Predict Hit vs No Hit Using Statcast Data

Ruslan Davtian

October 27, 2019

Data Description:

The Trackman pitch level data (Pitch_Data_v3.csv) represent pitch-by-pitch and at bat data for many of the pitchers and batters in MLB for the 2018 season across 30 different parks. The data provide information such as pitch speed, pitch location, pitch break, pitch release, pitch type, exit speed, and launch angle. The data set contains a total of 115,137 pitches and 57 columns. The goal of this analysis is to build a model on some subset of the data (training set) to predict the estimated probability of each pitch resulting in a hit or out and use that model to make predictions for some testing set.

GameId	BallparkId	GameEventId	GameTime	Season	AtBatId	TrackManUid	BatterId	BatterHitting	BatterPositionId
344124	8	226194255	2018-03-29 16:00:00.0000000	2018	78636835	29719057-338F-11E8-8704-0CC47A42EF89	100942	L	8
344124	8	226194222	2018-03-29 16:00:00.0000000	2018	78636827	982D1DDB-3390-11E8-8704-0CC47A42EF89	48371	R	2
344124	8	226194302	2018-03-29 16:00:00.0000000	2018	78636847	F0878B37-3395-11E8-8704-0CC47A42EF89	34306	L	7
344124	8	226194248	2018-03-29 16:00:00.0000000	2018	78636832	861CD51C-338E-11E8-8704-0CC47A42EF89	99078	R	6
344124	8	226194260	2018-03-29 16:00:00.0000000	2018	78636836	776B5C7B-338F-11E8-8704-0CC47A42EF89	76713	R	2
344124	8	226194286	2018-03-29 16:00:00.0000000	2018	78636841	168950B9-3394-11E8-8704-0CC47A42EF89	42011	R	9

Above, are the first six rows and 10 columns of the Trackman data set. My first step is to understand the variables and choose ones that can be predictive of a pitch resulting in a hit or out. For simplicity, I did not look into many of the game characteristic variables such as GameId, BallparkId, GameEventId, Gametime, BatterId, PitcherId, etc. I focused on the more intuitive variables which would provide me most of the variation explained in the response. Therefore, the variables of interest that I have identified as possible useful predictor variables are…

batterTimesFaced, cumulativeBattersFaced, PlateAppearance, HorzBreak, VertBreak, StartSpeed, RelHeight, RelSide, Extension, VertRelAngle, HorzRelAngle, SpinRate, SpinAxis, ExitSpeed, Angle, Bearing, and Direction.

Missing Values and Data Manipulation:

After investigating the data set, 1,676 rows are missing spin rate. The values seem to be missing at random since these missing rows are from all 30 different ballparks and 342 different pitchers. Therefore, I felt it appropriate to filter out those rows. After removing those rows, I am left with just 170 missing values for the BatterPositionId (1-9). I decided to keep those missing values as I will not be using that column anyway. Lastly, I also created separate indicator variables for each pitch type to be able to use a specific pitch type as a variable into the model.

Analysis Approach:

My approach is to build a few machine learning classification algorithms to compare performance and choose the model with the highest accuracy of cross validated predictions to predict the pitch type in the test set. I chose to compare five algorithms (logistic regression, decision tree, random forest, k-th nearest neighbor and neural network) using 5-fold cross validation on a training set for each one for consistency between the algorithms. Also, I compared each one by computing overall accuracy and kappa statistics for each classifier on the training and testing sets. I will select the model that best maximizes test accuracy in predictions.

Balance of Classes:

To understand the proportion of pitches resulting in hits or outs, I created a table below displaying the proportions. About 65% of the pitches resulted in outs and the other 35% of pitches resulted in hits.

Hit	Out
0.353	0.647

Data Visualizations:

Now that I have my official list of variables, I would like to plot their averages by result (hit/out) as well as result and pitch type. Below, is a data set showing the averages of the remaining variables for each result. Most of the variables do not show any significant mean differences between hits and outs except for ExitSpeed, Angle, Bearing, and Direction.

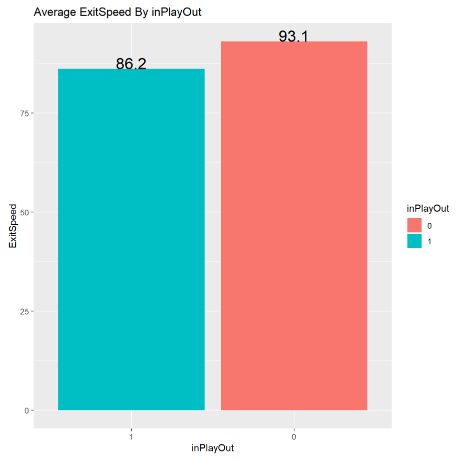
inPlayOut	batterTimesFaced	cumulativeBattersFaced	PlateAppearance	HorzBreak	VertBreak	StartSpeed	RelHeight	RelSide	Extension	VertRelAngle	HorzRelAngle	Sp
0	1.6	9.0	2.8	-1.4	4.3	89.0	5.9	-0.8	6.1	-1.5	1	
1	1.5	9.1	2.8	-1.4	4.2	88.9	5.9	-0.8	6.1	-1.4	1	

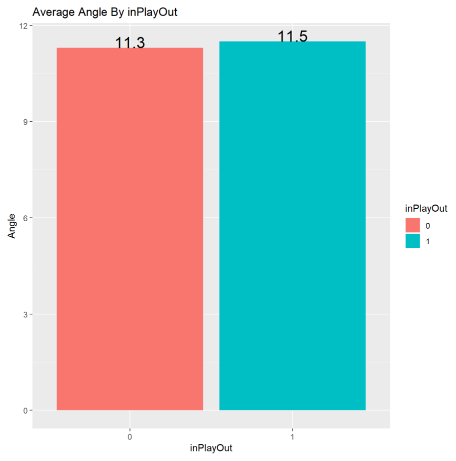
It is also useful to look into variables averaged out by pitch type and result. The averages can be found in the data set below. Similarly, the only significant differences are from ExitSpeed, Angle, Bearing, and Direction. However, pitch types may still be interesting to throw into a model as one of the pitches may lead to more outs or hits than others.

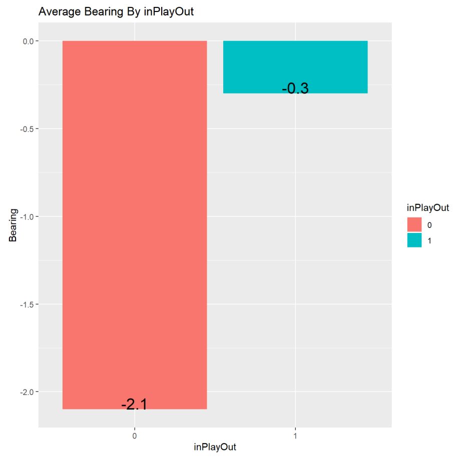
PitchType	inPlayOut	batterTimesFaced	cumulativeBattersFaced	PlateAppearance	HorzBreak	VertBreak	StartSpeed	RelHeight	RelSide	Extension	VertRelAngle	HorzR
CB	0	1.7	10.3	2.9	2.1	-6.1	78.9	6.0	-0.8	5.8	0.7	
CB	1	1.7	10.2	2.9	2.1	-6.1	78.9	6.0	-0.7	5.8	0.7	
CH	0	1.7	10.4	2.8	-1.9	3.6	84.2	5.9	-0.4	6.2	-1.2	
CH	1	1.7	10.6	2.8	-1.9	3.6	84.2	5.9	-0.4	6.2	-1.3	
CT	0	1.6	9.2	2.9	0.5	4.5	88.7	5.9	-0.6	6.0	-1.5	
CT	1	1.5	9.1	2.9	0.5	4.5	88.6	5.9	-0.5	6.1	-1.5	
FB	0	1.5	8.2	2.8	-1.9	8.2	93.1	6.0	-0.8	6.3	-2.2	
FB	1	1.5	8.3	2.8	-1.9	8.3	93.2	6.0	-0.8	6.3	-2.2	
KN	0	1.5	8.5	3.1	-1.0	-1.4	75.6	5.9	-0.8	5.8	1.0	
KN	1	1.5	8.7	2.7	-0.9	-1.1	75.5	5.9	-0.8	5.9	0.9	
SI	0	1.6	9.3	2.8	-4.0	4.8	92.1	5.9	-0.8	6.1	-1.7	
SI	1	1.6	9.2	2.8	-4.0	4.7	92.2	5.9	-0.8	6.2	-1.7	
SL	0	1.5	8.9	2.9	1.4	0.5	84.5	5.9	-1.0	5.9	-0.7	

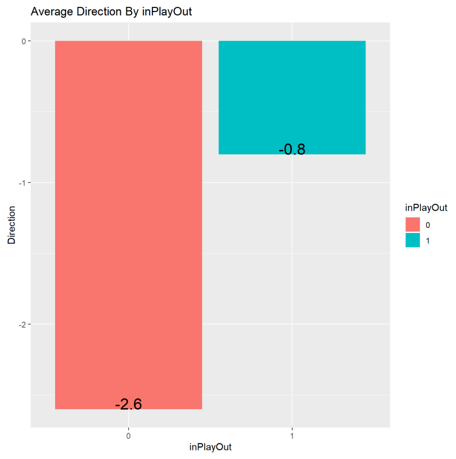
PitchType	inPlayOut	batterTimesFaced	cumulativeBattersFaced	PlateAppearance	HorzBreak	VertBreak	StartSpeed	RelHeight	RelSide	Extension	VertRelAngle	HorzR
SL	1	1.5	9.0	2.9	1.5	0.6	84.5	5.9	-1.0	5.9	-0.7	
SP	0	1.5	8.4	2.9	-4.5	2.1	85.1	6.0	-1.3	5.8	-1.2	
SP	1	1.5	8.3	2.8	-4.6	2.1	85.1	6.0	-1.3	5.8	-1.3	

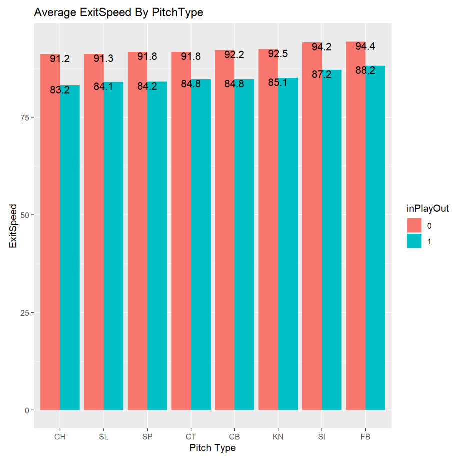
The following plots below compare ExitSpeed, Angle, Bearing, and Direction across result of pitch and across both result of pitch and pitch type.

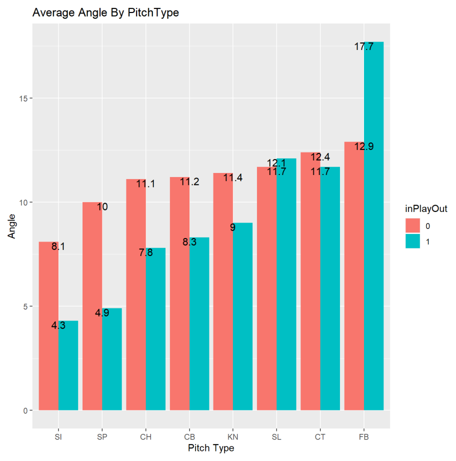


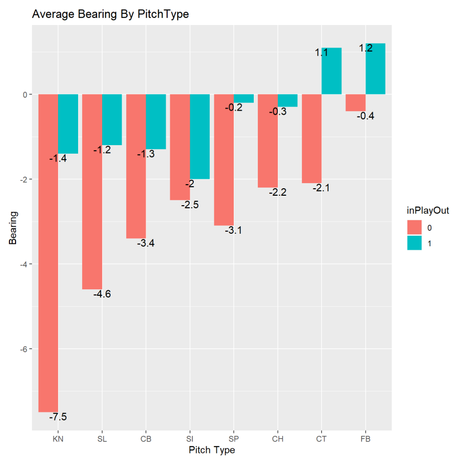


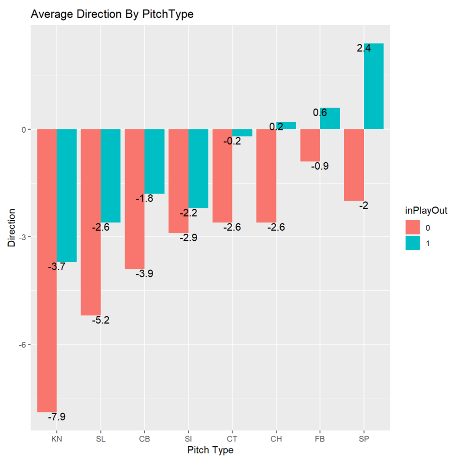












Multicollinearity Check:

Before I perform any modeling, I want to check if any of the variables selected as candidates for the models explain similar variation in the pitch types. Therefore, I checked Variance Inflation Factors (VIFs) among the variables. Any two variables over 9 reveal severe multicollinearity among them and one of them should be removed. The VIFs are located below.

ExitSpeed	Angle	Direction	Bearing
1.03	1.03	7.57	7.57

There does not seem to be any issue with severe multicollinearity. However, Bearing and Direction seem to be somewhat colinear. The correlation between these two variables is 0.926 which means I would need to select only one of them, but not both for modeling.

Pre-Processing and Machine Learning:

First, I split the overall Trackman in the data set into a 70/30 train, test split. For consistent and improved results, I standardized all predictor variables in the training and testing sets and used 5-fold cross validation across each algorithm. I used cross validated accuracy and kappa performance metrics to rank the algorithms.

Logistic Regression:

I decided to build a logistic regression model first to use as a base model to compare against. The output below is from the final logistic model chosen. My first model also incorporated pitch type and pitcher throwing side and only the sinker pitch type was statistically significant so I chose to use sinker as an indicator variable in the final model. Lastly, I chose to use angle instead of bearing as they are both colinear with each other. The final model incorporates exit speed, angle, direction, sinker, and batter hitting side.

Below, are the performance metrics for the test set. I looked at accuracy by result (hit/out) as well as overall accuracy and kappa metrics that will be used to compare against other algorithms. As the base model, the logistic regression model's overall accuracy on the test set is about 68%.

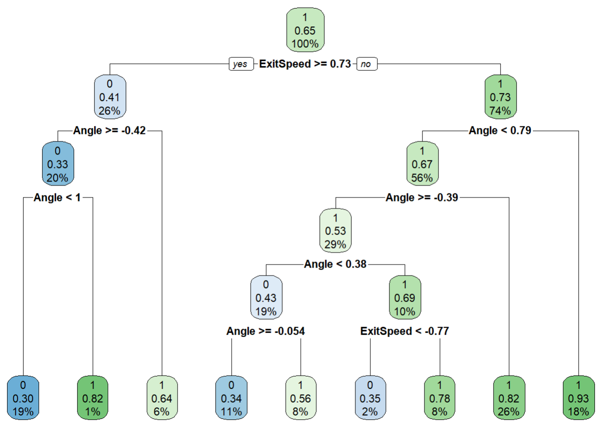
	Estimate	Std. Error	z value	p value
(Intercept)	0.6319781	0.0124615	50.714381	0.0000000
ExitSpeed	-0.5777474	0.0088284	-65.441575	0.0000000
Angle	0.0844215	0.0081685	10.335036	0.0000000
Direction	0.0858575	0.0084356	10.177981	0.0000000
SI	0.0267063	0.0076626	3.485285	0.0004916
BatterHittingR	0.0458151	0.0166803	2.746664	0.0060205

	0	1
Accuracy	0.662	0.68

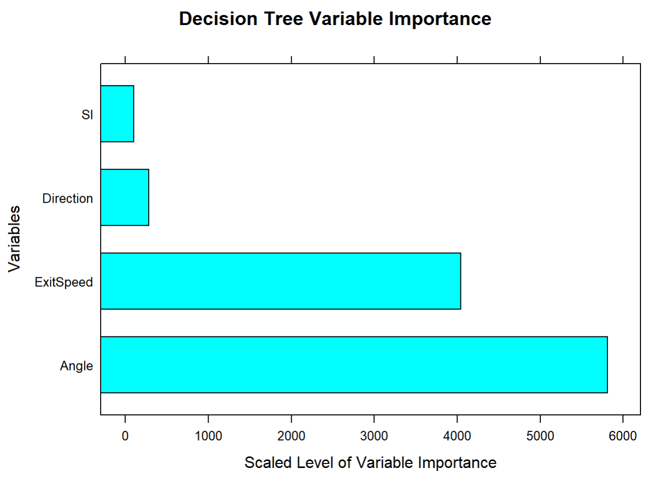
Ov	erall Test Accuracy
Accuracy	0.678
Kappa	0.161

Decision Tree:

I decided to build a decision tree first to look at the first few splits in order to determine the most important variables. I fixed the complexity parameter to not over fit or produce too long of a tree and chose 10 as minimum number of observations that must exist in a node in order for a split to be attempted. A plot of the decision tree can be found below.



At each split, the tree shows which variables the split was made but the values shown do not make sense because they are standardized. The final tree has depth 4 with 9 terminal nodes. The variables used in this tree as nodes are only exit speed and angle. It is important to note that a deeper tree many use all of the variables but there is a risk of over-fitting the training data set and performing worse on the testing set.



Above, is a bar chart showing the scaled levels of variable importance. The top 4 most important variables in order of significance for the decision tree is angle, exit speed, direction, and sinker.

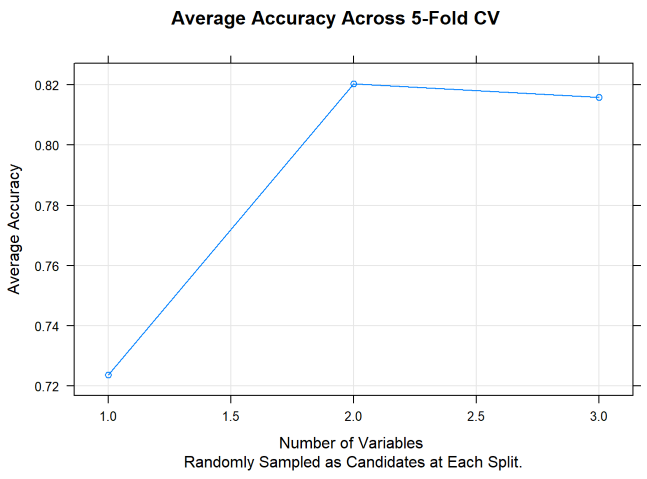
Below, is the accuracy by result as well as overall test accuracy and kappa values. The overall accuracy for the decision tree is significantly higher than that of the logistic regression at about 76%.

	0	1
Accuracy	0.679	0.796

	Ov erall Test Accuracy
Accuracy	0.759
Kappa	0.461

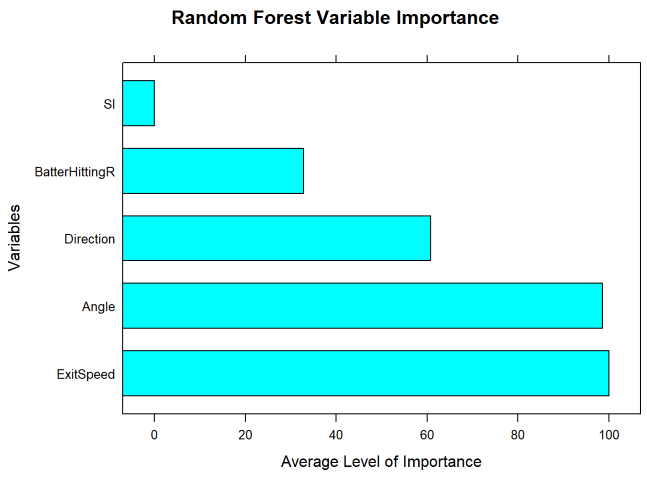
Random Forest:

Decision Trees might perform well on trained data but there could be possible issues with generalizing to new, unseen data. Therefore, I will try random forest as the next modeling technique. They are more difficult to interpret than decision trees but usually have higher predictive power. I used the same variables as the previous model and I chose the number of trees parameter to be 250. Also, I determined the number of variables (1 to 4) to be randomly sampled as candidates at each split through cross validation. A plot of this can be found below.



At each of the 250 trees, a random sample of variables to be elected as candidates for each split is important to not allow the same splits to happen over and over for each tree. It is determined that the best number of variables to have for random selection on each split is 2 from cross validated accuracy.

Below, is the average, scaled variable importance chart for the random forest algorithm. The top 5 most important variables in order of significance is angle, exit speed, direction, batter hitting side, and sinker.



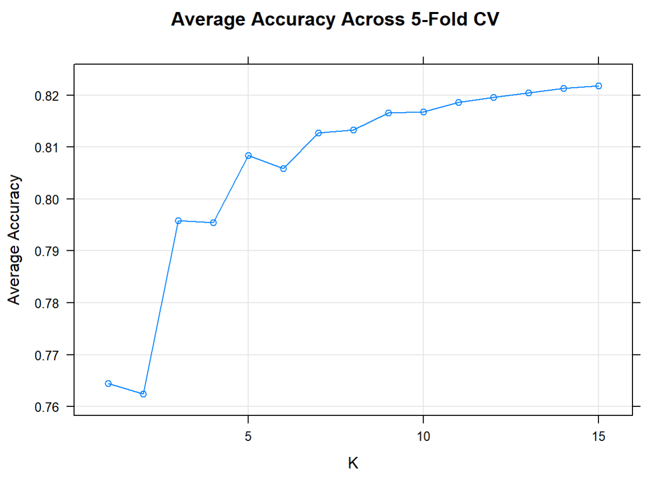
The accuracy by result and overall accuracy and kappa values for the random forest algorithm is below. The random forests algorithm performed significantly better than the decision tree with about 82% accuracy.

	0	1
Accuracy	0.771	0.845

Accuracy	Kappa
0.82	0.598

K-th Nearest Neighbors:

KNN is one of the simplest algorithms as it calculates the Euclidean distances between each of the predictors and a class for the testing observation is determined through the majority class of its k closest neighbors. K is chosen through cross validation that maximizes accuracy and kappa values. A plot of cross validated accuracy versus K is shown below.



The plot converges quickly after K = 10 but in this case, the K which produced the best overall accuracy and kappa values is 15.

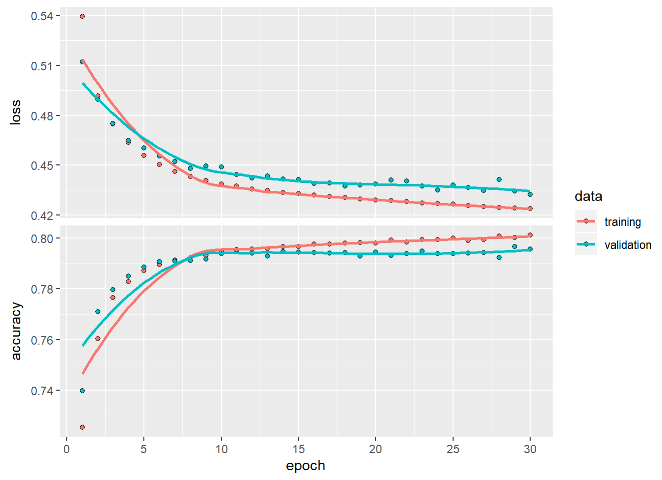
Below, is the accuracy by result and overall accuracy and kappa values for the KNN algorithm. The algorithm performs very similarly to the random forest (82% accuracy) and much better than the decision tree or logistic regression.

	0	1
Accuracy	0.74	0.899

Accuracy	Kappa
0.822	0.603

Neural Network:

For the last algorithm, I created a multi-hidden layer neural network to classify the result. Neural networks are very difficult to explain or interpret the results but when hyperparameterized correctly, can produce strong predictive results. I chose to have only two hidden layers where the first layer has 64 nodes and the second layer has 32 nodes. Both layers use relu as the activation function and the output layer has 2 units for the 2 result types. Also, the output layer uses the softmax activation function to transform the outputs into probabilities between 0 and 1 for multiclass classification. This process of feeding observations into the input layers and calculating the output layer is called forward propagation. In order to improve the accuracy of the predictions, the back propagation process needs to be optimized to minimize some loss function (categorical cross-entropy). In this process, I chose to use only 30 iterations or epochs with batch sizes of 128 for faster training. Also, I used 0.2 as the validation split during the training. The plot of the accuracy and loss for both training and validation sets throughout the 30 iterations can be found below.



The neural network converged quickly after the first 10-15 iterations and the accuracy from both training and validation sets seem to be around 80% which is similar to random forest and KNN algorithms. The overall accuracy for the neural network is significantly better than the decision tree and logistic regression model and only slightly worse among random forest and KNN.

accuracy
0.801

	0	1
Accuracy	0.647	0.881

Ranking Algorithms

In order of overall accuracy, the algorithms from highest to smallest are as follows, (knn, random forest, neural network, decision tree, logistic regression). However, the overall accuracies and kappa values from KNN and random forest models are not significantly different from each other. Therefore, I chose to use the random forest model to make predictions on the testing set instead of the KNN model due to easier methods of extracting class probabilities. To make sure the testing predictions make sense, it is important to compare the distribution of results for the predictions against the distribution of actual results from the testing set.

Below is the distribution of results for the test data.

Hit	Out
0.354	0.646

Below is the distribution of results for the predictions made using the random forest model.

Hit	Out
0.324	0.676

Below is the two way table Confusion Matrix comparing the actual testing results (columns) versus the predicted results (rows).

	0	1
0	8621	2558
1	3621	19742

The proportion of results for both sets are very similar so the predictions from the random forest model make sense. Below, the predicted test result probabilities and the actual results are shown for the first 30 rows.

Actual	Predicted	Hit Probability	Out Probability
1	1	0.020	0.980
0	1	0.380	0.620
1	1	0.208	0.792
1	1	0.180	0.820
1	1	0.160	0.840
1	1	0.172	0.828
0	1	0.376	0.624
1	0	0.648	0.352
0	0	0.636	0.364
0	0	0.972	0.028
0	0	0.816	0.184
0	0	0.824	0.176
0	0	0.820	0.180
1	1	0.008	0.992
0	0	0.820	0.180
0	0	0.916	0.084
1	1	0.000	1.000
1	0	0.788	0.212
1	1	0.008	0.992
1	1	0.028	0.972
0	0	0.988	0.012
0	1	0.432	0.568
1	1	0.432	0.568
1	1	0.000	1.000
1	1	0.092	0.908
1	1	0.008	0.992
1	0	0.808	0.192
0	0	0.944	0.056
1	0	0.552	0.448
1	0	0.616	0.384