Predicting Amazon Rating Using Spark ML and Azure ML

Monika Mishra, Amogh Mahesh, Aakanksha Tasgaonkar

Department of Information Systems, California State University

Los Angeles

e-mail : [mmishra2@caltstatela.edu](mailto:mmishra2@caltstatela.edu), [amahesh@calstatela.edu](mailto:amahesh@calstatela.edu), [atasgao@calstatela.edu](mailto:atasgao@calstatela.edu)

**Abstract:** The aim of the project is the application several Machine Learning models to Amazon product review data in order to analyze, predict rating and recommend products. For this purpose we have used algorithms in both Azure ML and Spark ML.The analyzed Amazon product review dataset contained 15 attributes and has about 6.93 millions records. Comparative conclusions have been made related to efficiency of Spark ML asnd Azure ML for this dataset.

**1. Introduction**

E-commerce has developed rapidly in the past years. Both businesses and customers have embraced online sales as a convenient way to shop. The purchasing decisions are made based on reviews and ratings. Therefore, it’s very important for the business to have the insights of business and predict it’s future based on reviews and ratings.

An added advantage of using this dataset was a good understanding of the dataset and past experience of data analysis on this data. Previously, we analyzed the same dataset to find significant insights such as product popularity, ratings and reviews statistics, customer’s sentiments, etc. This project has been the continuation of our work in terms of applying our knowledge of Machine Learning algorithms.

Our dataset is of size 3.63 GB of TSV file format. It has 15 columns. Star\_rating column has been considered as the label column.

2. Related Work

Bhavesh [1] and Max [2] performed analysis on Amazon product review dataset but their goals and techniques were quite different from ours.

Bhavesh’s work was to classify Amazon product review to positive and negative. He performed sentimental analysis for one of the baby products. The tools used in his approach were Python, GraphLab and S Frame. Our focus was mainly on predictive analysis. We used the Azure ML and Spark ML platform to predict product rating.

Another similar research study was done by Max [2]. Max performed descriptive analysis using *Sparklyr* platform. In contrast, our research is about predictive analysis. We did the predictive analysis by using various machine learning models while Max’s work was restricted just to descriptive analysis.

3. Hardware Specifications

For this project, we have used Microsoft Azure Machine Learning Studio and Databricks community edition to implement Spark ML. We have also used Hadoop spark cluster on the Oracle Big Data Cloud platform for the rating prediction. The specification is given below:

|  |  |  |
| --- | --- | --- |
| **Azure** | **Databricks** | **Oracle BDCE \*** |
| * Memory – 10 GB * Nodes - 1 * No. of experiment - * 100 | * Memory - 6 GB * Nodes- 1 * Driver (0.88 cores, 1 DBU), 0 Worker​ | * Memory – 180 GB * Storage – 682 GB * Nodes – 6 * OCPU - 12 |

Table 1. Hardware specifications

4. Background/Existing Work

In our project we have implemented several algorithms in AzureML and SparkML. Most of our models are based on previous existing works.

**4.1 Recommender**

Our Matchbox Recommender module is based on the lab work where we constructed and evaluated a recommender using a sample of user movie rating data. Movie Ratings and Movie Titles datasets were joined in AzureML. The four different score recommenders recommend different metrics. After this step, each metric is evaluated by Evaluate Recommender module and the success of the model is determined.

Our SparkML recommender is based on Collabarative Filtering project, that uses movie titles and rating datasets. Similar to AzureML, the datasets were joined in Spark ML.We have used ALS (Alternating Least Squares) algorithm to build the recommender. The evaluation is conducted to show us the RMSE.

**4.2 Regression**

The Decision Forest Regression and the Boosted Decision Tree Regression is based on the study of the prediction of heating load. In the prediction of heating load for the energy efficiency dataset, we used Linear Regression and Decision Forest Regression model. We used cross-validation module and hypertune module for the training purpose. We also used Permutation feature module to check the features importance and accordingly pruned the features. In Spark ML, for our project we have used Decision Tree Regression and Gradient Boosted Tree Regression. In the lab work we didn’t perform the above two models but the work is similar to the lab work where we predicted arrival delay of the flight dataset. We used the Linear Regression model in the lab and crossvalidation module was used for the training purpose. RMSE was used as an evaluation parameter.

**4.3 Text Analytics**

The text analytics part of our project is based on the lab work where we performed sentiment analysis on the tweets data on Spark ML. Logistic Regression was used to predict the sentiment level – 1 for positive and 0 for negative. Tokenizer was used to split the text into individual words, StopWordsRemover to remove common words. A HashingTF class was used to generate numeric vectors from the text values. A Logistic Regression algorithm to train a binary classification model. So, the stopwords are removed and the sentiments are predicted with respect to relevant text. The pipeline is used as an estimator and run with fit() method on training data to train the model.

5. Our Work

We implemented various models in Azure ML and Spark ML to predict the star\_rating. As there is size restriction in Azure ML and Databricks, we sampled the original dataset of 3.63 GB to 73 MB on Azure ML platform by using Partition and Sample module. We used stratified sampling by selecting the star\_rating column to ensure that the sampled dataset is a true representative of the original dataset. It took around 5.30 minutes for sampling.

For Oracle BDCE, we used the full dataset.

A screenshot of a cell phone

Description automatically generated

Fig 1. Sampling

**5.1 Matchbox Recommender (Azure ML)**

The goal of the recommender is to provide Amazon customers with recommendations for product categories based on their previous ratings, as well as the ratings of other users. Moreover, the model has a feature to predict the future ratings by user for a category.

The dataset was split into training and testing fractions by .75 to .25 ratio. After the split, the training fraction is connected to Train Matchbox Recommender module and test fraction to four Score Recommender modules. Each of the four score recommenders represent different metrics: a) item recommendation b) rating prediction, c) similar items and d) similar users.

The model took 6 minutes to run.

A screenshot of a cell phone

Description automatically generated

Fig 2. Item Recommendation (From Rated Items)

This option enables evaluation mode, and the module makes recommendations only from those items in the input dataset that have been rated. This model is evaluated by Normalized Discounted Cumulative Gain (NDCG) which in this case is 0.97. That is a very encouraging result.

A screenshot of a cell phone

Description automatically generated

Fig 3. Related Items (Categories)

This feature finds related items. Results are evaluated by the similarity of the ratings using both L1(Manhattan) and L2 (Euclidian) average normalized discounted cumulative gain (NDCG) averaged over all the pairs selected. In this case it’s 0.97 for both which is pretty good.

A screenshot of a cell phone

Description automatically generated

Fig 4. Rating Prediction

This feature predicts ratings. When one predicts ratings, the model calculates how a given user will react to a particular item, given the training data. This model is evaluated by the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) which in this case is 0.65 and 1.22 respectively.

A screenshot of a cell phone

Description automatically generated

Fig 5. Related Users

This feature finds related users. Results are evaluated by the L1 similarity NDCG and L2 similarity NDCG which in this case is 0.87 in both the cases.

**5.2 Decision Forest Regression (Azure ML)**

We took 2% sample of the original dataset and split the dataset into 70:30 ratio for the training and testing. We used Cross-validation model and the Tune Model Hyperparameters for the training.

A screenshot of a cell phone

Description automatically generated

Fig 6. Evaluation Metrics (Decision Forest Regression)

The Tune Model Hyperparametrs provided a better result than the cross-validation model.

Permutation feature model was also used to check the importance of various feature columns. Removing less important features resulted in the slightly better performance of the Decision Forest Regression model.

A screenshot of a cell phone

Description automatically generated

Fig 7. Permutation Feature Importance (Decision Forest Regression)

The model took 30 minutes to run.

**5.3 Boosted Decision Tree Regression (Azure ML)**

We took 2% sample of the original dataset and split the dataset into 70:30 ratio for the training and testing. We used Cross-validation model and the Tune Model Hyperparameters for the training.

A screenshot of a cell phone

Description automatically generated

Fig 8. Evaluation Metrics (Boosted Decision Tree Regression)

The Tune Model Hyperparametrs provided a better result than the cross-validation model.

A screenshot of a cell phone

Description automatically generated

Fig 9. Permutation Feature Importance (Boosted Decision Tree Regression)

Permutation feature model was also used to check the importance of various feature columns. Removing less important features didn’t improve the evaluation result of the Boosted Decision Tree Regression model.

The model took 1 hour and 30 minutes to run.

**5.4 Collaborative Filtering Recommender (Spark ML - Databricks)**

Our SparkML recommender is based on Collabarative Filtering project. The dataset used was a cleaned and transformed dataset created in Azure ML. The columns used are customer\_id, product\_category and star rating. Product\_category feature was transformed into integer using StringIndexer function.The label was star\_rating. The dataset is split to train and test fractions by .7 to .3 ratio. We have used ALS (Alternating Least Squares) algorithm to build the recommender. Additionally, we’ve defined parameters and used fit method to train the model. Then we test the model to see the recommended category for each user.

It took around 30 minutes to run and gave an RMSE = 1.73.

**5.5 Text Analytics Using Logistic Regression (Spark ML - Databricks)**

Our SparkML text analysis is done using Logistic Regression. We used the classification model to predict if the sentiment will be positive or negative. A rating greater than 3 is considered positive else negative. We used pipeline algorithm with Tokenizer to split the text into individual words, StopWordsRemover to remove common words such as "a" or "the" that have little predictive value. A HashingTF class to generate numeric vectors from the text values. A Logistic Regression algorithm to train a binary classification model. So, the stopwords are removed and the sentiments are predicted with respect to relevant text. The pipeline is used as an estimator and run with fit() method on training data to train the model.

|  |  |
| --- | --- |
| **Metrics** | **Value** |
| TP | 32503 |
| FP | 3920 |
| TN | 3516 |
| FN | 1799 |
| Precision | 0.89 |
| Recall | 0.95 |

Table 2. Evaluation Metrics – Logistic Regression (Databricks)

It took around 4 minutes to run and gave an AUR of 0.71.

**5.6 Decision Tree Regression (Spark ML - Databricks)**

We used Decision Tree Regression model for the rating prediction. As most of the feature columns in our data set was categorical, we used StringIndexer feature to add index to them. Vectorassemler was also used. Pipeline feature was used. The dataset is split into 70:30 ratio for training and testing. Crossvalidation method is used for training with number of folds as 5. The pipeline is used as an estimator and run with fit() method on training data to train the model.

The model was then evaluated using RegressionEvaluator. The Root Mean Square Error(RMSE) is 1.19. It took 5 minutes to run.

**5.7 Gradient Boosted Tree Regression (Spark ML - Databricks)**

We used Gradient Boosted Tree Regression model for the rating prediction. As most of the feature columns in our data set was categorical, we used StringIndexer feature to add index to them. Vectorassemler was also used. Pipeline feature was used. The dataset is split into 70:30 ratio for training and testing. Initially, we tried to train the model with Crossvalidation method but it was running for longer than 2 hours and the cluster was detaching itself. So we used TrainValidation Split method with train ratio of 0.8. The pipeline is used as an estimator and run with fit() method on training data to train the model. The model was then evaluated using RegressionEvaluator. The Root Mean Square Error(RMSE) is 1.11. It took 15 minutes to run.

**5.8 Collaborative Filtering Recommender (Spark ML – Oracle BDCE)**

Our SparkML recommender is based on Collabarative Filtering project. The full size of 3.63 GB is used. The columns used are customer\_id, product\_category and star rating. Product\_category feature was transformed into integer using StringIndexer function.The label was star\_rating. The dataset is split to train and test fractions by .7 to .3 ratio. We have used ALS (Alternating Least Squares) algorithm to build the recommender. It took around 1 hour to run and gave an RMSE = 2.10.

**5.9 Text Analytics Using Logisting Regression (Spark ML – Oracle BDCE)**

Our SparkML text analysis is done using Logistic Regression. We used the classification model to predict if the sentiment will be positive or negative. A rating greater than 3 is considered positive else negative. The model used in the Databricks were used here.

|  |  |
| --- | --- |
| **Metrics** | **Value** |
| TP | 1651435 |
| FP | 177198 |
| TN | 189777 |
| FN | 605277 |
| Precision | 0.90 |
| Recall | 0.96 |

Table 3. Evaluation Metrics – Logistic Regression (Oracle BDCE)

It took around 20 minutes to run and gave an AUR of 0.74.

**5.10 Decision Tree Regression (Spark ML - Oracle BDCE)**

We used Decision Tree Regression model for the rating prediction. As most of the feature columns in our data set was categorical, we used StringIndexer feature to add index to them. Vectorassemler was also used. Pipeline feature was used. The dataset is split into 70:30 ratio for training and testing. Crossvalidation method is used for training with number of folds as 2. The pipeline is used as an estimator and run with fit() method on training data to train the model.

The model was then evaluated using RegressionEvaluator. The Root Mean Square Error(RMSE) is 0.98. It took 1 hour and 20 minutes to run.

**5.11 Gradient Boosted Tree Regression (Spark ML – Oracle BDCE)**

We used Gradient Boosted Tree Regression model for the rating prediction. As most of the feature columns in our data set was categorical, we used StringIndexer feature to add index to them. Vectorassemler was also used. Pipeline feature was used. The dataset is split into 70:30 ratio for training and testing. We used TrainValidation Split method with train ratio of 0.8. The pipeline is used as an estimator and run with fit() method on training data to train the model. The model was then evaluated using RegressionEvaluator. The Root Mean Square Error(RMSE) is 1.03. It took 1 hour to run.

**6. Conclusion**

Azure ML and Spark ML are powerful platforms for machine learning. Below is the summary of our experiment:

|  |  |  |
| --- | --- | --- |
| **Azure ML (73 MB)** | | |
| **Model** | **RMSE** | **Time Taken** |
| Matchbox Recommender | 1.22 | 6 minutes |
| Decision Forest Regression | 1.14 | 30 minutes |
| Boosted Decision Tree Regression | 0.91 | 1 hour 30 minutes |
| **Spark ML – Databricks (73 MB)** | | |
| **Model** | **RMSE** | **Time Taken** |
| Collaborative Filtering | 1.73 | 30 minutes |
| Decision Tree Regression | 1.19 | 5 minutes |
| Gradient Boosted Tree Regression | 1.11 | 15 minutes |
| **Model** | **AUR** | **Time Taken** |
| Text analytics using Logistic Regression | 0.71 | 4 minutes |
| **Spark ML – Oracle BDCE (3.63 GB)** | | |
| **Model** | **RMSE** | **Time Taken** |
| Collaborative Filtering | 2.10 | 1 hour |
| Decision Tree Regression | 0.98 | 1 hour 20 minutes |
| Gradient Boosted Tree Regression | 1.03 | 1 hour |
| **Model** | **AUR** | **Time Taken** |
| Text analytics using Logistic Regression | 0.74 | 20 minutes |

Table 4. Summary/Comparison Table

Following conclusion can be drawn from our project:

* Recommendation model is implemented to predict the item recommendation and rating prediction .
* It can help in finding customers with the preferred items.
* Based on RMSE, for recommendation model, AzureML performed better than the Spark ML.
* Boosted Decision Tree performed better than the Decision Forest in rating prediction in Azure ML.
* Text Analysis – Helps to understand customer sentiment and satisfaction of a product.

**6. Challenges Faced**

We faced few challenges while doing the project. These are listed below:

* In Azure ML, we received the error : *Error 0138: Memory has been exhausted, unable to complete running of module. Process exited with error code -2.* We fixed the error by saving the experiment under a new name and then deleting the old experiment. This released the memory.
* In Databricks, we received the following error many times while running the experiment : *The spark driver has stopped unexpectedly and is restarting. Your notebook will be automatically reattached.* The root cause of the issue is unknown to us.
* In Databricks, for GBT Regression model, we were using cross validation to train the dataset. It was taking long to run and around 2 hours of the run, the experiment ran into error with the error message: *Internal error, sorry. Attach your notebook to a different cluster or restart the current cluster.* So, we changed our training method to TrainValidation Split method.
* While uploading larger dataset in Databricks, an error message was shown, so we completed the experiment with 73 MB size.

### References

[1] A. S. Jain, “Top 10 Benefits of Online Shopping (and 10 Disadvantages),” *ToughNickel*, 2018. [Online]. Available: https://toughnickel.com/frugal-living/Online-shopping-sites-benefits.

[2] M. Woolf, “Playing with 80 Million Amazon Product Review Ratings Using Apache Spark,” minimaxir, 02-Jan-2017. [Online]. Available: https://minimaxir.com/2017/01/amazon-spark/.

[3] Github Link: https://github.com/monika2403/mmishra2/tree/master/CIS%205560

[4] Dataset Link: https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon\_reviews\_multilingual\_US\_v1\_00.tsv.gz