POLITECNICO DI TORINO

Corso di Laurea Magistrale
in INGEGNERIA INFORMATICA

Tesi di Laurea Magistrale

Sviluppo e validazione di un trading system quantitativo basato su tecniche di regressione

Relatori

prof. Luca Cagliero
prof. Paolo Garza

firma dei relatori
.....
.....

Candidato

Pietro Fardella

firma del candidato
.....

A.A. 2017/2018

Questo lavoro rappresenta la fine di un percorso durato tre anni, un percorso pieno non solo di difficoltà e momenti difficili, ma anche di momenti significativi.

Ci tengo a ringraziare i miei relatori, il professor Cagliero ed il professor Garza, per la loro competenza e disponibilità sempre dimostrate.

Un sentito ringraziamento va a tutti i miei amici e compagni di avventura, con cui ho condiviso difficoltà, ma soprattutto momenti di felicità. Non vi nomino singolarmente, ma sappiate che il vostro supporto è stato essenziale, chi più chi meno.

Ringrazio infinitamente anche colei che mi ha instillato nella mente il pensiero di cambiare vita, di affrontare un nuovo percorso a Torino. Uscire dalla propria comfort zone è fondamentale per crescere e migliorare, mi ci è voluto del tempo per capirlo ma ho deciso comunque di buttare anima e corpo in questa nuova avventura. Non ti nomino, ma sappi che apprezzo il supporto che ricevo ogni giorno da parte tua.

Un enorme ringraziamento infine va alla mia famiglia, che mi ha sempre supportato in tutto con tutti i mezzi possibili. Anche se il "protagonista" del percorso sono io, avrete sempre un contributo fondamentale nei miei successi, presenti e futuri, e di questo vi sarò sempre riconoscente.

Indice

1. Introduzione

Il trading è il processo mediante il quale investitori privati o operatori finanziari effettuano la compravendita di strumenti finanziari, quali azioni, obbligazioni e derivati sui vari mercati finanziari, tipicamente utilizzando piattaforme di trading online, con l'obiettivo di realizzare un profitto.

Una classe di strategie d'investimento è il trading quantitativo, che comprende tutte le strategie basate su analisi quantitative, e, tipicamente, include strategie d'investimento automatizzate e a corto termine.
Tra le tante strategie d'investimento a corto termine attuabili, una delle più utilizzate è la pratica del cosiddetto day trading, che consiste nel comprare e vendere i prodotti finanziari in un arco di tempo ridotto, solitamente inferiore alla giornata di borsa.

Le azioni, o stock, sono tra gli strumenti finanziari più utilizzati per il trading. Esistono centinaia di azioni diverse, elencate nei mercati azionari di tutto il mondo, dunque la scelta delle azioni giuste su cui investire rappresenta un'operazione non banale.
Poichè i prezzi delle azioni variano continuamente, gli investitori devono costantemente monitorare l'andamento degli stessi, al fine di intercettare e prevedere i trend di mercato futuri.
Tale compito è tradizionalmente supportato da due tipi di analisi, quella tecnica e quella fondamentale.
L'analisi tecnica[1] è una disciplina che viene impiegata per valutare strumenti finanziari al fine di identificare opportunità di trading, utilizzando soltanto statistiche derivanti dai movimenti azionari. Secondo questo tipo di analisi valgono le seguenti assunzioni:

- i mercati sono efficienti e presentano valori che riflettono tutti i fattori in grado di influenzare il prezzo di un titolo. In base a questa assunzione, il prezzo di un'azione in un dato istante di tempo rappresenta il valore effettivo della stessa;

- i movimenti dei prezzi di mercato non sono casuali, ma seguono pattern e trend che tendono a ripetersi nel tempo. Dunque è possibile identificare tali pattern e trarne profitto.

L'analisi fondamentale[2], differentemente, cerca di valutare il valore intrinseco di uno strumento finanziario esaminando fattori di diverso tipo, quali microeconomici (es. la condizione finanziaria della società, la gestione della stessa) e macroeconomici. In questo modo, gli analisti cercano di determinare lo stato di salute di una compagnia così da confrontarlo con il prezzo delle sue azioni, al fine di determinare se tali azioni sono sopravvalutate o sottovalutate.

L'obiettivo di questo lavoro di tesi è quello di fornire uno strumento di supporto all'analisi tecnica delle azioni mediante la progettazione e la realizzazione di un trading system basato su metodi predittivi che effettuano regressione, con lo scopo di selezionare da un portafoglio di azioni quelle più promettenti su cui investire nei diversi slot temporali di una giornata di mercato borsistico.
In particolare, al fine di consigliare le migliori azioni su cui investire seguendo la strategia del day trading quantitativo, sono stati analizzati dati del mercato azionario statunitense aventi frequenza di campionamento maggiore di quella giornaliera e si è studiato il comportamento della tecnica di data mining detta regressione, applicandone i principali metodi predittivi e valutando la qualità dei modelli generati al variare del livello di aggregazione dei dati.

Gli algoritmi di regressione fanno parte dei metodi predittivi con apprendimento supervisionato (*supervised learning*) e ipotizzano una relazione funzionale tra il valore di una variabile dipendente continua (detta label) ed una o più variabili indipendenti (dette predittori). In questo contesto, si è cercato di predire il valore del prezzo massimo che acquisisce un determinato stock nel periodo di tempo considerato per il trading sulla base dei dati storici di quella stessa azione, in modo da identificare gli stock più promettenti per l'investimento.

Tali algoritmi, dopo essere stati calibrati nella fase di *learning* (o *training*), permettono agli investitori di automatizzare l'analisi e l'esplorazione dei dati storici delle azioni e, di conseguenza, di estendere l'analisi tecnica ai mercati azionari comprendenti un numero molto elevato di azioni.

Il sistema sviluppato per l'analisi dei dati azionari intraday è costituito dai seguenti moduli:

- raccolta e preparazione dei dati storici delle azioni;
- generazione e calibrazione (*tuning*) dei modelli predittivi;
- identificazione e raccomandazione delle azioni su cui investire.

Il primo modulo è stato realizzato integrando un crawler della *Investitors Exchange API* (RESTful), implementato in linguaggio Java, che permette la raccolta di dati storici finanziari secondo due diverse granularità (giornaliera e per minuto). I dati collezionati vengono poi preprocessati con operazioni di pulizia e aggregazione (nel caso dei dati per minuto secondo unità di tempo inferiori al giorno di mercato) mediante alcuni tool, anch'essi implementati in Java, che eseguono queste ed altre operazioni utili nel contesto considerato.

Il secondo modulo è dedicato all'addestramento di modelli di regressione finalizzati a predire la futura variazione di prezzo delle azioni nell'intervallo successivo a quello corrente. Si sono dunque progettati ed implementati processi appositi per la calibrazione dei parametri propri delle tecniche di regressione considerate e per il learning dei modelli predittivi, valutandone successivamente le performance in termini di errore di predizione. Tali processi sono stati realizzati mediante il supporto del tool di visual workflow design RapidMiner.

L'ultimo modulo ha lo scopo di raccomandare le azioni considerate più promettenti per l'investimento, sulla base delle predizioni generate. È stato implementato mediante uno specifico algoritmo che richiede in input le predizioni del valore del prezzo massimo

degli stock considerati relative al slot temporale (*timeslot*, identificato da un *timestamp*) di intervento sul mercato. Tale algoritmo classifica le azioni in base al valore di una funzione di fitness, volta ad identificare quelle più promettenti per l'investimento. Il fitness è calcolato confrontando i valori predetti con il prezzo di chiusura degli stock dell'intervallo di tempo precedente a quello considerato per il trading, in modo da identificare le variazioni di prezzo maggiori rispetto all'intervallo precedente. Vengono dunque consigliati gli *n* stock con il valore più alto di fitness, se tale valore è più alto di una data soglia di guadagno desiderato.

Una validazione sperimentale dell'efficienza ed efficacia del sistema sviluppato è stata condotta sui dati storici di sei mesi dell'anno 2018 relativi alla borsa americana, considerando le azioni dell'indice *Standard & Poor's 500* (*S&P 500*), il quale comprende le azioni di 500 grandi aziende statunitensi ed è considerato l'indice più significativo del mercato americano; l'analisi si è focalizzata sui seguenti tre aspetti: (i) redditività delle strategie, (ii) comportamento e precisione delle predizioni al variare della strategia adottata ed, infine, (iii) tempistiche e scalabilità del framework presentato.

La campagna sperimentale prevedeva l'uso di tre diverse granularità dei dati (30 minuti, un'ora e due ore) e di una finestra a scorrimento di dimensione fissa che campionasse i dati, raggruppando i dati storici relativi ad alcuni timestamp che precedono quello della predizione (*windowing*).
Gli esperimenti hanno mostrato il comportamento dei metodi predittivi di regressione considerati in base ai tre criteri sopra discussi. Sono state valutate, inoltre, l'influenza della diversa granularità dei dati e della dimensione della finestra di campionamento sulla qualità delle predizioni e sui guadagni possibili derivanti dalla raccomandazione degli stock, scegliendo di raccomandare un solo stock per ciascun periodo temporale d'intervento sul mercato.
I risultati mostrano che il sistema proposto riesce ad ottenere errori di predizione mediamente bassi (inferiori allo 0,5%, valutati con la metrica MAPE) utilizzando dimensioni piccole per la finestra di campionamento (pari ad 1-2 periodi temporali

precedenti) e con granularità ridotte dei dati (30 minuti). Sotto queste condizioni, i modelli predittivi migliori risultano essere la regressione lineare e le SVM con kernel *dot*, con le reti neurali aventi un solo layer nascosto poco peggiori (con un errore superiore di meno dello 0,2%).
In termini di redditività delle strategie, valgono le stesse considerazioni. I profitti maggiori per intervallo, investendo sulle azioni consigliate, si ottengono con le tecniche di regressione lineare e SVM con kernel dot, utilizzando valori piccoli per la dimensione della finestra di campionamento (1-2) e le frequenze di campionamento dei dati più alte (30 minuti). Con queste condizioni, si ottengono profitti medi percentuali che vanno dallo 0,208% allo 0,274% (a seconda che il metodo predittivo scelto sia regressione lineare o SVM dot, rispettivamente) per ciascun periodo temporale di investimento, per quanto riguarda le posizioni di mercato lunghe; relativamente a quelle corte i profitti medi per timeslot risultano essere circa lo 0,4%. Usando i dati aggregati per trenta minuti, il numero elevato di raccomandazioni, circa 400, porta ad un guadagno potenziale complessivo significativo su tutto il periodo di esperimenti (maggiore del +70%).
Per quanto riguarda le tempistiche necessarie al framework per operare, si è dimostrato che tutte le strategie sono potenzialmente attuabili, posto che si effettui il training dei modelli nei momenti di chiusura del mercato azionario (es. a fine giornata).

Come sviluppi futuri, è auspicabile l'applicazione di tecniche simili per valutare gli investimenti su altri mercati azionari, quali il *FTSE MIB* (borsa italiana), o sui mercati delle valute, quali il *Foreign Exchange* (*Forex*) ed il mercato delle *cryptovalute*.
Sarebbe parecchio d'interesse, inoltre, integrare il trading system con un modulo che predica il valore del prezzo minimo delle azioni per gli stessi intervalli, in modo da poter calcolare il rapporto *risk*/*reward* di un investimenti e, qualora fosse positivo, usare il prezzo massimo predetto come *target price* della posizione ed il prezzo minimo predetto come *stop loss*.

2. Contesto e stato dell'arte

In questo capitolo si vuole descrivere il contesto operativo e di ricerca in cui si inserisce questo lavoro di tesi.

Si danno prima alcuni cenni fondamentali sui mercati finanziari e sulle differenti categorie di strategie d'investimento, per poi accennare ad alcuni tra gli studi più significativi in questo ambito di ricerca e allo stato dell'arte.

2.1 Cenni sui mercati borsistici e sulle strategie d'investimento

Un mercato finanziario è un luogo (fisico o non) in cui è possibile effettuare compravendita di strumenti finanziari. Tra i numerosi strumenti finanziari disponibili per l'investimento, i principali sono:

- le azioni;
- le obbligazioni;
- i derivati.

Le **azioni** (o *stock*) sono titoli rappresentativi di quote di partecipazione in una società, e costituiscono il diritto del loro possessore a ricevere una quota dei profitti della società (i cosiddetti dividendi) e il possedimendo di una parte degli asset della stessa. Moltiplicando il numero di azioni, presenti sul mercato, di una società quotata per il valore di ogni azione si ottiene la **capitalizzazione di mercato** della suddetta società, che ne indica il valore sul mercato.

In ciascun paese del mondo è presente almeno una borsa valori, cioè un mercato regolamentato dedicato alla compravendita di strumenti finanziari, tra i quali le azioni delle società quotate in quel paese. Le dieci maggiori borse per capitalizzazione di mercato, relativamente al mese di Marzo 2018, sono quelle elencate nella **Tabella 1**.

Tabella 1
Le dieci maggiori borse valori del mondo, Marzo 2018.

Nome borsa	Paese	Capitalizzazione di mercato (miliardi di dollari)
NYSE	USA	22.755,5
Nasdaq	USA	10.823,3
Japan Exchange Group Inc.	Giappone	6.520,0
Shanghai Stock Exchange	Cina	5.568,9
Hong Kong Exchanges and Clearing	Hong Kong	4.756,2
Euronext	Europa	4.693,7
LSE Group	Regno Unito	4.626,7
Shenzhen Stock Exchange	Cina	3.720,6
BSE India Limited	India	2.404,8
Deutsche Boerse AG	Germania	2.398,3

È possibile valutare nell'insieme l'andamento di uno specifico settore del mercato o delle azioni di società con vari livelli di capitalizzazione (es. *large caps*, *mid-caps*, *small caps*). In questi casi, si definisce un **indice**, che permette di tracciare insieme le azioni appartenenti a diverse società. Alcuni esempi di indici sono:

- Standard & Poor's 500 – contiene le 500 aziende statunitensi (che possono però avere sede legale all'estero) a maggiore capitalizzazione di mercato della borsa NYSE;
- FTSE MIB – contiene le 40 aziende italiane (anche con sede legale all'estero) a maggiore capitalizzazione di mercato della borsa di Milano.

L'andamento di un indice dipende dall'andamento dei titoli azionari delle società che lo compongono.

Nella **Tabella 2** si mostrano i componenti più importanti dell'indice S&P 500.

Tabella 2
Principali società componenti dell'indice S&P 500 e il loro peso relativamente all'indice.

Società	Peso (%)
Apple Inc.	3.988170
Microsoft Corporation	3.571632
Amazon.com Inc.	2.982324
Berkshire Hathaway Inc. Class B	1.820040
Johnson & Johnson	1.657474
JPMorgan Chase & Co.	1.589967
Facebook Inc. Class A	1.486104
Exxon Mobil Corporation	1.455456
Alphabet Inc. Class C	1.395408
Alphabet Inc. Class A	1.368553

Scegliendo di utilizzare le azioni come strumento finanziario per l'investimento, è possibile distinguere due tipi fondamentali di strategie seguite dagli investitori, anche dette *posizioni*:

- le **posizioni lunghe**, in cui si punta su un andamento crescente del valore delle azioni di una società, dunque si cerca di ottenere un profitto da questo incremento di valore acquistandole prima che esso avvenga per poi rivenderle;
- le **posizioni corte**, in cui, viceversa, si punta ad un ribasso del valore delle azioni di una società e si investe in quella direzione, vendendo le azioni prima del ribasso e riacquistandole a prezzo ridotto.

Poichè i mercati presentano per la maggior parte del tempo un andamento al rialzo, le posizioni lunghe sono quelle più comuni. Un andamento al ribasso del mercato si presenta un numero minore di volte, ma è comunque possibile guadagnare molto anche investendo con posizioni corte.

Per valutare una strategia d'investimento, si deve anche tenere conto, necessariamente, dell'orizzonte delle operazioni di compravendita. Le diverse strategie possono riguardare investimenti della durata di pochi minuti ma anche di anni, il che introduce grosse differenze nella pianificazione e nella gestione delle stesse.

Una delle strategie a corto termine più diffuse è quella del **day trading**. Con questo tipo di strategia, un investitore apre e chiude posizioni in un lasso di tempo molto breve,

tipicamente inferiore alla giornata di mercato borsistico. Durante la stessa giornata, l'investitore effettua più volte la compravendita di titoli azionari, e le posizioni non sono mai mantenute oltre la chiusura giornaliera del mercato azionario.
L'uso di tale strategia è motivato dal voler capitalizzare un aumento a corto termine del valore di alcune azioni, ma anche dall'ottenere un ricavo dalla perdita di valore delle stesse, vendendole per poi riacquistarle a prezzo minore. In questo modo non si sfrutta l'andamento generale di base del titolo su cui si investe, ed è dunque possibile ottenere profitti sfruttando movimenti dei prezzi a cortissimo termine.
Gli investitori che operano con questa strategia (*day traders*) affrontano alcuni tra i livelli più alti di rischio dell'intero mercato azionario, in quanto partecipano a condizioni di mercato molto variabili in periodi molto brevi, e capitalizzare su queste variazioni rappresenta la fonte del guadagno di questo tipo di investimenti.

Investendo con questo tipo di strategie, è opportuno definire un valore di **stop loss** per evitare perdite troppo elevate ed incontrollate. Tale soglia indica il limite delle perdite accettabili, e quando il prezzo di un'azione assume un valore che causerebbe una perdita pari a questo valore viene automaticamente emesso un ordine per chiudere la posizione.

Una classe di strategie d'investimento è il **trading quantitativo**[3], che comprende tutte le strategie basate su analisi quantitative. Gli investitori che impiegano l'analisi quantitativa[4] sfruttano, ad esempio, alcuni *ratio* finanziari molto importanti, quali il rapporto prezzo-guadagno (*price-earnings* o *P/E*) o i guadagni per dividendo (*earnings per share* o *EPS*), nel loro processo decisionale legato agli investimenti. Altri strumenti tipici dell'analisi quantitativa sono le *medie mobili* e gli *oscillatori*.
In questo tipo di tecniche di trading, si considera una strategia d'investimento e si crea un modello matematico di essa, il quale viene ottimizzato mediante *backtesting*.
Ottenuto il modello calibrato, si implementa un algoritmo che applica tale modello ai dati storici di mercato, con l'obiettivo di eseguire un investimento con profitto.

Le tecniche di trading quantitativo includono il *trading ad alta frequenza*, il *trading algoritmico* e l'*arbitraggio*, le quali sono tutte tecniche a "fuoco rapido" e, tipicamente, con un orizzonte d'investimento a breve termine.
L'uso di questo tipo di strategie consente di automatizzare l'esplorazione e l'analisi degli strumenti finanziari, oltre a rimuovere completamente la componente sentimentale e la soggettività dalla parte decisionale del trading che spesso possono condurre a perdite.

2.2 Stato dell'arte

L'applicazione di tecniche di *machine learning* sui dati storici finanziari acquisiti dai mercati borsistici rappresenta, attualmente, un ambito di ricerca molto significativo. Una delle assunzioni fondamentali dell'analisi tecnica dei mercati è che i movimenti dei prezzi di mercato non sono casuali, ma seguono alcuni pattern e trend che tendono a ripetersi nel tempo. In questo senso, l'uso delle tecniche di machine learning è volto ad indentificare i più importanti pattern che, ripetendosi nel tempo, possano portare un profitto, distinguendo ed escludendo i pattern casuali. L'obiettivo è quello di proporre nuove strategie d'investimento.

Sotto questo grande cappello, è possibile identificare due principali filoni di ricerca: (i) La pianificazione di strategie d'investimento a lungo termine basate sull'analisi dei dati storici finanziari. (ii) La predizione di trend di mercato a corto termine.

L'approccio qui presentato è assimilabile al trading quantitativo con strategie a corto termine. Al riguardo del trading quantitativo, si vogliono citare alcuni approcci presenti in letteratura.
In [5], i due autori propongono un approccio basato sulle reti neurali (il cui modello è denominato *Quantitative Neural Net*), in cui sfruttano informazioni relative sia all'analisi tecnica che all'analisi fondamentale, in modo da consigliare gli investimenti sulla base di due tipi di decisione: comprare o vendere. Il modello è composto da due parti, basate rispettivamente sui dati propri dell'analisi tecnica e su quelli propri

dell'analisi fondamentale. I dati utilizzati, presi dall'analisi tecnica, sono i dati storici giornalieri delle azioni, mentre quelli propri dell'analisi fondamentale sono, ad esempio, informazioni sui dividendi e sui tassi di sconto. L'obiettivo di questo approccio è eliminare l'incertezza derivante dal comportamento umano.
In [6], gli autori utilizzano le BPNN (*Back Propagation Neural Networks*) e il *Fisher Linear Discriminant* (metodo di riconoscimento dei pattern) per predire l'andamento dei *futures* delle azioni appartenenti all'indice *Shangai and Shenzen 300*. I dati utilizzati sono i dati storici con frequenza di campionamento pari al minuto, e sono state testate diverse strategie di trading, con diversi orizzonti temporali tutti inferiori all'ora.

Per quanto riguarda la predizione dei trend di mercato a corto termine, negli anni sono stati presentati numerosi approcci.
In [7] l'autore utilizza una versione modificata dei classificatori SVM, le *Volume Weighted SVMs*, per classificare i trend delle azioni come crescenti o decrescenti, in base alla percentuale di ritorno giornaliero. In queste SVMs, la funzione di penalizzazione contiene un termine di penalità che dipende dal volume transazionale valutato al momento t per ciascun input x_i . Come dataset, sono stati utilizzati i dati storici giornalieri *OHLC* (*open*, *high*, *low*, *close*), relativi a dieci anni di mercato, di venti azioni prese casualmente, da cui sono stati derivati diversi indicatori propri dell'analisi tecnica. L'autore ha inoltre applicato un meccanismo di selezione degli attributi basato sul *Fisher score*.
In [8] gli autori utilizzano il popolare classificatore *k-NN* per classificare i nuovi trend in base alla loro similarità con i trend già classificati, in quanto è un classificatore tra i più rapidi. Per valutare la similarità dei trend è stata utilizzata la metrica euclidea. I dati storici utilizzati sono quelli relativi ai prezzi di chiusura e ai volumi di compravendita di quindici azioni del *Sao Paulo Stock Exchange*, per un periodo di circa dieci anni. In aggiunta, gli autori hanno utilizzato alcuni indicatori e strumenti (es. stop loss) propri dell'analisi tecnica dei mercati. Questo approccio consente di generare segnali di trading per investimenti sia a lungo che a corto termine.

In [9], gli autori presentano, oltre ad un approccio basato su tecniche di regressione e *windowing* similmente a quanto fatto in questo lavoro di tesi, un approccio basato sul *sequence mining*. Tale approccio permette di scoprire sequenze di elementi ricorrenti da grandi dataset. In questo contesto, le sequenze estratte rappresentano insiemi di azioni che co-occorrono nel periodo temporale considerato, in un numero elevato di finestre temporali. Considerando il supporto delle sequenze estratte (la cui definizione è stata estesa per considerare i profitti giornalieri degli stock in esse contenute), l'idea generale è che se uno stock è tra i primi *k* stock (ordinati secondo il profitto) di un determinato giorno, allora è consigliato investire, per il giorno successivo, sulle azioni che co-occorrono con esso nelle sequenze estratte, considerando le prime *K* sequenze in termini di supporto "esteso".
In [10] e in [11], gli autori cercano di predire i momenti di svolta dei prezzi delle azioni (es. i cosiddetti *bull-flag patterns*) utilizzando due approcci diversi.
Nel primo, gli autori elaborano un modello di riconoscimento dei pattern basato su due tecniche proprie di questo ambito, il "*PIP bull-flag pattern matching*" ed il "*floating-weighted bull-flag template*". I dati storici utilizzati sono quelli *OHLC* e il volume di compravendita delle azioni, da cui sono stati derivati i pattern delle azioni (es. dai prezzi di chiusura) e diversi indicatori tecnici.
Nel secondo, invece, gli autori hanno applicato modelli di reti neurali non lineari e alberi decisionali (costruiti mediante algoritmi genetici o *GA*), utilizzando sia dati storici di mercato (e indicatori tecnici) sia i dati testuali delle notizie recenti. L'uso di notizie del mondo della finanza per guidare gli investimenti è solitamente indicato con il termine *media-aware trading*, e viene spesso accompagnato da una *sentiment analysis* per identificare la positività o la negatività di una notizia testuale. Questa tecnica è giustificata dal fatto che permetterebbe di cogliere maggiori informazioni sull'andamento delle azioni, riuscendo anche a dare, nella maggior parte dei casi, una spiegazione sul cambiamento dei trend delle stesse.

Parlando delle reti neurali, in letteratura sono presenti numerosi utilizzi delle stesse per quanto riguarda la predizione dei mercati azionari, oltre ai già citati [5] e [6].

In [12], gli autori presentano un trading system adattivo, in cui il miglior modello per un determinato indice è ricavato mediante tecniche euristiche. Gli autori ipotizzano, infatti, che non esista un unico modello ottimo valido per tutti gli indici, ma che il modello migliore vada ricavato di volta in volta in base all'indice considerato. Il sistema proposto sfrutta le reti neurali artificiali per la previsione dei trend delle azioni, classificandoli come crescenti o decrescenti, e genera alcuni segnali di trading (es. *buy*, *hold*, *sell*) in modo da dare indicazioni all'investitore. I dati utilizzati sono i prezzi di chiusura e il volume di compravendita delle azioni, a cui si aggiungono diversi indicatori dell'analisi tecnica.
In [13] gli autori utilizzano le reti neurali artificiali e le SVMs per predire la direzione dei prezzi di chiusura giornalieri dell'indice *Instanbul Stock Exchange National 100 Index*, usando come variabili di input i valori di dieci indicatori tecnici.
In [14], l'autore utilizza due modelli di reti neurali, *multi-layer perceptron* (MLP) e *generalized regression neural network* (*GRNN*), per ottenere lo stesso obiettivo rispettivamente ai dati del *Kuwait Stock Exchange*.

Un'altro approccio per ottenere una predizione accurata dei trend delle azioni, generalmente molto presente in letteratura, è quello di combinare diversi modelli predittivi.
In [15], gli autori impiegano un *ensemble* di reti neurali artificiali, ottimizzate mediate la *Particle Swarm Optimization*, per predire il trend del *Mexican Stock Exchange*.
In [16], gli autori sfruttano una fusione a due stadi di alcuni modelli di machine learning. Il primo stadio è costituito dalla SVR (*Support Vector Regression*) ed ha il compito di preparare gli input per il secondo stadio, per il quale sono stati valutati le reti neurali artificiali, il *Random Forest* e, nuovamente, la SVR. I dati di input dei modelli sono dieci indicatori tecnici, ed il sistema è stato testato sugli indici *CNX Nifty* e *S&P Bombay Stock Exchange Sensex*.
In [17], gli autori presentano un nuovo approccio al *rule-based evidential reasoning* (*RBER*) per generare segnali di trading che guidino gli investimenti nel mercato *Foreign Exchange* (*FOREX*).

A differenza degli approcci sopra presentati per guidare gli investimenti a corto termine, il sistema descritto in questo lavoro di tesi può generare molteplici raccomandazioni di azioni nei distinti momenti di una giornata di mercato borsistico. L'uso di dati con frequenza di campionamento minore di quella giornaliera e la costruzione di diverse aggregazioni di essi permettono l'intervento sul mercato nei diversi slot temporali della singola giornata di borsa, potendo valutare con quale frequenza farlo, in modo inversamente proporzionale al livello di aggregazione scelto dei dati storici.

Il sistema proposto, inoltre, fa uso di tecniche di regressione. L'uso di tali tecniche è poco comune in letteratura, in quanto nella maggior parte degli approcci elencati si cerca di classificare i trend piuttosto che prevedere il valore dei prezzi delle azioni nello specifico. Adoperando le tecniche di regressione, è possibile ordinare le azioni in esame secondo il valore percentuale del guadagno predetto, il che comporta una maggiore interpretabilità dei segnali di trading rispetto a quelli basati su classificatori.

La strategia d'investimento adoperata si discosta radicalmente da quelle presenti in letteratura. La predizione dei trend delle azioni per investimenti a corto termine riguarda, solitamente, la predizione del prezzo di chiusura delle stesse. Nel sistema proposto, differentemente, si cerca di predire il valore del prezzo massimo delle azione negli slot temporali successivi.
Il sistema proposto è in grado, dunque, di predire il *target price* di un eventuale segnale di trading intraday generato su una specifica azione. Questa caratteristica lo rende utilizzabile a complemento di strategie di generazione (semi-)automatica di segnali intraday, sia per le posizioni lunghe che per quelle corte.

3. Raccolta dei dati e preparazione

In questa sezione, verranno discusse le fasi di raccolta e preparazione dei dati azionari. La raccolta avviene tramite un crawler specifico per la IEX API[18], appositamente progettato e scritto in linguaggio Java.
Viene inizialmente presentata l'API scelta, indicandone le principali caratteristiche tecniche e limiti connessi. Successivamente, si descriveranno le scelte progettuali relative al crawler, tra le quali i framework del linguaggio utilizzati e l'architettura software scelta, oltre ad una valutazione sull'efficienza del processo di crawling.

3.1 Investitors Exchange API

La IEX API è un insieme gratuito di servizi pensati per fornire supporto alla progettazione di servizi e applicazioni a valore aggiunto. Essa è un'API di tipo REST e fornisce numerosi endpoint, tutti prefissi dalla URL *https://api.iextrading.com/1.0*, ai quali si possono effettuare richieste di tipo GET. Tutti gli endpoint supportano esclusivamente JSONP come rappresentazione dei dati restituiti.

Gli endpoint utilizzati dal crawler, relativamente a questo lavoro di tesi, sono sostanzialmente due.

Il primo è l'endpoint */ref-data/symbols*, che risponde con la lista dei simboli delle compagnie supportate da IEX. I simboli sono rappresentati nella notazione **INET** (Nasdaq Integrated simbology). Questa lista è aggiornata periodicamente, con cadenza giornaliera.

Il secondo è l'endpoint */stock/{SYMBOL}/chart/*, utilizzato nello specifico con le due versioni */stock/{SYMBOL}/chart/{RANGE}* e */stock/{SYMBOL}/chart/date/{yyyyMMdd}*. La prima versione risponde con i dati storici, con frequenza di campionamento giornaliera, dei grafici OHLC (più altre informazioni, quali il volume di compravendita

di un simbolo e il numero di transazioni) per lo stock *SYMBOL*e per il periodo *RANGE*, il quale può assumere diversi valori predefiniti di intervalli di tempo (es. 5y, 2y, 1y, 6m, ecc.).
La seconda versione, differentemente, permette di ottere gli stessi dati storici indicati precedentemente con frequenza di campionamento pari al minuto, per lo stock *SYMBOL* e il giorno *yyyyMMdd* (indicato con il formato data di Java). Al momento delle prime raccolte dati (03/2018), l'API non supportava date antecedenti a 30 giorni rispetto a quella in cui veniva effettuato il crawling; al momento della scrittura del presente elaborato (08/2018), risultano invece essere supportate.

Dalla struttura del secondo endpoint, risultano evidenti i seguenti limiti dell'API, in termini di efficienza del crawling:

- è necessario effettuare una richiesta per ciascuno stock d'interesse; diventa problematico nel caso di indici azionari molto estesi, quali lo S&P500 considerato in questo lavoro di tesi;
- le frequenze di campionamento disponibili dei dati azionari sono quella giornaliera e quella per minuto. Nel caso di studio di questo lavoro di tesi, risulterebbe molto utile disporre di altre frequenze di campionamento, quali quella oraria o per due/tre ore; ciò impone l'utilizzo della frequenza per minuto, con conseguente raccolta di un'eccessiva mole di dati e aggregazioni da effettuare successivamente al crawling.

Date le considerazioni dei punti precedenti, è necessario effettuare una richiesta per ciascuna data di interesse, per ciascuna azione considerata. Da quanto detto al primo punto, il numero di richieste all'API per un crawling completo, dato un periodo T di N giorni di interesse e un insieme S di M azioni, risulta essere pari a $N \times M$; si noti che, per la struttura dell'API stessa, non è possibile ridurre tale numero.

3.2 Data crawler

Il crawler progettato è stato implementato con il linguaggio Java ed è specifico per la IEX API. Permette la raccolta dei dati storici degli stock (compatibilmente con le caratteristiche dell'API) con frequenza di campionamento sia giornaliera che per minuto, la seconda delle quali scelta per collezionare i dati per il presente lavoro di tesi.

Il crawler richiede che sia presente nel classpath un file di configurazione *.properties*, il quale deve contenere, oltre ad alcuni campi che verranno descritti successivamente, il percorso di un file CSV con la lista delle azioni delle quali raccogliere i dati.

Il crawler produce un **dataset strutturato**, in cui ogni record è un insieme di **item** rappresentati come coppie attributo-valore.

Definizione 1. Dataset strutturato. Sia T un periodo temporale e TS un sottoinsieme di periodi temporali $t_k \in T$ di dimensione fissa e definita, ciascuno dei quali indicato con il timestamp relativo alla fine del periodo. Sia $m \in \{o, h, l, c, v, n\}$ una grandezza misurabile delle azioni e si indichi con $\overline{m}$ il valore della misurazione della stessa. Sia inoltre I un insieme di item rappresentati come coppie del tipo $\langle m_{s_j}, \overline{m_{s_j}}^{k} \rangle$, in cui il primo elemento della coppia è una delle grandezze misurabili relativa allo stock $s_j \in S$ (con S l'insieme degli stock) ed è indicato come attributo del dataset, mentre il secondo elemento rappresenta il valore della stessa, osservata per lo slot temporale t_k. Un record $R(t_k)$ è l'insieme di tutti gli item $\langle m_{s_j}, \overline{m_{s_j}}^{k} \rangle \in I$ associati al periodo t_k. Un dataset strutturato è un insieme di record $R(t_k)$, uno per ciascuno slot temporale $t_k \in T$.

Nel presente lavoro di tesi, le grandezze misurabili relative alle azioni considerate (primo elemento della coppia in un item) sono grandezze misurabili semplici, nello specifico il prezzo di apertura, il prezzo massimo, quello minimo, il prezzo di chiusura, il volume di compravendita e il numero di operazioni che verranno indicati come *o, h, l, c, v, n* rispettivamente. I dataset prodotti conterranno tali grandezze o un sottoinsieme

di esse. È comunque possibile estendere l'insieme delle grandezze misurabili, in modo da arricchire i dataset prodotti con nuovi attributi rappresentativi di grandezze composte, ad esempio gli indicatori e gli oscillatori comunemente usati nell'analisi tecnica (es. medie mobili, RSI).

Gli slot temporali, ai quali le grandezze misurate si riferiscono, si indicheranno da qui in avanti indifferentemente come *timeslot* o *timestamp*, posto che il timestamp indicato si riferisce al periodo temporale precedente di dimensione fissa, definita e pari ad una delle granularità trattate.

La versione attuale del crawler è disponibile liberamente per scopi di ricerca, previa richiesta.

3.2.1 Scelte implementative e framework Java utilizzati

Il crawler sfrutta esclusivamente due librerie preesistenti, il client del framework Jersey e Jackson (nella versione ad-hoc per Jersey).

Il framework open-source *Jersey RESTful Web Services*[19] è sostanzialmente un'implementazione estesa del framework JAX-RS che permette di realizzare servizi RESTful in Java. Fornisce una vera e propria API che estende il toolkit di JAX-RS. Nel dettaglio, la parte client è una *fluent* API che permette la comunicazione con servizi di tipo REST, e dunque di effettuare/gestire richieste HTTP/HTTPs.

Jackson è un interprete JSON ad alte prestazioni, realizzato nella forma di una libreria Java per il mapping tra oggetti JSON e POJOs (Plain Old Java Object). Nel crawler viene utilizzata tale libreria per la conversione del corpo delle risposte HTTP (come descritto precedentemente, IEX API supporta esclusivamente JSONP) in oggetti JAVA, manipolabili in modo semplice e dinamico.

3.2.2 Architettura software

Il crawler è stato implementato sotto forma di libreria seguendo l'*Abstract Factory Pattern*, un pattern di programmazione a oggetti in cui si utilizza un'implementazione concreta di un'interfaccia *Factory* per generare gli oggetti concreti con cui il client interagisce. Tale pattern di programmazione é stato scelto in quanto permette di nascondere al client i dettagli implementativi degli oggetti trattati, mostrandone soltanto il loro uso generale tramite le interfacce. Inoltre, isola la creazione degli oggetti dal loro uso, rendendo molto semplice l'estensione del software mediante l'aggiunta di nuovi tipi derivati che differiscono nel funzionamento da quelli già presenti.

Nel suddetto crawler, le classi *CrawlerFactory* e *CrawlerFactoryImpl* costituiscono l'interfaccia e l'implementazione della factory, rispettivamente. Come mostrato in figura 3.1, gli oggetti concreti sono le classi *CrawlerImpl*, *ConcurrentCrawlerImpl* e *IntradayCrawlerImpl*, le quali implementano tutte l'interfaccia *Crawler* differenziandosi per logica di funzionamento e presenza o assenza di concorrenza.

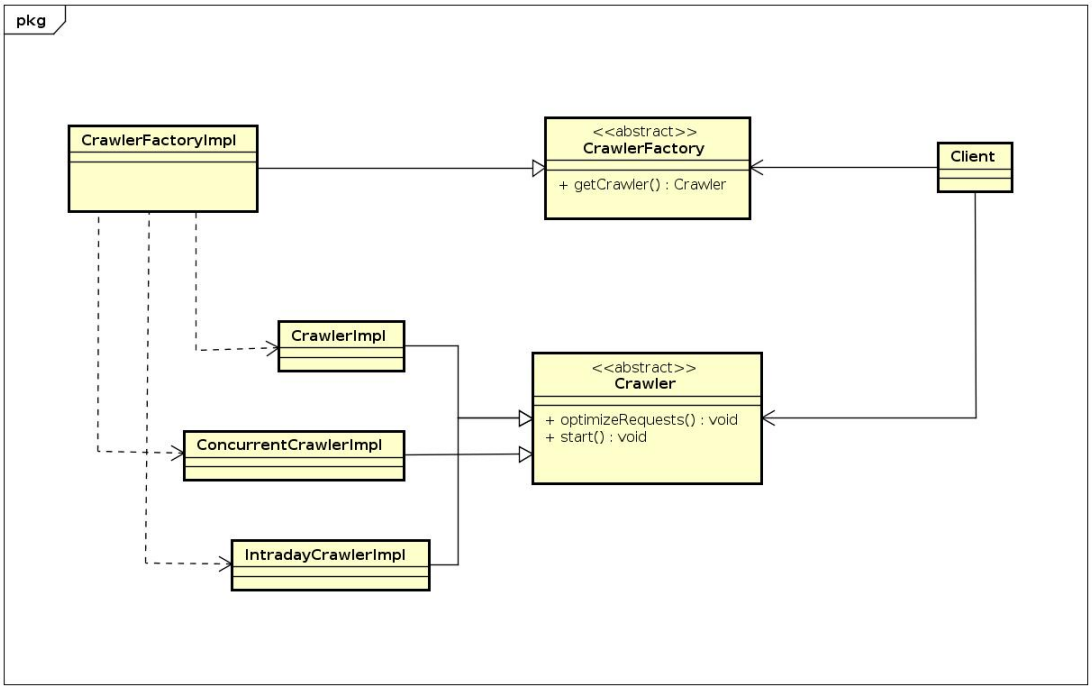

Fig. 3.1 Diagramma delle classi del crawler, in cui sono mostrate anche le principali relazioni.

3.2.3 La classe factory *CrawlerFactoryImpl*

La versione di factory proposta, in carica di istanziare la versione più appropriata del Crawler, è configurabile liberamente tramite un file contenente i parametri di configurazione, il file *config.properties*. I file *.properties* sono uno strumento molto usato, principalmente nelle tecnologie Java, per memorizzare tali parametri, e hanno il seguente formato: ciascuna linea conserva una proprietà ed il relativo valore (*nome_proprietà=valore*), il file contiene tante linee quante sono le proprietà da configurare.

Nel dettaglio, è possibile configurare i seguenti parametri: il periodo dei dati storici di interesse (secondo i valori previsti dall'IEX API), la loro granularità (*d* giornaliera e *m* per minuto), il percorso del file relativo alla lista degli stock di cui richiedere i dati (in formato CSV, uno stock per riga, con il simbolo in notazione INET come primo campo),

il nome del dataset strutturato di output (anch'esso in formato CSV), il separatore da utilizzare tra i vari campi di un record, e un flag che indica se si vuole utilizzare l'implementazione multi-thread del Crawler. Alcune proprietà, quali la lista degli stock, sono richieste obbligatoriamente, pena il sollevamento dell'eccezione *CrawlerFactoryException*.

Il file di configurazione deve chiamarsi *config.properties* e viene caricato dalla factory ricercandolo nel suo classpath, dunque è consigliato inserirlo nella cartella *resources* della libreria. Se non è presente, viene sollevata nuovamente l'eccezione *CrawlerFactoryException* con un messaggio apposito.

La factory carica le proprietà di configurazione, e a seconda di quali sono indicate e del loro valore istanzia un'implementazione del crawler, tra le tre disponibili (i dettagli della logica di scelta sono indicati nel diagramma di flusso in figura 3.2).
In particolare, l'utente può scegliere se utilizzare o no la versione concorrente del crawler in caso di dati a granularità giornaliera, mentre la classe *IntradayCrawlerImpl* è sempre istanziata se il periodo di campionamento è pari ad un mese e la frequenza scelta è quella per minuto.

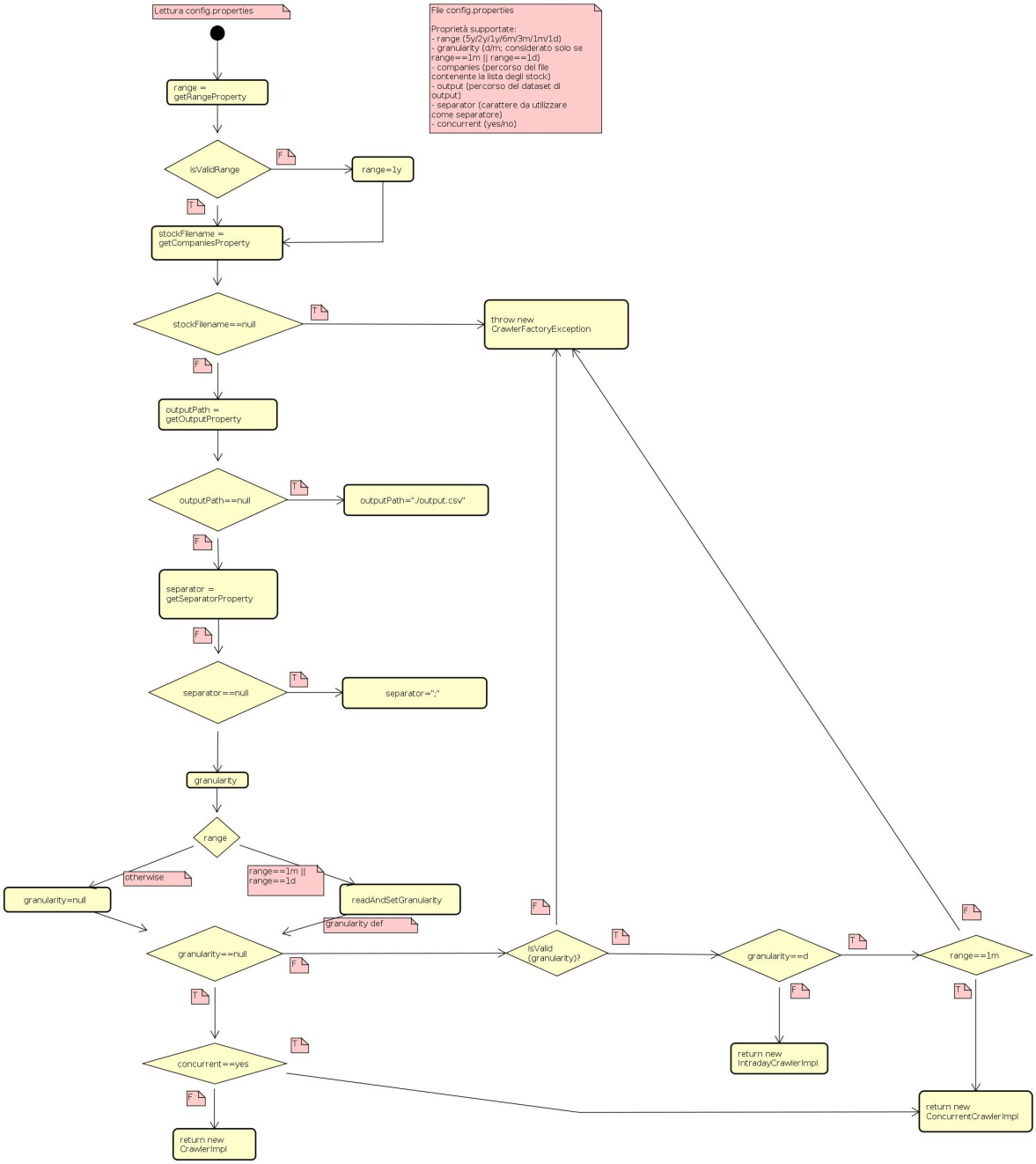

Fig. 3.2 Rappresentazione in flow-chart della logica con cui la classe CrawlerFactoryImpl istanzia una delle implementazioni del crawler.

3.2.4 Le classi *CrawlerImpl* e *ConcurrentCrawlerImpl*

Queste due classi implementano l'interfaccia *Crawler* e collezionano i dati storici degli stock con frequenza di campionamento giornaliera.

L'interfaccia richiede l'implementazione del metodo *optimizeRequests()*, che serve a confrontare le azioni della lista in input con quelle supportate dall'IEX API (ottenibili con l'endpoint */ref-data/symbols*) con lo scopo di ridurre, se possibile, il numero di richieste all'API non effettuando quelle per gli stock i quali dati storici non sono disponibili. Questo tentativo di riduzione comporta una richiesta aggiuntiva, ma è sempre opportuno effettuarlo, in quanto il supporto dell'IEX API relativo ad alcuni stock non è costante: può accadere che un particolare simbolo non sia più supportato al momento del crawling ma che il suo endpoint sia ancora attivo, dunque il periodo di campionamento richiesto (es. "5y") varia tra i diversi simboli di interesse, generando dataset molto sparsi.
Si segnala che il metodo *optimizeRequests()* è invocato automaticamente in fase di costruzione nell'implementazione *ConcurrentCrawlerImpl*, mentre non lo è nella classe *CrawlerImpl*.

L'implementazione del metodo *start()*, invece, differisce tra le due versioni considerate.

Nel crawler classico, *CrawlerImpl*, un unico thread effettua una richiesta per volta all'endpoint */stock/{SYMBOL}/chart/{RANGE}*, aspetta i dati, li processa e li inserisce in una mappa di mappe con il timestamp come chiave esterna ed il simbolo dello stock come chiave interna. Tale mappa verrà utilizzata per scrivere il dataset di output.

Nel crawler concorrente, *ConcurrentCrawlerImpl*, viene istanziato un pool di *Java Executors*, i quali si occupano di creare i thread necessari, capaci di eseguire task asincroni in modo molto efficiente in quanto ogni thread viene riutilizzato.

Il pool contiene un numero di esecutori pari a $\frac{N_{SYMBOLS}}{N_{REQUESTS}}+1$, con $N_{SYMBOLS}$ pari al

numero di azioni richieste e $N_{REQUESTS}$ che indica il numero di stock gestiti da ciascun esecutore (fissato a 15 in seguito a considerazioni euristiche sull'efficienza).
Vengono poi generati tanti *Task*, uno per ciascuno stock presente nella lista di input (eventualmente filtrata dal metodo *optimizeRequests()*), ciascuno incaricato di collezionare i dati storici per il periodo richiesto, con frequenza di campionamento giornaliera, per un singolo stock.
I task sono eseguiti in modo asincrono, e restituiscono il risultato al thread principale, quando pronto, sotto forma di mappa.
Le mappe ottenute come risultato dei task vengono combinate in una mappa di mappe (avente lo stesso formato di quella relativa a *CrawlerImpl*, indicato precedentemente) che rappresenta il dataset di output.

Ambedue le classi implementano il metodo *writeCsv(Map csv)* nello stesso modo. Tale metodo privato è chiamato alla fine del processo di crawling, e riceve la mappa risultante dalla combinazione delle mappe restituite dai task. Il metodo scrive la mappa sul file di output (indicato nella configurazione) utilizzando il separatore scelto. Tale file di output è un dataset strutturato (definito in 3.2), contenente gli item $\langle o_{s_i}, \overline{o_{s_i}}^k \rangle$, $\langle h_{s_i}, \overline{h_{s_i}}^k \rangle$, $\langle l_{s_j}, \overline{l_{s_j}}^k \rangle$, $\langle c_{s_j}, \overline{c_{s_j}}^k \rangle$ e $\langle v_{s_j}, \overline{v_{s_j}}^k \rangle$ per ciascuno stock $s_j \in S$ e per ciascun timestamp $t_k \in T$.

Date	A-Open	A-High	A-Low	A-Close	A-Volume	AAP-Open	AAP-High	AAP-Low	AAP-Close	AAP-Volume
2013-03-11	42,99	43,01	41,73	42,81	5472837	76,88	78,06	76,84	77,53	689362
2013-03-12	42,8	43	42,59	42,62	3767613	77,26	77,66	76,89	77,36	660535
2013-03-13	42,56	43,05	42,4	43	2241086	77,54	78,85	77,32	78,77	644976
2013-03-14	43,03	43,38	43	43,32	2840603	78,9	79,12	77,36	77,67	763234
2013-03-15	43,1	43,22	42,88	43,01	2579690	77,9	79,59	77,41	79,57	1237958
2013-03-18	42,46	42,7	42,18	42,55	1847766	79,09	80,76	79,0101	80,7	1219201
2013-03-19	42,68	42,82	41,59	42	6246443	80,64	80,9399	79,77	80,07	1370222
2013-03-20	42,25	42,785	42,22	42,7	3240314	80,07	80,704	79,8005	80,22	728484
2013-03-21	42,45	42,76	41,63	41,65	4409084	79,91	80,72	79,85	80,03	616238
2013-03-22	41,79	41,79	40,91	41,25	5425027	80,12	80,44	78,972	79,81	586660
2013-03-25	41,49	41,654	40,79	41,13	2797510	79,83	80,94	79,33	80,53	1150900
2013-03-26	41,4	41,82	41,32	41,82	2513554	80,86	81,61	80,56	81,6	969581
2013-03-27	41,4	42,19	41,21	42,09	3369968	81,02	81,7	80,9601	81,64	551193
2013-03-28	42,08	42,08	41,4995	41,97	2147900	81,34	83,07	81,29	82,65	1287266

Fig. 3.3 Esempio di dataset generato con granularità giornaliera. Sono mostrati i dati per 14 giorni e per le azioni A e AAP.

3.2.5 La classe *IntradayCrawlerImpl*

La raccolta di dati storici con frequenza di campionamento maggiore della giornata di borsa è effettuata da questa implementazione del crawler.
In particolare, essa è progettata per richiedere i dati storici all'IEX API con granularità pari al minuto, che è l'unica messa a disposizione dall'API che sia minore di quella giornaliera.

Da quanto detto in 3.1, è possibile effettuare il crawling di dati storici con tale granularità per un periodo di 30 giorni per volta, con una richiesta per ciascuna data e per ciascuno stock d'interesse. Dunque, l'implementazione adoperata sfrutta la concorrenza nelle stesse modalità già descritte in 3.2.4 per la classe *ConcurrentCrawlerImpl*.

Il dataset di output è anch'esso un dataset strutturato (definito in 3.2), con in aggiunta gli item $\langle n_{s_j}, \overline{n_{s_j}}^{k} \rangle$, che rappresentano il numero di transazioni per lo stock $s_j \in S$ osservato al timestamp $t_k \in T$, per ciascuno stock e ciascun timestamp.

Data la granularità dei dati raccolti, il dataset prodotto non possiede i dati OHLC per tutti i timestamp considerati di ciascuno stock. È molto probabile infatti che in un dato minuto e per una data azione non sia avvenuta compravendita della stessa; in questo caso, i valori di apertura, chiusura, massimo e minimo non possono essere definiti per ovvie ragioni. L'IEX API ritorna tali valori come nulli, mentre il crawler li inserisce nel dataset di output con il valore negativo -1. È dunque necessaria una pulizia successiva del dataset prodotto, in quanto tale valore introdurrebbe del rumore nei modelli generati per l'analisi.

Timestamp	A-Open	A-High	A-Low	A-Close	A-Volume	A-NumOfTrades	AAL-Open	AAL-High	AAL-Low	AAL-Close	AAL-Volume	AAL-NumOfTrades
20180718 09:30	63	63	63	63	100	1	38,06	38,24	38,06	38,2	1400	14
20180718 09:31	-1	-1	-1	-1	0	0	38,3	38,3	38,29	38,3	300	3
20180718 09:32	-1	-1	-1	-1	0	0	38,3	38,37	38,3	38,37	500	5
20180718 09:33	-1	-1	-1	-1	0	0	38,41	38,44	38,335	38,335	600	5
20180718 09:34	-1	-1	-1	-1	0	0	38,37	38,42	38,37	38,42	400	4
20180718 09:35	-1	-1	-1	-1	0	0	38,41	38,54	38,41	38,54	731	6
20180718 09:36	62,78	62,78	62,78	62,78	100	1	38,51	38,53	38,5	38,5	714	8
20180718 09:37	62,8	62,8	62,8	62,8	29	1	38,5	38,51	38,5	38,51	613	7
20180718 09:38	-1	-1	-1	-1	0	0	38,5	38,5	38,49	38,5	187	5
20180718 09:39	-1	-1	-1	-1	0	0	38,485	38,52	38,485	38,52	1078	8
20180718 09:40	62,95	62,95	62,95	62,95	100	1	38,5	38,52	38,5	38,52	1300	8
20180718 09:41	-1	-1	-1	-1	0	0	38,46	38,46	38,46	38,46	100	1
20180718 09:42	62,95	62,95	62,95	62,95	100	1	38,45	38,455	38,45	38,455	200	2
20180718 09:43	63,09	63,09	63,09	63,09	92	1	38,495	38,62	38,49	38,595	500	6
20180718 09:44	-1	-1	-1	-1	0	0	-1	-1	-1	-1	0	0
20180718 09:45	63,18	63,18	63,16	63,16	300	3	38,69	38,71	38,69	38,71	710	5

Fig. 3.4 Esempio di dataset generato con granularità dei dati pari al minuto. Sono mostrati i dati per 15' e per i simboli A e AAL. Si nota la presenza dei valori nulli, codificati come -1, in corrispondenza dei valori di volume e del numero di scambi entrambi pari a 0. Si rendono necessarie operazioni di pulizia.

3.2.6 Considerazioni sull'efficienza del crawling

Da quanto detto precedentemente, è evidente che l'efficienza della raccolta dei dati storici è fortemente limitata dalla struttura degli endpoint dell'API REST scelta come fonte. Ciascuno stock necessita di una richiesta nel caso di dati giornalieri, di una richiesta per ciascun giorno del periodo di interesse nel caso di dati per minuto.

Una risposta dell'API contiene un oggetto JSON per ciascuna unità di tempo (minuto o giorno) del periodo storico richiesto. Dunque contiene $D(T)$ oggetti, con T pari al periodo richiesto, nel caso della granularità giornaliera, mentre restituirà sempre $N_D = 390$ oggetti nel caso di dati per minuto (in questo caso infatti, una risposta contiene i dati del giorno di borsa specifico richiesto, che inizia alle ore 9:30 e si conclude alle 16:00). Tali oggetti JSON differiscono a seconda della granularità del dato trasportato, contenendo 12 e 20 campi per dati giornalieri e per minuto, rispettivamente.

Premesso ciò, è banale effettuare una comparazione tra le diverse dimensioni possibili delle risposte, senza pretesa di totale accuratezza.

Nel caso di dati storici giornalieri, il massimo periodo richiedibile all'API è pari a 5 anni; considerando un numero di settimane annuali pari a 52 e che i giorni di mercato azionario sono 5 a settimana, si ha che la dimensione massima di una risposta è pari a 1310 oggetti di 12 campi ciascuno, per un totale di 15720 campi totali.
Nel caso dei dati storici per minuto, una risposta contiene sempre i dati di una giornata di mercato, dunque 390 oggetti di 20 campi ciascuno, per un totale di 7800 campi totali.
È evidente la differenza, in ordine di grandezza, tra un crawling di dati storici giornalieri e uno di dati storici per minuto, in quanto i dati di un solo giorno con dati per minuto sono pari a circa il 50% dei dati storici giornalieri di 5 anni.

Si segnala che l'IEX API mette a disposizione un parametro di richiesta su tutti gli endpoint, per filtrare i risultati restituendo solo i campi di interesse per ciascun oggetto JSON. Il filtro è indicato dal parametro *?filter=[Comma_separated_list_of_wanted_fields]* e permette di ridurre in modo sensibile il peso di ciascuna risposta nel caso si vogliano i dati relativi ad un sottoinsieme dei campi disponibili, al prezzo di un'implementazione aggiuntiva specifica di una classe per il mapping da corpo della risposta a oggetto Java.

Il numero delle richieste, come detto in precedenza, non è in alcun modo riducibile e costituisce il principale collo di bottiglia del crawler, soprattutto nel caso di dati storici per minuto.

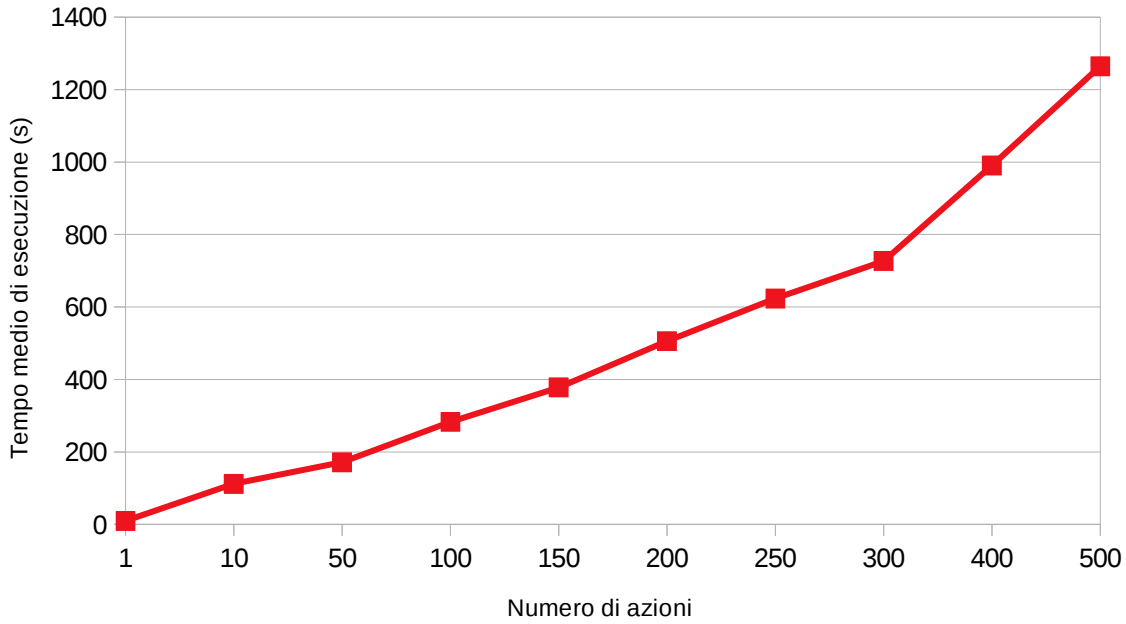

Fig. 3.5 Scalabilità del crawler al variare del numero degli stock. Per ogni azione vengono raccolti i 30 giorni di dati storici, con granularità pari al minuto, precedenti al giorno in cui si effettua il crawling. I tempi indicati si riferiscono al crawling con una connessione a 12 Mbps in downstream e ad una CPU con 4 core.

Si vuole, infine, fornire un'indicazione generale sulla scelta dell'implementazione più adatta nei vari casi.

Nel caso del crawling di dati storici giornalieri, la scelta dipende dal periodo di interesse e dal numero di azioni richieste.
Se il periodo di campionamento richiesto è pari ad un anno o inferiore e il numero di stock d'interesse è nell'ordine di 10^2 o inferiore, è consigliata l'implementazione a singolo thread, *CrawlerImpl*: tale implementazione effettuerà, sequenzialmente, $N_{SYMBOLS}$ richieste e processerà le risposte, il cui peso non sarà molto significativo (ogni risposta HTTP conterrà circa 262 oggetti JSON, con 12 campi). L'uso di questa implementazione, per un numero ridotto di simboli, risulta vantaggiosa in quanto annulla l'overhead di creazione e gestione degli esecutori, che diventa un'operazione esosa in termini di risorse e non giustificata in questo caso.

In tutti gli altri casi, si consiglia l'uso delle implementazioni concorrenti del crawler. Per quanto riguarda i dati storici giornalieri, l'uso del pool di esecutori migliora considerevolmente i tempi di esecuzione della raccolta dei dati quando il numero di stock richiesti è elevato e la risposta HTTP è pesante (periodo d'interesse maggiore di un anno e numero standard di campi degli oggetti JSON), mentre, considerando i dati storici per minuto, la pesantezza della risposta HTTP descritta precedentemente (per un singolo giorno) e il numero elevato di richieste necessarie per l'intera raccolta dei dati giustificano ampiamente l'utilizzo del pool di esecutori, in quanto l'overhead aggiunto dalla gestione dei thread è ampiamente trascurabile rispetto alla durata dell'intera operazione e ne riduce sensibilmente il tempo di esecuzione.

3.3 Qualità dei dati e operazioni di pulizia

In questa sezione si discuteranno la qualità dei dataset generati dal crawler (figure 3.3 e 3.4) e le eventuali operazioni di pulizia necessarie all'ottenimento di un dataset utile per le successive analisi. In particolare, si terrà conto della presenza (possibile) di *rumore* e della possibilità che ci siano dati mancanti, dovuti ad un disservizio dell'IEX API o di una delle sue fonti di terze parti.
Tale analisi è descritta separatamente per i due tipi di dataset generati dalla fase di raccolta.

3.3.1 Dataset con dati storici giornalieri

Per tali dataset (ad esempio quello in figura 3.3), l'API scelta fornisce dati coerenti e, generalmente, privi di rumore.

Il rumore è però generato dal crawler in fase di scrittura del dataset quando, per un dato simbolo e un dato giorno di borsa, il volume di compravendita è pari a zero; l'API, infatti, ritorna valori nulli per i valori OHLC che vengono codificati dal crawler come -1. Tale condizione si verifica per i simboli caratterizzati generalmente da un basso volume

di compravendita, che risultano però essere in numero poco significativo, specie se si considerano mercati vasti o indici quali lo *S&P500*.

Per risolvere questo problema di qualità dei dati, si è scelto di applicare il metodo *Last Observation Carried Forward* (LOCF), in cui vengono utilizzati, come elementi del grafico OHLC per un timestamp con volume pari a zero, i valori OHLC relativi all'ultimo timestamp in cui si è verificata compravendita azionaria del simbolo considerato. I dati sul volume scambiato vengono, ovviamente, mantenuti pari a zero, in modo da rendere distinguibili tali timestamp da quelli in cui sono avvenuti scambi azionari e per cui ha senso che siano definiti i valori OHLC.

Per quanto riguarda i dati mancanti relativamente ad uno stock e ad una (o più) giornata di borsa, si è deciso di non effettuare alcuna interpolazione e, semplicemente, di non considerare lo stock per le analisi successive, scartandone i dati. Cercare di interpolare i dati mancanti, infatti, potrebbe non essere appropriato e potrebbe introdurre informazioni fuorvianti.

3.3.2 Dataset con dati storici per minuto

Per i dataset contenenti dati con frequenza di campionamento pari al minuto (un esempio in figura 3.4) è molto comune che, per un determinato simbolo, siano presenti molti timestamp in cui non si sono verificati scambi azionari. Come già detto, questo si traduce con la presenza di dati corretti per quanto riguarda l'informazione sul volume di scambio e sul numero di scambi, ma con valori pari a -1 per quanto riguarda i dati OHLC, per quello stock e per quel particolare timestamp.

Come nel caso dei dataset con dati storici giornalieri, si è applicato il metodo LOCF, con le stesse modalità già descritte nel paragrafo precedente.

Per quanto riguarda il problema dei dati mancanti per un certo stock e per un determinato timestamp, invece, si è scelto di non scartare lo stock e mantenere il dato come non disponibile. La computazione di alcune statistiche sui dataset generati ha permesso di verificare che le informazioni non disponibili sono molto rare (12 timestamp mancanti su circa 6 mesi di dati storici, considerando lo stock con il numero maggiore di dati mancanti), e il problema non ha impatto sulle analisi effettuate nel presente elaborato, in quanto è d'interesse l'utilizzo di versioni aggregate del dataset, aventi granularità maggiore di quella per minuto.

3.4 Aggregazione dei dati con frequenza per minuto

I dataset contenenti i dati storici degli stock con frequenza di campionamento pari al minuto (descritti nel paragrafo 3.2.5) vengono trasformati in modo da ottenere frequenze di campionamento minori. Nel presente lavoro sono state considerate granularità pari a trenta minuti, un'ora e due ore, generando tre versioni corrispondenti del dataset.

Dato un periodo T contenente un numero di timeslot di dimensione pari a una delle granularità elencate, i dati vengono aggregati come segue: siano $R(t_k), \forall t_k \in T, k \in \{1,..,N\}$ i record del dataset relativi al periodo T, il risultato dell'operatore di aggregazione è un nuovo record $R(T)$ tale che

$$\langle o_{s_j}, \overline{o_{s_j}}^T \rangle = \langle o_{s_j}, \overline{o_{s_j}}^1 \rangle , \langle c_{s_j}, \overline{c_{s_j}}^T \rangle = \langle c_{s_j}, \overline{c_{s_j}}^N \rangle , \langle h_{s_j}, \overline{h_{s_j}}^T \rangle = \max_{\overline{h_{s_j}}^k} \{ \langle h_{s_j}, \overline{h_{s_j}}^k \rangle \} ,$$

$$\langle l_{s_j}, \overline{l_{s_j}}^T \rangle = \min_{\overline{l_{s_j}}^k} \{ \langle l_{s_j}, \overline{l_{s_j}}^k \rangle \} , \langle v_{s_j}, \overline{v_{s_j}}^T \rangle = \sum_{\overline{v_{s_j}}^k} \langle v_{s_j}, \overline{v_{s_j}}^k \rangle , \langle n_{s_j}, \overline{n_{s_j}}^T \rangle = \sum_{\overline{n_{s_j}}^k} \langle n_{s_j}, \overline{n_{s_j}}^k \rangle$$

con $s_j \in S$ il j-esimo stock dell'insieme degli stock S presenti nel dataset. La stessa operazione è effettuata per tutti i periodi della stessa dimensione di T presenti nel dataset.

Quanto detto è applicato con la limitazione di non includere in ciascun record aggregato dati relativi a due giorni di mercato diversi. Dunque, vista la durata della giornata di borsa pari a 6 ore e 30 minuti, nei dataset con granularità pari a un'ora e a due ore gli ultimi timestamp di ciascun giorno di mercato rappresentano periodi di tempo più grandi delle rispettive granularità, aggregando anche i dati relativi all'ultima mezz'ora della giornata.

4. Generazione dei modelli

In questo capitolo verrà discussa la fase di generazione dei modelli di predizione, ottenuti applicando tecniche di regressione sui dataset strutturati contenenti i dati storici delle azioni, e la loro validazione in termini di errore di predizione.

Nel primo paragrafo vengono dati dei cenni sugli algoritmi di machine learning utilizzati, mentre negli altri viene descritta, nel dettaglio, la metodologia utilizzata per il tuning dei loro parametri, per la creazione dei modelli e per la valutazione delle performance.

Le operazioni di calibrazione dei parametri degli algoritimi, learning dei modelli e valutazione delle loro performance sono state realizzate sfruttando il tool di visual workflow design *RapidMiner*.

4.1 Cenni sui modelli predittivi di regressione[20]

Un problema di regressione consiste nell'approssimare una funzione *f* a partire da variabili di input *X* (detti *predittori*) ad un valore *y* continuo ed a valori reali (*predizione*). Gli output dei problemi di regressione sono dunque valori reali e continui, differentemente dai problemi di classificazione i cui output sono invece valori discreti.

Nel seguito, vengono illustrati alcuni tra i modelli predittivi più noti in letteratura applicabili sui problemi di regressione, i quali sono stati utilizzati e valutati per il problema in oggetto.

4.1.1 Regressione lineare

Un modello di regressione lineare assume che la funzione di regressione $f(x)=E(Y|X=x)$ sia lineare negli input $X_1, X_{2,\dots}, X_p$. I modelli lineari sono, di solito, semplici e

forniscono una descrizione adeguata ed interpretabile di come gli input hanno effetto sull'output *Y*.

Sia dato un vettore di input $X^T=(X_1, X_{2,\dots}, X_p)$ e si voglia predire una variabile di output *Y* a valori reali. Il modello di regressione lineare ha la forma

$$f(X)=\beta_0+\sum_{j=1}^{p} X_j\beta_j .$$

I parametri β_j sono i coefficienti della regressione, mentre le variabili X_j sono input ottenibili da diverse fonti (es. input quantitativi, trasformazioni di essi, espansioni di base, ecc.). Indipendentemente dalla fonte degli X_j, il modello è lineare nei suoi parametri.

Tipicamente, si ha un set di dati di training $(x_1, y_1)\dots(x_N, y_N)$ da cui si effettua una stima dei parametri β. Il metodo di stima più comune è quello dei minimi quadrati, in cui si scelgono i coefficienti $\beta=(\beta_0, \beta_{1,\dots.}, \beta_p)^T$ tali da minimizzare la somma dei residui al quadrato

$$RSS(\beta)=\sum_{i=1}^{N}(y_i-f(x_i))^2=\sum_{i=1}^{N}\left(y_i-\beta_0-\sum_{j=1}^{p} x_{ij}\beta_j\right)^2 .$$

È possibile dimostrare, tramite il teorema di Gauss-Markov, che lo stimatore dei minimi quadrati è quello avente la più piccola varianza tra gli estimatori lineari *unbiased*; esso stesso è *unbiased*, lineare ed efficiente nella classe degli stimatori aventi queste caratteristiche (dunque, è *BLUE*).

La funzione $RSS(\beta)$ è una funzione quadratica dei suoi parametri (convessa), dunque il minimo esiste sempre ma potrebbe non essere unico.

In questo caso, si può pensare di ricodificare e/o scartare le colonne di **X** (attributi) per rendere i coefficienti univocamente definiti; altrimenti, l'insieme delle soluzioni $\boldsymbol{B}$ ne contiene infinite, ma è comunque sempre possibile scegliere la soluzione $\hat{\beta}^*$ di lunghezza minima in norma 2

$$\|\hat{\beta}^*\| \leqslant \|\hat{\beta}\|, \forall\, \hat{\beta} \in \boldsymbol{B} \;\; et \;\; \hat{\beta}^* \neq \hat{\beta}\,.$$

Tale soluzione vale

$$\hat{\beta}^* = \boldsymbol{X}^+ \boldsymbol{y}$$

dove $\boldsymbol{X}^+$ è la matrice *pseudo-inversa* di Moore-Penrose di $\boldsymbol{X}$.

È possibile utilizzare tecniche di *subset-selection* per migliorare l'accuratezza del modello, sacrificando un po' di bias in cambio di una varianza minore (*bias-variance tradeoff*). Tali tecniche consistono nel mantenere un sottoinsieme dei predittori e scartare i restanti, ottenendo anche un modello più interpretabile in caso di numero elevato degli stessi. Essendo però processi discreti (le variabili sono mantenute o scartate) il modello prodotto spesso mostra comunque alta varianza e, dunque, non ne riduce l'errore di predizione.

I metodi di regolarizzazione (*shrinkage methods*) invece non soffrono molto di questa elevata variabilità, e vengono spesso utilizzati per migliorare l'accuratezza del modello. Una di queste tecniche è quella della *ridge regression*. Il ridge regola i coefficienti della regressione imponendo una penalità sulla loro dimensione.
I coefficienti di ridge minimizzano una somma residua dei quadrati con penalità

$$\hat{\beta}^{ridge} = \underset{\beta}{argmin}\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2\}\,.$$

Qui, $\lambda \geq 0$ è un parametro di complessità che controlla l'ammontare della regolarizzazione: quanto maggiore esso è, tanto maggiore è la quantità di regolarizzazione.
Quando, in una regressione, sono presenti molte variabili correlate, i loro coefficienti potrebbero essere scarsamente determinati ed esibire alta varianza. Un grande coefficiente positivo in una variabile potrebbe essere annullato da un simile coefficiente negativo in una variabile correlata alla precedente. Imponendo un vincolo sulla dimensione dei coefficienti, questo problema è alleviato.

4.1.2 Reti neurali

Una *rete neurale artificiale* (per la regressione), comunemente chiamata soltanto rete neurale, è un modello di regressione a due stadi, tipicamente rappresentato da un diagramma di rete. La struttura della rete consta di alcuni livelli (*layer*) di *unità* o nodi.

Il primo layer consta di *p* nodi di input, assumendo di avere un vettore di input *X* con *p* componenti. Nel caso della regressione, la rete contiene un solo nodo di output Y_1 nell'ultimo livello.

I nodi degli strati intermedi (*hidden layers*) servono a costruire le *feature derivate* Z_m, ottenute mediante combinazioni lineari degli input; nello strato finale, l'output Y_1 è modellato come funzione di combinazioni lineari delle feature derivate Z_m. Poichè i valori Z_m non sono direttamente osservabili, tali nodi vengono chiamati *hidden units*.

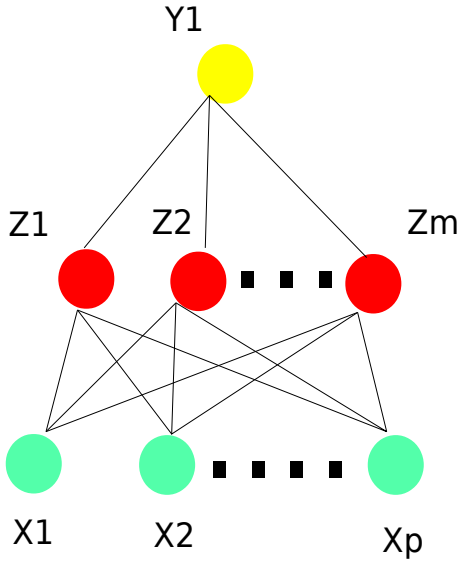

Fig. 4.1 Diagramma di rete di una MLP avente un singolo hidden layer, un layer di input e uno di output (costituito da un solo nodo nel caso della regressione).

Nel dettaglio

$$Z_m = \sigma(\alpha_{om} + \alpha_m{}^T X),\ m = 1,\ldots,M,$$
$$T_k = \beta_{0k} + \beta_k{}^T Z,\ k = 1,\ldots,K,$$
$$f_k(X) = g_k(T),\ k = 1,\ldots,K,$$

dove $Z = (Z_1, Z_{2,\ldots}, Z_M)$ e $T = (T_1, T_{2,\ldots}, T_K)$.

Per la funzione di attivazione $\sigma(v)$ viene spesso scelta la *sigmoide* $\sigma(v) = \frac{1}{1+e^{-v}}$,

mostrata in figura 4.1. Si noti che se fosse scelta come funzione di attivazione la funzione identità, l'intero modello diverrebbe un modello lineare negli input.

La funzione di output $g_k(T)$ permette un'ulteriore trasformazione del vettore di output T. Per la regressione, viene tipicamente scelta la funzione identità $g_k(T) = T_k$.

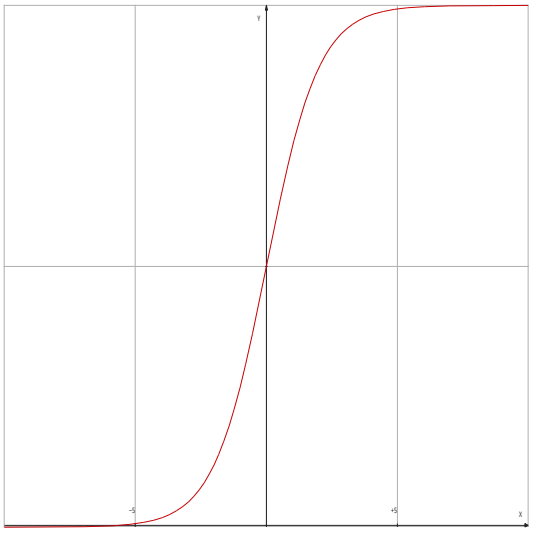

Fig. 4.2 Plot della funzione sigmoide $\sigma(x)=1/(1+e^{-x})$ *, usata come funzione di attivazione delle hidden unit.*

Il modello di rete neurale ha alcuni parametri da definire, chiamati pesi (*weights*). Denotiamo con θ l'insieme completo dei pesi, che comprende

$$\{\alpha_{0m}, \alpha_m;\ m=1,2,\ldots,M\}\ M(p+1) \text{ pesi},$$
$$\{\beta_{0k}, \beta_k;\ k=1,2,\ldots,K\}\ K(M+1) \text{ pesi}.$$

Per la regressione, viene utilizzata la somma degli errori quadrati come misura della stima dei pesi

$$R(\theta)=\sum_{k=1}^{K}\sum_{i=1}^{N}(y_{ik}-f_k(x_i))^2 .$$

Tipicamente, si tende a non voler minimizzare $R(\theta)$ in modo globale, in quanto questo porterebbe ad un modello *overfitted* sui dati di training. L'approccio generale per minimizzare tale funzione consiste nell'utilizzare il *gradient descent*, chiamato *back-propagation* in queste condizioni. Tale processo consta di due fasi, la *propagazione* e

l'*aggiornamento dei pesi*. Ad ogni passo, il valore di output viene confrontato con quello reale per calcolare il valore della funzione d'errore. Tramite alcune tecniche, l'errore è ri-propagato nella rete e i pesi vengono modificati in modo opportuno per abbassare il valore dell'errore.
Questo passo è ripetuto per un numero sufficiente di cicli di training (*training cycles*), fino alla convergenza della rete in uno stato in cui l'errore è sufficientemente piccolo.

Una rete in cui le connessioni tra i nodi non formano un ciclo diretto viene detta *feed-forward neural network*.[21]
Una rete di questo tipo, avente tutte le caratteristiche sopraelencate e diversi livelli di nodi connessi in un grafo diretto, è detta *multi-layer perceptron* (*MLP*).[21]

Spesso le reti neurali tendono ad avere un numero elevato di pesi, il che comporta un modello *overfitted* sui dati di training. Un metodo esplicito di regolarizzazione è il cosiddetto *weight decay*, che consiste nell'aggiungere un termine di penalità alla funzione dell'errore $R(\theta)+\lambda J(\theta)$, dove

$$J(\theta)=\sum_{km}\beta_{km}^2+\sum_{ml}\alpha_{ml}^2$$

e $\lambda \geq 0$ è un parametro di tuning. Maggiori valori di λ tendono a regolarizzare i pesi verso lo zero.

Parlando della struttura delle reti da adottare, generalmente è meglio avere un numero elevato di unità nascoste piuttosto che un numero troppo basso. Con pochi nodi nascosti il modello potrebbe non avere la necessaria flessibilità per catturare la non-linearità dei dati; avendo un numero elevato di essi, invece, i pesi extra possono essere regolarizzati verso lo zero utilizzando tecniche quali il weight decay.

Il numero di hidden units, per ciascun hidden layer, considerato in questo lavoro di tesi è pari a $(number\ of\ attributes + number\ of\ outputs)/2+1$, dove *number of outputs* è pari a 1 nei problemi di regressione.

La notazione adottata per descrivere le reti neurali nel presente elaborato è la seguente: *NN[layer1_size,layer2_size,...,layerN_size]*, e quando il numero di nodi di un layer è pari a quello illustrato precedentemente, si indicherà tale valore con il termine *Standard*.

4.1.3 Random forest[22]

Il *random forest* è un modello che comprende un numero specificato di *random trees*, i quali sono costruiti e allenati su sottoinsiemi *bootstrapped* del dataset di input.[21]

Un metodo di *bootstrap* applicato su un dataset di training consiste nel'estrarre campioni casuali con rimpiazzamento dal dataset di training. Denotiamo tale dataset con $\boldsymbol{Z}=\{z_1, z_{2,...}, z_N\}$ con $z_i=(x_i, y_i),\ i=1,2,...,N$. Qui x_i è un input 1-dimensionale, mentre y_i è la variabile dipendente, reale e continua nel caso della regressione. Sia $M<N$ la dimensione del sottoinsieme da estrarre dal dataset $\boldsymbol{Z}$. Viene dunque costruito il dataset $\boldsymbol{Z}^*$ estraendo casualmente con rimpiazzo *M* campioni z_i dal dataset $\boldsymbol{Z}$.

Sia *B* il numero scelto di alberi randomici che comporranno la foresta. Per ciascun *b*, con $b=1,2,...B$, viene costruito un dataset $\boldsymbol{Z}^*$ con il metodo di bootstrap sovradescritto; un albero T_b viene costruito e allenato su tale dataset.
Dopo *B* passi, viene restituito in output l'insieme degli alberi $\{T_b\}_1^B$.

La costruzione ed il training degli alberi avviene ripetendo i seguenti passi per ogni nodo terminale dell'albero, fino a che non è raggiunta la dimensione minima del nodo n_{min} :

i. Vengono prese casualmente $m<p$ variabili, scelte dalle *p* di input.

ii. Si sceglie la migliore variabile (*punto di split*) tra le *m*, secondo un criterio specifico che, nel caso della regressione, è quello dei minimi quadrati.
iii. Il nodo corrente viene diviso, generando due nodi figli.

Nel caso della regressione, gli inventori dell'algoritmo suggeriscono i seguenti valori per m e n_{min} : $m=\lfloor p/3 \rfloor$, mentre la dimensione minima dei nodi vale 5.

Dopo che sono stati costruiti i *B* alberi della foresta $\{T_b\}_1^B$, il predittore per la regressione del modello random forest è

$$\hat{f}_{rf}^{\,B}(x)=\frac{1}{B}\sum_{b=1}^{B} T_b(x) .$$

È facilmente intuibile che riducendo *m* verrà ridotta anche la correlazione tra i vari alberi della foresta, e di conseguenza la varianza della media.

Un'importante caratteristica dei modelli random forest è l'uso delle osservazioni *out-of-bag* (OOB). In pratica, per ogni osservazione $z_i=(x_i, y_i)$ viene costruito il suo predittore mediando soltanto gli alberi costruiti a partire dai bootstrapped dataset in cui z_i non compare.

In ultimo, si vogliono dare alcune considerazioni sulle performance di tali modelli. Quando il numero di variabili *p* è alto, ma il numero di variabili significative è piccolo, i random forest tendono a performare male con piccoli valori di *m*. Questo perché la possibilità che, ad ogni split, vengano scelte casualmente le variabili significative è molto bassa.
Un'altra caratteristica di tali modelli che viene spesso paventata è che i random forest non risentono di overfitting sui dati di training. È certamente vero che l'aumento del numero di alberi *B* nella foresta non causa overfitting, ma il mediare tanti alberi pienamente "cresciuti" (molto estesi) può portare ad un modello troppo ricco che

esibisce una varianza eccessiva. Spesso, dunque, si utilizza un valore di *profondità massima* degli alberi per fermarne la crescita al raggiungimento di una soglia definita.

4.1.4 Support Vector Machines per la regressione

Per quanto riguarda tale classe di modelli, si vuole prima descrivere il Support Vector Classifier, per poi adattare i concetti fondamentali di tale classificatore nel caso della regressione.

Nel caso della classificazione, si vuole ricavare un iperpiano che separi i dati di training appartententi a due classi perfettamente separabili, supponendo che lo siano linearmente. Il suddetto iperpiano è definito come $\{x : f(x) = x^T \beta + \beta_0 = 0\}$.
Poichè è possibile scegliere un numero elevato di iperpiani che separino le due classi, è necessario un criterio di scelta ottimale; viene dunque scelto l'iperpiano che massimizza il margine M tra i dati di training appartenenti alle due classi.
Supponendo adesso che le classi sia sovrapposte nello spazio degli attributi (mantenendo l'ipotesi che siano separabili linearmente), il criterio di scelta dell'iperpiano viene dunque modificato, permettendo ad alcune osservazioni di trovarsi nel lato sbagliato dell'iperpiano (*misclassified*).
Questo problema di ottimizzazione può essere scritto come

$$min\|\beta\| \text{ soggetto ai vincoli } y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i \ \ \forall i \ \wedge \ \ \xi_i \geq 0, \sum \xi_i \leq cost \ .$$

Il margine M è stato definito come $1/\|\beta\|$, mentre le $\xi_i \in \{\xi_1, \xi_2, \ldots, \xi_N\}$ sono variabili di slack.
È possibile dimostrare che la soluzione per β ha la forma

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i \ ,$$

dove i coefficienti $\hat{\alpha}_i$ sono diversi da zero solo per un sottoinsieme delle osservazioni *i*-esime. Queste osservazioni sono chiamate vettori di supporto (*support vectors*) poiché $\hat{\beta}$ è rappresentato solo in termini di essi. Sono dunque tali osservazioni che contribuiscono a dare la forma all'iperpiano di separazione, a differenza di quelle molto ben classificate che contribuiscono scarsamente.

Nel caso in cui i dati di training non siano linearmente separabili, è possibile trasformare i dati in modo da mapparli in un altro spazio. È possibile scrivere la funzione *f(x)* (non lineare) come

$$f(x)=\sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i)\rangle + \beta_0 ,$$

dove $h(x_i)$ sono le feature trasformate da una funzione di base *h(x)*. Scegliendo la funzione *h(x)* in modo consono, è possibile calcolare i prodotti interni $\langle h(x), h(x_i)\rangle$ in modo molto semplice. In particolare, non occorre neppure conoscere la trasformazione *h(x)*, ma solo la cosiddetta funzione kernel utilizzata

$$K(x, x') = \langle h(x), h(x')\rangle$$

la quale calcola il prodotto interno nello spazio trasformato.

Alcune tra le funzioni kernel molto utilizzate in letteratura sono

$$\textit{Kernel dot}:\ K(x, x') = \langle x, x'\rangle ,$$

$$\textit{Kernel polinomiale}:\ K(x, x') = (1 + \langle x, x'\rangle)^d \text{ , con } d \text{ il grado del polinomio.}$$

Tutto quando detto in precedenza è adattabile al caso dei problemi di regressione, con le opportune differenze. Nel caso lineare della regressione, $f(x)=x^T\beta+\beta_0$, si stima β minimizzando

$$H(\beta,\beta_0)=\sum_{i=1}^{N} V(y_i-f(x_i))+\frac{\lambda}{2}\|\beta\|^2,$$

dove

$$V_\epsilon(r)=\text{ o se } |r|<\epsilon,\ |r|-\epsilon \text{ altrimenti.}$$

L'ultima è una funzione di ε-insensibilità all'errore, in quanto ignora tutti gli errori aventi dimensione minore di ε. Questa considerazione rappresenta una sorta di analogia al classificatore a vettori di supporto, dove i punti classificati correttamente e lontani dal margine vengono ignorati nel processo di ottimizzazione.
Se $\hat{\beta},\beta_0$ sono le soluzioni che minimizzano *H*, si può mostrare che la funzione che risolve il problema ha la forma

$$\hat{f}(x)=\sum_{i=1}^{N}(\hat{\alpha_i^*}-\hat{\alpha}_i)\langle x,x_i\rangle+\beta_0$$

con $\hat{\alpha_i^*}$, $\hat{\alpha}_i$ valori positivi. Anche qui, soltanto un sottoinsieme di tutti i valori $(\hat{\alpha_i^*}-\hat{\alpha}_i)$ conterrà valori diversi da zero, e i valori associati costituiscono i vettori di supporto.

Per i dati di training il cui modello di regressione non è lineare, è possibile estendere quanto ricavato per il caso lineare.
Come nella classificazione, la soluzione dipende soltanto dal prodotto interno $\langle x,x_i\rangle$, dunque è possibile generalizzare tali metodi a spazi più ricchi definendo un appropriato prodotto interno. L'uso di funzioni kernel come quelle descritte precedentemente permette, ancora, di ottenere tale prodotto interno già mappato in un altro spazio ed è facile dimostrare che i valori predetti per un arbitrario *x* soddisfano

$$\hat{f}(x)=\sum_{k=1}^{N} \hat{\alpha}_i K(x, x_i) .$$

4.2 Regressione sui dataset strutturati

L'obiettivo è predire il prezzo massimo raggiunto da uno stock (es. $\langle h_{s_j}, \overline{h_{s_j}}^{k} \rangle$) nel timestamp successivo (es. t_k) al più recente tra quelli dei valori di input (es. $t_{k-1}, t_{k-2}, \ldots$). Poichè l'output della predizione è un valore reale continuo, il suddetto costituisce un problema di regressione. In aggiunta, i dati sono ordinati con rispetto al timestamp degli stessi, dunque il problema in oggetto rappresenta un problema di *time series forecasting*.

A questo scopo, si sono considerati i dati storici per ogni stock separatamente, campionati successivamente utilizzando tecniche di *windowing*, e sono stati allenati diversi modelli di regressione al fine di comprendere eventuali pattern.

4.3 Windowing

Il campionamento dei dati con una finestra a scorrimento [9] consiste nel campionare i dati storici delle azioni basandosi sui corrispondenti timestamp.

Per predire il valore di massimo $\langle h_{s_j}, \overline{h_{s_j}}^{k} \rangle$ dello stock s_j al timestamp t_k , viene considerata la lista dei dati storici dello stock nei precedenti timestamp. Più precisamente, data una dimensione fissa della finestra *W*, vengono considerati i dati storici dello stock ai timestamp $t_{k-1}, t_{k-2}, \ldots, t_{k-W}$. Una finestra a scorrimento viene utilizzata per cambiare il valore di *k* e la finestra viene spostata nel modo corrispondente.

Tabella 3
Windowing (W=1) applicato sui dati storici strutturati dello stock a.

a-High(t_{k-1})	a-Low(t_{k-1})	a-Volume(t_{k-1})	a-NumOfTrades(t_{k-1})	a-High(t_k)
51.5	50.2	100	20	51.6
51.6	51	78	12	51.4

In questo lavoro di tesi, si è utilizzato l'operatore *Windowing* di RapidMiner per effettuare il suddetto campionamento con finestra a scorrimento dei dati storici delle azioni.

4.4 Implementazione dei processi per il model learning

Per predire il valore del prezzo massimo per ogni azione, si sono applicate quattro tecniche di regressione (descritte in 4.1): regressione lineare, Random Forest, reti neurali e Support Vector Machines.

L'implementazione di tali algoritmi è quella offerta da RapidMiner e dai rispettivi operatori (si è utilizzata l'implementazione *mySVM* tra quelle disponibili per le SVMs).

4.4.1 Tuning dei parametri degli algoritmi

Per la scelta dei parametri ottimali dei vari metodi predittivi, si è generato un processo di calibrazione apposito.

Questo processo di tuning sfrutta l'operatore *OptimizeParameters(Evolutionary)* offerto da RapidMiner; tale operatore utilizza un algoritmo genetico per individuare il range ottimale di valori dei parametri relativi ai vari algoritmi.

Per raggiungere tempi accettabili di convergenza, si è deciso di utilizzare una popolazione ridotta di candidati (10) e un numero di generazioni non molto elevato (20); si è inoltre scelto di fermare l'algoritmo dopo 5 generazioni senza miglioramenti (*early stopping*) e di usare una probabilità di *crossover* leggermente minore di quella

proposta di default (0,7 invece che 0,9) per diminuire il rischio di rimanere fermi in un ottimo locale.

Come meccanismo di selezione, si è scelto il *tournament*, con una frazione di 0,2 partecipanti rispetto al totale della popolazione; con i parametri enunciati, è implicito affermare che soltanto due candidati per ogni generazione partecipano alla selezione, in modo da assicurare una certa varietà alla popolazione.

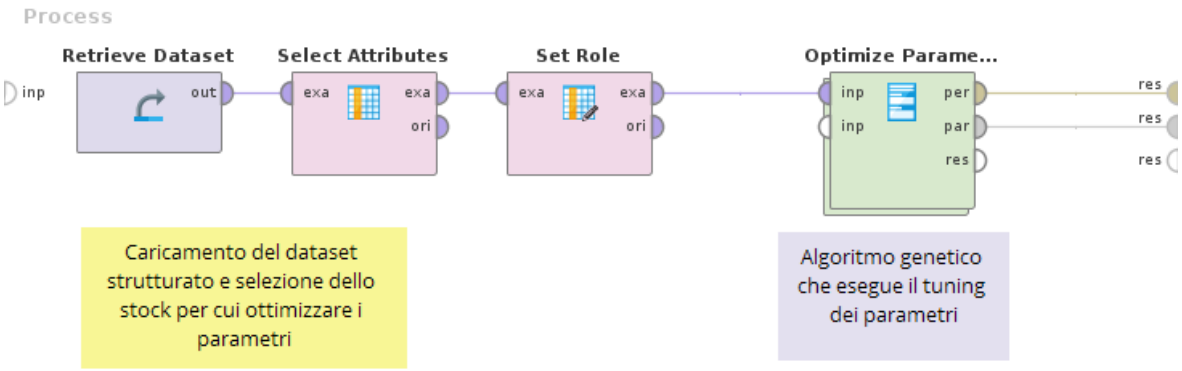

Fig. 4.3: Processo di tuning dei parametri degli algoritmi. L'operatore Select Attributes seleziona lo stock per cui eseguire l'ottimizzazione, mentre l'interno dell'operatore Optimize Parameters è analogo al processo di analisi descritto nel paragrafo successivo.

I parametri da ottimizzare variano in base all'algoritmo:

- metodo di selezione dei predittori (*Feature selection*), tolleranza minima per l'eliminazione dei predittori (*min_tolerance*) e ammontare della quantità di regolarizzazione (*ridge*) per la regressione lineare;
- numero di alberi decisionali (*number_of_trees*), il numero minimo di osservazioni per foglia (*minimal_leaf_size*), profondità massima degli alberi

(*maximal_depth*) e dimensione minima di un nodo per lo split (*minimal_size_for_split*) per Random Forest;
- valore di C e di *epsilon* per SVM con kernel dot;
- valore di C, grado del kernel (*kernel_degree*) e *espilon* per SVM con kernel polinomiale.

Per quanto riguarda le reti neurali, sono state testate diverse configurazioni di esse che differiscono per numero di hidden layers e di unità nascoste per ciascun livello, oltre che per la presenza o l'assenza di *weight decay*.

Il processo di calibrazione dei parametri è molto dispendioso in termini computazionali nonostante le considerazioni sull'algoritmo genetico descritte precedentemente. Poichè si è interessati ad ottenere modelli i cui errori relativi di predizione siano bassi in media per tutte le azioni, e visto l'elevato numero di stock presente in alcuni indici o mercati, si è deciso di ottimizzare i parametri dei modelli solo per le azioni caratterizzate da maggiore *volatilità*. Tali stock sono quelli con il più alto errore relativo di predizione, indipendentemente dal modello particolare scelto e dalle sue caratteristiche; dagli esperimenti effettuati nel presente elaborato (che verranno discussi successivamente) si è osservata la presenza costante di alcuni stock tra quelli aventi questa caratteristica, e si è dunque deciso di eseguire il task di ottimizzazione solo su di essi.

Le configurazioni di parametri ottenute da questo passo, sono state successivamente applicate ai modelli da allenare con i dati storici di tutte le altre azioni, ottenendo risultati comunque migliori rispetto all'applicazione delle configurazioni di default dei metodi predittivi.

In aggiunta a quanto detto, sono stati considerati molteplici valori per la dimensione (fissa) della finestra di campionamento a scorrimento (parametro *window_size* dell'operatore *Windowing* di RapidMiner).

4.4.2 Processo di training dei modelli

Sia *N* il numero delle azioni i quali dati storici sono contenuti nel dataset strutturato di input. Si devono dunque allenare e validare *N* modelli, uno per ciascuno stock, basandosi sui dati storici del singolo stock; quanto appena detto, va ripetuto per ciascuno dei metodi predittivi considerati.

Si è dunque progettato un processo di RapidMiner che itera sul dataset strutturato di input e seleziona i dati di una singola azione per volta, i quali sono input di un ulteriore processo che costituisce il processo di analisi vero e proprio.
Questo processo principale sfrutta gli operatori di *Loop* e *Select stock* (operatore *Execute script* rinominato) per la selezione di uno stock per volta, sfruttando alcune macro definite con gli operatori *Generate Macro* e *Set Macro* (le macro *%{stock}* e *%{wsize}*).

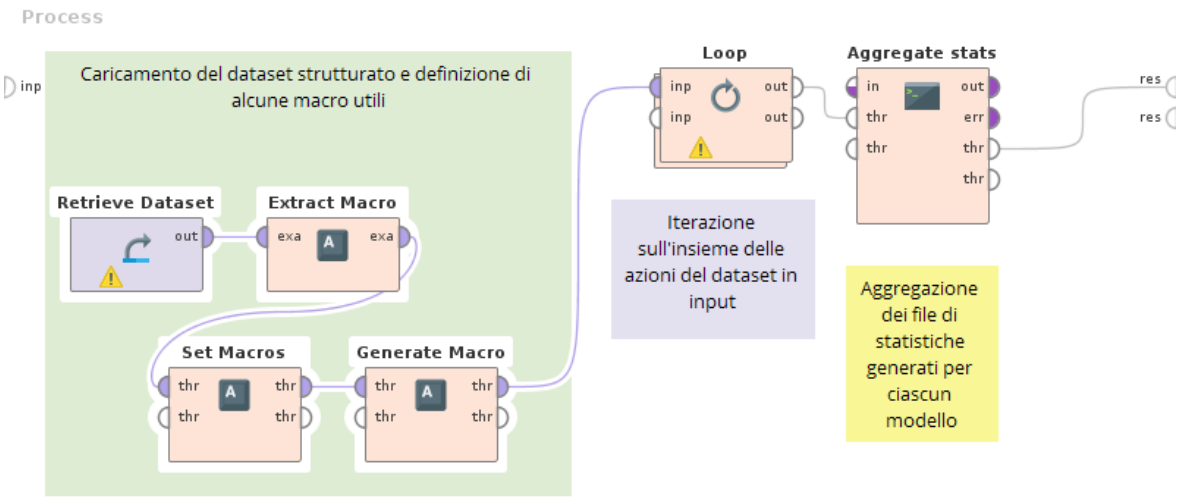

Fig. 4.4: Vista del processo di RapidMiner in carica di iterare sugli stock del dataset strutturato, eseguire il processo di analisi per ciascuno di essi e aggregare le statistiche sulle performance dei modelli generati.

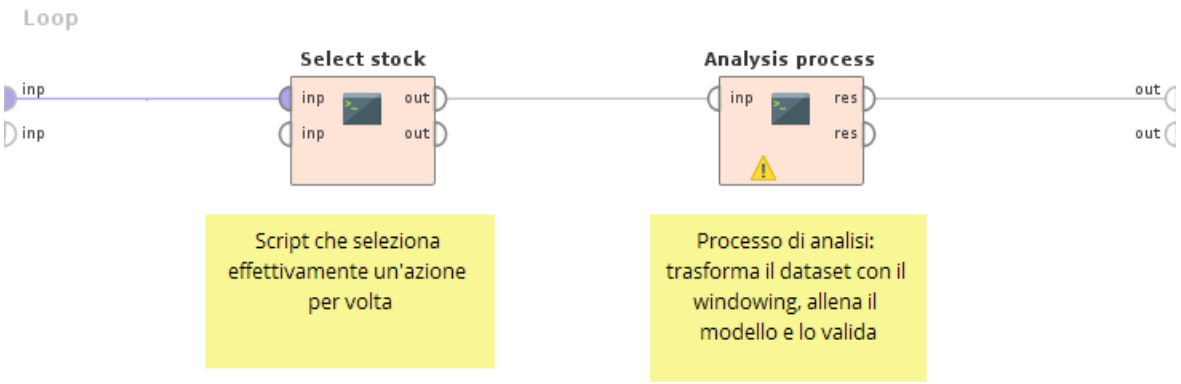

Fig. 4.5: Vista interna dell'operatore Loop.

Il processo di analisi trasforma il dataset strutturato di input con l'operatore *Windowing*, applicando il campionamento con finestra a scorrimento sui dati storici dello stock selezionato.

L'allenamento e la valutazione delle performance dei modelli avviene mediante l'operatore *SlidingWindowValidation*, il quale esegue automaticamente la divisione del dataset trasformato in *training set* e *test set*, applica il modello predittivo scelto al training set (eventualmente normalizzato) e ne valuta le performance di predizione utilizzando il test set (la validazione del modello verrà discussa nel paragrafo 4.6).
Le predizioni vengono memorizzate su disco mediante l'operatore *Write CSV* nel file *%{stock}.csv*, creato nella cartella *%{algo}* che raccoglie i risultati relativi all'algoritmo di regressione considerato; *%{stock}* è la macro che individua lo stock considerato e %

{algo} è la macro che individua l'algoritmo di regressione corrente, entrambe definite dall'operatore *Set Macro* nel processo principale.

Un operatore *Performance (Regression)* viene consegnato come output dell'operatore *SlidingWindowValidation*, contenente le misure dell'errore di predizione del modello secondo alcune metriche scelte. Esso viene infine convertito dall'operatore *Performance to Data* e scritto su file da un'altro operatore *Write CSV*. Il file è nominato *%{stock}Performance.csv*, ed è memorizzato nella stessa cartella delle predizioni (individuata dalla macro *%{algo}*).

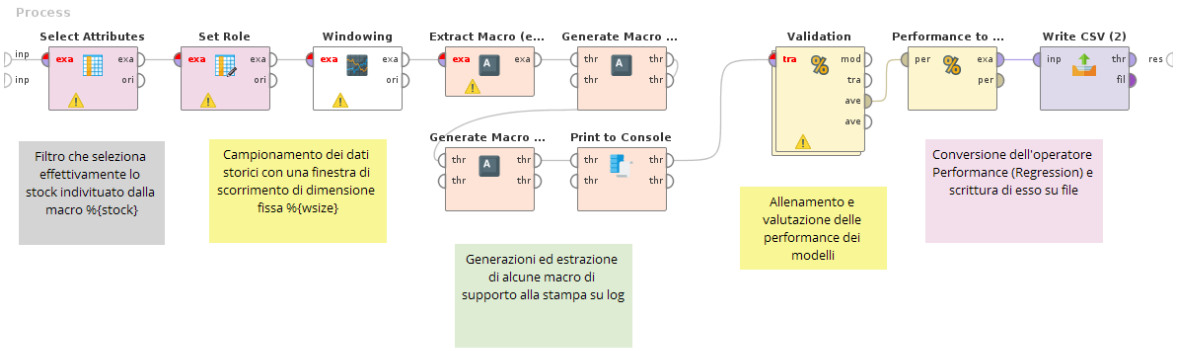

Fig. 4.6: Processo di analisi, che allena e valuta un modello sullo stock individuato dalla macro %{stock}. I dati storici dello stock vengono selezionati dal dataset strutturato e vengono trasformati dall'operatore Windowing, il quale applica il campionamento con finestra a scorrimento.

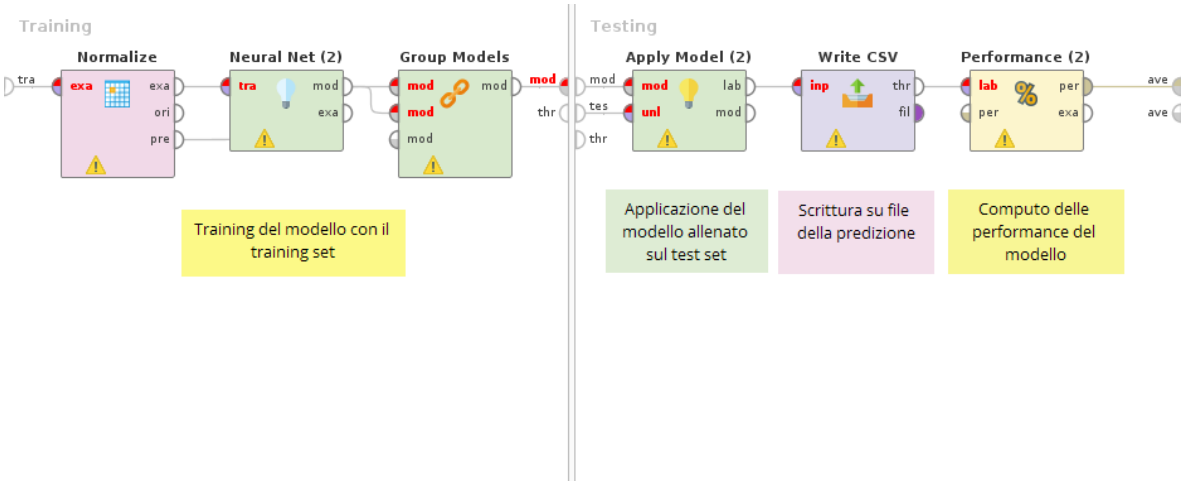

Fig. 4.7: Vista interna all'operatore SlidingWindowValidation. È possibile osservare la separazione tra le parti di allenamento e test del modello (in questo caso una rete neurale).

In ultimo, l'operatore *Aggregate stats* nel processo principale esegue uno script progettato appositamente per aggregare i file di statistiche *%{stock}Performance.csv* (uno per ciascuna azione) in un unico file CSV risultante, il file *aggregatedPerformances.csv*. Esso facilita l'analisi delle performance dei modelli sulle differenti azioni, permettendo di ordinarle in base ad una metrica scelta.

4.5 Metrica per la misura dell'errore di predizione

Per valutare le performance dei modelli generati si è utilizzato il *Mean Relative Error*, indicato d'ora in avanti come *MRE*.

Tale metrica ha valori in $\mathbb{R}_{\geq 0}$ ed è *scale-indipendent*, dunque utilizzabile per la comparazione tra azioni diverse delle performance dei modelli allenati.

È definita come

$$MRE = \frac{\sum_{t=1}^{T} \frac{|\hat{y}_t - y_t|}{y_t}}{T}$$

dove T è il periodo di predizioni considerato, $\hat{y}_t$ è il valore predetto della variabile dipendente per il timestamp t e y_t è il valore reale. La differenza $|\hat{y}_t - y_t|$ è detta *scarto*.

Nel presente elaborato viene sempre indicato il *mean absolute percentage error*, che è definito come

$$MAPE = MRE * 100\,\%$$

e corrisponde al valore di *MRE* in percentuale.

4.6 Validazione dei modelli

Per validare i modelli e stimarne il rendimento in termini di errore di predizione, si è scelto di utilizzare una validazione con finestra a scorrimento (*sliding window validation*).

La *K-fold Cross Validation*, uno dei metodi di validazione tra i più diffusi ed efficaci, risulta essere troppo naive nel caso di analisi di serie temporali. In presenza di dati sequenziali, infatti, l'assunzione di indipendenza statistica e di distribuzione identica delle osservazioni (propria delle tecniche di *apprendimento supervisionato*) non vale. Tale metodo ignora l'ordinamento naturale, in base al timestamp, dei dati e questo può portare (in alcuni split) all'utilizzo di dati futuri per il training del modello e alla conseguente predizione di dati passati; quanto detto risulta in una stima ottimistica del comportamento del modello, che invece mostrerebbe risultati peggiori qualora venisse adoperato in una situazione reale.

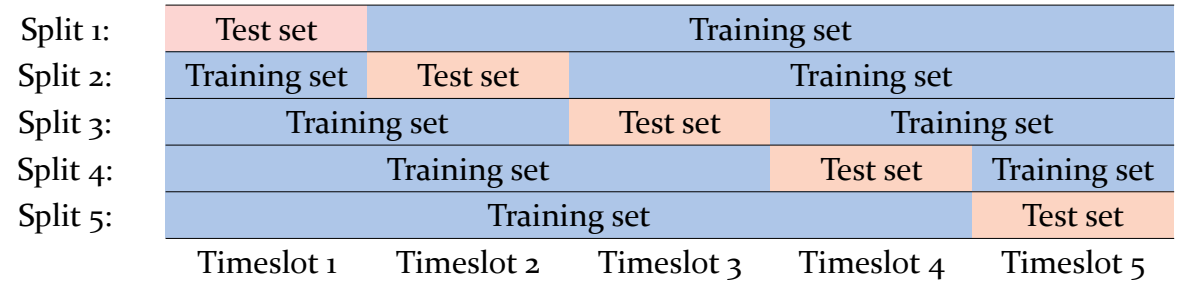

Fig. 4.8: Esempio grafico di una 5-fold Cross Validation.

Per evitare il suddetto comportamento (spesso indicato come *look-ahead bias*), si è utilizzato dunque uno *sliding window approach*, utilizzando due finestre a scorrimento per individuare il training set ed il test set a partire dal dataset strutturato iniziale trasformato dal *windowing*, ordinato in base al timestamp dei record.

L'approccio funziona specificando le dimensioni (fisse) delle finestre di training e test, oltre all'orizzonte della predizione. Siano *Wtraining*, *Wtest* e *H* tali valori, rispettivamente. Il modello è validato come segue:

1) vengono utilizzati, *Wtraining* osservazioni per il training del modello, prese tenendo conto dell'attributo *Timestamp* e partendo dalla prima disponibile;
2) la prima osservazione della finestra di test è quella ottenuta partendo da *H* osservazioni successive all'ultima della finestra di training; viene utilizzato il modello allenato per predire le *Wtest* osservazioni seguenti, a partire da tale valore;
3) si computano le performance del modello in termini di errore di predizione, utilizzando una metrica scelta (es. MRE);
4) la finestra di training è spostata in avanti di *Wtest* osservazioni e si ripete l'algoritmo dal punto 1).

La validazione prosegue fino ad esaurimento del dataset in ingresso, fino a che, cioè, non sarà più disponibile una finestra di test contenente un numero di osservazioni pari a *Wtest*.

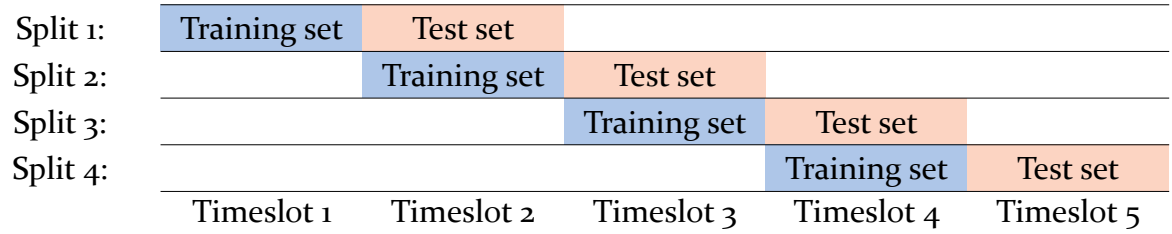

Fig. 4.9: Esempio grafico di una validazione con finestra a scorrimento.

Per il presente lavoro di tesi, si è utilizzato l'operatore *SlidingWindowValidation* di RapidMiner. In particolari, si è scelto *H* pari a 1, mentre i valori *Wtraining* e *Wtest* scelti verranno discussi nel capitolo che illustra i risultati sperimentali.

5. Raccomandazione delle azioni per l'investimento

L'ultima fase relativa al trading system sviluppato consiste nel consigliare ai day traders le azioni più appetibili su cui investire nei timestamp successivi all'ultimo timestamp di training dei modelli di regressione generati sui dati storici delle azioni.

Il periodo di predizione considerato per ciascuno dei suddetti modelli è di circa 8 giorni di mercato, in cui ogni giorno contiene un numero di slot di intervento potenziale sul mercato pari a 13, 7 e 3 per le frequenze di campionamento dei dati storici pari a trenta minuti, un'ora e due ore, rispettivamente. Dopo tale periodo di predizione, è consigliabile rigenerare i modelli con i nuovi dati, in modo da predire gli 8 giorni di mercato borsistico successivi. Maggiori dettagli verranno illustrati nella sezione dei risultati sperimentali.

Come già descritto nella sezione 4.2, per ciascuno stock è stato predetto il valore di prezzo massimo nei successivi slot di tempo basandosi sui dati storici dello stesso singolo stock, comprendenti diversi indicatori e misure per ciascun timestamp. È stata definita un'apposita funzione di *fitness* che permette di classificare gli stock ad ogni timestamp, in modo da poter raccomandare gli *n* stock più promettenti per l'investimento. Il valore di fitness per lo stock s_j al timestamp t_k vale

$$fitness_{s_j}^{k} = \frac{\hat{massimo}_{s_j}^{k} - chiusura_{s_j}^{k-1}}{chiusura_{s_j}^{k-1}} * 100\,\%$$

dove $\hat{massimo}_{s_j}^{k}$ è il valore predetto.

Il trading system sviluppato è in grado di raccomandare stock sia per le posizioni lunghe che per quelle corte. La funzione di fitness descritta funziona bene anche per le

posizioni corte, al patto di raccomandare gli ultimi *n* stock per l'investimento. Il che equivale a considerare la funzione di fitness

$$fitness\_short_{s_j}^{k} = -fitness_{s_j}^{k}$$

e raccomandare i primi *n* stock, per gli investimenti su posizioni corte.

Ogni transazione ha un costo, indicato con *transaction_cost*. Si deve tenere conto del costo della transazione nella raccomandazione degli stock, in quanto il guadagno effettivo potrebbe essere talmente basso da costituire una perdita proprio a causa di tale costo.
Anche se il valore di fitness, a cui viene sottratto il costo della transazione, dovesse essere positivo, potrebbe non essere conveniente raccomandare lo stock se tale differenza dovesse risultare troppo piccola, in quanto l'investimento sullo stock potrebbe portare un guadagno irrisorio. Occorre dunque definire una soglia di guadagno desiderato, chiamata *threshold*, in modo da scartare le raccomandazioni il cui fitness, diminuito del costo della transazione, non supera tale valore.

Considerato quanto detto, uno stock viene raccomandato (nel caso delle posizioni lunghe) solo se

$$fitness_{s_i}^{k} - transaction\_cost > threshold\ .$$

La stessa relazione vale nel caso delle posizioni corte, avendo cura di sostituire il valore della funzione di fitness con quello del fitness relativo alle posizioni corte.

Per ogni timestamp di predizione, vengono dunque computati i valori di fitness di tutte le azioni considerate, sia per le posizioni lunghe che per quelle corte, e le azioni per le quali i valori di fitness rispettano le condizioni sopraelencate vengono inserite in due

strutture dati ordinate in senso decrescente, che contengono gli stock raccomandabili per le posizioni lunghe e quelli raccomandabili per le posizioni corte, rispettivamente. Gli stock effettivamente raccomandati sono i primi *n* contenuti nelle strutture dati; in caso di pareggio, viene considerato l'ordine alfabetico dei simboli. Se le strutture dati sono vuote, non viene effettuata alcuna raccomandazione per quel dato timestamp. **Algoritmo 1** mostra l'algoritmo di raccomandazione delle azioni per il trading in pseudo-codice.

Algoritmo 1 Raccomandazione degli stock.

Input: Record $\hat{R}(t_k)$ relativo al timestamp di intervento sul mercato, contenente la lista delle predizioni delle azioni considerate
Input: Numero di stock da raccomandare *n*, soglia di guadagno *threshold*, costo delle transazioni *transaction_cost*
Output: Dizionario *D* contenente le due liste di azioni consigliate, una per le posizioni lunghe ed una per quelle corte

1: $L_{long} = \emptyset$
2: $L_{short} = \emptyset$
/ Dizionario di output, che conterrà le due chiavi "long" e "short" */*
3: $D \leftarrow \emptyset$
/ Calcolo del fitness di ogni stock nel record per le posizioni lunghe e corte */*
4: **for all** $\overline{\hat{h}_{s_j}^{k}}$ in $\hat{R}(t_k)$ **do**
/ Recupero del prezzo di chiusura dello stock s_j relativo al timestamp precedente t_{k-1} */*
5: $\overline{c_{s_j}^{k-1}}$ = readClosurePrice(s_j , t_{k-1})
/ Calcolo del fitness dello stock per posizione lunga/corta e inserimento nella relativa struttura dati */*
6: $fitness_{s_j}^{k}$ = computeFitnessL($\overline{\hat{h}_{s_j}^{k}}$, $\overline{c_{s_j}^{k-1}}$)
7: **if** $fitness_{s_j}^{k}$ - *transaction_cost* > *threshold*
8: insert s_j into L_{long}
9: **endif**
10: $fitness\_short_{s_j}^{k} = - fitness_{s_j}^{k}$
11: **if** $fitness\_short_{s_j}^{k}$ - *transaction_cost* > *threshold*
12: insert s_j into L_{short}
13: **endif**
14: **end for**
/ Generazione delle raccomandazioni degli stock per le posizioni lunghe e per le posizioni corte */*
/ Selezione dei primi n stock dalle strutture dati e inserimento nel dizionario di output */*
15: D["*long*"] = select(L_{long} , n)
16: D["*short*"] = select(L_{short} , n)
17: **return** *D*
18: **end**

Supponendo che venga raccomandato almeno uno stock per il trading in ogni slot temporale di una giornata di borsa, risulta evidente l'elevata frequenza di intervento sul mercato relativamente al singolo giorno. Per questa ragione, è consigliato scegliere $n=1$ nell'algoritmo, in modo da non incorrere in una spesa troppo elevata in costi di transazione, la quale potrebbe condurre a perdite di capitale.

Nella sezione relativa ai risultati sperimentali verrà valutato l'impatto dei costi di commissione sui rendimenti, stimando un costo della transazione in valore percentuale fisso (es. 0.15%). Si noti che tali costi dipendono dalle politiche dei broker, e la loro simulazione non fa parte degli obiettivi di questo lavoro di tesi.

6. Risultati sperimentali

In questo capitolo vengono presentati i risultati sperimentali ottenuti dall'applicazione del sistema sviluppato su dati reali degli stock, acquisiti tramite la IEX API. Tutti gli esperimenti sono stati eseguiti su un calcolatore non molto prestante, con un processore dual core Intel Corei3-5010U operante a 2.1 Ghz e 12 Gb di RAM, avente come sistema operativo Linux Mint 18.2. L'addestramento dei modelli e la generazione delle predizioni sono stati effettuati, come già detto precedentemente, mediante l'uso di RapidMiner con una licenza di tipo *Educational*, che permette l'utilizzo di un solo thread di processamento.

Nei paragrafi successivi verranno descritte nel dettaglio le caratteristiche dei dataset utilizzati e le scelte di design sperimentale. Verrà inoltre valutato il framework sotto tre aspetti: (i) redditività delle strategie adoperate, (ii) comportamento e precisione delle predizioni al variare della strategia adottata ed, infine, (iii) tempistiche necessarie al sistema per operare e scalabilità.

6.1 Dataset con i dati storici delle azioni

I dataset utilizzati per la sperimentazione sono dataset strutturati (definiti in 3.2), dove l'insieme delle grandezze misurabili relative alle azioni comprende i valori del prezzo massimo, del prezzo minimo, del volume di compravendita e del numero di transazioni, per ciascuna azione e per ciascun intervallo temporale considerato.

L'insieme degli stock comprende i dati storici di 496 azioni dell'indice *Standard & Poor's 500* (*S&P 500*), relativi ad un periodo di circa sei mesi. L'indice Standard & Poor's 500 è uno degli indici più significativi del mercato statunitense, ed è basato sulle capitalizzazioni di mercato di 500 grandi aziende degli indici *NYSE* e *NASDAQ*. Le aziende presenti nell'indice S&P 500 appartengono a tanti settori dell'economia

statunitense; al momento della scrittura del presente elaborato, i settori sono 11 e riguardano tecnologia, salute, finanza, industria ed altri ancora.

Il dataset di base, di cui sono state utilizzate le versioni con diverse aggregazioni, è la collezione *U.S. Feb-Jul 2018*, che contiene i dati storici delle azioni con frequenza di campionamento pari al minuto del periodo temporale che va dal giorno 22/02/2018 al giorno 27/07/2018.

Per la sperimentazione, sono state considerate le aggregazioni di tali dati in periodi di trenta minuti, un'ora e due ore. Si sono dunque generate le seguenti collezioni:

- *U.S Feb-Jul 2018-30m*. Contiene 1411 record con i valori del prezzo massimo, del prezzo minimo, del volume di compravendita e del numero di transazioni per ogni stock, dove ogni record rappresenta uno slot temporale di 30 minuti, indicato dal timestamp di fine periodo.
- *U.S Feb-Jul 2018-1h*. Contiene 760 record con i valori del prezzo massimo, del prezzo minimo, del volume di compravendita e del numero di transazioni per ogni stock, dove ogni record rappresenta uno slot temporale di un'ora, indicato dal timestamp di fine periodo.
- *U.S Feb-Jul 2018-2h*. Contiene 326 record con i valori del prezzo massimo, del prezzo minimo, del volume di compravendita e del numero di transazioni per ogni stock, dove ogni record rappresenta uno slot temporale di due ore, indicato dal timestamp di fine periodo.

6.2 Design sperimentale

Per valutare l'efficacia e l'efficienza del trading system proposto, si è utilizzata la validazione con finestra a scorrimento descritta in 4.6, in modo da misurare le performance di ciascun algoritmo in termini di errore di predizione e di redditività per ciascuno dei dataset a differente granularità. Nel dettaglio, è stata fissata una finestra di training di dimensione pari al 70% dell'intero dataset, e si è scelto di generare quattro

modelli per predire il restante 30% del dataset. Dunque, la finestra di test ha dimensioni pari al 7.5% dell'intero dataset.

Considerato quanto detto e la dimensione dei dataset precedentemente descritta, ogni modello è allenato di volta in volta con 986 record, 531 record e 227 record per i dati con granularità pari a 30 minuti, un'ora e due ore, rispettivamente. Questi record corrispondono ad un periodo di circa 78 giorni di mercato borsistico.
I record predetti da ciascun modello allenato sono, invece, 105/106, 56/57 e 24 per i dati con granularità pari a 30 minuti, un'ora e due ore, rispettivamente. Questi record corrispondono a circa 8 giorni di mercato borsistico.

La **Tabella 4** mostra il periodo di dati predetto complessivamente dai quattro modelli per ciascun algoritmo, al variare della granularità dei dati e della dimensione della finestra di campionamento. Si noti che tale periodo è variabile, seppur di poco; la variabilità è dovuta alla differenza in termini di numero di record dei tre dataset campionati con la finestra a scorrimento.

Tabella 4
Periodo complessivo dei dati predetti dai quattro modelli.

Granularità	Dimensione finestra	Periodo complessivo di predizione
30m	1	11/06 15:59 – 27/07 15:29
	2	11/06 15:59 – 27/07 13:59
	3	11/06 15:59 – 27/07 13:59
	4	11/06 15:59 – 27/07 13:59
1h	1	11/06 15:59 – 27/07 15:59
	2	11/06 15:59 – 27/07 15:59
	3	11/06 15:59 – 27/07 15:59
	4	12/06 10:29 – 27/07 12:29
2h	1	12/06 11:59 – 27/07 11:59
	2	
	3	
	4	

Per quanto riguarda la misura dell'errore di predizione delle strategie considerate, le performance dei quattro modelli ottenute in fase di validazione vengono mediate in modo da ottenere il valore di MAPE del metodo predittivo per il singolo stock. Per ottenere il valore medio di MAPE per la strategia considerata, tenendo conto di tutte le azioni, si calcola ulteriormente la media tra i valori di MAPE relativi ai modelli delle singole azioni.

Gli stessi dati predetti durante la validazione delle strategie in termini di errore di predizione vengono utilizzati per simulare l'investimento sulle azioni, in modo da valutare il guadagno possibile derivante dall'applicazione delle differenti strategie.

Per simulare le posizioni lunghe, si è deciso di acquistare lo stock consigliato dal framework all'apertura dell'intervallo per cui si è predetto il valore di prezzo massimo. Si calcola, dunque, il **guadagno potenziale**, ottenibile vendendo lo stock acquistato nel momento migliore dell'intevallo che coincide con il momento in cui lo stock ottiene il valore massimo. Per semplicità, si è posto il prezzo di acquisto uguale al valore del prezzo di apertura dello stock, mentre il prezzo di vendita è posto uguale al valore del prezzo massimo raggiunto dallo stock nell'intervallo.
Similmente, per simulare le posizioni corte si è assunto di vendere lo stock consigliato all'apertura dell'intervallo di predizione, e di acquistarlo nel momento migliore dell'intervallo, coincidente con il momento in cui lo stock assume il valore del prezzo minimo.
Se la chiusura di una posizione non dovesse risultare in un profitto potenziale per un determinato slot temporale d'intervento, si assume che la posizione sia stata chiusa alla fine dell'intervallo, al prezzo di chiusura dello stock; questa scelta è determinata dal fatto che la raccomandazione risulterebbe errata (es. nel caso delle posizioni lunghe, il valore del prezzo massimo sarebbe uguale al prezzo d'acquisto dell'azione, o di poco superiore, ma costituirebbe comunque una perdita a causa del costo di transazione), e si ipotizza che nessun day trader chiuderebbe la posizione quando il valore dell'azione

consigliata non è sufficiente a realizzare un guadagno: in questo caso dunque si aspetterebbe fino al termine dello slot d'intervento per chiudere la posizione.

Per semplicità, le posizioni lunghe e quelle corte sono state testate separatamente. Inoltre, si è posto il costo della transazione pari allo 0,15% e la soglia di guadagno desiderato pari all'1%. Si è inoltre impostata una soglia di stop loss pari allo 0,5%, la quale permette la chiusura automatica di una posizione se la perdita percentuale supera il valore definito.

Si sono dunque generate le raccomandazioni basandosi sulla predizione dei diversi modelli di regressione, e si è calcolato il profitto potenziale medio per timestamp di ogni strategia basandosi sui prezzi massimi, su quelli minimi, sui volumi di compravendita e sulle quantità di scambi delle azioni.

Per calcolare il profitto potenziale medio per timestamp, si è eseguita ogni strategia una volta per ciascun timestamp predetto dai modelli allenati, usando come periodo di training i circa 78 giorni di mercato borsistico precedenti e rigenerando i modelli dopo un numero di predizioni che copre un periodo di durata pari a circa 8 giornate di mercato. Si è scelto, inoltre, di raccomandare una sola azione per ciascuno slot d'investimento. I profitti derivanti da slot in cui viene raccomandata un'azione vengono mediati, differentemente dagli slot per cui non si è ottenuta una raccomandazione che vengono invece scartati.

6.3 Valutazione delle strategie in termini di profitto

Le tabelle da **Tabella 5** a **Tabella 10** mostrano i risultati ottenuti da tutte le strategie sia per le posizioni lunghe che per quelle corte, sui dataset con i dati a diversa granularità descritti precedentemente. La qualità delle strategie proposte è stata valutata in termini di metriche matematiche standard, quali media e deviazione standard, calcolate sui profitti potenziali ottenibili dalle azioni consigliate per ciascuno slot d'investimento.

Il guadagno potenziale, per le posizioni lunghe, di un generico stock acquistato all'apertura dello slot t_k è quello ottenibile se si vendesse lo stock nel miglior momento dell'intervallo t_k , che corrisponde al momento in cui lo stock assume valore massimo. Tale guadagno è dato da

$$\frac{massimo(t_k) - apertura(t_k)}{apertura(t_k)} - costo\_transazione\ .$$

Similmente, per le posizioni corte, il guadagno potenziale di un generico stock venduto all'apertura dello slot t_k è quello ottenibile se si riacquistasse lo stock nel peggior momento dell'intervallo t_k , che corrisponde al momento in cui lo stock assume valore minimo. Tale guadagno è dato da

$$\frac{apertura(t_k) - minimo(t_k)}{minimo(t_k)} - costo\_transazione\ .$$

In entrambe le definizioni di guadagno potenziale, si è fissato il costo della transazione pari allo 0,15%.

Le tabelle mostrano il profitto medio per timestamp, la sua deviazione standard, il numero totale di raccomandazioni e il profitto massimo ottenuto, al variare della finestra di campionamento dei dati, per ciascuna granularità dei dati. Non viene mostrata la perdita massima, in quanto equivalente al valore di stop loss.
Il profitto potenziale medio per timestamp e la sua deviazione standard consentono di valutare ciascuna strategia in base alla sua redditività e alla sua variabilità, rispettivamente. È preferibile scegliere strategie con alti valori di guadagno medio per timestamp e con bassi valori di deviazione standard.
Il numero di raccomandazioni è una statistica che indica quanto frequentemente il sistema è riuscito a raccomandare un'azione per l'investimento nel periodo temporale

considerato per la simulazione degli investimenti. Si noti che un numero ingente di raccomandazioni non comporta necessariamente un guadagno elevato, in quanto la stop loss e i costi delle transazioni potrebbero ridurre il guadagno sensibilmente.

Tabella 5
Dati con granularità pari a 30 minuti. Sono mostrati i profitti potenziali medi per timestamp (%), le loro deviazioni standard, il numero di raccomandazioni e il profitto potenziale massimo (%), investendo con posizioni lunghe e al variare della finestra di campionamento dei dati.

U.S Feb-Jul 2018-30m					
Posizioni lunghe					
	W	Profitto potenziale medio per timestamp (%)	Deviazione standard	# di raccomandazioni	Profitto potenziale massimo (%)
Regressione lineare	1	0,223	0,582	255	2,328
	2	0,208	0,552	393	2,513
	3	0,228	0,567	413	2,716
	4	0,226	0,585	419	3,891
NN[Standard]	1	0,200	0,609	424	2,754
	2	0,198	0,601	420	2,754
	3	0,154	0,558	414	2,716
	4	0,145	0,584	419	2,754
NN[Standard, Standard]	1	0,232	0,603	392	2,754
	2	0,183	0,584	408	3,788
	3	0,191	0,608	418	2,754
	4	0,200	0,600	416	2,754
Random Forest	1	0,183	0,591	424	2,780
	2	0,162	0,573	420	2,754
	3	0,187	0,588	420	2,780
	4	0,171	0,547	420	2,754
SVM dot	1	0,226	0,556	217	2,383
	2	0,274	0,617	214	3,307
	3	0,201	0,578	233	2,576
	4	0,244	0,544	263	2,576
SVM polinomiale	1	0,054	0,479	424	3,389
	2	0,036	0,476	420	3,389
	3	0,023	0,447	420	2,554
	4	-0,046	0,385	420	3,788

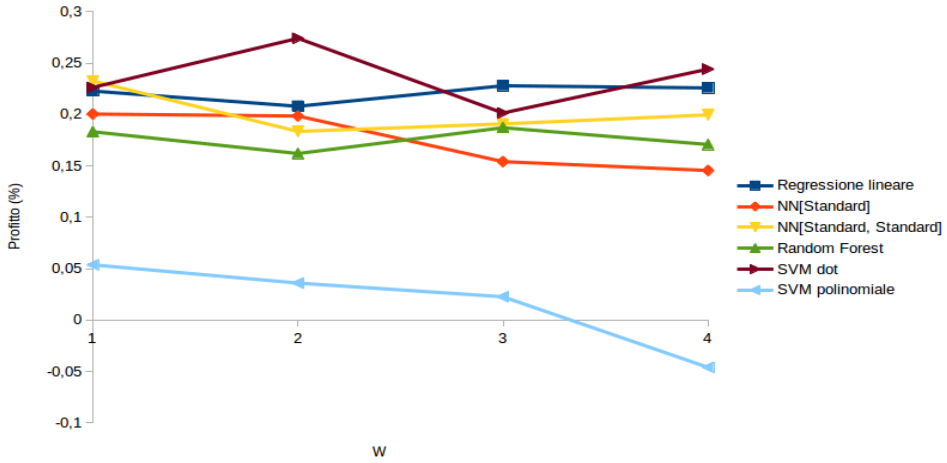

Fig. 6.1 Confronto tra i profitti medi percentuali per timestamp ottenuti dai metodi predittivi con posizioni lunghe, al variare della finestra di campionamento. Dati con granularità pari a 30 minuti.

Tabella 6
Dati con granularità pari a 30 minuti. Sono mostrati i profitti potenziali medi per timestamp (%), le loro deviazioni standard, il numero di raccomandazioni e il profitto potenziale massimo (%), investendo con posizioni corte e al variare della finestra di campionamento dei dati.

U.S Feb-Jul 2018-30m

Posizioni corte

	W	Profitto potenziale medio per timestamp (%)	Deviazione standard	# di raccomandazioni	Profitto potenziale massimo (%)
Regressione lineare	1	0,454	0,882	104	5,643
	2	0,430	0,733	192	4,821
	3	0,301	0,707	297	4,821
	4	0,196	0,662	374	4,821
NN[Standard]	1	0,033	0,460	394	2,454
	2	0,038	0,471	387	2,454
	3	0,055	0,474	364	2,454
	4	0,039	0,462	378	2,454
NN[Standard, Standard]	1	0,042	0,471	416	2,454
	2	0,053	0,478	408	2,454
	3	0,029	0,453	413	2,454
	4	0,028	0,456	414	2,454
Random Forest	1	0,070	0,487	424	2,454
	2	0,076	0,481	420	2,454
	3	0,065	0,485	420	2,454
	4	0,066	0,474	420	2,454
SVM dot	1	0,009	0,678	290	7,271
	2	0,359	0,866	32	3,379
	3	-0,027	0,430	218	3,379
	4	0,325	0,858	71	3,379
SVM polinomiale	1	-0,028	0,367	421	2,181
	2	-0,012	0,450	420	3,379
	3	-0,022	0,474	420	4,434
	4	-0,034	0,357	420	3,191

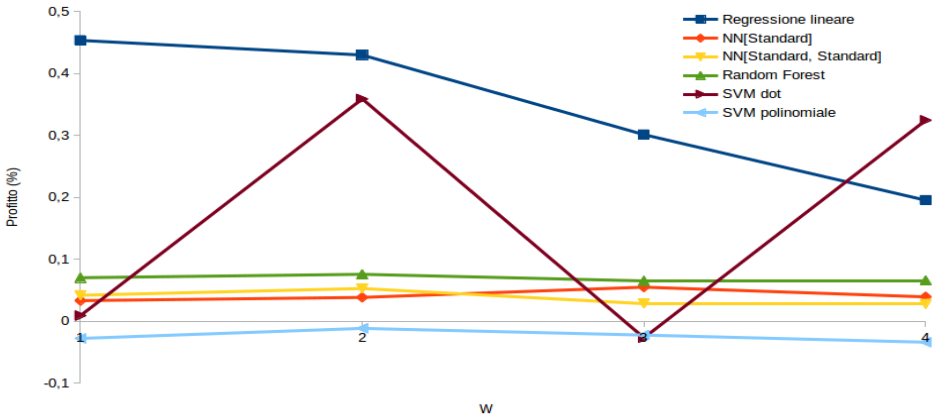

Fig. 6.2 Confronto tra i profitti medi percentuali per timestamp ottenuti dai metodi predittivi con posizioni corte, al variare della finestra di campionamento. Dati con granularità pari a 30 minuti.

Tabella 7
Dati con granularità oraria. Sono mostrati i profitti potenziali medi orari (%), le loro deviazioni standard, il numero di raccomandazioni e il profitto potenziale massimo (%), investendo con posizioni lunghe e al variare della finestra di campionamento dei dati.

U.S Feb-Jul 2018-1h

Posizioni lunghe

	W	Profitto potenziale medio per timestamp (%)	Deviazione standard	# di raccomandazioni	Profitto potenziale massimo (%)
Regressione lineare	1	0,334	0,749	216	4,282
	2	0,357	0,769	228	4,943
	3	0,396	0,785	228	4,943
	4	0,388	0,702	224	4,282
NN[Standard]	1	0,413	0,759	228	3,771
	2	0,397	0,805	228	4,943
	3	0,405	0,813	228	4,943
	4	0,365	0,818	224	4,943
NN[Standard, Standard]	1	0,444	0,796	228	4,943
	2	0,370	0,734	228	3,771
	3	0,416	0,750	228	3,771
	4	0,358	0,706	224	3,771
Random Forest	1	0,401	0,763	228	3,771
	2	0,410	0,767	228	3,771
	3	0,428	0,789	228	3,771
	4	0,416	0,789	224	3,771
SVM dot	1	0,216	0,658	228	3,771
	2	0,097	0,523	228	3,771
	3	0,000	0,402	228	2,844
	4	0,013	0,372	224	1,312
SVM polinomiale	1	0,401	0,778	194	4,282
	2	0,156	0,577	218	3,771
	3	0,052	0,564	228	4,808
	4	0,126	0,475	222	2,208

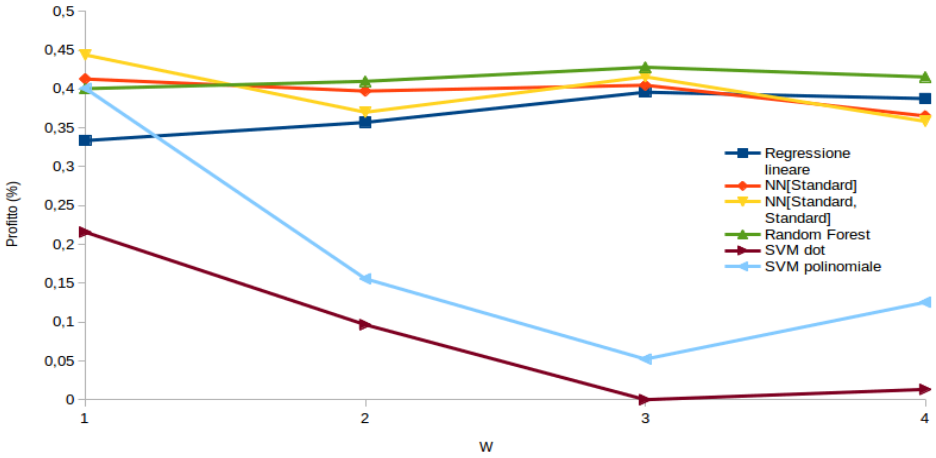

Fig. 6.3 Confronto tra i profitti medi percentuali per timestamp ottenuti dai metodi predittivi con posizioni lunghe, al variare della finestra di campionamento. Dati con granularità oraria.

Tabella 8
Dati con granularità oraria. Sono mostrati i profitti potenziali medi orari (%), le loro deviazioni standard, il numero di raccomandazioni e il profitto potenziale massimo (%), investendo con posizioni corte e al variare della finestra di campionamento dei dati.

U.S Feb-Jul 2018-1h

Posizioni corte

	W	Profitto potenziale medio per timestamp (%)	Deviazione standard	# di raccomandazioni	Profitto potenziale massimo (%)
Regressione lineare	1	0,536	0,765	77	3,256
	2	0,427	0,929	159	7,889
	3	0,359	0,645	205	2,942
	4	0,356	0,903	220	8,945
NN[Standard]	1	0,148	0,548	217	3,563
	2	0,163	0,630	217	3,563
	3	0,171	0,653	213	3,563
	4	0,174	0,668	211	3,563
NN[Standard, Standard]	1	0,114	0,540	226	3,563
	2	0,170	0,603	225	3,563
	3	0,144	0,584	228	3,563
	4	0,171	0,615	224	3,563
Random Forest	1	0,139	0,543	228	3,249
	2	0,148	0,553	228	3,249
	3	0,164	0,597	228	3,249
	4	0,150	0,553	224	3,249
SVM dot	1	0,543	0,879	44	3,256
	2	0,134	0,534	141	3,256
	3	0,070	0,470	228	1,986
	4	0,014	0,386	224	2,199
SVM polinomiale	1	0,660	0,923	31	3,256
	2	0,144	0,753	159	6,573
	3	0,109	0,530	199	3,663
	4	0,086	0,532	186	3,256

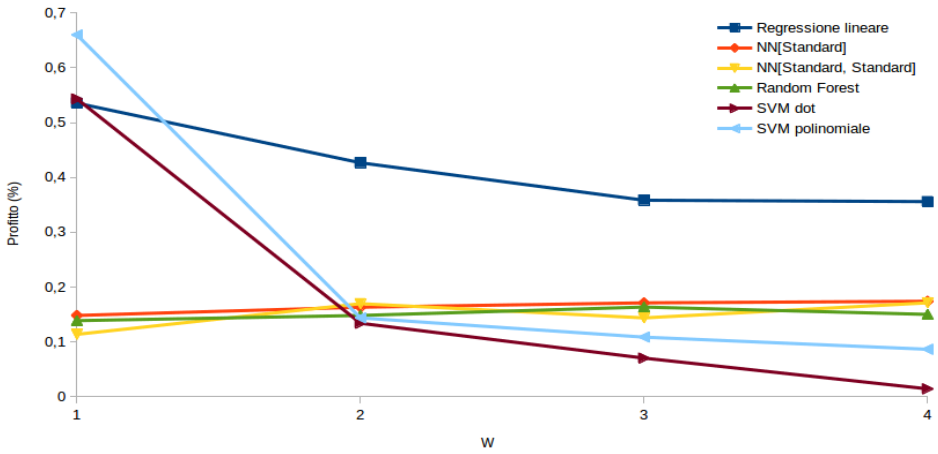

Fig. 6.4 Confronto tra i profitti medi percentuali per timestamp ottenuti dai metodi predittivi con posizioni corte, al variare della finestra di campionamento. Dati con granularità oraria.

Tabella 9
Dati con granularità pari a due ore. Sono mostrati i profitti potenziali medi per timestamp (%), le loro deviazioni standard, il numero di raccomandazioni e il profitto potenziale massimo (%), investendo con posizioni lunghe e al variare della finestra di campionamento dei dati.

U.S Feb-Jul 2018-2h

Posizioni lunghe

	W	Profitto potenziale medio per timestamp (%)	Deviazione standard	# di raccomandazioni	Profitto potenziale massimo (%)
Regressione lineare	1	0,684	0,921	96	4,226
	2	0,589	0,793	96	3,378
	3	0,561	0,778	96	3,378
	4	0,648	0,807	96	3,153
NN[Standard]	1	0,672	0,964	96	5,198
	2	0,892	1,078	96	5,198
	3	0,847	1,106	96	5,198
	4	0,718	0,884	96	3,378
NN[Standard, Standard]	1	0,564	0,855	96	3,216
	2	0,801	1,098	96	5,198
	3	0,667	0,820	96	3,216
	4	0,542	0,835	96	3,216
Random Forest	1	0,752	1,084	96	5,198
	2	0,687	1,074	96	5,198
	3	0,709	1,063	96	5,198
	4	0,647	1,055	96	5,198
SVM dot	1	0,577	0,904	96	4,728
	2	0,538	0,807	95	3,216
	3	0,415	0,682	96	2,870
	4	0,335	0,560	96	2,585
SVM polinomiale	1	0,360	0,826	96	2,972
	2	0,372	0,817	96	2,972
	3	0,413	0,873	96	4,075
	4	0,167	0,446	96	1,903

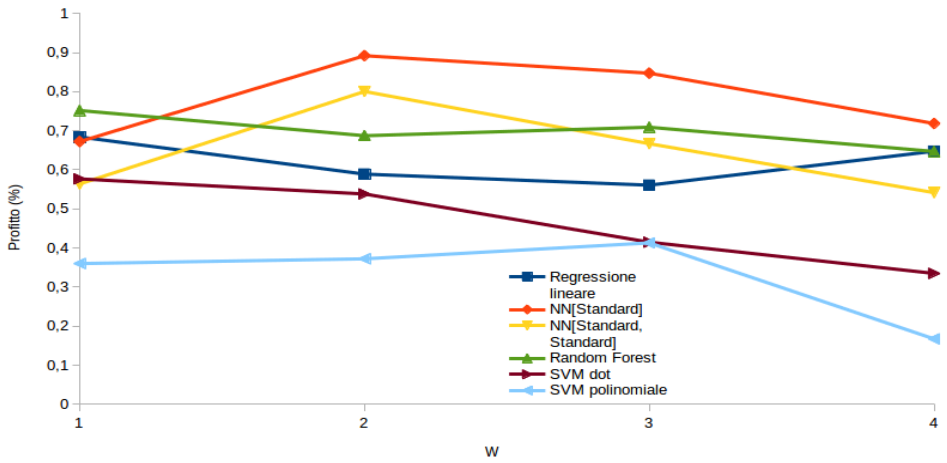

Fig. 6.5 Confronto tra i profitti medi percentuali per timestamp ottenuti dai metodi predittivi con posizioni lunghe, al variare della finestra di campionamento. Dati con granularità pari a due ore.

Tabella 10
Dati con granularità pari a due ore. Sono mostrati i profitti potenziali medi per timestamp (%), le loro deviazioni standard, il numero di raccomandazioni e il profitto potenziale massimo (%), investendo con posizioni corte e al variare della finestra di campionamento dei dati.

U.S Feb-Jul 2018-2h

Posizioni corte

	W	Profitto potenziale medio per timestamp (%)	Deviazione standard	# di raccomandazioni	Profitto potenziale massimo (%)
Regressione lineare	1	0,536	0,835	56	3,256
	2	0,420	0,836	84	3,681
	3	0,512	0,814	96	3,681
	4	0,498	0,785	96	3,681
NN[Standard]	1	0,394	0,714	92	3,827
	2	0,489	0,780	92	3,827
	3	0,468	0,768	93	3,827
	4	0,341	0,690	96	3,249
NN[Standard, Standard]	1	0,397	0,706	96	3,827
	2	0,372	0,697	96	3,827
	3	0,261	0,578	96	2,637
	4	0,382	0,676	96	3,827
Random Forest	1	0,278	0,674	96	2,899
	2	0,278	0,672	96	2,899
	3	0,275	0,623	96	2,823
	4	0,221	0,559	96	2,637
SVM dot	1	0,582	0,815	35	3,256
	2	0,492	0,716	45	3,256
	3	0,075	0,744	86	5,220
	4	0,304	0,597	96	2,979
SVM polinomiale	1	0,381	1,005	96	7,722
	2	0,247	0,696	96	3,256
	3	0,261	0,651	96	3,054
	4	0,198	0,926	96	7,722

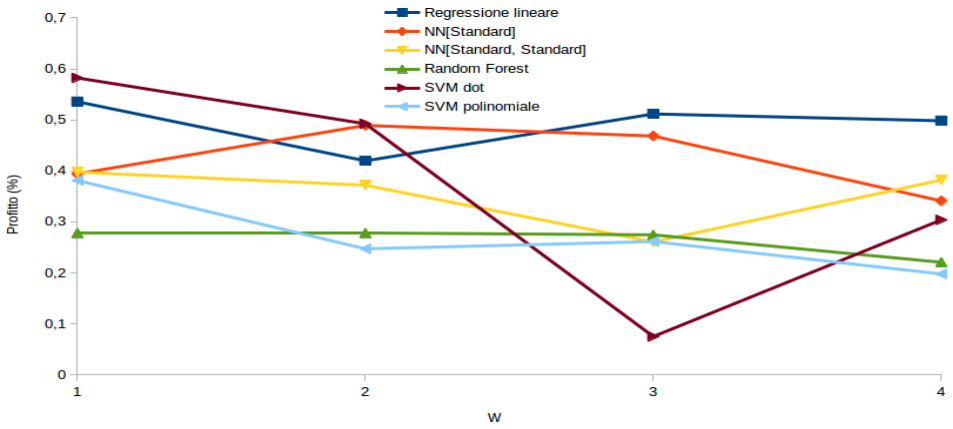

Fig. 6.6 Confronto tra i profitti medi percentuali per timestamp ottenuti dai metodi predittivi con posizioni corte, al variare della finestra di campionamento. Dati con granularità pari a due ore.

Dai risultati mostrati sopra si evince che la media dei guadagni potenziali medi per timestamp, tra tutte le strategie, cresce all'aumentare della granularità dei dati. Questo risultato è giustificabile se si osserva la dimensione degli intervalli d'intervento sul mercato per le varie granularità dei dati. È più probabile, infatti, riuscire ad intercettare variazioni di prezzo maggiori se il periodo osservato è più lungo (es. con gli slot da trenta minuti si ha una media, per le posizioni lunghe, pari a +0,171%, mentre nelle stesse condizioni ma con slot da due ore la media sale a +0,6%).
Il numero di raccomandazioni medio, tra tutte le strategie, segue l'andamento opposto. È implicito che all'aumentare della granularità diminuisca il numero di slot temporali a disposizione per il trading, il che comporta anche una riduzione del numero medio di raccomandazioni. Ciò che è interessante notare è l'aumento della frequenza delle raccomandazioni ottenuto all'aumentare della granularità dei dati.
Tenuto conto che, infatti, l'intero periodo di simulazione per gli investimenti è composto da 420, 228 e 96 slot temporali rispettivamente per le granularità dei dati pari a trenta minuti, un'ora e due ore, si hanno i risultati mostrati in **Tabella 11**.

Tabella 11
Statistiche sul numero di raccomandazioni generato in media da tutte le strategie, sia per le posizioni lunghe che per quelle corte, al variare della granularità dei dati. Risulta evidente il crescere della frequenza media all'aumentare della granularità.

Statistica	**Granularità dei dati**		
	30m	1h	2h
# di raccomandazioni medio (p. lunghe)	378,6	224,5	96
# di raccomandazioni medio (p. corte)	338	189,25	88,3
Frequenza raccomandazioni p. lunghe (%)	90,14	98,51	100
Frequenza raccomandazioni p. corte (%)	80,48	83	91,98
Frequenza media raccomandazioni (%)	85,31	90,76	95,99

I valori migliori di guadagno potenziale per timestamp si ottengono, generalmente, con valori bassi della finestra di campionamento dei dati ($W=1$ o $W=2$). In questo senso, la

regressione lineare risulta, quasi sempre, essere uno dei migliori metodi predittivi. In dettaglio:

- per la granularità pari a trenta minuti, la regressione lineare è il miglior metodo predittivo nel complesso (*W=1*, +0,223% per le posizioni lunghe, +0,45% per quelle corte), insieme alle SVMs con kernel dot (*W=2*, +0,274% per le posizioni lunghe, +0,359% per quelle corte) che però risulta penalizzata complessivamente dal numero minore di raccomandazioni generate;
- per la granularità oraria, la regressione lineare eccelle nei profitti potenziali per timestamp delle posizioni corte (*W=1*, +0,334% per le posizioni lunghe, +0,536% per quelle corte), ma le NN con le due diverse strutture risultano essere più redditizie nelle posizioni lunghe (NN[Standard] con *W=1* presenta +0,413% per le posizioni lunghe e +0,148% per quelle corte, mentre NN[Standard, Standard] con *W=1* mostra +0,413% per le posizioni lunghe e +0,148% per quelle corte). Anche le SVMs con kernel polinomiale presentano, per questo livello di aggregazione dei dati, dei buoni profitti per timestamp, ma solo per *W=1* (+0,401% per le posizioni lunghe, +0,381% per quelle corte);
- per la granularità pari a due ore, le reti neurali con un solo layer nascosto, NN[Standard], risultano essere il metodo migliore (*W=2*, +0,892% per le posizioni lunghe, +0,489% per quelle corte), anche se la regressione lineare rimane comunque uno dei migliori metodi predittivi (*W=1*, +0,684% per le posizioni lunghe, +0,536% per quelle corte).

Le tabelle da **Tabella 12** a **Tabella 14** mostrano, invece, i profitti potenziali complessivi (sommando quelli delle posizioni lunghe e corte) ottenuti dalle strategie, calcolati sul periodo della simulazione degli investimenti, per ciascuno dei tre dataset a diversa granularità.

Tabella 12
Profitti potenziali complessivi delle strategie, ottenuti sommando i profitti derivanti da posizioni lunghe con quelli delle posizioni corte. Dati con granularità a trenta minuti.

U.S Feb-Jul 2018-30m

	W	Profitti lunghe (%)	Profitti corte (%)	Profitto totale (%)
Regressione lineare	1	56,801	47,184	103,986
	2	81,708	82,563	164,272
	3	94,120	89,538	183,658
	4	94,594	73,125	167,719
NN[Standard]	1	84,945	13,114	98,059
	2	83,334	14,897	98,231
	3	63,776	20,052	83,827
	4	60,936	14,866	75,803
NN[Standard, Standard]	1	91,131	17,453	108,585
	2	74,863	21,586	96,449
	3	79,743	11,825	91,568
	4	82,999	11,566	94,565
Random Forest	1	77,642	29,822	107,464
	2	68,020	31,859	99,879
	3	78,567	27,399	105,966
	4	71,700	27,613	99,313
SVM dot	1	49,096	2,654	51,750
	2	58,640	11,493	70,133
	3	46,922	-5,817	41,105
	4	64,185	23,055	87,240
SVM polinomiale	1	22,751	-11,680	11,071
	2	15,094	-4,850	10,245
	3	9,488	-9,399	0,089
	4	-19,360	-14,217	-33,577

Tabella 13
Profitti potenziali complessivi delle strategie, ottenuti sommando i profitti derivanti da posizioni lunghe con quelli delle posizioni corte. Dati con granularità oraria.

U.S Feb-Jul 2018-1h

	W	Profitti lunghe (%)	Profitti corte (%)	Profitto totale (%)
Regressione lineare	1	72,078	41,241	113,319
	2	81,419	67,860	149,279
	3	90,290	73,497	163,786
	4	86,863	78,325	165,188
NN[Standard]	1	94,155	32,159	126,314
	2	90,623	35,437	126,060
	3	92,328	36,473	128,801
	4	81,851	36,728	118,579
NN[Standard, Standard]	1	101,234	25,716	126,950
	2	84,409	38,194	122,603
	3	94,802	32,788	127,590
	4	80,303	38,374	118,678
Random Forest	1	91,325	31,633	122,958
	2	93,477	33,800	127,276
	3	97,606	37,280	134,886
	4	93,095	33,630	126,725
SVM dot	1	49,226	23,906	73,132
	2	22,044	18,890	40,935
	3	0,039	16,043	16,081
	4	3,005	3,211	6,216
SVM polinomiale	1	77,782	20,462	98,244
	2	33,962	22,848	56,809
	3	11,942	21,634	33,576
	4	27,871	16,086	43,956

Tabella 14
Profitti potenziali complessivi delle strategie, ottenuti sommando i profitti derivanti da posizioni lunghe con quelli delle posizioni corte. Dati con granularità pari a due ore.

U.S Feb-Jul 2018-2h

	W	Profitti lunghe (%)	Profitti corte (%)	Profitto totale (%)
Regressione lineare	1	65,679	29,990	95,669
	2	56,557	35,250	91,807
	3	53,845	49,114	102,958
	4	62,184	47,833	110,018
NN[Standard]	1	64,548	36,275	100,823
	2	85,643	45,004	130,648
	3	81,345	43,552	124,897
	4	68,971	32,748	101,719
NN[Standard, Standard]	1	54,141	38,160	92,301
	2	76,850	35,694	112,544
	3	64,021	25,022	89,043
	4	52,004	36,693	88,698
Random Forest	1	72,179	26,682	98,861
	2	65,981	26,682	92,663
	3	68,075	26,356	94,430
	4	62,119	21,192	83,311
SVM dot	1	55,377	20,377	75,755
	2	51,136	22,155	73,291
	3	39,831	6,470	46,301
	4	32,157	29,183	61,339
SVM polinomiale	1	34,579	36,581	71,160
	2	35,740	23,727	59,467
	3	39,656	25,099	64,755
	4	16,067	18,983	35,051

Tenendo conto dei risultati mostrati precedentemente, è interessante descrivere come il numero di raccomandazioni che il sistema fornisce influisca sui guadagni potenziali totali.

Il numero di raccomandazioni prodotte, infatti, varia (anche sensibilmente) in base alla strategia adoperata. Si mostra, dunque, il profitto potenziale complessivo ottenibile dai migliori metodi predittivi, per ciascuna granularità:

- per quanto riguarda i dati aggregati per trenta minuti, il numero di raccomandazioni basso penalizza le SVMs con kernel dot, che con *W=2* raggiungono il profitto potenziale complessivo di +70,13%. La regressione lineare,

con *W=1*, ottiene +104%, mentre con *W=2* ottiene +164,3%. La differenza di profitto complessivo tra la regressione lineare e le SVMs dot è significativa;

- per i dati a granularità oraria, la regressione lineare presenta un profitto complessivo minore rispetto alle reti neurali (con entrambe le configurazioni). Per la regressione lineare, con *W=1*, il profitto complessivo vale +113,32%, mentre le reti neurali, entrambe le configurazioni di esse con *W=1*, ottengono profitti complessivi, quasi equivalenti, pari a +126,31% e +126,95% rispettivamente per NN[Standard] e NN[Standard, Standard]. La differenza, significativa, è di circa 13 punti percentuali.

  Anche questo risultato dipende dal numero di raccomandazioni che il sistema genera, differente per i metodi predittivi considerati. La somma delle quantità di raccomandazioni generate (tra posizioni lunghe e corte) vale 293 per la regressione lineare, contro le 445 e le 454 delle due configurazioni diverse di NN. La configurazione con i modelli SVMs aventi kernel polinomiale e *W=1* ottiene un profitto complessivo di +98,24% (il minore tra i quattro), nuovamente a causa del basso numero di raccomandazioni generate (225);
- per quanto riguarda i dati con granularità pari a due ore, il numero di raccomandazioni generate è, generalmente, equivalente tra tutti i metodi predittivi considerati. Fa eccezione la regressione lineare che, per *W=1* e per le posizioni corte, genera soltanto 56 raccomandazioni sui 96 slot temporali. Questo penalizza tale modello, in termini di profitto potenziale complessivo, nei confronti delle configurazioni con reti neurali (con *W=2)*. La regressione lineare (per *W*=1) ottiene un profitto potenziale complessivo pari a +95,67%, contro i +130,65% e +112,54% delle reti neurali rispettivamente con uno e due layer nascosti, aventi *W=2*.

Si vogliono adesso valutare le strategie proposte in termini di affidabilità delle raccomandazioni. In particolare, per ogni strategia, si vuole valutare in che percentuale le raccomandazioni generate possano portare effettivamente ad un guadagno potenzialmente ottenibile.

Le tabelle da **Tabella 15** a **Tabella 17** mostrano il rapporto tra il numero di guadagni potenziali ottenuti e il numero di raccomandazioni totali generate, quest'ultimo dato dalla somma delle raccomandazioni per le posizioni lunghe e di quelle per le posizioni corte.

Tabella 15
Rapporto tra il numero di guadagni potenziali e il numero di raccomandazioni prodotte da ogni configurazione. Dati con granularità pari a trenta minuti.

U.S Feb-Jul 2018-30m

	W	Frequenza del guadagno potenziale (%)
Regressione lineare	1	70,7521
	2	69,402
	3	69,859
	4	66,204
NN[Standard]	1	57,946
	2	57,373
	3	56,298
	4	54,705
NN[Standard, Standard]	1	59,777
	2	59,804
	3	58,363
	4	58,434
Random Forest	1	58,844
	2	58,095
	3	58,691
	4	60,238
SVM dot	1	53,452
	2	68,293
	3	54,546
	4	65,868
SVM polinomiale	1	51,361
	2	49,881
	3	48,333
	4	42,262
	Media:	58,699

Tabella 16
Rapporto tra il numero di guadagni potenziali e il numero di raccomandazioni prodotte da ogni configurazione. Dati con granularità oraria.

U.S Feb-Jul 2018-1h

	W	Frequenza del guadagno potenziale (%)
Regressione lineare	1	73,720
	2	73,127
	3	75,520
	4	73,874
NN[Standard]	1	68,764
	2	66,966
	3	65,760
	4	63,218
NN[Standard, Standard]	1	69,163
	2	67,991
	3	67,544
	4	68,750
Random Forest	1	67,763
	2	67,982
	3	68,202
	4	66,964
SVM dot	1	66,912
	2	61,247
	3	51,754
	4	54,464
SVM polinomiale	1	76
	2	63,395
	3	55,504
	4	63,725
	Media:	66,596

Tabella 17
Rapporto tra il numero di guadagni potenziali e il numero di raccomandazioni prodotte da ogni configurazione. Dati con granularità pari a due ore.

U.S Feb-Jul 2018-2h

	W	Frequenza del guadagno potenziale (%)
Regressione lineare	1	82,237
	2	81,667
	3	81,250
	4	82,292
NN[Standard]	1	79,787
	2	82,979
	3	80,952
	4	78,646
NN[Standard, Standard]	1	77,604
	2	77,604
	3	76,042
	4	76,563
Random Forest	1	69,792
	2	67,188
	3	70,313
	4	68,229
SVM dot	1	80,153
	2	79,286
	3	65,934
	4	78,646
SVM polinomiale	1	71,875
	2	68,229
	3	71,354
	4	67,188
	Media:	75,659

Risulta evidente che la media della frequenza del guadagno potenziale aumenta all'aumentare della granularità dei dati. L'incremento è significativo e pari a circa il 9%, per ciascun incremento della granularità dei dati.
In media, dunque, le strategie con dati a più alta granularità incorrono in un numero minore di raccomandazioni errate.

6.4 Valutazione della precisione delle strategie

Le tabelle da **Tabella 19** a **Tabella 21** mostrano l'errore di predizione, mediato tra tutti i modelli relativi alle singole azioni, commesso dalle strategie in esame e misurato con la metrica MAPE (descritta in 4.5). Le tabelle sono suddivise per granularità dei dati, e

mostrano il MAPE medio, la sua deviazione standard e il valore massimo di errore di predizione commesso da un singolo modello di un'azione, relativamente alla strategia considerata.

Tabella 18
Configurazione dei metodi predittivi. I parametri scelti sono quelli ottenuti dal processo di calibrazione dei metodi.

Parametri scelti dei metodi predittivi

	Granularità	Configurazione
Regressione lineare	30m/1h/2h	Feature selection = M5 prime min_tolerance = 0,05 ridge = 1,0E-8
NN[Standard]	30m/1h/2h	weight decay = on 1 layer nascosto
NN[Standard, Standard]	30m/1h/2h	weight decay = off 2 layer nascosti
Random Forest	30m	number_of_trees = 14 minimal_leaf_size = 6 maximal_depth = 21 minimal_size_for_split = 42
	1h	number_of_trees = 248 minimal_leaf_size = 8 maximal_depth = 29 minimal_size_for_split = 18
	2h	number_of_trees = 149 minimal_leaf_size = 16 maximal_depth = 56 minimal_size_for_split = 21
SVM dot	30m	C = 20,626 epsilon = 0,092
	1h	C = 21,467 epsilon = 0,291
	2h	C = 19,71 epsilon = 0,101
SVM polinomiale	30m	C = 12,216 epsilon = 0,513 kernel_degree = 0,68
	1h	C = 23,087 epsilon = 0,154 kernel_degree = 0,554
	2h	C = 15,745 epsilon = 0,599 kernel_degree = 0,323

Tabella 19
Dati con granularità pari a trenta minuti. Sono mostrati il valore del MAPE medio, la sua deviazione standard, ed il valore di errore massimo di ogni strategia, al variare della finestra di campionamento dei dati.

U.S Feb-Jul 2018-30m

	W	MAPE medio	Deviazione standard	MAPE Massimo
Regressione lineare	1	0,265	0,071	0,609
	2	0,327	0,113	1,182
	3	0,425	0,151	1,671
	4	0,506	0,188	2,06
NN[Standard]	1	0,389	0,24	1,974
	2	0,37	0,224	1,999
	3	0,367	0,214	1,746
	4	0,372	0,225	1,929
NN[Standard, Standard]	1	0,47	0,337	2,84
	2	0,479	0,33	3,045
	3	0,486	0,332	3,036
	4	0,484	0,35	3,235
Random Forest	1	0,948	0,718	7,748
	2	0,966	0,695	6,394
	3	1,122	0,818	8,47
	4	0,911	0,621	3,775
SVM dot	1	0,268	0,549	9,306
	2	0,226	0,063	0,499
	3	0,326	1,622	35,38
	4	0,287	0,982	22,066
SVM polinomiale	1	0,412	0,347	3,299
	2	0,402	0,325	3,212
	3	0,527	1,539	28,346
	4	0,841	3,875	71,113

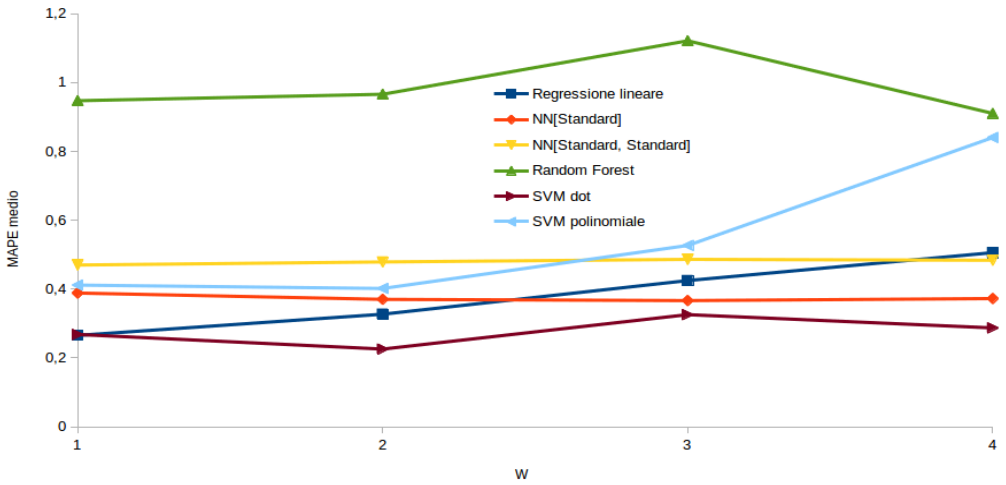

Fig. 6.7 Confronto tra gli errori di predizione medi (misurati con la metrica MAPE) dei diversi metodi predittivi, al variare della finestra di campionamento dei dati. Dati con granularità pari a trenta minuti.

Tabella 20
Dati con granularità oraria. Sono mostrati il valore del MAPE medio, la sua deviazione standard, ed il valore di errore massimo di ogni strategia, al variare della finestra di campionamento dei dati.

U.S Feb-Jul 2018-1h

	W	MAPE medio	Deviazione standard	MAPE Massimo
Regressione lineare	1	0,364	0,102	0,786
	2	0,472	0,167	1,744
	3	0,575	0,235	2,51
	4	1,263	0,292	2,816
NN[Standard]	1	0,494	0,268	2,205
	2	0,497	0,276	2,44
	3	0,475	0,257	2,427
	4	0,495	0,281	2,618
NN[Standard, Standard]	1	0,683	0,412	3,669
	2	0,631	0,398	3,552
	3	0,677	0,404	3,359
	4	0,647	0,451	6,523
Random Forest	1	1,135	0,682	5,633
	2	0,992	0,672	5,38
	3	1,125	0,747	6,735
	4	0,948	0,625	4,56
SVM dot	1	0,366	0,169	1,792
	2	0,411	0,913	19,511
	3	1,551	10,31	203,342
	4	2,16	12,906	238,049
SVM polinomiale	1	0,323	0,097	0,841
	2	0,634	3,901	79,265
	3	1,674	13,844	271,125
	4	1,226	9,7	165,129

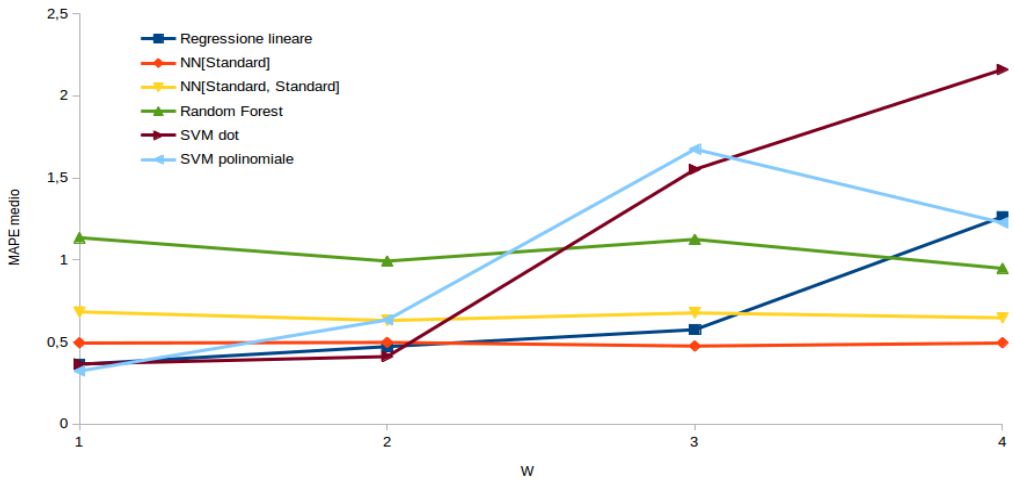

Fig. 6.8 Confronto tra gli errori di predizione medi (misurati con la metrica MAPE) dei diversi metodi predittivi, al variare della finestra di campionamento dei dati. Dati con granularità oraria.

Tabella 21
Dati con granularità pari a due ore. Sono mostrati il valore del MAPE medio, la sua deviazione standard, ed il valore di errore massimo di ogni strategia, al variare della finestra di campionamento dei dati.

U.S Feb-Jul 2018-2h

	W	MAPE medio	Deviazione standard	MAPE Massimo
Regressione lineare	1	0,485	0,163	1,374
	2	0,59	0,261	2,758
	3	0,665	0,326	2,37
	4	0,684	0,35	2,695
NN[Standard]	1	0,71	0,331	3,233
	2	0,721	0,344	3,695
	3	0,697	0,334	3,814
	4	0,722	0,347	3,331
NN[Standard, Standard]	1	0,947	0,51	6,275
	2	0,926	0,538	7,112
	3	0,933	0,568	7,059
	4	0,947	0,507	4,955
Random Forest	1	1,52	1,069	10,563
	2	1,532	1,053	9,871
	3	1,619	1,072	9,983
	4	1,521	0,99	8,925
SVM dot	1	0,514	0,144	1,171
	2	0,51	0,148	1,152
	3	0,766	2,618	44,373
	4	2,631	12,312	155,176
SVM polinomiale	1	0,656	0,388	3,943
	2	0,652	0,379	3,928
	3	0,655	0,38	4,034
	4	0,916	1,954	31,233

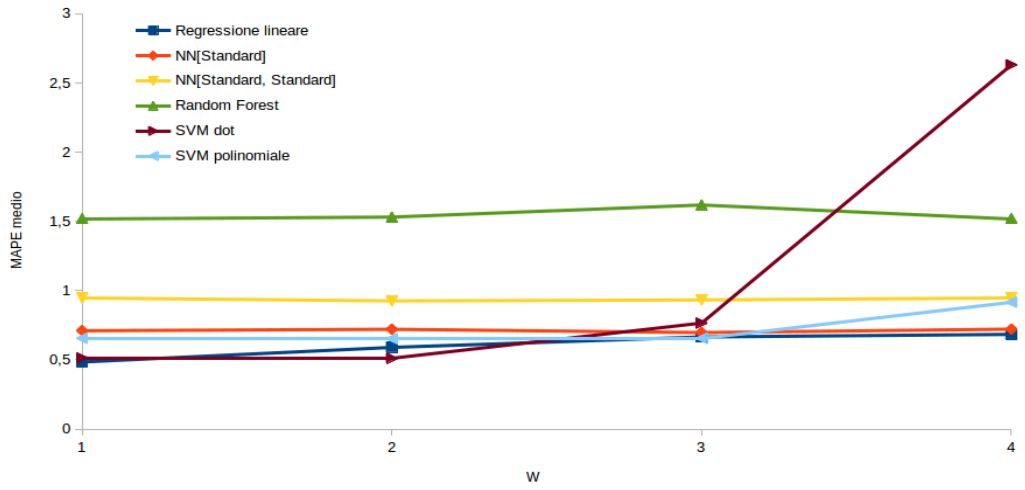

Fig. 6.9 Confronto tra gli errori di predizione medi (misurati con la metrica MAPE) dei diversi metodi predittivi, al variare della finestra di campionamento dei dati. Dati con granularità pari a due ore.

Dai risultati mostrati, si evince che gli errori di predizione minori delle strategie si ottengono, generalmente, per valori piccoli di dimensione della finestra di campionamento dei dati ($W=1$ o $W=2$). Questo andamento dell'errore di predizione, in funzione della dimensione della finestra, è evidente per metodi predittivi quali la regressione lineare e le SVMs con entrambi i tipi di kernel considerati, mentre il Random Forest e le reti neurali presentano un andamento dell'errore qualitativamente costante.
All'aumentare della dimensione della finestra di campionamento dei dati, dunque, la qualità della predizione è generalmente minore, a causa del fatto che la correlazione tra i valori passati delle azioni e quelli futuri diventa più debole.

I metodi predittivi migliori, valutati secondo la media del MAPE dei modelli di una stessa strategia relativamente a tutti gli stock, risultano essere la regressione lineare e le SVMs con kernel dot, che ottengono i più bassi errori di predizione con le dimensioni piccole della finestra di campionamento. In esempio, con granularità dei dati pari a trenta minuti, la regressione lineare mostra errori di predizione 0,265% e 0,327% per le dimensioni della finestra pari a uno e due, rispettivamente; le SVMs con kernel dot, per

le stesse dimensioni della finestra e per i dati con la stessa granularità, presentano 0,268% e 0,226%.

Si noti anche che la regressione lineare presenta le deviazioni standard del MAPE medio più basse rispetto a tutti gli altri metodi predittivi, a parità di granularità dei dati e di dimensione della finestra. Fanno eccezione le SVMs con kernel dot che per la granularità pari a trenta minuti e per W=2 ottengono il miglior risultato in assoluto per quanto riguarda l'errore di predizione.

La rete neurale con un solo layer nascosto e con la regolarizzazione data dal weight decay presenta, per tutte le granularità dei dati e per tutti i valori di dimensione della finestra, un errore di predizione minore rispetto alla configurazione di rete avente due layer nascosti e priva di tecniche di regolarizzazione.

Il Random Forest risulta essere, quasi sempre, il metodo predittivo meno preciso, ottenendo valori di MAPE medio di circa 1% per le granularità pari a trenta minuti e ad un'ora, e di circa 1,5% per la granularità pari a due ore.

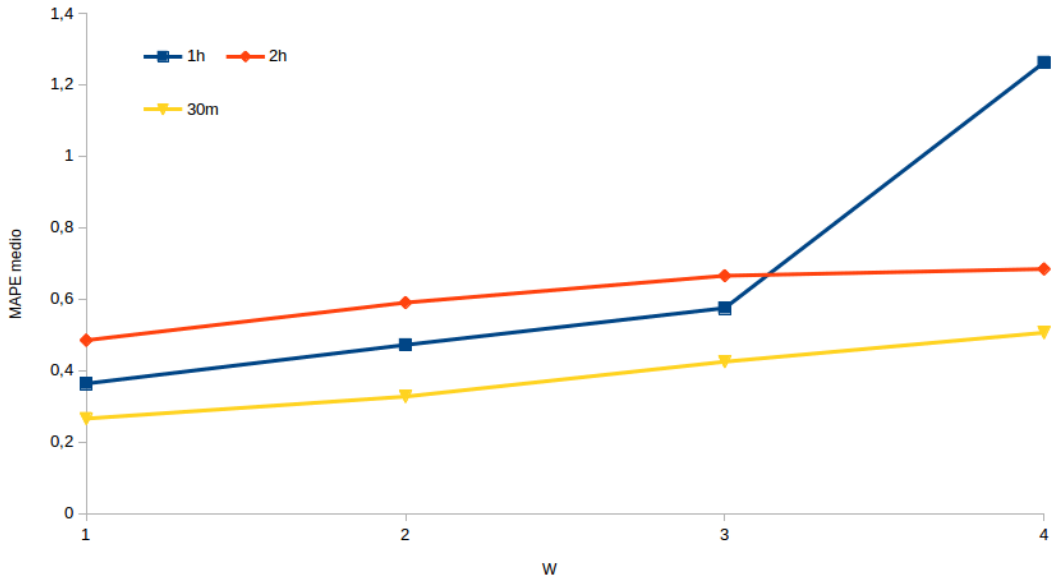

Fig. 6.10 Andamento del valore di MAPE medio per la regressione lineare, al variare della dimensione della finestra di campionamento dei dati. Sono mostrati i valori per le tre granularità considerate.

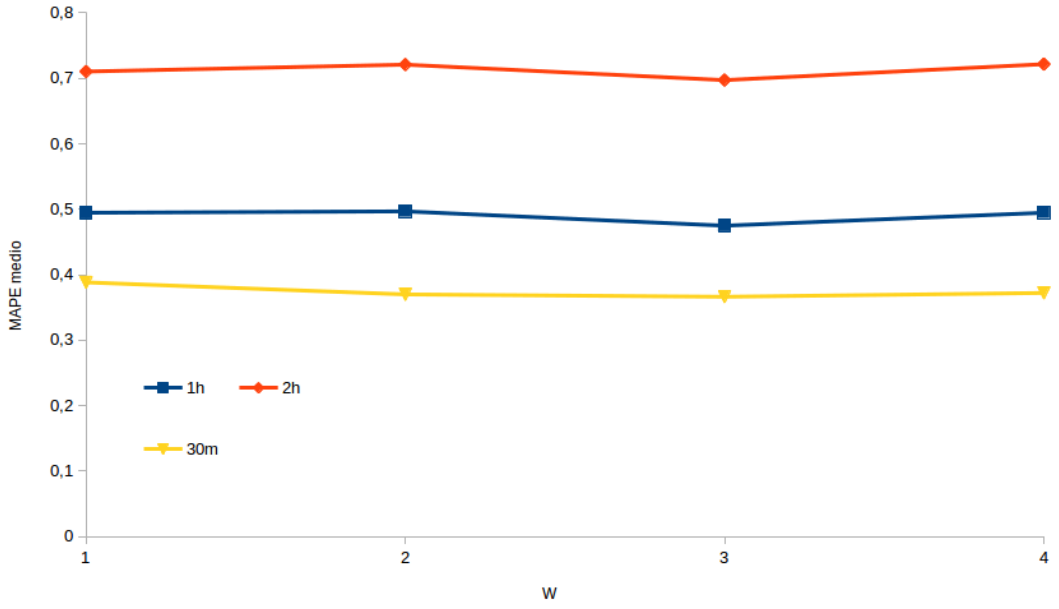

Fig. 6.11 Andamento del valore di MAPE medio per NN[Standard], al variare della dimensione della finestra di campionamento dei dati. Sono mostrati i valori per le tre granularità considerate.

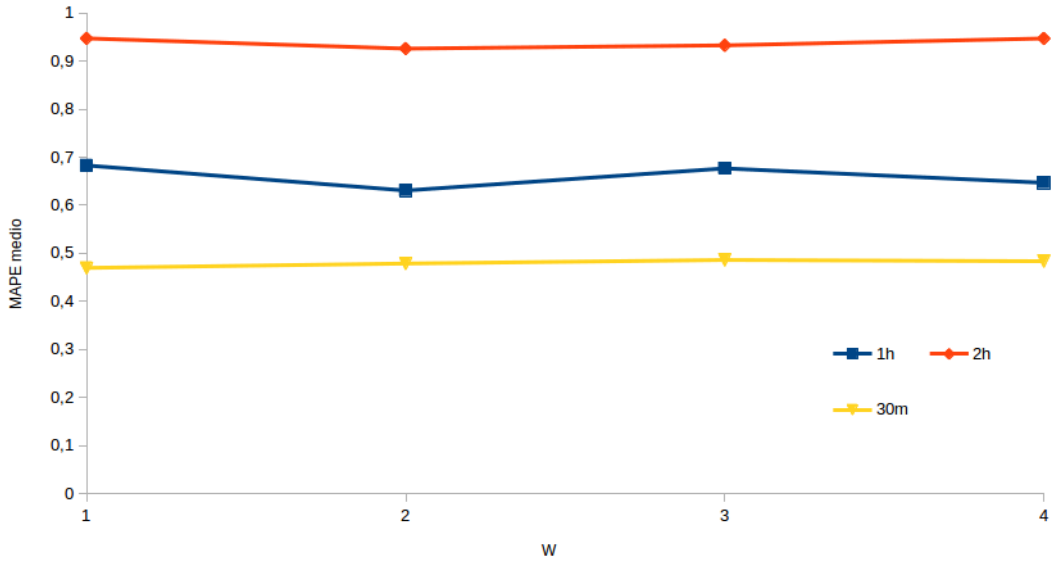

Fig. 6.12 Andamento del valore di MAPE medio per NN[Standard, Standard], al variare della dimensione della finestra di campionamento dei dati. Sono mostrati i valori per le tre granularità considerate.

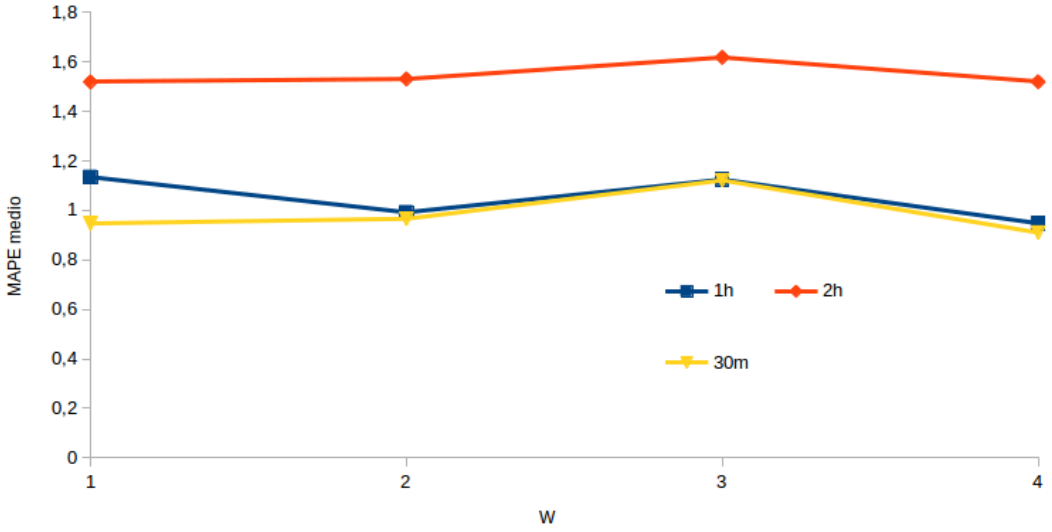

Fig. 6.13 Andamento del valore di MAPE medio per il Random Forest, al variare della dimensione della finestra di campionamento dei dati. Sono mostrati i valori per le tre granularità considerate.

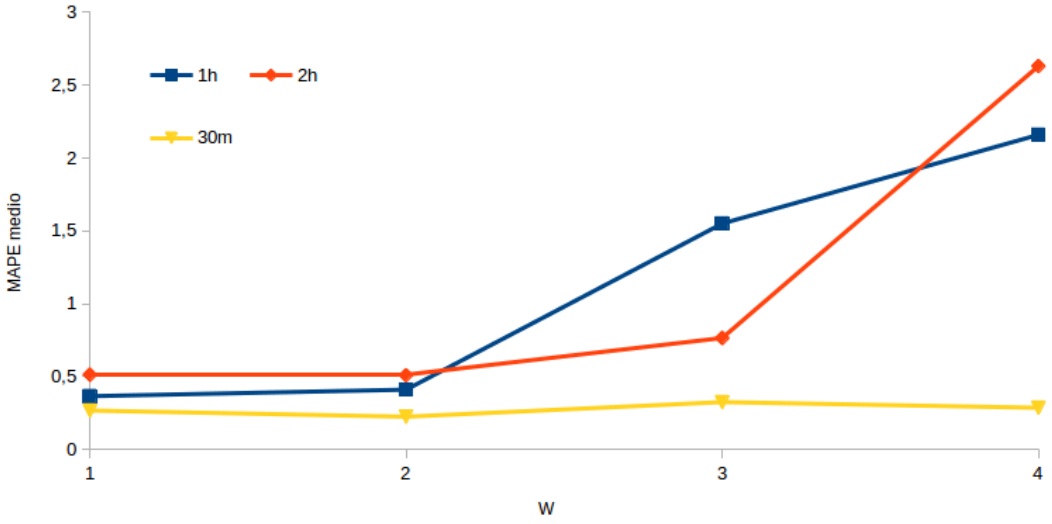

Fig. 6.14 Andamento del valore di MAPE medio per le SVMs con kernel dot, al variare della dimensione della finestra di campionamento dei dati. Sono mostrati i valori per le tre granularità considerate.

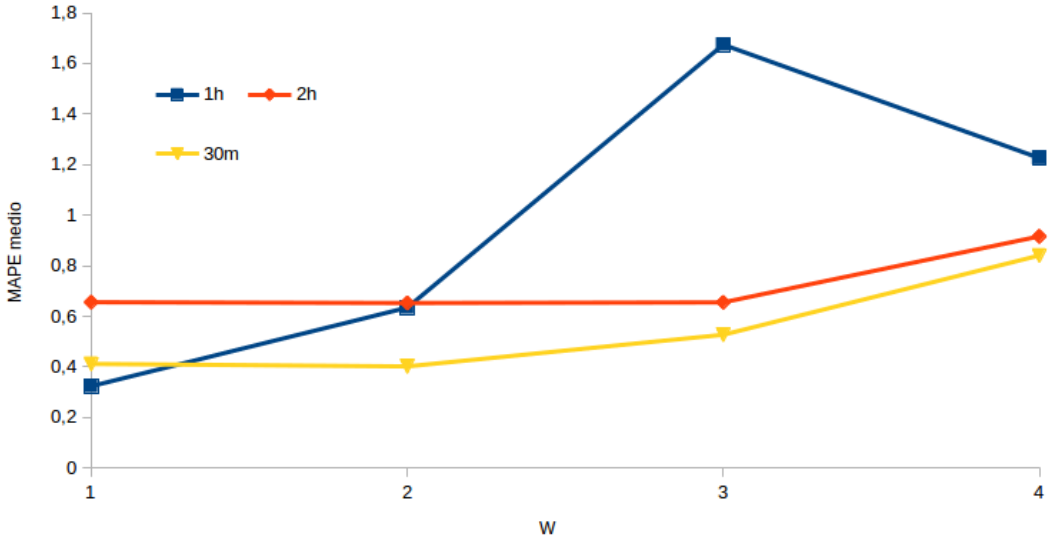

Fig. 6.15 Andamento del valore di MAPE medio per le SMVs con kernel polinomiale, al variare della dimensione della finestra di campionamento dei dati. Sono mostrati i valori per le tre granularità considerate.

I grafici da **Fig. 6.10** a **Fig. 6.15** mostrano l'andamento del valore di MAPE medio per ciascun metodo predittivo e per le tre granularità dei dati considerate, al variare della dimensione della finestra di campionamento. È possibile osservare che, generalmente e per valori piccoli (es. 1-2) di dimensione della finestra, quanto maggiore è la frequenza di campionamento dei dati tanto minore è l'errore di predizione misurato. L'unica eccezione è data dalle SVMs con kernel polinomiale (**Fig. 6.15)** che, per *W=1*, ottengono un errore di predizione medio più basso per i dati con granularità oraria (0,323%) rispetto ai dati con granularità pari a trenta minuti (0,412%).

Considerando i due migliori metodi predittivi, la regressione lineare e le SVMs con kernel dot, si vogliono mostrare nei grafici da **Fig. 6.16** a **Fig.6.39** i confronti tra le predizioni generate e la grandezza predetta (prezzo massimo delle azioni), per ciascuna granularità dei dati e per i valori di dimensione della finestra pari a uno e due; si mostrano i casi riguardanti le azioni per le quali le strategie in esame hanno commesso il più basso e il più alto errore di predizione.

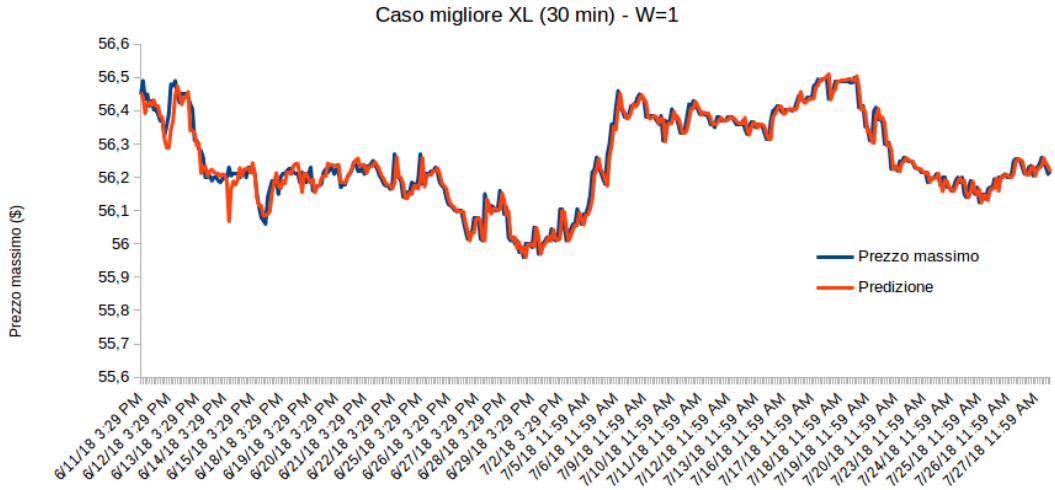

Fig. 6.16 Il più basso errore di predizione della regressione lineare, per W=1 e per i dati con granularità pari a trenta minuti, si ottiene per l'azione XL.
MAPE = 0,035%.

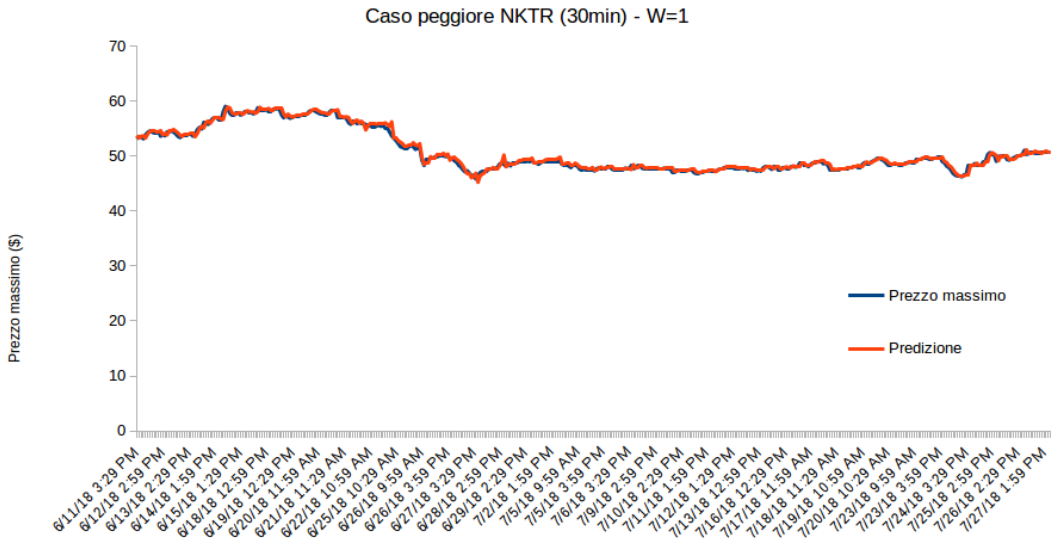

Fig. 6.17 Il più alto errore di predizione della regressione lineare, per W=1 e per i dati con granularità pari a trenta minuti, si ottiene per l'azione NKTR.
MAPE = 0,609%.

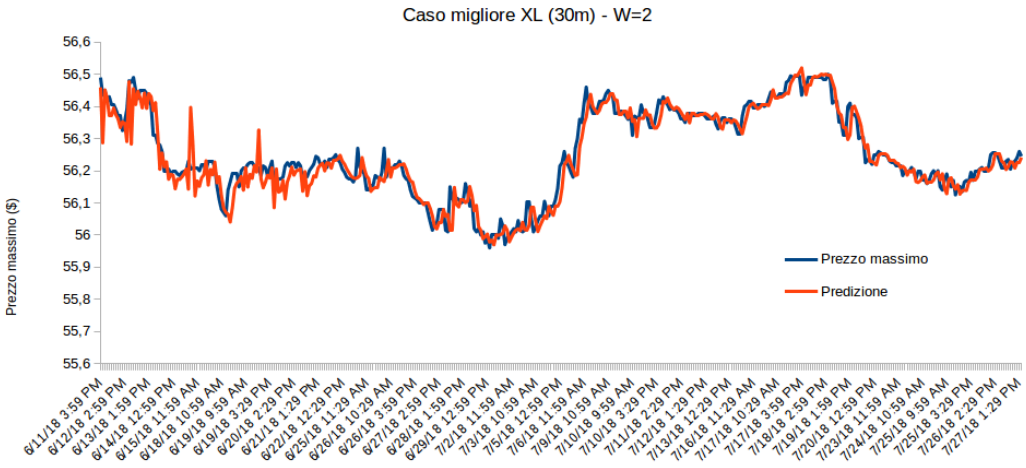

Fig. 6.18 Il più basso errore di predizione della regressione lineare, per W=2 e per i dati con granularità pari a trenta minuti, si ottiene per l'azione XL.
MAPE = 0,049%.

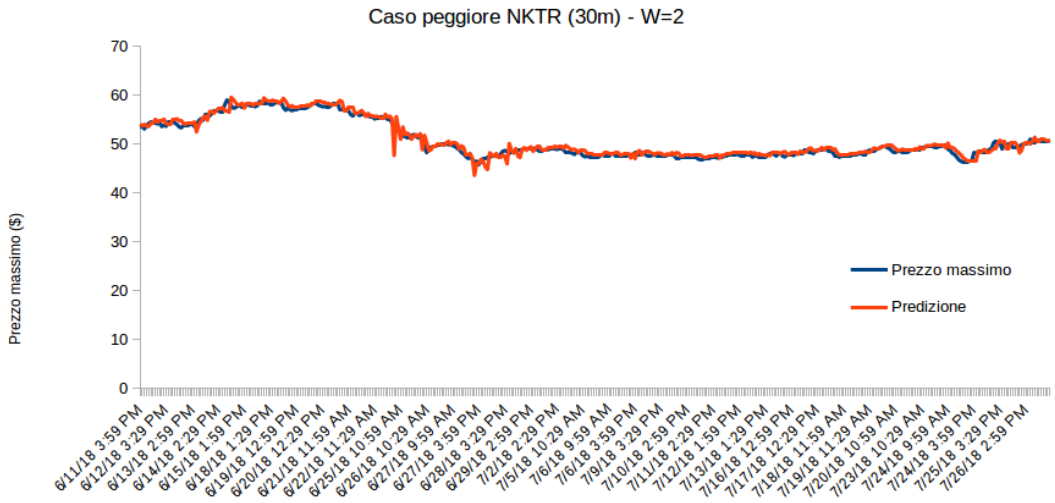

Fig. 6.19 Il più alto errore di predizione della regressione lineare, per W=2 e per i dati con granularità pari a trenta minuti, si ottiene per l'azione NKTR.
MAPE = 1,182%.

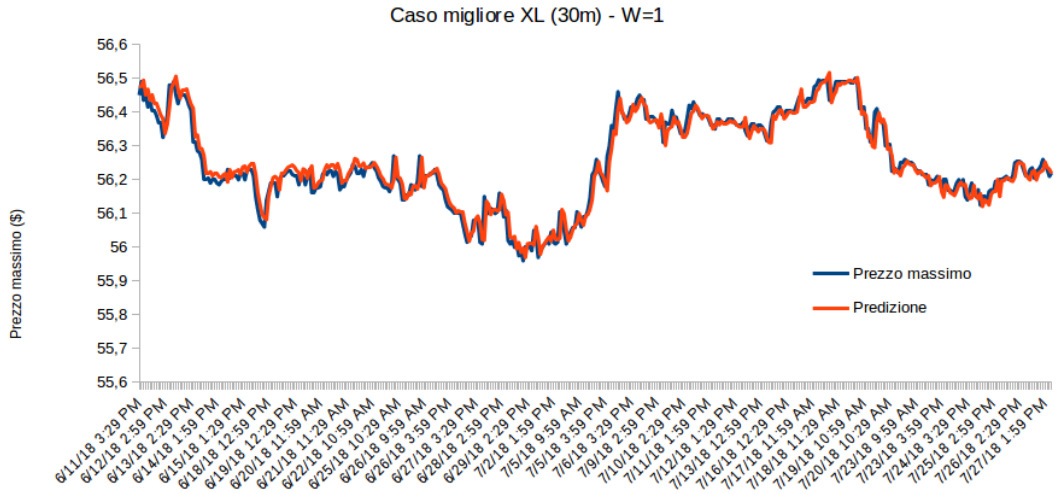

Fig. 6.20 Il più basso errore di predizione delle SVMs con kernel dot, per W=1 e per i dati con granularità pari a trenta minuti, si ottiene per l'azione XL.
MAPE = 0,037%.

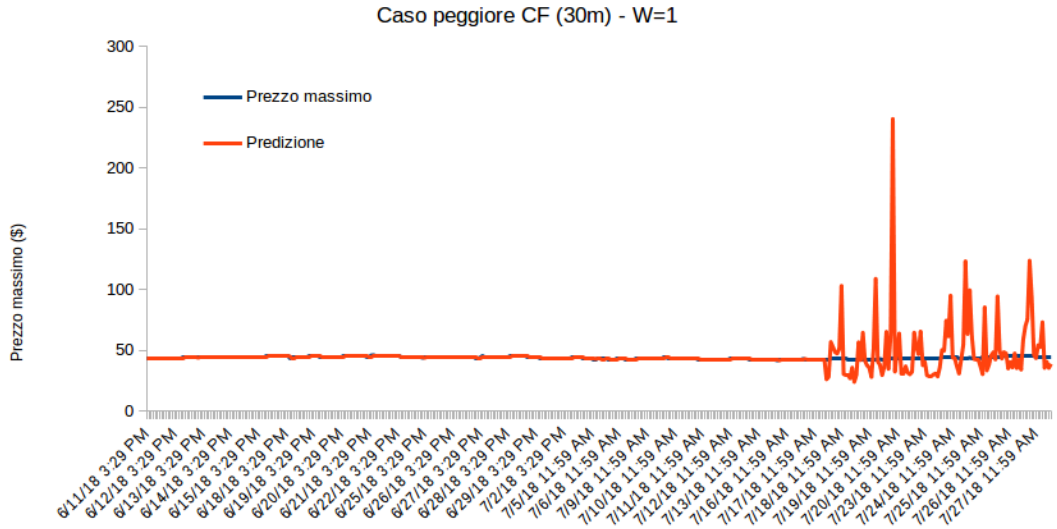

Fig. 6.21 Il più alto errore di predizione delle SVMs con kernel dot, per W=1 e per i dati con granularità pari a trenta minuti, si ottiene per l'azione CF.
MAPE = 9,306%. Si notino le previsioni completamente errate relativamente al quarto periodo di predizioni, le quali causano l'elevato valore di MAPE.

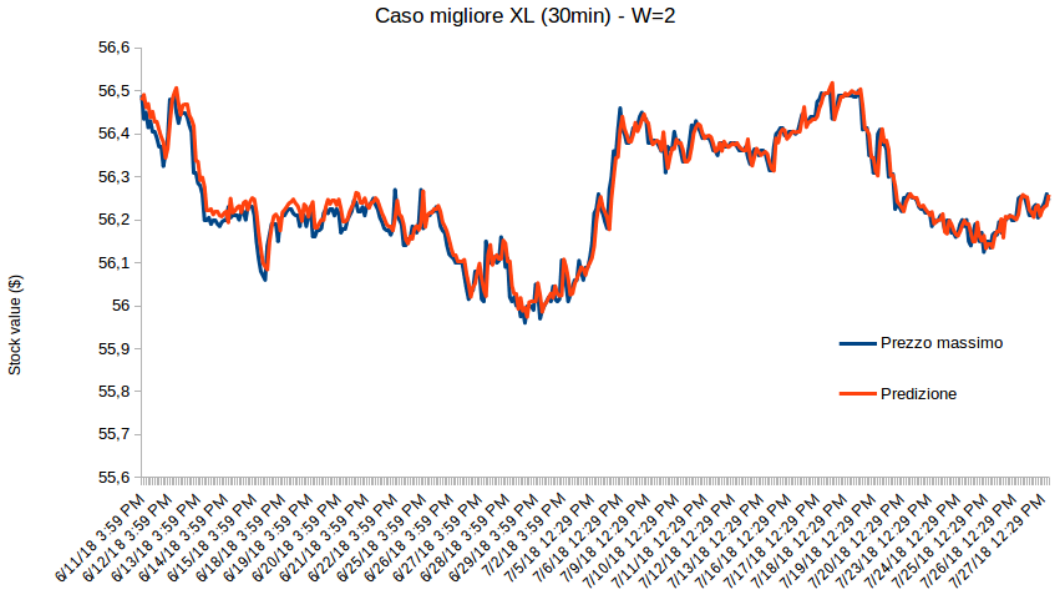

Fig. 6.22 Il più basso errore di predizione delle SVMs con kernel dot, per W=2 e per i dati con granularità pari a trenta minuti, si ottiene per l'azione XL.
MAPE = 0,037%.

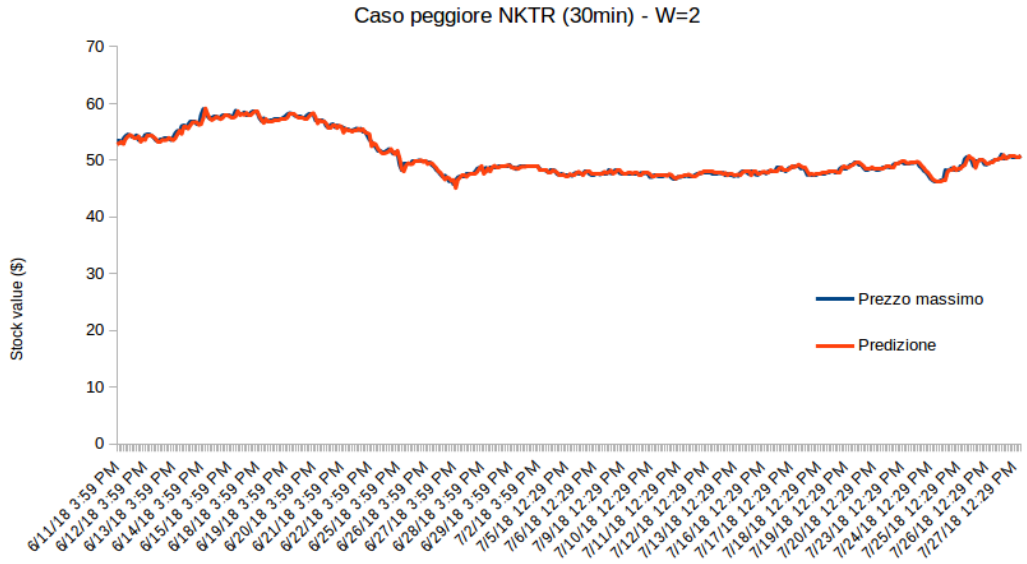

Fig. 6.23 Il più alto errore di predizione delle SVMs con kernel dot, per W=2 e per i dati con granularità pari a trenta minuti, si ottiene per l'azione NKTR.
MAPE = 0,499%. Si noti che la precisione delle predizioni è significativamente maggiore rispetto al caso peggiore della stessa configurazione, ma con W=1.

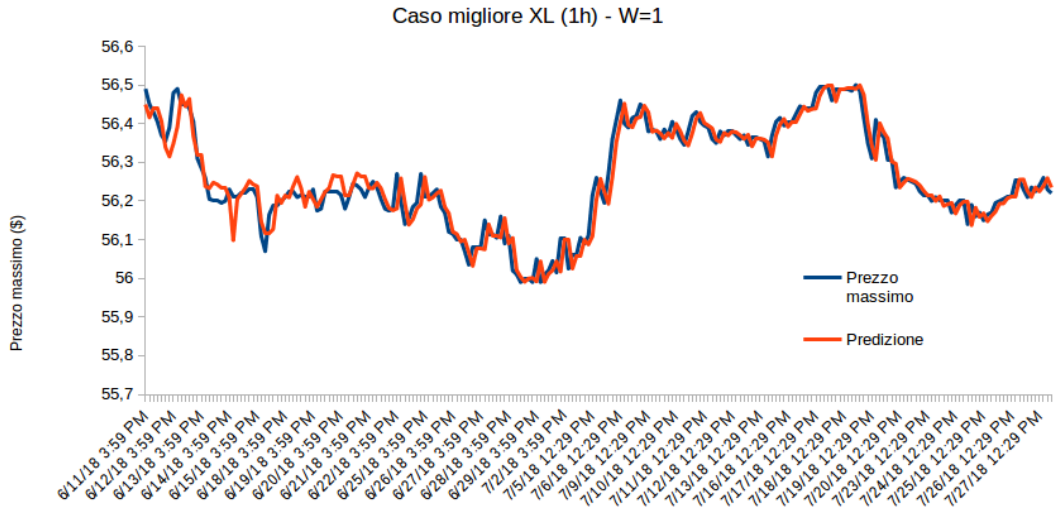

Fig. 6.24: Il più basso errore di predizione della regressione lineare, per W=1 e per i dati con granularità oraria, si ottiene per l'azione XL.MAPE = 0,046%.

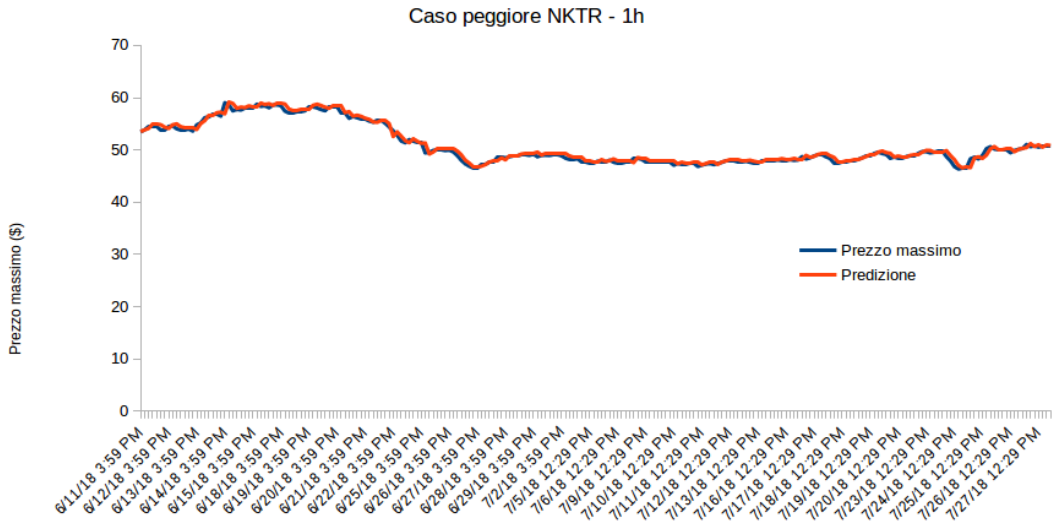

Fig. 6.25 Il più alto errore di predizione della regressione lineare, per W=1 e per i dati con granularità oraria, si ottiene per l'azione NKTR.
MAPE = 0,786%.

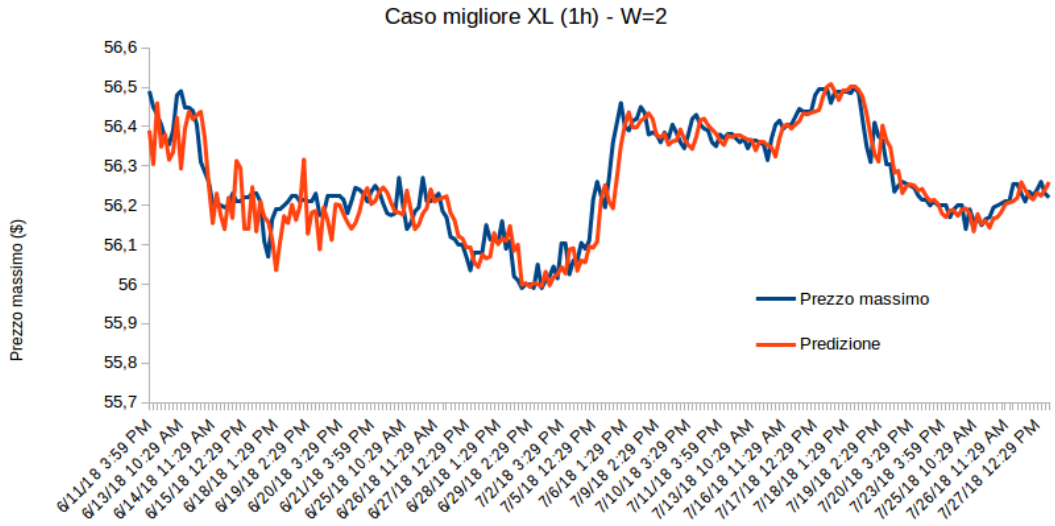

Fig. 6.26 Il più basso errore di predizione della regressione lineare, per W=2 e per i dati con granularità oraria, si ottiene per l'azione XL.

MAPE = 0,067%.

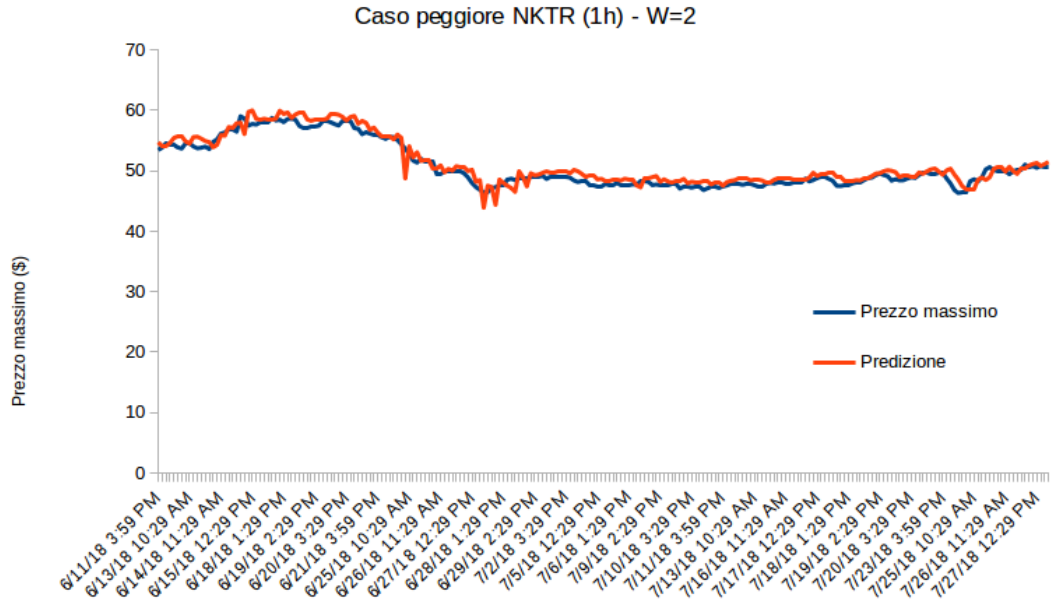

Fig. 6.27 Il più alto errore di predizione della regressione lineare, per W=2 e per i dati con granularità oraria, si ottiene per l'azione NKTR.

MAPE = 1,744%.

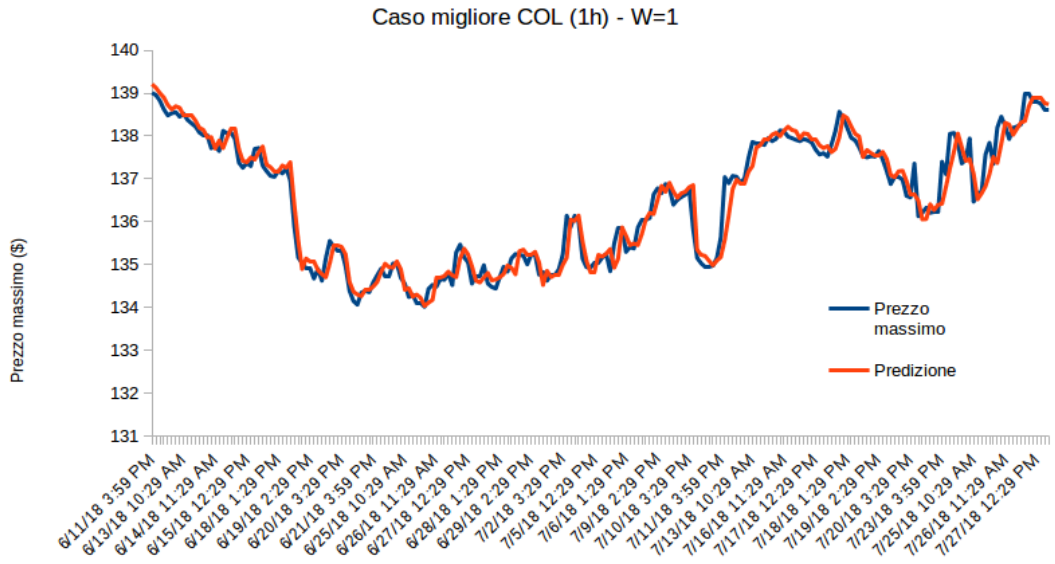

Fig. 6.28 Il più basso errore di predizione delle SVMs con kernel dot, per W=1 e per i dati con granularità oraria, si ottiene per l'azione COL.
MAPE = 0,164%.

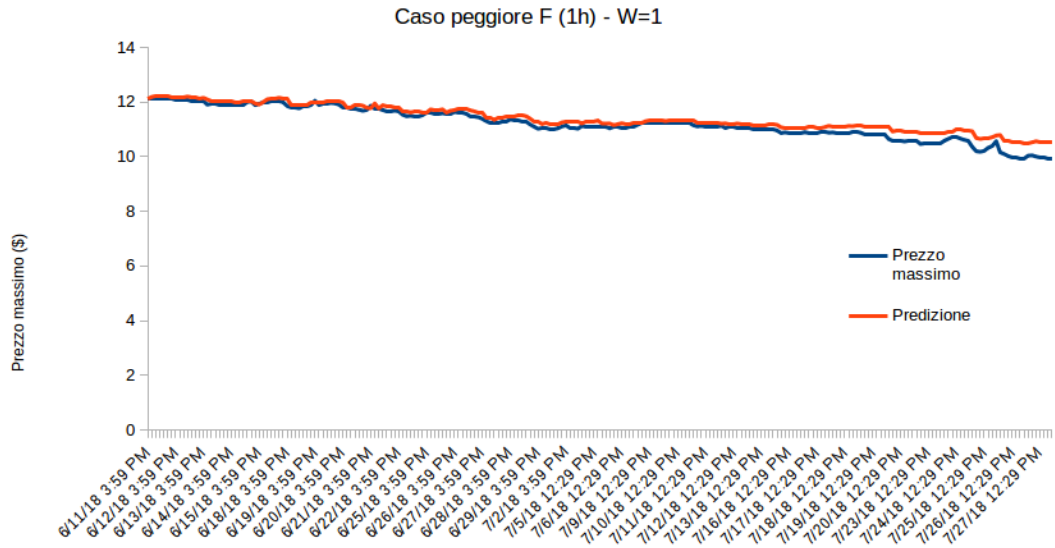

Fig. 6.29 Il più alto errore di predizione delle SVMs con kernel dot, per W=1 e per i dati con granularità oraria, si ottiene per l'azione F.
MAPE = 1,792%.

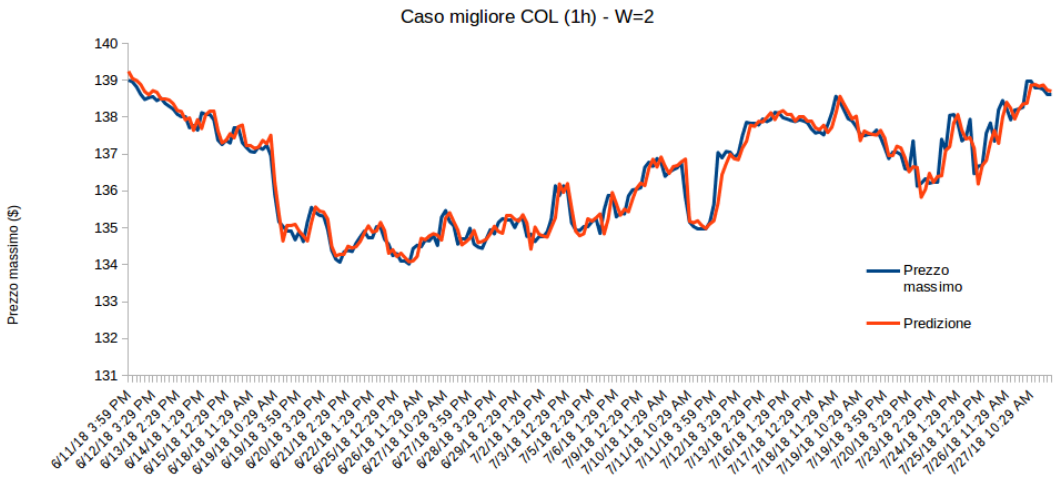

Fig. 6.30 Il più basso errore di predizione delle SVMs con kernel dot, per W=2 e per i dati con granularità oraria, si ottiene per l'azione COL.
MAPE = 0,16%.

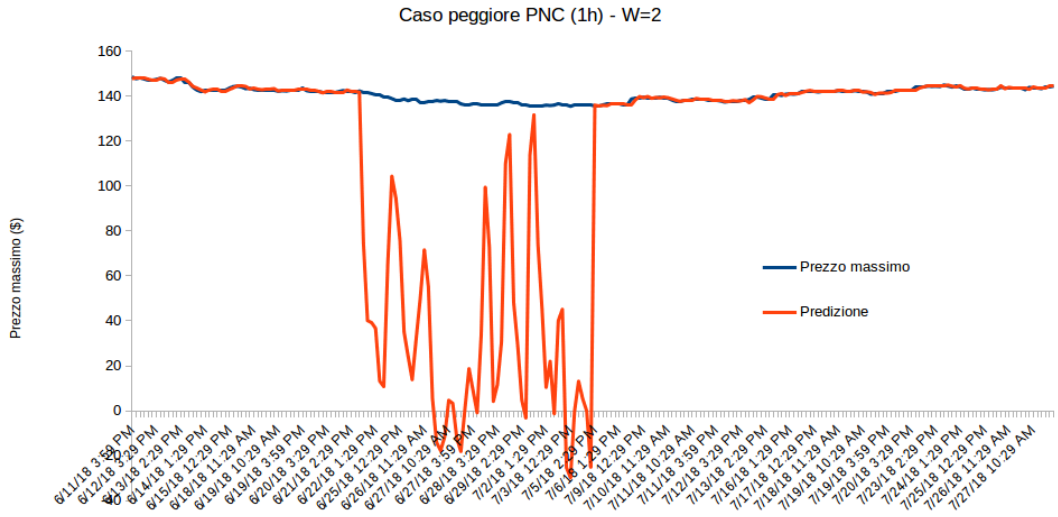

Fig. 6.31 Il più alto errore di predizione delle SVMs con kernel dot, per W=2 e per i dati con granularità oraria, si ottiene per l'azione PNC.
MAPE = 19,511%. Si notino le previsioni completamente errate relativamente al secondo periodo di predizioni, le quali causano l'elevato valore di MAPE.

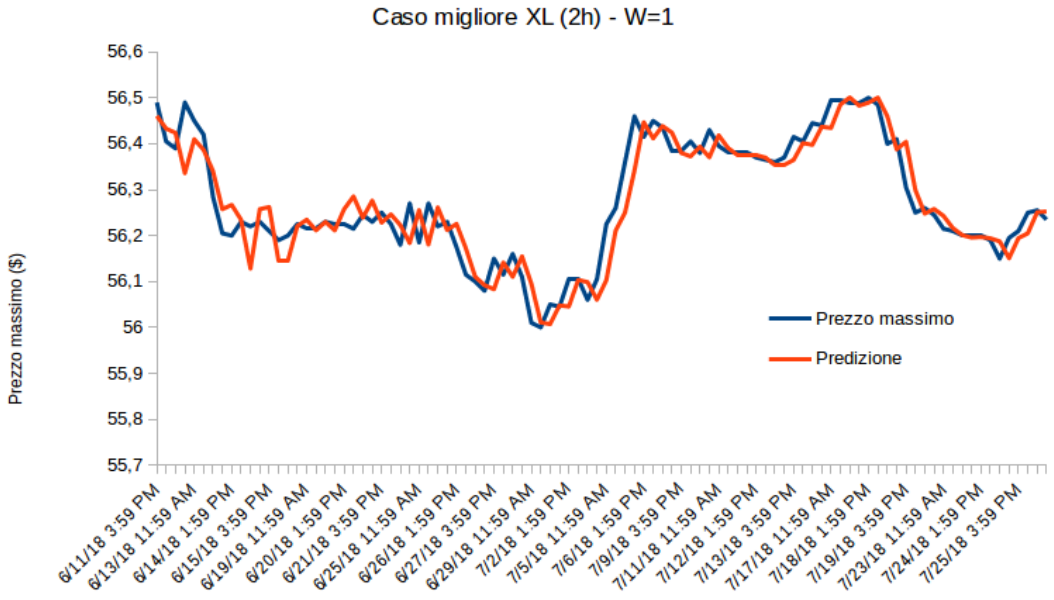

Fig. 6.32 Il più basso errore di predizione della regressione lineare, per W=1 e per i dati con granularità pari a due ore, si ottiene per l'azione XL.
MAPE = 0,062%.

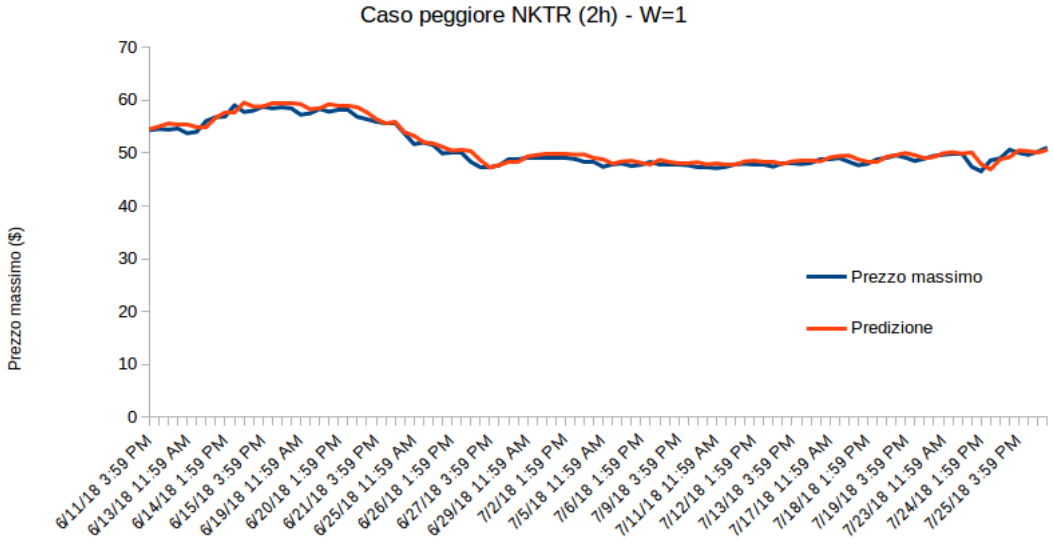

Fig. 6.33 Il più alto errore di predizione della regressione lineare, per W=1 e per i dati con granularità pari a due ore, si ottiene per l'azione NKTR.
MAPE = 1,374%.

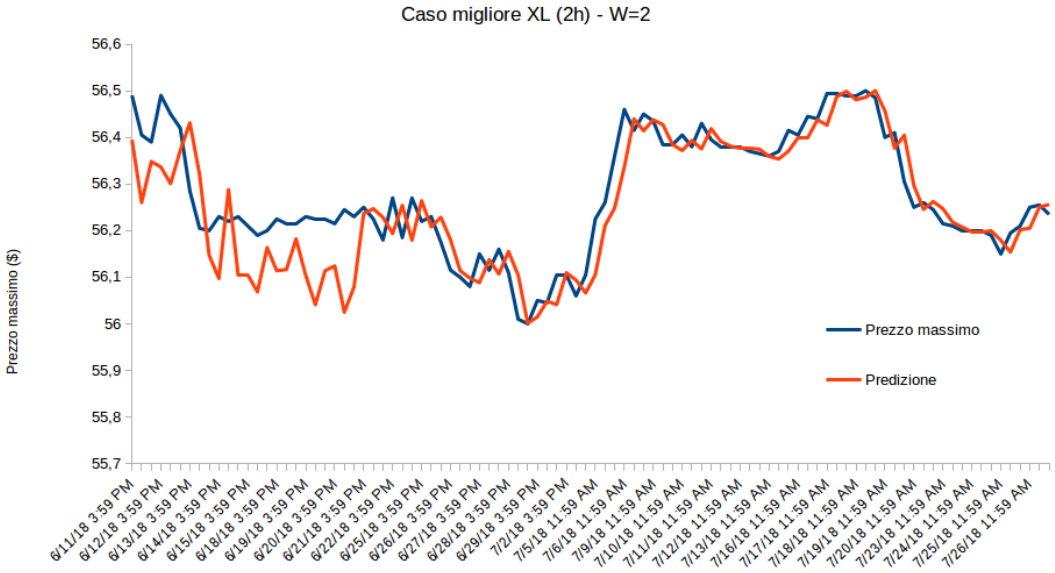

Fig. 6.34 Il più basso errore di predizione della regressione lineare, per W=2 e per i dati con granularità pari a due ore, si ottiene per l'azione XL.
MAPE = 0,093%.

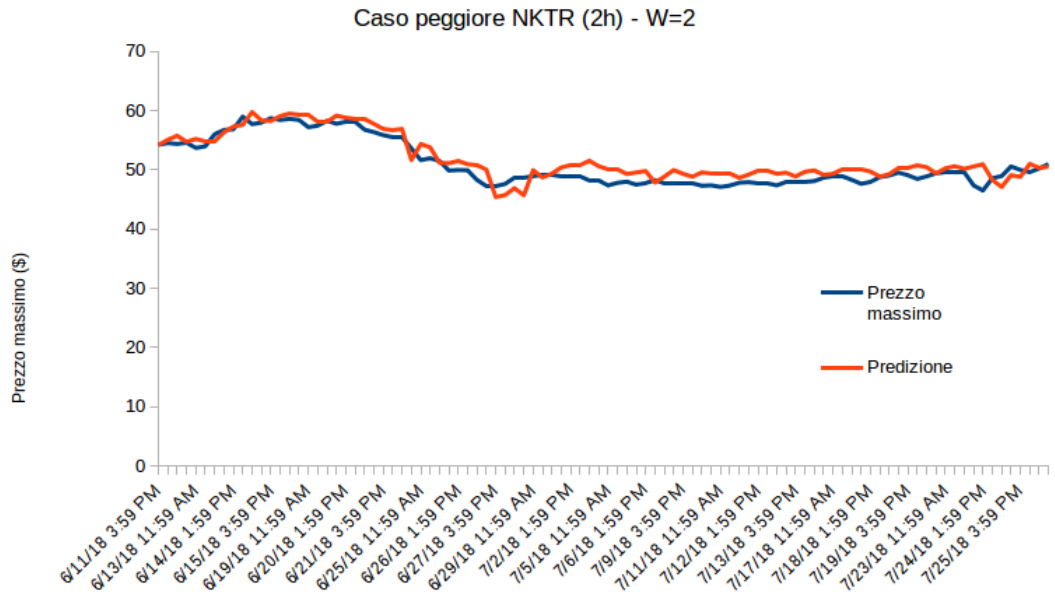

Fig. 6.35 Il più alto errore di predizione della regressione lineare, per W=2 e per i dati con granularità pari a due ore, si ottiene per l'azione NKTR.
MAPE = 2,758%.

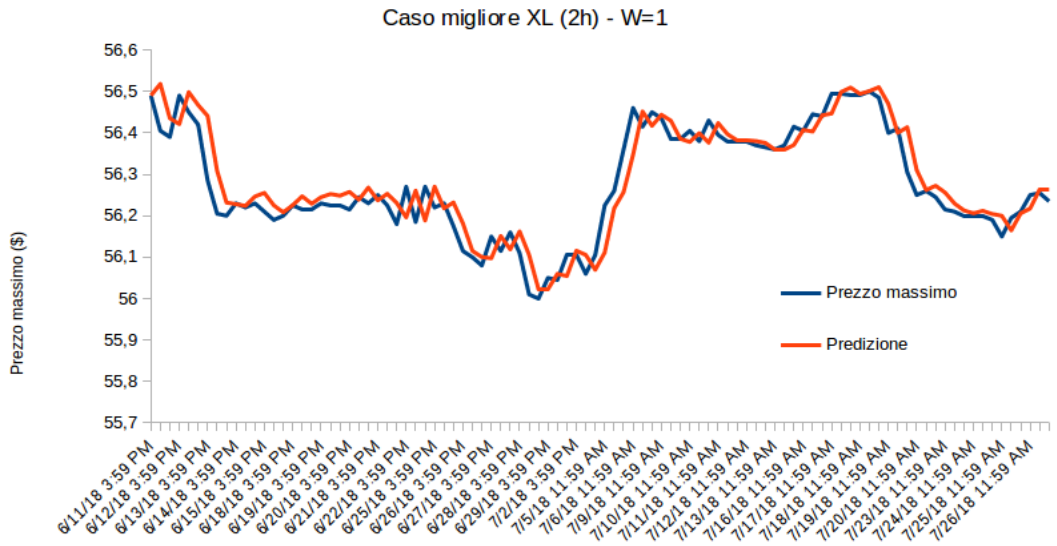

Fig. 6.36 Il più basso errore di predizione delle SVMs con kernel dot, per W=1 e per i dati con granularità pari a due ore, si ottiene per l'azione XL.
MAPE = 0,062%.

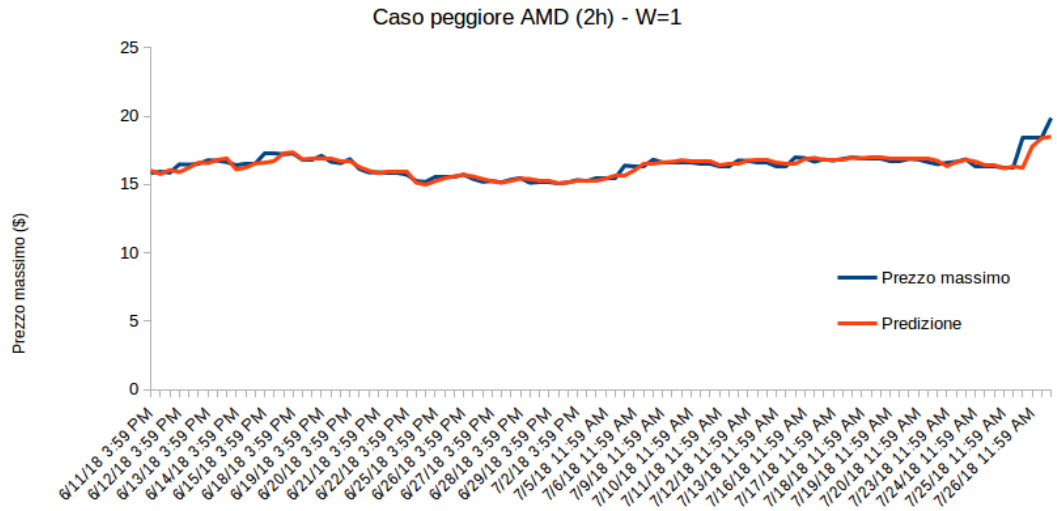

Fig. 6.37 Il più alto errore di predizione delle SVMs con kernel dot, per W=1 e per i dati con granularità pari a due ore, si ottiene per l'azione AMD.
MAPE = 1,171%.

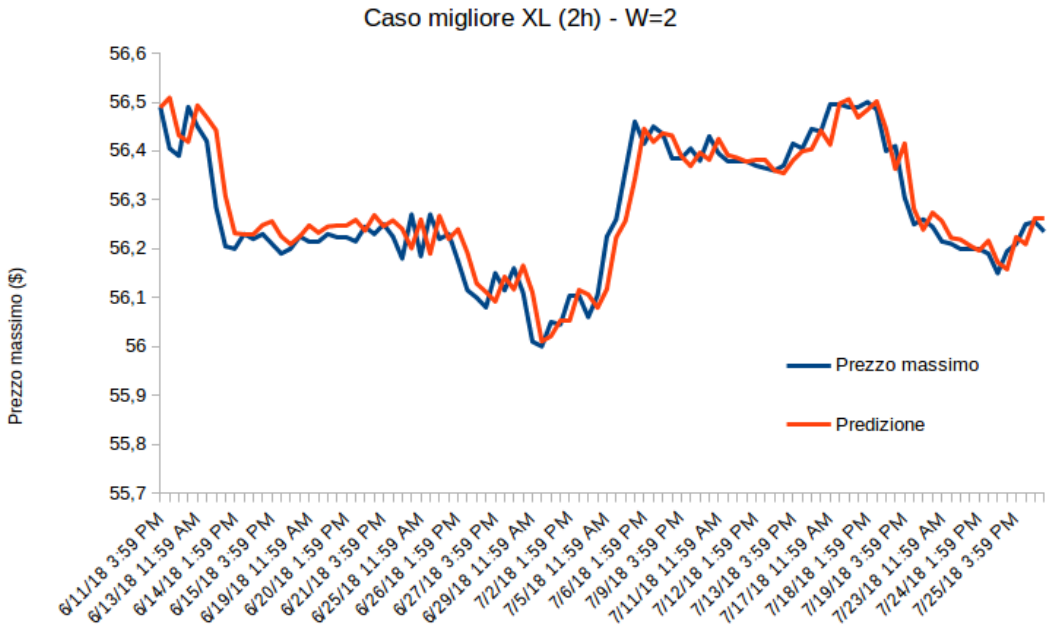

Fig. 6.38 Il più basso errore di predizione delle SVMs con kernel dot, per W=2 e per i dati con granularità pari a due ore, si ottiene per l'azione XL.

MAPE = 0,063%.

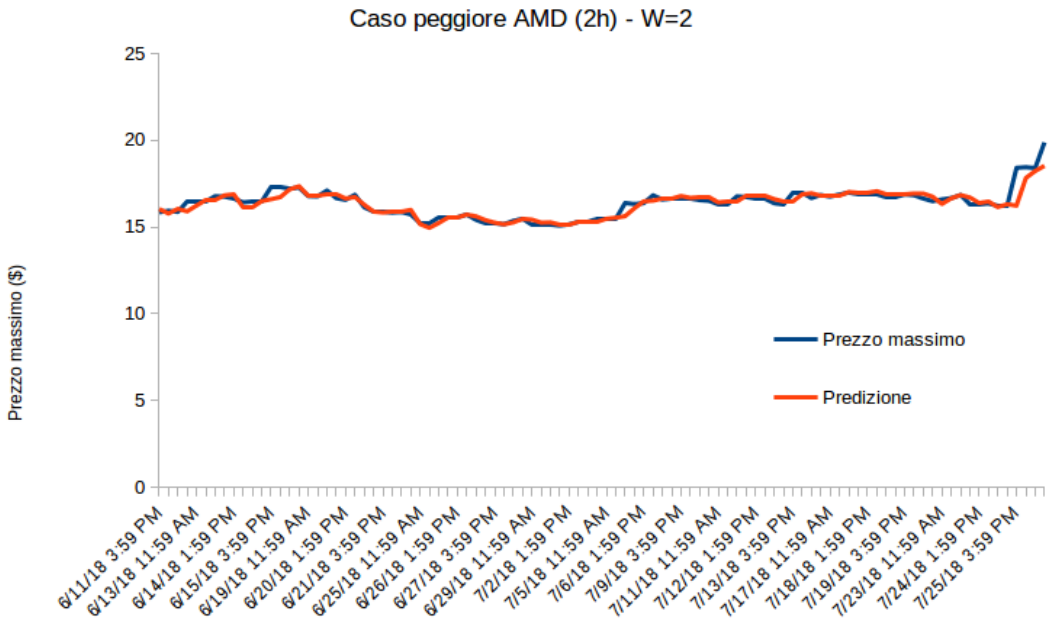

Fig. 6.39 Il più alto errore di predizione delle SVMs con kernel dot, per W=2 e per i dati con granularità pari a due ore, si ottiene per l'azione AMD.

MAPE = 1,152%.

Si noti che i casi peggiori mostrati, in termini di errore di predizione, corrispondono alle azioni per le quali le strategie considerate hanno ottenuto la precisione minore nelle predizioni; il valore di MAPE delle strategie, per queste azioni, è pari al valore di MAPE massimo indicato nelle tabelle precedenti.
Nella scelta della strategia da utilizzare, dunque, si deve tenere conto anche della deviazione standard del MAPE medio e del MAPE massimo ottenuto dalle strategie. In questo senso, nonostante le SVMs con kernel dot presentino valori bassi di MAPE medio per *W=1* e dati con granularità pari a trenta minuti (0,268%) o per *W=2* e dati con granularità oraria (0,411%), le elevate deviazioni standard della misura dell'errore (0,549% e 0,913%, rispettivamente) giustificano i casi peggiori mostrati per queste configurazioni, aventi errori di predizione elevati e pari a 9,306% e 19,511%, rispettivamente.

6.5 Scalabilità del sistema

Considerato quanto detto precedentemente al riguardo della redditività e degli errori di predizione delle varie strategie, si sono valutate le prestazioni del framework utilizzando i dati campionati con finestre a scorrimento di dimensioni piccole (*W=1* e *W=2*). Quanto viene detto è applicabile, con piccole variazioni, per gli altri valori di finestra.

La valutazione in termine di tempistiche è stata effettuata misurando i tempi di esecuzione delle tre fasi principali che il sistema, qualora messo in campo, dovrà svolgere: training dei modelli, generazione delle predizioni e generazione delle raccomandazioni.
Lo scopo della valutazione è determinare se il sistema possa essere utilizzato in tempo reale per l'investimento.

Le tre fasi sono valutate in due sottosezioni separate.

6.5.1 Addestramento dei modelli

Poichè il training dei modelli avviene sequenzialmente, allenando (per ciascuna strategia) un modello sui dati di un'azione per volta, il numero di azioni considerate non ha influenza sulla complessità di un singolo modello e sulle tempistiche di allenamento dello stesso.
Il numero di azioni ha però influenza sul tempo totale di training dei modelli relativamente alla singola strategia. L'operazione avviene sequenzialmente, dunque non è interessante mostrare la durata totale del training in funzione del numero di azioni considerato, in quanto è una relazione lineare.

Ciò che influenza la complessità dei singoli modelli ed il tempo richiesto per allenare ciascuno di loro è la dimensione della finestra di campionamento dei dati (W). Valori maggiori di questo parametro corrispondono ad un incremento degli attributi (*features*) dei modelli, i quali hanno un impatto sulla complessità degli stessi, in modo differente per i diversi metodi predittivi.

I grafici da **Fig. 6.40** a **Fig. 6.42** mostrano i tempi totali di training (per tutti i 496 stock) dei differenti metodi predittivi considerati, al variare della dimensione della finestra e per ciascuna granularità dei dati. Si noti che tali tempistiche non sono confrontabili tra le diverse granularità, in quanto, per uno stesso periodo considerato, le aggregazioni maggiori corrispondono ad un numero minore di record del dataset utilizzato per l'addestramento dei modelli.

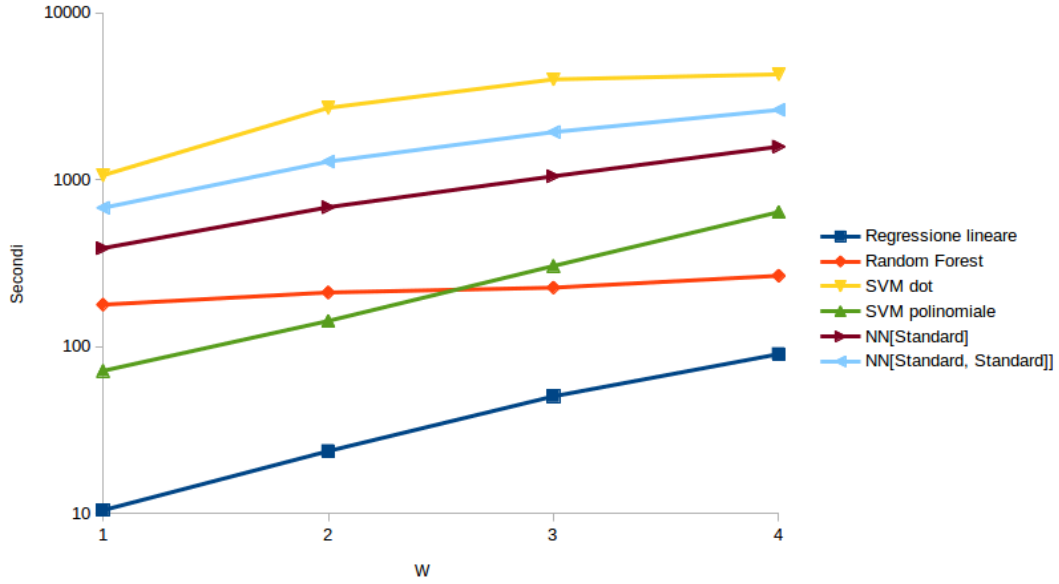

Fig. 6.40 Tempistiche di training delle strategie al variare della dimensione della finestra di campionamento. Dati con granularità pari a trenta minuti.

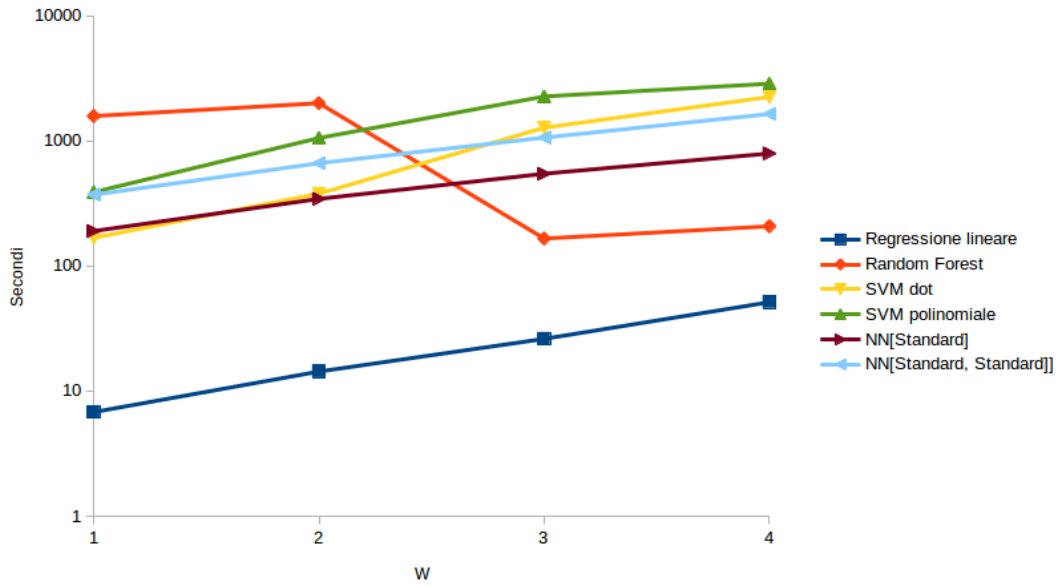

Fig. 6.41 Tempistiche di training delle strategie al variare della dimensione della finestra di campionamento. Dati con granularità oraria.

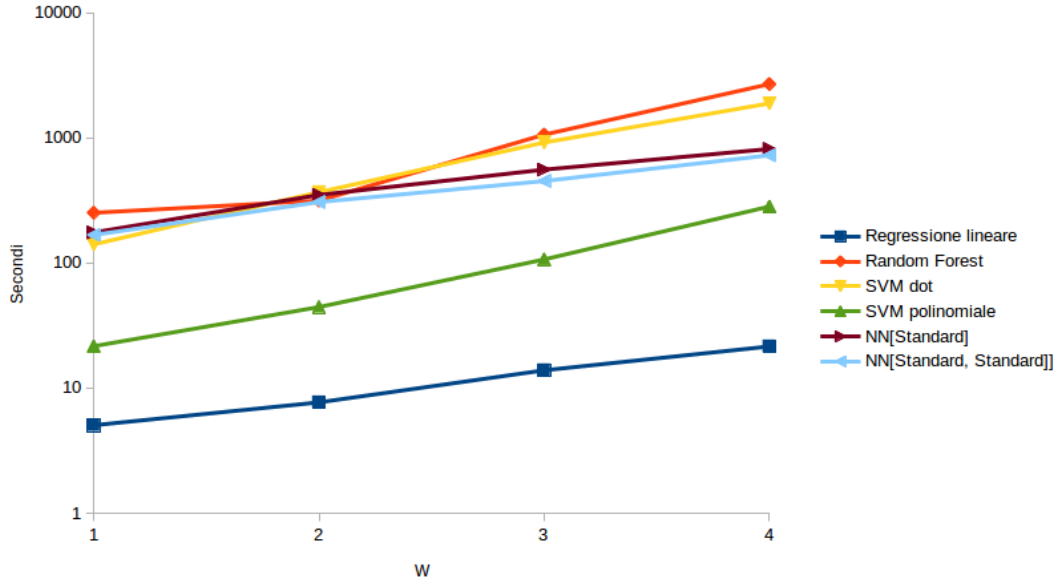

Fig. 6.42 Tempistiche di training delle strategie al variare della dimensione della finestra di campionamento. Dati con granularità pari a due ore.

Dai grafici mostrati si deduce che, a causa del numero ridotto di attributi, la regressione lineare è l'algoritmo più veloce da allenare per tutte le granularità dei dati. Il tempo richiesto per allenare tutti i modelli della regressione lineare relativi ai singoli stock è quasi sempre nell'ordine dei dieci secondi, eccetto che per i dati con granularità pari a trenta minuti, per i quali si mantiene comunque inferiore ai cento secondi.
Gli altri metodi predittivi richiedono tempi di training decisamente superiori.

Per ottenere tutti i modelli allenati di una strategia scelta e poter operare sul mercato in tutti gli slot temporali di una giornata di borsa è opportuno eseguire il training dei modelli nei periodi di chiusura del mercato borsistico (es. a fine giornata). In questo modo, tutte le strategie considerate sono attuabili, per quel che riguarda le tempistiche di addestramento.

6.5.2 Generazione delle predizioni e delle raccomandazioni

Per quanto riguarda le predizioni, i tempi richiesti sono quelli per la (eventuale) normalizzazione del record contenente i dati della finestra relativi alla singola azione e per il calcolo della valore della funzione relativa al modello allenato usando il record come input, per ciascun modello della strategia adottata.
In questa analisi delle tempistiche non si è tenuto conto del tempo necessario a generare i record relativi a tutte le azioni, ma questo tempo è trascurabile, in quanto è possibile produrre i record in modo incrementale durante l'intervallo precedente a quello scelto per generare le predizioni e per l'intervento sul mercato (es. i valori di massimo e minimo, così come quelli relativi al volume di compravendita e al numero delle transazioni sono aggiornabili in tempo reale).

Il tempo richiesto dalla generazione delle raccomandazioni, invece, consiste nel tempo necessario all'algoritmo per calcolare un numero di valori della funzione di fitness pari al numero delle azioni considerate, inserire i valori in due strutture dati ordinate (una per le posizioni lunghe, una per quelle corte) e restituire i primi k simboli delle strutture. Si noti che il tempo necessario per queste operazioni è indipendente dalla complessità dei modelli scelti e dalla dimensione della finestra di campionamento: l'algoritmo richiede soltanto il valore della predizione e del prezzo di chiusura relativo all'intervallo precedente per ciascuno stock considerato.

Il grafico in **Fig. 6.43** mostra i tempi cumulati delle due fasi descritte in questa sezione, per ciascun metodo predittivo ed al variare del numero delle azioni. Poichè le variazioni della dimensione della finestra di campionamento e della granularità dei dati, le quali influenzano solo i tempi di generazione delle predizioni, causano differenze molto piccole e non significative (es. < 0,1s), si è deciso di mostrare i tempi solo per $W=2$ e per la granularità oraria dei dati.

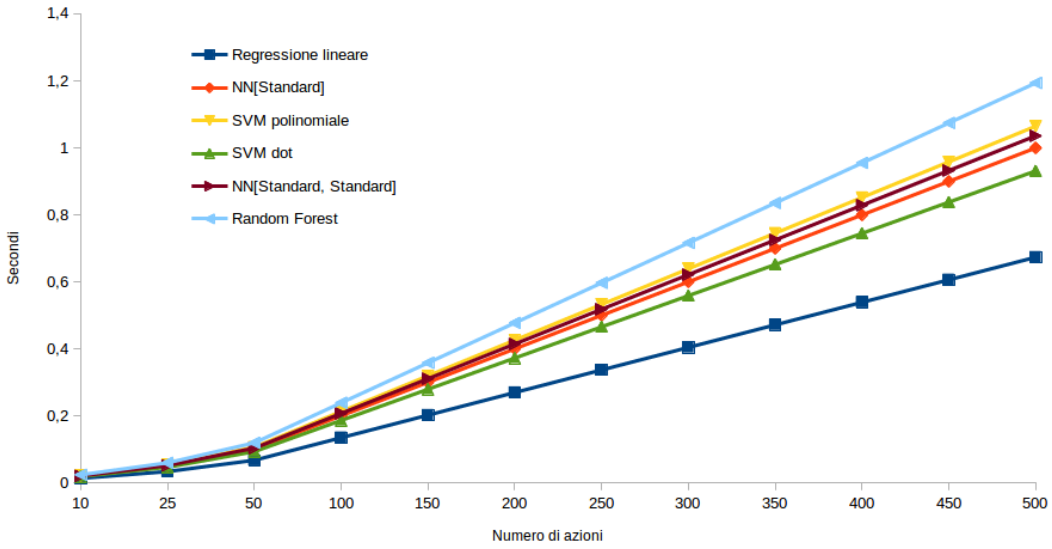

Fig. 6.43 Tempistiche cumulate delle operazioni da eseguire in tempo reale (online) per ciascuna strategia, al variare del numero di azioni considerate. Dati con granularità oraria e W=2.

La componente delle curve legata al tempo di generazione delle raccomandazioni è uguale per tutte le strategie e cresce linearmente all'aumentare del numero delle azioni considerate. Tali tempistiche sono riducibili mediante un'implementazione concorrente dell'algoritmo di raccomandazione delle azioni, avendo cura di arbitrare l'accesso concorrente alle strutture dati.
Le piccole differenze tra le tempistiche delle varie strategia sono causate esclusivamente dalla fase di generazione delle predizioni.

Il tempo totale delle operazioni da eseguire online è, in media, inferiore a 1,2 secondi, il che rende sempre possibile aprire una posizione per lo stock consigliato pochi secondi dopo l'apertura dell'intervallo scelto per l'investimento, tenendo conto di eventuali ritardi.

6.6 Valutazione complessiva delle strategie

Tenendo conto di quanto detto al riguardo dei tempi necessari al framework per operare, tutte le strategie considerate sono attuabili in termine di tempistiche, posto che si effettui l'addestramento dei modelli offline.

Occorre, dunque, verificare quali tra tutte le strategie proposte permettono di ottenere il guadagno potenziale complessivo maggiore nel periodo considerato per la validazione del framework, in relazione all'errore di predizione delle stesse strategie. Nella scelta della strategia da adottare, bisogna tenere conto dell'errore di predizione associato e cercare un compromesso tra la strategia con l'errore più basso e quella con il guadagno potenziale complessivo più alto.
L'obiettivo primario di un modello predittivo è, di fatto, quello di ottenere livelli adeguati di precisione/accuratezza. Le strategie con alto errore di predizione e alti valori di guadagno potenziale potrebbero, infatti, essere non affidabili e i guadagni potenziali derivanti da esse potrebbero essere non replicabili in fase di applicazione.

L'approccio mostrato in questo paragrafo consiste sì nello scartare le strategie con alto errore di predizione in media su tutte le azioni (MAPE medio della strategia), ma anche nello scartare quelle strategie che presentano valori di MAPE per alcune azioni troppo lontani dal valore del MAPE medio delle strategie stesse (*outliers*). In questo modo, si definisce l'insieme delle strategie ammissibili e si individuano in esso le migliori in termini di profitto potenziale complessivo.

Entrambi gli approcci possono essere descritti come casi particolari del problema di ottimizzazione vincolata

$$max\ guadagno(j)$$

$$\text{con } errore_j \leq \delta$$

dove $j \in J$ è la strategia considerata, intesa come tripletta metodo predittivo-granularità dei dati-dimensione della finestra di campionamento, J è l'insieme delle strategie considerate in questo elaborato e δ è la soglia di tolleranza di una quantità scelta relativamente all'errore di predizione della strategia j-esima.

Nel seguito si tratterà questo tipo di problema con riferimento al secondo approccio, considerando cioè non solo l'errore medio di predizione della strategia ma anche la presenza di eventuali *outliers*, in modo da definire l'insieme delle strategie ammissibili.

Premesso ciò, nei grafici da **Fig. 6.44** a **Fig. 6.49** sono mostrate le distribuzioni del MAPE di ciascuna strategia su tutte le azioni considerate, per tutte le dimensioni di finestra e le granularità dei dati considerate.

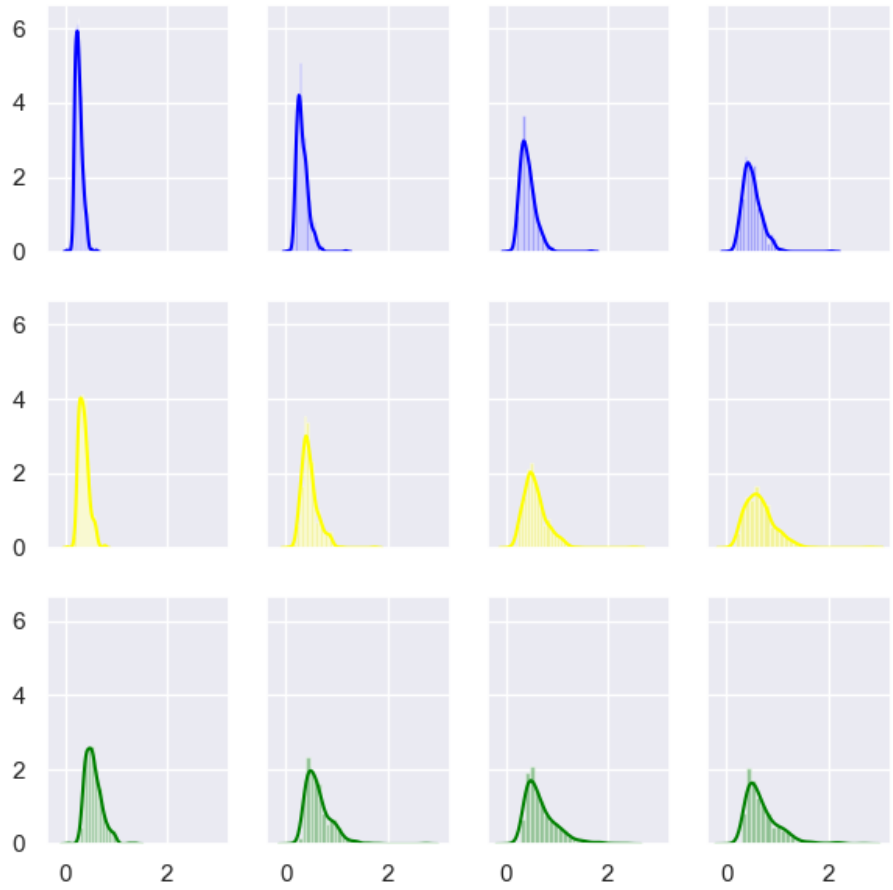

Fig. 6.44 Distribuzione del MAPE su tutte le azioni usando la regressione lineare. In orizzontale sono mostrate le W (da 1 a 4), in verticale le tre granularità considerate (blu per i trenta minuti, giallo per quella oraria e verde per le due ore).

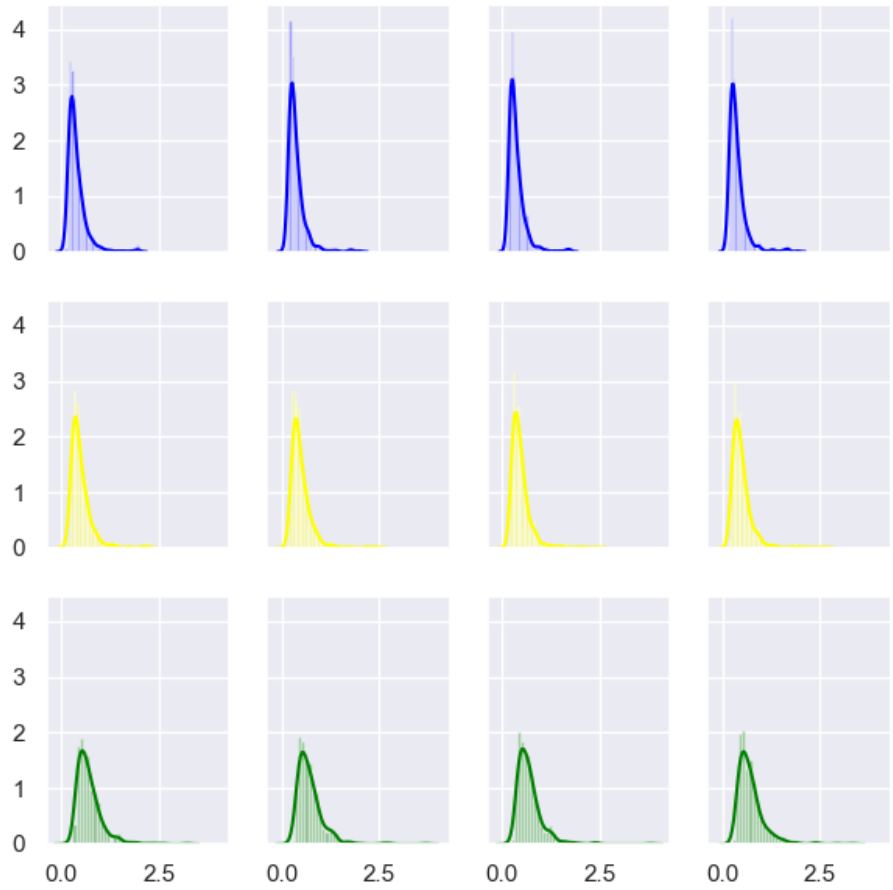

Fig. 6.45 Distribuzione del MAPE su tutte le azioni usando NN[Standard]. In orizzontale sono mostrate le W (da 1 a 4), in verticale le tre granularità considerate (blu per i trenta minuti, giallo per quella oraria e verde per le due ore).

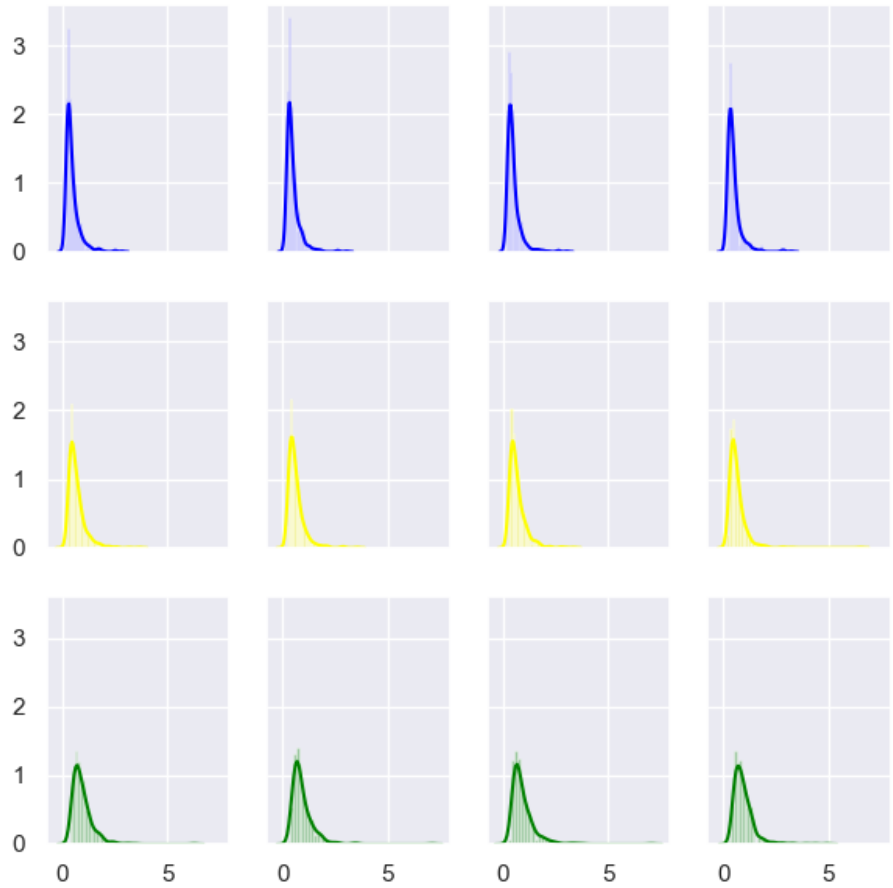

Fig. 6.46 Distribuzione del MAPE su tutte le azioni usando NN[Standard, Standard]. In orizzontale sono mostrate le W (da 1 a 4), in verticale le tre granularità considerate (blu per i trenta minuti, giallo per quella oraria e verde per le due ore).

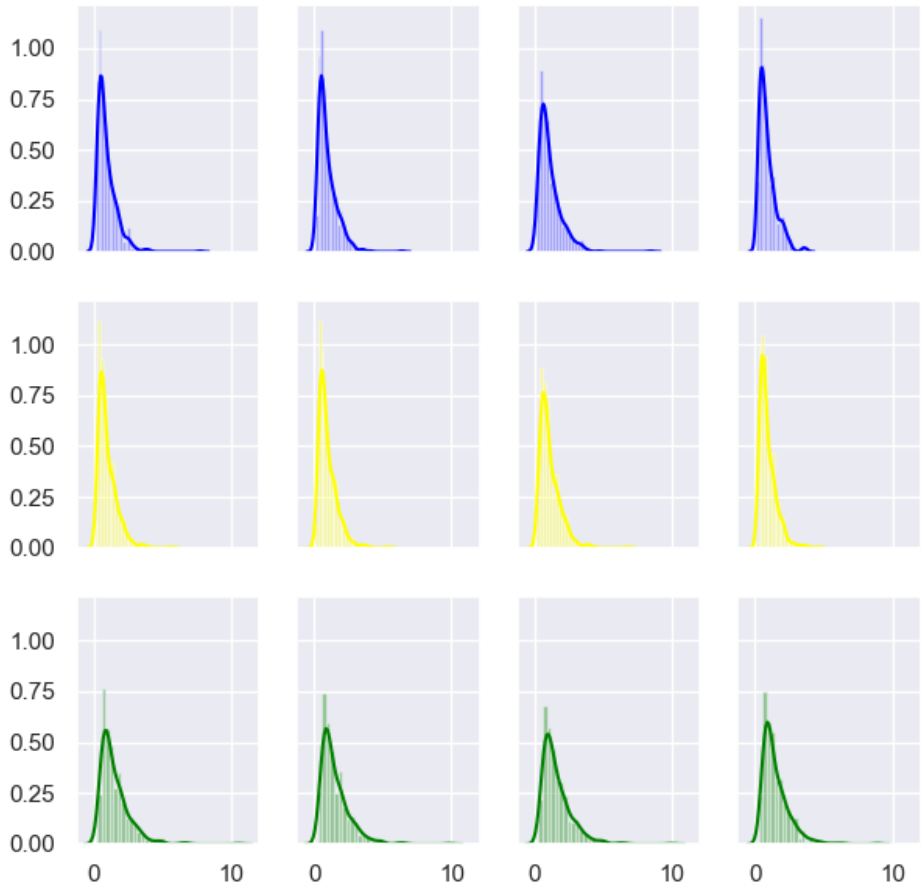

Fig. 6.47 Distribuzione del MAPE su tutte le azioni usando il Random Forest. In orizzontale sono mostrate le W (da 1 a 4), in verticale le tre granularità considerate (blu per i trenta minuti, giallo per quella oraria e verde per le due ore).

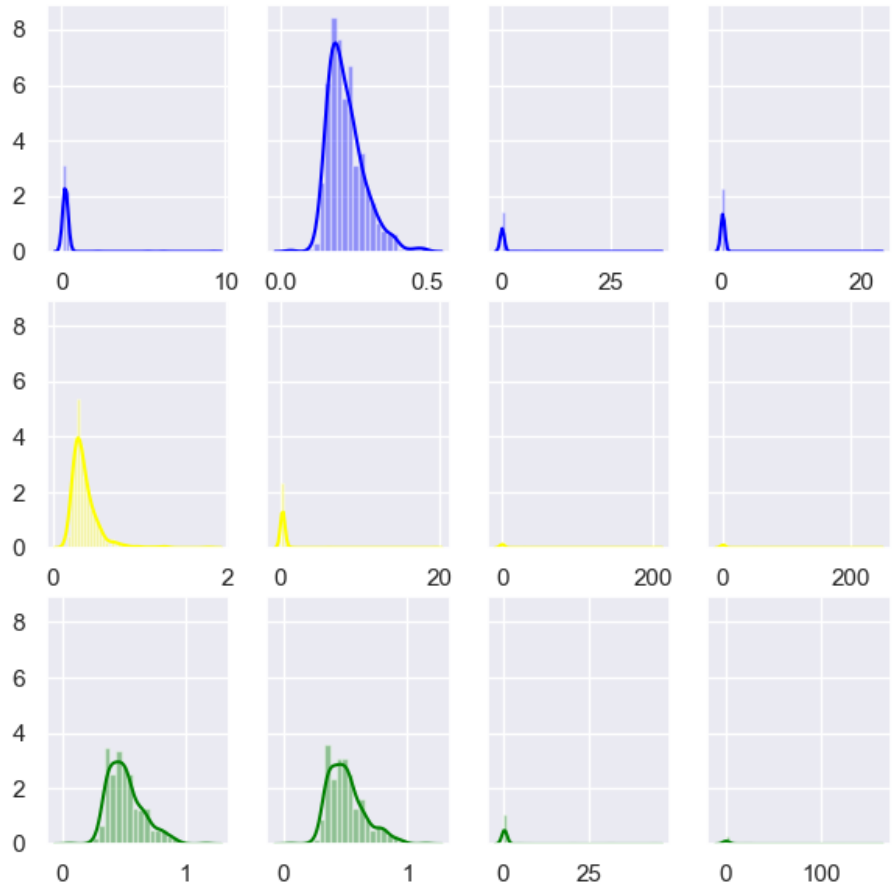

Fig. 6.48 Distribuzione del MAPE su tutte le azioni usando le SVMs con kernel dot. In orizzontale sono mostrate le W (da 1 a 4), in verticale le tre granularità considerate (blu per i trenta minuti, giallo per quella oraria e verde per le due ore).

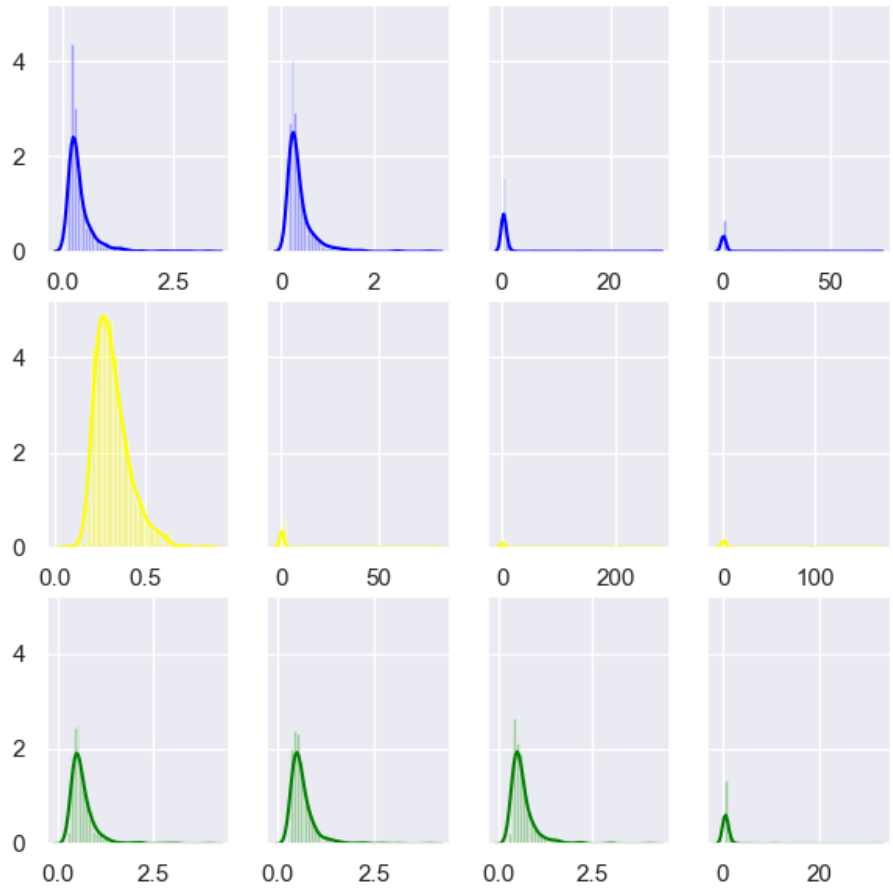

Fig. 6.49 Distribuzione del MAPE su tutte le azioni usando le SVMs con kernel polinomiale. In orizzontale sono mostrate le W (da 1 a 4), in verticale le tre granularità considerate (blu per i trenta minuti, giallo per quella oraria e verde per le due ore).

Si vogliono dunque analizzare il guadagno potenziale massimo e quello minimo delle strategie ammissibili, al variare di alcune soglie δ_k di tolleranza dell'errore.
Data una soglia di tolleranza $\delta_k \in \Delta$, il guadagno potenziale massimo è quello della strategia che ottiene il più alto guadagno potenziale complessivo; la strategia appartiene ad un sottoinsieme di quelle considerate in questo lavoro di tesi, ed è il sottoinsieme delle strategie ammissibili, cioè le strategie che rispettano il seguente vincolo per l'errore

$$MAPE\_medio_j + 2\,\sigma_j \leq \delta_k ,$$

per ogni strategia $j \in J$, dove *J* è l'insieme delle strategie considerate e σ_j è la deviazione standard del MAPE mediato su tutte le azioni relativo alla strategia *j*.
Questo vincolo limita la scelta delle strategie d'investimento in relazione all'errore medio (in termini di MAPE) e alla sua dispersione che si accetti le strategie compiano sulle azioni considerate.
Allo stesso modo, il guadagno potenziale minimo è quello della strategia (appartentente al sottoinsieme delle strategie ammissibili) che ottiene il più basso guadagno potenziale complessivo.

Le soglie di tolleranza sono scelte in modo da mostrare le maggiori variazioni delle due grandezze. Esse corrispondono ad un sottoinsieme dei valori delle quantità scelte come parte sinistra del vincolo, relativi a tutte le strategie e ordinati in modo decrescente. A tale sottoinsieme appartengono i valori più significativi per lo scopo, ossia le soglie che causano una variazione delle quantità analizzate (guadagno potenziale massimo e minimo); la variazione è causata dall'esclusione di una delle due strategie corrispondenti, dovuta al fatto che la strategia in questione non rispetta il vincolo per una determinata soglia.

La scelta di includere nel vincolo il doppio della deviazione standard deriva da considerazioni empiriche sulla distribuzione del MAPE delle strategie considerate su tutte le azioni. In particolare, come mostrato in **Tabella 22**, il numero di azioni per cui il relativo errore di predizione commesso da ciascuna strategia è minore o uguale la somma del MAPE medio della strategia e di due volte la deviazione standard è pari ad almeno il 94,556% del numero di azioni totali, considerando tutte le strategie.
Scegliendo di formulare il vincolo in questo modo, si escludono, per valori bassi delle soglie, le strategie aventi un valore basso di MAPE medio ma un valore alto della sua deviazione standard: è preferibile, infatti, che le strategie considerate commettano un errore di predizione basso su quante più azioni possibili, piuttosto che un errore di predizione basso in media ma elevato su un numero piccolo di azioni.

Tabella 22
Percentuale di azioni per le quali l'errore commesso dalle strategie è inferiore o uguale all'errore medio della strategia (tra tutte le azioni) sommato a due volte la sua deviazione standard.

Percentuale delle azioni con errore minore o uguale al MAPE medio della strategia più due volte la deviazione standard, per ciascuna strategia.

	Granularità	W	%
Regressione lineare	1h	1	94,556
		2	95,363
		3	95,766
		4	95,766
	2h	1	95,565
		2	96,573
		3	95,363
		4	95,968
	30m	1	95,565
		2	95,363
		3	96,169
		4	96,371
NN[Standard]	1h	1	96,371
		2	96,976
		3	96,573
		4	96,573
	2h	1	95,565
		2	97,581
		3	96,976
		4	95,968
	30m	1	96,169
		2	96,169
		3	96,169
		4	96,169
NN[Standard, Standard]	1h	1	95,565
		2	95,766
		3	95,363
		4	96,774
	2h	1	96,976
		2	97,177
		3	96,169
		4	96,371
	30m	1	95,363
		2	95,565
		3	96,169
		4	95,968
Random Forest	1h	1	95,766
		2	95,766
		3	96,371
		4	95,968
	2h	1	96,371

		2	95,766
		3	95,766
		4	95,968
		1	95,363
	30m	2	95,565
		3	95,565
		4	94,758
		1	95,968
	1h	2	99,597
		3	98,790
		4	97,782
		1	94,556
SVM dot	2h	2	94,758
		3	99,194
		4	97,984
		1	99,194
	30m	2	95,565
		3	99,597
		4	99,798
		1	95,565
	1h	2	98,992
		3	98,992
		4	99,194
		1	96,573
SVM polinomiale	2h	2	95,968
		3	95,565
		4	98,185
		1	96,169
	30m	2	95,565
		3	99,194
		4	98,589

Nel grafico in **Fig. 6.50** si mostra quanto detto. L'insieme delle soglie è l'insieme

$$\Delta = \{0{,}351\%, 0{,}407\%, 0{,}553\%, 0{,}727\%, 1{,}052\%, 3{,}605\%, 8.591\%\} .$$

Il guadagno massimo per $\delta_k \geq 0{,}727\%$ è dato dalla regressione lineare con i dati aventi granularità pari a trenta minuti e campionati con finestra di dimensione 3. Questa strategia presenta un profitto potenziale complessivo pari a +183,658%.
Decrementando la soglia di tolleranza, questa strategia viene scartata ed il corrispondente guadagno potenziale massimo diminuisce. Si noti che per soglie $\delta_k \geq 0{,}727\%$ nell'insieme delle strategie ammissibili sono incluse anche strategie con

guadagno potenziale prossimo allo +0% (es. SVM kernel polinomiale-30m-2 e SVM kernel polinomiale-30m-3).

Il guadagno potenziale massimo in corrispondenza della soglia più bassa vale +70,133% ed è dato dalle SVMs con kernel dot e dati con granularità pari a trenta minuti, campionati con finestra di dimensione 2. Si noti che al prezzo di un aumento irrisorio dell'errore tollerato (pari a 0,056%) si ha un aumento significativo del guadagno potenziale complessivo (guadagno potenziale +103,986%), dato dalla regressione lineare con granularità dei dati pari a trenta minuti e dati campionati con finestra di dimensione 1.
Queste due strategie sono, dunque, le strategie consigliate per l'investimento. Esse vantano un errore di predizione minore dello 0,408% sul 95,565% del totale delle azioni considerate, bassi valori di MAPE massimo (0,499% e 0,609% rispettivamente) e ottimi guadagni potenziali derivanti dalle raccomandazioni per le posizioni lunghe e corte.

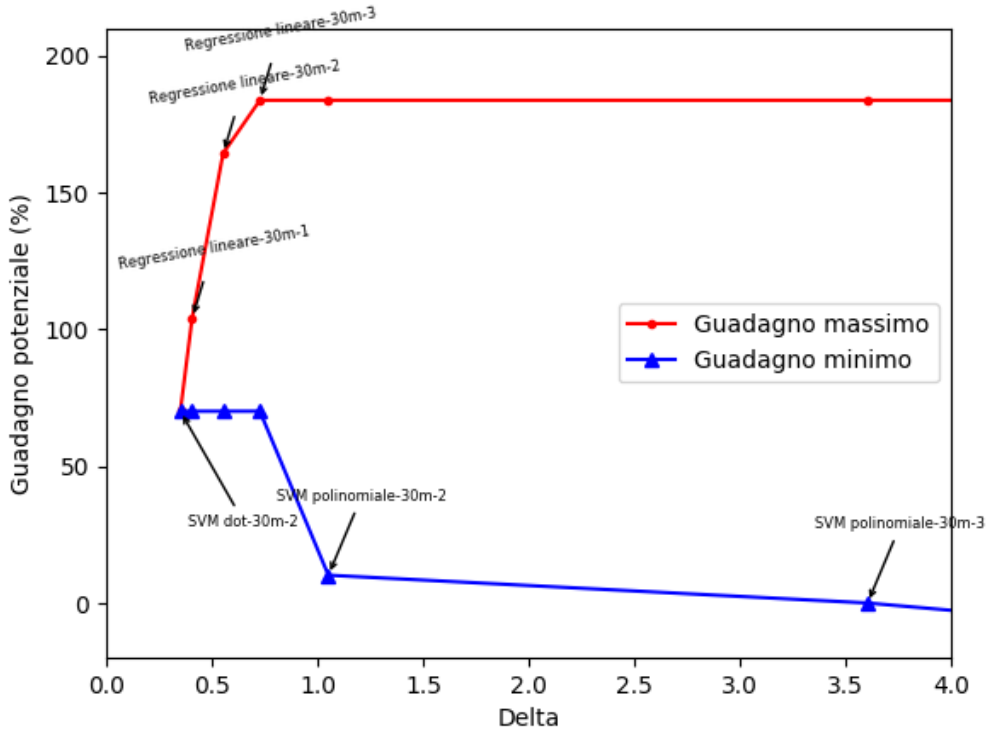

Fig. 6.50 Sono mostrati il guadagno potenziale massimo (rosso) e quello minimo (blu), dati rispettivamente dalla migliore e dalla peggiore strategia, scelte tra quelle che rispettano il vincolo descritto, al variare della soglia di tolleranza δ_k *. Le strategie corrispondenti sono indicate nel formato metodo_predittivo-granularità_dei_dati-dimensione_finestra.*

7. Conclusioni e sviluppi futuri

In questo lavoro di tesi si è progettato e validato un trading system quantitavo che analizzasse i dati storici intraday delle azioni con lo scopo di guidare gli investimenti seguendo la strategia del day trading.
L'approccio proposto, basato su alcuni dei metodi predittivi in grado di effettuare regressione e su tecniche di *windowing* per il campionamento dei dati, stima il prezzo massimo delle azioni negli intervalli considerati per l'investimento, e le predizioni vengono poi utilizzate per raccomandare le azioni su cui investire. Può essere utilizzato dunque per definire il *target price* di una posizione.

Per poter investire nei diversi momenti della giornata di mercato borsistico si è reso necessario, innanzitutto, progettare ed implementare un crawler specifico per una API che fornisse i dati storici azionari con frequenza di campionamento maggiore di quella giornaliera. L'API scelta per lo scopo è la *Investitors Exchange API*, la quale fornisce i dati storici azionari con le granularità pari al minuto e al giorno.
Il crawler è stato implementato seguendo il pattern *abstract factory*, che permette di estenderlo in futuro con eventuali nuove implementazioni specifiche. Esso è stato progettato per raccogliere i dati storici delle azioni, specificate con una lista di simboli fornita come input, con entrambe le granularità disponibili. L'implementazione specifica per i dati storici intraday fa uso di tecniche della programmazione concorrente per rendere l'operazione di raccolta efficiente e scalabile con il numero delle azioni delle quali i dati sono richiesti.
Sono stati raccolti i dati storici relativi ad un periodo di circa sei mesi del mercato azionario, relativamente alle azioni dell'indice *S&P 500*. I dati raccolti sono stati poi preprocessati con operazioni di pulizia e aggregazione, in modo da ottenere tre dataset strutturati a diversa granularità (30 minuti, un'ora e due ore), da utilizzare per l'addestramento dei modelli.

Il training dei modelli, realizzato mediante l'uso del tool RapidMiner, consta di un processo per la calibrazione dei parametri relativi ai metodi predittivi considerati e di un processo che allena i modelli effettivamente, utilizzando i parametri stimati nel primo processo. Ciascun modello è stato allenato con i dati storici di un'azione per volta, utilizzando il prezzo massimo dell'azione, il prezzo minimo, il volume di compravendita e il numero di transazioni di ciascuno slot temporale (di durata pari alla granularità considerata) del periodo scelto per il training, per tutte le azioni e per ciascuna delle tre granularità. I dati sono stati campionati con una finestra a scorrimento che mettesse in relazione i dati storici relativi ad un numero fissato di slot temporali passati; tale numero è definito dalla dimensione della finestra, e si sono testati i valori da 1 a 4 per questa grandezza.
I modelli allenati, dati in input i record relativi ai dati storici delle azioni campionati con la finestra, predicono il prezzo massimo delle azioni nell'intervallo successivo all'ultimo presente nella finestra di input.

È stato poi implementato un algoritmo di raccomandazione delle azioni che, ricevendo in input le predizioni dei modelli allenati su tutti gli stock, permette di ordinare gli stock in base all'aumento di prezzo previsto, il quale è stato stimato confrontando il valore delle azioni alla chiusura dell'intervallo precedente con le predizioni; vengono restituite in output le *k* azioni più promettenti per l'investimento, valutando sia le posizioni corte che quelle lunghe.
Nell'implementazione dell'algoritmo si è tenuto conto dei costi delle transazioni (fissati allo 0,15% per semplicità) e di una soglia di guadagno desiderato, che permette di filtrare le raccomandazioni.

La campagna di esperimenti effettuata ha permesso di valutare l'efficacia e l'efficienza del sistema, confrontando i risultati ottenuti con i vari metodi predittivi al variare della dimensione della finestra di campionamento e delle diverse aggregazioni dei dati.
Per allenare i modelli sono stati utilizzati i dati storici delle azioni relativi a 78 giorni di mercato, e i modelli allenati sono stati valutati in base all'efficacia delle predizioni

relative agli 8 giorni di mercato successivi. Questa valutazione è stata ripetuta per 4 volte, allenando nuovamente i modelli dopo aver fatto avanzare la finestra dei dati di training in modo da includere gli 8 giorni usati per la predizione dai modelli precedenti. Il sistema proposto è stato valutato in termini di redditività delle strategie, precisione delle predizioni e scalabilità. Per ottenere una valutazione dei profitti, si è valutato il guadagno potenziale della posizione, ottenuto chiudendo la posizione nel miglior momento possibile dell'intervallo scelto per l'intervento sul mercato.

I risultati ottenuti hanno mostrato che è preferibile utilizzare valori piccoli per la finestra di campionamento e la granularità minore tra le tre considerate, in quanto la qualità delle predizioni peggiora al crescere della dimensione della finestra e al diminuire della frequenza di campionamento dei dati.
L'errore di predizione commesso dalle strategie, di fatto, vincola le strategie ammissibili per l'investimento: alcune tra le strategie valutate, che hanno mostrato alti guadagni potenziali complessivi elevati, hanno anche riportato un errore di predizione non trascurabile, dunque tali guadagni potrebbero non essere replicabili in fase applicativa. La regressione lineare e le SVMs con kernel dot sono risultati essere i metodi predittivi migliori in termine di precisione delle predizioni sotto le condizioni descritte, e le raccomandazioni generate da questi modelli hanno mostrato ottimi guadagni potenziali in percentuale.
Dal punto di vista dell'analisi delle tempistiche e della scalabilità del framework, si è mostrato come tutte le strategie sono potenzialmente attuabili in tempo reale sul mercato, posto che il training dei modelli venga eseguito nei momenti di chiusura del mercato. Il sistema è riuscito sempre a produrre le predizioni e le raccomandazioni in tempo utile per l'apertura delle posizioni consigliate pochi secondi dopo l'inizio dell'intervallo scelto per il trading.

Per quanto riguarda gli sviluppi futuri, sarebbe opportuno estendere la dimensione dei dataset strutturati, contenenti i dati storici delle azioni, utilizzati in questo lavoro di tesi

(es. da circa sei mei ad un anno di mercato azionario) in modo da valutare l'uso di dati a maggiore granularità oltre alle tre già valutate in questo elaborato (es. 3h, 5h).

Si potrebbe, inoltre, valutare l'estensione dei dataset strutturati prodotti dal crawler in modo da includere ulteriori caratteristiche e informazioni sulle azioni, quali indicatori tecnici complessi comunemente utilizzati nell'analisi tecnica del mercato azionari. A questo scopo, basta estendere l'insieme *M* delle grandezze misurabili delle azioni.

Le tecniche adoperate per lo sviluppo di questo trading system sono, inoltre, generalmente applicabili su qualsiasi altro indice azionario o mercato finanziario. Si potrebbe considerare la loro applicazione sulle azioni della borsa italiana (*FTSE MIB*) o sui mercati delle valute quali il *Foreign Exchange* (*Forex*) ed il mercato delle *cryptovalute*. Nell'ultimo caso, bisognerà però tenere conto della maggiore volatilità che caratterizza le cryptovalute rispetto alle azioni.

Una delle possibilità di maggiore interesse, è quella di integrare quanto sviluppato in questo lavoro di tesi con un modulo che predica i valori dei prezzi minimi delle azioni per gli stessi intervalli. Avere entrambi i tipi di predizione (prezzo massimo e prezzo minimo) permette agli investitori di calcolare il rapporto *remunerazione/rischio* (il cosiddetto *risk/reward ratio* o R/R) di un investimento. Tale rapporto è molto usato dai trader per confrontrare i guadagni attesi di un investimento con l'ammontare del rischio preso per realizzare il profitto.
Una volta ottenuti i due prezzi stimati e calcolato il rapporto R/R per ciascuna azione, si potrebbe scegliere di investire sull'azione il cui R/R stimato è il più alto tra quelli calcolati, e utilizzare il valore del prezzo massimo predetto come *target price*, mentre il valore del prezzo minimo predetto può essere usato come *stop loss*.

Bibliografia e riferimenti

[1] Investopedia Staff, Technical Analysis. Ricavate da https://www.investopedia.com/terms/t/technicalanalysis.asp
[2] Investopedia Staff, Fundamental Analysis. Ricavate da https://www.investopedia.com/terms/f/fundamentalanalysis.asp
[3] Investopedia Staff, Quantitative Trading. Ricavate da https://www.investopedia.com/terms/q/quantitative-trading.asp
[4] Investopedia Staff, Quantitative Analysis (QA) website, .
[5] Noor, Faizul F. and Hossain, Mohammad Farhad, "A Quantitative Neural Network Model (Qnnm) for Stock Trading Decisions", Jahangirnagar Review, Part II: Social Science(2005) 177-194.
[6] Li, Zi-Yu & Zhang, Yuan-Biao & Zhong, Jia-Yu & Yan, Xiao-Xu & Lv, Xin-Guang, "Research on Quantitative Trading Strategy Based on Neural Network Algorithm and Fisher Linear Discriminant", International Journal of Economics and Finance(2017) 1916-971X.
[7] Kamil Zbikowski, "Using Volume Weighted Support Vector Machines with walk forward testing and feature selection for the purpose of creating stock trading strategy", Expert Systems with Applications(2015) 1797-1805.
[8] Lamartine Almeida Teixeira, Adriano Lorena Inácio de Oliveira, "A method for authomatic stock trading combining technical analysis and nearest neighbor classification", Expert Systems with Applications(2010) 6885–6890.
[9] Elena Baralis, Luca Cagliero, Tania Cerquitelli, Paolo Garza, Fabio Pulvirenti, "Discovering profitable stocks for intraday trading", Information Sciences(2017) 91-106.
[10] Tai-liang Chen, Feng-yu Chen, "An intelligent pattern recognition model for supporting investment decisions in stock market", Information Sciences(2016) 261-274.
[11] Tomer Geva, Jacob Zahavi, "Empirical evaluation of an automated intraday stock recommendation system incorporating both market data and textual news", Decision Support Systems(2014) 212-223.
[12] Wen-Chyuan Chiang, David Enke, Tong Wu, Renzhong Wang, "An adaptive stock index trading decision support system", Expert Systems With Applications(2016) 195-207.

[13] Yakup Kara, Melek Acar Boyacioglu, Ömer Kaan Baykan, "Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange", Expert Systems with Applications(2011) 5311-5319.
[14] Mohamed M. Mostafa, "Forecasting stock exchange movements using neural networks: Empirical evidence from Kuwait", Expert Systems with Applications(2010) 6302-6309.
[15] Martha Pulido, Patricia Melin, Oscar Castillo, "Particle swarm optimization of ensemble neural networks with fuzzy aggregation for time series prediction of the Mexican Stock Exchange", Information Sciences(2014) 188-204.
[16] Jigar Patel, Sahil Shah, Priyank Thakkar, K Kotecha, "Predicting stock market index using fusion of machine learning techniques", Expert Systems with Applications(2015) 2162-2172.
[17] Ludmila Dymova, Pavel Sevastjanov, Krzysztof Kaczmarek, "A Forex trading expert system based on a new approach to the rule-base evidential reasoning", Expert Systems with Applications(2016) 1-13.
[18] Investitors Exchange, Investitors Exchange API website, https://iextrading.com/developer/docs .
[19] Jersey, Jersey - RESTful Web Services in Java website, https://jersey.github.io .
[20] Trevor Hastie, Robert Tibshirani, Jerome Friedman, "The Elements of Statistical Learning", Springer, 2009.
[21] RapidMiner, RapidMiner operators' documentation website, https://docs.rapidminer.com/latest/studio/operators .
[22] Leo Breiman, "Random Forests", (2001) .