Fundamentals of Big Data Analytics
Project Phase 2 Report

Submitted to: Sir Kifayat Alizai

Submitted by:

Muhammad Owais

Muhammad Soban

Wasif Mehboob

Muhammad Shamil Umar

Table of Contents

Main Objective

Previously in phase 1 we cleaned our dataset and performed EDA to better understand our dataset. In this phase we had to select and train a ML model which will be used for recommending products to users and integrate it with apache kafka and Flask with RESTful API.

Implementation

Preprocessing

Before sending data to the ML model for training we need to make sure that all the data is in numeric form. We decided to use the Alternating Least Squares (ALS) recommendation model from pyspark. This model needs 3 columns to train:

- Items
- Rating
- user

So we extracted asin, reviewerID and overall column from or dataset which looks like:

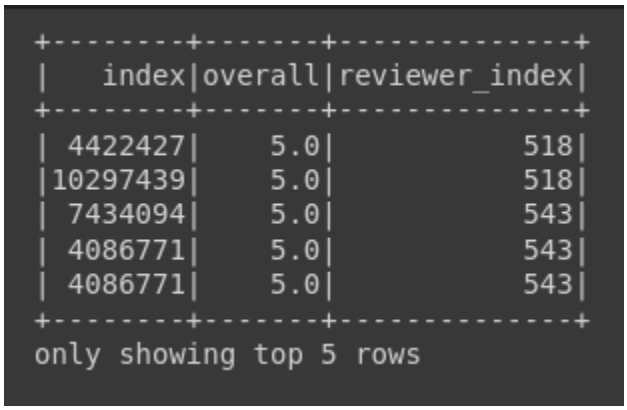
```
+----------+-------+--------------+
|      asin|overall|    reviewerID|
+----------+-------+--------------+
|B01C6DXMX0|    5.0|A1MWPEXTKRZTVU|
|B00BWYVWLO|    5.0|A1QYUQDUVUV05V|
|B01C6DXMX0|    4.0|A15Q1O8A4WWL0M|
+----------+-------+--------------+
only showing top 3 rows
```

Next step is to convert asin and reviewerID. For that we extracted a unique asin and a unique reviewerID. After that we used assigned integer value to each row which resulted in:

```
+----------+-----+
|      asin|index|
+----------+-----+
|0000000116|    1|
|0000000868|    2|
|0000001589|    3|
+----------+-----+
only showing top 3 rows
```

```
+--------------------+-----+
|          reviewerID|index|
+--------------------+-----+
|A0000040I1OM9N4SGBD8|    1|
|A0000074RA15UCBH3ON5|    2|
|A000013090ZI3HIT9N5V|    3|
+--------------------+-----+
only showing top 3 rows
```

After this we used inner join with sql query to join them with main dataframe:



```
+--------+-------+--------------+
|   index|overall|reviewer_index|
+--------+-------+--------------+
| 4422427|    5.0|           518|
|10297439|    5.0|           518|
| 7434094|    5.0|           543|
| 4086771|    5.0|           543|
| 4086771|    5.0|           543|
+--------+-------+--------------+
only showing top 5 rows
```

Reducing dataset:

At this point our dataset was ready and we could send it to our model for training but we can not train our model by using the whole dataset. Firstly it would take a lot of time to train and will consume a lot of ram which would lead to OutOfMemoryError. Secondly even if we somehow train it, the model would take a lot of time to generate products. We came up with 2 ideas:

- Only include those products that are reviewed more than 30 times and have an average of 3 overall rating.
- We include only those users who have reviewed more than 50 times.

We didn't used first idea because:

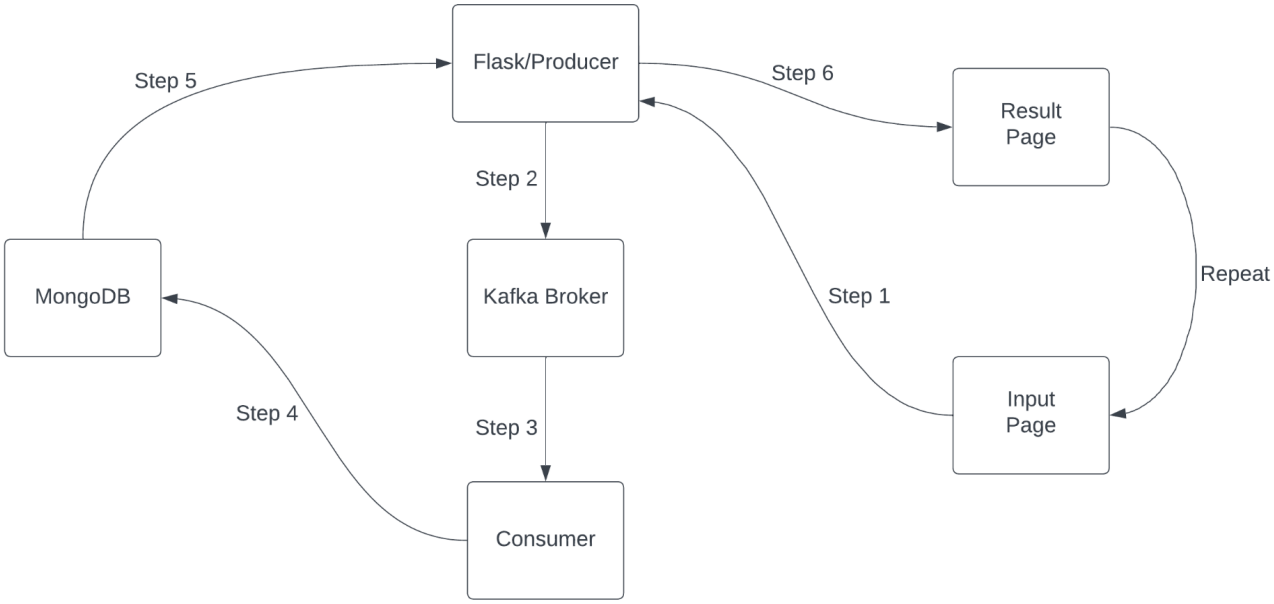
- Nowadays people give fake reviews for money so we had to somehow avoid those people. If those users are caught their accounts are terminated but their reviews still exist there so threshold like this was must.
- Secondly we can not remove any product from the list because even if that product is only reviewed 1 time, it still holds value for users, maybe not for the majority but at least for some.
- Lastly amazon sells every kind of product ranging from normal grocery items to big home appliances. People who live in the USA, UK etc buy a lot of products from amazon on a daily basis so buying 50 items is normal for them so we believe that a second idea is a much better option to go for.

Training Model:

We extracted all the users who have reviewed more than 50 times, divided it into training and testing data with 80% for training and 20% for testing and we fitted it into our model. We used RMSE metric score to evaluate our model. The closer RMSE is to 0, the better our model is for prediction and we were able to get a RMSE score of 1.0328847009803104.

Flask Application

We had to integrate our model to work with flask and apache spark. For apache spark we need a producer which fetches data and a consumer who processes the data. So we decided to make our Flask script a producer which will get reviewerID and productIDs entered by the user from the html page and send it to the consumer. Consumers will send a model for getting recommended products and save the list of products in mongoDB. After 10 seconds of sleep, the producer will fetch the latest entry from mongoDB and send it to the result.html page where it will be displayed. Following flowchart describes whole cycle:



How To Run Our Code

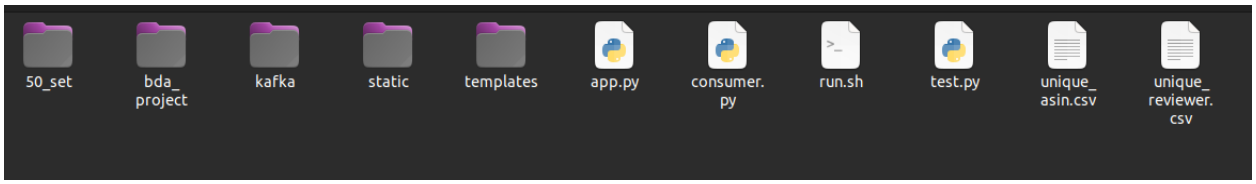
****THIS CODE IS MADE FOR UBUNTU****

Requirements:

1. Ubuntu
2. Python must be installed.
3. Spark must be installed with all global paths.
4. Mozilla Firefox browser must be installed.
5. MongoDB server should be installed.

Steps To Run Code:

Following files will be available to you:



1. Open this folder in terminal
2. Run: ***pip3 install -r bda_project/requirements.txt*** This will install all required libraries.
3. Run: ***source bda_project/bin/activate*** This will activate your virtual environment.
4. Run: ***./run.sh*** This will open all the programs automatically.

Contribution

In this phase Muhammad Soban and Wasif Mehboob worked on designing flask applications and Muhammad Shamil Umar worked on processing data and training models. Integration of model with flask application was done by Muhammad Shamil Umar.