AMAZON FINE FOOD REVIEWS: PROGETTO DI TEXT MINING
Clustering e Classification

Francesco Anello $_{829213}$, Francesco Fustini $_{830697}$, Laura Rapino $_{831346}$
MSc in Data Science

Abstract. Lo studio riportato parte da un insieme di dati riguardanti delle recensioni di Amazon su una serie di prodotti cibari, si analizzano e si applicano dei metodi di classificazione e clustering. I modelli usati nel primo caso sono l'XGBoost, l'SVM e il k-NN che hanno riportato risultati piuttosto simili, mentre sono stati usati il k-Means e il metodo gerarchico per il Clustering. I risultati ottenuti non propendono verso un'unica soluzione che abbia metriche nettamente migliori.

Keywords: Text Mining, Reviews, Clustering, Classification

Introduzione

Il progetto si pone come obiettivo l'analisi del dataset "Amazon Fine Food Reviews" tramite tecniche di Text Mining. A un'opportuna preparazione dei dati tramite una fase preprocessing e di text representation è seguita un'elaborazione atta alla riduzione delle dimensioni del dataset. Tali procedimenti fondamentali per lo studio dei dati, hanno permesso di arrivare a conseguire l'obiettivo preposto, ovvero concludere uno studio di classificazione e di clustering usando i seguenti modelli: XGBoost, SVM, k-NN, k-Means e metodo gerarchico per il clustering.

1. Dati

1.1. **Il dataset.** Il dataset riporta 568'454 recensioni su 74'258 prodotti acquistati sulla piattaforma di Amazon nel corso di 13 anni, da Ottobre 1999 a Ottobre 2012.

1.2. **Le variabili.** Le variabili comprendono:

- *Id*: Id di riga
- *ProductId*: Id del prodotto
- *UserId*: Id dell'utente su Amazon
- *ProfileName*: Nome del profilo utente su Amazon
- *HelpfulnessNumerator*: Numero di utenti che hanno trovato utile la recensione
- *HelpfulnessDenominator*: Numero di utenti che hanno segnalato la recensione come utile o meno
- *Score*: Punteggio assegnato alla review
- *Time*: Timestamp Unix in secondi della review
- *Summary*: Titolo della review

- *Text*: Testo della review

1.3. **Analisi esplorativa.** Si riportano le analisi compiute sul dataset fornito:

- La variabile *Score* presenta una distribuzione asimmetrica come si mostra di seguito, ovvero la maggior parte delle recensioni hanno un valore di *Score* pari a 5:

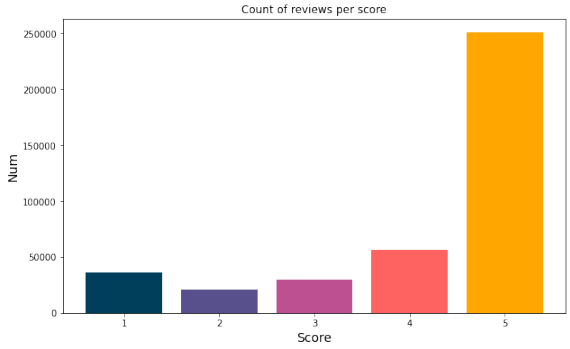

Figure 1. Distribuzione della variabile *Score*.

- Si visualizza la lunghezza media della variabile *Text*, quindi il numero medio di caratteri delle recensioni per punteggio di *Score*:

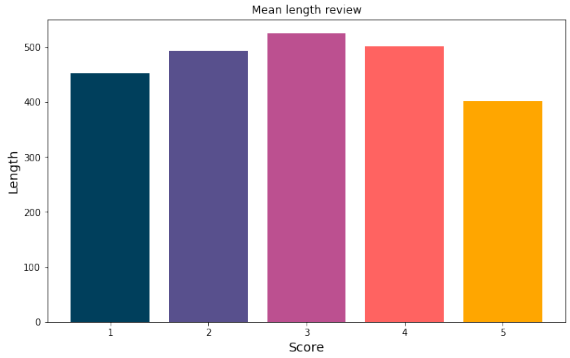

Figure 2. Lunghezza media delle recensioni (in numero di caratteri) per *Score*.

2. Preprocessing

2.1. **Data Cleaning.** Innanzitutto si identificano di eventuali duplicati considerando le colonne "UserId","ProfileName","Time", "Text" per poi procedere alla loro eliminazione. Inoltre per gli obiettivi successivi é utile applicare una tecnica di Undersampling per bilanciare le classi di Score mantenendo tutte le classi alla numerosità della classe minore.

2.2. **Lower Case, Decontraction, Remove Blankspace and Numbers.** Successivamente tutte le parole sono convertite in minuscolo, le contrazioni vengono eliminate convertendole in parole intere (es: "...n't" diventa "... not"). Poi vengono rimossi i numeri e gli spazi bianchi.

2.3. **Tokenization.** L'obiettivo in questa fase, fondamentale per l'elaborazione successiva dei contenuti, consiste nel suddividere i testi in singole unità.

2.4. **Remove Stopwords.** Con la rimozione delle stopwords, il dataset viene ripulito di parole che non contribuiscono a delineare il senso della recensione. Sono state perciò rimosse tutte le parole contenute in una lista precostruita di un pacchetto di Python, "Natural Language ToolKit".

2.5. **Lemmatization and Stemming.** La lemmatization consente di ridurre le parole alla propria radice, per snellire così l'analisi delle parole usate nelle recensioni. Questo è stato possibile grazie alla funzione di Wordnet, un database lessicale in inglese in grado di stabilire relazioni semantiche tra le parole.
Infine, lo stemming permette di riportare una parola dalla sua forma flessa alla sua radice: da una parte vi è il rischio di perdere informazioni precise del significato di alcune parole ma dall'altra permette un notevole guadagno in termini di memoria acquisita. In particolare lo stemming utilizzato é lo Stemmer di Porter.

3. Text Representation

Nella fase di Text Representation, utile all'Information Retrieval, si usano tecniche che aiutino a strutturare i dati a partire da un dataset di dati non strutturati. Usando scikit-learn é possibile convertire i dati lemmizzati e stemmizzati in matrici di conteggio di token (Bag of Words) e in matrici Tf-idf. Per entrambe le tipologie di rappresentazioni si considerano unigrammi e bigrammi, selezionandone i 30'000 più frequenti.

3.1. **Bag of Words.** Il modello Bag of Words permette di elaborare il linguaggio naturale usato nelle recensioni senza tener conto dell'ordine delle parole.

3.2. **TF-IDF.** Come ultimo step di questa fase, ci si propone di creare e applicare dei pesi per le parole considerate. Più specificatamente, si calcola un peso per i termini tramite l'indice TF-IDF che è ricavato dalla combinazione del *term frequency* normalizzato e del peso idf_t, ottenendo così la formula:

$$W_{t,d} = TF_{t,d} \times IDF_t$$

Ovvero:

$$W_{t,d} = \frac{tf_{t,d}}{max_{ti}(tf_{ti,d})} \times log(\frac{N}{df_t})$$

Dove $tf_{t,d}$ rappresenta il numero di volte in cui il termine t appare nel documento d e df_t è il numero di documenti in cui appare il medesimo termine.
Tale misura è fondamentale nell'Information Retrieval e cresce all'aumentare delle occorrenze di un termine in un documento e al diminuire di una certa parola nel documento.

4. Riduzione della dimensionalità

Per prevenire problematiche legate all'overfitting e alleggerire il carico computazionale, si procede a una riduzione della dimensionalità tramite l'approccio dell'SVD: la decomposizione ai valori singolari è una tecnica proveniente che permette di fattorizzare una matrice riducendo così la mole di dati. Vengono infatti ridotte le dimensione della matrice di co-occorrenza X tramite metodi dell'algebra lineare. E' stata applicato questo metodo, piuttosto che la PCA, dal momento che risulta piuttosto efficiente con matrici sparse. Tale fase di riduzione della dimensionalità permetterà di ottenere delle stime più robuste ma soprattutto più computazionalmente eseguibili, risultando necessaria per proseguire con le procedure descritte nei prossimi paragrafi. Sia per le BoW sia per le matrici Tf-idf si decide di tenere 300 componenti.

5. Text Classification

Per la classificazione delle recensioni si é deciso di considerare tre metodi differenti: XGboost, Support Vector Machine (SVM) e k-nearest neighbors (k-NN).
Prima di applicare i modelli, si suddividono i dati in training (70%) e test (30%) per tutte e quattro le diverse rappresentazioni; si procede, poi, ad una loro standardizzazione.
Per ognuno dei tre algoritmi sopracitati si applica un GridSearch CV su specifici parametri dei modelli, con una cross validation in 3 fold, in modo da valutare possibili valori, diversi da quelli di default, che possano portare a risultati migliori. Ciò viene fatto sulla rappresentazione stemmed BoW. I parametri che risultano essere i migliori per questa rappresentazione vengono poi utilizzati per tutte le altre. Per valutare le performance di questi modelli si visualizza un report che mostri le principali metriche di classificazione: precision, recall, f-score e, infine, l'accuracy. Per ultimo viene calcolata la matrice di confusione per avere una rappresentazione più efficace dell'accuratezza della classificazione.

5.1. **XGBoost.** La classificazione con metodo XGBoost è applicata tramite l'implementazione di XGBClassifier: i parametri su cui é stato utilizzato il GridSearchCV sono gamma (con valori 0, 1, 1.5) e max-depth (1, 3, 6). Le performance migliori sul stemmed BoW sono ottenute con gamma=0 e max-depth=6, che solo anche i valori di default del modello. Questi parametri vengono quindi applicati per gli altri modelli XGBoost trainati e testati sulle altre tre diverse rappresentazioni testuali. Le accuracy oscillano tra il 42% e il 44% con f-score sempre superiori allo 0.50 per le due classi più estreme. Si riportano qui nel dettaglio le metriche e la matrice di confusione del modello migliore, il quale é stato ottenuto sui dati stemmed tfidf (accuracy=44%).

Class	***Precision***	***Recall***	***F-score***
1	0.52	0.57	0.54
2	0.34	0.32	0.33
3	0.36	0.33	0.35
4	0.39	0.38	0.38
5	0.55	0.60	0.57

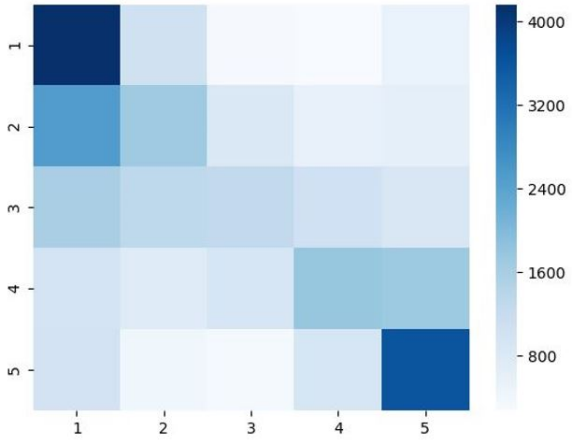

Figure 3. Sull'asse delle X sono rappresentati i valori previsti mentre sull'asse dell'Y i valori reali: si osserva come l'intensità dei colori delle celle diminuisce allontanandosi dalla diagonale

5.2. **SVM.** Tramite scikit-learn, in particolare SVC, viene implementato un algoritmo di Support Vector Machine. In questo caso il GridSearchCV viene applicato su due parametri: il parametro di regolazione C (di default 1) e il kernel gamma (di default 'scale'). Qui il GridSearch porta solo ad una modifica del kernel che risulta più performante se posta uguale a 'auto'. Quindi, il modello SVC con parametri C=1 e gamma='auto' viene applicato alle altre rappresentazioni ottenendo accuracy intorno al 46%, raggiungendo il 48% nel caso dello stemmed tfidf. Anche per questo metodo si vede come la classificazione avvenga più accuratamente per le classi estreme e in generale si vede che spesso le misclassificazioni avvengono nelle classi adiacenti a quella reale:

Class	***Precision***	***Recall***	***F-score***
1	0.57	0.61	0.59
2	0.39	0.36	0.37
3	0.40	0.38	0.39
4	0.43	0.41	0.42
5	0.59	0.66	0.62

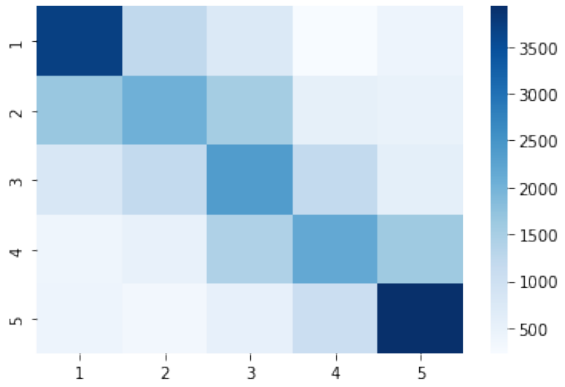

Figure 4. Confusion Matrix SVM

5.3. **k-NN.** Anche l'algoritmo di k-NN viene implementato tramite la libreria scikit-learn. Qui il numero di neighbor più adiacenti é l'unico parametro su cui si valutano valori diversi da quelli di default (k=5): si considerano tutti i valori tra 5 e 50, incrementando con un passo pari a 5. Il valore più alto considerato porta a migliori performance rispetto ai valori più bassi; tuttavia l'accuracy che si ottiene risulta essere più basso rispetto ai due modelli precedenti, essendo tra il 30% del lemmed Tdidf e il 34% del lemmed BoW. Le peggiori performance del k-NN sono ancora più rimarcabili considerando le 3 metriche numeriche e la matrice di confusione:

Class	***Precision***	***Recall***	***F-score***
1	0.34	0.65	0.45
2	0.28	0.17	0.21
3	0.29	0.13	0.18
4	0.32	0.18	0.23
5	0.38	0.56	0.45

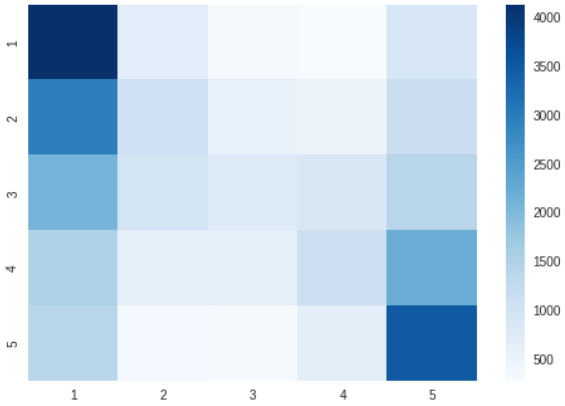

Figure 5. Confusion Matrix k-NN

Se si considera l'F-score si vede come si passa da valori tra il 54% e il 62% al 45% per le classi 1 e 5 e da valori tra 33% e il 42% a valori tra il 18% e il 23% per le rimanenti classi. Queste differenze sono visualmente visibili dalla matrice di confusione che, in questo caso, non presenta colori più intensi sulla diagonale o la sua prossimità bensì sulle due classi estreme.

6. Text Clustering

Oltre alla Text Classification è stato effettuato anche un Text Clustering utilizzando diversi approcci.

6.1. **k-Means.** Il dataset utilizzato è il lemmed BoW. Dopo aver standardizzato i dati la prima cosa fatta è stata analizzari per diversi valori di k (da 2 a 10) il valore dell'indice silhuette e la distorsione. I due valori di k trovati sono 3 e 5.

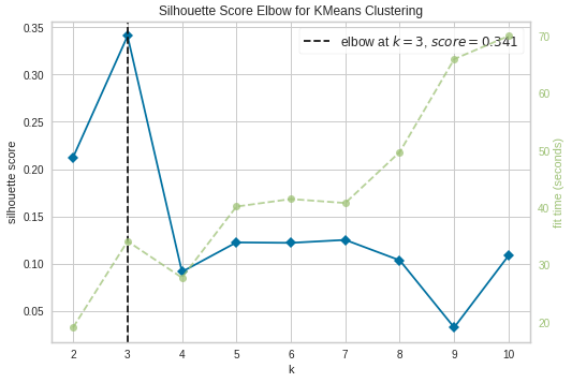

Figure 6. Valore della silhouette con diversi k.

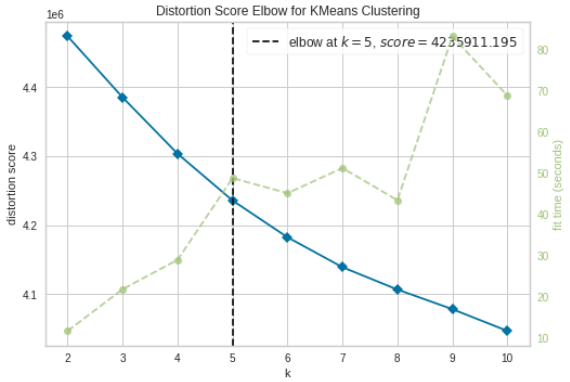

Figure 7. Valore della distorsione con diversi k.

I risultati sono stati rappresentati in due modi:

- Tramite una matrice che mette a confronto gli le etichette *Score* con i cluster previsti.

- Riducendo ulteriormente il dataset in due dimensioni tramite la tecnica t-SNE che riduce la dimensionalità mantenendo la distanza tra le osservazioni. Successivamente è stato possibile fare un grafico di dispersione colorando ogni osservazione con il rispettivo cluster previsto tramite l'algoritmo k-Means.

Come già notato dai valori della silhuette, le performance della clusterizzazione sono basse, questo è confermato anche dalle matrici seguenti e dai grafici di dispersione che non identificano nessun cluster particolarmente definito rispetto agli altri.

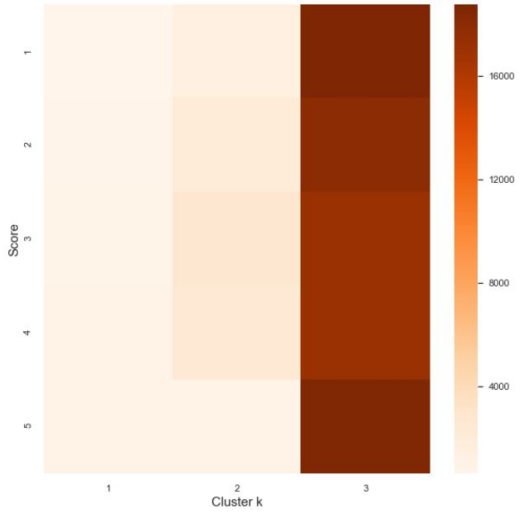

Figure 8. Matrice Score vs Cluster, k-Means k = 3.

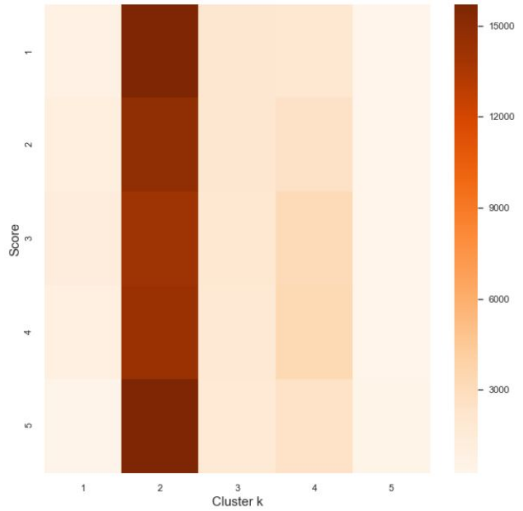

Figure 9. Matrice Score vs Cluster, k-Means k = 5.

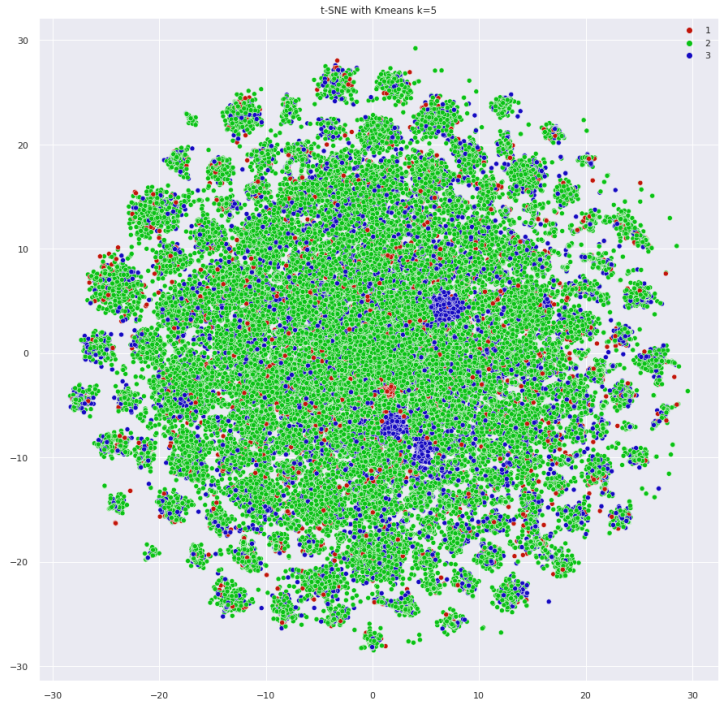

Figure 10. t-SNE su 3 cluster.

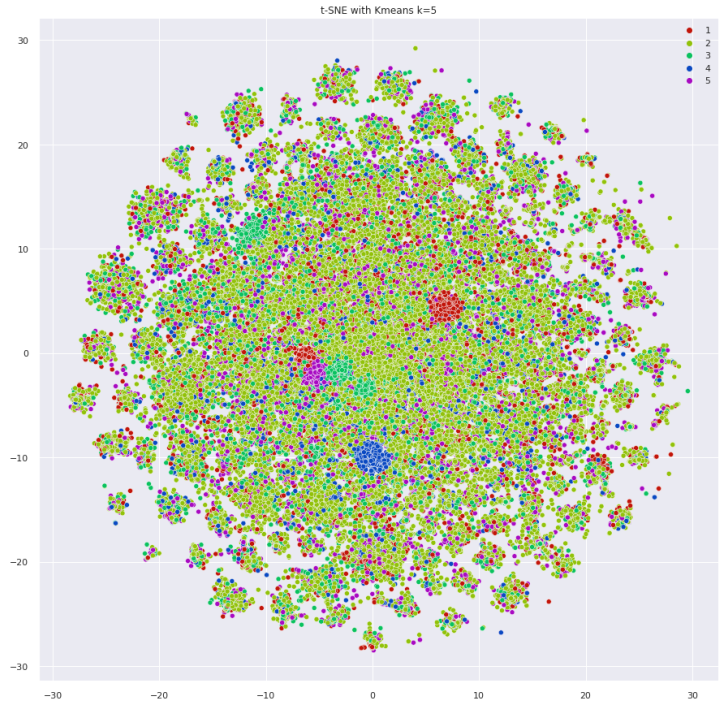

Figure 11. t-SNE su 5 cluster.

6.2. **Gerarchico Agglomerativo.** Un ulteriore metodo di clustering utilizzato è quello gerarchico agglomerativo. Sfortunatamente impostando un numero fisso di 5 cluster questo metodo accentua il fenomeno osservano col k-Means di associare quasi tutte le osservazioni ad un unico gruppo fino al punto di creare un unico gruppo con quasi tutte le osservazioni e i restanti 4 con una osservazione ciascuno. Sono stati provati diversi legami: singolo, completo e diverse metriche: euclidea, Manhattan, ma senza ottenere un miglioramento significativo delle performance. Per questo è stato scartato.

7. Conclusioni

Per quanto riguarda il task di Text Classification si osserva come nessun modello riesca a prevedere particolarmente bene lo Score delle recensioni, presentando però performance migliori con XGBoost e SVM. Tenendo conto di un trade-off tra performance e tempi computazionali si potrebbe affermare che l'XGBoost sia la tecnica preferibile. Anche l'utilizzo di rappresentazioni differenti non porta a risultati significativamente differenti. Più in generale si é visto anche come spesso le misclassificazioni spesso non si discostavano dalla classe reale: qualora le 5 classi fossero aggregate in 2 o 3 classi, la classificazione apportata da questi metodi sarebbe sicuramente migliore, presentando metriche più elevate. Il task di Text Clustering applicato su questo dataset non ha portato grande valore ad una maggior comprensione e interpretazione dei dati, infatti tutti gli approcci usati identificano un gruppo principale che contiene quasi tutti le osservazioni.

8. Ulteriori ricerche

Procedendo in una prospettiva di miglioramento dei risultati ottenuti, sarebbe utile esplorare da una parte nuovi metodi di classificazioni più performanti non solo in termini di risultati ma anche di tempi di esecuzione, e dall'altra lavorare più intensamente sulla messa a punto del tuning dei parametri dei modelli utilizzati.
Considerazioni analoghe possono essere fatte anche per quanto riguarda il clustering, esaminando algoritmi e approcci differenti.