UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea in informatica

CCFDetector: Using ML against Credit Card Frauds

Progetto realizzato per l'esame di Fondamenti di Intelligenza Artificiale

Realizzato da:

Marco Santoriello
Mat. 0512114100

ANNO ACCADEMICO 2023/2024

Indice

1 Introduzione

Negli ultimi anni, sempre più piede hanno preso i pagamenti elettronici, al punto che, anche in Italia, per legge, ogni commerciante deve essere munito di un dispositivo che permetta al cliente di pagare utilizzando la propria carta di credito, rischiando, in caso di mancato adempimento a questa legge, ingenti sanzioni pecuniarie.
Come è facile immaginare, questo cambiamento nel modo in cui il denaro viene messo in circolazione ha interessato notevolmente criminali e truffatori (i cosiddetti *scammers*), i quali hanno trovato non pochi modi di impossessarsi illecitamente delle carte di credito altrui, o dei relativi dati associati ad esse. Basti pensare che negli Stati Uniti, secondo la *Federal Trade Commission*, la tipologia di furto di identità più diffusa è proprio relativa alle frodi che interessano pagamenti elettronci.
Queste frodi possono avvenire in svariati modi: si parte dal furto vero e proprio della carta di credito, fino ad arrivare, tramite diversi metodi, all'appropriazione dei soli dati della carta che abilitano al pagamento, passando per la clonazione delle carte e per l'utilizzo di dispositivi *contactless* in posti affollati in prossimità dei portafogli delle ignare vittime.
Questo progetto nasce con lo scopo di addestrare un modello di Machine Learning che permetta una facile ed affidabile individuazione delle transazioni fraudolente, con lo scopo di bloccarle tempestivamente per evitare sottrazioni di denaro alle vittime. Il progetto è consultabile nella sua interezza su GitHub.

1.1 Specifica PEAS

La specifica PEAS (Performance, Environment, Actuators, Sensors) è un sistema che permette di descrivere l'ambiente operativo di un agente intelligente. L'obiettivo principale del progetto è quello di massimizzare la capacità dell'agente di rilevare transazioni fraudolente. Di seguito la specifica PEAS:

- Performance: ROC-AUC Curve, precision, recall, f1.
- Environment: ambiente delle transazioni finanziarie.
- Actuators: algoritmo di classificazione.
- Sensors: dati ricevuti in input relativi ad una transazione.

1.2 Caratteristiche dell'ambiente

L'ambiente operativo dell'agente risulta essere:

- **Single-agent**.
- **Completamente osservabile**: attraverso i sensori, l'agente godrà di una conoscenza completa dell'ambiente.
- **Dinamico**: l'ambiente delle transazioni finanziarie è in continua evoluzione. Di conseguenza lo sono anche i patterns relativi alle frodi.
- **Discreto**:le variabili assumono valori in un limitato intervallo; alcune, invece, assumono valori distinti e separati.
- **Episodico**: L'esperienza dell'agente è divisa in episodi atomici, dove ciascun episodio consiste dall'eseguire una singola azione.
- **Stocastico**: Lo stato dell'ambiente non è determinato dallo stato corrente o dall'azione eseguita dall'agente.

1.3 Analisi del Problema

Il progetto mira, dunque, alla costruzione di un modello di Machine Learning, volto alla risoluzione di un problema di **apprendimento supervisionato**. In particolare, come verrà reso evidente da questo documento nelle successive sezioni, il problema da affrontare risulta essere di **classificazione binaria**, avendo da predire una variabile che puo' assumere valori 0, oppure 1.

Gli strumenti utilizzati per la realizzazione sono elencati di seguito:

- **Python**.
- **Jupyter Notebook**.
- **PyCharm** come IDE.
- **LaTeX** per la documentazione.
- **Canva** per la presentazione.
- **GitHub** per il versioning.

2 Data Understanding

2.1 Data Collection

Definito il Problem Statement, passo all'individuazione del dataset adatto per l'addestramento e la validazione del modello.

Trattandosi di dati particolarmente sensibili, il numero di dataset presenti online non è molto alto. Il primo dataset ritenuto particolarmente interessante per gli scopi di questo progetto, risultava avere quasi tutte le features criptate (per ovvi motivi di privacy) e delle quali non veniva fornita alcuna spiegazione, trattandosi di transazioni reali, rivelandosi non ottimale in termini di explainability. La scelta è poi ricaduta su un dataset contenente dati simulati, rilasciato da IBM.

2.2 Data Description

Il dataset in questione comprende dati, ottenuti attraverso un processo di simulazione che non è stato reso noto, di oltre 24 milioni di transazioni, di cui 30.000 fraudolente (cioè lo 0.1%). Essendo dati simulati, ci si aspetta siano fedeli rappresentazioni delle transazioni reali e delle loro caratteristiche.

Le features di questo dataset comprendono tutti i dettagli relativi a un pagamento elettronico (ID utente, ID carta, data, ora, modalità di pagamento, ecc.) e tutti i dettagli del venditore (i cui nomi sono stati codificati). La variabile target è rappresentata dalla colonna *Is Fraud?*.

Per ragioni legate alla capacità dell'hardware, decido di caricare soltanto 300.000 righe del dataset, verificando subito la presenza di transazioni fraudolente tra quelle importate. Procedo, a questo punto, con la documentazione dei dati.

Il dataset contiene 15 colonne che riporto di seguito, associando ad ognuna una breve descrizione di essa:

i. **User**: Id dell'utente che ha effettuato la transazione (Dtype: int64)

ii. **Card**: Id della carata che ha effettuato la transazione (Dtype: int64)

iii. **Year**: Anno in cui è avvenuta la transazione (Dtype: int64)

iv. **Month**: Mese in cui è avvenuta la transazione (Dtype: int64)

v. **Day**: Giorno in cui è avvenuta la transazione (Dtype: int64)

vi. **Time**: Istante in cui è avvenuta la transazione (Dtype: object)

vii. **Amount**: Ammontare (in dollari) della transazione (Dtype: object)

viii. **Use Chip**: Modalità di pagamento (Dtype: object)

ix. **Merchant Name**: Nome del venditore presso cui si ha acquistato (Dtype: int64)

x. **Merchant City**: Città del venditore presso cui si ha acquistato (Dtype: object)

xi. **Merchant State**: Stato del venditore presso cui si ha acquistato (Dtype: object)

xii. **Zip**: Zip code del venditore presso cui si ha acquistato (Dtype: float64)

xiii. **MCC**: Merchant Category Code (Dtype: int64)

xiv. **Errors?**: Se è avvenuto un errore, ne descrive la tipologia (Dtype: object)

xv. **Is Fraud?**: Transazione fraudolenta oppure lecita (Dtype: object)

Per quanto riguarda le modalita' di pagamento, rappresentate nella colonna *Use Chip*, osservo che sono di tre tipi:

- *Chip Transaction*: pagamento effettuato inserendo la carta, dotata di chip, nel lettore del terminale POS. Il chip, affinché la transazione possa essere autorizzata, richiede l'inserimento di un codice PIN.
- *Swipe Transaction*: pagamento effettuato utilizzando la striscia magnetica della carta. Tale tipologia di pagamento non prevede l'inserimento del PIN (perlomeno se il pagamento è al di sotto di una determinata soglia) per autorizzare la transazione.
- *Online Transaction*: pagamento online, effettuato inserendo i dati della carta manualmente, all'interno di un form. Anche in questo caso, a seconda dell'importo della transazione, potrebbe essere richiesta una misura di sicurezza aggiuntiva, oppure no.

Per poter continuare ad esplorare i dati, devo effettuare delle operazioni di conversione e splitting, anticipando, così, la fase di **data cleaning**. Innanzitutto, la feture *Amount*, contenente l'importo di ogni transazione in dollari, non risulta essere in formato numerico, per cui vado a convertirne tutti i suoi valori in float, non prima di aver rimosso il simbolo della valuta che precede le cifre.

Gestisco, poi, la caratteristica *Time* che provvedo a dividere nelle caratteristiche *Hour* e *Minute*, rimuovendola, infine, dal dataset. Vado ora ad analizzare la distribuzione delle varie features, per cercare di capire meglio eventuali patterns legati con le frodi.

In particolare, dal seguente grafico, osservo che la maggior parte degli importi che caratterizzano le transazioni fraudolente sono relativamente bassi, compresi, infatti nell'intervallo che va da 0 a 250 USD.

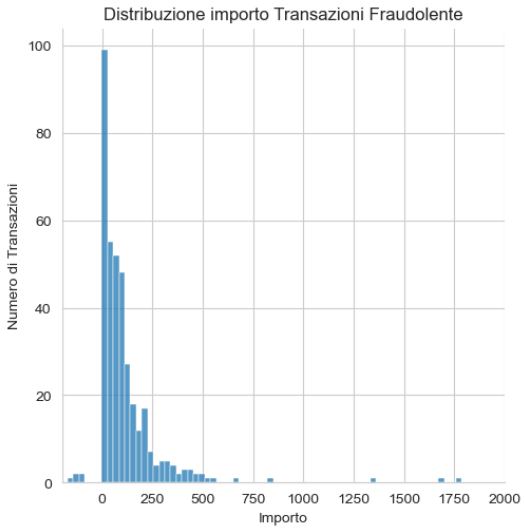

Figura 1: Distrubuzione degli importi relativi alle transazioni fraudolente

Inoltre, il metodo di pagamento maggiormente interessato da questo fenomeno, risulta essere il pagamento online. Invece, come mi aspettavo dalla precedente analisi fatta sui metodi di pagamento, quello tramite chip risulta essere il meno interessato, rivelandosi, dunque, il metodo piu' sicuro.

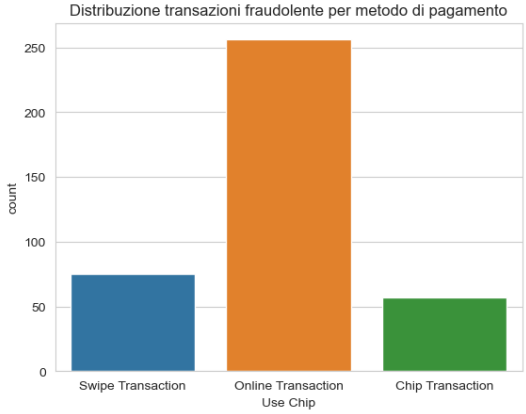

Figura 2: Distribuzione dei metodi di pagamento

Infine, analizzando la distribuzione dei valori della categoria dei venditori, risulta che ci siano alcune categorie più soggette a frodi rispetto ad altre, come mostrato nel seguente grafico.

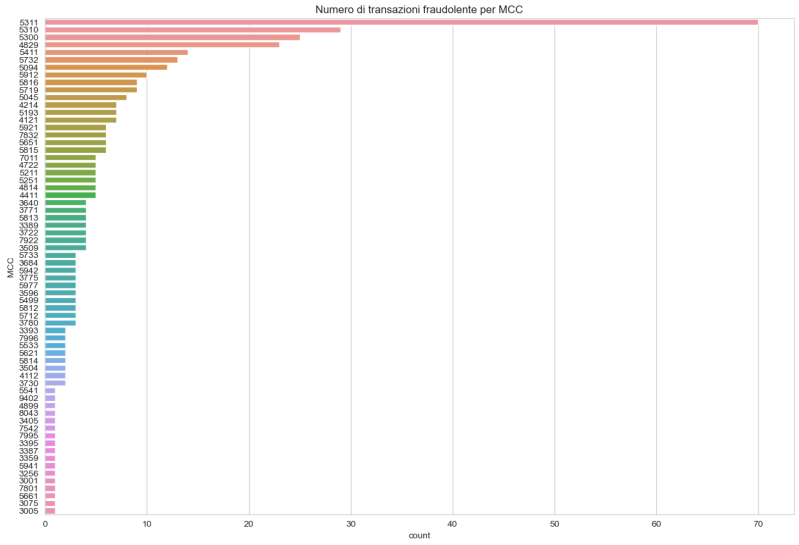

Figura 3: Distribuzione n. frodi per categoria di venditore

2.3 Data Quality

Relativamente alla qualità dei dati, la prima cosa che noto è che le colonne *Merchant State*, *Zip*, ed, in particolare *Errors*, contengono dei valori nulli.

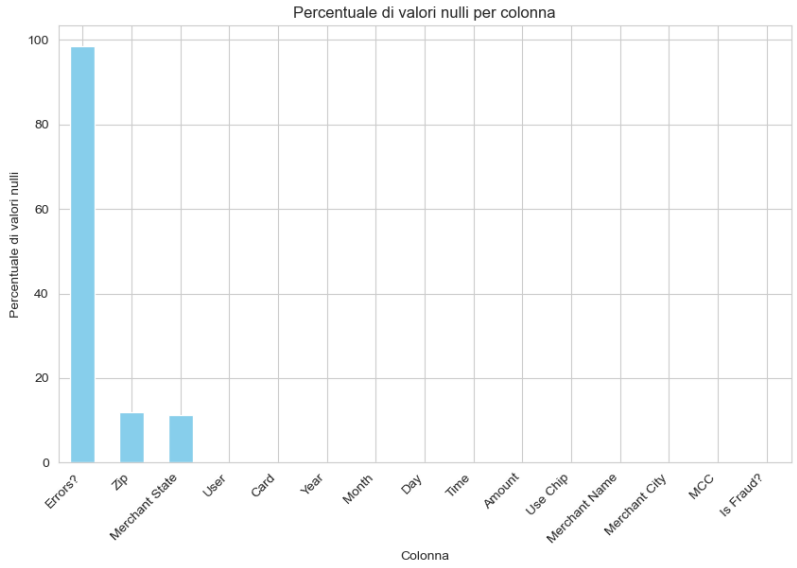

Figura 4: Percentuale valori nulli per colonna

Inoltre, le caratteristiche *Merchant State* e *Zip* possono essere ricavate da *Merchant City*, dunque potrebbe essere conveniente, contenendo valori nulli, rimuoverle. Procedo, dunque, a verificare è se il dataset è bilanciato o meno. Il numero di transazioni fraudolente è pari a 388, cioe' lo 0.129% del totale delle transazioni. Il dataset risulta essere altamente sbilanciato, come mostrato dal seguente grafico:

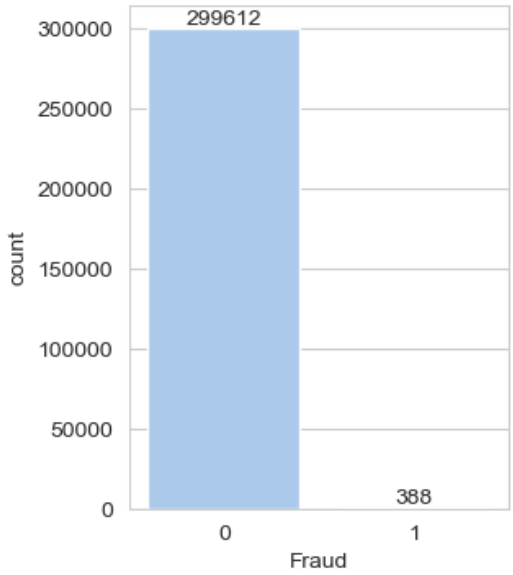

Figura 5: Distrubuzione delle transazioni fraudolente e lecite

L'utilizzo di un dataset così sbilanciato, porterebbe alla costruzione di un modello altamente impreciso e che, probabilmente, si *adatterà troppo* (*overfitting*) al problema, in quanto andrebbe ad assumere la maggior parte delle transazioni come non fraudolente, siccome queste costituiscono la stragrande maggioranza dei dati.

3 Data Preparation

In questa fase, vado ad effettuare le operazioni necessare per la pulizia dei dati, in modo che possano essere pronti per essere dati in input al modello.

3.1 Data Cleaning

Innanzitutto vado a gestire i valori nulli. Parto dalla caratteristica *Errors?*. Come analizzato in fase di Data Understanding, questa colonna contiene le varie tipologie di errore a cui una transazione puo' andare incontro. Come risulta dall'osservazione dei valori di questa feature, i valori nulli sembrano corrispondere al caso in cui non si verifica alcun errore. Per tale ragione, decido di rimpiazzare tutti i valori nulli con il valore *No error*. Tale modo di gestire i valori nulli si definisce **imputazione deduttiva**.

Come secondo step, vado a rimuovere le colonne *Merchant State* e *Zip*. Esse, infatti, posso essere ricavate dalla feature *Merchant City*, per cui, contenendo specialmente dei valori nulli, risulta conveniente la loro rimozione dal dataset.

Necessitando per il futuro addestramento del modello di dati numerici, vado a convertire le variabili categoriche *Use Chip*, *Merchant City* ed *Errors* in variabili numeriche, utilizzando il metodo **LabelEncoder()**, della libreria *scikit-learn*. Tale metodo assegna un intero univoco ad ogni categoria, in ordine di scoperta.

3.2 Feature Scaling

Nella fase di Feature Scaling, proseguo con l'osservazione delle distribuzioni dei valori delle varie caratteristiche, poiché addestrare un modello su un dataset che contiene colonne con insiemi di valori molto diversi tra loro, potrebbe portare tale modello a confondersi, sovrastimando o sottostimando l'importanza delle varie caratteristiche.
Le varie features, tra loro, assumono valori compresi in diversi ranges, per tale ragione, decido di effettuare la normalizzazione su tutte, tranne che sulla variabile dipendente.
Per lo scopo, utilizzo la tecnica di **Normalizzazione min-max**, che è una delle più comuni e diffuse tecniche di normalizzazione. Di seguito riportata la formula della Normalizzazione min-max:

$$x_{\text{norm}} = a + \frac{x - \min(X)}{\max(X) - \min(X)} \cdot (b - a)$$

Normalizzati i valori, passo alla fase di Feature Selection.

3.3 Feature Selection

Procedo con la fase di Feature Selection, in cui, come suggerisce il nome, vado ad individuare e selezionare le features che hanno potenza predittiva più alta, scartando quelle con potenza predittiva più bassa. Infatti, queste ultime potrebbero rendere il futuro modello più impreciso o peggiorarne le perfomances.
La tecnica che utilizzo è nota come **Matrice di Correlazione** (senza escludere la possibilità di integrare l'analisi con le osservazioni fatte in fase di data exploration). Tale matrice mostra i *coefficienti di correlazione* tra coppie di variabili. I coefficienti possono assumere valori nell'intervallo [-1, 1]. Valori vicini a 1 indicano che le due variabili sono **correlate positivamente**; valori vicini a -1 indicano che le due variabili sono **correlate negativamente**; valori vicini a 0, infine, indicano che tra le due variabili **non itercorre alcuna relazione**.

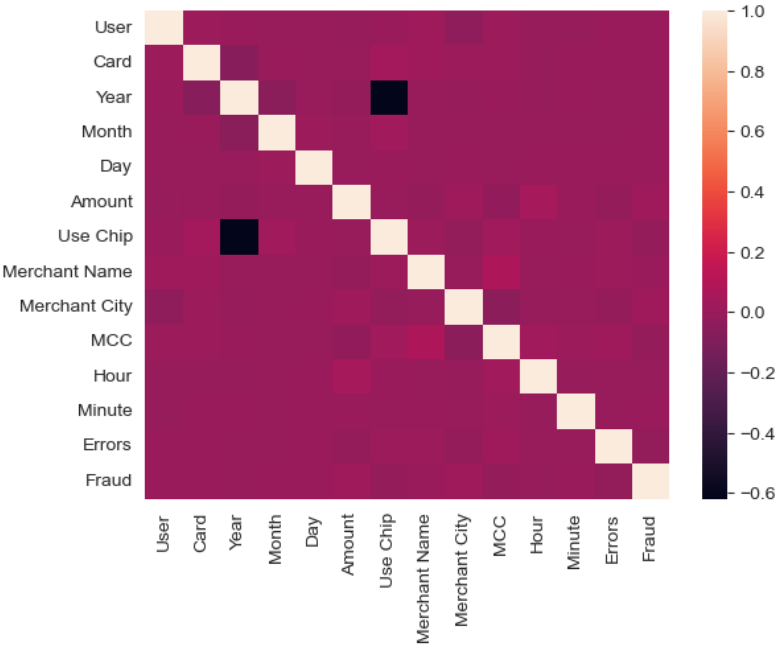

Figura 6: Matrice di correlazione

Per avere una rappresentazione ancora più chiara, vado a mostrare solo il grado di relazione che ciascuna variabile ha con la variabile dipendente.

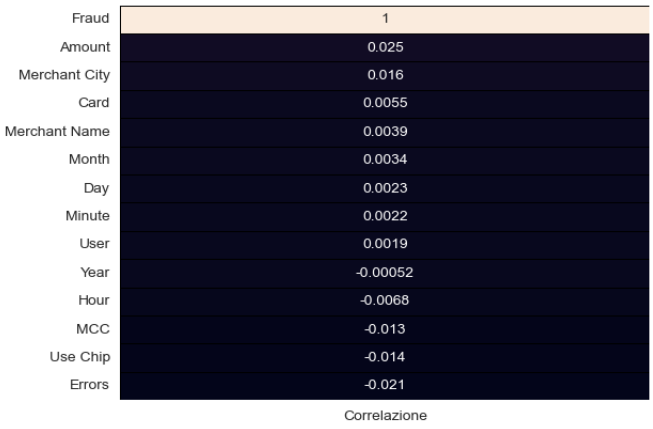

Figura 7: Correlazione tra features e variabile target

Le variabili maggiormente correlate con il target sono:

- *Amount*
- *Merchant City*
- *MCC*
- *Use Chip*
- *Errors*

Dunque, vado a scartare le restanti features, ad eccezione di *Year* e *Hour*, che dall'analisi iniziale, effettuata nella fase di esplorazione dei dati, sembravano avere una qualche correlazione con le transazioni fraudolente.

3.4 Data Balancing

Come analizzato nella fase di Data Understanding, il dataset risulta essere altamente *unbalanced*. Addestrare un modello su un dataset del genere, comporterebbe un elevato rischio di overfitting. Per ridurlo, è necessario andare a bilanciare il dataset.

3.4.1 Data Splitting

Prima di procedere, divido il dataset in *training set* e *test set*. E' cruciale effettuare questa divisione prima di effettuare qualsiasi operazione di sampling per evitare di incappare in problemi di **data leakage**. Infatti, sarebbe un errore testare e valutare il modello su dei dati bilanciati poiché questi dovrebbero rispecchiare la realtà quanto più è possibile. Se così non fosse, infatti, il modello effettuerebbe predizioni accurate in fase di addestramento, ma non in fase di rilascio.

Dunque, innanzitutto separo le features dalla variabile target. Per fare questo, utilizzo il metodo *iloc* della libreria pandas: in particolare, nella prima istruzione seleziono tutte le righe, tramite *:*, delle prime sette colonne (da 0 a 6) e poi, tramite *values*, converto in un array numpy; nella seconda istruzione, invece, seleziono tutte le righe della ottava colonna (di indice 7) del dataframe.

```
# Separating features and target
```

```
X = df1.iloc[:,:7].values
y = df1.iloc[:,7].values
```

Successivamente, divido il dataset assegnando il 20% dei dati al test set, mentre il restante 80% di essi costituiranno i dati di addestramento.

3.4.2 Sampling Techniqe Selection

Per bilanciare un dataset, posso utilizzare principalmente due tecniche (nelle varie declinaizoni):

- **undersampling**,
- **oversampling**.

La tecnica di undersampling, in generale, consiste nella rimozione di samples dalla classe di maggioranza che, nel nostro caso, è quella delle transazioni non fraudolente. Tuttavia, scegliendo di utilizzare questa tecnica, mi ritroverei con un dataset eccessivamente piccolo, che vedrebbe perdute la maggior parte delle informazioni, avendo soltanto, come osservato in precedenza, 388 samples della classe di minoranza (Frauds).

Allo stesso tempo, la tecnica di oversampling, che, invece, consiste nella generazione casuale di nuovi samples della classe di minoranza, potrebbe condurre il modello verso l'overfitting, in quanto, ancora, il numero dei samples della classe di minoranza e' eccessivamente basso.

Per tale ragione, decido di utilizzare una tecnica nota come **Synthetic Minority Oversampling Tecnhique** (**SMOTE**). SMOTE e' una tecnica di oversampling che, invece di creare copie di istanze esistenti della classe di minoranza, genera nuove istanze (dette istanze sintetiche) attraverso l'interpolazion. Dunque, mi permette di mitigare il rischio di overfitting. Di seguito illustrato il funzionamento:

1. Seleziona una istanza random all'interno della classe di minoranza
2. Identifica k vicini per il dato appena selezionato
3. Seleziona uno di questi vicini per il nuovo dato da creare
4. Calcola la distanza vettoriale tra il punto e il vicino selezionati
5. Moltiplica tale distanza per un numero casuale compreso tra 0 e 1

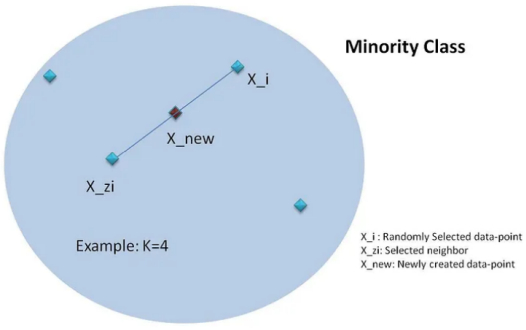

Figura 8: Meccanismo SMOTE

Prima dell'applicazione di SMOTE, all'interno dei dati di training si hanno:

- N. transazioni fraudolente: 307
- N. transazioni non fraudolente: 239693

Dopo l'applicazione di SMOTE, invece, si hanno:

- N. transazioni fraudolente: 239693
- N. transazioni non fraudolente: 239693