Predicting Yards Gained on NFL Handoff

Travis McGuire, Bradley Barnhart, Dahlia Radif

travis.mcguire@stanford.edu, bbarnhar@stanford.edu, dradif@stanford.edu

Stanford University

Motivation and Objective

The NFL is a huge business, with an estimated $8.1B in yearly revenue. An important problem, tackled by coaches, analysts, and fans, is how many yards will be gained on a handoff. We hope to answer that question using machine learning. The objective is to minimise the cost

$$C = \frac{1}{199n}\sum_{i=1}^{n}\sum_{j=-99}^{99}(P(Y \leq j) - H(x - Y_i))^2$$

where P is the predicted CDF, H is the Heaviside function (H(x) = 1 for x ≥ 0 and 0 otherwise), and Y is actual yards.

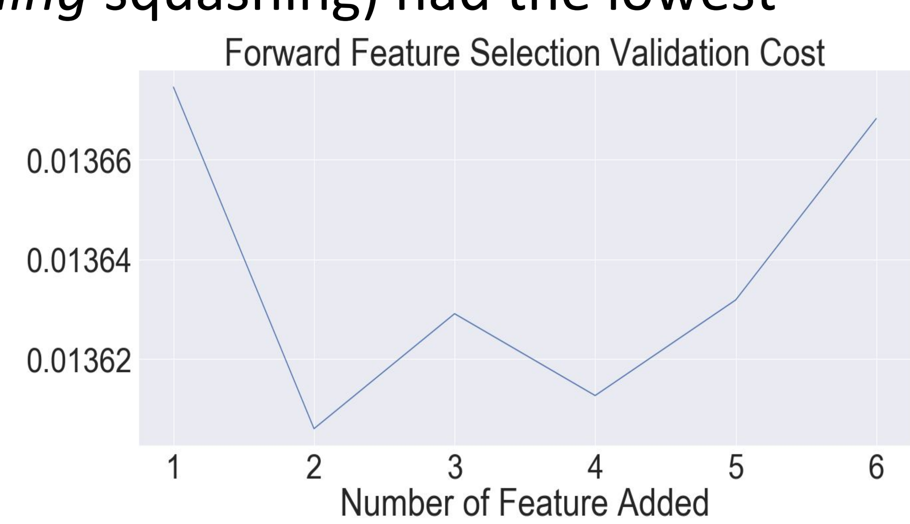
Data and Features

Data

- Next Gen Stats data obtained from Kaggle, contains 23k plays (samples) from the 2017/2018 NFL season
- Originally 500k points with 22 for each play, but downsampled to keep only the rusher data point
- Each sample contains 49 features, including actual yards gained (ground truth) for each play
- Cleaned data for missing, rare, & inconsistent values

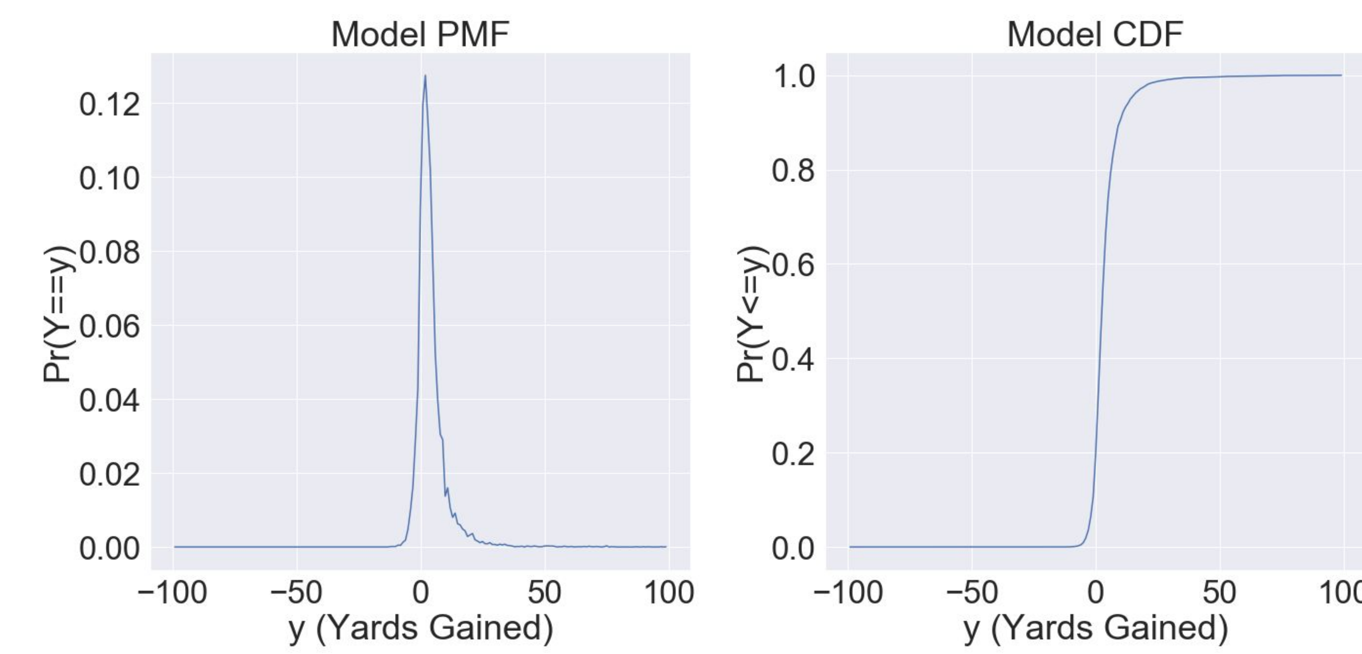
Features

- Several new features were engineered such as *YardsAvgOffense*, *YardsAvgDefense*, *YardsRemaining*, *Carries*, *RusherAvgYards*, & *YardsRemaining*
- It is believed only a few features are good predictors For that reason, **forward feature selection** is used
- Two features, *Acceleration* and *DefendersInTheBox*, (with *YardsRemaining* squashing) had the lowest validation loss
- These make sense, as they are how fast the runner goes and how many people are in the way

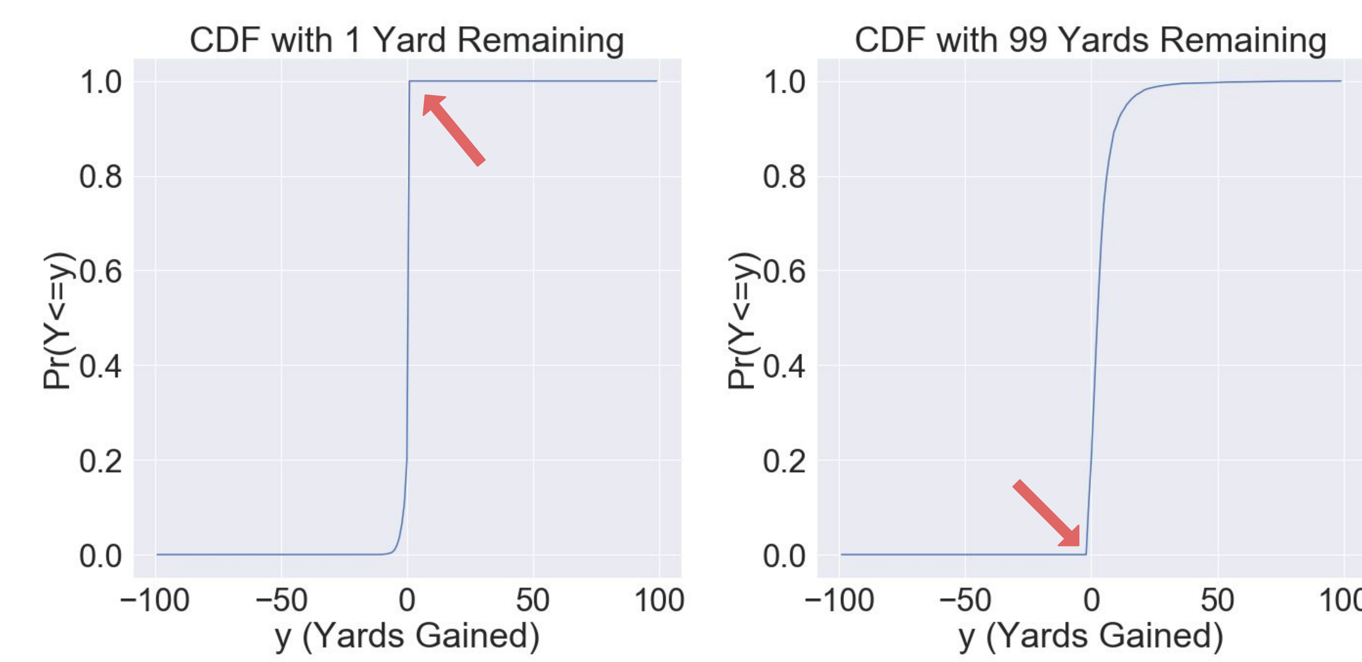


Models

- Each model tries to predict a multiclass PMF. For example, MLE gives:



- Squashing is a modification to all models that takes into consideration the most yards gained possible is a touchdown and least is a safety



- **Maximum Likelihood Estimation** (MLE) directly the probability of gaining j yards as $\hat{P}(Y = j) = \frac{\sum_{i=1}^{n} 1\{y^{(i)} = j\}}{n}$
- **Gaussian Kernel MLE** (Kernelized MLE) fits MLE *YardsRemaining* and bags predictions with Gaussian kernel weights on *YardsRemaining* $K(x, z) = \exp\left(-\frac{||x - z||^2}{2\sigma^2}\right)$
- **Softmax Regression** (Sm. Reg.) was fit with L2 where the unregularized form is $\hat{P}(Y = j|x;\theta) = \frac{e^{\theta_j^T x}}{\sum_{i=1}^{k} e^{\theta_i^T x}}$
- **Random Forests** which fits multiple trees with random features
- **LightGBM** fits trees on residuals of prior trees and builds trees by leaf
- **XGBoost** fits trees on residuals of prior trees and builds trees by level

$$\mathcal{X} = \bigcup_{i=0}^{n} R_i \quad \text{s.t.} \quad R_i \cap R_j = \emptyset \text{ for } i \neq j$$

$$L_{Gini}(R) = 1 - \sum_{j=1}^{k}\left(\frac{\sum_{i \in R} 1\{y^{(i)} = j\}}{|R|}\right)^2$$

Results

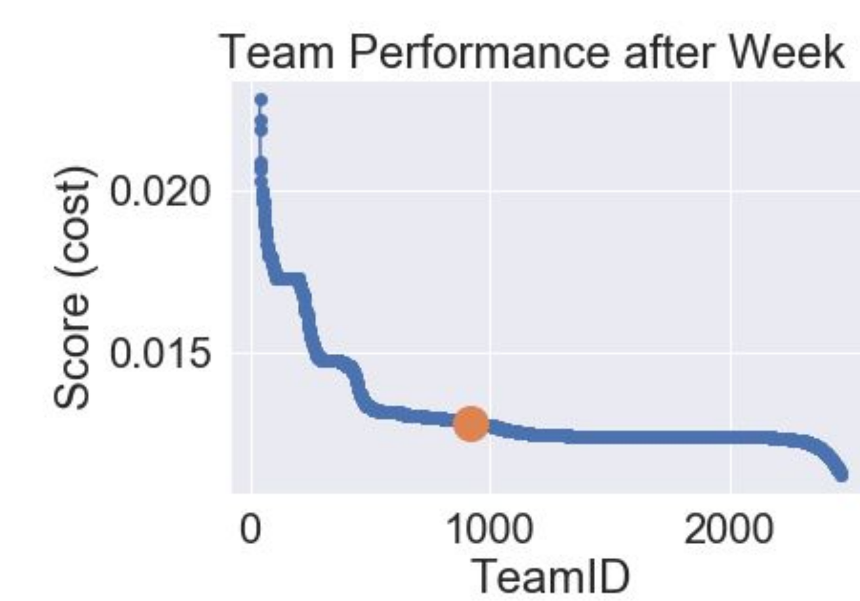
Model performance is summarized in the table.

Model	Train Error	Validation Error
MLE	0.01393	0.01404
Kernelized MLE	0.01380	0.01404
Sm. Reg.	0.01355	0.01366
Random Forests	0.01346	0.01361
LightGBM	0.02108	0.02123
XGBoost	**0.01326**	**0.01359**

There are 16k samples in train and 7k in validation.

Discussion and Future Work

- All models were found to perform reasonably well, with tree based methods outperforming the naive baselines. In particular, XGBoost performed best
- Gains in performance above the baseline were minimal due to the randomness of yards gained
- Given another six months, we would perform a more thorough feature selection/engineering and hyperparameter search.



Random forest was submitted; XGBoost was not yet fit.

References

1. Various Authors, Stanford CS229 Notes. Accessed on Dec. 10, 2019. *cs229.stanford.edu/syllabus.html*.