UNIVERSITÀ DI PISA

Distributed Data Analysis and Mining

Fraud Detection in Banking

Authors:

Lorenzo Cascone

Salvatore Puccio

Alessia Ferrero

Emiliano Marrale

ANNO ACCADEMICO 2024/2025

Indice

Capitolo 1

Data understanding and preparation

The dataset used in this analysis is a synthetic Bank Account Fraud Dataset, simulating real-world banking fraud scenarios. This dataset reflect diverse characteristics associated with fraudulent banking transactions, providing a robust foundation for developing and evaluating fraud detection models. Before releasing the dataset it's been completely anonymized to safeguard the privacy of the people data in it. Our **analysis** will **focus** on understanding **which attributes are most important in determining fraud** and how to prevent them.

1.1 Features Description

As described by table 1.1 the dataset is composed by 32 columns.

Bank Account Fraud Dataset Features		
Feature	**Description**	**Range / Values**
fraud_bool	Fraud label (1 = fraud, 0 = legit)	0, 1
income	Applicant's annual income (quantiles)	[0, 1]
name_email_similarity	Similarity between name and email	[0, 1]
prev_address_months_count	Months at previous address	[-1, 383]
current_address_months_count	Months at current address	[-1, 406]
customer_age	Applicant's age in decade bins	[10, 90]
days_since_request	Days since application submission	[0, 78]
intended_balance_amount	Initial balance at application	[-1, 108]
payment_type	Anonymized payment plan type	5 distinct values
zip_count_4w	Applications in same zip in 4 weeks	[1, 5767]
velocity_6h	Application velocity in last 6 hours	[-211, 24763]
velocity_24h	Application velocity in last 24 hours	[1329, 9527]
velocity_4w	Application velocity in last 4 weeks	[2779, 7043]
bank_branch_count_8w	Applications in the same branch in 8 weeks	[0, 2521]
date_of_birth_distinct_emails_4w	Unique emails with same birth date in 4 weeks	[0, 42]
employment_status	Anonymized employment status	7 distinct values
credit_risk_score	Internal credit risk score	[-176, 387]
email_is_free	Free (1) or paid (0) email domain	0, 1
housing_status	Current residential status (anonymized)	7 distinct values
phone_home_valid	Validity of home phone number	0, 1
phone_mobile_valid	Validity of mobile phone number	0, 1
bank_months_count	Months at previous bank (if any)	[-1, 31]
has_other_cards	Other cards with the bank (0 or 1)	0, 1
proposed_credit_limit	Proposed credit limit for applicant	[200, 2000]
foreign_request	Request from outside bank's country	0, 1
source	Application source (browser or app)	INTERNET, TELEAPP
session_length_in_minutes	Session length on banking website	[-1, 107]
device_os	Operating system of the device	5 distinct values
keep_alive_session	Session keep-alive setting	0, 1
device_distinct_emails_8w	Unique emails from device in last 8 weeks	[0, 3]
device_fraud_count	Count of fraud applications by device	0, 1
month	Month when application was submitted	[0, 7]

Figura 1.1: Description of features in the Bank Account Fraud dataset.

1.2 Distribution of fraud transactions

Understanding the distribution of fraud transactions is crucial, through that we can explore potential biases and identify the need for resampling. Our analysis of the target variable led, as expected from this kind of datasets, in observing an unbalanced dataset towards the positive class. The dataset contains 990000 non fraudolent transaction and 10000 fraudolent transactions.

1.3 Feature Distribution Comparison: Fraudulent vs. Non-Fraudulent Transactions

To gain deeper insights into the characteristics that distinguish fraudulent transactions, it is essential to analyze the distribution of each feature with a comparative focus on both fraudulent and non-fraudulent transactions. While general feature distribution provides an overview, examining the distribution relative to fraud labels allows us to identify specific patterns, anomalies, or behaviors that may correlate strongly with fraudulent activity.

1.3.1 Missing values

We examined the missing values distribution among the features containing them and we found that when examining features like prev_address_months_count, intended_balcon_amount, and bank_months_count, it becomes apparent that a considerable number of missing values exist in their original distribution. However, despite these missing values, these features still hold informative value and exhibit correlation with the target feature.

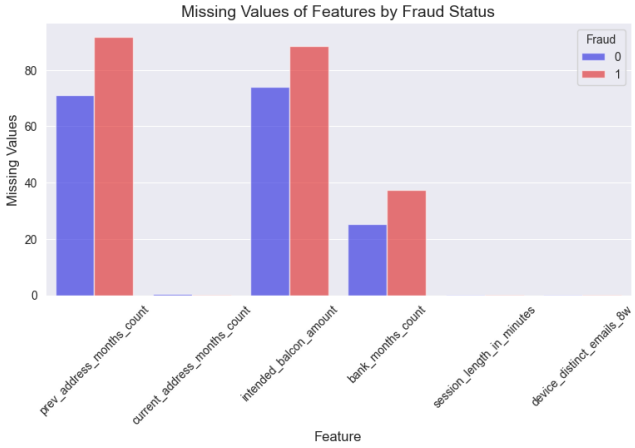

Figura 1.2: Distribution of missing values

1.3.2 Distributions of other values

Looking at the distribution of numerical values in image 1.3 we can observe that many features shows distinct patterns in fraud and non fraud cases; this could help us understanding when fraud cases occurs. For example velocity_4w, name_email_similarity and credit_risk_score patterns could suggest that fraud cases tend to occur at specific similarity rates, with higher velocity ranges and among certain credit scores. In contrast,

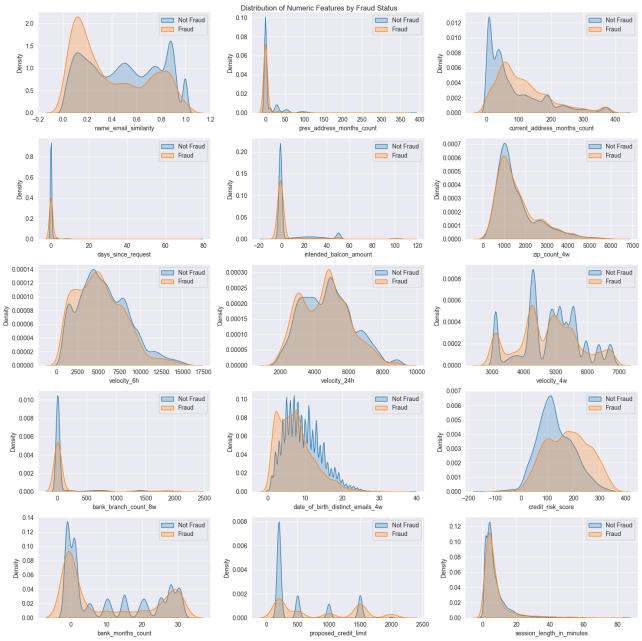

Figura 1.3: Distribution of numerical values

features like days_since_request have similar distributions for both fraud and non-fraud cases and this could indicate that they may contribute less to fraud detection. These patterns highlight which features could be valuable for a fraud detection model, suggesting that those with clear distribution differences between fraud and non-fraud are likely to enhance predictive performance.

1.3.3 Outliers

During the boxplot analysis, we observed the presence of potential outliers in several features. While outliers are often removed to ensure a cleaner dataset and reduce noise, we decided to retain them in this case, outliers may represent unusual but genuine behavior that could be indicative of fraudulent activity. Removing these values could mean losing valuable information that may help the model distinguish between normal and anomalous transactions.

1.4 Feature selection

Before performing classification tasks and building our classification model, we explored the existence of *zero variance* features and *correlation*; after that we preprocessed our data.

1.4.1 Removal of zero variance features

Features with zero variance do not provide discriminatory information for classification and can introduce noise into the model. To identify and remove such features, we calculated the variance for each numerical column in the data set. Columns with zero variance were dropped; this ensured that only informative features remained for subsequent steps. In this step we removed the 'device_fraud_count' feature, with a variance of zero.

1.4.2 Pearson correlation analysis

Highly correlated features can introduce redundancy in the dataset, which can negatively affect the classification model by increasing complexity without adding useful information. To address this, we computed the pairwise Pearson correlation between the features. In the end we noticed that no features had a correlation that high to justify it's removal so the dataset remained the same.

1.4.3 Feature scaling and encoding

To prepare the features for machine learning algorithms, we applied the following preprocessing techniques:

- **Numerical Features**: Numerical features were normalized using *min-max normalization* to scale the values to a range of 0 to 1. This ensures that features with larger ranges do not dominate the training process, especially for distance-based models such as Gradient Boosted Trees (GBT).
- **Categorical Features**: Categorical features were processed differently according to the algorithm requirements:
  - For algorithms requiring numerical input, we applied *String Indexing* to convert categorical values into integer indices.
  - For other models, we used *One-Hot Encoding* to represent categorical features as binary vectors, preserving complete categorical information without introducing artificial ordinal relationships.

This combination of normalization and encoding ensured that our features were appropriately scaled and formatted for the subsequent modeling steps.

Capitolo 2

Clustering Methods

Introduction

In this section, we explored clustering methods in order to analyze bank account fraud detection data. To do this, we compared KMeans, DBSCAN, and Bisecting KMeans, evaluating their performance with respect to specific parameters and using them to identify patterns and biases within the dataset.

2.1 Data Preparation

The dataset was first prepared to ensure compatibility with clustering algorithms, particularly those that rely on distance metrics. Categorical attributes (`payment_type`, `employment_status`, `email_is_free`, `housing_status`, `device_os`, `source`) were handled by dropping low-variance features (e.g., `source`) and applying a two-step encoding process. First, the `StringIndexer` was used to convert categorical values into numerical indices, followed by `OneHotEncoder` to eliminate hierarchical assumptions. Afterward, the target variable `fraud_bool` were removed.

2.2 Methodology

We applied three clustering algorithms: KMeans, DBSCAN, and Bisecting KMeans, each evaluated based on metrics like the Silhouette Score and PCA visualizations. Key points for each method include:

2.2.1 KMeans

We first explored cluster sizes from 2 to 20, selecting $k = 11$ as the optimal number based on a trade-off between Silhouette Score and SSE. Then, we used PCA to reduce dimensionality for visualization (2.2), revealing significant overlap between some clusters (e.g., Cluster 1 and Cluster 5) and distinct separation for others (e.g., Cluster 0 or Cluster 4). The cluster sizes obtained with this algorithm were satisfying, as we can see from Figure (2.1) Finally, a Centroid analysis showed that large clusters like Cluster 0 represented low-risk, financially stable individuals, while small ones like Cluster 1 captured high-activity or potentially at-risk users.

2.2.2 DBSCAN

We implemented this algorithm using a combination of PySpark and scikit-learn, with parameters (`eps` and `min_samples`) determined adaptively based on data density. The model dentified 8 clusters and a noise group (-1). Cluster 0 dominates in size, containing almost all the data. We finally applied PCA visualization, which highlighted smaller, distinct clusters (e.g., Clusters 4 and 8) alongside the predominant Cluster 0 (2.3). Principal components were interpreted as reflecting socioeconomic patterns and behavioral stability.

2.2.3 Bisecting KMeans

We used PySpark's MLlib for hierarchical clustering, evaluating cluster sizes from $k = 2$ to $k = 10$ based on the Silhouette Score. The PCA visualization showed distinct cluster separations with varying sizes, where larger clusters captured general behaviors and smaller clusters reflected specific patterns or anomalies (2.4).

2.3 Conclusion

The clustering analysis provided valuable insights into the dataset's structure and highlighted key strengths and limitations of each method: KMeans, while effective at identifying general groupings, clusters were not well-separated due to potential noise and overlaps. Then, DBSCAN proved robust to outliers and capable of identifying arbitrary-shaped clusters, but relied heavily on density-based parameter tuning. Finally, Bisecting KMeans offered hierarchical insights and handled large-scale data efficiently, but some clusters showed overlapping behaviors.

In all three methods, we observed issues with overlapping clusters, likely due to the presence of noise and outliers, which we deliberately decided to retain in the dataset as they provide significant insights. Furthermore, across all methods, we noted differences in cluster sizes: larger clusters consistently represented the general behaviors of users, while smaller clusters captured fraudulent operations or anomalous patterns.

Overall, combining PCA, Silhouette Scores, and clustering visualizations allowed for a thorough evaluation of the methods, offering interpretable insights into customer behaviors and potential fraud patterns.

Fraud Patterns Identified

The clustering analysis revealed several fraud patterns across the dataset:

- **High Transaction Velocities:** Fraudulent clusters, such as smaller clusters identified in KMeans and DBSCAN (e.g., Cluster 1 in KMeans), often exhibit elevated transaction velocities over short and long periods (e.g. with ce troids, `velocity_24h`, `velocity_4w`). This indicates frequent transaction activities that may deviate from typical user behavior.
- **Low Stability Indicators:** Fraudulent clusters showed lower stability in features such as `prev_address_months_count` and `current_address_months_count`, reflecting users with transient behaviors or inconsistent residence histories. This is often a red flag for fraudulent accounts.
- **Higher Credit Risk Scores:** Clusters identified as potentially fraudulent tend to have above-average `credit_risk_score`, reflecting higher default or risk likelihood.
- **Low Intended Balance Amounts:** Fraudulent patterns are associated with lower `intended_balance_amount` values, which could indicate attempted misuse or fraud rather than genuine, high-value transactions.
- **Unusual Combinations of Features:** Clusters with anomalies in behavioral and socioeconomic indicators, such as mismatched `income` and `name_email_similarity` values, highlight potential identity-related fraud.

These patterns consistently emerged across different clustering methods, indicating that smaller clusters often represent fraudulent or anomalous user behaviors that deviate significantly from general user activity.

Prediction	Count
1	213467
6	128301
3	69364
5	115499
9	99449
4	26198
8	69434
7	225477
10	24988
2	24090
0	3733

Figura 2.1: Cluster sizes based on predictions.

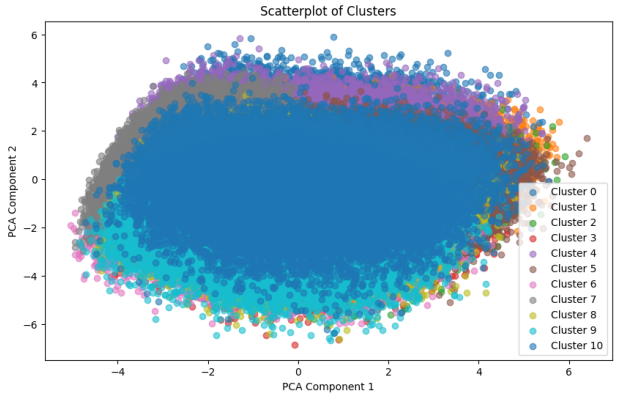

Figura 2.2: KMeans clustering results visualized using PCA.

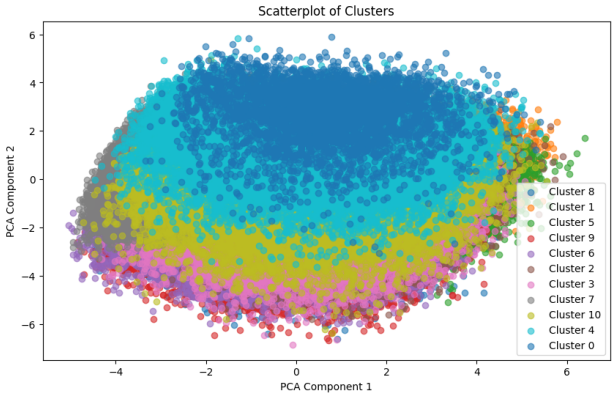

Figura 2.3: DBSCAN clustering results visualized using PCA.

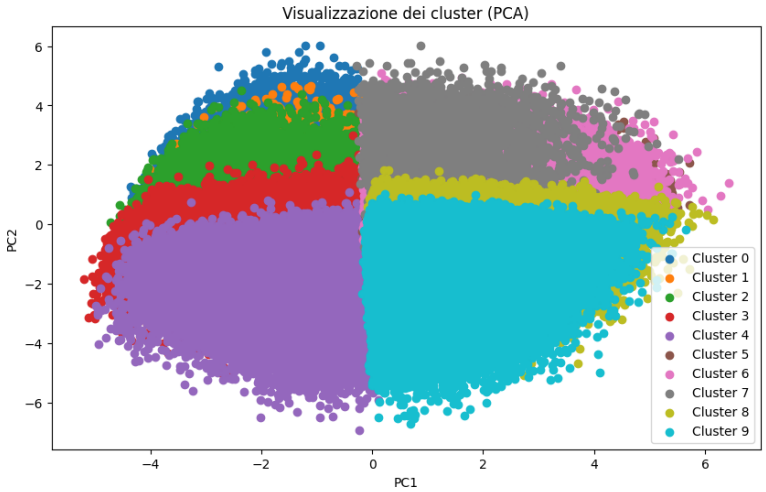

Figura 2.4: Bisecting KMeans clustering results visualized using PCA.

Capitolo 3

Classification

3.1 Gradient-Boosted Trees

We decided to apply Gradient-Boosted Tree in three different ways:

1 Using the base dataset;

2 Using a weighted dataset;

3 Using a random undersampled dataset.

We decided to compare all the models with a model trained on the plain dataset. To evaluate it's results we used the AUC score in combination with measure like Recall, Precision and F1-score.

3.1.1 Baseline Model

The baseline model, as expected, exhibited the poorest performance among all evaluated models. This outcome is attributed to the extreme class imbalance in the dataset, where the majority of samples belong to the negative class. The model's predictions are heavily biased towards the majority class, effectively ignoring the minority class. Figure 3.1 illustrates the confusion matrix for the baseline model. It clearly demonstrates that the model is overfitted to the negative class, leading to an inability to correctly identify positive samples (fraudulent transactions). The performance of the baseline model for the positive class, measured using standard classification metrics, highlights its inadequacy in handling imbalanced datasets: with a Recall of 0 and an F1 score of 0.01 we the model fails to detect any positive instances, demonstrating a complete inability to handle the minority class.

3.1.2 Undersampled Model

In this approach, we utilized a randomly undersampled dataset to address the class imbalance challenge. The undersampling process was performed randomly due to the absence of specialized undersampling techniques in PySpark. The model was trained on the undersampled training dataset and subsequently evaluated on the original, imbalanced test dataset to assess its generalizability.

The training dataset created through undersampling comprised approximately 17,732 instances, with 8,899 belonging to the negative class and 8,833 to the positive class.

The metrics for the positive class were disappointing as in the previous case, with a Recall of 0 and an F1 score of 0.01. The confusion matrix for the undersampled model is shown in Figure 3.2. The results clearly illustrate the model's significant bias towards the negative class. The model was trained using a grid search and cross-validation approach to optimize its hyperparameters.

3.1.3 Weighted Model

The weighted model yielded the most promising results. In this approach, we addressed the severe class imbalance by assigning a weight to the positive class that was 90 times higher than that of the negative class. This weighting strategy allowed the model to place greater emphasis on correctly predicting the minority (positive) class, reducing the bias toward the majority (negative) class.

The confusion matrix of the weighted model is shown in Figure 3.3. Key performance metrics of the model for the positive class were a Recall of 0.82, a precision of 0.05 and an f1 score of 0.09. The high recall indicates that the model effectively identifies the positive class. However, the precision is low due to the high number of false positives, which is expected when the model emphasizes detecting the minority class.

Also for this model we used grid search and cross validation to optimize it's hyperparameters.

3.2 Gradient-Boosted Trees Confusion Matrix

The confusion matrices for the baseline, undersampled, and weighted models presented together.

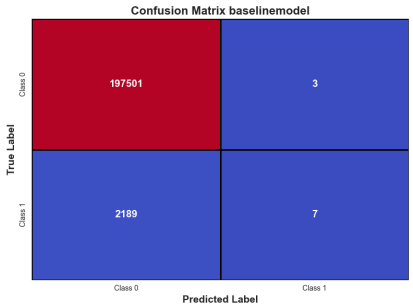

Figura 3.1: (a) Baseline Model

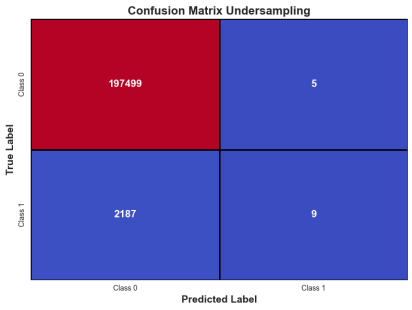

Figura 3.2: (b) Undersampled Model

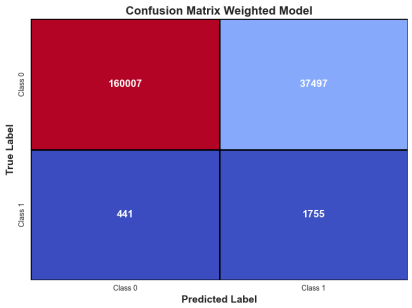

Figura 3.3: * (c) Weighted Model

Figura 3.4: Confusion Matrices of Gradient-Boosted Trees Models

3.2.1 Conclusions

In the end we analyze the ROC curve of the various models. We see that in the undersampled model while

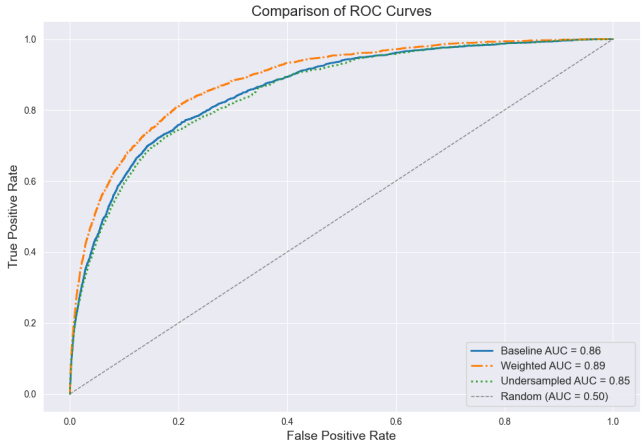

Figura 3.5: ROC curve comparsion

undersampling effectively balanced the training dataset, the model failed to generalize adequately, emphasizing the need for more sophisticated techniques to handle class imbalance effectively. We can see that the weighted model is the one with the bigger area under the curve, in our opinion the weighted model is an effective way to detect frauds, as we can see we get a lot of false positives 18.98% but we still manage to detect 79.91% of the frauds cases.

3.3 Random Forest

After applying Gradient-Boosted Trees we preceded with Random Forest. We performed with raw data, random undersampling, random oversampling and applying weights to rows based on class. We performed a GridSearch to find the best parameters possible. After a long run we ended up with numTrees: 150, maxDepth: 10, minInstancesPerNode: 1.

3.3.1 Raw data

Dealing with raw data results in high precision and recall for the majority class, while achieving zero for the minority class. Given the "blind" predictions made by the dummy classifier, which always predicts the

majority class with a probability of 1, the AUC is exactly 0.5. In contrast, Random Forest achieves an AUC of approximately 0.86, indicating that it attempts to infer patterns by varying the probability of predicting the majority class, even though it ultimately fails to classify the minority class effectively.

3.3.2 Random Undersampling

We applied random undersampling to reduce the majority class down to the size of the minority class, achieving a 1:1 ratio. As a result, the model was trained on 8,888 records from the majority class and 8,827 from the minority class. Nonetheless, the model achieved good recall for both the True and False classes. However, precision remained very poor for the minority class and misleadingly high for the majority class.

3.3.3 Random Oversampling

We performed random oversampling to balance the dataset by replicating the minority class to match the size of the majority class. A ratio of 89.66 was computed, representing the imbalance between the two classes. Using this ratio, the minority class was oversampled with replacement to achieve a nearly 1:1 balance. The resulting dataset contained 791,463 non-fraudulent and 791,109 fraudulent transactions.
With the significant increase in the size of the minority class, one might expect substantial improvements. However, the reality is that the minority class remains small, and even with a 1:1 ratio, there is no meaningful improvement. This is because the minority class instances are heavily duplicated, resulting in very low variance. Consequently, the results are comparable to those obtained with the previous ones.

3.3.4 Weighted Model

We decided to push further the previous weighted approach, we recalculated the weights based on the inverse proportionality of class distributions. This resulted in a weight of approximately 45.33 for the minority class and 0.51 for the majority class.
The updated weighting scheme aimed to penalize the misclassification of the minority class more heavily. While this adjustment led to slight improvements in recall for the minority class, precision remained low, and the overall performance showed only marginal gains. The AUC score reflected these limitations, indicating that class weighting alone is insufficient to fully address the extremely imbalance in the dataset.

Technique	Model	Precision (F,T)	Recall (F,T)	F1-Score (F,T)	AUC
Raw Data	Dummy Classifier	0.99, 0.00	1.00, 0.00	0.99, 0.00	0.5000
Raw Data	Random Forest	0.99, 0.00	1.00, 0.00	0.99, 0.00	0.8655
Random Undersampling	Random Forest	1.00, 0.04	0.81, 0.79	0.89, 0.08	0.8789
Random Oversampling	Random Forest	1.00, 0.05	0.84, 0.74	0.91, 0.09	0.8754
Weights	Random Forest	1.00, 0.05	0.86, 0.70	0.92, 0.10	0.8737

Tabella 3.1: Results of different techniques applied to unbalanced fraud detection dataset.

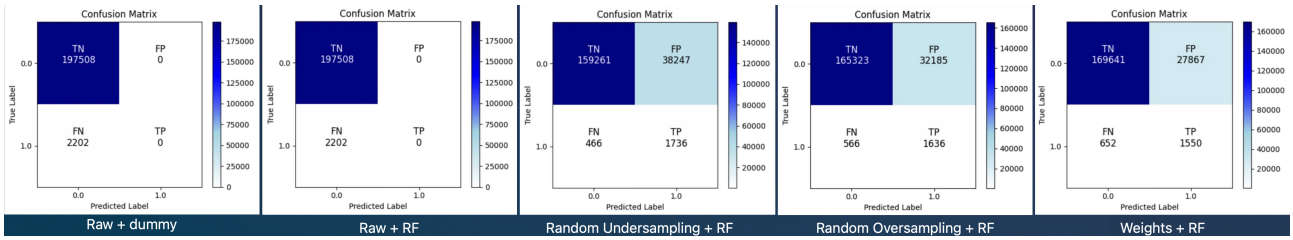

Figura 3.6: Confusion Matrices

3.3.5 Conclusions

Models trained on raw data, whether using a dummy classifier or Random Forest, fail entirely to detect the minority class, resulting in zero recall and a useless 100% precision on majority class.
With Random Undersampling or Random Oversampling, we observe a significant increase in recall for the minority class. However, this comes at the cost of a high number of false positives: 38,247 for Random Undersampling and 32,185 for Random Oversampling.
The use of class weights (Weights + RF) offers the most balanced approach. This method achieves an acceptable recall for the minority class while reducing false positives to 27,867, the lowest among the sampling-based

techniques.
Overall, in the field of fraud detection, the priority is to achieve high recall. It is generally preferable to flag more transactions as potentially fraudulent rather than risk missing actual fraud cases. However, in this scenario, the relatively high recall comes at the cost of a substantial number of false positives, which require manual verification and represent a significant operational expense.

3.4 Multilayer Perceptron

PySpark's Multilayer Perceptron (MLP) is not implemented to handle weighted datasets. Therefore, for this model, we evaluate only the baseline model and the random undersampled version.

We configure the MLP with four layers: the input layer consisting of 30 neurons, three hidden layers with 16, 8 and 5 neurons respectively, and an output layer with 2 neurons corresponding to the two classes. Unfortunately, PySpark's MLP lacks many customization options, such as alternative activation functions—ReLU is used for hidden layers and Softmax for the output layer. Additionally, features like early stopping are not supported.

3.4.1 Baseline MLP

We trained it plain on the preprocessed dataset already employed for the other models. We trained it with 100 iterations, a block size of 256 and seed 1234. As with the other models, the baseline MLP performs poorly due to the significant class imbalance. The evaluation metrics are summarized in the table below:

Class	Precision	Recall	F1-Score	Support
0.0	0.99	1.00	0.99	198,205
1.0	0.29	0.00	0.01	2,221

Tabella 3.2: Performance metrics for the baseline MLP model.

The results indicate that the baseline model struggles to correctly identify instances of the positive class (fraud). Intrestingly though, the precision for class 1 is higher with respect to the undersampled model.

3.4.2 Random Undersampled MLP

The data was split into 80% for training and 20% for testing, with undersampling applied exclusively to the training set. A grid search cross-validation with 3 folds was conducted to identify the optimal parameters. The best parameters identified were:

- **BlockSize:** 128 chosen from [128, 256, 512]
- **MaxIter:** 400 chosen from [100, 200, 300, 400]
- **Solver:** l-bfgs chosen from ['gd', 'l-bfgs']

To enhance performance, the model weights were initialized manually.
The undersampled version of the MLP produced more promising results. We experimented with majority-to-minority class ratios of 1:1 and 2:1, achieving best results with 1:1 ratio. The final evaluation metrics for the classifier are presented below:

Class	Precision	Recall	F1-Score	Support
0.0	1.00	0.81	0.89	198,205
1.0	0.04	0.78	0.08	2,221

Tabella 3.3: Performance metrics for the undersampled MLP model.

These results demonstrate that the undersampled version achieves a significant improvement in recall for the positive class at the expense of precision. This trade-off is further reflected in the Area Under the Curve (AUC) score, which stands at 0.79.

Capitolo 4

Explainability

4.1 GBT

As anticipated during our project presentation one of our final tasks was to try understand which features are the most importants in understanding the fraud and non-fraud cases, for that task we took GBT as standard model. The results show that the most important feature in deciding a fraud or non fraud case is device_os_encoded, we think that this might suggest that most fraud cases come from windows/mac terminals. Followed by phone_home_valid and has_other_cards; also housing_status_encoded is very significant, potentially correlating with financial stability. Another important measure is name_email_similarity, as already saw in the understanding phase, people with an high similarity between name and email tends to be the most vulnerable to fraud cases.

4.2 LIME

Another analysis that we did was with LIME: Local Interpretable Model-agnostic Explanations, to understand why the GBT model predict in that way. LIME explainer can be set up using two main steps: (1) import the lime module, and (2) fit the explainer using the training data and the targets.

4.2.1 Negative Class

We inspected a negative class and saw that features like device_distinct_emails_8w and keep_alive_session dominate in pulling the prediction towards the negative class. This could imply that patterns in email behavior and session durations are strong indicators of non-fraudulent activity (or the negative class in your case). It's also perfectly explainable because in typical day-to-day activity one user uses every day the same bank account and don't tend to have more than one in the same bank.

4.2.2 Positive Class

We can see that in this fraud transaction the income has the largest positive contribution to the model's prediction. Features like phone_home_valid and keep_alive_session also contribute to the fraud prediction, as expected from the previous example. We can also see that device_distinct_email is the main contributor as

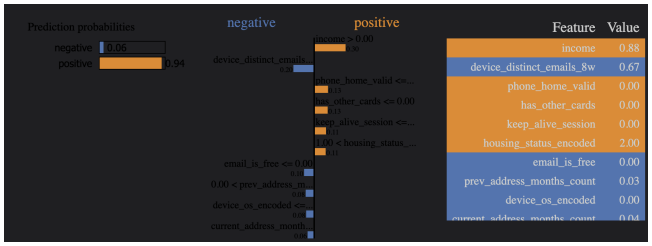

Figura 4.1: LIME Positive Class

negative impact.

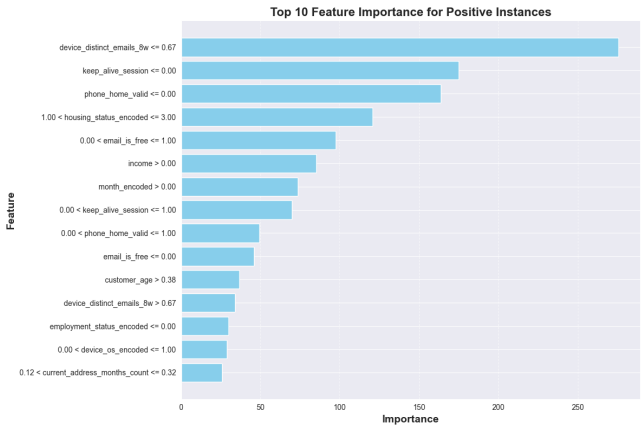

Figura 4.2: LIME feature importance

4.2.3 Sums on positive class

We used a for to sum the feature importances for the positive class: We saw that features like device_distinct_email and keep_alive_session are very important, that could be cause people in internet points are the most exposed to these kinds of risks.

4.2.4 Removing device related features

We removed the device related features from the dataset and trained GBT again to try to ask again LIME the feature importance.

Class	Precision	Recall	F1-Score
0.0	1.00	0.80	0.89
1.0	0.04	0.77	0.08

Tabella 4.1: Metrics for GBT after removing device related features

The model performance downgraded a bit, in particular in recall. But we can see that the lack of device related features impact on LIME explaination.

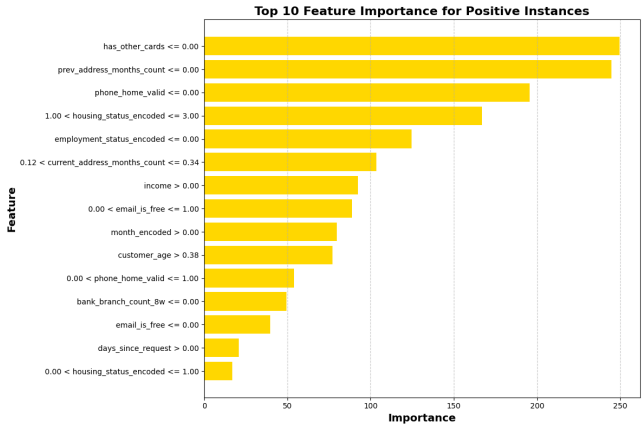

Figura 4.3: LIME feature importance second execution

We can see now that device-related features are removed new features rise up becoming the most important. The most important indicates that the fact that the customer does not have other cards impacts the fraud risk.