NFL Pass Rushing Statistics: Iterative and Cross-Validating Modeling Methods to Determine Number of Pass Rushers from NFL NextGen Stats

Joshua White
University of Central Florida

***Abstract*—By utilizing the 2020-2021 NFL NextGen Stats, segmented for passing data, numerous modeling techniques are conducted to determine the optimal model to predict the number of pass rushers from analyzing the on-field pre-snap formation and game statistics to make predictions in real time.**

***Index Terms*—Grid Search, Cross Validation, Model Selection, Generalized Linear Models (GLM), Support Vector Machines (SVM), Random Forrest, Gradient Boosted Decision Trees, XG-Boost, NFL NextGen Stats**

I. Project Motivation and Objectives

Each year the NFL introduces a data challenge on Kaggle. The NFL utilizes their NextGen Stats platform to record the motion of all players on the field (position, velocity, orientation, etc.) as well as the location of the football. All of this information can be used in conjunction with one another to result in some powerful geo-spatial analytics and statistics.

In Data Mining I, I performed in-depth analysis on the likelihood of a kickoff resulting in a touchback based on the player location. I used several different techniques (LDA, Logistic Regression, Lasso, and Ridge Regression). I extended this data set for Dimension Reduction in Regression by applying several dimension reduction techniques (PCA, Kernel PCA, MDS, LLE, and Diffusion Maps) to see if the data can be reduced in to result in a more efficient model. In the end Logistic Regression with Stepwise Dimension Reduction was used to produce the most effective and significant model in using player's geo-spatial game data to predict whether or not a kickoff would be returned.

My intent for this project is to continue this type of analysis to predict the success of a defensive formation in measuring positive and negative defensive outcomes (sacks, large offensive gains, offensive touchdowns, etc.). While some of the similar methods will be used to model the data, I expect to use different models for both exploratory and predictive analysis.

With the plethora of information provided about every passing play of every game of the 2020-2021 NFL season, there are plenty of insights that can be extracted from the data. In the proper format and with the proper models set up on the data, valuable information can be extracted about how teams perform on the field. In the era of sports analytics it is reasonable to anticipate an NFL team wanting to use these specific insights to make predictions on a type of play a team is likely to run.

This data set is pre-segmented for passing plays in the NFL, so the extracted insights must answer questions as it pertains to a play related to a pass. Upon examination of the data I noticed that the number of pass rushers is recorded on every play. This may be a valuable metric for the offensive team to be able to predict, as it provides information as to how much time the quarterback may have to throw the football before receiving pressure. A team may be able to use this prediction to organize their offensive scheme to respond to the number of players that rush the quarterback.

Several other features within the data set can be used to predict the number of players that will rush the quarterback. Since the NextGen stats provides the player locations and player details of each player on defense, these locations will be used to predict the number of players that rush the quarterback. Additional game statistics are also used as additional features. All of this aggregated data is then modeled to make the prediction. If this data can be modeled to provide an accurate prediction of the number of pass rushers, this would provide a reasonable predictive method for NFL teams to use to coordinate their game plan.

The data as provided is not in the format to begin modeling or even to assess, so it must first be formatted in a tidy format for analysis. All relevant table information will be joined to a single table. Irrelevant information will be removed from the data set. Additionally any inconsistencies will be adjudicated. The output table should be in the format of all passing plays with all relevant features in a tabular format. This will allow for seamless analysis and modeling.

Once the data is properly formatted, initial analysis will be conducted. A general understanding of the predictors is very important. It is necessary to understand if the output behavior of the number of pass rushers should be treated as a continuous random variable and regression methods should be used, or if it is better suited by utilizing a classification model. By creating histograms, the data can be analyzed. It can further be analyzed by performing different clustering methods to determine if the data fits into a specific bucket of classifications.

Once the format of the response variable is properly determined, then specific modeling techniques will be used. These models will all be tuned to find the hyperparameters associated with each of the respective models. From there the models will be trained and tested. The corresponding accuracies and errors will be determined for each of the models. The best model will

then be used to perform some post-modeling diagnostics on the model.

All of this is intended to perform the best model for predicting the number of pass rushers based on a given pre-snap field formation. Ideally the model selected can be used in real-time to predict the number of pass rushers for the corresponding play.

II. Explanation of Dataset

The 2020-2021 NextGen Stats files contains a file for every single game, player, play, and frame of every passing down throughout the season. The data contains both offensive and defensive player metrics associated with each of those plays. Below is a listing of the tables provided from the NextGen Stats.

- games: A summary of all game data throughout the past year. This includes high level data points including the date and time of the game, and the teams participating in it.
- plays: All plays conducted within the past year, which can be mapped to each game by the gameId index. This contains metrics like the time of play, down, yardage, and other summary metrics. However, this does contain more specific data such as kickBlockerId (the ID of the player who blocked a kick) and penaltyJerseyNumber (the ID of the player who got penalized) which will be used in analyzing and classifying particular data.
- players: Details of all players who played within an NFL game within the past year, inclusive of their height, weight, data of birth, and position that they played.
- tracking: The physical position of each player on the field during an NFL game. This data is available for all 17 weeks (in a seperate file for each) of the NFL regular season, and contain a significant portion of the population data. The X and Y coordinates are shown for each player on the field for every 0.1 seconds of time during the game. The coordinate plan for the data is shown in Fig. 1.

A more detailed listing of each of the predictive variables can be found here at the challenges official website [4].

III. Data Preparation

Due to the size of the data sets, it is most efficient to store the data from each of the data sets locally. The games, plays, players, and all 17 tracking files are stored to a local repository for data extraction. Each of the files are read into the R to prepare for analysis.

The games and players tables are read in as is. The plays table is imported with three additional columns added the possession indicator, score differential, and minutes remaining columns. Each of these are calculated columns from other features within the plays data table.

The tracking table is created by iterating through each of the season's weeks of tracking data. Each week is binded to the previous weeks tracking data. In order to get a meaningful metric for the position on the field, it is necessary to measure the field as distance from the possessing teams goal line. This is done by inverting the field if the possessing team is facing the left as indicated in Fig. 1.

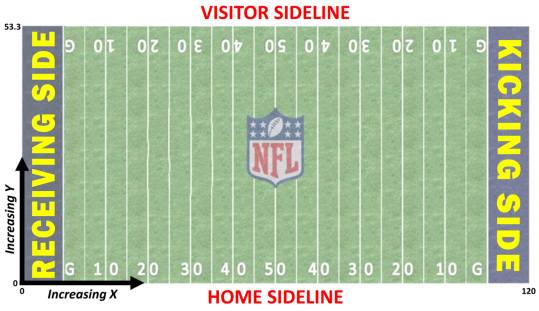

Fig. 1: Adjusted field diagram.

In order to perform analysis of the pre-snap defensive metrics, it is critical to know the location of the football. This is not stored natively in any of the tables. Instead it can be extracted by taking the location of the ball for each play prior to any noticeable movement (0.02 yards in any direction to account for any rounding or measurement errors). The frame ID is also taken at this point to take the location of all of the players at the moment before the snap.

In examining the number of defensive players, right before the snap a discrepancy is noted. The number of players tracked each play for defense is not always equal to 11 (the number of players on the defensive side of the ball). In fact, most plays from this data set only track 7 defensive players as indicated in Table I.

TABLE I: Number of Tracked Defense Players by Play Count

Players	Play Count
4	1
5	30
6	124
7	10158
8	5031
9	3113
10	699
11	72
16	1
17	1
18	1

In order to ensure consistent data without missing values, only plays that track between 7 and 11 defensive players will be used. Any plays that track more than 7 defensive players, 7 defensive players will be taken at random and used for analysis. Of these 7 defensive players the Euclidean distance of each of the players from the football are recorded as a feature value within the table.

The Euclidean distances of each player to the football will be ranked for each play. These rankings will be to pivot the data in a wide format to include player metrics of each feature with suffix values of 1 to 7. from This outputs a table with each play as a row and the columns containing all of the relevant game, play, and player features for that given play.

Random variability may be introduced by randomly selecting the defensive players from the data set and using the

rank of the Euclidean distances from the football as a feature label. However, due to the limitations of the number of players tracked and the inconsistency of the position assignments in the player data, this is determined to be the best method.

IV. Exploratory Analysis

Dimension Reduction

The plays training data set has 12,931 rows and 80 columns (79 independent variables and the dependent variable number of pass rushers). This becomes quite a computationally expensive modeling problem, especially considering the need to tune the hyperparameters via grid search cross validation. The first thing to be investigated is whether or not the dimensionality of the data set can be reduced in a systematic manner.

First t-SNE will be instantiated to determine whether the number of pass rushers can be defined on a 2-dimensional space from its 79 feature set of pre-snap data. After running t-SNE, the first two dimensions are plotted with the corresponding values for the number of pass rushers in Fig. 2.

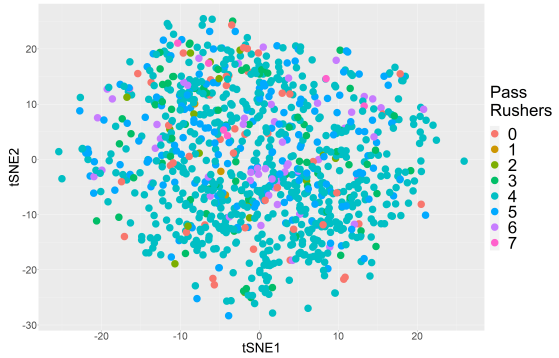

Fig. 2: t-SNE clustering of the plays dataset.

It is seen that there is no clear separation among the number of pass rushers in the two-dimensional subspace. 4 pass rushers sprawls across all clusters of pass rushers in the two-dimensional space, so there is no way to segment it accordingly.

PCA is then performed on the data to see the linear behavior of the data when projecting the 79 dimensional data set on a lower dimensional subspace. The variance explained by each of the principal components is extracted from the prcomp function and plotted accordingly. The output is depicted in Fig. 3.

From Fig. 3 it is noted that the given data does not accurately define the data in lower dimensions. Projecting the given pre-snap data to 2 dimensions only explains 17% of the variance of the data. Plotting the data in 2 dimensions using PCA yields a similar result of the t-SNE method in two dimensions. The plot of the first two dimensions of PCA and corresponding number of pass rushers is plotted in Fig. 4. While there are some outliers within the data, there is no indication of proper segmentation of the data to determine the number of pass rushers in lower dimensions.

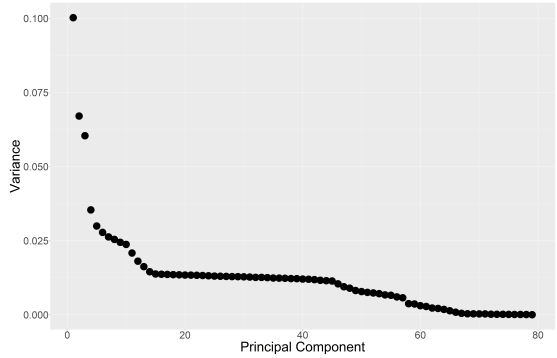

Fig. 3: Explained variance plot for each principal component of plays.train.

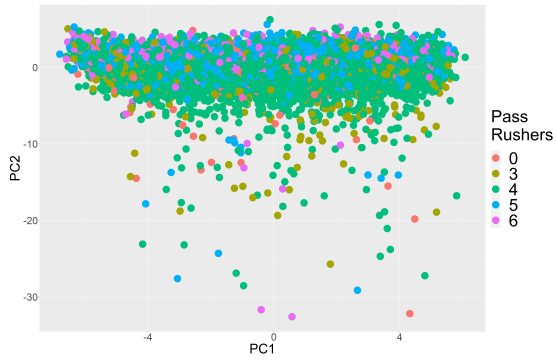

Fig. 4: Plot of first two principal components of the plays.train dataset.

Variance of this pre-snap data is not well defined in smaller dimensions. In fact, it takes 30 PCs to explain 70% of the variance. Thus there is no effective to reduce the dimensions of the given data systematically.

Diagnostics and Clustering

Another important step in prior to the model selection is further investigation of the predictor variable. Since the number of pass rushers is a small set of count based values, it can either be interpreted as a regression variable or a classification variable. Performing clustering on the given data with a label as the number of pass rushers, some investigation on the number of pass rushers will assist with the model building.

For the cluster analysis, K-Means is chosen as the clustering method. The amount of clusters, K, is optimized by finding the loss associated with each value of K and its corresponding silhouette score. This is calculated using the sil_sweep_kmeans function [3] to calculate the values at each value of K. This is then plotted for each value of K from 2 to 10 in Fig. 5.

The elbow plot from the k-means method shows a slight dip when k=5, and that is before a large dip in the silhouette model. Thus making 5 clusters a number of possible classifications for the number of pass rushers. It can also be seen from the plots that k=9 and k=10 are possible candidates for clustering. However, since there are only 9 values for the

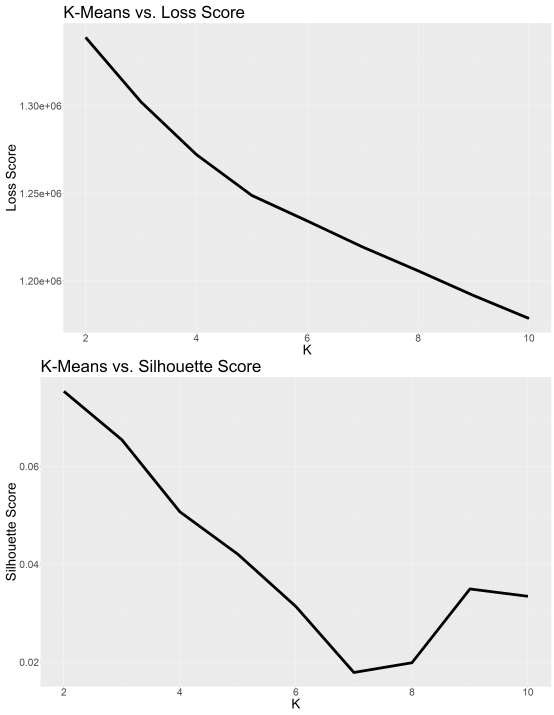

Fig. 5: Elbow plot and sillhouette plot of the kmeans of all plays data.

number of rushers in the data set, this would signify the need for a regression model.

Thus from the data only 5 classifications of data will be considered in each of the models, but both classification and regression models will be considered for predicting the number of pass rushers.

It is determined that the number of classifications should be 5, the data is examined to determined the most appropriate segmentation. Fig. 6 shows a histogram of all of the values for the number of pass rushers for all given passing plays.

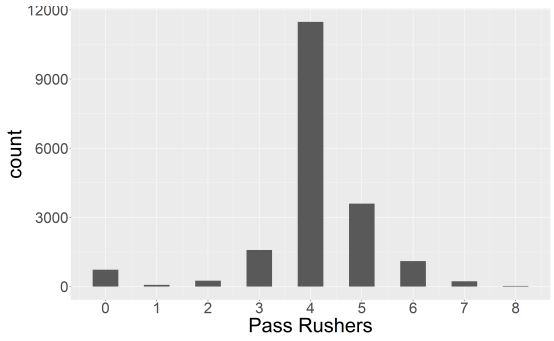

Fig. 6: Histogram of the number of pass rushers of all plays.

The data is heavily skewed towards 4 pass rushers. It is also noted that there are very few values for plays with 1, 2, 7, or 8 pass rushers. Since there is supporting evidence from K-means clustering to show that 5 classifications is justified within the data, those predictions with few values will be removed from the data set. The grouping from the actual data shows a model with 9 groupings (rushers from 0 to 8) but 1,2,7, and 8 are sparse so they will be ignored to improve the accuracy of the predictions.

This step is deemed to be crucial for the model building process. All of the models were initially ran without determining the predictor variable to have 5 classification levels. There is a heavy skew in the number of pass rushers with 4 pass rushers as the majority, with very few with values of 1, 2, 7, or 8. Thus when all of the models were initially trained with all 9 levels of the number of rushers being in the training set, it caused most models to train the data to nearly always predict 4 pass rushers. By modifying the number of classification levels for number of pass rushers within the training set to 5, it allowed for a lot more accurate predictions for number of pass rushers not equal to 4.

A. *Partitioning*

Upon proper segmentation of the overall plays data tables, the data is partitioned to training and test data sets for analysis. The training and test data sets are partitioned from the plays.regression data set. 70% of the data is used for training, and 30% for testing. Fig. 7 shows the histogram of number of pass rushers from the plays.train data set.

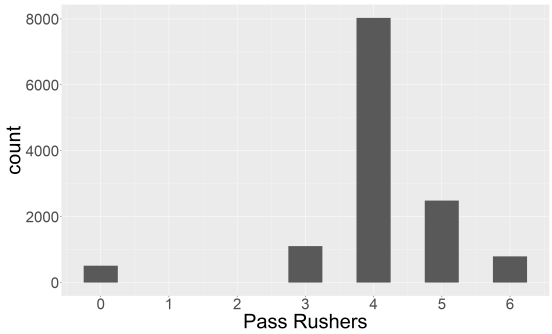

Fig. 7: Histogram of the number of pass rushers in the adjusted training data.

It can be see that the predictor values heavily skewed, the model will need to account for the all values within the data set proportionally, and modeling data can lead to an oversaturation of predictions with a predictor value of 4. Thus this will have to be taken into consideration during the model selection process.

It will be difficult to associate this data with a known linear model. The data behaves in a quasi-binomial manner, however due to the heavy skew on 4, it will be difficult to make an accurate prediction with any type of binomial GLM model.

Finally, it is important for the corresponding models to have values for each of the number of rushers. Table II shows that there are an adequate amount of counts from each of the remaining classifications for the number of rushers.

V. Model Selection

Now that the data has been filtered for only records with the top 5 most frequent number of pass rushers for a given

TABLE II: Records for the Train and Test Partitions

Number of Pass Rushers	Train	Test
0	510	220
3	1106	481
4	8032	3447
5	2488	1110
6	795	309

play, the modeling can be conducted. Based on the behavior of the output data, 3 families of models is used to predict the number of pass rushers for a given play: generalized linear models (GLM), support vector machines (SVM), and tree based models. For each of these families of models, the hyperparameters of each model is determined using grid search cross-validation. Once the proper parameters are determined each one of the models are trained, and their corresponding accuracy and errors are recorded for each of the trained models.

Since this process is quite iterative, I created a package called buildmodels and stored it to my GitHub [1]. This package has function called buildmodels which takes in a list of models, optimizes the hyperparameters via grid search, learns the models with those parameters and then tests the models based on their hyperparameters. This reports the corresponding accuracy and error for each of the models in the list. The buildmodels function calls on the gridCV function also in the buildmodels package which performs a k-fold cross-validation over a grid search of all of the defined ranges of hyperparameters. This allows for diligent determination of the best models of all of the models passed through the buildmodels function. Table III defines each of the models passed through the buildmodels function with the corresponding defined parameters and tuned hyperparameters.

TABLE III: Determining the Hyperparameters

Model	**Parameters**	**Hyperparameters**
GLM	Default	Family: Gaussian
Hurdle	Default	0 Distribution: NegBin Distribution: NegBin
SVM Linear	Cost = 0.1	None
SVM Gaussian	Cost = 0.1	Gamma: 0.01
Random Forest	Trees = 500 Tree Depth = p/3	None
XGBoost	Rounds = 500 Max Depth = p/3	Eta: 0.001

Each one of these models are passed through the gridCV function of the buildmodels package to determine the optimal hyperparameters. The values to the right of the colon in Table III under the Hyperparameters indicate the resulting value for each of the hyperparameters of that model from the gridCV function. For the GLM and Hurdle models, the distribution families were tuned as hyperparameters. It was determined that the optimal model for GLM was the Gaussian model, and for the Hurdle model it was the negative binomial for both the value of 0 or greater and for the non-zero values of the number of rushers.

Due to the computational complexity of the models on the data set, the cost was assumed for the SVM models, and the number of iterations and tree depth was assumed for the tree models. While it was not found via grid search cross-validation, it was manually cross-validated through a few iterations of each of the models, and determined the strongest tested parameters for each of the respective models. The SVM Linear model and the Random Forest model had no other tuning parameters. SVM Gaussian and XGBoost both have learning parameters that are tuned with magnitudes of 10. Those models are optimized when gamma is 0.01 and eta is 0.001 for the respective models.

Upon tuning of the hyperparameters, the buildmodels function continues to train the models based on the hyperparameters and test the model with another dataset with the same parameters. Both a plays.train and plays.test data set are passed through to the build models function. Each of the models are learned by passing the training data through the model with the corresponding parameters and optimized hyperparameters. That learned model is then used to predict the results for the number of pass rushers on the training data set.

The best model for determining the number of pass rushers can be determined by two key characteristics. The first is if the model has the highest percentage of correctly predicted values for number of pass rushers. This is if the model prediction from the given data within the test data set yields the correct number of pass rushers in the test data set. For regression models (GLM and Hurdle), the prediction is a numeric value which may contain a decimal. This value is rounded to the nearest whole number in this case and compared to the actual number of pass rushers. The average number of matches is deemed to be the accuracy of this model. Another measure of the success of the model is the mean square error of the models. This is calculated by the average of the square of the differences in the actual number of pass rushers and the predicted number of pass rushers in each model. This is a measurement of how far each one of the models are from predicting the actual number of pass rushers. Both the accuracy and MSE are important measurements of the usefulness of the model, and each of the evaluated models are recorded with their corresponding Accuracy and MSE in Table IV.

TABLE IV: Model Accuracy and Errors for All Tested Models

Model	**Accuracy**	**MSE**
GLM Gaussian	0.53	0.82
Hurdle (NegBin, NegBin)	0.62	0.62
SVM Linear (Cost = 0.1)	0.62	1.00
SVM Gaussian (Cost = 0.1)	0.61	0.95
Random Forest (Trees = 500, Tree Depth = p/3)	0.64	0.50
XGBoost (Rounds = 500, Max Depth = p/3, Eta: 0.001)	0.65	0.49
Predicting Only 4	0.62	1.14

Predicting Only 4 was added as another model to Table IV. This is not a model that was learned, but instead used as a baseline for predicting the number of pass rushers from the given data. Since the data for the number of pass rushers

is heavily skewed in favor of 4 pass rushers, a reasonable prediction would be that there is always going to be 4 pass rushers. While this prediction does yield an Accuracy score that is similar to the other models, the MSE is the highest amongst all of the models; which is 14% greater than the next closest model (SVM Linear). This shows that performing a model-based approach does help predict the number of pass rushers closer to the true amount.

Ideally, the accuracy of the model would be high, while the MSE of the model is low. This is the case for the XGBoost model. Both Random Forest and XGBoost correctly predict the number of pass rushers with a higher prediction accuracy than that of always predicting 4, and the MSE is significantly lower than all of the other models. But XGBoost does perform slightly better than that of the Random Forest model, so that will be chosen as the ideal model for predicting the number of pass rushers from the given data of a pre-snap play.

VI. Analysis Results

With there being 5 numerical values for the number of pass rushers, XGBoost correctly predicts the actual number correctly 65% of the time. The MSE of this model is 0.49, meaning the root mean square error (RMSE) is 0.70. This states that the predicted number of pass rushers is approximately off by 0.70 from the actual number of pass rushers. These numbers show that the model is quite useful for an NFL Analyst. To further analyze the accuracy of this model, the confusion matrix is evaluated as Fig. 8 below [5].

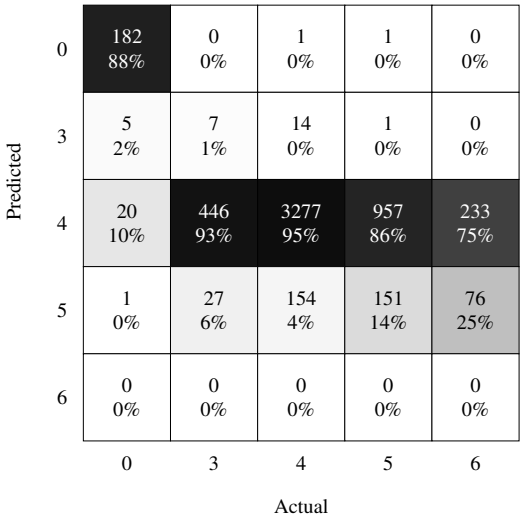

Fig. 8: Confusion Matrix of XGBoost

Fig. 8 shows that even the best performing model still heavily favors 4 pass rushers. 4 pass rushers was predicted 89% of the time using the XGBoost model on the test data set. However, it is the Accuracy and MSE that are the measurements of success for this model. When the actual number of pass rushers was 4, XGBoost correctly predicted it 95% of the time. It is important to note that the XGBoost model also correctly predicted that there were 0 pass rushers 88% of the time. While the XGBoost model did not correctly predict 6 pass rushers from the 309 plays that it occurred in the test data, it did predict that those plays had 5 pass rushers 25% of the those plays and 4 for 75% of those plays, thus resulting in minimal increase in the MSE. Similar behavior can be observed when there was 3 or 5 pass rushers.

Variable Importance

The confusion matrix, accuracy, and MSE for XGBoost has shown that the model performs quite well in predicting the number of pass rushers. Now it is a point of interest in determining which features from the data contribute the most to the model. Fig. 9 depicts the top 10 most important features of the XGBoost decision tree in terms of percentage gain.

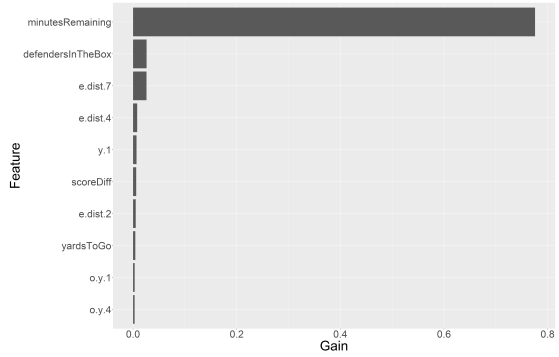

Fig. 9: Most important variables of the Gradient Boosted Decision Tree.

Here it can be seen that the minutes remaining in the game is the most important variable in predicting the number of pass rushers for a given passing play in terms of gain to the XGBoost model. This separated by a large margin of gain to the model's next most significant features in predicting the number of pass rushers: defenders in the box, the Euclidean distance of the 7th furthest defender from the football, and the Euclidean distance of the 4th furthest defender from the football. Fig. 10 shows the correlation plot of each of these features, and the dependent variable, number of pass rushers.

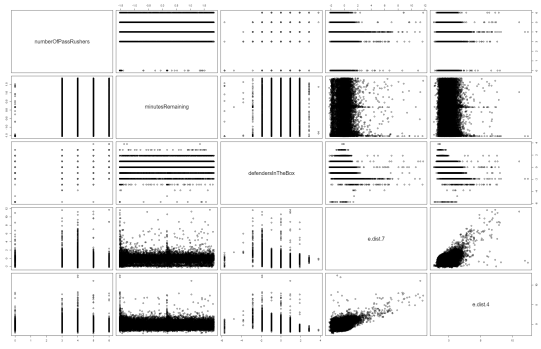

Fig. 10: Correlation of the most important variables of the tree.

It can be seen by Fig. 10 that despite the minutes remaining having quite a significantly variable importance to the model,

there appears to be no direct correlation between minutes remaining and the number of pass rushers. There does seem to be some sort of functional relationship between minutes remaining and the Euclidean distances (both 4 and 7). With this information the model was ran including only the minutes remaining, and without the minutes remaining. Both greatly decreased the accuracy and increased the MSE of the predictions.

This confirms the exploratory analysis, that the variance of the predictions is best explained in the large dimensional space, and the multivariate behavior of the model leads to a more accurate prediction of the number of pass rushers for a given play.

VII. Conclusion

The XGBoost Gradient Boosted Decision Tree was determined to be the most accurate model in predicting the number of pass rushers from the provided pre-snap game data, play data, and geo-spatial defensive player metrics which yielded a correct prediction of the number of pass rushers 65% of the time. It was also provided accurate prediction for near misses with a minimal MSE of 0.49 (0.70 RMSE) pass rushers. This could certainly be beneficial to NFL teams in predicting defensive schemes. While there have not been any studies on how the model performs against subject matter expert predictions, the model performs significantly better than predicting the average number of pass rushers for each play.

This method also performs in near real time. The model is captured and 2 predictions from the given data in 0.12 seconds. Assuming that the data is readily available and runs in near realtime this model could be used as a real time predictive analytics tool.

All code used for analysis and modeling can be found on my GitHub [2].

VIII. Future Work

The model performs fairly well despite having data limitations. Obtaining a data set of all 11 defenders could help in reducing the variability of the independent variables and increase the accuracy and reduce the error of the prediction of the number of pass rushers. Additional player characteristics could also be introduced as additional features (i.e. weight, heigh, speed) to provide more independent variables to provide more options for the model to take into consideration. If player positions were considered in a more consistent manner, they could be used as column labels instead of the Euclidean rank from the football, which would reduce the collinearity of the position columns.

Finally some alternative models and dimension reduction techniques could be examined to make better predictions on the number of pass rushers. My model selection was limited to GLM, SVM, and tree models and my dimension reduction techniques only examined t-SNE and PCA. Introducing sparse models and dimension reduction techniques may better associate the given data with the dependent variables.

IX. References and Code

My Code

[1] Joshua White. buildmodels. https://github.com/joshuaderekwhite/buildmodels, Apr 2022. Package built for this project to learn and predict models.

[2] Joshua White. nfl-pass-rushing. https://github.com/joshuaderekwhite/nfl-pass-rushing, Apr 2022. Code used to generate all models and content in this document.

References

[3] Mitchell Hill. kmeans_gmm.rmd. https://webcourses.ucf.edu/files/92680364/, Apr 2022. sil_sweep_kmeans function.

[4] Kaggle. Nfl big data bowl 2021. https://www.kaggle.com/competitions/nfl-big-data-bowl-2021, Apr 2022. NFL NextGen Stats Data.

[5] vsantos. Follows a solution used for a paper about a neural network based classifier... https://tex.stackexchange.com/a/572545, Nov 2020. Design for Confusion Matrix.