NFL Big Data Bowl Sub-Contest

Anonymous

12/17/2019

1 Introduction

Modern machine learning algorithms exhibit state-of-the-art predictive performance across an array of applications ranging from health care[1] and bioinformatics[2] to computer vision[3] and natural language processing[4]. One important reason for their success is the ability to learn complex, highly nonlinear functions from an exceedingly large range of function classes. In other words, the analyst need not specify details of the model but the resultant model will still learn a highly accurate predictive function seemingly automatically from the data. The resultant models, however, are difficult for humans to interpret and thus earned the epithet 'black-box'.

Despite the drawbacks in human interpretability, these methods are a go-to for many analysts regardless of the task at hand. Deep convolutional neural networks (CNNs) are one of the most widely-utilized methods particularly due to the model *learning* distributed feature representations *automatically* from the data and combining these features into the high-performing nonlinear function described above[5]. It is often unclear what these learned features correspond to, or what concepts they represent, since the model is composed of millions of weights combined nonlinearly. This has in turn led to a bevy of techniques being developed to 'peek inside the black box' and thus understand what the model has learned[6].

One approach is to visualize which parts of an input image *excite* (read as: *attain high values*) different parts of the trained model. In the task of discriminating cats versus dogs, a collection of convolutional filters may be interpreted as learning, for instance, a cat ear is pointier than a dog ear. It is by no means trivial to make these interpretations, however the *predictive gains* and *automatic feature extraction* from using such a CNN model, as opposed to a simpler but more interpretable linear model which encodes cat ear pointedness directly, often trumps the *sacrifices in interpretability.* This is referred to as the interpretability-performance trade off and Kaggle competitions are excellent examples.

But abandonment of interpretability is by no means universally the winner. For instance, doctors may consult a 'black-box' model when determining a patient's best treatment option but ultimately rely upon their expertise to determine best course of action. For the doctor to be successful and

[1]Yu, K., Beam, A.L. & Kohane, I.S. Artificial intelligence in healthcare. Nat Biomed Eng 2, 719–731 (2018) doi:10.1038/s41551-018-0305-z

[2]Zou, J., Huss, M., Abid, A. et al. A primer on deep learning in genomics. Nat Genet 51, 12–18 (2019) doi:10.1038/s41588-018-0295-5

[3]Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," Computational Intelligence and Neuroscience, vol. 2018, Article ID 7068349, 13 pages, 2018. https://doi.org/10.1155/2018/7068349

[4]Young, Tom, et al. "Recent trends in deep learning based natural language processing." ieee Computational intelligenCe magazine 13.3 (2018): 55-75

[5]LeCun Y., Haffner P., Bottou L., Bengio Y. (1999) Object Recognition with Gradient-Based Learning. In: Shape, Contour and Grouping in Computer Vision. Lecture Notes in Computer Science, vol 1681. Springer, Berlin, Heidelberg

[6]For example see Ancona, Marco, et al. "Towards better understanding of gradient-based attribution methods for deep neural networks." arXiv preprint arXiv:1711.06104 (2017)

avoid malpractice suits they must have a strong mental representation of the patient's current state. In a computational framework, this means they understand which features are important for the predictive task at hand. The doctor's ability to do this effectively stems from their experience and the process of doing so may be referred to as *meaningful feature engineering*; that is to say *the doctor* has learned relevant indicators of patient success as opposed to relying upon *a 'black-box' model* to do so for her.

Combining domain knowledge through meaningful feature engineering with the predictive performance of machine learning models is often the formula for a winning gameplan[7]. The task of meaningful feature engineering for predicting run success will be the primary focus of my report. For a feature to be meaningful it must capture global associations with yards gained and not be player-specific. In other words, I will only consider features which may generalize to all ball carriers as opposed to individual player-specific features such as, for example, a *Dalvin Cook effect.*

At a high level I follow a basic supervised learning analytic framework. Specifically, I begin with data importing and a brief description of basic formatting, namely getting the raw Next-Gen Stats positional data into a format I can use. I then describe the task of feature engineering and how these are inspired by my extensive Sunday afternoon armchair quarterbacking. I next transition into model fitting and evaluation with a brief focus on model interpretability. I conclude with a discussion on limitations and future directions.

A markdown and code for full reproducibility is shareable upon request via a private GitHub repo. This will be made public after the competition deadline.

2 Setup

I begin by loading the training data and sourcing a functions file. The functions file is available in my GitHub repo and contains functions to derive the features and assess the model fit.

I next clean the data by standardizing player direction, team names, weather, and scoring. The standardization of the tracking data follows the framework provided on the NFL Big Data message boards; namely the offense is always moving from the left-to-right and the player orientation is zeroed to offensive-player-direction left. Future player position is computed using current speed and acceleration with a time duration of $\frac{1}{2}$ seconds.

3 Feature engineering

Next we need to derive features on all available data. The table below contains descriptions for the non-obvious features. It is worth noting that all features are lagged in the sense that they do not use current play yardage information.

For example, the variable `CurrentGameSuccessRate` encodes how many successful rushes (Yards gained > one-half yards needed) the possession team has already run in the current game as a fraction of how many total rushes they ran. This is computed using all plays leading up to the

[7]For example Islam, Sheikh Rabiul, et al. "Infusing domain knowledge in AI-based" black box" models for better explainability with application in bankruptcy prediction." arXiv preprint arXiv:1905.11474 (2019)

current play within a given game but does not consider the current play. Failure to properly lag features would cause leakage from the outcome variable.

Feature	Description
Runner_type	Ball carrier position (RB, QB, WR)
DefendersBlocked	Estimated number of defenders engaged in block at handoff
effectiveDownfieldSpeed	Runner speed directed downfield
effectiveDownfieldAcceleration	Runner acceleration directed downfield
effectiveCrossfieldAcceleration	Runner acceleration directed laterally to the line of scrimmage (LOS)
xDistanceTravelled	Runner distance travelled downfield since handoff
MinimumTacklerYardsAway	Nearest unblocked defender yards downfield from runner
MinimumTacklerDistanceAway	Nearest unblocked defender distance from runner
MinimumTacklerEffectiveDownfieldSpeed	Nearest unblocked defender speed attacking LOS
MinimumTacklerEffectiveDownfieldAcceleration	Nearest unblocked defender acceleration attacking LOS
MinimumTacklerEffectiveCrossfieldAcceleration	Nearest unblocked defender acceleration directed laterally to LOS
MinimumTacklerXDistanceTravelled	Nearest unblocked defender distance travelled towards LOS since handoff
UnblockedDefendersYardsAway	All unblocked defenders mean yards downfield from runner
UnblockedDefendersDistanceAway	All unblocked defenders mean distance from runner
DefensiveSD__Yards	Standard deviation measuring spread of defenders in yards downfield
DefensiveSD__Dis	Standard deviation measuring spread of defenders in distance from runner
S	Runner speed
A	Runner acceleration
Dis	Runner distance
Run_type	Is the run a sweep, off tackle, or dive? Is it field-side or to the boundary?
cumwinpct	Cumulative team winning percentage prior to the current game
YardsToFirstDown	Yards to go to a first down
OffensiveScoreDifference	Possession team points margin prior to play
OffensiveScoreBeforePlay	Possession team points total prior to play
MeanRushesPerGame	Average number of rushes per game prior to the current game, per team
MeanRunSuccess	Proportion of rushes per team attaining at least one-half the requisite yards
MeanYardsPerAttempt	Average yards per attempt, per team
MeanYardsPerGame	Average total rush yards per team per game
Within4Mins	Gameclock is within 4 minutes of a half
LastPlayYards	Last rushing attempt yards gained
CurrentGameYards	Running total of current game rush yards
CurrentGameAttempts	Running total of current game rush attempts
CurrentGameSuccessRate	Running proportion of number of successful rushes to total number of rushes
ProgressIntoRunScheme	Number of current rushes in game relative to the team average rushes
RelativeRunSuccess	Current game yards per attempt divided by the team average YPA
OlinemenPushDownField	Current play offensive linemen mean push downfield since handoff
OffensiveSD__X	Current play O-line spread (standard deviation) in the downfield direction
OffensiveSD__Y	Current play O-line spread (standard deviation) in the lateral direction
HomeTeamAdvantage	Possession team is home team

3.1 Viewing engineered features

3.1.1 Sample plays

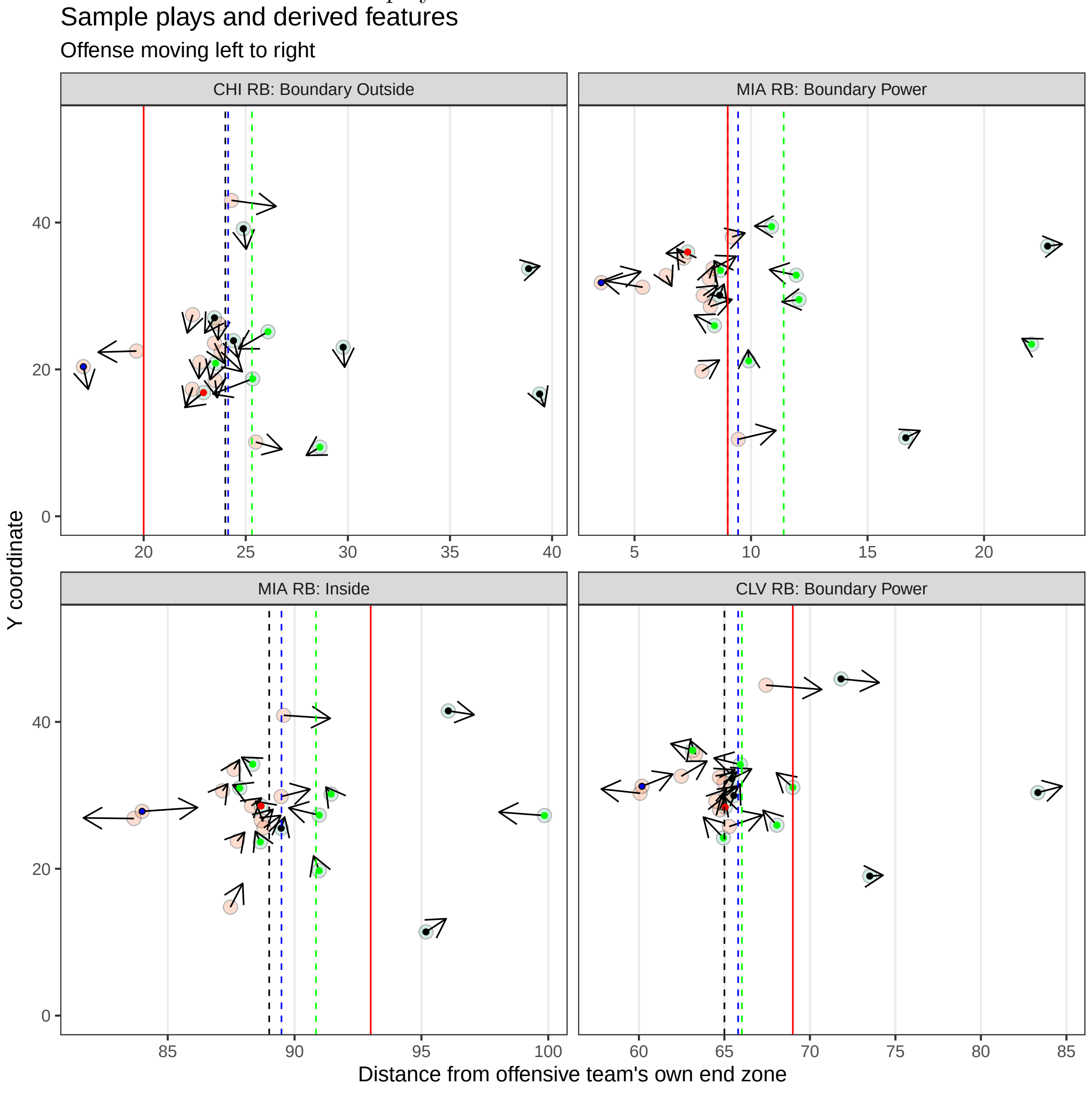
Next let's randomly sample some plays and check the engineered features. The offense is colored red and is moving left to right. The defense is green and arrows indicate direction. The blue dot signifies ball carrier, the dashed black lin indicates line of scrimmage (LOS) and the red line denotes the yards gained.

Several engineered features are shown for four sample plays. For instance, defenders who are able to make a play on the ball carrier (ie unblocked) are denoted with a green dot while defenders who cannot impact the play, according to my method, are denoted with black dots. The mean of the unblocked defenders is shown with the dashed green line and the defender who is most likely to make the play is denoted with the red dot. Lastly, the offensive linemen drive downfield is denoted with the blue dashed line. The run type (eg boundary power, field inside) estimated from ball carrier intended direction is provided in the facet header along with the team and ball carrier position.

There are several interesting nuggets to be seen in this figure and collectively they support the claim that the derived features are relevant and highly interpretable. For instance, in the upper left panel

(CHI RB: Boundary Outside) the nearest unblocked defender (red dot) appears to end up making the tackle for a loss by maintaining outside leverage against the right tackle. If a casual fan were to look at this play without any derived features overlaid my hunch is that they would pick out the player with the red dot as the most likely tackler. It is satisfying to see that the features agree with intuition.

In the upper right figure (MIA RB: Boundary Power) it appears that the nearest blocker (red dot) gets kick blocked by the left tackle and the ball carrier is able to squeeze through the hole but is met immediately by the unblocked defenders (green dots on linebackers and interior DL). A similar story may be the case for the bottom left panel although here the free safety (furthest right green dot signifying unblocked and ability to make a play on the ball carrier) satisfies his inside-out run support and attacks the ball carrier for a ~8 yards gain. In the final bottom right play we see all of the DBs are out of the play but the linebackers flow to the ball and make the tackle.

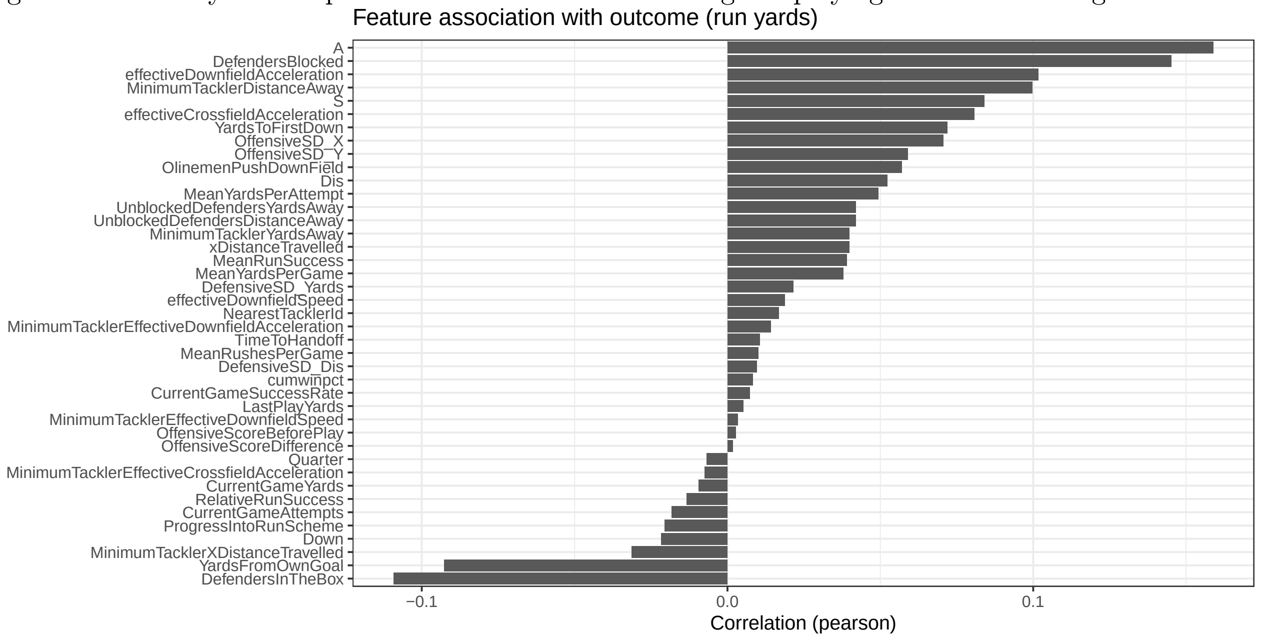


3.1.2 Feature associations with outcome yards.

The figure below shows the Pearson correlation between the derived features and the outcome variable yards gained. Moderate correlation is observed for many of the features with the sign and relative magnitude agreeing with intuition. For instance, values of ball carrier effective downfield acceleration (`effectiveDownfieldAcceleration`) are positively associated with yards gained. This of course makes sense as runners who are speeding up through a hole after the handoff will have a larger force and momentum and therefore more yards gained on average.

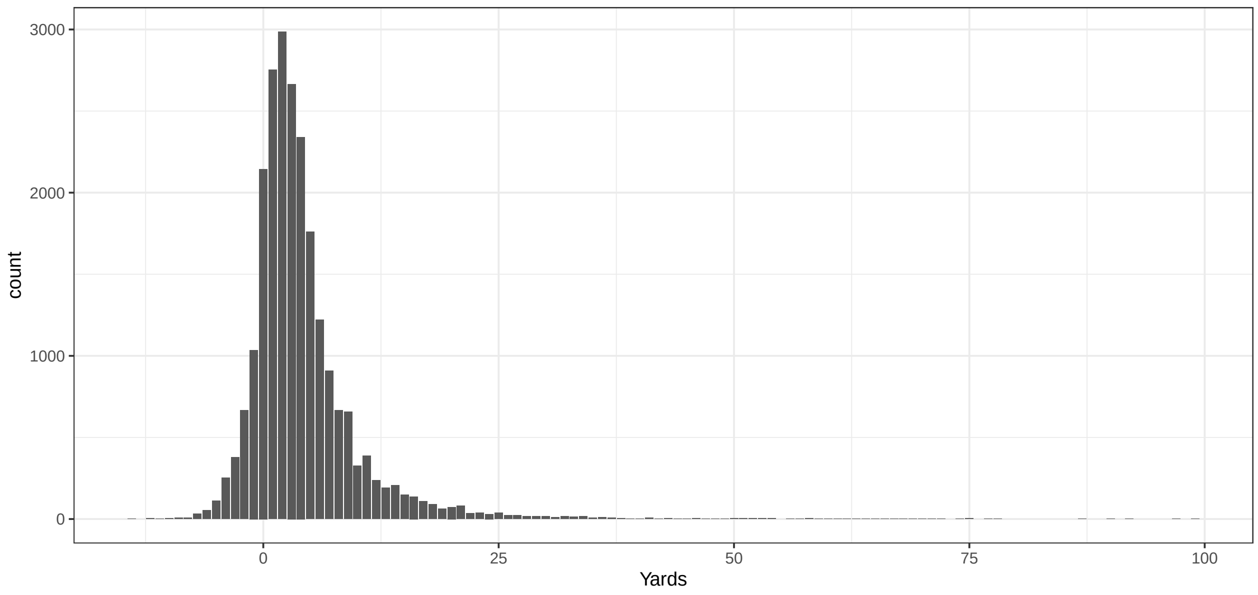
Similarly, a strongly negatively-correlated feature (`MinimumTacklerXDistanceTravelled`) is the nearest potential tackler's distance travelled upfield towards the runner. In this case, for increasing values of the tackler closing in (strictly in the X-axis direction, ie upfield) relative to the ball carrier at time of handoff, on average the ball carrier will gain fewer yards. This feature captures linebackers, for example, attacking and plugging holes in the O-line.

Interestingly, the last running play yards gained is hardly correlated with the next play yards gained. This may be interpreted as momentum in the run game playing little role in run game success.



Now let's plot the outcome variables yards gained. The distribution is heavily skewed right meaning there's a long tail on the distribution to the larger values of yards gained. To address this deviation from normality I'll be transforming the outcome variable via the log transform.

Distribution of outcome yards

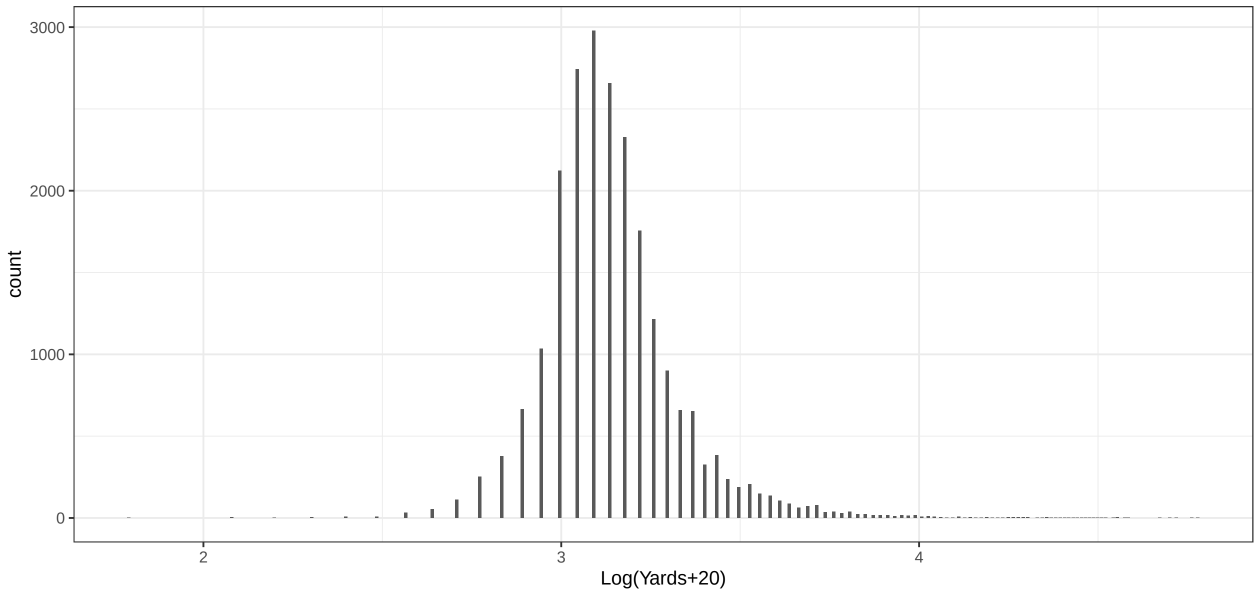


3.2 Build model matrix

Now I'll define the outcome and build the model matrix. I considered both binning the realized yards and also leaving the variable un-transformed. Ultimately the high outlier points (seen in the histogram above) unduly influence the objective function when left un-transformed. Additionally it's not clear where to bin the yards gained, and thus I determined to simply use the log transformed values after shifting the values into the positive domain (I added 20 yards to each value).

We see in the histogram below the effect of logging the values - namely the distribution appears normal (besides the discretization due to rounding of yards to the nearest integer). No longer does the large right tail exist and those outlier points will no longer disproportionately influence the loss.

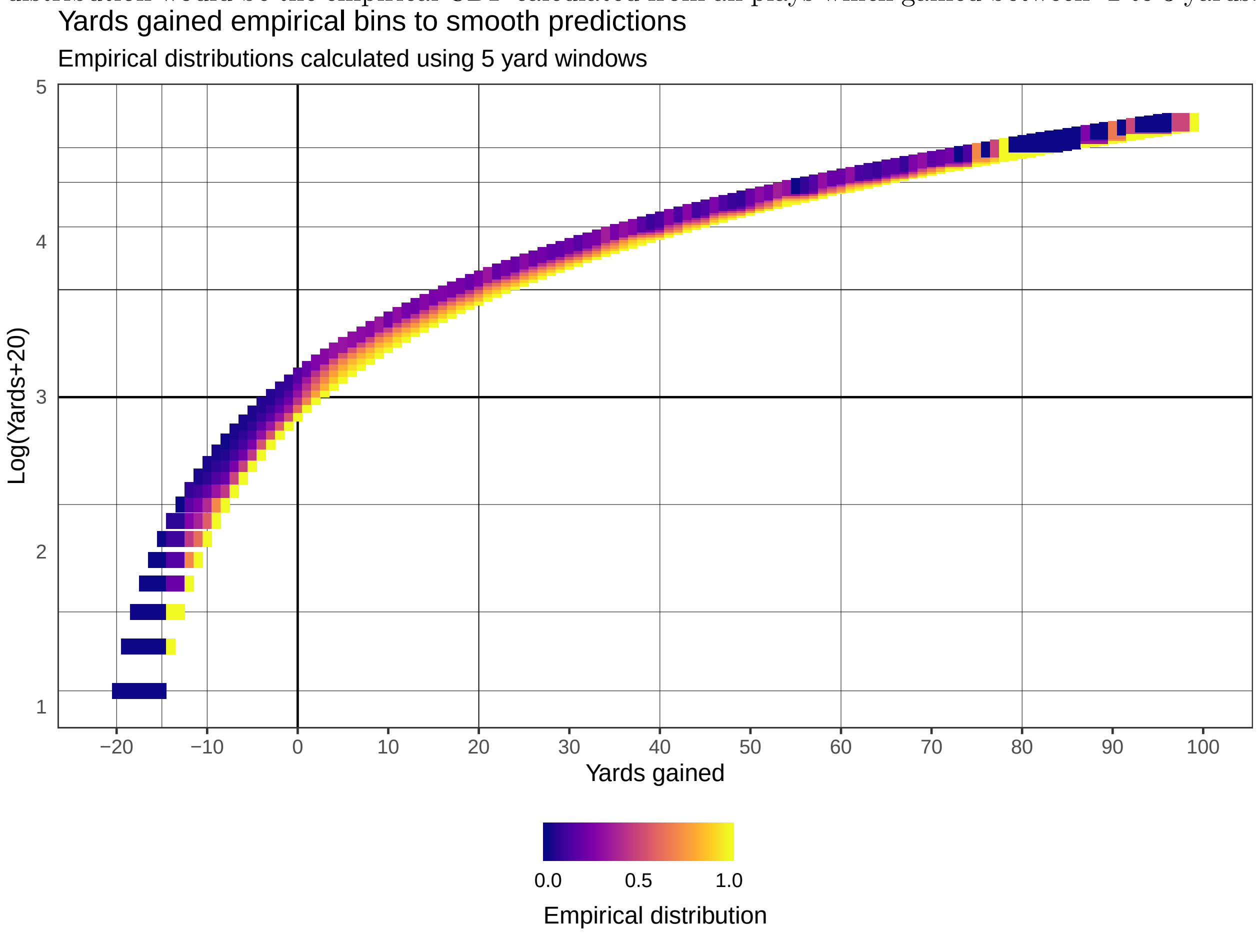
Distribution of transformed outcome yards



3.2.1 Empirical distribution for smoothing

Next I calculate the empirical distributions of yards within sliding windows around each value for smoothing the predictions. Specifically I use a five yard window around each yardage, and within each window I calculate the empirical distribution of yards gained. This procedure is a simple way to incorporate uncertainty into my model.

A plot of the empirical bins below shows the yards gained along the x-axis and the transformed yards along the y-axis. Coloring is by the values calculated empirically above. Within each window (horizontal bar for a fixed Y value) the values sum to 1 as they are cumulative distributions. For instance, if the predicted point estimate for the transformed value for a given test set play is 3 (thickest black horizontal line corresponding to a realized yards of 0), then the prediction bin encompasses yards $\in [-2, 3]$ and the reported predicted cumulative distribution would be the empirical CDF calculated from all plays which gained between -2 to 3 yards.



4 Train model

I'll use gradient boosting machines via the xgboost method[8] to obtain point estimates for each test set observation. Note that the test set is constructed of all 2018 week 17 games. It is critical to

[8] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016.

set up the test partition split to include entire games, and not allow plays from the same game to exist in both the training and the testing sets. This causes leakage and allows the analyst to overfit their model to the training set while still achieving excellent test set performance. In other words it is trivially easy for a machine learning model to memorize the training data but learning a generalizable function is much more difficult.

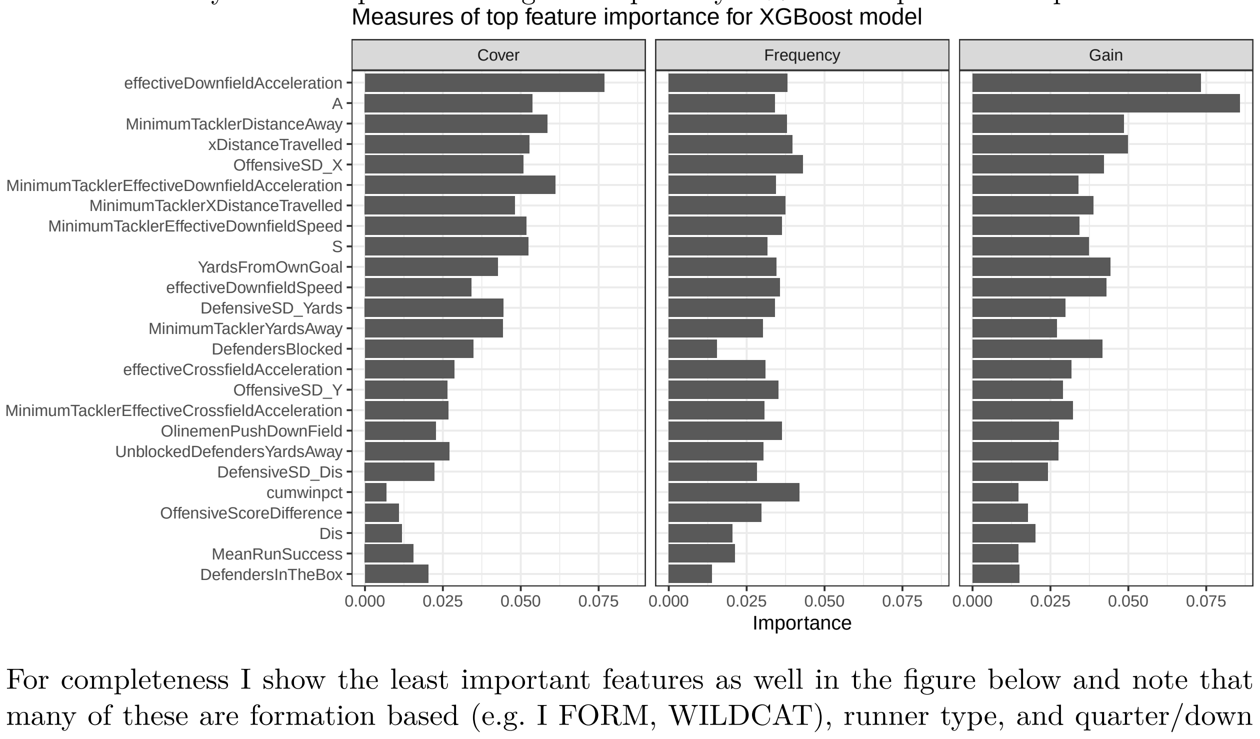
The model boosts 100 regression trees of depth 8 for 1000 iterations with a learning rate of 0.0075, L_2 regularization of 0.05 and L_1 regulatization of 0.02.

5 Model diagnostics

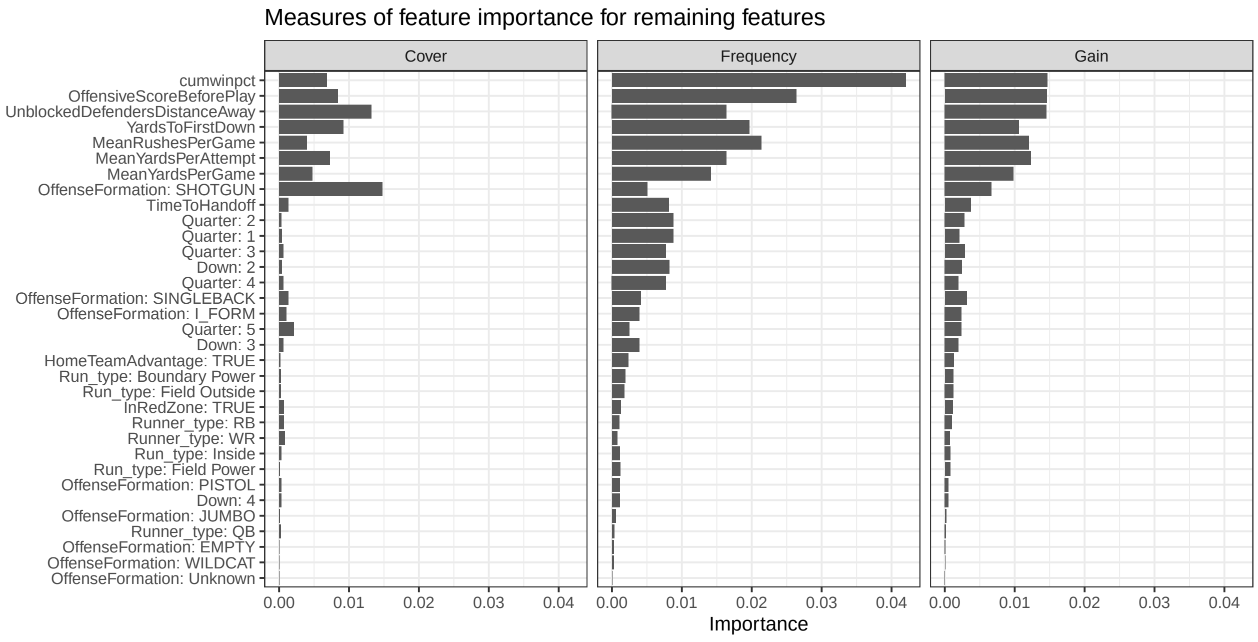
5.1 Variable importance plot.

Now I'll plot the feature importance from the model fit. Three measures of feature importance are included and while the definitions of each will not be covered in this analysis, it suffices to say that the ordering is what we should focus on and the larger values signify more important features.

Sorting along the Y-axis indicates that the spatial features are found to be the most important features for the model predictions. Specifically the ball carrier acceleration and nearest defender who is capable of making the tackle make up the majority of the top features. The offensive linemen are also important, as evident by the large value for the `OffensiveSD_X`, `OffensiveSD_Y`, and `OlinemenPushDownField` features, which capture the spread of the offensive linemen in the X and Y directions and the mean push downfield, respectively. Collectively the engineered features based on my armchair quarterbacking make up nearly 80% of the top 25 most important features.



For completeness I show the least important features as well in the figure below and note that many of these are formation based (e.g. I FORM, WILDCAT), runner type, and quarter/down information. Note the X-axis scaling is different from that above to highlight the smaller differences.

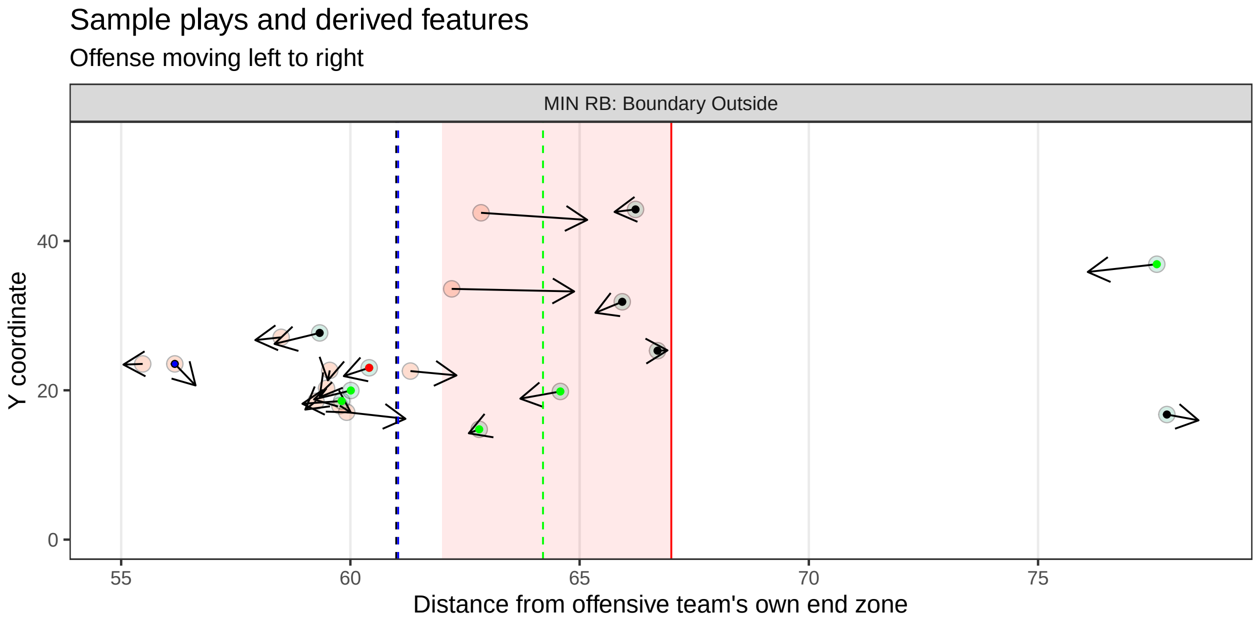


5.2 Training set performance

5.2.1 Sample plays

Given the trained model we now need to assess the model fit. During the training procedure I used the 2018 week 17 as a validation split as not to contaminate my training set. Here we'll use all the plays from a randomly chosen set of 3 games sampled from the training set.

The predicted CDF (orange shading) for a single play is provided below. The dashed lines represent the same features described in the earlier section.



5.2.2 Error

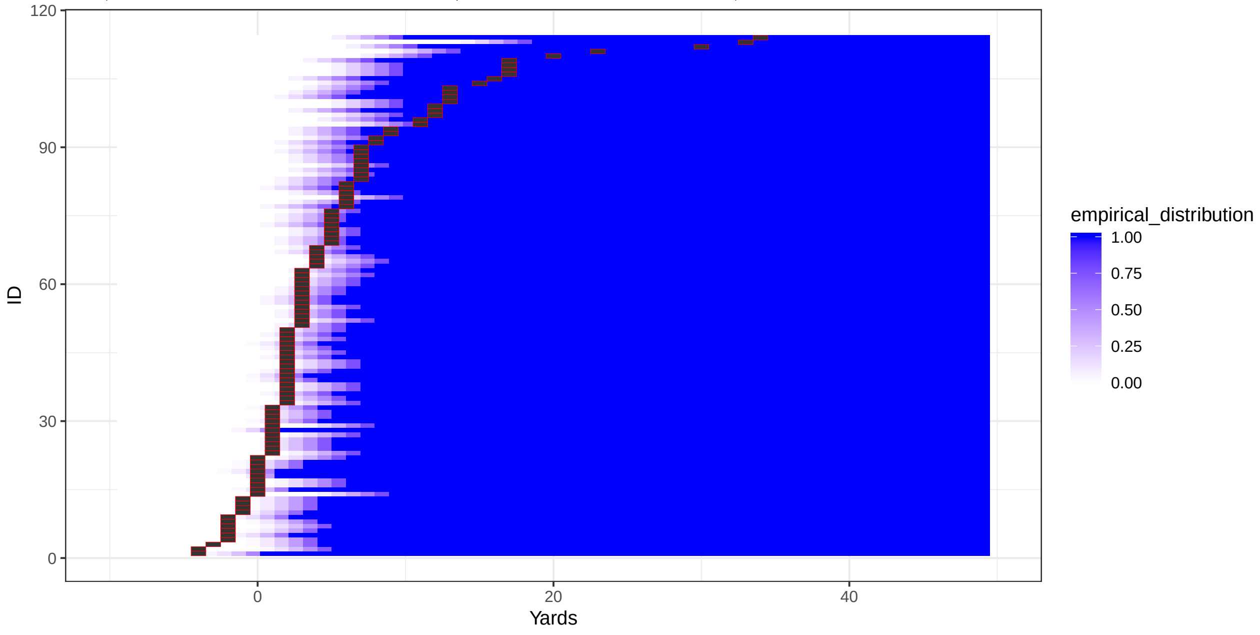
The cumulative distribution function (CDF) error for the three training set games sampled above is reported below.

```
## [1] "Training set CDF error: 0.0146120547611214"
```

5.2.3 CDF error

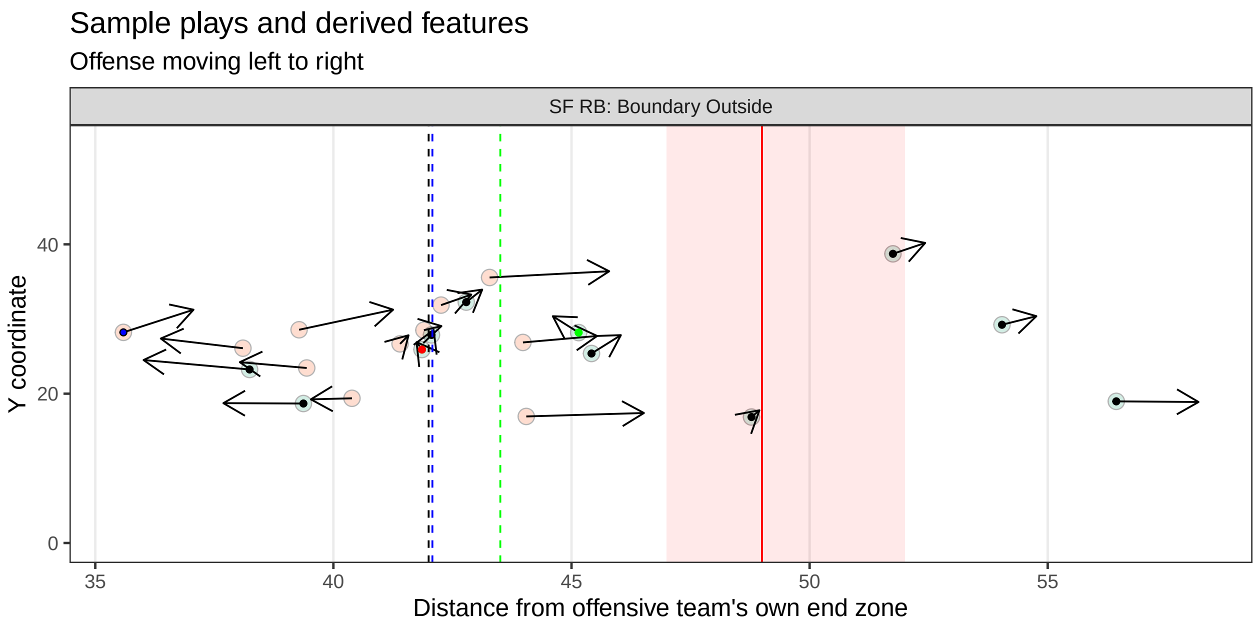
A plot of the CDF error for all plays from the sampled games is provided below. I show the yards along the x axis and sort the observations along the y axis by their realized value (red boxes). Each row is a single play and the blue coloring signifies the empirical distribution of yards gained within the 5 yard bin constructed around the predicted point estimate.

Qualitatively it appears that the model has learned a relevant predictive function (the blue somewhat tracks the red) although it is difficult to accurately predict yards gained for runs which attained more than 10 yards. While there is doubtlessly room for improvement, this model failure is not necessarily shocking when one considers how these runs usually come about: Based on my experience watching football large runs are often the result of a playmaker making a play. This may be in the form of a broken or missed tackle, a nasty juke, or even a stiff arm into the shadow realm (hello Derrius Guice). Regardless the cause, this is to say that features calculated *at the time of handoff* will not be predictive of whether a ball carrier makes a play against the would-be tackler. Further, as described in the introduction, the purpose of this analysis is to engineer features which capture global trends in running plays; a better model could surely be fit by incorporating player-specific effects (eg a feature for each team/runner in the model) but that is not the focus of this work.



5.3 Test set performance

Next I'll repeat the procedure from above for the test set plays. A plot of the predicted CDF for a single test example is shown below with the predicted CDF highlighted in orange. In this example our prediction is spot on (red line lies within the orange shading).



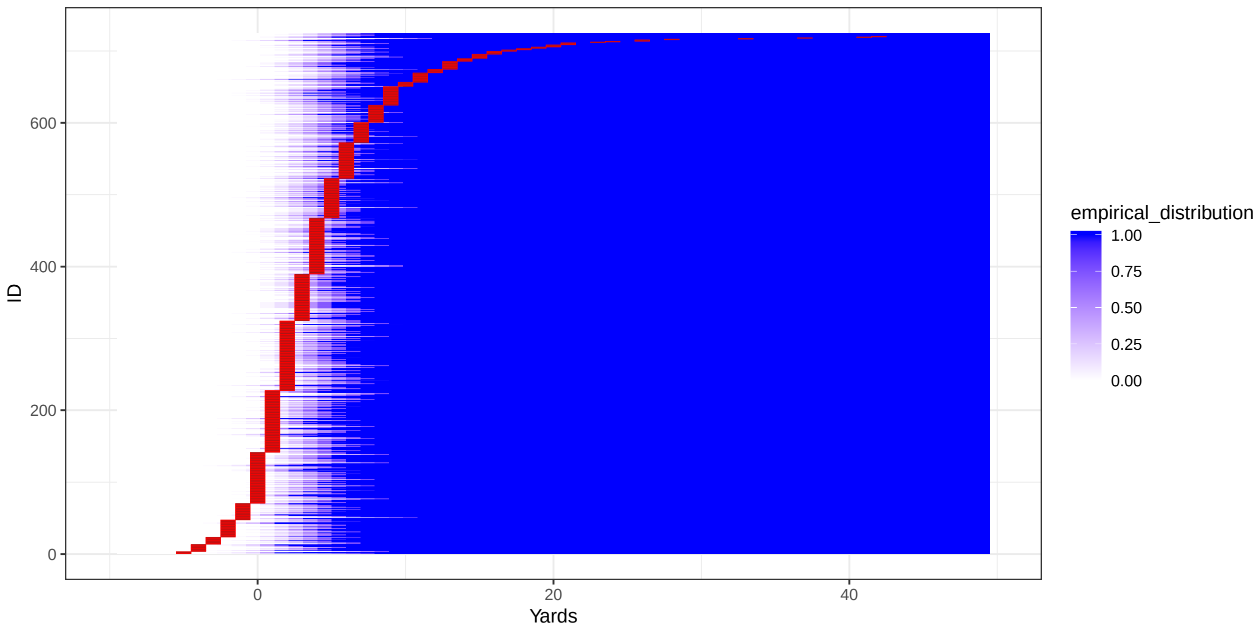
5.3.1 Error

The cumulative distribution function (CDF) error for the entire testing set (week 17 of the 2018 season) is reported below. The model fit on the test set performs worse than on the training set indicating some degree of overfitting. While this could be remedied through more time spent on parameter tuning, I reiterate that the purpose of this report is not attaining the best predictive performance but instead is an investigation into engineering relevant and interpretable features for predicting run yards. The advantage of tackling the problem in this latter manner is that football fans understand these features and can use their intuition to further refine them as opposed to using a complex black-box model which may automatically extract features, such as a CNN, without the ability to incorporate domain knowledge.

```
## [1] "Testing set CDF error: 0.0152816863201441"
```

5.3.2 CDF

The by-play error is provided below for the test set. Again the x axis represents yards, the red boxes are the realized values, and the blue shading is the predicted CDF. The poorer test set performance is evident, especially for plays gaining a larger amount of yards.



6 Discussion

This report investigates relevant feature engineering in the space of predicting running play yards gained given NFL Next Gen Stats at time of handoff. In particular I focus on crafting features which generalize across all ball carriers (global features) as opposed to player-specific features, such as a *Lamar Jackson effect*. Inclusion of these latter features may make the trained model more prone to overfitting whereas the former feature types hold potential to generalize across all levels of football, teams, and styles of play. Indeed, it is through these generalizable features from which we as domain experts may interpret relevant effects and leverage model findings to "take it to the house" on offense or "bottle up the run" on defense.

Despite predictive performance lagging behind the top performing models submitted to the first-round competition, these hand-crafted features incorporate domain knowledge and tend to agree with football intuition. Undoubtedly improvements may be made to the model which could improve predictive performance however this often comes at the cost of model interpretability, thus providing a rather clear example of the interpretability-performance trade off.