Juri Kren

Performance of random forest and logistic regression classifiers on the BankSim dataset using PySpark

Proposal

Introduction and dataset overview

Fraud and fraud detection is a significant issue both globally and in the UK. According to Nilson Report [1], card fraud losses were $28.6bn globally in 2020, with total losses in the upcoming decade forecast at $408.5bn. Furthermore, according to UK Finance, more than £1.3bn was stolen from the banking and finance industry through fraud last year [2] while estimates from the Telephone-operated Crime Survey for England and Wales [3] revealed that the number of fraud offences jumped by 25% in the year to March 2022 compared with the year to March 2020.

Lopez-Rojas and Axelsson [4] claimed that in order to investigate, develop, test and improve fraud detection techniques, one needs detailed information about the domain and its specific problems. However, Zareapoor and Shamsolmoali [5] argued that the lack of available real data becomes a serious obstacle to research as banks and financial institutions are not prepared to disclose their clients' transaction data for privacy reasons. Therefore, for my study I chose a Kaggle dataset generated by BankSim [6]. According to its authors Lopez-Rojas and Axelsson [4], "BankSim is an agent-based simulator of bank payments based on a sample of aggregated transactional data provided by a bank in Spain. The main purpose of BankSim is the generation of synthetic data that can be used for fraud detection research".

There are eight categorical features in the dataset: step, customer, age, gender, zipcodeOrigin, merchant, zipMerchant and merchandise category, one numerical feature (transaction amount) and one binary target variable (fraud). In total, there are 594643 observations generated as a result of randomised simulation: 7200 fraud and 587443 non-fraud transactions. The share of fraud transactions is only 1.2% of total transactions, meaning that the dataset is highly imbalanced. Lopez-Rojas and Axelsson [4] claim that as the dataset contains no personal information or disclosure of legal and private customer transactions, it can be shared by anyone.

Table 1 below shows the number of distinct categories for each categorical variable.

Table 1. Number of distinct categories for categorical variables

step	customer	age	gender	zipcodeOrigin	merchant	ZipMerchant	merchandise category	fraud
180	4112	8	4	1	50	1	15	2

Table 2 below shows statistics for the transaction amount variable, the only numerical feature. It is substantially positively skewed, with the maximum value markedly exceeding the mean value.

Table 2. Statistics for the transaction amount variable (in euros)

Mean	Standard deviation	Minimum	Maximum	Skewness
37.9	111.4	0	8330.0	32.4

Operational definitions

1. A step represents a day of commercial activity, so there are 180 distinct steps in total corresponding to around six months of commercial activity.
2. There are eight distinct age categories: <=18, 19-25, 26-35, 36-45, 46-55, 56-65, >65 and Unknown.
3. There are four distinct gender categories: Female, Male, Enterprise and Unknown.

4. zipcodeOrigin and ZipMerchant represent zip code locations of origin and merchant, respectively. There is only one distinct value per each category, so I remove both of them immediately.
5. There are two distinct categories for the fraud (target) variable: 0 (no fraud) and 1 (fraud).

Aims and objectives

The aims of my project were: to evaluate the performance of the random forest and logistic regression classifiers on the BankSim dataset using PySpark, and to compare the results with each other as well as with past fraud-related research done in Apache Spark. To achieve the aims, I:

1. Collected the dataset from Kaggle, stored it on Hadoop's HDFS, transformed it to a PySpark dataframe and cleaned it to make it neat and suitable for further analysis and models.
2. Analysed the data with help of functions in PySpark.
3. Conducted Chi-square and correlation tests for choosing appropriate features in my models.
4. Prepared classifiers, fit them and evaluated their performance with help of Spark MLlib.

Past research

A number of studies discussed various machine learning techniques for fraud detection (see Minastireanu and Mesnita [7], Kiwanuka [8], Phua et al. [9] and Yousefi, Garibay and Alaghband [10] for research review). Only few (in particular, Ananthu S, Nithin Sethumadhavan and Hari Narayanan AG [11] and Armel and Zaidouni [12]) looked into fraud detection using Apache Spark but with no indication of PySpark use. In both [11] and [12] the authors compared the performance of different algorithms: decision tree, logistic regression and random forest classifiers in the first one and simple anomaly detection, decision tree, random forest algorithm and naive Bayes classifiers in the second one. While in the first study they used transaction data of European credit cardholders, in the second one they used randomly generated data, with accuracy the only common performance metric for both datasets. In the first case all the algorithms scored the same accuracy of 99.9%, while in the second one the random forest algorithm outperformed other classifiers with the accuracy of 98.2%.

Performance metrics

However, as argued by Czakon [13], accuracy is not an appropriate metric in case of imbalanced datasets as it measures a share of correctly classified both positive and negative cases in the dataset. That means that if the dataset is markedly imbalanced, one can obtain a very high accuracy score by simply predicting that all observations are part of the majority class. Therefore, for the comparison of classifiers with each other I also used another metric: area under the precision-recall (PR) curve which, according to Davis and Goadrich [14], provides a more informative picture of an algorithm's performance in case of highly skewed datasets. On the graph, recall values (measure the share of correctly labelled positive cases) are shown on the x-axis and precision values (measure the share of positively classified cases that are truly positive) on the y-axis.

Parameters of classifiers and train/test split

In my study I compared the performance of two classifiers: random forest and logistic regression. I split the dataset into random train and test sets at 70%/30% ratio with a seed value of 50. Furthermore, for consistency I left almost all classifiers' parameters at default values. The only three parameters I set manually were: the label column (fraud) – for both classifiers, and the seed value of 50 as well as the maximum number of bins at 4200 – for the random forest classifier as the number of distinct categories for the customer feature is 4112. For both classifiers I used Binary Classification Evaluator for obtaining performance metrics on the area under the PR curve and Multiclass Classification Evaluator for obtaining accuracy metrics.

In addition, with the logistic regression classifier trained on the oversampled train dataframe after one-hot encoding, I used grid search for choosing the best two parameters from a small grid. In particular, I tried the maximum number of iterations at 50 and 100 and the regularisation parameter at 0 and 0.1.

Features’ association with fraud

For choosing appropriate features in my models, I tested a null hypothesis that categorical features were not associated with fraud and used p-values of Chi-square test to determine that. According to Hayes [15], a chi-square (χ2) statistic is a measure of dissimilarity between the observed and expected frequencies of the outcomes of a set of events or categorical variables. Furthermore, I calculated the coefficient of point biserial correlation (equivalent to Pearson correlation – see DeJesus [16]) to determine the strength of association between the transaction amount variable and the fraud variable.

Techniques used

Before conducting the Chi-square test and evaluating the classifiers’ performance, I converted the values of all categorical features to numbers with StringIndexer. In addition, as the dataset was highly imbalanced, I did both oversampling (by populating fraud samples) and undersampling (by reducing non-fraud samples) on the train set. I did it using the ratio of non-fraud rows to fraud rows in the set, in line with Wan [17] approach.

As the logistic regression classifier gave undue preference to higher labels of ordinally encoded variables, potentially leading to bias and poorer model performance, I encoded the indexed features using one-hot encoding. One-hot encoding is an algorithm where each ordinally encoded categorical variable is converted into binary vectors [18]. Following one-hot encoding, I evaluated the classifiers’ performance again.

To combine multiple algorithms into a single workflow, I built pipelines. Each pipeline included a transformer – a vector assembler that took features as input columns and returned a single vector of features as an output column, and an estimator – a relevant classifier model [19].

Summary and Conclusions

I think that I have achieved the aims of my project (to evaluate the performance of the random forest and logistic regression classifiers on the BankSim dataset using PySpark, and to compare the results with each other as well as with past fraud-related research done in Apache Spark), and all the objectives.

I set the null hypothesis that categorical features were not associated with fraud and used Chi-square test to determine that. The p-values showed that only the step feature was almost certainly not associated with fraud. In addition, the point biserial correlation coefficient between the numerical transaction amount variable and the fraud variable indicated moderate positive correlation of 0.49. Therefore, after tests I only removed the step variable from the dataframe.

With the numerical indexing of categorical features, the logistic regression classifier underperformed the random forest classifier on the oversampled train dataframe, in particular in the area under PR metric (68.7% vs 77.8%, respectively). Unlike the random forest classifier, the logistic regression classifier gave unreasonable preference to higher labels of ordinally encoded variables, leading to bias and poorer model performance.

However, after one-hot encoding the logistic regression classifier outperformed the random forest classifier both on the oversampled and the undersampled train dataframes, especially in the area under PR metric. While the logistic regression classifier's performance improved on the oversampled train dataframe, the one of the random forest classifier worsened on both dataframes as one-hot encoding led to a marked increase in dimensionality because of a large number of categories for categorical features.

Furthermore, after one-hot encoding both classifiers registered lower scores in the area under PR metric when fit on the undersampled train dataframes compared to ones fit on the oversampled train dataframes. The outcome is not surprising as undersampling probably caused the loss of valuable information about the majority class, with the chosen sample not accurately representing the population.

Both the random forest classifier after numerical indexing of categorical features and the logistic regression classifier after one-hot encoding scored the best accuracies of around 99%, something expected for a highly imbalanced dataset and similar to past fraud-related research results in Apache Spark. More importantly, the random forest classifier's best score in the area under PR metric came slightly above the logistic regression classifier's one (77.8% vs 76.5%, respectively), with both classifiers' scores indicating their decent ability to identify fraud.

For the logistic regression classifier trained on the oversampled train dataframe after one-hot encoding, I used grid search to choose the best two parameters. I tried the maximum number of iterations at 50 and 100 and the regularisation parameter at 0 and 0.1. The results showed a small improvement in the area under PR metric (to 76.6% from 76.5%) with the maximum number of iterations at 50 compared to the default maximum number of iterations at 100.

While I used a dataset with less than 600,000 observations, the same modelling principles would apply to much larger datasets.

My project has limitations. The first one is that because of time constraints I used only two classifiers: random forest and logistic regression and did not include others, like a decision tree, a naive Bayes or a linear support vector machine. Secondly, I used grid search for choosing the best model parameters only once – for a logistic regression model – and with a small grid, as otherwise the model's fitting time would increase markedly. It would be interesting to conduct large-scale research on the topic, addressing both the above limitations.

References

1. Nilson Report (2021). Issue 1209 [Online]. Available from: https://nilsonreport.com/upload/content_promo/NilsonReport_Issue1209.pdf [11 September 2022].
2. Ramsey, T. (2022) Fraud losses continue to climb with over £1.3bn stolen in 2021. Which? [Online]. Available from: https://www.which.co.uk/news/article/fraud-losses-continue-to-climb-with-over-1.3bn-stolen-in-2021-ayXMX4b9mgQW [11 September 2022].
3. Office for National Statistics (2022). Crime in England and Wales: year ending March 2022 [Online]. Available from: https://www.ons.gov.uk/peoplepopulationandcommunity/crimeandjustice/bulletins/crimeinenglandandwales/yearendingmarch2022 [11 September 2022].
4. Lopez-Rojas, E.A., Axelsson, S. (2014). Banksim: A bank payments simulator for fraud detection research [Online]. Inproceedings 26th European Modeling and Simulation Symposium, EMSS 2014, Bordeaux, France, pp. 144–152, Dime University of Genoa, 2014, ISBN: 9788897999324. Available from: https://www.researchgate.net/publication/265736405_BankSim_A_Bank_Payment_Simulation_for_Fraud_Detection_Research [11 September 2022].
5. Zareapoor, M., Shamsolmoali, P. (2015). Application of Credit Card Fraud Detection: Based on Bagging Ensemble Classifier [Online]. Available from: https://www.researchgate.net/publication/277949914_Application_of_Credit_Card_Fraud_Detection_Based_on_Bagging_Ensemble_Classifier [11 September 2022].
6. Lopez-Rojas, E.A. Synthetic data from a financial payment system [Online]. Available from: https://www.kaggle.com/datasets/ealaxi/banksim1 [11 September 2022].
7. Minastireanu, E.A., Mesnita, G. (2019). An Analysis of the Most Used Machine Learning Algorithms for Online Fraud Detection [Online]. Available from: https://www.researchgate.net/publication/332268831_An_Analysis_of_the_Most_Used_Machine_Learning_Algorithms_for_Online_Fraud_Detection [11 September 2022].
8. Kiwanuka, F.N. (2018). A state-of-the-art review of machine learning techniques for fraud detection research [Online]. Available from: https://www.researchgate.net/publication/326565861_A_state-of-the-art_review_of_machine_learning_techniques_for_fraud_detection_research [11 September 2022].
9. Phua, C. et al. (2010). A Comprehensive Survey of Data Mining-based Fraud Detection Research [Online]. Available from: https://www.researchgate.net/publication/46887451_A_Comprehensive_Survey_of_Data_Mining-based_Fraud_Detection_Research [11 September 2022].
10. Yousefi, N., Garibay, I., Alaghband, M. (2019). A Comprehensive Survey on Machine Learning Techniques and User Authentication Approaches for Credit Card Fraud Detection [Online]. Available from: https://www.researchgate.net/publication/337781401_A_Comprehensive_Survey_on_Machine_Learning_Techniques_and_User_Authentication_Approaches_for_Credit_Card_Fraud_Detection [11 September 2022].
11. Ananthu S, Nithin Sethumadhavan, Hari Narayanan AG (2021). Credit Card Fraud Detection using Apache Spark Analysis. In 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI) (1-66544-687-0, 978-1-66544-687-7), p. 998.
12. Armel, A., Zaidouni, D. (2019). Fraud Detection Using Apache Spark. Published in: 2019 5th International Conference on Optimization and Applications (ICOA). Publisher: IEEE. Electronic ISBN:978-1-7281-1482-8.
13. Czakon, J. (2022). F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should You Choose? [Online]. Available from: https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc [11 September 2022].
14. Davis, J., Goadrich, M. (2006). The Relationship Between Precision-Recall and ROC Curves [Online]. Available from: https://www.biostat.wisc.edu/~page/rocpr.pdf [11 September 2022].
15. Hayes, A. (2022). Chi-Square (χ2) Statistic [Online]. Available from: https://www.investopedia.com/terms/c/chi-square-statistic.asp [11 September 2022].
16. DeJesus, J. (2019). Point Biserial Correlation with Python [Online]. Available from: https://towardsdatascience.com/point-biserial-correlation-with-python-f7cd591bd3b1 [11 September 2022].

17. Wan, J. (2020). Oversampling and Undersampling with PySpark [Online]. Available from: https://medium.com/@junwan01/oversampling-and-undersampling-with-pyspark-5dbc25cdf253 [11 September 2022].
18. GeeksforGeeks (2022). ML | One Hot Encoding to treat Categorical data parameters [Online]. Available from: https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/ [11 September 2022].
19. Drabas, T., Lee, D. (2017). Learning PySpark. Birmingham - Mumbai: Packt Publishing.