Credit Card Fraud Detection

Devadharshini Ayyappan
Electrical Engineering
NCSU
dayyapp@ncsu.edu

Kriti Singh
Electrical Engineering
NCSU
ksingh23@ncsu.edu

Sushanth Reddy Chilla
Electrical Engineering
NCSU
schilla@ncsu.edu

Sai Venkata Kaushik Pillalamarri
Electrical Engineering
NCSU
spillal2@ncsu.edu

I. Problem Statement

Credit card fraud is a significant problem for both individuals and organizations worldwide. This project aims to develop a credit card fraud detection system that can accurately detect fraudulent transactions while minimizing false negatives.The Dataset we intend to use for this project is taken from Kaggle[1]. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. To handle unbalanced dataset[2] used few resampling and synthetic data generation techniques. We plan to implement a few other resampling techniques (detailed in the later section) and compare their performances. Along with these we intend to use some feature selection methods along with Decision Tree classifier(DTC)[2].

II. Related Work

This [2] implements various techniques on an Imbalanced dataset. In this project we follow the methodologies followed in this paper and add few novel methods aswell.
This[5] uses bagging method to classify unbalanced data, we use this idea to implement our own bagging model which will be briefed in further sections.
These[3][4] describes the Oversampling techniques that we use in this project[6]. Explains the use of clustering for undersampling majority class in an unbalanced dataset. We implemented cnetroid clustering and compared performances.

[7]Implements Weighted cross entropy and focal loss for object identification using an imbalanced dataset, same logic is used for dealing with our imbalanced dataset

III. DATASET

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172 It contains only numerical input variables which are the result of a PCA transformation. Due to confidentiality issues, the original features and background information is not provided.Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which 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 dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise

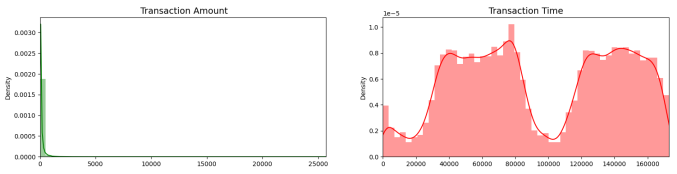


Fig. 1. Distribution of values in Transaction amount and Time

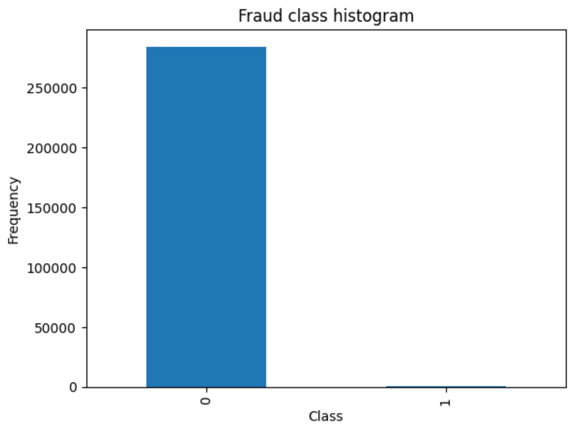


Fig. 2. Counts of each class before applying SMOTE

IV. DATA PROCESSING

We have 31 characteristics in our dataset.We processed the data by removing features which are redundant. With PCA's best features already given in the dataset,the 'time' feature is not useful for our problem statement and therefore has been removed. In addition,the data has been cleaned.Duplicate rows have been entirely removed.

V. DATA BALANCING TECHNIQUES

Undersampling and oversampling methods are popular for balancing a dataset.We have analyzed two oversampling algorithms : ADASYN[3] and SMOTE[4] and also analyzed Cluster Centroid Undersampling and RandomUndersampling with SMOTE.

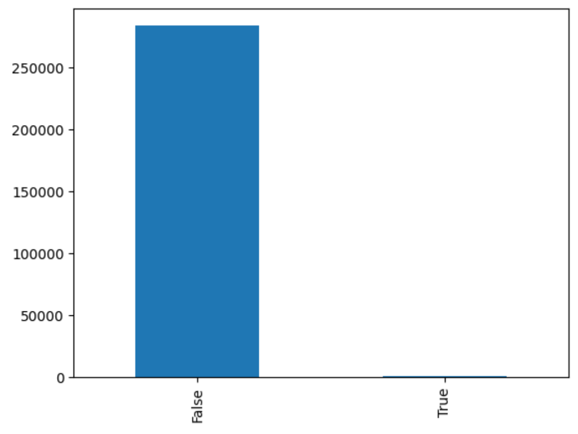


Fig. 3. Counts of duplicates in the dataset

A. SMOTE

SMOTE generates synthetic observations for the minority class. For a given minority class observation, synthetic observations are generated in a random range between the observation and its k-nearest minority class neighbors. First it finds the n-nearest neighbors in the minority class for each of the samples in the class . Then it draws a line between the the neighbors an generates random points on the lines.It finds the 5 nearest neighbors to the sample points and then draws a line to each of them.Then it creates samples on the lines with class == minority class.This procedure is done for every minority class observation.

B. ADASYN

ADASYN is an improved version of SMOTE.The procedure is same as SMOTE just with a minor improvement. After creating those samples it adds random small values to the points thus making it more realistic. In other words instead of all the sample being linearly correlated to the parent they have a little more variance in them i.e they are bit scattered.

C. Random Undersampling

The Random Undersampling involves randomly removing instances from the majority class to balance the class distribution with the minority class. The main idea behind random undersampling is to reduce the number of instances in the majority class so that it is comparable to the number of instances in the minority class. This can help prevent the machine learning model from being biased towards the majority class and improve its ability to predict instances from both classes.

D. Centroid Clustering

The Cluster Centroid Undersampling involves clustering the majority class instances and then replacing each cluster with its centroid. The resulting dataset has a reduced number of majority class instances while preserving the structure and distribution of the original data.

E. Balanced Cross Entropy[7]

To address class imbalance, a common approach is to use a weighting factor α that ranges from 0 to 1 for class 1 and 1-α for class 0. α can be determined by inverse class frequency or treated as a hyperparameter set through cross-validation. For convenience, α_t is defined similarly to pt. The α-balanced cross-entropy (CE) loss is denoted by

$$CE(p_t) = -\alpha_t log(pt) \tag{1}$$

and is considered as an experimental baseline for the proposed focal loss.

F. Focal Loss[7]

Easily classified negatives comprise the majority of the loss and dominate the gradient. While α balances the importance of positive/negative examples, it does not differentiate between easy/hard examples.Focal loss reshape's the loss function to down-weight easy examples and thus focus training on hard negatives.it add's a modulating factor $(1 - p_t)^\gamma$ to the cross entropy loss, with tunable focusing parameter $\gamma \geq 0$.

$$FL(pt) = -(1 - p_t)^\gamma log(p_t) \tag{2}$$

For phase 1, we sampled training data using ADASYN and SMOTE and analyzed the performance of the baseline classifiers, logistic regression, decision trees, and other baseline models. Now we show the performance of the oversampled data on XGboost, ensemble2 and esnemble3 models. We implemented the abovementioned undersampling techniques to overcome the challenge of overfitting the data we faced in phase 1. We used RandomUnderSampling with SMOTE for LR. We implemented CNN and ANN (i) without sampling and (ii) undersampled data using Centroid clustering and compared the performances. Performances using Weighted Cross entropy loss and Focal loss for CNN and ANN are also compared.

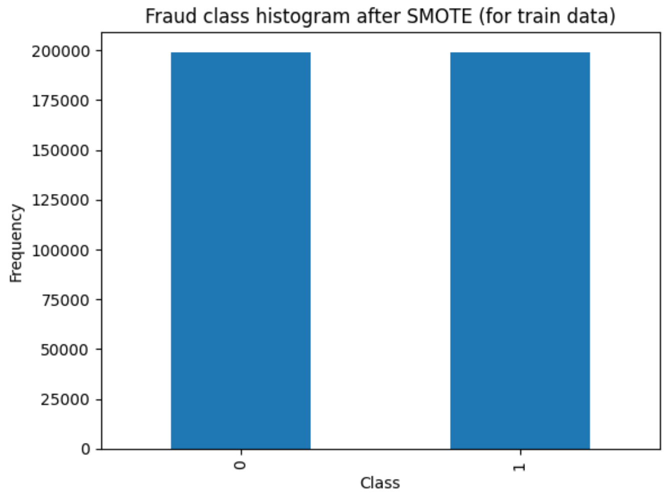


Fig. 4. Class counts after applying SMOTE

The oversampling techniques used in this study were SMOTE and ADASYN and we had also used undersampling for this dataset. The oversampling techniques aim to balance the class distribution by generating synthetic samples of the minority class. It was useful for this dataset as it had imbalanced classes, with only 0.17% of the data being fraudulent transactions. Oversampling helped balance the class distribution and improved the model's Recall. The Oversampling techniques were used on the baseline models, Logistic regression and Decision trees and the results were compared. We had also implemented a combination of oversampling and undersampling techniques on Logistic Regression models to choose the best parameters. From this, we chose the best parametric ratio of oversampling to undersampling and implemented the same on ANN and CNN models for classification.

Logistic regression was an appropriate choice for this dataset as it is a simple and interpretable model that works well with binary classification problems. It is also robust to noise in the data, making it suitable for the noisy credit card dataset.
Decision trees were also appropriate for this dataset as they can handle both categorical and numerical data and are easy to interpret. They are also useful in identifying the most informative features for classification.
This[5] implements a bagging model on weak classifiers to build a robust classifier for an Imbalanced dataset. In this project, we implemented a bagging model with the help of a decision tree stump with maxdepth=1 and used only two features sampled at random for each bagging instance. We use class 0 labels (split into batches of size 1000, one batch for each decision stump) and the entire class 1 training data to train each bagging instance. SMOTE is used to oversample class1 labels so that each decision tree is trained on data which has 50/50 ratio of each class.
We also implemented three different voting classifiers aggregating baseline classifiers(LR, DecisionTree Classifier, AdaBoost Classifier, GradientBoosting Classifier, KNeighbors Classifier, MLP Classifier) to approach higher accuracy for prediction. To further improve the model performance for the True Negatives, we implemented CNNs and ANNs on a balanced dataset created using undersampling, custom loss fucntions that takes class probability into account.

VI. Hypotheses

Given the very high class imbalance in the dataset, the use of oversampling and Undersampling techniques such as SMOTE, ADASYN and Random Undersampling would improve the performance of the baseline models in detecting credit card fraud. But given the number of features in the dataset, using more complex techniques such as CNNs and ANNs or other ensemble methods will perform better than a model implemented using logistic regression, decision trees or Random Forest techniques. The description for models tested and the results observed in related to the hypothesis are shown in the following sections.

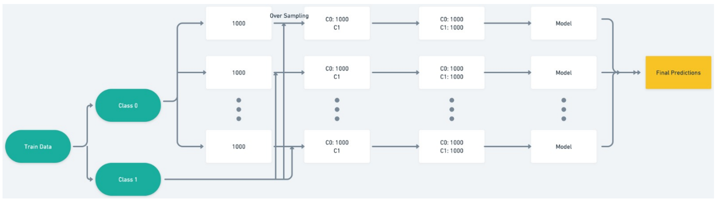


Fig. 5. Architecture used for Bagging

VII. Experimental Design

The dataset was first preprocessed by dropping the 'Time' and 'Amount' columns and scaling the 'Amount' column using StandardScaler. The data was then split into training, validation and test sets. The SMOTE and ADASYN resampling techniques were applied to the training set to address the issue of class imbalance. The Logistic Regression and Decision Tree models were implemented, and their performance was evaluated using the evaluation metrics of Recall, Precision, F1 Score, and PRAUC.

In the previous phase, the following steps were taken to perform the fraud detection task:
(i) The SMOTE and ADASYN resampling techniques were applied to the training set, and their performance was compared to the baseline models.
(ii) The baseline models were implemented and its performance was compared with each other.
(iii) We Implemented bagging method which utilizes decision stumps that are trained using only two features chosen at random. We use bootstrapping on our training data for each decision stump, which has 1000 data points of class0 and all the training data of class1. Then we perform oversampling on class1 data to make sure class's ratio is 50-50.
(iv) For the Stacking ensemble model we use 6 different baseline models. Table1 shows result's of 2 ensemble model each representing the metrics in case of tie break.

In this phase, we have improvised the fraud detection system by implementing ANNs and CNNs using a balanced dataset that was obtained using both oversampling and undersampling. The below steps were taken for the same:
(i) The combination of SMOTE and Undersampling techniques were applied to the training set, and their performance was compared to the experiments performed without oversampling the dataset.
(ii) The CNN and ANN models were implemented and its performance was compared with each other.
(iii) We trained and evaluated the model using different hyperparameters such as focal loss, custom loss using weighted loss functions.
(iv) Using feature selection techniques, we implemented CNN with the selected input features and compared the results with the previously trained CNN models.

All experiments were conducted using Python, and the software artifacts created include Python scripts for data preprocessing, model implementation, and performance evaluation.

VIII. MODELS IMPLEMENTED

For the first phase of the project we implemented our baseline classifiers like logistic regression and decision trees.

Logistic Regression: It is one of the supervised algorithms that is utilized for the purpose of classifying the dataset into distinct categories. The value of a categorical or numerical variable, dependent on each other, may be predicted with the help of this classifier. Logistic Regression is the method that has the capacity to categorize fresh data by making use of both continuous and discrete datasets at the same time. This ability is what gives the algorithm its name. Due to the fact that it possesses this quality, it is considered to be one of the essential machine learning algorithms. This classifier's primary function is to provide predictions on the probability associated with a variety of scenarios.

Decision tree classifiers: This classifier can be utilized for classification-based difficulties as well as regression-based issues; nevertheless, for the most part, it is recommended for classification applications. The structure of this classifier resembles a tree, with the core nodes representing the attributes of the dataset, the branches representing the decision rules, and the leaf nodes of the tree representing the final outputs. Therefore, we can also say that this classifier gives a graphical method for finding all of the potential answers to a specific problem based on the conditions that have been provided. When the size of the tree increases, it also indicates that the tree is becoming deeper, which means that more conventional decision rules and models will fit more precisely.

We used gridsearch to select the best hyperparameters for training on data sampled using ADASYN and SMOTE and compared their performance with relevant metrics.

Ensemble-Bagging: Decision tree classifiers are high-variance models. It means a slight change in the training data, will lead to an entirely different model. Decision trees usually overfit. To overcome this situation, ensemble technique — bagging can be used. This is a variant of Random Forest where decision stumps were trained on parts of the whole dataset, all of the stumps were underfit to the actual data as most of them only learned about a few examples belongning to only two of the given features. This can be seen in fig(5).

Ensemble model- Stacking: this method involved stacking of different models which were trained with the help of algorithms such as logistic regression, KNN, GradientBoosting, Decision tree, and AdaBoost, XGboost. All the models were separately trained using the whole of the training data without any oversampling technique, and later were trained twice on training data which was oversampled once with SMOTE and later with ADASYN. The initial predictions consisted of the predictions made by each model, and the final predictions consisted of predictions which got the majority votes for that particular test set example. The same can be visually observed in fig(6)

To further improvise the performance, we implemented ANNs and CNNs with different hyperparameters. The model architecture that we used is explained below:

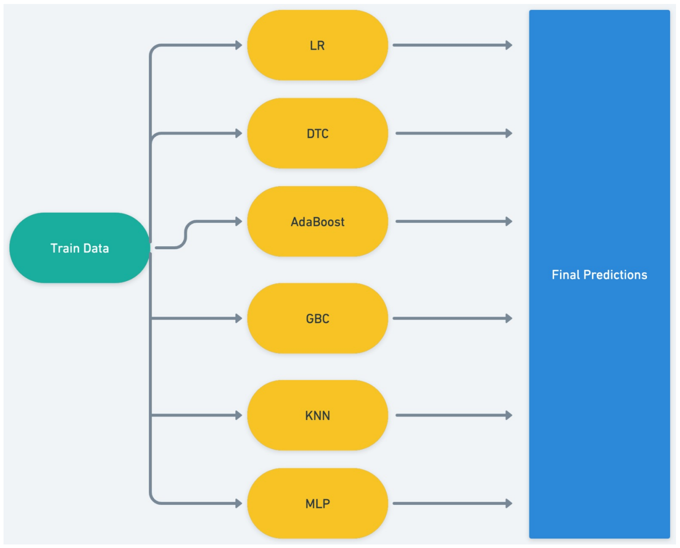
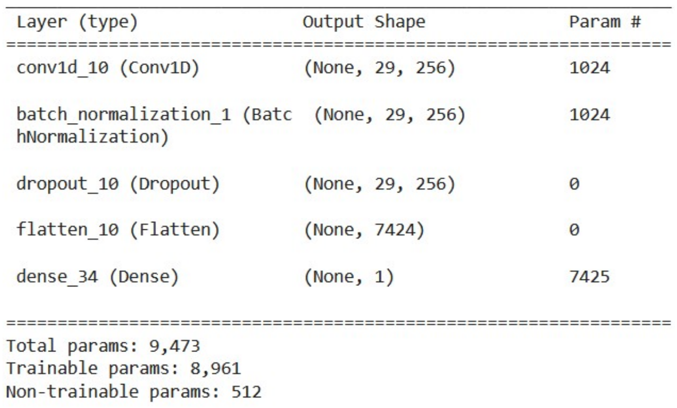


Fig. 6. Ensemble Model Architecture

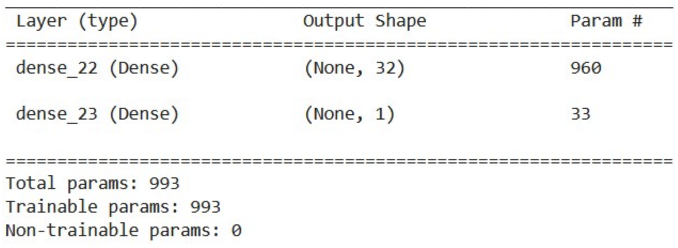
CNN Model: The model that we have implemented has five layers: one Conv1D layer, one BatchNormalization layer, one Dropout layer, one Flatten layer, and one Dense layer. The first layer is a Conv1D layer with 256 output filters and a kernel size of 3. The BatchNormalization layer normalizes the outputs of the previous layer to have zero mean and unit variance. The third layer is a Dropout layer, which randomly sets a fraction of the input units to zero during training, to prevent overfitting. The fourth layer is a Flatten layer, which flattens the output of the previous layer into a 1D array. The fifth and final layer is a Dense layer with one output unit, which is used for binary classification. This layer uses the sigmoid activation function, which outputs a value between 0 and 1, indicating the probability of the input belonging to the positive class which is the non-fraud class (0) in our experiment. The model has a total of 9,473 parameters, out of which 8,961 are trainable parameters that will be updated during the training process, while 512 are non-trainable parameters used by the BatchNormalization layer.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_10 (Conv1D)           (None, 29, 256)           1024

batch_normalization_1 (Batc  (None, 29, 256)           1024
hNormalization)

dropout_10 (Dropout)         (None, 29, 256)           0

flatten_10 (Flatten)         (None, 7424)              0

dense_34 (Dense)             (None, 1)                 7425

=================================================================
Total params: 9,473
Trainable params: 8,961
Non-trainable params: 512
```

Fig. 7. CNN Model Architecture

ANN Model: The ANN model has two dense layers. The first layer is a Dense layer with 32 output units, using the ReLU (Rectified Linear Unit) activation function.The second and final layer is also a Dense layer, but with a single output unit, using the sigmoid activation function, which is a common choice for binary classification tasks.



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_22 (Dense)             (None, 32)                960

dense_23 (Dense)             (None, 1)                 33

=================================================================
Total params: 993
Trainable params: 993
Non-trainable params: 0
```

Fig. 8. ANN Model Architecture

IX. Phase I Results

The results from the Phase I experiments are given below which involved the implementation of Ensemble Architectures and the baseline models. This has been depicted clearly in the TABLE I, II and III.
Ensemble1 represents the tie-break case1 : predicting as 0 in the case of tiebreak
Ensemble2 represents the tiebreak case2 : predicting as 1 in the case of tiebreak

TABLE I
Results For class 1 on SMOTE

Classifier	F1	Recall	Precision
LR + SMOTE	0.12	0.92	0.12
GBC + SMOTE	0.25	0.89	0.15
DTree + SMOTE	0.53	0.78	0.4
AdaBoost + SMOTE	0.14	0.88	0.08
KNN + SMOTE	0.64	0.86	0.51
MLP+SMOTE	0.77	0.79	0.75
Ensemble1 + SMOTE	0.74	0.86	0.65
Ensemble2 + SMOTE	0.40	0.88	0.20
Bagging + SMOTE	0.81	0.82	0.82

TABLE II
Results For class 1 without oversampling

Classifier	F1	Recall	Precision
LR	0.64	0.66	0.62
GBC	0.32	0.22	0.57
DTree	0.66	0.67	0.65
AdaBoost	0.68	0.61	0.78
KNN	0.07	0.04	1
MLP	0.5	0.76	0.37
Ensemble	0.71	0.56	0.91
Bagging	0.64	0.53	0.83

X. Phase II Results

For phase two, we used Cluster Centroid Undersampling to balance the dataset and trained the Neural Networks and

TABLE III
Results For class 1 on ADASYN

Classifier	F1	Recall	Precision
LR + ADASYN	0.04	0.95	0.02
DTree + ADASYN	0.52	0.76	0.39
Bagging + ADASYN	0.77	0.73	0.81

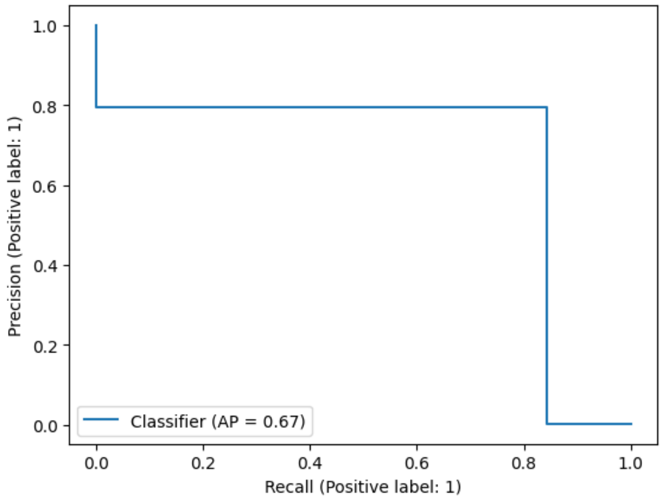


Fig. 9. PRAUC curve for Ensemble1+ smote model

also, used a mixture of smote oversampling with random undersampling to train the Neural Network and analyzed the results from both.

In the first part, we trained the neural networks with different hyperparameters on the unsampled datasets but the dublicates were removed from it. The results have been tabulated for reference in Table IV

TABLE IV
Results of Unsampled Data

Model + Optimizer	F1	Recall	Precision
CNN + ADAM	0.71	0.6	0.87
CNN + RMSPROP	0.78	0.77	0.8
ANN +RMSPROP	0.74	0.68	0.81
CNN + Weighted CrossEntropy + RMSPROP	0.73	0.78	0.68

In the second part, we experimented with different ratios of smote oversampling and random undersampling and analysed the effect of oversampling and undersampling on performance of logistic regression models to come up with fixed ratio of undersampling and oversampling.The sklearn library provides framework for smote and undersampling where parameter sampling strategy controls the amount of sampling to be done with respect to the target class. The results for the same has been summarized in TABLE VI. The values for Random Undersampling and Smote oversampling was taken as 0.3 and 0.1 respectively from the table as the false positive is lowest in this case, meaning the detection of the fraud transactions are higher. We use this ratio and implement deep learning methods on the transformed data and compare the performance with training the deep learning model without transforming data.

TABLE V
RESULTS USING SMOTE

Model + Optimizer	F1	Recall	Precision	PR-AUC
XGBoost	0.8	0.84	0.76	0.6
Ensemble3	0.82	0.84	0.79	0.67

TABLE VI
RESULTS FOR LOGISTIC REGRESSION WITH SMOTE AND UNDERSAMPLING

Smote	Random Undersampling	FN	FP
0.1	0.5	19	1096
0.3	0.5	19	1121
0.5	0.5	19	1077
0.5	0.6	22	2070
0.5	0.8	19	1383
0.1	0.3	21	827
0.1	0.4	20	956
0.3	0.3	21	857

With the selected sampling rate in the dataset, the neural network was trained with different hyperparameters and the corresponding results have been tabulated in TABLE VII.

TABLE VII
RESULTS OF RANDOM UNDERSAMPLED AND SMOTE OVERSAMPLED DATA

Model + Optimizer	F1	Recall	Precision
CNN + ADAM	0.38	0.87	0.24
CNN + RMSPROP	0.41	0.85	0.27
ANN +RMSPROP	0.43	0.89	0.28
CNN + Weighted CE + RMSPROP	0.16	0.9	0.09

TABLE VIII
RESULTS WITHOUT SAMPLING THE DATA FOR FEATURE SELECTION

Model + Optimizer	F1	Recall	Precision	PR-AUC
CNN + RMSPROP	0.75	0.69	0.83	0.58
ANN +RMSPROP	0.81	0.78	0.84	0.65

We also tried Centroid Clustering sampling on the dataset as the third experiment and trained the neural networks. The results for the same has been given in Table IX and Table X.

Feature Selection: From the Fig.10, we can also observe that not all columns have a variance greater than 1. So, we also trained the CNNs using the selected features based on the variance distributions by taking the features with high variance. The results for the same has been given in the tables VIII, X, XI and XIII.

XI. INTERPRETATION AND ANALYSIS OF RESULTS

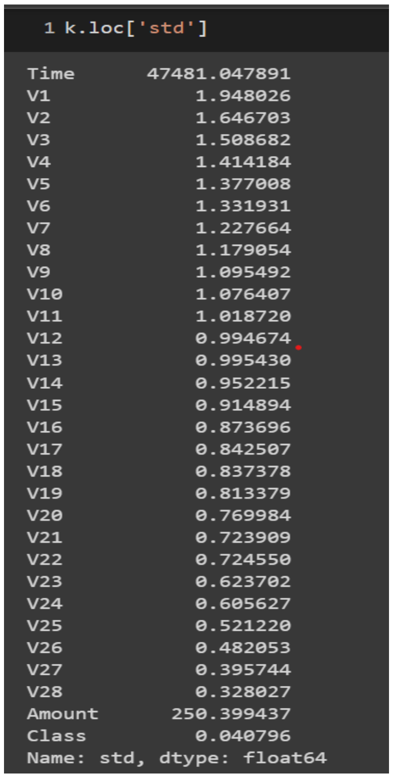
We used Recall and PRAUC to evaluate the performance of models. Among all the implemented models, LR had the highest Recall. As it is a simple model, it couldn't generalize,

TABLE IX
RESULTS WITH CLUSTERING UNDERSAMPLING TTECHNIQUE

Model + Optimizer	F1	Recall	Precision	PR-AUC
CNN + RMSPROP	0.75	0.8	0.78	0.77
ANN +RMSPROP	0.84	0.79	0.82	0.81

TABLE X
UNDERSAMPLING USING CENTROID CLUSTERING FOR FEATURE SELECTION

Model + Optimizer	F1	Recall	Precision	PR-AUC
CNN + RMSPROP	0.76	0.7	0.82	0.69
ANN +RMSPROP	0.79	0.76	0.82	0.72



```
1 k.loc['std']

Time      47481.047891
V1            1.948026
V2            1.646703
V3            1.508682
V4            1.414184
V5            1.377008
V6            1.331931
V7            1.227664
V8            1.179054
V9            1.095492
V10           1.076407
V11           1.018720
V12           0.994674
V13           0.995430
V14           0.952215
V15           0.914894
V16           0.873696
V17           0.842507
V18           0.837378
V19           0.813379
V20           0.769984
V21           0.723909
V22           0.724550
V23           0.623702
V24           0.605627
V25           0.521220
V26           0.482053
V27           0.395744
V28           0.328027
Amount      250.399437
Class         0.040796
Name: std, dtype: float64
```

Fig. 10. Variance of independent attributes

thus Precision took a huge hit. Considering the Precision-Recall Area under the curve, the unsampled ANN model and undersampled ANN model had the best performance, but the clustering-based undersampled ANN model had higher Recall than the unsampled ANN model even though they had similar PR-AUC values. If Precision and Recall are given equal importance, then Bagging +SMOTE, which scored 0.82 precision and 0.82 recall, is the highest among all the implemented models. The feature selection classifier trained on clustering-based undersampled data using Focal loss performed equally well, achieving 0.79 Precision, 0.79 Recall and 0.79 PR-AUC.

XII. CONCLUSION

The ANN model performed best with undersampling, while the clustering-based undersampled ANN model had higher

TABLE XI
WEIGHTED CROSSENTROPY LOSS FOR FEATURE SELECTION

Model + Optimizer	F1	Recall	Precision	PR-AUC
CNN + RMSPROP	0.75	0.72	0.79	0.73
ANN +RMSPROP	0.77	0.78	0.76	0.71

TABLE XII
FOCAL LOSS FOR UNSAMPLED

Model + Optimizer	F1	Recall	Precision	PR-AUC
CNN + RMSPROP	0.82	0.78	0.8	0.65
ANN + RMSPROP	0.84	0.78	0.76	0.66

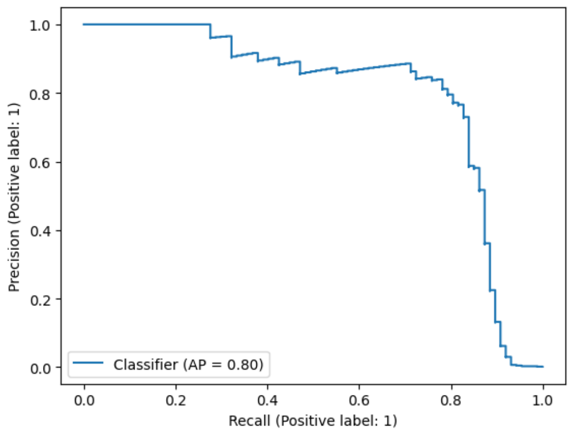
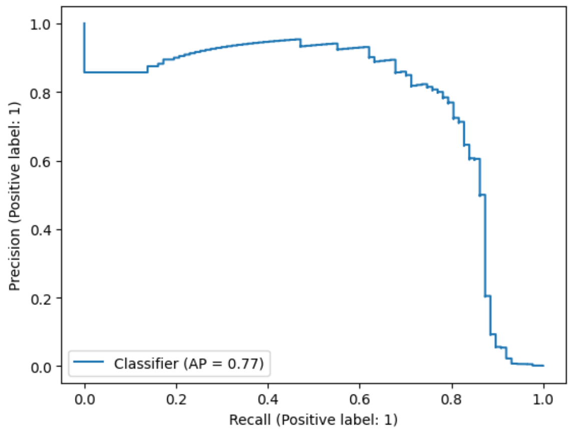


Fig. 11. PRAUC curve for Clustering based undersampling with ANN



]

Fig. 12. PRAUC curve for Clustering based undersampling with CNN

recall. The Ensemble classifier performed well if Precision and Recall are given equal importance. Finally, the feature selection classifier trained on clustering-based undersampled data using Focal loss performed equally well with high values of Precision, Recall, and PRAUC.

We observed that since the dataset is hugely imbalanced, applying smote on the entire dataset would lead to overfitting the data on positive samples, so oversampling the minority class to a particular ratio gives better results. As part of the mid-way report, we concluded that the Bagging ensemble model performed better than baseline models based on the PRAUC curve. After implementing NN-based models we conclude that these architectures learn complex patterns present in the dataset, thus resulting in higher PR-AUC than the bagging model.

TABLE XIII
FOCAL LOSS FOR FEATURE SELECTION

Model + Optimizer	F1	Recall	Precision	PR-AUC
CNN+ RMSPROP	0.79	0.8	0.77	0.78
ANN + RMSPROP	0.79	0.79	0.79	0.79

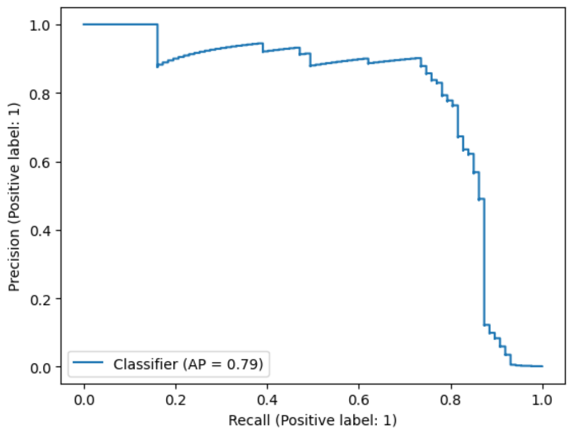


Fig. 13. PRAUC curve for Clustering based undersampling + focal loss with ANN

REFERENCES

[1] Machine Learning Group - ULB (Owner) - https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
[2] Yuru Jing, Machine Learning Performance Analysis to Predict Stroke Based on Imbalanced Medical Dataset, https://doi.org/10.48550/arXiv.2211.07652.
[3] Haibo He, Yang Bai, Edwardo A. Garcia, Shutao Li, ADASYN: Adaptive synthetic sampling approach for imbalanced learning, 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence).
[4] SMOTE: Synthetic Minority Over-sampling Technique N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, arXiv:1106.1813
[5] Solving the class imbalance problem using ensemble algorithm: application of screening for aortic dissection Lijue Liu,1,2 Xiaoyu Wu,1 Shihao Li,1 Yi Li,corresponding author1,2 Shiyang Tan,1 and Yongping Bai3
[6] Clustering-based undersampling in class-imbalanced data Lin Wei-Chao , Tsai Chih-Fong , Hu Ya-Han , Jhang Jing-Shang
[7] Focal Loss for Dense Object Detection Tsung-Yi Lin Priya Goyal Ross Girshick Kaiming He Piotr Dollar´Facebook AI Research (FAIR)

XIII. SOURCE CODE

The source code for this project is availabe at : https://github.com/psvkaushik/Credit_Card_Fraud_Detection