Modeling Target Probability to Assess Defensive Players in the NFL using XGBoost and GLMM

David Wonesh

May 2024

1 Introduction

Starting in 2018, the National Football League hosts an annual "Big Data Bowl" competition in which the NFL provides contestants with a dataset and each contestant submits a proposal for unique and impactful approaches to measure relevant metrics. One metric that was introduced from these competitions was target probability. Target probability is the probability that, for any given play and at any moment within that play, a particular player will be targeted for a pass thrown by quarterback. Modeling target probability is a relatively new task. The first approach I saw to model target probability was in Burke (2019), where Burke used deep neural networks to predict which offensive player would be targeted for a pass from their quarterback. Yurko et. al. (2019) built upon this and used XGBoost and LASSO regression to create a target probability model, however, their target probability model was used as a sub model for their much larger "Ball-Carrier Model".

Both of these authors used deep machine learning methods to create their target probability models. These deep machine learning methods are very susceptible to over fitting and we lose a lot of inference, but we gain more predictive power. Is it possible to use a model such as a GLMM, which is more robust to over fitting and gains a lot more inference? Additionally, these authors were only concerned with the offensive players. What about the defenders on the play? Defenders are currently evaluated by metrics where they are targeted such as the number of interceptions and pass breakups, but what about averting the targets? This paper explores modeling target probability using XGBoost and a GLMM and compares both methods. Additionally, we will attempt to model the target probability for defenders on a given play, not the receivers on offense.

2 Description of the Data

The data that will be used in this paper is player tracking data from all pass plays from the 2018 regular season. Yurko et. Al. (2019) provides an excellent description of this data. Essentially, computer chips are placed in each player's shoulder pads and the ball. These chips emit a signal to sensors that track the location of the chip on the field. The data is recorded 10 times per second and measures the location (in x and y coordinates), speed, and direction of each player and the football on the play. Each observation is put into a frame and given a frame ID. The data also provides event annotations for each frame (ball snapped, pass thrown, pass complete, etc.) Thus, the frames in this dataset act like frames in a movie reel. If we were to plot all the data points within each frame and plot the frames successively, we would get an animation of one play from the data set. Provided by Yurko et. Al. (2019), Table 1 is an example of the tracking data for a 47-yard touchdown run by wide receiver Cordarrelle Patterson and figure 1 shows what the data observations look like when you plot them on a football field.

frame.id	**x**	**y**	**s**	**direction**	**event**	**displayName**
24	60.64	29.70	7.55	175.34	handoff	Cordarrelle Patterson
25	60.77	28.94	7.61	177.10	NA	Cordarrelle Patterson
⋮	⋮	⋮	⋮	⋮	⋮	⋮
44	55.20	14.62	8.92	226.45	firstContact	Cordarrelle Patterson
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 1: Example of tracking data for Cordarrelle Patterson's 47-yard TD run.

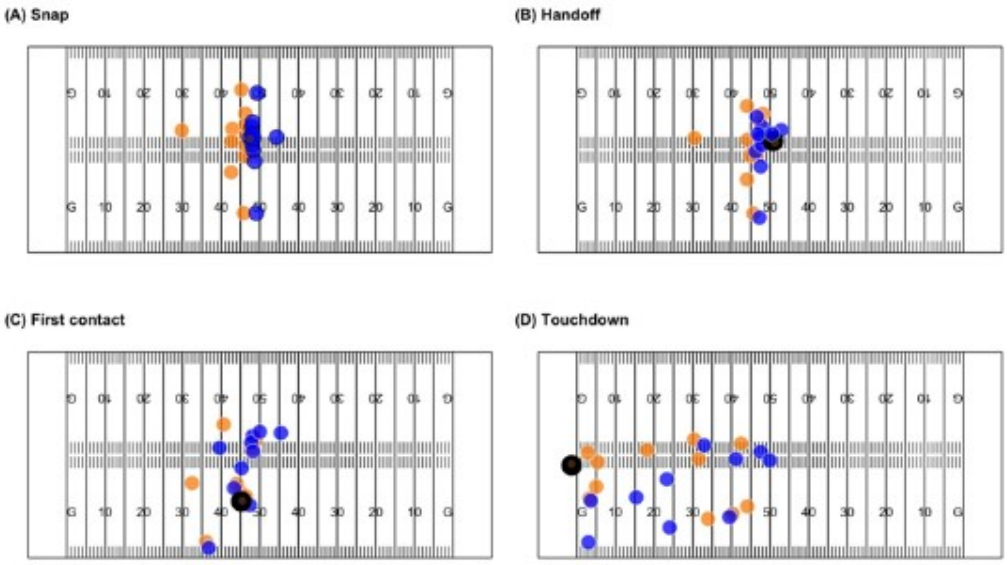

Figure 1: A display of the tracking data for Cordarrelle Patterson's 47-yard TD run with the offense (blue), defense (orange), and ball-carrier (black) at (A) snap, (B) handoff, (C) first contact, and (D) crossing the endzone. Figure provided by Yurko et. Al. (2019)

Play-by-play data was also added to this dataset. This includes variables such as the numbered down, the distance it takes to get a 1st down, the yard line at which the play begins, and the seconds left on the game clock. Each player, play, and game have an identification number called nflId, playId, and gameId respectively. To aid in the analysis, I also added variables that could be relevant, such as the Euclidean distance a defender is to the nearest defender, the Euclidean distance a defender is to the football, a binary variable called "target" which gives a 1 to a defender that was targeted on the play and 0 otherwise. To determine whether a defender was targeted, I chose the defender that was closest to the targeted receiver at the frame where the event annotation was "pass arrived".

Acquiring these additional variables required tremendous computing power. These distances were calculated for each player (on both offense and defense) in each frame of every play from every game. To make sure that these distances were calculated correctly, I randomly selected 30 plays to animate and had to make sure that each distance for each player was correct in all 30 plays. I could not remove frames as I needed the entire sequence of each play to determine whether a defender was targeted for a pass. The data did not provide a variable that indicated which receiver was targeted on the play so that also needed to be manually calculated. If a metric was found to be calculated incorrectly on a particular play, I would fix the issue and create a new sample of plays to check. It is important that each play had all the correct calculations for proper predictions and inference.

I removed all plays where the quarterback was sacked or purposefully threw an incomplete pass for a competitive advantage. Additionally, I only considered games which the Miami Dolphins played in and I only considered the first 13 games of their season. For model fitting purposes, such as the random effects model that will be introduced later, I only considered plays in which there were 7 possible defenders to be targeted. Since we are only interested in the probability that a defender is targeted, we only need to consider the frames in between the time the ball is snapped (the beginning of the play) and the frame that the quarterback decides to throw the ball. Thus, our final dataset for training consisted of 44,716 observations on 8 variables.

3 Modeling Approaches

In this paper, we will be focusing on 2 types of logistic regression models: a generalized linear mixed model (GLMM) and an XGBoost model. Let us first discuss the generalized linear mixed model.

3.1 Generalized Linear Mixed Model

Logistic regression has the same goal as that of any model-building technique used in statistics: to find the best fitting and most simple yet reasonable model to describe the relationship between a response variable and a set of independent variables. The most common example of modeling is linear regression. However, a key difference between logistic regression and linear regression is that the response variable for logistic regression is binary instead of continuous like in linear regression. For logistic regression, the specific form of the model is:

$$\pi(x) = \frac{e^{\beta_0+\beta_1(x)}}{1+e^{\beta_0+\beta_1(x)}} \tag{1}$$

However, to obtain the desirable properties of a linear regression model, we transform the equation into:

$$ln(\frac{\pi(x)}{1-\pi(x)}) = \beta_0 + \beta_1(x) \tag{2}$$

This is the logit function and it is linear in its parameters, can be continuous, and range from negative infinity to positive infinity depending on what x is. Thus,

we now have all the assumptions we had with linear regression:

1. All observations are independent from each other
2. The covariates are independent from each other
3. The errors are independent from each other
4. The variance of the errors is constant
5. The relationship between the response and the covariate is linear

Now as it applies to our data, we do have some problems with the first assumption. When we model target probability and incorporate factors such as distance from the receiver, the speed, the time of the game, and other variables, we have to understand that there is correlation amongst the observations from the same player. Not every player plays exactly the same way and some players might perform a lot better on average. We need to account for this variability so we introduce a random effect to the model that I will call the random player effect. This will account for the correlation amongst measurements from the same player so that we can have accurate standard error estimates for our coefficients to determine if variables are significant enough to be incorporated into our model. Additionally, this random effect can be an interesting variable to look at as we can see if certain players have a higher or lower target probabilities than the overall average. Since we have added a random effect to the logistic model, we now have a generalized linear mixed model. The same applies to observations within the same play and the same game.

3.2 XGBoost Model

Now that we have discussed the generalized linear mixed model, let us move onto the XGBoost model. Introduced by Chen et. al. (2016), XGBoost, short for Extreme Gradient Boosting, is a highly efficient and implementation of gradient boosting machines, which are a type of ensemble learning method. At its core, XGBoost combines the predictions of multiple individual models to produce a single, more accurate prediction. These individual models are decision trees. Decision trees divide the feature space, which includes all the possible combinations of feature values, into smaller regions. Each region corresponds to a set of conditions or rules based on the features.

Once the feature space is divided into regions, decision trees make predictions for new data points by assigning them to the region that matches their feature values. Each decision tree is constructed to optimize a specific objective function, which determines how well the tree fits the training data and how it contributes to the overall model's performance. XGBoost uses gradient descent to find the direction in which the objective function decreases the fastest and adds new trees to the model in that direction. The objective function of XGBoost includes two parts: training error and regularization:

$$X_{obj} = \sum_{i=1}^{n} I(y_i, \hat{y}_i + \sum_{k=1}^{K} \Omega(f_k) \tag{3}$$

where $\sum_{i=1}^{n} I(y_i, \hat{y_i})$ is used to measure the difference between the predicted value and the real value of the loss function and $\sum_{k=1}^{K} \Omega(f_k)$ is the regularization

term.

XGBoost adds news trees by continuously splitting features. XGBoost adds one new tree at a time instead of getting all the trees at once, and continuously repairs the previous test results by fitting a new loss function to the residuals of the last prediction. Below is a flow chart of XGBoost.

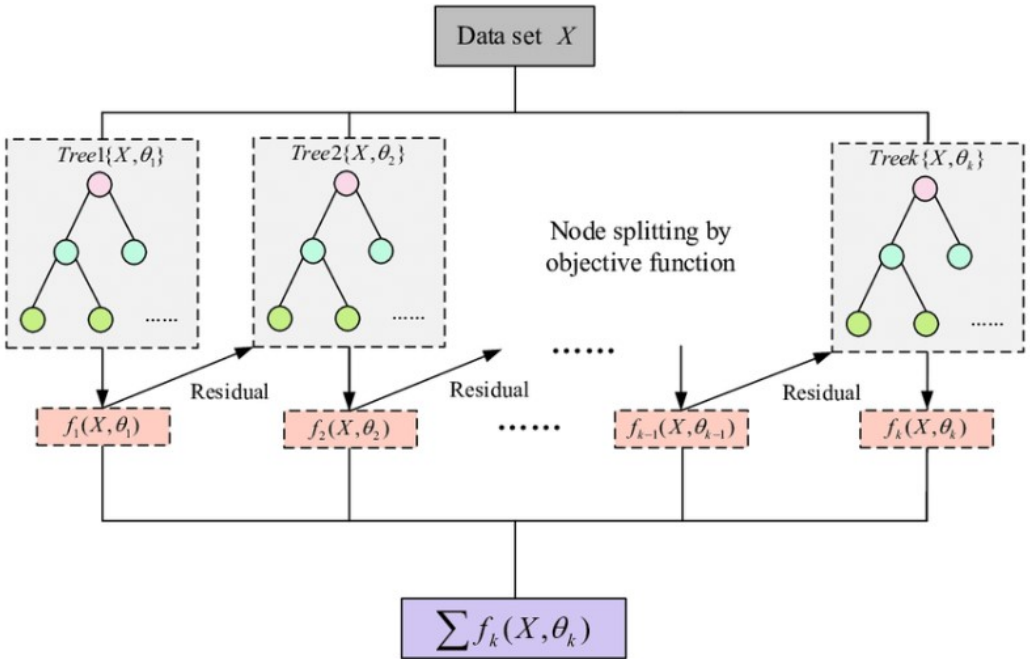

Figure 2: XGBoost flow chart. Chart provided by Guo et. al. (2020)

When we perform variable selection using XGBoost, we do not have P values like in the GLMM. Instead, we often look at three features: Gain Importance, Cover Importance, and Frequency Importance. The most meaningful measure of variable selection is often the Gain Importance which measures the reduction in the objective function, such as log loss, resulting from each feature split. Features that lead to larger improvements in model performance are considered more important. Cover Importance measures the average coverage of data points associated with each feature across all the trees in the model. Frequency Importance is based on the frequency of each feature appearing in the top splits of the trees. There is no steadfast rule on what is considered a high importance metric so we must look at these metrics across all the variables and determine ourselves which variables are worth keeping.

4 Model Fitting

Now that we have discussed both approaches, it is time to fit both models to the data. For both models, the training dataset included the first 9 games of the Miami Dolphins NFL season and the testing dataset included the 10th, 11th , 12th , and 13th game of the season. For the GLMM, I fit the variables “down”, “YardsToGo”, “yardlineNumber”, “gameClock”, “distance to nearest defender”, and “distance to football” as fixed effects. The random effects included the nflId and playId. After fitting this model, I fitted a reduced model that did not include any random effects. This was to test if any of the random effects are important to include in the model.

After performing the likelihood ratio test, we found that there was not enough evidence to suggest that the random effect of playId was significant to include in the model (P value was approximately 1). Another likelihood ratio test was performed to test the significance of the nflId random effect. For the nflId random effect, we found that there was enough evidence to suggest that we should include this in the model (P value < 0.001).

For the fixed effects, we found that there was sufficient evidence to claim that the variables "distance to nearest offender", "distance to football", "yardsToGo", and the intercept were significant predictors in modeling target probability. "down", "yardlineNumber", and "gameClock" did not have sufficient evidence to claim that they are significant predictors in modeling target probability. Thus, our GLMM selected the random effect of nflId and the fixed effects "distance to nearest offender", "distance to football", "yardsToGo". The results are provided in the table below.

Effect	Estimate	Std. Error	Z value	P value
quarter	0.003623	0.012846	0.282	0.7779
down	-0.008913	0.017948	-0.497	0.6195
yardsToGo	0.010221	0.004179	2.446	0.0144
yardlineNumber	0.001184	0.001087	1.089	0.2764
gameClock	-0.001598	0.003078	-0.519	0.6036
distToNearestOff	-0.056769	0.003898	-14.563	$<$0.001
distToFootball	-0.010336	0.002468	-4.188	$<$0.001

Table 2: Table of Fixed Effects results from fitting the full model of the GLMM

For the XGBoost model, as stated earlier, we do not have P values to perform model selection. Instead, we have the importance metrics. The following table shows these metrics when we fit the XGBoost model with the same variables as the GLMM.

Feature	Gain	Cover	Frequency
nflId	0.20563027	0.14104802	0.20453413
playId	0.19964835	0.18903952	0.17115097
gameClock	0.19125199	0.129414740	0.15794718
yardlineNumber	0.13029926	0.10069136	0.12232187
distToNearestOff	0.08947899	0.16059378	0.11684106
yardsToGo	0.06473348	0.09240174	0.06053812
distToFootball	0.06157063	0.16386998	0.11235675
down	0.05738701	0.02294091	0.05430992

Table 3: Table of importance features from the XGBoost model

Like the GLMM, we see that the nflId was an important feature in predicting target probability as it has the most gain in the loss function and frequency in the decision trees. The XGBoost model, unlike the GLMM, claims that the "playId" is one of the most important features of the model as well as "gameClock" and "yardlineNumber". To compare the model performance of both models, we chose

to include the same number of features for both models. Thus, our GLMM will include the variables, “nflId”, “yardsToGo”, “distance to nearest offender”, and “distance to football”. Our XGBoost model will include the variables “nflId”, “playId”, “gameClock”, and “yardlineNumber”. Contingency tables on the training and test data for both models are provided below:

		Predicted	
		0	**1**
Actual	**0**	33201	5127
	1	5127	1261

Table 4: Contingency Table of Actual vs Predicted of GLMM on Training

		Predicted	
		0	**1**
Actual	**0**	38321	7
	1	7	6381

Table 5: Contingency Table of Actual vs Predicted of XGBoost on Training

		Predicted	
		0	**1**
Actual	**0**	21945	3633
	1	3633	630

Table 6: Contingency Table of Actual vs Predicted of GLMM on Testing

		Predicted	
		0	**1**
Actual	**0**	21991	3587
	1	3587	676

Table 7: Contingency Table of Actual vs Predicted of XGBoost on Testing

As you can see, the XGBoost model performs much better on the training data set than the GLMM, but both models perform similarly on the testing dataset. The variables I selected for the XGBoost model may be too specific on the training dataset, so I decided to run another XGBoost model using the variables selected by the GLMM. The results are provided in the two tables below:

		Predicted	
		0	**1**
Actual	**0**	38208	120
	1	120	6268

Table 8: Contingency Table of Actual vs Predicted of XGBoost remodeled on Training

		Predicted	
		0	**1**
Actual	**0**	21951	3627
	1	3627	636

Table 9: Contingency Table of Actual vs Predicted of XGBoost remodeled on Testing

From these tables we can see that the XGBoost behaves similarly to the GLMM which makes sense as they use the same variables.

5 Discussion

From section 4, we see that the XGBoost is very susceptible to overfitting. Only 120 observations were incorrectly predicted in the training data set, but when it came to the testing data, the XGBoost model performed only slightly better than the GLMM. The XGBoost is also very sensitive to the hyperparameters such as the number of boosting iterations and the max depth of the trees. Including too large of a number of either of these hyperparameters will cause the XGBoost model to overfit the data even more than it already has. For this dataset, the optimal number of boosting iterations was 50 with a max depth of 3. The GLMM tends to be more robust to overfitting especially when one does not include many predictors in the model. Since the XGBoost model only performed slightly better than the GLMM, we do not lose much predictive power choosing the GLMM and we will gain a lot more inference using the GLMM.

To evaluate defenders, I propose to use the GLMM as it incorporated the random effect in nflId. This means that we can check to see if certain players inherently have a different target probability than the team average. The following table below displays the random effect of each player.

In this table, I have included the intercept, the probability of being targeted on any given play, the players' name, the position, and the percentage of plays that particular player has played. The players are ordered based on the percentage of plays participated so that we can pay particular attention to the players we have the most observations on.

Name	Intercept	Probability	Snap Pct	Position
Kiko Alonso	0.38810894	0.5958274	92.19	Linebacker
T.J. Mcdonald	0.02451816	0.5061292	87.42	Safety/Strong Safety
Minkah Fitzpatrick	0.10852910	0.5271057	86.69	Safety/Strong Safety
Raekwon McMillan	-0.02183951	0.4945403	76.31	Linebacker
Reshad Jones	-0.13950643	0.4651798	75.76	Safety
Bobby McCain	0.44478227	0.6093980	75.57	Cornerback
Xavien Howard	-0.02152712	0.4946184	73.74	Cornerback
Jerome Baker	0.20896861	0.5520529	62.26	Linebacker
Torry McTyer	0.17000181	0.5423984	31.77	Cornerback

Table 10: Table of Random Effect on Target Probability for Each Player

One thing to pay attention to are the two cornerbacks Bobby McCain and Xavien Howard. Their target probability is different by about 10 percent. One may be able to infer that quarterbacks may target Bobby McCain more than Xavien Howard because Bobby McCain may not play as good of coverage than Xavien Howard and so Bobby McCain may inherently be chosen before the play even starts.

6 Conclusion and Future Works

As machine learning approaches become more common to use than classical statistics methods, we should not just assume that these machine learning approaches are inherently better than classical statistical approaches because they can work with high dimensional and complex data structures. As seen in this paper, we compared the XGBoost to a classical logistic regression model with fixed and random effects. We saw that the classical model performed slightly worse

than the XGBoost in predictive power, but we gain a lot of inference on evaluating the target probability of defenders.

One drawback of this attempt to model target probability is that we did not subset the data on man and zone coverage. Man coverage is when a defender is assigned a particular offender to cover. Zone coverage is when a defender is assigned an area of the field to cover. Thus, when we evaluate the random effect, we have to consider the players performance in man and zone coverage. We would have to run an additional model to determine which plays are in man or zone. This is where I believe that the XGBoost model could be useful as it is often cited to be good with classification.

Some future works to this target probability model is to also combine this with a catch probability model to see if some defernders are under or over valued. If a defender gets targeted a lot but has a low catch probability, then this a highly valued player. If a defender does not get targeted a lot, but has a high catch probability, then offenses can take advantage of this.

7 References

Burke, B. (2019). Deepqb: Deep learning with player tracking to quantify quarterback decision-making & performance. MIT Sloan Sports Analytics Conference. Retrieved from http://www.sloansportsconference.com/wp-content/uploads/2019/02/DeepQB.pdf

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (pp. 785–794). ACM. http://doi.acm.org/10.1145/2939672.2939785

Guo, R., Zhao, Z., Wang, T., Liu, G., Zhao, J., & Gao, D. (2020). *Degradation state recognition of piston pump based on ICEEMDAN and XGBoost. Journal of Applied Sciences, 10*(18), 6593. https://doi.org/10.3390/app10186593

Hosmer, D. W., Jr., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (3rd ed.). Wiley.

Yurko, R., Matano, F., Richardson, L. F., Granered, N., Pospisil, T., Pelechrinis, K., & Ventura, S. L. (2020). Going deep: models for continuous-time within-play valuation of game outcomes in American football with tracking data. *Journal of Quantitative Analysis in Sports, 16*(2), 163-182. https://doi.org/10.1515/jqas-2019-0056