UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea in Informatica

TESI DI LAUREA IN

SISTEMI INTELLIGENTI PER LA COMUNICAZIONE DIGITALE

ANALISI DELLA ROBUSTEZZA DI ALGORITMI DI DATA STREAM

Relatore:
Prof.ssa Gabriella Casalino

Laureando:
Alessandro Mastrorilli

ANNO ACCADEMICO 2020 – 2021

Sommario

Capitolo 1 – Introduzione

1.1 Data Stream

L'analisi di streaming dei dati è un'area di ricerca il cui obiettivo è quello di fornire nuova conoscenza da grandi quantità di dati continuamente generati.

I flussi di dati sono un caso particolare di Big Data, i quali sono caratterizzati da quattro dimensioni: Volume, Velocità, Varietà e Verità.

Possono essere prodotti da reti, sensori, transazioni online, social networks, learning analytics, ecc.

Come esempio di utilizzo, si immagini una fabbrica con 500 sensori che processano 10 KB di informazioni ogni secondo, in un'ora vengono processati circa 36 GB di informazioni e 432 GB al giorno.

Queste enormi informazioni devono essere analizzate in tempo reale per rilevare irregolarità o malfunzionamenti nel sistema per poi reagire rapidamente.

Il termine "flusso di dati" definisce una sequenza di dati che viene trasmessa a un sistema in maniera continua e mutevole.

A differenza di insiemi di dati statici con un numero fisso di dati, nei flussi di dati il numero di dati è in continuo aumento. Un sistema che gestisce flussi di dati può produrre costantemente enormi quantità di dati.

Si suppone che il volume della sequenza di dati è così grande che i campioni possono essere utilizzati alcune volte (o solo una volta) per l' analisi.

In questo modo si può procedere con l'analisi dei dati senza memorizzare l'intero flusso di dati in memoria, cosa che sarebbe impraticabile, perché richiederebbe memorie enormi.

I data stream soddisfano le seguenti caratteristiche:

- Flusso continuo di dati : elevata quantità di dati in un flusso infinito
- Concept Drifting : i dati cambiano o si evolvono nel tempo
- Volatilità dei dati : il sistema non memorizza i dati ricevuti[1]

Quando i dati vengono analizzati o scartati, la gestione e l'elaborazione dei flussi di dati solleva nuove sfide e nuovi problemi di ricerca.

Solitamente non è possibile archiviare semplicemente i dati in arrivo in un tradizionale database, ma i flussi di dati devono essere elaborati online[2].

Gli algoritmi tradizionali sono in grado di elaborare questo tipo di dati perché estraggono modelli dai dati considerando proprietà globali , piuttosto che considerare quelle locali.

Inoltre, richiedono la disponibilità dell'intero set di dati di addestramento.

L' analisi e la classificazione dei flussi di dati è diventata sempre più importante[3].

La natura intrinseca dei dati di stream richiede lo sviluppo di algoritmi in grado di eseguire elaborazioni veloci e incrementali dei dati, gestendo opportunamente i limiti di tempo e di memoria.

Un approccio per analizzare i flussi di dati sfrutta una generazione incrementale di pattern di informazioni che rappresentano in maniera sintetica tutti i record di dati analizzati in passato e che si evolvono progressivamente man mano nel tempo con nuovi record di dati.

Algoritmi incrementali e online sono utili per gestire l'arrivo continuo di dati variabili nel tempo in tempi rapidi.

Inoltre gestiscono flussi potenzialmente illimitati, in quanto incorporano continuamente informazioni nel loro modello [4][5].

1.2 Classificazione di flussi di dati

Un approccio comune per la classificazione del flusso di dati si basa su finestre scorrevoli dove un sottoinsieme di dati viene considerato per la creazione e l'evoluzione di un classificatore [3]. Il blocco di dati contiene i dati più recenti disponibili da un flusso; quando arrivano nuovi dati, il blocco cambia sostituendo i dati più vecchi con i più nuovi o sostituendo tutti i dati non appena ci sono sufficienti nuovi dati per riempire l'intero blocco.

L' approccio delle finestre scorrevoli è molto generale, in quanto consente l'applicazione della maggior parte dei metodi di progettazione dei classificatori. Tuttavia, sono vincolati a utilizzare solo una parte del flusso [2].

Diversi approcci possono essere adottati per progettare modelli di classificazione da flussi di dati organizzati in blocchi, tra cui i metodi d'insieme [6,7] e gli schemi in evoluzione [8](spesso usati insieme[9]).

In particolare, schemi in evoluzione per classificatori data-driven possono essere usati per addestrare i classificatori su blocchi di dati tenendo conto delle conoscenze acquisite nei blocchi precedenti.

Molti metodi di apprendimento automatico classici sono stati adattati a schemi in evoluzione.

Yang et al. hanno adottato classificatori naïve bayesiani per il loro basso costo di costruzione e la facilità di manutenzione incrementale [10]. Ikonomovska et al. propongono un metodo incrementale per apprendere modelli di regressione e alberi da dati di flusso [11].

Zhang & Zhou propongono un nuovo parametro, chiamato stima di trasferimento, per adattare dinamicamente la stima della classe e regolare un classificatore di conseguenza [12].

Il metodo è stato applicato all' algoritmo Online Expectation-Minimization per la classificazione di sequenze di dati non stazionari [13].

Pang et al. hanno sviluppato un'analisi discriminante lineare incrementale che è in grado di far evolvere un autospazio discriminante su flussi di dati veloci e grandi [14].

Guarracino et al. propongono la classificazione incrementale regolarizzata con autovalore generalizzato (I-ReGEC), che è uno schema evolutivo di un algoritmo di apprendimento supervisionato per regolare dinamicamente le funzioni kernel utilizzate per la classificazione [15].

Annapoorna et al. hanno implementato algoritmi Random Forest con campionamento casuale stratificato e filtraggio Bloom al fine di velocizzare l'addestramento in presenza di dati ad alta velocità [16].

Gomes et al. hanno adattato l'algoritmo Random Forest per il contesto del data stream mining con ricampionamento e operatori adattivi che possono far fronte a diversi tipi di derive concettuali senza un' ottimizzazione complessa [17].

Capitolo 2 – Disordine bipolare

I cambiamenti d'umore sono comuni nella vita , in particolare di fronte a eventi stressanti.

Tuttavia, quando gli sbalzi d'umore sono evidenti e persistenti a tal punto da provocare notevole angoscia o menomazione , potrebbe esserci un disturbo affettivo.

I disturbi affettivi possono essere classificati lungo uno spettro definito dall' entità e dalla gravità dell'aumento dell' umore.

Il disturbo bipolare è una grave malattia mentale con un alto tasso di recidiva che colpisce più del 2% della popolazione mondiale [18].

Nel corso di questa malattia cronica, ci sono fluttuazioni tra diverse fasi dell' umore che vanno dalla depressione a casi maniacali(o entrambi gli stati d' animo)[19].

Gli individui con disturbo unipolare presentano solo episodi depressivi e quelli con disturbo bipolare II o I mostrano episodi di innalzamento dell'umore sempre più pronunciati.

La principale differenza tra i disturbi bipolari I e II risiede nella gravità di episodi maniacali causati da ogni tipo.

Un individuo con disturbo bipolare I vivrà un episodio maniacale completo , mentre un individuo con disturbo bipolare II sperimenterà solo un episodio ipomaniacale.

Il disturbo bipolare è una malattia episodica che dura tutta la vita con un decorso variabile che spesso può provocare una compromissione funzionale e cognitiva causando una riduzione della qualità della vita.

Il disturbo bipolare, precedentemente noto come malattia maniaco depressiva, è un grave disturbo cronico dell'umore caratterizzato da episodi di mania, ipomania ed

episodi alternati o intrecciati di depressione come mostrato nella figura sottostante :

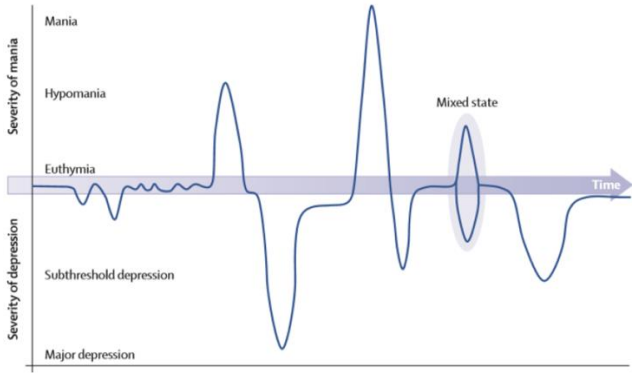

Figura 1 - Grafico che mostra la progressione del disturbo bipolare

I disturbi bipolari sono classificati secondo il decorso longitudinale, che è spesso caratterizzato dalla presenza di sintomi sotto-soglia.

Anche se il disturbo bipolare I potrebbe sembrare avere un'evoluzione più tortuosa e una prognosi più grave del disturbo bipolare II a causa della gravità trasversale dei sintomi, il disturbo bipolare II ha un'alta frequenza di episodi, alti tassi di comorbidità psichiatriche e comportamenti suicidi ricorrenti che compromettono la qualità della vita [20].

Nelle indagini sulla salute mentale mondiale (OMS), il disturbo bipolare è stato classificato come la malattia con il secondo maggiore effetto sui giorni fuori ruolo. Poiché il disturbo bipolare viene diagnosticato principalmente nella giovane età adulta, la malattia colpisce la popolazione economicamente attiva e, pertanto, comporta costi elevati per la società.

L'insorgenza della mania in età avanzata potrebbe essere indicativa di una comorbidità medica latente.

A causa della ricorrenza e della cronicità del disturbo bipolare, non solo è fondamentale il trattamento acuto per la gestione degli episodi ma anche gli approcci farmacologici e psicologici per la prevenzione di ulteriori episodi sono importanti.

Il rischio di un nuovo episodio può essere ridotto significativamente da un trattamento appropriato.

Tuttavia, la frequenza delle visite di controllo è solitamente insufficiente per fornire un intervento farmacologico precoce all'inizio dell'episodio, e i pazienti da soli non sono consapevoli della necessità del trattamento.

Pertanto, negli ultimi anni, c'è un crescente interesse per il monitoraggio in tempo reale degli episodi di disturbo bipolare con l'uso di smartphone.

Questa novità ha aperto nuove opportunità nel monitoraggio dello stato affettivo dei pazienti con dei database.

Lavori recenti, vedi ad esempio [21] mostrano che i dati oggettivi basati su smartphone diventano un valido strumento in grado di rilevare episodi ricorrenti di sintomi depressivi e maniacali.

Mania e ipomania

Gli episodi di mania o ipomania rappresentano stati di umore elevato e aumento della spinta motoria che durano in un tempo finito e differiscono per gravità e lunghezza.

Anche se un episodio maniacale compromette il rendimento sociale o lavorativo e possa comprendere sintomi psicotici o persino portare a un ricovero in ospedale, in un episodio di ipomania, un disturbo nel rendimento può essere visto dagli altri ma in genere non causa gravi danni o non richiede il ricovero in ospedale.

In alcuni casi di ipomania, il rendimento professionale potrebbe anche migliorare transitoriamente a causa di maggiore produttività e buon umore.

Circa il 75% dei pazienti con un episodio maniacale acuto presenta sintomi psicotici.

I deliri possono essere coerenti con l'umore nella mania, con individui che mostrano grandiosità, megalomania o ideazioni messianiche.

Tuttavia, la psicosi incoerente con l'umore non è rara, si ha la percezione di essere invidiati, in pericolo e perseguitati dai nemici.

Le caratteristiche psicotiche, anche in casi di umore incoerente , non escludono la diagnosi di disturbo bipolare [22].

Un episodio ipomaniaco è definito nel DSM-51 come persistente per almeno 4 giorni consecutivi, mentre un episodio maniacale dura almeno 1 settimana.

Il cutoff temporale di 4 giorni non è rispecchiato da un chiaro cutpoint nei dati riportati, e alcuni ricercatori hanno suggerito che questo periodo dovrebbe essere accorciato a 2 giorni in modo che i pazienti classificati con depressione unipolare possano essere diagnosticati con un disturbo bipolare.

Lo studio transculturale BRIDGE mirava ad accertare la frequenza dei sintomi del disturbo bipolare secondo il DSM-5 e i criteri molto più ampi di specificazione bipolare

in pazienti che cercano un trattamento per un episodio depressivo maggiore.

9034 (16%) su 5635 pazienti con diagnosi di disturbo depressivo maggiore soddisfacevano i criteri DSM-IV-TR per il disturbo bipolare mentre 2647 (47%) soddisfacevano i criteri bipolari specifici.

Così, il 31% dei pazienti con una diagnosi di disturbo depressivo grave aveva sintomi ipomaniacali o maniacali sottosoglia con i criteri più ampi [22].

Senza un marcatore oggettivo, non è chiaro se questo risultato rappresenta una sottodiagnosi del disturbo bipolare o una diagnosi falsamente positiva.

In particolare, basando la diagnosi su stati irritabili molto comuni e non specifici e su rapidi cambiamenti d'umore ,si potrebbe rischiare di incorrere ai disturbi di personalità borderline e correlati e ai disturbi dell'adattamento.

Depressione

All'esordio, la maggior parte dei pazienti con disturbo bipolare si presenta con un episodio depressivo che differisce sottilmente dalla depressione unipolare.

I criteri del DSM-5 per un episodio depressivo grave sono gli stessi per la depressione bipolare e unipolare, e la gravità dell'episodio è valutata con le stesse scale: la Hamilton Depression Rating Scale (HDRS) o la Montgomery-Asberg Depression Rating Scale (MADRS).

Sebbene la depressione bipolare o unipolare non abbia caratteristiche patognomoniche, i ricercatori hanno descritto alcune caratteristiche cliniche che sono utili per discriminare la depressione bipolare e quella unipolare.

La depressione bipolare di solito ha un'età di insorgenza, ha episodi più frequenti e di durata più breve, ha un inizio e una compensazione brusca,è legata all'abuso di sostanze in comorbilità, è scatenata da fattori di stress nelle fasi iniziali e ha più rischio post-partum.

I sintomi atipici - come ipersonnia, labilità e instabilità del peso - sono comuni anche nella depressione bipolare, essendo riportati nel 90% degli episodi, ma sono descritti solo nella metà degli episodi depressivi unipolari.

Psicosi, ritardo psicomotorio e catatonia sono anche più caratteristici della depressione bipolare, mentre i disturbi somatici sono più frequenti nella depressione unipolare. Una storia familiare di mania è anche un indicatore rilevante di depressione bipolare.

.

.

Capitolo 3 – Esperimenti

3.1 Apprendimento supervisionato

Nel task di apprendimento supervisionato si considera un insieme di esempi di training descritti in termini di feature di input e fetaure target.

L' obiettivo è quello di predire il valore della feature target per nuovi esempi dati i loro valori di input.

Il task di apprendimento supervisionato è detto :

- classificazione se le variabili di target Y_i sono discrete

- regressione se le variabili di target Y_i sono continue.

3.2 Algoritmi utilizzati

3.2.1 K – Nearest – Neighbor (KNN)

L' obiettivo dell'algoritmo del K-Nearest-Neighbor è quello di trovare un numero predefinito di campioni di addestramento più vicini per distanza al nuovo punto e predire l' etichetta target da questi [23].

Il numero di campioni può essere una costante definita dall' utente (k-nearest-neighbor), o variare in base alla densità locale dei punti(radius-based neighbor).

La distanza può essere definita da qualsiasi metrica: la distanza euclidea standard è la scelta più comune.

Nonostante la sua semplicità, il nearest neighbors ha avuto successo in un gran numero di problemi di classificazione e regressione, tra cui cifre scritte a mano e scene di immagini satellitari.

Essendo un metodo non parametrico, ha spesso successo in situazioni di classificazione in cui il confine della decisione è molto irregolare.

3.2.2 Naive Bayes

I metodi Naive Bayes sono un insieme di algoritmi di apprendimento supervisionato basati sull'applicazione del teorema di Bayes con l'assunzione "ingenua" dell'indipendenza condizionata tra ogni coppia di feature dato il valore della variabile target[24].

Il teorema di Bayes afferma la seguente relazione , data la variabile di target y e il vettore di feature dipendenti x1,…xn: $P(y|x_1,\ldots,x_n) = \frac{\mathrm{P(y)P}(x_1,\ldots,x_n|\mathrm{y})}{\mathrm{P}(x_1,\ldots,x_n)}$

A questo punto è necessario esplicitare l' ipotesi "ingenua" di indipendenza condizionata:$\mathrm{P}(x_i|\mathrm{y},x_1,..,x_{i-1},x_{i+1},\ldots,x_n) = \mathrm{P}(x_i|y)$

Per ogni i , l' assunzione è semplificata nella seguente maniera :

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

Dal momento che $\mathrm{P}(x_1,\ldots,x_n)$ è una costante data in input , si può utilizzare la seguente regola di classificazione :

$$P(y|x_1,\ldots,x_n) \propto \mathrm{P(y)}\prod_{i=1}^{n}\mathrm{P}(x_i|\mathrm{y}) \longrightarrow \hat{y} = argmax_y P(y)\prod_{i=1}^{n}\mathrm{P}(x_i|\mathrm{y})$$

A questo punto è possibile usare la stima Maximum A Posteriori (MAP) per stimare P(y) e $\mathrm{P}(x_i|\mathrm{y})$.

I diversi classificatori naive Bayes differiscono principalmente per le assunzioni che fanno riguardo alla distribuzione P(xi|y).

Nonostante le assunzioni troppo semplificate , i classificatori Naive Bayes sono abbastanza performanti in alcune situazioni del mondo reale come la classificazione dei documenti e il filtraggio dello spam.

Possono essere estremamente veloci rispetto a metodi più sofisticati.

Il disaccoppiamento delle distribuzioni delle feature target condizionali dimostra che ogni distribuzione può essere stimata in maniera indipendente come una distribuzione monodimensionale.

D’ altra parte , anche se il Naive Bayes è conosciuto come un classificatore decente , è noto per essere un pessimo stimatore, per cui i risultati probabilistici non devono essere presi troppo sul serio.

3.2.3 Decision Tree

L'algoritmo Decision Tree (DT) è una tecnica per creare un modello predittivo che utilizza la rappresentazione ad albero per risolvere problemi di classificazione.

La struttura dei dati ad albero è caratterizzata da nodi interni (nodi di decisione) per testare gli attributi dei dati, da bordi per ramificare i risultati del test e foglie che rappresentano le etichette di classe del problema.

L'algoritmo per la costruzione di un albero decisionale identifica i modi per dividere i dati di addestramento determinando l'attributo dei dati più informativo per dividere i dati ad ogni nodo.

La procedura di costruzione viene applicata durante la fase di addestramento e inizia l'esecuzione dalla radice dell'albero che contiene l'intero set di dati.

La procedura trova prima il miglior attributo di dati che divide i dati di addestramento alla radice in sottoinsiemi.

Di conseguenza, tutti i possibili risultati del test assegnati alla radice, sono rappresentati come gli archi in uscita dalla radice.

In seguito, il metodo di costruzione genera ricorsivamente nuovi nodi dividendo ogni sottoinsieme di dati che passa attraverso ogni ramo.

Il metodo non estende ulteriormente un ramo, se raggiunge una profondità massima o quando il restante sottoinsieme in questo ramo è quasi interamente della stessa classe.

L'algoritmo genera invece una foglia etichettata con questa etichetta di classe maggioritaria e che rappresenta la decisione presa dal classificatore.

Un certo numero di criteri e misure di qualità possono essere usati per determinare l'attributo migliore su cui dividere il dataset disponibile ad ogni nodo, così come il test basato su quell'attributo.

Per esempio, l'indice di Gini [25] e l’ information Gain [26] possono essere usati per il task di classificazione.

Per classificare il nuovo esempio in arrivo, l'albero decisionale prima lo testa contro il test alla radice usando il valore del suo attributo.

In base al risultato del test, l'esempio segue uno dei rami e si sposta al nodo successivo.

Questo processo è ricorsivo e viene ripetuto per ogni nodo fino a quando non si raggiunge una foglia che fornisce la classificazione all'esempio.

3.2.3.1 Hoeffeding Tree Classifier

Hoeffding Tree (HT) [27] è un metodo ad albero decisionale all'avanguardia progettato per l'apprendimento da flussi di dati massicci.

È stato proposto per la prima volta da Domingos et.al in [27] e il suo nome deriva dal limite di Hoeffding che è usato per dividere le decisioni nell'albero.

L'idea principale dietro l'algoritmo è che un piccolo sottoinsieme delle istanze di addestramento può essere sufficiente per scegliere il miglior attributo per la divisione in un dato nodo.

L' Hoeffding bound supporta matematicamente questa idea ed è spiegato come segue.

Si suppone che la variabile casuale r, che ha range R, sia la misura di selezione degli attributi che è usata per scegliere il miglior attributo di divisione in ogni nodo dell'albero di Hoeffding.

Si suppone anche che N osservazioni indipendenti di r siano state analizzate e che il valore stimato della loro media sia $\bar{r}$.

Di conseguenza, l' Hoeffding bound assicura che con probabilità 1-δ , la vera media di r è almeno $\bar{r}$-ε , dove δ è specificato dall'utente ed ε è :$\varepsilon = \sqrt{\frac{R^2 ln(\frac{1}{\delta})}{2n}}$

Concretamente, il limite di Hoeffding viene applicato quando ogni nuovo esempio arrivato raggiunge un nodo di frontiera, che non è una foglia e non è ancora stato diviso durante l'addestramento dell'albero.

Al nodo frontiera, il test di Hoeffding è soddisfatto quando c'è una confidenza di 1-δ che l'attributo ottimale sia scelto dal metodo di misurazione della qualità. A questo punto, la divisione è fatta e il nodo di frontiera diventa un nodo interno.

3.2.4 Random Forest

Random forest (RF)[28] è un algoritmo ensemble sviluppato da Breiman ed è ampiamente utilizzato nell'apprendimento in batch o non-stream per la regressione, la classificazione e altri compiti.

L'algoritmo sviluppa un certo numero di alberi di decisione e li combina in un unico modello.

Per i problemi di classificazione, la previsione finale fatta dall'algoritmo si basa sulla maggioranza dei voti di tutti gli alberi dove ogni singolo albero fa una previsione di classe.

L'idea fondamentale dietro RF è quella di costruire una foresta di alberi che hanno basse correlazioni tra loro o sono non correlati, al fine di evitare l'overfitting e fare una previsione robusta.

Mentre alcuni alberi nella foresta sono sensibili al rumore e predicono male, molti altri alberi diversi possono essere performanti.

Finché gli alberi non sono correlati, la pluralità dei loro voti è meno incline all'overfitting e l'errore atteso è più basso.

Per diminuire le correlazioni tra gli alberi e aumentare la diversità nella foresta, l'algoritmo usa due metodi per costruire ogni singolo albero: selezione casuale delle features e Bagging (aggregazione bootstrap) [29].

Il Bagging [29] è un algoritmo d'insieme progettato per generare un certo numero di set di dati diversi, ciascuno di N record da un dato set di dati di addestramento di N record.

L'algoritmo genera i set di dati campionando casualmente i record con sostituzione. Quindi, ogni record del dataset originale può essere ripetuto in ogni campione bootstrapped K volte, dove P(K = k) rappresenta la distribuzione binomiale [30].

Questa distribuzione binomiale tende a una distribuzione Poisson(1), se la dimensione N del campione di bootstrapping è grande.

I classificatori (alberi decisionali) nell'algoritmo Bagging, anche nella RF, sono poi addestrati ciascuno su uno dei set di dati generati invece che sul set di dati originale.

La selezione casuale delle features è usata durante la crescita degli alberi e la suddivisione di ogni nodo.

Nell'algoritmo standard dell'albero decisionale, l'intero set di features viene ispezionato ad ogni nodo per trovare la feature più importante da dividere.

Al contrario, per dividere ogni nodo nella foresta casuale, viene considerato un sottoinsieme casuale di m<M features , dove M è il numero totale di features.

L'uso del bagging e la selezione casuale delle features rendono l'algoritmo RF più potente di semplici insiemi di alberi e meno incline all'overfitting se la foresta ha abbastanza alberi.

È stato anche dimostrato che l'algoritmo Random Forest è veloce da addestrare ed efficace quando testato su diversi set di dati [28].

Inoltre, l'algoritmo è in grado di gestire tutte le forme di dati, compresi i dati con valori mancanti e produce previsioni molto accurate.

3.2.4.1 Adaptive Random Forest

Adaptive Random Forest (ARF) [31] è un'estensione dell'originale Random Forest progettato per gestire flussi di dati in evoluzione.

L'algoritmo adattivo combina tecniche da Random Forest con metodi usati per far fronte dinamicamente a diversi tipi di cambiamenti concettuali.

La foresta casuale standard è un insieme batch addestrato utilizzando tutti i dati statici disponibili. RF non è appropriato per l'apprendimento da flussi sequenziali di dati che possono arrivare continuamente.

Quindi, per far funzionare l'algoritmo RF in modalità online sono necessari degli adattamenti .

Uno di questi è che i learner di base in ARF sono alberi di Hoeffding, che sono in grado di apprendere da flussi di dati massicci al contrario degli alberi di decisione standard usati in RF.

Nella RF originale, il Bagging [31] e selezione casuale delle features, sono applicati per diminuire le correlazioni tra i modelli di base e aumentare la diversità dell'insieme.

Allo stesso modo, ARF aggiunge diversità attraverso la selezione di un sottoinsieme casuale di features per le suddivisioni dei nodi mentre produce ogni HoeffdingTree nella foresta.

Questo viene eseguito modificando l'algoritmo HoeffdingTree che considera tutti gli attributi per trovare il migliore per la divisione.

Nell'apprendimento online, non è possibile utilizzare il metodo non-stream Bagging per prelevare campioni casuali con sostituzione dai dati di addestramento originali.

Questo perché il metodo ha bisogno di più passaggi sui dati, che hanno una dimensione sconosciuta nell'impostazione online ed è costantemente in aumento.

Tuttavia, ARF include un efficace algoritmo di ricampionamento che si basa sull'algoritmo Online Bagging, un processo di aggregazione bootstrap per il flusso di dati proposto in [32].

Nell'algoritmo di bagging online [32], quando arriva ogni nuova istanza di addestramento e per ogni modello di base nell'insieme, l'istanza corrente viene utilizzata per addestrareil modello di base W volte di seguito.

Questo significa pesare l'istanza con un valore W dove w è un numero casuale generato dalla distribuzione Poisson ($\lambda = 1$).

Per cui , il bagging online simula il campionamento con sostituzione, il Bagging originale utilizza Poisson(1) per la ponderazione delle istanze.

Questo si basa sul fatto che la distribuzione binomiale usata in non-stream bagging tende ad una Poisson(1) quandola dimensione del flusso di dati è infinita, come nel caso dell'apprendimento online.

Tuttavia, il cambiamento nella strategia ARF per il ricampionamento riguarda l'aumento del valore della diversità nella distribuzione Poisson a $\lambda = 6$.

Di conseguenza, questo attribuisce una diversa gamma di pesi ai campioni e quindi aumenta la diversità dello spazio di input all'interno dell'ensemble.

Una sfida comune nell'apprendimento online dove i dati sono raccolti nel tempo, è che l' insieme di dati può derivare in modo imprevedibile, il che ha un impatto negativo sulle performance del modello predittivo e lo rende obsoleto nel tempo.

Tuttavia, l'obiettivo di ARF è quello di includere meccanismi per far fronte a diversi tipi di derive concettuali.

Concretamente, ARF utilizza un rilevatore di derive per ogni albero nell' ensemble per monitorare gli ammonimenti e le derive.

Non appena viene rilevato un avvertimento in un albero, l'algoritmo crea un albero di background e inizia l'addestramento.

L'albero di background può essere usato in seguito per le previsioni al posto dell'albero attivo se l'avvertimento diventa una deriva.

Questa strategia differisce da dall'approccio predefinito che resetta gli alberi di base immediatamente dopo aver rilevato una deriva e li usa senza essere pre-addestrati su nessuna istanza, il che può influire negativamente sulla predizione del modello.

3.2.5 Learning Vector Quantization

Il Learning Vector Quantization (LVQ), originariamente proposto da Kohonen [33,34] e conosciuto con il nome LVQ1, è un metodo di apprendimento supervisionato online.

Sono state proposte molte variazioni sullo schema di base di LVQ1, tra cui LVQ2.1 e LVQ3 [34,35], GLVQ [36] e RSLVQ [37,38], con lo scopo di ottenere una migliore capacità di generalizzazione.

Durante l'apprendimento, i campioni di dati e le loro etichette di classe sono presentati in modo sequenziale, o cosiddetto 'on-line'.

Da un insieme di vettori prototipo, definiti nello stesso spazio, questo viene determinato e aggiornato in modo tale che se l'etichetta di classe coincide con l'etichetta di classe del campione di dati, il prototipo sia attratto dai dati, altrimenti respinto.

Si suppone che i dati, che portano etichette di classi diverse, siano distribuiti intorno ad un numero specificato di prototipi.

Si noti che ci può essere più di un prototipo per classe, permettendo un buon adattamento dei prototipi ai dati che contengono confini di classe complessi.

Dopo l'addestramento, la classificazione viene fatta determinando il più vicino tra tutti i prototipi e restituendo l'etichetta di classe corrispondente a questo prototipo vincente.

I confini decisionali tra i prototipi possono anche essere considerati come la tassellazione di Voronoi dello spazio delle features.

Le variazioni degli algoritmi LVQ differiscono principalmente in quali prototipi specifici sono aggiornati (per esempio solo i prototipi in conflitto più vicini o il prototipo più vicino con etichetta corrispondente e il più vicino con etichetta in conflitto) e come questi prototipi sono aggiornati.

La struttura generica di un algoritmo LVQ può essere espressa nel modo seguente:

$$\mathfrak{w}_l^{\mu} = \mathfrak{w}_l^{\mu-1} + \Delta\mathfrak{w}_l^{\mu} = \mathfrak{w}_l^{\mu-1} + \frac{\eta}{N} f(\{\mathfrak{w}_i^{\mu-1}\}, \xi^{\mu}, \sigma^{\mu}, \dots)(\xi^{\mu} - \mathfrak{w}_l^{\mu-1})$$

Con l ,i = 1,…c ,μ = 1,2…

Dove :

- $\mathfrak{w}_l^{\mu}$ è il prototipo di $\mathfrak{w}_l$ (della classe l) al passo temporale μ
- η : è il tasso di apprendiemnto
- N : è la dimensionalità del sistema.

3.2.5.1 Robust Soft Learning Vector Quantization

Le varianti LVQ come LVQ2.1 o Learning From Mistakes (LFM) [39] applicano aggiornamenti con forza di aggiornamento 0 o 1, indicato come un aggiornamento duro o nitido.

Robust Soft LVQ utilizza la distanza relativa tra un campione di dati e i prototipi per ammorbidire questa forza di aggiornamento. La formula di aggiornamento di RSVLQ, come definita da Seo e Obermayer [38] è la seguente:

$$\begin{aligned} \boldsymbol{w}_{\tilde{l}}^{\mu} &= \boldsymbol{w}_{\tilde{l}}^{\mu-1} + \Delta\boldsymbol{w}_{\tilde{l}}^{\mu} \\ &= \boldsymbol{w}_{\tilde{l}}^{\mu-1} + \tilde{\eta} \begin{cases} \left(P_l(\tilde{l}|\boldsymbol{\xi}^{\mu}) - P(\tilde{l}|\boldsymbol{\xi}^{\mu})\right) \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_{\tilde{l}}^{\mu})}{\partial \boldsymbol{w}_{\tilde{l}}^{\mu}} & \text{if } l = \sigma^{\mu} \\ -P(\tilde{l}|\boldsymbol{\xi}^{\mu}) \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_{\tilde{l}}^{\mu})}{\partial \boldsymbol{w}_{\tilde{l}}^{\mu}} & \text{if } l \neq \sigma^{\mu} \end{cases} \end{aligned}$$

Figura 2 - forrmula di aggiornamento di RSVLQ

In questo caso $P_l(\tilde{l}|\xi^{\mu})$ e $P(\tilde{l}|\xi^{\mu})$ sono assegnazioni probabilistiche :

$$P_l(\tilde{l}|\boldsymbol{\xi}^{\mu}) = \frac{p(\tilde{l})\exp\left(f(\boldsymbol{\xi}, \boldsymbol{w}_{\tilde{l}}^{\mu})\right)}{\sum_{\tilde{l}=\sigma^{\mu}} p(\tilde{l})\exp\left(f(\boldsymbol{\xi}, \boldsymbol{w}_{\tilde{l}}^{\mu})\right)}$$

$$P(\tilde{l}|\boldsymbol{\xi}^{\mu}) = \frac{p(\tilde{l})\exp\left(f(\boldsymbol{\xi}, \boldsymbol{w}_{\tilde{l}}^{\mu})\right)}{\sum_{\tilde{l}} p(\tilde{l})\exp\left(f(\boldsymbol{\xi}, \boldsymbol{w}_{\tilde{l}}^{\mu})\right)}$$

Figura 3 - assegnazioni di $\mathbf{P_l}(\tilde{\mathbf{l}}|\boldsymbol{\xi}^{\mu})$ e $P(\tilde{\mathbf{l}}|\boldsymbol{\xi}^{\mu})$

$P_l(\tilde{l}|\xi^{\mu})$ descrive la probabilità a posteriori che il campione di dati $\{\xi\mu, \sigma\mu\}$ sia assegnato al prototipo $\mathfrak{w}_{\tilde{l}}$ della classe l , dato che il campione di dati è stato generato nella classe corretta.

$P(\tilde{l}|\xi^{\mu})$ descrive la probabilità a posteriori che il campione di dati sia assegnato al prototipo $\mathfrak{w}_{\tilde{l}}$ di tutti i prototipi di tutte le classi.

$f(\xi, \mathfrak{w}_{\tilde{l}}^{\mu})$ descrive la distribuzione presunta dei dati intorno ai prototipi in modo tale che $K(\tilde{l})\exp(f(\xi, \mathfrak{w}_{\tilde{l}}^{\mu}))$ dà la probabilità che il vettore di dati ξ^{μ} sia assegnato al prototipo $\mathfrak{w}_{\tilde{l}}$

3.2.6 Very Fast Decision Rules

Il Very Fast Decision Rules (VFDR) [40] è un classificatore di apprendimento incrementale di regole.

Il processo di apprendimento di VFDR è simile a quello di Hoeffding Tree, ma invece di un albero usa una collezione di regole.

Il nucleo di VFDR è costituito dalle sue regole che mirano a creare un classificatore altamente interpretabile grazie alla loro natura.

Ogni regola è una congiunzione di condizioni basate sui valori degli attributi e la struttura per mantenere statistiche sufficienti.

Le statistiche sufficienti determinano la classe prevista dalla regola:

IF $att_1<1att1<1$ and $att_2=0att2=0$ THEN class 0.

3.2.7 Perceptron

Nell'apprendimento automatico, il percettrone è un tipo di classificatore binario che mappa i suoi input x in un valore di output f(x) calcolato in questo modo[41] :

f(x) = $\chi(< w, x > + b)$

dove:

- w è un vettore di pesi con valori reali
- L' operatore < .,.> è il prodotto scalare
- b è il 'bias' (una costante)
- χ (y) è la funzione di output.

Le scelte più comuni di χ (y) sono :

- χ (y) = sign(y)
- χ (y) = y$\Theta(y)$
- χ (y) = y

dove :

- $\Theta(y)$ è la funzione di Heaviside.
- Il primo caso corrisponde a un classificatore binario

Modificando il vettore dei pesi w, è possibile modulare l'output di un percettrone, con lo scopo d'ottenere delle proprietà di apprendimento o di memorizzazione ; le capacità computazionali di un singolo percettrone sono tuttavia limitate, e le prestazioni che è possibile ottenere dipendono fortemente sia dalla scelta degli input che dalla scelta della funzione che si desidera implementare ,g(x).

In misura minore, dipendono anche da come viene quantificata la distanza tra gli output effettivi e quelli attesi.

Una volta che viene definito il problema dell' apprendimento , si può cercare di trovare l' assegnazione ottimale dei pesi $\overline{w}$ per il problema dato.

L' algoritmo di apprendimento standard è definito nella seguente maniera :

a ogni iterazione t , un vettore di input x^t viene passato al percettrone che calcola l' output f(x^t) e lo confronta con il risultato desiderato g(x^t).

Il vettore dei pesi w^t viene aggiornato nella seguente maniera :

$w^{t+1} = w^t + \alpha(g(x^t) - f(x^t))x^t$, dove α è una costante positiva che regola la velocità dell'' apprendimento.

3.3 Scikit-Multiflow

Scikit-Multiflow è un framework di apprendimento automatico open source per i flussi di dati ispirato da MOA, un popolare framework open source per l'apprendimento automatico per i flussi di dati, e scikit-learn, il più popolare framework per l'apprendimento automatico in Python.

Consideriamo un flusso continuo di dati A =($\underset{xt}{\rightarrow}$,yt)|t = 1, . . . ,T dove T $\rightarrow\infty$. $\underset{xt}{\rightarrow}$ è un vettore di feature e yt il corrispondente target dove y è continuo nel caso della regressione e discreto per la classificazione.

L'obiettivo è prevedere il target y per un'incognita x.

Due valori di target sono considerati nella classificazione binaria, y ∈ 0,1, mentre K > 2 etichette sono usate nella classificazione multiclasse y ∈ 1, . . . K.

Sia per la classificazione binaria che per quella multiclasse viene assegnata solo una classe per istanza.

Diversamente dall'apprendimento batch, dove tutti i dati sono disponibili per la fase di addestramento train(X,y); nell'apprendimento in flusso, l'addestramento viene eseguito man mano che nuovi dati sono disponibili train($\underset{xi}{\rightarrow}$, yi).

La performance P di un dato modello è misurata secondo qualche funzione di perdita che valuta la differenza tra l'insieme delle etichette attese Y e quelle previste $\hat{Y}$.

Una delle caratteristiche più interessanti di multiflow è la sua capacità di eseguire automaticamente la valutazione delle prestazioni.

Ne esistono due tipi:

- Valutazione hold-out: i test vengono eseguiti utilizzando set di test separati
- Valutazione prequenziale o valutazione interleaved-test-then-train: è più adatta per il dominio dei flussi.

I nuovi dati sono usati come set di test prima di usarli per l'addestramento del modello.

Numerosi metodi di apprendimento sono già implementati in multiflow e sono organizzati in base al loro compito:

- Rilevamento di anomalie: Half–Space Trees
- Metodi di Bayes: Naive Bayes classifier
- Metodi di apprendimento pigro: KNN
- Metodi Ensemble
- Reti neurali
- Metodi basati su prototipi
- Metodi basati su regole
- Metodi basati su alberi: Hoeffding Tree
- Drift Detection

La classe BaseSKMObject è la classe base in scikit-multiflow.

È basata su sklearn.BaseEstimator per supportare la compatibilità tra i vari framework.

Essa aggiunge funzionalità extra che sono rilevanti nel contesto di scikit multiflow.

Qualsiasi modello di flusso in scikit-multiflow è creato estendendo la classe BaseSKMObject e il corrispondente task mixin[1], in questo caso ClassifierMixin.

ClassifierMixin definisce i seguenti metodi:

- fit: addestra un modello in modo batch. Funziona come un'interfaccia per i metodi batch che implementano una funzione fit() come i metodi scikit-learn.
- partial_fit: addestra in modo incrementale un modello di flusso
- predict: predice i valori target nei metodi di apprendimento supervisionato.

[1] Nei linguaggi di programmazione object-oriented un mixin è una classe che contiene metodi che possono essere utilizzati da altre classi senza essere la classe genitore di queste classi.

- predict_proba: calcola la probabilità che un campione appartenga ad una data classe nei problemi di classificazione

Per testare e valutare il modello viene utilizzato Prequential Evaluator, un popolare metodo di valutazione delle prestazioni per l'impostazione del flusso, in cui i test vengono eseguiti su nuovi dati prima di usarli per addestrare il modello.

La valutazione prequenziale è progettata specificamente per le impostazioni nel senso che ogni campione serve per due scopi, e che i campioni sono analizzati sequenzialmente, in ordine di arrivo, e diventano immediatamente inaccessibili.

Questo metodo consiste nell'utilizzare ogni campione per testare il modello, il che significa fare una previsione, e poi lo stesso campione viene utilizzato per addestrare il modello (adattamento parziale). In questo modo il modello viene sempre testato su campioni che non sono stati già visti.

È possibile utilizzare il parametro batch_size per considerare più di un campione alla volta, cioè blocchi di dati.

Riassumendo, i passi da eseguire per addestrare e valutare i modelli sono:

- creare un oggetto FileStream con i dati da usare. In questo modo si crea un flusso da un file sorgente. Sono supportati solo i file csv e restituisce un DataFrame di Pandas o numpy.ndarray con i dati.
  L'oggetto stream fornisce su richiesta un numero di campioni, in modo tale che i vecchi campioni non siano accessibili in un secondo momento. Questo è fatto per simulare correttamente il contesto del flusso.

```
s t r e am = F i l e S t r e am ( f i l e )
```

- preparare i dati per l'uso

```
s t r e am . p r e p a r e _ f o r _ u s e ( )
```

Questa operazione separa le feature dalle etichette target

- creare un oggetto modello.
  Il modello può essere uno di quelli che sono già implementati in multiflow o qualsiasi nuovo modello personalizzato implementato estendendo la classe ClassifierMixin

```
model = model ( . . . )
```

- creare un oggetto valutatore.

Questo valutatore può elaborare un singolo learner per tracciare le sue prestazioni; o più learner alla volta, per confrontare diversi modelli sullo stesso flusso.

ev = E v a l u a t o P r e q u e n t i a l (. . .)

Tra i possibili parametri da impostare è importante menzionare

- batch_size: il numero di campioni da passare alla volta al/i modello/i.

Esso definisce la dimensione del blocco.

- pretrain_size: il numero di campioni da utilizzare per addestrare il modello prima di iniziar la valutazione. Utilizzato per imporre un inizio "caldo".

- metrics: la lista delle metriche da tracciare durante la valutazione.

Definisce anche le metriche che saranno visualizzate nei grafici e/o registrate nel file di output.

Le opzioni valide sono: accuracy, kappa, precision,recall,f1,. . .

- output_file: nome del file dove salvare il riassunto della valutazione

- show_plot: se True, un grafico mostrerà l'andamento della valutazione

- valutare il modello calcolando le metriche che sono definite nell'oggetto ev . e v a l u a t e (s t r e am , model)

Ciascun modello è caratterizzato dai seguenti metodi :

- partial_fit: addestra parzialmente il modello.

Viene usato per passare un nuovo esempio al modello.

I prototipi prodotti sono immagazzinati in memoria e sono utilizzati per il task di classificazione.

La memoria M del blocco elaborato viene prodotta e salvata come un dizionario

p a r t i a l _ f i t (X, y , c l a s s , s amp l e_we i ght)

I parametri richiesti sono:

- X: ndarray (n_chuck_samples, n_features)

- y: ndarray (1, n_chuck_samples) contiene le etichette associate ai campioni.

- classes: ndarray con tutte le classi possibili/conosciute (default None)

- sample_weight: non usato

- predict_proba: questo metodo calcola la probabilità di ogni campione nel record di dati X di appartenere ad una certa classe.

```
p r e d i c t _ p r o b a (X)
```

dove X sono i dati utilizzati per testare il modello, restituisce un vettore con le probabilità di tutte le etichette per tutte le istanze di X.

Un esempio di utilizzo dell' Hoeffding Tree Classifier è descritto qui di seguito :

1. Creare lo stream

```
s t r e am = F i l e S t r e am ( " p a t h / f i l e . c sv " )

s t r e am . p r e p a r e _ f o r _ u s e ( )
```

2. Instanziare l' Hoeffding Tree Classifier

```
h t = HoeffdingTreeClassifier()
```

3. Impostare il valutatore

```
ev = E v a l u a t e P r e q u e n t i a l ( b a t c h _ s i z e =100 ,

p r e t r a i n _ s i z e =100 , me t r i c s =[ ' a c cur a cy ' , ' p r e c i s i o n
' , ' r e c a l l ' ] , o u t p u t _ f i l e =" p a t h / Ev a l u a t i o n . c sv " )
```

4. Run del valutatre

```
ev . e v a l u a t e ( s t r e am ,ht)
```

Chiamando la funzione evaluate(), passiamo il controllo al valutatore, che eseguirà i seguenti sottotask:

- Controllare se ci sono campioni nel flusso
- Passare i prossimi campioni al classificatore
- Testare il classificatore (usando predict())
- Aggiornare il classificatore (usando partial_fit())

- Aggiornare i risultati della valutazione e tracciare un grafico

3.4 Dati e metriche

L' oggetto fondamentale di questo lavoro è quello di analizzare i dati relativi a pazienti affetti da disordine bipolare.

Il dataset in questione è caratterizzato da valori numerici molto complessi e da 3 classi che rappresentano diversi stati di paziente "malato".

Oltre al dataset sul disordine bipolare , gli esperimenti sono stati condotti sui seguenti dataset di benchmark per un confronto numerico :

- KDDCUP : Questo è l'insieme di dati usato per la terza competizione internazionale sugli strumenti di scoperta della conoscenza e di estrazione dei dati, che è stata tenuta insieme a KDD-99, la quinta conferenza internazionale sulla scoperta della conoscenza e l'estrazione dei dati. Il compito della competizione era quello di costruire un rilevatore di intrusioni di rete, un modello predittivo capace di distinguere tra connessioni "cattive", chiamate intrusioni o attacchi, e connessioni normali "buone". Questo database contiene un set standard di dati da controllare, che include un'ampia varietà di intrusioni simulate in un ambiente di rete militare.
- Susy : I dati sono stati prodotti utilizzando simulazioni Monte Carlo. Le prime otto features rappresentano proprietà cinematiche misurate dai rilevatori di particelle nell' acceleratore. Le ultime dieci features sono funzioni delle prime otto features; queste sono features di alto livello derivate dai fisici per aiutare a discriminare tra le due classi.
- Higgs : I dati sono stati prodotti usando simulazioni Monte Carlo. Le prime 21 caratteristiche (colonne 2-22) sono proprietà cinematiche misurate dai rivelatori di particelle nell'acceleratore. Le ultime sette caratteristiche sono funzioni delle prime 21 caratteristiche; queste sono caratteristiche di alto livello derivate dai fisici per aiutare a discriminare tra le due classi.

- Hepmass : è un dataset di firme di particelle esotiche apprese da simulazioni Monte Carlo delle collisioni che producono queste particelle e i prodotti di decadimento risultanti. L' obiettivo è quello di separare le collisioni che producono le particelle da una sorgente di fondo. La massa della nuova particella è sconosciuta e in particolare il 50% dei dati proviene da un processo di segnale , mentre il 50% restante proviene dal processo di sfondo.
- RLCPS : dataset caratterizzato da due classi che rappresenta i dati raccolti nel workshop "Real-Life Cryptographic Protocols and Standardization". Si tratta di un evento il cui obiettivo è quello di raccogliere le esperienze dei progettisti e degli implementatori crittografici.

In ogni esperimento sono state utilizzate le seguenti metriche per valutare le performance dei vari classificatori :

- Accuracy: indica la percentuale di elementi predetti correttamente
- Precision: indica quanti elementi rilevanti sono selezionati
- Recall : indica quanti elementi selezionati sono rilevanti
- Kappa : è una metrica che misura l' accordo tra gli annotatori. Calcola la kappa di Cohen , un punteggio che esprime il livello di accordo tra due annotatori su un problema di classificazione. E' definito come :

  k = $(p_0 - p_e)$ / (1- p_e)

  dove :

  a) p_0 è la probabilità empirica di accordo sull'etichetta assegnata a qualsiasi campione (il rapporto di accordo osservato)
  b) p_e è l'accordo previsto quando entrambi gli annotatori assegnano le etichette in modo casuale. Esso è stimato utilizzando una priorità empirica per annotatore sulle etichette di classe
- F1 : E' una media armonica delle metriche di precision e recall. E' definita come :

$$F = \frac{2PR}{P+R} = \frac{2}{\frac{1}{R}+\frac{1}{P}}$$

3.5 Setting sperimentale

Tabella 1 - Setting Sperimentale

Dataset	#%istances	#istances	#batch	#samplexbatch	#maxistances
			18	56290	1013220
			36	28145	1013220
KDDCUP	20%	4898000	72	14072,5	1013220
			18	55600	1000800
			36	27800	1000800
RLCPS	20%	5000000	72	13900	1000800
			18	55600	1000800
			36	27800	1000800
Susy	20%	5000000	72	13900	1000800
			18	55600	1000800
			36	27800	1000800
Heapmass	10%	11000000	72	13900	1000800
			19	55600	1056400
			38	27800	1056400
Higgs	10%	11500000	76	13900	1056400
			25	36840	921000
			50	18420	921000
Bipolar			100	9210	921000

L' obiettivo di questo lavoro di tesi è quello di studiare l' influenza della grandezza del batch su algoritmi di data stream.

È stata considerata solo una piccola parte dei 6 dataset (10%/20%) , in quanto hanno dimensioni molto elevate.
Per ciascuno di questi dataset , fissato il numero massimo di istanze da considerare , verranno valutati diversi algoritmi al variare della dimensione del batch.

Per definire la dimensione del batch su scikit-multiflow , verrà indicato il numero massimo di istanze da considerare e il numero di batch.

Gli algoritmi iniziano con un numero di batch piccolo , il quale viene poi raddoppiato e quadruplicato (quindi la quantità di dati in ogni batch viene dimezzata e divisa per quattro).

L' obiettivo è quello di comprendere se , dal momento che nei batch ci sono meno dati , gli algoritmi saranno lo stesso in grado di ottenere dei risultati performanti.

3.2 Risultati

Tabella 2 - Risultati sul disordine bipolare

Dataset	Batch	Accuracy	Precision	Recall	F1	Kappa	Classi	Testing time	Training time	Algoritmo
	25	0,95	0,96	0,9	0,93	0,9		29,06	38,97	
	50	0,97	0,97	0,94	0,95	0,84		13,72	29,92	
	100	0,98	0,98	0,96	0,96	0,96		14,6	30,21	KNN
	25	0,78	0,75	0,48	0,52	0,35		26,86	13,35	
	50	0,3	0,4	0,25	0,25	0,03		59,22	14,14	
	100	0,28	0,36	0,28	0,22	0,22		62,7	14,42	Naive Bayes
	25	0,95	0,96	0,9	0,9	0,9		29,06	38,97	
	50	0,93	0,93	0,83	0,84	0,84		37,76	50,34	
	100	0,79	0,61	0,59	0,6	0,6		53,87	79,7	Hoeffding Tree
	25	0,93	0,95	0,85	0,84	0,84		11,95	293,71	
	50	0,9	0,82	0,88	0,78	0,78		12,79	247,84	
	100	0,95	0,9	0,93	0,92	0,88		13,64	252,92	Robust Soft Learning Vector Quantization
	25	0,95	0,96	0,9	0,9	0,9		272,24	2800,08	
	50	0,97	0,97	0,94	0,95	0,95		305,47	2961,25	
	100	0,99	0,99	0,98	0,97	0,97		318,77	2988,34	Adaptive Random Forest
	25	0,91	0,9	0,76	0,79	0,79		0,07	0,28	
	50	0,95	0,95	0,86	0,88	0,88		0,06	0,28	
	100	0,97	0,97	0,94	0,94	0,94		0,07	0,39	Perceptron Mask
	25	0,95	0,96	0,9	0,93	0,9		31,76	15,95	
	50	0,97	0,97	0,94	0,95	0,95		37,01	22,12	
Bipolar	100	0,97	0,97	0,94	0,94	0,94	3	43,65	27,82	Very Fast Decision Rules

Tabella 3 - Risultati Higgs

Dataset	Batch	Accuracy	Precision	Recall	F1	Kappa	Classi	Testing time	Training time	Algoritmo
	18	0,99	0,99	0,99	0,99	0,99		25,51	25,51	
	36	0,99	0,99	0,99	0,99	0,99		15,59	24,42	
	72	0,99	0,99	0,99	0,99	0,99		16,55	25,41	KNN
	18	0,99	1	0,99	0,99	0,99		72,41	29,75	
	36	0,99	1	0,99	0,99	0,99		75,53	28,95	
	72	0,99	1	0,99	0,99	0,99		79,81	29,84	Naive Bayes
	18	0,99	1	0,99	0,99	0,99		38,12	70,91	
	36	0,99	1	0,99	0,99	0,99		40,27	67,87	
	72	0,99	1	0,99	0,99	0,99		42,71	70,36	Hoeffding Tree
	18	1	1	1	1	1		4,97	91,2	
	36	1	1	1	1	1		5,1	86,63	
	72	1	1	1	1	1		5,33	88,98	Robust Soft Learning Vector Quantization
	18	1	1	1	1	1		256,85	2153,74	
	36	1	1	1	1	1		257,32	2051,33	
	72	1	1	1	1	1		273,87	2132,43	Adaptive Random Forest
	18	1	1	1	1	1		0,04	0,21	
	36	1	1	1	1	1		0,06	0,16	
	72	1	1	1	1	1		0,04	0,19	Perceptron Mask
	18	0,99	1	0,99	0,99	0,99		39,97	31,98	
	36	0,99	1	0,99	0,99	0,99		39,22	32,25	
Higgs	72	0,99	1	0,99	0,99	0,99	2	42,37	33,01	Very Fast Decision Rules

Tabella 4 - Risultati RLCPS

Dataset	Batch	Accuracy	Precision	Recall	F1	Kappa	Classi	Testing time	Training time	Algoritmo
	18	0,99	0,99	0,99	0,99	0,99		25,51	25,51	
	36	0,99	0,99	0,99	0,99	0,99		15,59	24,42	
	72	0,99	0,99	0,99	0,99	0,99		16,55	25,41	KNN
	18	0,99	1	0,99	0,99	0,99		72,41	29,75	
	36	0,99	1	0,99	0,99	0,99		75,53	28,95	
	72	0,99	1	0,99	0,99	0,99		79,81	29,84	Naive Bayes
	18	0,99	1	0,99	0,99	0,99		38,12	70,91	
	36	0,99	1	0,99	0,99	0,99		40,27	67,87	
	72	0,99	1	0,99	0,99	0,99		42,71	70,36	Hoeffding Tree
	18	1	1	1	1	1		4,97	91,2	
	36	1	1	1	1	1		5,1	86,63	
	72	1	1	1	1	1		5,33	88,98	Robust Soft Learning Vector Quantization
	18	1	1	1	1	1		256,85	2153,74	
	36	1	1	1	1	1		257,32	2051,33	
	72	1	1	1	1	1		273,87	2132,43	Adaptive Random Forest
	18	1	1	1	1	1		0,04	0,21	
	36	1	1	1	1	1		0,06	0,16	
	72	1	1	1	1	1		0,04	0,19	Perceptron Mask
	18	0,99	1	0,99	0,99	0,99		39,97	31,98	
	36	0,99	1	0,99	0,99	0,99		39,22	32,25	
RLCPS	72	0,99	1	0,99	0,99	0,99	2	42,37	33,01	Very Fast Decision Rules

Tabella 5- Risultati Hepmass

Dataset	Batch	Accuracy	Precision	Recall	F1	Kappa	Classi	Testing time	Training time	Algoritmo
	18	0,99	0,99	0,99	0,99	0,99		25,51	25,51	
	36	0,99	0,99	0,99	0,99	0,99		15,59	24,42	
	72	0,99	0,99	0,99	0,99	0,99		16,55	25,41	KNN
	18	0,99	1	0,99	0,99	0,99		72,41	29,75	
	36	0,99	1	0,99	0,99	0,99		75,53	28,95	
	72	0,99	1	0,99	0,99	0,99		79,81	29,84	Naive Bayes
	18	0,99	1	0,99	0,99	0,99		38,12	70,91	
	36	0,99	1	0,99	0,99	0,99		40,27	67,87	
	72	0,99	1	0,99	0,99	0,99		42,71	70,36	Hoeffding Tree
	18	1	1	1	1	1		4,97	91,2	
	36	1	1	1	1	1		5,1	86,63	
	72	1	1	1	1	1		5,33	88,98	Robust Soft Learning Vector Quantization
	18	1	1	1	1	1		256,85	2153,74	
	36	1	1	1	1	1		257,32	2051,33	
	72	1	1	1	1	1		273,87	2132,43	Adaptive Random Forest
	18	1	1	1	1	1		0,04	0,21	
	36	1	1	1	1	1		0,06	0,16	
	72	1	1	1	1	1		0,04	0,19	Perceptron Mask
	18	0,99	1	0,99	0,99	0,99		39,97	31,98	
	36	0,99	1	0,99	0,99	0,99		39,22	32,25	
Hepmass	72	0,99	1	0,99	0,99	0,99	2	42,37	33,01	Very Fast Decision Rules

Tabella 6 - Risultati KDDCUP

Dataset	Batch	Accuracy	Precision	Recall	F1	Kappa	Classi	Testing time	Training time	Algoritmo
	18	0,99	0,99	0,99	0,99	0,99		25,51	25,51	
	36	0,99	0,99	0,99	0,99	0,99		15,59	24,42	
	72	0,99	0,99	0,99	0,99	0,99		16,55	25,41	KNN
	18	0,99	1	0,99	0,99	0,99		72,41	29,75	
	36	0,99	1	0,99	0,99	0,99		75,53	28,95	
	72	0,99	1	0,99	0,99	0,99		79,81	29,84	Naive Bayes
	18	0,99	1	0,99	0,99	0,99		38,12	70,91	
	36	0,99	1	0,99	0,99	0,99		40,27	67,87	
	72	0,99	1	0,99	0,99	0,99		42,71	70,36	Hoeffding Tree
	18	1	1	1	1	1		4,97	91,2	
	36	1	1	1	1	1		5,1	86,63	
	72	1	1	1	1	1		5,33	88,98	Robust Soft Learning Vector Quantization
	18	1	1	1	1	1		256,85	2153,74	
	36	1	1	1	1	1		257,32	2051,33	
	72	1	1	1	1	1		273,87	2132,43	Adaptive Random Forest
	18	1	1	1	1	1		0,04	0,21	
	36	1	1	1	1	1		0,06	0,16	
	72	1	1	1	1	1		0,04	0,19	Perceptron Mask
	18	0,99	1	0,99	0,99	0,99		39,97	31,98	
	36	0,99	1	0,99	0,99	0,99		39,22	32,25	
KDDCUP	72	0,99	1	0,99	0,99	0,99	2	42,37	33,01	Very Fast Decision Rules

Tabella 7 - Risultati Susy

Dataset	Batch	Accuracy	Precision	Recall	F1	Kappa	Classi	Testing time	Training time	Algoritmo
	18	0,99	0,99	0,99	0,99	0,99		25,51	25,51	
	36	0,99	0,99	0,99	0,99	0,99		15,59	24,42	
	72	0,99	0,99	0,99	0,99	0,99		16,55	25,41	KNN
	18	0,99	1	0,99	0,99	0,99		72,41	29,75	
	36	0,99	1	0,99	0,99	0,99		75,53	28,95	
	72	0,99	1	0,99	0,99	0,99		79,81	29,84	Naive Bayes
	18	0,99	1	0,99	0,99	0,99		38,12	70,91	
	36	0,99	1	0,99	0,99	0,99		40,27	67,87	
	72	0,99	1	0,99	0,99	0,99		42,71	70,36	Hoeffding Tree
	18	1	1	1	1	1		4,97	91,2	
	36	1	1	1	1	1		5,1	86,63	
	72	1	1	1	1	1		5,33	88,98	Robust Soft Learning Vector Quantization
	18	1	1	1	1	1		256,85	2153,74	
	36	1	1	1	1	1		257,32	2051,33	
	72	1	1	1	1	1		273,87	2132,43	Adaptive Random Forest
	18	1	1	1	1	1		0,04	0,21	
	36	1	1	1	1	1		0,06	0,16	
	72	1	1	1	1	1		0,04	0,19	Perceptron Mask
	18	0,99	1	0,99	0,99	0,99		39,97	31,98	
	36	0,99	1	0,99	0,99	0,99		39,22	32,25	
Susy	72	0,99	1	0,99	0,99	0,99	2	42,37	33,01	Very Fast Decision Rules

Capitolo 4 – Conclusioni

Abbiamo analizzato diversi stream data di benchmarck e un dataset relativo a pazienti affetti da disordine bipolare. Sono stati valutati utilizzando la tecnica prequential train e test, che utilizza sottoinsiemi di dati prima come test set e poi come training set. Abbiamo voluto valutare se la dimensione del batch influenzi i risultati degli algoritmi di stream. A tale scopo è stata utilizzata la libreria scikit-multiflow per la gestione degli stream data. Sono stati considerati 6 algoritmi di diverse categorie, e sono state valutate le performance di classificazione.

Si è osservato che nei dati di benchmarck la dimensione del batch e il tipo di algorimo utilizzato non influenzano i risultati che sono tra loro comparabili e presentano valori di accuratezza molto alti se non pari a 1.

I dati relativi al disordine bipolare hanno restituito risultati differenti a seconda dell'algoritmo utilizzato e della dimensione del batch. Tale risultato era prevedibile vista la complessità di questo tipo di dati.

In particolare l' algoritmo che in questo contesto risulta essere il più performante è l' Adaptive Random Forest , il quale ha un accuracy maggiore rispetto agli altri (0,99) nel momento in cui la dimensione del batch diminuisce.

Sviluppi futuri saranno orientati alla valutazione di diverse tipologie di validazione dei risultati, diversi algoritmi e dati.

Bibliografia e Sitografia

[1]: Zikopoulos, P., Eaton, C., DeRoos, D., Deutsch, T., and Lapis, G. (2011). Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw Hill
Professional.

[2]: Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2009). Data Stream Mining. In Data Mining and Knowledge Discovery Handbook, volume 8, pages 759–787. Springer
US, Boston, MA.

[3]: Babcock, B., Babu, S., Datar, M., Motwani, R., andWidom, J. (2002). Models and issuesin data stream systems.
Proceedings of 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pages 1–16.

[4] : Chandak, M. B. (2016). Role of big-data in classification and novel class detection in data streams.
Journal of Big Data, 3.

[5] : Chen, M., Mao, S., and Liu, Y. (2014). Big Data: A Survey. Mobile Networks and Applications, 19(2):171–209.

[6] : Kuncheva, L. I. (2004). Classifier Ensembles for Changing Environments. In Roli,

F., Kittler, J., and Windeatt, T., editors, International Workshop on Multiple Classifier Systems. MCS 2004, volume 3077 of Lecture Notes in Computer Science, pages 1–15. Springer.

[7] : Olorunnimbe, M. K., Viktor, H. L., and Paquet, E. (2018). Dynamic adaptation of online ensembles for drifting data streams. J. Intell. Inf. Syst., 50(2):291–313.

[8] : Angelov, P., Lughofer, E. D., and Zhou, X. (2008). Evolving fuzzy classifiers usingdifferent model architectures.

[9] : Pratama, M., Pedrycz, W., and Lughofer, E. (2018). Evolving Ensemble Fuzzy Classifier.
IEEE Transactions on Fuzzy Systems, 26:2552–2567

[10] : Yang, J., Yan, X., Han, J., and Wang, W. (2003). Discovering an Evolutionary Classifier over a High-speed Nonstatic Stream. In Advanced Methods for Knowledge Discovery from Complex Data, pages 337–363. Springer-Verlag, New York.

[11]: Ikonomovska, E., Gama, J., and Dzeroski, S. (2010). Learning model trees from evolving data streams. Data Mining and Knowledge Discovery, 23:128–168.

[12] : Zhang, Z. and Zhou, J. (2010). Transfer estimation of evolving class priors in data stream classification.
Pattern Recognition, 43:3151–3161.

[13] : Yang, C. and Zhou, J. (2008). Non-stationary data sequence classification using online class priors estimation.
Pattern Recognition, 41(8):2656–2664.

[14] : Pang, S., Ozawa, S., and Kasabov, N. K. (2005). Incremental linear discriminant analysis for classification of data streams. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 35:905–914.

[15] Guarracino, M. R., Cuciniello, S., and Feminiano, D. (2009). Incremental generalized eigenvalue classification on data streams. In International workshop on data stream management and mining, pages 1–12.

[16]: Annapoorna, P. V. S. and Mirnalinee, T. T. (2016). Streaming data classification. 2016 International Conference on Recent Trends in Information Technology (ICRTIT), pages 1–7.

[17]: Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., Holmes, G., and Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification.
Machine Learning, 106:1469–1495.

[18] : Grande, I., Berk, M., Birmaher, B., and Vieta, E. (2016). Bipolar disorder. The Lancet,387(10027):1561–1572.

[19] : Lee, R. (2008). Bipolar disorders: Mixed states, rapid cycling and atypical forms. Journal of Mental Health, 17.

[20] : Grunze, H. (2015). Bipolar disorder. Neurobiology of brain disorders, pages 655–673.

[21]: Faurholt-Jepsen, M., J. Busk, M. F., Bardram, J. E., Vinberg, M., and Kessing, L. V.(2019). Objective smartphone data as a potential diagnostic marker of bipolar disorder.
Australian & New Zealand Journal of Psychiatry, 53(2):119–128.

[22] : J, A., J-M, A., CL, B., and study Group, B. (2011). Prevalence and characteristics of undiagnosed bipolar disorders in patients with a major depressive episode: the BRIDGE study.
Arch Gen Psychiatry, 68:791–79

[23] : https://scikit-learn.org/stable/modules/neighbors.html#classification

[24] : https://scikit-learn.org/stable/modules/naive_bayes.html

[25] : Stephen Marsland. Machine learning: an algorithmic perspective. CRC press, 2015.

[26]: Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An introduction to statistical learning, volume 112. Springer, 2013.

[27] : P Domingos and G Hulten. Mining high-speed data streams.[in:] proceedings of the sixth acm sigkdd international conference on knowledge discovery and data mining. Boston, 71:80, 2000.

[28] : Leo Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

[29] : Leo Breiman. Bagging predictors. Machine learning, 24(2):123–140, 1996.

[30] : Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging bagging for evolving data streams. In Joint European conference on machine learning and knowledge discovery in databases, pages 135–150. Springer, 2010.

[31] : Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabr´ıcio Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. Machine Learning, 106(9-10):1469–1495, 2017.

[32] : Nikunj C Oza. Online bagging and boosting. In 2005 IEEE international conference on systems, man and cybernetics, volume 3, pages 2340–2345. Ieee, 2005.

[33] : Kohonen, T. In: Learning vector quantization. MIT Press, Cambridge, MA, USA (1995) 537–540

[34]: Kohonen, T.: Improved versions of learning vector quantization. Procedings of the International Joint conference on Neural Networks 1 (1990) 545–550

[35] : Kohonen, T.: Self-Organizing Maps. Springer, Berlin (1997)

[36] : Sato, A., Yamada, K. In: Generalized learning vector quantization. Volume 7. (1995) 423–429

[37]: Seo, S., Bode, M., Obermayer, K.: Soft nearest prototype classification. IEEE Transactions on Neural Networks 13 (2003) 390–398

[38] : Seo, S., Obermayer, K.: Soft learning vector quantization. Neural computation 15 (2003) 1589–1603

[39] : Ghosh, A., Biehl, M., Hammer, B.: Dynamical analysis of lvq type learning rules. In: Workshop on the Self-Organizing-Map WSOM05. Univ. de Paris (I. (2005)

[40] : https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.rules.VeryFastDecisionRulesClassifier.html

[41] : https://it.wikipedia.org/wiki/Percettrone