Fraud Detection - Credit Card Transactions

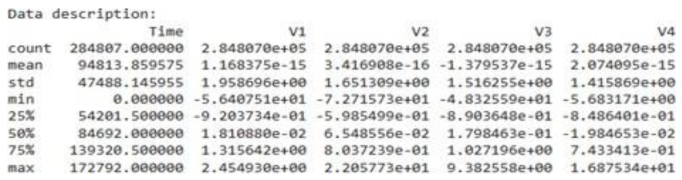
Rithvij Pasupuleti
KU Id: 3143730
Department of EECS University of Kansas
rithvij.pasupuleti@ku.edu

I. Introduction

In today's digital era, the ease of making purchases through online platforms and mobile apps has revolutionized the way we shop. Credit cards have become indispensable tools in facilitating these transactions, sparing us the hassle of carrying cash and offering the flexibility of deferred payments. However the increase in credit card transactions coupled with huge surge in digital payments has been leading to lot of malicious and fraudulent transactions. To mitigate these risks, credit card companies must possess robust and efficient mechanism to detect the fraudulent activities comprehensively. Machine learning plays a vital role in the fraud detection due to its ability to analyze large volumes of transaction data and identify complex patterns to make predictions. Hence supervised and unsupervised machine learning models have been explored to address the significant issue of false negatives in credit card transaction dataset.

II. Objective

Minimizing false negatives in fraud detection is crucial because failing to identify fraudulent transactions can have severe consequences. False negatives occur when a fraudulent transaction is incorrectly classified as non-fraudulent, allowing suspicious activity to go undetected. This can result in financial losses for both the cardholder and the financial institution, as well as an increased risk of further fraudulent activity. Therefore, mitigating false negatives is essential to protect financial assets and maintain trust in the system. To tackle this challenge, supervised learning ensemble models like Balanced Random Forest Classifier and XGBoost, along with Artificial Neural Networks such as Multi-layer Perceptron, are utilized for fraud detection tasks to effectively minimize false negatives. Additionally, unsupervised learning models like Principal Component Analysis and autoencoders are explored, especially in cases where credit card transactions are unlabeled and fraudulent transactions are rare compared to non-fraudulent cases. PCA and Autoencoder help analyze and mitigate false negatives by identifying patterns in the data. These models are then compared to determine which one is most effective at minimizing false negatives.



```
Data description:
                Time            V1            V2            V3            V4
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  1.168375e-15  3.416908e-16 -1.379537e-15  2.074095e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01
```

Fig. 1. Snippet of summary statistics of some features - The dataset does have any missing values

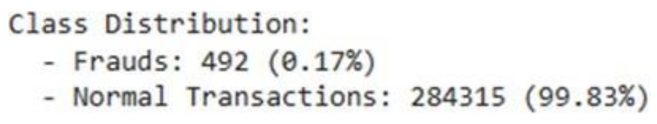
III. Dataset description

The dataset comprises credit card transactions conducted by European cardholders in September 2013. Over a span of two days, there were 492 instances of fraud out of a total of 284,807 transactions, resulting in a highly imbalanced dataset where fraudulent transactions constitute only 0.172% of all transactions.The dataset features include V1 through V28, representing anonymized features, while the "Time" feature indicates the seconds elapsed since the first transaction. The "Amount" feature denotes the transaction amount, which can be utilized for cost-sensitive learning based on specific examples. Finally, the "Class" feature serves as the response variable, with a value of 1 denoting fraud and 0 indicating legitimate transactions.

Dataset link: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

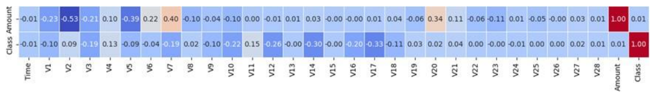
IV. Data analysis

- To gain insights into the structure and characteristics of the dataset, We utilized the df.describe() function to compute **"summary statistics"** for each numerical feature in the dataset. A snippet of the summary of statistics can be observed from Fig - 1
- Next class distribution is analyzed and significant class imbalance has been observed between fraudulent and non-fraudulent transactions which can be observed from Fig- 2



```
Class Distribution:
  - Frauds: 492 (0.17%)
  - Normal Transactions: 284315 (99.83%)
```

Fig. 2. Out of 2,84,807 transactions



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
Amount	-0.01	-0.23	-0.53	-0.21	0.10	-0.39	0.22	0.40	-0.10	-0.04	-0.10	0.00	-0.01	0.01	0.03	-0.00	-0.00	0.01	0.04	-0.06	0.34	0.11	-0.06	-0.11	0.01	-0.05	-0.00	0.03	0.01	1.00	0.01
Class	-0.01	-0.10	0.09	-0.19	0.13	-0.09	-0.04	-0.19	0.02	-0.10	-0.22	0.15	-0.26	-0.00	-0.30	-0.00	-0.20	-0.33	-0.11	0.03	0.02	0.04	0.00	-0.00	-0.01	0.00	0.00	0.02	0.01	0.01	1.00

Fig. 3. snippet of correlation matrix obtained for our dataset

• The **"correlation matrix"** serves as a valuable tool for understanding the relationships between features in the dataset. Features that are highly correlated may have a stronger influence on the model's predictions. By focusing on these influential features, we can gain insights into the factors driving fraudulent transactions.A snippet of correlation matrix can be observed from Fig- 3

For instance, features 'V2' and 'Amount' show strong negative correlation Features 'V7' and 'Amount' show strong positive correlation.Most of the features have weak negative correlation with each other.

• **"Box plots"** are used to analyze the distribution of features in the dataset and to observe potential variability in the data among the given classes. By observing inter-quartile ranges and spread of data for both the classes for each of the features, we can compare the distribution of each feature with respect to classes in our dataset. The medians of the boxes for each feature for each of the classes can be observed to identify whether the distributions are similar and potential outliers can also be observed. A sample boxplot can be observed from Fig - 4

• It is crucial to conduct **"normality test"** to understand if the features of our dataset follow normal distribution or not. This is because some of machine learning classifers which are parametric models like Linear Discriminant Analysis, Gaussian na¨ıve Bayes etc assume that the features are normally distributed perform better if the features follow specific distribution. Whereas non-parametric models such as decision trees and random forest models are less sensitive to distribution assumptions.

Hence normality test has been done and the obtained p-values are all close to "0" indicating that all the features are not normally distributed.A snippet of p-values obtained from normality test can be seen from Fig- 5

• Since we observed that the data does not follow normal distribution, we can perform a non-parametric test- **"Mann-Whitney U test"** to observe the **statistical difference between distribution of classes**. A low p-value here suggests that there is significant difference for each feature of the dataset among different classes and high p-value(typically greater than 0.05) would suggest there is not much statistical difference between features of the dataset among classes.

Features 'V13', 'V15' and 'V22' are features which can be seen are not statistically different among the 2 classes and this information can be useful for feature selection.

• To observe which feature contributes the most to the prediction of a model, we can use tree-based models like Random forest and gradient boosting classifier. Using feature importance scores from the training model object, we can observe the impact of each feature on the model prediction Using Random Forest and gradient Boosting Classifiers to get feature importance and getting the average of feature importance scores. Plot of average feature importance can be observed from Fig - 6

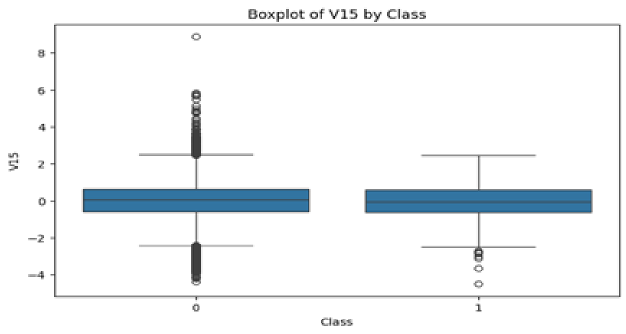
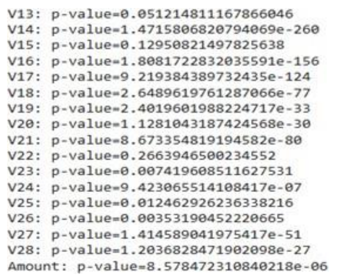


Fig. 4. box plot for feature 'V15'



```
V13: p-value=0.05121481116786046
V14: p-value=1.4715806820794069e-260
V15: p-value=0.12950821497825638
V16: p-value=1.8081722832035591e-156
V17: p-value=9.219384389732435e-124
V18: p-value=2.6489619761287066e-77
V19: p-value=2.4019601988224717e-33
V20: p-value=1.1281043187424568e-30
V21: p-value=8.673354819194582e-80
V22: p-value=0.2663946500234552
V23: p-value=0.007419608511627531
V24: p-value=9.423065514108417e-07
V25: p-value=0.012462926236338216
V26: p-value=0.00353190452220665
V27: p-value=1.414589041975417e-51
V28: p-value=1.2036828471902098e-27
Amount: p-value=8.578472310840218e-06
```

Fig. 5. Normality test- snippet of class p-values obtained

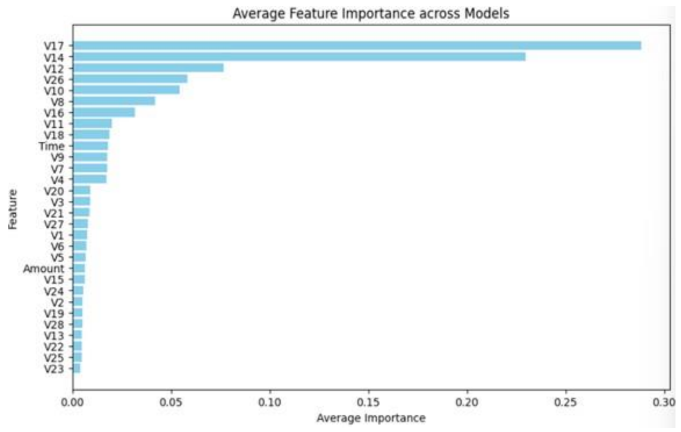


Fig. 6. Average of feature importance scores for Random Forest and gradient Boosting Classifiers

V. Models Implemented

Supervised Learning Models:

A. *Balanced Random Forest Classifier*

Balanced Random Forest Classifier is a machine learning algorithm used for classification tasks with imbalanced datasets. It addresses a common problem where a dataset has a significant skew in the number of data points belonging to different classes.

Regular Random Forest vs Balanced Random Forest: A regular random forest algorithm trains multiple decision trees on random subsets of data (with replacement) called bootstrap samples. This process is known as bagging. However, in an imbalanced dataset, the majority class ends up dominating the bootstrap samples, leading the model to prioritize it during training. Balanced Random Forest Classifier specifically tackles this issue.

Why Balanced Random Forest Classifier is a Preferred Choice for Credit Card Fraud Detection:

• Imbalanced Data: Fraudulent transactions are thankfully rare compared to legitimate purchases. This creates a classic imbalanced dataset where the model might prioritize classifying transactions as legitimate (the majority class) and miss fraudulent ones (the minority class). Balanced Random Forest Classifier addresses this by ensuring both classes are well-represented during training, making the model more sensitive to fraud.

• Handles Complex Patterns: Credit card fraud often involves complex patterns across various factors like location, purchase amount, time of day, and past spending habits. Random forests are known for their ability to capture these non-linear relationships between features, making them suitable for fraud detection.

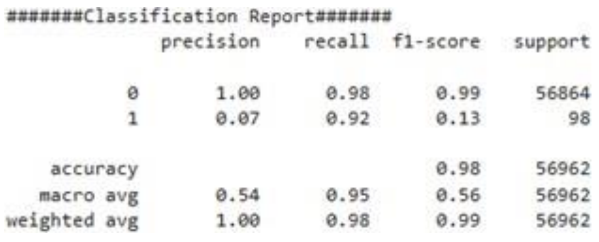
• Robust to Outliers: Fraudulent transactions can be outliers in the data compared to normal spending patterns. Balanced Random Forest Classifier's ensemble of decision trees helps reduce the impact of any single outlier on the overall model's performance.

Implementation and results:

Imbalanced Dataset Handling: The model demonstrates a high ROC AUC score on the test set (0.978), suggesting strong performance in distinguishing between positive and negative classes. Additionally, the low Brier score (0.0308) indicates good calibration of the model's predicted probabilities. These results showcase the model's effectiveness in handling imbalanced datasets, where fraudulent transactions are relatively rare compared to legitimate ones.

Best Parameters: The grid search identifies optimal parameters for the model, favoring deeper trees ('max_depth' set to None), relatively low 'min_samples_split', and 'n_estimators' of 100. These parameters are likely well-suited for capturing complex patterns in the data while mitigating the risk of overfitting.

Performance Metrics: The classification report for the model can be seen from Fig: 7. While the model exhibits high recall (0.92) for the minority class (fraudulent transactions),



```
#######Classification Report#######
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     56864
           1       0.07      0.92      0.13        98

    accuracy                           0.98     56962
   macro avg       0.54      0.95      0.56     56962
weighted avg       1.00      0.98      0.99     56962
```

Fig. 7. Classification report for Balanced Random Forest using best parameters to minimize overall false negatives

precision is relatively low (0.07). This discrepancy indicates a high number of false positives, where legitimate transactions are incorrectly classified as fraudulent. Despite the high recall, the low precision highlights the need for improvement in accurately identifying fraudulent transactions. The F1-score, which balances precision and recall, is also low (0.13) for the minority class. This suggests a trade-off between precision and recall, further emphasizing the need for optimizing the model's performance metrics.

Since the primary objective here would be to minimize the False negatives, we still reduced our overall **false negatives** to very small number - **8** out of **56,962 testing instances** using best parameters for Balanced Random Forest Classifier

The false positives and false negatives obtained from confusion matrix is as follows-

Balanced Random Forest	**False Negatives**	**False Positives**
Using Best Parameters(Minimal False Negatives)	8	1386

TABLE I
VALUES OBTAINED FROM CONFUSION MATRIX FOR BALANCED RANDOM FOREST CLASSIFIER

The confusion matrix reveals a notable number of false positives (1386), representing legitimate transactions predicted as fraudulent. Conversely, false negatives (fraudulent transactions incorrectly classified as legitimate) are relatively low, with only 8 instances which is still pretty impressive given our main objective.

B. *Xgboost*

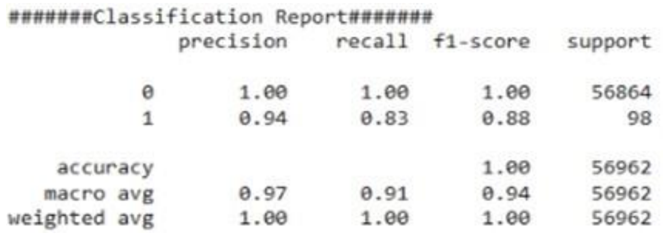
XGBoost, short for Extreme Gradient Boosting, stands as a prominent algorithm in the realm of machine learning, revered for its efficacy and versatility.

Fundamental principles:

• XGBoost belongs to the ensemble learning category, specifically the gradient boosting framework. It operates by sequentially combining the predictions of multiple individual models, typically decision trees, to form a robust ensemble model

• The algorithm iteratively adds weak learners to the ensemble, with each new learner focusing on correcting the errors made by its predecessors.

• XGBoost employs a gradient descent optimization technique to minimize a predefined loss function during



```
#######Classification Report#######
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.94      0.83      0.88        98

    accuracy                           1.00     56962
   macro avg       0.97      0.91      0.94     56962
weighted avg       1.00      1.00      1.00     56962
```

Fig. 8. classification report for XGBoost before applying best threshold

model training. By iteratively adjusting the model parameters based on the gradient of the loss function, XGBoost enhances its predictive accuracy.

Unique features:

• Its ensemble learning framework, coupled with gradient boosting, allows it to capture complex relationships in data and achieve high accuracy in predictions

• Its optimization techniques, such as cache awareness and out-of-core computing, maximize hardware usage and computational resources when compared to balanced random forest models

• XGBoost is highly scalable and capable of handling large datasets.

• Various hyper parameters such as "early-stopping", "scale-pos-weight" can be incorporated to **address potential over-fitting** and **class imbalance**. By taking these features into context, Xgboost would be very much suitable and efficient algorithm that can be used on credit card transaction dataset because of its ability to handle imbalanced data and prevent overfitting ensuring that fraudulent transactions are accurately identified and mitigating overall false negatives.

Implementation and results

The XGBoost classifier was trained with adjusted class weights to counter class imbalance, using the hyperparameter "scale_pos_weight" while "early stopping rounds" were employed to mitigate overfitting.

Brier score and ROC AUC score are commonly used metrics for evaluating the performance of classification models. Brier score measures the mean squared distances between predicted probabilities and actual values. ROC AUC score indicates how better a model can effectively distinguish between fraudulent and non-fraudulent transactions.

The obtained brier score was 0.00034 indicating model's predictions are closer to actual outcomes and obtained ROC AUC score was 0.9764 indicating classifier's excellent ability to distinguish between positive and negative classes.

The Classification report for Xgboost model can be seen at Fig: 8. We have just **17 false negatives** and only **5 false positives** which is impressive and one of the best possible results that can be obtained but still our main objective here is to minimize false negatives since false negatives are crucial in fraud detection.

For this, the approach used is that to minimize false negatives while ensuring that the false positive rate does not exceed 10%.By using this approach, for different thresholds, we can compute predictions and confusion matrix for each of them to check our condition of within 10% false positive rate and find which threshold suites best that gives us minimal false negatives.

By using this approach the best threshold obtained is 0.001, we can observe the resulting false positives and false negatives from Table : II

XgBoost	False Negatives	False Positives
Without applying Best threshold (after addressing class imbalance)	17	5
After applying Best threshold (Minimal False Negatives)	7	1395

TABLE II
VALUES OBTAINED FROM CONFUSION MATRIX FOR XGBOOST MODEL

The false negatives now are just 7 whereas false positives are 1395 among 56,962 testing instances. It's **a trade-off between false negatives and false positives in classification tasks and since we are focusing mainly on minimizing false negatives** this outcome is still impressive.

C. *Multi-layer Perceptron*

It is a type of deep neural network that is commonly used in supervised learning tasks where relationships in the data are learned using layers of neurons. The MLP model consists of an input layer, multiple hidden layers of neurons, and an output layer. Each neuron in a layer is connected to all neurons in the previous layer and these connections have weights that are adjusted during training. MLP uses backpropagation for training which is a method to adjust the weights of the neural network in terms of the error rate obtained in the previous iteration. The binary cross-entropy loss function for binary classification with a Multi-Layer Perceptron (MLP) is given by:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i))$$

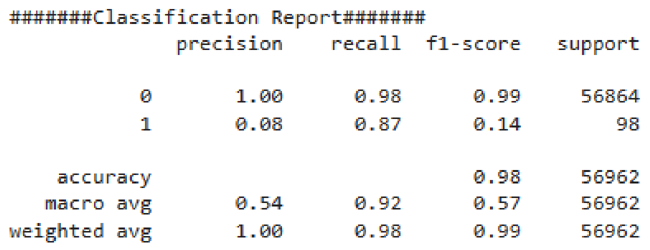
where:

- N is the number of samples,
- y_i is the true label of sample i (0 or 1),
- $\hat{y}_i$ is the predicted probability that sample i belongs to class 1 (range between 0 and 1).

Unique features:

• MLPs are highly flexible models capable of learning complex nonlinear relationships between input features and output targets

• MLPs efficiently handle large datasets using parallel computing resources and techniques like mini-batch gradient descent, enabling training on millions of samples

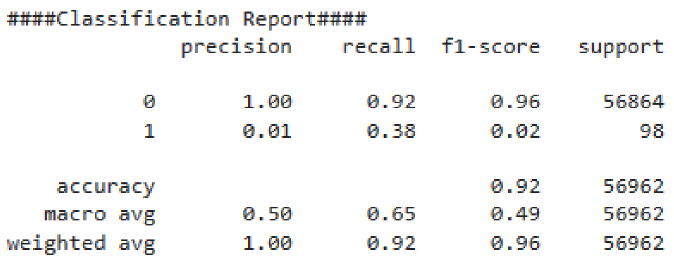


```
#######Classification Report#######
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     56864
           1       0.08      0.87      0.14        98

    accuracy                           0.98     56962
   macro avg       0.54      0.92      0.57     56962
weighted avg       1.00      0.98      0.99     56962
```

Fig. 9. classification report for Multi-layer perceptron after applying best threshold



```
####Classification Report####
              precision    recall  f1-score   support

           0       1.00      0.92      0.96     56864
           1       0.01      0.38      0.02        98

    accuracy                           0.92     56962
   macro avg       0.50      0.65      0.49     56962
weighted avg       1.00      0.92      0.96     56962
```

Fig. 10. Classification Report for PCA after using best threshold to obtain minimal false negatives

Implementation:

• Parameters and Training: The MLP model we designed has a learning rate of 0.001, it has 30 epochs and it has a batch size of 20. We used Adam optimizer since it is very adaptive in learning rate capabilities

• We split the data into train, validation and test dataset so that we can guarantee no overfitting and good prediction. The model is trained on the scaled training set and validated on a scaled validation set to monitor overfitting. Overfitting was monitored using the validation loss curve. Dropout layers and early stopping using model checkpoints were employed for regularization

• Each with 32 neurons and ReLU activation which is used for efficiency and ability to reduce the likelihood of the vanishing gradient problem. A dropout rate of 20% is applied to help prevent overfitting by randomly setting the outgoing edges of hidden units to 0

• Imbalanced Dataset Handling: The model demonstrates a reasonably high ROC AUC score on the test set (0.880). This suggests strong performance in distinguishing between positive and negative classes. Additionally, the low Brier score (0.0010) indicates good calibration of the model's predicted probabilities. These results showcase the model's effectiveness in handling imbalanced datasets, where fraudulent transactions are relatively rare compared to legitimate ones

Results:

Performance Metrics: The model exhibits moderately high recall (0.87) for the minority class (fraudulent transactions) suggesting that the model missed 13% of actual fraud cases (false negatives) which is critical. Precision is relatively low (0.08). This discrepancy indicates a high number of false positives, where legitimate transactions are incorrectly classified as fraudulent.The clasification report can be seen from Fig - 9

Multi-Layer Perceptron	**False Negatives**	**False Positives**
Using Best Threshold(Minimal False Negatives)	13	998

TABLE III
VALUES OBTAINED FROM CONFUSION MATRIX FOR MULTI-LAYER PERCEPTRON

The confusion matrix reveals a notable number of false positives (998), representing legitimate transactions predicted as fraudulent which can be seen from Table : III . Conversely, **false negatives** (fraudulent transactions incorrectly classified as legitimate) are relatively low, with only **13** instances which is still pretty impressive given our objective.

Unsupervised Learning Models:

D. *Principal Component Analysis*

Principal Component Analysis (PCA) in the context of fraud detection focuses on detecting anomalies which could be indicative of fraudulent transactions.Simple Principal Component Model has been applied aligning with the main objective of minimizing overall false negatives.

Implementation:

• The dataset is split into train dataset and test datasets to ensure the model can be independently validated. The class feature has been dropped to ensure unsupervised learning

• Similarly, scaling is done to ensure the PCA is not biased by any natural variance in the data.

• PCA is applied so that we retain 95% of the variance in the training data. This reduces dimensionality while capturing most of the data's information.

• The original data is reconstructed from data transformed by PCA. This step generates reconstruction errors by comparing original and reconstructed datasets.

Error is calculated and visualized. The histogram in the error shows that the right-tailed anomalies are potential fraud transactions. We experimented with various thresholds to classify transactions as normal or anomalous. To achieve this, we set a fixed threshold at the 99th percentile and employed a dynamic thresholding approach to minimize false negatives, while also ensuring that false positives remained below 10%.

Results:

Performance Metrics: The model exhibits moderately high recall (0.38) for the minority class (fraudulent transactions) suggesting that the model missed 62% of actual fraud cases (false negatives) which is critical. The precision of is very low at 0.01 which implies that some legitimate transactions were determined as fraud. The overall accuracy is 0.92 which is expected since that dataset is skewed.The classification report can be seen from Fig: 10.

The obtained false negatives and false positives from the confusion matrix can be seen from the Table IV-

The confusion matrix reveals a notable number of false positives (4631), representing legitimate transactions predicted as

Principal Component Analysis	False Negatives	False Positives
Using Best Threshold(Minimal False Negatives)	61	4631

TABLE IV
VALUES OBTAINED FROM CONFUSION MATRIX FOR PCA

fraudulent. Conversely, false negatives (fraudulent transactions incorrectly classified as legitimate) are relatively low, with 61 instances which is critical. The imbalance in misclassifications underscores the model's challenge in accurately identifying fraudulent transactions. **PCA exhibits a higher number of false positives and false negatives compared to other models such as XGBoost and Multi-Layer Perceptron.**

E. *Autoencoder*

Identified as a foundational structure within the field of neural networks, autoencoders are celebrated for their capacity to assimilate effective data depictions via unsupervised learning.

Fundamental principles:

• An autoencoder compresses input data into a latent space and reconstructs the original input from this representation.

• It consists of two components: an encoder transforms input data into the latent space, and a decoder reconstructs the input from the latent space.

• During training, the autoencoder minimizes the reconstruction error, usually using a loss function like mean squared error, to identify significant features in the data

Special characteristics:

• Independent learning: Autoencoders efficiently learn data representations without needing labels, making them ideal for scenarios with limited or no labeled data.

• Dimensionality reduction: By capturing a concise representation of input data, autoencoders effectively reduce its dimensionality, focusing on core features while removing noise and irrelevant details.

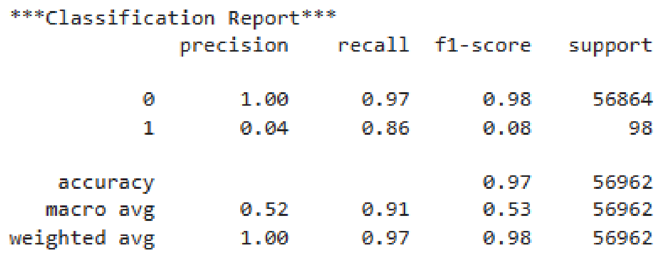
• Feature extraction: Autoencoders learn a hidden space representation that often reveals significant data features or patterns, useful for tasks such as data denoising, anomaly detection, and feature extraction.

• Creativity: Variations of autoencoders, like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs), can generate new data samples by drawing from learned latent space distributions, enabling tasks such as image creation and data synthesis.

Recognizing these core principles and unique traits, autoencoders stand out as a hopeful method for duties like anomaly detection in credit card operations. Here, identifying irregular patterns or fraudulent behavior without labeled data is vital. Autoencoders are capable of learning significant representations of regular transactions and can spot anomalies by pinpointing variations from the learned patterns.

Implementation and results:

Data Preparation: In a similar vein to XGBoost, the dataset is divided into training, validation, and testing sets. The data undergoes standardization through the StandardScaler() function



```
***Classification Report***
              precision    recall  f1-score   support

           0       1.00      0.97      0.98     56864
           1       0.04      0.86      0.08        98

    accuracy                           0.97     56962
   macro avg       0.52      0.91      0.53     56962
weighted avg       1.00      0.97      0.98     56962
```

Fig. 11. Classification Report for Autoencoder after using best threshold to obtain minimal false negatives

Designing the Autoencoder: We design an autoencoder model with Keras, which includes an input layer, a hidden layer (the encoder), and an output layer (the decoder).

Autoencoder Training: The autoencoder model is trained using the training dataset, employing mean squared error loss and the Adam optimizer for the process.

Choosing the Threshold: For this, the approach used is that to minimize false negatives while ensuring that the false positive rate does not exceed 10%. This optimal threshold derived from the previous method is utilized to distinguish between normal and anomalous data. This threshold is enforced on the validation set, followed by the generation of a confusion matrix to estimate false positives (FP) and false negatives (FN).

The calculation of False Negative Rate: The effectiveness of the model in detecting fraudulent transactions, while keeping the False Positive rate below 10%, is evaluated through the results yielded from the validation set. This is how the False Negative rate is computed. By using this approach, the best threshold obtained is 1.75. By using this best threshold, we can obtain the false positives and false negatives obtained from confusion matrix from Table : V as follows-

Autoencoder	False Negatives	False Positives
Using Best Threshold(Minimal False Negatives)	14	1874

TABLE V
VALUES OBTAINED FROM CONFUSION MATRIX FOR AUTOENCODER

While the autoencoder model achieved a minimal **false negatives** count of **14** with **1874 false positives**, it may not exhibit the same performance level as other supervised learning models such as XgBoost. However, it still demonstrates effectiveness in mitigating false negatives, which is noteworthy.

F. *XGBoost-Feature Selection*

Since Xgboost has shown the best performance among the models, by dropping features 'V13', 'V15' and 'V22' since these features did not show statistical difference among the classes as we have seen in **mann-whitney U test**, and performing same procedure again by taking below 10% false positive condition we can obtain best threshold again. Using best threshold, we can observe the false negatives and false positives from confusion matrix from the Table : VI

XgBoost- After Feature Selection	False Negatives	False Positives
Using Best Threshold(Minimal False Negatives)	6	3796

TABLE VI
VALUES OBTAINED FROM CONFUSION MATRIX FOR XGBOOST

We observed one less false negative, totaling to **6**, but the number of false positives increased significantly to **3796**. Once again, there exists a **trade-off between false negatives and false positives in pursuit of the best possible outcome**. In comparison to the previous outcome with XGBoost, where we only encountered 1395 false positives and 7 false negatives, the performance of the XGBoost model remains strong, whether with or without feature selection, when compared to other models.

VI. CONCLUSION-SUMMARY

All Models (Both Supervised and Unsupervised models)	False Negatives	False Positives
Balanced Random Forest Classifier	8	1386
XgBoost (Before applying best threshold)	17	5
XgBoost(After applying best threshold to minimize false negatives)	7	1395
XgBoost (After feature Selection applying best threshold)	6	3796
Multi-Layer Perceptron	13	998
Principal Component Analysis	61	4631
Autoencoder	14	1874

TABLE VII
FALSE POSITIVES AND FALSE NEGATIVES OBSERVED FOR ALL MODELS FOR BEST INSTANCES

A comprehensive analysis encompassing both supervised and unsupervised machine learning models was conducted to ensure their suitability for credit card fraud detection. Each model's characteristics and features were meticulously studied to ascertain their applicability. Extensive data analysis and preprocessing were undertaken to understand the distribution of features and their potential utility in parametric or non-parametric models.

For each model, rigorous analysis was conducted to prevent overfitting while addressing the class imbalance inherent in the dataset. Optimal parameters were applied to observe accuracies and, notably, to minimize overall false negatives. The approach employed to determine the best threshold aimed to minimize false negatives while ensuring that the false positive rate remained below 10%. Subsequently, the best thresholds for each model were utilized to minimize false negatives.

Of all the models assessed, XGBoost exhibited the most promising performance, with only **17 false negatives and 5 false positives** (prior to applying the best threshold). **After applying the best threshold**, XGBoost continued to outperform other models, with just **7 false negatives and 1395 false positives**. Notably, **XGBoost** achieved a **lowest false negatives count of 6** but incurred more false positives (3796) after feature selection after applying best threshold which is the minimal number of false negatives observed overall. **Balanced Random Forest Classifier** also demonstrated strong performance, with only **8 false negatives and 1386 false positives**. Among the unsupervised models, the **autoencoder yielded the best results, with just 14 false negatives and 1874 false positives**.While some models exhibit lower false negatives, they may incur higher false positives, and vice versa.This trade-off underscores the need for careful optimization of model parameters to strike a balance between false negatives and false positives, ultimately aligning with the objective of effective fraud detection while minimizing the impact of misclassifications. Our main objective in the analysis was to **minimize false negatives because failing to identify fraudulent transactions can have severe consequences**. In conclusion, **XGBoost** emerges as the top-performing model, achieving minimal false negatives, which aligns closely with the objective of minimizing false negatives in fraud detection tasks.

The **implementation** of all the above models can be found here - https://colab.research.google.com/drive/1gFZErqyXNC6F77siPh2hBfGwbBPwCCdK?usp=sharing

REFERENCES

[1] "Enhancing Credit Card Fraud Detection: An Ensemble Machine Learning Approach" 21 November 2023. https://doi.org/10.3390/bdcc8010006

[2] "A machine learning based credit card fraud detection using the GA algorithm for feature selection" 25 February 2022. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00573-8

[3] "Credit Card Fraud Detection Using XGBoost Algorithm" 01 March 2022. https://doi.org/10.1109/DeSE54285.2021.9719580

[4] "An Autoencoder Based Model for Detecting Fraudulent Credit Card Transaction" 16 April 2020. https://doi.org/10.1016/j.procs.2020.03.219

[5] "A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions" 14 January 2023. https://doi.org/10.1016/j.dajour.2023.100163

[6] "Machine Learning for Credit Card Fraud Detection" 30 March 2023. https://medium.com/@luuisotorres/machine-learning-for-credit-card-fraud-detection-1edf98efaf5a