BUS 212a S1 – Analyzing Big Data II, Spring 2019

Project Assignment 3: Classification

Team:

Data Incubator

- **Data Preprocessing**

In this assignment, we applied classification analysis on FIFA 2019 data set downloaded from Kaggle. By classifying players as “Strikers” or “Non-strikers” , we wanted to help players find their best position on the field and to help the coaches make customized training plans for their players.

We chose 30 columns from the original dataset to implement classification analysis. The variables include “Crossing”, “Finishing”, “Heading Accuracy”, “Short Passing”, “Volleys”, “Dribbling”, “Curve”, “FK Accuracy”, “Long Passing”, “Ball Control” and so on. Most of the variables are measurements on the player’s specific football skills and are centesimal. The only exception was the “Release Clause”, so we changed it to centesimal to keep all the variables at the same scale.

We mutated a new variable “ST” for classification. “ST” equals 1 for all players in “Striker” position, and “ST” equals 0 for all players that are not strikers. Then, we transformed the “ST” from numeric variable to factor in order to apply classification. However, this caused another problem. The total number of strikers was 1924, while the total number of non-strikers was 12820. The imbalanced dataset tends to provide unsatisfactory results, because the majority tends to dominate over the minority. In order to balance the data, we oversampled the strikers and under-sampled the non-strikers randomly, and our sample size became 4000.

Then we partitioned our dataset to training data and valid data. We assigned 60% of the original data to the training data and 40% to the valid data.

- **Logistic Regression**

(1) Initial Model

We first estimate a logistic regression model using all the variables other than position.

The stepAIC function automatically generates the best model in terms of AIC. We find that the coefficients for a few variables are not statistically significant. Therefore, we adjust the model by putting in significant variables. All the coefficients for the adjusted model(logit.reg) are statistically significant.

```
Call:
glm(formula = ST ~ Crossing + Finishing + HeadingAccuracy + ShortPassing +
    Volleys + BallControl + Strength + Marking + StandingTackle +
    Release.Clause, family = "binomial", data = train.df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.5271  -0.1145  -0.0080   0.2544   3.4978

Coefficients:
                Estimate Std. Error z value             Pr(>|z|)
(Intercept)     -1.601732   0.949881  -1.686             0.091749 .
Crossing        -0.049275   0.011676  -4.220     0.00002440834980 ***
Finishing        0.166753   0.019691   8.468 < 0.0000000000000002 ***
HeadingAccuracy  0.060147   0.011852   5.075     0.00000038762692 ***
ShortPassing    -0.071513   0.021568  -3.316             0.000914 ***
Volleys          0.046719   0.013438   3.477             0.000508 ***
BallControl     -0.093956   0.025900  -3.628             0.000286 ***
Strength         0.033120   0.009545   3.470             0.000521 ***
Marking         -0.034651   0.008732  -3.968     0.00007241999551 ***
StandingTackle  -0.062114   0.009032  -6.877     0.00000000000609 ***
Release.Clause  -0.044286   0.016610  -2.666             0.007671 **
---
```

The out-of-sample accuracy is 0.9388, which is pretty good.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 766  28
         1  70 736


               Accuracy : 0.9388
                 95% CI : (0.9259, 0.95)
    No Information Rate : 0.5225
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.8775
 Mcnemar's Test P-Value : 0.00003449

            Sensitivity : 0.9163
            Specificity : 0.9634
         Pos Pred Value : 0.9647
         Neg Pred Value : 0.9132
             Prevalence : 0.5225
         Detection Rate : 0.4788
   Detection Prevalence : 0.4963
      Balanced Accuracy : 0.9398
```

(2) Model with Interaction Terms

To discover potential models generate higher out-of-sample accuracy, we try to add interaction and quadratic terms. We first consider all the potential interaction term and then keeps the ones that are significant (Crossing*HeadingAccuracy, Finishing*HeadingAccuracy). However, we find that the coefficient for HeadingAccuracy in model logit.reg1 is not significant. Therefore, we drop HeadingAccuracy and rebuild the model (logit.reg1.update). After that, all the coefficients for the variables are statistically significant.

```
Call:
glm(formula = ST ~ Crossing + Finishing + ShortPassing + Volleys +
    BallControl + Strength + Marking + StandingTackle + Release.Clause +
    Crossing:HeadingAccuracy + Finishing:HeadingAccuracy, family = "binomial",
    data = train.df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2079  -0.0959  -0.0053   0.2546   3.7411

Coefficients:
                            Estimate Std. Error z value           Pr(>|z|)
(Intercept)                1.2223818  1.2010339   1.018             0.30879
Crossing                  -0.3619749  0.0535446  -6.760 0.0000000000137756
Finishing                  0.3861999  0.0507705   7.607 0.0000000000000281
ShortPassing              -0.0710218  0.0218887  -3.245             0.00118
Volleys                    0.0512566  0.0139564   3.673             0.00024
BallControl               -0.0884777  0.0271529  -3.258             0.00112
Strength                   0.0388463  0.0098725   3.935 0.0000832624518698
Marking                   -0.0353909  0.0089472  -3.956 0.0000763631208549
StandingTackle            -0.0638542  0.0092727  -6.886 0.0000000000057263
Release.Clause            -0.0454443  0.0160901  -2.824             0.00474
Crossing:HeadingAccuracy   0.0051789  0.0008288   6.248 0.0000000004146575
Finishing:HeadingAccuracy -0.0036508  0.0007336  -4.977 0.0000006472957107

(Intercept)
Crossing                  ***
Finishing                 ***
ShortPassing              **
Volleys                   ***
BallControl               **
Strength                  ***
Marking                   ***
StandingTackle            ***
Release.Clause            **
Crossing:HeadingAccuracy  ***
Finishing:HeadingAccuracy ***
```

Also, the out-of-sample accuracy increases to 0.9394. Therefore, we decide to add interaction terms of Crossing*HeadingAccuracy and Finishing*HeadingAccuracy to the model.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 764  25
         1  72 739

               Accuracy : 0.9394
                 95% CI : (0.9265, 0.9506)
    No Information Rate : 0.5225
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.8788
 Mcnemar's Test P-Value : 0.000003003

            Sensitivity : 0.9139
            Specificity : 0.9673
         Pos Pred Value : 0.9683
         Neg Pred Value : 0.9112
             Prevalence : 0.5225
         Detection Rate : 0.4775
   Detection Prevalence : 0.4931
      Balanced Accuracy : 0.9406
```

(3) Model with (Polynomial)Quadratic Terms

We also consider all potential quadratic terms and keeps the ones that are significant (square of StandingTackle, square of Release.Clause). However, the accuracy of the model decreases to 0.9356.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 765  32
         1  71 732

               Accuracy : 0.9356
                 95% CI : (0.9225, 0.9472)
    No Information Rate : 0.5225
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.8713
 Mcnemar's Test P-Value : 0.0001809

            Sensitivity : 0.9151
            Specificity : 0.9581
         Pos Pred Value : 0.9598
         Neg Pred Value : 0.9116
             Prevalence : 0.5225
         Detection Rate : 0.4781
   Detection Prevalence : 0.4981
      Balanced Accuracy : 0.9366

       'Positive' Class : 0
```

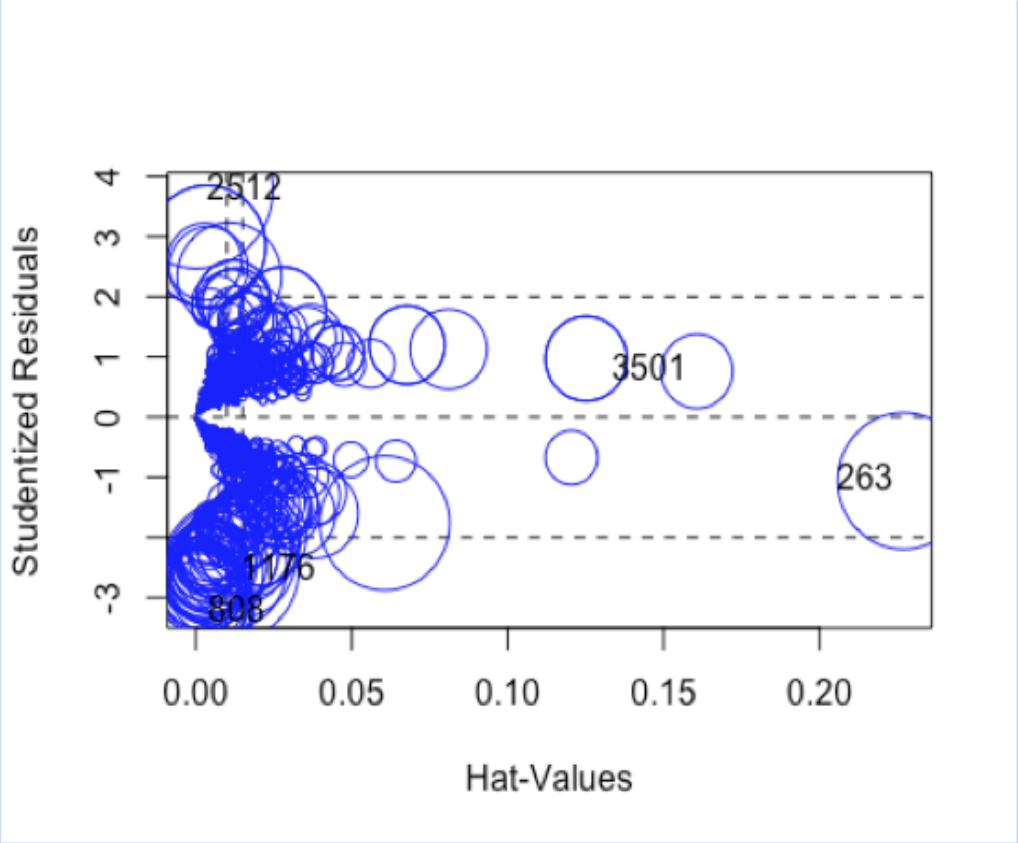
Since adding quadratic terms will actually harm the out-of-sample accuracy of the model, we decide to not include quadratic terms in the model.

Based on the above discussion, our best logistic model will be the one has Crossing, Finishing, ShortPassing, Volleys, BallControl, Strength, Marking, StandingTackle, Release.Clause and two interaction terms -- Crossing*HeadingAccuracy, Finishing*HeadingAccuracy as predictors.

(4) Remove Outliers and Determine Best Model

We use outlierTest and influencePlot function to detect potential outliers in the model. After running the outlier test, we find one outlier. The influncepPlot function helps us further identify 4 more potential outliers with large standardized residuals. We delete those five outliers and rebuild our model (logit.reg.new). This will be our final logistic regression model.

```r
outlierTest(logit.reg1.update)

## No Studentized residuals with Bonferonni p < 0.05
## Largest |rstudent|:
##      rstudent unadjusted p-value Bonferonni p
## 2512 3.782275          0.0001554      0.37296

influencePlot(logit.reg1.update, col = "blue")
```




```
##         StudRes          Hat       CookD
## 2512  3.782275 0.0002829419 0.025797844
## 1176 -2.549734 0.0110873592 0.020626860
## 263  -1.061742 0.2269023420 0.019211717
## 3501  0.761881 0.1606524123 0.005682384
## 808  -3.227030 0.0007193466 0.010244249
```

The out-of-sample accuracy for our final model is 0.9394, which is the same as the one without outliers removal. Therefore, we could conclude the removed outliers basically have little effect on the model.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 764  25
         1  72 739

               Accuracy : 0.9394
                 95% CI : (0.9265, 0.9506)
    No Information Rate : 0.5225
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.8788
 Mcnemar's Test P-Value : 0.000003003

            Sensitivity : 0.9139
            Specificity : 0.9673
         Pos Pred Value : 0.9683
         Neg Pred Value : 0.9112
             Prevalence : 0.5225
         Detection Rate : 0.4775
   Detection Prevalence : 0.4931
      Balanced Accuracy : 0.9406
```

The out-of-sample accuracy for our final model is 0.9394. The sensitivity for our final model is 0.9139 and the specificity is 0.9673. All those metrics indicated that our final model generalizes well to validation data and has a fairly high out-of-sample predictability.

The following is the result of our best logistic regression model:

```r
# rebuild model
logit.reg.new <- update(logit.reg1.update, subset = c(-2512,-1176,-263,-3501,-808))
data.frame(summary(logit.reg.new)$coefficients, odds = exp(coef(logit.reg.new)))
```

```
##                                Estimate   Std..Error   z.value
## (Intercept)                1.217520551 1.2007715358  1.013949
## Crossing                  -0.362202468 0.0535411924 -6.764931
## Finishing                  0.386440494 0.0507746248  7.610898
## ShortPassing              -0.071027676 0.0218831196 -3.245775
## Volleys                    0.051183369 0.0139548638  3.667780
## BallControl               -0.088379201 0.0271461204 -3.255684
## Strength                   0.038846364 0.0098697818  3.935889
## Marking                   -0.035344506 0.0089461688 -3.950798
## StandingTackle            -0.063837875 0.0092717935 -6.885170
## Release.Clause            -0.045421833 0.0160871482 -2.823486
## Crossing:HeadingAccuracy   0.005183668 0.0008288140  6.254320
## Finishing:HeadingAccuracy -0.003655507 0.0007336354 -4.982730
##                                          Pr...z..      odds
## (Intercept)               0.31060730592821395035 3.3787998
## Crossing                  0.00000000001333728203 0.6961414
## Finishing                 0.00000000000002721979 1.4717328
## ShortPassing              0.00117131502183407126 0.9314361
## Volleys                   0.00024466563785086547 1.0525159
## BallControl               0.00113119405418669926 0.9154137
## Strength                  0.00008288933226883908 1.0396107
## Marking                   0.00007789102156258012 0.9652728
## StandingTackle            0.00000000000577185107 0.9381571
## Release.Clause            0.00475045429828507577 0.9555943
## Crossing:HeadingAccuracy  0.00000000039925165033 1.0051971
## Finishing:HeadingAccuracy 0.00000062693420734799 0.9963512
```

Each predictor is measured as a rating on a scale of 100. Based on the above odds ratios, we can reach the following conclusions:

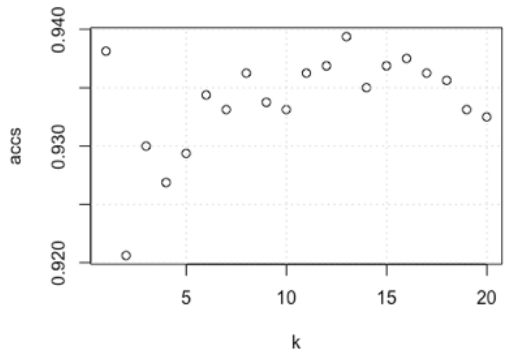
(1) As a player's Crossing rating increase by 1 point, keeping HeadingAccuracy unchanged, the odds(relative probability) of the player being a striker decrease by around 31%
(2) As a player's Finishing rating increase by 1 point, keeping HeadingAccuracy unchanged, the odds of the player being a striker increases around 47%
(3) As a player's ShortPassing rating increase by 1 point, the odds of the player being a striker decrease around 7%
(4) A one-point increment in a player's Volleys score will increase the odds of the player being a striker around 5%
(5) A one-point increment in a player's BallControl score will decrease the odds of the player being a striker around 9%
(6) A one-point increment in a player's Strength score will increase the odds of the player being a striker around 3%

(7) When the Marking score of a player increases by 1 point, the odds of the player being a striker will decrease around 4%
(8) When the StandingTackle score of a player increases by 1 point, the odds of the player being a striker will decrease around 7%
(9) When the Release Clause score of a player increases by 1 point, the odds of the player being a striker will decrease around 5%
(10) For each single point increase in HeadingAccuracy, keeping Crossing score and Finishing score unchanged, the odds of a player being a striker will increase around 0.15%

Based on the above result, we can conclude that a player's Finishing score has the strongest positive effect on determining if the player will be a striker, whereas a player's Crossing score has the strongest negative effect on determining if the player will be a striker. Therefore, to categorize whether a player will be a striker, we should focus more on these two perspectives.

- **K-NN**
    - **Step 1: Select K**

For this part, we started by selecting the best K from 1 to 20.



And from the graph above, we can see that accuracy reachs the highest when K equals to 13. Therefore, we chose 13 to be our best K for the K-NN model.

- **Step 2: Build Models and Model Selection**

(1) Original Model

Here is the result of our original K-NN model with features including “Crossing”, “Finishing”, “HeadingAccuracy”, “ShortPassing” and etc.

		Reference	
		0	1
Prediction	0	762	23
	1	74	741

Confusion Matrix and Statistics

```
               Accuracy : 0.9394
                 95% CI : (0.9265, 0.9506)
    No Information Rate : 0.5225
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.8789
 Mcnemar's Test P-Value : 0.000000384

            Sensitivity : 0.9115
            Specificity : 0.9699
         Pos Pred Value : 0.9707
         Neg Pred Value : 0.9092
             Prevalence : 0.5225
         Detection Rate : 0.4763
   Detection Prevalence : 0.4906
      Balanced Accuracy : 0.9407

       'Positive' Class : 0
```

From the above, the K-NN model could correctly predict 94% of the cases.

(2) Adding Polynomial Terms

To further refine the original model, we added two polynomial terms into the original model: “Square of Heading Accuracy” and “Square of Standing Tackle”.

Since adding polynomial term will change the scale, we further scale the polynomial terms to 100, which is consistent with the other features in the original data.

		Reference	
		0	1
Prediction	0	763	26
	1	73	738

Confusion Matrix and Statistics

```
               Accuracy : 0.9381
                 95% CI : (0.9252, 0.9494)
    No Information Rate : 0.5225
    P-Value [Acc > NIR] : < 0.0000000000000022

                  Kappa : 0.8763
 Mcnemar's Test P-Value : 0.000003779

            Sensitivity : 0.9127
            Specificity : 0.9660
         Pos Pred Value : 0.9670
         Neg Pred Value : 0.9100
             Prevalence : 0.5225
         Detection Rate : 0.4769
   Detection Prevalence : 0.4931
      Balanced Accuracy : 0.9393

       'Positive' Class : 0
```

As we can see from the above result, after adding polynomial terms, the model predicts less positive cases and more negative cases, which lead to higher sensitivity and lower specificity. But due to the accuracy decreases from the original model, we decided not to use this model. (Here, we assume we want a model with higher accuracy.)

(3) Adding Interaction Terms

We further add two interaction terms into the original K-NN model and scale them to 100 as well.

		Reference	
		0	1
Prediction	0	759	23 --
	1	77	741 --

Confusion Matrix and Statistics

```
              Accuracy : 0.9375
                95% CI : (0.9245, 0.9489)
   No Information Rate : 0.5225
   P-Value [Acc > NIR] : < 0.00000000000000022

                 Kappa : 0.8751
Mcnemar's Test P-Value : 0.0000001158

            Sensitivity : 0.9079
            Specificity : 0.9699
         Pos Pred Value : 0.9706
         Neg Pred Value : 0.9059
             Prevalence : 0.5225
         Detection Rate : 0.4744
   Detection Prevalence : 0.4888
      Balanced Accuracy : 0.9389

       'Positive' Class : 0
```

From the above, the new model predicts less negative cases and more positive cases. Therefore, the accuracy will reasonably decrease due to this change. Hence, we still decide not to adopt this model.

Model Selection

Comparing with the accuracy of different KNN models, we adopt the original K-NN model without any polynomial or interaction terms for the final result.

- **Step 3: Result Interpretation**

```
               Accuracy : 0.9394
                 95% CI : (0.9265, 0.9506)
    No Information Rate : 0.5225
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.8789
 Mcnemar's Test P-Value : 0.000000384

            Sensitivity : 0.9115
            Specificity : 0.9699
         Pos Pred Value : 0.9707
         Neg Pred Value : 0.9092
             Prevalence : 0.5225
         Detection Rate : 0.4763
   Detection Prevalence : 0.4906
      Balanced Accuracy : 0.9407

       'Positive' Class : 0
```

Final Model Metrics

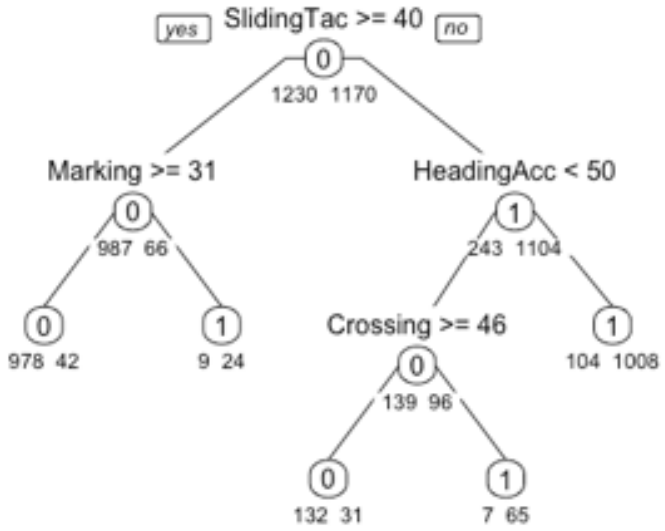
From the above, we find that:

The K-NN model correctly predicts 94% of the cases on the validation set. Since our class label is balanced, if with a random guess we could only correctly predict nearly 50% of the cases. Therefore, this model could greatly help us predict whether a specific player is a good fit for the "Striker" position. But there are things should be noticed: since the sensitivity of this model is lower than other models, the final result might include more irrelevant cases predicted even though the accuracy is high. That means when using this model to predict whether a player fits the "Striker" position, some of the players who actually don't fit are selected. Therefore, clubs might need to further evaluate the player's performance to decide the best position. And due to this attribute, we think this model is good to use when a club wants to select potential players from a large pool.

- **Classification Tree**

(1) With all variables

We first built a classification tree simply by throwing all the variables in the data frame and the results are shown below.



		Reference	
		0	1
Prediction	0	731	54
	1	105	710

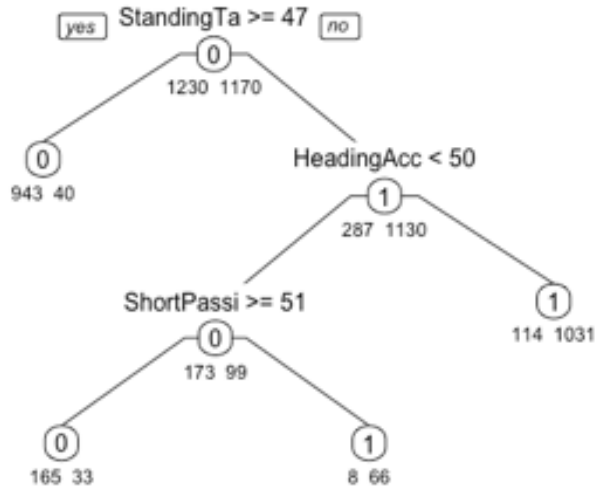
Confusion Matrix and Statistics

```
               Accuracy : 0.9006
                 95% CI : (0.8849, 0.9149)
    No Information Rate : 0.5225
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8014
 Mcnemar's Test P-Value : 7.332e-05

            Sensitivity : 0.8744
            Specificity : 0.9293
         Pos Pred Value : 0.9312
         Neg Pred Value : 0.8712
             Prevalence : 0.5225
         Detection Rate : 0.4569
   Detection Prevalence : 0.4906
      Balanced Accuracy : 0.9019

       'Positive' Class : 0
```

This model gives an out-of-sample accuracy of 0.9006.

(2) With significant variables only

Then by only using the significant variables shown in the logistic regression part, we built a second classification tree.



	Reference	
	0	1

Prediction	0	734	48
	1	102	716

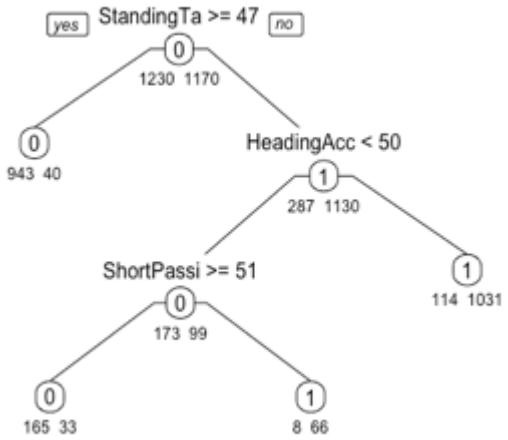
```
               Accuracy : 0.9062
                 95% CI : (0.8909, 0.9201)
    No Information Rate : 0.5225
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8127
 Mcnemar's Test P-Value : 1.509e-05

            Sensitivity : 0.8780
            Specificity : 0.9372
         Pos Pred Value : 0.9386
         Neg Pred Value : 0.8753
             Prevalence : 0.5225
         Detection Rate : 0.4587
   Detection Prevalence : 0.4888
      Balanced Accuracy : 0.9076

       'Positive' Class : 0
```

Compared with the first model, which includes all the variables, the second model gives slightly better performance of accuracy (0.9062)

(3) Adding polynomial terms

Next, we also want to know if polynomial terms could further improve the model, so we added several polynomial terms onto the second model, including the square of HeadingAccuracy, the square of StandingTackle, and the square of ReleaseClause.

Before we processing these new variables into tree, we scaled them into a 100 scale in order to make it consistent with other variables.



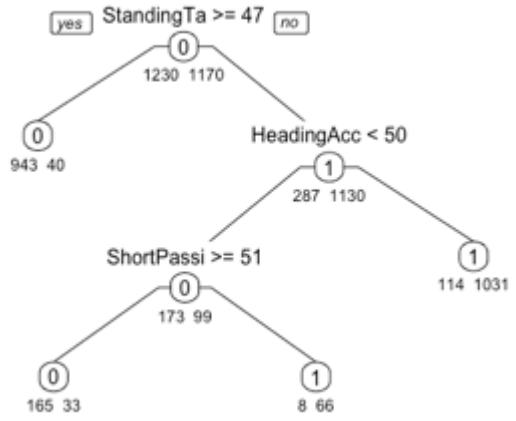
		Reference	
		0	1
Prediction	0	734	48
	1	102	716

```
               Accuracy : 0.9062
                 95% CI : (0.8909, 0.9201)
    No Information Rate : 0.5225
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8127
 Mcnemar's Test P-Value : 1.509e-05

            Sensitivity : 0.8780
            Specificity : 0.9372
         Pos Pred Value : 0.9386
         Neg Pred Value : 0.8753
             Prevalence : 0.5225
         Detection Rate : 0.4587
   Detection Prevalence : 0.4888
      Balanced Accuracy : 0.9076

       'Positive' Class : 0
```

The results shown above are exactly the same as the second model (with all significant variables). Therefore, adding polynomial terms would not improve the overall performance.

Model Selection and Interpretation

Through comparison of accuracy for the three models above, we would choose the second one (with significant variables only) as our final model in terms of the classification tree.




```
               Accuracy : 0.9062
                 95% CI : (0.8909, 0.9201)
    No Information Rate : 0.5225
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8127
 Mcnemar's Test P-Value : 1.509e-05

            Sensitivity : 0.8780
            Specificity : 0.9372
         Pos Pred Value : 0.9386
         Neg Pred Value : 0.8753
             Prevalence : 0.5225
         Detection Rate : 0.4587
   Detection Prevalence : 0.4888
      Balanced Accuracy : 0.9076

       'Positive' Class : 0
```

The result shows that classification tree (with significant variables only) might be a good way to predict whether a certain player is suitable to be a striker or not since it gives a quite high accuracy.

This classification tree model asks three questions to determine whether a certain player is a striker or not.

- Players with Standing Tackle less than 47 are classified as non-striker.
- Players with Standing Tackle greater than 47, Handing Accuracy less than 50, and Short Passing greater than 51 are also classified as non-striker.
- Players with Standing Tackle greater than 47 and Handing Accuracy greater than 50 are classified as strikers.
- Players with Standing Tackle greater than 47, Handing Accuracy less than 50, and Short Passing less than 51 are also classified as strikers.

- **Summary**

(1) Compare Logistic Regression with Classification Tree

Compare the best model from logistic regression to the best model form classification tree, we found that these models reinforced each other.

In our best logistic regression model, which is the logistic regression model with interaction terms, the variable “Standing Tackle” is significant at 0.001 level, and the coefficient of “Standing Tackle” is negative, which indicates that the higher the rating is on Standing Tackle, the less likely the player is classified as a striker. In our best classification tree model, if the player’s rating on “Standing Tackle” is higher than 47, then the player is classified as “non-striker”. The rule of classification tree is consistent with our interpretation of the logistic regression model’s coefficient, and it is also consistent with our common sense. Standing Tackle is seldom considered to be an important skill for strikers.

An interesting fact is that the “Heading Accuracy” variable is dropped from our logistic regression model because of low significance, but it plays an important role in classification tree. In the second level of the tree, if the “Heading Accuracy” is equal to or higher than 50, then the player will be classified as striker. This makes sense since strikers are expected to excel in scoring skills including heading. However, it's unfair to say that these models are contradicting each other, because in our initial logistic regression model, heading accuracy is significant with positive coefficient. It just lost the significance after introducing interactions.

Similar to the “Standing Tackle”, the “Short Passing” variable is a significant predictor in our best logistic regression and is also an essential rule in the classification tree. According to the rule of the tree, if the “Short Passing” is higher than or equal to 51, the player is considered “non-striker”. Its coefficient in logistic regression is negative, indicating that the higher the

value is, the less likely the player will be classified as a striker. These models reinforced each other's conclusion.

Looking at the confusion matrices, the accuracy of logistic regression is 0.9394, while the accuracy of classification tree is 0.9062. The logistic regression also beats classification tree on sensitivity and specificity. One possible explanation for classification tree's unsatisfactory performance is that it is overly simple and some important classification information is missing from the model.

(2) Compare KNN with Classification Tree

In our best KNN model, we chose k = 13 because it gave us the highest accuracy, compare to 2400 observations in our training data set, it's a fairly small number. It is relatively sophisticated than the models where k is a larger number, however, with an accuracy of 0.9394, it has good performance on the validation data.

The best classification tree model only contains three rules and is a pretty simple model for a data set with more than 30 variables. Its performance is not as good as KNN model. The accuracy is 0.9062.

Comparing the two models and their confusion matrices, we can discover that KNN model is more complex but has better performance while tree model is simple but has worse performance in terms of accuracy. It seems that the two models are contradicting each other.

Best Model Selection

Compare the three best models in different methods, logistic regression model and KNN model surpass classification tree because they have higher accuracy. The logistic regression model and KNN model have the same accuracy as 0.9394. However, the sensitivity of the logistic model is 0.9139, while the sensitivity of KNN model is 0.9115. Since our model is built to decide whether a player is suitable for the striker position, we care more

about whether a striker is classified correctly or not. Taking this into consideration, we think the logistic regression model is our best model.