Report for Project 2
Team Members: Frank Xu, Madeline Lee, Mohammad Ali Yektaie, Tucker Moore

Part 1: Basic Statistical Analysis and data cleaning insights

The Exploratory Analysis is designed for researchers to gain insight into the distributions of the variables in order to determine what acceptable ranges are, if outliers exist, or if some attributes are redundant or derived.

Mean/Mode/Median and Standard deviation of at least 10 attributes

- Kickoff barometer
  - The mean barometer reading is 29.737 inches. Barometers measure pressure, and the average barometer reading is around 30 inches (in general). Thus, the mean and std deviation follow the standard pattern (within a reasonable range).
- Kickoff humidity
  - Kickoff humidity measured in percentage, has a mean of 60.169 and a standard deviation of 20.09. We expect the standard deviation to be large because the season and location can be extremely variable between teams.
- Fantasy points per game
  - The mean fantasy points per game (actual recorded) is 4.0577 with a standard deviation of 4.99. This indicates that predicted fantasy points are highly variable between players for any given game.
- Age
  - The mean age (measured in years) of football players across the past 5 years (starting from 2013) is 26.52. The standard deviation is 3.56, indicating that around 68% of the players in the NFL are in their twenties. The max age is 45, which is interesting because the 75th percentile is only 28, indicating that it is a significant outlier.
- For the next attributes, it makes the most sense to split the data by position and then analyze, because not every position will typically have any data for these attributes. Including all positions in analysis of these attributes could potentially cause skewed data. Thus, we will analyze both holistically and with respect to position.

- Rush yards (historical)
  - Mean is 7.95 and the standard deviation is 21.62. This indicates that the data is highly variable and probably skewed right by players that happen to rush a lot of yards in any game. It is important to note that the 75th quartile was at 1 yard, indicating that for all types of players, it is fairly uncommon for them to gain any rushing yards during a game. However, when we break this down by position, it is evident that holistic data is

impacted by running back data. Running backs (RB) are the only offensive position whose specific job is to gain rush yards. Thus, when we analyze RB data, we can see that while the mean is 24.72 yards and the standard deviation is 34 yards. The RB 75th quartile is 39 yards. However, none of the other positions have a 75th quartile above 16 yards. Thus, the RB data probably accounts for the right skew in the data for historical rush yards.

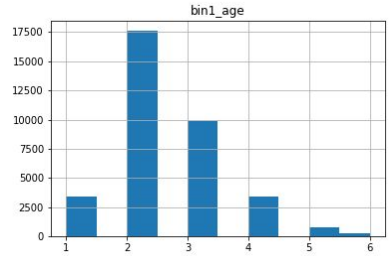
- Receiving yards (historical)
  - The mean is 18.06 yards and the standard deviation is 29.38, while the 50th percentile is 1 yard. This indicates that if we are looking at any arbitrary player, the receiving yards can vary widely. When breaking down by position, we still found high amounts of variability. For example, for tight ends (TE), we found that the mean receiving yards was 17.84 yards and the standard deviation was 26.08, while the 50th percentile was 6 yards per game. Another example are wide receivers, where the mean receiving yards was 31.65 yards and the standard deviation was 37.34. These two positions, which comprise over half of the observations in the dataset, could affect the overall variability of all observations of receiving yards.
- Receiving yards (predicted)
  - The mean is 15.622 and the standard deviation is 23.27.

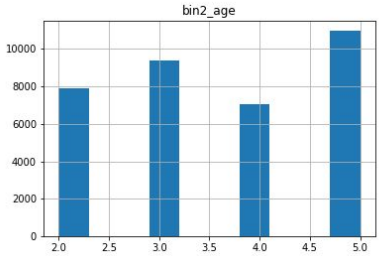
Rec (historical)

  - The mean number of receptions in a game is 1.57 receptions and the standard deviation is 2.19. This indicates that for any arbitrary player, they will make about 1.57 receptions in any game. When breaking the number of receptions down by position, we can see that ⅗ of the positions have a mean of at least 1. However, QB position and kicker position have means of 0.04 and 0, respectively.
- Pass touchdowns (historical)
  - The mean number of pass touchdowns (touchdowns made by QB throwing to another player) was 0.11 and the standard deviation was 0.54. When we break it down by position, we can see that the QB has the highest mean because they are almost always in possession of the ball at the beginning of the play and in the capacity to throw. The mean for QB pass yards is 1.26 and the standard deviation is 1.19 - this probably accounts for the increase in the mean and standard deviation of the holistic data.
- Pass yards (historical)
  - The mean is 18.19 yards and the standard deviation is 67.05. This indicates high variability for any arbitrary player. We can also see this by examining the min and max, which are -3 and 522, respectively. When examining by position, we can see that the QB position has the highest mean for passing yards at 205 and a standard deviation of 110. Its first,

second and third quartiles are 147, 223, and 232, respectively. These are significantly higher than the overall stats, and the other 4 positions have means less than 1 pass yard.

- This part want us to make sure the assumptions of p1 for cleaning our data were valid
    - Outlier detection
        - In order to detect potential outliers, we used boxplots on our data. (see example below in histograms)
        - Method to detect outliers and why we decided to keep or remove them: We decided to keep our outliers because it was possible that someone could have a great game or run a play that had never been run before.
        - It was important to note that fantasy points, rushing and receiving yards, etc could be negative values, but those were considered outliers instead of invalid data because it is possible (ie. if a player somehow was pushed backwards or ran backwards on a play during the game). Thus, we did not remove them.
    - Strategy to handle missing values
        - When examining missing values, we determined that they were true zeroes. For example, a QB would rarely have any receiving yards, because they are generally throwing the ball. Thus, when entering the data, it seems that individuals merely left them blank instead of filling in zeroes. We changed all NaN values to zeros to reflect the true zeroes.
    - Do we need a different data cleaning from the one we did in p1
- Choose one attribute for binning and … (Refer to the original project description)
    - We chose to bin age. We binned using equi-width(bin1) and equi-depth (bin2) techniques and found that if we bin using equi-width of 4, then the binning histogram represented a fairly normal distribution (a little right-skewed). This follows the histogram for age while also reducing the number of classifications for age.





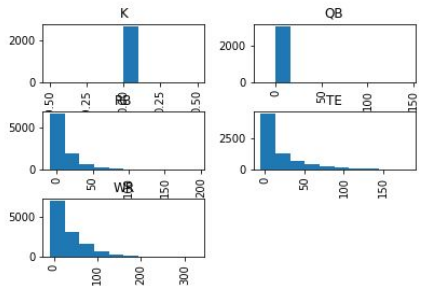
LOF outlier detection

We ran the LOF algorithm on our data. The results were not as we would have expected. The LOF discussed in class had only a K parameter indicating the number of neighbors to consider when calculating the neighborhood density. It then uses the distance of the K neighbors to

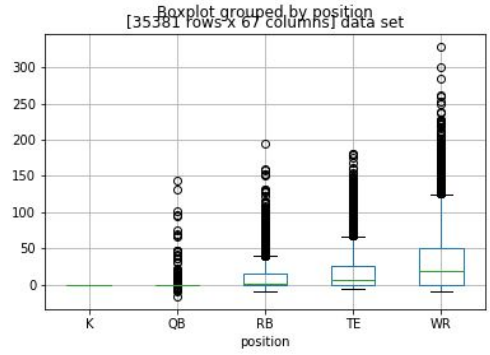
assign a score of how outliery a data point is. The LOF algorithm in Scikit package uses a different approach to define outliers. In addition to K, it requires another parameter ‘contamination’ which specify the ratio of outliers you want to have. For example, if you choose the contamination as 0.01, it will select 1% of your dataset that is the furthest from other points as outliers[1]. By doing so, the K value is basically ignored.

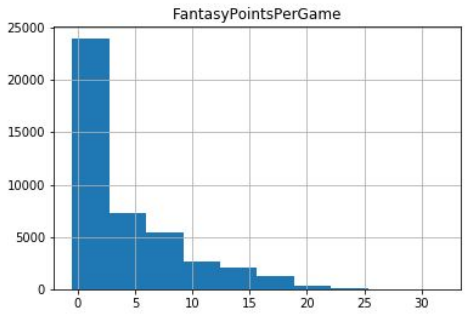
Part 2: Histograms and Correlations

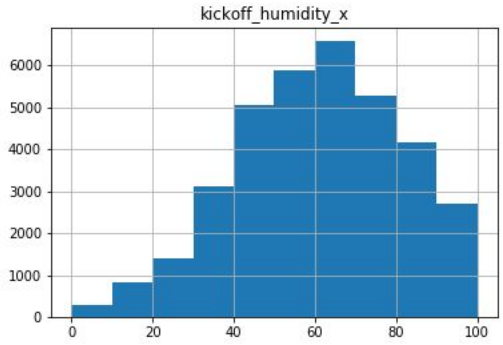
Rec_yds histogram by position

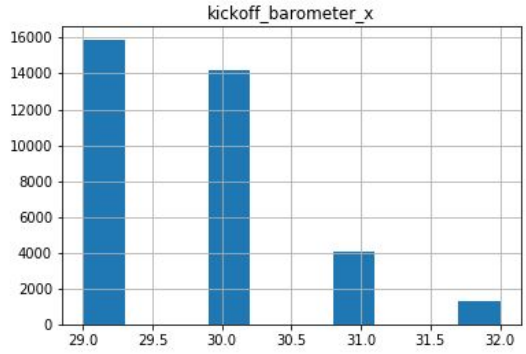


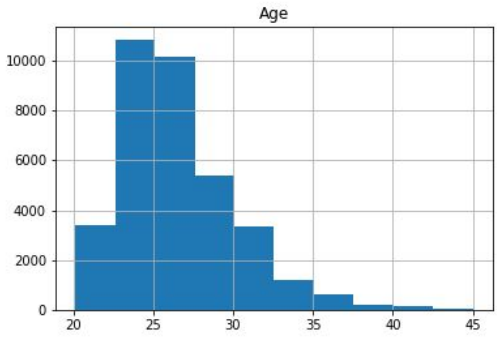
rec_yds by position boxplot











[1] http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html

Kickoff humidity shows a fairly normal distribution, although a little left skewed. The range is between 0 and 100, which makes sense as humidity is measured as a percentage. It appears that the mean lies somewhere between 40 and 60 (accounting for the left skew).

Kickoff barometer shows a left skewed, non-normal distribution. In addition, the precision of the measurements were only in inches (not half-inches, etc), which accounts for the break in the frequency bars for the data.

Age distribution shows a right skew. It appears that the algorithm naturally binned at around 2.5 years per bin (equi-width). The highest frequency of age for players was between 22.5 to 25 years old. It is important to note that although youth might be an advantage when playing pro football, usually, players will play for a university first, which may explain the high influx of players after age 22.5 (after college) instead of right at 20.

Fantasy Points per game is also right skewed. However, this makes sense as the data has a soft limit at zero (it is possible to get negative points but is rare), but no upper limit. Fantasy points are also based on the amount of data per person per game, which indicates that many players, which do not play that many minutes in a game, would have lower fantasy point totals.

Correlations

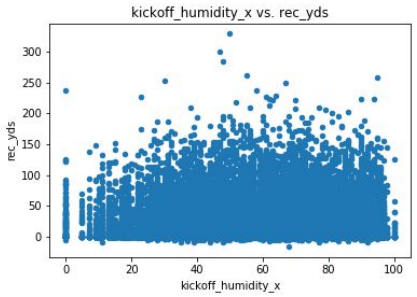
x/y	age	rush_yds	rec_yds	humidity
age	1.0	-0.07	-0.033	-0.007
rush_yds		1.0	-0.023	-0.07
rec_yds			1.0	-0.006
humidity				1.0

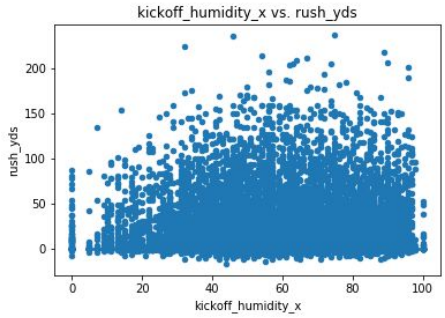
This correlation output indicates that there are very weak correlations between all of these variables. A weak relationship indicates that there is probably no linear relationship. Looking at the scatterplots for these relationship confirms that there does not appear to be a linear relationship between any of the variables sampled here.

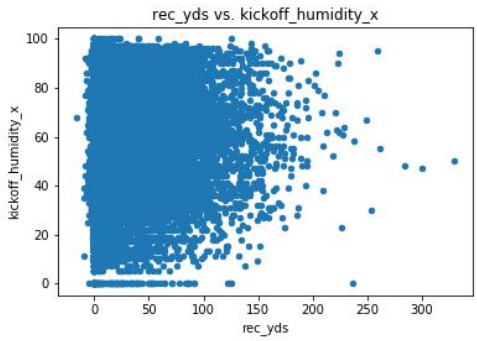
We can group a correlation by position and analyze it further. For example. Grouping receiving yards by position could change the correlations. In the table below, we can see that although still weak, some correlations are now positive instead of negative. Below are the scatterplots.

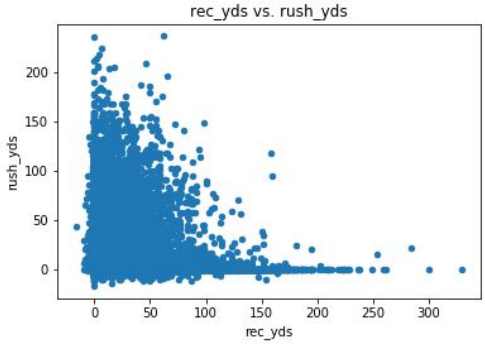
	WR	QB	K	TE	RB
Age vs. receiving yards	0.081	-0.029	NaN	0.139	0.022

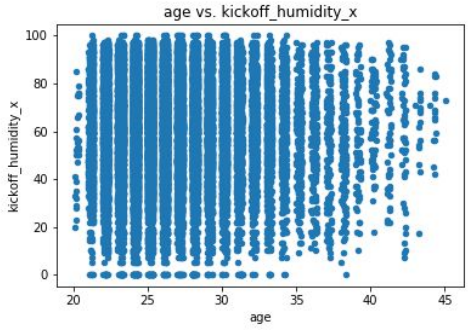
Scatterplots

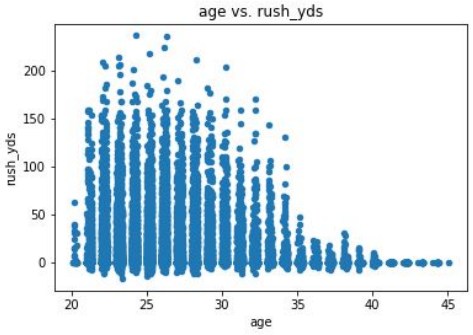


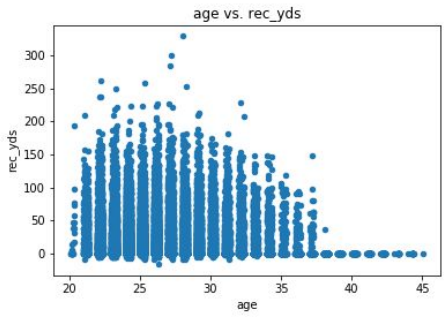




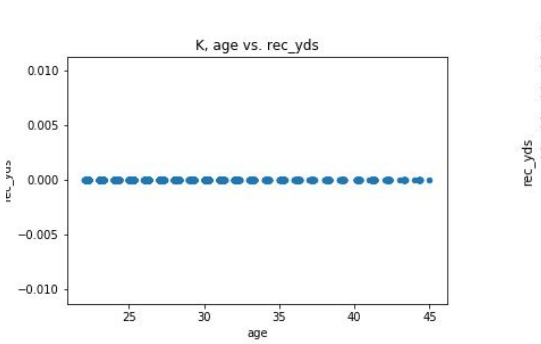


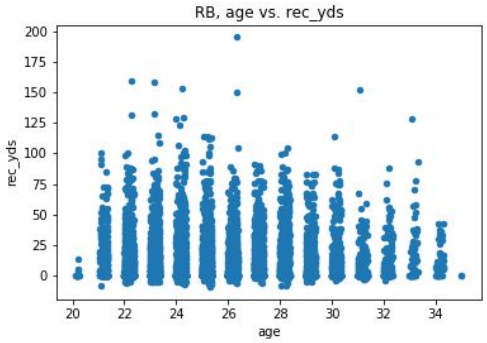


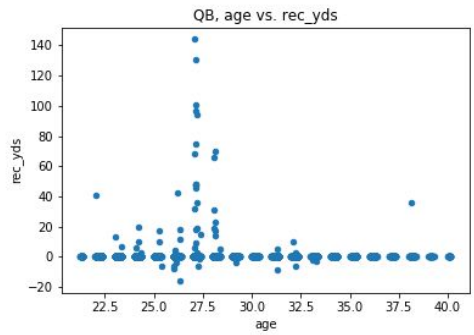


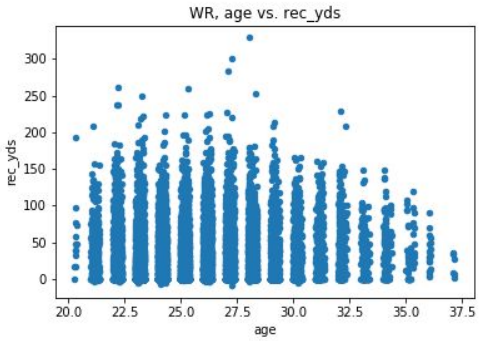


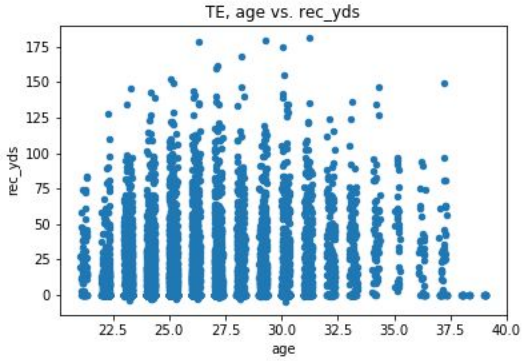
Age vs. Receiving Yards, Grouped by Position











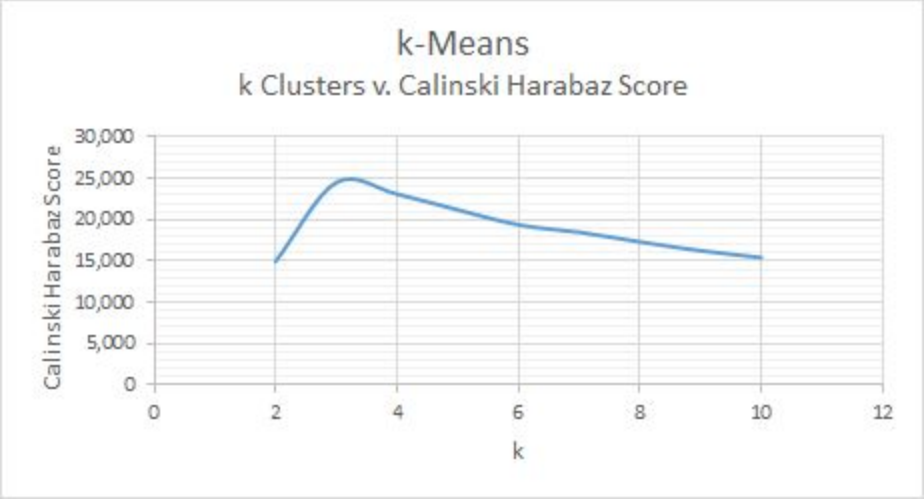
Part 3: Cluster Analysis

We clustered the player game statistics using three different techniques: DBScan, K-Means, and Hierarchical (Ward). Using the Calinski-Harabaz procedure to evaluate the labels, K-Means created the best clusters, followed by Ward, and then DBScan.

Technique	Calinski-Harabaz Score	Clusters
K-Means	24,517.54	3
Hierarchical (Ward)	24,032.66	3
DBScan	10,837.92	4

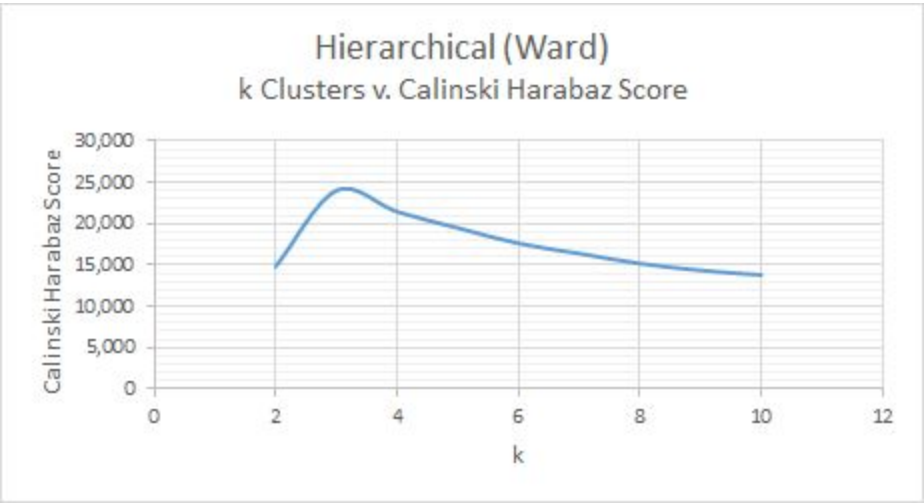
K-Means

Clustering the data using K-means requires the user to specify *k*, the number of clusters. To decide on which *k* to use, we ran the algorithm using *k* values from 2 to 10 and compared Calinski-Harabaz scores to find the best one. In this case, 3 clusters yielded the best labels for our data.



Hierarchical (Ward)

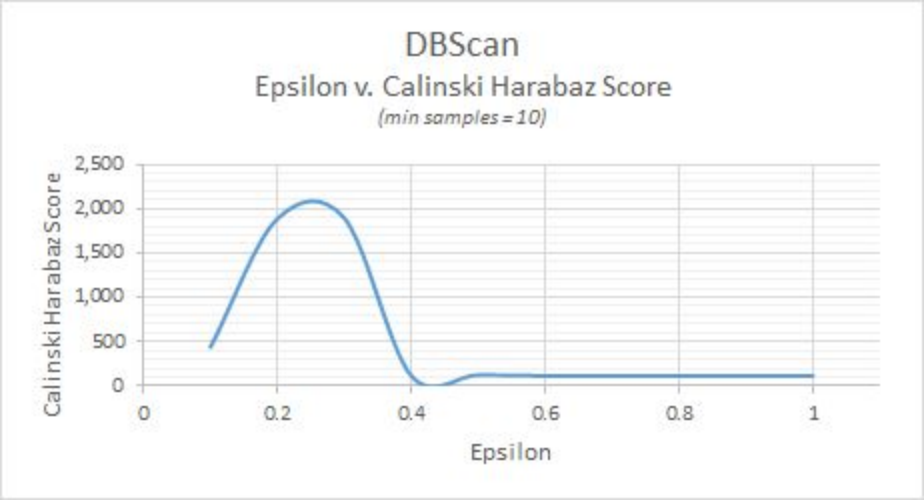
Clustering the data using Hierarchical (Ward) also requires the user to specify *k*, the number of clusters. Using this technique, the data is grouped into one large cluster and then cut at the specified *k* to create those clusters. Again, we ran the algorithm using *k* values from 2 to 10 and compared Calinski-Harabaz scores to find the best one. And again, 3 clusters yielded the best labels for our data.



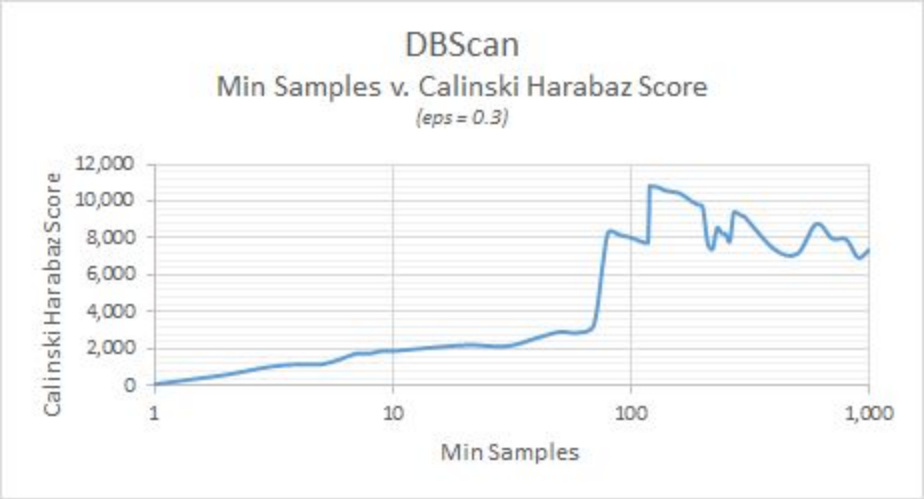
DBScan

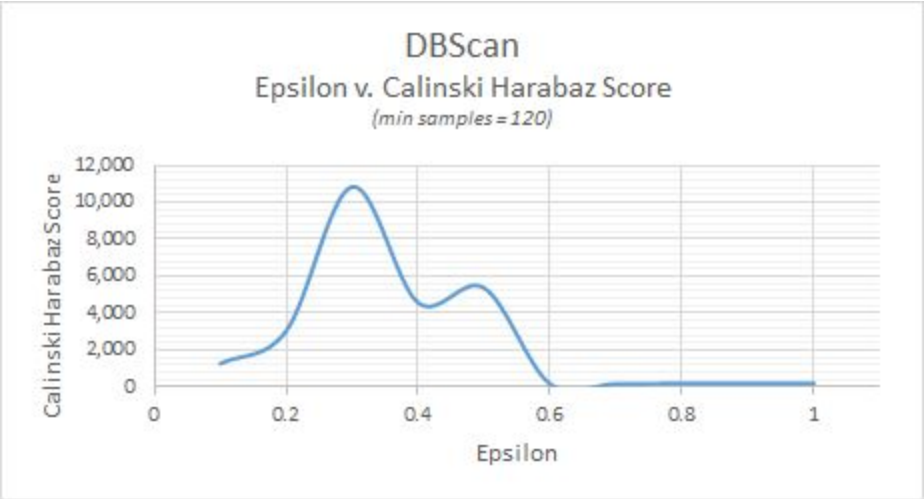
Clustering the data using DBScan, instead of requiring the number of clusters, epsilon and min-samples are required. Epsilon represents the distance between core points and the other points and min-samples represents the minimum number of points to fall within epsilon distance for a point to be considered core.

First, we held min-samples constant at 10 and ran the algorithm with epsilon values from 0.1 to 1.0 in increments of 0.1 to find the optimal epsilon using Calinski-Harabaz scores to compare the runs[2]. This yielded 0.3 as the best epsilon to use. Next, we kept epsilon = 0.3 constant and ran the algorithm again using many different min-sample values between 1 and 1000. The optimal min-samples value according to the Calinski-Harabaz scores was 120. Finally, we kept min-samples = 120 constant and again ran DBScan using epsilon values between 0.1 and 1.0 in increments of 0.1 to find the optimal epsilon at that min-samples value. Again, 0.3 was the optimal epsilon value.



[2] Since the clustering procedure is computationally expensive, we could not run tests for optimizing eps and min-sample at the same time. We had to optimize them separately.





Analysis

The clustering of the data ended up being a fairly good way to determine what position a player plays based on their statline. Each technique was very good at clustering quarterbacks and kickers while they each had trouble clustering wide receivers, running backs, and tight ends. Wide receivers and tight ends are very similar positions, so it was expected that those two would be difficult to split. However, it is surprising these algorithms had such a hard time clustering on running backs.

The following are the defined labels and the distributions of positions falling in each one.

K-Means					
Label	**QB**	**WR**	**RB**	**TE**	**K**
0	398	12,702	9,589	7,289	237

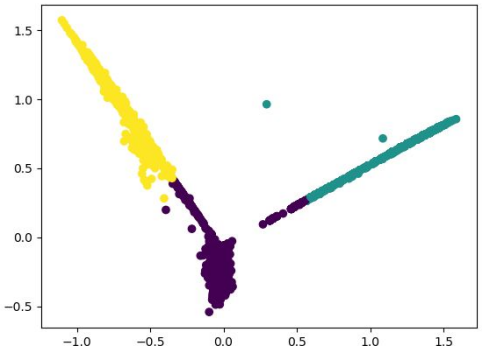
1	0	0	0	0	2,403
2	2,727	26	9	0	1

Hierarchical (Ward)					
Label	**QB**	**WR**	**RB**	**TE**	**K**
0	294	12,700	9,589	7,289	129
1	0	0	0	0	2,511
2	2,831	28	9	0	1

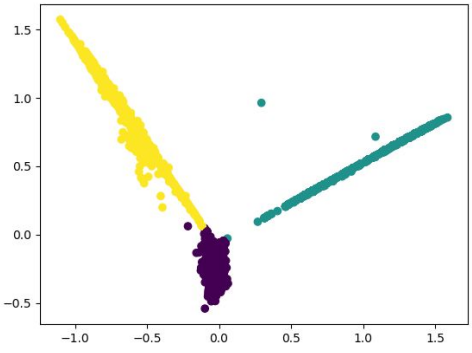
DBScan					
Label	**QB**	**WR**	**RB**	**TE**	**K**
0	315	12,371	9,117	7,233	51
1	1,957	0	0	0	0
2	0	0	0	0	382
3	0	0	0	0	1,795
-1	853	357	481	56	413

The following are the PCA projections of the clusters. The clusters were built using 40 attributes. These projections attempt to reform the data to 2-dimensions. Because of the large reduction in dimensionality, it is very difficult to gain any additional insight on the original datasets and clusters.

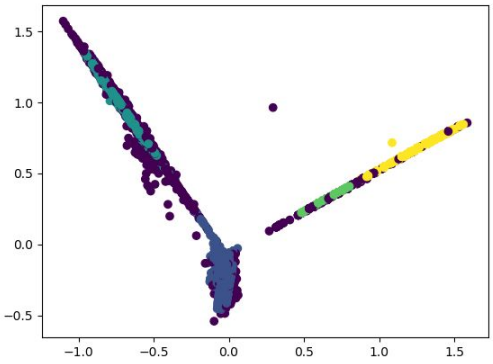
K-Means



Hierarchical (Ward)



DBScan



Part 4: Association Rules / Frequent Item Set Analysis

Finding rules that are frequent in a dataset helps having a better understanding the of the data. We used the apriori algorithm to find the frequent patterns that exists in out data.
But like other steps, we needed to prepare our data for the apriori algorithm. Apriori requires a list of transactions made from a list of items. Out dataset input file is a CSV file which is equivalent to a database table. Our first step was to convert this table-like dataset into a transaction set. Here are the list of operations we performed to get the transaction list:

- **Remove all zero (default) values:** As mentioned earlier, apriori requires a list of transaction. It is useless to specify which items were not present in a transaction, only those which are present matter. Our dataset is a high dimensional dataset and it suffers from the curse of dimensionality, meaning that it is sparse. In such sparse dataset, the support for 0 and default values is high and the combinations of these zero values is transferred from one one iteration of apriori to the next, leading to a huge computational overhead. The initial run with the zero values took 5 hours and halted with an out of memory exception.
- **Removing rows:** Not all the records that exists in our dataset matters to this part of the analysis. As an example, gs [historical] is a field indicating if a player played in the game or not. Naturally, if the value of gs is 0, other fields of the record are meaningless for our analysis. All rows where gs was 0 were removed.
- **Removing columns:** A few columns of the dataset were removed prior to running apriori. The reasons for column removal could be categorized as follow:
    - **ID column:** The id columns does not provide any information about the entities of our analysis. Hense, they were excluded.
    - **Same value column:** In the previous step, all rows with gs=0 were removed. We also removed the attribute gs in this step, since it could only have the value of 1.
    - **Derived attributes**
    - **Player/Team:** We are looking for general rules in our analysis, not specific rules related to a player or a team.

- **Binning and naming values:** When preparing the list of transaction for apriori, an X value for pass_int should be different than the same X value for pass_sacked. To distinguish these values, we added a prefix to each item in our dataset. Also, we mostly have numeric values and we need to convert them to items. We chose equi-width binning to convert numerical data to categorical data.

After preparing the data, a transaction list was extracted from it and feed to apriori algorithm with different support and confidence threshold. The statistical results of these run are illustrated in the following table.

	TE	**WR**	**QB**	**RB**
min_support=0.3	4188	4000	8084	11768
min_support=0.5	588	496	1284	1500
min_support=0.7	128	192	304	148
min_support=0.9	32	52	40	32

Number of rule extracted based on the position and minimum support value. Note that A -> B and B -> A are considered two different rules.

Results

In this section, we will discuss a few rules extracted by apriori, since it is impossible to discuss all the extracted rules.

Rule 1: (support: 1.0, confidence: 1.0)

TE -> kick_ret_yds_per_ret:pos
kick_ret_yds_per_ret:pos -> TE

Both these rules are absolute i.e. it always happen. Surely, this rule is not surprising since the 'Kick Return Yards per Return' for tight end should always be positive.

Rule 2: (support: 0.96)

WR, kick_ret_yds_per_ret:pos -> visibility:0_15 (confidence: 0.96)
WR, visibility:0_15 -> kick_ret_yds_per_ret:pos (confidence: 0.99)
visibility:0_15, kick_ret_yds_per_ret:pos -> WR (confidence: 0.96)
Confidence SD: 0.021

This rule is close to previous rule, except it is for wide receivers. What is interesting about this rule is the confidence levels of each derivation of the rule. In our analysis, the position and weather visibility are inputs, while kick_ret_yds_per_ret is a dependent variable. Apriori only consider the co-occurence of different items and has no clue about their causality relation. But in this case, the confidence of the most sound rule is higher[3].

Rule 3: (support: 0.8)

TE -> dome:N (confidence: 0.8)
dome:N -> TE (confidence: 1.0)

This rule is an example of a completely useless rule! Playing under a dome and the position are both inputs of our analysis.

Rule Pruning

As rule 3 from the last show, not all the rules extracted by apriori are valid. We have defined a step after the rule extraction to remove those rules that are invalid according to the inputs and outputs of our analysis. In this step, we remove the rules that falls into one of these categories:

- All the items in a rule set are inputs
- All the items in the consequence are inputs

With these two simple rule, we prune the extracted rules of the apriori output. The statistics of this pruning procedure is illustrated in the following table.

	TE	**WR**	**QB**	**RB**
min_support=0.3	2362 (37%)	2362 (41%)	5026 (37%)	8158 (30%)
min_support=0.5	368 (37%)	272 (45%)	854 (33%)	108 (32%)
min_support=0.7	64 (50%)	96 (50%)	176 (42%)	80 (45)
min_support=0.9	16 (50%)	32 (38%)	22 (45%)	16 (50%)

Number of rule extracted after pruning based on the position and minimum support value. Note that A -> B and B -> A are considered two different rules. The value in parenthesis is the percentage of Rule pruned.

[3] This may be a pattern in the rule set.

Part 5: Hypothesis Testing

Hypothesis 1:
Since both wide receivers and tight end catch the ball, we are interested to see if there is difference in the number of receptions when receptions is over 5
Null Hypothesis: The mean of receptions of wide receivers is not different from the mean of receptions of tight end.
Alternative Hypothesis: The mean of receptions of wide receivers is different from the mean of receptions of tight end.
We perform t-test for this hypothesis.
P-value is 0.351 > 0.05. Therefore, we reject out alternative hypothesis. There is no difference in the mean of receptions, when receptions is over 5.

Hypothesis 2:
We are interested to see if the more intense the wind is, the less passing is completed.

Null Hypothesis: Pass completion percentage of players have no relationship with wind intensity.
Alternative Hypothesis: The more intense the wind is, the less passing is completed.

We perform linear regression for this hypothesis.
P-value is 0.270 > 0.05. Therefore, we reject out alternative hypothesis. There is no relationship between wind intensity and pass completion percentage.

Hypothesis 3:

We are interested to see if we can predict Quarterback's pass completion percentage based on the weather. Since this is a classifying problem, we use machine learning methods for this hypothesis including Decision tree, KNN, Naive Bayes, SVM, and Random Forest.

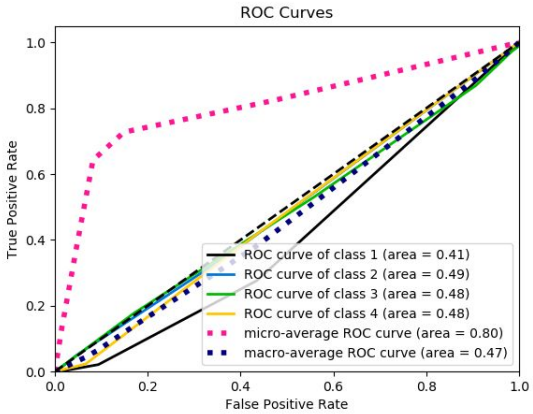
To get a better result, we filter out all the pass completion percentages equal to 0. Then we bin the percentages at a width of 25, resulting in four classes, which are 0%-25%, 26%-50%, 51%-75%, 76%-100%.

Null Hypothesis: We can not predict Quarterback's pass completion percentage based on the weather.

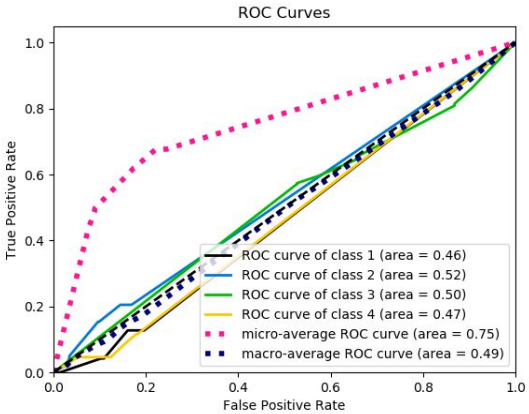
Alternative Hypothesis: We can predict Quarterback's pass completion percentage based on the weather.

For this hypothesis, we are going to use KNN, decision tree, Random forest, SVM and Naive Bayes, and see which method performs better.

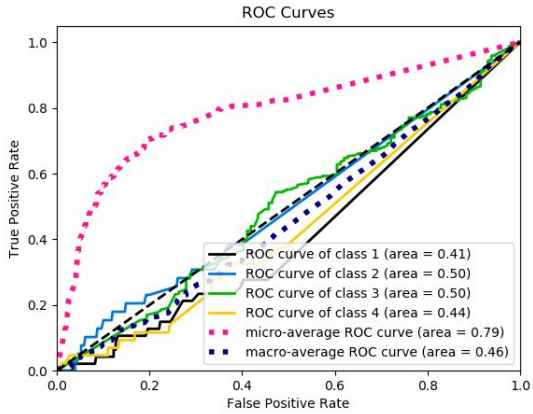
KNN ROC Graph



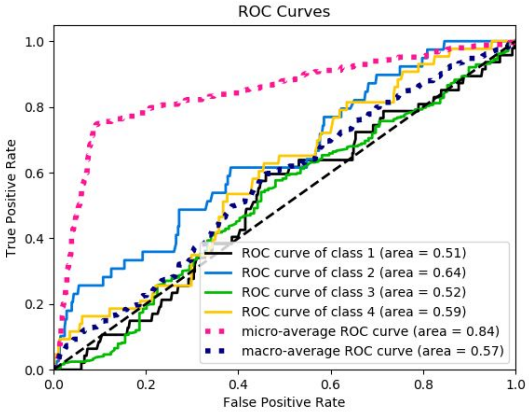
Decision Tree ROC Graph



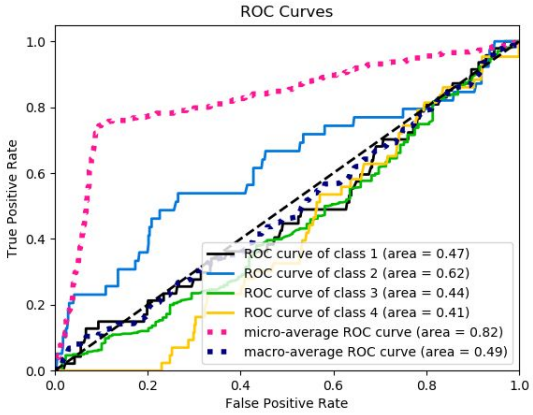
Random Forest ROC Graph

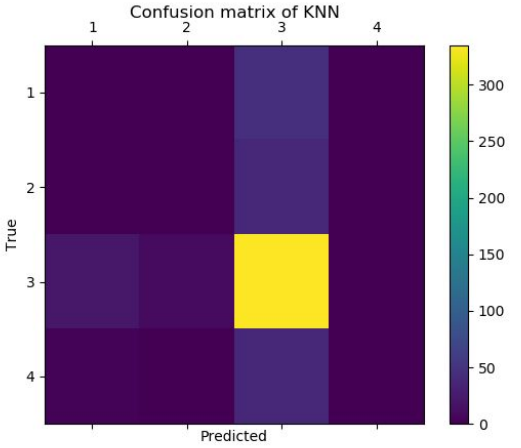


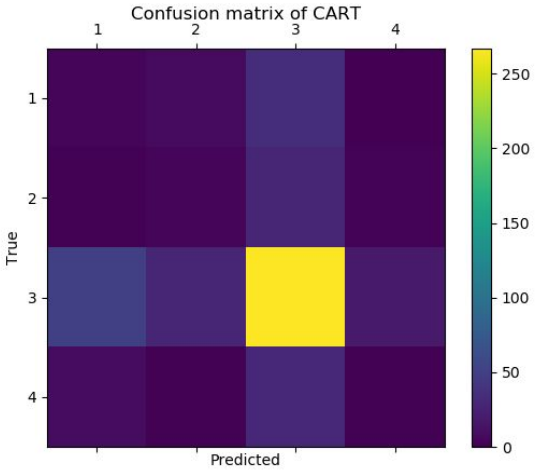
SVM ROC Graph

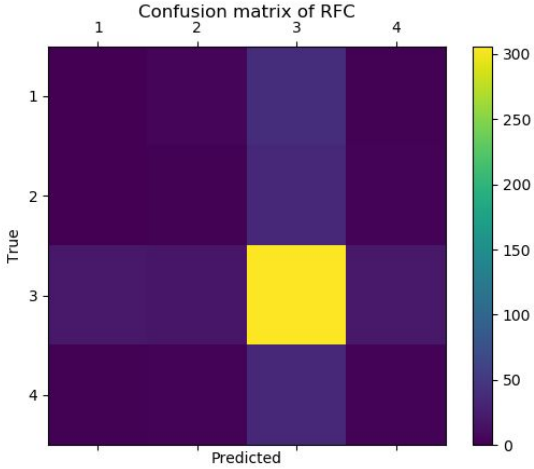


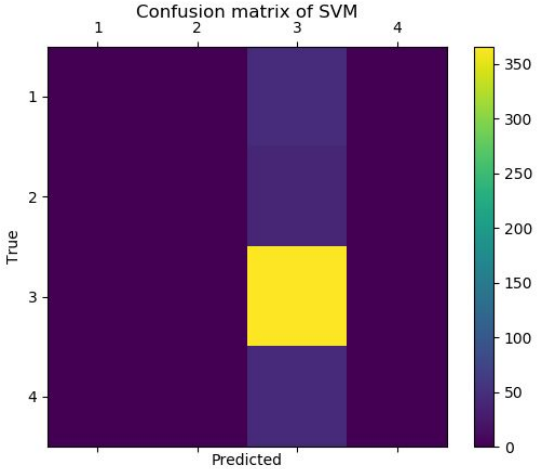
Naive Bayes ROC Graph

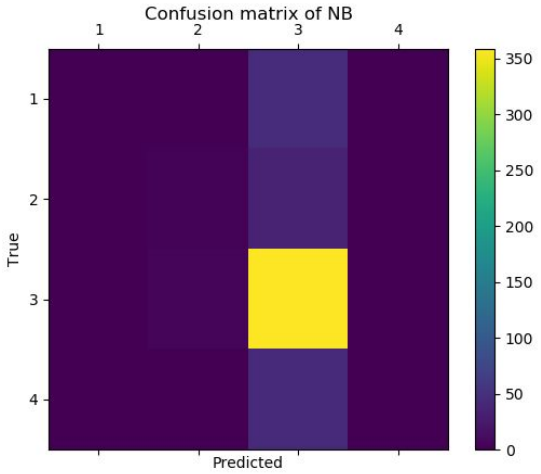

ROC Curves
True Positive Rate
False Positive Rate
ROC curve of class 1 (area = 0.47)
ROC curve of class 2 (area = 0.62)
ROC curve of class 3 (area = 0.44)
ROC curve of class 4 (area = 0.41)
micro-average ROC curve (area = 0.82)
macro-average ROC curve (area = 0.49)


Confusion matrix of KNN
True
Predicted


Confusion matrix of CART
True
Predicted


Confusion matrix of RFC
True
Predicted


Confusion matrix of SVM
True
Predicted



As we can see from the graphs, most of the models perform averagely. KNN: 0.651515 (0.034698) CART: 0.510606 (0.034362) RFC: 0.578283 (0.031236) NB: 0.699495 (0.035299) SVM: 0.714141 (0.041965).

For all methods, the third class has the highest true positive rate. Support vector machine performs the best. It performs the best on class 3, and for other classes, it all performs better than guessing. We think it's because SVM performs well when data has many attributes.

Naive Bayes performs the second best. It performs the best on class 3, while averagely on other classes. We think it's because the weather data are mostly independent to each other.

KNN performs averagely. According to the ROC graph, for all classes, it's prediction is the same as guessing. We think it's because we did not scale the data.

Both decision tree, and random forest perform not so well. We think it's because the weather attributes don't interact with each other that much.

We think we can keep our alternative hypothesis, that we can predict Quarterback's pass completion percentage based on the weather. However, we may need better binning strategies for the attributes, and improve the parameters of support vector machine.