IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 3, Issue 9, September 2016

Ecommerce User Data Analysis and Product Recommendation Using PredictionIo

Ujwal U J[1], Dr. Antony P J[1], Sachin D N[1]

Department of Computer Science & Engineering, KVG college of engineering, sullia, D.K[1]

Abstract: One of the biggest challenges for software developers in the world today is to build real-world applications. Hence to overcome this problem, the PredictionIO, an open source machine learning server is being used, which provides a step-by-step graphical user interface for developers and data scientists. The PredictionIO helps to evaluate, compare and deploy scalable learning algorithms, and also to evaluate model training status. An API also comes with this system to communicate with the software application to perform the events like data collection to send to training model and Prediction Retrieval [3].

Keywords: HBase, Elasticsearch, Scala, Big Data, Data Mining.

I. INTRODUCTION

E-Commerce online business is playing a major role in the world today. To make it easier and better a Recommendation system is very much required. Predicting the required data is the task to be done. So a PredictionIO framework is used in order to predict the product ratings. PredictionIO is an open source Machine Learning Server Built for the developers and data scientists for developing the predictive engines. The predictive analysis can be done on Spam filtering, Face recognition, Recommendation engines when there exists a large amount of data sets. Usually the retrieval of huge amount of data was a tedious job until the concept of clustering was introduced. Clustering is a process of grouping the data sets into meaningful subclasses called clusters. A cluster always consists of data (or objects) which have similar characteristics to one another but will be dissimilar when compared to the other clusters. Hence clustering forms an efficient approach to retrieve the large data sets. The PredictionIO is gaining much importance nowadays as it is built on the top of open source technology such as Scala, Apache spark, HBase and Elasticsearch. PredictionIO is written in scala to gain the JVM support and a best-distributed system called Spark. It is the very first interesting open source approach that allows the prediction algorithms to run on the Hadoop Platform. Hadoop is used to process the large data sets in the distributed computing environment. Here we propose the PredictionIO framework as it is a dynamic engine which responds to the queries in the real-time. The Separation of Concerns is used to provide the reusability of code, Development of the individual section of the code and its maintainability. It also allows incorporating the data modeling tasks on to the DASE architecture in order to allow data modeling. [1]

II. RELATED WORK

Recommender systems began to appear in the market in 1996. In the context of recommender systems design and implementation, four different approaches can be distinguished [2].

Content-based recommender systems: In this recommender system they try to find products, services or contents that are similar to those already evaluated by the user. In this kind of systems, user's feedbacks (that can be collected in many ways) are essential to support and accomplish recommendations.

Knowledge-based recommender systems: In this model the user profile in order to, through inference algorithms, identify the correlation between their preferences and existing products, Services or content.

Collaborative filtering recommender systems: Here they create/classify groups of users that share similar profiles/behaviors in order to recommend products, services or content that has been well evaluated by the group to which a user belongs.

Hybrid recommender systems: They combine two or more techniques to improve the ''quality'' of recommendations.

Pure content-based systems suffer from what it is known as overspecialization, that is, to specialize to an extreme degree. By trying to find contents similar to the ones users have already seen, the system tends to end up recommending the same group of contents. Since each technique has its own pros and cons, most solutions combine different techniques so facilitating the improvement of their overall performance.

The knowledge-based systems are becoming very important. For example, AVATA is a recommender system for TV content that uses Semantic Web technologies to produce a knowledge-based system that allows the system to infer and reason about ontologies, a well known and powerful tool for intelligent inference.

Systems that make use of collaborative filtering have existed for a long time, such as the well-known Tapestry, a system developed to help users to ''filter'' electronic documents received by e-mail; Ringo, a recommender system for music; Group Lens, a recommender system for

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 3, Issue 9, September 2016

Usenet articles/messages, Movie Lens or PHOAKS a recommender system for links/contents, among others.
Collaborative filtering-based systems propelled commercial services such as Amazon, CDNow and Netflix. The Netflix has invested so much in recommender systems that in 2006, launched a competition of US$ 1,000,000 to the first person who managed to improve their recommender system by a 10%.
Nowadays hybrid systems are gaining momentum. For instance, recommends, a movie recommender system, makes use of semantics in its content-based module together with a collaborative filtering technique. This system makes fairly interesting use of ‘‘semantic feedback’’ from users inside the recommender engine. FOA Finding-the-music, a hybrid system that is entirely based on trust in social networks, also uses semantic analysis of contents [2].

III. IMPLEMENTATION

In order to process and predict the large data set, a Quick Start E-Commerce Recommendation Engine Template is used. The engine template has some of the features by default, they are:-Exclude out-of-stock items, Provide recommendation to new users who sign up after the model is trained, Recommend unseen items only, Recommend popular items only, Recommend popular items if no information about the user is available. Here the PredictionIO of version 0.9.6 has been used. The template takes data from the Event Server by default, they are:- Users’ view events, Users’ buy events, Items’ with categories properties, Constraint unavailable Items set events. The train model is fed with training data and the template treats the view events as training data. If the unseen only parameter is set to true, engine will recommend the unseen items only. And if the parameter is set to seenEvents, then the seenEvents are displayed. To exclude the unavailable events, the constraint unavailableItems is set. PredictionIO has some components, they are;-PredictionIO platform, Event Server and Template Gallery [1].

1. PredictionIO platform-It is the open source machine learning server used by the developers and data scientists.

2. Event server-It is machine learning analytics layer which is responsible for unifying the events from multiple platforms. It continuously collects the data that is provided from the application and the engine will build the predictive models using algorithms. After the data is collected, it will be provided to the engine for model training, evaluation and for data analysis. Later the queries from application will be processed and the results will be predicted for real-time environment.

3. Template Gallery-It is the space to download the engine templates for different machine learning applications.

The general architecture of PredictioIO Engine is shown below.

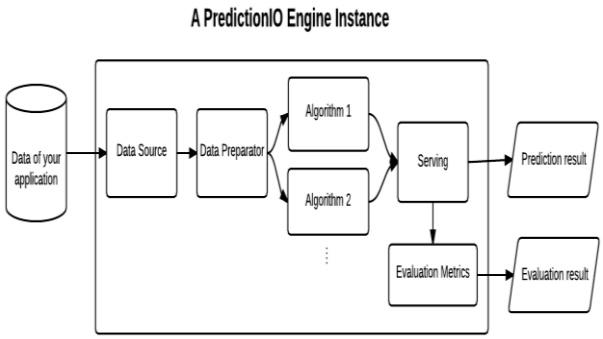


Fig1.A PredictionIO Engine Instance

Engine can be built using following steps

- Initially Linux mint Rosa (64bit) which is an open source OS is installed.
- Then PredictionIO version 0.9.6 is installed, since it can be easily customized.
- Once the PredictionIO is installed, it will be in the path /home/yourname/PredictionIO/bin, and later checked for components like PredictionIO Event Server, HBase and Elasticsearch to be up and running.
- The PredictionIO Event Server is started using the command $ pio eventserver &.
- To start all the components, the command used is $ pio-start-all.
- The status is checked by running the command $ pio status.
- Next a new Engine will be built by using the engine template.
- The next task is to develop an application. Each app will be provided with a unique App ID and Access Key.
- The created applications can be listed using the command $ pio app list.
- Next step is to collect the data. By default the recommendation engine consists of two events- buy and rate.
- Next the event server is checked for whether the events are imported successfully. Later same querying task will be continued to check for importing larger events.
- Now it’s the time to build, train and deploy the engine.
- Building the engine might take much time, which is done by using the command $ pio build –verbose.
- Later the data will be trained by sending it to the training model. Thus finally the Engine will be deployed.[1]

The Proposed Algorithm for PredictionIO

The algorithm that is being used is ALS (Alternating Least Squares). The Apache Mahout’s ALS recommender is a matrix Factorisation algorithm that uses Alternating Least Squares approach. It runs the ALS algorithm in a parallel fashion. This recommendation algorithm is used in E-commerce platform to recommend the products to the users. Mahout’s ALS recommendation algorithm takes the user preferences by item as input and generates an output of recommending items to the user. The input customer

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 3, Issue 9, September 2016

preferences could either be explicit user ratings or implicit feedback such as user's click on a webpage [1].
Alternating Least Squares is a method that alternates between two matrices in a product such as Y=UY' where Y is data. ALS represents a different approach to optimising the loss function. The ALS fixes the each one of the alternatives. When one is fixed, the other one is computed, and vice versa. There are two main benefits of this approach:

- First, this is very easy to parallelize.
- Second, whenever dealing with the implicit datasets, which are usually not sparse.

ALS is much more efficient in these cases.
At present Mahout has a map-reduce implementation of ALS, which is composed of 2 jobs: a parallel matrix factorization job and a recommendation job. The matrix factorization job computes the user-to-feature and item-to-feature matrix given the user to item ratings. Its input includes:

--input: directory containing files of explicit user to item rating or implicit feedback;
--output: output path of the user-feature matrix and feature-item matrix;
--lambda: regularization parameter to avoid overfitting;
--alpha: confidence parameter only used on implicit feedback
--implicitFeedBack: Boolean flag to indicate whether the input dataset contains implicit feedback;
--numFeatures: dimensions of feature space;
--numThreadsPerSolver: number of threads per solver mapper for concurrent execution;
--numIterations: number of iterations
--usesLongIDs: Boolean flag to indicate whether the input contains long IDs that need to be translated
And it outputs the matrices in the sequence file format.
The recommendation job uses the user feature matrix and item feature matrix calculated from the factorization job to compute the top-N recommenders per user its input includes:
--input: directory containing files of user ids;
--output: output path of the recommended items for each input user id;
--userFeatures: path to the user feature matrix;
--itemFeatures: path to item feature matrix;
--numRecommendations: maximum number of recommendations per user, default is 10;
--maxRating: maximum rating available;
--numThreads: number of threads per mapper;
--usesLongIDs: Boolean flag to indicate whether the input contains long IDs that need to be translated;
--userIDIndex: index for user long IDs (necessary if usesLongIDs is true);
--itemIDIndex: index for item long IDs (necessary if usesLongIDs is true)

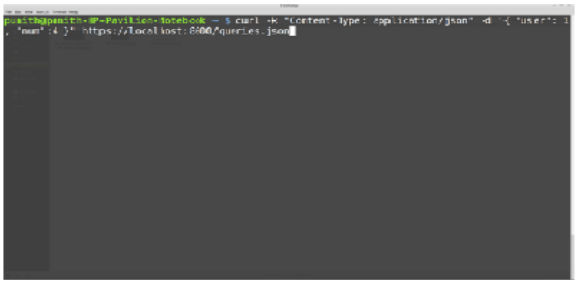
And it outputs a list of recommended item ids for each user. The predicted rating between user and item is a dot product of the user's feature vector and the item's feature vector. Hence ALS algorithm is well applicable for Recommendation systems.

IV. EVALUATION AND PERFORMANCE ANALYSIS

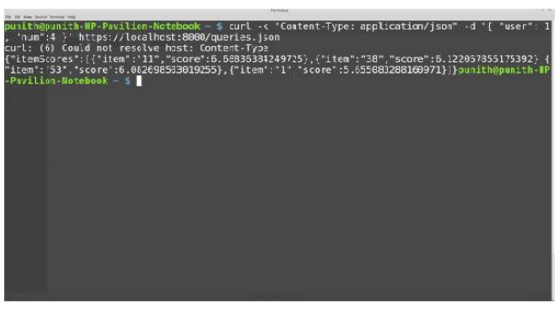
In predictionIO, evaluation can be done in few lines of code. The evaluation Metric used here is a function which takes QPA-tuple as a input and outputs the scores.QPA tuple-(Querry, Predicted Result, Actual Result) the accuracy and precision can be found by using accuracy metric and precision metric.
Performance is the process of determining if a program behaves as expected. In the process, one may discover errors in the program under test. When testing reveals an error, the process used to determine the cause of this error and to remove it is known as Debugging. Testing and Debugging are often used as two related activities in cyclic manner. Testing activities takes place throughout the software lifecycle. In the proposed project a PredictionIO framework is being used, where the main intension is to recommend the product ratings to a customer based on the previous views of that product.
Computation: The time complexity of machine learning algorithms often times depends on the number of features used. That is, the more features one uses for prediction, the more time it takes to train the model.
Prediction performance: Often times there will be features that, when used in training, will actually decrease the predictive performance of a particular algorithm.
When the userId and num are provided, the score for the product is displayed, which is the expected output.

Consider the different cases:

Case1: INPUT:
UserId is 1 and the num is 4



OUTPUT:
The expected output is:



IARJSET

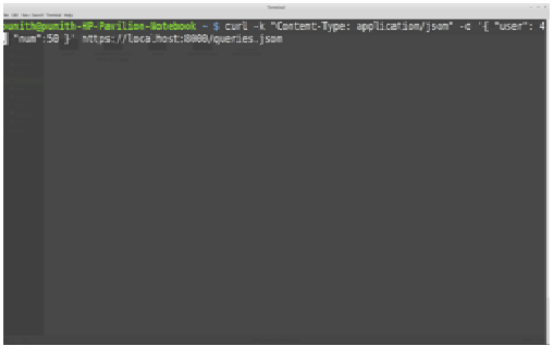
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

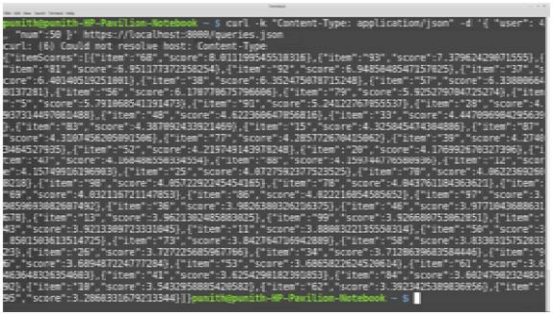
ISO 3297:2007 Certified

Vol. 3, Issue 9, September 2016

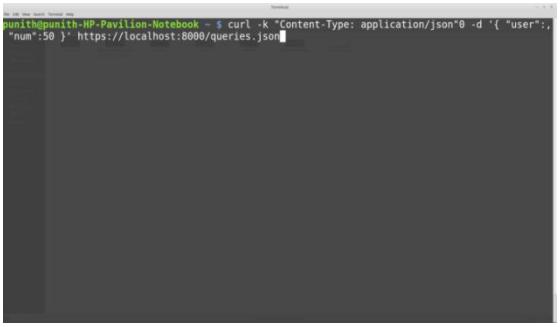
Case2: INPUT:
UserId is 4 and num is 50



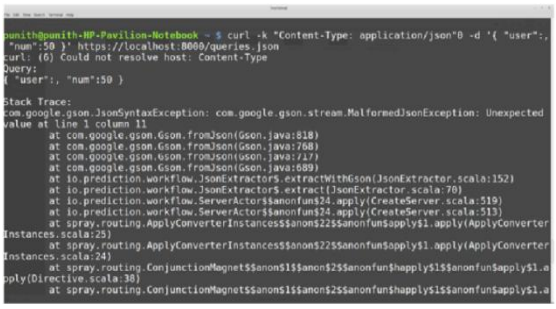
OUTPUT:
The expected output is:



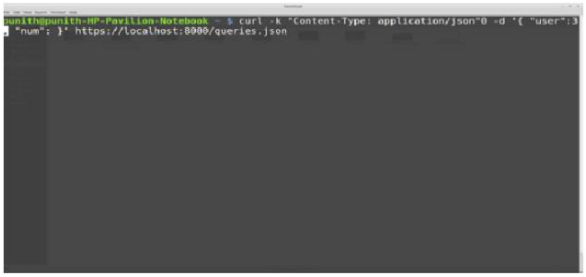
Case3 INPUT:
UserId is not inputted and the num is 50



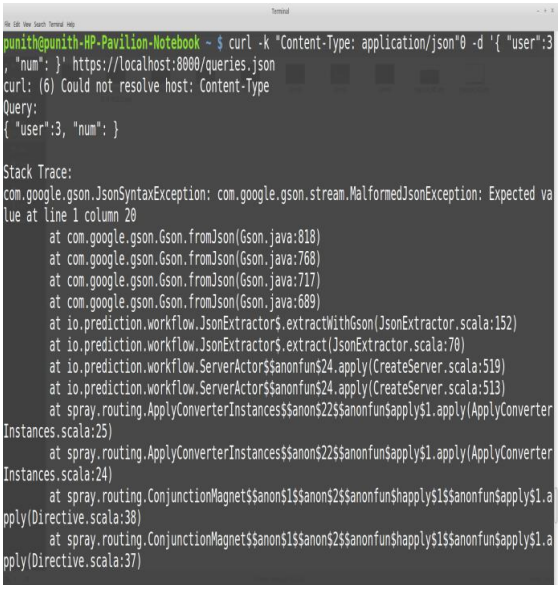
OUTPUT:
The expected output is:



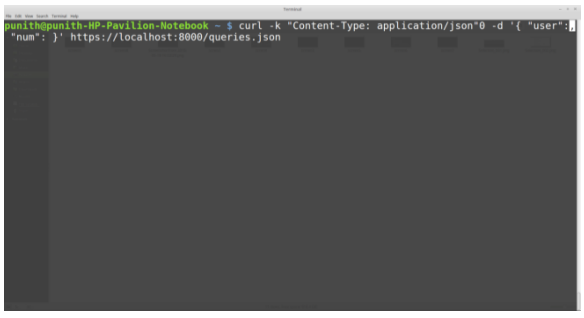
Case 4: INPUT:
UserId is 3 and the num is not inputted



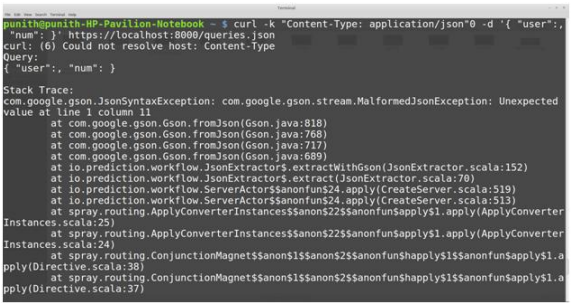
OUTPUT:
The expected output is:



Case 5: INPUT:
UserId is not inputted and the num is also not inputted



OUTPUT:
The expected output is:



IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified

Vol. 3, Issue 9, September 2016

Hence in this project, for the given inputs, the expected outputs are obtained. Thereby the conclusion is that, the system has high performance.

V. RESULT

The expected output is to obtain the score for a product to a given user ID. The score must be displayed in the descending order as per the views that is the highest rating at first followed by the next highest and so on.

The output obtained is shown below in fig(3). The input is given by the command- "$ curl –k "Content-Type:/application/json" –d '["user": "20", "num": "1000"]' " .

As the values for user id and number changes, the respective outputs will be obtained.

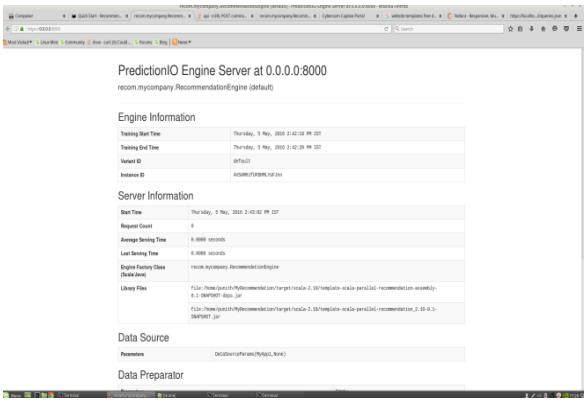


Fig-2 Deployed Engine in port: 8000

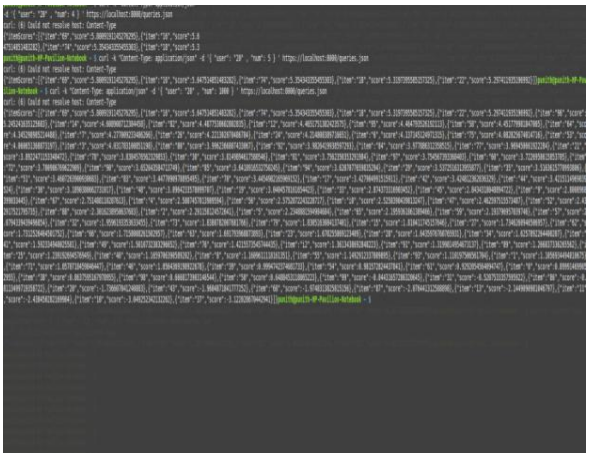


Fig-3 Output of the Product Recommendation Engine

VI. CONCLUSION

In this project the PredictionIO framework is used for Recommendation of products. This method is very easy and efficient as it reduces the work of developing the code from scratch. Recommendation system helps the customers to get ideas about how good the product is and what is the market rating at present. The main effort is to make the task easy, simple and efficient.

REFERENCES

[1] https://docs.prediction.io.com
[2] Reference-J. Ben Schafer, Joseph Konstan, John Riedl Group Lens Research Project Department of Computer Science and Engineering
[3] University of Minnesota
[4] Reference-Walter Carrer-Neto, Maria Luisa Hernandez-Alcaraz, Rafael Valencia-Garcia ⇑, Francisco Garcia-Sanchez Departamento de Informática y Sistemas, Universidad de Murcia, 30100 Murcia, Spain