Worcester Polytechnic University

Kaggle NFL Big Data Bowl
A Survey of Regression Approaches

Barger Mia
Hershey Quincy
Moore Alexander
Prihar Ethan

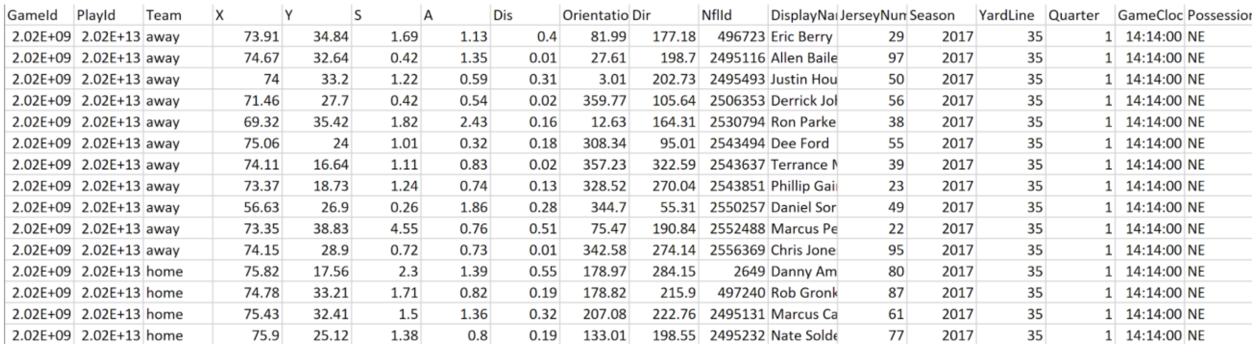
Data Science 502 / Mathematics 543
Oren Mangoubi
Fall 2019

Problem Description

This Kaggle competition sponsored by the NFL calls upon data scientists and football fanatics to team up in order to predict the yard gain of any running play. Understanding the game factors that correspond to yards gained on a play provides insight to the coaches, players, and fans of this 11-billion-dollar franchise. This insight can be used to improve the strategies of football teams and lead to more interesting, dynamic football games.

Data Description

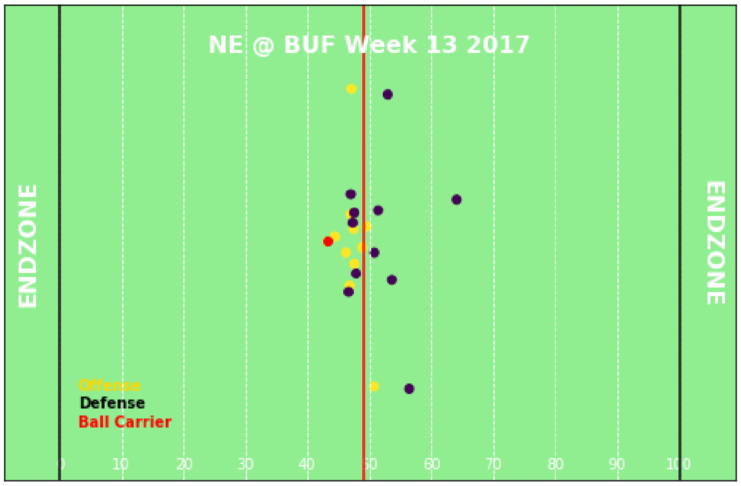
The format of the data provides unique challenges for inference and prediction. The data includes over 506,000 observations across 49 features. Each of these observations is a single player's involvement in a running play of a football game. Each observation contains player and game-state information. The player information includes the player's latitudinal and longitudinal position on the field, direction, speed, acceleration at time of handoff, and irrelevant features such as jersey number. The game-state information includes information which remains constant during a play, such as the team's positional strategy, offensive and defensive arrangements, stadium information, and weather information. A collection of 22 observations with the same PlayID gives the full available information for a single play, which must be transformed into a single sample. Below is an example of some of the available information. A description of all available features can be found in Appendix A.



GameId	PlayId	Team	X	Y	S	A	Dis	Orientatio	Dir	NflId	DisplayNa	JerseyNum	Season	YardLine	Quarter	GameCloc	Possessior
2.02E+09	2.02E+13	away	73.91	34.84	1.69	1.13	0.4	81.99	177.18	496723	Eric Berry	29	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	away	74.67	32.64	0.42	1.35	0.01	27.61	198.7	2495116	Allen Baile	97	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	away	74	33.2	1.22	0.59	0.31	3.01	202.73	2495493	Justin Hou	50	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	away	71.46	27.7	0.42	0.54	0.02	359.77	105.64	2506353	Derrick Jol	56	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	away	69.32	35.42	1.82	2.43	0.16	12.63	164.31	2530794	Ron Parke	38	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	away	75.06	24	1.01	0.32	0.18	308.34	95.01	2543494	Dee Ford	55	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	away	74.11	16.64	1.11	0.83	0.02	357.23	322.59	2543637	Terrance N	39	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	away	73.37	18.73	1.24	0.74	0.13	328.52	270.04	2543851	Phillip Gai	23	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	away	56.63	26.9	0.26	1.86	0.28	344.7	55.31	2550257	Daniel Sor	49	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	away	73.35	38.83	4.55	0.76	0.51	75.47	190.84	2552488	Marcus Pe	22	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	away	74.15	28.9	0.72	0.73	0.01	342.58	274.14	2556369	Chris Jone	95	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	home	75.82	17.56	2.3	1.39	0.55	178.97	284.15	2649	Danny Am	80	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	home	74.78	33.21	1.71	0.82	0.19	178.82	215.9	497240	Rob Gronk	87	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	home	75.43	32.41	1.5	1.36	0.32	207.08	222.76	2495131	Marcus Ca	61	2017	35	1	14:14:00	NE
2.02E+09	2.02E+13	home	75.9	25.12	1.38	0.8	0.19	133.01	198.55	2495232	Nate Solde	77	2017	35	1	14:14:00	NE

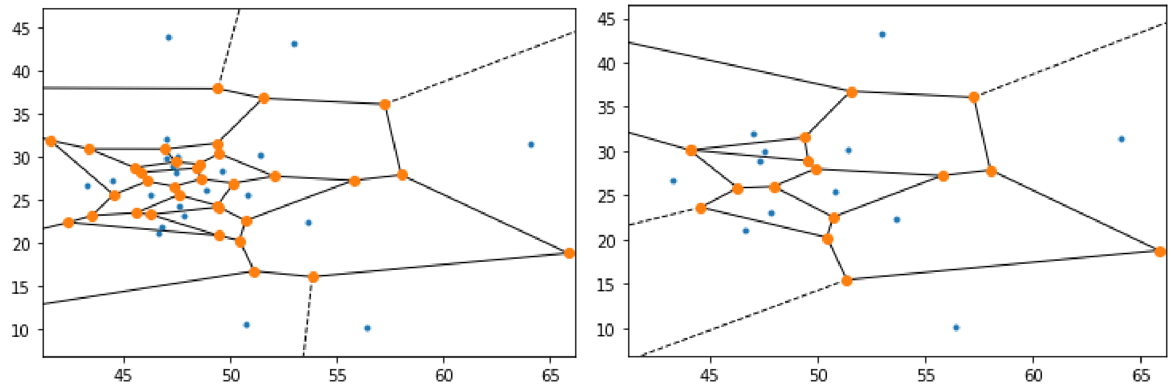
Data Visualization

Data visualization allowed manual inspection of the data, ensuring that data standardization had transformed the data as expected while also guiding the overarching analysis. Through visualization it was possible to analyze player behaviors and team formations as well as examine common characteristics between plays that had resulted in similar yardage gains. The figure below details player position for one specific high yardage play that occurred in 2017, after the data had been standardized with all offenses moving from left to right. It is obvious from visual inspection that a large offensive presence had created a hole in the defensive line in the middle of the field in front of the ball carrier, allowing for a significant gain.

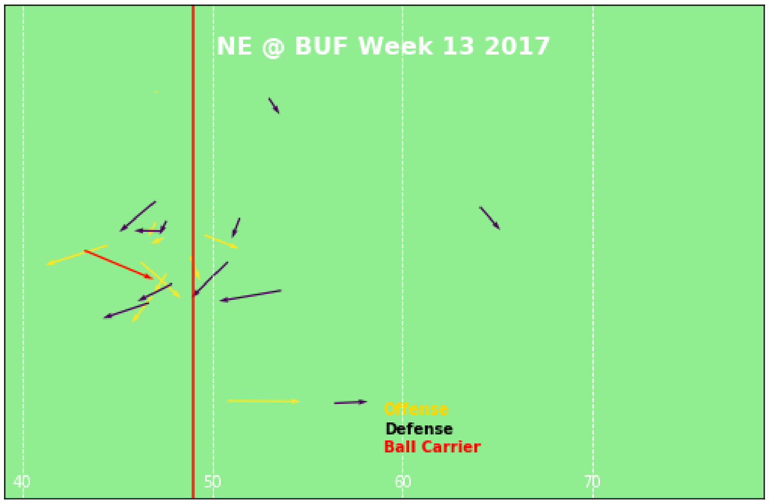


Voronoi Visuals

Voroni visualizations are another way to manually inspect the data for feature engineering purposes. Voroni diagrams visualize player distribution, spatial control, and field availability for runners. The examples below correspond to the same play visualized above, with the left figure showing all player positions, and the right figure indicating the ball carrier (leftmost point) and defensive players. In the right figure, the aligned slanted and horizontal lines directly ahead of the ball carrier indicate the presence of a gap and lack of sufficient offset among the players within the defensive formation. In this regard, the cascading horizontal and diagonal lines in front of the running back can be viewed as a crease or path of least resistance through the defense.



Further resulting from the data standardization, the vectorized player position diagram below indicates the same play as above but with the addition of player trajectories at the time of handoff. In this chart, each arrow length roughly corresponds to the anticipated travel over the next second of play. As expected, the crease through the defense is again obvious, but it also becomes clear that the ball carrier is moving at a high relative velocity compared to the other players, and that the other members of the offense are acting to either block or redirect the velocity of the defensive players away from the ball carrier.



These visualizations were used to develop features from the raw data, which could then be used to predict yards gained during a play.

Modeling Methods and Results

Model methods include all of our classical statistical learning predictive models. These methods all take the play data as an input and predict yards gained. Each method uses a training data set to train the model, a validation data set to optimize the model parameters, and an unseen testing data set to calculate a score for each model.

Naive Estimate

The naive estimate can be considered the most baseline model, upon which all others should improve. The naive estimator uses the mean of the training samples as the prediction for each testing sample, regardless of any data features.

The mean estimator has the advantage of ignoring the individual feature's coincidental relationship to yard gain, which is a benefit over models that can erroneously model the random relationships in data. We find the results of the mean estimator to be both the baseline and state-of-the art compared to classical regression techniques, due to the complicated and highly nonlinear relationship of the features to the yards gained.

To score the models discussed in the following sections, we calculate the percent improvement over the naive model's MSE. This makes the small and mostly insignificant improvements made by the following models clearer. To calculate this score, we used the following equation.

$$\frac{Naive\ MSE - Model\ MSE}{Naive\ MSE} \times 100$$

Parametric Models

Parametric Models include linear, polynomial, ridge, and lasso regression. Linear regression learns optimal coefficients for linear combinations of the predictors such that the MSE of the model is minimized over the training data.

Our interpretation of polynomial regression bends the typical formulation by allowing inter-feature products. This effort tries to remedy the concerns about intervariable relationships being significant to the prediction of yardage. The polynomial regression considers all possible pairwise combinations of features up to the second order.

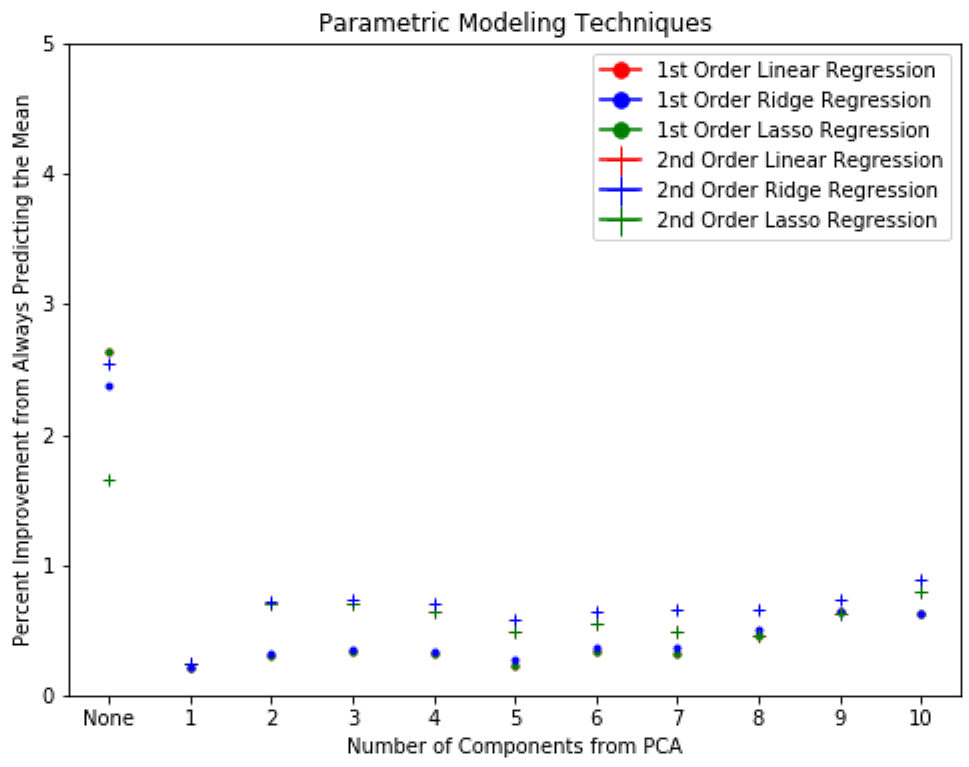
Lasso regression attempts to improve on the typical linear regression by enforcing a penalty term on the learned coefficients of the model. This penalty added to the SSE on the training data is equal to a hyper parameter lambda coefficient times the sum of the model coefficients. The goal of this penalty is to descend the bias-variance tradeoff by limiting the variability of the model. We optimized the lasso model's lambda parameter using cross-validation.

Ridge regression follows the same logic as the Lasso regression but squares the penalty lambda for the model coefficients. This change loses the benefit of driving some number of the coefficients to zero. We again used cross validation to find the optimal value of lambda.

Before creating any parametric models, we converted the raw data into usable observations of features which we extracted from the raw data. The following is the list of features we were able to extract for the purposes of linear regression: distance to touchdown, minutes since the start of the game, which down the play is on, how many yards to go until the first down, the time since the play started, the temperature, humidity, one-hot encoded offensive formation, the number of each kind of player on the offense, the number of defenders in the box, the number of each kind of player on defense, and the runners position, orientation, velocity, acceleration, height, weight, and age.

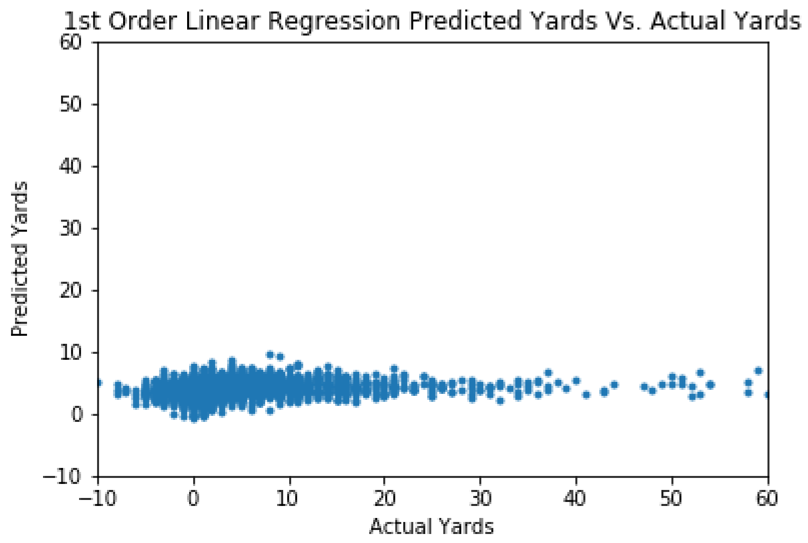
Once we extracted the features from the raw data, we used PCA on the features and calculated the ten most significant components, then we created first and second order linear, ridge, and lasso regression models using the original values, and the PCA components. In total, 66 models were created and used to predict the test dataset.

To compare these models to each other, below is a plot of the percent improvement each model made to the mean squared error of the naive model.



From the plot, one can see that using the original features has a more significant impact on improving from the naive model than regressing on the components from PCA. However, none of the 66 models made any significant improvements over the naive model. The best performing model, a first order linear regression using the original features, only had an MSE of 2.64% better than the naive model, which predicts the average of the training data.

To illustrate that the best parametric model is still not a good model, below is a plot of the model's prediction vs the actual yards gained during a play.

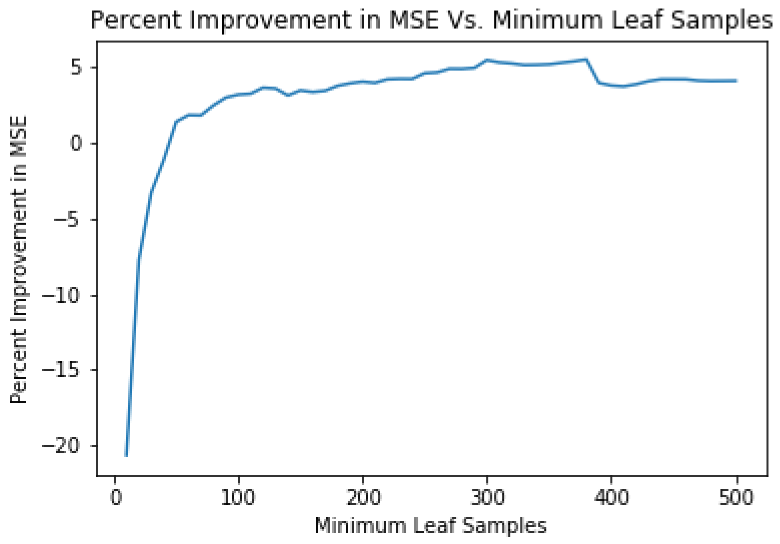


From the plot one can see that the linear regression is almost guessing the mean for each prediction, but with a slight increase in value as the actual yards increase, which is why there is a 2.64% increase in accuracy when compared to the naive model.

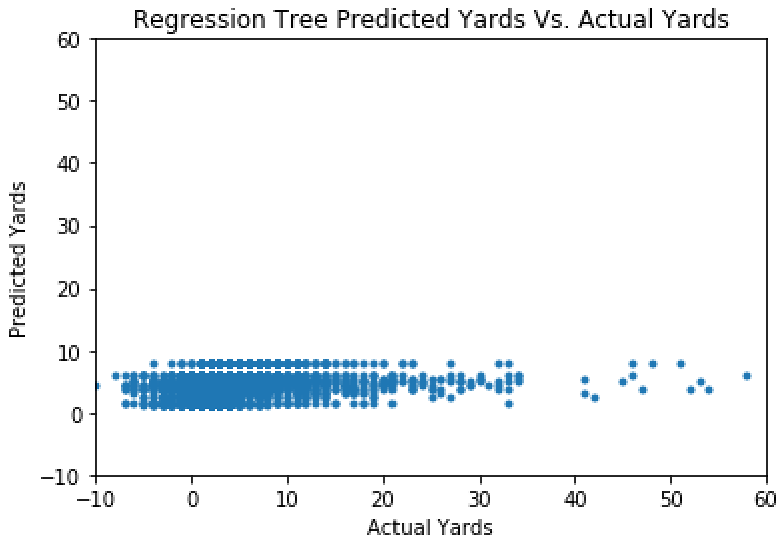
Regression trees

Regression trees are grown over iterative feature splitting which maximize the entropy of the sets the division creates. Regression trees are nonparametric and flexible, as well as able to model successive subsets given conditional features. This could have been a benefit for potentially finding data subsets where the runner has a spatial opening.

Using the same features extracted for the parametric models, and using the validation set to determine the optimal post-pruning strategy, the plot below shows the percent improvement in MSE from the naive model to the regression tree on the validation set when choosing a minimum number of samples to form a leaf node.



Based on the plot, a minimum samples per leaf value of 370 was chosen when predicting the test data. When the regression tree was used to predict the test data, an MSE improvement over the naive model of 5.06% was achieved. A plot of the tree's predictions vs. the actual number of yards gained is shown below.

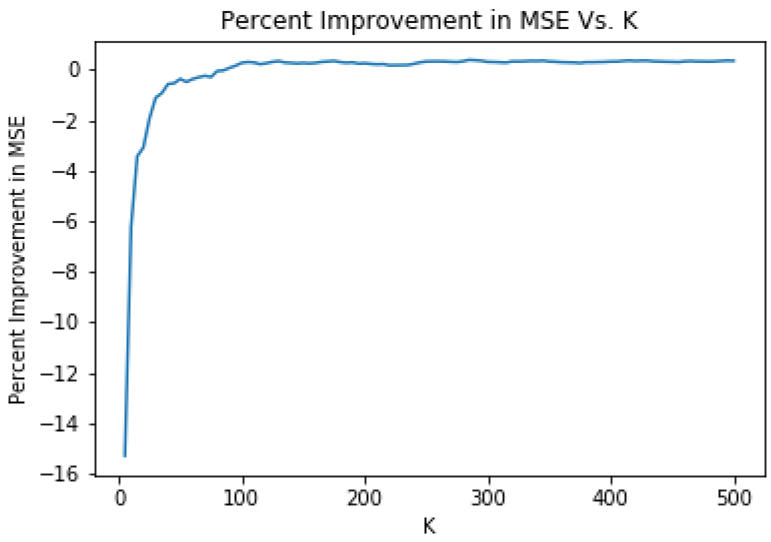


Just like the linear regression model, the regression tree is still predicting very close to the average yards gained regardless of input. The structure of the tree prevents it from ever predicting a large outlier because the values it can predict are confined to a set of values determined by taking a mean of many points. This regularization helps the tree outperform linear regression, but it still fails to positively correlate its predicted yards with actual yards, particularly for extrema.

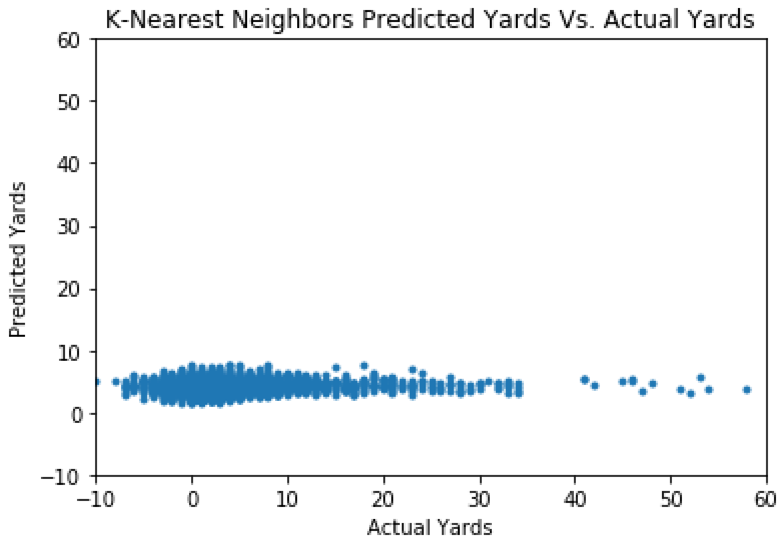
K-Nearest Neighbors

K-nearest neighbors is used to estimate the yardage gain of unseen plays by calculating a weighted average of the K-nearest neighbors yards gained, weighted by the Euclidean distance from the Kth neighbor to the current point. The downside of KNN is the high dimension of the data, where Euclidean distance loses meaning.

We used the validation set to determine the optimal number of neighbors. Using the plot below, 110 was chosen as the optimal K value.



Using a K of 110, the K-nearest neighbors regressor was able to predict yards 0.92% more accurately than the naive model on the testing data, worse than both the parametric models and the regression tree. A plot of the K-nearest neighbors predicted yards gained vs. actual yards gained is shown below, again the model barely predicts anything different than the average value.



Neural Networks

Feed-Forward Neural Network

The first network approach we tried was to offer the same information used in the previous regression techniques to a standard feed-forward neural network in order to learn the most meaningful representations from the features to predict yards gained. Using cross validation, a feed-forward neural network with 25 RELU nodes in a single hidden layer, trained using ADAM, was determined to be the optimal network structure. Using this structure, a percent MSE improvement over the naive model of 2.62% was achieved. The feed-forward network performed about as well as the linear regression.

Convolutional Network

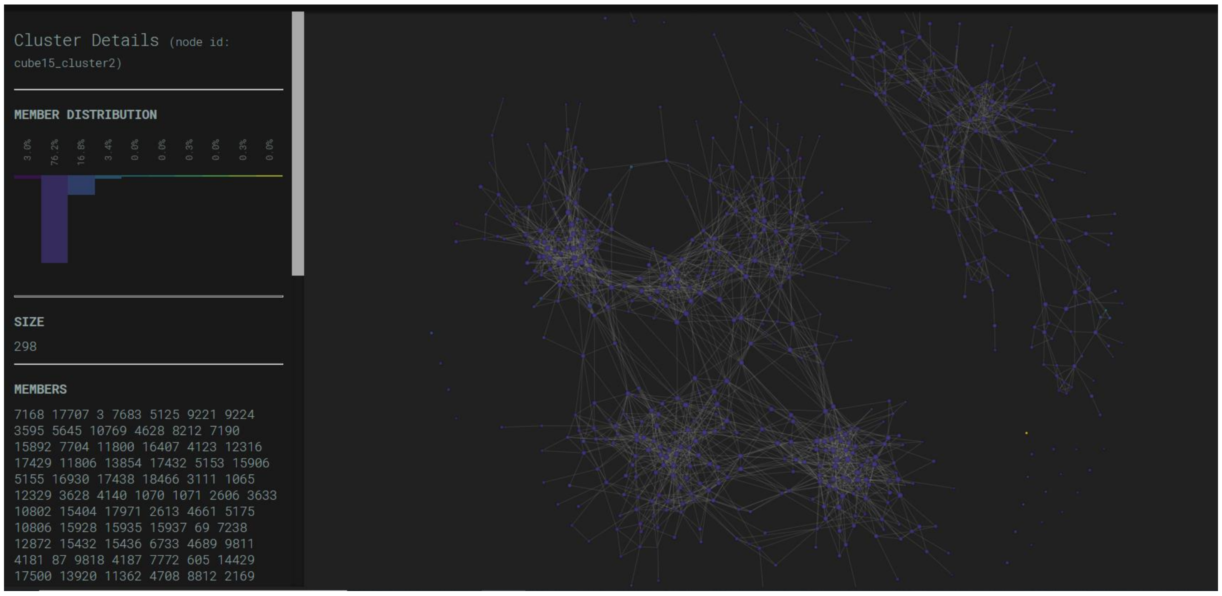
The convolutional network takes the idea that everything we need to know to predict yard gains is encoded in the play distribution over the field. This method works with minimal assumptions and preprocessing, by seeing the field as an image from a birds-eye view, where image depth at each pixel encodes the player data (such as height, velocity, direction, acceleration, age, and weight) at that location. This convolution extremizes the intuition that one could predict yardage by inspecting the player distribution at time of handoff. Ideally the convolutional network would derive human-like intuition on the nature of spatial availability of the runner, which is highly correlated to yard gain. In practice, this network likely failed to compete with the naive methods because of the complexity of the field and the huge spaces with no players. Reducing the dimension of the problem by squishing the image leads to overlapping players and difficulties with finding player openings in the reduced dimension. The relative performance is significantly worse than other models.

Split Neural Networks

Split neural networks try to remedy the difficulty of learning meaningful representations by splitting the load of predictors between two initial networks. One network only interprets game-state information which is constant between the two teams involved in the play, and the other network only interprets the player distribution information such as each players location, direction, velocity, and weight. These networks are then merged for a final hidden layer outputting the predicted yardage. This distributed load has the benefit of letting each half of the network compute more complicated representations among the feature subsets they handle. Combining all the features into one network creates vastly more potential combinations, making it is less likely the network will discover a deeply meaningful representation in the hidden layers. Other attempts at a split network model make an additional assumption on the nature of the player distribution data. The distribution of players over the field can be seen as a deep image, where X and Y positions are preserved and characteristics of players at each position are encoded into the depth of the data. This allows a convolutional network to learn the filtered representations that are meaningful to predicting yardage. However, an entirely convolutional network was shown to have too many features when modelling the entire field as an image. Ideally the network would learn how the weight, speed, and direction of players near the ball carrier affect the yardage of the play. The assumption that treating player data as an image corresponds to the complex spatial information encoded in the data. The split neural network comprises between these two methods to limit the dimensionality of the field convolution while still incorporating this spatial learning. The network simply learns to predict the mean of the target distribution regardless of predictive features and matches the naive methods. Both split networks failed to improve on the naive estimator.

Topological Data Analysis

It can be useful to visualize the data in order to understand its patterns more holistically, rather than from summary statistics and abstraction. To this end, we utilized PCA to determine the most significant singular values of the averages of the data. The first six singular values explained 95% of the variance. The first three singular values explained 87%. While 87% is not the high amount of variance hoped for in visuals, we did still use TDA to form a simplicial complex around the data projected onto its first three principal components in order to attempt to visualize its form. We chose to use the first three principal components purely for this visual as they still explained a large amount of variance while giving rise to a visual interpretation which can be readily understood. However, other calculations and predictions made on the data were formed using more principal components in order to give more accurate results.


Cluster Details (node id: cube15_cluster2)

MEMBER DISTRIBUTION

SIZE

298

MEMBERS

7168 17707 3 7683 5125 9221 9224
3595 5645 10769 4628 8212 7190
15892 7704 11800 16407 4123 12316
17429 11806 13854 17432 5153 15906
5155 16930 17438 18466 3111 1065
12329 3628 4140 1070 1071 2606 3633
10802 15404 17971 2613 4661 5175
10806 15928 15935 15937 69 7238
12872 15432 15436 6733 4689 9811
4181 87 9818 4187 7772 605 14429
17500 13920 11362 4708 8812 2169

Simplicial complex formed about a 3D projection of the data, colored by yards gained.

In order to construct this image, we projected the data onto its first three principal components, then clustered into nodes using K-means with a K of four. The nodes were assigned colors determined by the average amount of yards gained for all the points in the cluster. We then formed a simplicial complex about these nodes. It is interesting to note that the simplicial complex identifies two larger configurations and a number of outliers. Even colored by yards gained, there is no discernible difference between all of the clusters. When analyzing the clusters, it appears that the points in the clusters share close x-coordinate, y-coordinate, and offense formation. However, as stated earlier, these do not seem to have a significant correlation to the number of yards gained when considered individually.

Discussion and Conclusion

Overall, none of our models were able to significantly out-perform the naive model, where the mean of the training data was used to predict the test data. When comparing the models, it seems that the determining factor of quality is largely the model’s susceptibility to outliers. Linear regression and feed-forward neural networks both have unbounded outputs, and because the improvements over the naive model are so small, any outlier in the test data was significant enough to strongly affect the model’s percent MSE improvement score. The K-nearest neighbors model is bounded by the data because it’s output is always an average of a subset of training targets, but the best K-nearest neighbors model was using 110 neighbors, which is enough data to be almost equivalent to the naive model that averages all of the data. This is one reason K-nearest neighbors performed poorly, as the large dimension and noisy data makes defining neighbors less informative. Regression trees worked the best because they have the capacity to identify highly non-linear relationships in the data, as do neural networks, but the output of a regression tree is bounded, like in a K-nearest neighbors model to averages of different subset of the output, which are inherently resistant to outliers. The regression tree was able to capture some slight trends in the data without being susceptible to outliers, which is why its performance was the best of all the models.

As evidenced by higher-scoring users on the Kaggle leaderboard, more meaningful feature extraction is both possible and improves predictive accuracy. Of the predictive models, the performance of tree regression provides insights to the nature of the data. The asymptotic behavior of accuracy with tree simplicity shows that splitting on independent features fails to separate magnitudes of yardage gain. Since the tree algorithm should split on maximizing information gain, this demonstrates why other models would fail due to lack of intervariable interaction. Moreover, intervariable interaction from a critical perspective alone is not enough to translate football plays into accurate yardage prediction. The variables present describe only the initial state of a complex game where 22 players constantly update their velocities, trajectories, and coordinates. The impasse between these two understandings of football are negotiated in such data comprehension methods as Voroni diagrams, whereby inspection one might manually separate low and high yardage gains or turn to cumulative density of yardage methods to understand coarser but more generally accurate data representations. Our findings corroborate the intuition that yard gain of running plays is a very challenging regression problem. Football plays naturally encode some degree of randomness, where the same player and game state distribution could potentially correspond to any number of potential yardage gains. A player with few running possibilities might be tackled within feet of ball reception, or potentially go on to a 99-yard touchdown. These kinds of stochastic processes make this regression highly variable even with perfect information extracted from features.

Appendix A: Detailed Data Description

- GameId - a unique game identifier
- PlayId - a unique play identifier
- Team - home or away
- X - player position along the long axis of the field. See figure below.
- Y - player position along the short axis of the field. See figure below.
- S - speed in yards/second
- A - acceleration in yards/second^2
- Dis - distance traveled from prior time point, in yards
- Orientation - orientation of player (deg)
- Dir - angle of player motion (deg)
- NflId - a unique identifier of the player
- DisplayName - player's name
- JerseyNumber - jersey number
- Season - year of the season
- YardLine - the yard line of the line of scrimmage
- Quarter - game quarter (1-5, 5 == overtime)
- GameClock - time on the game clock
- PossessionTeam - team with possession
- Down - the down (1-4)
- Distance - yards needed for a first down
- FieldPosition - which side of the field the play is happening on
- HomeScoreBeforePlay - home team score before play started
- VisitorScoreBeforePlay - visitor team score before play started
- NflIdRusher - the NflId of the rushing player
- OffenseFormation - offense formation
- OffensePersonnel - offensive team positional grouping
- DefendersInTheBox - number of defenders lined up near the line of scrimmage, spanning the width of the offensive line
- DefensePersonnel - defensive team positional grouping
- PlayDirection - direction the play is headed
- TimeHandoff - UTC time of the handoff
- TimeSnap - UTC time of the snap
- Yards - the yardage gained on the play (you are predicting this)
- PlayerHeight - player height (ft-in)
- PlayerWeight - player weight (lbs)
- PlayerBirthDate - birth date (mm/dd/yyyy)
- PlayerCollegeName - where the player attended college
- Position - the player's position (the specific role on the field that they typically play)
- HomeTeamAbbr - home team abbreviation
- VisitorTeamAbbr - visitor team abbreviation
- Week - week into the season
- Stadium - stadium where the game is being played
- Location - city where the game is being played
- StadiumType - description of the stadium environment
- Turf - description of the field surface
- GameWeather - description of the game weather
- Temperature - temperature (deg F)
- Humidity - humidity
- WindSpeed - wind speed in miles/hour
- WindDirection - wind direction

Bibliography

1. James, Gareth, et al. An Introduction to Statistical Learning: with Applications in R. Springer, 2017.
2. https://www.kaggle.com/c/nfl-big-data-bowl-2020, 11 November, 2019, Kaggle.
3. Barger, Hershey, Moore, Prihar (2019). Project4, DS501. Worcester Polytechnic Institute.
4. Stats by Michael Lopez, Michael Lopez. “NFL Tracking: Wrangling, Voronoi, and Sonars.” Kaggle, Kaggle, 15 Oct. 2019, https://www.kaggle.com/statsbymichaellopez/nfl-tracking-wrangling-voronoi-and-sonars.

All code is available in the project file, as well as online at https://github.com/moorea1/DS502_Final