**Credit Card Fraud Detection: Methods of Balancing Data and Machine Learning in Classification Problem**

Shaofan Du

Bachelor of Arts in Statistics and Economics

Case Western Reserve University

**Abstract**

Fraudulent transactions frequently occur for every credit card, and it is important for credit card companies to recognize those transactions and protect the interests of their clients so that they will not be charged for those items or services they did not purchase. The dataset included credit card transactions in September 2013 by European cardholders and was downloaded from Kaggle[1], an open-source data collection website. The main goal of this project is to design and optimize classification algorithms that can automatically detect fraudulent transactions. Framing suitable algorithms by analyzing statistical attributes of the data. Using statistical methods to process the data and enable it to be the best form for training machine learning algorithms to avoid under or over-fitting situations. Taking advantage of the specific indicators to evaluate the precision of different algorithms and find the best one. Programming is based on Python.

**Preview**

***Classification***

Classification problems are very common and an important area of supervised machine learning. Supervised machine learning algorithms create models that depict data relationships. The goal of classification is to determine from a set of features which class or category a given entity falls into. For this task, we are interested in analyzing what kind of transactions will be classified as fraudulent or normal given the data set, which is a classification problem. Given those features, or variables, we can use different machine learning algorithms to build models to describe the relationship among the features, and classify new transactions with a more precise result.

***Close Reading to Data***

This data set contains only numerical values with 30 features. There is no original features and more information about the data because of confidentiality issues. Features V1, V2, …, V28 are the principal components obtained with PCA, which reduces the data dimension and perfectly summarizes the features of the credit cards transactions. The two features that have not been transformed with PCA are “Time” and “Amount”. Feature “Time” contains the seconds elapsed between each transaction and the first transaction in the data set. The feature “Amount” is the transaction amount. Feature “Class” is the responsive variable, and it takes value “1” for the case of fraud transaction and “0” for the normal one. Table 1 shows a brief sample of the data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Time | V1 | V2 | … | V28 | Amount | Class |
| 0.0 | -1.359807 | -0.072781 | … | -0.021053 | 149.62 | 0 |
| 0.0 | 1.191857 | 0.266151 | … | 0.014724 | 2.69 | 0 |
| 1.0 | -1.358354 | -1.340163 | … | -0.059752 | 378.66 | 0 |
| 1.0 | -0.966272 | -0.185226 | … | 0.061458 | 123.5 | 0 |

**Table 1**. Brief sample of the credit card fraud detection dataset.

***Feature Processing***

Recall that the main goal of this project is to detect the fraudulent transactions given the 28 PCA transformed features and 1 non-processed feature (“Amount”). Therefore, feature “Time” is useless in this case, and can be dropped. The feature “Amount” has much larger numbers and variance than those of other features, which will take a dominant place in machine learning algorithms and make them not learn from other features that have smaller numbers. By calculation, the variance of feature “Amount” is 62559.85. But the variance of feature “V1” is only 3.84. This is a huge difference. We can transform every feature example to follow a standard normal distribution by using “StandardScaler” in Python[2]. It calculates the mean () and standard deviation () of the target feature, and then uses the following formula to transform and standardize every value in the feature:

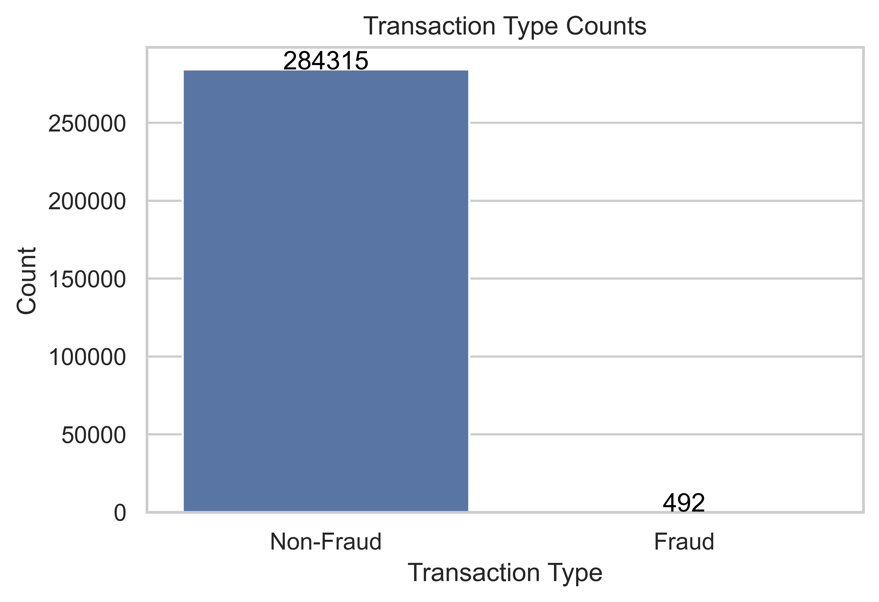
**Table 2** shows the sample of the dataset after feature dropping and transformation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| V1 | V2 | … | V28 | Amount | Class |
| -1.359807 | -0.072781 | … | -0.021053 | 0.244964 | 0 |
| 1.191857 | 0.266151 | … | 0.014724 | -0.342475 | 0 |
| -1.358354 | -1.340163 | … | -0.059752 | 1.160686 | 0 |
| -0.966272 | -0.185226 | … | 0.061458 | 0.140534 | 0 |

**Table 2.** Brief sample of the credit card fraud detection dataset after standardization.

***Highly Imbalanced Data***

The data contains credit card transactions in two days, where there are 492 frauds out of 284,807 transactions. This indicates that the data is highly imbalanced that the proportion of positive class (fraud transactions, or number “1” in column *Class*) is only 0.17%, approximately. In this case, our goal is detecting the fraudulent transactions that is the minority of our data. Given the high-imbalanced one, predictive models based on machine learning algorithms may not be able to make precise forecasts. This is because most of the machine learning algorithms incline to focus on the major category and ignore the features of minor category. The minority could be treated as noises or neglected. Therefore, there is a greater probability of classifying the minority by mistake comparing to the majority. Also, it is important to use suitable evaluation standards. For instance, if we have an algorithm that can predict every transaction in our dataset as a negative class (normal transactions, or number “0” in column *Class*), there will be only 0.17% of the data predicted by mistake. Then accuracy, one of the most common evaluation indicators, can be calculated by (100%-0.17%) = 99.83%, which is high enough to say this algorithm has a great performance with high accuracy. But it cannot detect any fraud transactions. Therefore, it is critical to use specific methods to process the imbalanced data and other suitable evaluation indicators. It will be mentioned in the later section.



**Figure 1**. Comparison of the frequency of normal and fraud transactions. The number of normal transactions is too large to see how many fraudulent transactions are there.

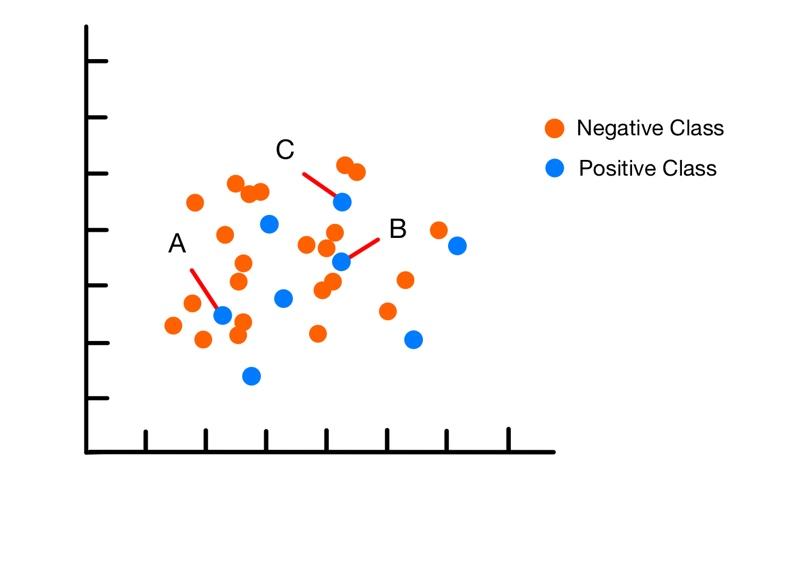
**Data Pre-processing**

***Random Under-sampling***

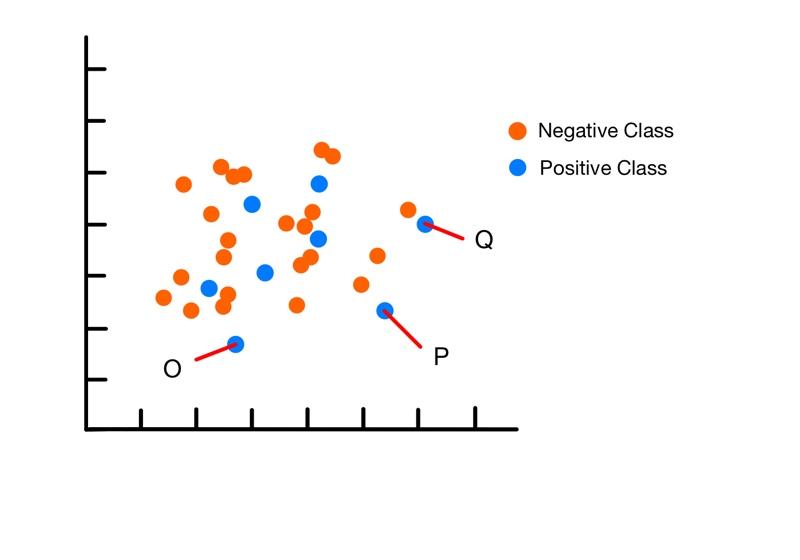
Random under-sampling is a sampling technique by randomly dropping samples in majority class to balance the dataset. This method can be reached by RandomUnderSampler from imblearn in Python[3]. For solving the imbalance problem in our case, random under-sampling is a simple and quick technique. But it may cause the loss of important information of data and over-fitting problem in training process.

***NearMiss Under-sampling***

NearMiss[4] is one of the under-sampling techniques for solving classification problems in imbalanced datasets. It functions to balance the dataset by deleting some of the samples from the majority class. What NearMiss different from random under-sampling is that essentially it is a prototype selection technique that keeps the samples in majority that have smallest average distance to the minority, which are considered to be more representative and important samples. It solves the problem of information loss that usually occurs in random under-sampling method. There are three different methods of NearMiss. Recall that we have positive examples (Class 1: fraudulent transactions) and negative examples (Class 0: normal transactions). For each positive example, the first method (NearMiss-1) selects negative examples whose average distances to three positive examples are smallest. Three points in minority class are randomly chosen, and then NearMiss-1 selects the points from majority class which have the smallest average distance to the three points and make sure the under-sampled data is balanced. As a result, those samples in the majority class that are closest to the minority class example in the overlapping region are retained. But the three randomly chosen positive points probably are not representative to the whole positive class. This method is especially suitable for those cases where the distribution of minority samples is relatively close and the boundary between majority samples and minority samples is clear.



**Figure 2.** Example of NearMiss-1 Under-sampling data with 2 features. NearMiss-1 Algrithom randomly selects three points, A, B, and C, from the positive class. Then it selects the nearest points from negative class to A, B, and C. The number of those points will finally be the same as the number of points in positive class to achieve under-sampling.



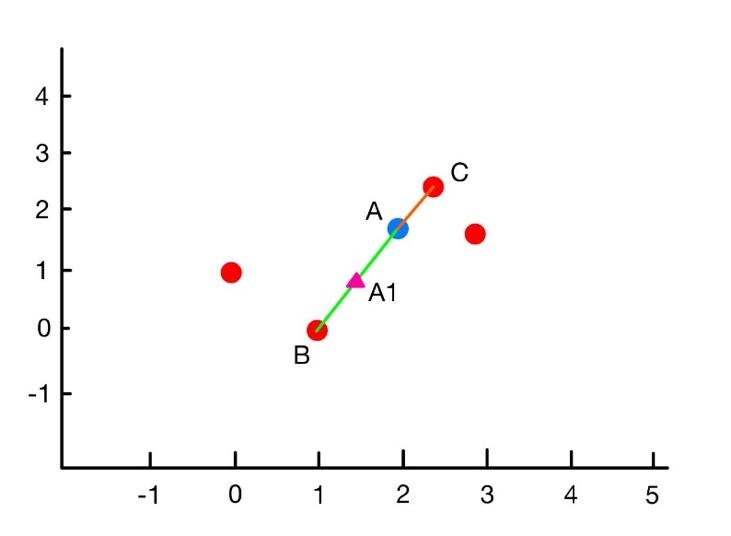
**Figure 3.** Example of NearMiss-2 Under-sampling data with 2 features. NearMiss-2 selects the three farthest positive points O, P, and Q, and chooses those negative points whose average distances to O, P, and Q are smallest.

The second method (NearMiss-2) selects the nearest negative points to three farthest positive points. It focuses more on the negative examples that have farther distance, which may help to find the border between the classes better. However, the third method (NearMiss-3) is different. It directly selects the same number of negative points as that of positives. Each negative point has the smallest average distance to every positive point. It ensures that every positive is surrounded by a negative. This strategy can better preserve the overall structure of the majority class samples and reduce the sensitivity to outliers and noise.

In conclusion, NearMiss under-sampling method will enhance the performance and generalization of the machine learning model. It selects the sample by nearest neighbor from the majority class, which remains the boarder information better between the classes. Later, we will apply the data under-sampled by 3 different methods of NearMiss technique on machine learning models to test the performance.

***SMOTE Over-sampling***

Synthetic Minority Over-sampling Technique (SMOTE)[5] is another popular method to balance the data by synthesizing new samples of the positive class. It is also based on k-nearest neighbors (KNN) algorithm. Based on the k nearest neighbor points of each sample point, it randomly selects N neighbor points by calculating the distance between them and find the nearest ones and lines. Any random points on the lines are selected to be new synthesized samples and then added to the dataset, repeating until the data reaches balance. For example, in **Figure 4**, assume data point A has 4 neighbors (red-colored points) in its feature space. If N is 2, SMOTE will randomly select 2 nearest neighbors, for example, B and C. Then it will calculate the distance between A and B, and A and C. As the green and orange lines showed in the figure, any sample point on the two lines will be selected as a new sample point, like A1. Finally, SMOTE will multiply a random number between 0 and 1 to ensure the data points will not be overlapped.



**Figure 4.** Example visualization of SMOTE algorithm.

SMOTE does not simply repeat generalizing the current minority samples, but produce new synthetic samples by interpolation. This may help to increase the diversity of the dataset and avoid overfitting problems. Also, it keeps the local and global structural characteristics of the original dataset since it uses the nearest neighbors of minority samples, so that the newly generated samples will have similar distribution characteristics to the original one.

***Hybrid Sampling***

A very popular way of hybrid sampling is SMOTE-Tomek Links technique. Tomek Links, introduced firstly by Tomek(1976)[6], is a modified under-sampling technique of Condensed Nearest Neighbors (CNN). CNN randomly selects the samples with its k nearest neighbors from the majority class that are considered to be removed, but Tomek Links employs the rule to choose the pair of observations that satisfies the properties below:

Specifically, it finds a pair , which is defined as a Tomek Link if and are nearest neighbor and belong to different classes, and then removes the one from the majority class to keep the data balance. Then it repeatedly finds such pairs and removes them until it cannot find them.

Noise data refers to those data points far from their own class samples and close to other class samples. Noise data can interfere with the learning process of classifiers, leading to reduced classification performance. In the method of Tomek Links, it effectively removes noise data by recognizing and dropping the adjacent data points from different classes. Furthermore, Tomek Links also delineate the boundary between the two classes and reduce class overlap. In a feature space, class overlap refers to the intermingling of different class data points, resulting in blurred boundaries. This makes it difficult for classifiers to accurately distinguish these data points. By applying Tomek Links method, those adjacent data points from different classes can be identified and removed.

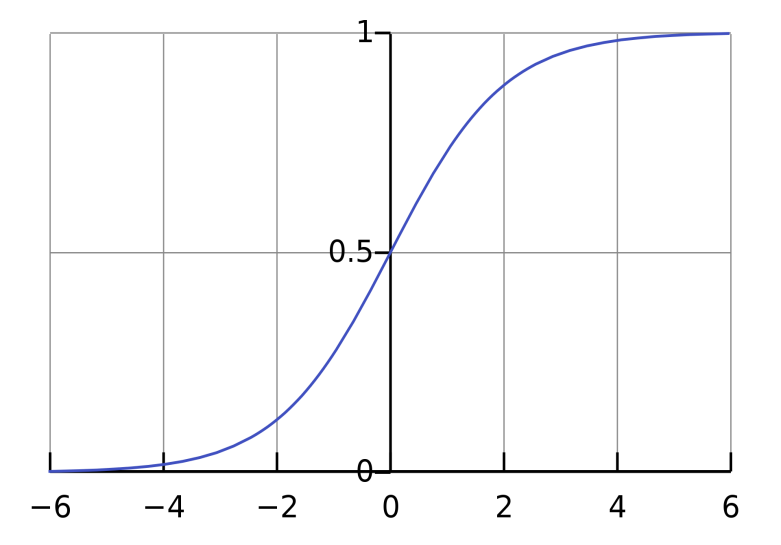
The hybrid sampling method, SMOTE-Tomek Links, was introduced first by Batista *et al.* (2003)[7]. First, it uses SMOTE to generate synthetic minority samples. Then, using Tomek Links to clean the data.

**Machine Learning Models**

***Logistic Regression***

Logistic regression[8] is a classification algorithm for predicting the probability of discrete variables by fitting a linear or non-linear decision boundary, and the basis of the model is the logistic function (Sigmoid function), given by:

.



**Figure 5.** Logistic function (Sigmoid function) graph. By Qef (talk) - Created from scratch with gnuplot, Public Domain,https://commons.wikimedia.org/w/index.php?curid=4310325

Figure 5 shows the plot of Sigmoid function. Logistic regression basically maps a predicted value given by linear regression to the Sigmoid function, which achieves the transformation from value to probability. We define our model as with the parameter . So, the prediction function can be expressed as:

.

In our task, there will be only two predicted values: 0 and 1, separately representing normal transactions and fraudulent transactions. Now, the probability of these two predicted values can be calculated by:

,

and also, can be arranged to one expression:

.

Ideally, we stop fitting the model when the difference between the predicted value and the true value reaches the minimum. Assuming the difference of a single sample is , which is a cost function. Given multiple samples, we calculate the average value of , and define the cost function of all samples as, where

To minimize the cost function , we have:

min .

Therefore, every estimator can be calculated by a continued updated method, also called Batch gradient descent:

,

where and represent the value of iterated to and sample. refers to the Learning rate. Besides, a very common problem using logistic regression in machine learning is over fitting. Over fitting means that the model has high accuracy in fitting training set but low accuracy in fitting testing set, which is mainly because there are numerous features, and the model is very complicated. To avoid over fitting, one popular approach is regularization that lower down the magnitude of parameters by adding regularization factors in the Batch gradient descent cost function. L1 and L2 regularization are two popular methods to prevent overfitting. L1 adds the absolute value of the sum of the parameters to the cost function, while L2 adds the sum of square of the parameters to it, which can be expressed by: (L1 Regularization),

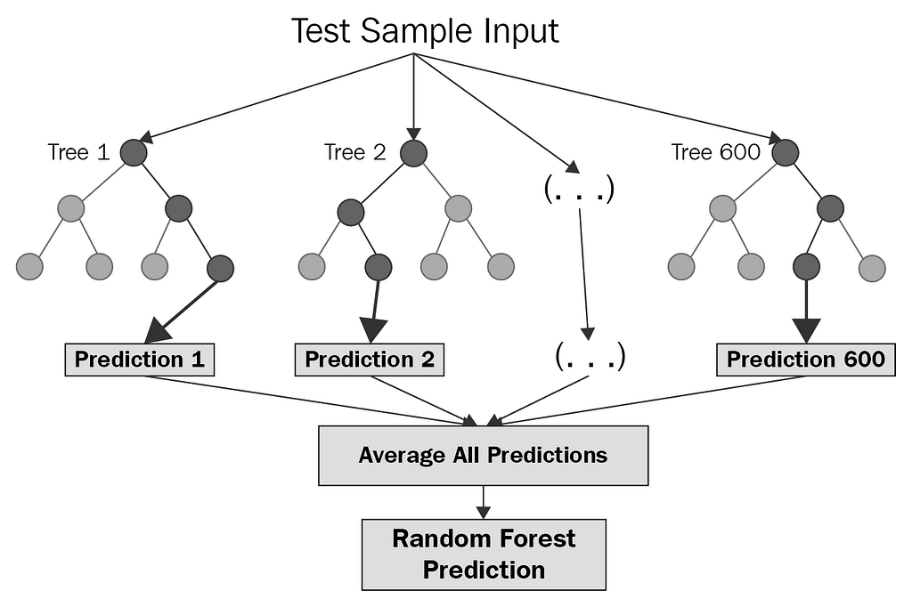
(L2 Regularization),

where is called the regularization parameter. The larger the , the greater the penalty it adds to . L1 regularization is useful for feature selection, which produces sparse solutions and is based on the absolute values of the model’s parameters. L2 regularization, on the other hand, produces non-sparse solutions and is based on the squares of the model’s parameters, which makes it useful for creating more straightforward models.

In conclusion, logistic regression is the simplest and very popular algorithm in machine learning. It learns the data by defining the loss function, and it finds best parameters by minimizing the loss function. We can also enhance the model performance by using L1 and L2 regularization to avoid the overfitting.

***Random Forest***

A decision tree[9] is a supervised learning algorithm that is often used for classification problems. Every decision tree represents a tree structure in which its branches classify objects of that type by attributes. “Tree learning comes closest to meeting the requirements for serving as an off-the-shelf procedure for data mining”, say [Hastie](https://en.wikipedia.org/wiki/Trevor_Hastie" \o "Trevor Hastie) *et al.*[10], “because it is invariant under scaling and various other transformations of feature values, is robust to inclusion of irrelevant features, and produces inspectable models. However, they are seldom accurate”. Deep trees tend to learn highly irregular patterns, known as overlearning, with low bias and high variance on the training set.



**Figure 6.** Random forests algorithm. Cited from Towards AI (2023), <https://towardsai.net/p/machine-learning/why-choose-random> forest-and-not-decision-trees

Decision trees are trained on different parts of a data set, while random forest is a method to average multiple deep decision trees to reduce the variance and improve the performance. Because given a set of independent observations , the variance of each observation is given by , while the variance of the mean is given by . Hence averaging a set of observations reduces variance. The foundational idea of a random forest is bagging. “A neutral way to reduce the variance and hence increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions” [11]. If the prediction of the tree for every observation can be expressed as using separate training sets, and a model with reduced variance by averaging them is given by:

.

By taking repeated samples from single training dataset (Bootstrap), we can get different bootstrapped training datasets. Then training them on the th bootstrapped training set to get , and average them to get:

,

which is called bagging.

When creating these decision trees using the random forests method[12], a random sample of predictors is selected as split candidates from the entire set of p predictors. One of those predictors can only be used in the split. The number of predictors examined at each split is roughly equal to the square root of the total number of predictors. A new sample of predictors is chosen at each split, and normally we select . In other word, features are used in each split for a classification problem with features. Also, feature importance can be computed by that is defined by:

,

where is the proportion of training observations in the th region that are from the th classes. It is a measure of total variance in classes.

**Model Selection**

***k-Fold Cross-Validation***

k-Fold Cross-Validation[13] is a method of evaluating and optimizing the performance of machine learning models. It randomly samples sets of the data. Selecting set as the testing set and sets as the training sets. It repeats the selecting by times so that every subset of the data has a chance to become a testing set. Then different models can be trained and obtained given those training sets. Those models then are tested for the best parameters given the testing sets. Finally, the average of resulting parameters given the test is considered as the best parameter for the model.

***Grid Search with Cross-Validation in Python***

Grid Search[14] is a technique of selecting parameters. Similar to exhaustive search, it iterates through each possibility and selects the best performing parameter out of all the candidate parameter choices. Grid Search with Cross-Validation achieved in Python Skit-learn Model is the combination of k-Fold Cross-Validation and Grid Search to find the best model parameters. We will use this evaluation method in our task.

***Recall, Precision, and F1-Score***

In order to check if the prediction result is good or not, it is important to use suitable evaluation scores as mentioned before. In a classification problem, the classifiers predict four kinds of results:

True Positive. The predicted and actual values are both positive.

False Positive. The predicted value is positive, but the actual value is negative.

True Negative. The predicted and actual values are both negative.

False Negative. The predicted value is negative, but the actual value is positive.

We can define a probability space and view recall score as the probability of the event in the probability space occurs. Given the positive examples are frauds and negative examples are normal in the fraudulent transaction detect task, it can be defined by:

Recall score is the ratio of the number of frauds correctly predicted to the actual total number of frauds:

Precision is another evaluation score. The probability space can be defined by:

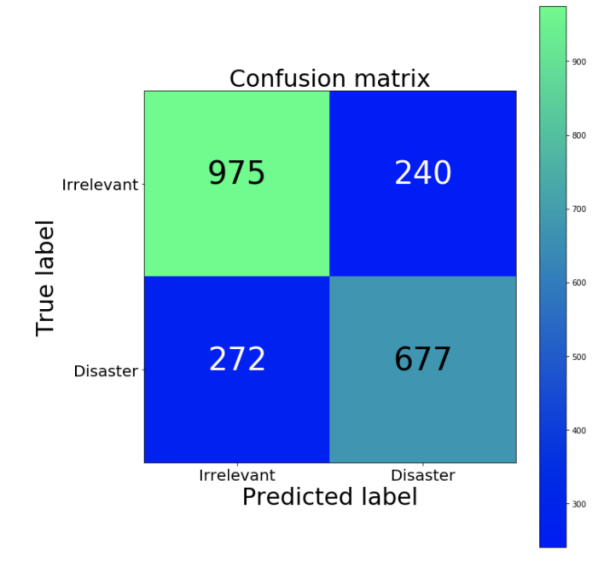
Precision score is the probability of the number of correctly predicted frauds to the total number of predicted frauds:

F1-score is the harmonic average of precision score and recall score, and can be calculated by:

F1-score can more comprehensively evaluate the performance of credit card fraud detection models. In the credit card frauds detection task, it is more important to focus on the recall score because higher recall score means that less fraudulent transactions will be missed, which help lowering down the cardholder’s loss and protecting their interest. But sometimes extremely high recall score leads to higher false alarm. [15]

***Confusion Matrix***

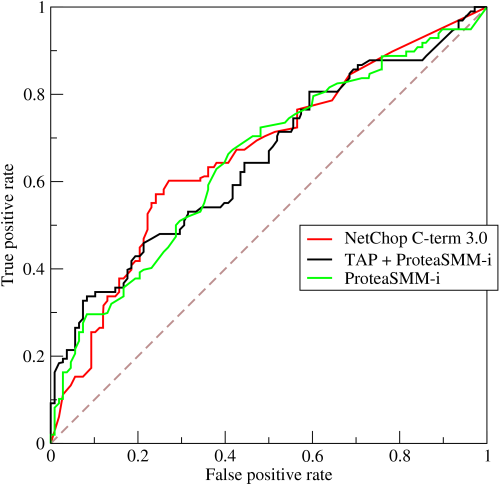
Confusion matrix[16] is a matrix that summarizes the number of , , , and . It can be expressed in a coordinate system constructed by x and y axis. A typical confusion matrix is showed by Figure 7.



**Figure 7.** An example of confusion matrix. By Emmanuel(2018), [How to solve 90% of NLP problems: A step-by-step guide – O’Reilly (oreilly.com)](https://www.oreilly.com/content/how-to-solve-90-of-nlp-problems-a-step-by-step-guide/)

***ROC-AUC***

ROC curve expresses the relationship between true positive rate and false positive rate. True positive rate represents the proportion of fraudulent transactions are correctly predicted as fraudulent transactions, and false positive rate represents the proportion of normal transactions are incorrectly predicted as fraudulent transactions. AUC is the area under ROC curve. If a fraud sample and a normal sample are randomly selected, the probability that the classifier correctly determines that it is highly likely to be fraudulent. The probability is AUC. [17]



**Figure 8.** An example of ROC-AUC curve. By BOR in Wikipedia, CC BY-SA 3.0，https://commons.wikimedia.org/w/index.php?curid=10714489

**Results**

First and the most importantly, we need to partition the data for the training set and the testing set. The training set is used by the machine learning model to learn and fit the data. The testing set is used for testing the performance of the machine learning models. The partition standard is 80% training set and 20% testing set. ‘X\_train’ and ‘X\_test’ are the training and testing sets for the independent variables, and ‘y\_train’ and ‘y\_test’ are the training and testing sets for the dependent variable (target variable: ‘Class’).

|  |  |  |
| --- | --- | --- |
| Name | Type | Size |
| X\_train | DataFrame | (227845, 29) |
| X\_test | DataFrame | (56962, 29) |
| y\_train | Series | (227845, ) |
| y\_test | Series | (56962, ) |

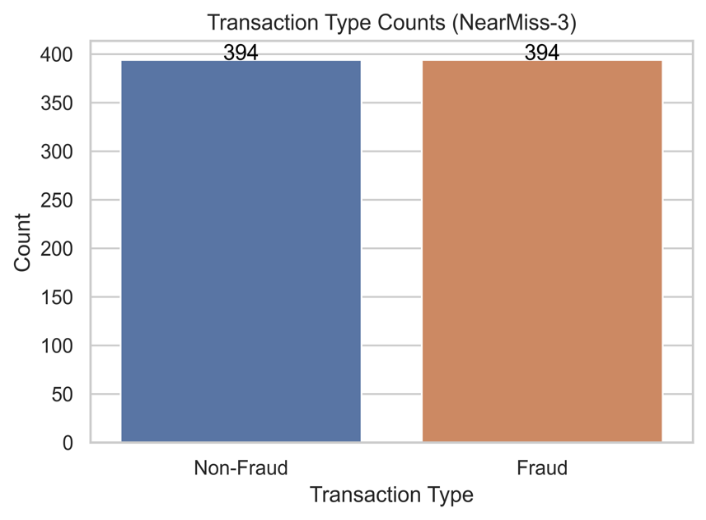
**Table 3.** Training set and testing set partitioning results in Python.

By Random Under-sampling, NearMiss-3 Under-sampling, SMOTE Over-sampling, and SMOTE-Tomek Links Hybrid sampling, different results were obtained. Due to there being 29 features, it is difficult to draw scatter plots to visualize the sampling method, but a frequency comparison between ‘Class: 0’ and ‘Class: 1’ can be showed.

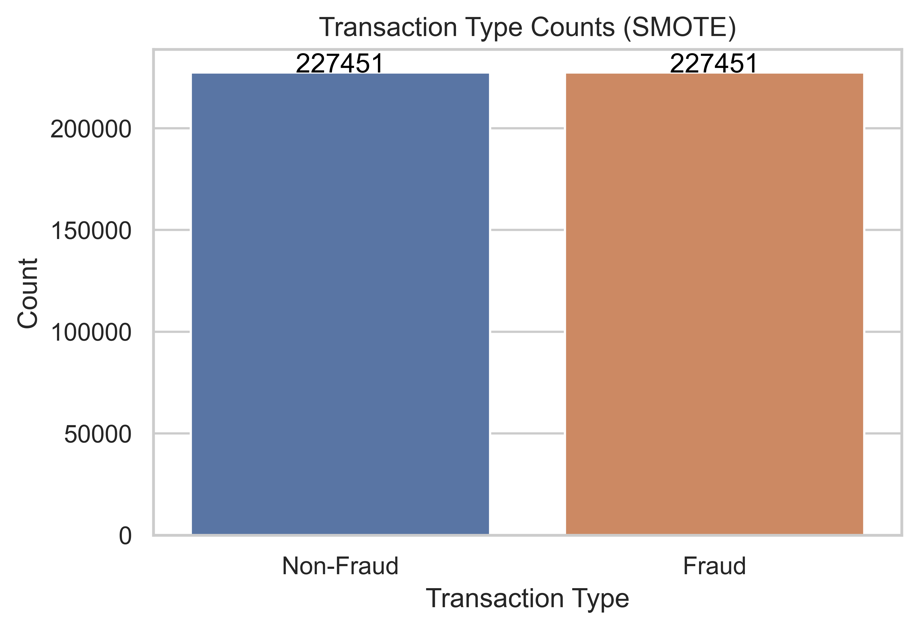
图表, 条形图

描述已自动生成

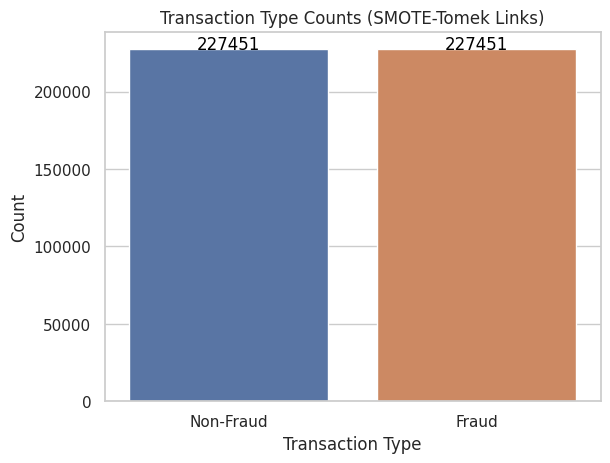
**Figure 9.** Comparison of the frequency of normal and fraud transactions after random under-sampling method. The frequencies of two classes are same.



**Figure 10.** Comparison of the frequency of normal and fraud transactions after NearMiss-3 under-sampling method. The frequencies of two classes are same.



**Figure 11.** Comparison of the frequency of normal and fraud transactions after SMOTE under-sampling method.

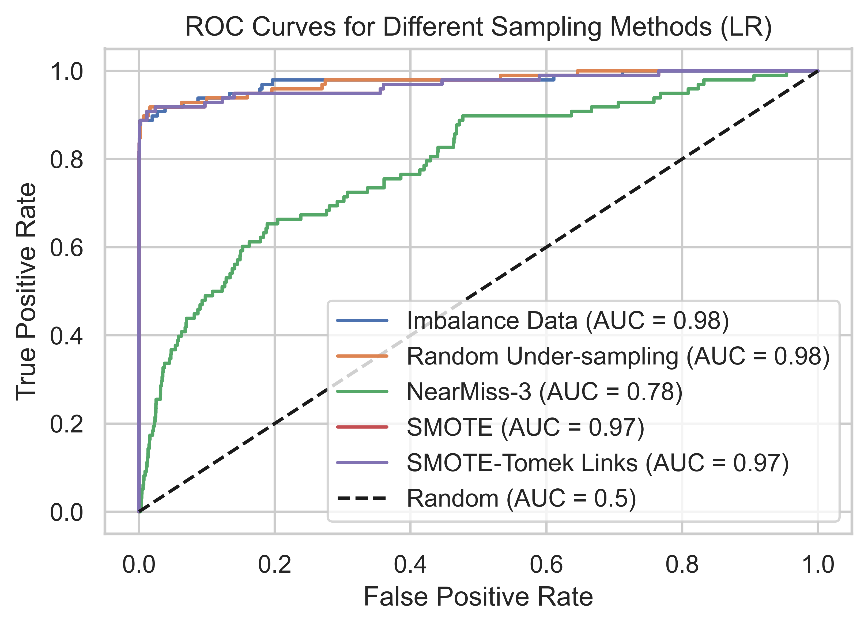


**Figure 12.** Comparison of the frequency of normal and fraud transactions after SMOTE-Tomek Links hybrid-sampling method.

Now, it is time to use logistic regression to train the data. In order to choose the best parameters for the model using Grid Search with Cross-Validation, {0.001, 0.01, 0.1, 1, 10, 100, 1000} are the candidates of regularization parameter, and {L1, L2} are the candidates of the type of regularization. The GridSearchCV method in Python will try those candidates and find the best model and finally fit the data. As a result, it selects L2 as the type of regularization and 0.1 as the regularization parameter for the optimized learning model. The comparison of the results among the imbalanced data and the other four processed data are showed as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 Score | ROC-AUC | Confusion Matrix  [[ TP FP]  [ FN TN]] |
| Imbalance | 0.82608 | 0.58163 | 0.68263 | 0.97689 | [[ 57 12 ]  [ 41 56852]] |
| Random | 0.03705 | 0.91836 | 0.07123 | 0.97646 | [[ 90 2339 ]  [ 8 54525]] |
| NearMiss-3 | 0.00255 | 0.89795 | 0.00508 | 0.78478 | [[ 88 34397]  [ 10 22467]] |
| SMOTE | 0.05597 | 0.91836 | 0.10551 | 0.96996 | [[ 90 1518 ]  [ 8 55346 ]] |
| SMOTE-TL | 0.05597 | 0.91836 | 0.10550 | 0.96996 | [[ 90 1518 ]  [ 8 55346 ]] |

**Table 4.** Result of logistic regression.



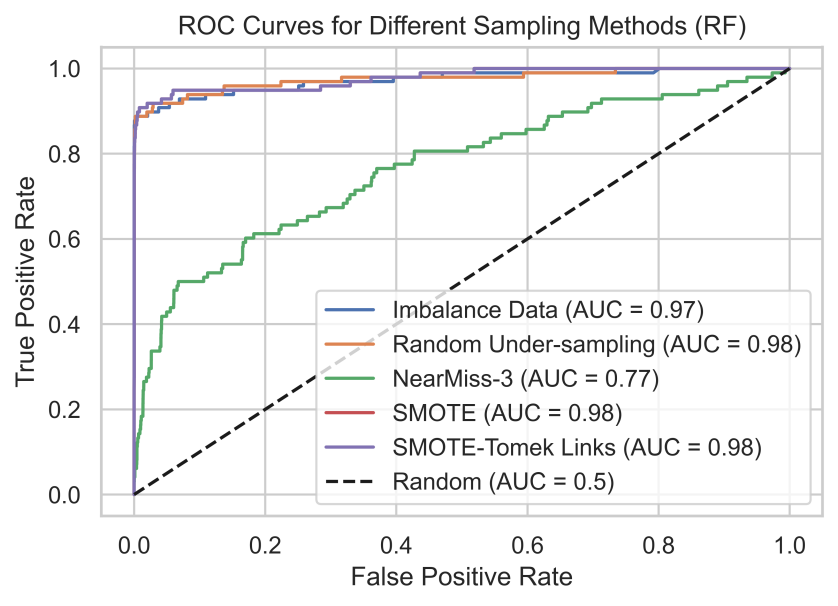
**Figure 13.** ROC-AUC Curve for logistic regression.

First, the original (imbalanced) has the highest precision score, F1 Score, and similar ROC-AUC score as random under-sampling, SMOTE, and SMOTE-Tomek Links. The reason of it has the highest precision score is because the data is highly imbalanced, and so the F1 Score is higher than others. However, it has the lowest recall score, which means it has only 60% probability to detect a fraudulent transaction, but others have approximately 90% to detect that. NearMiss-3 has the lowest ROC-AUC score and the second lowest Recall score. This probably is because it has very high false alarm that sometimes predict a normal transaction as a fraudulent one. Taking a look at the confusion matrix. For the imbalanced data, the model predicts the least fraudulent transactions, but others have nearly 90 detections of frauds. For NearMiss-3, although the model can correctly detect 88 frauds, it predicts 34397 actual normal transactions as frauds. The false alarm is too high. SMOTE and SMOTE-Tomek Links have the same results. This is because the data over-sampled by SMOTE has a very clear boundary between the two classes, so there is no noise or overlapping data needed to be removed by Tomek-Links.

Now, let us see how well the performance of the random forest is. We also use Grid Search with Cross-Validation to choose the parameters of the random forest, including the number of trees in the forest, the criterion, and the maximum depth of the tree. The optimized parameters are {n\_estimators: 100, criterion: gini, max\_depth: 5}. The comparison results are shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 Score | ROC-AUC | Confusion Matrix  [[ TP FP]  [ FN TN]] |
| Imbalance | 0.95121 | 0.79591 | 0.86666 | 0.97327 | [[ 78 4 ]  [ 20 56860]] |
| Random | 0.05666 | 0.89795 | 0.10660 | 0.97556 | [[ 88 1465 ]  [ 10 55399]] |
| NearMiss-3 | 0.00172 | 0.98979 | 0.00343 | 0.76616 | [[ 97 56207 ]  [ 1 657 ]] |
| SMOTE | 0.27301 | 0.87755 | 0.41646 | 0.97805 | [[ 86 229 ]  [ 12 56635 ]] |
| SMOTE-TL | 0.27301 | 0.87755 | 0.41646 | 0.97805 | [[ 86 229 ]  [ 12 56635 ]] |

**Table 5.** Result of random forest.



**Figure 14.** ROC-AUC Curve for logistic regression.

The recall score of imbalanced datasets is increased to nearly 0.8, and the ROC-AUC score is very high, 0.97. Given the confusion matrix, it can detect 78 fraudulent transactions, which is much better than in the logistic regression model. The dataset resampled by random under-sampling is also better. The false alarm number decreased from 2339 to 1465, and it also can detect 88 frauds. Recall score and ROC-AUC is still very high. For NearMiss-3, the recall score is extremely high, leading to predicting 56207 normal transactions as fraud even though it can correctly detect 97 frauds, which is the highest one. SMOTE and SMOTE-Tomek Links are still the same. It has higher precision but lower recall score. The ROC-AUC is still very high. It can correctly detect 86 frauds and only 229 false alarms.

**Conclusion**

From the prediction results and evaluations using logistic regression model, it will be suitable to choose random under-sampling or SMOTE over-sampling method as processed data to build the model because both of the recall scores and ROC-AUC are pretty high. On the other hand, given the result of the random forest model, imbalanced data has better performance than under-sampled data probably because it includes more original information, and random forest is not very sensitive to imbalanced data. But as a result, SMOTE over-sampled data performs the best, especially in random forest model. Not only it has the highest ROC-AUC score, but it can also detect many fraudulent transactions and has low false alarms.

**References:**

[1] Credit Card Fraud Detection (2021), *Kaggle,* <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>.

[2] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830.

[3] Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, *18*(1), 559-563.

[4] Mani, I., & Zhang, I. (2003, August). kNN approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets* (Vol. 126, pp. 1-7). ICML.

[5] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.

[6] Tomek, I. (1976). AN EXPERIMENT WITH THE EDITED NEAREST-NIEGHBOR RULE.

[7] Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, *6*(1), 20-29.

[8] Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (Vol. 398). John Wiley & Sons.

[9~13] Gareth, J., Daniela, W., Trevor, H., & Robert, T. (2013). *An introduction to statistical learning: with applications in R*. Spinger.

[14] Brownlee, J. (2016). How to grid search hyperparameters for deep learning models in python with keras. *linea]. Disponible en*.

[15] Yacouby, R., & Axman, D. (2020, November). Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In *Proceedings of the first workshop on evaluation and comparison of NLP systems* (pp. 79-91).

[16] Liang, J. (2022). Confusion Matrix: Machine Learning. *POGIL Activity Clearinghouse*, *3*(4).

[17] Bowers, A. J., & Zhou, X. (2019). Receiver operating characteristic (ROC) area under the curve (AUC): A diagnostic measure for evaluating the accuracy of predictors of education outcomes. *Journal of Education for Students Placed at Risk (JESPAR)*, *24*(1), 20-46.