# Introduction

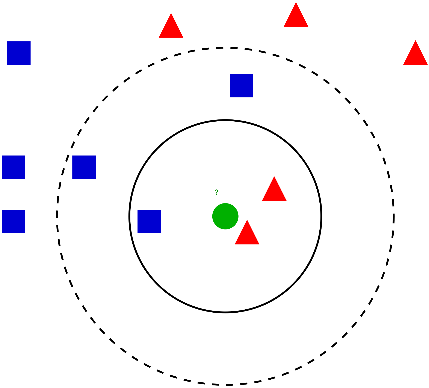
In this paper, we will discuss about K-nearest neighbors’ (KNN) algorithm and its working. We will use KNN regressor model to predict the target variable of NFL dataset. We will also evaluate independent variable of the data set to select important feature with the highest positive or negative impact on target variable.

## k-Nearest Neighbors

k-Nearest Neighbors, also known as KNN is one of the simplest machine learning classification/regression algorithms. KNN is a supervised Machine learning (ML) techniques, it uses the labels or the data point in training set to train a model which can be further used to predict label on unlabeled new data or use regression model to predict the value of target variable.

KNN regressor works by calculating the average of the numerical target of the k nearest neighbors. KNN model simply remember the position of each data instance from training set in a n-dimensional space where n is the number of feature or independent variable used to train the model. When predicting target variable for new data set, it plots the coordinate of each datapoint in n-dimensional space and evaluates the dependent variable of nearest neighbors. It uses different distance functions such as Euclidean, Manhattan and Minkowski distance to calculate the distance between datapoints.

The k in the KNN model is the number of neighbors to be considered to evaluate the result. If the value of k is 1, Knn regressor will only consider single neighbor nearest to the evaluation case. Similarly, when is more than 1, let say 10, it will search for 10 neighbors which has least distance away from test case. It further evaluates for the bias or majority among selected 10 neighbors to calculate the result of the test case.



We can use above image the explain the working of KNN algorithm. In above image, the inner solid circle represents the evaluation when K=4, we are looking into 4 nearest neighbors of our test case. In this case, we have 1 blue square, one green circle and 2 red triangles, based on this observation, the algorithm decides that the test case should be labelled red triangle based on majority.

# Dataset and data dictionary

Given data set is Football record for NFL running plays. In this data set different records for each instance is given along with yard gained by the player. We will consider yard\_gained, the distance the running player was able to gain or lose as target variable. We will be using feature variable along with target variable to train a knn model and evaluate how effected the model will be able to predict the yards gained by player in the given test case.

If the model can predict the distance/yards gained by player accurately, we can evaluate the correlation between the features and target variable. Based on the importance of feature, ideally, we should be able to reverse implement the same in the game to improve the play of running player to gain more yards.

By the end of this project, we will train KNN model, test the model accuracy, evaluate the correlation between each feature and yards gain and will attempt to draft out suggestion to football coach for next game.

We will use given data dictionary to understand meaning of each feature column in the dataset.

|  |  |
| --- | --- |
| Column | Meaning |
| play\_id | ID (Not Meaningful) |
| PlayOfGame | Number of Plays In The Game Up Until That Point |
| yardline\_100 | What Yard Is the Play Starting On |
| game\_seconds\_remaining | How Long Is Left in The Game |
| drive | How Many Times Has the Team Had The Ball |
| qtr | What Quarter Is It |
| down | Which Down Is It (You Have 4 Downs/Plays) |
| ydstogo | How Many Yards to Get First Down |
| yards\_gained | How Many Yards Were Gained |
| shotgun | Type of Play |
| no\_huddle | Type of Play |
| qb\_dropback | Type of Play |
| Run To Left | Runner went left |
| Run To Right | Runner went right |
| Run Gap Center | There was a gap in the center |
| Run Gap End | There was a gap on the end |
| Run Gap Middle Right | There was a gap near the end |
| score\_differential\_post | How Many Points Behind Is the Team With The Ball |

There are three type of play, shotgun, no\_huddle and qb\_dropback which indicated as binary, shotgun has value means team is playing shotgun offense type.

Similarly, run to left and Run to Right are binary indicator indicating which direction player chose to run. Three gaps, Run Gap Center, Run Gap End and Run gap middle Right is also indicated by binary number.

# Analysis and Implementation

We will use python as data processing language as we will be multiple package and function from sklearn library.

We will start by loading all the libraries we will be using in this project.



## Data cleaning and preparation

This dataset is fairly clean, and all the variable are numerical. However, we still have few modifications to do before we can start using on KNN model. We will start by removing “play\_id” column which will not have any relevance in the model, if used can be misleading.

data.drop('play\_id', axis=1, inplace=True)

Now we will check for any mission values. With isnull() function we are able to find out that there are only 11 missing value in the dataset, all the value missing are from “down” column. If significantly high number of rows were missing values, we would have used different techniques to fill in missing value, however 11 rows is insignificant compared to total 5298 rows of data. We will drop 11 rows and now will be working with 5287 rows of data.

1. data.dropna( axis = 0, inplace = True)
2. data.isnull().values.sum()

Dataset loaded in object “data” includes both independent and dependent variable, we will now divide target variable and feature variable into two separate data frame which we will use in our model.

1. X = data.drop(columns='yards\_gained')
2. y = data.yards\_gained

## Splitting Dataset

We will now split data into training and testing set in 7:3 ratio. We are splitting data into two part before we want to make sure our model is not overfitted by revaluating it with data with we have not used to train it.

1. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state= 21)

**Underfitting and Overfitting**

Underfitting occurs when there is not enough data to train a model to identify the important of each underlying feature, which is important to make correct prediction. To avoid underfitting we should make sure we have fed enough data while training mode. Since we have a big dataset, 70% is good enough to train model.

Overfitting occurs when model fit to wells with the dataset used to train the model. This could result bad prediction when new dataset or unseen data is pushed into model to make prediction. To make sure our model is not overfitted, we will test with testing set which is not used while training model.

Now we fit a model with KNeighborsRegressor from sklearn library, we will use for loops to observe performance of model at different k value. Since, we are working with regression problem, Root Mean square Error (RMSE) will be the best metric to evaluate our model. RMSE measure the average error between predicted value of target variable and actual value, lower the RMSE , better the model.



Output:

|  |
| --- |
| RMSE of Model is : 0.290 when value of k is 1  RMSE of Model is : 4.409 when value of k is 2  RMSE of Model is : 5.731 when value of k is 5  RMSE of Model is : 6.143 when value of k is 10 |

Above prediction is close to perfect when K is 1 (single neighbor), however above prediction was on training set and it is expected to have high accuracy. To make sure good result above is not due to overfitting, we will now test the model with testing set.



Output:

|  |
| --- |
| RMSE of Model is : 9.572 when value of k is 1  RMSE of Model is : 7.858 when value of k is 2  RMSE of Model is : 7.085 when value of k is 5  RMSE of Model is : 6.674 when value of k is 10  RMSE of Model is : 6.461 when value of k is 20 |

From above RMSE output of model at different K-value, we can say that model is not overfitted nor underfitted. The RMSE value at K=10 is almost similar for both testing and training set. For testing set, RMSE seems to be decreasing with the increment in K value, however we can see that there isn’t much improvement when k is 10 or 20. By this we can assume, optimal value of k is 10.

## Cross Validation Prediction

K-fold cross validation is more advance way of fitting the model to avoid the problem of overfitting and underfitting. We will use sklearn.model\_selection.**cross\_val\_predict,** which will split the data according to cross validation (cv) parameter, also called k-fold. The data will be split into k folds, it uses one-fold for testing and the rest as training for k number of runs, the performance of model is evaluating as average over the k folds.



|  |
| --- |
| RMSE of Model is : 9.022 when value of k is 1  RMSE of Model is : 9.022 when value of k is 1  RMSE of Model is : 7.021 when value of k is 5  RMSE of Model is : 6.736 when value of k is 10 |

Both manual test/train model and cross validation model have similar accuracy, this indicate this model can predict yards gained with high accuracy with error as small as 6.736.

## Feature selection

In the above model, we are working with all 16 of feature variable. All the feature may not be important to target variable which in turn may act like noise to and reduce the accuracy of the model.

We will evaluate correlation between target variable and feature variable by using corr() function, we can visualize correlation matrix in boxplot but since of dataset have 16 column, boxplot looks overwhelming and its difficult to get insights from it. We evaluate feature importance by using correlation coefficient matrix.

1. abs(data.corr()["yards\_gained"])

Output:

|  |
| --- |
| PlayOfGame 0.006797  yardline\_100 0.109934  game\_seconds\_remaining 0.002239  drive 0.004059  qtr 0.001386  down 0.017062  ydstogo 0.084169  yards\_gained 1.000000  shotgun 0.076844  no\_huddle 0.008392  qb\_dropback 0.106822  Run To Left 0.036077  Run To Right 0.012007  Run Gap Center 0.025956  Run Gap End 0.061783  Run Gap Middle Right 0.040231  score\_differential\_post 0.015041  Name: yards\_gained, dtype: float64 |

From the above absolute value of correlation coefficient, we can select feature with highest correlation with target variable “yards\_gained”. The selected features are ydstogo, yardline\_100, shotgun, qb\_dropback, Run Gap End.

We now filter out selected feature to a new data frame.

1. XF = X[['ydstogo','yardline\_100','shotgun','qb\_dropback', 'Run Gap End']]

Now, we will again split the new selected feature variable into test and train set with same random state as before to reproduce the same split.

1. X\_train, X\_test, y\_train, y\_test = train\_test\_split(XF, y, test\_size=0.3, random\_state= 21)

Now, we will train model with only selected 5 feature variables and evaluate the model using testing set.



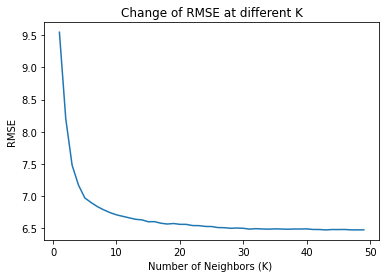
Output:

|  |
| --- |
| RMSE of Model is : 9.791 when value of k is 1  RMSE of Model is : 8.031 when value of k is 2  RMSE of Model is : 6.878 when value of k is 5  RMSE of Model is : 6.576 when value of k is 10 |

There is not much change in RMSE value even when we used only 5 features, however we can notice a small improvement from 6.674 to 6.576 at K=10. This could be because most of the function in sklearn are programmed to reject variable with very less impact on target variable automatically.

In the given dataset, the range of Yards gained is 103 units, prediction of out model only has Root mean square error of only 6.576 units, which indicate model is making good predictions.

**Error at different K value**

Now we will visualize change in RMSE at different value to conform if 10 is the optimum value of K.

We can see the gradual decrease in error which the increase in the value of R, this indicate we are making better prediction when we are considering a greater number of neighbors.

However, looking clearly at the RMSE values, we see there hasn’t been much change when we move higher than k at 10. Based on this diagram, we can consider the best value of K for this model is 10.

# Conclusion

A gain in football is when a player forwards the ball closer to the area that they are attempting to score on. The team has two ways of gaining yards, either by throwing or by running the ball. In this project we evaluated multiple identifiers of game, such as down type, type of play, player ran on which gap, what yard is the play starting on and more.

By the end of the project, we were able to identify which feature were beneficial to gain yards and which works opposite. Based on mathematical relation between different feature and target variable, we will be draw out suggestion for football coach.

With respect to game strategy, quarterback drop back seems to be quite favorable to gain yards followed by shotgun formation as offensive attack. No huddle offense style has very small impact on yards gain, in fact QB drop back is 12 type more effective than No huddle, and shotgun is 9 times more effective than no huddle. So, we respect to formation of game, team shot prefer QB drop back and shotgun.

Runner seems to gain more yards when they run left, in fact run to right is found to be counterproductive. Coach should advice to player to keep run to the left as preferences over the other side.

Based on the data we evaluated in this project, running to gap in center and to middle right is counter productive to gain yard relative to running to gap end. This is not very surprising, as player would have high success rate if the scoring end has gap. However, player should be advised that if there are multiple gaps opening, players should always go for end gap.

Other discoveries of the evaluation are more general and seems intuitive, such as player seems to have more chances of yard gain around the starting half (quarter 1 or 2) of the game compared to ending half (quarter 3 or 4). In the starting plays of the game, player will certainly be more energetic and have good stamina to gain more yard than later when players are tired and worn out.

Similarly, it is clear the player in down 1 or 2 will easy gain yard compared to down 3 or 4 due to opponent’s defense. Also, runner starting at higher yard gains more yard than player starting at smaller yard. Player should try to pass wide receiver who is closer to the scoring post.

# References

Kotu V, Deshpande B(2018), Data Science, 2nd Edition, ISBN: 9780128147627

Brownless. J (2020), How to Calculate Feature Importance With Python, Retrieved from <https://machinelearningmastery.com/calculate-feature-importance-with-python/>

Saedsayad.com (n.d), K Nearest Neighbors – Regression, Retrieved from <https://www.saedsayad.com/k_nearest_neighbors_reg.htm>

Scikit-Learn (n.d), Sklearn.Neighboars: Nearest Neighbors, Retrieved from <https://scikit-learn.org/stable/modules/classes.html#module-sklearn.neighbors>

Rookie Road(n.d), American Football, Retrieved from <https://www.rookieroad.com/football/>