PREDICTIVE SYSTEM FOR FRAUDS DETECTION

July 18, 2025

Disclaimer The work proposed here is published for training purposes and does not want to advise third parties on how to cope with financial risk.

Author Eros Masarin

Software used *Jupyter notebook. Libraries: numpy, pandas, matplotlib, sklearn.tree, sklearn.metrics, SMOTE, RandomUnderSampler, RandomForestClassifier, sickit-learn, imbalanced-learn, scipy, joblib, time, sklearn.preprocessing, tensorflow.keras.models, tensorflow.keras.layers.*

Link to code (optional) *Github repository*

Abstract

Fraudulent credit card transactions are a serious matter. They cause millions euro of losses each year and loss of trust of card holders towards financial institutions. To prevent this risk, entities can adopt a proactive approach by using a predictive system that detects and blocks doubtful transactions as a preventive measure. The purpose of this research is to explore a predictive approach to fraud detection by comparing the performance of different classifiers, identifying the features that classify a transaction as fraud, and quantifying the cost of missing a fraud or mistakenly classifying a legit transactions as fraudulent.

Keywords: Credit card fraud, Predictive analytics, Cost of misclassifications, Decision Tree, Neural Network, Random forest, SVM, Logistic regression, Training dataset, Testing dataset, Validation set, Grid Search.

1 Introduction

Fraudulent transactions have a negative impact on the balance sheet of financial institutions and their corporate image. Millions of euros are lost every year due to fraud. The 2024 report on payment fraud published by the EBA (i.e., European Banking Authority) (1) estimated fraud losses of Euro 4.3 billion in 2022 and Euro 2 billion for the first half of 2023, with credit transfers (Euro 967 million) and card payments (Euro 537 million) the main liability bearers. Financial loss is only one side of the problem, the other being reputational damage. In fact, potential clients would certainly not open a new savings or investment account with a bank that cannot cope with the risk of fraud. The development of a predictive analytic system that classifies transactions as fraudulent or non-fraudulent sounds compelling to financial institutions to address the identified issues.

As the number of frauds increases with bank operations, it is natural to ask how fraudsters can gain access to consumer data. According to Bhattacharyya (2), credit card fraud generally falls into two categories: application and behavioral fraud. The former occurs when a fraudster is impersonating someone by using false information to gain access to credit cards. The latter is categorized into four types: stolen or lost card, mail theft, counterfeit card, and "card holder not present" fraud. Although machine learning cannot tackle social engineering, it can be used as a predictive tool to learn complex patterns and detect suspicious activity. Due to the nature of the abstract task at hand (i.e. a classification problem), five supervised models are developed: decision tree, random forest, support vector machine, logistic regression and neural network. This variety allows for a better exploration of the performance. Other goals include quantifying the cost of misclassifications and identifying relevant features for fraud detection.

2 Related works

Decision trees are popular for classification problems in the financial and healthcare industries. For instance, Yu *et al.* (3) predict the better or worse health wellness of elders with decision trees based on a number of features, namely heart rate, sleep duration, steps, body temperature, blood pressure, blood oxygen level. The classifier was then used to target the most vulnerable senior citizens class through the adoption of prevention programs. In the financial industry, Patil *et al.* (4) designed a framework for fraud detection that consists of a Hadoop network, which stores and converts raw data and decision trees to classify frauds. The research analyzes 1000 transactions completed with credit cards issued in Germany. The results are encouraging, with an accuracy greater than 70%, but the data set considered is balanced. This represents a limitation, as frauds are not as frequent as legitimate transactions in the real world. The problem of class imbalance was examined by Afriyie *et al.* (5), who tested more than half a million simulated credit card transactions in 2020 with only 0.4% of them being fraudulent. To handle the imbalance, the authors made use of the synthetic minority oversampling technique, an approach also used in this project to handle the imbalance.

One of the first uses of the neural network to detect frauds was explored by Ghosh and Reilly (6), who used a P-RCE neural network that operates purely in a feedforward manner. Such a model uses a more traditional way of feedforward neural network through backpropagation, with functions that can iteratively adjust the weights of the connected neurons. This increases the model complexity, it allows to learn non-linear relations in the dataset and improve the ability to tackle new fraud techniques.

As fraudulent activities evolve in complexity, traditional rule-based fraud detection methods struggle to keep up with sophisticated attack strategies. Consequently, machine learning-based approaches have emerged as powerful alternatives, offering improved accuracy and adaptability in identifying fraudulent transactions. In addition to neural networks and decision trees, the random forest classifier (RF) has gained attention due to its robustness, interpretability, and ability to handle high-dimensional datasets. RF operates by constructing an ensemble of decision trees, each trained in random subsets of data, and aggregating their predictions to reduce overfitting while improving generalization, an approach explored by Breiman (7). Liu *et al.* (8) demonstrated that fraud detections trained with random forest outperform traditional parametric models. This is done by capturing complex fraud patterns and key financial indicators, namely the Debt-to-Equity Ratio (DEQUTY), which was identified as a critical predictor in the study.

In addition to the ensemble method, logistic regression is a widely used method for classification problems, largely due to its interpretability and ease of deployment in real-world environments. However, as highlighted by Li *et al.* (9), logistic regression struggles with highly imbalanced datasets, a typical situation in fraud detection, where fraudulent transactions account for less than 1% of the data. To address this, Li *et al.* (9) propose combining logistic regression with data preprocessing, feature selection, and threshold tuning, allowing proper optimization of the trade-off between false positives and false negatives, depending on the specific need. Furthermore, Dal Pozzolo *et al.* (10) emphasize the need for sampling techniques, namely SMOTE (i.e. Synthetic Minority Oversampling Technique) and Random Under-Sampling, to improve model performance on imbalanced datasets.

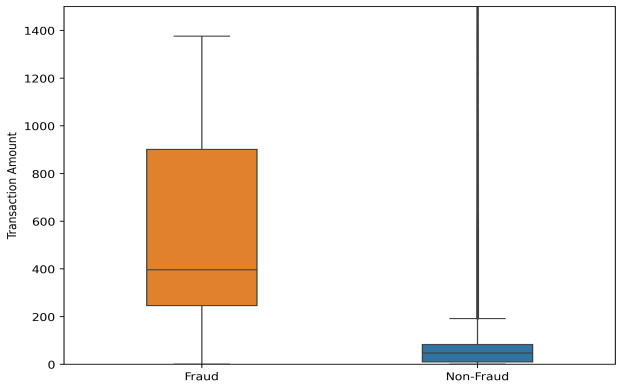
3 Research Method

The predictive system for frauds is trained and tested with two datasets available on Kaggle (11). These datasets are popular among the machine learning community, with almost 70,000 downloads and just under 500,000 views. The training set and the test set comprise simulated legitimate and fraudulent transactions in the period 1 January 2019 - 31 December 2020. Transactions are completed by a pool of 1000 customers and 800 merchants, mainly in the USA. The training set contains 1,296,675 entries, while the testing set contains 555,719 instances. These numbers ensure that the classifier captures uncertainty and leaves out statistical testing, which is needed with fewer than 1000 instances to see if the observed effect is genuine or random. In fact, with more than a million instances, observed correlations are more likely to be true and not spurious. In Appendix Table 9 a total of 23 features are included, including but not limited to demographic information and details of transactions. The data is labeled with a specific target variable that discriminates between fraudulent transactions and legitimate ones. The variable *is fraud* must be learned by the model. The logical steps to make data available for training are data profiling, cleansing, transformation and reduction. This is the recipe for the data preprocessing discussed next.

3.1 Data Preprocessing

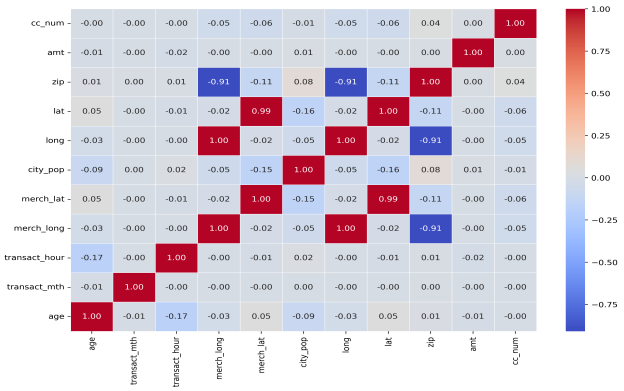
The data profiling reveals that there are no missing data, yet a large class imbalance can be seen. In fact, in Appendix Fig. 8, only 0.58% of transactions are fraudulent. No differences can be seen with gender in Appendix Fig. 9. Feature engineering is used to perform a temporal analysis. New features of hour, month and age are derived from the original features. The age of the consumer may reveal some surprises. For instance, by looking at frauds by age in Appendix Fig. 10 one could ask how a 14-year-old person can complete a transaction with a credit card. In addition, Appendix Fig. 11 suggests that frauds are more likely at certain hours of the day (10pm-3am).

Figure 1: Fraud/Non-fraud amount



A large amount of noisy data (i.e. outliers) was detected in the amount of transactions labeled as *non fraud* in the training dataset. This is expected since transactions can vary significantly in their amount. However, no outliers were identified in the number of transactions labeled *fraud*, whose median value is $400 in Fig. 1.

Figure 2: Correlation of features



Interestingly, the correlation matrix for the quantitative variable in Fig. 2 reveals how the characteristics of the longitude, latitude and zip are closely correlated. Due to this correlation, these features strongly influence the performance of the model. Before discarding these features, data transformation is performed to transfer the latitude and longitude coordinates in the district of the region within or outside the USA. Next, one-hot coding is applied to the newly created feature of regions. The process gives rise to 10 geographical locations for the customer and the merchant, while the old geographical features are discarded.

The reasons for this data transformation are threefold. Firstly, it solves the correlation problem. Secondly, one-hot coding avoids ordering unordered data, which helps the classifier to discriminate between fraud and non-fraud cases by selecting those features that maximize the information gain. Lastly, this process allows to uncover hidden insights. For instance, a graphic analysis in Appendix Fig. 12 reveals how frauds are more likely in New York, Virgin Islands and the Southeast.

For the same reasons above, one hot coding is also applied to the categories of shopping, with 14 new features generated with the process. A visual inspection in Appendix Fig. 13 reveals that frauds are likely to occur in the categories of grocery and shopping. Data transformation is also applied with binary encoding for the feature of gender, while percentage encode is used to normalize the features of credit card number and merchant. In fact, knowing how frequently a card number appears or how common a merchant is might add a predictive value to the classifier. Lastly, data reduction is done to remove high-cardinality features, namely zip, longitude, latitude, and features from which relevant data have been extracted and may be a concern for privacy, for example date of birth, first and last name.

The cleaned data, whose final attributes can be seen in the Appendix Table 10, represents the basis for training models.

3.2 Baseline: study of Afriyie et al.

To compare the performance of the models designed to address the research problem, the three models developed by Afriyie *et al.* (5) are used as a baseline. Indeed, the training dataset and testing dataset used by the authors are the same datasets used in this research. In addition, the authors employ 3 of the five models used in this research: logistic regression, decision tree and random forest. Such models are used in this study following a different training process explained in the following sections. Thus, the comparative study represents a strong baseline due to the same source of data for modeling and the similar choice of classifiers.

3.3 Decision tree

Decision tree models non-linear relationships between numerical and categorical features. In this project, the ID3 algorithm is used to iteratively divide the training data set into subsets until all records belong to the same class, *fraud* or *non-fraud*, or have identical attribute values. The algorithm preserves the order property of ordinal attributes, it is robust to outliers identified in the amount of *non-fraud* transactions in Fig. 1 and it is simple and easy to understand. In fact, it allows for quick identification of the attributes that best determine the split at each node. In order to determine the best split at each node, ID3 uses the impurity measure at each node, which is computed as the weighted impurity of the child nodes. The impurity measure used by the algorithm is the Gini index in Equation 1. Other algorithms make use of the entropy in Eq. 2, which gives marginal improvements, but is more computationally expensive than the Gini index due to the logarithmic function. Due to the large number of instances in the dataset, the Gini index is used.

$$\text{Gini Index} = 1 - \sum_{i=1}^{c} p_i(t)^2 \tag{1}$$

$$\text{Entropy} = -\sum_{i=1}^{c} p_i(t) \log_2 p_i(t) \tag{2}$$

where $p_i(t)$ represents the frequency of class i at node t, c is the total number of classes.

Gini index ranges between values [0, 0.5]. At each node, ID3 chooses the attribute test condition that produces the lowest measure of the Gini index or the highest gain of the impurity measure. The Information Gain in Equation 3 measures how well a particular feature separates the data in subset: the higher, the better. In Eq. 3, F represents the feature to split, S is the set of instances before the split, index v iterates over the values of the feature F, S_v represents the instances of S that have value v for the feature F, while $H(S)$ is the entropy of class distribution among the instances in the set S.

$$\text{IG}(F) = H(S) - \sum_{v \in \text{F}} \frac{|S_v|}{|S|} H(S_v) \tag{3}$$

Surprisingly, Tal *et al.* (12) remarked that the choice of impurity has little effect on the performance of the induction algorithm. Far more important is the strategy used to prune the tree to avoid overfitting. Indeed, decision tree has the tendency to overfit the training data: as the number of features chosen as nodes in the tree increases (i.e. the tree depth increases), the model becomes more and more complex and test errors can start increasing even though training error may decrease. Mitchell (13) suggests to prune decision nodes, which consists of removing the subtree rooted at that node, making it a leaf node, and assigning the most common classification of the training examples. Choosing the right tree depth to prune to increase model performance is a hard task without the use of validation sets. Moreover, the training dataset is highly imbalanced (Fig. 8), which means that the algorithm spends more time learning on the majority class, instead of learning what the patterns of fraudulent transactions are.

The problems at hand are two then: handle the class imbalance and find the right depth of the tree to avoid overfitting. It turns out that the use of validation sets is the answer for both problems. If the training set is split into validation sets and tested with different values of tree depth, then it is possible to know the optimal number of features to use as tree nodes. To address the imbalance, the majority class *non-fraud* is undersampled, while the rare class *fraud* is oversampled with SMOTE by finding small clusters of points in the minority class and generating their mean as a new minority class point.

The size of the majority class is set to be equal to the size of the minority class. The correct resampled size of the minority class works as *hyperparameter*. In Fig. 8, the number of frauds is approximately 7,500. Thus, a reasonable assumption is to allow the hyperparameter to vary from a slightly larger value to approximately twice its initial value. In other words, the number of fraudulent instances that the algorithm should learn to detect equals the number of legit instances, and such a number - called *hyperparameter*- can vary between the discrete interval [8,000 - 15,000] by steps of 500. This means that 15 validation sets must be created from the original training dataset in order to test separately the choice of the *hyperparameter* on the model performance. In fact, the model performs differently and the optimal tree depth changes with a different sample size of the majority and minority class. To create the validation sets, the 70/30 rule is used. To ensure that data is being sampled uniformly, a sanity check confirms the uniform distribution of frauds in the training dataset in Appendix Fig. 14. This mitigates the risk of sampling bias. Hence, the training set is split into 70% for training purposes, while the remaining 30% is split into 15 different validation sets to test the hyperparameter in the range of [8,000 - 15,000] by steps of 500. Instances are selected randomly without replacement from the original training dataset to form the new training set and validation sets.

Thus, the solution to the two identified problems above is outlined as follows: the search of the right hyperparameter is done through trial and error by looking at how model performance changes between the 15 runs of testing (i.e. each of the 15 validation tests is tested once to measure performance). At the same time, the choice of the optimal depth is done by minimizing the generalization gap in each validation set. The difference between validation performance and training performance will tell whether or not the model is overfitting. In other words, the generalization gap is the amount of performance that does not generalize to the training data.

Figure 3: Generalization gap between training (70%) & validation set (k=9,500) in decision tree

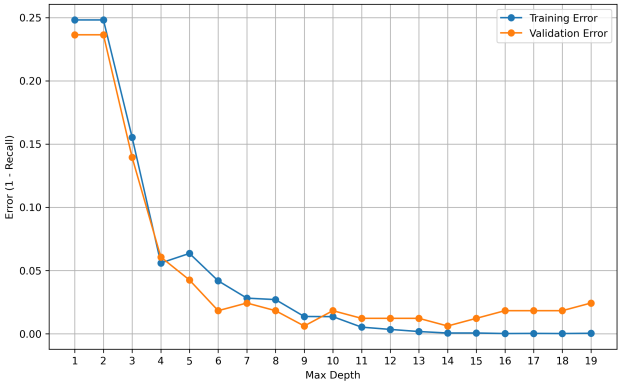
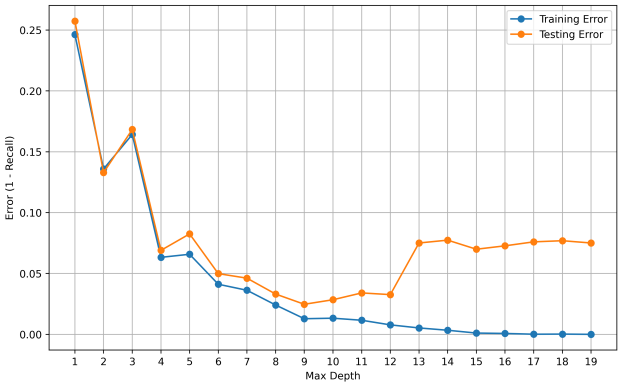


Figure 4: Generalization gap between training set (100%) & testing set in decision tree



In Fig. 3, the optimal tree depth that minimizes the generalization gap is 9, with an error of 0.01 (defined as 1-recall). For this validation set, the hyperparameter is equal to 9,500 samples for the minority *fraud* class and 9,500 for the *non-fraud* class. The choice of this hyperparameter and depth of the decision tree is the optimal one. It maximizes performance and reduces the generalization gap, if compared with the results of the other 14 validation sets in the Appendix Table 11.

With balanced data and the best decision on tree pruning available, the ID3 algorithm is executed with a final run on the original training dataset (i.e., the training dataset before the 70/30 split). Performance is measured on the test dataset. In Fig. 4, the generalization gap is minimized with a tree depth equal to 9, where the error is equal to 0.25 (with the error defined as 1-recall). This fact alone confirms the good guess given with validation sets on the optimal tree depth. In Fig. 3 and Fig. 4, the Error is defined as (1-Recall) and it is used to measure the generalization gap. All metrics for assessing and comparing model performance are discussed in Section 3.8.

3.4 Random forest

Random forest is a combination of tree predictors, where each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest (7). Each tree in the model is trained on a randomly sampled subset of data using a technique known as *bootstrap training* or *bagging* (7), which reduces variance and mitigates the risk of overfitting. During prediction, each tree independently classifies the input, and the final classification is determined by majority vote, according to Breiman (7).

Given the high class imbalance in the dataset, standard training would lead to a bias toward the majority class. To address the issue, *class weighting* is applied, increasing the penalty for misclassifying fraud cases to make the model more sensitive to the prediction of the minority class. Various hyperparameters are tuned to optimize the trade-off between false negatives and false positives, most notably the number of trees, maximum tree depth and the minimum samples per split. Despite increasing performance, the application of boosting methods is not used. In fact, if boost is applied in this model, it should also be applied to the baseline, a circumstance that is not applied by Afriyie *et al.* (5) in the training of their random forest used as baseline.

3.5 Support vector machine

The support vector machine is a supervised learning method that uses a "hard" classification: an instance is classified as either positive or negative. The idea behind is to plot two support vectors (i.e. the margins) and then to fit a hyperplane that maximizes the distance between the two support vectors. SVM can handle high-dimensional data, but requires some tuning, with the most important hyperparameters being the type of kernel and the value of C. In this study, a linear kernel is chosen, as it is computationally efficient for higher-dimensional data. The other hyperparameter, C, is the trade-off between smaller margins with strict categorization and larger margins with a softer categorization. A low value C of 0.3 creates a larger margin that is more generalizable, yet prone to misclassification. Conversely, a larger value of 10 uses very strict margins and has a better categorization rate at the expense of overfitting. In the modeling, a lower value of C is chosen as the optimal value of C. The choice is made by using the halving grid search with a factor of two, which halves the possible values for the next iteration, and three folds: two for the training set and one for the test set.

3.6 Logistics regression

Logistic regression is a statistical method for binary classification problems that estimates the probability that a given input belongs to one of the 2 classes. This is done by applying the logistic sigmoid function in Eq. 4 to the weighted sum of the input features. The sigmoid function outputs a value between 0 and 1 that determines the class label. The weighted sum is represented by z in Eq. 5, where x is the input feature and W and b are the learned parameters.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4}$$

$$z = W_1 x_1 + W_2 x_2 + \cdots + W_n x_n + b \tag{5}$$

To address imbalance, the majority class is undersampled and SMOTE is used to increase the representativeness of the minority class *fraud*. Grid search is also used in logistic regression for hyperparameter tuning: the regularization strength value is optimized over a predefined range of values in the interval [0.001, 100]. The other optimization is the decision threshold. A decision boundary of 0.5 means that a transaction having a 50% probability of being fraudulent is classified as such by the classifier. In this study, the threshold is set to a lower value of 0.24 to achieve better performance. This means that the model will classify transactions as fraudulent even if their probability of being a fraud is only 24%.

3.7 Neural network

The neural network trained in this study consists of multiple layers of neurons that progressively refine the input data with hidden layers. In detail, the Feedforward neural network or multilayer perceptrons (MLP) consists of seven layers, five of which are hidden. The first layer (i.e. the input layer) collects the raw data and passes it to the hidden layer without computations. The five hidden layers contain 2^n neurons (512, 256, 128, 64 and 32, respectively) to optimize computational efficiency and increase the performance. Similarly to a descending pyramid, the MLP leaves out irrelevant features early on: it progressively filters out features until the most abstract are left over. This behavior enforces the model to learn complex relationships. At the same time, it prevents overfitting by eliminating irrelevant features, as Srivastava *et al.* (14) remarked.

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0 \end{cases} \tag{6}$$

Each of the five hidden layers uses an activation function, which is described by the Eq. 6. In this equation, α is the negative slope, which is set to 0.05. The equation says that positive values in the datasets remain unchanged, while negative values are multiplied by the negative slope. This operation brings the value close to zero, as opposed to the standard ReLU, which brings the value to exactly zero. In other words, by setting the negative values close to and not to zero, neurons in the network do not die. This means that more neurons are available for training, which makes the model more efficient since it can learn from the extra neurons. In fact, such neurons ultimately stimulate the flow of gradients during backpropagation.

The final layer of the model uses the similar sigmoid function used in the logistic regression described by Eq. 4. The function computes a value between 0 and 1 and can be interpreted as a probability score: the higher the probability, the higher the likelihood that the transaction is flagged as fraudulent. Thus, a lower value of the function suggests a legitimate transaction. The input z of the sigmoid function is described by Eq. 7, where W_i are the weights of connections with the previous hidden layers, a_i is the activation (i.e. output) of the previous layer, and b is the bias term.

$$z = W_1 a_1 + W_2 a_2 + \cdots + W_n a_n + b \tag{7}$$

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{8}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{9}$$

Optimization and parameters tuning are required to improve the performance of the neural network. The optimization algorithm used in this model is the adaptive moment estimate (Adam). To stabilize data fluctuations and prevent overfitting, two moving averages are used in neural network training: the first movement estimate m_t in Eq. 8 and the second movement estimate v_t, in Eq. 9. The former is responsible for the momentum and direction of the gradient. The latter captures the magnitude of the gradients, which means that it can scale the learning rate of each parameter. For both equations, β_i represents the decay rate and g_t the gradient at the time stamp t.

3.8 Assess the performance

To assess the performance of the models, the recall, precision, accuracy and F1-score are calculated, starting from the confusion matrix of the model represented in Table 1.

Table 1: Confusion matrix

	Actual value	
Predicted value	Fraud	Non Fraud
Fraud	True Positive	False Positive
Non Fraud	False Negative	True Negative

The accuracy in Eq. 10 measures the number of correct predictions on the total number of predictions.

$$\textit{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{10}$$

Precision in Eq. 11 measures the number of correctly classified *frauds* on the total number of transactions predicted to be frauds.

$$\textit{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{11}$$

Recall in Eq. 12 measures the number of correctly classified *frauds* on the actual number of frauds.

$$\textit{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{12}$$

F1-Score in Eq. 13 measures the weighted mean of precision and recall. Hence, it provides a single number to express such trade-offs. Low values of F1 suggest a situation of imbalance, where one metric cannot increase without reducing the other.

$$\textit{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

In addition to these metrics, the value of the AUC (i.e. the area under the ROC curve) is calculated to determine the probability that the classifier makes a ranking error or to discriminate between *frauds* and *non frauds*. The ROC plots the False Positive rate on the x-axis and the True Positive rate along the y-axis. The model performs better if the ROC is closer to the top left corner and the AUC value is close to 1. However, this metric loses value in situations of class imbalance. This is what happens with fraud detection.

The problem of imbalance makes not only the AUC lose its value, but also the metric of accuracy, and precision to some extent. In Fig. 1, the median cost of a fraud is higher than the median cost of a legitimate transaction. This means that recall is more important than precision, since the bank does not want to miss a costly fraud during detection, even if this means reviewing more false positive cases of frauds. Thus, the model should be tuned to increase recall, as a trade-off to lower precision. This is exactly what is done through the choice of validation sets and tree depth in the decision tree. Recall is crucial in credit card fraud detection as it ensures that no fraudulent transaction goes unnoticed. For the bank, this translates into a problem of cost minimization arising from fraudulent transactions to minimize losses, even if it means flagging some legitimate transactions mistakenly.

Yet, precision plays a role even in the situation of class imbalance. Recall is more important than precision, but the false positive should not be ignored. In fact, plotting the PR curve (i.e., precision-recall curve) and analyzing the F1-score allow one to understand how much the recall can be improved. Further, plotting the Coverage Curve (FP vs. TP) helps to understand the trade-off between capturing more true positives and having more false positives.

Arguably, banks may translate FN and FP in misclassification cost. There is no free lunch: frauds cost money, but also the time taken by the bank staff to review transactions is not free. This problem translates to the question: is it better for the bank to save money on inspections or reduce as much as possible the frauds? Prof. Bloem (15) suggests that a bank may decide that missing one fraud is as costly as inspecting 500 frauds. Thus, the performance can be judged with a target of 1 false negative every 500 false positive. Is this the case? Looking at Fig. 1, the median value of fraud is \$400. This can be interpreted as the cost of missing one fraud. In this study, the median is considered instead of the average because it is less affected by outliers and skewed data than the mean. In fact, in Fig. 1 data for frauds is skewed, while data for non-frauds contains outliers.

From data on salaries available on Glassdoor (16), \$19 is the hourly rate of a KYC analyst (i.e. Know Your Customer, the profession that examines customer files). The only data missing is the number of transactions reviewed per hour. Normally, when the system detects a possible fraud, the transaction is blocked. The analyst would then call the credit card holder by phone to ask security questions to verify that the credentials on the credit card match the answer of the person answering the phone call. A related study of Melin and Sandellon (17) in a speech-controlled telephone banking system shows that, among 21 successful enrollment calls, participants took at most 5:21 minutes for the identification process. However, the person called may not answer the phone for different reasons, including the person called being the fraudster. Thus, the average time taken to review a fraudulent credit card transaction is the average time taken by the bank operator to verify the credentials of the respondent in the interval [0-5:21] minutes, which is 2:40 minutes. This is equivalent to saying that in one hour, a KYC analyst can review 22.5 transactions flagged as fraudulent by the system. Hence, the approximate cost of reviewing a possible fraudulent transaction is the hourly cost of the KYC employee divided by the number of transactions reviewed in an hour, which gives \$0.84. This means that missing a fraud, whose median cost is \$400, is as costly as inspecting 476 transactions, with a single review costing as little as \$0.84. Thus, the suggested ratio of Prof. Bloem (15) of 1 False Negative to every 500 False Positive sounds reasonable and is used as metrics to evaluate the performance of models along with recall. The lower the recall, the better, and the closer the model can get to the ratio FN: FP of 1:500, the better.

Interestingly, there exists another perspective to measure the model performance: the total cost of imbalance or misclassifications. The cost of missing out one fraud is higher than the cost of reviewing a transaction, but both contribute to the cost imbalance, although in different measures. It is possible to use this newly perspective on the cost of misclassifications to judge the performance of the models. In detail, the cost of misclassification is illustrated in Equation 14 and can be calculated as the median price of frauds, \$400 in Fig.1, times the number of frauds that go undetected plus the cost of reviewing a suspicious transaction, \$0.84, times the number of legitimate transactions that the model predicts as fraudulent.

$$\text{C} = (C_{\text{fraud}} \times FN) + (C_{\text{review}} \times FP) \tag{14}$$

Measuring the cost of misclassifications goes beyond the purpose of quantifying the cost of frauds for the bank. It allows to compare more objectively the performance of the classifiers. If recall

and FN:FP ratio were used as the only indicators to assess performance, then they would stop to be reliable metrics of fraud predictions. The heart of the problem lies in Goodhart's law, which states that any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes. The use of a third metric to measure the performance sounds appealing then, given Goodhart's law. In other words, the recall and the FN: FP ratio of 1:500 are used to optimize and train the classifier, while the total cost illustrated in Equation 14 is used to measure the effectiveness of the classifier.

Finally, it is not only about the cost, but also about interpretability and transparency. Some banks may decide to let some frauds go undetected to save budgeting hours of employees. Other institutions may decide that frauds are indeed very risky, not only financially, but also in terms of reputations and corporate image. Promising clients zero tolerance towards frauds may pay off in terms of goodwill, which is far more valuable than payroll costs for some multinational financial corporations. For this reason, in this report the other metrics of F1 score and precision are presented to give a holistic view.

4 Results

The decision tree resulting from the testing dataset can be seen in Appendix Fig. 16. The data in Table 2 and Appendix Fig. 15 illustrate the main features used as nodes in the decision tree that discriminate between *frauds* and *non-frauds*, order by *Importance*. Such metric measures the normalized total reduction in impurity as defined in Eq. 1 that the feature contributes across the entire decision tree classifier.

Table 2: Top 10 features for fraud detection

Feature	**Importance**
Amount	43 %
City population	8 %
CC number	7 %
Age	7 %
Transaction hour	7 %
Category misc pos	5 %
Merchant	4 %
Category food&dining	4 %
Category travel	4 %
Category shopping net	2 %

As expected, the most important features are the amount and hour of transaction, as well as the age of the credit card holder, in line with the data preprocessing in Section 3.1. Surprisingly, and in contrast with the data exploration, the geographical location does not play an important role in predicting frauds. Yet, the city population plays an important role in predicting *frauds*, as well as the number of the credit card. This hints at the fact that fraudsters may follow a certain pattern, targeting individuals in larger cities and reusing their credit card to complete the fraud. Financial institutions in these cities should then raise their level of warning, especially if consumers have different cards on their name. In addition, frauds are more likely in the categories of travel, food and dining, and miscellaneous. Hence, financial entities should pay particular attention to transactions that follow under unclassified purchases, namely subscriptions or gift purchases.

Table 3: Confusion matrix decision tree

	Actual	
Predicted	Fraud	Non Fraud
Fraud	2,092	27,374
Non Fraud	53	526,200

Table 3 illustrates the confusion matrix for the decision tree. The model was able to correctly identify 97.5% cases of frauds in a highly imbalanced testing dataset where only 0.4% of the 555,719 transactions are labeled as fraudulent. This result is higher than the percentage of 93.0% of the baseline in Table 8, which makes the decision tree a good classifier.

Figure 5: ROC curve decision tree

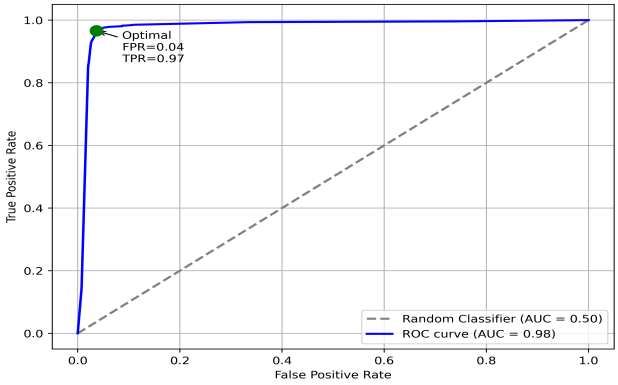


Figure 6: Coverage curve decision tree

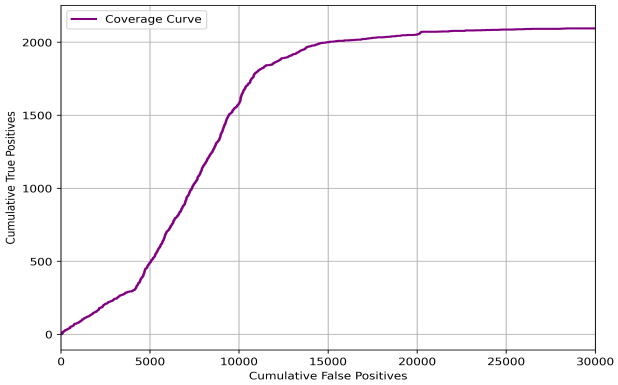
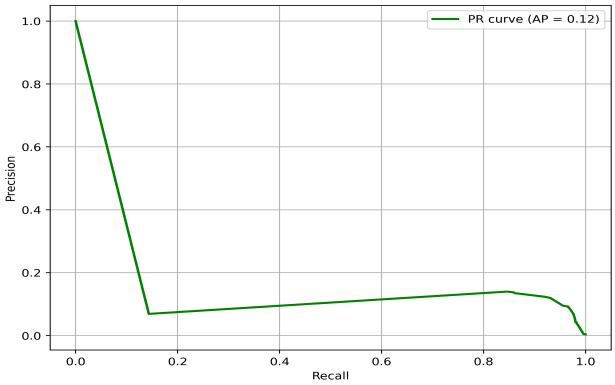


Fig. 5 shows the ROC and AUC value for the decision tree classifier. The AUC value of 97.9% sounds impressing if compared with the value of 94.5% of the baseline in Table 8. However, since the classes are imbalanced, the ROC-AUC metric loses its value. It is possible to tell that the classes

are unbalanced by looking at the coverage graph in Fig. 6: it is not square. The x-axis is stretched out compared to the y-axis, which suggests a high number of false positives for each true positive case of fraud.

Indeed, in a class imbalance situation, there is a trade-off between optimizing recall or precision. This trade-off can be clearly seen in the PR curve depicted in Fig. 7: the precision decreases rapidly if the recall increases to approximately 18%. Then, it grows only slightly, while the recall increases to 95%, when precision falls significantly. The F1-score also captures the trade-off well, with a value of 0.13 for the decision tree. In general, the F1-score in Table 8 is low across all models in this research problem, suggesting that recall cannot be improved at the expense of precision. This finding is expected: due to the high cost of frauds, models are optimized for recall, as explained in Section 3.8.

Figure 7: Precision-Recall curve decision tree



The following tables illustrate the results of the other models. Table 4 shows the confusion matrix for random forest, while Table 5 presents the result for SVM. The confusion matrices for logistic regression and neural network are presented in Table 6 and Table 7, respectively.

Table 4: Confusion matrix random forest

	Actual	
Predicted	Fraud	Non Fraud
Fraud	1,852	7,120
Non Fraud	293	546,454

Table 5: Confusion matrix SVM

	Actual	
Predicted	Fraud	Non Fraud
Fraud	1,434	21,560
Non Fraud	711	532,014

Table 6: Confusion matrix logistics regression

	Actual	
Predicted	Fraud	Non Fraud
Fraud	1,734	157,974
Non Fraud	411	395,600

Table 7: Confusion matrix neural network

	Actual	
Predicted	Fraud	Non Fraud
Fraud	2,092	43,090
Non Fraud	53	510,484

Table 8 provides a summary of the performance of the models. Among all, the decision tree and neural network perform the best in terms of recall. In addition, the decision tree has a closer ratio FN: FP to the target 1:500 defined in Section 3.8. If the findings are compared with the baseline in Table 8), the performance of the decision tree beats the performance of the random forest, as opposed to the conclusion drawn by the authors in their paper.

Indeed, Afriyie *et al.* (5) suggest that the random forest performs best in fraud detection. However, the data in Table 4 and Table 8 give a different conclusion that can be explained by the following challenges in this study. Adjusting the class weights makes the model more sensitive to fraud cases and increases the recall. However, the number of false positives also increases, leading to a situation of trade-off between precision and recall identified with the decision tree. The main challenge faced during the application of random forest is the long training time. During training, steps are taken to tune the hyperparameter, namely tree depth, minimum samples, and feature selection. However, achieving better performance translates into extremely long training times. This suggests that the use of alternative approaches, namely boosting with XGBoost, may improve the model performance and make training easier. However, this circumstance is not explored, as boosting is not used in the baseline, as noted in Section 3.4

Logistic regression and SVM perform poorly as classifiers, compared to the decision tree and neural network. This can be explained with the fact that the dataset is high-dimensional and is not linearly separable. The poor performance of the logistic classifier can also be explained by the poor choice of the decision boundary, which prevents the classifier from capturing complex patterns. In fact, frauds can be explained by more than one feature, as identified in Table 2 and Appendix Fig. 15. However, logistic regression treats these features independently, which prevents the model from capturing the complexity of the fraud. This finding is also in line with the baseline: Afriyie *et al.* (5) consider logistic regression the worst classifier.

Nevertheless, to judge models performance, the total cost of misclassifications as defined in Equation 14 should be considered, to avoid the pitfall stated with Goodhart's law in Section 3.8. Undoubtedly, the model that minimizes the total cost of misclassifications is the decision tree, with a total cost as low as $ 44,194. The higher cost of the other models is explained in terms of false negatives that the classifiers are not able to identify. In fact, SVM, logistics regression and Random Forest miss 711, 411 and 293 cases of fraud (i.e. false negative), which translates into a loss of $ 284,400, $ 164,400 and $ 117,200 respectively, considered the median cost of fraud of $ 400 in Fig. 1. The same number of frauds go undetected with the neural network and decision tree, 53 for both models. However, the former has a lower precision compared to the latter. This translates into a higher cost for the neural network of $ 13,201. In other words, a higher number of transactions are erroneously classified as frauds instead of legit transactions with the neural network compared to the decision tree.

Table 8: Comparison of models performance.

Model Name	Accuracy	F1-Score	Recall	Precision	AUC	FN:FP	Cost
Decision Tree	0.95	0.13	0.98	0.07	0.98	1:516	$ 44,194
Random Forest	0.99	0.33	0.86	0.21	0.98	1:24	$ 123,181
SVM	0.96	0.11	0.67	0.06	0.85	1:30	$ 302,510
Logistic Regression	0.71	0.02	0.81	0.01	0.87	1:384	$ 297,098
Neural Networks	0.92	0.09	0.98	0.05	0.99	1:813	$ 57,396
Baseline (5)							
Decision Tree	0.92	0.09	0.93	0.05	0.95	1:270	n.a.
Random Forest	0.96	0.17	0.97	0.09	0.99	1:225	n.a.
Logistic Regression	0.92	0.08	0.76	0.04	0.87	1:76	n.a.

5 Conclusion

In this research problem, the decision tree is the classifier that has the best performance, as it allows the bank to reduce the cost of misclassifications, with the neural network being the second best model. Important benefits of the decision tree are that it allows one to easily inspect what the system is learning, what features are being used to discriminate between the *fraud* and *non-fraud* classes (Table 2) and to pick up on the fact that certain patterns are coming out. For instance, what does it tell that fraudulent transactions are concentrated in large cities, at night and in the categories travel, food&dining and miscellaneous? This allows the fact that domain experts can be consulted. A financial advisor can look at the features used and tell if their use as a discriminator is wrong. This in turn helps to see if there is a fundamental flaw in the classifier design. However, the total cost of misclassifications may increase as a result of this expert review.

An area of improvement with the decision tree is the search for the right size of the minority and majority class samples to train the classifier. In the modeling, the validation sets are created by splitting the training set with the 70/30 rule and then creating 15 equal-size validation sets. The trial-and-error approach is used to test the hyperparameter for convenience. This is because the preliminary data exploration gives a rough idea on how data is distributed and what features are important for fraud classification. However, if nothing is known about the distribution of data and its characteristic, one could use gradient descent to minimize the cost of misclassifications in Equation 14. Choosing this approach also means that a new metric to measure and compare the performance must be assessed to avoid the pitfall of Goodhart's law.

Despite the cost difference of $ 13,201 between the decision tree and neural network, it is possible to beat the performance of the former with a tuning of the latter. For instance, changing the number of neurons used per layer and the number of hidden layers may increase the performance of the neural network, with the downside being the long training time. In addition, trying different optimizers, namely stochastic gradient descent, and experimenting with the learning function might help to identify a better configuration than the one discussed in this study. Changing the loss function with one that uses weighted binary cross-entropy and assigning higher weights to the minority class may also yield better results. This would make the classifier more cautious in classifying fraudulent transactions. It also represents an improvement over the regular loss function that uses binary cross-entropy, which would treat every misclassification as equal and would eventually lead to biases in the modeling. Thus, the improvements to the neural network may give this classifier an upper hand in fraud prediction compared to the less flexible decision tree classifier.

References

[1] European Banking Authority. **2024 Report on Payment Fraud**. 2024. 1

[2] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J. Christopher Westland. **Data mining for credit card fraud: A comparative study**. *Decision Support Systems*, **50**(3):602–613, 2011. On quantitative methods for detection of financial fraud. 1

[3] Lisha Yu, Wai Man Chan, Yang Zhao, and Kwok-Leung Tsui. **Personalized Health Monitoring System of Elderly Wellness at the Community Level in Hong Kong**. *IEEE Access*, **6**:35558–35567, 2018. 1

[4] Suraj Patil, Varsha Nemade, and Piyush Kumar Soni. **Predictive Modelling For Credit Card Fraud Detection Using Data Analytics**. *Procedia Computer Science*, **132**:385–395, 2018. International Conference on Computational Intelligence and Data Science. 1

[5] Jonathan Kwaku Afriyie, Kassim Tawiah, Wilhemina Adoma Pels, Sandra Addai-Henne, Harriet Achiaa Dwamena, Emmanuel Odame Owiredu, Samuel Amening Ayeh, and John Eshun. **A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions**. *Decision Analytics Journal*, **6**:100163, 2023. 1, 3, 4, 8, 9

[6] Sushmito Ghosh and Douglas L. Reilly. **Credit card fraud detection with a neural-network**. *1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences*, **3**:621–630, 1994. 1

[7] Leo Breiman. **Random forests**. *Machine Learning*, **45**(1):5–32, 2001. 1, 4

[8] Chengwei Liu, Yixiang Chan, Syed Hasnain Alam Kazmi, and Hao Fu. **Financial Fraud Detection Model: Based on Random Forest**. *International Journal of Economics and Finance*, **7**(7):178–188, 2015. 1

[9] Xueming Li, Xiaoqing Wang, Qingqing Liu, Zheng Li, and Zhongyi Li. **Credit Card Fraud Detection using Logistic Regression**. *International Journal of Advanced Computer Science and Applications (IJACSA)*, **13**(4):706–712, 2022. 1

[10] Andrea Dal Pozzolo, Olivier Caelen, Raquel A. Johnson, and Gianluca Bontempi. **Calibrating Probability with Undersampling for Unbalanced Classification**. In *Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2015*, **9284** of *Lecture Notes in Computer Science*, pages 159–175. Springer, 2015. 1

[11] Brandon Harris Kartik Shenoy. **Credit Card Transactions Fraud Detection Dataset**. 2020. 2

[12] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Anuj Karpatne, and Vipin Kumar. **Introduction to Data Mining - 4.3.7 Characteristics of Decision Tree Induction**. pages 168–172, 2014. 3

[13] Tom Mitchell. **Machine Learning - Chapter 3: Decision Tree learning**. pages 52–71, 1997. 3

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. **Dropout: A Simple Way to Prevent Neural Networks from Overfitting**. *Journal of Machine Learning Research*, **15**:1929–1958, 2014. 5

[15] Peter Bloem. **Model evaluation - Part 3; Evaluation metrics**. *Machine Learning VU Amsterdam*, 2025. 6

[16] Glassdoor. **Salaries by position at ABN AMRO**, 03 2025. 6

[17] Hkan Melin and Anna Sandell. **CTT-bank: A speech controlled telephone banking system - an initial evaluation**. 07 2001. 6

List of Figures

1 Fraud/Non-fraud amount . . . 2
2 Correlation of features . . . 2
3 Generalization gap between training (70%) & validation set (k=9,500) in decision tree . . . 4
4 Generalization gap between training set (100%) & testing set in decision tree . . . 4
5 ROC curve decision tree . . . 7
6 Coverage curve decision tree . . . 7
7 Precision-Recall curve decision tree . . . 8
8 Class imbalance: Fraud vs Non-fraud transactions . . . 11
9 Frauds by gender . . . 11
10 Frauds by age . . . 11
11 Frauds by hour . . . 11
12 Frauds by merchant location . . . 12
13 Frauds by category . . . 12
14 Uniform distribution of fraud transactions in the training dataset . . . 12
15 Relevant features for fraud detection . . . 12
16 Decision tree classifier on testing dataset . . . 12

Figure 8: Class imbalance: Fraud vs Non-fraud transactions

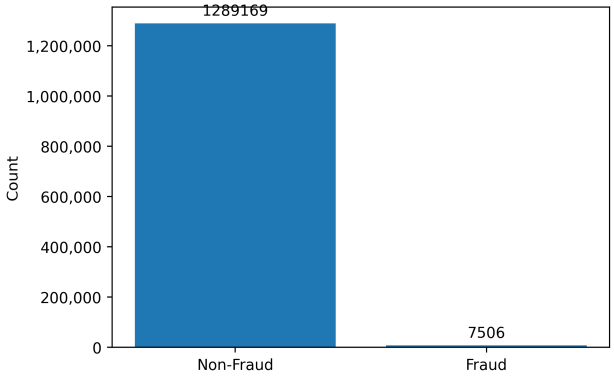


Figure 9: Frauds by gender

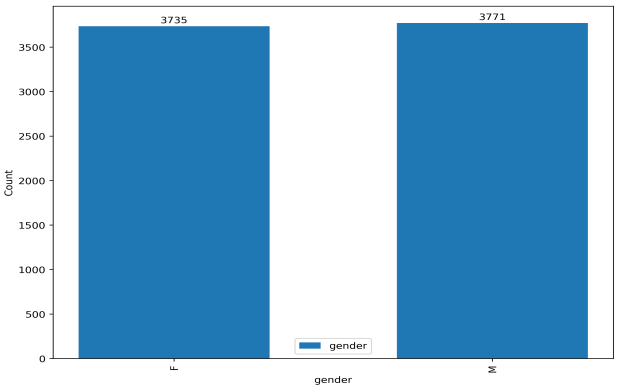


Figure 10: Frauds by age

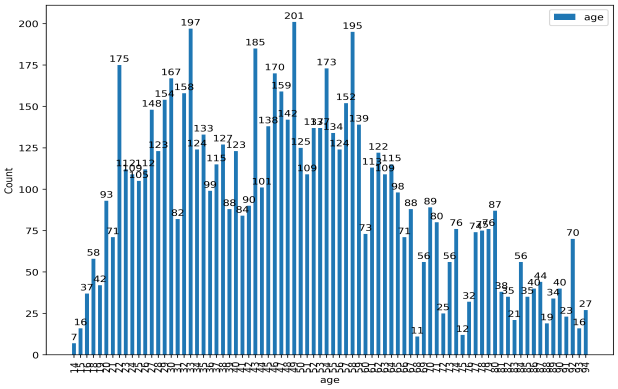


Figure 11: Frauds by hour

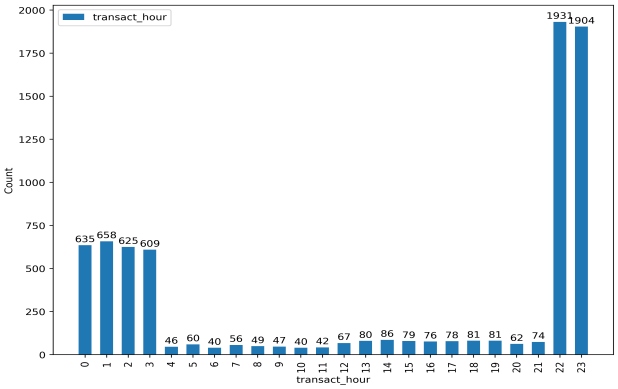


Figure 12: Frauds by merchant location

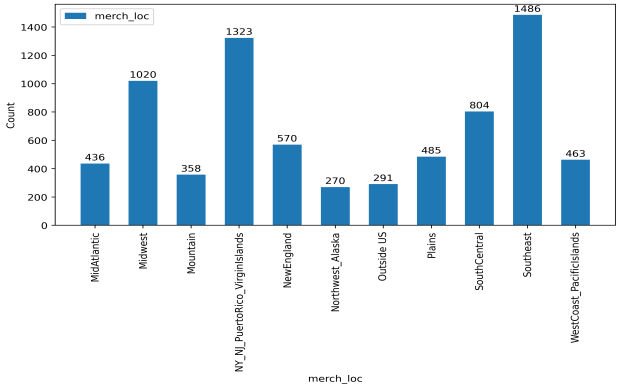


Figure 13: Frauds by category

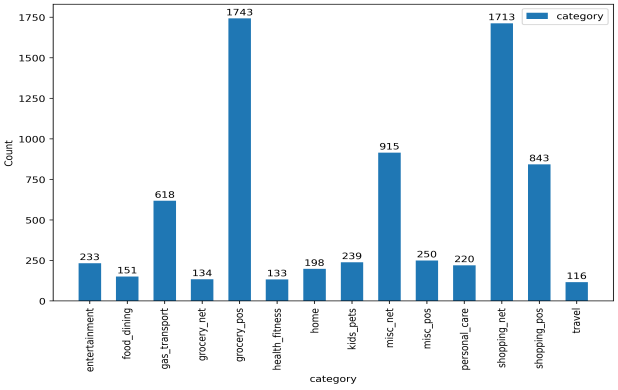


Figure 14: Uniform distribution of fraud transactions in the training dataset

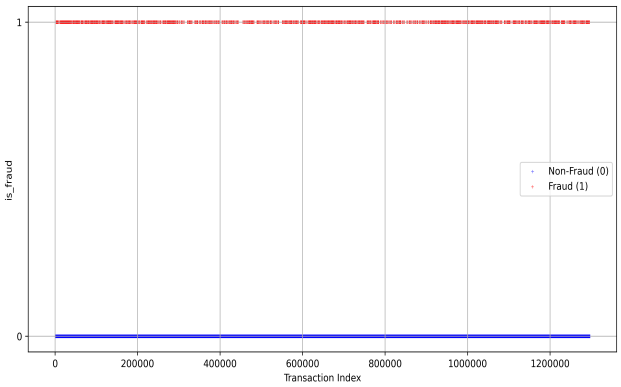


Figure 15: Relevant features for fraud detection

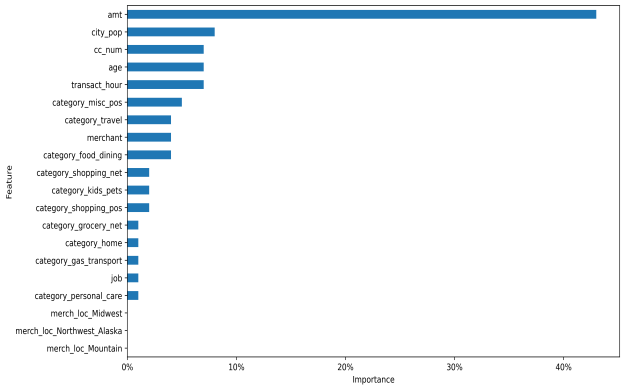
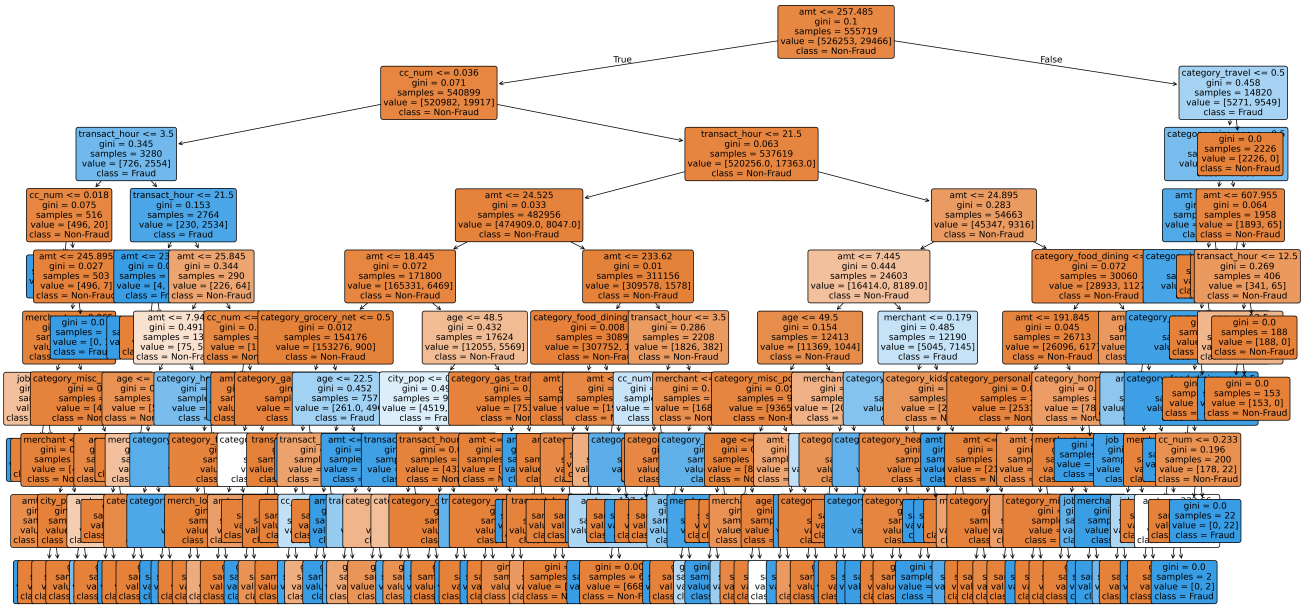


Figure 16: Decision tree classifier on testing dataset



List of Tables

1 Confusion matrix . . . 5
2 Top 10 features for fraud detection . . . 7
3 Confusion matrix decision tree . . . 7
4 Confusion matrix random forest . . . 8
5 Confusion matrix SVM . . . 8
6 Confusion matrix logistics regression . . . 8
7 Confusion matrix neural network . . . 8
8 Comparison of models performance. . . . 9
9 Initial Features . . . 13
10 Features after Data Preprocessing . . . 13
11 Performance of validation sets (30% of training set) for different values of hyperparameter k (size of the minority and majority class) in decision tree. . . . 14

Table 9: Initial Features

Feature
Unnamed: 0
trans_date_trans_time
cc_num
merchant
category
amt
first
last
gender
street
city
state
zip
lat
long
city_pop
job
dob
trans_num
unix_time
merch_lat
merch_long
is_fraud

Table 10: Features after Data Preprocessing

Feature
cc_num
merchant
amt
gender
city_pop
job
transact_hour
transact_mth
age
category_food_dining
category_gas_transport
category_grocery_net
category_grocery_pos
category_health_fitness
category_home
category_kids_pets
category_misc_net
category_misc_pos
category_personal_care
category_shopping_net
category_shopping_pos
category_travel
cust_loc_Midwest
cust_loc_Mountain
cust_loc_NY_NJ_PuertoRico_VirginIslands
cust_loc_NewEngland
cust_loc_Northwest_Alaska
cust_loc_Outside US
cust_loc_Plains
cust_loc_SouthCentral
cust_loc_Southeast
cust_loc_WestCoast_PacificIslands
merch_loc_Midwest
merch_loc_Mountain
merch_loc_NY_NJ_PuertoRico_VirginIslands
merch_loc_NewEngland
merch_loc_Northwest_Alaska
merch_loc_Outside US
merch_loc_Plains
merch_loc_SouthCentral
merch_loc_Southeast
merch_loc_WestCoast_PacificIslands
is_fraud

Table 11: Performance of validation sets (30% of training set) for different values of hyperparameter k (size of the minority and majority class) in decision tree.

Validation Set	K	Instances	FN:FP	Recall	AUC	Tree Depth
1	8,000	25,934	1:166.40	0.9671	0.9855	9
2	8,500	25,934	1:448.50	0.9868	0.9867	8
3	9,000	25,934	1:327.67	0.9811	0.9798	10
4	9,500	25,934	1:987.00	0.9939	0.9911	9
5	10,000	25,934	1:214.00	0.9726	0.9876	9
6	10,500	25,934	1:119.17	0.9545	0.9722	8
7	11,000	25,934	1:193.00	0.9686	0.9845	7
8	11,500	25,934	1:128.86	0.9548	0.9834	7
9	12,000	25,933	1:364.00	0.9859	0.9844	8
10	12,500	25,933	1:304.00	0.9811	0.9836	7
11	13,000	25,933	1:319.00	0.9792	0.9897	7
12	13,500	25,933	1:127.25	0.9437	0.9821	7
13	14,000	25,933	1:322.67	0.9808	0.9839	8
14	14,500	25,933	1:112.38	0.9503	0.9777	7
15	15,000	25,933	1:314.67	0.9815	0.9854	7
Tot. val. set		**389,003**				
70% training set		907,672				
100% training set		1,296,675				