Illinois Institute of Technology

CS484 Introduction to Machine Learning

ML algorithm for credit card fraud detection

Group Members

Sri Samhitha Bobba - A20541559

Lokareddy Prishitha Reddy – A2055934

Marina Oberemok - A20524168

Table of Contents

INTRODUCTION

Credit card fraud has become a major issue, raising concerns for financial institutions, credit card users, and businesses. The widespread acceptance of credit cards in place of cash or checks-in-person and online-made it easier for their convenience to be stolen, giving rise to several fraudsters taking advantage of the loopholes existing in several credit card payment system designs. Yet, with every increase in the volume of credit card transactions, losses due to fraud also rise in billions of dollars each year. The timely and accurate detection of such transactions could help blot out most of these losses, maintain confidence in the banking system, and assure safe financial worlds.

The reasons for selecting this topic are the continuous upsurge of chunk-and-matched patterns and organizing diversifying fraud incidents.

According to payments industry research company Nilson Report, in 2023, approximately 60% of credit card holders experienced some form of attempted fraud. Global card losses due to fraud reached billion in 2022, and are projected to rise significantly in the coming years. Apart from that, AI is now being groomed to expand the attacks launched by the fraudsters, making the present call for fraud detection solution tools that provide more rigorous insurances. Though financial institutions are investing vast sums these days for a new era of advancements in fraud detection capabilities, they have their prior fights within to ensure that while they keep their proficiencies intact, they don't miss out on the detection of every robbery attempt overpowered with false alarms.

In recent years, machine learning (ML) algorithms have gained prominence in credit card fraud detection because they can learn patterns from large datasets and predict fraudulent activities more effectively (Dornadula & Geetha, 2019). The astonishing premises of the class imbalance, drift-in-the-patterns-of-fraud, and the excessive false alarms bar adjustments done through other forms of machine learning.

Project objective

This project aims to apply machine learning for the detection of fraudulent credit card transactions by comparing the following algorithms such as logistic regression, K-Nearest Neighbors (KNN), and naïve Bayes (Afriyie et al., 2023; Varmedja et al., 2019), and compare their capabilities to improve accuracy, reduce false positives, and enhance the robustness of fraud detection systems.

Related work

Credit card fraud detection is a well-researched area in machine learning, with various algorithms applied to enhance accuracy and reduce false positives (Bin Sulaiman et al., 2022).

Different Supervised machine learning algorithms like Decision Trees, Naive Bayes Classification, Logistic Regression, and SVM are used to detect fraudulent transactions in real-time datasets (Dornadula & Geetha, 2019).

Logistic Regression (LR) is a classification algorithm used to predict binary outcomes (e.g., fraud or no fraud). It can handle both numerical and categorical variables, but it lacks the flexibility to model non-linear relationships effectively (Afriyie et al., 2023).

Decision Trees (DT) are non-linear classification and regression models that use a tree-like structure of decisions. They split the dataset into branches based on feature values, leading to a final prediction. They are easy to interpret but prone to overfitting.

This limitation of overfitting is mitigated by *Random Forests (RF)*, which aggregate multiple trees for robust predictions by reducing overfitting and is effective for both classification and regression tasks. However, RF can be biased toward variables with more levels, affecting performance on imbalanced datasets (Dornadula & Geetha, 2019).

To address data imbalance, resampling methods such as under sampling and oversampling are commonly used with Random Forest, XGBoost, and Decision Tree models proving effective approaches (Flondor et al., 2024).

Supervised algorithms *like Support Vector Machines (SVM), Gradient Boosting, Naïve Bayes (NB), and K-Nearest Neighbors (KNN)* have been explored for fraud detection, with Gradient Boosting and Random Forests showing high efficacy. Ensemble techniques like stacking classifiers further enhance prediction performance, achieving high accuracy and recall (Bin Sulaiman et al., 2022)

SVM is a supervised learning algorithm used for classification and regression. It finds the optimal hyperplane that separates data points from different classes by maximizing the margin between them, making it effective for high-dimensional data (Gyamfi & Abdulai, 2018).

Gradient Boosting is an ensemble learning technique that builds a series of weak learners (typically decision trees) sequentially, with each tree learning from the errors of the previous one.

It is highly effective for both classification and regression tasks and is widely used in competitions due to its high accuracy (Taha & Malebary, 2020).

K-means is an unsupervised learning algorithm for clustering data into K distinct groups based on some similarity. It minimizes intra-cluster distances to ensure that similar data points are grouped together. It is commonly used for segmentation and grouping tasks (Huang et al., 2024).

Naïve Bayes is a probabilistic classifier based on Bayes' theorem, assuming independence between features. It is efficient, works well with small datasets, and is commonly used for spam filtering and text classification (Gupta et al., 2021).

Dimensionality reduction algorithms, such as PCA, reduce the number of feature/set of the feature space and also contain important information. They help in reducing computational complexity and mitigating the curse of dimensionality (Choi et al., 2005).

Deep learning models, including neural networks, have also been applied, achieving significant results but requiring large datasets and being computationally expensive (Lei et al., 2023).

Machine learning algorithms can detect fraudulent transactions, classify them, and potentially stop the transaction if needed. Credit card fraud detection involves modeling past transactions, including records of fraudulent activity, to predict whether new transactions are genuine or fraudulent (Bontempi, 2021).

METHODOLOGY

1.The Dataset

1. The dataset contains transactions made by credit cards in September 2013 by European cardholders (Kaggle, 2013). The dataset is highly unbalanced, the positive class (frauds) accounts for 0.172% of all transactions.
2. 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 which have not been transformed with PCA are 'Time' and
3. 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

As evident, frauds represent such a minute percentage that they are scarcely visible in the chart above. For the same reason, we would have to bring in some data balancing techniques to increase the performance of our machine learning model.

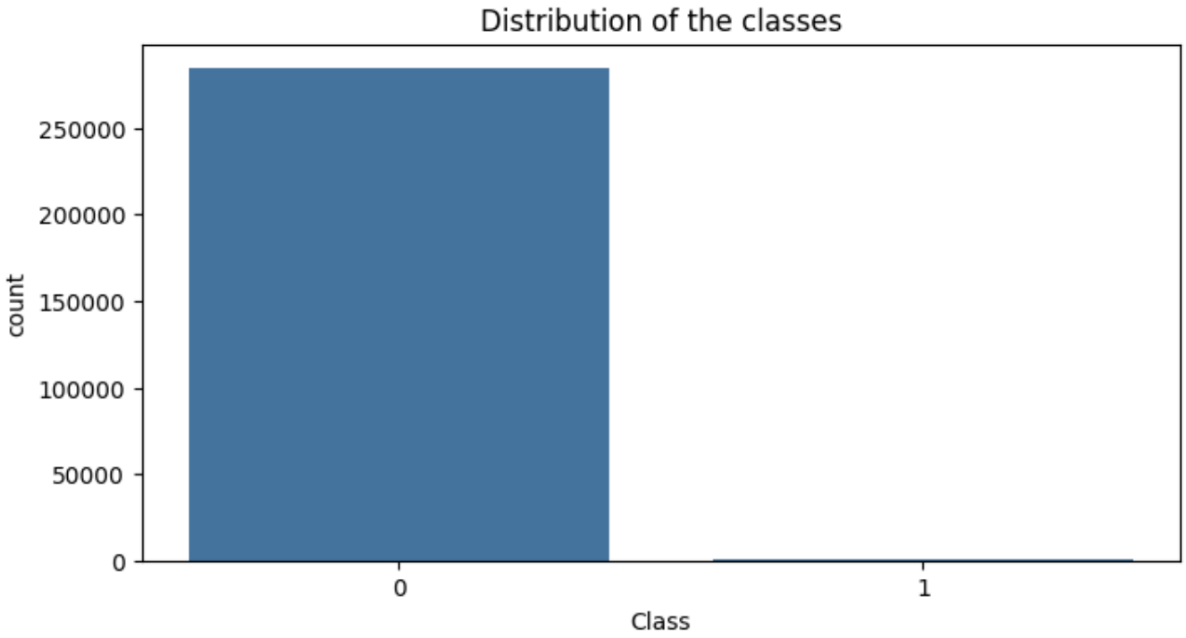

Figure 1. Distribution of the classes

Given the class imbalance ratio, we recommend measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.

2 .Investigating normal and fraudulent transaction patterns

Prior to this, we'll examine whether there exists any correlation between transaction amount and its legitimacy, or if there are specific time periods more susceptible to fraudulent activity. To accomplish this, we'll generate histograms to visualize the distribution of values for the Time and Amount columns in both normal and fraudulent transactions.

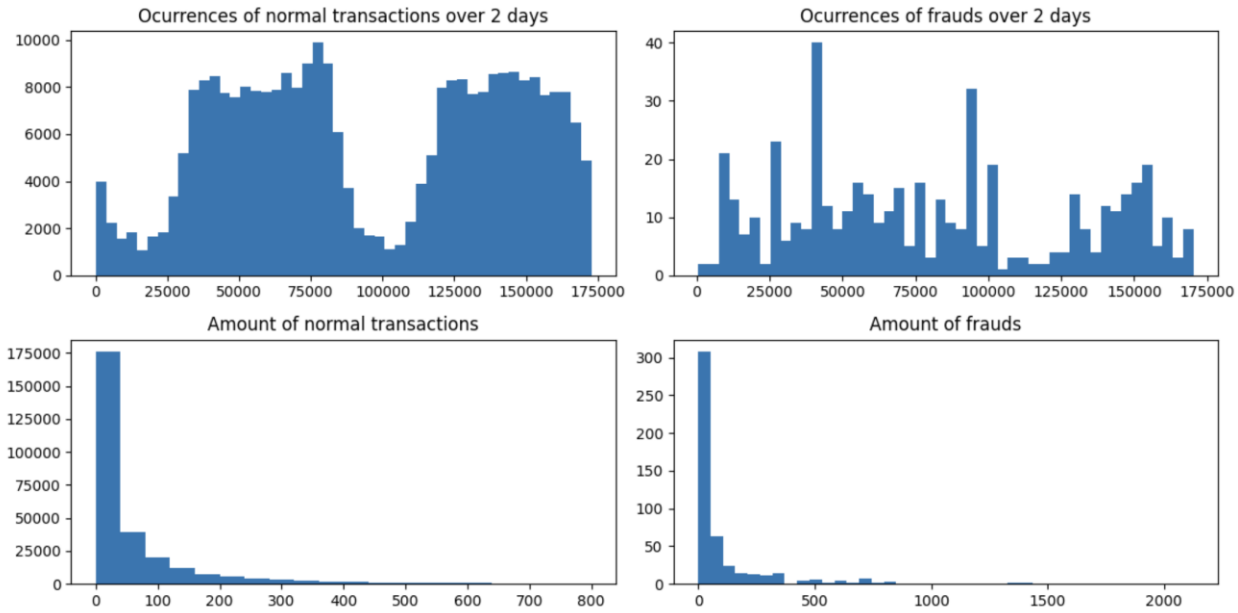

Figure 2. Distribution of values for the Time and Amount columns

A distinct pattern is noticeable in the upper left chart, depicting the volume of normal transactions concerning the time of occurrence. Since the time of the initial transaction was not provided, nor was it specified whether it occurred on a business day or weekend, it is presumed that these transactions transpired over two weekdays, given the similarity in patterns observed on both halves of the chart. Furthermore, the first transaction likely took place close to midnight, as indicated by the decline in transaction volume, followed by a resurgence starting from the 6th hour at the onset of the day.

Regarding illegitimate transactions, they appear to mirror the patterns observed in normal transactions, albeit less clearly, possibly due to the low frequency of fraudulent activity.

To compare the amounts involved in both types of transactions, we will generate a box plot, which presents the averages, medians, and quartiles.

From the graph below, we observe that the average value for frauds (represented by the small green triangle) is slightly higher than that for normal transactions. Additionally, in the case of normal transactions, the distribution of values appears to be more dispersed. However, this could be attributed to the relatively low volume of frauds compared to legitimate purchases.

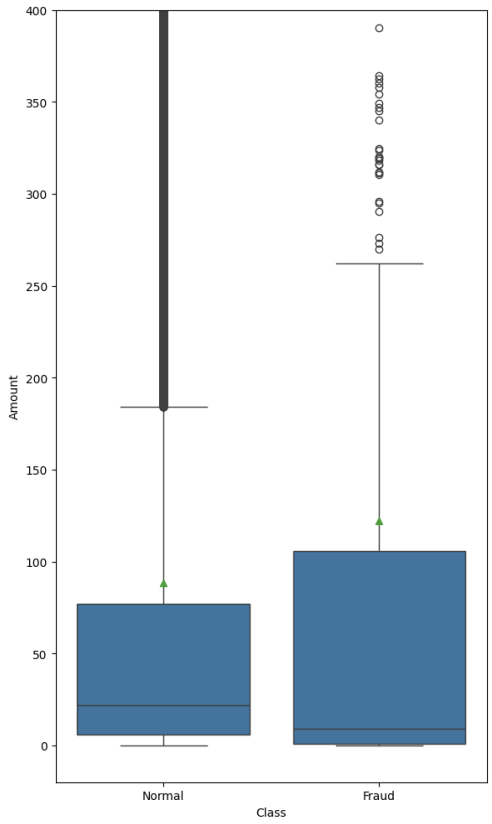

Figure 3. Distribution of the classes

3.Analyzing the relationship between variables

Next, we will examine whether there exists any correlation among the various variables in this dataset by plotting a correlation matrix.

The heatmap depicted below reveals a weak correlation among the variables, with only a slight correlation observed between the variables V2 and Amount. Now, let's delve into the behavior of the variables from V1 to V28 concerning the Class variable, distinguishing between instances where Class equals 1 (indicating fraud) and when its value is 0 (indicating normal transactions). To accomplish this, we will generate charts known as "Kernel Density Estimation", which aim to estimate the probability density function (PDF) of each of these variables.

Some variables, such as V3, V4, V11, and V14 present a behavior very different according to the type of transaction. As for variables such as V15, V22 and V26, they present a similar behavior. We've already seen that data are very imbalanced and Amount and Time variables are not standardized.

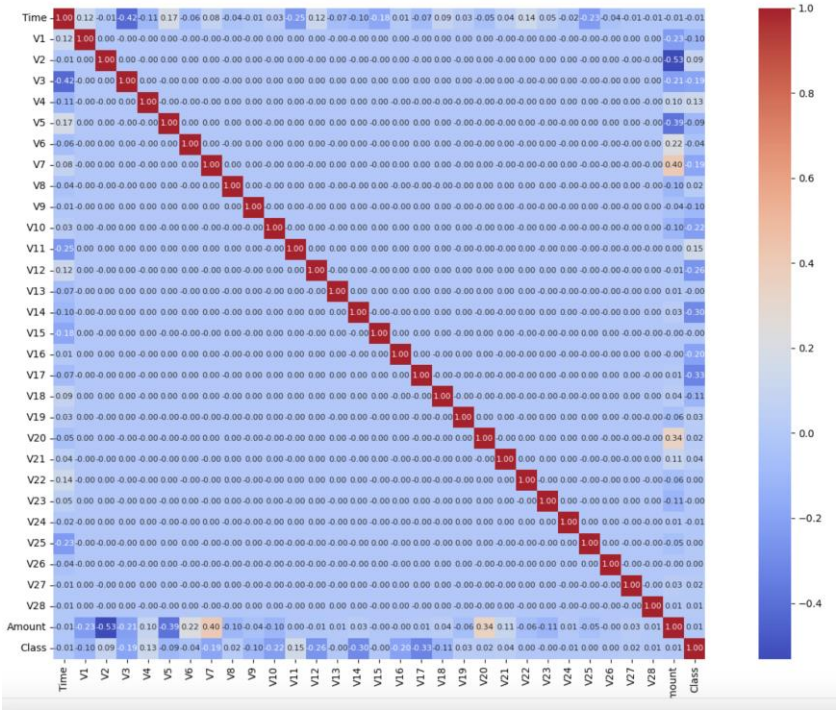

Figure 4. The correlation among the variables

4. Data preprocessing

1) Standardization.

Firstly, we will standardize the columns that did not undergo the PCA treatment. Standardization involves adjusting the values in each column to have a mean of 0 and a standard deviation of 1. To perform data standardization, we will utilize the StandardScaler class.

2) Splitting into training and validation set

We will split the data into two groups, training and test groups, with the intention of to check if the balancing process, was made properly.

3) Balancing

In situations of data imbalance, such as the present dataset, our machine-learning model may suffer from inaccuracies. Thus, the present study will take on the understudying problem by sampling, undersampling. Undersampling consists of taking a random subset of the majority class without rejecting all instances of the minority class

Now, with data balanced, let's plot the correlation matrix among the variables:

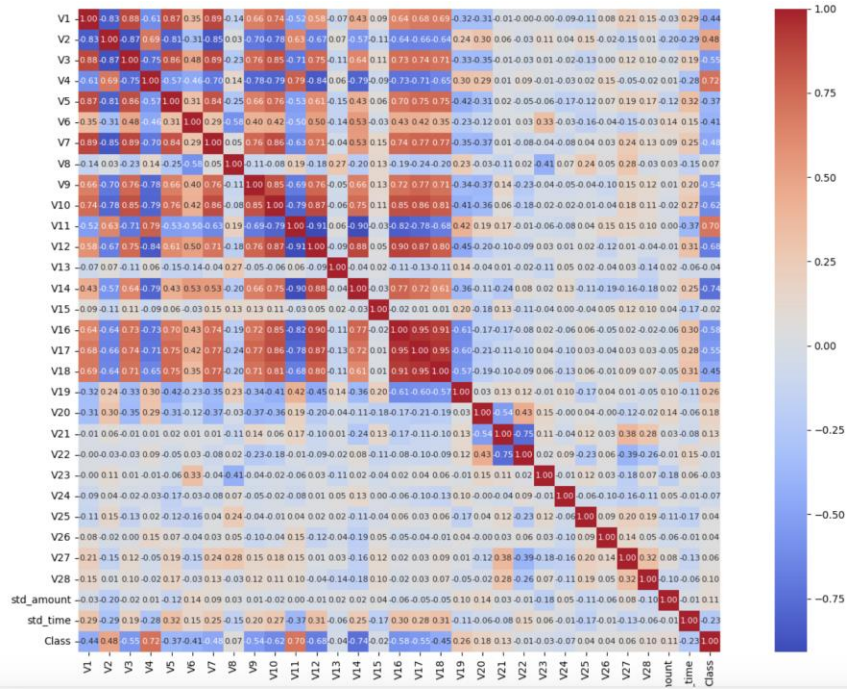

Figure 5. The correlation among the variables (balanced)

5.Testing Predictions on the Test Set

During the execution of code for making predictions, it was observed that performance metrics varied each time due to the random nature of training/test splitting. To obtain more reliable results, all machine learning predictions were run 10 times, and the median of these executions was extracted.

At the outset of this project, we set aside a portion of the data for testing and subsequently subdivided the remaining data into training and validation sets. Before proceeding, we need to normalize the test data as we did with the other datasets.

6.Logistic Regression

With all the data prepared, let's proceed to build a classifier model based on logistic regression. This model will be trained using the X_rus and Y_rus data. Subsequently, we will make predictions on the test data.

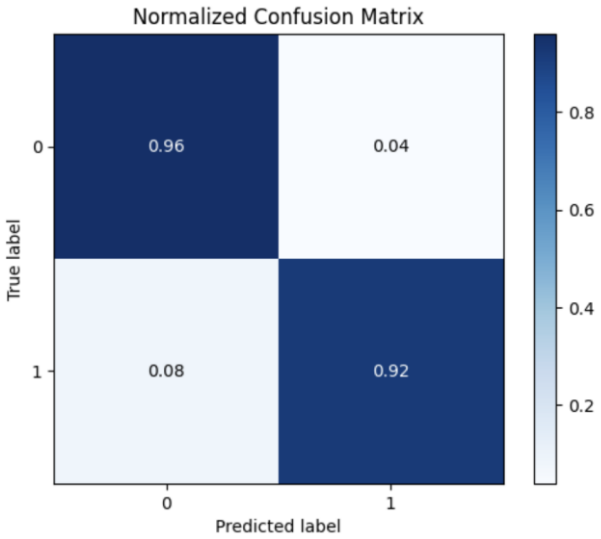

Figure 6. Confusion matrix for logistic regression (test data)

The model demonstrated an average detection rate of 89% for fraudulent transactions; however, it exhibited numerous false positive predictions, resulting in a notably low precision of less than 5%.

Final Results: Accuracy: 96.2% Precision: 4.1% Recall: 90.7% F1-score: 0.083 AUC: 0.937.

7.Naïve Bayes

We will commence by executing the Naive Bayes model.

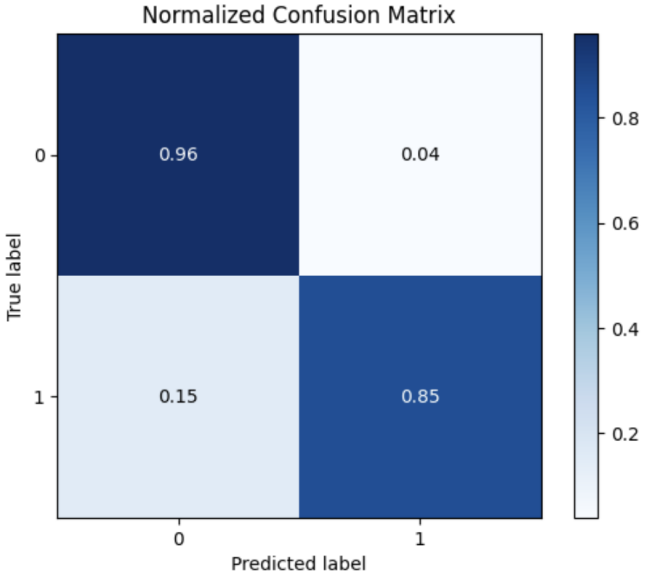

Figure 7. Confusion matrix for Naive Bayes

The Naive Bayes model exhibited the lowest performance among all tested models. It also suffered from a high number of false positives, although it was not the worst performer in this regard.

Final Result: Accuracy: 96.9% Precision: 4.7% Recall: 86.2% F1-score: 0.088 AUC: 0.919.

8.K-Nearest Neighbors

Next, let's explore the K-Nearest Neighbors model.

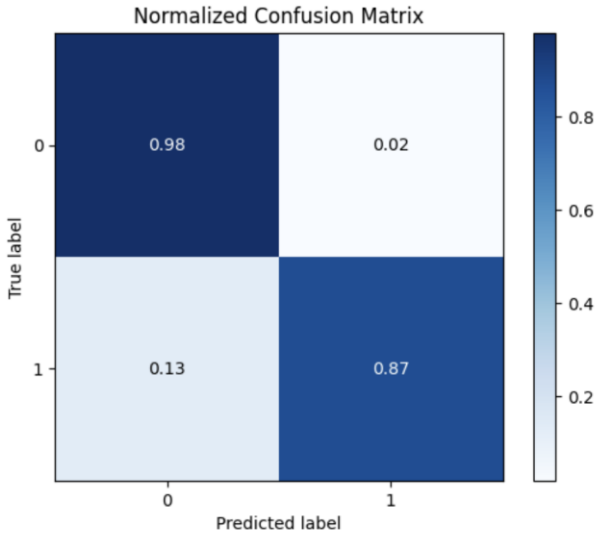

Figure 8. Confusion matrix for KNN

Overall, this model achieved the highest accuracy and precision among all models that utilized the under-sampling technique. However, its capability to accurately detect frauds was only marginally better than the Naive Bayes model, which performed the worst in this regard.

Final Results: Accuracy: 98.4% Precision: 8.7% Recall: 87.6% F1-score: 0.162 AUC: 0.926.

9.Decision Tree

Lastly, we will develop a decision tree machine learning model.

This approach yielded the lowest accuracy and precision among all methods, both less than 3%; however, it excelled in accurately identifying positive results, with a precision of 92.3%.

Final Results: Accuracy: 93.3% Precision: 2.3% Recall: 92.3% F1-score: 0.046 AUC: 0.923.

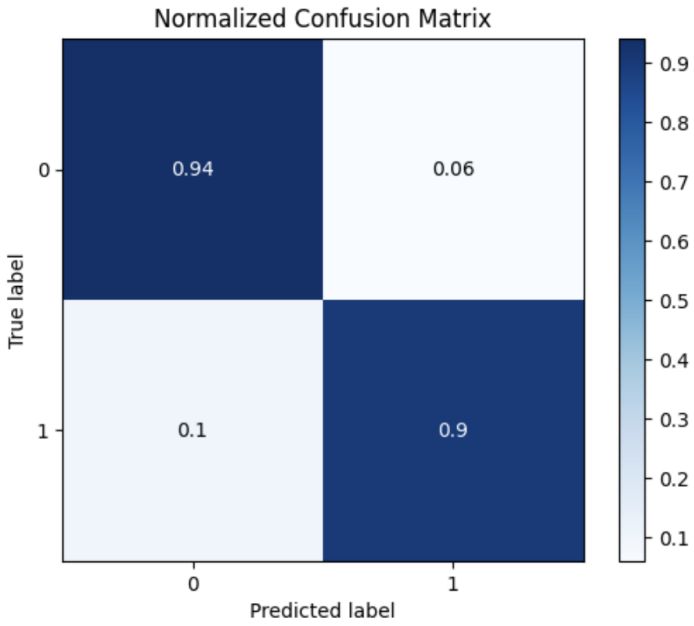

Figure 9. Confusion matrix for Decision tree

10.Logistic Regression using Over-sampling

Up to this point, all models have utilized a dataset balanced using the under-sampling technique, which involves removing records from the majority class. Now, we will employ another technique called "over-sampling," which involves creating dummy records for the minority class. Our focus is on the improvement of performance metrics, with precision improvement highlighted for not being well in any of the previously heterogeneously designed models.

To implement over-sampling, we will use an algorithm named SMOTE, which generates samples of the minority class using the K-Nearest Neighbors method.

Initially, we will balance the data.

The application of the over-sampling technique has achieved quite a significant decrease in false negatives, thus pushing precision to a value of 97% for this model.

Final Results: Accuracy: 94.6% Precision: 97.0% Recall: 92.2% F1-score: 0.945 AUC: 0.946.

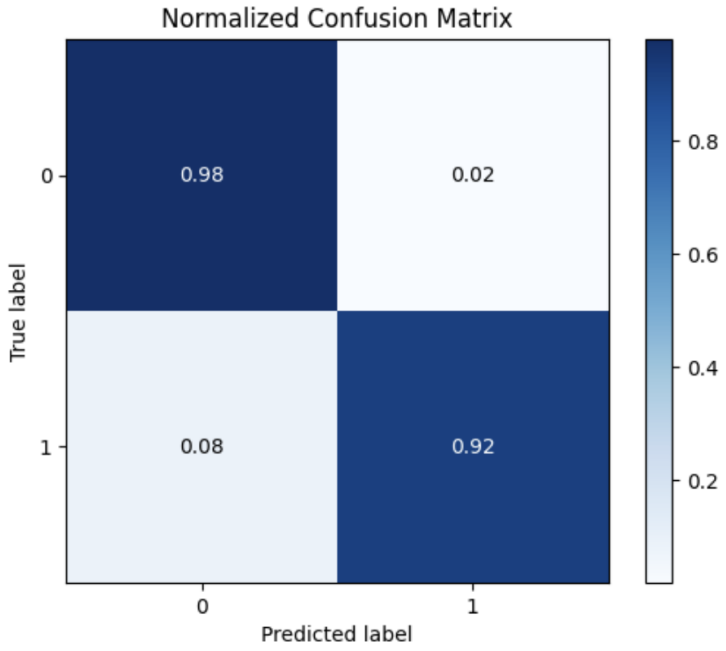

Figure 9. Confusion matrix after oversampling

Finally, let's evaluate the performance of our model on the original test data.

When testing the ML model with logistic regression and under-sampling, the ability to detect fraudulent transactions improved slightly, but precision deteriorated once again.

Final Results: Accuracy: 97.3% Precision: 5.7% Recall: 94.5% F1-score: 0.108 AUC: 0.961

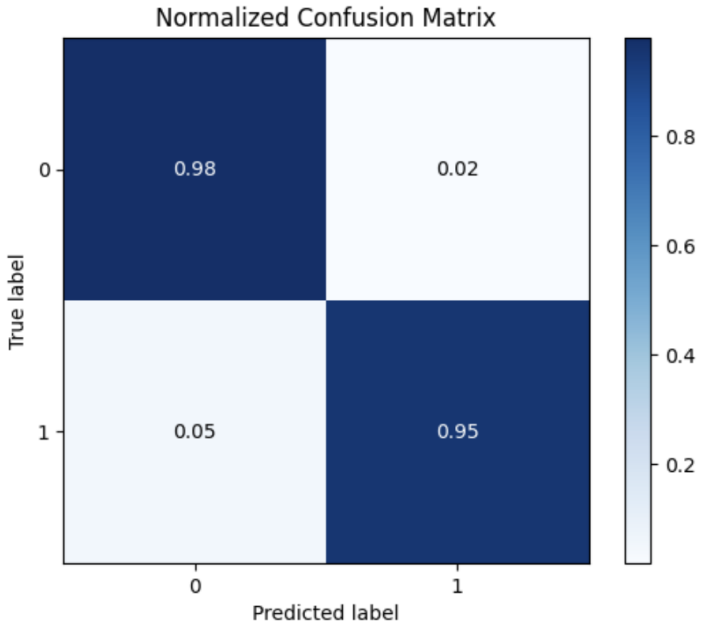

Figure 10. Confusion matrix after oversampling on the original test data

11.Comparison of all machine learning methods

Below is a summary comparison of the performance of all machine learning models used in this project. The worst value in each measure is marked in red, while the best one is marked in green, considering only the predictions made upon validation data.

ML Model	balancing method	Prediction data	Accuracy *	Precision	Recall	F1-Score	AUC	Positives	Negatives
Logistic regression	under-sampling	Validation	0.9619	0.0406	0.9075	0.0779	0.9367	105	60417
Naive Bayes	under-sampling		0.9690	0.0455	0.8716	0.0865	0.9206	105	60417
K-Nearest-Neighbours	under-sampling		0.9836	0.0853	0.8819	0.1552	0.9324	105	60417
Decision tree	under-sampling		0.9327	0.0235	0.9226	0.0458	0.9228	105	60417
Logistic regression	over-sampling		0.9459	0.9702	0.9217	0.9446	0.9459	105	60417
Logistic regression	under-sampling	Test	0.9630	0.0428	0.9257	0.0819	0.9499	74	42647
Logistic regression	over-sampling		0.9730	0.0575	0.9459	0.1083	0.9609	74	42647
		* For all measures median of the 10 executions appear on summary							

Figure 11. Comparison of the applied ML algorithms

Regarding predictions made on the test data, the model that performed the best was a logistic regression with over-sampling, surpassing the one using under-sampling in all performance measures.

CONCLUSION

Overall, there were any differences among the methods in correctly identifying frauds. Based on the conducted research, we can come to the following conclusions:

- Logistic Regression with SMOTE (over-sampling) provided the best precision-recall and overall AUC, with precision improving significantly over the under-sampling approach.
- K-Nearest Neighbors (KNN) achieved high recall but had limited precision, resulting in a lower F1 score due to false positives.
- Decision Tree and Naive Bayes struggled with low precision, even though they detected most fraud cases (high recall).

The primary weakness of all these models is their tendency to output many false positive predictions. Only logistic regression with over-sampling demonstrated good performance in this regard, but only on balanced data. When applied to the original test data, which was imbalanced, this model also generated numerous false positive predictions. Addressing this issue will likely require the utilization of more advanced ML models to reduce the occurrence of false positives.

Contributions

Sri Samhitha Bobba was responsible for data preprocessing and analyzing the results

Lokareddy Prishitha Reddy implemented the three machine learning algorithms for fraud detection.

Marina Oberemok conducted a literature review on related works, researched the implementation of ML algorithms for fraud detection, compared the algorithms' outcomes, and created the presentation and report.

References

1. Afriyie, J. K., Tawiah, K., Pels, W. A., Addai-Henne, S., Dwamena, H. A., Owiredu, E. O., Ayeh, S. A., & Eshun, J. (2023). A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions. *Decision Analytics Journal*, *6*, 100163. https://doi.org/https://doi.org/10.1016/j.dajour.2023.100163
2. Bin Sulaiman, R., Schetinin, V., & Sant, P. (2022). Review of Machine Learning Approach on Credit Card Fraud Detection. *Human-Centric Intelligent Systems*, *2*. https://doi.org/10.1007/s44230-022-00004-0
3. Bontempi, G. (2021). Reproducible machine learning for credit card fraud detection-practical machine learning for credit card fraud detection-practical handbook foreword. May. In: May.
4. Choi, S. W., Martin, E. B., Morris, A. J., & Lee, I.-B. (2005). Fault detection based on a maximum-likelihood principal component analysis (PCA) mixture. *Industrial & engineering chemistry research*, *44*(7), 2316-2327.
5. Dornadula, V. N., & Geetha, S. (2019). Credit Card Fraud Detection using Machine Learning Algorithms. *Procedia Computer Science*, *165*, 631-641. https://doi.org/https://doi.org/10.1016/j.procs.2020.01.057
6. Flondor, E., Donath, L., & Neamtu, M. (2024). Automatic Card Fraud Detection Based on Decision Tree Algorithm. *Applied Artificial Intelligence*, *38*(1), 2385249. https://doi.org/10.1080/08839514.2024.2385249
7. Gupta, A., Lohani, M., & Manchanda, M. (2021). Financial fraud detection using naive bayes algorithm in highly imbalance data set. *Journal of Discrete Mathematical Sciences and Cryptography*, *24*(5), 1559-1572.
8. Gyamfi, N. K., & Abdulai, J.-D. (2018). Bank fraud detection using support vector machine. 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON),
9. Huang, Z., Zheng, H., Li, C., & Che, C. (2024). Application of machine learning-based k-means clustering for financial fraud detection. *Academic Journal of Science and Technology*, *10*(1), 33-39.
10. Kaggle. (2013). *Fraud Detection.*
11. *Anonymized credit card transactions labeled as fraudulent or genuine*. https://www.kaggle.com/datasets/whenamancodes/fraud-detection
12. Lei, Y.-T., Ma, C.-Q., Ren, Y.-S., Chen, X.-Q., Narayan, S., & Huynh, A. N. Q. (2023). A distributed deep neural network model for credit card fraud detection. *Finance Research Letters*, *58*, 104547. https://doi.org/https://doi.org/10.1016/j.frl.2023.104547
13. Taha, A. A., & Malebary, S. J. (2020). An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine. *IEEE Access*, *8*, 25579-25587.
14. Varmedja, D., Karanovic, M., Sladojevic, S., Arsenovic, M., & Anderla, A. (2019, 20-22 March 2019). Credit Card Fraud Detection - Machine Learning methods. 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH),