Predicting Kick Return Yardage on Punting Plays in the NFL

Kelvin Tiongson
Data Science, Lewis University
Romeoville, USA
KelvinJTiongson@lewisu.edu

***Abstract*—In American football, there are three main parts in the game: offense, defense, and special teams. Special teams is a part that tends to get less focus. In this paper, I attempt to predict the result of the receiving team yardage gained during a special teams play using self-organizing maps, linear regression, and multiclass logistic regression. In building the models, I use certain features of a dataset provided by the National Football League. Since there are a vast number of features available, I implement self-organizing maps to find the more important ones. Self-Organizing Maps is advantageous because it clusters the samples together based on similarity throughout the features, thus identifying which ones are essential within a dataset. Then I use those features in a linear regression and multiclass logistic regression model. I use the multiclass logistic regression model by binning the yardage and using the bins as the different classes for the dependent variable.**

***Keywords*— Self-Organizing Map, Linear Regression, Multiclass Logistic Regression**

I. Introduction

Since 2018, the National Football League hosts an analytics contest known as the Big Data Bowl [1]. The NFL shares their competition worldwide through Kaggle allowing the general public to compete and discover potential new metrics for professional American football. So far, the topics of the Big Data Bowl have been focused on wide receivers, running backs, and defensive backs. Wide receivers are players that catch the ball from the passer on offense, running backs are the players that run the football on offense, and defensive backs are the players on the defense that defends the pass catchers from catching the ball. This year's competition aims to evaluate performance on special teams [2].

Special teams in American football refers to the unit that has to deal with situations in which the ball is kicked for points or change of possession. When a team is kicking for points, this situation is known as a "field goal" attempt. Here, the kicking team is attempting to add three points to their score by kicking the ball through a goal post. In this situation, the other team is looking to block the attempt. The defense can also get the ball and return it back if the kick does not make it past the endzone. When a team is kicking the ball to change possession, this happens during kickoff which is the beginning of the game or after the team has scored. On offensive possessions, teams only get four attempts to score or get past a certain point on the field. Otherwise, they must give the ball to the other team to allow them to score. During an offensive possession, the team can kick the ball to the other team to change possession which is known as a "punt". On punts, the kicking team is trying to kick the ball further to put the receiving team in a worse position to start their offense. The receiving team can get the ball and attempt to run it back to put their team in a better position. The receiving team can also score a touchdown if they are able to make it that far without getting tackled.

This paper looks to focus on punting during special teams. Punting is a unique part of American football that can help make a difference for a team when changing possessions. Punting can help put the defense in a better position to defend or the offense in a better position to score. There has been quite a few research on American football with the past couple of NFL Big Data Bowls, but they have not been centered around special teams. However, some work can be used and built upon along the use of this topic. For example, one of the grand finalists of the 2019-2020 Big Data Bowl used domain knowledge to implement feature engineering to identify key features that led to the success of estimating yards for a running back [3]. Here, the author added over forty new features that he believed were important to the outcome of the estimated yards for a running back. He then utilized a Pearson correlation between the features and the outcome of the variable of the yards gained to identify what positively or negatively correlates with estimated yards. After applying xgboost to obtain point estimates for test set observations, he took a look at the model and the features included to identify the most important ones. This same technique can be used to estimate yards for kick returners on special teams plays that involve the return team returning the kick.

Another example of a winner in last year's data bowl quantified performance of a defensive back using logistic regression [4]. The author decided to look and compute probability models using logistic regression at different moments within a football passing play. The author looked at three different moments. The first moment is when the ball is snapped to the moment the ball is thrown. The second moment is when the ball is thrown to the moment the ball reaches the receiver. The third moment is when the ball reaches the receiver to the actual outcome. Then she furthered her evaluation of players through clustering. A third example of winners in the first big data bowl used clustering to identify passing route patterns that consistently yielded positive outcomes [5]. All researchers utilized the tracking data in some way, which is what I intend to do and is mentioned later in this proposal.

In section II, I dive into the data from all the datasets that are provided from the data bowl. This section includes a description of some important features. Section III starts with feature engineering. This describes new features that were generated from the tracking dataset. This section also goes over more preprocessing that was required for the unsupervised and supervised learning techniques. Then I discuss the implementation of SOM, linear regression, and multiclass logistic regression. In section IV, the

results of the techniques are discussed. There were some adjustments to each technique in order to achieve better results. Last, section V concludes the work of this paper and considers future work.

II. Data Description

This year's data bowl includes five datasets. Primarily, I will take a look at the plays, tracking, and scouting datasets that are provided. The plays dataset includes certain information from each play for each game. The tracking dataset includes tracking data of the players on the field at each stamp of the play. The scouting dataset is something that is new this year and is brought over from Pro Football Focus which includes scouting information for each play within a game. The tracking dataset that was provided contain datasets from 2018, 2019, and 2020. All tracking datasets have at least 1GB of data. As a result, I will initially look at a few rows of data from the 2020 season. The datasets have a common identifier with the game ID and the play ID. The game ID and play ID are unique except the play ID is not unique across games. Therefore, when looking for specific plays, it was important for me to query data with the matching game and play ID. There were a few features that were derived from the dataset. Section III discusses how the features from the tracking data were generated. Combining all datasets together led to an enormous number of features. Table I shows a description of a few of the features.

TABLE I. FEATURES

Attribute	Type	Example Value	Description
WEEK	Ordinal (real)	1	Week of the season
QUARTER	Ordinal (real)	2	Quarter of the game
KICK LENGTH	Numeric (real)	66	Kick length in the air of kickoff or punt
KICK RETURN YARDAGE	Numeric (real)	5	Yards gained by return team if there was a return
YARD LINE NUMBER	Numeric (real)	22	Yard line at line-of-scrimmage
YARDS TO GO	Numeric (real)	10	Distance needed for a first down
PRESNAP HOME SCORE	Numeric (real)	20	Home score prior to the play
PRESNAP VISITOR SCORE	Numeric (real)	22	Visitor score prior to the play
OPERATION TIME	Numeric (real)	14	Timing from snap to kick in seconds
HANG TIME	Numeric (real)	20	Hangtime of kick attempt in seconds
SNAP DETAIL	Nominal (string)	H	Whether snap was on target or not
KICK TYPE	Nominal (string)	D	Kickoff or punt type
KICK DIRECTION INTENDED	Nominal (string)	L	Intended kick direction
KICK DIRECTION ACTUAL	Nominal (string)	R	Actual kick direction
RETURN DIRECTION INTENDED	Nominal (string)	R	Intended kick return direction
RETURN DIRECTION ACTUAL	Nominal (string)	L	Actual kick return direction
KICK CONTACT TYPE	Nominal (string)	CFFG	Detail on how punt was fielded

III. METHODOLOGY

A. Feature Engineering

The tracking data provides the geometric location of the players from both teams and the football on the field at all seconds of a specific play. There are several moments in a play that a tracking data can identify as specific events. With the help of the Kaggle community and the matplotlib library documentation, I was able to graph the location of the players and the football on a football field [6].

Fig. 1. Beginning of the play

Fig. 2. Moment the ball was punted by the kicking team

Fig. 3. Moment the ball was received by the receiving team

Figure 1 represents the moment the ball was snapped which was the beginning of the play. Figure 2 represents the moment the ball was kicked. Figure 3 represents the moment the ball was received by the returner on a punt return play. In these plays, the navy dot represents the Chicago Bears, sky-blue dots represent the Detroit Lions, and the yellow dot represents the football. The goal of this paper was to predict kick return yardage. Figure 3 has the most interesting point of view in all figures since this is the instance that the receiving team first gets the ball. At this second, I generate new features to use in my model such as the distance from the returner to other players on the field. The features that I derive are the kicking team mean distance from the returner, kicking team distance standard deviation from the returner, kicking team minimum distance from the returner, and the same three metrics for the returning team. However, the aggregate brought back the returner distance as the minimum return team distance, so this was not used. In order to obtain the features from the tracking dataset, I used Pandas "groupby" function. First, I grabbed the punt plays in the tracking dataset at the moment the ball was received by the return team. Then I grouped the data by the game, play, and team. Next I iterated through each group, identified the returner by their NFL ID, and calculated the distance from each player to the returner. There was one play that had to be removed because there were two returners on the play. The rest of the punt plays only had one returner.

B. More Preprocessing

After acquiring features from the tracking dataset, I looked to preprocess the features from the PFF dataset. I applied one-hot encoding to these features and looked at their relationship with a correlation matrix. The correlation of each is shown through a correlation heatmap in the figures below. Since there were several features that came over from PFF, I split out their correlation and focused on the one-hot encoded that were transformed from their original column. Originally, I had put all the features in one correlation matrix, but the heat map was too congested to label and was difficult to pinpoint the correlation between features. The features in Figures 4 and 5 do not show high correlation with each other. However, from Figures 6 and 7 the intended kick and return direction are highly correlated with the actual kick and return direction. Therefore, I remove the features intended kick and return direction moving forward.

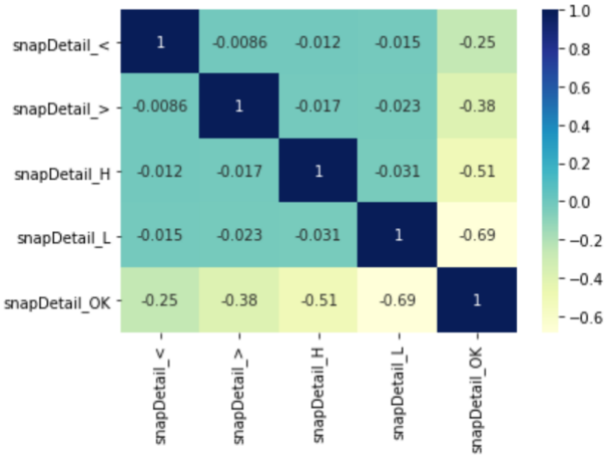

Fig. 4. Snap detail correlation

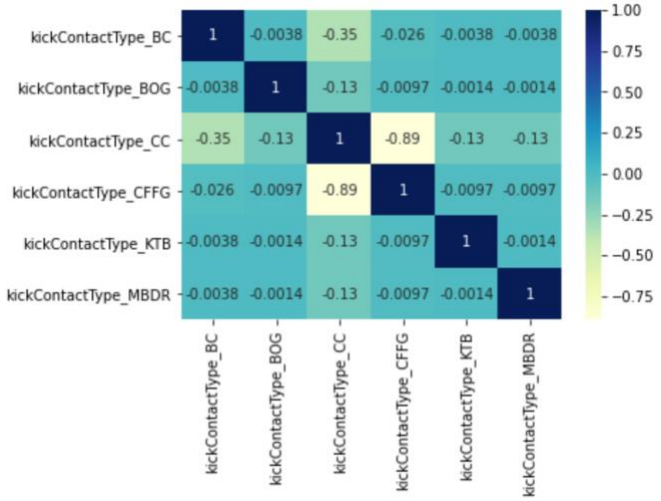

Fig. 5. Kick contact type correlation

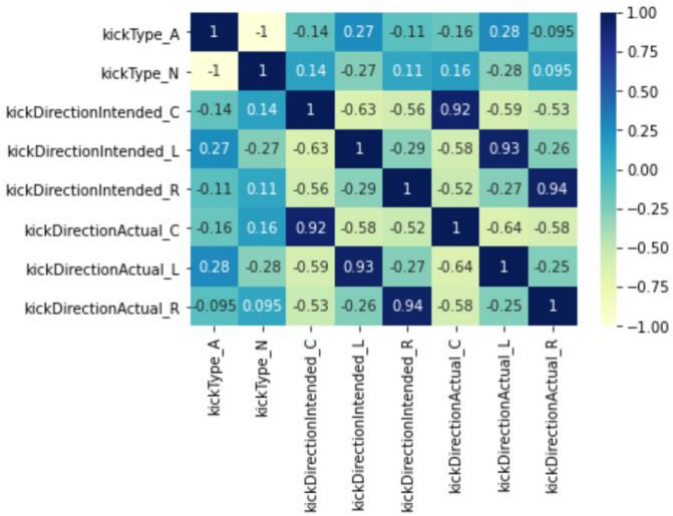

Fig. 6. Kick type and kick direction correlation

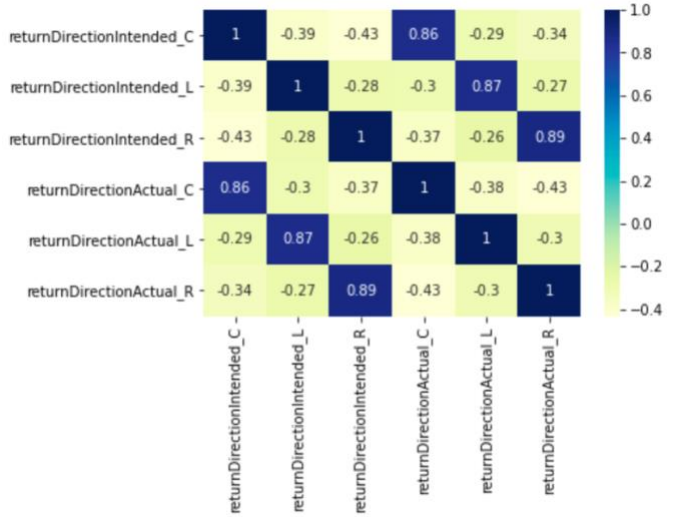

Fig. 7. Return direction correlation

C. Self-Organizing Map

Now that I have my features in place, I begin my application of a self-organizing map. One of the reasons for using SOM is to identify features that are important to use in the regression models. Using the technique, I generate neurons that represent different clusters of the points. In each neuron, the points will share the same set of features. Those features will be the features that indicate importance in the dataset. There is a python library known as "MiniSom" that I used in the application of the self-organizing map [7]. In order to proceed with the minisom class and its methods, I needed to only use the numerical features that are in the dataset and remove null values in the target feature. Starting out, the preprocessed data frame had 72 features. I dropped several columns that I believed were not useful such as game ID, play ID, IDs of players in specific roles on the fields, and many more. I also dropped the columns that had high correlation with each other that was discovered in the preprocessing phase above such as kick direction intended and kick return intended. After dropping the unnecessary columns, I was left with 36 features. Kick return yardage also needed some preprocessing as some of the values had a NaN value. I replaced the NaN values with a 0 interpreting them as a return that produced zero yardage. Then, in order to produce a better SOM mapping, following the package's documentation I changed the kick return yardage into a categorical column that represents the bins of the kick return yardage. I used the pandas cut method to create 4 bins. The bins are less than zero, zero to ten, ten to twenty, and greater than twenty. Next, I normalized the data so that the values of one feature do not affect the other if they are not on the same scale.

The MiniSom class takes a few parameters when initializing. The parameters are the x and y dimension of the SOM map, number of elements of the input, sigma which is the spread of the neighborhood function, and learning rate. The neighborhood function used was Gaussian and the activation distance method used was Euclidean. To start off, the default sigma was 1.5 and the learning rate was 0.5. The documentation also recommended the use of their method "pca_weights_init" before training the SOM. This method initializes the weights to span the first two principal components. Next, I trained the SOM with the dataset with a maximum of 1,000 iterations in random order. The results of the self-organizing map are discussed in the next section.

D. Linear Regression and Multiclass Logistic Regression

From the self-organizing map, I was able to identify the features that seem to be most important. Then I used those features to build the linear regression model with kick return yardage as the target variable. To build the model, I used python's sci-kit learn package [8]. In early experimentation, I was interested only in building a linear regression model. But after seeing the outcome of the linear regression model, which are discussed in the next section, I was interested in seeing how a logistic regression model would perform since I categorized kick return yardage in the self-organizing map. I also used the sci-kit learn package [9]. This led to better results. The logistic regression class from sci-kit learn will be able to detect my target variable as a multiclass variable. In doing so, it will switch the training algorithm to use cross-entropy loss. To train the model, the dataset was split 70% training and 30% testing.

IV. Results and Discussion

A. Self-Organizing Map

The results of the self-organizing map are shown in the figures below. Figure 8 is the SOM with the initial parameters.

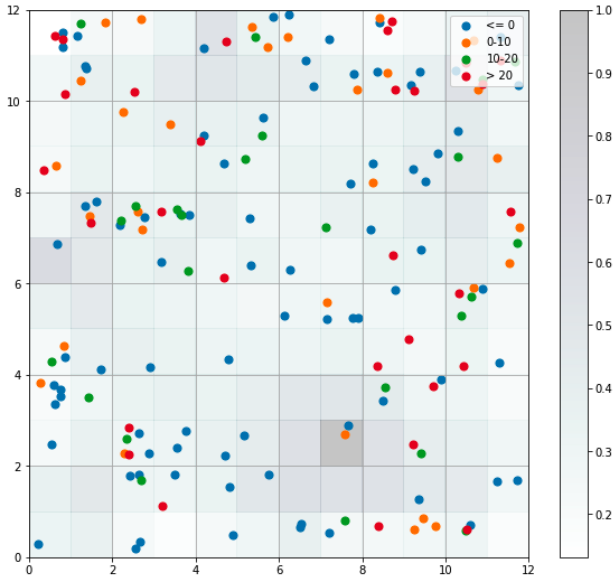

Fig. 8. SOM with initial parameters

In Figure 8, there are no neurons or clusters to be found. After changing different values for the parameters, I was able to successfully generate clustering of the dataset. Figure 9 below represents the successful clustering of the dataset. The parameters that I changed were the increase of the dimensions of the SOM to be 50x50, sigma to 1.0, and increase the number of max iterations to 10,000.

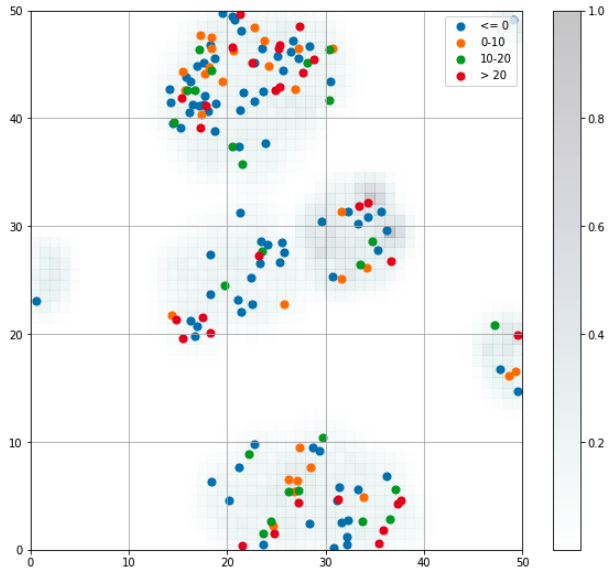

Fig. 9. SOM with adjusted parameters

From the map, you can see that there are at least 3 distinct clusters. There is a cluster at the top, middle, and bottom of the graph. There is one point in the left side of the map and there looks to be another cluster on the right side of the map. But since a majority of the points lie in the top, middle, and bottom clusters, I decided to focus on these 3.

The different colors of points represent the bin that they respectively belong to in the target variable Kick Return Yardage. The

blue points represent samples where kick return yardage was less than or equal to zero. The orange points represent samples where kick return yardage was between 0 and 10. The green points represent samples where kick return yardage was between 10 and 20. The red points represent samples where kick return yardage was greater than 20. Investigating each cluster more, I looked at 5 different points for each bin in each cluster. From observation, the features that seemed to be most important were the features derived from the tracking data, kick contact type, return direction, kick length, and hang time.

B. Correlation Coefficient

The features that I found to be most important from the self-organizing map through observation were the tracking data features, kick contact type, return direction, kick length, and hang time. To look further in the importance of these features, I take 5 samples of two categories from each cluster. From these samples, I determine similar features between them by looking at their correlation coefficient. The similar features are the features that are in fact most important for the clusters. The next couple of features are correlation heatmaps that show the correlation coefficients of the samples.

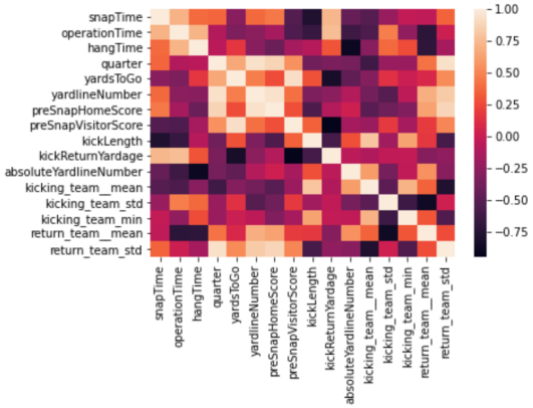

Fig. 10. Bottom cluster: 1-10 yards

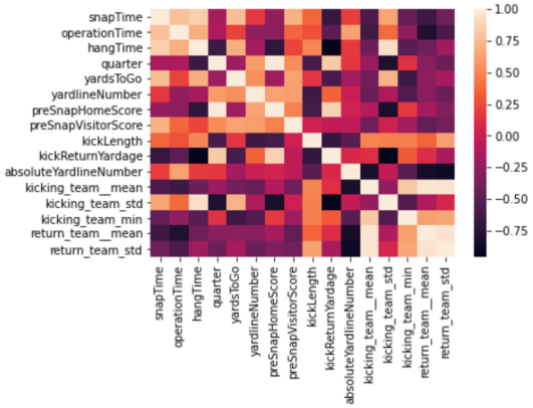

Fig. 11. Bottom cluster: 10-20 yards

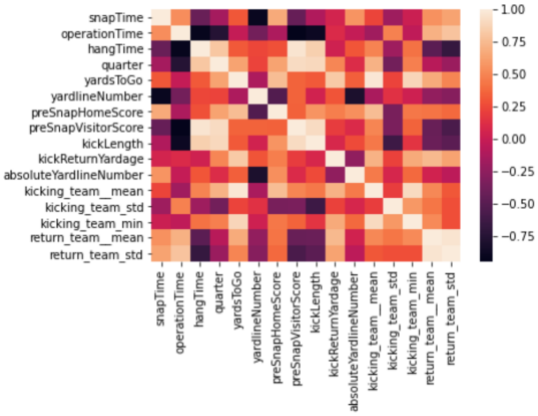

Fig. 12. Middle cluster: 1-10 yards

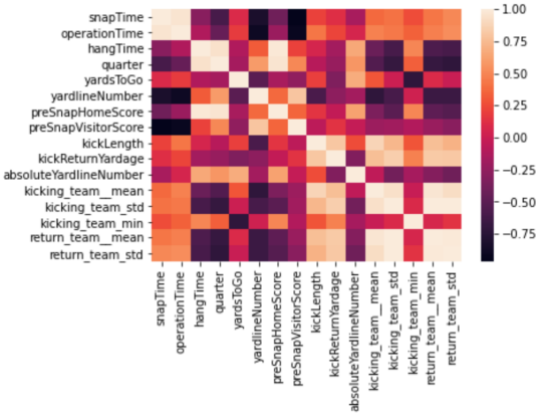

Fig. 13. Middle cluster: 10-20 yards

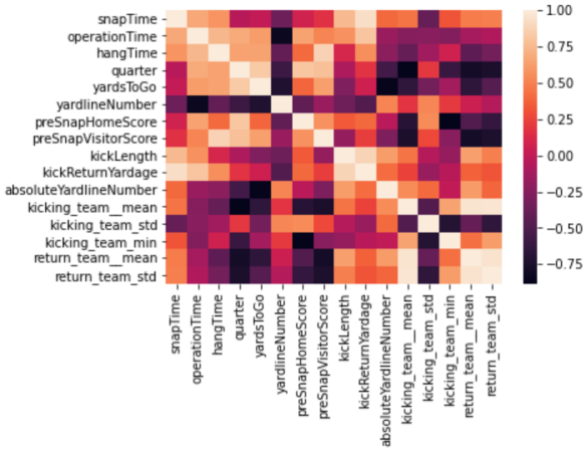

Fig. 14. Top cluster: greater than 20 yards

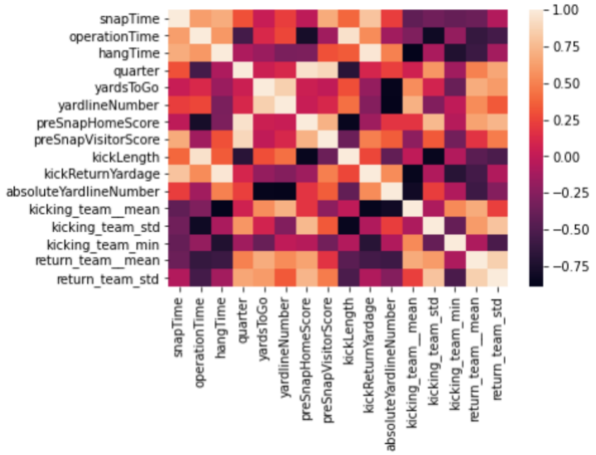

Fig. 15. Top cluster: 1-10 yards

In the six correlation coefficients that are displayed, there are a variety of similar features for each. Overall, the coefficients show that my observation of the important features are correct. In a majority of the coefficients, the tracking data features, kick length, and hang time are similar across the clusters for the samples.

C. Linear Regression and Multiclass Logistic Regression

As mentioned earlier, my first regression model was linear regression using the features I found to be most important from the self-organizing map with kick return yardage as my target variable. My initial coefficient of determination was 0.205. Since this was really low, I reran the regression using different combinations of input features. Unfortunately, the highest coefficient of determination I could get was 0.231. Since the linear regression model needed significant improvement, I decided to look towards the results of the logistic regression model. Using the same dependent and independent variables, I was able to achieve an accuracy of up to 0.562. Figure 16 represents the confusion matrix along with the precision, recall, and f1-score.

```
Confusion Matrix:
[[ 5 20  6  0]
 [ 3 95 11  0]
 [ 2 36 20  0]
 [ 0  6  7  0]]

              precision    recall  f1-score   support

           0       0.50      0.16      0.24        31
           1       0.61      0.87      0.71       109
           2       0.45      0.34      0.39        58
           3       0.00      0.00      0.00        13

    accuracy                           0.57       211
   macro avg       0.39      0.34      0.34       211
weighted avg       0.51      0.57      0.51       211
```

Fig. 16. Confusion matrix and classification report

In an attempt to improve the accuracy of the logistic regression model, applied cross validation to the technique in three-folds and ten-folds [10]. Unfortunately, the scores were no better than what I received with my model. So, in another attempt to improve the model I removed samples where kick return yardage was greater than 25 and samples where kick return yardage was null instead of replacing the value with zero. This improved the accuracy of my logistic regression model to 0.596. Since this

improved the accuracy of my logistic regression model, I applied cross validation again. The cross validation with three-folds did not produce better results but, cross validation with ten-folds did. Table II below shows the scores.

TABLE II. CROSS-VALIDATION WITH TEN-FOLDS

Accuracy
0.6029
0.5147
0.5588
0.5588
0.5588
0.5373
0.5522
0.5522
0.6418
0.6119

From the results of the cross-validation there are some scores that perform lower than the new logistic regression model's score of 0.596. However, some of the scores from Table II do show a new threshold by reaching accuracy over sixty percent.

V. CONCLUSION

From the data that was provided for the big data bowl, I merged attributes from the datasets together and generated new attributes from the tracking dataset. The generated features are derived from the moment a ball was received by a returner. Then I applied more preprocessing by one-hot encoding some of the features from the PFF dataset. After one-hot encoding the features, I took a look at their correlation coefficient and removed the columns that had high correlation with each other. Once the features were finalized, I implemented a self-organizing map to cluster the similar samples together. To improve the results of the SOM, I binned the return yardage into four categories: less than or equal to zero yards, one to ten yards, ten to twenty yards, and greater than twenty yards. The cluster of the SOM put samples together that had similar features. The features that were similar were considered important. I created more correlation heatmaps to further examine the importance of these features. Once the features were fully examined, I used the features to build a linear regression and multiclass logistic regression model. To improve the accuracy of the logistic regression model, I removed outliers and null values in the target variable and applied cross validation with three-folds and ten-folds.

There are many variables that are at play in the attempt of successfully making the prediction of kick return yardage on special teams plays. There are also many outliers that make the forecast challenging. However, there is plenty of potential for future work that could be applied to improve the results that I had acquired. The bins could change to look for smaller yard returns that do not include return yardage greater than 20. Other machine learning techniques could be used such as Bayesian Optimization, Neural Networks and other variations such as Convolutional Neural Networks, or boosting methods. More features could be engineered with the tracking data. These are just a few improvements out of several that could be made. Predicting kick return yardage on punt plays was not as successful as I would have liked it to be. However, this paper provides a foundation for future work to be applied.

REFERENCES

[1] *Past Big Data Bowl Recaps.* NFL Football Operations. (n.d.). https://operations.nfl.com/gameday/analytics/big-data-bowl/past-big-data-bowl-recaps/

[2] *NFL Big Data Bowl 2021*. Kaggle. (n.d.). https://www.kaggle.com/c/nfl-big-data-bowl-2022/overview

[3] Ploenzke, Matt. (2019, December). *NFL Big Data Bowl Sub-Contest.* NFL Big Data Bowl. https://operations.nfl.com/media/4204/bdb_ploenzke.pdf

[4] Reiner, Jill. (2020, December). *Evaluating and Clustering Coverage Skill.* NFL Big Data Bowl. https://www.kaggle.com/reinerj1/evaluating-and-clustering-coverage-skill

[5] D. Chu, L. Wu, M. Reyers, J. Thomson. (2018, December). *NFL Big Data Bowl.* NFL Big Data Bowl. https://operations.nfl.com/media/3670/big-data-bowl-sfu.pdf

[6] Mulla, Rob. (2020, December). *NFL Big Data Bowl – Plotting Player Position.* NFL Big Data Bowl. https://www.kaggle.com/robikscube/nfl-big-data-bowl-plotting-player-position

[7] *MiniSom*. (n.d.). https://github.com/JustGlowing/minisom

[8] *Linear Regression.* Sci-Kit Learn. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[9] *Logistic Regression.* Sci-Kit Learn. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[10] *Cross-Validation.* Sci-Kit Learn. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html