FIFA Project: An Analysis of FIFA 19 and players across Major Leagues

Akshay Sridharan, Bill Luo, Jun Son, Kyuyeon Kim

It is ok to use an anonymized version of this report as an example of a great project for future classes.

Abstract

In the FIFA 19 dataset, each player contains numerous in-game attributes that distinguish them from one another. The attributes of each player are carefully engineered by the game developers to accurately represent their real-life counterparts. After extensive data cleaning, we conducted Principal Component Analysis before implementing classification techniques to see if the players' attributes showed any distinctive patterns based on their positions. After conducting PCA analysis on players' positions, classifying players according to their positions alone seemed simpler than we expected. To complicate the classification process, we proceeded to create a new set of classifiers that combined the players' ages and positions, namely 'pagegroup' (Position + Age group). The classification techniques we implemented include K-Nearest Neighbors, Random Forest, LDA, and various methods through scikit-learn in Python. Fortunately, all classification techniques performed better than the expected accuracy from random guessing, but certain classification techniques, such as XGBoosting, Gradient Boosting, and Bagging, had significantly higher test accuracies than other classification methods.

In the second part of the project, we wanted to test how accurately the player attributes from FIFA 19 predict the players' real-life performances. We imported another dataset that gave us the real-life performances of the players across 5 major European Leagues. Based on the real-life performances of the players, we constructed another classifier 'perf' that divided the players based on their performance ratings. Similar to our analysis done on FIFA 19 attributes, we conducted Principal Component Analysis on players' performances. Among the classification techniques, XGBoosting, Logistic Regression, and LDA performed the best. Then, we used several regression techniques such as PCR, Ridge, and Lasso to test how accurately the players' attributes from FIFA 19 predicted the players' real-life ratings.

Data

Our FIFA 19 complete players dataset was taken from Kaggle and contained 18,207 observations. 60 observations contained missing values in position and age, and were therefore eliminated. The data consisted of players' in-game and background characteristics such as stamina, agility, release-clause, and international reputation. Because our goal was to classify the players according to their positions and a combined classifier 'pagegroup', we eliminated unnecessary variables such as nationality and flag. Our data overall were relatively clean; the main adjustments made were converting height and weight into integer form, and deleting symbols that were unrecognizable by R when run. In creating the combined classifier 'pagegroup', we created a new categorical variable for age - young, middle, and old - according to the players' age. As the ages given in the data were discrete numbers, it was difficult to place an equal number of players into each category. However, the intervals we chose divided the players equally with negligible differences in the number of observations.

Our main goal of using the second dataset was whether the player attributes from FIFA 19 will accurately predict the players' real-life performances. The data were taken from kaggle and contained roughly 2500 observations, but only the players from 5 most recognized European Leagues were considered. Since we only needed the rating for each player (namely 'WhoScored' in the data), we manually extracted the ratings for the selected players and appended the data to our original FIFA 19 dataset.

Methods

For our exploratory data analysis in both parts of our project, we conducted Principal Component Analysis to observe any patterns among the players. In the first part of our project we classified the players according to their 'pagegroup', we applied KNN, LDA, SVM, and Random Forest in R. For the remaining classification techniques shown in the following sections (e.g. XGBoosting, Adaboosting), we used the scikit-learn library in Python. The 12 classifiers trained through sci-kit learn were evaluated with a 10-iteration random permutation cross-validator. From this, we obtained the mean train and test accuracies for each technique along with the 3 standard deviation value for each test accuracy. After retrieving the second set of data (WhoScored ratings), we ran identical classification techniques in Python based on the variable 'perf', as well as regression techniques such as Lasso, Ridge, Elastic Net, PCR, and Partial Least Squares on the WhoScored ratings in R.

'Pagegroup' Classification

Our first classification objective utilized the artificially constructed 'pagegroup' factor variables. The groupings were constructed based on the position and age groups our team formed to describe the patterns in the represented players. The positions were grouped such that players with values corresponding to any defensive position were labeled as 'DEF', with the same process following for offensive, midfielder, and goal-keepers. Age groups were likewise constructed using age cut-offs, with all players under the age of 23 being considered 'young', all players who had an age of 23 to 27 being considered 'middle', and all players above the age of 27 being considered 'old'. With these groups created, the 'pagegroup' factor variable was formed by combining the two classifications, thereby forming 12 'pagegroup' level.

Exploratory Data Analysis

Figure 1: PC Graph with colors denoting positions

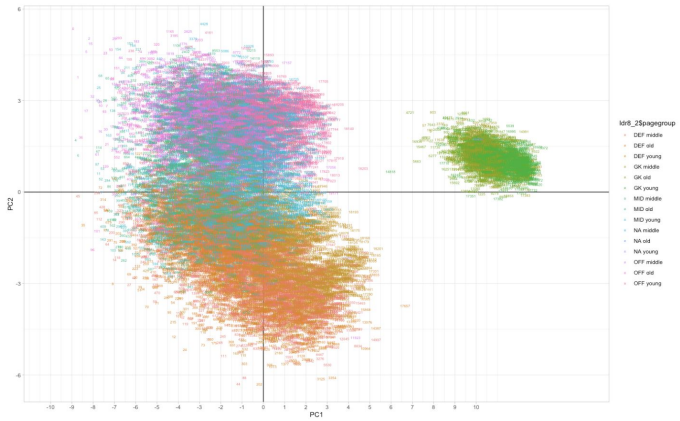

Figure 2: PC Graph with colors denoting pagegroup

Following the creation of the variable, our team decided to conduct an exploratory data analysis using Principal Component Analysis based on the newly constructed classifier 'pagegroup' (Figure 2). It was observable from the plot that the first principal component successfully separated the goalkeepers from other positions, while the second principal component direction provided the separation of the field players into three clusters – offense, defense, and midfielders. However, as the aforementioned interpretation of PCA was also able to be obtained through PCA on just the players' position classifications (Figure 1), our second PCA on 'pagegroup' did not give further insight into the data.

K-Nearest Neighbors

Given the nature of our dataset, we believed that a player would be best classified by observing a player or set of players with very similar statistics to themself. For this reason, we decided to train a K-Nearest Neighbors model. We trained the model in R with K values ranging from 1 to 25. The result from conducting sensitivity and specificity analysis is shown below (Figure 4).

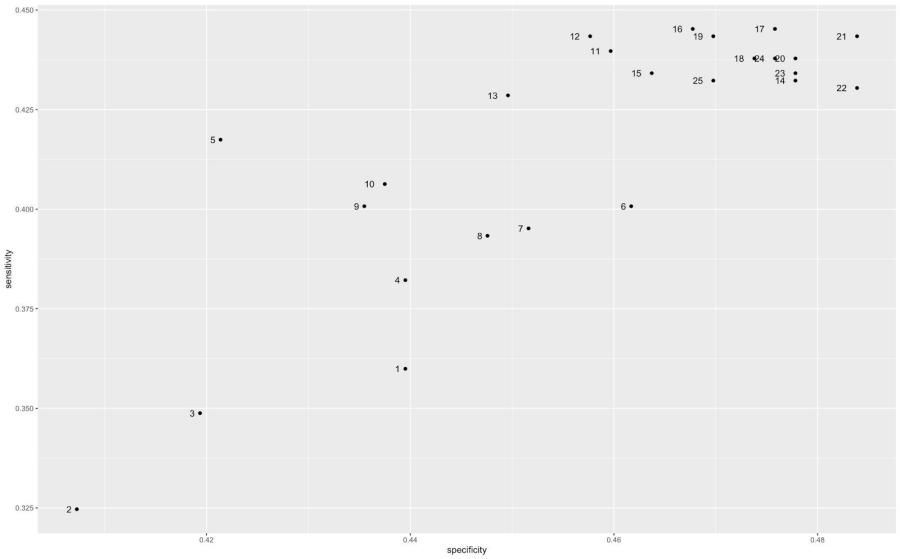

Figure 3: Sensitivity and Specificity plot for each k value

As observed in Figure 4, the K-value that had the highest value for sensitivity and specificity was 21, with sensitivity and specificity values of 44.34% and 48.39%, respectively. This means that the model performed best

when the class of the observation was determined by 21 data points that had the smallest Euclidean distance to the point, by majority class votes. After observing that any K-values higher than 21 showed a decreasing trend in sensitivity and specificity values, the analysis was done with K-values of up to 25 to avoid computationally intensive calculations.

Random Forest

After our use of KNN, we determined that the implementation of tree-based methods would likely produce robust results for this particular classification. We proceeded to build a Random Forest classifier in R to both determine the effectiveness of such methods for our data and to additionally view the variable importance measures provided by the method. From the method, we found that using 7 variables randomly sampled as candidates at each split and constructing 500 trees gave a sensitivity of 68.27% and a specificity of 82.46%.

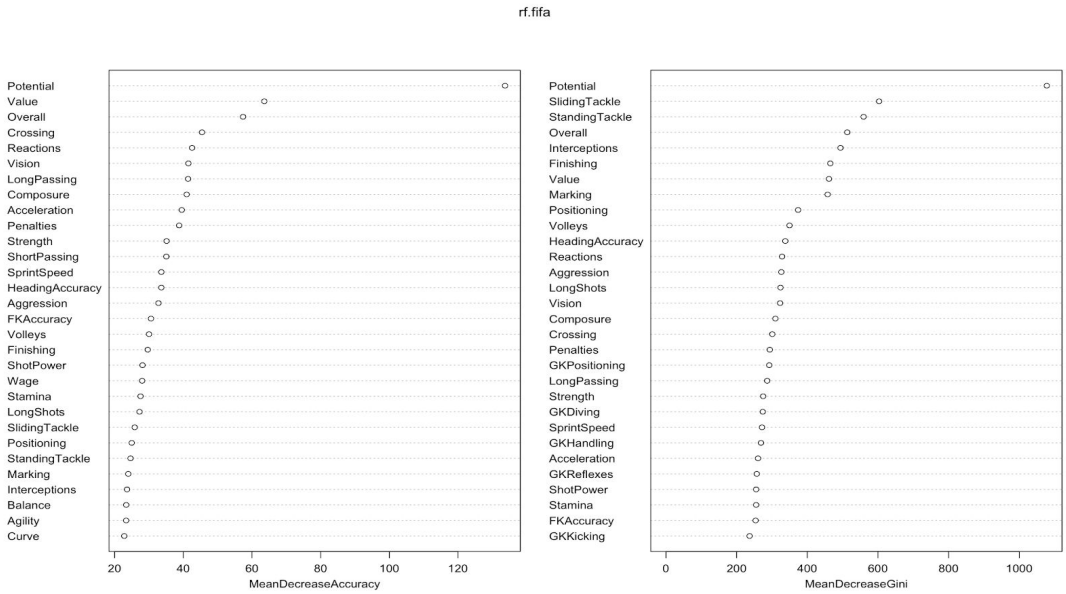

Figure 4: Variable Importance Plot for Random Forest Classifier

Variables that were important across both plots included ‘Potential,’ Overall,’ and ‘Value.’ For the left plot, the removal of those variables signifies the decrease in the predictive accuracy of the decision trees produced by the Random Forest algorithm. For the plot on the right, the addition of the variables decreased the node impurity of the tree by the amount that was specified on the plot.

Linear Discriminant Analysis

Following the implementation of the Random Forest technique in R, we proceeded to classifiers constructed in Python using the sci-kit learn package. One of the notable models from this group with respect to test performance was Linear Discriminant Analysis (LDA). Before training the model we checked the conditional normality of each variable across classes using R. Since the number of variables was 42, we constructed 42 histograms per class – a total of 42 * 12 histograms. As all variables followed a relatively normal distribution, we determined that the use of these variables would still produce a robust classifier and we thereby proceeded in training our LDA classifier. From the training methods, we found that our classifier had an average test accuracy of 65.02%, indicating that the model was a factor of 7.8024 better than the expected accuracy of random guessing.

XGBoost

The next notable classifier trained for ‘pagegroup’ classification was the XGBoost classifier. This classifier is an implementation of gradient-boosted decision trees, which we believed would provide effective classification given the data’s large numeric feature space. The model was trained using the same cross-validation methodology as the LDA classifier, yet produced better results with respect to test accuracy. From the training methods, we found that our classifier had an average test accuracy of 65.14%, indicating that the model was a factor of 7.8168 better than the expected accuracy of random guessing.

Gradient Boosting

The best performing classifier trained for the ‘pagegroup’ classification was made using the Gradient Boosting method. This method constructs shallow trees that tend to underfit the data and counteracts this underfitting by building many of them in sequence. The model, also trained using a 10-iteration random permutation cross-validator, gave an average test accuracy of 66.23%, indicating the model was a factor of 7.9476 better than the expected accuracy of random guessing.

Following the training methods, we produced normalized confusion matrices to illustrate the predictive capacities of the models using a stratified shuffle split cross-validation methodology, which returns stratified randomized folds that preserve the percentage of samples for each class in each fold.

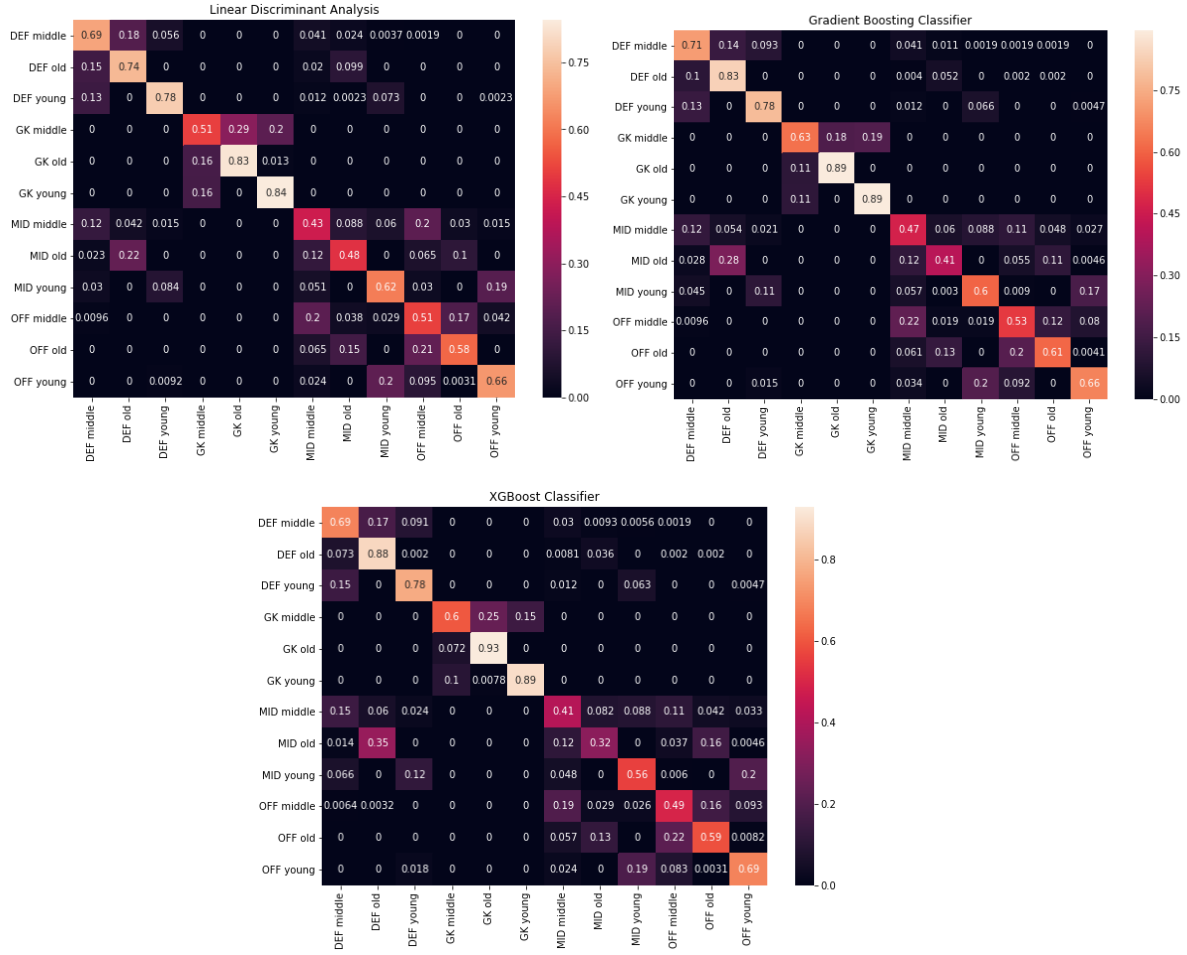

Figure 5: Normalized Confusion Matrices for Linear Discriminant Analysis, Gradient Boosting, and XGBoost

As observed from Figure 3, the three techniques tend to correctly classify players within both their age and position. However, it is interesting to notice that misclassified points tend to exist within either the correct age or position group. These self-similar 3-by-3 blocks are indicative of general misclassification across only one of the two groups for the majority of misclassified players.

WhoScored:

Our second classification objective utilized 'WhoScored' ratings, which we imported from a second dataset that rated the soccer players on a scale from 0 to 10, based on their overall performance in the 2018-19 season. Although the observations in the second dataset exceeded our first, we decided to only classify the players who belonged to the 5 most notable European Leagues, which amounted to around 2500 observations. Furthermore, to run classification techniques on the imported data, we created another categorical variable 'Perf' that divided the players into Excellent, above, average, below, and poor - according to their individual ratings.

Exploratory Data Analysis

The second PCA shown below (Figure 6) was conducted to observe if the newly constructed classifier 'Perf' produced any visible clusters. Similar to the PCA conducted on pagegroup, PC1 still separated the goalkeepers from field players. Also, we could observe that among the field players (the bigger cluster formed on the right), the players with higher performance ratings tended to form a semicircular band towards the right portion of the cluster.

Figure 6: PC Graph with colors denoting classes 'Perf'

'Perf' (Classification)

Our second classification objective focused on using players' FIFA 19 in-game attributes to predict their performance in the following season from 2018 to 2019. The groupings were constructed based on the rating scale shown on WhoScored's website. Because the number of players with WhoScored ratings of below 6.0 and above 7.5 was very small, players with a WhoScored rating of 6.0 or below were categorized as 'Poor,' and players with WhoScored rating of 7.5 or above were categorized as 'Excellent.' For the rest of the players, we

divided them up into three groups: players with a WhoScored rating between 6.01 and 6.50 as ‘Below Average’, players with a WhoScored rating between 6.51 and 7.00 ‘Average’, and players with a WhoScored rating between 7.01 and 7.50 as ‘Above Average.’ Using the same method from pagegroup classification, we constructed various models to predict players’ performance.

The classification model that had the highest testing prediction accuracy was logistic regression. Its testing prediction accuracy was 60.277%. Nevertheless, the logistic regression model obtained such a high prediction accuracy by classifying the majority of the players as “Average.” The confusion matrix below demonstrates this problem of over-predicting average or below-average players. The entries of the second column average have significantly higher values than other columns, indicating it was erroneously predicted. The group “Below Average” is the only other column that is lightly colored, suggesting that “Below Average” was the second most predicted group by the logistic classifier. Such pattern suggests the skewness in our dataset, with the majority of the observations falling into two classes: “Below Average” and “Average”.

The classification model with the second-best performance was Gradient Boosting. Its testing prediction accuracy was 58.628%. Unlike logistic regression, the gradient boosting classifier was able to correctly classify players across different groups. The diagonal entries of the confusion matrix represent the prediction accuracy of each group. Compared to the confusion matrix of logic regression, it is obvious that the gradient boosting classifier correctly classified significantly more players in “Above” and “Excellent.” A similar pattern can be discovered in the confusion matrix of LDA, which had the third-highest prediction accuracy.

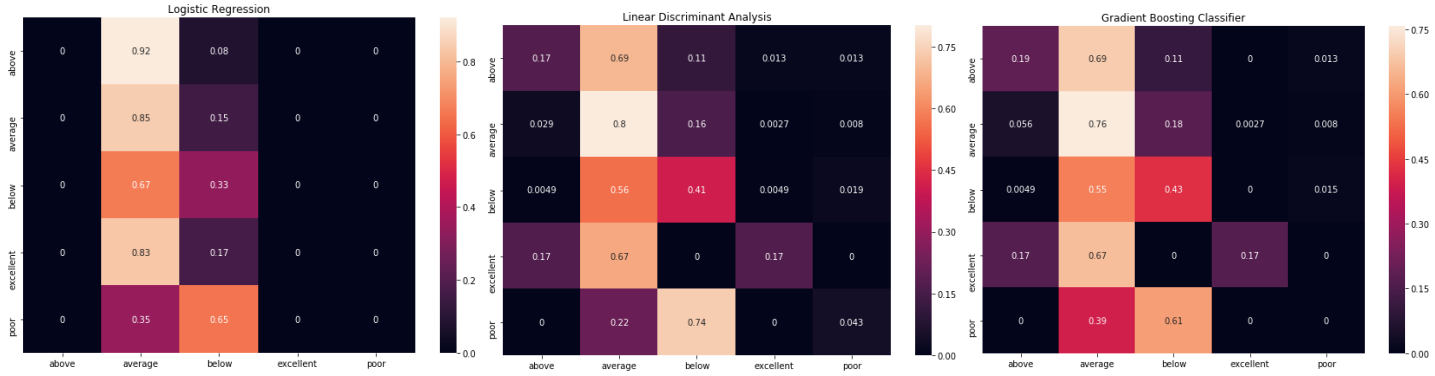

Figure 7: Normalized confusion matrices for Logistic Regression, Linear Discriminant Analysis, and Gradient Boosting Classifier

In addition to the previously mentioned classification methods, we also conducted other various classification techniques in Python. Figure 12 details the mean test accuracies of all classification methods conducted in Python using the sci-kit learn library.

WhoScored Regression

Apart from classifying soccer players into five groups, we also employed players’ FIFA 19 in-game attributes to predict their WhoScored rating in the following season, from 2018 to 2019. In total, six regression models were used: linear, Ridge, Lasso, principal components, partial least squares, and elastic net. For Ridge and Lasso models, we ran a 5-fold cross-validation in order to find the optimal lambda values of (0.01714, 0.00286). The mean squared prediction errors for linear, Ridge, and Lasso model are shown in Figure 8 below.

Model	Test MSPE
Linear	0.08237
Ridge	0.07795
Lasso	0.07818

Figure 8: Mean squared prediction error values for Linear, Ridge, and Lasso regression

For Lasso, the model eliminated 35 variables out of 39 variables, and the variables with non-zero coefficients were: Overall, Potential, Stamina, and Interceptions. Overall and potential have the two biggest coefficients, indicating that they are the two most significant predictors. In FIFA 19, potentially represents a player's predicted or peak overall rating. Thus, players with high potential will more likely improve their skills, consequently having a greater chance of performing well in the following season.

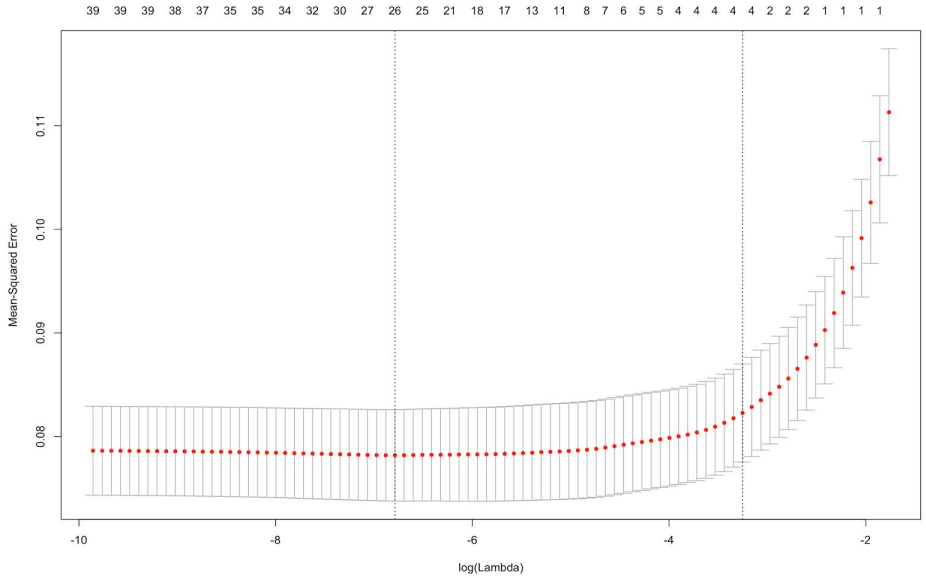

Figure 9: Mean squared error and number of non-zero coefficients for Lasso

For PCR and PLS, we created a validation plot to find the optimal number of components. The elbow of the curve indicates the optimal number of components.

Model	Test MSPE	Optimal Number of Components
PCR	0.08443	6
PLS	0.08241	4

Figure 10: Mean squared prediction error values and Optimal Component numbers for PCR and PLS

For elastic net, by using 5-fold cross validation, we selected the optimal model based on the smallest value of RMSE.

Model	Test MSPE	Optimal lambda value	Optimal alpha value
Elastic Net	0.07844	0.00343	1

Figure 11: Mean squared prediction error, optimal lambda and alpha values for Elastic Net

Raw Results

MLA Name	Pagegroup Test Accuracy Mean	WhoScored Test Accuracy Mean
Gradient Boosting	0.662311	0.581606
XGBoost	0.651378	0.594891
LDA	0.650184	0.586277
Bagging	0.631948	0.564526
QDA	0.62802	0.543796
Decision Tree	0.555489	0.472409
Random Forest	0.553675	0.55927
Logistic Regression	0.371497	0.602774
KNN	0.316307	0.545401
AdaBoost	0.275953	0.469343
Linear SVC	0.257005	0.443504
SVC	0.147634	0.549927

Figure 12: Test accuracies of classification methods conducted in Python

Conclusions

Through our analysis, we found that a number of our methods produced impressive results in both classification and regression. With regard to the ‘pagegroup’ classification, our best-performing methods performed better than random guessing by a factor of 8 on average. Particularly, gradient-boosted tree-based methods such as Gradient Boosting and XGBoosting performed very well, exceeding mean test accuracies of 65% in cross-validation. In respect to the ‘perf’ classification, we found similarly accurate results, with logistic regression and XGBoosting achieving greater than 60% mean test accuracy. Furthermore, while the pagegroup test accuracies drastically differed depending on the classification methods used, the test accuracies for ‘Perf’ or ‘WhoScored’, remained relatively constant across all methods. In addition, all of our regression models produced very similar results, with a test MSPE of approximately 0.08 across all regression methods.

Improvements

Throughout our project, we found instances where we could improve upon. In the WhoScored player classification, we discovered that some classification models, such as logistic regression, predicted most players as “Average” or “Below Average,” which are the two groups with the largest relative mass of players. In the future, we could improve by dividing the players into 5 equally-sized groups to avoid the issues presented by largely unequal groupings.

In addition, another possible improvement would be implementing a neural network to classify the players into “pagegroup” and “perf.” We believe that this would improve the test accuracy, as neural networks have the ability to learn and model non-linear and complex relationships, which we believe these groupings are best modelled by.

Team Roles

Akshay Sridharan: ‘pagegroup’ classification, data cleaning, Python coding, Plot-making
Bill Luo: (significant) data entry (5+ hours) ((very significant)), Python coding, R coding
Jun Son: Abstract, data, R coding, Python coding, and various sections in final report
Kyuyeon Kim: Data cleaning, (significant) data entry,R coding