IEEE-CIS Fraud Detection

Li, Zhengyuan (Donald)
Zhang, Wencheng (Bill)
Zhou, Weiyan (Vivian)

A Hybrid Approach

Mar 5, 2023

Project GitHub Repo: https://github.com/wenchengking/dataMining

Abstract

According Cybersource[1], fraud management is regarded as challenging by nine in ten merchants. Our goal is to develop a **hybrid** approach for fraud detection, especially for cards. Many of the studies in fraud detection in cards were conducted on modest-sized samples with a few familiar features and mainly focused on transaction-based approaches to detect fraud[2]. The dataset[3] We use comes from the well-received IEEE-CIS Fraud Detection competition on Kaggle. We used an aggregation approach, conducted dimensionality reduction and then tried several prediction algorithms.The complete procedure can be observed within the methodology framework (Figure 4). Our final best combination is autoencoders + neural network that achieved an AUC of 99.53% with the FNR (predicted not fraud but actually fraud) at 0.88%.

Paper Review and Model Approach

Per our experience, the fraud detection problem is famous for two reasons: data imbalance and the curse of dimensionality.

How data imbalance could affect model performance has been systematically studied[4] in the past. For the credit fraud detection problem, the data distribution is skewed towards non-fraudulent transactions, suggesting low false alarm

[1] Cybersouce - Global Fraud Report 22'
[2] Nghia Nguyen et. al (2022). A Proposed Model for Card Fraud Detection Based on CatBoost and Deep Neural Network
[3] Dataset
[4] Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study

rates might be an issue when being fed with large-scale data streams. In that scenario, Synthetic Minority Oversampling Technique (SMOTE)[5] is considered the more superior approach versus undersampling or ensemble methods[6]. While past literature has also suggested using frequent itemset mining (FIM)[7] as an alternative, we decided to proceed with the tried and true SMOTE, also for the fact that we have far more numerical columns than categorial.

Curse of dimensionality is another issue we want to fix due to the sheer volume of the data. Incentivised by the research on transaction aggregation[8], we decided to unmask the "user" to center transactions around. In other words, we attempt to classify an account as being in a fraudulent state, despite the possible presence of legitimate transactions. This allows us to fix some of the missing value problems that arise, and we fix the rest using k-nearest neighbors (KNN) imputer.

For dimension reduction, in addition to the vanilla linear principal component analysis (PCA), we used autoencoder as our nonlinear alternative, as studies has shown that the autoencoder performed either better or on the same level as kernel PCA[9] for real-world anomaly detection.

Our modeling approaches are a hybrid of both supervised and unsupervised learning algorithms. For supervised learning methods, there are two families: machine learning-based approach and deep learning-based approach. For machine learning-based approach, we tried random forest as it is came in just after a voting classifier on several comprehensive studies (an example[10]). We also saw example utilizing light gradient-boosting machine (lightGBM)[11] so we tried extreme gradient boosting (XGBoost) as an alternative. For deep-learning based approach, we tried the artificial neural network (ANN) approach since it's shown that the NN tends to check the pattern that has been used by a fraudster and compare it with the pattern of

[5] Chawla, N.V., et. al (2002). SMOTE: synthetic minority over-sampling technique
[6] Andrea Dal Pozzolo, et. al (2014). Learned lessons in credit card fraud detection from a practitioner perspective
[7] Masoumeh Zareapoor, Jie Yang (2018). A Novel Strategy for Mining Highly Imbalanced Data in Credit Card Transactions
[8] C. Whitrow et. al (2009). Transaction aggregation as a strategy for credit card fraud detection
[9] Mayu Sakurada & Takehisa Yairi (2014). Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction
[10] Hassan Najadat et. al (2020). Credit Card Fraud Detection Based on Machine and Deep Learning
[11] Altyeb A.T. & Sharaf J.M. (2020). An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine

the original cardholder to ensure if both patterns are a match or not[12]. We tried the multilayer perceptron (MLP) as a feedforward ANN.

We also tried unsupervised approaches. Starting with a vanilla k-means we learnt in-class, we stumbled upon Isolation Forest , Local Outlier Factor (LOF) as suggested by other researchers[13].

Data Analysis and Model Development

Data Analysis

The dataset used in our study is derived from real-world e-commerce transactions conducted by Vesta, a leading provider of secure payment solutions. The dataset contains a diverse set of features, ranging from device type to product attributes. Upon conducting an in-depth analysis of the data, we found that there are twelve different types of variables in the transaction table (table 1.1), with each type providing specific information about the transactions. For instance, the 'addr' variables offer details about the address, while the 'card1' through 'card6' variables contain information about payment cards, such as the card type and country. Additionally, we discovered four distinct types of variables in the identity table, each containing specific information about the identities associated with the transactions.

After a thorough examination of the datasets (Table 1.5 and Table 1.6), we have identified a major issue with a large number of missing values (NA) present in the data. Our plan is to investigate this matter further and develop a strategy to address it during the data engineering stage.To streamline our analysis, we merged the two data tables into a single, distinct dataset using a left join approach due to the presence of numerous transactions without a corresponding identity (table 1.7). "identity" and "transaction" are linked by a unique identifier called "TransactionID". The dataset consists of 435 columns and 590,505 rows, named df (table 1.8) . Such a large-scale dataset provides us with a wealth of information and allows us to explore various patterns and relationships between the different variables.

We can observe interesting patterns in the data. Mobile device users are more prone to fraud compared to desktop users, as shown

[12] R. Patidar, L. Sharma et al. (2011). Credit card fraud detection using neural network

[13] Narendra Zadafiya et. al (2022). Detecting Credit Card Frauds Using Isolation Forest And Local Outlier Factor - Analytical Insights

in Figure 2. Limited screen space and small keyboards, as well as the use of public Wi-Fi and less secure apps, may contribute to this trend. We also identified some correlations between variables under the same type, such as a positive linear relationship between D12 and D4, which is expected as they both describe time deltas. However, we need to address two major issues in the dataset.

Upon conducting a more in-depth analysis of the df table, we found that several features have an extremely high missing percentage, with some as high as 99%, and an average of around 195.623 missing values currently. These features do not carry much significance for our analysis, creating a problem that we need to address at a later stage.

After analyzing the target value, we have concluded that there is a substantial imbalance in the data, with a significantly higher number of non-fraudulent instances (569,877) than fraudulent instances (20,663), as shown in the plots. To address this issue, we may need to consider using techniques such as SMOTE to rebalance the data.

Data Engineering

Remedy to Curse of Dimensionality

Much of the focus of the data engineering stage has been put on missing value imputations. Since this is a past Kaggle competition, we had an abundance of experience to draw from. Most sources we saw chose to simply drop columns with a certain percentage (e.g. 80%) and then proceed with feature selection. Since we want to showcase what we have learnt in this course, we choose to perform a series of steps so that we can conduct dimension reduction

Transaction Aggregation

Inspired by the 1st place in the competition, we built upon their work to aggregate the transactions (details on how it works can be found here). In short, they realized that we are not predicting fraudulent transactions. Once a client (credit card) has fraud, their entire account is converted to isFraud=1. Therefore we are predicting fraudulent clients (credit cards). This motivated them to generate an algorithm to impute missing values for transactions by grouping them and assigning an UID. We adapted this method and found users with >= 2 transactions, filled the missing values with

mean or mode, for numerical and categorical features, respectively. Then we dropped transitions with no UIDs afterwards.

Imputation of Numerical Features

After the previous step, we still see 177 missing values per row. Therefore, we ran a KNN imputer.

Imputation of Categorical Features

For categorical features, we start by filling the blanks with 'unknown', because we believe that the fact that a field is missing carries some information as well. But there are still a handful of features with double-digit categorical values. Here, we borrowed the method by Leonardo Ferreira to group categories based on bigger families (e.g., "ie 11.0 for desktop" and "ie 11.0 for tablet" both belong to IE as we have a different column that represents device type).

We then applied one-hot encoding to the categorical features. We dropped the "DeviceInfo" column (since it is merely a contented string of the device-related column) and all so all the date-related columns (i.e., TransactionDT, D1-D15), since, as stated earlier, we are predicting fraud users not transactions.

Remedy to Class Imbalance

Now that we have dealt with the curse of dimensionality, we move onto class imbalance. The majority of observations in our dataset is non-fraudulent. To address this issue, we first used stratified sampling to split our dataset into training and testing sets, ensuring that the class proportions were maintained in both sets. Then, we employed the Synthetic Minority Over-sampling Technique (SMOTE) algorithm, which generates synthetic minority class samples by interpolating between existing minority class samples. This helped to balance the class distribution in the training set, enabling our machine learning models to better learn the patterns and features associated with the minority class. By applying these data preprocessing techniques, we were able to develop a more robust fraud detection model that could effectively identify both fraudulent and non-fraudulent transactions.

Dimension Reduction

Our methodology is tailored to overcome the challenges posed by high-dimensional datasets, which are computationally intensive and challenging to analyze. We address this by employing both linear and nonlinear techniques to reduce the

dimensionality of the dataset while retaining the most relevant features and minimizing the loss of information. To avoid leakage of information from the test set to the training sets, we apply dimension reduction techniques to the training set and subsequently to the test set.

Linear

For linear dimension reduction, we utilize PCA to project the data onto a lower-dimensional space while preserving as much variance as possible. Specifically, we reduce the 510 columns in the dataset to 80 principal components, which capture 85% of the information in the original dataset.

Non-linear

Nonlinear techniques such as autoencoders are used to preserve the complex structure and relationships between data points in the reduced dimensional space. This is achieved by training the neural network to learn a compressed representation of the data and reconstruct the input data from a lower-dimensional latent space. In our implementation, we use 10 bottlenecks in the autoencoder to capture as much information as possible, a choice made based on subjective judgment and empirical experience.

To evaluate the performance of PCA and autoencoders in modeling, we initially attempted to use PCA with XGBoost. However, we found that this approach was both computationally expensive and did not yield nearly the same level of predictive power as using autoencoders (the corresponding notebook can be found on Github). As a result, after trying a couple options for bottleneck, we decided to use the 10 bottleneck (nodes) generated by the autoencoder as our dimension-reduced features for modeling.

Hybrid ML Model Development

Our primary objective was to compare and contrast the effectiveness of supervised and unsupervised machine learning techniques in detecting fraudulent activities. By utilizing hybrid machine learning algorithms, we were able to examine the strengths and limitations of different approaches and provide valuable insights into the efficacy of each technique. This, in turn, can inform the development of more effective fraud detection systems.

Supervised Algorithms

For supervised algorithms, we tried a range of algorithms, including random forest, neural networks, and XGBoost, in the context of supervised machine learning. These algorithms were trained on labeled data to predict instances of fraud in new data.

- **Random Forest**

Random forest classifiers are a popular ensemble learning method for classification tasks, where multiple decision trees work together to make predictions. The algorithm constructs several decision trees using a random subset of features and data, and the final output is determined by the mode of predictions from all the trees.

Random forest classifiers can handle large datasets with high dimensionality, are robust against overfitting and can manage noisy and missing data. They provide a measure of feature importance, making them an excellent choice for feature selection. Additionally, they can handle both categorical and continuous data, making them suitable for large-scale applications.

- **Neural Network**

The neural network classifier is a machine learning technique that mimics the structure and function of the human brain. It comprises interconnected nodes that analyze and transmit data to forecast a class label for a given input. As it learns, the neural network modifies its weights and biases via backpropagation and gradient descent to reduce the difference between predicted and actual outputs.

Neural networks offer several benefits, such as the capacity to comprehend intricate, nonlinear associations between inputs and outputs, automatic feature extraction, and adaptability to diverse applications.

- **XGBoost**

XGBoost is a popular supervised machine learning algorithm that optimizes objective functions by sequentially adding decision trees to minimize the difference between predicted and actual values.The algorithm employs a gradient-based approach to improve the model's performance by addressing the residuals of the previous trees and regularization techniques such as L1 and L2 regularization to prevent overfitting and enhance generalization.

One of the main advantages of XGBoost is its capability of handling large datasets, as long as the data fits into memory. XGBoost's

robustness to noise enables it to perform well even when the data is noisy, and it can handle missing data, making it suitable for diverse datasets. Additionally, XGBoost is interpretable.

Non-supervised Algorithm

We also employed Isolation Forest, LOF, and K-means as our unsupervised machine learning techniques. These techniques were trained on unlabeled data to identify unusual patterns or anomalies in the data.

- **K-means**

In unsupervised machine learning, K-means is a widely used clustering algorithm that was covered in class. It groups a given dataset into a predefined number of k clusters by minimizing the sum of squared distances between each data point and its assigned cluster centroid.

This method is considered simple yet effective for clustering data points.

- **Isolation Forest**

Isolation Forest is an unsupervised anomaly detection algorithm that efficiently handles large datasets and detects both global and local anomalies. The algorithm creates a tree-like structure by randomly selecting features and split values until all instances are isolated, and the number of splits required to isolate an instance is used as an indicator of its anomaly score.

Isolation Forest is advantageous in handling large and high-dimensional datasets with many irrelevant attributes since it does not use distance or density measures to detect anomalies. This results in lower computational effort and memory usage.

- **LOF**

The Local Outlier Factor (LOF) is an unsupervised machine learning approach that identifies outliers within a dataset. It measures the local deviation of a data point concerning its neighboring points. By comparing the density of a data point to its k nearest neighbors, the LOF algorithm identifies any data point whose density is much lower than its neighbors as an outlier.

LOF is particularly useful for datasets with a high-dimensional feature space or when traditional distance-based methods like clustering and nearest-neighbor methods fail to detect outliers.

Findings and Conclusion

Model Evaluation

For model evaluation in supervised machine learning and unsupervised machine learning model, we choose AUC as our decision standard because AUC is a robust metric that takes into account both the true positive rate (TPR) and false positive rate (FPR). AUC provides a single value that summarizes the overall performance of the model across different threshold values, making it easy to compare different models and select the best one for the specific fraud detection problem. Additionally, AUC is less sensitive to class imbalance than other metrics such as accuracy or F1 score, which can be important in fraud detection where the number of fraudulent transactions is often much smaller than the number of non-fraudulent transactions.

Besides, we also consider FPR and false negative rate (FNR) individually for the robustness of model evaluation. FNR is the rate at which the model fails to detect actual fraud, or in other words, it incorrectly classifies fraudulent transactions as non-fraudulent. **A high FNR** means that the fraud detection system is missing a significant number of fraudulent transactions, which **could result in financial losses for the business**. On the other hand, FPR is the rate at which the model incorrectly classifies non-fraudulent transactions as fraudulent. **A high FPR** means that the fraud detection system is generating a large number of false alarms, which **could lead to customer dissatisfaction and increased operational costs for the business**.

Through fitting both supervised and unsupervised machine learning models, we determined that supervised machine learning achieved superior performance in predicting fraudulent cases. After conducting cross-validation and grid search on the training set, we selected the best hyperparameters and in turn, the top models - Random Forest, XGBoost, and neural networks - based on their cross-validation AUC scores. We then proceeded to visualize their performance using table 1.9. Among them, the neural network model with 100 layers achieved the highest cross-validation AUC score of approximately 0.99998, followed by the XGBoost classifier with a score of 0.99815, and the Random Forest classifier with a score of 0.95881. To evaluate the robustness of these models on new observations, we tested them on a separate 20% of the dataset that was not used in training. The results indicated that the neural network model had the highest AUC score of approximately 0.99530, followed by the XGBoost classifier with a score of 0.98350, and the Random Forest

classifier with a score of 0.87550. Therefore, the neural network model was found to be the best-performing model for fraud detection in this study.

Upon a detailed examination of the model performance, it was found that the neural network model had the lowest FNR and FPR. Specifically, the FNR for the neural network model was approximately 0.00876, and the FPR was about 0.00070. For the XGBoost classifier, the FNR was approximately 0.03220, and the FPR was about 0.00077. The Random Forest classifier exhibited the highest FNR and FPR, which were 0.20608 and 0.04293, respectively. Moreover, in the test dataset, the neural network model was still observed to be the most effective in detecting fraudulent transactions in terms of false positive rate and false negative rate.

Based on our research, we have discovered that when it comes to unsupervised learning, the k-means clustering method has an AUC of 0.65455 during cross-validation and an AUC of 0.34746 during testing. Additionally, k-means clustering displays a higher incidence of false negatives and false positives compared to supervised machine learning models. In our investigation, we also found that the local outlier factor outperforms k-means clustering in terms of test AUC and FPR.

Although Isolation Forest and LOF produce similar outcomes, LOF has a slightly better test AUC and FNR. By examining all unsupervised machine learning models, we conclude that LOF is the optimal model in terms of AUC for an untrained test dataset.

In general, supervised machine learning models exhibit superior performance compared to unsupervised machine learning models. In this study, it has been found that neural networks are the most effective model for detecting fraud.

Feature Importance

As suggested during the presentation, we conducted a feature importance analysis to see if we could extract some useful information. Since our best model is one neural network loaded on another, we needed to do this step-by-step.

We started off with the final result we got from our MLP and worked backwards. The MLPClassifier in scikit-learn does not have an inherent feature importance attribute, but we were able to use permutation importance as an alternative. It works by shuffling each feature independently and measuring the impact on the model's performance. The final result is an array of

mean importance for each feature (Table 1.10).

In the table, the higher the mean importance, the more important the feature. We can see that the fourth feature has the highest permutation importance - it's about 50% more than the second most important one. The partial dependence plot of that feature shows the same thing (Table 1.11, for comparison, other plots can be found in the notebook as well). The 2nd, 8th and 9th are also features of greater importance.

Once we had the features of interest, we went back to the fitted autoencoder model to obtain the original feature representation, which would allow us to find out which original feature contributed most. One such method is to analyze the weights of the encoder's layers. Specifically, we could look at the weights of the first layer of the encoder, which corresponds to the linear transformation from the original feature space to the reduced feature space.

The final result incorporated ten most important original features for the ten features we got from our autoencoder. We now have a list of 100 features. For the ones we picked (2nd, 4th, 8th, 9th), we got 40 original features that we could attempt a deeper dive and extract meanings.

The very first thing we noticed is that almost 60% of these important features are 'Vxxx' (with V256 repeating twice). These are Vesta engineered rich features, including ranking, counting and other entity relations. We couldn't really tell what they mean but I believe these could be very helpful for Vesta. Out of the remaining features (Table 1.12), 'id_30' and 'id_31' took our notice immediately. They represent device type and browser type. We thought it's pretty interesting because if you are a user of any kinds of mobile cloud service, you are no strangers to dual authentication and biometric password (e.g., fingerprint or face lock). We know some credit card companies would send you a text when they suspect a fraudulent transaction. But maybe biometric password is one of the features credit card companies can push for in the future as well. We also noticed there is a group of 'Mx', which were used to show if name/address/other information matches or not. This is not counter-intuitive, as shoppers are usually required to show an ID when shopping with credit cards. What's also interesting is the 'card4' columns. There are five types of credit card issuers in this dataset - AMEX, Discover, Mastercard, Visa, and unknown (only 603 rows). According to the feature importance analysis, transactions using cards with

AMEX and Visa carried the most predictive power. It's not so surprising to say the least since AMEX is generally regarded as having a strong reputation for fraud detection and prevention. We believe that could translate to the scrutiny of how these companies look at fraud in general.

Recommendation and Suggestions

We employed two dimension reduction techniques to address the issue of high dimensionality, and implemented three supervised and three unsupervised machine learning algorithms to develop a fraud detection system. Although our approach has shown promising results, there is still room for improvement.

In terms of modeling, we suggest exploring the use of graph-based methods for detecting fraud patterns. Graph-based methods have been shown to be effective in detecting fraud in complex systems such as social networks, where fraudsters often operate by creating multiple fake accounts and establishing connections between them. By modeling the connections between entities in our dataset as a graph, we may be able to detect fraudulent patterns that are not visible in the original data.

Additionally, hybrid supervised machine learning may also prove to be effective in detecting fraud patterns.Hybrid-supervised learning combines the advantages of supervised and unsupervised learning by using a small amount of labeled data and a large amount of unlabeled data. By leveraging both labeled and unlabeled data, hybridi-supervised learning may be able to identify subtle patterns that are difficult to detect using supervised or unsupervised learning alone.

Table 1.1 : Information of two dataset

Dataset	Variable Names	Variable Description
Transaction Table	TransactionDT	timedelta from a given reference datetime (not an actual timestamp)
	TransactionAMT	transaction payment amount in USD
	ProductCD	product code, the product for each transaction
	card1 - card6	payment card information, such as card type, card category, issue bank, country, etc.
	addr	address
	dist	distance
	P_ and (R__) emaildomain	purchaser and recipient 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 transaction, etc.
	M1-M9	match, such as names on card and address, etc
	Vxxx	Vesta engineered rich features, including ranking, counting, and other entity relations
	isFradu	Whether this transaction is fraud or not
Identity Table	DeviceType	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. The field names are masked and pairwise dictionary will not be provided for privacy protection and contract agreement
	DeviceInfo	
	id_xx	
	TransactionID	

Table 1.2 : Total number of fraud vs not fraud in detection

Fraud	Not fraud
569877	20663

Table 1.3 : The number of missing value per row

The average number of missing value per row
195.62277407118907

Table 1.4 : Total percentage of missing value of each feature in df table

Column	% of missing value
id_24	0.991962
id_25	0.991310
id_07	0.991271
...	...
C1	0.000000
isFraud	0.000000
TransactionID	0.000000

Table 1.5 : First five rows of transaction table

	TransactionID	isFraud	TransactionDT	TransactionAmt	ProductCD	card1	card2	card3	card4	card5	...	V330	V331	V332	V333	V334	V335	V336	V337	V338	V339
0	2987000	0	86400	68.5	W	13926	NaN	150	discover	142	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2987001	0	86401	29	W	2755	404	150	mastercard	102	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2987002	0	86469	59	W	4663	490	150	visa	166	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2987003	0	86499	50	W	18132	567	150	mastercard	117	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2987004	0	86506	50	H	4497	514	150	mastercard	102	...	0	0	0	0	0	0	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Table 1.6: First five rows of identity table

	TransactionID	id_01	id_02	id_03	id_04	id_05	id_06	id_07	id_08	id_09	...	id_31	id_32	id_33	id_34	id_35	id_36	id_37	id_38	DeviceType	DeviceInfo
0	2987004	0	70787	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	samsung b	32	2220x1080	match_stat	T	F	T	T	mobile	SAMSUNG SM-G892A Build/NRD90M
1	2987008	-5	98945	NaN	NaN	0	-5	NaN	NaN	NaN	...	mobile safa	32	1334x750	match_stat	T	F	F	T	mobile	iOS Device
2	2987010	-5	191631	0	0	0	0	NaN	NaN	0	...	chrome 62.	NaN	NaN	NaN	F	F	T	T	desktop	Windows
3	2987011	-5	221832	NaN	NaN	0	-6	NaN	NaN	NaN	...	chrome 62.	NaN	NaN	NaN	F	F	T	T	desktop	NaN
4	2987016	0	7460	0	0	1	0	NaN	NaN	0	...	chrome 62.	24	1280x800	match_stat	T	F	T	T	desktop	MacOS
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Table 1.7: Unique transaction ID in each table

Unique Transaction ID number in identity table	Unique Transaction ID number in transaction table
144233	590540

Table 1.8: df : Combined dataset

	Transactio	isFraud	Transactio	Transactio	ProductCD	card1	card2	card3	card4	card5	...	id_31	id_32	id_33	id_34	id_35	id_36	id_37	id_38	De
0	2987000	0	86400	68.5	W	13926	NaN	150	discover	142	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1	2987001	0	86401	29	W	2755	404	150	mastercard	102	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	2987002	0	86469	59	W	4663	490	150	visa	166	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	2987003	0	86499	50	W	18132	567	150	mastercard	117	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	2987004	0	86506	50	H	4497	514	150	mastercard	102	...	samsung b	32	2220x1080	match_stat	T	F	T	T	mo
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	

Table 1.9: Result

	CV AUC	Test AUC	Test FNR	Test FPR
Neural Networks Classifier	0.99998	0.99530	0.00876	0.00070
XGBoost Classifier	0.99815	0.98350	0.03220	0.00077
Random Forest Classifier	0.95881	0.87550	0.20608	0.04293
Isolation Forest	0.49300	0.50828	0.96574	0.01771
Local Outlier Factor	0.48777	0.56581	0.77331	0.09507
K Means Clustering	0.65455	0.34746	0.56826	0.73682

Table 1.10: Mean permutation importance of each feature generated by autoencoder

```
1st feature: 0.077
2nd feature: 0.268
3rd feature: 0.103
4th feature: 0.405
5th feature: 0.119
6th feature: 0.127
7th feature: 0.084
8th feature: 0.217
9th feature: 0.243
10th feature: 0.068
```

Table 1.11: Partial dependence plot for the 4th feature (starting_index=0)

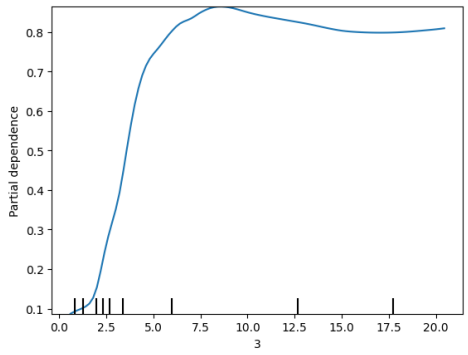

Table 1.12: Important original features other than 'Vxxx'

```
['D9',
 'M5_F',
 'M5_T',
 'M6_F',
 'M9_T',
 'TransactionAmt',
 'card4_american express',
 'card4_visa',
 'id_12_NotFound',
 'id_14',
 'id_30_Linux',
 'id_30_Mac',
 'id_31_Edge',
 'id_31_Firefox',
 'id_31_other',
 'id_37_unknown']
```

Figure 1: Fraudulent data vs Non-Fraudulent data

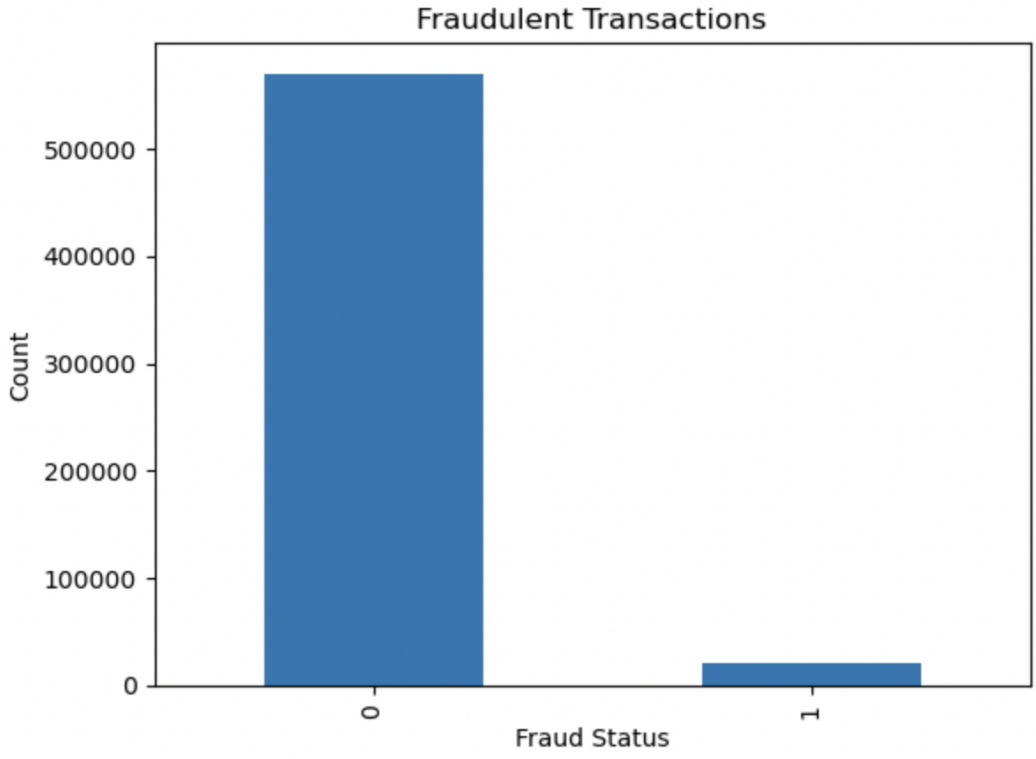

Figure 2: Comparison of Fraudulent Transactions between Desktop and Mobile Users

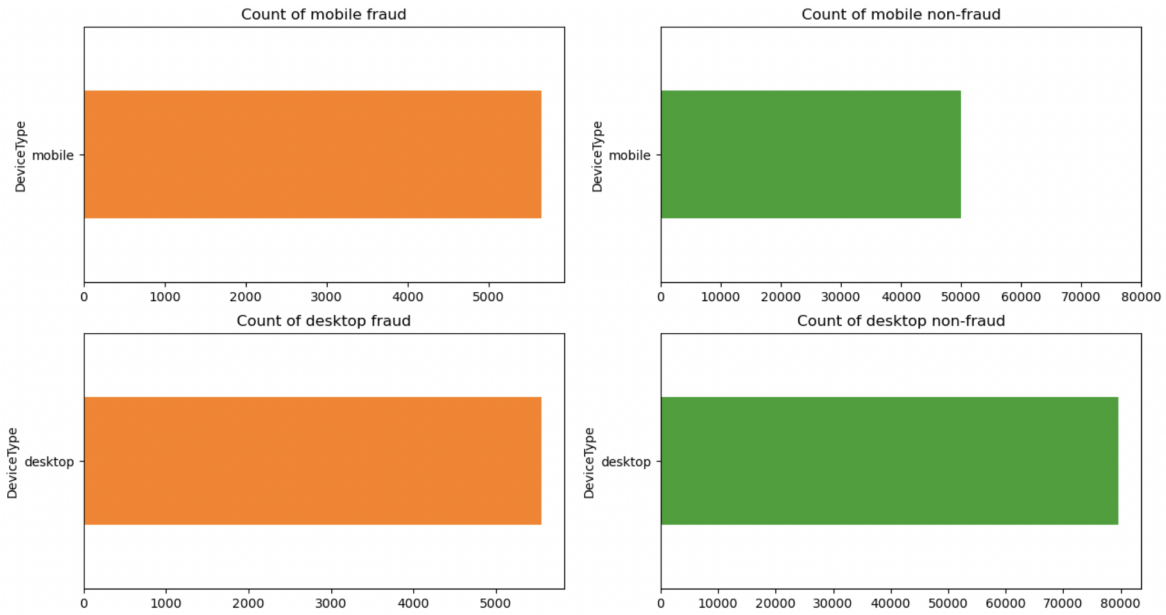

Figure 3: Relationship between D's type

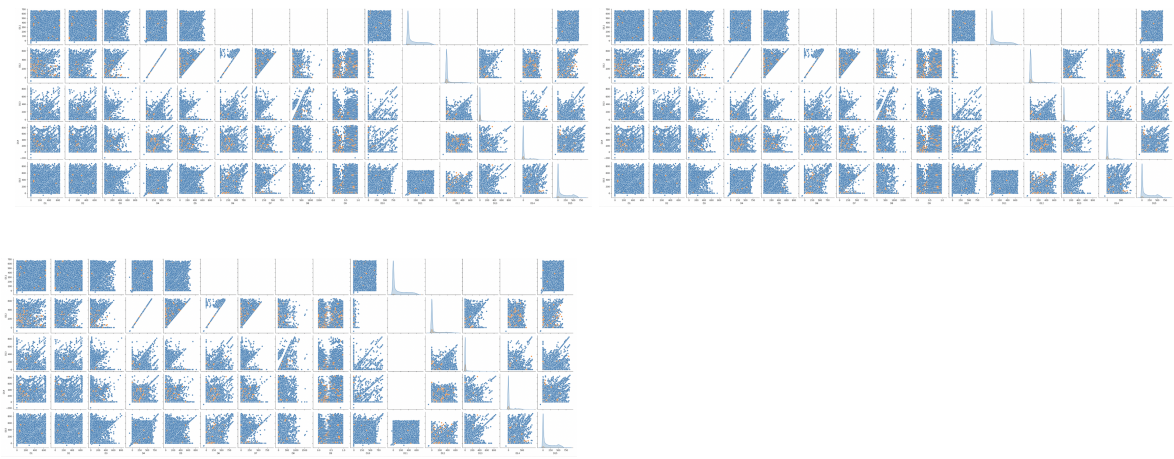

Figure 4: Methodology framework

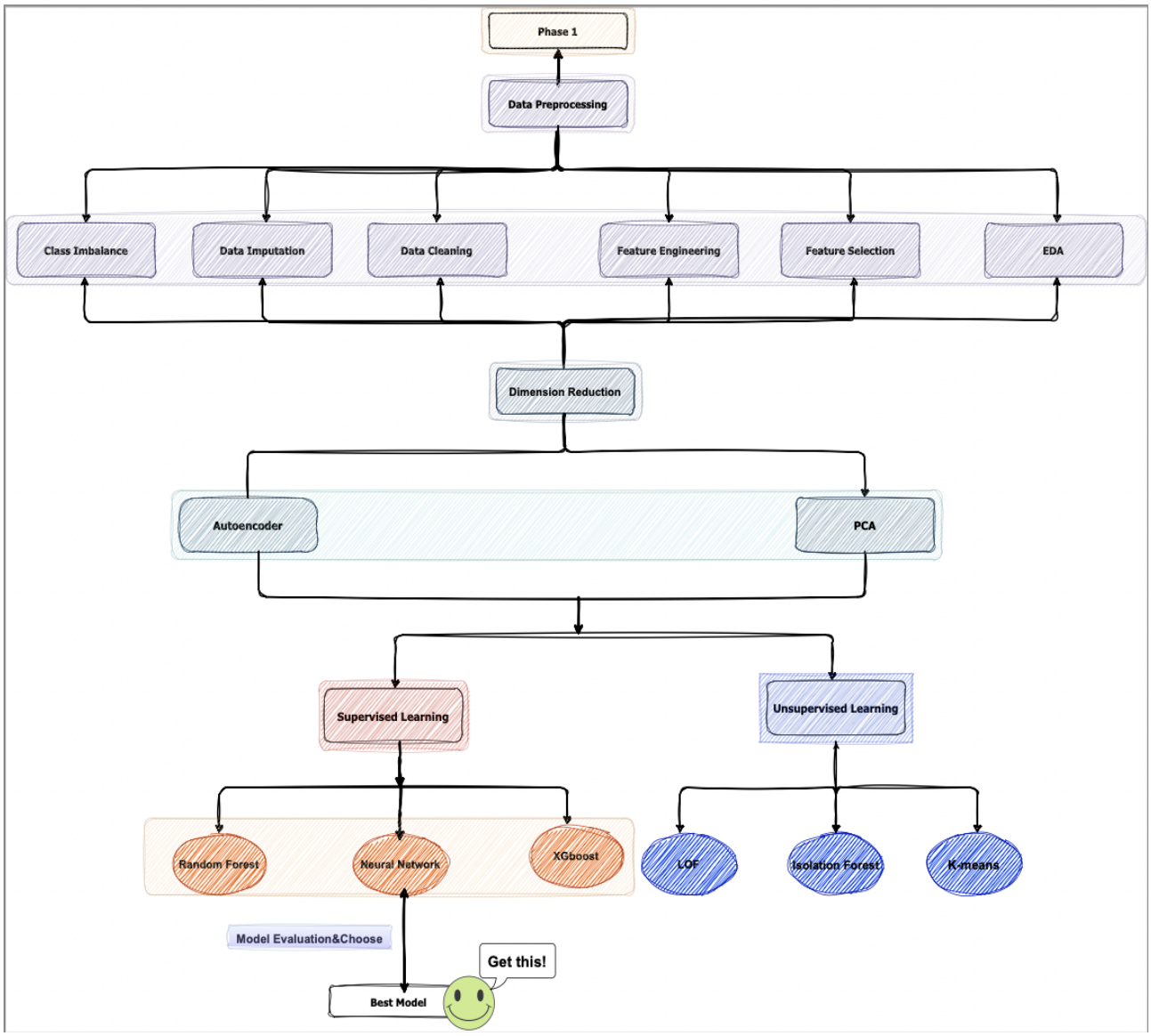

Figure 5: Correlation Heatmap