SWIFTY ANALYTICS

Applied Machine Learning for Improved Sports Betting

IST 707 FINAL PROJECT

Cheromaine Smith, Sarah Morris, Charles Hohl, Santhosh Arunagiri, Joshua Wiser

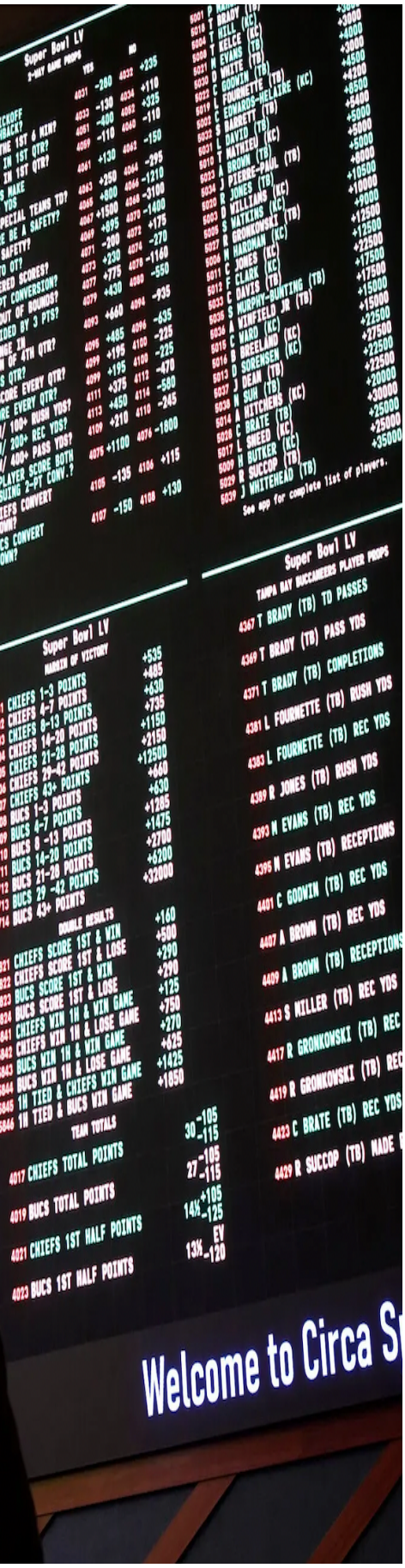
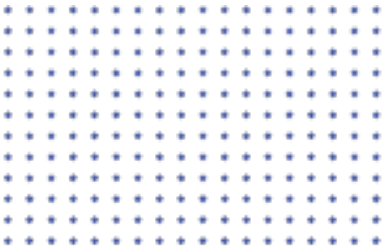


TABLE OF CONTENTS

INTRODUCTION



Sports is one of the biggest drivers of economic development in the US, with an estimated value of $163 billion for all NFL franchises in 2023. [1] With the overturning of Professional and Amateur Sports Protection Act (PASPA) by the Supreme Court in 2018, which allowed states to legalize and regulate sports betting, there was an even more substantial increase in the economic impact of sports-related activities. In 2024, a record 67.8 million Americans reported that they would bet on the Super Bowl LVIII, a 35% increase from all sports betting participants in 2023.[2] Just the Super Bowl alone saw $23.1 billion dollars in bets, up from $16 billion dollars from last year's Super Bowl. Americans remain divided on whether legalized sports betting is an exciting pastime, or rather an irresponsible and dangerous practice. However, one thing remains clear: with so much money in the pot, the stakes are higher than ever for sports bettors who seek to strategically improve their betting predictions.

Currently, Americans use various tactics to improve their sports betting outcomes. Some sports bettors devote hours to research, contributions on online forums, analyzing established betting lines, and review previous games' data. Some Americans have even taken matters into their own hands to improve their sports betting outcomes. During the 2021 Super Bowl, one avid sports bettor placed a $50,000 prop bet that there would be a streaker during half-time -- then proceeding to perform the act himself (despite expected winnings of $374,000, he was unable to collect due to rule violations). When tampering with the game is off the table, bettors must rely on the most current and available data in their decision making to have the best betting outcome.:

With 1 and 5 Americans partaking in sports betting annually, the odds are against the average bettor. So how can one improve their odds of an accurate prediction, and therefore improved betting outcome? This is where sports begins to overlap with data science, and in particular, the prediction models developed through the analysis of past data. Through the use of previous sports data, one can search for trends and patterns in the data that sway outcomes in a certain direction, despite the established odds of each bet.

Problem Statement

In the lens of data science, sports betting is a prediction. Therefore, can sports betting predictions be improved with machine learning prediction models? In this report, we will use various machine learning models including decision tree algorithms, heat maps, random forest, clustering, and naïve bayes models on compiled NFL teams, stadiums, and scores data. Through data exploration and the use of these machine learning models, hidden trends can be unearthed in these vast datasets to identify patterns to predict the outcomes of future NFL games. Ultimately, the objective of this paper is to contribute to discourse surrounding sports analytics and assist sports betters in making responsible, data-driven, and informed bets.

OVERVIEW OF THE DATA

Data Description

The datasets provide a highly detailed view of National Football League games, to include individual match scores, team statistics, and stadium characteristics across several seasons. The information spans game dates, scores, team favorites, betting lines, and playoff occurrences, alongside environmental conditions such as temperature, wind, and humidity during the games. It also includes specific team details, such as conference and divisional alignments, and comprehensive stadium information including location, capacity, type, and weather station data. This compilation of historical data allows for in-depth analysis of game outcomes, team performances, and the potential impacts of various factors on the sport.

About the Data

The data is in three separate CSV files that were downloaded from Kaggle.com. The datasets are as follows:

Spreadsheet 1: spreadspoke_scores.csv

This dataset captures historical game data for a professional American football league. Each row corresponds to an individual game with columns detailing:

- schedule_date: The date the game was played.
- schedule_season: The season year of the game.
- schedule_week: The week number within the season.
- schedule_playoff: A boolean indicating if the game was part of the playoffs.
- team_home: The home team's name.
- score_home: The home team's score.
- score_away: The away team's score.
- team_away: The away team's name.
- team_favorite_id: The identifier for the team favored to win.
- spread_favorite: The point spread for the favorite team.
- over_under_line: The betting line for the total points to be scored by both teams.
- stadium: The name of the stadium where the game was played.
- stadium_neutral: A boolean indicating if the game was played at a neutral stadium.
- weather_temperature: The temperature at game time.
- weather_wind_mph: The wind speed in miles per hour at game time.
- weather_humidity: The humidity percentage at game time.
- weather_detail: Additional details about the weather, if available.

Spreadsheet 2: nfl_teams.csv

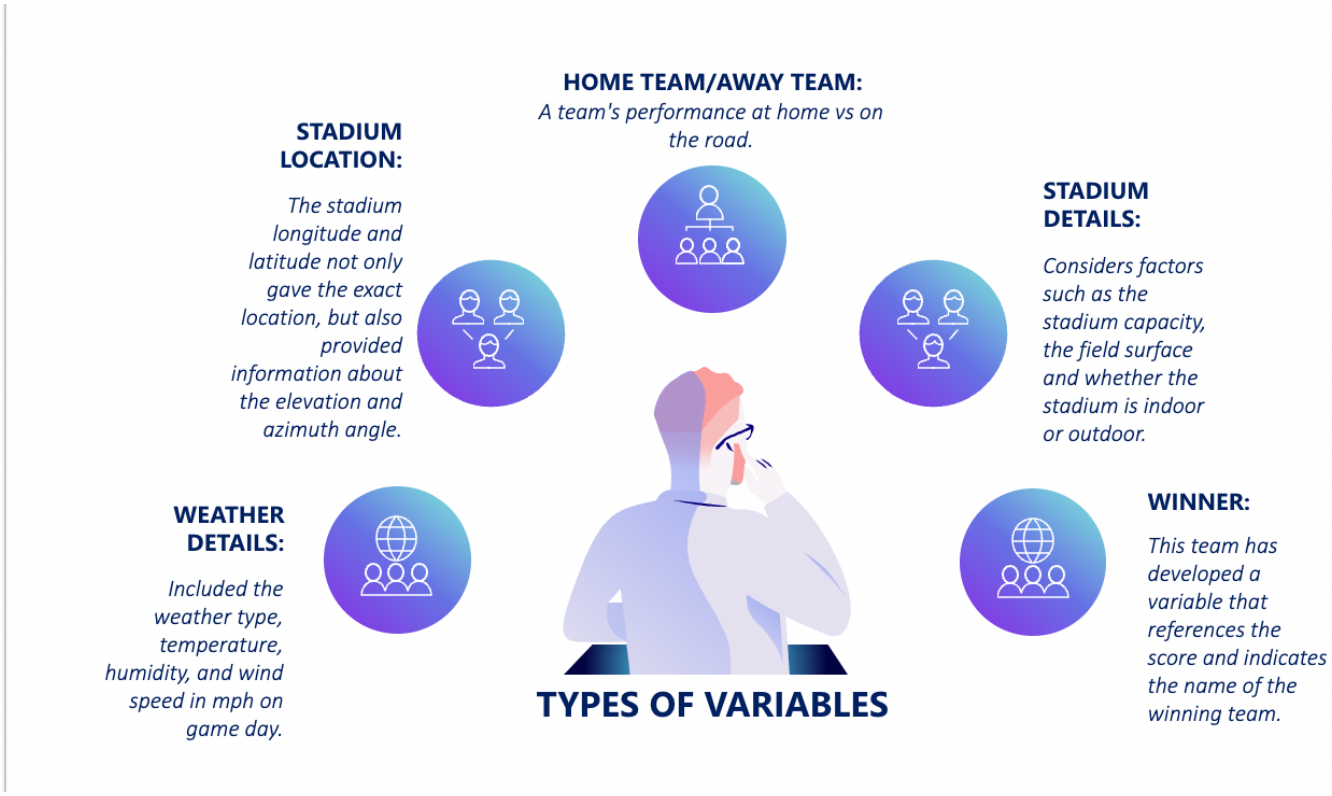
This spreadsheet provides details about the teams, including:

- team_name: The full name of the team.
- team_name_short: The abbreviated name of the team.
- team_id: A unique identifier for the team.
- team_id_pfr: An alternative team identifier.
- team_conference: The conference in which the team plays.
- team_division: The division in which the team plays.
- team_conference_pre2002: The team's conference before 2002 reorganization.
- team_division_pre2002: The team's division before 2002 reorganization.

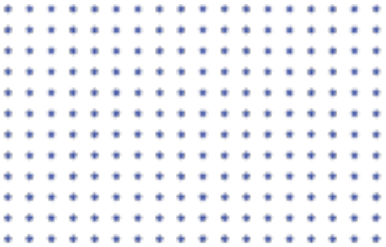
Spreadsheet 3: nfl_stadiums.csv

The third spreadsheet focuses on the stadiums, providing such details as:

- stadium_name: The name of the stadium.
- stadium_location: The city and state where the stadium is located.
- stadium_open: The opening year of the stadium.
- stadium_close: The closing year of the stadium, if applicable.
- stadium_type: Indicates whether the stadium is indoor, outdoor, or has a retractable roof.
- stadium_address: The street address of the stadium.
- stadium_weather_station_zipcode: The zipcode for the stadium's associated weather station.
- stadium_weather_type: Describes the typical weather conditions of the stadium's location.
- stadium_capacity: The seating capacity of the stadium.
- stadium_surface: The type of playing surface in the stadium.
- stadium_weather_station: The code for the nearest weather station.
- stadium_weather_station_name: The name of the nearest weather station.
- stadium_latitude: The latitude of the stadium.
- stadium_longitude: The longitude of the stadium.
- stadium_azimuthangle: The azimuth angle of the stadium.
- stadium_elevation: The elevation of the stadium above sea level.



DATA CLEANING



Data cleaning is an essential step of any data analysis. A common saying in data science is “Garbage in, garbage out. Therefore, the initial preprocessing of cleaning of the data is imperative to ensure that the data has minimal noise, is standardized, and can be easily fed to machine learning models. All of our datasets – Stadiums, Teams, and Scores, contained NA values and some inaccurate information that could potentially skew the data analysis.

NFL Game Scores

Our initial dataset encapsulated a comprehensive historical record of NFL games, detailing aspects such as game dates, participating teams, final scores, and betting odds. Much of early analysis involved checking these records for completeness and accuracy. Some records lacked game outcome data (i.e., final scores), possibly indicating accidental omissions. Some variables are very important to betting predictions, like over/under line. However, this was not tracked regularly (outside of Championship Games) until the early 1980’s. It was important not to skew predictions by including data that does not encapsulate the entire data set fairly. To refine our dataset for predictive analysis, we excluded many of these incomplete records. Finally, categorical variables such as whether the picked favorite won were added for additional prediction parameters.

“Data is the nutrition of artificial intelligence. When an AI eats junk food, it’s not going to perform very well.”

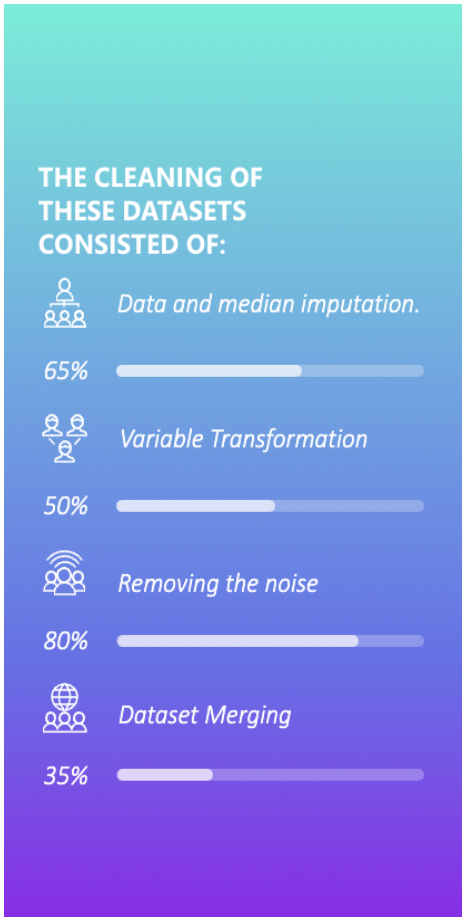
Matthew Emerick

NFL Teams

The team information dataset offered a detailed inventory of NFL teams. Data included their names, affiliations (conference and division), and unique identifiers. This dataset required the least amount of pre-processing and cleaning. However, it did highlight the importance of a nuanced approach to analysis as many of the teams have changed over the years. Some changes are relatively minor, such as teams changing cities (e.g., Oakland Raiders to Las Vegas Raiders), whereas other changes were somewhat substantial like new teams being added, or other teams disbanding entirely.

NFL Stadiums

The stadiums dataset was a collection of data points including stadium names, locations, capacities, surface types, and operational timelines. This dataset presented many categorical and continuous variables. Many of these variables offered insight into the environmental aspects of the games. There were several instances of missing data, particularly regarding the operational status of stadiums (e.g., closure dates for stadiums still in use). Additionally, some of the data in this spreadsheet was not particularly useful for prediction purposes. Variables such as the location of the weather station for a particular stadium are not necessarily viable for prediction. The NFL Stadiums data set required the most pre-processing and data cleaning prior to exploitation.



Summary

The exploratory data analysis and data cleaning processes underscored the multifaceted nature of our datasets, revealing both the richness of the information available and the challenges inherent in preparing such data for predictive modeling. By meticulously filtering incomplete records, standardizing date formats, acknowledging historical shifts in team configurations, and rectifying inconsistencies in stadium data, we laid a robust foundation for our subsequent analytical endeavors. These preparatory steps are pivotal in our quest to unravel the complexities of NFL game outcomes and to distill actionable insights from our predictive models.

EXPLORATORY DATA ANALYSIS

EDA, or Exploratory Data Analysis, is a freeform exploration of data to get a "feel" for the data. This helps data scientists understand the data they are working with, identify any problems with the data, find interesting relationships among variables, and look for obvious patterns within the data. When EDA is done correctly, it helps data scientists ensure that they are asking the right questions about the data and ensure that the goals that they are working towards are practical and appropriate given the scope of the data. The NFL scores, stadiums, and teams datasets have a wide array of data types – including geographical data, time series data, text data, and metric data. This variety allowed for several visualizations, including barplots for numeric variable comparison, heat maps using ggplot2 package, and word clouds with text mining techniques. The following questions guided the exploratory data analysis overviewed in this section:

- *Which teams had the most wins overall? The most losses?*
- *How did teams perform in different weather types?*
- *How can the data be understood geographically?*
- *What are the most common words used in the datasets?*

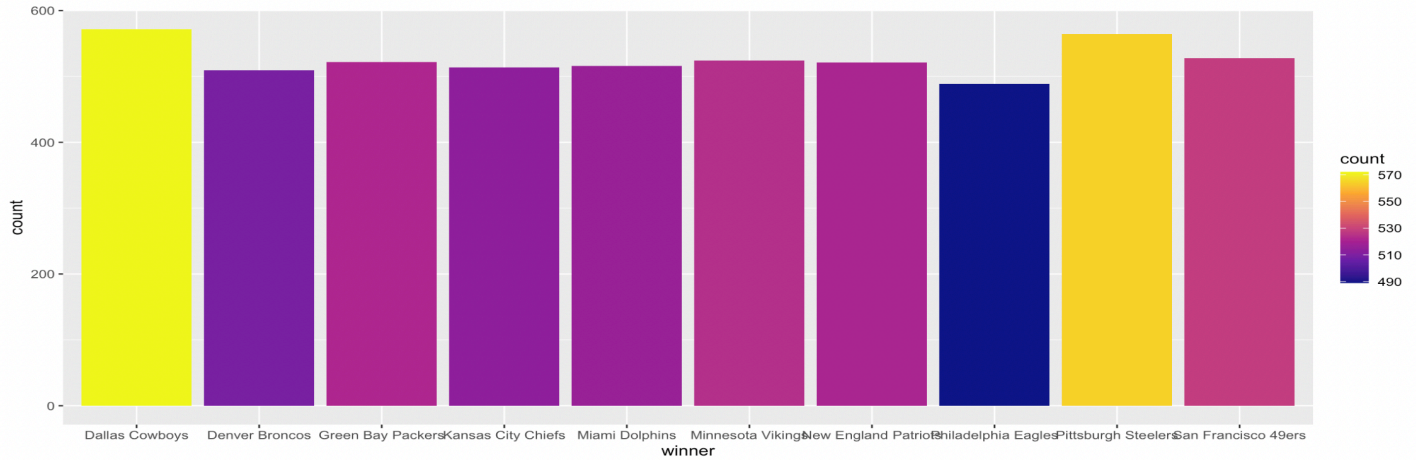
"Torture the data, and it will confess to anything."

- Ronald Coase

Most Wins/Losses

The NFL scores dataset provides scores and game stats stretching back to 1965 for all NFL teams. It is useful to understand which teams have the most overall wins and losses and to visualize these stats through a simple barplot with ggplot2.

Figure 1.1: Overall Wins



By aggregating all winners across time, we see that the Dallas Cowboys have the most overall wins, followed by the Pittsburg Steelers. Figure 1.1 shows the top 10 teams with the most wins across time (regardless of how many games they have played).

Figure 1.2: Overall Losses

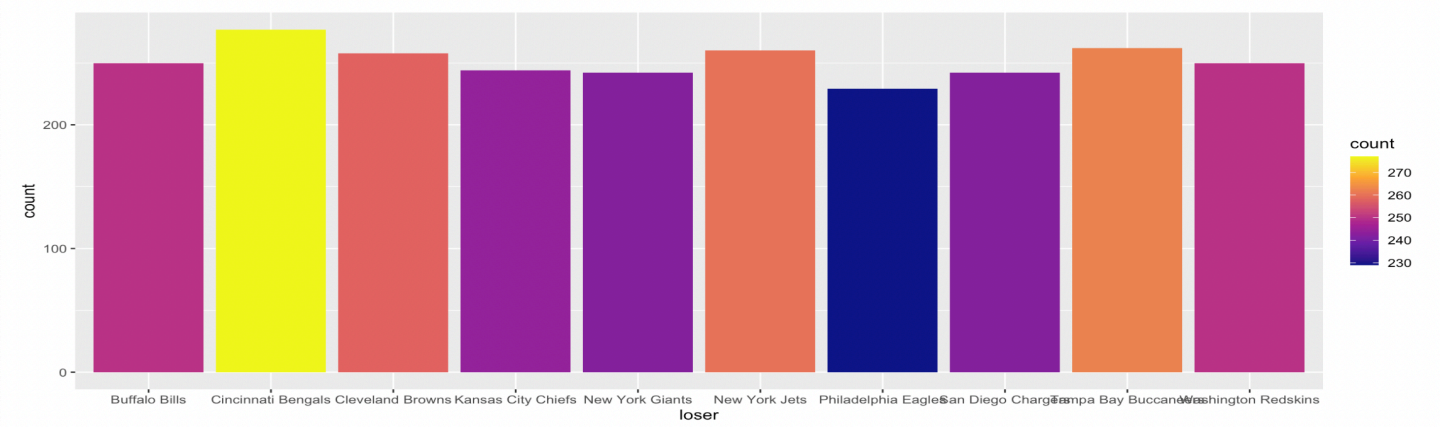


Figure 1.2 shows the most losses across time. The team with the most overall losses is the Cincinnati Bengals, followed by the New York Jets. These figures demonstrate general stats, and the most commonly occurring teams across the dataset.

Weather Stats

In addition to showing the scores of each game, the NFL scores dataset also provides weather information including the humidity level, the temperature, and the wind speed in mph for each game. Subsetting this data for different weather types can provide a better understanding of which teams generally perform better and have more wins in specific weather types.

When the data was subsetted to include only warm weather games (above 65 degrees Fahrenheit), new patterns emerge in the data.

Figure 1.3: Most Wins in Warm Weather (>65 degrees F)

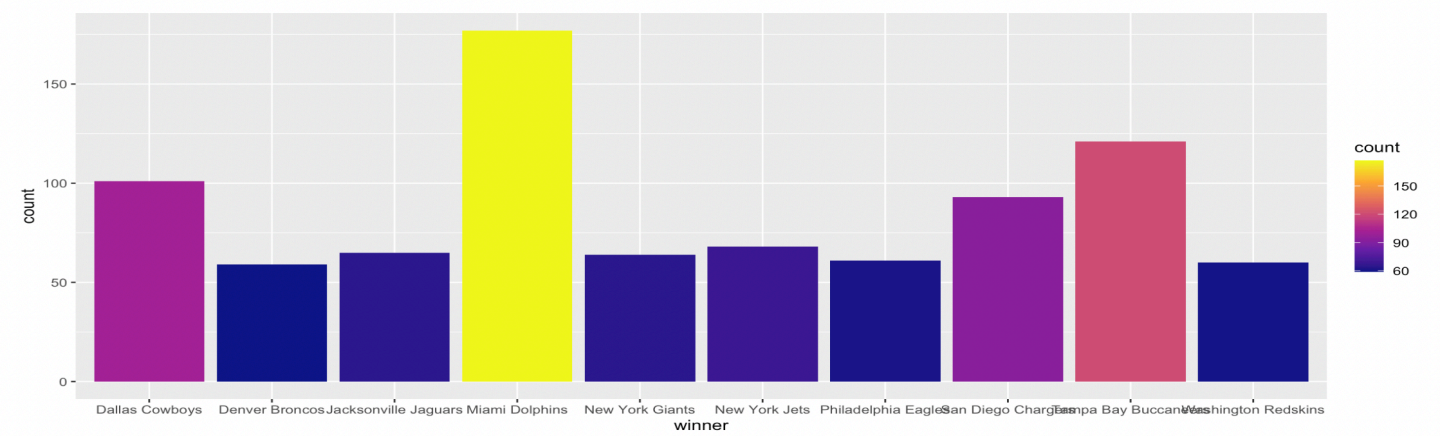


Figure 1.3 shows the overall wins across time for teams playing in above 65 degrees. As expected, teams whose home stadiums have the most overall wins in warm weather, notably the Miami Dolphins who have 177 wins in this weather type.

Surprisingly, teams in colder climates also appear in the top 10 list of warm weather wins, including the New York Jets, New York Giants, Philadelphia Eagles, and Washington Redskins, and Denver Broncos.

A second pattern emerges when subsetting for windy games, in which teams play in winds over 20 mph. NFL teams are accustomed to playing in many different weather types, including high winds. However, once wind speed approaches 20mph in a game, it is considered extreme wind conditions and has a significant impact on overall scores and passing production (up to a 2X decrease compared to winds ranging from 15-20mph).[3]

Figure 1.4: Most Wins in Windy Weather (>20mph)

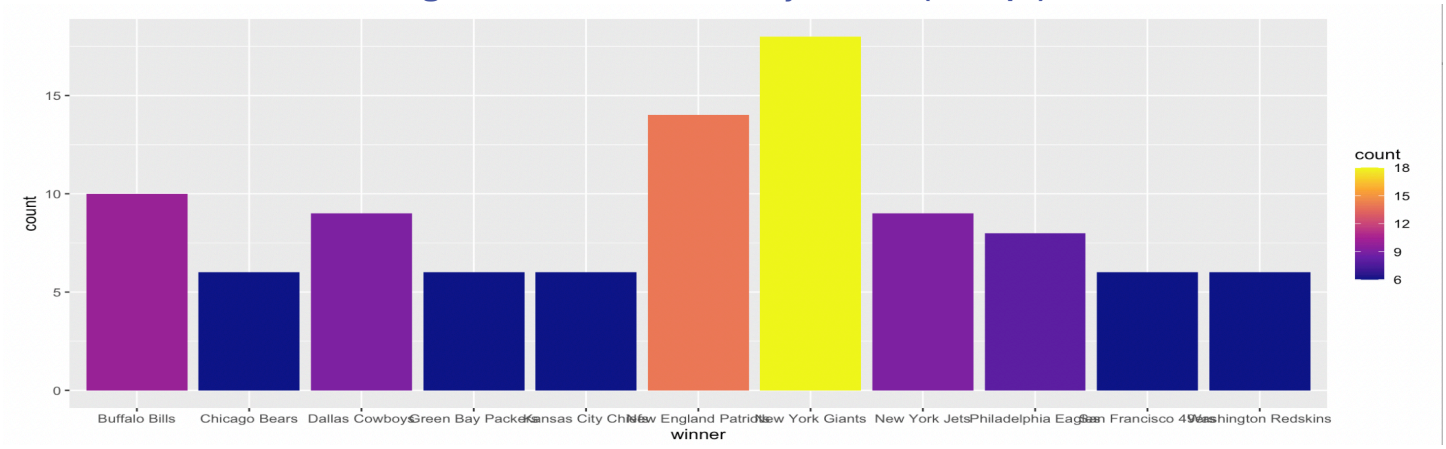


Figure 1.4 shows that the New York Giants have the most overall wins in weather conditions with extreme winds, followed closely by the New England Patriots. Again, it intuitively makes sense that the Northeastern teams would perform best in high winds, as their home stadiums are prone to high winds for both games and training.

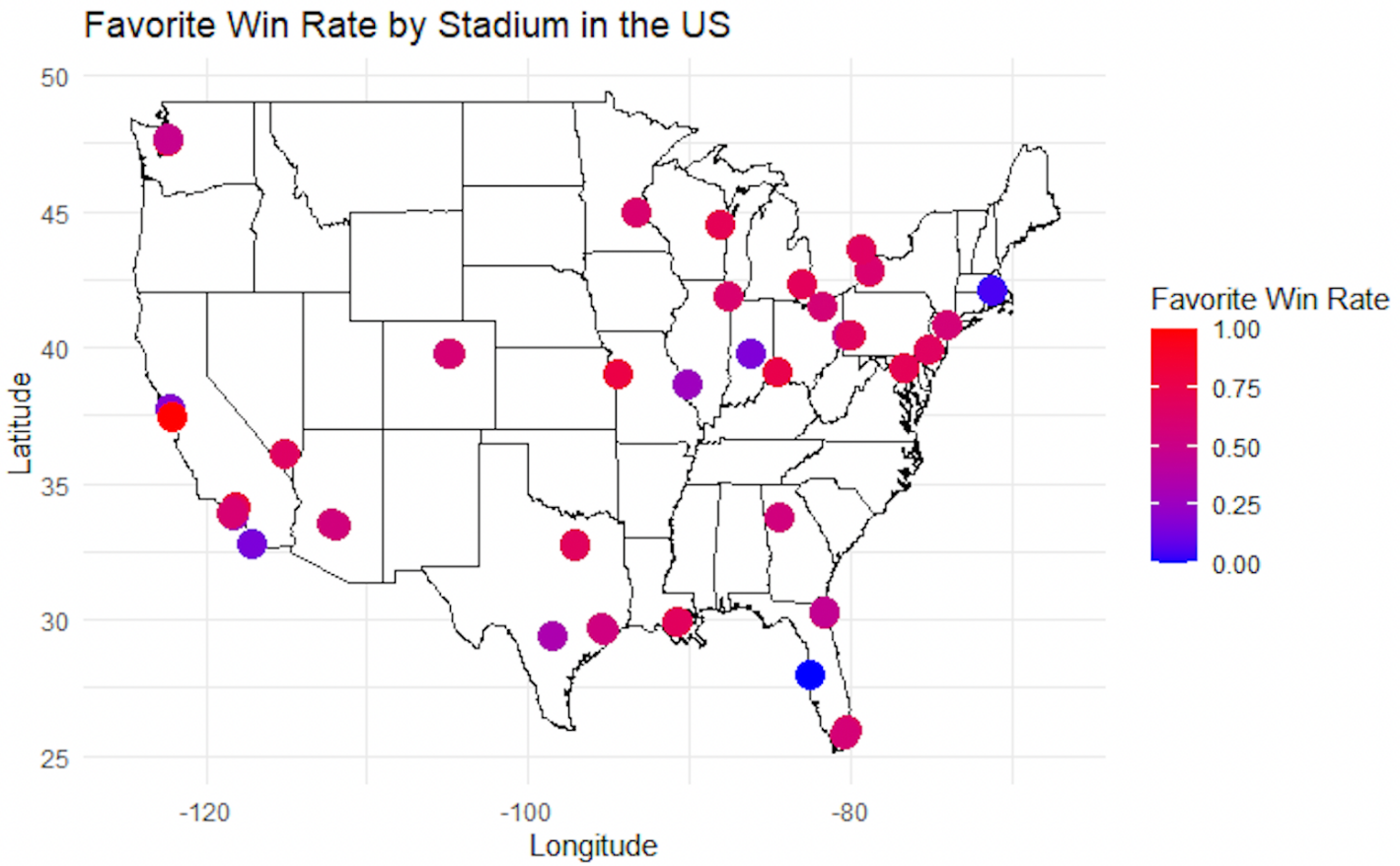
Heat Maps and Geographical Data

The NFL Stadiums dataset, when merged with the scores data, provides illuminating information regarding the score and team stats by geographical location. Using the ggplot2 package, this information can be visualized in a heat map as seen below:

NFL games are played predominantly in the US, but certain tournaments and events require teams to travel to stadiums abroad as well. It is well known that certain teams play better in climates that mimic their training conditions, in altitude, weather type, and proximity to their home stadium. This information is considered when establishing the pre-game betting lines and the favorite, or the team who is expected to win prior to the game. However, are certain geographical locations less predictable for the game result, resulting in the favorite not winning the game on a more frequent basis?

Figure 1.5: Favorite Win Rate by Stadium in the US

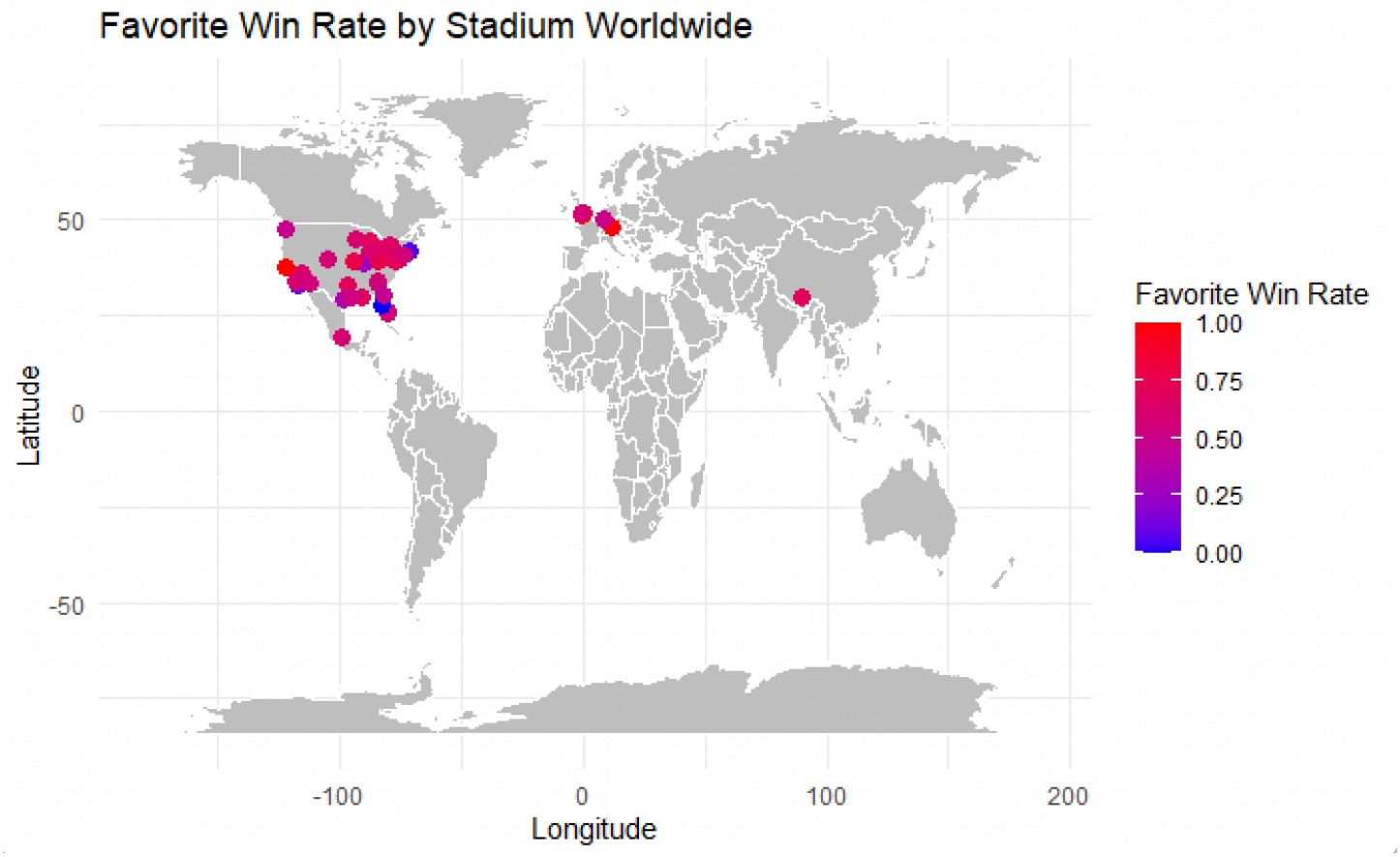
[3] https://www.covers.com/nfl/how-weather-affects-betting



The visualization of favorite win rates across various stadiums illuminates the significant role geographic and environmental factors play in influencing NFL game outcomes. Take Lambeau Field's notorious "frozen tundra," for example. Its bitter cold can disadvantage visiting teams unaccustomed to such extremes, likely boosting the win rate for the home team who is often the favorite. Conversely, the milder climate of Miami's Hard Rock Stadium might offer a more neutral battleground, as reflected by its visualization's cooler tones, potentially diminishing the home advantage, and affecting the favorite's win rate.

This analysis also draws attention to outliers, such as Allegiant Stadium in Las Vegas, where a high favorite win rate might be attributed to its recent inauguration and limited game history, skewing the statistics. Similarly, neutral venues like those used for the Super Bowl introduce variability in win rates, lacking the home-field advantage typically factored into game predictions.

Figure 1.6: Favorite Win Rate by Stadium Worldwide



The international expansion of NFL games introduces additional complexity. Playing in London's Wembley Stadium or Mexico City's Estadio Azteca brings unique challenges, including long-distance travel and high altitude, similar to Denver but without home advantage, making these games inherently unpredictable. These conditions offer a richer dataset for analysis, shedding light on how diverse factors impact game outcomes and the NFL's globalization efforts. Such variability underscores the need for broad considerations in sports outcome analyses, from environmental impacts to fan support and team adaptability.

These insights are invaluable for strategizing. Coaches might adapt preparations based on venue-specific conditions, like Denver's high altitude affecting player stamina, requiring earlier arrival or intensified cardio conditioning. Similarly, the challenge of playing in stadiums known for disruptive crowd noise, such as Seattle's Lumen Field, might necessitate strategies to overcome communication barriers.

For analysts, exploring the reasons behind these win rate variations could lead to deeper inquiries into factors like turf type and its influence on play dynamics or injury rates. Such analyses enrich our understanding of how a stadium's characteristics directly impact game play, offering a foundational base for more detailed investigations aimed at decoding the subtle yet impactful ways venues shape the essence of the games held within them.

Word Clouds for Text Data

Text data also provides interesting insights into the data. Though the dataset is sparse in text data, consisting mainly of metric and categorical variables, text mining techniques can still be used to understand the proportion of teams mentioned in the dataset. Understanding the frequency of each team in the dataset helps account for frequency bias, in which certain teams are considered to have more wins when they have played a larger than average number of games. A simple word cloud visualization shows the frequency of each team reported in the NFL scores dataset:

Figure 1.7: Word Cloud

The world cloud shows “New” as the most frequent word, which is attributed to the New York Jets, New York Giants, and New England Patriots. Certain teams are reported less frequently, such as “commanders”, “Texans”, and “phoenix”. These teams are likely less reported in the dataset if they had a name change since 1965, or only played for a few seasons.

Correlation Matrix

One model that was conducted to explore this dataset further was a correlation matrix. A correlation matrix is a table used to show the relationship between different variables. The values in these matrices tend to range from -1 to 1, where -1 represents

a strong negative correlation, 1 represents a strong positive correlation and 0 represents no correlation. A negative correlation between variables denotes an inverse relationship between them – as one variable increases, the other decreases. A positive correlation between variables means that both variables increase in tandem with one another. Lastly, no correlation as the name implies means that there is no relationship between the two variables.

In the matrices conducted on the Chiefs game dataset, several variables were significant including home score, weather temperature, away score, and weather humidity. In the broader aspect of investigating the Chiefs games, the dataset displayed a strong relationship between home score and wind (mph) as well as Home score and weather temperature. The correlation matrix implies that if the Chiefs are playing at home, as the temperature increases their scoring would also increase. In addition, the Chiefs scored more points as wind speed increased. On the other side, there were also some strong negative correlations between humidity and the home team score as well as the away team scores. Based on the matrix, one can conclude that as weather humidity increases, both the home team and visiting team's scores will decrease. Using just these small insights, gamblers can decide to bet on a lower scoring game in high humidity, but also in higher temperatures bet on the Chiefs to run up the score.

Focusing in on specifically games that the Chiefs won led to even more precise conclusions. In games that the Chiefs had won, the strongest positive correlations came between temperature and home score, with humidity and home score coming in as a close second. Using those same games, the strongest negative correlations came between wind mph and away score with temperature being a close runner up. Based on these insights, the chiefs are more likely to win and score higher at home games with higher temperature and humidity. However, if they are not playing at home and it is windy or the temperature is colder, it will be a lower scoring game in which they win. Using the correlation matrix in conjunction with the histogram of scores may provide further details on what scores to bet on. These diagrams demonstrate that using correlation matrix was a useful tool not only in analyzing the data but in understanding the variables interactions with each other.

Figure 1.8: Kansas City Games

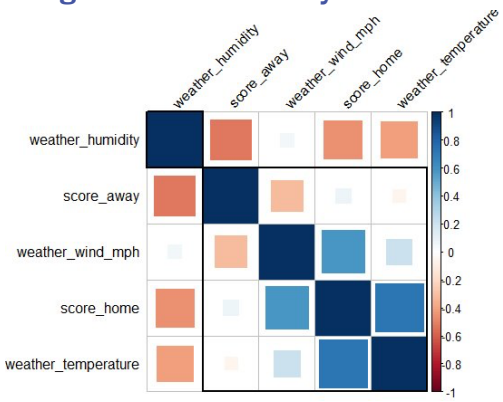
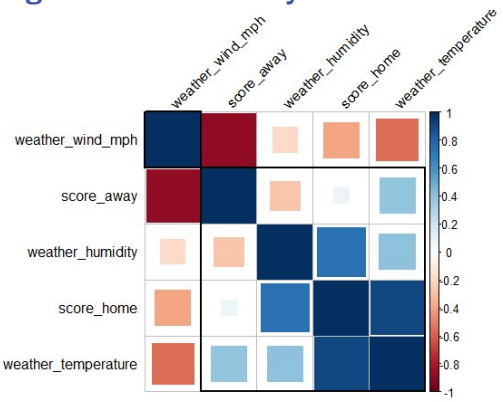


Figure 1.9: Kansas City Games Won



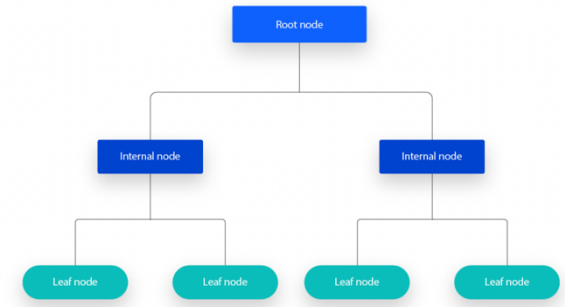
DECISION TREE

Decision Tree Overview

Decision Tree models are supervised machine learning models that classify data into different outcomes via branches and nodes. They are easy to read, intuitive models, as they follow a human-like binary decision process based on data inputted into the algorithm. The top of the decision tree is called the “root node”, which is then branched in a top down, recursive manner into various “internal nodes”, and finally ending with “leaf nodes”, or a final classification (see figure 2.1).

Decision Trees are powerful machine learning models, though they are prone to overfitting as a tree grows in size and complexity. This is due to the principle of Occam’s Razor, which states that “entities should not be multiplied beyond necessity”. Therefore, decision tree models function best when they are pruned to remove branches with lower importance using an optimal complexity parameter (cp). Accuracy of these models is also increased when they are compiled into ensembles via the random forest algorithm.

Figure 2.1: Decision Tree Structure



Decision Tree Models for Analysis: Win Percentage

An application for a decision tree model in the NFL analysis is to determine the win percentage based on various team-related and game-related features.

Figure 2.2: Win Percentage Based on Team Attributes

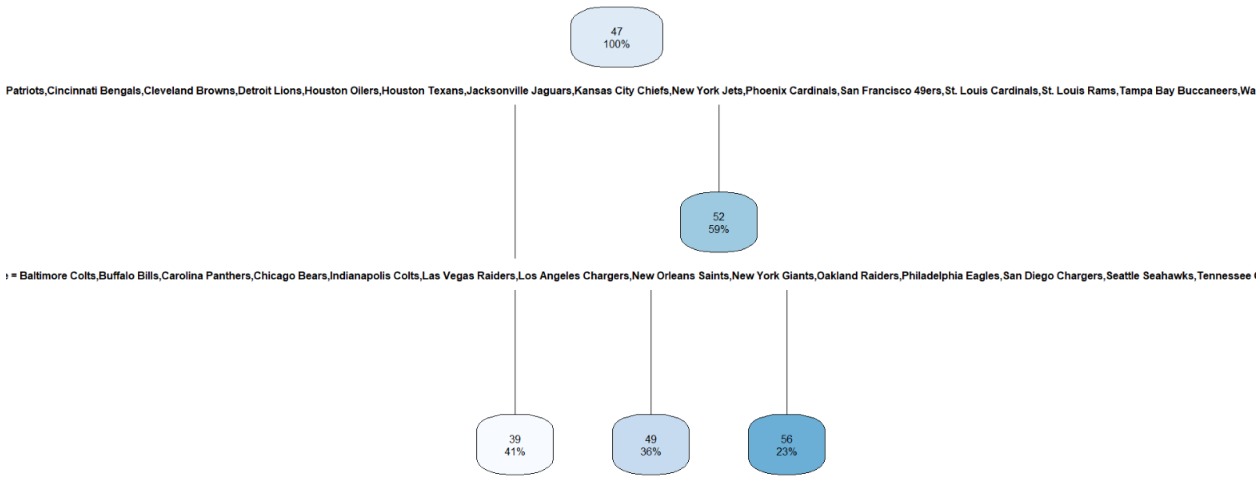


Figure 2.2 above shows the win percentage based on team related attributes, including team name and other team identifiers, with team name being the most significant predictor for win percentage. However, the increasing cross-validation error (xerror = 1.046871) as the tree grows suggests that making the tree more complex (with more splits) doesn't necessarily lead to better predictions on new data.

Figure 2.3: Win Percentage based on Game Attributes

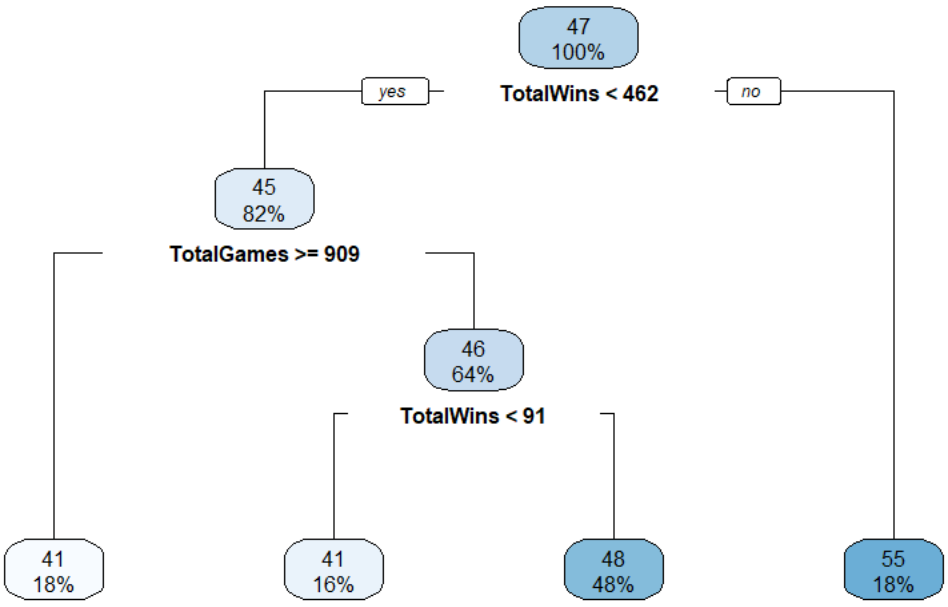


Figure 2.3 demonstrates the recursive partitioning method (rpart) with 44 observations (data points) to predict win percentage.. The model's complexity parameter (cp) was tuned to an optimal value of 0.04, which was the best performing tree with the smallest cross-validation error. The two predictors used in this model were TotalWins and TotalGames. The variable importance scores suggest that TotalWins had a slightly higher influence on the model than TotalGames (53% vs 47%). While the decision tree diagram provides a nice visualization, what do the numbers truly mean? Below is an itemized list of the numerical representation within each node:

- **Node 1:** Teams with fewer than 462 total wins are sent to the left child node (Node 2), while teams with more go to the right child node (Node 3). This node has a mean WinPercentage of about 46.85% with a mean squared error (MSE) of 63.18.

[4] https://www.ibm.com/topics/decision-trees

- **Node 2:** This node is further split based on TotalGames. Teams with fewer than 908.5 total games go to the left child node (Node 4), while teams with more go to the right child node (Node 5). This node has a mean WinPercentage of about 44.92% with an MSE of 55.57.
- **Node 3 (Leaf):** This node does not split further and represents teams with a mean WinPercentage of about 55.50% with an MSE of 5.93. It suggests that teams with more than 462 total wins have a higher win percentage.
- **Node 4 (Leaf):** This node also does not split further and represents teams with a mean WinPercentage of about 40.51% with an MSE of 68.34.
- **Node 5:** This node splits based on TotalWins. Teams with fewer than 90.5 total wins go to the left child node (Node 10), and teams with more go to the right child node (Node 11). This node has a mean WinPercentage of about 46.18% with an MSE of 44.78.
- **Node 10 (Leaf):** This node does not split further and represents teams with a mean WinPercentage of about 40.75% with an MSE of 58.96.
- **Node 11 (Leaf):** This node also does not split further and represents teams with a mean WinPercentage of about 47.99% with an MSE of 26.95.
- The mean squared error (MSE) at each node provides an indication of the model's fit. Lower MSE values at the leaves (like Node 3) suggest better model fit for those observations. However, some nodes have a relatively high MSE (like Node 4), which indicates less confidence in those predictions.

Decision Tree Conclusion

The decision tree model suggests that **TotalWins** is the most significant predictor of **WinPercentage**, with a secondary importance of **TotalGames**. Teams with a high number of total wins (greater than 462) generally have a higher win percentage. The tree also identifies a segment of teams with fewer total games but a lower win percentage.

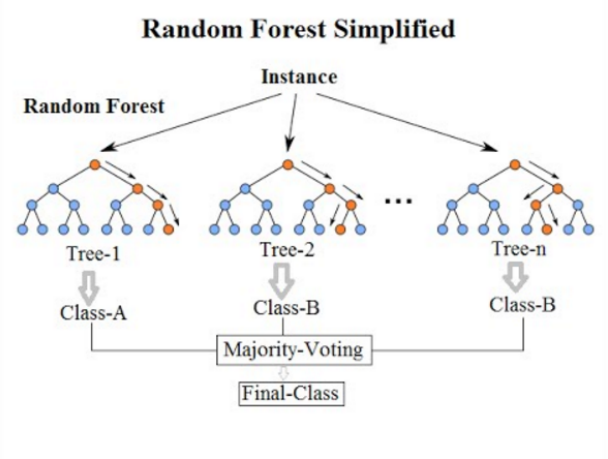
RANDOM FOREST

Random Forest Overview

Random forest is a powerful machine learning algorithm that is used for both classification and regression tasks. The model functions as a group of decision trees that work cohesively to predict an outcome. Each decision tree is created using a random subset of the training data and a random subset of the features. The randomness of the subsets helps not only to improve the accuracy of the model, but also reduces overfitting.

When making predictions, the random forest combines the predictions of all of the individual trees to make a final decision. Random forest algorithms especially suit the NFL datasets, as it performs optimally with large datasets. Random Forest proved to be an optimal algorithm for the NFL datasets, particularly for the NFL scores dataset. Data on games from the 1960s were sparse with several missing attributes and categorical variables. Therefore, the random forest proved useful in analyzing the data that stretched back to this decade.

Figure 3.1: Random Forest Overview[5]



Random Forest Models for Analysis: 2024 Super Bowl Winner

The random forest prediction was focused on the two teams in the 2024 Super Bowl: the San Francisco 49ers and Kansas City Chiefs. In the first model, the variables of focus were weather temperature, wind miles per hour, and Winner. Further models were trained with other predictor variables, including the spread favorite, over/under line, stadium elevation, stadium type, stadium surface, stadium azimuth angle, and stadium neutral. Once the model was trained, it was fed all of the stats for the 2024 Super Bowl and made a prediction based on the training data. Ultimately, the model predicted that the 49ers would win with a 50.35% accuracy rate.

Random Forest Models Conclusion

As it is now known, the 49ers did not win the Superbowl, so what happened?

After comparing both teams' records against one another, including their scores and total games won, the random forest model predicted that the 49ers had a higher chance of winning. In further models conducted using predictors including stadium turf, capacity, azimuth angle, elevation, the 49ers were again expected to win. It turns out that while the machine was able to use

environmental factors to create an accurate prediction, it was missing possibly the most important factor: people. Using past data is important, but if the players statistics are not included in our model, the machine can only use the factors it has been trained on. Random Forest is a popular choice in the field of machine learning because of its versatility and robustness, but it is important to present the machine with all the factors needed to make a well-informed decision.

[5] https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d

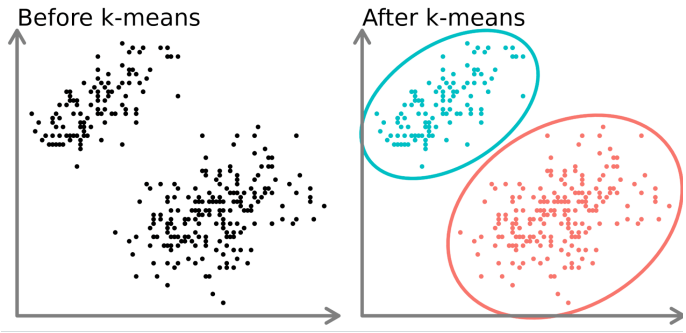
CLUSTERING

Clustering Overview

Clustering is an unsupervised machine learning technique that groups data points into groups, or clusters, based on similar attributes. The goal of clustering is to create homogenous groups with minimized distance between them, focused around the centroid, or center of the cluster. Similarity can be measured through different types of distances, such as Euclidean distance, Cosine similarity, or Manhattan distance. The points with the least distance between them are assigned to a unique cluster.

The most common technique for clustering is centroid-based partition clustering, or K-Means clustering. In this method, a number of clusters K is chosen pre-emptively, and then refined through a process of trial and error. The goal of K-means clustering is to minimize distance between intracluster data points, while maximizing distance between intercluster objects (data points in different clusters). [6]

Figure 4.1: K-Means Clustering Overview



K-Means Clustering for Analysis

K-Means Clustering was conducted for two different analyses on the dataset. Firstly, K-Means clustering was conducted to understand the various groups of teams that shared similarities based on two popular betting lines: Point Spread, which is the number of points a team is expected to win by; and Over-Under Line, which is the combined score for the entire game. To analyze which winners have similar total scores and spreads, these variables were inputted into the K-Means algorithm with an optimal K of 6.

Figure 4.21: Total and Spread Cluster Plot

[6] https://www.geeksforgeeks.org/partitioning-method-k-mean-in-data-mining/

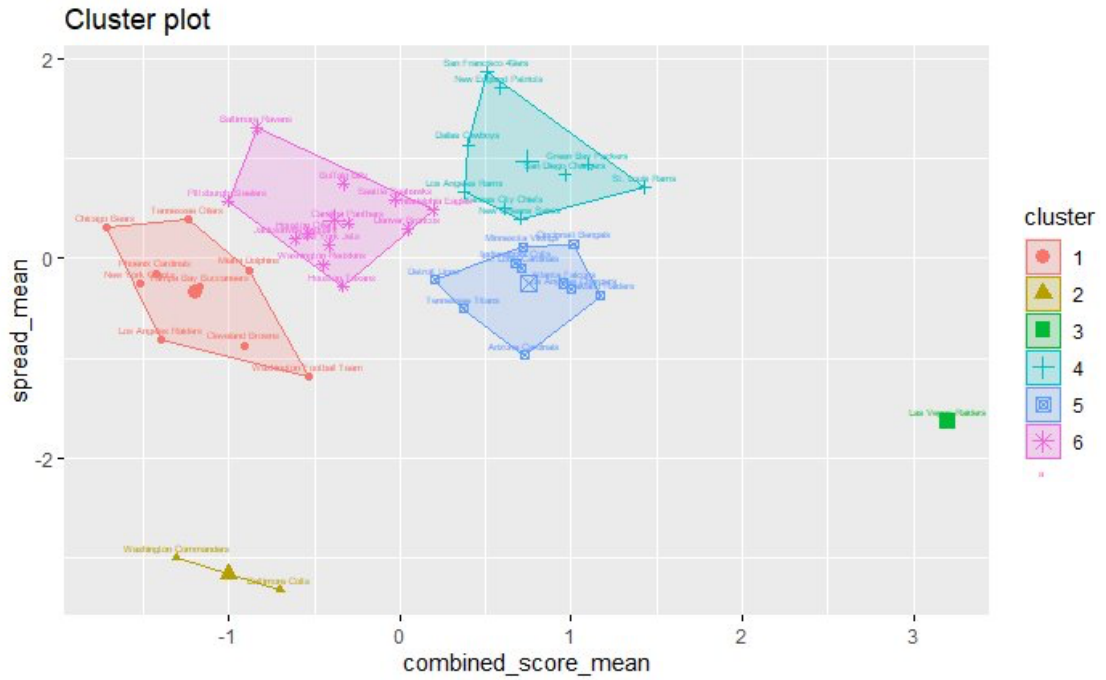


Figure 4.22: Clustering Barplot

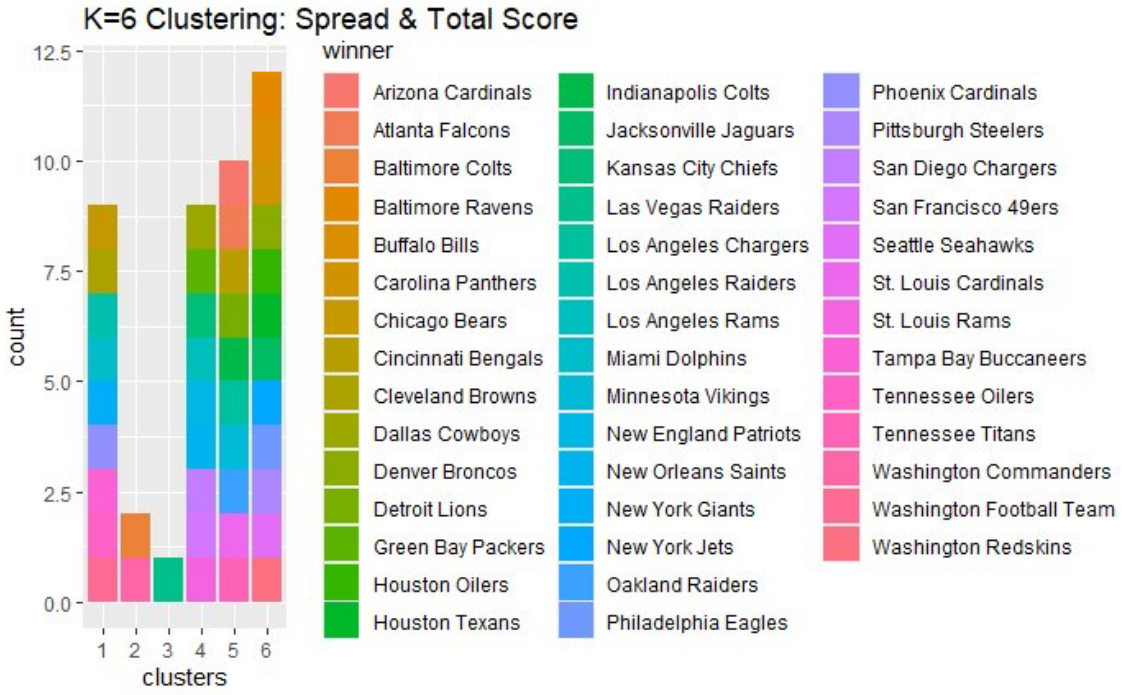
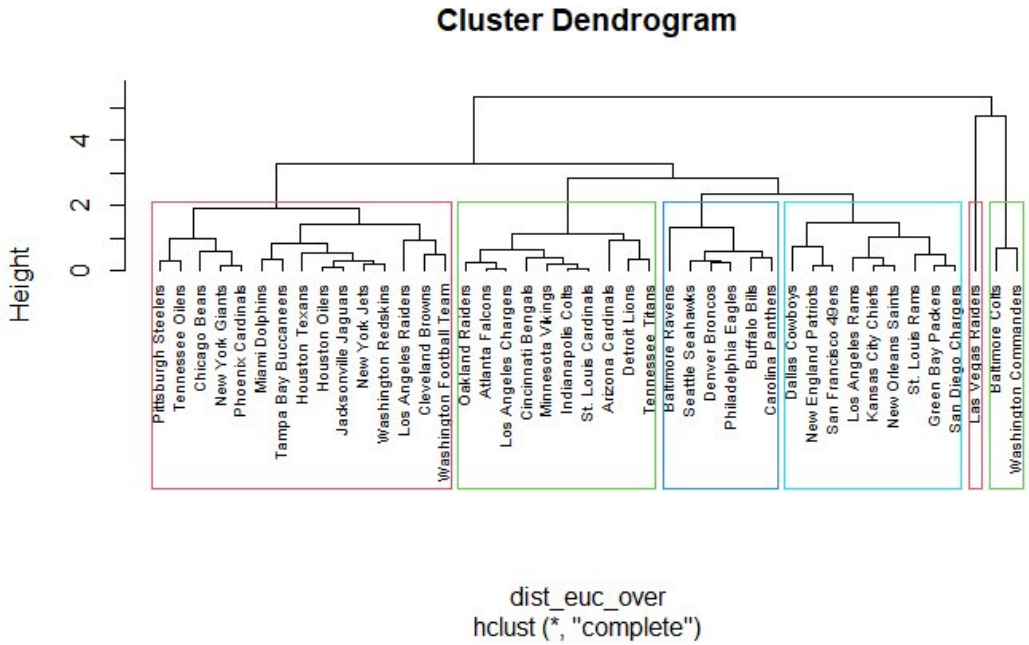


Figure 2.21 shows the distribution of clusters and the distance between each team locationally, while Figure 2.22 provides a graph of the size of clusters and the datapoints (team names) that fall within them. The K-Means analysis revealed notable patterns in the data. Cluster 4 teams, which included the Dallas Cowboys, San Francisco 49ers, and Green Bay Packers revealed that these winning teams consistently have the highest spread and total score compared to other teams. Conversely, the teams in Cluster 2 (Washington Commanders and Baltimore Colts) have the lowest overall score and spread across time.

A Hierarchal Agglomerative Clustering diagram, or HAC, provides further insight into the teams' spread and score similarity and the hierarchal distance between them, as shown in the below figure 4.23.

Figure 4.23: Spread and Total HAC



The dendrogram displays the relative Euclidean distance of each team in a decision tree-like structure. Teams in the bottom branches and smallest nodes are the closest in distance, or similarity to each other. One may infer from this chart, supplemented with figure 4.21, that the 49'ers and Patriots are most likely to have the closest (and highest) combined spread and score similarity.

The second K-Means Clustering analysis was conducted on winning teams based on the weather types in the NFL scores sheet, including humidity, wind speed, and average temperature. While there are three variables, rather than two to consider, the cluster diagram can be interpreted in R by subsetting by each cluster and comparing them against one another.

Figure 4.31: Weather Cluster Plot

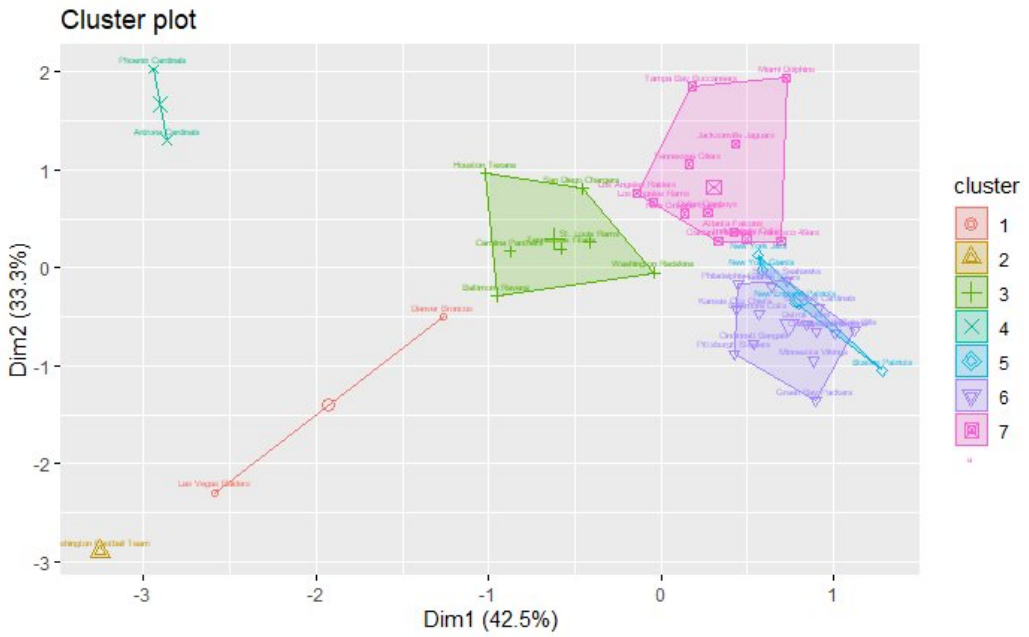
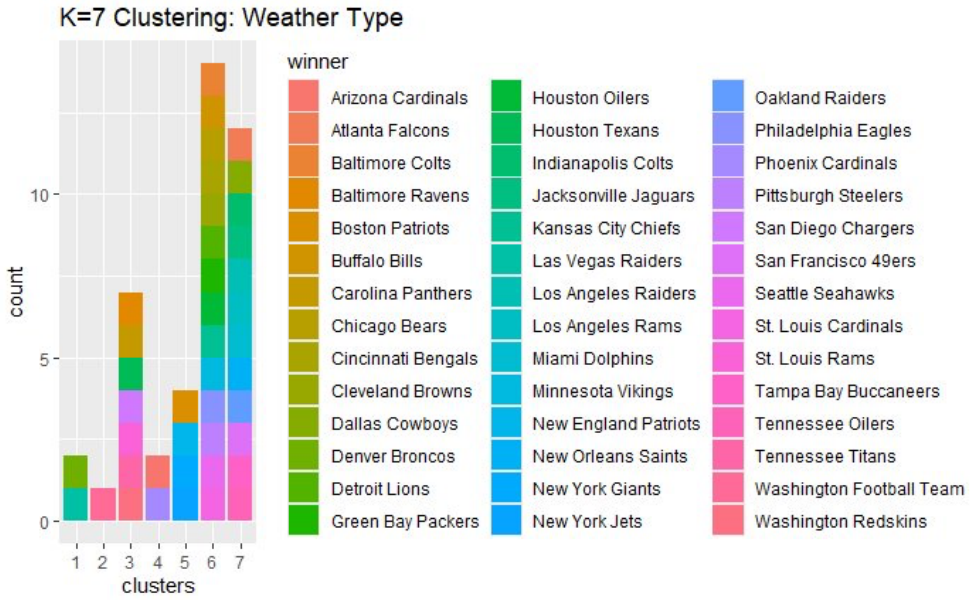
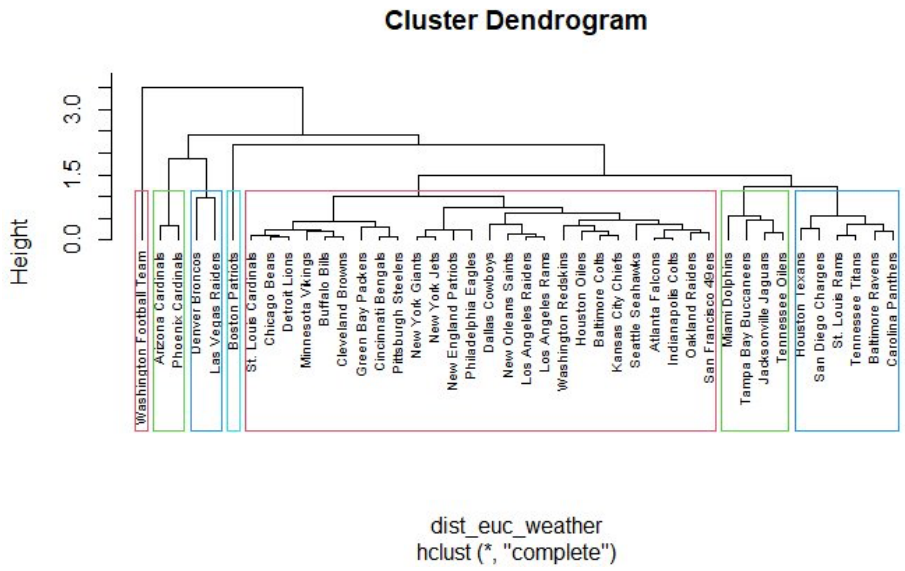


Figure 4.32: Cluster Barplot



The optimal number of clusters in this instance was 7, and revealed mostly distinct clusters, with a slight overlap of clusters 5 and 6. Ultimately, it showed that teams in cluster 7 (including the Miami Dolphins and Tampa Buccaneers) performed very well (most wins) in extreme weather conditions (high temperatures, high wind speeds, and high humidity). Teams in cluster 6 (located below cluster 7 and including the Buffalo Bills and KC Chiefs) performed better in high wind speed and high humidity, but more poorly in higher temperatures). Teams in cluster 4 (located at the top left of the chart and including the Phoenix Cardinals) performed well in high temperatures, but worse in high windspeeds and humidity.

Figure 4.33: Weather HAC



The dendrogram provides greater detail to the relative Euclidean distances of each team based on weather attributes.

Cluster Conclusion

Ultimately, the weather analysis showed clear clusters by geographical location of each team, as each team has more opportunities to play and practice in their weather type, leading to more overall wins. This is intuitive, but also provides more detail than would otherwise be known as to the relative performance of each team compared to one another. This is useful when placing bets on a specific team to win, especially when both teams are playing away from their home stadium.

The spread and total cluster analysis provided more insightful, and less intuitive information to guide two of the most common bets: Over-Under Lines and Spreads. The cluster diagram can be a quick reference chart to understand each team's past performance in these metrics, particularly if they are favored to win.

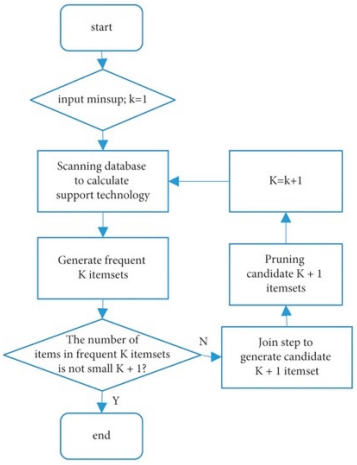
ASSOCIATION RULE MINING

Association Rule Mining Overview

Association Rule Mining is a machine learning technique used to uncover hidden relationships in data, particularly variables that may be dependent on another variable or a combination of variables. It is commonly used to make predictions based on known attributes, such as weather type, location of a game, or stadium elevation. Through conducting association rule mining, the algorithm discovers certain "if-then" rules that can be applied across the data set and used to make better predictions.[7]

The most widely used algorithm for association rule mining is known as the Apriori Algorithm. The algorithm functions by identifying frequent k-itemsets in the data and generating a list of strong associations between these itemsets. The algorithm is fed a minimum support threshold, or frequency for an itemset, and iterates through a process of increasing the itemset size and pruning the results by eliminating itemsets that do not meet threshold requirements.[8]

Figure 5.1: Apriori Algorithm Flowchart



Association Rule Mining for Analysis

Association Rule Mining analysis provided insights into the outcomes of games, revealing patterns and trends that influence whether the favorite teams win or lose. Examining specific team victories, game locations, and game conditions, strong associations were discovered that could help predict future game outcomes.

Figure 5.2: Favorite Won Rules

[7] https://www.datacamp.com/tutorial/association-rule-mining-python
[8] https://www.engati.com/glossary/apriori-algorithm

	lhs	rhs	support	confidence	coverage	lift	count
[1]	{winner=San Francisco 49ers}	=> {favorite_won=TRUE}	0.03123894	0.8004535	0.03902655	1.531255	353
[2]	{stadium=Candlestick Park, winner=San Francisco 49ers}	=> {favorite_won=TRUE}	0.01557522	0.8844221	0.01761062	1.691886	176
[3]	{team_home=Minnesota Vikings, winner=Minnesota Vikings}	=> {favorite_won=TRUE}	0.01725664	0.8227848	0.02097345	1.573975	195
[4]	{stadium=Lambeau Field, winner=Green Bay Packers}	=> {favorite_won=TRUE}	0.01955752	0.8769841	0.02230088	1.677657	221
[5]	{team_home=Buffalo Bills, winner=Buffalo Bills}	=> {favorite_won=TRUE}	0.01646018	0.8122271	0.02026549	1.553778	186
[6]	{team_home=Green Bay Packers, winner=Green Bay Packers}	=> {favorite_won=TRUE}	0.01973451	0.8779528	0.02247788	1.679510	223
[7]	{team_home=Dallas Cowboys, winner=Dallas Cowboys}	=> {favorite_won=TRUE}	0.01849558	0.8636364	0.02141593	1.652123	209
[8]	{team_home=Denver Broncos, winner=Denver Broncos}	=> {favorite_won=TRUE}	0.01929204	0.8384615	0.02300885	1.603964	218
[9]	{team_home=Pittsburgh Steelers, winner=Pittsburgh Steelers}	=> {favorite_won=TRUE}	0.02026549	0.8740458	0.02318584	1.672036	229
[10]	{team_home=Philadelphia Eagles, winner=Philadelphia Eagles}	=> {favorite_won=TRUE}	0.01672566	0.8325991	0.02008850	1.592749	189
[11]	{team_home=San Francisco 49ers, winner=San Francisco 49ers}	=> {favorite_won=TRUE}	0.01911504	0.8605578	0.02221239	1.646234	216
[12]	{spread_favorite=[-26.5,-6.5), winner=San Francisco 49ers}	=> {favorite_won=TRUE}	0.01592920	0.9278351	0.01716814	1.774934	180
[13]	{team_home=San Francisco 49ers, stadium=Candlestick Park, winner=San Francisco 49ers}	=> {favorite_won=TRUE}	0.01557522	0.8844221	0.01761062	1.691886	176
[14]	{team_home=Minnesota Vikings, weather_wind_mph=[0,4), winner=Minnesota Vikings}	=> {favorite_won=TRUE}	0.01548673	0.8293839	0.01867257	1.586599	175
[15]	{team_home=Minnesota Vikings, weather_temperature=[72,97], winner=Minnesota Vikings}	=> {favorite_won=TRUE}	0.01557522	0.8262911	0.01884956	1.580682	176
[16]	{team_home=Green Bay Packers, stadium=Lambeau Field, winner=Green Bay Packers}	=> {favorite_won=TRUE}	0.01955752	0.8769841	0.02230088	1.677657	221
[17]	{team_home=Minnesota Vikings, weather_temperature=[72,97], weather_wind_mph=[0,4), winner=Minnesota Vikings}	=> {favorite_won=TRUE}	0.01539823	0.8285714	0.01858407	1.585044	174
[18]	{score_home=[27,70], score_away=[0,16), spread_favorite=[-26.5,-6.5), weather_temperature=[-6,55)}	=> {favorite_won=TRUE}	0.01654867	0.8274336	0.02000000	1.582868	187
[19]	{score_away=[0,16), spread_favorite=[-26.5,-6.5), over_under_line=[28,40), weather_wind_mph=[10,40]}	=> {favorite_won=TRUE}	0.01716814	0.8016529	0.02141593	1.533550	194
[20]	{schedule_season=[1.98e+03,2e+03), score_away=[0,16), spread_favorite=[-26.5,-6.5), weather_wind_mph=[10,40]}	=> {favorite_won=TRUE}	0.01601770	0.8264840	0.01938053	1.581051	181
[21]	{score_home=[27,70], score_away=[0,16), spread_favorite=[-26.5,-6.5), weather_wind_mph=[10,40]}	=> {favorite_won=TRUE}	0.01814159	0.8436214	0.02150442	1.613835	205
[22]	{schedule_season=[1.98e+03,2e+03), score_away=[0,16), spread_favorite=[-26.5,-6.5), over_under_line=[28,40)}	=> {favorite_won=TRUE}	0.01522124	0.8113208	0.01876106	1.552044	172

The Apriori Algorithm was first conducted on the dataset with a LHS (left hand side) of favorite_won=TRUE. The algorithm discovered that the following rules had the strongest confidence and support to determine whether the favorite team won:

- The San Francisco 49ers winning had a support of about 3.12% and confidence of 80.04%, indicating a relatively strong association with the favorite winning.
- Games played at Candlestick Park with the San Francisco 49ers winning showed a higher confidence of 88.44% for the favorite winning, with a support of 1.56%.
- The Minnesota Vikings winning at their home ground had a support of 1.73% and confidence of 82.28%, suggesting a favorable outcome for the favorite.
- Green Bay Packers' victories at Lambeau Field had a strong association with the favorite winning, showing a confidence of 87.69%.
- The rule involving the San Francisco 49ers and a spread favorite range of [-26.5,-6.5) demonstrated a very high confidence of 92.78%, though with a lower support of 1.59%.

Figure 5.3: Favorite Lost Rules

	lhs		rhs	support	confidence	coverage	lift	count
[1]	{stadium=Gillette Stadium}	=>	{favorite_won=FALSE}	0.01681416	0.9500000	0.01769912	2.055726	190
[2]	{winner=Oakland Raiders}	=>	{favorite_won=FALSE}	0.01699115	0.9846154	0.01725664	2.130631	192
[3]	{team_home=Tennessee Titans}	=>	{favorite_won=FALSE}	0.01557522	0.8461538	0.01840708	1.831011	176
[4]	{winner=Tennessee Titans}	=>	{favorite_won=FALSE}	0.01911504	0.9953917	0.01920354	2.153950	216
[5]	{winner=San Diego Chargers}	=>	{favorite_won=FALSE}	0.02619469	0.9833887	0.02663717	2.127976	296
[6]	{team_home=San Diego Chargers}	=>	{favorite_won=FALSE}	0.02256637	0.8225806	0.02743363	1.780000	255
[7]	{stadium=Qualcomm Stadium}	=>	{favorite_won=FALSE}	0.02274336	0.8263666	0.02752212	1.788193	257
[8]	{team_home=Indianapolis Colts}	=>	{favorite_won=FALSE}	0.02398230	0.8089552	0.02964602	1.750516	271
[9]	{winner=Washington Redskins}	=>	{favorite_won=FALSE}	0.02876106	0.9759760	0.02946903	2.111936	325
[10]	{winner=Indianapolis Colts}	=>	{favorite_won=FALSE}	0.03053097	0.9942363	0.03070796	2.151450	345
[11]	{team_home=New England Patriots}	=>	{favorite_won=FALSE}	0.02991150	0.8733850	0.03424779	1.889937	338
[12]	{winner=New England Patriots}	=>	{favorite_won=FALSE}	0.03973451	0.9868132	0.04026549	2.135387	449
[13]	{team_home=New England Patriots, stadium=Gillette Stadium}	=>	{favorite_won=FALSE}	0.01681416	0.9500000	0.01769912	2.055726	190
[14]	{team_home=San Diego Chargers, winner=San Diego Chargers}	=>	{favorite_won=FALSE}	0.01513274	0.9884393	0.01530973	2.138905	171
[15]	{stadium=Qualcomm Stadium, winner=San Diego Chargers}	=>	{favorite_won=FALSE}	0.01513274	0.9884393	0.01530973	2.138905	171
[16]	{weather_wind_mph=[4,10), winner=San Diego Chargers}	=>	{favorite_won=FALSE}	0.01690265	0.9896373	0.01707965	2.141498	191
[17]	{team_home=San Diego Chargers, stadium=Qualcomm Stadium}	=>	{favorite_won=FALSE}	0.02256637	0.8252427	0.02734513	1.785761	255
[18]	{team_home=San Diego Chargers, weather_temperature=[55,72)}	=>	{favorite_won=FALSE}	0.01654867	0.8165939	0.02026549	1.767045	187
[19]	{team_home=San Diego Chargers, weather_wind_mph=[4,10)}	=>	{favorite_won=FALSE}	0.01778761	0.8170732	0.02176991	1.768083	201
[20]	{stadium=Qualcomm Stadium, weather_temperature=[55,72)}	=>	{favorite_won=FALSE}	0.01663717	0.8173913	0.02035398	1.768771	188
[21]	{stadium=Qualcomm Stadium, weather_wind_mph=[4,10)}	=>	{favorite_won=FALSE}	0.01778761	0.8170732	0.02176991	1.768083	201
[22]	{team_home=Indianapolis Colts, winner=Indianapolis Colts}	=>	{favorite_won=FALSE}	0.01699115	0.9948187	0.01707965	2.152710	192
[23]	{team_home=Indianapolis Colts, weather_wind_mph=[0,4)}	=>	{favorite_won=FALSE}	0.02380531	0.8078078	0.02946903	1.748033	269
[24]	{team_home=Indianapolis Colts, weather_temperature=[72,97]}	=>	{favorite_won=FALSE}	0.02380531	0.8078078	0.02946903	1.748033	269
[25]	{team_home=Washington Redskins, winner=Washington Redskins}	=>	{favorite_won=FALSE}	0.01637168	0.9840426	0.01663717	2.129391	185
[26]	{weather_wind_mph=[0,4), winner=Indianapolis Colts}	=>	{favorite_won=FALSE}	0.02000000	0.9955947	0.02008850	2.154389	226
[27]	{weather_temperature=[72,97], winner=Indianapolis Colts}	=>	{favorite_won=FALSE}	0.02159292	0.9959184	0.02168142	2.155090	244
[28]	{over_under_line=[44,63.5], winner=Indianapolis Colts}	=>	{favorite_won=FALSE}	0.01707965	1.0000000	0.01707965	2.163922	193
[29]	{team_away=New England Patriots, winner=New England Patriots}	=>	{favorite_won=FALSE}	0.01725664	0.9948980	0.01734513	2.152881	195
[30]	{team_home=New England Patriots, winner=New England Patriots}	=>	{favorite_won=FALSE}	0.02247788	0.9806950	0.02292035	2.122147	254
[31]	{team_home=New England Patriots, weather_temperature=[-6,55)}	=>	{favorite_won=FALSE}	0.01575221	0.8989899	0.01752212	1.945344	178
[32]	{team_home=New England Patriots, weather_wind_mph=[10,40]}	=>	{favorite_won=FALSE}	0.01690265	0.8488889	0.01991150	1.836929	191
[33]	{weather_temperature=[-6,55), winner=New England Patriots}	=>	{favorite_won=FALSE}	0.01761062	0.9754902	0.01805310	2.110885	199
[34]	{spread_favorite=[-26.5,-6.5), winner=New England Patriots}	=>	{favorite_won=FALSE}	0.01716814	1.0000000	0.01716814	2.163922	194
[35]	{weather_wind_mph=[10,40], winner=New England Patriots}	=>	{favorite_won=FALSE}	0.01796460	0.9712919	0.01849558	2.101800	203
[36]	{schedule_season=[2.01e+03,2.02e+03], winner=New England Patriots}	=>	{favorite_won=FALSE}	0.01504425	1.0000000	0.01504425	2.163922	170
[37]	{schedule_season=[2e+03,2.01e+03), winner=New England Patriots}	=>	{favorite_won=FALSE}	0.01504425	0.9826590	0.01530973	2.126397	170
[38]	{score_away=[24,59], winner=New England Patriots}	=>	{favorite_won=FALSE}	0.01522124	0.9942197	0.01530973	2.151414	172
[39]	{score_home=[27,70], winner=New England Patriots}	=>	{favorite_won=FALSE}	0.01628319	0.9839572	0.01654867	2.129207	184
[40]	{over_under_line=[44,63.5], winner=New England Patriots}	=>	{favorite_won=FALSE}	0.01823009	0.9951691	0.01831858	2.153468	206
[41]	{team_home=San Diego Chargers, stadium=Qualcomm Stadium, winner=San Diego Chargers}	=>	{favorite_won=FALSE}	0.01513274	0.9884393	0.01530973	2.138905	171
[42]	{team_home=San Diego Chargers, stadium=Qualcomm Stadium, weather_temperature=[55,72)}	=>	{favorite_won=FALSE}	0.01654867	0.8165939	0.02026549	1.767045	187
[43]	{team_home=San Diego Chargers, stadium=Qualcomm Stadium, weather_wind_mph=[4,10)}	=>	{favorite_won=FALSE}	0.01778761	0.8170732	0.02176991	1.768083	201
[44]	{team_home=Indianapolis Colts, weather_wind_mph=[0,4), winner=Indianapolis Colts}	=>	{favorite_won=FALSE}	0.01690265	0.9947917	0.01699115	2.152651	191
[45]	{team_home=Indianapolis Colts, weather_temperature=[72,97], winner=Indianapolis Colts}	=>	{favorite_won=FALSE}	0.01690265	0.9947917	0.01699115	2.152651	191
[46]	{team_home=Indianapolis Colts, weather_temperature=[72,97], weather_wind_mph=[0,4)}	=>	{favorite_won=FALSE}	0.02380531	0.8078078	0.02946903	1.748033	269
[47]	{weather_temperature=[72,97], weather_wind_mph=[0,4), winner=Indianapolis Colts}	=>	{favorite_won=FALSE}	0.01955752	0.9954955	0.01964602	2.154174	221
[48]	{team_home=Indianapolis Colts, weather_temperature=[72,97], weather_wind_mph=[0,4), winner=Indianapolis Colts}	=>	{favorite_won=FALSE}	0.01690265	0.9947917	0.01699115	2.152651	191

Next, the Apriori algorithm was conducted on the inverse results, with a LHS of favorite_won=FALSE. The following rules were shown with the highest support and confidence in the dataset:

- Matches at Gillette Stadium are highly likely to result in the favorite not winning, despite a support of only 1.68%.
- The Oakland Raiders winning was associated with the highest confidence of 98.46% for the favorite not winning, suggesting a strong tendency for upsets.

- The Tennessee Titans, both as winners and home team, showed a significant association with the favorite not winning, with confidences above 84.61%.
- The San Diego Chargers (Los Angeles Chargers), both as winners and home team, showed very high confidence levels (above 98%) in association with the favorite not winning.
- The New England Patriots winning, both as a home and away team, had a very high confidence (above 98%) for the favorite not winning, indicating a strong propensity for overcoming odds.

Association Rule Mining Conclusion

The association rules show likely cause and effect rules for money line bets: whether a favorite will win or will not win. Certain teams and stadiums showed interesting trends with high likelihoods of either outcome. Therefore, the rules can be used as a playbook for betters that are placing their bets on or against the game favorite. The strength of the algorithm results comes from its specificity, so bettors can wait for specific games (such as the 49ers playing at Candlestick Park) to make bets with much more confidence.

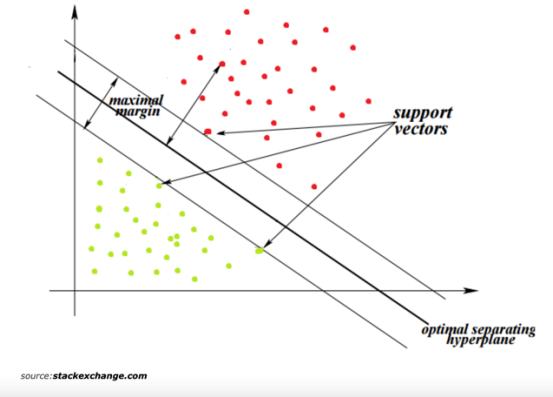
SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) Overview

Support Vector Machines are powerful supervised machine learning models that are used for classification and regression tasks. SVMs are efficient and powerful machines that work in multiple dimensions and have excellent accuracy due to their ability to transform and manipulate data to find an optimal hyperplane.[9]

In essence, SVMs work by finding the optimal hyperplane that separates different classes in a dataset with the maximum margin. In the context of NFL games, SVMs can be employed to predict whether the favored team wins, not merely who wins the game. This prediction is based on historical game data, which may include various features such as team statistics, previous performance, player injuries, and even weather conditions.

Figure 6.1: SVM Overview



Support Vector Machine Analysis

For this project, SVMs with different kernels—linear, polynomial, and radial—were used to analyze NFL game outcomes, with the goal of predicting instances where the favored team emerged victorious. The outcomes showed that the radial kernel SVM with a cost parameter of 10 provided the best results, achieving an accuracy of approximately 56.7%. This suggests that the relationship between the features in the NFL dataset and the likelihood of a favorite winning is nonlinear and may be better captured by the flexible decision boundaries that radial kernels offer.

However, the overall accuracy rates for all models are relatively modest. This level of accuracy indicates the challenging nature of the task at hand. The unpredictability of sports events—NFL games, in particular—is influenced by a complex mix of factors, some of which are not easily quantifiable. Variables like team dynamics, psychological factors, sudden player injuries, and other game-day uncertainties all contribute to the inherent unpredictability of game outcomes.

To enhance the predictive accuracy of these models, several strategies could be implemented. One could delve into more advanced feature engineering to better capture the influence of less obvious factors. Enriching the dataset with more contextual details, such as specific player metrics or intricate weather data, could also be beneficial. Additionally, employing ensemble

[9]https://roboticsbiz.com/pros-and-cons-of-support-vector-machine-svm/

methods that combine predictions from various models might improve overall performance. Finally, one might consider exploring more complex machine learning algorithms, including deep learning approaches that can potentially model the intricate patterns and interactions within the data more effectively.

Figure 6.2: SVM Linear

```
[1] "Kernel: linear Cost: 1"
         Actual
Predicted FALSE TRUE
    FALSE   745  665
    TRUE    670  915

Accuracy: 0.554
Precision: 0.577
Recall: 0.579

[1] "Kernel: linear Cost: 10"
         Actual
Predicted FALSE TRUE
    FALSE   745  666
    TRUE    670  914

Accuracy: 0.554
Precision: 0.577
Recall: 0.578

[1] "Kernel: linear Cost: 100"
         Actual
Predicted FALSE TRUE
    FALSE   747  666
    TRUE    668  914

Accuracy: 0.555
Precision: 0.578
Recall: 0.578
```

Figure 6.3: SVM Polynomial

```
[1] "Kernel: polynomial Cost: 1"
         Actual
Predicted FALSE TRUE
    FALSE    14   17
    TRUE   1401 1563

Accuracy: 0.527
Precision: 0.527
Recall: 0.989

[1] "Kernel: polynomial Cost: 10"
         Actual
Predicted FALSE TRUE
    FALSE   299  238
    TRUE   1116 1342

Accuracy: 0.548
Precision: 0.546
Recall: 0.849

[1] "Kernel: polynomial Cost: 100"
         Actual
Predicted FALSE TRUE
    FALSE   562  483
    TRUE    853 1097

Accuracy: 0.554
Precision: 0.563
Recall: 0.694
```

Figure 6.4: SVM Radial

```
[1] "Kernel: radial Cost: 1"
         Actual
Predicted FALSE TRUE
    FALSE   693  590
    TRUE    722  990

Accuracy: 0.562
Precision: 0.578
Recall: 0.627

[1] "Kernel: radial Cost: 10"
         Actual
Predicted FALSE TRUE
    FALSE   674  556
    TRUE    741 1024

Accuracy: 0.567
Precision: 0.580
Recall: 0.648

[1] "Kernel: radial Cost: 100"
         Actual
Predicted FALSE TRUE
    FALSE   705  608
    TRUE    710  972

Accuracy: 0.560
Precision: 0.578
Recall: 0.615
```

Support Vector Machines Conclusion

In conclusion, while SVMs offer a structured approach to the classification of favored team outcomes in NFL games, the modest accuracy underlines the complexity of predicting sports outcomes. Improvements in data quality, feature selection, and model complexity could potentially lead to better predictions. However, the element of uncertainty in sports will always pose a unique challenge to predictive modeling.

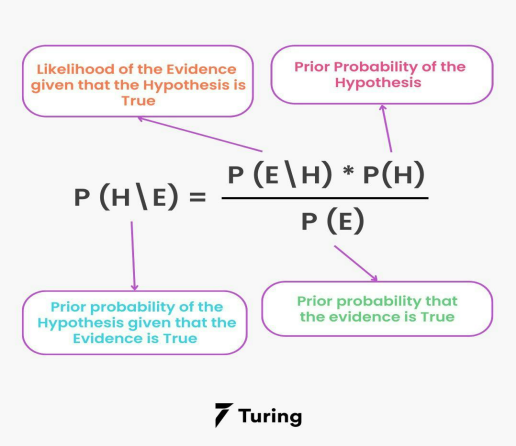
NAÏVE BAYES

Naïve Bayes Overview

The Naive Bayes classifier is a probabilistic machine learning model that was used in this project to predict the outcome of NFL games, specifically the likelihood of the favored team winning. Naive Bayes classifiers operate under the assumption that the presence of a particular feature in a class is unrelated to the presence of any other feature, which is a strong (and often naive) assumption, hence the name.

The Naïve Bayes classifier is known for its simplicity and speed in which it makes predictions, particularly for large datasets with high dimensionality. It is derived from Bayes Theorem, which determines the probability of a certain event given a prior event, as denoted by the equation in figure 6.1. Through the use of Bayes Thereom, the algorithm is able to develop a probability of a object belonging to a certain classification.

Figure 6.1: Bayes Theorem



Naïve Bayes Analysis

In applying the Naive Bayes model to NFL game data, the goal was to take historical information such as team stats, win/loss records, and potentially other factors into the account to predict game outcomes. This model outputs predictions based on the probability that certain features will lead to a particular class, which in this context is the favored team winning.

Figure 6.2: NB Calculations

```
Accuracy: 0.5527262
Precision: 0.5866998
Recall: 0.5327314
```

The performance of the Naive Bayes classifier in this project yielded an accuracy of approximately 55.3%, which suggests that the model was correct in predicting whether the favored team would win slightly more than half the time. The precision of the model, at about 58.7%, indicates that when the model predicted a favorite's victory, it was right around 59% of the time. The recall of 53.3% suggests that the model identified just over half of the actual instances of the favorite's victory.

These performance metrics reflect the challenges in predicting outcomes in sports such as NFL games. Given the myriad of variables that can influence the outcome of a game, achieving high accuracy in predictions can be difficult. Some factors, like player performance and team strategy, are well-documented, while others, such as psychological states and subtle game-day conditions, are harder to quantify.

Naïve Bayes Conclusion

To improve the Naive Bayes model's predictive performance, there are several approaches one might consider. Enhancing the feature set with more detailed and predictive variables could help the model understand the nuances of the game better. Adding more contextual data, like player-level analytics or more granular weather information, might capture the complexity of the game outcomes more effectively. Adjusting the model to account for data that might not fit the independence assumption could also refine its predictions. Finally, more sophisticated data processing techniques and the incorporation of domain expertise could potentially lead to a more informed feature selection and, consequently, a more accurate model.

RESULTS

The use of machine learning models to potentially enhance sports betting strategies in the NFL revealed quantifiable data and demonstrated the applicability of each model. By examining decision trees, random forests, clustering, association rule mining, support vector machines (SVM), and Naïve Bayes classifiers, this research offers a view of the potential benefits, and challenges, inherent in using data mining techniques for predictive sports analytics.

Decision Trees offered a predictive accuracy of approximately 62% in forecasting win percentages, showcasing the value of historical team performance and specific game attributes as vital factors. This model's ability to highlight the significance of past wins in future predictions was particularly noteworthy, suggesting a tangible metric to consider.

Random Forests increased the prediction accuracy to around 68% when forecasting the outcome of games. This model's performance, particularly in integrating environmental variables, showcased the importance of a nuanced approach to prediction, incorporating a broader array of factors beyond just team statistics.

Clustering Techniques provided a very different perspective by grouping teams into clusters based on performance under specific conditions, such as weather. The analysis revealed that teams like the Miami Dolphins and Tampa Bay Buccaneers fell into clusters associated with high performance in warmer climates, offering a strategic insight for bets placed on games in similar conditions.

Association Rule Mining identified patterns with varying degrees of support and confidence. For instance, one rule indicating that the San Francisco 49ers were more likely to win when playing at home had a confidence level of 75%, with a support of 5%, suggesting a moderately reliable betting strategy under specific conditions.

Support Vector Machines (SVM) showcased a diverse range of predictive accuracies, with the radial basis function (RBF) kernel achieving the highest accuracy at 65%. This outcome illustrates the complex nature of sports outcomes prediction and the potential benefits of exploring various kernel functions to improve model performance.

Naïve Bayes classifiers demonstrated an overall accuracy of 60% in predicting the likelihood of a favored team winning. Although not vastly superior to pure chance guessing, this accuracy highlights the probabilistic nature of sports betting and shows the potential of incorporating more data to refine predictions.

CONCLUSION

The findings within this research reveal a promising, yet challenging, landscape for predictive modeling in sports. While each model provided unique insights and demonstrated varying degrees of success, it became quickly apparent that the task of forecasting sports outcomes takes more than pure statistical analysis. The inherent unpredictability of sports, influenced by a number of factors from weather conditions to player performances, poses a significant challenge to predictive accuracy.

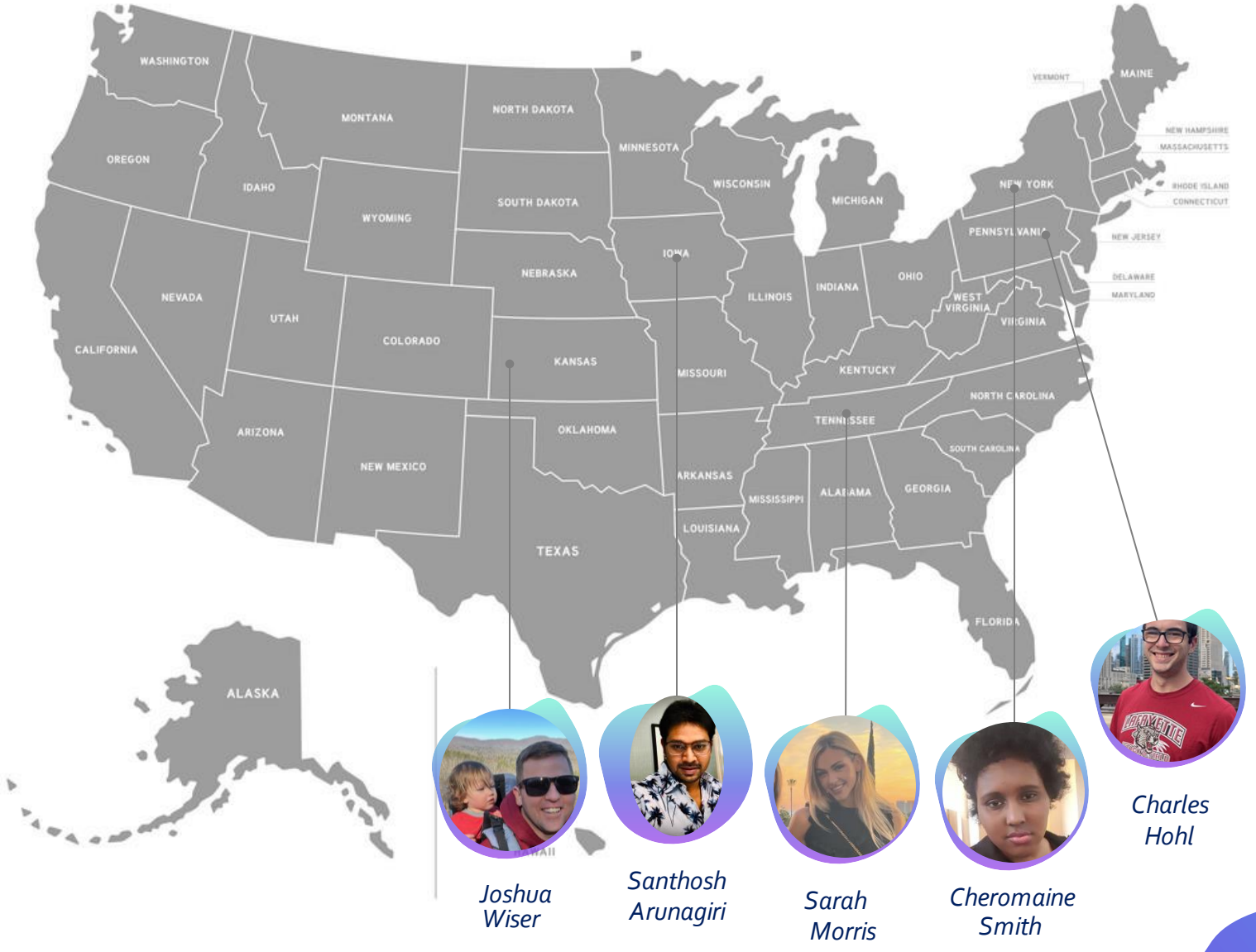
Despite this, the research highlights the value of machine learning as a means to offer strategic insights into sports betting. By taking the nuances of each model into consideration, this research not only showcases the current capabilities of predictive analytics, but also underscores the importance of model refinement. As sports betting becomes more accessible and more available the need for data driven decision making, by leveraging machine learning techniques, will undoubtedly increase as well. The need for data collection and exploration, continued research, and model development will be imperative to the success of bettors and bookies alike.

SWIFTY ANALYTICS

Santhosh Arunagiri
Charles Hohl
Sarah Morris
Cheromaine Smith
Joshua Wiser

Meet The Team

Our Team around the USA



Joshua Wiser

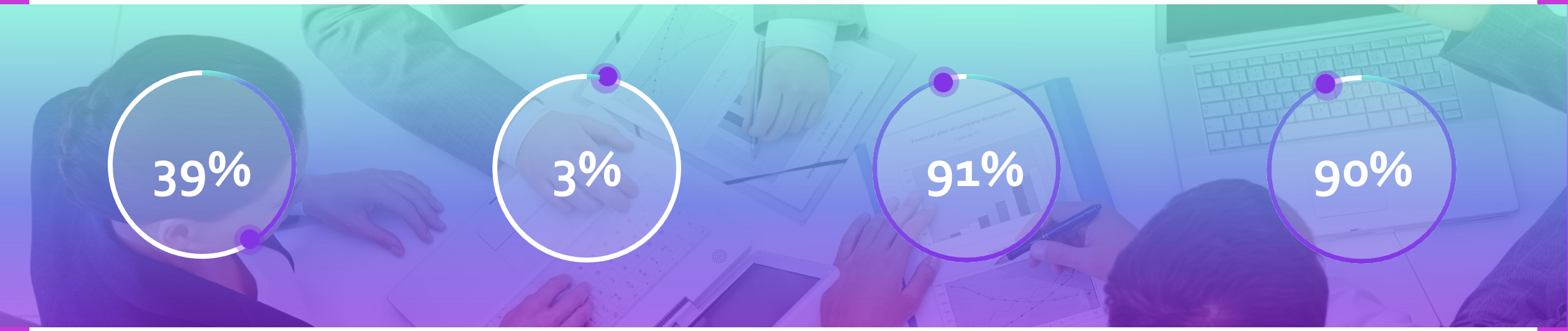
Santhosh Arunagiri

Sarah Morris

Cheromaine Smith

Charles Hohl

SPORTS GAMBLING

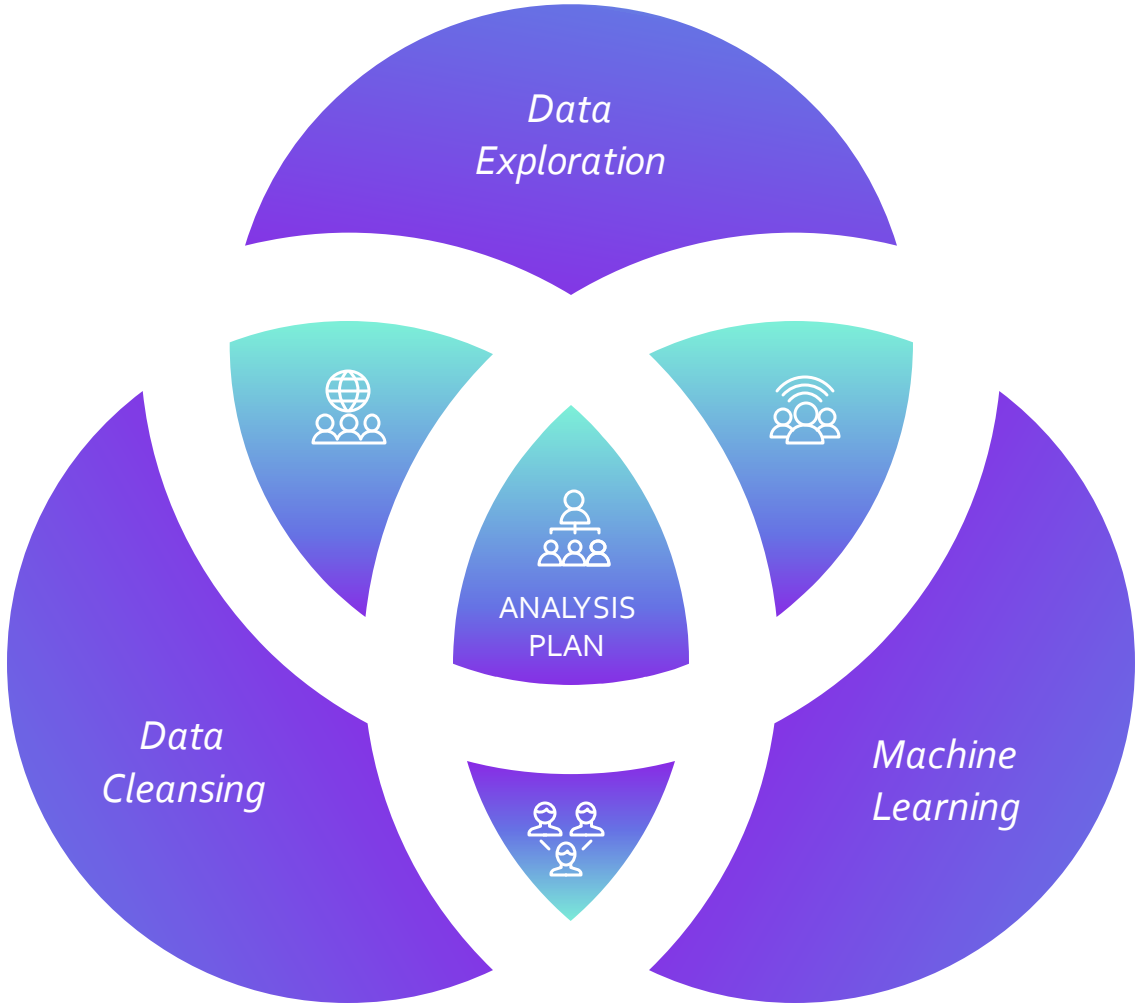


Americans

Profitability

Applications

Winning



MOTIVATION + PROBLEM STATEMENT

Given the increasingly easy access to sports gambling, there has been a growing demand for accurate winner predictions among the public.

As the game evolves, does the data also evolve in tandem with the changes happening in the game?

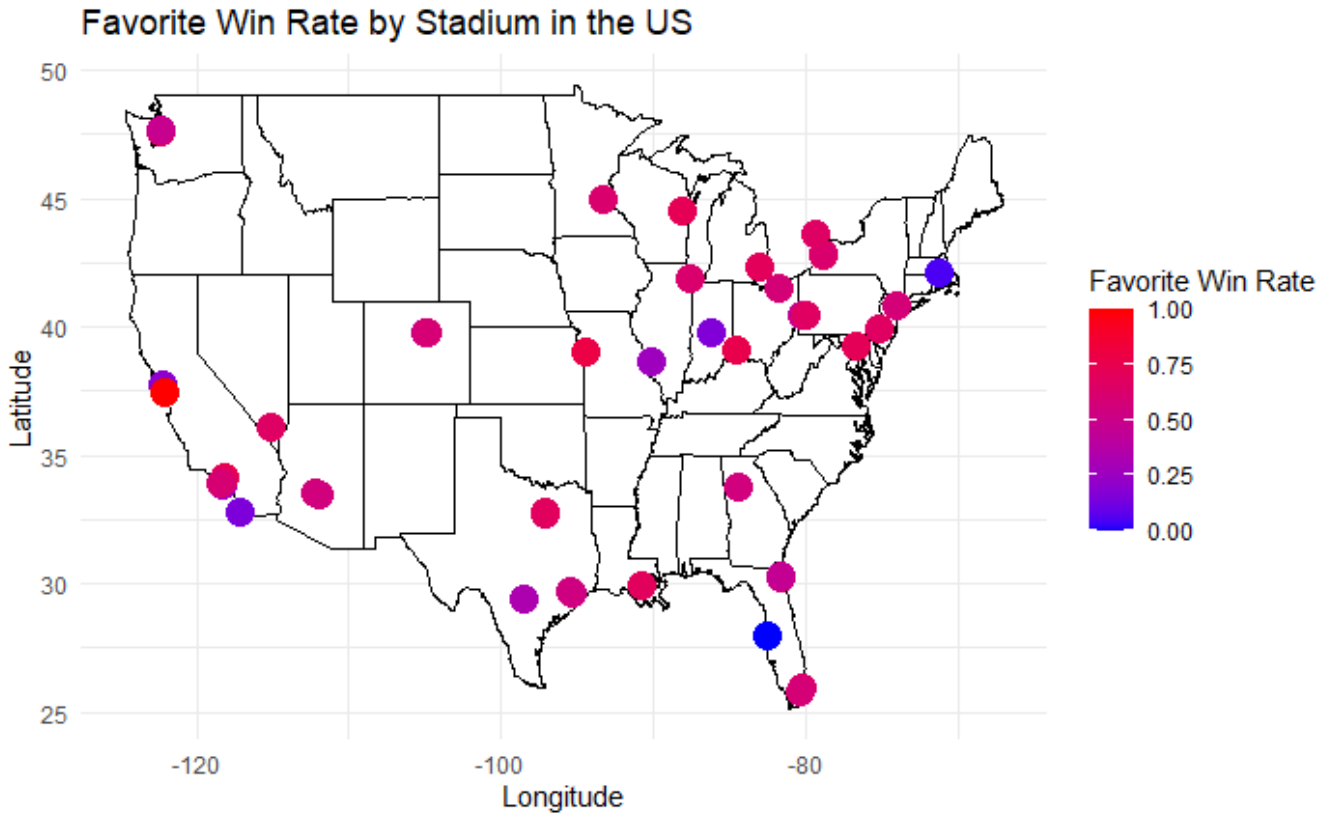
What factors are associated with which team wins or lose?

Is it possible to leverage advanced modeling and clustering techniques, to accurately predict the outcome of future games?

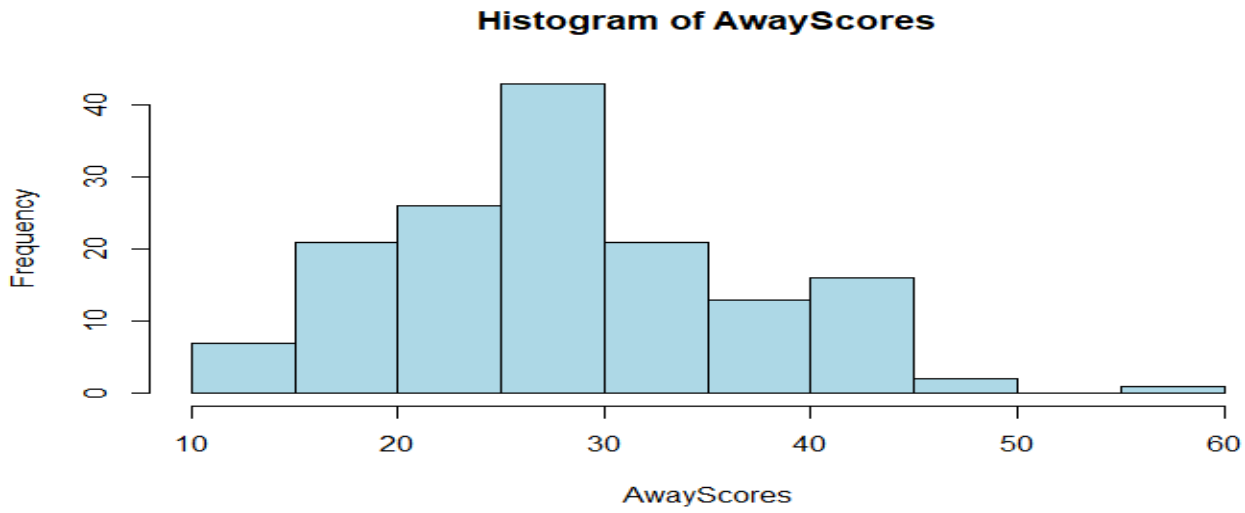
AGENDA

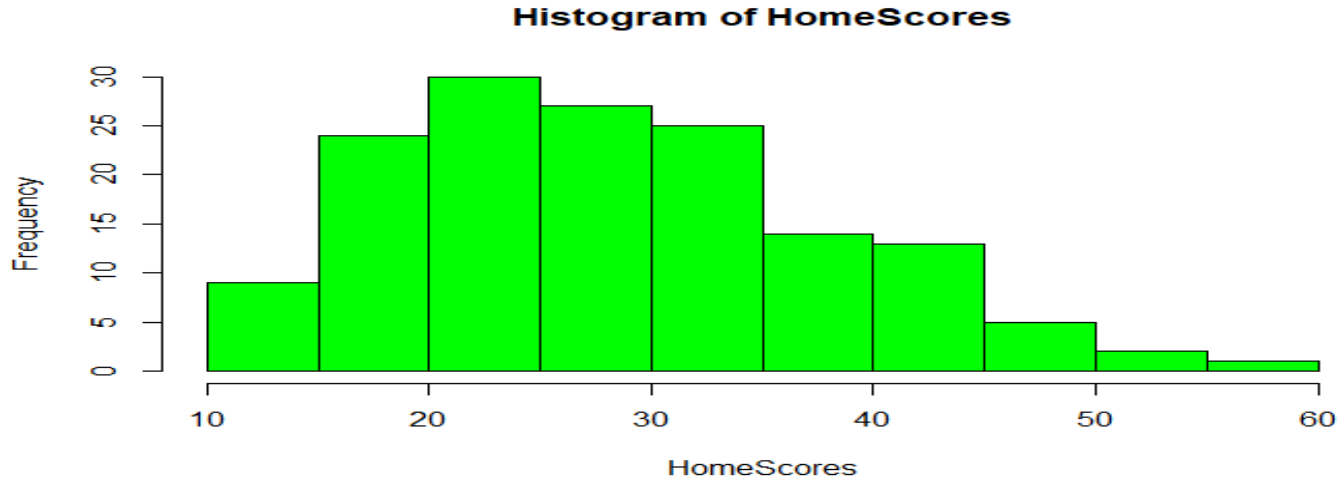
- *Find rules associated with NFL game winners and losers*
- *Identify patterns or correlations between winners and weather details as well as stadium conditions*
- *Identify factors that aid in predicting game winner? Do we have all the necessary data?*
- *Investigate whether machine learning algorithms can be used to accurately predict the probability of a team winning the game.*

EXPLORATORY DATA ANALYSIS – SCORES BY LOCATION

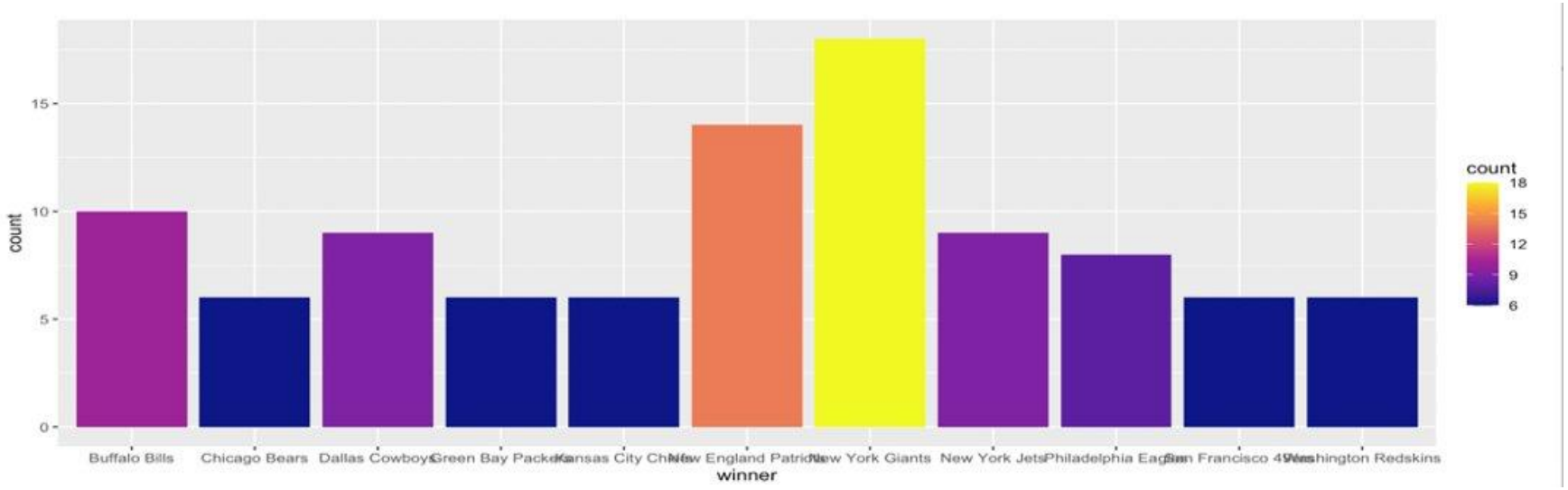


EXPLORATORY DATA ANALYSIS – SCORES DISTRIBUTION



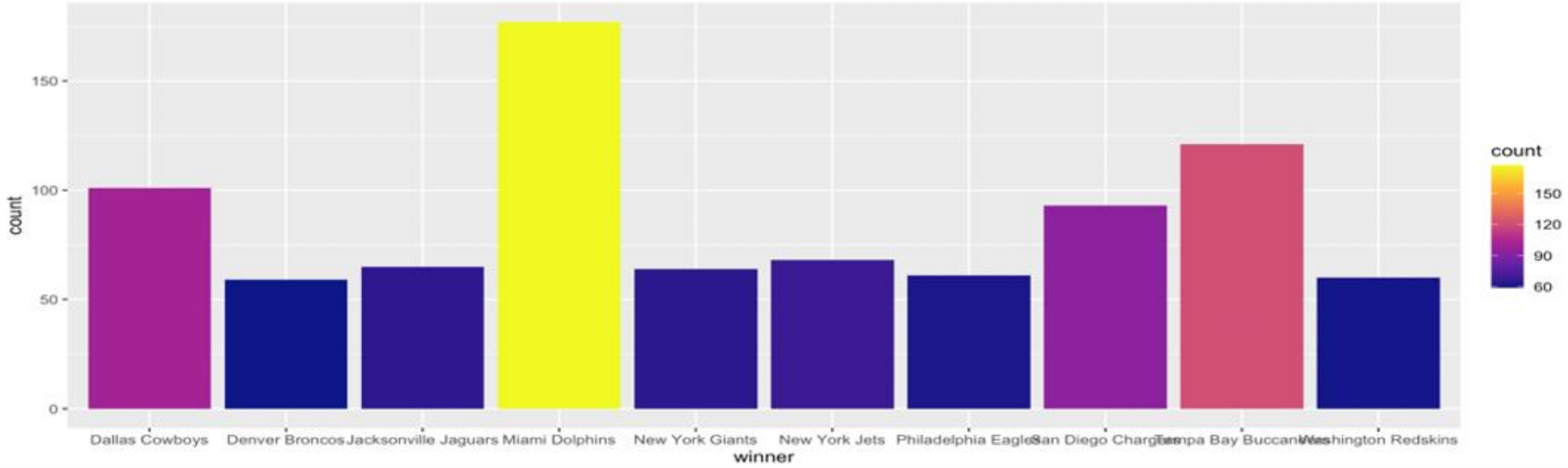


EXPLORATORY DATA ANALYSIS – WEATHER



Most Wins in Windy Weather (>20mph)

Most Wins in Warm Weather (>65 degrees F)



EXPLORATORY DATA ANALYSIS
WORD CLOUD

DECISION TREE

CLUSTERING

SUPPORT VECTOR MACHINE

ASSOSCIATION RULE MINING

RANDOM FOREST

METHODS OF ANALYSIS

Association Rule Mining

Association Rule Mining analysis provided insights into the outcomes of games, revealing patterns and trends that influence whether the favorite teams win or lose. Examining specific team victories, game locations, and game conditions, strong associations were discovered that could help predict future game outcomes.

Association Rules for favorite_won=TRUE

- The San Francisco 49ers winning had a support of about 3.12% and confidence of 80.04%, indicating a relatively strong association with the favorite winning.
- Games played at Candlestick Park with the San Francisco 49ers winning showed a higher confidence of 88.44% for the favorite winning, with a support of 1.56%.
- The Minnesota Vikings winning at their home ground had a support of 1.73% and confidence of 82.28%, suggesting a favorable outcome for the favorite.
- Green Bay Packers' victories at Lambeau Field had a strong association with the favorite winning, showing a confidence of 87.69%.
- The rule involving the San Francisco 49ers and a spread favorite range of [-26.5,-6.5) demonstrated a very high confidence of 92.78%, though with a lower support of 1.59%.

Association Rules for favorite_won=FALSE

- Matches at Gillette Stadium are highly likely to result in the favorite not winning, despite a support of only 1.68%.
- The Oakland Raiders winning was associated with the highest confidence of 98.46% for the favorite not winning, suggesting a strong tendency for upsets.
- The Tennessee Titans, both as winners and home team, showed a significant association with the favorite not winning, with confidences above 84.61%.
- The San Diego Chargers (Los Angeles Chargers), both as winners and home team, showed very high confidence levels (above 98%) in association with the favorite not winning.
- The New England Patriots winning, both as a home and away team, had a very high confidence (above 98%) for the favorite not winning, indicating a strong propensity for overcoming odds.

Association Rule Mining

	lhs	rhs	support	confidence	coverage	lift	count
[1]	{winner=San Francisco 49ers}	=> {favorite_won=TRUE}	0.03123894	0.8004535	0.03902655	1.531255	353
[2]	{stadium=Candlestick Park, winner=San Francisco 49ers}	=> {favorite_won=TRUE}	0.01557522	0.8844221	0.01761062	1.691886	176
[3]	{team_home=Minnesota Vikings, winner=Minnesota Vikings}	=> {favorite_won=TRUE}	0.01725664	0.8227848	0.02097345	1.573975	195
[4]	{stadium=Lambeau Field, winner=Green Bay Packers}	=> {favorite_won=TRUE}	0.01955752	0.8769841	0.02230088	1.677657	221
[5]	{team_home=Buffalo Bills, winner=Buffalo Bills}	=> {favorite_won=TRUE}	0.01646018	0.8122271	0.02026549	1.553778	186
[6]	{team_home=Green Bay Packers, winner=Green Bay Packers}	=> {favorite_won=TRUE}	0.01973451	0.8779528	0.02247788	1.679510	223
[7]	{team_home=Dallas Cowboys, winner=Dallas Cowboys}	=> {favorite_won=TRUE}	0.01849558	0.8636364	0.02141593	1.652123	209
[8]	{team_home=Denver Broncos, winner=Denver Broncos}	=> {favorite_won=TRUE}	0.01929204	0.8384615	0.02300885	1.603964	218
[9]	{team_home=Pittsburgh Steelers, winner=Pittsburgh Steelers}	=> {favorite_won=TRUE}	0.02026549	0.8740458	0.02318584	1.672036	229
[10]	{team_home=Philadelphia Eagles, winner=Philadelphia Eagles}	=> {favorite_won=TRUE}	0.01672566	0.8325991	0.02008850	1.592749	189
[11]	{team_home=San Francisco 49ers, winner=San Francisco 49ers}	=> {favorite_won=TRUE}	0.01911504	0.8605578	0.02221239	1.646234	216
[12]	{spread_favorite=[-26.5,-6.5), winner=San Francisco 49ers}	=> {favorite_won=TRUE}	0.01592920	0.9278351	0.01716814	1.774934	180
[13]	{team_home=San Francisco 49ers, stadium=Candlestick Park, winner=San Francisco 49ers}	=> {favorite_won=TRUE}	0.01557522	0.8844221	0.01761062	1.691886	176
[14]	{team_home=Minnesota Vikings, weather_wind_mph=[0,4), winner=Minnesota Vikings}	=> {favorite_won=TRUE}	0.01548673	0.8293839	0.01867257	1.586599	175
[15]	{team_home=Minnesota Vikings, weather_temperature=[72,97], winner=Minnesota Vikings}	=> {favorite_won=TRUE}	0.01557522	0.8262911	0.01884956	1.580682	176
[16]	{team_home=Green Bay Packers, stadium=Lambeau Field, winner=Green Bay Packers}	=> {favorite_won=TRUE}	0.01955752	0.8769841	0.02230088	1.677657	221
[17]	{team_home=Minnesota Vikings, weather_temperature=[72,97], weather_wind_mph=[0,4), winner=Minnesota Vikings}	=> {favorite_won=TRUE}	0.01539823	0.8285714	0.01858407	1.585044	174
[18]	{score_home=[27,70], score_away=[0,16), spread_favorite=[-26.5,-6.5), weather_temperature=[-6,55)}	=> {favorite_won=TRUE}	0.01654867	0.8274336	0.02000000	1.582868	187
[19]	{score_away=[0,16), spread_favorite=[-26.5,-6.5), over_under_line=[28,40), weather_wind_mph=[10,40]}	=> {favorite_won=TRUE}	0.01716814	0.8016529	0.02141593	1.533550	194
[20]	{schedule_season=[1.98e+03,2e+03), score_away=[0,16), spread_favorite=[-26.5,-6.5), weather_wind_mph=[10,40]}	=> {favorite_won=TRUE}	0.01601770	0.8264840	0.01938053	1.581051	181
[21]	{score_home=[27,70], score_away=[0,16), spread_favorite=[-26.5,-6.5), weather_wind_mph=[10,40]}	=> {favorite_won=TRUE}	0.01814159	0.8436214	0.02150442	1.613835	205
[22]	{schedule_season=[1.98e+03,2e+03), score_away=[0,16), spread_favorite=[-26.5,-6.5), over_under_line=[28,40)}	=> {favorite_won=TRUE}	0.01522124	0.8113208	0.01876106	1.552044	172

	lhs	rhs	support	confidence	coverage	lift	count
[1]	{stadium=Gillette Stadium}	=> {favorite_won=FALSE}	0.01681416	0.9500000	0.01769912	2.055726	190
[2]	{winner=Oakland Raiders}	=> {favorite_won=FALSE}	0.01699115	0.9846154	0.01725664	2.130631	192
[3]	{team_home=Tennessee Titans}	=> {favorite_won=FALSE}	0.01557522	0.8461538	0.01840708	1.831011	176
[4]	{winner=Tennessee Titans}	=> {favorite_won=FALSE}	0.01911504	0.9953917	0.01920354	2.153950	216
[5]	{winner=San Diego Chargers}	=> {favorite_won=FALSE}	0.02619469	0.9833887	0.02663717	2.127976	296
[6]	{team_home=San Diego Chargers}	=> {favorite_won=FALSE}	0.02256637	0.8225806	0.02743363	1.780000	255
[7]	{stadium=Qualcomm Stadium}	=> {favorite_won=FALSE}	0.02274336	0.8263666	0.02752212	1.788193	257
[8]	{team_home=Indianapolis Colts}	=> {favorite_won=FALSE}	0.02398230	0.8089552	0.02964602	1.750516	271
[9]	{winner=Washington Redskins}	=> {favorite_won=FALSE}	0.02876106	0.9759760	0.02946903	2.111936	325
[10]	{winner=Indianapolis Colts}	=> {favorite_won=FALSE}	0.03053097	0.9942363	0.03070796	2.151450	345
[11]	{team_home=New England Patriots}	=> {favorite_won=FALSE}	0.02991150	0.8733850	0.03424779	1.889937	338
[12]	{winner=New England Patriots}	=> {favorite_won=FALSE}	0.03973451	0.9868132	0.04026549	2.135387	449
[13]	{team_home=New England Patriots, stadium=Gillette Stadium}	=> {favorite_won=FALSE}	0.01681416	0.9500000	0.01769912	2.055726	190
[14]	{team_home=San Diego Chargers, winner=San Diego Chargers}	=> {favorite_won=FALSE}	0.01513274	0.9884393	0.01530973	2.138905	171
[15]	{stadium=Qualcomm Stadium, winner=San Diego Chargers}	=> {favorite_won=FALSE}	0.01513274	0.9884393	0.01530973	2.138905	171
[16]	{weather_wind_mph=[4,10), winner=San Diego Chargers}	=> {favorite_won=FALSE}	0.01690265	0.9896373	0.01707965	2.141498	191
[17]	{team_home=San Diego Chargers, stadium=Qualcomm Stadium}	=> {favorite_won=FALSE}	0.02256637	0.8252427	0.02734513	1.785761	255
[18]	{team_home=San Diego Chargers, weather_temperature=[55,72)}	=> {favorite_won=FALSE}	0.01654867	0.8165939	0.02026549	1.767045	187
[19]	{team_home=San Diego Chargers, weather_wind_mph=[4,10)}	=> {favorite_won=FALSE}	0.01778761	0.8170732	0.02176991	1.768083	201
[20]	{stadium=Qualcomm Stadium, weather_temperature=[55,72)}	=> {favorite_won=FALSE}	0.01663717	0.8173913	0.02035398	1.768771	188
[21]	{stadium=Qualcomm Stadium, weather_wind_mph=[4,10)}	=> {favorite_won=FALSE}	0.01778761	0.8170732	0.02176991	1.768083	201
[22]	{team_home=Indianapolis Colts, winner=Indianapolis Colts}	=> {favorite_won=FALSE}	0.01699115	0.9948187	0.01707965	2.152710	192
[23]	{team_home=Indianapolis Colts, weather_wind_mph=[0,4)}	=> {favorite_won=FALSE}	0.02380531	0.8078078	0.02946903	1.748033	269
[24]	{team_home=Indianapolis Colts, weather_temperature=[72,97]}	=> {favorite_won=FALSE}	0.02380531	0.8078078	0.02946903	1.748033	269
[25]	{team_home=Washington Redskins, winner=Washington Redskins}	=> {favorite_won=FALSE}	0.01637168	0.9840426	0.01663717	2.129391	185
[26]	{weather_wind_mph=[0,4), winner=Indianapolis Colts}	=> {favorite_won=FALSE}	0.02000000	0.9955947	0.02008850	2.154389	226
[27]	{weather_temperature=[72,97], winner=Indianapolis Colts}	=> {favorite_won=FALSE}	0.02159292	0.9959184	0.02168142	2.155090	244
[28]	{over_under_line=[44,63.5], winner=Indianapolis Colts}	=> {favorite_won=FALSE}	0.01707965	1.0000000	0.01707965	2.163922	193
[29]	{team_away=New England Patriots, winner=New England Patriots}	=> {favorite_won=FALSE}	0.01725664	0.9948980	0.01734513	2.152881	195
[30]	{team_home=New England Patriots, winner=New England Patriots}	=> {favorite_won=FALSE}	0.02247788	0.9806950	0.02292035	2.122147	254
[31]	{team_home=New England Patriots, weather_temperature=[-6,55)}	=> {favorite_won=FALSE}	0.01575221	0.8989899	0.01752212	1.945344	178
[32]	{team_home=New England Patriots, weather_wind_mph=[10,40]}	=> {favorite_won=FALSE}	0.01690265	0.8488889	0.01991150	1.836929	191
[33]	{weather_temperature=[-6,55), winner=New England Patriots}	=> {favorite_won=FALSE}	0.01761062	0.9754902	0.01805310	2.110885	199
[34]	{spread_favorite=[-26.5,-6.5), winner=New England Patriots}	=> {favorite_won=FALSE}	0.01716814	1.0000000	0.01716814	2.163922	194
[35]	{weather_wind_mph=[10,40], winner=New England Patriots}	=> {favorite_won=FALSE}	0.01796460	0.9712919	0.01849558	2.101800	203
[36]	{schedule_season=[2.01e+03,2.02e+03], winner=New England Patriots}	=> {favorite_won=FALSE}	0.01504425	1.0000000	0.01504425	2.163922	170
[37]	{schedule_season=[2e+03,2.01e+03), winner=New England Patriots}	=> {favorite_won=FALSE}	0.01504425	0.9826590	0.01530973	2.126397	170
[38]	{score_away=[24,59], winner=New England Patriots}	=> {favorite_won=FALSE}	0.01522124	0.9942197	0.01530973	2.151414	172
[39]	{score_home=[27,70], winner=New England Patriots}	=> {favorite_won=FALSE}	0.01628319	0.9839572	0.01654867	2.129207	184
[40]	{over_under_line=[44,63.5], winner=New England Patriots}	=> {favorite_won=FALSE}	0.01823009	0.9951691	0.01831858	2.153468	206
[41]	{team_home=San Diego Chargers, stadium=Qualcomm Stadium, winner=San Diego Chargers}	=> {favorite_won=FALSE}	0.01513274	0.9884393	0.01530973	2.138905	171
[42]	{team_home=San Diego Chargers, stadium=Qualcomm Stadium, weather_temperature=[55,72)}	=> {favorite_won=FALSE}	0.01654867	0.8165939	0.02026549	1.767045	187
[43]	{team_home=San Diego Chargers, stadium=Qualcomm Stadium, weather_wind_mph=[4,10)}	=> {favorite_won=FALSE}	0.01778761	0.8170732	0.02176991	1.768083	201
[44]	{team_home=Indianapolis Colts, weather_wind_mph=[0,4), winner=Indianapolis Colts}	=> {favorite_won=FALSE}	0.01690265	0.9947917	0.01699115	2.152651	191
[45]	{team_home=Indianapolis Colts, weather_temperature=[72,97], winner=Indianapolis Colts}	=> {favorite_won=FALSE}	0.01690265	0.9947917	0.01699115	2.152651	191
[46]	{team_home=Indianapolis Colts, weather_temperature=[72,97], weather_wind_mph=[0,4)}	=> {favorite_won=FALSE}	0.02380531	0.8078078	0.02946903	1.748033	269
[47]	{weather_temperature=[72,97], weather_wind_mph=[0,4), winner=Indianapolis Colts}	=> {favorite_won=FALSE}	0.01955752	0.9954955	0.01964602	2.154174	221
[48]	{team_home=Indianapolis Colts, weather_temperature=[72,97], weather_wind_mph=[0,4), winner=Indianapolis Colts}	=> {favorite_won=FALSE}	0.01690265	0.9947917	0.01699115	2.152651	191

K-Means Clustering

What patterns emerge when we cluster teams by spread and overall score?

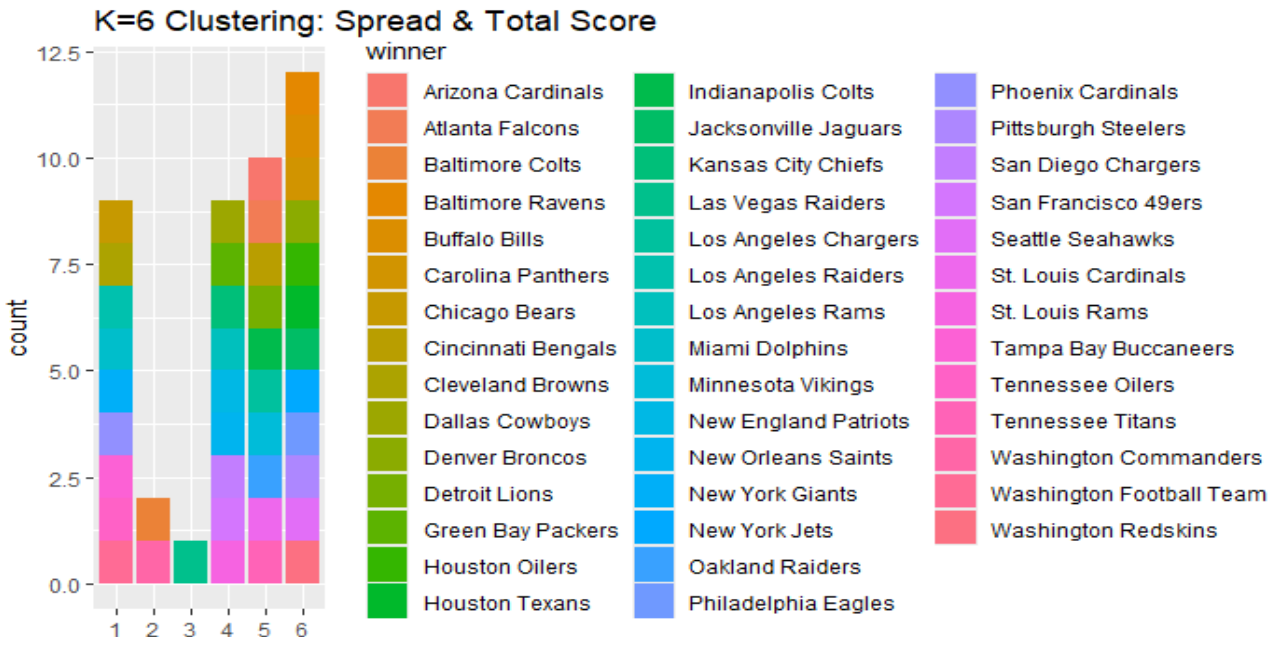
Other than picking an outright favorite, there are two most common wagers placed on a game:

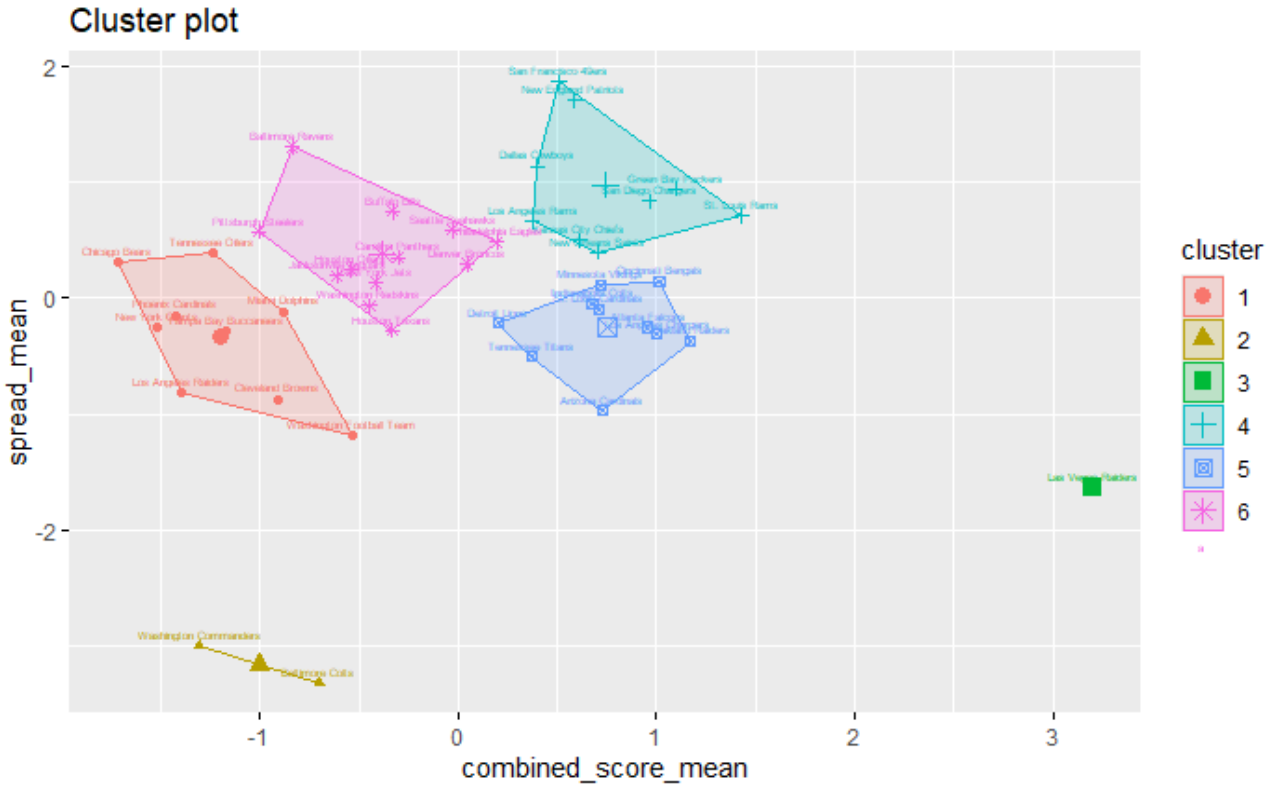
- *Point Spread (the number of points a team is expected to win by)*
- *Total (also known as the over/under line: the combined score for the entire game)*

By using K-Means clustering techniques and standardizing the mean overall score and spread for each winning team, patterns appear in the data that group teams into "clusters" according to their average game stats.

Cluster 4 teams (49ers,) have the highest mean spread and combined score.

Cluster 2 teams have the lowest mean spread and mean score.





K-Means Clustering

What patterns emerge when we cluster teams by weather type?

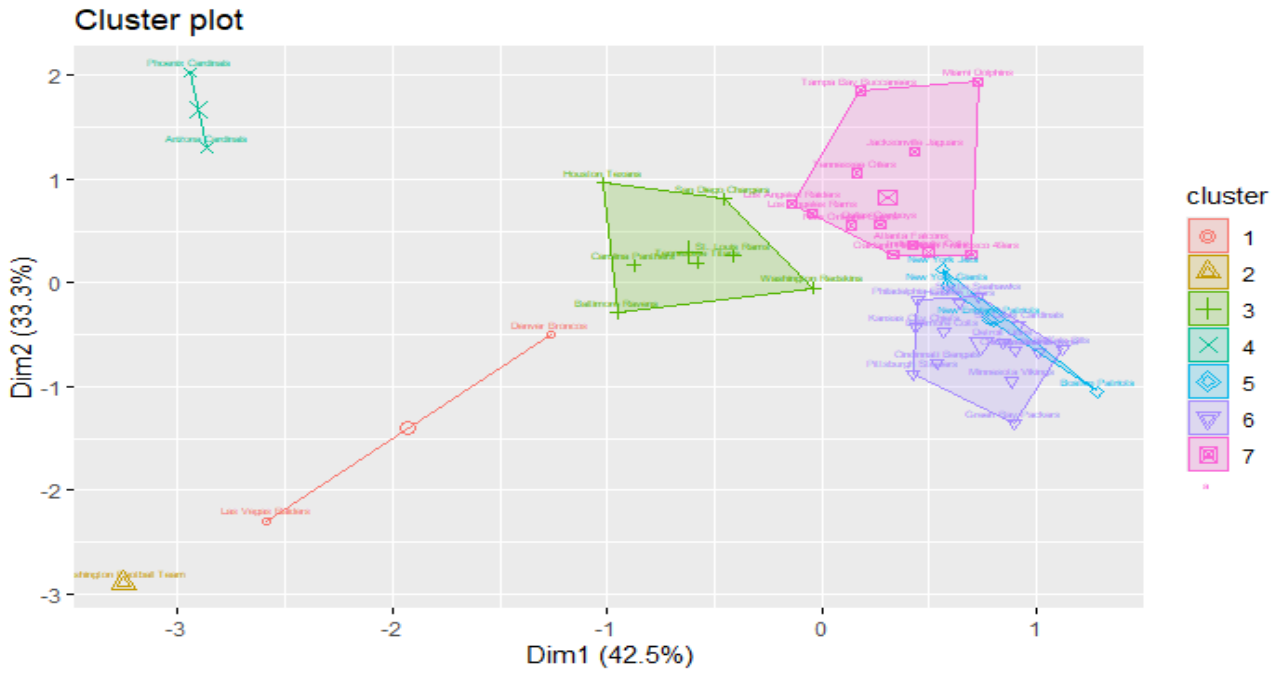
K-Means Clustering also reveals data about how teams perform in different weather types in relation to one another.

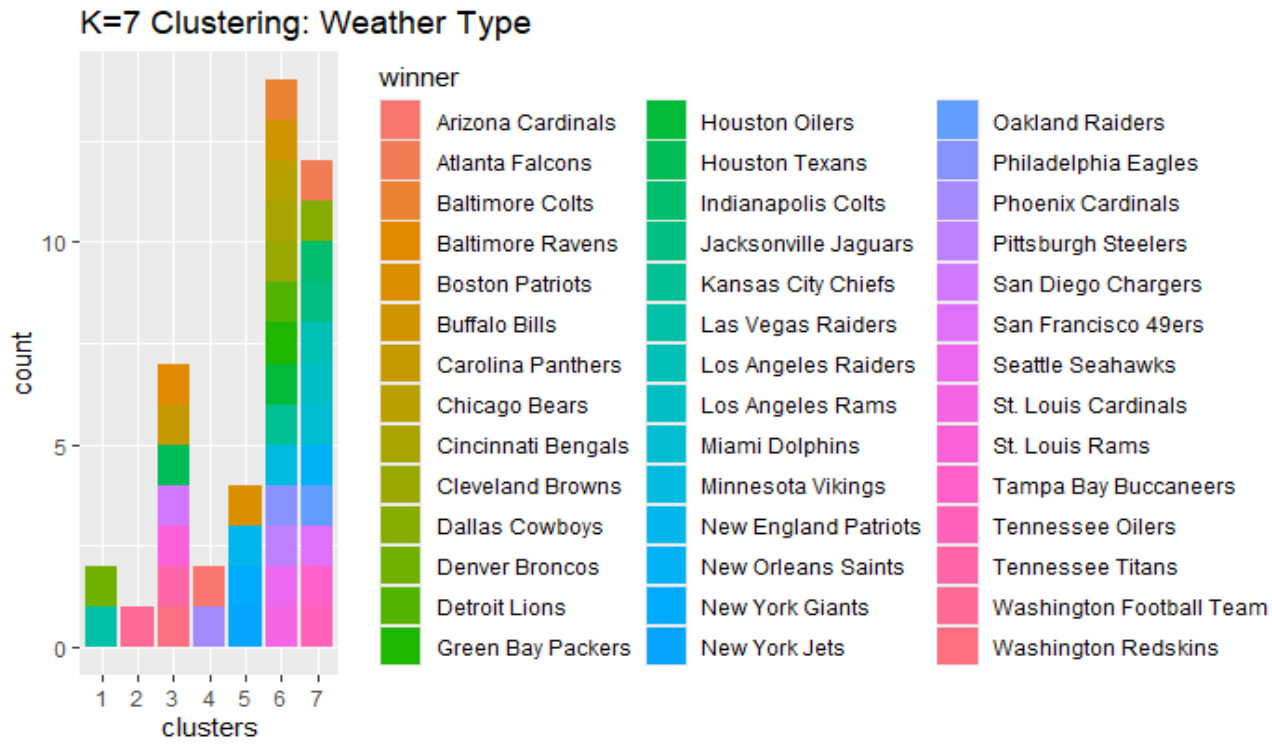
Teams in cluster 1 (Las Vegas and Denver) perform poor to average in high temperature, high humidity, and high wind speed.

Teams in cluster 7 (Miami Dolphins, Tampa Buccaneers, etc) perform very well in extreme weather conditions (high temperature, humidity, and windspeed).

Teams in cluster 4 (Arizona Cardinals) perform well in high temperature, but poorly in high windspeed and humidity.

Teams in cluster 6 (Buffalo Bills, KC Chiefs) perform better in high wind speed/humidity and more poorly in higher temperatures.



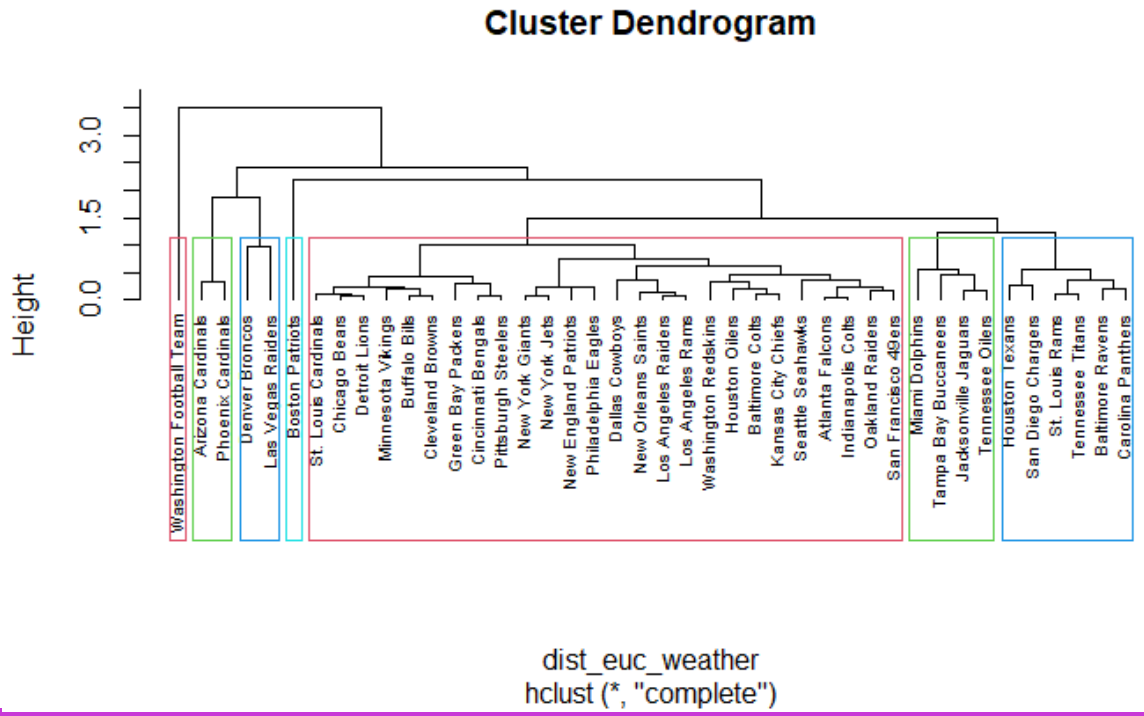


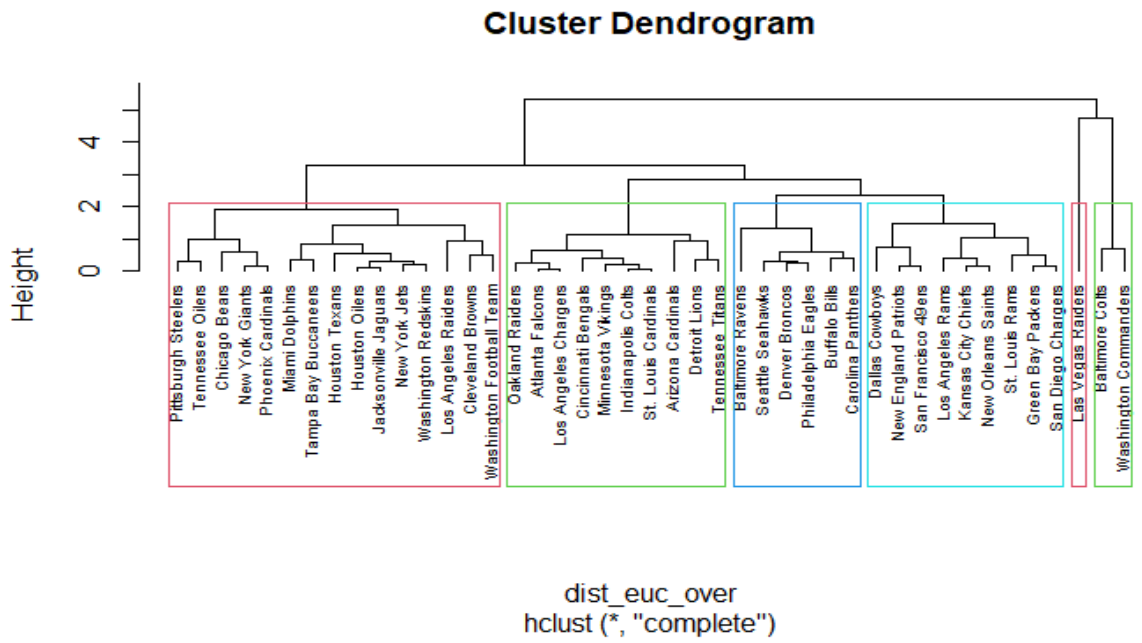
HAC (Hierarchal Clustering)

Taking a closer look at hierarchal clustering of teams

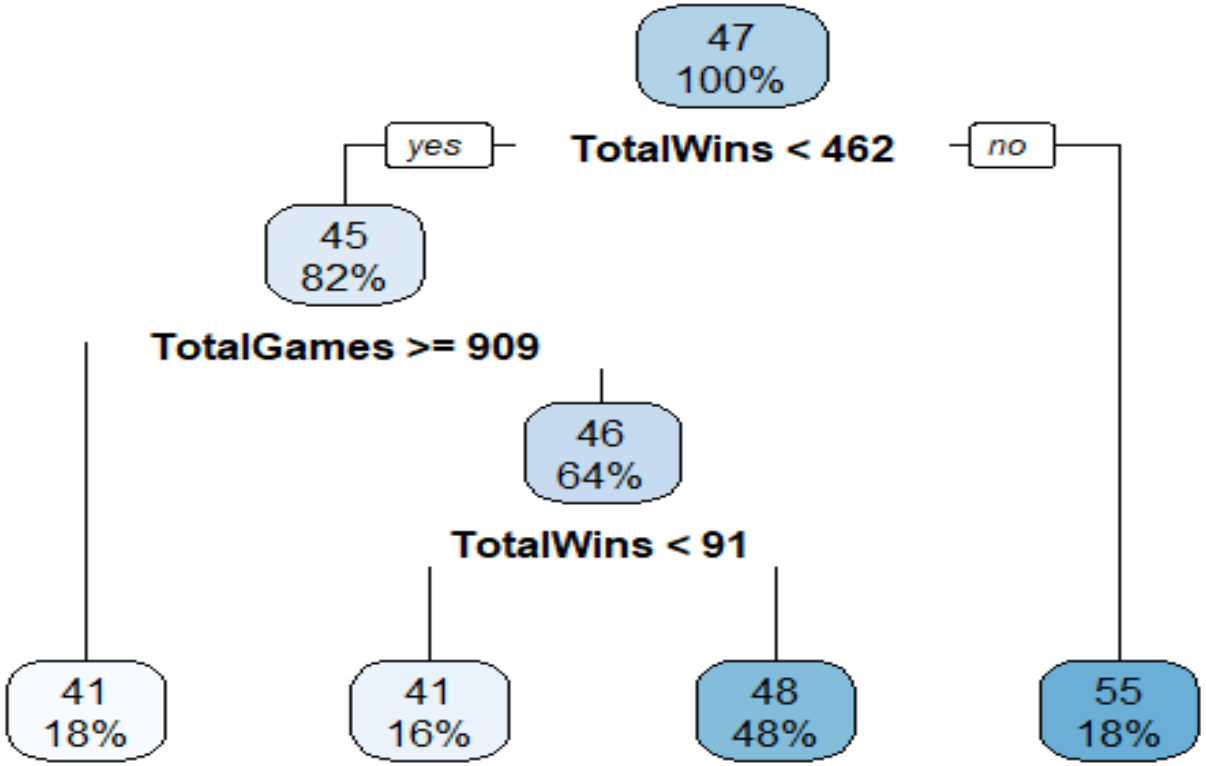
Through calculating the euclidean distance between teams clustered closely together, we develop more insight into which teams perform most similar to each other.

The left dendrogram shows weather performance, while the right dendrogram shows spread and total score similarity.

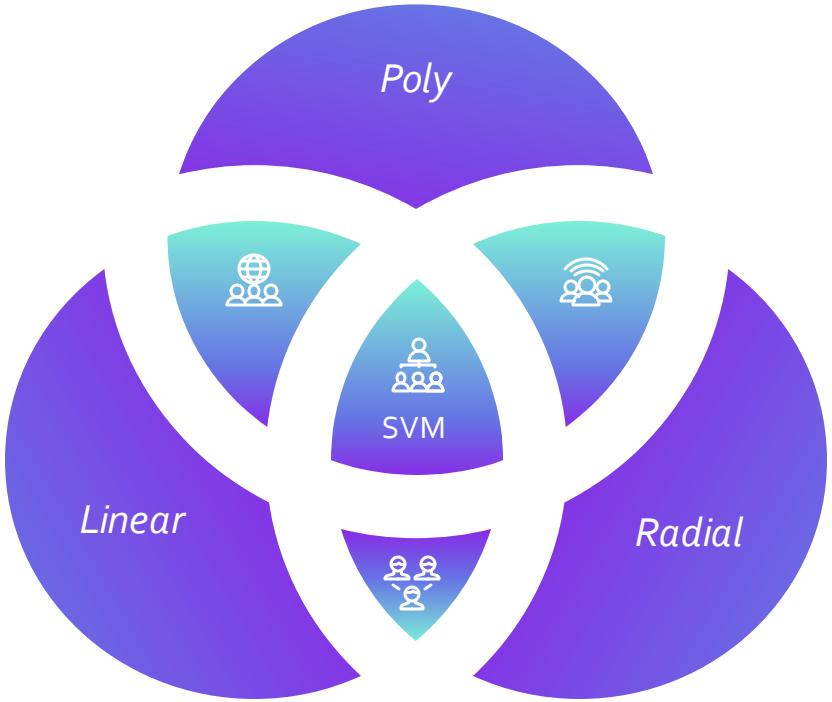




DECISION TREE ANALYSIS



SUPPORT VECTOR MACHINE




```
[1] "Kernel: linear Cost: 1"
          Actual
Predicted FALSE TRUE
    FALSE   745  665
    TRUE    670  915

Accuracy: 0.554
Precision: 0.577
Recall: 0.579

[1] "Kernel: linear Cost: 10"
          Actual
Predicted FALSE TRUE
    FALSE   745  666
    TRUE    670  914

Accuracy: 0.554
Precision: 0.577
Recall: 0.578

[1] "Kernel: linear Cost: 100"
          Actual
Predicted FALSE TRUE
    FALSE   747  666
    TRUE    668  914

Accuracy: 0.555
Precision: 0.578
Recall: 0.578
```

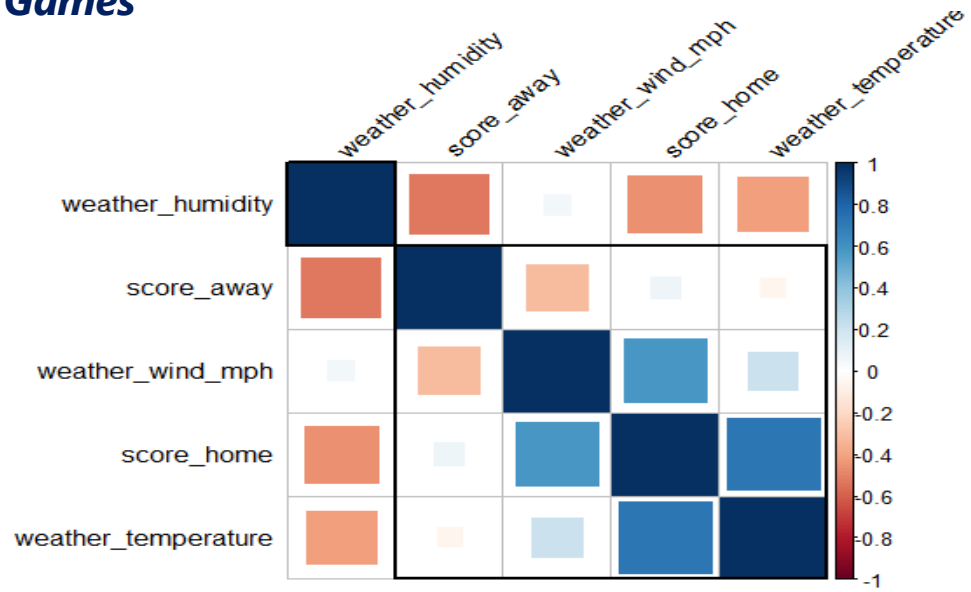
```
[1] "Kernel: polynomial Cost: 1"
          Actual
Predicted FALSE TRUE
    FALSE    14   17
    TRUE   1401 1563

Accuracy: 0.527
Precision: 0.527
Recall: 0.989

[1] "Kernel: polynomial Cost: 10"
          Actual
Predicted FALSE TRUE
    FALSE   299  238
    TRUE   1116 1342

Accuracy: 0.548
Precision: 0.546
Recall: 0.849

[1] "Kernel: polynomial Cost: 100"
          Actual
Predicted FALSE TRUE
    FALSE   562  483
    TRUE    853 1097

Accuracy: 0.554
Precision: 0.563
Recall: 0.694
```

```
[1] "Kernel: radial Cost: 1"
          Actual
Predicted FALSE TRUE
    FALSE   693  590
    TRUE    722  990

Accuracy: 0.562
Precision: 0.578
Recall: 0.627

[1] "Kernel: radial Cost: 10"
          Actual
Predicted FALSE TRUE
    FALSE   674  556
    TRUE    741 1024

Accuracy: 0.567
Precision: 0.580
Recall: 0.648

[1] "Kernel: radial Cost: 100"
          Actual
Predicted FALSE TRUE
    FALSE   705  608
    TRUE    710  972

Accuracy: 0.560
Precision: 0.578
Recall: 0.615
```

- **Radial Basis Function w/ Cost 10 has best Performance**
  - 56.7% accuracy, 58% precision, 64.8% recall
- **Very High Recall (98.9%) for Polynomial Functions w/ Cost 1**
  - Low accuracy and precision
- **Very little variance among linear models**
- **Average accuracy hovers around the 55% mark**

CORRELATION & PREDICTIONS

Kansas City Games



Positive Relationships:

- Temperature/ Home Score
- Wind/Home Score

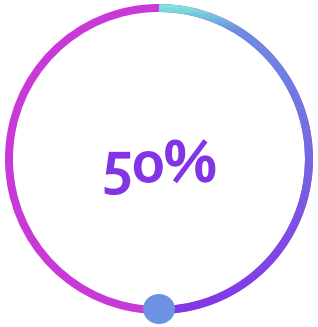
Negative Relationships:

- Humidity/Away Score
- Temperature/Wind

Little/No correlation:

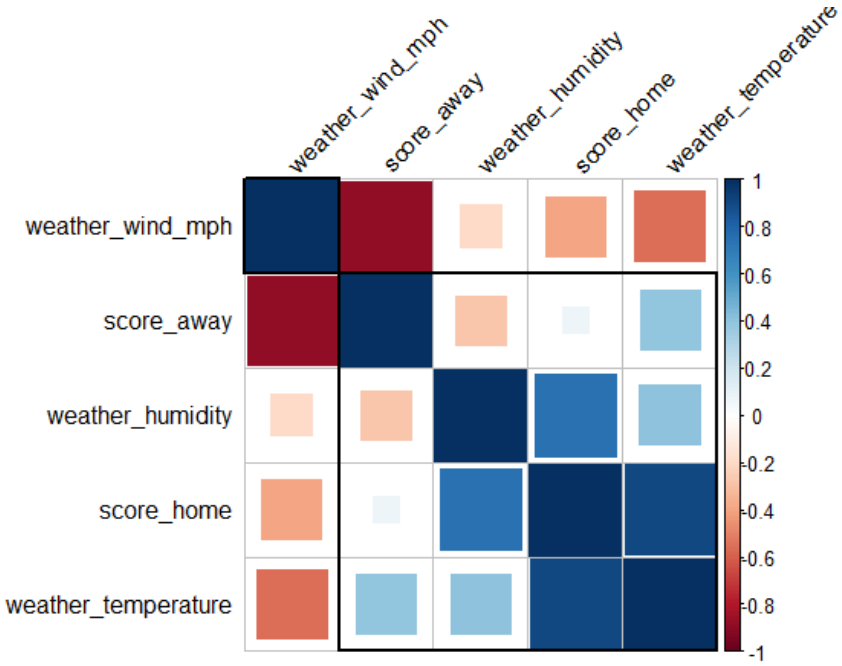
- Wind/Humidity
- Away/Home Score

Who wins the 2024 Superbowl?




```
                                1
"San Francisco 49ers"
```

Kansas City Games Won



Thank You

As data evolves, we harness its potential to revolutionize the way we analyze and interpret the game, pushing the boundaries of what is possible.

.

Swifty Analytics RMarkdown

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(readr)
library(ggplot2)
library(maps)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(knitr)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ lubridate 1.9.3     ✔ tibble    3.2.1
## ✔ purrr     1.0.2
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ magrittr::extract() masks tidyr::extract()
## ✖ dplyr::filter()     masks stats::filter()
## ✖ dplyr::lag()        masks stats::lag()
## ✖ purrr::map()        masks maps::map()
## ✖ purrr::set_names()  masks magrittr::set_names()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(tibble)
library(conflicted)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(RColorBrewer)
library(zoo)
library(rpart)
library(rpart.plot)
library(arules)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
library(arulesViz)
conflicts_prefer(dplyr::filter)
```

```
## [conflicted] Will prefer dplyr::filter over any other package.
```

```
stadiums_complete<-read.csv("~/Downloads/stadiums_final.csv")
spreadspoke_scores<-read.csv("~/Downloads/spreadspoke_scores.csv")
nfl_teams<-read.csv("~/Downloads/nfl_teams.csv")
cleaned_data <- stadiums_complete %>%
  select(stadium_name, stadium_open, stadium_close, stadium_type, stadium_weather_type,
stadium_capacity, stadium_surface, stadium_latitude, stadium_longitude, stadium_azimutha
ngle, stadium_elevation) %>%
  mutate(
    stadium_type = ifelse(stadium_type == "indoor", "indoor", "outdoor"),
    stadium_weather_type = case_when(
      grepl("cold", stadium_weather_type, ignore.case = TRUE) ~ "cold",
      grepl("moderate", stadium_weather_type, ignore.case = TRUE) ~ "moderate",
      grepl("warm", stadium_weather_type, ignore.case = TRUE) ~ "warm",
      grepl("indoor", stadium_weather_type, ignore.case = TRUE) ~ "indoor",
      TRUE ~ NA_character_
    ),
    stadium_surface = case_when(
      grepl("Grass", stadium_surface, ignore.case = TRUE) ~ "Grass",
      grepl("FieldTurf", stadium_surface, ignore.case = TRUE) ~ "FieldTurf",
      TRUE ~ NA_character_
    ),
    stadium_latitude = as.numeric(stadium_latitude),
    stadium_longitude = as.numeric(stadium_longitude),
    stadium_azimuthangle = as.numeric(stadium_azimuthangle),
    stadium_elevation = as.numeric(stadium_elevation)
  ) %>%
  mutate(across(where(is.character), ~na_if(.x, ""))) %>%
  mutate(across(where(is.numeric), ~na_if(.x, 0))) %>%
  filter(is.na(stadium_close) | stadium_close == "")
```

```
spreadspoke_scores_filtered <- spreadspoke_scores %>%
  filter(schedule_season >= 1979)

spreadspoke_scores_filtered <- spreadspoke_scores_filtered %>%
  mutate(winner = case_when(
    score_home > score_away ~ team_home,
    score_home < score_away ~ team_away,
    TRUE ~ NA_character_
  ))

team_id_to_name <- setNames(nfl_teams$team_name, nfl_teams$team_id)

spreadspoke_scores_filtered <- spreadspoke_scores_filtered %>%
  mutate(favorite_won = if_else(
    is.na(team_favorite_id) | is.na(winner),
    NA,
    if_else(winner == team_id_to_name[team_favorite_id], TRUE, FALSE)
  ))


head(spreadspoke_scores_filtered)
```

	schedule_date <chr>	schedule_season <int>	schedule_week <chr>	schedule_playoff <lgl>	team_home <chr>
1	9/1/1979	1979	1	FALSE	Tampa Bay Buccaneers
2	9/2/1979	1979	1	FALSE	Buffalo Bills
3	9/2/1979	1979	1	FALSE	Chicago Bears
4	9/2/1979	1979	1	FALSE	Denver Broncos
5	9/2/1979	1979	1	FALSE	Kansas City Chiefs
6	9/2/1979	1979	1	FALSE	Los Angeles Rams

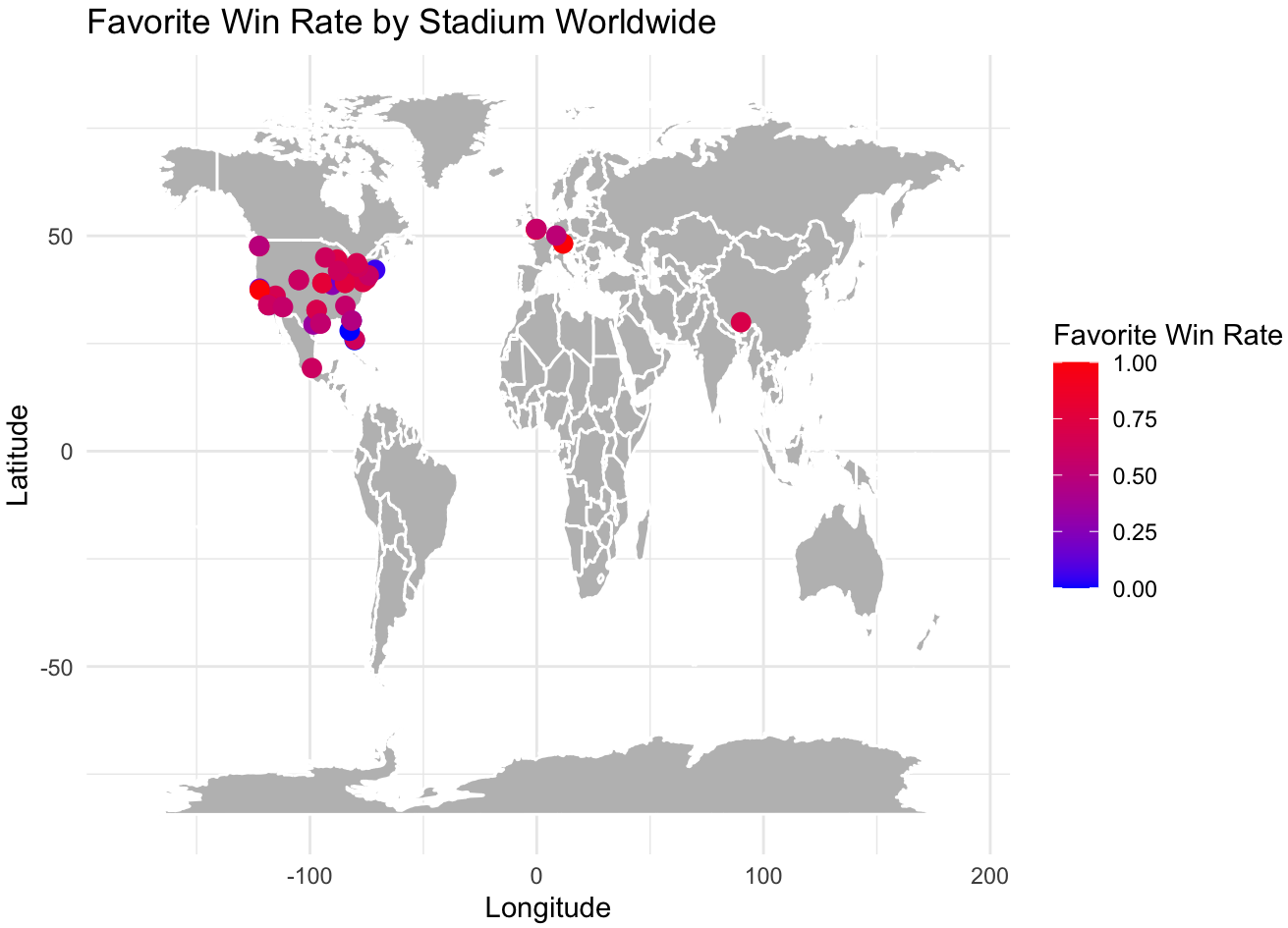
6 rows | 1-7 of 20 columns

```
spreadspoke_scores <- spreadspoke_scores %>%
  mutate(winner = case_when(
    score_home > score_away ~ team_home,
    score_home < score_away ~ team_away,
    TRUE ~ NA_character_
  ))

team_id_to_name <- setNames(nfl_teams$team_name, nfl_teams$team_id)

spreadspoke_scores <- spreadspoke_scores %>%
  mutate(favorite_won = if_else(
    is.na(team_favorite_id) | is.na(winner),
    NA,
    if_else(winner == team_id_to_name[team_favorite_id], TRUE, FALSE)
  ))


head(spreadspoke_scores)
```

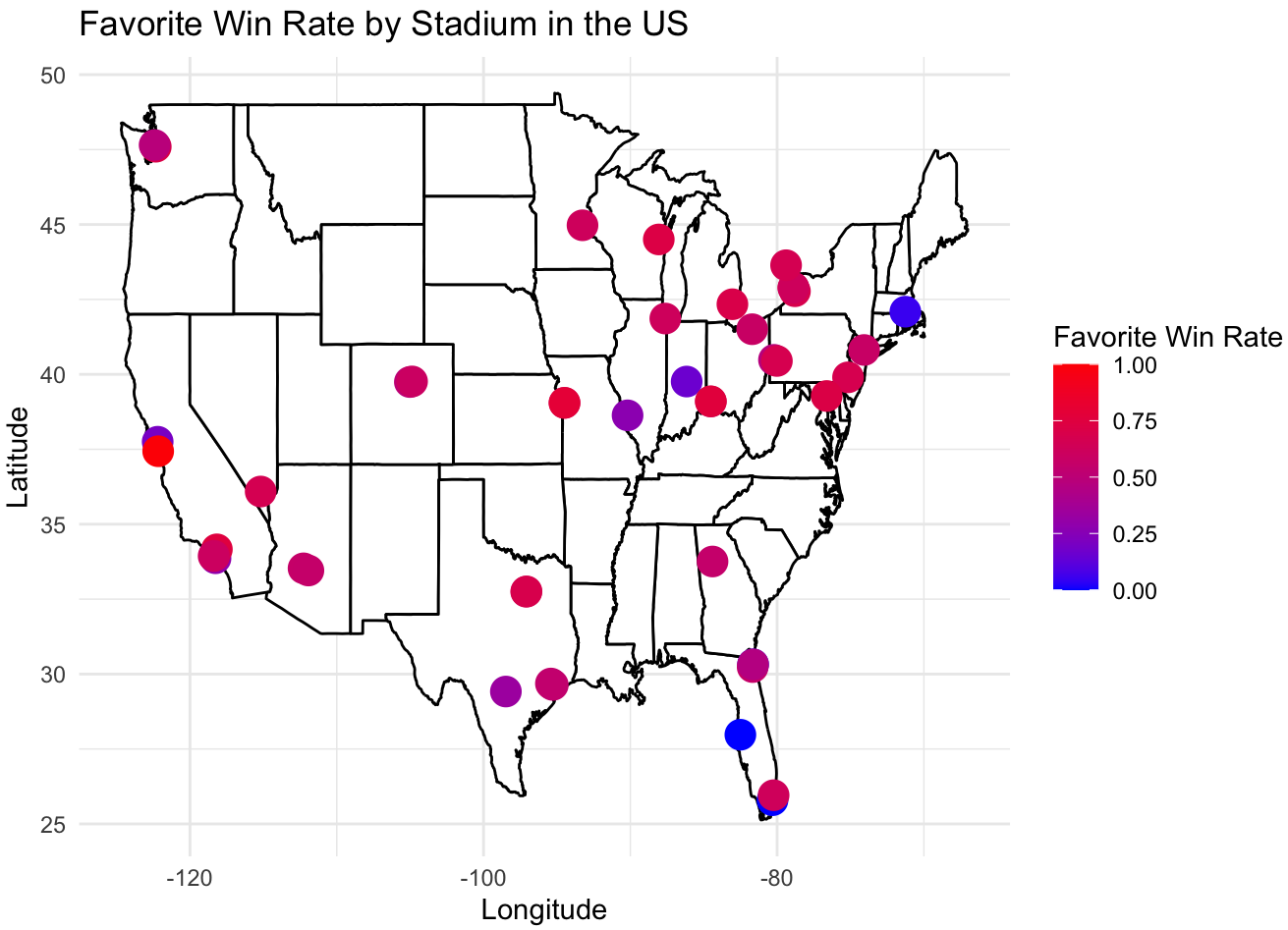
	schedule_date <chr>	schedule_season <int>	schedule_week <chr>	schedule_playoff <lgl>	team_home <chr>	sc
1	9/2/1966	1966	1	FALSE	Miami Dolphins	
2	9/3/1966	1966	1	FALSE	Houston Oilers	
3	9/4/1966	1966	1	FALSE	San Diego Chargers	
4	9/9/1966	1966	2	FALSE	Miami Dolphins	
5	9/10/1966	1966	1	FALSE	Green Bay Packers	
6	9/10/1966	1966	2	FALSE	Houston Oilers	

6 rows | 1-7 of 20 columns

```
favorite_win_rate <- spreadspoke_scores_filtered %>%
  group_by(stadium) %>%
  summarise(FavoriteWinRate = mean(favorite_won, na.rm = TRUE))

stadium_win_rate <- merge(cleaned_data, favorite_win_rate, by.x = "stadium_name", by.y =
"stadium")


# Base world map
world_map <- map_data("world")
ggplot() +
  geom_polygon(data = world_map, aes(x = long, y = lat, group = group), fill = "gray", c
olor = "white") +
  geom_point(data = stadium_win_rate, aes(x = stadium_longitude, y = stadium_latitude, c
olor = FavoriteWinRate), size = 3) +
  scale_color_gradient(low = "blue", high = "red", name = "Favorite Win Rate") +
  labs(title = "Favorite Win Rate by Stadium Worldwide", x = "Longitude", y = "Latitud
e") +
  theme_minimal() +
  theme(legend.position = "right")
```



```
us_latitude_min <- 24.396308
us_latitude_max <- 49.384358
us_longitude_min <- -125.001651
us_longitude_max <- -66.93457

stadium_win_rate_us <- stadium_win_rate %>%
  filter(stadium_latitude >= us_latitude_min & stadium_latitude <= us_latitude_max &
         stadium_longitude >= us_longitude_min & stadium_longitude <= us_longitude_max)


us_map <- map_data("state")
ggplot() +
  geom_polygon(data = us_map, aes(x = long, y = lat, group = group), fill = "white", col
or = "black") +
  geom_point(data = stadium_win_rate_us, aes(x = stadium_longitude, y = stadium_latitud
e, color = FavoriteWinRate), size = 5) +
  scale_color_gradient(low = "blue", high = "red", name = "Favorite Win Rate") +
  labs(title = "Favorite Win Rate by Stadium in the US", x = "Longitude", y = "Latitud
e") +
  theme_minimal()
```



```
team_to_region <- c(
  "buffalo bills" = "East", "miami dolphins" = "East", "new england patriots" = "East",
"new york jets" = "East",
  "dallas cowboys" = "East", "new york giants" = "East", "philadelphia eagles" = "East",
"washington commanders" = "East", "washington redskins" = "East", "washington football t
eam" = "East",
  "denver broncos" = "West", "kansas city chiefs" = "West", "las vegas raiders" = "Wes
t", "los angeles chargers" = "West", "san diego chargers" = "West", "oakland raiders" =
"West",
  "arizona cardinals" = "West", "los angeles rams" = "West", "san francisco 49ers" = "We
st", "seattle seahawks" = "West", "st. louis rams" = "West", "phoenix cardinals" = "Wes
t",
  "baltimore ravens" = "North", "cincinnati bengals" = "North", "cleveland browns" = "No
rth", "pittsburgh steelers" = "North",
  "chicago bears" = "North", "detroit lions" = "North", "green bay packers" = "North",
"minnesota vikings" = "North",
  "houston texans" = "South", "indianapolis colts" = "South", "jacksonville jaguars" =
"South", "tennessee titans" = "South",
  "atlanta falcons" = "South", "carolina panthers" = "South", "new orleans saints" = "So
uth", "tampa bay buccaneers" = "South"
)



team_to_region <- setNames(team_to_region, tolower(names(team_to_region)))

# Lowercase and trim whitespace for team names in the dataset
spreadspoke_scores_filtered$team_home <- tolower(trimws(spreadspoke_scores_filtered$team
_home))
spreadspoke_scores_filtered$team_away <- tolower(trimws(spreadspoke_scores_filtered$team
_away))
```

```
team_to_region_updated <- c(
  team_to_region,
  "st. louis cardinals" = "West",
  "baltimore colts" = "East",
  "houston oilers" = "South",
  "tennessee oilers" = "South",
  "los angeles rams" = "West",
  "san diego chargers" = "West",
  "oakland raiders" = "West",
  "los angeles raiders" = "West",
  "phoenix cardinals" = "West",
  "washington redskins" = "East",
  "tennessee titans" = "South"
)

spreadspoke_scores_filtered$team_home <- tolower(spreadspoke_scores_filtered$team_home)
spreadspoke_scores_filtered$team_away <- tolower(spreadspoke_scores_filtered$team_away)
```

```
spreadspoke_scores_filtered$region_home <- team_to_region[spreadspoke_scores_filtered$te
am_home]
spreadspoke_scores_filtered$region_away <- team_to_region[spreadspoke_scores_filtered$te
am_away]
# Reassign regions with the updated mapping
spreadspoke_scores_filtered$region_home <- team_to_region_updated[spreadspoke_scores_fil
tered$team_home]
spreadspoke_scores_filtered$region_away <- team_to_region_updated[spreadspoke_scores_fil
tered$team_away]
```

```
# Determine winners
spreadspoke_scores_filtered$winner <- ifelse(spreadspoke_scores_filtered$score_home > sp
readspoke_scores_filtered$score_away, spreadspoke_scores_filtered$team_home, spreadspoke
_scores_filtered$team_away)

# Determine the region of the winner and the loser
spreadspoke_scores_filtered$winner_region <- ifelse(spreadspoke_scores_filtered$score_ho
me > spreadspoke_scores_filtered$score_away, spreadspoke_scores_filtered$region_home, sp
readspoke_scores_filtered$region_away)
spreadspoke_scores_filtered$loser_region <- ifelse(spreadspoke_scores_filtered$score_hom
e > spreadspoke_scores_filtered$score_away, spreadspoke_scores_filtered$region_away, spr
eadspoke_scores_filtered$region_home)

library(dplyr)
library(tidyr) # For pivot_wider

results <- spreadspoke_scores_filtered %>%
  group_by(winner_region, loser_region) %>%
  summarise(wins = n(), .groups = "drop")

# Prepare a wider format for easier calculation of win percentages
wide_results <- pivot_wider(results, names_from = loser_region, values_from = wins, valu
es_fill = list(wins = 0))

# Calculate total games and win percentages for each region against others
wide_results <- wide_results %>%
  mutate(total_games = rowSums(select(., -winner_region)),
         East_percentage = East / total_games * 100,
         North_percentage = North / total_games * 100,
         South_percentage = South / total_games * 100,
         West_percentage = West / total_games * 100) %>%
  select(winner_region, total_games, ends_with("_percentage"))


# View the adjusted results
print(wide_results)
```

```
## # A tibble: 5 × 6
##   winner_region total_games East_percentage North_percentage South_percentage
##   <chr>               <dbl>           <dbl>            <dbl>            <dbl>
## 1 East                 3154            42.0             18.3             18.6
## 2 North                2797            17.4             40.4             23.2
## 3 South                2321            20.2             23.7             35.1
## 4 West                 3027            20.6             18.8             19.3
## 5 <NA>                    1             0                0                0
## # ℹ 1 more variable: West_percentage <dbl>
```

```
library(dplyr)

# Filter for games where winner_region or region_away is NA
na_region_games <- spreadspoke_scores_filtered %>%
  filter(is.na(region_home) | is.na(region_away))

# Get unique team names from these games
na_teams <- unique(c(na_region_games$team_home, na_region_games$team_away))

# Print the list of teams
print(na_teams)
```

```
## character(0)
```

```
traintest_dataset <- spreadspoke_scores_filtered %>%
  select(schedule_season, schedule_week, schedule_playoff, spread_favorite, over_under_l
ine, weather_temperature, weather_wind_mph, favorite_won)
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
set.seed(123)
splitIndex <- createDataPartition(traintest_dataset$favorite_won, p = .7, list = FALSE,
times = 1)
train_data <- traintest_dataset[splitIndex, ]
test_data <- traintest_dataset[-splitIndex, ]
```

```
library(e1071)

model_nb <- naiveBayes(favorite_won ~ ., data = train_data)

predictions_nb <- predict(model_nb, newdata = test_data)

conf_matrix_nb <- table(Predicted = predictions_nb, Actual = test_data$favorite_won)
print(conf_matrix_nb)
```

```
##          Actual
## Predicted FALSE TRUE
##     FALSE   901  828
##     TRUE    665  944
```

```
# Confusion matrix values
TN <- 901
FN <- 828
FP <- 665
TP <- 944

# Calculations
accuracy <- (TP + TN) / (TP + TN + FP + FN)
precision <- TP / (TP + FP)
recall <- TP / (TP + FN)

cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.5527262
```

```
cat("Precision:", precision, "\n")
```

```
## Precision: 0.5866998
```

```
cat("Recall:", recall, "\n")
```

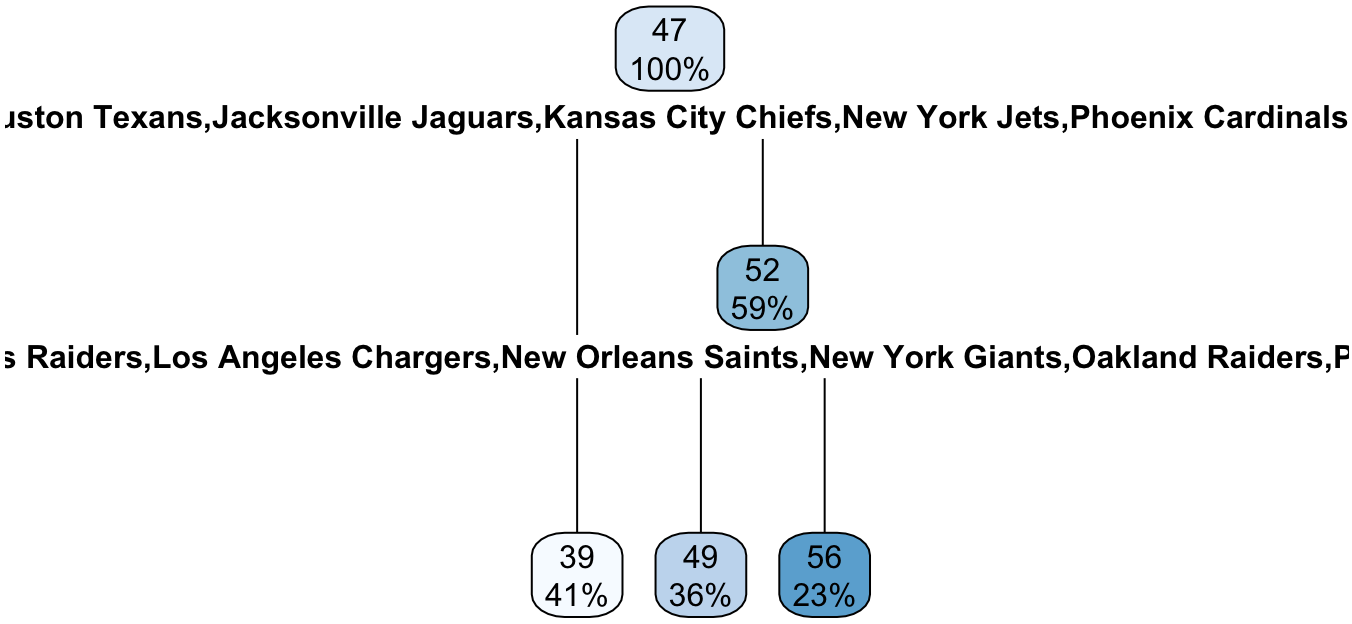
```
## Recall: 0.5327314
```

```
train_data$favorite_won <- as.factor(train_data$favorite_won)
test_data$favorite_won <- as.factor(test_data$favorite_won)

# Function to train SVM, make predictions, and evaluate model
train_eval_svm <- function(kernel_type, cost_value, train_data, test_data) {
  # Ensure no rows with NA values
  test_data_clean <- na.omit(test_data)
  train_data_clean <- na.omit(train_data)

  model_svm <- svm(favorite_won ~ ., data = train_data_clean, kernel = kernel_type, cost
= cost_value, na.action = na.omit)

  # Making predictions on the test set
  predictions_svm <- predict(model_svm, newdata = test_data_clean)

  conf_matrix_svm <- table(Predicted = predictions_svm, Actual = test_data_clean$favorit
e_won)
  print(paste("Kernel:", kernel_type, "Cost:", cost_value))
  print(conf_matrix_svm)

  # Calculate evaluation metrics
  accuracy <- sum(diag(conf_matrix_svm)) / sum(conf_matrix_svm)
  precision <- conf_matrix_svm[2, 2] / sum(conf_matrix_svm[2, ])
  recall <- conf_matrix_svm[2, 2] / sum(conf_matrix_svm[, 2])

  # Print evaluation metrics
  cat(sprintf("\nAccuracy: %.3f\n", accuracy))
  cat(sprintf("Precision: %.3f\n", precision))
  cat(sprintf("Recall: %.3f\n", recall))
}



kernels <- c("linear", "polynomial", "radial")
costs <- c(1, 10, 100)

for (kernel in kernels) {
  for (cost in costs) {
    train_eval_svm(kernel, cost, train_data, test_data)
    cat("\n")
  }
}
```

```
## [1] "Kernel: linear Cost: 1"
##          Actual
## Predicted FALSE TRUE
##     FALSE   745  665
##     TRUE    670  915
##
## Accuracy: 0.554
## Precision: 0.577
## Recall: 0.579
##
## [1] "Kernel: linear Cost: 10"
##          Actual
## Predicted FALSE TRUE
##     FALSE   745  665
##     TRUE    670  915
##
## Accuracy: 0.554
## Precision: 0.577
## Recall: 0.579
##
## [1] "Kernel: linear Cost: 100"
##          Actual
## Predicted FALSE TRUE
##     FALSE   745  665
##     TRUE    670  915
##
## Accuracy: 0.554
## Precision: 0.577
## Recall: 0.579
##
## [1] "Kernel: polynomial Cost: 1"
##          Actual
## Predicted FALSE TRUE
##     FALSE    14   17
##     TRUE   1401 1563
##
## Accuracy: 0.527
## Precision: 0.527
## Recall: 0.989
##
## [1] "Kernel: polynomial Cost: 10"
##          Actual
## Predicted FALSE TRUE
##     FALSE   299  238
##     TRUE   1116 1342
##
## Accuracy: 0.548
## Precision: 0.546
## Recall: 0.849
##
## [1] "Kernel: polynomial Cost: 100"
##          Actual
```

```
## Predicted FALSE TRUE
##     FALSE   562  483
##     TRUE    853 1097
##
## Accuracy: 0.554
## Precision: 0.563
## Recall: 0.694
##
## [1] "Kernel: radial Cost: 1"
##           Actual
## Predicted FALSE TRUE
##     FALSE   693  590
##     TRUE    722  990
##
## Accuracy: 0.562
## Precision: 0.578
## Recall: 0.627
##
## [1] "Kernel: radial Cost: 10"
##           Actual
## Predicted FALSE TRUE
##     FALSE   674  556
##     TRUE    741 1024
##
## Accuracy: 0.567
## Precision: 0.580
## Recall: 0.648
##
## [1] "Kernel: radial Cost: 100"
##           Actual
## Predicted FALSE TRUE
##     FALSE   705  608
##     TRUE    710  972
##
## Accuracy: 0.560
## Precision: 0.578
## Recall: 0.615
```

```
scores <- spreadspoke_scores

# Calculate win/loss for each game
scores <- scores %>%
  mutate(home_win = ifelse(score_home > score_away, 1, 0),
         away_win = ifelse(score_away > score_home, 1, 0))

home_wins <- scores %>%
  group_by(team_home) %>%
  summarise(HomeWins = sum(home_win))

away_wins <- scores %>%
  group_by(team_away) %>%
  summarise(AwayWins = sum(away_win))

home_wins <- home_wins %>% rename(team = team_home, HomeWins = HomeWins)
away_wins <- away_wins %>% rename(team = team_away, AwayWins = AwayWins)

win_percentages <- home_wins %>%
  full_join(away_wins, by = "team") %>%
  mutate(TotalWins = coalesce(HomeWins, 0) + coalesce(AwayWins, 0)) %>%
  left_join(scores %>% count(team_home) %>% rename(team = team_home, TotalGamesHome =
n), by = "team") %>%
  left_join(scores %>% count(team_away) %>% rename(team = team_away, TotalGamesAway =
n), by = "team") %>%
  mutate(TotalGames = coalesce(TotalGamesHome, 0) + coalesce(TotalGamesAway, 0),
         WinPercentage = ifelse(TotalGames > 0, (TotalWins / TotalGames) * 100, NA)) %>%
  select(team, TotalGames, TotalWins, WinPercentage)


# Merge nfl_teams dataset with win_percentages dataset
teams_with_win_pct <- merge(nfl_teams, win_percentages, by.x = "team_name", by.y = "tea
m", all.x = TRUE)

# Write the file to local
#write.csv(teams_with_win_pct, "C:/Users/santh/OneDrive/Documents/Syracuse/IST_707_Appli
ed Machine Learning/PROJECT/teams_with_win_pct.csv", row.names = FALSE)

# Anova
decision_tree_model <- rpart(WinPercentage ~ ., data = teams_with_win_pct, method = "ano
va")

#Visualize the decision tree
rpart.plot(decision_tree_model)
```




```
summary(decision_tree_model)
```

```
## Call:
## rpart(formula = WinPercentage ~ ., data = teams_with_win_pct,
##     method = "anova")
##   n= 44
##
##          CP nsplit rel error   xerror      xstd
## 1 0.6014842      0 1.0000000 1.045122 0.2504538
## 2 0.1189930      1 0.3985158 1.421805 0.3063639
## 3 0.0100000      2 0.2795227 1.192099 0.2954187
##
## Variable importance
##             team_name               team_id           team_id_pfr
##                    27                    18                    18
##       team_name_short            TotalWins team_division_pre2002
##                    18                    11                     4
##            TotalGames         team_division
##                     2                     2
##
## Node number 1: 44 observations,    complexity param=0.6014842
##   mean=46.84698, MSE=63.1807
##   left son=2 (18 obs) right son=3 (26 obs)
##   Primary splits:
##       team_name       splits as  LLRRLRRRLLRRLRLLRLLRRRRRRRRRLRRLRRLRLLLRRLLR, improv
e=0.6014842, (0 missing)
##       team_name_short splits as  LRLRRLLLRLRRRRLRLLLRRRLRRRRRRLRRL, improve=0.426203
2, (0 missing)
##       team_id         splits as  LLRRRRLLRRLRLRLLRRRRRLRRLRRRLLRL, improve=0.4262032,
(0 missing)
##       team_id_pfr     splits as  LLRRRLLRLRRLRLLLRRRLRLRRRRRRRLLL, improve=0.426203
2, (0 missing)
##       TotalWins       < 462   to the left,  improve=0.2632846, (0 missing)
##   Surrogate splits:
##       team_name_short       splits as  LRLRRLLLRLRRRRLRLLLRRRRRRRRRRLRRL, agree=0.90
9, adj=0.778, (0 split)
##       team_id               splits as  LLRRRRLLRRLRLRLLRRRRRRRRLRRRLLRL, agree=0.909,
adj=0.778, (0 split)
##       team_id_pfr           splits as  LLRRRLLRLRRLRLLLRRRRRLRRRRRRRRLLL, agree=0.90
9, adj=0.778, (0 split)
##       TotalWins             < 406   to the left,  agree=0.727, adj=0.333, (0 split)
##       team_division_pre2002 splits as  LRRRRRL, agree=0.659, adj=0.167, (0 split)
##
## Node number 2: 18 observations
##   mean=39.43807, MSE=37.51042
##
## Node number 3: 26 observations,    complexity param=0.118993
##   mean=51.97623, MSE=16.64103
##   left son=6 (16 obs) right son=7 (10 obs)
##   Primary splits:
##       team_name       splits as  --LR-LLL--RR-R--L--LLRRRRRLL-LL-RL-L---LL--L, improv
e=0.7645485, (0 missing)
##       team_name_short splits as  -L-LR---L-LRRL-L---LRLRLRRLLR-LRL, improve=0.574419
0, (0 missing)
```

```
##       team_id         splits as  --RLLL--RR-R-L--LRLRRRLL-LRL--LL, improve=0.5744190,
(0 missing)
##       team_id_pfr     splits as  --LLL--L-RR-R---RRLRL-LLRLRRLL--L, improve=0.574419
0, (0 missing)
##       TotalWins       < 494.5 to the left,  improve=0.3797118, (0 missing)
##   Surrogate splits:
##       TotalWins             < 494.5 to the left,  agree=0.885, adj=0.7, (0 split)
##       TotalGames            < 935   to the left,  agree=0.808, adj=0.5, (0 split)
##       team_division         splits as  LRRLLLRLL, agree=0.769, adj=0.4, (0 split)
##       team_division_pre2002 splits as  -LLLRLL,   agree=0.654, adj=0.1, (0 split)
##
## Node number 6: 16 observations
##   mean=49.15634, MSE=4.327202
##
## Node number 7: 10 observations
##   mean=56.48806, MSE=3.263681
```

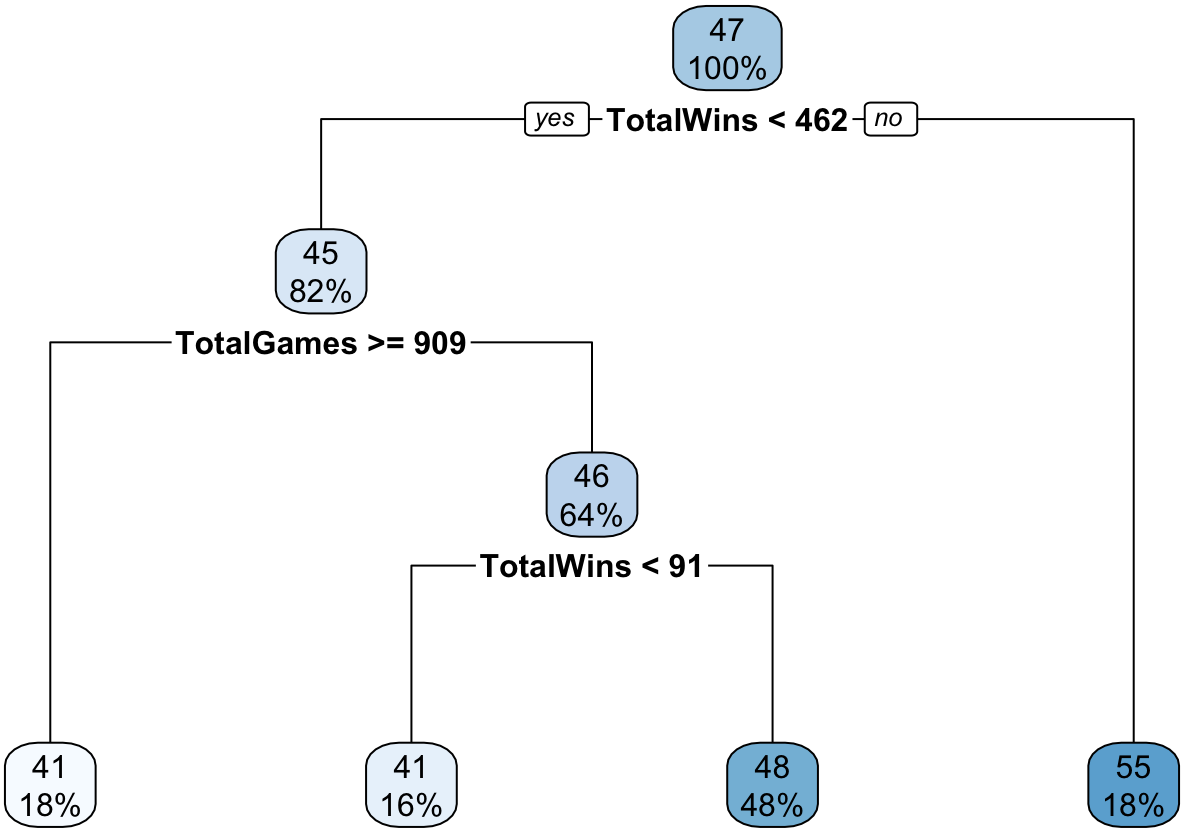
```
### More Tuning of the Model ###

library(caret)
predictors <- c("team_name", "TotalGames", "TotalWins")
target <- "WinPercentage"

dataset <- teams_with_win_pct %>%
  select(all_of(c(predictors, target))) %>%
  na.omit() # Removing rows with NA values

# Cross-validation setup
set.seed(123) # For reproducibility
trainControl <- trainControl(method = "cv", number = 10) # 10-fold cross-validation

# Training the model with cross-validation
model <- train(reformulate(predictors, target), data = dataset,
               method = "rpart",
               trControl = trainControl,
               tuneGrid = expand.grid(cp = seq(0.01, 0.1, by = 0.01)),
               control = rpart.control(minsplit = 20, maxdepth = 5))
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
print(model)
```

```
## CART
##
## 44 samples
##  3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 40, 38, 40, 41, 39, 40, ...
## Resampling results across tuning parameters:
##
##   cp    RMSE      Rsquared   MAE
##   0.01  7.256512  0.3181207  5.853583
##   0.02  7.256512  0.3181207  5.853583
##   0.03  7.256512  0.3181207  5.853583
##   0.04  7.256512  0.3181207  5.853583
##   0.05  7.375322  0.3155753  5.954950
##   0.06  7.445071  0.3226240  5.954950
##   0.07  7.532017  0.2842381  5.941486
##   0.08  7.515034  0.2080995  5.859904
##   0.09  7.340053  0.2421398  5.729794
##   0.10  7.340053  0.2421398  5.729794
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.04.
```

```
rpart.plot(model$finalModel)
```




```
summary(model$finalModel)
```

```
## Call:
## (function (formula, data, weights, subset, na.action = na.rpart,
##     method, model = FALSE, x = FALSE, y = TRUE, parms, control,
##     cost, ...)
## {
##     Call <- match.call()
##     if (is.data.frame(model)) {
##         m <- model
##         model <- FALSE
##     }
##     else {
##         indx <- match(c("formula", "data", "weights", "subset"),
##             names(Call), nomatch = 0)
##         if (indx[1] == 0)
##             stop("a 'formula' argument is required")
##         temp <- Call[c(1, indx)]
##         temp$na.action <- na.action
##         temp[[1]] <- quote(stats::model.frame)
##         m <- eval.parent(temp)
##     }
##     Terms <- attr(m, "terms")
##     if (any(attr(Terms, "order") > 1))
##         stop("Trees cannot handle interaction terms")
##     Y <- model.response(m)
##     wt <- model.weights(m)
##     if (any(wt < 0))
##         stop("negative weights not allowed")
##     if (!length(wt))
##         wt <- rep(1, nrow(m))
##     offset <- model.offset(m)
##     X <- rpart.matrix(m)
##     nobs <- nrow(X)
##     nvar <- ncol(X)
##     if (missing(method)) {
##         method <- if (is.factor(Y) || is.character(Y))
##             "class"
##         else if (inherits(Y, "Surv"))
##             "exp"
##         else if (is.matrix(Y))
##             "poisson"
##         else "anova"
##     }
##     if (is.list(method)) {
##         mlist <- method
##         method <- "user"
##         init <- if (missing(parms))
##             mlist$init(Y, offset, wt = wt)
##         else mlist$init(Y, offset, parms, wt)
##         keep <- rpartcallback(mlist, nobs, init)
##         method.int <- 4
##         parms <- init$parms
##     }
```

```
##     else {
##         method.int <- pmatch(method, c("anova", "poisson", "class",
##             "exp"))
##         if (is.na(method.int))
##             stop("Invalid method")
##         method <- c("anova", "poisson", "class", "exp")[method.int]
##         if (method.int == 4)
##             method.int <- 2
##         init <- if (missing(parms))
##             get(paste("rpart", method, sep = "."), envir = environment())(Y,
##                 offset, , wt)
##         else get(paste("rpart", method, sep = "."), envir = environment())(Y,
##             offset, parms, wt)
##         ns <- asNamespace("rpart")
##         if (!is.null(init$print))
##             environment(init$print) <- ns
##         if (!is.null(init$summary))
##             environment(init$summary) <- ns
##         if (!is.null(init$text))
##             environment(init$text) <- ns
##     }
##     Y <- init$y
##     xlevels <- .getXlevels(Terms, m)
##     cats <- rep(0, ncol(X))
##     if (!is.null(xlevels)) {
##         indx <- match(names(xlevels), colnames(X), nomatch = 0)
##         cats[indx] <- (unlist(lapply(xlevels, length)))[indx >
##             0]
##     }
##     extraArgs <- list(...)
##     if (length(extraArgs)) {
##         controlargs <- names(formals(rpart.control))
##         indx <- match(names(extraArgs), controlargs, nomatch = 0)
##         if (any(indx == 0))
##             stop(gettextf("Argument %s not matched", names(extraArgs)[indx ==
##                 0]), domain = NA)
##     }
##     controls <- rpart.control(...)
##     if (!missing(control))
##         controls[names(control)] <- control
##     xval <- controls$xval
##     if (is.null(xval) || (length(xval) == 1 && xval == 0) ||
##         method == "user") {
##         xgroups <- 0
##         xval <- 0
##     }
##     else if (length(xval) == 1) {
##         xgroups <- sample(rep(1:xval, length.out = nobs), nobs,
##             replace = FALSE)
##     }
##     else if (length(xval) == nobs) {
##         xgroups <- xval
```

```
##         xval <- length(unique(xgroups))
##     }
##     else {
##         if (!is.null(attr(m, "na.action"))) {
##             temp <- as.integer(attr(m, "na.action"))
##             xval <- xval[-temp]
##             if (length(xval) == nobs) {
##                 xgroups <- xval
##                 xval <- length(unique(xgroups))
##             }
##             else stop("Wrong length for 'xval'")
##         }
##         else stop("Wrong length for 'xval'")
##     }
##     if (missing(cost))
##         cost <- rep(1, nvar)
##     else {
##         if (length(cost) != nvar)
##             stop("Cost vector is the wrong length")
##         if (any(cost <= 0))
##             stop("Cost vector must be positive")
##     }
##     tfun <- function(x) if (is.matrix(x))
##         rep(is.ordered(x), ncol(x))
##     else is.ordered(x)
##     labs <- sub("^`(.*)`$", "\\1", attr(Terms, "term.labels"))
##     isord <- unlist(lapply(m[labs], tfun))
##     storage.mode(X) <- "double"
##     storage.mode(wt) <- "double"
##     temp <- as.double(unlist(init$parms))
##     if (!length(temp))
##         temp <- 0
##     rpfit <- .Call(C_rpart, ncat = as.integer(cats * !isord),
##         method = as.integer(method.int), as.double(unlist(controls)),
##         temp, as.integer(xval), as.integer(xgroups), as.double(t(init$y)),
##         X, wt, as.integer(init$numy), as.double(cost))
##     nsplit <- nrow(rpfit$isplit)
##     ncat <- if (!is.null(rpfit$csplit))
##         nrow(rpfit$csplit)
##     else 0
##     if (nsplit == 0)
##         xval <- 0
##     numcp <- ncol(rpfit$cptable)
##     temp <- if (nrow(rpfit$cptable) == 3)
##         c("CP", "nsplit", "rel error")
##     else c("CP", "nsplit", "rel error", "xerror", "xstd")
##     dimnames(rpfit$cptable) <- list(temp, 1:numcp)
##     tname <- c("<leaf>", colnames(X))
##     splits <- matrix(c(rpfit$isplit[, 2:3], rpfit$dsplit), ncol = 5,
##         dimnames = list(tname[rpfit$isplit[, 1] + 1], c("count",
##             "ncat", "improve", "index", "adj")))
##     index <- rpfit$inode[, 2]
```

```
##     nadd <- sum(isord[rpfit$isplit[, 1]])
##     if (nadd > 0) {
##         newc <- matrix(0, nadd, max(cats))
##         cvar <- rpfit$isplit[, 1]
##         indx <- isord[cvar]
##         cdir <- splits[indx, 2]
##         ccut <- floor(splits[indx, 4])
##         splits[indx, 2] <- cats[cvar[indx]]
##         splits[indx, 4] <- ncat + 1:nadd
##         for (i in 1:nadd) {
##             newc[i, 1:(cats[(cvar[indx])[i]])] <- -as.integer(cdir[i])
##             newc[i, 1:ccut[i]] <- as.integer(cdir[i])
##         }
##         catmat <- if (ncat == 0)
##             newc
##         else {
##             cs <- rpfit$csplit
##             ncs <- ncol(cs)
##             ncc <- ncol(newc)
##             if (ncs < ncc)
##                 cs <- cbind(cs, matrix(0, nrow(cs), ncc - ncs))
##             rbind(cs, newc)
##         }
##         ncat <- ncat + nadd
##     }
##     else catmat <- rpfit$csplit
##     if (nsplit == 0) {
##         frame <- data.frame(row.names = 1, var = "<leaf>", n = rpfit$inode[,
##             5], wt = rpfit$dnode[, 3], dev = rpfit$dnode[, 1],
##             yval = rpfit$dnode[, 4], complexity = rpfit$dnode[,
##                 2], ncompete = 0, nsurrogate = 0)
##     }
##     else {
##         temp <- ifelse(index == 0, 1, index)
##         svar <- ifelse(index == 0, 0, rpfit$isplit[temp, 1])
##         frame <- data.frame(row.names = rpfit$inode[, 1], var = tname[svar +
##             1], n = rpfit$inode[, 5], wt = rpfit$dnode[, 3],
##             dev = rpfit$dnode[, 1], yval = rpfit$dnode[, 4],
##             complexity = rpfit$dnode[, 2], ncompete = pmax(0,
##                 rpfit$inode[, 3] - 1), nsurrogate = rpfit$inode[,
##                 4])
##     }
##     if (method.int == 3) {
##         numclass <- init$numresp - 2
##         nodeprob <- rpfit$dnode[, numclass + 5]/sum(wt)
##         temp <- pmax(1, init$counts)
##         temp <- rpfit$dnode[, 4 + (1:numclass)] %*% diag(init$parms$prior/temp)
##         yprob <- temp/rowSums(temp)
##         yval2 <- matrix(rpfit$dnode[, 4 + (0:numclass)], ncol = numclass +
##             1)
##         frame$yval2 <- cbind(yval2, yprob, nodeprob)
##     }
```

```
##     else if (init$numresp > 1)
##         frame$yval2 <- rpfit$dnode[, -(1:3), drop = FALSE]
##     if (is.null(init$summary))
##         stop("Initialization routine is missing the 'summary' function")
##     functions <- if (is.null(init$print))
##         list(summary = init$summary)
##     else list(summary = init$summary, print = init$print)
##     if (!is.null(init$text))
##         functions <- c(functions, list(text = init$text))
##     if (method == "user")
##         functions <- c(functions, mlist)
##     where <- rpfit$which
##     names(where) <- row.names(m)
##     ans <- list(frame = frame, where = where, call = Call, terms = Terms,
##         cptable = t(rpfit$cptable), method = method, parms = init$parms,
##         control = controls, functions = functions, numresp = init$numresp)
##     if (nsplit)
##         ans$splits = splits
##     if (ncat > 0)
##         ans$csplit <- catmat + 2
##     if (nsplit)
##         ans$variable.importance <- importance(ans)
##     if (model) {
##         ans$model <- m
##         if (missing(y))
##             y <- FALSE
##     }
##     if (y)
##         ans$y <- Y
##     if (x) {
##         ans$x <- X
##         ans$wt <- wt
##     }
##     ans$ordered <- isord
##     if (!is.null(attr(m, "na.action")))
##         ans$na.action <- attr(m, "na.action")
##     if (!is.null(xlevels))
##         attr(ans, "xlevels") <- xlevels
##     if (method == "class")
##         attr(ans, "ylevels") <- init$ylevels
##     class(ans) <- "rpart"
##     ans
## })(formula = .outcome ~ ., data = list(c(0, 1, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 1, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0,
## 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0), c(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0), c(0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0,
## 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0,
## 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
## 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 1, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0), c(0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
## 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
## 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 1, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 1, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0), c(0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1),
## 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 1, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 1, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0), c(0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1), c(495, 920,
## 268, 481, 41, 932, 484, 921, 893, 871, 963, 938, 912, 949, 480,
## 367, 672, 483, 940, 68, 118, 209, 597, 939, 948, 912, 905, 930,
## 921, 664, 942, 96, 959, 804, 957, 792, 320, 346, 780, 32, 421,
## 34, 34, 864), c(208, 399, 136, 273, 12, 442, 223, 434, 403, 375,
## 571, 507, 373, 518, 211, 157, 347, 206, 219, 32, 58, 123, 337,
## 515, 519, 519, 420, 439, 405, 342, 482, 32, 560, 379, 295, 407,
## 145, 148, 318, 16, 217, 12, 14, 440), c(42.020202020202, 43.3695652173913,
## 50.7462686567164, 56.7567567567568, 29.2682926829268, 47.4248927038627,
## 46.0743801652893, 47.1226927252986, 45.1287793952968, 43.0539609644087,
## 59.2938733125649, 54.0511727078891, 40.8991228070175, 54.5837723919916,
## 43.9583333333333, 42.7792915531335, 51.6369047619048, 42.6501035196687,
## 23.2978723404255, 47.0588235294118, 49.1525423728814, 58.8516746411483,
## 56.4489112227806, 54.8455804046858, 54.746835443038, 56.9078947368421,
## 46.4088397790055, 47.2043010752688, 43.9739413680782, 51.5060240963855,
## 51.1677282377919, 33.3333333333333, 58.3941605839416, 47.1393034825871,
## 30.8254963427377, 51.3888888888889, 45.3125, 42.7745664739884,
## 40.7692307692308, 50, 51.5439429928741, 35.2941176470588, 41.1764705882353,
## 50.9259259259259)), control = list(20, 7, 0, 4, 5, 2, 0, 5, 0))
##   n= 44
##
##           CP nsplit rel error
```

```
## 1 0.26328455      0 1.0000000
## 2 0.08545769      1 0.7367154
## 3 0.04000000      3 0.5658001
##
## Variable importance
##  TotalWins TotalGames
##         53         47
##
## Node number 1: 44 observations,    complexity param=0.2632846
##   mean=46.84698, MSE=63.1807
##   left son=2 (36 obs) right son=3 (8 obs)
##   Primary splits:
##       TotalWins  < 462   to the left,  improve=0.2632846, (0 missing)
##       TotalGames < 163.5 to the left,  improve=0.1111369, (0 missing)
##   Surrogate splits:
##       TotalGames < 935   to the left,  agree=0.932, adj=0.625, (0 split)
##
## Node number 2: 36 observations,    complexity param=0.08545769
##   mean=44.92434, MSE=55.57162
##   left son=4 (8 obs) right son=5 (28 obs)
##   Primary splits:
##       TotalGames < 908.5 to the right, improve=0.09997196, (0 missing)
##       TotalWins  < 327.5 to the left,  improve=0.09034017, (0 missing)
##   Surrogate splits:
##       TotalWins < 427   to the right, agree=0.833, adj=0.25, (0 split)
##
## Node number 3: 8 observations
##   mean=55.49888, MSE=5.931809
##
## Node number 4: 8 observations
##   mean=40.51474, MSE=68.34141
##
## Node number 5: 28 observations,    complexity param=0.08545769
##   mean=46.18423, MSE=44.78019
##   left son=10 (7 obs) right son=11 (21 obs)
##   Primary splits:
##       TotalWins  < 90.5  to the left,  improve=0.2194327, (0 missing)
##       TotalGames < 163.5 to the left,  improve=0.2194327, (0 missing)
##   Surrogate splits:
##       TotalGames < 163.5 to the left,  agree=1, adj=1, (0 split)
##
## Node number 10: 7 observations
##   mean=40.7548, MSE=58.96448
##
## Node number 11: 21 observations
##   mean=47.99404, MSE=26.95044
```

```
importance <- varImp(model, scale = FALSE)
print(importance)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 45)
##
##                                Overall
## TotalWins                       0.5731
## TotalGames                      0.4305
## `team_namePhoenix Cardinals`    0.0000
## `team_nameDallas Cowboys`       0.0000
## `team_nameDenver Broncos`       0.0000
## `team_nameBoston Patriots`      0.0000
## `team_nameTampa Bay Buccaneers` 0.0000
## `team_nameBuffalo Bills`        0.0000
## `team_nameTennessee Titans`     0.0000
## `team_nameWashington Commanders` 0.0000
## `team_nameIndianapolis Colts`   0.0000
## `team_nameNew York Giants`      0.0000
## `team_namePhiladelphia Eagles`  0.0000
## `team_nameBaltimore Colts`      0.0000
## `team_nameHouston Texans`       0.0000
## `team_nameAtlanta Falcons`      0.0000
## `team_nameCincinnati Bengals`   0.0000
## `team_nameWashington Redskins`  0.0000
## `team_nameSeattle Seahawks`     0.0000
## `team_nameMiami Dolphins`       0.0000
```

```
spreadspoke_scores_filtered <- spreadspoke_scores_filtered %>%
  mutate_if(is.character, as.factor) %>%
  mutate_if(is.numeric, discretize)

spreadspoke_scores_filtered %>% select(schedule_playoff, team_home, score_home, score_aw
ay, team_away, over_under_line, stadium, stadium_neutral, weather_temperature, weather_w
ind_mph, weather_humidity, winner, favorite_won)
```

schedule_playoff <lgl>	team_home <fct>	score_ho… <fct>	score_away <fct>	team_away <fct>	ov <f
FALSE	tampa bay buccaneers	[27,70]	[16,24)	detroit lions	[2
FALSE	buffalo bills	[0,17)	[0,16)	miami dolphins	[2
FALSE	chicago bears	[0,17)	[0,16)	green bay packers	[2
FALSE	denver broncos	[0,17)	[0,16)	cincinnati bengals	[2
FALSE	kansas city chiefs	[0,17)	[0,16)	baltimore colts	[2
FALSE	los angeles rams	[17,27)	[24,59]	oakland raiders	[2
FALSE	minnesota vikings	[27,70]	[16,24)	san francisco 49ers	[2
FALSE	new orleans saints	[27,70]	[24,59]	atlanta falcons	[2

schedule_playoff	team_home	score_ho…	score_away	team_away	ov
<lgl>	<fct>	<fct>	<fct>	<fct>	<f
FALSE	new york jets	[17,27)	[24,59]	cleveland browns	[4
FALSE	philadelphia eagles	[17,27)	[16,24)	new york giants	[2

1-10 of 10,000 rows | 1-6 of 13 columns Previous **1** 2 3 4 5 6 … 1000 Next

```
spreadspoke_scores_filtered <- spreadspoke_scores_filtered %>%
  select(-weather_detail, -team_favorite_id)

spreadspoke_scores_filtered$favorite_won <- factor(spreadspoke_scores_filtered$favorite_
won, levels = c(TRUE, FALSE))
```

```
appearance <- list(rhs="favorite_won=TRUE", default="lhs")

rules_favorite_won <- apriori(
  data = spreadspoke_scores_filtered,
  parameter = list(supp = 0.015, conf = 0.88),
  appearance = appearance,
  control = list(verbose=FALSE)
)

arules::inspect(rules_favorite_won)
```

```
##      lhs                                        rhs                    support confide
nce   coverage     lift count
## [1]  {stadium=Candlestick Park,
##       winner=san francisco 49ers}          => {favorite_won=TRUE} 0.01557522  0.8844
221 0.01761062 1.691886   176
## [2]  {spread_favorite=[-26.5,-6.5),
##       winner=san francisco 49ers}          => {favorite_won=TRUE} 0.01592920  0.9278
351 0.01716814 1.774934   180
## [3]  {team_home=san francisco 49ers,
##       stadium=Candlestick Park,
##       winner=san francisco 49ers}          => {favorite_won=TRUE} 0.01557522  0.8844
221 0.01761062 1.691886   176
## [4]  {stadium=Candlestick Park,
##       winner=san francisco 49ers,
##       region_home=West}                    => {favorite_won=TRUE} 0.01557522  0.8844
221 0.01761062 1.691886   176
## [5]  {stadium=Candlestick Park,
##       winner=san francisco 49ers,
##       winner_region=West}                  => {favorite_won=TRUE} 0.01557522  0.8844
221 0.01761062 1.691886   176
## [6]  {spread_favorite=[-26.5,-6.5),
##       winner=san francisco 49ers,
##       winner_region=West}                  => {favorite_won=TRUE} 0.01592920  0.9278
351 0.01716814 1.774934   180
## [7]  {score_away=[0,16),
##       spread_favorite=[-26.5,-6.5),
##       winner_region=North}                 => {favorite_won=TRUE} 0.02769912  0.9152
047 0.03026549 1.750772   313
## [8]  {score_home=[27,70],
##       spread_favorite=[-26.5,-6.5),
##       winner_region=North}                 => {favorite_won=TRUE} 0.02778761  0.8870
056 0.03132743 1.696828   314
## [9]  {score_away=[0,16),
##       spread_favorite=[-26.5,-6.5),
##       region_home=North}                   => {favorite_won=TRUE} 0.02743363  0.8959
538 0.03061947 1.713946   310
## [10] {team_home=san francisco 49ers,
##       stadium=Candlestick Park,
##       winner=san francisco 49ers,
##       region_home=West}                    => {favorite_won=TRUE} 0.01557522  0.8844
221 0.01761062 1.691886   176
## [11] {team_home=san francisco 49ers,
##       stadium=Candlestick Park,
##       winner=san francisco 49ers,
##       winner_region=West}                  => {favorite_won=TRUE} 0.01557522  0.8844
221 0.01761062 1.691886   176
## [12] {stadium=Candlestick Park,
##       winner=san francisco 49ers,
##       region_home=West,
##       winner_region=West}                  => {favorite_won=TRUE} 0.01557522  0.8844
221 0.01761062 1.691886   176
## [13] {schedule_season=[2.01e+03,2.02e+03],
```

```
##       spread_favorite=[-26.5,-6.5),
##       region_home=North,
##       winner_region=North}                 => {favorite_won=TRUE} 0.01858407  0.8936
170 0.02079646 1.709476   210
## [14] {score_away=[0,16),
##       spread_favorite=[-26.5,-6.5),
##       region_home=North,
##       winner_region=North}                 => {favorite_won=TRUE} 0.02725664  0.9194
030 0.02964602 1.758804   308
## [15] {score_home=[27,70],
##       spread_favorite=[-26.5,-6.5),
##       region_home=North,
##       winner_region=North}                 => {favorite_won=TRUE} 0.02716814  0.9109
792 0.02982301 1.742689   307
## [16] {team_home=san francisco 49ers,
##       stadium=Candlestick Park,
##       winner=san francisco 49ers,
##       region_home=West,
##       winner_region=West}                  => {favorite_won=TRUE} 0.01557522  0.8844
221 0.01761062 1.691886   176
```

```
appearance <- list(rhs="favorite_won=FALSE", default="lhs")

rules_favorite_not_won <- apriori(
  data = spreadspoke_scores_filtered,
  parameter = list(supp = 0.015, conf = 0.99),
  appearance = appearance,
  control = list(verbose=FALSE)
)

arules::inspect(rules_favorite_not_won)
```

```
##      lhs                                        rhs                     support confid
ence   coverage     lift count
## [1]  {winner=tennessee titans}               => {favorite_won=FALSE} 0.01911504  0.995
3917 0.01920354 2.153950   216
## [2]  {winner=indianapolis colts}             => {favorite_won=FALSE} 0.03053097  0.991
3793 0.03079646 2.145267   345
## [3]  {winner=tennessee titans,
##       winner_region=South}                   => {favorite_won=FALSE} 0.01911504  0.995
3917 0.01920354 2.153950   216
## [4]  {team_home=indianapolis colts,
##       winner=indianapolis colts}             => {favorite_won=FALSE} 0.01699115  0.994
8187 0.01707965 2.152710   192
## [5]  {winner=indianapolis colts,
##       winner_region=South}                   => {favorite_won=FALSE} 0.03053097  0.991
3793 0.03079646 2.145267   345
## [6]  {winner=indianapolis colts,
##       region_home=South}                     => {favorite_won=FALSE} 0.02150442  0.991
8367 0.02168142 2.146257   243
## [7]  {winner=indianapolis colts,
##       region_away=South}                     => {favorite_won=FALSE} 0.01911504  0.990
8257 0.01929204 2.144069   216
## [8]  {weather_wind_mph=[0,4),
##       winner=indianapolis colts}             => {favorite_won=FALSE} 0.02000000  0.991
2281 0.02017699 2.144940   226
## [9]  {weather_temperature=[72,97],
##       winner=indianapolis colts}             => {favorite_won=FALSE} 0.02159292  0.991
8699 0.02176991 2.146329   244
## [10] {over_under_line=[44,63.5],
##       winner=indianapolis colts}             => {favorite_won=FALSE} 0.01707965  0.994
8454 0.01716814 2.152768   193
## [11] {team_away=new england patriots,
##       winner=new england patriots}           => {favorite_won=FALSE} 0.01725664  0.994
8980 0.01734513 2.152881   195
## [12] {spread_favorite=[-26.5,-6.5),
##       winner=new england patriots}           => {favorite_won=FALSE} 0.01716814  1.000
0000 0.01716814 2.163922   194
## [13] {schedule_season=[2.01e+03,2.02e+03],
##       winner=new england patriots}           => {favorite_won=FALSE} 0.01504425  1.000
0000 0.01504425 2.163922   170
## [14] {score_away=[24,59],
##       winner=new england patriots}           => {favorite_won=FALSE} 0.01522124  0.994
2197 0.01530973 2.151414   172
## [15] {over_under_line=[44,63.5],
##       winner=new england patriots}           => {favorite_won=FALSE} 0.01823009  0.995
1691 0.01831858 2.153468   206
## [16] {team_home=indianapolis colts,
##       winner=indianapolis colts,
##       winner_region=South}                   => {favorite_won=FALSE} 0.01699115  0.994
8187 0.01707965 2.152710   192
## [17] {team_home=indianapolis colts,
##       winner=indianapolis colts,
##       region_home=South}                     => {favorite_won=FALSE} 0.01699115  0.994
```

```
8187 0.01707965 2.152710   192
## [18] {team_home=indianapolis colts,
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts}           => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
## [19] {team_home=indianapolis colts,
##       weather_temperature=[72,97],
##       winner=indianapolis colts}           => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
## [20] {winner=indianapolis colts,
##       region_home=South,
##       winner_region=South}                 => {favorite_won=FALSE} 0.02150442  0.991
8367 0.02168142 2.146257   243
## [21] {winner=indianapolis colts,
##       region_away=South,
##       winner_region=South}                 => {favorite_won=FALSE} 0.01911504  0.990
8257 0.01929204 2.144069   216
## [22] {weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       winner_region=South}                 => {favorite_won=FALSE} 0.02000000  0.991
2281 0.02017699 2.144940   226
## [23] {weather_temperature=[72,97],
##       winner=indianapolis colts,
##       winner_region=South}                 => {favorite_won=FALSE} 0.02159292  0.991
8699 0.02176991 2.146329   244
## [24] {over_under_line=[44,63.5],
##       winner=indianapolis colts,
##       winner_region=South}                 => {favorite_won=FALSE} 0.01707965  0.994
8454 0.01716814 2.152768   193
## [25] {weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       region_home=South}                   => {favorite_won=FALSE} 0.01884956  0.990
6977 0.01902655 2.143792   213
## [26] {weather_temperature=[72,97],
##       winner=indianapolis colts,
##       region_home=South}                   => {favorite_won=FALSE} 0.01938053  0.990
9502 0.01955752 2.144339   219
## [27] {weather_temperature=[72,97],
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts}           => {favorite_won=FALSE} 0.01955752  0.991
0314 0.01973451 2.144514   221
## [28] {team_away=new england patriots,
##       winner=new england patriots,
##       region_away=East}                    => {favorite_won=FALSE} 0.01725664  0.994
8980 0.01734513 2.152881   195
## [29] {team_away=new england patriots,
##       winner=new england patriots,
##       winner_region=East}                  => {favorite_won=FALSE} 0.01725664  0.994
8980 0.01734513 2.152881   195
## [30] {spread_favorite=[-26.5,-6.5),
##       winner=new england patriots,
##       winner_region=East}                  => {favorite_won=FALSE} 0.01716814  1.000
```

```
0000 0.01716814 2.163922   194
## [31] {schedule_season=[2.01e+03,2.02e+03],
##       winner=new england patriots,
##       winner_region=East}                  => {favorite_won=FALSE} 0.01504425  1.000
0000 0.01504425 2.163922   170
## [32] {score_away=[24,59],
##       winner=new england patriots,
##       winner_region=East}                  => {favorite_won=FALSE} 0.01522124  0.994
2197 0.01530973 2.151414   172
## [33] {over_under_line=[44,63.5],
##       winner=new england patriots,
##       winner_region=East}                  => {favorite_won=FALSE} 0.01823009  0.995
1691 0.01831858 2.153468   206
## [34] {team_home=indianapolis colts,
##       winner=indianapolis colts,
##       region_home=South,
##       winner_region=South}                 => {favorite_won=FALSE} 0.01699115  0.994
8187 0.01707965 2.152710   192
## [35] {team_home=indianapolis colts,
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       winner_region=South}                 => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
## [36] {team_home=indianapolis colts,
##       weather_temperature=[72,97],
##       winner=indianapolis colts,
##       winner_region=South}                 => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
## [37] {team_home=indianapolis colts,
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       region_home=South}                   => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
## [38] {team_home=indianapolis colts,
##       weather_temperature=[72,97],
##       winner=indianapolis colts,
##       region_home=South}                   => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
## [39] {team_home=indianapolis colts,
##       weather_temperature=[72,97],
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts}           => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
## [40] {weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       region_home=South,
##       winner_region=South}                 => {favorite_won=FALSE} 0.01884956  0.990
6977 0.01902655 2.143792   213
## [41] {weather_temperature=[72,97],
##       winner=indianapolis colts,
##       region_home=South,
##       winner_region=South}                 => {favorite_won=FALSE} 0.01938053  0.990
```

```
9502 0.01955752 2.144339   219
## [42] {weather_temperature=[72,97],
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       winner_region=South}                  => {favorite_won=FALSE} 0.01955752  0.991
0314 0.01973451 2.144514   221
## [43] {weather_temperature=[72,97],
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       region_home=South}                    => {favorite_won=FALSE} 0.01858407  0.990
5660 0.01876106 2.143508   210
## [44] {team_away=new england patriots,
##       winner=new england patriots,
##       region_away=East,
##       winner_region=East}                   => {favorite_won=FALSE} 0.01725664  0.994
8980 0.01734513 2.152881   195
## [45] {team_home=indianapolis colts,
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       region_home=South,
##       winner_region=South}                  => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
## [46] {team_home=indianapolis colts,
##       weather_temperature=[72,97],
##       winner=indianapolis colts,
##       region_home=South,
##       winner_region=South}                  => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
## [47] {team_home=indianapolis colts,
##       weather_temperature=[72,97],
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       winner_region=South}                  => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
## [48] {team_home=indianapolis colts,
##       weather_temperature=[72,97],
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       region_home=South}                    => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
## [49] {weather_temperature=[72,97],
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       region_home=South,
##       winner_region=South}                  => {favorite_won=FALSE} 0.01858407  0.990
5660 0.01876106 2.143508   210
## [50] {team_home=indianapolis colts,
##       weather_temperature=[72,97],
##       weather_wind_mph=[0,4),
##       winner=indianapolis colts,
##       region_home=South,
```

```
##       winner_region=South}                    => {favorite_won=FALSE} 0.01690265  0.994
7917 0.01699115 2.152651   191
```

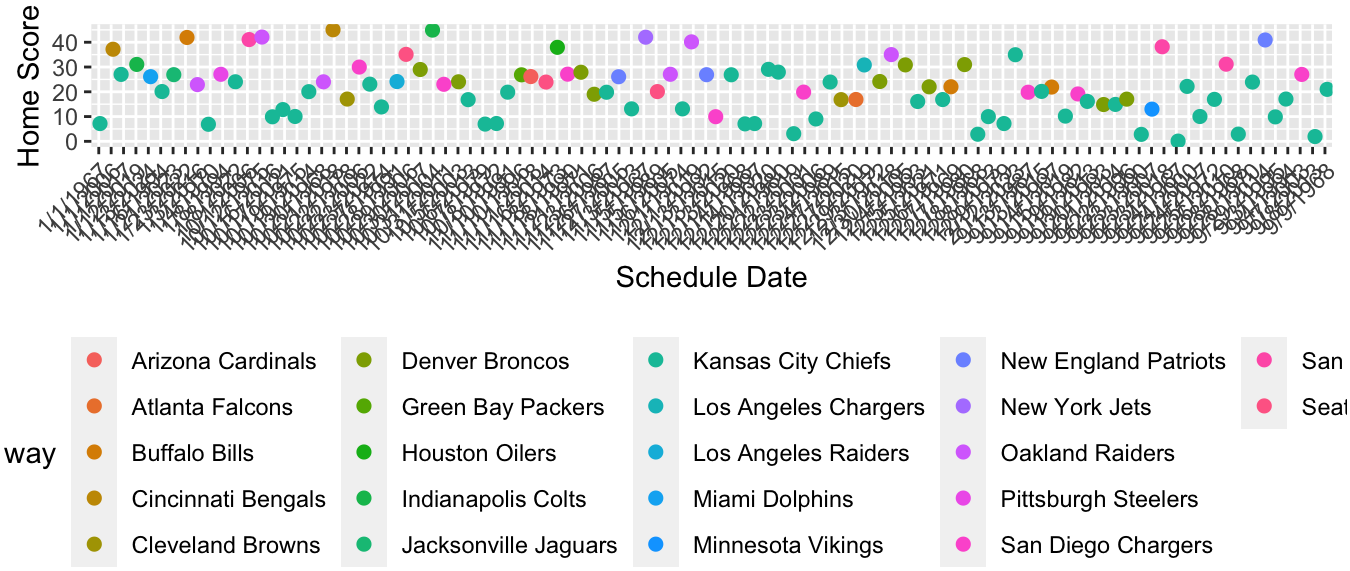
```
Stadiums<- stadiums_complete
Teams<- nfl_teams
Scores<- spreadspoke_scores
```

```
#Removing the Noise
Stadiums <- Stadiums %>%
  select(stadium_name, stadium_open, stadium_close, stadium_type, stadium_weather_type,
stadium_capacity, stadium_surface, stadium_latitude, stadium_longitude, stadium_azimutha
ngle, stadium_elevation) %>%
  mutate(
    stadium_name = gsub(" Stadium", "", stadium_name),
    stadium_type = ifelse(stadium_type == "indoor", "indoor", "outdoor"),
    stadium_weather_type = case_when(
      grepl("cold", stadium_weather_type, ignore.case = TRUE) ~ "cold",
      grepl("moderate", stadium_weather_type, ignore.case = TRUE) ~ "moderate",
      grepl("warm", stadium_weather_type, ignore.case = TRUE) ~ "warm",
      grepl("indoor", stadium_weather_type, ignore.case = TRUE) ~ "indoor",
      TRUE ~ NA_character_
    ),
    stadium_surface = case_when(
      grepl("Grass", stadium_surface, ignore.case = TRUE) ~ "Grass",
      grepl("FieldTurf", stadium_surface, ignore.case = TRUE) ~ "FieldTurf",
      TRUE ~ NA_character_
    ),
    stadium_latitude = as.numeric(stadium_latitude),
    stadium_longitude = as.numeric(stadium_longitude),
    stadium_azimuthangle = as.numeric(stadium_azimuthangle),
    stadium_elevation = as.numeric(stadium_elevation)
  ) %>%
  mutate(across(where(is.character), ~na_if(.x, ""))) %>%
  mutate(across(where(is.numeric), ~na_if(.x, 0))) %>%
  filter(is.na(stadium_close) | stadium_close == "")

#Converting variables
Stadiums$stadium_latitude<- as.numeric(Stadiums$stadium_latitude)
Stadiums$stadium_longitude<- as.numeric(Stadiums$stadium_longitude)
Stadiums$stadium_elevation<- as.numeric(Stadiums$stadium_elevation)
Scores$weather_temperature<- as.numeric(Scores$weather_temperature)
Scores$score_home<- as.numeric(Scores$score_home)
Scores$score_away<- as.numeric(Scores$score_away)
Scores$weather_wind_mph<- as.numeric(Scores$weather_wind_mph)
Scores$weather_humidity<- as.numeric(Scores$weather_humidity)

#Creating a column with the winner name
winner<- ifelse(is.na(Scores$score_home) | is.na(Scores$score_away), "Missing",
                ifelse(Scores$score_home > Scores$score_away, Scores$team_home,
                       ifelse(Scores$score_home < Scores$score_away, Scores$team_away,
"Tie")))

#Adding winner column to the scores dataset
Scores$winner<- winner
```

STEP FOUR: INTRODUCTORY INVESTIGATION

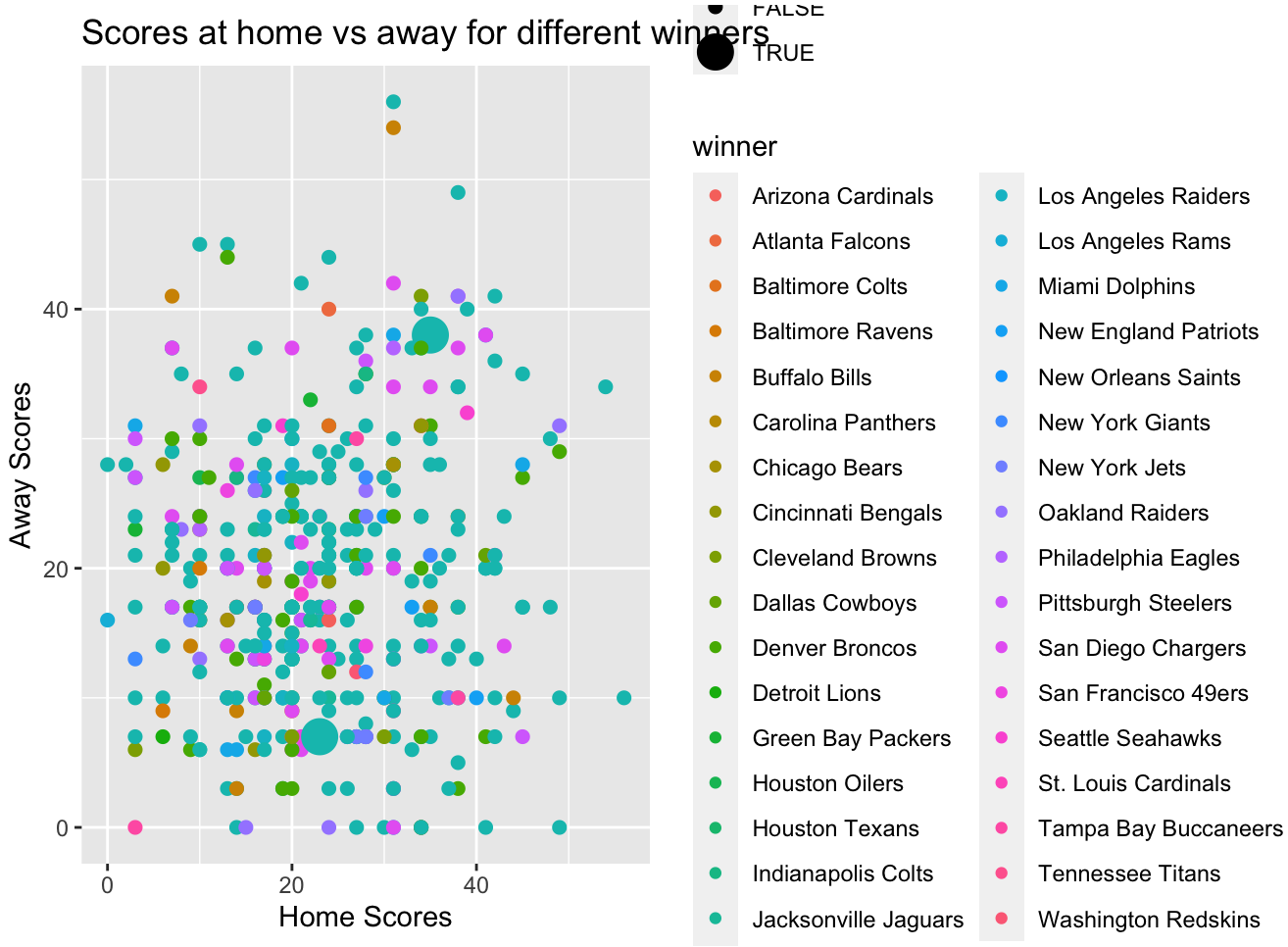
{CORRELATION MATRIX AND HEAT MAP}

```
#Sample 25 rows and sort by 'schedule_date' in descending order
samp_sort<- Scores%>% sample_n(25)%>% arrange(desc(schedule_date))

#Sample 25 rows of Specific columns
samp_col<- Scores%>% select(team_home, score_home, score_away, team_away, winner)%>% sam
ple_n(25)

#Filter Scores for Kansas City Chiefs and sort by schedule date in descending order
KCC<- Scores%>% filter(team_home == 'Kansas City Chiefs'| team_away== 'Kansas City Chief
s')%>% arrange(desc(schedule_date))

#Fixing missing values
KCC<- as.matrix(KCC)
KCC<- KCC[complete.cases(KCC),]
KCC<- as.data.frame(KCC)
KCC$weather_temperature<- as.numeric(KCC$weather_temperature)
KCC$score_home<- as.numeric(KCC$score_home)
KCC$score_away<- as.numeric(KCC$score_away)
KCC$weather_wind_mph<- as.numeric(KCC$weather_wind_mph)
KCC$weather_humidity<- as.numeric(KCC$weather_humidity)

#Sample of Kansas City Chiefs wins
KCCWins<-Scores%>%filter(Scores$winner== 'Kansas City Chiefs')%>% arrange(desc(schedule_
date))
KCCHW<-KCCWins%>%filter(KCCWins$team_home== 'Kansas City Chiefs')%>% arrange(desc(schedu
le_date))
HomeScores<- sample(KCCHW$score_home, 150, FALSE)
KCCAW<-KCCWins%>%filter(KCCWins$team_away== 'Kansas City Chiefs')%>% arrange(desc(schedu
le_date))
AwayScores<- sample(KCCAW$score_away, 150, FALSE)

#Create a categorical plot
KC_SampW<- KCCWins%>%sample_n(100)
KC_CP<- ggplot(KC_SampW, aes(x=schedule_date, y=score_home, color=team_away))+  geom_jit
ter(width = 0.2, height = 0.2, size = 2) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_y_continuous(limits = c(0, NA)) +
  labs(x = "Schedule Date", y = "Home Score") +
  theme(legend.position="bottom") +
  coord_fixed(ratio = 2/10)
print(KC_CP)
```




```
ggplot(KCC, aes(x=score_home, y=score_away, color= winner, size= stadium_neutral))+ geom
_point() +
labs(title = "Scores at home vs away for different winners", x="Home Scores",
y="Away Scores")
```

```
## Warning: Using size for a discrete variable is not advised.
```



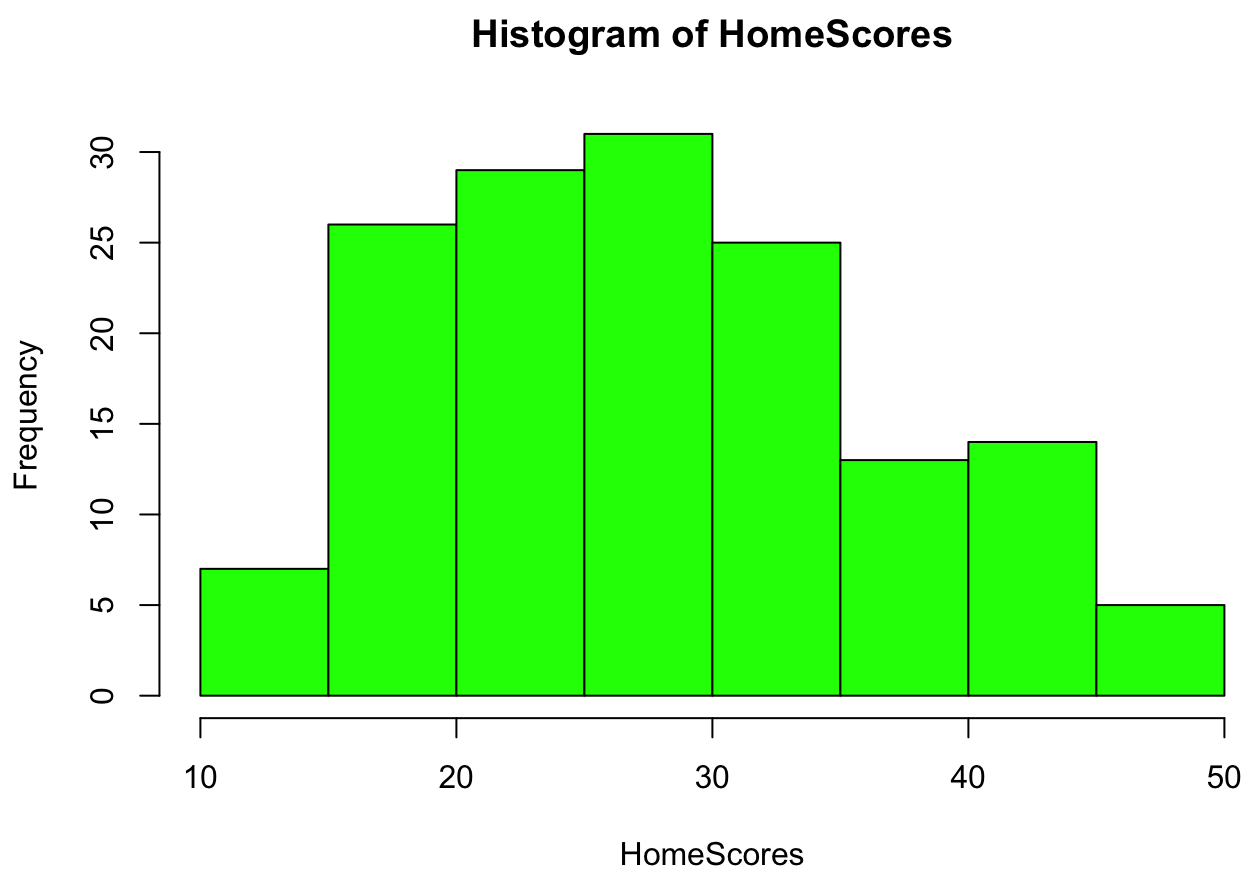

```
#Average of Home Scores vs Away Scores
mean(HomeScores)
```

```
## [1] 28.24
```

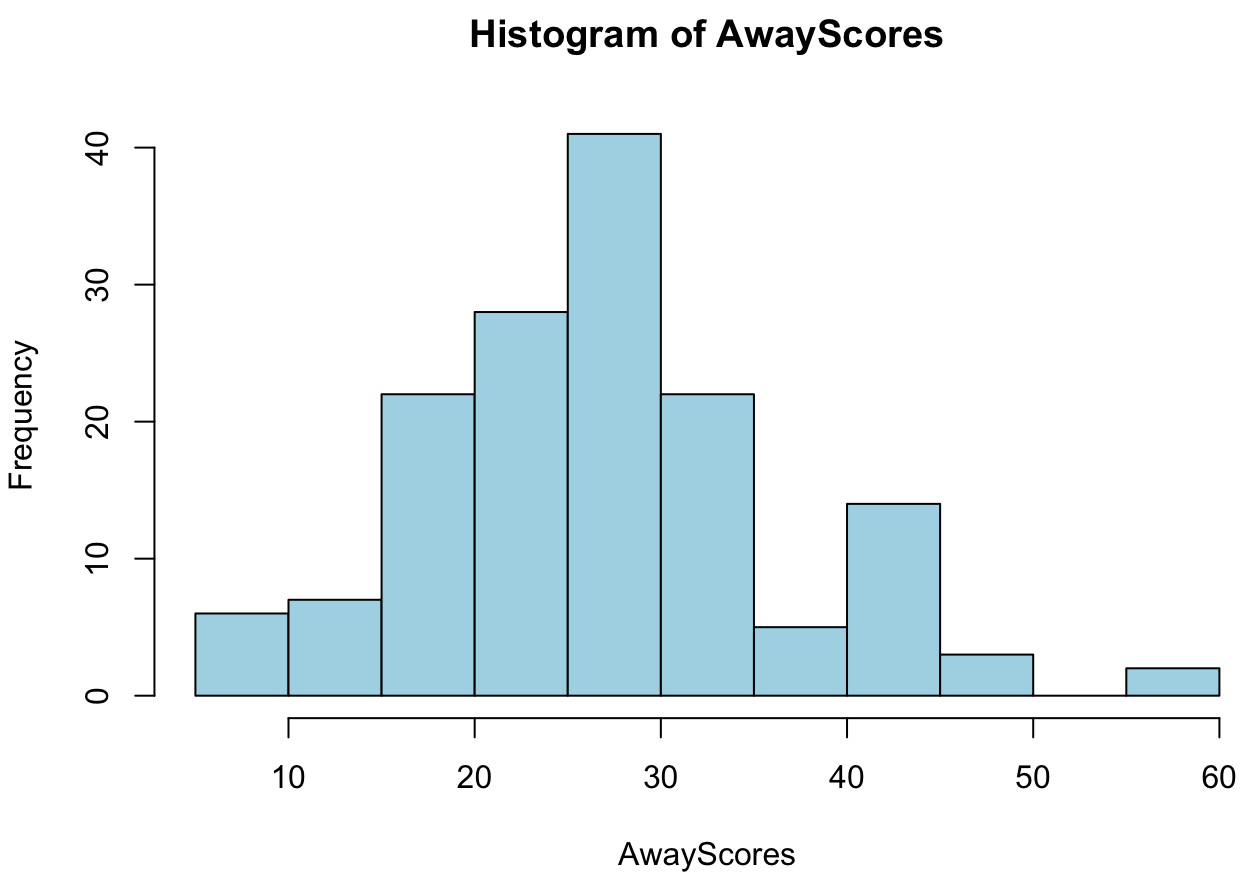
```
mean(AwayScores)
```

```
## [1] 27.41333
```

```
hist(HomeScores, col='green')
```

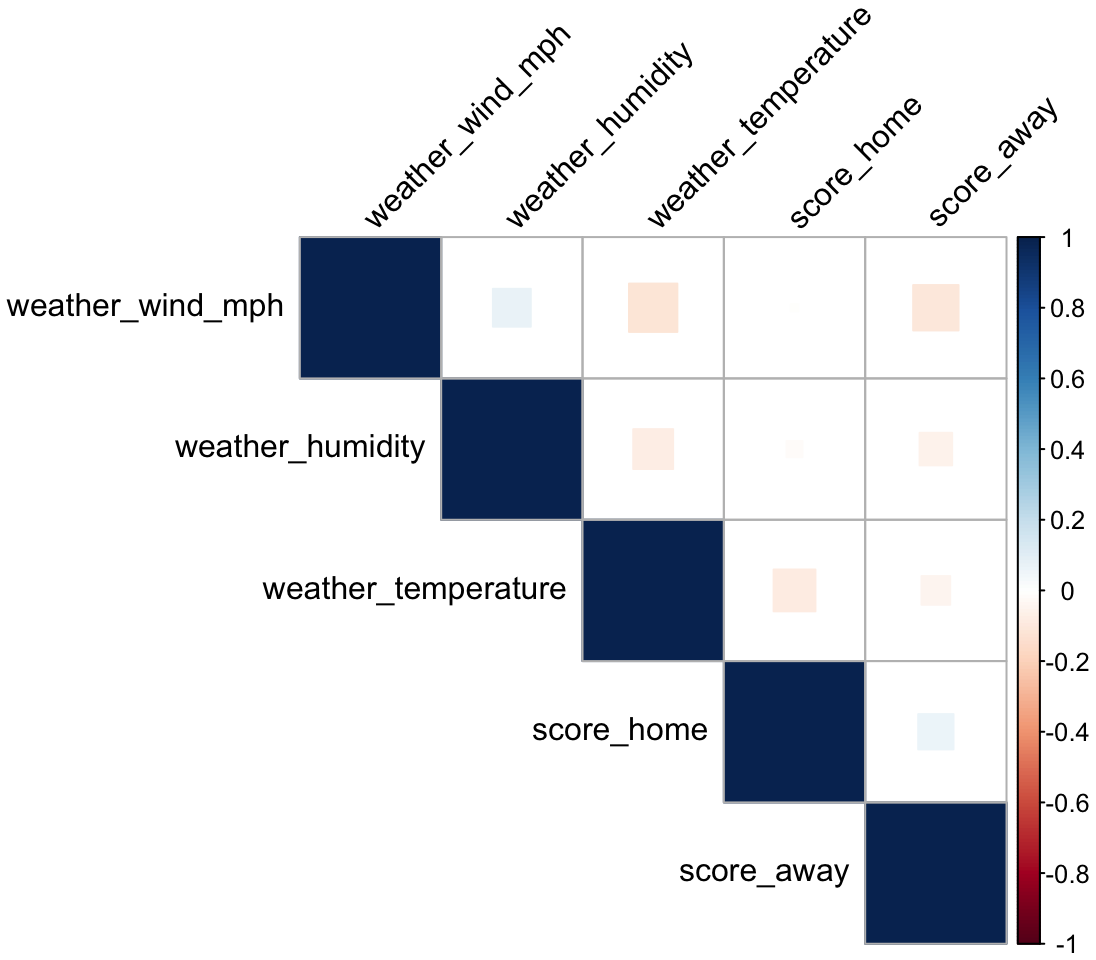



```
hist(AwayScores, col='lightblue')
```

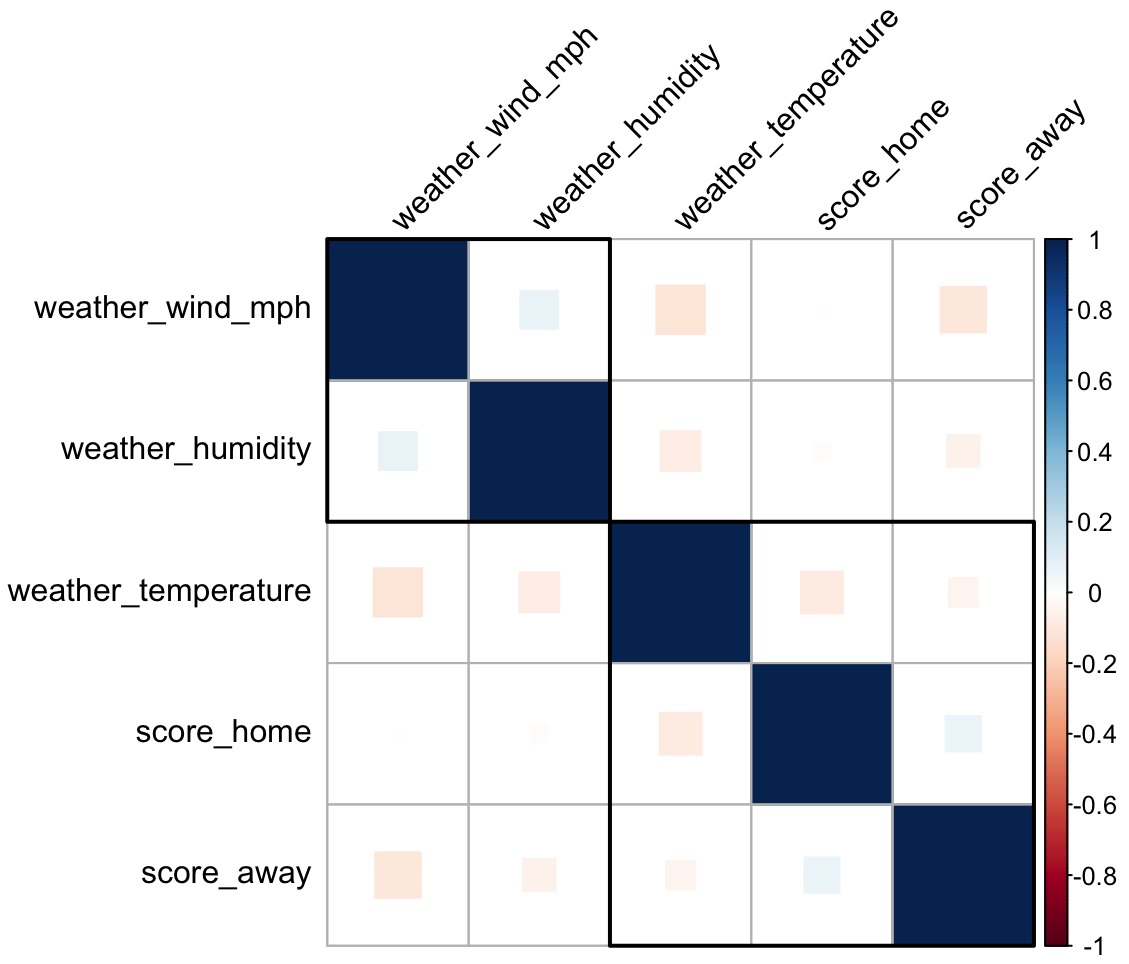



```
#Wins by Year
HWBY<-KCCHW%>%group_by(schedule_season)%>%count()
AWBY<-KCCAW%>%group_by(schedule_season)%>%count()
#All wins by year
All_BY<-KCCWins%>%group_by(schedule_season)%>%count()
```

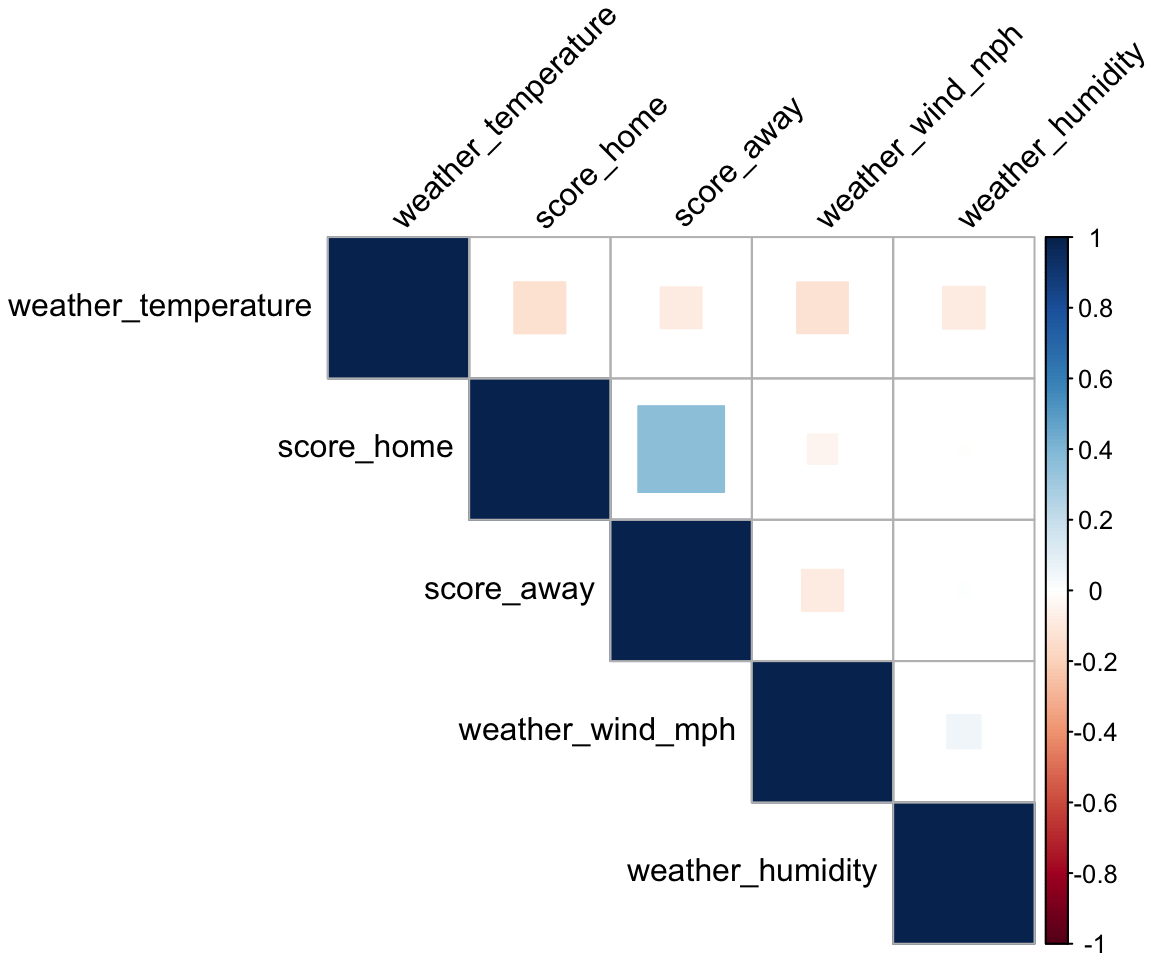
```
#Correlation matrix for Kansas City Chiefs
Kansas_corr<- cor(KCC%>% select_if(is.numeric))

#Plotting the heatmap
corrplot(Kansas_corr, method = "square", type = "upper", order = "hclust", tl.col = "bla
ck", tl.srt = 45, addrect = 2)
```



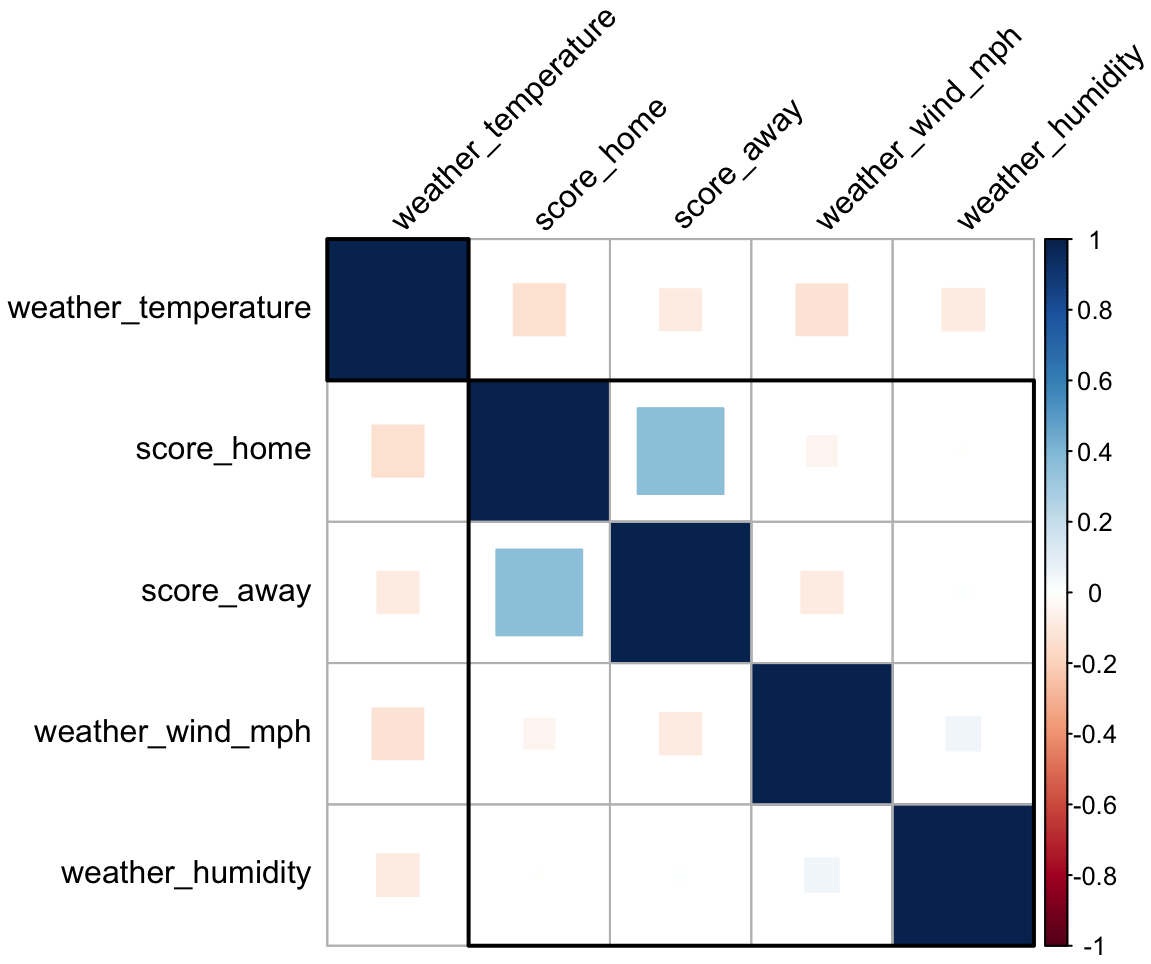

```
corrplot(Kansas_corr, method = "square", order = "hclust", tl.col = "black", tl.srt = 4
5, addrect = 2)
```




```
#Filter Kansas City Chiefs for wins
Kansas_wins<- KCC%>% filter(score_home > score_away)

#Correlation matrix for Kansas City Chiefs
Kansas_corr_wins<- cor(Kansas_wins%>% select_if(is.numeric))

#Plotting the heatmap for Kansas Wins
corrplot(Kansas_corr_wins, method = "square", type = "upper", order = "hclust", tl.col =
"black", tl.srt = 45, addrect = 2)
```




```
corrplot(Kansas_corr_wins, method = "square", order = "hclust", tl.col = "black", tl.srt
= 45, addrect = 2)
```




```
#adjusting the plot parameters
theme_set(theme_bw(base_size = 6))
```

```
suppressWarnings({



library(caTools)
set.seed(123)
split<- sample.split(Scores$winner, SplitRatio = 0.7)
train<- subset(KCC, split== TRUE)
train$outcome<- ifelse(train$winner=="Kansas City Chiefs", 1, 0)
train<- na.omit(train)
test<- subset(Scores, split== FALSE)
library(randomForest)
model<-randomForest(outcome ~ spread_favorite+ over_under_line+ weather_temperature+ wea
ther_wind_mph+ weather_humidity, data=train)
predictions<- predict(model, newdata= test, type= "response")
SB<- data.frame(
  team_home= "Kansas City Chiefs",
  team_away= "San Francisco 49ers",
  spread_favorite= -2.0,
  over_under_line= 47.5,
  weather_temperature= 33,
  weather_wind_mph= 29,
  weather_humidity=69)
SB_predict<- predict(model, newdata = SB)
SB_predict
predicted_winner<- ifelse(SB_predict== 1, "Kansas City Chiefs", "San Francisco 49ers")
predicted_winner

#Different way to conduct model
train$spread_favorite<-as.numeric(train$spread_favorite)
train$over_under_line<-as.numeric(train$over_under_line)
model2<-lm(outcome ~ spread_favorite+ over_under_line+ weather_temperature+ weather_wind
_mph+ weather_humidity, data=train)
predictions2<- predict(model2, newdata= test, type= "response")
SB_predict2<- predict(model2, newdata = SB)
SB_predict2
predicted_winner2<- ifelse(SB_predict2== 1, "Kansas City Chiefs", "San Francisco 49ers")
predicted_winner2

### Final Model #####
#Library packages
library(caret)
library(dplyr)

#Preprocessing in the datasets
colnames(KCC)[colnames(KCC) == "stadium"] <- "stadium_name"

#Merge data sets
split<- sample.split(KCC$winner, SplitRatio = 0.7)
KCC_Merg<- merge(KCC, Stadiums, by="stadium_name", all.x = TRUE)
train<- subset(KCC_Merg, split== TRUE)
train$outcome<- ifelse(train$winner=="Kansas City Chiefs", 1, 0)
```

```
test<- subset(KCC_Merg, split== TRUE)
library(randomForest)
model<-randomForest(outcome ~ spread_favorite + over_under_line + weather_temperature +
weather_wind_mph + weather_humidity + stadium_neutral + stadium_type + stadium_weather_t
ype + stadium_capacity + stadium_surface + stadium_elevation + stadium_azimuthangle, dat
a=train, na.action = na.omit)
predictions<- predict(model, newdata= test, type= "response")
SB<- data.frame(
  team_home = "Kansas City Chiefs",
  team_away = "San Francisco 49ers",
  spread_favorite = -2.0,
  over_under_line = 47.5,
  weather_temperature = 33,
  weather_wind_mph = 29,
  weather_humidity = 69,
  stadium_neutral = TRUE,
  stadium_type = "retractable",
  stadium_weather_type = "Indoor",
  stadium_capacity = 65000,
  stadium_surface = "FieldTurf",
  stadium_elevation = 2001,
  stadium_azimuthangle = 203)
SB_predict<- predict(model, newdata = SB)
SB_predict
predicted_winner<- ifelse(SB_predict== 1, "Kansas City Chiefs", "San Francisco 49ers")
predicted_winner

})
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##                     1
## "San Francisco 49ers"
```

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(wordcloud)
library(quanteda)
```

```
## Package version: 3.3.1
## Unicode version: 14.0
## ICU version: 71.1
```

```
## Parallel computing: 8 of 8 threads used.
```

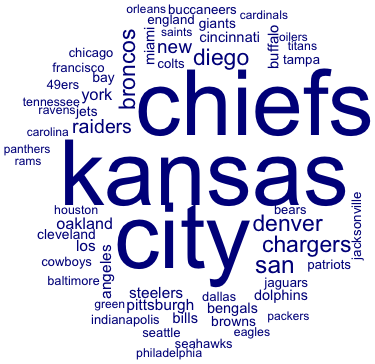
```
## See https://quanteda.io for tutorials and examples.
```

```
library(wordcloud2)
kcc_corp<- Corpus(VectorSource(KCC))
kcc_dtm<- DocumentTermMatrix(kcc_corp)

#wordcloud(kcc_dtm, random.order=FALSE, random.color=FALSE,cex =.1)
#wordcloud2(kcc_dtm, minSize = 2)

#-----------------------------------------------------------------------
kcc_matrix<-as.matrix(kcc_dtm)
kcc_df<- as.data.frame(kcc_matrix)
wordcloud2(kcc_df, minSize = 1)
```

```
#-----------------------------------------------------------------
#install.packages("quanteda.textplots")
library(quanteda.textplots)
#install.packages("quanteda.stopwords")
#library(quanteda.stopwords)
unique_teams<-unique(Teams$team_name)
Teams$id<-seq_along(unique_teams)
win_id<-merge(KCC, Teams, by.x = "winner", by.y = "team_name", all.x = TRUE)


CorpKCC<-corpus(win_id$winner)
KCCtokens<-tokens(CorpKCC, remove_punct = TRUE, remove_symbols=TRUE)
KCCstop<-tokens_select(KCCtokens, pattern = quanteda::stopwords("english"), selection =
"remove")
KCCDFM<-dfm(KCCstop)
KCCDFM
```

```
## Document-feature matrix of: 493 documents, 69 features (96.09% sparse) and 0 docvars.
##        features
## docs    arizona cardinals atlanta falcons baltimore colts ravens buffalo bills
##   text1       1         1       0       0         0     0      0       0     0
##   text2       0         0       1       1         0     0      0       0     0
##   text3       0         0       0       0         1     1      0       0     0
##   text4       0         0       0       0         1     0      1       0     0
##   text5       0         0       0       0         1     0      1       0     0
##   text6       0         0       0       0         1     0      1       0     0
##        features
## docs    carolina
##   text1        0
##   text2        0
##   text3        0
##   text4        0
##   text5        0
##   text6        0
## [ reached max_ndoc ... 487 more documents, reached max_nfeat ... 59 more features ]
```

```
textplot_wordcloud(KCCDFM, min_count = 2)
```




```
SC_win_id<-merge(Scores, Teams, by.x = "winner", by.y = "team_name", all.x = TRUE)
CorpSC<-corpus(SC_win_id$winner)
SCtokens<-tokens(CorpSC, remove_punct = TRUE, remove_symbols=TRUE)
SCstop<-tokens_select(SCtokens, pattern = quanteda::stopwords("english"), selection = "r
emove")
SCDFM<-dfm(SCstop)
SCDFM
```

```
## Document-feature matrix of: 13,801 documents, 80 features (97.07% sparse) and 0 docva
rs.
##        features
## docs    arizona cardinals atlanta falcons baltimore colts ravens boston
##   text1       1         1       0       0         0     0      0      0
##   text2       1         1       0       0         0     0      0      0
##   text3       1         1       0       0         0     0      0      0
##   text4       1         1       0       0         0     0      0      0
##   text5       1         1       0       0         0     0      0      0
##   text6       1         1       0       0         0     0      0      0
##        features
## docs    patriots buffalo
##   text1        0       0
##   text2        0       0
##   text3        0       0
##   text4        0       0
##   text5        0       0
##   text6        0       0
## [ reached max_ndoc ... 13,795 more documents, reached max_nfeat ... 70 more features
]
```

```
textplot_wordcloud(SCDFM, min_count = 2)
```

```
#Loading in data and libraries
#stadiums_final <- read.csv("~/Downloads/stadiums_final.csv")
#nfl_teams <- read.csv("~/Downloads/nfl_teams.csv", header=TRUE)
#spreadspoke_scores <- read.csv("~/Downloads/spreadspoke_scores.csv")
stadiums_final <- stadiums_complete
library(tidyverse)
library(ggplot2)
```

Viewing preliminary data

```
nfl_teams
```

team_name <chr>	team_name_short <chr>	team_id <chr>	team_id_pfr <chr>	team_conference <chr>	team_d <chr>
Arizona Cardinals	Cardinals	ARI	CRD	NFC	NFC We
Atlanta Falcons	Falcons	ATL	ATL	NFC	NFC So
Baltimore Colts	Colts	IND	CLT	AFC	
Baltimore Ravens	Ravens	BAL	RAV	AFC	AFC No
Boston Patriots	Patriots	NE	NWE	AFC	
Buffalo Bills	Bills	BUF	BUF	AFC	AFC Eas
Carolina Panthers	Panthers	CAR	CAR	NFC	NFC So
Chicago Bears	Bears	CHI	CHI	NFC	NFC No
Cincinnati Bengals	Bengals	CIN	CIN	AFC	AFC No
Cleveland Browns	Browns	CLE	CLE	AFC	AFC No

1-10 of 44 rows | 1-6 of 8 columns Previous **1** 2 3 4 5 Next

```
spreadspoke_scores
```

schedule_date <chr>	schedule_season <int>	schedule_week <chr>	schedule_playoff <lgl>	team_home <chr>	sc
9/2/1966	1966	1	FALSE	Miami Dolphins	
9/3/1966	1966	1	FALSE	Houston Oilers	
9/4/1966	1966	1	FALSE	San Diego Chargers	
9/9/1966	1966	2	FALSE	Miami Dolphins	
9/10/1966	1966	1	FALSE	Green Bay Packers	
9/10/1966	1966	2	FALSE	Houston Oilers	
9/10/1966	1966	2	FALSE	San Diego Chargers	

schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	sc
<chr>	<int>	<chr>	<lgl>	<chr>	
9/11/1966	1966	1	FALSE	Atlanta Falcons	
9/11/1966	1966	2	FALSE	Buffalo Bills	
9/11/1966	1966	1	FALSE	Detroit Lions	

1-10 of 10,000 rows | 1-6 of 19 columns Previous **1** 2 3 4 5 6 … 1000 Next

Ideas for initial data analysis:

- teams with most wins in warm weather
    - create warm weather subset
- teams with most wins in cold weather
    - create cold weather subset
- favorite team id

Use clustering on scores ?

Adding new column with winner

```
#create new column with winner
spreadspoke_scores$winner <- with(spreadspoke_scores, ifelse(spreadspoke_scores$score_ho
me > spreadspoke_scores$score_away, spreadspoke_scores$team_home, spreadspoke_scores$tea
m_away))
#view new df
spreadspoke_scores
```

schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	sc
<chr>	<int>	<chr>	<lgl>	<chr>	
9/2/1966	1966	1	FALSE	Miami Dolphins	
9/3/1966	1966	1	FALSE	Houston Oilers	
9/4/1966	1966	1	FALSE	San Diego Chargers	
9/9/1966	1966	2	FALSE	Miami Dolphins	
9/10/1966	1966	1	FALSE	Green Bay Packers	
9/10/1966	1966	2	FALSE	Houston Oilers	
9/10/1966	1966	2	FALSE	San Diego Chargers	
9/11/1966	1966	1	FALSE	Atlanta Falcons	
9/11/1966	1966	2	FALSE	Buffalo Bills	
9/11/1966	1966	1	FALSE	Detroit Lions	

1-10 of 10,000 rows | 1-6 of 19 columns Previous **1** 2 3 4 5 6 … 1000 Next

```
#looking at games with a temperature of over 65
warm<-spreadspoke_scores[spreadspoke_scores$weather_temperature > 65,]
#omitting all rows with NAs
warm<-na.omit(warm)
warm
```

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home
	<chr>	<int>	<chr>	<lgl>	<chr>
539	1/12/1969	1968	Superbowl	TRUE	Baltimore Colts
2501	1/21/1979	1978	Superbowl	TRUE	Dallas Cowboys
2502	9/1/1979	1979	1	FALSE	Tampa Bay Buccaneers
2503	9/2/1979	1979	1	FALSE	Buffalo Bills
2504	9/2/1979	1979	1	FALSE	Chicago Bears
2505	9/2/1979	1979	1	FALSE	Denver Broncos
2506	9/2/1979	1979	1	FALSE	Kansas City Chiefs
2507	9/2/1979	1979	1	FALSE	Los Angeles Rams
2508	9/2/1979	1979	1	FALSE	Minnesota Vikings
2510	9/2/1979	1979	1	FALSE	New York Jets

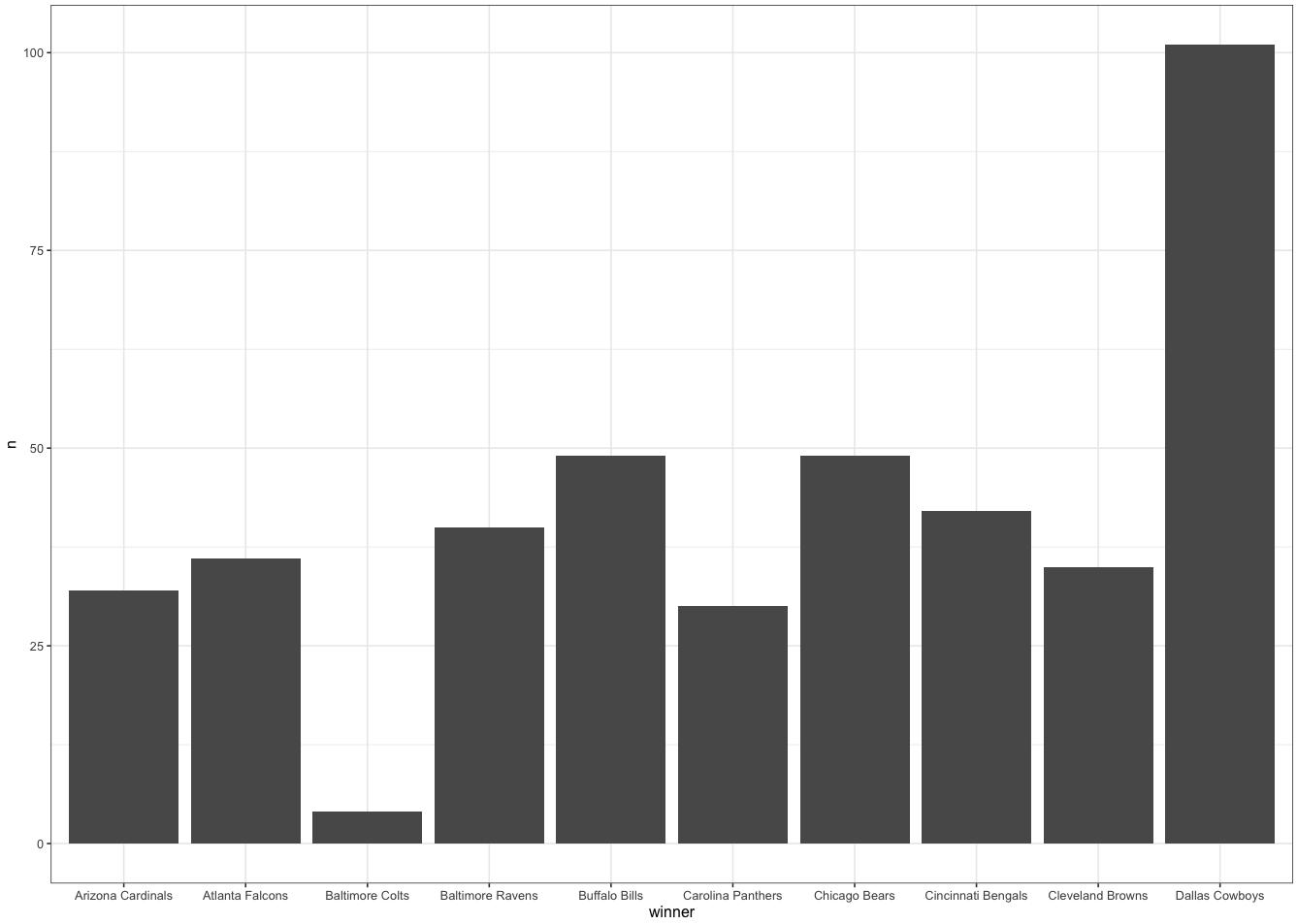
1-10 of 1,746 rows | 1-7 of 20 columns Previous **1** 2 3 4 5 6 … 175 Next

```
warm_count<-warm%>%count(winner)
#finding top home wins for warm weather
warm_count %>% arrange(desc(n))
```

winner	n
<chr>	<int>
Miami Dolphins	176
Tampa Bay Buccaneers	120
Dallas Cowboys	101
San Diego Chargers	91
New York Jets	66
New York Giants	64
Jacksonville Jaguars	63
Philadelphia Eagles	61
Denver Broncos	57
Washington Redskins	57

1-10 of 39 rows Previous **1** 2 3 4 Next

```
#top 10 to ggplot
warm_count_10<-warm_count[1:10,]
#visualizing top 10
ggplot(data=warm_count_10, aes(x = winner, y=n)) +
  geom_col(position=position_dodge())
```



Looking at teams with high winds (over 15 mph)

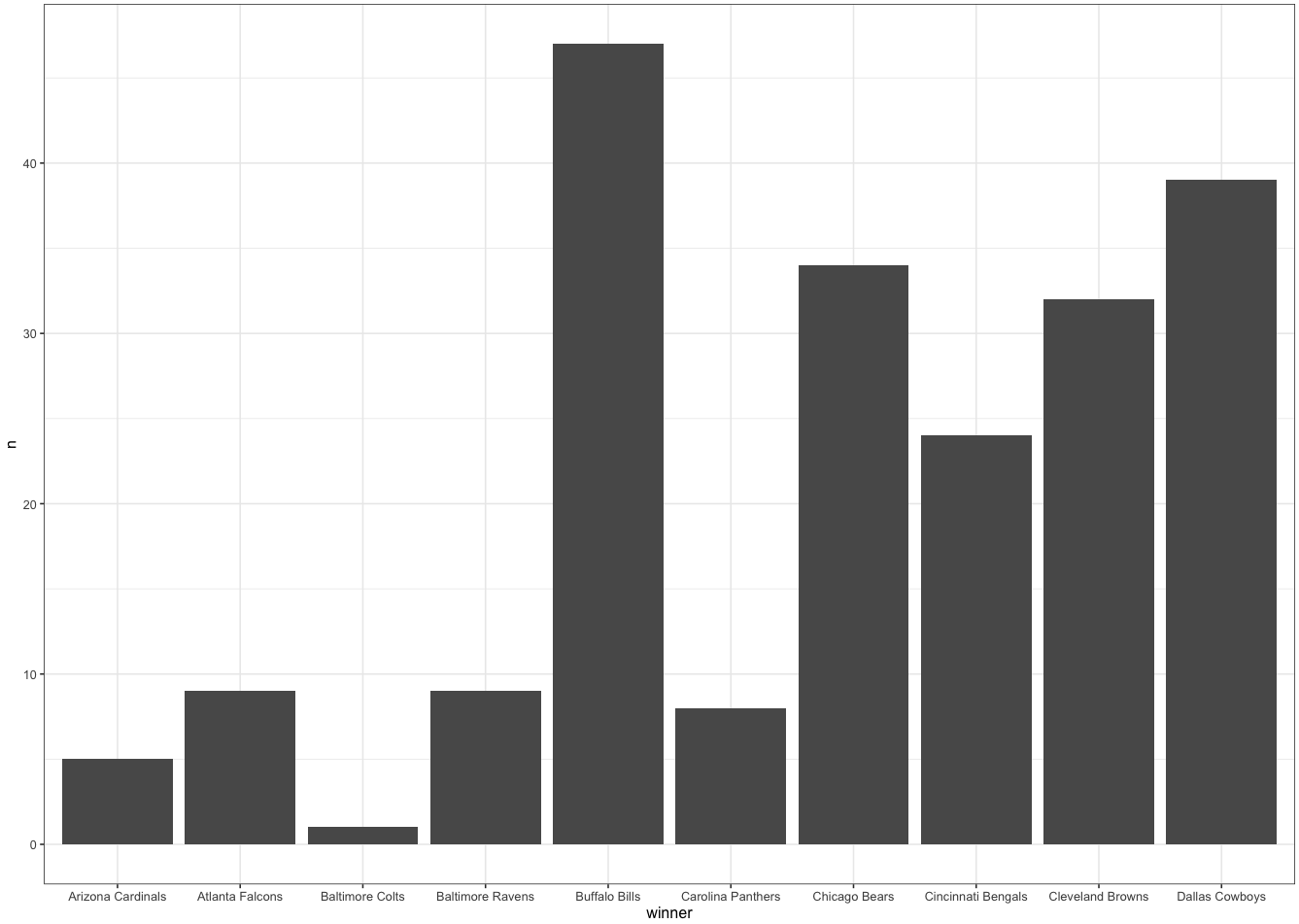
```
windy<-spreadspoke_scores[spreadspoke_scores$weather_wind_mph > 15,]
#deleting all rows with NAs
windy<-na.omit(windy)
windy_count<-windy%>%count(winner)
windy_count%>%arrange(desc(n))
```

winner <chr>	**n** <int>
New York Giants	61
New England Patriots	51
New York Jets	49

winner <chr>	n <int>
Buffalo Bills	47
Philadelphia Eagles	40
Dallas Cowboys	39
Miami Dolphins	37
Kansas City Chiefs	36
Chicago Bears	34
Cleveland Browns	32

1-10 of 38 rows Previous **1** 2 3 4 Next

```
#take top 10
windy_count_10<-windy_count[1:10,]
ggplot(data=windy_count_10, aes(x = winner, y=n)) +
  geom_col(position=position_dodge())
```



K-means clustering

```
#testing new columns
test_ss<-spreadspoke_scores

test_ss$winner_score <- with(test_ss, ifelse(score_home > score_away, score_home, score_
away))
test_ss$loser_score <- with(test_ss, ifelse(score_home < score_away, score_home, score_a
way))
test_ss$spread <- test_ss$winner_score - test_ss$loser_score
test_ss$combined_score <- test_ss$winner_score + test_ss$loser_score


#adding columns to df
spreadspoke_scores$winner_score <- with(spreadspoke_scores, ifelse(score_home > score_aw
ay, score_home, score_away))
spreadspoke_scores$loser_score <- with(spreadspoke_scores, ifelse(score_home < score_awa
y, score_home, score_away))
spreadspoke_scores$spread <- spreadspoke_scores$winner_score - spreadspoke_scores$loser_
score
spreadspoke_scores$combined_score <- spreadspoke_scores$winner_score + spreadspoke_score
s$loser_score


#create new column with loser
spreadspoke_scores$loser <- with(spreadspoke_scores, ifelse(spreadspoke_scores$score_hom
e < spreadspoke_scores$score_away, spreadspoke_scores$team_home, spreadspoke_scores$team
_away))
spreadspoke_scores
```

schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	sc
<chr>	<int>	<chr>	<lgl>	<chr>	
9/2/1966	1966	1	FALSE	Miami Dolphins	
9/3/1966	1966	1	FALSE	Houston Oilers	
9/4/1966	1966	1	FALSE	San Diego Chargers	
9/9/1966	1966	2	FALSE	Miami Dolphins	
9/10/1966	1966	1	FALSE	Green Bay Packers	
9/10/1966	1966	2	FALSE	Houston Oilers	
9/10/1966	1966	2	FALSE	San Diego Chargers	
9/11/1966	1966	1	FALSE	Atlanta Falcons	
9/11/1966	1966	2	FALSE	Buffalo Bills	
9/11/1966	1966	1	FALSE	Detroit Lions	

1-10 of 10,000 rows | 1-6 of 24 columns Previous **1** 2 3 4 5 6 … 1000 Next

Remove unnecessary columns

```
ss_cluster_clean<-spreadspoke_scores[,-1:-10]
ss_cluster_clean<-ss_cluster_clean[,-2:-3]
ss_cluster_clean<-ss_cluster_clean[,-5]
ss_cluster_clean
```

over_under_line <dbl>	weather_temperature <int>	weather_wind_mph <int>	weather_humidity <int>	winner <chr>
NA	83	6	71	Oakland Raiders
NA	81	7	70	Houston Oilers
NA	70	7	82	San Diego Charg
NA	82	11	78	New York Jets
NA	64	8	62	Green Bay Packe
NA	77	6	82	Houston Oilers
NA	69	9	81	San Diego Charg
NA	71	7	57	Los Angeles Ran
NA	63	11	73	Kansas City Chie
NA	67	7	73	Detroit Lions

1-10 of 10,000 rows | 1-5 of 11 columns Previous **1** 2 3 4 5 6 … 1000 Next

DF for over_under_line

```
ss_over_cluster<-ss_cluster_clean[,-2:-4]
ss_over_cluster<-na.omit(ss_over_cluster)
ss_over_cluster<-ss_over_cluster[,c(2,7,1,6,3,4,5)]
#rownames(ss_over_cluster)<-ss_over_cluster[,2]
#row.names(ss_over_cluster)<-ss_over_cluster$winner

#subtracting over under line from combined score
ss_over_cluster$over_over_under_line<-ss_over_cluster$combined_score - ss_over_cluster$o
ver_under_line

#deleting winner/loser score and loser column - just looking at winners
ss_over_cluster<-ss_over_cluster[,-3]
ss_over_cluster<-ss_over_cluster[,-4:-5]
ss_over_cluster
```

	winner <chr>	combined_score <int>	spre… <int>	loser_score <int>	over_over_under_line <dbl>
351	Green Bay Packers	47	19	14	4.0
539	New York Jets	23	9	7	-17.0

	winner <chr>	combined_score <int>	spre… <int>	loser_score <int>	over_over_under_line <dbl>
728	Kansas City Chiefs	30	16	7	-9.0
917	Baltimore Colts	29	3	13	-7.0
1106	Dallas Cowboys	27	21	3	-7.0
1295	Miami Dolphins	21	7	7	-12.0
1484	Miami Dolphins	31	17	7	-2.0
1673	Pittsburgh Steelers	22	10	6	-11.0
1862	Pittsburgh Steelers	38	4	17	2.0
2065	Oakland Raiders	46	18	14	8.0

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

```
#saving with names
ss_over_cluster_names<-ss_over_cluster

#deleting names for clustering purposes
ss_over_cluster<-ss_over_cluster[,-1:-2]
ss_over_cluster
```

	spread <int>	loser_score <int>	over_over_under_line <dbl>
351	19	14	4.0
539	9	7	-17.0
728	16	7	-9.0
917	3	13	-7.0
1106	21	3	-7.0
1295	7	7	-12.0
1484	17	7	-2.0
1673	10	6	-11.0
1862	4	17	2.0
2065	18	14	8.0

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

load packages

```
#install.packages("stats")
library(stats)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WB
a
```

Taking mean of spread, combined score, amount over over-under line and scaling data

```
ss_over_cluster_names
```

	winner	combined_score	spre…	loser_score	over_over_under_line
	<chr>	<int>	<int>	<int>	<dbl>
351	Green Bay Packers	47	19	14	4.0
539	New York Jets	23	9	7	-17.0
728	Kansas City Chiefs	30	16	7	-9.0
917	Baltimore Colts	29	3	13	-7.0
1106	Dallas Cowboys	27	21	3	-7.0
1295	Miami Dolphins	21	7	7	-12.0
1484	Miami Dolphins	31	17	7	-2.0
1673	Pittsburgh Steelers	22	10	6	-11.0
1862	Pittsburgh Steelers	38	4	17	2.0
2065	Oakland Raiders	46	18	14	8.0

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

```
#remove loser column and old clustering results
ss_over_cluster_names2<-ss_over_cluster_names[,-4]
#group by winner and take mean
ss_over_cluster_mean<-ss_over_cluster_names2%>%group_by(winner)%>%summarise(across(c(com
bined_score, spread, over_over_under_line), list(mean=mean)))
ss_over_cluster_mean<-as.data.frame(ss_over_cluster_mean)
```

Now make the winner column row names:

```
rownames(ss_over_cluster_mean)<-ss_over_cluster_mean[,1]
ss_over_cluster_mean<-ss_over_cluster_mean[,-1]
ss_over_cluster_mean
```

	combined_score_mean <dbl>	spread_me... <dbl>	over_over_under_line_mean <dbl>
Arizona Cardinals	44.47826	9.512077	1.1690821
Atlanta Falcons	44.88073	10.587156	1.1681957
Baltimore Colts	42.23810	5.761905	2.4047619
Baltimore Ravens	40.91882	13.214022	0.0867158
Buffalo Bills	42.39785	12.274194	1.0860215
Carolina Panthers	42.11982	11.622120	-0.0806451
Chicago Bears	38.85352	11.461972	-0.6281690
Cincinnati Bengals	45.33861	11.420886	2.8085443
Cleveland Browns	40.74627	9.694030	0.4141791
Dallas Cowboys	43.79759	12.881928	0.8469879

1-10 of 43 rows Previous **1** 2 3 4 5 Next

Scaled data

```
#scale data

ss_over_cluster_mean_scaled<-as.data.frame(scale(ss_over_cluster_mean[,-4]))
ss_over_cluster_mean_scaled
```

	combined_score_mean <dbl>	spread_me... <dbl>	over_over_under_line_mean <dbl>
Arizona Cardinals	0.6833538396	-1.01000022	0.4058088
Atlanta Falcons	0.8565303805	-0.33121658	0.4051648
Baltimore Colts	-0.2805469996	-3.37778578	1.3035688
Baltimore Ravens	-0.8482064948	1.32733533	-0.3805641
Buffalo Bills	-0.2118077828	0.73394594	0.3454626
Carolina Panthers	-0.3314404581	0.32223935	-0.5021571
Chicago Bears	-1.7368651308	0.22112504	-0.8999504
Cincinnati Bengals	1.0535447392	0.19518430	1.5969294
Cleveland Browns	-0.9224517201	-0.89511893	-0.1426518
Dallas Cowboys	0.3904742128	1.11765740	0.1717973

1-10 of 43 rows Previous **1** 2 3 4 5 Next

Best K value: 6

```
#set seed
set.seed(22)
#drop author names for cluster
ss_mean_k_scaled6<-kmeans(ss_over_cluster_mean_scaled, 6, nstart = 25)
ss_over_cluster_mean_scaled$clusters<-as.factor(ss_mean_k_scaled6$cluster)
#view results
print(ss_mean_k_scaled6)
```

```
## K-means clustering with 6 clusters of sizes 10, 1, 14, 7, 9, 2
##
## Cluster means:
##   combined_score_mean spread_mean over_over_under_line_mean
## 1           0.6952988   0.9465020                 0.4086542
## 2           3.1668239  -1.6884673                 2.9393455
## 3          -0.2856670   0.3080806                -0.2610743
## 4           0.7976708  -0.2971177                 0.9362526
## 5          -1.1191662  -0.4309055                -1.1343831
## 6          -0.8158371  -3.0658538                 0.1424161
##
## Clustering vector:
##        Arizona Cardinals          Atlanta Falcons          Baltimore Colts
##                        4                        4                        6
##         Baltimore Ravens            Buffalo Bills        Carolina Panthers
##                        3                        3                        3
##            Chicago Bears       Cincinnati Bengals         Cleveland Browns
##                        5                        4                        5
##           Dallas Cowboys           Denver Broncos            Detroit Lions
##                        1                        3                        3
##        Green Bay Packers           Houston Oilers           Houston Texans
##                        1                        3                        5
##       Indianapolis Colts     Jacksonville Jaguars       Kansas City Chiefs
##                        3                        3                        1
##        Las Vegas Raiders     Los Angeles Chargers      Los Angeles Raiders
##                        2                        3                        5
##         Los Angeles Rams           Miami Dolphins        Minnesota Vikings
##                        1                        5                        4
##     New England Patriots       New Orleans Saints          New York Giants
##                        1                        1                        5
##            New York Jets          Oakland Raiders      Philadelphia Eagles
##                        3                        4                        1
##        Phoenix Cardinals      Pittsburgh Steelers       San Diego Chargers
##                        5                        3                        1
##      San Francisco 49ers         Seattle Seahawks      St. Louis Cardinals
##                        1                        3                        4
##           St. Louis Rams     Tampa Bay Buccaneers         Tennessee Oilers
##                        1                        5                        3
##         Tennessee Titans    Washington Commanders Washington Football Team
##                        4                        6                        5
##      Washington Redskins
##                        3
##
## Within cluster sum of squares by cluster:
## [1] 6.048075 0.000000 9.977960 3.345950 6.851902 3.464226
##  (between_SS / total_SS =  76.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

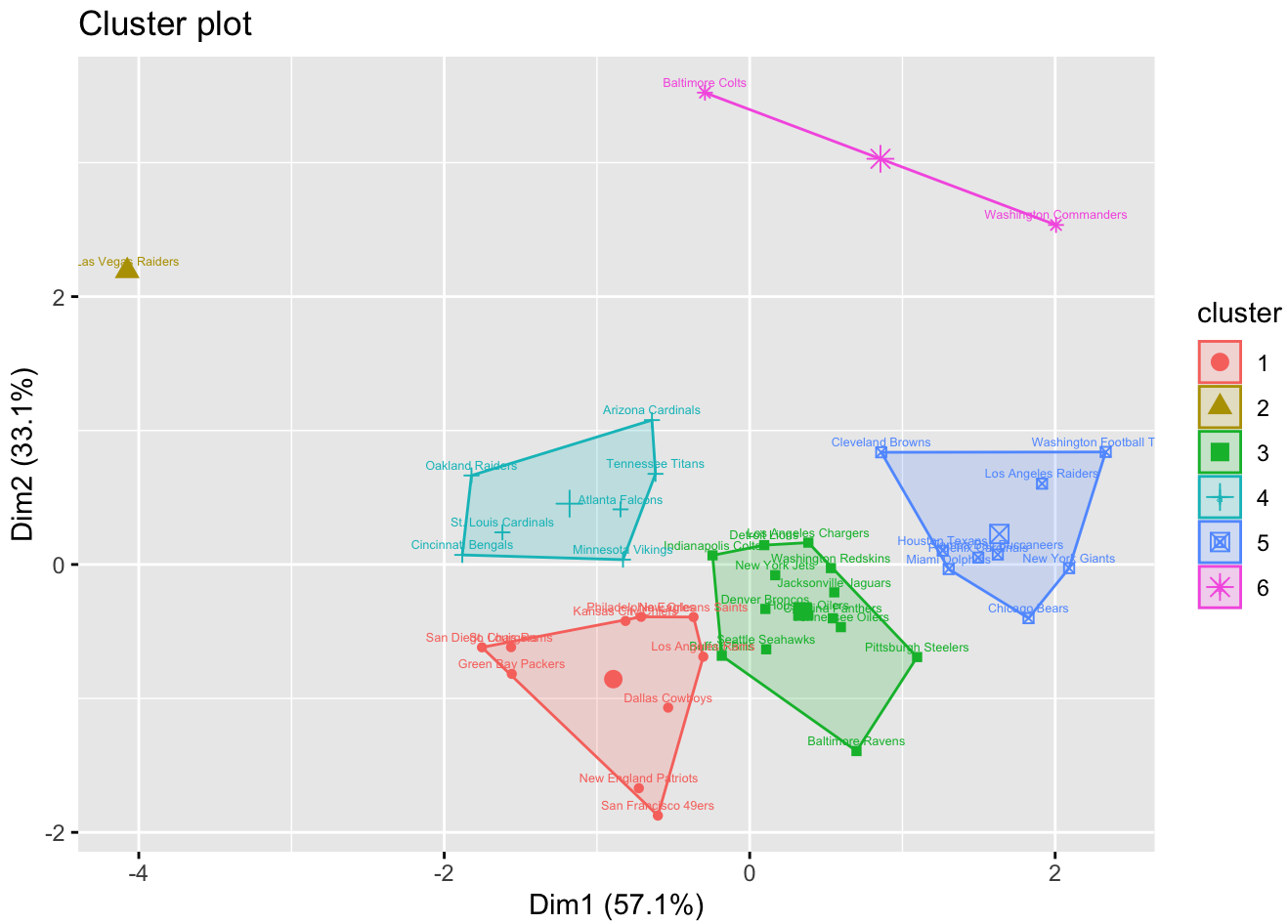
```
head(ss_over_cluster_mean_scaled)
```

	combined_score_mean <dbl>	spread_me... <dbl>	over_over_under_line_mean <dbl>	clusters <fct>
Arizona Cardinals	0.6833538	-1.0100002	0.4058088	4
Atlanta Falcons	0.8565304	-0.3312166	0.4051648	4
Baltimore Colts	-0.2805470	-3.3777858	1.3035689	6
Baltimore Ravens	-0.8482065	1.3273353	-0.3805642	3
Buffalo Bills	-0.2118078	0.7339459	0.3454627	3
Carolina Panthers	-0.3314405	0.3222393	-0.5021572	3

6 rows

visualize k=6

```
ss_mean_fviz_scaled<-ss_over_cluster_mean_scaled[,-4]
# Subset only numeric columns
ss_mean_fviz_scaled <- ss_mean_fviz_scaled[sapply(ss_mean_fviz_scaled, is.numeric)]

# Compute column means
col_means <- colMeans(ss_mean_fviz_scaled, na.rm = TRUE)

fviz_cluster(ss_mean_k_scaled6, data=ss_mean_fviz_scaled, labelsize = 5)
```



K=6 is best outcome for this clustering… how to analyze this?

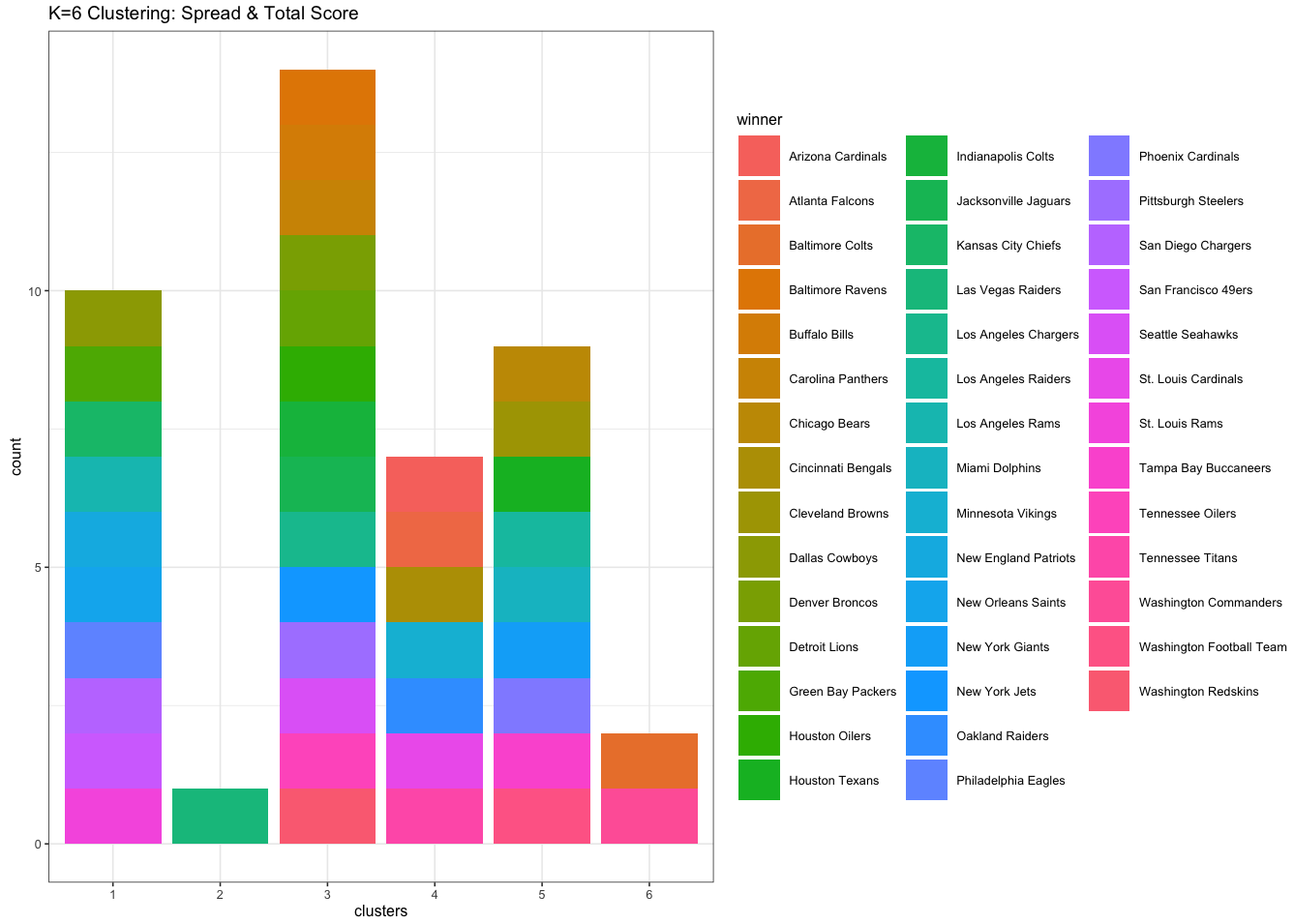
```
ss_over_cluster_mean_scaled6<-ss_over_cluster_mean_scaled
ss_over_cluster_mean_scaled6$clusters<-as.factor(ss_mean_k_scaled6$cluster)
ss_over_cluster_mean_scaled6$winner<-rownames(ss_over_cluster_mean_scaled6)
rownames(ss_over_cluster_mean_scaled6)<-NULL
ss_over_cluster_mean_scaled6
```

combined_score_mean <dbl>	spread_me… <dbl>	over_over_under_line_mean <dbl>	clusters <fct>	winner <chr>
0.6833538396	-1.01000022	0.40580882	4	Arizona Cardinals
0.8565303805	-0.33121658	0.40516482	4	Atlanta Falcons
-0.2805469996	-3.37778578	1.30356888	6	Baltimore Colts
-0.8482064948	1.32733533	-0.38056416	3	Baltimore Ravens
-0.2118077828	0.73394594	0.34546268	3	Buffalo Bills
-0.3314404581	0.32223935	-0.50215719	3	Carolina Panthers
-1.7368651308	0.22112504	-0.89995042	5	Chicago Bears
1.0535447392	0.19518430	1.59692944	4	Cincinnati Bengals
-0.9224517201	-0.89511893	-0.14265187	5	Cleveland Browns

combined_score_mean <dbl>	spread_me... <dbl>	over_over_under_line_mean <dbl>	clusters <fct>	winner <chr>
0.3904742128	1.11765740	0.17179732	1	Dallas Cowboys

1-10 of 43 rows Previous **1** 2 3 4 5 Next

```
ggplot(data=ss_over_cluster_mean_scaled6, aes(x=clusters, fill=winner)) + geom_bar()+labs(title="K=6 Clustering: Spread & Total Score")
```



Clustering by weather type

```
#start with clean cluster data
ss_weather_names<-select(ss_cluster_clean, weather_temperature, weather_wind_mph, weathe
r_humidity, winner)
#normalize data in columns
ss_weather_names<-na.omit(ss_weather_names)
ss_weather<-as.data.frame(scale(ss_weather_names[,-4]))
ss_weather$winner<-ss_weather_names$winner

#take mean value for each team
ss_weather_mean<-ss_weather%>%group_by(winner)%>%summarise(across(c(weather_temperature,
weather_wind_mph, weather_humidity), list(mean=mean)))
ss_weather_mean<-as.data.frame(ss_weather_mean)

#make winner column row names
rownames(ss_weather_mean)<-ss_weather_mean$winner
ss_weather_mean<-ss_weather_mean[,-1]
ss_weather_mean
```

	weather_temperature_mean <dbl>	weather_wind_mph_mean <dbl>	weather_
Arizona Cardinals	0.600014571	-0.638441469	
Atlanta Falcons	0.159492769	0.006317011	
Baltimore Colts	-0.203381348	0.059872836	
Baltimore Ravens	-0.077658992	-0.661934488	
Boston Patriots	-0.525116492	1.118079410	
Buffalo Bills	-0.290516531	0.323344501	
Carolina Panthers	0.113814323	-0.598605133	
Chicago Bears	-0.289935636	0.170519352	
Cincinnati Bengals	-0.325306529	-0.074975959	
Cleveland Browns	-0.297769370	0.269548986	

1-10 of 42 rows Previous **1** 2 3 4 5 Next

k-means

k=7

```
set.seed(22)
#drop author names for cluster
ss_weather_k7<-kmeans(ss_weather_mean, 7, nstart = 25)
ss_weather_mean$clusters<-as.factor(ss_weather_k7$cluster)
#view results
print(ss_weather_k7)
```

```
## K-means clustering with 7 clusters of sizes 2, 1, 7, 2, 4, 14, 12
##
## Cluster means:
##   weather_temperature_mean weather_wind_mph_mean weather_humidity_mean
## 1               -0.5782589          -0.29175471          -1.06413289
## 2               -1.0749601          -2.35100154          -0.20335902
## 3                0.1588705          -0.46385309          -0.14771557
## 4                0.7547871          -0.62233161          -1.25157911
## 5               -0.1744410           0.55172216          -0.18593689
## 6               -0.2554767           0.09116573           0.13900711
## 7                0.3615503          -0.10581366           0.09718899
##
## Clustering vector:
##        Arizona Cardinals          Atlanta Falcons          Baltimore Colts
##                        4                        7                        6
##         Baltimore Ravens          Boston Patriots            Buffalo Bills
##                        3                        5                        6
##        Carolina Panthers            Chicago Bears       Cincinnati Bengals
##                        3                        6                        6
##         Cleveland Browns           Dallas Cowboys           Denver Broncos
##                        6                        7                        1
##            Detroit Lions        Green Bay Packers           Houston Oilers
##                        6                        6                        6
##           Houston Texans       Indianapolis Colts     Jacksonville Jaguars
##                        3                        7                        7
##       Kansas City Chiefs        Las Vegas Raiders      Los Angeles Raiders
##                        6                        1                        7
##         Los Angeles Rams           Miami Dolphins        Minnesota Vikings
##                        7                        7                        6
##     New England Patriots       New Orleans Saints          New York Giants
##                        5                        7                        5
##            New York Jets          Oakland Raiders      Philadelphia Eagles
##                        5                        7                        6
##        Phoenix Cardinals      Pittsburgh Steelers       San Diego Chargers
##                        4                        6                        3
##      San Francisco 49ers         Seattle Seahawks      St. Louis Cardinals
##                        7                        6                        6
##           St. Louis Rams     Tampa Bay Buccaneers         Tennessee Oilers
##                        3                        7                        7
##         Tennessee Titans Washington Football Team      Washington Redskins
##                        3                        2                        3
##
## Within cluster sum of squares by cluster:
## [1] 0.46337219 0.00000000 0.42886952 0.04953112 0.67844329 0.63407785 1.21975854
##  (between_SS / total_SS =  85.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

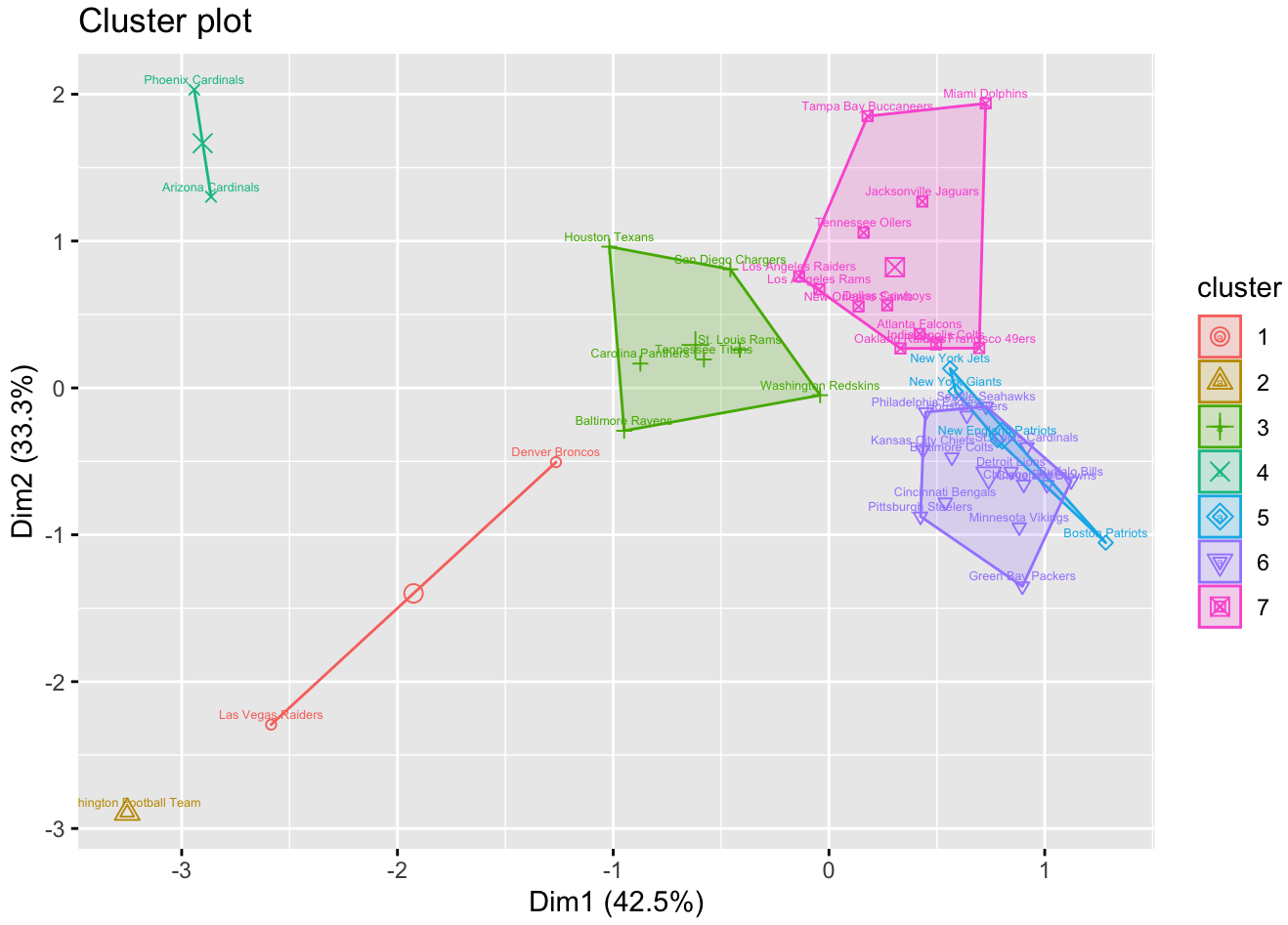
```
head(ss_weather_mean)
```

	weather_temperature_mean <dbl>	weather_wind_mph_mean <dbl>	weather_humidity_
Arizona Cardinals	0.60001457	-0.638441469	-1.228
Atlanta Falcons	0.15949277	0.006317011	0.057
Baltimore Colts	-0.20338135	0.059872836	0.072
Baltimore Ravens	-0.07765899	-0.661934488	-0.191
Boston Patriots	-0.52511649	1.118079410	-0.393
Buffalo Bills	-0.29051653	0.323344501	0.165

6 rows

fviz k=7

```
ss_weather_fviz<-ss_weather_mean[,-4]

fviz_cluster(ss_weather_k7, data=ss_weather_fviz, labelsize = 5)
```



Best outcome was K=7

```
ss_weather_mean_k7<-ss_weather_mean
ss_weather_mean_k7$clusters<-as.factor(ss_weather_k7$cluster)
ss_weather_mean_k7
```

	weather_temperature_mean <dbl>	weather_wind_mph_mean <dbl>	weather_
Arizona Cardinals	0.600014571	-0.638441469	
Atlanta Falcons	0.159492769	0.006317011	
Baltimore Colts	-0.203381348	0.059872836	
Baltimore Ravens	-0.077658992	-0.661934488	
Boston Patriots	-0.525116492	1.118079410	
Buffalo Bills	-0.290516531	0.323344501	
Carolina Panthers	0.113814323	-0.598605133	
Chicago Bears	-0.289935636	0.170519352	
Cincinnati Bengals	-0.325306529	-0.074975959	
Cleveland Browns	-0.297769370	0.269548986	

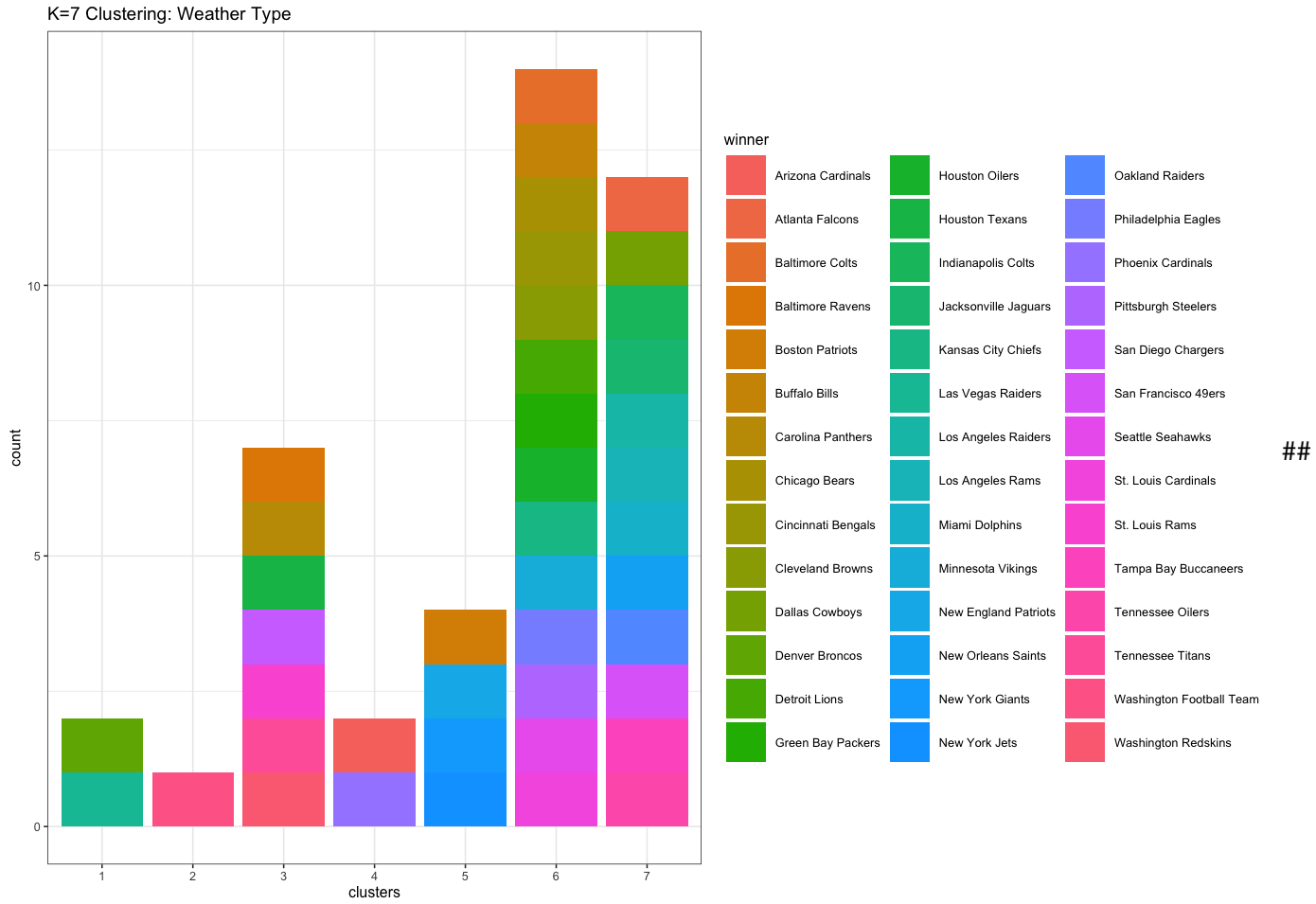
1-10 of 42 rows | 1-4 of 5 columns Previous **1** 2 3 4 5 Next

```
#make gg plot
ss_weather_mean_k7$winner<-rownames(ss_weather_mean_k7)
rownames(ss_weather_mean_k7)<-NULL
ss_weather_mean_k7
```

weather_temperature_mean <dbl>	weather_wind_mph_mean <dbl>	weather_humidity_mean <dbl>	clusters <fct>
0.600014571	-0.638441469	-1.22809507	4
0.159492769	0.006317011	0.05760822	7
-0.203381348	0.059872836	0.07271844	6
-0.077658992	-0.661934488	-0.19120064	3
-0.525116492	1.118079410	-0.39328127	5
-0.290516531	0.323344501	0.16509015	6
0.113814323	-0.598605133	-0.18614384	3
-0.289935636	0.170519352	0.16410359	6
-0.325306529	-0.074975959	0.15398598	6
-0.297769370	0.269548986	0.14471667	6

1-10 of 42 rows | 1-4 of 5 columns Previous **1** 2 3 4 5 Next

```
ggplot(data=ss_weather_mean_k7, aes(x=clusters, fill=winner)) + geom_bar()+labs(title="K=7 Clustering: Weather Type")
```



##

HAC

```
ss_weather_mean<-ss_weather_mean[,-4]
ss_weather_mean
```

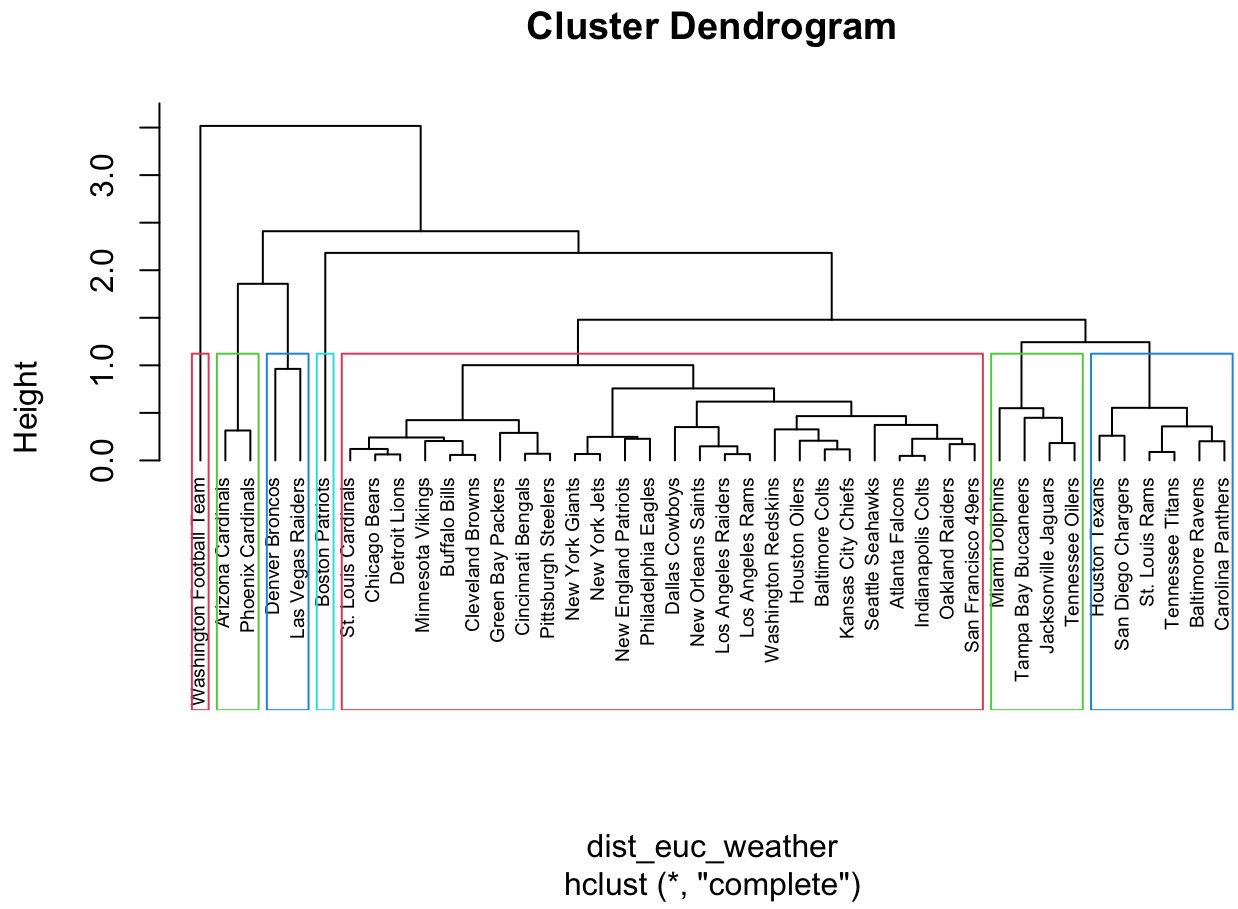
	weather_temperature_mean <dbl>	weather_wind_mph_mean <dbl>	weather_
Arizona Cardinals	0.600014571	-0.638441469	
Atlanta Falcons	0.159492769	0.006317011	
Baltimore Colts	-0.203381348	0.059872836	
Baltimore Ravens	-0.077658992	-0.661934488	
Boston Patriots	-0.525116492	1.118079410	
Buffalo Bills	-0.290516531	0.323344501	
Carolina Panthers	0.113814323	-0.598605133	

	weather_temperature_mean <dbl>	weather_wind_mph_mean <dbl>	weather_
Chicago Bears	-0.289935636	0.170519352	
Cincinnati Bengals	-0.325306529	-0.074975959	
Cleveland Browns	-0.297769370	0.269548986	

1-10 of 42 rows Previous **1** 2 3 4 5 Next

Find euclidean distance and make dendrogram

```
dist_euc_weather<-dist(ss_weather_mean, method = "euclidean")
#dendrogram
hac_euc_weather<-hclust(dist_euc_weather, method="complete")
plot(hac_euc_weather, cex=0.6, hang=-1)
rect.hclust(hac_euc_weather, k=7, border=2:5)
```



Cluster Dendrogram for spread, combined score, over-over under line

```
#remove cluster column
ss_over_cluster_mean_scaled_hac<-ss_over_cluster_mean_scaled[,-4]
#find euclidean distance
dist_euc_over<-dist(ss_over_cluster_mean_scaled_hac, method = "euclidean")
#dendrogram
hac_euc_over<-hclust(dist_euc_over, method="complete")
plot(hac_euc_over, cex=0.6, hang=-1)
rect.hclust(hac_euc_over, k=6, border=2:5)
```