**NFL Box Score Analysis: Predicting Team Trends using Association Rule Mining**

By Sachin Muralidharan

Objective and Project Description:

The NFL is one of the largest sports leagues in the world and the most-watched league in America. In recent years, the league has undergone a massive transformation, and teams have begun to change their methods of talent acquisition and evaluation. Teams are now increasingly relying upon data and analytics to gain a competitive advantage. The league's new data-driven approach has produced many intriguing new applications regarding player safety, fan engagement, and revenue optimization. Moreover, complex machine learning models have been developed to predict a given player's success within the league, which have heavily influenced teams' free agency and draft decision-making. Moreover, the win probability model has gone viral in the past few years, and this model can predict the outcome of a game beforehand with a relatively high success rate. While many of these techniques are complex and difficult to implement, the objective of this project is to use association rule mining to achieve similar results. In this project, I will be analyzing data from the 2022 NFL season, and using association rule mining, I will attempt to find hidden patterns within the box score data of a certain game to determine what factors had the greatest impact on the outcome of the game.

For example, if a quarterback has more than 25 pass attempts in a game, this could lead to a higher win probability for a team. Complex machine learning models would undoubtedly perform better in terms of predicting very specific statistics, however, I am hoping to use the more accessible and intuitive approach used with association rule mining to discover if a game’s outcome can be broadly classified by a few key statistics which may hold a greater weight against others. I will be web scraping the Pro Football Reference website to obtain data from the 2022 NFL season, and then perform association rule mining on the dataset using the Apriori algorithm.

Pro Football Reference:

Pro Football Reference is a website that provides historical data on NFL statistics, including player and team statistics, game history, standings, and schedules. There are many other ways to access NFL data online, however, Pro Football Reference provides data in a structured format, making it easy to access targeted information. This makes the website ideal for web scraping purposes. However, because PFF does not provide a way to query data from the website with an official API format, there would typically be concerns about the legality of scraping data from the site. PFF allows users to retrieve data from the site, but the user is only allowed a certain amount of requests per hour. This proved to be an issue because I was unable to gather the information I needed in a short period of time, and instead, there were instances where I needed to wait for multiple hours before getting the rest of the data. The issue is that if a user exceeds their limited requests, the site automatically forbids future requests because it deems the entity to be a web crawler, and there are several restrictions regarding this within the robots.txt file. There were some issues with gathering data from PFF, but it was key in implementing this project. In the future, if I wanted to expand upon this project, I would likely have to pay for an API that would not limit my requests.

Beautiful Soup:

Beautiful Soup is a Python library used for web scraping, web crawling, and data extraction. The library is beginner friendly, simply requiring a user to input a URL to analyze, and it outputs the HTML contents of the page with a relatively low run-time. Beautiful Soup is a versatile library and can be applied to far more complex tasks, however, my main usage for the library was to retrieve the contents of web pages from the PFF site.

Regex:

Regex is a popular Python library used to easily extract certain text from a given string. Regex is especially useful when handling unstructured data, and is an efficient way to parse data into manageable blocks. In this project, Regex was used to extract certain portions of HTML from structured data, obtaining blocks of text from the resulting beautiful soup object. One of the issues with the python pandas read HTML method is that it is unable to read the entire HTML contents of a webpage, due to issues with javascript notations. One remedy to this problem is to use the Selenium web driver library, which can adapt and retrieve all the contents of a webpage, despite the occurrence of different javascript notations in the code. However, my issue was resolved using a regex expression to identify table patterns within the beautiful soup output. This process I used is known as web scraping. Web scraping is when a user extracts certain data from a webpage for a specific purpose, which in my case, was to get the advanced statistics for each game played by a team. Recently, processes such as web scraping and web crawling have raised ethical concerns, however, PFF allows users to request materials from the site, therefore there are no such problems in this case.

y = (re.search('<div class="table\_container" id="div\_defense\_advanced">', x))

Regex expression used in my code.

Python Pandas:

Python Pandas is a library that provides data analysis and manipulation tools through its data frame structures. This library is lightweight, fast, and flexible making it an ideal fit for storing data for this project’s purposes. Once the data was scraped from the PFF site, it was stored in a pandas data frame and prepared for association rule mining.

MLxtend:

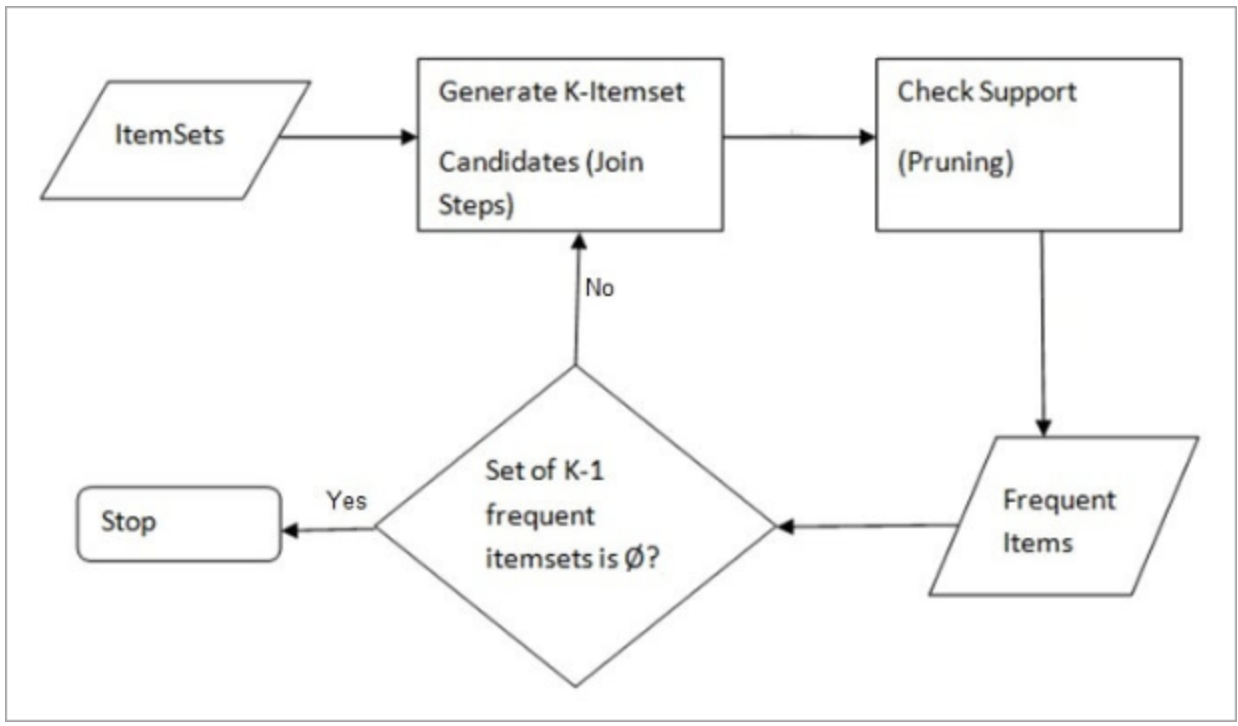
MLxtend is a free library used by data scientists for machine learning in Python. This is a library that is used to expand upon current machine learning libraries in Python such as Sklearn, Scipy, and TensorFlow, offering more functionalities that are not included in the original packages. In this project, MLxtend was used to expand upon the Apriori library in Python. MLxtend allows for an easy transition from a pandas data frame into a frequent itemset class developed by the apriori library. This was crucial for my project, because my data was not in the form of market basket transactions, but was instead in the original pandas' data frame stage for storing the relevant data. Mlxtend allows the user to filter results from the Apriori algorithm. For example, in this project, I wanted to filter results based on specific columns, like the player and win or loss columns. Additionally, the MLxtend library allows the user to incorporate metrics such as lift and confidence in the filtering process. Overall, this library proved to be very useful in accomplishing my analysis of the association rules within my dataset.

Datasets:

The datasets for this project were obtained from the box score data from each game that a team played in the 2022 NFL season. There were four groups of data that I selected from each game. These were the advanced rushing, passing, receiving, and defense statistics, and this data encompasses each phase of the game. The rushing statistics block includes items such as the number of broken tackles, attempts, yards, and touchdowns. The receiving statistics include targets, receptions, and yards among others. The passing statistics include figures like bad throw percentage, number of sacks, and number of drops. The defense statistics include quarterback hurries, yards per target, and completion percentage. Each of these datasets included key factors which provided insights into a team’s performance, making it ideal to analyze with association rule mining.

Apriori Algorithm:

I implemented my project using the Apriori algorithm. The Apriori algorithm was created in 1994, by data scientists Rakesh Agarwal and Ramakrishnan Srikant, which is a mark of how machine learning algorithms have been developed at a high rate in recent years. This algorithm classifies as an association rule mining technique. Association rule mining is the process of finding hidden patterns in a set of data to find relationships between the set of data points or “transactions” within the same set. One of the most common ways of displaying the algorithm is with a list of grocery transactions, where the algorithm discovers a certain combination of items that are likely to be purchased together. A key metric in this process for finding frequent itemsets is the support threshold. This is a value that indicates the minimum amount of times at which a set of items must appear together in that dataset, in order to meet the frequency requirements. In larger datasets, the support threshold would be set to a lower value, and vice versa for smaller datasets, however, the user can determine what support threshold they select, and can be considered as a hyperparameter, a value selected by the data scientist to “control the learning process”. The algorithm is broken down into a few steps. Firstly, the machine will form pairs of two sets from the items. Next, it will keep track of the number of times this set occurred within the transactions, and erase the sets which do not meet the support threshold set by the user. The algorithm then begins an iterative process, where it increases itemsets from size 2 to 3, 4, etc. One of the key tenets of this algorithm is the fact that if one item set is frequent, then each subset must be frequent as well. This is a concept taken from set theory in discrete mathematics and reduces the runtime of the algorithm by reducing the number of sets that need to be examined. Additionally, there are two other metrics that were used in this project to ascertain the validity of each item set, and these metrics are lift and confidence. Lift is an association rule mining statistic that calculates the correlation between two items. In plainer terms, lift measures the likelihood of the occurrence of B given A’s presence. Lift is determined by dividing the support of rule A implies B by the support of A multiplied by the support of B. In interpreting lift, a value that exceeds 1 indicates a higher correlation between the two components, while if the value is less than 1, this implies the opposite. One of the drawbacks of this measure is that lift can be heavily affected by the high frequency of one specific item. For example, if item A occurs in over 50% of the transactions within a list, any other item that occurs with A will naturally have a higher lift value, which will lead to incorrect correlation interpretations. Confidence is calculated by dividing the support of rule A implies B by the support of A. Confidence can be considered as the probability that the consequent item appears in a set containing the antecedent item. Once again, higher confidence values, typically above 0.5, implying that the itemsets are likely to have stronger associations, while values below 0.5 tend to mean the opposite. The drawbacks of confidence are somewhat similar to the concerns with lift values, as confidence can misrepresent the importance of an association due to the weight of the support of the antecedent within the calculation. There are different machine learning algorithms that may produce different results for this project. For example, a linear regression model would work for this numerical data and may be able to produce a model which predicts a player's statistic for a given game. However, the Apriori algorithm is sufficient for this project and will be useful for analyzing a given team's patterns during the games they played in the prior NFL season.



Apriori Algorithm Visualization

Apriori Algorithm Drawbacks:

One of the main drawbacks of the apriori algorithm is its lack of specificity with regard to a broad dataset. This is well illustrated within the NFL dataset. For example, if we look at a team's statistics for a specific game, we can usually have it in a format of column and value. An example of this is the passing yards column, with its corresponding value set equal to 300 yards. The apriori algorithm is best suited to deal with datasets in the form of a “basket of transactions”. The most commonly illustrated uses of the apriori algorithm are used with datasets of grocery transactions, or movies watched by selected users. These examples can be easily translated into the Apriori algorithm with little changes.

Computational Concerns:

Another drawback of the Apriori Algorithm comes in the form of its computational abilities, due to its “combinatorial nature”. The Apriori Algorithm implementation within Python is heavily dependent on the size of the datasets which are given by the user, along with the parameters by which the user filters out frequent item sets. For example, the algorithm had a large runtime when the support levels were set to any values below 0.2. This is likely because there were far too many association rules for the algorithm to output in some cases, causing a longer than expected-time. For large datasets, the number of itemsets to analyze is higher, and this leads to long processing times and memory usage, especially because the algorithm uses iterative processes to filter for frequent item sets. The slow runtimes make the usage of data structures and different algorithms even more crucial, and they must be accounted for in order to perform a high-level analysis of the data. Additionally, data preprocessing is necessary to filter out any unnecessary information which would needlessly add to the runtime.

Data Preprocessing:

One of the main components of data preprocessing is to remove “dirty data”. This may include noisy, inconsistent, and incomplete data. In my project, there were not many issues with outlier points. While the apriori algorithm certainly has a sensitivity to outliers, for the NFL datasets, I decided that the outliers could remain in the data, because the data was very relevant to the analysis being made, and therefore would not be a large concern. One of the complexities with the dataset was orienting it to fit the apriori algorithm. One of the assumptions for the apriori algorithm is that baskets of transactions have eliminated any unnecessary data because if this data remains in the set, there is a very high probability that the results will be skewed. I had successfully implemented the apriori algorithm, however many of the results had unwanted values associated with them, which was confusing. A column named “Unnamed 0”, which was a bizarre column added when retrieving the data was mistakenly added to the dataset, causing a distortion of the results. Another key portion of the preprocessing is the handling of NAN values. This was easily resolved in the NFL dataset. NFL statistics are discrete values, and a player will either have a value associated with a certain statistic, or it will simply be set to 0, as the player would not have a recorded value for that stat. Therefore, all the NAN values could simply be set to 0 in the dataset. However, this brought up another key issue, which was the incorporation of datasets. Within the project, there were four key sets for a team in a given game, which were advanced passing, rushing, receiving, and defense statistics. Ideally, these four sets would be combined to form one large dataset, encompassing the entire team's performance. However, this would result in crucial errors. For example, a running back would have much of their statistics concentrated in rushing stats, and therefore their stats for the other categories would be set to 0. This would theoretically occur very often throughout each game, and therefore would skew the dataset. Therefore, these datasets were required to be individually analyzed using the apriori algorithm. The final key issue with the data pre-processing for this project came in the format of the data. When retrieved from the website, the data was in the form of the column value, as indicated within a pandas data frame. However, due to the discrete nature of the data, this was not a good fit for the apriori algorithm. For example, if the algorithm was run on the raw data, this would result in association rules X->Y, where X and Y would likely be 2 numbers, providing the user with no context and way to interpret the results. Therefore a decision had to be made on how to prepare the data to provide real results. I made the decision to average out the numbers for a discrete-valued column, and list the values of the cells in each column in the form of >avg or <avg. This boolean style preprocessing allowed for tangible results, because depending on a player being above or below this boundary, it gives the user a clear indication of how their statistics impacted the game. Of course, there are drawbacks to this strategy, as it misses out on some of the intricacies of the data, and a potential solution could be to include more segments within the value. For example, the user could add if the player’s value is one-fourth of the value of the average or one-eighth of the value, but this would also make it more difficult for interpretation. Another point of note was that I manually added a win or loss column to the datasets, because this was one of the key aspects of my project, to determine if a certain player could sway the outcome of a match. Ultimately, while I obtained a sufficient amount of total data, the individual computations for the apriori algorithm likely did not have enough data to generate a robust frequent itemset.

Interpreting Results:

Ultimately, the results of the experiment proved to be mixed for this project. There were a few key takeaways from these results. Firstly, the idea that a specific player could have a proportionately large impact on a game, and studying this potential phenomenon with the apriori algorithm, proved to be difficult to analyze. The initial hope for this project was to obtain frequent itemsets in the form of player implies win or loss. However, after conducting this project, it's clear that this goal would have been incorrect. For example, suppose a team performed very well during the season, and their proportion of wins to games played was very high. An implication of a player implying a win would be very unclear and would be difficult to interpret. This itemset may be more useful if the project incorporates analyzing past seasons as well because this would provide more data about each season, and it would be more applicable to see which players led to wins or losses, given the fact that different players would depart teams during each offseason. However, when filtering through the itemset results, it became clear that there was one dataset where the combination of a player's name in a specific itemset, alongside the other statistics, was quite common, and this was the passing statistics section. Further analysis indicates that these results have a certain relevance because the quarterback of the team is the player who occurs the most in the passing datasets. Moreover, the quarterback of a team always touches the ball on each offensive play, and therefore, they will naturally be tied to these results more often than not. One result of the association rule mining which I found to be quite relevant occurred when analyzing the Kansas City Chiefs, whose quarterback was the MVP of the NFL. Below is a list of two interesting itemsets involving Chiefs quarterback Patrick Mahomes, which both occurred at a support threshold of 0.684211.

support itemsets

0.684211 (Patrick Mahomes, <avgBad%)

0.684211 (>avgIAY/PA, Patrick Mahomes)

In the first example, the Bad% is defined as the “Percentage of poor throws per pass attempt, excluding spikes and throwaways”. This indicates that Mahomes' percentage of bad throws was below his average for most of these games, which shows that most of his bad throws only occurred in a few games, which skewed the average. The second statistic, avgIAY/PA, is defined as “Intended air yards per pass attempt - Average depth of the target, whether completed or not”. For much of the season, Mahomes was above his average IAY/PA. This shows a heightened aggressiveness when passing down the field, as he tended to go for plays that would gain more yardage. Overall, these two results indicate some success with the project, because the league MVP, proved to have some very good results after evaluating the Apriori algorithm. However, while this provided good results for the quarterbacks, it did not showcase any benefits for other important players. For example, the offensive player of the year, along with the defensive player of the year for the 2022 NFL season did not appear in any of the frequent itemsets when filtering for these players. These players, unlike the quarterbacks, are not guaranteed to touch the ball on each play, and there is a higher proportion of different players within the dataset as well, which will decrease the likelihood of one player appearing in a set. Below, I’ve listed some of the interesting results I’ve found for three teams. One important thing to note is that the Mlxtend library does not allow for filtration in certain cases. For example, I filtered one of the frequent itemsets by the Player column in my data frame, but this does not produce the lift or confidence values, and therefore for some cases, I’ve filled in these values as NAN.

| Team | Category | Rule | Support | Confidence | Lift |
| --- | --- | --- | --- | --- | --- |
| San Francisco 49ers | Receiving | (<avgAtt/Br)-> (W) | 0.798 | 0.798 | 1.039 |
| San Francisco 49ers | Defense | (<avgSk, <avgQBKD, <avgHrry, <avgTD) -> (<avgPrss) | Support(A) = 0.55  Support(B) = 0.619403 | 1.0 | 1.614 |
| San Francisco 49ers | Passing | (<avgCAY/PA, Jimmy Garoppolo) | 0.37037 | NAN | NAN |
| Minnesota Vikings | Passing | (>avgAtt)  -> (Kirk Cousins, >avgCmp) | 0.60 | 1.0 | 1.667 |
| Minnesota Vikings | Rushing | (<avgBrkTkl)-> (W) | 0.641791 | NAN | NAN |
| Kansas City Chiefs | Passing | (<avgBltz)->(W) | 0.656126 | NAN | NAN |
| Kansas City Chiefs | Rushing | (<avgAtt/Br)->(W) | 0.6750 | NAN | NAN |

\*I’ve only listed some key results for 3 teams listed here, as there are too many results to list in this report

Project Expansion and Conclusion:

After reviewing the results, I’ve decided on a few ways in which this project could be improved, and expanded upon in the future. Firstly, incorporating results from past seasons would help increase the relevance that specific players have in the results. The NFL is known for its lack of continuity between seasons, as players frequently change teams. I suspect that if I perform association rule mining on a more historical dataset, a clear pattern will emerge where a team's win or loss column will be determined by the statistics of certain players. Additionally, combining the different statistics into one whole dataset would be necessary to have a more thorough analysis. Currently, each team’s data comprises four groups, and it is difficult to combine the groups because this would produce inaccurate results. For example, a defensive player’s statistics could somehow be correlated to a receiving statistic, even though this clearly does not have any correlation. Discovering a way to “normalize” the data could provide for an overall team analysis for each game, although I am unsure of how this could be done given the current datasets. Another possibility would be to include features describing the circumstances for which each game was played. For example, I could add a feature describing if the game was played at home or on the road. Additionally, there could be a metric to describe a team's performance at the beginning of the season versus the end of the season, as some teams may start faster than others. These are just a few metrics that could be used to expand the analysis, but ultimately, the expansion of this project would result in acquiring more data and finding a way to combine specific sets together for a “meta” analysis.

References:

*Https://Www.researchgate.net/Publication/351168385\_Research\_and\_Case\_Analysis\_of\_Apriori\_Algorithm\_Based\_on\_Mining\_Frequent\_Item-Sets*.

*Https://Www.researchgate.net/Publication/305333727\_Apriori-based\_diagnostical\_analysis\_of\_passings\_in\_the\_football\_game*.

*Https://Stmorse.github.io/Journal/Pfr-Scrape-Python.html*.