Christopher Tams A17559278

Emmanuel Gutierrez A15769736

Raam Chaklashiya A16154997

Professor Mukamel

COGS109

NFL Positional Predicting Using Multi-Class KNN

Introduction:

i. The Real Problem:

In the National Football League (NFL), there are hundreds of players with many distinct characteristics, skills, and physical attributes. However, an issue that arises often is when players who seem extremely talented and have demonstrated great abilities on the field in the past or college, either fail to meet expectations or begin to decline in performance. This can be for a variety of reasons, but we believe that one of the main reasons that causes a player to decline in performance is due to their physical attributes limiting them in their position which led us to wonder if the best performing players in each position share physical characteristics. We want to create a predictive model that could assess whether a person is suited for a particular position in football based on their physical attributes. This could help players find the most optimal position for them, rather than waste their talent in a position where their physical attributes hold them back from their full potential.

We believe this is an important question that is worth looking into as teams invest millions of dollars into their teams and having players who are in the wrong position can cost millions of dollars, especially if the team is struggling to win games, as it may cause the team to lose sponsors, tickets sales, and merchandise sales. Additionally, players who aren't in their most

ideal position could also be at major risk of injury due to their physical characteristics not meeting the requirements of the rigorous work needed for their position. Injured players also can set teams back tremendously as the plays and game plans are built around their roster. Players change positions all the time and find more success after their switch. We hope this model can help determine if a switch can be beneficial, earlier on rather than wasting precious time in a league where one mistake can cause a team to lose important games.

ii. Data and Data Cleaning:

The dataset we will be using is an NFL statistics dataset from Kaggle (https://www.kaggle.com/datasets/kendallgillies/nflstatistics). It contains basic statistics, career statistics, and game logs for past and present players taken directly from the official NFL website. The career statistics and game logs have multiple spreadsheets categorized into specific subcategories such as passing and receiving as well as various positions like offensive and defensive line. The basic stats contains information for all 17,172 players in the form of 16 variables: age, birthplace, birthday, college, current status, current team, experience, height (inches), high school, high school location, name, number, player id, position, weight (lbs), and years played. The relevant variables we are most concerned with are age, height, and weight since they are the physical characteristics of the players and we will use them to classify their position.

In the NFL, there are many types of positions that are either offensive positions, defensive positions, or special team positions. Meaning that some players are skilled in scoring on the offense, defending against offensive players in order to prevent the other team from scoring, or also the more uniquely skilled players whose job is to kick the ball accurately and far. Our data includes 27 unique positions that all fall under one of the previous three types

mentioned. The data also reflects accurate proportional numbers to the number of players in each position in a league. For example, our data has 378 listed “Wide Receivers (WR)” and only 126 “Quarterbacks (QB)”, which is an approximate proportion compared to the players on a field during a typical offensive play where there are three WRS and only one QB.

The basic stats data is appropriate for our question since it gives us the physical characteristics of players to use as predictors for their position and the data itself is reliable since it comes directly from the official NFL website. The career statistics and game logs do not contain relevant data to our problem so we will not be using them.

To clean our data, we started by dropping all columns aside from age, height, weight, and position from the basic stats since these are the only variables we are concerned with. Next, we dropped the players with missing data for these variables so that our model doesn’t have to deal with missing data. Dropping players for missing information like their age or position, caused us to drop about 83% of our data, starting at 17,172 players/observations and only left with 2,959 players/observations. Finally, we renamed the “height (inches)” and “weight (lbs)” to simply be “height” and “weight” for convenience.

iii. Hypothesis:

We hypothesized that age, weight, and height will be high predictors for the position that is most optimal for a given player. The physical attributes of an individual player is important and could lead to different abilities of speed, power, and flexibility. Therefore we believe that age, weight, and height would be the biggest indicators of these physical abilities.

Model/Methods:

While discussing the type of model we needed to use for this goal, we emphasized that it had to be a classification model and not a regression since we are predicting a categorical label

(player position) rather than a continuous value. KNN seemed the most suitable for our purposes since we have only a few predictors. Another reason why KNN is most suitable is because we wish to categorize our players with those who are most similar, which is a main goal for a KNN model. Additionally, KNN is capable of multi-class classification, which is appropriate for our problem due to the large number of positions in our dataset that we're trying to predict. The main goal is to see if there is a relationship between the physical attributes like height, weight, and age, and the position a player should take. Therefore this model will help us infer whether or not these parameters are related to position. We will start by performing a traditional train/test split with a training set size of 80% of the data and test set size of 20% of the data and fit a basic KNN model using 3 neighbors to get a benchmark performance score to try and beat. Then, we will use cross validation to find the optimal value of k and use that value in our final model. To maintain consistency and reproducibility for our model, we used a specified random_state of 1 for our train/test split.

Methods - Cross-Validation:

To find the most optimal k-value, we used grid search cross-validation to test different values of k to see which gives the highest mean test accuracy. Sklearn's GridSearchCV function takes in an input of a classifier, a parameter grid, and an integer to specify the number of folds for the cross-validation, which is 5-fold by default and results in a 80/20 training and test/validation split. In our case, the classifier is a KNeighborsClassifer which is an implementation of the K-nearest neighbors algorithm. The main parameter in a KNN model is the number of neighbors used to determine a data point's class. It takes a majority vote over the K neighbors and assigns the data point that class. This is a hyperparameter and there is typically no intuitive way of determining the optimal number of neighbors to use, so we use grid search

with the number of neighbors as a parameter in the parameter grid and the grid search will exhaustively test every value of K that we specify in order to determine which value results in the highest average accuracy across all the cross-validation folds.

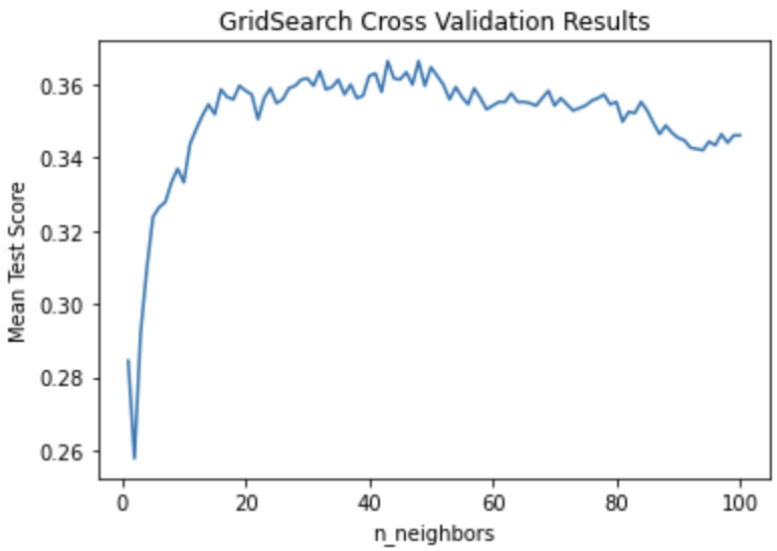
The K-fold cross-validation that grid search uses works by splitting the data into k consecutive folds of training and testing splits such that each fold is used exactly once as the test set while the other k-1 folds are used for training. We don't shuffle the order of the data before cross-validating since the data is already unordered due to dropping a lot of players and the fact that the original data was unordered to begin with. This also is beneficial because it allows us to reproduce the same test values for a given number of neighbors so we get the same results each time we run our code and can compare models more easily.

Results:

i. Model Selection:

Our base model using a 80/20 train/test split resulted in an accuracy score of 30.9% which seemed very low at first but keeping in mind that this is a multi-class classification task with 27 possible positions to predict, the lower accuracy makes sense for an unoptimized model.

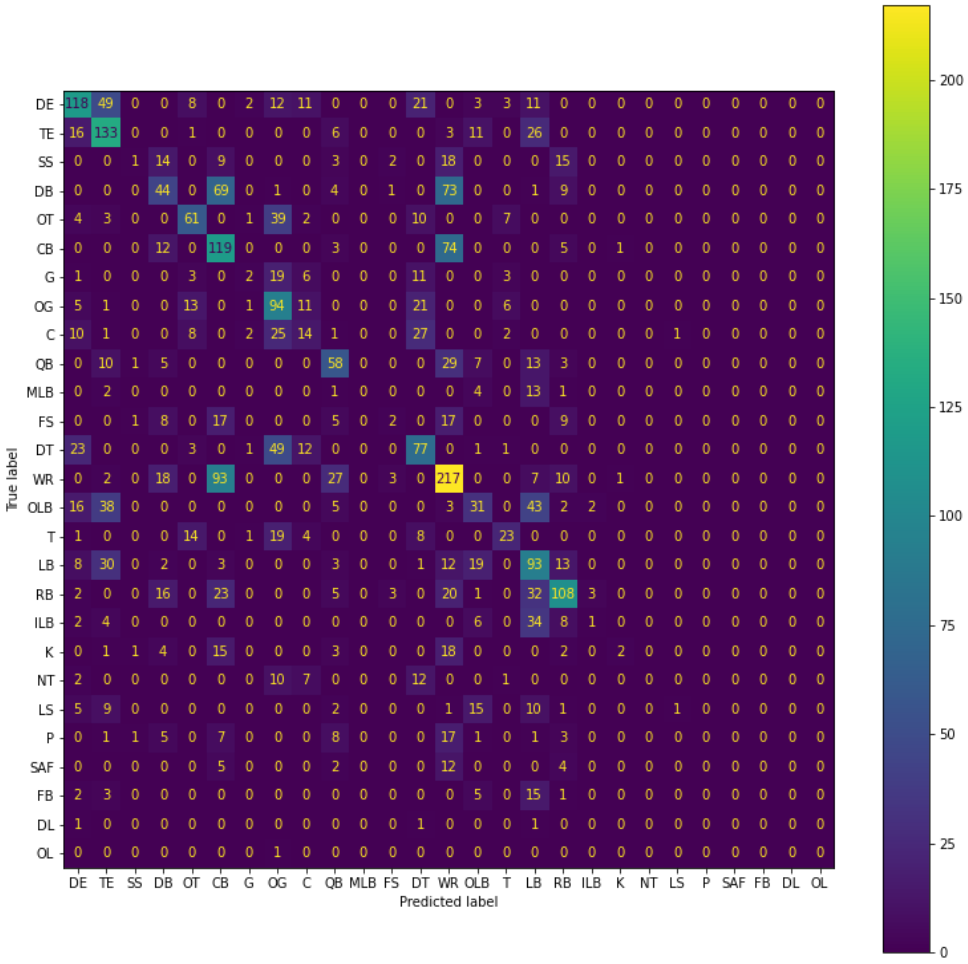
Next, to select the optimal model, we used gridsearch cross-validation to test the performance of k-values 1 through 100. We compared the mean test accuracy across the 5 folds of cross-validation for each model and the validation curve of mean test accuracy of each value of n_neighbors is shown in the plot below.



From this curve, we can see that n_neighbors of 1-10 clearly have lower average performance than the others. The curve is relatively stable until around k=50, where it starts to follow a general downward trend. The k-value with the maximum average test score ended up being k=48 at 36.6344% accuracy. However, we noticed that there is another peak on the graph to the left of k=48 with a very similar mean test score and decided to take a look at it in more detail. It ended up being k=43 with an average accuracy of 36.6342% which is only 0.0002% less than the k=48 model yet uses 5 less neighbors which is important to mention because despite having a (negligibly) lower average accuracy than the k=48 model on the 5-fold gridsearch cross-validation, we decided to go with 43 as our value of k for a number of reasons. First, the average accuracies on the cross-validation are basically identical yet k=43 is a simpler model as a whole, using 5 less neighbors to make a prediction. By the parsimony principle, simpler models are usually favored when the accuracy is comparably similar since they help avoid overfitting. Neither of these models showed evidence of overfitting on the cross-validation since the mean accuracies for both models were low.

ii. Model Estimation:

After selecting our model, we tested its performance on the entire dataset to obtain the final, optimal accuracy of a KNN model on our dataset. Using 43 neighbors, the model achieves an accuracy of 40.5% for its predictions, which is actually lower than the accuracy achieved by the k=3 base model of 52.6% but we believe the lower k-values to be overfitting as evidenced by their much lower performance on the cross-validation curve which indicates that they don't generalize as well to new data. We created a confusion matrix to display the various predictions our model made compared to the ground truth labels of the data. The diagonal from top-left to bottom-right represents the predictions our model got correct while the off-diagonals are the misclassifications.



As seen in the confusion matrix above, we are able to see how many misclassifications there were between the predicted and true labels. Some of these do make sense however as their physical characteristics are extremely similar. For example, 73 Corner Backs (CB) were misclassified as Wide Receivers and 93 WRs were misclassified as CBs. A reason for the large misclassification, based on current player observations, is that both positions share similar characteristics like height and weight as those lead to them being quicker and more agile on the field, which is a necessity for these positions to find success. Similarly, another major misclassification that we had in our model was 69 Defensive Backs (DB) being falsely labeled as CBs. Again, this makes sense as both are secondary defensive positions which require speed and agility that derives from height and weight.

The issue of similar positions being misclassified as each other is something we would like to improve on in future work for this project since we didn't prepare for this problem. We hope to be able to find some new feature or characteristic that would allow us to make clearer distinctions between positions whose players share similar physical attributes in order to increase accuracy.

Conclusions and Discussion:

After finding the low accuracy results of 36.6%, we can conclude that age, weight, and height alone may not be the best predictors of position, despite what we hypothesized. This could be because there is no/little relationship between physical attributes and positions but it could also mean that there are more physical attributes we didn't take into consideration. Wingspan, stamina, strength, speed, and hand size may all play a role in relation to a football player's most optimal position. If more free data was available on this information, it would be interesting to apply it to this model.

It also may be possible that more than just physical structure has relation to a football player's most optimal position. When we made experience in the NFL a parameter, it increased the accuracy score by 0.5%. This would lead us to believe that other factors such as on field percentage (how long the player is on the field per game) might have a relationship with a football player's most optimal position as well.

Code: https://github.com/e1gutier/COGS-109-Final-Project