Credit Card Fraud Detection

Springboard Data Science Career Track Capstone Project I

Reuben Yang

October 2017

1. Introduction

This project deals with the problem of building a machine learning model for fraud detection – identifying the fraudulent transactions from a set of credit card transactions, and hence the clients are financial institutions. The dataset used in this project is published by Kaggle[1], which contains credit card transactions from September 2013, made by European cardholders. This dataset presents transactions that occurred in two days, where there are 492 frauds out of 284,807 transactions. Therefore, the dataset is highly imbalanced as the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables, which are the result of a Principal Component Analysis (PCA) transformation. Unfortunately, due to confidentiality issues, the original features and the detailed background information about the data are not provided. Coded features V1 to V28 are the principal components obtained after applying PCA to the raw dataset. See Table 1 for more details.

Total Transactions	Legal	Fraudulent	Fraud Ratio	Number of Features
284807	284315	492	0.17%	28

Table 1: Summary of Dataset

Feature labeled as 'Class' is the response variable and it takes value 1 in case of fraud, and 0 otherwise. There are no missing values in the dataset.

2. Methodology

This project applies three different approaches. One is the baseline that works directly on the original data without any resampling. The other two are based on resampling techniques, i.e., undersampling and oversampling. A 75%-25% training-test split of the original dataset, with stratification, is used in all three approaches. Therefore, training and test datasets have the same proportion of fraudulent transactions as the original dataset and they are summarized in Table 2.

	Training	Test
Fraud	369 (0.17%)	123 (0.17%)
Non-fraud	213,236 (99.83%)	71,079 (99.83%)

Table 2: Summary of Training and Test Datasets

[1] https://www.kaggle.com/dalpozz/creditcardfraud

The three approaches are discussed further as follows.

1) Baseline: the baseline approach simply takes the original dataset as it comes and trains the models directly on the highly skewed data.

2) Undersampling: to make a dataset with 50-50 fraud and non-fraud transactions, this approach under-samples the majority class (the non-fraudulent transactions in this case) by only randomly selecting a fraction (i.e. the number of frauds) of the non-frauds and pair up these sampled non-frauds with all the frauds to form a balanced dataset. This is illustrated in Figure 1.

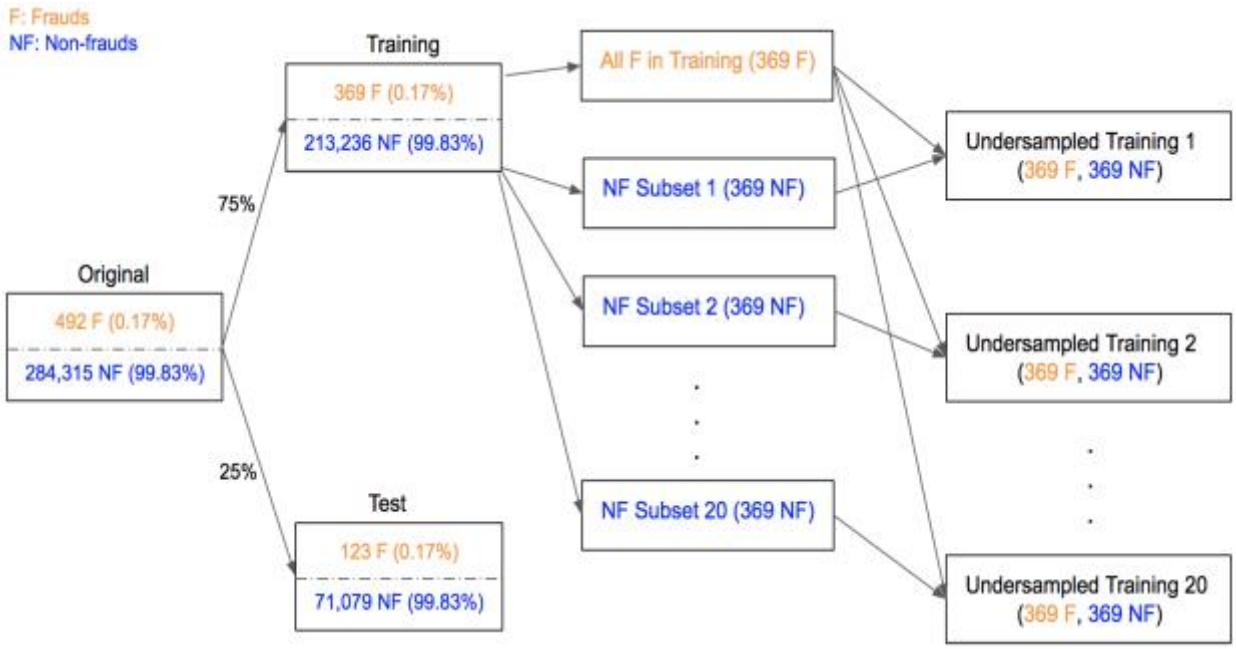

Figure 1: Illustration of Undersampling Approach

More specifically, 20 disjoint subsets of the non-frauds are sampled from the training dataset as indicated in Figure 1. Appending each of the 20 samples of non-frauds to all the frauds in the training data yields an undersampled and balanced training set. The undersampled training sets share the same fraudulent transactions and have different non-frauds subsets taken from the original dataset. Using each undersampled training set to train a model will create 20 classifiers of the same type, and applying them on the test dataset will eventually produce 20 predicted outcomes (0 or 1) for every transaction in the test dataset. Performing a majority vote on those predictions will determine the final prediction of the transaction ("fraud", if more than 10 1's out of the 20 outcomes; "non-fraud", otherwise). This voting

system makes this approach an ensemble-like learning algorithm since the algorithms that take place in the voting are all of the same type.

3) Oversampling: this approach is to generate new samples in the class (the fraudulent transactions in this case) which is under-represented and to make the two classes balanced. This is illustrated in Figure 2.

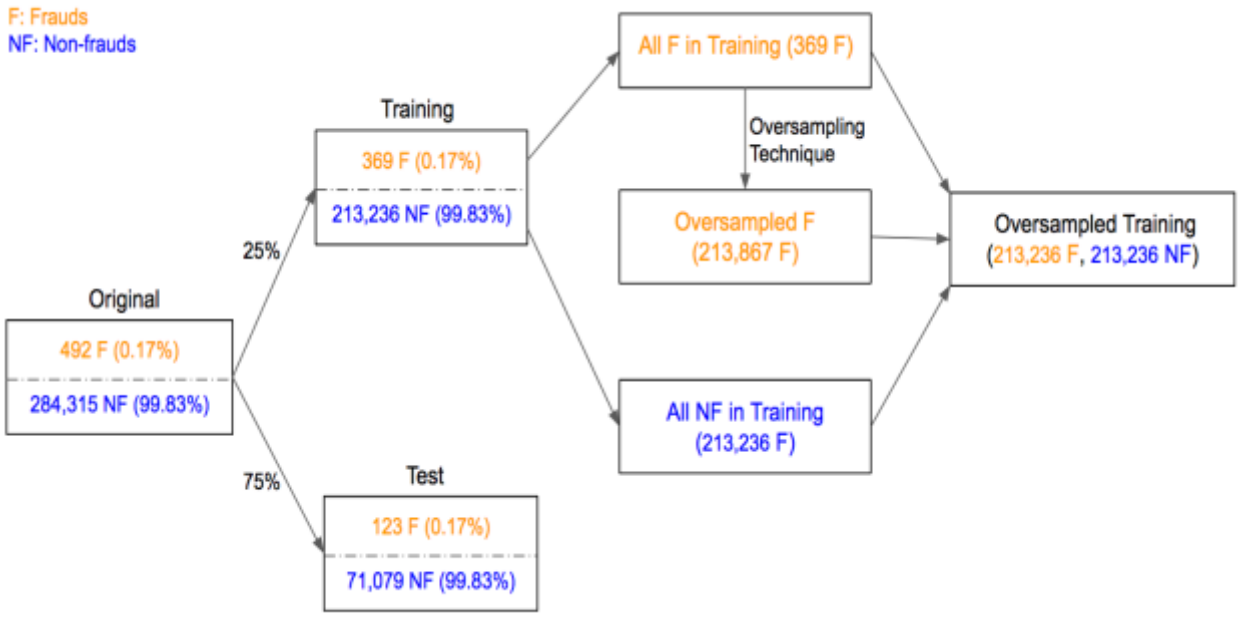

Figure 2: Illustration of Oversampling Approach

The imbalanced-learn package[2] provides a variety of resampling techniques for imbalanced datasets. Three different oversampling techniques are applied in this project:

1. Random Oversampling (RO): randomly sample, with replacement, the current available samples, so each fraud instance from the original dataset will be replicated multiple times to generate the new dataset.

2. Synthetic Minority Oversampling Technique (SMOTE): creates "synthetic" examples by considering a sample from the original dataset and its *k* nearest neighbors. Randomly select one of those *k* neighbors and take the vector between the selected neighbor and the current sample. Multiplying the vector by a random number between 0 and 1, and adding it to the current sample will create a new, synthetic sample. In other words, this created synthetic sample lies on the line segment joining the current sample and one of its *k* nearest neighbors.

[2] http://contrib.scikit-learn.org/imbalanced-learn/stable/index.html

3. Adaptive Synthetic (ADASYN): like SMOTE, ADASYN also creates synthetic data points for the minority class but it focuses on generating samples next to the original samples which are wrongly classified using a k-Nearest Neighbors classifier (these samples are called “difficult to learn”). ADASYN places more weight on minority class examples that are harder to learn and hence shifts the decision boundary towards them.

The following graphs illustrate the major difference of the three oversampling approaches based on variables V1 and V2.

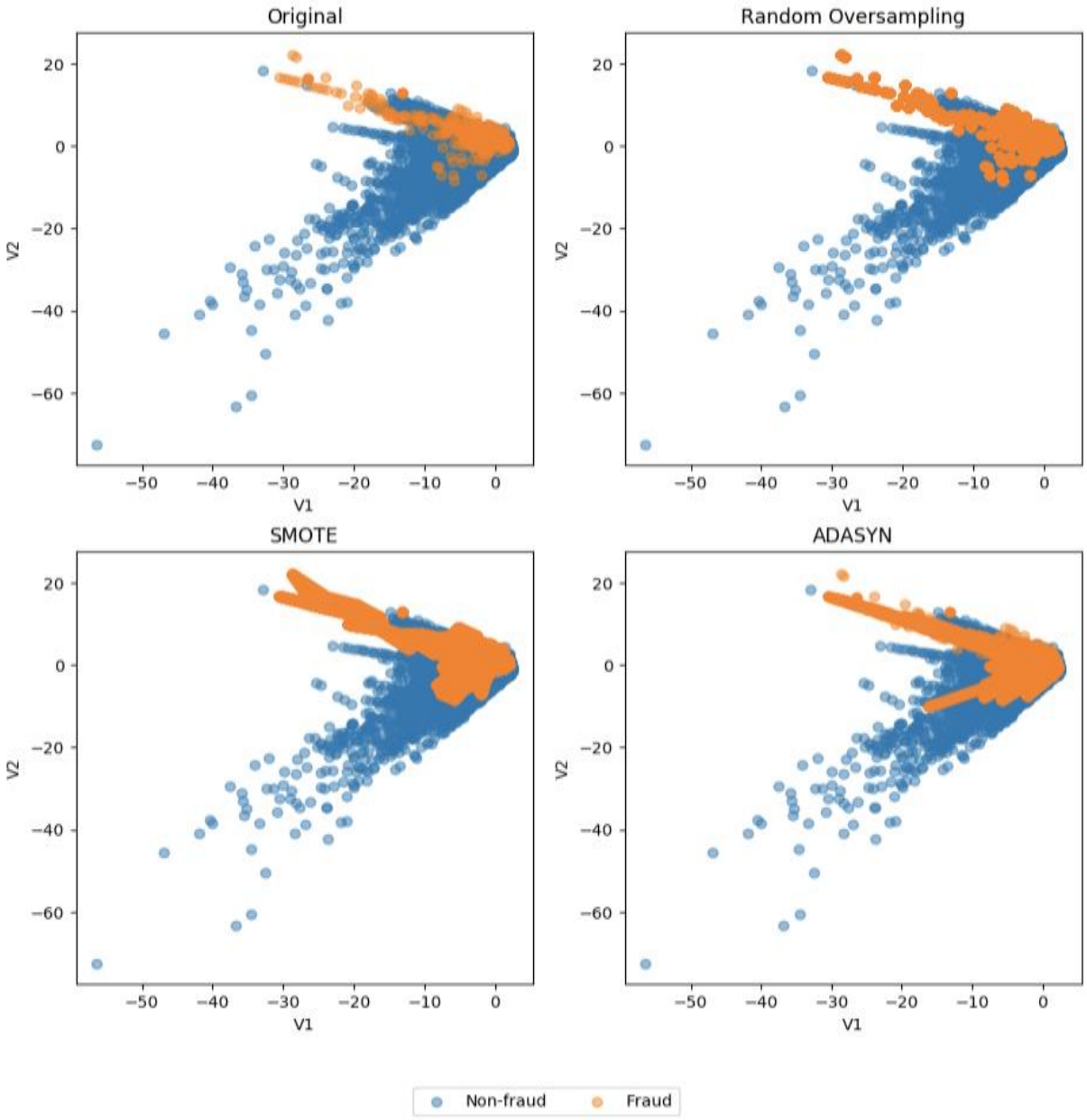

Figure 3: Illustration of Oversampling Techniques

Logistic regression is trained for all three approaches and *L2* norm regularization is used. Five-fold cross validation is applied to tune the regularization parameter for logistic regression, where 0.01, 0.1, 1, 10, and 100 are the candidates considered. Random forest is trained for the baseline and undersampling approach, but it is not currently used for oversampling approach due to the long execution time it requires. Each integer in the closed interval [40, 50] is considered as the number of trees in every random forest and "sqrt" is assigned to *max_features*, i.e. 5 features are checked when looking for a split. As discussed above, for the undersampling approach, 20 logistic regressions and random forests are trained, respectively.

3. Performance Metrics

In learning extremely imbalanced data, the overall classification accuracy is often not an appropriate measure of performance. A trivial classifier that predicts every case as the majority class can still achieve a very high accuracy. In this case, it is clear that the performance of the model should be emphasized on its capability of detecting the fraud transactions (labeled as "1" in the dataset). Hence, the following two scores are employed:

- Recall (true positive rate or sensitivity): fraction of fraud transactions that are successfully detected, i.e. number of correctly predicted frauds divided by the total number of actual frauds.
- Precision (positive predictive value): fraction of predicted fraud transactions that are accurate, i.e. number of correctly predicted frauds divided by the total number of predicted frauds.

This project aims at achieving the highest recall, which means selecting the model parameters based on recall scores, and uses precision as auxiliary.

4. Results

The training-test split process discussed in Section 2 is performed 5 times, i.e. the models are trained and tested on 5 different cases, and each time they are tested on the same samples (123 frauds and 71,079 non-frauds). Figure 4 shows the scatter plot for one of the 5 test cases based on V1 and V2.

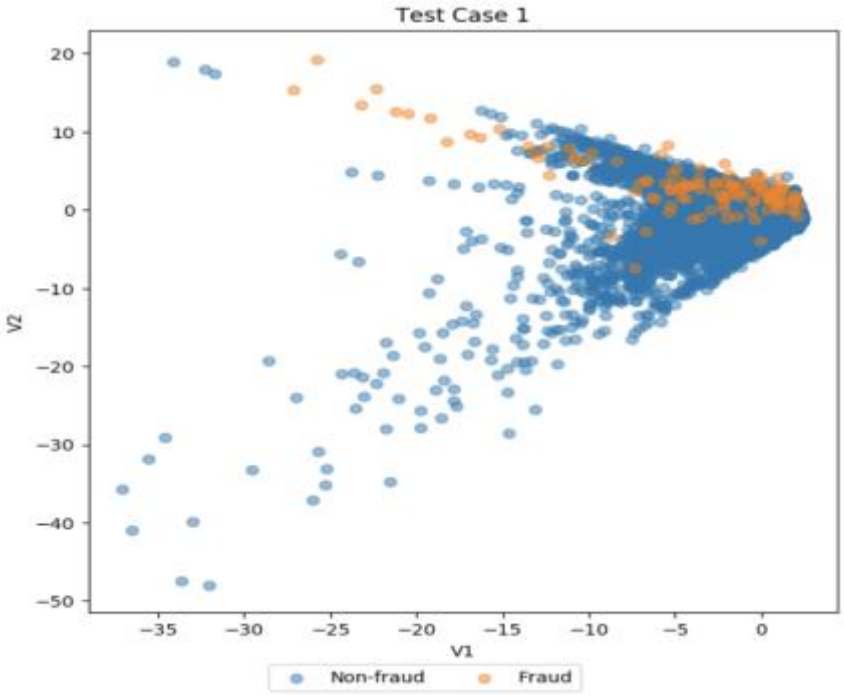

Figure 4: Illustration of Test Set

The recall and precision for the logistic regressions using the baseline, undersampling, and oversampling are shown in table 3 and 4, respectively. The training performance for undersampling approach is the average performance on 20 training sets.

	Baseline		Undersampling		Random Oversampling		SMOTE		ADASYN	
Case	Training	Test	Training	Test	Training	Test	Training	Test	Training	Test
1	0.645	0.642	0.930	0.902	0.924	0.902	0.920	0.902	0.716	0.943
2	0.645	0.618	0.932	0.886	0.922	0.878	0.923	0.869	0.732	0.943
3	0.623	0.650	0.942	0.862	0.938	0.862	0.938	0.862	0.731	0.894
4	0.604	0.642	0.924	0.919	0.922	0.911	0.918	0.919	0.720	0.975
5	0.615	0.602	0.924	0.911	0.919	0.894	0.918	0.902	0.717	0.959
Avg.	0.627	0.627	0.930	0.896	0.925	0.889	0.924	0.891	0.723	0.943

Table 3: Recall for Logistic Regressions

It is clear that the recall is materially enhanced when applying the resampling techniques as opposed to the baseline approach. While the training performance for logistic regression with ADYSYN is not as good as the other resampling approaches, based on the test performance, it is the apparent winner and has

the average recall of 94.3% (116 out of 123 are detected on average). As discussed in Section 2, ADYSYN focuses on the outliers when generating new samples. Also, as illustrated in Figure 3, ADYSYN creates more frauds that overlap with the non-frauds. As a result of this, ADAYSYN leads to a lower training performance, which is due to the fact that the training set it creates has more "hard" cases to learn. The remaining models using resampling have similar recall performance where undersampling is slightly better than Random Oversampling and SMOTE.

	Baseline		Undersampling		Random Oversampling		SMOTE		ADASYN	
Case	Training	Test	Training	Test	Training	Test	Training	Test	Training	Test
1	0.898	0.859	0.973	0.045	0.976	0.064	0.976	0.059	0.839	0.012
2	0.875	0.894	0.973	0.046	0.977	0.068	0.975	0.063	0.854	0.013
3	0.871	0.870	0.967	0.036	0.976	0.058	0.974	0.055	0.849	0.012
4	0.861	0.868	0.967	0.038	0.975	0.060	0.973	0.056	0.826	0.011
5	0.866	0.851	0.969	0.038	0.975	0.060	0.973	0.056	0.847	0.013
Avg.	0.874	0.869	0.970	0.041	0.976	0.062	0.974	0.058	0.843	0.012

Table 4: Precision for Logistic Regressions

From the perspective of precision, the baseline approach is substantially better than all models using resampling techniques based on the test dataset. With the introduction of resampling, the models are trained on the balanced datasets. Therefore, it is actually expected that their precision scores for test sets would drop tremendously given the serious imbalance of the test data. Contrary to the recall scores, the logistic regression with ADYSYN becomes the dead last for precision with the average score about 1.2% which means on average, the 116 successfully detected actual frauds are out of the 9869 predicted frauds. Say something about relevant less false positives.

The recall and precision for random forests using the baseline and undersampling are shown in table 5. Since random forests are designed to give the perfect prediction on the training set, out-of-bag (OOB) performance is used as a measure for the training performance. Here, OOB recall and precision are used instead of the typical OOB error which evaluates the prediction accuracy. In implementation, sklearn.metrics's *recall_score* and *precision_socre* are employed, where the actual classes of the training observations are the ground truth values and the OOB predictions by taking majority vote on the predictions for each training sample x_i using only the trees that did not have x_i in their bootstrap sample are the estimates.

	Recall				Precision			
	Baseline		Undersampling		Baseline		Undersampling	
Case	OOB	Test	OOB	Test	OOB	Test	OOB	Test
1	0.800	0.732	0.907	0.886	0.955	0.928	0.966	0.060
2	0.794	0.732	0.913	0.886	0.948	0.928	0.966	0.050
3	0.789	0.724	0.919	0.862	0.957	0.947	0.966	0.046
4	0.786	0.813	0.908	0.886	0.954	0.901	0.965	0.053
5	0.770	0.789	0.907	0.886	0.944	0.942	0.965	0.052
Avg.	0.788	0.758	0.911	0.881	0.952	0.929	0.966	0.052

Table 5: Recall and Precision for Random Forests

Similar to the logistic regression models, random forests with undersampling boost the recall compared to the baseline approach. The random forests have better performance under baseline than logistic regression that the average recall and precision for test sets increase by 13% and 6%, respectively. Using undersampling, random forest has an overall performance that is pretty identical to that of logistic regression. It's worth noting that only a limited number of tress (integers in the closed interval [40, 50]) are considered for random forests due to the computation time and the performance for random forests will possibly improve when more trees are included.

5. Conclusion

This project applies two resampling approaches, undersampling and oversampling, to detect fraudulent credit transactions and compares their performance with a baseline approach where no resampling is used. With the primary objective of improving the recall scores, both logistic regression and random forest with resampling techniques are effective approaches in capturing the minority class (fraud) in the exceedingly imbalanced data and have performance superior to the baseline approach.

However, the models with resampling approaches clearly have room for improvement as their precision scores suggest. In real practice, a "suspect" fraud transaction reported by machine is usually followed a subsequent manual investigation. The falsely predicted frauds inevitably increase the workload for human staff and may cause inconvenience for credit card customers. Therefore, it is also in the interest of a financial institution to diminish the cost and manpower caused by verifying the large number of false

positives. Further study should be carried out to see how the precision scores can be enhanced without causing too much sacrifice for recall.

6. Recommendations for Client

Since both undersampling and oversampling are useful techniques in detecting the frauds from the extremely skewed credit transaction dataset as the recall scores are pretty satisfactory, it is recommended that the financial institutions consider using these resampling approaches when developing the machine learning models for fraud detection. Based on the experiments, logistic regression with ADYSYN is especially suggested as it has the best performance for recall.

7. Future Work

Several options can be considered:

1) To reduce the false positives, F-score can be used to train the models, which considers both precision and recall. In addition, a second-stage model can be built using the initially predicted frauds which optimizes the precision score during training.
2) For undersampling approach, when assembling the predictions of the 20 models, assign different weights to them based on their individual training performance instead of simply assigning them equal weights.
3) In addition to the resampling techniques used here, there are other methods, for instance, SMOTETomek and SMOTEENN, the combination of oversampling and undersampling, which refines the resulted space obtained after over-sampling.
4) The resampling can be performed first on the original dataset (either undersampling the non-frauds or over-sampling the frauds) and then split the new dataset to training and test. SMOTE and ADASYN can be performed first on the original dataset and then split the new dataset to training and test instead of splitting and only oversampling the train set used here. By doing so, all frauds in the original dataset are employed to create the new samples which may provide more information in the training set for the models to learn and potential further improve the performance. However, it needs to ensure that the test set should include at least some real frauds from the original dataset.