Prediction Analysis on Wage for FIFA 2018 Players

Content

I. Executive Summary

In this project, we utilized second-hand dataset *FIFA 2018 Complete Player* to build a predictive model by using soccer ball players' past performance scores to predict their wages. We implemented 3 major analysis, including a decision tree, the decision tree with clustering, and random forest with clustering analysis. We compared each of the model and decided on using random forest with cluster as our final model. It is hoped that the predictive model can help company to estimate the wages of the soccer ball players with maximum accuracy and could be implemented the model for different business implication.

Our project is organized as follows. Section II described the motivation and background knowledge of the project. In Section III, we had an overall view of our data through exploratory data analysis and did basic data cleaning. Section IV pointed out all the challenges we have and we gave corresponding solutions. Continue with Section V, we introduced two models, includes decision tree algorithm and random forest algorithm. Section VI presented the managerial implication and gave real world implementations. Section VII concluded the project.

II. Project Motivation/Background

Sports Analytics has been around since decades, but it became popular in the mainstream sports culture when it was picturized in the movie 'Money Ball'. In the movie, sports team manager Mr. Billy Beane used sabermetrics- a type of statistics specific to baseball game and to select those players who would yield more runs in the field. The result of the move was that the team had more runs than strike-out. The success of this strategy completely changed the on-field and off-field understanding of the game.

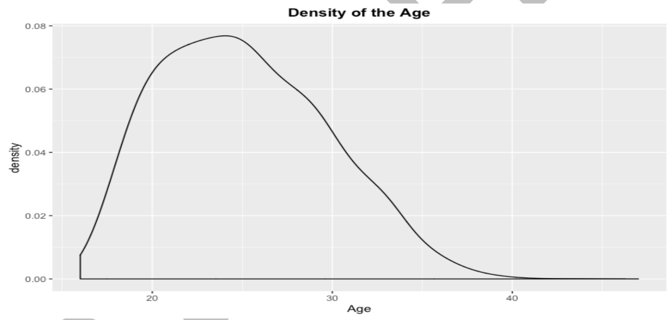
In today's world, sports analytics is continuously evolving. Almost every other team has an analytics expert who makes data analysis for the team. On-field, teams use sports analytics to gain competitive advantage over the other team whereas off-field various stockholders analyze data to make a profit. Sports Analytics gained momentum because of the huge availability of the data. One of the popular use of sports analytics is to estimate the wage of the players. By estimating the wage, we can optimize the value of the team. We can make sure that the players are being paid according to their skill level and the market value. Thus, our project goal is to build a predictive model by using players' performance score on soccer.

III. Data Description

The data we are using is called the *FIFA 2018 Complete Player.* It is the second-hand data set that we acquired from *Kaggle* website. It listed all the players who participated in FIFA 2018. The information includes but is not limited to player's names, nationalities, wages, belonging clubs as well as their scores on athletics performance.

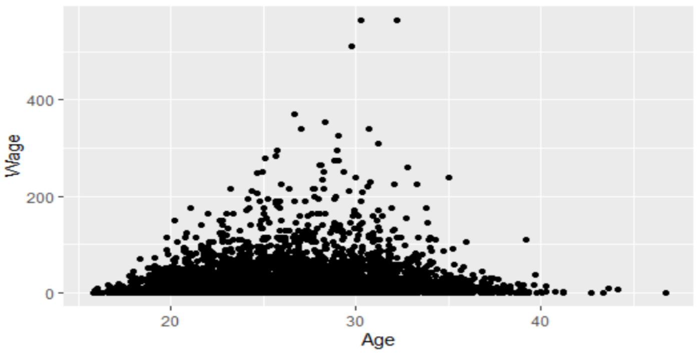
It has 17,981 records with 75 columns in the data. After using R code to explore the data, we found out that there are around 200 records that include missing values and some columns contained currency symbols. In order to accurately analyze the data, our group removed all the records with missing values and eliminated the currency symbols in the columns. Furthermore, we deleted columns that were not useful in our analysis, including but not limited to: ID (has 2 columns), Photo, Flag, Club Logo, and Preferred Position.

Next, our group did exploratory data analysis to better understand our data. We created different charts to gain insight of the data. First, we created a density chart of the soccer ball players' age (see Chart_a- Density of the Age). From the chart, we see that the age of the players is skewed to the right. Majority of the players who participated in the FIFA 2018 are in their 20's, especially 24 years old.



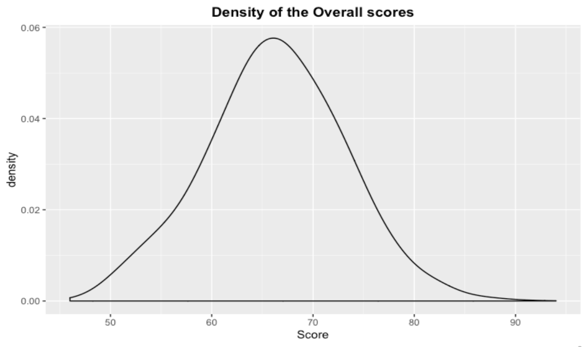
Chart_a- Density of the Age

We also created the following scatter plot (Scatter_Plot_a- Wage and Age). This scatter plot shows the relationship between age and the wage. It is shown that there is no strong relationship between age and wage through the graph. However, we can see that there are only few players earns high wages and the majority of the wage that player earns is less than 200K.



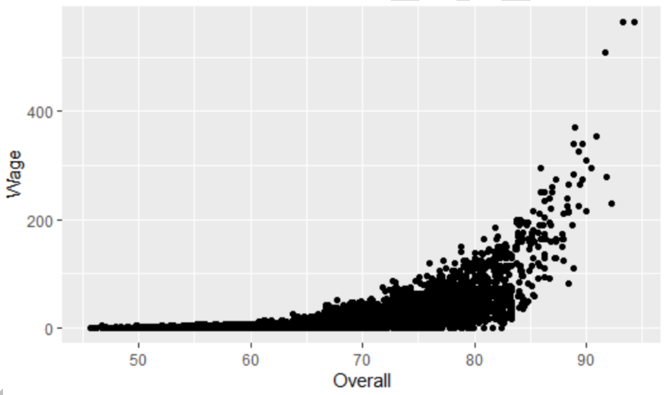
Scatter_Plot_a- Wage and Age

The third chart (Chart_b- Density of the Overall Scores) shows the density chart of the overall score. The overall score of the soccer player is normally distributed and it is around the score of 66.



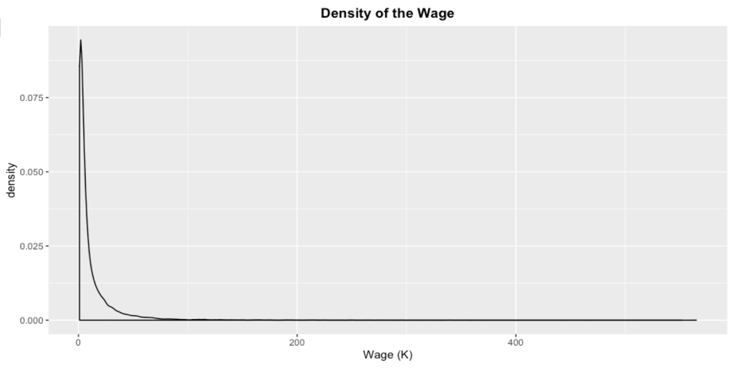
Chart_b- Density of the Overall Scores

Scatter plot below shows overall score and wage. As the overall score increases, the wage also increases. This is reasonable because the higher overall score the player got, the more competitive he is. Therefore, the players' wages are higher.



Scatter_Plot_b- Wage and Score

The last chart (Chart_c- Density of the Wage) is the density score of wage. It can be seen that the wage is not normally distributed. The density area is toward the bottom.

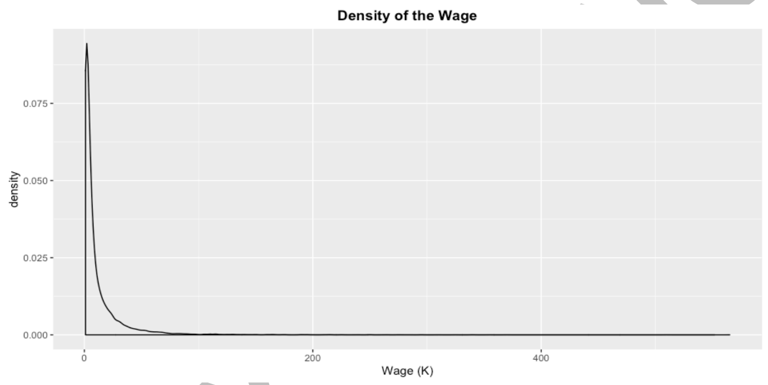


Chart_c- Density of the Wage

IV. Challenges and solution

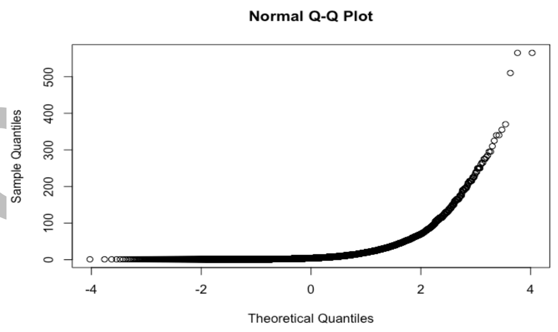
The first challenge we faced was the skewed distribution of our dataset. The distribution of wage is highly skewed, so it was difficult for us to run the classification analysis. There are many reasons caused the skewness. Two major possible reasons are whether a player is substitution or first-line player and which league the play belongs to. It is expected that there are wage differences between substitution players and first-line players. It is also expected that there are wages' payment differences in different league.

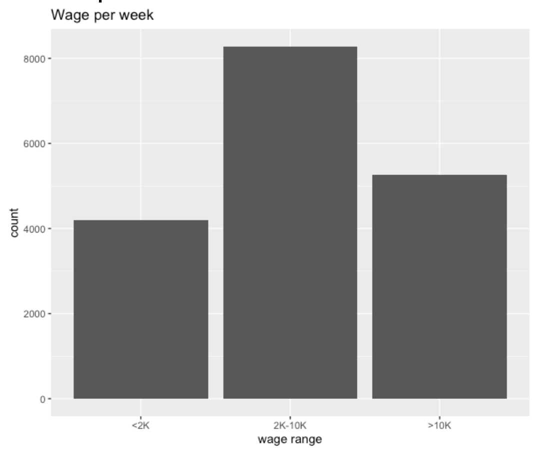
In order to solve skewed distribution of the wage, we decided to categorize it. We wanted to categorize the players by their experience. However, from the dataset, we cannot distinguish players who are substitution or experienced. Therefore, we tried to use leagues of football to categorize the players. In the dataset, we only had nationality and club information and the dataset didn't have the league information. So, the nationality and club data can neither be used. Hence, we didn't use league to categorize the data either.



Chart_c - Density of the Wage

Thus, we considered categorized wage data. Because the wage distribution is not normal, we hope to separate the data in different range levels to help the analysis. By an iterative process through using R, we found out that distributing the wage in 3 buckets would satisfy our goal of making wage dataset to a normal distribution. The 3 buckets are the wage under 2k per week, between 2K-10K per week and one larger than 10k per week.



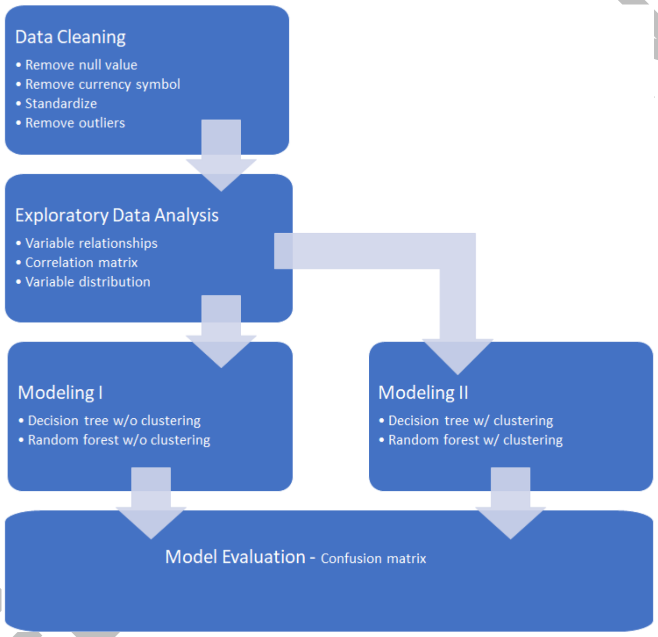


Q-Q Plot Bar_Chart_a - Wage per Week

The second challenge is how to improve the accuracy rate of the model. In our first trial of the prediction analysis, the accuracy rate is poor so we have tried several other methods, such as bucket adjusting, PCA, Neural Network, and etc., but they didn't work well. Thus, we take a deeper look on the relationship between variables. We find out that some of the variables are uncorrelated. They created noise of the model and lead to low accuracy rate. Moreover, we checked range of bucket and find out that range of the bucket actually affected our accuracy rate.

In order to improve the accuracy rate of the model, we decided to remove the noise of the model and divided our wage into two buckets instead of three for further analysis. We tested out that multi-algorithms (cluster and decision tree, random forest and cluster) can successfully solve the problem of low accuracy rate. Thus, we are using the multi-algorithms methods to conduct our analysis.

V. BI Model


Data Cleaning
• Remove null value
• Remove currency symbol
• Standardize
• Remove outliers
Exploratory Data Analysis
• Variable relationships
• Correlation matrix
• Variable distribution
Modeling I
• Decision tree w/o clustering
• Random forest w/o clustering
Modeling II
• Decision tree w/ clustering
• Random forest w/ clustering
Model Evaluation - Confusion matrix

Data Cleaning

Before starting modeling, we removed all the NULL values in our datasets as well as eliminated useless columns which are not related to our research, such as the columns contain flag information, photo URL and so on.

What's more, we found out that the wage under two thousand dollars per week is not the target for us to do the modeling, so we removed them. We then grouped the value of wage into 2 buckets. One is smaller than ten thousand dollars per week as "<10K" and rest is greater than 10K shown as ">10K".

Code:

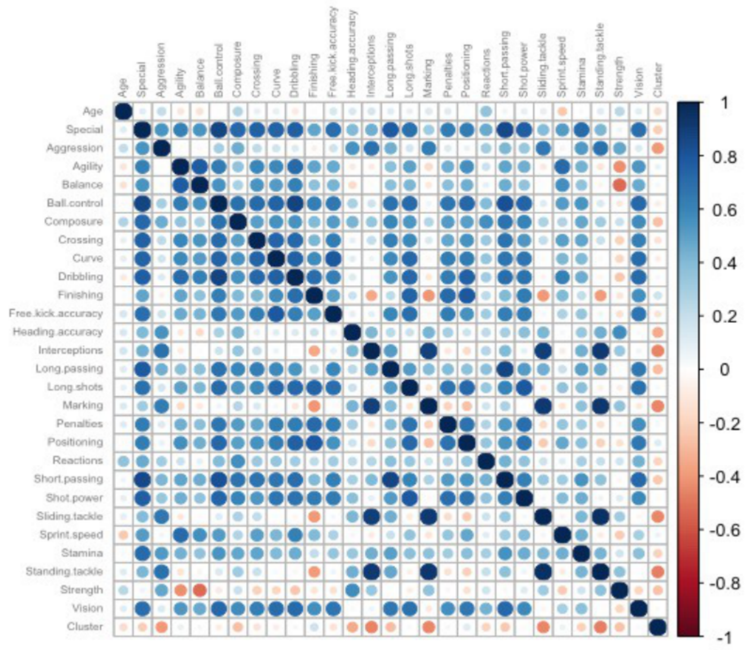
```
#Remove useless columns
useless_columns = c(1, 4, 6, 8, 10, 48:75)
df = df.raw[, -useless_columns]
df[, 10:41] = sapply(df[, 10:41], as.numeric)

#Remove currency symbols
df$Value = as.numeric(str_extract(df$Value, "\\d+\\.*\\d*"))
df$Wage = as.numeric(str_extract(df$Wage, "\\d+\\.*\\d*"))

# Remove the rows with wage that is smaller than 2K/week
df = df[-which(df$Wage <= 2), ]

#Create 2 buckets for wage
df.num = select_if(df, is.numeric)
df.num$WageRange = ifelse(df$Wage < 10,"<10K", ">10K")
df.num$WageRange = factor(df.num$WageRange, levels = c("<10K",">10K"))
```

Exploratory Data Analysis

We did the exploratory data analysis to better understand the insight of the data. We created correlation matrix between variables. We also created the density charts of age and wage, scatter plots of age and wage as well as scatter plots of overall score and wage.

Code:

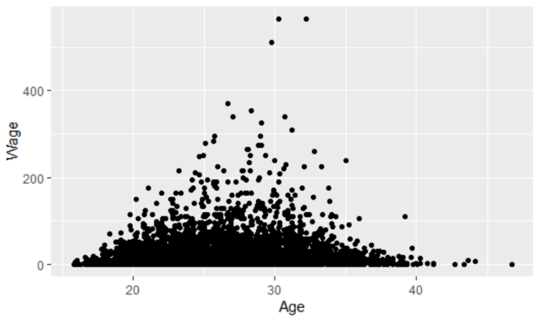
```
#Run correlation matrix
library("corrplot")
cor = cor(df.num[, c(1,4:36)], method = "pearson", use = "complete.obs")
corrplot(cor, tl.cex= 0.5, tl.col = "gray50")
```

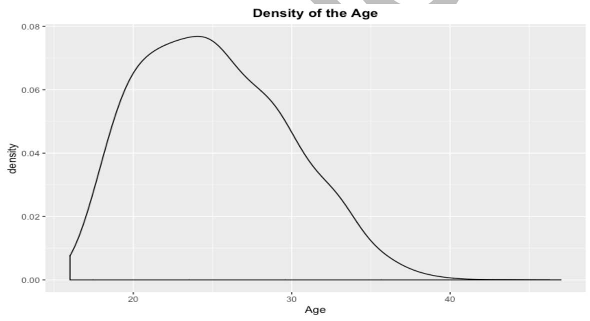


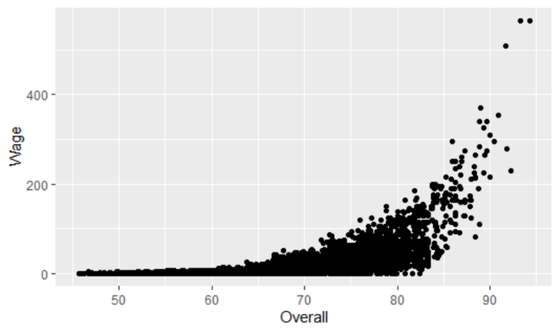
Correlation Matrix

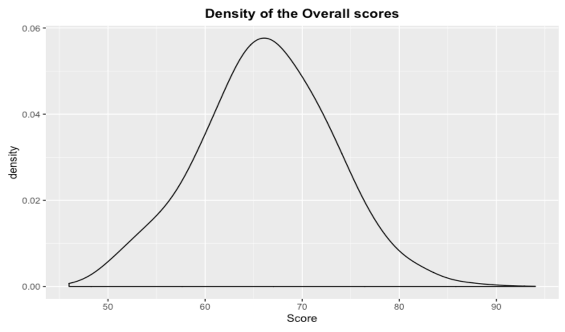
Code:

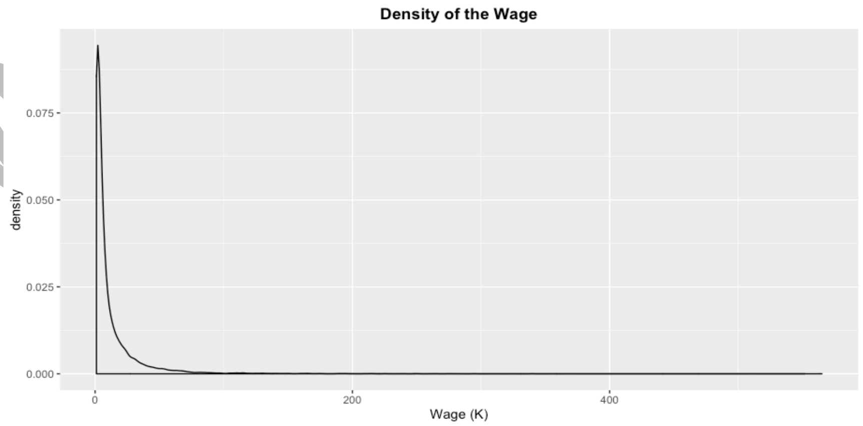
```
#Plot relationship between Age and Wage
age_nationality = ggplot(df.num, aes(df$Age, df$Wage))
age_nationality + geom_point(position = "jitter") + xlab("Age") + ylab("Wage")

#Plot relationship between Overall and Wage
overall_wage = ggplot(df.num, aes(df$Overall, df$Wage))
overall_wage + geom_point(position = "jitter") + xlab("Overall") + ylab("Wage")

#Plot density of Wage
plt = ggplot(df, aes(df$Wage))
plt + geom_density(adjust = 1) +
  labs(title = "Density of Wage", x = "Wage (K)")
theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```









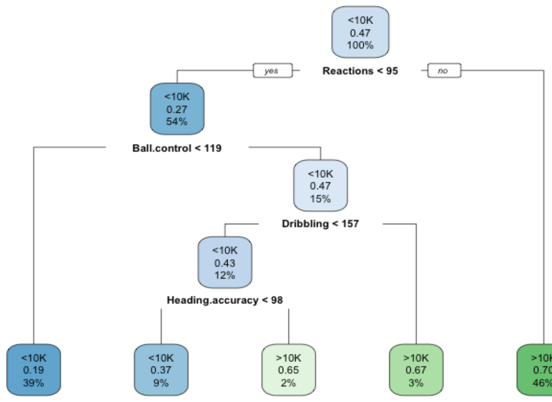


Modeling I

First of all, we divided our raw data into 70% training data and 30% validation data. After doing this, we used all the variables to predict Wage status.

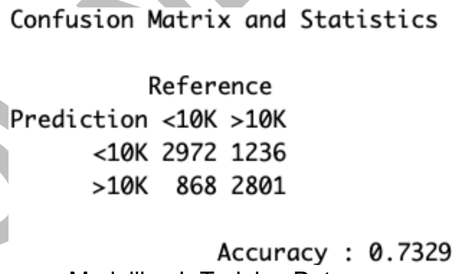
Code:

```
set.seed(321)
train = sample((1:nrow(df)), 0.7*nrow(df))
df.train = df.num[train, ]  # training data set
df.valid = df.num[-train, ] # validation data set

df.tree = rpart(WageRange ~. , data = df.train, method = "class",
             parms = list(split = 'information'))
rpart.plot(df.tree, extra = "auto")

train_df.pred = predict(df.tree, type = "class")
confusionMatrix(df.train$WageRange, train_df.pred)
valid_df.pred = predict(df.tree, df.valid, type = "class")
confusionMatrix(df.valid$WageRange, valid_df.pred)
```



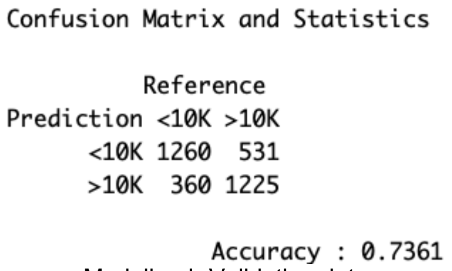
Modeling I- Decision Tree

Then, we computed the confusion matrices based on our training data and validation data as following:



```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K 2972 1236
      >10K  868 2801

               Accuracy : 0.7329
```

Modelling I- Training Data



```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K 1260  531
      >10K  360 1225

               Accuracy : 0.7361
```

Modeling I- Validation data

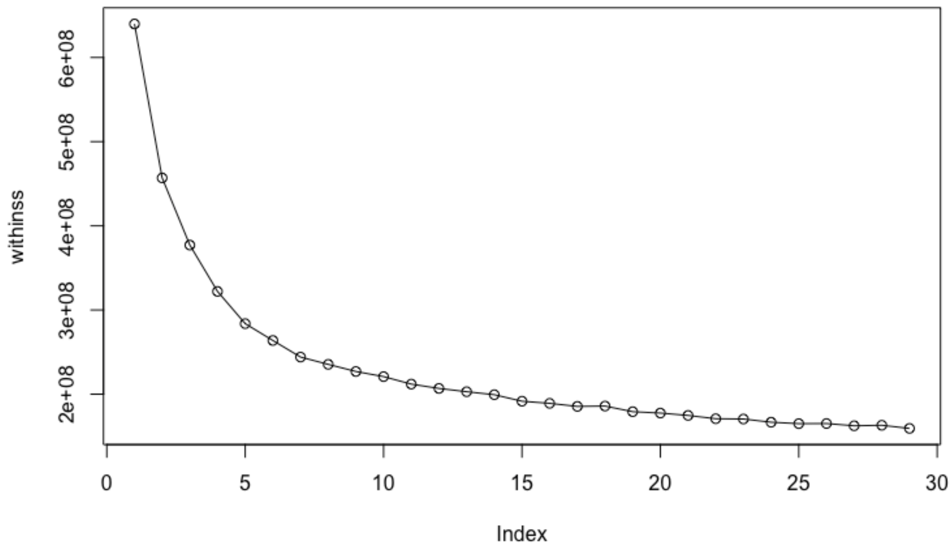
According to the confusion matrices, we can see that the training data and validation data have an accuracy rate at approximately 73% and 74% respectively. There is not a large difference between these two. However, the accuracy rate is lower than what we expected. We decided to cluster the data first and then apply to the model to improve the accuracy level.

Modeling II

To start with modeling II, we firstly plot a within line chart to help us determine how many clusters should we divided into.

Code:

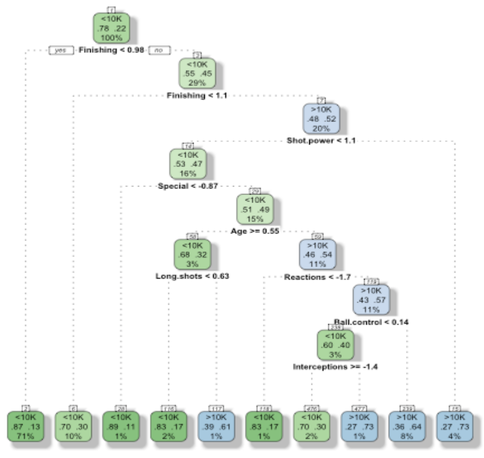
```
####### How many clusters? #######
withinss = c()
for (i in 2:30){
   k = kmeans(df.num[, 2:ncol(df.num)], i)
   withinss = c(withinss, k$tot.withinss)
}
plot(withinss)
lines(withinss)

# Choose 6 clusters
df.cl = kmeans(scale(df.num[2:ncol(df.num)]), 6)
```

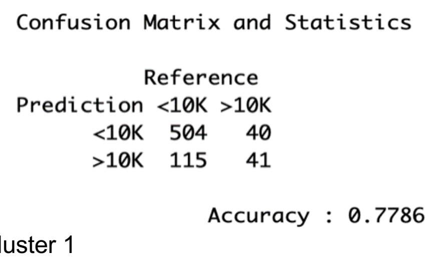


From the above line chart, we see that begin in within 6, the line becomes flatter which indicates that 6 clusters will be a good number for us to do the clustering.

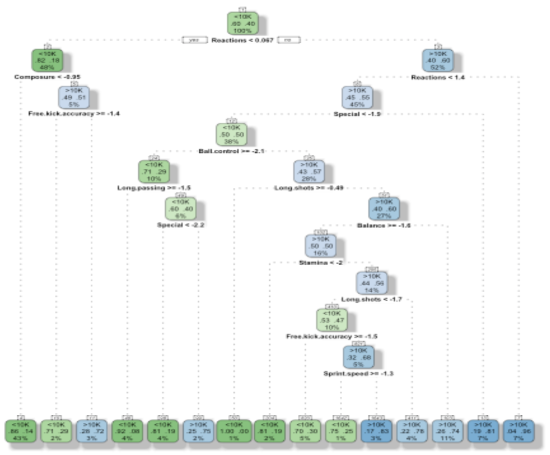
```
####### Scaling #######
df.num = cbind(df.num, Cluster = df.cl$cluster)
maximum = sapply(df.num[, 2:30], max)
minimum = sapply(df.num[, 2:30], min)
df.num[, 2:29] = as.data.frame(sapply(df.num[, 2:29], scale))
set.seed(999)

#### tree 1 ####
df.cl1 = df.num[df.num$Cluster == 1, ]
train = sample((1:nrow(df.cl1)), 0.7*nrow(df.cl1))
df.cl1.train = df.cl1[train, ]  # training data set
df.cl1.valid = df.cl1[-train, ] # validation data set
df.tree1 = rpart(WageRange ~. , data = df.cl1.train, method = "class",
              parms = list(split = 'information'))
fancyRpartPlot(df.tree1, caption = "")
pred1 = predict(df.tree1, type = "class")   # training
confusionMatrix(df.cl1.train$WageRange, pred1)
pred1 = predict(df.tree1, df.cl1.valid, type = "class") # validation
confusionMatrix(df.cl1.valid$WageRange, pred1)
printcp(df.tree1)

#### tree 2 ####
df.cl2 = df.num[df.num$Cluster == 2,]
train = sample((1:nrow(df.cl2)), 0.7*nrow(df.cl2))
df.cl2.train = df.cl2[train, ]  # training data set
df.cl2.valid = df.cl2[-train, ] # validation data set
df.tree2 = rpart(WageRange ~. , data = df.cl2.train, method = "class",
               parms = list(split = 'information'))
fancyRpartPlot(df.tree2, caption = "")
pred2 = predict(df.tree2, type = "class")   # training
confusionMatrix(df.cl2.train$WageRange, pred2)
pred2 = predict(df.tree2, df.cl2.valid, type = "class") # validation
confusionMatrix(df.cl2.valid$WageRange, pred2)

#### tree 3 ####
df.cl3 = df.num[df.num$Cluster == 3,]
train = sample((1:nrow(df.cl3)), 0.7*nrow(df.cl3))
df.cl3.train = df.cl3[train, ]
df.cl3.valid = df.cl3[-train, ]
df.tree3 = rpart(WageRange ~. , data = df.cl3.train, method = "class",
               parms = list(split = 'information'))
fancyRpartPlot(df.tree3, caption = "")
pred3 = predict(df.tree3, type = "class")   # training
confusionMatrix(df.cl3.train$WageRange, pred3)
pred3 = predict(df.tree3, df.cl3.valid, type = "class") # validation
confusionMatrix(df.cl3.valid$WageRange, pred3)

#### tree 4 ####
df.cl4 = df.num[df.num$Cluster == 4,]
train = sample((1:nrow(df.cl4)), 0.7*nrow(df.cl4))
```

```
df.cl4.train = df.cl4[train, ]
df.cl4.valid = df.cl4[-train, ]
df.tree4 = rpart(WageRange ~. , data = df.cl4.train, method = "class",
               parms = list(split = 'information'))
fancyRpartPlot(df.tree4, caption = "")
pred4 = predict(df.tree4, type = "class")   # training
confusionMatrix(df.cl4.train$WageRange, pred4)
pred4 = predict(df.tree4, df.cl4.valid, type = "class") # validation
confusionMatrix(df.cl4.valid$WageRange, pred4)

#### tree 5 ####
df.cl5 = df.num[df.num$Cluster == 5, ]
train = sample((1:nrow(df.cl5)), 0.7*nrow(df.cl5))
df.cl5.train = df.cl5[train, ]
df.cl5.valid = df.cl5[-train, ]
df.tree5 = rpart(WageRange ~. , data = df.cl5.train, method = "class",
               parms = list(split = 'information'))
fancyRpartPlot(df.tree5, caption = "")
pred5 = predict(df.tree5, type = "class")   # training
confusionMatrix(df.cl5.train$WageRange, pred5)
pred5 = predict(df.tree5, df.cl5.valid, type = "class") # validation
confusionMatrix(df.cl5.valid$WageRange, pred5)

#### tree 6 ####
df.cl6 = df.num[df.num$Cluster == 6,]
train = sample((1:nrow(df.cl6)), 0.7*nrow(df.cl6))
df.cl6.train = df.cl6[train, ]
df.cl6.valid = df.cl6[-train, ]
df.tree6 = rpart(WageRange ~. , data = df.cl6.train, method = "class",
               parms = list(split = 'information'))
fancyRpartPlot(df.tree6, caption = "")
pred6 = predict(df.tree6, type = "class")   # training
confusionMatrix(df.cl6.train$WageRange, pred6)
pred6 = predict(df.tree6, df.cl6.valid, type = "class") # validation
confusionMatrix(df.cl6.valid$WageRange, pred6)
```



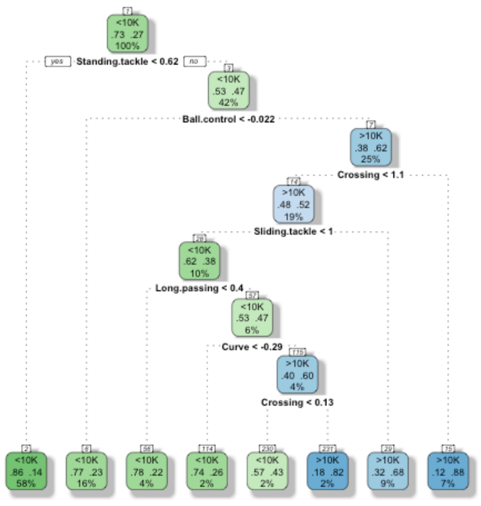

```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K  504   40
      >10K  115   41

               Accuracy : 0.7786
```

Modeling II- Cluster 1



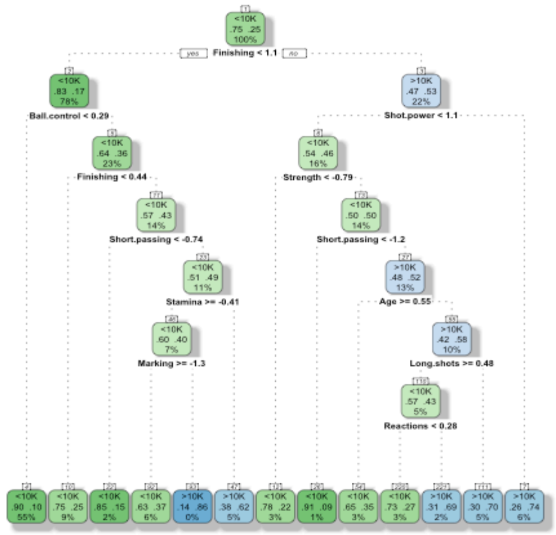
```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K  145   19
      >10K   71   60

               Accuracy : 0.6949
```

Modeling II- Cluster 2



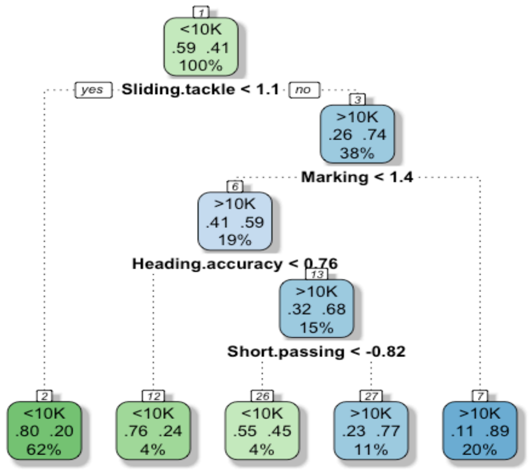
```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K  384   27
      >10K   94   64

               Accuracy : 0.7873
```

Modeling II- Cluster 3



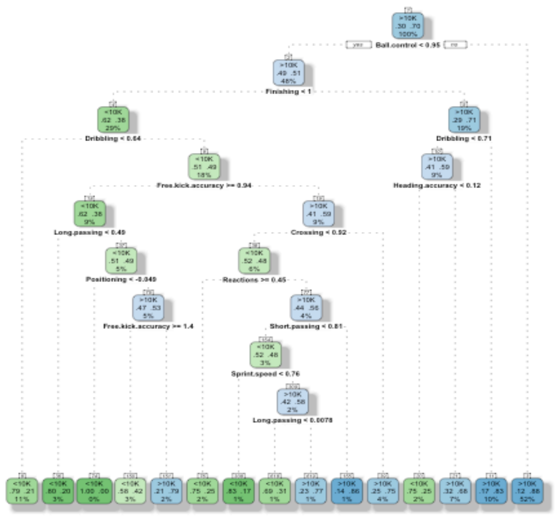
```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K  466   62
      >10K  102   64

               Accuracy : 0.7637
```

Modeling II- Cluster 4



```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K  299   28
      >10K   94  138

               Accuracy : 0.7818
```

Modeling II- Cluster 5



```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K   98  111
      >10K   64  402

               Accuracy : 0.7407
```

Modeling II- Cluster 6

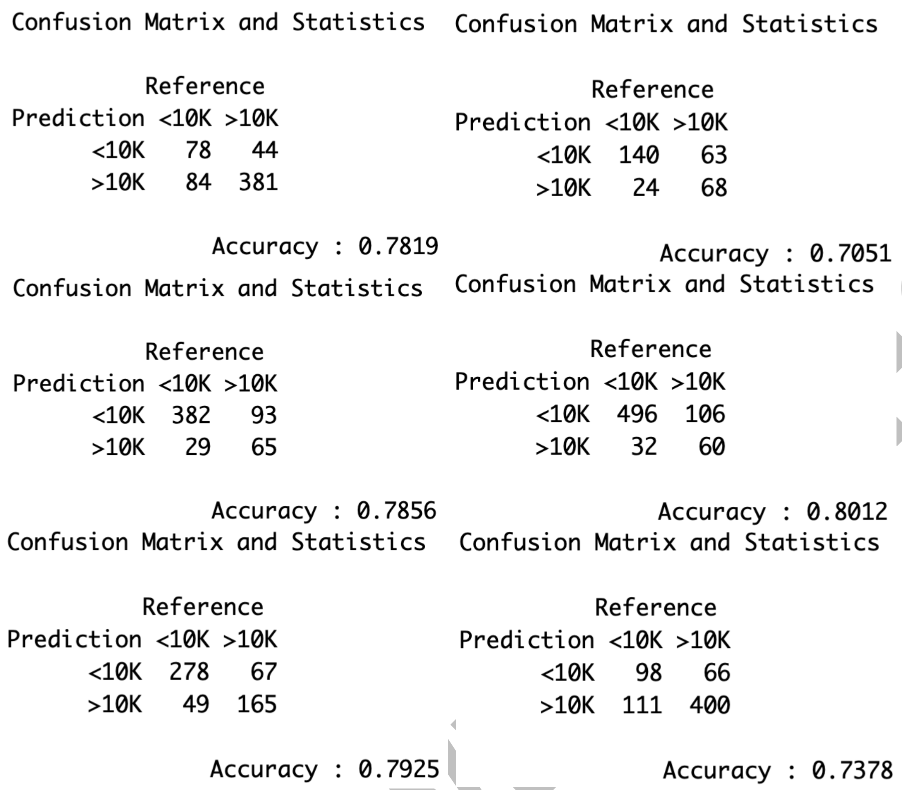
For each cluster, we grew a decision tree. As we can see from above, the accuracy rate for decision tree for tree 1 is 77.9%, which is higher than the Model 1 decision tree, 73.6%, when we didn’t cluster the data.

Theoretically, more trees get a better decision. It’s more reliable to make a decision based on more than one opinion. Therefore, we start to think about whether if the random forest will improve the accuracy.

Code:

```
library("randomForest")

#### random forest 1 ####
df.rf1 = randomForest(WageRange ~ ., data = df.cl1.train, ntree = 10000)
rf.pred1 = predict(df.rf1, type = "class")  # training
confusionMatrix(rf.pred1, df.cl1.train$WageRange)
```

```
rf.pred1 = predict(df.rf1, df.cl1.valid, type = "class")    # validation
confusionMatrix(rf.pred1, df.cl1.valid$WageRange)

#### random forest 2 ####
df.rf2 = randomForest(WageRange ~ ., data = df.cl2.train, ntree = 10000)
rf.pred2 = predict(df.rf2, type = "class")  # training
confusionMatrix(rf.pred2, df.cl2.train$WageRange)
rf.pred2 = predict(df.rf2, df.cl2.valid, type = "class")    # validation
confusionMatrix(rf.pred2, df.cl2.valid$WageRange)

#### random forest 3 ####
df.rf3 = randomForest(WageRange ~ ., data = df.cl3.train, ntree = 10000)
rf.pred3 = predict(df.rf3, type = "class")  # training
confusionMatrix(rf.pred3, df.cl3.train$WageRange)
rf.pred3 = predict(df.rf3, df.cl3.valid, type = "class")    # validation
confusionMatrix(rf.pred3, df.cl3.valid$WageRange)

#### random forest 4 ####
df.rf4 = randomForest(WageRange ~ ., data = df.cl4.train, ntree = 10000)
rf.pred4 = predict(df.rf4, type = "class")  # training
confusionMatrix(rf.pred4, df.cl4.train$WageRange)
rf.pred4 = predict(df.rf4, df.cl4.valid, type = "class")    # validation
confusionMatrix(rf.pred4, df.cl4.valid$WageRange)

#### random forest 5 ####
df.rf5 = randomForest(WageRange ~ ., data = df.cl5.train, ntree = 10000)
rf.pred5 = predict(df.rf5, type = "class")  # training
confusionMatrix(rf.pred5, df.cl5.train$WageRange)
rf.pred5 = predict(df.rf5, df.cl5.valid, type = "class")    # validation
confusionMatrix(rf.pred5, df.cl5.valid$WageRange)

#### random forest 6 ####
df.rf6 = randomForest(WageRange ~ ., data = df.cl6.train, ntree = 10000)
rf.pred6 = predict(df.rf6, type = "class")  # training
confusionMatrix(rf.pred6, df.cl6.train$WageRange)
rf.pred6 = predict(df.rf6, df.cl6.valid, type = "class")    # validation
confusionMatrix(rf.pred6, df.cl6.valid$WageRange)
```



```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K   78   44
      >10K   84  381

               Accuracy : 0.7819
```

```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K  140   63
      >10K   24   68

               Accuracy : 0.7051
```

```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K  382   93
      >10K   29   65

               Accuracy : 0.7856
```

```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K  496  106
      >10K   32   60

               Accuracy : 0.8012
```

```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K  278   67
      >10K   49  165

               Accuracy : 0.7925
```

```
Confusion Matrix and Statistics

          Reference
Prediction <10K >10K
      <10K   98   66
      >10K  111  400

               Accuracy : 0.7378
```

Figure. (Left to right, top to buttom) We respectively demonstrate the accuracy rate across random forest with six different clusters.

In Cluster 1, after putting clustering into random forest, we observe that the accuracy slightly improves. Furthermore, if we shed light on the confusion matrix above, the model successfully reduces the false true proportion, in the sense that model becomes more conservative, thus can cost down the risk of overestimation. By comparing the models, the random forest with clustering gives us the best prediction analysis on wage.

VI. Finding and Managerial Implications

Based on our model, we are able to predict player's wage by using the soccer ball players' previous performance score. It would have business implications in below scenarios.

The model could help a league or company dream team within budget. Assume that a company want to start a soccer team with a limited budget. Our model can provide recommendation that which players could be considered within the budget. For example, a company can using the players' previous performance to predict his/her wage and got a picture how much expense on each player. The company then can decide whether it wants the player or not based on his/her wage expense and performance.

The model could also explore potential players. Suppose one company already owns a soccer team and he/she want(s) to poach players from other company to strengthen the team. He/she can use the model. The company can apply the players' previous performance score into the model and generated the players wage level and see which player is underpaid/underestimated in other company or team. The company could then hire the players who underestimated into the team.

VII. Conclusion

At the beginning of analyzing (Modeling I), we used all the variables without clustering them to predict the players' wages. We find out that it has caused a critical problem because the players in the same scoring performance might have large difference on their wage. In order to solve the issue, we decided to use k-means (Modeling II) to cluster the data into 6 clusters. It makes the scoring performance more consistent in each cluster. Furthermore, we analyzed the decision tree and random forest for each cluster separately. By doing so, it is shown that the scoring performance and the wage prediction in each cluster becomes more accurate.

Comparing Modeling I with Modeling II, we conclude that Modeling II (random forest with clustering) is more accurate and more reasonable for the following reasons. First, k-means clustering significantly improved the accuracy rate. Second, Model II reduced the proportion on overestimated cases. Last but not least, clustering players before modeling conforms the real-world scenarios better.

VIII. Reference

Shrivastava, Aman. “FIFA 18 Complete Player Dataset.” *Kaggle*, 30 Oct. 2017, https://www.kaggle.com/thec03u5/fifa-18-demo-player-dataset#PlayerPlayingPositionData.csv

Steinberg, Leigh. “CHANGING THE GAME: The Rise of Sports Analytics.” Forbes, Forbes Magazine, 18 Aug. 2015, https://www.forbes.com/sites/leighsteinberg/2015/08/18/changing-the-game-the-rise-of-sports-analytics/#6efc002c4c1f.