Università del Salento

Dipartimento di Ingegneria dell'Innovazione

Corso di Laurea Triennale in Ingegneria dell'Informazione

TESI DI LAUREA

Uno strumento di Transaction Risk Analysis basato su tecniche di machine learning a supporto dell'identificazione di frodi nei pagamenti digitali b2b

Relatore

Prof. Roberto Vergallo

Laureando

Roberto Vadacca

Matricola n° 20060531

ANNO ACCADEMICO 2021/2022

"Si vive di ricordi, | signori, e di giochi, | di abbracci sinceri, | di baci e di fuochi, | di tutti i momenti, | tristi e divertenti, | e non di momenti | tristemente divertenti.“ -Caparezza, Fuori dal tunnel

Grazie a chi c’è stato
nei momenti tristi
e divertenti!

Indice

Capitolo 1

Introduzione

1.1 Contesto

Le frodi sono una minaccia che la maggior parte dei fornitori di servizi online deve affrontare nello sviluppo dei propri sistemi per garantire security policy efficienti. La tendenza nel settore bancario è di provare a identificare possibili frodi nei pagamenti digitali, sia Business to Business (B2B) che Business to Customer (B2C) tramite tecniche di ML.

Le tecniche di Machine Learning (ML) e Deep Learning (DL) forniscono degli strumenti estremamente utili per una vasta gamma di applicazioni di uso comune. Tramite questi algoritmi e modelli è possibile processare immense quantità di dati la cui elaborazione consente di prendere decisioni in modo più veloce e più affidabile rispetto a metodi tradizionali. Sistemi di sicurezza bancaria di tipo statico, inoltre, non si adattano ai possibili cambiamenti di abitudine del cliente, sia esso customer o business. La pandemia, infatti, ha portato numerosi cambiamenti nella vita delle persone e nelle spese aziendali. Tramite il continuo flusso di dati in ingresso e in uscita dal sistema bancario è possibile creare un sistema di sicurezza che evolve insieme ai clienti.

Il ML si basa sulla progettazione di algoritmi che consentono a un computer di imparare. Il processo di apprendimento non coinvolge necessariamente una coscienza ma consiste nel trovare regolarità statistiche

o riconoscere schemi e andamenti nei dati. Infatti, molti algoritmi di ML non emulano il procedimento di apprendimento umano ed è per questo motivo che questi algoritmi individuano pattern che gli esseri umani difficilmente individuerebbero.

Una delle criticità nell'applicazione di algoritmi di ML nel settore bancario è la poca disponibilità da parte degli istituti finanziari e delle aziende specializzate nel bank equipment & service nel pubblicare i loro database. Data la difficoltà nel reperire questi dati di dominio, questi ultimi sono di notevole valore economico e di conseguenza le compagnie li custodiscono in vault virtuali. Primo tra gli ostacoli è l'effettiva raccolta dei dati, spesso disaggregati e dislocati in più uffici di competenza, e quindi in sistemi diversi; questa operazione già complessa dal punto di vista tecnico è resa ancora più complicata da policy di sicurezza e privacy restrittive. Per questo motivo un gran numero di strumenti di ML sono allenati su dataset pubblici, ossia casi ideali destinati a portare il sistema ad avere prestazioni differenti rispetto al caso reale.

1.2 Obiettivi

L'obiettivo della tesi è di rilevare anomalie in un dataset fornito da una banca popolare italiana composto da bonifici b2b applicando algoritmi di ML. I dati sono stati criptati alla sorgente per garantire l'anonimia dei clienti. Essendo il dataset sprovvisto di una label che distingua le frodi dai bonifici ordinari è scopo di questo lavoro illustrare e utilizzare possibili tecniche che possano risolvere questa criticità. Considerando che la maggior parte dei dati non dovrebbe essere fraudolento, vogliamo trovare le anomalie nei nostri dati, e.g. dati significativamente diversi dagli altri che chiameremo "outlier" ovvero valore anomalo. Se il dataset è abbastanza grande e se le frodi costituiscono la minoranza, ci aspettiamo che i bonifici anomali possano essere frodi, usi impropri o semplicemente casi molto rari.

Per esplicitare, dunque, le informazioni necessarie per identificare un anomalia verrà costruito un albero decisionale.

Gli obiettivi di questo studio saranno dunque:

- Studiare e pulire il dataset;
- Utilizzare algoritmi di ML per evidenziare anomalie nel dataset;
- Confrontare gli algoritmi utilizzati;
- Generare un albero decisionale;
- Scrivere una semplice interfaccia REST per valutare nuovi dataset.

Capitolo 2

Stato dell'arte

2.1 Review Scientifica

2.1.1 Tipi di tecniche di ML

Gli algoritmi di ML sono divisi principalmente in 4 categorie: Supervised learning, Unsupervised learning, Semi-supervised learning, Reinforcement learning.

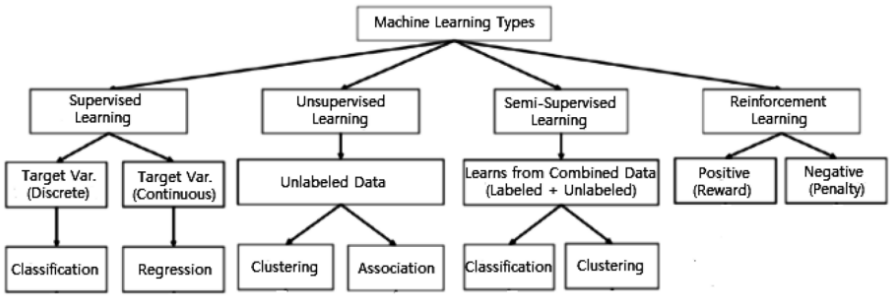

Figura 2.1: Albero delle categorie di ML; fonte:[1]

Algoritmi non supervisionati

L'apprendimento non supervisionato analizza dataset non etichettati senza la necessità di un intervento umano, come ad esempio nei data-driven process. È largamento usato per estrarre generative features, identificare trend e strutture, raggruppare elementi e a scopi esplorativi. Le task più comuni sono: clustering, density estimation, feature learning, dimen-

sionality reduction, trovare regole di associazione, anomaly detection, etc.

Algoritmi supervisionati

L'apprendimento supervisionato utilizza dataset con label per allenare un algoritmo nella classificazione e nella predizione accurata di dati. L'algoritmo utilizza un training set per insegnare al modello a produrre l'output desiderato. Il training set contiene l'input e l'output corretto, questo consente al modello di imparare nel corso del tempo. Questa categoria di algoritmi viene utilizzata per risolvere principalmente 2 tipi di problemi:

- Classification: dividere l'input in specifiche categorie.
- Regression: comprendere le relazioni tra varibili dipendenti e indipendenti.

Algoritmi semi-supervisionati

L'apprendimento semi-supervisionato utilizza dati con label insieme a dati sprovvisti di label in modo da migliorare la qualità dell'apprendimento. L'obiettivo è di poter dare una label ai dati che ne sono sprovvisti tramite le conoscenze apprese dai punti con label, che solitamente rappresentano la minoranza del dataset.

Algoritmi per rinforzo

L'apprendimento per rinforzo ha come obiettivo la costruzione di un agente che attraverso ai dati ottenuti da interazioni con l'ambiente incrementa le sue performance. La metrica che consente di misurare la qualità di un'azione è detta ricompensa, dove migliore il comportamento, maggiore è la ricompensa. in questo modo il sistema autonomo è indirizzato verso il comportamento desiderato per massimizzare la ricompensa.

Fasi del processo di ML

Il ciclo di vita del ML é delineato nelle 5 fasi sottostanti

1. **Raccolta dei dati:** Nella prima fase si raccolgo i dati da una o più fonti, interne o esterne al dominio del committente e si riuniscono in un uno o più dataset. Per la buona riuscita degli step successivi è necessario disporre di dati a sufficienza contenenti informazioni utili a risolvere il problema.

2. **Preparazione dei dati e ingegneria delle funzionalità:** Dopo aver raccolto i dati è fondamentale studiarli e comprenderli anche con l'ausilio della loro visualizzazione. Per ottimizzare la procedura di apprendimento occorre effettuare una pulizia del dataset, rimuovendo variabili poco utili, valori mancanti ed errori umani; migliore la qualità dei dati, migliori le performance. Nel caso in cui alcune colonne presentino valori di tipo categorico talvolta è conveniente convertirle in valori di tipo numerico. Dai dati grezzi sarà possibile estrarre caratteristiche, proprietà e attributi del caso di studio da utilizzare durante l'addestramento del modello e l'inferenza per effettuare previsioni.

3. **Scelta e realizzazione del modello ML:** La scelta del modello dipende principalmente dall'obiettivo, dall'input/output desiderato, complessità, scalabilità e interpretabilità. L'allenamento e la creazione del modello dipendono dalla tipologia dell'algoritmo di ML selezionato.

4. **Valutazione del modello:** Si scelgono una o più metriche con cui è possibile quantificare la qualità del modello allo scopo di poter effettuare la messa a punto dei parametri dell'algoritmo e per poterlo confrontare con altri.

5. **Distribuzione del modello:** Nella fase finale viene rilasciato il modello nell'ambiente di produzione. Nuovi dati verranno utiliz-

zati per il continuo apprendimento del modello e per monitorare le sue prestazioni. È opportuno creare una dashboard utilizzando strumenti di visualizzazione per supportare le decisioni di business del cliente.

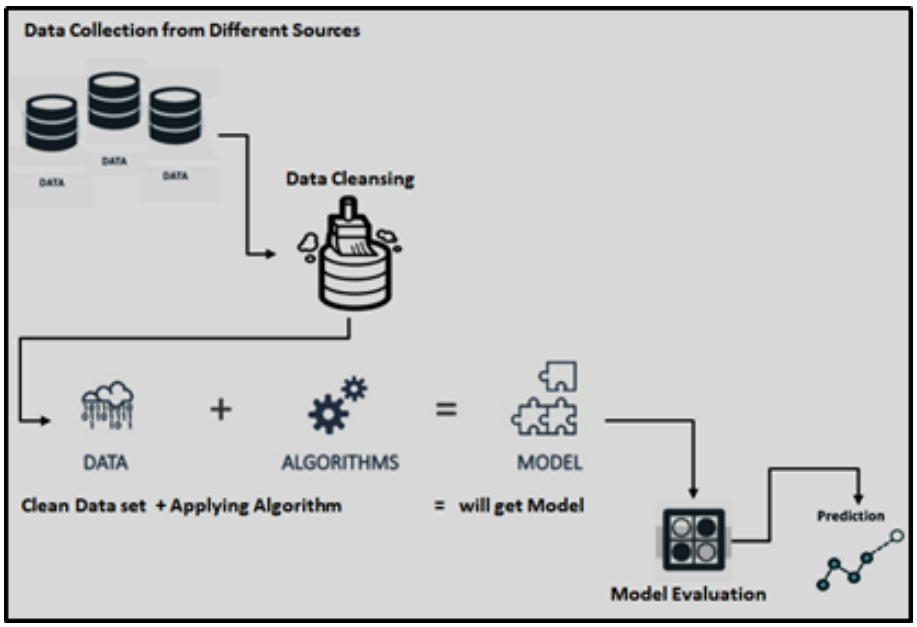

Figura 2.2: Workflow del processo ML

2.1.2 Anomaly Detection

La ricerca di anomalie all'interno di un dataset attraverso algoritmi di ML e reti neurali è oggetto di studio da diversi anni. In paper più recenti come [2] e [3] vengono descritti e confrontanti gli algoritmi più performanti. Da queste review datate 2022 si nota come nel caso non supervisionato non esista un algoritmo più performante in generale, ma vi è una forte dipendenza dal dataset [2]. Per quanto riguarda le financial fraud detection nello specifico, in [3] lo studio si concentra nell'esporre vantaggi e svantaggi di particolari modelli in base alle limitazioni che lo studio di un particolare tipo di frode può comportare. Sono state anche trattate architetture di DL come: convolutional neural networks (CNN), autoencoders (AE) and generative adversarial networks (GAN). Le reti neurali si dimostrano uno strumento molto accurato a costo di una serie

di svantaggi quali tuning e design molto complessi e alta sensibilità nella scelta di hyperparameters e architecture structure. Una volta rilevate le anomalie ci si interroga sulla causa di esse.
Identificare le variabili che contribuiscono all'etichettare un dato come anomalo possono aiutare gli esperti di dominio ad effettuare verifiche e a risolvere le criticità eventualmente riscontrate. Etichettando il dataset con le anomalie rilevate è possibile utilizzare algoritmi di ML supervisionato per costruire un classification and regression tree al fine di rilevare i tipi di anomalia e quali variabili la rendono tale [4]

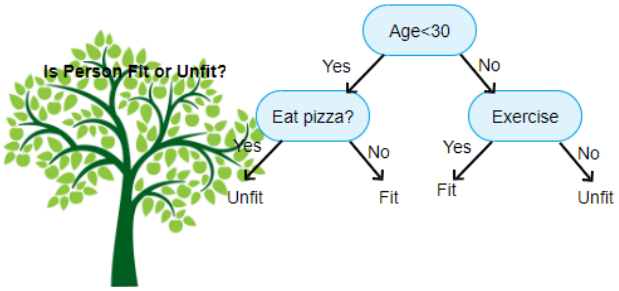

Figura 2.3: Semplice esempio di Classification and Regression Tree
fonte:[5]

2.2 Review Tecnologica

L'utilizzo di Python e le sue librerie nell'ambito del ML è ben documentato in letteratura [6] e [7], citiamo a titolo di esempio le librerie con il maggior numero di contributors: Scikit-learn, framework per hardware non specializzato; Tensorflow, Pytorch, e Keras con supporto alle GPU; Apache-Spark con MapReduce. Data la vasta scelta di algoritmi implementabili è utile consultare dei benchmark [2] basati sul calcolo dell'AUCROC, ovvero "Area under the ROC curve", dove la curva ROC è un grafico che mostra le prestazioni di un modello di classificazione in tutte le soglie di classificazione sulla base della percentuale di veri positivi e falsi positivi. Il valore dell'AUC è compreso tra 0 e 1. Un modello le cui previsioni sono errate al 100% ha un AUC di 0,0; uno le cui previsioni

sono corrette al 100% ha un AUC di 1,0. IForest, CBLOF, PCA, ECOD, COPOD sono i 5 algoritmi analizzati con l'AUCROC medio più alto, questo non significa che altri algoritmi non vadano presi in considerazione in quanto nessun algortimo non supervisionato nello studio surclassa altri a livello statistico. Il ranking distingue anche i diversi tipi di anomalia: LOF per le local anomalies, KNN per le global e dependency anomalies, OCSVM per le clustered anomalies. Per quanto riguarda gli algoritmi supervisionati si nota che la maggior parte superano in performance il metodo non supervisionato, citiamo CatB e FTTransformer per via delle performance documentate. Nonostante il calcolo delle performance sia sicuramente un punto fondamentale nella scelta di un algoritmo vi è da tenere in considerazione anche il tipo di parametri che esso richiede e più in generale la difficoltà d'implementazione. Non è un caso, infatti, che algoritmi molto utilizzati siano ad esempio Gaussian-Mixture [8], K-means [9] e Naive Bayes [10].

2.2.1 Algoritmi

Extended Isolation Forest

Extended Isolation Forest è un algoritmo di rilevamento anomalie, 'isola' i punti selezionando una feature in modo casuale e successivamente selezionando casualmente un punto di split tra il valore massimo e il valore minimo della feature selezionata. Questo partizionamento ricorsivo può essere rappresentato da una struttura ad albero, dove il numero di split richiesto per isolare un campione è uguale alla lunghezza del percorso dalla root al nodo terminale. Tale lunghezza è la misura della "normalità" di un campione e dunque fondamentale per rilevare le anomalie. Più è corta la lunghezza, più il campione è anomalo.

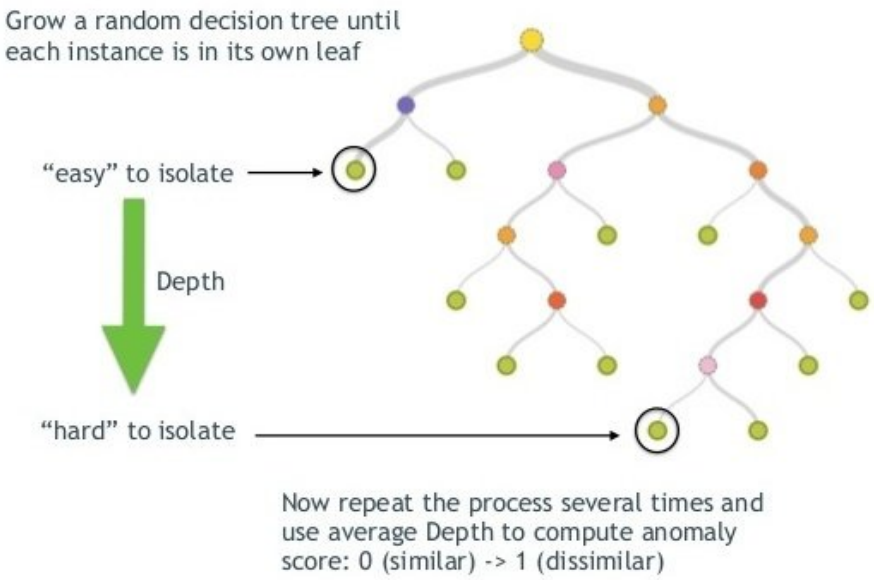

Figura 2.4: Rappresentazione ad alto livello del funzionamento di Isolation Forest; fonte:[11]

Algorithm 3 $PathLength(\vec{x}, T, e)$

Input: $\vec{x}$ - an instance, T - an iTree, e - current path length; to be initialized to zero when first called
Output: path length of $\vec{x}$

1: **if** T is an external node **then**
2: **return** $e+c(T.size)\{c(.)$ is defined in Equation (2)$\}$
3: **end if**
4: $\vec{n} \leftarrow T.Normal$
5: $\vec{p} \leftarrow T.Intercept$
6: **if** $(\vec{x} - \vec{p}) \cdot \vec{n} \leq 0$ **then**
7: **return** $PathLength(\vec{x}, T.left, e + 1)$
8: **else if** $(\vec{x} - \vec{p}) \cdot \vec{n} > 0$ **then**
9: **return** $PathLength(\vec{x}, T.rigth, e + 1)$
10: **end if**

Figura 2.5: Algoritmo per calcolare il path lenght; fonte:[12]

Empirical-Cumulative-distribution-based Outlier Detection

Empirical-Cumulative-distribution-based Outlier Detection, abbreviato ECOD, è un algoritmo di outlier detection di tipo probabilistico pubblicato nel 2022 basato su funzione di ripartizione empirica. I vantaggi nell'utilizzo di questo algoritmo sono:

- No hyperparameters.
- Fast and computationally efficient.
- Easy to understand and interpretable.

L'idea alla base dell'algoritmo è che gli outliers sono punti che si trovano in zone a bassa densità della distribuzione di probabilità. Se la distribuzione è unimodale, allora gli eventi rari sono localizzati nella coda della distribuzione

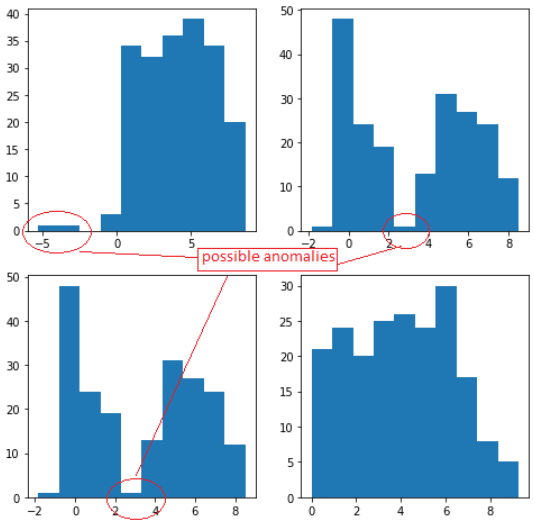

Figura 2.6: Quattro distribuzioni generate casualmente. Tre degli esempi hanno bin di valore basso. Punti appartententi in questi bins possono essere considerati outliers
fonte:[13]

Algorithm 1 Unsupervised OD Using ECDF (ECOD)

Inputs: input data $\mathbf{X} = \{X_i\}_{i=1}^n \in \mathbb{R}^{n \times d}$ with n samples and d features; $X_i^{(j)}$ refers to the value of j-th feature of the i-th sample

Outputs: outlier scores $\mathbf{O} := \text{ECOD}(\mathbf{X}) \in \mathbb{R}^n$

1: **for** each dimension j in $1, \ldots, d$ **do**

2: Estimate left and right tail ECDFs (using equations (1) and 2, which we reproduce below):

$$\text{left tail ECDF: } \widehat{F}_{\text{left}}^{(j)}(z) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{X_i^{(j)} \leq z\} \text{ for } z \in \mathbb{R},$$

$$\text{right tail ECDF: } \widehat{F}_{\text{right}}^{(j)}(z) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{X_i^{(j)} \geq z\} \text{ for } z \in \mathbb{R}.$$

3: Compute the sample skewness coefficient for the j-th feature's distribution:

$$\gamma_j = \frac{\frac{1}{n}\sum_{i=1}^{n}(X_i^{(j)} - \overline{X^{(j)}})^3}{\left[\frac{1}{n-1}\sum_{i=1}^{n}(X_i^{(j)} - \overline{X^{(j)}})^2\right]^{3/2}},$$

where $\overline{X^{(j)}} = \frac{1}{n}\sum_{i=1}^{n} X_i^{(j)}$ is the sample mean of the j-th feature.

4: **end for**

5: **for** each sample i in $1, \ldots, n$ **do**

6: Aggregate tail probabilities of X_i to obtain outlier score O_i: // §3.2.2

$$O_{\text{left-only}}(X_i) = -\sum_{j=1}^{d} \log(\widehat{F}_{\text{left}}^{(j)}(X_i^{(j)})),$$

$$O_{\text{right-only}}(X_i) = -\sum_{j=1}^{d} \log(\widehat{F}_{\text{right}}^{(j)}(X_i^{(j)})),$$

$$O_{\text{auto}}(X_i) = -\sum_{j=1}^{d}[\mathbb{1}\{\gamma_j < 0\}\log(\widehat{F}_{\text{left}}^{(j)}(X_i^{(j)})) + \mathbb{1}\{\gamma_j \geq 0\}\log(\widehat{F}_{\text{right}}^{(j)}(X_i^{(j)}))].$$

7: Set the final outlier score for point X_i to be

$$O_i = \max\{O_{\text{left-only}}(X_i), O_{\text{right-only}}(X_i), O_{\text{auto}}(X_i)\}.$$

8: **end for**

9: **Return** outlier scores $\mathbf{O} = (O_1, \ldots, O_n)$

Figura 2.7: Algoritmo ECOD; fonte:[14]

Local Outlier Factor

Local Outlier Factor è un metodo di rilevamento delle anomalie non supervisionato che calcola la local density deviation di un punto rispetto ai suoi neighbors. Vengono considerati anomali i punti che hanno una densità notevolmente inferiore rispetto a quella dei neighbors.

Algorithm 2 $LOFk, m, D$

Input: k - number of near neighbor, m - number of outliers, D - outlier candidate dataset.
Output: $topm$ outliers.

1: **for** $j = 1$ to $lenD$ do **do**
2: compute k - $distp$
3: compute N_kp
4: **end for**
5: calculate $reach$ - $dist_kp, r$ and $lrdp$
6: calculate $lofp$
7: sort the lof values of all points in descending order
8: **return** the m data objects with the large lof values, which are the outliers

Figura 2.8: Algoritmo LOF; fonte:[15]

Decision Forest

Un albero decisionale è un modello predittivo espresso come partizione ricorsiva dello spazio delle covariate in sottospazi che costituiscono una base per la predizione. Una foresta decisionale è quindi un insieme di alberi dei quali vengono combinate le predizioni. In questo modo, l'errore di un singolo albero viene compensato dal resto della foresta.

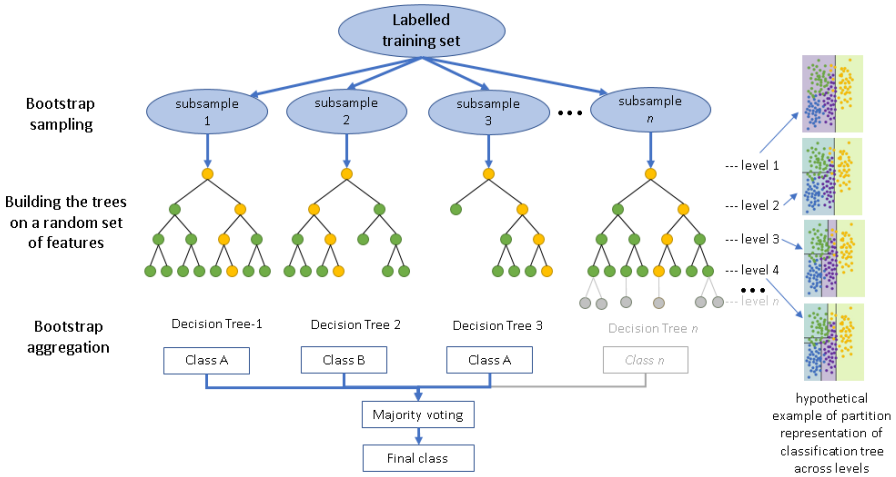

Figura 2.9: Rappresentazione algoritmo Decision Forest

Synthetic Minority Over-sampling Technique

Synthetic Minority Over-sampling Technique (SMOTE) è una tecnica statistica per bilanciare il dataset generando nuove istanze da casi di minoranza esistenti forniti come input.

Algorithm 1 SMOTE algorithm

1: **function** SMOTE(T, N, k)
   Input: T; N; k ▷ #minority class examples, Amount of oversampling, #nearest neighbors
   Output: $(N/100)$ * T synthetic minority class samples
   Variables: *Sample*[][]: array for original minority class samples;
   newindex: keeps a count of number of synthetic samples generated, initialized to 0;
   Synthetic[][]: array for synthetic samples
2: **if** $N < 100$ **then**
3: Randomize the T minority class samples
4: $T = (N/100)*T$
5: $N = 100$
6: **end if**
7: $N = (int)N/100$ ▷ The amount of SMOTE is assumed to be in integral multiples of 100.
8: **for** $i = 1$ to T **do**
9: Compute k nearest neighbors for i, and save the indices in the *nnarray*
10: POPULATE($N, i, nnarray$)
11: **end for**
12: **end function**

Figura 2.10: Algoritmo SMOTE; fonte:[16]

KNeighborsClassifier

KNeighborsClassifier è un classificatore di apprendimento supervisionato non parametrico, che utilizza la prossimità per effettuare classificazioni o previsioni di punti. L'idea alla base del funzionamento presuppone che punti simili possono essere trovati l'uno vicino all'altro. Per questo motivo l'etichetta di un punto viene assegnata tramite un voto pluralistico da parte dei K punti più vicini.

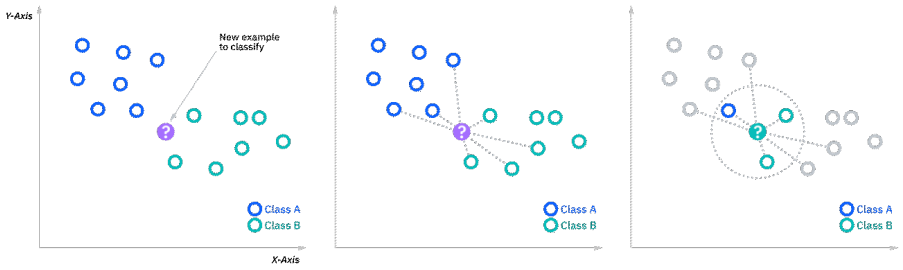

Figura 2.11: Voto pluralistico con K=3; fonte:[17]

Naive Bayes

Naive Bayes è un algoritmo per risolvere problemi di classificazione e apprendimento automatico che utilizza il teorema di Bayes. Il termine "naive" indica l'assunzione della condizione d'indipendenza tra tutte variabili indipendenti. La semplicità dell'algoritmo lo rende meno sensibile a problemi di alta dimensionalità a discapito di perdere informazione sulla correlazione.

Algoritmo L'algoritmo si basa sul teorema di Bayes; data la variabile y e il vettore di variabili indipendenti da x_1 a x_n:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{2.1}$$

con la condizione d'indipendenza:

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$$

quindi l'equazione 2.1 diventa:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Essendo $P(x_1, \ldots, x_n)$ costante dato l'input, possiamo usare la seguente regola di classificazione:

$$\begin{gathered} P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \\ \Downarrow \\ \hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y), \end{gathered} \tag{2.2}$$

possiamo usare una stima del massimo a posteriori per $P(y)$ e $P(x_i \mid y)$; quest'ultima è la frequenza relativa della classe y del training set.

Nel caso di Categorical Naive Bayes la probabilità della categoria t della feature i data la classe c è stimata come:

$$P(x_i = t \mid y = c\,;\, \alpha) = \frac{N_{tic} + \alpha}{N_c + \alpha n_i},$$

dove $N_{tic} = |\{j \in J \mid x_{ij} = t, y_j = c\}|$ è il numero di volte in cui la categoria t è presente nei samples x_i che appartengono alla classe c, $N_c = |\{j \in J \mid y_j = c\}|$ è il numero di samples di classe c, α è un parametro di lisciamento e n_i è il numero di categorie disponibili delle feautures i.

Capitolo 3

Studio di Fattibilità

Lo studio di fattibilità si occupa di valutare la fattibilità di un prodotto o un sistema dal punto di vista tecnico, operazionale, economico, legale e di schedule. In questo capitolo verrà affrontato lo studio dal punto di vista tecnologico.

3.1 Architettura Logica

In questa sezione approfondiremo la struttura del sistema in cui verrà collocata la soluzione proposta per supportare l'identificazione delle frodi. Grazie ai dati provvisti dalla banca si vuole creare un modello grazie ad algoritmi di ML supervisionati e non supervisionati. Successivamente sarà possibile fare inferenza sul suddetto modello in modo da distinguere le anomalie dai casi standard. Verrà quindi popolato un database costituito dalle anomalie rilevate così da dare la possibilità ad un operatore esperto del dominio di poter etichettare definitivamente il bonifico. L'interazione tra l'operatore e il database sarà agevolata grazie ad una dashboard.

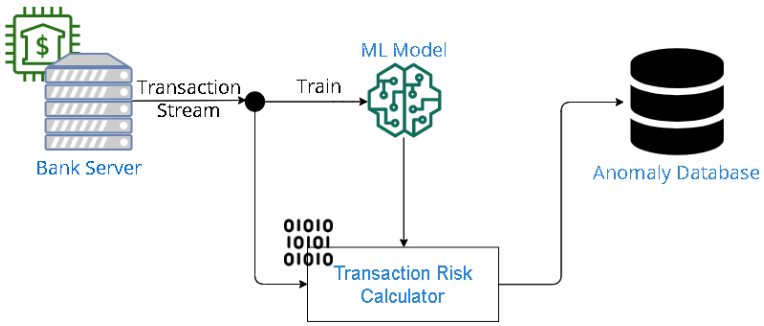

Figura 3.1: Architettura logica Transaction Risk Tool

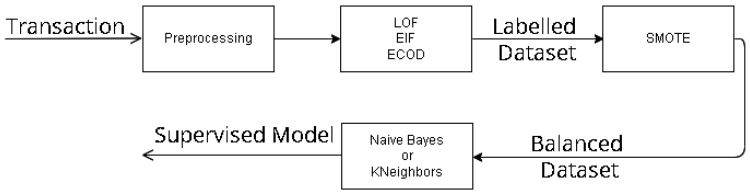

Figura 3.2: Pipeline che descrive il processo di creazione del modello supervisionato

In figura 3.2 viene illustrato il processo di creazione del modello supervisionato. Nel preprocessing verranno effettuate le operazioni di pulizia, conversione dei dati e label encoding per sostituire dati categorici con dati numerici. In questo modo sarà possibile utilizzare algoritmi di ML non supervisionato, in particolare LOF, EIF e ECOD per etichettare e quindi distinguere le frodi dai bonifici ordinari. Per via della scarsa quantità di anomalie sarà necessario utilizzare l'algoritmo SMOTE per bilanciare il dataset allo scopo di allenare i modelli supervisionati Naive Bayes e KNearestNeighbor. L'utilizzo combinato di algoritmi supervisionati a quelli non supervisionati è stato suggerito dalla lettura di "Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection [18] e e "Unsupervised Anomaly Detection Based on Minimum Spanning Tree Approximated Distance Measures and its Application to Hydropower Turbines" [4].

3.2 Scelte Tecnologiche

3.2.1 Software

È stato scelto Python come linguaggio di programmazione per la semplicità con cui è possibile integrare moduli di data analysis e ML.

Librerie

Le librerie utilizzate sono:

- Pandas: strumento di manipolazione e analisi dati veloce, potente, flessibile e facile da usare [19].
- Scikit-learn: libreria open source di ML per Python. Contiene algoritmi di classificazione, regressione e clustering. [20].
- Flask: un micro web framework scritto in Python [21].
- Pickle: implementa un algoritmo per trasformare un oggetto arbitrario Python in un una serie di byte [22].
- Numpy: un progetto open source per calcolo numerico [23].
- PyOD: libreria scalabile e completa per rilevare anomalie in dati multivariati [24].
- Tensorflow: Una piattaforma open source per ML specializzata nella risoluzione di compiti percettivi e di comprensione del linguaggio [25].
- Imbalanced Learn:collezione di metodi per under-sampling e over-sampling compatibile con scikit-learn [26].

Algoritmi

Gli algoritmi sono stati selezionati in base alle loro performance [2], alla facilità di implementazione e alla disponibilità nelle librerie sopracitate.

Unsupervised Learning

- LOF: disponibile su scikit-learn, utile nel rilevamento di Local e Dependency anomalies
- EIF: disponibile su GitHub, utile nel rilevamento di Global e Clustered anomalies
- ECOD: disponibile su PyOD, good overall
- Kmeans e Gaussian Mixture: disponibile su scikit-learn, pochi parametri e semplice interpretazione statistica

Supervised Learning KNN e Naive Bayes disponibili su scikit learn e di facile interpretazione.

Incompatibilità riscontrate in Kmeans e Gaussian Mixture Da un'analisi preliminare sono state riscontrate sia con Kmeans che con Gaussian Mixture criticità che ne hanno reso sconsigliato l'utilizzo per questo specifico studio:

Silhouette coefficient Il Silhouette coefficient è calcolato usando la distanza tra cluster media (a) e la distanza nearest-cluster (b) per ogni punto. Il Silhouette coefficient per un punto è $b-a/max(a,b)$. Il miglior valore è 1 e il peggior valore è -1. Valori prossimi a 0 indicano cluster sovrapposti. Nel nostro caso per entrambi gli algoritmi il Silhouette score è prossimo a 0 con il massimo valore ottenuto in entrambi i casi per 2 cluster:

- GMM: silhouette_score medio = 0.0247
- K-Means: silhouette_score medio = 0.2384

IDE

Gli IDE utilizzati sono:

- Jetbrains Dataspell: IDE dedicata ai data scientist, utile per data analysis. Supporta file di tipo .ipynb (IPython Notebook), un ambiente computazionale interattivo in cui è possibile combinare esecuzione di codice, elementi testuali, formule matematiche e file multimediali.
- Jetbrains Pycharm: IDE per python.
- Google Colab: IDE web hostata da server Google, utile per la possibilità di fare code sharing, supporta .ipynb.

3.2.2 Hardware

Lo studio verrà effettuato su:

- CPU: Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz.
- RAM: 12,0 GB DDR3 1333MHz.

3.3 Architettura Fisica

In questa sezione approfondiremo la struttura del Server responsabile dell'identificazione delle frodi. Il sistema operativo scelto è GNU/Linux, in particolare la distribuzione Ubuntu Server è specializzata nell'ambito server. L'intero codice sarà scritto in Python versione 3.10.7 e verranno utilizzate le librerie di ML disponibili su scikit-Learn, Pyod e alcune repository Github; tutte le librerie di cui faremo uso non necessitano di hardware specilizzato come GPU ad alte prestazioni, TPU o NPU. I componenti principali saranno il modello finale, da costruire grazie ai 3 algoritmi di apprendimento non supervisionato ECOD, LOF ed EIF e all'algoritmo di apprendimento supervisionato Naive Bayes o KNearest Neighbour, e l'encoder dictionary responsabile della conversione di dati categorici in dati numerici. A supporto dell'algoritmo supervisionato vi sarà un algoritmo di Synthetic Over-Sampling "SMOTE" utile per il bilanciamento del dataset. Il servizio verrà reso disponibile all'esterno grazie a un'interfaccia erogata da python mediante Flask utilizzando il protocollo HTTPS.

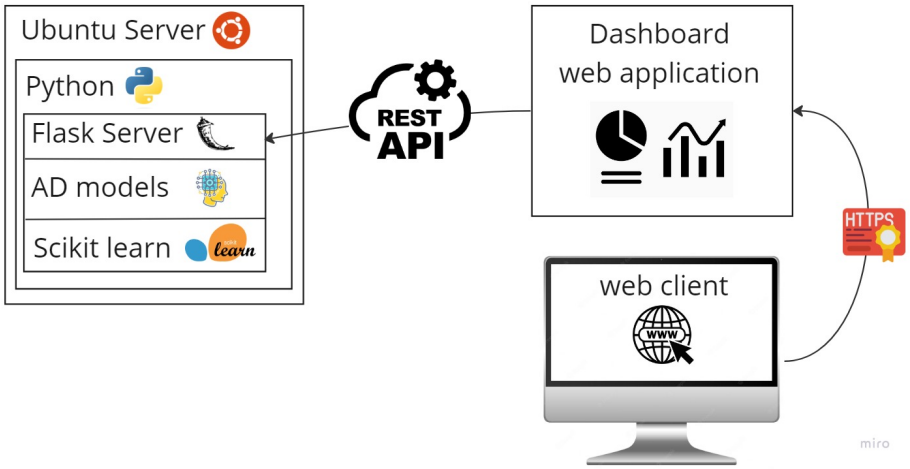

Figura 3.3: Architettura Fisica Transaction Risk Server

Capitolo 4

Progettazione

In questo capitolo verrano descritte le funzionalità del codice attraverso l'uso di diagrammi UML (Unified Modeling Language), nello specifico:

- Diagramma dei casi d'uso, per descrivere le funzionalità del sistema.
- Diagramma di sequenza, per una visione dinamica in cui si mostra l'evoluzione delle parti del sistema nel tempo.

4.1 Diagramma dei casi d'uso

In UML, gli Use Case Diagram sono diagrammi dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Sono impiegati soprattutto nel contesto della Use Case View (vista dei casi d'uso) di un modello, e in tal caso si possono considerare come uno strumento di rappresentazione dei requisiti funzionali di un sistema. Di seguito, in 4.1 verrà riportato il diagramma dei casi d'uso.

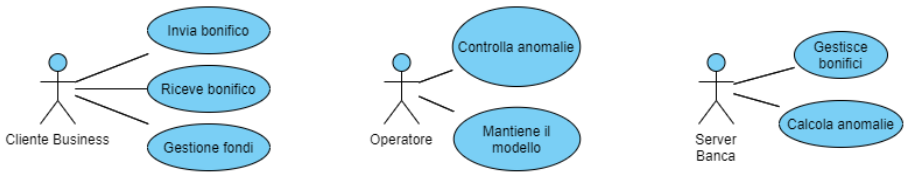

Figura 4.1: Use case diagram

4.2 Diagramma di sequenza

I diagrammi d'interazione descrivono la collaborazione di un gruppo di oggetti che devono implementare collettivamente un certo comportamento analizzando l'interazione tra le varie componenti in esame, il passaggio di parametri e il ritorno di dati. UML definisce diverse forme di diagrammi d'interazione, ma la più usata è il diagramma di sequenza. I diagrammi di sequenza forniscono la rappresentazione dei vari scenari che si possono verificare nel ciclo di vita del sistema. Di seguito verranno riportati i diagrammi di sequenza relativi ai casi d'uso esaminati.

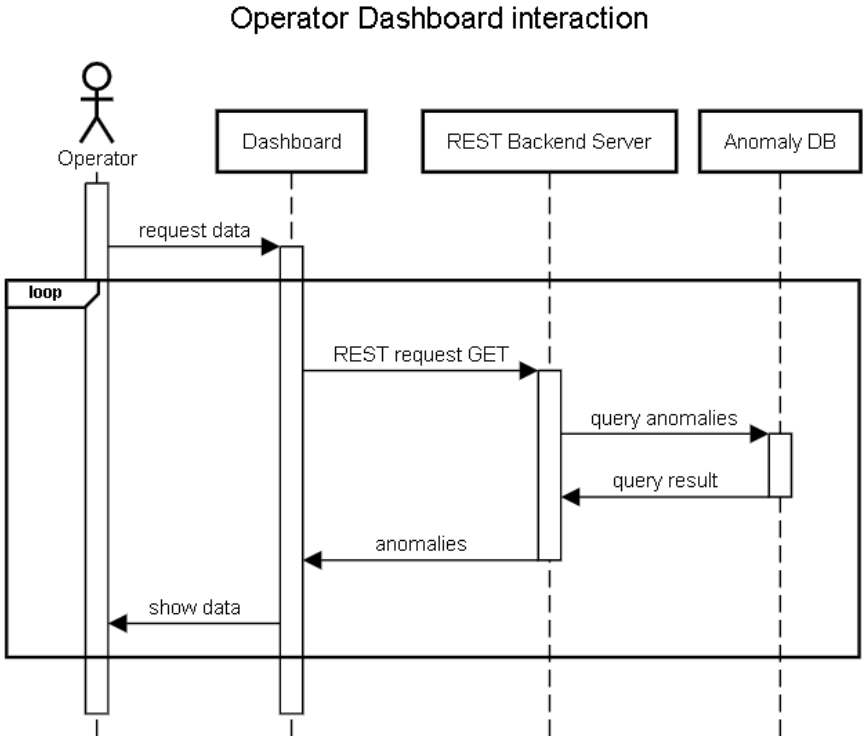

Figura 4.2: Sequence Diagram che descrive l'interazione tra l'operatore, la dashboard e il backend

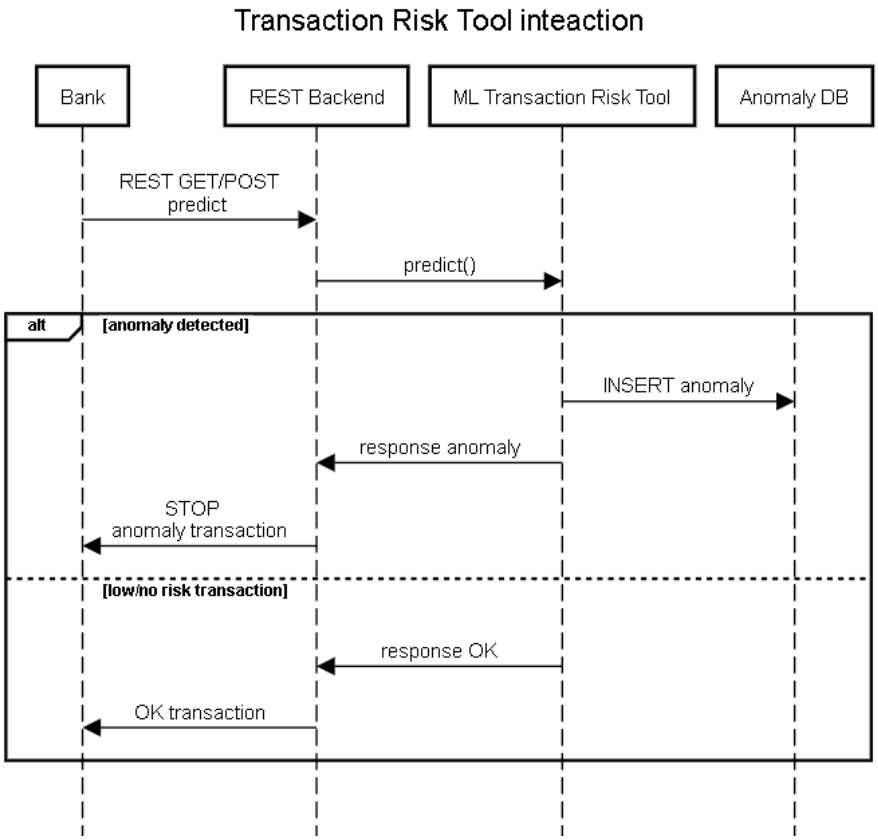

Figura 4.3: Sequence Diagram che descrive il processo d'inferenza e popolamento dell'anomalyDB

Capitolo 5

Sviluppo

5.1 Presentazione Dataset

Una banca popolare italiana è interessata nel rilevare bonifici fraudolenti e ha fornito per questo studio un dataset in formato .csv composto da 317287 righe e 25 colonne. Allo scopo di analizzarlo utilizzeremo il modulo pandas.

```
df.info()
```

```
Data columns (total 25 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   customerId               317287 non-null  int64
 1   timestamp                317287 non-null  object
 2   bankCode                 317287 non-null  int64
 3   country                  317269 non-null  object
 4   amount                   317287 non-null  float64
 5   businessName             317287 non-null  object
 6   loginMode                317287 non-null  object
 7   isFraud                  317287 non-null  int64
 8   userAgent_isMobile       317287 non-null  int64
 9   userAgent_isTablet       317287 non-null  int64
 10  userAgent_isPc           317287 non-null  int64
 11  userAgent_isTouch        317287 non-null  int64
 12  userAgent_isBot          317287 non-null  int64
 13  userAgent_browserFamily  317287 non-null  object
 14  userAgent_osFamily       317287 non-null  object
 15  IBAN_countryCode         317287 non-null  object
 16  IBAN_bankCode            316912 non-null  object
 17  BIC_branchCode           110720 non-null  object
 18  BIC_countryCode          113647 non-null  object
 19  BIC_locationCode         113647 non-null  object
 20  SIA                      317287 non-null  object
 21  CAP                      116346 non-null  object
 22  SAE                      117409 non-null  float64
 23  RAE                      117409 non-null  float64
 24  companyType              117409 non-null  object
```

Dalla colonna non-null si può notare come il dataset sia incompleto e necessiti di essere pulito per essere utilizzato come input per generare il modello.

5.2 Pulizia Dataset

In questa sezione viene mostrato il codice utilizzato per la pulizia del dataset: sono state eliminate colonne deprecate, non utili o incomplete, sono state eliminate le tuple con valori nulli, sono state effettuate conversioni su alcuni tipi di dati.

```
#PULIZIA DATASET E RIMOZIONE COLONNE DEPRECATE

def data_cleaning(df):
  df = df.drop(columns=['loginMode', 'isFraud', 'bankCode']) #deprecati
  df = df.drop(columns=['BIC_locationCode','BIC_branchCode','BIC_countryCode']) #incompleti
  df = df.drop(columns=['userAgent_isMobile','userAgent_isTablet','userAgent_isTouch']) #sono 99% pc
  df = df.drop(df[df.companyType=='****'].index)
  df = df.dropna()

  df = df[pd.to_numeric(df['CAP'], errors='coerce').notnull()] #solo cap numerici, filtra 'MONTE'
  df.CAP=df.CAP.astype(int)

  df.SAE = df.SAE.astype(int) #era float

  df.RAE = df.RAE.astype(int) #era float

  df['IBAN_bankCode'] = df['IBAN_bankCode'].astype('str')

  return df

df=data_cleaning(df)
```

```
df.shape
```

```
(90960, 15)
```

Dopo il processo di pulizia dei dati sono rimaste 90960 righe e 15 colonne

5.3 Preprocessing

Gli algoritmi di apprendimento non supervisionato che andremo ad utilizzare non sono in grado di effettuare operazioni su dati categorici. Per questo motivo è necessario utilizzare un LabelEncoder in modo da trasformare stringhe in numeri interi. Salveremo le label ottenuto in un defaultdict, in modo da poter riconvertire i dati da numeri interi a stringhe.

```
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict #per creare un dizionario di label

columns_to_label = ['country','userAgent_browserFamily','userAgent_osFamily','IBAN_countryCode',
                    'IBAN_bankCode','CAP','SIA','SAE','RAE','companyType']


#fit+transform e aggiornamento del dizionario, restituisce il df labellato
def fit_transform_dict (df,enc_dict,columns):
  labeled=df.copy()
  for clmn in columns:
    enc_dict[clmn].fit(labeled[clmn])
    labeled[clmn]=enc_dict[clmn].transform(labeled[clmn])
  return labeled

labeled_df_SRL=fit_transform_dict(df_SRL,encoder_dict,columns_to_label)

labeled_df_SRL.head()
```

5.4 Unsupervised Anomaly Detection

5.4.1 Extended Isolation Forest

Implementazione

L'algoritmo e la documentazione è disponibile nella repository di github `https://github.com/sahandha/eif`

```
import eif as iso
forest  = iso.iForest(X, ntrees=200, sample_size=256, ExtensionLevel=1)
paths = forest.compute_paths(X_in=X)
```

Tabella Anomalie EIF

In 5.1 sono riportate le anomalie in ordine decrescente di "scores", dove più lo score è alto, più il bonifico è anomalo.

CustomerId	Country	Amount	isPc	Browser	OS	IBAN	SIA	CAP	SAE	RAE	company	scores
7617525401	Austria	264986.74	1	Chrome	Windows	AT34000	BUV4U	39031	430	161	SRL	0,7141
1886395602	Italy	182955.25	1	Safari	Mac OS X	DE28050100	AMU3P	39100	615	11	SS	0,7038
49733508	Austria	200000.0	1	Edge	Windows	AT37431	BZHIN	39036	430	660	SRL	0,6969
7098914101	Austria	190000.0	1	Chrome	Windows	AT15000	A56UA	39049	430	830	SRL	0,6968
41999520	Italy	270000.0	0	Other	Linux	IT3493	347D3	39038	430	830	GMBH	0,6967
93764501	Italy	500000.0	1	Chrome	Windows	NLDEUT	ASR05	39100	491	11	SS	0,6933
93764501	Italy	500000.0	1	Chrome	Windows	NLDEUT	ASR05	39100	491	11	SS	0,6933
41999520	Italy	126814.0	1	Firefox	Windows	DE30030880	347D3	39038	430	830	GMBH	0,6883
1886395602	Italy	110180.0	1	Chrome	Mac OS X	NLRABO	AMU3P	39100	615	11	SS	0,6877

Figura 5.1: Tabella anomalie EIF

5.4.2 Empirical-Cumulative-distribution-based Outlier Detection

Implementazione

La documentazione è disponibile su `https://pyod.readthedocs.io/en/latest/pyod.models.html`

```
from pyod.models.ecod import ECOD
clf = ECOD(contamination=0.0001)
clf.fit(X)
labels=clf.labels_
scores = clf.decision_scores_
```

Parametri

Per la generazione del modello è stato utilizzato un singolo parametro:

- **contamination=0.0001** è la proporzione di outliers nel dataset, usato nel fitting per definire il threshold per gli scores dei samples

Per il nostro dataset, contamination=0.0001 è stato deciso basandosi su [27] uno studio a cura dell'ufficio centrale antifrode dei mezzi di pagamento

Tabella anomalie ECOD

In 5.2 sono riportate le anomalie in ordine decrescente di "scores", dove più lo score è alto, più il bonifico è anomalo.

customerId	country	amount	isPc	browser	OS	IBAN	SIA	CAP	SAE	RAE	company	scores
45915153	Italy	299	0	Other	Linux	IT8904	082X3	30024	614	342	DI	38,1412
45915153	Italy	687	0	Other	Linux	IT8904	082X3	30024	614	342	DI	37,6082
45915153	Italy	1000	0	Other	Linux	IT8904	082X3	30024	614	342	DI	37,3883
45915153	Italy	1000	0	Other	Linux	IT8904	082X3	30024	614	342	DI	37,3883
68863923	Italy	8,2	1	Chrome	Mac OS X	BE0	B6R6Z	39053	288	0	SRL	36,4081
87157886	Italy	3585	0	Other	Linux	CZ100	CTA95	35131	430	259	SRL	35,7734
1886395602	Italy	49779	1	Chrome	Mac OS X	LVUNLA	AMU3P	39100	615	11	SS	35,4966
1886395602	Italy	16962	1	Chrome	Mac OS X	LVUNLA	AMU3P	39100	615	11	SS	34,2071
1886395602	Italy	16548	1	Chrome	Mac OS X	LVUNLA	AMU3P	39100	615	11	SS	34,1927

Figura 5.2: T abella anomalie ECOD

5.4.3 Local Outlier Factor

Implementazione

La documentazione è disponibile su

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html

```
from sklearn.neighbors import LocalOutlierFactor


clf = LocalOutlierFactor(n_neighbors=20,contamination=0.0001)
clf.fit_predict(X)
scores=clf.negative_outlier_factor_
label=clf.fit_predict(X)
```

subsubsection Per la generazione del modello sono stati utilizzati 2 parametri:

- **n_neighbors=20** è il numero di neighbors, il valore di default è 20, funziona bene in generale.
- **contamination=0.0001** è la proporzione di outliers nel dataset, usato nel fitting per definire il threshold per gli scores dei samples

Tabella anomalie LOF

In 5.3 sono riportate le anomalie in ordine crescente di "scores", più lo score è negativo, più il bonifico è anomalo.

customerId	country	amount	isPc	browser	OS	IBAN	SIA	CAP	SAE	RAE	company	scores
2814040702	Italy	388	1	Chrome	Windows	IT8749	CLMWZ	31040	430	231	SRL	-1,0075E+12
11276952	Italy	183	1	Chrome	Windows	IT8749	CKZYT	31038	430	314	SRL	-8,3707E+11
58140407	Italy	300	1	Chrome	Mac OS X	BE967	CKMJR	31020	430	161	SRL	-8,3012E+11
4094604	Italy	260	1	Chrome	Windows	IT8904	CLKXT	35010	430	314	SRL	-7,5226E+11
76176189	Italy	320	1	Chrome	Mac OS X	IT8728	CM5CR	30174	492	614	SAS	-7,247E+11
24096878	Italy	179,14	1	Chrome	Windows	IT8590	CR502	31033	430	11	SRL	-5,4182E+11
76176189	Italy	200	1	Safari	Mac OS X	IT8794	CM5CR	30174	492	614	SAS	-5,3378E+11
3555243001	Italy	317,7	1	Chrome	Mac OS X	IT8904	CJ4TM	31025	430	473	SRL	-5,0546E+11
28140407	Italy	260	1	Chrome	Windows	IT8399	CLMWZ	31040	430	231	SRL	-4,5752E+11

Figura 5.3: Tabella anomalie LOF

5.5 Classification and Regression Tree

Nelle tabelle , 5.1, 5.2 e 5.3 sono state elencate le anomalie trovate dagli algoritmi EIF, ECOD e LOF. Per evidenziare le possibili cause di tali anomalie si utilizzerà un Classification and regression tree (CART).

5.5.1 Decision Forest

L'algortimo utilizzato è Tensorflow Decision Forest la cui documentazione è disponibile su `https://www.tensorflow.org/decision_forests`. TensorFlow Decision Forests (TF-DF) è una raccolta di algoritmi all'avanguardia per l'addestramento, il servizio e l'interpretazione dei modelli Decision Forest . La libreria è una raccolta di modelli Keras e supporta la classificazione e la regressione.

Modello

```
import tensorflow_decision_forests as tfdf


X=tfdf.keras.pd_dataframe_to_tf_dataset(df, label="is_anomaly")
model = tfdf.keras.RandomForestModel()
model.fit(X)
print(model.summary())
```

Il metodo `summary()` rende disponibili importanti informazioni rigurdanti il modello

```
   Variable Importance: MEAN_MIN_DEPTH:
    1.     "__LABEL"  6.185732 ################
    2.     "country"  6.091496 ################
    3.        "isPC"  6.058083 ################
    4.         "SAE"  5.654172 ##############
    5. "companyType"  5.566608 #############
    6.     "browser"  4.837116 ###########
    7.          "OS"  4.831958 ###########
    8.         "SIA"  4.615351 ##########
    9.         "CAP"  4.593877 ##########
   10.          "CC"  3.143710 #####
   11.          "BC"  2.922421 ####
   12.      "amount"  2.727535 ###
   13.         "RAE"  1.627456

   Number of nodes by tree:
Count: 300 Average: 43.0067 StdDev: 6.51664
Min: 25 Max: 65 Ignored: 0
------------------------------------------------
[ 25, 27)  1   0.33%   0.33%
[ 27, 29)  1   0.33%   0.67%
[ 29, 31)  2   0.67%   1.33%
[ 31, 33)  6   2.00%   3.33% #
[ 33, 35) 14   4.67%   8.00% ###
[ 35, 37) 18   6.00%  14.00% ####
[ 37, 39) 21   7.00%  21.00% ####
[ 39, 41) 31  10.33%  31.33% #######
[ 41, 43) 47  15.67%  47.00% ##########
[ 43, 45) 30  10.00%  57.00% ######
[ 45, 47) 39  13.00%  70.00% ########
[ 47, 49) 29   9.67%  79.67% ######
[ 49, 51) 24   8.00%  87.67% #####
[ 51, 53) 15   5.00%  92.67% ###
[ 53, 55)  5   1.67%  94.33% #
[ 55, 57)  4   1.33%  95.67% #
[ 57, 59)  5   1.67%  97.33% #
[ 59, 61)  5   1.67%  99.00% #
[ 61, 63)  2   0.67%  99.67%
[ 63, 65]  1   0.33% 100.00%
```

```
tfdf.model_plotter.plot_model_in_colab(model, tree_idx=0, max_depth=3)
```

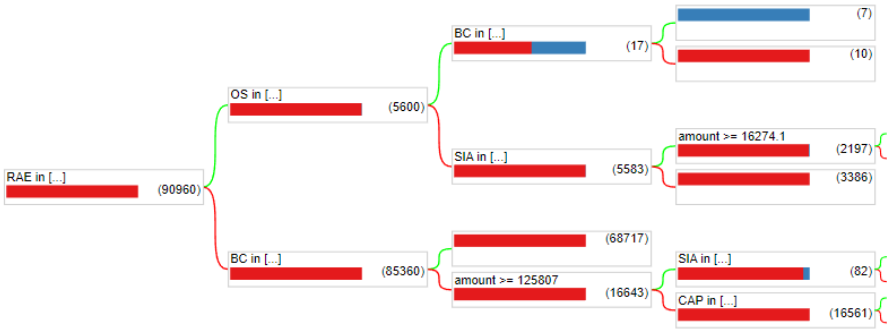

Figura 5.4: Decision Tree con max_depth=3

5.6 Synthetic Minority Over-sampling Technique for Nominal and Continuous

Synthetic Minority Over-sampling Technique (SMOTE) è una tecnica statistica per bilanciare il dataset generando nuove istanze da casi di minoranza esistenti forniti come input. Nel nostro caso il rapporto tra le transazioni frodate e il totale delle transazioni genuine è stato stimato come 0,0105% [27] ed aumentato fino al 50%. SMOTE non è in grado di gestire dati di tipo categorico, per questo motivo è necessario utilizzare una sua variante: Synthetic Minority Over-sampling Technique for Nominal and Continuous (SMOTE-NC). SMOTE-NC è grado di gestire dataset contenenti feautures categoriche e numeriche.

5.6.1 Implementazione

```
from imblearn.over_sampling import SMOTENC
smote_nc = SMOTENC(categorical_features=categorical_features_list, random_state=0)
X_resampled, y_resampled = smote_nc.fit_resample(X, y)
print('PRE-SMOTENC DATASET SIZE: ', X.shape)
print('POST-SMOTENC DATASET SIZE: ', X_resampled.shape)
```

```
PRE-SMOTENC DATASET SIZE:  (90960, 12)
POST-SMOTENC DATASET SIZE:  (181862, 12)
```

5.7 Supervised Learning

Grazie agli algoritmi di apprendimento non supervsionato EIF, ECOD e LOF siamo in grado di assegnare una label che distingue le possibili frodi dalle transazioni genuine. Le frodi rilevate costituiscono lo 0,01% del dataset e non sono sufficienti per allenare un modello di apprendimento supervisionato. Per questo motivo è stato utilizzato SMOTE-NC per portare a 1:1 il rapporto tra frodi e bonifici non anomali.

5.7.1 KNeighborsClassifier

Implementazione

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
                    X, y, test_size=0.2, random_state=1)

neigh = KNeighborsClassifier(n_neighbors=k)
neigh.fit(X_train, y_train)
```

5.7.2 Naive Bayes

Implementazione

```
from sklearn.naive_bayes import CategoricalNB
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
                    X, y, test_size=0.2, random_state=1)

model=CategoricalNB()
model.fit(X_train, y_train)
```

5.8 API REST

Al fine di fornire una semplice interfaccia utente con l'opportunità di caricare un file .csv di cui calcolare l'anomaly score è stata progettata una API REST con Flask [21]. **Flask** è un micro-framework Web scritto in Python, basato sullo strumento Werkzeug WSGI e con il motore di template Jinja2. È distribuito con licenza libera BSD. È stato inoltre utilizzato il modulo **pickle** per caricare il modello e l'encoder dictionary.

```
model = pickle.load(open('model.pkl', 'rb'))
enc_dict = pickle.load(open('enc_dict.pkl','rb'))
```

Dati il modello e l'encoding dictionary è possibile calcolare l'anomalia di nuovi punti forniti dall'utente in un file .csv. Questo file verrà processato nella stessa modalità dopo aver eseguito la stessa procedura di pulizia dei dati 5.2 e preprocessing 5.3. Il calcolo dell'anomaly score viene effettuato tramite i metodi sottostanti.

```
scores = model.decision_function(labeled_df_copy)
anomaly_score = model.predict(labeled_df_copy)
```

Usando alcuni bonifici di test il risultato è il seguente:

b2b anomaly detection

Scegli il file | Nessun file scelto | predict csv

	country	amount	isPc	browser	osFamily	IBAN_CC	IBAN_BC	SIA	CAP	SAE	RAE	companyType	scores	anomaly_score
0	Italy	500.0	1	Chrome	Windows	IT	2008	CEFGV	31039	430	505	SRL	0.279952	1
1	Italy	100000.0	1	Chrome	Windows	IT	2008	CEFGV	31039	430	505	SRL	0.158824	1
2	Italy	100000.0	1	Opera	Windows	IT	2008	CEFGV	31039	430	505	DI	0.005055	1
3	Italy	100000.0	1	Safari	Mac OS X	IT	2008	CEFGV	31039	430	505	GMBH	-0.022378	-1

Figura 5.5: Anomaly score tramite API Flask

Capitolo 6

Analisi delle performance

In questo capitolo verranno analizzate le performance degli algoritmi supervisionati utilizzati nella sezione 5.7, ovvero KNN e Naive Bayes.

6.1 Metriche

Le metriche utilizzate nella valutazione sono:

- $Precision = \frac{tp}{tp+fp}$
- $Recall = \frac{tp}{tp+fn}$
- $F1 = \frac{2 \cdot precision \cdot recall}{precision+recall}$
- $Accuracy = \frac{tp+tn}{tp+tn+fp+fn}$

tp: vero positivo

tn: vero negativo

fp: falso positivo

fn: falso negativo

6.2 Tabella delle performance

	KNN	NB
Precision	0.9408	0.9877
Recall	0.9775	0.9979
F1	0.9588	0.9877
Accuracy	0.9583	0.9938

Tabella 6.1: Tabella delle performance.

Nella tabella 6.1 vengono confrontate le performance degli algoritmi supervisionati presentati nella prima riga, ovvero, K-Nearest Neighbor (KNN) e Naive Bayes (NB). La prima colonna elenca le metriche di confronto: precision, recall, F1 e accuracy, nella seconda colonna vi sono i risultati relativi a KNN e sulla terza quelli relativi a NB. Si nota che su tutte le metriche considerate, NB presenta i punteggi più alti e questo suggerisce che questo algoritmo sia più indicato per il caso d'uso.

6.3 Discussione dei risultati

L'utilizzo di 3 distinti algoritmi di apprendimento non supervisionato ha dato in output 3 distinte tabelle di anomalie. Lo scopo di utilizzare più di un algoritmo era di effettuare una cross-validation: se più algoritmi avessero dato lo stesso output dati gli stessi input, avremmo avuto maggiore confidenza riguardo alla definizione di un bonifico più rischioso rispetto alla norma. In questo studio si è deciso di effettuare l'operazione di unione su tutte le anomalie rilevate, di bilanciare il dataset mantenendo un rapporto 1:1 di bonifici anomali-conformi grazie l'utilizzo di SMOTE e infine di creare un modello tramite un algoritmo di apprendimento supervisionato. In questo modo è stato possibile riunire in un unico modello l'identificazione dei diversi tipi di anomalie riscontrati nei casi non supervisionati. Nonostante SMOTE abbia reso possibile l'addestramento del modello finale è importante considerare che circa il 50% del dataset ribilanciato era costituito da dati artificiali creati a partire dallo 0,001% del dataset iniziale.

Capitolo 7

Conclusioni

7.1 Conclusioni

Allo scopo di rilevare le anomalie in un dataset non etichettato composto da bonifici B2B sono state utilizzate in cascata tecniche di apprendimento supervisionato e non supervisionato. La scelta di Python come linguaggio di programmazione è giustificata dall'elevata disponibilità di librerie specializzate nell'analisi, pulizia dei dati e nel ML. Operazione preliminare fondamentale è stata la pulizia del dataset tramite gli strumenti forniti dalla libreria pandas: rimozione di tuple con dati mancanti o non compatibili e conversione di tipi di dato. Inizialmente si disponeva di 317287 transazioni descritte da 25 colonne che in seguito sono diventate 90960 su 15 colonne. Successivamente è stato utilizzato il LabelEncoder disponibile su scikit-learn per effettuare l'encoding di variabili categoriche. Una volta conclusa la fase di preprocessing è stato possibile utilizzare ECOD dalla libreria PyOD, LOF da scikit-learn, EIF disponibile su una GitHub repo, 3 algoritmi di apprendimento non supervisionato specializzati nel rilevamento di anomalie, queste poi sono state riunite in un'unico dataset. L'utilizzo combinato di algoritmi supervisionati a quelli non supervisionati è stato suggerito dalla lettura di "Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection"[18] e "Unsupervised Anomaly Detection Based on Minimum

Spanning Tree Approximated Distance Measures and its Application to Hydropower Turbines" [4] Dato l'alto sbilanciamento tra anomalie e valori nella norma è stato utilizzato un metodo di synthetic over-sampling chiamato SMOTE per portare il rapporto a 1:1. In tal modo è stato possibile allenare 2 algoritmi di apprendimento supervisionato quali KNN e Naive Bayes, seguentemente confrontati per livello di performance ottenute. Per indagare sulle possibili cause che rendessero anomalo un bonifico è stato costruito un Classification and Regression Tree grazie alla libreria TensorFlow Decision Forests, strumento utile per gli esperti di dominio per valutare la genuinità delle anomalie. Allo scopo invece di prototipare una dashboard per un operatore bancario è stata scritta una primitiva interfaccia supportata da REST; in questo caso le librerie utilizzate sono: pickle per il salvataggio e successivo caricamento del modello e del labelencoder, flask per implementare REST.

7.2 Sviluppi futuri

L'apprendimento automatico è un campo in costante crescita e per questo motivo sarà possibile, negli anni, analizzare lo stesso dataset con algoritmi più efficienti e accurati. Per quanto riguarda possibili implementazioni e miglioramenti attuabili nel breve periodo, sarebbe interessante confrontare gli algoritmi qui adottati con sistemi di reti neurali, più complesse nella configurazione iniziale ma con risultati promettenti nel campo dell'anomaly detection. Ulteriori sviluppi consistono nel focalizzare l'attenzione sull'utilizzo di algoritmi specializzati nel real time anomaly detection for streaming data e nell'anomaly detection in time series. Rispetto al lato front-end, lo studio deficita dell'implementazione di una dashboard per supportare le decisioni di un esperto di dominio, vi è dunque la possibilità di integrare e successivamente espandere l'interfaccia REST qui implementata.

Elenco delle figure

Bibliografia

[1] *Various-types-of-machine-learning-techniques*. https://www.researchgate.net/figure/Various-types-of-machine-learning-techniques_fig4_350297716. Accessed: 2022-09-30.

[2] Songqiao Han et al. *ADBench: Anomaly Detection Benchmark*. 2022. DOI: 10.48550/ARXIV.2206.09426. URL: https://arxiv.org/abs/2206.09426.

[3] Waleed Hilal, S. Andrew Gadsden e John Yawney. «Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances». In: *Expert Systems with Applications* 193 (2022), p. 116429. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.116429. URL: https://www.sciencedirect.com/science/article/pii/S0957417421017164.

[4] Imtiaz Ahmed, Aldo Dagnino e Yu Ding. «Unsupervised Anomaly Detection Based on Minimum Spanning Tree Approximated Distance Measures and its Application to Hydropower Turbines». In: *IEEE Transactions on Automation Science and Engineering* 16.2 (2019), pp. 654–667. DOI: 10.1109/TASE.2018.2848198.

[5] *Fundamentals of Classification and Regression Trees (CART)*. https://mathanrajsharma.medium.com/fundamentals-of-classification-and-regression-trees-cart-e9af0b152503. Accessed: 2022-09-30.

[6] Migran N. Gevorkyan et al. «Review and comparative analysis of machine learning libraries for machine learning». In: *Discrete*

and Continuous Models and Applied Computational Science 27.4 (2019), pp. 305–315. ISSN: 2658-4670. URL: https://journals.rudn.ru/miph/article/view/22913.

[7] Giang Nguyen et al. «Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey». In: *Artificial Intelligence Review* 52.1 (giu. 2019), pp. 77–124. ISSN: 1573-7462. DOI: 10.1007/s10462-018-09679-z. URL: https://doi.org/10.1007/s10462-018-09679-z.

[8] V Dheepa e R Dhanapal. «Analysis of credit card fraud detection methods». In: *International journal of recent trends in engineering* 2.3 (2009), p. 126.

[9] Pooja Chougule et al. «Genetic K-means algorithm for credit card fraud detection». In: *International Journal of Computer Science and Information Technologies (IJCSIT)* 6.2 (2015), pp. 1724–1727.

[10] Fayaz Itoo, Meenakshi e Satwinder Singh. «Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection». In: *International Journal of Information Technology* 13.4 (ago. 2021), pp. 1503–1511. ISSN: 2511-2112. DOI: 10.1007/s41870-020-00430-y. URL: https://doi.org/10.1007/s41870-020-00430-y.

[11] *L14. Anomaly Detection*. https://www.slideshare.net/mlvlc/l14-anomaly-detection. Accessed: 2022-09-30.

[12] *Isolation Forest*. https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf?q=isolation-forest. Accessed: 2022-09-30.

[13] *Replace Outlier Detection by Simple Statistics with ECOD*. https://medium.com/geekculture/replace-outlier-detection-by-simple-statistics-with-ecod-f95a7d982f79. Accessed: 2022-09-30.

[14] *ECOD*. https://zhuanlan.zhihu.com/p/511406695. Accessed: 2022-09-30.

[15] Zhangyu Cheng, Chengming Zou e Jianwei Dong. «Outlier detection using isolation forest and local outlier factor». In: *Proceedings of the conference on research in adaptive and convergent systems*. 2019, pp. 161–168.

[16] *SMOTE*. https://www.jair.org/index.php/jair/article/view/11192/26406. Accessed: 2022-09-30.

[17] *Algoritmo K-Nearest Neighbors*. https://www.ibm.com/it-it/topics/knn. Accessed: 2022-09-30.

[18] Fabrizio Carcillo et al. «Combining unsupervised and supervised learning in credit card fraud detection». In: *Information sciences* 557 (2021), pp. 317–331.

[19] The pandas development team. *pandas-dev/pandas: Pandas*. Ver. latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[20] F. Pedregosa et al. «Scikit-learn: Machine Learning in Python». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[21] *Flask*. URL: https://flask.palletsprojects.com/en/2.2.x/.

[22] *pickle — Python object serialization*. URL: https://docs.python.org/3/library/pickle.html.

[23] Charles R. Harris et al. «Array programming with NumPy». In: *Nature* 585.7825 (set. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[24] Yue Zhao, Zain Nasrullah e Zheng Li. «PyOD: A Python Toolbox for Scalable Outlier Detection». In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7. URL: http://jmlr.org/papers/v20/19-011.html.

[25] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[26] Guillaume Lemaître, Fernando Nogueira e Christos K. Aridas. « Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning». In: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5. URL: http://jmlr.org/papers/v18/16-365.html.

[27] Antonio Adinolfi e Sogei S.p.A.: Francesca di Brisco Stefano Grossi Alessandra de Castro Eugenio Cangiano Ufficio VI UCAMP: G. di F. – Col. Giovanni Palma G. di F. – Cap. Sergio Paolocci G. di F. – App.Sc. Riccardo Valenza. «Rapporto statistico sulle frodi con le carte di pagamento 2020». In: (). URL: https://www.dt.mef.gov.it/export/sites/sitodt/modules/documenti_it/antifrode_mezzi_pagamento/antifrode_mezzi_pagamento/Rapporto_statistico_sulle_frodi_con_le_carte_di_pagamento_-_edizione_2020.pdf.