Outbrain Ad Prediction using Spark ML and Azure ML

Anjali Baldawa
925-577-5193
California State University
Los Angeles
abaldaw@calstatela.edu

Dhruvi Patel
562-486-9904
California State University
Los Angeles
dpatel86@calstatela.edu

Digesh Kansara
562-450-9556
California State University
Los Angeles
dkansar@calstatela.edu

Verusca Aimie Capuno
415-707-9950
California State University
Los Angeles
vcapuno@calstatela.edu

Abstract: In this project, we will work on the advertising data of "Outbrain" to create an algorithm that predicts customer ad clicks using Machine Learning and Spark ML. The goal is to predict which recommended content or ads will users likely click based on certain features. In Azure ML, we sampled and did an analysis of the data using five different algorithms and three in Spark ML Databricks and Oracle BDCE. We also did a comparison of the results of AUC, accuracy, precision and recall to determine which algorithm yields the best results for our prediction.

1. Introduction

Targeting the right customers is still a major challenge in online marketing given the fact that we now live in a digital era. We chose this topic because online marketing is rapidly taking over the traditional marketing strategies. Companies are now choosing to advertise their products and/or services online such as in various social media platforms and websites that consumers often visit. Our topic is important because advertising online can be costly if companies are advertising to customers that will not likely buy their products. In this project, we will be working with Outbreak dataset to create prediction models using Machine Learning, Spark ML and Oracle BDCE. Our prediction models can help recommend what ads/contents will likely get clicked based on certain features. This can help companies to take corrective measures. Below is the process of how we integrated this project to predict ad clicks:

- Master training data: 5 GB
- Sampling/Filtering data via Jupyter Notebook
  - Azure ML: 40MB
  - Databricks: 105MB
  - Oracle BDCE: 1GB
- In the dataset, we have selected the following feature columns: Display_id, Document_id, platform, Campaign_id, Advertiser_id, ad_id
- Building/Analysis of 5 algorithms on Azure ML: Logistic Regression, Support Vector Machine, Two-Class Boosted Decision Tree, Two-Class Decision Forest and Two-Class Decision Jungle
- Building/Analysis of 3 algorithms on Spark ML and Oracle BDCE: Logistic Regression, Random Forest and Gradient Boosting
- Visualization and comparison of the models

2. Related Works

(G Limaye, P Jaiswal, V Gopinath - pdfs.semanticscholar.org) in this they predicted ad click – through rates CTR which is a popular learning problem that is central to multibillion dollar online advertising industry, whereas we considered the ads must be recommended for appropriate users. In the (J Hoachuck, S Singh, L Yamada - cs229.stanford.edu) the platform provides information with numerical problems and generated the output the likelihood of an ad being clicked. In this they used Amazon web services (AWS) Redshift which uses massive parallel processing to manage and query the large dataset and apache spark on Microsoft azure and google cloud platforms to train the model using distributed machine learning algorithms over the cloud. We tried to use several algorithms to be used in Azure ML and then use those algorithms in spark ML and oracle BDCE to check the accuracy and AUC of the algorithm.

3. Importing/Loading Dataset

We downloaded our data from Kaggle and to conduct our predictive analysis, we imported some python libraries to lay out pre-executed Machine learning models as well as to sample data into a reduced csv file of 40MB.

4. Azure ML

We uploaded the dataset to Azure ML to create an experiment and consolidated three csv files. As part of data cleaning, we used the "project columns" module to select necessary feature columns. Additionally, we used the "normalize data" with the transformation method set to "minmax" and selected all columns by name. We also brought in the "split column" module and set the following properties.

- Splitting mode: splitting Rows
- Fraction: 0.7
- Randomized split: selected
- Random split: 1234

We then prepared the data for training and testing into the following multiple algorithms to measure and compare their accuracy and AUC.

4.1 Logistic Regression

From this algorithm, we selected the column "clicked" for our prediction and obtained the following output using score and evaluate model:

- AUC: 0.898
- Accuracy: 0.893

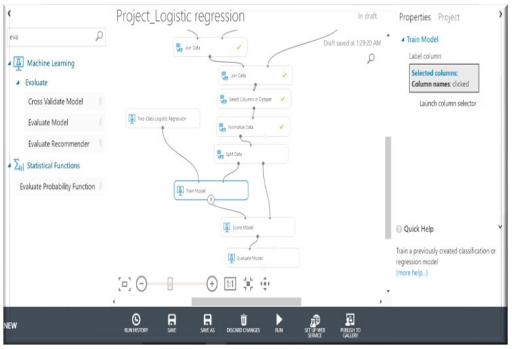


Figure 1. Logistic Regression Model in Azure ML

4.2 Support Vector Machine

The following results were achieved using the "Two-class support vector machine" model into the experiment:

- AUC: 0.855
- Accuracy: 0.882

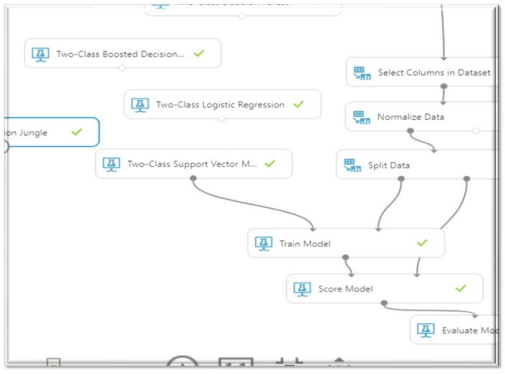


Figure 2. Support Vector Machine model in Azure ML

4.3 Two-Class Boosted Decision Tree

In this algorithm, the properties were adjusted as shown below:

- Create trainer mode: Single Parameter
- Maximum number:20
- Minimum number: 10
- Learning rate: 0.2
- Number of trees constraints: 100

With this experiment, the AUC resulted with .996 and the accuracy was .989 as illustrated below.

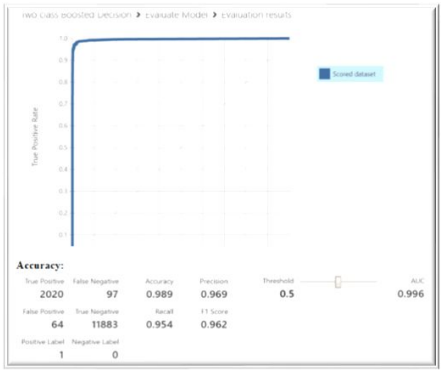


Figure 3. AUC score ot Two-Class Boosted Decision Tree

4.4 Two-Class Decision Forest

We set the properties as detailed below for this algorithm which yielded an AUC of .995 and an accuracy of .974

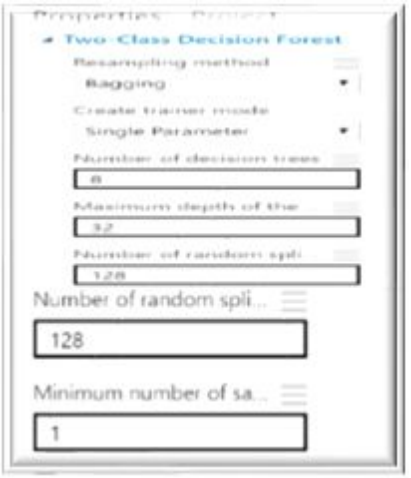


Figure 4. Two-class Decision Forest property

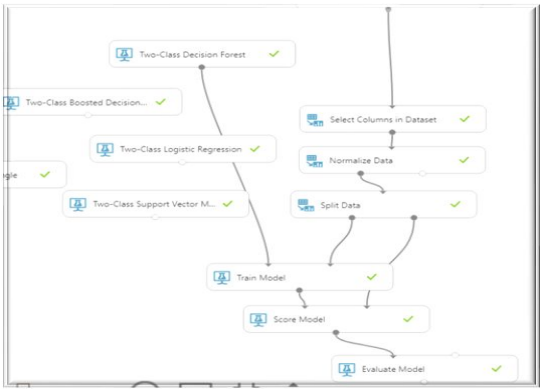
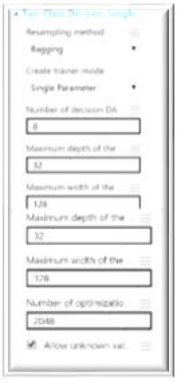


Figure 5.Two-class Decision Forest model in Azure ML

4.5 Two-Class Decision Jungle

Using this algorithm, the properties were set as follows:



AUC: .998
Accuracy: .989

Figure 6. Two-class decision jungle property

5. Results Comparison (Azure ML)

To compare and get a better understanding of the results, we created bar graphs to show the AUC, Accuracy, Precision, Recall and Time outputs of the 5 algorithms displayed below.

Algorithm	Accuracy	AUC	Precision	Recall	Time(Minute)
Logistic Regression	0.893	0.893	0.775	0.409	3
Support Vector Machine	0.882	0.855	0.742	0.333	4.5
Two class Decision Boosted	0.989	0.996	0.969	0.954	5
Two class Decision Forest	0.974	0.995	0.996	0.838	5
Two class Decision Jungle	0.989	0.998	0.997	0.932	4

Table 1. Comparison graph AzureML

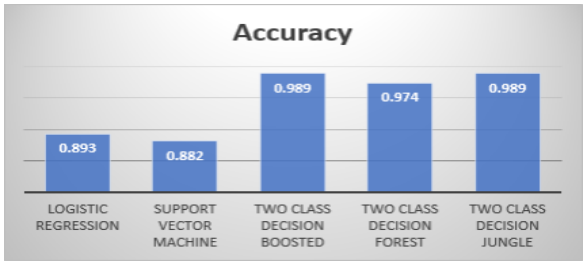


Figure 7. Comparison graph (accuracy) Azure ML

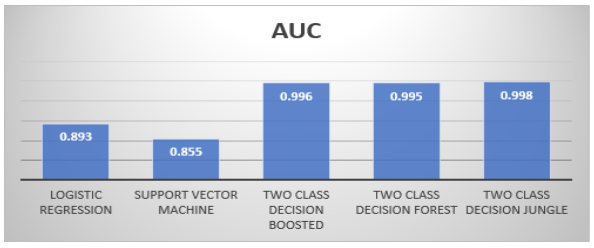


Figure 8. Comparison graph (AUC) Azure ML

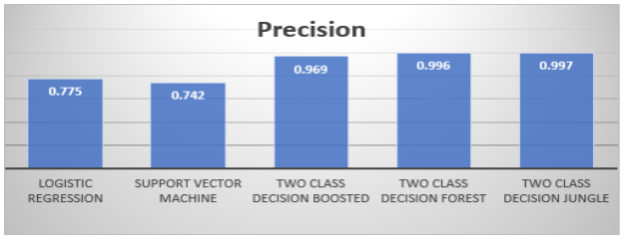


Figure 9.Comparison graph (Precision) Azure ML

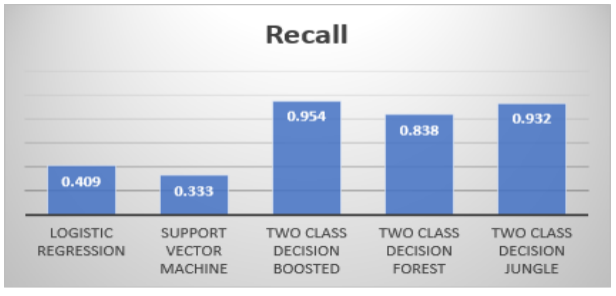


Figure 10. Comparison graph (Recall) Azure ML

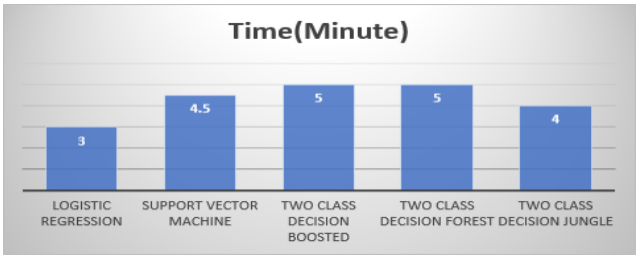


Figure 11. Comparison graph (Time) Azure ML

6. Spark ML (Databricks)

In addition to Azure ML, we sampled our dataset to do ad prediction using spark in Databricks. We created a cluster and used a version compatible with our project. A table was created from the uploaded csv file (35MB) with data types. For databricks, we used the algorithms that yielded the lowest score as well as the top 2 highest values from Azure ML to determine if databricks will generate the same results.

6.1 Logistic Regression

For this algorithm, we imported the Logistic_regression.ipynb file and ran the codes shown below to calculate AUC and accuracy.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluation = MulticlassClassificationEvaluator(
    labelCol="trueLabel", predictionCol="prediction", metricName="accuracy")
accuracy = evaluation.evaluate(prediction)
print("Accuracy of Logistic Regression is: ",accuracy)
```

```
Accuracy of Logistic Regression is:  0.8506215324109839
```

Figure 12. Logistic Regression code

Results:

```
print("Test Error = %g" % (1.0 - accuracy))
```

```
Test Error = 0.149378
```

```
rf_evaluator = MulticlassClassificationEvaluator(labelCol="trueLabel", predictionCol="prediction")
rf_auc = rf_evaluator.evaluate(prediction)
print("AUC for Logistic Regression is= ", rf_auc)
```

```
AUC for Logistic Regression is=  0.7819618641388818
```

Figure 13. AUC and Accuracy results for Logistic Regression

6.2 Random Forest

We imported the Random_Forest.ipynb notebook and played all the codes in the cells shown below which generated an AUC of 0.939 and an accuracy of .943 as illustrated below.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluation = MulticlassClassificationEvaluator(
    labelCol="trueLabel", predictionCol="prediction", metricName="accuracy")
accuracy = evaluation.evaluate(prediction)
print(accuracy)

0.9431866421395217

Command took 2.71 minutes -- by a user at 5/6/2020, 10:56:26 PM on unknown cluster

#Following Code will give total Error in project

print("Test Error = %g" % (1.0 - accuracy))

Test Error = 0.0568134

rf_evaluator =  MulticlassClassificationEvaluator(labelCol="trueLabel", predictionCol="prediction")
rf_auc = rf_evaluator.evaluate(prediction)
print("AUC for Random Forest is= ", rf_auc)

AUC for Random Forest is=  0.9398119382945996

Command took 6.30 minutes -- by a user at 5/6/2020, 11:48:52 PM on unknown cluster
```

Figure 14. Random Forest codes and AUC/Accuracy output

6.3 Gradient Boosting

Following the same steps as above, we ran the codes below from Gradient_boost.ipynb notebook and the results are as follows:

AUC: .876
Accuracy: .897

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluation = MulticlassClassificationEvaluator(
    labelCol="trueLabel", predictionCol="prediction", metricName="accuracy")
accuracy = evaluation.evaluate(prediction)
print(accuracy)

0.8978065802592223

Command took 26.91 seconds -- by a user at 5/6/2020, 11:49:32 PM on unknown cluster

print("Test Error = %g" % (1.0 - accuracy))

Test Error = 0.102193

Command took 0.01 seconds -- by a user at 5/6/2020, 11:49:32 PM on unknown cluster

rf_evaluator =  MulticlassClassificationEvaluator(labelCol="trueLabel", predictionCol="
rf_auc = rf_evaluator.evaluate(prediction)
print("AUC for Gradient Boost is= ", rf_auc)

AUC for Gradient Boost is=  0.8760900114611925

Command took 35.14 seconds -- by a user at 5/7/2020, 12:02:29 AM on unknown cluster
```

Figure 15. Gradient Boosting codes and AUC/Accuracy output

7. Results Comparison (Spark ML Databricks)

Algorithm	accuracy	AUC	Precision	Recall	Time Taken(Minute)
Logistic Regression	0.853	0.753	0	0	50
Decision Tree Classifier	0.943	0.941	0.88	0.727	23
Random Forest classifier	0.952	0.949	0.918	0.744	40
Gradient Boosted tree Classifier	0.898	0.875	0.984	0.314	31

Table 2. Comparison graph Databricks

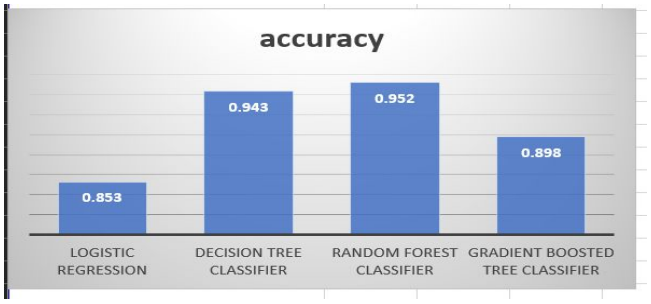


Figure 16. Comparison graph (accuracy) Databricks

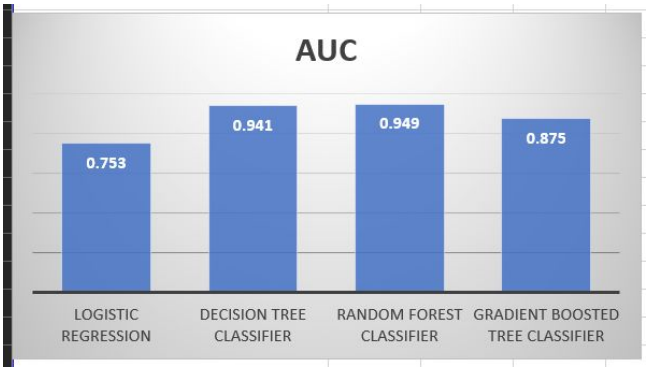


Figure 17. Comparison graph (AUC) Databricks

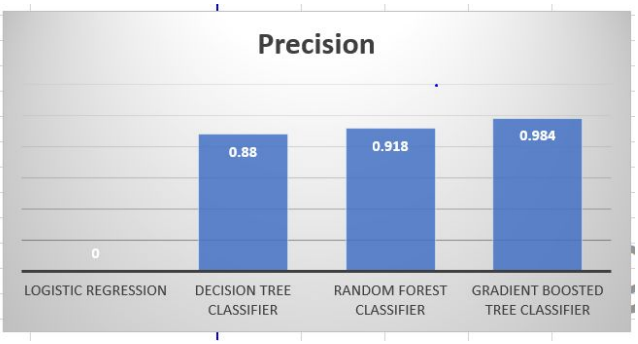


Figure 18. Comparison graph (Precision) Databricks

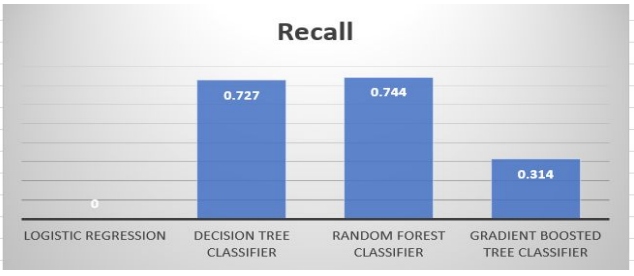


Figure 19. Comparison graph (Recall) Databricks

Figure 20. Comparison graph (Time) Databricks

From the graphs image shown above, we are able to conclude that the Random Forest algorithm yields the highest value of AUC and accuracy.

8. Spark (Oracle BDCE)

We also sampled our dataset (1GB) to Oracle BDE cluster through the provided terminal 129.150.76.160 and imported the data using the proper scp command. We exported the same notebook files used for Databricks and used the same scp command to import to Oracle. We ran these files using the "spark-submit" code to measure AUC and Accuracy

output.The following results for each algorithm were generated.

Algorithm	accuracy	AUC	Precision	Recall	Time (Minutes)
Logistic Regression	0.83	0.753	0	0	25
Decision Tree Classifier	0.938	0.935	0.878	0.737	60
Random Forest classifier	0.932	0.928	0.92	0.707	50
Gradient Boosted tree Classifier	0.92	0.92	0.806	0.609	120

Table 3. Comparison graph Oracle BDCE

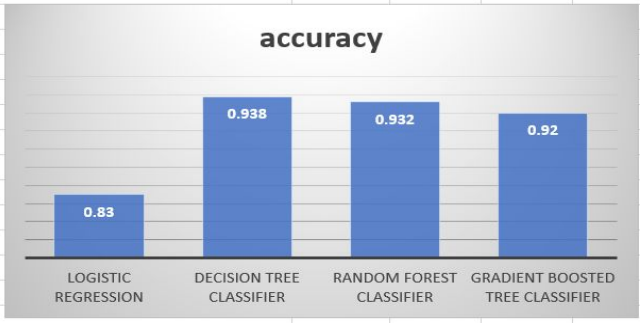


Figure 21. Comparison graph(Accuracy) Oracle BDCE

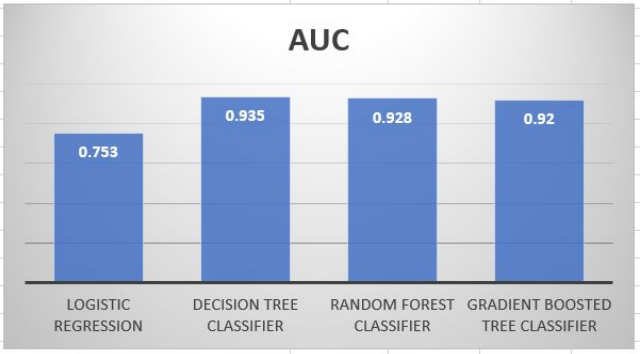


Figure 22. Comparison graph(AUC) Oracle BDCE

Figure23. Comparison graph(Precision) Oracle BDCE

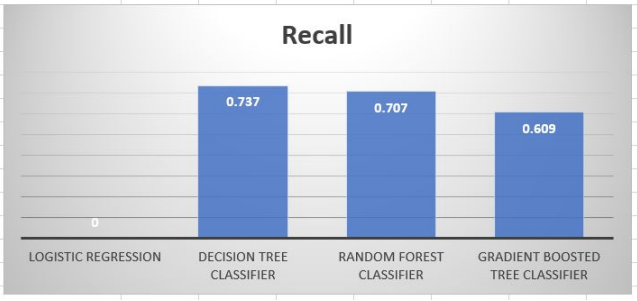


Figure 24. Comparison graph(Recall) Oracle BDCE

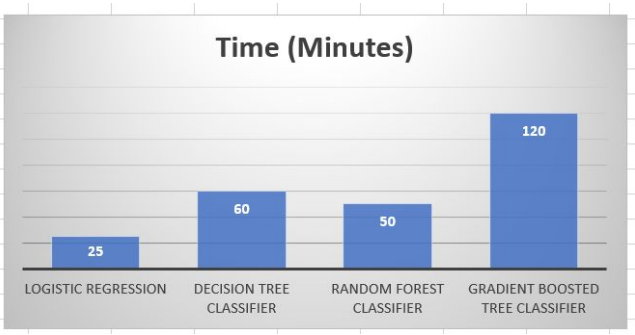


Figure 25. Comparison graph (Time) Oracle BDCE

9. Permutation Feature Importance (Azure)

To measure the importance of our features, we included the "Permutation feature importance" model on each of the algorithms used from Azure ML and determined the increase in our prediction error. The following graph illustrates the scores of each feature from each algorithm used.

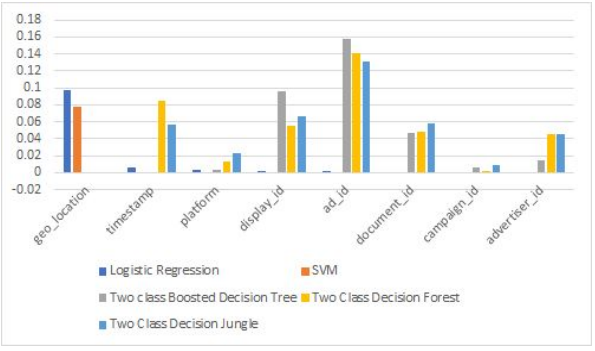


Figure 26. Graph of Permutation Feature Importance (Azure ML algorithms)

10. Conclusion

Prediction is an important key in helping to forecast which recommended content users will likely click and can be achieved by using algorithms. In conclusion, based on selected features and analysis of multiple algorithms sampled from both Azure ML and Spark ML, the Gradient Boosted and Random Forest yielded the highest values of AUC and accuracy out of all algorithms.

11. References

- https://www.kaggle.com/c/outbrain-click-prediction/overview/description
- https://www.semanticscholar.org/paper/Outbrain-Click-Prediction-Hoachuck-Singh/270046171591fe3bf17fbe2e3a59baa27ebf9bc9
- https://medium.com/kaggle-blog/outbrain-click-prediction-competition-winners-interview-2nd-place-team-brain-afk-darragh-610d072fa3fc

12. Github URL

https://github.com/anjalibaldawa/Outbrain-Ad-Prediction