# 

# **ESOF 0151 - Large Scale Data Analytics**

# **Date Submitted: October 30th, 2020**

# **Project Progress Report: Stage 3**

# **Contributing Members:**

# **David Adese, Lucas Dillistone, Kevin Genereux**

**Materials**

To train and test our models, we will be leveraging the CPU and GPU provided by Google Colaboratory. The experimental analysis for the gradient boosting models will be conducted with an Intel (R) XEON E5-2660 CPU which contains 8 cores / 16 threads operating at a base frequency of 2.20 GHz and has 16 GB of memory. The experimental analysis for the Artificial Neural Network (ANN) will be conducted with an NVIDIA containing 2560 CUDA Cores that can perform 8.1 TFlops of single-precision point calculations and contains 16 GB of memory.

**Methods**

Dataset

The dataset has been acquired from Vesta’s real e-commerce transactions and was used for a previous Kaggle competition. The dataset consists of two tables: a transaction table (394 features) and identity table (41 features). The transaction table contains information about the transaction time and date, product code, transaction amount, and the address of the associated user. The identity table contains information about the associated user to each transaction, such as network information, browser, and operating system. Vesta has already partitioned the transaction dataset into training (590,541 rows of data) and testing (506,692 rows of data) files. While most features contain no missing values, some features have up to 96% of their data missing. The dataset consists of mostly numerical data but also has some categorical features. The dataset is highly imbalanced as the positive class (fraudulent transactions) accounts for only 3.5% of the dataset.

Some aspects of the data were not fully described due to the confidentiality involved in financial transactions. The table below describes in further detail the features in the transaction and identity files:

|  |  |  |
| --- | --- | --- |
| **Transaction Table** | | |
| **Features** | **Description** | **Type** |
| TransactionDT | Time delta from a given reference datetime. | numerical |
| TransactionAMT | Transaction payment amount in USD. | numerical |
| ProductCD | Product code, the product for each transaction. | categorial |
| Card 1 - Card 6 | Payment card information, ushc as card type, card category, issue bank, country, etc. | categorical |
| Addr | Address. | categorial |
| Dist | Distance between customer and product purchased. | numerical |
| P\_ and (R\_\_) email domain | Purchaser and Recipient Email Domain | categorial |
| C1-C14 | Counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked. | categorical |
| D1-D15 | Timedelta, such as days between previous transactions for the customer, etc. | categorical |
| M1-M9 | match, such as names on card and address, etc. | categorical |
| V1 - V339 | Vesta engineered rich features, including ranking, counting, and other entity relations. | mixed |

**Table 1:** lists the features and their characteristics in the transactions table.

|  |  |  |
| --- | --- | --- |
| **Identity Table** | | |
| **Features** | **Description** | **Type** |
| TransactionID | Transaction ID of the client. | numerical |
| ID1 - ID38 | Device rating, domain rating, proxy rating, login times. Lots of this information is concealed due to privacy. | numerical |
| DeviceType | Digital signature (browser, operating system) | categorical |
| DeviceInfo | Either desktop or mobile. | categorical |

**Table 2:** lists the features and their characteristics in the identity table.

Data Preprocessing

Our highly imbalanced dataset will be addressed by using the Synthetic Minority Over-sampling Techniques (SMOTE), the Adaptive Synthetic (ADASYN) algorithm and the NearMiss-3 algorithm . An imbalanced dataset is problematic as classifiers will have a tendency to misclassify fraudulent transactions because it is the minority class.

SMOTE is an oversampling approach that synthesizes new instances of the minority class between existing minority classes. The SMOTE algorithm we used creates new instances of the minority class by randomly selecting instances of the minority class. Once a minority class instance is selected, it then determines the instance with the closest Euclidean distance to it. This distance is then randomly multiplied with a rumber number between 0 to 1 and is added to the test object’s value to create a new sample.

The ADASYN algorithm, much like the SMOTE algorithm, oversamples the data by creating synthetic data to increase the number of samples in the set, but it improves on the SMOTE algorithm by adding random small values to the points to make it more realistic. Oversampling is usually preferred as no data is lost in the process, but there can be an issue of generalization so we will also be testing out some undersampling techniques.

Undersampling is a method of dealing with imbalanced data by reducing the amount of majority classes in the set. While undersampling does have the issue of removing data it can reduce on the generalization found in oversampling. We will be looking at the NearMiss-3 algorithm for our undersampling. NearMiss chooses data based on the similarities between the majority and minority class with NearMiss-3 selecting the X nearest examples from the majority class for each minority class. We could use a combination of undersampling and oversampling to generate more balanced data.

All int and float columns will be converted to contain the least number of bytes to store them to reduce memory usage and improve computational speed. For example, a column can be converted from an int64 type to int8 type if all values within the column are less than the maximum value of an int8 type. Categorical values will be replaced by using label encoding. Normalization will not be applied to our dataset as it would likely result in the fraudulent transactional data more closely resembling the normal transactional data, making the fraudulent transactions more difficult to detect. Similarly, the techniques for replacing missing values must be carefully selected to avoid the fraudulent transactions resembling the normal transactions. To accomplish this, we simply replace the missing values with a value of ‘-999’. If missing values are not replaced, the gradient boosting models will assign all the missing values to either the left child or right child during a tree node split and cause overfitting.

In addition, the DeviceType column will be simplified by removing the version number on the device. For example, a device name of ‘Windows.12.15’ could simply be replaced with ‘Windows’. This simplification will prevent the classifiers from treating two devices having slightly different version numbers from being treated completely differently.

Feature Engineering

Dimensionality reduction is a critical aspect of this project as there are many features. Feature selection is effective in reducing noise and redundant features that may reduce classifier accuracy. For example, features V1-V339 were all engineered by Vesta containing redundant information. Our project will reduce the number of features by using the following dimensionality reduction techniques: Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and correlation analysis. Additionally, new features will be generated to replace existing features that better correlate to the output feature. For example, the transaction timestamp can be replaced by two columns: day of week and hour of day.

Model Construction

Five different unsupervised classifiers will be developed, namely: Artificial Neural Network (ANN), AdaBoost, XGBoost, LightGBM, and an Ensemble classifier that combines the results of these models. We will explore creating an ensemble classifier by using majority voting and stacking with different combinations of our top performing models. We will explore creating Artificial Neural Networks and optimizing them to identify fraudulent transactions. Artificial Neural Networks are made up of layers of nodes and learn by iterating through each node, comparing outcomes and readjusting the network. We will be experimenting with different numbers of nodes, numbers of levels, activation functions, optimizers and epochs until we find the most accurate prediction. GridSearch will primarily be used to identify the most effective parameters to use for each model.

Gradient Boosting Models have recently become a popular choice for machine learning involving tabular data. Gradient Boosting trains models in a gradual, additive, and sequential manner. Boosting is a method of converting weak learners into strong learners.

In AdaBoost (Adaptive Boosting), a stump (tree with one node and two leaves) for each feature is created where each stump is considered a weak learner. Each stump is created by taking the previous stump’s mistakes into account. The stumps are assigned weights to determine the amount of say each has in making the final classification. XGBoost (Extreme Gradient Boosting) starts by creating an initial prediction for all observations in the dataset and then calculating the residuals to the actual values. A gradient boost tree is then fit to the residuals by calculating the similarity score at various thresholds. Once predictions have been made using the fitted gradient boost tree, the outputs of each leaf are scaled by a learning rate to generate new predictions. Compared to XGBoost and other gradient boosting methods, LightGBM has a much faster training speed and less memory usage. LightGBM leverages a technique of Gradient-based One-Side Sampling (GOSS) to filter out the data instances. CatBoost is particularly powerful at handling datasets containing many categorical features like ours. Unlike the other classifiers, it does not require having to change categorical values into a numerical representation using techniques like label encoding or one hot encoding. In terms of parameter tuning the gradient boosting models, we will primarily experiment with modifying the following: number of epochs, minimum number of observations a node must contain to split, maximum depth of a tree, learning rate, number of estimators, and loss function.

Evaluation

The performance of each classifier will be evaluated by using Repeated K Fold Cross Validation. Since there is six months of transactional data, we intend on using four months for training, one month for validation, and one month for testing. The K Fold Cross Validation process will be repeated for five epochs.

The performance of our models will be optimized to obtain the largest area under an ROC curve (AUC score). An ROC curve shows a graphical representation of the relationship between sensitivity and specificity. The AUC Score is an effective metric for measuring performance when handling a severe class imbalance. In addition, we will be using these common metrics to evaluate performance and compare the performance of our classifiers to similar research papers: overall accuracy, specificity, sensitivity, and F1 score.

|  |  |  |
| --- | --- | --- |
| **Metric** | **Formula** | **Description** |
| Overall Accuracy |  | How often the classifier predicts correctly. |
| Sensitivity |  | Measure of how well the classifier identifies fraudulent transactions |
| Specificity |  | Measure of how well the classifier identifies non-fraudulent transactions |
| F1 Score |  | Harmonic mean between precision and recall metrics. |

**Table 3:** The metrics used to measure the performance of the classifiers.