Streaming Data Analytics Assignment

Problem statement:

Assume that you are working as analyst for “Pizzario”, a pizza delivery chain. The group has collected some interesting characteristics of customers who had purchased their pizza earlier. (Refer the attached pizza_customers.csv file for the same). The marketing team is planning a campaign to increase the sales of a newly launched pizza. Before that they want to analyse the segmentation of existing customers so that they can have the clearer picture about the customer categories.

Exercise 1:

To write a Python program that will take the pizzario customers dataset as input and produce the clusters which represents the customer segments present in the dataset.

Reading the customer dataset:

We have used the pandas library to read the input csv file.

```
In [3]: dataset = pd.read_csv('pizza_customers.csv')
        dataset
```

Out[3]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

For the **data exploration** and **data visualization** part, please check the notebook file (SPA_Assignment_EX_1_and_EX_2.ipynb).

Filtering the data:

For visualization convenience, we are going to take Annual Income and Spending score as our data.

```
: X = dataset.iloc[:, [3, 4]].values
  X.shape
```

```
: (200, 2)
```

Now X is two matrix of shape (200,2).

To find out the optimal number of clusters for the given pizzario customer dataset, we have used elbow method.

To decide the best value for K (number of clusters):

we use a method called ELBOW method to find the appropriate number of clusters. The parameter which will be taken into consideration is Sum of squares of distances of every data point from its corresponding cluster centroid which is called WCSS (Within-Cluster Sums of Squares).

Steps involved in ELBOW method are:

1. Perform K means clustering on different values of K ranging from 1 to any upper limit. Here we are taking the upper limit as 10.

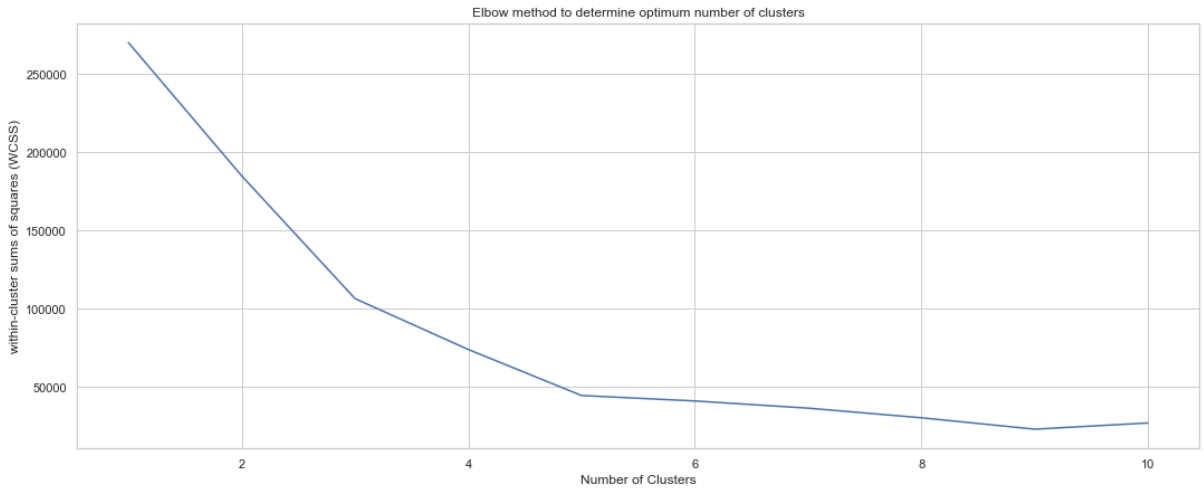
2. For each K, calculate WCSS

3. Plot the value for WCSS with the number of clusters K.

4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters. i.e the point after which WCSS doesn't decrease more rapidly is the appropriate value of K.

Implemented part is present in SPA_Assignment_EX_1_and_EX_2.ipynb file.

Result of Elbow Method:



Observation from Elbow method:

The idea is that we want a small WCSS, but that the WCSS tends to decrease toward 0 as we increase k (the WCSS is 0 when k is equal to the number of data points in the dataset, because then each data point is its own cluster, and there is no error between it and the centre of its cluster). So our goal is to choose a small value of k that still has a low WCSS, and the elbow usually represents where we start to have diminishing returns by increasing k.

Based on these observations we choose 5 as optimum number of clusters.

k Means clustering:

We are ready to implement our Kmeans Clustering steps. Let's proceed:

Step 1: Initialize the centroids randomly from the data points: Centroids is a n x K dimensional matrix, where each column will be a centroid for one cluster.

Step 2:

(a) For each training example compute the Euclidian distance from the centroid and assign the cluster based on the minimal distance

(b) We need to regroup the data points based on the cluster index C and store in the Output dictionary and also compute the mean of separated clusters and assign it as new centroids. Y is a temporary dictionary which stores the solution for one particular iteration.

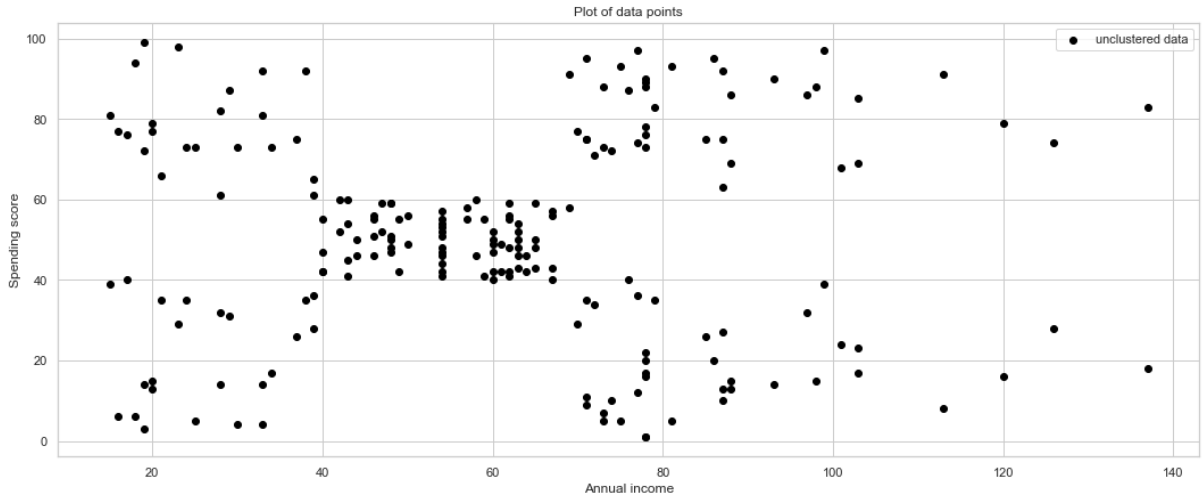
The output of our algorithm should be a dictionary with cluster number as Keys and the data points which belong to that cluster as values. So let's initialize the dictionary.

We find the Euclidian distance from each point to all the centroids and store in a m X K matrix. So every row in Euclidian Distance matrix will have distances of that particular data point from all the centroids. Next, we shall find the minimum distance and store the index of the column in a vector C.

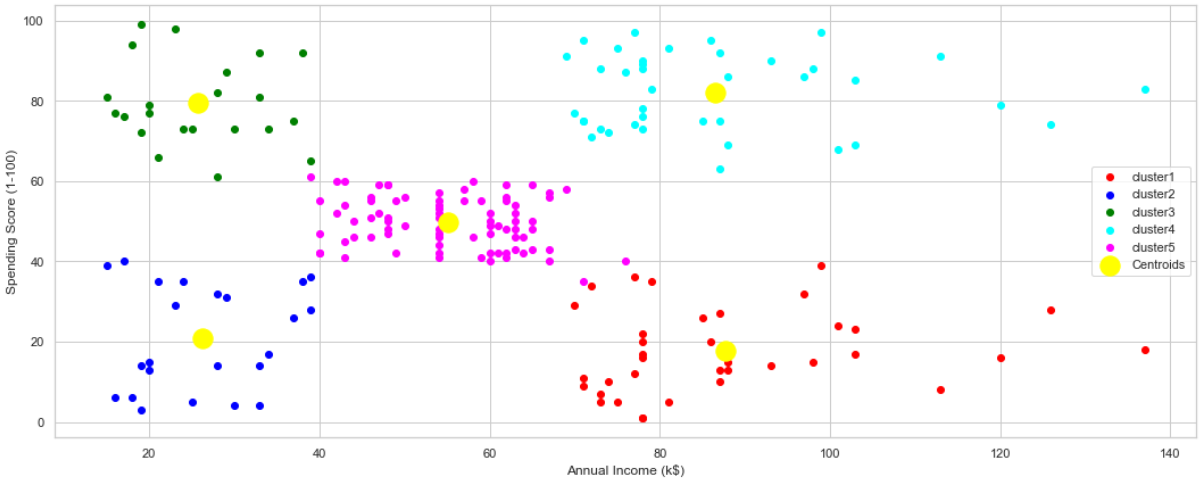
Now we need to repeat step 2 till convergence is achieved. In other words, we loop over n_iter and repeat the step 2 (a) and 2 (b).

Now it's time to visualize the algorithm and notice how the original data is clustered. To start with, let's scatter the original uncluttered data first.

Uncluttered data:



Clustered Data:



Our data has become a clustered data from uncluttered raw data. We can observe that there are five categories of clusters which are:

Inference:

- **Cluster 1:** Customers with **High income and a low spending score**: This cluster people have high annual income but spends very little.

- **Cluster 2:** Customers with **low income and a low spending score**: Maybe these types of customers are too busy saving their money.

- **Cluster 3:** Customers with **low income but a high spending score**: For this type may be the company can recommend products with low price. This cluster contains of by young adults. This cluster has unique behaviour. They have a small annual income but a large spending score. When we're looking at the age, it consists of very young adults.

- **Cluster 4:** Customers with **High income and a high spending score**: This cluster consists of middle-age adults.

- **Cluster 5:** Customers with **medium income and a medium spending score**: This cluster is filled by Old-aged adults.

As an analyst, I'd suggest the Pizzario Marketing team to give extra special treatment to cluster 3 and 4 customers. So, they keep remain loyal to our Pizzario. The spending score of clusters 1 and 2 is very small, then the cluster should be given a discount in order to increase its spending score.

So, the Pizzario company can divide their customers into 5 classes and design different strategies for a different type of customers to increase their sales.

Output the cluster number, centroid used and number of records belonging to that cluster:

```
for k in range(K):
    print(f"\nCluster: {k + 1}")
    print(f"\nCentroid Point: {round(Centroids[0,k],2)},{round(Centroids[1,k],2)}")
    print(f"\nTotal number of records belonging to that cluster: {len(Output[k+1][:,0])}")
    print("\n----------------------------------------------------------------------")
```

```
Cluster: 1

Centroid Point: 55.09,49.71

Total number of records belonging to that cluster: 80

----------------------------------------------------------------------

Cluster: 2

Centroid Point: 87.75,17.58

Total number of records belonging to that cluster: 36

----------------------------------------------------------------------

Cluster: 3

Centroid Point: 26.3,20.91

Total number of records belonging to that cluster: 23

----------------------------------------------------------------------

Cluster: 4

Centroid Point: 86.54,82.13

Total number of records belonging to that cluster: 39

----------------------------------------------------------------------

Cluster: 5

Centroid Point: 25.73,79.36

Total number of records belonging to that cluster: 22

----------------------------------------------------------------------
```

So, we used K-Means clustering to understand customer data. K-Means is a good clustering algorithm. Almost all the clusters have similar density. It is also fast and efficient in terms of computational cost.

Now the marketing team is aware about the spending behaviour of your customers.

As a marketing professional we have to think of marketing strategy to attract these customers to newly launched pizza.

Exercise 2:

To apply appropriate labels to those clusters after careful look at the points belonging to those clusters.

Implemented part is present in SPA_Assignment_EX_1_and_EX_2.ipynb file.

```
labels = ['Miser', 'Careful', 'Spendthrift', 'Target', 'General']

for c in range(K):
    print(f"Cluster: {c + 1}")
    print(f"Length of the cluster: {len(Output[c+1][:,0])}")
    for k in range(len(Output[c+1][:,0])):
        for ind in dataset.index:
            if (Output[c+1][k,0] == dataset["Annual Income (k$)"][ind]) and (Output[c+1][k,1] == dataset["Spending Score (1-100)"
                print(f"Assigning label to the data point {k + 1} as {labels[c]}")
                dataset.at[ind, 'label'] = ''
                dataset["label"][ind] = labels[c]
    print("\n-------------------------------------------------")
```

Logic behind nomenclature of the clusters:

This Clustering Analysis gives us a very clear insight about the different segments of the customers in the Pizzario.

There are clearly Five segments of Customers namely **General (Cluster 1), Spendthrift (Cluster 2), Target (Cluster 3), Miser (Cluster 4), and Careful (Cluster 5)** based on their Annual Income and Spending Score which are reportedly the best factors/attributes to determine the segments of a customer in a Pizzario.

- **For Cluster 1:** Customers **with medium income and a medium spending score**: This set of customers falls under **General category** as these customers earn medium income and spent the money medium.

- **Cluster 2:** Customers with **low income but a high spending score**: This set of customers falls under **Spendthrift category** as these customers are who spends money in an extravagant, irresponsible way even though they earn low income.

- **Cluster 3:** Customers with **High income and a high spending score**: This set of customers falls under **Target category** as these customers earn high and spends high. These customers require special treatment and they remain loyal to pizzario.

- **Cluster 4:** Customers **with High income and a low spending score**: This set of customers falls under **Miser category** as these customers earn high and spend low. These persons are who hoards wealth and spends as little money as possible. They tend to save money instead of spending.

- **Cluster 5:** Customers with **low income and a low spending score**: This set of customers falls under **Careful category** as these customers earn low and spend low. They will be very careful in spending the money.

Show the sample dataset:

```
dataset.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	label
0	1	Male	19	15	39	Careful
1	2	Male	21	15	81	Spendthrift
2	3	Female	20	16	6	Careful
3	4	Female	23	16	77	Spendthrift
4	5	Female	31	17	40	Careful

We are saving this latest dataset as "**pizza_customers_updated.csv**".

Based on this segmentation, think of some marketing offers that can be given to the customers.

KMeans Clustering is a powerful technique in order to achieve a decent customer segmentation. Customer segmentation is a good way to understand the behaviour of different customers and plan a good marketing strategy accordingly.

There isn't much difference between the spending score of women and men, which leads us to think that our behaviour when it comes to purchase of pizza is pretty similar.

Observing the clustering graphic, it can be clearly observed that the ones who spend more money in pizzario are young people. That is to say they are the main target when it comes to marketing, so doing deeper studies about what they are interested in may lead to higher profits.

Although younglings seem to be the ones spending the most, we can't forget there are more people we have to consider, like people who belong to the cluster 1, they are what we would commonly name after "middle class" and it seems to be the biggest cluster.

Promoting discounts on some pizza can be something of interest to those who don't actually spend a lot and they may end up spending more!

Explanation of the logic behind the offers to be made to the various customer categories:

- **For Cluster 1:** Customers **with medium income and a medium spending score**: **General category**

Customers with medium income and a medium spending score: This cluster is filled by Old-aged adults.

Marketing strategy: Pure royalty programs (these segments want them to be valued more than the product/service)

Cashback and gift back programs (like G-pay scratch cards) and rare discounts up to 25%.

Discount: 30%

- **Cluster 2:** Customers with **low income but a high spending score**: **Spendthrift category**

For this type may be the company can recommend products with low price. This cluster contains of by young adults. This cluster has unique behaviour. They have a small annual income but a large spending score. When we're looking at the age, it consists of very young adults.

Marketing Strategy: Mid-range Coupons (to make them returning customer)

Give discount coupons worth 10-15% with a safe cap for next orders

Discount: 25%

- **Cluster 3:** Customers with **High income and a high spending score**: **Target category**

Customers with High income and a high spending score: This cluster consists of middle-aged adults.

Marketing strategy: Royalty programs and low range coupons. Give discount coupons worth 5-10% for next orders

Discount: 10%

- **Cluster 4:** Customers **with High income and a low spending score**: **Miser category**

Customers with High income and a low spending score: This cluster people have high annual income but spends very little.

Marketing strategy: Coupons and Discounts (Targeted Advertisements makes these customers returning)

Discount: 20%

- **Cluster 5:** Customers with **low income and a low spending score**: **Careful category**

Customers with low income and a low spending score: Maybe these types of customers are too busy saving their money.

Marketing strategy: Initial Heavy discounts followed by royalty programs (to make them use the product/service and understand the value and make them returning customers).

Discount: 15%

We are well aware about your existing customer base. We know the customers spending habit, we have decided upon offers to be made, now it's time to test it out for the existing customers.

Let's assume that they frequently visit the mall where your pizza shop is located. For this purpose, you have to have the mechanisms through which their visits to the mall are captured.

Exercise 3:

To write a Python program that will simulate the movement of existing customers around a mall in near real time fashion.

You can assume that customer's cell phones are enabling the transfer of location data to your centralized server where it's stored for further analysis.

With access to the real time movements of customers around mall, you are feeling more powerful and ready to target these customers. Your accumulated knowledge of streaming data processing and analytics can be leveraged for the same purpose. Now think for an architecture to bring it to the reality through a streaming data pipeline.

The python program should generate the customer ID with Timestamp, Latitude and Longitude of visiting the mall. This should be a random generation of customers. This has to be connected to ingestion part of streaming system by using Kafka.

A simple python program to mimic the movement of the customers to the mall.

Exercise 3 is implemented in the **kafka_producer.py** file.

Sample output:

Exercise 4:

Construct a streaming data pipeline integrating the various technologies, tools and Programmes covered in the course that will harvest this real time data of customer's movements and produces the offers that can be sent on the customers mobile devices. You can think of various aspects related to streaming data processing such as:

- Real time streaming data ingestion.
- Data's intermittent storage.
- Data pre-processing – cleaning, transformations etc.
- Data processing – filters, joins, windows etc.
- Business logic for placing the offers.
- Final representation of the outcome.

Exercise 4 is implemented in the **spark_streaming.py** file.

To implement this, we have use spark streams. We have to fed the output of Exercise 3 into Streaming system (Exercise 4) and output will be marketing offers (Discounts). For simplicity, assume Kafka topic for offers. Dump the customer movement data into one Kafka topic. The streaming system will be collecting the data - Real time ingestion. Data pre-processing is done to convert it into the required format. The output will be the discount offers. This is passed to the Kafka consumer.

Setup of Apache Spark Structured Streaming with Kafka using Python (PySpark):

STEP 1: Download the spark structured streaming with kafka dependency JAR files

https://repo1.maven.org/maven2/org/apache/spark/spark-sql-kafka-0-10_2.12/3.0.3/

https://repo1.maven.org/maven2/org/apache/kafka/kafka-clients/1.1.0

Place this under utils folder

STEP 2: Download JAVA 8 version SDK

Make sure you installed JAVA 8 SDK on your system. You can use chocolatey (https://chocolatey.org/) windows package manager for the same. Click Start and type "powershell". Right-click Windows Powershell and choose "Run as Administrator". Paste the following command into Powershell and press enter.

Command to install chocolatey:

Set-ExecutionPolicy Bypass -Scope Process -Force; ` iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

Command to install Java: (Right-click command prompt and choose "Run as Administrator")

choco install jdk8

To check the java version:

java -version

STEP 3: Download Spark

Link: https://spark.apache.org/downloads.html

Select Spark release: 3.0.3

Choose package type: Apache Hadoop 2.7

You will get the respective tgz file. Click on it and the file gets downloaded.

Extract the files and place it under C:\spark\spark

You will also find the same file from here:

Link: https://dlcdn.apache.org/spark/spark-3.0.3/

STEP 4: Download Hadoop

Link: https://github.com/cdarlint/winutils

This contains winutils of hadoop version. Extract the required hadoop version files and place it under C:\spark\hadoop

STEP 5: Download Apache Kafka binaries

We will use Apache Kafka binaries for installing Apache Kafka. Go to Apache Kafka official download page (https://kafka.apache.org/downloads) and download the binaries. Download the scala 2.11.

STEP 6: Create Data folder for Zookeeper and Apache Kafka:

Create “data” folder under kafka directory and create Kafka and Zookeeper directories inside data folder. Create "logs" folder under the upper kafka directory.

STEP 7: Change the default configuration value

Update zookeeper data directory path in “config/zookeeper.Properties” configuration file.

dataDir=C:\kafka\data\zookeeper

Update Apache Kafka log file path in “config/server.properties” configuration file.

log.dirs=C:\kafka\data\kafka

STEP 8: Setup of Environment variables:

HADOOP_HOME = C:\spark\hadoop

JAVA_HOME = C:\Program Files\Java\jdk1.8.0_211

SPARK_HOME = C:\spark\spark

PYSPARK_PYTHON = C:\Python39\python.exe;C:\spark\spark\python\lib\py4j-0.10.9-src.zip;C:\spark\spark\python

Ensure you add the below path to PATH system variable:

C:\Python39\Scripts

C:\Python39\

C:\spark\hadoop\bin

C:\spark\spark\bin

C:\spark\spark\python

Optional: (In case we use Juypter Notebook)

PYSPARK_DRIVER_PYTHON = jupyter

PYSPARK_DRIVER_PYTHON = C:\Users\user\Anaconda3\Scripts\jupyter.exe

PYSPARK_DRIVER_PYTHON_OPTS = notebook

PYSPARK_PYTHON = C:\Users\user\Anaconda3\python.exe

STEP 9: Install all the python packages mentioned in the requirements.txt file.

STEP 10: Start Zookeeper

Now time to start zookeeper from command prompt. Change your directory to bin\windows and execute zookeeper-server-start.bat command with config/zookeeper.Properties configuration file.

To start the zookeeper:

zookeeper-server-start.bat ../../config/zookeeper.properties

And make sure zookeeper started successfully

STEP 11: Start Apache Kafka

Finally time to start Apache Kafka from command prompt. Run kafka-server-start.bat command with kafka config/server.properties configuration file.

To start the Apache Kafka:

kafka-server-start.bat ../../config/server.properties

This will start our Apache Kafka successfully.

STEP 12: To list the topics under the zookeeper

kafka-topics.bat --zookeeper localhost:2181 --list

STEP 13: To create a topic in kafka

kafka-topics.bat --zookeeper localhost:2181 --create --topic malloutput --partitions 1 --replication-factor 1

STEP 14: To describe a topic in kafka

kafka-topics.bat --zookeeper localhost:2181 --describe --topic malloutput

STEP 15: To start the consumer

Before starting the consumer, ensure all the files are being deleted under py_checkpoint folder.

kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic malloutput

(or)

kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic malloutput --from-beginning

STEP 16: Start Apache Spark streaming

Before executing this, create py_checkpoint folder

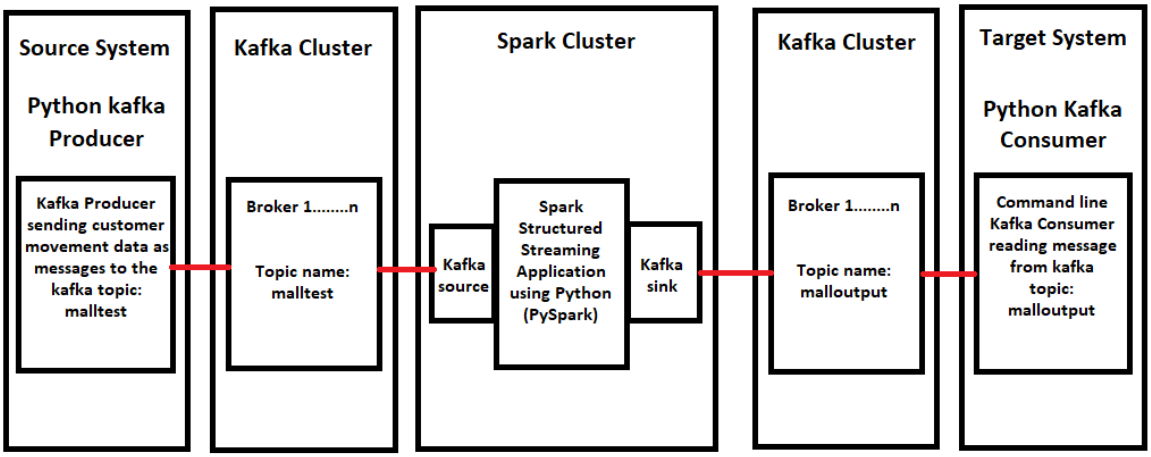
spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.0.3 spark_streaming.py

STEP 17: To start the producer

python kafka_producer.py

The streaming data pipeline architecture:

Spark Structured Streaming Application using Python (PySpark)

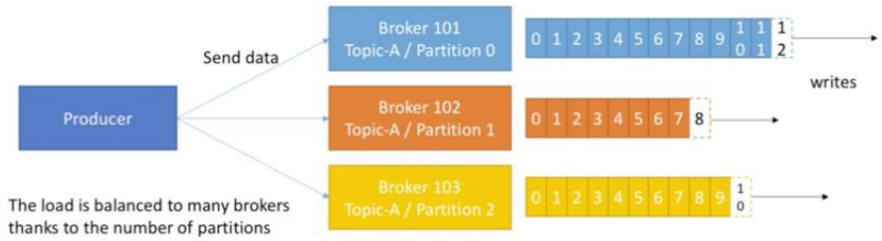


Components used and its purposes:

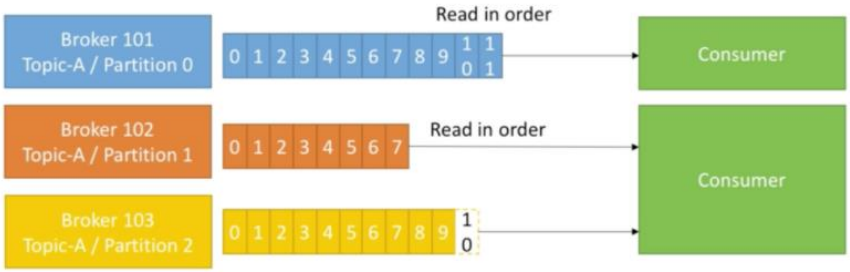
Kafka:

Kafka is a distributed system consisting of servers and clients that communicate via a high-performance TCP network protocol. It can be deployed on bare-metal hardware, virtual machines, and containers in on-premise as well as cloud environments.

Kafka Producer: Producers write data to topics (which is made of partitions). Producers automatically know to which broker and partition to write to. In case of broker failures, Producers will automatically recover.

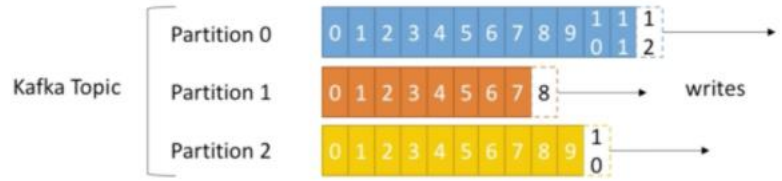


Kafka Consumer: Consumers read the data from a topic (identified by name). Consumers know which broker to read from. In case of broker failures, consumer know how to recover. Data is read in order within each partitions.



Kafka topic: Topic is a particular stream of data. Similar to a table in a database (without all the constraints). You can have as many topics as you want. A topic is identified by its name. They are split

in partitions. Each partition is order and each message within a partition gets an incremental id, called offset.



Brokers: A Kafka Cluster is composed of multiple brokers (servers). Each broker is identified with its ID (integer). Each broker contains certain topic partitions. After connecting to any broker (called a bootstrap broker), you will be connected to the entire cluster.

Zookeeper: This manages brokers (keeps a list of them). Zookeeper helps in performing leader election for partitions. It sends notification to Kafka in case of changes (e.g. new topic, broker dies, broker comes up, delete topics, etc..) Kafka can't work without Zookeeper. It by design operates with an odd number of servers (3,5,7). Zookeeper has a leader (handles writes) the rest of the servers are followers (handle reads).

Spark Cluster: Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in your main program (called the driver program). Once connected, Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for your application.

Data flows:

Basically, the streaming application is implemented via Apache spark structured streaming with Kafka

using Python.

Explanation for the data flow:

There will be many sources who will be generating or pushing data to the streaming application. We have taken Python Kafka Producer which is responsible for sending the customer movement data. Kafka topic we use here is "malltest". In between sources and streaming application, we will be having message broker or messaging layer. That is called as Kafka cluster. This is used to manage the messages send from the Kafka Producer. This also ensures that messages are not lost.

In spark cluster, we are going to develop a streaming application using pyspark. For a streaming application,we will have a source and a sink. Source is where we receive messages from. Sink is responsible to push the desired data to target or destination who is going to consume the data. Here, we use Kafka as a source and sink.

Whatever the computation we perform in spark streaming, that we are going to push it into the Kafka topic called "malloutput".We are using a command line Kafka consumer to read the resultant messages from the kaka topic called "malloutput".

Business logic:

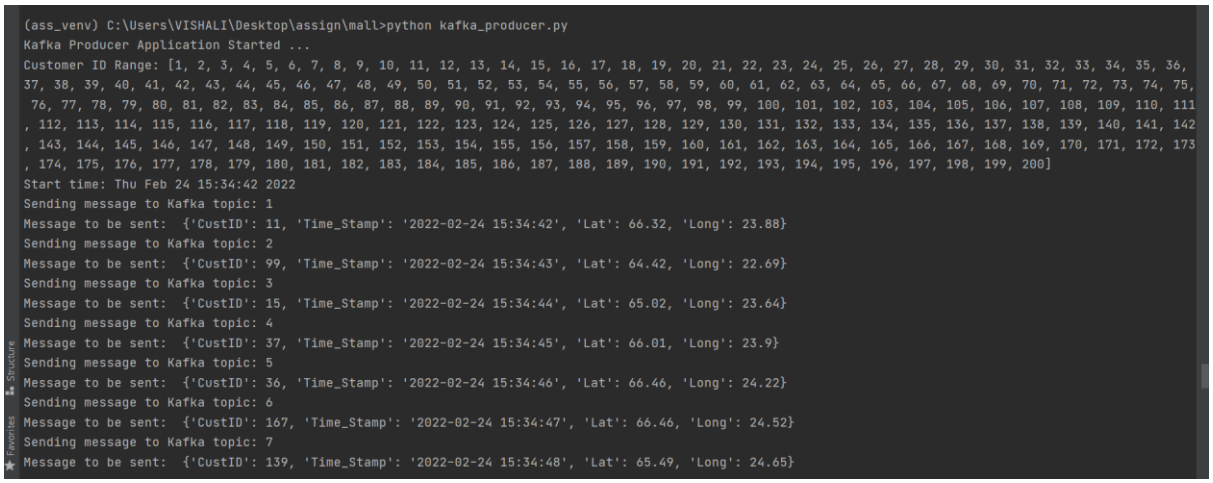
In the "kafka_producer.py" script, we are using Kafka producer which is generating the customer movement data which contains Customer ID, Time stamp, Latitude and Longitude. Here, we are using Kafka topic called "malltest" where we dump all the messages which are generated. The message is in the form of JSON format.

In the “spark_streaming.py” script, we are developing the streaming application. We need to download two jar files which are used as a configuration file in setting up the spark session object. To remove other type of loss, we set the setloglevel as “ERROR”. Next, you have to read streams from Kafka producer topic. We use spark readStream for it. This will return the streaming dataframe. We usually receive it in key value pair. We have to convert it into the actual format we require so that we can do analysis and computation on it. We are converting the value into string type and also we are interested in the timestamp of that message. Then, we will define the schema of the message. We will use from json functionality which is from spark to convert the value into the schema defined earlier. Then, select all the columns by select *. We are going to find the distance between the customer and mall using latitude and longitude position. We are assuming the mall longitude and latitude as 23.34 and 65.42. After the calculation of the distance, we are selecting only the customers who are within the threshold distance (i.e. who are near to the mall region). The threshold distance we are choosing as 100 meters. Then we are reading the pizza customer updated csv file. This file contains the label to which each customer belongs to. We are joining this dataframe with the customer movement dataframe. Then we are allocating the discount offers according to the labels of each customer. We are also generating the coupon code for each offer. Finally, we are selecting the customer id, distance, discount and coupon code column and passing it to the kafka sink using writeStream. Kafka topic used in consumer side is “malloutput”.

In the command line Kafka consumer interface, you will see the resultant output which contains customer ID, distance from the mall, discount offers and coupon code.

Sample output:

Execute “Kafka_producer.py”:



Execute “spark_streaming.py”:

```
Printing Schema of streaming dataset received from kafka producer:

root
 |-- CustID: integer (nullable = true)
 |-- Time_Stamp: string (nullable = true)
 |-- Lat: float (nullable = true)
 |-- Long: float (nullable = true)
 |-- timestamp: timestamp (nullable = true)


Printing Schema of customer_data_df:

root
 |-- _c0: string (nullable = true)
 |-- CustomerID: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: string (nullable = true)
 |-- Annual Income (k$): string (nullable = true)
 |-- Spending Score (1-100): string (nullable = true)
 |-- label: string (nullable = true)
```

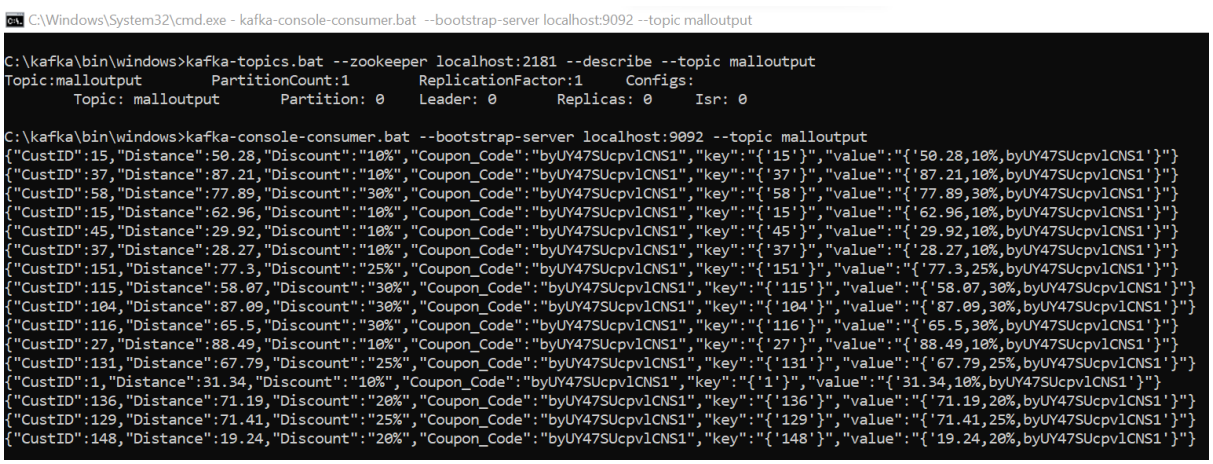
```
Printing Schema of final dataframe received:

root
 |-- CustID: integer (nullable = true)
 |-- Distance: double (nullable = true)
 |-- Discount: string (nullable = false)
 |-- Coupon_Code: string (nullable = false)

-------------------------------------------
Batch: 0
-------------------------------------------
+------+--------+--------+-----------+---+-----+
|CustID|Distance|Discount|Coupon_Code|key|value|
+------+--------+--------+-----------+---+-----+
+------+--------+--------+-----------+---+-----+

-------------------------------------------
Batch: 1
-------------------------------------------
+------+--------+--------+-----------+---+-----+
|CustID|Distance|Discount|Coupon_Code|key|value|
+------+--------+--------+-----------+---+-----+
+------+--------+--------+-----------+---+-----+
```

```
-------------------------------------------
Batch: 2
-------------------------------------------
+------+--------+--------+----------------+------+------------------------------+
|CustID|Distance|Discount|Coupon_Code     |key   |value                         |
+------+--------+--------+----------------+------+------------------------------+
|15    |50.28   |10%     |byUY47SUcpvlCNS1|{'15'}|{'50.28,10%,byUY47SUcpvlCNS1'}|
+------+--------+--------+----------------+------+------------------------------+

-------------------------------------------
Batch: 3
-------------------------------------------
+------+--------+--------+----------------+------+------------------------------+
|CustID|Distance|Discount|Coupon_Code     |key   |value                         |
+------+--------+--------+----------------+------+------------------------------+
|37    |87.21   |10%     |byUY47SUcpvlCNS1|{'37'}|{'87.21,10%,byUY47SUcpvlCNS1'}|
+------+--------+--------+----------------+------+------------------------------+
```

```
-------------------------------------------
+------+--------+--------+----------------+------+------------------------------+
|CustID|Distance|Discount|Coupon_Code     |key   |value                         |
+------+--------+--------+----------------+------+------------------------------+
|58    |77.89   |30%     |byUY47SUcpvlCNS1|{'58'}|{'77.89,30%,byUY47SUcpvlCNS1'}|
+------+--------+--------+----------------+------+------------------------------+

-------------------------------------------
Batch: 17
-------------------------------------------
+------+--------+--------+-----------+---+-----+
|CustID|Distance|Discount|Coupon_Code|key|value|
+------+--------+--------+-----------+---+-----+
+------+--------+--------+-----------+---+-----+

-------------------------------------------
Batch: 18
-------------------------------------------
+------+--------+--------+----------------+------+------------------------------+
|CustID|Distance|Discount|Coupon_Code     |key   |value                         |
+------+--------+--------+----------------+------+------------------------------+
|15    |62.96   |10%     |byUY47SUcpvlCNS1|{'15'}|{'62.96,10%,byUY47SUcpvlCNS1'}|
+------+--------+--------+----------------+------+------------------------------+
```

```
-------------------------------------------
Batch: 97
-------------------------------------------
+------+--------+--------+----------------+------+------------------------------+
|CustID|Distance|Discount|Coupon_Code     |key   |value                         |
+------+--------+--------+----------------+------+------------------------------+
|61    |77.96   |30%     |byUY47SUcpvlCNS1|{'61'}|{'77.96,30%,byUY47SUcpvlCNS1'}|
+------+--------+--------+----------------+------+------------------------------+


-------------------------------------------
Batch: 98
-------------------------------------------
+------+--------+--------+----------------+-------+------------------------------+
|CustID|Distance|Discount|Coupon_Code     |key    |value                         |
+------+--------+--------+----------------+-------+------------------------------+
|144   |84.53   |20%     |byUY47SUcpvlCNS1|{'144'}|{'84.53,20%,byUY47SUcpvlCNS1'}|
+------+--------+--------+----------------+-------+------------------------------+


-------------------------------------------
Batch: 99
-------------------------------------------
+------+--------+--------+-----------+---+-----+
|CustID|Distance|Discount|Coupon_Code|key|value|
+------+--------+--------+-----------+---+-----+
```

Kafka consumer:



Demo available:

Google drive link:
https://drive.google.com/file/d/1CG1Y67LDJHXPTeGVjKsv03Jpjq1yoVjD/view?usp=sharing