Write Up on Using Generative Adversarial Network to Oversample The Imbalanced Data and Applying XGBoost Classifier

## Introduction :

The rise of online payment systems has brought forth new avenues for fraudsters to exploit cardholders. These card frauds have posed significant challenges for both banks and individuals, leading to the development of various fraud detection techniques.

Initially, banks relied on card insurance and protection schemes to retain customers affected by card frauds. However, as fraudsters adapted to these measures, the need for more sophisticated solutions emerged.

Machine learning emerged as a powerful tool for combating card frauds. These systems, fueled by clean and comprehensive data, continuously adapt to evolving fraud patterns. The IEEE-CIS Fraud Detection competition, organized by IEEE-CIS and Vesta Corporation, sought innovative machine learning solutions utilizing Vesta's real-world e-commerce transaction data.

## Business Problem:

The traditional approaches for detecting transaction fraudulence are inefficient and expensive. They involve manual monitoring through humans, which is time-consuming and often results in legitimate transactions being flagged as fraudulent. To address this issue, we need an automated screening system that requires minimal human intervention to detect the legitimacy of transactions using machine learning. We will be building an ML model using a real-world e-commerce dataset to predict whether a transaction is fraudulent or not.

The prediction must be as accurate as possible since predicting a legitimate transaction as fraudulent will lead to bad customer experience and predicting a fraudulent transaction as legitimate will lead to huge financial losses. Additionally, the prediction should be instant so that customers can be alerted immediately if a fraudulent transaction occurs. Interpretability is also important, especially in cases where a transaction has been declared as fraud, since it is necessary to know why a transaction has been flagged as fraudulent.

Generative Adversarial Networks (GANs) can be used to generate synthetic data that can be used to train the machine learning model. GANs can generate realistic data that can be used to augment the existing dataset, which can improve the accuracy of the model.

## Machine Learning Problem:

The datasets provided by the Competition Host are as follows,

train\_transaction.csv : The transaction dataset comprising the transaction details to be used for training the model.

train\_identity.csv : The identity dataset comprising the additional details about the identity of the payer and the merchant between whom the transaction was performed and the details of transactions are present in the train\_transaction.csv.

test\_transaction.csv : The transaction dataset comprising the transaction information to test the performance of the trained model.

test\_identity.csv : The identity dataset comprising the additional identity information about the transactions present in the test\_transaction.

**Description of Transaction Dataset**

TransactionID — Id of the transaction and is the foreign key in the Identity Dataset.

isFraud — 0 or 1 signifying whether a transaction is fraudulent or not.

TransactionDT — timedelta from a given reference datetime (not an actual timestamp)

TransactionAMT — Transaction Payment Amount in USD.

ProductCD — Product Code.

card1 — card6 — Payment Card information, such as card type, card category, issue bank, country, etc.

addr — Address

dist — Distance

P\_emaildomain — Purchaser Email Domain.

R\_emaildomain — Receiver Email Domain.

C1-C14 — counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked.

D1-D15 — timedelta, such as days between previous transactions, etc.

M1-M9 — match, such as names on card and address, etc.

V1-V339— Vesta engineered rich features, including ranking, counting, and other entity relations.

Following Features are Categorical in the Transaction Dataset,

ProductCD

card1-card6

addr1, addr2

P\_emaildomain

R\_emaildomain

M1-M9

Description of the Identity Dataset

Following Features are present in the Identity Dataset,

TransactionID — Foreign key to the Transaction Dataset.

id\_01-id\_38 — Masked features corresponding to the identity of the card holders.

DeviceType — Type of Device used to make the Transaction.

DeviceInfo — Information regarding the characteristics of the Device.

Following Features are Categorical in the Identity Dataset,

DeviceType

DeviceInfo

id\_12 — id\_38

Variables in this table are identity information — network connection information (IP, ISP, Proxy, etc) and digital signature (UA/browser/os/version, etc) associated with transactions. They’re collected by Vesta’s fraud protection system and digital security partners

**Performance Metric:**

The problem of detecting transaction fraudulence is a binary classification problem where we need to classify a transaction as fraudulent or non-fraudulent. ROC-AUC will be our Key Performance Indicator (KPI). We will also use the Confusion Matrix to add more interpretability to the models.

ROC-AUC is preferred over Accuracy in tasks like Fraud Detection because it considers the trade-offs between precision and recall, whereas Accuracy only considers the number of correct predictions. Additionally, ROC-AUC is ideal for imbalanced datasets because it uses prediction probabilities. For example, in fraud detection, 90% of observations belong to non-fraudulent, and the remaining 10% to the fraudulent class as frauds are rare compared to non-fraudulent ones.

## E.D.A

**Some key findings**

It seems that the dataset is extremely imbalanced. Imbalanced datasets can lead to biased models and misleading accuracy scores. There are several techniques to handle imbalanced datasets, such as oversampling, undersampling, and synthetic data generation.

Regarding the missing values, it’s interesting to note that the V\_features have a nice pattern of missing values. This could be an indication that these features are not relevant to the problem at hand.

It’s also intriguing that the majority of transactions had a value of “W” corresponding to ProductCD. The second most found value was “C”, but the number of fraudulent transactions with value “W” and “C” were almost comparable. This helped in concluding that the transactions done for ProductCD “C” had the highest chance of being fraudulent as compared to other ProductCD categories.

It’s good to know that the addr2 feature corresponds to the Country Code and that most of the transactions belong to the same country. This could be useful information for further analysis.

Finally, it’s understandable that the majority of transactions did not have the R\_emaildomain value since not every transaction needed a transaction receipt and hence no information about the Receiver was present .

It’s interesting to note that the M1-M9 features were of the form True and False signifying whether a certain characteristic (like name on card match or not) was satisfied or not. This could be useful information for further analysis.

It’s great that the TransactionDT feature had the minimum value of 86400, which on further analysis helped in concluding that this feature is actually the number of seconds elapsed, since, 86400 = 246060. This feature was very helpful and some of the very important features were created using this feature which are discussed later on in the Feature Engineering Section.

It’s good to know that the Train and Test Split was done based on time and there was some time gap between the train and test set. This is a common practice in machine learning to avoid data leakage and ensure that the model is generalizable to new data.

Regarding the C\_features, it’s intriguing that they were highly intercorrelated. This could be an indication that these features are redundant and could be removed to reduce the dimensionality of the dataset.

Finally, it’s interesting to note that the amount spent was highest for ProductCD “R” and was lowest for ProductCD “S” and “C”. For ProductCD “W”, “H” and “R” if the TransactionAmt was high then that transaction was generally fraudulent. This could be useful information for further analysis.

## Data Cleaning

* Features having more than 90% missing values, a single value for the complete column, or more than 90% values the same were removed from the dataset.
* Collinearity was checked amongst features belonging to the same category like C\_features and D\_features, V\_features. Highly collinear features were removed if it did not lead to a reduction in Cross Validation AUC.
* To reduce V\_features, only that feature from each subgroup which had the most number of missing values was used, and the chosen feature acted as a representative of all the features in that group.
* Features which change over time were removed from the dataset.

## Feature Engineering

The dataset includes 312 features, of which 153 are newly added during the Feature Engineering Process. These new features are expected to enhance the predictive power of the model. For more details, please refer to the notebook. Also, Adversarial Validation for each of the features has been performed. By excluding features with AUC>0.7 from the final and training prediction, we can effectively prevent overfitting and ensure the model generalizes well with the unseen data. And the final number of features is 296 for the training dataset.

## Prediction:

Even though our XGBoost model is doing well overall with a high cross-validation score of 0.93, we're facing an issue. The precision of the model, which indicates how accurately it identifies true positives, is not meeting our expectations. It seems like the model is focusing more on catching all the actual positive cases but might be mistakenly flagging too many non-fraudulent transactions as fraudulent. To figure out why this is happening and make our model more balanced and accurate, we need to dig deeper into the data and the model's predictions.

## Generative Adversarial Network (GAN)

GANs can potentially be used to address the issue of low precision in the model. By generating synthetic minority class examples, GANs can help balance the dataset and reduce the bias towards the majority class. This can lead to a more accurate representation of the true class distribution and improve the model's ability to correctly identify minority class instances. Additionally, GANs can be used to generate high-quality examples that capture the underlying patterns and relationships in the data, which can further enhance the model's generalization performance.

### 8.1 cGAN Structure:

The code starts by defining the two main components of the cGAN: the generator and the discriminator. The generator is responsible for generating new data samples, while the discriminator is responsible for distinguishing between real and fake data samples.

The generator is a neural network that takes as input a noise vector and a label, and outputs the transaction data. The discriminator is also a neural network that takes as input the transaction data, and outputs a probability that the data is real.

**Structure of Generator**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense (Dense) (None, 512) 152064

leaky\_re\_lu (LeakyReLU) (None, 512) 0

dense\_1 (Dense) (None, 256) 131328

leaky\_re\_lu\_1 (LeakyReLU) (None, 256) 0

dropout (Dropout) (None, 256) 0

dense\_2 (Dense) (None, 128) 32896

leaky\_re\_lu\_2 (LeakyReLU) (None, 128) 0

dropout\_1 (Dropout) (None, 128) 0

dense\_3 (Dense) (None, 1) 129

=================================================================

Total params: 316417 (1.21 MB)

Trainable params: 316417 (1.21 MB)

**Structure of Discriminato**r

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_4 (Dense) (None, 128) 39040

leaky\_re\_lu\_3 (LeakyReLU) (None, 128) 0

batch\_normalization (Batch (None, 128) 512

Normalization)

dense\_5 (Dense) (None, 256) 33024

leaky\_re\_lu\_4 (LeakyReLU) (None, 256) 0

batch\_normalization\_1 (Bat (None, 256) 1024

chNormalization)

dense\_6 (Dense) (None, 512) 131584

leaky\_re\_lu\_5 (LeakyReLU) (None, 512) 0

batch\_normalization\_2 (Bat (None, 512) 2048

chNormalization)

dense\_7 (Dense) (None, 296) 151848

=================================================================

Total params: 359080 (1.37 MB)

Trainable params: 357288 (1.36 MB)

**Final Structure of cGAN**

Model: "model"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

=====================================================================

input\_5 (InputLayer) [(None, 304)] 0 []

input\_6 (InputLayer) [(None, 1)] 0 []

Generator (Functional) (None, 296) 359688 ['input\_5[0][0]', 'input\_6[0][0]']

Discriminator (Functional) (None, 1) 317009 ['Generator[0][0]', 'input\_6[0][0]']

=====================================================================

Total params: 676697 (2.58 MB)

Trainable params: 357896 (1.37 MB)

Non-trainable params: 318801 (1.22 MB)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

This trains a Conditional Generative Adversarial Network (CGAN) model using the X\_gans and y\_train\_gans variables as input data. The pos\_index and neg\_index variables are used to balance the number of positive and negative samples in the training data. The model is trained for 4000 epochs.

### 8.2 Applying cGAN to Oversample:

The application of GANs to oversample the imbalanced dataset for fraud detection yielded remarkable results. After applying on XGBoost with hyperparameter tuning on the generated balanced dataset the model achieved a perfect AUC score of 1.0 on the training data and a near-perfect score of 0.99 on cross-validation, indicating outstanding predictive performance. Moreover, the model demonstrated good precision and recall, suggesting that it can effectively identify both fraudulent and legitimate transactions.

### 8.3 Using K-Fold Cross Validation:

Implemented a cross-validated training loop using XGBoost's GroupKFold for the particular binary classification task. The dataset is split into six folds. This can help to prevent the model from Over-fitting and to evaluate the model's performance in a more realistic scenario, especially in time-series data where patterns can change over time.

### 8.4 Limitation of GANs for Oversampling:

* Training Instability: GANs are known to be difficult to train and can often suffer from instability issues, such as mode collapse. This can lead to the generation of low-quality and unrealistic synthetic data.
* Data Quality: The quality of the generated data can vary depending on the specific GAN structure and training parameters. In some cases, the generated data may not be representative of the real world fraudulent transactions which can negatively impact the performance of the fraudulent transactions, which can negatively impact the performance of the fraud detection model.
* Computational cost: Training GAN’s can be computationally expensive, especially for large datasets. This can make it impractical to use GAN’s for real time fraud detection applications.

## Conclusion:

Using GAN (Generative Adversarial Network) oversampling has significantly improved the model's ability to detect fraudulent transactions. The number of false positives (legitimate transactions incorrectly flagged as fraudulent) and false negatives (fraudulent transactions not detected) has been notably reduced. This means that the model is now more accurate in identifying both genuine and fraudulent transactions, making it a more reliable tool for fraud detection. The GAN oversampling technique has enhanced the model's performance by providing it with a more diverse and balanced dataset, ultimately leading to better outcomes in distinguishing between normal and fraudulent activities.

GAN’s offer a promising approach for oversampling transaction data for fraud detection models, but they also face certain limitations. Future improvements with above mentioned approaches may enhance further effectiveness of GAN’s fraud detection model.

### Future Opportunities:

GNN : Graph Neural Networks (GNNs) represent a promising avenue for future advancements in fraud detection. GNNs excel at analyzing complex relationships between entities and are well-suited for detecting fraudulent activity that involves coordinated networks of individuals or transactions.

Anomaly detection: This approach identifies data points that deviate significantly from normal patterns, which can be indicative of fraudulent activity. Techniques like one-class SVM, local outlier factor, and isolation forest can be used to detect anomalies without requiring labeled examples of fraud.

Semi-supervised learning: This approach can leverage both labeled and unlabeled data, which is particularly valuable when labeled fraudulent data is scarce. Techniques like self-training and co-training can be used to effectively utilize unlabeled data and improve model generalizability.

Reinforcement learning: This technique allows AI agents to learn by interacting with the environment and receiving rewards for desired outcomes. This can be applied to fraud detection by creating agents that learn to identify and respond to fraudulent activity in real-time.