Credit Card Fraud Detection using Machine Learning

A Comprehensive Study

Aditya Singh

15 July 2024

Introduction

According to the annual FTC report, Over 8,53,935 cases of credit card fraud were reported 21% of which reported a dollar loss of over 2.7 Billion. Of the over 2.5 million fraud reports, 27% indicated that money was lost. In 2023, people reported losing over $10 billion to fraud, marking an increase of over $1 billion compared to 2022. Notably, more than $4 billion of these losses were due to investment-related scams. Bank transfers and payments accounted for the highest aggregate losses reported in 2023, totalling $1.86 billion. This was closely followed by cryptocurrency losses, which amounted to $1.41 billion. Despite these figures, credit cards were the most frequently identified payment method in fraud [1].

Machine learning are very effective in detecting fraudulent credit card transactions as compared to traditional rule based learning techniques owing to their ability to adapt to historical data and considering non-linear relationships in data to correctly classify transactions. In this study we will utilise supervised machine learning algorithms to classify credit card dataset.

Literature Review

Numerous studies such as presented by Wen-Fang YU and Na Wang used Outlier mining, Outlier detection mining and Distance sum algorithms to accurately predict fraudulent transactions. Similarly GJUS&T at Hisar HCE presented techniques like Supervised and Unsupervised learning for credit card fraud detection. However these failed to provide a consistent solution to detection of fraudulent transactions. Maniraj et.al. utilised various supervised learning algorithms to classify a large imbalanced dataset, however the approach led to low F1 scores on the results [2] while Nguyen et al. implemented deep learning methods such as CNN's and LSTM's on Credit card data to achieve an F1 score of 84.85% by utilising synthetic data augmentation techniques such as SMOTE [3].

Aim

The previous works on this topic indicate that there is an absence of machine learning algorithms to detect frauds. Here we will try to present data augmentation as well as supervised machine learning models such as Ensemble Learning and

SVM's to try to create a model which is less sensitive to the inherent class imbalance present in the dataset and utilises less computational resources to train as compared to deep learning architectures as LSTM's and 1-D CNN's.
Hence we will have the following objectives for the study:-

i) Understand and Transform the Data.
ii) Train and compare various models on the Data
iii) Utilise tools to fix imbalance in the Data.

Dataset

For this study, we will be utilising the Credit Card Fraud Detection dataset available on Kaggle. This data consists of transactions made by credit cards in September 2013 by European cardholders 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.
It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features

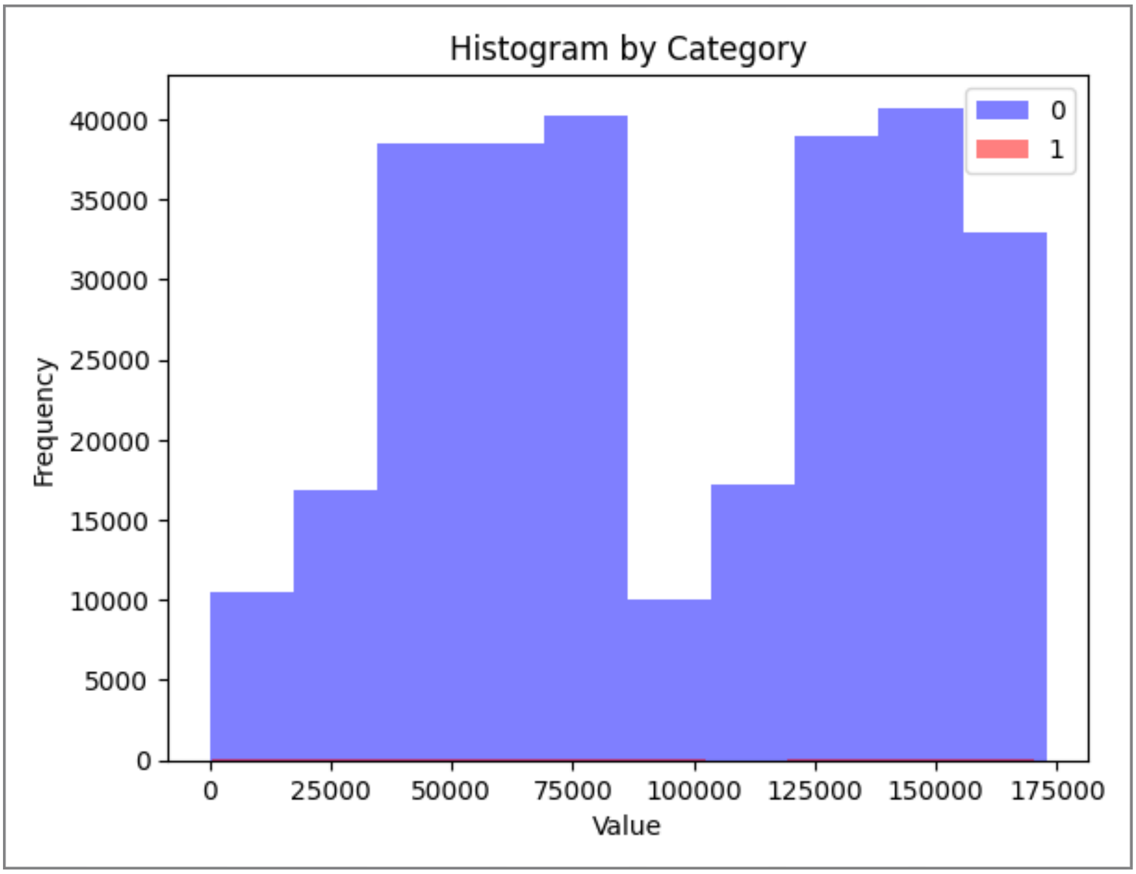
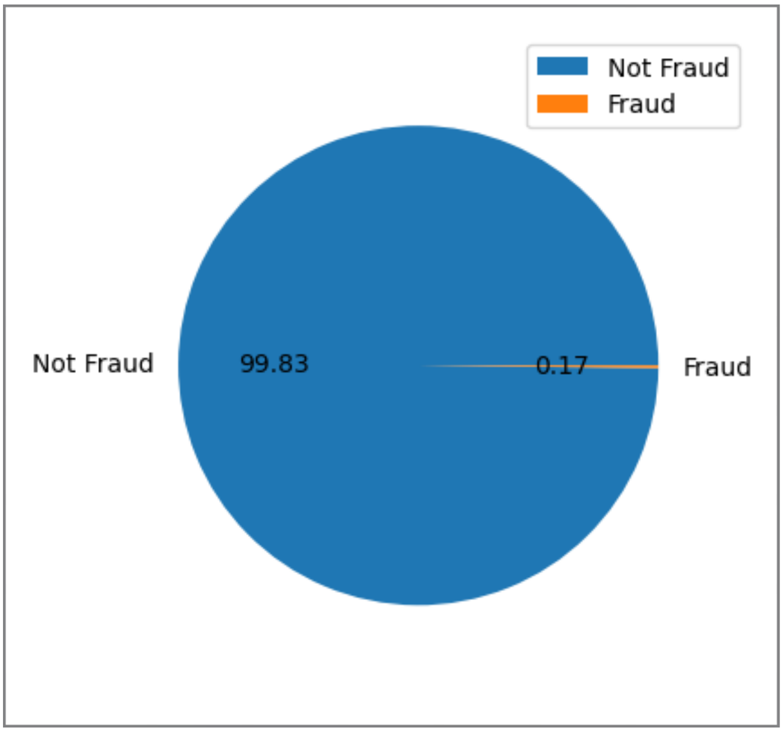


Fig. 1 Distribution of Transactions

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 [4].



This shows the extreme skew in the data due to less instances of fraud. To tackle this we may use downsampling techniques or generate synthetic data for Fraud cases using techniques such as SMOTE to decrease this skew.

Due to less no. of Fraud Instances, we will focus on F1 score and recall for model evaluation metrics.

Fig.2 Class Distribution of Data

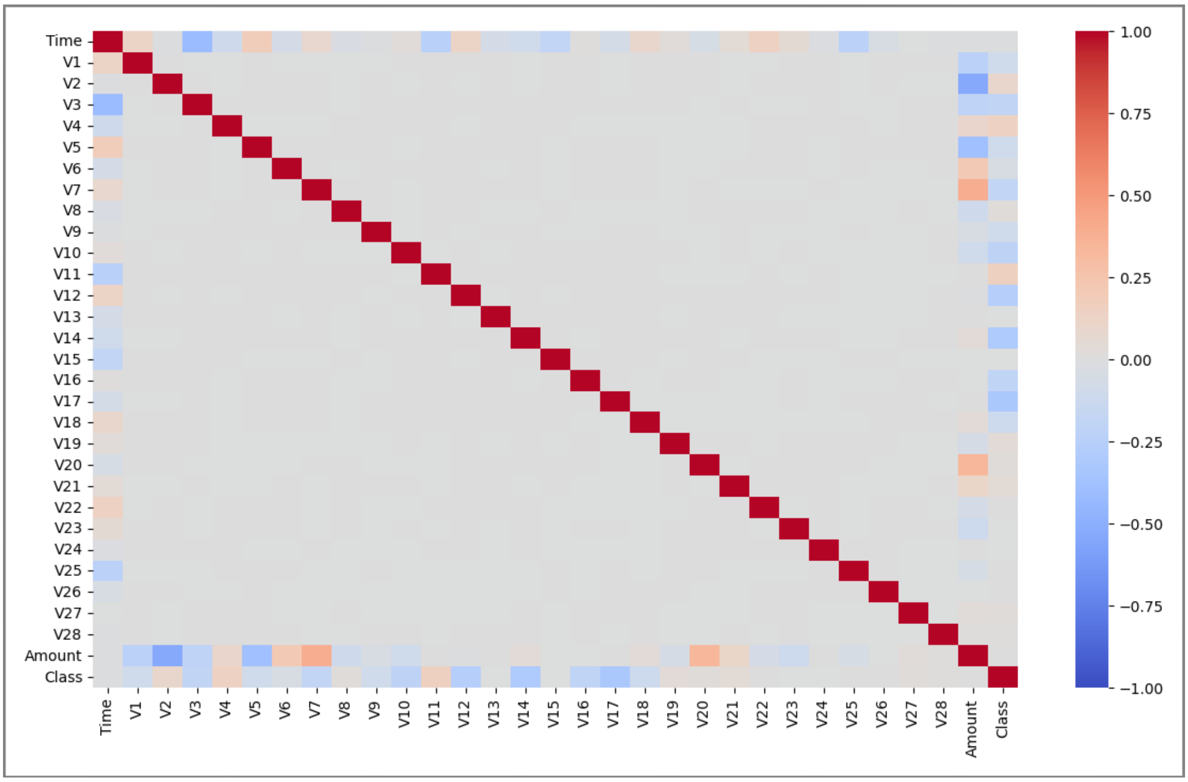


Fig. 3 Correlation Heat map

The individual PCA features are weakly co-related with the feature Class except for a few features such as V17, V12, V14 and V10 as visible above.

Modelling

To better train ML algorithms on the data, we first need to preprocess and scale these columns, specifically :-

a) Amount : Robust Scaling to handle outliers present in the data.

b) Time : Min-Max Scaling to convert large values of the variable.

```
from sklearn.preprocessing import RobustScaler , StandardScaler, MinMaxScaler

cleaned_df = data.copy()
cleaned_df['Amount'] = RobustScaler().fit_transform(cleaned_df['Amount'].to_numpy().reshape((-1,1)))
cleaned_df['Time'] = MinMaxScaler().fit_transform(cleaned_df['Time'].to_numpy().reshape((-1,1)))
```

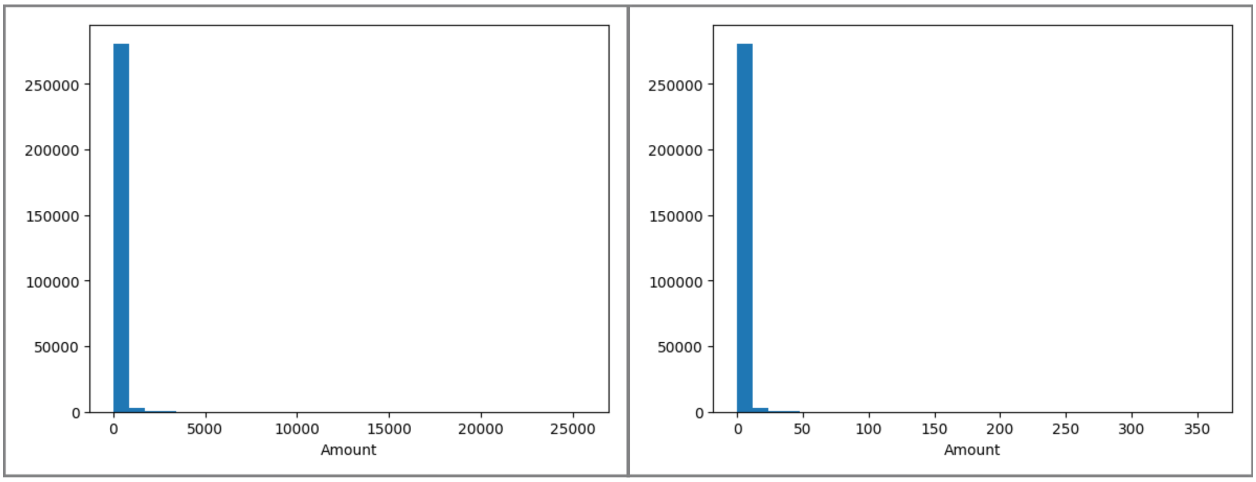


Fig. 4(a) Unscaled Amount

Fig. 4(b) Scaled Amount

After scaling, we split the data into train, test and validation splits utilising 80% data for training, 15% for test and 5% for validation.

a) Support Vector Machine Classifier :

Support Vector Machine (SVM) is a powerful machine learning algorithm which can be used for linear or nonlinear classification, regression, and even outlier detection tasks. Since the data is separated into PCA components, SVM can perform well on this data.

Here we are able to achieve an F1 Score of 86% on validation and 80% on testing which is a reasonable score for the classifier.

```
from sklearn.svm import LinearSVC
svc = LinearSVC(class_weight='balanced')
svc.fit(x_train, y_train)
print(classification_report(y_val, svc.predict(x_val), target_names=['Not Fraud', 'Fraud']))
```

```
              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00      4803
       Fraud       1.00      0.75      0.86         4

    accuracy                           1.00      4807
   macro avg       1.00      0.88      0.93      4807
weighted avg       1.00      1.00      1.00      4807
```

b) Random Forest Classifier :

Random Forest is a tree like algorithm which works by creating a number of Decision Trees during the training. Each tree is made using a random subset of the data to measure a random subset of features in each partition. This randomness introduces variability among individual trees, making it less prone to overfitting and improving overall prediction performance of the model.

The resilience to overfitting makes Random Forest classifier ideal for this task.

We are using the Snap ML Library developed by IBM to train the models faster.

```
from snapml import RandomForestClassifier as SnapRandomForestClassifier
model = SnapRandomForestClassifier(max_depth=3, n_estimators=100, n_jobs=4, random_state=42, use_histograms=True)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(classification_report(y_test, y_pred, target_names=['Not Fraud', 'Fraud']))
```

```
              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00     39949
       Fraud       0.97      0.57      0.72        51

    accuracy                           1.00     40000
   macro avg       0.98      0.78      0.86     40000
weighted avg       1.00      1.00      1.00     40000
```

This achieves a F1 score of 72% on testing.

c) Gradient Boosting :

Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimise the loss function such as mean absolute error of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimise this gradient.

```
gbc_b = GradientBoostingClassifier(n_estimators=50, learning_rate=1.0, max_depth=2, random_state=0)
gbc_b.fit(x_train_b, y_train_b)
print(classification_report(y_val_b, gbc.predict(x_val_b), target_names=['Not Fraud', 'Fraud']))
```

```
              precision    recall  f1-score   support

   Not Fraud       0.81      1.00      0.89        72
       Fraud       1.00      0.76      0.86        70

    accuracy                           0.88       142
   macro avg       0.90      0.88      0.88       142
weighted avg       0.90      0.88      0.88       142
```

This achieves an F1 Score of 86% on validation.

d) Artificial neural networks :

While not a Machine learning technique, ANN can be used in this example as plenty of data is available for training and regularisation techniques can be used in the model to prevent overfitting to handle the imbalanced dataset.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import InputLayer, Dense, BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint

shallow_nn = Sequential()
shallow_nn.add(InputLayer((x_train.shape[1],)))
shallow_nn.add(Dense(10, 'relu'))
shallow_nn.add(BatchNormalization())
shallow_nn.add(Dense(2, 'relu'))
shallow_nn.add(BatchNormalization())
shallow_nn.add(Dense(1, 'sigmoid'))

checkpoint = ModelCheckpoint('shallow_nn', save_best_only=True)
shallow_nn.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
with tf.device('/device:GPU:0'):
  shallow_nn.fit(x_train, y_train, validation_data=(x_val, y_val), epochs=5, callbacks=checkpoint)
```

```
Epoch 1/5
8125/8125 [==============================] - 31s 4ms/step - loss: 0.0284 - accuracy: 0.9928 - val_loss: 0.0021 - val_accuracy: 0.9998
Epoch 2/5
8125/8125 [==============================] - 30s 4ms/step - loss: 0.0033 - accuracy: 0.9994 - val_loss: 0.0022 - val_accuracy: 0.9998
Epoch 3/5
8125/8125 [==============================] - 31s 4ms/step - loss: 0.0032 - accuracy: 0.9993 - val_loss: 0.0019 - val_accuracy: 0.9998
Epoch 4/5
8125/8125 [==============================] - 29s 4ms/step - loss: 0.0030 - accuracy: 0.9994 - val_loss: 0.0019 - val_accuracy: 0.9998
Epoch 5/5
8125/8125 [==============================] - 28s 3ms/step - loss: 0.0029 - accuracy: 0.9994 - val_loss: 0.0019 - val_accuracy: 0.9998
```

```
def neural_net_predictions(model, x):
  return (model.predict(x).flatten() > 0.5).astype(int)
neural_net_predictions(shallow_nn, x_val)
```

```
151/151 [==============================] - 1s 6ms/step
array([0, 0, 0, ..., 0, 0, 0])
```

```
print(classification_report(y_val, neural_net_predictions(shallow_nn, x_val), target_names=['Not Fraud', 'Fraud']))
```

```
151/151 [==============================] - 1s 4ms/step
              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00      4803
       Fraud       1.00      0.75      0.86         4

    accuracy                           1.00      4807
   macro avg       1.00      0.88      0.93      4807
weighted avg       1.00      1.00      1.00      4807
```

This approach provides an F1 score of 86% on validation data.

Data Augmentation

Since the imbalance in the dataset is extreme, it is better to use sampling techniques to improve model performance on the data. To achieve this two multiple methods can be used :

i) Oversampling : Duplicating minority data to match the majority class.

ii) Undersampling : Systematically scaling down majority class.

iii) Synthetic Data : Techniques like SMOTE to generate additional data.

Synthetic Minority Oversampling Technique (SMOTE) is a sampling technique specifically designed to tackle imbalanced datasets by generating synthetic samples for the minority class which is helpful in dealing with class imbalance, and is known for its application in improving the performance of classifier models [5].

```
from imblearn.over_sampling import SMOTE
from collections import Counter
counter = Counter(y_train)
print('Before', counter)
smtom = SMOTE(sampling_strategy = 'minority')
X_train_smtom, y_train_smtom = smtom.fit_resample(x_train[:40000], y_train[:40000])
counter = Counter(y_train_smtom)
print('After', counter)
```

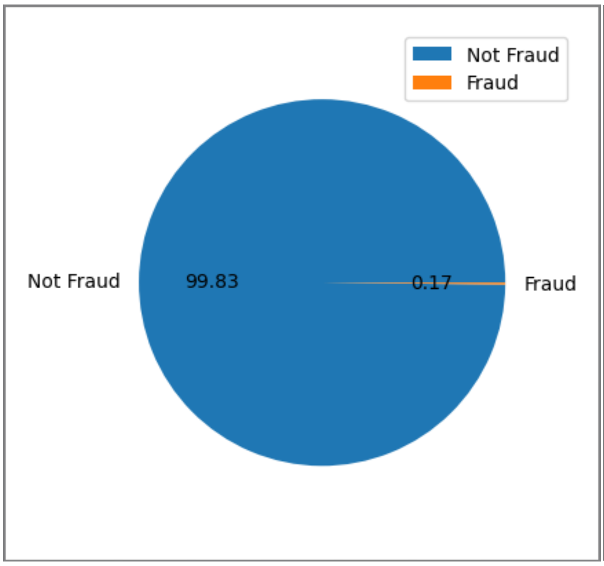
```
Before Counter({0: 259529, 1: 471})
After Counter({0: 39896, 1: 39896})
```

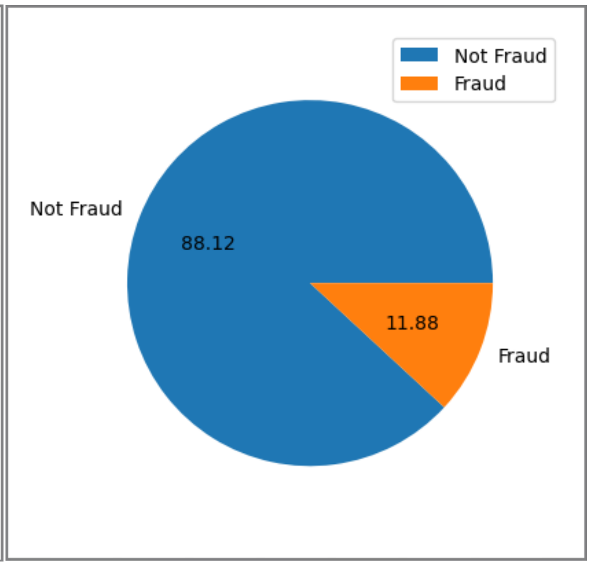
```
X_train = np.concatenate((x_train, X_train_smtom), axis=0)
Y_train = np.concatenate((y_train, y_train_smtom), axis=0)
```

```
Counter(Y_train)
```

```
Counter({0: 299425, 1: 40367})
```

By utilising SMOTE, we have generated 40,000 additional examples of Fraud in the train dataset to help models learn better by significantly increasing the number of Fraud class labels.





Now we can retrain the models for checking accuracy on the augmented dataset.

a) Augmented Support Vector Machine :

```
from sklearn.metrics import classification_report
print(classification_report(y_val, svc.predict(x_val), target_names=['Not Fraud', 'Fraud']))
```

```
              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00     22789
       Fraud       0.33      0.39      0.36        18

    accuracy                           1.00     22807
   macro avg       0.67      0.69      0.68     22807
weighted avg       1.00      1.00      1.00     22807
```

The F1 Score of SVM classifier dropped significantly after SMOTE due to tendency of SMOTE to distort clear decision boundaries.

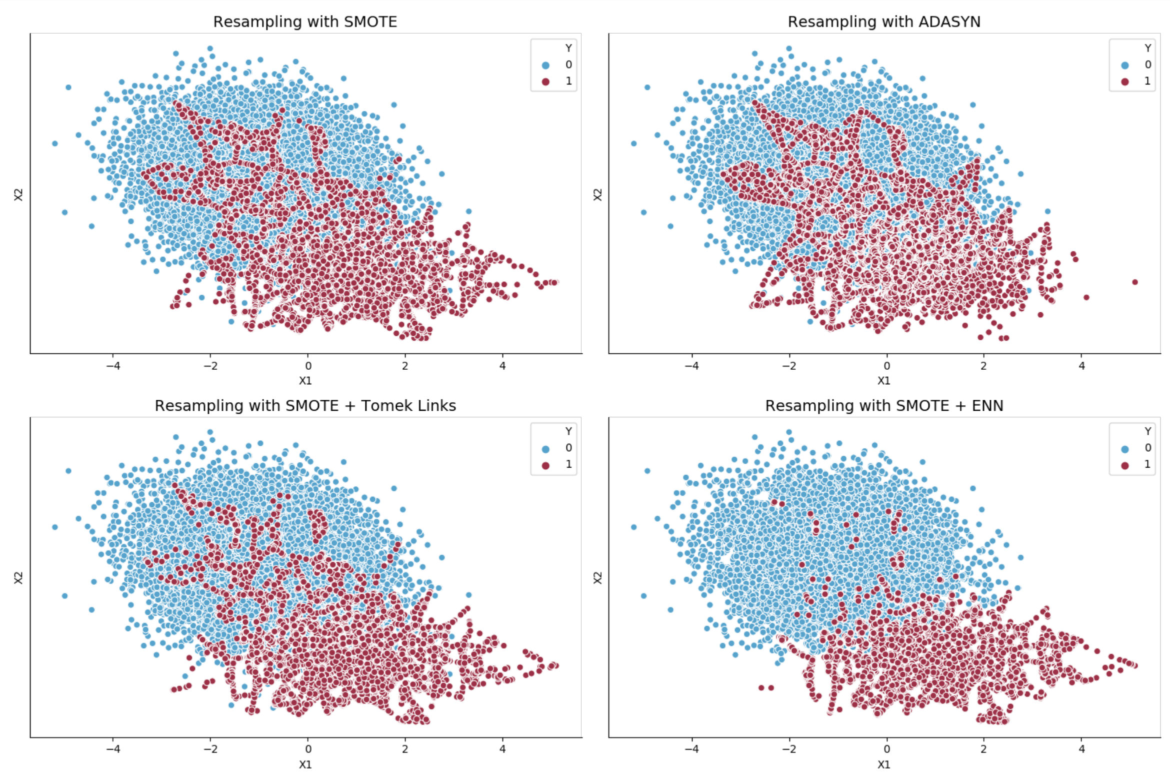


Fig. 5 Impact of SMOTE techniques on decision boundaries

b) Augmented Random Forest Classifier :

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(max_depth=2, n_jobs=-1)
rf.fit(x_train, y_train)
print(classification_report(y_val, rf.predict(x_val), target_names=['Not Fraud', 'Fraud']))
```

```
              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00     22789
       Fraud       0.92      0.61      0.73        18

    accuracy                           1.00     22807
   macro avg       0.96      0.81      0.87     22807
weighted avg       1.00      1.00      1.00     22807
```

The F1 score of Random forest algorithm improved marginally while its recall improved significantly to 61%. The Increments are small as Random Forest model is already quite resilient towards imbalanced datasets.

c) Augmented Gradient Boosting :

```
from sklearn.ensemble import GradientBoostingClassifier
gbc = GradientBoostingClassifier(n_estimators=50, learning_rate=1.0, max_depth=1, random_state=0)
gbc.fit(x_train, y_train)
print(classification_report(y_val, gbc.predict(x_val), target_names=['Not Fraud', 'Fraud']))
```

```
              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00     22789
       Fraud       0.60      0.50      0.55        18

    accuracy                           1.00     22807
   macro avg       0.80      0.75      0.77     22807
weighted avg       1.00      1.00      1.00     22807
```

The F1 score incase for Gradient boosting Tree Classifier dropped significantly due to SMOTE augmentations tendency to invade other clusters making designs boundaries distorted which causes overfitting.

d) Augmented Artificial neural networks :

```
print(classification_report(y_val, neural_net_predictions(shallow_nn, x_val), target_names=['Not Fraud', 'Fraud']))
```

```
713/713 [==============================] - 1s 1ms/step
              precision    recall  f1-score   support

   Not Fraud       1.00      1.00      1.00     22789
       Fraud       0.71      0.56      0.63        18

    accuracy                           1.00     22807
   macro avg       0.86      0.78      0.81     22807
weighted avg       1.00      1.00      1.00     22807
```

Similarly F1 score for ANN dropped due to overfitting caused by synthetic oversampling from SMOTE.

Conclusion

Gradient Boosting offers robust F1 score of 86% and precision of 76%, outperforming other Classifiers while taking less runtime than ANN and other deep learning architectures as proposed by Nguyen et al. [5]. In the study we compared effect of SMOTE over sampling on Models resulting in improvement in Random Forest classifier however more systematic sampling methods can resolve the overfitting issues caused by SMOTE on models which utilise gradient descent.

Scope for Future

More systematic algorithms such as SMOTE + ENN can produce more sharp clusters as compared to SMOTE however they are more computationally intensive to run. Such over sampling methods can be used in time series models such as bi-directional RNN's and LSTM's to better classify fraudulent transactions owing to the periodic nature of the transactions [8].

References

[1] Federal Trade Commission. (2024). *Consumer Sentinel Network Data Book 2023*. [online] Available at: https://www.ftc.gov/reports/consumer-sentinel-network-data-book-2023.

[2] S P Maniraj, Aditya Saini, Shadab Ahmed and Swarna Deep Sarkar (2019). Credit Card Fraud Detection using Machine Learning and Data Science. *International Journal of Engineering Research and*, 08(09). doi:https://doi.org/10.17577/ijertv8is090031.

[3] Thanh Thi Nguyen, Tahir, H., Abdelrazek, M. and Babar, A. (2020). Deep Learning Methods for Credit Card Fraud Detection. arXiv (Cornell University). doi:https://doi.org/10.48550/arxiv.2012.03754.

[4] Kaggle (2018). *Credit Card Fraud Detection*. [online] www.kaggle.com. Available at: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.

[5] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16(16), pp.321–357. doi:https://doi.org/10.1613/jair.953.

[6] Sarker, A., Must. Asma Yasmin, Md. Atikur Rahman, Harun, M. and Bristi Rani Roy (2024). Credit Card Fraud Detection Using Machine Learning Techniques. *Journal of computer and communications*, 12(06), pp.1–11. doi:https://doi.org/10.4236/jcc.2024.126001.

[7] Gupta, P., Varshney, A., Khan, M.R., Ahmed, R., Shuaib, M. and Alam, S. (2023). Unbalanced Credit Card Fraud Detection Data: A Machine Learning-Oriented Comparative Study of Balancing Techniques. Procedia Computer Science, 218, pp.2575–2584. doi:https://doi.org/10.1016/j.procs.2023.01.231.

[8] I. D. Mienye and Y. Sun, "A Deep Learning Ensemble With Data Resampling for Credit Card Fraud Detection," in IEEE Access, vol. 11, pp. 30628-30638, 2023, doi: 10.1109/ACCESS.2023.3262020. keywords: {Credit cards;Fraud;Random forests;Boosting;Stacking;Machine learning algorithms;Deep learning;Credit card;deep learning;ensemble learning;fraud detection;machine learning;neural network}.

[9] Noviandy, T. R., Idroes, G. M., Maulana, A., Hardi, I., Ringga, E. S. and Idroes, R. (2023) "Credit Card Fraud Detection for Contemporary Financial Management Using XGBoost-Driven Machine Learning and Data Augmentation Techniques", Indatu Journal of Management and Accounting, 1(1), pp. 29–35. doi: 10.60084/ijma.v1i1.78.