UNIVERSITÁ DEGLI STUDI DI MILANO-BICOCCA

Scuola di Economia e Statistica

Corso di laurea in

STATISTICA E GESTIONE DELLE INFORMAZIONI

Analisi delle recensioni: un approccio basato sull'elaborazione del linguaggio naturale

Relatore: Prof. Roberto Boselli

Tesi di laurea di:
Davide Vettore
Matr. 868855

Anno Accademico 2022/23

Indice

Introduzione

L'elaborazione del linguaggio naturale, comunemente nota come *Natural Language Processing* (NLP), è un settore interdisciplinare che coinvolge la linguistica, l'informatica e l'intelligenza artificiale. L'obiettivo è permettere ai computer di comprendere il contenuto dei documenti prestando attenzione al contesto e alle ambiguità del linguaggio umano.

Già nella pubblicazione di Alan Turing intitolata "Computing Machinery and Intelligence" [19] del 1950, vengono affrontati temi che possono essere considerati precursori del NLP. Turing introduce il concetto del "gioco dell'imitazione", anche noto come "Test di Turing", il quale coinvolge un giudice umano che interagisce tramite un terminale con un essere umano e una macchina. L'obiettivo del test è che la macchina riesca a convincere il giudice di essere un essere umano attraverso la comunicazione in linguaggio naturale. Questa è una delle prime menzioni di un test basato sull'elaborazione del linguaggio naturale come criterio per valutare l'intelligenza di una macchina. Sebbene Turing non utilizzi esplicitamente il termine "Natural Language Processing" nel suo articolo, il concetto di comunicazione e interazione con le macchine attraverso il linguaggio naturale è centrale nel discorso.

Fino agli anni '80 i tentativi di sviluppo dei sistemi NLP vertevano intorno alla traduzione automatica dei testi [1] o alla realizzazione di chatbot [2] utilizzando complesse regole linguistiche determinate durante la fase di progettazione. Il paradigma cambia a favore delle metodologie statistiche a metà degli anni '90: si adotta l'inferenza statistica per apprendere automaticamente le regole linguistiche analizzando grandi corpora di dati reali, inizialmente utilizzando modelli semplici come gli alberi decisionali. Nell'ultimo ventennio l'interesse si è spostato sulle reti neurali: dai più semplici Multi-Layer Perceptron (MLP) fino ai Deep Neural Network. L'evoluzione è stata guidata dalla necessità di affrontare l'ambiguità linguistica e di gestire i molteplici significati delle parole in base al contesto.

[1] Primo esempio fu la collaborazione tra l'università di Georgetown e IBM, l'obiettivo era tradurre automaticamente oltre sessanta frasi russe. Venne presentato come un successo, solo dopo 10 anni di ricerca l'ALPAC si dimostrò scettica riguardo agli sviluppi della ricerca, portando il governo americano a tagliare i fondi.

[2] Software pensati per imitare conversazioni umane tramite interazioni testuali, i più famosi furono ELIZA e PARRY.

Recentemente, l'avanguardia della ricerca nell'elaborazione del linguaggio naturale, consiste nell'utilizzo di modelli linguistici preaddestrati come GPT-3 (*Generative Pre-trained Transformer*) e BERT (*Bi-directional Encoder Representations from Transformers*). Questi modelli possono essere ulteriormente addestrati (fine-tuned) su dati specifici per poter eseguire una vasta gamma di compiti NLP.

Precedenti studi condotti sulle recensioni online

L'addestramento di modelli capaci di risolvere task di NLP richiede un enorme quantità di dati testuali. Nel 2011 Andrew L. Maas [11] utilizza per la prima volta dati estratti da *Internet Movie Database* (IMDb): sito internet contenente informazioni su oltre 500.000 film, nonché centinaia di migliaia di recensioni prodotte dagli utenti. Maas sviluppa inizialmente un modello probabilistico capace di catturare le similarità semantiche tra le parole, esso si basa sugli stessi fondamenti probabilistici della *Latent Dirichlet Allocation* (LDA) [2]. Introduce successivamente una fase supervisionata: utilizzando i rating associati alle recensioni, normalizzati a "positivo" o "negativo", addestra un modello *Support Vector Machine* lineare[3] per riconoscere il sentimento. Vengono utilizzate 25.000 recensioni, si ottiene un dizionario contenente i 5000 termini più frequenti, stopword e caratteri speciali ("!", ":-)") non vengono rimossi poiché indicativi di sentimento. Rispetto al primo task, i risultati vengono confrontati con quanto ottenuto utilizzando l'ampiamente diffusa *Latent Semantic Analysis* (LSA) [7]. Viene dimostrato come entrambe le versioni del modello di Maas, *Semantic only* e *Sentiment+Semantic*, performino meglio della LSA nel riconoscimento delle parole semanticamente simili. Nel secondo task si vuole prevedere, dato il testo di una recensione, qualora esprima sentimento positivo o negativo. Anche in questo caso, il paper dimostra un notevole aumento delle prestazioni rispetto ai modelli precedentemente introdotti. Sul dataset IMDb contenente 25.000 reviews con rating positivo-negativo, il miglior modello presentato ottiene un'accuracy cross-validata pari all'88.33%, LDA e LSA raggiungono rispettivamente il 67.42% e l'83.96%.

Nel paper "Exploring thematic composition of online reviews: A topic modeling approach" [20] viene proposto un approccio che, mediante analisi quantitative e qualitative, ha lo scopo di studiare diversi aspetti riguardanti una raccolta di recensioni online. I dati utilizzati provengono da Yelp[4], in totale vengono utilizzate 13.456 recensioni estratte da 200 ristoranti americani.

[3] La SVM viene addestrata tramite LIBLINEAR [8]

[4] Applicazione contente recensioni di crowdsourcing riguardanti ristoranti, negozi, etc.

Tralasciando una prima fase in cui vengono determinate quali sono le caratteristiche di una recensione in grado di influenzare i futuri clienti, si osservano i risultati ottenuti nella topic modelling. Il modello LDA viene utilizzato per estrarre tre topic, un numero maggiore o minore avrebbe prodotto risultati di più difficile interpretazione. Osservando le distribuzioni di parole rappresentative degli argomenti latenti, si deduce come le recensioni riguardino: informazioni generiche riguardanti il ristorante, informazioni riguardanti il servizio e informazioni specifiche riguardanti le portate. Partendo dai topic ottenuti, viene infine condotta un'analisi esplorativa con lo scopo di identificare l'influenza di caratteristiche quali prezzo medio, lunghezza della recensione, rating e altre metriche specifiche.

Similarmente agli studi precedentemente condotti, la prima fase del progetto consiste nel collezionare i dati: nel primo capitolo verranno presentate le metodologie specifiche utilizzate per l'estrazione e l'organizzazione delle recensioni di interesse. I capitoli successivi riporteranno le analisi condotte sui dati ottenuti: ad una prima fase di analisi del sentimento seguirà la ricerca degli argomenti latenti. Le metodologie di analisi del sentimento utilizzate spazieranno dai più semplici modelli basati su regole lessicali fino a modelli di apprendimento automatico più complessi e competitivi, come i MLP e le SVM. Le prestazioni ottenute mediante i diversi approcci verranno confrontate utilizzando metriche affermate quali accuracy, precision, recall e f1 score. La ricerca degli argomenti latenti verrà condotta utilizzando due approcci differenti: in un primo momento verrà utilizzato un algoritmo di segmentazione generico (K-means), verrà poi implementata la LDA, modello probabilistico allo stato dell'arte rispetto alla ricerca degli argomenti latenti. In entrambi gli approcci verrà estratto un numero contenuto di argomenti, essi verranno interpretati ampiamente osservando le distribuzioni di parole e alcune recensioni particolarmente rappresentative.

Capitolo 1

Estrazione dei dati

Lo scopo dell'analisi è applicare diverse metodologie di elaborazione del linguaggio naturale a documenti testuali ottenuti in un contesto reale, ovvero le recensioni di prodotti presenti sul Marketplace Amazon.com. Le recensioni verranno circoscritte ai prodotti Apple con lo scopo di mantenere, in particolare durante la topic modelling, un'ottica di analisi aziendale. La prima fase consiste nel collezionare i dati: in rete sono disponibili molteplici dataset contenenti recensioni di Amazon ma, essendo l'analisi limitata ad alcuni prodotti, la soluzione migliore risulta essere l'estrazione dei dati di interesse direttamente da Amazon.com.

L'estrazione di dati dai siti internet è un argomento delicato, soprattutto quando vengono coinvolte piattaforme leader di mercato, come Amazon. Per tutelare l'enorme quantità di dati, talvolta sensibili, i siti internet mettono a disposizioni API [1] che permettono di accedere ed estrarre una gamma limitata di informazioni, Amazon fornisce tuttavia questi strumenti solo a sviluppatori autorizzati. Una possibile alternativa consiste nell'utilizzo di tecniche di web-scraping: mediante software e strumenti appositamente realizzati, è possibile raccogliere dati da più pagine web per poi organizzarli in un formato strutturato. Questa tecnica richiede accortezze specifiche: eseguendo scraping su dati sensibili o con scopi di lucro si rischia infatti di violare le politiche sulla protezione dei dati di Amazon. Ci limiteremo quindi ad estrarre i testi contenuti nelle recensioni e verrà tralasciata qualsiasi informazione personale riguardante l'utente. Oltre alle API di Amazon, esistono piattaforme di terzi online che offrono servizi di scraping a pagamento. Si decide tuttavia di utilizzare il linguaggio Python dato che, grazie alle numerose librerie specializzate, permette di eseguire sia il web-scraping sia tutte le analisi successive. La libreria *Requests* [15] consente di effettuare richieste HTTP [2] e di riceverne il contenuto come risposta: questo procedimento è semplificato se si abbina all'utilizzo di *Splash*

[1] *Application Programming Interface*: si tratta di procedure che semplificano la comunicazione tra più applicazioni, nel nostro contesto permettono ad esempio di rappresentare siti web in una forma astratta (dati strutturati)

[2] *HyperText Transfer Protocol:* protocollo applicativo utilizzato per la trasmissione di informazioni web

[16], un servizio di rappresentazione di pagine web basato su *Docker* [9]. Questi strumenti supportano il caricamento di pagine web in parallelo e, disattivando le immagini e filtrando alcuni contenuti, permettono in generale una rappresentazione delle pagine web più rapida. Il contenuto HTML [3] viene poi analizzato mediante l'utilizzo della libreria *BeautifulSoup* [17], quest'ultima permette di rappresentare le pagine web tramite una struttura ad albero, rendendo i dati accessibili mediante ordinarie iterazioni Python. Il risultato ottenuto in questo passaggio viene comunemente chiamato *Soup* (zuppa).

Come accennato in precedenza, si tratta di un'operazione delicata: nella fase di scrittura del codice per eseguire il web-scraping è fondamentale prendere delle precauzioni per evitare che gli algoritmi di Amazon rilevino la natura automatizzata delle ricerche e blocchino di conseguenza l'indirizzo IP [4] di provenienza. Occorre infatti utilizzare alcuni accorgimenti, come ad esempio l'impostazione di un tempo di attesa casuale tra le richieste e la modifica dello User-Agent [5] per riuscire ad estrarre grosse quantità di dati in modo consistente.

1.1 Codice

L'unico input richiesto dal programma realizzato è il link di una ricerca Amazon: la parola chiave di ricerca è "Apple" ed i contenuti sono filtrati per Brand: *Apple*, e Seller : *Amazon.com* e *Adorama*. La ricerca è stata ristretta a due soli venditori per semplicità computazionale, essendo il web-scraping un'operazione lenta. Questo primo passaggio è stato diviso in due blocchi principali.

1.1.1 Estrazione dei link

Tramite una prima serie di definizioni si ottiene una lista contenente il link di ciascun prodotto di interesse presente nella ricerca. Nonostante sia il passaggio più semplice, richiede diverse accortezze per essere consistente per qualsiasi ricerca sulla piattaforma. Anche solo capire quante pagine di risultati emergano dalla ricerca richiede una definizione specifica che tenga in considerazione molteplici casi HTML (fig. 1.1).

[3] *HyperText Markup Language:* linguaggio utilizzato per la formattazione e l'impaginazione delle pagine web

[4] *Internet Protocol address:* numero identificativo univoco della rete e del dispositivo da cui parte la richiesta

[5] Si tratta di un parametro del protocollo HTTP il quale fornisce al sito web informazioni sul dispositivo che esegue la richiesta

Successivamente vengono generati i link specifici di ciascuna pagina e, a ciascuno dei link ottenuti, viene applicata una definizione che permette di estrarre i link di ciascuno dei singoli prodotti.

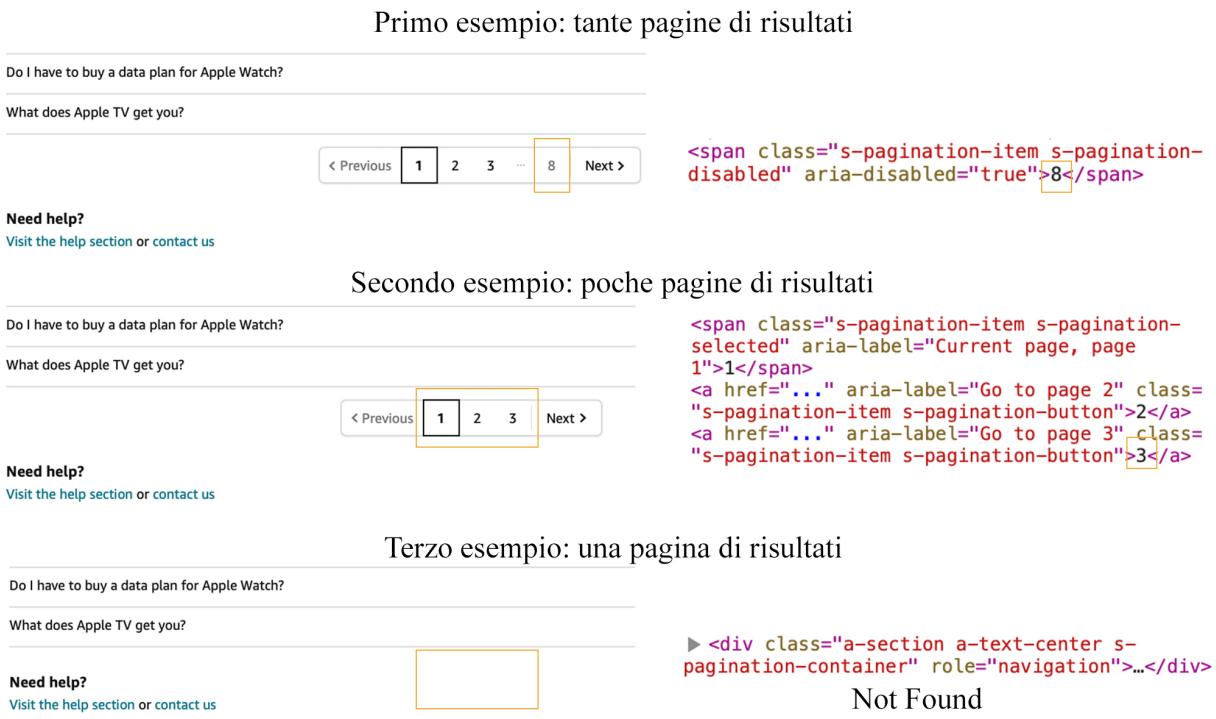

Figura 1.1: Variazione dell'impaginazione web e del rispettivo codice HTML al variare del numero di pagine prodotte dalla ricerca.

1.1.2 Estrazione delle recensioni

Dopo aver ottenuto i link dei singoli prodotti, si può procedere ad estrarre le recensioni di ciascun prodotto. Le pagine contenenti le recensioni sono strutturate diversamente rispetto alle pagine ottenute tramite le ricerche, bisogna quindi seguire un approccio diverso. Una prima definizione, dato il link di una pagina di recensioni, identifica ogni occorrenza del data-hook <*review*> all'interno del Soup HTML. Ciascuno di questi oggetti rappresenta una recensione e segue un formato standard (fig. 1.2), si procede quindi estraendo il titolo, la valutazione ed il testo. Si tratta ora di applicare la definizione precedente a ciascuna delle pagine di recensioni (Amazon ne conserva al massimo 500) appartenenti a ciascuno dei prodotti.

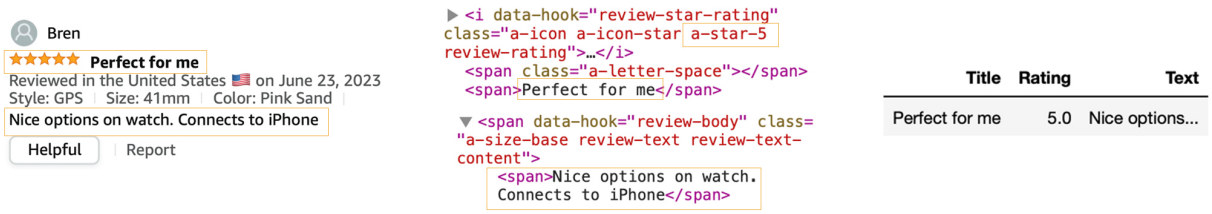

Figura 1.2: Formato delle recensioni presenti sul marketplace Amazon.com.

Iterare rispetto alla totalità delle pagine di recensioni è il passaggio più complesso: bisogna considerare sia i molteplici fattori che potrebbero indicare la fine delle recensioni di interesse, e la conseguente necessità di passare al prodotto successivo, che alcuni limiti di connessione legati al caricamento delle delle pagine web. Partendo dal link di un prodotto, la definizione sarà strutturata come segue:

1. Iterando per ciascuna delle 500 (massimo) pagine di recensioni, essa cercherà innanzitutto di caricare la pagina web, proseguendo fino all'ottenimento del codice HTML corretto.

2. Verranno poi cercati all'interno del Soup due oggetti importanti, i quali potrebbero indicare la fine delle recensioni di interesse:

   - *<dp-global-reviews-header>*: indica l'inizio delle recensioni straniere, essendo interessati alle recensioni in lingua inglese, qualora l'oggetto venga trovato nella pagina, si passa al prodotto successivo.
   - *<a-disabled a-last>*: indica che il tasto "next page" è disabilitato, qualora l'oggetto venga trovato nella pagina, significa che si tratta dell'ultima pagina di recensioni del prodotto in analisi.

3. Qualora i due oggetti non vengano trovati, applicando la definizione descritta in precedenza, si estrae ciascuna delle recensioni presenti nella pagina. Si potrebbe riscontrare un problema nel caso in cui la pagina di recensioni appaia vuota (ad esempio un prodotto che non ha recensioni: la pagina esiste ma è vuota) o qualora non siano ancora state caricate. Prima di confermare che la pagina sia vuota, e passare quindi al prodotto successivo, la definizione ricarica la pagina per cinque volte.

Le due definizioni principali descritte funzionano in modo consistente e permettono di estrarre tutte le recensioni di interesse. Dopo aver rimosso alcuni prodotti privi di recensioni, si ottiene il dataset finale contenente 63.265 recensioni estratte da 199 prodotti diversi. I dati vengono salvati sia in formato JSON non strutturato (poiché ciascun prodotto ha un certo numero di recensioni), sia in formato EXCEL, in cui ogni riga contiene la recensione ed il link del prodotto associato. Viene inoltre definita una colonna "product" per classificare in macro-categorie i prodotti sulla base di parole chiave presenti nel nome del prodotto.

Link	Product_name	Title	Rating	Text	product
https://www.amazon.com/Apple-Pencil-1st-Genera...	Apple Pencil (1st Generation): Pixel-Perfect P...	Very Apple	3	As someone who is not completely bound by the ...	Pencil
https://www.amazon.com/Apple-MHXH3AM-A-MagSafe...	Apple MagSafe Charger - Wireless Charger with ...	Fragility	1	Don't waste your money, did not last long , fe...	Cables
https://www.amazon.com/Apple-MacBook-13-inch-2...	Apple 2020 MacBook Air Laptop M1 Chip, 13"...	Fast, responsive and beautifully crafted	5	Replaced a previous MacBook Air with the new v...	Mac
https://www.amazon.com/Apple-AirPods-Charging-...	Apple AirPods (2nd Generation) Wireless Earbud...	Great use!	5	This was a great purchase	Pods
https://www.amazon.com/Apple-MU8F2AM-A-Pencil-...	Apple Pencil (2nd Generation): Pixel-Perfect P...	Price	4	Price was good, working fine. Pleased with pur...	Pencil

Figura 1.3: Cinque osservazioni estratte casualmente dal dataset ottenuto

Capitolo 2

Sentiment Analysis

Verranno utilizzate diverse metodologie di analisi del sentimento per classificare correttamente il rating (valore continuo da 1 a 5) delle recensioni. Per confrontare le prestazioni dei diversi approcci, verranno utilizzate le seguenti metriche:

- Accuracy: valore percentuale di previsioni esatte.
- Accuracy "off-by-one": valore percentuale di previsioni esatte e di previsioni sbagliate al massimo di un rating.
- Precision: viene calcolata per ciascuna classe, è definita come il numero di veri positivi diviso la somma dei veri positivi e dei falsi positivi.
- Recall: viene calcolata per ciascuna classe, corrisponde al numero dei veri positivi diviso la somma di veri positivi e falsi negativi.
- F1 score: viene calcolata per ciascuna classe, si tratta della media armonica tra precision e recall.

Si osserverà inoltre la media e la media pesata (rispetto al numero di osservazioni) delle metriche calcolate per le singole classi.

2.1 VADER - Lexicon rule based approach

VADER (*Valence Aware Dictionary and sEntiment Reasoner*) è uno strumento di analisi del sentimento specifico per analizzare file di testo estratti dai social network o dalle recensioni online, esso valuta il sentimento utilizzando delle caratteristiche lessicali e delle regole specifiche, riesce a gestire correttamente emoji e slang.

Per ciascuna frase, esso fornisce tre punteggi di polarità (positiva, negativa, neutrale) e un punteggio normalizzato tra -1 e 1, chiamato compound score. Per determinare i punteggi esso presta particolare attenzione ad alcuni punti chiave:

- Punctuation: caratteri come ! e ? pesano sugli score.
- Capitalization: le parole in uppercase (maiuscolo) enfatizzano il sentimento.
- Degree modifiers: alcuni termini come 'extremely' (estremamente) e 'marginally' (marginalmente) intensificano il sentimento.
- Conjunctions: termini come 'but' implicano un cambiamento della polarità.
- Preceding tri-gram: ciascuna parola viene analizzata insieme alle due precedenti, questo per identificare eventuali inversioni di polarità "this phone is not that great".

Applicando VADER al testo delle recensioni, si ottiene il compound score di ciascuna di esse: una prima classificazione intuitiva può essere eseguita suddividendo l'intervallo continuo -1, 1 tramite quattro soglie equidistanti (-0.6, -0.2, 0.2, 0.6). Questo significa, ad esempio, che ad una frase con compound score minore di -0.6 verrà assegnato 1 come rating previsto, mentre ad una frase con compound score maggiore di 0.6 corrisponderà il rating massimo. Possiamo rappresentare graficamente l'accuratezza mediante la matrice di confusione: le righe rappresentano i rating reali mentre le colonne rappresentano i rating predetti. All'aumentare delle prestazioni classificative ci aspettiamo una diagonale principale, ovvero i valori classificati correttamente, più densa (fig. 2.1).

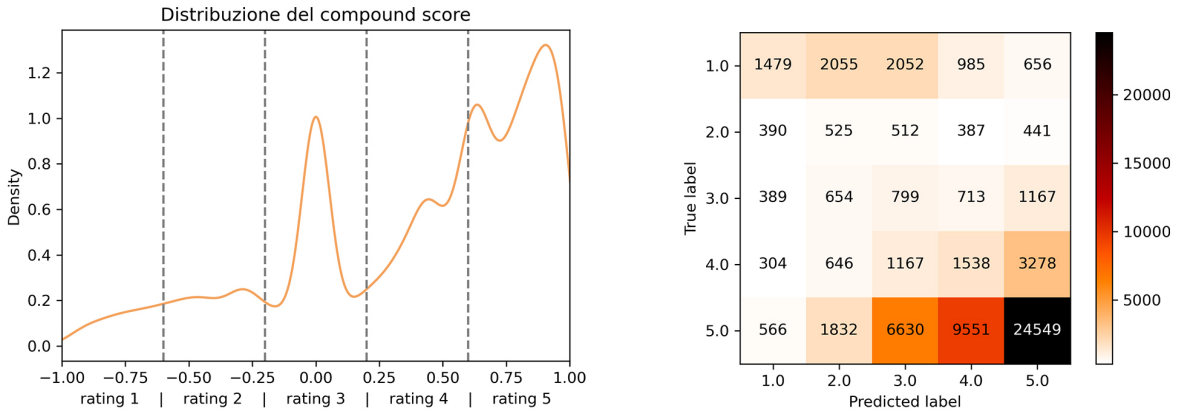

Figura 2.1: Distribuzione e suddivisione del compound score tramite le soglie equidistanti e matrice di confusione

Possiamo ora verificare le prestazioni della classificazione: l'accuracy risulta essere pari al 45.7%, i rating classificati peggio sono quelli centrali (fig. 2.2). Osservando il supporto, ovvero

la numerosità delle osservazioni reali, notiamo l'ampio sbilanciamento delle classi: questo non risulta essere un problema per la classificazione mediante VADER, ma andrà sicuramente preso in considerazione durante la realizzazione dei modelli di machine learning.

```
Accuracy: 0.45665059669643565
Accuracy off-by-one: 0.7462261914170553

              precision    recall  f1-score   support

           1       0.47      0.20      0.29      7227
           2       0.09      0.23      0.13      2255
           3       0.07      0.21      0.11      3722
           4       0.12      0.22      0.15      6933
           5       0.82      0.57      0.67     43128

    accuracy                           0.46     63265
   macro avg       0.31      0.29      0.27     63265
weighted avg       0.63      0.46      0.52     63265
```

Figura 2.2: Performance classificative ottenute con VADER

2.1.1 Scelta delle soglie ottimali

Per migliorare le prestazioni della classificazione mediante VADER, è possibile modificare le soglie. Si mantengono quindi gli stessi compound score ottenuti in precedenza ma, utilizzando una gridsearch [1] contenente le soglie equidistanti modificate di ±0.1 e ±0.2, si verifica il variare dell'accuracy rispetto a ciascuna combinazione di soglie. Si ottiene inizialmente una lista contenente ogni possibile combinazione, procedendo poi con la verifica delle metriche rispetto a ciascuna di esse, emerge un valore massimo di accuracy del 51.1% in corrispondenza della combinazione (-0.4, -0.3, 0.1, 0.5). Con questo semplice accorgimento si ottiene un incremento dell'accuracy superiore al 5% (fig. 2.3).

```
Accuracy: 0.5107879554255907
Accuracy off-by-one: 0.7393819647514424

              precision    recall  f1-score   support

           1       0.44      0.36      0.40      7227
           2       0.09      0.04      0.06      2255
           3       0.07      0.23      0.11      3722
           4       0.13      0.19      0.15      6933
           5       0.81      0.64      0.71     43128

    accuracy                           0.51     63265
   macro avg       0.31      0.29      0.29     63265
weighted avg       0.63      0.51      0.56     63265
```

Figura 2.3: Performance classificative ottenute con VADER utilizzando le soglie ottimali

[1] Metodo utilizzato per l'ottimizzazione degli iperparametri in diversi contesti: si verifica come variano le prestazioni di un modello rispetto a una determinata metrica tentando tutte le combinazioni di un sottinsieme di iperparametri definito

2.2 Machine Learning approach

Nell'ambito dell'analisi dei dati testuali, si possono utilizzare diversi approcci basati su metodi di machine learning per classificare le opinioni degli utenti. L'applicazione di metodi di apprendimento automatico richiede che i dati siano preprocessati: in particolare si rimuovono i caratteri non appartenenti all'alfabeto inglese e le stopwords (parole che non aggiungono significato alla frase come "in", "the", "and"), si normalizzano inoltre i testi in minuscolo. Altra pratica comune, nel contesto dell'*Information Retrival* (IR) [2], è rappresentare i dati testuali tramite un modello *Term Frequency - Inverse Document Frequency* (TF-IDF): questa rappresentazione permette di ridurre la dimensionalità rappresentando i testi come vettori numerici. La funzione di peso del modello TF-IDF è composta da due parti:

- Term Frequency: cresce proporzionalmente al numero di volte in cui il termine appare all'interno del documento.
- Inverse Document Frequency: cresce in modo inversamente proporzionale rispetto al numero di documenti in cui il termine appare.

Si utilizza la funzione TF-IDF implementata della libreria Python scikit-learn [13], questa rappresentazione, alternativamente alle semplici frequenze (rappresentazione Bag-of-Word), riduce l'impatto dei termini molto frequenti nel corpus, i quali sono di fatto meno informativi. In questa fase si riduce inoltre il numero di termini per rendere più veloce la fase di addestramento dei modelli. Il vocabolario completo contiene 27220 token, si conservano in analisi il 25% dei termini reputati più importanti dal modello TF-IDF.

Ultimo passaggio fondamentale, prima di poter iniziare ad allenare i modelli di apprendimento automatico, è bilanciare il dataset. Dato che il 68% delle recensioni corrispondono al rating 5, i modelli di apprendimento automatico saranno propensi a classificare tutte le recensioni allo stesso modo, difficilmente cattureranno le altre classi correttamente. La soluzione a questo problema è bilanciare il dataset eseguendo undersampling, per le classi con troppe occorrenze, e oversampling, per le classi con poche occorrenze. Si esegue quindi undersampling per le recensioni con rating 5, 4, 1 e oversampling per le recensioni con rating 2 e 3. Si ottiene un dataset bilanciato con 4000 osservazioni per ciascuna classe. I dati vengono ora suddivisi in un set di train (sulla quale verranno allenati i modelli) e un set di test (sulla quale verranno calcolate le performance classificative) utilizzando uno split 80/20. La suddivisione garantisce che le metriche calcolate siano robuste e che non siano distorte dall'eccessivo adattamento del modello sui dati.

[2] Branca che si occupa di estrarre informazioni dai documenti, in questo caso testuali

2.2.1 Naive Bayes

Il primo modello di apprendimento automatico introdotto è il multinomial Naive Bayes, viene implementato nella libreria Python scikit-learn [13] [3]. Si tratta di un metodo di apprendimento probabilistico basato sul teorema di Bayes, la probabilità che un documento d appartenga alla classe c è calcolata come:

$$P(c \mid d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k \mid c) \tag{2.1}$$

Dove $P(t_k \mid c)$ è la probabilità condizionata del termine t_k di apparire in un documento di classe c. Si interpreta la probabilità come misura di quanto la presenza di t_k contribuisca a confermare che c sia la classe corretta. $P(c)$ è la probabilità a priori che un documento appartenga alla classe c. Qualora i termini di un documento non permettano di collocarlo in una classe specifica, viene scelta quella con la maggiore probabilità a priori. n_d è il numero di termini presenti nel documento in analisi. Trattandosi di un problema di classificazione, l'obiettivo è trovare il rating che meglio rappresenti il documento. Secondo il modello Naive Bayes, la classe migliore corrisponde alla classe con massima verosimiglianza, o *maximum a posteriori* (MAP):

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c \mid d) = \arg\max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k \mid c) \tag{2.2}$$

I parametri $\hat{P}(c)$ e $\hat{P}(t_k \mid c)$ vengono stimati dai dati di addestramento. Nella maggior parte delle implementazioni viene tuttavia utilizzata la log-verosimiglianza: invece che moltiplicare le probabilità se ne sommano i logaritmi. La formula diventa:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \left[\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k \mid c) \right] \tag{2.3}$$

Ogni parametro condizionato della sommatoria rappresenta un peso, esso denota quanto il termine t_k indichi l'appartenenza alla classe c. Similarmente, il valore a priori $\log \hat{P}(c)$, è un peso che indica la frequenza relativa della classe c. I parametri necessari vengono stimati dai dati di train come segue:

$$\hat{P}(c) = \frac{N_c}{N} \qquad\qquad \hat{P}(t \mid c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}} \tag{2.4}$$

Dove N_c corrisponde al numero di documenti appartenenti alla classe c mentre N è il numero totale di documenti. T_{ct} corrisponde al numero di occorrenze del termine t all'interno dei docu-

[3] La documentazione della libreria scikit-learn fa riferimento alla pubblicazione *Introduction to information retrieval* [12]

menti di classe c, contando le occorrenze multiple nei documenti. V corrisponde al vocabolario contenente tutte le parole, il denominatore rappresenta quindi la numerosità del vocabolario limitato alla classe in analisi. Qualora nei dati di test dovessero apparire termini non incontrati durante l'addestramento, si verificherebbe il problema di "probabilità nulla", la soluzione è aggiungere un termine di smorzamento α (usiamo il *Laplace Smoothing* il quale comporta $\alpha = 1$, alternativamente il *Lidstone Smoothing* può assumere valori minori). Le probabilità a posteriori vengono quindi stimate come segue:

$$\hat{P}(t \mid c) = \frac{T_{ct} + 1}{\sum_{t' \in V} (T_{ct'} + 1)} \tag{2.5}$$

Il modello multinomial Naive Bayes, implementato mediante scikit-learn, permette l'elaborazione di documenti rappresentati come vettori TF-IDF. Le prestazioni classificative ottenute dell'algoritmo, valutate sul dataset di test, mostrano un'accuracy inferiore rispetto a quella ottenuta tramite VADER, a favore di un'accuracy off-by-one superiore (fig. 2.4). Confrontando le metriche relative alle singole classi, e le rispettive medie, si nota tuttavia la superiorità dell'algoritmo di apprendimento automatico. L'accuracy ottenuta mediante il modello lexicon-rule based è infatti falsata dalla maggiore numerosità della classe con rating 5, questo poichè, non richiedendo addestramento, il modello è stato applicato sulla totalità dei dati e non su un sottoinsieme bilanciato.

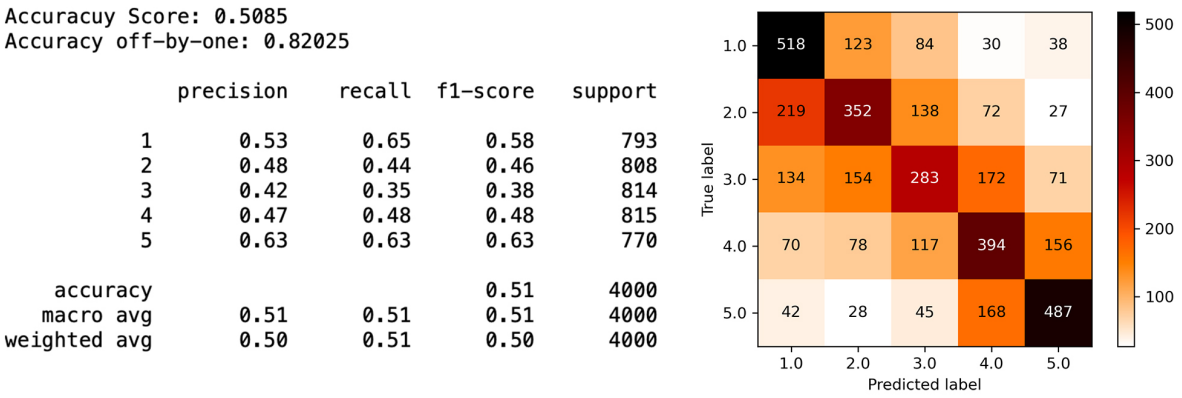

Figura 2.4: Performance classificative ottenute con il multinomial Naive Bayes e matrice di confusione

2.2.2 MultiLayer Perceptron

Il *MultiLayer Perceptron* (MLP) è un algoritmo di apprendimento automatico supervisionato il quale opera imparando la funzione non lineare $f(\cdot) : R^v \rightarrow R^c$ nella fase di addestramento. La dimensione v corrisponde agli input mentre c è il numero di dimensioni in output, nel nostro caso si tratta dei rating. I MLP sono costituiti da almeno tre strati di neuroni: uno strato di input, uno "strato nascosto" e lo strato di output (fig. 2.5).

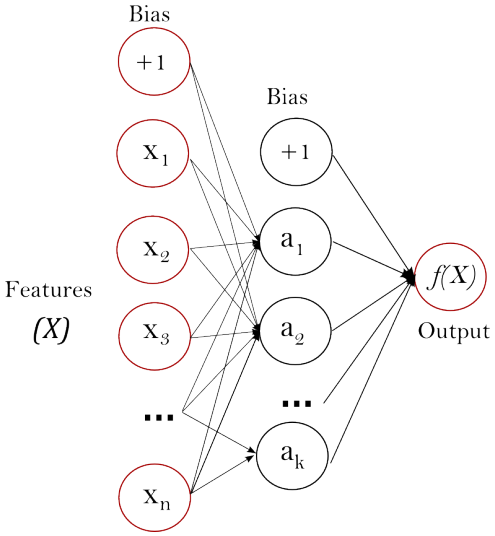

Figura 2.5: MLP con uno strato nascosto

Il primo strato contiene le caratteristiche del modello. Ciascun neurone del livello nascosto trasforma i valori dello strato precedente eseguendo una somma lineare pesata, seguita da una funzione "di attivazione" non lineare $g(\cdot) : R \rightarrow R$ (logistica, tangente iperbolica, relu...). La libreria scikit-learn [13] implementa un algoritmo MLP che supporta problemi previsivi e classificativi. L'addestramento del modello viene eseguito sfruttando la backpropagation: ad ogni iterazione il modello modifica i suoi parametri (pesi e bias) con l'obiettivo di minimizzare una funzione di perdita. Trattandosi di un problema classificativo, la funzione di perdita da minimizzare è l'entropia incrociata (Cross-Entropy loss function), nei problemi previsivi va invece minimizzato il *Mean Square Error* (MSE). L'algoritmo supporta la classificazione multi-classe applicando la funzione softmax (2.7) all'output invece che una funzione logistica.

Vediamo la formulazione matematica nel caso più semplice: un singolo percettrone che esegue una classificazione binaria. Dato un train set $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ dove $x_i \in R^n$ e $y_i \in \{0, 1\}$ la funzione che il modello dovrà imparare è la seguente:

$$f(x) = W_2 g\left(W_1^T x + b_1\right) + b_2 \tag{2.6}$$

Dove W_1, W_2, b_1, b_2 sono i parametri del modello, rispettivamente il peso del livello di input, del livello nascosto e i due rispettivi bias. $g(\cdot) : R \rightarrow R$ è la funzione di attivazione, nel nostro caso viene utilizzata la *REctified Linear Unit* (RELU) la quale restituisce $f(x) = \max(0, x)$. Trattandosi di una problema di classificazione, la funzione ottenuta passerà infine attraverso la funzione logistica $g(z) = 1/(1 + e^{-z})$. Nei problemi multi-classe, invece che la funzione logistica, viene utilizzata la funzione softmax: essa restituisce un vettore contenente le probabilità di appartenenza a ciascuna classe.

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{l=1}^{k} \exp(z_l)} \tag{2.7}$$

Come visto in precedenza, la funzione di perdita da minimizzare varia in base al contesto di analisi. Trattandosi di un problema di classificazione binaria, la funzione da minimizzare è l'entropia incrociata media:

$$\text{Loss}(\hat{y}, y) = -\frac{1}{n} \sum_{i=0}^{n} (y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i)) \tag{2.8}$$

Dove $\hat{y}$ è la classe prevista mentre y è la classe reale, si può notare come uno dei due elementi interni alla sommatoria vada sempre a zero, mentre l'altro aumenti all'aumentare dell'errore di previsione.

Il modello MLPClassifier utilizzato è caratterizzato da tre livelli nascosti composti rispettivamente da 128, 64 e 32 nodi, si tratta di una configurazione classica nell'ambito dell'apprendimento automatico. L'algoritmo utilizzato per ottimizzare l'aggiornamento dei pesi è Adam [10]: si tratta di un algoritmo che ottimizza il gradient descend stocastico basandosi su stime adattive dei momenti di ordine inferiore. L'obiettivo principale è accelerare la convergenza del modello durante l'addestramento regolando in modo adattivo il tasso di apprendimento per ciascun parametro del modello. Il tempo computazionale risulta essere molto superiore rispetto ai modelli addestrati in precedenza, si inizia tuttavia ad osservare un notevole aumento nelle performance classificative (fig. 2.6).

```
Accuracuy Score: 0.5565
Accuracy off-by-one: 0.8125

              precision    recall  f1-score   support

           1       0.60      0.52      0.56       793
           2       0.67      0.79      0.73       808
           3       0.48      0.47      0.47       814
           4       0.43      0.41      0.42       815
           5       0.58      0.59      0.59       770

    accuracy                           0.56      4000
   macro avg       0.55      0.56      0.55      4000
weighted avg       0.55      0.56      0.55      4000
```

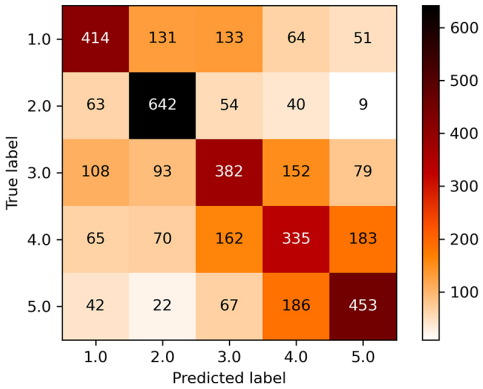

Figura 2.6: Performance classificative ottenute con MLP e matrice di confusione

2.2.3 Support Vector Classification

Le *Support Vector Machines* (SVM) sono una serie di metodi di apprendimento automatico supervisionato utilizzati per risolvere problemi di regressione, di classificazione e di identificazione degli outliers. Questi algoritmi operano costruendo un iperpiano, o una serie di iperpiani, in uno spazio di grosse dimensioni contenente le caratteristiche del modello. La migliore separazione si ottiene con l'iperpiano che ha distanza maggiore rispetto ai punti più vicini di ciascuna classe. Questo poichè maggiore è il margine, minore è l'errore di generalizzazione del classificatore.

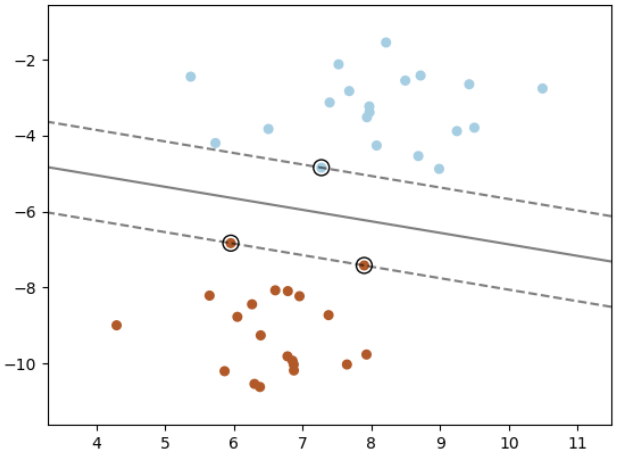

Figura 2.7: Iperpiano generato da un problema classificativo linearmente separabile in uno spazio bidimensionale

La libreria Python scikit-learn [13] implementa un *C-Support Vector Classification* (C-SVC)[4] applicabile a problemi di classificazione binaria, vedremo quindi come opera questa specifica formulazione. L'estensione ai problemi multi-classe è supportata seguendo un approccio "One-vs-One", vengono quindi addestrati classificatori binari per ciascuna coppia di classi.

Il problema di ottimizzazione ha lo scopo di ricercare l'iperpiano che generi i più grandi margini di separazione. Dati $x_i \in R^n, i = 1,\ldots,l$ i vettori di addestramento e le rispettive classi $y \in R^l$ tali che $y_i \in \{1,-1\}$, il problema viene rappresentato dalla seguente formula vincolata:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i\left(w^T\phi\left(x_i\right)+b\right) \geq 1-\xi_i, \\ & \xi_i \geq 0,\ i = 1,\ldots,l \end{aligned} \tag{2.9}$$

w rappresenta il vettore dei pesi, la sua norma al quadrato (spesso riportata mediante trasposizione come $w^T w$) rappresenta una penalizzazione sulla complessità del modello, promuovendo una soluzione che massimizzi il margine di separazione tra le classi utilizzando un vettore dei pesi di dimensioni ridotte. Nella seconda parte della formula, le variabili ξ_i sono chiamate Slack: rappresentano gli errori di classificazione nella fase di addestramento. Il parametro di regolarizzazione *C*, da cui prende il nome la formulazione, controlla il bilanciamento tra la massimizzazione del margine e la minimizzazione degli errori di classificazione. Per elevati valori di C, l'ottimizzazione prediligerà margini inferiori qualora portino l'iperpiano a classificare meglio i dati di addestramento. Un valore inferiore porterà la funzione a massimizzare il margine a costo di commettere errori di misclassificazione (fig. 2.8).

[4] Basato sulla libreria LIBSVM [6]

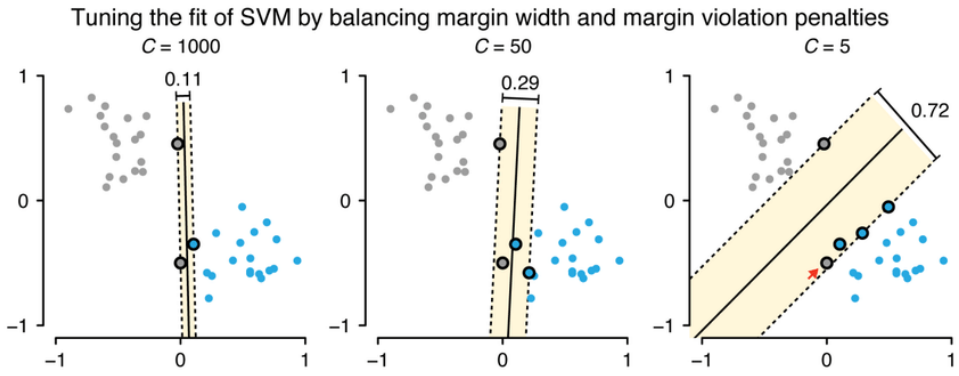

Figura 2.8: Bilanciamento tra massimizzazione del margine e minimizzazione degli errori al variare del parametro di regolarizzazione C [5]

Il vincolo presente nella formula [2.9] impone che i campioni utilizzati per l'addestramento vengano correttamente classificati, o abbiano una violazione controllata del vincolo di separazione, misurata dalle variabili Slack. Si permette un certo scostamento poichè difficilmente esiste un iperpiano tale che $y_i \left(w^T \phi\left(x_i\right) + b\right) \geq 1$ per ciascuna osservazione (ovvero perfetta classificazione). b rappresenta l'intercetta del modello, $\phi\left(x_i\right)$ mappa i vettori di addestramento in uno spazio di dimensione maggiore sfruttando il "Kernel Trick" [18]. Si tratta di una tecnica che permette di effettuare calcoli non lineari senza dover esplicitamente trasformare le caratteristiche originali in uno spazio di dimensioni superiori. In particolare viene utilizzato il kernel *Radial Basis Function* (RBF):

$$K\left(\mathbf{x}, \mathbf{x}'\right) = \exp\left(-\frac{\left\|\mathbf{x} - \mathbf{x}'\right\|^2}{2\sigma^2}\right) \tag{2.10}$$

$\left\|\mathbf{x} - \mathbf{x}'\right\|^2$ rappresenta la distanza euclidea al quadrato tra due osservazioni mentre σ è un parametro che regola l'ampiezza del kernel. Dato che nella pratica il vettore $\mathbf{w}$ potrebbe essere ad alta dimensionalità, viene alternativamente risolto il seguente problema di ottimizzazione duale:

$$\begin{aligned} \min_{\alpha} \quad & \tfrac{1}{2}\alpha^T Q \alpha - e^T \alpha \\ \text{subject to} \quad & y^T \alpha = 0, \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, l \end{aligned} \tag{2.11}$$

Dove e rappresenta un vettore composto da soli 1. La matrice Q ha dimensione $l * l$: gli elementi vengono calcolati come $Q_{ij} = y_i y_j K\left(x_i, x_j\right)$, con $K\left(x_i, x_j\right)$ funzione kernel [2.10]. i termini α_i sono chiamati coefficienti duali. Dopo aver risolto il problema duale, si può mettere in relazione con il primo problema di ottimizzazione. Si determina che il vettore dei pesi otti-

male soddisfa $\mathbf{w} = \sum_{i=1}^{l} y_i \alpha_i \phi(x_i)$. Utilizzando i parametri ottenuti, la funzione decisionale è la seguente:

$$\text{segno}\left(w^T \phi(x) + b\right) = \text{segno}\left(\sum_{i=1}^{l} y_i \alpha_i K(x_i, x) + b\right) \tag{2.12}$$

Come introdotto in precedenza, trattandosi di una problema classificativo multi-classe, vengono automaticamente costruiti $5 * (5-1)/2 = 10$ classificatori C-SVC, ciascuno addestrato rispetto a due classi. Alternativamente si può seguire l'euristica del "One-vs-Rest", in quel caso sarebbero stati costruiti cinque classificatori. Il parametro di regolarizzazione C viene impostato a 5, superiore al valore di default di 1, si tratta di un valore ragionevole rispetto al contesto di analisi e fornisce i risultati migliori. Le performance del modello mostrano un notevole miglioramento rispetto agli approcci visti precedentemente (fig. 2.9). Il modello cattura estremamente bene le caratteristiche che gli permettono di discriminare le classi più complesse correttamente. Osservando le 1601 osservazioni con rating reale pari a 1 o 2, solo il 26% delle previsioni non sono esatte, utilizzando il multinomial Naive Bayes si otteneva un errore del 46%. La media e la media pesata delle metriche calcolate sulle singole classi misurano un aumento di 8 punti percentuali rispetto al MLP.

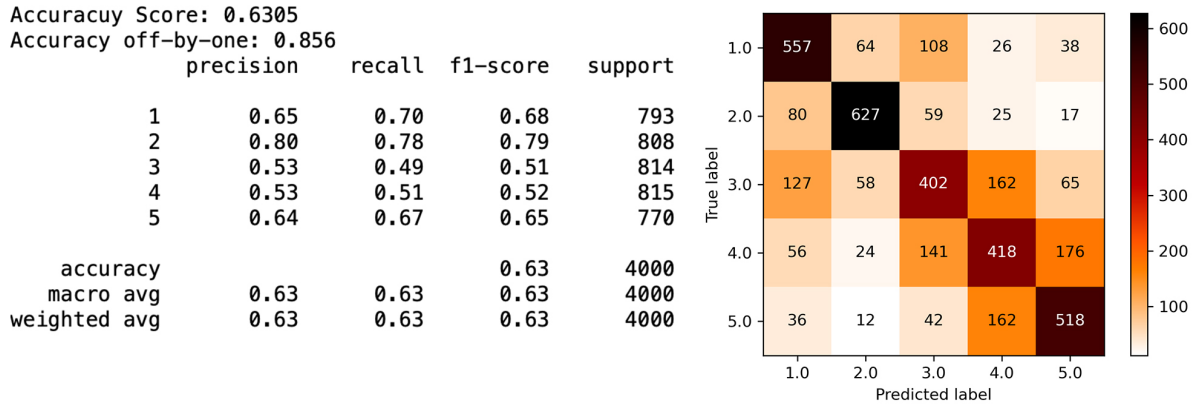

Figura 2.9: Performance classificative ottenute con C-SVC e matrice di confusione

2.2.4 Random Forest

Viene ora introdotto un classificatore appartenente alla famiglia dei modelli "ensemble": essi operano combinando le predizioni di diversi modelli di base con l'obiettivo di aumentarne la robustezza e la generalizzazione. Si tratta di un approccio vincente poiché la variabilità della media dei classificatori è inferiore rispetto alle singole varianze. L'intuizione consiste nello sfruttare modelli che, presi singolarmente, risultano essere instabili per creare modelli più performanti. I classificatori alla base delle random forest sono gli alberi decisionali, vediamo come operano.

Gli alberi decisionali sono metodi di apprendimento automatico non parametrici e supervisionati, vengono utilizzati per la classificazione e la regressione. Il modello, nella fase di addestramento, apprende semplici regole decisionali (fig. 2.10). Negli anni sono stati sviluppati diversi algoritmi: differiscono rispetto agli input supportati, il numero di split che eseguono e la funzione utilizzata per misurare la qualità delle suddivisioni. Gli alberi decisionali, implementati nella libreria Python scikit-learn [13], utilizzano l'algoritmo *Classification and Regression Trees* (CART). Dati $x_i \in R^n, i = 1, \ldots, l$ i vettori di addestramento e le rispettive classi $y \in R^l$, l'albero decisionale partiziona ricorsivamente le osservazioni in modo da raggrupparle rispetto alla classe di appartenenza. I dati in ciascun nodo m vengono rappresentati da Q_m, contenente n_m osservazioni. Ogni possibile separazione viene rappresentato da un vettore di due dimensioni θ, esso consiste in una caratteristica dei dati j e in una soglia t_m. Ciascuna possibile separazione suddivide i dati in due sottoinsiemi Q_m^{left} e Q_m^{right}.

$$\begin{aligned} Q_m^{\text{left}}(\theta) &= \left\{(x,y) \mid x_j \leq t_m\right\} \\ Q_m^{\text{right}}(\theta) &= Q_m \backslash Q_m^{l_{eft}}(\theta) \end{aligned} \tag{2.13}$$

La qualità delle possibili separazioni viene determinata utilizzando una funzione di impurità $H()$, nel nostro contesto di analisi viene utilizzata l'impurità di Gini:

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk}) \tag{2.14}$$

Quest'ultima viene calcolata per entrambi i sottoinsiemi ottenuti e pesata per le rispettive numerosità. In ciascun nodo viene quindi ricercata la separazione che minimizza la seguente quantità:

$$G(Q_m, \theta) = \frac{n_m^{\text{left}}}{n_m} H\left(Q_m^{\text{left}}(\theta)\right) + \frac{n_m^{\text{right}}}{n_m} H\left(Q_m^{\text{right}}(\theta)\right) \tag{2.15}$$

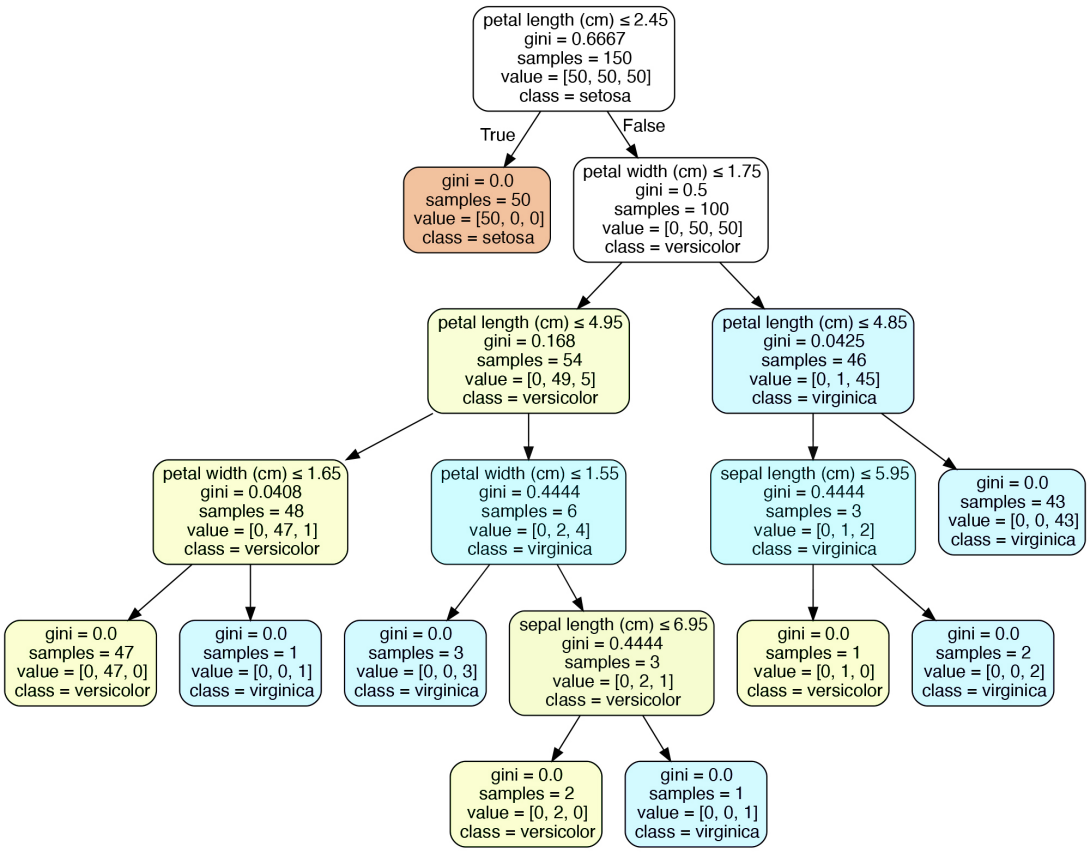

Figura 2.10: Albero decisionale classificativo allenato sulle caratteristiche dell'Iris

La random forest implementata, utilizza una tecnica chiamata "perturbe-and-combine": i modelli di base vengono costruiti sfruttando una componente casuale per aggiungere rumore. I singoli alberi decisionali vengono costruiti a partire da un campionamento casuale di osservazioni estratte del training set. In particolare, l'implementazione utilizzata, estrae campioni con reimmisione: questa tecnica prende il nome di "bootstrap sampling". Ulteriore rumore viene introdotto nella fase di costruzione dei singoli classificatori: ciascuno split viene eseguito su un campionamento casuale di predittori. In contrapposizione rispetto alla pubblicazione originale di Breiman [4], l'implementazione utilizzata combina i classificatori calcolando la media delle previsioni probabilistiche. L'alternativa proposta da Breiman consiste nell'approccio "majority-vote", in cui ciascun albero decisionale vota per una classe e viene scelta la predominante.

Il modello ensemble ottiene performance classificative simili all'algoritmo C-SVC, si registra un lieve aumento dell'accuracy a discapito di 3 punti percentuali nell'accuracy off-by-one. Le metriche calcolate rispetto alle singole classi mostrano un lieve aumento della precision.

```
Accuracuy Score: 0.6315
Accuracy off-by-one: 0.82625
              precision    recall  f1-score   support

           1       0.58      0.77      0.66       793
           2       0.93      0.75      0.83       808
           3       0.58      0.46      0.51       814
           4       0.55      0.52      0.54       815
           5       0.58      0.66      0.62       770

    accuracy                           0.63      4000
   macro avg       0.64      0.63      0.63      4000
weighted avg       0.64      0.63      0.63      4000
```

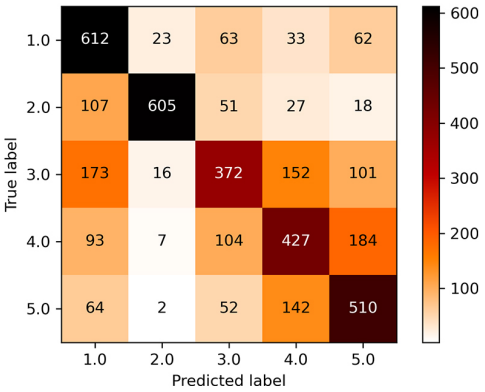

Figura 2.11: Performance classificative ottenute con RF e matrice di confusione

Capitolo 3

Topic Modelling

I topic models sono modelli probabilistici utilizzati per ricercare argomenti astratti all'interno di una collezione di documenti. Si tratta di tecniche non supervisionate dato che, contrariamente al problema classificativo visto in precedenza, non disponiamo degli argomenti reali. I modelli vengono adottati come "black box": si tratta di procedure incapsulate di cui non si conoscono i fondamenti, bisogna avere un certo livello di comprensione del fenomeno o si rischia di trarre conclusioni fallaci. I modelli che verranno applicati definiscono gli argomenti come un insieme di parole correlate, quando appaiono con un valore di frequenza simile in diversi documenti, questo fa presupporre che essi abbiano contenuti simili. Non disponendo a priori degli argomenti, una volta ottenute le parole rappresentative di ciascuno, dovremo manualmente etichettarli: un buon modello ci fornirà argomenti coerenti con il contesto di analisi. La topic modelling richiede una fase di data preparation in cui i documenti vengono processati rimuovendo gli spazi bianchi, i numeri, la punteggiatura, le stopwords e normalizzando il testo in lowercase (come in precedenza). Si può eseguire inoltre stemming o lemmatization: si tratta di due tecniche alternative per ridurre le parole rispettivamente alla loro radice o al loro lemma. A tale scopo è stato utilizzato il WordNetLemmatizer implementato dalla libreria Python *Natural Language ToolKit* (NLTK) [1].

3.1 K-means

In un primo momento viene utilizzato un algoritmo di segmentazione generico, non sviluppato appositamente per l'estrazione degli argomenti latenti. L'algoritmo K-means permette di partizionare le recensioni in K cluster, dove K è un iperparametro da ottimizzare. Le segmentazioni hanno lo scopo di suddividere i dati in gruppi con varianza uguale, minimizzando un criterio noto come inerzia o *Within-Cluster Sum-of-Squares* (WCSS) tramite un approccio iterativo. L'algoritmo riesce a gestire un elevato numero di osservazioni e per questo motivo viene

utilizzato in diversi campi. Nel contesto del topic detection, è stato dimostrato come corpus di grandi dimensioni portino a prestazioni migliori [21]. Dopo aver rappresentato i dati testuali in un formato numerico TF-IDF, come visto in precedenza, l'algoritmo K-means opera come segue:

1. Vengono selezionate casualmente K osservazioni all'interno dello spazio dei dati, ciascuna osservazione rappresenta il centroide di un cluster, ovvero il punto centrale.

2. Viene calcolata una misura di distanza tra ciascuna osservazione e i centroidi, ciascuna osservazione viene quindi assegnata al cluster il cui centroide è più vicino. La metrica utilizzata dall'algoritmo K-means, implementato nella libreria scikit-learn [13], è la distanza euclidea. Viene calcolata come segue:

$$D_{uv} = \sqrt{\sum_{k=1}^{n} (x_{ku} - x_{kv})^2} \tag{3.1}$$

3. Assegnate le osservazioni al cluster più vicino, vengono ora ricalcolati i centroidi. In altre parole viene aggiornato il punto centrale di ciascun cluster.

4. Qualora i centroidi dovessero rimanere invariati (o variare entro una certa soglia), significa che l'algoritmo ha raggiunto la convergenza. Nel caso contrario vengono ripetuti i punti 2. e 3. in modo iterativo, fino a un numero massimo di volte deciso a priori.

Dato abbastanza tempo, l'algoritmo convergerà sempre, potrebbe tuttavia trattarsi di un minimo locale. Per questo motivo l'algoritmo viene inizializzato più volte, cambiando i centroidi di partenza. Come accennato in precedenza, il numero K di cluster va ottimizzato in base alla natura del problema: si può scegliere a priori (qualora si dovesse già avere un'idea sulla segmentazione reale), si può impostare convenzionalmente come $K = \sqrt{n/2}$ (per dataset di piccole dimensioni), altrimenti si possono osservare metriche di omogeneità interna al cluster come la WCSS. Questa metrica andrebbe formalmente minimizzata ma, nella pratica, la soluzione è trovare un compromesso. Altra metrica che viene talvolta utilizzata per la valutazione dell'algoritmo è la *Between-Cluster Sum-of-Squares* (BCSS). La WCSS si ottiene con la seguente formula:

$$\sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{3.2}$$

Dove k è il numero di cluster, S_i è il cluster i-esimo e μ_i è la media del cluster, precedentemente definita come centroide. Durante la convergenza dell'algoritmo K-means, a ciascuna iterazione i centroidi sono calcolati come segue:

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x \tag{3.3}$$

Per scegliere il numero ottimale di cluster, viene quindi rappresentato graficamente il variare della WCSS all'aumentare del numero di cluster, da 1 a 10. Viene poi applicata una comune euristica, definita "Elbow method", utilizzata per l'ottimizzazione matematica (fig. 3.1). In pratica viene scelto il punto in cui la riduzione di WCSS inizia a stabilizzarsi, questo significa che i rendimenti decrescenti non valgono più il costo aggiuntivo, ovvero l'aumento del numero di cluster.

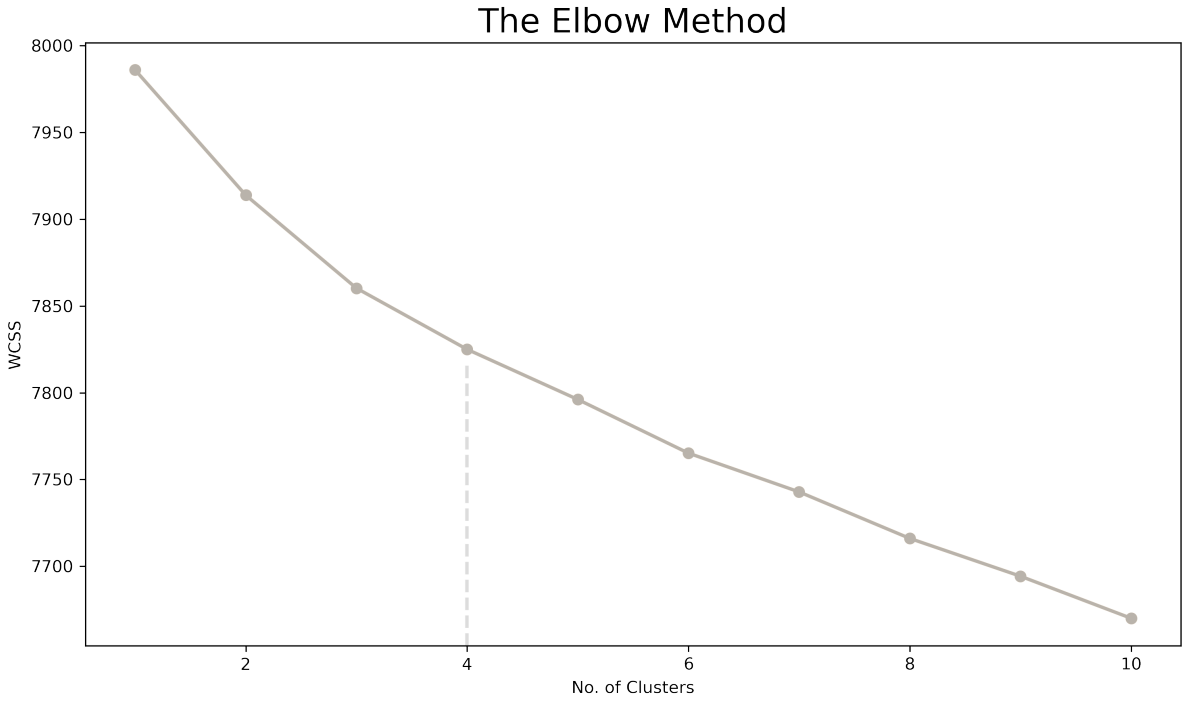

Figura 3.1: Variazione del WCSS all'aumentare del numero di cluster

Osservando il grafico ottenuto si decide di impostare il problema con un numero di segmentazioni pari a 4. Il "gomito" in figura non è eccessivamente marcato, probabilmente inizializzare l'algoritmo con un numero superiore di cluster fornirebbe risultati simili, sarebbero tuttavia di più difficile interpretazione. I topic ottenuti, elaborando le recensioni con almeno 30 caratteri, sono identificati dalle parole presenti nella tabella 3.1, ordinate per importanza decrescente. I cluster ottenuti hanno dimensioni molto variabili: nel terzo topic vengono collocate più della metà delle recensioni. Provando a interpretare i termini più significativi, il terzo topic sembra contenere recensioni positive, in generale apprezzamenti verso il brand. Disponendo del centroide di ciascun cluster, è possibile osservare la recensione ad esso più vicina. Nel terzo cluster si tratta della seguente:

"These are actual apple buds. Came in authentic apple box. sound great and nice heavier duty cord. Ad pictures do not show the mic/control pod, but it is there. Don't take chances on a cheap knock-off ... these are the quality buds."

I restanti topic, meno numerosi, isolano argomenti specifici e di facile interpretazione, *Pods*, *Cable* e *Keyboard* sono i prodotti più numerosi e questo influenza chiaramente l'algoritmo. Si nota inoltre come la parola "apple" sia importante rispetto a tutti i topic.

topic 1	topic 2	topic 3	topic 4
4898	7751	23693	2879
charger	sound	apple	keyboard
charge	ear	wa	ipad
phone	noise	work	key
charging	airpods	product	magic
cord	quality	ipad	apple
apple	great	great	use
work	cancellation	use	like
iphone	good	love	case
case	fit	like	work
cable	headphone	new	wa

Tabella 3.1: Numerosità e termini più rappresentativi rispetto a ciascun topic

Non disponendo degli argomenti reali, non è possibile valutare il modello con le metriche viste in precedenza (accuracy, precision, etc.). L'algoritmo K-means si può valutare utilizzando il coefficiente di Silhouette: questa metrica viene calcolata rispetto a ciascuna osservazione e viene interpretata la media complessiva. Rispetto a una singola osservazione, viene calcolato come segue:

$$s = \frac{b-a}{\max(a,b)} \tag{3.4}$$

Dove a equivale alla distanza media tra l'osservazione e tutte le osservazioni appartenenti allo stesso cluster mentre b equivale alla distanza media tra l'osservazione e tutte le osservazioni appartenenti al cluster più vicino. Il coefficiente di Silhouette calcolato sulla totalità dei dati assume valore da -1 (clusterizzazione errata) a 1 (cluster densi e separati), dove 0 significa che i cluster sono essenzialmente sovrapposti. Calcolando il coefficiente rispetto alle segmentazioni precedentemente definite, si ottiene un valore pari a 0.008. Aumentare il valore di K potrebbe sicuramente migliorare il coefficiente di Silhouette, tuttavia, dato il contesto di analisi, una certa sovrapposizione è prevista.

Per enfatizzare la ricerca degli argomenti latenti sulle problematiche che emergono dalle recensioni negative, si decide di restringere l'analisi alle recensioni con rating inferiore a 3 e di rimuovere i nomi propri dei prodotti e la parola "apple". Il numero di topic non viene modificato, le parole identificative sono visibili nella tabella 3.1. Si riescono a isolare argomenti non presenti nell'analisi precedente, essi risultavano essere mascherati dalle numerose recensioni positive. Nel quarto topic osserviamo parole come "used", "return", "box": vengono probabilmente collocate recensioni negative nei confronti di Amazon, del venditore o del sistema di spedizione in generale. Questa interpretazione viene confermata osservando la recensione più prossima al centroide:

> *"This was NOT a new product. This was the same price as a brand new product, but it was definitely used, and not in the original packing. SCAM."*

Il secondo topic è facilmente interpretabile, contiene recensioni che lamentano prodotti i quali smettono di funzionare in una ristretta finestra temporale. Allo stesso modo il terzo topic, similarmente all'analisi condotta in precedenza, evidenzia chiaramente problematiche legate al caricamento. Si nota inoltre come, nel sottoinsieme di recensioni negative, i cavi sono i prodotti più presenti. Il primo topic, seppur il più numeroso, è di più difficile interpretazione. Confrontandolo con la segmentazione eseguita in precedenza, possiamo aspettarci che contenga recensioni riguardanti problemi di audio; tuttavia, le parole più importanti, non sembrano evidenziare un argomento specifico.

topic 1	topic 2	topic 3	topic 4
3753	573	1170	1379
work	stopped	charger	wa
product	working	charge	amazon
use	month	phone	product
month	worked	charging	new
time	week	work	box
like	product	battery	item
ear	charging	case	used
buy	charger	usb	received
bought	bought	plug	return
sound	return	doe	work

Tabella 3.2: Numerosità e termini più rappresentativi rispetto a ciascun topic negativo

Nonostante la semplicità computazionale, l'algoritmo K-means fornisce risultati interessanti e facilmente interpretabili. Verrà ora utilizzato un modello probabilistico sviluppato apposta per l'estrazione degli argomenti latenti.

3.2 Latent Dirichlet Allocation

La *Latent Dirichlet Allocation* (LDA) viene descritta come un modello probabilistico utilizzato per collezioni di dati discreti, come ad esempio documenti testuali. L'idea alla base della LDA è che i documenti vengano rappresentati come miscele di argomenti latenti, dove ogni argomento viene rappresentato da una distribuzione di parole. Il procedimento eseguito a mano potrebbe somigliare a quanto contenuto in figura 3.2. Vengono evidenziate con colori diversi parole che identificano argomenti specifici, il documento può essere quindi visto come una combinazione di topic diversi, in diverse proporzioni.

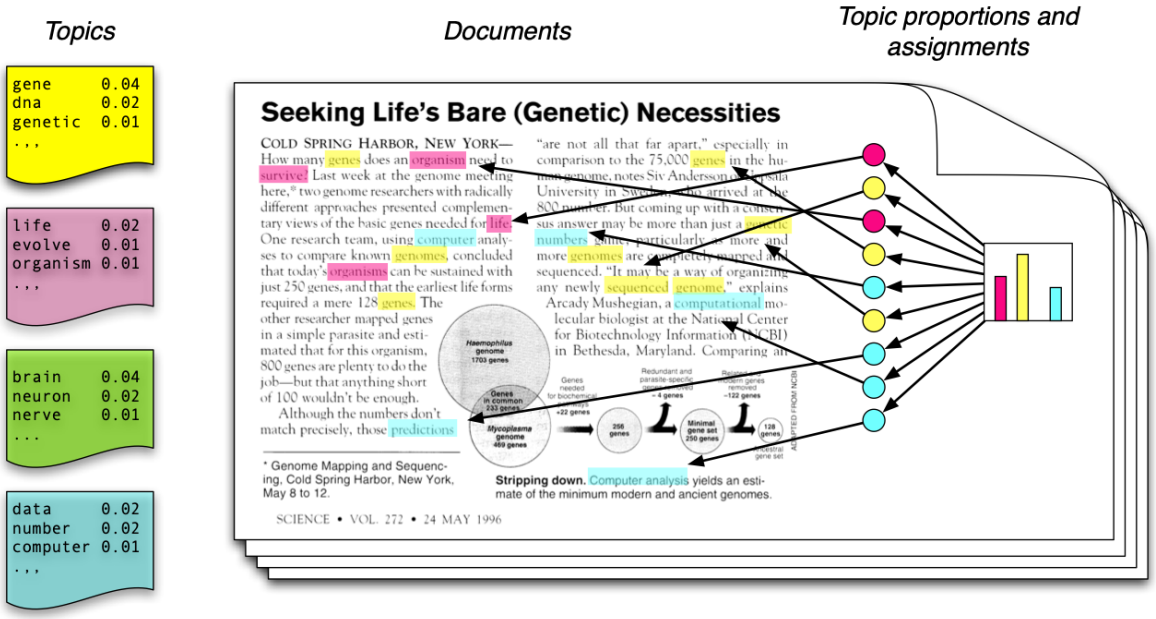

Figura 3.2: Intuizione dietro la Latent Dirichlet Allocation [3]

Per comprendere l'intuizione probabilistica dietro al modello, bisogna immaginare che i nostri dati reali nascano da un processo generativo, esso definisce una distribuzione di probabilità congiunta tra le variabili osservate e le variabili nascoste. La distribuzione congiunta viene utilizzata per calcolare la distribuzione condizionata delle variabili nascoste date le variabili osservate. Le variabili osservate sono le parole dei documenti, quelle nascoste sono invece le strutture dei topic. Il problema computazionale risiede quindi nel calcolare la distribuzione a posteriori, ovvero la distribuzione condizionale delle variabili nascoste.

Il modello LDA è definito dalla notazione 3.5. Gli argomenti latenti sono $\beta_{1:K}$, dove ogni β_k è una distribuzione di parole nel vocabolario. La proporzione di argomenti per il d-esimo documento è θ_d, dove $\theta_{d,k}$ è la proporzione dell'argomento k nel documento d. Gli argomenti assegnati al documento d-esimo sono z_d, dove $z_{d,n}$ è l'argomento assegnato alla parola n-esima

del documento d. Infine, le parole osservate nel documento d sono w_d, con $w_{d,n}$ n-esima parola del documento.

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p\left(z_{d,n} \mid \theta_d\right) p\left(w_{d,n} \mid \beta_{1:K}, z_{d,n}\right) \right) \quad (3.5)$$

Il processo generativo a cui fa riferimento la notazione, viene rappresentato graficamente come in figura 3.3. α e η sono distribuzione Dirichlet, θ e β sono distribuzioni Multinomiali. W e Z sono rispettivamente le parole e gli argomenti generati, in relazione alla collezione N di parole nel documento d e alla totalità dei documenti D. K definisce infine gli argomenti astratti.

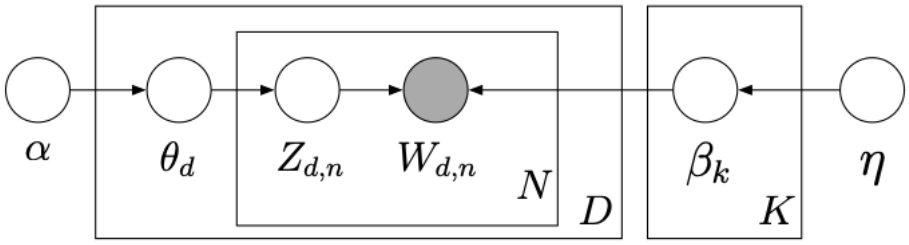

Figura 3.3: Rappresentazione grafica del modello LDA [3]

Il modello LDA utilizzato viene implementato nella libreria Python Gensim [14]. Oltre al pre-processing eseguito in precedenza, vengono generati i bigrammi: le coppie di parole che appaiono vicine almeno un numero n di volte, generano un nuovo token (*es:* dato che i termini "stopped" e "working" appaiono spesso adiacenti, viene creato un nuovo token "stopped_working"). Le osservazioni vengono inoltre convertite in formato *Bag of Word* (BoW): si tratta di una rappresentazione numerica diversa dal formato TF-IDF. Ciascun documento viene rappresentato come un vettore di frequenza delle parole, si perde informazione rispetto all'ordine delle parole nel documento, per questo motivo è utile introdurre i bigrammi. Viene preferita questa rappresentazione poichè, nel contesto di analisi, fornisce prestazioni migliori: elaborando i dati in formato TF-IDF, vengono identificati argomenti più di nicchia. Questo fenomeno si verifica poichè la rappresentazione TF-IDF valuta maggiormente i termini rari, per rendere il modello più efficiente bisognerebbe aumentare il numero di topic, a discapito dell'interpretabilità.

Applicando il modello alle recensioni con almeno 30 caratteri, come in precedenza, si ottengono i quattro topic caratterizzati dalle parole in tabella 3.2. Anche in questo caso, ricercare gli argomenti latenti sulla totalità delle recensioni, non fornisce risultati particolarmente interessanti. La prevalenza di recensioni positive e di una certa gamma di prodotti è resa ancora più evidente dalla composizione degli argomenti ottenuti con la LDA. Il terzo e il quarto topic riguardano rispettivamente le recensioni relative a *Pods* e *Cable*, come confermato osservando la distribuzione dei prodotti rispetto ai topic. Il primo raggruppa soprattutto le recensioni relative agli *Ipad* e alle *Keyboard*, il secondo topic contiene invece un elevato numero di recensioni relative a *Watch* e *Iphone*. Si nota inoltre come la parola "apple" risulta essere il termine più rilevante in due dei quattro topic.

topic 1	topic 2	topic 3	topic 4
11944	5004	11748	16577
ipad	apple	sound	apple
keyboard	watch	airpods	work
use	phone	ear	product
apple	use	quality	one
pro	case	great	cable
work	iphone	noise	charge
like	one	good	charger
great	time	battery	charging
screen	device	life	new
macbook	get	headphone	amazon

Tabella 3.3: Numerosità e termini più rilevanti rispetto a ciascun topic

In precedenza, i cluster generati, potevano essere interpretati osservandone i rispettivi centroidi, il modello LDA opera però diversamente rispetto al modello K-means. Si può osservare un elemento rappresentativo, rispetto a ciascun topic, estraendo l'osservazione con metrica "relevance" più alta, si tratta di un valore compreso tra 0 a 1 che indica la probabilità di appartenenza dell'osservazione a ciascun topic. Si può osservare, ad esempio, la recensione con maggiore probabilità di appartenenza al terzo topic:

> *"The new AirPods Pro 2nd generation have several notable improvements over their predecessor. One of the most significant upgrades is the addition of Spatial Audio technology, which creates an immersive sound experience by simulating a surround sound environment. [...] Overall, the AirPods Pro 2nd generation are a worthy upgrade over the previous generation, offering improved sound quality, noise cancellation, battery life, and a more comfortable fit. If you are looking for a high-quality and reliable pair of wireless earbuds, the AirPods Pro 2nd generation are definitely worth considering."*

Come in precedenza, si osserva come varia la composizione dei topic limitando l'analisi alle sole recensioni con rating inferiore a 3 e rimuovendo i nomi propri dei prodotti (tab. 3.2). Gli argomenti estratti risultano essere gli stessi ottenuti in precedenza, tuttavia i termini più rilevanti sono di più facile interpretazione. Il modello riesce a catturare al meglio gli argomenti latenti presenti nel corpus, tra le parole rilevanti notiamo meno sovrapposizioni e una distinzione complessivamente più accentuata.

topic 1	topic 2	topic 3	topic 4
1828	2813	1470	1750
month	charge	ear	product
working	battery	work	new
one	charger	sound	amazon
stopped	use	time	box
product	charging	issue	one
year	usb	one	item
stopped_working	get	noise	work
bought	case	get	return
charger	adapter	pair	key

Tabella 3.4: Numerosità e termini più rilevanti rispetto a ciascun topic negativo

Il bigramma "stopped_working", affiancato da diverse locuzioni temporali, riassume inequivocabilmente il contenuto delle recensioni appartenenti al primo topic. Il secondo topic è il più diffuso, ci aspettiamo che vi appartengano un alto numero di recensioni relative ai *Cable*, ma anche prodotti con problemi di batteria o di caricamento indipendente dai dispositivi di alimentazione. Anche in questa fase emerge chiaramente un topic riguardante problemi di audio, imputabile in gran parte a *Pods* e *Beats*. Tralasciando i termini già osservati in precedenza, emerge per la prima volta il termine "pair", sicuramente importante rispetto al contesto evidenziato. Per concludere, il topic riguardante problematiche di spedizione, imballaggio, prodotto errato o usato, è composto pressoché dagli stessi termini rappresentativi visti in precedenza. In un ottica di analisi aziendale, il quarto topic non è imputabile al brand, questo poichè non è Apple a gestire le vendite su amazon, bensì venditori terzi. Nella tabella 3.5 vengono riportate, rispetto a ciascun topic negativo, due recensioni, non eccessivamente lunghe, caratterizzate rispettivamente da un valore di relevance molto elevato e da un valore di relevance basso. Le prime ci mostrano come il modello catturi egregiamente la problematica descritta. Le recensioni con relevance bassa risultano essere generalmente più ambigue, osservandole attentamente si nota tuttavia come la scelta del topic sia comunque corretta in tre casi su quattro.

problem	product	text	relevance
early breakage	Cable	*Very poor quality. I bought it on 5/16/20, and it's not been two months yet. It doesn't work anymore. It costs $29, and not cheap for two months.*	0.95
	Cable	*my phone has not even loaded 50% and this battery already died*	0.36
battery/charging problems	Cable	*My adapters get heavy use and these do not hold up. Going to try one of the cheaper brands... Even if they break, I can get 4 of them for the same price.*	0.95
	Cable	*As others have said, stay away from this power adapter. Only worked 9 months for me, which is longer than some of the horror stories on here.*	0.37
sound/earphone problems	Pods	*I had upgraded from my gen 1 airpods to these but switched back :(The noise cancellation on them was so weird and made my ears hurt as well as my head*	0.95
	Pods	*They work perfect only problem is the light on the case not working, besides that great*	0.35
amazon/dispatch problems	Watch	*I received a used band. I paid for a new band not a used one. Box was open and everything was stuffed back in the box.*	0.95
	Cable	*Wire seems low quality and can't see but the inside looks wired bad and seems like it would bend and crack the wire*	0.35

Tabella 3.5: Recensioni brevi con rilevanza elevata e con rilevanza bassa rispetto a ciascun topic

Conclusioni

Il procedimento di estrazione dei dati ha permesso di confrontarsi con problematiche diffuse nel contesto del web-scraping. Il codice realizzato risulta essere consistente rispetto a ciascuna ricerca sul marketplace di Amazon.com, si tratta inoltre di un ottimo punto di partenza qualora si vogliano estrarre informazioni simili da altri siti internet. Gestire dati reali si è rivelato complesso in diverse fasi del progetto: le recensioni scaricate, seppur filtrate per lingua inglese, contenevano piccole percentuali di lingue straniere, le quali rovinavano le prestazioni dei modelli utilizzati. Altre recensioni, originariamente contenenti immagini o video, presentavano righe di testo standard, esse sono state rimosse poiché influenzavano notevolmente i risultati dei topic models. Utilizzare dati didattici già processati e bilanciati, come ad esempio i numerosi dataset di recensioni presenti su Kaggle, avrebbe reso più semplici le analisi. La scelta di utilizzare dati reali ha tuttavia permesso di misurarsi con sfide interessanti, ha inoltre fornito un'idea della mole di dati generati da certe piattaforme. Per limiti computazionali l'analisi è stata condotta su una "piccola" quantità di dati, è tuttavia degno di nota come, rimanendo nei confini di Amazon, la ricerca "Apple" senza l'applicazione di filtri avrebbe fornito più di 50.000 prodotti diversi.

L'analisi del sentimento è stata impostata ordinando i modelli in ordine crescente di prestazioni classificative. Nonostante la semplicità computazionale, l'approccio basato su caratteristiche lessicali e regole sintattiche ha fornito prestazioni discrete. Va inoltre tenuto in considerazione come l'analisi coinvolga un numero di classi superiore rispetto al negativo-neutrale-positivo che VADER è solito produrre. La soluzione di ricercare le soglie ottimali ha permesso di aumentare le prestazioni classificative, tuttavia, non avendo calcolato le metriche rispetto ad un sottoinsieme di dati di validazione, le soglie seguono sicuramente il rumore dei dati. Va prestata attenzione qualora si vogliano confrontare le prestazioni di VADER con quelle ottenute mediante i modelli di apprendimento automatico introdotti in seguito: nonostante il discreto valore di accuracy, esso risulta essere falsato dall'elevata numerosità di certe classi. I rating centrali sono più difficili da catturare, come si evince osservando le metriche calcolate sulle singole classi. Non necessitando di una fase di addestramento, VADER è stato applicato al dataset completo, seppur sbilanciato. Per poter eseguire un confronto equo con i modelli introdotti in

seguito, avremmo dovuto applicarlo esclusivamente al sottoinsieme di dati di validazione utilizzato nella fase successiva. I modelli di apprendimento automatico introdotti hanno permesso di raggiungere elevate prestazioni classificative. Non si possono confrontare con gli studi precedentemente eseguiti sulle recensioni, riportati nell'introduzione, dato che essi normalizzano il problema classificativo strettamente a due classi. L'analisi condotta ha confermato tuttavia le elevate prestazioni delle SVM addestrate su dati in formato TF-IDF, solo i modelli ensamble riescono a superare leggermente l'accuratezza raggiunta. Disponendo di una potenza computazionale maggiore si sarebbe potuto valutare l'utilizzo di modelli fine-tuned. Utilizzando modelli state-of-art come BERT, ulteriormente addestrati sui nostri documenti, si sarebbero potute raggiungere probabilmente prestazioni classificative leggermente più elevate.

La ricerca degli argomenti latenti tramite entrambi gli approcci lascia spazio ad analisi successive. Si sarebbe potuto limitare il numero di recensioni inerente a ciascuna tipologia di prodotto per evitare l'evidente distorsione dovuta alla predominanza di *Pods* e *Cable*. Alternativamente, sarebbe stato interessante ricercare gli argomenti latenti rispetto a ciascuna tipologia di prodotti singolarmente. In entrambe le analisi condotte è stata seguita una specifica pipeline di preprocessing con l'obiettivo di alleggerire il carico computazionale, disponendo di risorse maggiori si sarebbero potuti ricavare anche i trigrammi, i quali avrebbero incrementato il numero di token da elaborare. Complessivamente l'analisi relativa alla recensioni negative ha fornito risultati facilmente interpretabili, i quali confermano tuttavia la necessità di conoscere il contesto di analisi per poter valutare le prestazioni degli algoritmi. Come osservato in precedenza, qualora la priorità dovesse essere quella di riconoscere una grande quantità di problematiche astratte contenute nelle recensioni a discapito dell'interpretabilità, potrebbe essere interessante aumentare il numero di topic, o alternativamente utilizzare rappresentazioni vettoriali differenti.

Bibliografia

[1] Steven Bird, Ewan Klein e Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[2] David M Blei, Andrew Y Ng e Michael I Jordan. «Latent dirichlet allocation». In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[3] David M. Blei. «Introduction to Probabilistic Topic Models». In: 2010.

[4] Leo Breiman. «Random forests». In: *Machine learning* 45 (2001), pp. 5–32.

[5] Danilo Bzdok, Martin Krzywinski e Naomi Altman. «Machine learning: Supervised methods». In: *Nature Methods* 15 (gen. 2018). DOI: 10.1038/nmeth.4551.

[6] Chih-Chung Chang e Chih-Jen Lin. «LIBSVM: A library for support vector machines». In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 27:1–27:27.

[7] Scott Deerwester et al. «Indexing by latent semantic analysis». In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407.

[8] Rong-En Fan et al. «LIBLINEAR: A library for large linear classification». In: *the Journal of machine Learning research* 9 (2008), pp. 1871–1874.

[9] Docker Inc. *Docker*. 2013. URL: https://www.docker.com.

[10] Diederik P Kingma e Jimmy Ba. «Adam: A method for stochastic optimization». In: *arXiv preprint arXiv:1412.6980* (2014).

[11] Andrew Maas et al. «Learning word vectors for sentiment analysis». In: *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 2011, pp. 142–150.

[12] IC Mogotsi. *Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze: Introduction to information retrieval: Cambridge University Press, Cambridge, England, 2008, 482 pp, ISBN: 978-0-521-86571-5*. 2010.

[13] F. Pedregosa et al. «Scikit-learn: Machine Learning in Python». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[14] Radim Rehurek e Petr Sojka. «Gensim–python framework for vector space modelling». In: *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3.2 (2011).

[15] Kenneth Reitz. *Requests: HTTP for Humans*. 2011. URL: https://requests.readthedocs.io/en/latest/.

[16] Scrapinghub Revision. *Splash - A javascript rendering service*. 2014. URL: https://splash.readthedocs.io/en/stable/index.html#splash-a-javascript-rendering-service.

[17] Leonard Richardson. *Beautiful Soup*. 2004. URL: https://www.crummy.com/software/BeautifulSoup/.

[18] Bernhard Schölkopf. «The kernel trick for distances». In: *Advances in neural information processing systems* 13 (2000).

[19] Alan M Turing. *Computing machinery and intelligence*. Springer, 2009.

[20] Vamsi Vallurupalli e Indranil Bose. «Exploring thematic composition of online reviews: A topic modeling approach». In: *Electronic Markets* 30 (2020), pp. 791–804.

[21] Dan Zhang e Shengdong Li. «Topic detection based on K-means». In: *2011 International Conference on Electronics, Communications and Control (ICECC)*. IEEE. 2011, pp. 2983–2985.

Elenco delle figure

Elenco delle tabelle