**UNIVERSITÁ DEGLI STUDI DI MILANO-BICOCCA**

Scuola di Economia e Statistica

Corso di laurea Magistrale in

SCIENZE STATISTICHE ED ECONOMICHE


# Natural Language Processing in finance
# Un'applicazione basata sul modello FinBERT

Relatore: Prof. Matteo Maria Pelagatti

Correlatore: Prof. Antonio Candelieri

Tesi di Laurea di:

Moreno Sanna

Matr. N. 783008

Anno Accademico 2020/2021

*Alla mia famiglia*

# Indice

# Elenco delle figure

# Elenco delle tabelle

# Introduzione

L'epoca moderna sempre più digitale è governata da una moltitudine di informazioni fornite dalla persone in via diretta e indiretta, anche tramite le piattaforme social che ogni giorno raccolgono milioni di messaggi condivisi dagli utenti. I mercati finanziari d'altra parte si muovono sulla base delle informazioni che caratterizzano gli asset finanziari, tramite i prezzi che oscillano in maniera più o meno accentuata. Lo scopo di questa tesi è quello di andare ad indagare la possibile relazione esistente fra i messaggi condivisi sui social dalle persone e il movimento dei mercati. Per estrarre informazioni dalle frasi degli utenti è stato utilizzato un modello di elaborazione del linguaggio naturale.

Il Natural Language Processing è un campo interdisciplinare che riguarda le relazioni fra gli elaboratori e il linguaggio umano. Più in particolare riguarda come i computer possono leggere, capire ed elaborare le informazioni provenienti dalla comunicazione umana, analizzando uno o più testi di grandi dimensioni, indipendentemente dal loro contenuto o linguaggio. E' una branca relativamente recente, che ha avuto un notevole impulso intorno al 1980, in relazione con la sempre più notevole potenza computazione degli elaboratori e l'introduzione di algoritmi di machine learning. I modelli di elaborazione del linguaggio naturale possono essere utilizzati per archiviare vari compiti, tra cui:

- La traduzione di un testo da una lingua di origine ad una lingua target.
- La comprensione dei legami fra le parole di una frase e più in generale della semantica di un dato testo.
- Question Answering, cioè la risposta automatica a domande espresse in una data lingua.
- Sentiment Analysis, cioè l'apprendimento del tipo di significato della frase, inteso come il sentimento che l'autore ha voluto trasmettere formulando l'espressione.

Proprio su quest'ultima capacità di analisi si è basata la tesi. Per prima cosa si sono estratti tutti i messaggi condivisi dagli utenti nel social network Twitter nel periodo che va dal 01 gennaio 2019 al 31 Marzo 2020, connessi a due particolari aziende presenti sul mercato automobilistico, Tesla e BMW. Dopodiché i tweet sono stati analizzati tramite un modello di NLP, specifico per trattare i testi con contenuto prettamente finanziario. Ad ogni frase sono stati associati gli indicatori del sentimento della frase prodotti dal modello, che identificano il contenuto positivo, negativo o neutrale espresso dall'utente tramite il messaggio. Su queste nuove variabili si sono poi costruiti vari modelli statistici per prevedere le serie storiche degli asset connessi ai tweet estratti, cioè i valori e i segni dei rendimenti e degli extrarendimenti relativi a Tesla e BMW

riferiti sempre allo stesso periodo in cui sono stati condivisi i messaggi. I modelli sono stati allenati sulle informazioni relative all'anno 2019 per prevedere i primi tre mesi dell'anno 2020.

E' stato scelto di utilizzare un modello di NLP specifico per i testi finanziari in modo da avere più probabilità di cogliere il sentimento delle persone relativo all'andamento delle due aziende sul mercato.

Questo lavoro quindi si prefigge di indagare i seguenti aspetti:

1. Cercare di capire se è presente della relazione (e la sua entità) fra le informazioni contenute nei messaggi condivisi sui social network e l'andamento dei mercati, prendendo come campione i tweet del social Twitter e due aziende del settore automobilistico, Tesla e BMW.
2. Indagare se questa relazione è più forte per un titolo "mainstream" come può essere Tesla e meno per un titolo più tradizionale come può essere BMW.
3. Capire se questa relazione cambia se si considerano diverse serie storiche relative allo stesso asset (valori o segni dei rendimenti o extra-rendimenti).
4. Comprendere se è possibile costruire una strategia finanziaria basata sulle informazioni estratte dai messaggi degli utenti.

La tesi verrà presentata come segue.

- Un primo capitolo sarà dedicato alla spiegazione dei modelli utilizzati, sia di NLP sia statistici.
- Un secondo capitolo verrà dedicato alla spiegazione del processo che ha portato alla costruzione del dataset su cui si sono costruite le analisi, propedeutico alla spiegazione dei risultati ottenuti.
- Un terzo capitolo di presentazione dei risultati, spiegando anche i modelli utilizzati come confronto, le metriche utilizzate per la valutazione delle performance e la costruzione dei metodi.
- Un capitolo con le conclusioni e considerazioni finali.

Il tutto sarà seguito dalla bibliografia e dalla sitografia. Il link alla repository Github contenete i codici sviluppati durante questa tesi verrà riportato alla fine del documento in allegato.

# 1. Modelli utilizzati

## 1.1. Introduzione al modello FinBERT

Il modello di elaborazione del linguaggio naturale utilizzato durante questa tesi si chiama FinBERT. Si tratta di una implementazione basata su un precedente modello NLP di nome BERT, rendendolo specifico per l'analisi di testi prettamente finanziari. Il fine ultimo del modello è quello della Text classification, più nello specifico della Sentiment Analysis, cioè l'attribuzione di un livello di "sentimento" associato alla frase analizzata, che può essere positivo, negativo o neutrale. Una frase che per esempio indica che i mercati aprano in negativo il giorno successivo avrà una valenza negativa, viceversa sarà positiva se si augura al rialzo. FinBERT è pensato per cogliere questo, attribuendo ad ogni frase un'etichetta (ed un sentiment score) che indica il suo livello di sentimento.

Per capire il funzionamento del modello FinBERT dobbiamo fare un salto indietro di 3 modelli. Infatti come accennato precedentemente FinBERT si basa su un precedente modello chiamato BERT, (acronimo di Biderctional Encoder Representation from Transformers), che a sua volta eredita la propria architettura da un modello NLP di nome Transformers.

Di seguito verrà quindi analizzato il modello Transformers e la sua architettura, per poi passare al modello BERT, ed infine a FinBERT.

## 1.2. Il modello Transfomers

### *1.2.1. Introduzione*

Il modello Transformers[1] è stato sviluppato nel 2017 come alternativa a modelli come le Recurrent Neural Network, le Long Short-Term Memory e le Gated Recurrent Neural Networks, che in quegli anni erano all'avanguardia nella modellizzazione di testi linguistici per portare a termine compiti come la risposta automatica di domande o problemi di traduzione. Questi modelli funzionano in maniera sequenziale, dove ad ogni istante t generano una sequenza di hiddent state h(t) come funzione del precedente hidden state h(t-1) e dell'input al tempo t. Questa sequenzialità preclude la parallelizzazione dei modelli in fase di training, che può diventare critica se si trattano delle frasi particolarmente lunghe a causa della memoria

[1] Sviluppato dagli autori del seguente articolo, su cui si è costruita questa spiegazione: Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

finita degli elaboratori. Nonostante le migliorie applicate a questi metodi, il problema iniziale rimane.

Per sopperire a questo il modello transformers applica un nuovo meccanismo chiamato "Attention Mechanism", che permette di modellare le relazioni fra le parole di una frase indipendentemente dalla loro distanza nel testo. Il vantaggio di questo metodo è che non essendo ricorrente, permette di essere parallelizzato ottimizzandone il costo computazione.

Transformers è il primo modello che si basa interamente su questo meccanismo, escludendo l'utilizzo di metodi sequenziali per disegnare le relazioni fra le frasi di input e quelle di output.

Questo come vedremo in seguito con l'analisi del modello BERT, non costituisce solo un vantaggio computazionale ma permette anche il training del modello non solo in una direzione (da sinistra a destra o da destra verso sinistra) ma in entrambe le direzioni (questo è il motivo di Biderectional nel nome), in quanto le parole della frase di input non vengono caricate in maniera sequenziale nell'architettura ma in parallelo.

La parallelizzazione nel modello Transformers conduce oltre ad un già citato miglioramento nel costo computazionale anche a performance superiori nei compiti inerenti al mondo NLP.

Lo scopo per cui è stato costruito Transformers è quello della traduzione. Tuttavia può essere adattato ad altri compiti del Natural Language Processing.

### *1.2.2. Architettura del modello*

Il modello è caratterizzato da due componenti principali, l'encoder ed il decoder. L'encoder ha il compito di mappare una sequenza numerica che rappresenta la frase in input $x = (x_1 \dots . x_n)$ con un'altra sequenza numerica continua $z = (z_1, \dots . . z_n)$. Questo vettore z viene poi passato al decoder che genera ad ogni step un simbolo $y_i$, fino a formare alla fine di ogni step una sequenza di simboli da restituire in output $y = (y_1, \dots . . y_n)$. Il modello è autoregressivo, nel senso che ad ogni step il decoder prende in input oltre a z anche la rappresentazione numerica del simbolo $y_i$ che ha generato al passo precedente come informazione aggiuntiva. Di seguito viene mostrata una rappresentazione dell'architettura del modello. Sulla sinistra abbiamo le layers caratterizzanti la parte di encoder, mentra sulla destra quelle caratterizzanti la parte di decoder. Sebbene le due parti siano molto simili, presentano alcune differenze e per questo verranno analizzate separatamente di seguito.

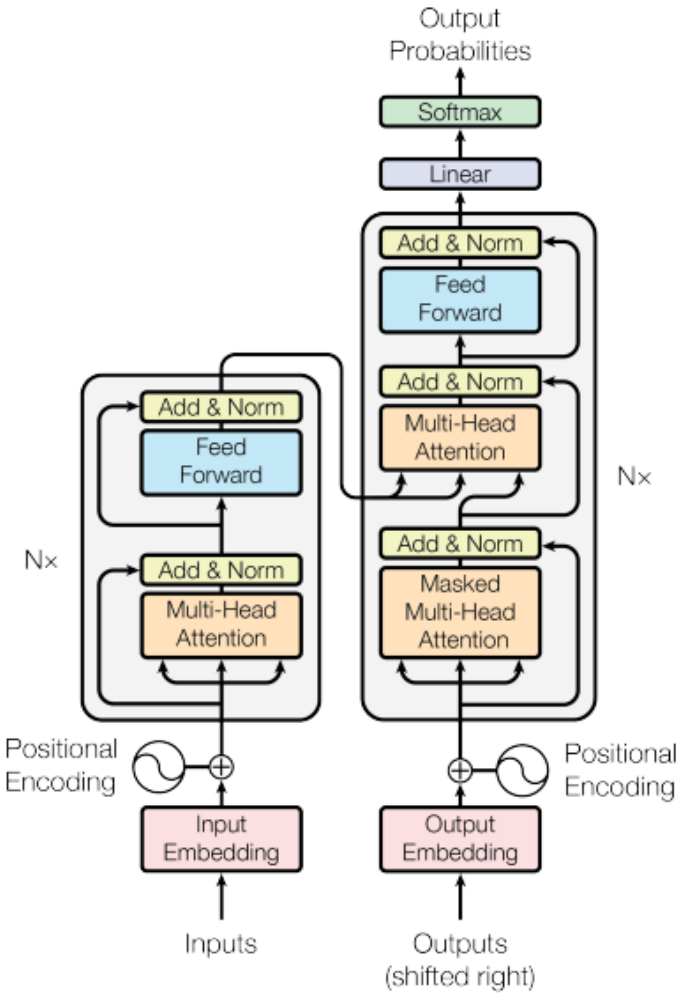


*Figura 1.1 Architettura del modello Transformers* [2]

### *1.2.3. Flusso dei dati*

Nella spiegazione dell'architettura verrà seguito il "percorso" che i dati seguono all'interno del modello. Per prima cosa verrà quindi data particolare attenzione ai meccanismi di word embedding e posizional encoding, che trasformano le parole delle frasi in input in sequenze numeriche che le rappresentano (input embeddings) e che possono essere interpretate dall'algoritmo, tenendo conto anche della posizione che ciascuna parola ha all'interno della frase. Dopodiché verrà spiegato l'encoder, il decoder, ed infine le layers finali.

Il modello analizza una frase alla volta, composta da un numero arbitrario di parole. Queste vengono passate in input dopo essere state trasformate in vettori numerici tramite gli algoritmi di word embedding e positional encoding. Attraversano quindi l'encoder, composto da 6 layers

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

sequenziali identiche tra di loro. L'output di queste layers viene restituito al decoder che è composto anche lui da 6 layers identiche tra loro. La parte di decoding viene completata in più step, in cui ad ogni step i dati attraversano ognuna delle 6 layers del decoder, più le due layers finali che trasformano l'output in simboli. Ad ogni passo l'output del precedente step (una parola tradotta in sostanza) viene dato in input alla prima layer del decoder (dopo che questo è stato codificato tramite i due algoritmi di word emebedding e positional encoding), mentre l'output dell'encoder viene passato ad ogni layer del decoder, come spiegato dall'immagine che segue:

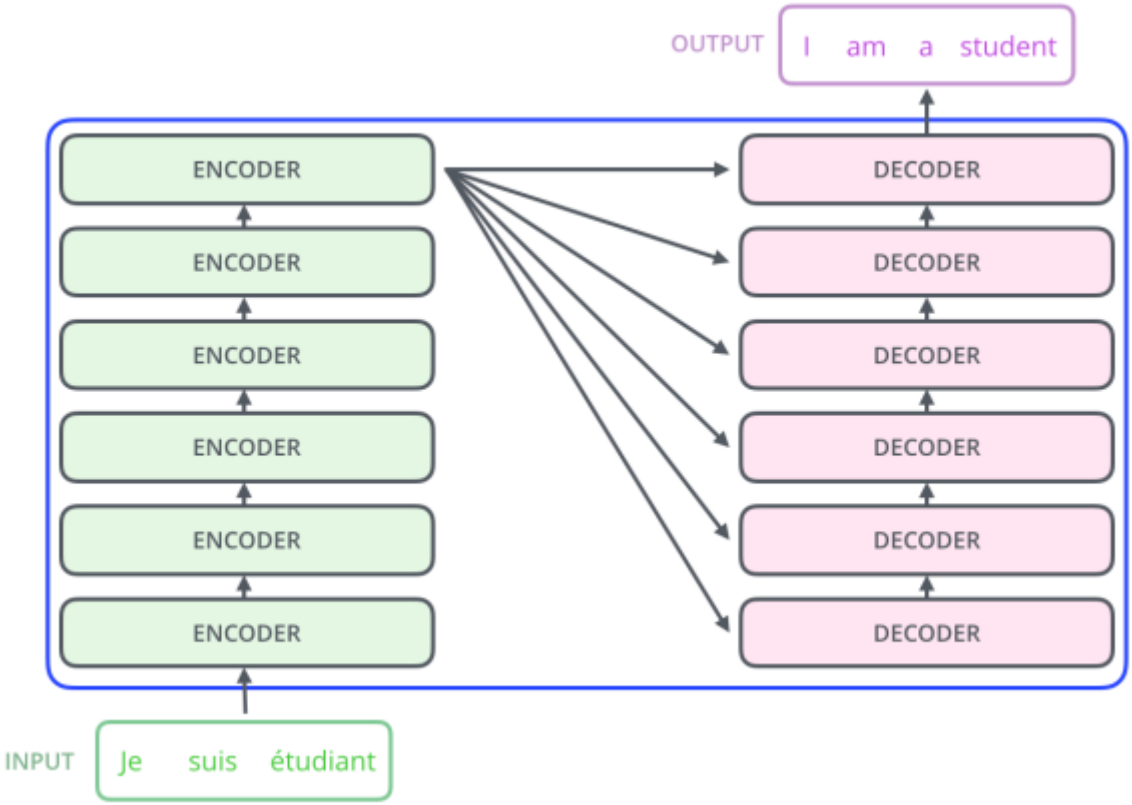


*Figura 1.2 Flusso dei dati nel modello Transformers*[3]

### *1.2.4. Word embeddings: Word2Vec* [4]

Come è usuale nei modelli di NLP, le parole contenute nelle singole frasi vengono prima codificate in una sequenza numerica che le rappresenta, in modo tale che possano essere elaborate dal modello. Questo può avvenire tramite vari algoritmi, che sono usati per mappare le parole o le frasi da un vocabolario ad un corrispondente vettore di numeri reali. Questa

[3] Immagine presa da Jay Allammar, *The Illustrated Transformer*, https://jalammar.github.io/illustrated-transformer/

[4] Per questa parte ci si è riferiti a Jaron Collis, *Glossary of Deep Learning: Word Embedding* 04 – 2017, https://medium.com/deeper-learning/glossary-of-deep-learning-word-embedding-f90c3cec34ca

rappresentazione è sia più efficiente perché riduce la dimensionalità, sia più espressiva perché codifica la parola sulla base del suo significato, e di conseguenza parole con contenuto simile avranno associate vettori che nel piano d-dimensionale saranno più vicine, dove d è la dimensione del vettore. Lo scopo del word embedding è proprio quello di creare una rappresentazione vettoriale d-dimensionale delle parole contenute in un vocabolario di n termini, dove $n >> d$. I vettori rappresentativo finali, (i cosiddetti word vectors), devono mantenere le relazioni fra le parole di origine (per esempio la relazione di appartenenza al mondo animale dei termini cane e gatto). Il word embedding quindi conduce ad una rappresentazione di dimensione contenuta di un testo da un determinato corpus, mantenendo le similitudini fra le parole.

Un unsupervised learning algorithm per produrre Word embedding è Word2Vec. Si tratta di una feed-forward neural network con un singolo strato nascosto lineare (hidden layer neural neurons) ed uno strato di output (output layer softmax classifier). La stringa di testo in input viene codificata in un one-hot encoded vector (un vettore di lunghezza pari al numero totale di parole contenute nel vocabolario, in cui l'elemento corrispondente alla posizione della parola in input nel vocabolario avrà valore 1, mentre gli altri elementi avranno valore 0), che verrà passato nelle unità della hidden layer, (di dimensione inferiore rispetto alla lunghezza del vettore) ed infine nelle unità della layer di output, che avrà dimensione pari alla lunghezza del vettore di input e quindi del vocabolario. La layer di output è una softmax, dove ad ogni unità è associata una parola del vocabolario, che ne condivide la posizione (la prima unità sarà associata alla parola in posizione 1 nel vocabolario, la seconda unità a quella in posizione due, ecc…). Ogni unità calcola la probabilità che la parola a cui è associata possa trovarsi vicino alla parola data in input in una determinata frase. Rappresenta quindi un indice di quanto le due parole siano collegate, ed è utile per fare una previsione sulle parole vicine a quella in input. La Softmax Output Layer restituisce quindi un vettore di probabilità, che sommano a uno. Per quanto riguarda la hidden layer, essendo lineare senza attivazione essa è in sostanza una matrice di pesi con cui viene moltiplicato il vettore in entrata. Questa matrice avrà dimensione pari a $n\ X\ d$, cioè avrà un numero di righe pari alla dimensione del vettore in entrata (e quindi alla lunghezza del vocabolario), ed un numero di colonne pari alla dimensione dell'hidden layer, che è molto inferiore rispetto alla dimensione del vocabolario. Il vettore risultante da questa operazione verrà poi moltiplicato matricialmente per un'altra matrice di dimensione $d\ X\ n$, in modo che il risultato sia un vettore di dimensione pari a quella del vocabolario. Ad ogni suo

elemento verrà applicata la softmax nell'output layer. Di seguito uno schema della rete neurale, ipotizzando n=10000 e d=300.

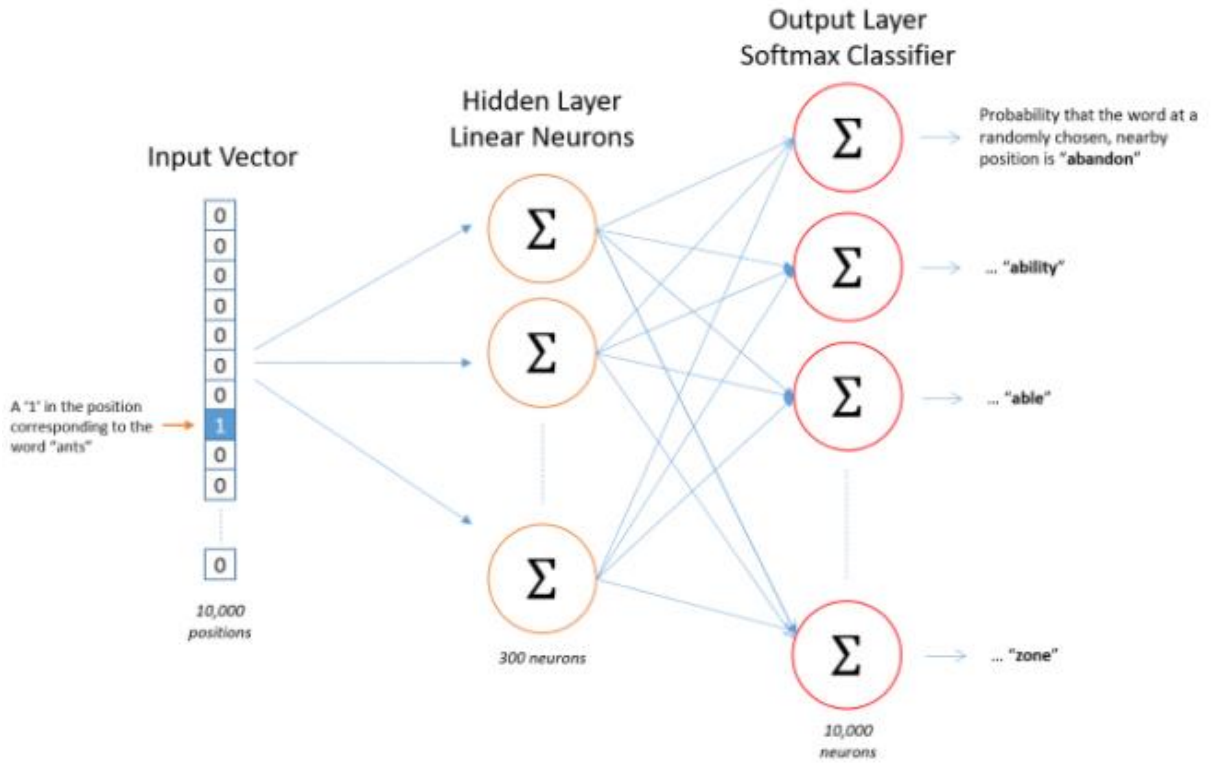


*Figura 1.3 Architettura Word2Vec* [5]

La cosa interessante in tutta la rete neurale è appunto la matrice di pesi della hidden layer, in quanto permette di ridurre la dimensionalità del vettore iniziale producendone un altro di dimensione minore, ma contenente le relazioni che quella parola ha con il resto dei termini nel vocabolario. Infatti la matrice è stimata in fase di training in modo da trovare la rappresentazione migliore della parola, cioè quella che conduce a delle probabilità in grado di indicare quali parole sono più connesse ad essa nel vocabolario. Una volta calcolata la matrice è facile ottenere il vettore rappresentativo di ogni parola del vocabolario come mostra l'esempio qui sotto:

[5] Immagine presa da Jaron Collis, *Glossary of Deep Learning: Word Embedding* 04 – 2017, https://medium.com/deeper-learning/glossary-of-deep-learning-word-embedding-f90c3cec34ca

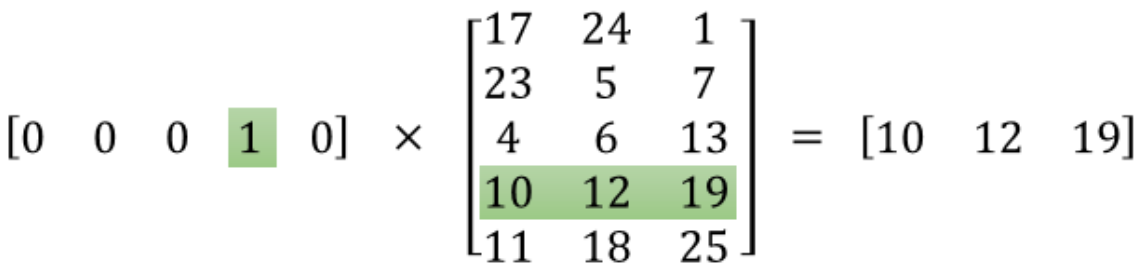

*Figura.1.4 Esempio Word2Vec* [6]

Nell'esempio, la parola corrispondente alla posizione 4 nel vocabolario avrà un vettore rappresentativo di dimensione pari a 3, che coglie le relazioni che questa parola ha anche con le altre parole del vocabolario. Da qui la parola Word2Vec. Questo vettore sarà quello che verrà dato in input all'encoder del modello Transformers, con dimensionalità 512. La dimensione del vettore presenta un trade off. Vettori rappresentativi di dimensione maggiore permettono di cogliere meglio le relazioni fra le parole, ma necessitano di potenza computazionale maggiore per essere trattati.

Word2Vec non è l'unico algoritmo per fare questo (un altro esempio è GloVe), ma il risultato finale rimane ottenere un vettore numerico rappresentativo della parola che gli algoritmi di NLP possono interpretare e gestire.

### *1.2.5. Positional Encoding*

Prima di entrare nel flusso dell'encoder, il vettore subisce una ulteriore trasformazione. Per catturare l'informazione contenuta nella posizione che le singole parole hanno all'interno della frase, al vettore rappresentativo della parola ne viene aggiunto un altro chiamato "positional encoding", della stessa dimensionalità degli input embeddings (i vettori rappresentativi della parola). Data la stessa dimensione i vettori possono essere sommati tra di loro. I valori del vettore positional encoding vengono generati dalle seguenti sinosoidi:

$$PE_{pos,2i} = sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{1.1}$$

[6] Immagine presa da Jaron Collis, *Glossary of Deep Learning: Word Embedding* 04 – 2017, https://medium.com/deeper-learning/glossary-of-deep-learning-word-embedding-f90c3cec34ca

$$PE_{pos,2i+1} = cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (1.2)$$

Dove $pos$ è la posizione della parola (del token) all'interno della frase, $d_{model}$ è la dimensione dell'input embedding/del positional embedding (o la hidden size del modello), e *i* è la posizione del singolo elemento i-esimo del positional embedding, che può assumere valori da 1 a $d_{model}$. La funzione seno restituisce i valori a tutti gli elementi del positional embedding che occupano una posizione pari, mentre la funzione coseno a tutti gli elementi che occupano una posizione dispari. Di seguito un esempio di positional encoding dei token di una frase composta da 10 parole, ipotizzando una dimensione del modello pari a 64. Viene considerato anche il valore 0.

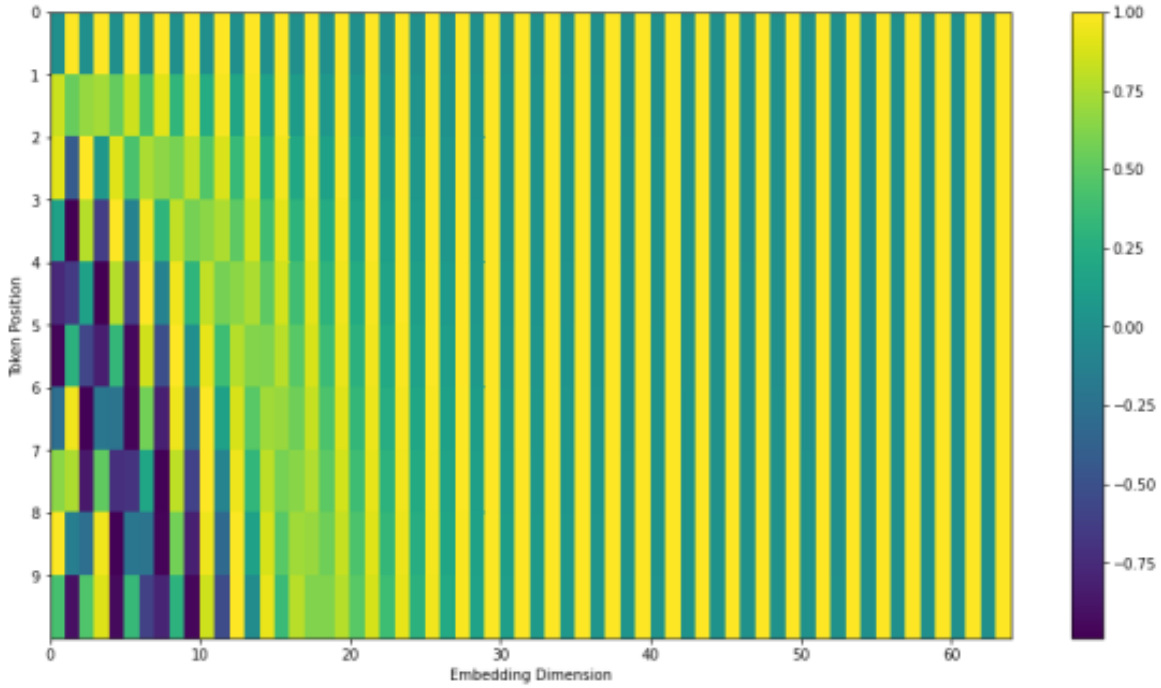


*Figura 1.5 Esempio positional encoding* [7]

Il significato del positional embedding è quello di attribuire a ciascun input embedding una informazione aggiuntiva riguardo la sua posizione all'interno della frase, sotto forma di vettori numerici che cambiano a seconda del punto in cui la parola compare all'interno del testo. Questo perché essendo il modello privo di ricorrenze o convoluzioni, non riuscirebbe a cogliere

[7] Esempio ripreso da Jay Allammar, *The Illustrated Transformer*, https://jalammar.github.io/illustrated-transformer/

in maniera autonoma l'ordine delle parole nelle frase (non entrando queste in maniera sequenziale all'interno dell'algoritmo).

### *1.2.6. Encoder*

Dopo essere state codificate, le parole entrano nell'encoder, che è composto da un gruppo di 6 identici strati (layers). Ogni layer è caratterizzata da due sottostrati (o sublayer). Nella prima sublayer è contenuto un multi head self attention mechanism, mentra nella seconda è presente una feed forward neural network. Le parole vengono analizzate in parallelo e costituiscono tutte insieme l'input dell'encoder. Prima di proseguire il suo percorso, l'output di ogni sublayer viene trasformato seguendo la formula

$$LayerNorm\big(x + Sublayer(x)\big) \tag{1.3}$$

indicata in figura con label Add & Norm. Ogni sublayer restituisce un output di dimensione pari a $d_{model} = 512$. Questa è una cosa comune sia alle layer dell'encoder sia alla layer del decoder. La feed forward neural network viene applicata in parallelo ad ogni vettore $z_1, \dots z_n$ restituito dalla sublayer self-attention. L'output della feed forward diventerà l'input della layer successiva dell'encoder.

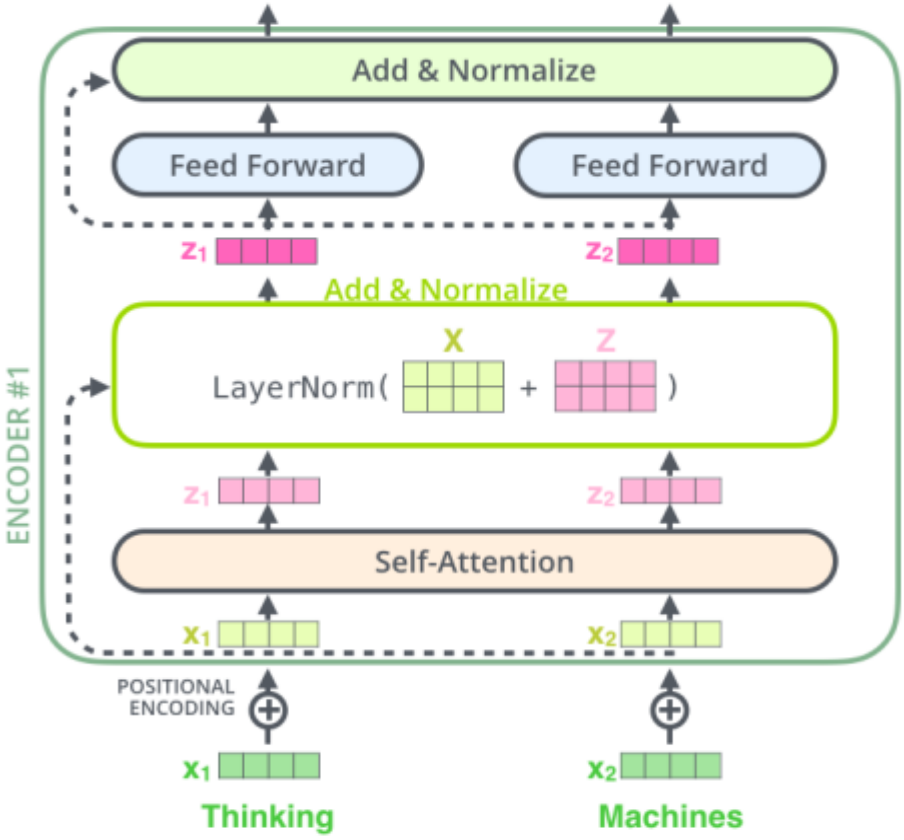


*Figura 1.6 Dettaglio layer encoder* [8]

### *1.2.7. Self Attention Mechanism*

Una funzione di attenzione può essere descritta come una funzione di una query e di un set di coppie di valori/chiave (dove le query, le chiavi e i valori sono tutti dei vettori/matrici[9]) che produce un output. Vi sono due modi per implementare questo, lo Scaled dot product attention e una sua parallelizzazione, il Multi head attention.

[8] Immagine presa da Jay Allammar, *The Illustrated Transformer*, https://jalammar.github.io/illustrated-transformer/

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention is all you need*. 2017. Pg 3

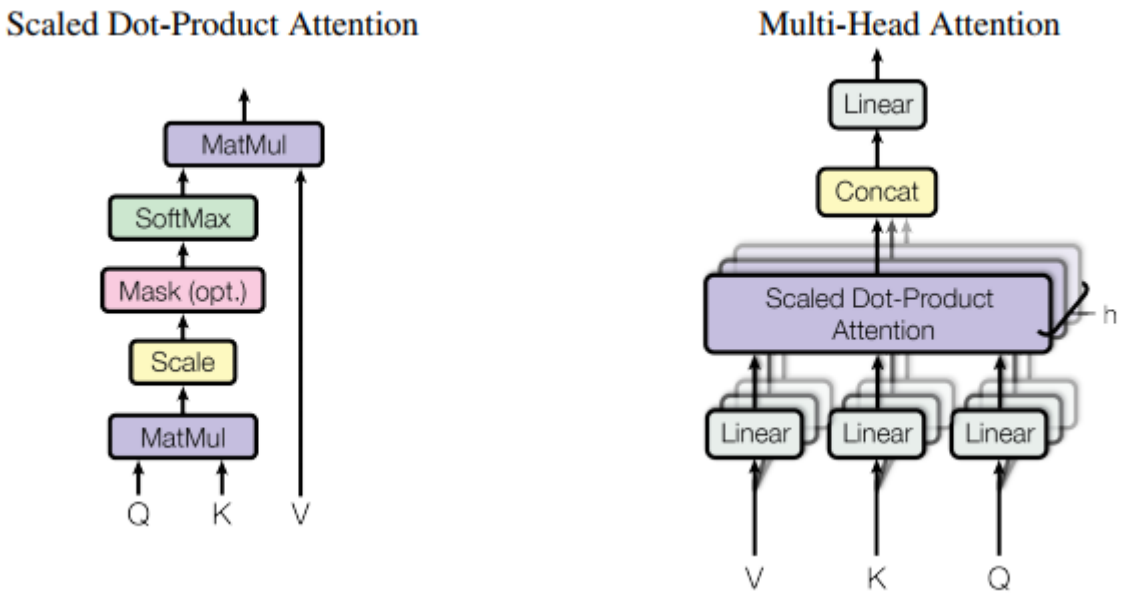


*Figura 1.7 Rappresentazione Scaled & Multi-Head attention* [10]

Il meccanismo del self attention è il concetto chiave che diversifica transformers dai precedenti modelli di NLP. Durante l'analisi delle parole della frase di input, esso permette al modello di capire e codificare meglio le relazioni che sono presenti fra di loro, in modo tale che l'output prodotto trattenga al suo interno anche questa informazione. Questo metodo sfrutta il fatto che le parole vengono analizzate tutte in una volta ed in parallelo.

Di seguito un esempio di questo approccio[11], che permette di far capire al modello a cosa si riferisce il pronome "*it*" all'interno della seguente frase:

*"The animal didn't cross the street because it was too tired"*

Il meccanismo permette di associare il pronome *it* con *animal,* in quanto utilizza per l'analisi di *it* anche le informazioni delle altre parole della frase che permettono di capire meglio il significato del pronome. Di seguito una rappresentazione grafica di tutti i collegamenti che il self-attention attribuisce a "*it".*

[10] Immagine presa da Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention is all you need*. 2017.
[11] Esempio ripreso da Jay Allammar, *The Illustrated Transformer,* https://jalammar.github.io/illustrated-transformer/

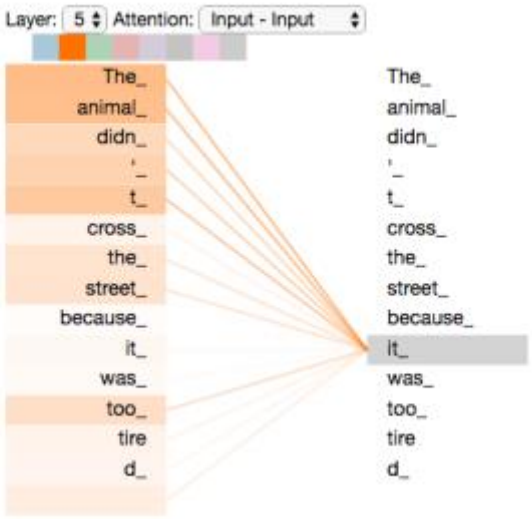


*Figura 1.8 Legami self attention* [12]

### *1.2.8. Scaled Dot Product Attention*

Lo Scaled Dot-Product Attention è il particolare meccanismo formulato dagli autori ed utilizzato nel modello, rappresentato dalla seguente formula:

$$Attention(Q;K;V) = softmax\left(\frac{QK^T}{\sqrt{(d_k)}}\right)V \tag{1.4}$$

Q, K, V sono rispettivamente i vettori della Query e della coppia chiave-valore associati al vettore rappresentativo per la quale si sta calcolando l'attention. Q e K hanno dimensione $d_k$, mentre V ha dimensione $d_v$. Producono un vettore Z che è il prodotto fra il risultato di una funzione esponenziale normalizzata pesata per i valori contenuti nel vettore V.

In realtà gli oggetti Q, K e V sono delle matrici, in quanto il calcolo viene eseguito una sola volta in maniera matriciale per tutte gli input embeddings. Il risultato sarà quindi una matrice Z con un numero di righe pari al numero di parole in input.

[12] Immagine presa da Jay Allammar, *The Illustrated Transformer*, https://jalammar.github.io/illustrated-transformer/

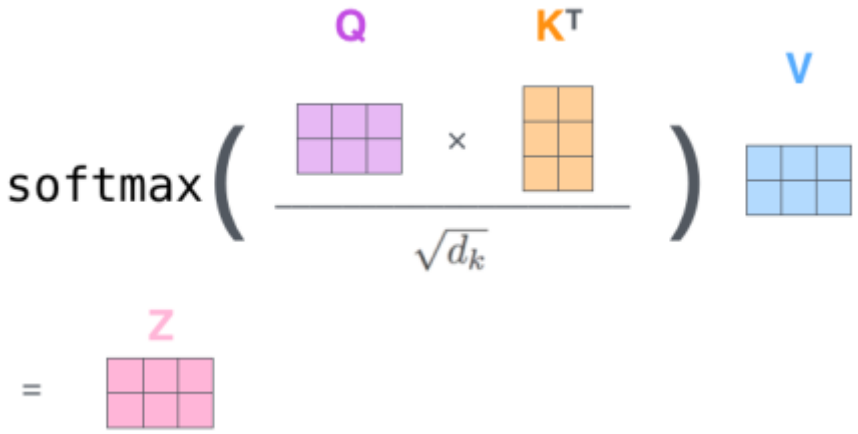


*Figura 1.9 Formula softmax* [13]

Questa formula non è l'unico modo per calcolare l'attention. Le due formule più comuni sono l'additive attention e la dot-product attention, dove l'ultima corrisponde alla formula utilizzata dagli autori, al netto della divisione con $sqrt(d_k)$. Questa è stata scelta in quanto computazionalmente più veloce. Dato che l'additive attention performa meglio se non si divide per il termine $sqrt(d_k)$ quando $d_k$ è molto grande, si è reso necessario scalare per questo termine in modo da ottenere gradienti più stabili[14].

Le matrici Q, K, e V vengono calcolate moltiplicando matricialmente la matrice dei vettori di input (ottenuta concatenando uno sotto l'altro gli input embeddings), per delle matrici di parametri associate a Q, K e V. La matrice Z avrà quindi una dimensione pari a $\text{n°righe}(X)\text{Xd}_{\text{model}}$. Moltiplicando i singoli input embeddings $x_i$ per le matrici associate a Q, K, V, si ottengono i vettori $q_i$, $k_i$e $v_i$ associati ai vari $x_i$. Difatto Q, K e V possono vedersi come questi singoli vettori concatenati uno sotto l'altro. Le matrici dei parametri vengono calcolate in fase di training dal modello.

Per capire meglio cosa rappresentino questi tre oggetti in cui viene di fatto smistato il vettore di input, la Query può essere vista come un set di vettori associati a delle parole *per le quali* stiamo calcolando l'attention, mentre la Key è un set di vettori associati a delle parole *contro le quali* stiamo calcolando l'attention. Il prodotto interno fra queste due matrici rappresenta lo score delle varie parole fra di loro, ossia una misura della loro relazione. I Value sono un set di vettori associati a delle parole rappresentanti il loro *valore.* I Value sono poi moltiplicati per lo

[13] Immagine ripreso da Jay Allammar, *The Illustrated Transformer*, https://jalammar.github.io/illustrated-transformer/

[14] Gli autori ipotizzano che per elevati valori di $d_k$, il prodotto interno esplode, conducendo la funzione softmax in regioni in cui ha gradienti molto bassi. Per evitare questo dividono per il termine $sqrt(d_k)$

score calcolato prima (trasformato in probabilità), in modo da ponderare questi termini per il valore della relazione che essi hanno con la parola *per cui* abbiamo calcolato l'attention, *per cui* stiamo producendo un output, che stiamo codificando. K e V possono anche coincidere, ma non devono per forza essere uguali.[15]

Dato che il calcolo è eseguito in maniera matriciale, ogni elemento della n-esima riga della matrice $QK^T$ è il risultato fra il prodotto matriciale della n-esima riga di Q per la prima, seconda, terza … colonna di $K^T$ e cosi via a seconda del numero di parole in input (e quindi del numero di colonne di $K^T$). Da questa riga verrà poi ricavata la n-esima riga della matrice di output Z, cioè l'output dell'attention relativo all'n°esimo input embedding e quindi all'n-esima parola codificata. Ogni colonna della matrice $K^T$, per come la matrice è stata calcolata è riferita ad un determinato input embedding. Ogni prodotto matriciale fra la n-esima riga della matrice Q e l'r-esima colonna della matrice K rappresenta lo score fra l'r-esima parola nella frase e la n-esima parola per la quale stiamo producendo la n-esima riga della matrice di output Z. Questo valore rappresenta quanta attenzione mettere sulle altre parole della frase quando stiamo codificando una certa parola in una certa posizione, e quindi quanto le altre parole sono connesse e sono importanti per la parola che stiamo analizzando. Tramite questo passaggio si ricavano le relazioni fra le parole nella frase. Questi score vengono poi trasformati in probabilità dalla funzione softmax, essendo i valori tutti positivi e con somma 1. I softmax score determinano quanto ogni parola può essere spiegata nella posizione del n-esimo termine che stiamo considerando, dove la parola che effettivamente occupa quella posizione avrà la probabilità più alta. Questo è utile per vedere quali altre parole possono venire spiegate bene in quel punto oltre a quella che effettivamente occupa quella posizione. A questo punto la matrice V che contiene i valori di tutti gli input embeddings viene moltiplicata matricialmente per il risultato della softmax, in modo da pesare di più i valori riferiti a parole che possono essere spiegate maggiormente in quella posizione. Si ottiene così un vettore di output associato all'n-esimo input embedding che è di fatto una somma ponderata dei vettori contenenti i valori associati a tutti i token della frase da analizzare, con peso maggiore verso quelli più significativi in quella posizione e meno verso gli altri. Essendo il calcolo matriciale, verrà eseguito per ogni riga di Q, ottenendo tutti i vettori di output associati ai vari input embeddings in entrata (cioè le varie righe della matrice di output Z finale).

[15] Attention is all you need; Attentional Neural Network Models | Łukasz Kaiser | Masterclass, *https://www.youtube.com/watch?v=rBCqOTEfxvg&t=946s*

### *1.2.9. Multi-Head Self Attention*

La Multi head self attention è una parallelizzazione della Scaled dot product attention spiegata qui sopra. Invece che utilizzare una singola funzione di attention con oggetti Q, K e V di dimensione pari al modello, gli autori hanno preferito proiettare linearmente questi oggetti h volte, con differenti proiezioni lineari ognuna di dimensione $d_k$, $d_k$, e $d_v$. Ad ognuno di questi set di proiezioni lineari viene applicata in parallelo la funzione di attention spiegata sopra, che conduce ad un output Z di dimensione $d_v$. Questi h output vengono poi concatenati insieme per colonna e proiettati linearmente ancora una volta, per ottenere un output finale Z con dimensione $d_{model}$. La Multi head self attention permette al modello di considerare congiuntamente tutte le informazioni che provengono dalle differenti rappresentazioni lineari degli oggetti Q, K e V. Questo non veniva considerato usando una singola funzione di attenzione. La formula che descrive la Multi head self attention è la seguente:

$$MultiHead(Q,K,V) = Concat(head_1,\dots,head_h)W^O$$

$$where\ head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (1.5)$$

Dove $W_i^Q \in R^{d_{model} X d_k}$, $W_i^K \in R^{d_{model} X d_k}$, $W_i^V \in R^{d_{model} X d_v}$ sono matrici di parametri che proiettano linearmente Q, K e V, mentre $W_i^O \in R^{hd_v X d_{model}}$ è la matrice di parametri che proietta linearmente la concatenazione per colonna di tutte le Z, output delle singole $head_i$, per ottenere un'unica matrice di output di dimensione $N\ X\ d_{model}$ dove $N$ è il numero totale di parole contenute nella frase. Le matrici di parametri per le proiezioni lineari vengono inizializzate in maniera casuale e stimate in fase di training. Nella versione originale il parametro h è uguale ad 8, il che significa che vi sono 8 head nella multi-attention, e che $d_k = d_v = d_{model}/h = 64$ . Per ogni head vi sono diverse matrici che proiettano linearmente i tre oggetti. Nell'immagine le proiezioni lineari sono indicate dai riquadri lineari. Essendo la dimensionalità degli oggetti di input ridotta per ogni head, il costo computazionale di una multi-head self attention è simile a quello di una singola scaled dot product attention con oggetti di input di dimensione pari a $d_{model}$.

L'utilizzo di un meccanismo multi-head aumenta le performance[16], dato che

1. Incrementa l'abilità del modello di focalizzarsi sulle parole nelle differenti posizioni. Con una singola head infatti la softmax restituisce una probabilità molto elevata per la posizione n-esima dell'input $x_i$ a cui ci stiamo riferendo, dominando difatto le

[16] Jay Allammar, *The Illustrated Transformer*, https://jalammar.github.io/illustrated-transformer/

probabilità calcolate per le altre posizioni. Questo porta ad identificare meno le relazioni fra le parole.

2. Produce più sottospazi rappresentativi grazie alle vare proiezioni lineari.

Nelle attention layer dell'encoding i valori delle Query, delle Key e dei Value vengono ricavati dall'output delle precedenti layer.

### *1.2.10. Feed Forward Neural Network*

L'output restituito dalla multi - head, dopo essere stato trasformato e normalizzato, viene passato alla Positional - Wise feed forward neural network, che è applicata ad ogni posizione (cioè ad ogni vettore $z_i$) della matrice di input Z. Essa è espressa dalla seguente formula:

$$FNN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{1.6}$$

I parametri della FNN sono specifici per i diversi tipi di layer. Si tratta di una neural network con un singolo stato nascosto. La dimensione dell'input e dell'output è pari a $d_{model} = 512$ mentre la dimensione dell'hidden layer è pari $d_{ff} = 2048$.

### *1.2.11. Decoder*

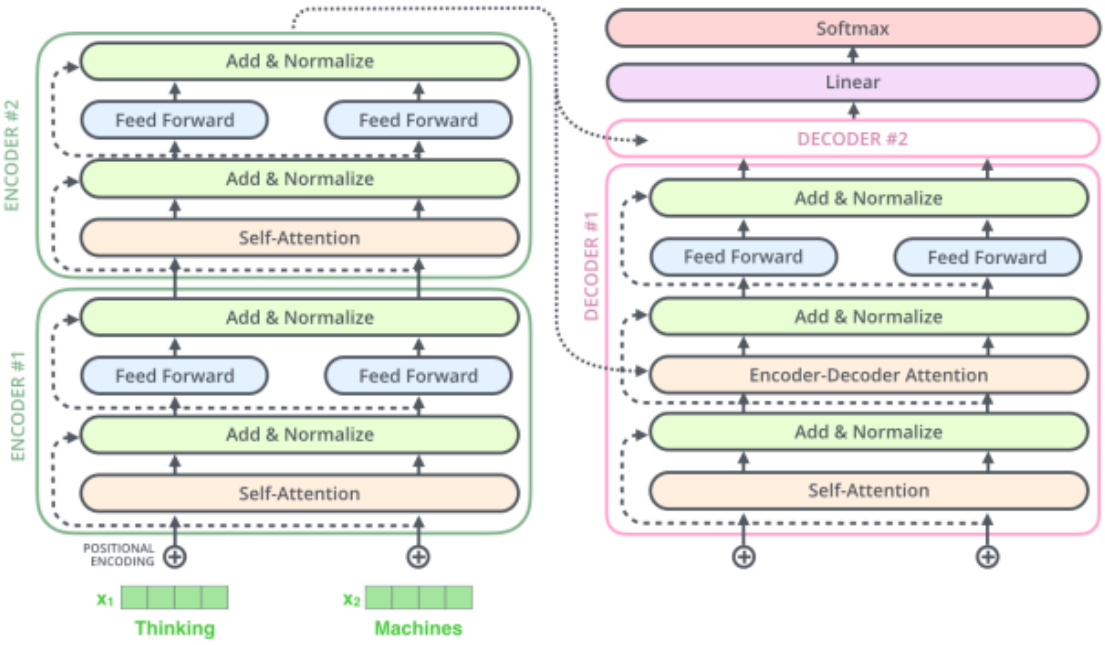


*Figura 1.10 Dettaglio layer encoder & decoder*[17]

[17] Immagine presa da Jay Allammar, *The Illustrated Transformer*, https://jalammar.github.io/illustrated-transformer/

Completata l'elaborazione dell'ultima layer dell'encoding, si arriva al decoder. Il compito del decoder è quello di produrre nuovi simboli partendo dalle informazioni ricavate dall'encoder. E' composto da 6 layer identiche fra loro, molto simili a quelle dell'encoding ad eccezione del fatto che in questo caso troviamo una multi-head in più, l'encoder decoder attention. Anche qui ogni output delle sublayer viene trasformato e normalizzato prima di essere passato alla sublayer successiva.

L'encoder passa direttamente alla sublayer Encoder-decoder attention del decoder le matrici K e V, mentre ricava la matrice Q dall'output della self attention precedente. L'utilizzo delle matrici K e V direttamente dall'encoder aiutano il decoder a focalizzarsi in maniera corretta sulle posizioni della frase di input.

Così come nell'encoder, anche qui l'output delle varie layer diventa l'input di quelle successive, sino ad arrivare all'ultimo strato. Dopodichè l'output finale viene trasformato in un simbolo da due successive layer, (una trasformazione lineare che funge da classificatore ed una softmax che ricava le probabilità per ogni termine). Nel caso della versione originale di transformers questo simbolo è una parola tradotta dalla lingua della frase di input alla lingua target. Si termina quindi il primo step del decoder, che avrà tanti step quante sono le parole in output. Questa nuova parola diventerà un input aggiuntivo dello step successivo del decoder, dopo essere stata trasformata in vettore numerico tramite gli algoritmi di word embedding e positional encoding. Nel caso del primo step, il decoder viene inizializzato con un input di defaul ([start]).[18] La fase del decoder termina quando si arriva all'ultimo step. La fine della frase viene indicata con un token speciale prodotto alla fine dell'ultimo passo (<end of sentence>)[19].

La self attention del decoding presenta delle differenze rispetto a quella dell'encoding. Infatti per preservare la proprietà di autoregressione gli autori prevengono il flusso di informazioni verso sinistra, mascherando (impostando a $-inf$ ) tutti i valori dell'input della funzione softmax che corrispondono ad connessioni irregolari. In questo modo la probabilità connessa ai valori v associati a questi legami non ammessi sarà 0, dato che

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{1.7}$$

[18] Michael phi , *Illustrated Guide to Transformers- Step by Step Explanation*, https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0
[19] Jay Allammar, *The Illustrated Transformer*, https://jalammar.github.io/illustrated-transformer/

, annullando la presenza di questi valori nell'output finale.

Il masking si rende necessario durante la fase di training del modello. Infatti, mentre nell'encoding tutto viene completato in una singola iterazione e tutti i token della frase in input vengono passati alle varie layer, nel decoding il tutto avviene in maniera sequenziale e in più step. Tuttavia durante il training in entrata viene passata sia la frase di input (nella parte dell'encoding) sia la frase di output (nella parte di decoding). Dato che la self attention come abbiamo visto permette di cogliere le relazioni fra ogni parola della frase che sta analizzano, la prima sublayer del decoding andrebbe ad analizzare le relazioni fra parole che, per il funzionamento sequenziale del decoding, non avrebbe disponibili in fase di previsione. Per questo gli score associati a collegamenti non ammessi vengono mascherati in questa fase, in maniera tale che anche nella fase di training il modello non utilizzi informazioni che prevedendo (o traducendo) non avrebbe. L'impostazione a $-inf$ di questi valori avviene sommando alla matrice degli score $QK^T$ una matrice dove nel triangolo superiore ha tutti elementi pari $-inf$ mentre i restanti elementi sono nulli. In questo modo, gli score relativi alle relazioni con parole *"future"* saranno mascherati, simulando quello che succede in più step nella fase di previsione[20]

La feed forward network si comporta come già spiegato nella parte di encoding.

Di seguito un'immagine che rappresenta il flusso dei dati e i vari step del modello completo, in fase di previsione.

[20] Samuel Kierszbaum, *Masking in Transformers' self-attention mechanism, https://medium.com/analytics-vidhya/masking-in-transformers-self-attention-mechanism-bad3c9ec235c*

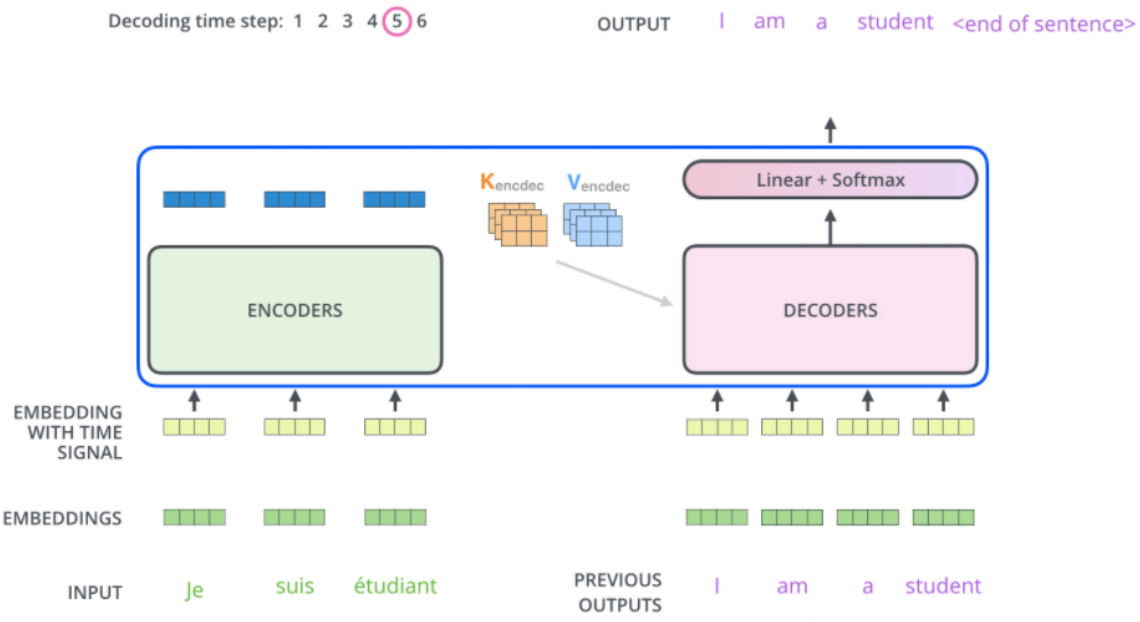


*Figura 1.11 Fase di previsione Transformers*[21]

### *1.2.12. Layer finale lineare e softmax*

La layer finale del modello prende in input i valori rilasciati dall'ultima layer del decoder, ed agisce da classificatore. Il suo compito è trasformare i valori del decoder in parole come output finale del modello. Essa è una fully connected neural network che proietta l'output del decoder in uno spazio più grande, un vettore con valori logit di dimensione molto più elevata. Precisamente avrà dimensione pari a tutti i possibili valori (parole) che possono essere assunti, cioè tutte le parole del vocabolario della lingua target che sono state imparate dal modello in fase di training.

Successivamente il vettore dei valori logit viene passato all'ultima layer del modello, una softmax, che lo trasforma in un vettore di probabilità. I valori diventano tutti positivi e sommano ad uno. La dimensionalità del vettore non viene alterata, rimanendo pari alla dimensione del vocabolario target.

A questo punto viene scelto il termine del vocabolario che occupa la posizione con probabilità più alta nel vettore softmax.

[21] Immagine presa da Jay Allammar, *The Illustrated Transformer*, https://jalammar.github.io/illustrated-transformer/

### *1.2.13. Vantaggi del self attention*

Gli autori indicano varie motivazioni che hanno portato alla scelta del maccanismo attention nel modello, rispetto all'utilizzo di una RNN o convolution NN:

1. Il costo computazionale complessivo è di gran lunga minore, per via dell'assenza di convoluzioni o ricorrenze e per il fatto che le macchine sono ottimizzate per effettuare i calcoli matriciali[22]
2. Dato che vi sono meno calcoli da eseguire sequenzialmente, anche la quantità di dati parallelizzabile è superiore.
3. La terza è la capacità superiore di analizzare le relazioni del meccanismo di Attention, in quanto riesce a cogliere meglio i riferimenti fra le parole, dove gli altri due metodi presentavano più difficoltà.

Di seguito una tabella che riporta i costi computazionali dei vari metodi.

- Con Complessity per Layer viene indicato quante operazioni vengono effettuate per ogni layer.
- Con Sequential Operations viene indicato il numero di equazioni sequenziali eseguite dal metodo.
- Con Maximum Path Lenght viene indicata la lunghezza del collegamento più lungo fra le parole nella frase di input

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. $n$ is the sequence length, $d$ is the representation dimension, $k$ is the kernel size of convolutions and $r$ the size of the neighborhood in restricted self-attention.

| Layer Type | Complexity per Layer | Sequential Operations | Maximum Path Length |
|---|---|---|---|
| Self-Attention | $O(n^2 \cdot d)$ | $O(1)$ | $O(1)$ |
| Recurrent | $O(n \cdot d^2)$ | $O(n)$ | $O(n)$ |
| Convolutional | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(log_k(n))$ |
| Self-Attention (restricted) | $O(r \cdot n \cdot d)$ | $O(1)$ | $O(n/r)$ |

*Tabella 1-1 Costi computazionali*[23]

Nella maggior parte dei casi n<<d, e questo conduce ad una complessità in genere minore del modello Transformers. Anche il Maximum path length è il minore rispetto agli altri modelli.

[22] Attention is all you need; Attentional Neural Network Models | Łukasz Kaiser | Masterclass, https://www.youtube.com/watch?v=rBCqOTEfxvg&t=946s

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017. Pg 6.

La lunghezza inferiore del collegamento fra le parole è propedeutica alla maggior capacità di Transformers di cogliere le relazioni fra le parole

### *1.2.14. Training & datasets utilizzati*

Durante il training del modello le matrici contenenti i parametri del modello vengono inizializzate in maniera random, e vengono allenate tramite un processo di supervised learnig, fornendo al modello un training set provvisto sia di input che di output.

Per la stima dei parametri viene utilizzato un algoritmo di ottimizzazione stocastica chiamato Adam. Si basa sulla minimizzazione del valore atteso di una funzione obiettivo stocastica scalare $E[f(\theta)]$, differenziabile rispetto al vettore di parametri $\theta$.

Tramite il gradiente della funzione $g_t = \nabla_\theta \mathrm{f_t}(\theta)$ , dove t indica il tempo di realizzazione della funzione, vengono aggiornati ad ogni passo i processi exponential moving average $m_t$e $v_t$, dove gli iperparametri $\beta_1, \beta_2 \in [0,1)$ controllano il tasso di aggiornamento degli MA.

I MA sono una stima del primo e del secondo momento del gradiente della funzione obiettivo. Dato che i MA sono inizializzati con valore 0, questa stima sarà spostata verso questo valore soprattutto nei passi iniziali e soprattutto con parametri di controllo $\beta_1, \beta_2 \in [0,1)$ molto grandi (vicino a 1), dato che l'MA impiegherà più passi ad allontanarsi dal valore iniziale. Per questo i MA vengono corretti, conducendo ad una stima non distorta, indicati con $\hat{m}_t$ e $\hat{v}_t$. I parametri vengono aggiornati sulla base di questi valori, passo dopo passo, finché non si raggiunge una convergenza. Di seguito lo pseudocodice dell'algoritmo.

**Algorithm 1:** *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. $g_t^2$ indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With $\beta_1^t$ and $\beta_2^t$ we denote $\beta_1$ and $\beta_2$ to the power $t$.

**Require:** $\alpha$: Stepsize
**Require:** $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
**Require:** $f(\theta)$: Stochastic objective function with parameters $\theta$
**Require:** $\theta_0$: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
**while** $\theta_t$ not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
**end while**
**return** $\theta_t$ (Resulting parameters)

*Figura 1.12 Pseudocodice ADAM* [24]

Gli autori di Transformers utilizzano Adam con i seguenti parametri; $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-9}$, mentre fanno evolvere il tasso di apprendimento $\alpha$ con la seguente formula

$$\alpha = d_{model}^{-0.5} * min(step.num^{-0.5}, step.num * warmup.steps^{-1.5}) \qquad (1.8)$$

Con $warmup.steps = 4000$.

Implementano inoltre alcuni tipi di ***regolarizzazioni:***

- ***residual dropout:*** Prima di essere normalizzato e passato alla sublayer successiva, parte dell'output prodotto dalle sublayer viene scartato, e lo stesso procedimento viene applicato agli algoritmi di embedding e di positional encoding, per entrambe le parti di encoder e decoder. Il tasso di dropout è pari a $P_{drop} = 0.1$. Questo aiuta a prevenire l'overfitting nelle neural network[25]. Il termine dropout si riferisce all'esclusione delle unità (sia nascoste che visibili) nelle neural network.

[24] Ulteriori dettagli possono essere trovati sul loro articolo: Diederik P. Kingma, Jimmy Lei Ba, *Adam: A method for stochastic optimization* 2017
[25] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. *Dropout: a simple way to prevent neural networks from overfitting*. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

- ***Label smoothing:*** Per aumentare le performance, durante la fase di training applicano il metodo del label smoothing alla layer classificatore con un parametro pari a $\epsilon_{ls} = 0.1$. Il metodo si basa sul rimpiazzare la distribuzione originale delle label $q(k|x) = \delta_{\mathrm{k|x}}$ con

$$q'(k|x) = (1 - \epsilon)\delta_{k|x} + \epsilon u(k) \tag{1.9}$$

  Dove u(k) è una qualche distribuzione fissata.[26]

La Loss function utilizzata è un confronto fra la probabilità che restituisce Transformers e quella del vero valore, che può essere espressa tramite una *cross-entropy loss function*

$$H(p, q) = -\Sigma_{x \in \mathrm{X}} p(x) log\big(q(x)\big) \tag{1.10}$$

Dove p(x) è la probabilità calcolata da Transformers mentre q(x) quella vera[27].

I dataset utilizzati per allenare il modello sono:

- ***WMT 2014 English-German:*** costituito da circa 4.5 coppie di frasi
- ***WMT 2014 English-French:*** costituito da circa 36M di coppie

Per diminuire la dimensionalità dei termini da gestire, sono stati creati dei vocabolari di dimensione minore basati su particolari *token* (una rappresentazione alternativa delle parole in input), partendo dalle parole contenute nei due dataset. Questi token verranno poi trasformati i vettori numerici. Per fare questo sono stati utilizzati due metodi:

- ***Byte-pair encoding***[28]: Associato al problema della traduzione *English-German*, che costruisce circa 37.000 token
- ***WordPiece***: Associato al problema della traduzione *English-French*, che conduce ad un vocabolario circa 32.000 token

Essendo condiviso da BERT il metodo WordPiece verrà spiegato in seguito, mentre per Byte-pair encoding si rimanda alla letteratura di riferimento.

[26] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. *Rethinking the inception architecture for computer vision*. CoRR, abs/1512.00567, 2015
[27] Jay Allammar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/
[28] Rico Sennrich and Barry Haddow and Alexandra Birch, *Neural Machine Translation of Rare Words with Subword Units*

## 1.3. Il modello BERT

Sulla base dell'architettura di transformers, è stato sviluppato[29] un ulteriore modello di NLP chiamato Bidirectional Encoder Rappresentation from Transformers. La caratteristica principale di BERT che lo differenzia dagli altri modelli è il fatto che può essere allenato usando un training set senza label, ma semplicemente mascherando alcuni token associati a delle parole nelle frasi del dataset. Questo permette al modello di essere allenato in entrambe le direzioni, mascherando sia i token delle parole all'inizio della frase sia quelli alla fine. Questo porta ad un miglioramento delle performance del modello, ed inoltre permette di fare a meno di training set forniti di label e di utilizzare qualsiasi corpora per allenare il modello.

Per utilizzare i modelli NLP pre-allenati per compiere attività di modellizzazione del linguaggio, vi sono due approcci: il featured based o il fine tuning.[30]

- **Feaured based:** Questo metodo si basa su costruire specifiche architetture per ogni task.
- **Fine tuning:** E' il metodo usato da BERT, e si basa su allenare ulteriormente i parametri che sono stati preallegati nella fase di pretraining per renderli specifici per la task da compiere. Introduce pochi parametri in più rispetto alla fase di pretraining.

Oltre al masking-tokens, BERT viene ulterioremente preallenato tramite il "next sentence prediction", cioè cercando di prevedere se le frasi in input sono connesse o meno.

L'articolo spiega l'importanza del training bidirezionale per i modelli di rappresentazione del linguaggio, e come questo conduca a dei modelli più semplici per compiti di NLP, senza la necessità di costruire architetture complesse e specifiche per ogni task. In molti casi BERT si è rivelato il modello più performante nelle NLP task.

### *1.3.1. Architettura del modello*

Come anticipato il modello riprende l'architettura vista per il modello transformers, tuttavia presenta alcune differenze. La prima è che non vengono considerate le layer del decoder, ma si basa solo sulla parte dell'encoder più l'ultima layer che agisce da classificatore e la softmax per calcolare le probabilità.

[29] Dagli autori dell'articolo Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
[30] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

Sono state prodotte due versioni del modello BERT[31]:

1. **$\mathbf{BERT_{base}}$:** si basa su un encoder formato da 12 layer sequenziali, più la layer finale che funge da classificatore e la layer softmax per il calcolo delle probabilità. L'hidden size del modello ($d_{model}$ in Transformers) è pari a $H = 768$, e il numero di *head* nelle sublayer multi attention (parametro *h* in Transformers) è pari a 12. Questo conduce ad un numero di parametri pari a 110M.
2. **$\mathbf{BERT_{large}}$:** E' una versione molto più grande della base. E' composto da un encoder con 24 layers sequenziali, un hidden size pari a $H = 1024$ e 16 *head*, per un totale di 340M di parametri.

### *1.3.2. WordPiece embeddings*

Similmente a transformers, gli input embeddings vengono gestiti tramite WordPiece[32], un'algoritmo che costruisce un vocabolario di 30.000 Token. Viene costruita una rappresentazione degli input che è in grado di gestire in maniera non ambigua le frasi in entrata, tramite l'utilizzo di due speciali token, cioè:

- [CLS] che si colloca all'inizio della frase prima della prima parola, sta per classificatore.
- [SEP] che si colloca fra una frase ed un'altra nel caso in input vi siano più espressioni (è necessario durante la fase di *"next sentence prediction"* per dividere le frasi)

Una caratteristica del WordPiece embeddings su cui è bene soffermarsi è la capacita di gestire le parole Out Of Vocabulary (OOV). Infatti i problemi di NLP hanno a che fare la maggior parte delle volte con un numero non preciso di vocaboli, mentre i modelli di language processing si basano su un vocabolario fisso dal quale vengono codificate le parole tramite algoritmi di embeddings. Questo è problematico quando si incappa in una parola al di fuori del vocabolario utilizzato (appunto Out Of Vocabulary). Per evitare cioò è stato sviluppato un metodo[33] basato su un algoritmo chiamato WordPiece per gestire la rappresentazione di vocaboli Coreani e Giapponesi, che frequentemente non presentano spazi tra le parole rendendone difficile la codifica.[34]. Si basa sulla costruzione di un nuovo vocabolario con un

[31] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
[32] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc VLe, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. *Google's neural machine translation system: Bridging the gap between human and machine translation.* arXiv preprint arXiv:1609.08144
[33] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc VLe, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. *Google's neural machine translation system: Bridging the gap between human and machine translation.* arXiv preprint arXiv:1609.08144
[34] Schuster, M., and Nakajima, K. *Japanese and Korean voice search*. 2012 IEEE International

numero di token molto più ridotto rispetto al numero massimo di vocaboli, lasciandosi guidare dai termini contenuti in un training set. Le parole contenute nelle frasi del training set vengono spaccate in più subunità, dove il numero di subunità è ottenuto sulla base della massimizzazione della verosimiglianza del language model sul training set[35]. In questo modo si ottiene un vocabolario di pezzi di parole che unite insieme formano i termini del vocabolario originale, dove le subunità sono indicate con un simbolo speciale (## in BERT). Questo approccio amplia la capacità del modello di riconoscere le parole e riduce la possibilità di incappare in termini OOV. Inoltre una differenza del nuovo metodo è che i token di base sono stati ulteriormente ridotti di numero in modo da avere una quantità di token gestibile, sulla base del tipo di lingua a cui sono associati. Cosa più importante, è stato inserito uno speciale token a cui sono ricondotti tutti i caratteri rari che il modello eventualmente non riesce ad interpretare. Questi token vengono poi considerati come parole vere e proprie e rappresentate in vettore numerici con i metodi visti precedentemente. Ogni subunità ha un codice ID, che come vedremo dovrà essere previsto quando alcuni token verranno mascherati durante il pretraining del modello. Di seguito un'immagine che spiega come vengono rappresentati gli input nel modello BERT, dati dalla somma dei token embeddings ricavati dal modello WordPiece, dei segment embeddings che indicano la frase a cui appartengono le parole, e dei positional embeddigs che catturano l'informazione della posizione dei token all'interno della frase.

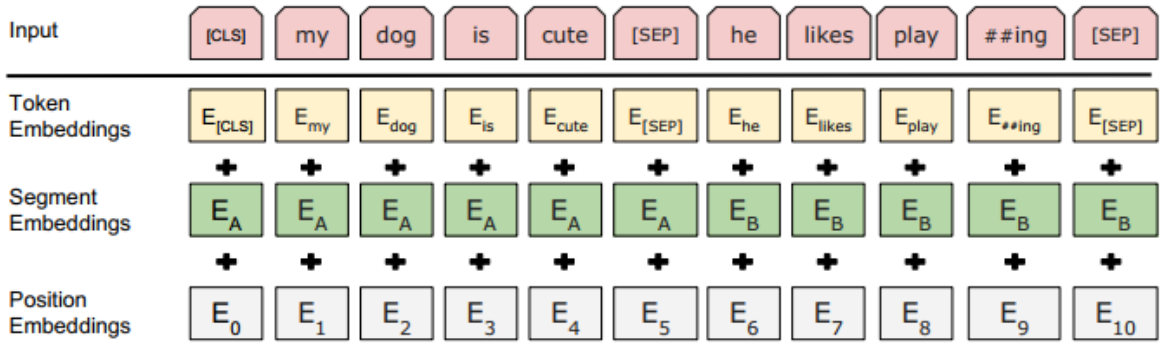


*Figura 1.13 Rappresentazione input BERT* [36]

Come vediamo anche dall'immagine, all'inizio di ogni frase viene inserito un token speciale [CLS]. Nel caso di problemi di classificazione, il vettore associato a questo token risultante

Conference on Acoustics, Speech and Signal Processing (2012).

[35] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc VLe, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. *Google's neural machine translation system: Bridging the gap between human and machine translation.* arXiv preprint arXiv:1609.08144

[36] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

dall'output dell'ultima layer dell'encoding ($C \in R^H$ nell'articolo) verrà passato al classificatore finale del modello, e sulla base del quale si genererà la classificazione finale. Questo vettore è usato come un aggregatore che è rappresentativo della sequenza analizzata e quindi delle informazioni provenienti dagli altri token, che poi così aggregate verranno mandate al classificatore.

Come anticipato, BERT viene pre-allenato attraverso due unsupervised task diverse, *Masked LM e Next sentence prediction.*

### *1.3.3. Pretraining BERT: Masked-LM*

Rappresenta la caratteristica principale del modello e a cui vengono ricondotte le sue superiori performance. Chiaramente fare del training in maniera bidirezionale (utilizzando un dataset senza label) condizionandosi sia sulle parole a destra che a sinistra della frase rappresenta un modo migliore per allenare il modello, tuttavia questo non è sempre applicabile. Nei modelli che seguono un determinato flusso dei token in input ed in output questo non è possibile per via del fatto che i dati vengono passati in modo sequenziale in base ad un determinato pattern. Per quanto riguarda i modelli bidirezionali come BERT i dati entrano in parallelo senza seguire un determinato flusso, e dato che sono interconnessi tra loro in ogni momento (per via dell'architettura già spiegata ereditata da Transformers), si incappa nel problema secondo il quale i dati da prevedere vengono visti durante la fase di training. E' un problema simile alla fase di training nella parte del decoding, dove il modello aveva informazioni sui token che poi avrebbe dovuto prevedere. Per risolvere questo problema, gli autori usano un metodo chiamato masked LM[37], anche chiamato *Cloze* task. Il 15% dei token presenti in input viene mascherato, attribuendogli un valore arbitrario, ed il modello avrà il compito di prevedere l'id di questi token. Per evitare mismatching fra il dataset di pre-training e quello fine-tuning[38], i token scelti vengono rimpiazzati con altri valori nel seguente modo:

- L'80% delle volte il token scelto viene rimpiazzato con il simbolo speciale [MASK].
- Il 10% delle volte con un token random.
- Il 10% delle volte con il suo stesso token, cioè il valore non viene modificato.

A questo punto, il vettore associato al token che è stato mascherato risultante dall'ultima layer dell'encoding ($T_i \in R^H$ nell'articolo) viene utilizzato per prevedere il valore dell'ID del vero

[37] Wilson L Taylor. 1953. *Cloze procedure: A new tool for measuring readability*. Journalism Bulletin,30(4):415–433.
[38] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

token in quella posizione. Nel caso in cui il token sia stato rimpiazzato con se stesso, ci si aspetta che il modello preveda lo stesso token. Il problema del mismatching nasce dal fatto che il token [MASK] non comparirà mai nella fase di fine tuning, in quanto creato appositamente per rimpiazzare i token scelti per essere mascherati. Seguendo questo approccio, il modello non risente di questa sostituzione (sulla base delle performance osservate). Inoltre in questo modo viene affrontato un doppio problema, cioè anche quello di identificare quali token sono stati effettivamente mascherati. Il modello è cosi costretto non solo a prestare attenzione al token mascherato, ma a tutti i token della frase.

### *1.3.4. Pretraining BERT: Next Sentence Prediction*

È la seconda fase processo di pretraining del modello BERT. Nasce dalla necessità di rendere il modello più adatto ad affrontare problematiche che riguardano l'apprendimento e la previsione delle relazioni che intercorrono fra le diverse frasi in un input.

Il modello viene allenato su un dataset provvisto di label, dove sono presenti varie coppie di frasi, l'espressione A e B. Il 50% delle volte la B è l'espressione che effettivamente segue la frase A nel testo, mentre nella restante metà dei casi è scelta a caso dal corpus. Nel primo caso verrà associata una label [IsNext], mentre nel secondo caso la label sarà [NotNext]. Essendo questo un problema di classificazione, il vettore *C* associato al token [CLS] verrà usato come aggregatore degli altri token e passato alla layer classificatore per produrre le previsioni.

Nonostante si utilizzi un dataset con delle label per questo pre-training, questo è facilmente costruibile partendo da qualsiasi corpus monolingua.

Durante il pretraining di BERT, queste due fasi (MLM & NSP) vengono combinate insieme con l'obiettivo di minimizzare la funzione di perdita cumulata, costituita dalla somma della verosimiglianza media dei token mascherati (per la parte MLM) e la verosimiglianza media dei token di classificazione del next sentence prediction[39].

Di seguito un'immagine che rappresenta il flusso dei token all'interno del modello, per tutti e due gli stadi del pre-training. Nell'immagine è assente il classificatore finale

[39] Rani Horev BERT Explained: State of the art language model for NLP, *https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270*

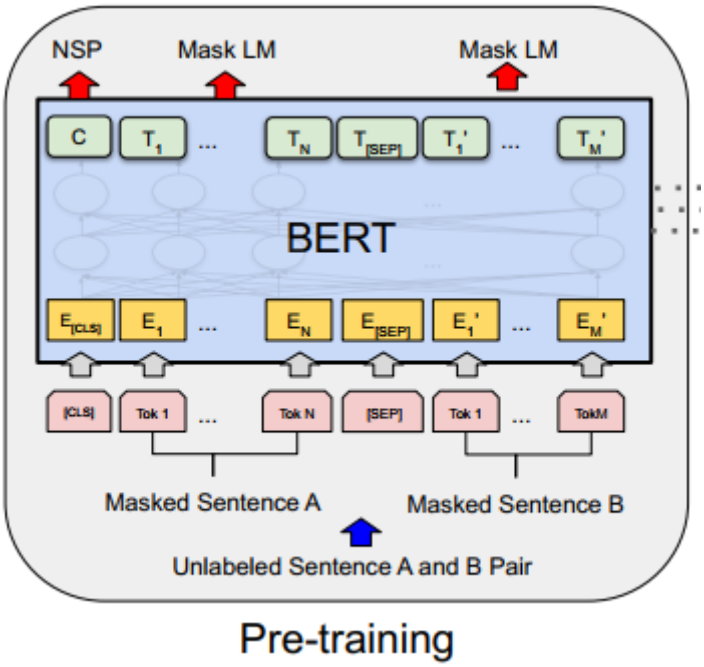


*Figura 1.14 Pretraining BERT* [40]

### *1.3.5. Dataset utilizzati*

Gli autori utilizzano per il pretraining del modello 2 dataset[41]:

- **BookCorpus:** Composto da 11.038 libri non coperti da copyright. Sono stati inclusi solo quelli che presentano più di 20K parole, in modo da filtrare possibili testi contenenti poca informazione. Il dataset è composto da libri appartenenti a 16 generi diversi[42] e contiene circa 800M parole.
- **English Wikipedia:** La versione inglese di Wikipedia, contente circa 2500M parole. Gli autori hanno considerato solo il testo, depurato dalle tabelle, dalle liste e dai titoli.

Successivamente alla pubblicazione dell'articolo, è stata sviluppata anche una versione multilingua di BERT, che comprende le 100 lingue al mondo con le più grandi Wikipedia. Il pre-training per questa versione è stato effettuato proprio su tutte le parole contenute nelle

[40] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
[41] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
[42]Yukun Zhu , Ryan Kiros, Richard Zemel Ruslan Salakhutdinov Raquel Urtasun Antonio Torralba Sanja Fidler *Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books, 2015*

pagine Wikipedia, per ogni lingua diversa[43]. La versione multilingua presenta la stessa composizione di $BERT_{base}$.

### *1.3.6. Fine tuning & dettagli implementativi*

Il meccanismo di Self - Attention su cui si basa BERT (ereditato da Transformers) permette a BERT di gestire molti compiti tipici del NLP, come *Question Answering, Sequence Tagging, Sentiment Analysis, Entailment,* dove i primi due sono *token-level*, (cioè che riguardano una sola parola che viene restituita in output), mentre gli altri sono *sentence level* (riguardano tutta la frase, e verrà restituita una label riferita a questa).

Il fine-tuning è gestito semplicemente fornendo a BERT i dati di input e di output di una determinata *task*, e partendo dai parametri ottenuti con il pretraining questi vengono ritoccati in maniera da renderli efficienti per il compito specifico. I dati di input sono passati al modello alla stessa maniera del pretraining, mentre i vettori risultanti dall'ultima layer dell'encoding vengono passati direttamente alla layer finale, che sarà al livello dei token per le *task token level* (specifica per i vari token, come nel caso del Masking-LM), oppure una layer classificatore se il problema è *sentence level*.

Appare chiaro che la parte di fine-tuning è molto più veloce e computazionalmente meno onerosa della precedente, in quanto i parametri devono essere solo ritoccati e se ne aggiungono pochi di nuovi. Infatti per ogni task basta implementare una specifica output layer (al livello del token [CLS] se si tratta di un problema di classificazione o al livello del token se si tratta di *task token level*) che restituisce gli output specifici per quel compito, con un'aggiunta minima di parametri da ritoccare. Per quanto riguarda i problemi di classificazione, se vi sono presenti più di due label in output, è necessario aggiungere più layer all'interno del classificatore. Di seguito un'immagine rappresentativa del processo di fine tuning, dove le task a) e b) sono *sentence level*, c) e d) sono *token level.*

[43] *BERT multilingual* https://github.com/google-research/bert/blob/master/multilingual.md

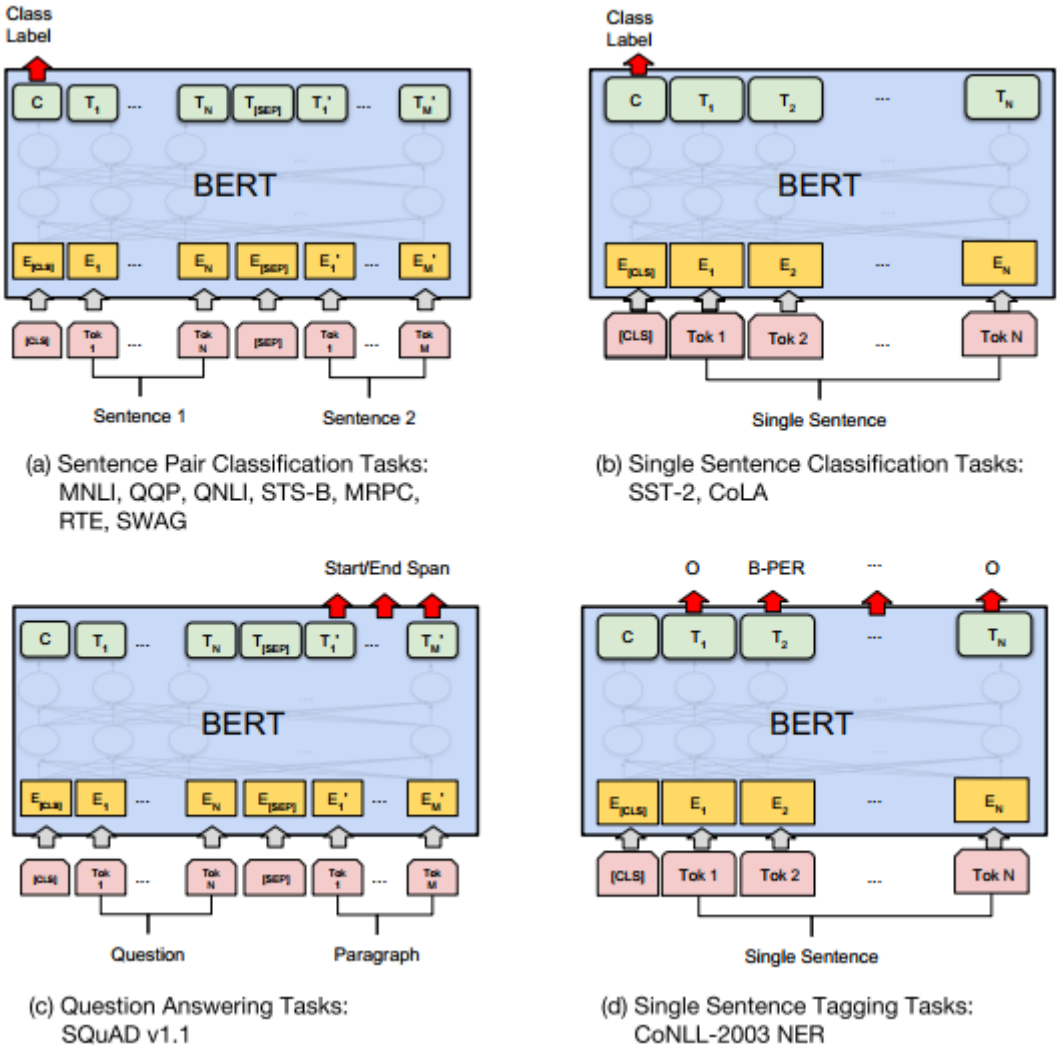


*Figura 1.15 Fine tuning BERT* [44]

Per allenare il modello è stato usato un ottimizzatore Adam con $\alpha = 0.0001, \beta_1 = 0.9, \beta_1 = 0.999$, $L2\ weight\ decay = 0.01$[45], learning rate warmup sui primi 10.000 steps, decadimento lineare del learning rate ($\alpha$) e $P_{drop} = 0.1$ su tutte le layer. Utilizzano una GeLU[46] activation al posto della ReLU nelle FFNN.

[44] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

[45] Si tratta di aggiungere alla funzione di costo delle reti neurali un termine di regolarizzazione del tipo $\lambda/(2n) * \Sigma_1^n \omega_i^2$ dove $\lambda \in [0,1]$ . Con $\lambda = 0$ non si ha nessuna regolarizzazione, con $\lambda = 1$ massima.

[46] Gaussian Error Linear Unit, $GeLU(x) = x * P(X \leq x) = x * \Phi(x)$ ; Dan Hendrycks, Kevin Gimpel, *Gaussian Error Linear Unit*

Per il fine tuning tutti i parametri rimangono invariati, ad eccezione del learning rate di Adam, dove la scelta ottima è ricaduta su questi tre valori (a seconda della task specifica) $\alpha = 5e^{-5}, \alpha = 3e^{-5}, \alpha = 2e^{-5}$[47].

## 1.4. Il modello FinBERT

### *1.4.1. Introduzione*

Dopo aver richiamato i modelli su cui si basa, parliamo di FinBERT[48]. E' nato per la necessità di analizzare corpora prettamente finanziari, che sono caratterizzati da un lessico specifico in relazione al contesto finanziario. Questo li rende difficili da trattare con i vari modelli NLP. Per questo l'autore sviluppa FinBERT, un modello basato su BERT che si poggia sull'idea che un ulteriore pre-training del modello su corpora finanziari possa aumentare le performance per le task su testi di questo genere. Come vedremo, questo approccio conduce effettivamente a superare i risultati ottenuti da altri modelli non specifici per questo contesto, anche basandosi su pre-trainig e fine tuning con dataset piccoli.

Nei mercati finanziari, i prezzi degli asset si muovono sulla base delle informazioni che circolano sui sottostanti a cui si riferiscono (aziende, stati, oggetti ecc…). E' perciò di vitale importanza avere accesso ed elaborare in poco tempo tutte le informazioni relative agli asset in cui si sta investendo. Avendo ormai a disposizione in tempi brevi una enorme quantità di informazioni finanziarie sotto forma di articoli, news, messaggi social … rimane il problema di analizzare in maniera veloce tutte queste fonti, in modo da non arrivare tardi sui mercati.

Vi sono quindi due ordini di problemi da affrontare[49]

1) Trovare modelli che siano in grado di analizzare efficacemente queste informazioni. Le reti neurali costruite per fare classificazione hanno infatti bisogno di una quantità notevole di dati etichettati per essere allenate, che scarseggiano in contesti finanziari e richiedono un team di esperti per produrli.

[47] Ulteriori parametri come il batch size ed il numero di epoche possono essere trovati sul loro articolo,
Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
[48] Sviluppato dall'autore dell'articolo Dogu Tan Araci, *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019*
[49] Dogu Tan Araci, *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019*

2) I modelli NLP per la sentiment analysis perdono di performance se sono allenati su corpora generali, per via del già citato problema del lessico specifico che caratterizza il contesto.

Per provare a risolvere queste problematiche, l'autore sfrutta la capacità del modello BERT di essere preallenato su un dataset senza etichette. L'idea è quella di effettuare un pretraining su un corpora prettamente finanziario, in modo che il modello possa capire come rappresentare le informazioni semantiche di questo particolare contesto. Questi parametri così inizializzati verranno poi ulteriormente ritoccati tramite il processo di fine tuning, utilizzando un dataset con etichette. Questo approccio risolve il problema della scarsità dei *labeled data* finanziari, per la caratteristica del processo di fine tuning di non necessitare di molti dati, in quanto le relazioni fra le parole sono state precedentemente imparate tramite il processo di pretraining.

Lo scopo preciso per il quale è stato sviluppato il modello è la sentiment analysis dei testi finanziari. Precisamente si tratta di attribuire alle singole frasi un particolare livello di sentimento, che può essere positivo, negativo o neutrale. Una differenza rispetto alla sentiment analysis classica è lo scopo per cui viene effettuata, cioè quello di prevedere l'andamento dei mercati finanziari, obiettivo che per avere senso necessita di tempi brevi di completamento. In passato sono stati applicati modelli di machine learning che si basavano sul conteggio delle parole ("word counting") per estrarre informazioni dal testo, piuttosto che modelli di deep learning. Mentre con i primi vi è la difficolta di catturare le relazioni fra le parole[50], i secondi necessitano di molti dati per essere allenati in quanto presentano un gran numero di parametri. L'articolo si pone come obiettivo quello di trovare una soluzione a questi problemi, tramite lo sviluppo di un nuovo modello, FinBERT.

### *1.4.2. Caratteristiche del modello FinBERT*

Come già introdotto precedentemente FinBERT si basa sul modello BERT, e da questo ne eredita l'architettura. Nulla infatti viene cambiato da questo punto di vista. Quello che viene modificato è l'implementazione del modello, che passa per vari punti[51]:

- ***Ulteriore pretraining:*** Il modello BERT viene ulteriormente allenato su corpora con linguaggio prettamente finanziario, per cogliere meglio le relazioni fra le frasi di questo specifico contesto. Il pretraining aggiuntivo è stato effettuato utilizzando sia un corpus

[50] Questo perché si basano sulla somma di determinati score attribuiti alle varie parole che compongono la frase analizzata. Tuttavia in questo modo non si colgono le relazioni fra le parole della frase
[51] Dogu Tan Araci, *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019*

finanziario molto grande, sia utilizzando solo le frasi provenienti dal dataset sul quale verrà poi perfezionato il problema di classificazione. Questo per capire se ha qualche valenza perfezionare il pretraining direttamente sul linguaggio target.

- ***Stratagie per il corretto Fine tuning:*** Per evitare che il processo di fine tuning cancelli l'informazione acquisita durante il pretraining, stravolgendo completamente il valore dei parametri, vengono adottate delle strategie dette *training strategies*[52]*:*
    - *Slanted triangular learning rates:* Il learning rate $\alpha$ cresce linearmente sino ad un certo punto, per poi decrescere sempre linearmente
    - *Discriminative fine-tuning:* Si tratta di usare bassi $\alpha$ nelle layer inferiori del modello. Ci si basa sull'assunzione che le layer superiori contengano informazioni riguardanti l'attuale problema di classificazione (per il quale stiamo facendo il fine-tuning) mentre quelle più basse si riferiscano alle relazioni semantiche del linguaggio, imparate nel pretrainig. Per questo vengono allenate "di meno", diminuendo il tasso di apprendimento. Impostando un discriminative rate $\theta$, $\alpha$ evolve nel seguente modo: $\alpha_{l-1} = \theta\alpha_l$ dove $l \in [1,12]$ indica una delle 12 layer del modello.
    - *Gradual freezing:* Si allena il modello con tutte le layer *"ghiacciate"* a parte quella del classificatore. Durante il processo di training, si iniziano ad allenare mano a mano anche le altre layer partendo da quella superiore. In questo modo le layer più basse vengono allenate di meno.

Inoltre viene implementato un modello sia per la classificazione sia per la regressione, dove in quest'ultimo caso si utilizza per il fine-tuning un dataset con varabili risposta continue e la loss function viene cambiata da cross entropy a mean squared error.

### *1.4.3. Datasets*

Vengono utilizzati 3 tipi di dataset[53]:

- ***TRC2-financial:*** E' utilizzato per il pretraining ulteriore di FinBERT. Consiste in 46.143 articoli pubblicati dalla rivista Reuterstra il 2008 e il 2010, per un totale di 29M di parole.

[52] Queste strategie sono state proposte da Howard and Ruder nel loro articolo del 2018: *Universal Language Model Finetuning for Text Classifcation. (jan 2018).* arXiv:1801.06146 http://arxiv.org/abs/ 1801.06146

[53] Dogu Tan Araci, *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019*

- ***Financial PhraseBank[54]:*** E' il dataset principale dell'articolo utilizzato per il fine-tuning del modello per la sentence classification. Si tratta di 4845 frasi inglesi dal contenuto prettamente finanziario selezionate in maniera casuale dal LexisNexis database, che sono state etichettate da 16 persone con conoscenze in finanza. Ad essi è stato chiesto di classificare le frasi con 3 livelli (Positive, Negative e Neutral), e di esprimere il livello di accordo fra di loro riguardo all'etichetta scelta per ciascuna frase. Le frasi sono state classificate riferendosi all'impatto che queste potrebbero avere sui rendimenti degli asset sui mercati finanziari a cui si riferiscono. Il dataset è stato splittato in due set dove quello di training include 3101 frasi, allenando il modello usando 10-fold cross validation. Il dataset presenta uno sbilanciamento verso la classe neutrale (neutral=59.4%, positive=28.1%, negative=12.4%)
- ***FiQA Sentiment[55]:*** E' composto da 1.174 tweet e titoli finanziari ognuno associato al proprio livello di sentimento (sentiment score). La variabile risposta è quindi continua, $\in [-1,1]$, dove 1 indica la positività massima. Anche qui è stata applicata una 10-fold cross validation per valutare il modello.

### *1.4.4. Metriche utilizzate & dettagli implementativi*

Le performance ottenute da FinBERT sono state confrontate con quelle ottenute da altri modelli NLP[56] (baseline model):

- ***LSTM classifiers[57]:*** Si tratta di due classificatori bidirezionali LSTM, entrambi con un $hidden_{size} = 128$ per le layer a parte l'ultima che presenta una $hidden_{size} = 256$, dove uno utilizza dei GloVe embeddings[58], mentre l'altro usa Elmo embeddings[59]
- ***ULMFit[60]:*** Il modello viene utilizzato pre-allenato, ma viene ulteriormente allenato sul dataset TRC2-financial e sottoposto a un processo di fine-tuning specifico per la task di

[54] Il dataset può essere trovato qui https://www.researchgate.net/publication/251231364_FinancialPhraseBank-v10.
[55] I dati possono essere scaricati qui: *https://sites.google.com/view/fqa/home*.
[56] Dogu Tan Araci, *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019*
[57] *Long Short-Term Memory* è una Recurrent Neural Network che permette di apprendere le relazioni fra le parole di una frase tramite un meccanismo di cancellazione ed aggiornamento ("*forget and updates").* I token che costituiscono la frase vengono rappresentati utilizzano algoritmi di rappresentazione delle parole come GloVE o ELMo.
[58] GLoVe è un modello per la rappresentazione delle parole, costituito da una log regressione bivariata perfezionata su una matrice di co-occorenza fra le parole di un grande corpora. Permette di rappresentare i token su uno spazio d-dimensionale minore, ma non tiene conto del contesto in cui sono inserite i token.
[59] ELMo è un modello di rappresentazione delle parole che tiene conto della frase in cui risiedono i token. Si tratta di un biderection language model con layer LSTM multiple, e restituisce un vettore rappresentativo per ogni token che tiene conto delle relazioni fra di loro.
[60] ULMFit è un *transefering learning model* utilizzato per *down-stream NLP tasks*, che fa uso delle tecniche di pre-training. E' basato su AWD-LSTM, una versione regolarizzata delle LSTM.

interesse sul Financial PhraseBank dataset. Viene aggiunta una layer classificatore al modello pre-allenato.

Le metriche utlizzate sono state:

- ***Accuracy:***

$$ACC = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(\mathrm{y_i} = \hat{y}_i) \tag{1.11}$$

dove $y_i$ è l'i-esimo valore mentre $\hat{y}_i$ è la sua previsione

- ***Cross-entropy loss function:***

$$H(p,q) = -\Sigma_{x\in \mathrm{X}} p(x)\ log\big(q(x)\big) \tag{1.12}$$

ponderata per la radice quadrata dell'inverso della frequenza di valori predetti dalla layer a cui la loss è riferita.

- ***Macro F1 average:***

$$F1 = \frac{tp}{\left(tp + \frac{1}{2}(fp + tp)\right)} \tag{1.13}$$

dove tp sta per truepositve e fp per falsepositive. La metrica è calcolata per ogni classe, e quella totale è una media delle metriche.

- $\boldsymbol{R^2}$

$$R^2 = 1 - \frac{\Sigma_{i=1}^{\mathrm{n}}(y_i - \hat{y}_i)^2}{\Sigma_{i=1}^{\mathrm{n}}(y_i - \bar{y}_i)^2} \tag{1.14}$$

- ***MSE:***

$$MSE = \frac{1}{n}\Sigma_{i=1}^{\mathrm{n}}(y_i - \hat{y}_i)^2 \tag{1.15}$$

Il modello è stato implementato con un $P_{drop} = 0.1, \alpha = 2e^{-5}$ per la parte di pretraining, mentre per la parte di fine tuning è stato allenato partendo da solo la layer classificatore libera, (sbloccando le altre nel corso del training) con $\theta = 0.85$.

Di seguito un'immagine esplicativa del processo di training di FinBERT.

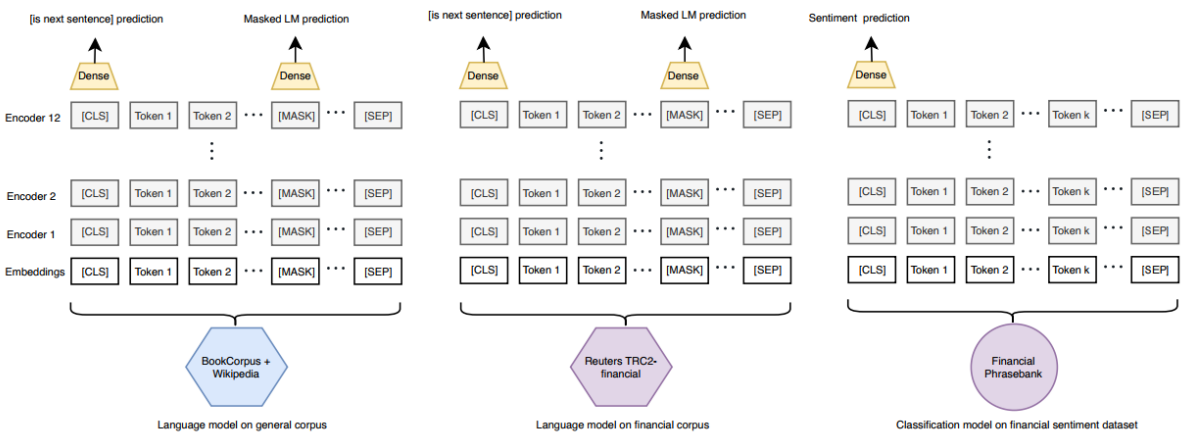


*Figura 1.16 Processo training FinBERT*[61]

### *1.4.5. Risultati FinBERT*

FinBERT for classification è stato testato su Financial Phares Bank, messo a confronto con i risultati ottenuti sia con i modelli che costituiscono la baseline sia con i modelli utilizzati in letteratura per modellare questo dataset. I risultati sono presentati sia considerando l'intero dataset sia considerando solo le etichette dove gli appuntatori erano completamente d'accordo fra di loro. Si nota che in entrambi i casi, FinBERT archivia delle nuove performance ottime, con un grande distacco fra gli altri modelli ad eccezione di ULMFit, che archivia comunque delle performance molto simili. Si nota inoltre che i modelli basati esclusivamente su tecniche di machine learning faticano a performare bene (LPS e HSC), nonché l'importanza dell'input embeddings e del fine tuning. Infatti LSTM che è sprovvisto sia di pretraining che di una rappresentazione degli input che tenga conto del contesto performa peggio. Viene superato da LSTM con ELMo (che utilizza una contest rappresentation), che viene superato a sua volta da ULMFit, provvisto di entrambe le tecniche. ULMFit supera anche FinSSLX, un modello di ML fornito sia di una semplificazione del testo sia di tecniche di pretraining. Di seguito una tabella riepilogativa dei risultati.

[61] Dogu Tan Araci, *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019*

**Table 2: Experimental Results on the Financial PhraseBank dataset**

| | All data | | | Data with 100% agreement | | |
|---|---|---|---|---|---|---|
| Model | Loss | Accuracy | F1 Score | Loss | Accuracy | F1 Score |
| LSTM | 0.81 | 0.71 | 0.64 | 0.57 | 0.81 | 0.74 |
| LSTM with ELMo | 0.72 | 0.75 | 0.7 | 0.50 | 0.84 | 0.77 |
| ULMFit | 0.41 | 0.83 | 0.79 | 0.20 | 0.93 | 0.91 |
| LPS | - | 0.71 | 0.71 | - | 0.79 | 0.80 |
| HSC | - | 0.71 | 0.76 | - | 0.83 | 0.86 |
| FinSSLX | - | - | - | - | 0.91 | 0.88 |
| FinBERT | **0.37** | **0.86** | **0.84** | **0.13** | **0.97** | **0.95** |

**Bold face** indicates best result in the corresponding metric. LPS [17], HSC [8] and FinSSLX [15] results are taken from their respective papers. For LPS and HSC, overall accuracy is not reported on the papers. We calculated them using recall scores reported for different classes. For the models implemented by us, we report 10-fold cross validation results.

*Tabella 1-2 Risultati Finbert classification* [62]

Per testare il modello applicato a problemi di regressione è stato utilizzato FiQA, confrontando i risultati di FinBERT con i modelli utilizzati in letteratura per analizzare questo dataset. Anche qui FinBERT performa meglio degli altri modelli, sulla base delle metriche $R^2$ e MSE. Di seguito la tabella riepilogativa.

[62] Dogu Tan Araci, *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019*

**Table 3: Experimental Results on FiQA Sentiment Dataset**

| Model | MSE | $R^2$ |
|---|---|---|
| Yang et. al. (2018) | 0.08 | 0.40 |
| Piao and Breslin (2018) | 0.09 | 0.41 |
| FinBERT | **0.07** | **0.55** |

**Bold face** indicated best result in corresponding metric. Yang et. al. (2018) [31] and Piao and Breslin (2018) [24] report results on the official test set. Since we don't have access to that set our MSE, and $R^2$ are calculated with 10-Fold cross validation.

*Tabella 1-3 Risultati FinBERT regression* [63]

### *1.4.6. Effetto dell'ulteriore pretraining su un dataset finanziario*

Per testare l'efficacia dell'ulteriore pretraining del modello BERT su un corpora prettamente finanziario, sono stati costruiti 3 modelli ognuno dei quali valutati sullo stesso test set per il problema di classificazione (FinancialPhraseBank):

- Vanilla-BERT: Modello senza ulteriore pre-training.
- FinBERT-Task: Modello pre-allenato direttamente sul classification training set.
- FinBERT-Domain: Modello pre-allenato su un corpora finanziario molto grande (TRC2-fnancial).

Dai risultati non si nota un grande incremento delle performance attribuibile ad un ulteriore pretraining del modello. Secondo l'autore, questo è dovuto al fatto che già il modello senza pretrainig archivia delle buone performance, e non ci sia più molto margine di miglioramento. Del resto, l'accuracy del VanillaBERT su FinancialPhaseBank con 100% di livello di accordo fra gli etichettatori è del 0.96. Altre spiegazioni potrebbero essere che la distribuzione dei token nel TRC2-financial sia comunque diversa rispetto a quella presente in FinancialPhaseBank, o che effettivamente il pretraining non incida nell'incremento delle performance finali del modello. Di seguito la tabella riassuntiva dei risultati.

[63] Dogu Tan Araci, *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019*

**Table 4: Performance with different pre-training strategies**

| Model | Loss | Accuracy | F1 Score |
|---|---|---|---|
| Vanilla BERT | 0.38 | 0.85 | 0.84 |
| FinBERT-task | 0.39 | 0.86 | **0.85** |
| FinBERT-domain | **0.37** | **0.86** | 0.84 |

**Bold face** indicates best result in the corresponding metric. Results are reported on 10-fold cross validation.

*Tabella 1-4 Risultati FinBERT con diversi pretraining* [64]

### *1.4.7. Effetto delle strategie per il corretto fine tuning*

Per valutare l'effetto delle diverse strategie di fine-tuning, FinBERT è stato implementato in diverse maniere, ognuna di queste valutata sullo stesso test set per il problema di classificazione. Le implementazioni sono:

- ***NA:*** Modello implementato senza strategie.
- ***STL:*** E' stata applicata la solo strategia *slanted triangular learning rate.*
- ***STL+GU:*** Sono state applicate sia la strategia *slanted triangular learning rate* sia la *gradual unfreezing.*
- ***STL+DFT:*** Sono state applicate sia la strategia *slanted triangular learning rate* sia la *discriminative fine-tuning.*
- ***All Three:*** Sono state applicate tutte e 3 le strategie.

Dai risultati si nota che utilizzare tutte e tre le strategie porta ad un miglioramento in termini di cross entropy loss e di Accuracy, mentre il valore ottimo per la metrica F1 Score si ottiene con le strategie STL+GU applicate insieme. Una cosa interessante da notare è che applicare insieme le strategie STL e DFT conduce ad un peggioramento delle performance in termini di Accuracy e F1 Score rispetto ad applicare solo STL o anche senza applicare nulla. Di seguito la tabella riepilogativa dei risultati.

[64] Dogu Tan Araci, *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019*

**Table 5: Performance with different fine-tuning strategies**

| Strategy | Loss | Accuracy | F1 Score |
|---|---|---|---|
| None | 0.48 | 0.83 | 0.83 |
| STL | 0.40 | 0.81 | 0.82 |
| STL + GU | 0.40 | 0.86 | **0.86** |
| STL + DFT | 0.42 | 0.79 | 0.79 |
| All three | **0.37** | **0.86** | 0.84 |

**Bold face** indicates best result in the corresponding metric. Results are reported on 10-fold cross validation. STL: slanted triangular learning rates, GU: gradual unfreezing, DFT: discriminative fine-tuning.

*Tabella 1-5 Risultati con diverse strategie di fine tuning* [65]

Inoltre un altro modo in cui è possibile vedere l'impatto delle strategie è quello di osservare l'andamento della funzione di loss nel validation set rispetto all'aumentare delle epoche[66]. Si nota che con tutte e tre le strategie applicate, la funzione di loss è molto più stabile.

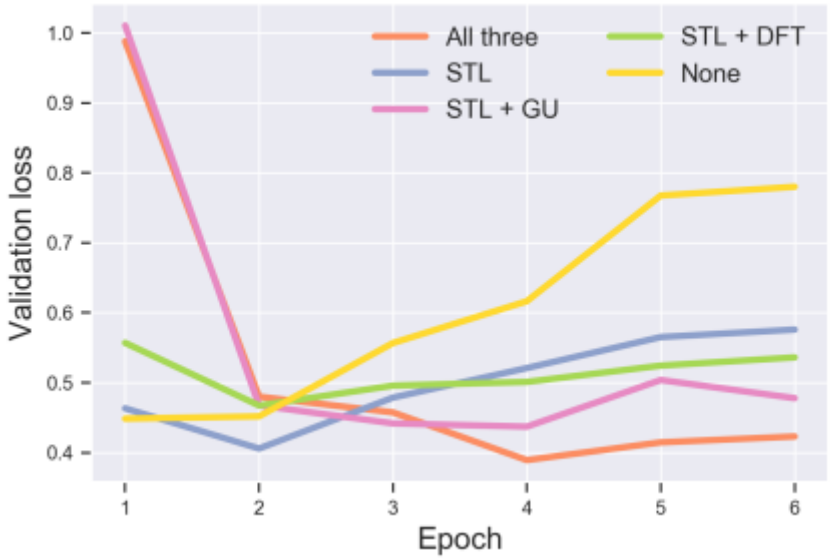


*Figura 1.17 Andamento loss function* [67]

[65] Dogu Tan Araci, FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019
[66] Indica il numero delle volte in cui l'intero training set è usato per aggiornare i parametri del modello tramite l'ottimizzatore. Un altro parametro è il **batch size**, che indica la dimensione del set, sottoinsieme del training set, usato dall'ottimizzatore per completare un solo step, cioè per aggiornare di una solo volta i parametri.
[67] Dogu Tan Araci, FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019

### *1.4.8. Identificazione layer migliore*

L' autore investiga anche quale sia la layer che permette di raccogliere più informazioni fra i token, e conseguentemente conduca ad un miglioramento maggiore delle performance delle previsioni. Fa questo mettendo la layer classificatore non sempre dopo la 12esima layer dell'encoder (avendo in entrata sempre il token classificatore [CLS]), ma ogni volta dopo una layer diversa. Successivamente misura le performance di queste varie implementazioni con le tre metriche e sul dataset FinancialPhaseBank. Si nota che la layer che produce un miglioramento maggiore delle metriche è la layer 6. Questo può indicare due cose secondo l'autore:

- Più il modello diventa grande più aumenta le sue performance.
- Le layer più basse catturano le relazioni semantiche più profonde, che faticano poi a sviluppare per il problema di classificazione.

Di seguito la tabella riassuntiva dei risultati:

**Table 6: Performance on different encoder layers used for classification**

| Layer for classification | Loss | Accuracy | F1 Score |
|---|---|---|---|
| Layer-1 | 0.65 | 0.76 | 0.77 |
| Layer-2 | 0.54 | 0.78 | 0.78 |
| Layer-3 | 0.52 | 0.76 | 0.77 |
| Layer-4 | 0.48 | 0.80 | 0.77 |
| Layer-5 | 0.52 | 0.80 | 0.80 |
| Layer-6 | 0.45 | 0.82 | 0.82 |
| Layer-7 | 0.43 | 0.82 | 0.83 |
| Layer-8 | 0.44 | 0.83 | 0.81 |
| Layer-9 | 0.41 | 0.84 | 0.82 |
| Layer-10 | 0.42 | 0.83 | 0.82 |
| Layer-11 | 0.38 | 0.84 | 0.83 |
| Layer-12 | 0.37 | 0.86 | 0.84 |
| All layers - mean | 0.41 | 0.84 | 0.84 |

*Tabella 1-6 Risultati per layers* [68]

[68] Dogu Tan Araci, FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019

### *1.4.9. Allenare il modello solo su un subset di layers*

Data la dimensione notevole di FinBERT l'autore si chiede se sia necessario allenare il modello intero oppure solo un numero inferiore di layers, per quanto riguarda il fine tuning. Per questo prova ad allenare solo le layers più in alto nel modello (a partire dalla layer classificatore) misurandone le performance, aumentando ogni volta il numero di strati sui cui performa il fine tuning. Si nota che dopo la layer 9 le performance rimangono sostanzialmente uguali, a fronte di un aumento del costo in termini di tempo. Di seguito la tabella riepilogativa

**Table 7: Performance on starting training from different layers**

| First layer unfreezed | Loss | Accuracy | Training time |
|---|---|---|---|
| Embeddings layer | 0.37 | 0.86 | 332s |
| Layer-1 | 0.39 | 0.83 | 302s |
| Layer-2 | 0.39 | 0.83 | 291s |
| Layer-3 | 0.38 | 0.83 | 272s |
| Layer-4 | 0.38 | 0.82 | 250s |
| Layer-5 | 0.40 | 0.83 | 240s |
| Layer-6 | 0.40 | 0.81 | 220s |
| Layer-7 | 0.39 | 0.82 | 205s |
| Layer-8 | 0.39 | 0.84 | 188s |
| Layer-9 | 0.39 | 0.84 | 172s |
| Layer-10 | 0.41 | 0.84 | 158s |
| Layer-11 | 0.45 | 0.82 | 144s |
| Layer-12 | 0.47 | 0.81 | 133s |
| Classification layer | 1.04 | 0.52 | 119s |

*Tabella 1-7 Risultati per subset di layers* [69]

### *1.4.10. Matrice di confusione*

Infine, riporto di seguito la matrice di confusione delle previsioni del modello FinBERT sul dataset di classificazione FinancialPhaseBank. Gli errori maggiori vengono compiuti nel prevedere le classi neutrali previste come positive, e positive previste come neutrali.

[69] Dogu Tan Araci, FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019

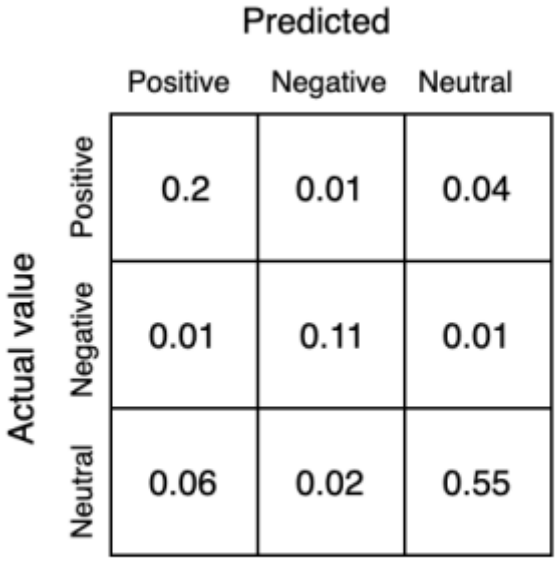


**Figure 4: Confusion matrix**

*Figura 1.18 Matrice di confusione*[70]

## 1.5. Modello AR + eGARCH

Dopo aver attribuito un sentiment score ai tweet estratti, sono stati provati alcuni modelli previsi per cercare di prevedere i rendimenti ed il segno degli asset finanziari scelti. Sono stati utilizzati vari modelli ed il primo presentato è una combinazione fra un modello AR e un modello exponential GARCH.

### *1.5.1. Definizione modello e stima dei parametri*

Si tratta di un modello autoregressivo di ordine uno per spiegare la media della serie dei rendimenti, con errori che seguono un modello eGARCH (0,1), dove la varianza condizionata di questi è spiegata solo dai propri valori ritardati. L'AR(1) è caratterizzato da regressori esterni e nessuna intercetta. La parte stocastica del modello segue una Generalised Error Distribution di media 0 e varianza 1. In formule:

$$(1\text{-}\phi_1 \mathrm{B})(Y_t\text{-}\boldsymbol{\alpha}^T \boldsymbol{X_t}) = \varepsilon_t$$

$$\varepsilon_t = \xi_t \sigma_t$$

$$ln(\sigma_t^2) = \omega + \beta_1 ln(\sigma_{t-1}^2) \qquad (1.16)$$

[70] Dogu Tan Araci, FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019

dove $\xi_t \sim ged(0,1,r)\ i.i.d$

$$f(\xi_t, r) = \frac{re^{-\frac{1}{2}\left(\left|\frac{\xi_t-\mu}{\lambda\sigma}\right|\right)^r}}{\sigma\lambda 2^{\frac{(r+1)}{r}}\Gamma\left(\frac{1}{r}\right)} \tag{1.18}$$

Con $\mu = 0, \sigma = 1, \lambda = \left(\frac{\Gamma\left(\frac{1}{r}\right)}{2\Gamma\left(\frac{3}{r}\right)}\right)^{\frac{1}{2}}, \Gamma(x) = \int_0^{+\infty} t^{x-1}\, e^{-t} dt, r > 0$

Dove con $\boldsymbol{\alpha^T X_t}$ vengono indicati rispettivamente il vettore dei parametri e il vettore dei regressori esterni (al tempo *t*) a cui sono associati i parametri.

Dato che il modello scelto segue un exponential GARCH, non vi è la necessità di imporre vincoli sulla varianza affinché essa sia positiva.

Dato che l'unica componente stocastica del modello segue una legge di distribuzione ged, la distribuzione della probabilita condizionata di $Y_t$ all'informazione al tempo precedente $\gamma_{t-1}$ sarà:

$$Y_t|\gamma_{t-1} \sim ged\ (\mu_t, \sigma_t, r) \tag{1.19}$$

Dove $\mu_t = \boldsymbol{\alpha^T X_t} + \phi_1 \mathrm{B}(Y_t\text{-}\boldsymbol{\alpha^T X_t})$ e $\sigma_t = \left(e^{\omega+\beta_1 ln\left(\sigma_{t-1}^2\right)}\right)^{1/2}$

Questa distribuzione verrà utilizzata per costruire la funzione di verosimiglianza

$$L(\boldsymbol{\theta}) \propto f_{\boldsymbol{\theta}}(Y_t|\gamma_{t-1}) * f_{\boldsymbol{\theta}}(Y_{t-1}|\gamma_{t-2}) \dots\dots\dots * f_{\boldsymbol{\theta}}(Y_2|\gamma_1) * f_{\boldsymbol{\theta}}(Y_1)$$

Dove tuttavia non conosciamo la funzione di densita non condizionata di $Y_t$. Pertanto si ipotizzerà di conoscere $\mu_1$ e $\sigma_1$ in modo da poter calcolare la densità condizionata di $Y_1$, basando la massimizzazione della verosimiglianza condizionata, appunto, a questi parametri iniziali. La verosimiglianza condizionata è definita come:

$$L^c(\boldsymbol{\theta}) \propto f_{\boldsymbol{\theta}}(Y_t|\gamma_{t-1}) * f_{\boldsymbol{\theta}}(Y_{t-1}|\gamma_{t-2}) \dots\dots\dots * f_{\boldsymbol{\theta}}(Y_2|\gamma_1) * f_{\boldsymbol{\theta}}(Y_1|\gamma_0) \tag{1.20}$$

Che, applicando la trasformazione in logaritmi diventa

$$l^c(\boldsymbol{\theta}) = ln\left(L^c(\boldsymbol{\theta})\right) \propto ln\left(f_{\boldsymbol{\theta}}(Y_t|\gamma_{t-1})\right) + ln\left(f_{\boldsymbol{\theta}}(Y_{t-1}|\gamma_{t-2})\right) \dots\dots\dots + ln\left(f_{\boldsymbol{\theta}}(Y_1|\gamma_0)\right)$$

$$l^c(\boldsymbol{\theta}) = \Sigma_{i=1}^{t} ln\left(f_{\boldsymbol{\theta}}(Y_i|\gamma_{i-1})\right) \tag{1.21}$$

La massimizzazione di questa funzione rispetto al vettore $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \phi_1, \omega, \beta_1)$ restituisce le stime di massima verosimiglianza dei parametri del modello, $\widehat{\boldsymbol{\theta}}$.

La varianza asintotica degli stimatori di massima verosimiglianza dei parametri è data dall'inverso della matrice di informazione attesa di Fisher

$$I(\boldsymbol{\theta}) = -lim_{n\to\infty}\frac{1}{n}E\left[\sum_{i=1}^{t}\frac{\partial^2 lnf_{\boldsymbol{\theta}}(Y_t|\gamma_{t-1})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta_0}}\right] \tag{1.22}$$

Che può essere stimata tramite

$$I_n(\widehat{\boldsymbol{\theta}}) = -\frac{1}{n}\sum_{i=1}^{t}\frac{\partial^2 lnf_{\boldsymbol{\theta}}(Y_t|\gamma_{t-1})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T}\bigg|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}} \tag{1.23}$$

Dove $\boldsymbol{\theta_0}$ è il vero valore del parametro $\boldsymbol{\theta}$

Dato che asintoticamete gli stimatori di massima verosimiglianza si distribuiscono come:

$$\widehat{\boldsymbol{\theta}}\sim N\left(\boldsymbol{\theta_0}, \left(nI(\boldsymbol{\theta})\right)^{-1}\right) \tag{1.24}$$

con $I(\boldsymbol{\theta})$ che può essere stimato tramite $I_n(\widehat{\boldsymbol{\theta}})$, la statistica test

$$\frac{\left(\hat{\theta}_i - \theta_{0,i}\right)}{\sqrt{v_{ii}}} \tag{1.25}$$

può essere usata per testare l'ipotesi nulla che $\hat{\theta}_i = \theta_{0,i}$ e quindi la significatività dei paramentri.[71]

$v_{ii}$ indica l'i-esimo elemento sulla diagonale principale di $(n\boldsymbol{I_n}(\widehat{\boldsymbol{\theta}}))^{-1}$, mentre $\hat{\theta}_i$ e $\theta_{0,i}$ l'i-esima componente dei rispettivi vettori. La statistica test si distribuisce asintoticamente come una distribuzione normale con media 0 e varianza 1.

### *1.5.2. Stime di quasi verosimiglianza*

Inoltre le stime vengono calcolate in maniera robusta, rispetto alla cattiva specificazione del modello, tramite il metodo della quasi massima verosimiglianza[72]. Queste consistono nella stima di quel valore $\dot{\boldsymbol{\theta}}$ che rende minima la quantità definita come *criterio di informazione di Kullback-leibler*

[71] Matteo Maria Pelagatti, *Time Series Modelling with Unobserved Components*
[72] Matteo Maria Pelagatti, *Statistica dei mercati monetari e finanziari*, novembre 2017

$$KLIC(g:f) = E_g[ln\ g(X)] - E_g[ln\ f(X|\boldsymbol{\theta})] \tag{1.26}$$

Dove g(X) è la vera distribuzione del modello parametrico ed $f(X|\ \boldsymbol{\theta})$ è una funzione di densità relativa alla famiglia di funzione di ripartizione $\{F(x|\ \boldsymbol{\theta}): \boldsymbol{\theta} \in \Theta\}$, con $\Theta$ lo spazio parametrico di $\theta$. X è un vettore di variabili casuali, $X\_t;\ t = 1, \ldots n$. Il valore che rende minimo il criterio è il valore che rende massimo il secondo valore atteso. Uno stimatore di $E_g[ln\ f(X|\boldsymbol{\theta})]$ è

$$l_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{t=1}^{n} lnf(X_t|\boldsymbol{\theta}) \tag{1.27}$$

cioè la funzione di logverosimiglianza del modello con specificazione potenzialmente sbagliata chiamata *quasi verosimiglianza.* La massimizzazione di questa funzione restituisce il valore $\dot{\boldsymbol{\theta}}$, chiamato *stima di quasi verosimiglianza.* Quello che cambia è la distribuzione asintotica degli stimatori di quasi massima verosimiglianza, cioè

$$\dot{\boldsymbol{\theta}} \sim N\left(\boldsymbol{\theta_0}, \frac{1}{n}\boldsymbol{I}(\boldsymbol{\theta})^{-1}\boldsymbol{J}\boldsymbol{I}(\boldsymbol{\theta})^{-1}\right) \tag{1.28}$$

dove

$$\boldsymbol{J} = lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{t} E\left[\left[\frac{\partial lnf_{\boldsymbol{\theta}}(Y_t|\gamma_{t-1})}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta_0}}\right]\left[\frac{\partial lnf_{\boldsymbol{\theta}}(Y_t|\gamma_{t-1})}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta_0}}\right]^T\right] \tag{1.29}$$

che può essere stimata con

$$\hat{\boldsymbol{J}}_{\boldsymbol{n}} = \frac{1}{n}\sum_{i=1}^{t}\left[\frac{\partial lnf_{\boldsymbol{\theta}}(Y_t|\gamma_{t-1})}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\dot{\boldsymbol{\theta}}}\right]\left[\frac{\partial lnf_{\boldsymbol{\theta}}(Y_t|\gamma_{t-1})}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\dot{\boldsymbol{\theta}}}\right]^T \tag{1.30}$$

Se il modello è correttamente identificato, $\boldsymbol{J} = \boldsymbol{I}(\boldsymbol{\theta})$**,** e si ritorna al caso di stime di massima verosimiglianza. Questa nuova distribuzione può essere usata per costruire test sulla significatività dei parametri delle stime robuste[73].

Di seguito vengono spiegati alcuni test per la corretta diagnostica del modello.

[73] Matteo Maria Pelagatti, *Time Series Modelling with Unobserved Components*

### *1.5.3. Weighted Ljung-Box test on standardized and standardized squared residuals*

Si tratta di un test per verificare la presenza di eventuale autocorrelazione fra i residui. L'ipotesi nulla è

$$H_0: \rho_1 = \rho_2 = \ldots\ldots = \rho_k = 0$$

Contro l'alternativa che almeno un parametro sia significativamente diverso da 0.

La statistica test è

$$Q(K) = n \sum_{k=1}^{K} \frac{(n+2)}{(n-k)} [\rho(k)]^2 \qquad (1.31)$$

dove $\rho(k)$ è la funzione di autocorrelazione al ritardo $k$.

La statistica Q si distribuisce come una $X^2$ con $K - (p + q + 1)$ g.d.l, dove p e q indicano l'ordine della parte AR e MA. Il test viene applicato a diversi lag sui residui.[74]

### *1.5.4 Weighted ARCH LM tests*

E' un test che verifica l'ipotesi che i residui del modello non presentino eteroschedaticità condizionata autoregressiva, cioè

$$H_0: \alpha_1 = \alpha_2 = \ldots\ldots = \alpha_q = 0$$

Con $\alpha_i$ parametri di un ARCH(q). Invece di costruire un ARCH(q) e testare i paramentri, viene strutturato un altro test chiamato ARCH LM Test, spiegato di seguito.

Il test viene strutturato come segue:

1. I residui del modello $\varepsilon_t^2$ si regrediscono sui ritardi $\varepsilon_{t-1}^2, \ldots\ldots \varepsilon_{t-q}^2$ più una costante.
2. Si calcola $nR^2$ dove $R^2$ è il coefficiente di determinazione della regressione sopra ed n la numerosità campionaria

La statistica $nR^2 \sim \mathrm{X}^2(q)$, dalla quale si possono confrontare valori della statistica con i quantili critici.[75]

[74] Biancamaria Zavanella, Appunti di lezione Introduzione alle serie storiche M CLAMSES
[75] Matteo Maria Pelagatti, *Statistica dei mercati monetari e finanziari*, novembre 2017

### *1.5.4. Nyblom stability test*

E' un test che verifica se le relazioni fra le variabili cambiano nel corso degli istanti, cioè se la serie storica presenta un cambio strutturale nei dati. L'ipotesi nulla è che la varianza dei parametri nel corso del tempo sia uguale a 0, mentre l'alternativa è che sia positiva, e di conseguenza che i parametri cambino il loro valore all'aumentare degli istanti.[76]

### *1.5.5. Sign Bias tests*

Si tratta di tests per verificare l'ipotesi di asimmetria delle serie. Essi si basano sul calcolo delle stime $\hat{\sigma}_t^2$ e $\hat{\xi}_t = \varepsilon_t/\sigma_t^2$ tramite un GARCH(1,1), e sulla regressione di $\hat{\xi}_t^2$ su

$$\hat{\xi}_t^2 = z^T\gamma + I_{-\infty,0}(\varepsilon_{t-1})\lambda + \varepsilon_t \tag{1.32}$$

$$\hat{\xi}_t^2 = z^T\gamma + I_{-\infty,0}(\varepsilon_{t-1})\varepsilon_{t-1}\lambda + \varepsilon_t \tag{1,33}$$

$$\hat{\xi}_t^2 = z^T\gamma + I_{0,+\infty}(\varepsilon_{t-1})\varepsilon_{t-1}\lambda + \varepsilon_t \tag{1.34}$$

$$\hat{\xi}_t^2 = z^T\gamma + I_{-\infty,0}(\varepsilon_{t-1})\lambda_1 + I_{-\infty,0}(\varepsilon_{t-1})\varepsilon_{t-1}\lambda_2 + I_{0,+\infty}(\varepsilon_{t-1})\varepsilon_{t-1}\lambda_3 + \varepsilon_t \tag{1.35}$$

Dove $z = (1, \varepsilon_{t-1}^2, \hat{\sigma}_{t-1}^2)^T$.

Per ognuna di queste regressioni si vanno a testare con un test t la significatività dei parametri di regressione associati alle variabili indicatori, verificando se uno shock informativo è significativo nello spiegare i residui quadrati standardizzati del modello GARCH. La prima testa la significatività di shock sia positivo che negativo, non ponderandolo per la sua intensità (sign bias test). La seconda (sign bias negative test) e la terza (sign bias positive test) testano la significatività rispettivamente di uno shock negativo e positivo all'informazione, ponderandolo per la propria intensita. L'ultima regressione (join sign bias) testa la significatività di tutti e tre questi shock.[77]

### *1.5.6. Adjusted Pearson Goodness-of-Fit test*

Si tratta di un test che confronta i quantili della distribuzione empirica dei residui standardizzati del modello con quelli teorici della distribuzione scelta. L'ipotesi nulla testa la perfetta uguaglianza fra questi quantili[78].

[76] Logical Errors, *Garch – Modeling Conditional Variance & Useful Diagnostic Tests*, https://logicalerrors.wordpress.com/2017/08/14/garch-modeling-conditional-variance-useful-diagnostic-tests/
[77] Matteo Maria Pelagatti, Statistica dei mercati monetari e finanziari, novembre 2017
[78] Logical Errors, Garch – Modeling Conditional Variance & Useful Diagnostic Tests, https://logicalerrors.wordpress.com/2017/08/14/garch-modeling-conditional-variance-useful-diagnostic-tests/

## 1.6. Least Absoulute Shrinkage and Selection Operator (LASSO)

Uno dei metodi utilizzati nel corso del lavoro, sia come modello predittivo sia come "variable selector" è la regressione lasso. Questa fa parte dei "Shrinkage Methods", ovvero dei metodi che aggiungono una penalità alla funzione da minimizzare rispetto ai parametri della regressione, ottenendo delle stime distorte verso lo 0.

Dal Teorema di Gauss-Markov sappiamo che, per un modello $\boldsymbol{y} = \boldsymbol{X\beta_0} + \boldsymbol{\varepsilon}$ con X matrice non stocastica a rango pieno, $E[\boldsymbol{\varepsilon}] = 0\ Var[\boldsymbol{\varepsilon}] = \sigma^2\boldsymbol{I}$, il miglior stimatore lineare dei parametri tra quelli non distorti è lo stimatore OLS

$$\widehat{\boldsymbol{\beta}}^{OLS} = argmin_{\boldsymbol{\beta}\in R^p}\left|\left|\boldsymbol{y} - \boldsymbol{X\beta}\right|\right|_2^2 = (\boldsymbol{X^TX})^{-1}\boldsymbol{X^Ty} \qquad (\mathbf{1.36})$$

nel senso che è quello che presenta un errore $E\left[(\widehat{\boldsymbol{\beta}}^{\boldsymbol{OLS}} - \boldsymbol{\beta^0})(\widehat{\boldsymbol{\beta}}^{\boldsymbol{OLS}} - \boldsymbol{\beta^0})^{\boldsymbol{T}}\right] = Var(\widehat{\boldsymbol{\beta}}^{\boldsymbol{OLS}})$ minore rispetto a qualsiasi altro stimatore non distorto.

Tuttavia introducendo un po' di distorsione, la quantità espressa prima per un qualsiasi stimatore diventa

$$E\left[(\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta^0})(\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta^0})^{\boldsymbol{T}}\right] = Var(\widetilde{\boldsymbol{\beta}}) + E[(\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta^0})]\boldsymbol{E}[(\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta^0})]^{\boldsymbol{T}} \qquad (1.37)$$

che può essere inferiore rispetto all'errore commesso dallo stimatore dei minimi quadrati, seppur producendo uno stimatore distorto.[79]

La regressione Lasso si basa proprio su questo. Introduce della distorsione nei parametri, stimandoli come quelli che minimizzano la seguente funzione obiettivo "ristretta":

$$(\hat{\mu}_\lambda^L, \widehat{\boldsymbol{\beta}}_\lambda^L) = argmin_{(\boldsymbol{\mu},\boldsymbol{\beta})\in RxR^p}\frac{1}{2n}\left|\left|\boldsymbol{y} - \mu\mathbf{1}\ -\ \boldsymbol{X\beta}\right|\right|_2^2 + \lambda\left|\left|\boldsymbol{\beta}\right|\right|_{\mathbf{1}};\ \lambda \geq \mathbf{0} \qquad (1.38)$$

Dove il parametro $\lambda$ viene scelto come quello che minimizza l'RMSE del modello, stimato per esempio tramite *cross-validation*. Il risultato sarà dei parametri stimati più piccoli, distorti verso il valore 0.

La penalità introdotta è pari alla norma 1 del vettore dei parametri $\boldsymbol{\beta}$, dove la formula che definisce la norma è $\left|\left|\boldsymbol{x}\right|\right|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$. Una particolarità della regressione Lasso è che essa è *"sparse"*, nel senso stima alcuni $\beta_j = 0$, basandosi sull'assunzione che solo una parte dei regressori è significativa, cioè presenta un parametro diverso da 0. Questo permette alla

[79] Aldo Solari, Appunti di Lezione: La regressione Ridge, Data Mining M CLAMSES

Lasso di effettuare anche una Best Subset Selection delle variabili esplicative disponibili nel modello[80]. Di seguito un'immagine che evidenzia l'operazione di contrazione dei parametri a 0 da parte del Lasso.

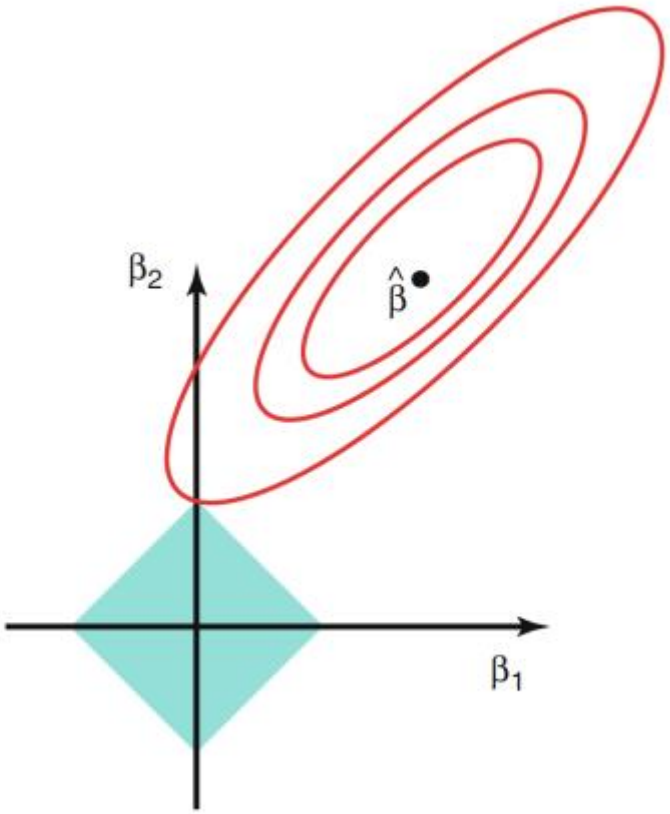


*Figura 1.19 Stima coefficienti Lasso* [81]

## 1.7. La regressione logistica

Nell'ambito dell'analisi del segno, per gestire la natura dicotomica della nuova variabile risposta, è stato utilizzata una regressione logistica regolarizzata tramite la stessa penalità utilizzata nella Lasso.

Infatti, nella regressione lineare classica vengono fatte delle ipotesi che sono:

- Variabili casuali risposta $y_i \sim N(\mu, \sigma^2)\ ;\ i = 1, \dots . n$ e di conseguenza termini di errore $\varepsilon_i \sim N(0, \sigma^2)\ ;\ i = 1, \dots . n$
- Linearità fra $y_i$ e la sua media $E[y_i] = \mu = \beta_0^T X_i$.

[80] Aldo Solari, Appunti di Lezione: La regressione Lasso, Data Mining M CLAMSES
[81] Aldo Solari, Appunti di Lezione: La regressione Lasso, Data Mining M CLAMSES

Sia l'ipotesi di normalità che di linearità sono un problema quando si tratta con variabili discrete (in questo caso, che presentano solo due modalità), in quanto:

- Non possono distribuirsi secondo una distribuzione normale, avendo questa supporto continuo, e di conseguenza nemmeno gli errori del modello potranno seguire questa distribuzione.
- $E[y_i]$ indica una probabilità, e modellandola con una regressione lineare non vi è nessuna garanzia che le previsioni giacciano nell'intervallo [0,1].

Uno dei modelli indicati in questo caso, ed applicato nel corso di questo lavoro, è una regressione logistica, definita da una di queste due formule:

$$\boldsymbol{Y} = \frac{e^{\boldsymbol{\beta^T X}}}{\left(\mathbf{1_n} + e^{\boldsymbol{\beta^T X}}\right)} + \boldsymbol{\varepsilon}$$

$$\boldsymbol{p} = \boldsymbol{E}[\boldsymbol{Y}] = \frac{e^{\boldsymbol{\beta^T X}}}{\left(\mathbf{1_n} + e^{\boldsymbol{\beta^T X}}\right)} = \Lambda(\boldsymbol{\beta^T X})^{82} \qquad (\mathbf{1.39})$$

dove $\boldsymbol{Y}, \boldsymbol{\varepsilon}, \boldsymbol{p} \in \boldsymbol{R^n}, \boldsymbol{\beta} \in \boldsymbol{R^q}, \boldsymbol{X} \in \boldsymbol{R^{nxq}}$ sono rispettivamente il vettore delle variabili risposta, degli errori, delle medie, dei parametri e la matrice fissata dei regressori. Le $y_i$ seguono una legge di distribuzione binomiale $bin(1, p)$

$$y_i \sim p_i^y (1 - p_i)^{1-y_i} \qquad (1.40)$$

Che verrà utilizzata per costruire la funzione di verosimiglianza

$$L(\boldsymbol{\beta}|\boldsymbol{Y}, \boldsymbol{X}) = \prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} p_i^{y_i}(1 - p_i)^{1-y_i} = \prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} \Lambda(\boldsymbol{\beta^T X_i})^{y_i}\left(1 - \Lambda(\boldsymbol{\beta^T X_i})\right)^{1-y_i} \qquad (\mathbf{1.41})$$

In logaritmi

$$l(\boldsymbol{\beta}|\boldsymbol{Y}, \boldsymbol{X}) = ln\left(L(\boldsymbol{\beta}|\boldsymbol{Y}, \boldsymbol{X})\right) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{y_i ln}\left(\Lambda(\boldsymbol{\beta^T X_i})\right) + (1 - y_i) ln\left(1 - \Lambda(\boldsymbol{\beta^T X_i})\right)$$

$$= \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{y_i \beta^T X_i} - ln\left(1 - \mathrm{e}^{\boldsymbol{\beta^T X_i}}\right)^{83} \qquad (1.42)$$

[82] La funzione Λ applicata ad un vettore $\boldsymbol{X_i}$ è definita nel seguente modo: $\Lambda(\boldsymbol{\beta^T X_i}) = \frac{e^{\boldsymbol{\beta^T X_i}}}{\left(1+e^{\boldsymbol{\beta^T X_i}}\right)}$

[83] Matteo Manera Microeconometria Appunti di lezione

La massimizzazione della precedente rispetto al vettore di parametri $\boldsymbol{\beta^T}$ restituisce le stime di massima verosimiglianza per i parametri della regressione logit.

Applicando alla funzione di massima verosimiglianza la stessa penalizzazione applicata alla regressione lasso e massimizzando rispetto al vettore di parametri $\boldsymbol{\beta}$, otteniamo le stime dei parametri distorte verso lo 0.

$$(\widehat{\beta_0}, \widehat{\boldsymbol{\beta_1}})_\lambda^{LL} = argmax_{(\beta_0, \boldsymbol{\beta_1}) \in RxR^{p-1}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} [\boldsymbol{y_i}(\beta_0 + \boldsymbol{\beta_1}^T \boldsymbol{X_i}) - ln\left(1 - e^{(\beta_0 + \boldsymbol{\beta_1}^T \boldsymbol{X_i})}\right)] + \lambda \sum_{j=1}^{q-1} |\beta_{1,j}| \quad (1.43)$$

Dove $\widehat{\boldsymbol{\beta}}_\lambda^{LL} = (\hat{\beta}_0, \widehat{\boldsymbol{\beta_1}})_\lambda^{LL}$.[84]

Oltre ad ottenere le stime dei parametri, questa regressione performa in automatico una variable selection dei regressori del modello, per i motivi espressi durante la spiegazione della lasso. Anche qui il parametro $\lambda$ viene scelto come quello che minimizza l'RMSE del modello, calcolato tramite *cross-validation.*

Sia per il caso della lasso che per il caso della regressione logistica le stime sono ottenuto tramite metodi numerici di minimizzazione o massimizzazione delle funzioni obiettivo. In questo lavoro è stato usato il pacchetto glmnet implementato su R per ottimizare le funzioni, che utilizza rispettivamente un coordinate descent algorithm e un "proximal Newton" algorithm.[85]

## 1.8. Best subset selection whit AIC-based stopping rule

Oltre alla lasso, la variable selection è stata effettuata tramite un metodo backward basato sul criterio di informazione di Akaike

$$AIC = -2loglikelihood(\hat{\beta}, \hat{\sigma}^2) + 2p \quad (1.44)$$

L'algoritmo implementato è cosi definito: Si consideri una matrice di regressori $X \in R^{nxp}$

1. Si imposta $S_p$ come il modello con tutti i regressori e $S_0$ il modello nullo, $k = p$, $n = 1$, e si calcola $AIC(S_p)$
2. Si stimano tutti i modelli con $k = p - n$ regressori, e si tiene quello che presenta l'AIC maggiore, chiamandolo $S_k$

[84] Hastie, Tibshirani, Friedmann, *The Elements of Statistical Learning*, 2019
[85] Trevor Hastie Junyang Qian Kenneth Tay, *An Introduction to glmnet*

3. Si valuta se $AIC(S_k) > AIC(S_{k+1})$, dove $S_{k+1}$ è il modello migliore con $k+1$ regressori. Se la condizione è vera, si prosegue ripartendo dal punto 2 aggiornando $n = n + 1$, altrimenti ci si ferma e si tiene il modello $S_{k+1}$.

L'algoritmo è computazionalmente accettabile, ma non indaga se vi possono essere ulteriore migliorie dopo che la condizione di stop viene raggiunta.

## 1.9. Support Vector Machines

Le support vector machine sono un metodo di machine learning nato come generalizzazione dei linear decision boundaries, sviluppato prima solo per i problemi di classificazione e poi generalizzato anche ai problemi di regressione. Si basano sull'idea di trovare degli iperpiani ottimi per separare i dati, sia che essi siano perfettamente separabili sia che non lo siano. Presentano un'estensione non lineare nel caso in cui i dati non siano separabili con iperpiani lineari. Presentano caratteristiche interessanti come [86]

- Sono basati su un modello teorico.
- Hanno garanzie teoriche sulle loro performance.
- Non presentano minimi locali.
- Non soffrono del problema della dimensionalità.

### *1.9.1. Support Vector Classification*

Consideriamo il caso in cui abbiamo n coppie di valori $(x_i, y_i)$ ; $i = 1, \ldots . n$ con $x_i \in R^p$ e $y_i \in [-1,1]$ e definiamo un iperpiano con

$$[\boldsymbol{x}: f(\boldsymbol{x}) = \boldsymbol{x^T\beta} + \beta_0 = 0] \qquad (1.45)$$

dove $\boldsymbol{\beta}$ è un vettore unitario $\left|\left|\boldsymbol{\beta}\right|\right| = \mathbf{1}$. Una funzione di classificazione può essere $G(\boldsymbol{x}) = sign[\boldsymbol{x^T\beta} + \beta_0]$. Di seguito una rappresentazione del problema nel caso in cui $x_i \in R^2$

[86] Antonio Candelieri, Appunti di Lezione, Lezione 7 , A.A 2020/2021

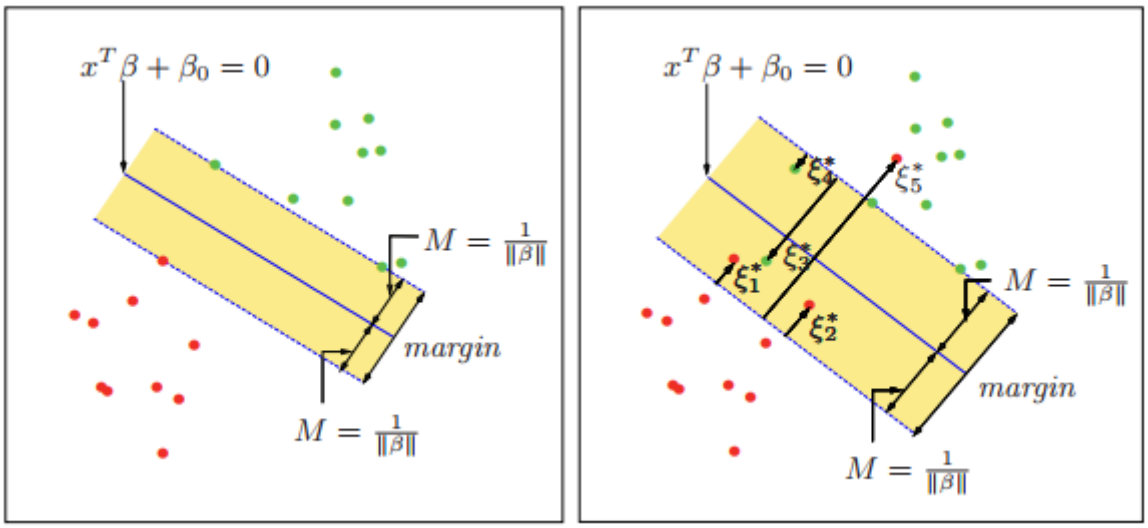


*Figura 1.20 Rappresentazione del problema* [87]

A sinistra abbiamo il caso con dati perfettamente separabili mentre a destra vi è il caso con dati non separabili. La SVM cerca dei parametri dell'iperpiano in modo da massimizzare la distanza perpendicolare fra l'iperpiano stesso e tutti punti del training set, dove questa distanza è chiamata margine. Il problema è quindi completamente definito dai punti più vicini all'iperpiano, che nella figura giacciono vicino alle due linee tratteggiate. Questi punti sono chiamati Support vectors, sono i più difficili da classificare e condizionano direttamente la posizione dell'iperpiano. Sono i punti più importanti del training set perché se rimossi possono cambiare la soluzione del problema.

Partendo dal caso perfettamente separabile, possiamo trovare una funzione $f(x)$ definita come $f(\boldsymbol{x}) = \boldsymbol{x^T\beta} + \beta_0$ tale che la distanza fra l'iperpiano e ogni punto $\boldsymbol{x_i}$, calcolata da $y_i f(\boldsymbol{x_i})$ sia sempre positiva. La quantità $y_i f(\boldsymbol{x_i})$ definisce il margine che il punto $\boldsymbol{x_i}$ ha con l'iperpiano. Lo scopo è quello di trovare i parametri $\boldsymbol{\beta}$ che massimizzano questa quantità. Si viene quindi a creare un problema di massimo vincolato, cioè

$$\max_{\boldsymbol{\beta},\beta\_0,||\boldsymbol{\beta}||=1} M$$

$$sub y_i(\boldsymbol{x_i^T\beta} + \beta_0) \geq M; i = 1,2,\dots\dots n \qquad (1.46)$$

Possiamo eliminare la condizione di $||\boldsymbol{\beta}|| = 1$, normalizzando il vettore $\boldsymbol{\beta}$ per la sua norma 1

$$\max_{\boldsymbol{\beta},\beta\_0,} M$$

[87] Hastie, Tibshirani, Friedmann, *The Elements of Statistical Learning*, 2019

$$sub\,\frac{1}{||\boldsymbol{\beta}||}y_i(\boldsymbol{x_i^T\beta}+\beta_0)\geq M; i=1,2,\dots\dots n \tag{1.47}$$

Imponendo che la norma sia uguale a $||\boldsymbol{\beta}||=\frac{1}{M}$, il tutto diventa

$$\max_{\boldsymbol{\beta},\beta\_0,}\frac{1}{||\boldsymbol{\beta}||}$$

$$sub\;\; y_i(\boldsymbol{x_i^T\beta}+\beta_0)\geq 1; i=1,2,\dots\dots n$$

Che è equivalente a

$$\min_{\boldsymbol{\beta},\beta\_0,}||\boldsymbol{\beta}||$$

$$suby_i(\boldsymbol{x_i^T\beta}+\beta_0)\geq 1; i=1,2,\dots\dots n \tag{1.48}$$

Questo è il criterio con cui si calcolano le SVM per il caso con dati separabili.

Per trattare il caso con dati non separabili, dobbiamo rilassare i vincoli sui margini, permettendo a qualche punto di cadere nella zona sbagliata dell'iperpiano (riquadro a destra della figura). Matematicamente si traduce che alcuni punti possono avere distanze negative con l'iperpiano, cioè $y_i(\boldsymbol{x_i^T\beta}+\beta_0)<0$ . I vincoli del problema diventano

$$y_i(\boldsymbol{x_i^T\beta}+\beta_0)\geq M(1\;-\;\xi_i)\forall i;\xi_i\geq 0;\sum_{i=1}^{n}\xi_i\leq K \tag{1.49}$$

dove K è una costante, e l'obiettivo del problema rimane sempre l'ottimizzazione di M. Le varie $\xi_i,\forall i$ misurano di quanto alcuni punti presentano con l'iperpiano una distanza inferiore a quella ottima M, misurata in termini relativi $M\xi_i$. L'errore di classificazione sorge quindi quando $\xi_i>1$, di conseguenza il vincolo $\sum_{i=1}^{n}\xi_i\leq K$, ha la funzione di limitare gli errori di classificazione ad un ammontare totale pari a K (nel dataset di training) [88]

Il problema in questo caso diventa.

$$\min_{\beta,\beta\_0,}||\boldsymbol{\beta}||$$

$$sub\; y_i(\boldsymbol{x_i^T\beta}+\beta_0)\geq 1-\xi_i;\forall i$$

[88] Hastie, Tibshirani, Friedmann, *The Elements of Statistical Learning*, 2019

$$\xi_i \geq 0; \sum_{i=1}^{n} \xi_i \leq K \tag{1.50}$$

Si nota che i punti che cadono ben oltre la distanza ottima M influiscono poco nella determinazione dei parametri dell'iperpiano.

Il problema precedente può essere espresso in maniera equivalente come

$$\min_{\boldsymbol{\beta},\beta_0,} \frac{1}{2}\left|\left|\boldsymbol{\beta}\right|\right|^2 + C\sum_{i=1}^{n} \xi_i$$

$$suby_i\left(\boldsymbol{x_i^T\beta} + \beta_0\right) \geq 1 - \xi_i; \xi_i \geq 0; \forall i. \tag{1.51}$$

dove C è il parametro di costo che sostituisce la costante K. Con $C = \infty$ si ottiene il caso perfettamente separabile. La corrispondente lagrangiana è definita come

$$L_p = \frac{1}{2}\left|\left|\boldsymbol{\beta}\right|\right|^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i\left[y_i\left(\boldsymbol{x_i^T\beta} + \beta_0\right) - (1 - \xi_i)\right] - \sum_{i=1}^{n} \mu_i\xi_i \tag{1.52}$$

che minimizzata per $\boldsymbol{\beta}, \beta_0, \xi_i$ impostando le loro derivate prime pari a 0, conduce a

$$\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x_i} \tag{1.53}$$

$$\mathbf{0} = \sum_{i=1}^{n} \alpha_i y_i \tag{1.54}$$

$$\alpha_i = C - \mu_i \forall i. \tag{1.55}$$

Insieme ai vincoli di positività

$$\alpha_i, \mu_i, \xi_i \geq 0 \forall i. \tag{1.56}$$

Per ottenere i parametri $\alpha_i$ necessari per la determinazione di $\boldsymbol{\beta}$ costruiamo il problema duale sostituendo le equazioni trovate sopra dentro la lagrangiana, e otteniamo

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{i'=1}^{n} \alpha_i\alpha_{i'}y_iy_{i'}\boldsymbol{x_i^T x_{i'}} \tag{1.57}$$

Massimizzando $L_D$ per $\alpha_i$ sotto ai vincoli di $0 \leq \alpha_i \leq C$ e $\sum_{i=1}^{n} \alpha_i y_i = 0$, otteniamo le stime $\hat{\alpha}_i$. Questo ci permette trovare i parametri $\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \boldsymbol{x_i}$ soluzione del problema. Per trovare

il parametro $\beta_0$, sappiamo che nel caso un punto $x_i$ sia un support vector, vale la seguente condizione

$$y_i(\boldsymbol{x_i^T}\boldsymbol{\beta} + \beta_0) = 1 \tag{1.58}$$

dato che support vectors presentano la distanza massima con l'iperpiano. Sostituendo $\widehat{\boldsymbol{\beta}}$ dentro l'equazione e esplicitando per $\beta_0$, troviamo la stima di questo parametro[89].

### *1.9.2. Estensione non lineare*

Vi sono dei casi in cui non si riesce a separare perfettamente i dati usando degli iperpiani lineari, ma vi è la possibilità di separarli usando iperpiani non lineari. Questi casi sono chiamati "*non linearly separable data*". Invece di costruire iperpiani non lineari si preferisce mappare lo spazio di input, dove i dati sono separabili in maniera non lineare, in uno spazio in cui i dati siano separabili linearmente. Queste due soluzioni sono equivalenti ma la seconda è più semplice da attuare. Di seguito un'immagine che rappresenta la situazione.

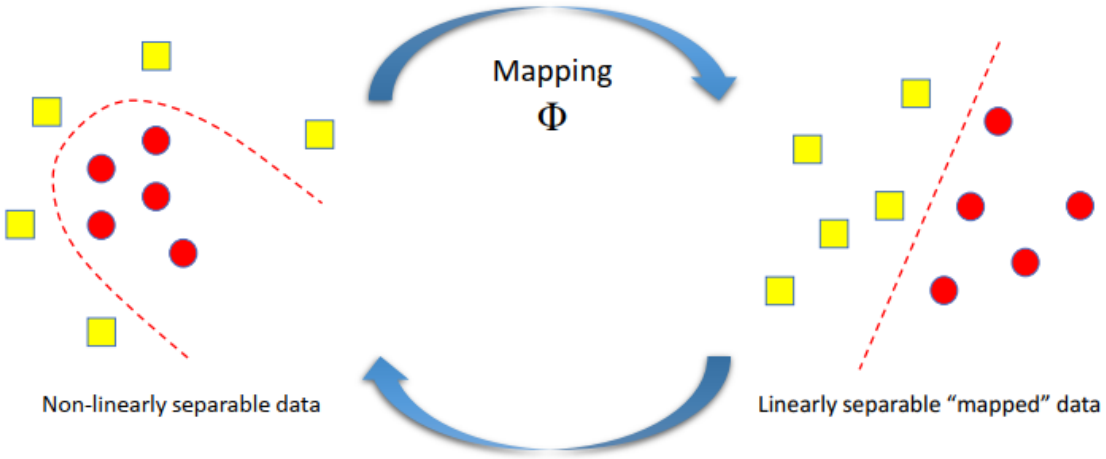


*Figura 1.21 Estensione non lineare SVM for classification* [90]

Matematicamente questo conduce ad una funzione del tipo $f(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})^{\boldsymbol{T}}\boldsymbol{\beta} + \beta_0$, dove sostituendo $\boldsymbol{\beta}$ con $\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \boldsymbol{h}(\boldsymbol{x_i})$, otteniamo

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i \boldsymbol{y_i} \prec \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x_i}) \succ + \beta_0 \tag{1.59}$$

I parametri $\alpha_i$ si ottengono dalla soluzione del problema duale definito dalla lagrangiana

[89] Hastie, Tibshirani, Friedmann, *The Elements of Statistical Learning*, 2019
[90] Antonio Candelieri, Appunti di Lezione, Lezione 7 , A.A 2020/2021

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} \prec \boldsymbol{h}(\boldsymbol{x_i}), \boldsymbol{h}(\boldsymbol{x_{i'}}) \succ \quad (1.60)$$

Le formule per $\widehat{\boldsymbol{\beta}}$ e $L_D$ si ottengono ripetendo il problema di prima usando la nuova $f(\boldsymbol{x})$**,** dove $\boldsymbol{h}()$ è una funzione che trasforma le variabili in input come polinomi o splines, e performa la mappatura dello spazio degli input (Φ nell'immagine).

In realtà la funzione entra nel problema solo attraverso il prodotto interno $K(x, x') = \prec \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x'}) \succ$ denominato Kernel, quindi abbiamo bisogno di definire solo questa funzione e non l'intera transformazione $\boldsymbol{h}()$. Questa procedura è chiamata *"Kernel Trick"*. I tipi di Kernel più usati sono

- dth-Degree polynomial:

$$K(x, x') = (1 + \prec \boldsymbol{x}, \boldsymbol{x'} \succ)^d \quad (1.61)$$

- Radial basis:

$$K(x, x') = e^{-\gamma \left|\left|x - x'\right|\right|^2} \quad (1.62)$$

- Neural network:

$$K(x, x') = tanh(\kappa_1 \prec \boldsymbol{x}, \boldsymbol{x'} \succ + \kappa_2)^{91} \quad (1.63)$$

### *1.9.3. Support Vector Machine for regression*

Le support vector machine vengono adattate ai problemi di regressione tramite la soluzione del seguente problema di minimo

$$\min_{\boldsymbol{w} \in R^d} \ \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2 + C\sum_{i=1}^{n} |\xi_i|$$

$$sub\ |\boldsymbol{y_i} - \prec \boldsymbol{w}, \boldsymbol{x_i} \succ| \leq \varepsilon - |\xi_i|;\ \forall i. \quad (1.64)$$

che è rappresentato graficamente dalla seguente immagine (caso con $\boldsymbol{x_i} \in R^2$). La funzione di regressione sarà $f(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{w}$ con $\boldsymbol{X} \in \boldsymbol{R^{nxd}}$**.**

[91] Hastie, Tibshirani, Friedmann, *The Elements of Statistical Learning*, 2019

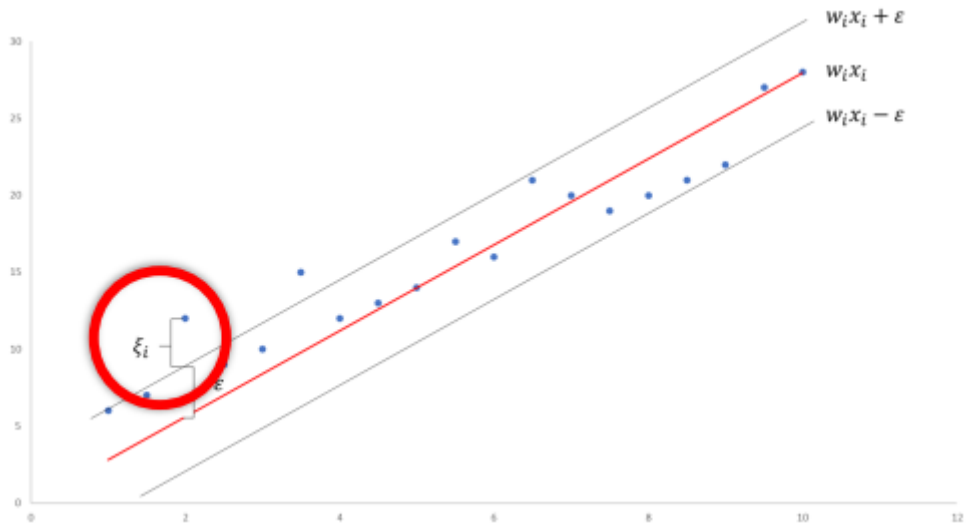


*Figura 1.22 Rappresentazione del problema SVM for regression* [92]

Il problema è quello di minimizzare la norma 2 del vettore dei parametri, dove i termini di errore sono aggiunti al problema tramite un vincolo. Questo porta a considerare solo errori maggiori della quantità $\varepsilon > 0$. Le svm-regression sono quindi caratterizate da una funzione di perdita ε-insitive. La soluzione porta alla definizione dei parametri **w** delle rette espresse in figura, dove le due rette "margine" cercano di contenere al loro interno tutti i punti $(\boldsymbol{x_i}, y_i)$. Come per il problema di classificazione le variabili $\xi_i$ sono state aggiunte per permettere ad alcuni punti di giacere al di fuori delle due rette limite, dove il parametro C definisce il trade off fra la tolleranza per i punti al di fuori dei margini e la complessità del modello. Con C=0 si ha nessuna tolleranza e poca adattabilità ai dati. Solitamente il parametro C viene stimato tramite metodi computazionali, scegliendo il valore che permette di contenere più punti fra le due rette limite. Di seguito un'immagine rappresentativa dell'impatto del parametro C.

[92] Antonio Candelieri, Appunti di Lezione, Lezione 7 , A.A 2020/2021

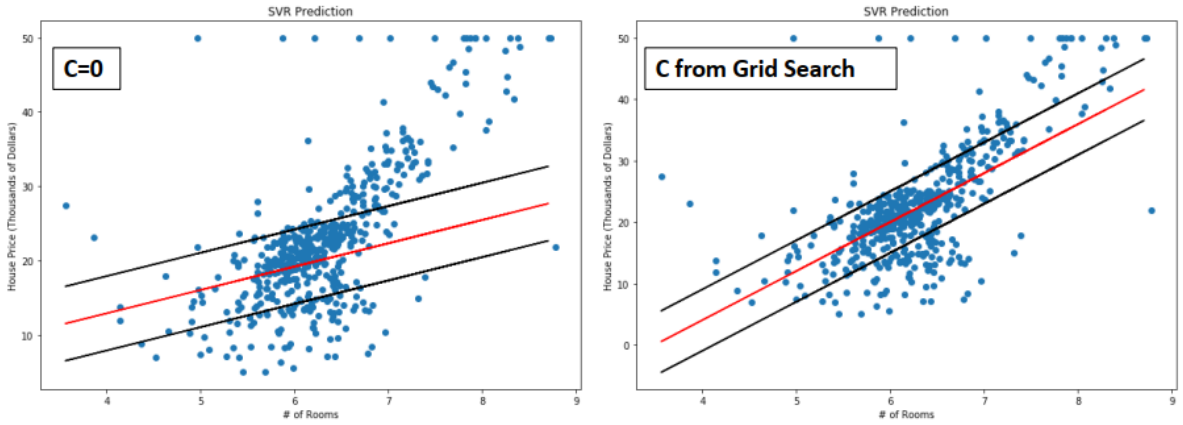


*Figura 1.23 Impatto del parametro C* [93]

### *1.9.4. Estensione non lineare*

L'estensione non lineare viene introdotta allo stesso modo del caso di classificazione. Si considera una trasformazione delle variabili in input $\boldsymbol{h}(\boldsymbol{x_i})$**,** che entra nel problema tramite il prodotto interno $K(\boldsymbol{x}, \boldsymbol{x'}) =\prec \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x'}) \succ$ denominato Kernel. Questo permette di mappare lo spazio degli input in cui questi sono separabili in maniera non lineare in uno in cui lo sono linearmente. Valgono gli stessi kernel espressi precedentemente.

## 1.10. Ensemble methods: Random Forest

Sono stati inoltre implementati dei metodi chiamati "Ensemble Learnings", basati sull'assemblamento di multipli modelli base dal poco potere predittivo denominati "weak learners". Questo permette di creare un modello in grado di effettuare previsioni molto precise, a discapito però dell'aumento del tempo computazionale richiesto e della riduzione dell'interpretabilità dei risultati.

Il metodo Random Forest si basa proprio su questo, vengono fatti crescere molti alberi che verranno combinati tra loro per produrre una singola previsione. Questo conduce ad un miglioramento della precisione della previsone, a discapito della sua interpretabilità.

Le random forest utilizzano un ricampionamento Bootstrap, che permette di costruire un campione di grandezza n $(\tilde{\boldsymbol{x}}_i \tilde{y}_i); i = 1, \dots . n, \tilde{x}_i \in R^p$ estraendoli in maniera casuale con reinserimento da un dataset di training $(\boldsymbol{x_i} y_i); i = 1, \dots . n, x_i \in R^p$ . Con questo metodo circa $\frac{1}{3}$

[93] Antonio Candelieri, Appunti di Lezione, Lezione 7 , A.A 2020/2021

delle osservazioni non viene stratto, formando un campione chiamato out-of-bag. Questo campione può essere usato come test set per testare le performance del modello.

Le Random Forest si basano su algoritmo chiamato Bootstrap Aggregator (Bagging) che permette di ridurre la varianza di un modello, in questo caso degli alberi di regressione o classificazione.

- Vengono generati B campioni bootstrap $(\tilde{\boldsymbol{x}}^{\boldsymbol{b}}{}_{\boldsymbol{i}}\tilde{y}^{b}{}_{i}); i = 1, \dots . n; b = 1, \dots . B$ da un training set $(\boldsymbol{x_i}y_i); i = 1, \dots . n$.
- Viene fittato un albero di regressione $\hat{f}^b$ o di classificazione $\hat{c}^b$ per ogni campone bootstrap
- Prendo una media delle previsioni fatte dagli alberi fittati sui campioni bootstrap:
    - Per gli alberi di regressioni avremo

$$\bar{f}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x) \tag{1.65}$$

    - Per gli alberi di classificazione avremo

$$\bar{c}(x) = Mode[\hat{c}^b(x), b = 1, \dots B] \tag{1.66}$$

      Se vogliamo una stima delle probabilità di ogni classe, possiamo prendere la media delle probabilità previste per ogni classe k dai vari alberi di classificazione allenati sui campioni bootstrap x: $\hat{p}_k^b(x), k = 1, \dots K$, calcolata come il rapporto fra il numero delle $y_i$ che risiedono nelle regioni associate alle varie classi sul numero totale delle $y_i$ presenti nel campione.

$$\bar{p_k}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{p}_k{}^b(x)\,; k = 1, \dots K \tag{1.67}$$

Una spiegazione del perché il metodo bagging, e quindi le random forest, funzionano la sia ha calcolando l'errore di previsione di un albero di regressione $\hat{T}_D$ stimato su un training set $D = [(X_i, Y_i); i = 1, \dots N]$. Indichiamo $\bar{T} = E[\hat{T}_D]$ e $\bar{T}(\dot{X}) = E[\hat{T}_D(\dot{X})|\dot{X}]$. Calcolando l'errore di previsione dell'albero applicato ad un nuovo campione $(\dot{X}, \dot{Y})$ abbiamo

$$E\left[\left(\dot{Y} - \hat{T}_D(\dot{X})\right)^2\right] = E\left[\left(\dot{Y} - \bar{T}(\dot{X})\right)^2\right] + E\left[Var\left[\hat{T}_D(\dot{X})|\dot{X}\right]\right] \tag{1.68}$$

Le random forest e il bagging cercano di prevedere $\bar{T}$, che ci aspettiamo abbia un errore di previsione basso, in quanto dovrebbe riuscire ad approssimare bene $E(\dot{Y}|\dot{X} = \dot{x})$. Questo a

discapito di $E[Var[\hat{T}_D(\dot{X})|\dot{X}]]$ che avendo fatto crescere molto l'albero senza potarlo dovrebbe essere molto complesso, e quindi molto variabile.

Essendo calcolati su dei dati ricampionati da uno stesso set, gli alberi del bagging presenano una elevate correlazione fra di loro. Per cercare di diminuirla le random forest variano le colonne di ogni campione boosting, selezionando prima di ogni split $m \leq p$ predittori in maniera casuale. m è un parametro di tuning che si può calcolare ad esempio con cross validation. Quando $m = p$, si ritorna al caso del bagging.

Infatti se prendiamo un set di variabili casuali $Z_1, \dots . Z_B$ identicamente distribuite fra di loro, con $Var(Z_j) = \sigma^2$ e $Corr(Z_j, Z_l) = \rho$, la varianza della media fra queste variabili, $\bar{Z}$, è pari

$$Var(\bar{Z}) = \rho\sigma^2 + \frac{(1-\rho)}{B}\sigma^2 \tag{1.69}$$

Dove è importante ridurre $\rho$, in quanto al crescere di B (che può essere aumentato di molto) il secondo termine tende a 0.[94]

## 1.11. Ensemble methods: Boosting

### *1.11.1. Introduzione*

Otre alle Random Forest, anche il Boosting è un metodo basato sull'aggregazione di più week learners e come queste si basa sugli alberi.

Esso segue i seguenti step generali:

- Allena il week learner sul training set
- Successivamente allena nuovamente il metodo, attribuendo più peso alle osservazioni che sono state fittate male.
- Ripete il processo, finché non si raggiunge una condizione di stop.

Si vede quindi come il Boosting sia un algoritmo in grado di apprendere dai propri errori e correggersi durante il processo.

Esistono alcuni algoritmi boosting, nel corso di questo lavoro è stato applicato il gradient boosting.

[94] Aldo Solari, Appunti di Lezione Bagging & Random Forest, Data Mining M

### *1.11.2. Gradient Boosting*

Il gradient boosting si basa sulla seguente regola chiamata *"gradient boosting descendent rule"*. Prendiamo una funzione di perdita continua e differenziabile $L: R^n \to R$ e impostiamo il seguente problema di minimo:

$$\min_{\boldsymbol{f} \in R\hat{}n} L(\boldsymbol{f})$$

partendo da un punto iniziale $\boldsymbol{f_0}$ il *"gradient descendent"* minimizza iterativamente la funzione L muovendosi ad ogni passo nella direzione indicata da $-\nabla L(\boldsymbol{f})$. Infatti se prendiamo l'espansione di taylor al primo ordine di:

$$L(\boldsymbol{f} + \boldsymbol{d}) \approx L(\boldsymbol{f}) + \nabla L(\boldsymbol{f})^T \boldsymbol{d} \qquad (\mathbf{1.70})$$

dove **d** è un vettore unitario, cerchiamo dato un punto iniziale **f,** la direzione **d** da prendere in maniera che L(**f+d**) sia minimizzata. Il vettore **d** soluzione di questo è

$$\boldsymbol{d} = -\frac{\boldsymbol{\nabla} \mathrm{L}(\boldsymbol{f})}{\left|\left|\boldsymbol{\nabla} \mathrm{L}(\boldsymbol{f})\right|\right|_{\mathbf{2}}} \qquad (\mathbf{1.71})$$

Di conseguenza viene prodotto un algoritmo iterativo del tipo

$$\boldsymbol{f}^{b+1} = \boldsymbol{f}^b + \lambda \boldsymbol{g} \qquad (\mathbf{1.72})$$

dove $\boldsymbol{g} = -\boldsymbol{\nabla} L(\boldsymbol{f})$ è il gradiente negativo della funzione L mentre $\lambda = \left|\left|\boldsymbol{\nabla} \mathrm{L}(\boldsymbol{f})\right|\right|_{\mathbf{2}}^{-\mathbf{1}}$ è una misura della grandezza dello step.[95]

Sulla base di questa regola viene generato il seguente algoritmo (Gradient Boosting)

1. Si inizializza $\hat{f}^0(x) = 0$, si fissa B e il parametro di shrinkage $\lambda$.
2. Per ogni b=1, …. B
    1. Si calcola il gradiente negativo a quel passo della funzione di perdita scelta

$$r_i = \frac{\partial L(y_i, f_i)}{\partial f_i}\Big|_{f_i = \hat{f}_i{}^{b-1}(x_i)}, i = 1, \ldots . n \qquad (1.73)$$

    2. Si approssima il gradiente con un albero g di profondita d

$$(x_i, r_i), i = 1, \ldots n \to \hat{g}(x) \qquad (1.74)$$

    3. Si aggiornano le stime

$$\hat{f}^{b+1}(x) = \hat{f}^b(x) + \lambda \hat{g}(x) \qquad (1.75)$$

[95] Aldo Solari, Appunti di Lezione Bagging & Random Forest, Data Mining M

3. Si restituiscono i valori $\hat{f}^b(x), b = 1, \ldots B$

A seconda della funzione di perdita scelta, il gradient boosting può essere adattato a problemi di regressione (per esempio con una mean squared error loss ) o a problemi di classificazione (come una cross-entropy loss).

I parametri di tuning $(\lambda, B, d)$ possono essere stimati tramite cross-validation.[96]

[96] Aldo Solari, Appunti di Lezione, Boosting, Data Mining M

# 2. Costruzione dataset

## 2.1. Estrazione dati da Twitter

Il dataset utilizzato per cercare di prevedere i rendimenti dei titoli finanziari scelti è stato costruito dal principio, tramite un processo che è stato caratterizzato da vari step.

Per prima cosa vi era la necessità di reperire le frasi da far analizzare a FinBERT, cioè i messaggi condivisi dagli utenti del social network Twitter, chiamati tweet. Non sono stati scelti a caso, infatti ognuno dei tweet estratti contiene delle parole che si riferiscono ai titoli in analisi. In particolare, sono stati scelti quei tweet che avevano all'interno del testo almeno uno di questi termini:

- Nome dell'azienda quotata in borsa (nello specifico Bayerische Motoren Werke o BayerischeMotorenWerke per l'estrazione riguardante BMW e Tesla per l'estrazione riguardante Tesla)
- Hashtag del nome dell'azienda (#BayerischeMotorenWerke per BMW o #Tesla per Tesla)
- Sigla finanziaria dell'azienda (BMW per BMW o Tsla per Tesla)
- Hashtag della sigla finanziaria dell'azienda (#BMW per BMW o #Tsla per Tesla)
- Cashtag della sigla finanziaria dell'azienda ($BMW per BMW o $Tsla per Tesla)

La ricerca e l'estrazione dei tweet è stata effettuata tramite Twitter API[97] v2, un insieme di procedure informatiche fornite da Twitter atte alla costruzione di query contenti i messaggi degli utenti del social network.

L'estrazione si è eseguita impostando le *keyword* indicate precedentemente. Questa performa una ricerca su tutti i tweet condivisi dall'anno 2006, dove vengono confrontati i token[98] delle parole contenute in ogni tweet con le parole chiavi indicate in input. Dato che le parole del testo vengono separate in più token sulla base di punteggiatura, simboli e caratteri unicode separatori[99], la *keyword* nome dell'azienda permette già di selezionare anche i tweet che hanno al loro interno l'hashtag o il cashtag della denominazione aziendale, e lo stesso vale per il

[97] Application Programming Interface
[98] Cioè come le parole del testo vengono immaganizzate all'interno del database di Twitter. Il concetto è simile a quello spiegato quando si trattava l'algoritmo WordPiece embeddings, dove le parole del testo venivano *tokenizzate* in più pezzi chiamati *sub-words*.
[99] https://developer.twitter.com/en/docs/twitter-api/tweets/search/integrate/build-a-query

simbolo finanziario. Tuttavia per sicurezza le altre condizioni sono state comunque indicate in maniera esplicita.

Sono stati estratti tutti i tweet condivisi dal 01/01/2019 al 31/03/2020, dove quelli emessi durante il 2019 costituiscono il training set dell'analisi, mentre quelli riferiti ai primi tre mesi del 2020 il test set.

Essendo Twitter un social network globale, i messaggi estratti sono caratterizzati da lingue diverse (più di 65) dove circa la meta sono scritti in inglese. Alcuni Tweet presentano inoltre caratteri come emoticons o link esterni.

FinBERT è in grado di:

- analizzare più lingue diverse, essendo basato su BERT[100].
- gestire i caratteri particolari che non riconosce, associandoli ad uno speciale token, avendo alla base il meccanismo del WordPiece embeddings.

Per queste motivazioni non è stato operato alcun tipo di filtro ai dati estratti, evitando perdita di informazioni.

Vi sono 4 tipologie diverse di tweet estratti[101]:

1. Tweets originali.
2. Tweet condivisi da utenti diversi rispetto all'autore del messaggio, chiamati retweet
3. Retweet con aggiunta di informazioni (immagini, testo ecc …) da parte dell'utente che sta condividendo il messaggio, chiamati "quoted to".
4. Tweet in risposta a messaggi condivisi in precedenza, chiamati "replied to".

A parte la prima tipologia, gli altri tweets si riferiscono a dei messaggi scritti e condivisi precedentemente. Per ognuno di questi tweets, l'API permette di estrarre a quale messaggio si riferiscono, nonché una tabella contenente le informazioni riguardanti il tweet originale (le stessa estratte per i tweet "figli"). Sono state costruite quindi due tabelle, una contenente i testi e le informazioni relative ai tweet, e un'altra contenente i testi e le informazioni relative ai tweet originali a cui i primi si riferiscono.

[100] In realtà l'articolo non specifica esattamente su quale versione (base o multilingua) è basato FinBERT, tuttavia empiricamente si è visto che è in grado di valutare anche lingue diverse dall'inglese. Inoltre la versione base riesce comunque a gestire i caratteri che non riconosce grazie al meccanismo del WordPiece emebeddings.
[101] https://help.twitter.com/en/using-twitter/types-of-tweets.

Mentre i tweets della tipologia 3 e 4 sono dei messaggi diversi rispetto ai tweets a cui si riferiscono, i retweet hanno lo stesso testo del tweet originale, condiviso da un utente diverso in un momento successivo.

Twitter API è stata utilizzata tramite il linguaggio di programmazione R[102][103]

Per ogni tweet sono state estratte le seguenti informazioni (che si tramutano nelle varie colonne delle tabelle estratte):

- Codice id con cui viene identificato il tweet.
- Testo condiviso.
- Codice id dell'utente autore del messaggio.
- Data di creazione del tweet.
- Lingua con cui è stato scritto il messaggio.
- Indicazione se il tweet appartiene alla categoria 2.
- Indicazione se il tweet appartiene alla categoria 3.
- Indicazione se il tweet appartiene alla categoria 4.
- Codice id del tweet originale a cui si riferisce il tweet appartenete alla categoria 2.
- Codice id del tweet originale a cui si riferisce il tweet appartenete alla categoria 3.
- Codice id del tweet originale a cui si riferisce il tweet appartenete alla categoria 4.

Come si vede è stata creata una colonna per ogni tipologia di tweet, invece di crearne una sola indicante la tipologia di appartenenza per ogni messaggio. Questo perché ogni tweet può appartenere a più categorie diverse.

Per quanto riguarda Tesla sono stati estratti:

- riferiti all'anno 2019, 9.722.231 tweet (di cui 4.703.676 retweet).
- riferiti ai primi 3 mesi dell'anno 2020, 2.680.951 tweet (di cui 1.325.566 retweet).

Per quanto riguarda BMW sono stati estratti:

- riferiti all'anno 2019, 4.793.200 tweet (di cui 2.345.833 retweet).
- riferiti ai primi 3 mesi dell'anno 2020, 1.193.362 tweet (di cui 660.712 retweet).

[102] Per l'implementazione dell'estrazione si è partiti dal sample code fornito nella seguente pagina https://github.com/twitterdev/Twitter-API-v2-sample-code/blob/main/Full-Archive-Search/full-archive-search.r

[103] Il codice scritto si può trovare insieme agli altri script della tesi al seguente link https://github.com/Moreno-Sanna/Mythesis, nel file "01_Tweets_extraction.R".

Sia per Tesla che per BMW, sono stati estratti tutti i tweet riferiti al periodo indicato.

## 2.2. Analisi con FinBERT

Una volta estratti, i tweet sono stati analizzati tramite FinBERT. E' stata utilizzata la stessa configurazione degli autori del modello, contenuta nell'articolo spiegato sopra, per due ragioni principali:

1. Non vi è disponibilità di tweets etichettati con cui poter fare del fine-tuning al modello, o almeno non ne sono a conoscenza. Questo rientra nel problema della scarsità di data labeled, già spiegato in precedenza.
2. L'ulteriore pre-tuning effettuato dagli autori del modello su un corpora prettamente finanziario non ha portato ad ulteriori miglioramenti in termini di performance, ed è inoltre un processo oneroso dal punto di vista computazionale, temporale e di difficile implementazione.

FinBERT è stato implementato utilizzando il linguaggio Python[104][105]. Ogni tweet è stato analizzato singolarmente, iterando il processo per ogni messaggio. Per quanto riguarda i retweet non sono stati analizzati direttamente, ma gli sono stati associati i risultati ricavati dalle analisi dei tweets a cui sono relazionati. Questo perché come spiegato precedentemente, presentano un testo identico a quello dei messaggi originali. I retweet identici fra di loro sono stati poi conteggiati sia sulla base del tweet a cui si riferiscono, sia sulla base del giorno di condivisione. Sono stati contati insieme quindi i tweet che condividono lo stesso tweet originale e lo stesso giorno di creazione. Si è scelto di non operare nessuna trasformazione sul conteggio, ogni retweet in questo modo ha avuto la stessa valenza dei tweet originali, ma si è lasciata l'indicazione della categoria di appartenenza. E' stato così ottenuto un dataset, caratterizzato dalle seguenti colonne:

- data_retweet: Il giorno in cui i/il retweet è stato condiviso. Compilata soltanto se la riga si riferisce ad uno o a più retweet.
- n_retweet: Conteggio dei retweet associati ad un unico tweet originale condivisi in quel giorno. Compilata soltanto se la riga si riferisce ad uno o a più retweet.
- is_retweet: Indicazione se la riga si riferisce a uno/più retweet oppure no.

[104] Per l'implementazione di FinBERT ci si è riferiti alla repository Github del modello, https://github.com/ProsusAI/finBERT, nonché alla guida base della libreria Transformers, sempre su Github https://github.com/huggingface/transformers.
[105] Il codice scritto si può trovare insieme agli altri script della tesi al seguente link https://github.com/Moreno-Sanna/Mythesis, nel file "03_Finbert_analysis.py".

- id: Codice del tweet analizzato (a cui eventualmente sono riferiti i retweet, se la riga è riferita a questi).
- data: Data di creazione del tweet analizzato.
- sentence: Testo del tweet analizzato.
- logit_positive: Probabilità che il testo abbia un significato positivo, calcolata dalla layer classificatore del modello (che trasforma i valori in probabilità tramite la funzione softmax).
- logit_negative: Probabilità che il testo abbia un significato negativo, calcolata dalla layer classificatore del modello (che trasforma i valori in probabilità tramite la funzione softmax).
- logit_neutral: Probabilità che il testo abbia un significato neutrale, calcolata dalla layer classificatore del modello (che trasforma i valori in probabilità tramite la funzione softmax).
- prediction: Label attribuita al testo. Può essere Positive, Neutral o Negative, e si attribuisce sulla base di quale probabilità tra quelle precedenti presenta il valore maggiore.
- sentiment_score: E' il livello di sentimento della frase. Viene calcolato sottraendo al valore della probabilità logit_positive il valore della probabilità logit_negative.

## 2.3. Aggregazione su base giornaliera

Successivamente i dataset creati (riferiti separatamente a Tesla e a BMW) sono stati aggregati su base giornaliera, prendendo come riferimento la colonna data_retweet se si trattava di retweet, altrimenti la colonna data. In questo modo si sono ottenute le serie storiche giornaliere degli indicatori descritti sopra, sintetizzati tramite somma e media. Non è inusuale che i retweet siano usati per scopo pubblicitario, venendo condivisi molte volte da account non collegati a persone reali. Di conseguenza questi messaggi non rispecchiano una reale informazione o sentimento delle persone, nonostante siano presenti in misura elevata all'interno del dataset. Per tenere conto di questo, gli indici sono stati calcolati sia sulla base dei tweet totali condivisi (tweet + retweet), sia escludendo i retweet dal calcolo. I nuovi dataset presentano 455 righe (365 righe nel training set e 90 nel test set, una per ogni giorno del periodo di riferimento) per 26 colonne. Di seguito la descrizione degli indicatori:

1. Giorno: data di riferimento.
2. n_Tweet: numero di tweet no retweet condivisi.

3. n_retweet: numero di retweet condivisi.
4. n_tweet_totali: numero di tweet totali (tweet + retweet) condivisi
5. n_positive: conteggio delle label positive (no retweet).
6. n_negative: conteggio delle label negative (no retweet).
7. n_neutral: conteggio delle label neutral (no retweet).
8. sum_positive: somma delle probabilità logit_positive dei tweet (no retweet).
9. sum_negative: somma delle probabilità logit_negative dei tweet (no retweet).
10. sum_neutral: somma delle probabilità logit_neutral dei tweet (no retweet).
11. sum_sentiment_score: somma del valore sentiment score (no retweet).
12. mean_positive: media delle probabilità logit_positive dei tweet (no retweet).
13. mean_negative: media delle probabilità logit_negative dei tweet (no retweet).
14. mean_neutral: media delle probabilità logit_neutral dei tweet (no retweet).
15. mean_sentiment_score: media del valore sentiment score (no retweet).

Gli indicatori dal numero 5 al numero 15 sono stati anche calcolati come anticipato considerando tutti i tweet (tweet + retweet), e vengono indicati aggiungendo la desinenza *"_pond"* alla fine del nome, per un totale quindi di 26 colonne.

A questi dataset si è aggiunta la serie storica dei rendimenti dell'asset a cui si riferiscono i tweet, calcolata sulla base dei prezzi estratti da yahoo finance[106] nel periodo di riferimento dell'analisi (dal 1 gennaio del 2019 al 31 dicembre del 2019 per il training set e dal 1 gennaio 2020 al 31 marzo 2020 per il test set).[107]

Oltre all'analisi sui rendimenti puri, si è interessati a capire la capacità delle informazioni ricavate dai tweet di prevedere gli extrarendimenti degli asset, cioè tutti i rendimenti non riconducibili alle fluttuazioni del settore di appartenenza delle due aziende (in questo caso automotive). Come indice dei rendimenti del settore è stata presa la serie storica del fondo di investimento CARZ, che è composto da 33 partecipazioni in aziende del settore auto, tra cui anche Tesla e BMW[108]. Anche qui i prezzi del fondo sono stati ricavati da yahoo finance[109]. Gli

[106] Per tesla: https://finance.yahoo.com/quote/TSLA
Per BMW: https://it.finance.yahoo.com/quote/bmw.de
[107] Il codice con cui è stata implementata questa aggregazione si può trovare insieme agli altri script della tesi al seguente link https://github.com/Moreno-Sanna/Mythesis, nel file "07_Create aggregate dataset by day.R".
[108] L'elenco delle partecipazioni si può trovare al sito: https://www.ftportfolios.com/retail/etf/ETFholdings.aspx?Ticker=CARZ
[109] https://finance.yahoo.com/quote/CARZ

extrarendimenti sono stati calcolati sottraendo ai rendimenti puri di BMW e Tesla, il rendimento del fondo CARZ.

Dato che i giorni di attività dei mercati finanziari sono inferiori rispetto al calendario, è stato necessario effettuare un'ulteriore aggregazione per allineare le date presenti nel dataset e quelle associate ai rendimenti e agli extrarendimenti dei due asset. Per tutte le date che presentavano questo problema è stata fatta una media fra i giorni che non avevano associato nessun rendimento e il primo giorno in cui né veniva registrato uno, allineando così le varie date[110]. Inevitabilmente con questa operazione le righe dei dataset si sono ridotte.

Infine, per poter utilizzare durante l'analisi anche le informazioni relative non solo al giorno stesso in cui si manifesta il rendimento ma anche quelle dei giorni precedenti, sono state aggiunte ai dataset anche le colonne relative al primo e al secondo ritardo di tutti gli indici descritti precedentemente (escludendo la prima colonna indicante il giorno di riferimento) aggiungendo anche il primo e secondo ritardo del rendimento stesso.

In totale quindi sono state analizzate quattro serie storiche:

- Rendimenti Tesla.
- Extrarendimenti Tesla.
- Rendimenti BMW.
- Extrarendimenti BMW.

dove per ogni serie storica è stato prodotto un dataset particolare, con lo stesso numero di colonne (79 in totale) ma con numero di righe diverso (311 per i rendimenti e extrarendimenti Tesla, 312 per i rendimenti BMW e 304 per gli extrarendimenti BMW). Questo perchè, anche se con differenze minime, i giorni in cui si manifestavano i rendimenti non erano precisamente uguali fra le tre serie storiche (rendimenti Tesla, BMW e CARZ).

Tuttavia nonostante un leggero disallineamento sui giorni, i rendimenti e gli extrarendimenti Tesla sono stati analizzati utilizzando la sintesi delle informazioni ricavate dai tweet collegati all'asset Tesla, mentre per i rendimenti e gli extrarendimenti BMW sono state utilizzate le informazioni ricavate dai tweet collegati a BMW.

[110] Ad esempio i valori dell'indice "mean_sentiment_score" associati ad ogni sabato e domenica, giorni in cui la borsa è chiusa, sono stati aggregati con il valore associato al lunedì successivo, giorno in cui la borsa è aperta e a cui è possibile collegare un rendimento, facendo una media fra questi tre valori. A questo valore di sintesi è stato associato il rendimento registrato nella giornata di lunedì. Questo allineamento è stato effettuato per ogni indice presente nel dataset, e per tutti i giorni in cui si presentava questa problematica.

Inoltre, per ogni serie storica, si è provato a prevedere non solo i valori ma anche i segni dei rendimenti / extrarendimenti.

Il codice con cui è stata performata questa ultima aggregazione insieme al calcolo degli extrarendimenti e all'unione dei dataset con i rendimenti degli asset, si può trovare nella repository Github collegata alla tesi[111].

[111] https://github.com/Moreno-Sanna/Mythesis. Questa aggregazione è stata ripetuta per ogni rendimento e extrarendimento analizzato, e si trova all'interno dei file R contenenti i codici con cui si sono performate le analisi ( da file 08 a file 11).

# 3. Analisi e risultati ottenuti

## 3.1. Introduzione

Dopo aver costruito i dataset le serie storiche dei rendimenti e degli extrarendimenti sono state analizzate seguendo due obiettivi diversi:

- La previsione dell'entità dei rendimenti, cioè il valore preciso che si manifesta sui mercati, dando vita ad un problema di regressione.
- La previsione del segno dei rendimenti, cioè se il rendimento sta crescendo o meno (è positivo o negativo) rispetto al giorno precedente, dando vita ad un problema di classificazione.

Come si è detto precedentemente durante la spiegazione della costruzione della base dati, per l'analisi si sono tenute in considerazione le informazioni dello stesso giorno e ritardate di vari lag (di uno e due periodi) rispetto alla data di manifestazione dei rendimenti. Naturalmente in un'ottica di costruzione di una strategia finanziaria, i dati dello stesso periodo non si possono utilizzare, ma solo quelli associati ai vari lag. Tuttavia questo è utile per cercare di capire quanto dura l'impatto eventuale sui mercati finanziari delle informazioni condivise dagli utenti e se l'entità di questo impatto è tale da guidare o quantomeno condizionare l'andamento dei mercati, aldilà della formazione di una strategia vera e propria.

Prima di spiegare i risultati ottenuti con le analisi di ognuna delle quattro serie storiche indicate sopra, verranno spiegati i modelli usati come confronto a quelli scelti per performare le analisi (***Baseline model***) e le metriche utilizzate.

### *3.1.1. Baseline model*

Per la parte di regressione è stato scelto come modello di confronto una previsione basata sulla media dei rendimenti calcolata sul training set (che per i rendimenti finanziari è tipicamente prossima a 0).

Per la parte di classificazione non è stato scelto un particolare modello di confronto. Tuttavia facendo varie prove si è visto che in media cercando di prevedere il segno in maniera del tutto casuale ipotizzando una perfetta uguaglianza fra le probabilità di realizzazione delle due classi che caratterizzano il segno dei rendimenti (utilizzando per esempio il segno ricavato dalle realizzazioni da una normale standard), si ottiene un'Accuracy del 50%. Si è preso questo valore come soglia minima che un modello dovrebbe superare per essere considerato interessante.

### *3.1.2. Metriche utilizzate*

Per i problemi di regressione sono state utilizzate le seguenti metriche:

- ***Root Mean Square Error***:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.1}$$

- $\boldsymbol{R^2}$***:***

$$R^2 = 1 - \frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{\Sigma_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{3.2}$$

- ***RRMSE***: Il rapporto fra RMSE ($RMSE_1$) ottenuto dal modello scelto e l'RMSE del modello di benchmark ($RMSE_0$):

$$\frac{RMSE_1}{RMSE_0} \tag{3.3}$$

Per i problemi di classificazione invece le seguenti:

- ***Accuracy:***

$$ACC = \frac{1}{n}\Sigma_{i=1}^{n} I(y_i = \hat{y}_i) \tag{3.4}$$

  dove $y_i$ è l'i-esimo valore mentre $\hat{y}_i$ è la sua previsione

- ***AUC:*** Area Under Curve, cioè la misura dell'area creatasi sotto la curva ROC. Il su valore è compreso fra 0 e 1, dove 0 è il risultato presentato dal classificatore che sbaglia tutte le classi previste, 1 e il risultato del classificatore che le prevede tutte in maniera corretta.

E' stato inoltre calcolato:

- ***L'indice di correlazione di Pearson:***

$$P = \frac{\sigma_{x,y}}{\sigma_x \sigma_y} \tag{3.5}$$

  per la valutazione della correlazione fra le variabili esplicative e i valori dei rendimenti / extrarendimenti.

- ***L'indice di correlazione di Kendall:*** per la valutazione della correlazione fra le variabili esplicative e il segno dei rendimenti. La formula è la seguente

$$\tau = \frac{nc - nd}{\frac{n(n-1)}{2}} \tag{3.6}$$

dove n è la numerosità del campione, nc indica il numero di coppie concordanti ed nd il numero di coppie discordanti. Due coppie sono concordanti se per $i < j$ entrambe le variabili $(x_i, y_i)$ seguono lo stesso andamento, cioè se $x_i > x_j$ allora anche $y_i > y_j$, o viceversa. Altrimenti sono discordanti

- ***GLMdeviance***:

$$Dev_{glm} = 2(loglik_{sat} - loglik) \tag{3.7}$$

dove $loglik_{sat}$ è il valore della logverosimiglianza del modello saturato (un modello con un parametro per ogni osservazione) mentre loglik è il valore della logverosimiglianza del modello scelto[112]. Questa metrica è stata utilizzata durante la stima della regressione logistica regolarizzata, per la scelta del parametro $\lambda$.

- ***Matrice di confusione:*** Una tabella a doppia entrata dove le colonne sono relative ai valori reali della variabile risposta, mentre le righe ai valori predetti dal modello. Misura le frequenze delle combinazioni fra i valori delle righe e quelli delle colonne[113]. Di seguito un'immagine che la rappresenta.

| | | Vera classe | | |
|---|---|---|---|---|
| | | 0 | 1 | Totale |
| Classe Prevista | 0 | $a$ | $b$ | $a+b$ |
| | 1 | $c$ | $d$ | $c+d$ |
| | Totale | $a+c$ | $b+d$ | $n$ |

*Figura 3.1 Rappresentazione matrice di confusione* [114]

- ***Curva ROC:*** Si tratta di un grafico in cui:
  - in ascissa viene riportato il valore 1-specificità, ovvero il tasso dei falsi positivi, calcolato come $1 - \frac{a}{(a+c)}$, dove i valori a, b, c, e d sono le frequenze delle combinazioni riportate nella matrice di confusione rappresentata come sopra.
  - in ordinata viene riportato il valore della sensibilità, ovvero il tasso dei veri positivi, calcolato come $\frac{d}{(d+b)}$.

[112] https://rdrr.io/cran/glmnet/man/deviance.glmnet.html
[113] La matrice di confusione si può calcolare anche rappresentando i veri valori sulle righe e i valori predetti sulle colonne. E' stata utilizzata quest'ultima configurazione nel corso di questo lavoro.
[114] Nicola Lunardon, Analisi Discriminante, appunti di lezione.

La curva ROC può essere vista come un grafico che spiega il valore della sensibilità in funzione del valore di 1-specificità. Essa passa per i punti [0,0] e [1,1]. E' importate notare che il grafico in cui viene rappresentata la curva ROC presenta al suo interno anche altre due curve di benchmark:

- La retta bisettrice che congiunge direttamente i punti [0,0] e [1,1], presenta una AUC pari a 0.5 ed è relativo al risultato compiuto dal classificatore completamente random.
- Una curva che passa per i punti [0,0], [0,1] e [1,1], presenta una AUC pari a 1 e rappresenta il risultato conseguito dal classificatore che non sbaglia mai.

La curva Roc dovrebbe stare all'interno di queste due curve. Più si avvicina alla retta bisettrice, più è un cattivo classificatore. Viceversa più si avvicina alla curva che passa per il punto [0,1], più è ottimale.

Verranno presentate prima le analisi riguardanti l'asset Tesla e successivamente quelle riguardanti BMW, con una breve introduzione iniziale, soffermandosi prima sui rendimenti e poi sugli extarendimenti.

## 3.2. Analisi e risultati asset Tesla

### *3.2.1. Introduzione*

Le informazioni ricavate dai tweet collegati a Tesla e sintetizzate tramite gli indici descritti precedentemente sono state utilizzate per cercare di prevedere le due serie storiche collegate a questo asset e per valutare il livello di legame con i mercati finanziari. Di seguito viene presentata la serie storica dell'indice che meglio sintetizza le informazioni ricavate dai tweet, cioè la media giornaliera del livello di sentimento delle frasi condivise dagli utenti, considerando anche i retweet. Il periodo di osservazione va dal primo gennaio 2019 al 31 marzo 2020.

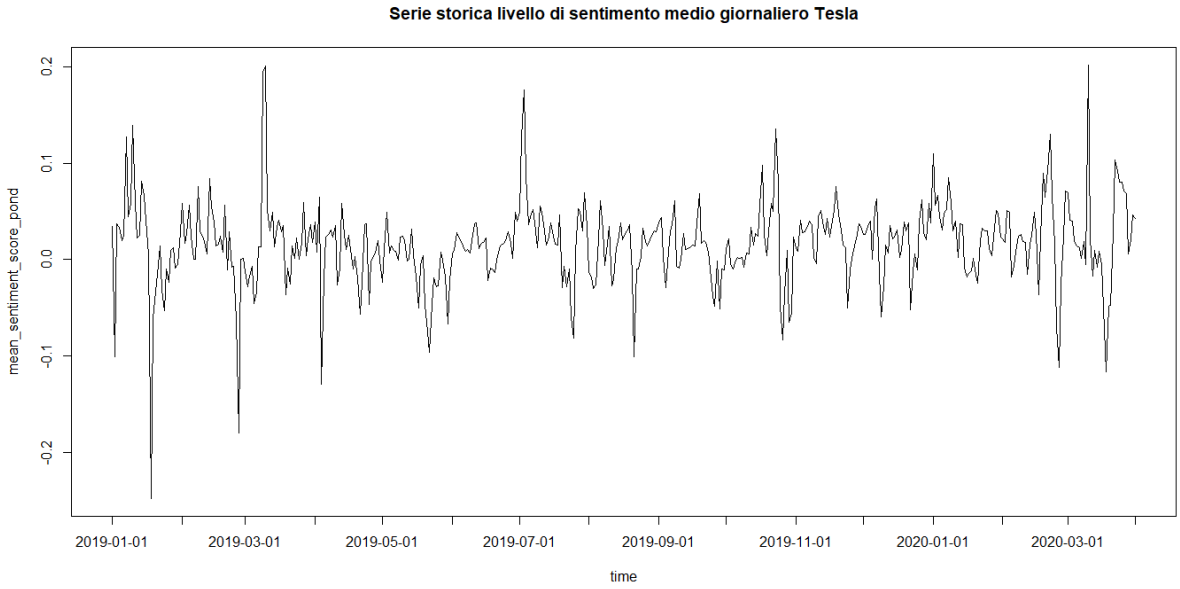


*Figura 3.2 Serie storica sentimento medio giornaliero con retweet - Tesla*

La serie è stazionaria e presenta una media intorno allo 0 (media dello 0.016), indicandoci che il sentimento delle persone rispetto a Tesla è rimasto in media inalterato nel corso del tempo ed è più o meno neutrale (leggermente positivo). Presenta comunque una standard deviation pari a 0.043, che ci indica la sua dinamicità. Si notano infatti dei picchi, (il più negativo il 18 gennaio del 2019 mentre quello più positivo il 10 marzo del 2020), e un aumento della volatilità dopo metà febbraio 2020 (standard deviation del 0.0628). Questo potrebbe essere un effetto dello scoppio della pandemia COVID-19 avvenuto proprio in quel periodo, ed una analogia con il crollo del mercato azionario causato dal covid, iniziato il 20 febbraio del 2020[115].

### *3.2.2. Previsione valori rendimenti Tesla*

Di seguito viene presentata la serie storica dei rendimenti tesla.

[115] https://it.wikipedia.org/wiki/Crollo_del_mercato_azionario_del_2020

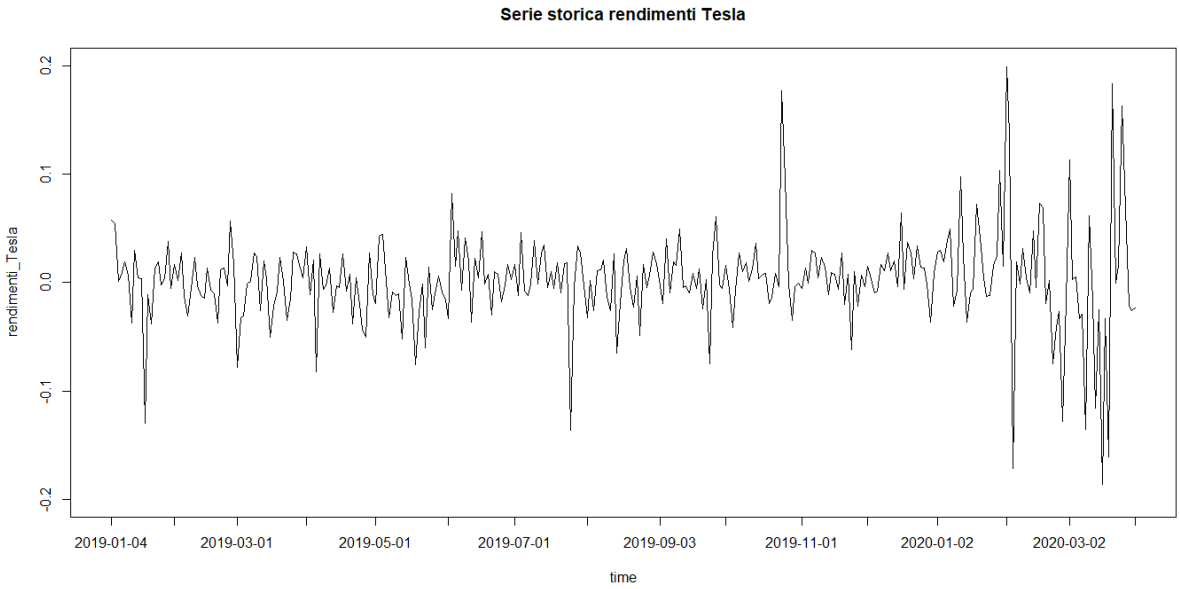


*Figura 3.3 Serie storica valori rendimenti Tesla*

Com'è tipico nei rendimenti finanziari la serie presenta una media intorno allo 0 (precisamente 0.0026) ed è stazionaria. Ha una volatilità di 0.043, e si nota un aumento di variabilità a partire da febbraio 2020 (la standard deviation nel periodo 01.02.2020-31.03.2020 è pari a 0.089), qui più marcato rispetto a quanto osservato per la serie del livello medio di sentimento.
Nella seguente tabella vengono riportate le variabili del dataset che presentano una correlazione maggiore o uguale a |0.2| con i rendimenti Tesla, in ordine decrescente

| | Correlazione |
|---|---|
| sum_sentiment_score | 0.45 |
| mean_sentiment_score | 0.41 |
| mean_positive | 0.36 |
| mean_sentiment_score_pond | 0.35 |
| sum_sentiment_score_pond | 0.35 |
| mean_positive_pond | 0.25 |
| n_positive_pond | 0.22 |
| n_positive | 0.21 |
| sum_negative | -0.21 |
| n_negative_pond | -0.22 |
| n_negative | -0.26 |
| mean_negative_pond | -0.30 |
| mean_negative | -0.32 |

*Tabella 3-1 Variabili esplicative più correlate con valori rendimenti Tesla*

Le variabili più correlate sono gli indicatori che sintetizzano il livello del sentimento giornaliero, sia con la somma che con la media, sia considerando i retweet sia senza. Anche la media giornaliera delle probabilità positive ottiene una correlazione pari al 35%. Come ci si aspetta le variabili che sintetizzano il livello di sentimento negativo (sia tramite somma o media delle probabilità di livello di sentimento negativo sia tramite il conteggio delle label "negative", con o senza retweet) presentano una correlazione negativa, viceversa per gli altri indici, che hanno una correlazione positiva. Si nota che nessuna variabile ritardata è presente nella tabella. Per prevedere i rendimenti della serie storica il miglior modello trovato è una regressione Lasso. Il parametro $\lambda$ è stato stimato tramite una *leave one out cross validation*, prendendo il valore che minimizza i MSE. Il grafico che mostra l'andamento dell'errore compiuto dal modello all'aumentare del (logaritmo del) parametro $\lambda$ è il seguente:

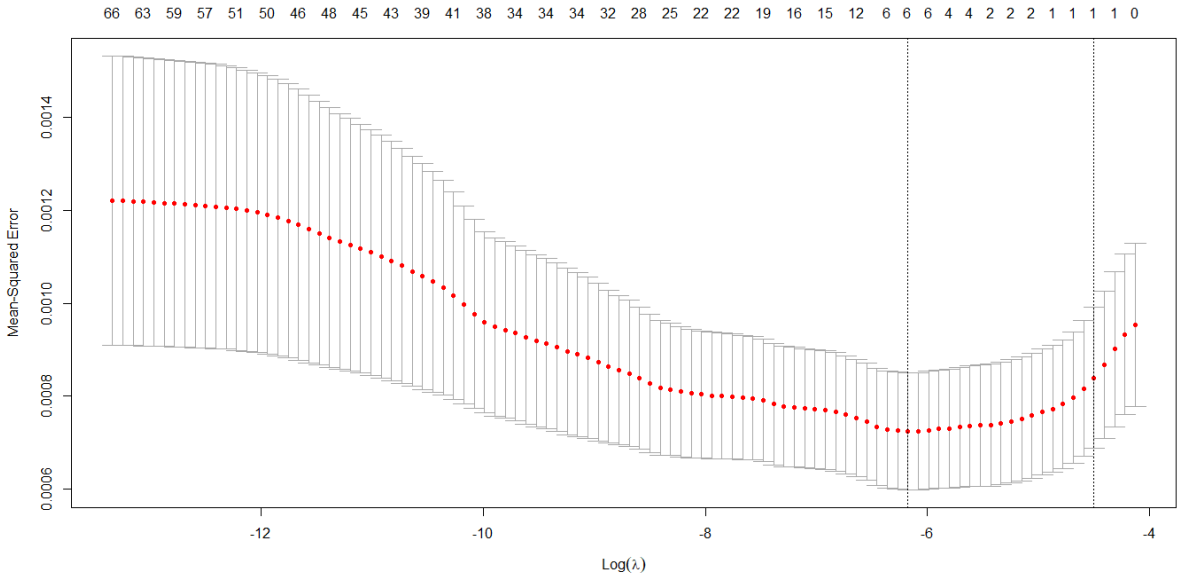


*Figura 3.4 Stima del parametro lambda - valori rendimenti Tesla*

dove i numeri in testa al grafico rappresentano il numero di variabili utilizzate dal modello, via via che il parametro aumentava.

Le variabili selezionate sono:

*sum_negative, sum_sentiment_score, mean_positive, mean_negative_lag1, mean_neutral_pond_lag1, n_retweet_lag2*

Vi è stata quindi una forte riduzione del numero delle variabili esplicative da utilizzare (6 contro le 75 totali) e nonostante i regressori ritardati non presentano una forte correlazione con i rendimenti, tre di questi sono stati selezionati dalla lasso. Non compaiono mai inoltre due indici di sintesi (somma, media, o conteggio) riferiti alla stessa variabile "base".

Di seguito viene mostrata la tabella riepilogativa dei risultati ottenuti dal modello scelto e dal modello di benchmark

| | RMSE | RRMSE | R^2 |
|---|---|---|---|
| **Lasso** | 0.0697 | 0.93 | 0.13 |
| **Benchmark** | 0.0749 | 1.00 | 0.00 |

*Tabella 3-2 Performance Lasso & Benchmark - valori rendimenti Tesla*

Vediamo che la Lasso riesce ad ottenere delle performance migliori del modello privo di informazione, ma non di tanto. I due metodi presentano un RMSE molto simile, e quello associato al modello scelto è circa il 93% di quello del modello di benchmark. L'indice di $R^2$ viene migliorato di 13 punti percentuali rispetto al modello di confronto, che è pari approssimativamente a 0. Questo perché, essendo la serie storica stazionaria intorno alla media, il valore medio calcolato sul training set è circa uguale a quello calcolato sul test set.
Questi valori sono confermati dal grafico che confronta i valori previsti dei rendimenti con i valori veri nel test set.

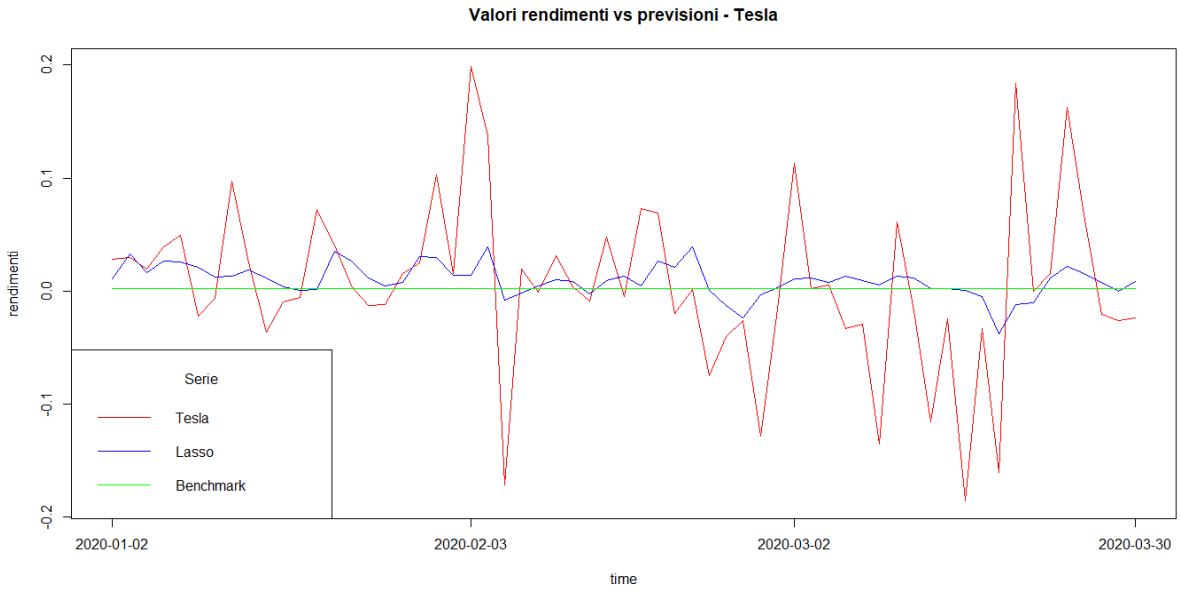


*Figura 3.5 Serie valori e previsioni - rendimenti Tesla*

Vediamo infatti che la previsione fornita dalla lasso riesce solo ad intuire l'andamento dei rendimenti Tesla, ma non a cogliere bene tutti i movimenti, soprattutto per quanto riguarda i vari picchi presenti nella serie.

In conclusione fra le informazioni riguardanti il sentimento delle persone estratte dai messaggi condivisi dagli utenti del social network Twitter e i rendimenti vi e sicuramente una certa correlazione, soprattutto in maniera concorrente e non ritardata, che fa intuire la connessione fra i mercati e ciò che la gente condivide. Tuttavia avendo effettuato l'analisi su base giornaliera, non sappiamo se sia stato il movimento dei mercati a provocare la reazione degli utenti e la condivisione dei messaggi o viceversa. E' chiaro solo che questa correlazione è presente, sebbene non abbastanza da riuscire a costruire un modello previsivo interessante sulla base di queste informazioni.

### *3.2.3. Previsione segno rendimenti Tesla*

Passiamo ora alla previsione del segno della serie dei rendimenti Tesla. La variabile risposta indicante il valore dei redimenti qui è stata rimpiazzata con una variabile dicotomica che assume il valore 1 quando il rendimento è positivo e 0 quando è negativo. Il numero di volte in cui il rendimento sale (163) e circa uguale al numero di volte in cui il rendimento scende (148), e l'andamento di questa nuova variabile è espresso nel grafico che segue:

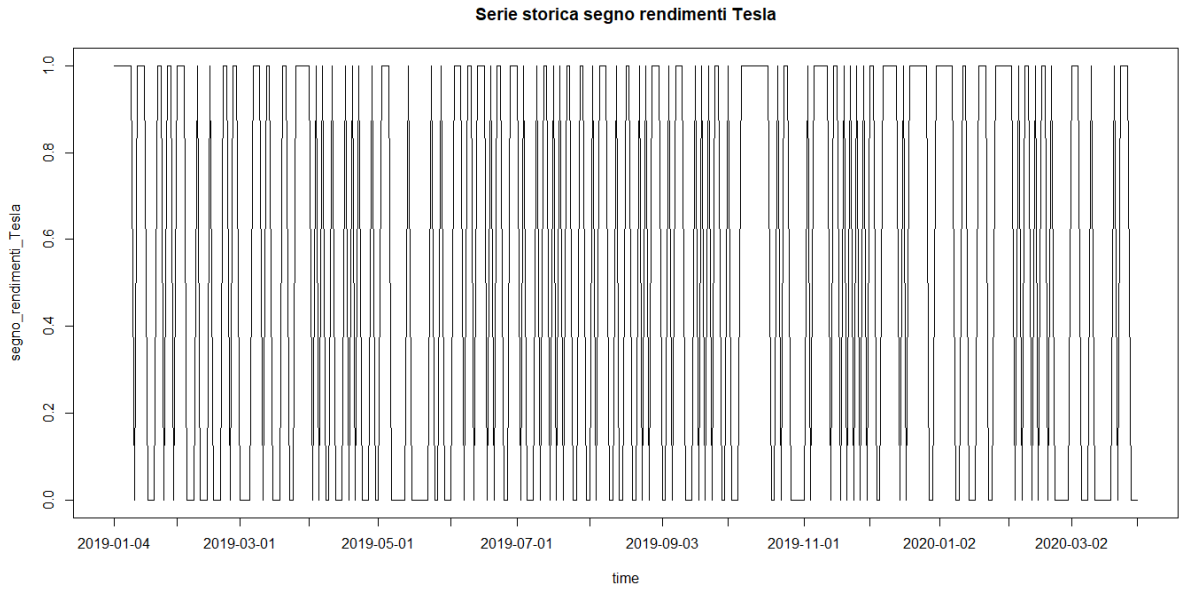


*Figura 3.6 Serie storica segno rendimenti Tesla*

E' interessante notare come il segno non si distribuisca in maniera alternata, ma vi sono periodi in cui è sempre positivo altri in cui è sempre negativo.

Data la natura dicotomica della nuova variabile risposta, per analizzare la correlazione fra questa e le variabili esplicative è stata scelto l'indice di Kendall. Anche qui vengono mostrate solo le variabili che presentano una correlazione superiore a |0.2|, ordinate in ordine decrescente.

| | Correlazione |
|---|---|
| sum_sentiment_score | 0.26 |
| mean_sentiment_score | 0.26 |
| mean_sentiment_score_pond | 0.22 |
| sum_sentiment_score_pond | 0.21 |
| mean_negative | -0.21 |

*Tabella 3-3 Variabili esplicative più correlate con segno rendimenti Tesla*

Vediamo che anche in questo caso le variabili che sintetizzano il sentimento su base giornaliera sono quelle che presentano più correlazione e che superano seppur di poco il 20%. E' presente un solo indice associato alla probabilità che il sentimento abbia un significato negativo, anche in questo caso con relazione negativa. In generale, il livello di correlazione si è abbassato rispetto al problema di regressione.

Data la numerosità delle variabili esplicative prima di applicare il modello scelto per prevedere il segno dei rendimenti è stata effettuata una selezione delle variabili. Per tenere conto della nuova tipologia della risposta indicante il segno dei rendimenti, è stata applicata una regressione logistica al problema, stimando i parametri ottimizzando una funzione di logverosimiglianza regolarizzata tramite una penalità di tipologia identica a quella applicata nella lasso (norma 1 del vettore dei parametri). In questo modo per lo stesso principio della regressione lasso è stata performata una *variable selection.* $\lambda$ è stato stimato utilizzando una *leave one out cross validation*, prendendo il valore che minimizza la metrica di errore. Di seguito viene riportato il grafico che mostra la riduzione del numero di variabili e l'andamento della GLM Deviance[116] all'aumentare del valore (del logaritmo) del parametro $\lambda$.

[116] I valori riportati nel grafico in GLM Deviance hanno una scala diversa rispetto ai veri valori della Devianza del modello calcolati con la formula espressa precedentemente. I valori della devianza associati ai vari lambda vengono infatti divisi per il numero di osservazioni della variabile risposta. https://stats.stackexchange.com/questions/117732/exact-definition-of-deviance-measure-in-glmnet-package-with-crossvalidation

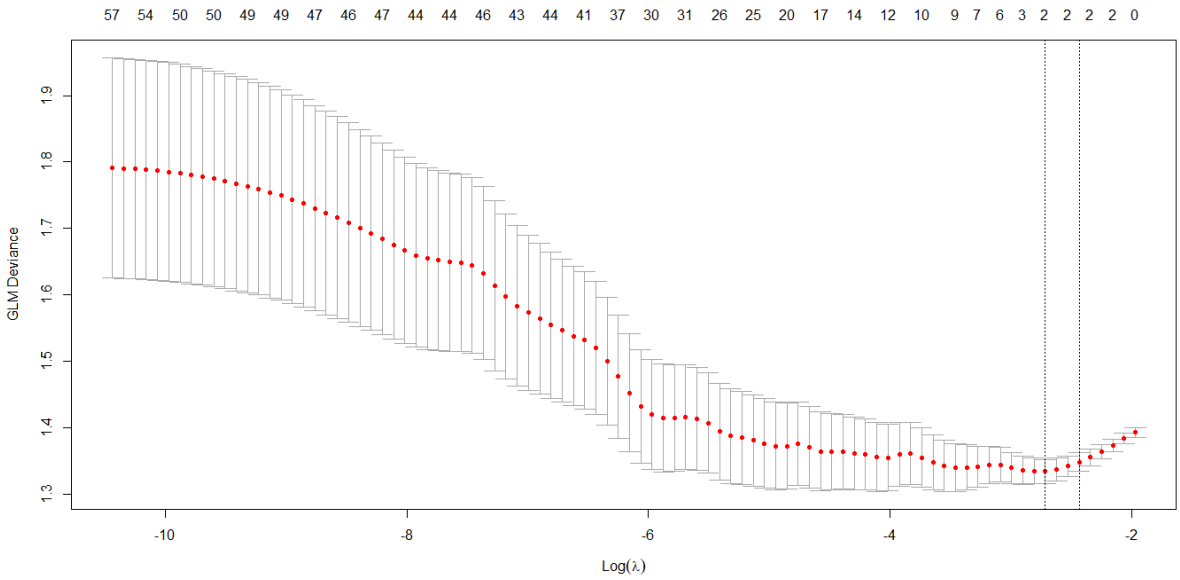


*Figura 3.7 Stima del parametro lambda - segno rendimenti Tesla*

Le variabili selezionate attraverso questo metodo sono state utilizzate per il problema di classificazione, e vengono riportate qui di seguito:

*sum_sentiment_score, mean_sentiment_score*

Si notano essenzialmente due cose:

- Vi è stata una forte riduzione del numero delle variabili, da 75 a 2, anche superiore alla diminuzione avvenuta nel problema di regressione
- Le variabili esplicative selezionate rappresentano essenzialmente la stessa informazione riepilogata in maniera diversa. Per questo è stato stato scelto di considerare solo una delle due per il problema di classificazione, cioè *sum_sentiment_score.*

Per prevedere il segno il modello che performava meglio è stata una SVM con kernel radial. I parametri utilizzati per il problema sono stati: $\gamma = 1, C = 1$, individuati tramite una *10 folds cross validation* sul training set. E' stata utilizzata una versione standardizzata della *sum_sentiment_score.* I risultati ottenuti sono espressi nella tabella di seguito

| | Accuracy | AUC |
|---|---|---|
| **SVM** | 0.721 | 0.718 |
| **SVM vs benchmark** | 0.221 | 0.218 |

*Tabella 3-4 Performance SVM vs Benchmark - segno rendimenti Tesla*

I valori della seconda riga sono stati ottenuti sottraendo 0.5 ai risultati ottenuti dalla SVM, cioè le performance ottenute dal classificatore random. Vediamo che le due metriche presentano valori molto simili. Il modello riesce a classificare correttamente più del 70% delle osservazioni, circa 20% in più del modello senza informazione. Il valore dell'AUC è anche qui più del 70%, indicando il comportamento del modello quando deve prevedere le perdite della serie dei rendimenti (circa un 20% in più del modello base), in relazione ai casi in cui il rendimento è positivo.

Dalla matrice di confusione espressa qui di seguito, notiamo che il modello riesce a prevedere meglio i segni positivi che quelli negativi. Gli ultimi vengono classificati in modo errato quasi la metà delle volte, contro i primi che vengono previsti correttamente quasi sempre. Infatti, la previsione è sbilanciata verso la classe positiva (42 previsioni positive vs 19 previsioni negative), a fronte di una parità nel numero delle vere classi.

| | pred_value_0 | pred_value_1 | Total |
|---|---|---|---|
| True_value_0 | 16 | 14 | 30 |
| True_value_1 | 3 | 28 | 31 |
| Total | 19 | 42 | 61 |

*Tabella 3-5 Matrice di confusione SVM - segno rendimenti Tesla*

Infine, viene riportato il grafico rappresentante la curva ROC associata al modello, dove l'area sottesa alla curva è pari al valore AUC espresso precedentemente.

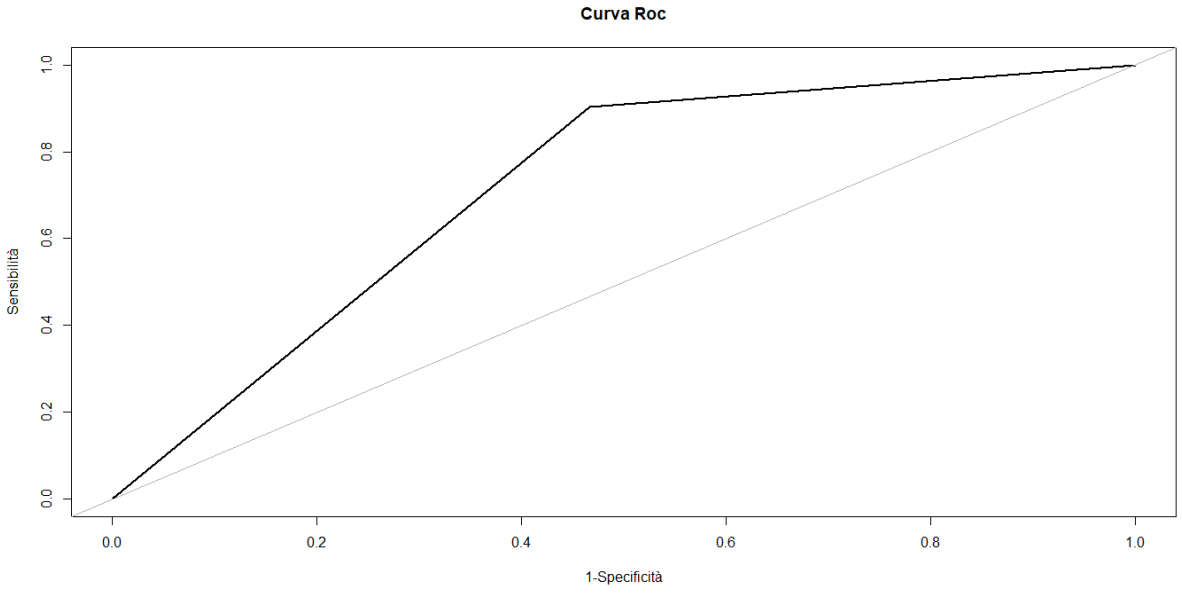


*Figura 3.8 Curva Roc SVM - segno rendimenti Tesla*

In conclusione, vediamo come tramite le informazioni ricavate da Twitter si riesce a prevedere meglio il segno dei rendimenti rispetto al loro valore, nonostante fra le variabili esplicative e la variabile risposta la correlazione sia sostanzialmente minore rispetto a quella osservata nel problema di regressione.

Il codice relativo all'analisi sia del segno che dei valori dei rendimenti Tesla si può trovare nella repository Github della tesi.[117]

### *3.2.4. Previsione valori extrarendimenti Tesla*

Le stesse informazioni sono state usate per prevedere sia il valore che il segno degli extrarendimenti Tesla, dove gli extrarendimenti sono stati calcolati nella maniera spiegata in precedenza. Di seguito viene mostrata la serie storica degli extrarendimenti.

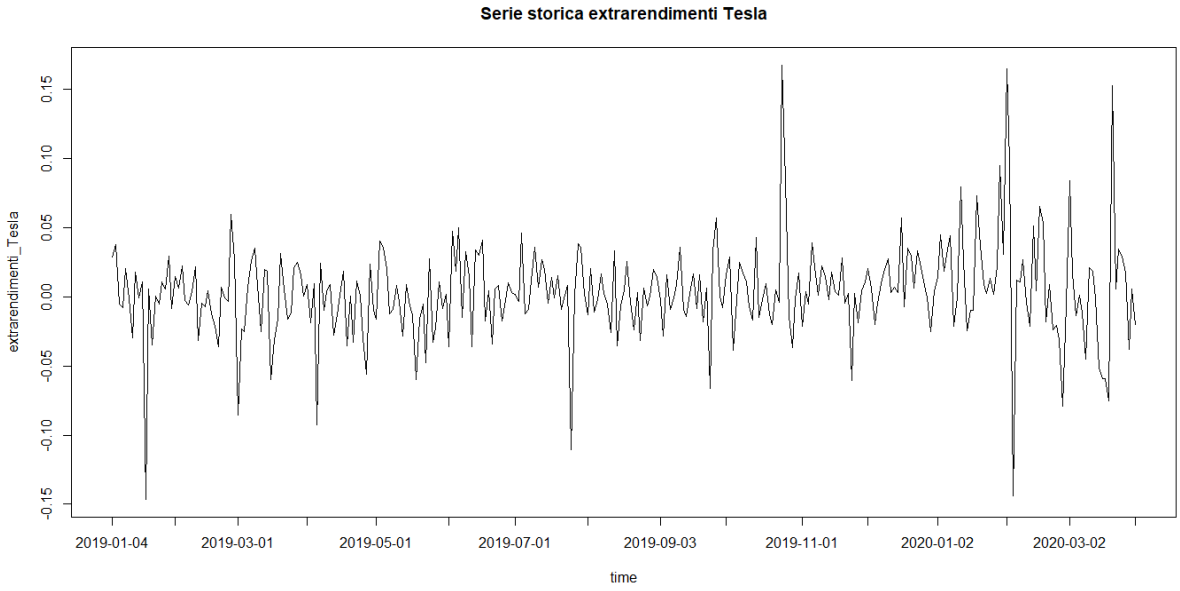


*Figura 3.9 Serie storica extrarendimenti Tesla*

Vediamo che la serie presenta un andamento simile a quella dei rendimenti, con i picchi principali ancora presenti, sebbene il loro valore sia stato leggermente diminuito con la sottrazione dei rendimenti CARZ. Questo ci indica che i rendimenti Tesla sono molto più dinamici di quelli del fondo. La serie appare un minimo più regolare, e questo è confermato anche dalla standard deviation marginale minore del processo (0.034). E' stazionaria intorno alla sua media, che anche qui è di circa 0 (0.003). Si è ridotta anche la volatilità della serie nel periodo che va da febbraio a marzo 2020 (0.058)

[117] https://github.com/Moreno-Sanna/Mythesis, file: "08_Creation aggregate dataset by financial date & Tesla yield analysis.R".

Per completezza viene riportata anche la serie dei rendimenti CARZ:

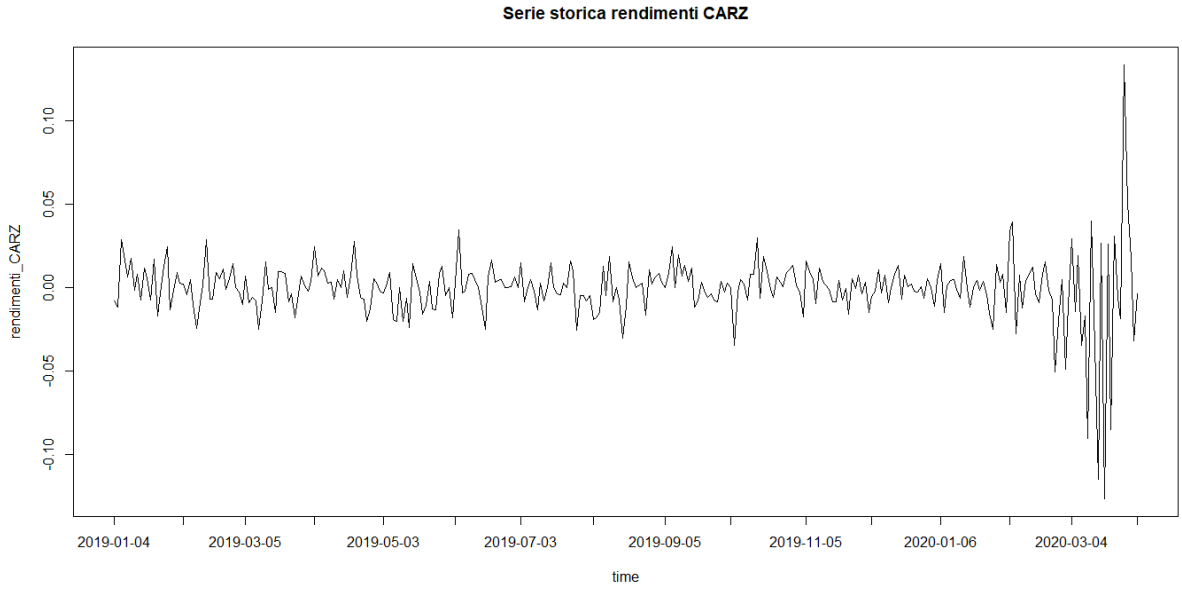


*Figura 3.10 Serie storica rendimenti CARZ*

La serie presenta una volatilità marginale pari a 0.019, inferiore a quella dei rendimenti Tesla ed infatti appare più regolare. Ad eccezione del periodo post inizio della pandemia covid, dove si vede nettamente l'incremento della variabilità del fondo (standard deviation di 0.046 nel periodo che va da febbraio a marzo 2020). La media è al solito prossima a 0.

Di seguito vengono elencate le variabili esplicative con una correlazione superiore o uguale a |0.2| con gli extrarendimenti, utilizzando per il calcolo l'indice di Pearson:

| | Correlazione |
|---|---|
| sum_sentiment_score | 0.50 |
| mean_sentiment_score | 0.45 |
| mean_positive | 0.39 |
| sum_sentiment_score_pond | 0.36 |
| mean_sentiment_score_pond | 0.36 |
| mean_positive_pond | 0.25 |
| n_positive | 0.22 |
| n_positive_pond | 0.21 |
| sum_negative | -0.25 |
| n_negative_pond | -0.25 |
| n_negative | -0.30 |
| mean_negative_pond | -0.31 |
| mean_negative | -0.35 |

*Tabella 3-6 Variabili esplicative più correlate con valori extrarendimenti Tesla*

Si nota che le variabili che presentano una correlazione maggiore sono le stesse che sono state trovate nel caso dei rendimenti, ma la sottrazione con i rendimenti CARZ ha portato ad un aumento, seppur limitato, del livello dell'indice. Dato che le variabili sono le stesse valgono le stesse considerazioni espresse durante il focus sulla correlazione dei rendimenti.

Prima di effettuare la previsione è stata applicata una variable selection, performata con una lasso con parametro $\lambda$ stimato tramite una *leave one out cross validation.* Di seguito il grafico che mostra l'andamento del MSE e del numero di variabili utilizzate all'aumentare del valore del logaritmo del parametro $\lambda$.

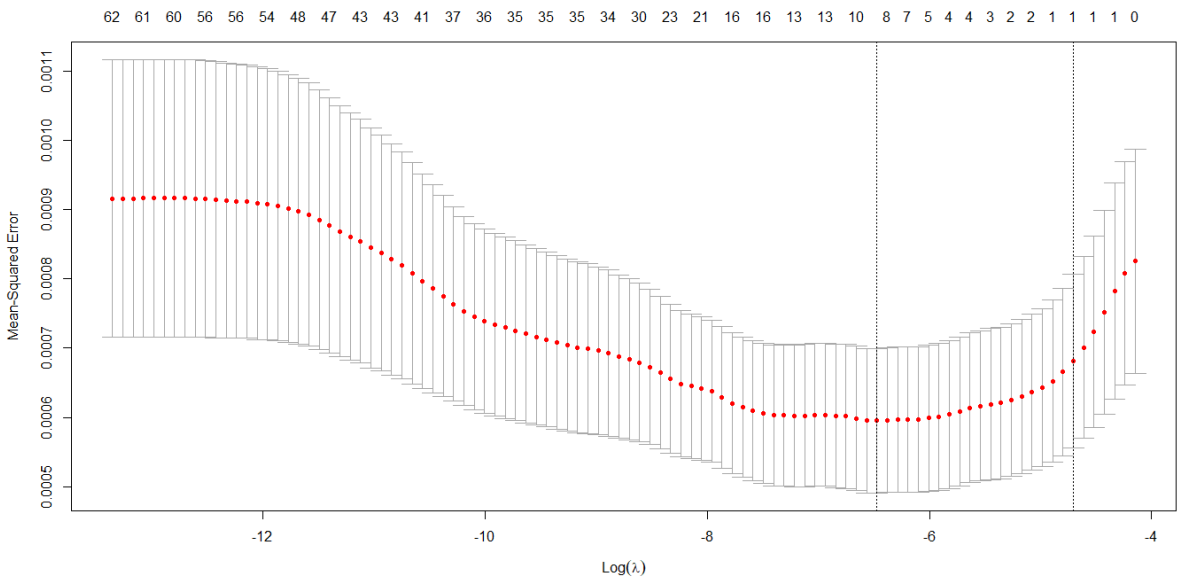


*Figura 3.11 Stima del parametro lambda - valori extrarendimenti Tesla*

Anche qui le variabili selezionate sono 8, molte di meno rispetto alle variabili totali, ma due in più rispetto al caso dei rendimenti. Sono state selezionate sia variabili ritardate che non, anche quelle che non hanno presentato una forte correlazione lineare. Di seguito l'elenco:

*n_neutral, sum_sentiment_score, mean_positive, mean_neutral_pond, mean_negative_lag1, mean_neutral_pond_lag1, n_retweet_lag2, rend_lag2*

Vediamo anche qui che non compare nessuna variabile collegata allo stesso indice non aggregato. Esattamente la metà delle variabili sono ritardate, ed a differenza del caso con i rendimenti qui compare anche l'extrarendimento ritardato di due periodi, che non è propriamente una variabile che esprime il livello di informazione derivante dai tweet. Da sola comunque ottiene delle performance molto peggiori rispetto a quelle ottenute quando usata insieme alle altre variabili (nel modello che a breve verrà presentato), quasi uguali a quelle del modello privo di informazione. Viene mantenuta perché nel complesso porta ad un lieve miglioramento delle metriche.

Il modello utilizzato è una SVM – regression con kernel polinomiale con i seguenti parametri: $\gamma = 1.5 * 10^{-4}, \epsilon = 0.5, coef0 = 3.5, degree = 5,\ C = 1$ [118]. Questi sono stati selezionati tramite una *10 fold cross validation* nel training set. I valori della variabile risposta e delle esplicative sono stati standardizzati prima di essere applicati al modello, ma le metriche sono

[118] Il kernel polinomiale applicato dal pacchetto svm in R è una generalizzazione di quello riportato in precedenza, dove al posto di sommare il valore 1 al prodotto interno viene sommato un parametro coef0 che è possibile scegliere.

state calcolate utilizzando i valori destandardizzati della variabile risposta e della sua previsione. Alle previsioni sono state aggiunte la media e la standard deviation della variabile risposta calcolate nel training set. Di seguito vengono riportate i valori delle metriche ottenute.

| | RMSE | RRMSE | R^2 |
|---|---|---|---|
| **SVM** | 0.0453 | 0.88 | 0.19 |
| **Benchmark** | 0.0513 | 1.00 | -0.03 |

*Tabella 3-7 Risultati SVM & Benchmark – valori extrarendimenti Tesla*

Vediamo un leggero miglioramento rispetto al caso dei rendimenti. L'RMSE è in generale sceso per entrambi i modelli, ma lo è anche il rapporto fra l'RMSE del modello scelto con quello di benchmark, che migliora di circa 5 punti percentuali. Anche l'indice $R^2$ passa dal 13% al 19%, e considerando che il modello di benchmark presenta un valore negativo, si ottiene un miglioramento del 22%.

Infine, viene riportato il grafico di confronto della serie storica con le previsioni.

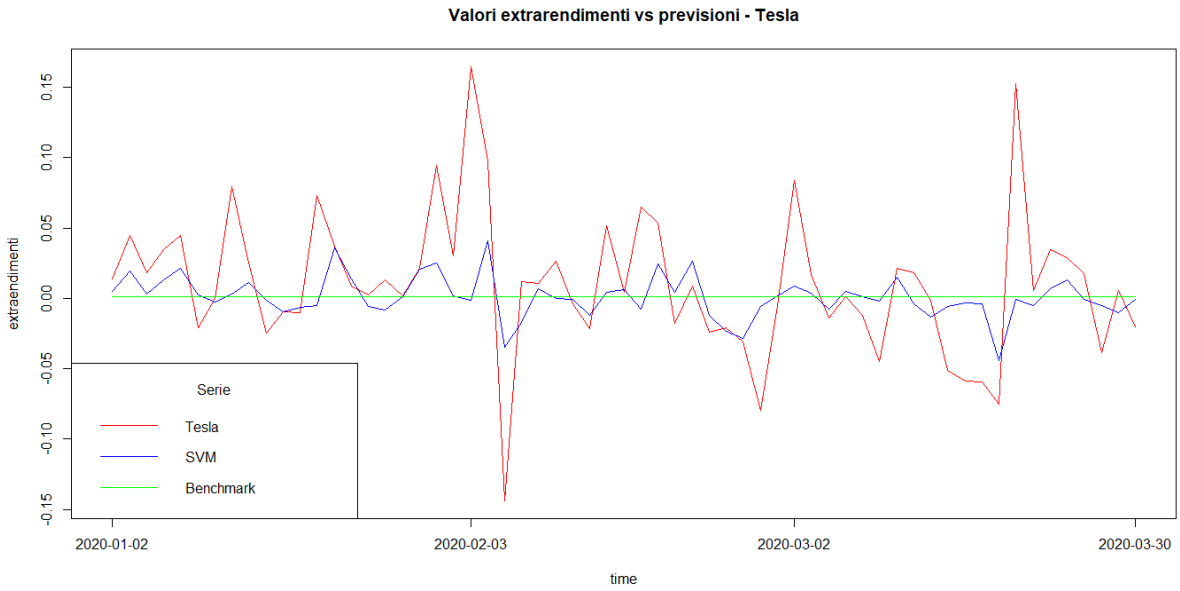


*Figura 3.12 Serie valori e previsioni - extrarendimenti Tesla*

Vediamo che le informazioni dateci dalle metriche vengono rispecchiate nel grafico. Adesso vi è un lieve miglioramento nel cogliere l'andamento delle serie, rispetto ai rendimenti puri, ma comunque il modello non riesce a prevedere i picchi in maniera adeguata.

In conclusione vi è un lieve miglioramento rispetto alla previsione dei rendimenti e rispetto all'utilizzo del modello di benchmark. Inoltre le informazioni ricavate dai tweet sono

lievemente più correlate adesso. Questo può indicare che l'attenzione generata dalle persone per il titolo porta a dei movimenti significativi anche al netto dell'andamento generale del mercato. Oppure, viceversa, che sono i rendimenti particolarmente distanti dall'andamento generale del mercato in quel momento a scatenare la reazione delle persone. Probabilmente questi due meccanismi avvengono simultaneamente.

### *3.2.5. Previsione segno extrarendimenti Tesla*

Come per i rendimenti, si è cercato di prevedere il segno della serie, in modo da cogliere quando l'extrarendimento è positivo e quando è negativo. Adesso, vi sono 134 extrarendimenti negativi contro 177 positivi, indicando che ci sono state volte in cui Tesla è andata in negativo meno rispetto all'andamento generale (sono aumentati i segni positivi). Di seguito il grafico dell'andamento del segno degli extrarendimenti.

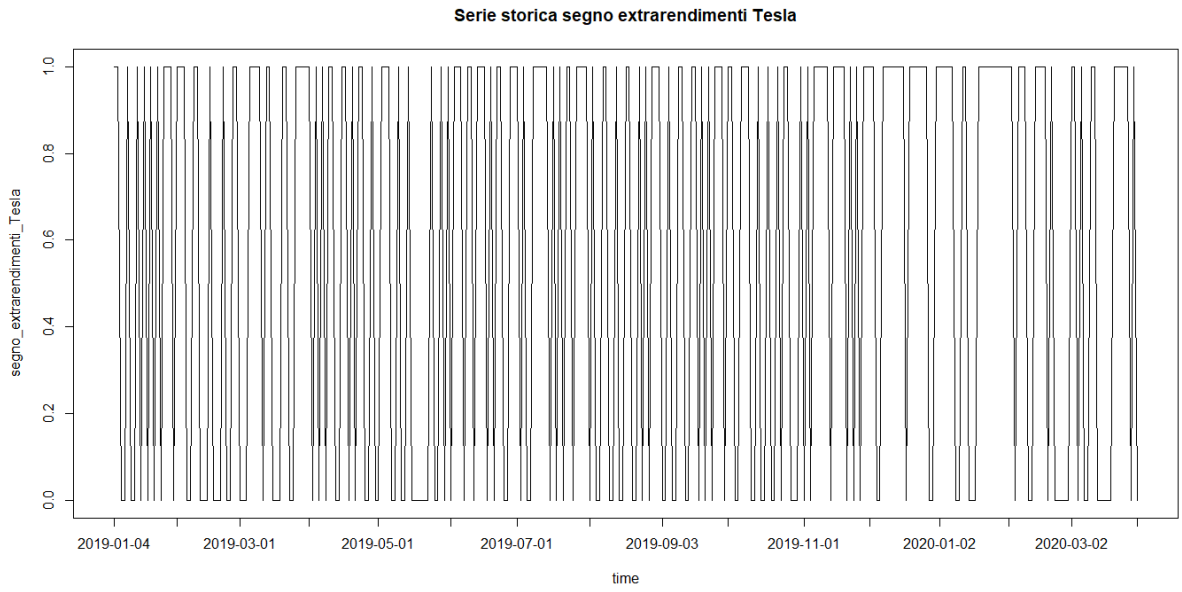


*Figura 3.13 Serie storica segno extrarendimenti Tesla*

Notiamo che anche qui sono frequenti i casi in cui ad un rendimento positivo ne segue un altro dello stesso segno, stessa cosa per i segni negativi.

Per calcolare la correlazione fra le variabili è stato utilizzato l'indice di kendall, come per il caso dei segni dei rendimenti puri. Di seguito la tabella riepilogativa.

| | Correlazione |
|---|---|
| sum_sentiment_score | 0.27 |
| mean_sentiment_score | 0.26 |
| sum_sentiment_score_pond | 0.22 |
| mean_sentiment_score_pond | 0.21 |
| mean_negative | -0.21 |

*Tabella 3-8 Variabili più correlate con segno extrarendimenti Tesla*

Notiamo che, come durante i problemi di regressione, le variabili esplicative che presentano una correlazione maggiore o uguale a |0.2| con la variabile risposta sono quelle trovate durante il problema di classificazione dei rendimenti. Questa volta però l'indice non migliora di molto e presenta più o meno gli stessi valori.

Anche qui è stata performata una variable selection utilizzando una regressione logistica penalizzata con la norma 1 del vettore dei paramentri, per via della natura dicotomica della variabile risposta. $\lambda$ è stato stimato utilizzando una *leave one out cross validation* e di seguito viene mostrato il grafico dell'andamento della metrica di errore e del numero di variabili esplicative selezionate in funzione dell'aumento del valore del parametro $\lambda$ .

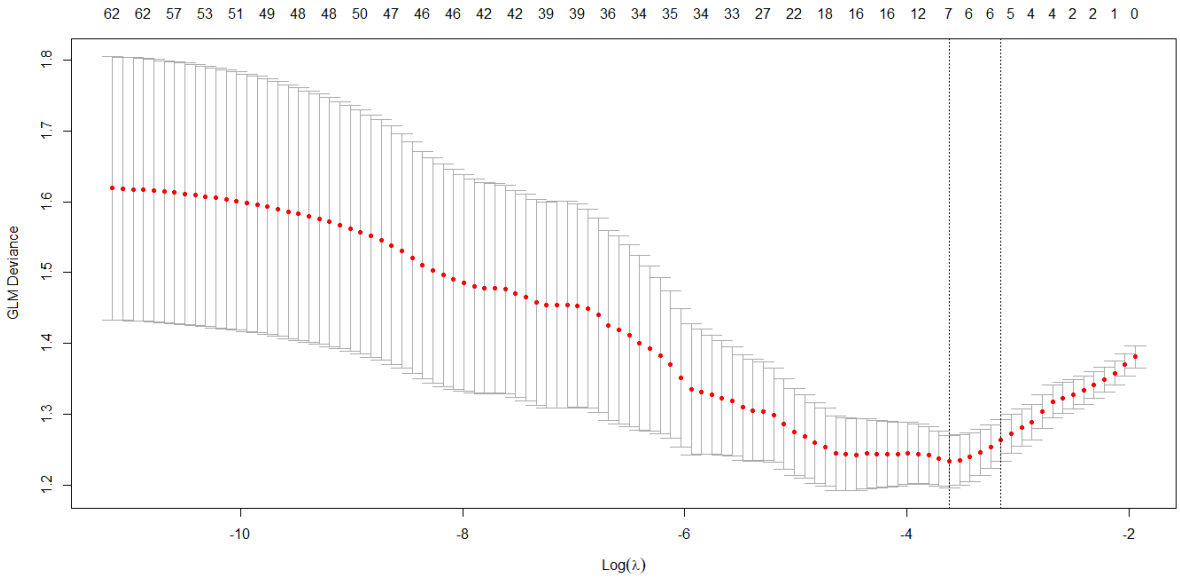


*Figura 3.14 Stima del parametro lambda - segno extrarendimenti Tesla*

Vediamo che il numero di variabili selezionate è più alto rispetto al problema di regressione. Queste vengono elencate di seguito:

*sum_sentiment_score, mean_sentiment_score, mean_sentiment_score_lag1, sum_negative_pond_lag2, n_positive_pond_lag2, sgn_lag1, sgn_lag2*

Il numero di variabili è stato ridotto da 75 a 7. Anche qui vengono selezionate insieme *sum_sentiment_score, e mean_sentiment_score,* che indicano la stessa informazione solo riepilogata in maniera diversa. Come prima verrà mantenuta solo la prima. Ora insieme alle altre variabili ritardate vengono selezionate anche il segno dell'extrarendimento ritardato di un periodo e di due periodi. Come nel precedente problema di regressione, queste non indicano delle informazioni derivanti dall'analisi dei tweet, e la loro presenza non è la principale spiegazione delle performance del modello, ma lievemente le migliorano. Per questo motivo verranno considerate lo stesso.

Il modello scelto per prevedere la risposta è lo stesso usato per la previsione dei segni dei rendimenti, cioè una SVM con kernel radiale applicata alle variabili precedentemente selezionate. I parametri utilizzati sono $\gamma = \frac{1}{6}, C = 1$ individuati tramite una *10 folds cross validation* nel training set, e le variabili sono state standardizzate prima di essere utilizzate nel modello. Di seguito i valori delle metriche ottenuti dal modello.

| | Accuracy | AUC |
|---|---|---|
| **SVM** | 0.754 | 0.731 |
| **SVM vs benchmark** | 0.254 | 0.231 |

*Tabella 3-9 Risultati SVM vs Benchmark - segno extrarendimenti Tesla*

Notiamo di nuovo un lieve aumento delle performance rispetto all'analisi sui rendimenti. Il modello riesce a cogliere circa il 75% dei segni del test set, il 25% in più del modello random. Non si comporta male nemmeno in termini di AUC, presentando un valore pari a 73% circa.

Di seguito la matrice di confusione:

| | pred_value_0 | pred_value_1 | Total |
|---|---|---|---|
| **True_value_0** | 15 | 9 | 24 |
| **True_value_1** | 6 | 31 | 37 |
| **Total** | 21 | 40 | 61 |

*Tabella 3-10 Matrice di confusione SVM - segno extrarendimenti Tesla*

Il modello riesce a prevedere correttamente la maggior parte dei segni indipendentemente dalla classe di appartenenza, anche se continua a fare più fatica con i segni negativi. Anche qui vi è

una numero di classi previste positive maggiori rispetto a quelle negative. Adesso tuttavia, sono presenti più classi positive nei veri valori.

Infine, viene riportato il grafico della curva ROC

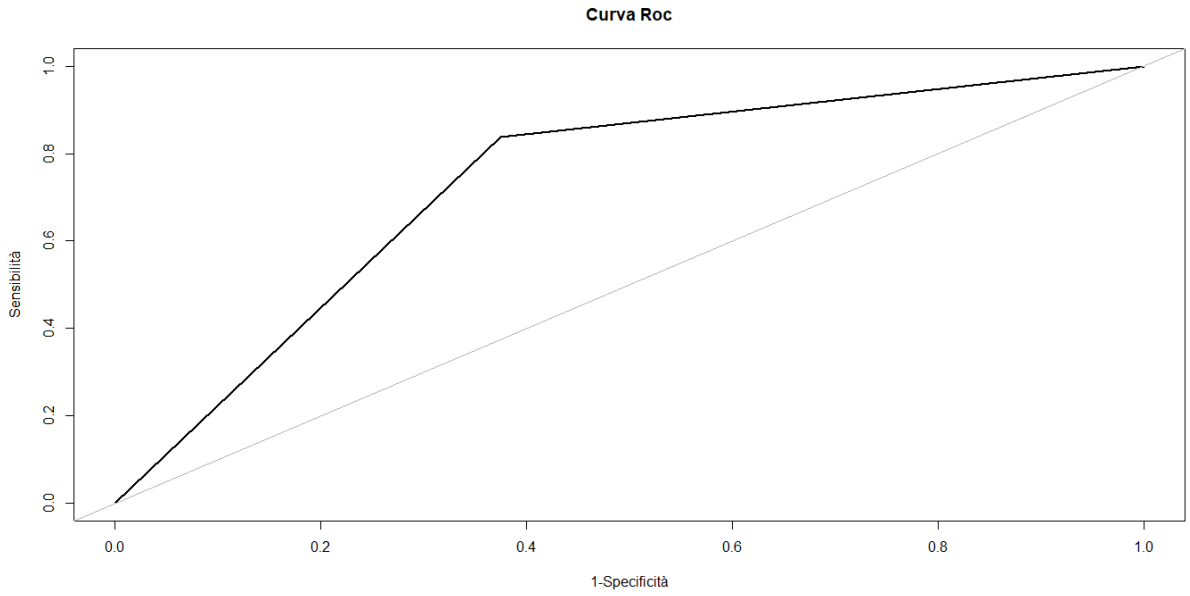


*Figura 3.15 Curva Roc SVM - segno extrarendimenti Tesla*

L'area sottesa alla curva è pari al valore dell'AUC riportato nelle tabella delle metriche. In conclusione, osserviamo che lo stesso modello utilizzato per i rendimenti performa meglio, indicando il maggior legame delle informazioni con gli extrarendimenti.

Il codice relativo all'analisi sia del segno che dei valori degli extra rendimenti Tesla si può trovare nella repository Github della tesi[119]

## 3.3. Analisi e risultati asset BMW

### *3.3.1. Introduzione*

Come per l'asset Tesla anche per BMW le informazioni ricavate dai tweet connessi sono state utilizzate per cercare di prevedere il valore e i segni dei rendimenti ed extrarendimenti del titolo. Prima di mostrare i risultati ottenuti, verrà fatto un breve focus sull'indice che meglio rappresenta le informazioni ricavate dai messaggi degli utenti, la media del livello di sentimento giornaliero considerando anche i retweet. Di seguito il grafico che mostra la serie storica di questo indice.

[119] https://github.com/Moreno-Sanna/Mythesis, file: "09_Creation aggregate dataset by financial date & Tesla extra yields analysis.R".

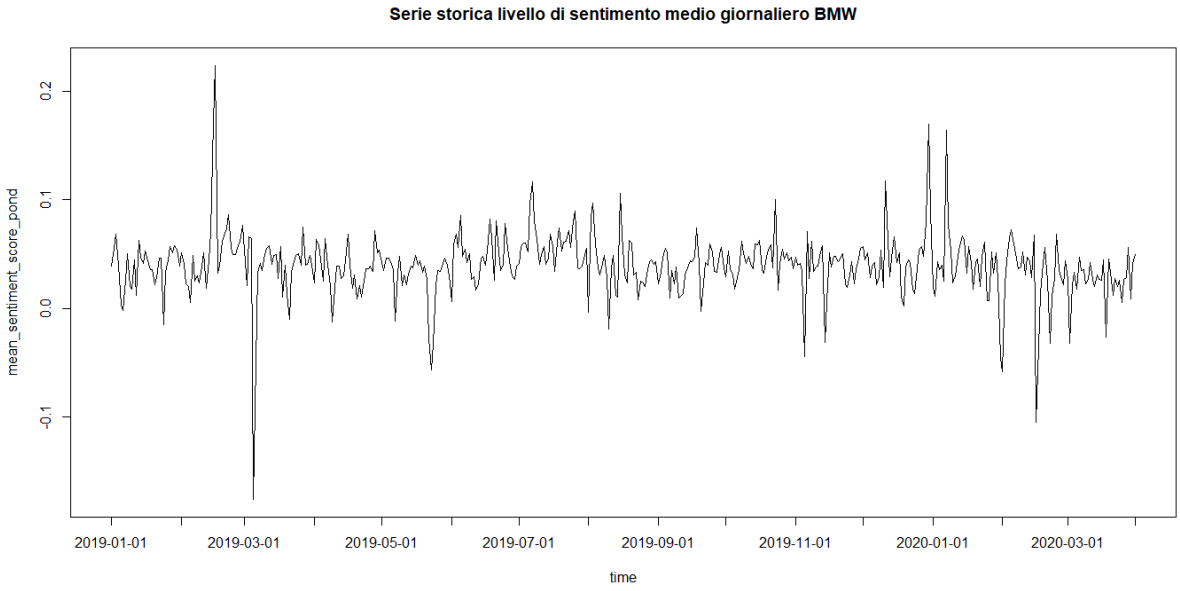


*Figura 3.16 Serie storica sentimento medio giornaliero con retweet - BMW*

E' una serie storica stazionaria intorno alla sua media, che anche se è prossima a 0 è leggermente positiva (0.04), a differenza di quella caratterizzante il sentimento di Tesla. Questo ci indica che il sentimento delle persone non è cambiato in media nel periodo osservato. E' caratterizzata da due grandi picchi, uno in positivo e l'altro negativo, rispettivamente intorno a meta febbraio 2019 e agli inizi di marzo dello stesso anno. Non sembra aumentare la sua variabilità nel periodo coincidente l'inizio della pandemia, fatta eccezione per due picchi negativi non troppo profondi, che potrebbero essere correlati a questo. La standard deviation di lungo periodo della serie è pari a 0.029, più bassa di quella di Tesla.

### *3.3.2. Previsione valori rendimenti BMW*

Di seguito viene riportata la serie storica dei valore dei rendimenti BMW:

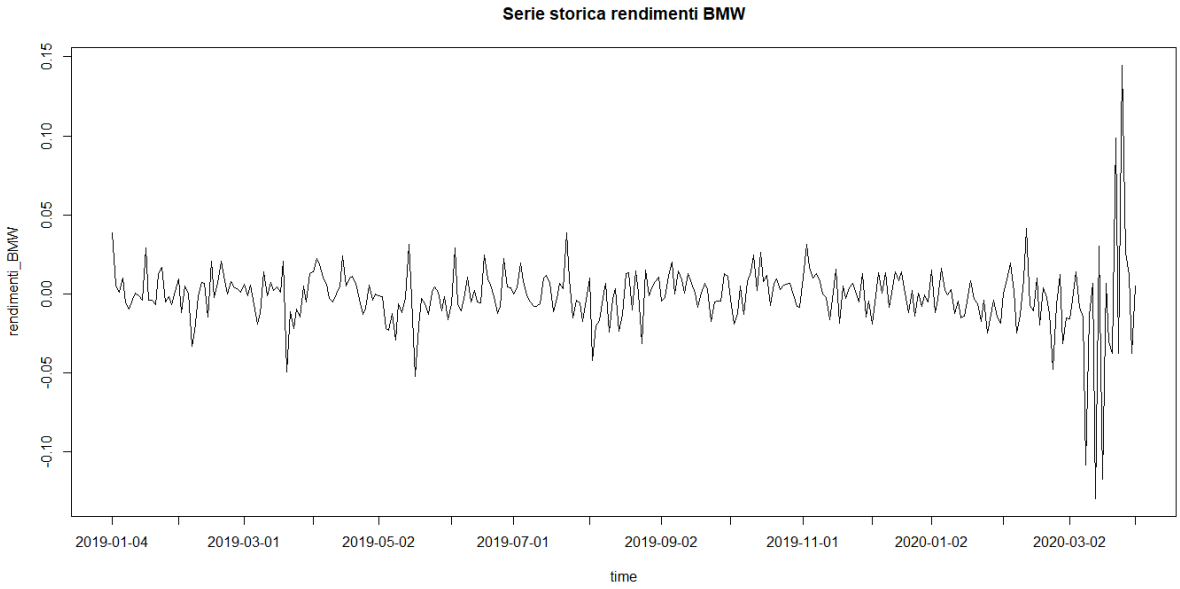


*Figura 3.17 Serie storica valori rendimenti BMW*

Si presenta come una serie storica abbastanza regolare, stazionaria con una media di lungo periodo intorno allo 0 (precisamente leggermente negativa, -0.001) e una standard deviation marginale di 0.021, circa la metà di quella di Tesla. Si nota chiaramente il cambio di andamento causato dal covid, qui un po' più in ritardo rispetto a Tesla, a partire da marzo 2020. (la volatilità della serie in questo periodo è pari a 0.064).

Di seguito vengono mostrate le variabili esplicative più correlate con la serie, ordinate in maniera decrescente. La correlazione è stata calcolata utilizzando l'indice di correlazione di Pearson.

| | Correlazione |
|---|---|
| rend_lag2 | 0.25 |
| mean_sentiment_score_pond | 0.10 |
| mean_sentiment_score_pond_lag1 | 0.10 |
| sum_sentiment_score_pond | 0.09 |
| sum_sentiment_score_pond_lag1 | 0.09 |
| mean_negative_pond_lag1 | -0.08 |

*Tabella 3-11 Variabili esplicative più correlate con valori rendimenti BMW*

Il livello di correlazione scende drasticamente rispetto a quanto osservato per Tesla. Qui sono state mostrate solo le variabili che presentano una correlazione maggiore o uguale a |0.08|, dato che solo il rendimento ritardato di due periodi (che tra l'altro non rappresenta della

informazione estratta dai tweet) presenta un livello maggiore del 20%, e solo la media del sentimento giornaliero considerando anche i retweet supera il valore del 10%.

Per cercare di prevedere comunque la serie, si è proceduto nel solito modo, operando prima una variable selection e poi applicando il modello scelto. La variable selection è stata performata tramite una lasso, dove il parametero $\lambda$ è stato scelto tramite una *leave one out cross validation*. Di seguito il grafico riportato anche nei precedenti casi che mette in relazione i valori del parametro con la metrica che misura l'errore ed il numero di variabili selezionate:

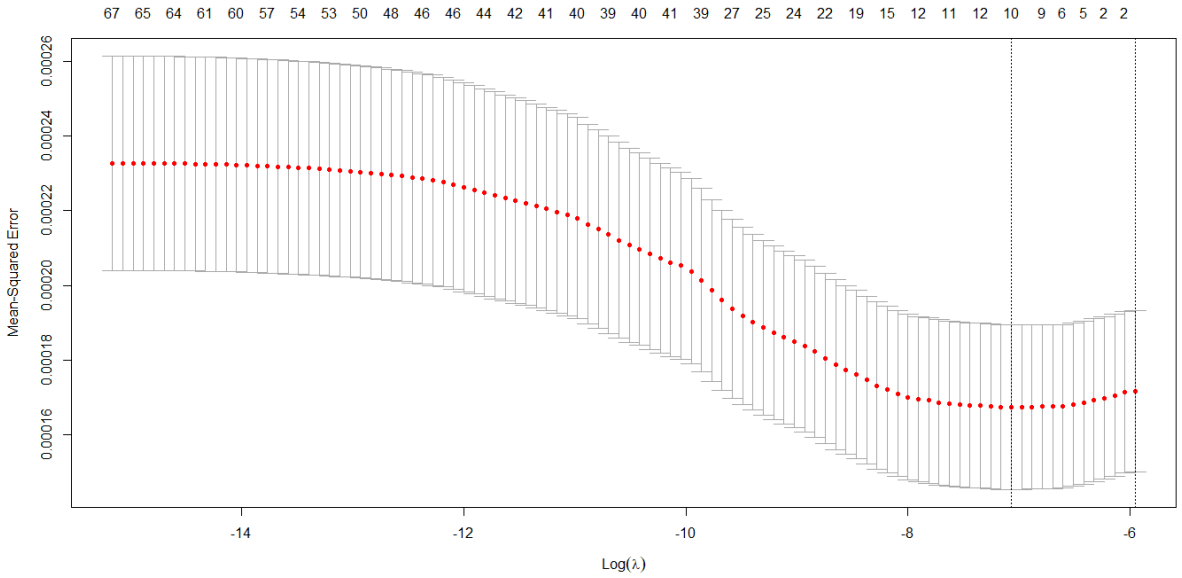


*Figura 3.18 Stima del parametro lambda – valori rendimenti BMW*

Le variabili selezionate sono:

*sum_negative, mean_negative, sum_negative_lag1, mean_negative_lag1, mean_sentiment_score_pond_lag1, rend_lag1, n_positive_pond_lag2, mean_negative_lag2, mean_sentiment_score_lag2, mean_sentiment_score_pond_lag2*

Vediamo che vengono selezionate solo due variabili senza ritardi, che inoltre esprimono la stessa informazione solo sintetizzata in maniera diversa. Nelle variabili ritardate vengono selezionati sempre gli indicatori del sentimento medio giornaliero, sia considerando i retweet sia senza.

Nessuno dei modelli provati riesce ad ottenere delle performance significative nella previsione dei rendimenti. Tutti quanti performano peggio o in maniera simile del modello che prevede tutto sulla base della media della serie calcolata sul training set. Per completezza verranno riportate le metriche risultanti dalla lasso con la quale si è operata la variable selection.

| | RMSE | RRMSE | R^2 |
|---|---|---|---|
| **Lasso** | 0.0391 | 1.02 | -0.09 |
| **Benchmark** | 0.0382 | 1.00 | -0.04 |

*Tabella 3-12 Risultati Lasso & Benchmark - valori rendimenti BMW*

Come si è detto le metriche identificano un modello peggiore di quello di benchmark, con un RMSE leggermente superiore e $R^2$ inferiore di 5 punti percentuale. Di seguito il grafico che mette a confronto la serie dei rendimenti e le sue previsioni.

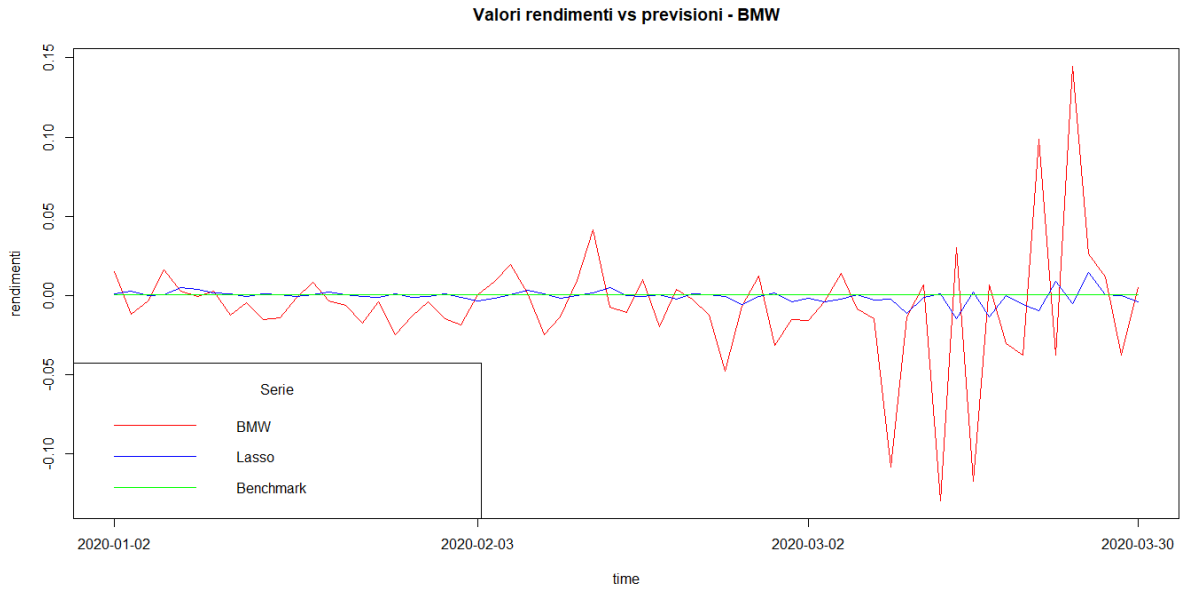


*Figura 3.19 Serie valori e previsioni - rendimenti BMW*

Come si è visto dai risultati presentati, non sembra esserci una forte correlazione fra le informazioni ricavate dai Tweet connessi a BMW e il suo andamento sui mercati. La poca relazione presente non è sufficiente per poter costruire un modello che sia significativo nella previsione dei rendimenti nel futuro.

### *3.3.3. Previsione segno rendimenti BMW*

Si è provato lo stesso a prevedere il segno dei rendimenti nonostante le scarse performance nel problema di regressione. Al solito viene mostrato di seguito l'andamento dei segni nel corso del tempo.

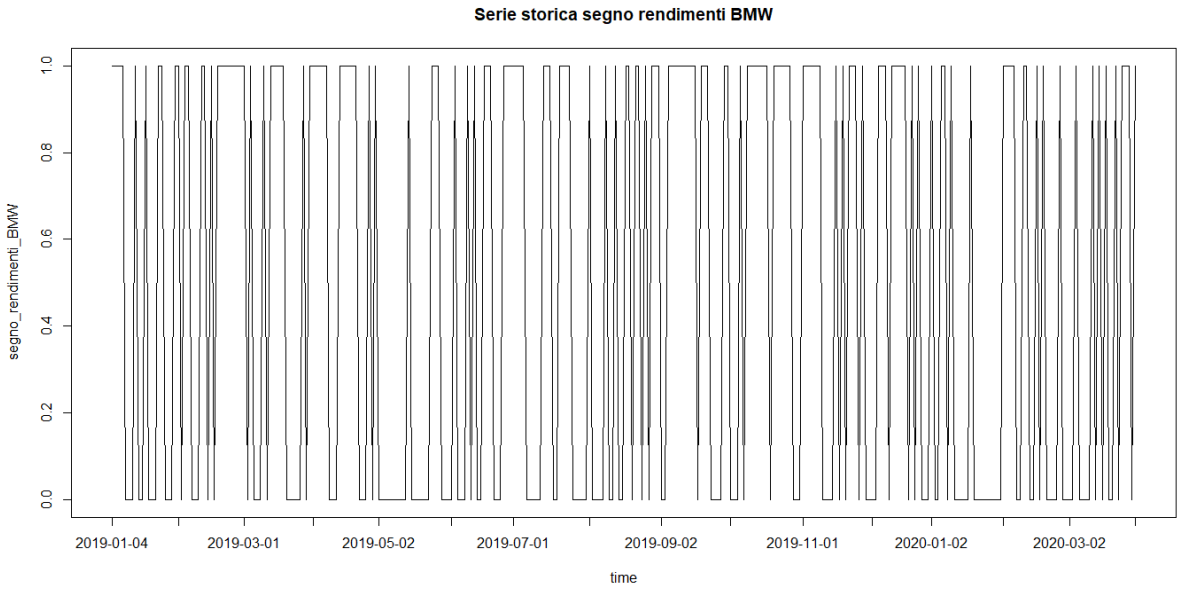


*Figura 3.20 Serie storica segno rendimenti BMW*

Qui a differenza di quanto osservato per Tesla si nota di più la successione non alternata dei segni. Frequentemente ad un segno negativo ne segue uno negativo e ad un segno positivo ne segue uno positivo. Il numero di segni negativi è più o meno pari al numero di segni positivi (158 vs 154).

Di seguito verranno mostrate le variabili più correlate con la serie dei segni, cioè quelle che superano o raggiungono almeno un livello di correlazione pari allo |0.08|. Verrà usato l'indice di Kendall, per le motivazioni espresse più volte in precedenza.

| | Correlazione |
|---|---|
| sgn_lag1 | 0.18 |
| rend_lag1 | 0.17 |
| sgn_lag2 | 0.13 |
| mean_sentiment_score_pond_lag1 | 0.10 |
| sum_neutral | 0.09 |
| n_neutral | 0.09 |
| n_Tweet | 0.09 |

*Tabella 3-13 Variabili esplicative più correlate con segno rendimenti BMW*

Anche qui non si notano forti relazioni fra le variabili che esprimono le informazioni ricavate dai tweet e i segni dei rendimenti BMW. Solo la media del sentimento giornaliero, considerando anche i retweet, ritardata di un periodo riesce a raggiungere il 10 % di correlazione.

Per cercare di prevedere i segni il modello scelto, quello che performava meglio, è una random forest senza variable selection, con parametro $m = 8$ selezionato tramite una *10 folds cross validation* nel training set. Di seguito le performance ottenute.

| | Accuracy | AUC |
|---|---|---|
| Random Forest | 0.603 | 0.53 |
| Random Forest vs benchmark | 0.103 | 0.03 |

*Tabella 3-14 Risultati Radom Forest vs Bechmark - segno rendimenti BMW*

A differenza del problema di regressione, adesso il modello scelto riesce a performare in maniera leggermente migliore rispetto al modello privo di informazione. Riesce a prevedere correttamente circa il 60% dei casi, e presenta una AUC del 53%, comunque superiore al modello random. Per dare un'indicazione delle variabili più importanti utilizzate nel modello, verranno presentate le prime otto che, se tolte, fanno decrescere l'accuracy del modello in misura maggiore[120], ordinate in maniera decrescente.

| | Mean Decrease Accuracy |
|---|---|
| rend_lag1 | 6.576250e-03 |
| sum_neutral | 4.402423e-03 |
| n_Tweet | 3.385141e-03 |
| mean_positive_pond_lag1 | 3.205175e-03 |
| n_neutral | 2.082047e-03 |
| sum_negative_lag1 | 1.655190e-03 |
| sum_neutral_pond | 1.495832e-03 |
| n_neutral_pond | 1.330577e-03 |

*Tabella 3-15 Variable importance - segno rendimenti BMW*

Si nota che la variabile più importante, seguendo questa metrica, non è indicativa delle informazioni estratte dai tweet connessi a BMW. Le altre tuttavia, sono relative a questo.

Di seguito la matrice di confusione del classificatore:

[120] La Mean Decreasing Accuracy riferita ad un variabile viene calcolata sottraendo all'errore ottenuto dal modello che utilizza quella esplicativa l'errore ottenuto dal modello che utilizza quella variabile permutata, normalizzando il tutto con la deviazione standard della differenza. Più una variabile produce una riduzione dell'accuracy, più è importante

| | pred_value_0 | pred_value_1 | Total |
|---|---|---|---|
| True_value_0 | 32 | 8 | 40 |
| True_value_1 | 17 | 6 | 23 |
| Total | 49 | 14 | 63 |

*Tabella 3-16 Matrice di confusione Random Forest - segno rendimenti BMW*

Dalla matrice di confusione si vede che il classificatore prevede maggiormente le classi con segno negativo piuttosto che quelle con segno positivo. Questo porta ad una classificazione sbagliata di gran parte dei segni positivi, che anche se può essere positivo in un'ottica di prevedere le perdite in ogni caso rinunciando alla previsione dei segni positivi, non è una classificazione ottimale.

Infine, di seguito viene mostrato il grafico contente la Curva ROC.

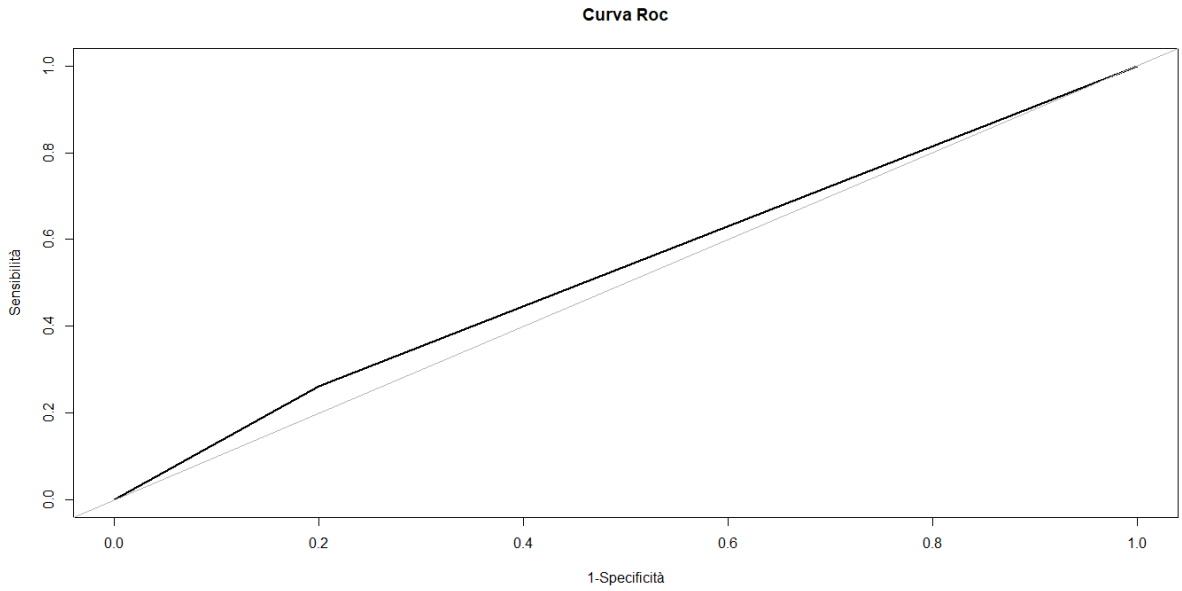


*Figura 3.21 Curva Roc Random Forest - segno rendimenti BMW*

Essendo l'area sottesa alla curva pari a 0.53, la curva ROC è molto vicina alla bisettrice, indicandoci che il classificatore come performance è molto simile al classificatore random.

In conclusione, le informazioni ricavate dai tweet riescono a prevedere meglio i segni dei rendimenti piuttosto che i rendimenti stessi, anche se con performance di poco superiori rispetto al modello di benchmark. Le variabili più correlate con la serie infatti sono quelle che non contengono informazioni ricavate dai tweet, e una di queste è il regressore più importante del modello scelto. Non sembra ci sia quindi, una particolare connessione fra le informazioni che gli utenti condividono e l'andamento dei mercati.

Il codice relativo all'analisi sia del segno che dei valori dei rendimenti BMW si può trovare nella repository Github della tesi[121]

### *3.3.4. Previsione valori extrarendimenti BMW*

Successivamente, si è cercato di prevedere sempre con le informazioni ricavati dai tweet, la serie storica degli extrarendimenti BMW, calcolati allo stesso modo di quanto fatto per Tesla, sottraendo a questi la serie storica dei rendimenti CARZ. Verrà prima presentato il problema di regressione e poi quello di previsione.

Di seguito viene mostrata la serie storica degli extrarendimenti BMW da prevedere:

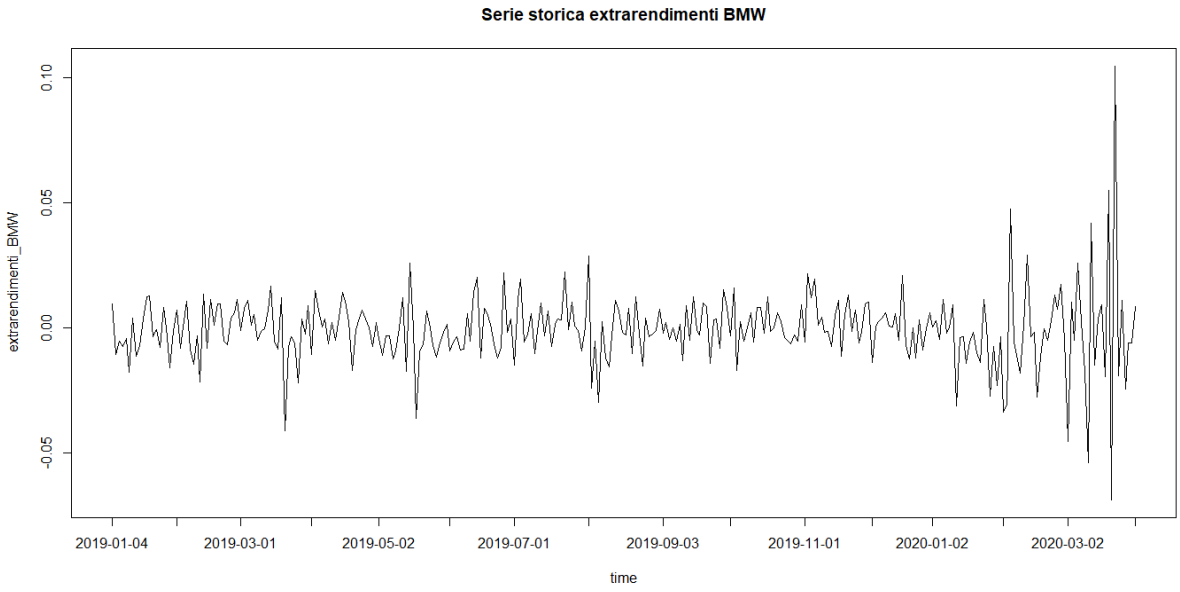


*Figura 3.22 Serie storica valori extrarendimenti BMW*

Notiamo che la sottrazione con i rendimenti del fondo CARZ regolarizza un minimo la serie BMW soprattutto nel tratto finale caratterizzato dall'elevata volatilità dovuta allo scoppio del covid. Per il resto la serie non presenta grossi picchi né in positivo né in negativo, fatta eccezione per quelli che si verificano da febbraio 2020. La serie è sempre stazionaria, con una media intorno allo 0 (-0.001) e una standard deviation pari a 0.014, più bassa rispetto a quella dei rendimenti (0.038 nel periodo che va da febbraio a marzo 2020).

Dato che il modello scelto per la previsione dei rendimenti è un modello econometrico, verranno inserite di seguito anche i grafici rispettivamente dell'autocorrelazione e dell'autocorrelazione parziale della serie.

[121] https://github.com/Moreno-Sanna/Mythesis, file=" 10_Creation aggregate dataset by financial date & BMW yield analysis.R".

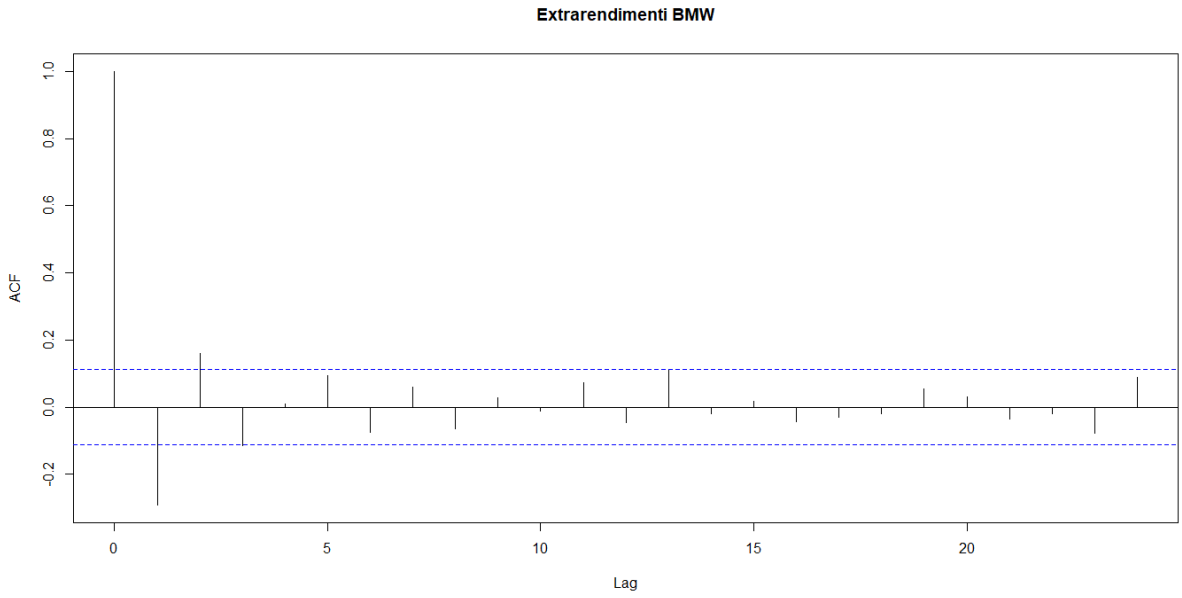


*Figura 3.23 Funzione di autocorrelazione - valori extrarendimenti BMW*

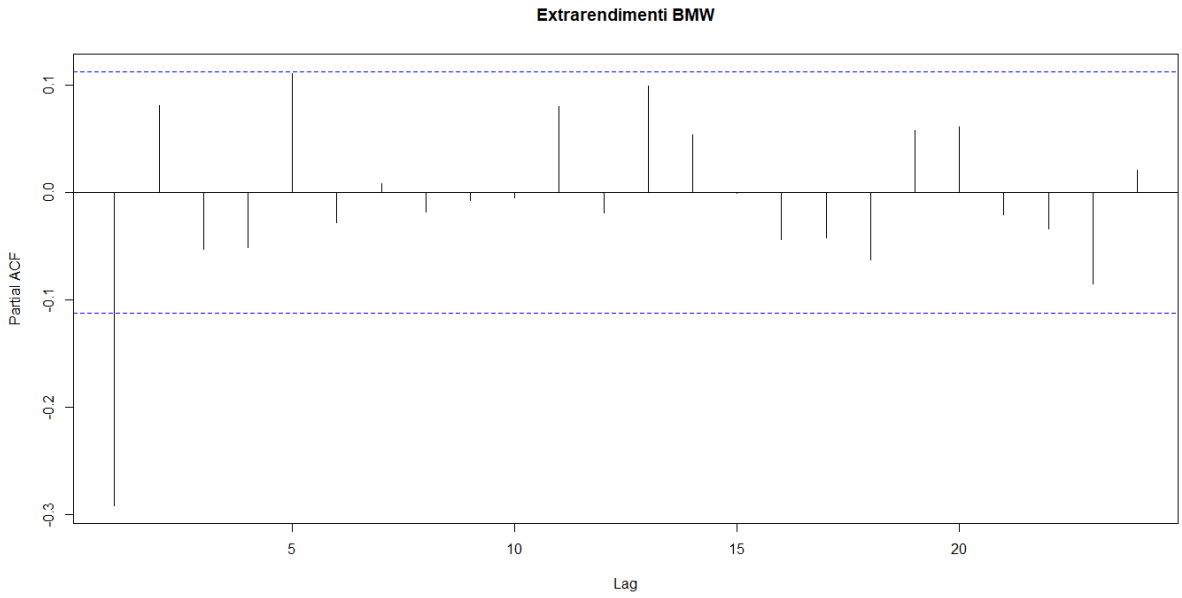


*Figura 3.24 Funzione di autocorrelazione parziale - valori extrarendimenti BMW*

Dai grafici notiamo che la funzione di autocorrelazione decresce con una combinazione di onde sinosuidali mentre la funzione di autocorrelazione parziale si annulla dopo il primo lag. Seguendo la procedura di Box-Jenkins[122], questo ci induce a pensare ad un modello autoregressivo di ordine uno per tentare di spiegare almeno la media della serie degli extrarendimenti.

[122] Lisa Crosato *Serie Storiche Economiche*. Dispesa del corso di Introduzione alla serie storiche

Per l'analisi della correlazione viene riportata di seguito la tabella indicante le variabili maggiormente correlate con la serie degli extrarendimenti. Dato che in generale la correlazione non è altissima, sono state prese le variabili che possiedono una correlazione pari o maggiore di |0.08|

| | Correlazione |
|---|---|
| rend_lag2 | 0.16 |
| mean_sentiment_score_pond_lag1 | 0.11 |
| mean_negative_lag2 | 0.11 |
| sum_sentiment_score_pond_lag1 | 0.09 |
| n_negative_lag2 | 0.08 |
| mean_sentiment_score_lag2 | -0.09 |
| mean_negative_pond_lag1 | -0.09 |
| rend_lag1 | -0.29 |

*Tabella 3-17 Variabili esplicative più correlate con valori extrarendimenti BMW*

Notiamo che le variabili più correlate sono quelle che non contengono informazione relativa ai Tweet degli utenti, che invece superano di poco il 10% di correlazione. Il grado di connessione della variabile risposta è infatti aumentato con i ritardi degli extrarendimenti rispetto a quello osservato per i rendimenti puri.

Prima di applicare il modello scelto è stata effettuata una variable selection, questa volta con il metodo Backward spiegato in fase di presentazione dei metodi utilizzati, con parametro k=1. Le variabili trovate sono le seguenti:

*mean_positive_lag2, mean_negative_lag2, mean_negative_pond, rend_lag1, sum_positive_pond, sum_neutral_lag1, n_Tweet_lag1, sum_negative_lag1, sum_positive_lag1, sum_neutral_pond, n_tweet_totali, sum_sentiment_score_pond, n_positive_pond_lag2, n_retweet_lag2, n_neutral_pond_lag2, n_negative_pond_lag2, n_negative_lag2, n_positive_lag2, n_neutral_lag2, sum_neutral_lag2, sum_positive_lag2, sum_negative_lag2, n_positive_pond_lag1, n_tweet_totali_lag1, n_neutral_pond_lag1, n_negative_pond_lag1, sum_negative, n_negative, sum_positive_pond_lag1, sum_sentiment_score_pond_lag1*

Il metodo seleziona più variabili rispetto alla lasso, per un totale di 33 regressori su 75. Sono presenti sia variabili ritardate che non, ed è presente solo il primo lag dei rendimenti. La variabile mean_negative_lag2 è stata esclusa nel modello in quanto non significativa.

Questi sono stati i regressori esterni applicati al modello econometrico spiegato precedentemente, composto da un AR(1) appunto con variabili esterne, dove gli errori seguono un modello exponential Garch (0,1), con solo la parte autoregressiva che modella la varianza condizionata. Il termine stocastico del modello segue una distribuzione ged. Prima di mostrare le performance ottenute viene riportato l'output del modello. Per prima cosa vengono mostrati i parametri stimati.

```
*----------------------------------*
*          GARCH Model Fit         *
*----------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : eGARCH(0,1)
Mean Model      : ARFIMA(1,0,0)
Distribution    : ged


Optimal Parameters
------------------------------------
         Estimate  Std. Error     t value Pr(>|t|)
ar1     -0.205760    0.663596 -3.1007e-01 0.756509
mxreg1   0.157933    0.161394  9.7856e-01 0.327799
mxreg2  -0.128715    0.184295 -6.9842e-01 0.484913
mxreg3   0.083991    0.621942  1.3505e-01 0.892576
mxreg4  -0.015791    0.000017 -9.4518e+02 0.000000
mxreg5  -0.008759    0.000008 -1.1328e+03 0.000000
mxreg6   0.008758    0.000007  1.1916e+03 0.000000
mxreg7  -0.008755    0.000013 -6.6099e+02 0.000000
mxreg8  -0.008751    0.000012 -7.4796e+02 0.000000
mxreg9  -0.007904    0.000006 -1.4010e+03 0.000000
mxreg10  0.007903    0.000005  1.5158e+03 0.000000
mxreg11  0.007892    0.000011  7.2168e+02 0.000000
mxreg12  0.005820    0.000009  6.6992e+02 0.000000
mxreg13 -0.005818    0.000005 -1.1384e+03 0.000000
mxreg14  0.005818    0.000004  1.3030e+03 0.000000
mxreg15  0.005817    0.000009  6.5267e+02 0.000000
mxreg16 -0.005131    0.000014 -3.6675e+02 0.000000
mxreg17 -0.005105    0.000012 -4.3509e+02 0.000000
mxreg18 -0.005085    0.000005 -1.0199e+03 0.000000
mxreg19 -0.000736    0.000001 -6.7977e+02 0.000000
mxreg20 -0.000718    0.000005 -1.4118e+02 0.000000
mxreg21 -0.000676    0.000012 -5.5147e+01 0.000000
mxreg22 -0.002799    0.000005 -5.2193e+02 0.000000
mxreg23  0.002799    0.000003  1.0548e+03 0.000000
mxreg24 -0.002799    0.000003 -1.0384e+03 0.000000
mxreg25 -0.002794    0.000006 -5.0023e+02 0.000000
mxreg26 -0.000046    0.000061 -7.5724e-01 0.448907
mxreg27  0.000027    0.000072  3.8310e-01 0.701644
mxreg28 -0.000012    0.000002 -5.5235e+00 0.000000
mxreg29  0.000008    0.000004  1.9732e+00 0.048477
omega    0.000098    0.008672  1.1337e-02 0.990954
beta1    0.900000    0.000120  7.5153e+03 0.000000
shape    2.000000    0.118171  1.6925e+01 0.000000
```

*Figura 3.25 Stima paramentri AR+eGARCH - valori extrarendimenti BMW*

```
Robust Standard Errors:
         Estimate  Std. Error     t value Pr(>|t|)
ar1     -0.205760    0.223575 -9.2032e-01 0.357407
mxreg1   0.157933    0.078909  2.0014e+00 0.045344
mxreg2  -0.128715    0.014072 -9.1471e+00 0.000000
mxreg3   0.083991    0.186960  4.4925e-01 0.653254
mxreg4  -0.015791    0.000010 -1.5266e+03 0.000000
mxreg5  -0.008759    0.000007 -1.2484e+03 0.000000
mxreg6   0.008758    0.000007  1.2564e+03 0.000000
mxreg7  -0.008755    0.000000 -2.3036e+04 0.000000
mxreg8  -0.008751    0.000004 -2.2480e+03 0.000000
mxreg9  -0.007904    0.000007 -1.1504e+03 0.000000
mxreg10  0.007903    0.000006  1.2382e+03 0.000000
mxreg11  0.007892    0.000003  2.2677e+03 0.000000
mxreg12  0.005820    0.000003  1.9758e+03 0.000000
mxreg13 -0.005818    0.000007 -8.9028e+02 0.000000
mxreg14  0.005818    0.000002  2.7783e+03 0.000000
mxreg15  0.005817    0.000003  2.2267e+03 0.000000
mxreg16 -0.005131    0.000008 -6.5766e+02 0.000000
mxreg17 -0.005105    0.000006 -8.5645e+02 0.000000
mxreg18 -0.005085    0.000003 -1.7631e+03 0.000000
mxreg19 -0.000736    0.000000 -6.5121e+03 0.000000
mxreg20 -0.000718    0.000005 -1.4733e+02 0.000000
mxreg21 -0.000676    0.000013 -5.1854e+01 0.000000
mxreg22 -0.002799    0.000001 -2.4872e+03 0.000000
mxreg23  0.002799    0.000001  2.3350e+03 0.000000
mxreg24 -0.002799    0.000001 -2.4734e+03 0.000000
mxreg25 -0.002794    0.000002 -1.4141e+03 0.000000
mxreg26 -0.000046    0.000007 -6.6359e+00 0.000000
mxreg27  0.000027    0.000003  8.8659e+00 0.000000
mxreg28 -0.000012    0.000001 -1.5954e+01 0.000000
mxreg29  0.000008    0.000002  3.2436e+00 0.001180
omega    0.000098    0.018772  5.2370e-03 0.995821
beta1    0.900000    0.002467  3.6487e+02 0.000000
shape    2.000000    0.006736  2.9690e+02 0.000000

LogLikelihood : -177.1886
```

*Figura 3.26 Stima robusta paramentri AR+eGARCH - valori extrarendimenti BMW*

Vediamo che tutti i parametri sono significativi a parte il parametro dell'intercetta del modello eGARCH che presenta una forte evidenza verso l'accettazione dell'ipotesi nulla. Vediamo che anche il parametro dell'AR e del primo ritardo del rendimento accettano l'ipotesi nulla di non significatività, il parametro dell'AR tuttavia con un livello $\alpha$ inferiore. Questo probabilmente perché entrambi i parametri sono legati alla stessa variabile. Togliendo dal modello il primo ritardo del rendimento il parametro AR torna significativo. Tuttavia questo peggiore le stime finali, per questo consapevolmente è stata tenuta la specificazione indicata.

Si nota che il parametro stimato shape per la ged ha valore 2, il che significa che la distribuzione collassa ad una normale standard.

Vediamo che il valore della logverosimiglianza è correttamente negativo.

```
Information Criteria
------------------------------------

Akaike       1.7299
Bayes        2.2043
Shibata      1.6986
Hannan-Quinn 1.9210

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic   p-value
Lag[1]                      10.68 1.083e-03
Lag[2*(p+q)+(p+q)-1][2]     11.82 1.001e-11
Lag[4*(p+q)+(p+q)-1][5]     14.63 8.943e-06
d.o.f=1
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic p-value
Lag[1]                      2.115  0.1458
Lag[2*(p+q)+(p+q)-1][2]     2.196  0.2321
Lag[4*(p+q)+(p+q)-1][5]     2.340  0.5406
d.o.f=1

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[2]    0.1592 0.500 2.000  0.6899
ARCH Lag[4]    0.1703 1.397 1.611  0.9671
ARCH Lag[6]    0.3817 2.222 1.500  0.9838
```

*Figura 3.27 Criteri di informazione e test sui residui AR+ eGARCH - valori extrarendimenti BMW*

I criteri di informazione restituiscono tutti valori simili e seppur bassi tutti positivi

Il test sui residui standardizzati rifiuta l'ipotesi nulla di assenza di correlazione fra i residui, tuttavia questa viene accettata quando il test è applicato al quadrato dei residui standardizzati.

Il test ARCH LM inoltre accetta l'ipotesi nulla di assenza di eteroschedasticità autoregressiva condizionata.

```
Nyblom stability test
------------------------------------
Joint Statistic:  no.parameters>20 (not available)
Individual Statistics:
ar1      0.562515
mxreg1   0.394065
mxreg2   0.376100
mxreg3   0.076096
mxreg4   0.551535
mxreg5   0.427465
mxreg6   0.429200
mxreg7   0.409128
mxreg8   0.452519
mxreg9   0.510638
mxreg10  0.511330
mxreg11  0.710359
mxreg12  0.008673
mxreg13  0.607840
mxreg14  0.472025
mxreg15  0.028720
mxreg16  0.238810
mxreg17  0.144153
mxreg18  0.269161
mxreg19  0.267228
mxreg20  0.227949
mxreg21  0.274581
mxreg22  0.474216
mxreg23  0.348240
mxreg24  0.340001
mxreg25  0.439607
mxreg26  0.487910
mxreg27  0.532050
mxreg28  0.292337
mxreg29  0.608017
omega   76.477687
beta1   33.719758
shape   79.177142

Asymptotic Critical Values (10% 5% 1%)
Individual Statistic:    0.35 0.47 0.75
```

*Figura 3.28 Nymblom stability test AR + eGARCH - valori extrarendimenti BMW*

A parte il coefficiente del regressore *sum_sentiment_score_pond* tutti gli altri parametri accettano l'ipotesi nulla di non cambiare valore in maniera significativa nel corso del tempo.

```
Sign Bias Test
------------------------------------
                    t-value   prob sig
Sign Bias           0.5796 0.5627
Negative Sign Bias  0.6410 0.5221
Positive Sign Bias  0.7292 0.4666
Joint Effect        1.0052 0.8000


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20      2085            0
2    30      3231            0
3    40      4315            0
4    50      5417            0


Elapsed time : 0.2383249
```

*Figura 3.29 Sign Bias Test AR + eGARCH - valori extrarendimenti BMW*

Infine, tutti i test di asimmetria accettano l'ipotesi nulla di assenza di asimmetria nei residui, mentre il test Adjusted Pearson Goodness of fit rifiuta l'ipotesi nulla.In generale quindi, il modello sembra essere correttamente specificato.

Passiamo adesso ai risultati ottenuti.

| | RMSE | RRMSE | R^2 |
|---|---|---|---|
| AR + eGARCH | 0.0235 | 0.93 | 0.12 |
| Benchmark | 0.0252 | 1.00 | -0.01 |

*Tabella 3-18 Risultati AR + eGARCH & Benchmark - valori extrarendiemnti BMW*

Come ci aspettavamo, il modello non riesce a ottenere delle performance notevoli, ma tuttavia a differenza del caso dei rendimenti, riesce a superare le metriche presentate dal modello di benchmark.

Di seguito riportiamo il grafico con il confronto fra la vera serie degli extrarendimenti e le previsioni dei due modelli nel test set.

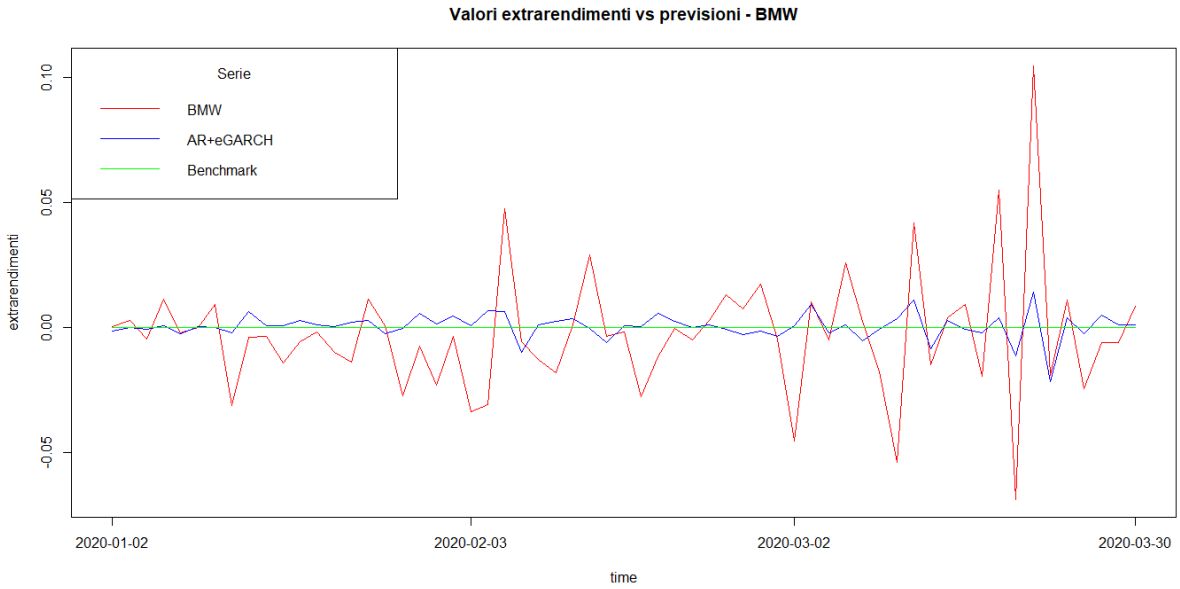


*Figura 3.30 Serie valori e previsioni extarendimenti BMW*

Vediamo che il modello econometrico è in grado solo di intuire l'andamento della serie storica e performa un po' meglio del modello di benchmark. I picchi principali infatti non vengono colti dal modello, che rimane schiacciato intorno alla media della serie.

In conclusione, le informazioni ricavate dai tweet sono sicuramente significative, questo verificato anche dai test sui parametri. Tuttavia non riescono a cogliere davvero i movimenti della serie sui mercati. Vi è comunque un miglioramento rispetto alla previsione dei rendimenti puri, proprio come osservato in Tesla. Questo ad avvalorare la tesi che le informazioni sono collegate con i movimenti degli asset diversi dal trend generale del mercato più che con i rendimenti in se. Su BMW tuttavia non si è visto un aumento della correlazione fra i rendimenti ed extrarendimenti come osservato su Tesla.

### *3.3.5. Previsione segno extrarendimenti BMW*

Infine, trattiamo l'ultimo problema di classificazione riguardante la previsione del segno degli extrarendimenti. Al solito il segno viene rappresentato da una variabile dicotomica che assume valore 0 quando il valore è negativo e 1 quando è positivo. Per visualizzare l'andamento della variabile viene riportato di seguito il grafico dei suoi movimenti nel corso del tempo.

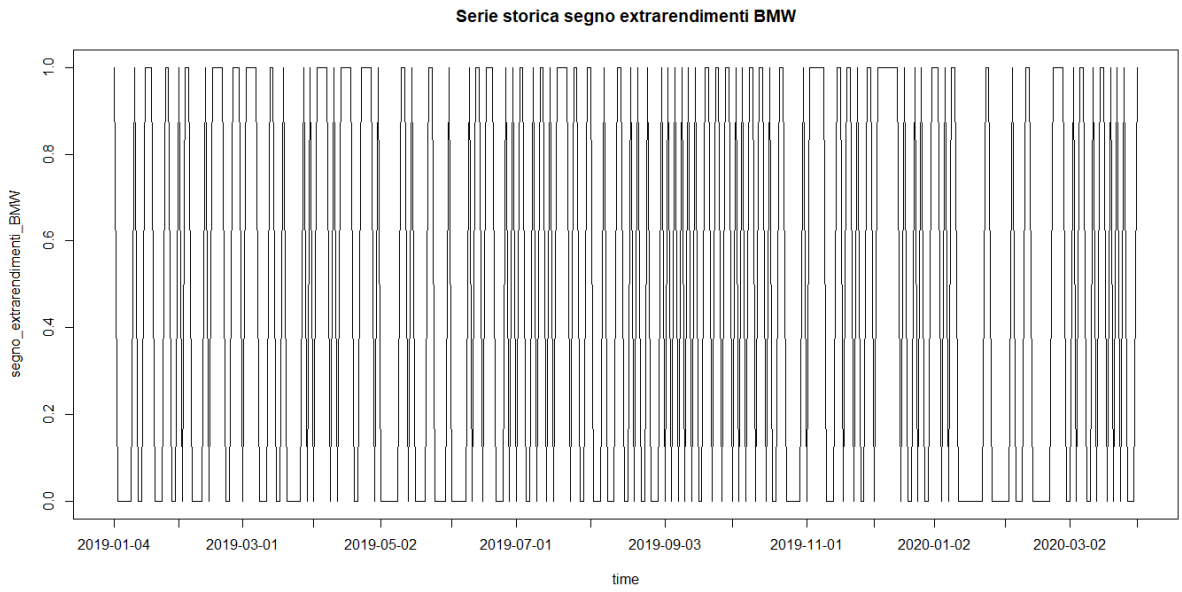


*Figura 3.31 Serie storica segno extrarendimenti BMW*

Anche qui si nota come vi siano parecchi periodi in cui il rendimento o e sempre positivo oppure è sempre negativo. In questo caso gli extrarendimenti negativi sono in numero lievemente maggiore degli extraredimenti positivi (160 vs 144).

Di seguito vengono riportate le variabili maggiormente correlate con la variabile risposta, dove il livello di correlazione è al solito calcolato con l'indice di kendall. Dato il basso livello di correlazione, vengono considerate tutte le variabili con un livello di correlazione maggiore o uguale a |0.08|.

| | Correlazione |
|---|---|
| n_negative | 0.09 |
| mean_neutral_pond | -0.09 |

*Tabella 3-19 Variabili esplicative più correlate con segno extrarendimenti BMW*

Il livello di correlazione generale è diminuito rispetto al caso dei segni dei rendimenti. Solo due variabili vengono selezionate, che comunque presentano un livello inferiore al 10%.

Per la previsione del segno non è stata effettuata nessuna variable selection, ed è stato utilizzata una SVM con kernel polinomiale con i seguenti paramentri: $\gamma = \frac{1}{79}, coef0 = 0, degree = 3$, $C = 1$, individuati tramite una 10 folds cross validation nel training set. La matrice dei regressori è stata standardizzata prima di essere utilizzata all'interno del modello. Le metriche ottenute sono le seguenti.

| | Accuracy | AUC |
|---|---|---|
| SVM | 0.656 | 0.57 |
| SVM vs benchmark | 0.156 | 0.07 |

*Tabella 3-20 Risultati SVM vs Benchmark - segno extrarendiemnti BMW*

Anche per BMW le metriche migliorano quando si cercano di classificare gli extrarendimenti. Vengono classificate correttamente più del 65% delle classi, e il comportamento del modello nel classificare le classi negative rispetto a quelle positive è migliore del modello completamente random di circa 7 punti percentuali.

Di seguito viene riportata la matrice di confusione

| | pred_value_0 | pred_value_1 | Total |
|---|---|---|---|
| True_value_0 | 36 | 1 | 37 |
| True_value_1 | 20 | 4 | 24 |
| Total | 56 | 5 | 61 |

*Tabella 3-21 Matrice di confusione SVM - segno extrarendiemnti BMW*

Dalla matrice di confusione vediamo che il classificatore prevede quasi sempre il segno negativo e poche volte il segno positivo. Anche se questo può essere positivo in un'ottica di previsione delle perdite a tutti i costi, non è molto sensato un classificatore che prevede solo una classe.

Infine viene riportata la curva ROC associata al classificatore. L'area sottesa alla curva è pari al valore AUC riportato nella tabella delle metriche.

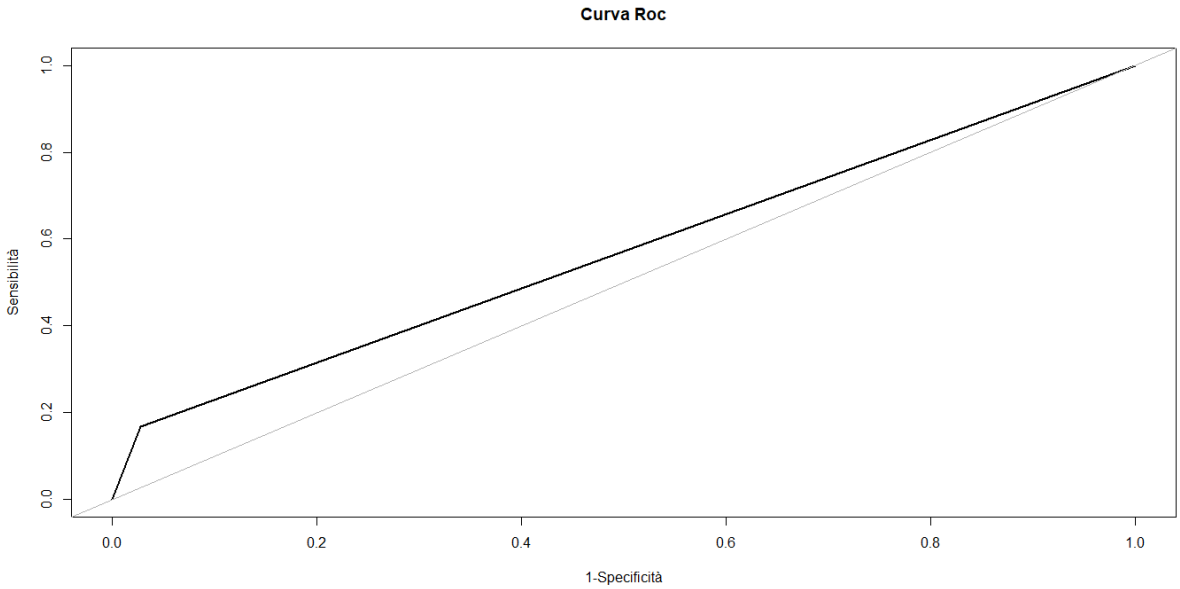


*Figura 3.32 Curva Roc SVM - segno extrarendimenti BMW*

In conclusione, anche in questo caso abbiamo visto un lieve miglioramento delle performance rispetto alla previsione del segno dei rendimenti. Tuttavia anche qui non vi è stato un aumento dell'indice di correlazione che fra le variabili esplicative e la risposta è molto basso, quasi assente.

Il codice relativo all'analisi sia del segno che dei valori degli extra rendimenti BMW si può trovare nella repository Github della tesi[123]

[123] https://github.com/Moreno-Sanna/Mythesis, file " 11_Creation aggregate dataset by financial date & BMW extra yield analysis.R".

# Conclusioni

Osservando i risultati ottenuti dalle analisi precedenti possiamo giungere alle seguenti considerazioni:

1. In linea con quanto ci aspettavamo, le informazioni relative ad un argomento (in questo caso un'azienda) più in voga come può essere Tesla presentano più relazione con l'andamento sul mercato dell'asset a cui si riferiscono, rispetto ad aziende meno "chiacchierate" come può essere BMW. Questo può essere dovuto al fatto che:
    - Le aziende più in voga presentano più informazioni sui canali social, in quanto le persone condividono più spesso opinioni su di esse. Questo è facilmente riscontrabile anche dal fatto che i tweet totali riguardanti Tesla sono circa il doppio rispetto a quelli condivisi connessi a BMW.
    - Le aziende più chiacchierate sono più soggette a ciò che condividono le persone, per il semplice fatto che se ne parla di più e gli viene concessa più attenzione.
    - Molto probabilmente il rumore presente nei tweet Tesla è maggiore rispetto al rumore presente nei tweet BMW, dove il rumore viene inteso come tweet non collegati direttamente con l'azienda di riferimento cioè retweet pubblicitari, utenti che hanno le parole Tesla o BMW nel nome, casualità ecc… Indirettamente tuttavia, anche il rumore può essere connesso con l'andamento dei titoli presi in esame, e questo può favorire la correlazione fra il mondo social e i mercati.

    Oltre alla correlazione maggiore, i rendimenti ed extrarendimenti Tesla si prevedono meglio rispetto all'asset BMW, questo per via del maggior legame fra informazioni e asset. Su BMW inoltre per quanto riguarda i valori dei rendimenti non si è riusciti a trovare un modello che performasse in maniera significativamente diversa del modello di benchmark.
2. Gli extrarendimenti presentano una correlazione con le informazioni leggermente più forte rispetto ai rendimenti puri, almeno per quanto riguarda i valori della serie rispetto che dei segni. Ciò che si nota, è che la serie degli extrarendimenti riesce a essere predetta leggermente meglio rispetto a quella dei rendimenti, indipendentemente dal tipo di asset e di problema trattato. Questo sembra indicare che le informazioni ricavate dai tweet sono connesse in misura maggiore con i movimenti degli asset diversi da quelli del mercato. Del resto è logico pensare che il sentimento delle persone collegato ad un

determinato asset influenzi solo quel titolo, facendolo muovere in maniera diversa rispetto al mercato.

3. Si ottengono risultati migliori nella previsione dei segni piuttosto che dei valori della serie, dove in un caso (extrarendimenti Tesla) si è riusciti a migliorare le performance del classificatore random del 50% in termini di accuracy e poco meno di AUC. Questo nonostante la correlazione dei segni con le informazioni sia decisamente minore, almeno per quanto riguarda l'asset Tesla. Su BMW non si nota questa diminuzione in termini di correlazione, probabilmente perché già molto bassa anche calcolandola rispetto ai valori dei rendimenti o extrarendimenti.
4. Utilizzando le informazioni ricavate dai tweet non si è riusciti in ogni caso ad ottenere dei modelli davvero interessanti per la previsione dell'andamento degli asset collegati, siano essi espressi tramite valori, segni, rendimenti o extrarendimenti, nemmeno su Tesla dove alcune variabili presentano una correlazione intorno al 50%. Infatti anche il miglior modello per la previsione dei valori, quello riguardante gli extrarendimenti, presenta un $R^2$ comunque inferiore al 20%. Questo fatta eccezione per i segni dei rendimenti ed extrarendimenti Tesla, dove come già indicato si ottengono delle performance se non buone quanto meno decenti.
5. Prendendo in considerazione l'asset Tesla in cui si nota una certa correlazione fra le informazioni e i mercati e si costruiscono i migliori modelli di previsione, vediamo che le variabili con correlazione più alta, quelle che vengono maggiormente selezionate e quelle più significative nei problemi di regressione e classificazione sono tutte non ritardate, dello stesso periodo della variabile risposta. Questo conduce a due differenti tipi di considerazioni:
    - Queste variabili non possono essere utilizzate per costruire una strategia finanziaria davvero perseguibile, in quanto sarebbero disponibili nello stesso momento di realizzazione dell'asset ed utilizzabili quando ormai questo si è già manifestato.
    - Avendo a disposizione i redimenti giornalieri dell'asset e non con un dettaglio maggiore, resta il dubbio di capire chi ha causato cosa. Infatti nell'aggregazione per giorno vengono sintetizzati tutti i tweet condivisi ad una determinata ora della giornata, e confrontati con il rendimento ottenuto dall'asset in quel giorno, senza indagare l'ora in cui sono stati condivisi i messaggi. Questi potrebbero essere stati twittati alla sera con i mercati già chiusi, magari commentando il rendimento ottenuto nella giornata, oppure

condivisi pochi minuti dopo un brusco aumento o diminuzione del prezzo dell'asset. In questi casi vi è sicuramente della correlazione con il mercato, ma causata da questo e non viceversa.

In sintesi, si è visto che almeno per quanto riguarda Tesla vi è una certa connessione con le informazioni condivise e l'andamento dell'asset, che è maggiore negli extra-rendimenti. Infatti si riescono a prevedere meglio questi rispetto che ai rendimenti, soprattutto per quanto riguarda i segni della serie. Tuttavia, questo legame non permettere di costruire una reale ed interessante strategia finanziaria, perché non è molto forte ed è basato prevalentemente su variabili non ritardate.

# Bibliografia


1) Dogu Tan Araci: *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models*, University of Amsterdam, 06 – 2019, arXiv:1908.10063v1

2) Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 05 – 2019, arXiv:1810.04805v2

3) Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin : *Attention Is All You Need,* 12-2017, arXiv:1706.03762v5

4) Diederik P. Kingma, Jimmy Lei Ba: *Adam: A Method For Stochastic Optimization,* 01-2017, arXiv:1412.6980v9

5) Yukun Zhu, Ryan Kiros, Richard Zeme, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, Sanja Fidler: *Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books* 07-2015, arXiv:1506.06724v1

6) Mike Schuster, Kaisuke Nakajima, *Japanese and Corean Voice Search,* in ICASSP, 2012

7) Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi: *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*, 10-2016, arXiv:1609.08144v2

8) Denny Britz, Anna Goldie , Minh-Thang Luong, Quoc Le: *Massive Exploration of Neural Machine Translation Architectures*, 03 – 2017, arXiv:1703.03906v2

9) Rico Sennrich, Barry Haddow, Alexandra Birc: *Neural Machine Translation of Rare Words with Subword Units*, in *Proceedings of the 54th Annual Meeting of the*

*Association for Computational Linguistics*, pages 1715–1725, Berlin, Germany, August 7-12, 2016

10) Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov: *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, in *Journal of Machine Learning Research 15* (2014), pages 1929-1958, 06-2014

11) Chuan-Ju Wang, Ming-Feng Tsai, Tse Liu, Chin-Ting Chang: *Financial Sentiment Analysis for Risk Prediction*, in *International Joint Conference on Natural Language Processing*, pages 802–808,Nagoya, Japan, 14-18 October 2013.

12) Dan Hendrycks, Kevin Gimpel, *Gaussian Error Linear Units (Gelus),* 11-2018, arXiv:1606.08415v3

13) Trevor Hastie Junyang Qian Kenneth Tay, *An Introduction to glmnet*, 11-2021

14) Bernard Lutz, Nicolas Prollochs, Dirk Neumann, *Sentence-Level Sentiment Analysis of Financial News Using Distributed Text Representations and Multi-Instance Learning*, 12-2018, arXiv:1901.00400v1

15) Felix Ming Fai Wong, Zhenming Liu, Mung Chiang, *Stock Market Prediction from WSJ:Text Mining via Sparse Matrix Factorization*, in IEEE International Conference on Data Mining, 2014.

16) Mukul Jaggi, Priyanka Mandal, Shreya Narang, Usman Naseem and Matloob Khushi: *Text Mining of Stocktwits Data for Predicting Stock Prices*, 02 2021, in *Appl. Syst. Innov.* 2021, 4, 13. https://doi.org/10.3390/asi4010013

17) Robert F. Engle, Victor K. Ng: *Measuring and Testing the Impact of News on Volatility*, in *The Journal Of Finance* V Vol. Xlviii, No. 5 V December 1993

18) Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, *Rethinking the Inception Architecture for Computer Vision,* 12 - 2015, arXiv:1512.00567v3

19) Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio: *Neural Machine Translation By Jointly Learning To Align And Translate,* Published as a conference paper at ICLR 2015, 05 - 2016 , arXiv:1409.0473v7

20) Yunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, Yoshua Bengio: *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation,* 09-2014, arXiv:1406.1078v3

21) Wilson L Taylor. 1953. *Cloze procedure: A new tool for measuring readability.* in Journalism Bulletin,30(4), pages: 415–433.

22) Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical, Learning, Second Edition,* ISBN: 978-0-387-84857-0

23) Matteo Maria Pelagatti, *Time Series Modelling with Unobserved Components*, CRC Press, 06- 2015, ISBN 13: 978-1-4822-2501-3

24) Matteo Maria Pelagatti, *Statistica dei Mercati Monetari e Finanziari,* 12-2020, dispensa del corso *Computational Finance and Financial Econometrics.* UNIMIB, A.A. 2020/2021

25) Lisa Crosato, *Serie storiche economiche,* dispensa del corso *Introduzione alle Serie Storiche M.* UNIMIB

26) Antonio Candelieri, *Slide di lezione* del corso *Machine Learning M*, UNIMIB, A.A. 2020/2021

27) Aldo Solari, *Slide di lezione* del corso *Data Mining M*, UNIMIB, A.A. 2020/2021

28) Nicola Lunardon, *Slide di lezione* del corso *Statistica Multivariata*, UNIMIB, A.A. 2019/2020

29) Matteo Manera, *Slide di lezione* del corso *Microeconometria*, UNIMIB, A.A. 2019/2020

30) Bancamaria Zavanella, *Slide di lezione* del corso *Introduzione alle Serie Storiche M*, UNIMIB, A.A. 2019/2020

# Sitografia


1) Jay Alammar, *The Illustrated Transformer* , https://jalammar.github.io/illustrated-transformer/

2) Jay Alammar, *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*, https://jalammar.github.io/illustrated-bert/

3) Samuel Kierszbaum, *Masking in Transformers' self-attention mechanism*, 01-2020, https://medium.com/analytics-vidhya/masking-in-transformers-self-attention-mechanism-bad3c9ec235c

4) Michael Phi, *Illustrated Guide to Transformers- Step by Step Explanation*, 05 – 2020, https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

5) Rani Horev, *BERT Explained: State of the art language model for NLP*, https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

6) Jaron Collis ,*Glossary of Deep Learning: Word Embedding*, 04 – 2017, https://medium.com/deeper-learning/glossary-of-deep-learning-word-embedding-f90c3cec34ca

7) Logicalerrors, *Garch – Modeling Conditional Variance & Useful Diagnostic Tests*, 08-2017, https://logicalerrors.wordpress.com/2017/08/14/garch-modeling-conditional-variance-useful-diagnostic-tests/

8) Łukasz Kaiser, *Attention is all you need; Attentional Neural Network Models, Masterclass*, https://www.youtube.com/watch?v=rBCqOTEfxvg&t=946s

9) Jacob Devlin, *BERT multilingual*, https://github.com/google-research/bert/blob/master/multilingual.md

10) Twitter Developers, *Building queries for Search Tweets*, https://developer.twitter.com/en/docs/twitter-api/tweets/search/integrate/build-a-query

11) Twitter help center, *About different types of Tweets*, https://help.twitter.com/en/using-twitter/types-of-tweets

12) *Exact definition of Deviance measure in glmnet package, with crossvalidation?*, https://stats.stackexchange.com/questions/117732/exact-definition-of-deviance-measure-in-glmnet-package-with-crossvalidation

13) Wikipedia, *Crollo del mercato azionario del 2020*, https://it.wikipedia.org/wiki/Crollo_del_mercato_azionario_del_2020

14) rdrr.io, *deviance.glmnet: Extract the deviance from a glmnet object* https://rdrr.io/cran/glmnet/man/deviance.glmnet.html

15) Yahoo finance, *Tesla, Inc. (TSLA)*, https://finance.yahoo.com/quote/TSLA

16) Yhaoo finance, *Bayerische Motoren Werke Aktiengesellschaft (BMW.DE)*, https://it.finance.yahoo.com/quote/bmw.de

17) First Trust, *First Trust NASDAQ Global Auto Index Fund (CARZ)* https://www.ftportfolios.com/retail/etf/ETFholdings.aspx?Ticker=CARZ

18) Yhaoo finance, *First Trust NASDAQ Global Auto Index Fund (CARZ)* https://finance.yahoo.com/quote/CARZ

19) Dogu Araci, *ProsusAI/finBERT*, https://github.com/ProsusAI/finBERT

20) Nicolas Patry, *huggingface/transformers*, https://github.com/huggingface/transformers.

21) Andy Piper, *Twitter-API-v2-sample-code*, https://github.com/twitterdev/Twitter-API-v2-sample-code/blob/main/Full-Archive-Search/full-archive-search.r.

# Allegati

Link alla repository Github contenente i codici collegati alla tesi: https://github.com/Moreno-Sanna/Mythesis