UNIVERSITÀ DI PISA

DEPARTMENT OF COMPUTER SCIENCE
Data Science and Business Informatics

Distributed Data Analysis and Mining

Bank Account Fraud Detection

Simone Di Luna 544322

Academic year 2022/2023

Contents

1 Introduction

The Bank Account Fraud (BAF) is a suite of six artificial datasets related to bank account opening applications. They were created by Jesus et al. (2022) and published at NeurIPS 2022. The analyses was conducted on the `Base.csv` dataset, since it is the one most similar to the original data used to train the generator. Indeed, the dataset is composed of synthetic instances generated using a CTGA, a collection of deep learning based synthetic data generators, which can learn from real data and generate synthetic records with high fidelity.

Each record represents an application to open a bank credit line. This analysis aimed to find common characteristics of groups of clients and thus build models that would help distinguish between fraudulent and legitimate applications. The dataset does not contain rejected requests.

2 Exploratory Analysis

The dataset is composed of 1 000 000 records and 32 columns, both qualitative and quantitative. Specifically, the attributes have the following data types:

- 8 binary variables: `bool_fraud`, `email_is_free`, `phone_home_valid`, `phone_mobile_valid`, `has_other_cards`, `foreign_request`, `keep_alive_session`, `device_fraud_count`;
- 5 nominal variables: `payment_type`, `employment_status`, `housing_status`, `source`, `device_os`;
- 19 numerical, both discrete and continuous variables: `prev_address_months_count`, `current_address_months_count`, `bank_months_count`, `intended_balcon_amount`, `zip_count_4w`, `velocity_6h`, `proposed_credit_limit`, `days_since_request`, `customer_age`, `date_of_birth_distinct_emails_4w`, `credit_risk_score`, `income`, `name_email_similarity`, `velocity_24h`, `device_distinct_emails_8w`, `month`, `velocity_4w`, `bank_branch_count_8w`, `session_length_in_minutes`.

The target variable is `bool_fraud`. It assumes the value 1 when the credit request is fraudulent, and 0 otherwise. The problem is that the dataset is heavily unbalanced, only about 1.1% of the applications are fraudulent, which complicates our analysis.

Initially, we performed a stratified random sampling (80/20) in order to split data between train and test sets, maintaining the original proportions between fraudulent and legitimate applications.

We then calculated some basic quantitative statistics to get a clue on which variables to investigate in more detail. We immediately noticed that some attributes were affected by missing values indicated by the value -1. The variables in question are as follows: `prev_address_months_count`, `current_address_months_count`, `bank_months_count`, `session_length_in_minutes`, `device_distinct_emails_8w`.

The `prev_address_months_count` indicates the number of months the applicant lived in the old house. This variable was characterized by a considerable amount of missing values (about 72%), hence we removed it from the dataset.

The `current_address_months_count` represents the number of months elapsed since the applicant has been living in the new residence. Since the percentage of missing values was negligible (less than 1%) and the distribution of the variable right-skewed, we decided to fill them with the median value.

The `bank_months_count` attribute indicate how old is the customer previous account (if held) in months. Since the missing values represents the mode for this variable (24.34%) we drop it from the dataset.

The `session_length_in_minutes` measures the length (in minutes) of the user's session in the banking website when the application was made. The same considerations we made for the `current_address_months_count` variable applied and thus we used the median to fill the nulls.

The `device_distinct_emails_8w` counts the number of distinct emails used by the user on the bank's website over the last eight weeks. The missing values was negligible (only 290), therefore we decided to fill them with the mode of the distribution because its variance is very low (96.81% of the records have the same value).

Looking at the other attributes, there are other facts that should be noted. The `days_since_request` (which is self-explanatory) is almost completely positively skewed, indicating that these are all new accounts. The (annual) `income` variable was discretized in in decile form by the original authors. Its distribution is bimodal, but slightly unbalanced towards the high-income customers. Analyzing the binary variable `source`, it emerged that about 99.29% of the applications where collected from the internet (the rest from teleapp). The other categorical variables were originally anonymized and their values encoded in acronyms.

Generally speaking, the variables are not high correlated with each other both considering linear and non linear correlation coefficients (Pearson, Spearman). Some interesting exceptions are represented by `velocity_24h, moth, velocity_4w`, sharing a large negative correlation. This may indicate seasonality of the demand in the business area considered or a decrease in turnover. The `credit_risk_score` and `proposed_credit_limit` have a positive correlation of 0.606, indicating that (current) customers making higher credit applications are assigned a higher risk score, probably due to greater financial exposure from the bank.

We also compared categorical and binary attributes with the target one by using the chi-squared test, a tool dedicated to measure the relationship between these types of variables. Specifically, we tested (for each pair) the null hypothesis that the variables are statistically independent. Using a 1% significance level, the test failed for almost all the pairs of attributes, hence highlighting some sort of correlations between them. The only two variables for which the null hypothesis cannot be rejected due to high p-values are the `payment_type_AD` and the `payment_type_AB`.

An interesting correlation is that between the target variable and the age of the customers. In fact, Table 1 shows that middle-aged people (30 to 60 years old) are almost three times more likely to commit fraud than other ages.

Table 1. Risk of fraud by age group.

Age (decades)	10	20	30	40	50	60	70	80	90
Fraudulent applications	58	956	2056	2287	2248	919	229	56	3

In the next sections we will use again the chi-squared test in order to perform the feature selection analysis.

3 Clustering

Cluster analysis was a difficult task with this dataset. We tested different types of variable aggregations: quantitative only, both quantitative and qualitative, PCA-transformed variables; and subsets of these. However, none of them produced satisfactory results.

At first, we encoded the nominal variables with numeric indices and then disaggregated their values by means of the *one hot encoder* encoder. Therefore, for each categorical variable with n attributes, we were able to create $n - 1$ binary variables. Then we aggregated and standardized all binary and numeric variables producing a vector column having a row cardinality of 44.

To perform a sort of feature selection, we conducted a preliminary analysis by running the simple k-means algorithm using first only the strictly numerical variables and then doing the same with the vector containing all variables. For both features, we evaluated all clusterings from two to ten clusters and compared them with both the sum of squared errors (SSE) and the silhouette score.

The clusters produced using only the numerical variables scored higher than the others. This is probably due to the metric used to calculate the distances, namely the Euclidean distance. PySpark also offers another metric, the cosine distance, however, it is closely related to the Euclidean distance. Our tests produced very similar results. This type of metric cannot handle categorical and continuous data at the same time.

The best result, in terms of silhouette score, obtained using only the 17 numerical variables, was the one with 7 clusters (silhouette = 0.2279). On the other hand, the analysis using both the numeric and binary variables maximized the silhouette score (0.1177) with the clustering having a cardinality of 6.

It is not possible to use SSE to compare clusterings generated using a different set of attributes, as SSE is directly proportional to the number of dimensions. Nevertheless, we can use this measure to try to select the optimal number of clusters. Unfortunately, the SSE for both sets of attributes decreases at approximately a constant rate. Moreover, also looking at the centroid's distribution in the space we was not able to extract helpful insights

3.1 Principal components analysis

In order to find a better partitioning of the data, we also decided to perform a clustering analysis in the latent space using Principal Component Analysis (PCA).

The first attempt was to apply PCA to all 44 attributes. We immediately noticed that the components taken individually could only explain a very small percentage of the total variance. In particular, the first component, the best one, could only explain just over 7% of the variance. Because of this, a scatter plot produced in a two-dimensional space, obtained by extracting the first two components, is not able to reproduce spatially the information contained in the original dataset. Overall, to be able to explain a significant percentage of variance (say >90%), we need at least 30 components.

Using plain k-means and evaluating all clusterings produced by setting the parameter k ranging from 2 to 10, we obtained almost the same results as illustrated in the previous section, where we used all 44 normalized attributes. However, by plotting the centroids in space, we noticed that regardless of the value of k, the last 9 dimensions were almost perfectly aligned. For this reason it was considered appropriate to remove the last 9 components.

We then repeated the analysis using the best 35 components. This led to the identification of a model that produces very similar results to those we discussed in the previous section with the vector containing only numerical attributes. In particular, the best clustering produced is composed of 5 clusters and has a silhouette score of 0.2363.

Finally, we tested the following attributes configuration. Along with the 17 normalized numeric variables, we took the 27 extracted by applying PCA to the binary variables only. With this features vector, the best model was obtained using 6 clusters and produced a silhouette of 0.1177. The same score we got with the 44 attributes without PCA. Again, we noticed that the last four components had no discriminating power (indeed, together they could not explain even 1% of the total variance), so we removed them and repeated the analysis, however without any improvement.

In conclusion, if we were to look solely at the silhouette score, we would have to choose the best 35 components produced with PCA as the reference feature vector. However, in this latent space of 35 dimensions, we are unable to attribute meaning to the features we use. There is also to keep in mind that from a theoretical point of view, a higher score could be attributed not so much to a better quality of the selected attributes, but simply to the fact that the number of attributes is smaller (curse of dimensionality). By applying, on the other hand, PCA only to binary variables we are able to mitigate the negative effects due to the lack of a metric suitable for working with mixed data, without losing too much from a semantic point of view.

3.2 Comparing different clustering algorithms

The starting point for this analysis is the vector of 40 attributes we discussed in the previous section. The goal is to compare the results obtained using different clustering algorithms.

For the k-means, we have already ascertained that the optimal value of the k-parameter is 6. However, to speed up the computational process, we set the other parameters of the model using lower values than the default ones. Therefore, we fine-tuned the model by increasing the number of iterations to 50 and the number of initializations through the k-means|| to 10. However, we did not obtain any performance improvements, an indication that the previous model had reached convergence anyway.

For this, as with the other models discussed below, we also tried and analyzed the clustering produced using the cosine distance, however, the results obtained were always worse than those produced by the Euclidean distance. The second clustering algorithm we used is bisecting k-means. Again, we did a sort of hyper-parameter tuning, going to evaluate the performance on all clusterings having a cardinality ranging from 2 to 10, and this was done both using the Euclidean and cosine distance. The maximum silhouette score it achieved is 0.095 .

Finally, the Gaussian mixture model also proved to be unable to produce satisfactory clustering. Even, it returned negative values of both silhouette and loglikelihood scores. In particular, the latter finding can be interpreted as a failure of the assumptions on which the model is based, namely that the points in the dataset were not generated from a set of Gaussian distributions.

Table 2. Distribution of the target variable inside the clusters.

Cluster	0	1	2	3	4	5
Cardinality	377204	20792	9489	366856	20273	5675
Cardinality (%)	47.13	2.6	1.19	45.84	2.53	0.71
Fraud count	2318	82	241	5573	504	94
Fraud (%)	0.615	0.394	2.54	1.656	2.486	1.519

3.3 Clustering characterization

In the previous section, we have seen that neither the bisection k-means nor the Gaussian mixture model are able to produce better results than the k-means. The optimal clustering produced by the latter consists of 6 clusters. Let us now analyze the composition of these clusters.

As shown in Table 2, the clusters generated are very unbalanced. Most of the records (about 93%) are distributed between clusters 0 (47.13%) and 3 (45.84%). The remaining ones go to form three small satellite clusters.

The target variable is distributed mainly in clusters 3 and 0. However, what is most revealing is the percentage of fraudulent applications calculated with respect to the cardinality of the cluster. In particular, we can see that almost half of the records are located within clusters 0 and 1. In these two clusters, the percentage of fraudulent applications is the absolute lowest, which means that any classifier trained within any of the two clusters will predict the positive class with a frequency close to 0. The clusters within which fraudulent records are most likely to be found are 2 and 4, however summed these two clusters account for just 4 percent of the total records in the dataset. In section 4 we will try to train different models for each of the six clusters.

We evaluated the distribution within clusters for the other variables as well. For the quantitative ones, in particular, we decided to discretize them into 5 buckets using the quantiles of the distribution. In general, no particular pattern seems to emerge, so with the clustering we were able to create it is difficult to adequately profile customers.

4 Classification

Before embarking on the predictive analysis, we decided to perform as an intermediate step a feature selection. We then trained and performed a hyperparameter tuning of three different classifiers (random forest, support vector machine and neural network) on the unbalanced train set and then compared the results obtained. Given the poor results, we tried to improve our analysis by exploiting two different strategies: oversampling and intra-cluster classification.

4.1 Feature selection

As a starting point, we selected the vector of 44 standardized features containing both numeric and binary variables and augmented it by also appending the variable of labels produced by the k-means algorithm discussed in the previous section.

We then performed a feature selection analysis by combining an embedded method, such as the feature importances produced by the random forest classifier, with the scores obtained from the univariate attribute analysis.

Specifically, we trained the classifier by constructing an ensemble of 100 trees, each with a maximum depth set at 10. We then extracted the feature importances from the model and used them to filter out all variables with a score of less than 1%.

The univariate analysis was instead divided into two parts by evaluating the correlation of the attributes with the target variable according to their data type. Indeed, the correct method for assessing the existing relationship between two categorical variables is the chi2-test. On the other hand to study the existing relationship between numerical and categorical variables, it is necessary to use the F-statistic. We therefore performed a chi2-test for each possible combination of target and binary variables and an F-test for each combination of target and numeric variable.

The arbitrary decision rule we followed is as follows: eliminate all variables that simultaneously have a feature importance smaller than 1% and are in the last ten positions in the ranking of the scores obtained in the univariate analysis. By applying this strategy, we came to eliminate 10 of the 27 binary variables we had. In particular, these are some of the variables we had previously obtained by applying the one hot encoder on nominal attributes.

4.2 Unbalanced classification

In this section, we illustrate the analysis conducted on the unbalanced dataset without adopting any particular technique other than hyperparameter tuning.

In particular, we trained and tuned 3 models: support vector machine, random forest and neural network. Due to the computational complexity of the research, it was decided to modify only a few parameters.

Support Vector Machine. We trained the model using cross validation on 5 folds. The search was carried out on the regularization parameter only, considering the following values: 0, 1, 5, 15. As the dataset is heavily unbalanced, we considered it appropriate to use the air under the Precision-Recall curve as the optimal model selection metric. In the end, the best model turned out to be the one with the regularization parameter set to 0.

Random Forest For this model, it was decided to only make a search for the number of base classifiers to be built in the ensemble, setting a maximum tree depth of 15. The following three values were considered: 50, 100, 150. The number of folds used is 5. The model that obtained a higher average score was found to be the one with 150 base classifiers.

Multilayer Perceptron. A cross validation was carried out on 3 folds, taking into account the learning rate and the type of the neural network solver. The first parameter was searched for between the following values: 0.01, 0.3, 0.5; while for the second, l-bfgs and gd were considered. The structure used includes a hidden layer of 10 neurons. In the end, the optimal model was the one using the l-bfgs solver and a learning rate of 0.01.

At the end of the cross-validation, we trained again the classifiers on the entire train set using the optimal parameters found. For the multi-layer perceptron, we also incremented the number of iterations. Then we evaluated and compared them on the test set. However, despite hyper-parameter tuning, all three models performed extremely poorly. In particular, the support vector machine and the random forest behaved as dummy classifiers predicting only the negative class. The multi-layer perceptron, on the other hand, did slightly better, predicting the positive class 64 times out of 199711, with 33 true positives and 31 false positives. Given these poor results, it was decided here not to comment further on the metrics obtained.

4.3 Oversampling

In this section, an attempt was made to compensate for the unbalanced train set with some strategy. Since PySpark does not offer a distributed version of the KNN on which to then build an unbalanced algorithm, it was decided to opt for simple oversampling. However, instead of

duplicating records of the positive class by choosing them at random within the dataset, it was deemed more appropriate to try to duplicate those that are more difficult to classify. Specifically, the strategy adopted is to train different models iteratively within a loop, while evaluating them on the train set itself. Therefore, at each iteration, we train a classifier, evaluate it, and duplicate misclassified positive records. The loop stops either when the maximum number of iterations is reached (set to 20) or when the cardinality of the positive class exceeds 400000.

In implementing this strategy, we adopted a plain logistic regressor. The loop stopped when the second condition was reached. At the end of the procedure, we obtained a new train set consisting of 571704 positive records (41.94

We then trained two models on this dataset: random forest and multilayer perceptron, this time without performing new cross-validation, but using the old optimal parameters.

The random forest classifier continued to perform poorly, but at least it predicts the positive class with some positive frequency (1% of the time, compared to 0% previously). However, the number of false positives is about twenty times higher than the number of true positives (precision 5.22%). And the same is true for false negatives (recall 5.19%). F-measure, thus, is 5.2%.

On the other hand, the multilayer perceptron predicts the positive class once in three times, however, with a very low precision of 1.78%. Overall, however, the number of true positives has increased and jointly that of false negatives has halved. This has led to an increase in recall to 52.8%.

Overall, however, we can say that this technique produced some positive results.

4.4 Intra-cluster classification

Finally, we decided to take advantage of what we learned during the unsupervised learning phase to try to increase the predictive power of our classifiers. Specifically, we went to train distinct classifiers within each cluster. The goal of this analysis was to try to exploit the clustering information to obtain better predictions. It should be remembered, however, that up to this point we have made use of this information anyway because, as discussed earlier, we had already included cluster labels within our features vector.

In order not to have redundant information, we first proceeded to remove the labels from our attribute set. Then we went to iteratively filter the subsets of the train set containing only the records of the single cluster considered in that iteration. Immediately after training, during the same iteration, we applied the same filter on the test set to evaluate the model only on the set of records labeled in the same way.

We conducted this analysis with both the random forest classifier and multilayer perception. However as evidenced by the plotted confusion matrices, this strategy proved to be unsuccessful. The two classifiers predict positive class with a frequency close to 0%, behaving in the same way as a dummy classifier.

References

Jesus, Sérgio, José Pombal, Duarte Alves, André Cruz, Pedro Saleiro, Rita P. Ribeiro, João Gama, and Pedro Bizarro (2022). *Turning the Tables: Biased, Imbalanced, Dynamic Tabular Datasets for ML Evaluation*. DOI: 10.48550/ARXIV.2211.13358.