Machine Learning nella Selezione di Titoli Azionari

Modelli di Machine Learning per la selezione di titoli azionari da allocare in portafoglio.

di Emilio De Renzis, anno accademico 2019/2020

Abstract

Il lavoro di ricerca svolto propone un nuovo approccio per la selezione di azioni da allocare in portafoglio. Partendo dai modelli presenti nella letteratura finanziaria si punta a stimare il movimento dei prezzi azionari implementando modelli di Machine Learning su dati finanziari. Il dataset analizzato è composto da dati estratti dai bilanci pubblici riguardanti circa 5000 aziende quotate nei mercati NYSE, Nasdaq e titoli scambiati Over The Counter (OTC).

Saranno analizzati 5 modelli: Logistic Regression, Polynomial Feature, Decision Tree, Random Forest Classifier e Artificial Neural Network. Mediante la combinazione dei modelli più performanti verranno selezionate le azioni da inserire nel portafoglio titoli. I rendimenti ottenuti saranno così confrontati con quelli ottenibili tramite l'utilizzo dei metodi già presenti in letteratura.

Si trarranno infine considerazioni riguardo i limiti dei metodi esposti e possibili lavori futuri.

Introduzione

Prevedere l'andamento dei corsi azionari è un compito molto complesso a causa delle numerose variabili che influenzano il valore di mercato di un titolo. Ottenere stime puntuali permetterebbe di costruire portafogli ottimi in ottica prezzo-rendimento.

Esistono vaste letterature che si concentrano sulle strategie per la selezione di titoli da allocare in portafoglio. I due approcci maggiormente utilizzati sono l'analisi fondamentale e l'analisi tecnica. L'analisi fondamentale ha lo scopo di determinare il valore intrinseco di strumenti finanziari attraverso una attenta valutazione delle metriche fondamentali, nonché utili, rischio, tasso di crescita e posizione competitiva. L'analisi fondamentale può quindi essere utilizzata dagli investitori per identificare i titoli sottovalutati o sopravalutati al fine di compiere scelte di investimento.

L'analisi tecnica ha come obiettivo quello di identificare i titoli *mispriced* (sottovalutati o sopravalutati) sul mercato attraverso un attento esame dei prezzi passati e correnti.

Un importante lavoro riguardante l'applicazione di analisi fondamentale è riconducibile a Ou & Penman (1989) che utilizzano l'analisi di bilancio per ottenere

una misura sintetica in grado di cogliere le future variazioni degli utili e dunque dei rendimenti azionari futuri.

Ou e Penman improntano il loro studio sulla ricerca di indicatori in grado di classificare le variazioni degli utili futuri (variazioni positive e variazioni negative) ed analizzano la possibilità di battere il mercato investendo in titoli selezionati tramite questi indicatori.

Partendo dalla strategia di Ou & Penman, Holthausen & Larcker intuiscono la possibilità di apportare un miglioramento alla strategia focalizzando l'attenzione sugli abnormal return (AR) anziché sulla variazione dell'utile Δeps.

Ciò che differenzia le due strategie è cosa prevedere attraverso il modello; Ou & Penman perseguono il fine di prevedere la variazione degli utili, mentre Holthausen & Larcker mirano a predire la variazione dei rendimenti partendo dai medesimi indicatori di bilancio.

Lo scopo dell'elaborato è quello di esaminare la possibilità di migliorare l'accuracy ottenuta da Ou and Penman implementando modelli di Machine Learning. Piuttosto che predire la variazione degli utili o la variazione degli AR si punterà a stimare il movimento dei prezzi azionari.

Il dataset utilizzato per la costruzione del modello è composto da dati finanziari estratti dai bilanci pubblici riguardanti circa 5000 aziende quotate nei mercati NYSE, Nasdaq GS, Nasdaq GM, Nasdaq CM e azioni scambiate OTC.

I dati riferiti agli anni compresi tra il 2017 e il 2019 riguardano misure di redditività, indici di rotazione, indici di leva, ibridi ottenuti combinando valori di bilancio e prezzi di mercato, così come proposto nella trattazione di Ou & Penman.

Questi indicatori sono utilizzati per predire se il prezzo delle azioni aumenterà negli anni successivi in modo da costruire un portafoglio titoli e misurarne la performance.

I dati saranno analizzati partendo da un modello di regressione logistica. Combinando tra loro le variabili indipendenti (indicatori di bilancio) verrà costruito un modello polinomiale. Aumentando il numero dei predittori, crescerà il numero dei parametri da stimare. Per evitare che le stime dei coefficienti rispecchino troppo i dati di training perdendo di generalità (overfitting), verranno applicate delle regolarizzazioni ai coefficienti delle variabili indipendenti.

Si procederà in seguito all'implementazione di algoritmi non parametrici come il Decision Tree e il Random Forest; tali algoritmi non richiedono particolari assunzioni sulla forma dei dati in input, al contrario dei modelli logistici, sono di facile interpretazione. Inoltre, hanno il vantaggio di selezionare quali tra le variabili indipendenti hanno una significatività al fine di una corretta classificazione.

Infine, verrà costruita un Artificial Neural Network per tracciare meglio le possibili non linearità dei dati.

Una volta implementati, i vari modelli saranno confrontati mediante l'accuracy che consiste in una misura di bontà del modello. Ciò permetterà di selezionare il

modello più performante in termini di capacità di classificare i titoli in ribassisti e rialzisti. Tale modello sarà utilizzato per implementare la stock selection di un portafoglio titoli.

Nel primo paragrafo verrà analizzato il Dataset utilizzato per stimare e testare i parametri degli algoritmi di Machine Learning di cui al secondo paragrafo. Verrà successivamente proposta una strategia di investimento basata sui risultati ottenuti dai modelli. Nell'ultima parte si procederà ad un confronto tra i risultati dei modelli proposti e i risultati ottenuti mediante modelli già presenti in letteratura, inoltre verrà fatta menzione di spunti per futuri lavori.

Il Dataset

Il dataset analizzato nel presente elaborato è composto da dati finanziari estratti dai bilanci pubblici riguardanti circa 5000 aziende quotate nei mercati NYSE, Nasdaq GM, Nasdaq GS, Nasdaq CM e altri titoli OTC (Over The Counter).

I dati riferiti agli anni compresi tra il 2017 e il 2019 riguardano misure di redditività, indici di rotazione, indici di leva, ibridi ottenuti combinando valori di bilancio e prezzi di mercato, così come proposto nella trattazione di Ou & Penman.

Nel presente elaborato sono presenti 224 indicatori finanziari come mostra la *Figura 7 (è riportato unicamente un estratto di 17 indicatori di bilancio utilizzati).* Nella colonna *Unnamed* sono riportati i Tickers azionari.

Unnamed:	Revenue	Revenue Growth	Cost of Revenue	Gross Profit	R&D Expenses	SG&A Expense	Operating Expenses	Operating Income	Interest Expense
A	4.472e+09	0.0643	2.063e+09	2.409e+09	3.39e+08	1.229e+09	1.568e+09	8.41e+08	7.9e+07
AA	1.1652e+10	0.2505	8.991e+09	2.661e+09	3.2e+07	2.8e+08	1.398e+09	1.263e+09	1.04e+08
AABA	7.6e+07	-0.9853	nan	5.446e+07	nan	nan	4.7809e+07	1.51e+10	-7.0041e+07
AAL	4.2622e+10	0.0618	1.7636e+10	2.4986e+10	0	1.3431e+10	2.0755e+10	4.231e+09	1.053e+09
AAMC	1.816e+07	-0.0916	0	1.816e+07	0	2.5507e+07	2.5507e+07	-7.347e+06	0
AAME	1.81114e+08	0.0905	1.17515e+08	6.3599e+07	0	4.3446e+07	5.652e+07	7.079e+06	1.723e+06
AAN	3.38371e+09	0.0549	2.58934e+08	3.12477e+09	0	1.40398e+09	2.87008e+09	2.54699e+08	2.0538e+07
AAOI	3.82329e+08	0.4665	2.16049e+08	1.6628e+08	3.5365e+07	4.3964e+07	7.9329e+07	8.6951e+07	858000
AAON	4.05232e+08	0.0554	2.81835e+08	1.23397e+08	0	4.9249e+07	4.9294e+07	7.4103e+07	0
AAP	9.37378e+09	-0.0203	5.28874e+09	4.08505e+09	0	3.51484e+09	3.51484e+09	5.70212e+08	5.8801e+07
AAPL	2.29234e+11	0.063	1.41048e+11	8.8186e+10	1.1581e+10	1.5261e+10	2.6842e+10	6.1344e+10	0
AAT	3.14983e+08	0.0674	1.16677e+08	1.98306e+08	0	2.1382e+07	1.0466e+08	9.3646e+07	5.3848e+07
AAU	0	0	0	0	0	1.21818e+06	4.47741e+06	-4.47741e+06	0
AAWW	2.15646e+09	0.1722	9.73772e+08	1.18269e+09	0	6.00774e+08	9.40714e+08	2.41974e+08	9.9687e+07
AAXN	3.43798e+08	0.2817	1.3671e+08	2.07088e+08	5.5373e+07	1.38692e+08	1.94065e+08	1.3023e+07	0
AB	0	0	0	0	0	0	0	0	0
ABB	2.5196e+10	0.0107	1.735e+10	7.846e+09	1.013e+09	4.765e+09	5.778e+09	2.068e+09	2.34e+08
ABBV	2.8216e+10	0.1006	7.042e+09	2.1174e+10	5.007e+09	6.295e+09	1.1629e+10	9.545e+09	1.004e+09
ABC	1.53144e+11	0.0429	1.48598e+11	4.546e+09	0	2.12873e+09	3.48566e+09	1.06034e+09	1.45185e+08
ABCB	3.56218e+08	0.1094	0	3.56218e+08	0	2.24926e+08	2.31936e+08	1.24282e+08	0

Unnamed:	Asset Growth	Book Value per Share Growth	Debt Growth	R&D Expense Growth	SG&A Expenses Growth	Sector	2018 PRICE VAR [%]	Class
A	0.0811	0.1527	0.0562	0.0304	-0.0192	Healthcare	0.717881	1
AA	0.0422	-0.2093	-0.0284	-0.0303	-0.2135	Basic Materials	-51.8216	0
AABA	0.6888	-0.2466	0.0539	nan	nan	Financial Services	-20.7712	0
AAL	0.0295	-1.2231	0.0296	0	0.0928	Industrials	-38.8179	0
AAMC	-0.0815	-0.1426	0	0	0.0491	Financial Services	-63.6152	0
AAME	0.0773	0.0697	0	0	-0.0302	Financial Services	-25.3898	0
AAN	0.0293	0.1771	-0.2592	0	0.0386	Industrials	6.82513	1
AAOI	0.4053	0.3201	0.1481	0.1128	0.3673	Technology	-59.2983	0
AAON	0.1569	0.1599	0	0	0.279	Basic Materials	-4.2392	0
AAP	0.0201	0.1666	0.0013	0	0.0134	Consumer Cyclical	48.6722	1
AAPL	0.1667	0.096	0.3292	0.1529	0.0752	Technology	-7.05434	0
AAT	0.1374	-0.0006	0.2482	0	0.1947	Real Estate	8.37083	1
AAU	0.4844	0.3148	0	-1	0.1513	Basic Materials	-34.9515	0
AAWW	0.1667	0.161	0.2029	0	0.0882	Industrials	-27.1959	0
AAXN	0.2155	0.1086	0	0.809	0.2833	Industrials	64.7834	1
AB	0.0027	0.0151	0	0	0	Financial Services	20.3233	1
ABB	0.1086	0.1131	0.0889	0.0476	0.0514	Technology	-27.0601	0
ABBV	0.0709	0.1176	0.0143	0.1418	0.0704	Healthcare	-2.66743	0

Figura 1 – Indicatori Finanziari utilizzati come variabili indipendenti nei modelli. Sulle righe troviamo vengono illustrati i titoli azionari, nelle colonne sono riportati i predittori.

Segue la scomposizione del dataset in due parti:

1. Train Stock;
2. Test Stock.

Il dataset *Train Stock* contiene 224 indicatori finanziari di 4960 società quotate nel mercato americano riferite al periodo 2017. Questi indicatori sono utilizzati per predire se il prezzo delle azioni a cui fanno riferimento è incrementato nell'anno 2018. La colonna "*Class"* del dataset registra gli andamenti di prezzo tra il 2017 e il 2018:

- se $P_{2018} - P_{2017} > 0$ la colonna *Class* assume valore 1;
- se $P_{2018} - P_{2017} < 0$ la colonna *Class* assume valore 0;

Il dataset *Test Stock* contiene i medesimi 200+ indicatori finanziari su 4960 azioni listate nel mercato americano. Gli indicatori fanno riferimento al 2018 e sono usati

per predire se il prezzo delle companies nel dataset è incrementato durante il 2019.

Per ottimizzare le prestazioni dei modelli di machine learning si sono rese necessarie diverse operazioni di *cleaning* del dataset, tra cui il controllo di composizione del dataset, il numero di zeri e di valori nulli nelle colonne e nelle diverse righe.

Come mostrato nella *Figura 8,* le colonne[1]: "Cash Conversion Cycle", "Operating Cycle" e "Short Term Coverage Ratio" presentano una percentuale di valori nulli (NaN) maggiore del 50%, ciò comporta la decisione di eliminare le suddette colonne dalla trattazione.

Index	0
cashConversionCycle	99.8992
operatingCycle	99.8992
shortTermCoverageRatios	50.8065

Figura 2 – Variabili indipendenti con valori nulli (NaN) per più del 50%.

In seguito, risulta necessario computare il numero di valori uguali a zero in ogni colonna.

Index	# of 0	% of 0
Net Income - Discontinued ops	3681	0.742137
Preferred Dividends	3637	0.733266
Class	3590	0.72379
Deposit Liabilities	3490	0.703629
Net Income - Non-Controlling int	2779	0.560282
R&D to Revenue	2764	0.557258
Deferred revenue	2749	0.554234
Short-term investments	2669	0.538105
Dividends per Share Growth	2629	0.53004
R&D Expenses	2602	0.524597
R&D Expense Growth	2588	0.521774

Figura 3 – Numeri di zero nel dataset.

[1] Nelle diverse colonne sono disposti gli indicatori di bilancio.

La *Figura 9* mostra la presenza di 10 predittori (sotto la voce index) con valori uguali a zero per almeno il 50%. La colonna "# of 0" riporta il numero di zeri.

Successivamente si procede alla eliminazione delle colonne:

- "Deposit Liabilities", presenta il 70% di valori uguali a zero;
- "Net Income – Discontinued ops", per l'inattendibilità di una differenza uguale a zero;
- "Net Income – Non Controlling Interest", per l'inattendibilità di una differenza uguale a zero;
- "R&D to Revenue", per l'inattendibilità di un rapporto uguale a zero.

Le colonne che seguono sono state trasformate in variabili dummy (con valori uguali a 0 se $x_i = 0$; 1 se $x_i \neq 0$):

- "Preferred Dividends"
- "Deferred revenue"
- "Short-term investments"
- "Dividends per Share Growth"
- "R&D Expenses"
- "R&D Expense Growth"

Unnamed:	Preferred Dividends	Deferred revenue	Short-term investments	Dividends per Share Growth	R&D Expenses	R&D Expense Growth
A	0	1	0	1	1	1
AA	0	0	0	0	1	1
AABA	0	1	1	0	0	0
AAL	0	0	1	0	0	0
AAMC	1	0	1	0	0	0
AAME	1	0	0	0	0	0
AAN	0	1	1	1	0	0
AAOI	0	0	1	0	1	1
AAON	0	1	1	1	0	0
AAP	0	0	0	0	0	0
AAPL	0	1	1	1	1	1
AAT	0	0	0	1	0	0
AAU	0	0	0	0	0	1
AAWW	0	0	1	0	0	0
AAXN	0	1	1	0	1	1
AB	0	0	0	1	0	0
ABB	0	0	1	1	1	1
ABBV	0	0	1	1	1	1
ABC	0	0	0	1	0	0
ABCB	0	0	0	1	0	0

Figura 4 – Trasformazione da variabili continue a variabili dummies.

Inoltre, la colonna "Sectors" composta da:

- Consumer Defensive;
- Basic Materials;
- Healthcare;
- Consumer Cyclical;
- Industrials;
- Real Estate;

- Communication Services;
- Energy;
- Financial Services;
- Utilities;
- Technology;

è stata trasformata in variabili dummies come mostra la *Figura 12*.

Unnamed:	Basic Materials	Communication Services	Consumer Cyclical	Consumer Defensive	Energy	Financial Services	Healthcare	Industrials	Real Estate	Technology	Utilities
A	0	0	0	0	0	0	1	0	0	0	0
AA	1	0	0	0	0	0	0	0	0	0	0
AABA	0	0	0	0	0	1	0	0	0	0	0
AAL	0	0	0	0	0	0	0	1	0	0	0
AAMC	0	0	0	0	0	1	0	0	0	0	0
AAME	0	0	0	0	0	1	0	0	0	0	0
AAN	0	0	0	0	0	0	0	1	0	0	0
AAOI	0	0	0	0	0	0	0	0	0	1	0
AAON	1	0	0	0	0	0	0	0	0	0	0
AAP	0	0	1	0	0	0	0	0	0	0	0
AAPL	0	0	0	0	0	0	0	0	0	1	0
AAT	0	0	0	0	0	0	0	0	1	0	0
AAU	1	0	0	0	0	0	0	0	0	0	0
AAWW	0	0	0	0	0	0	0	1	0	0	0
AAXN	0	0	0	0	0	0	0	1	0	0	0
AB	0	0	0	0	0	1	0	0	0	0	0
ABB	0	0	0	0	0	0	0	0	0	1	0
ABBV	0	0	0	0	0	0	1	0	0	0	0
ABC	0	0	0	0	0	0	1	0	0	0	0

Figura 6 – Creazione di variabili Dummies per i vari settori.

Il passaggio successivo è stato quello di computare il numero di valori nulli in ogni riga[2]; come mostra la *Figura 13*, vi sono diversi titoli azionari con più del 50% di valori nulli (NaN). In totale sono stati eliminati dal dataset 830 titoli azionari.

[2] Nelle diverse righe sono disposti i vari titoli azionari con i relativi Tickers.

Index	Tickers	#NaN	% of NaN
3655	PFO	214	0.986175
3801	FT	214	0.986175
3786	BYM	214	0.986175
3788	CUBA	214	0.986175
3790	DHF	214	0.986175
3792	EGF	214	0.986175
3796	FCT	214	0.986175
3797	FLC	214	0.986175
3802	GDL	214	0.986175
3739	LGI	214	0.986175
3803	GGM	214	0.986175
3804	GGT	214	0.986175
3805	GLQ	214	0.986175
3809	HTD	214	0.986175
3812	JPT	214	0.986175
3816	MHI	214	0.986175
3785	BNY	214	0.986175
3784	BKT	214	0.986175
3783	BFY	214	0.986175
3779	AFT	214	0.986175
3766	NCA	214	0.986175

Index	Tickers	#NaN	% of NaN
2875	TDW	141	0.64977
1601	BATRA	141	0.64977
1525	LSXMA	136	0.626728
168	CHSCP	134	0.617512
298	BBL	131	0.603687
1715	UONEK	131	0.603687
4593	LFIN	130	0.599078
449	DRD	129	0.59447
1349	UA	129	0.59447
1380	CUK	129	0.59447
2428	ZG	127	0.585253
1663	RUSHB	127	0.585253
2590	AMX	126	0.580645
4827	BELFB	125	0.576037
4282	EDN	124	0.571429
2574	TRMT	123	0.56682
2990	CEF	122	0.562212
355	TX	121	0.557604
213	TS	121	0.557604
2330	DDR	120	0.552995
4323	MFGP	114	0.525346

Figura 7 – Numero di NaN e percentuale di valori nulli in ogni riga del Dataset. Le riche corrispondono ai vari titoli azionari.

Una criticità rilevata implementando i vari modelli di Machine Learning di seguito esposti, è stata quella di dover rimuovere i vari valori NaN dalle restanti colonne per un corretto funzionamento dei modelli.

La soluzione a tale problematica è stata quella di utilizzare la funzione *fillna.median* presente in Python al fine di sostituire i valori mancanti con valori mediani calcolati per settore (come mostra la *Figura 14*).

```
# Filling the NaN with the median grouping by sector
data_train = data_train.groupby("Sector").transform(lambda x: x.fillna(x.median()))
```

Figura 8 – Codice Python utilizzato per rimpiazzare i valori nulli presenti nel dataset con le medie dei valori delle variabili calcolate per settore.

In sostanza, per ogni valore NaN x_{ijk} nelle righe quel valore è stato sostituito con il valore $\overline{x_{ik}}$.

Dove x_{ijk} rappresenta l'$i-esimo$ indicatore di bilancio del titolo j appartenente al settore k. Mentre $\overline{x_{ik}}$ rappresenta il valore mediano dell'$i-esimo$ indicatore di bilancio per i titoli del settore k.

Infine, da una ispezione grafica del dataset sono state rimosse le colonne:

- *"Operating Profit Margin",* presenza di valori anomali;
- *"2018 Price Variation %",* rappresentante la variazione di prezzo nell'anno 2018[3];

Variabile dipendente e variabili indipendenti

Una volta eseguite le opportune modifiche il dataset Train Stock presenta 224 indicatori finanziari e 4130 titoli azionari. Di cui 206 variabili di tipo continue e

18 variabili dummies:

- 'Class'
- 'Preferred Dividends'
- 'Deferred revenue'
- 'Short-term investments'
- 'Dividends per Share Growth'
- 'R&D Expenses'
- 'R&D Expense Growth'
- 'Consumer Defensive'
- 'Basic Materials'
- 'Healthcare'
- 'Consumer Cyclical'
- 'Industrials'
- 'Real Estate'
- 'Communication Services'
- 'Energy'
- 'Financial Services'
- 'Utilities'
- 'Technology'

I diversi algoritmi di Machine Learning hanno prestazioni migliori (convergono più velocemente) quando le caratteristiche delle variabili continue sono su scala relativamente simile e/o hanno una distribuzione vicina a quella di una normale.

Al fine di modificare le variabili in modo da ottenere una scala di valori simili, le variabili indipendenti sono state scalate utilizzando il metodo *StandardScaler*, il

[3] È interessante notare come i modelli logaritmici e i modelli base di machine learning non colgano l'importanza della colonna "2018 Price Variation %", infatti l'accuracy dei vari modelli non cambia di molto con la presenza di tale colonna. Mentre, implementando modelli più complessi l'accuracy della predizione sale al 99%. Questo accade perché tali modelli catturano la relazione non lineare che intercorre tra i valori nella colonna "2018 Price Variation %" e la classificazione del titolo come dummy 0,1.

quale standardizza le variabili sottraendo ad ogni valore la media e dividendo il risultato per la deviazione standard in modo da ottenere una varianza unitaria.

La distribuzione, dopo l'applicazione di *StandardScaler*, risulta avere una deviazione standard pari a 1 e circa il 68% dei valori è compreso tra -1 e 1.

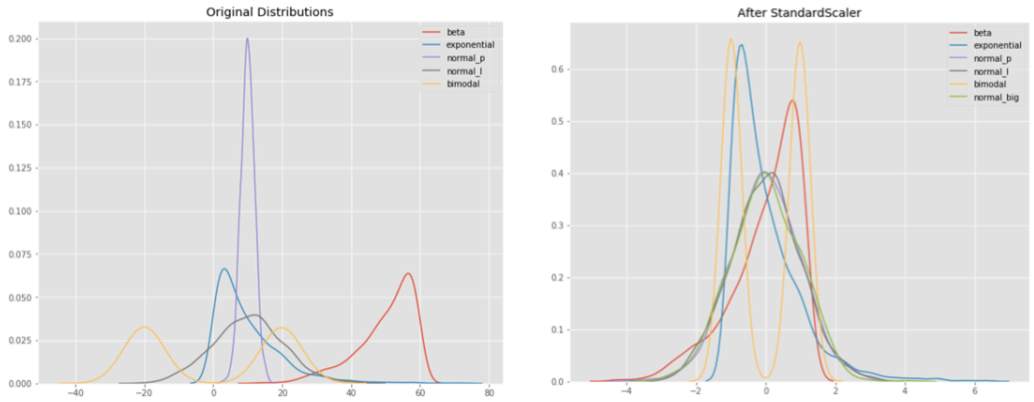


Figura 9 – Before and After StandardScaler.

Nella parte destra della *Figura 18* si nota come le distribuzioni abbiano una media vicina allo 0 e varianza unitaria. I valori sono su scala simile, ma in questo caso il range dei valori è maggiore rispetto a quello osservato dopo l'applicazione di *MinMaxScaler*.

Scalate le variabili continue, la variabile indipendente X è ora composta dalle 206 variabili scalate e dalle 17 variabili dummies.

La variabile dipendente è composta unicamente dalla colonna *"Class",* la quale registra gli andamenti di prezzo tra il 2017 e il 2018:

- se $P_{2018} - P_{2017} > 0$ la colonna *Class* assume valore 1;
- se $P_{2018} - P_{2017} < 0$ la colonna *Class* assume valore 0.

I modelli di Machine Learning

Il primo modello che verrà fittato nel presente paper è una regressione logistica così come proposto da Ou & Penman. Dal momento che si sta analizzando un problema di classificazione, quello che otterremo è un output compreso tra due valori 0,1 (motivo per il quale una regressione lineare OLS non rappresenta un modello utilizzabile).

Partendo da un modello di regressione logistica semplice verrà aggiunta flessibilità al modello consentendo la non linearità delle caratteristiche.

Verranno analizzate le performance di un modello polinomiale aggiungendo iterazioni tra le variabili per analizzare il grado di non-linearità presente nei dati. Incorporando gradi e interazioni tra le variabili aumenta il numero dei predittori rispetto al numero delle osservazioni, saranno quindi applicate alcune regolarizzazioni ai Betas della regressione logistica: L2 (Ridge Regularization) e L1 (Lasso Regularization), in modo da ridurre l'overfitting del modello.

Gli Hyperparameters da calibrare per quanto riguarda le regolarizzazioni di Ridge e Lasso saranno:

- Il valore dei parametri “penalty” da applicare al quadrato/valore assoluto dei Betas.

Potendo ragionevolmente affermare che la non linearità gioca un ruolo, si passerà ad un modello non parametrico, come la Random Forest.

Gli Hyperparameters che andranno calibrati nel caso del Random Forest Classifier, sono:

- la lunghezza dei diversi trees;
- il minimo numero di osservazioni nei final leaves;
- il numero minimo di osservazioni necessarie per procedere allo splitting;
- il criterio utilizzato per lo splitting.

Infine, verrà costruita un Artificial Neural Network, per tracciare meglio le possibili non linearità dei dati. Un importante problema sarà quello dell’overfitting del modello che sarà fronteggiato aggiungendo livelli di dropout, modificando il tasso di apprendimento e inserendo vincoli sui pesi.

Gli Hyperparameters che andranno calibrati nel caso del Artificial Neural Network, sono:

- dropout rate;
- vincoli sui pesi;
- learning rate;
- batch size;
- epoch.

Ou & Penman nella loro trattazione per la stima dei coefficienti suddividono il loro dataset in un periodo di Train (durante il quale hanno stimato i valori) e in un periodo di Test (durante il quale hanno validato i valori).

Nel presente lavoro di tesi la stima dei parametri è stata ottenuta sul dataset Train tramite *Cross Validation*.

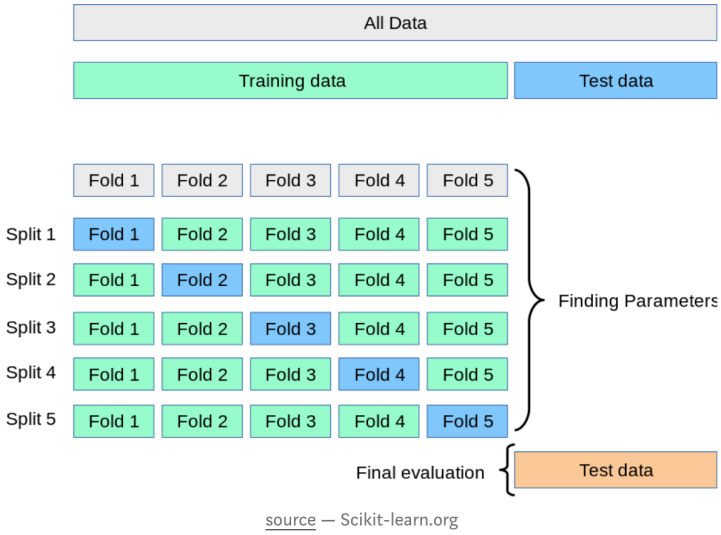


Figura 10 – Cross Validation.

La Cross Validation divide il campione in K gruppi, e per $j = 1 \dots K$:

- realizza il train del modello su tutti i gruppi eccetto il gruppo j;
- misura la qualità del modello $V[j]$ sul gruppo j.

Infine, la performance del modello $\bar{V}$ sarà una media delle $V[j]$ con $j = 1 \dots K$.

Nella *Figura 19* troviamo un esempio di *Cross Validation* con CV=5. Il dataset viene così suddiviso in un dataset di train (in verde) e in un dataset di test (in arancione).

La stima dei parametri avviene suddividendo il dataset di train in 5 sottogruppi. In ognuno dei 5 split, quattro gruppi (in verde) sono utilizzati per stimare i parametri e il gruppo rimanente (in blu) è utilizzato per la validazione. La qualità del modello è ottenuta come media dei risultati del modello sui sottogruppi sui quali è effettuata la validazione.

Nel presente elaborato per la valutazione dei metodi di classificazione si opta per l'utilizzo delle seguenti misure:

- ***Sensitivity****:* è il rapporto tra *true positive* (osservazioni correttamente identificate nella classe 1) e *total number of positive* (numero totale di osservazioni appartenenti alla classe 1 nel dataset);
- ***Specificity****:* è il rapporto tra *true negative* (osservazioni correttamente identificate nella classe 1) e *total number of negative* (numero totale di osservazioni appartenenti alla classe 1 nel dataset);
- ***Positive Predictive Value****:* è il rapporto tra il numero di *true positive* e la somma del numero di *true positive* e di *false positive;*
- ***Negative Predictive Value****:* è il rapporto tra il numero dei *true negative* e la somma del numero dei *true negative* e di *false negative;*
- ***Classification Accuracy****:* è il rapporto tra il numero dei casi correttamente identificati sul totale dei casi.

Nella *Figura 20* è riportato il funzionamento della Confusion Matrix, consistente in un metodo di valutazione dei modelli nel caso di problemi di classificazione.

Confusion Matrix:

	Predicted Class = 0	Predicted Class = 1
Actual Class = 0	True Negatives (TN)	False Positives (FP)
Actual Class = 1	False Negatives (FN)	True Positives (TP)

N = number of observations

Overall accuracy = (TN + TP)/N

Overall error rate = (FP + FN)/N

Sensitivity = TP/(TP + FN)

False Negative Error Rate = FN/(TP + FN)

Specificity = TN/(TN + FP)

False Positive Error Rate = FP/(TN + FP)

Figura 11 – Confusion Matrix

Regressione Logistica

La regressione logistica è uno dei più frequenti algoritmi utilizzati per risolvere problemi di classificazione. È un metodo per predire una variabile dipendente dato un set di variabili indipendenti, dove la variabile dipendente è di tipo categorica.

Un problema ricorrente nell'applicazione dei metodi di Machine Learning è quello dell'overfitting, l'algoritmo si adatta (fitting) perfettamente (over) ai dati di training perdendo di generalità; il rischio che ne consegue è quello di costruire un modello perfetto *in sample*, ma errato *out of sample.*

Per fronteggiare la problematica dell'overfitting una soluzione è quella di aggiungere una penalizzazione ai parametri:

partendo dalla ***Loss Function*** del modello logistico

$$\min_{\beta_0,\dots,\beta_p} \sum_{i=1}^{n} -Y_i \left[\frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\dots+\beta_p x_p)}}\right] - (1-Y_i)\, log\left[1-\frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\dots+\beta_p x_p)}}\right]$$

viene aggiunto un termine di penalizzazione (Ridge)

$$\min_{\beta_0,\dots,\beta_p} L(\beta_0,\dots,\beta_p) + \boldsymbol{\lambda} \sum_{j=1}^{p} |\boldsymbol{\beta}_j|^2$$

La *Ridge Logistic Regression* (Hoerl and Kennard, 1970; Cessie and Houwelingen, 1992; Schaefer et al., 1984) è ottenuta massimizzando la funzione della massima verosimiglianza applicando una penalizzazione ad ogni coefficiente eccetto al coefficiente dell'intercetta. Lo stimatore *Ridge* riduce i coefficienti di regressione, in modo che le variabili, con un contributo minore al risultato, abbiano i loro coefficienti vicini allo zero. La Ridge Regression ha il limite di includere nella selezione del modello tutte le variabili indipendenti (James et al., 2013).

La stima dei parametri β_p dipende dalla scelta del parametro $\lambda \geq 0$. All'aumentare del parametro λ, i coefficienti β_p tendono a zero.

Nel presente elaborato questa metodologia è stata implementata tramite il linguaggio di programmazione *Python*, come mostra la *Figura 22.*

```
1   alphas = [0.0001,0.001,0.01,0.1,0.5,1,5]
2   cv = 10
3
4   grid = {"C":alphas}
5
6   log_reg = LogisticRegression(solver = 'saga',
7                                max_iter = 1000,
8                                random_state = 90,
9                                penalty = 'l2')
10
11  clf = GridSearchCV(log_reg,
12                     param_grid = grid,
13                     cv = cv,
14                     scoring=["accuracy", "neg_log_loss"],
15                     refit = "neg_log_loss",
16                     n_jobs = -1)
17  clf.fit(X, Y)
```

Figura 12 – Implementazione della Ridge Logistic Regression tramite Python.

Nella *riga 1* possiamo vedere come siano stati testati diversi valori per il parametro λ (alphas in Python): $[0.0001,0.001,0.01,0.1,0.5,1,5]$ così da incidere in modo diverso sui β_p della regressione.

Come mostra la *Figura 22 riga 2*, il valore della Cross Validation è stato posto uguale a 10. Nella *riga 6* vediamo l'implementazione della regressione logistica con *Ridge Penalization* (*riga 9*).

L'*accuracy* del modello sul dataset Train utilizzando una *Cross Validation* pari a 10 è del 69% con una standard deviation dello 0,031.

Potrebbe sembrare un valore poco superiore a quello ottenuto nella trattazione di Ou and Penman (67%), ma in realtà va precisato che il risultato ricavato è una media delle *accuracy* derivanti dalla *Cross Validation*. Inoltre, vanno sottolineati due importanti aspetti:

1. il risultato è stato ottenuto guardando a tutti i titoli presenti nel dataset, non unicamente i titoli con $Pr_j \geq 0.60\ e\ Pr_j < 0.40$;
2. nel presente elaborato si sta predicendo l'andamento del prezzo in $t + 1$, mentre nel trattato di Ou & Penman si guardava alla variazione dell'EPS.

Nella *Figura 23* viene riportata la *confusion matrix* per il caso della regressione logistica con penalizzazione L2 (*Ridge penalization*).

Il modello identifica correttamente il movimento a ribasso del prezzo per 1810 titoli azionari ed il movimento rialzista per 255 titoli azionari.

	0	1
0	1810	564
1	675	255

Figura 13 – Confusion Matrix, Ridge Logistic Regression

Il primo modello analizzato sembrerebbe in grado di cogliere la relazione che c'è tra indicatori di bilancio e ribasso dei prezzi, pertanto intercetta le criticità presenti nei bilanci aziendali che portano poi, negli anni successivi, ad una decrescita del prezzo.

Nel campione analizzato sono presenti 2374 titoli azionari per i quali nell'anno 2018 è stato osservato una decrescita dei prezzi e 930 titoli azionari per i quali è stato osservato un aumento dei prezzi, pertanto il dataset potrebbe risultare sbilanciato.

Regressione Logistica con penalizzazione – Lasso Model

Nella *Lasso Logistic Regression* (Park and Casella, 2008; Tibshirani, 1996). viene utilizzata una differente penalizzazione (L1) per la stima dei parametri β_p:

$$\min_{\beta_0,\dots,\beta_p} L(\beta_0, \dots, \beta_p) + \boldsymbol{\lambda} \sum_{j=1}^{p} |\boldsymbol{\beta_j}|$$

Nella *Ridge Regression*, il termine di penalizzazione è dato dalla somma dei quadrati dei coefficienti, mentre nella *Lasso Regression* la penalizzazione è uguale alla somma dei valori assoluti dei coefficienti.

Nella *Ridge Regression* i coefficienti non possono assumere valore zero, quindi hanno tutti impatto sulla classificazione a seconda dei pesi. Nel caso della *Lasso Regression*, invece, avviene sia una contrazione del valore dei parametri (per evirare l'overfitting) sia una selezione delle variabili, perché la penalizzazione azzera i coefficienti delle variabili collineari. Ciò comporta che il *Lasso* può anche essere visto come un'alternativa ai metodi di *features selection* per eseguire la selezione delle variabili al fine di ridurre la complessità del modello.

Nella *Figura 24* è riportato il codice utilizzato per l'implementazione della *Lasso Logistic Regression* in *Python*. Gli *alphas* utilizzati per la penalizzazione del modello sono uguali a quelli utilizzati nella *Ridge Logistic Regression* (riga uno), così come il valore della *Cross Validation* (riga 12).

```
1   alphas_lasso = [0.0001,0.001,0.01,0.1,0.5,1,5]
2   grid_lasso = {"C":alphas_lasso}
3
4
5   log_reg_lasso = LogisticRegression(solver = 'saga',
6                                      max_iter = 1000,
7                                      random_state = 90,
8                                      penalty = 'l1')
9
10  clf_lasso = GridSearchCV(log_reg_lasso,
11                           grid_lasso,
12                           cv = cv,
13                           scoring = ["accuracy","neg_log_loss"],
14                           refit = 'accuracy',
15                           n_jobs = -1)
16  clf_lasso.fit(X, Y)
```

Figura 14 – Implementazione della Lasso Logistic Regression tramite Python.

Il valore di λ (alpha) che garantisce l'accuracy più alta è $\lambda = 0{,}01$ (*Figura 25*). Il valore dell'accuracy è 0.62, molto inferiore a quello osservato utilizzando la *Ridge Regression* (0.69). La standard deviation dell'*accuracy* è dello 0.04 (0.03 nel modello precedente).

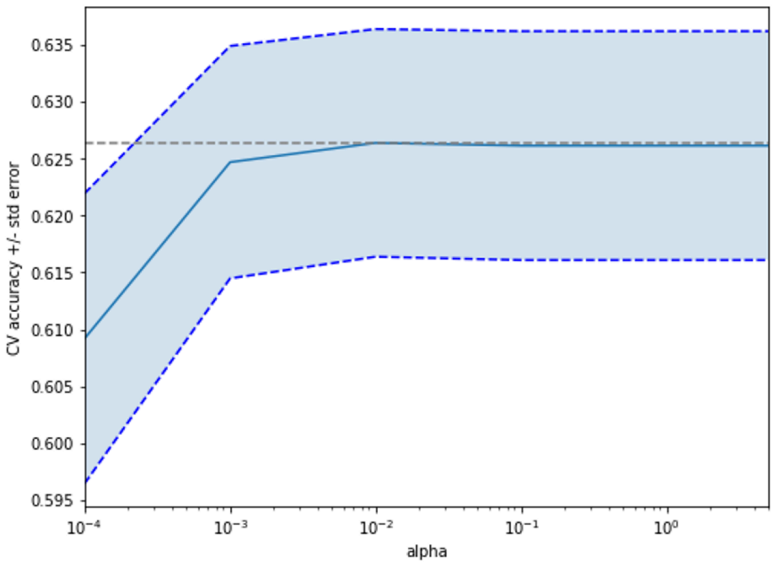


Figura 15 – Best alpha - Lasso Model.

Nella *Figura 26* è riportata la *Confusion Matrix*. La *Lasso Logistic Regression* identifica con maggior precisione le aziende che nel 2018 presentano una variazione negativa del prezzo, ossia 2363 aziende (contrapposte alle 1810

identificate con la Ridge. Di contro la Lasso Regression classifica erroneamente in "Class 0" 918 titoli, i quali vedono invece un incremento di prezzo nel 2018.

	0	1
0	2363	601
1	918	248

Figura 16 – Confusion Matrix, Lasso Logistic Regression

Regressione Logistica con penalizzazione – Elastic Net

Come analizzato nei precedenti paragrafi, l'utilizzo della *Lasso Regression* risulta conveniente quando i modelli contengono molte variabili e tra queste poche hanno una significatività statistica per la predizione del valore della Y, mentre la *Ridge Regression* risulta profittevole quando tutte le variabili nel modello hanno una certa significatività.

Quando un modello include molte variabili, può risultare difficile analizzare ogni singola variabile e trarre conclusioni generali su quale modello sia preferibile implementare.

Una soluzione intermedia alle due precedentemente analizzate è rappresentata dalla regressione *Elastic Net* (Zou, H., Hastie, T., 2005), la quale combina le proprietà delle regressioni *Ridge* e *Lasso* penalizzando il modello sia tramite la norma L1 che la norma L2:

$$\min_{\beta_0,\dots,\beta_p} L(\beta_0,\dots,\beta_p) + \boldsymbol{\lambda_L} \sum_{j=1}^{p} |\boldsymbol{\beta_j}| + \boldsymbol{\lambda_R} \sum_{j=1}^{p} |\boldsymbol{\beta_j}|^2$$

Come mostra l'equazione, L1 e L2 assumono diversi valori di λ. Tramite la *Cross Validation* è possibile stimare i migliori λ_L per la *Lasso penalization* e λ_R per la *Ridge penalization*.

Quando $\lambda_L = \lambda_R = 0$, quello che si otterrà è una regressione logistica senza penalizzazioni così come proposta da Ou & Penman.

Quando $\lambda_L = 0, \lambda_R > 0$ si otterrà una *Ridge Logistic Regression*.

Quando $\lambda_L > 0, \lambda_R = 0$ si otterrà una *Lasso Logistic Regression*.

Quando $\lambda_L > 0,\ \lambda_R > 0$ si otterrà un modello ibrido in grado di lavorare bene in situazioni di correlazione tra le variabili. Ciò è spiegato dal fatto che la *Lasso Penalization* tende a mantenere un parametro tra le variabili correlate ed eliminare le altre, mentre *Ridge Penalization* fa in modo che tutti i parametri correlati tendano a zero. L'*Elastic Net Regression* – combinando L1 e L2 – mantiene solo alcune variabili fra quelle correlate e penalizza i loro coefficienti in modo che assumano valori prossimi a zero.

```python
log = LogisticRegression(solver = 'saga', max_iter= 500, random_state = 90)
params = [{'penalty': ['elasticnet'], 'l1_ratio' : [0.000001,0.0001,0.001,0.01,0.1,0.5,1.5,2]}]

clf_enet = GridSearchCV(log, cv = 10,
                        param_grid = params,
                        scoring = ['accuracy','neg_log_loss'],
                        return_train_score = True,
                        refit = 'accuracy' )

clf_enet.fit(X, Y)
```

Figura 17 – Elastic Net in Python.

Nella *Figura 27* è riportato il codice utilizzato per implementare la regressione logistica con penalizzazione *Elastic Net* in *Python*. Come mostra la figura (*riga due*), i valori di λ_L dai quali estrarre il miglior modello sono λ_L : [0.000001, 0.0001, 0.001, 0.01, 0.1, 0.5, 1.5, 2], mentre per λ_R è stato selezionato il valore 0.1, ossia il valore che garantisce la convergenza nella *Ridge Regression.*

L'accuracy ottenuta tramite l'applicazione dell'Elastic Net è del 63%, con una standard deviation dello 0,033.

Guardando alla *Confusion Matrix* (*Figura 28*) dell'*Elastic Net Model* si rileva una maggiore precisione nel classificare correttamente i titoli azionari ribassisti: 2381 contro i 2363 del *Lasso Model* e i 1810 del *Ridge Model.*

	0	1
0	2381	583
1	918	248

Figura 18 – Confusion Matrix, Elastic Net

Polynomial Features

In alcuni casi le variabili indipendenti interagiscono con la variabile dipendente in modo non lineare. Per identificare e monitorare questa caratteristica si possono costruire nuove interazioni tra le variabili indipendenti e/o trasformare le variabili dipendenti (es. elevandole alla potenza) e verificare se il risultato del modello migliora (Novara C., 2015).

Si parte dalla relazione lineare tra la probabilità di osservare un dato evento e le variabili indipendenti:

$$\Pr(Y_i = 1 | X_i) = F(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)$$

Successivamente segue l'aggiunta di trasformazioni delle variabili al modello

$$\Pr(Y_i = 1 | X_i) = F(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_1^3 + \dots)$$

E di interazioni tra i termini,

$$\Pr(Y_i = 1 | X_i) = F(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_1^3 + \beta_5 x_1 x_2 + \beta_6 x_1^2 x_2 + \cdots)$$

La crescita del numero dei parametri (complessità del modello) comporta il rischio di realizzare un modello che lavora perfettamente *in sample*, ma che non può essere generalizzato (in quanto tenderebbe all'overfitting).

Nel caso della *Logistic regression,* costruire un modello prono all'overfitting equivale ad avere come *decision boundary* aree specifiche, ossia piccole regioni disconnesse.

Nella *Figura 30 riga 2* possiamo vedere come tramite la funzione *PolynimialFeatures* presente in *Python* sono state create interazioni e trasformazioni delle 223 variabili indipendenti ottenendo così una nuova variabile *X_trasformed*. Questa nuova variabile contiene 25200 colonne e 4130 righe.

```
1   from sklearn.preprocessing import PolynomialFeatures
2   poly = PolynomialFeatures(2)
3   X_transformed = poly.fit_transform(X)
4   X_transformed = pd.DataFrame(X_transformed)
```

Figura 19 – Creazione delle iterazioni tra le variabili indipendenti mediante linguaggio di programmazione Python.

5.2.1 Polynomial Features con penalizzazione – Ridge

```
1  clf_poly = GridSearchCV(log_reg, grid, cv = cv,
2                          scoring = ["accuracy","neg_log_loss"],
3                          refit = 'accuracy',
4                          n_jobs = -1)
5  clf_poly.fit(X_transformed, Y)
```

Figura 20 – Polynomial Features con Ridge Penalization in Python.

La nuova variabile (X_trasformed) è stata utilizzata nella funzione *log_reg* (creata nel ***paragrafo 5.1.1***), ottenendo una regressione logistica con penalizzazione l2 (Ridge Penalization).

L'accuracy ottenuta dal modello è pari al 72%. A primo impatto potrebbe sembrare un risultato particolarmente interessante, guardando però alla *Confusion Matrix* (*Figura 32*) la funzione classifica l'intero campione come Classe 0 ad eccezione di 7 titoli.

	0	1
0	2961	3
1	1162	4

Figura 21 – Confusion Matrix, Polynomial Features con penalizzazione Ridge.

Il valore dell' λ che garantisce la convergenza è pari allo 0.001. Una criticità rilevata riguarda il tempo di esecuzione dell'algoritmo: 18552 sec (circa 5 ore e 15 minuti).

Polynomial Features con penalizzazione – Lasso

```
1  clf_poly_lasso = GridSearchCV(log_reg_lasso, grid, cv = cv,
2                                scoring = ["accuracy","neg_log_loss"],
3                                refit = 'accuracy',
4                                n_jobs = -1)
5  clf_poly_lasso.fit(X_transformed, Y)
```

Figura 22 – Polynomial Features con Lasso Penalization in Python.

Un ulteriore tentativo è stato fatto applicando la Lasso penalization. Come possiamo vedere dalla *Figura 33 riga 1* la funzione *log_reg_lasso* creata nel ***paragrafo 5.1.2*** è stata applicata alle variabili indipendenti contenute nel data frame *X_trasformed* e alla variabile dipendente Y.

I risultati ottenuti sono molto vicini a quelli della Ridge Penalization. Infatti, volgendo lo sguardo alla confusion matrix di cui alla *Figura 34*, noteremo che i due algoritmi classificano in modo identico le osservazioni (di conseguenza il valore dell'accuracy è il medesimo in entrambi i casi). Nel caso della Lasso Penalization, però, il tempo di esecuzione cresce notevolmente: 25306 sec (circa 7 ore).

	0	1
0	2961	3
1	1162	4

Figura 23 – Confusion Matrix, Polynomial Features con Penalizzazione Lasso.

Decision Tree

La regressione logistica è in grado di classificare nel migliore dei modi le osservazioni quando le diverse classi sono ben definite. Ci sono casi – come quello analizzato nel presente elaborato – nei quali è meno agevole definire i confini decisionali con delle *decision boundaries lineari* (si è visto nei precedenti paragrafi come sia possibile aggiungere iterazioni tra le variabili per considerare la non-linearità).

In queste situazioni possono essere implementati modelli di Machine Learning non parametrici come il *Decision Tree Model*, poiché l'algoritmo lavora efficientemente con grandi e complessi dataset senza la necessità di ipotesi iniziali sui dati.

Questo metodo è utilizzato in molte discipline per la sua facile implementabilità, per l'interpretabilità e per la robustezza dei risultati anche nel caso di valori mancanti presenti nel dataset (Hastie TJ, Tibshirani RJ, Friedman JH, 2009).

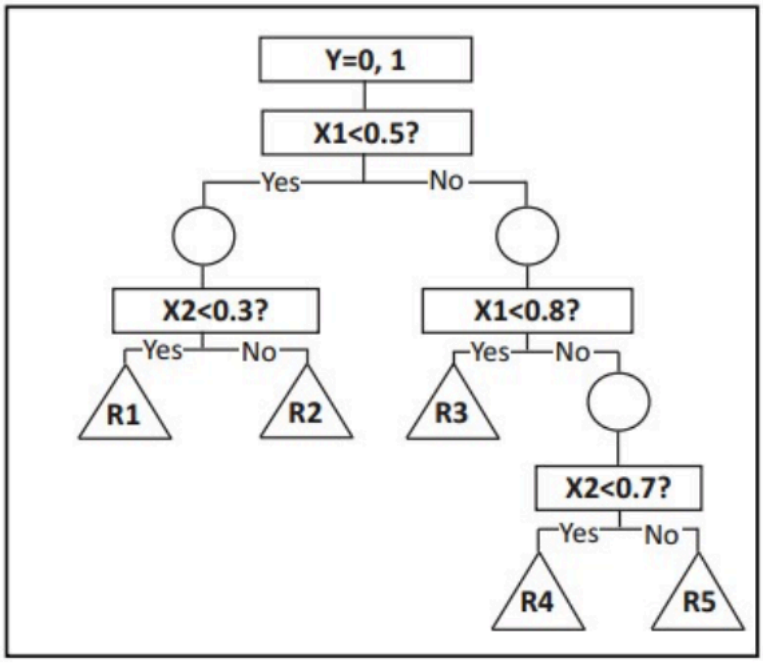


Figura 24 – Esempio di modello Decision Tree Classification.

La *Figura 35* illustra un semplice *Decision Tree model* che include una variabile binaria Y $(0,1)$ e due variabili continue x_1, x_2 che assumono valori compresi tra $[0,1]$.

Le componenti principali del modello sono i *nodi* e i *rami* dell'albero.

- $[x_1 < 0.5]$ rappresenta il *root node*, ossia la prima scelta che divide l'albero in due rami.
- $[x_1 < 0.8]$ rappresentano l'*internal node.*
- $[x_2 < 0.3]$ e $[x_2 < 0.7]$ rappresentano i *leaf nodes* o nodi finali, ossia il risultato di una combinazione di decisioni.

I collegamenti tra i vari nodi di un albero sono chiamati *rami.* Ogni percorso dal *root node* attraverso gli *internal node* fino al *leaf node* rappresenta una regola di classificazione.

La fase più complessa del modello rappresenta la metodologia adottata per la divisione dei vari rami tramite le variabili indipendenti x_i. Nella maggior parte dei casi non tutte le variabili indipendenti (indicatori di bilancio nel caso analizzato nel presente elaborato) sono utilizzate per la costruzione del modello ed in alcuni casi una specifica variabile è utilizzata più volte a diversi livelli dell'albero (come accade per la variabile x_2 nella *Figura 36*).

In ogni nodo è necessario scegliere il predittore (x_i) ottimale e il valore soglia ottimale per dividere il modello in più rami. Le scelte effettuate in questa fase hanno un forte impatto sulla forma delle regioni (le regioni illustrate nella Figura 37 sono il risultato delle scelte effettuate in termini di soglie e di variabili x_i).

La scelta del predittore (variabile indipendente x_i) da utilizzare per lo splitting dei nodi è effettuata attraverso il calcolo dell'*Entropia* e dell'*Information Gain*.

L'entropia è definita come la "misura del disordine" o misura di purezza. Matematicamente è indicata come,

$$E(S) = \sum_{i=1}^{c} - p_i log_2 p_i$$

Dove p_i rappresenta la probabilità di osservare un elemento appartenente alla $Classe\ i$ nel dataset.

Introdotta la misura di disordine, il passaggio successivo è quello di illustrare una metrica per misurare la riduzione del disordine all'arrivo di nuove informazioni (aggiunta di nuove variabili indipendenti). La metrica utilizzata è l'*Information Gain:*

$$IG(Y, X) = E(Y) - E(Y|X)$$

Si tratta semplicemente di sottrarre all'Entropia di Y, l'Entropia di Y derivante dalle informazioni apportate dalla variabile X. Maggiore è la riduzione, migliore risulterà lo splitting.

Esempio

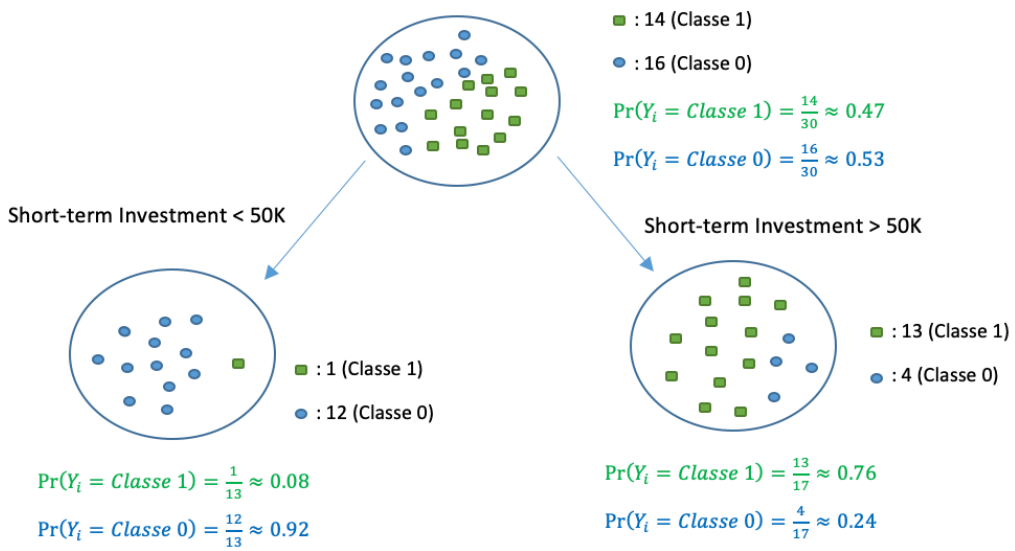
Si ipotizzi di avere un dataset composto da 30 titoli 14 dei quali appartenenti alla Classe 1 (prezzo in aumento nell'anno t+1) e 16 appartenenti alla Classe 0 (prezzo in diminuzione nell'anno t+1); e unicamente due variabili indipendenti $x_1\ e\ x_2$, dove:

$$x_1 = Short\ term\ Investment$$

$$x_2 = R\&D\ Expenses$$

Mediante il calcolo dell'Entropia e dell'Information Gain è possibile selezionare la variabile indipendente ed il valore soglia da utilizzare nel nodo.

<u>Feature 1</u>: *Short-term Investment*



Nella figura in alto viene illustrato uno splitting dell'albero attraverso la variabile $x_1(Short - term\ Investment)$ e con un valore soglia uguale a 50K. Il nodo a sinistra è composto da 13 osservazioni, 12 delle quali di Classe 0 e una di Classe 1. Il nodo a destra, invece, è composto da 17 osservazioni, 13 delle quali di Classe 1 e 4 appartenenti alla Classe 0.

Calcolando l'Entropia per i vari nodi otterremo:

- Entropia per il root node:

$$-\frac{16}{30}\log_2\left(\frac{16}{30}\right) - \frac{14}{30}\log_2\left(\frac{14}{30}\right) \approx 0.99$$

- Entropia per il nodo Short Term Investment < 50K:

$$-\frac{12}{13}\log_2\left(\frac{12}{13}\right) - \frac{1}{13}\log_2\left(\frac{1}{13}\right) \approx 0.39$$

- Entropia per il nodo Short Term Investment > 50K:

$$-\frac{4}{17}\log_2\left(\frac{4}{17}\right) - \frac{13}{17}\log_2\left(\frac{13}{17}\right) \approx 0.79$$

L'entropia pesata dei due leaf node è:

$$\begin{aligned} E(Balance) &= \frac{13}{30} \times 0.39 + \frac{17}{30} \times 0.79 \\ &= 0.62 \end{aligned}$$

Da qui l'Information Gain,

$$E(root\ node) - E(Balance) = 0.99 - 0.62 = 0.37$$

Quindi, dividendo l'albero tramite la variabile $Short - term\ Investment$ si ottiene un Information Gain dello 0.37.

Ripetendo i medesimi passaggi utilizzando la seconda variabile x_2 $(R\&D\ Expenses)$ si ottiene un Information Gain dello 0.13.

L'Information Gain ottenuto utilizzando x_1 è circa tre volte superiore a quello ottenuto utilizzando la variabile x_2, da qui la scelta di utilizzare x_2 come variabile per suddividere l'albero nel primo nodo.

L'algoritmo di Decision Tree utilizza i risultati ottenuti calcolando l'Information Gain per effettuare le suddivisioni dei nodi. Questo processo è utilizzato ad ogni suddivisione.

In uno scenario con più di due variabili indipendenti (come nel caso analizzato nel presente trattato), la prima suddivisione viene effettuata attraverso la caratteristica con l'IG minore. Ad ogni successivo nodo verrà ricalcolato l'Information Gain per ciascuna delle variabili indipendenti in modo da selezionare la nuova caratteristica da utilizzare nella divisione dell'albero.

L'algoritmo Decision Tree ripete questo processo fin quando le variabili aggiuntive non apportano nessuna riduzione dell'Entropia. Questo procedimento conduce ad una accuracy del 100%, ossia una perfetta classificazione delle osservazioni in Classe 1 e Classe 0.

Ottenere un valore del 100% di accuracy è un chiaro caso di overfitting. Quindi, un modello poco generalizzabile, poiché ogni minima variazione del dataset iniziale causerebbe errori nella classificazione finale e un incremento della varianza.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
dt = DecisionTreeClassifier(random_state = 90)
dt.fit(X, Y)
```

Figura 25 – Decision Tree Classifier Model in Python.

Implementando un simile modello in Python (*Figura 38*) otterremo la seguente *confusion matrix*:

	0	1
0	2964	0
1	0	1166

Figura 26 – Confusion Matrix, Decision Tree Classifier.

Notiamo che l'algoritmo ha classificato perfettamente i 4130 titoli azionari. Come vedremo nel capitolo finale, applicando questo modello a nuovi dati i risultati che otterremo saranno completamente errati.

Al fine di prevenire l'overfitting del modello, una strada percorribile è quella di stabilire una *stopping condition* (condizione di arresto). Perciò, riducendo la lunghezza di un albero (tree depth) aumenterà il bias (errore di classificazione),

ma si otterrà una minore varianza tra i risultati del modello *in sample* e quelli *out of sample.*

La lunghezza (depth) ottima di un modello ad albero può essere determinata valutando l'algoritmo su diverse *depth* tramite l'utilizzo della *Cross-Validation.* Otterremo così un modello con una minor accuracy sul dataset Train ma con una minor varianza dei risultati.

```
1   dt = DecisionTreeClassifier(criterion = 'gini', random_state = 90)
2   parameters = {'max_depth': range(5,15),
3                 'min_samples_leaf': range(5,15),
4                 'min_samples_split': range(5,15)}
5
6   clf_dt = GridSearchCV(dt, parameters, cv = 10,
7                         iid = True, scoring = 'accuracy',
8                         refit = 'accuracy',
9                         return_train_score = True,
10                        n_jobs=-1)
11  clf_dt.fit(X, Y)
```

Figura 27 – Decision Tree Classifier con Cross Validation in Python.

Nella *Figura 39* è illustrata l'implementazione dell'algoritmo *DecisionTreeClassifier* tramite *Cross Validation:*

- *riga 2*, il valore della lunghezza massima (depth) è ricercato nell'intervallo [5,15];
- *riga 3,* il minimo nodi finali è ricercato nell'intervallo [5,15];
- *riga 4,* il minimo di rami ricercato nell'intervallo [5,15];

Come mostra la *Figura 40* il modello ottenuto con la *Cross Validation* risulta meno prono all'overfitting. L'algoritmo identifica correttamente 2786 titoli rialzisti e 432 titoli ribassisti con una accuracy del 78%.

	0	1
0	2786	178
1	734	432

Figura 28 – Confusion Matrix, Decision Tree Classifier con Cross Validation.

La *Figura 41* mostra che gli Hyperparameters utilizzati per ottenere una accuracy del 78% sono:

- max depth: 7;
- min sample leaf: 14;
- min sample split: 5.

```
tuned hpyerparameters :(best parameters)  {'max_depth': 7,
'min_samples_leaf': 14, 'min_samples_split': 5}
```

Figura 29 – Best Hyperparameters ottenuti tramite Cross Validation.

Random Forest

Il modello di classificazione *Random Forest Classifier* (Breiman, 2001) consiste in un elevato numero di algoritmi di *Decision Tree* che operano come se fossero un unico modello (*ensemble method*). Ogni singolo *Tree* del *Random Forest Classifier* fornisce come output una data classe (nel nostro caso *Class 0* o *Class 1*). La classe con il maggior numero di voti viene adottata come previsione del modello.

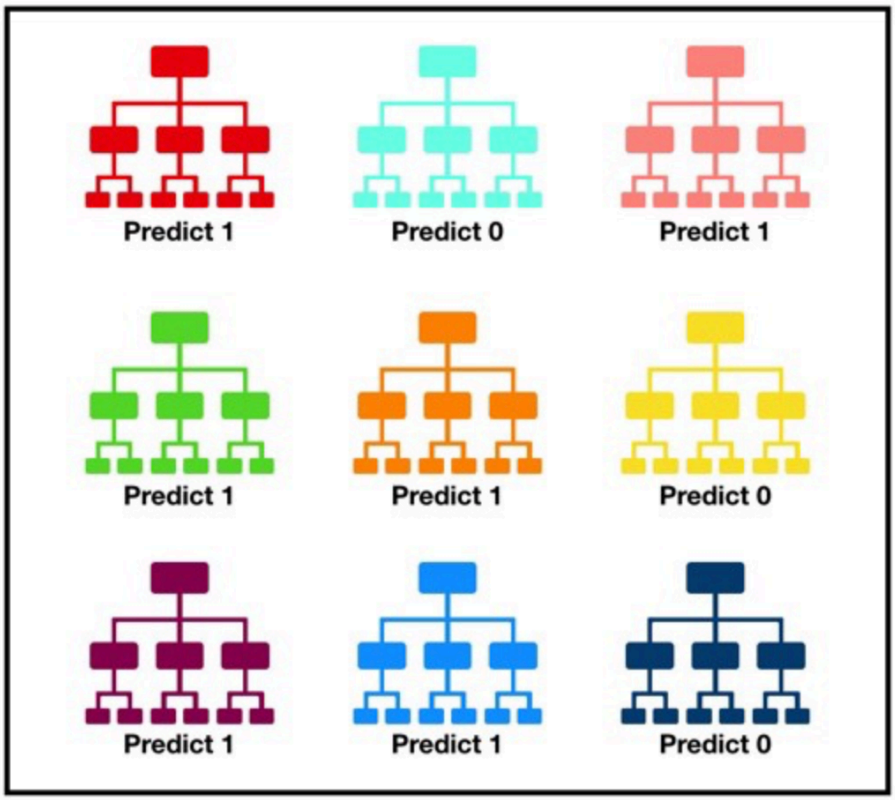


Figura 30 – Esempio di Random Forest Classifier.

Un semplice esempio di *Random Forest Classifier* è riportato nella *Figura 42.* Il modello è composto da nove algoritmi *Decision Tree.* Come si può notare la classe con il maggior numero di voti è Class 1, così la previsione del *Random Forest Classifier* è *Class 1*.

Quello che rende il *Random Forest Classifier* un modello semplice e potente è avere alla base un gran numero di algoritmi *(Decision Trees)* relativamente non correlati tra di loro che operano come unico modello.

Al fine di garantire una bassa correlazione tra i singoli algoritmi all'interno del modello *Random Forest* viene utilizzato il seguente metodo:

BAGGING (Bootstrap Aggregation)
Nel modello di *Decision Tree* la divisione dei singoli nodi è effettuata considerando ogni possibile variabile indipendente e selezionando quella

che produce il minor *Information Gain* (o la maggior riduzione dell'Entropia).
I modelli di *Decision Tree* sono molto sensibili ai dati sui quali viene fatto il training dei parametri; piccole variazioni del dataset portano a strutture ad albero molto diverse.
Il modello *Random Forest* trae vantaggio da questa particolarità dei modelli ad albero. Infatti, l'algoritmo consente ad ogni *tree* di selezionare casualmente un sotto campione del dataset di partenza ottenendo così modelli con strutture diverse.
Questo approccio porta ad una maggior variazione dei dataset di partenza degli algoritmi e di conseguenza alla struttura degli alberi, ciò si traduce in una minor correlazione tra gli algoritmi e in una maggior diversificazione.

Nella *Figura 43* l'algoritmo di *Decision Tree* (in blu) opera la divisione dei nodi scegliendo tra le 4 variabili del modello. La scelta ricadrà sulla variabile che permette di minimizzare l'Entropia (per esempio **Feature 1**).
Guardando la parte destra della *Figura 44*, l'algoritmo di *Random Forest* prende in considerazione unicamente due variabili del dataset (selezionate in modo casuale). L'algoritmo *Random Forest Tree 1* prende in considerazione le variabili x_2, x_3 per la scissione del nodo. Come analizzato nel modello tradizionale (*Decision Tree*, in blu) la variabile x_1è quella che permette di minimizzare l'Entropia del modello, ma l'algoritmo *Random Forest Tree 1* può selezionare unicamente una tra le variabili: x_2, x_3. La scelta ricade così su **Feature 2**.
L'algoritmo *Random Forest Tree 2,* invece, è in grado di selezionare la variabile x_1.

Di conseguenza nel modello *Random Forest* i due algoritmi di Decision Tree non solo sono "addestrati" su dataset diversi, ma utilizzano anche diverse variabili per dividere i nodi. Questo crea algoritmi non correlati tra di loro riducendo la possibilità di commettere errori nella selezione delle variabili.

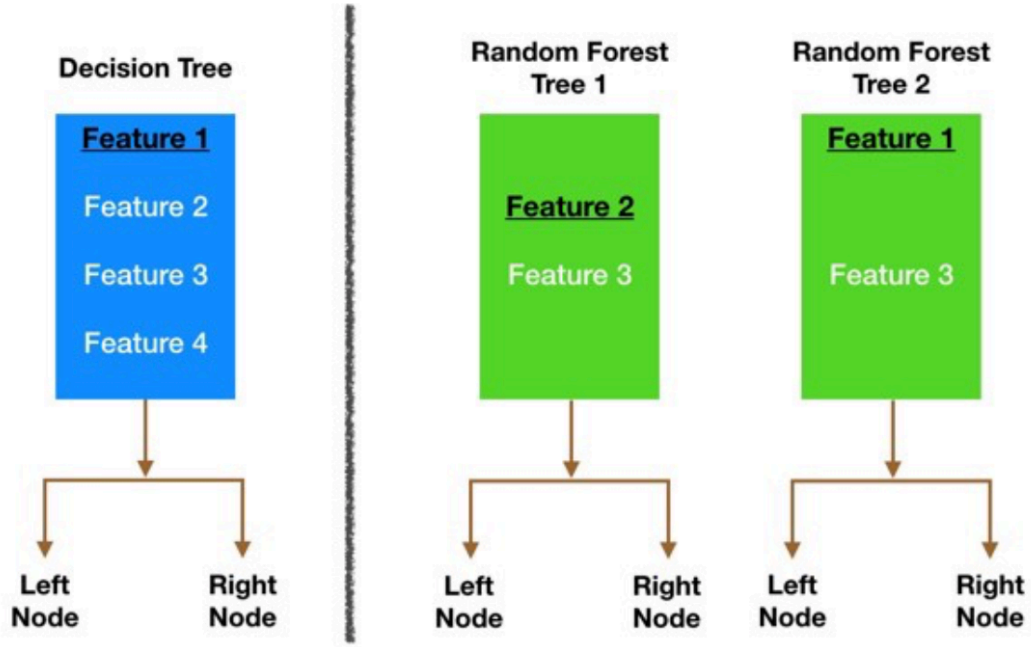


Figura 31 – Divisione dei nodi, Decision Tree vs Random Forest.

Nella *Figura 44* viene illustrato il codice utilizzato per implementare l'algoritmo di *Random Forest Classifier* in *Python.* Nella *riga 1* possiamo notare che il valore della *Cross Validation* è posto uguale a 10. Il numero di alberi utilizzati per predire la classe di ogni titolo è pari a 150 (*riga 2*, n_estiamtors = 150).

Anche nel caso del *Random Forest Classifier* permane il problema dell'*overfitting* degli algoritmi; così nelle *righe 3,4,5* sono stati definiti range di valori per i seguenti parametri:

- la lunghezza dei diversi trees;
- il minimo numero di osservazioni nei final leaves;
- il numero minimo di osservazioni necessarie per procedere allo splitting;

I valori ottimi (minimizzazione dell'Entropia vincolata ai valori imposti per gli Hyperparameters) sono ottenuti mediante Cross-Validation (riportati nella *Figura 45*).

```
1   clf_rf = GridSearchCV(cv=10,
2                 estimator = RandomForestClassifier(n_jobs=-1, random_state=90, n_estimators = 150),
3                 param_grid = {'max_depth': np.array([5, 10, 15, 20, 25]),
4                               'min_samples_leaf': np.array([2,  5,  8, 11, 14]),
5                               'min_samples_split': np.array([5,  7,  9, 11])},
6                 refit = 'accuracy', return_train_score = True,
7                 scoring = 'accuracy')
8
9
10  clf_rf.fit(X,Y)
```

Figura 32 – Random Forest Classifier in Python

```
In [132]: print("tuned hpyerparameters :(best parameters) ",
clf_rf.best_params_)
tuned hpyerparameters :(best parameters)  {'max_depth': 5,
'min_samples_leaf': 14, 'min_samples_split': 5}
```

Figura 33 – Best Hyperparameters tramite Cross Validation.

La *Confusion Matrix* ottenuta è riportata nella *Figura 46.* Come viene illustrato l'algoritmo identifica correttamente 2949 titoli rialzisti (contro i 2786 identificati dal modello Decision Tree) e 104 titoli ribassisti (contro i 432 correttamente identificati dal modello Decision Tree).

	0	1
0	2949	15
1	1062	104

Figura 34 – Confusion Matrix, Random Forest Classifier.

L'accuracy ottenuta dal modello è del 74%, leggermente inferiore a quella ottenuta nel modello Decision Tree (78%).

Artificial Neural Network

I modelli di *Neural Network* sono una classe di algoritmi che tenta di imitare il processo di apprendimento della mente umana.

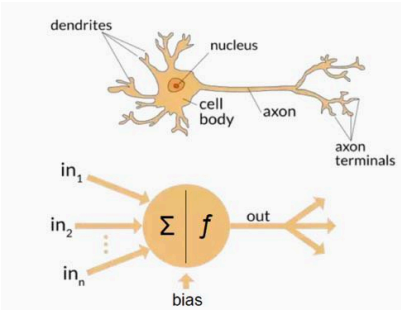


Figura 35

La mente umana consiste in una massa di neuroni interconnessi. I segnali tra neuroni sono trasmessi mediante le sinapsi. Nella *Figura 47* è illustrata una analogia tra il funzionamento dei neuroni nella mente umana, e quello nei modelli di machine learning, entrambi ricevono informazioni da altri neuroni, le processano e le trasmettono ad altri neuroni (Michael N., 2015).

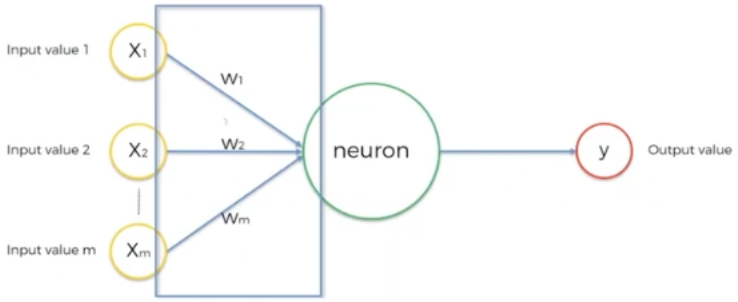


Figura 36 – Schema di un neurone.

Nella versione più semplice una *Rete Neurale* è una funzione (lineare, logistica, etc.) che prende come input valori $x_1, x_2, \dots, x_m$ (variabili indipendenti) con pesi rispettivamente $w_1, w_2, \dots, w_m$ e restituisce valori in output (variabile dipendente) come mostrato nella *Figura 48*. L'output può essere di tipo continuo (es. prezzo di uno stock), binario (es. il prezzo del titolo crescerà o decrescerà nel periodo t+1), o categorico (es. classificazione di immagini).

Un modello con un singolo neurone (come quello riportato nella *Figura 49*) non ha alcun vantaggio se comparato con un classico modello lineare o logistico. Il vantaggio di utilizzare un algoritmo di Neural Network è quello di combinare tra loro più neuroni (*Figura 50*).

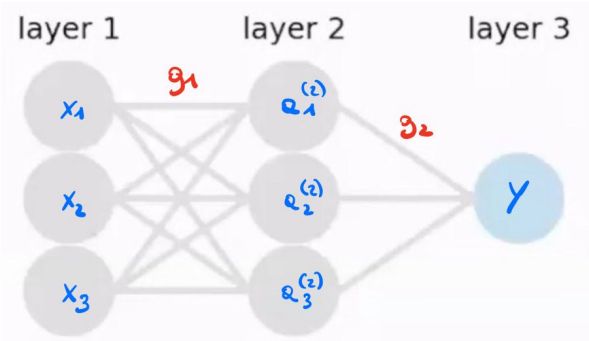


Figura 37 – Neural Network con 3 layers.

Il modello dal punto di vista matematico è composto dal ***layer 1*** in cui sono riportate le variabili in input; esse sono combinate attraverso la prima *funzione di attivazione* g_1 ottenendo così il ***layer 2*** (la variabile $a_0^{(2)}$ non è riportata nel grafico in quanto rappresenta l'intercetta del modello):

$$a_0^{(2)} = 1$$

$$a_1^{(2)} = g_1(w_{10}^{(1)} + w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3)$$

$$a_2^{(2)} = g_1(w_{20}^{(1)} + w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g_1(w_{30}^{(1)} + w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3)$$

Le variabili ottenute $a_1^{(2)}, a_2^{(2)}, a_3^{(2)}$ sono combinate nuovamente nel ***layer 3*** tramite la *funzione di attivazione* g_2, ottenendo:

$$Y = g_2(w_{10}^{(2)} a_0^{(2)} + w_{11}^{(2)} a_1^{(2)} + w_{12}^{(2)} a_2^{(2)} + w_{13}^{(2)} a_3^{(2)})$$

dove,

$$a_i^{(j)} = unità\ i\ nel\ layer\ j;$$

$$w_{ni}^{(j)} = peso\ utilizzato\ nel$$
$$layer\ j-1\ per\ combinare\ le\ variabili, riferito\ all'unità\ i.$$
$$n = 0, \dots, 3$$

La *funzione di attivazione* $g(\cdot)$ non è necessariamente la stessa in ogni layer del neural network. La funzione utilizzata è parzialmente determinata dal tipo di problema in analisi (*Figura 50*).

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Figura 38 – Activation functions.

Le funzioni maggiormente utilizzate nell'ambito del *neural netwok* sono:

- la funzione **ReLU** → se il valore $\sum_{i=1}^{m} w_i x_i$ ottenuto nel neurone è negativo la funzione assegna valore zero, altrimenti restituisce il valore del neurone *(Figura 51)*;
- la funzione **Sigmoid** → quando $\sum_{i=1}^{m} w_i x_i$ assume valori molto elavati, la funzione assegna valore 1, quando al contrario il valore è molto piccolo assegna valore 0. La particolarità della funzione è avere una transizione

"morbida" tra i due valori. È la tipica funzione utilizzata quando l'output è di tipo binario (*Figura 52*).

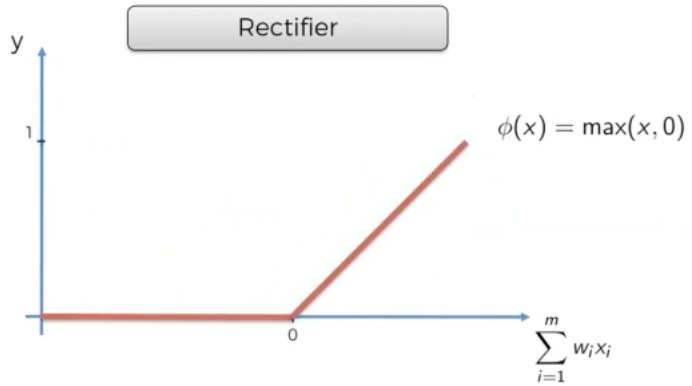


Figura 39 – ReLu function.

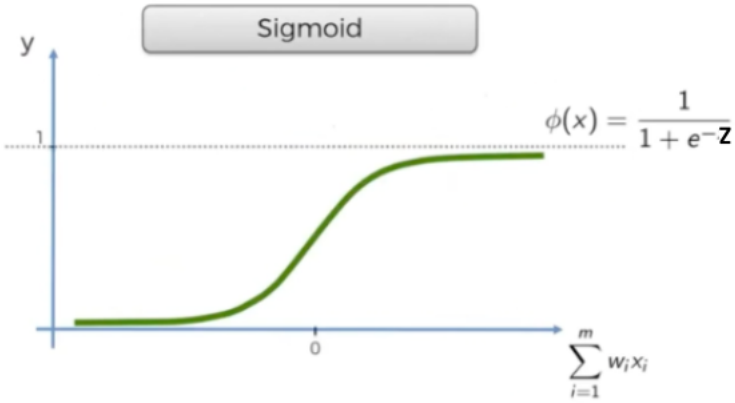


Figura 40 – Sigmoid Function.

Matematicamente la funzione *Sigmoid* è espressa con:

$$y = \frac{1}{1 + e^{-z}}$$

$$z\ rappresenta\ l'input\ \text{pesato:}$$

$$z = w^T x + b$$

$$dove,$$

$$w^T = matrice\ dei\ pesi$$

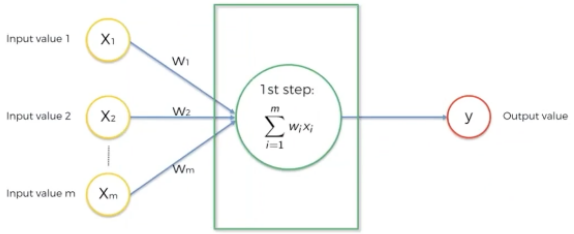
$$b = bias$$

I vari pesi w_i sono inizializzati dal modello come valori casuali diversi da zero, mentre la costante o errore b permette all'algoritmo di avere un ulteriore parametro da calibrare per migliorare la predizione (solitamente inizializzato con il valore 0) (Xavier Gorot e al., 2011).

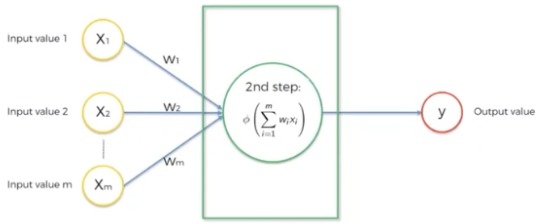
Le fasi fondamentali del modello prendono il nome di *forward propagation* e *backpropagation*.

Durante la fase di *forward propagation* sono preformati una serie di calcoli che possono essere sintetizzati in 3 passaggi:

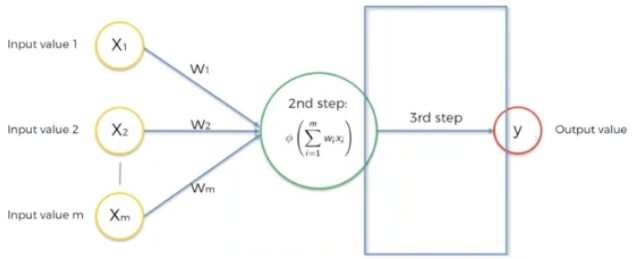
1. ricevere come input le x_m variabili indipendenti e inizializzare il processo con pesi w_m selezionati casualmente e diversi da zero;



2. assegnare una funzione $\phi\ (funzione\ di\ attivazione)$ per la computazione dei valori in input;



3. propagare il calcolo attraverso il valore di output;



Una volta ottenuta la predizione, tramite una *funzione di costo* viene calcolato l'errore sulla stima; l'errore dipende direttamente dai valori assunti dai pesi, in quanto sono gli unici parametri che possono variare all'interno del modello. La funzione di costo utilizzata nei problemi di classificazione prende il nome di *Cross-entropy*, e viene espressa come:

$$-\frac{1}{m}\sum_{i=1}^{m} y_i log\hat{y_i} + (1-\hat{y_i})\log(1-\hat{y_i})$$

$$dove, \hat{y_i}\ \text{è}\ il\ valore\ stimato, e\ y\ \text{è}\ il\ valore\ osservato.$$

La funzione di costo viene minimizzata tornando indietro nel processo e modificando i pesi w_i. Questa seconda fase di minimizzazione della funzione di costo prende il nome di *backpropagation* (Yann LeCun e al., 1998).

La criticità che va sollevata è capire come modificare i pesi al fine di minimizzare correttamente la Loss Function che si realizza attraverso il metodo *Gradient Descent.* La variazione del valore dei pesi assume il nome di *Learning Rate*.

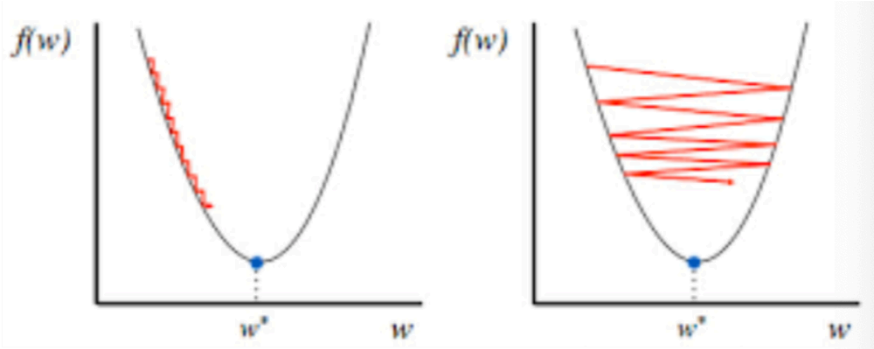


Figura 41 – Gradient Descendent.

Come mostra la *Figura 53 sx* assumere un valore del *Learning Rate* relativamente basso richiederebbe moltissimo tempo e una potenza di calcolo enorme. D'altra parte, assumere un valore del Learning Rate troppo alto implicherebbe il mancato raggiungimento del valore minimo della funzione di costo $f(w)$ causando un'oscillazione infinita (*Figura 53 dx*).

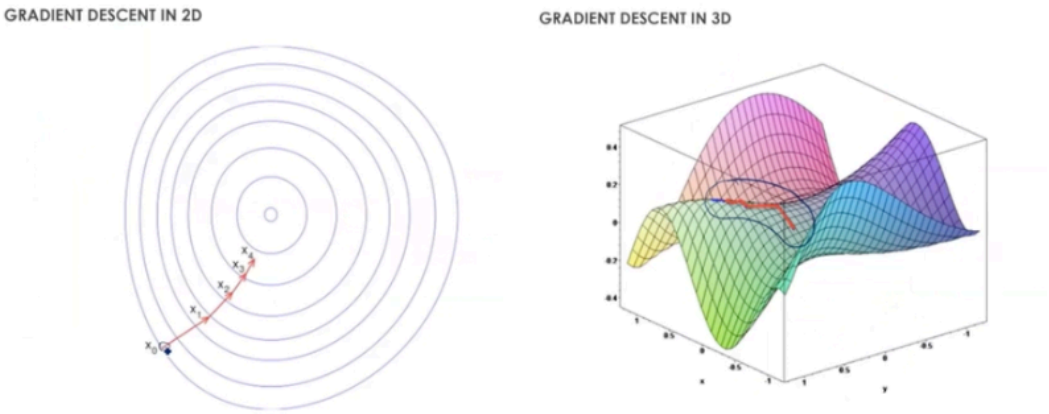


Figura 42 – Gradient Descent in 2D e 3D.

La *Figura 54* mostra la minimizzazione della funzione di costo tramite Gradient Descent in 2D ed in 3D.

La *Figura 53* riporta una funzione di costo ideale che differisce da quella effettiva, che invece assume forme molto più complesse con la presenza di minimi locali (*Figura 55*).

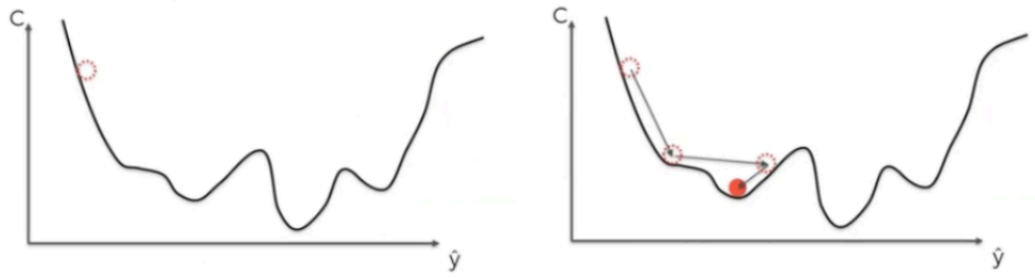


Figura 43 – Cost Function.

Per evitare che il problema di minimizzazione venga risolto in un minimo locale è stato introdotto un diverso tipo di Gradient Descent: *Stochastic Gradient Descent.*

Nel *Gradient Discent* classico viene calcolato l'errore tramite la funzione di costo dopo ogni singola epoch, successivamente si procede alla modifica dei pesi.

Nello *Stochastic Gradient Descent*, invece, viene calcolata la cost function e modificati i pesi delle variabili input dopo ogni singola osservazione. È un metodo dispendioso in termini di calcoli, ma sorprendentemente più veloce rispetto al metodo classico, poiché non necessita che tutti i dati siano caricati nella memoria della macchina che eseguirà l'algoritmo.

Ulteriore alternativa è il *Batch Stochastic Gradient Descent*, dove viene preso un subset random del dataset (batch), calcolata la cost function e modificati i pesi delle variabili input (Andrew Task, 2015).

Ricapitolando, nel presente modello:

1. si inizializzano in modo casuale i pesi con numeri vicini allo zero (ma non zero);
2. viene presa in input la prima osservazione del dataset (223 variabili del titolo i);
3. Forward-Propagation: da sinistra a destra sono attivati i neuroni fino ad arrivare a stimare $\widehat{y_i}$;
4. Si compara il valore di $\widehat{y_i}$ con il valore di y;
5. Back-Propagation: il processo viene invertito da destra a sinistra. Vengono modificati i pesi in accordo con quanto sono responsabili dell'errore della previsione;
6. Si ripetono gli step 1-5 e si aggiornano i pesi dopo ogni osservazione (*Stochastic Gradient Descent*). In alternativa, si ripetono gli step 1-5 e si aggiornano i pesi dopo un gruppo di osservazioni (*Batch Stochastic Gradient Descent*);
7. Quando l'intero dataset è processato attraverso la Neural Network, la prima epoca è completata;
8. La seconda epoca viene inizializzata con i pesi derivanti dalla prima epoca.

```
1   # first layer (input)
2   model.add(Dense(units = units, activation = 'relu', input_dim = 223))
3
4   # dropout(to avoid overfitting)
5   model.add(Dropout(.2))
6
7   # second hidden layer
8   model.add(Dense(units = units,  activation = 'relu'))
9
10  # dropout(to avoid overfitting)
11  model.add(Dropout(.2))
12
13  # Adding the output layer
14  model.add(Dense(units = 1,  activation = 'sigmoid'))
15
16  model.summary()
17
18  model.compile(optimizer = Adam(), loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Figura 44 – Codice utilizzato per implementare la rete neurale in Python.

Nella *Figura 56* è riportato il codice utilizzato per l'implementazione della rete neurale in *Python*. Come si può notare nelle *righe 1, 7 e 13* la rete proposta ha 3 layers. Nei primi due layers è stata implementata la funzione ReLU, nell'ultimo una funzione Sigmoid (che permette di ottenere un output di tipo binario).

Nella *riga 18* della *Figura 57* vediamo che la funzione utilizzata per ottimizzare i pesi è il *Gradient Descent* (Adam) e la *loss function* è di tipo *Cross Entropy*. La metrica di valutazione – come nei modelli implementati fin qui – è l'*accuracy*.

Giacchè permane il problema dell'overfitting, in questo modello si è optato di risolverlo utilizzando un livello di dropout del 20% dopo ogni layer (come mostrano le *righe 4 e 10* della *Figura 57*).

```
1   params = {"batch_size":[100,150,200,300],
2             "epochs":[5,10,15,20],
3             "units":[50,100,150,200]}
4
5   classifier = KerasClassifier(build_fn = nn_model)
6
7   gridsearch = GridSearchCV(classifier,
8                             param_grid = params,
9                             cv = 10,
10                            n_jobs = -1,
11                            scoring = ['accuracy'],
12                            refit = 'accuracy',
13                            return_train_score = True)
14
15  gridsearch.fit(X, Y)
```

Figura 45 – Scelta dei Best Hyperparameters tramite Cross Validation.

Nella *figura 57 riga 1* sono riportati gli Hyperparameters tra i quali viene selezionato il miglior modello. Come si può notare i valori utilizzati per il *batch* nel *gradient descendent* sono: [100, 150, 200, 300]. Le *epochs* testate sono: [5, 10, 15, 20], e i neuroni in ogni layer (units): [50, 100, 150, 200].

L'*accuracy* ottenuta è del 77%, di poco inferiore a quella del miglior modello Decision Tree. I *best parameters* tramite i quali si è ottenuto tale valore sono di seguito riportati:

- Batch size: 250;
- Epochs: 10;
- Units: 150;
- Learning rate: 0.01.

	0	1
0	2911	53
1	902	264

Figura 46 – Confusion Matrix, Neural Network.

Nella *Figura 58* viene illustrata la confusion matrix per il modello Neural Network. Vediamo come il modello è in grado di identificare correttamente 2911 titoli ribassisti e 264 titoli rialzisti.

Strategia di Investimento

Osservando i modelli di machine learning implementati nel presente elaborato sul dataset Train (contenente dati riferiti al 2017) si può rilevare come l'accuracy più alta (78%) sia ottenuta mediante il modello Decision Tree (*Figura 59*).

MODELLO train dataset	ACCURACY	PRECISION TP/(TP+FP)	NEGATIVE PREDICITVE VALUE TN/(TN+FN)	CROSS-VALIDATION	TIME TO FIT THE MODEL (secondi)
Logistic Regression Ridge Penalization	69%	68,90%	72,80%	10	157
Logistic Regression Lasso Penalization	62%	70,80%	72%	10	162
Logistic Regression Elasticnet Penalization	63%	29,80%	72,10%	10	192
Polynomial Features Ridge Penalization	72%	57,10%	71,80%	10	18552
Polynomial Features Lasso Penalization	72%	57,10%	71,80%	10	25306
Decision Tree Cross-Validation	78%	70,80%	77,10%	10	1404
Random Forest	74%	87,40%	76,00%	10	1803
ANN	77%	83%	74,30%	10	93

Figura 47 – Risultati dei modelli di Machine Learning implementati.

La colonna *Precision* nella *Figura 59* indica la percentuale di aziende rialziste correttamente identificate, ossia quelle aziende per le quali è stato correttamente predetto un incremento del prezzo nell'anno successivo al periodo di stima.

La colonna *Negative Predictive Value,* al contrario, indica la percentuale di aziende per le quali è stato correttamente predetto un decremento del prezzo nell'anno successivo al periodo di stima.

Nell'ultima colonna della *Figura 59* è riportato il tempo impiegato dai modelli per la stima dei parametri.

Poiché lo scopo del presente lavoro di tesi è quello di selezionare un paniere di titoli (rialzisti/ribassisti) da inserire in portafoglio, un'idea potrebbe essere quella di selezionare il miglior modello per l'individuazione di titoli rialzisti (sui quali assumere una posizione *long*) e il miglior modello per la selezione di titoli ribassisti (sui quali assumere una posizione *short*).

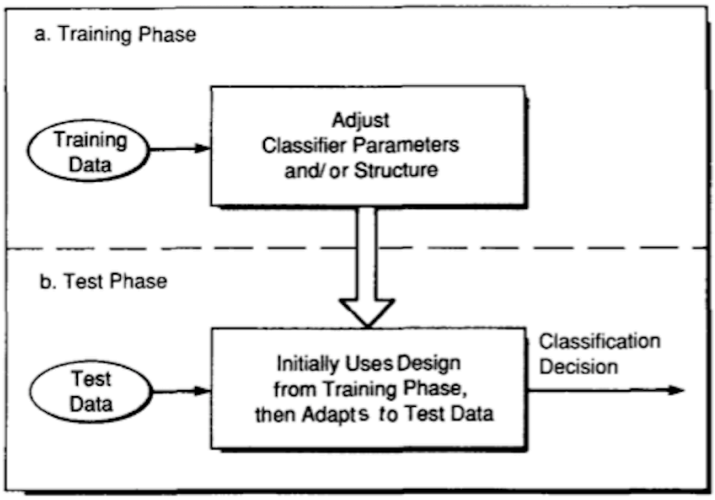


Figura 48 – Fasi della strategia.

Terminata la *fase di Training,* nella quale sono stati stimati i coefficienti dei modelli, si entra nella *fase di Testing*. In questa seconda fase i coefficienti stimati nella fase di train sono utilizzati per predire la classe di appartenenza delle diverse azioni (*Figura 60*).

Nella *Figura 61* sono riportate le accuracy ottenute nella fase di test dai diversi modelli.

MODELLO test dataset	ACCURACY
Logistic Regression Ridge Penalization	39%
Logistic Regression Lasso Penalization	39%
Logistic Regression Elasticnet Penalization	39%
Polynomial Features Ridge Penalization	32%
Polynomial Features Lasso Penalization	32%
Decision Tree Cross-Validation	41%
Random Forest	33%
ANN	36%

Figura 49 – Accuracy sul Test.

Come si può notare i valori sono lontani dall'essere soddisfacenti in quanto le capacità predittive dei modelli sono relativamente basse.

Al fine di migliorare l'accuracy della predizione sono stati implementati due accorgimenti:

1. implementare la strategia utilizzando due diversi modelli;

2. applicare un cut-off.

I due diversi modelli implementati sono stati selezionati sulla base della *Tabella* riportata nella *Figura 59* in particolare tenendo conto dei risultati della colonna *Precision,* della colonna *Negative Predictive Value* e del tempo impiegato dai modelli nella stima dei parametri.

Criticità: osservando la colonna *Precision,* il modello *Random Forest* a primo impatto potrebbe sembrare una valida scelta per la selezione dei titoli rialzisti. In realtà la quasi totalità dei titoli presenta una probabilità di rialzo del prezzo nell'anno t+1 inferiore al 50%. Per questa ragione si è deciso di utilizzare il modello del Random Forest per la selezione di titoli ribassisti.

Dall'altro lato la scelta del modello per la selezione dei titoli sui quali assumere una posizione long è ricaduta sul modello *Artificial Neural Network*, poiché riporta un valore della Precision dell'83% ed ha il vantaggio di essere il modello più veloce nella stima dei parametri (93 secondi).

Per la selezione delle azioni da inserire in portafoglio è stato utilizzato un Cutoff 80% 20%:

- Tramite il modello *ANN* sono state selezionate le azioni con una probabilità di aumento del prezzo nell'anno t+1 maggiore dell'80% (*Figura 62 sx*);
- Tramite il modello *Random Forest* sono state selezionate le azioni con una probabilità minore del 20% (*Figura 62 dx*).

Index	Prob_up	Y	Y_hat
SID	0.9982059	1	1
IROQ	0.99594855	1	1
PMTS	0.98644626	0	1
IGLD	0.9737234	0	1
INOD	0.96361214	0	1
PFE	0.95861876	0	1
EGAN	0.95309466	1	1
EHTH	0.9465132	1	1
SAH	0.94040257	1	1
CSSE	0.93876374	1	1
AMGN	0.9270668	1	1
PBR	0.9260841	1	1
JNJ	0.9154911	1	1
GNL	0.91134655	1	1
DPZ	0.9054855	1	1
COHN	0.8914768	0	1
NEE	0.88490844	1	1
ABBV	0.8779921	1	1
BNSO	0.8704891	1	1
PEP	0.84765065	1	1
D	0.84116375	1	1
GIGM	0.83092964	0	1
FHB	0.8278273	1	1
YUM	0.8203994	1	1
RH	0.81970465	1	1
EDAP	0.81512654	1	1
PEN	0.8151253	1	1
AVGO	0.81416166	1	1
DUK	0.8140901	1	1
DTE	0.8137617	1	1
WEC	0.81278515	1	1
ED	0.8112744	1	1
CCUR	0.8072082	1	1
QCOM	0.8053876	1	1
ELS	0.80085814	1	1

Index	Prob_up	Y	Y_hat
ABIL	0.199779	0	0
MEIP	0.199719	0	0
SNSS	0.199575	0	0
RSLS	0.199542	0	0
RIOT	0.198802	0	0
INPX	0.198038	0	0
NAOV	0.19786	0	0
PULM	0.196179	0	0
ANY	0.196124	0	0
SGLB	0.19499	0	0
MARA	0.193966	0	0
MKGI	0.193636	1	0
PSTI	0.193398	0	0
FUV	0.193389	0	0
BPMX	0.193181	0	0
JAGX	0.192882	0	0
CVM	0.192264	1	0
CREG	0.191571	0	0
SEII	0.186591	0	0
ASTC	0.185853	0	0
PRPO	0.185208	0	0
BIOC	0.18382	0	0
ACRX	0.181707	0	0
GNUS	0.180583	0	0
OPGN	0.179955	0	0

Figura 50 – Le tabelle riportano i due modelli utilizzati per la selezione dei titoli da inserire in portafoglio. Nella parte sx sono riportati i titoli selezionati mediante metodo ANN (lato long), mentre nella parte dx sono riportati i titoli selezionati mediante metodo Random Forest Classifier (lato short).

Nella *Figura 62 sx* si illustrano le azioni selezionate tramite il modello *Artificial Neural Network.* Come si può notare sono presenti tre colonne, la prima contenente le probabilità di un rialzo dei corsi azionari nell'anno successivo a quello di stima, la seconda contenente i valori osservati sul mercato (1 se l'azione

ha visto nell'anno t+1 un rialzo dei prezzi, 0 altrimenti), la terza colonna il valore predetto dal modello.
Come si può notare il modello ha classificato correttamente l'83% dei titoli.

Nella *Figura 62 dx* sono, invece, riportate le azioni selezionate tramite il modello *Random Forest*. Si nota come in questo caso il modello classifica correttamente il 92% dei titoli azionari.

LONG

	Ticker	Company Name	Market	Sector	PE ratio 2017	PE ratio 2018	PE ratio 2019	Market cap	Price 31/18	Price 31/19
1	SID	Companhia Siderúrgica National	NYSE	Basic Materials	2.06	2.39	3.70	3.768 B	2,24	3,45
2	IROQ	IF Bancorp, Inc.	NasdaqCM	Financial Services	18.39	52.23	13.71	53.434 M	20	23,38
3	PMTS	CPI Card Group Inc.	Other OTC	Industrials	N/A	N/A	N/A	26.053 M	2,25	0,85
4	IGLD	Internet Gold - Golden Lines Ltd.	Other OTC	Communication	N/A	N/A	N/A	2.074 M	153	8,2
5	INOD	Innodata Inc.	NasdaqGM	Technology	1.33	1.51	N/A	51.854 M	1,45	1,15
6	PFE	Pfizer Inc.	NYSE	Healthcare	10.21	22.97	N/A	210.383 B	43,21	38,79
7	EGAN	eGain Corporation	NasdaqGS	Technology	N/A	N/A	N/A	380.232 M	6,69	7,99
8	EHTH	eHealth, Inc.	NasdaqGS	Financial Services	12.04	N/A	N/A	1.592 B	36,7	95,67
9	SAH	Sonic Automotive, Inc.	NYSE	Consumer Cyclical	8.72	11.37	N/A	1.837 B	13,51	30,68
10	CSSE	Chicken Soup for the Soul Entertainment, Inc.	NasdaqGS	Consumer Cyclical	4.49	2.03	N/A	177.464 M	7,85	7,76
11	AMGN	Amgen Inc.	NasdaqGS	Healthcare	65.313	15.32	N/A	148.069 B	191,93	240,28
12	PBR	Petróleo Brasileiro S.A. - Petrobras	NYSE	Energy	N/A	11.82	N/A	52.379 B	13,21	15,78
13	JNJ	Johnson & Johnson	NYSE	Healthcare	290.06	N/A	26.91	402.769 B	128,18	145,1
14	GNL	Global Net Lease, Inc-	NYSE	Real Estate	1.82	1.78	77.52	1.595 B	17,67	20,08
15	DPZ	Domino's Pizza, Inc.	NYSE	Consumer Cyclical	30.89	28.67	37.12	16.381 B	252,14	292,69
16	COHN	Cohen & Company Inc.	NYSE	Financial Services	4.79	N/A	N/A	20.097 M	8,31	3,52
17	NEE	NextEra Energy, Inc.	NYSE	Utilities	13.51	12.39	38.52	136.553 B	173,01	241,57
18	ABBV	AbbVie Inc.	NYSE	Healthcare	29.73	25.12	20.85	166.423 B	91,6	88,21
19	BNSO	Bonso Eletronics International Inc.	NasdaqGS	Technology	4.76	N/A	40.88	16.001 M	1,84	2,27
20	PEP	PepsiCo, Inc.	NasdaqGS	Consumer Defensive	34.72	12.5	28.23	192.985 B	110,58	139,69
21	D	Dominion Energy, Inc.	NYSE	Utilities	17.01	19.11	137.30	65.06 B	73,32	82,27
22	GIGM	GigaMedia Limited	NasdaqGS	Technology	30.5	N/A	N/A	29.399 M	3,05	2,46
23	FHB	First Hawaiian, Inc.	NasdaqGS	Financial Services	22.63	11.66	10.94	2.166 B	22,28	28,83
24	YUM	Yum! Brands, Inc.	NYSE	Consumer Cyclical	21.14	19.15	28.84	29.064 B	91,79	100,67
25	RH	RH	NYSE	Consumer Cyclical	204.15	N/A	42.23	6.462 B	120,23	210,78
26	EDAP	EDAP TMS S.A.	NasdaqGS	Healthcare	N/A	N/A	N/A	106.075 M	1,62	4,35
27	PEN	Penumbra, Inc.	NYSE	Healthcare	8.24	10.85	N/A	8 B	125,82	163,16
28	AVGO	Broadcom Inc.	NasdaqGS	Technology	62.49	N/A	61.09	136.847 B	313,12	257,83
29	DUK	Duke Energy Corporation	NYSE	Utilities	19.09	22.95	28.29	58.217 B	86,34	90,89
30	DTE	DTE Enerrgy Company	NYSE	Utilities	17.16	17.26	18.46	22.51 B	110,38	129,27
31	WEC	WEC Energy Group, Inc.	NYSE	Utilities	17.26	20.61	25.06	29.25 B	69,21	92,35
32	ED	Consolidated Edison, Inc.	NYSE	Utilities	16.82	17.26	17.63	23.544 B	76,15	90,19
33	CCUR	CCUR Holdings, Inc.	Other OTC	Technology	2.17	3.12	3.56	27.906 M	3,38	4,26
34	QCOM	Qualcom Incorporated	NasdaqGS	Technology	31.09	N/A	49.64	131.906 B	57,12	87,99
35	ELS	Equity LifeStyle Properties, Inc.	NYSE	Real Estate	40.55	40.64	50.96	11.853 B	48,19	69,66

Figura 51 – Lato Long.

Nella *Figura 63* sono riportati i titoli selezionati dal modello *ANN* da inserire nel Lato Long del portafoglio. Per ogni azienda è riportato il settore di riferimento e la relativa capitalizzazione. Sarebbe stata interessante un'analisi riguardante l'andamento dei PE negli anni antecedenti l'investimento. Tale analisi non è stata resa possibile dalla mancanza di dati.

SHORT

	Ticker	Company Name	Market	Sector	PE ratio 2017	PE ratio 2018	PE ratio 2019	Market cap	Price 31/18	Price 31/19
1	ABIL	Ability Inc.1	Other OTC	Technology	N/A	N/A	N/A	1.905 M	1,97	0,2
2	MEIP	MEI Pharma, Inc.	NasdaqGM	Healthcare	1.76	3.31	N/A	280.908 M	2,44	2,42
3	SNSS	Sunesis Pharmaceuticals, Inc.	NasdaqGM	Healthcare	4.39	3.94	N/A	52.18	0,41	0,34
4	RSLS	ReShape Lifesciences Inc.	Other OTC	Healthcare	0.43	N/A	N/A	18.994 M	28,8	5,55
5	RIOT	Riot Blockchain Inc.	NasdaqCM	Financial Services	3.62	4.48	N/A	160.217 M	1,58	1,14
6	INPX	Inpixon	NasdaqCM	Technology	N/A	N/A	N/A	50.825 M	133,2	5,22
7	NAOV	NanoVibronix, Inc.	NasdaqCM	Healthcare	6.17	50.78	N/A	6.407 M	3,35	2,73
8	PULM	Pulmatrix, Inc.	NasdaqCM	Healthcare	N/A	9.91	N/A	38.19 M	2,6	0,85
9	ANY	Sphere 3D Corp.	NasdaqGS	Technology	3.46	N/A	N/A	12.525 M	3,38	0,8
10	SGLB	Sigma Labs, Inc.	NasdaqCM	Technology	3.66	7.78	N/A	9.816 M	1,44	1,02
11	MARA	Marathon Patent Group, Inc.	NasdaqGS	Industrials	3.81	5.14	N/A	75.857 M	1,64	0,86
12	MKGI	Monaker Group, Inc.	NasdaqGS	Consumer Cyclical	N/A	N/A	N/A	35.328 M	1,26	2,1
13	PSTI	Pluristem Therapeutics, Inc.	NasdaqGS	Healthcare	4.06	4.88	N/A	276.086 M	7,6	4
14	FUV	Arimoto, Inc.	NasdaqGS	Consumer Cyclical	3.89	5.06	N/A	206.105 M	2,65	1,62
15	BPMX	Timber Pharmaceuticals Inc	NYSE	Healthcare	5.86	2.24	N/A	14.439 M	3758,4	692,92
16	JAGX	Jaguar Health, Inc.	NasdaqCM	Healthcare	N/A	N/A	N/A	15.591 M	15,68	0,65
17	CVM	CEL-SCI Corporaion	NYSE	Healthcare	N/A	N/A	N/A	524.129 M	2,95	8,56
18	CREG	China Recycling Energy Corp.	NasdaqCM	Industrials	0.15	0.08	N/A	6.817 M	6,7	2,8
19	SEII	Sharing Economy International Inc.	Other OTC	Industrials	0.28	0.05	N/A	1.105 B	0,34	0,21
20	ASTC	Astrotech Corporation	NasdaqGS	Industrials	1.09	3.68	N/A	16.439 M	5,15	1,82
21	PRPO	Precipio, Inc.	NasdaqCM	Healthcare	N/A	N/A	N/A	36.027 M	2,43	1,97
22	BIOC	Biocept, Inc.	NasdaqGS	Healthcare	17.58	0.79	N/A	87.954 M	0,88	0,28
23	ACRX	AcelRx Pharmaceuticals, Inc.	NasdaqGS	Industrials	N/A	N/A	N/A	110.195 M	2,35	2,09
24	GNUS	Genius Brand International, Inc.	NasdaqCM	Consumer Cyclical	412.90	N/A	N/A	250.789 M	2,09	0,25
25	OPGN	OpGen, Inc.	NasdaqCM	Healthcare	N/A	4.81	N/A	45.518 M	25,6	1,16

Figura 52 – Lato Short.

La *Figura 64* riporta il lato short della strategia, ossia i titoli selezionati tramite il modello *Random Forest* per i quali si prevede una caduta del prezzo negli anni successivi l'investimento. Nel lato short la mancanza di dati per l'analisi dell'andamento del P/E è ancora più evidente.

	Ticker	Company Name	TOT. Investito	Q.tà titoli	Price 31/18	Price 31/19	Capital Gain	Profitto
LONG	SID	Companhia Siderúrgica National	428,57 €	191	2,24	3,45	1,21	231,51 €
	IROQ	IF Bancorp, Inc.	428,57 €	21	20	23,38	3,38	72,43 €
	PMTS	CPI Card Group Inc.	428,57 €	190	2,25	0,85	-1,4	-266,67 €
	IGLD	Internet Gold - Golden Lines Ltd.	428,57 €	3	153	8,2	-144,8	-405,60 €
	INOD	Innodata Inc.	428,57 €	296	1,45	1,15	-0,3	-88,67 €
	PFE	Pfizer Inc.	428,57 €	10	43,21	38,79	-4,42	-43,84 €
	EGAN	eGain Corporation	428,57 €	64	6,69	7,99	1,3	83,28 €
	EHTH	eHealth, Inc.	428,57 €	12	36,7	95,67	58,97	688,63 €
	SAH	Sonic Automotive, Inc.	428,57 €	32	13,51	30,68	17,17	544,68 €
	CSSE	Chicken Soup for the Soul Entertainment, Inc.	428,57 €	55	7,85	7,76	-0,09	-4,91 €
	AMGN	Amgen Inc.	428,57 €	2	191,93	240,28	48,35	107,96 €
	PBR	Petróleo Brasileiro S. A. - Petrobras	428,57 €	32	13,21	15,78	2,57	83,38 €
	JNJ	Johnson & Johnson	428,57 €	3	128,18	145,1	16,92	56,57 €
	GNL	Global Net Lease, Inc-	428,57 €	24	17,67	20,08	2,41	58,45 €
	DPZ	Domino's Pizza, Inc.	428,57 €	2	252,14	292,69	40,55	68,92 €
	COHN	Cohen & Company Inc.	428,57 €	52	8,31	3,52	-4,79	-247,03 €
	NEE	NextEra Energy, Inc.	428,57 €	2	173,01	241,57	68,56	169,83 €
	ABBV	AbbVie Inc.	428,57 €	5	91,6	88,21	-3,39	-15,86 €
	BNSO	Bonso Eletronics International Inc.	428,57 €	233	1,84	2,27	0,43	100,16 €
	PEP	PepsiCo, Inc.	428,57 €	4	110,58	139,69	29,11	112,82 €
	D	Dominion Energy, Inc.	428,57 €	6	73,32	82,27	8,95	52,31 €
	GIGM	GigaMedia Limited	428,57 €	141	3,05	2,46	-0,59	-82,90 €
	FHB	First Hawaiian, Inc.	428,57 €	19	22,28	28,83	6,55	125,99 €
	YUM	Yum! Brands, Inc.	428,57 €	5	91,79	100,67	8,88	41,46 €
	RH	RH	428,57 €	4	120,23	210,78	90,55	322,77 €
	EDAP	EDAP TMS S.A.	428,57 €	265	1,62	4,35	2,73	722,22 €
	PEN	Penumbra, Inc.	428,57 €	3	125,82	163,16	37,34	127,19 €
	AVGO	Broadcom Inc.	428,57 €	1	313,12	257,83	-55,29	-75,68 €
	DUK	Duke Energy Corporation	428,57 €	5	86,34	90,89	4,55	22,59 €
	DTE	DTE Enerrgy Company	428,57 €	4	110,38	129,27	18,89	73,34 €
	WEC	WEC Energy Group, Inc.	428,57 €	6	69,21	92,35	23,14	143,29 €
	ED	Consolidated Edison, Inc.	428,57 €	6	76,15	90,19	14,04	79,02 €
	CCUR	CCUR Holdings, Inc.	428,57 €	127	3,38	4,26	0,88	111,58 €
	QCOM	Qualcom Incorporated	428,57 €	8	57,12	87,99	30,87	231,62 €
	ELS	Equity LifeStyle Properties, Inc.	428,57 €	9	48,19	69,66	21,47	190,94 €
SHORT	ABIL	Ability Inc.1	600,00 €	305	1,97	0,2	1,77	539,09 €
	MEIP	MEI Pharma, Inc.	600,00 €	246	2,44	2,42	0,02	4,92 €
	SNSS	Sunesis Pharmaceuticals, Inc.	600,00 €	1463	0,41	0,34	0,07	102,44 €
	RSLS	ReShape Lifesciences Inc.	600,00 €	21	28,8	5,55	23,25	484,38 €
	RIOT	Riot Blockchain Inc.	600,00 €	380	1,58	1,14	0,44	167,09 €
	INPX	Inpixon	600,00 €	5	133,2	5,22	127,98	576,49 €
	NAOV	NanoVibronix, Inc.	600,00 €	179	3,35	2,73	0,62	111,04 €
	PULM	Pulmatrix, Inc.	600,00 €	231	2,6	0,85	1,75	403,85 €
	ANY	Sphere 3D Corp.	600,00 €	178	3,38	0,8	2,58	457,99 €
	SGLB	Sigma Labs, Inc.	600,00 €	417	1,44	1,02	0,42	175,00 €
	MARA	Marathon Patent Group, Inc.	600,00 €	366	1,64	0,86	0,78	285,37 €
	MKGI	Monaker Group, Inc.	600,00 €	476	1,26	2,1	-0,84	-400,00 €
	PSTI	Pluristem Therapeutics, Inc.	600,00 €	79	7,6	4	3,6	284,21 €
	FUV	Arimoto, Inc.	600,00 €	226	2,65	1,62	1,03	233,21 €
	BPMX	Timber Pharmaceuticals Inc	600,00 €	0	3758,4	692,92	3065,48	489,38 €
	JAGX	Jaguar Health, Inc.	600,00 €	38	15,68	0,65	15,03	575,13 €
	CVM	CEL-SCI Corporaion	600,00 €	203	2,95	8,56	-5,61	-1.141,02 €
	CREG	China Recycling Energy Corp.	600,00 €	90	6,7	2,8	3,9	349,25 €
	SEII	Sharing Economy International Inc.	600,00 €	1765	0,34	0,21	0,13	229,41 €
	ASTC	Astrotech Corporation	600,00 €	117	5,15	1,82	3,33	387,96 €
	PRPO	Precipio, Inc.	600,00 €	247	2,43	1,97	0,46	113,58 €
	BIOC	Biocept, Inc.	600,00 €	682	0,88	0,28	0,6	409,09 €
	ACRX	AcelRx Pharmaceuticals, Inc.	600,00 €	255	2,35	2,09	0,26	66,38 €
	GNUS	Genius Brand International, Inc.	600,00 €	287	2,09	0,25	1,84	528,23 €
	OPGN	OpGen, Inc.	600,00 €	23	25,6	1,16	24,44	572,81 €

Capitale da investire	30.000,00 €
Capitale da investire LONG	15.000,00 €
Capitale da investire SHORT	15.000,00 €
Capitale per singolo titolo LONG	428,57 €
Capitale per singolo titolo SHORT	600,00 €
Guadagno lato LONG	3.391,79 €
Guadagno lato SHORT	6.005,27 €
Rendimento lato LONG	11,31%
Rendimento lato SHORT	20,02%
Rendimento TOT.	31,32%

Figura 53 – Investment Strategy.

Nella *Figura 65* è illustrata la strategia di investimento. È una strategia hedgata, ossia il capitale derivante dalla vendita delle azioni sul lato short è investita sulle azioni del lato Long. Si ipotizza di essere esposti ad un investimento di 15000 euro su entrambi i lati del mercato. Come possiamo notare dalla parte destra della figura, il rendimento dal lato long è del 11,31% mentre quello dal lato short è del 20,02%. Il rendimento totale della strategia è del 31,32% su un *holding period* di un anno. Interessante notare come il maggior beneficio apportato al portafoglio derivi dal lato short.

Al fine di confrontare i rendimenti del portafoglio proposto con le strategie implementate da O&P e H&L sono stati calcolati i rendimenti anomali.

La misura di Abnormal Return utilizzata è quella del ***market adjusted return***:

$$AR = r_{im} - r_m$$
$$r_{im}:\ i - esima\ azione\ dell'indice\ m$$
$$r_m:\ rendimento\ medio\ dell'indice\ m$$

In base alla disponibilità dei dati è stata proposta una variazione della formula:

$$AR = r_{is} - r_s$$
$$r_{is}:\ i - esima\ azione\ del\ settore\ s$$
$$r_s:\ rendimento\ medio\ del\ settore\ s$$

```
In [5]: print(r.mean())
Sector
Basic Materials           20.972785
Communication Services     7.073853
Consumer Cyclical         15.893584
Consumer Defensive        20.486332
Energy                    -5.629604
Financial Services        20.323551
Healthcare                26.949686
Industrials               22.336964
Real Estate               24.916260
Technology                29.867591
Utilities                 23.577136
```

Figura 54 – Rendimenti medi per settore su un orizzonte temporale di un anno.

Nella *Figura 66,* sono riportati i vari settori analizzati nel presente elaborato con i rispettivi rendimenti medi calcolati sulla finestra temporale: 2018 – 2019.

Quindi, sottraendo ai rendimenti dei titoli in portafoglio i relativi rendimenti medi per settore di appartenenza si ottiene una misura dell'Abnormal Return generato.

Abnormal Return lato LONG	2,65%
Abnormal Return lato SHORT	8,42%

Rendimento TOT.	11,07%

Figura 55 – Abnormal Return

Nella *Figura 67* vengono illustrati i rendimenti anomali, possiamo notare come il lato Short sia decisivo per la performance di portafoglio segnando un rendimento anomalo del 8,42%, contro il 2,65% derivante dal lato long.

La strategia proposta è in grado di generare un rendimento anomalo dell'11,07% su un Holding Period di un anno. I rendimenti anomali *market adjusted* riportati dalle strategie di Ou & Penman e Holthausen & Larker su un Holding Period di un anno sono rispettivamente 8,34% (non includendo i mercati OTC) e 7,33%.

Conclusioni

Il presente paper propone un approccio per la selezione dei titoli azionari da inserire in portafoglio basato su tecniche di Machine Learning. Partendo dalle strategie di Ou & Penman e Holthausen & Larker è stata indagata la possibilità di migliorare i modelli da loro proposti. A differenza delle due coppie di studiosi è stata vagliata la possibilità di predire quali titoli avranno variazioni di prezzo positive negli anni successivi all'implementazione della strategia partendo da una serie di indicatori di bilancio, piuttosto che classificare la variazione degli utili (O&P) o la variazione degli Abnormal Return (H&L).

I risultati ottenuti, sebbene soddisfacenti, non sono direttamente confrontabili con gli studi precedenti. In primis perché si guarda a grandezze diverse, in secundis perché vengono analizzati mercati di riferimento diversi.

Sono stati analizzati 5 modelli: Logistic Regression, Polynomial Feature, Decision Tree, Random Forest Classifier e Artificial Neural Network. Tutti i modelli hanno riportato valori dell'accuracy molto interessanti quando testati tramite Cross-Validation sui dati di Training (78% tramite Decision Tree, 77% mediante Artificial Neural Network).

Guardando all'accuracy dei modelli sui dati di test, si è deciso di implementare una strategia di investimento basata su due diverse tecniche di Machine Learning: *Random Forest Classifier* e *Artificial Neural Network*. Con lo scopo di migliorare ulteriormente la predizione si è applicato un cut-off 80-20 in modo da selezionare unicamente i titoli per i quali è stata predetta una probabilità di aumento del prezzo nell'anno t+1 maggiore dell'80% (assumendo su tali titoli una posizione long) e le azioni per le quali la probabilità di aumento del prezzo è risultata minore del 20% (impostando una strategia short). Il cut-off selezionato ha permesso di inserire in portafoglio 60 titoli, su 35 dei quali è stata assunta una posizione long e sui restanti 25 una posizione short.

La strategia proposta ha permesso di classificare correttamente l'83% dei titoli sul lato long e il 92% dei titoli sul lato short.

Il rendimento grezzo ottenuto dalla strategia sul lato long si attesta intorno all'11,30%, mentre il lato short segna un rendimento del 20% su un Holding Period di un anno. Guardando alla misura dei rendimenti anomali (calcolati come rendimento in eccesso rispetto al settore di riferimento) la strategia registra sul lato long un rendimento del 2,65%, mentre sul lato short riporta un rendimento dell'8,42%, ottenendo così un AR dell'11,07% su un Holding Period di un anno.

Sarebbe stato interessante poter analizzare se la strategia di analisi fondamentale proposta è di tipo momentum o contrarian, ma l'impossibilità di accedere a piattaforme dati ha minato questo filone di analisi.

Una criticità da rilevare riguarda l'implementazione del modello ANN. Infatti, i modelli di *Neural Network* sono delle "scatole nere" perché risulta impossibile conoscere quanto ogni singola variabile indipendente influenzi la variabile dipendente. Tuttavia, va precisato che i modelli proposti rappresentano meramente uno strumento utile ad affiancare il lavoro e le conoscenze dei gestori di portafoglio, senza la pretesa di sostituirli.

Per i futuri lavori sarebbe interessante validare i modelli proposti sui mercati analizzati dalle due coppie di studiosi, così come approfondire il bias che la strategia proposta nel presente elaborato va a colpire, confrontandolo con il bias sfruttato dalla strategia di Ou & Penman (l'investitore naive dimentica la mena reversion dei corsi azionari).