Predicting Yelp User Rating Using Azure ML, Spark ML and Oracle DBCE

Oluwatofunmi Alabi, Raymond Delgado, Sweta Tripathi, Uchechukwu Okafor

Department of Information Systems, California State University

Los Angeles

e-mail : [oalabi@caltstatela.edu](mailto:mmishra2@caltstatela.edu), [rdelga40@calstatela.edu](mailto:amahesh@calstatela.edu), [stripat@calstatela.edu](mailto:atasgao@calstatela.edu), [uokafor3@calstatela.edu](mailto:atasgao@calstatela.edu)

**Abstract:** The aim of the project is the application of several Machine Learning models to review data in order to analyze, predict rating of business based on Yelp reviews. For this purpose we have used both Azure ML and Spark ML platforms.The analyzed Yelp users’ reviews dataset contains 9 attributes and has about 6.6 millions records. Comparative conclusions have been made related to efficiency of Spark ML and Azure ML for this dataset.

**1. Introduction**

The rise of the internet has provided a platform for people to share their ideas with the world. Both businesses and customers have embraced this fact which has led to an increased importance on online reviews. The overall rating of a business is based on an aggregate of user reviews. Therefore, it’s very important for the business and consumer to have the insights of a business and predict its future quality based on the reviews of others.

We have decided to use this dataset because it clearly states the whole review of the user. Upon our analysis of the dataset, we found significant insights such as the user sentiments, ratings, and other users' reaction to the reviews, etc. This project is our group’s attempt on applying our knowledge of Machine Learning algorithms to develop models that can be useful for companies like Yelp.

Our dataset is of size 4.74 GB of CSV file format. It has 9 columns. Star 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. Hamid[3] performed analysis on users' tweets about Starbucks to gain insight for that company, which is more specialized than our project.

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 future user 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. The third research study was performed by Hamid[3]. They used latent semantic analysis and support vector machines to gain key insight for Strabucks from users’ sentiment. However, we use the user Azure ML and Databricks to predict a company’s performance from user sentiment.

**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: 10GB  Nodes : 1  No. of experiment: 100 | Memory: 6 GB  Nodes: 2  Driver: (0.88 cores, 1 DBU), 0 Worker​ | Memory: 180GB  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. It must be noted that our group decided to use a multiclass logistic regression model which was not covered by the lab work.

**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 Collaborative 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 Decision Forest** **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 a hypertune module for the training purpose. We also used the 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**

After downloading our dataset, we did some data engineering to clean up our dataset and get it ready for the various models we trained. Data analysis was done on the dataset for better understanding of the relationships between our various columns. We implemented various models in Azure ML and Spark ML to predict the stars column. As there is size restriction in Azure ML and Databricks, we sampled the original dataset of 4.73 GB to 78 MB on Azure ML platform by using Partition and Sample module. We used stratified sampling by selecting the star 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.

**5.0 Data Engineering and Analysis (Jupyter Notebook)**

Our dataset had 9 colucommas and so we had to delete them using a python code. User\_id, business\_id, and review\_id all had alphanumeric entries. To build a successful model, these values were converted to integers. We took a subset of 120,000(78MB) rows from the full dataset for our Azure ML and Databricks Spark ML. The dataset was queried to understand its content and below is a brief summary of the subset.

* Total number of rows: 120,000
* unique user\_id number: 93921
* unique business\_id: 14766
* unique review\_id: 120,000

unique stars: 5mns namely: user\_id, text, date, review\_id, business\_id, cool, useful, funny and stars. All entries in the dataset had a “b” and inverted

The first subset was used successfully in building the ALS Recommendation model and the text analysis model but because the unique values of our user\_id and business\_id were not categorical, we had error messages while building our Gradient Boosted Tree Regression Model and Decision Tree Regression model in Databricks. This prompted us to make a dataframe copy from the first subset, normalize the user\_id, review\_id and business\_id columns, write it into local computer with the name “scaled\_subset” before importing the normalized dataset named scaled\_subset into databricks. We successfully built our Gradient Boosted Tree Regression Model and Decision Tree Regression model in Databricks with the normalized dataset without error messages. For Azure MLwe successfully built our models with the first subset created because we could easily use the Normalize Data module to normalize our dataset.

All data engineering and data analysis files are attached to this project

**5.1 Matchbox Recommender (Azure ML)**

The goal of the recommender is to provide Yelp users with predictions on future ratings based on the ratings of other users.

The dataset was split into training and testing fractions by .75 to .30 ratio. After the split, the training fraction is connected to train the Matchbox Recommender module and test fraction to the Score Recommender module. The score recommenders represent how accurate our model rating predictions are. The metrics that are used are Mean Absolute Error(MAE) and Root Mean Square Error(RMSE).  
Here are the result of this model:

* **MAE: 1.254**
* **RMSE: 1.558**

We also used the Permutation Feature Importance module to find what features influence the results of the model. All the features had a positive importance score--which means that all the features impact our model. Overall, the model took 6 minutes to run.

**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 regular training, Cross-validation model and the Tune Model Hyperparameters for the training. The Tune Model Hyperparameters provide the best results. Which are listed below:

* Area Under the Curve: .710
* Accuracy: .863
* Precision: .864
* Recall: .998
* F1 Score: .926

These results are promising because it shows the model is more accurate than random guessing. It must be noted that random guessing has an AUC of .5. This means that our model’s predictions are 14.1 percent more accurate than guessing.

We also used the Permutation Feature Importance module to find what features influence the results of the model. All the features had a positive importance score--which means that all the features impact our model. Overall, the model took 2 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, the Tune Model Hyperparameters, and default model training for the training of our model.

Once again the Tune Model Hyprepamaters provided the best results. Which are as follows:

* Area Under the Curve: .716
* Accuracy: .869
* Precision: .874
* Recall: .990
* F1 Score: .928

These results are better than the results that we got from the Decision Forest algorithm. However, it must be noted that this model does not find as many relevant elements. Nonetheless, Boosted Decision Tree is ideal over Decision Forest.

We also used the Permutation Feature Importance module to find what features influence the results of the model. All the features had a positive importance score--which means that all the features impact our model. Overall, the model took 4 minutes to run.

**5.3 Multiclass Decision Forest (Azure ML)**

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

The Tune Model Hyperparatmers provided the best results. Which are listed below:

* Overall Accuracy: .434
* Average Accuracy: .773

These are lacking but they still can show important insight. For instance, our model is capable of grouping the user ratings based on the stars. However, the model as whole is not generating the results that we are looking for.

We also used the Permutation Feature Importance module to find what features influence the results of the model. All the features had a positive importance score--which means that all the features impact our model. Overall, the model took 3 minutes to run

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

Our SparkML recommender is based on Collaborative Filtering project. The dataset used was cleaned and transformed in Jupyter Notebook. The columns used are user\_id, business\_id and stars(label). The star column was transformed into a positive(1) and negative(0) values. User\_id and business\_id were both transformed from alphanumeric values to numeric values during the data engineering process in Jupyter notebook. The dataset was splitted into train and test fractions by .7 to .3 ratio. We used the ALS (Alternating Least Squares) algorithm to build the recommender. Additionally, parameters were defined, model was trained using the fit method and tested.

It took around 20 minutes to run and gave an RMSE = 0.685.

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

Text analysis in databricks was done using the Logistic Regression algorithm to predict if the sentiment will be positive or negative. Values of column stars > 2 are considered positive(1) else negative(0). Tokenizer was used 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. Logistic Regression algorithm was used to train a binary classification model. Model trained and tested. We calculated the following metrics listed in the table below.

* AUR: 0.89
* TP: 26711
* FP: 2577
* TN: 5671
* FN: 1217
* Precision: 0.91
* Recall: 0.96

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

We used the Decision Tree Regression algorithm to build a model that could predict if a business would get a positive rating or a negative rating. However, our dataset columns weren’t categorical. User\_id had about 93921 unique values in 120000 entries and the review\_id had 120000 unique values and business\_id had a total of 14766 unique values. To train this model successfully, we had to normalize these columns. Normalization of the columns was done on jupyter notebook before the dataset was imported. Values of column stars were transformed with the command, stars > 2 = positive(1) else negative(0). The dataset is split into 70:30 ratio for training and testing. Cross Validation method is used for training with number of folds as 5. The TrainValidationSplit was also used in training this model. Both models were evaluated using the multiclassificationevaluator and AUC results compared. Time taken to run is 7 minutes.

TrainValidationSplit AUC = 0.675

CrossValidation AUC = 0.5.

* TP: 27967
* FP: 8196
* TN: 0
* FN: 0
* Precision: 0.77
* Recall: 1

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

We used the Gradient Boosted Tree Regression algorithm to build a model that could predict if a business would get a positive rating or a negative rating. However, our dataset columns weren’t categorical. User\_id had about 93921 unique values in 120000 entries and the review\_id had 120000 unique values and business\_id had a total of 14766 unique values. To train this model successfully, we had to normalize these columns. Normalization of the columns was done on jupyter notebook before the dataset was imported. Values of column stars were transformed with the command, stars > 2 = positive(1) else negative(0). The dataset is split into 70:30 ratio for training and testing. We couldn’t train the model with Cross Validation method because it ran for 3hours 30minutes and the cluster detached itself. The dataset was trained using the TrainValidationSplit method and tested. The model was then evaluated using MultiClassificationEvaluator.

* AUR: 0.675
* TP: 28083
* FP: 8204
* TN: 7
* FN: 12
* PRECISION: 0.774
* RECALL: 0.99

**5.8 Errors that occured on Oracle BDCE**

We were unable to work on Oracle BDCDE due to the errors that we kept on getting. The error messages stated that string type was not supported. We were able to identify that the error occurred when we tried to train our model and the error message we had stated that the business\_id column is a stringtype and not numeric. Thus, we discovered that the Oracle BDCE is treating the business\_id as a string.

This is what we did to ensure that busines\_id was treated as an integer:

* remove all rows with null (missing value) in Jupyter notebook and count the total number of rows. this remained at 120,000 rows showing no rows had missing or null values.
* We analyze the dataset in Azure ML using the clean missing data module and after visualization, the total number of rows remained at 120,000. This clearly shows that we have no missing value in our dataset.
* we deleted the file in hadoop using the remove command and imported a fresh again just to be sure and we still had the same error message.
* We also included a schema and set busienss\_id as an integer
* We also set inferschema as false in the .py file as and tried again but got the same error.

Sadly, nothing we did was able to resolve the issue.

**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 | 1.55 | 6 minutes |
| **Model** | **AUC** | **Time Taken** |
| Boosted Decision Tree | .716 | 4 minutes |
| Decision Forest | .711 | 2 minutes |
| **Model** | **Average Accuracy** | **Time Taken** |
| Multiclass Forest | .773 | 3 minutes |
| **Spark ML – Databricks (73 MB)** | | |
| **Model** | **AUR** | **Time Taken** |
| Text analytics using Logistic Regression | 0.89 | 4 minutes |
| Decision Tree Regression | 0.675 | 7 minutes |
| Gradient Boosted Tree Regression | 0.675 | 35 minutes |
| **Model** | **RMSE** | **Time Taken** |
| Collaborative Filtering (ALS) | 0.685 | 24 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 worse 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 business.

**7. 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 Azure ML, we noticed that our models were biased to large value of start ratings. To resolve this issue we use the sampling module to sample our dataset for a wide range of starts.
* 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.
* The major challenge that we faced with our dataset was that the unique values user\_id, business\_id, and review\_id were too large. To resolve this issue we normalized these features and remove duplicates
* In Oracle BDCE, we encountered an error that states data type string was not supported. This error occured when we were training(dtModel = dt\_tvs.fit(train) our model and it was related to the business\_id. We tried many methods to ensure that the business\_id was an integer. However, nothing was able to resolve this issue.

### 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]H.Shirdastian,”Using big data analytics to study brand authenticity sentiments: The case of Starbucks on TwitterUsing big data analytics to study brand authenticity sentiments: The case of Starbucks on Twitter”[I](https://www.sciencedirect.com/science/journal/02684012)nternational Journal of Information Management, Oct-2019. [Online] Available: <https://www.sciencedirect.com/science/article/abs/pii/S0268401217302657>

[4] Github Link: <https://github.com/Uokafor3/5560-Intro-to-Big-Data>

[5] Dataset Link: <https://www.kaggle.com/darshank2019/review>

[6] <https://spark.apache.org/docs/latest/ml-features.html>

[7] <https://medium.com/@connectwithghosh/basic-data-preparation-in-pyspark-capping-normalizing-and-scaling-252ee7acba7d>

[8] <https://towardsdatascience.com/machine-learning-with-pyspark-and-mllib-solving-a-binary-classification-problem-96396065d2aa>