the first 3 principal components and 33 nearest neighbors.

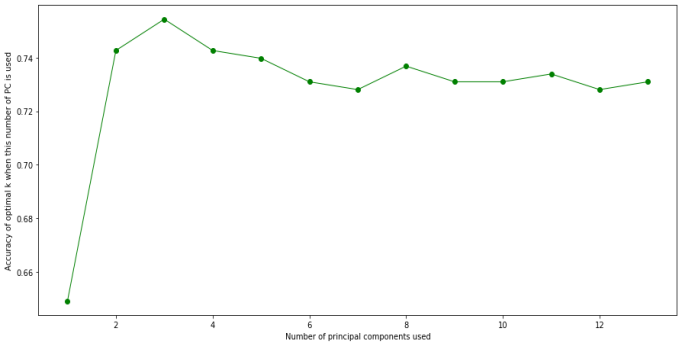

Figure 24. How accuracy of each optimal k changed when the number of principal components used increased, with standard scaler.

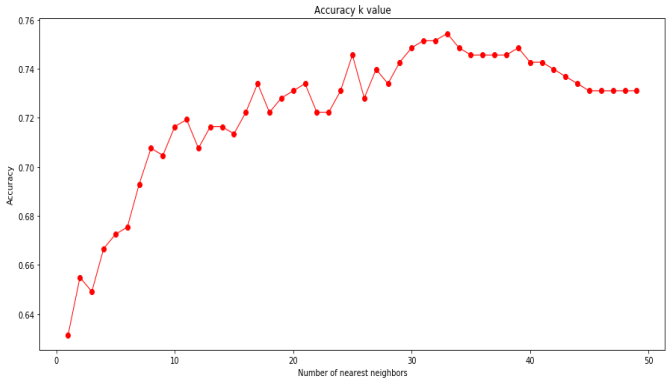

Figure 25. How accuracy changed as k increased when pc_count = 3

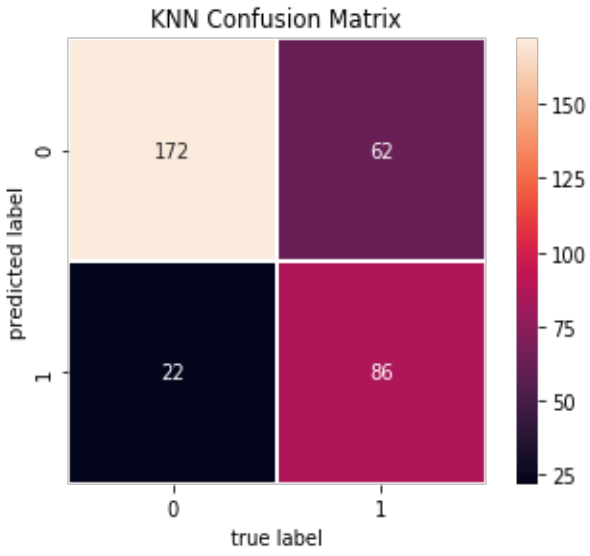

Figure 26. Confusion matrix when k = 33 and pc_count = 3.

	precision	recall	f1-score	support
0	0.74	0.89	0.80	194
1	0.80	0.58	0.67	148
accuracy			0.75	342
macro avg	0.77	0.73	0.74	342
weighted avg	0.76	0.75	0.75	342

Figure 27. Classification report when k = 33 and pc_count = 3.

There was a significant rise in accuracy between pc_count = 1 and pc_count = 2. In each combination of principal components, the accuracy (and the mean error) behaved similarly to when the Min-Max scaler was used.

4.2.3. KNN on data without PCA

On the other hand, when KNN was applied on the data prior to PCA, the accuracy peaked at 63.16% at k = 25. In this case, all of the variables were used, and no preprocessing (standardization or normalization) was done.

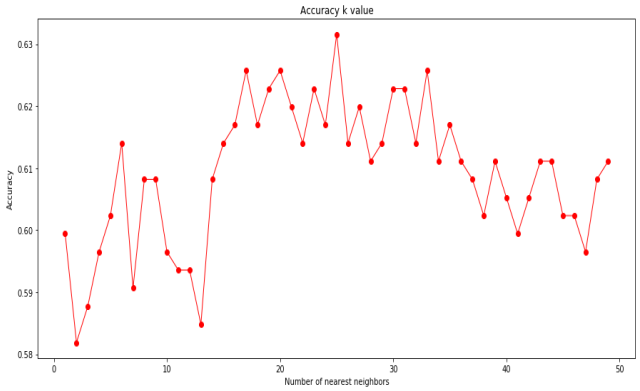

Figure 28. How accuracy changed as k increased when data was not transformed with PCA

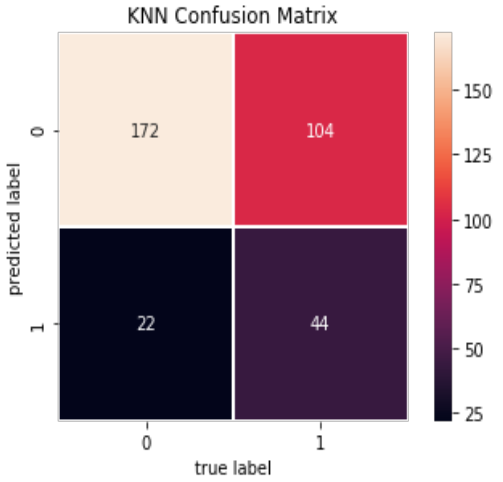

Figure 29. Confusion matrix when k = 25 and data was not transformed with PCA.

	precision	recall	f1-score	support
0	0.62	0.89	0.73	194
1	0.67	0.30	0.41	148
accuracy			0.63	342
macro avg	0.64	0.59	0.57	342
weighted avg	0.64	0.63	0.59	342

Figure 30. Classification report when k = 25 and data was not transformed with PCA.

However, the accuracy increased by roughly 10% when the data was either standardized (73.1%, k=17) or normalized (73.68%, k=18).

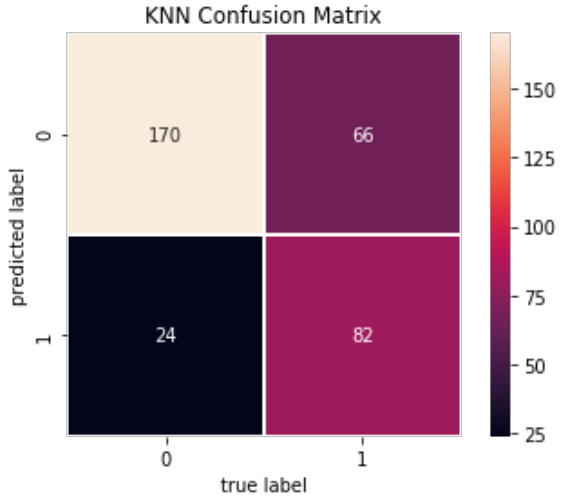

Figure 31. Confusion matrix when k = 18, and data was normalized (no PCA).

4.2.4. Dropping 'HTR'

When 'HTR' was dropped from the data, the standard scaler achieved a higher result than did the Min-Max scaler. After PCA, the latter achieved an accuracy of 73.39% with 25 nearest neighbors and 5 principal components, while the former achieved an accuracy of 74.85% with 17 nearest neighbors and 5 principal components.

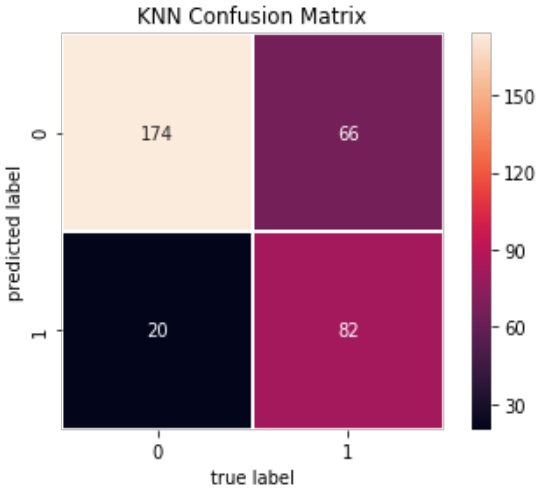

Figure 32. Confusion matrix when k = 17, pc_count = 5, and data was standardized.

5. Conclusion and discussion

5.1. Regression

According to the accuracy scores in these experiments, the most helpful method is by using multiple strongest indicators to train the linear regression model. Although it is the best, its accuracy is low and far from what we had expected. The remaining experiments' results showed similarly unhelpful. The results also reported that while the accuracy is low, the mean squared error could be significantly decreased as we can see in the second and third experiment. We can conclude that using multiple components or elements to predict a variable will yield better results than using just one. Additionally, we learned that dimension reduction techniques such as PCA will help a lot in multi-dimensional data processing and is easy to do.

From this we can conclude that using Linear Regression to train the datasets and predict a match's full-time goals is not helpful in this case. This is mainly because Linear Regression will be accurate when the datasets are linear, while the datasets we are using are not linear. Perhaps a more useful technique would be using Quadratic Regression.

5.2. Classification

In short, the highest accuracy (or the lowest mean error, because mean error = 1 - accuracy) was achieved when: PCA was applied to the data; the data were

normalized with either the Min-Max scaler or the standard scaler prior to PCA transformation; either 8 nearest neighbors and the first principal component were used (Min-Max scaler), or 33 nearest neighbors and the first 3 principal components were used (standard scaler). Although the accuracy scores were exactly the same (by comparison with no rounding) in both cases, there were slight differences in the confusion matrices.

Although it is difficult to determine which combination would be optimal, it is clear that applying KNN to un-preprocessed data before PCA would yield worse results than applying KNN to data after PCA. This is because KNN is based on Euclidean distance measure, and features being on different scales cannot contribute equally (Raschka, 2014). In contrast, the difference between applying KNN on PCA-transformed data and on non-PCA-transformed data became minuscule after the data was either standardized or normalized (without PCA).

When the Min-Max scaler was used, it only took the first principal component, which accounted for 30.88% of the total variance, and 8 nearest neighbors to achieve maximal accuracy. In contrast, when the standard scaler was used, it took the first 3 principal components, which accounted for 50.66% of the total variance, and 33 nearest neighbors to achieve the same maximal accuracy.

One conclusion is that cumulative explained variance was not a good indicator of how best to choose the number of principal components with regard to classification because the accuracy might even drop after using more principal components (refer to figures with green lines).

Another conclusion is that the method used to preprocess the data prior to PCA projection played a noticeable role in the performance of the k-nearest neighbor classifier. This is because the Min-Max scaler normalized the data to a range of [0,1], while the standard scaler standardizes the data such that the transformed data had zero mean and unit variance. Furthermore, the Min-Max scaler captured a crucial part of the data in the first principal component, whereas the standard scaler only accomplished that in the third principal component.

Another conclusion is that KNN performed better with data transformed with PCA. In other words, PCA improved the results of the k-nearest neighbor classifier. This is because PCA aims to maximize the variance, and KNN depends on distance measure which in turn depends on PCA, normalization, and standardization (Raschka, 2014). Also, normalization resulted in a higher retained variance for the first principal component than did standardization (30.88% compared to 26.02%).

The 'HTR' feature was the only variable with negative (-1) values; therefore, dropping it from the data had a noticeable effect on the performance of the scalers combined with the classifier. After dropping 'HTR', the models built with both of the scalers reached maximal accuracy when pc_count equaled 5, which suggested that 'HTR' was an important feature in classification.

Regarding classifier performance on the majority class and the minority class (Weiss & Provost, 2001), it was indeed the case that the KNN classifier produced worse results on the minority class (the 'Interesting' (1) class) than on the majority class (the 'Not Interesting' (0) class) in all experiments. For instance, in the first experiment in which the Min-Max scaler was used, and the accuracy reached the highest at pc_count = 1 and k = 8, the classifier only misclassified 28 samples of the majority class, whereas it misclassified 56 samples of the minority class (two times higher).

References

Herbinet, C. (2018, June 20). Predicting Football Results Using Machine Learning Techniques. Retrieved from https://www.imperial.ac.uk/media/imperial-college/faculty-of-engineering/computing/public/1718-ug-projects/Corentin-Herbinet-Using-Machine-Learning-techniques-to-predict-the-outcome-of-profressional-football-matches.pdf

Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. *JMLR, 12*, pp. 2825-2830.

Raschka, S. (2014, July 11). Retrieved from http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

Weiss, G. & Provost, F. (2001). The Effect of Class Distribution on Classifier Learning: An Empirical Study. *Technical Report ML-TR, 44*, pp. 4-5.