**ENGR 516 - Assignment 2**

by [sdiware@iu.edu](mailto:sdiware@iu.edu)

**Spark Installation:**

**Below are the installation steps for Spark:**

1. I have created a docker file called docker-compose.yaml.
2. I added below content in the docker file.

A screen shot of a computer

Description automatically generated

1. Using this docker image I have installed PySpark in Jupyter.
2. I started the docker container using sudo docker-compose up.

A black background with white and blue lines

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

**PART 1 - NY Parking Violations**

**Spark Session:**

A screenshot of a computer program

Description automatically generated

**Data File:** **Parking\_Violations\_Issued\_-\_Fiscal\_Year\_2023\_20231115.csv**

A close-up of a web page

Description automatically generated

**Loading Data:**

A screenshot of a computer

Description automatically generated

**Dataset Schema:**

A screen shot of a computer screen

Description automatically generated

**Checking Null Values in Dataset:**

A screenshot of a computer

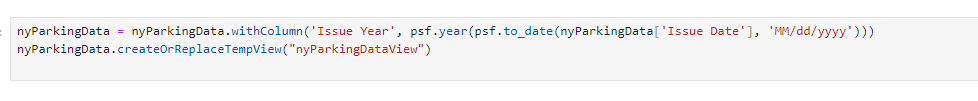
Description automatically generated

**Preprocessing and Handling Null Values:**

A screenshot of a computer program

Description automatically generated

**Creating View for NY Parking Dataset:**



**Question 1: When are tickets most likely to be issued?**

* **Query:**

SELECT `Violation Time`, COUNT(\*) AS Ticket\_Frequency FROM nyParkingDataView GROUP BY `Violation Time` ORDER BY Ticket\_Frequency DESC

* **Approach:**

I have used Violation Time to find out when were the most tickets issued.

* **Answer:**

At 8:36 AM most tickets were issued, with exact count of 35300.

* **Screenshot:**

**A screenshot of a computer

Description automatically generated**

**Question 2: What are the most common years and types of cars to be ticketed?**

* **Query:**

SELECT `Vehicle Body Type` as `Type of Car`,`Issue Year`, COUNT(\*) AS `Violation\_Count` FROM nyParkingDataView WHERE (`Vehicle Year` > 0) GROUP BY `Vehicle Body Type`, `Issue Year` ORDER BY `Violation\_Count` DESC

* **Approach:**

There was no column with issue year so I created a Issue Year column with the help of Issue date column and I have used Issue year column to find out common years to be ticketed and Vehicle Body Type column to find out type of cars to be ticketed.

* **Answer:**

Most Common Year is 2023 And Type of Car is SUBN which is ticketed.

* **Screenshot:**

Issue Year column creation:



A screenshot of a computer

Description automatically generated

**Question 3: Where are tickets most commonly issued?**

* **Query:**

SELECT `Violation Location`, COUNT(\*) AS No\_of\_tickets FROM nyParkingDataView GROUP BY `Violation Location` ORDER BY count(\*) DESC

* **Approach:**

I have used Violation Location to find out the location where the tickets are most commonly issued.

* **Answer:**

Location with most tickets issued is 19.

* **Screenshot:**

A screenshot of a computer screen

Description automatically generated

**Question 4: Which color of the vehicle is most likely to get a ticket?**

* **Query:**

SELECT `Vehicle Color`, COUNT(\*) as Ticket\_Count FROM nyParkingDataView GROUP BY `Vehicle Color` ORDER BY COUNT(\*) DESC

* **Approach:**

I have used Vehicle colour column to find out most likely which coloured vehicle is getting the tickets.

* **Answer:**

Vehicle Colour is WH and ticket count for this Vehicle Color: 2227522

* **Screenshot:**

A screenshot of a computer

Description automatically generated

**Question 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 a ticket?**

* **Approach:**

1. Below are the possible black colours for vehicles and street codes for illegal parking:

Input for the K-means:

**A screenshot of a computer

Description automatically generated**

1. I have only kept the columns which we need:



1. I have added a new column named "Street\_codes," which is a vectorized representation of the columns "Street Code1," "Street Code2," and "Street Code3" using the Spark VectorAssembler.



1. This function initializes and fits a K-Means clustering model with k=4 clusters on a Spark DataFrame inputData that includes a "Street\_codes" column, and it returns the transformed DataFrame along with the cluster centers as a NumPy array.

A screen shot of a computer code

Description automatically generated

1. This function calculates the probability of having black-colored vehicles in each cluster based on a Spark DataFrame inputData containing a 'prediction' column and a 'Vehicle Color' column, using the specified black colours in the blackColor list.

A screen shot of a computer code

Description automatically generated

1. This function finds the cluster with the closest center to a specified set of streets by calculating the Euclidean distance between each cluster center and the streets, then returning the index of the closest cluster.

A screenshot of a computer code

Description automatically generated

1. Printing the cluster ID and Probability of that specific cluster using below function printClusterProb and calculatePrintProbability is the main function to call all the functions.

A screenshot of a computer program

Description automatically generated

1. Below is the function call and out of the Kmeans.

A screenshot of a computer

Description automatically generated

* Answer: For K value 4 I have built the model and probability is **0.14615634143629297.**

**Output:**

A computer screen shot of a computer code

Description automatically generated

**Part 2: NBA Shot Logs:**

Data File: shot\_logs.csv

**Spark Session:**

**A screenshot of a computer program

Description automatically generated**

**Loading the Data:**

**A screenshot of a computer program

Description automatically generated**

**Dataset Schema:**

**A screenshot of a computer program

Description automatically generated**

**Question 1:**

**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”.**

**Approach:**

1. In below, two conditions, madeCond and missedCond, are defined to represent whether a basketball shot was made or missed. The NBA dataset is then grouped by player and closest defender, and a new DataFrame, unwantedDf, is created to aggregate the counts of made and missed shots for each player and defender pair, providing insights into their scoring performance.

A screenshot of a computer program

Description automatically generated

1. In below a new column "HitRate" is added to the DataFrame unwantedDf, representing the ratio of made shots to the total shots (made plus missed), providing a measure of scoring efficiency for each player and defender pair.

A screenshot of a computer

Description automatically generated

1. In first step it filters out rows in the DataFrame unwantedDf where the "HitRate" column is not null, removes duplicate rows based on "Player ID" and "HitRate," calculates the minimum "HitRate" for each player, and then joins this information back to the original DataFrame. After joining with the NBA dataset (nbaData), it creates a new DataFrame unwantedDf containing information about the most unwanted defender for each player. Finally, it drops duplicate records based on "Player ID" and "Defender ID" and selects and displays the columns "Player Name" and "Most Unwanted Defender" for the first 10 rows.

A screenshot of a computer

Description automatically generated

1. **Final Output:**

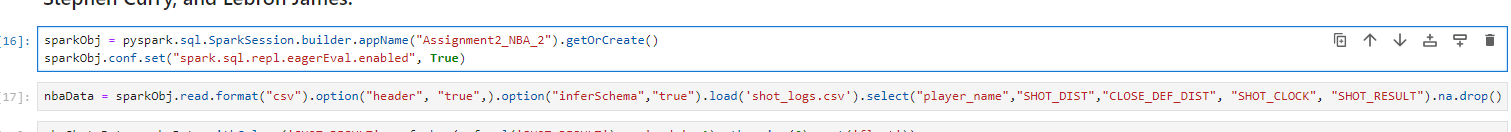
* From below final output we can see top 10 players with their most unwanted defenders.
* Let’s say if Kevin Garnett is the shooter, the most the unwanted defender is the Exum, Dante.

**A screenshot of a computer

Description automatically generated**

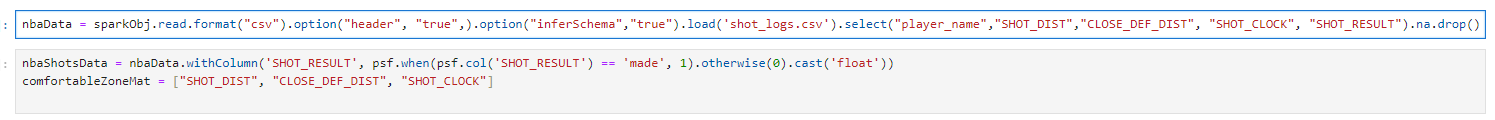
**Question 2: For each player, we define the comfortable zone of shooting is a matrix of, {SHOT DIST, CLOSE DEF DIST, SHOT CLOCK} Please develop a Spark-based algorithm to classify each player’s records into 4 comfortable zones. Considering the hit rate, which zone is the best for James Harden, Chris Paul, Stephen Curry, and Lebron James.**

**Spark Sesson Creation and Loading the data:**

****

**Approach:**

1. It selects specific columns ("player\_name," "SHOT\_DIST," "CLOSE\_DEF\_DIST," "SHOT\_CLOCK," and "SHOT\_RESULT") and drops rows with missing values. The "SHOT\_RESULT" column is then transformed into a binary float column (1 for 'made' shots, 0 for 'missed' shots). The list comfortableZoneMat is created, specifying columns to be used as features for later analysis, such as shot distance, defender distance, and shot clock time.

****

1. Below loop iterates over the columns specified in comfortableZoneMat for the PySpark DataFrame nbaShotsData and casts each column to a float type.

A white rectangle with black text

Description automatically generated

1. It uses VectorAssembler to combine the specified features (comfortableZoneMat) into a single vector column named "shooting\_zone" for each player in the nbaShotsData DataFrame. It initializes a K-Means clustering model with k=4 clusters and fits the model to the transformed data, creating kmeansFitData. It filters the player data for specific players ('james harden', 'chris paul', 'stephen curry', 'lebron james'). It uses the fitted K-Means model to predict the cluster assignments for the selected players, resulting in a DataFrame pred containing player names, predicted cluster labels, and shot results.

A white background with text

Description automatically generated

1. Below code is calculating the average shot result for each player and cluster prediction, ordering the results. It then identifies the maximum average shot result for each player, performing a join to pinpoint the rows where the average is the highest. The final DataFrame, named bestZone, retains all columns from the original DataFrame, and the results are displayed.

A screenshot of a computer

Description automatically generated

1. **Final Output:**

* Zone-1 corresponds to a prediction value of 0, Zone-2 to 1, and Zone-3 to 2 and Zone 4 to 3 in the 'prediction' column.
* To determine each player's comfort zone, we grouped the data by player and zone, calculating the average score for each group.
* As seen in the table Lebron, James, Stephen has best Zone 1 and Chris has best Zone 4.

A white paper with black text

Description automatically generated