Ingegneria della conoscenza

Antonio Francesco Fiore, Gianluca Losciale

April 2021

Contents

1 Introduzione

È molto importante che le società di carte di credito siano in grado di riconoscere le transazioni **fraudolente** da quelle **lecite**, in modo tale che i clienti non vengano addebitati per articoli che non hanno esplicitamente acquistato.

2 Frode con carta di credito

Cos'è la **frode** con carta di credito? La frode con carta di credito si verifica quando qualcuno utilizza la carta di credito o le informazioni del conto di un'altra persona per effettuare acquisti non autorizzati o accedere a fondi tramite anticipi di cassa.

Le frodi con carta di credito non si verificano solo online; succede anche nei negozi fisici. In qualità di imprenditore, puoi evitare gravi mal di testa e pubblicità indesiderata riconoscendo un utilizzo potenzialmente fraudolento delle carte di credito nel tuo ambiente di pagamento.

2.1 Sfida

Non è sempre facile essere d'accordo sulla verità di base per ciò che significa "*frode*". Indipendentemente da come definisci la verità di base, la stragrande maggioranza delle accuse non è fraudolenta. La maggior parte dei commercianti non è esperta nella valutazione dell'impatto aziendale delle frodi.

Il problema di rilevamento delle frodi con carta di credito include la modellazione delle transazioni passate con carta di credito con la conoscenza di quelle che si sono rivelate essere una frode. Questo modello viene quindi utilizzato per identificare se una nuova transazione è fraudolenta o meno.

Il nostro obiettivo quindi è rilevare tutte le transazioni fraudolente riducendo al minimo le classificazioni di frode errate.

3 Osservazioni preliminari sul DataSet

Da una prima fase di osservazione preliminare dei dati, si osserva uno sbilanciamento dei dati tra quelli fraudolenti e non; Infatti risultano esserci soltanto **492** transazioni fraudolente su **284.807**, quindi solo lo **0,172%.**

Il set di dati è costituito da valori numerici dalle 28 caratteristiche trasformate "Principal Component Analysis (**PCA**)", per motivi di privacy tranne per "*Tempo*" e "*Quantità*" non sono dati trasformati.

```
Grandezza transizzioni fraudolente: (492, 31)
Grandezza transizzioni non-fraudolente: (284315, 31)
```

3.1 Influenza degli squilibri sulle prestazioni del modello

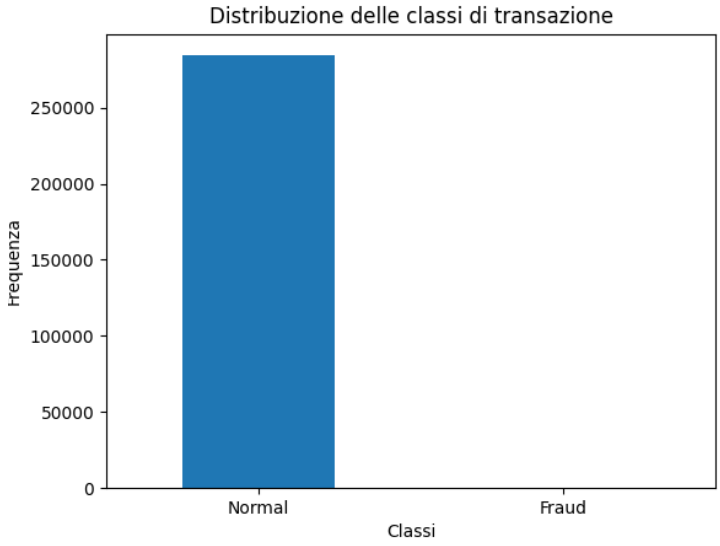
In generale, vogliamo massimizzare il richiamo limitando "*False Positive Rate (**FPR**)*", ma puoi classificare molti addebiti in modo errato e mantenere comunque un FPR basso perché hai un gran numero di veri negativi.

Ciò favorisce la scelta di una soglia relativamente bassa, che si traduce in un **recall** elevato ma con una **precision** estremamente bassa.

3.2 Miglioramento del DataSet

L'addestramento di un modello su un DataSet bilanciato migliora le prestazioni sui dati di convalida. Una soluzione a questo problema è: utilizzare tutte le transazioni fraudolente, ma sottocampionare le transazioni non fraudolente secondo necessità per raggiungere la nostra tariffa target.

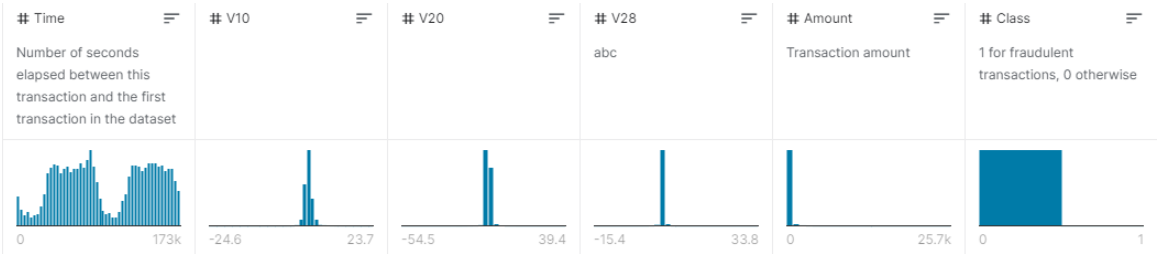
È necessario trovare un equilibrio che funzioni meglio nella produzione.



4 Analisi Distribuzione Dati

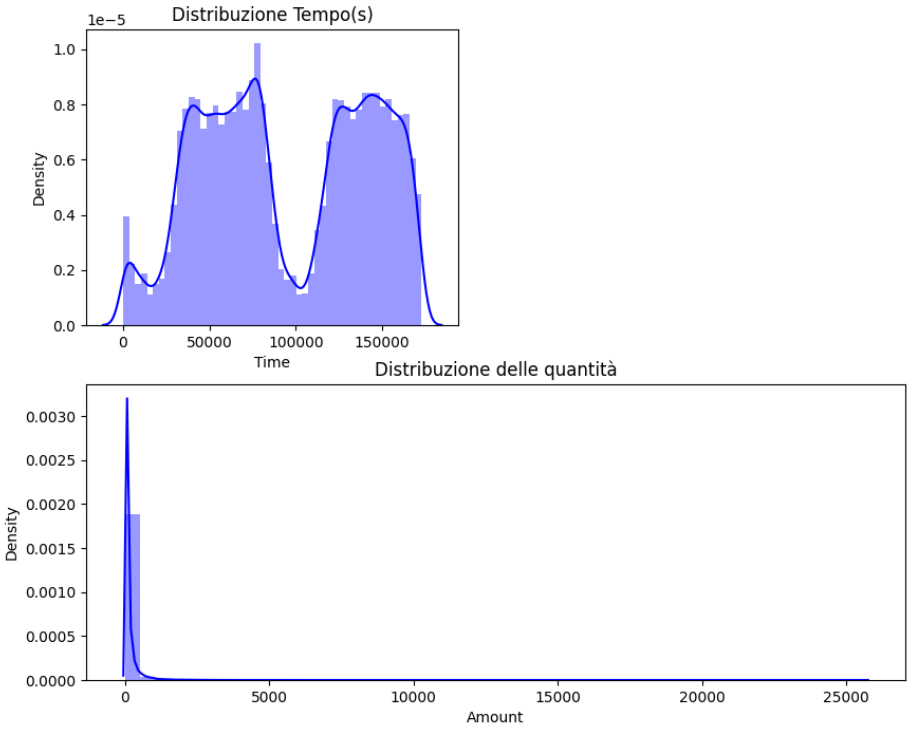
Partendo da quanto affermato in precedenza, occorre concentrare la nostra analisi sulle funzionalità non anonime: Tempo, Quantità.

Osserviamo prima le distribuzioni fornite già sul dataset:

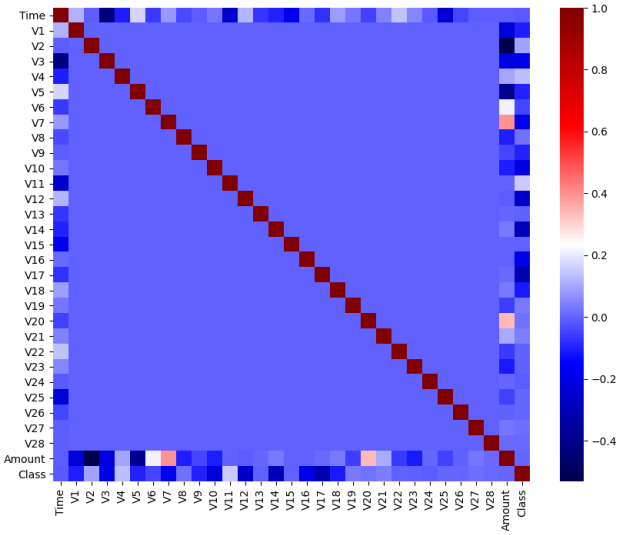


4.1 Tempo e Quantità

L'idea è che le transazioni fraudolente potrebbero verificarsi più spesso in un determinato peridodo di tempo



Osservando la figura soprastante, e le distribuzioni dei vari Valori dal dataset originale, non sembra che il momento della transazione sia davvero importante qui come da osservazione sopra. Ma sfruttiamo un'altro strumento:



La matrice di correlazione mostra anche che nessuna delle componenti PCA da V1 a V28 ha alcuna correlazione tra loro, tuttavia se osserviamo che la classe ha una qualche forma di correlazioni positive e negative con le componenti V ma non ha alcuna correlazione con il tempo e l'importo.

5 Apprendimento automatico

Ricordiamo che l'apprendimento è la capacità di un sistema di sfruttare l'esperienza per migliorare un comportamento, ai fini di migliorare l'accuratezza e la precisione di un sistema ed estendere le sue funzionalità.

L'apprendimento automatico è una variante rispetto alla programmazione tradizionale nella quale in una macchina si predispone l'abilità di apprendere dai dati forniti in input in maniera autonoma, senza l'utilizzo di istruzioni esplicite.

5.1 Apprendimento supervisionato

L'apprendimento supervisionato è una tecnica di apprendimento automatico il cui obiettivo è istruire il sistema in modo da poter elaborare automaticamente previsioni. Queste previsioni vengono fatte sui valori di output rispetto ad un preciso input fornito.

È possibile istruire il sistema alla previsione grazie alla **supervisione** di esempi forniti al sistema, sotto forma di coppia <***input***, ***output***>.

6 Preprocessing

Come prima fase di elaborazione dei dati, necessitiamo di portare il "tempo" e "quantità" nello stesso formato delle altre colonne.

Dividiamo il dataset in due gruppi che rappresentano le transazioni **"fraud"** e **"non fraud"** basandoci sulla colonna "***Class***", e da qui ricomponiamo un altro gruppo con un numero diverso di righe ma mantenendo una proporzionalità tra fraudolente e non.

In fase di test, abbiamo usato per motivi di tempo di elaborazione solo tra il 5-25%.

Splittiamo in 2 gruppi, **"x"** e **"y"**, nella x inseriamo tutte le colonne tranne "class" (che indica con un booleano se è fraudolenta o no) e nella y solo "class".

Successivamente divide ulteriormente per ricavare *4* gruppi per il **training** e **test** (per x e y) e altri 4 equivalenti per la fase di **validation**.

```
Addestramento:   x: (149523, 30),    y: (149523, 1)
-------------------------------------------------
Validazione:     x: (49841, 30),     y: (49841, 1)
-------------------------------------------------
Test:            x: (85443, 30),     y: (85443, 1)
```

7 Metodi e Modelli

Utilizzando la libreria ***sklearn*** sono stati utilizzati diversi modelli di classificazione e confrontati successivamente per capirne le potenzialià sul dataset preso e trovare possibili discrepanze.

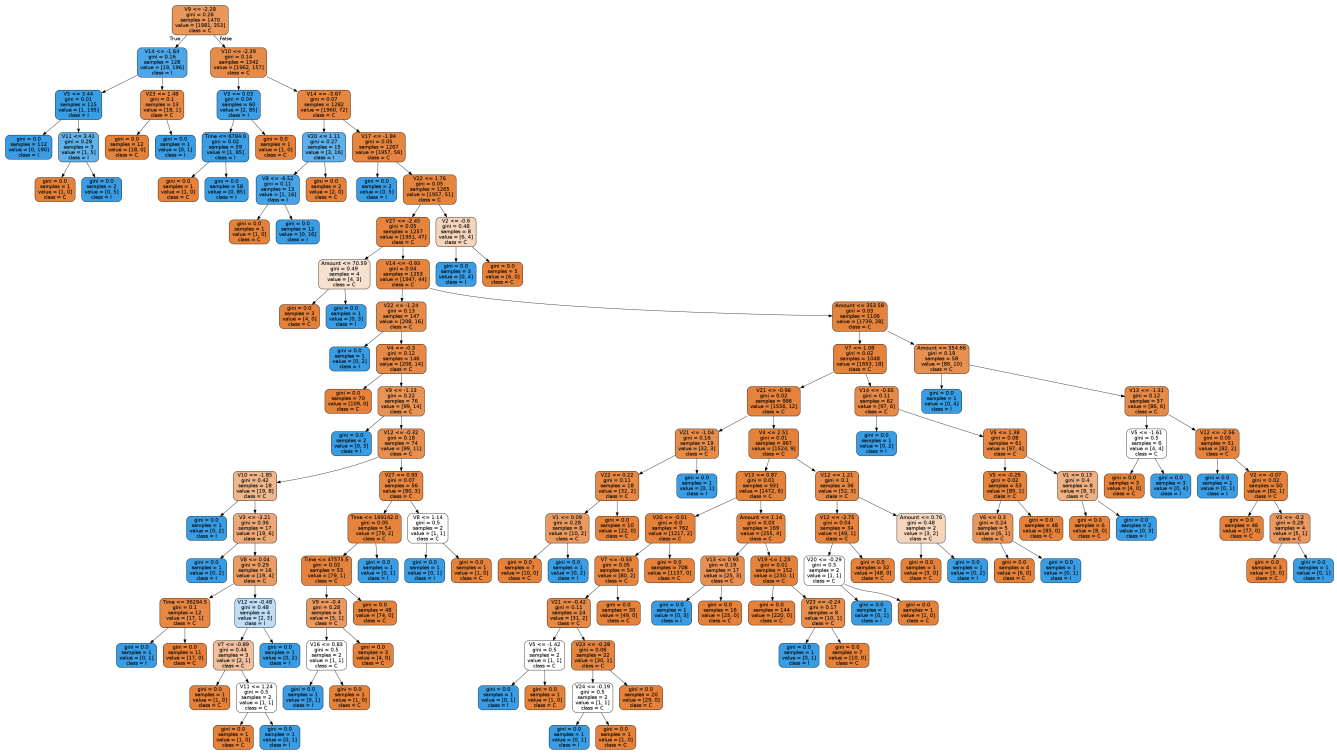
7.1 DecisionTree

Per la classificazione delle azioni fraudolente o meno, si può ricorrere agli **alberi di decisione** che possono essere visualizzate come una serie di domande booleano (si/no), poste ai dati per arrivare navigando nodo per nodo(non foglia), ai **nodi foglia** che indicano come classificare i dati;

Questo è molto simile ai ragionamenti svolti dalle persone, che osservano e si pongono domande fino ad arrivare ad una conclusione.

7.1.1 Perchè RandomForest

È un modello ottenuto dall'aggregazione tramite **bagging** di alberi di decisione. Esso è un metastimatore che si adatta ad una serie di alberi decisionali addestrati su vari sotto-campioni del dataset e *utilizza la* ***media*** di ogni singolo output di ogni albero per migliorare l'accuratezza predittiva e il controllo del sovradattamento.



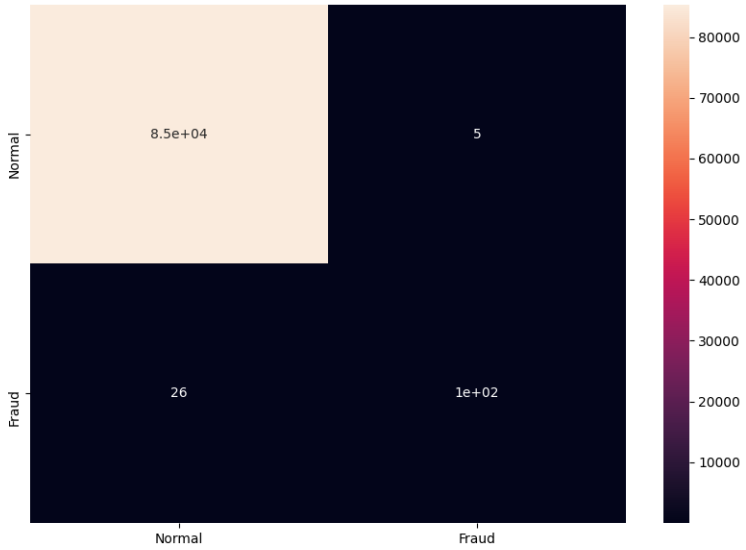
[L'immagine è presente per intero su Github.]

Il Gini Impurity di un nodo è la probabilità che un campione scelto a caso in un nodo di essere etichettato in modo errato se fosse etichettato dalla distribuzione dei campioni nel nodo.

Si calcola il valore usando:

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$$

Per un nodo n è 1 meno la somma su tutte le classi J(per un'attività di classificazione binaria è 2) della frazione di esempi in ogni classe al p_i quadrato.



Random Forest Accuracy: 99.96%			
	Normale	Fraudolenta	Macro avg
Precision	0.999695	0.954128	0.976912
Recall	0.999941	0.800000	0.899971
F1-score	0.999818	0.643068	0.935056
Support	85313	130	85443

7.2 RNA

Partendo dall'idea di replicare il funzionamento di una *rete di neuroni* celebrali, le **RNA/ANN** realizzano una unità artificiale, diversa in numero(minore) rispetto ai corrispettivi biologici. Sono metodi popolari per ragionare a basso livello, con molti dati disponibili.

Ogni rete è costituita da "*Neuroni*" che sono collegati a tutti i neuroni dello strato successivo, tramite archi **pesati**. Ogni neurone prende i valori pesati di titti i neuroni ad esso collegati e li somma aggiungendo un valore **bias**, e il risultato viene mutato tramite "***funzione di attivazione***" che verrà passato allo stato successivo, cosi fino al "*neurone* ***Output***".

La funzione di attivazione serve ad approssimare i dati in maniera più precisa, e a volte senza non si potrebbe proprio ottenere un risultato, basti pensare alle distribuzioni circolari.

Abbiamo utilizzato la funzione di attivazione **ReLu** e **Sigmoide**

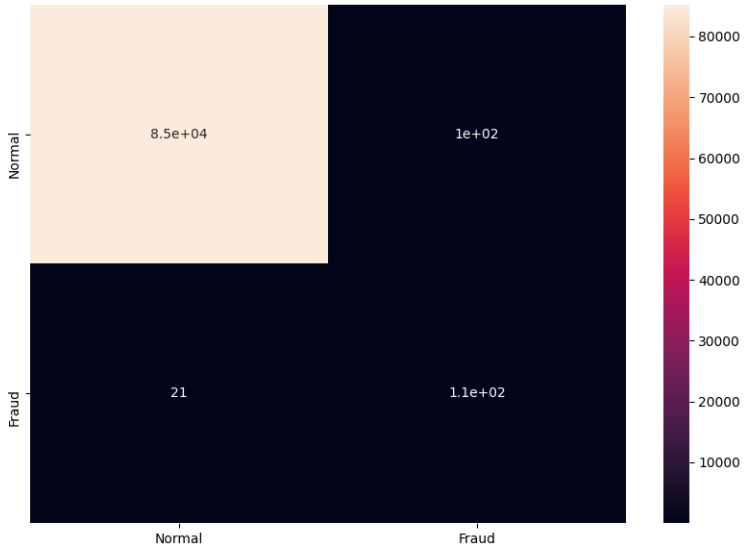
Per quanto riguarda la funzione Sigmoide, questa effettua un passaggio da **0** a **+1** in maniera graduale, con andamento a forma di "S", questa comprime quindi i valori (tra 0 e 1).

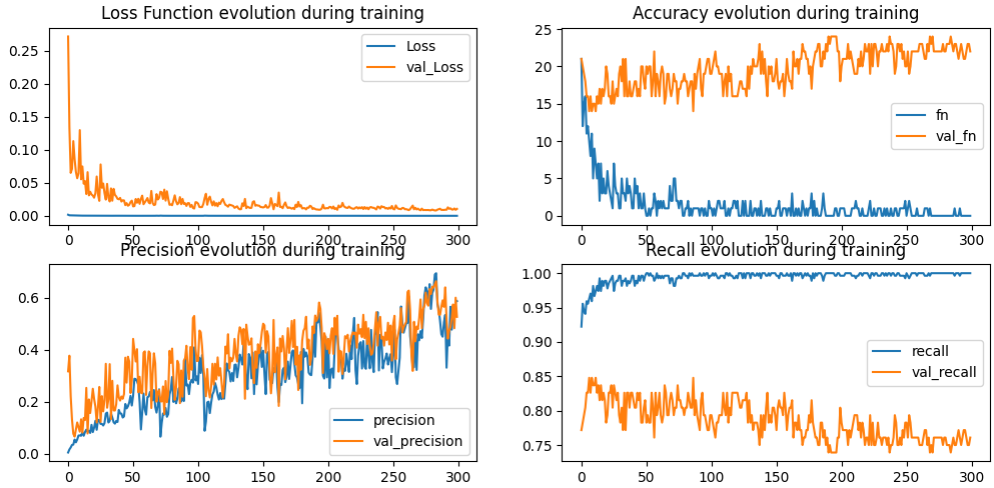
$$P(t) = \frac{1}{1 + e^{-t}}$$

La ReLu (rectifier linear unit), invece, è una funzione facile da calcolare, che appiattisce a 0 tutti i valori negativi e lascia tutto invariato per i positivi.

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise.} \end{cases}$$

Questo è utile perchè nei punti in cui abbiamo 0, la derivata sarà 0, nei positivi sarà 1.





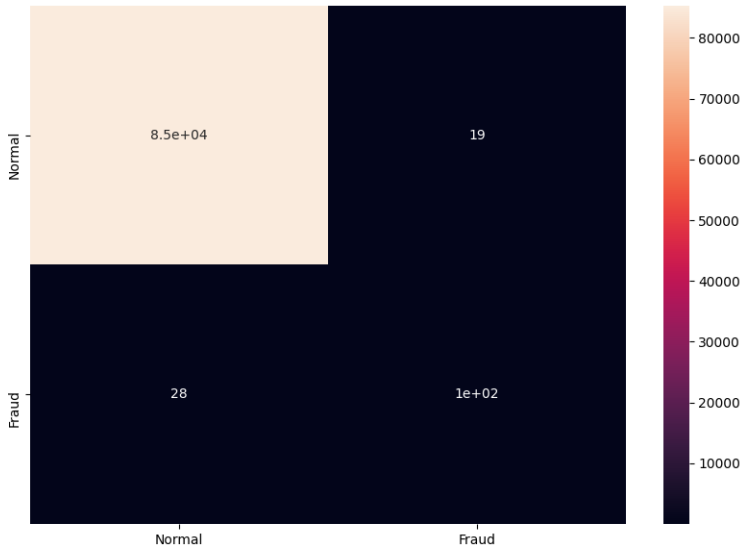
RNA Accuracy: 99.86%			
	Normale	Fraudolenta	Macro avg
Precision	0.999754	0.521531	0.760642
Recall	0.998828	0.838462	0.918645
F1-score	0.999291	0.643068	0.821179
Support	85313	130	85443

7.3 SVM

Il Support Vector Machine ha l'obiettivo di identificare l'**iperpiano** che meglio divide i vettori di supporto in classi. Per farlo esegue i seguenti step:
Cerca un iperpiano linearmente separabile o un limite di decisione che separa i valori di una classe dall'altro. Se ne esiste più di uno, cerca quello che ha margine più alto con i vettori di supporto, per migliorare l'accuratezza del modello.

Se tale iperpiano non esiste, SVM utilizza una mappatura non lineare per trasformare i dati di allenamento in una **dimensione superiore** (se siamo a due dimensioni, valuterà i dati in 3 dimensioni). In questo modo, i dati di due classi possono sempre essere **separati da un iperpiano**, che sarà scelto per la suddivisione dei dati.
I nuovi esempi sono mappati nell'iperpiano e la predizione della categoria alla quale appartengono viene fatta individuando il lato dell'iperpiano nel quale ricade.
L'algoritmo SVM ottiene la massima efficacia nei problemi di classificazione binari.

SVM Accuracy: 99.94%			
	Normale	Fraudolenta	Macro avg
Precision	0.999672	0.842975	0.921324
Recall	0.999777	0.784615	0.892196
F1-score	0.999725	0.812749	0.906237
Support	85313	130	85443



Boosting
Nell'ensemble learning, si combinano le predizioni di un certo numero di modelli appresi dal learner di base(ad esempio random forest).

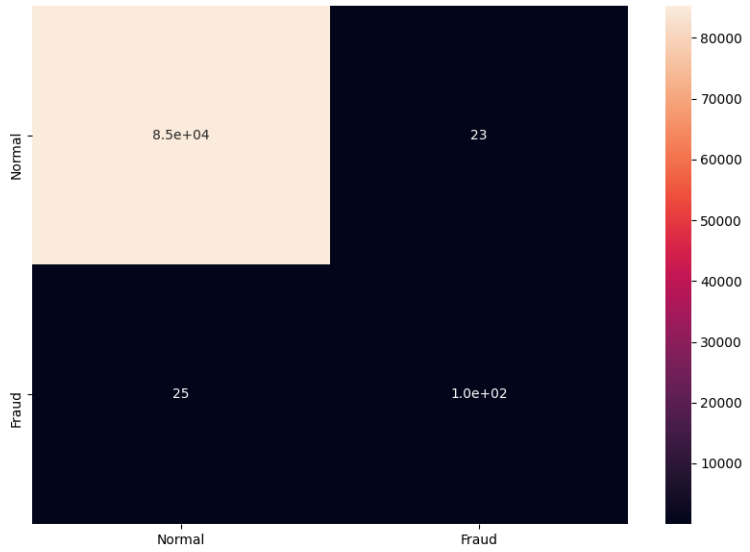
Nel Boosting, i modelli in squenza sono costruiti imparando dagli errori dei modelli precedenti.

7.4 AdABoost

L'Adaptive Boosting è un modello di ensemble boosting che utilizza alberi decisionale.
L'output del metaclassificatore (alberi decisionali) è dato dalla somma pesata delle predizioni dei singoli modelli.
Ogni qual volta un modello viene addestrato, ci sarà una fase di ripesaggio delle istanze.
L'algoritmo di boosting tenderà a dare un peso maggiore alle istanze misclassificate, nella speranza che il successivo modello sia più esperto su quest'ultime.

Sostanzialmente, ad ogni iterata calcola il tasso di errore ponderato dell'albero decisionale, ovvero il numero di predizioni sbagliate sul totale delle predizioni,che dipende dai pesi associati agli esempi nel dataset;
successivamente in base all'errore calcola il learning rate dell'albero decisionale.

maggiore è il tasso di errore di un albero, minore sarà il potere decisionale che l'albero avrà durante la predizione successiva;
Minore è il tasso di errore di un albero, maggiore sarà il potere decisionale assegnato all'albero durante la predizione successiva.



ADABoost Accuracy: 99.94%			
	Normale	Fraudolenta	Macro avg
Precision	0.999707	0.820312	0.910010
Recall	0.999730	0.807692	0.903711
F1-score	0.999719	0.813953	0.906836
Support	85313	130	85443

7.5 XGBoost

Un'altro metodo di classificazione è l'XGBoost(*eXtreme Gradient Boosting*) che è un'implementazione specifica del metodo **Gradient Boosting** che utilizza approssimazioni più accurate per trovare il miglior modello ad albero.

E' molto efficace grazie ad alcuni accorgimenti, tra cui:
calcolare le *derivate secondarie parziali della funzione* ***loss*** (simile al metodo di Newton), che fornisce maggiori informazioni sulla direzione dei gradienti e su come arrivare al minimo della funzione di perdita. Mentre il **Gradient Boosting** utilizza la funzione di perdita del modello base (ad es. Albero decisionale) per ridurre al minimo l'errore del modello complessivo, XGBoost *utilizza la derivata del 2 ° ordine come approssimazione.*

Utilizzare la **regolarizzazione** (L1 e L2)per riduce significativamente la varianza del modello (senza un sostanziale aumento del suo Bias); oltre alla formazione molto veloce e che può essere parallelizzata / distribuita tra i cluster.

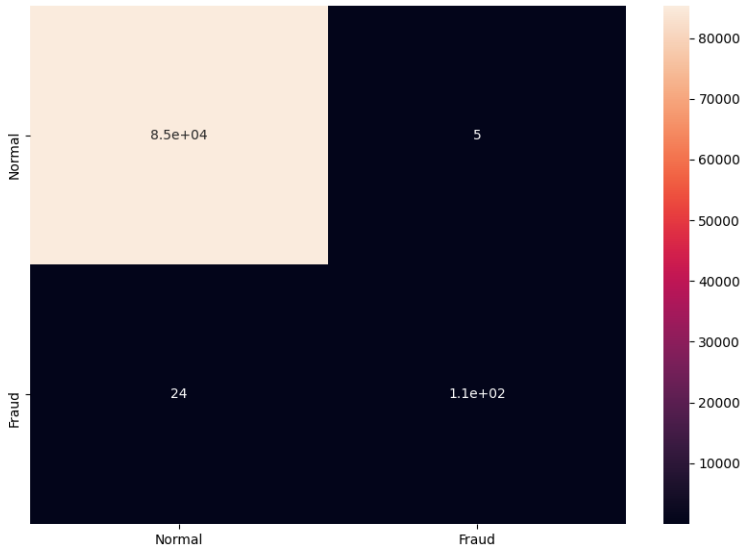
La formazione di un XGBoost è una procedura iterativa che calcola ad ogni passo la migliore suddivisione possibile per il k-esimo albero elencando tutte le possibili strutture ancora disponibili in quel punto del percorso.

Duarante la costruzione dell'albero XGBoost, si utilizza lo split dei figli tramite **Similarity Score**:

$$Similarity\ Score = \frac{(\sum residui)^2}{N° residui + \lambda}$$

sulla base dei risultati si crea un'ipotesi di suddivisione che va ad assegnare al figlio sinistro i valori sotto la media ottenuta, e al figlio destro quello maggiori, e si riprocede per i figli.

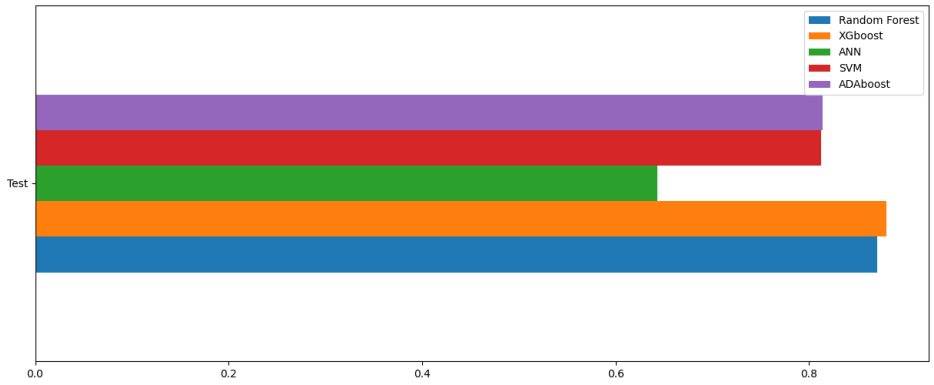
Andiamo successivamente ad utilizzare una tecnica di **Pruning** per migliorare ulteriormente le prestazioni dell'albero, che consiste nel rimuovere i rami che fanno uso di caratteristiche che hanno poca importanza (*riducendo quindi la complessità dell'albero*).



XGBoost Accuracy: 99.97%			
	Normale	Fraudolenta	Macro avg
Precision	0.999719	0.954955	0.977337
Recall	0.999941	0.815385	0.907663
F1-score	0.999830	0.879668	0.939749
Support	85313	130	85443

8 Risultati a confronto

Ora analizziamo i risultati dei vari modelli per capirne le potenzialità sul DataSet preso in esame, sappiamo già che essendo binario, alcuni metodi tenderanno ad essere più performanti.



Osservando questo grafico, la prima cosa che salta all'occhio, la **f1-score** di xGBoost e RandomForest, mentre l'ANN risulta essere la peggiore.
Ma dobbiamo entrare più nel dettaglio e vedere quanta precision riguarda le fraudolente in maniera precisa. Osservando anche i dati di ogni metodo, ci rendiamo subito conto che i valori di precision si aggirano tra lo *0.52%* e il *0.95%*, ma anche in questo caso il peggiore risulta sempre l'ANN e il migiore **xGBoost**.

Contatti

Progetto realizzato in *Python*, da:

- **Fiore Antonio Francesco**, 676538, *a.fiore102@studenti.uniba.it*
- **Losciale Gianluca**, 667283 , *g.losciale1@studenti.uniba.it*

Repository GitHub: https://github.com/AntoFlow/ICon-CCFraud
DataSet fornito da: Machine Learning Group - ULB.