CREDIT CARD FRAUD DETECTION WITH IMBALANCED CLASSIFICATION APPROACHES

Rugved Sai Karnati
New York University
rsk517@nyu.edu

Michael Chang
New York University
hyc410@nyu.edu

Abstract:

Credit card fraud detection is a cause for severe repercussions for financial institutions and individuals. Hence detection of fraud is an important topic as fraud could lead to changing the lives of people in a bad way. Usually fraud detection is time consuming and many approaches have been developed to alleviate it but most of them are ineffective. One of the major problems is imbalanced data classification. Imbalance classification consists of having a small number of observations of the minority class compared with the majority in the data set. Usually ML researchers try to improve the algorithm and tend to ignore the imbalanced datasets and In turn have poor performance on minority class although the performance on minority class is the one which is important. In this work we test different sampling methods with different ML architectures to find the appropriate design for fraud detection.

Introduction:

Credit card fraud detection is not a simple issue which can taken lightly as it affects a lot of people in day to day life. A lot of people can lose their livelihoods due to this. It is the one of the most common type of fraud which is the fake use of credit card. This happens when fraudsters use the credit card without the knowledge of the credit card holder or financial institution. It is known as criminal activity and is punishable by law.

So to alleviate this problem a lot of researches have proposed different techniques to the detection of credit card fraud. Detecting using traditional method is infeasible because of the big data. A lot of institutions and researchers are working with computational methods. Classification can be used according to those researchers but the performance with the available classification approaches isn't too high and there is a problem of imbalanced datasets. That could lead to a huge problem if ignored as misclassifying fraud might lead to severe repercussions. Here fraud is in minority class and this is the important one that we need to identify so handling imbalanced dataset is important. Minority class means accounting to very less part of the dataset for example it could be 2% of the dataset.

As the majority class percentage is so high that most of the optimization techniques performed by classification aims to correctly classify dominant class which is not what we want to do. Several works have been done in regards to this problem. Class sampling is a common method to handle with the imbalanced data classification problem. This method operates on the data itself (rather than the model) to increase its balance. It includes oversampling which is add to the dataset and undersampling is to remove from the dataset.

In this paper we test different sampling techniques along with modeling the produced data after this sampling. We model the data using techniques such as ANN, CNN and auto encoder.

Related Work:

A lot of different techniques are tested by various researchers in order to alleviate the problem of credit card fraud. Techniques such as artificial neural networks , support vector machines, genetic algorithms, regression, decision tree etc. A comparative analysis of logistic regression and Naïve Bayes is carried out in [1]. Weston et al. [2] used peer group analysis on real credit card transaction data to find outliers and suspicious transactions. In Duman and Ozcelik [3] genetic algorithms were used to identify wrongfully classified transactions. Ogwueleka used an artificial neural network with a rule based component.

The work to reduce the problem caused by data imbalance was also done using various techniques. Class imbalance means to have most of the data points from the one of the classes which makes it hard for the classifier to classify the minority class. One of the techniques is to work with data and reduce it at that step. The research shows that sampling methods have been used to reduce this. Krawczyk [4] reduced the problem of skewed class distribution using said sampling techniques. Wei et al. [5] approached this problem by utilizing neural networks and decision forest. Buda et al. [6] studied the approach of using CNN to alleviate this problem. Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang at [7] proposed a convolutional neural network (CNN) based approach to find fraud.

Background:

Autoencoder Neural Network

Autoencoder is an artificial neural network used to learn efficient data encoding in an unsupervised manner. It’s a self supervised technique. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal “noise”. The purpose is to reconstruct its own inputs instead of predicting the target value from the given inputs.

ANN (Artificial Neural Network)

It is simply a computing system inspired by the neural networks that make up the animal brain. It is a collections of units known as neurons, each of these neurons have receive an input and produce an output which can be sent to the other neurons or the output layer. The outputs of the final output neurons of the neural net accomplish the task. The network usually consists of an input layer, hidden layer(s) and an output layer. The weights used are first set randomly, then using the training set these weights are adjusted to minimize the error, using specific algorithms like backpropagation.

CNN (Convolutional Neural Network)

Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. CNN usually detects basic features, extracts more complex features, detects higher-level features. CNNs integrate automatic feature extraction and discriminative classifier in one model, which is the main difference between them and traditional machine learning techniques.

Oversampling

Random oversampling is one such technique where random data points from the minority class are selected with replacement and these are used to supplement the dataset by making copies of these points. But there are issues with this as this could lead to overfitting and this also doesn’t provide much new information to the dataset.

SMOTE (Synthetic Minority Oversampling Technique) is one of the most popular oversampling technique. It usually works by selecting examples that are close in the feature space then drawing a line between those examples in the feature space and then drawing a new sample on that line. One of the issues with this is that the synthesized samples are creating without taking into the consideration the majority class. ADASYN also known as adaptive synthetic sampling approach is another method of oversampling where the idea to use weighted distribution for different minority class examples according to their level of difficulty in learning, where more synthetic data is generated for minority class examples that are harder to learn compared to those minority examples that are easier to learn is used.

Undersampling

Undersampling techniques remove examples from the training dataset that belong to the majority class in order to better balance the class distribution.
Random undersampling is a method where we randomly select data points from the majority class and delete them from the training dataset. The issue with this is that we randomly delete data points without concern for how useful they might be in detecting fraud i.e the decision boundary of the class.

Tomek Link uses some defined rules to select the pair of observation to be removed. The rules are that the pair should be the nearest neighbor to each other and that the pair are of different classes. In Cluster centroids approach it uses the concept of finding cluster centroids and then deleting the data points that are farthest from the cluster centroid for the majority class and the one nearest to the centroid is important.

Dataset

The dataset contains transactions made by credit cards in September 2013 by European cardholders. The dataset is imbalanced, there are only 492 transactions (0.172%) which are fraudulent and the other 284315 transactions (99.828%) are not. It contains only numerical input variables which are the result of a PCA transformation. The information regarding the columns are not provided but the columns that are provided as a sequence of PCA transformation are named as V1, V2 .. V28. These are the principal components obtained after PCA. Time and Amount are features which are given other than the principal components.

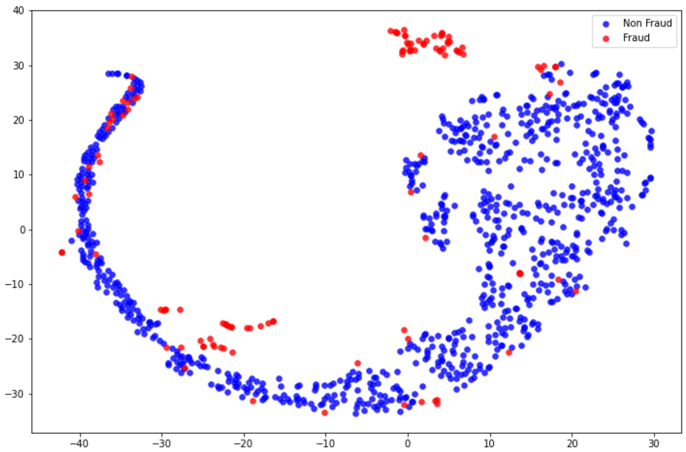


Fig 1. Dataset

Methodology

The experiments conducted in our study are described in this section.

Data Processing and Sampling

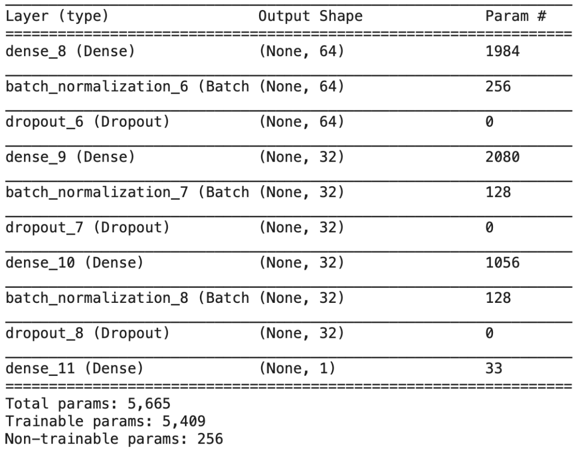
We import data and randomly select 4428 data from non fraud data (9:1 proportion to fraud data) and do a 80/20 train test split. For the preprocessing step we first standardized the selected train data. Also, all networks were trained from a random initialization of weights and no pretraining was applied.

We perform sampling with the help of imbleran. We perform sampling before training the model with the data. So we train the selected model with the sampled data. Different sampling techniques that were used were Random Oversampling and Undersampling, Cluster Centroids, Tomek Links, ADASYN, SMOTE.

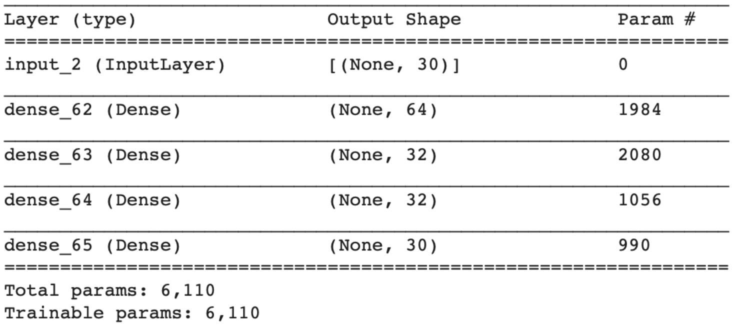
Before sampling the dataset contained 3936 examples and after oversampling with Randomness or SMOTE we have around 7000 example records and with ADASYN we have around 7125 examples. When we use undersampling we have 3936 examples before the sampling is applied. After sampling is applied we have 784 examples when Cluster method or Random undersampling is applied and we have around 3913 examples when we apply the Tomek Link method. We can see that Tomek Link method didn't reduce the dataset that much.

Modeling

After the sampling, we use the sampled data as input for the different models we tested. The models include ANN, CNN, Autoencoder. These have different architectures. After experimentation the below architecture produced better results. In the experiment with ANN (shown in Fig 2.) the network is composed of 3 Dense layers with relu activation function, and Batch Normalization and Dropout are applied to the these layers to improve the learning rate and also reduce overfitting. The final layer is a dense layer with sigmoid activation function. 100 epochs were run with Adam optimizer and minimized the cross entropy loss. Furthermore we used a batch size of 128 and learning rate of 0.01.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 64)                1984
_________________________________________________________________
batch_normalization_6 (Batch (None, 64)                256
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_9 (Dense)              (None, 32)                2080
_________________________________________________________________
batch_normalization_7 (Batch (None, 32)                128
_________________________________________________________________
dropout_7 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_10 (Dense)             (None, 32)                1056
_________________________________________________________________
batch_normalization_8 (Batch (None, 32)                128
_________________________________________________________________
dropout_8 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 33
=================================================================
Total params: 5,665
Trainable params: 5,409
Non-trainable params: 256
```

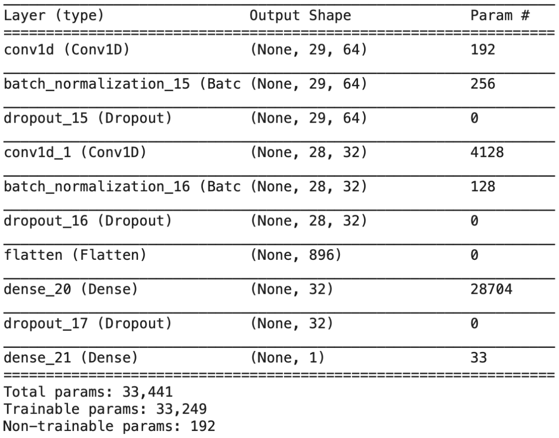


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 30)]              0
_________________________________________________________________
dense_62 (Dense)             (None, 64)                1984
_________________________________________________________________
dense_63 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_64 (Dense)             (None, 32)                1056
_________________________________________________________________
dense_65 (Dense)             (None, 30)                990
=================================================================
Total params: 6,110
Trainable params: 6,110
```

Fig 2. Artificial Neural Network architecture.

Fig 3. Autoencoder architecture

In the experiment with Autoencoders we have 1 input layer , 2 dense layers as encoder layers and 1 layer as decoder layer and 1 output layer and we add regularization to the first encoder layer to reduce overfit and is run for 100 epochs. MSE is used as a loss function and Adam optimizer is used.



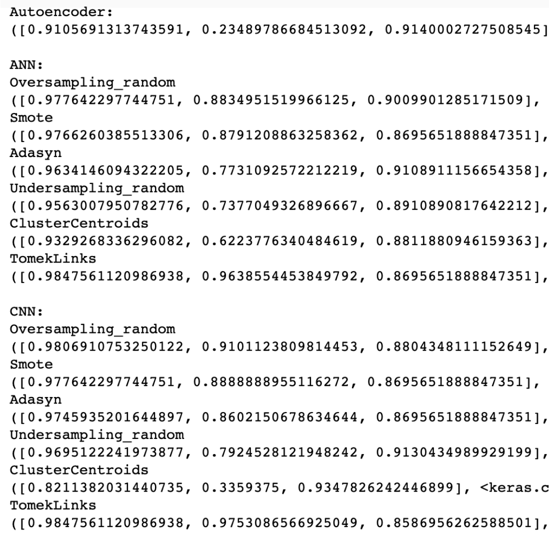
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 29, 64)            192
_________________________________________________________________
batch_normalization_15 (Batc (None, 29, 64)            256
_________________________________________________________________
dropout_15 (Dropout)         (None, 29, 64)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 28, 32)            4128
_________________________________________________________________
batch_normalization_16 (Batc (None, 28, 32)            128
_________________________________________________________________
dropout_16 (Dropout)         (None, 28, 32)            0
_________________________________________________________________
flatten (Flatten)            (None, 896)               0
_________________________________________________________________
dense_20 (Dense)             (None, 32)                28704
_________________________________________________________________
dropout_17 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_21 (Dense)             (None, 1)                 33
=================================================================
Total params: 33,441
Trainable params: 33,249
Non-trainable params: 192
```

Fig 4. 1D-CNN architecture

In the experiment with 1D-CNN (shown in Fig 4.) the network is composed of 2 Convolutional layers with 64 filters for the input layer and 32 filters for the middle layer with the kernel size as 2 and relu activation function, and Batch Normalization and Dropout are applied to the these layers to improve the learning rate and also reduce overfitting. We then flatten the input from the previous layer and apply 2 dense layers with the final layer with sigmoid activation function. 100 epochs were run with Adam optimizer as gradient descent technique and minimized the cross entropy loss.

Evaluation and Results

For evaluation metrics we chose to work with precision, recall and accuracy as these should be able to give enough information about the result of different sampling techniques and different models. All models are trained on GCP GPU V100.



```
Autoencoder:
([0.9105691313743591, 0.23489786684513092, 0.9140002727508545]

ANN:
Oversampling_random
([0.977642297744751, 0.8834951519966125, 0.9009901285171509],
Smote
([0.9766260385513306, 0.8791208863258362, 0.8695651888847351],
Adasyn
([0.9634146094322205, 0.7731092572212219, 0.9108911156654358],
Undersampling_random
([0.9563007950782776, 0.7377049326896667, 0.8910890817642212],
ClusterCentroids
([0.9329268336296082, 0.6223776340484619, 0.8811880946159363],
TomekLinks
([0.9847561120986938, 0.9638554453849792, 0.8695651888847351],

CNN:
Oversampling_random
([0.9806910753250122, 0.9101123809814453, 0.8804348111152649],
Smote
([0.977642297744751, 0.8888888955116272, 0.8695651888847351],
Adasyn
([0.9745935201644897, 0.8602150678634644, 0.8695651888847351],
Undersampling_random
([0.9695122241973877, 0.7924528121948242, 0.9130434989929199],
ClusterCentroids
([0.8211382031440735, 0.3359375, 0.9347826242446899], <keras.c
TomekLinks
([0.9847561120986938, 0.9753086566925049, 0.8586956262588501],
```

Fig 5. Accuracy, precision, recall results for different techniques

From the above results we can see that Tomek Links performed the best in terms of accuracy and precision but cluster centroids worked the best in terms of recall and the worst was tomek links in terms of recall and cluster centroids in terms of accuracy and autoencoder in terms of precision.

From the below graphs we can identify which of the methods works best for different performance measures. Accuracy shouldn't be considered as an important measure for this dataset due to the imbalance as the skew is too large, it is easy for it to learn the majority class and so it will predict majority class properly but we want to predict minority class. For this recall is a better option.

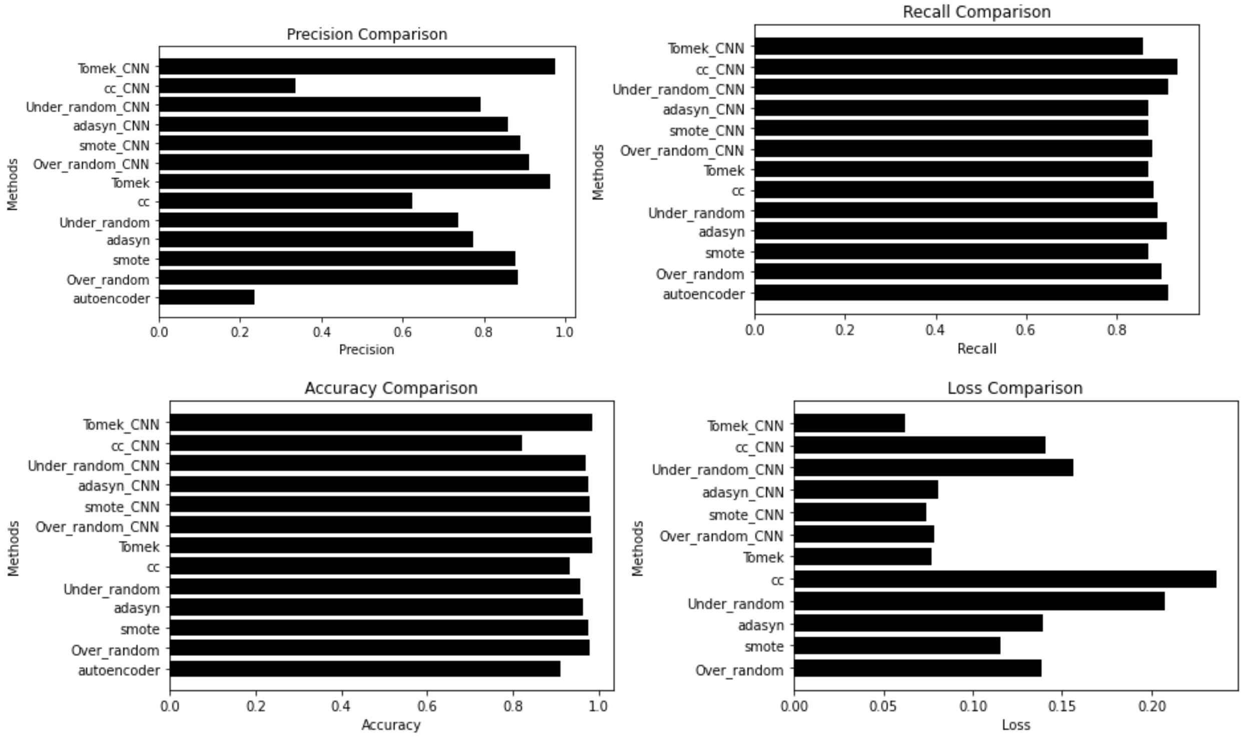


Fig 6. Performance measure comparisons

When recall is considered we find that CC with CNN and ADASYN with ANN performed better than the others which means higher chance of detecting fraud transaction.

Here cluster centroid recall is high mostly because the data is correctly spaced and the data points which are might lead to wrong classification are getting deleted as they are not that important. These data points are farthest from the centroid so these data points are really not important for classification. But here with CC precision is very low, so we can understand that this method classifies more fraudulent transactions than the other but at the same time the number of false positives are also more, that means it predicts more non-fraud transactions as fraud which is also a big issue because it is time consuming to check the transaction and also affects the customer.

On average the best working method is Tomek Link as it has good accuracy, precision, recall and low loss which means that it doesn't misclassify predictions as much as the others. Usually CNN is working better here than the other models and ANN has a higher loss than CNN. Undersampling techniques with CNN have a higher loss than the oversampling techniques with CNN.

Conclusion

Fraud detection is a very important topic which needs to be researched on because there are too many fraud transactions going on which affect thousands or millions of lives. Credit card fraud is the most common out of them. It is very challenging because of the imbalanced data. It's tough to work on that without handling the imbalance. On account of the drawbacks of traditional method, sampling algorithm with different neural network architectures can be used.

We compared the different sampling techniques with model architectures such as ANN, CNN, and auto encoders. We found that CNN performs overall better than ANN on most resampling methods except cluster centroids. We also find that for undersampling, performance: Tomek > random > cc and for oversampling, performance: random > smote > adasyn and for overall performance: Tomek > over random > smote > adasyn > under random > cc.

The best model for detecting as much fraud should be cc on CNN and the best model for overall performance should be Tomek. Several improvements can still be made by trying to improve the model architectures to better handle the imbalance and also data with different known columns could be used to get more information.

References

[1] Ng, A. Y., and Jordan, M. I., (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. Advances in neural information processing systems, 2, 841- 848.

[2] David Weston, David Hand, Niall Adams, Piotr Jusczak, Christopher Whitrow, (2008). Plastic card fraud detection using peer group analysis. Advances in Data Analysis and Classification, 2, 45-62.

[3] Ekrem Duman, Mehmet Hamdi Özçelik, (2011). Detecting credit card fraud by genetic algorithm and scatter search. Expert Systems with Applications.

[4] Bartosz Krawczyk, (2016). Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence**,** 5, 221–232.

[5] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen, “Effective detection of sophis- ticated Online banking fraud on extremely imbalanced data,” World Wide Web, vol. 16, no. 4, pp. 449–475, 2013.

[6] Mateusz Buda, Maciej Mazurowski, (2018).A systematic study of the class imbalance problem in convolutional neural networks. Neural Networks, 106.

[7] Kang Fu, Dawei Cheng, Yi Tu, Liqing Zhang, (2016). Credit Card Fraud Detection Using Convolutional Neural Networks. Neural Information Processing.

[8] Sara makki , Zainab assaghit, Yehia taher, Rafiqul haque et al, (2019). An Experimental Study With Imbalanced Classification Approaches for Credit Card Fraud Detection. Special section advanced software and data engineering for secure societies.