POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Self-tuning di un trading system azionario basato su tecniche di machine learning

Relatore
prof. Luca Cagliero

Correlatore:
prof. Paolo Garza

Laureando
Marco Povero
matricola: 202264

Marzo/Aprile 2020

Sommario

Il trading quantitativo è una strategia di investimento in cui si impiegano software e strumenti di analisi automatica dei dati finanziari per riuscire ad operare sui mercati in modo sistematico. Combinare il trading quantitativo e il data mining, che studia come estrarre informazioni implicite e nascoste all'interno di grandi quantità di dati, significa elaborare i dati finanziari attraverso algoritmi di machine learning e generare automaticamente i modelli matematici che permettono di riconoscere opportunità di investimento e supportare le decisioni di trading. Uno dei più grandi benefici del trading quantitativo è quello di poter usare i dati storici per validare una strategia di investimento, simulando quanto si sarebbe potuto guadagnare se si fosse applicato uno specifico modello nel passato. Nonostante ciò, la natura dinamica e aleatoria dei mercati azionari rende complesso, anche per i trader sistematici, individuare il modello più adatto alla previsione dell'andamento futuro delle azioni. Per affrontare tale complessità, in questa tesi si propone un sistema di *self-tuning* di modelli quantitativi, basati su machine learning, che mira a determinare automaticamente quale metodo applicare a fronte dell'evoluzione delle condizioni di mercato. In particolare, il sistema proposto analizza le prestazioni di più configurazioni di diversi modelli di classificazione applicati sui dati storici dei mercati azionari, al fine di apprendere quale algoritmo e quali parametri sono i più adatti a classificare le azioni in diversi periodi di tempo. I risultati ottenuti simulando uno scenario di intraday trading dimostrano che è possibile incrementare e mantenere discreti rendimenti applicando il self-tuning su base trimestrale, in modo da cambiare strategia di trading nel tempo e fare affidamento su più metodi per affrontare l'indeterminatezza dei mercati.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

Negli ultimi anni si è sentito parlare sempre più spesso del trading online come una grande opportunità per guadagnare denaro "giocando in borsa" comodamente dal proprio PC di casa. Tecnicamente tutto ciò è possibile, poiché fare trading online significa comprare e vendere strumenti finanziari usando software che accedono via Internet ai servizi di *broker online*, messi a disposizione da banche e società finanziare. Accedere ed operare facilmente sui mercati finanziari non implica, però, generare profitti in modo altrettanto semplice: il trading è un'attività che comporta dei rischi e, pertanto, richiede l'adozione di opportune strategie di investimento.

Generalmente le strategie di investimento si possono dividere in due modalità operative: il trading discrezionale e il trading quantitativo. Nel primo caso, il trader usa la sua abilità e la sua esperienza per studiare i dati finanziari seguendo i principi dell'analisi fondamentale e dell'analisi tecnica, nel secondo, si affida ad un sistema di trading automatico che, tramite l'impiego di algoritmi e formule matematiche, analizza i dati finanziari e segnala al trader quando e come operare sul mercato.

Considerando che al giorno d'oggi la quantità di informazioni disponibili sui mercati finanziari e sulle società quotate (come ad esempio prezzi storici, volumi scambiati, news, report, analisi, bilanci) è talmente vasta da rendere impossibile analizzare manualmente ogni singola fonte, diventa necessario impiegare tecniche automatiche per l'analisi di tali fonti dati. In tale contesto, il data mining è uno strumento molto potente, in quanto dispone di diverse tecniche e metodologie in grado di analizzare automaticamente grandi quantità di dati, allo scopo di estrarre da essi informazioni e conoscenze utili su un determinato fenomeno. Quando tali tecniche sono applicate allo studio delle informazioni raccolte dai mercati finanziari, si parla di "*financial data mining*" e la conoscenza estratta dai dati diventa utile per prevedere l'andamento dei titoli quotati in borsa.

Questa tesi è svolta nel contesto del financial data mining ed è incentrata sull'applicazione di tecniche di classificazione nei sistemi di trading quantitativo. La classificazione è una tecnica di data mining supervisionata che, analizzando dati categorizzati in più classi, genera un modello in grado di identificare a quale classe

appartengono dati non ancora categorizzati. Tipicamente viene usata per prevedere le tipologie di clienti a cui offrire servizi di marketing diretto, oppure per riconoscere patologie cliniche, o ancora per marcare correttamente i messaggi di spam. Gli scenari applicativi possono comunque indirizzarsi verso molti altri obiettivi, come nel caso di questa tesi, dove la classificazione è usata per prevedere come varieranno i prezzi di strumenti finanziari.

Numerosi lavori precedenti hanno investigato l'applicabilità delle tecniche di regressione e di classificazione per la predizione dei prezzi giornalieri di strumenti azionari quotati sui principali mercati finanziari mondiali. Mentre i sistemi basati sulla regressione sono finalizzati alla predizione dell'esatto valore del prezzo di chiusura di un'azione nel futuro (ad esempio il giorno successivo), nel caso della classificazione, i modelli predittivi hanno l'obiettivo di identificare il range di valori all'interno del quale, con elevata probabilità, varierà il prezzo futuro dell'azione.

Sebbene questi ultimi modelli siano risultati più efficaci rispetto a quelli di regressione, la principale limitazione dei modelli di classificazione è la complessità della fase di configurazione, la quale dipende fortemente della distribuzione dei dati analizzati. Infatti, modificando i parametri di configurazione del sistema, capita spesso che le performance di predizione varino in modo sostanziale. Le predizioni dei valori futuri, inoltre, devono essere combinate opportunamente per automatizzare il processo di generazione dei segnali di trading attraverso piattaforme di trading quantitativo e, anche in questo caso, è richiesta nuovamente la configurazione di alcuni parametri. Risulta dunque complesso per l'utente finale (spesso non esperto di analisi dei dati) scegliere a priori la configurazione più opportuna.

L'obiettivo di questa tesi è studiare un sistema che identifichi automaticamente come configurare nel modo più appropriato i classificatori usati a supporto delle strategie di trading sui mercati azionari.

L'approccio che si propone è basato su un sistema predittivo ibrido composto da molteplici tecniche di classificazione e da un modello di regressione. Le tecniche di classificazione sono usate per prevedere l'andamento delle azioni simulando di applicare sui prezzi storici degli stock differenti algoritmi e più parametri di configurazione. I risultati di ciascuna simulazione servono poi per addestrare il modello regressione a valutare quale configurazione dei classificatori massimizzerà i profitti nell'attuale condizione di mercato. Il procedimento con cui il sistema analizza le prestazioni dei diversi metodi di classificazione, finalizzato alla auto-configurazione del modello da utilizzare, è definito *self-tuning*.

La sperimentazione del sistema proposto ha simulato l'impiego di 7200 diverse configurazioni di classificatori per raccomandare i titoli FTSE MIB su cui investire giornalmente dal 2010 al 2017, dimostrando che solo pochi modelli riescono a conseguire discreti guadagni a fronte del cambiamento delle condizioni di mercato, mentre altri, con specifiche configurazioni di parametri, conducono ad ottimi profitti solo in determinati periodi. Sfruttare un algoritmo di regressione polinomiale per stimare le future performance dei classificatori, permette di rendere dinamica la

scelta dell'algoritmo e dei parametri da utilizzare per le raccomandazioni giornaliere e, inoltre, permette di conseguire risultati più stabili nel tempo.

Calcolando i guadagni ottenuti dalle simulazioni nell'arco di un anno di trading, tra il 2010 e il 2017 la maggior parte dei modelli non si è dimostrata in grado ottenere profitti, mentre, le poche configurazioni di classificatori che sono risultate più efficienti, hanno registrato rendimenti giornalieri compresi tra lo 0.26% e il 2.13%. Nessuna di queste configurazioni, comunque, ha mantenuto un rendimento positivo per più di due anni consecutivi. Solamente applicando il *self-tuning* dei classificatori e cambiando dinamicamente strategia ogni trimestre è stato possibile ottenere un rendimento giornaliero dello 0.36% dal 2011 al 2017: ciò conferma la maggiore robustezza del metodo proposto a fronte della mutevolezza del mercato.

La tesi è organizzata come segue. Il capitolo 2 tratta i concetti relativi al data mining e alle sue applicazioni. Nel capitolo 3 si presenta lo scenario del trading quantitativo, si definiscono i principali termini usati nel mondo del trading e si descrivono gli obiettivi del data mining in tale scenario. Il capitolo 4 approfondisce tali obiettivi riportando diversi esempi di metodologie di trading quantitativo allo stato dell'arte, tra i quali si tratta con maggiore attenzione il sistema BEst STock Finder (BESt), poiché rappresenta il punto di partenza del lavoro svolto in questa tesi. Il capitolo 5 è interamente dedicato alla descrizione del metodo proposto per la sperimentazione delle tecniche di classificazione e alle scelte progettuali coinvolte nel disegno del modello predittivo e della procedura di self-tuning. I risultati conseguiti da tale metodo sono illustrati nel capitolo 6. Ed infine, nel capitolo 7, si riassumono le osservazioni e le conclusioni tratte a seguito del lavoro svolto e si indicano potenziali sviluppi futuri della soluzione realizzata.

Capitolo 2

Introduzione alle tecniche di data mining

Questo capitolo è rivolto alle principali tecniche di data mining utilizzate all'interno di questo lavoro di tesi, evidenziando la loro utilità e applicabilità nell'ambito dell'analisi dei dati finanziari.

Quando si fa riferimento al termine *tecniche di data mining* o semplicamente *data mining*, si considera l'insieme di algoritmi e metodologie che appartengono ad un campo interdiciplinare all'intersezione tra intelligenza artificiale, machine learning, statistica e sistemi di database[1].

Ciò che contraddistingue l'insieme di questi metodi è la capacità di analizzare *automaticamente* i dati di un determinato fenomeno, *estraendo* (in inglese *"mining"*) informazioni utili per apprendere e comprendere meglio ciò che i dati rappresentano.

Il concetto di *estrazione di informazioni utili* viene enfatizzato da un'ulteriore definizione del data mining, in cui l'applicazione delle tecniche viene contestualizzata all'interno del processo denominato *"Knowledge Discovery Process from Data"* (*KDD Process*), espresso come *"l'estrazione non banale di informazioni implicite, non note a priori e potenzialmente utili dai dati"*[2].

Come si può vedere dallo schema in figura 2.1, nell'ambito del KDD Process, il data mining costituisce la componente *algoritmica* del processo, che elabora i dati di input per generare modelli astratti, detti *pattern*, con cui è possibile rappresentare quelle informazioni che non erano evidenti osservando i dati prima del processo.

Dal momento che le fasi del KDD Process saranno utilizzate anche successivamente per descrivere i passaggi principali della metodologia proposta, di seguito si definisce brevemente il significato di ogni fase rappresentata nella figura 2.1:

1. Integration
   Consiste nella raccolta e consolidamento dei dati provenienti da più sorgenti (*Data Sources*) in un unico contenitore (*Data Store*).

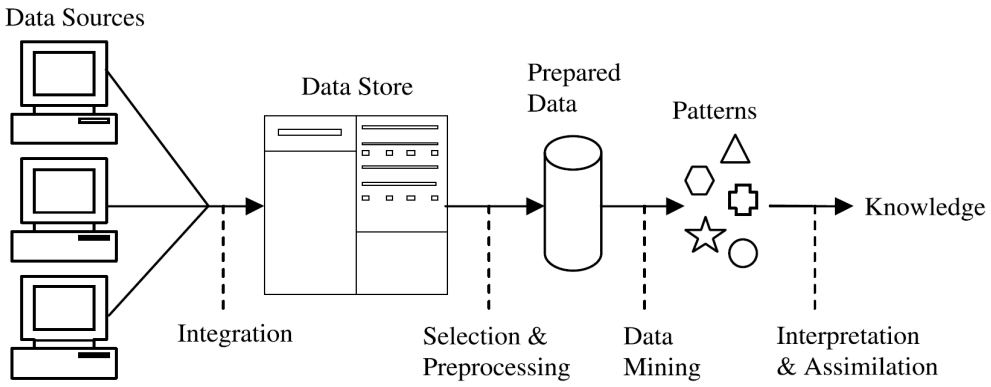

Figura 2.1: Knowledge Discovery Process[2]

2. Preprocessing
   Comprende tutte le attività necessarie alla creazione dell'insieme di dati che dovrà essere analizzato dagli algoritmi di data mining. Tra queste attività ci sono, ad esempio, la selezione degli attributi più rilevanti per l'analisi (*feature selection*) o la trasformazione e la pulizia (*cleansing*) dei dati da valori anomali o mancanti.

3. Data mining
   La parte algoritmica del processo, detta anche Modeling, consiste nell'applicare i metodi e gli algoritmi di data mining ai dati di input per estrarre un modello (*pattern*) in grado di fornire una risposta sufficientemente accurata al problema in esame.

4. Interpretation
   Infine, l'interpretazione dei modelli è la fase che porta all'acquisizione di quelle informazioni non note a priori da cui consegue una maggiore conoscenza dei dati analizzati.

Nella sezione seguente si descrive il funzionamento della fase data mining indicando quali sono le principali tecniche utilizzate per l'estrazione dei pattern e quali sono i rispettivi obiettivi. La parte restante del capitolo sarà poi dedicata all'approfondimento sulla classificazione, che è la tecnica principalmente usata in questa tesi.

2.1 Tecniche principali

L'algoritmo scelto per eseguire l'*estrazione della conoscenza* produce un pattern più facilmente interpretabile o meno in base alla tecnica che è stata utilizzata. Le tipologie di pattern sono distinte in due categorie[3]:

- Pattern *Predittivi*:
  permettono di predire il valore di determinate variabili, a fronte di nuovi dati non ancora conosciuti, in funzione delle variabili conosciute.

- Pattern *Descrittivi*:
  offrono una diversa rappresentazione dei dati di partenza, interpretabile dall'uomo per descrivere i dati.

I pattern di tipo predittivo sono generati elaborando i dati di input attraverso tecniche di apprendimento *supervisionato* (*supervised learning*), mentre invece i pattern descrittivi sono determinati da tecniche di apprendimento *non supervisionato* (*unsupervised learning*). Tali metodologie di apprendimento rappresentano le due principali macro-categorie di tecniche di data mining:

- Tecniche *supervisionate*

  Suddividono l'insieme dei dati da analizzare in variabili di input x e variabili di output Y e, attraverso un *algoritmo*, costruiscono una funzione $f(x) = Y$ che definisce le regole per mappare le variabili di ingresso sulle relative uscite.

  L'obiettivo di questo procedimento è approssimare al meglio la funzione $f(x)$ per poterla applicare su nuove istanze di x, di cui non è ancora noto il futuro valore di Y. In tal senso, i dati analizzati sono usati come campione (*training set*) per inferire la funzione di mapping.

- Tecniche *non supervisionate*

  Considerano solamente le variabili di input x per determinare automaticamente gli output Y, modellando nella forma più opportuna la struttura e la distribuzione delle informazioni disponibili, in modo da rappresentare le similarità o le differenze presenti nei dati.

Partendo dalle due categorie evidenziate, di seguito si elencano le principali tecniche di data mining, descrivendone brevemente il funzionamento e la tipica applicazione in scenari reali. Nella descrizione sono usati i seguenti termini:

- *Data set*:
  insieme dei dati di input all'algoritmo di data mining, inteso come collezione di *esempi*.

- *Esempio*:
  singolo oggetto (detto anche istanza o record) del data set.

- *Attributo*:
  proprietà dell'esempio (es. colonna di un record) di tipo *discreto* o *continuo* (rispettivamente definito anche come tipo *nominale/categorico* o tipo *numerico*).

NB. I termini indicati sono gli stessi usati del software RapidMiner che, come descritto nell'appendice B, è stato usato per la realizzazione del progetto.

Tecniche Supervisionate

- CLASSIFICAZIONE

  La classificazione analizza insiemi di dati in cui ogni esempio è assegnato ad una specifica classe di appartenenza. Tale classe è rappresentata da un attributo definito *attributo di classe* oppure ***label*** ("etichetta"). Nel caso in cui l'attributo di classe non sia incluso nei dati iniziali, deve essere opportunamente generato prima di sottoporre il dataset agli algoritmi di classificazione.

  Il modello estratto da queste tecniche esprime il valore dell'attributo di classe in funzione dei valori assunti dagli altri attributi. Quando il modello viene applicato su nuovi esempi, assegna ad essi la classe più appropriata usando le informazioni apprese dal training set[6].

  Come indicato nell'introduzione, tra le tipiche applicazioni della classificazione si citano il marketing diretto, la diagnosi di patologie cliniche e lo spam detection[8].

  Nel caso del financial data mining, come verrà descritto nei capitoli successivi, i dati elaborati dagli algoritmi di classificazione sono prevalentemente numerici e devono, pertanto, essere opportunamente pre-processati al fine di generare una *label discreta*, utilizzabile per il training dei modelli predittivi. In tal senso, la classificazione applicata all'analisi dei dati finanziari più che prevedere i prezzi, definisce un pattern che esprima *il modo in cui variano i prezzi* degli strumenti finanziari.

- REGRESSIONE

  La regressione è un metodo statistico usato per predire il valore di una variabile a valori continui sulla base del valore delle altre variabili, assumendo un modello di dipendenza lineare e non lineare.

  A differenza della classificazione, che produce un output *nominale*, i modelli di regressione lavorano esclusivamente su dati numerici per determinare un risultato *numerico* (*continuo*).

  Trattando dati numerici, le tecniche di regressione sono state le prime ad essere utilizzate per la previsione dei prezzi del mercato finanziario e per l'analisi dei prezzi storici. Altri impieghi riguardano in generale l'analisi statistica nelle scienze applicate, come ad esempio il calcolo della velocità del vento in funzione di temperatura, umidità e pressione atmosferica[8].

La dimensione e il contenuto dei dati di training, usati dalle tecniche supervisionate, costituiscono un aspetto molto delicato, poiché possono condurre a problemi di *overfitting* (o *overtraining*), ovvero di situazioni in cui il modello generato è estremamente performante nella predizione degli esempi conosciuti (utilizzando anche più parametri di quelli necessari) ma non in quella dei dati nuovi (*out-of-sample*).

Come verrà descritto nella sezione di approfondimento sulle tecniche di classificazione (sezione 2.2), alcuni algoritmi di data mining prevedono dei meccanismi per minimizzare l'overfitting, mentre altri richiedono l'impiego di tecniche di ottimizzazione.

Tecniche non supervisionate

- **Clustering**

  Il *clustering* modella gli esempi come un insieme di punti e definisce una funzione per misurarne la *similarità*. Sulla base della misura ottenuta, si creano raggruppamenti, detti *cluster*, tali per cui gli esempi appartenenti allo stesso cluster sono *più simili* tra di loro e *meno simili* rispetto a quelli appartenenti ad altri raggruppamenti[7].

  Tipicamente il clustering viene usato nelle strategie di marketing basate sulla segmentazione del mercato, oppure per creare automaticamente raggruppamenti di documenti contenenti gli stessi termini o raggruppamenti di geni con carateristiche analoghe[8].

  Non esistono molti approcci all'analisi dei dati finanziari basati sul clustering. In genere queste tecniche sono impiegate per riassumere e descrivere le caratteristiche dei dati analizzati, producendo una serie di attributi qualitativi utilizzabili successivamente da metodi predittivi. Per esempio, il clustering può essere applicato per definire gli attributi di classe categorici necessari all'impiego delle tecniche di classificazione.

- **Regole di Associazione**

  Identificano regole di dipendenza che descrivono l'occorrenza di un elemento come conseguenza dell'occorrenza degli altri elementi[5].

  Tali regole sono determinate attraverso l'impiego di misure statistiche, come *supporto* e *confidenza*, che stabiliscono il grado di probabilità sufficiente per formare un'associazione tra gli elementi.

  Anche queste tecniche trovano applicazione nel marketing, in particolare nell'analisi delle abitudini di acquisto dei consumatori, che stanno alla base della *market basket analysis* e del *cross-selling*, con cui si pubblicizzano proattivamente i prodotti comprati spesso insieme[8].

Classificazione, regressione, clustering e regole di associazione costituiscono i principali metodi di data mining. Su di essi si basano molte altre metodologie distinguibili per la loro specializzazione in determinati scenari di analisi, come ad esempio il *sequence mining* e il *time series mining*, di cui di seguito viene data una breve descrizione, poiché si tratta di approcci riproposti nel capitolo 4 per illustrare lo stato dell'arte delle metodologie di trading quantitativo.

Tecniche adatte a tipologie di dati specifiche

- SEQUENCE MINING

  Detto anche *sequencial pattern mining*, è tipicamente basato sullo stesso approccio delle regole di associazione, ma con una specializzazione nel riconoscimento di *elementi ordinati* secondo *sequenze ricorrenti*[3].

  Sono applicate, per esempio, per identificare i motivi nell'analisi di dati biologici o, riproponendo lo scenario del marketing, per determinare la strategia di presentazione dei prodotti esposti sugli scaffali dei supermercati (*shelf marketing*). In questa tesi sono citate in riferimento al sistema BEST (descritto nella sezione 4.2), dove vengono sfruttate per modellare l'occorrenza delle azioni più profittevoli in sequenze di giorni consecutivi.

- TIME SERIES MINING

  Le serie temporali permettono di individuare pattern ricorrenti o atipici in sequenze di dati complesse sfruttando altre tecniche, come il clustering, la classificazione e le regole di associazione, ma introducendo uno stretto legame con la dimensione del tempo in cui si verificano gli eventi del fenomeno osservato[3].

  Per trattare la serie cronologica con gli algoritmi di mining è richiesto un particolare preprocessing dei dati, volto a rappresentare il tempo in un altro dominio (per esempio applicando la trasformata di Fourier) o in un formato compresso che approssimi i punti più rilevanti per la descrizione degli eventi in esame.

  Un esempio rappresentativo di questi metodi, applicato nel settore della sanità, è quello dell'analisi dei dati registrati dall'elettrocardiogramma per fare diagnosi in base alle variazioni dei battiti cardiaci. Mentre un altro esempio tipico è costituito dall'analisi dei trend dei dati finanziari.

  Fondamentalmente tutti metodi che verranno trattati in questa tesi includono la componente del tempo, poiché si basano sull'analisi di eventi occorsi nel passato per prevedere ciò che avverrà in futuro. Gli eventi relativi al passato sono rappresentati dai dati storici dei mercati finanziari, mentre gli eventi da prevedere nel futuro riguardano il valore dei prezzi e delle loro variazioni.

2.2 Tecniche di classificazione

Dopo aver presentato le carattersitiche delle principali tecniche di data mining, di seguito si approfindicono gli aspetti relativi alla classificazione e agli algoritmi che implementano questa tecnica supervisionata[6].

Dal punto di vista procedurale, la classificazione è un processo composto da due fasi: prima la generazione del modello e poi la sua applicazione.

Il primo passaggio è quello in cui gli algoritmi di classificazione definiscono il modello con cui mappare gli esempi del dataset sulla corretta classe di appartenenza. Il secondo è quello in cui si usa il modello prodotto per assegnare l'attributo di classe più appropriato a nuovi dati non ancora etichettati. Non conoscendo a priori il valore corretto della label, la predizione della classe di ogni esempio è definita in un intervallo di *confidenza*, ovvero secondo un valore che esprime la probabilità di aver classificato correttamente il dato in base alle informazioni conosciute.

Il dataset usato l'apprendimento supervisionato, chiamato *training set*, contiene al suo interno gli esempi già classificati.

Ogni algoritmo di classificazione usa specifica struttura dati per rappresentare e processare in modo strategico i dati del training set al fine di determinare matematicamente il valore dell'attributo di classe.

In base alla struttura dati e alla strategia adottata da ogni algoritmo, le tecniche di classificazione si distinguono nelle seguenti categorie:

- Alberi di decisione
- Regole di classificazione
- Classificatori associativi
- Classificatori bayesiani
- Reti neurali
- Support Vector Machines
- Classificatori instance-based

Per valutare qual è la tecnica più appropriata per il problema di classificazione che si deve affrontare, si considerano i seguenti parametri:

- Accuratezza
  - qualità della predizione
- Efficienza
  - tempo di costruzione del modello (*learning time* o *training time*)
  - tempo di classificazione
- Scalabilità
  - rispetto alla dimensione del training set
  - rispetto al numero di attributi
- Robustezza
  - gestione di dati anomali o mancanti
- Interpretabilità
  - comprensione del modello
  - compattezza del modello

Tenendo conto dei parametri di valutazione indicati, di seguito si descrivono gli aspetti caratteristici di ogni tipologia di algoritmo di classificazione.

Alberi di decisione

Gli alberi decisionali, come sottintende il nome, si basano su un modello rappresentato da una struttura ad albero. Tale struttura viene costruita nella fase di training del modello e viene applicata successivamente per classificare i nuovi esempi.

All'inizio della fase di addestramento del modello si definisce il nodo alla radice dell'albero, che costituisce l'insieme di tutti gli esempi del training set. Attraverso un processo iterativo, si seleziona un attributo e si partizionano i dati iniziali in tanti sottoinsiemi quanti sono i valori dell'attributo. Dopodiché, se tutti gli esempi di un sottoinsieme appartengono alla stessa classe si genera un nodo foglia, altrimenti si definisce un nodo intermedio e di ripete il processo di partizionamento sull'attributo successivo fino all'esaurimento degli attributi disponibili o dei dati di training.

Al termine del training del modello si ottiene una struttura analoga a quella rappresentata in figura 2.2, dove ogni percorso che congiunge la radice dell'albero ai nodi foglia contiene le condizioni di partizionamento sugli attributi che, combinate in un'unica espressione logica, definiscono le regole di classificazione degli esempi.

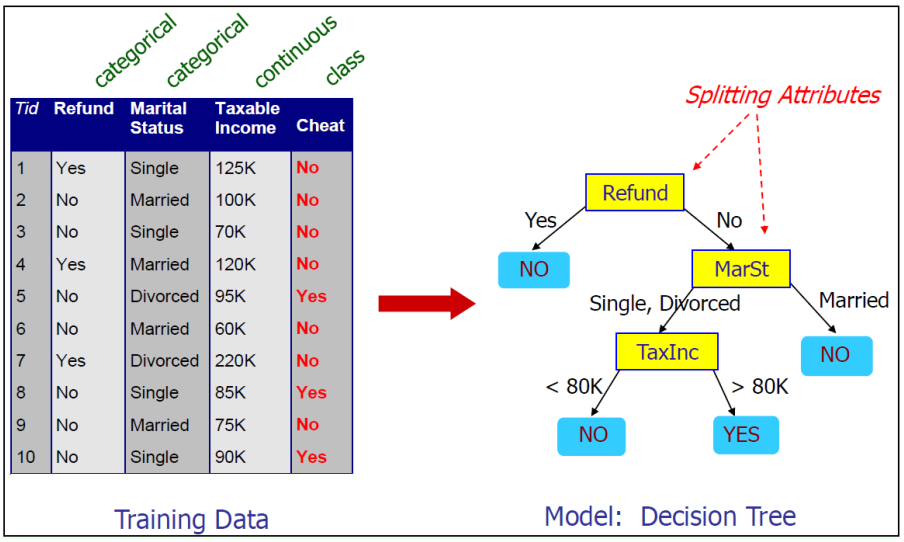

Figura 2.2: Esempio albero decisionale[6]

Dal momento che applicare il modello su nuovi esempi implica processare ogni esempio attraverso l'albero decisionale, per ottimizzare i tempi di classificazione è necessario riuscire a partizionare l'insieme dei dati iniziali attraversando il minor numero di nodi intermedi: pertanto è fondamentale scegliere l'ordine più efficace in cui collocare gli attributi di partizionamento (*splitting attributes*).

Per scegliere il miglior attributo ad ogni iterazione del processo di addestramento, gli algoritmi *decision tree* valutano tutti gli attributi applicando un *criterio di partizionamento* che si basa su due fattori:

- *Information gain*

  - adatto per il confronto di attributi discreti (nominali);
  - misura l'*entropia* per quantificare l'informazione necessaria per classificare il training set prima e dopo aver applicato il partizionamento sull'attributo considerato.

- *GINI index*
  - adatto per il confronto di attributi continui (numerici);
  - valuta tutti i possibili punti di *split* per ogni attributo, al fine di identificare quello migliore.

Per evitare un'eccessiva specializzazione del modello sui dati di training e potenziali problemi di overfitting, al termine della costruzione dell'albero gli algoritmi *decision tree* eseguono un *pruning* dei percorsi non necessari. Gli algoritmi di tipo *random forest*, invece, generano N alberi decisionali distinti e scelgono le classi determinate dalla maggioranza dei modelli: in questo modo generano modelli più robusti e minimizzano anche la varianza degli errori di classificazione.

Gli alberi decisionali generalmente soddisfano la maggior parte dei requisiti di accuratezza, efficienza e scalabilità. Inoltre sono interpretabili con estrema facilità: fattore che determina spesso la scelta di questo tipo di tecniche rispetto ad altre. Alcuni algoritmi, però, non sono in grado di gestire valori mancanti o attributi numerici. In questi casi si deve superare tale limitazione attraverso una preparazione (*preprocessing*) adeguata del training set da sottoporre agli algoritmi.

Regole di classificazione

Le regole di classificazione funzionano in modo analogo agli alberi decisionali in quanto estraggono un pattern che calcola l'attributo di classe attraverso una serie di condizioni sugli attributi. Nel caso dei classificatori basati sulle regole, le condizioni che determinano il valore della classe sono espresse come una combinazione di predicati logici (*if...then*), secondo la definizione "$Regola: \ (condizione) \rightarrow Y$".

Le regole possono essere estratte indirettamente, derivandole da un albero decisionale già costruito, oppure direttamente dai dati di training.

Rispetto agli alberi decisionali, che etichettano i nuovi esempi seguendo le regole nell'ordine in cui compaiono nella struttura ad albero, i metodi rule-based possono commutare l'ordine di predicati logici ed eventualmente rimuovere delle condizioni per generalizzare il modello (e minimizzare potenziali problemi di overfitting): per questo motivo risultano particolarmente veloci nella fase di applicazione del modello.

Nel caso in cui si verifichino situazioni dove un esempio soddisfa contemporaneamente le condizioni definite da regole diverse, ovvero dove viene *coperto* da più regole, per decidere quale classe associare all'esempio, gli algoritmi prevedono due alternative:

(a) scegliere la classe della prima regola che copre l'esempio;

(b) scegliere la classe prevista dalla maggioranza delle regole che coprono l'esempio.

Se le regole identificate non riescono a coprire tutti i nuovi esempi, le istanze rimanenti vengono associate alle classe di maggioranza (come avviene con gli alberi di decisione).

Ad eccezione della migliore efficienza in fase di classificazione, per queste tecniche valgono le stesse considerazioni fatte in merito alla valutazione degli alberi di decisione.

Classificatori associativi

La classificazione associativa si basa sull'applicazione di regole di classificazione in modo analogo a quanto indicato nel punto precedente. La generazione del modello, però, avviene in modo differente, poiché le regole estratte sono regole di associazione adattate alla classificazione.

Come indicato nella sezione 2.1, le regole di associazione identificano le correlazioni tra l'occorrenza di un elemento e quella di altri elementi all'interno del dataset analizzato. Per esprimere tali regole nella forma "$Regola: (condizione) \rightarrow Y$", i classificatori associativi costruiscono il corpo della regola (la parte sinistra) estraendo tutte le combinazioni delle coppie $attributo = valore$ presenti nel training set e lasciando l'attributo di classe nella testa della regola (parte destra). In questo modo possono determinare le relazioni di dipendenza tra le occorrenze degli attributi non di classe e l'attributo di classe.

Rispetto agli alberi decisionali, il metodo adottato dai classificatori associativi richiede più tempo per generare le regole di classificazione e non è applicabile su numeri elevati di attributi. Nonostante ciò, i modelli basati sulla correlazione delle combinazioni più significative di attributi non sono influenzati da valori mancanti e risultano, inoltre, più accurati nella predizione delle classi.

Classificatori bayesiani

I classificatori bayesiani adottano un approccio puramente probabilistico alla classificazione, che è basato sull'applicazione del teorema di Bayes.

In tale contesto, il teorema di Bayes è usato per esprime la probabilità congiunta $P(C|X)$, secondo la formula:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

dove:

C	-	indica la classe a cui appartiene un esempio del training set
X	-	rappresenta il valore degli attributi $\{x_1, \ldots, x_n\}$ di tale esempio
$P(C)$	-	è la probabilità della classe C nota a priori (N_C/N_{TOT})
$P(X)$	-	è la probabilità *costante* dell'insieme di attributi per tutte le C
$P(X\|C)$	-	è la probabilità condizionata degli attributi X, data la classe C.

Dal momento che le probabilità condizionate di ogni attributo x_i rispetto alla classe C non possono essere quantificate in modo esaustivo, il calcolo del termine $P(X|C)$ deve essere approssimato.

Una soluzione semplice a questo problema è quella di ipotizzare che tutti gli attributi $\{x_1, \ldots, x_n\}$ siano statisticamente indipendenti tra loro, così da poter approssimare $P(X|C)$ come la combinazione delle probabilità condizionate dei singoli attributi: $P(x_1|C) \cdot P(x_2|C) \cdot \ ... \ \cdot P(x_n|C)$. In questo caso, tali probabilità sono ottenute contando in quanti esempi lo stesso attributo risulta associato alla stessa classe ($P(x_i|C) = |x_{i_C}|/N_C$).

L'ipotesi relativa all'indipendenza statistica delle variabili X è spesso una forte approssimazione dello scenario reale e, per questo motivo, è definita Naïve (ingenua). Pertanto i classificatori che adottano questa strategia per semplificare il calcolo delle probabilità sono detti *Naïve*.

Altri classificatori bayesiani, invece, applicano l'ipotesi Naïve solo per alcuni attributi e cercano di definire statisticamente l'esistenza di condizioni di dipendenza su pochi attributi significativi. Questi classificatori, qualitativamente migliori, prendono il nome di *reti bayesiane*.

I modelli prodotti dalla classificazione bayeasiana sono generati ed applicati molto velocemente e sono facilmente interpretabili, in quanto permettono di dedurre facilmente quali sono gli attributi che influenzano maggiormente l'assegnazione della classe. Inoltre, essendo basati su modelli statistici, risultano robusti alla presenza di valori anomali o irrilevanti. L'ipotesi Naïve, anche se necessaria, in alcune situazioni potrebbe ridurre significativamente l'accuratezza dei modelli.

Reti neurali

Le tecniche basate su reti neurali, chiamate anche *Neural Networks* o *Artificial Neural Networks* (*ANNs*), definiscono un modello di classificazione ispirato ad alcuni elementi dell'apprendimento del cervello umano. Tale modello prevede, infatti, la classificazione di ogni esempio del dataset attraverso una rete di *neuroni*, connessi tra di loro da *sinapsi*, come mostrato nell'esempio in figura 2.3.

Il valore di ogni attributo, preso in considerazione per la classificazione dell'esempio, viene associato ad un neurone di ingresso, che inoltra tale valore a tutti i nodi intermedi a cui è connesso. I nodi intermedi della rete sono chiamati neuroni nascosti. Ogni neurone nascosto definisce un proprio un vettore di *pesi* ed un valore di *offset*, che usa per trasformare gli attributi in ingresso in un risultato intermedio normalizzato nell'intervallo [0,1]. I valori ottenuti dallo strato nascosto della rete (*hidden layer*) convergono infine sui neuroni di uscita, che determinano la classe da assegnare all'esempio.

Nel caso di classificazione binomiale, ovvero di due possibili etichette, è sufficiente un solo neurone di output per determinare la classe da assegnare, mentre, nel caso di classificazione polinomiale, la rete sarà composta a tanti neuroni di

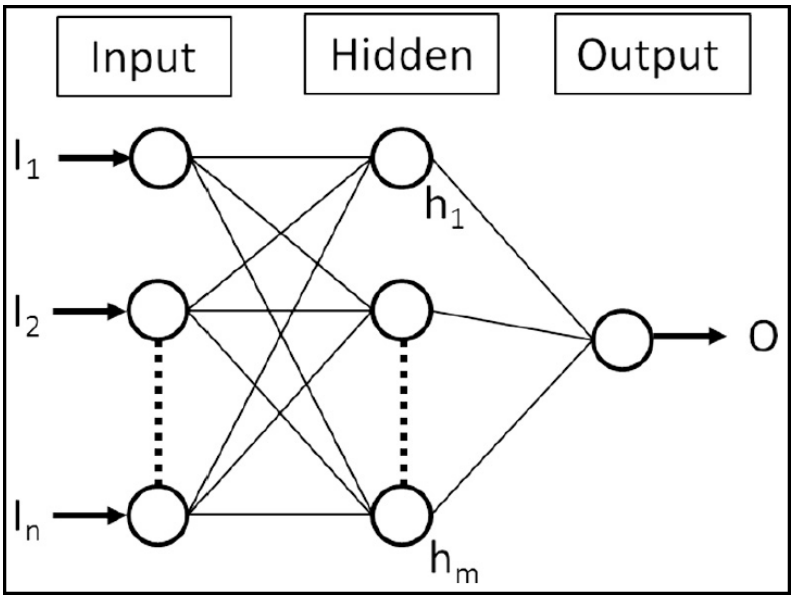

Figura 2.3: Esempio rete neurale[31]

output quanti sono i possibili valori dell'attributo di classe: ogni nodo indicherà la probabilità del valore previsto rispetto agli altri possibili valori della classe.

In fase di generazione del modello, il processo con cui viene classificato un esempio, attraverso la rete, è eseguito per tutte le istanze del training set in modo iterativo e, ad ogni iterazione, si calcola l'errore commesso sulla previsione delle classi. Il valore dell'errore di classificazione è propagato dall'uscita all'ingresso della rete, attraverso un processo definito *back propagation*, il cui scopo è quello correggere i pesi e gli offset di ogni neurone e ottimizzare l'accuratezza del modello di classificazione. Dopo un numero massimo di iterazioni, se non è stato raggiunto il livello di accuratezza desiderato, oppure i pesi e gli offset dei neuroni non devono essere ulteriormente modificati, il processo iterativo si arresta e la configurazione dei neuroni raggiunta definisce il modello di classificazione.

Il fatto che il pattern di classificazione sia definito dall'insieme di pesi e offset non permette di interpretare i modelli prodotti dagli algoritmi basati sulle reti neurali. Nonostante ciò, il metodo iterativo basato sulla back propagation permette a questa tipologia di algoritmi di costruire modelli molto accurati anche a fronte di dati sporchi o anomali, al costo, però, di tempi elevati di training e di potenziali problemi di overfitting che, spesso, sono affrontanti implementando ulteriori meccanismi di ottimizzazione dei pesi della rete.

Infine, l'ultimo aspetto che caratterizza le reti neurali, diversamente dalle altre tecniche di data mining presentate fino ad ora, è la possibilità di utilizzarle per prevedere sia valori discreti che valori numerici, ovvero generando modelli che

possono essere applicati per la regressione[1].

Support Vector Machines

Le *Support Vector Machines* (*SVM*) sono tecniche di classificazione *binomiale* che rappresentano i dati da analizzare come un insieme di punti all'interno di uno spazio multi-dimensionale e mirano a trovare l'*iperpiano* che massimizza la separazione tra l'insieme di punti appartenenti ad una classe e quelli appartenenti all'altra classe.

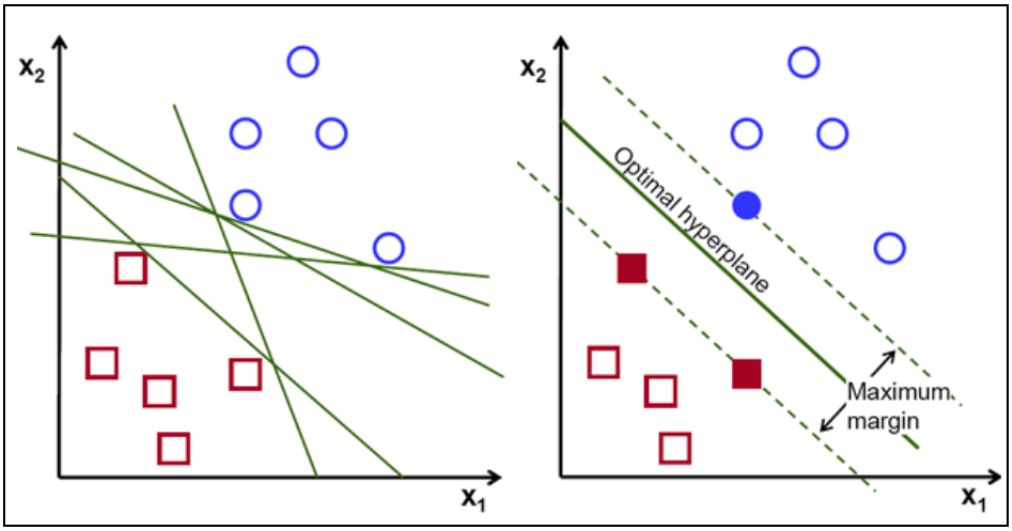

Figura 2.4: Esempio SVM[9]

Come mostrato nella figura 2.4, per costruire un iperpiano avente la maggiore distanza dal punto più vicino di ognuna della classi, si misura il *margine* rispetto a due linee parallele che passano per i punti estremi delle classi, in modo tale da definire i limiti dell'iperpiano da scegliere. I punti più vicini all'iperpiano prendono il nome di *support vectors*, in quanto supportano la misura del margine e la scelta della soluzione ottimale.

La rappresentazione degli esempi nello spazio multi-dimensionale è realizzata tramite l'impiego di *funzioni kernel* che, mappando i valori degli attributi nello spazio scelto, permettono di trasformare i dati di partenza in insiemi di punti linearmente separabili anche nel caso in cui il problema originale non sia lineare. Il tipo di kernel da utilizzare è un parametro degli algoritmi SVM.

Gli algoritmi SVM possono essere anche adattati alla previsione di valori continui implementano metodi di Support Vector *Regression* (*SVR*), dove l'obiettivo diventa

[1]Per la predizione delle variabili continue è necessario un preprocessing del training set volto a normalizzare i valori degli attributi continui in di un range [0,1] o [−1,1].

trovare l'iperpiano che comprende il maggior numero di punti possibile, così da poter applicare su di essi la regressione.

I modelli SVM risultano molto accurati e molto veloci nella classificazione, anche se, come le reti neurali, sono lenti nella fase di learning e non sono interpretabili. Inoltre, la misura del margine, unita ad una funzione di penalizzazione dell'errore di classificazione, rende questo tipo di algoritmi nativamente resistenti all'overfitting.

Classificatori instance-based

I classificatori *instance-based* differiscono da tutte le altre tecniche, perché invece di costruire un modello, memorizzano tutti i dati di training e li usano direttamente per classificare i nuovi esempi. In tal senso sono definiti *lazy learners*, poiché generano il modello dinamicamente per le istanze da classificare sul momento. L'algoritmo più rappresentativo di questa categoria è il *K-Nearest Neighbor* (*K-NN*).

Il K-NN memorizza gli esempi del training set organizzandoli come un insieme di punti e, a fronte di un nuovo esempio, misura la distanza di tutti i punti conosciuti dal quello rappresentato dal nuovo esempio. Quindi, considerando i K punti più vicini, assegna la classe a cui appartiene la maggioranza degli esempi di training. La misura di distanza e K sono parametri dell'algoritmo.

Non disponendo di un modello, la fase di classificazione del K-NN risulta più lenta rispetto alle altre tecniche, in quanto si devono ricalcolare ogni volta le distanze da tutti i punti del training set.

Anche questo algoritmo è applicabile sia per la classificazione che per la regressione (nel secondo caso la previsione del futuro valore continuo è calcolata come la media dei valori dei K esempi vicini).

2.2.1 Validazione dei modelli di classificazione

L'ultima considerazione da fare in merito alla classificazione riguarda il metodo di validazione dei modelli[6]..

La *validazione* di un classificatore è una procedura attraverso la quale si misura la qualità delle previsioni ottenute e si valuta se il modello soddisfa i requisiti del problema in esame. La qualità delle previsioni dipende da *quante* e da *quali* classi sono state assegnate correttamente e viene misurata attraverso diverse metriche, tra cui:

$$\text{Accuratezza} \quad A = \frac{\text{Numero di predizioni corrette}}{\text{Numero totale di predizioni}}$$

$$\text{Richiamo} \quad r = \frac{\text{Numero di predizioni corrette della classe } C}{\text{Numero totale di esempi appartenenti a } C}$$

$$\text{Precisione} \quad p = \frac{\text{Numero di predizioni corrette della classe } C}{\text{Numero totale di esempi } \textit{classificati } C}$$

Tali metriche sono organizzate all'interno di una *matrice di confusione* (*confusion matrix*) che riassume, in forma statistica, le prestazioni del modello di classificazione testato, come mostrato nell'esempio riportato nella tabella 2.1.

Accuratezza: 84.54%			
	Classe *A*	*Classe B*	***Precisione***
Predizione A	87	27	76.32%
Predizione B	5	88	94.62%
Richiamo	94.57%	76.52%	

Tabella 2.1: Esempio confusion matrix

Generalmente si tende a considerare un classificatore migliore di un altro solamente in base all'accuratezza che, però, non è sempre una metrica affidabile. Infatti, in particolari scenari applicativi, dove la classe che si vuole prevedere è rappresentata da un numero di esempi molto inferiore rispetto a quelli delle altre classi, un classificatore riuscirebbe facilmente ad ottenere un elevato livello di accuratezza solamente assegnando i nuovi esempi alla classe di maggioranza. In questi casi, per valutare qual è il modello migliore, è più importante considerare la *precisione* nella classificazione degli esempi appartenenti alla classe di minoranza.

Come verrà approfondito nel capitolo 5, anche per la metodologia applicata in questa tesi la misura di accuratezza non è un requisito fondamentale: per valutare i modelli si utilizza infatti una metrica applicativa relativa al rendimento.

I modelli predittivi estratti dalle tecniche di classificazione si sono dimostrati efficienti nell'analisi dei dati finanziari grazie alla loro capacità di valutare, secondo i metodi descritti in questo capitolo, l'influenza di molteplici attributi su un determinato fenomeno. In particolare sono risultati efficienti nel modellare la natura dinamica e non lineare che caratterizza i mercati azionari. Per esempio, la non linearità è gestita particolarmente bene dalle reti neurali e dalle support vector machines, mentre la componente aleatoria può essere analizzata con i modelli probabilistici implementati dai classificatori bayesiani. Le decisioni di trading possono invece essere facilmente rappresentate per mezzo di regole di classificazione.

Queste considerazioni verranno comunque descritte in modo più approfondito nei capitoli successivi, dove si trattano i temi relativi al financial data mining. Tali temi sono introdotti nel capitolo 3, presentando il mondo del trading quantitativo e descrivendone gli aspetti rilevanti per il lavoro svolto in questa tesi.

Capitolo 3

Il trading quantitativo

Il termine *trading*, letteralmente tradotto in *commercio* o in *scambio*, rappresenta l'attività di compravendita di strumenti finanziari finalizzata ad ottenere un ritorno economico.

Gli strumenti finanziari[11] sono una forma di investimento di natura finanziaria a cui appartengono azioni, obbligazioni, fondi di investimento, materie prime, valute, cripto-valute. Ogni prodotto può essere scambiato operando all'interno del relativo mercato, come ad esempio il mercato azionario per le azioni o il forex per le valute.

Prima della diffusione dei personal computer (PC) e dell'accesso alla rete Internet, lo scambio degli strumenti finanziari poteva essere svolto nei luoghi fisici della *Borsa Valori* (*Stock Exchange*), come la Borsa di commercio di Milano[13]. Oggi invece è possibile utilizzare software specializzati, chiamati *piattaforme di trading*, per disporre direttamente sul proprio PC di un'ampia varietà di mezzi con cui osservare ed operare sui mercati di tutto il mondo in modo autonomo: praticando il trading *online*.

Fare trading online non significa solo accedere ai servizi di un broker via Internet, ma significa anche sfruttare i nuovi strumenti tecnologici per guidare e facilitare le decisioni di investimento. Infatti, l'evoluzione tecnologica ha reso sempre più semplice e veloce operare sui mercati, tanto da dare vita a nuove strategie di investimento, come ad esempio l'*intraday trading* e lo *scalping*, dove si punta a guadagnare velocemente comprando e rivendendo prodotti nel breve termine. Nel caso dell'*intraday trading* tutte le negoziazioni vengono aperte e chiuse entro l'ora di chiusura della borsa, mentre con lo *scalping* si effettuano più transazioni nell'arco di pochi minuti o addirittura secondi. In questi scenari, per scegliere continuamente ed efficacemente i prodotti su cui investire, è necessario elaborare un'elevata quantità di informazioni e prendere decisioni rapidamente: cosa irrealizzabile senza l'aiuto di sistemi automatici di analisi che sfruttano la potenza elaborativa dei computer.

Il data mining, che per definizione usa tecniche automatiche di analisi (come visto nel capitolo 2), trova perfettamente applicazione in questo scenario, permettendo

di elaborare i dati finanziari in modo meccanico e di sfruttare nuove modalità di esplorazione e di predizione dei trend del mercato.

L'approccio meccanico all'analisi dei dati, dove i trader ricorrono all'aiuto di sistemi automatici per estrarre regole e strategie sistematiche di investimento, è definito *trading quantitativo*.

Il trading quantitativo può essere adottato come alternativa o come metodo complementare al *trading discrezionale*, dove, invece, ogni scelta di investimento è affidata alle sole conoscenze e capacità di analisi del trader.

3.1 Confronto tra trading discrezionale e quantitativo

La differenza principale tra l'approccio quantitativo e quello discrezionale consiste nella modalità operativa: nel primo caso, la decisione di comprare o vendere uno strumento finanziario è dettata dai segnali generati algoritmicamente da un *trading system*, mentre, nel secondo, ogni operazione è frutto dell'osservazione del mercato e della valutazione *soggettiva* del trader[17].

Tipicamente i trader discrezionali prendono in considerazione tutte le informazioni rilevanti ai fini di un'operazione basandosi sui principi dell'*analisi fondamentale* e dell'*analisi tecnica* del mercato.

L'analisi fondamentale mira a definire il valore intrinseco di uno strumento finanziario concentrandosi su aspetti macro-economici[18]. Molti di questi aspetti sono descritti da indicatori sintetici (come il PIL di una nazione, il tasso di disoccupazione, il rapporto tra domanda e offerta, ecc.), altri riguardano notizie di mercato, eventi politici ed ulteriori elementi che possono potenzialmente influenzare il prezzo dei prodotti finanziari.

L'analisi tecnica, diversamente, assume che il prezzo degli strumenti finanziari sia in grado di riflettere tutte le informazioni già disponibili sul mercato[19] e, pertanto, si focalizza sullo studio delle variazioni storiche dei prezzi e dei volumi[1] scambiati per determinare se e quando tali variazioni potranno verificarsi nel futuro[20]. Lo studio tecnico dei dati storici è principalmente svolto mediante l'uso di grafici e indicatori statistici, ottenuti applicando formule matematiche, che permettono di modellare alcuni comportamenti derivanti dall'osservazione del mercato[10].

I trader che basano le loro decisioni sull'analisi fondamentale cercano di stimare se uno specifico strumento finanziario è sopra o sotto stimato e operano facendo principalmente investimenti a lungo termine. Al contrario, i trader che si concetrano maggiornamente sui fattori tecnici operano nel breve termine[21], cercando di

[1] Per *volume* si intende il numero totale di negoziazioni che sono avvenute per uno strumento finanziario in uno specifico intervallo di tempo.

intercettare prontamente i trend rializisti o ribassisti dei prezzi. Le due metodologie, comunque, possono spesso essere usate anche congiuntamente, poiché l'analisi fondamentale fornisce le informazioni necessarie ad identificare *quali* sono i prodotti da negoziare, mentre l'analisi tecnica supporta meglio la scelta di *quando* comprare o vendere un determinato prodotto.

Il metodo applicato algoritmicamente dai trading systems è basato, invece, sull'*analisi quantitativa*.

L'analisi quantitativa ripropone l'approccio statistico allo studio dei dati finanziari affiancandovi il contributo degli algoritmi e dei modelli di data mining, per mezzo dei quali cerca di costruire un sistema ottimizzato per valutare le scelte di trading.

Ottimizzare i modelli quantitativi a fronte della natura dinamica e non deterministica dei mercati rappresenta il problema principale dei sistemi automatici di trading, che rischiano di incorrere in situazioni di *overfitting* diventando inefficienti nel lungo termine[22]. Questo problema richiede ai trader sistematici di validare costantemente le prestazioni del sistema scelto tramite attività di *backtesting* e induce gli analisti finanziari ad impiegare metodi euristici e procedure complesse per adattare i modelli alla variabilità delle condizioni di mercato.

Quindi, considerando questi problemi da affrontare, si può affermare che, sebbene il trading quantitativo da una parte possa ridurre l'incidenza dei fattori emotivi sulle scelte di trading ed evitare agli investitori di analizzare manualmente i dati finanziari, dall'altra richiede una particolare attenzione nella configurazione del sistema di trading da utilizzare. Anche se le piattaforme di trading offrono numerose possibilità di backtesting, resta comunque complesso per l'utente finale (non esperto di analisi di dati) scegliere correttamente quale sistema e quali parametri utilizzare.

Per trarre vantaggio dal meglio dei due mondi (discrezionale e quantitativo), esiste una terza modalità operativa adottata dai trader, ovvero quella del *trading semi-automatico* (o *semi-discrezionale*)[23].

L'approccio semi-automatico prevede due possibili utilizzi di un trading system. Nel primo caso si sfrutta il sistema a monte per filtrare le possibili scelte di investimento, in modo da concentrare l'analisi discrezionale solo sugli strumenti finanziari selezionati. Nel secondo caso i trader possono, per esempio, usare l'analisi tecnica per individuare le condizioni di entrata nel mercato e, in seguito, affidare la gestione della posizione alle logiche meccaniche del sistema di trading.

Questa tesi si basa sul primo tipo di approccio semi-automatico, ovvero quello in cui il sistema meccanico non viene impiegato per la generazione di segnali di trading, ma per la selezione di strumenti finanziari, nella fattispecie di azioni. Lo scenario applicativo del trading nel mercato azionario è descritto nella prossima sezione.

3.2 Trading nel mercato azionario

In questa sezione si introducono i termini base usati nell'ambito del trading di azioni e si definiscono gli obiettivi che il financial data mining (che in questo specifico contesto viene anche chiamato *stock data mining*) deve essere in grado di raggiungere per supportare i trader nelle scelte di investimento, mettendo in evidenza gli elementi su cui si focalizza il lavoro svolto in questa tesi.

3.2.1 Terminologia

Per chiarire il significato delle espressioni a cui si farà riferimento nei prossimi argomenti trattati, di seguito si elencano le definizioni di alcuni termini strettamente legati al trading e ai mercati azionari.

- *Mercato azionario*

  È il servizio tramite il quale si possono scambiare (comprare e vendere) le *azioni* delle società quotate in borsa. Sono esempi di mercato azionario:

  – Borsa Italiana
  – London Stock Exchange
  – NASDAQ (titoli delle aziende statunitensi)

- *Azione* (*Stock*)

  È un titolo rappresentativo di una quota della proprietà di una società per azioni. Il possessore dell'azione è detto azionista ed è ha diritto a ricevere una quota dei profitti dell'azienda sotto forma di dividendi, nonché la facoltà di vendere la quota sul mercato azionario[14]. Alcuni esempi dei titoli di aziende quotate nella Borsa Italiana sono:

  – ENI.MI (Eni S.p.A.)
  – FCA.MI (Fiat Chrysler Automobiles N.V.)
  – UNI.MI (Unipol Gruppo S.p.A.)

- *Indice azionario* (*Stock market index*)

  È un indice che sintetizza il valore di un paniere di specifici titoli azionari di un mercato[15]. Tra gli *stock market index* ci sono ad esempio:

  – FTSE MIB:
    indice di 40 società italiane con maggiore valore di mercato
  – NASDAQ-100:
    indice delle 100 maggiori società del mercato NASDAQ
  – SP&500:
    indice delle 500 maggiori società statunitensi

- *Aprire e Chiudere un trade*

  Nel momento in cui un trader *entra nel mercato* per comprare o vendere strumenti finanziari, si dice che *apre una posizione* (detta anche ***trade***). La posizione resta "aperta" fino all'*uscita dal mercato*, ovvero quando il trader *chiude la posizione*.

  L'entrata e l'uscita dal mercato dipendono dall'andamento del valore delle azioni scambiate: gli stock per cui si prevede un rialzo dei prezzi vengono *comprati*, per poter poi essere rivenduti prima che il prezzo cali (così da ottenere il maggior profitto possibile dallo scambio). Analogamente, gli stock per cui si prospetta un *ribasso* dei prezzi sono *venduti allo scoperto*, per poter essere ricomprati prima che il prezzo torni a crescere. Secondo quanto enunciato, il ritorno di un *trade* è calcolato come:

  $$return = \frac{\mid closePrice - openPrice \mid}{openPrice} * amount$$

  dove:

  $closePrice$ = prezzo dello stock alla chiusura della posizione
  $openPrice$ = prezzo dello stock all'apertura della posizione
  $amount$ = denaro investito per l'apertura del trade.

- *Intraday trading* (o *day trading*)

  È una modalità operativa che prevede l'apertura e la chiusura di una o più posizioni nella stessa giornata. Anche se tipicamente viene svolto su più mercati finanziari, in questo contesto si fa riferimento alla sua applicazione nello scambio di azioni.

- *Broker*

  È la figura che opera come intermediario tra gli investitori ed i mercati: gestisce le transazioni tra un acquirente ed un venditore, guadagnando una *commissione* alla chiusura dello scambio.

  Il broker può essere una persona fisica o un'agenzia d'affari, ma nel mondo del trading online è rappresentato dalla piattaforma di trading, detta anche *broker online*.

  Dal momento che il broker riceve una commissione, per la sua funzione di mediazione tra domanda e offerta, per calcolare in modo più realistico il ritorno di un investimento, occorre includere nella formula indicata precedentemente anche il costo di ogni transazione:

  $$return = \frac{\mid closePrice - openPrice \mid}{openPrice} * amount - transaction\_cost$$

- *Stock portfolio*

  Rappresenta l'insieme di azioni possedute da un investitore e sottintende anche il fatto che sia stata seguita una particolare strategia per scegliere tali azioni, per esempio distribuendo e diversificando le quote in base al settore di mercato, oppure in base alla volatilità e al rischio.

- LONG-*selling trade* e SHORT-*selling trade*

  Quando un trader entra nel mercato *acquistando* un titolo, si dice che apre una posizione LONG, se invece sceglie la strategia della *vendita allo scoperto*, si dice che apre una posizione SHORT. Quindi:

  - previsione di un rialzo del prezzo ⇒ LONG-*selling trade*
    - se $(closePrice > openPrice) \Rightarrow profit$ (guadagno)
    - se $(closePrice < openPrice) \Rightarrow loss$ (perdita)
  - previsione di un ribasso del prezzo ⇒ SHORT-*selling trade*
    - se $(closePrice > openPrice) \Rightarrow loss$ (perdita)
    - se $(closePrice < openPrice) \Rightarrow profit$ (guadagno).

- *Stop Loss*

  È un parametro configurabile sulle piattaforme di trading per minimizzare le perdite, chiudendo automaticamente le posizioni aperte quando il prezzo del titolo scambiato scende (caso LONG) o sale (caso SHORT) oltre una soglia massima:

  - LONG-*selling trade*
    $\Rightarrow stopLossValue = openPrice - stopLoss\%$
    - se $currentPrice \leq stopLossValue \Rightarrow$ close trade!
  - SHORT-*selling trade*
    $\Rightarrow stopLossValue = openPrice + stopLoss\%$
    - se $currentPrice \geq stopLossValue \Rightarrow$ close trade!

Chiaramente il mondo del trading non è limitato ai termini elencati, ma è bensì molto più articolato e complesso. Per i temi affrontati in questa tesi, è comunque sufficiente prendere in considerazione solamente i concetti appena definiti, i quali saranno utilizzati per approfondire lo scenario applicativo dell'intraday trading, definito anche come *stock intraday trading*.

3.2.2 Stock intraday trading

Investire sulle azioni in modalità intraday, ovvero *giornalmente*, comporta le stesse regole e gli stessi aspetti indicati nella sezione 3.2, ma introduce ulteriori considerazioni in merito allo stretto intervallo di tempo in cui tali investimenti sono eseguiti.

Prima di tutto, per analizzare i dati finanziari e determinare quali decisioni di trading prendere, è necessario osservare il mercato attraverso una finestra temporale ridotta ai giorni precedenti a quello in cui si vuole operare. Infatti, gli andamenti e i comportamenti osservati nel breve periodo potranno facilmente continuare a verificarsi anche nei giorni successivi e risulteranno più attendibili di un'analisi svolta in un periodo più ampio, che potrebbe lasciarsi sfuggire pattern relativi a brevi intervalli di tempo.

In secondo luogo, nell'arco di una giornata anche le oscillazioni dei prezzi degli stock saranno contenute e, pertanto, produrranno rendimenti relativamente minori rispetto a quelli ottenibili a lungo termine.

Tipicamente i trader intraday cercano di incrementare i profitti eseguendo più operazioni nello stesso giorno, fino a raggiungere un obiettivo di guadagno prefissato, determinato dalla somma dei ritorni ottenuti da tanti piccoli investimenti.

Questa operatività, chiaramente, comporta un maggiore rischio, in quanto aumenta la probabilità di commettere errori e perdere il denaro investito[2], tanto è vero che, se tanti piccoli successi determinano un guadagno discreto, tanti errori, sommati, conducono potenzialmente a gravi perdite. Inoltre, occorre tenere conto anche del costo di ogni transazione che, sebbene sia minimo, applicato su tante piccole operazioni, può avere un impatto notevole sul profitto complessivo.

Quindi, riassumendo, per riuscire a massimizzare il successo degli investimenti e limitare le perdite, i trader intraday devono riuscire ad ottimizzare il numero di operazioni giornaliere sulla base di analisi svolte su dati temporalmente vicini al momento in cui si sceglie di operare. Per ottimizzare il numero di operazioni è fondamentale indirizzare le scelte di investimento sulle sole azioni per cui si prospetta un'alta probabilità di successo e un alto rendimento.

Il sistema semi-automatico alla base di questo progetto di tesi è orientato su quest'ultimo obiettivo, ovvero supportare i trader intraday nella scelta degli stock più promettenti. In particolare gli stock sui quali si raccomanda di concentrare l'attenzione sono quelli per cui, applicando tecniche di data mining, si prevede una variazione significativa del prezzo nella successiva giornata di trading.

Dopo aver filtrato le azioni più interessanti, gli investitori dovranno solamente occuparsi di valutare il momento migliore per comprare o vendere. A tale scopo, potranno scegliere se applicare un approccio discrezionale, oppure se fare affidamento su un sistema che scateni i segnali di trading in modo automatico.

Considerando gli aspetti del trading nel mercato azionario descritti in questa sezione, nel capitolo 4 si illustra lo stato dell'arte dello stock data mining, presentando alcuni esempi delle metodologie esistenti e dei possibili approcci quantitativi sviluppati per facilitare le decisioni di investimento.

[2]Secondo i dati della piattaforma eToro, nel 2016 circa l'80% degli iscritti ha perso soldi nel giro di un anno[16].

Capitolo 4

Analisi stato dell'arte

Il mercato azionario, come già anticipato, è caratterizzato da un comportamento dinamico ed aleatorio a causa dell'influenza di molteplici fattori che includono sia aspetti macro-economici sia elementi connessi alle aspettative e alle reazioni degli investitori. Scegliere le azioni e il momento più opportuno per investire non richiede solo analizzare una vasta quantità di dati, ma anche individuare le variabili più significative per affrontare i cambiamenti non deterministici del mercato.

In questo capitolo si presentano alcuni approcci tipici dello stock data mining, utilizzati per analizzare le informazioni relative alle azioni e per supportare i trader nelle scelte di investimento. Ogni metodologia affronta il problema da una diversa prospettiva, concentrando l'analisi dei dati su specifiche caratteristiche del mercato azionario. Tra questi metodi, si descrive in modo più approfondito l'approccio proposto dal DAUIN del Politecnico di Torino con il framework Best Stock Finder, che sarà il punto di partenza del lavoro svolto in questo progetto di tesi.

4.1 Approcci esistenti

Tra gli approcci descritti di seguito, si possono considerare quattro parametri rappresentativi del modo in cui ogni approccio concepisce il problema di data mining e determina la relativa soluzione:

- Dati di input

  I dati di input costituiscono l'insieme di informazioni usate per costruire il modello di data mining, ovvero le caratteristiche del mercato su cui ogni metodo concentra l'analisi per identificare i pattern in grado predire i movimenti che si verificheranno in futuro.

  Essenzialmente tutti i metodi analizzano sempre i prezzi storici degli stock ma, in molti casi, considerano anche dati aggiuntivi, come indicatori statistici

o eventi che possono influire sull'andamento del mercato e delle scelte degli investitori.

- Orizzonte temporale degli investimenti

  I metodi proposti possono essere indirizzati verso strategie di investimento a lungo termine, a breve termine o entrambe. Nel primo caso si mantengono le posizioni aperte per più anni per trarre profitto dal crescere di valore dei titoli acquistati; nel secondo, invece, si cerca di guadagnare eseguendo più negoziazioni in uno stretto lasso di tempo. Il tipo di strategia è un elemento che caratterizza l'obiettivo del problema di data mining ed il tipo di output del modello realizzato.

- Tecnica/tecniche di data mining applicate

  Il modello che dovrà approssimare il movimento dei prezzi può essere costruito tramite tecniche statistiche, di machine learning o intelligenza artificiale. In alcuni casi si applicano contemporaneamente più tecniche combinando i diversi risultati ottenuti oppure affiancando ai metodi di data mining il contributo di alcuni strumenti di cui si serve l'analisi tecnica.

- Dati di output

  Il tipo di output rappresenta infine il pattern e la conoscenza che deriva da esso. Per gli investimenti sul lungo periodo, i modelli estraggono tipicamente un *portfolio* di stock, opportunamente diversificati per gestire il rischio di perdite, mentre, nel caso del trading a breve termine, il modello estrae il prezzo futuro di uno stock (*single time series forecasting*), oppure assume la forma di *segnali di trading*, che esprimono la decisione da prendere a fronte di determinati eventi, per esempio "COMPRA/VENDI *lo stock X se il prezzo diminuisce/cresce dell'*1%".

I parametri indicati sono riportati nella tabella 4.1 per riassumere le categorie in cui si distinguono le metodologie presentate.

Il primo metodo affrontato è quello descritto nell'articolo [26], che rappresenta un esempio *media-aware trading*, basato sulla componente psicologica degli investitori. In questo caso l'aspetto *emozionale* dei trader viene codificato estraendo i termini attinenti ed eventi ed opinioni (*sentiment analysis*) da dati raccolti su social network e siti che pubblicano notizie di mercato, per poi valutare le variazioni di prezzo avvenute nell'intervallo di tempo successivo alla pubblicazione di tali informazioni sul web. La struttura dati che include i movimenti di prezzo e la codifica delle emozioni è usata per costruire un modello SVR (*Support Vector Regression*), attraverso il quale si prevede il valore futuro del prezzo degli stock in funzione del *sentimento* degli investitori.

Diversamente da questo esempio, i metodi che seguono i principi dell'analisi tecnica, si basano esclusivamente sull'osservazione dei valori storici assunti dagli

Ref.	Input	Periodo	Tecnica	Output
[26]	Web media e prezzi stock al secondo	Breve	Sentiment analysis e SVR	Time series forecasting (trend direction)
[27]	Prezzie volume, indicatori tecnici	Breve	Pattern mining	Segnali di trading
[28]	Prezzi e volume, indicatori tecnici	Breve	Rule-based fuzzy model	Segnali di trading
[29]	Prezzi e volume, indicatori tecnici	Breve	K-Nearest Neighbor (K-NN)	Segnali di trading
[30]	Prezzi e volume, indicatori tecnici	Breve	Support Vector Machines (SVM)	Segnali di trading
[31]	Prezzi e volume, indicatori tecnici	Breve	Bayesian Neural Network	Time series forecasting
[32]	Prezzi e indicatori macro-economici	Lungo	Bayesian Factor Graph	Time series forecasting (trend direction)
[33]	Prezzi e volume	Lungo e breve	Genetic Network Programming (GNP)	Stock portfolio
[34]	Prezzi e volume, indicatori tecnici	Lungo	Genetic algorithm (GA)	Segnali di trading
[35]	Indicatori tecnici	Breve	ANOVA-ANN, Genetic Algorithm (GA)	Segnali di trading
[36]	Prezzi e volume	Lungo	Pattern mining, ANN, GA	Segnali di trading
[37]	Prezzi e volume, indicatori tecnici	Breve	SVR, ANN, Random Forest	Segnali di trading
[38]	Indicatori macro-economici	Lungo	Ensemble ANN, Decision Tree, Logistic Regression	Trend direction
[39]	Prezzi e volume, indicatori tecnici	Breve	Ensemble decision trees	Segnali di trading

Tabella 4.1: Classificazione approcci esistenti

stock, ed eventualmente su indicatori statistici derivati da essi, per comprendere e determinare i trend emergenti di mercato.

Il metodo [27], per esempio, studia i prezzi storici imitando i modelli grafici usati nell'analisi tecnica. Per fare ciò, rappresenta la serie temporale dei prezzi degli

stock come un grafico in cui ricercare automaticamente, tramite tecniche di *pattern matching*, gli schemi che gli analisti tecnici tipicamente associano ad un segnale rialzista (*bull-flag pattern*). I punti determinati da questi schemi, sono quelli in cui si verifica un'inversione del trend e sono quelli che *segnalano* il momento migliore per entrare nel mercato, in quanto promettono di massimizzare i profitti.

Sempre ispirandosi alle tecniche basate sull'osservazione dei grafici, in [28] si propone un sistema basato sull'analisi delle candele giapponesi, che costituiscono uno strumento ampiamente usato in analisi tecnica per guidare le decisioni di investimento sulla base dei cosiddetti *candlestick pattern*[24]. L'interpretazione di questi pattern è molto soggettiva e, secondo l'approccio proposto, non può essere approssimata da semplici regole binarie *compra/vendi*. Pertanto si cerca di ottimizzare la scelta dei *pattern candelstick*, modellando le informazioni rappresentate dalle candele per mezzo delle variabili appartenenti alla logica fuzzy[25], che introducono nel modello decisionale, e nelle risultanti regole di trading, una misura probabilistica di affidabilità di ogni pattern, grazie alla quale si può fornire ai trader una metrica da valutare per impostare la quantià di denaro da investire, oltre che l'indicazione del momento in cui entrare nel mercato.

Le soluzioni proposte in [29, 30, 31], invece, provano a costruire i modelli predittivi sfruttando gli indicatori statistici dell'analisi tecnica. In [29], per esempio, si usa un algoritmo K-NN (*K-Nearest Neighbor*) per identificare la prossima variazione del valore degli stock servendosi di alcuni degli strumenti tecnici più noti (*stop loss*, *stop gain* e *RSI*[1]), mentre, in [30], lo stesso tipo di classificazione viene realizzato tramite un algoritmo SVM (*Support Vector Machines*), che analizza diversi indicatori derivabili dai prezzi degli stock ed ottimizza l'accuratezza delle predizioni introducendo un peso legato al volume di transazioni. In [31], invece, gli indicatori costituiscono l'input di una rete neurale (ANN) che sfrutta un algoritmo di regolarizzazione bayesiana per penalizzare automaticamente la scelta delle soluzioni più complesse.

Anche la metodologia proposta in [32] analizza i fenomeni del mercato attraverso le metriche fornite dagli indicatori statistici, in questo caso, però, fa affidamento su indicatori macro-economici (più affini agli aspetti considerati nell'analisi fondamentale). Nello specifico, gli indicatori sono rappresentati come un insieme di fattori che descrivono la situazione economica in diversi periodi di tempo. Per ogni periodo di tempo, si costruisce un grafo aciclico che codifica la distribuzione congiunta di probabilità tra i diversi fattori e, osservando i cambiamenti nella topologia di ogni grafo prodotto, si riesce a determinare automaticamente se il trend di ogni periodo cambia o resta invariato.

[1] *Relative Strength Index*: indice di misura delle oscillazioni massime del prezzo.

Senza ricorrere all'ausilio di indicatori statistici, in [33] si sfrutta un algoritmo di intelligenza artificiale per gestire un portfolio di stock in modo automatico. Come visto in altri esempi, il problema di data mining è definito in termini di classificazione binaria di semplici *regole di trading*: "se (condizione) → *compra/vendi*". In questo caso le regole sono generate e simulate per mezzo della programmazione genetica (GNP[2]), che valuta l'efficacia delle soluzioni attraverso metriche di guadagno/rischio.

Pur essendo rivolto alla gestione del portfolio per sole posizioni LONG, questo approccio offre la flessibilità di adattarsi anche agli investimenti a breve termine: infatti gli stock classificati per l'acquisto possono restare nel portfolio per lunghi periodi di tempo, oppure essere venduti e comprati da un giorno all'altro in base alle regole generate.

Un altro approccio basato sull'apprendimento di tipo evoluzionistico per estrarre *regole di trading* è proposto in [34], dove, al posto della programmazione genetica, si sceglie di implementare un *algoritmo genetico* (GA[3]). Come nell'esempio precedente, si cerca di evolvere la risposta al problema selezionando solo le soluzioni che soddisfano determinati requisiti, in funzione del guadagno ottenuto ed del rischio degli investimenti. In questo caso, però, si sfrutta la capacità dell'algoritmo di risolvere efficacemente problemi di ottimizzazione anche per migliorare la selezione e la discretizzazione degli attributi iniziali in sottoinsiemi tali da generare regole di trading ottimali.

Lo stesso principio di ottimizzazione viene utilizzato anche in [35] per combinare efficacemente le previsioni ottenute da più tecniche di classificazione addestrate a risolvere lo stesso problema. In questo caso l'algoritmo genetico realizzato ha l'obiettivo di selezionare le combinazioni che permettono di ridurre la varianza dei diversi errori di classificazione e conseguire un livello di accuratezza complessivamente migliore di quello ottenuto dai singoli classificatori.

Analogamente a quest'ultimo esempio, altri diversi approcci, come [36, 37, 38, 39], hanno dimostrato che la combinazione di più tecniche determina risultati migliori, sia in termini di accuratezza che di ritorni sugli investimenti.

In [36] si descrivono quattro possibili architetture utili a realizzare sistemi di apprendimento basati su più classificatori[4]:

- Condizionale
  Si applica un primo classificatore e, solamente se questo fallisce o non produce predizioni con un minimo livello di confidenza, si usa un metodo alternativo.

[2]GNP: *Genetic Network Programming*.

[3]GA: *Genetic Algorithm*.

[4]Anche se in [36], si fa riferimento a sistemi multi-classificatore, le architetture descritte sono generalmente applicabili a diverse tecniche di data mining.

- Seriale
  I classificatori sono applicati in successione: ogni risultato intermedio diventa l'input del classificatore successivo.
- Ibrida
  Il risultato di un primo classificatore è usato per scegliere quale metodo usare successivamente.
- Parallela (ensemble learning)
  Si applicano simultaneamente più classificatori e le predizioni di ogni metodo sono combinate per produrre una decisione finale. Quando la soluzione finale viene scelta attraverso un meccanismo di voti (con o senza peso), il sistema realizzato è detto *ensemble classifier*.

Secondo queste configurazioni, la metodologia citata in [35] è riconducibile all'architettura parallela.

La strategia proposta in [37], invece, si basa su una topologia seriale composta da due tecniche applicate sequenzialmente: SVR nella prima fase e tre diversi modelli, proposti alternativamente, nella seconda. La prima tecnica essenzialmente esegue una previsione (*forecasting*) di dieci indicatori tenici ad una distanza variabile di N giorni. Il valore futuro di questi indicatori è poi processato alternativamente da una rete neurale (ANN), un albero decisionale di tipo Random Forest (RF) e nuovamente SVR, per estrarre il prezzo di chiusura di un singolo stock attraverso tre modelli seriali indipendenti: SVR-ANN, SVR-RF e SVR-SVR.

Infine, i metodi proposti in [38, 39], rappresentano veri e propri esempi di *ensemble learning*, dove si applicano più classificatori in parallelo, per poi decidere quali sono le migliori predizioni da considerare.

In [38], si analizzano le prestazioni raggiungibili impiegando *majority voting* e *bagging* come meccanismo di scelta delle decisioni ottenute da tre diversi classificatori, confermando, in entrambi i casi, la superiorità dei modelli di insieme rispetto ai modelli applicati singolarmente.

Il *majority voting* rappresenta il metodo più semplice per combinare i classificatori, in quanto assegna un voto ad ogni predizione estratta da ogni modello e decide il risultato della classificazione finale scegliendo le predizioni aventi la maggioranza dei voti. Il *bagging*, invece, applica più campionamenti sui dati di training generando, per ogni campionamento eseguito, diversi modelli indipendenti, che vengono poi combinati con metodi simili al majority voting.

Infine, l'approccio *ensemble* proposto in [39] applica una particolare strategia per combinare i modelli di più classificatori basata sul concetto di *predizione distribuita*. In questo caso l'approccio parte dal presupposto che il rumore intrinseco dei dati del mercato azionario non permetta di estrarre pattern efficaci applicando un unico modello sull'intero dataset analizzato e che, quindi, sia necessario individuare dei sotto-modelli con cui, separatamente, si riescano ad ottenere prestazioni ottimali su sottoinsiemi ridotti dei dati iniziali. In tal senso, il metodo di *predizione*

distribuita proposto, seleziona e combina i modelli generati da più alberi di regressione addestrati su diverse porzioni del dataset iniziale, implementando la logica di partizionamento dei dati e di combinazione per mezzo di un ulteriore albero di regressione, che prende in considerazione le metriche relative alle prestazioni ottenute dai classificati sui diversi sottoinsiemi di dataset e diversi indicatori tecnici, rappresentativi delle condizioni di mercato legate ad ogni porzione di dati.

Quest'ultimo caso rappresenta un esempio di come combinare i risultati ottenuti da più tecniche sfruttando un ulteriore modello di classificazione.

Come si può notare dagli esempi descritti, diverse metodologie cercano di costruire il modello predittivo concentrandosi su un particolare aspetto del mercato azionario, proponendo soluzioni che possono essere comprese e spiegate dalla teoria del settore; altre, invece, adottano un approccio puramente *data-driven*, modellando i dati in modo da ottenere risultati funzionali, ma non sempre facilmente interpretabili. I metodi [27, 28], per esempio, rielaborano i pattern definiti dallo studio dei grafici derivato dall'analisi tecnica, mentre gli approcci che impiegano algoritmi di apprendimento evoluzionistico ([33, 34]) o la combinazione di più tecniche ([35, 37, 36, 38, 39]), si basano esclusivamente sulla costruzione di modelli ottimizzati in termini di accuratezza e di ritorni sugli investimenti.

Il sistema proposto in questa tesi sceglie l'approccio data-driven, basandosi esclusivamente sui prezzi giornalieri delle azioni, e si dedica a previsioni utilizzabili per l'intraday trading: pertanto, si distingue dai metodi di media-aware trading ([26]), di analisi dei pattern grafici ([27, 28]) e dai metodi [32, 33, 34, 36, 38], che propongono strategie di investimento a lungo termine. A differenza, invece, delle tecniche utilizzabili nel breve periodo, il metodo proposto in questa tesi non viene impiegato per generare segnali di trading o per prevedere il prezzo futuro degli stock, ma bensì per fornire un portfolio di raccomandazioni relative agli stock da seguire nella successiva giornata di trading.

Come verrà approfondito nel capitolo 5, ci sono aspetti condivisi sia dal metodo proposto che da alcuni esempi indicati in questa sezione, tra cui, soprattutto, la scelta di dedicare un componente del sistema di trading alla simulazione delle strategie di investimento ([30, 34]) e la scelta di combinare più tecniche per generare un modello più robusto ed adattabile alla variabilità delle condizioni di mercato. In particolare, per ottenere un modello migliore, si cerca di individuare quale classificatore, tra più candidati, è più appropriato a formulare le previsioni per un determinato periodo, in modo analogo a quanto visto in [39], dove si valutano e combinano le previsioni di diversi modelli addestrati su diverse porzioni del training set inziale. Nella metodologia proposta, oltre a valutare i singoli modelli, si simulano anche le prestazioni dei classificatori a fronte di diverse combinazioni di parametri, tra i quali la modalità di discretizzazione degli attributi iniziali ed anche la dimensione della finestra temporale usata per il training che, come dimostrato da [34], può influenzare significativamente la qualità dei risultati ottenuti.

Per entrare nel merito del funzionamento del sistema proposto, è necessario presentare il sistema BEst STock Finder, che rappresenta la base della sperimentazione svolta in questa tesi.

4.2 Best Stock Finder

BEst STock Finder (BeSt)[40] è un sistema di stock data mining, realizzato dal Dipartimento di Automatica e Informatica (DAUIN) del Politecnico di Torino, che si rivolge allo scenario applicativo dell'intraday trading.

Come indicato nella sezione 3.2.2, per avere successo nell'intraday trading è necessario massimizzare il rendimento di tanti piccoli investimenti, eseguiti su base giornaliera o nell'arco dello stesso giorno, facendo attenzione a limitare il numero di operazioni ai soli strumenti finanziari per cui si prospetta un'alta probabilità di successo. Decidere costantemente su quali azioni investire, secondo questi requisiti, è però un compito molto impegnativo ed è il problema che BEst STock Finder mira a risolvere, proponendo un sistema automatico per la *raccomandazione* degli stock da considerare in ogni sessione di intraday trading.

Ogni azione viene raccomandata solo se si prevede che, nel breve termine, subirà una variazione significativa del prezzo. Per estrarre questa previsione, BEst STock Finder impiega un approccio che si distingue dalle altre metodologie di stock data mining, in quanto non considera solamente l'andamento dei prezzi del titolo interessato, ma analizza anche i prezzi storici di altre azioni appartenenti allo stesso mercato. In tal senso, si cerca di addestrare il pattern predittivo a trovare una correlazione tra la variazione di prezzo di uno stock e quella degli altri stock.

I modelli generati dal sistema sono ottenuti applicando due tecniche di data mining ortogonali[5], rispettivamente basate sulla regressione e sul sequence mining (il cui funzionamento è stato descritto nella sezione 2.1), che analizzano la serie temporale dei prezzi di chiusura giornalieri, per determinare di quanto varieranno i prezzi degli stock nella successiva giornata di trading.

Gli stock per cui è stata prevista la variazione di prezzo più alta, sono quelli che verranno inclusi nelle *raccomandazioni* fornite dal sistema, consentendo agli investitori di tenere sotto controllo un insieme ridotto di azioni e cogliere prontamente importanti opportunità di investimento a breve termine.

Nella sezione seguente si approfondisce il funzionamento dei componenti del sistema BEst STock Finder a cui, da questo momento, si farà riferimento con l'abbreviazione BeSt.

[5]Tecniche indipendenti la cui esecuzione non influenza il risultato dell'altra.

4.2.1 Componenti e funzionamento

BeSt è un sistema composto da più elementi che gestiscono in modo automatico l'intero processo di data mining, a partire dalla raccolta dei dati finanziari alla selezione degli stock su cui concentrare l'attenzione per il trading nel giorno successivo. Ogni componente assolve i compiti di una fase specifica del processo, come indicato di seguito:

1. Stock data retrival and preparation
   È la fase iniziale del processo, dove avviene la raccolta dei dati storici degli stock, sfruttando le API messe a disposizione da YAHOO! FINANCE, e dove si applicano le trasformazioni necessarie per preparare il dataset all'applicazione degli algoritmi di data mining nella fase successiva.

2. Model generation
   La generazione del modello avviene applicando le tecniche di data mining sul dataset dei prezzi giornalieri pre-processato. In questo stadio, si estraggono i modelli di basati sul sequence mining oppure sulla regressione.

3. Stock recommendation
   L'ultimo componente è quello che sfrutta i pattern estratti nella fase precedente per determinare di quanto varieranno i prezzi di ogni stock nella successiva giornata di trading e, sulla base dei risultati ottenuti, valuta quali sono gli stock più interessanti da raccomandare agli investitori.

Come rappresentato nella figura 4.1, il processo eseguito da BeSt segue i passaggi del KDD process, dove i modelli estratti tramite il data mining permettono di acquisire la conoscenza necessaria per interpretare il problema da risolvere e scegliere la soluzione più opportuna.

Lo stesso processo è utilizzato anche per la sperimentazione svolta in questa tesi, dove però, come si dettaglierà nel capitolo 5, le varie fasi saranno modificate ed adattate ad un problema di classificazione.

Per chiarire i punti su cui sono state introdotte le variazioni a supporto della classificazione, si descrivono ora gli aspetti che caratterizzano la metodologia [40] basata su sequence mining e regressione.

Preparazione dei dati

Il primo aspetto da considerare è il modo in cui si modellano i dati nella fase di preprocessing per costruire i dataset che verranno analizzati rispettivamente dagli algoritmi di sequence mining e di regressione.

Ogni rappresentazione dei dati organizza gli stock all'interno di un dataset pesato (*weighted dataset*), in cui ogni oggetto è definito dall'accoppiata $\langle s_j, r_k^j \rangle$, dove s_j si riferisce allo stock j-esimo dell'insieme di stock S ($s_j \in S$) e r_k^j esprime il

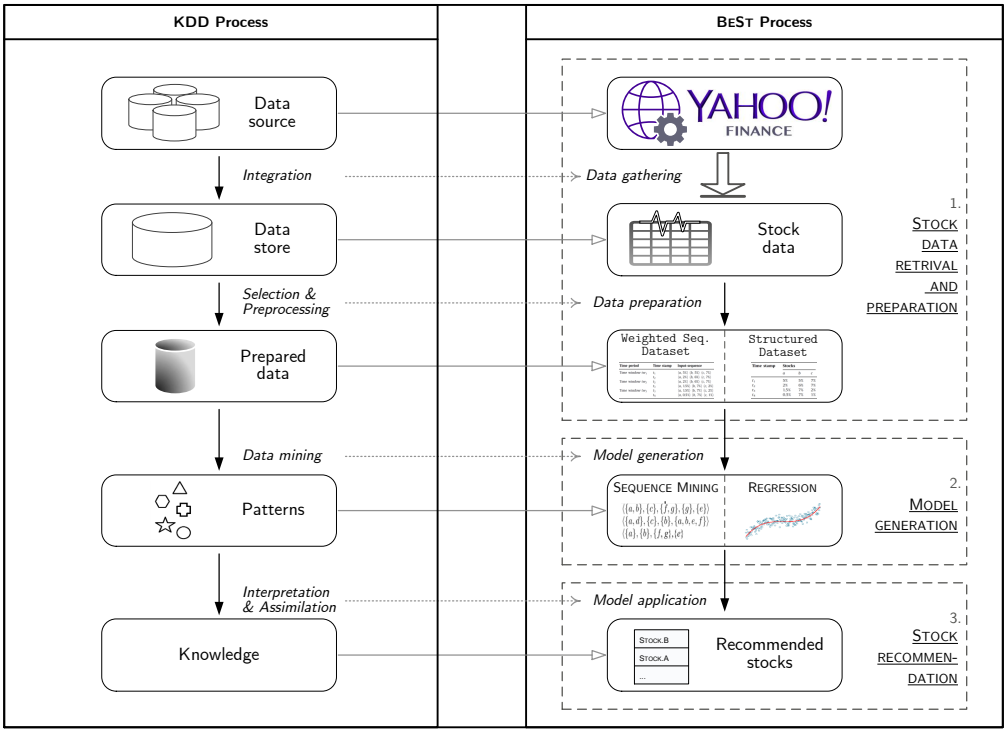

Figura 4.1: BeSt Process in riferimento a KDD Process

peso dello stock s_j in funzione del ritorno relativo giornaliero, calcolato come variazione percentuale tra il prezzo di chiusura del giorno attuale e quello del giorno precedente:

$$r_k^j = \frac{closePrice_{t_k} - closePrice_{t_{k-1}}}{closePrice_{t_{k-1}}}$$

dove:

$$\begin{aligned} closePrice_{t_k} &= \text{prezzo di chiusura dello stock } s_j \text{ nel giorno } t_k \\ closePrice_{t_{k-1}} &= \text{prezzo di chiusura dello stock } s_j \text{ nel giorno precedente } t_{k-1} \\ t_k &= k\text{-esimo giorno di trading} \in \text{intervallo di tempo } T. \end{aligned}$$

Nel caso del sequence mining, la rappresentazione del dataset viene modellata, come mostrato nella tabella 4.2, in modo da considerare ogni esempio come una sequenza ordinata di stock (successione coppie $\{\langle s_a, r_k^a\rangle, ..., \langle s_j, r_k^j\rangle, ...\}$) che occorrono in giorni *consecutivi* all'interno della stessa finestra temporale osservata. Così facendo, gli algoritmi di sequence mining potranno analizzare diversi intervalli di tempo per identificare le sequenze di stock più ricorrenti.

Per la regressione, invece, il dataset viene rappresentato come una struttura di record identificati dalla data del giorno di trading t_k e dal ritorno relativo di ogni stock nel giorno t_k, come esemplificato nella tabella 4.3.

Time period	Time stamp	Input-sequence
Time window tw_1	t_1	$\langle a, 5\% \rangle \langle b, 5\% \rangle \langle c, 7\% \rangle$
	t_2	$\langle a, 2\% \rangle \langle b, 6\% \rangle \langle c, 7\% \rangle$
Time window tw_2	t_2	$\langle a, 2\% \rangle \langle b, 6\% \rangle \langle c, 7\% \rangle$
	t_3	$\langle a, 1.5\% \rangle \langle b, 7\% \rangle \langle c, 2\% \rangle$
Time window tw_3	t_3	$\langle a, 1.5\% \rangle \langle b, 7\% \rangle \langle c, 2\% \rangle$
	t_4	$\langle a, 0.5\% \rangle \langle b, 7\% \rangle \langle c, 1\% \rangle$

Tabella 4.2: Esempio *weighted sequencial* stock dataset

Time stamp	Stocks		
	a	b	c
t_1	5%	5%	7%
t_2	2%	6%	7%
t_3	1.5%	7%	2%
t_4	0.5%	7%	1%

Tabella 4.3: Esempio dataset strutturato per tecniche di regressione

Generazione del modello

Il modello generato dalle tecniche di sequence mining è usato per prevedere le sequenze più ricorrenti di stock con il maggiore ritorno relativo giornaliero. Dal momento che gli algoritmi di sequence mining tipicamente ricercano solamente le sequenze con la maggiore occorrenza, per sfruttare il peso dei rendimenti derivabili dagli stock è stata proposta una versione modificata dell'algoritmo SPADE, dove si introduce la misura dell'*average sequence profit*, con cui si definisce il peso da associare ad ogni sequenza, in funzione del peso degli stock contenuti in essa.

Le tecniche di regressione, invece, prevedono il valore numerico (*continuo*) che ogni stock assumerà nella successiva giornata di trading analizzando il dataset strutturato dei ritorni giornalieri, opportunamente trasformato applicando una funzione di *windowing*[6], per mezzo della quale si mettono a confronto i ritorni relativi di ogni giorno con quelli dei w giorni precedenti, come esemplificato dalla rappresentazione nella tabella 4.4.

[6]Le tecniche di *windowing* sono usate per eseguire trasformazioni dei dati in funzione del tempo. In questo caso la funzione viene sfruttata per rappresentare, come attributi di ogni esempio, i prezzi associati dei giorni precedenti alla data che identifica ogni record del dataset.

Time stamp	**Stocks**								
	$a_{t_{k-2}}$	$a_{t_{k-1}}$	a_{t_k}	$b_{t_{k-2}}$	$b_{t_{k-1}}$	b_{t_k}	$c_{t_{k-2}}$	$c_{t_{k-1}}$	c_{t_k}
t_3	5%	2%	1.5%	5%	6%	7%	7%	7%	2%
t_4	2%	1.5%	0.5%	6%	7%	7%	7%	2%	1%

Tabella 4.4: Esempio *windowing* sul dataset strutturato con $w = 2$

Metodo di raccomandazione degli stock

Infine, il metodo di raccomandazione si occupa di selezionare le predizioni per le quali si stima il maggior profitto. Nel caso del sequence mining questo si traduce nella scelta delle sequenze con il maggior profitto medio (*average sequence profit*): ovvero delle sequenze in cui occorrono gli stock che hanno avuto il massimo rendimento nell'ultimo giorno analizzato. Mentre, per le predizioni ottenute dai metodi basati sulla regressione, si sceglie semplicemente lo stock per cui è stato previsto il massimo ritorno relativo r_k^j.

4.2.2 Metodo di valutazione dei risultati

Per valutare il prestazioni dei modelli di sequence mining e di regressione impiegati da BeSt, sono stati svolti diversi esperimenti in cui si è supposto di investire effettivamente sugli stock raccomandati dal sistema giorno dopo giorno, per poi analizzare i potenziali rendimenti ottenibili nell'arco di un anno di trading. Nello specifico, ogni tecnica è stata confrontata con le altre in termini di:

- Guadagno medio giornaliero
- Deviazione standard del guadagno medio giornaliero
- Numero totale di raccomandazioni
- Guadagno giornaliero massimo

Gli esperimenti hanno coinvolto le azioni appartenenti a tre indici di mercato distinti, quali NASDAQ-100, Dow Jones e FTSE.MIB. Per ogni gruppo di stock sono stati considerati i prezzi di chiusura giornalieri nell'arco degli anni 2011 e 2013, che rappresentano rispettivamente una condizione di mercato sfavorevole, quindi caratterizzata dal calo dei prezzi, ed una situazione di mercato favorevole, contraddistinta da tendenze rialziste.

In ogni esperimento si è supposto di investire sugli stock raccomandati giornalmente dal sistema BeSt, aprendo le posizioni poco prima della chiusura del mercato di ogni giorno d_n, per poi chiuderle poco prima della chiusura del giorno successivo d_{n+1}. In questo contesto, considerando una costante del 15% per approssimare il costo di ogni operazione, il profitto giornaliero r_{d_n} di ogni raccomandazione è calcolato come:

$$r_{d_n} = \frac{closePrice_{d_{n+1}} - closePrice_{d_n}}{closePrice_{d_n}} - 0.15$$

Come verrà dettagliato nel capitolo 6, per valutare la metodologia di classificazione sperimentata in questa tesi si sono utilizzate le stesse metriche basate sulle statistiche calcolate sui profitti giornalieri, ottenendo risultati simili a quelli misurati per le tecniche di sequence mining e di regressione.

La metodologia in oggetto è esposta in modo dettagliato nel capitolo 5, dove si descrivono anche le modifiche apportate al sistema BEST per supportare la classificazione e le nuove strategie decisionali.

Capitolo 5

Metodo proposto

Il metodo proposto in questa tesi si basa su un'estensione del framework BeSt, finalizzata ad esplorare l'applicabilità di diverse tecniche di classificazione nei sistemi di trading quantitativo e la loro configurazione automatica (*self-tuning*).

Lo sviluppo dell'estensione non ha solo reso necessario adattare i componenti esistenti ad un problema di classificazione, ma ha anche richiesto varie modifiche alla struttura stessa del sistema BeSt. Prima di tutto si è parametrizzato l'intero di processo di estrazione delle raccomandazioni affinché si potessero simulare e valutare più metodi per costruire i modelli di classificazione. Poi, per analizzare automaticamente i risultati ottenuti da queste simulazioni, è stato introdotto il processo di *self-tuning* che, tramite un algoritmo di regressione, ha permesso di stimare le prestazioni future delle diverse configurazioni dei classificatori e di scegliere quale di queste fosse quella "*migliore*" per fornire le raccomandazioni giornaliere.

Tutte le decisioni che hanno portato alla realizzazione del nuovo sistema sono state ispirate da alcune constatazioni emerse analizzando le metodologie di stock data mining viste nel capitolo 4: in particolare dal fatto che esistano più soluzioni valide per elaborare i dati mercati azionari e dal fatto che si possa costruire un modello più efficace combinando insieme più tecniche predittive. Per questo motivo, si è cercato di integrare le tecniche di classificazione nel sistema BeSt considerando il maggior numero di possibili alternative.

Le alternative in questione riguardano: la tipologia di algoritmo di classificazione, il metodo di discretizzazione da applicare per creare la classe che categorizzi gli esempi osservati e l'intervallo di tempo da considerare per il training del modello. Non conoscendo a priori quale tecnica performi meglio nella raccomandazione degli stock e quali siano i parametri più efficienti, si simula l'applicazione di tutte le possibili configurazioni di classificatori sui dati storici, imitando il comportamento di un trader quantitativo nell'attività di backtesting di diverse strategie. L'obiettivo di questo approccio è quello di analizzare le prestazioni delle diverse simulazioni per riuscire a:

- ➢ individuare quali classificatori usare per estrarre l'insieme di stock da raccomandare agli investitori;
- ➢ scegliere i parametri di configurazione dei modelli di classificazione usati per il trading.

L'implementazione della funzione di *self-tuning* permette di automatizzare l'analisi delle prestazioni e di operare autonomamente una scelta tra N possibili combinazioni $\{algoritmo, \{parametri\}\}$. Dal momento che questa scelta si basa su un metodo di regressione, il sistema finale si può considerare a tutti gli effetti un modello predittivo ibrido.

Nel resto del capitolo si descrivono dettagliatamente le caratteristiche del nuovo sistema e si riportano le scelte di progettazione affrontate per integrare la metodologia proposta nel framework BeSt. La sezione 5.1 descrive ad alto livello il funzionamento del modello predittivo ibrido. La sezione 5.2 approfondisce le modifiche apportate ai componenti originali di BeSt. Mentre le sezioni 5.3 e 5.4 sono dedicate alle nuove funzionalità introdotte nell'estensione del framework.

5.1 Modello predittivo ibrido

Il sistema ibrido che estende il framework BeSt di base è caratterizzato da due nuovi *processi* eseguiti in parallelo: il primo, con cui si estraggono le previsioni sulla variazione di prezzo degli stock attraverso più metodi di classificazione, e il secondo, con cui si valuta e si sceglie quale metodo utilizzare per raccomandare effettivamente gli stock su cui investire.

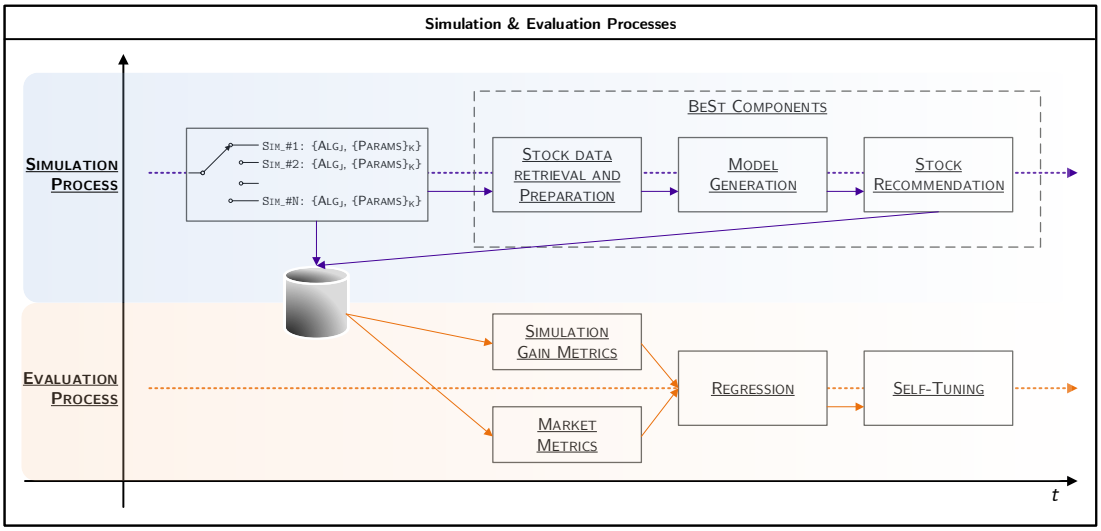

Figura 5.1: Processi di Simulazione e Valutazione

Come rappresentato dalla figura 5.1, il primo processo, essenzialmente, riproduce le stesse fasi implementate dai componenti di BeSt descritti nella sezione 4.2.1, utilizzando, però, più algoritmi di classificazione per generare i modelli e più parametri per preparare il training set e per selezionare le raccomandazioni. Questo processo è definito simulation process, in quanto *simula* la raccomandazione degli stock provando tutte le possibili combinazioni di tecniche e parametri.

Il secondo processo, chiamato evaluation process, è quello che implementa la funzione di self-tuning *valutando* i risultati ottenuti dalle simulazioni eseguite nel primo processo. Nello specifico, la valutazione dei classificatori e delle relative configurazioni viene svolta in funzione di due insiemi di misure statistiche: uno dei quali esprime le performance di ogni simulazione e l'altro che rappresenta le condizioni di mercato relative ai periodi in cui tali simulazioni sono state eseguite.

Le metriche utilizzate per la valutazione delle prestazioni, analogamente a quanto descritto nella sezione 4.2.2, quantificano il guadagno che potenzialmente si sarebbe ottenuto supponendo di investire sugli stock raccomandati dal sistema per un determinato intervallo di tempo. Le misure relative alle condizioni di mercato, invece, sono derivate dalla serie dei prezzi storici dell'indice di mercato a cui appartengono gli stock raccomandati e dai prezzi delle azioni stesse, sempre considerando lo stesso intervallo di tempo.

Le operazioni eseguite dai due processi definiti nel metodo proposto sono riassunte nello schema in figura 5.2, che mostra l'architettura del sistema BeSt esteso e rappresenta le principali fasi del modello predittivo ibrido:

1. Simulation (`For Each {Algorithm, {Parameters}}`)
   1.1. Data Gathering & Data Preparation
   1.2. Model Generation
   1.3. Stock Recommendation
2. Evaluation
   2.1. Statistics Processing
   2.2. Self-Tuning

Nello schema i processi di simulazione e valutazione sono identificati rispettivamente dai blocchi `"Simulation"` e `"Evaluation"`, che al loro interno contengono i nuovi componenti aggiunti al sistema BeSt. Tali componenti sono descritti brevemente di seguito.

Il `"Simulator"` è il componente che si occupa di gestire il processo di simulazione, pilotando l'esecuzione delle tre fasi del processo BeSt originale e registrando i risultati di tutte le combinazioni $\{algoritmo, \{parametri\}\}$ all'interno del database transazionale `"Simulation DB"`. Nello stesso blocco viene rappresentato anche il componente `"RapidMiner Executor"` al solo scopo di distinguere le operazioni eseguite tramite processi `RapidMiner` da quelle implementate da classi `Java`.

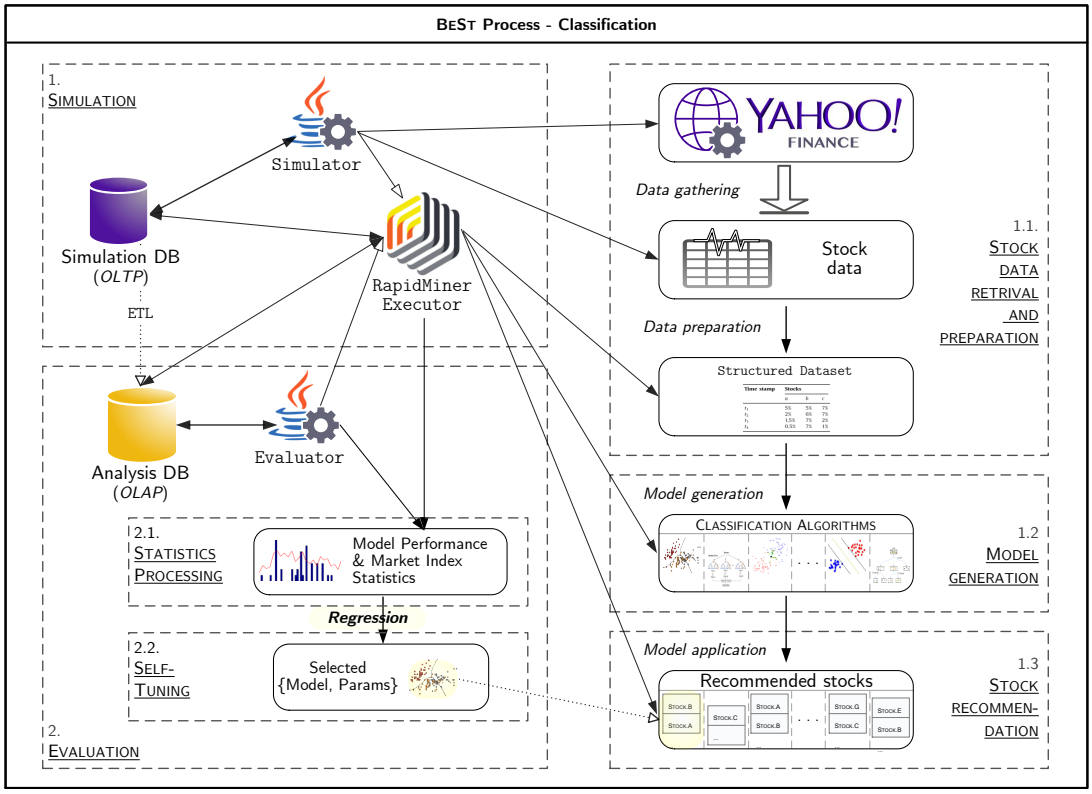

Figura 5.2: Architettura sistema BeSt esteso

Il componente `"Evaluator"`, nel secondo blocco, elabora le statistiche necessarie a costruire il dataset di input del modello di regressione e le memorizza in un database dedicato, l'`"Analysis DB"`, opportunamente popolato ed aggiornato tramite un processo ETL[1], che elabora i dati letti dal `"Simulation DB"`. Utilizzando tali statistiche, l'`"Evaluator"` addestra il modello di regressione a prevedere il guadagno che otterranno le diverse configurazioni dei classificatori e seleziona il modello per cui si prospetta il rendimento più alto. Questa scelta determinerà, infine, quali saranno le previsioni da considerare effettivamente per raccomandare le azioni nelle successive giornate di intraday trading.

Procedendo secondo la stessa sequenza temporale in cui è stato realizzato il progetto, nella sezione successiva si descrivono le modifiche apportate ai componenti del sistema BeSt originale per impiegare le tecniche di classificazione nella raccomandazione degli stock.

[1]ETL: *Extraction, Transformation and Loading*. Questo processo, che carica i dati delle simulazioni nel database dedicato all'analisi, è realizzato tramite un processo RapidMiner pilotato dal componente `"Evaluator"`.

5.2 Modifiche ai componenti del sistema BeSt

I tre componenti originali del sistema BeSt (4.2.1), precedentemente indicati ai punti `1.x`, continuano ad seguire lo stesso flusso di operazioni finalizzate all'estrazione degli stock da raccomandare per la successiva giornata di intraday trading. Le modifiche apportate riguardano solo i processi interni ad ogni blocco e sono indirizzate verso i seguenti obiettivi:

- preparare un training set adatto all'addestramento di modelli di classificazione;
- selezionare gli stock da raccomandare sulla base dei pattern di classificazione ottenuti;
- parametrizzare le variabili di configurazione dei classificatori per supportare il processo esterno di simulazione (controllato dal nuovo componente `Simulator`).

Di seguito si descrivono le variazioni apportate ad ogni componente, specificando quali variabili sono state parametrizzate.

5.2.1 Preparazione dei dati

La preparazione dei dati si basa sullo stesso dataset strutturato usato da BeSt per applicare le tecniche di regressione (di cui si è già esposto un esempio nella tabella 4.3). Adattare tale dataset all'applicazione degli algoritmi di classificazione consiste in due attività: la *pulizia* dei valori mancanti o anomali dal set di prezzi storici scaricati dal web, definita Data cleansing, e la *generazione dell'attributo di classe*, necessario per costruire il modello predittivo, denominata Label generation.

Data cleansing

La maggior parte degli algoritmi di classificazione non è in grado di elaborare dataset con valori mancanti: a partire da alcune implementazioni di alberi decisionali fino a metodi basati su reti neurali e support vector machines. Di conseguenza è necessario implementare un'opportuna gestione dei *missing values* per trasformare l'insieme dei prezzi storici, scaricati da Yahoo! Finance, in un dataset consistente.

Seguendo lo stesso principio di consistenza, si devono gestire anche particolari casi relativi alla presenza di dati sporchi (*noise data*) all'interno della serie temporale dei prezzi di chiusura di ogni stock. Questi casi si sono potuti osservare sia attraverso una prima analisi della distribuzione dei prezzi storici, sia a seguito della verifica delle previsioni ottenute dai classificatori: dove si sono riscontrati profitti e perdite particolarmente elevate in corrispondenza di variazioni anomale dei prezzi da un giorno all'altro.

Per la gestione dei valori mancanti e anomali è stata scelta la seguente strategia:

(i) Classificare come *outliers*[2] solo i prezzi di chiusura che, rispetto all'oscillazione media nel corso di un anno, tra due giorni consecutivi si scostano dal valore medio di un fattore superiore al 90%.

(ii) Data la serie dei prezzi di chiusura di uno generico j-esimo stock nell'intervallo delle date di un anno di trading:
   - *escludere* l'intera serie di prezzi dal dataset, se il rapporto tra i valori mancanti o anomali e il numero totale di esempi è superiore all'1%;
   - *rimpiazzare* i singoli valori mancanti o anomali con il valore precedente della serie (o successivo, nel caso in cui il presente non sia disponibile), se questi sono inferiori all'1% del totale.

Label generation

La generazione della *label* consiste nell'aggiungere un nuovo attributo *discreto* al training set sulla base delle proprietà che si vogliono classificare. Definire le regole per generare l'attributo di classe comporta, come viene indicato di seguito, diverse decisioni.

Decisione 1. Scelta dell'attributo di classe

L'attributo di classe deve rappresentare ciò che i modelli di classificazione dovranno prevedere. In questo caso l'obiettivo è quello di individuare gli stock che tenderanno a variare significativamente nella successiva giornata di trading, perciò la scelta ricade sull'attributo che contiene l'informazione relativa a tale variazione: ovvero il ritorno relativo giornaliero r_k^j, utilizzato nel dataset pesato definito dal sistema BeSt[3].

Essendo r_k^j un valore numerico, non può essere usato direttamente come *label*, ma è necessario *generare* l'attributo di classe attraverso un'operazione di *discretizzazione*:

$$classAttribute = Discretization(r_k^j)$$

$$r_k^j = \frac{closePrice_{t_k} - closePrice_{t_{k-1}}}{closePrice_{t_{k-1}}}$$

La funzione di discretizzazione $Discretization(x)$ può essere applicata in diverse forme e, per questo motivo, costituisce un'ulteriore punto da definire.

[2] In questo casi gli *outliers* identificano le variazioni anomale del prezzo di chiusura di uno stock da un giorno all'altro.

[3] Rif. "Preparazione dei dati" in BeSt, pp.47.

Decisione 2. Discretizzazione attributo di classe

Scegliere il tipo di discretizzazione per l'attributo di classe, ovvero della *label*, implica definire in *quante* e in *quali* classi categorizzare le variazioni relative dei prezzi. Nell'analisi iniziale del problema si sono previste le seguenti opzioni:

- ➢ 2 classi
  - `UP`, `DOWN`
  - classificazione *binomiale* per discriminare le variazioni di prezzo positive da quelle negative;
- ➢ 3 classi
  - `UP`, `ZERO`, `DOWN`
  - classificazione *polinomiale* in cui si prevede anche una variazione nulla, ovvero quando il prezzo di chiusura resta lo stesso del giorno precedente;
- ➢ 4 classi
  - `UP`++, `UP`, `DOWN`, `DOWN`−−
  - classificazione *polinomiale* in cui si distingue, oltre alla direzione della variazione, anche la *quantità* di variazione (`UP`++: il prezzo cresce *molto*; `DOWN`−−: il prezzo decresce *molto*);
- ➢ 5 classi
  - `UP`++, `UP`, `ZERO`, `DOWN`, `DOWN`−−
  - analogo alla classificazione *polinomiale* con 3 classi, ma considerando diverse classi di variazione;
- ➢ N classi
  - si considerano $N (N > 5)$ suddivisioni per descrivere la *quantità* di variazione subita dei prezzi degli stock.

NB. Non tutti i classificatori sono in grado di gestire più di 2 classi, come ad esempio gli algoritmi basati su SVM. Per questi metodi, come sarà indicato nella descrizione della logica di simulazione, si valuteranno solamente le prestazioni della classificazione binomiale.

Il numero di classi di tutte le opzioni presentate determina il numero di *cut point* da definire ed utilizzare per suddividere l'intervallo di valori continui assunti da r_k^j. Per esempio:

- 2/3 classi $\rightarrow$ 1 $cutPoint = \{0\}$

$$\begin{aligned}
\texttt{UP}: &\quad r_k^j \in \{x \geq 0\} \quad \text{oppure } r_k^j \in \{x > 0\} && (x \in \mathbb{R}) \\
\texttt{DOWN}: &\quad r_k^j \in \{x \leq 0\} \quad \text{oppure } r_k^j \in \{x < 0\} && (x \in \mathbb{R}) \\
\texttt{ZERO}: &\quad r_k^j \in \{x = 0\} && (x \in \mathbb{R})
\end{aligned}$$

- 4/5 classi $\rightarrow$ 2 $cutPoint = \{0, \alpha\}$

$$\begin{aligned}
\texttt{UP++}: &\quad r_k^j \in \{x \geq \alpha\} && (\{x, \alpha\} \in \mathbb{R}) \\
\texttt{DOWN}--: &\quad r_k^j \in \{x \leq -\alpha\} && (\{x, \alpha\} \in \mathbb{R}) \\
\texttt{UP}: &\quad r_k^j \in \{0 > x \geq \alpha\} && (\{x, \alpha\} \in \mathbb{R}) \\
\texttt{DOWN}: &\quad r_k^j \in \{-\alpha \leq x \leq 0\} && (\{x, \alpha\} \in \mathbb{R}) \\
\texttt{ZERO}: &\quad r_k^j \in \{x = 0\} && (x \in \mathbb{R})
\end{aligned}$$

Identificare la classe `ZERO` ha senso nel caso in cui il modello predittivo serva a generare segnali di trading del tipo $\{Buy, Sell, Hold\}$, ovvero quando è prevista anche la possibilità di tenere aperte posizioni per stock il cui prezzo non subirà modifiche significative. In questo caso però, dove conta identificare le variazioni di prezzo più alte, si sceglie di *non considerare* la classe `ZERO` e, quindi, di discretizzare i ritorni per ottenere le altre etichette specificate.

Per contenere il numero di esperimenti, si è deciso anche di scartare modelli di classificazione polinomiale con più di 4 classi (caso N classi). Di conseguenza, la scelta finale relativa al tipo di discretizzazione da adottare per l'attributo di classe ricade nelle opzioni:

- classificazione binomiale { `UP`, `DOWN` }
- classificazione polinomiale a 4 classi { `UP++`, `UP`, `DOWN`, `DOWN--` }

Anche se, come mostrato negli esempi precedenti, si possa scegliere semplicemente il valore 0 per definire le classi del problema, nel caso della classificazione polinomiale è necessario identificare almeno un $cutPoint \neq 0$ per definire la soglia al di sopra o al di sotto della quale si considera più alta la variazione relativa del prezzo degli stock. Per questo punto è necessario prendere un'altra decisione.

Decisione 3. Scelta dei cut point

Per la scelta dei cut point in cui discretizzare l'intervallo dei valori r_k^j si sono valutate due possibilità:

(a) usare valori fissi
(b) determinare algoritmicamente i valori

Inizialmente è stata provata la seconda opzione, sfruttando un algoritmo di discretizzazione[4] in cui si implementa la funzione:

$$\{sub\_interval_1, ..., sub\_interval_N\} = f(\{interval\}, N)$$

dove:

$$\{interval\} = \text{insieme dei valori di } r_k^j$$
$$N = \text{numero di sotto-intervali da identificare}$$

[4]Operatore RapidMiner `Discretize by Binning`[41].

Il fatto che l'algoritmo riuscisse a delimitare i sotto-intervalli nel modo più uniforme possibile si è rivelato però un effetto indesiderato, poiché lo stesso numero di sotto-intervalli, ovvero di variazioni associate alla stessa classe, veniva identificato sia per stock caratterizzati da un'alta oscillazione del prezzo che per quelli dove il prezzo variava solo lievemente:

Time stamp	**Stocks**					
	a	$label_a$	b	$label_b$	c	$label_c$
t_1	5%	`UP++`	5%	`UP`	7%	`UP++`
t_2	2%	`UP++`	6%	`UP`	7%	`UP++`
t_3	1.5%	`UP`	7%	`UP++`	2%	`UP`
t_4	0.5%	`UP`	7%	`UP++`	1%	`UP`

Tabella 5.1: Esempio discretizzazione automatica dell'attributo di classe

Come si vede nell'esempio in tabella 5.1, discretizzare l'attributo di classe usando intervalli uniformi causa l'assegnazione dell'etichetta `UP++` alla variazione del 2% dello stock a in t_2, che, a confronto degli altri valori nel dataset, non dovrebbe essere considerato un ritorno significativo.

Per discretizzare correttamente l'attributo di classe, si è quindi scelto di definire delle soglie fisse.

Nella tabella 5.2 si può notare come, rispetto all'esempio precedente, impostare il cut point al 3% determina l'assegnazione dell'etichetta `UP` al ritorno di a in t_2.

Time stamp	**Stocks**					
	a	$label_a$	b	$label_b$	c	$label_c$
t_1	5%	`UP++`	5%	`UP`	7%	`UP++`
t_2	2%	`UP`	6%	`UP`	7%	`UP++`
t_3	1.5%	`UP`	7%	`UP++`	2%	`UP`
t_4	0.5%	`UP`	7%	`UP++`	1%	`UP`

Tabella 5.2: Esempio discretizzazione usando $cutPoint = 3\%$

Dovendo scegliere un valore fisso per i cut point, è stata adottata la strategia riportata nella tabella 5.3:

Label Disc. type	**Fixed Cut Point**	**Rule**	
		LONG *trade*	SHORT *trade*
Binominal	0%	$\texttt{UP} = \{r_k^j \geq 0\%\}$	$\texttt{UP} = \{r_k^j > 0\%\}$
		$\texttt{DOWN} = \{r_k^j < 0\%\}$	$\texttt{DOWN} = \{r_k^j \leq 0\%\}$
	1%	$\texttt{UP} = \{r_k^j \geq 1\%\}$	$\texttt{UP} = \{r_k^j > -1\%\}$
		$\texttt{DOWN} = \{r_k^j < 1\%\}$	$\texttt{DOWN} = \{r_k^j \leq -1\%\}$
	2%	$\texttt{UP} = \{r_k^j \geq 2\%\}$	$\texttt{UP} = \{r_k^j > -2\%\}$
		$\texttt{DOWN} = \{r_k^j < 2\%\}$	$\texttt{DOWN} = \{r_k^j \leq -2\%\}$
	3%	$\texttt{UP} = \{r_k^j \geq 3\%\}$	$\texttt{UP} = \{r_k^j > -3\%\}$
		$\texttt{DOWN} = \{r_k^j < 3\%\}$	$\texttt{DOWN} = \{r_k^j \leq -3\%\}$
	4%	$\texttt{UP} = \{r_k^j \geq 4\%\}$	$\texttt{UP} = \{r_k^j > -4\%\}$
		$\texttt{DOWN} = \{r_k^j < 4\%\}$	$\texttt{DOWN} = \{r_k^j \leq -4\%\}$
Polynominal 4 Bins	1%	$\texttt{UP++} = \{r_k^j \geq 1\%\}$ $\texttt{DOWN}-- = \{r_k^j \leq -1\%\}$ $\texttt{UP} = \{0\% < r_k^j < 1\%\}$ $\texttt{DOWN} = \{-1\% < r_k^j \leq 0\%\}$	
	2%	$\texttt{UP++} = \{r_k^j \geq 2\%\}$ $\texttt{DOWN}-- = \{r_k^j \leq -2\%\}$ $\texttt{UP} = \{0\% < r_k^j < 2\%\}$ $\texttt{DOWN} = \{-2\% < r_k^j \leq 0\%\}$	
	3%	$\texttt{UP++} = \{r_k^j \geq 3\%\}$ $\texttt{DOWN}-- = \{r_k^j \leq -3\%\}$ $\texttt{UP} = \{0\% < r_k^j < 3\%\}$ $\texttt{DOWN} = \{-3\% < r_k^j \leq 0\%\}$	
	4%	$\texttt{UP++} = \{r_k^j \geq 4\%\}$ $\texttt{DOWN}-- = \{r_k^j \leq -4\%\}$ $\texttt{UP} = \{0\% < r_k^j < 4\%\}$ $\texttt{DOWN} = \{-4\% < r_k^j \leq 0\%\}$	

Tabella 5.3: Strategia discretizzazione label con cut point fisso

Secondo il metodo di discretizzazione adottato, i cut point $\neq 0$ sono stati usati anche per la classificazione binomiale definendo strategie diverse in funzione del tipo di investimento da raccomandare: quando si predicono le variazioni per selezionare gli stock da comprare, si considerano solo gli stock il cui prezzo è salito *oltre* il cut point, mentre nel caso di stock da vedere, si considerano le variazioni *al di sotto*

del valore negativo del cut point.

È importante sottolineare che i valori scelti sono stati determinati a seguito dell'analisi dei dati iniziali e dei risultati ottenuti dal processo di data mining. In particolare, come verrà dimostrato nel capitolo 6, i cut point $\neq 0$ hanno permesso di ridurre il numero di raccomandazioni ottenute dalla classificazione binomiale che, come già indicato più volte, devono essere limitate per ridurre il rischio di esposizione al mercato.

Dopo aver definito i cut point per discretizzare i ritorni giornalieri degli stock r_k, è possibile generare l'attributo di classe e preparare un dataset idoneo all'applicazione degli algoritmi di classificazione.

Generazione della label

Come indicato nella decisione 1, per le tecniche di classificazione deve essere generato un attributo di classe discreto in funzione del ritorno r_k^j, poiché il ritorno relativo giornaliero costituisce ciò che i modelli costruiti dovranno prevedere.

Nello specifico, il valore r_k^j da discretizzare per generare la label è quello dello stock j-esimo in rapporto con le variazioni subite dal prezzo di j e di quello di tutti gli altri stock analizzati: ovvero il valore definito nel dataset strutturato su cui è stata applicata la funzione di *windowing* per *trasporre* su ogni riga t_k i ritorni dei w giorni precedenti:

Time stamp	**Stocks**									**Label$_j$**
	$a_{t_{k-2}}$	$a_{t_{k-1}}$	a_{t_k}	$b_{t_{k-2}}$	$b_{t_{k-1}}$	b_{t_k}	$j_{t_{j-2}}$	$j_{t_{k-1}}$	~~j_{t_k}~~	
t_3	5%	2%	1.5%	5%	6%	7%	7%	7%	~~2%~~	`UP`
t_4	2%	1.5%	0.5%	6%	7%	7%	7%	2%	~~1%~~	`UP`

Tabella 5.4: Esempio *windowing* sul dataset strutturato con $w = 2$ - classificazione

Come si può osservare nella tabella 5.4, che riprende l'esempio 4.4 del sistema BeSt originale, *solo il rendimento dello stock j-esimo viene discretizzato* e trasformato in attributo di classe, mentre invece i valori relativi alle variazioni dei giorni precedenti e di tutti gli altri stock sono lasciati in forma numerica. Tali attributi sono detti *regolari*[5] e, non ricoprendo il ruolo di label, vengono trattati dagli algoritmi come variabili, in funzione delle quali estrarre il modello per predire l'attributo di classe.

[5]Attributo *regolare* è un altro termine utilizzato dal software RapidMiner, con cui si indicano gli attributi che non hanno nessun ruolo speciale (es. `label`, `ID`, `prediction`, ecc.) e sono considerati come variabili del problema di data mining.

Decisione 4. Discretizzazione degli attributi regolari

La maggior parte degli algoritmi di classificazione elabora set di dati composti da attributi numerici e label discreta, come nel caso del dataset dell'esempio 5.4.

Esistono però alcuni algoritmi, come il `CHAID` o l'`ID3`, che lavorano solamente su dati nominali e, per essere supportati, richiederebbero l'applicazione della discretizzazione anche agli attributi regolari, eventualmente usando le stesse logiche definire per l'attributo di classe.

Considerando che nel software usato per la sperimentazione (RapidMiner), sono disponibili oltre 60 algoritmi che supportano attributi numerici, *si è scelto di non applicare ulteriori trasformazioni* al dataset di input, escludendo dalle simulazioni solo un numero ristretto di classificatori.

A favore di questa decisione occorre osservare inoltre che l'operazione di discretizzazione, per definizione, implica un'approssimazione e, quindi, perdita di precisione e potenzialmente anche una maggiore probabilità di errore.

Riepilogo

Sintetizzando i punti chiave della fase di preparazione dei dati: l'introduzione della classificazione ha richiesto l'applicazione di diverse trasformazioni al dataset strutturato usato nel sistema BeSt originale. Tali modifiche sono il risultato del processo di pulizia e delle decisioni prese in merito alla generazione dell'attributo *label*:

0. Criterio di pulizia dei dati mancanti o anomali

   Decisione: Sostituire i valori mancanti o anomali presenti in una percentuale inferiore all'1% del totale de prezzi storici di un anno di trading. Escludere dall'analisi i gli stock con una percentuale superiore all'1%.

1. Scelta dell'attributo di classe

   Decisione: Usare il valore discretizzato del ritorno relativo giornaliero r_k^j per classificare la variazione futura di ogni stock.

2. Discretizzazione dell'attributo di classe

   Decisione: 2 tipi di discretizzazione

   - 2 classi (`UP`, `DOWN`) → per classificazione binomiale
   - 4 classi (`UP`++, `UP`, `DOWN`, `DOWN`−−) → per classificazione polinomiale.

3. Scelta dei cut point

   Decisione: 9 cut point (0%, ±1%, ±2%, ±3%, ±4%) applicati secondo la strategia indicata in tabella 5.3.

4. Discretizzazione degli attributi regolari

   Decisione: No. Si escludono dal processo di simulazione i classificatori che non supportano attributi numerici.

Parametri per la preparazione dei dati

Contemplando le diverse possibilià per il tipo di discretizzazione da applicare e per i relativi cut point, la fase di preparazione dei dati, definisce 2 parametri per la simulazione dei modelli:

`LabelDiscretization` { `BINOMINAL_FIXED`, `POLY_4BINS_FIXED` }
`FixedCutPoint` { -4%, -3%, -2%, -1%, 0%, 1%, 2%, 3%, 4% }.

Anche se sono state previste diverse metodologie da applicare alla pulizia dei dati, il componente di *data cleansing* è stato utilizzato come unica procedura *costante* e, pertanto, non è stato parametrizzato.

I diversi metodi di discretizzazione hanno invece definito i parametri che saranno sfruttati dal componente di simulazione per costruire i diversi training set con cui addestrare i modelli di classificazione, secondo le combinazioni definite nella tabella 5.3. In questo contesto le variabili legate alla discretizzazione dell'attributo di classe sono strettamente legate alla fase di Model generation, in quanto influenzano il modo in cui gli algoritmi dovranno costruire i pattern di classificazione. Però, essendo tali variabili caratterizzate dall'insieme di decisioni prese nella fase di preparazione dei dati, si è preferito argomentarle in questa sezione, enfatizzando l'aspetto di trasformazione dei dati.

5.2.2 Classificazione e raccomandazione delle azioni

I componenti relativi alle fasi di Model generation e di Stock recommendation sono strettamente connessi tra loro, in quanto non appena viene generato il modello, questo viene immediatamente applicato per prevedere il ritorno di uno stock (r_k^j) nella successiva giornata di trading e, sulla base della classificazione tutti gli stock, si selezionano le azioni da raccomandare.

Per chiarire il flusso di operazioni svolte dai due componenti a seguito delle modifiche apportate, si descrivono tali operazioni rispettivamente nel contesto del Processo di classificazione e in quello del Processo di raccomandazione.

Processo di classificazione

Il processo di classificazione viene eseguito iterativamente per ognuno degli stock osservati in funzione dei dati storici relativi alle ultime T giornate di trading dove, come descritto nella sezione precedente, si applicano le funzioni di windowing e di creazione dell'attributo di classe ($label_j$) per ogni j-esimo stock, in modo da ottenere un training set organizzato secondo una struttura dati come quella esemplificata nella tabella 5.4.

I prezzi degli stock dell'ultima giornata di trading t_k non vengono inclusi nel training set, poiché l'attributo di classe di queste azioni deve essere determinato applicando il modello di classificazione appena generato.

Pseudo-codice 1 Classification process

```
1: FirstTradingDay ← GetCurrentDate() − T;
2: LastTradingDay ← GetCurrentDate();
3:
4: for each (s_j in S) do
5:     TrainingSet ← Filter(StockDataset,
                            FirstTradingDay, (LastTradingDay − 1));
6:     TrainingSet_j ← Windowing(TrainingSet, w)
7:     LabelDiscretization(TrainingSet_j, r_k^j,
                           LabelDisc, FixedCutPoint);
8:
9:     // Train classification model for stock s_j
10:    Model ← GenerateModel(TrainingSet_j);
11:
12:    UnlabeledSet ← Filter(StockDataset,
                             (LastTradingDay − w), LastTradingDay);
13:    UnlabeledSet_j ← Windowing(UnlabeledSet, w);
14:
15:    // Predict stock s_j class on next trading day
16:    ResultSet[j] ← ApplyModel(UnlabeledSet_j, Model);
17: end for each
18:
19: /* Stock recommendation */
20: Portfolio_LONG ← Recommendation(ResultSet, LONG,
                                   LabelDisc, MinConfidence);
21: Portfolio_SHORT ← Recommendation(ResultSet, SHORT,
                                    LabelDisc, MinConfidence);
```

Per chiarire ulteriormente il processo descritto, si sfrutta la rappresentazione dello pseudo-codice 1, dove si mettono in rilevo i seguenti punti:

- Il `TrainingSet` è inizialmente costituito da $(\mathtt{T}-1)$[6] giornate di trading, ovvero $(\mathtt{T}-1)$ esempi, ma l'operazione di windowing riduce la sua cardinalità a $(\mathtt{T}-1-\mathtt{w})$ esempi[7].

[6] Si sottrae un giorno perché nel training set si esclude l'ultima giornata di trading disponibile.

[7] Il windowing di w giorni riduce il numero di timestamp t_k di w giorni, in quanto i valori di quegli istanti di tempo sono trasposti sulle colonne del dataset.

- Anche nel set di dati non etichettati `UnlabeledSet`, gli ultimi $(w + 1)$ giorni di trading sono ridotti ad 1 solo esempio dopo l'operazione di windowing: in questo modo si trasformano i dati non etichettati nello stesso formato con cui è stato generato il modello predittivo.
- Il processo di classificazione viene eseguito *per ogni stock*, generando, per ogni j-esimo stock, l'attributo di classe in funzione del ritorno r_k^j, che viene sempre messo a confronto con le variazioni di prezzo di tutti gli altri stock negli ultimi w giorni.

In coda al processo di classificazione, il `ResultSet` delle previsioni delle variazioni di tutti gli stock viene sottoposto al processo di raccomandazione, che opera secondo determinate politiche, descritte di seguito.

Processo di raccomandazione

Per descrivere il processo di raccomandazione degli stock si utilizza nuovamente la rappresentazione in pseudo-codice, riportando nell'algoritmo 2 lo sviluppo delle funzioni `RECOMMENDATION` indicate alle righe 20 e 21 del processo di classificazione.

Pseudo-codice 2 Stock recommendation process

```
1:  function RECOMMENDATION(ResultSet, Trade,
                            LabelDisc, MinConfidence)
2:      ResultSet ← Filter(ResultSet, MinConfidence);
3:      switch (LabelDisc) do
4:          case LabelDisc.BINOMINAL_FIXED
5:              switch (Trade) do
6:                  case TradeType.LONG
7:                      RecommendationSet ← Filter(ResultSet, UP);
8:                  case TradeType.SHORT
9:                      RecommendationSet ← Filter(ResultSet, DOWN);
10:         case LabelDisc.POLY_4BINS_FIXED
11:             switch (Trade) do
12:                 case TradeTypes.LONG
13:                     RecommendationSet ← Filter(ResultSet, UP++)
14:                 case TradeTypes.SHORT
15:                     RecommendationSet ← Filter(ResultSet, DOWN−−);
16:     return RecommendationSet
17: end function
```

Nella logica di selezione degli stock si evidenzia la dipendenza funzionale dai

parametri `Trade` e `LabelDisc`, che rappresentano rispettivamente il *tipo di investimento* per cui devono essere fatte le raccomandazioni e il *tipo di discretizzazione dell'attributo di classe*. Infatti quando si discretizza la label in 2 classi, si selezionano *tutte* le predizioni `UP` o `DOWN` (a seconda del `Trade`); mentre, nel caso della discretizzazione su 4 classi, si scelgono *solo* le previsioni delle classi di rendimento *più alte* `UP`++ o `DOWN`−−.

A differenza del meccanismo di raccomandazione degli stock del sistema BeSt originale, dove si potevano selezionare i migliori *top-N* stock, ordinati in base al valore numerico del rendimento previsto, le classi nominali non consentono una forma diretta di *ranking* delle predizioni e, quindi, necessitano di applicare ulteriori metodi per affinare e filtrare i risultati prodotti dal modello che, come nel caso della classificazione binomiale, possono raggiungere anche un elevato numero di stock.

Per ottimizzare il set di risultati prodotto dai classificatori, nella logica di raccomandazione degli stock è stata implementata la seguente strategia:

- Filtrare l'insieme di stock selezionando solo le predizioni con una confidenza[8] al di sopra della soglia definita dall'argomento `MinConfidence` (riga 2).
- Definire 4 classi per distinguere la *quantità* di ritorno relativo giornaliero e selezionare solo quelle migliori.
- Usare $cutPoint \neq 0$ per alzare/abbassare la soglia con cui discretizzare l'attributo di classe, così da indirizzare le previsioni del modello verso le classi di rendimento migliori.

Parametri per la classificazione e raccomandazione degli stock

Nell'ambito del simulation process, per i componenti relativi alla generazione del modello e alla raccomandazione degli stock si definiscono ulteriori parametri, che si aggiungo a quelli identificati per la fase di preparazione dei dati:

- `Algorithm`
  Algoritmo usato per estrarre il pattern di classificazione (implementato dalla funzione `GenerateModel` alla linea 10 del processo 1).
- `MinConfidence`
  Confidenza minima delle previsioni.
- `Sliding Window`
  Ampiezza della finestra di windowing in cui considerare variazioni consecutive dei rendimenti giornalieri degli stock.

[8] La confidenza, come indicato nella sezione 2.2, esprime una misura della probabilità del verificarsi della previsione ottenuta dal modello.

- `TrainingDays`
  Numero di giorni di trading T_k da utilizzare per il training del modello di classificazione. Questo numero corrisponde ai $(\mathtt{T}-1)$ esempi inclusi nell'intervallo di date estratto nelle prime due righe dello pseudo-codice 1.

- `TradeType`
  Tipo di investimento in base al quale raccomandare gli stock per cui si prevede il miglior rendimento.

Riproducendo la rappresentazione dei parametri di simulazione specificata nelle conclusioni della sezione precedente, di seguito si indicano i valori delle variabili relative alla classificazione e raccomandazione degli stock (senza indicare esplicitamente tutti gli algoritmi testati, che saranno poi specificati nel capitolo 6):

`Algorithm`	{ `K_NN`, `Naive Bayes`, `Decision Tree`, . . . }
`MinConfidence`	{ 0.5 }
`SlidingWindow`	{ 2, 3, 4, 5 }
`TrainingDays`	{ 10, 15, 20, 25, 30 }
`TradeType`	{ `LONG`, `SHORT` }.

La scelta di variare i parametri `SlidingWindow` e `TrainingDays` implica simulare l'effetto di diversi intervalli di tempo per il training di ogni classificatore. Nello specifico l'ampiezza w della `SlidingWindow` definisce *per quanto tempo* osservare la variazione dei prezzi di chiusura degli stock, mentre invece il numero di `TrainingDays` indica *quanti esempi diversi* considerare per estrarre un pattern di classificazione basato sulle w variazioni consecutive.

Nella sezione 5.3 si approfondisce il funzionamento della simulazione dei diversi parametri per valutare la migliore configurazione dei modelli di classificazione.

5.3 Simulazione e valutazione delle prestazioni

Il metodo di valutazione delle diverse configurazioni dei classificatori è stato ciò che ha maggiormente influenzato questo progetto di tesi, in quanto ha fatto emergere l'esigenza di introdurre il meccanismo di simulazione per testare non solo algoritmi diversi, ma anche molteplici combinazioni dei parametri legati alla preparazione dei dati e all'estrazione delle raccomandazioni.

Tale esigenza costituisce un aspetto molto importante della metodologia adottata, in quanto la valutazione dei modelli non si basa sulla loro accuratezza, ma sul potenziale rendimento che può essere ottenuto applicando le predizioni dei classificatori in uno scenario reale.

Generalmente un modello di classificazione deve essere in grado di assegnare il maggior numero di esempi alla corretta classe di appartenenza e, di conseguenza, tende ad essere considerato migliore il classificatore avente la più alta percentuale

di predizioni corrette. Però, nello scenario applicativo del mercato azionario, l'accuratezza non è più un requisito sufficiente, poiché le predizioni possono avere un peso diverso.

A titolo di esempio si considera un classificatore in grado di prevedere correttamente circa l'80% delle prossime variazioni dei prezzi (es. `UP`, `DOWN`): se l'80% delle previsioni corrette permette di fare trading su stock il cui prezzo è variato solo di pochi punti percentuali, allora, nonostante il pattern sia accurato all'80%, il ritorno sugli investimenti sarebbe minimo e, potenzialmente, potrebbe anche essere nullo o negativo se quel 20% delle restanti classificazioni avesse portato a scelte di trading sbagliate, includendo stock il cui prezzo è invece variato in modo molto più significativo.

Per questo motivo, l'affidabilità di un classificatore e della relativa configurazione di parametri viene validata simulando le scelte di investimento sui dati storici e calcolando il potenziale guadagno derivabile dalle raccomandazioni ottenute in ogni simulazione.

Si procede quindi descrivendo di seguito il simulation process implementato per testare l'effetto delle possibili combinazioni dei parametri identificati nelle sezioni 5.2.1 e 5.2.2.

5.3.1 Processo di simulazione

Una *simulazione* identifica l'insieme delle raccomandazioni ottenute, giorno per giorno nell'arco di *un anno di trading*, applicando un modello di classificazione definito dalla combinazione delle seguenti variabili:

- `StockMarketIndex` → indice azionario relativo al paniere contenente gli stock analizzati;
- `StartDate, EndDate` → date di inizio e fine della simulazione (coincidono con il primo e l'ultimo giorno di trading dell'anno);
- `Algorithm` → algoritmo di classificazione (definito in 5.2.2);
- `MinConfidence` → confidenza minima delle previsioni (definito in 5.2.2);
- `TrainingDays` → numero di giorni usato come training set (definito in 5.2.2);
- `SlidingWindow` → numero di variazioni di prezzo consecutive osservate (definito in 5.2.2);
- `LabelDiscretization` → tipo di discretizzazione usato per generare l'attributo di classe (definito in 5.2.1);
- `FixedCutPoint` → cut point a valore fisso usato per la discretizzazione (definito in 5.2.1);
- `TradeType` → tipo di investimento per cui raccomandare gli stock (definito in 5.2.2).

Il componente `Simulator` prova le diverse combinazioni dei parametri indicati attraverso il processo rappresentato dallo pseudo-codice 3.

Pseudo-codice 3 Simulation Process

```
1:  /* Initialize simulation process */
2:  CleansedStockQuotes ←InitSimulationData(
                StockMarketIndex, TrainingDays, StartDate, EndDate);
3:  TradingDates ←ExtractTradeDates(CleansedStockQuotes);
4:
5:  /* Loop through each parameter value */
6:  for each (Algorithm in ClassificationAlgorithmSet) do
7:    for each (LabelDiscretization in LabelDiscretizatonSet) do
8:      for each (FixedCutPoint in FixedCutPointSet) do
9:        for each (TradeType in TradeTypeSet) do
10:         // Skip unsupported simulation scenarios
11:         if (! IsParameterSetValid(Algorithm, LabelDiscretization
                                      FixedCutPoint, TradeType) ) then
12:           continue;
13:         end if
14:         for each (TrainingDays in TrainingDaysSet) do
15:           for each (SlidingWindow in SlidingWindowSet) do
16:             // Create simulation object
17:             Simulation ← new Simulation(StartDate, EndDate,
                            SlidingWindow, Algorithm, MinConfidence,
                            LabelDiscretization, FixedCutPoint, TradeType);
18:
19:             /* Run simulation through each trading date */
20:             for each (TradeDate in TradeDates) do
21:               FirstTradingDay ← TradeDate;
22:               LastTradingDay ← (TradeDate + TrainingDays);
23:               Predictions ← RmExecutor.Run(BeSt_Process, Simulation
                                        FirstTradingDay, LastTradingDay);
24:               DbManager.Store(Simulation, Predictions);
25:             end for each
26:
27:           end for each      // end for each SlidingWindow
28:         end for each        // end for each TrainingDay
29:       end for each          // end for each TradeType
30:     end for each            // end for each FixedCutPoint
31:   end for each              // end for each LabelDiscretization
32: end for each                // end for each Algorithm
```

Come si può notare dai passaggi svolti nel processo di simulazione, la combinazione delle variabili considerate è ottenuta attraverso una serie di cicli annidati (linee 6, 7, 8, 14, 15), all'interno dei quali si costruisce l'oggetto `Simulation` (linea 17) e lo si sottopone al processo di classificazione (linea 23), memorizzando le predizioni ottenute per ogni giornata di trading (`TradeDate`) nel database dedicato ai dati delle simulazioni (linea 24).

Per evitare che gli algoritmi che supportano solamente la classificazione binomiale vengano utilizzati in simulazioni dove si genera una label su più di due classi, si esegue un controllo sul valore dei parametri, a seguito del quale possono essere saltate le iterazioni contenenti combinazioni non valide. Questa logica viene rappresentata dalla funzione `IsParameterSetValid` (linea 11) che, nello svolgere l'ispezione delle variabili, assicura anche che le combinazioni dei parametri siano conformi con la strategia di discretizzazione con cut point fisso, definita nella sezione 5.2.1. Lo pseudo-codice 4 descrive le verifiche eseguite dalla funzione di controllo, evidenziando le condizioni per cui una combinazione dei parametri può essere non valida (ritornando il valore booleano *false*).

Pseudo-codice 4 Supported simulation parameters validation

```
1:  function IsParameterSetValid(Algorithm, LabelDiscretization,
                                 FixedCutPoint, TradeType)
2:
3:      if (! LabelDiscretization.IsBinominal()) then
4:          // Polynomial classification
5:          if (! Algorithm.IsPolynominalLabelSupported()) then
6:              return false;
7:          end if
8:          if (FixedCutPoint < 0) then
9:              return false;
10:         end if
11:     else
12:         // Binomial classification
13:         if (TradeType == TradeTypes.LONG && FixedCutPoint < 0) then
14:             return false;
15:         end if
16:         if (TradeType == TradeTypes.SHORT && FixedCutPoint > 0) then
17:             return false;
18:         end if
19:     end if
20:     // Parameter set valid
21:     return true;
22: end function
```

Memorizzare nel Simulation DB le previsioni ottenute da ogni modello, permette di calcolare *a posteriori* i profitti e le perdite che si sarebbero ottenuti nell'intervallo di tempo simulato se le raccomandazioni dei modelli fossero state realmente usate per le scelte di trading giorno dopo giorno.

Il calcolo dei profitti e delle perdite a posteriori richiede di archiviare nel database anche i prezzi storici di tutti gli stock scaricati dal web. A tale scopo, l'inizializzazione del processo di simulazione pilota il componente relativo al Data Gathering & Data Preparation e salva i prezzi storici e i dataset pre-processati nel database (linee 2 e 3), come indicato in modo più dettagliato nello pseudo-codice 5.

Pseudo-codice 5 Simulator Data Initialization

```
1:  function InitSimulationData(StockMarketIndex, TrainingDays,
                                StartDate, EndDate)
2:      // Check if cleansed stock quotes are available already
3:      CleansedStockQuotes ← DbManager.GetCleansedStockQuotes(
                StockMarketIndex, TrainingDays, StartDate, EndDate);
4:
5:      if (CleansedStockQuotes == null) then
6:          // Check if stock quotes are downloaded already
7:          StockQuotes ← DbManager.GetStockQuotes(
                StockMarketIndex, TrainingDays, StartDate, EndDate);
8:
9:          if (StockQuotes == null) then
10:             // Download stock quotes into simulation database
11:             StockQuotes ← StockDownloader.DownloadStockQuotes(
                    StockMarketIndex, TrainingDays, StartDate, EndDate);
12:             DbManager.Store(StockQuotes);
13:         end if
14:
15:         // Run data cleansing process and store cleansed stock
16:         // quotes in simulation DB
17:         CleansedStockQuotes ← RmExecutor.Run(
                DataCleansingProcess, StockQuotes);
18:         DbManager.Store(CleansedStockQuotes);
19:     end if
20: end function
```

Oltre a descrivere il modo in cui i dati per le simulazioni sono memorizzati nel database, la rappresentazione della funzione InitSimulationData serve anche a mettere in evidenza il fatto che il *download* dei prezzi storici e il *pre-processing* dei dati (*data cleansing*) siano eseguiti solamente la prima volta, mentre, per tutte le

successive inizializzazioni del processo di simulazione, sarà sufficiente recuperare i dati necessari dal database.

5.3.2 Processo di valutazione

Assolta la funzione di raccolta delle informazioni sulle raccomandazioni determinate dai modelli di classificazione a fronte di diversi parametri, si può procedere con la fase di *valutazione* (tramite l'evaluation process), usando i dati delle simulazioni per generare una metrica di confronto delle prestazioni dei modelli simulati.

Come già indicato, la metrica di confronto da usare è il *guadagno* ottenibile nell'arco di un intervallo di tempo pari a un anno, usando le raccomandazioni dei modelli di classificazione per decidere quali stock comprare e vendere giorno dopo giorno.

Il guadagno di una generica i-esima simulazione, denominato G^i_{TP}, è ottenuto dalla differenza tra i profitti, P^i_{TP}, e le perdite, L^i_{TP}, accumulati nell'arco di un periodo di tempo TP, dove TP è pari a un anno Y. Profitti e perdite sono determinati dalla somma dei ritorni relativi giornalieri r^j_k ottenuti rispettivamente per ogni raccomandazione corretta o sbagliata del j-esimo stock in ogni k-esimo giorno di trading:

$$G^i_Y = P^i_Y - L^i_Y \qquad (\forall\ sim_i \in \text{SIM}_Y) \tag{5.1}$$

$$P^i_Y = \sum\ p^{\{i,j\}}_k \qquad (\forall\ sim_i \in \text{SIM}_Y,\quad \forall\ s_j \in S,\quad \forall\ t_k \in T) \tag{5.2}$$

$$L^i_Y = \sum\ l^{\{i,j\}}_k \qquad (\forall\ sim_i \in \text{SIM}_Y,\quad \forall\ s_j \in S,\quad \forall\ t_k \in T) \tag{5.3}$$

$$p^{\{i,j\}}_k = \begin{cases} r^j_k & \textbf{if } (\mathtt{r^j_k} > 0 \ \ \&\& \ \ \mathtt{Prediction}^{\{\mathtt{i,j}\}}_{\mathtt{k}} \in (\mathtt{UP}, \mathtt{UP{+}{+}})) \\ -r^j_k & \textbf{if } (\mathtt{r^j_k} < 0 \ \ \&\& \ \ \mathtt{Prediction}^{\{\mathtt{i,j}\}}_{\mathtt{k}} \in (\mathtt{DOWN}, \mathtt{DOWN}{-}{-})) \end{cases} \tag{5.4}$$

$$l^{\{i,j\}}_k = \begin{cases} r^j_k & \textbf{if } (\mathtt{r^j_k} > 0 \ \ \&\& \ \ \mathtt{Prediction}^{\{\mathtt{i,j}\}}_{\mathtt{k}} \in (\mathtt{DOWN}, \mathtt{DOWN}{-}{-})) \\ -r^j_k & \textbf{if } (\mathtt{r^j_k} < 0 \ \ \&\& \ \ \mathtt{Prediction}^{\{\mathtt{i,j}\}}_{\mathtt{k}} \in (\mathtt{UP}, \mathtt{UP{+}{+}})) \end{cases} \tag{5.5}$$

$$r^j_k = \frac{closePrice_{t_k} - closePrice_{t_{k-1}}}{closePrice_{t_{k-1}}} \qquad (\forall\ s_j \in S,\quad \forall\ t_k \in T) \tag{5.6}$$

Le equazioni evidenziano quindi che, per calcolare il guadagno (5.1) di ogni simulazione, è necessario *aggregare* per anno i profitti (5.2) e le perdite (5.3) ottenuti su ogni singola raccomandazione.

Una singola raccomandazione è identificata dall'etichetta predetta dal modello simulato per uno stock s_j nel giorno di trading t_k: ovvero dal termine $\mathtt{Prediction}^{\mathtt{j}}_{\mathtt{k}}$. Se la previsione è concorde con la variazione subita dal prezzo dello stock, il ritorno relativo r^j_k di tale stock è considerato un *profitto* (5.4); in caso contrario il ritorno r^j_k determina una *perdita* (5.5).

NB. Le previsioni `DOWN` o `DOWN−−` implicano l'apertura di posizioni SHORT, che hanno successo se il prezzo dello stock decresce, ovvero se il valore del ritorno r_k^j è negativo (5.6). Di conseguenza, per conteggiare i profitti $p_k^{\{i,j\}}$ derivanti dalle raccomandazioni SHORT, si inverte il segno del ritorno ($-r_k^j$ in 5.4), mentre per le perdite $l_k^{\{i,j\}}$ è sufficiente conteggiare il valore negativo (r_k^j in 5.5).

Per generare le aggregazioni dei profitti e delle perdite, si estraggono i dati raccolti nel `Simulation DB` e si caricano all'interno di una seconda base dati, dedicata all'analisi delle informazioni statistiche generate per ogni simulazione. Questo ulteriore database è quello a cui si fa riferimento con la notazione di `Analysis DB` nello schema dell'architettura BeSt modificata (5.2), indicato all'inizio di questa sezione.

Il processo ETL attraverso il quale viene popolato l'`Analysis DB`, viene eseguito per mezzo dei seguenti passaggi:

1. Estrazione di tutti i *prezzi degli stock*, delle *simulazioni* e delle relative *predizioni*, nell'intervallo dell'*anno di trading* considerato.
2. Identificazione delle previsioni *corrette* e *sbagliate*, in funzione dei prezzi di chiusura degli stock relativi alle date per cui sono state determinate le raccomandazioni.
3. Calcolo dei profitti $p_k^{\{i,j\}}$ e delle perdite $l_k^{\{i,j\}}$ di ogni *singola* predizione, in funzione del ritorno relativo giornaliero (r_k^j) e della correttezza della predizione.
4. Conteggio delle previsioni corrette e sbagliate.
5. Conteggio dei profitti P_Y^i e delle perdite L_Y^i.
6. Calcolo del guadagno G_Y^i *complessivo* dell'anno, da cui poi si derivano il guadagno medio $G^i_{Y\_\text{AVG}}$ e il guadagno giornaliero G_{Y_DAILY}.

Al temine del processo è possibile disporre di un dataset contenente i dati di tutte le simulazioni eseguite per ogni anno di trading, descritte dalla combinazione di parametri utilizzata e dalle metriche di guadagno, grazie alle quali è possibile determinare il *ranking* dei modelli più performanti, come rappresentato dall'esempio in figura 5.5:

Year	**Simulation parameters**						$\mathbf{G_{Y\_AVG}}$	$\mathbf{G_{Y\_DAILY}}$
	Training Days	*Sliding Window*	*Algorithm*	*Label Disc.*	*Cut Point*	*Trade Type*		
2011	10	2	`N. Bayes Kern.`	`BINOMINAL`	3	`LONG`	1.27	1.51
2011	10	3	`N. Bayes Kern.`	`BINOMINAL`	-3	`SHORT`	1.20	1.57
2011	10	3	`Rule Induction`	`BINOMINAL`	-3	`SHORT`	0.76	1.35

Tabella 5.5: Esempio dataset usato per la valutazione dei modelli simulati

Dal momento che il guadagno *complessivo* di un anno di trading G_Y^i dipende dal *numero* di raccomandazioni ottenute dai modelli di classificazione giorno dopo

giorno, che vengono sommate per calcolare i profitti e le perdite complessive, si è scelto di introdurre le metriche del guadagno medio $G^i_{Y\_\text{AVG}}$ e del guadagno giornaliero G_{Y_DAILY} al fine di *normalizzare* il valore aggregato del guadagno complessivo dividendolo rispettivamente per il numero di raccomandazioni applicate e per il numero di giornate di trading:

$$G^i_{Y\_AVG} = \frac{G^i_Y}{\text{COUNT}(\texttt{Prediction}^{\{\texttt{i,j}\}}_{\texttt{k}})} = \text{AVG}(G^i_Y) \tag{5.7}$$

$$G^i_{Y\_\text{DAILY}} = \frac{G^i_Y}{\text{COUNT}(\text{DISTINCT } t_k)} \tag{5.8}$$

Il guadagno medio (5.7) fornisce quindi una misura più accurata delle prestazioni dei classificatori, definendo la percentuale media di guadagno ottenibile da ogni singola raccomandazione nell'anno di trading testato. Mentre il guadagno giornaliero (5.8) esprime quanto, mediamente, si sarebbe guadagnato ogni singola giornata di trading simulando l'applicazione di uno specifico metodo.

Oltre a queste metriche, i risultati delle simulazioni mantenuti nell'`Analysis DB` possono essere usati per generare le stesse metriche definite nella sezione 4.2.2, in modo da adottare lo stesso criterio di valutare delle prestazioni di modelli usato nel sistema BeSt originale ([40]):

- Numero totale di raccomandazioni — $\texttt{NumP} = \text{COUNT}(\texttt{Prediction}^{\{\texttt{i,j}\}}_{\texttt{k}})$
- Guadagno medio con commissione (TAX) — $G\text{TAX}^i_{Y\_\text{AVG}} = \text{AVG}(G\text{TAX}^i_Y)$
- Deviazione standard del guadagno medio con commissione (TAX) — $G\text{TAX}^i_{Y\_\text{DEVSTD}} = \text{DEVSTD}(G\text{TAX}^i_Y)$
- Guadagno massimo con commissione (TAX) — $G\text{TAX}^i_{Y\_\text{MAX}} = \text{MAX}(G\text{TAX}^i_Y)$

Facendo sempre riferimento ai risultati delle simulazioni, l'evaluation process analizza anche uno scenario applicativo più vicino a quello reale, introducendo un ulteriore parametro di configurazione del sistema: lo *stop loss*, con il quale è possibile limitare le perdite a fronte di decisioni di trading sbagliate. Nella sezione 5.3.3 si descrive come si possa usare il sistema per valutare l'effetto dell'impostazione di una soglia di stop loss sui ritorni giornalieri.

5.3.3 Ottimizzazione dei ritorni

Ottimizzare i ritorni nello scenario dell'intraday trading implica massimizzare i profitti e limitare le perdite, come già definito nella sezione 3.2.2. In considerazione

di questo obiettivo, nella fase di valutazione dei risultati, si considera anche la possibilità di limitare le perdite simulando l'impiego di una strategia di stop loss.

Per questo tipo di simulazione non è richiesto l'intervento diretto sui modelli di classificazione, ma è sufficiente sfruttare le informazioni relative ai prezzi storici degli stock per ri-calcolare i profitti e le perdite nel caso in cui il trader avesse potuto chiudere automaticamente le posizioni in perdita, una volta raggiunta la soglia di stop loss.

Ponendo una maggiore attenzione sull'applicazione delle raccomandazioni in uno scenario di intraday trading più simile a quello realmente svolto dai trader, per ogni simulazione sono stati definiti i parametri `OperationCost` e `StopLossPerc`, che rappresentano rispettivamente la percentuale della commissione del broker e la percentuale di perdita da usare per determinare la soglia di stop loss.

Il costo della commissione del broker, oltre a rendere più realistica la simulazione, ha anche fornito uno strumento per penalizzare i modelli che generano un alto numero di raccomandazioni, poiché, sottraendo il costo di ogni operazione al valore del guadagno, si abbassano le misure di performance di tali modelli.

Lo stop loss, invece, può determinare un miglioramento o, come verrà descritto nel capitolo 6, anche un peggioramento del guadagno complessivo e, essendo valutato insieme alle variabili delle simulazioni, può aiutare a definire una strategia di trading più elaborata da offrire al trader, nei casi in cui fosse in grado di incrementare il rendimento.

Come fatto per le altre variabili di simulazione, si riportano di seguito i possibili valori dei due nuovi parametri:

`OperationCost` { `0.15` }
`StopLossPerc` { `1.0`, `1.5`, `2.0` }.

NB. Contrariamente ai parametri indicati nelle sezioni 5.2.1 e 5.2.2, queste variabili non vengono usate nel processo di simulazione, ma solamente in quello di valutazione, in quanto sono definite esclusivamente all'interno dell'`Analysis DB` per costruire nuove statistiche e misure di performance.

Le principali metriche di confronto dei modelli simulati, generate con l'introduzione dei nuovi parametri sono il *guadagno tassato*, $G\text{TAX}_Y^i$, e il guadagno ottenuto applicando una *strategia di stop loss*, $G\text{SL}_Y^i$:

$$G\text{TAX}_Y^i = P\text{TAX}_Y^i - L\text{TAX}_Y^i$$

$$G\text{SL}_Y^i = P\text{SL}_Y^i - L\text{SL}_Y^i$$

$$r\text{TAX}_k^j = \frac{closePrice_{t_k} - closePrice_{t_{k-1}}}{closePrice_{t_{k-1}}} - \texttt{OperationCost}$$

$$r\text{SL}_k^j = \begin{cases} -\texttt{StopLossPerc} - \texttt{OperationCost} & \textbf{if } (StopLoss\ used) \\ r\text{TAX}_k^j & \textbf{if } (StopLoss\ NOT\ used) \end{cases}$$

Fondamentalmente i passaggi per il calcolo dei due nuovi guadagni sono gli stessi indicati dalle equazioni 5.2, 5.3, 5.4 e 5.5, con l'unica differenza nel valore del *ritorno relativo giornaliero*, che è quello interessato dalle modifiche dell'`OperationCost` e dello `StopLossPerc`.

Nella sezione seguente si approfondisce il modo in cui è stato simulato e valutato l'intervento dello stop loss sulle prestazioni dei modelli.

Strategia di Stop Loss

Utilizzare una strategia di stop loss permette di uscire automaticamente dal mercato nel momento in cui si verifica una variazione di prezzo contraria alla scelta di investimento fatta, tale per cui la perdita di denaro supera una soglia di tolleranza prefissata.

Per simulare questa procedura avendo a disposizione i prezzi storici degli stock, registrati solamente al termine di ogni giornata di trading, non è possibile identificare il momento esatto in cui si è verificata una variazione oltre la soglia dello stop loss, ma si possono sfruttare le informazioni relative al prezzo *minimo* e *massimo* per scoprire, almeno, se la soglia sia stata superata o meno.

Fortunatamente i repository dei dati finanziari disponibili sul web, tra cui YAHOO! FINANCE, rendono accessibili non solo i prezzi di chiusura, ma anche quelli di apertura, minimo e massimo di ogni giorno di trading. Pertanto il componente di Data Gathering & Data Preparation può scaricare il set completo dei prezzi OHLC[9] direttamente su database, per rendere disponibili tali informazioni alla simulazione della strategia di stop loss.

Avendo quindi a disposizione tutte le informazioni necessarie, per determinare l'effetto della strategia di stop loss sui guadagni dei modelli simulati, occorre integrare nel processo ETL che popola l'`Analysis DB` la *soglia di stop loss* $\texttt{SL\_}Price_{t_k}$, che viene calcolata come segue:

$$\texttt{SL\_}Price_{t_k} = \begin{cases} closePrice_{t_k} - closePrice_{t_k} * \texttt{StopLossPerc}/100 & \textbf{if } (\texttt{TradeType} == \texttt{LONG}) \\ closePrice_{t_k} + closePrice_{t_k} * \texttt{StopLossPerc}/100 & \textbf{if } (\texttt{TradeType} == \texttt{SHORT}) \end{cases}$$

Come descritto dalla formula matematica, nel caso di posizioni LONG, la soglia viene fissata al prezzo che corrisponde alla percentuale `StopLossPerc` di *decremento* del prezzo di acquisto, mentre, nel caso SHORT, la soglia corrisponde alla variazione percentuale di *incremento* del prezzo di vendita. Per esempio: se si acquista una quota ad un prezzo di chiusura pari a 100, si imposta il parametro `StopLossPerc = 1.0%` e, nel corso della successiva giornata, il valore dello stock scende al di sotto

[9] Acronimo per indicare i prezzi Open, High, Low, Close degli strumenti finanziari.

di 99 ($100 - 1\% = 99$), allora si raggiunge la soglia di stop loss e la posizione LONG viene automaticamente chiusa con una perdita pari all'1% del denaro investito.

Dopo aver calcolato ed associato il valore della soglia di stop loss ad ogni stock raccomandato dai modelli di classificazione, si può determinare il ritorno $r\texttt{SL}_k^j$ di ogni raccomandazione, considerando di sfruttare la strategia di stop loss:

$$r\texttt{SL}_k^j = \begin{cases} -\texttt{StopLossPerc} - \texttt{OperationCost} & \textbf{if } (\ (\texttt{TradeType} == \texttt{LONG}\ \ \&\& \\ & \qquad lowPrice_{t_k} \leq \texttt{SL\_}Price_{t_k}) \\ & \quad \mid\mid (\texttt{TradeType} == \texttt{SHORT}\ \ \&\& \\ & \qquad highPrice_{t_k} \geq \texttt{SL\_}Price_{t_k})\ ) \\ r_k^j - \texttt{OperationCost} & \textbf{otherwise} \end{cases}$$

In quest'ultima rappresentazione si vede come sono utilizzati i prezzi minimo ($lowPrice_{t_k}$) e massimo ($highPrice_{t_k}$) per verificare se, nel corso della giornata di trading, il prezzo dello stock raccomandato ha superato la soglia di stop loss. Se lo stop loss viene applicato, il ritorno è costituito da una perdita pari alla percentuale `StopLossPerc`, meno il costo dell'operazione. Se invece non si verificano le condizioni necessarie a limitare le perdite, il ritorno giornaliero viene calcolato in base al prezzo di chiusura e al costo della commissione del broker, ovvero $r\textsc{tax}_k^j$.

Sebbene applicando questa strategia si riescano a limitare le perdite al valore prefissato di $-\texttt{StopLossPerc}$, l'uscita dal mercato si attiva solamente se le variazioni oltre la soglia di stop loss si verificano durante le ore di trading. Nel caso di eventuali movimenti dei prezzi *overnight*, tali per cui uno stock, comprato o venduto alla chiusura del giorno precedente, si presenti all'apertura del mercato nel giorno successivo con un prezzo già oltre la soglia di tolleranza, il meccanismo attuale non può intervenire.

Per contenere le perdite anche a fronte di un prezzo di apertura inatteso, è stata introdotta una variante delle strategia, dove si rende *dinamico* il valore della soglia di stop loss.

Dynamic stop loss

Lo stop loss dinamico è stato implementato solo dopo aver osservato che i risultati di alcune simulazioni conseguivano ritorni molto peggiori rispetto a quelli ottenuti senza applicare lo stop loss (ovvero calcolando il rendimento direttamente sul prezzo di chiusura del giorno seguente). In particolare si è notato che lo stop loss causava l'uscita dal mercato al prezzo di apertura ma, nel corso della giornata, il prezzo dello stock tornava a muoversi nella direzione desiderata. In molte di queste situazioni il prezzo di apertura ($openPrice_{t_k}$) *coincideva* addirittura con il prezzo minimo di tutta la giornata ($lowPrice_{t_k}$). Pertanto, per evitare situazioni di questo tipo e provare ad *attendere* una possibile variazione favorevole del prezzo, si è introdotto il meccanismo dello stop loss dinamico.

Con lo stop loss dinamico si modifica il valore della soglia di tolleranza, rendendola uguale al prezzo di apertura, in modo da impedire la chiusura delle posizioni all'apertura del mercato. Tuttavia, per giustificare una modifica della soglia di tolleranza impostata inizialmente, si procede con l'attivazione della strategia di stop loss dinamico solo nel caso in cui il prezzo di apertura causi un'uscita dal mercato con una perdita maggiore a quella prevista dallo stop loss statico, sommata al costo della transazione.

Sintetizzando quanto indicato, la soglia dello stop loss dinamico $\texttt{DYNSL\_}Price_{t_k}$ è calcolata come indicato di seguito:

$$\texttt{DYNSL\_}Price_{t_k} = \begin{cases} openPrice_{t_k} & \textbf{if } (\ (\texttt{TradeType} == \texttt{LONG}\ \ \&\& \\ & \qquad openPrice_{t_k} \leq (\texttt{SL\_}Price_{t_k} - \\ & \qquad\ \ \texttt{OperationCost}/100 * closePrice_{t_{k-1}})) \\ & \quad || (\texttt{TradeType} == \texttt{SHORT}\ \ \&\& \\ & \qquad openPrice_{t_k} \geq (\texttt{SL\_}Price_{t_k} + \\ & \qquad\ \ \texttt{OperationCost}/100 * closePrice_{t_{k-1}}))\ \ ) \\ \texttt{SL\_}Price_{t_k} & \textbf{otherwise} \end{cases}$$

Come si vede dall'espressione che descrive la condizione di attivazione dello stop loss dinamico, la soglia viene fissata al prezzo di apertura nel tentativo di evitare una perdita maggiore a quella gestita dallo stop loss statico.

In conclusione, integrando la strategia di stop loss, sia statico che dinamico, all'interno del dataset usato per la valutazione dei modelli simulati, risultano disponibili nuove metriche di confronto e una cardinalità sei volte superiore a quella iniziale, poiché sei sono le combinazioni dei parametri `OperationCost`, `StopLossPerc`, `IsDynamicStopLoss`.

Year	**Sim. param.**		**Operation Cost %**	**StopLoss%**	**Is DYN$_{\mathbf{SL}}$**	**G$_{\mathbf{Y}_\mathrm{TAX}}$**	**G$_{\mathbf{Y}_\mathrm{SL}}$**
	. . .	. . .					
2011	. . .	. . .	0.15	1.0	*false*	1.12	0.24
2011	. . .	. . .	0.15	1.0	*true*	1.12	0.30
2011	. . .	. . .	0.15	1.5	*false*	1.12	0.31
2011	. . .	. . .	0.15	1.5	*true*	1.12	0.35
2011	. . .	. . .	0.15	2.0	*false*	1.12	0.47
2011	. . .	. . .	0.15	2.0	*true*	1.12	0.47
2011	. . .	. . .	0.15	1.0	*false*	0.95	0.20
2011	. . .	. . .	0.15	1.0	*true*	0.95	0.24
2011	. . .	. . .	0.15	1.5	*false*	0.95	0.20
2011	. . .	. . .	0.15	1.5	*true*	0.95	0.22
2011	. . .	. . .	0.15	2.0	*false*	0.95	0.24
2011	. . .	. . .	0.15	2.0	*true*	0.95	0.24

Tabella 5.6: Esempio dataset con le metriche derivanti da OperationCost e StopLoss

5.4 Self-tuning dei modelli di classificazione

La fase finale del processo di valutazione è quella in cui si considerano tutte le informazioni archiviate nell'`Analysis DB` per scegliere il modello di classificazione e la combinazione di parametri più adatta a fornire le raccomandazioni degli stock per l'intraday trading.

Come verrà dimostrato nel capitolo 6, dove si presentano i risultati ottenuti nel corso della sperimentazione, le performance dei classificatori cambiano in modo significativo in funzione della distribuzione dei dati e dei parametri scelti: di conseguenza diventa necessario *valutare* continuamente se la strategia usata per estrarre le raccomandazioni risulta sempre attendibile o meno. La funzione di self-tuning cerca di soddisfare questa esigenza auto-validando le performance del sistema e modificando la tecnica di classificazione utilizzata per rispondere efficacemente all'evoluzione delle condizioni di mercato, mirando così alla realizzazione di un sistema più robusto ed affidabile.

Anche l'implementazione della fase di self-tuning ha richiesto diverse scelte progettuali. Per spiegare la metodologia adottata, di seguito si approfondiscono le principali decisioni che sono state prese.

Decisione 1. Modello predittivo

Il modello predittivo impiegato per il self-tuning dei modelli di classificazione, fondamentalmente, svolge il ruolo di *selettore* di una tra N possibili combinazioni $\{algoritmo, \{parametri\}\}$ e, per prendere questa decisione, il modello deve riuscire a prevedere quali saranno le prestazioni future di ogni combinazione.

Il primo metodo proposto per questo scopo è stato un albero decisionale che, data la sua capacità di generare un modello altamente interpretabile, avrebbe permesso di comprendere chiaramente non solo come scegliere la combinazione migliore, ma anche di identificare i parametri più influenti sulle perfomance della classificazione.

Sperimentalmente, però, diversi algoritmi di tipo `decision tree` non si sono dimostrati sufficientemente efficaci nella classificazione delle simulazioni: principalmente a causa di un forte sbilanciamento nella distribuzione dei dati relativi ai guadagni, tali per cui solo poche combinazioni $\{algoritmo, \{parametri\}\}$ possono essere effettivamente classificate come "*migliori delle altre*". Inoltre, per eseguire la classificazione in funzione delle metriche di guadagno, è nuovamente necessario definire un meccanismo di discretizzazione dell'etichetta di classe e di selezione delle predizioni finali.

A seguito delle difficoltà incontrate nell'adozione di un ulteriore metodo di classificazione, la scelta del modello predittivo si è spostata verso le tecniche di regressione, che si sono rivelate più adeguate ad affrontare il problema di data mining in esame: ovvero prevedere un valore numerico (il guadagno delle simulazioni) in funzione delle variabili conosciute (i parametri di simulazione e le condizioni di mercato).

Sfruttare un metodo di regressione per prevedere i guadagni futuri delle diverse configurazioni dei classificatori, inoltre, permette di individuare subito la combinazione da scegliere, semplicemente estraendo la simulazione per cui è stato previsto il guadagno più alto degli altri.

A seguito degli esperimenti svolti, è stato scelto un algoritmo di regressione polinomiale che, sebbene non offra lo stesso livello di interpretabilità di un albero decisionale, permette comunque di rappresentare il peso di ogni attributo considerato attraverso i coefficienti del polinomio risultante.

Decisione 2. Struttura dati

Stabilita la tecnica da usare, la scelta successiva riguarda la definizione del dataset con cui addestrare il modello predittivo. Tale struttura deve rappresentare il problema di data mining che si vuole risolvere che, come indicato nel punto precedente, implica prevedere il guadagno futuro di ogni simulazione in funzione delle condizioni di mercato. Quindi, le variabili che si considerano sono:

- le combinazioni $\{algoritmo,\ \{parametri\}\}$;
- l'intervallo di tempo {TP} in cui si misurano i rendimenti di ogni configurazione dei classificatori
- le condizioni di mercato relative periodo di tempo precedente $\texttt{TP}_{\texttt{k}-1}$ a quello in cui sono state eseguire le simulazioni;
- il rendimento ottenuto da ogni combinazione $\{algoritmo,\ \{parametri\}\}$;.

Time period	**Simulation parameters**							**Market metrics-1**		**Gain metric**
{TP}	Training Days	Sliding Window	Algorithm	Label Disc.	Cut Point	Trade Type	Stop Loss	...	...	...

Tabella 5.7: Esempio struttura dati

La tabella 5.7 rappresenta in modo generico la struttura che dovrà avere il training set del modello di regressione e mette in evidenza l'esigenza di definire: la misura delle *condizioni di mercato* (`Market metrics`), la misura delle *prestazioni* delle simulazioni (`Gain metrics`) e l'*intervallo di tempo* ({TP}) su cui calcolare tali metriche. I periodi di tempo considerati saranno poi chiariti ed approfonditi meglio alla decisione 6.

Decisione 3. Rappresentazione delle condizioni di mercato

Come già anticipato, per rappresentare le condizioni di mercato si derivano delle statistiche a partire dalla serie dei prezzi storici degli stock coinvolti nel processo di simulazione e dei prezzi di chiusura dell'indice di mercato a cui appartengono tali stock. Le statistiche considerate sono elencate nella tabella 5.8:

Variabile	Descrizione
Serie dei prezzi storici dei singoli stock	
`sum_stock_close-1`	somma prezzi di chiusura;
`avg_stock_close-1`	media prezzi di chiusura;
`stddev_stock_close-1`	deviazione standard prezzi di chiusura;
`avg_stock_varLowHigh-1`	media dell'ampiezza(*1);
`stddev_stock_varLowHigh-1`	deviazione standard dell'ampiezza(*1);
Serie dei prezzi storici dell'indice di mercato	
`delta_avg_UPDOWN_Ratio-1`	rapporto tra la media dei prezzi in salita e quella dei prezzi in discesa;
`count_UPDOWN_Ratio-1`	rapporto tra il numero dei giorni con prezzo in salita e quello con prezzo in discesa;
`delta_avg_trendChange_Ratio-1`	rapporto tra la media dei prezzi nei giorni in cui il trend è cambiato con quelli in cui è stato mantenuto;
`count_trendChange_Ratio-1`	rapporto tra il numero dei giorni in cui il trend è cambiato con quelli in cui è stato mantenuto;
`delta_avg-1`	media;
`delta_median-1`	mediana;
`delta_stddev-1`	deviazione standard;
`mean_delta_deviation-1`	media della deviazione aritmetica;
`mean_delta_amplitude-1`	media dell'ampiezza(*1);
`mean_delta_gradient-1`	media del gradiente di regressione lineare;
`mean_delta_centroid-1`	media del centroide(*2);
`sma10-1`	media mobile ultimi 10 giorni;
`sma20-1`	media mobile ultimi 20 giorni;
`sma50-1`	media mobile ultimi 50 giorni.

(*1) differenza tra il prezzo massimo e minimo;
(*2) valore mediamente centrale della serie dei prezzi di chiusura.

Tabella 5.8: Indicatori tecnici derivati dalla serie storica dei prezzi

Le misure estratte dalle serie dei prezzi storici dei singoli stock servono a descrivere la variabilità dei prezzi nel periodo in cui sono state eseguite le simulazioni. A queste metriche, che includo i dati effettivamente usati per il training dei classificatori simulati, si aggiungono le misure calcolate sull'indice di mercato, che si possono considerare come indicatori generici dell'andamento del mercato di riferimento.

Tutte le variabili indicate in tabella terminano con il suffisso `"-1"`, in quanto, come verrà spiegato nella decisione 6, si riferiscono all'intervallo di tempo precedente a quello per cui si vogliono prevedere i guadagni delle simulazioni.

Decisione 4. Attributo label

Il ruolo di *label* deve essere assegnato ad un attributo relativo ad una delle metriche di guadagno calcolate dall'evaluation process, come descritto nella sezione 5.3.2. Questo attributo sarà quello che il modello di regressione polinomiale cercherà di calcolare in funzione dei parametri di simulazione ($\{algoritmo,\ \{parametri\}\}$) e delle metriche relative alle condizioni di mercato.

La metrica del rendimento che si è scelto di utilizzare come attributo di classe è il guadagno medio giornaliero a meno di commissioni ($\text{GTAX}_{\text{DAILY}}$), in quanto, come si potrà osservare in modo più dettagliato nel capitolo 6, esprime l'efficacia complessiva di tutte le predizioni ottenute dalla classificazione, penalizzando, attraverso il costo di ogni operazione, un numero troppo alto di raccomandazioni: pertanto costituisce la misura più adeguata a rappresentare l'obiettivo di guadagno che si vuole raggiungere.

Decisione 5. Discretizzazione degli attributi nominali

L'algoritmo di regressione polinomiale, che è stato scelto per prevedere i futuri guadagni giornalieri delle simulazioni, implementa una forma di regressione lineare in cui la relazione tra la label e le variabili del problema è modellata come un polinomino di grado n[43]. Tutti gli attributi usati per il training del modello, pertanto, devono essere valori numerici (utilizzabili per costruire il polinomio che approssima le previsioni dei rendimenti).

Nella struttura dati scelta per questo problema (decisione 2), le uniche variabili non numeriche sono:

`Algorithm`	{ `K_NN`, `Naive Bayes`, `Decision Tree`, . . . }
`LabelDiscretization`	{ `BINOMINAL_FIXED`, `POLY_4BINS_FIXED` }
`TradeType`	{ `LONG`, `SHORT` }.

Per esprimere in forma numerica queste variabili si è scelto di applicare una semplice conversione, trasformando ogni valore in un intero[10]:

`Algorithm`	{ 0, 1, 2, . . . , 19 }
`LabelDiscretization`	{ 0, 1 }
`TradeType`	{ 0, 1 }.

Decisione 6. Periodi di tempo da usare per il training del modello

Addestrare il modello di regressione a prevedere i guadagni ottenibili da ogni combinazione $\{algoritmo,\ \{parametri\}\}$, in funzione delle condizioni di mercato, implica costruire un training set attraverso il quale si possa modellare opportunamente la relazione [*condizioni di mercato attuali*]–[*rendimenti futuri*]. Per preparare i dati di training è quindi necessario:

[10] Per la trasformazione è stato usato l'operatore RapidMiner `Nominal to Numerical`[42].

- ➢ associare alle metriche di guadagno calcolate nel periodo $\texttt{TP}_\texttt{k}$, le metriche di mercato calcolate nel periodo precedente $\texttt{TP}_{\texttt{k}-1}$;
- ➢ definire quanti esempi, ovvero quanti periodi temporali ($\{\texttt{TP}\}$), considerare per il training del modello.

Per risolvere il primo requisito, si parte dalla struttura dati indicata nella tabella 5.7 (decisione 2) e si applica una trasformazione in modo da "*sfasare*" le metriche di mercato rispetto a quelle di guadagno, ottenendo una disposizione come quella rappresentata nella tabella 5.9.

Time period	**Simulation parameters**	**Market metrics** (-1)	**Gain metric**
$\texttt{TP}_\texttt{k}$	... ...	... $\{\texttt{metrics}\}\_\texttt{TP}_{\texttt{k}-1}$ ...	$\text{GTAX}^{\texttt{i}}_{\text{DAILY}}\_\texttt{TP}_\texttt{k}$

Tabella 5.9: Struttura dati con metriche di mercato al periodo precedente

A questo punto, per decidere quanti periodi temporali usare per addestrare il modello, bisogna stabilire anche quanto deve essere ampio ogni periodo. A seguito degli esperimenti svolti, si è scelto di aggregare le metriche in esame per trimestre e di costruire il training set usando i dati degli ultimi 4 trimestri, ovvero dell'anno precedente:

`Time period`	1 trimestre
`Training set size`	ultimi 4 trimestri (ultimo anno).

La scelta del trimestre, sperimentalmente, è risultata l'unica in grado di fornire performance accettabili. Mentre la scelta del numero di periodi deriva dal fatto che le prestazioni di ogni simulazione sono *valutate* per anno, come indicato nella sezione 5.3.2, pertanto, addestrare il modello sui dati prestazionali di un anno, ha significato usare le metriche relative a 4 trimestri.

Per visualizzare correttamente quanto indicato, nelle tabelle 5.10 e 5.11 si riporta un esempio in cui si prevede il guadagno giornaliero che le simulazioni otterrebbero nel secondo trimestre del 2011, addestrando il modello di regressione sui dati di un anno, a partire dal secondo trimestre del 2010 al primo trimestre del 2011. In ogni `time period` ($\{\texttt{TP}_\texttt{k}\}$), le metriche che si considerano sono sempre relative a tutte le simulazioni eseguite in quell'intervallo di tempo: quindi, ogni riga indicata nelle tabelle 5.10 e 5.11 è in realtà costituita dalle N combinazioni $\{algoritmo, \{parametri\}\}$. A fronte della previsione di tutti i valori futuri $G\text{TAX}_{\text{DAILY}}$, il sistema selezionerà la combinazione che ha ottenuto il rendimento più alto degli altri.

Time period	**Simulation parameters**	**Market metrics**	**G**$\text{TAX}_{\text{DAILY}}$
2010-2	... ...	... $\texttt{metrics}\_{2010-1}$ ...	...
2010-3	... ...	... $\texttt{metrics}\_{2010-2}$ ...	...
2010-4	... ...	... $\texttt{metrics}\_{2010-3}$ ...	...
2011-1	... ...	... $\texttt{metrics}\_{2010-4}$ ...	...

Tabella 5.10: Esempio training set

Time period	**Simulation parameters**	**Market metrics**	[*Target*] **G**$\text{TAX}_{\text{DAILY}}$
2011-2	... ...	... $\texttt{metrics}\_{2011-1}$ ...	**?**

Tabella 5.11: Esempio su cui si applica il modello di regressione

Riepilogo

Per concludere questa sezione, di seguito si riassumono brevemente i punti chiave di tutte le scelte progettuali relative al self-tuning dei modelli di classificazione:

1. Modello predittivo

   Decisione: Modello di regressione polinomiale per prevedere i guadagni futuri ottenibili da ogni simulazione. Il classificatore e la relativa configurazione di parametri saranno selezionati estraendo la combinazione per cui è stato previsto il guadagno più alto degli altri.

2. Struttura dati

   Decisione: Struttura composta da:
   - periodo di tempo {`TP`};
   - parametri di simulazione;
   - metriche che esprimono le condizioni di mercato calcolate sul periodo precedente $\texttt{TP}_{\texttt{k}-1}$;
   - metriche di guadagno, calcolate sul periodo $\texttt{TP}_{\texttt{k}}$.

   La struttura definitiva è rappresentata nella tabella 5.9.

3. Rappresentazione delle condizioni di mercato

   Decisione: Metriche definite nella tabella 5.8.

4. Attributo label

   Decisione: Guadagno giornaliero a meno di commissioni ($G\text{TAX}_{\text{DAILY}}$).

5. Discretizzazione degli attributi nominali

   Decisione: Conversione degli attributi `LabelDiscretization`, `Algorithm` e `TradeType` in interi a partire da 0.

6. Periodi di tempo da usare per il training del modello

   Decisione: I 4 trimestri precedenti al trimestre da prevedere.

Nel capitolo successivo si descrivono gli esperimenti svolti sul sistema BeSt esteso riportando sia i risultati ottenuti dai singoli classificatori che quelli conseguiti tramite la funzione di self-tuning.

Capitolo 6

Esperimenti svolti

Gli esperimenti svolti in questa tesi hanno coinvolto i prezzi delle azioni che fanno parte dell'indice FTSE.MIB, considerando i dati storici da metà del 2009 alla fine del 2017. Tali esperimenti sono stati eseguiti su una workstation virtuale `Ubuntu Linux 14.04 LTS` con 4 processori da 2.60 GHz e 10 GB di RAM.

I risultati ottenuti dalla sperimentazione possono essere suddivisi in due categorie distinte: una relativa alle raccomandazioni degli stock per l'intraday trading, ottenute dal sistema BeSt modificato per supportare la classificazione, l'altra, relativa alla scelta del modello e dei parametri di configurazione più adatti ad affrontare i cambiamenti delle condizioni di mercato. Nella sezione 6.1 si descrivono gli elementi relativi alla simulazione delle diverse configurazioni dei modelli di classificazione e si analizzano le prestazioni ottenute da tali simulazioni. Nella sezione 6.2 si valutano invece i risultati ottenuti sfruttando la funzionalità di self-tuning dei modelli che, come visto nel capitolo 5, svolge anch'essa l'analisi delle simulazioni, ma attraverso un processo automatico che include un algoritmo di regressione.

6.1 Simulazione dei classificatori

La simulazione dei metodi di classificazione comprende in totale **57600** esperimenti: ognuno dei quali si distingue dagli altri per dati storici, algoritmo e parametri di configurazione. Prima di analizzare i risultati di questi esperimenti, di seguito si descrivono brevemente le scelte progettuali e le motivazioni che hanno portato ad ottenere questo numero di simulazioni.

In un primo disegno della metodologia di sperimentazione il simulation process era inteso esclusivamente per automatizzare l'esecuzione degli algoritmi di classificazione con tutte le combinazioni possibili dei parametri identificati nelle sezioni 5.2.1 e 5.2.2, allo scopo di individuare i classificatori più performanti per raccomandare gli stock nell'intraday trading. Osservando i risultati ottenuti sui dati storici degli anni 2011 e 2013 (gli stessi utilizzati per la sperimentazione descritta

in [40]) si è constatato, però, che esiste più di una soluzione valida al problema e che scegliere quella migliore non è un compito semplice. Pertanto si è deciso di delegare la funzione di selezione del modello e dei parametri ad un ulteriore algoritmo di machine learning, così da rendere *automatica* la scelta del classificatore ottimale.

Per poter addestrare (e validare) il modello predittivo a scegliere qual è la configurazione più valida in base ai dati raccolti dalle simulazioni, si è reso necessario disporre di più dati di performance dei classificatori e, quindi, si è esteso il processo di simulazione da due a otto anni di trading, considerando le raccomandazioni ottenute sui dati storici dal 2010 al 2017. Da questo incremento dei dati da analizzare è conseguito un notevole aumento del tempo e dello spazio di archiviazione richiesti per gestire tutte le simulazioni e, a sua volta, ha reso necessario ridurre il numero di algoritmi di classificazione testati da 61 candidati[1] a 20.

A questo punto, per ognuno dei 20 algoritmi scelti, si simulano le raccomandazioni ottenibili nell'arco di un anno trading combinando i parametri di configurazione come rappresentato nello schema seguente:

```
Algorithm
├── (n = 1) StockMarketIndex ..... {FTSE.MIB}
├── (n = 5) TrainingDays ..... {10, 15, 20, 25, 30}
├── (n = 4) SlidingWindow ..... {2, 3, 4 5}
├── (n = 1) MinConfidence ..... {0.5}
├── (n = 1) LabelDiscretization ..... {BINOMINAL}
│   ├── (n = 1) TradeType ..... {LONG}
│   │   └── (n = 4) FixedCutPoint ..... {1, 2, 3, 4}
│   └── (n = 1) TradeType ..... {SHORT}
│       └── (n = 4) FixedCutPoint ..... {−1, −2, −3, −4}
└── (n = 1) LabelDiscretization ..... {POLY_4BINS}
    └── (n = 2) TradeType ..... {LONG, SHORT}
        └── (n = 5) FixedCutPoint ..... {0, 1, 2, 3, 4}
```

In totale si considerano **360** combinazioni di parametri per ogni algoritmo in un anno di trading. Il conteggio delle combinazioni è riassunto nei seguenti punti:

- in ogni simulazione ricorrono sempre tutti i valori assegnati alle variabili `StockMarketIndex`, `TrainingDays`, `SlidingWindow` e `MinConfidence`
  $\rightarrow 1 * 5 * 4 * 1 = 20$

[1] L'elenco di tutti gli algoritmi di classificazione testati è riportato in appendice nella tabella A.1.

- a questi si aggiungono i valori di `LabelDiscretization`, `FixedCutPoint` e `TradeType`, che sono organizzati secondo la *strategia di discretizzazione della label*, definita nella tabella 5.3:
  → 4 `LONG` + 4 `SHORT` = 8 per discretizzazione binomiale
  → 5 `LONG` + 5 `SHORT` = 10 per discretizzazione su 4 classi
- Tot → $20 * (4 + 4) + 20 * 10) = 360$.

Quindi, considerando 20 algoritmi di classificazione, nell'arco di un anno di trading si eseguono $360 * 20 =$ **7200** simulazioni che, per 8 anni di trading distinti, diventano $7200 * 8 =$ **57600** simulazioni:

```
Simulazioni per algoritmo        :    360
Simulazioni per anno di trading  :   7200
Simulazioni totali               :  57600
```

6.1.1 Scelta degli algoritmi per le simulazioni

Come già indicato, i 20 classificatori usati nel processo di simulazione sono stati scelti a partire da un insieme di 61 algoritmi applicabili. L'elenco degli algoritmi scelti è riportato nella tabella 6.1.

Categoria	**Algoritmo**	**Classificazione polinomiale**
Decision tree	`Decision_Tree`	✓
Decision tree	`Decision_Stump`	✓
Decision tree	`Random_Tree`	✓
Decision tree	`W_REPTree`	✓
Decision tree	`W_RandomForest`	✓
Decision tree	`W_BFTree`	✓
Decision tree	`W_Decision_Stump`	✓
Decision tree	`W_J48`	✓
Decision tree	`W_J48graft`	✓
Decision tree	`W_LADTree`	✓
Decision tree	`W_NBTree`	✓
Decision tree	`W_SimpleCart`	✓
Bayesian classifier	`Naive_Bayes`	✓
Bayesian classifier	`Naive_Bayes_Kernel`	✓
Rule induction	`Rule_Induction`	✓
Rule induction	`W_Jrip`	✓
Bayesian network	`W_BayesNet`	✓
Bayesian network	`W_BIFReader`	✓
Bayesian network	`W_BayesNetGenerator`	✓
Bayesian network	`W_EditableBayesNet`	✓

Tabella 6.1: Algoritmi di classificazione selezionati

Per decidere quali modelli escludere sono stati considerati principalmente i risultati ottenuti dagli esperimenti eseguiti sui prezzi storici degli anni 2011 e 2013 (nella prima fase della sperimentazione): in particolare, sul *numero medio di predizioni giornaliere* e sul *guadagno assoluto* conseguiti da usando ciascun algoritmo. Considerando lo scenario applicativo dell'intraday trading, sono stati esclusi tutti i modelli che, mediamente, hanno prodotto meno di una raccomandazione al giorno e, procedendo in modo analogo per i guadagni, sono stati scartati anche gli algoritmi che, nei periodi osservati, generavano rendite nettamente inferiori rispetto agli altri[2].

Oltre a questi due criteri, per garantire che fossero simulate sempre tutte le modalità di discretizzazione dell'attributo di classe, si è scelto di escludere tutti i modelli che supportano solamente attributi di classi binomiali. Infatti, come evidenzia la tabella 6.1, tutti gli algoritmi selezionati supportano la classificazione polinomiale[3].

A questo punto, chiariti i criteri utilizzati per filtrare il numero di algoritmi e ridurre il numero di simulazioni, nelle sezioni seguenti si riportano i risultati osservati per i 20 classificatori selezionati.

6.1.2 Prestazioni generali degli algoritmi

Per valutare *globalmente* le prestazioni di ogni algoritmo, si calcolano le seguenti metriche sui risultati ottenuti complessivamente da tutte le combinazioni di parametri sugli otto anni di trading in cui sono state eseguite le simulazioni:

- `NumP` → numero medio di predizioni per giornata di trading;
- `A%` → accuratezza (rate di predizioni corrette);
- $\texttt{GTAX}_{\text{AVG}}$ → guadagno medio su ogni previsione a meno del costo delle transazioni (`OperationCost` $= 0.15\%$);
- $\texttt{GTAX}_{\text{DAILY}}$ → guadagno medio giornaliero a meno del costo delle transazioni (`OperationCost` $= 0.15\%$).

NB. Queste metriche saranno riprese nella sezione 6.1.3 per mostrare l'impatto dei parametri sul numero di predizioni e sui guadagni.

Numero di previsioni e accuratezza

I grafici in figura 6.1 e 6.2 mostrano il valore medio del numero di predizioni giornaliere estratte dagli algoritmi e l'accuratezza media di tali predizioni. In ogni

[2] Nell'appendice A sono riportati i dettagli del numero di raccomandazioni e guadagni osservati.

[3] Nonostante questa scelta, il processo di simulazione supporta sia classificatori binomiali che polinomiali, come dettagliato nella sezione 5.3.1 con lo pseudo-codice 4).

grafico le predizioni sono rappresentate dalle colonne verdi, mentre l'accuratezza dalla linea arancione.

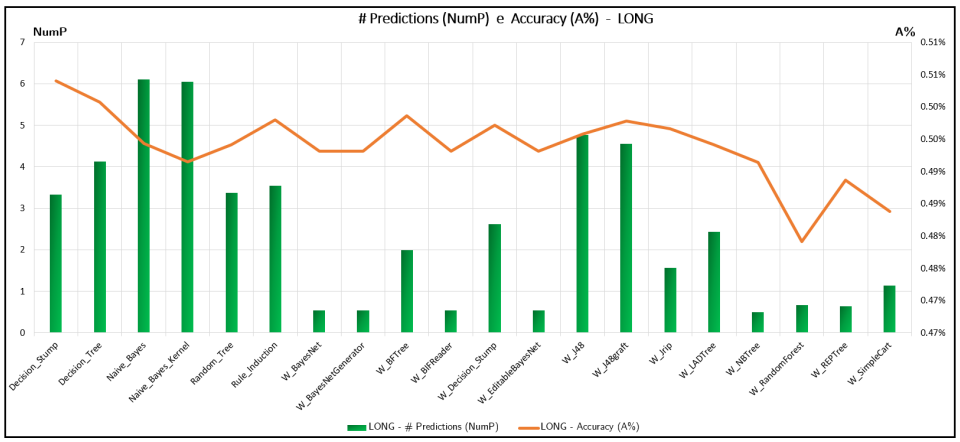

Figura 6.1: Prestazioni algoritmi - Numero previsioni e accuratezza - `LONG`

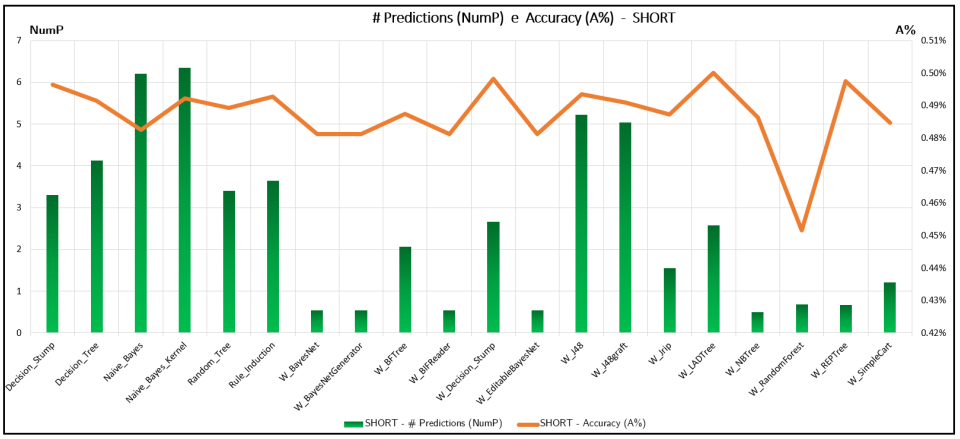

Figura 6.2: Prestazioni algoritmi - Numero previsioni e accuratezza - `SHORT`

Osservando questi grafici si nota che, indipendentemente dal numero di predizioni, la percentuale di predizioni *corrette* è mediamente del 50%. Pertanto, per misurare la qualità di un classificatore, occorre valutare quanto profitto deriva dalle previsioni corrette e quante perdite sono invece causa delle previsioni errate.

La tabella 6.2 mostra per esteso le statistiche riportate nei grafici precedenti, includendo nei dati anche la deviazione standard. Da questa tabella si osserva, prima di tutto, che la deviazione del rate di previsioni corrette è prossima allo zero (quindi l'accuratezza si scosta di poco dal 50%) e poi si nota che circa un terzo dei classificatori (contrassegnati con il simbolo `"*"`) produce un numero basso di raccomandazioni giornaliere: approssimativamente un solo trade `LONG` o `SHORT` al giorno.

Algorithm	Numero predizioni (NumP)				Accuratezza (A%)			
	Mean		StdDev		Mean		StdDev	
	LONG	SHORT	LONG	SHORT	LONG	SHORT	LONG	SHORT
Decision_Stump	3.34	3.31	1.97	2.06	0.50	0.50	0.03	0.03
Decision_Tree	4.12	4.13	2.84	2.90	0.50	0.49	0.03	0.04
Naive_Bayes	6.10	6.20	4.30	4.49	0.49	0.48	0.04	0.06
Naive_Bayes_Kernel	6.04	6.34	4.52	4.81	0.49	0.49	0.04	0.05
Random_Tree	3.37	3.40	2.44	2.47	0.49	0.49	0.03	0.04
Rule_Induction	3.54	3.65	2.34	2.50	0.50	0.49	0.04	0.04
W_BayesNet*	0.54	0.54	0.41	0.42	0.49	0.48	0.05	0.06
W_BayesNetGenerator*	0.54	0.54	0.41	0.42	0.49	0.48	0.05	0.06
W_BFTree	1.99	2.06	1.23	1.34	0.50	0.49	0.05	0.06
W_BIFReader*	0.54	0.54	0.41	0.42	0.49	0.48	0.05	0.06
W_Decision_Stump	2.62	2.66	2.12	2.25	0.50	0.50	0.03	0.04
W_EditableBayesNet*	0.54	0.54	0.41	0.42	0.49	0.48	0.05	0.06
W_J48	4.76	5.23	3.04	3.40	0.50	0.49	0.03	0.04
W_J48graft	4.55	5.03	3.24	3.60	0.50	0.49	0.04	0.05
W_Jrip	1.56	1.54	0.91	0.98	0.50	0.49	0.04	0.06
W_LADTree	2.43	2.58	1.45	1.53	0.49	0.50	0.03	0.04
W_NBTree*	0.50	0.50	0.31	0.32	0.49	0.49	0.05	0.06
W_RandomForest*	0.66	0.68	0.48	0.51	0.48	0.45	0.17	0.18
W_REPTree*	0.63	0.66	0.48	0.51	0.49	0.50	0.12	0.12
W_SimpleCart*	1.14	1.21	0.88	0.94	0.48	0.48	0.11	0.10

Tabella 6.2: Prestazioni algoritmi - Numero di predizioni / Accuratezza

Sulla base di queste informazioni si può dedurre che, mediamente, i classificatori che tendono ad estrarre un alto numero di predizioni potranno generare profitti o perdite relativamente più grandi degli algoritmi che ne estraggono poche, ma, per questi ultimi, sbagliare il 50% delle raccomandazioni ha un peso molto maggiore sulla qualità delle previsioni.

Guadagno medio predizioni e guadagno medio giornaliero

Le deduzioni fatte in merito al numero di predizioni si riflettono nelle metriche di guadagno riportate nei seguenti grafici:

- figure 6.3, 6.4, 6.5 e 6.6: GTAX_{AVG}
  guadagno medio delle predizioni ottenute da ogni algoritmo
  → esprime il *peso* di ogni raccomandazione;
- figure 6.7, 6.8, 6.9 e 6.10: $\text{GTAX}_{\text{DAILY}}$
  guadagno medio giornaliero ottenuto da ogni algoritmo
  → esprime il *peso* dell'insieme di raccomandazioni prodotte ogni giorno.

Il valore di ogni metrica è espresso in termini di:

- media (`Mean`) → colonne grige
- valore minimo (`Min`) → colonne blu
- valore massimo (`Max`) → colonne rosse
- deviazione standard (`StdDev`) → linea gialla

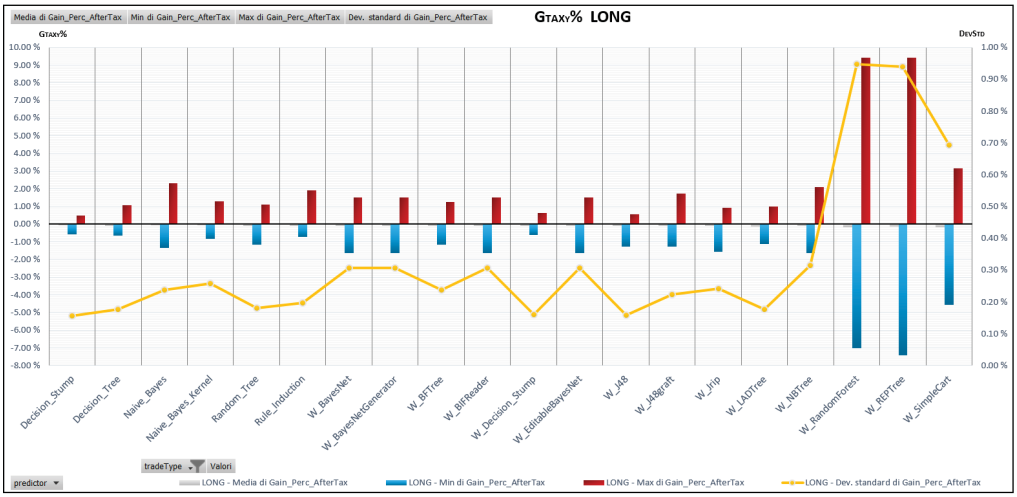

Figura 6.3: Prestazioni algoritmi - GTAX_{AVG} - `LONG`

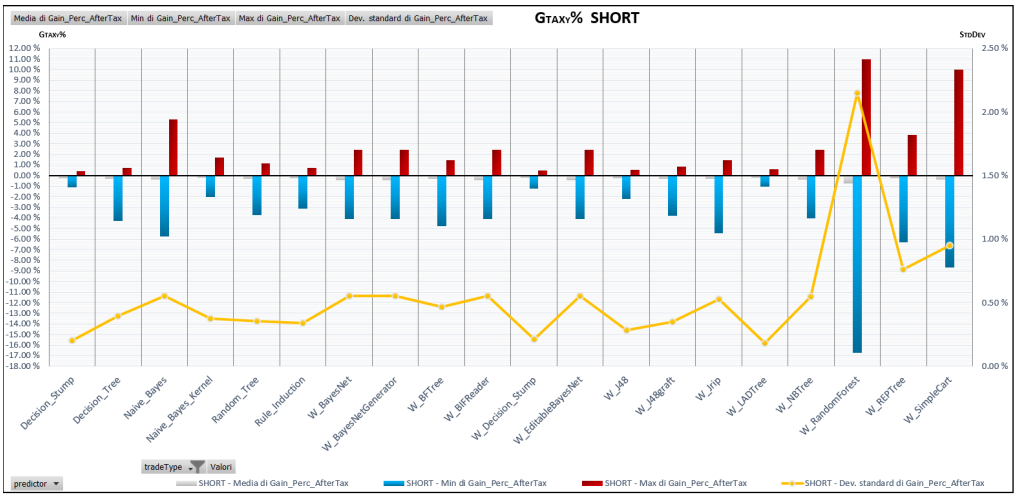

Figura 6.4: Prestazioni algoritmi - GTAX_{AVG} - `SHORT`

I grafici 6.3 e 6.4 permettono subito di visualizzare come il guadagno medio delle predizioni sia nettamente superiore per gli algoritmi che generano poche raccomandazioni: per essi, infatti, il peso di ogni previsione è superiore rispetto a quello dei classificatori che generano più di una raccomandazione al giorno.

Questi dati sono comunque relativi ai valori massimi e minimi misurati su tutte le simulazioni. Se si considera la media calcolata tra tutte le combinazioni di parametri testate, si nota che le predizioni dei classificatori, generalmente, conducono a perdite. I grafici rappresentati nelle figure 6.5 e 6.6, dove si omettono i valori estremi, permettono di constatare che solo alcuni parametri permettono di raggiungere i migliori risultati, mentre, complessivamente, il valore medio calcolato è sempre negativo.

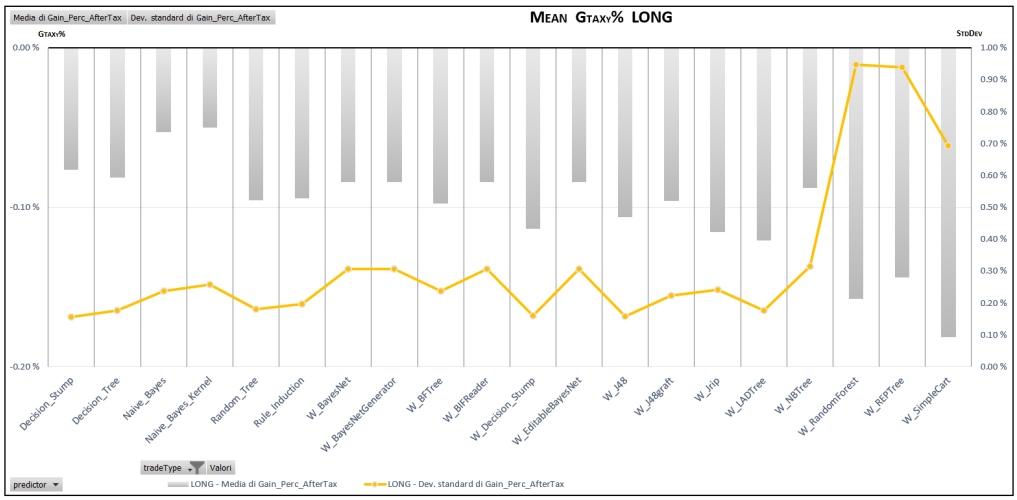

Figura 6.5: Prestazioni algoritmi - Media GTAX_{AVG} - LONG

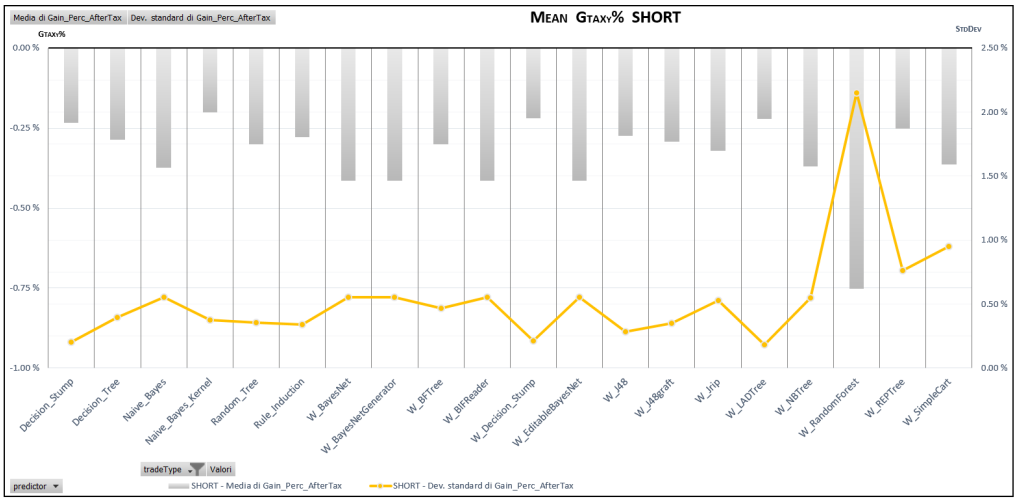

Figura 6.6: Prestazioni algoritmi - Media GTAX_{AVG} - SHORT

Per rimarcare in modo più significativo la differenza tra le prestazioni medie e quelle raggiungibili da un algoritmo con l'appropriata configurazione di parametri, nella tabella 6.3 si riportano i valori più alti (`LONG` o `SHORT`) ottenuti per ogni anno di trading in cui sono avvenute le simulazioni.

Year	`Algorithm`	**Guadagno medio** ($\mathbf{G}_{\mathrm{TAX}_{\mathrm{AVG}}}$)					
		Max		**Mean**		**StdDev**	
		`LONG`	`SHORT`	`LONG`	`SHORT`	`LONG`	`SHORT`
2010	`W_RandomForest`	3.26	3.08	-0.19	-2.07	0.84	3.79
	`W_REPTree`	2.66	1.19	-0.02	-0.38	0.66	0.75
	`W_SimpleCart`	2.21	0.85	-0.06	-0.77	0.59	1.59
2011	`W_RandomForest`	6.83	9.49	0.05	0.15	1.28	1.20
	`Naive_Bayes_Kernel`	1.19	1.49	0.05	0.24	0.37	0.35
	`W_NBTree`	0.89	2.41	-0.11	0.17	0.30	0.43
2012	`W_RandomForest`	6.53	10.97	-0.16	-0.16	0.96	1.66
	`W_REPTree`	1.98	1.19	0.03	-0.38	0.50	0.71
	`W_BayesNet`	1.51	0.17	0.11	-0.46	0.37	0.38
2013	`W_RandomForest`	3.15	6.38	-0.06	-0.52	0.66	1.33
	`W_SimpleCart`	2.24	1.63	0.09	-0.36	0.46	0.46
	`W_REPTree`	1.80	3.69	0.12	-0.20	0.44	0.75
2014	`W_REPTree`	1.14	2.43	-0.38	-0.23	0.79	0.57
	`Naive_Bayes`	1.12	0.84	-0.06	-0.27	0.23	0.30
	`W_SimpleCart`	0.19	10.00	-0.41	-0.12	0.56	1.36
2015	`W_RandomForest`	3.59	0.15	-0.21	-1.46	0.76	1.93
	`W_SimpleCart`	2.49	0.57	0.02	-0.48	0.43	0.75
	`W_REPTree`	2.10	3.06	0.05	-0.34	0.55	0.85
2016	`W_RandomForest`	3.79	2.13	-0.26	-1.03	1.01	2.15
	`W_REPTree`	1.53	0.37	-0.72	-0.42	1.33	0.47
	`W_NBTree`	0.47	1.96	-0.21	-0.29	0.28	0.36
2017	`W_RandomForest`	9.44	2.03	-0.24	-0.24	1.09	0.54
	`W_REPTree`	9.44	3.82	0.14	-0.18	1.61	1.15
	`Naive_Bayes`	2.33	5.30	-0.05	-0.21	0.32	0.69

Tabella 6.3: Massimi guadagni percentuali

Considerando, invece, le metriche relative al guadagno giornaliero ($\text{GTAX}_{\text{DAILY}}$) si può notare, come atteso, che gli algoritmi caratterizzati dal basso numero di raccomandazioni generano profitti e perdite limitate, mentre quelli che estraggono più di una predizione al giorno raggiungono dei massimi e dei minimi molto più elevati. Dai grafici in figura 6.7 e 6.8 si nota soprattutto che sono le perdite a toccare i valori più alti.

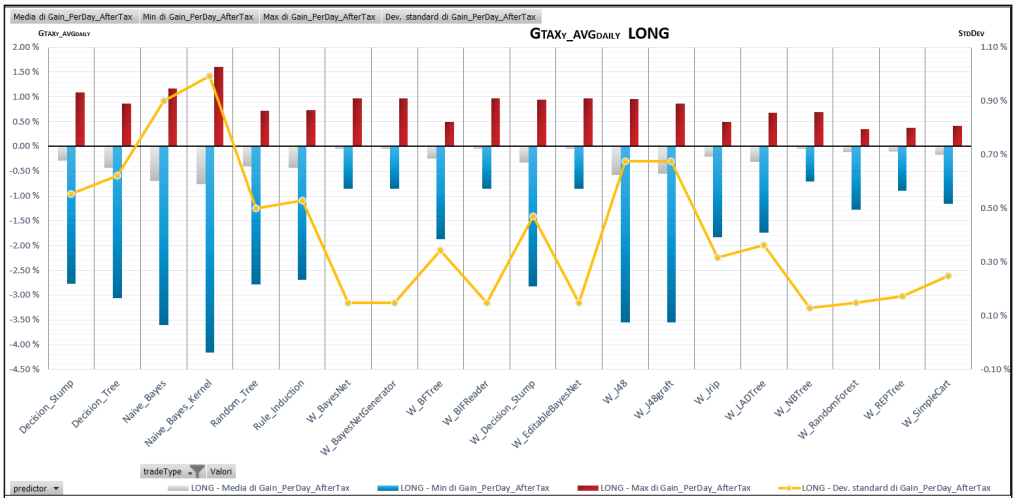

Figura 6.7: Prestazioni algoritmi - $\text{GTAX}_{\text{DAILY}}$ - LONG

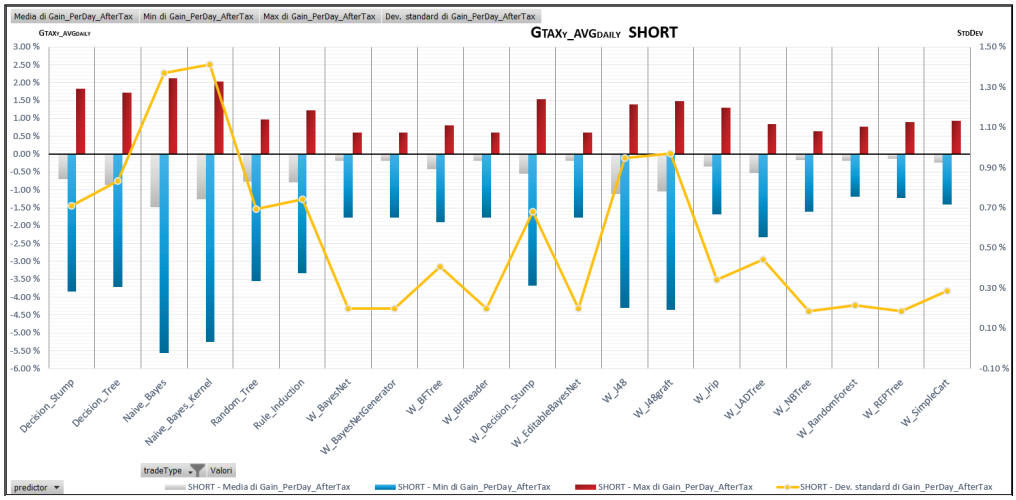

Figura 6.8: Prestazioni algoritmi - $\text{GTAX}_{\text{DAILY}}$ - SHORT

I guadagni giornalieri rappresentati nei grafici che considerano solo la media e la deviazione standard (fig. 6.9 e 6.10) mostrano un comportamento più lineare con quello osservato nei grafici precedenti, in quanto la metrica è calcolata già sul valore medio dei ritorni giornalieri. L'aggregazione sugli anni e sulle combinazioni di parametri nasconde il fatto che si ottengano ritorni positivi, ma non modifica più di tanto l'andamento delle perdite.

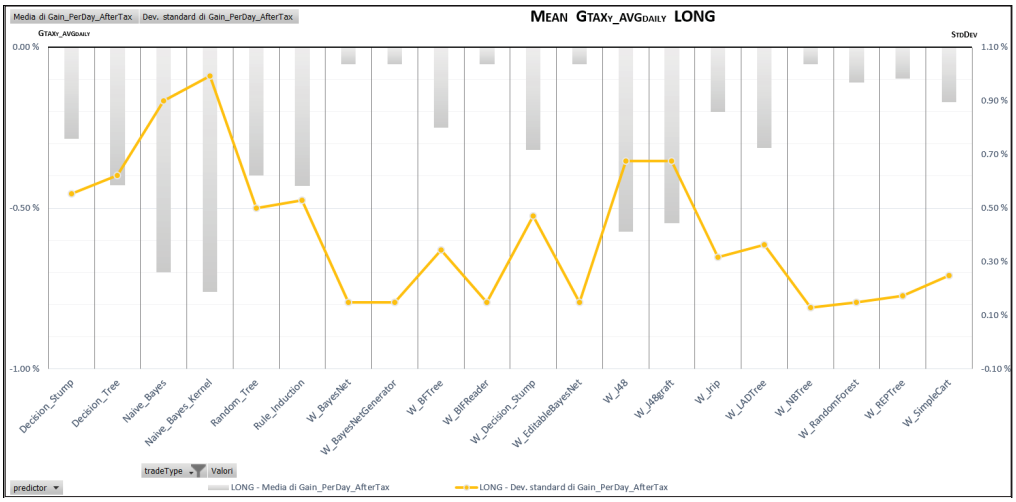

Figura 6.9: Prestazioni algoritmi - Media $\text{GTAX}_{\text{DAILY}}$ - `LONG`

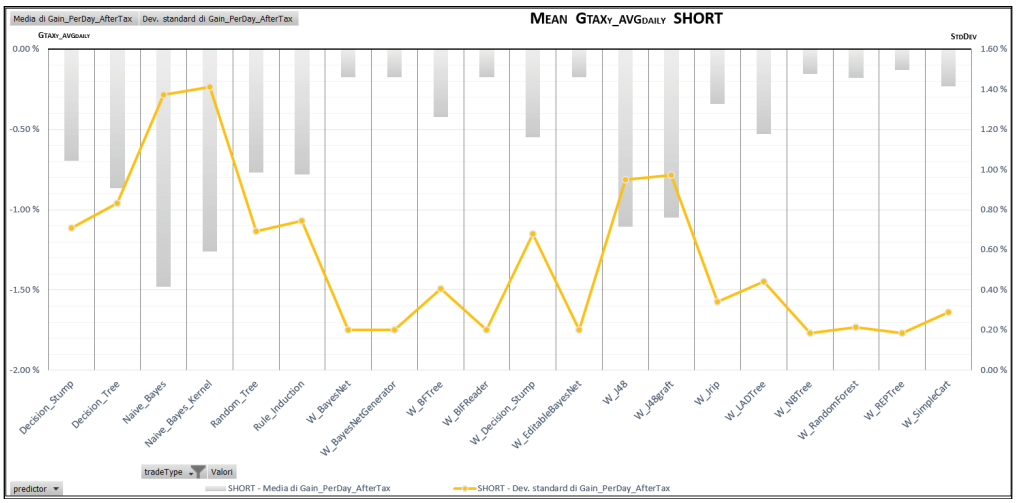

Figura 6.10: Prestazioni algoritmi - Media $\text{GTAX}_{\text{DAILY}}$ - `SHORT`

Come fatto per il guadagno medio delle predizioni, anche per la metrica del rendimento medio giornaliero si elencano i classificatori che, complessivamente, sono risultati più performanti. La tabella 6.4 riassume gli algoritmi che hanno determinato le migliori scelte `LONG` o `SHORT` nel corso degli anni simulati.

Year	Algorithm	Guadagno giornaliero ($G_{TAX_{DAILY}}$)					
		Max		Mean		StdDev	
		LONG	SHORT	LONG	SHORT	LONG	SHORT
2010	Decision_Stump	0.40	-0.08	-0.31	-0.80	0.39	0.40
	W_Decision_Stump	0.34	0.08	-0.30	-0.62	0.42	0.40
	Decision_Tree	0.29	0.03	-0.49	-0.99	0.50	0.45
	W_LADTree	0.16	0.23	-0.46	-0.46	0.37	0.35
	Naive_Bayes_Kernel	0.09	0.19	-1.14	-0.91	0.91	0.83
2011	Naive_Bayes_Kernel	1.60	2.04	-0.60	0.71	1.23	0.62
	Naive_Bayes	1.17	2.13	-0.57	0.14	1.13	0.75
	W_BayesNet	0.97	0.60	-0.08	0.07	0.18	0.16
	Decision_Stump	0.66	1.84	-0.41	0.41	0.60	0.46
	Decision_Tree	0.53	1.72	-0.79	0.34	0.75	0.48
2012	Decision_Stump	0.87	-0.14	-0.12	-0.88	0.39	0.55
	W_Decision_Stump	0.84	0.22	-0.16	-0.60	0.36	0.56
	Naive_Bayes_Kernel	0.71	0.72	-0.42	-0.97	0.69	0.82
	W_LADTree	0.42	0.60	-0.28	-0.45	0.29	0.31
	W_RandomForest	0.24	0.29	-0.03	-0.12	0.12	0.18
2013	Decision_Stump	0.99	0.29	0.23	-0.88	0.32	0.83
	W_J48	0.96	0.17	0.01	-1.62	0.35	1.12
	W_Decision_Stump	0.88	0.34	0.02	-0.75	0.29	0.76
	Decision_Tree	0.87	0.19	0.16	-1.19	0.28	0.98
	W_BFTree	0.44	0.18	0.01	-0.61	0.18	0.46
2014	W_LADTree	0.36	0.84	-0.35	-0.39	0.32	0.29
	Rule_Induction	0.32	0.16	-0.55	-0.61	0.43	0.50
	W_Decision_Stump	0.27	0.33	-0.52	-0.47	0.45	0.45
	Naive_Bayes	0.17	0.30	-0.92	-1.16	0.81	0.83
	W_BFTree	0.16	0.37	-0.41	-0.28	0.31	0.24
2015	Decision_Stump	1.08	-0.19	0.09	-1.20	0.30	0.62
	W_Decision_Stump	0.94	0.03	-0.09	-0.85	0.30	0.70
	Naive_Bayes_Kernel	0.82	0.09	-0.55	-1.60	0.97	1.32
	W_NBTree	0.61	0.03	0.04	-0.29	0.10	0.20
	W_REPTree	0.30	0.03	-0.02	-0.20	0.10	0.17
2016	Naive_Bayes_Kernel	0.77	-0.10	-1.14	-2.03	1.25	1.11
	Naive_Bayes	0.54	-0.20	-1.00	-1.73	1.03	0.82
	W_BFTree	0.49	0.08	-0.47	-0.51	0.40	0.31
	W_LADTree	0.24	0.30	-0.45	-0.65	0.32	0.41
	W_BayesNet	0.23	0.23	-0.12	-0.20	0.19	0.16
2017	Rule_Induction	0.26	0.00	-0.37	-0.88	0.34	0.70
	Naive_Bayes_Kernel	0.17	0.15	-0.79	-1.65	0.75	1.47
	Decision_Stump	0.17	0.00	-0.19	-0.75	0.21	0.55
	W_LADTree	0.12	0.19	-0.26	-0.62	0.27	0.50
	Naive_Bayes	0.06	0.09	-0.88	-1.70	0.79	1.47

Tabella 6.4: Massimi guadagni giornalieri

6.1.3 Osservazioni sulla variazione dei parametri

Dopo aver osservato le prestazioni *globali* dei classificatori simulati, si approfondisce ora l'impatto dei parametri utilizzati sul numero di predizioni (`NumP`), sul guadagno medio (GTAX_{AVG}) e su quello giornaliero ($\text{GTAX}_{\text{DAILY}}$). L'accuratezza non viene più considerata in quanto è già stato constatato che, in questo caso, non è una metrica rilevante per la valutazione delle prestazioni.

I parametri a cui si fa riferimento in questa sezione sono:

- `TrainingDays`
- `SlidingWindow`
- `LabelDiscretization`
- `FixedCutPoint`
- `TradeType`
- `StopLossPerc`

`TrainingDays` e `SlidingWindow` sono analizzati congiuntamente, perché entrambi coinvolgono la dimensione del tempo e la sua rappresentazione per addestrare i modelli di classificazione. Analogamente anche `LabelDiscretization` e `FixedCutPoint` sono considerati insieme, poiché determinano la modalità di discretizzazione dell'attributo di classe. Mentre il valore del parametro `TradeType` viene usato per dividere tutti i risultati prodotti in base alla decisione di investimento `LONG` o `SHORT`.

L'impatto del parametro `StopLossPerc` viene analizzato separatamente, perché lo stop loss non ha alcun legame con i modelli di classificazione, ma definisce una strategia di uscita dal mercato da applicare nel corso della giornata di trading, ovvero successivamente alla raccomandazione delle azioni su cui investire.

Numero di previsioni

Procedendo quindi con i primi parametri indicati, nei grafici in figura 6.11 e 6.12 si mostra l'impatto delle variabili relative al tempo e alla discretizzazione sul numero di predizioni mediamente estratte dai classificatori. Le colonne verdi rappresentano il numero di predizioni e la linea arancione la deviazione standard.

Nella figura 6.11 si nota che all'aumentare della dimensione del training set (`TrainingDays`) la media delle previsioni `NumP` diminuisce lievemente, mentre all'aumentare della `SlidingWindow` i classificatori estraggono mediamente più previsioni. In entrambi i casi si tratta di una variazione piuttosto piccola: meno di 0.5 raccomandazioni.

Nella figura 6.12 la variazione è invece molto più significativa, sia per la discretizzazione in 2 classi (a sinistra) che in 4 classi (a destra). Nel caso della discretizzazione binomiale si può notare che alzare il valore assoluto del cut point riduce il numero di previsioni: ciò dimostra che si riesce ad indirizzare la classificazione verso le *poche* vere variazioni rilevanti del prezzo delle azioni. Invece, nel caso della discretizzazione su più classi, si nota che il numero di previsioni diminuisce fino al

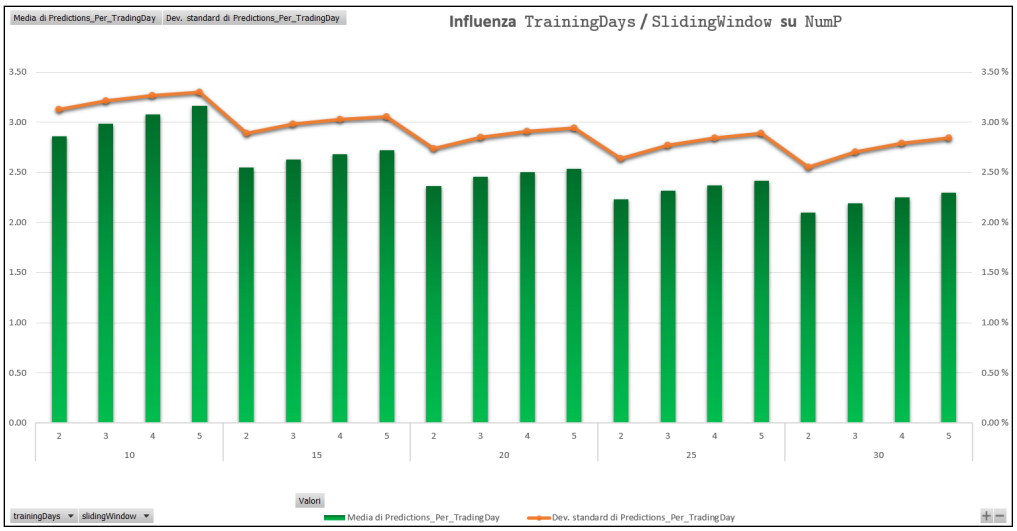

Figura 6.11: Influenza `TrainingDays` / `SlidingWindow` su `NumP`

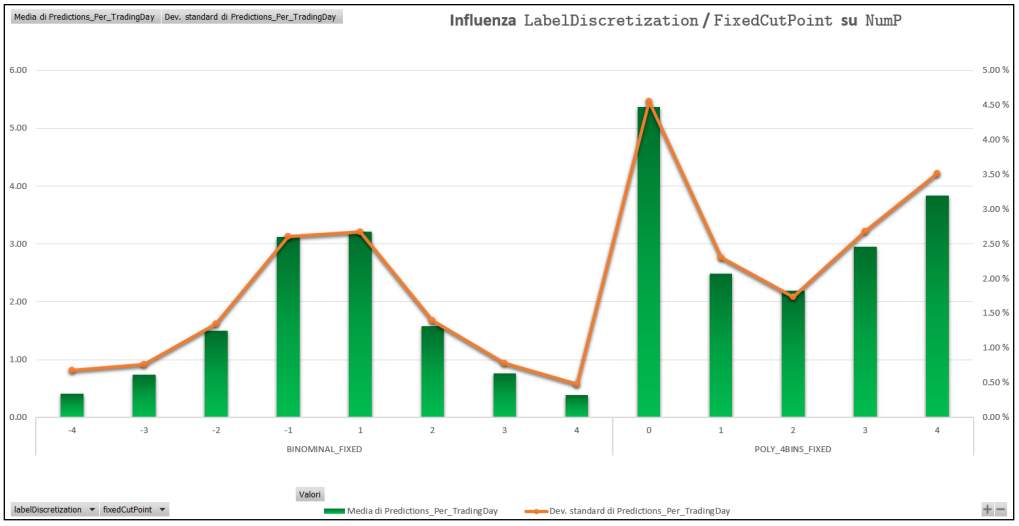

Figura 6.12: Influenza `LabelDiscretization` / `FixedCutPoint` su `NumP`

`FixedCutPoint` = 2, per poi risalire. Ciò significa che impostando il cut point a 1 o a 2, si riesce a filtrare il numero di raccomandazioni scegliendo la classe che esprime la massima variazione di prezzo, mentre, aumentando il valore del cut point, tali classi non riescono più ad essere previste e la strategia di selezione degli stock (vedi 5.2.2) ricade sulle sole classi rimaste, che esprimono semplicemente le variazioni positive o negative.

Guadagno medio previsioni e guadagno medio giornaliero

Nelle tabelle 6.5 e 6.6 si riportano i valori medi e massimi di guadagno medio delle predizioni (GTAX_{AVG}) e del guadagno medio giornaliero ($\text{GTAX}_{\text{DAILY}}$).

Training Days	Sliding Window	$G_{TAX_{AVG}}$				$G_{TAX_{DAILY}}$			
		Mean		**Max**		**Mean**		**Max**	
		LONG	SHORT	LONG	SHORT	LONG	SHORT	LONG	SHORT
10	2	-0.14	-0.33	1.95	2.41	-0.37	-0.62	1.60	1.78
	3	-0.12	-0.32	6.83	3.69	-0.37	-0.68	1.43	1.98
	4	-0.11	-0.35	3.72	5.13	-0.37	-0.70	0.87	1.98
	5	-0.10	-0.35	2.48	3.69	-0.36	-0.71	0.99	1.97
15	2	-0.10	-0.30	4.75	10.00	-0.28	-0.52	1.13	1.87
	3	-0.09	-0.30	3.59	10.00	-0.28	-0.55	1.17	2.01
	4	-0.09	-0.34	3.17	3.06	-0.31	-0.57	0.85	2.13
	5	-0.09	-0.32	6.92	5.30	-0.32	-0.59	0.91	1.79
20	2	-0.13	-0.37	3.26	1.17	-0.28	-0.52	1.17	1.87
	3	-0.09	-0.33	2.46	3.82	-0.26	-0.55	1.22	1.79
	4	-0.09	-0.33	3.35	3.82	-0.26	-0.56	1.10	2.04
	5	-0.09	-0.33	2.61	3.82	-0.28	-0.57	0.97	1.69
25	2	-0.13	-0.32	3.17	3.08	-0.28	-0.51	0.81	0.84
	3	-0.07	-0.31	5.16	6.38	-0.27	-0.53	0.75	1.61
	4	-0.08	-0.34	5.49	3.82	-0.26	-0.54	0.88	1.81
	5	-0.09	-0.34	4.56	4.11	-0.28	-0.55	0.87	1.14
30	2	-0.12	-0.34	9.44	2.43	-0.26	-0.48	0.74	1.06
	3	-0.09	-0.34	9.44	4.52	-0.25	-0.50	0.85	1.61
	4	-0.08	-0.37	9.44	1.44	-0.25	-0.52	0.85	1.64
	5	-0.09	-0.32	4.80	10.97	-0.27	-0.53	0.71	1.44

Tabella 6.5: Influenza `TrainingDays` / `SlidingWindow` sui guadagni

Label Discretization	Fixed CutPoint	$G_{TAX_{AVG}}$				$G_{TAX_{DAILY}}$			
		Mean		**Max**		**Mean**		**Max**	
		LONG	SHORT	LONG	SHORT	LONG	SHORT	LONG	SHORT
BINOMINAL_FIXED	-4		-0.67		10.97		-0.15		0.95
	-3		-0.56		4.52		-0.25		1.43
	-2		-0.41		3.58		-0.42		1.87
	-1		-0.29		1.14		-0.73		1.78
	1	-0.10		1.47		-0.32		1.18	
	2	-0.06		5.16		-0.08		1.60	
	3	-0.06		6.53		-0.02		1.33	
	4	0.01		9.44		-0.01		0.99	
POLY_4BINS_FIXED	0	-0.14	-0.21	0.52	0.46	-0.67	-1.09	1.08	1.87
	1	-0.15	-0.23	0.35	0.80	-0.30	-0.54	0.76	2.13
	2	-0.11	-0.22	0.63	0.54	-0.23	-0.47	0.77	1.98
	3	-0.15	-0.21	0.38	0.68	-0.42	-0.63	0.70	1.17
	4	-0.15	-0.21	0.45	0.46	-0.58	-0.78	0.74	1.34

Tabella 6.6: Influenza `LabelDiscretization` / `FixedCutPoint` sui guadagni

Per `TrainingDays` / `SlidingWindow` la media dei guadagni su tutte le simulazioni non è mai positiva. Si possono osservare i più alti guadagni per predizione, sia `LONG` che `SHORT`, solo su training set composti dagli esempi degli ultimi 30 giorni. Mentre i più alti rendimenti giornalieri sono stati ottenuti, per gli investimenti `LONG`, dai classificatori addestrati sugli 10 giorni con una finestra sui 2 giorni passati, mentre, nel caso `SHORT`, da quelli addestrati su 15 esempi, osservando attraverso la finestra sui 4 giorni precedenti.

Per i parametri relativi alla discretizzazione dell'attributo di classe si ha un solo guadagno medio positivo (0.01), nel caso della discretizzazione binomiale con `FixedCutPoint` = 4. Questo valore, tuttavia, ha poca rilevanza statistica poiché, come già osservato nel grafico in figura 6.12, all'aumentare del valore del cut point, la classificazione binomiale produce meno predizioni e, di conseguenza, la media di guadagno calcolata *sembra* esprimere un peso più importante. Passando quindi ai guadagni giornalieri, i rendimenti migliori sono stati ottenuti dalle combinazioni `LabelDiscretization` = `BINOMINAL_FIXED` con `FixedCutPoint` = |2| e `LabelDiscretization` = `POLY_4BINS_FIXED` con `FixedCutPoint` = 1.

Impatto dei parametri su una simulazione di riferimento

Per rappresentare in modo più concreto l'impatto della configurazione di parametri sui guadagni, invece di considerare l'insieme di tutte le simulazioni svolte, di seguito si prende come riferimento una delle combinazioni $\{algoritmo, \{parametri\}\}$ che ha generato il massimo rendimento nel corso di uno degli anni di trading simulati. A partire da tale esempio, quindi, si mostra come variano i guadagni se si cambia la configurazione iniziale dei parametri:

Year	2011
`Algorithm`	`Naive_Bayes_Kernel`
`TrainingDays`	10
`SlidingWindow`	2
`LabelDiscretization`	`BINOMINAL_FIXED`
`FixedCutPoint`	2
`TradeType`	`LONG`

Con questa combinazione di parametri, l'algoritmo `Naive_Bayes_Kernel` ha conseguito un guadagno medio giornaliero ($\text{GTAX}_{\text{DAILY}}$) pari a 1.60%, simulando investimenti `LONG` sui dati storici del 2011.

I grafici riportati nelle figure 6.13 e 6.14 mostrano come cambiano i guadagni modificando solamente la coppia di parametri relativi all'intervallo di tempo osservato e la coppia relativa alla discretizzazione dell'attributo di classe.

Nella figura 6.13 si può notare che all'aumentare del numero di esempi del training set e dell'ampiezza della finestra temporale i guadagni decrescono fino a scendere al di sotto dello zero. Mentre, nella figura 6.14, si osserva la netta superiorità della

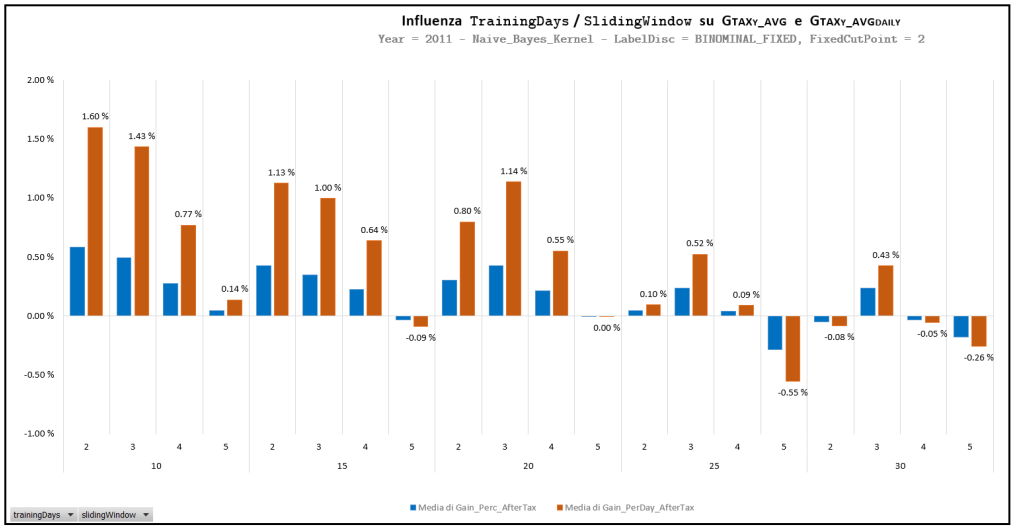

Figura 6.13: Influenza `TrainingDays` / `SlidingWindow` su una simulazione

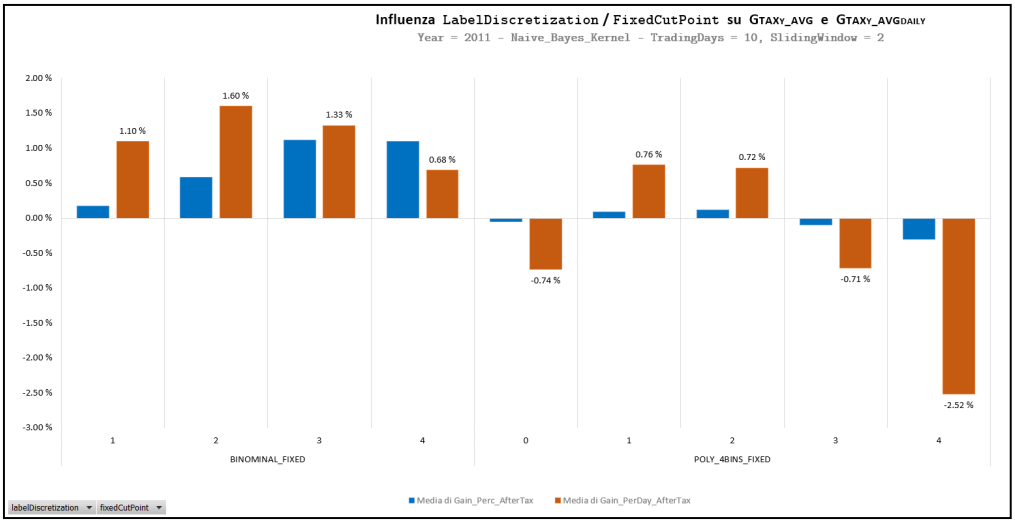

Figura 6.14: Influenza `LabelDiscretization` / `FixedCutPoint` su una simulazione

classificazione binomiale rispetto a quella basata su 4 classi: dove i cut point centrali (2 e 3) determinano le migliori prestazioni dell'algoritmo utilizzato.

A seguito di queste osservazioni si può affermare che la combinazione di parametri può influire in modo significativo sulle prestazioni di un modello di classificazione, ma è complesso determinare una regola universale che possa essere applicata in modo sistematico. Questa complessità viene affrontata delegando l'analisi delle metriche di performance all'algoritmo di regressione descritto nella sezione 6.2.

Stop Loss

L'ultimo parametro da considerare, prima di analizzare i risultati prodotti dal modello predittivo finale, è lo stop loss. Come accennato all'inizio di questa sezione il parametro `StopLossPerc` non è collegato con le prestazioni di classificatori, ma introduce una strategia di ottimizzazione dei guadagni, applicata solo a seguito dell'estrazione delle raccomandazioni.

I guadagni misurati applicando la strategia di stop loss sono rappresentati dalla metrica $\texttt{GSL}_{\text{DAILY}}$, che esprime il guadagno medio giornaliero ($\texttt{GTAX}_{\text{DAILY}}$) calcolato in funzione del parametro `StopLossPerc`.

Nei grafici in figura 6.15 e 6.16 si mettono a confronto i guadagni conseguiti con e senza stop loss, usando la seguente legenda:

- media di $\texttt{GTAX}_{\text{DAILY}}$ (senza stop loss) → colonna grigia
- massimo di $\texttt{GTAX}_{\text{DAILY}}$ (senza stop loss) → colonna rosso chiaro
- $\texttt{GSL}_{\text{DAILY}}$ con `StopLossPerc` = 1.0% → colonne blu
- $\texttt{GSL}_{\text{DAILY}}$ con `StopLossPerc` = 1.5% → colonne gialle
- $\texttt{GSL}_{\text{DAILY}}$ con `StopLossPerc` = 2.0% → colonne rosse

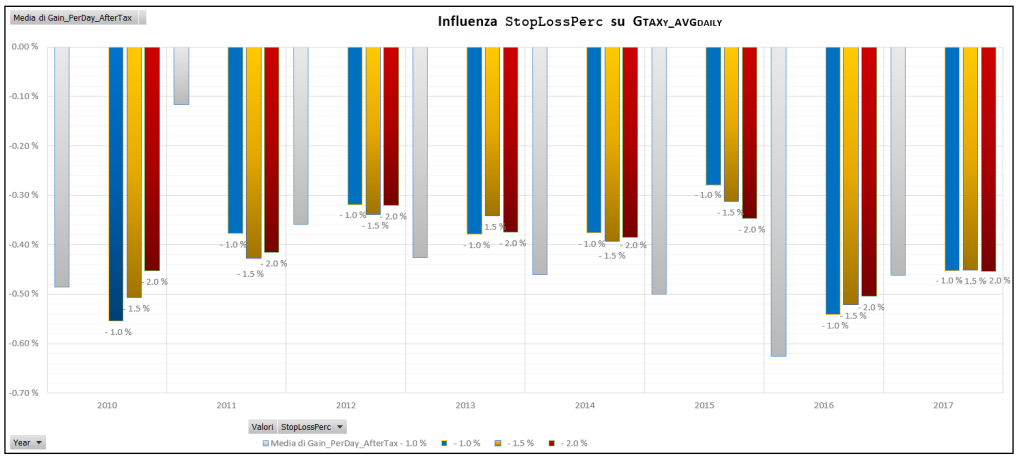

Figura 6.15: Influenza `StopLossPerc` sulla media dei guadagni giornalieri

Nella figura 6.15 si osserva che, mediamente, la strategia di stop loss ha permesso di minimizzare le perdite. Infatti, ad eccezione degli anni 2010 e 2011, i valori di guadagno, anche se negativi, sono più alti.

Ridurre le perdite, però, non permette di incrementare i valori massimi di guadagno, infatti, nel grafico 6.16 si nota che, mentre i guadagni massimi raggiungono le misure osservate negli esempi precedenti, i guadagni ottenuti applicando lo stop loss sono inferiori: ciò significa che, nelle condizioni in cui i classificatori hanno conseguito il massimo rendimento, lo stop loss ha causato l'uscita dal mercato prima che le azioni potessero raggiungere il prezzo di chiusura (che invece avrebbe condotto ad un guadagno superiore). Per maggiore chiarezza si riportano gli stessi dati nelle tabelle 6.7 e 6.8.

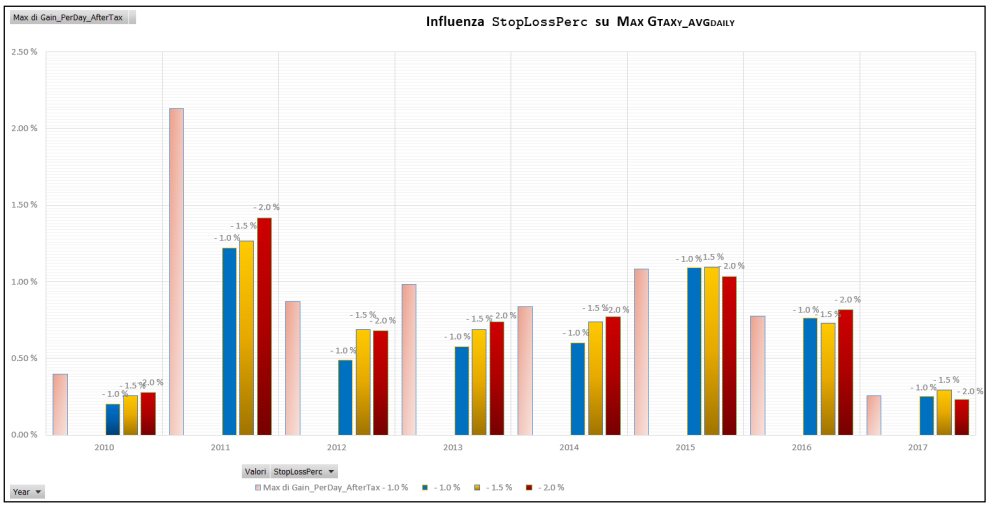

Figura 6.16: Influenza `StopLossPerc` sui guadagni giornalieri massimi

			Mean GSL_{DAILY} con StopLossPerc					
	No StopLoss		1.0%		1.5%		2.0%	
Year	LONG	SHORT	LONG	SHORT	LONG	SHORT	LONG	SHORT
2010	-0.40	-0.57	-0.48	-0.62	-0.47	-0.54	-0.43	-0.48
2011	-0.42	0.18	-0.57	-0.18	-0.67	-0.19	-0.67	-0.16
2012	-0.12	-0.60	-0.13	-0.51	-0.15	-0.53	-0.14	-0.50
2013	-0.01	-0.85	-0.10	-0.66	-0.06	-0.63	-0.07	-0.68
2014	-0.45	-0.47	-0.32	-0.43	-0.39	-0.39	-0.42	-0.35
2015	-0.11	-0.89	0.00	-0.56	-0.03	-0.60	-0.07	-0.62
2016	-0.56	-0.69	-0.51	-0.57	-0.51	-0.53	-0.49	-0.52
2017	-0.28	-0.65	-0.28	-0.63	-0.27	-0.63	-0.29	-0.62

Tabella 6.7: Influenza StopLoss sui guadagni medi giornalieri

			Max GSL_{DAILY} con StopLossPerc					
	No StopLoss		1.0%		1.5%		2.0%	
Year	LONG	SHORT	LONG	SHORT	LONG	SHORT	LONG	SHORT
2010	0.40	0.23	0.20	0.12	0.26	0.16	0.28	0.17
2011	1.60	2.13	0.35	1.22	0.42	1.27	0.56	1.42
2012	0.87	0.72	0.49	0.22	0.69	0.28	0.68	0.33
2013	0.99	0.34	0.58	0.25	0.69	0.30	0.74	0.36
2014	0.36	0.84	0.29	0.60	0.28	0.74	0.32	0.77
2015	1.08	0.09	1.09	0.22	1.10	0.16	1.04	0.17
2016	0.77	0.30	0.76	0.19	0.73	0.30	0.82	0.29
2017	0.26	0.19	0.25	0.11	0.29	0.15	0.23	0.17

Tabella 6.8: Influenza StopLoss sui guadagni massimi

Per rappresentare in modo più concreto l'impatto dello stop loss sui guadagni, come fatto per i parametri precedenti, si ripropone l'esempio della simulazione che ha ottenuto il massimo rendimento medio giornaliero nel grafico in figura 6.17.

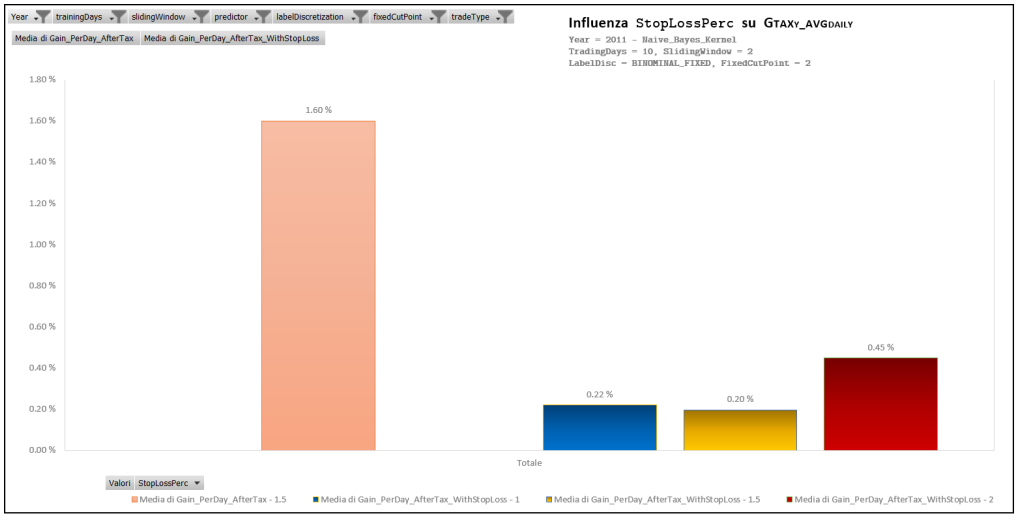

Figura 6.17: Influenza `StopLossPerc` su una simulazione

Anche osservando questo grafico risulta evidente che, nel corso dell'anno simulato, lo stop loss abbia più volte chiuso le posizioni prima che i prezzi potessero raggiungere un prezzo di chiusura molto maggiore rispetto a quello della giornata precedente.

6.2 Self-tuning dei classificatori

Conclusa la descrizione sui risultati ottenuti dalle simulazioni dei metodi di classificazione, in questa sezione ci si concentra sulla parte di self-tuning. Come descritto nella sezione 5.4, i dati generati dalle simulazioni sono successivamente impiegati per addestrare un modello di regressione a prevedere i potenziali guadagni ottenibili da ogni combinazione $\{algoritmo,\ \{parametri\}\}$, in modo tale da poter scegliere dinamicamente quale configurazione usare per raccomandare gli stock nel trimestre successivo. Di seguito si descrivono brevemente le caratteristiche dell'algoritmo di regressione scelto (6.2.1), si mostrano i risultati ottenuti applicando il self-tuning (6.2.2) e si confrontano sia con i singoli classificatori (6.2.3), sia con i risultati ottenuti dal sistema BeSt originale (6.2.4).

6.2.1 Modello di regressione

Come anticipato nella sezione 5.4 del capitolo precedente, inizialmente il modello proposto per la fase di self-tuning era un albero decisionale, ma poi è stato scelto

il modello di regressione, poiché è risultato più adatto al problema di data mining da risolvere. Per giustificare tale decisione e spiegare i limiti della classificazione in questo scenario, occorre considerare lo sbilanciamento nella distribuzione dei dati relativi ai guadagni delle simulazioni:

	Numero di simulazioni		
Metrica	$> 0\%$	$> 0.5\%$	$> 1\%$
GTAX_{AVG}	24.41%	2.50%	0.80%
$\text{GTAX}_{\text{DAILY}}$	24.41%	1.59%	0.31%

Tabella 6.9: Raggruppamento simulazioni 2010-2017 per guadagno

Come si può osservare nella tabella 6.9, dove si raggruppano le metriche di guadagno di tutte le simulazioni eseguite sui dati dal 2010 al 2017, si può notare che la maggior parte dei risultati ottenuti è costituita da perdite e che pochissime configurazioni di classificatori sono quelle più promettenti: ovvero quelle che si vorrebbero estrarre nella fase di self-tuning.

Sebbene alcuni algoritmi di tipo decision tree riuscissero a classificare correttamente diverse simulazioni, la quantità di esempi appartenenti alla classe di maggioranza è risultata abbastanza elevata da impattare negativamente sulla *precisione* e sul *richiamo* (rif. 2.2.1) della classe associata ai classificatori proficui.

Cambiando strategia a favore della regressione, il problema della vasta quantità di simulazioni con guadagni negativi persiste, ma risulta molto più semplice scegliere la configurazione di un classificatore, limitandosi ad estrarre l'esempio per cui è stato previsto il guadagno più alto degli altri. Inoltre, ai fini del self-tuning, non è nemmeno richiesto prevedere il valore esatto dei guadagni futuri, ma è sufficiente stimare quali configurazione renderà più di tutte le altre.

Quindi, in definitiva, il modello predittivo usato nella fase di self-tuning si basa su un algoritmo di regressione polinomiale configurato con i seguenti parametri:

`Max iterations`	1000
`Replication factor`	1
`Max degree`	5
`Min coefficient`	-10
`Max coefficient`	10

Il parametro `max iterations` pone un limite al numero massimo di iterazioni eseguite dall'algoritmo per determinare la funzione $y = f(x)$ con cui esprimere il guadagno giornaliero in relazione agli altri attributi del training set. `Replication factor` specifica che ogni attributo dovrà essere incluso una sola volta nella funzione $f(x)$, mentre `max degree` indica all'algoritmo che dovrà costruire un polinomio di

quinto grado. Infine, `min` e `max coefficient` specificano il range di valori da usare come offset e come coefficienti dei termini dell'equazione.

Sperimentalmente, questa configurazione di parametri è quella che, applicata sulle metriche aggregate per trimestre, senza considerare le variabili relative allo stop loss, ha permesso di ottenere le performance migliori. Tutti i risultati presentati di seguito sono quelli conseguiti impiegando questo modello.

6.2.2 Prestazioni dei classificatori scelti

Applicando il modello di regressione, secondo le specifiche indicate nella sezione 5.4, sono stati scelti i classificatori e i relativi parametri per ogni trimestre dal 2011 al 2017 (i 4 trimestri dell'anno 2010 sono stati usati esclusivamente come dati di training). La tabella 6.10, riporta l'elenco completo dei metodi scelti automaticamente:

Quarter	Algorithm	Training Days	Sliding Window	Label Disc.	Cut Point	Trade Type	$\mathbf{G}_{\mathrm{TAX}_{\mathrm{DAILY}}}$
2011-1	W_SimpleCart	10	5	POLY_4BINS	0.0	LONG	0.12
2011-2	W_SimpleCart	30	2	POLY_4BINS	4.0	SHORT	0.28
2011-3	Decision_Stump	10	5	POLY_4BINS	1.0	LONG	2.99
2011-4	W_SimpleCart	10	5	BINOMINAL	-2.0	SHORT	0.28
2012-1	Decision_Tree	10	2	POLY_4BINS	4.0	LONG	0.35
2012-2	W_SimpleCart	10	2	POLY_4BINS	0.0	SHORT	0.27
2012-3	Decision_Stump	10	2	POLY_4BINS	3.0	LONG	0.86
2012-4	W_SimpleCart	30	5	BINOMINAL	-2.0	SHORT	0.19
2013-1	Decision_Stump	10	5	BINOMINAL	4.0	LONG	1.08
2013-2	W_SimpleCart	10	2	POLY_4BINS	4.0	LONG	0.14
2013-3	Decision_Stump	10	2	BINOMINAL	-4.0	SHORT	-0.35
2013-4	Decision_Stump	30	2	BINOMINAL	4.0	LONG	0.41
2014-1	W_SimpleCart	30	5	POLY_4BINS	4.0	LONG	0.34
2014-2	W_SimpleCart	30	5	POLY_4BINS	1.0	SHORT	0.28
2014-3	Decision_Stump	30	5	BINOMINAL	4.0	LONG	0.10
2014-4	W_RandomForest	10	5	POLY_4BINS	4.0	SHORT	0.13
2015-1	W_SimpleCart	10	2	POLY_4BINS	0.0	LONG	1.05
2015-2	W_SimpleCart	30	5	BINOMINAL	-1.0	SHORT	-0.25
2015-3	W_LADTree	10	5	BINOMINAL	-4.0	SHORT	0.56
2015-4	W_SimpleCart	30	5	POLY_4BINS	3.0	LONG	0.06
2016-1	W_SimpleCart	30	2	BINOMINAL	-1.0	SHORT	0.17
2016-2	Decision_Tree	10	2	POLY_4BINS	4.0	SHORT	-0.64
2016-3	W_SimpleCart	10	2	BINOMINAL	1.0	LONG	0.60
2016-4	Decision_Stump	30	2	BINOMINAL	4.0	LONG	1.57
2017-1	Decision_Tree	30	2	POLY_4BINS	4.0	LONG	0.60
2017-2	Decision_Tree	10	5	BINOMINAL	-3.0	SHORT	0.03
2017-3	W_RandomForest	10	2	POLY_4BINS	4.0	LONG	-0.01
2017-4	Decision_Stump	30	2	BINOMINAL	4.0	LONG	0.14

Tabella 6.10: Classificatori scelti automaticamente per trimestre

Nella tabella 6.11, si riportano gli stessi classificatori, specificando solamente l'algoritmo e il tipo di investimento a cui si rivolgono, e si includono i dettagli dei risultati ottenuti per ogni trimestre.

I dettagli considerati includono: il numero di raccomandazioni (NumP), il guadagno giornaliero ($\text{GTAX}_{\text{DAILY}}$) e media, massimo e deviazione standard del guadagno medio (GTAX_{AVG}).

Quarter	Algorithm	Trade Type	NumP	$\text{GTAX}_{\text{DAILY}}$	GTAX_{AVG}		
					Mean	**Max**	**StdDev**
2011-1	W_SimpleCart	LONG	233	0.12	0.03	5.44	1.51
2011-2	W_SimpleCart	SHORT	103	0.28	0.17	4.00	1.53
2011-3	Decision_Stump	LONG	283	2.99	0.69	13.84	3.47
2011-4	W_SimpleCart	SHORT	74	0.28	0.24	8.81	3.76
2012-1	Decision_Tree	LONG	315	0.35	0.07	30.18	3.17
2012-2	W_SimpleCart	SHORT	189	0.27	0.09	8.66	2.96
2012-3	Decision_Stump	LONG	153	0.86	0.36	8.70	2.79
2012-4	W_SimpleCart	SHORT	16	0.19	0.75	5.51	2.41
2013-1	Decision_Stump	LONG	193	1.08	0.35	10.33	2.50
2013-2	W_SimpleCart	LONG	178	0.14	0.05	5.75	2.04
2013-3	Decision_Stump	SHORT	33	-0.35	-0.68	4.68	2.58
2013-4	Decision_Stump	LONG	61	0.41	0.42	5.91	1.95
2014-1	W_SimpleCart	LONG	84	0.34	0.25	10.84	2.28
2014-2	W_SimpleCart	SHORT	73	0.28	0.23	4.16	1.77
2014-3	Decision_Stump	LONG	81	0.10	0.08	10.91	2.40
2014-4	W_RandomForest	SHORT	81	0.13	0.10	7.58	2.14
2015-1	W_SimpleCart	LONG	143	1.05	0.46	9.53	2.02
2015-2	W_SimpleCart	SHORT	60	-0.25	-0.26	7.11	2.41
2015-3	W_LADTree	SHORT	50	0.56	0.74	7.35	2.63
2015-4	W_SimpleCart	LONG	66	0.06	0.06	3.91	1.57
2016-1	W_SimpleCart	SHORT	79	0.17	0.13	9.17	3.59
2016-2	Decision_Tree	SHORT	350	-0.64	-0.12	12.62	2.59
2016-3	W_SimpleCart	LONG	102	0.60	0.39	8.04	2.42
2016-4	Decision_Stump	LONG	118	1.57	0.85	10.26	2.56
2017-1	Decision_Tree	LONG	260	0.60	0.15	9.44	1.84
2017-2	Decision_Tree	SHORT	25	0.03	0.07	4.28	1.85
2017-3	W_RandomForest	LONG	85	-0.01	-0.01	3.72	1.10
2017-4	Decision_Stump	LONG	43	0.14	0.20	4.61	1.48

Tabella 6.11: Guadagni dei classificatori scelti automaticamente per trimestre

Osservando le colonne relative ai guadagni si può notare che la scelta operata dal modello di regressione non è sempre ottimale: in quattro occasioni infatti (2013-3, 2015-2, 2016-2 e 2017-3) è stata sezionata una configurazione che ha prodotto perdite. Tra queste, la scelta peggiore è quella del secondo trimestre 2016, dove è risultata una perdita giornaliera dello 0.64% a fronte di 350 raccomandazioni. Negli altri tre casi, invece, il numero minore di raccomandazioni ha determinato perdite

relativamente minori a quelle del 2016-2.

Se, comunque, si considerano i risultati nell'arco dei sette anni testati, la maggior parte delle scelte ha avuto esisto positivo, come si può osservare nella rappresentazione grafica della figura 6.18.

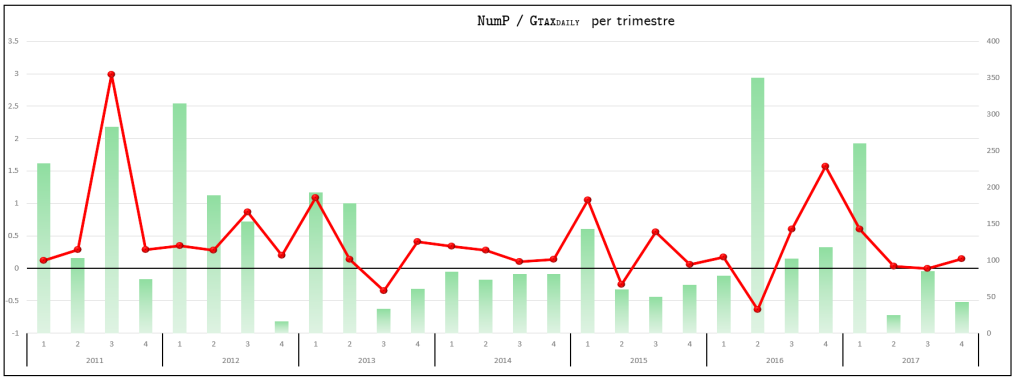

Figura 6.18: Prestazioni self-tuning per trimestre

A supporto di quest'ultima affermazione, nella tabella 6.12 si riportano le stesse metriche aggregate per anno, evidenziando sempre valori positivi dei guadagni.

Year	NumP	$G_{TAX_{DAILY}}$	$G_{TAX_{AVG}}$		
			Mean	Max	StdDev
2011	693	0.93	0.34	13.84	2.76
2012	673	0.42	0.16	30.18	3.01
2013	465	0.31	0.17	10.33	2.28
2014	319	0.21	0.17	10.91	2.16
2015	319	0.36	0.29	9.53	2.14
2016	649	0.43	0.17	12.62	2.72
2017	413	0.19	0.12	9.44	1.67

Tabella 6.12: Guadagni dei classificatori scelti automaticamente per anno

A questo punto è stato appurato che la funzione di self-tuning, generalmente, può supportare nella configurazione dei classificatori conseguendo discreti guadagni, anche se non tutte le scelte sono eseguite correttamente. Tuttavia, i guadagni calcolati per anno risultano minori di quelli osservati nella sezione 6.1, dove diverse simulazioni si sono distinte dalle altre per aver generato rendimenti elevati per un intero anno. Nella sezione che segue si confrontano i risultati del self-tuning con quelli delle singole migliori simulazioni.

6.2.3 Confronto con le simulazioni migliori

Analogamente a quanto fatto con i classificatori scelti dalla funzione di self-tuning, si procede ora con l'analisi delle prestazioni misurate sulle simulazioni che sono risultate migliori di tutte le altre nell'arco di un anno di trading: ovvero quelle a cui corrisponde il maggior guadagno giornaliero nella tabella 6.4. Lo scopo dell'analisi è dimostrare che è più efficiente cambiare strategia di classificazione nel corso del tempo, piuttosto che usare lo stesso metodo, anche se almeno una volta è risultato migliore di tutti gli altri metodi testati.

La tabella 6.13 riporta l'elenco delle migliori simulazioni per ogni anno e la tabella 6.14 ne mostra i dettagli relativi al numero di raccomandazioni e ai guadagni.

Year	Algorithm	Training Days	Sliding Window	Label Disc.	Cut Point	Trade Type	GTAX$_{DAILY}$
2010	Decision_Stump	30	2	BINOMINAL	1.0	LONG	0.40
2011	Naive_Bayes	15	4	POLY_4BINS	1.0	SHORT	2.13
2012	Decision_Stump	25	5	BINOMINAL	1.0	LONG	0.87
2013	Decision_Stump	10	5	BINOMINAL	4.0	LONG	0.99
2014	W_LADTree	10	4	BINOMINAL	-4.0	SHORT	0.84
2015	Decision_Stump	20	2	POLY_4BINS	0.0	LONG	1.08
2016	Naive_Bayes_Kernel	20	4	POLY_4BINS	2.0	LONG	0.77
2017	Rule_Induction	25	3	POLY_4BINS	2.0	LONG	0.26

Tabella 6.13: Migliori simulazioni di ogni anno

Year	Algorithm	Trade Type	NumP	GTAX$_{DAILY}$	GTAX$_{AVG}$		
					Mean	Max	StdDev
2010	Decision_Stump	LONG	955	0.40	0.11	20.78	2.39
2011	Naive_Bayes	SHORT	2248	2.13	0.24	15.65	2.96
2012	Decision_Stump	LONG	1117	0.87	0.20	30.18	2.99
2013	Decision_Stump	LONG	544	0.99	0.46	10.33	2.35
2014	W_LADTree	SHORT	566	0.84	0.37	13.56	2.13
2015	Decision_Stump	LONG	1203	1.08	0.23	9.53	2.16
2016	Naive_Bayes_Kernel	LONG	1445	0.77	0.14	12.90	2.66
2017	Rule_Induction	LONG	859	0.26	0.08	13.05	1.79

Tabella 6.14: Guadagni delle migliori simulazioni di ogni anno

Il confronto di questi risultati con quelli ottenuti attraverso il self-tuning risulta più evidente se mostrato graficamente[4]. Nei seguenti grafici le metriche relative al self-tuning sono marcate da dalla linea rossa più spessa:

- figure 6.19, 6.20, 6.21: confronto NumP, GTAX$_{AVG}$, GTAX$_{DAILY}$ per anno;
- figure 6.22, 6.23, 6.24: confronto NumP, GTAX$_{AVG}$, GTAX$_{DAILY}$ per trimestre.

[4]I dati numerici rappresentati nei grafici sono riportati in appendice nelle tabelle A.2 e A.3.

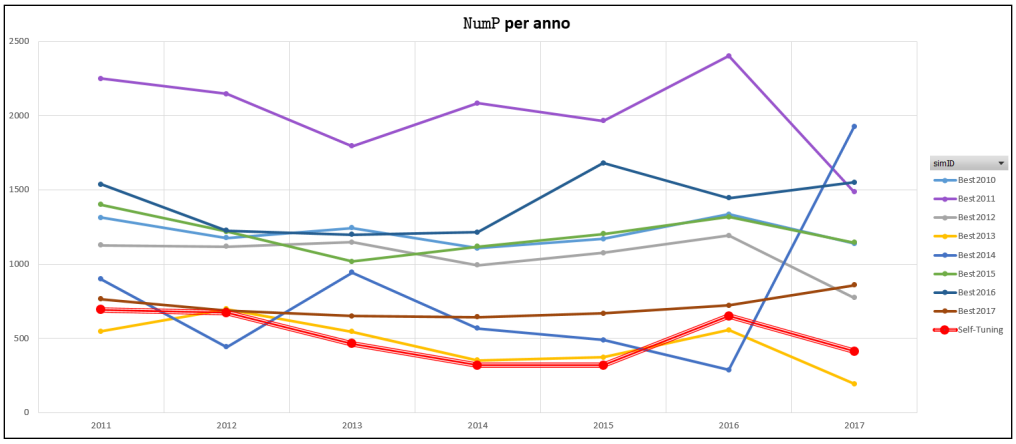

Figura 6.19: Confronto numero di raccomandazioni per anno

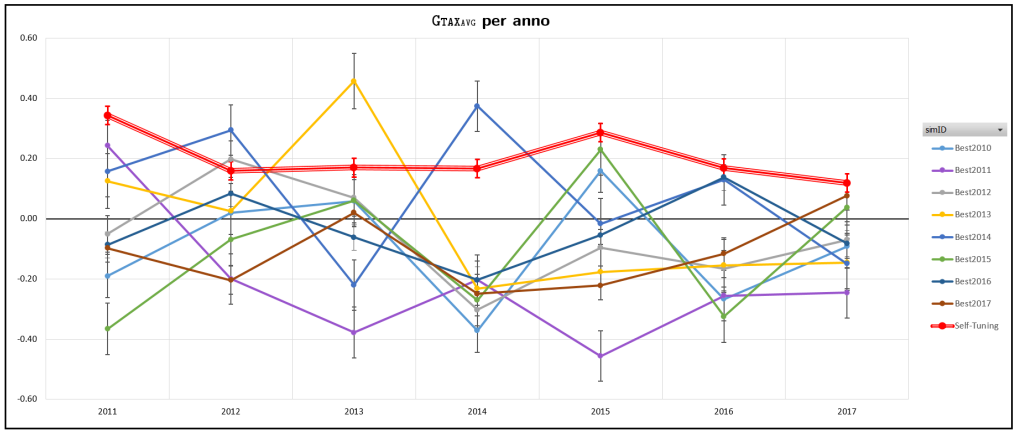

Figura 6.20: Confronto guadagno medio per anno

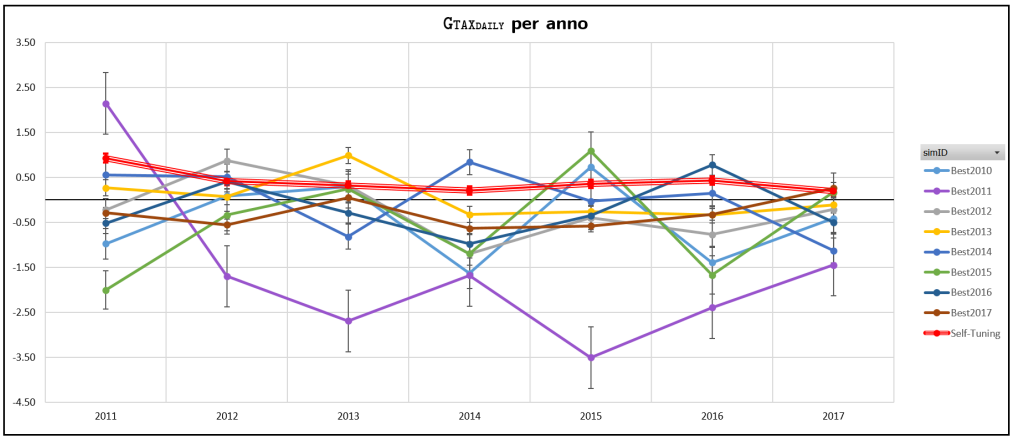

Figura 6.21: Confronto guadagno giornaliero per anno

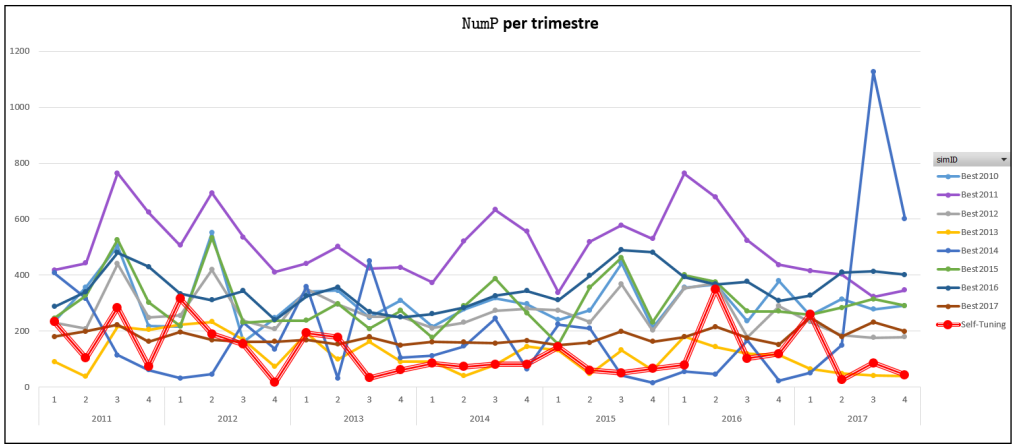

Figura 6.22: Confronto numero di raccomandazioni per trimestre

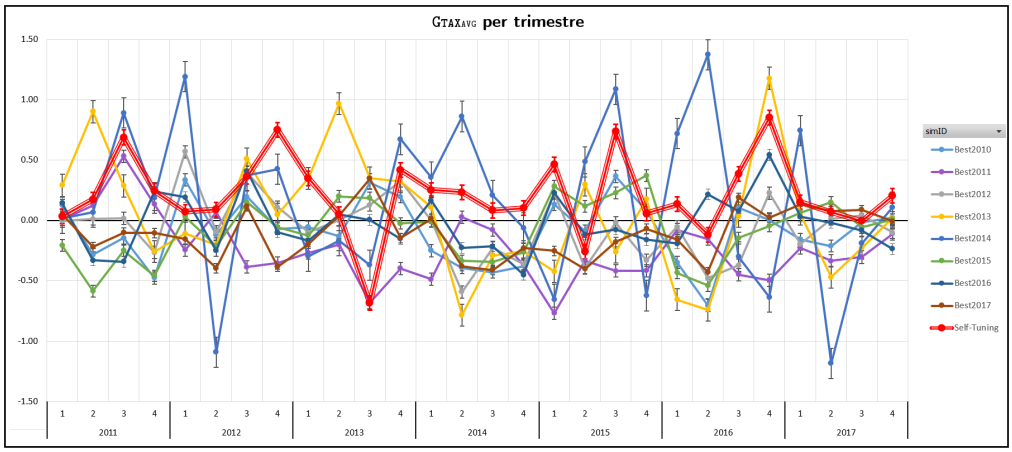

Figura 6.23: Confronto guadagno medio per trimestre

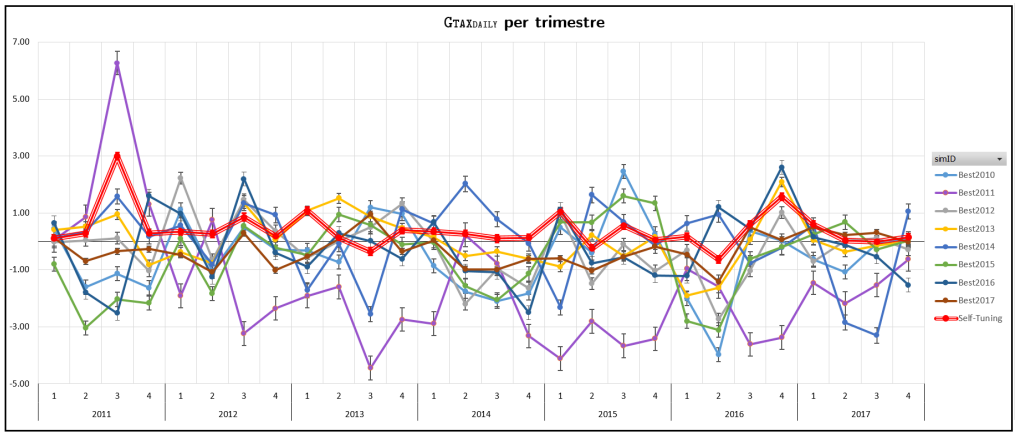

Figura 6.24: Confronto guadagno giornaliero per trimestre

Dai grafici riportati si possono trarre le seguenti conclusioni:

- le scelte del self-tuning non generano un alto numero di raccomandazioni rispetto alle altre configurazioni di classificatori (rif. 6.19, 6.22);
- su ogni anno dal 2011 al 2017, le scelte del self-tuning sono sempre positive, mentre le simulazioni che hanno performato meglio delle altre nel corso di almeno un anno di trading, non mantengono mai un rendimento positivo per più di due anni consecutivi (fig. 6.20, 6.21);
- su ogni trimestre dal 2011 al 2017, il guadagno giornaliero mostra un andamento più stabile delle altre simulazioni (fig. 6.24), il guadagno medio, invece, denota un andamento simile a quello degli altri modelli, però scende al di sotto dello 0% meno volte (6.23);
- la variabilità dei rendimenti, rappresentata nei grafici mediate le barre di errore, è nettamente più bassa per i risultati derivanti dal self-tuning rispetto a quelli ottenuti delle altre simulazioni (fig. 6.20, 6.21, 6.23 e 6.24).

Infine, per confermare definitivamente la robustezza dell'approccio basato sul self-tuning rispetto a quello basato sull'impiego di un unico modello, nella tabella 6.15 si riportano le metriche di performance aggregate su tutti gli anni di trading dal 2011 al 2017:

Classifier		**NumP**	**GTAX$_{DAILY}$**	**GTAX$_{AVG}$**		
				Mean	**Max**	**StdDev**
(Decision_Stump, LONG)	*Best of 2010*	9434	-0.36	-0.08	30.18	2.57
(Naive_Bayes, SHORT)	*Best of 2011*	15802	-1.56	-0.20	18.43	2.66
(Decision_Stump, LONG)	*Best of 2012*	8171	-0.23	-0.06	30.18	2.58
(Decision_Stump, LONG)	*Best of 2013*	3481	0.00	0.00	23.58	2.91
(W_LADTree, SHORT)	*Best of 2014*	7503	-0.20	-0.05	17.12	1.95
(Decision_Stump, LONG)	*Best of 2015*	9478	-0.55	-0.12	30.18	2.48
(Naive_Bayes_Kernel, LONG)	*Best of 2016*	11318	-0.37	-0.07	30.18	2.34
(Rule_Induction, LONG)	*Best of 2017*	5678	-0.34	-0.12	22.84	2.35
	Self-tuning	3531	0.36	0.21	30.18	2.53

Tabella 6.15: Confronto complessivo dal 2011 al 2017

Da quest'ultima tabella emerge che, a lungo termine, è necessario cambiare metodo di raccomandazione per evitare delle perdite: infatti, il guadagno associato ai classificatori scelti tramite self-tuning è l'unico ad essere positivo.

6.2.4 Confronto con sistema BeSt originale

In virtù del fatto che il sistema realizzato in questa tesi è fondato sul framework BeSt, descritto nella sezione 4.2, è opportuno misurare la metodologia proposta

anche con il sistema originale. Nello specifico si confrontano le performance degli algoritmi di regressione e sequence mining con quelli dei classificatori scelti dalla funzione di self-tuning.

I dati utilizzati per questa comparazione sono estratti dall'articolo [40] e fanno riferimento ai risultati ottenuti per le azioni del FTSE MIB negli anni 2011 e 2013. Di conseguenza, anche per la metodologia BeSt estesa, saranno utilizzate solamente le metriche misurate per l'anno 2011 e 2013.

Year	Algorithm	Trade Type	NumP	$GTAX_{AVG}$		
				Mean	Max	StdDev
2011	W-SEQ	SHORT	97	0.72	12.02	2.41
	UNW-SEQ	SHORT	245	0.22	9.15	1.69
	SVM_Linear	SHORT	191	0.40	20.85	2.39
	SVM_Polyn.	SHORT	191	0.48	20.85	2.40
	LR	SHORT	190	0.53	10.82	2.23
	RepTree	SHORT	190	0.38	20.85	2.32
	NN	SHORT	191	0.27	7.70	1.57
	Arima	SHORT	191	0.38	8.27	1.74
	Random	SHORT	244	0.31	5.58	1.51
	Self-tuning		**693**	**0.34**	**13.84**	**2.76**
2013	W-SEQ	LONG	125	0.63	22.83	2.72
	UNW-SEQ	LONG	261	0.36	22.83	2.06
	SVM_Linear	LONG	176	0.43	14.54	2.23
	SVM_Polyn.	LONG	175	0.36	14.54	2.02
	LR	LONG	175	0.38	14.54	1.86
	RepTree	LONG	175	0.75	16.29	2.52
	NN	LONG	176	0.43	14.54	2.23
	Arima	LONG	178	0.48	12.22	1.95
	Random	LONG	261	0.30	8.57	1.52
	Self-tuning		**465**	**0.17**	**10.33**	**2.28**

Tabella 6.16: Confronto 2011 e 2013 con sistema BeSt originale

Osservando i dati riportati nella tabella 6.16 si può notare che i metodi scelti tramite la funzione di self-tuning dei classificatori producono un numero nettamente maggiore di raccomandazioni rispetto ai modelli testati nel sistema BeSt originale. Questo numero ha impatto sul guadagno medio risultante che, nel 2011 risulta migliore solamente rispetto agli algoritmi `NN` e `Random`, mentre nel 2013 è il valore più basso rispetto a quello ottenuto da tutte le altre tecniche.

Pertanto, considerando i numeri di quest'ultimo confronto, la tecnica non supervisionata delle sequenze pesate (`W-SEQ`) si conferma nuovamente superiore agli altri metodi sperimentati.

Capitolo 7

Conclusioni e sviluppi futuri

Il lavoro svolto in questa tesi è stato incentrato sull'applicazione delle tecniche di classificazione nei sistemi di trading azionario quantitativo e sulla loro configurazione automatica attraverso lo sviluppo di un'estensione del framework BEst STock Finder (BeSt). Il sistema esteso che si è realizzato ha integrato i modelli di classificazione all'interno del processo di raccomandazione delle azioni ed ha permesso di sperimentare sistematicamente più tecniche e più parametri di configurazione dei modelli.

I risultati della sperimentazione attuata utilizzando tale sistema hanno portato alle seguenti constatazioni:

- L'accuratezza dei classificatori, in questo scenario applicativo, non è una metrica sufficiente a valutare le prestazioni dei modelli.
- Il numero di raccomandazioni estratte dai modelli dipende del valore del cut point usato per discretizzare l'attributo di classe: all'aumentare del valore assoluto del cut point, diminuisce il numero di previsioni.
- Le performance dei classificatori possono essere significativamente influenzate dalla scelta dei parametri utilizzati. Non è comunque possibile determinare un'unica configurazione applicabile sistematicamente nell'intraday trading.
- A lungo termine, la configurazione dinamica dei classificatori, per mezzo della funzione di self-tuning, risulta più robusta ai cambiamenti del mercato rispetto a qualsiasi altro modello di classificazione (di quelli testati) applicato singolarmente.
- Il procedimento del self-tuning si è dimostrato efficace solo se applicato su base trimestrale e solo sulle metriche di guadagno calcolate senza tenere conto di alcuna soglia di stop loss. Il sistema non è risultato né stabile né redditizio considerando intervalli di tempo differenti o le strategie di stop loss.

A fronte di quanto indicato, si possono considerare due aree su cui concentrare gli sviluppi futuri del sistema: una legata al miglioramento della procedura di self-tuning e, un'altra, relativa all'integrazione di nuovi modelli e nuovi parametri nel processo di simulazione.

Per quanto riguarda il self-tuning sarà necessario, prima di tutto, individuare le metriche di mercato, ed eventualmente tecniche alternative, per applicare l'auto-configurazione dei modelli e dei parametri su diversi intervalli temporali: per esempio su base mensile o settimanale. Successivamente, sempre su questo aspetto, si potrebbe pensare anche di integrare nel sistema ulteriori metodi di analisi, finalizzati a determinare dinamicamente anche la finestra di tempo (eliminando così la dimensione fissa del trimestre, mese o settimana).

Chiaramente, in uno scenario reale di trading online, è necessario impostare lo stop loss e, quindi, ha senso includere negli sviluppi futuri anche il self-tuning con i parametri relativi alla strategia di stop loss.

Considerando invece il processo di simulazione, occorre osservare che esso è stato progettato appositamente per applicare più algoritmi di machine learning e più combinazioni di parametri e, pertanto, può essere utilizzato nuovamente per misurare le prestazioni di altri modelli predittivi. Tra questi, potrebbe essere interessante recuperare quei classificatori che sono stati esclusi dagli esperimenti svolti in questa tesi (vedi tabella A.1 in appendice), poiché potrebbero risultare proficui in periodi che nelle fasi iniziali della sperimentazione non sono stati analizzati.

Il processo di simulazione, inoltre, potrà essere sfruttato per focalizzare le ricerche future nello studio dell'impatto di ulteriori parametri di configurazione degli algoritmi presenti in letteratura. Potenzialmente, ciò potrebbe permettere di scoprire nuove configurazioni efficaci dei classificatori e potrebbe anche produrre dati prestazionali che si prestano meglio alla procedura di self-tuning su molteplici intervalli di tempo.

In conclusione si può dunque affermare che le capacità del sistema BeSt, dell'estensione presentata in questa tesi e delle future estensioni costituiscono un valido strumento per coniugare trading quantitativo e machine learning.

Appendice A

Approfondimenti sulla sperimentazione svolta

L'appendice A raccoglie le tabelle che non sono state incluse direttamente nei capitoli della tesi a causa dell'elevata quantità di dati contenuti in esse.

Algoritmi di classificazione

La tabella A.1 espone l'elenco completo dei 61 algoritmi di classificazione utilizzabili all'interno del processo di simulazione del sistema BeSt esteso.

Come spiegato nella sezione 6.1.1, questi classificatori sono stati applicati solamente ai dati storci degli anni 2011 e 2013 in un primo stadio della sperimentazione. Poi sono stati filtrati al fine di risparmiare il tempo e lo spazio necessari a produrre i dati di performance con cui collaudare la funzione di self-tuning dei classificatori. La prima colonna della tabelle indica quali algoritmi stati mantenuti per svolgere gli esperimenti sui dati storici che includono gli anni di trading dal 2010 al 2017.

Usato	Categoria	Algoritmo	Classificazione polinomiale
✓	Decision tree	`Decision_Tree`	✓
✓	Decision tree	`Decision_Stump`	✓
✓	Decision tree	`Random_Tree`	✓
✓	Decision tree	`W_REPTree`	✓
✓	Decision tree	`W_RandomForest`	✓
✓	Decision tree	`W_BFTree`	✓
✓	Decision tree	`W_Decision_Stump`	✓
✓	Decision tree	`W_J48`	✓
✓	Decision tree	`W_J48graft`	✓
✓	Decision tree	`W_LADTree`	✓
✓	Decision tree	`W_NBTree`	✓

Tabella A.1 – continua

Tabella A.1 – continuazione

Usato	Categoria	Algoritmo	Classificazione polinomiale
✓	Decision tree	W_SimpleCart	✓
✓	Bayesian classifier	Naive_Bayes	✓
✓	Bayesian classifier	Naive_Bayes_Kernel	✓
✓	Rule induction	Rule_Induction	✓
✓	Bayesian network	W_Jrip	✓
✓	Bayesian network	W_BayesNet	✓
✓	Bayesian network	W_BIFReader	✓
✓	Bayesian network	W_BayesNetGenerator	✓
✓	Bayesian network	W_EditableBayesNet	✓
✗	Lazy modeling	K_NN	✓
✗	Lazy modeling	Default_Model	✓
✗	Lazy modeling	W_IB1	✓
✗	Lazy modeling	W_IBk	✓
✗	Lazy modeling	W_Kstar	✓
✗	Lazy modeling	W_LWL	✓
✗	Decision tree	Random_Forest	✓
✗	Decision tree	W_RandomTree	✓
✗	Decision tree	W_FT	✓
✗	Decision tree	W_LMT	✓
✗	Decision tree	W_LogisticBase	✓
✗	Decision tree	W_ADTree	✗
✗	Bayesian classifier	W_DMNBtext	✓
✗	Bayesian classifier	W_NaiveBayesMultinomial	✓
✗	Bayesian classifier	W_NaiveBayes-MultinomialUpdateable	✓
✗	Bayesian classifier	W_ComplementNaiveBayes	✓
✗	Bayesian classifier	W_Bayesian-LogisticRegression	✗
✗	Voting classifier	W_VFI	✓
✗	Rule induction	Single_Rule_Induction Single_Attribute	✓
✗	Rule induction	W_ConjunctiveRule	✓
✗	Rule induction	W_DTNB	✓
✗	Rule induction	W_DecisionTable	✓
✗	Rule induction	W_OneR	✓
✗	Rule induction	W_PART	✓
✗	Rule induction	W_Ridor	✓
✗	Rule induction	W_ZeroR	✓
✗	Neural network	Neural_Net	✓
✗	Neural network	AutoMLP	✓
✗	Neural network	Perceptron	✗
✗	SVM	SVM_LibSVM	✓

Tabella A.1 – continua

Tabella A.1 – continuazione

Usato	Categoria	Algoritmo	Classificazione polinomiale
✗	SVM	W_HyperPipes	✓
✗	SVM	SVM	✗
✗	SVM	SVM_Linear	✗
✗	SVM	SVM_Evolutionary	✗
✗	SVM	SVM_PSO	✗
✗	SVM	SVM_Fast_Large_Margin	✗
✗	SVM	SVM_Hyper_Hyper	✗
✗	SVM	SVM_LibSVM_Poly	✓
✗	SVM	SVM_Poly	✗
✗	SVM	SVM_Evolutionary_Poly	✗
✗	SVM	SVM_PSO_Poly	✗

Tabella A.1: Elenco completo algoritmi di classificazione applicabili

Facendo riferimento ai risultati ottenuti dai 61 classificatori negli esperimenti sui dati storici degli anni 2011 e 2013, si riportano nella figure A.1 e A.2 i grafici che mettono a confronto, rispettivamente, il *numero medio*[1] *di predizioni* e il *guadagno assoluto* misurati per ciascun modello.

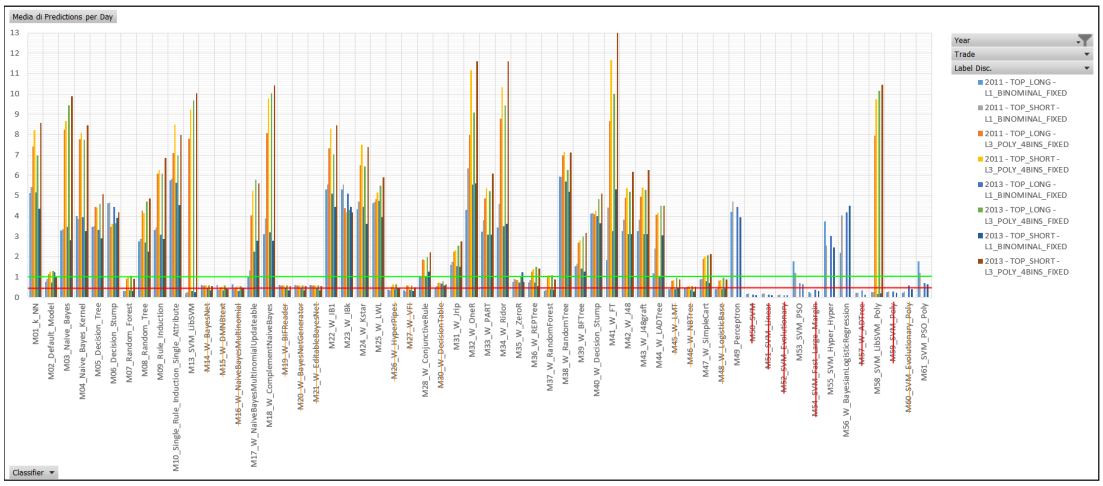

Figura A.1: Confronto delle predizioni giornaliere

Osservando il grafico in figura A.1 si può notare che molti classificatori generano, mediamente, meno di una raccomandazione al giorno, altri meno di una raccomandazione ogni due giorni. La linea rossa delimita la media di 0.5 predizioni e quella

[1]La media è calcolata considerando 255 giornate di trading in un anno.

verde la media di 1 predizione al giorno. Secondo tale rappresentazione, la linea verde identifica la soglia usata per filtrare i modelli di classificazione: tutti i modelli, il cui numero di predizioni è riportato al di sotto della linea verde (< 1), sono quelli esclusi.

Procedendo con lo stesso tipo di analisi grafica anche per le metriche di guadagno, nel grafico in figura A.2 la linea verde rappresenta la soglia di guadagno minimo richiesto a fronte di raccomandazioni `LONG` e quella rossa a fronte delle raccomandazioni `SHORT`. Per filtrare i classificatori in base al rendimento, in questo caso si è definita una soglia di guadagno *assoluto* pari a 150 (-150 nel caso `SHORT`).

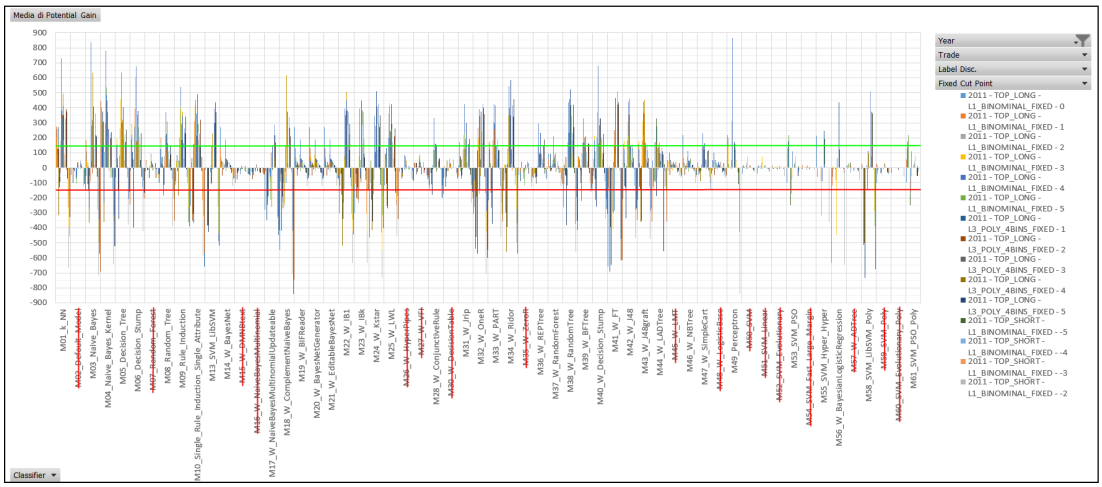

Figura A.2: Confronto dei potenziali guadagni

Risultati delle simulazioni

La tabella A.2 riporta per esteso le prestazioni ottenute dalle simulazioni che, in almeno uno degli anni di trading simulati, hanno realizzato il massimo rendimento. Al fondo della medesima tabella sono incluse anche le prestazioni dei classificatori scelti dalla funzione di self-tuning.

Questi dati sono quelli da cui sono stati derivati i grafici 6.19, 6.20 e 6.21.

Algorithm	Trade Type	Year	NumP	GTAX$_{DAILY}$	GTAX$_{AVG}$		
					Mean	Max	StdDev
best of 2010							
Decision_Stump	LONG	2011	1312	-0.98	-0.19	23.03	3.05
		2012	1176	0.09	0.02	30.18	2.96
		2013	1243	0.29	0.06	16.30	2.18
		2014	1106	-1.63	-0.37	12.64	2.20
		2015	1169	0.73	0.16	9.31	2.27
		2016	1335	-1.39	-0.27	14.30	3.21

Tabella A.2 – continua

Tabella A.2 – continuazione

Algorithm	Trade Type	Year	NumP	$G_{TAX_{DAILY}}$	$G_{TAX_{AVG}}$		
					Mean	Max	StdDev
		2017	1138	-0.42	-0.09	11.81	1.60
best of 2011							
Naive_Bayes	SHORT	2011	2248	2.13	0.24	15.65	2.96
		2012	2146	-1.70	-0.20	17.12	3.19
		2013	1794	-2.69	-0.38	9.41	2.31
		2014	2083	-1.68	-0.20	13.56	2.29
		2015	1965	-3.51	-0.46	13.22	2.29
		2016	2402	-2.40	-0.26	18.43	3.13
		2017	1486	-1.45	-0.25	7.64	1.75
best of 2012							
Decision_Stump	LONG	2011	1127	-0.22	-0.05	23.03	3.16
		2012	1117	0.87	0.20	30.18	2.99
		2013	1148	0.32	0.07	22.84	2.16
		2014	993	-1.19	-0.30	6.68	2.18
		2015	1075	-0.40	-0.10	8.36	2.28
		2016	1191	-0.77	-0.17	15.77	3.17
		2017	772	-0.21	-0.07	9.57	1.76
best of 2013							
Decision_Stump	LONG	2011	547	0.27	0.13	23.03	3.44
		2012	696	0.07	0.03	23.58	2.91
		2013	544	0.99	0.46	10.33	2.35
		2014	353	-0.32	-0.23	12.64	2.33
		2015	372	-0.26	-0.18	9.33	2.38
		2016	555	-0.33	-0.15	23.18	3.89
		2017	192	-0.11	-0.15	11.81	1.98
best of 2014							
W_LADTree	SHORT	2011	898	0.55	0.16	11.73	2.61
		2012	441	0.51	0.29	17.12	2.56
		2013	943	-0.82	-0.22	9.42	1.89
		2014	566	0.84	0.37	13.56	2.13
		2015	489	-0.03	-0.02	7.35	1.99
		2016	288	0.15	0.13	12.62	2.94
		2017	1927	-1.13	-0.15	7.09	1.46
best of 2015							
Decision_Stump	LONG	2011	1401	-2.00	-0.37	11.49	2.80
		2012	1219	-0.34	-0.07	30.18	2.95
		2013	1017	0.24	0.06	9.21	2.00
		2014	1117	-1.20	-0.27	12.64	2.13
		2015	1203	1.08	0.23	9.53	2.16
		2016	1317	-1.67	-0.33	14.30	3.01
		2017	1145	0.17	0.04	13.05	1.59
best of 2016							
Naive_Bayes	LONG	2011	1537	-0.52	-0.09	23.03	2.98
_Kernel		2012	1226	0.40	0.08	30.18	2.91

Tabella A.2 – continua

Tabella A.2 – continuazione

Algorithm	Trade Type	Year	NumP	$G_{TAX_{DAILY}}$	$G_{TAX_{AVG}}$		
					Mean	Max	StdDev
		2013	1199	-0.29	-0.06	10.22	2.23
		2014	1214	-0.98	-0.20	12.64	2.18
		2015	1679	-0.36	-0.05	9.33	2.07
		2016	1445	0.77	0.14	12.90	2.66
		2017	1550	-0.50	-0.08	10.21	1.55
best of 2017							
Rule_Induction	LONG	2011	763	-0.29	-0.10	14.48	2.87
		2012	688	-0.56	-0.20	12.43	2.69
		2013	649	0.05	0.02	22.84	2.31
		2014	643	-0.64	-0.25	5.22	2.04
		2015	668	-0.58	-0.22	8.98	2.10
		2016	721	-0.33	-0.12	9.47	2.69
		2017	859	0.26	0.08	13.05	1.79
self-tuning							
-	-	2011	693	0.93	0.34	13.84	2.76
		2012	673	0.42	0.16	30.18	3.01
		2013	465	0.31	0.17	10.33	2.28
		2014	319	0.21	0.17	10.91	2.16
		2015	319	0.36	0.29	9.53	2.14
		2016	649	0.43	0.17	12.62	2.72
		2017	413	0.19	0.12	9.44	1.67

Tabella A.2: Guadagni migliori simulazioni di ogni anno

Analogamente alla tabella precedente, in A.3 si riportano gli stessi risultati, ma calcolati su ogni trimestre. In questo caso i dati sono serviti a generare i grafici 6.22, 6.23, 6.24.

Algorithm	Trade Type	QuarterNumP		$G_{TAX_{DAILY}}$	$G_{TAX_{AVG}}$		
					Mean	Max	StdDev
best of 2010							
Decision_Stump	LONG	2011-1	234	0.45	0.12	9.53	1.88
		2011-2	356	-1.61	-0.28	23.03	2.31
		2011-3	504	-1.13	-0.15	10.85	3.72
		2011-4	218	-1.63	-0.48	11.37	3.36
		2012-1	216	1.11	0.34	30.18	3.48
		2012-2	551	-1.04	-0.12	14.11	3.01
		2012-3	162	0.50	0.20	10.46	3.39
		2012-4	247	-0.28	-0.07	7.10	1.82
		2013-1	341	-0.31	-0.06	10.33	2.26
		2013-2	345	-0.73	-0.13	7.64	2.35
		2013-3	248	1.20	0.31	12.01	1.93

Tabella A.3 – continua

Tabella A.3 – continuazione

Algorithm	Trade Type	QuarterNumP		$G_{TAX_{DAILY}}$	$G_{TAX_{AVG}}$		
					Mean	**Max**	**StdDev**
		2013-4	309	0.97	0.19	16.30	2.06
		2014-1	215	-0.86	-0.25	6.14	1.70
		2014-2	277	-1.75	-0.39	6.68	2.24
		2014-3	318	-2.10	-0.43	5.18	2.19
		2014-4	296	-1.81	-0.38	12.64	2.50
		2015-1	240	0.50	0.13	6.97	1.94
		2015-2	274	-0.40	-0.09	6.21	1.92
		2015-3	443	2.45	0.36	9.31	2.77
		2015-4	212	0.27	0.08	4.26	1.77
		2016-1	355	-2.00	-0.35	14.30	3.48
		2016-2	366	-3.97	-0.70	9.29	4.18
		2016-3	235	0.37	0.10	9.20	2.32
		2016-4	379	0.03	0.00	9.27	2.11
		2017-1	256	-0.62	-0.16	7.99	1.79
		2017-2	314	-1.07	-0.21	5.73	1.69
		2017-3	278	-0.07	-0.02	5.30	1.35
		2017-4	290	0.10	0.02	11.81	1.54
best of 2011							
Naive_Bayes	SHORT	2011-1	418	0.04	0.01	12.21	2.03
		2011-2	442	0.85	0.12	9.74	1.90
		2011-3	764	6.23	0.53	11.98	3.48
		2011-4	624	1.31	0.13	15.65	3.36
		2012-1	506	-1.91	-0.24	17.12	3.12
		2012-2	693	0.76	0.07	14.11	3.20
		2012-3	536	-3.23	-0.39	16.55	3.89
		2012-4	411	-2.35	-0.35	6.62	2.01
		2013-1	441	-1.91	-0.27	8.92	2.64
		2013-2	502	-1.60	-0.20	9.27	2.31
		2013-3	424	-4.44	-0.68	5.13	2.41
		2013-4	427	-2.74	-0.40	9.41	1.73
		2014-1	373	-2.89	-0.49	7.93	2.23
		2014-2	521	0.23	0.03	11.54	2.32
		2014-3	633	-0.77	-0.08	13.56	2.14
		2014-4	556	-3.32	-0.37	6.47	2.44
		2015-1	338	-4.12	-0.77	8.21	2.04
		2015-2	519	-2.80	-0.34	13.22	1.95
		2015-3	578	-3.67	-0.42	9.46	2.92
		2015-4	530	-3.45	-0.42	6.30	1.92
		2016-1	763	-0.96	-0.08	11.78	3.33
		2016-2	678	-1.58	-0.15	18.43	3.27
		2016-3	524	-3.61	-0.45	12.21	2.80
		2016-4	437	-3.38	-0.49	14.56	2.92
		2017-1	416	-1.45	-0.23	6.65	1.94
		2017-2	402	-2.17	-0.34	7.64	1.95
		2017-3	322	-1.53	-0.30	3.80	1.35

Tabella A.3 – continua

Tabella A.3 – continuazione

Algorithm	Trade Type	QuarterNumP	$G_{TAX_{DAILY}}$	$G_{TAX_{AVG}}$		
				Mean	**Max**	**StdDev**
		2017-4	346	-0.62	-0.11	6.15	1.58
best of 2012							
Decision_Stump	LONG	2011-1	231	-0.02	0.00	9.53	1.94
		2011-2	208	0.04	0.01	23.03	2.57
		2011-3	440	0.11	0.02	13.84	3.88
		2011-4	248	-1.03	-0.27	11.37	3.13
		2012-1	255	2.23	0.57	30.18	3.29
		2012-2	419	-0.64	-0.10	23.58	3.30
		2012-3	236	1.46	0.40	8.70	2.91
		2012-4	207	0.37	0.11	5.46	1.70
		2013-1	350	-0.57	-0.10	7.06	2.11
		2013-2	295	0.01	0.00	7.64	2.28
		2013-3	252	0.49	0.13	22.84	2.39
		2013-4	251	1.33	0.33	6.09	1.81
		2014-1	210	0.00	0.00	6.06	1.95
		2014-2	230	-2.20	-0.59	6.68	2.43
		2014-3	273	-0.96	-0.23	5.43	2.05
		2014-4	280	-1.64	-0.36	6.49	2.22
		2015-1	274	0.98	0.23	8.36	2.02
		2015-2	232	-1.48	-0.39	4.97	2.09
		2015-3	367	-0.10	-0.02	8.07	2.78
		2015-4	202	-1.04	-0.33	3.96	1.67
		2016-1	353	-0.31	-0.05	14.01	3.26
		2016-2	371	-2.74	-0.48	9.29	3.89
		2016-3	178	-1.02	-0.37	7.89	2.16
		2016-4	289	1.02	0.23	15.77	2.43
		2017-1	233	-0.71	-0.20	8.06	2.09
		2017-2	184	0.04	0.01	9.19	1.79
		2017-3	176	0.12	0.04	3.67	1.34
		2017-4	179	-0.29	-0.10	9.57	1.61
best of 2013							
Decision_Stump	LONG	2011-1	90	0.41	0.29	17.10	2.72
		2011-2	37	0.53	0.90	23.03	4.22
		2011-3	216	0.95	0.29	10.31	3.55
		2011-4	204	-0.82	-0.26	8.71	3.42
		2012-1	222	-0.37	-0.11	6.81	2.31
		2012-2	233	-0.77	-0.20	23.58	3.48
		2012-3	169	1.34	0.51	10.46	3.07
		2012-4	72	0.06	0.05	4.28	1.93
		2013-1	193	1.08	0.35	10.33	2.50
		2013-2	99	1.52	0.97	8.27	2.65
		2013-3	162	0.87	0.35	6.74	2.08
		2013-4	90	0.47	0.32	6.60	2.10
		2014-1	90	0.16	0.11	5.29	2.14
		2014-2	40	-0.51	-0.78	3.07	2.09

Tabella A.3 – continua

Tabella A.3 – continuazione

Algorithm	Trade Type	QuarterNumP	$G_{TAX_{DAILY}}$	$G_{TAX_{AVG}}$		
				Mean	**Max**	**StdDev**
		2014-3	78	-0.35	-0.29	5.07	2.17
		2014-4	145	-0.60	-0.26	12.64	2.57
		2015-1	132	-0.89	-0.42	9.33	2.62
		2015-2	47	0.22	0.29	4.72	1.96
		2015-3	132	-0.52	-0.26	4.83	2.33
		2015-4	61	0.17	0.18	6.59	2.19
		2016-1	180	-1.90	-0.66	5.98	3.40
		2016-2	143	-1.63	-0.74	7.60	5.13
		2016-3	118	0.06	0.03	12.90	3.08
		2016-4	114	2.09	1.18	23.18	3.24
		2017-1	64	0.05	0.05	4.40	2.22
		2017-2	48	-0.36	-0.47	2.78	1.35
		2017-3	41	-0.15	-0.23	2.82	1.62
		2017-4	39	0.01	0.02	11.81	2.50
best of 2014							
W_LADTree	SHORT	2011-1	407	0.11	0.02	7.34	2.17
		2011-2	316	0.33	0.07	9.74	1.80
		2011-3	114	1.56	0.89	11.73	4.48
		2011-4	61	0.18	0.19	9.49	3.78
		2012-1	31	0.57	1.19	17.12	4.10
		2012-2	46	-0.81	-1.09	5.07	3.96
		2012-3	229	1.34	0.37	13.61	2.20
		2012-4	135	0.92	0.42	6.01	1.86
		2013-1	359	-1.71	-0.30	9.42	2.21
		2013-2	31	-0.08	-0.17	3.43	1.95
		2013-3	449	-2.56	-0.37	4.58	1.60
		2013-4	104	1.13	0.67	4.34	1.63
		2014-1	111	0.63	0.36	6.79	2.09
		2014-2	146	2.02	0.86	11.54	2.01
		2014-3	245	0.78	0.21	13.56	2.24
		2014-4	64	-0.07	-0.06	4.17	1.90
		2015-1	223	-2.31	-0.65	6.47	1.83
		2015-2	209	1.63	0.48	4.32	1.71
		2015-3	42	0.69	1.09	7.35	2.79
		2015-4	15	-0.15	-0.62	3.19	2.22
		2016-1	55	0.64	0.72	8.13	3.69
		2016-2	45	0.95	1.37	12.62	4.35
		2016-3	166	-0.77	-0.30	12.21	1.93
		2016-4	22	-0.22	-0.63	4.61	2.84
		2017-1	50	0.57	0.74	4.96	2.01
		2017-2	149	-2.84	-1.18	5.36	2.53
		2017-3	1126	-3.33	-0.19	7.09	1.24
		2017-4	602	1.05	0.11	5.83	1.27
best of 2015							
Decision_Stump	LONG	2011-1	246	-0.80	-0.21	6.58	1.70

Tabella A.3 – continua

Tabella A.3 – continuazione

Algorithm	Trade Type	QuarterNumP		$G_{TAX_{DAILY}}$	$G_{TAX_{AVG}}$		
					Mean	**Max**	**StdDev**
		2011-2	327	-3.04	-0.59	5.01	1.82
		2011-3	526	-2.03	-0.25	11.49	3.52
		2011-4	302	-2.16	-0.46	8.71	2.95
		2012-1	217	0.11	0.03	30.18	3.69
		2012-2	533	-1.83	-0.21	11.43	2.93
		2012-3	231	0.54	0.15	9.51	3.16
		2012-4	238	-0.21	-0.05	7.10	1.85
		2013-1	238	-0.48	-0.12	5.42	2.05
		2013-2	298	0.94	0.20	9.02	2.24
		2013-3	208	0.59	0.18	9.21	2.05
		2013-4	273	-0.11	-0.02	6.60	1.60
		2014-1	176	-0.03	-0.01	6.14	1.94
		2014-2	289	-1.56	-0.34	6.33	2.04
		2014-3	389	-2.05	-0.34	6.06	1.93
		2014-4	265	-1.13	-0.26	12.64	2.57
		2015-1	153	0.68	0.28	9.53	1.98
		2015-2	356	0.68	0.12	8.98	2.12
		2015-3	462	1.60	0.23	8.90	2.42
		2015-4	232	1.34	0.37	4.80	1.75
		2016-1	400	-2.80	-0.43	14.30	3.39
		2016-2	375	-3.11	-0.54	9.57	3.67
		2016-3	271	-0.61	-0.15	7.75	2.24
		2016-4	271	-0.20	-0.05	7.84	1.79
		2017-1	257	0.25	0.06	6.48	1.56
		2017-2	284	0.68	0.15	13.05	1.89
		2017-3	314	-0.29	-0.06	5.30	1.30
		2017-4	290	0.04	0.01	9.57	1.59
best of 2016							
Naive_Bayes	LONG	2011-1	287	0.64	0.14	10.14	2.03
_Kernel		2011-2	340	-1.79	-0.33	23.03	2.44
		2011-3	480	-2.52	-0.34	10.85	3.21
		2011-4	430	1.59	0.24	14.48	3.54
		2012-1	333	0.98	0.19	30.18	3.29
		2012-2	310	-1.25	-0.25	14.11	3.25
		2012-3	343	2.19	0.41	9.21	2.73
		2012-4	240	-0.39	-0.10	7.40	1.95
		2013-1	325	-0.89	-0.17	10.22	2.29
		2013-2	357	0.29	0.05	8.27	2.32
		2013-3	268	0.02	0.00	7.32	2.24
		2013-4	249	-0.61	-0.15	6.37	1.99
		2014-1	261	0.68	0.16	8.90	2.03
		2014-2	283	-1.04	-0.23	9.07	2.05
		2014-3	326	-1.07	-0.21	6.57	2.11
		2014-4	344	-2.50	-0.45	12.64	2.42
		2015-1	310	1.12	0.23	9.33	2.04

Tabella A.3 – continua

Tabella A.3 – continuazione

Algorithm	Trade Type	QuarterNumP		$G_{TAX_{DAILY}}$	$G_{TAX_{AVG}}$		
					Mean	**Max**	**StdDev**
		2015-2	398	-0.76	-0.12	8.98	2.10
		2015-3	490	-0.56	-0.08	6.02	2.22
		2015-4	481	-1.21	-0.16	8.55	1.89
		2016-1	393	-1.21	-0.19	9.98	3.07
		2016-2	367	1.21	0.21	9.47	2.62
		2016-3	377	0.43	0.07	12.90	2.49
		2016-4	308	2.61	0.54	10.26	2.28
		2017-1	327	0.15	0.03	9.44	1.79
		2017-2	409	-0.13	-0.02	10.21	1.73
		2017-3	413	-0.54	-0.08	5.30	1.28
		2017-4	401	-1.53	-0.24	5.91	1.37
best of 2017							
Rule_Induction	LONG	2011-1	180	0.13	0.05	10.14	1.86
		2011-2	199	-0.70	-0.22	4.57	1.53
		2011-3	222	-0.35	-0.10	10.38	3.77
		2011-4	162	-0.26	-0.10	14.48	3.58
		2012-1	196	-0.47	-0.15	8.54	2.52
		2012-2	168	-1.07	-0.40	10.83	3.22
		2012-3	162	0.29	0.12	12.43	2.90
		2012-4	162	-1.01	-0.38	7.10	1.97
		2013-1	168	-0.53	-0.20	6.24	2.29
		2013-2	153	0.10	0.04	6.83	2.20
		2013-3	179	0.95	0.35	22.84	2.78
		2013-4	149	-0.35	-0.15	4.56	1.70
		2014-1	161	0.04	0.01	4.50	1.73
		2014-2	160	-0.98	-0.38	4.85	1.83
		2014-3	156	-0.99	-0.41	4.14	2.14
		2014-4	166	-0.61	-0.23	5.22	2.38
		2015-1	149	-0.60	-0.25	4.86	2.06
		2015-2	158	-1.03	-0.40	8.98	2.17
		2015-3	199	-0.54	-0.18	6.02	2.16
		2015-4	162	-0.18	-0.07	8.55	1.99
		2016-1	179	-0.46	-0.16	8.16	2.81
		2016-2	215	-1.42	-0.43	9.47	3.38
		2016-3	175	0.51	0.19	8.00	2.12
		2016-4	152	0.06	0.02	5.73	1.88
		2017-1	248	0.51	0.13	9.44	2.07
		2017-2	180	0.22	0.08	13.05	2.19
		2017-3	232	0.31	0.09	8.00	1.41
		2017-4	199	-0.02	-0.01	4.19	1.36
self-tuning							
–	–	2011-1	233	0.12	0.03	5.44	1.51
		2011-2	103	0.28	0.17	4.00	1.53
		2011-3	283	2.99	0.69	13.84	3.47
		2011-4	74	0.28	0.24	8.81	3.76

Tabella A.3 – continua

Tabella A.3 – continuazione

Algorithm	Trade Type	QuarterNumP	$G_{TAX_{DAILY}}$	$G_{TAX_{AVG}}$		
				Mean	**Max**	**StdDev**
		2012-1	315	0.35	0.07	30.18	3.17
		2012-2	189	0.27	0.09	8.66	2.96
		2012-3	153	0.86	0.36	8.70	2.79
		2012-4	16	0.19	0.75	5.51	2.41
		2013-1	193	1.08	0.35	10.33	2.50
		2013-2	178	0.14	0.05	5.75	2.04
		2013-3	33	-0.35	-0.68	4.68	2.58
		2013-4	61	0.41	0.42	5.91	1.95
		2014-1	84	0.34	0.25	10.84	2.28
		2014-2	73	0.28	0.23	4.16	1.77
		2014-3	81	0.10	0.08	10.91	2.40
		2014-4	81	0.13	0.10	7.58	2.14
		2015-1	143	1.05	0.46	9.53	2.02
		2015-2	60	-0.25	-0.26	7.11	2.41
		2015-3	50	0.56	0.74	7.35	2.63
		2015-4	66	0.06	0.06	3.91	1.57
		2016-1	79	0.17	0.13	9.17	3.59
		2016-2	350	-0.64	-0.12	12.62	2.59
		2016-3	102	0.60	0.39	8.04	2.42
		2016-4	118	1.57	0.85	10.26	2.56
		2017-1	260	0.60	0.15	9.44	1.84
		2017-2	25	0.03	0.07	4.28	1.85
		2017-3	85	-0.01	-0.01	3.72	1.10
		2017-4	43	0.14	0.20	4.61	1.48

Tabella A.3: Guadagni trimestrali migliori simulazioni di ogni anno

Appendice B

Documentazione del programma realizzato

La metodologia proposta in questa tesi, come indicato nel capitolo 5, si è basata su un'estensione del framework BeSt, indirizzata alla simulazione e configurazione automatica di modelli di classificazione. Dal punto di vista pratico, l'estensione del framework è consistita nello sviluppo di un programma con cui realizzare i processi di simulazione e valutazione dei classificatori. In questa appendice si completa la spiegazione della metodologia, documentando sinteticamente gli aspetti più rilevanti del programma realizzato.

Caratteristiche generali

Il programma che implementa il simulation e l'evaluation process, descritti nella sezione 5.3, è un'applicazione `Java` che mantiene ed organizza i dati su due database `SQLite` e che esegue algoritmi di data mining attraverso le librerie del software open source `RapidMiner 5.3`.

L'applicazione è strutturata sui più moduli che possono essere eseguiti indipendentemente oppure richiamati implicitamente all'interno del flusso operativo principale, che prevede:

1. **Simulazione** di modelli di classificazione
   (applicando i *componenti modificati* del sistema BeSt):
   1.1. download dei prezzi storici da Yahoo! Finance;
   1.2. pre-processing / data cleansing;
   1.3. training e applicazione del modello;
   1.4. salvataggio dati della simulazione nel `Simulation DB`.
2. Estrazione **statistiche** di performance delle simulazioni
   (processo `ETL` da `Simulation DB` a `Analysis DB`).
3. **Valutazione** / **self-tuning** dei modelli di classificazione
   (previsione dei rendimenti futuri tramite modello di regressione).

Database

I dati elaborati dall'applicazione , come già indicato, sono mantenuti all'interno di due database `SQLite`. Il `Simulation DB` è soggetto ad un carico di lavoro principalmente transazionale (OLTP), in quanto registra continuamente i risultati delle simulazioni eseguite sui prezzi storici degli stock. Mentre l'`Analysis DB`, creato automaticamente dal processo `ETL`, è dedicato all'aggregazione dei dati statistici relativi alle prestazioni delle simulazioni svolte.

Il diagramma in figura B.1 rappresenta lo schema delle tabelle e delle viste del `Simulation DB`, mentre la tabella B.1 descrive il significato di ogni entità.

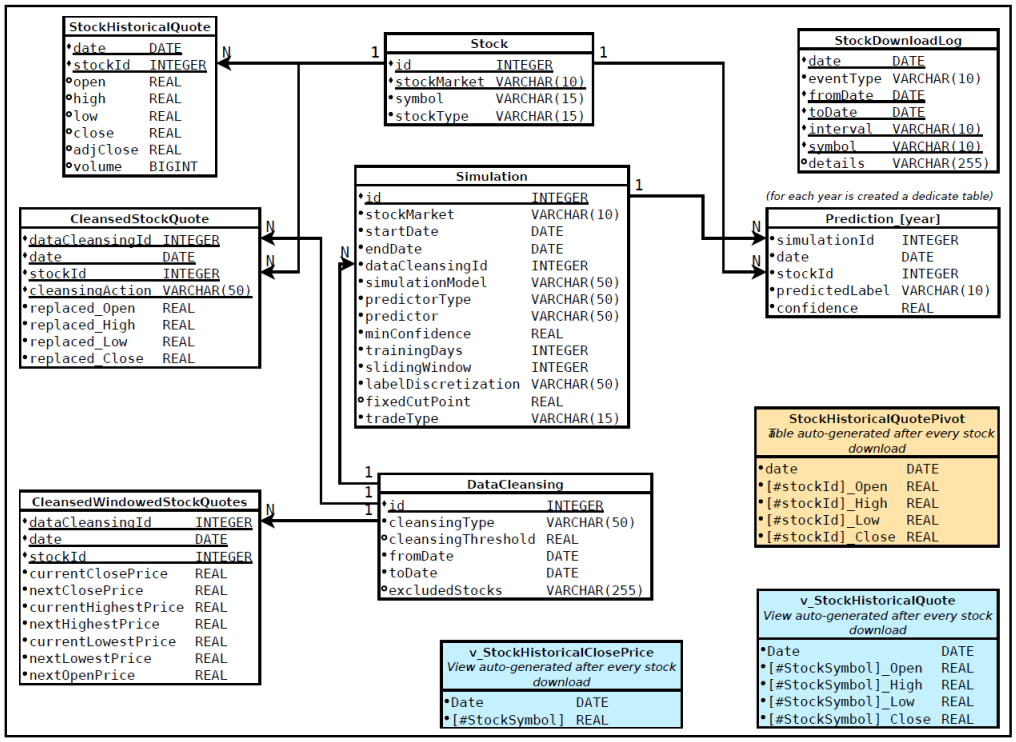

Figura B.1: Diagramma schema `Simulation DB`

Tabella / Vista	Descrizione
`Stock`	Simboli rappresentativi delle azioni. Un simbolo piò riferirsi a uno stock a all'indice di mercato di riferimento.
`StockDownloadLog`	Log eventi di download dei prezzi storici delle azioni da YAHOO! FINANCE.
`StockHistoricalQuote`	Prezzi OLHC storici delle azioni.

Tabella B.1 – continua

Tabella B.1 – continuazione

Tabella / Vista	Descrizione
`StockHistoricalQuotePivot`	Tabella generata programmaticamente[1] disponendo i prezzi di tutti gli stock della tabella `StockHistoricalQuote` per data.
`DataCleansing`	Parametri usati per il pre-processing dei prezzi storici: gestione dei *missing values* e *outliers* (rif. 5.2.1).
`CleansedWindowedStockQuotes`	Prezzi storici delle azioni pre-processati.
`Simulation`	Parametri della simulazione (rif. 5.3.1).
`Prediction_[year]`	Raccomandazioni estratte delle simulazioni (combinazioni $\{algoritmo, \{parametri\}\}$) relative ai prezzi storici dell'anno `[year]`.
`v_StockHistoricalClosePrice`	Vista dei prezzi OLHC utile alla fase di ETL per calcolare le soglie di stop loss e i risultati attesi di per ogni previsione estratta dai modelli simulati.
`v_StockHistoricalQuote`	Vista dei prezzi storici in cui si visualizza il simbolo dell'azione invece che il campo `stockId`.

Tabella B.1: Descrizione tabelle e viste del `Analysis DB`

Procedendo in modo analogo al primo database, per l'`Analysis DB` si rappresenta lo schema delle entità nel diagramma in figura B.2 e si riporta la descrizione nella tabella B.2.

Tabella	Descrizione
`Simulation`	Tabella che riassume tutti i parametri di ogni simulazione eseguita.
`StockQuotes`	Tabella che riassume tutti i prezzi storici considerati nelle simulazioni e, per ogni stock in ogni giorno, il valore corretto dell'etichetta (`UP`/ `DOWN`).
`PotentialOutcome_[year]`	Metriche di guadagno di ogni simulazione eseguita sui prezzi storici dell'anno `[year]`.
`SimulationStats`	Sintesi delle metriche di guadagno di tutte le simulazioni aggregate per anno (quelle commentate nelle sezioni 6.1.2 e 6.1.3).

Tabella B.2 – continua

[1] È necessario auto-generare una tabella perché `SQLite` non supporta l'istruzione `pivot`.

Tabella B.2 – continuazione

Tabella	Descrizione
`SimStats_[timePeriod]`	Sintesi delle metriche di guadagno, senza considerare lo stop loss, di tutte le simulazioni aggregate per `[timePeriod]` e usate per il training e l'applicazione del modello di regressione nella fase di *self-tuning*.
`SimStatsSL_[timePeriod]`	Tabella con la stessa funzione di `SimStats_[timePeriod]`, ma considerando i guadagni a fronte dell'impostazione dello stop loss.

Tabella B.2: Descrizione tabelle e viste dell'`Analysis DB`

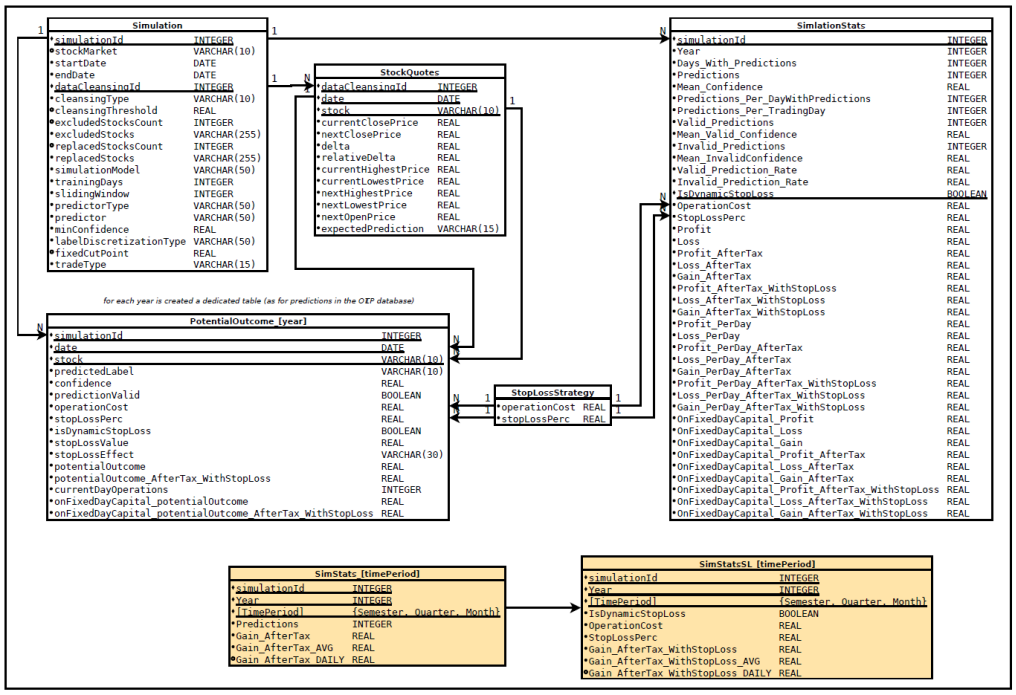

Figura B.2: Diagramma schema `Analysis DB`

NB. Nella descrizione dell'`Analysis DB` si vuole enfatizzare il fatto che l'applicazione realizzata è utilizzabile per l'analisi dei guadagni in diversi periodi temporali. La sperimentazione svolta ha infatti verificato l'efficacia della funzione di self-tuning aggregando le metriche di guadagno delle simulazioni per mese, trimestre e semestre. I dati che poi sono risultati più efficaci per il training del modello di regressione, come indicato nella sezione 6.2, sono stati quelli aggregati per trimestre senza considerare lo stop loss.

Package e classi

Le funzioni svolte dall'applicazione sono distribuite su diversi `package Java`. Quasi tutti i package contengono al loro interno una classe con un metodo `main()` statico, che permette di utilizzare una determinata funzionalità come un programma indipendente (ad esempio per avviare la sincronizzazione dei prezzi storici da YAHOO! FINANCE, lanciare un processo `RapidMiner` o eseguire una simulazione con specifici parametri) direttamente dalla linea di comando. Di seguito si descrivono brevemente i vari package / moduli realizzati:

- **`it.polito.s202264.besty`**
  `Besty` rappresenta la radice della gerarchia di package utilizzata nel progetto. Al suo interno sono definite le classi che controllano i componenti del framework e la relativa configurazione. Tra queste le più rilevanti sono:
  - **`Besty.java`**
    La classe `Besty` costituisce il vero e proprio *entry point* del programma, da cui è possibile avviare il simulation e l'evaluation process.
    - Il processo di simulazione è delegato ad un package dedicato, che implementa il componente `simulator` del framework.
    - Il processo di valutazione è invece gestito direttamente nella classe `Besty`, che definisce i metodi per avviare i processi `RapidMiner` relativi alla fase di `ETL` e a quella di self-tuning.
  - **`Configuration.java`**
    Questa classe carica la configurazione del sistema dal file **`config.xml`** e definisce i metodi di accesso ai parametri di configurazione.

- **`it.polito.s202264.besty.rmtools`**
  Le classi di questo package definiscono *metodi wrapper* per semplificare l'accesso alle operazioni implementate nella libreria **`rapidminer.jar`** (recuperata dai sorgenti di `RapidMiner 5.3`).
  - **`RmExecutor.java`**
    La classe `RmExecutor` serve principalmente per pilotare l'esecuzione di processi `RapidMiner`, ma fornisce anche i metodi per configurare i `repository` e le connessioni ai database `SQLite` nell'ambiente `RapidMiner`.
    - Un processo `RapidMiner` può essere eseguito specificando il percorso del file **`.rmp`**, la `macroMap` contenente i parametri (macro) del processo e eventualmente il percorso di un file di log.
    - La classe può anche essere richiamata direttamente da linea di comando, separatamente dal resto del framework.

- **`it.polito.s202264.besty.simulator`**
  Il package `simulator` è dedicato a tutte le operazioni del simulation process:

dalla preparazione dei dati alla scrittura nel `Simulation DB` delle raccomandazioni fornite dai classificatori testati.

- **`SimulatorV2.java`**
  `SimulatorV2` è la classe che implementa il processo descritto dallo pseudocodice 3 nella sezione 5.3.1.
  - Il processo di simulazione accede al `Simulation DB` sfruttando l'oggetto `SQLiteManager` e pilota la generazione e applicazione dei modelli di classificazione tramite `RmExecutor`.
  - Il `main()` di questa classe può essere richiamato direttamente da linea di comando per avviare una simulazione con specifici parametri.
  - `SimulatorV2` è a sua volta controllato da `Besty`.

- **`it.polito.s202264.besty.sqlitemgr`**
  In questo package è contenuta la classe **`SQLiteManager.java`**, che regola l'accesso ai due database `SQLite` e semplifica le operazioni di lettura e scrittura da parte degli altri componenti del framework.

- **`it.polito.s202264.besty.stockquotes`**
  Questo package è dedicato alle funzioni di data gathering. La principale classe, **`StockDownloader.java`**, proviene dal sistema BeSt originale ed è stata integrata nel sistema esteso affinché potesse memorizzare i prezzi storici delle azioni direttamente nel `Simulation DB`. Anche questa classe può essere usata in modo indipendente dalle altre.

- **`it.polito.s202264.besty.stockquotes.yahoofinance`**
  Il package `yahoofinance` è strettamente legato all'implementazione delle API di Yahoo! Finance ed è stato separato da **`stockquotes`**, perché le sue classi sono state estratte dal progetto open source *YahooFinanceAPI*[44].

- **`it.polito.s202264.besty.configuration.jaxb`**
  Le classi di questo package sono generate automaticamente a partire dal `XML Schema` **`xsd/config.xsd`**, associato al file di configurazione **`config.xml`**, usando `JAXB`[45]. L'impiego principale di queste classi è quello di semplificare l'accesso e la modifica dei parametri di configurazione globale del sistema.

File di configurazione globale - `config.xml`

Il file **`config.xml`**, citato nella descrizione della classe `Configuration` e del package generato tramite `JAXB`, è definito `global configuration file` in quanto svolge la funzione di *central store* (contenitore centralizzato) dei parametri usati da ogni componente del framework.

La configurazione è suddivisa in sezioni che fanno riferimento alle specifiche funzionalità del sistema BeSt esteso, secondo l'organizzazione esposta di seguito.

Impostazioni database

La sezione relativa alle impostazioni dei database definisce i seguenti parametri:

- i percorsi relativi alla posizione dei *data file*;
- i percorsi degli *script SQL* da usare per creare i database da zero;
- le proprietà del *driver JDBC* da usare per connettersi ai database `SQLite`.

```
<databaseSettings
    dbFilePath="data/sqlite_db/bestyDev.db"
    oltpSchemaScriptPath="script/sql/create_oltp_db_schema.sql"
    analysisDbFilePath="data/sqlite_db/bestyDev-Analysis.db"
    analysisSchemaScriptPath="script/sql/create_analysis_db_schema.sql"
    loggingDirPath="log/">

    <jdbcProperties
        name="SQLite"
        driverClass="org.sqlite.JDBC"
        driverJarFilePath="lib/sqlite-jdbc-3.7.2.jar"
        urlPrefix="jdbc:sqlite:"
    />
</databaseSettings>
```

Codice B.1: Sezione di configurazione dei database

In riferimento alla sezione riportata nel blocco B.1, il file associato all'attributo `dbFilePath` appartiene al `Simulation DB`, mentre quello specificato per l'attributo `analysisDbFilePath` appartiene all'`Analysis DB`.

Impostazioni RapidMiner

In questa sezione si definiscono tutti i parametri relativi all'esecuzione dei processi `RapidMiner` tramite il componente `RmExecutor`. Facendo riferimento alla configurazione contenuta nel blocco di codice B.2, di seguito si riportano alcune note relative agli elementi più importanti:

- Gli attributi `dbConnectionName` e `analysisDbConnectionName`, definiti nell'elemento `<rapidminerSettings>`, sono usati dal componente `RmExecutor` per configurare le connessioni ai database `SQLite` nel contesto di esecuzione dei processi `RapidMiner`.
  - Le connessioni ai database sono definite usando le informazioni contenute nell'elemento `<databaseSettings>`, descritto nella precedente sezione della configurazione.
- Gli elementi `<rmRepository>` definiscono i *nomi* e i *percorsi* dei repository (cartelle del file system locale) dove sono contenuti i file `.rmp` dei processi `RapidMiner` e i file dei dati letti e scritti dai processi.
  - Il repository `"BestyDev-Process"` è dedicato ai processi (file `.rmp`) usati dal sistema BeSt esteso.

  - Quando il componente `RmExecutor` inizializza l'ambiente di esecuzione di `RapidMiner`, oltre a configurare le connessioni ai database, configura anche i repository locali.

- Gli elementi `<rmProcess>`, definiscono *nome*, *percorso* e *struttura* dei processi `RapidMiner` al fine di facilitarne l'accesso e l'esecuzione da parte dei componenti `RmExecutor` e `Simulator`.
  - L'attributo `name` è la chiave identificativa di ogni processo, mentre l'attributo `location` specifica il percorso assoluto del file `.rmp` [2].
  - I sotto-elementi `<switchBlock>` sono usati per rappresentare una *condizione parametrizzata* che, in base agli argomenti specificati nelle *macro* di input al processo, permette di modificare il flusso delle operazioni eseguite dal processo. Per questi elementi è definita una relazione con gli elementi `<rmSwithBlock>`, che specificano quali operazioni possono essere scelte alternativamente nella fase di esecuzione di un processo.

- Gli elementi `<rmSwitchBlock>` identificano la presenza di un operatore `Select Subprocess`[46] all'interno di un processo `RapidMiner` al fine di mappare i sotto-processi definiti per tale operatore con le classi `Java` relative ala configurazione del framework. Nello specifico, questa funzione viene sfruttata principalmente per il processo di simulazione, dove occorre eseguire sempre la stessa sequenza di operazioni (ovvero lo stesso processo `RapidMiner`) variando solamente i parametri di costruzione del modello.
  - I parametri di simulazione che richiedono la definizione di operatori esclusivi in `RapidMiner` sono `LabelDiscetization`, `Algorithm` e `TradeType`. Per ognuno di questi parametri viene quindi creato un operatore `Select Subprocess`, le cui alternative sono mappate rispettivamente negli *switch* `<rmLabelDiscretization>`, `<rmPredictor>` e `<rmTradeType>`.
  - Per ogni *switch* (sotto-elemento di `<rmSwitchBlock>`) deve essere specificato l'attributo `switchNumber`, che corrisponde esattamente con l'intero progressivo associato al sotto-processo contenuto nell'operatore `Select Subprocess`.
  - La parametrizzazione di questi *switch* nel file di configurazione permette anche di abilitare o disabilitare i sotto-processi attraverso l'attributo booleano `enabled`. Tramite questo attributo sono stati filtrati gli algoritmi di classificazione da utilizzare nella sperimentazione da 61 a 20, come indicato nella sezione 6.1.1 e nella tabella A.1.

[2] Per svincolare la dipendenza dal repository locale, si preferisce usare il percorso assoluto per localizzare i file dei processi `RapidMiner`. I percorsi relativi ai repository, definiti con `<rmRepositoty>`, sono intesi per essere usati solo all'interno di processi che richiamano altri processi.

```
<rapidminerSettings
    numberOfThreads="6"
    dbConnectionName="Besty_DB"
    analysisDbConnectionName="Besty_DB_Analysis"
    loggingDirPath="log/">

    <rmRepository
        name="BestyDev-Process"
        location="src/rm_process/"
    />

    . . .

    <rmProcess
        name="Simulation-05"
        type="SIMULATE_MODEL"
        location="src/rm_process/simulation/SimulationModel-05.rmp">
        <switchBlock name="SWITCH_LABEL_DISC_TYPE" />
        <switchBlock name="SWITCH_CLASSIFICATION_MODEL" />
        <switchBlock name="SWITCH_TRADE_TYPE" />
    </rmProcess>

    . . .

    <rmSwitchBlock
        name="SWITCH_CLASSIFICATION_MODEL">
        . . .
        <rmPredictor name="Default_Model" type="Classification"
            numericalLabelSupported="true" binominalLabelSupported="true"
            polynominalLabelSupported="true" numericalAttributesSupported="true
            " binominalAttributesSupported="true"
            polynominalAttributesSupported="true" missingValuesSupported="true"
             switchNumber="2" enabled="false" />
        <rmPredictor name="Naive_Bayes" type="Classification"
            numericalLabelSupported="false" binominalLabelSupported="true"
            polynominalLabelSupported="true" numericalAttributesSupported="true
            " binominalAttributesSupported="true"
            polynominalAttributesSupported="true" missingValuesSupported="true"
             switchNumber="3" enabled="true" />
        . . .
    </rmSwitchBlock>

    . . .

</rapidminerSettings>
```

Codice B.2: Sezione di configurazione `RapidMiner`

Impostazioni delle simulazioni

Nonostante il nome, questa sezione include le impostazioni relative sia al processo di simulazione che a quello di valutazione. Considerando l'esempio riportato nel blocco di codice B.3, si spiegano di seguito gli elementi principali:

- Gli attributi `simulationDirPath` e `loggingDirPath` definiscono le cartelle temporanee in cui il componente `Simulator` scriverà dati e log delle simulazioni.
- L'elemento `<defaultParameters>` inizializza le principali variabili con dei valori predefiniti.
  - L'attributo `stock_Market` specifica l'indice di mercato composto dalle azioni considerate per le simulazioni.
  - Gli attributi `dataCleansingType` e `dataCleansingThreshold` specificano i parametri del processo di pre-processing (data cleansing) dei prezzi storici, secondo quanto indicato nella sezione 5.2.1. I nomi delle azioni inclusi nei sotto-elementi `excludedStock` sono automaticamente esclusi dalle simulazioni, senza esse pre-processati.
- L'elemento `<stockMarket>` specifica nome (`name`) e simbolo (`indexSymbol`) dell'indice di mercato considerato per le simulazioni e il percorso del file contenente la lista delle azioni per cui elaborare i prezzi storici (`symbolsFilePath`). Oltre al FTSE MIB sono indicati anche gli indici S&P 500 e NASDAQ-100, solamente a titolo di esempio, per indicare che è supportata la scelta di altri prezzi storici.
- Gli elementi `<paramterRange>` sono quelli che controllano quante combinazioni di parametri devono essere testate dal simulation process, secondo quanto indicato nella sezione 5.3.1.
- L'elemento `simResultsEvaluation` è quello relativo alla fase di evaluation e alla funzione di *self-tuning*.
  - L'attributo `timePeriod` specifica l'intervallo di tempo da considerare per il self-tuning.
  - L'attributo `evalAttribute` specifica la metrica di guadagno che dovrà essere prevista del modello di regressione.
  - L'attributo `trainingSetSize` specifica quanti intervalli di tempo usare per il training del modello di regressione.
  - L'attributo booleano `stopLoss` abilita o disabilita il self-tuning per le metriche di guadagno calcolate in relazione alla strategia di stop loss.
  - Gli attributi `resultDirPath` e `resultFilePrefix` specificano rispettivamente la cartella in cui scrivere il risultati ottenuti dal modello di regressione e il prefisso da usare per ogni file creato.
  - A differenza delle altre proprietà, i sotto-elementi `<stopLossStrategy>` e l'attributo `operationCost` sono legati esclusivamente al calcolo delle

prestazioni ottenute dalle simulazioni. Il valore percentuale specificato dall'attributo `stopLossPerc` indica come deve essere calcolata la soglia di stop loss, secondo quanto indicato nella sezione 5.3.3. Il valore di `operationCost` specifica invece la commissione da calcolare su ogni raccomandazione eseguita.

```
<simulationSettings
    simulationDirPath="data/simulation"
    loggingDirPath="log/">

    <defaultParameters
        stock_Market="FTSE_MIB"
        dataCleansingType="LIMITED_MISSINGS_VALUES_AND_OUTLIERS"
        dataCleansingThreshold="1.0">
        <excludedStock name="BMPS.MI" stock_Market="FTSE_MIB" />
    </defaultParameters>

    <stockMarket
        name="FTSE_MIB"
        indexSymbol="FTSEMIB.MI"
        symbolsFilePath="data/stocks-historical-quotes/ftsemib-list.txt" />
    <stockMarket name="S_P_500" indexSymbol="S&amp;amp;P 500" />
    <stockMarket name="NASDAQ_100" indexSymbol="NASDAQ-100" />

    <excludedStock name="BMPS.MI" stock_Market="FTSE_MIB" />

    <parameterRange paramName="MinConfidence" min="0.5" max="0.5" step="1" />
    <parameterRange paramName="TrainingDays" min="10" max="30" step="5" />
    <parameterRange paramName="SlidingWindow" min="2" max="5" step="1" />
    <parameterRange paramName="FixedCutPoint" min="-4.0" max="4.0" step="1" />

    <simResultsEvaluation
      operationCost="0.15"
        timePeriod="QUARTER"
        evalAttribte="GainDaily"
        trainingSetSize="4"
        stopLoss="False"
        resultDirPath="data/"
        resultFilePrefix="selfTuning_">
        <stopLossStrategy stopLossPerc="1.0" />
        <stopLossStrategy stopLossPerc="1.5" />
        <stopLossStrategy stopLossPerc="2.0" />
    </simResultsEvaluation>
</simulationSettings>
```

Codice B.3: Sezione di configurazione delle simulazioni

Impostazioni *data gathering*

Infine, la sezione `<stockDownloaderSettings>` specifica tutti i parametri richiesti per recuperare i prezzi storici delle azioni da Yahoo! Finance.

- L'attributo `downloadDirPath` definisce la cartella in cui scrivere i file `.csv` contenenti i prezzi storici. Si tratta di un parametro facoltativo inserito per rendere compatibile la versione modificata del componente con quella originale (tutti i dati storici degli stock sono aggiornati e mantenuti nel `Simulation DB` e possono essere esportati su file in qualsiasi momento).
- Gli attributi `stockListFileName`, `stockClosePricesFileName` e `stockQuotesFileName` definiscono i nomi dei file dove esportare, rispettivamente, la lista dei simboli degli stock da considerare per le simulazioni, il dataset dei prezzi di chiusura giornalieri di ogni azione e il dataset con l'insieme completo dei prezzi OHLC (apertura, massimo, minimo, chiusura).
  - I primi due file sono usati nel processo di simulazione, mentre il file contenente i prezzi OHLC è usato per i calcoli legati allo stop loss (rif. 5.3.3).
- Dal momento che ogni simulazione inizia raccomandando gli stock da considerare il primo giorno dell'anno, per addestrare i classificatori è necessario recuperare i prezzi storici delle giornate di trading precedenti al primo dell'anno: a questo scopo l'attributo `daysBeforeFirstDateToDownload` indica quanti sono i giorni precedenti da considerare.
  - Nella configurazione riportata in B.4 questo parametro è fissato a 60 per assicurarsi di disporre, con ampio margine, dei dati di almeno 30 giorni di trading prima della fine dell'anno.
- Analogamente, l'attributo `daysAfterLastDateToDownload` specifica i giorni aggiuntivi da considerare dopo l'ultima giornata di trading per cui estrarre raccomandazioni.
  - Questo parametro è utilizzato dal componente `StockDownloader` per verificare se i prezzi storici scaricati includono effettivamente l'ultima giornata di trading dell'anno (se si recuperano dati in una data successiva e questa appartiene al nuovo anno, allora si ha la conferma che la data precedente corrisponde effettivamente all'ultimo giorno di trading).
- L'elemento `<yahooFinanceAPIParams>` contiene i parametri specifici per il download di prezzi storici da Yahoo! Finance.
  - È utile modificare questi parametri solo nel caso in cui le API di Yahoo! Financesubiscano delle modifiche o non funzionino correttamente.

```
<stockDownloaderSettings
    downloadDirPath="data/stocks-historical-quotes"
    stockListFileName = "list.csv"
    stockClosePricesFileName = "prezzi.csv"
```

```
    stockQuotesFileName = "prezzi_stats.csv"
    daysBeforeFirstDateToDownload="60"
    daysAfterLastDateToDownload="10">

    <yahooFinanceAPIParams
        quotesCsvBaseUrl="http://download.finance.yahoo.com/d/quotes.csv"
        historicalQuotesBaseUrl="https://query1.finance.yahoo.com/v7/finance/
            download/"
        historicalQuotesCrumbUrl="https://query1.finance.yahoo.com/v1/test/
            getcrumb"
        historicalQuotesScrapeUrl="https://finance.yahoo.com/quote/%5EGSPC/
            options"
        connectionTimeout="10000"
        maxProtocolRedirectionCount="5"
        dateFormat="yyyy-MM-dd"
        csvDelimiter=","
    />
</stockDownloaderSettings>
```

Codice B.4: Sezione di configurazione `StockDownloader`

Setup e esecuzione

0. Requisiti
   - Windows OS o Linux OS
   - `Java SE Development Kit v7` (`JDK-7`)
   - `Java SE Runtime v7` o maggiore (> `JRE-7`)

1. Copiare i file del progetto `Java` in una cartella locale.

2. Eseguire i seguenti comandi da linea di comando

```
cd <percorso_cartella_progetto>
ant build
ant config-rm
```

3. Modificare il seguente script sostituendo il valore associato alla variabile `dirBase` con il percorso `<percorso_cartella_progetto>`:
   - `./script/bash/runBESTY.sh` (Linux)
   - `powershell.exe .\script\ps\runBESTY.ps1` (Windows)

4. Verificare ed eventualmente modificare i parametri contenuti del file di configurazione globale `config.xml`.

5. Lanciare lo script per eseguire il programma secondo i parametri indicati nel file di configurazione.

Bibliografia

[1] SIGKDD - ACM (Association for Computer Machinery), *Data Mining Curriculum*, kdd.org

[2] Max Bramer, *Principles of Data Mining*, Springer, 3rd ed. (2016), pp. 2

[3] Elena Baralis, Paolo Garza, *Data Mining: Introduction* (dispense), DataBase and Data Mining Group, Politecnico di Torino

[4] Elena Baralis, *Data Mining: Data preprocessing* (dispense), DataBase and Data Mining Group, Politecnico di Torino

[5] Elena Baralis, Silvia Chiusano, *Data Mining: Association Rules fundamentals* (dispense), DataBase and Data Mining Group, Politecnico di Torino

[6] Elena Baralis, *Data Mining: Classification fundamentals* (dispense), DataBase and Data Mining Group, Politecnico di Torino

[7] Elena Baralis, Tania Cerquitelli, *Data Mining: Clustering fundamentals* (dispense), DataBase and Data Mining Group, Politecnico di Torino

[8] Matteo Golfarelli, *Introduzione al Data Mining* (dispense), Alma Mater Studiorum, Università di Bologna

[9] Towards Data Science, *Support Vector Machine vs Logistic Regression*, towardsdatascience.com

[10] John J. Murphy, *Technical analysis of the financial markets*, New York Institute of Finance (1999)

[11] Wikipedia, *Strumento finanziario*, wikipedia.org

[12] Wikipedia, *Stock Exchange*, wikipedia.org

[13] Il Giorno, *Torna la Borsa Prima di internet: grida e gesti per comprare azioni*, www.ilgiorno.it

[14] Wikipedia, *Azione*, wikipedia.org

[15] Wikipedia, *Indice azionario*, wikipedia.org

[16] SoldiOnline, *La vita del day trader non è affatto divertente*, www.soldionline.it

[17] Money.it, *Trading meccanico vs trading discrezionale: come funzionano e quale scegliere*, www.money.it

[18] Wikipedia, *Analisi fondamentale*, wikipedia.org

[19] BorsaMercato, *Analisi tecnica: definizione e "pilastri"*, www.borsamercato.com

[20] Wikipedia, *Analisi tecnica*, wikipedia.org

[21] TradingMania, *Analisi Fondamentale*, tradingmania.it

[22] Traderpedia, *Aspirante Trader Quantitativo*, wikipedia.org

[23] Wikipedia, *Trading online - Tipologie principali*, wikipedia.org

[24] IG Group, *Analisi candlestick*, ig.com

[25] Wikipedia, *Logica fuzzy*, wikipedia.org

[26] Qing Li, Tiejun Wang, Qixu Gong, Yuanzhu Chen, Zhangxi Lin, Sa-kwang Song, *Media-aware quantitative trading based on public Web information*, Decision Support System 61 (2014) 93-105

[27] Tai-liang Chen, Feng-yu Chen, *An intelligent pattern recognition model for supporting investment decisions in stock market*, Information Sciences 346-347 (2016) 261-274

[28] Rodrigo Naranjo, Javier Arroyo, Matilde Santos, *Fuzzy modeling of stock trading with fuzzy candlesticks*, Expert Systems With Applications 93 (2018) 15-27

[29] Lamartine Almeida Teixeira, Adriano Lorena Inàcio de Oliveira, *A method for automatic stock trading combining technical analysis and nearest neighbor classification*, Expert System Application 37 (2010) 6885-6890

[30] Kamil Zbikowski, *Using volume weighted support vector machines with walk forward testing and feature selection for the purpose of creating stock trading strategy*, Expert System Application 42 (2016) 1797-1805

[31] Jonathan L. Ticknor, *A bayesian regularized artificial neural network for stock market forecasting*, Expert System Application 40 (2013) 5501-5506

[32] Lili Wang, Zitian Wang, Shuai Zaho, Shaohua Tan, *Stock market trend prediction using dynamicl Bayesian factor graph*, Expert Systems with Applications 42 (2015) 6267-6275

[33] Josè Manuel Berutich, Francisco Lòpez, Francisco Luna, David Quintana, *Robust technical trading strategies using GP for algorithmic portfolio selection*, Expert System Application 46 (2016) 307-315

[34] Youngmin Kim, Wonbin Ahn, Kyong Joo Oh, David Enke, *An intelligent hybrid trading system for discovering trading rules for the futures market using rough sets and genetic algorithms*, Applied Soft Computing 55 (2017) 127-140

[35] Myoung-Jong Kim, Sung-Hwan Min, Ingoo Han, *An evolutionary approach to the combination of multiple classifiers to predict a stock price index*, Expert Systems with Applications 31 (2006) 241-247

[36] William Leigh, Russell Purvis, James M. Ragusa, *Forecasting the NYSE composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support*, Decision Support Systems 32 (2002) 361-377

[37] Jigar Patel, Sahil Shah, Priyank Thakkar, K Kotecha, *Predicting stock market index using fusion of machine learning techniques*, Expert System Application 42 (2015) 2162-2172

[38] Chih-Fong Tsai, Yuah-Chiao Lin, David C. Yen, Yan-Min Chen, *Predicting stock returns by classifier ensembles*, Applied Soft Computing 11 (2011) 2452-2459

[39] Vasant Dhar, *Prediction in Financial Markets: The Case for Small Disjuncts*, ACM Transactions on Intelligent Systems and Technology, Vol. 2, No. 3, Article 19 (2011) 19:1-19:22

[40] Elena Baralis, Luca Cagliero, Tania Cerquitelli, Paolo Garza, Fabio Pulvirenti, *Discovering profitable stocks for intraday trading*, Information Sciences 405 (2017) 91-106

[41] RapidMiner Documentation, *Discretize by Binning*, docs.rapidminer.com

[42] RapidMiner Documentation, *Nominal to Numerical*, docs.rapidminer.com

[43] RapidMiner Documentation, *Polynominal Regression*, docs.rapidminer.com

[44] Wikipedia, *YahooFinanceAPI GitHub project*, github.com

[45] Java Documentation, *Java Architecture for XML Binding (JAXB)*, oracle.com

[46] RapidMiner Documentation, *Select Subprocess*, docs.rapidminer.com