# Classification Techniques on NFL Big Data Bowl 2025

Chris Jaeger
Aine Grubb

## Abstract:

The following data analysis investigates the effects of various pre and post-snap predictor variables from the NFL's 2025 Big Data Bowl. The overall analysis focuses on three outcome variables: the occurrence of Play-Action plays, and the success rate of both passing and rushing plays. This specific investigation focuses more on the success rates of plays. Classification predictive models were created for each target variable, revealing that random forest was the most successful for both passing and rushing success rate. Support vector machines were discovered to be the best at predicting the occurrence of a play-action pass. While the models exhibited moderate predictive success, there is still significant room for improvement. This could come with the addition of player and historical data. However, it still remains as sports are difficult to predict, as strategy and human error are variables that impact predictive modeling.


## Introduction:

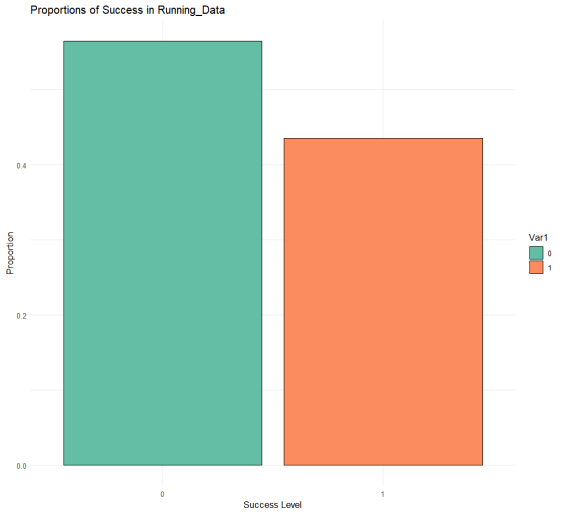
The Big Data Bowl is an annual public contest, hosted by the NFL in tandem with Kaggle, with the aim to use data to understand more about football. The topic for 2025 is pre-snap behavior- what pre-snap behavior and outlooks can indicate about a play. While the NFL provided 5 different types of datasets, the play data is the focus of this analysis, so we can focus on predicting individual play outcomes. This analysis is going to have a two approach: predicting if an occurrence of a play-action pass and predicting the success rates of passing and rushing plays. This specific analysis focuses on the latter, while the analysis created by Grubb focuses on the former. The definition of a successful play was taken from sports reference:

1. 1st down: 40% of required yards gained (or more)
2. 2nd down: 60% of required yards gained (or more)
3. 3rd/4th down: All of required yards gained (or more)

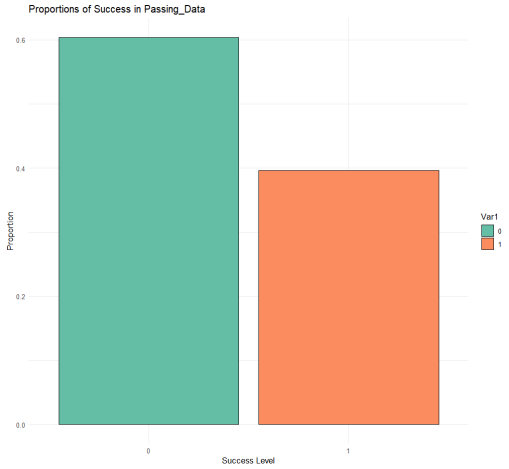
1 or x1 represents a successful play, while 0 or x0 represents an unsuccessful play. Due to the large amount of NA variables, as will be discussed in the next section, and the amount and one-hot encoded variables, the data was altered for support vector machines (SVMs), neural networks, SVM, and discriminant analysis to include only numeric variables. This created some difference in significant variables and the misclassification error may also be different from that of the random forest, classification tree and logistic regression. This is the case for both passing and rushing data. ChatGPT assisted in the coding process in order to handle the large magnitude of data.

## Exploratory Data Analysis:

A large majority of the data was used providing Amazon Web Services (AWS) which is automatically processed data. Thus, this causes some issues as there are times when the cameras were not able to identify the occurrence of some variables. Thus, this led to a large amount of missing (NA) observations, which caused errors in some of the models. In order to reduce these errors, the data was split into rushing and passing data. This reduced the amount of NA observations in each dataset as the variables were split by relevance to each play time. Below are the proportions of success in running and passing data, respectively.




Above- Proportion of Success in Running Plays, Orange Bar Success (.44)




Above- Proportion of Success in Passing Plays, Orange Bar Success (.4)

Although the success rates are close to a 50/50 split, specifically with the success proportion being .44 for Rushing Data, and .4 for passing data, the prediction on some models would predict only one outcome class. Even if this creates a low misclassification error, it is not predicting anything. To combat this issue, we applied the Synthetic Minority Oversampling Technique (SMOTE). The SMOTE technique provides more observations to the minority class until it closes the gap and creates a much more balanced dataset. It generates data that is closely related to the observations that are previously collected, as to not disturb the patterns that were already there. In both cases below, the gap was closed to a proportion of very close to .5.

| SMOTE Data | No/NoSuccess | Yes/Success |
|---|---|---|
| Passing Data | 5013 | 4987 |
| Rushing Data | 4984 | 5016 |

Above - New Class amounts for the datasets after SMOTE (through the ROSE package) was applied

```
[1] "Radial SVM Test Set Confusion Matrix:"
> print(svm_test_conf_matrix2)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0    0    0
         1 1495 1504

               Accuracy : 0.5015
                 95% CI : (0.4834, 0.5196)
    No Information Rate : 0.5015
    P-Value [Acc > NIR] : 0.5073
```

Right- Confusion matrix for SVM with only one class predicted

However, even after the balancing of classes, there were some issues. The image on the right is an example of how one class (only a success) was predicted. This one occurs after the balancing of classes, but this also occurred in almost every model before the balancing of classes. Additionally, as a lot of the rest of the variables are related to football, (location on field, game clock, down, etc.) it is not helpful to take averages or summary statistics of this data. In addition, many of the variables such as pff_passCoverage had many different options. Even when one-hot encoding these variables, discriminant analysis, neural network, and support vector models would not accurately work and would encounter errors. This is why the numeric only variables were used for this set of models.

## Statistical Modeling/Data Analysis

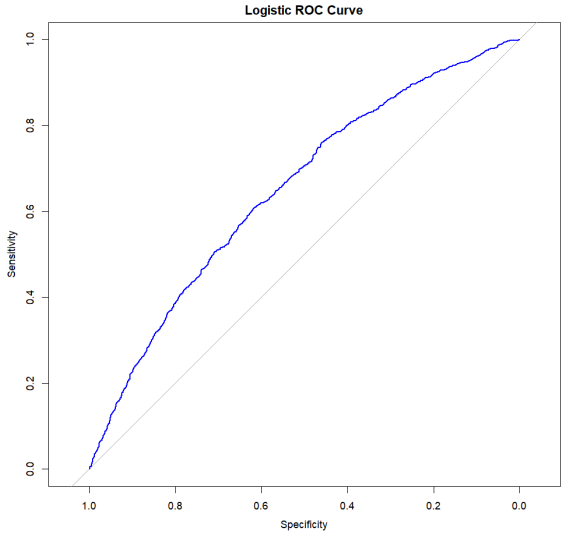
### Passing Data:

To begin the classification analysis, the first modeling technique used is Logistic Regression. In R, we used that backwards elimination technique, albeit manually, in order to reduce the model and focus on the most impactful predictors. The chart below shows the estimates and the significance for the variables that remained after model reduction. Many of these predictors are significant or close to the level of significance. However, some of the ones that aren't significant are shiftSinceCount, motionSinceCount, and inMotionCount, despite being important factors in pre-snap reads. Some of the most significant metrics are playClockAtSnap, PlayActionTrue, pff_passCoverageQuarters pff_passCoverageCover-3. All of these significantly contribute to a higher success rate in passing plays. Though not as significant as the previously listed predictors, dropbackDistance, timeToThrow and timeInTackleBox decrease the success rate as these increase. Overall, it is interesting to note that several of the pass coverage options have some sort of significance, which we will compare to the rushing data pass coverage. After running this logistic regression in order to obtain a confusion matrix, we are able to see that the accuracy of it is .5275, which is not a very high result, which also gets us to a test set misclassification rate of .4725. White it is better than solely guessing it is not that much higher. The model was also much more effective at predicting not successful plays (0) than successful plays (1).

Below- Estimates and Significance of Logistic Regression for Passing Data

```
Coefficients:
                                      Estimate Std. Error z value Pr(>|z|)
(Intercept)                         -1.398e+00  3.539e-01  -3.951 7.79e-05 ***
inMotionCount                        5.980e-02  5.632e-02   1.062 0.288276
shiftSinceCount                     -2.914e-02  5.018e-02  -0.581 0.561525
motionSinceCount                    -3.718e-02  3.816e-02  -0.974 0.329948
absoluteYardlineNumber               3.656e-03  1.534e-03   2.383 0.017192 *
playClockAtSnap                      3.117e-02  3.790e-03   8.223  < 2e-16 ***
targetX                              1.403e-03  1.400e-03   1.002 0.316219
targetY                              2.015e-03  1.602e-03   1.258 0.208322
playActionTRUE                       9.134e-01  7.478e-02  12.215  < 2e-16 ***
dropbackTypeDESIGNED_ROLLOUT_RIGHT  -1.017e-01  2.334e-01  -0.436 0.662934
dropbackTypeSCRAMBLE                -6.310e-01  2.588e-01  -2.438 0.014770 *
dropbackTypeSCRAMBLE_ROLLOUT_LEFT    1.035e+00  1.215e+00   0.852 0.394326
dropbackTypeSCRAMBLE_ROLLOUT_RIGHT  -1.249e+00  4.218e-01  -2.962 0.003061 **
dropbackTypeTRADITIONAL             -1.016e-01  1.921e-01  -0.529 0.597051
dropbackDistance                    -4.338e-02  1.484e-02  -2.924 0.003458 **
timeToThrow                         -6.982e-02  3.803e-02  -1.836 0.066402 .
timeInTackleBox                     -9.225e-02  4.260e-02  -2.165 0.030356 *
pff_runPassOption                   -1.459e-01  1.098e-01  -1.328 0.184045
pff_passCoverageBracket             -4.196e-01  5.495e-01  -0.764 0.445128
pff_passCoverageCover-0              3.954e-01  3.184e-01   1.242 0.214350
pff_passCoverageCover-1              3.899e-01  2.809e-01   1.388 0.165154
pff_passCoverageCover-1 Double      -7.998e-01  6.692e-01  -1.195 0.232048
pff_passCoverageCover-2              9.827e-01  2.826e-01   3.477 0.000507 ***
pff_passCoverageCover-3              1.212e+00  2.786e-01   4.351 1.36e-05 ***
pff_passCoverageCover-3 Cloud Left   5.672e-01  9.448e-01   0.600 0.548236
pff_passCoverageCover-3 Cloud Right  1.322e+00  6.164e-01   2.145 0.031924 *
pff_passCoverageCover-3 Double Cloud 1.549e+00  7.401e-01   2.093 0.036329 *
pff_passCoverageCover-3 Seam         1.082e+00  2.970e-01   3.642 0.000270 ***
pff_passCoverageCover-6 Right        9.029e-01  2.996e-01   3.013 0.002585 **
pff_passCoverageCover 6-Left         1.049e+00  3.005e-01   3.492 0.000479 ***
pff_passCoverageGoal Line            1.425e+00  6.916e-01   2.061 0.039283 *
pff_passCoverageMiscellaneous       -1.192e+01  1.840e+02  -0.065 0.948332
pff_passCoveragePrevent             -1.394e-02  7.234e-01  -0.019 0.984625
pff_passCoverageQuarters             1.110e+00  2.826e-01   3.928 8.57e-05 ***
pff_passCoverageRed Zone             5.265e-01  3.217e-01   1.637 0.101709
absoluteYardline2                   -1.281e-05  1.242e-05  -1.031 0.302335
targetX2                            -1.387e-05  1.125e-05  -1.232 0.217822
```

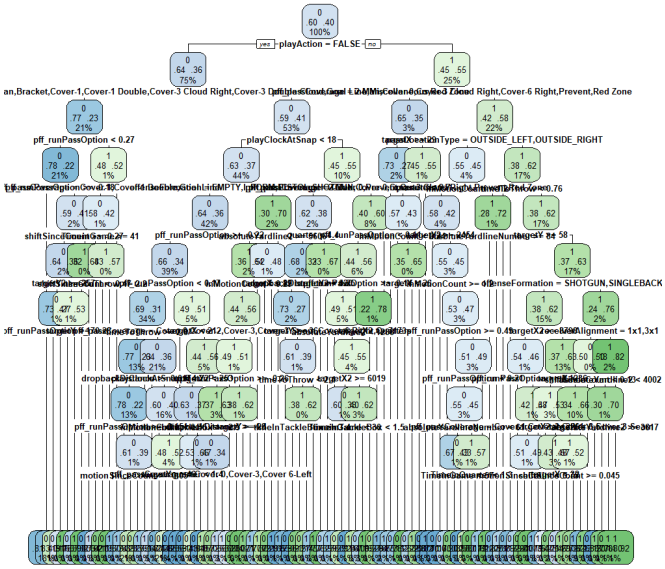
Below- Logistic ROC Curve for Passing Data

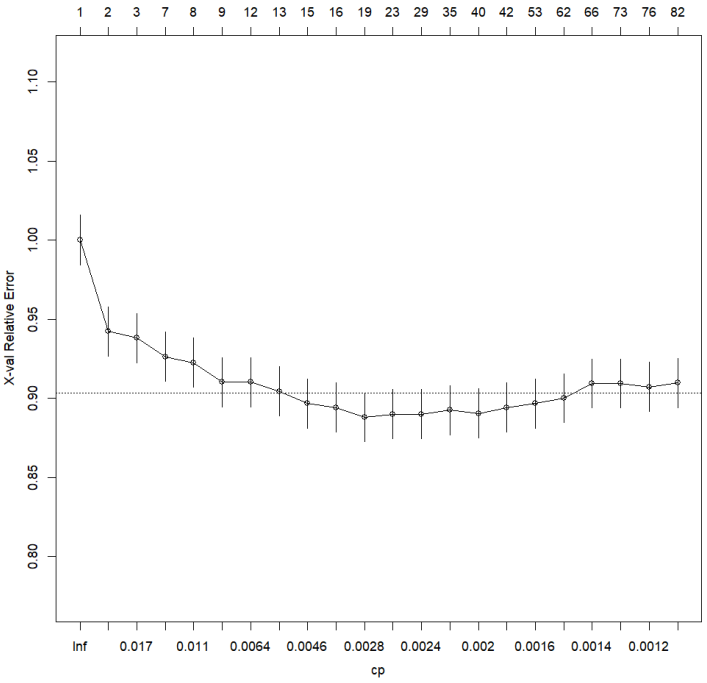



This all leads to a ROC curve that is higher and to the left of .5, and an AUC value of .65. This further indicates that the model is better than random guessing, but is closer to randomness than perfect prediction. Thus, as this model has low predictive power, it is important to continue to try other models.

The next modeling technique that we will use to predict passing play success is a classification tree. The first iteration of the model is down below and to the left, and obtained a test set misclassification rate of .416. As a rule for all of the passing data trees to the left at each split for all of these models means "yes" to each predictor, and to the right means "no." As seen below and on the left, the model is very complicated and almost impossible to comprehend. Thus, it was important in order to prune the tree

and test to see how it performed when it was a more readable tree. After testing different splits (pictured below and on the right ) and looking at which one had the lowest relative error, we came out with a suggested splits of around 18 with a cp of .0027.



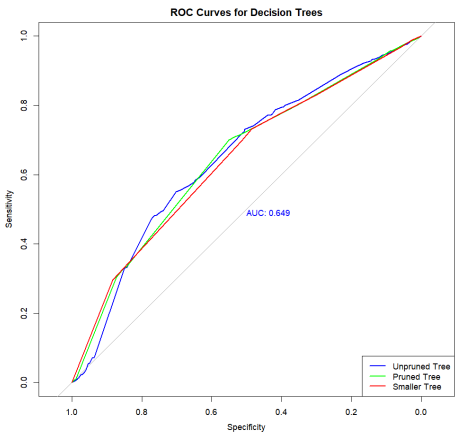




Above- unpruned tree for passing data

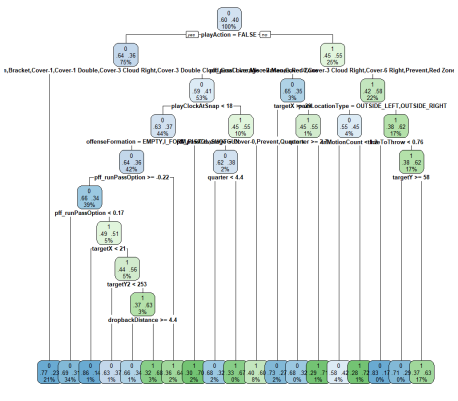
Above- Graphical representation of cp/number of splits on relative error

After creating this tree, depicted on the bottom left, we were able to see the classification error of this one which was a test set misclassification rate of .431, which is slightly worse than the misclassification rate of the initial model. Thus, the unpruned tree does better than the pruned tree for the testing set. Thus, this is the model that we would use in practice when classifying, despite the fact that it is much more complicated. Similarly, we can see that despite being almost identical, for the ROC curve, the unpruned tree (the blue line) has the highest AUC at .649. The other tree included is referenced as the smaller tree, and is depicted on the ROC curve as well. It performs slightly worse than the other models, as its misclassification error is .4609. However, as it does work similarly, it simplifies the model to understand what some of the most important factors are, and to easily see visibly how a basic form of this classification tree may occur.

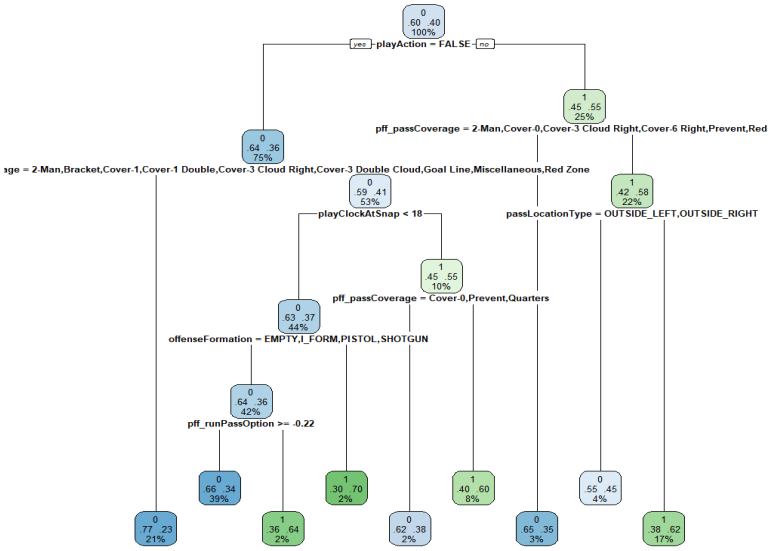
Below- ROC curves for each of the 3 decision trees for Passing Data




Below- Pruned tree for Passing Data

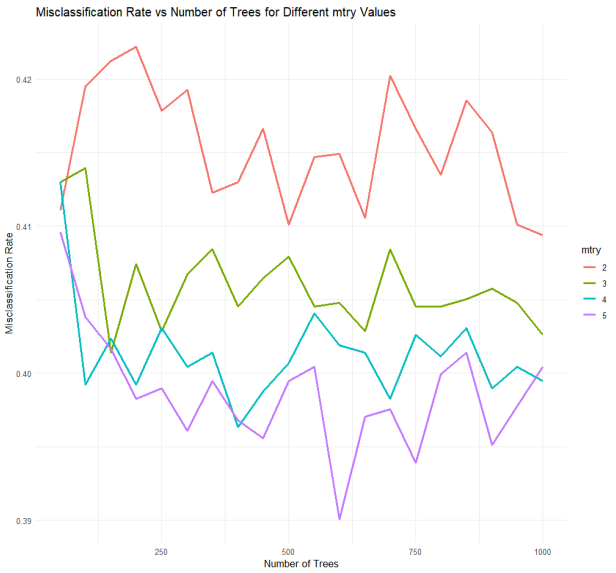


Below- The small tree model

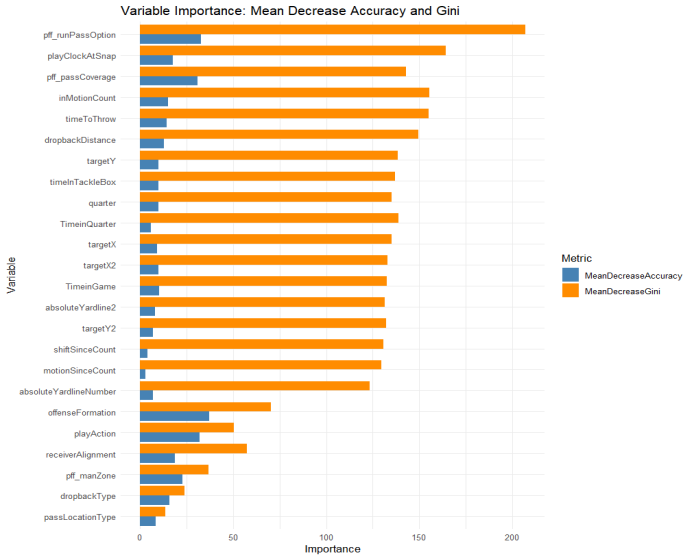



Thus as we can see for the model to the left, being the smaller model, the most significant/important variables are playAction, which is more related to successful play, passCoverage, and playClockAtSnap. While this model works better overall than the logistic regression with a lower misclassification rate, (.4725 for the logistic regression and .416 for the unpruned tree), there is still much more room for improvement in predictive power.

The next modeling technique that was investigated is the random forest model. To determine optimal parameters, the misclassification error was calculated for varying number of tree sizes and different variables considered at each split.



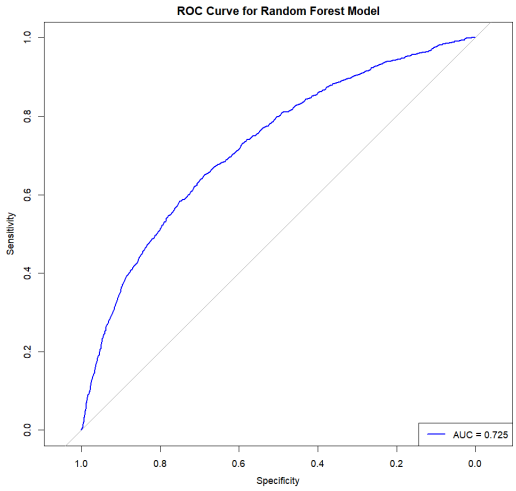

Above- Number of Trees/number of variables at each split vs misclassification rate




Above- Variable importance based off of Mean Decrease Gini and Mean Decrease Accuracy

The graph above depicts the investigation, in which the ideal parameters discovered were 600 trees and a mtry value of 5. This also leads to a misclassification error of .386, which is the lowest misclassification rate that we have seen for any of our models thus far. Above, we also have listed the variable importance by two different metrics, Mean Decrease Accuracy and Mean Decrease Gini. A higher value in both of

these variables indicates a higher importance. Thus some of the most important variables are pff_runPassOption, playClockAtSnap, pff_PassCoverage and inMotionCount and TimetoThrow. On the contrary, some variables that aren't important are passLocationType, dropbackType, pff_manZone and PlayAction. PlayAction has been one of the most significant factors in the mother models, thus it is noteworthy that it was not significant for this one. All of the other less -significant factors were only minor predictors for the other models. The graph to the right is the ROC curve for the random forest model, it is visible that the AUC is .725 here, and is higher than the other models that we have looked at so far. Therefore, there are multiple metrics that support random forest as the best model that we have looked into.



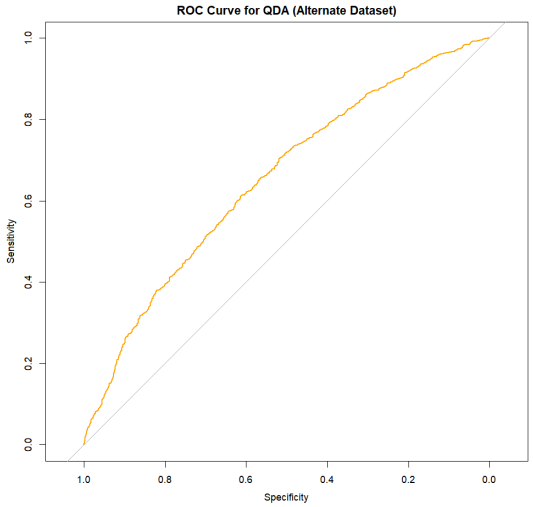

Right- ROC curve for the Passing Data random forest model

The next model investigated was discriminant analysis was the next model investigated. As stated above, for this model, neural networks and support vector models used a different, numeric variables only dataset. The first thing to apply in discriminant analysis is whether Linear Discriminant Analysis or Quadratic Discriminant Analysis is the proper one to use. A way to do this is Box's M-Test, which tests for homogeneity of covariance matrices. In this case, we received a p-value of <2.2*10^-16, which indicates that the covariance matrices are not equal, and thus we need to use Quadratic Discriminant Analysis (QDA) instead of Linear.

```
[1] "Box's M Test Results:"
> print(box_test_result2)

        Box's M-test for Homogeneity of Covariance Matrices

data:  train_data4[, -which(names(train_data4) == "Success")]
Chi-Sq (approx.) = 999.09, df = 136, p-value < 2.2e-16
```

Above- Box's M-Test for Passing Data

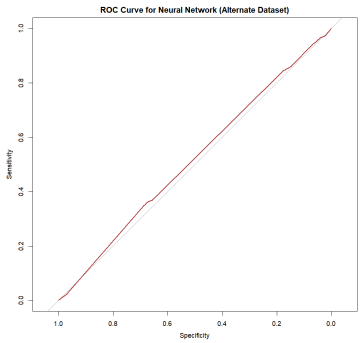
QDA obtained a misclassification error rate of the test set of .3889, and an AUC of .6525. While its AUC is closer to that of logistic regression and classification trees, its misclassification rate is on par with that of a random forest. Thus, when comparing to the other models, QDA is near the top, as it has a top misclassification rate and an AUC that is competitive with other models.

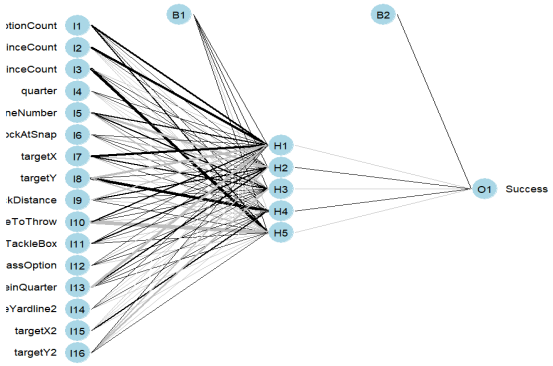



Right- ROC curve for Quadratic Discriminant Analysis for Passing Data

The next model that we are going to discuss is the Neural Network. After using cross validation to determine that the best parameters, we discovered the optimal parameters are decay of .1 and 5 hidden nodes, we achieved

the model shown below. The model had only one hidden layer used. The thickness and darkness of the lines in the mode displayed indicate the strength of the relationships- the thicker and darker a line the . Therefore, hidden node 2 and hidden node 4 are the highest related nodes to success, with their most impactful variable for node 4 being targetY. Some of the other variables that have strong relationship to the hidden nodes include timetoThrow, targetX, motionSinceCount, and shiftSinceCount. The variables that are not very impactful for any of the hidden nodes are pff_runPassOption, targetY2, absoluteYardline2 and playClockatSnap. However, this model was not very successful in predicting the test set, as its misclassification error on the test set was .4768, and the AUC was .5174, just above the mark for random guessing. Thus, we take these results and important variables with caution.

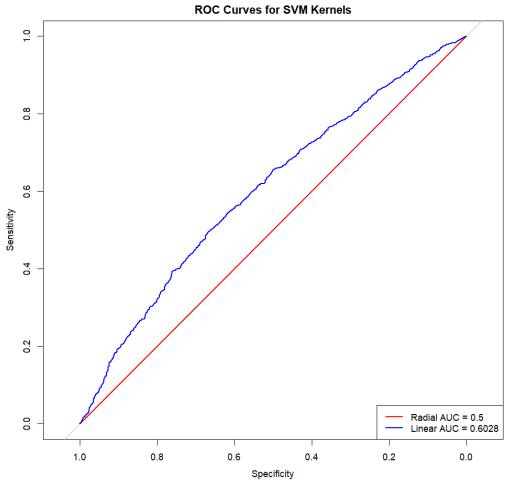



Left- ROC Curve for Neural Network Passing Data.

Right- Neural Network model for Passing Data

The next model for passing data tested was the Support Vector Machine (SVM). For this one, both linear and radial kernels were tested. The linear kernel performed better, as the radial kernel only achieved an AUC of .5, akin to guessing. This is a result of all of the predictions estimated as x1(for this model 0 and 1 are replaced by x0 and x1, respectively). The Linear Kernel was more accurate than wrong, but not by a wide margin, as its AUC came out to be .6028 and its misclassification error was .4231 rather than .4985. Thus the linear kernel is a model that we are more likely to use when predicting.




```
Confusion Matrix and Statistics

          Reference
Prediction   X0   X1
        X0    0    0
        X1 1495 1504
```

Above- Confusion Matrix for SVM Radial Kernel
Left- ROC Curve for both SVM Kernels
Below: Confusion Matrix for SVM Linear Kernel

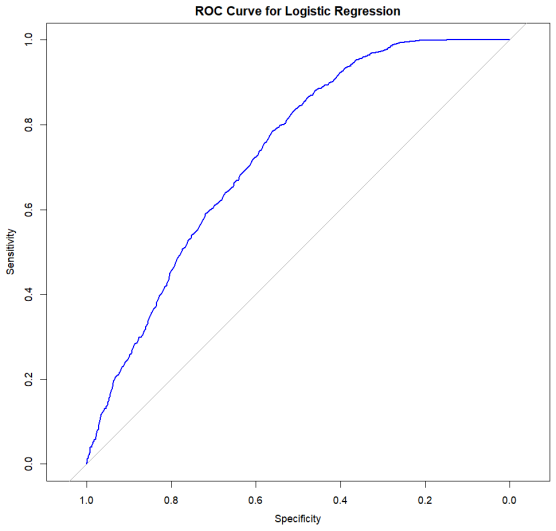
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 883 657
        X1 612 847
```

# Rushing Data:

Similar to the passing data, the first model we tested was logistic regression. We also used backwards elimination manually to decide the significant variables to maintain in the model. Compared to passing data, there are less significant variables. However, some non-significant variables were maintained in this model in order to compare with that of passing data. The rushLocation variable was very significant, with both Outside_Left and Ourside_Right being significantly associated with less successful plays, especially when compared to runs that go inside and to the left. Down was very significant. While this could have to do with the fact that down is in the equation of success (a possible design flaw with the data), it could also reflect the expectation of running on certain downs. The data suggests a running play is more successful on the early downs than the later one. While there are a few pff_passCoverage variables that are significant, their impact was weaker than it was in passing plays. This is logical as pass coverage should be more important on passing plays than rushing plays. Despite the decrease in significant variables, the misclassification error rate is much lower than that of the passing data. It turned out to be .328 on the test set and led to an AUC of .731. The ROC curves model the higher AUC of the logistic regression.

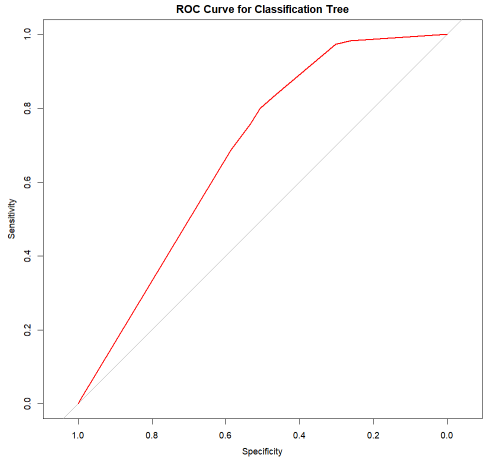
```
Coefficients:
                                        Estimate Std. Error z value Pr(>|z|)
(Intercept)                            6.875e-01  5.232e-01   1.314 0.188833
inMotionCount                          4.945e-02  4.724e-02   1.047 0.295225
shiftSinceCount                       -2.311e-02  3.524e-02  -0.656 0.511835
motionSinceCount                      -2.981e-03  4.405e-02  -0.068 0.946044
quarter                               -3.557e-02  1.951e-02  -1.824 0.068218 .
down                                  -6.352e-01  3.454e-02 -18.390  < 2e-16 ***
absoluteYardlineNumber                -1.291e-03  8.843e-04  -1.460 0.144245
rushLocationTypeINSIDE_RIGHT          -4.349e-02  6.473e-02  -0.672 0.501707
rushLocationTypeOUTSIDE_LEFT          -2.770e-01  7.414e-02  -3.736 0.000187 ***
rushLocationTypeOUTSIDE_RIGHT         -1.933e-01  7.271e-02  -2.659 0.007840 **
pff_runConceptPrimaryDRAW             -1.644e-01  1.750e-01  -0.940 0.347452
pff_runConceptPrimaryFB RUN           -2.463e+00  1.030e+00  -2.391 0.016789 *
pff_runConceptPrimaryINSIDE ZONE      -3.178e-02  1.178e-01  -0.270 0.787270
pff_runConceptPrimaryMAN               5.037e-02  1.205e-01   0.418 0.676039
pff_runConceptPrimaryNA               -4.107e-02  1.680e-01  -0.244 0.806928
pff_runConceptPrimaryOUTSIDE ZONE     -1.365e-01  1.131e-01  -1.207 0.227270
pff_runConceptPrimaryPOWER             3.164e-02  1.323e-01   0.239 0.811071
pff_runConceptPrimaryPULL LEAD        -4.271e-03  1.344e-01  -0.032 0.974656
pff_runConceptPrimarySNEAK            -2.564e-01  2.596e-01  -0.988 0.323201
pff_runConceptPrimaryTRAP              2.278e-01  2.099e-01   1.085 0.277720
pff_runConceptPrimaryTRICK             2.567e-01  1.837e-01   1.397 0.162445
pff_runConceptPrimaryUNDEFINED        -5.347e-03  2.803e-01  -0.019 0.984779
pff_runPassOption                     -9.332e-02  6.043e-02  -1.544 0.122497
pff_passCoverageBracket               -5.436e-01  1.248e+00  -0.436 0.663182
pff_passCoverageCover-0                5.903e-01  5.201e-01   1.135 0.256440
pff_passCoverageCover-1                5.460e-01  5.061e-01   1.079 0.280689
pff_passCoverageCover-1 Double        -1.266e+01  1.772e+02  -0.071 0.943013
pff_passCoverageCover-2                8.789e-01  5.097e-01   1.724 0.084637 .
pff_passCoverageCover-3                5.029e-01  5.053e-01   0.995 0.319689
pff_passCoverageCover-3 Cloud Left     5.723e-01  7.033e-01   0.814 0.415818
pff_passCoverageCover-3 Cloud Right    1.317e+00  1.052e+00   1.252 0.210748
pff_passCoverageCover-3 Seam           9.665e-01  5.288e-01   1.828 0.067616 .
pff_passCoverageCover-6 Right          1.161e+00  5.203e-01   2.232 0.025627 *
pff_passCoverageCover 6-Left           8.867e-01  5.173e-01   1.714 0.086506 .
pff_passCoverageGoal Line              6.074e-01  5.433e-01   1.118 0.263549
pff_passCoverageMiscellaneous          1.439e+01  5.354e+02   0.027 0.978559
pff_passCoverageNA                    -1.218e+01  3.180e+02  -0.038 0.969440
pff_passCoveragePrevent                1.446e+01  3.750e+02   0.039 0.969239
pff_passCoverageQuarters               6.931e-01  5.075e-01   1.366 0.172067
pff_passCoverageRed Zone               7.278e-01  5.199e-01   1.400 0.161522
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Above- Logistic Regression Results Estimates and Significance for Rushing Data

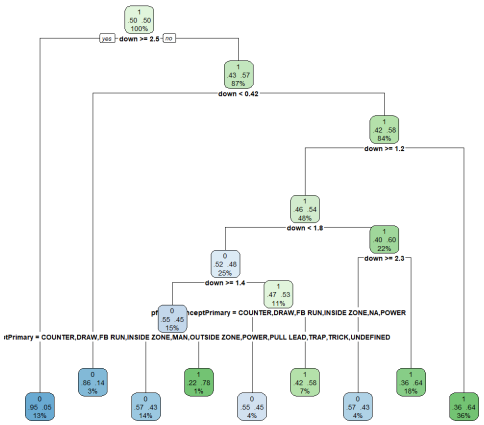



Above- ROC curve for Logistic Regression for Rushing Data

For rushing data, we also modeled classification trees. We used a similar process to that of passing data. We first just ran an unpruned tree, and obtained a misclassification error of .360. To optimize the model, we then were able to tests cp values and were able to discover that the ideal cp value is .0044, which leads to the classification tree seen below.This new classification tree then has an AUC of .682, as depicted by the accompanying ROC Curve below, and a new classification error of .354. Despite not being as efficient as the logistic regression, it is still comparable. Furthermore, we can also look at the ideal tree, to obtain some of the more important factors. The most important factors are clearly down and pff_runConceptprimary. Similar to logistic regression, down is the most important factor.

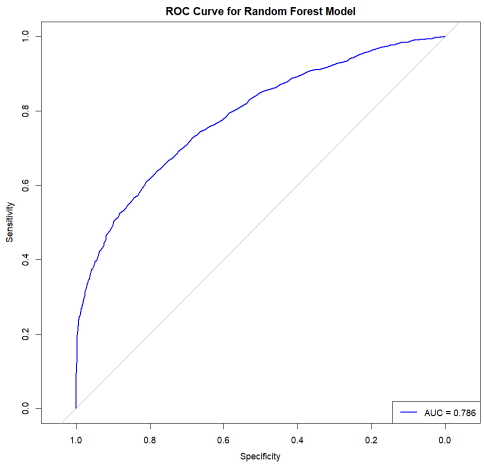


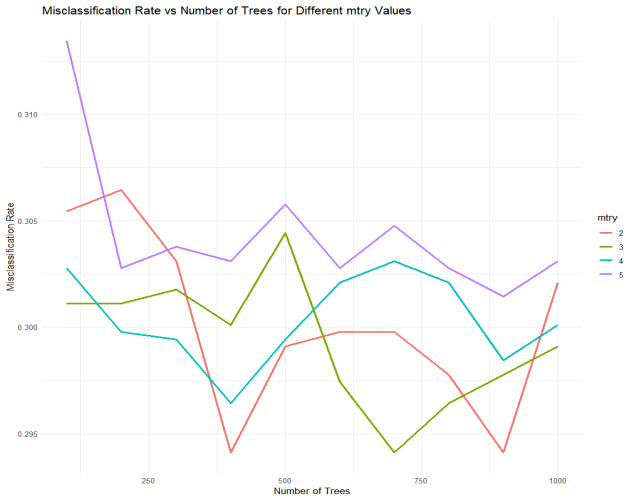

Above- The ROC Curve for Classification Tree for Rushing Data




Above-Ideal Classification Tree for rushing data

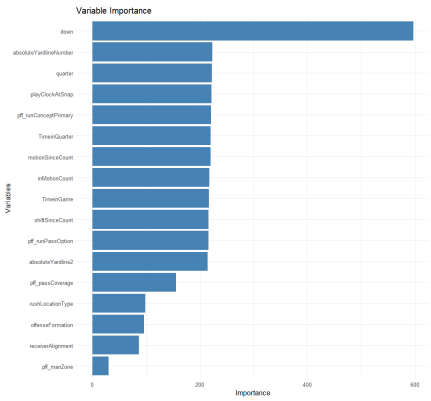
The next model that we evaluated for rushing data is random forests. First, we tested several values for tree size and mtry size. We discovered the optimal parameters were mtry=2 and trees=400. The misclassification error was .294 and the AUC value was .786. These are not only the best values for rushing, but are the best values overall.




Above- ROC Curve for Random Forest Rushing Data
Below- Variable Importance Random Forest Rushing Data

Above- Number of Trees and mtry vs misclassification Error Rushing Data



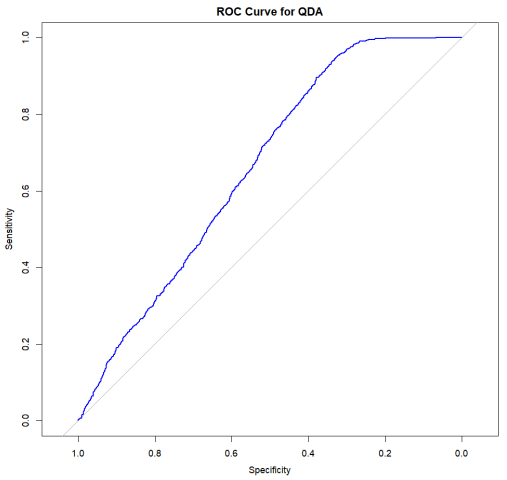

The key factor is unsurprisingly down with the highest meanDecreaseAccuracy. Some other factors that are important are quarter and absoluteYardlineNumber, though there are many variables that have the same or very similar importance. On the other hand, a few variables that had the lowest meanDecreaseAccuracy were pff_manZone and Receiver Alignment following suit of the other models previously shown.

Similar to passing data, we explored a numeric only rushing dataset in order to model Discriminant Analysis, Neural Networks and SVM more accurately and without errors caused from collinearity and NA observations. The first model we are going to evaluate on is discriminant analysis. We initiated the analysis with Box's M-Test first in order to discover which of the LDA OR QDA we need to use. The produced p-value is less than 2.2*10^-16, from which we rejected the null hypothesis of homogeneity in the covariance matrices.

```
[1] "Box's M Test Results:"
> print(box_test_result)

        Box's M-test for Homogeneity of Covariance Matrices

data:  train_data3[, -which(names(train_data3) == "Success")]
Chi-Sq (approx.) = 1190.2, df = 55, p-value < 2.2e-16
```

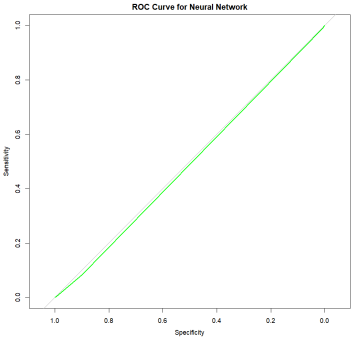
Left- Box's M-Test for Rushing Data

Consequently, as with passing data, to use Quadratic Discriminant Analysis instead of Linear. Running the Quadratic Discriminant Analysis achieved a misclassification error rate of .3748, which led to an AUC of .669. That is further shown in the graph below as the ROC curve is much higher than the .5 line which is attributed to guessing. While this model did not have the same predictive power as the random forest, it still has similar success as the other models (logistic and classification trees).




Right- ROC Curve for QDA Rushing Data

The next model we tested was a neural network model. Unfortunately, the model proved to be largely ineffective. The test set misclassification error rate was .5096, which is worse than random guessing. A possible explanation for this is overfitting, as there was much better performance when using cross validation. The lack of accuracy in guessing is depicted in the confusion matrix below, and supported by the ROC curve. The curve depicted is very similar to a line, and is below the line that signifies guessing.

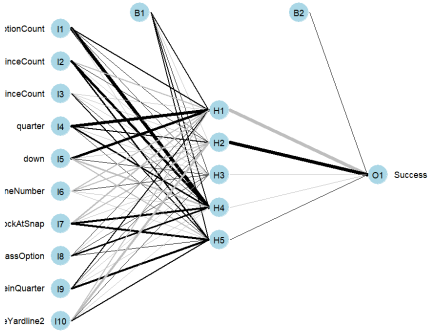



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1345 1378
         1  150  126
```

Left-Confusion Matrix for Rushing Data Neural Networks

- Right- ROC Curve for Rushing Data Neural Networks

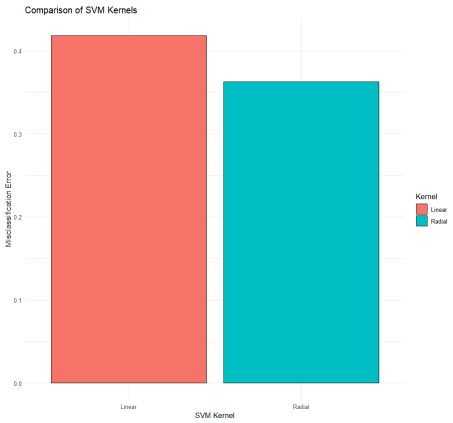
While there appear to be some important factors, namely quarter, shiftSinceCount, inMotionCount and playClockAtSnap, the truth of their importance is in question due to the poor model performance.The ones that are not important appear to be absoluteYardlineNumber, absoluteYardline2, and motionSinceCount.



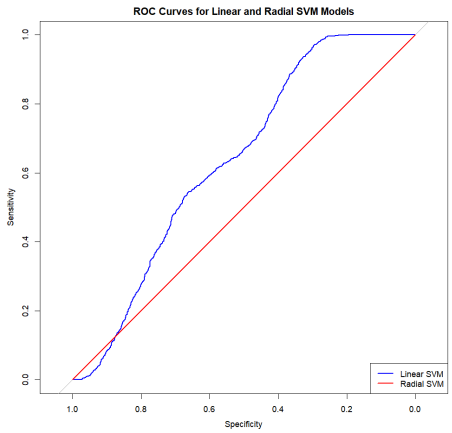

Right-Neural Network model Rushing Data

The last model that we are going to discuss is the support vector model for rushing data. Similarly to both the neural network for rushing and the SVM for the passing data, we are able to see that the radial model itself was not effective in predicting on the test set with a misclassification error of .4985 and a AUC value of .5. This could be due to overfitting as the radial model did better on the cross validation error. The linear model did much better on the test set despite doing worse on the cross validated set. It had a misclassification error of about .39 and an AUC value of .6. Thus the linear model is what we would use to put this in practice.

Below- Cross Validation SVM Linear vs. Radial Kernels Rushing Data




Below- ROC Curve for Linear and Radial SVM Models Rushing Data




## Discussion:

| | Passing Misclassification Rate | Rushing Misclassification Rate |
|---|---|---|
| Logistic Regression | 0.4725 | 0.329 |
| Classification Tree | 0.437 | 0.354 |
| Random Forest | 0.386 | 0.294 |
| Discriminant Analysis | 0.389 | 0.3748 |
| Neural Network | 0.4768 | 0.5095 |
| Support Vector Model | 0.4231 | 0.4021 |

The misclassification rate on the test set for all models is summarized in the chart above. The misclassification rate for support vectors models are those for the linear models as their performance was better than the radial models. On every model outside of the neural network the rushing models had higher predictive power, indicating that rushing plays are more accurately predicted. Random forest also has the lowest misclassification rates for both the rushing and passing data, and this is what I would use if necessary to predict again. The most significant variable for the passing data was playAction, which when it occurred helped teams be more successful. PassCoverage cover-3 and quarters  were also significant and allowed an offensive team to have a higher chance of a successful play. In addition, the quicker the quarterback got the ball

out of his hands, as displayed by dropbackDistance and timeToThrow, the more successful the play as well. Thus, I would inform coaches to use this data to make their game plan as this can assist them in having more successful plays. The other most important variables for the rushing data are the down and the runLoaction. We can use this information in the future in order to refine our models further. We can run further models and focus more on the variables that are impactful, and put less focus on the variables, such as man or zone defense, and timeInQuarter, as these weren't significant in either model. It is also important to explore in future models how the "down" variable affects the success rate for rushing. It may be too related to success rate, as it is involved with the formula, and that is why it is such an impactful variable amongst all models. It could also be that the formulas aren't accurate- that a success on 1st and 2nd down should require more yards than just 40% and 60%, respectively. Another possible explanation is that 1st or 2nd down runs are more successful than 3rd or 4th down runs, due to how the defense and offense are set up on these plays. As there are many possible explanations, it is important to investigate in further detail in order to understand exactly what the impact of down is. Another thing that should be investigated in further detail is the problems with the one-hot encoded data. As the presence of the variables caused errors in discriminant analysis, neural networks and support vector machines, numeric only data was used in order to use these models. Thus, troubleshooting more and understanding the errors more could help to fix this issue, so that the data tested can be uniform throughout all of the models. In addition, while there was investigation done with squared terms with absoluteYardline2, targetX2 and targetY2, we can further look into interactions between variables and what they indicate about a play. While some models like the neural network look into this data itself, the only numeric variables in this model reduce the amount of interaction between variables. For example, it could be extremely helpful to look into how the offensiveFormation interacted with the defensive coverage metrics, (both are character/factor data) as this could give more insight into what defensive plays counter which offensive plays and vice versa. Furthermore, at best, there was only roughly 70% accuracy when using the best model to predict play success. As football has 22 people on the field at a time, a lot of the accuracy can be related to the players' skill level not to mention the skills and history associated with each coach and team. Thus, had these variables been applied to the model, we expect the accuracy to improve over time. However, it is important to remember that as sports have 22 players playing at a time, with coaches trying to trick and out-do each other, a perfect model won't exist. There will always be human error on a random play. Thus, our goal is to continuously refine the model, and just gain a little better understanding of what is going on, to give our team or organization an edge.

References

*Introducing Chatgpt*, openai.com/index/chatgpt/. Accessed 10 Dec. 2024.

“ML: Handling Imbalanced Data with Smote and near Miss Algorithm in Python.” *GeeksforGeeks*, GeeksforGeeks, 14 Aug. 2024, www.geeksforgeeks.org/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/.

“NFL Big Data Bowl 2025.” *Kaggle*, NFL, www.kaggle.com/competitions/nfl-big-data-bowl-2025. Accessed 10 Dec. 2024.