Credit Card Fraud Detection

Machine Learning Project - Phase 4

Malak Gaballa
900201683
Data Science
The American University in Cairo
malakhgaballa@aucegypt.edu

Masa Tantawy
900201312
Data Science
The American University in Cairo
masatantawy@aucegypt.edu

Introduction

After choosing the best model in the previous phase, this model needs to be refined in order to best fit the data and yield the most accurate solution based on the evaluation metrics. The model concluded in phase 3 was the decision tree based on entropy; however, an important step that was missed was to check for overfitting. Given that all the testing and training of the models was based on the 75%-25% train-test split, no cross validation of any model using k-fold cross validation was implemented. Thus in this phase, it is crucial to check for overfitting before choosing the best model. After selecting the true final model, a detailed explanation of the steps that will be followed to refine the model will be provided. The utility app design and its architect will also be illustrated.

Model Choice

To check for overfitting, the top 3 models of the previous phase were implemented using k-fold cross validation on the entire dataset, unlike being split into training and testing sets like in phase 3. These 3 models were decision trees, multilayer artificial neural networks, and random forests. A 5-fold and 10-fold cross validation was used. The evaluation criteria used was the recall (of the fraudulent transactions) as explained in the previous phase due to the data's imbalance and importance of flagging a fraudulent transaction. The specificity was not of a major significance as it was extremely high (over 0.99) regardless of the model used as seen previously.

Table 1 shows each model's recall score for the 5-fold and 10-fold cross validation. Although the decision tree based on entropy had the highest recall score based on the 75%-25% train-test split, this result was inconsistent with cross-validation. The recall dropped from 0.85 to almost 0.81, indicating that the model was overfitting the data and similarly decision trees based on the gini index. On the other hand, the performance of neural networks and random forests based on entropy appeared to surpass decision trees as their recall values jumped to over 0.93 during cross validation. This is because both neural networks and decision trees are able to work with large amounts of data, such as in this case, and avoid overfitting. The best model is random forests based on gini or entropy with a recall value of 0.982 in the 10-fold cross validation. Because random forests take samples from the dataset, they yield better results.

Model	**Recall (5-Fold CV)**	**Recall (10-Fold CV)**
DT(Entropy)	0.806	0.8041
DT (Gini)	0.7837	0.7933
Neural Networks	0.9459	0.936
Random Forest (Entropy)	0.984	0.982
Random Forest (Gini)	0.984	0.982
Random Forest (Log Loss)	0.983	0.980

All the tables mentioned can be found at the appendix at the end of this document.

Model Design and Architecture

After picking random forests as the best model for the data, several enhancements would be made so that the model best fits the data and is able to accurately classify the fraudulent transactions thus gaining a high recall value. Before looking into the dataset for possible adjustments, some parameters in the *RandomForestClassifier()* will be looped over in order to find the parameter values that yield the highest recall. For instance, the parameter *n_estimators* which indicates the number of trees in the forest will take in multiple values in order to determine whether or not the number of trees affect the recall metric. Since the dataset provided is large (almost 2 million transactions) , the number of trees is expected to be large in order to fit the data. In addition, several features such as *max_depth*(depth of trees) ,*min_samples_split*(minimum samples required to split nodes) and *min_samples_leaf* (minimum samples in a leaf node) will also be tried in order to find the optimal values for each corresponding parameter. Another suggestion would be to assign weights to the features given that a feature importance attribute is provided after fitting the model and displays the effect of each feature on the model. These different combinations of parameters, including the criteria gini or entropy, will be tried to choose the best model.

The dataset itself could be modified in multiple ways to enhance classification. One way is to study whether variables, such as the city population adds to the tree or if eliminating it will affect the model's performance. Another is to examine the binning of the numerical features to be converted to categorical features in the tree. The current model depends on auto-binning done internally, so it is important to try different binning combinations for the numeric features in the dataset or to increase or decrease the default number of bins created to reach the optimal solution. Since the current model contains one-hot encoded features such as job and state, these features could be regrouped or classes could be unencoded back into one initial feature in order to simplify the forest structure and decrease the number of nodes. For the *job* feature for instance, the categories would be placed back into one column, making the feature consist of 9 unique values. As for the feature *state*, the 51 classes can be unencoded into 1 feature *state*, or they can be regrouped to reduce the number of classes without decreasing the model's performance. One suggestion is to divide the states according to geographical position or regions such as: South , West , Midwest and Northeast.

Another option is to try creating random forests based on the C4.5 algorithm instead of using the optimized very of CART which the *sckit-learn* package. Even though its performance may be worse than that of CART, the shape of each decision tree in the random forest will be simpler due to fewer nodes and more branches at each node. This is because each node will no longer yield a binary outcome. This will make the model easier to understand and implement.

Alternatively, it is pivotal for the model to learn from the incorrect classifications done during the training process. Hence, given the high imbalance of the dataset, the final model's error/loss function should endure higher cost/ penalty for misclassified fraudulent transactions compared to the legitimate transactions identified as fraud due to the importance of the matter as explained earlier. This will allow the model to fit the shape of the dataset and the inequality in the labels.

After implementing all of the motions mentioned above, the appropriate changes will be made to the dataset and the model parameters will be chosen, isolating the model that has the best performance and is able to correctly differentiate between fraudulent from non-fraudulent transactions the greatest number of times.

Utility App Design and Architecture

To facilitate the process of new transactions entry to the model and their classification, a simple GUI (graphical user interface) application will be created. This application, based on the *tkinter* python package, allows the user to enter the details of the transaction each in a required field, such as cardholder's city, latitude and longitude, and transaction amount. Then, this app will display whether the transaction is fraudulent or not and ask the user if the classification was correct or incorrect. Based on the choice of the user, the classification is updated and the transaction, with its details and the correct classification, will be recorded in the data to enhance the model's performance.

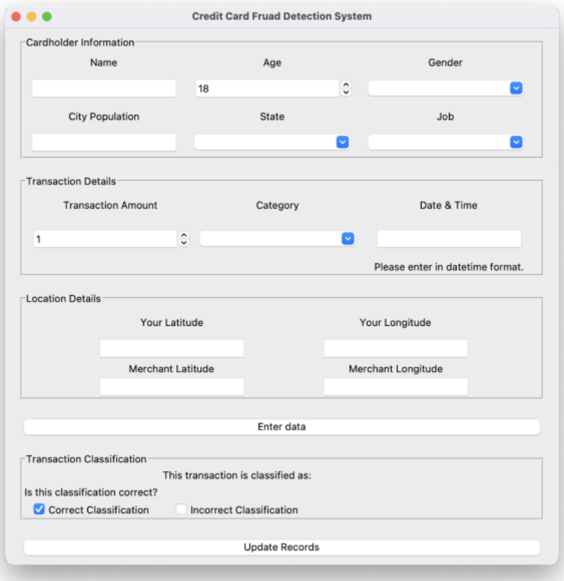
The application consists of 1 main window with multiple sections. The details of the transaction are collected in 3 sections: one for the cardholder information such as their age and job, one for the transaction details such as its amount and category, and a final section for the location details, latitude and longitude, of both the customer and the merchant. A button *enter data* following these 3 sections, and it displays the classification of the transaction according to the model's output in a 4th section. The user is then asked for feedback about this classification to update the data and store it with its correct class via checking one of 2 checkboxes indicating the classification's accuracy. These 2 checkboxes cannot be checked together; when one is checked, the other is not, and the default is that the classification is correct. The user interface is shown to the right.

The user input will be adjusted before being inputted to the model. These adjustments are similar to those performed on the initial dataset during preprocessing (phase 2), yet some of the inputted features require less preprocessing, such as *age* without the need to extract it from the date of birth, and *job* where the input is one of the 9 industries instead of entering a specific job that is assigned to an industry. Others need more preprocessing, such as the distance in km between the customer and the merchant which will be computed from their latitude and longitude coordinates. The transaction day, month, year, hour, and minute will all be extracted from the time entered by the user in datetime format into numeric features.

The categorical features gender, state, job and category will be encoded (label encoding for gender and one-hot for the remaining. As for the numeric features age , city population and transaction amount, they will be normalized using the same mean and variance used in the preprocessing (phase 2) which are stored separately. These details will constitute an instance with all the features required, and is now ready to be entered into the model for a classification decision. It is worth noting that the details of the preprocessing done on the data entered might differ depending on the outcome of the model design and architecture. For example, if the states were un-encoded, then one-hot encoding will be eliminated from the preprocessing.

Conclusion

For the final phase of the project, the final model will be complete as well as the app. The model will depend on the full dataset, consisting of all the instances without being divided into training and testing, and it will be in the format ready for model input (preprocessed).



Age: 18

Transaction Amount: 1

Please enter in datetime format.

This transaction is classified as:

Is this classification correct?

[✓ Correct Classification] Incorrect Classification

References

[1]
Credit Card Transactions Fraud Detection Dataset. *Kaggle*. Retrieved February 13, 2022 from https://www.kaggle.com/datasets/kartik2112/fraud-detection?resource=download