JIGSAW ACADEMY
THE ONLINE SCHOOL OF ANALYTICS

Case study for Spark MLlib

Retail Store Recommendation System

Business Case Scenario:

A retail store has thousands of branches. They have a database of customer sale data collected from the POS(Point of Sale) terminals. They have customers buying at a store as well as an online system to order items for immediate delivery. Whenever a customer logs into the system, a list of products need to be shown as recommendation to the customer. Based on the existing users and products bought, recommend a list of products to the customer when the customer logs into the system. Note that the recommendation is not based only on what the customer has bought before but also on what similar customers have bought. This will use custom dataset and use ALS based recommendation algorithm of Spark-MLLib.

Data Set:

The data set to be used for this case study is a custom data set taken from a retail store data. The data has 8 fields as follows:

Field1 : store id
Field 2: Sale date time
Field 3: A code for the item (Can be Naics code)
Field 4: Kind of business store
Field 5: Item Name
Field 6: Item Cost
Field 7: Customer Id
Field 8: sub item number (If the customer bought multiple items at the same time)

The fields Customer Id and the Item Name are to be used for this case study.

Once the model is built, print the recommendations for following two customers:
43124
99970

Problem Statement:

With the retail store data , build a recommendation model and provide product recommendations for a a couple of customers.

Expected Output:

List the product recommendations for the abovementioned two customers.

Recommendations for customer: 43124

```
Kitchen & Dining
Boots
```

© Jigsaw Academy Pvt Ltd

```
Handbags
Platinum jewelry
Head Lights
```

Recommendations for customer: 99970

```
Perfumes
Kitchen furnishings
Chocolates & Candies
Vehicle Power adapters
Outdoor furniture
```

Solution Approach:

Apache Spark MLlib with dataframe based API `pyspark.ml` can be used to provide a solution for this. Because of the distributed nature of Spark, the solution can scale to handle millions of records in retail data. Also Spark uses distributed machine learning algorithms which can be used here. Spark provides a recommendation model using ALS (Alternate Least Squares) algorithm which specifically can be used for this requirement.

So we can down load the data and use the recommendation algorithms of Spark to build a ALS model and then use the model to provide the recommendations for a customer.

Recommendation Engines:

When we visit any ecommerce site, it has become common to suggest products based on previous buying patterns. Sites like Amazon suggest that people who bought this product also bought these other products. This is based on analyzing the buying patterns of thousands of customers who bought the products that you bought. Such programs are called recommendation engines and machine learning algorithms are commonly used for this purpose. The recommendation has to process millions of records in real time and suggest products in near real time.

Alternate Least Squares (ALS) algorithm is one of the recommendation algorithms and is used by Spark ML library for recommending products. This model uses a column for product ids, a column for user ids and a column for rating. product id and user ids are expected to be integers where as rating is expected to be double or float value. Pipeline APIs are not suitable for this model so you need to use the columns from a dataframe. Given the data for user , product and rating, the recommender model is built. The new dataframe based model does not provide a method for recommending products for a given user, so you need to generate the ratings and then pick the best 5 ratings.

For example a video recommender system may have customer video rating data like below:

```
User_id    video_id   rating
100        20         5
100        10         3
200        30         5
200        10         3
```

From this, you can create a dataframe with three columns as :

```
(100,20,5.0)
```

© Jigsaw Academy Pvt Ltd

```
(100,10,3.0)
(200,30,5.0)
(200,10,3.0)
```

With this dataframe an ALS model can be built and from the model, you can recommend products for a given user.

In the above data, the ratings are explicit as users have given explicit ratings to products. In our data, there is no explicit rating. We can use the number of times a customer has bought as item as the rating. For this type of implicit rating, MLlib provides a flag `setImplicitPrefs` to be set to true. Once the model is built, `model.transform` method can be used to get the rating for a user and product.

Tasks in the project:
Task 1. Get retail data: This is available as `retail_2013.csv`, a comma separated file which is compressed as a zip file `retail_2013.zip`.
Create a separate directory for this case study like `casestudyALS` and copy the zip file and uncompress it in this directory.

Task 2. Load the data in PySpark.
Start your pyspark shell or Jupyter notebook and load the data into a dataframe. You can load the data using the `read.load` command into a dataframe. If you are reading the file from local file system then you can specify it as a regular file system file by using the prefix `file://` and then the path of the file. For example the path with prefix can be `file:///home/hduser/casestudyALS/retail_2013.csv`.

Task 3. Use `StringIndexer` from `pyspark.ml.feature` package to give an index number to each of the 160 distinct products in the data.
Field or column number 4 in the data represents the item name (Column numbering starts with 0). Use this column as the parameter for `StringIndexer` to get a new column added named ProductIndex which is the unique number for each product in the data.

Task 4. Prepare a dataframe containing `ProductIndex` and customer id.
Select the column `ProductIndex` from the above dataframe and column 6 which has the customer id aliasing it as `CustomerId`.

Task 5. Add the column Count that has the count of number of times a customer bought each product.
Use `groupBy` method on the above dataframe on the columns `ProductIndex` and `CustomerId` with `count` as the aggregation. Then change the type of the `count` column to `double` and also change the column name to `Count` (with an uppercase C).

© Jigsaw Academy Pvt Ltd

Now our dataframe is ready to be used for building the ALS model. We can check a few few rows in this dataframe using the `show` command:

```
+-----+------+---+
|items|custid|cnt|
+-----+------+---+
| 66  | 14506|1.0|
| 108 | 16892|1.0|
| 102 | 13853|2.0|
| 90  | 10859|1.0|
| 64  | 13632|1.0|
+-----+------+---+
```

Task 6. Define the ALS recommendation model .
Since the ratings are implicit, use the `setImplicitPrefs(true)` for the ALS model. The ALS model takes following parameters:

Iterations : Number of iterations to try. More iterations may give better model but will take more time. You can start with 5 iterations.
Lambda : A parameter to prevent overfitting. You can start with 0.01
Alpha : Relative weight given to the ratings. You can start with 1.0 We can experiment with different values to get different models.
We also need to specify the columns for User, Item and Ratings.

Task 7. Fit the model with the retail data i.e. train the model with the given data.
The model provides a `fit` method to train the model using our data. Use the dataframe created in task 5 to train the model.

Task 8. Create a dataframe containing `ProductIndex` column
Using `select` command on initial dataframe let us create a dataframe containing `ProductIndex` column. For easy mapping of the index number to the product name, we can add `ProductName` column also to this dataframe.

Task 9. Create a test dataframe for the given user
Using the above product dataframe let us create a test dataframe which containing two more columns `CustomerId` and `Count` besides the columns `ProductName` and `ProductIndex`. The `CustomerId` column will have the given user's id and the `Count` column `0.0` as the initial value.

Task 10. Get top 5 recommendations for the given user
Use `transform` method of the model to get the predicted ratings. We can pass the above test dataframe to this method which will return a dataframe containing the added column `predction` which will be a value between 1 and 0. Order this by descending `prediction` and with limit of 5 to list the top 5 recommendations.

© Jigsaw Academy Pvt Ltd