Multiple Linear Regression in Predicting Goals of Football Game and Using KNN Classifier to Label Games

Abstract

The problem of modelling football data has become increasingly hot in the past few years and many different models have been proposed with the aim of predicting the result of a game, or to estimate the goal of a particular match. We propose a multidimensional regression model to focus on the goals of football games using the data of the UK Premier league in seasons 2016, 2017 and 2018. We use PCA to reduce the dimension of the whole data and speed up our analysis. Furthermore, based on the final score of a match, we use KNN to classify the different matches and label it as interesting or not. After all the models were built, we use test data to evaluate our models, the result indicates our models have an excellent performance.

1. Introduction

Multiple linear regression is a mainstay of the application of regression. In real-world, it's pretty normal to deal with high-dimensional data. In our problem of this project, the goals of football game are obvious not only related to just one parameter. Since football has no doubt to be the most popular sport around the world, so it is worth to take some time in analyzing this interesting competition. Recently, there are many expert researchers put their attention in this scenario. Rothstein et al, used fuzzy logic model in formalizing football predictions, and genetic and neural optimization techniques in tuning their model. Koning took a Bayesian network method with Markov Chains Monte-Carlo, estimating the condition of football teams using this model.

2. Data Analysis

Our data consist of four *.csv file, the columns *FTG* and *Interest* are the features we are interested, which means the full-time goals and if this match is interesting or not, respectively. Beside these, we have 13 features can be used as our prediction factors. Obviously, the dimensionality of data is too high and with some useless information, we need to do preprocessing before starting to analyze.

2.1 Correlation

As we said above, the data was divided into four files, and we need to put the train data together into a pandas dataframe at first. Pearson correlation method was used to compute the correlation of all features and heatmap was used to plot it.

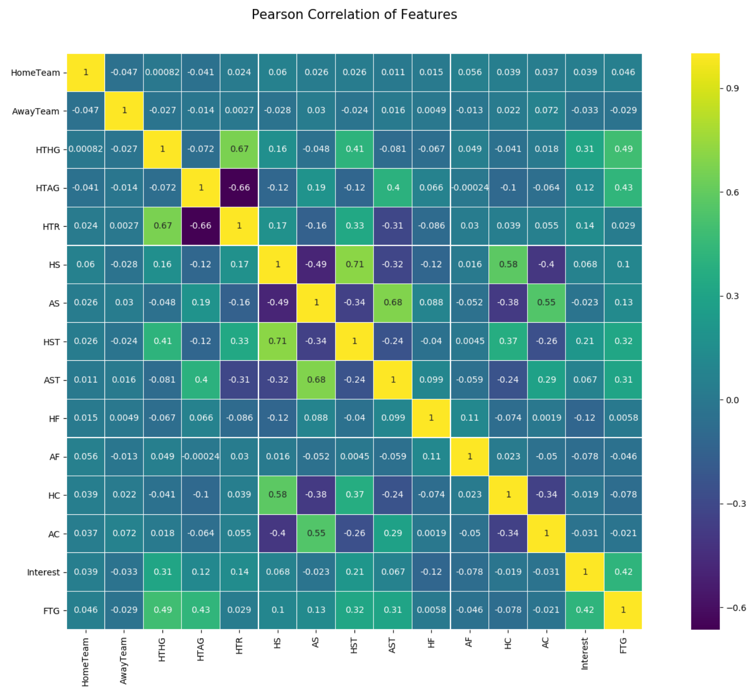

Figure 1: Pearson Correlation of Features.

From figure 1 we could know that there are some pairs of features have a strong relation, for example, *HTR*, *HTHG*, they represent the half time result and half-time home goals. We could detect from the experience in football games that these two features really have a big effect to each other. As for the influence from other features to *FTG* and

Interest, we can see that *HTHG*, *HTAG*, *HST*, *AST*, *HS*, *AS* have a unignorable correlation rate to *FTG*. On the other hand, other features only have a correlation rate that less than 0.1 which means it was not a meaningful influence factor to the result of the *FTG*, so we just need to ignore them when we construct the regression model to predict the *FTG*.

2.2 Pair Plot

By plotting the pair plot of two related features, we can clearly see the detailed vinculum between two particular features. There are six pair plots below:

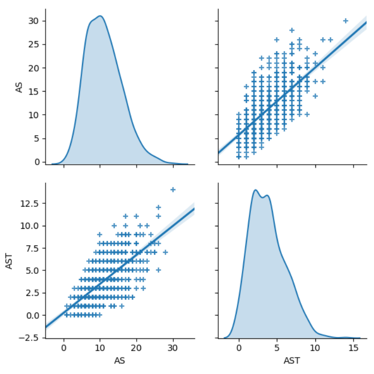

Figure 2: Pair plot of AS & AST.

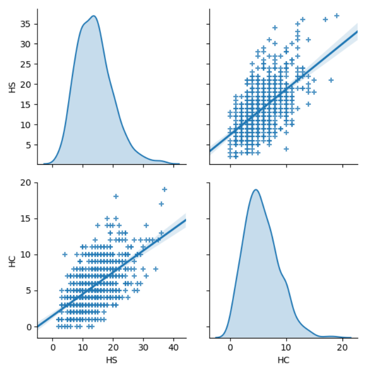

Figure 3: Pair plot of HS & HC.

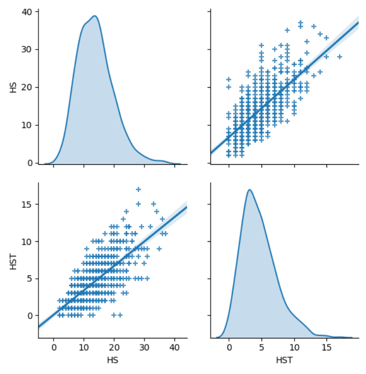

Figure 4: Pair plot of HS & HST.

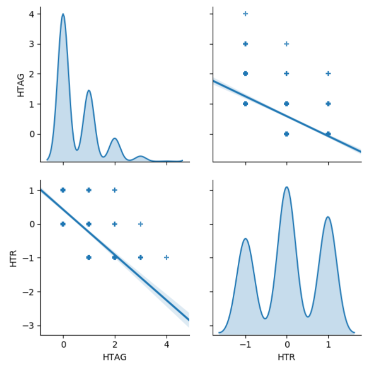

Figure 5: Pair plot of HTAG & HTR.

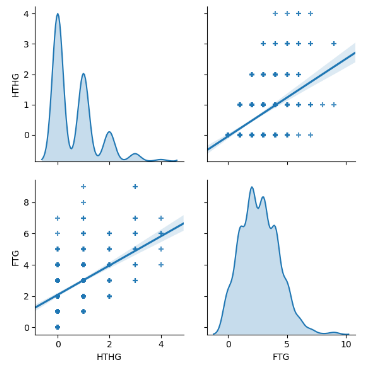

Figure 6: Pair plot of HTHG & FTG

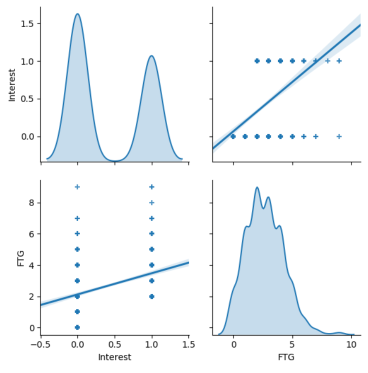

Figure 7: Pair plot of Interest & FTG

2.3 PCA

Principal Component Analysis (PCA) is a common way of speeding up a machine learning algorithm by reducing the dimensionality of high dimension data without lose variability. As we have dropped several features above and there are six interesting features (*HTHG*, *HTAG*, *HST*, *AST*, *HS*, *AS*) were selected to do the following analysis. First, we calculate the density of FTG and see what the main distribution of goals is. Since almost all number of FTG is in the interval [0, 6], we can just specific the goals between 1 to 6 when plot the PCA projection graph. We prepare a PCA projection using the first two components, and visualize it below:

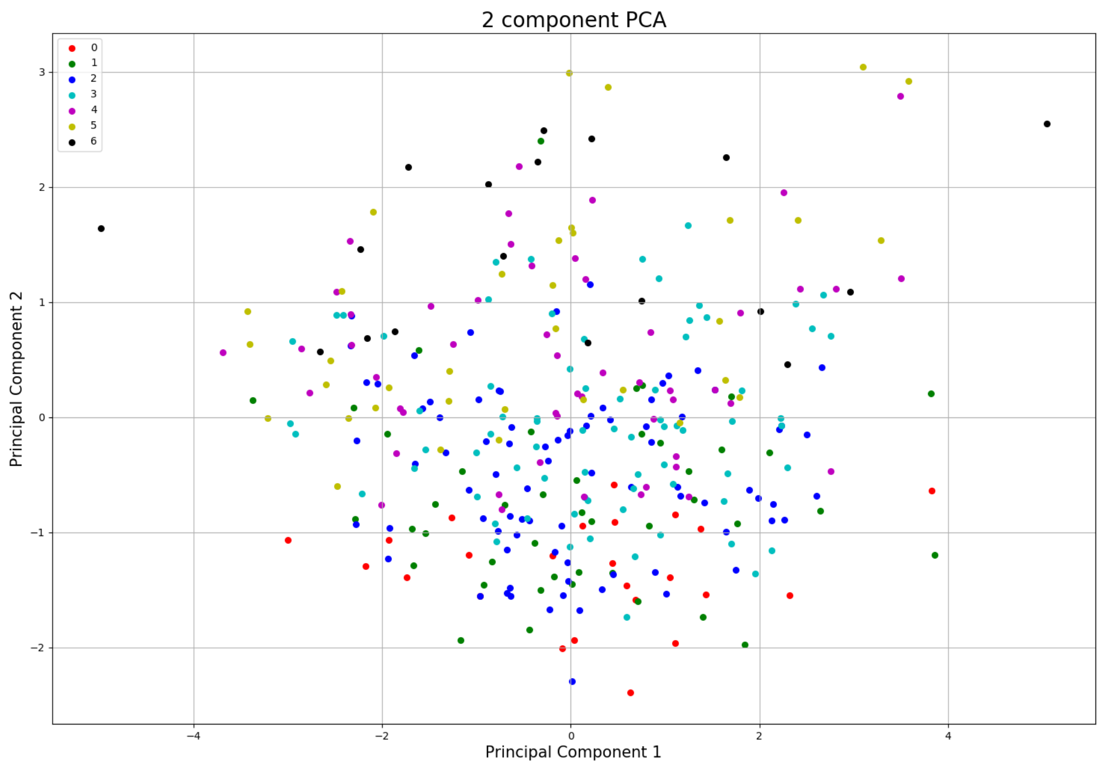

Figure 9: Two component PCA projection

Explained variance tells us how much information can be attributed to each of the principal component. We plot the five-component cumulative explained variance below:

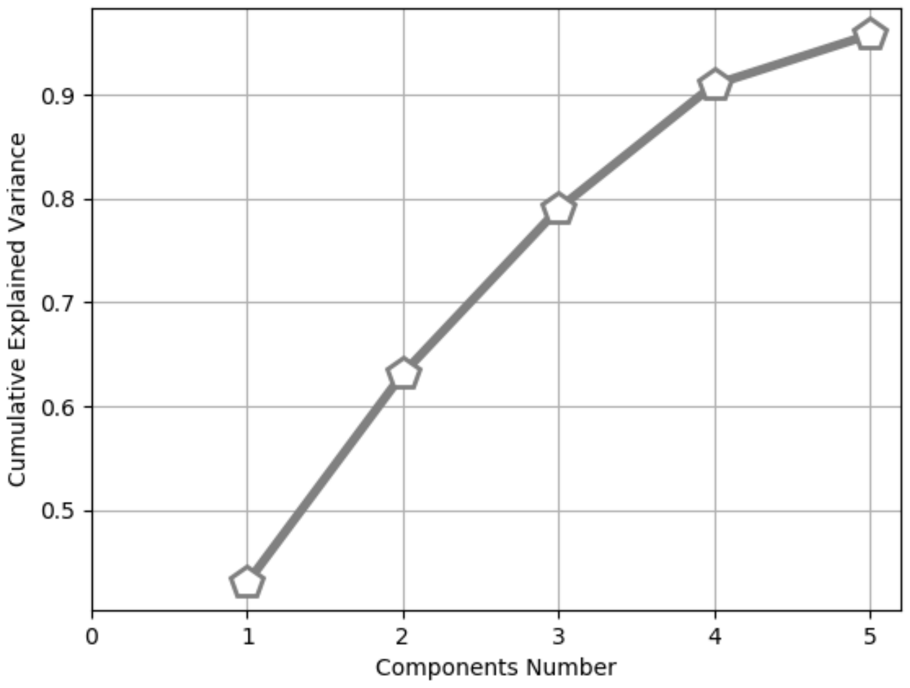

Figure 10: cumulative explained variance

From the figure above we could know that the first two components have more than 60% variance of the original data together, and when components number increase to 5, the cumulative explained variance is close to 1, which means it will not lose variability while reduce the dimensionality of the raw data.

3. Method

3.1 Multiple Linear Regression

Almost all the real-world problems that we are going to encounter will have more than one factors. Therefore, multiple linear regression will get a good performance in dealing with the real-world problems. Instead of implementing this popular model by ourselves, we use Python's Scikit-Learn library which has lots useful function for machine learning. We will take into account six features (*HTHG*, *HTAG*, *HST*, *AST*, *HS*, *AS*) and construct a multivariate linear regression model. First, we need to create an object of **LinearRegression()**, then use **fit()** method to train the algorithm, finally use **predict()**

function to predict and test our model.

3.2 K Nearest Neighbor

KNN is a non-parametric, lazy learning algorithm, which makes us immediately classify new data as they present themselves. It's pretty common and popular to use it for classification. For every new data m, we take the K nearest neighbors of the new data point according to their Euclidean distance, among these neighbors, count the number of data points in each category and assign the new data to the category that has more neighbors. We can also use Python's Scikit-Learn library to implement KNN method to solve our problem in this project.

4. Experiments and Results

4.1 Regression prediction result

We plot a bar graph to show the comparison of Actual values and Predicted values which generated by our model. First, we show the bar graph of regression model:

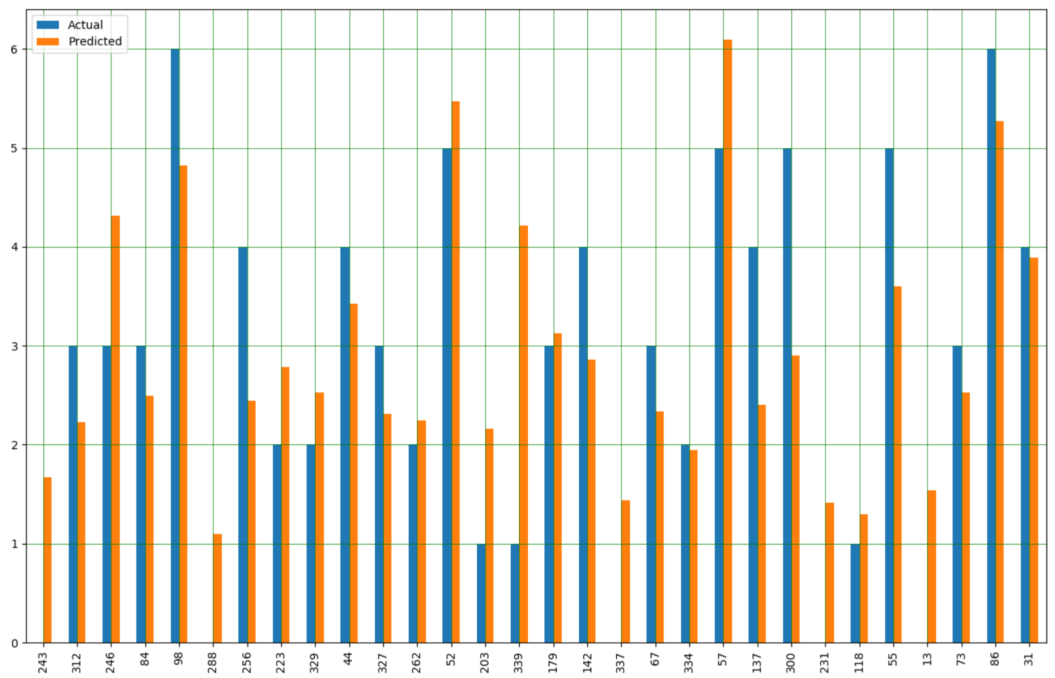

Figure 11: Bar graph showing the difference between Actual and predicted value.

Furthermore, we compute three evaluation metrics to see if our model is fine. They are:

Mean Absolute Error (MAE): 0.951998

Mean Squared Error (MSE): 1.435040

Root Mean Squared Error (RMSE): 1.197931

You can see that the values of three Error are all near 1, which means our model was not very accurate but can still make acceptable good prediction.

4.2 KNN classification result

First, we need to choose what's the optimal number of neighbors, we compute the error for K values between 1 and 40, and plot them to see which number is suitable for K.

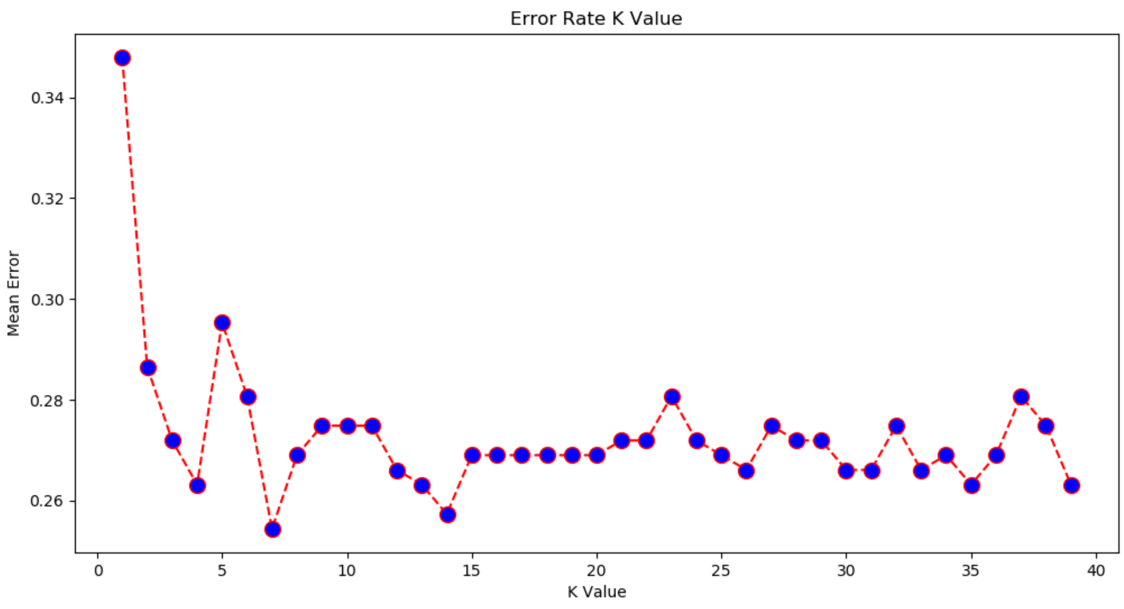

Figure 12: Error Rate for different K value.

We can see that when K=7, our model has the minimum mean error. So, we set K=7 as the number of neighbors in our KNN algorithm. The final classify result can be shown by the confusing matrix below:

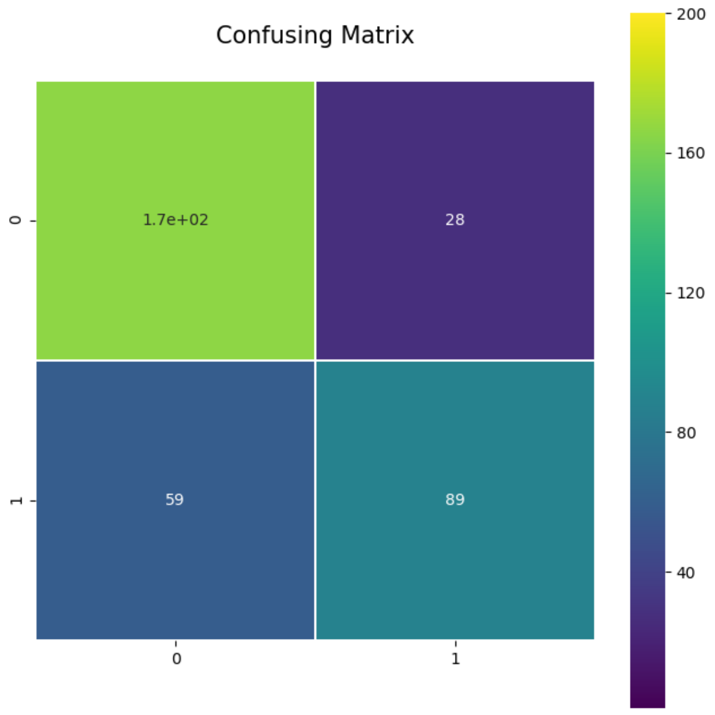

Figure 13: Confusing Matrix.

5. Conclusion and Discussion

In this project, we applied the most fundamental machine learning algorithm, linear regression to predict the goals of football game. We implement multiple linear regression with the help of Scikit-Learn Python library and get a nice prediction when compare to the real value. Then we did the job of classification using K Nearest Neighbors algorithm. KNN doesn't require any additional training when new data becomes available, so it's fast to put the data point into the nearest category and we also get a good performance when test our KNN classification mode. PCA was used throughout this project, we use it to reduce the dimension of raw data and it's important for speeding up our algorithm running time especially when deal with huge amount of data.

6. References

PCA using Python (scikit-learn)