Report

Note: *All of the questions had some open interpretations so I explained my assumptions and logic behind each question.*
I ran my scripts on Jetstream2's ubuntu instance.
To run the python scripts submitted with this report, go to the spark directory:
Command: bin/spark-submit final1.1.py

Question 1

I pushed the data into hdfs for running spark jobs and accessing files using hdfs:
`NYCdata = "hdfs://127.0.0.1:9000/NYCdata/input/NYdata.csv"`

Part 1:

I took the columns which I needed and then dropped all null values in it, so that my database does not have null entires while doing analysis
df = df.select(df[**'Summons Number'**],df[**'Issue Date'**],df[**'Vehicle Year'**],df[**'Vehicle Body Type'**],df[**'Violation Location'**],df[**'Vehicle Color'**])
df = df.na.drop()

This approach was selected over drop null values from all the dataset because doing all resulted all the data getting lost.

1) **When are tickets most likely to be issued? (15 pts)**
   Query:
   Q1 = spark.sql(**"SELECT COUNT(\`Summons Number\`) AS counts, \`Issue Date\` FROM NYdata GROUP BY \`Issue Date\` ORDER BY counts DESC LIMIT 1"**)

```
+------+----------+
|counts|Issue Date|
+------+----------+
| 39219|10/20/2022|
+------+----------+
```

   I took Issue date to find when are the tickets most likely to be issued. I got that maximum of 39219 tickets were issued on date: 10/20/2000

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

Q2 = spark.sql(**"SELECT count(`Summons Number`) as number_of_tickets,`Vehicle Year`,`Vehicle Body Type` FROM NYdata GROUP BY `Vehicle Body Type`,`Vehicle Year` ORDER BY number_of_tickets DESC LIMIT 20"**)

number_of_tickets	Vehicle Year	Vehicle Body Type
338028	0	SUBN
323238	0	4DSD
119154	2021	SUBN
104511	2020	SUBN
90758	2019	SUBN
80938	2022	SUBN
75124	2018	SUBN
71170	0	VAN
64669	0	DELV
62832	2017	SUBN
49804	2016	SUBN
49133	2017	4DSD
47575	2015	SUBN
46438	2018	4DSD
45696	0	TRAC
44139	2019	4DSD
42903	2020	4DSD
38942	2015	4DSD
38720	2019	VAN
38311	2014	SUBN

For common years and common types of cars to be ticketed, I grouped all vehicle year and vehicle body types of cars that printed top 20(most common) that were ticketed.

3) **Where are tickets most commonly issued? (15 pts)**

Query:

Q3 = spark.sql(**"SELECT count(`Summons Number`) as number_of_violations,`Violation Location` FROM NYdata GROUP BY `Violation Location` ORDER BY number_of_violations DESC LIMIT 20"**)

number_of_violations	Violation Location
146387	13
142695	19
119226	6
111133	114
102619	14
99656	18
86682	9
86344	1
64666	108
63874	20
63639	109
60178	10
58554	115
56399	70
56173	84
55257	17
51316	52
50860	112
48690	103
48437	43

I grouped all records with same violation location to find the most popular location where tickets were issued.

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

Query:

Q4=spark.sql(**"SELECT count(`Summons Number`) as number_of_tickets,`Vehicle Color` FROM NYdata GROUP BY `Vehicle Color` ORDER BY number_of_tickets DESC LIMIT 20"**)

number_of_tickets	Vehicle Color
613756	WH
548497	GY
468435	BK
357137	WHITE
214691	BLACK
193725	BL
157001	GREY
111775	RD
79456	BROWN
78900	BLUE
75586	SILVE
55068	RED
39955	GR
22540	TN
20154	OTHER
18477	BR
15400	BLK
14242	GREEN
12329	GL
11402	YELLO

I grouped all the tickets having the same vehicle color. As you can see that, white is most common car to be ticketed followed by grey.

Part 2:

1) **Kmeans**

I took all black car records into my data frame and filtered it using street code1, street code2, and street code3. Then, I ran kmeans on it

All black cars were stored in the data with different names so I included those variations while filtering:

```
black=['BK','BLACK','BLK','Black','BLBL','BL/','BK/','BLCK','BKBK','BLAK
','BLAC','BKL','BK.','BCK','BLC','B K','BKACK']
```

I tried kmeans with different cluster sizes(2-10) and chose the cluster with the highest silhouette score. High silhouette scores mean more coherent clusters.
Silhouette scores can be calculated and number of clusters:

```
silhouette scores and their respective number of clusters:
 [[0.7004796043389552, 2], [0.658374648456499, 3], [0.7032570634521899, 4], [0.6839188412933475, 5], [0.7005456233977602, 6], [0.6848187712732975, 7], [0.6
3261510404345, 8], [0.6966900004313312, 9]]
```

I select k=4 because of its silhouette score 0.703.

Data point = [34510, 10030, 34050]
I found to which cluster the data point belongs to using transform function.
First table is predicting to which cluster the new data point belongs to, and counting the number of black cars in that cluster:

```
+---------------+-----------------+
|black_car_count|predicted_cluster|
+---------------+-----------------+
|         185882|                2|
+---------------+-----------------+

+----------------+
|total_black_cars|
+----------------+
|         1324430|
+----------------+
```

Second table displayed is total number of black cars in all the clusters

I calculated the probability as
number of black cars in the cluster to which the new data point belongs / total number of black cars in all clusters = 185882/1324430=0.1403
Final probability:

```
+------------------------+-----------------+----------------+-------------------+
|black car count in cluster|predicted cluster|total black cars|        probability|
+------------------------+-----------------+----------------+-------------------+
|                  185882|                2|         1324430|0.14034867829934386|
+------------------------+-----------------+----------------+-------------------+
```

<u>Question 2:</u>

Part 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”. (10 pts)

I grouped each player, and his closest defender together, and found out the missed and total shots for each pair, hit rate will be minimum when when missed/total is the highest.
For each player, I selected the defender for which missed shots/total shots where the highest.
Missed shots = shots missed when Player A was playing and Player B was the closest defender
Total shots = total shots player when Player A was playing and Player B was the closest defender

hit_rate	player	defender
1.0	al jefferson	Hardaway Jr., Tim
1.0	cody zeller	Price, Ronnie
1.0	gary neal	Smart, Marcus
1.0	gerald henderson	Bazemore, Kent
1.0	lance stephenson	Fournier, Evan
1.0	dante exum	Williams, Mo
1.0	jeremy lin	Gobert, Rudy
1.0	kobe bryant	Jefferson, Al
1.0	ed davis	Snell, Tony
1.0	robert sacre	Payton, Elfrid
1.0	leandro barbosa	Neal, Gary
1.0	stephen curry	Matthews, Wesley
1.0	draymond green	Westbrook, Russell
1.0	mike scott	Griffin, Blake
1.0	demarre carroll	Ingles, Joe
1.0	garrett temple	Mbah a Moute, Luc
1.0	paul pierce	Afflalo, Arron
1.0	omer asik	Matthews, Wesley
1.0	tyreke evans	Parker, Tony
1.0	luke babbitt	Jones, Perry

only showing top 20 rows

The output was big so I submitted the csv file with the report.

Part 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. (10 pts)

I used kmeans for this to filter data on columns {SHOT DIST, CLOSE DEF DIST, SHOT CLOCK} and put those clolumns into 4 clusters for 4 comfortable zones. The best comfortable zone for each player is where his hit rate is higher, so for each cluster, I found the shots made/total shots in that cluster for a player, and put the player in the cluster with the highest hit rate. Below are results for the four players, James Harden, Chris Paul, Stephen Curry, and Lebron James

Player 1: James Harden belongs to comfort zone 1

made_shots	player_name	prediction_col	total_shots	player	cluster
153	james harden	1	273	james harden	1

player_name	comfort_zone_cluster	hit_rate
james harden	1	0.5604395604395604

Player 2: Chris Paul belongs to comfort zone 0

made_shots	player_name	prediction_col	total_shots	player	cluster
173	chris paul	0	352	chris paul	0

player_name	comfort_zone_cluster	hit_rate
chris paul	0	0.4914772727272727

Player 3: Stephen Curry belongs to comfort zone 0

made_shots	player_name	prediction_col	total_shots	player	cluster
200	stephen curry	0	463	stephen curry	0

player_name	comfort_zone_cluster	hit_rate
stephen curry	0	0.4319654427645788

Player 4: Lebron James belongs to comfort zone 1

made_shots	player_name	prediction_col	total_shots	player	cluster
166	lebron james	1	251	lebron james	1

player_name	comfort_zone_cluster	hit_rate
lebron james	1	0.6613545816733067