Università degli studi di Milano-Bicocca

Advanced Machine Learning

Final Project

Fraud Detection in Credit Card Transactions

Authors:
Samuele Crispino - 856339 - s.crispino2@campus.unimib.it

February 7, 2025

Abstract

This project focused on anomaly detection in credit card transaction fraud. A publicly available Kaggle dataset was used, and the primary objective was to explore different approaches to achieve the best performance according to a meaningful set of metrics. Additionally, the model performance scores were enriched with fairness metrics to monitor potential discriminatory biases, particularly those related to user gender.

1 Introduction

Financial fraud detection was crucial in banking and e-commerce, as fraudulent transactions could result in significant losses and security risks. The challenge was exacerbated by the imbalanced nature of fraud datasets, where fraudulent transactions were rare, and the constantly evolving tactics of fraudsters.

The study evaluated four deep learning approaches: a standard Feedforward Neural Network (FFN), an FFN with Focal Loss to address class imbalance, an Autoencoder detecting anomalies via reconstruction error, and a Long Short-Term Memory (LSTM) network analyzing transaction sequences. Each approach provided a unique perspective on fraud detection, balancing predictive performance and adaptability.

In addition to traditional performance scores, fairness metrics were incorporated to assess potential biases in the models, which was critical to avoid discrimination and ensure equitable financial access. The focus on fairness aligned with AI safety best practices and has been inspired by the Article 10 of EU's Artificial Intelligence Act [1].

2 Datasets

For the purpose of this project, the focus was on a fraud detection dataset containing transaction records with a range of attributes that described both the transactions and the individuals involved. The main dataset used in this study included 400000 instances with the following attributes:

`transDateTransTime` (Timestamp of the transaction), `ccNum` (Credit card number used for the transaction), `merchant` (Name of the merchant), `category` (Category of goods or services), `amt` (Amount spent in the transaction), `first`, `last` (Cardholder's first and last names), `gender` (Cardholder's gender), `street`, `city`, `state`, `zip` (Cardholder's address details), `lat`, `long` (Cardholder's address coordinates), `cityPop` (City population), `job` (Cardholder's occupation), `dob` (Cardholder's date of birth), `transNum` (Unique transaction identifier), `unixTime` (Unix timestamp of the transaction), `merchLat`, `merchLong` (Merchant's location coordinates), `isFraud` (Fraud indicator, 1 for fraud, 0 for not)

As shown in Figure Figure 1, dataset was highly imbalanced, with fraudulent transactions constituting only 0.5% of the total records.

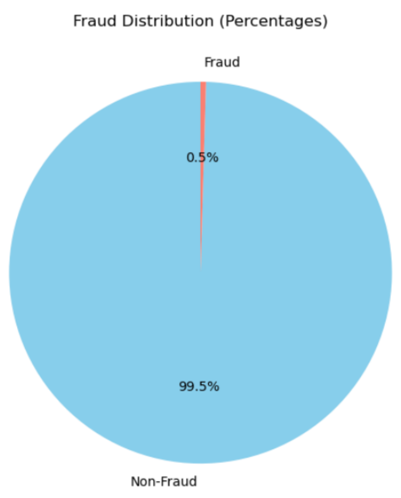


Figure 1: Fraud Distribution Percentages.

An analysis of the dataset revealed that the individual columns, without further manipulation, exhibit poor correlation with the target variable `is_fraud`. Among the features, only `amt`, which represents the monetary amount of the transaction, showed a slight positive correlation of 0.22 with the target variable.

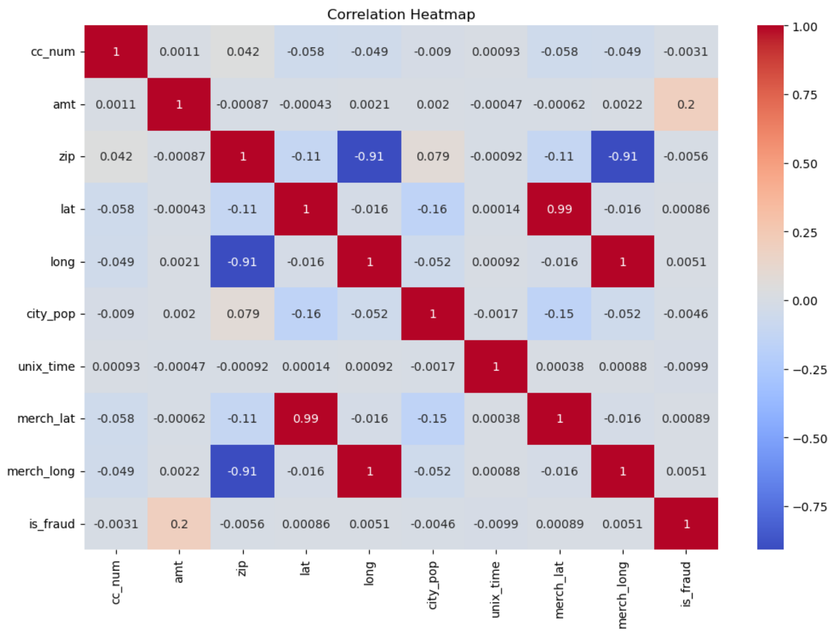


Figure 2: Original Features Correlation

This suggested that additional feature engineering or enrichment processes may be required to improve the predictive power of the dataset.

In order to create a second version of the dataset enriched with useful features, an analysis of the correlation between the target column and specific feature combinations was performed. This analysis focused exclusively on the training data to prevent the incorporation of insights from the test data, which had to remain "unseen."

The analysis began with investigating the fraud frequency by hour, grouping the data by the hour of the day. This showed that the last hours of the night and the first hour of the morning were more likely to be associated with fraudulent transactions than other times of the day Figure 3. On the other hand, the fraud frequency across the days of the week did not exhibit significant variations (Figure 4).

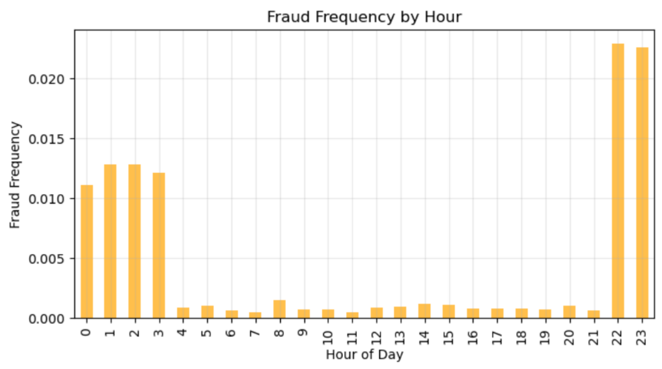


Figure 3: Fraud Frequency by Hour

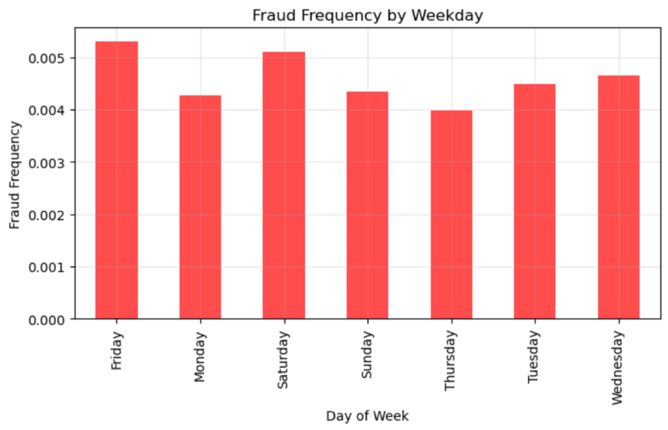


Figure 4: Fraud Frequency by Day of Week

Additional insights were gained by computing the fraud frequency per merchant and per transaction category. It was found that certain merchants and transaction categories were more likely to be associated with fraudulent transactions Figures 5 and 6.

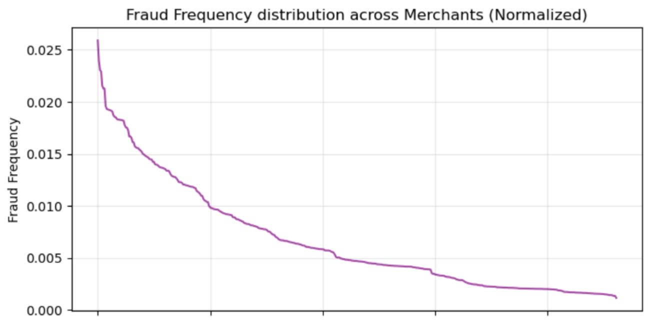


Figure 5: Fraud Frequency distribution across merchants: the X-axis represents the merchant index.

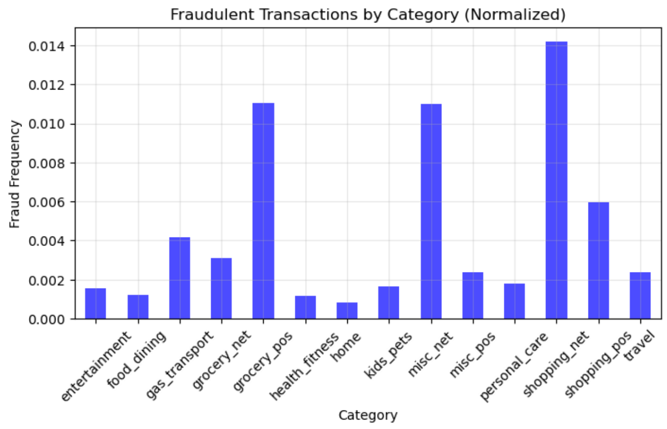


Figure 6: Fraud Frequency distribution across categories.

An further analysis was conducted to examine the fraud probability across specific ranges of the `amt` column. The transaction amounts were categorized into the following bins: `[0, 700, 900, 1000, 1200, 20000]`. This analysis revealed that transactions are more likely to be fraudulent when the monetary amount falls within the range of 900 to 1000 as shown in 7.

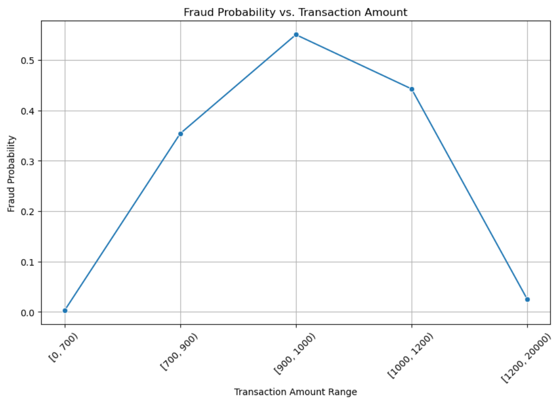


Figure 7: Fraud Frequency across amt ranges

Finally, the same investigations were performed on other columns, including `gender`, `city_pop` (city population), `distance_from_home_to_merchant` (this columns has been created by computing the haversine distance on the latitude and longitude features). However, no significant patterns or insights that could aid in fraud detection have been detected these columns.

2.1 Feature Engineering

Based on the findings from the exploratory data analysis, three additional features were engineered to enhance the predictive power of the model. These features leverage time-dependent cumulative statistics to capture nuanced patterns in the data. The features have been created with strict consideration of time-dependency to prevent `data leakage`, so for each transaction, the probability was calculated based solely on the information available up to that specific point in time. No data from future transactions was used in the computation, ensuring the integrity of the feature engineering process and maintaining a realistic simulation of real-world conditions. Below, the creation logic for `category_fraud_probability`, designed to capture the likelihood of fraud for each transaction category, has been described as an example:

- Transactions were sorted by `category` and `trans_date_trans_time` to ensure temporal order.
- For each category, cumulative counts of transactions and fraudulent transactions were maintained.

- For each transaction within a category, the fraud probability was calculated as the ratio of cumulative fraudulent transactions to total transactions observed up to that point.

Following the same approach the engineered columns `merchant_fraud_probability` and `fraud_prob_by_amt` have been created.

2.2 Further Data Processing

Finally, all procedures related to data processing, including dataset temporal range and dimension selection, data cleaning, the creation of new meaningful features, the removal of irrelevant columns, encoding and scaling of categorical and continuous variables, have been encapsulated in a hierarchy of functions. These functions are organized within a separate notebook, my_utils, which serves as a centralized repository for all custom functions that are re-used throughout the project.

3 The Methodological Approach

This section outlines the methodology adopted for the fraud detection task, detailing the steps undertaken to define, evaluate, and refine the decision models. The methodological pipeline is designed to standardize and, as far as possible, automate the process of model development and evaluation.

For the problem at hand, four distinct types of deep learning models were selected and implemented:

- **Feedforward Neural Networks (FNN)**. Standard architecture with multiple hidden blocks (Dense + BatchNormalization + Dropout) dynamically added, reducing the number of units at each new layer (according to a "reduction rate" hyperparameter) to perform a hierarchical feature selection process. ReLU has been used as Activation Function in the hidden layers, sigmoid as output function and binary cross-entropy as loss function.

- **Feedforward Neural Networks with a custom focal loss function**. Same dynamic structure as the previous FFN but with a custom focal loss function. Focal loss represents a modification of the standard binary cross-entropy (BCE) loss designed to handle class imbalance by focusing more on hard-to-classify examples, as shown in the article [2]. The focal loss function is defined as:

$$\mathcal{L}_{\text{focal}} = -\alpha\,(1 - p_t)^{\gamma}\log(p_t), \tag{1}$$

  where:

  - p_t is the predicted probability for the true class, given by:

$$p_t = \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{if } y = 0 \end{cases} \tag{2}$$

  - $\alpha \in [0, 1]$ is a weighting factor that balances the importance of different classes, particularly useful for imbalanced datasets.
  - $\gamma \geq 0$ is the focusing parameter, which down-weights the contribution of well-classified samples and emphasizes misclassified ones.

- **Autoencoder (AE) based on reconstruction error for fraud detection.** This approach consisted, firstly, in training an autoencoder to reconstruct normal transaction (by excluding frauds) and successively test the model against the whole dataset. The principle behind this was that fraudulent transactions were expected to have higher reconstruction errors (MSE) due to their deviation from normal patterns and that could be detected as anomalies.

  The distribution of MSE across the two target classes have been considered as evaluation of the following strategy.

  The reconstruction error distributions can be used as an "anomaly score feature" that could be added to the original set for further classification process.

- **Long Short-Term Memory (LSTM) networks**. This last deep learning strategy is based on LSTM in order to automatically capture time dependencies in the users' behaviour and use sequences of transactions to detect fraudulent patterns. The input sequences were created by grouping the dataset by ‘cc num‘ (credit card number) and iterating over each user's transaction history. For each ‘cc num‘, sequences of transactions were extracted with a specified length (‘seq length‘), starting from the ‘seq length‘-th transaction. The corresponding fraud label for each sequence was assigned based on the current transaction.
  For this specific kind of model, the time-dependent aggregations expressing fraud-probability across some columns have been excluded, since LSTM should ideally be able to learn these patterns itself by looking at the raw transaction data over time, so providing these pre-calculated features might be redundant or could even hinder the LSTM's ability to learn more complex temporal relationships.

To ensure consistency and comparability across all models, the following standardized workflow was applied to each:

1. **Feature Selection - Feature Engineering:** Identification and selection of relevant features to optimize model performance and fairness metrics. In particular, have been tried experiments excluding and including the time-dependent custom features described in the dataset section.

2. **Hyperparameter Tuning and Custom Model Abstraction:** Hyperparameter tuning was performed on the validation set to optimize model performance against an external dataset. In order to save time, EarlyStopping (patience=5) has been used during training-validation phase. A modular approach was adopted for constructing models, enabling flexible design and hyperparameter experimentation across architectures. This approach involved custom Python functions that dynamically build models with configurable parameters. The hyperparameter fine-tuning process followed a prioritized approach based on the expected impact of each parameter:

   (a) **Number of Blocks**: Adjusted first, as it directly determines model depth and significantly impacts the ability to capture patterns.

   (b) **Units and Reduction Rate**: Tuned to control the model's capacity and gradual complexity across layers.

   (c) **Learning Rate**: Fine-tuned next to ensure stable and effective optimization.

   (d) **Class Weights**: Adjusted to address class imbalance and improve the model's performance on minority classes.

   (e) **Regularization (L1/L2)**: Explored to mitigate overfitting, applied only after optimizing core architectural and training parameters and after comparing training and validation loss across epochs.

   During this phase, models' performance and fairness metrics, with respect to the validation dataset, have been collected and saved to a specific csv file for later evaluation.

3. **Model Selection:** Selection of the best-performing model based on validation scores. Specifically:

   - Precision-Recall AUC (PR-AUC) was used for evaluating the FFN, FFN with custom focal function, and LSTM models. Considering the low frequency of fraud, this metric is more sensible to class imbalance cause it focuses on FP and FN.
   - MSE was used for evaluating the Autoencoder.

4. **Threshold Definition:** Determination of a decision threshold for the sigmoid layer output, aiming to strike the best compromise between recall and F1-score. In particular, the threshold choice followed under the following rule: the possible thresholds were primarily filtered in order to achieve at least 0.7 as recall value and than, the one that best optimized the F1-Score has been chosen. The aim of this simple deterministic rule was to achieve the best compromise between recall and precision (F1-Score) but with a minimum level of recall constraint.

5. **Model Testing:** Evaluation of the selected model on the test set using a set of performance and fairness metrics to ensure robustness and generalization. Additionally, it is noted that for specific use cases, different compromises between recall and precision may be prioritized depending on the application context such as cost of FP vs cost of FN.

6. **Results Registration:** Documentation of the best model's hyperparameters and test set performance in a dedicated dataframe to facilitate comparisons across different model types. The recorded metrics include recall, F1-score, accuracy, PR-AUC, and fairness metrics such as the adverse impact ratio and standardized mean difference (with respect to gender).

To further contextualize the effectiveness of the deep learning approaches, two baseline models were included in the analysis:

- A random classifier, calibrated to mimic the fraud rate in the dataset
- A Decision Tree Classifier, as simpler ML model that can natively give insights about features importance.

These baseline models serve as reference points for evaluating the added value brought by the deep learning methodologies.

3.1 Fainess Metrics

As introduced, the overall metrics set has been enriched with some fairness metrics: Adverse Impact Ratio and Standardized Mean Difference.

The Adverse Impact Ratio has been measured according to the following formula:

$$AdverseImpactRatio(AIR) = \frac{P(\hat{Y} = 1|A = 0)}{P(\hat{Y} = 1|A = 1)} \tag{3}$$

where $A = 0$ represents the disadvantaged group and $A = 1$ represents the advantaged group.
AIR measured the ratio between the frequency of a positive outcome for a subpopulation and the frequency of a positive outcome for a control population. Generally, a positive outcome refers to a favorable event; therefore, in the implementation, it was calculated as the frequency at which a transaction was classified as non-fraudulent. So, to further clarify, in the formula, the term Y=1 represented the non-fraud (favorable) outcome. In the context of this report, the monitored subpopulation was the female group, while the control population was the male group. AIR was applicable only to binary classifications, and its score could theoretically range from 0 to 1. A value of 0 indicated that the protected population received no positive outputs while the control population received all positive outputs, whereas a value of 1 indicated no difference in the distribution of positive outputs between the two groups.

Generally, AIR values greater than or equal to 0.8 were considered indicative of low variation between the disadvantaged population and the control group, in accordance with the "four-fifths rule," also known as the "80% rule." This guideline was frequently applied to assess potential discrimination or disparate impact in employment and other decision-making processes. The rule was commonly used in the United States by regulatory agencies such as the Equal Employment Opportunity Commission (EEOC) as described in [3]. In the "Results and Evaluation" section, the AIC metric has been used to compute differences in both non-fraud transaction (favorable event) and accuracy (named as AIC Accuracy). The last one in order to monitor potentially shifts in prediction accuracy between the two gender.

The Standardized Mean Difference, instead, Measured the difference between the mean scores associated with a given subpopulation and the mean scores of a control population, with the result scaled by a measure of the standard deviation of both populations.

$$StandardizedMeanDifference(SMD) = \frac{\mu_0 - \mu_1}{\sigma} \tag{4}$$

where μ_0 and μ_1 are the mean prediction scores for the advantaged and disadvantaged groups, respectively, and σ is the pooled standard deviation.
SMD was a more flexible metric that could also be applied to continuous values: as the absolute value of SMD increased, the difference between the two subpopulations became more pronounced.

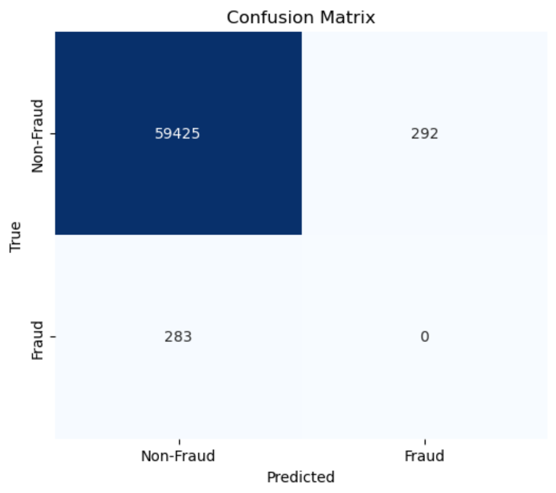
Both formulas were custom-implemented in the notebook my_utils.ipynb and have been computed during validation and test phase using a support dataframe to link predictions to gender column.
As observed in some experimental results, AIR values greater than 1 and negative SMD values could occur when the protected population was associated with a higher frequency of favorable outcomes than the control population.

4 Results and Evaluation

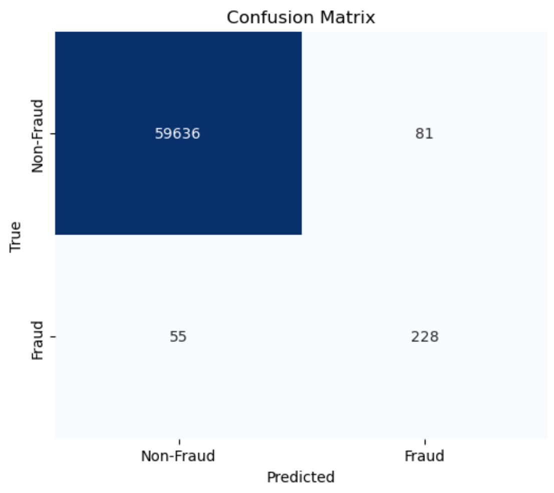
In this section have been presented the results on the validation set for the models of the same type and the results on test set of the best models for each different approach. Models have been evaluated according to confusion matrix and the following metrics: recall, F1-score and PR-AUC. For their purpose as performance baselines, validation phase and fairness metric monitoring have been excluded in RandomClassifier and DTC.

4.1 Baseline Models

In this subsection test results of baseline models have been reported.



(a) Random Classifier Confusion Matrix vs Test Data



(b) DTC Confusion Matrix vs Test Data

Figure 8: Comparison of Confusion Matrices for Random Classifier and DTC

Table 1: Baseline Models Performance

Random Classifier Performance

Metric	Value
Precision	0.0
Recall	0.0
F1 Score	0.0
Accuracy	0.9904
PR-AUC	0.0047

DTC Performance

Metric	Value
Precision	0.73
Recall	0.80
F1 Score	0.77
Accuracy	0.9977
PR-AUC	0.5954

As shown in Figure 9, the Decision Tree model assigns different levels of importance to the features used for classification. The most influential features included: `category`, `fraud_prob_by_amt`, `hourly_fraud_probability`, `amt`, and `age`.

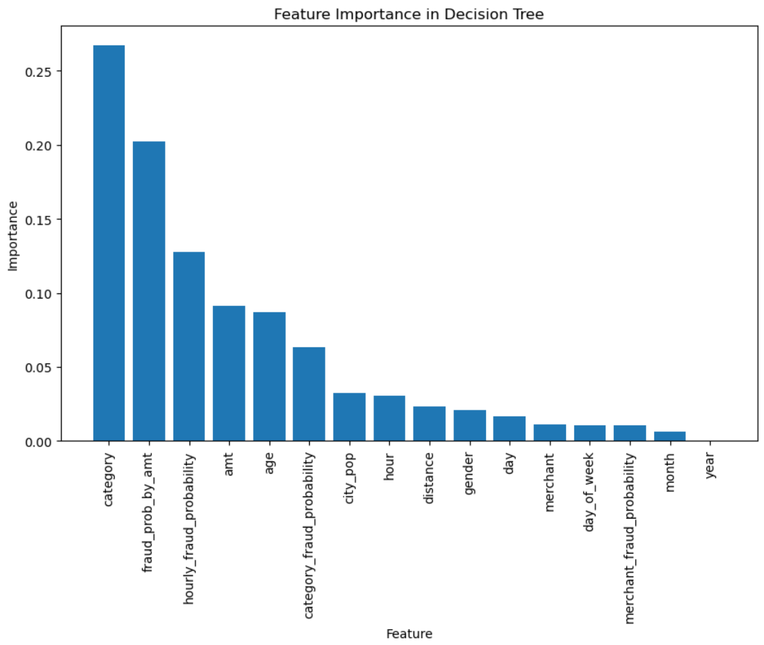


Figure 9: DTC Features importance test data

4.2 Feed-Forward Neural Network

4.2.1 Training History and validation results

Figure 10 illustrates the training history of the FFN model among loss function (BCE), Accuracy and Precision-Recall AUC (PR AUC).

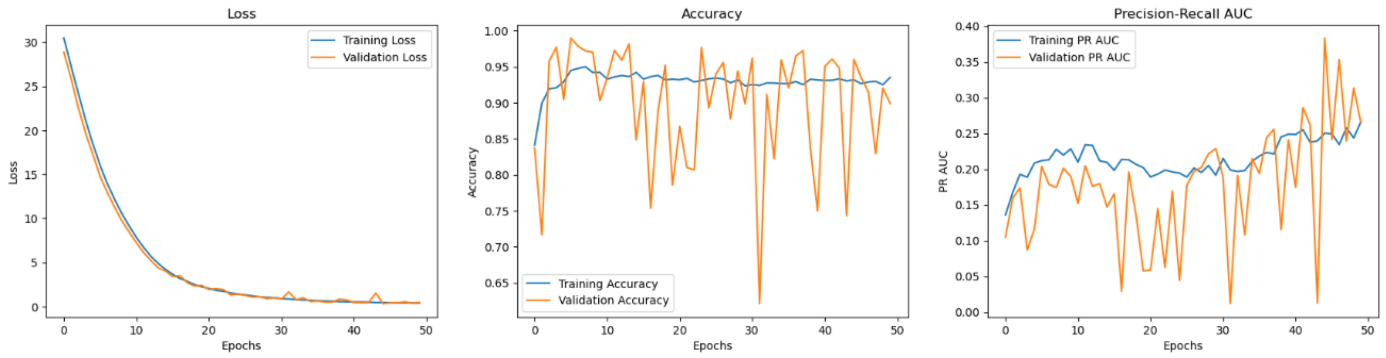


Figure 10: FFN Training History Example

Observations

- **Training PR AUC (Blue Line)**: The training PR AUC gradually increases over time and stabilizes after approximately 30 epochs. Although some fluctuations are present, the overall trend indicates that the model is improving in distinguishing fraudulent cases.
- **Validation PR AUC (Orange Line)**: The validation PR AUC exhibits high instability, with significant fluctuations across epochs. While it sometimes follows the training curve, it frequently drops to very low values.

Across the overall experiments, the best hyperparameters and validation performance for the Feed-Forward Neural Network (FFN) are summarized in Table 2.

Table 2: Best Hyperparameters and Validation Performance for FFN

Best Hyperparameters

Hyperparameter	Value
Num Blocks	9
Units	512
Reduction Rate	0.75
L1 Regularization	0.0001
L2 Regularization	0.0
Dropout	0.0
Learning Rate	5.00E-05
Batch Size	128
Class Weights	{0: 0.502, 1: 106.46}

Validation Performance

Metric	Value
Precision	0.0696
Recall	0.85
F1 Score	0.1287
Accuracy	0.9540
PR-AUC	0.5291
AIC	0.9629
SMD	0.1703

4.2.2 Threshold Definition

The best threshold found according to the recall constraint (Recall $\geq$ 0.7) and maximizing F1-score with the remaining data is 0.97, with Precision: 0.45, Recall: 0.71, and F1-Score: 0.55.

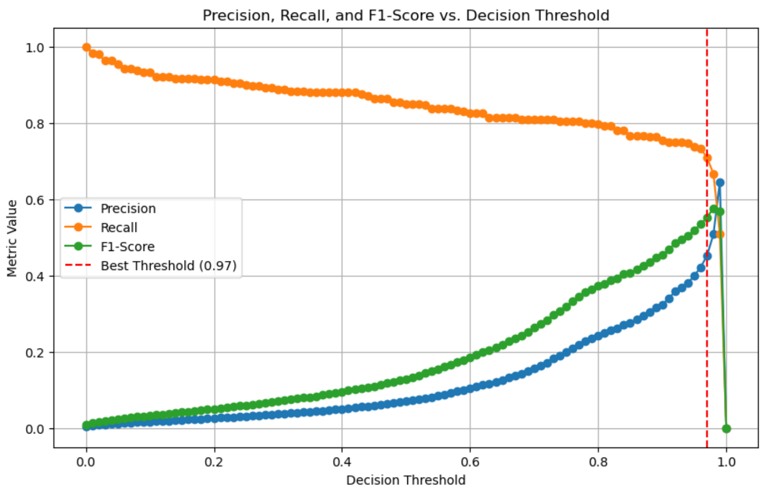


Figure 11: FFN best threshold analysis on validation

4.2.3 Test Results

Above the test results of the best FFN model have been presented:

Table 3: Test Results and Fairness Metrics for the Best FFN Model

Accuracy	0.9955
AUC	0.6192
Recall	0.7739
F1 Score	0.6195
Precision	0.5165
SMD (Non-Fraud Gender)	-0.0277
AIR (Non-Fraud for Gender)	1.0024
AIR (on Accuracy for Gender)	0.9983

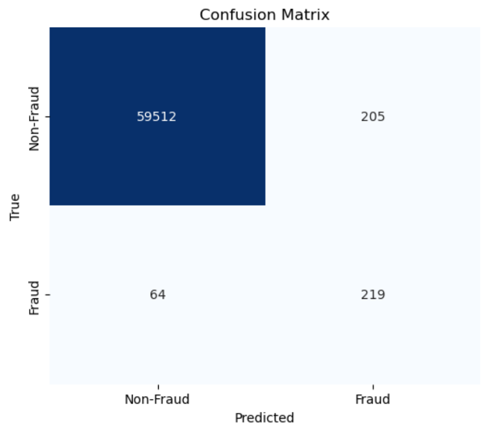


Figure 12: FFN Confusion Matrix vs Test Data

4.3 FFN with Custom Focal Loss

4.3.1 Training History

As for the previous standard FFN model, also in this case, an unstable behaviour has been observed in the validation performance curves.

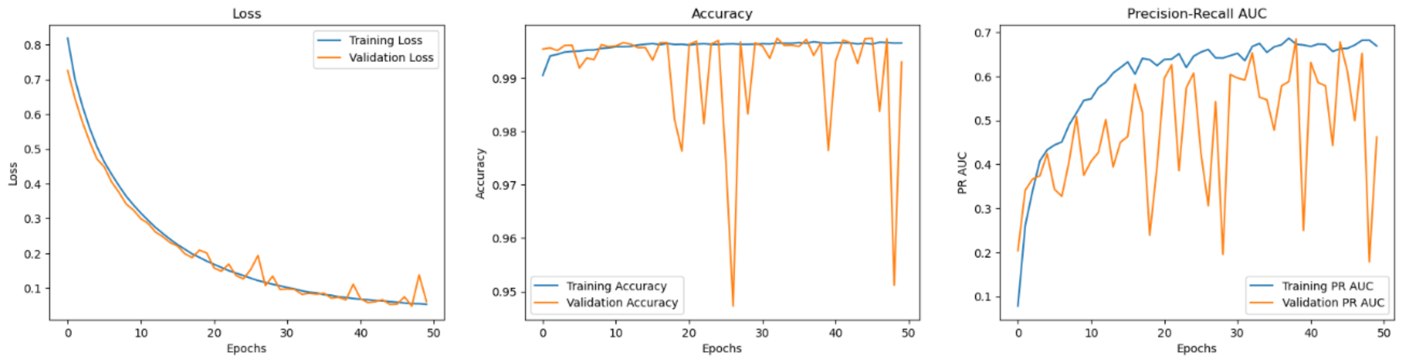


Figure 13: FFN Focal Loss Training History Example

4.3.2 Alpha and Gamma Effects

In the following scatterplots, the effect of alpha and gamma have been shown with respect to performance metric. For the meaning of this subsection only a small subset of the validation experiments have been used in order to remove noise associated to the combination of other hyperparameters. In particular, for the figures related to alpha, only alpha-1 has been used while alpha-0 has been set to a fixed value of 1 in order to better higlight the effects.

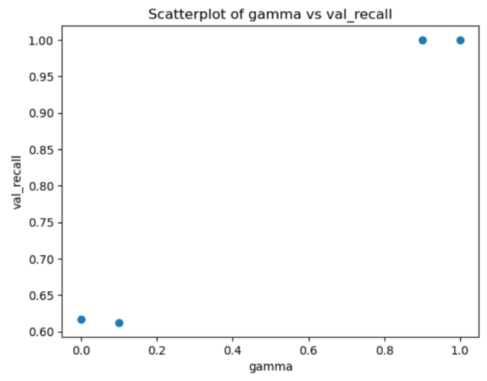


Figure 14: Validation Scatterplot gamma vs recall

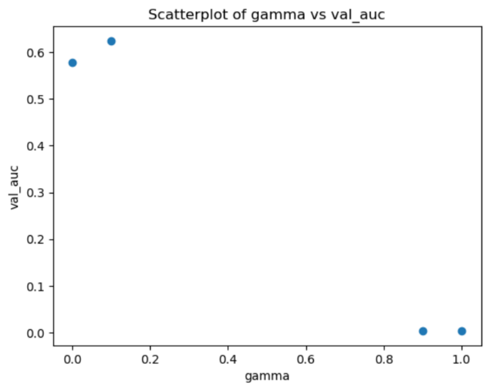


Figure 15: Validation Scatterplot gamma vs pr auc

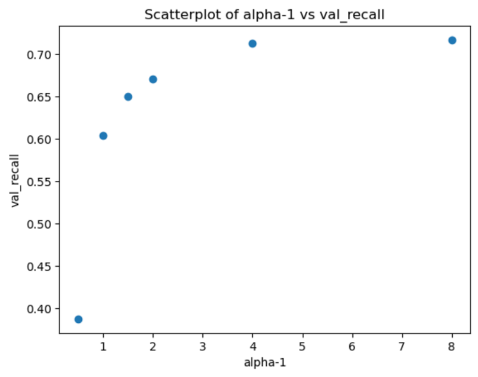


Figure 16: Validation Scatterplot alpha-1 vs recall

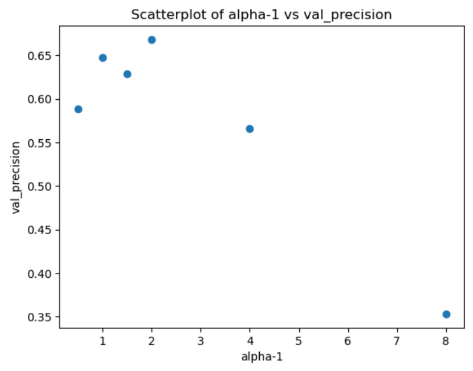


Figure 17: Validation Scatterplot alpha-1 vs precision

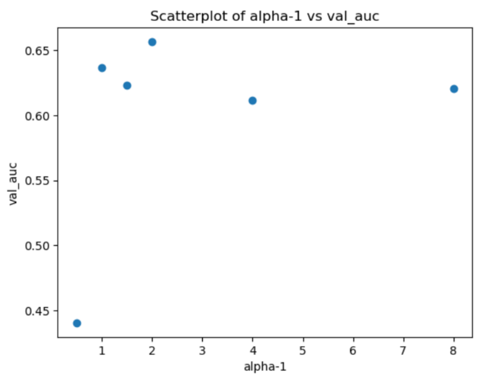


Figure 18: Validation Scatterplot alpha-1 vs PR-AUC

4.3.3 Validation Best Results

Across the overall experiments, the best hyperparameters and validation performance for the Feed-Forward Neural Network with Custom Focal Loss are summarized in Table 4

Table 4: Best Hyperparameters and Validation Performance for FFN with Focal Function

Best Hyperparameters

Hyperparameter	**Value**
Gamma	0.1
Alpha	[1, 2]
Num Blocks	7
Units	256
Reduction Rate	0.75
L1 Regularization	0.0001
L2 Regularization	0.0
Dropout	0.0
Learning Rate	5.00E-05
Batch Size	128
Class Weights	{0: 0.502, 1: 106.46}

Validation Performance

Metric	**Value**
Validation Precision	0.69
Validation Recall	0.61
Validation F1 Score	0.65
Validation Accuracy	0.997
Validation PR-AUC	0.64
Validation AIC	1.00

4.3.4 Threshold Definition

The best threshold found according to the recall constraint (Recall ≥ 0.7) and maximizing F1-score with the remaining data is 0.63, as shown in Figure 19 with Precision: 0.56, Recall: 0.70, and F1-Score: 0.62.

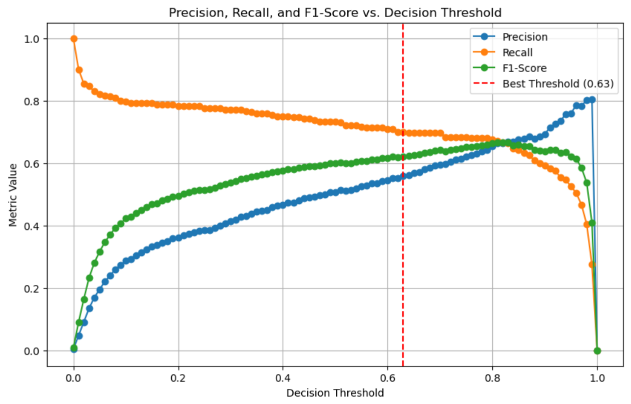


Figure 19: FFN with Focal Function best threshold analysis on validation

4.3.5 Test Results

Above are the test results of the best FFN model with Focal Function:

Table 5: Test Results and Fairness Metrics for the Best FFN Model

Test Performance	
Accuracy	0.9968
PR-AUC	0.7062
Recall	0.7915
F1 Score	0.7022
Precision	0.6310
Fairness Metrics	
SMD (Non-Fraud Gender)	-0.033
AIR (Non-Fraud for Gender)	1.0026
AIR (on Accuracy for Gender)	0.9989

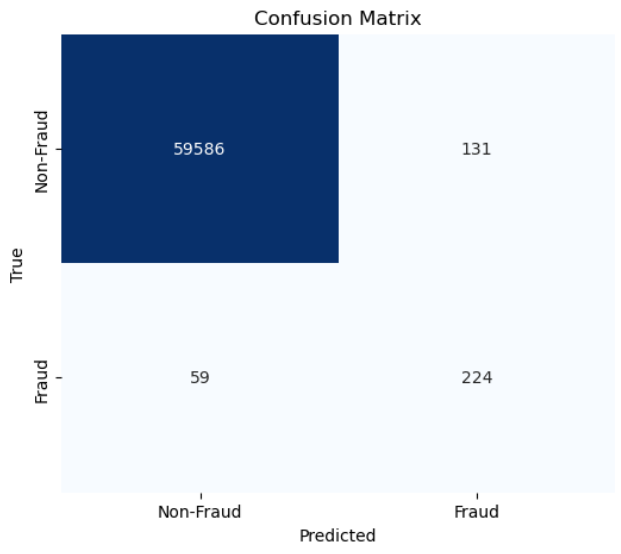
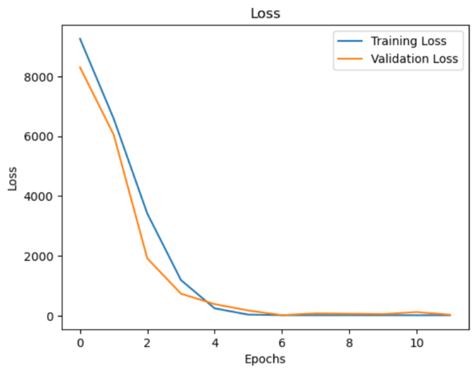


Figure 20: FFN Confusion Matrix vs Test Data

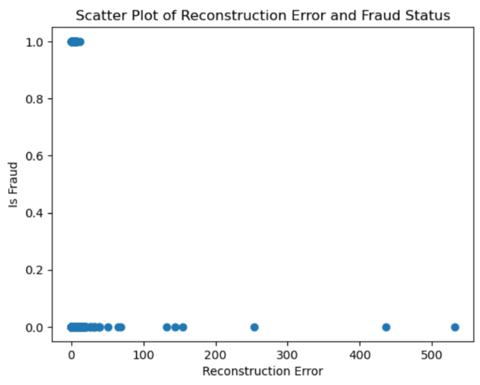
4.4 Autoencoder

4.4.1 Training History

The training history of the autoencoder is shown in Figure 21a. The training loss starts at a very high value of Mean Squared Error and decreases quickly in few epochs like the validation loss. In the Figure 21b instead, as been shown the reconstruction error values across the two label classes.



(a) Autoencoder training history



(b) Reconstruction Error grouped by is_fraud label

4.5 Long-Short Term Memory Network

4.5.1 Training History

As shown in the Figure 5.2.1 below, the Training History of the LSTM model was visualized across 14 epochs, displaying training and validation performance for loss, accuracy, and precision-recall AUC. Training loss decreased while validation loss initially mirrored this trend but later fluctuated and slightly increased. Training precision-recall AUC steadily rose, but validation precision-recall AUC, similar to accuracy, fluctuated.

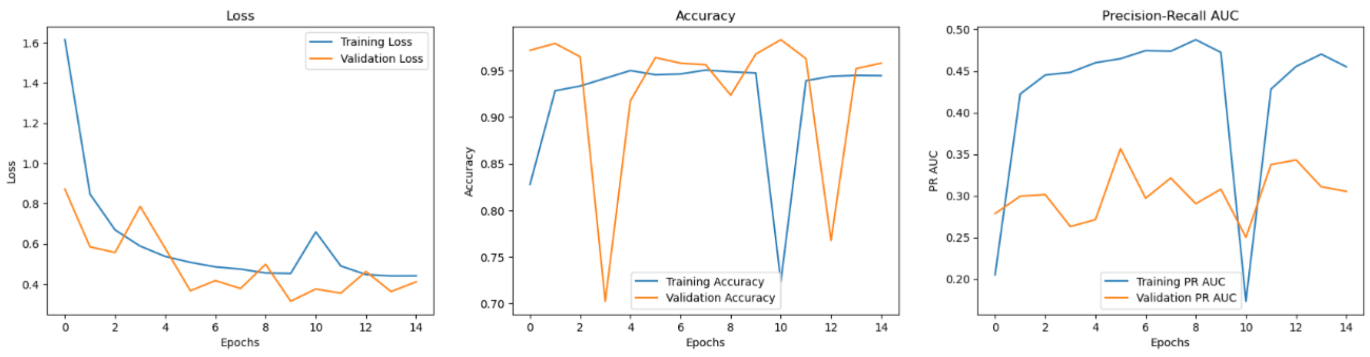


Figure 22: LSTM Training History Example

4.5.2 Validation Best Result

Across the overall experiments, the best hyperparameters and validation performance for the LSTM Network are summarized in Table 6

Table 6: Best Hyperparameters and Validation Performance for LSTM

Best Hyperparameters

Hyperparameter	**Value**
Num Blocks	3
Sequence Length	25
Units	512
Dropout	0.0
Learning Rate	0.0001
Batch Size	128
Regularization	L1
Reg. Strength	0.0001
Class Weights	{0: 0.502, 1: 109.398}

Validation Performance

Metric	**Value**
Validation Precision	0.063
Validation Recall	0.618
Validation F1 Score	0.115
Validation Accuracy	0.968
Validation AUC	0.332

4.5.3 Threshold Definition

The best threshold found according to the mentioned constraints was 0.35, as shown in Figure 23 with Precision: 0.05, Recall: 0.70, and F1-Score: 0.09.

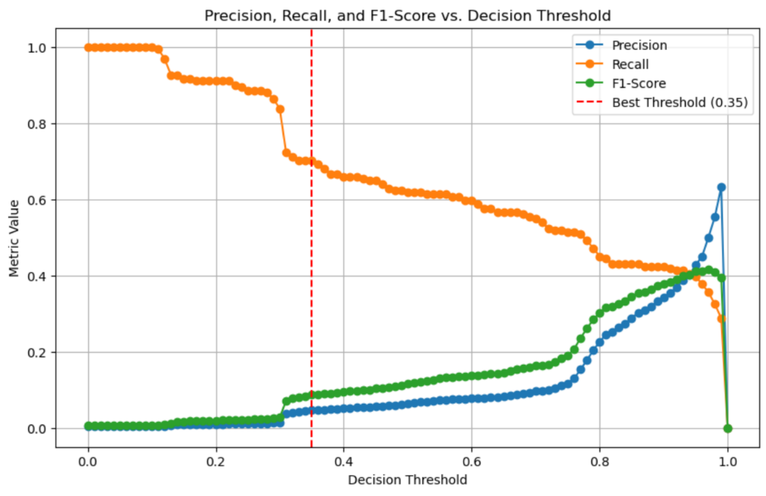


Figure 23: LSTM best threshold analysis on validation

4.5.4 Test Result

Above are the test results of the best LSTM model:

Table 7: Test Results and Fairness Metrics for the Best LSTM Model

Test Performance	
Accuracy	0.9514
PR-AUC	0.3527
Recall	0.6075
F1 Score	0.0864
Precision	0.0465

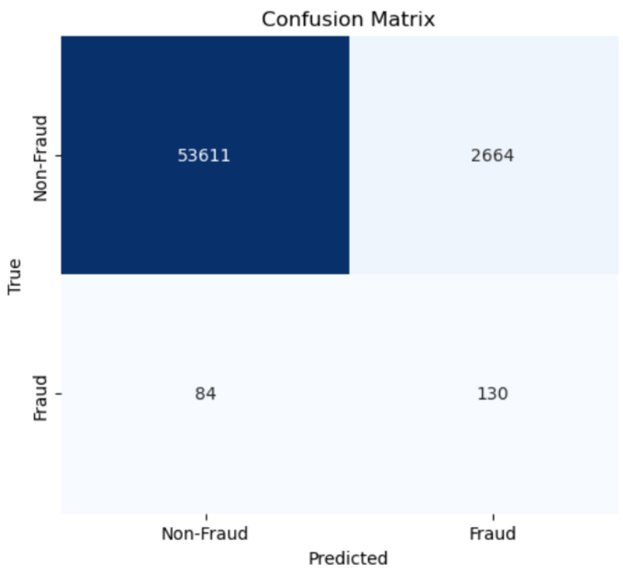


Figure 24: LSTM Confusion Matrix vs Test Data

4.6 Models Performance Comparison

In the following subsection the test results across all models have been summarized according to the main performance metrics:

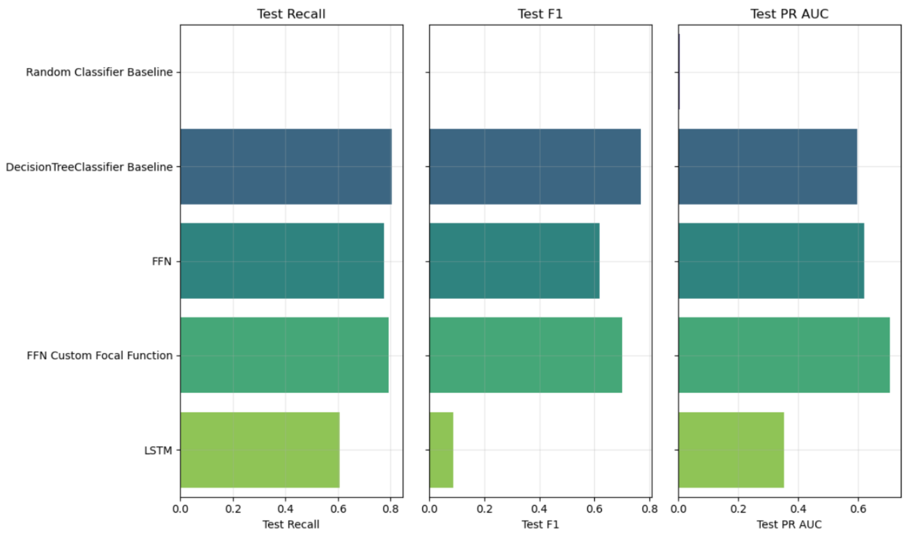


Figure 25: Models Performance Summary

As shown in the Figure 25 above, the DTC Baseline outperformed all the Deep Learning model on the Recall and F1-Score metrics. Only the FFN with Focal Loss presented a better performance on PR-AUC metric.

5 Discussion

In this section, intepretation and further considerations about the above results have been performed for each model hiligthing those results that have brought meaningful insights.

5.1 Baselines Models

5.1.1 RandomClassifier

As expected, due to the high unbalance nature of the distribution, the RandomClassifier got an overall high accuracy score related to the fact that almost all instances have been associated to the majority class. Moreover, as shown in the Table 1 left, the model ability to correctly classify fraud transaction has been happen to be zero.

5.1.2 Decision Tree Classifier

The DTC has shown overall good performance (Tab 1 Right) for a simple not fine-tuned model. In particular, it achieved great ability to correctly classify fraud instances (recall: 80.57%) and medium-high precision (73.79%) in reducing False-Positive. The model's performance suggested that:

- Fraudulent transactions likely follow well-defined rules based on a little subset of features, making them distinguishable by a simple decision tree.
- Dateset used has shown to be of adequate size for a ML classic techniques and with simple non-linear relationship that can be captured through hierarchical decision-making .

As shown in Figure 9 (DTC's Feature Importance), engineered features like historical fraud probabilities grouped by transaction amount, transaction hour and transaction category have been ranked among the top six most influential features in model decisions. This suggests that fraud likelihood is highly dependent on temporal and categorical patterns, likely due to specific fraud behaviors repeating within particular transaction types and time frames.

5.2 Feed-Forward NN

5.2.1 Training History

The high variance in validation Accuracy and PR-AUC, as shown in Figure 10, indicated that the model struggled to generalize well on validation data. This could possibly be explained by the combination of two conditions. The weakness of the regularization factor used during training phase could led a role in these result. However, most importantly, it was observed that the dataset contained very few fraud cases, a situation further exacerbated in the validation set, where the smaller size increased the likelihood of individual batches containing too few or no fraudulent transactions. This, in turn, could lead to unstable and unreliable performance metrics.

Considering the aspects highlighted so far, several potential pathways to improve the model's performance were identified:

- **Explored further regularization techniques and strengths**.
- **Improved Data Representation**: Oversampling of the minority class with synthetic data generation techniques, such as SMOTE or undersampling of the majority.
- **Experimented with a learning rate scheduler to stabilize training**.
- **Stratified Mini-Batches**: Ensured that each mini-batch contained a representative number of fraudulent cases to mitigate extreme fluctuations in validation metrics.

5.2.2 Hyperparameters Fine-Tuning

The analysis of the relationships between hyperparameters and validation performance proved to be complex, as the exploration was not exhaustive due to computational constraints. A hierarchical approach was employed to improve efficiency, allowing the search to progressively focus on the most promising hyperparameters following an elimination-based logic ("winner stays"). However, this approach potentially reduced visibility into more complex interactions between hyperparameters, introducing a selection bias: hyperparameter values that initially performed well received more opportunities for testing and refinement, while alternative values that might have yielded superior results remained underexplored. This effect could have been mitigated through more advanced optimization techniques, such as Bayesian search or population-based training.

5.2.3 Threshold Definition

The obtained threshold of 0.97 and the realative Figure 11 suggested that the greatest part of fraud instances has classified with a very high probability score (greater than 0.9), meaning that the model has shown good performance in recall score. However a significant number of non-fraud transactions have received medium-high probability score meaninig that the model struggled to reduce the number of False-Positives.

5.2.4 Test Results and Further Considerations

The confusion matrix and test scores (Table 3) reflected the previously stated observation that the model demonstrated a strong recall but did not achieve a high precision score. As further possible improvement, an additional analysis could have been conducted to determine whether the challenging cases related to false-positive classifications followed a specific pattern that could have been leveraged to enhance feature engineering and improve the model's performance.

5.3 FFN with Focal Function

5.3.1 Training History

Due to the similarity with the previous FFN case, intepretation and improvements related to the training history of the FFN with the Focal Loss function can be referenced in subsection 5.2.1.

5.3.2 Role of α and γ

This subsection has been written with the goal of analyze the effects on model's performance of the hyperparamters α and γ related to the focal function. This subject has been performed firstly theoretically and successively considering the results across the experiments.
The mathematical contribution of the focal function is shown considering the following formulas where equation 5 is the focal loss and equation 6 below the relative gradient computation:

$$\mathcal{L}_{\text{focal}} = -\alpha(1-p_t)^\gamma \log(p_t) \tag{5}$$

$$\frac{\partial \mathcal{L}_{\text{focal}}}{\partial p_t} = -\alpha(1-p_t)^\gamma \left(\frac{1}{p_t}\right) + \alpha\gamma(1-p_t)^{\gamma-1}\log(p_t) \tag{6}$$

The parameter α assigns different weights to each class label: if the dataset is unbalanced, setting α (alpha-1) higher for the minority class ensures that minority-class examples (with higher α) contribute more to weight updates.
The term $(1-p_t)^\gamma$, instead, suppresses the loss for well-classified examples (where p_t is close to 1). This increases the gradient magnitude for misclassified examples (small p_t), making the model learn more from difficult cases. In particular, "The focusing parameter γ smoothly adjusts the rate at which easy examples are downweighted" [2]. By down-weighting easy examples, focal loss prevents them from dominating gradient updates.
Theoretically, this allows for a more stable training process where misclassified examples remain influential longer, leading to a more refined decision boundary.

However, in practical terms, several experiments found that higher values of γ (near or greater than 1) led to better recall (Figure 14), but these results could not be considered optimal, as it was observed that for these values of γ, the model tended to classify all transactions as fraud. As a consequence, the PR AUC dropped drastically (Figure 15) in both training and validation data.
This behavior suggested that decreasing too much the magnitude of "simple cases" (1 - $p_t \approx 0$) biased the model, making it too sensitive to outliers or difficult examples that represented only a very small subset of the data, which could cause it to overfit to these cases at the expense of generalization.
Another possible interpretation is that high values of γ for well-classified examples led to shrink the whole gradient towards zero causing a vanishing gradient case.
At the end, the best performance has been achieved by setting γ to 0.1 so that the term $(1 - p_t)^\gamma$ became close to 1 mimicking a weighted cross entropy so the model retained sufficient gradient updates across both easy and hard examples (with a very weak down-weighting of the simple cases), ensuring better generalization on an imbalanced dataset but with a higher weight for class 1 (as the alpha term is conserved in the loss).

Instead, concerning the practical effect of α on model's performance, has been observed that increasing α_1 led to better recall Figure 16 but worse precision Figure 17. This was because a higher value of α_1 placed more emphasis on the minority class (fraud cases), causing the model to be more sensitive to these examples and, while this increased the model's ability to correctly identify fraudulent transactions, it also resulted in a higher number of false positives, which in turn degraded precision. After experimenting with different hyperparameters values, the best compromise was found to occur with $\alpha_0 = 1$ and $\alpha_1 = 2$, balancing recall and precision effectively for the fraud detection task (Figure 18).

5.3.3 Threshold Definition

According to Figure 19, unlike the standard FFN, the focal loss led to a more balanced compromise between recall and precision: a threshold of 0.63 was still able to capture more than 70% of fraud transactions with medium level of precision. This result represented a better ability (with respect to the standard FFN) to assign higher output scores to fraud instances while keeping low scores for non-fraud cases.

5.3.4 Test Results and Further Considerations

Considering the results shown in Table 5 and confusion matrix 20 the second approach brought a slight better recall and a much better balance between recall and precision. This result has been expected as the Focal Function has been precisely designed to achieve better performance on the fraud cases.
Beyond the improvements discussed in the section on the standard FFN, for this model, which demonstrated superior PR-AUC, further refinement of the decision threshold selection process could have been beneficial. The threshold should have been adjusted based on the primary objective—whether maximizing recall to capture more fraud cases or prioritizing precision to reduce false positives.

5.4 Autoencoder

The autoencoder training history indicated that the model effectively learned a latent space conducive to reconstructing non-fraudulent instances in both the training and validation sets (Figure 21a). However, reconstruction errors for fraudulent instances did not appear significantly higher than those for non-fraudulent instances, precluding their classification as anomalies based on this metric. This suggested that in the used dataset fraud didn't incorporate simple anomaly patterns. Instead, it appeared that the abundance of non-fraudulent instances resulted in a greater number of outlier-induced reconstruction errors within this majority class (Figure 21b). Consequently, the incorporation of reconstruction error as an additional feature was deemed unviable and subsequently abandoned.

A possible alternative use of the autoencoder model could be as the foundation of another classifier, where the latent space obtained from the encoder serves as the input for the classification task.

5.5 LSTM

5.5.1 Training History

Considering the training loss decrease in the Figure 22, has been possible to observe that the model actually learnt time patterns in the training sequences. However, focusing on the last plot on the right, the PR-AUC curve shown a low-medium performance level on the training data and a low performance on the validation dataset, meaning that the model didn't perform well on fraud cases in both training and validation (as expected due to the unbalanced nature of the dataset). Furthermore it struggled to generalize with respect to unseen data.

5.5.2 Threshold Definition

Figure 23 reported that the model could achieve sufficient level of Recall only reducing the classification threshold to 0.35, this highlighted the weakness of the LSTM with respect to False Positive as definitely showed by the poor level of precision.

5.5.3 Test Results and Performance Analysis

The results versus the test dataset confirmed what has been observed in validation performance with slight differences in the main metrics.

The suboptimal performance of this model was attributable to several factors. Firstly, LSTM training required significantly more time than that of other models, thus limiting the extent of hyperparameter exploration. Specifically, sequence length necessitated restriction to small values due to its substantial impact on training duration. Secondly, the data preparation process for LSTM input necessitated the filtering of certain cc_num values lacking a sufficient number of transactions for sequence creation. Consequently, LSTM models were trained on a reduced number of instances (ranging from 25,000 fewer to greater reductions with increasing sequence length). Furthermore, the data processing procedure slightly altered the fraud frequency distribution across the dataset, resulting, for instance, in some experiments, in a validation set with 26% less fraud and a test set with 17% less fraud. As previously noted, these models were trained with a different feature set, as the aggregated fraud probability engineered features were excluded. Lastly, it is possible that the temporal behavior of cc_num values does not constitute the optimal approach for discerning effective classification patterns.

5.6 Consideration with respect to Baselines

As shown in Figure 25 the DTC outperformed all the Deep Learning model across all the metrics except for PR-AUC where the FFN with focal loss has demostrate to be better. The reasons behind the fact that a simple DTC used as baseline performed so well on this task could be related to the following aspects:

Handling Imbalanced Data Effectively: Decision trees are inherently adept at handling imbalanced datasets. Their structure could have facilitated the creation of splits that effectively isolated the minority class (fraudulent transactions), even with a significantly lower number of instances.

Simplicity: Decision trees are simple models. They could learn relatively straightforward rules that distinguished fraudulent activities from legitimate ones. In fraud detection, the patterns indicative of fraud were potentially straightforward (e.g., large transactions within short periods, unusual locations), and a decision tree could capture these rules directly. Deep learning models, conversely, could have been often excessively complex for such uncomplicated patterns, leading to potential overfitting or difficulties in training.

Feature Engineering: The success of a decision tree is heavily dependent on effective feature engineering. Provided features were highly discriminative of fraudulent behavior, the decision tree could readily identify them and produce accurate classifications. The observed strong performance suggested that the provided features were well-suited to this type of model.

Limited Data for Complex Deep Learning: Although 400,000 instances might appear substantial, it was possible that, considering the complexity of patterns deep learning models attempt to learn, this dataset proved insufficient. Deep learning models, particularly LSTMs, typically require considerable amounts of data to learn intricate temporal dependencies. With insufficient data to learn complex patterns, they might overfit or underperform.

Hyperparameter Tuning: Deep learning models possess numerous hyperparameters requiring careful tuning. It was possible that the deep learning models were not optimally tuned, also for the greater training time needed, leading to suboptimal performance.

5.7 Fairness Considerations

The performance metrics were monitored and recorded during the various experiments conducted on the approaches with the best performance levels (FFN and FFN with focal loss). In no cases were found values of AIR lower than 0.8 or absolute values of SMD greater than 0.2 . This indicates that no models exhibited discriminatory biased toward any subgroup of the "gender" column.

6 Conclusions

Throughout this project, various Deep Learning approaches were explored and evaluated for the task of fraud detection. This investigation enabled the validation of both the strengths and limitations of Deep Learning methods for this specific use case, in comparison to simpler and computationally less intensive classical Machine Learning approaches.

The experiments yielded promising results for certain model categories, particularly Feedforward Networks (FFN) with focal loss functions. However, all approaches could benefit from further analyses and experimentation, as outlined in the corresponding discussion section.

Additionally, an operational example was provided to integrate performance metrics with fairness metrics, promoting best practices in scenarios where a model's behavior may have discriminatory impacts on a demographic subpopulation.

References

[1] European Union, "Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)," https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1689, 2024, official Journal of the European Union, L 277, 12 July 2024, pp. 1–60.

[2] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *arXiv preprint*, 2017. [Online]. Available: https://arxiv.org/abs/1708.02002

[3] P. Hall, J. Curtis, and P. Pandey, *Machine Learning for High-Risk Applications.* O'Reilly Media, Inc., April 2023. [Online]. Available: https://www.oreilly.com/library/view/machine-learning-for/9781098139819/