Predicting fraud of AD click using Traditional and Spark ML

Neha Gupta, Hai Anh Le, Maria Boldina, Jongwook Woo
Department of Information System, California State University
Los Angeles,CA, USA
[e-mail: *ngupta8@calstatela.edu, hle55@calstatela.edu,
mboldin@calstatela.edu, jwoo5@calstatela.edu]
*Corresponding author: Neha Gupta

Abstract

This paper is to build predictive analysis models using traditional and Big Data methods to determine whether a smart phone app will be downloaded after clicking an advertisement. We have used data named "TalkingData AdTracking Fraud Detection Challenge", which is of 7GB and given by a Kaggle competition. Four classification models are implemented with this massive data set in order to predict fraud in both traditional and Big Data methods. We define it fraud when the user clicked on an advertisement without downloading. The traditional platform has a resource limitation to build models with data set over a giga-byte so that we generate a sample data for the traditional models and adopt the full data set for the models in the Big Data Spark ML systems. We also present the accuracy and performance of the models implemented in both traditional and Big Data systems.

Keywords: Fraud, ad clcik, predictive analysis, hadoop, big data, spark, machine learning

1. Introduction

TalkingData is the largest independent big data service company in China. Their network covers 70% of the mobile services nationwide with 3 billion ad clicks per day. Amongst those clicks, 90% are potentially fraudulent. Click fraud is happening at an overwhelming volume leading to misusage of data and wasting money. Hence, Kaggle (a platform for predictive modelling and analytics competitions from the U.S.) has partnered up with TalkingData to help resolve this issue.

Here, the data set becomes too large to store and process, which initiates adopting Big Data. Big Data is defined as non-expensive frameworks, mostly on distributed parallel computing systems, which can store a large-scale data and process it in parallel. A large-scale data means a data of giga-bytes or more, which cannot be processed or expensive using traditional computing systems [4]. Hadoop and Spark are popular Big Data platforms and Spark is a popular computing engine for Big Data predictive analysis.

2. Related Work

Feng et al [1] broadly focuses on click fraud detection by
understanding the legitimacy of the click itself. It concentrates on taxonomy of mobile ad frauds and discusses about different aspects of mobile ad frauds and flow of user interactions with advertisement platform. Priyanka et al [2] predicts mobile fraud transaction using traditional and big data approach. The domain is different from advertisement. Besides, it does not present the experimental result of

multiple nodes in a cluster that should show the performance of the models.

In the paper, we focus on the usage patterns of real users not automated click bots. Our hypothesis stems from the fact these bots are designed to create fraudulent activities in an automated way. We are using SMOTE algorithm to balance the data skew. Furthermore, we adopt cloud computing (AWS and Azure) for storage and computation required for Big Data, Spark for building and evaluating models.

3. Data processing and constraints

3.1 Data Processing

In the dataset, there is only 0.19% of the Ad shown in **Fig. 1**. We categorize the data set to: downloaded as positive (1) and not downloaded as negative (0).

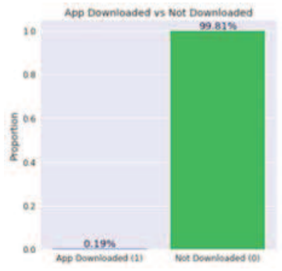


Fig. 1. Actual Data Comparison

Although after writing a code to reduce the data size, we had to partition and sample the data set in order to use 8% of 1GB, especially for the traditional platform.
We adopt two data balancing methods: SMOTE and Stratified Split.

3.2 Processing Imblanced Data

SMOTE stands for Synthetic Minority Over Sampling Technique which takes a subset of data from the minority class and creates new synthetic similar instances. It helps balancing data & avoid overfitting. SMOTE helps to generate data more as increasing percent of minority class from 0.19% to 11%.

Then, we ensure that the output dataset contains a representative sample of the values in the selected column. That is, when we randomly sample the data set, the results are not all 0's. We use 8% of the 1GB for the testing the models to implement in the traditional systems.

4. Traditional Predictictive Analysis

Microsoft's Azure Machine Learning Studio is used to build predictive models to classify the fraud. Two-Class Classification model is select as it is binary classification. The data set is composed of 7 feature columns that determine the prediction. It also has a label column which contains the actual values: 1 and 0. Our goal is to create a scored label column that contains predicted values (1 and 0) to compare with the actual values.

Two decision tree algorithms are selected: Decision Jungle and Decision Forest. Decision trees often perform well on imbalanced datasets because their hierarchical structure allows them to learn signals from both classes.

In the classification model, there are four main measures: accuracy, precision, recall and F1 score. It is important to correctly choose the appropriate metric. Both precision and recall work well if there's an uneven class distribution, which is our case. Additionally, False Positives (FP), which indicate the model predicted an app was downloaded when in fact it wasn't, are more important than False Negatives (FN) for our prediction because focusing on minimizing FP will help saving money and better targeting the customers and the advertisements. Therefore, given that we have an uneven dataset and FP cost more, precision is the key metric for our goal.

4.1 Two-Class Decision Jungle

Fig. 2 shows how we implement two class decision jungle model. In this model, we used 8% of the data, setting up SMOTE to 5000% to balance the data, using 70:30 Split train, setting *cross validation* as 10 folds. We also use tune model hyper-parameters and the score and evaluate models.

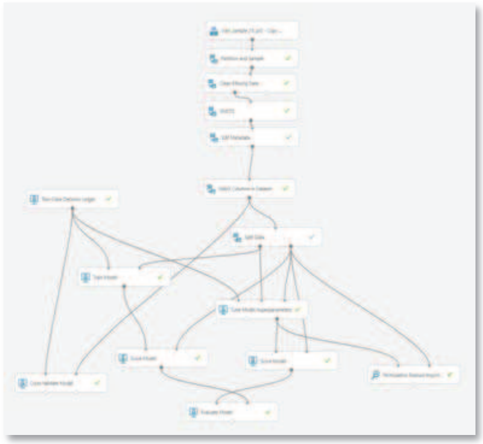
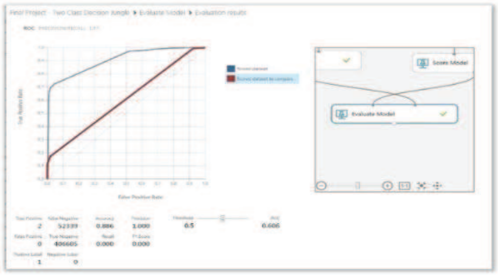


Fig. 2. Two-Class Decision Jungle model on Azure ML

Fig. 3 shows the evaluation results of the models to compare without/with tuning models:

Result #1: AUC = 0.606 (without tune model)

The result **2** of **Fig. 3** shows the better AUC.


Final Project · Two Class Decision Jungle ▸ Evaluate Model ▸ Evaluation results
Score Model
Evaluate Model
False Positive Rate
True Positive Rate

True Positive	False Negative	Accuracy	Precision	Threshold	AUC
2	52339	0.886	1.000	0.5	0.606
False Positive	**True Negative**	**Recall**	**F1 Score**		
0	406605	0.000	0.000		
Positive Label	**Negative Label**				
1	0				

Fig. 3. Two-Class Decision Jungle AUC score

Result #2: AUC = 0.905 (with tune hyper-parameters)

4.2 Two-Class Decision Forest

Two Class Decision Forest model is implemented for the second experiment. In this model, we use 8% of the data by setting up SMOTE to 5000%, using 70:30 Split train. Then, we set up cross validation as 10 folds by using tune model hyper-parameters. And, score and evaluate modules are added to evaluate the models and its result is as follows:

AUC = 0.987, precision = 1.0, TP = 508

We find out that the importance features to affect the result is: app and channel. As implementing a model with only these features, it is observed that the AUC increased from 0.987 to 0.997. However, the precision dropped from 1 to 0.968.

In order to improve the precision, we moved the threshold from 0.5 to 0.8 and the precision increased from 0.968 to 0.992 with lower FP = 377.

In the 8 columns, we have *is_attributed* column as **actual values** and scored labels column as **predicted values**.

4.3 Comparison between Two Models

Table 1. Comparison between two-class decision models

	Two-Class Decision Jungle	Two-Class Decision Forest
AUC	0.900	0.997
PRECISION	1.000	0.992
RECALL	0.001	0.902

	Two-Class Decision Jungle	Two-Class Decision Forest
TP	35	47,199
FP	0	377
TN	52,306	406,228
FN	406,605	5,142

Table 1 shows the experimental result of two models in Azure ML. It shows that Two-Class Decision Forest is better in AUC and *Recall* with less FN and even more TP.

5. Big Data with Spark ML

We have selected Binary Classification algorithms for Spark ML as well. It predicts a category to either downloaded (1) or not downloaded (0). The two main classes that we used are *Decision Tree* and *Random Forest Classifiers*. The following 6 features are extracted to transform data to get better accuracy for fraud prediction:

Feature -1: Prepare time based feature by extracting day of the week and hour of the day from the click time.
Feature -2: Prepare feature by grouping clicks by the combination of (Ip, Day_of_week_number and Hour).
Feature -3: Prepare feature by grouping clicks by the combination of (Ip, App, Operating System, Day_of_week_number and Hour).
Feature -4: Prepare feature by grouping clicks by the combination of (App, Day_of_week_number and Hour).
Feature -5: Prepare feature by grouping clicks by the combination of (Ip, App, Device and Operating System).
Feature -6: Prepare feature by grouping clicks by the combination of (Ip, Device and Operating System).

For defining and grouping different features, group by and join functions are written to select different groups of data to analyze. Next, we split/train the data set and define the pipeline with the values to both the Decision Tree Classifier and Random Forest Classifier models. In order to tune our models, we set the parameters (maxDepth, maxBins) for paramGrid builder and used *TrainValidationSplit* to tune our models with 80% of training data and 20% for validation. We run the data in Databricks with the sampled 1GB data set in order to write and test the Spark code. We also calculate the confusion matrix and RMSE (Root Mean Square Error) to measure the accuracy.

5.1 Instance Hardware Specifications

For the original data set (7GB), Amazon AWS EMR instances are adopted with the number of different instances, called node in Hadoop and Spark. The Spark code is executed with the 7GB data set in 2 clusters, which is built and tested in Databricks cluster environment

Each node has the hardware specification as: CPU: 8 vCore, memory: 15 GiB, storage: 80 SSD GB

And the node has the Big Data software installed as: Hive 2.3.2, Spark 2.3.0

Besides, we have 2 Hadoop Spark clusters composed of 5 and 10 nodes as *m3.xlarge* each in order to measure the performance and to observe how the linear scalability is achieved practically:

5.2 Result Matrix and Chart

Table 2 shows the execution result of the models in 2 clusters. The pecision in Random Forest algorithm is much better than Decision Tree model: 0.878 vs 0.593.

Table 2. Comparison between two classifier models in Spark ML

Model	#Nodes	Precision	Recall	RMSE
Decision Tree	5	0.593	0.454	0.126
Decision Tree	10	0.593	0.454	0.126
Random Forest	5	0.878	0.495	0.103
Random Forest	10	0.878	0.495	0.103

But, it is worse than the traditional models as the Spark codes do not have enough data engineering to improve the accuracy as shown in **Table 1**.

6. Conclusion

We have tested more than 20 experiments on the traditional and big data systems in order to conduct our final results, which are shown at **Table 3**.

Table 3. Comparison of All Four Models

	2-Class Decision Jungle	2-Class Decision Jungle Forrest	Decision Tree	Random Forest
AUC	0.935	0.997	0.815	0.746
Precision	1.000	0.992	0.822	0.878
Recall	0.001	0.902	0.633	0.495
Real Time	2hrs	2-3hrs	22 mins	50mins

Table 4. Confution Matrix of All Four Models

	2-Class Decision Jungle	2-Class Decision Jungle Forrest	Decision Tree	Random Forest
TP	35	47,199	86,683	67,726
FP	0	377	18,727	9,408
TN	52,306	406,228	711,2961	712,2280
FN	406,605	5,142	50,074	69,031
Real Time	2hrs	2-3hrs	22 mins	50mins

In conclusion, our experiments have shown an acceptable precision rate to predict who will download or not-download an app after clicking an Ad. The Two-Class Decision Forest in the traditional systems and Random Forrest in Big Data have the better accuracy.

It is also observed that which IP is fraudulent and which is not. Therefore, companies in China could better target their audiences and implement more efficient marketing plans. It will then avoid fraudulent practices from illegitimate accounts and spend the marketing expenses more efficiently. Besides, it is shown that Big Data systems can be adopted to store and process 7 GB data set for predictive analysis, which cannot be easy – even impossible - using the traditional systems. It also shows that the Big Data systems are 7 – 9 times faster with the 10 nodes than the traditional systems. However, it will require more detailed data engineering processes to have the better accuracy.

References

[1] Feng Dong1, Haoyu Wang, Li Li, Yao Guo, Tegawende F. Bissyande4, Tianming Liu, Guoai Xu, Jacques Klein, FraudDroid: Automated Ad Fraud Detection for Android Apps, Conference’17, July 2017, Washington, DC, USA

[2]Priyanka Purushu, Niklas Melcher, Bhagyashree Bhagwat, Jongwook Woo, "Predictive Analysis of Financial Fraud Detection using Azure and Spark ML", Asia Pacific Journal of Information Systems (APJIS), VOL.28, NO.4, December 2018, pp308~319

[3] Brad Miller, Paul Pearce, and Chris Grier, Christian Kreibich, Vern Paxson, What’s Clicking What? Techniques and Innovations of Today’s Clickbots, DMIVA 2011, LNCS 6739, pp. 164–183, 2011

[4] Woo , Jongwook, & Xu , Yuhang (July 18-21, 2011), Market Basket Analysis Algorithm with Map/Reduce of Cloud Computing, The 2011 international Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2011), Las Vegas.

[5] Azure Machine Learning Studio, https://studio.azureml.net/