THE TAXI GROUP 26

Predicting running play yardage in football

Machine Learning

Hunter Copp, Will Hunnicutt, Cole McGaughey, and Dan Mulloy

CS 4641 - Professor Borela

For our project, we are trying to create a model that can predict the outcome of an NFL handoff on a given play. This would be able to help football analysts and coaches better decide what plays to run that give them the highest number of expected yards gained given the current situation. The result of a play will fall between losing 100 yards and gaining 100 yards. Except in rare cases, a play will not lose more than 5 yards or gain more than 20. We initially saw this as a regression problem because the result was a number, but we determined it would be better to classify a range of yards gained.

Dataset

The dataset came from Kaggle and includes data for each player on the field per play (682,154 total data points of run plays). Since there are 22 players on the field per play, this means we have 31,007 unique run plays in our data set. Our data also has 49 different features for each data point. We decided to focus on run plays due to their decreased feature set (from 255 down to 49). The dataset includes features such as location, teams, down, and distance.

Data Cleaning

Before we implemented any learning methods, we wanted to get a feel for the distribution of yards gained per play. We created a histogram of bin length 1 yard and the y-axis being the number of times it occurred. This is shown in Figure 1. Looking at the histogram, we noticed that the distribution of yards gained follows a gaussian distribution with an average of 4.22 yards and a standard deviation of 6.44 yards. We found that 25% of the runs were a yard or less and 25% of the runs were over 6 yards. Therefore, we concluded that if a play gained less than -6.5 yards or more than 13.5 yards, it was an outlier. We had 31,007 unique plays and deemed 1865 (6.012%) to be outliers.

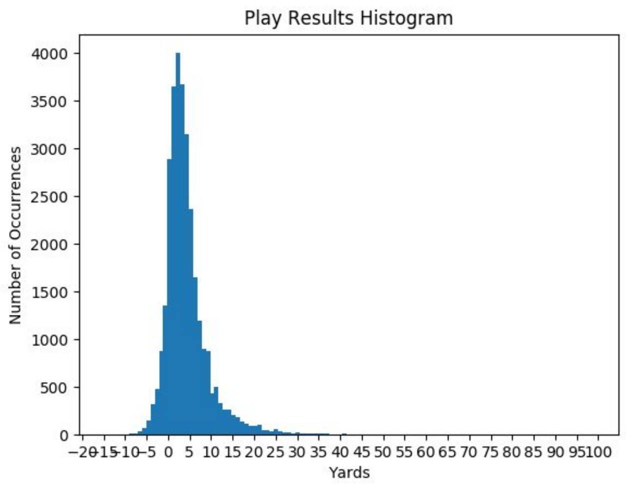

Figure 1 | Histogram consisting of each plays result

Once we settled on the data set and were familiar with the distribution of yards, we then had to decide on what features to use. When choosing features, we wanted to choose the ones that had a direct correlation with yards gained. We also wanted to eliminate or combine features where it made sense. To give an example of why we would eliminate a feature, we are given “Stadium” and “Location” as 2 different features. The stadium never changes where it is located, so the location feature is unnecessary to use. To give an example of why we would combine a feature, we are given “TimeSnap”, which is the time when the ball was snapped, and “TimeHandoff”, which is the time the ball was handed to the running back. We can combine these two features to its own feature that shows the amount of time taken between the snap and the handoff. This reduces the number of features we would have to use. Lastly, we created a correlation matrix (Figure 2) to see the relationship between the features in each data point. Looking at the correlation matrix, we can see that no features have any significant correlation to the play's yardage gain (by themselves at least).

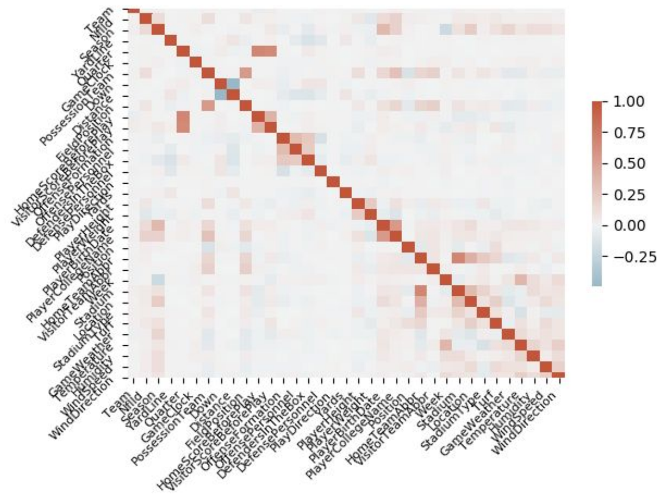

Figure 2 | Correlation matrix of each feature

Once we had determined our updated feature set, we began cleaning the data. With our dataset, we had 22 data points per play, one per player. We decided to remove all of them but the running back (for simplicity's sake), as his location and biographical information were the most important. Each data point per play also had all the duplicate data about the overall play. Next, we took the columns with string values and replaced them with a unique integer for every string value. Lastly, we scaled/normalized the data, using scikit's StandardScaler, which removed the mean and scaled by unit variance. This was done to help GMM not get influenced by insanely large numbers.

Unsupervised Learning

After the data had been cleaned, we implemented PCA to find how many principal components the data had. The visualization for PCA is shown in Figure 3. When we ran PCA, we got that our data had 33 principal components that kept 99% variance. We then created a scree plot showing the proportion of variance for each principal component. The plot is shown in Figure 4. As you can see, the first 6 principal components have the highest proportion of variance, and then the graph starts to drop off. We would take these 6 principal components to be the most important in terms of affecting how many yards the play gained.

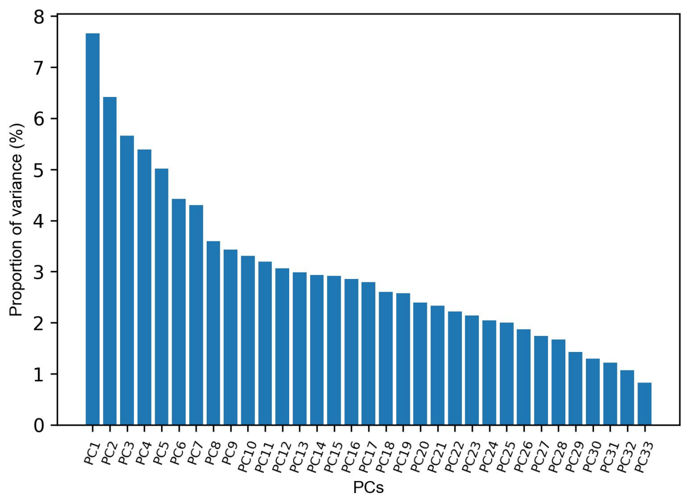

Figure 4 | Scree plot

We then wanted to implement GMM on the data. For GMM to determine the number of clusters we wanted to use, we used scikit to find the silhouette scores for different numbers of clusters using the features we found to be important. The silhouette scores show us the optimal number of clusters we should use for our dataset. The results of the silhouette scores are shown in Figure 5. From this, we saw that 3 clusters gave us the highest silhouette score by far. We then ran GMM with 3 clusters shown in Figure 6. As you can see the clustering did not work very well with this data set. This would make sense because if you go back to the heat map, there weren't too many features that had that strong of a correlation with other features. Therefore the data wasn't able to cluster very well.

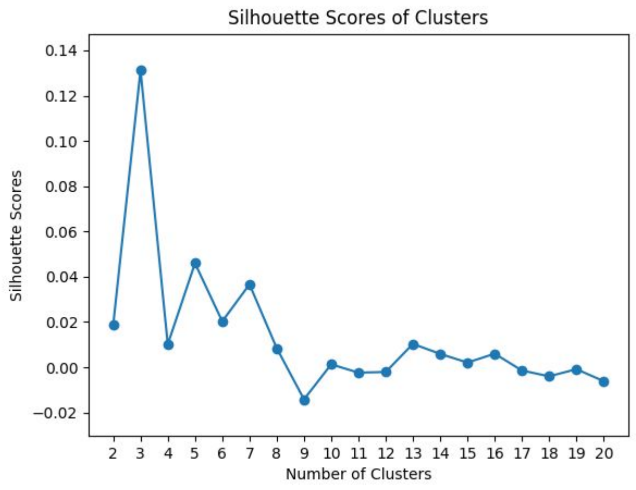

Figure 5 | Silhouette scores

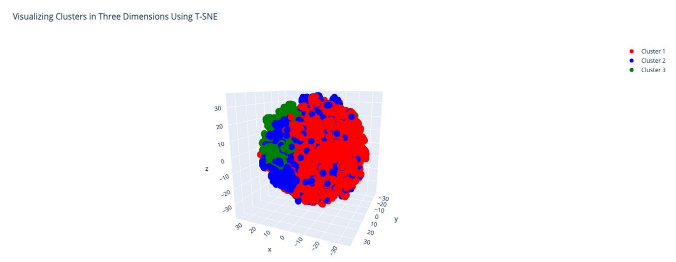

Figure 6 | GMM visualization

To summarize our work using unsupervised techniques, since this is a regression problem, and football plays are inherently similar and unpredictable, unsupervised learning did not work very well. GMM wasn't very helpful because the feature points were relatively similar, so the data did not cluster well.

Supervised Learning

When we first implemented unsupervised learning, we tried various methods such as linear regression, SVR regression, neural network, decision tree, and more. As we progressed, our numbers were not quite as successful as we had hoped. After the initial implementation using a linear regression model, we were predicting 18% of plays correctly within 1 yard. Doing SVR regression we were able to predict 26% of plays correctly within 1 yard. Obviously, these are not the results we had hoped to achieve. Since the model was not having much success with regression, we decided to turn this into a classification problem. Now instead of trying to predict an exact yardage, we changed our model to try and predict a range of yards. Now our results are categorized into ranges such as 0-2 yards, 3-5 yards, 5-7 yards, etc. Therefore, we will consider a prediction to be "correct" if the actual play fell within the range predicted by the model.

We then tried implementing a neural network using Keras. Our first attempt consisted of 3 hidden layers, each with 128 neurons. The hidden layers used ReLU activation, while the output layer used SoftMax. The SoftMax output allowed us to treat this as a classification problem, where we are predicting the probability that a play will be within our defined yardage ranges. Because it is a classification problem, we focused on categorical cross-entropy as the loss function. The network was trained using adam, a Keras-provided improvement upon stochastic gradient descent. We also decided on a standard 80/20 testing/validation split and shuffled the data beforehand.

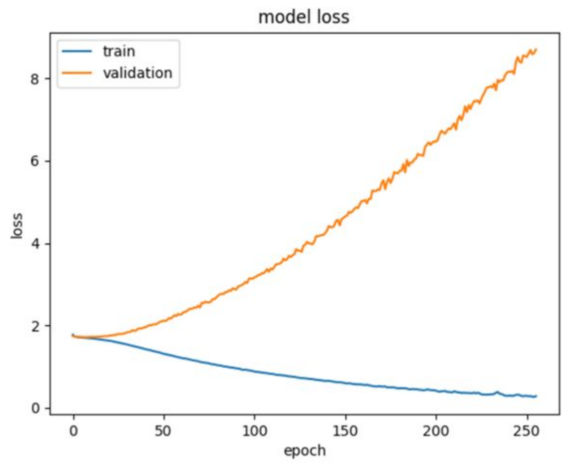

Figure 7 | Neural Net loss

The neural network very quickly fit the training data, but this model did not generalize well to the testing data. As shown above in Figure 7, the loss on the validation set quickly diverged from the training set and continued to grow. This is a classic example of overfitting, which we attempted to fix using a few different techniques. The results can be found in the Appendix. Some techniques we deployed included adding Keras-provided regularization terms of various kinds and strengths. Introducing a L2 regularization penalty of 0.001 was most successful at preventing overfitting, but this only seemed to delay the inevitable as the model continued to diverge just at a slower pace, as shown below in figure 8. Reducing the model complexity by removing hidden layers and lowering the number of neurons was somewhat effective as well. However, adding dropout layers, where Keras would remove a few features each layer, seemed to make the problem worse.

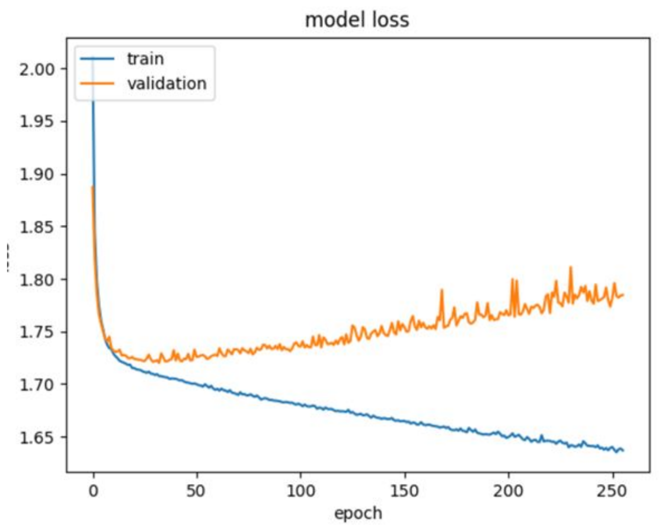

Figure 8 | Model with .001 regularization penalty

Another method that we implemented was the use of a decision tree. The goal of the decision tree was to try and turn this into a classification problem, so instead of trying to predict exact yardage, we wanted the decision tree to predict a range of how many yards were gained. Using some of the information from our feature selection portion, we deduced that the best layers to use at the beginning of the tree were yard line, distance, home team, and player weight. For reference, the yard line is the position of the ball on the field, and distance is the distance to a first down. This made sense to us since you are going to run different types of plays depending on where you are at on the field and how many yards you need to gain. For instance, if the ball is on the 3-yard line, you can only get 3 yards before you score a touchdown, so we shouldn't predict anything above 3 in that scenario. After implementing the decision tree, our results seemed pretty accurate. We had 50.72% of our predictions correct within 3 yards.

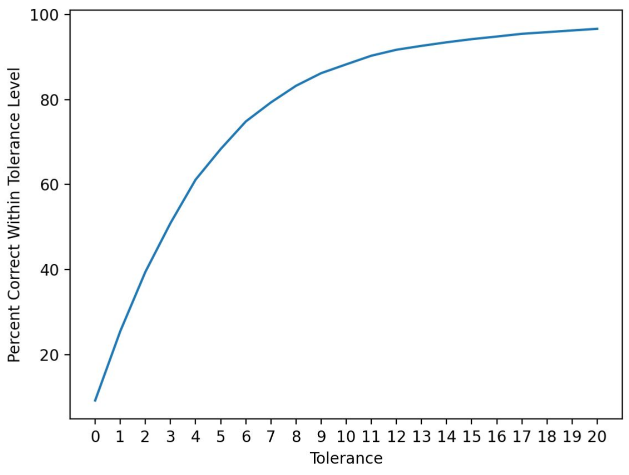

Figure 9 | Decision Tree Accuracy vs Tolerance

At first glance, we thought this was a good thing, and that we were on the right track. However, after doing the decision tree, we realized that the reason this model was doing so well was that so much of the data is centered around 3 or 4 yards. 56% of the plays in our dataset gained between 0 and 4 yards. This means that if we just guess 2 every time, 56% of our "guesses" would be within 2 yards. After realizing this, we looked at our decision tree and realized we were essentially guessing 2, 3, or 4 the whole time. This isn't much of a model, it's more just guesswork. After this realization, we were able to account for the fact that so much of our data was less than 5 yards.

Adding bins

After realizing that our labels were very clustered around a 3 or 4 yard gain, we chose to create bins to be more evenly distributed to accommodate for the possible "guessing" our models were doing. After making the bins, our overall accuracy went down; however, this decision was necessary to make the predictions more valid than simply guessing the most likely outcome.

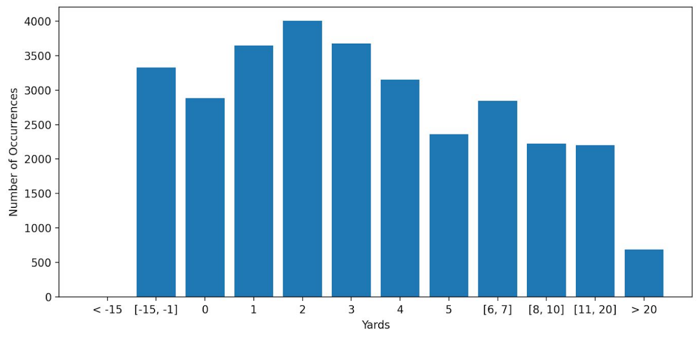

Figure 10 | Histogram of new bin distribution

Compared to the original distribution of labels (Figure 1), this was a much more evenly distributed dataset that we then used to retrain the decision tree. Since we switched to using bins instead of discrete integers, we could not efficiently use a tolerance. The following results will therefore contain zero tolerance which accounts for the lower correct prediction rate. The original decision tree with no bins scored around a 8.93% accuracy while the decision tree trained with bins scored around a 11.06% accuracy. This accuracy increase makes sense since we lowered the number of possible outcomes per play. We also retrained our random forest using the same bins which changed the accuracy from a 12.92% to a 13.43% using

100 estimators. Even though there is less room for error when using bins, the decision tree's ability to learn using the new data with bins made our model much more valid.

Conclusion

By doing different machine learning methods on our football dataset, we learned that football is overall very unpredictable. This did not come as a huge surprise to our team considering the nature of the game and the fact that the machine learning industry is not already very prevalent in the football community. One of our main goals was to make our models applicable to real time predicting by using less features and features that could be quickly calculated before a play begins. Even though doing this would result in a quicker prediction, the lack of thorough detail of each play may have also caused the models to lack higher accuracy. In the future if infinite computing power is achievable (or close to it), predicting using many more features may be feasible and a valuable way to predict a play. Until then, we are content with our learnings of the project and will watch for features that may further affect a play as we continue to watch football games.

Appendix

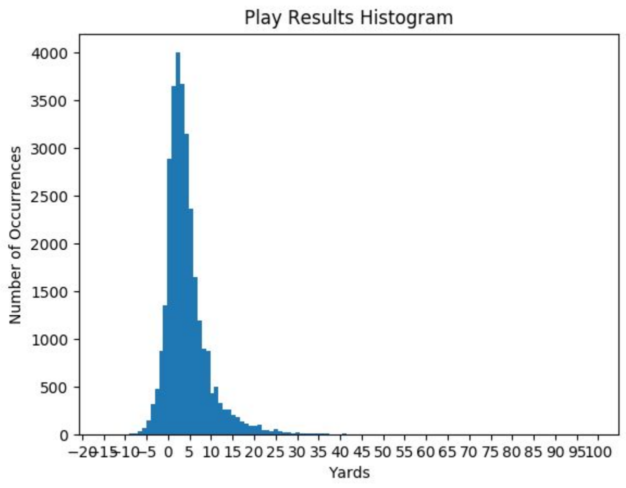

Figure 1 | Histogram consisting of each plays result

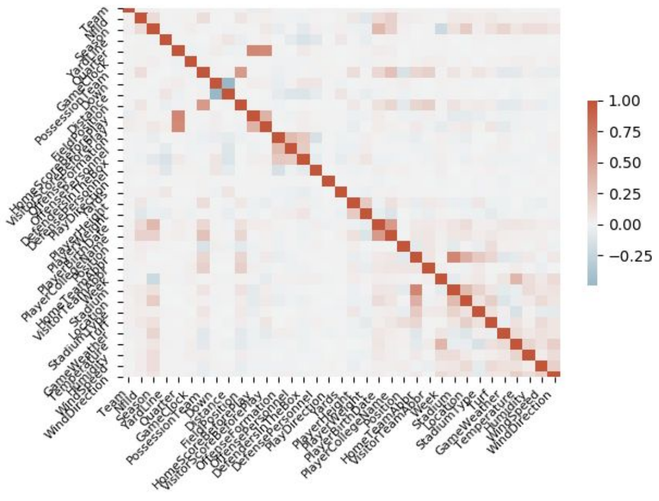

Figure 2 | Correlation matrix of each feature

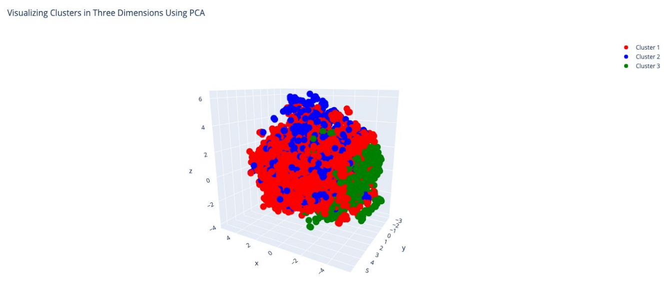

Figure 3 | PCA visualization of the dataset

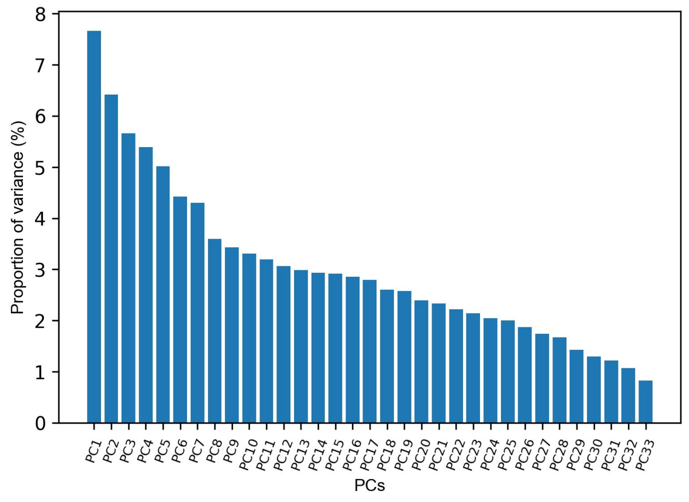

Figure 4 | Scree plot

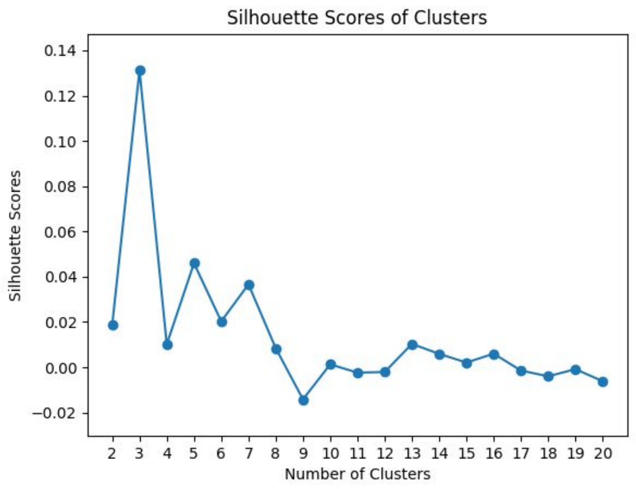

Figure 5 | Silhouette scores

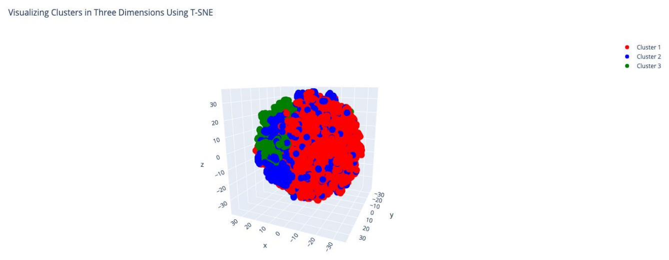

Figure 6 | GMM visualization

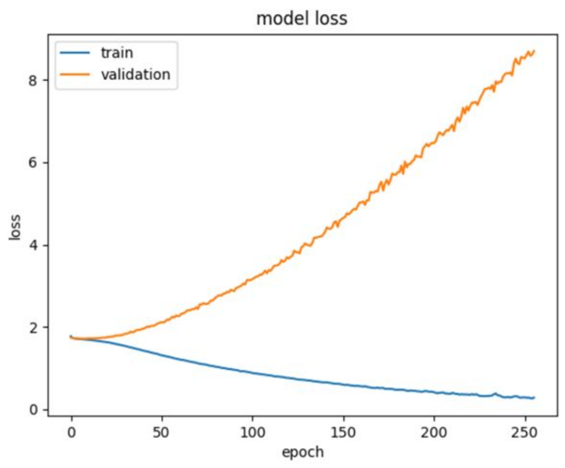

Figure 7 | Neural Net loss

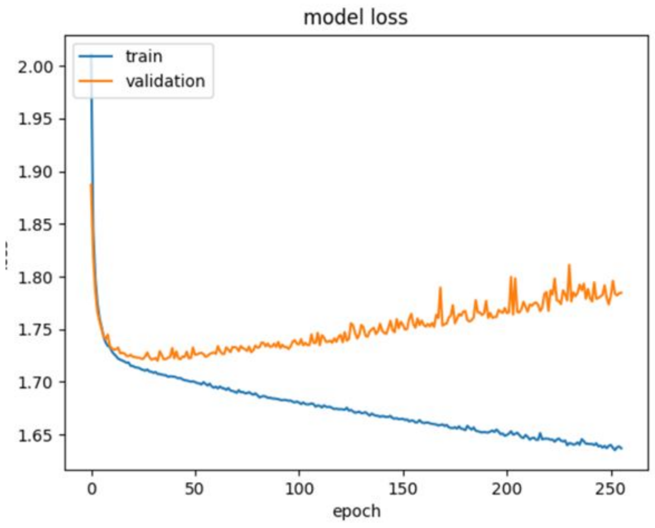

Figure 8 | Model with .001 regularization penalty

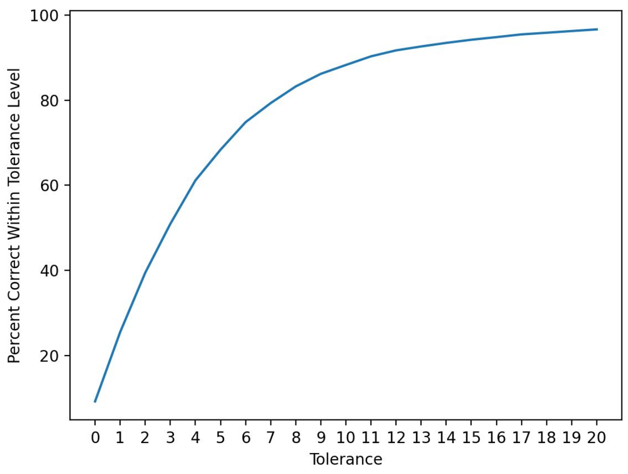

Figure 9 | Decision Tree Accuracy vs Tolerance

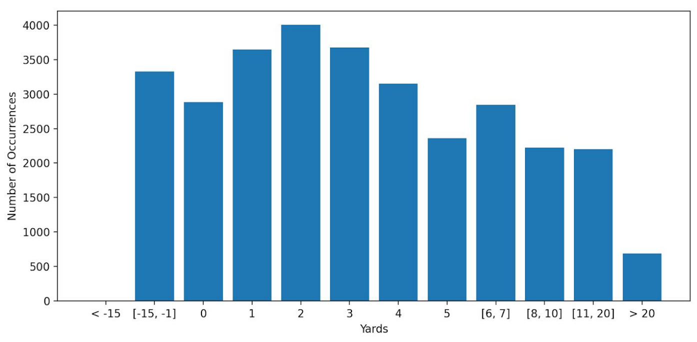

Figure 10 | Histogram of new bin distribution