Objective

The overall objective of my project was to find data from the Internet about the NBA, and do analysis on the information I found using machine learning. To do this, I scraped data off of websites like nba.com and ESPN, put this data into csv files to organize it, inserted this data into a sql database, queried specific information from this database and ran k-means clustering on a team's best player's points per game total and their teams overall offensive rating.

Web Scraping

Although we did not explicitly learn how to do this in class, the concept of web scraping was covered and this summer I gained experience with creating automation scripts for regression testing, so I wanted to challenge myself to apply my skills from that experience to class. To scrape the web, I utilized the Playwright library in Python which allows you to do browser tasks in your code, and for identifying web page elements I used XPath, which is an expression language that allows you to identify web elements based on conditions you give it. (Think of XPath as regex for html elements)

I created three scraping files to get information on Team Offensive Ratings, the type of shots teams' take, and players' individual statistics. For the first scraper script playerPPGScraper.py, I navigated to the website https://www.nba.com/stats/players/traditional?Season=2023-24 and used python lists to store a players name, points, team, field goal percentage, and games played, where each element in the list's represents a player. After this, I used python file processing to create put this information into a csv file named playerStatistics.csv located in the CSV folder. For the ShotDietScraper.py script, I elected to use a dictionary to store the information before putting it into a CSV due to the nature of the information being more complex and confusing to handle in a regular list. Once I processed this information into the dictionary, I put it into the CSV file shotDietByTeam.csv. For teamOFFRTG.py, I used the same strategy as the player statistic scraper due to the information being quite simple and used python lists to store each team's name and offensive rating in the CSV teamOffensiveRating.csv

Database Creation

To help create a database that will allow me to query information about each team and its players, I needed two tables. One table that contains team statistics such as offensive rating and the type of shots a team takes, and another table containing players' individual statistics. To connect these databases, we can use the team Name as a foreign key, because each player belongs to one team. To create these databases I used sqlite3 in python.

For the Team Database there was one issue with converting the CSV into a database that I had to solve. I had two CSVs that I wanted to merge into one Table in the db. To solve this problem, I elected to iterate through the shotDietByTeam.csv and create variables for each value, and after this iterate through the teamOffensiveRating.csv file until the team name matches that of the current iteration in shotDietByTeam.csv and only then create a variable for the offensive rating for that team. This could be made more efficient by sorting the csv's by an overlapping metric, such as the lexicographical order of the team name. This would take the algorithm from O(n^2) to O(nlogn). However the size of n is 30 so it does not change much. However if we were working with a much larger dataset this would be helpful. Once I initialized the variables for the given team, I inserted them into the table NBATEAMS and did so for each team.

For the NBAPLAYERS table, I was only pulling information from one CSV file so I just had to iterate through the playerStatistics.csv to insert each row into the table.

Database Queries

For my K-means clustering, I want to analyze the correlation between a team's best player's offensive ability and the team's offensive ability as a whole. To do this, I have to determine which player is the best on each team. I came up with a couple of conditions to determine which player on a team is the best. The player has to have shot over 43% from the field AND averaged the most points on that team. To find this information, I used the query titled “playerQuery” in the QueryAndKmeans.py file to find the best player. I also used “teamQuery” to find the offensive rating of each team. Based on these queries, I created a dict with the team name being the key, and the first value being a tuple of the best player's name along with their points per game, and the second value being the teams offensive rating.

K Means Implementation

Because I had a somewhat obscure data structure storing the two values of points per game and team offensive rating, I opted to implement my own k-means algorithm rather than use a library. I implemented this in the kmeans.py file. I created the function distanceDatapoint(A,B) to calculate the distance from one point to another. To determine when the algorithm was done I created parameters in the while loop that ensured the algorithm was limited to 300 iterations and that the percent change of error, measured in MSE, was greater than 0.1% to continue. Once the algorithm was done I returned the clusters data structure which was a list where the first item was the centroid's location and where the following values were data points that were closest to that centroid, along with the loss in MSE.

K-means visualization

To visualize the results, I used matplotlib.pyplot to graph a scatter plot where x values were the team's best player's points per game and the y values were the team's offensive rating. Each datapoint had a color that was consistent with those in the same cluster. For each cluster, I looped through its elements and created a list containing the x values and the y values where each element represented a given team. I also created a list called graphColors that contained a unique color for each cluster. I created about 7 graphs to represent clusters from 3 all the way up until 9.

K-means optimal centroid determination

To determine the optimal number of centroids for this graph, I used a line graph to show the relationship between loss, measured in MSE, and number of clusters. This was less helpful just because the number of data points was so small, however it is nice to visualize to see if there is a consistent point where the error stops reducing by a significant amount. If I were to go by the book, the optimal number of clusters would be 7 because after 7 the error stops decreasing by a significant amount, however this is not the best determination of the optimal number of centroids due to the lack of data points, and if I had to give my best judgement I would say the optimal number of clusters would be anywhere from 3 to 5.

Project Reflection

One thing with my project that I wish I could have found a work around was the lack of data points that I used. I ran k-means on only 30 data points. However, to the best of my knowledge there is no real workaround to this issue. When trying to come up with solutions to this problem I thought of using data points from past years, however this would not have been effective due to the nature of the development of NBA offenses. Since 1999, the average offensive rating has gone from 102.2 all the way up to 115.3 in

2024. However, each team’s best player’s points per game has not gone up to match this. Along with this, the style of play has also changed a lot. To compare these data points together would not be an honest representation of how a team’s best player’s offensive ability and a team’s offensive rating are correlated.

One conclusion I can make from the visualization of the k-means clustering is that having a player that averages at least 26 points per game is going to raise the floor of your team. If you look at the graph, the bottom right corner of the graph is noticeably empty while all other corners of the graph have a decent percentage of the data points. From this, we can conclude that having a player that averages at least 26 points per game gives your team’s offense a higher floor, meaning that your team will have an at least average offense. However, having a player of this caliber does not guarantee that you will have an above average or elite offense.

What would I do differently?

Despite me making a database that had more than enough information for me to run my k-means clustering and fulfill my original question, It could have been interesting to maybe scrape more data just to have such as teams defensive stats and more individual player’s stats, as I could have played around a bit more with my SQL queries to extract certain information and maybe come up with other data that I wanted to run analysis on.