Rahul Suresh Ubale

Report

Introduction:

In this project, we will be working with Spark programming to analyze parking violation data in New York City. The data is publicly available and contains details of every parking ticket issued in the city. Our aim is to answer several questions related to this data, such as when are tickets most likely to be issued, what are the most common types of cars to be ticketed, where are tickets most commonly issued, and which color of the vehicle is most likely to get a ticket.We also analysis NBA Shot Logs. Spark is a system that implements the Resilient Distributed Dataset (RDD) to maximize memory space in a cluster, and it enables most operations to be done in memory.

Design and Experiments:

NY Parking Violations:

First I started with analysis of data, understanding the column, and its contents.

Part1: When are tickets most likely to be issued?

I tried to solve this question with two different ways,

First , The analysis aims to determine when tickets are most likely to be issued based on the Violation Time data. The data is sourced from the Parking Violations Issued - Fiscal Year 2023 dataset, which is read into a PySpark DataFrame.

The PySpark SQL function count() is used to count the number of violations for each Violation Time. The resulting DataFrame is then sorted in descending order of the number of violations using the orderBy() function. The top 20 rows of the sorted DataFrame are displayed using the show() function.

Based on the results, it can be concluded that tickets are most likely to be issued at 8:36 AM (0836A). This time has the highest number of violations with 22,958 tickets issued. The next four most likely times for ticket issuance are all within a few minutes of 8:36 AM, with the second highest number of violations occurring at 8:39 AM (21,772 tickets issued).

Second, the analysis aims to determine when tickets are most likely to be issued based on the Issue Date data. The Parking Violations Issued - Fiscal Year 2023 dataset is read into a PySpark DataFrame, and the Issue Date column is converted to a date type using the to_date() function.

The data is then grouped by Issue Date, and the count() function is used to determine the number of tickets issued on each date. The resulting DataFrame is sorted in descending order of the number of tickets issued using the orderBy() function. The top 10 rows of the sorted DataFrame are displayed using the show() function.

Based on the results, it can be concluded that August 4, 2022 (Aug Month) is the day when tickets are most likely to be issued, with a total of 66,726 tickets issued.

Part2: What are the most common years and types of cars to be ticketed? (15 pts)

My code reads in the parking violation data from a CSV file, groups the data by vehicle year and make, counts the number of tickets for each group, filters out rows where vehicle year is 0(as it it null value), sorts the results in descending order by the number of tickets, and shows the top 10 rows of the sorted DataFrame.

The analysis shows that Toyota cars in the year 2021 received the most number of parking tickets, followed by Honda cars in the year 2019, and Honda cars again in the year 2021 in the third place. You can find whole ranking in ans folder.

Part3: Where are tickets most commonly issued? (15 pts)

I tried to solve this question with two different ways,

From violation county, the first ,code groups the data by violation county and counts the number of violations in each county. The results are then sorted in descending order and the top 10 counties are displayed using the show() method. The output shows that the most ticketed county is NY (which likely refers to Manhattan), followed by QN (Queens) and BK (Brooklyn). The total number of tickets issued in NY is 2,450,153, while the numbers for QN and BK are 1,858,441 and 1,732,079, respectively.

From violation precinct, the code groups the data by violation precinct and counts the number of violations in each precinct. The results are then sorted in descending order and the top 10 precincts are displayed using the show() method. Interestingly, the precinct numbered 0 has the highest number of tickets issued, which is 5,349,526. This could indicate that either there is some error in recording the precinct number or this number is being used as a default value for tickets with unknown precincts. The precincts numbered 19 and 13 have the second and third highest number of tickets issued, respectively, with 282,466 and 254,057 tickets each.

Part:4 Which color of the vehicle is most likely to get a ticket? (15 pts)

The analysis of the data suggests that the color of the vehicle most likely to get a ticket is GY, or gray. This is determined by grouping the data by vehicle color and counting the number of tickets issued for each color. The resulting data shows that 2,275,457 tickets were issued for gray vehicles, which is the highest number of tickets for any color.

Part:5 Based on a K-Means algorithm, please try to answer the following question:Given a Black vehicle parking illegally at 34510, 10030, 34050 (street codes). What is the probability that it will get an ticket? (very rough prediction). (20 pts)

To solve the problem, the PySpark framework is used, which allows processing a large dataset distributed across a cluster. The solution starts by reading the CSV file into a DataFrame and filtering the data to select only Black vehicles. Then, the selected Black vehicles' distance from the street codes 34510, 10030, 34050 is calculated and added as a new column to the DataFrame.

After that, the necessary columns are selected, and the Violation Code column is converted from a string to a double data type. Then, the VectorAssembler class is used to assemble the selected columns as features for K-Means clustering.

The KMeans algorithm is used to create a model, which is then used to predict the cluster for a new point created using the street codes 34510, 10030, 34050. The predicted cluster is then used to calculate the probability of getting a ticket for a Black vehicle at those street codes.

The probability of getting a ticket for a Black vehicle parking illegally at 34510, 10030, 34050 is found to be 0.445.

NBA Shot Logs:

First I started with analysis of data, understanding the column, and its contents.

Part1: For each pair of the players (A, B), we define the fear sore of A when facing B is the hit rate, such that B is closet defender when A is shooting. Based on the fear sore, for each player, please find out who is his ”most unwanted defender”. (10 pts)

The task in this code is to find the "most unwanted defender" for each player based on the hit rate, which is the fear score of a player when facing a defender. The data is read from a CSV file and the necessary columns are selected. The number of shots taken and made by each player with each defender is calculated and the hit rate is determined for each player-defender pair. A window is defined to partition the data by player name and order by hit rate in ascending order. Then, for each player, the defender with the lowest hit rate is selected as the most unwanted defender.

The code uses PySpark SQL functions to perform these operations. First, the data is grouped by player name and closest defender, and the count of shots taken and made by each player-defender pair is calculated. Then, the hit rate is calculated by dividing the shots made by shots taken. A window is defined to order the data by hit rate for each player in ascending order. The function row_number() is used to number the rows within each partition, and the row with number 1 is selected for each partition, which corresponds to the defender with the lowest hit rate. Finally, the most unwanted defender for each player is determined by grouping the data by player name and selecting the closest defender with the lowest hit rate.

You can see complete ans in folder ans.

Part2: For each player, we define the comfortable zone of shooting is a matrix of, {SHOT DIST, CLOSE DEF DIST, SHOT CLOCK} Please develop a Spark-based algorithm to classify each player’s records into 4 comfortable zones. Considering the hit rate, which zone is the best for James Harden, Chris Paul, Stephen Curry, and Lebron James. (10 pts)

My code snippet defines a Spark-based algorithm to classify each player's records into four comfortable zones based on the hit rate, and then determines the best comfortable zone for four basketball players, James Harden, Chris Paul, Stephen Curry, and Lebron James, based on their hit rate.

The code first loads the dataset "shot_logs.csv" into a Spark DataFrame using the CSV format and header option. It then calculates the hit rate for each player by grouping the records by player name and aggregating the average of the binary indicator variable based on whether the shot was made or missed.

The algorithm defines four comfortable zones based on the hit rate and assigns each record to a zone based on its hit rate using when-otherwise statements. The four comfortable zones are "Very High," "High," "Medium," and "Low," where a very high hit rate corresponds to a hit rate between 0.8 and 1.0, a high hit rate between 0.6 and 0.8, a medium hit rate between 0.4 and 0.6, and a low hit rate between 0.0 and 0.4.

To determine the best comfortable zone for each player, the algorithm joins the player_zones DataFrame with itself on player name and max_hit_rate to filter the rows with the maximum hit rate for each player. The resulting DataFrame contains three columns: player_name, max_hit_rate, and Comfortable Zone.

The algorithm then selects the rows corresponding to the four basketball players of interest using a list of player names and displays the player name, max_hit_rate, and Comfortable Zone for each player using a for loop.

Based on the output, the best comfortable zone for each of the four basketball players is "Medium," which is consistent with the hit rate being the highest in that zone among the four defined zones. Now see best comfortable zone of each player see NBAShotLogsPart2allPlayer.txt file in ans folder.

Conclusion:

In conclusion, Spark is a powerful and flexible tool for performing large-scale data processing and analysis. It provides a variety of functions and libraries that can be used to manipulate and analyze data, making it ideal for big data applications.