Fantasy Premier League Data Modeling

Insert Subtitle Here

Anu Tamang  
 Data Science  
 University of St Thomas  
 St Paul MN USA  
 tama3041@stthomas.edu

Chris McDonald  
 Data Science  
 University of St Thomas  
 St Paul MN USA  
 cmcdonald@stthomas.edu

Suyash Shrestha  
 Data Science  
 University of St Thomas  
 St Paul MN USA  
 shre4281@stthomas.edu

ABSTRACT

Fantasy premier league is an online game where players assemble a team of real-life football (soccer) players from the English Premier League. The player’s team scores points based on actual statistical performances or their perceived contribution on field of play of their chosen players. Each week, players score points based on the minutes played, goals scored, assists in goals, keeping clean sheets and other key statistics. The points of the players are based on their own performance and sometimes the performance of real team they played for.

The dataset we selected was a collection of player statistic from the past 3 seasons (years). The dataset contained weekly record of all players in the premier league. The dataset had player stats like goals scored, assists made, and total points scored on a given week.

The main goal of this project to experiment with different predictive models on the selected premier league players dataset and learn which one would produce the best results.

We wanted to see if our models would help us determine how well the player would do in a given week. For the scope of the project we decided to focus on goals scored and total points as the determining variables to decide how well the player performs. So, in the project we focused on applying same modeling techniques on 2 different target variables separately. Then, we examined the results to see if changing the target had significant improvements.

**Questions of Interest**

Many questions arose throughout the processes of our project. The questions started right from data selection process. Many other questions came up when we analyzed the data and they continued till the end of our project. The answers for most of the questions we came across will be answered throughout this report.

One of the first questions we had was if we needed all the columns that our dataset had. Some of the columns were not related to our target variable so do we include them in our experiments or not? Should we still use them and let backward elimination and/or PCA handle them?

We were also not sure if we should normalize our data before modeling or not. Most of the columns had numbers lower than 10 but a few columns had values greater than 10 and lesser than 100. So, we wanted to see what difference normalizing the data before modeling would do.

We were also curious to see which model would produce the best results, which columns would have the most weight and which ones would be discarded.

Source of Data

The raw dataset was collected from a GitHub repository created and maintained by user “vaastav”. Vaastav states that the data provided is property of [Fantasy Premier League](http://fantasy.premierleague.com/) and [Understat](https://understat.com/). The GitHub repository had data from the past 3 seasons along with data from current season as well.

We looked at a few other alternative datasets, but we decided that this was the best data for out project. Other datasets we came across did not have as many rows and did not go as far as 3 seasons back either. We could only find another dataset that was for only the 2018-2019 season. This dataset seems to have the greatest number of weekly data and seems to be well maintained as well which made our choice easier.

The data for each season was found in separate folders. Each seasonal folder had separate csv files of weekly player data for the season. So, each of the folders except current season folder had 38 csv files since there are 38 game weeks in a season.

Data Collection

Our first step was to download the repository and combine all the data so that we could get it in one file, so that we could load it in Jupyter. We concatenated all the files in each folder using `cat` command and then combined all the combined files into one final file named 'complete\_gws.csv' using the same command. The files were named as gw1, gw2 and so on. So, we used “cat gw\* > 2016\_gw.csv” command to concatenate game week data of 2016. We did the same on other folder as well and then finally combined these yearly game week csv files. After combining all the files, we ended up with 55 columns and 72,785 rows of data.

Data Analysis

Out of the 55 columns there were only a few were categorical variables. The “name” of the player, “kickoff\_time” and “kickoff\_time\_formatted” were the only non-numeric columns in our data.

Before starting our project, we wanted to make sense of the information given in different columns and better understand them. There were many columns that we found interesting and got us wondering how they would impact our results. The dataset consisted of a mixture of in game statistics and fantasy premier league statistics as columns. We were more interested towards the in-game statistics than the fantasy premier league data columns and assumed that the in-game features would perhaps be the more true and accurate values to consider. And from the start we were mostly looking at the “goals\_scored” and “total\_points” columns. “goals\_scored” gave us the number of goals scored by the player on the given week and the “total\_points” were the fantasy premier league points that the player achieved for the week based on his game statistics and performance. We were interested in making one of the 2 or both as our target variables.  
Besides those 2 columns, we tried to look at most of the other columns and see if all of them looked relevant to our project. Some columns we assumed would really have significant weight on our model while some looked redundant. The columns that we thought would be helpful to our cause were players in game attacking statistics like “assists”, “attempted\_passes” ,“big\_chances\_created, ”big\_chances\_missed”, “bonus”, "creativity", "dribbles", "influence", "key\_passes", "open\_play\_crosses", "minutes", "offside", "opponent\_team", "penalties\_missed", "penalties\_saved", "threat", "was\_home", "winning\_goals". Similarly, we found other features which did not look as useful. Most of the features that were non-game stats like “loaned\_in”, “loaned\_out”, "transfers\_balance", "transfers\_in", "transfers\_out" and "value" looked less helpful as they did not really reflect the true statistics of the players on the pitch. On researching more about some of the other fantasy premier league data columns, we did find some helpful columns as well. Even though columns like “influence”, “treat”,” creativity”, “ict\_index”, “bps”, “bonus” and “total\_points” were actually fantasy premier league data for the players, they were based on real life performances.

We knew that the “total\_points” column reflected the overall points collected by the player in the given week. But we had to learn more about the other columns to make sure what they meant and if it would make sense for us to use them in our modeling.

According to the fantasy premier league website, ‘influence” is a measurement that evaluated the degree to which that player has made an impact on a single match or throughout the season. It also says that it takes into account events and actions that could directly or indirectly affect the outcome of the fixture.

Similarly, it says that “Creativity” asses the player performance in terms of producing goal scoring opportunities.

And it describes “Threat” as a value that examines a player’s threat on goal. They say that it gauges individuals most likely to score goals. It also goes on to say that “while attempts are the key action, the Index looks at pitch location, giving greater weight to actions that are regarded as the best chances to score.”

All these three scores are combined to create an overall “ict\_index” score. Based on the description these values looked helpful to our cause.

We also discovered that “bps” meant Bonus Point system which utilizes a range of statistics supplied by Opta that capture actions on the pitch, to create a performance score for every player. This score is also reflected in the “bonus” column.

**Data Preparation**

After some analysis and getting the csv file ready, we tried to read the file in Jupyter using pandas dataset. We ran into some issues getting the file loaded to data frame at first. We discovered that names of some of the players contained special characters. After some research we found that the file was encoded in "ISO-8859-1" format. We were finally able to load the dataset to pandas dataset by using “pd.read\_csv ('complete\_gws.csv', encoding="ISO-8859-1")”. After loading the dataset, we had to drop the categorical columns “name”, “kickoff\_time” and “kickoff\_time\_formatted” from the pandas dataset.

We also discovered the data from row 67936 did not have all the columns. This was because the data from the current season had different structure that previous years data. Therefore, we decided to drop rows below 67936 from our dataset as well.

**Approach of Experimentation**

We decided to choose “goal\_scored” as our target variable. We decided to apply various regression models to see if we could predict our target variables. We then split the data into 70% train and 30% test set. Then we would examine the results of applying Multiple Linear Regression, Multiple Linear Regression after backward elimination, Decision Tree Regression, Random Forest. We also ran PCA model and K-fold Tests for cross validation.

We wanted to run Polynomial Linear Regression and Kernel PCA but unfortunately, we were not able to experiment with those due to lack of computing power in our local machine. These models would either crash the Jupyter or the computer. We also tried to use google Colab and its computing power but that did not help either. We even tried to see if we could use google cloud console or AWS services, but due to lack of experience were not able to completely get it to run in cloud with more computing power.

To get most of the answers to our questions we decided on a trying a set of experiments for each of the models specified. We want to check if normalizing our data would produce better results. We also wanted to see if only using columns that reflected the on-pitch statistics of the player (and ignoring most of the fantasy premier league) would yield better results. So, our approach to the issue was to run the these set of models 4 ways.

We first ran the models without normalizing the data using all the available columns. Then, we ran the not-normalized data with selected columns what reflect on-pitch statistic along with some Fantasy Premier league data that we thought were valid (as specified in our analysis section). We decided to drop columns "ea\_index", "loaned\_in", "loaned\_out", "selected", "id", "transfers\_balance", "transfers\_in", "transfers\_out", "element", "selected" and "value". The rest were going to be our selected columns for the project. We then repeated this with normalized data. The results from the models were really interesting to examine. In this report we will be going through each model individually and how they performed for the 4 approaches we took.

All approaches were experimented using different python and machine learning libraries learned in class. We used “numpy” and “pandas” library to prepare and manipulate data. For machine learning techniques we used “sklearn” and its models. We also used “statsmodels” library to help us with backwards elimination process.

Multiple Linear Regression

We were mostly interested in seeing what the weights of the different columns would be in this model. We were interested to see if our initial assumptions of influential features during data analysis was accurate or not. We were also interested in seeing what the effects of normalization and using limited features would have in the mean squared error of the model.

First, we tried without normalizing and using all the columns. We found that the mean square error was about 0.7%. The parameters with the 5 highest weights were “red\_cards”, “own\_goals”, “errors\_leading\_to\_goal”,” penalties\_missed”, “penalties\_conceded”. These parameters did not seem correct to us. On examining these columns, we noticed that most of the values in these columns were all 0s.

Then we ran the model again without normalization but with our selected columns. The results were not any different. We got the same mean squared error of about 0.7% and the same 5 top weights.

Then we decided to try to build the model after normalizing our data. First, we ran the normalized data with all columns. The mean square error did not improve as it stayed at 0.7%. However, after normalization we did get new top weights. This time we got “ict\_index”, “influence”, “id”. “bps”, and “total\_points” as the top weight parameters. These parameters do seem to make more sense without target variable as they seem more likely to determine the number of goals scored.

We got the same mean squared error of 0.7% for our normalized dataset with just our selected columns as well. But since column “id” was not one of our selected columns, it was replaced by “fouls”.

From the results we determined that normalizing the data was important for the model. Even though the mean square error did not change, the weight parameters made more sense after we normalized the data. We also saw that one of our unselected columns came up in the top 5 weight parameters which was really interesting.

Multiple Linear Regression after Backwards elimination

We were interested to see which columns backwards elimination would remove and check if any of them were for our unselected list of columns. We were then interested to check the weight parameters and MSE for our 4 approaches.

For all the 4 approaches MSE did not change. It remained at 0.7%.

When we ran with all columns Backward elimination removed 2 columns. It removed columns “transfers\_out” and “transfers\_balance” which were from our unselected list of columns. The weight parameters did not change after backwards elimination. They remained the same as non-normalized data with all columns specified above.

All the results for the selected columns with non-normalized data remained the same except that it did not remove any columns since they were already taken out by us. The weight parameters were also unchanged by the backward elimination process.

For normalized data, the results were same. When running with all columns 2 columns were removed and the weigh parameters were same as without backward elimination.

When ran with normalized data with selected columns no columns were removed and the weight parameters were not changed.

Backwards elimination did not really help us improve the model. But it helped us understand about 2 columns that would be dropped.

Decision Tree Regression

For all 4 approaches we got the same results without any significant improvements. We got Mean Squared error of about 0.5% with all our approaches. The results for these models do look better than our multiple linear regression models from above.

Random Forest

For all 4 approaches we ran random forest models with different “n\_estimators” hyperparameter. We started with 10 as the initial “n\_estimator” and continued till 300 with a step of 10 on each run. For all of our process we got a minimum mean squared error of about 0.2% but we found the "n\_estimators to be different. The results of the processes are shown in graphs below.

These n\_estimators counts for the 4 processes are:

|  |  |
| --- | --- |
| Process | Minimum n\_estimators |
| All Columns (Normalized) | 140 |
| All Columns ( Non Normalized) | 120 |
| Selected Columns(Normalized) | 130 |
| Selected Columns(Non Normalized) | 270 |

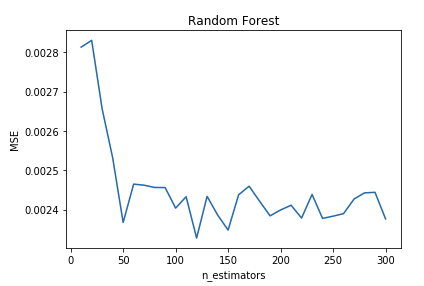


Figure 1: All Columns (Non-Normalized)

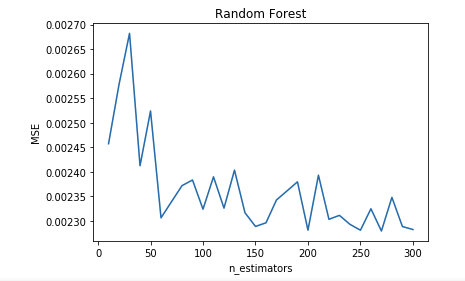


Figure 2: Selected Columns (Non-Normalized)

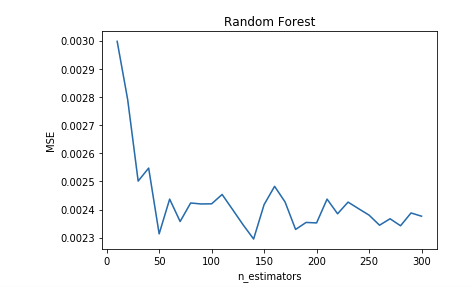


Figure 3: All Columns (Normalized)

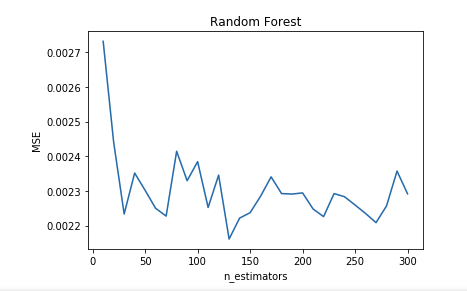


Figure 4: Selected Columns (Normalized)

PCA Model

The last technique to be used with this data is using PCA in conjunction with the Multi-Linear Regression Model. What will be examined is the lowest testing error and the associated number of components to meet that error value. After these values are determined a comparison will be made between the dataset with all inputs and selected inputs. The comparison between non-normalized and normalized will not be examined due results shown earlier.

When using PCA the lowest testing error for was 0.7% and number of components being 48 for the dataset that had all input columns. For selected inputs the lowest testing error was 0.7% and the number of components being 37.

These values recorded are nearly identical to those recorded for the Multiple-Linear Regression Model containing backward elimination. The results for the model can be found in the graphs shown below

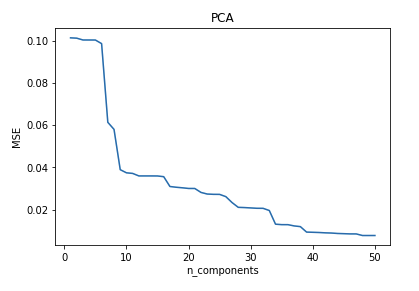


Figure 5: All Columns (Non-Normalized)

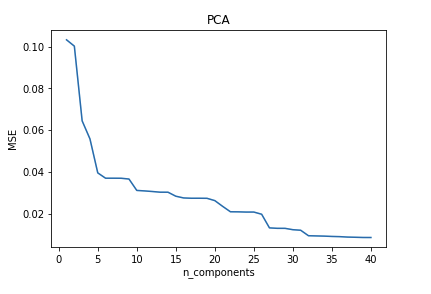


Figure 6: Selected Columns (Non-Normalized)

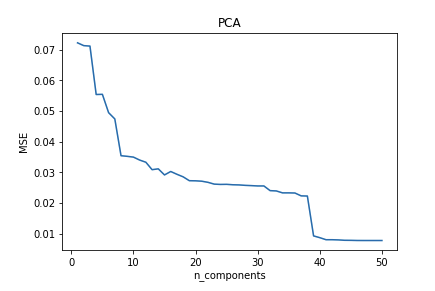


Figure 7: All Columns (Normalized)

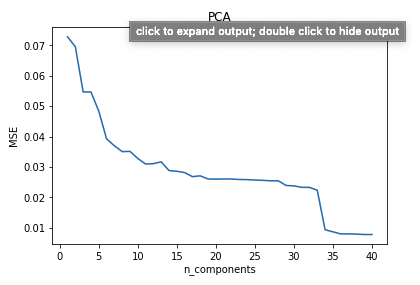


Figure 8: Selected Columns (Normalized)

K-Fold Cross Validation

The final test on the models created will be a K-Fold Cross validation. This test will display the model accuracies for Multi-Linear Regression, Multi-Linear Regression with backward elimination, Decision Tree, Random Forest and PCA. To show this a comparison will be made between non-normalized, normalized, large amounts of columns removed and the minimum number of columns removed.

The first comparison to be made is with the non-normalized and the normalized models. For the Multi-Linear Regression models the mean accuracy was 92% for both non-normalized and normalized. When backward elimination was introduced to the Multi-Linear Regression a mean accuracy was 92% for both non-normalized and normalized. This trend of the non-normalized and normalized accuracies being the same continues through all the models tested.

Through these observations seen above one can conclude that the data does not need to be normalized to produce the best model. This means that no one value in the data set determines the outcome of the prediction.

The next comparison is between the models created using 39 inputs/columns and all inputs/columns. For the Multi-Linear Regression model, the mean accuracy was 92% for 39 inputs and 93% for 41 inputs. These accuracies where seen again for the Multi-Linear Regression with backward elimination being used. The accuracies for the Decision Tree model were 94% for both 39 and 41 inputs used. For the Decision Tree model there was an accuracy of 94% for both. Lastly, for the Random Forrest model there was an accuracy of 97%.

Just like before there is little to no difference in the model accuracies between the changes in the amount of inputs. Again, this means that no one input drags the prediction one way or another. This could also indicate that there is so many inputs that we are overfitting the models.

The last comparisons that will be made is between each of the models that were used in a single run. The accuracies for all models are 92% for Multi-Linear Regression, 92% for Multi-Linear Regression with backward elimination, 94% for Decision Tree and 97% for Random Forest. Through this information one can conclude that the Random Forest model would be the best for examining the dataset selected.

The results of K folds experiments are shown in tables and graph below

Conference Name:ACM Woodstock conference

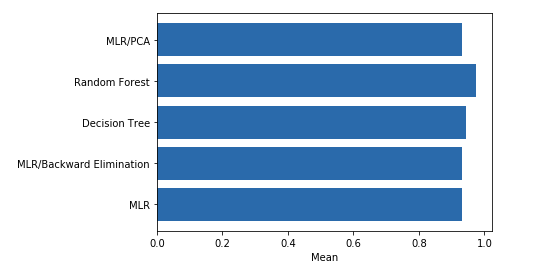
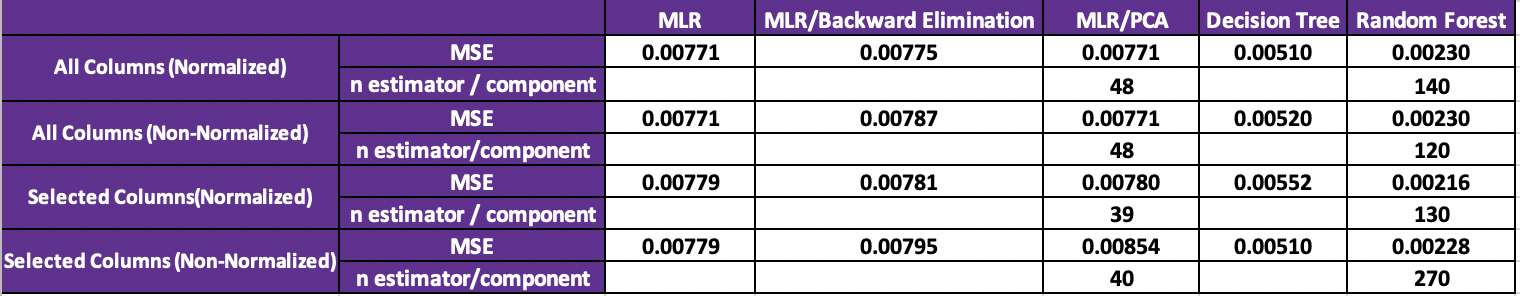


Figure 9: All Columns (Non-Normalized) Mean

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00

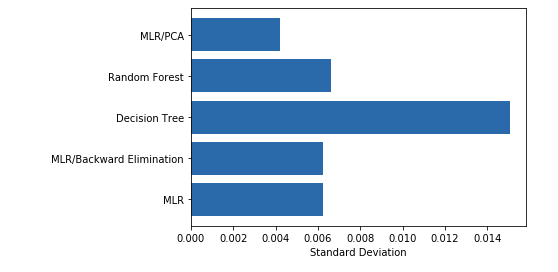


Figure 10: All Columns (Non-Normalized) Standard Deviation

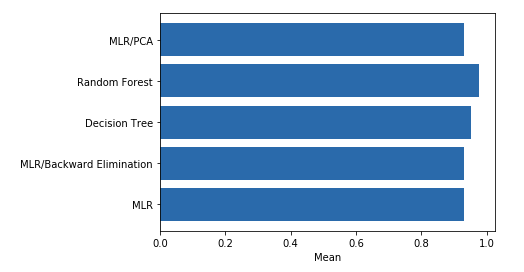


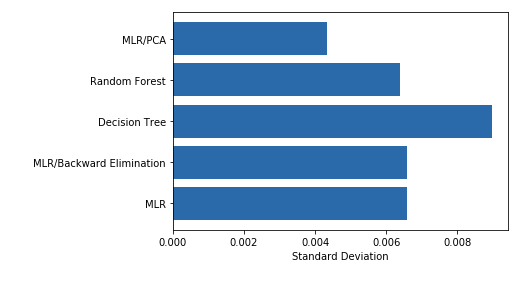
Figure 11: Selected Columns (Non-Normalized) Mean 

Figure 12: Selected Columns (Non-Normalized) Standard Deviation

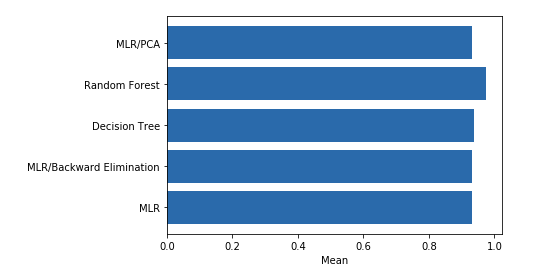


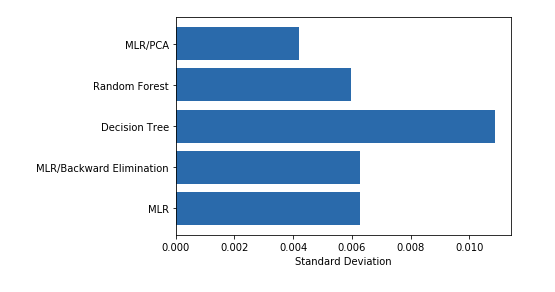
Figure 13: All Columns (Normalized) Mean 

Figure 14: All Columns (Normalized) Standard Deviation

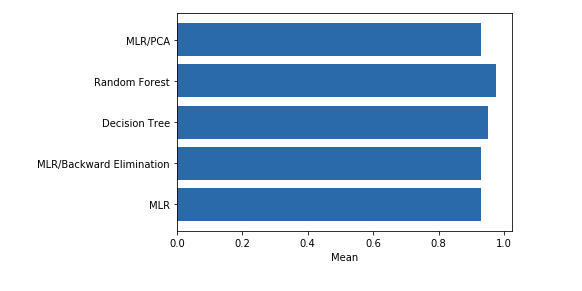


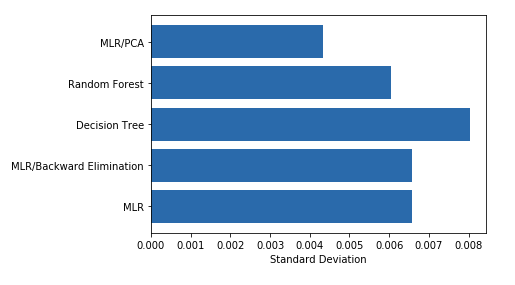
Figure 15: Selected Columns (Normalized) Mean 

Figure 16: Selected Columns (Normalized) Standard Deviation

Result Summary

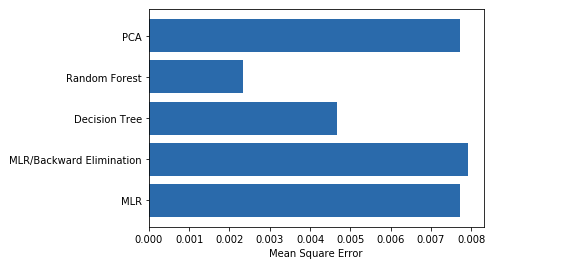
From all our results looks, the lowest MSE we got was for random forest with 0.2%. Decision tree was not very far behind with 0.5% 

Figure 17: Mean Squared Errors of Different Models

MSE. All our Regression models had about 0.7% MSE and we were not able to improve it significantly.



Figure 18: Results Summary

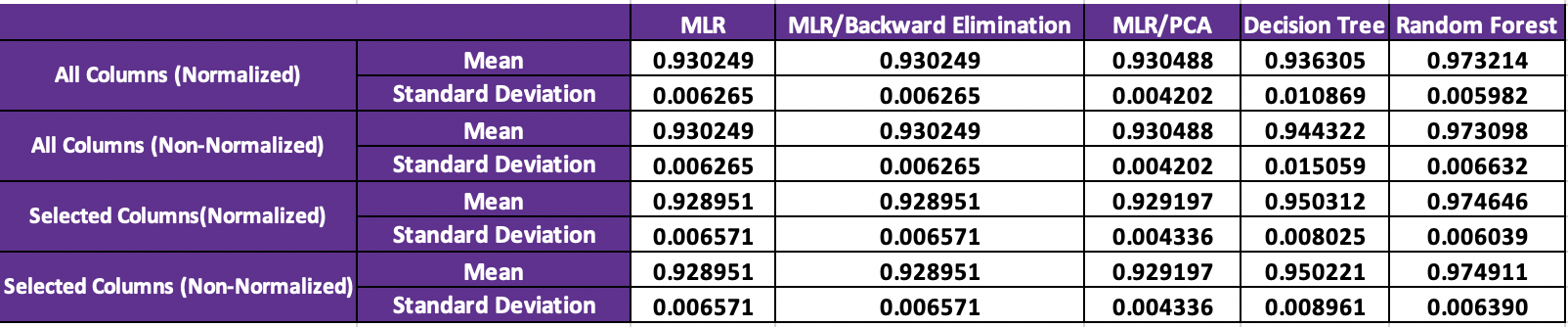


Figure 19: K Folds Results Summary

**Conclusion**

ACKNOWLEDGMENTS

All 3 of us would like to express special thanks to our professor Dr. Manjeet Rege for helping us through his lectures, notes and direct feedback throughout the project. His notes and lecture videos were very helpful throughout our project.

We would also like to express our gratitude to Vaastav whose GitHub repository helped us get our project up and running. His data was very clean and well maintained which made our experimentation much easier.

REFERENCES

[1] Anon. How the FPL Bonus Points System works. Retrieved December 13, 2019 from https://www.premierleague.com/news/106533

[2] Anon. How the ICT Index in Fantasy works. Retrieved December 13, 2019 from <https://www.premierleague.com/news/65567>

[3] Vaastav. vaastav/Fantasy-Premier-League. Retrieved December 13, 2019 from <https://github.com/vaastav/Fantasy-Premier-League>

[4] Anon. 2019. Value in FPL – 2019/20 Report. (July 2019). Retrieved December 13, 2019 from <https://whogottheassist.com/value-in-fpl-2019-20-report/>

[5] Anon. learn: machine learning in Python - scikit-learn 0.16.1 documentation. Retrieved December 13, 2019 from <https://scikit-learn.org/>

[6] Anon. Python Data Analysis Library¶. Retrieved December 13, 2019 from <https://pandas.pydata.org/>

[7] Anon. Overview¶. Retrieved December 13, 2019 from https://matplotlib.org/contents.html