UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA (LM DATA SCIENCE)
TEXT MINING AND INFORMATION RETRIEVAL

Riconoscimento titoli Clickbait: Tecniche di Clustering e Classificazione

Enrico Conte, Alessadro Fossati, Luca Pretini

Matricole: 852679, 819499, 864014

Sommario

L'informazione ha subito un netto cambiamento a causa della comunicazione digitale, si é infatti spostata prevalentemente online, portando le testate giornalistiche a mutare e ad aggiornare il loro linguaggio. In questo contesto nasce il "*clickbaiting*", fenomeno che porta gli autori a scegliere dei titoli accattivanti e spesso ingannevoli al solo fine di generare guadagni tramite "click", ovvero proponendo dei titoli che creino grandi aspettative e curiositá nei lettori con l'intenzione di portare gli utenti ad aprire l'articolo.

1 Introduzione

Il *clickbaiting*, neologismo inglese formato dai termini *"click"* e *"bait"* (lanciare l'esca), é un comportamento diffuso nell'informazione online. Nella odierna industria giornalistica, in cui il cartaceo é in declino e lascia sempre piú spazio alle controparti online, i guadagni derivanti da queste ultime sono sempre piú centrali e sostanziosi. Nell' industria digitale la fonte principale di introiti sono i guadagni pubblicitari, i quali sono proporzionali al numero di utenti che si riesce a portare all'interno della propria piattaforma.
In questo scenario nasce il *clickbaiting*, ovvero un insieme di pratiche volte ad ottenere il click dell'utente sul titolo dei propri articoli al fine di portarli sulla propria piattaforma, creando peró una discrepanza tra il contenuto dell'articolo e le aspettative date dal titolo. In questo progetto si andrá ad analizzare il testo di un insieme di titoli di diverse testate giornalistiche. Partendo da due dataset contenenti titoli di articoli clickbait e titoli non clickbait, si effettua una prima fase di preprocessing del testo. Successivamente si ricercano eventuali pattern ricorrenti mediante tecniche di clustering, per poi, nell'ultima sezione del report, presentare alcune operazioni di classificazione tramite differenti algoritmi.

2 Dataset e Pre-Processing

Il lavoro si basa su due dataset, contenenti titoli clickbait nel primo e titoli non clickbait nel secondo, ottenuti dal progetto del 2016, "*Stop Clickbait: Detecting and preventing clickbaits in online news media*"[1], volto allo sviluppo di una estensione per browser che riconosce questa tipologia di titoli e avverte l' utente in presenza di essi. I due dataset sono entrambi

composti da 16000 titoli di articoli, raccolti nel mese di settembre 2015. Il dataset clickbait é composto da titoli pubblicati dalle testate *'BuzzFeed', 'Upworthy', 'ViralNova', 'Thatscoop', 'Scoopwhoop' and 'ViralStories'*. I titoli clickbait sono stati riconosciuti e classficati manualmente da un gruppo di volontari. In questo dataset si riconoscono molte forme comuni nei titoli clickbait, come ad esempio "*20 things you didn't know about ...*" , o domande poste al lettore proponendo un format simile a quello dei quiz, "*can you guess ...*" "*which celebrity are you...*". Il dataset dei titoli non clickbait é invece composto da titoli ottenuti da *Wikinews*, una piattaforma che presenta sia standard rigorosi per la forma dei propri articoli, fornendo delle linee guida, sia un sistema di validazione degli articoli, che vengono verificati dalla community prima della pubblicazione.

Data l'origine giornalistica del corpus non necessita di operazioni di preprocessing che sarebbero necessarie in altri ambiti, come ad esempio tutte le operazioni relative a testo scritto da utenti nei social media, ma necessita comunque di alcuni accorgimenti.
In particolare, nel testo é presente la punteggiatura e si presenta con un uso del maiuscolo e minuscolo non regolare, molti titoli riportano una forma in cui tutte le iniziali sono in maiuscolo, in altri si usa una forma "standard", ovvero si hanno la prima lettera del titolo e i nomi propri con l'iniziale maiuscola.

Al fine di poter eseguire al meglio le analisi successive di clustering e classificazione, si effettuano diverse operazioni di preprocessing, in particolare lemmatizzazione del testo, processamento della punteggiatura e rimozione delle stopwords.
Per la lemmatizzazione si usa il lemmatizer fornito dalla libreria *nltk WordNetLemmatizer*, basato sul lexical database *WordNet*. In questa fase il testo viene prima tokenizzato, sempre tramite la libreria *nltk*, successivamente portato in forma minuscola, per poter applicare la lemmatizzazione.
Il risultato ottenuto, ovvero testo lemmatizzato e tokenizzato, viene quindi processato dal punto di vista della punteggiatura, in particolare vengono rimossi tutti i segni di punteggiatura presenti nel testo ad eccezione delle virgolette e dei trattini. Questa scelta é dovuta alla massiccia presenza di questi due elementi all'interno del dataset relativo agli articoli clickbait. Per quanto riguarda le virgolette, si puó notare nel dataset dei clickbait un uso massiccio di questo elemento, 4915 ricorrenze contro 450 del dataset non clickbait, caratteristica dovuta alla presenza frequente nei titoli clickbait di riferimenti a elementi pop, come ad esempio film o serie tv, ma anche alla maggior frequenza di neologismi.

Per quanto riguarda la scelta di mantenere i trattini, questa é dovuta alla volontà di non alterare la forma delle parole composte così come presenti nei titoli originali, poiché si é osservata una massiccia presenza di questa forma (circa 2600 elementi tra i due dataset) dovuta principalmente all'uso di neologismi e nomi propri presenti nei titoli. Essendo il testo giá tokenizzato, si eliminano i token contenenti punteggiatura (tranne gli elementi appena citati) mentre per quanto riguarda le parole composte, queste vengono riportate alla loro forma originale.
Dato che le analisi successive avverranno considerando i titoli come singoli corpus, questi verranno ricomposti, ovvero dai token processati si ricostruiranno delle stringhe contenenti il testo processato.
Per quanto riguarda le forme contratte (come "*you'll*" o "*won't*"), queste non sono state preprocessate, ovvero non hanno richiesto un preprocessing particolare, in quanto il lem-

matizer non ne ha alterato la forma, lasciando la forma abbreviata. É stato scelto di non introdurre modifiche, mantenendo la forma contratta laddove presente, dato che si é notata una maggiore frequenza di questa forma nel dataset dei titoli clickbait.

3 Clustering

3.1 Rappresentazione Vettoriale

La prima fase dell'attività sperimentale si è concentrata sulla ricerca di possibili clusters all'interno del dataset. Per poter calcolare delle misure di distanza tra documenti è necessario che essi vengano rappresentati in uno spazio vettoriale. A tal fine, si è deciso di utilizzare una funzione di peso $tf - idf$ (*term frequency-inverse document frequency*)[2] per calcolare la rilevanza delle parole all'interno del relativo titolo e rispetto allo loro frequenza all'interno degli altri titoli, così da penalizzare le parole molto comuni che appaiono di frequente. Formalmente, il punteggio $tf - idf$ per la $i - esima$ parola contenuta nel documento $j - esimo$ del corpus D è dato da:

$$tfidf(i, j, D) = tf(i, j)idf(i) \tag{3.1}$$

Dove:

$$tf(i, j) = \frac{n_{ij}}{d_j} \tag{3.2}$$

Con n_{ij} numero di occorrenze del termine i nel documento j e d_j numero totale di termini presenti nel documento $j - esimo$. E con:

$$idf(i) = log\left(\frac{|D|}{|d : i \in d|}\right) \tag{3.3}$$

dove D è il numero complessivo di documenti nel corpus e il denominatore rappresenta il numero di documenti che contengono il termine i. Il risultato di questa trasformazione è una matrice *TF-IDF* di dimensione 32000×18677.

3.2 Riduzione della Dimensionalità

Tuttavia, al fine di permettere una elaborazione efficiente dei dati, è necessario ridurre la sparsità della matrie ottenuta.

A tal proposito, si è deciso di adoperare una *decomposizione a valori singoli*[3][4] della matrice, troncata per un numero di componenti tale da permettere di spiegare una soglia ragionevole di varianza complessiva. Dopo aver sperimentato quale fosse il numero di componenti adeguato, la scelta è ricaduta su 3100, un valore che ci permette di ottenere un'approssimazione della matrice originale capace di cogliere il 75% della varianza complessiva.

La *decomposizione a valori singoli* (SVD) è una tecnica di fattorizzazione per matrici rettangolari reali e complesse basata sull'uso di *autovalori* e *autovettori*. La SVD può essere

utilizzata in forma completa o in forma troncata, in questa sede ci si concentra sulla variante troncata.

Sia A una matrice $m \times n$ di frequenza del tipo termine-documento di rango r con $r \leq n$. Ipotizziamo che $m \geq n$ così che ci sono più termini che documenti. La decomposizione a valori singoli per la matrice A si calcola come:

$$A = U\Sigma V^T \tag{3.4}$$

Dove U e una matrice ortogonale $m \times r$ le cui colonne sono i *vettori singolari di sinistra*, Σ è una matrice diagonale $r \times r$ i cui elementi nella diagonale maggiore sono i *valori singolari* della matrice A e V è una matrice ortogonale $r \times n$ le cui colonne sono i *vettori singolari di destra*. Inoltre, mentre la matrice Σ è unicamente determinata, le matrici U e V possono essere scelte in diversi modi. La Figura 3.1 fornisce una rappresentazione schematica di quanto esposto.

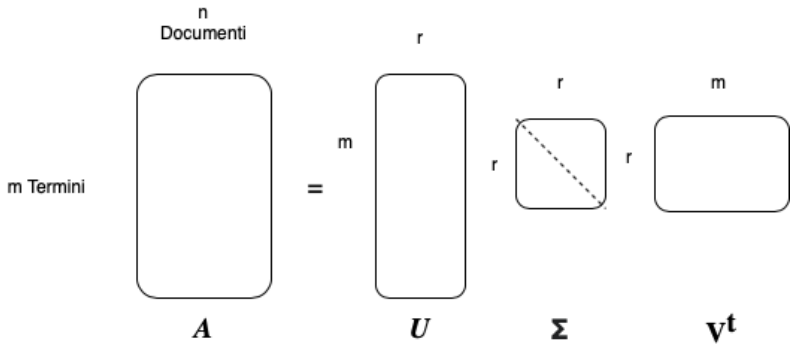

Figura 3.1: Esempio di Decoposizione a Valori Singoli

I valori singolari sono tutti uguali o maggiori a 0 e sono ordinati, convenzionalmente, in modo decrescente, cosicché il valore singolare più grande si trovi in alto a sinistra nella matrice Σ. Nella sua versione troncata, tutti i valori singolari sono maggiori di 0.

L' utilità di questa tecnica risiede nella sua capacità di ottenere, data una matrice A, la sua miglior approssimazione $k - dimensionale$ con $k \leq r$, dove r è il rango della matrice A. Questa proprietà può essere descritta in termini più intuitivi se si pensa alla SVD come alla somma si matrici di rango 1.

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{3.5}$$

La Matrice A può successivamente essere approssimata scegliendo un qualunque $k \leq r$. Il processo di approssimazione genera una matrice di rango k che è la miglior approssimazione di rango k della matrice originale in termini di fit dei minimi quadrati.

$$A \approx A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T = U_k \Sigma_k V_k^t \tag{3.6}$$

La Figura 3.2 mostra questa approssimazione in termini generici per un valore di k molto più piccolo del rango r.

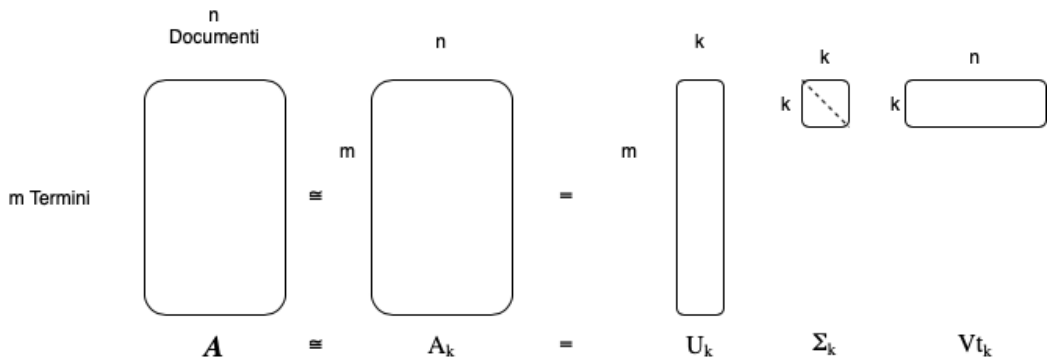

Figura 3.2: Esempio di approssimazione della matrice A tramite *Decomposizione dei Valori Singoli*

L'equazione 3.5 implica che si può ottenere un'approssimazione di A dal prodotto del primo valore singolare e la matrice formata dal prodotto esterno della prima colonna della matrice U per la prima colonna della matrice V, ovvero gli *autovettori*.

Il risultato sarà una matrice di rango 1 che è la miglior approssimazione di rango 1 della matrice A originaria. Per ottenere approssimazioni più fedeli basta ripetere il processo per k valori singolari e aggregare i risultati, ottenendo, quindi, la miglior approssimazione di rango k della matrice A di partenza.

Come accennato sopra, questa tecnica ci permette di ridurre la dimensionalità della matrice di rappresentazione da 18.677 a 3100 senza perdere una quantità eccessiva di informazioni.

3.3 Scelta del K ideale

L'algoritmo di clustering testato è il *KMeans*[5]. La prima decisione da prendere in fase di implementazione del clustering è il numero ottimale di clusters da ricercare. Dopo aver fatto ricerche sulle euristiche utilizzate per guidare questa delicata scelta, si è deciso di testare diversi valori (9, 16, 20, 32) e comparare la bontà dei clusters ottenuti secondo l'indice di *Calinski-Harabasz*[6], definito come il rapporto tra la dispersione all'interno dei cluster e la dispersione tra i clusters.

Alla fine della sperimentazione, la scelta è ricaduta su 9 come numero ideale di clusters da cercare all'interno dei dati, dal momento che per i valori scelti, l'indice CH risulta massimizzato.

CH_9	CH_16	CH_20	CH_32
78.19	49.58	57.51	45.88

Tabella 3.1: Risultati *Calinski-Harabasz*

3.4 Risultati Clustering

I clusters e i relativi centroidi risultanti sono visibili nella Figura 3.3 rappresentati rispetto alle prime due componenti principali.

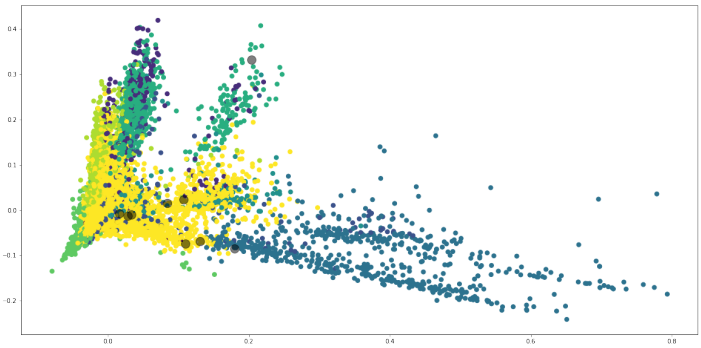

Figura 3.3: Scatterplot dei Clusters con relativi Centroidi rispetto alle prime due *componenti principali*

L'indice di Silhouette[7] per i clusters ottenuti è pari a 0.0099, ad indicare la presenza di cluster non ben definiti e parzialmente "overlapping". Tuttavia, l'esplorazione semantica dei clusters ha prodotto risultati interessanti, in quanto si delineano con chiarezza delle tematiche ben distinte le une dalle altre. Al fine di dare una rappresentazione visiva quanto più accattivante e comprensibile, vengono realizzate delle WordCloud per ognuno dei 9 clusters trovati. Successivamente, si ordinano i clusters per numerosità (il cluster in alto a sinistra è quello più numeroso con 26.873 elementi, mentre il meno numeroso in basso a destra conta 283 elementi) e si riuniscono tutte le WordCloud in un unico "mosaico" (Figura 3.4), così da fornire un'immagine globale del contenuto sintattico e semantico dei dati.

Figura 3.4: WordCloud dei Clusters ottenuti

Inoltre, in Figura 3.5 è possibile vedere la composizione dei clusters rispetto alle categorie di partenza (clickbait vs no clickbait)

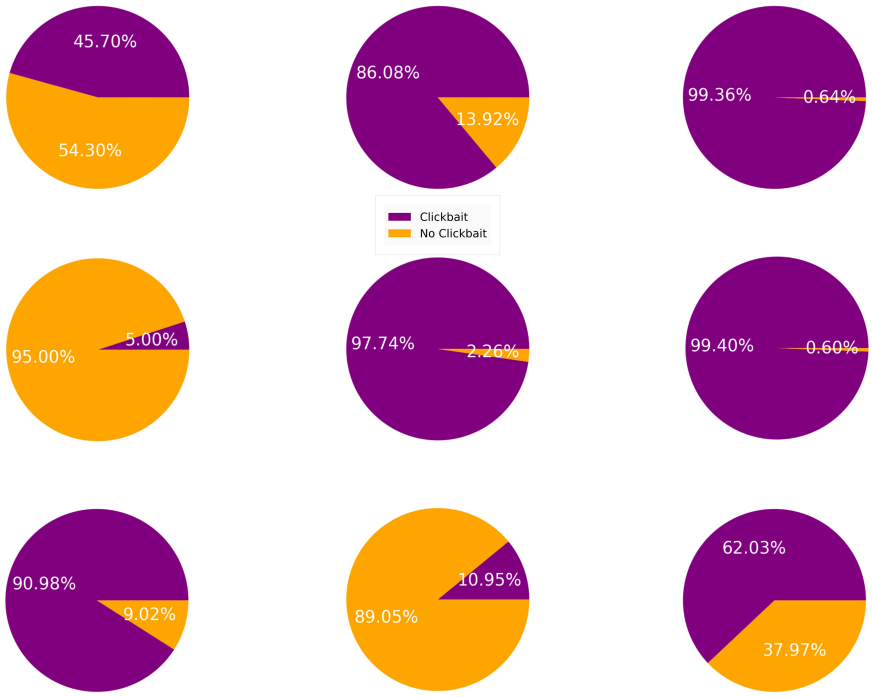

Figura 3.5: Composizione percentuale dei Clusters tra titoli clickbait e non clickbait

Come è possibile vedere dalle figure 3.4 e 3.5, il cluster più numeroso è composto in proporzioni simili da titoli clickbait e no_clickbait contenenti termini con un campo semantico piuttosto ampio, e non ascrivibili ad un "topic" in particolare.

D'altro conto, se consideriamo, invece, i clusters via via più piccoli, si può notare una maggiore polarizzazione sia in termini di classe originale di appartenenza, sia in termini di contenuto semantico. Si considerino, ad esempio, il cluster "kill" e il cluster "zodiac". Entrambi presentano un buon livello di coerenza del campo semantico dei termini più frequenti e sono composti per la quasi totalità da titoli riconducibili a solo una delle due classi(clickbait vs no_clickbait) di partenza.

Un caso curioso è rappresentato invece dal cluster "thing", che, se a prima vista può sembrare composto da parole di uso generico, ad una più attenta analisi si rivela essere composto da quei termini ampiamente diffusi nella "sintassi del clickbait", basti pensare a tutti quei titoli del tipo "10 things people need to know about....".

4 Classificazione

4.1 Rappresentazione vettoriale e riduzione della dimensionalitá

Per la fase di Text Classification ci si pone come obiettivo quello di classificare i titoli degli articoli come appartenenti alla classe clickbait oppure a quella non clickbait.
Per permettere questa operazione di classificazione binaria di tipo testuale risulta necessario vettorizzare i titoli degli articoli, come effettuato per la fase di clustering. Una scelta adatta a questo scopo è quella di far coincidere il set di features che si vogliono ottenere tramite

vettorizzazione con il set di parole contenute all'interno dei differenti titoli, secondo il modello ad unigrammi, conosciuto come *bag-of-word*. Tuttavia, anche in questo caso, adottando questa strategia si ottiene una matrice estremamente sparsa ed estremamente grande in termini dimensionali (32000x18677), dove sulle righe si trovano i titoli degli articoli e sulle colonne della matrice gli unigrammi.
Si decide dunque di effettuare un'operazione di *feature synthesis* tramite la decomposizione in valori singolari della matrice, selezionando le prime *k features* che permettono di superare una soglia di varianza spiegata pari al 75%.
Si riduce dunque la dimensione della matrice, conservando solamente 2000 delle 18677 features presenti inizialmente, rendendo di fatto le operazioni più semplici e veloci in termini di computazione.

4.2 Modelli e misure di performance

All'interno dell'analisi vengono utilizzati algoritmi di differente tipologia, al fine di valutare quale possa adattarsi meglio ai dati considerati, passando attraverso la valutazione di alcune misure di performance che verranno presentate in seguito. Ognuno di questi algoritmi di apprendimento supervisionato viene utilizzato per allenare un classificatore. Per la categoria dei modelli di separazione si seleziona *Support Vector Machine (SVM)*[8]. Per quanto concerne i modelli probabilistici si seleziona *Gaussian Naive Bayes (GaussianNB)*[9]. Infine, tra i modelli di regressione si seleziona *Logistic Regression*[10]. La libreria Python utilizzata per computare algoritmi e misure di performance è *sklearn*[11]. Per quanto riguarda le misure finalizzate alla valutazione della performance dei modelli di classificazione, si decide di utilizzare:

- Accuratezza:

$$Accuracy = \frac{TP + FN}{TP + TN + FP + FN}$$

  Indice che permette di calcolare la percentuale di istanze correttamente classificate (True Positive TP e True Negative TN). Un valore vicino a 1 di accuracy indica che il classificatore utilizzato si adatta in maniera buona ai dati, tuttavia, una stima puntuale dell'accuracy potrebbe non essere sufficiente per decretare se un classificatore possa essere ritenuto performante o meno.

- Recall:

$$Recall = \frac{TP}{TP + FN}$$

  Indice che permette di ottenere la percentuale di record positivi correttamente classificati. Come evidenziato in alcuni esempi successivi, spesso la Recall da sola non risulta essere un buon indicatore del fatto che un modello sia performante.

- Precision:

$$Precision = \frac{TP}{TP + FP}$$

Indice che permette di trovare la frazione di record effettivamente positivi tra tutti quelli che vengono classificati come positivi. Un valore vicino a 1 di questo indice comporta che ci sia un numero basso di falsi positivi (FP). Anche questa misura, se considerata da sola, potrebbe risultare fuorviante sulla valutazione della performance di un modello di classificazione.

- F1_measure:

$$F_1measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

È la media armonica tra Recall e Precision, è un indice il cui eventuale valore elevato garantirebbe valori elevati sia di Recall che di Precision. È dunque una misura di sintesi dei due indici precedentemente presentati.

- ROC Curve:
  Curva che presenta la percentuale di veri positivi sull'asse delle ordinate (%TP) e la percentuale dei falsi positivi sull'asse delle ascisse (%FP). Questo grafico permette di calcolare un indice per valutare la performance di un modello, denominato AUC (*area under curve*), che fornisce indicazioni riguardo all'area sottostante alla curva di ROC. Maggiore è l'AUC, maggiore è la probabilità di avere un modello performante.

4.3 Risultati della classificazione e confronti

Al fine di effettuare una classificazione accurata si decide di utilizzare come test set un numero di titoli corrispondenti al 20% dell'intero set, utilizzando il restante 80% delle righe della matrice di vettorizzazione per testare la validità della classificazione.
Per ottenere le ROC curve relative ad ognuno dei classificatori si decide di partizionare la matrice in 5 parti composte dallo stesso numero di righe, secondo la tecnica del *5-Fold Cross Stratified Validation*, la quale, mediante campionamento stratificato permette di utilizzare come test set ognuna delle 5 partizioni a turno, utilizzando le restanti 4 come training set, garantendo una equa rappresentazione delle due classi all'interno del test set.
Di conseguenza, per ognuno degli algoritmi si riportano i 5 valori di AUC ottenuti in seguito ad ogni iterazione della 5-Fold SCV oltre ad un valore medio di AUC accompagnato dalla relativa deviazione standard. La classe positiva risulta essere non-clickbait.
Vengono di seguito presentate le curve di ROC relative ad ognuno dei 3 modelli utilizzati per la classificazione:

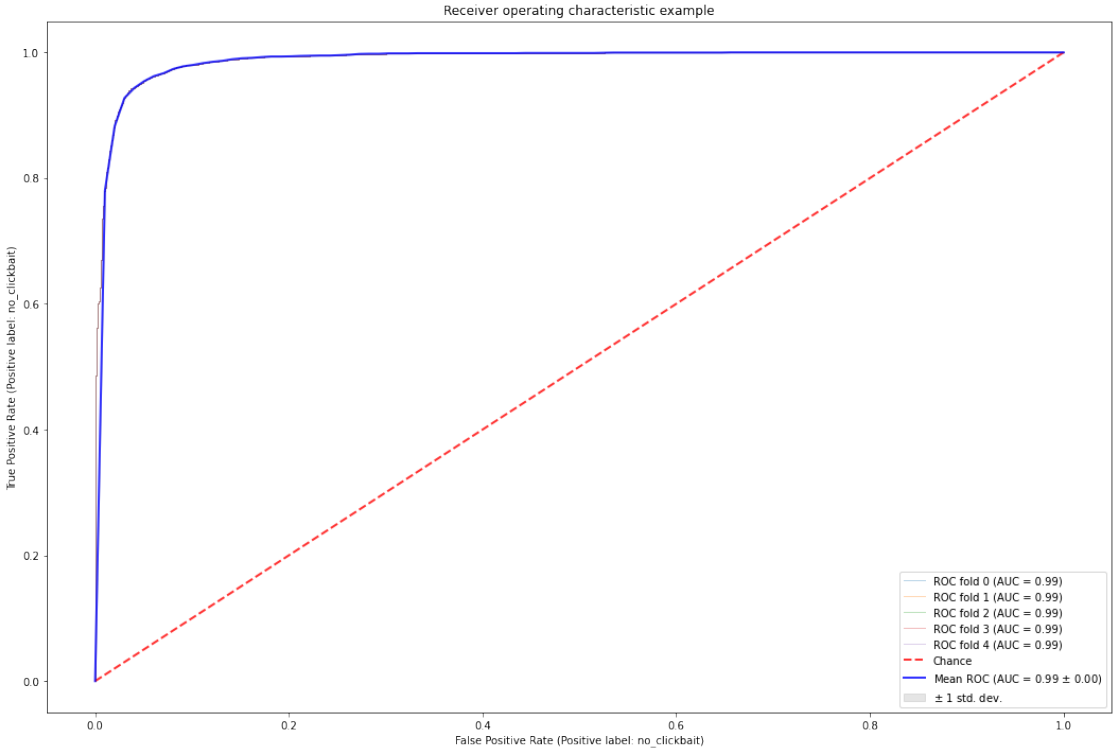

Figura 4.1: Support Vector Machine

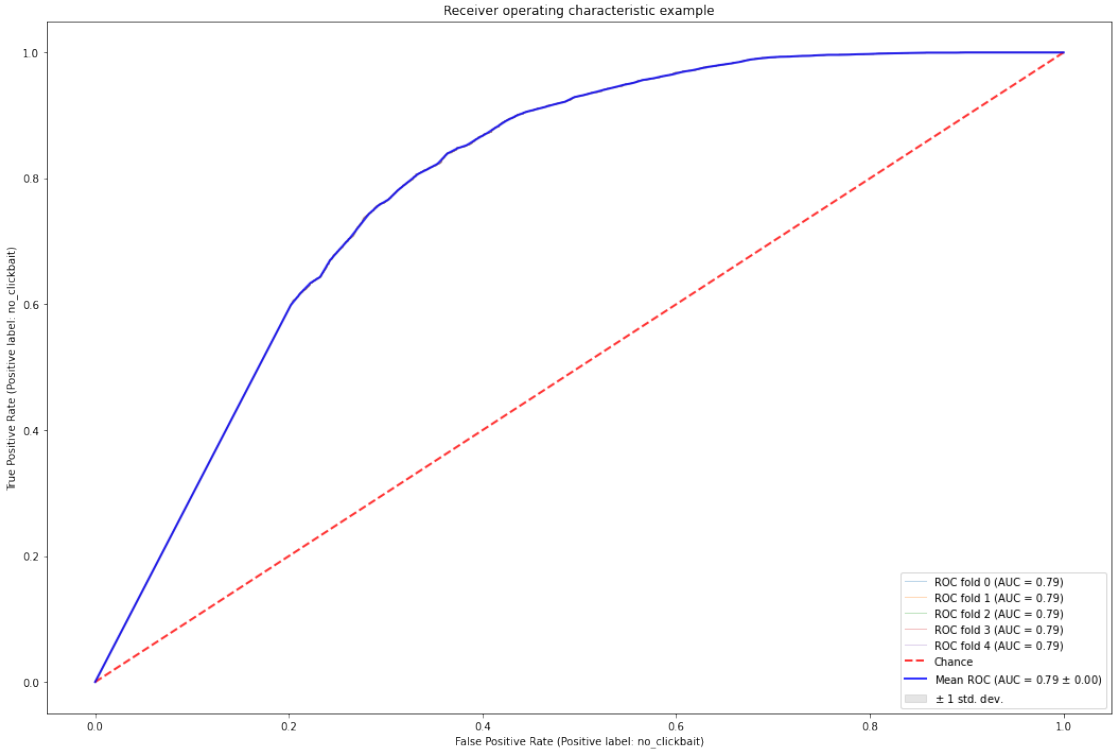

Figura 4.2: Gaussian Naive Bayes

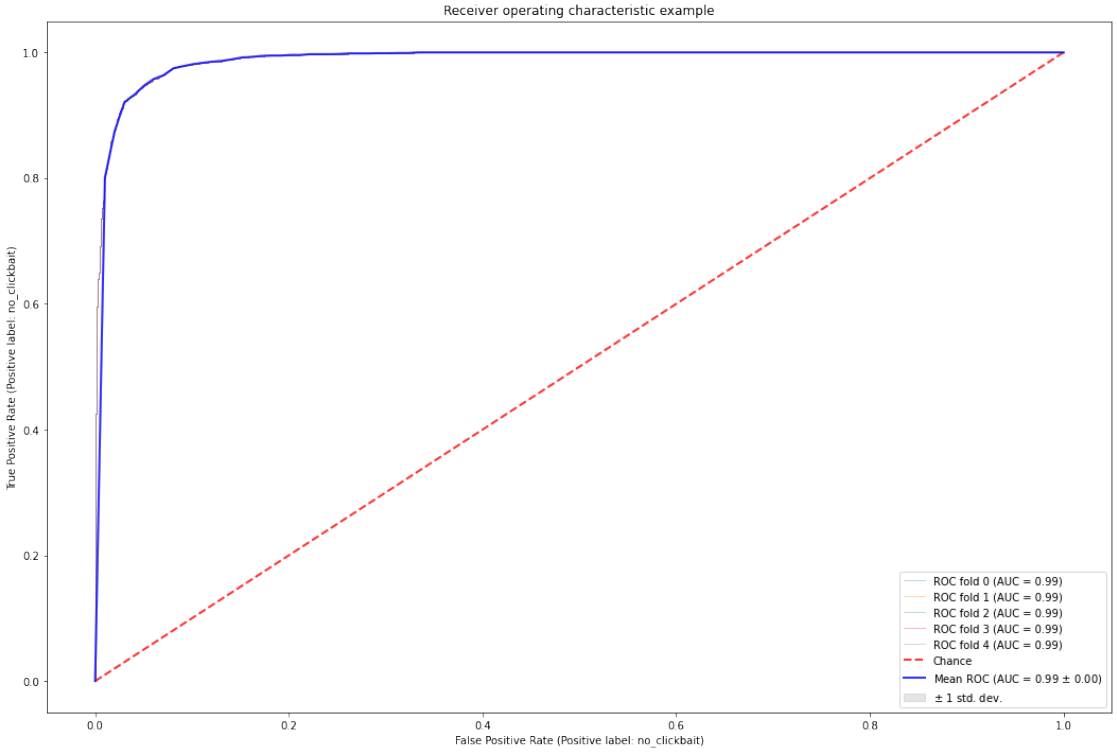

Figura 4.3: Logistic Regression

Analizzando le curve di ROC sopra riportate si evince che tutti e 3 gli algoritmi funzionano in maniera abbastanza soddisfacente ad ogni iterazione di 5Fold-SVC. Si evince un' ottima precisione nel risultato, dal momento che i risultati di AUC ottenuti ad ogni iterazione sono approssimativamente gli stessi, con deviazione standard circa pari a zero.
Tuttavia, GaussianNB, a parità di percentuale di True Positive (%TP) con gli altri due modelli restituisce in media percentuali più alte di False Positive (%FP), risultando dunque meno performante degli altri due basandosi sulla misura di AUC. SVM e LogisticRegression sembrano essere modelli molto performanti invece.
Si riportano inoltre le *confusion matrix* relative ad ognuna delle 3 classificazioni (Figura 4.4). All'interno di queste matrici 2x2 viene riportata sulla diagonale principale la quantità di classificazioni effettuate correttamente (TP e TN), mentre sulla diagonale secondaria viene riportata la quantità di classificazioni errate (FP e FN):

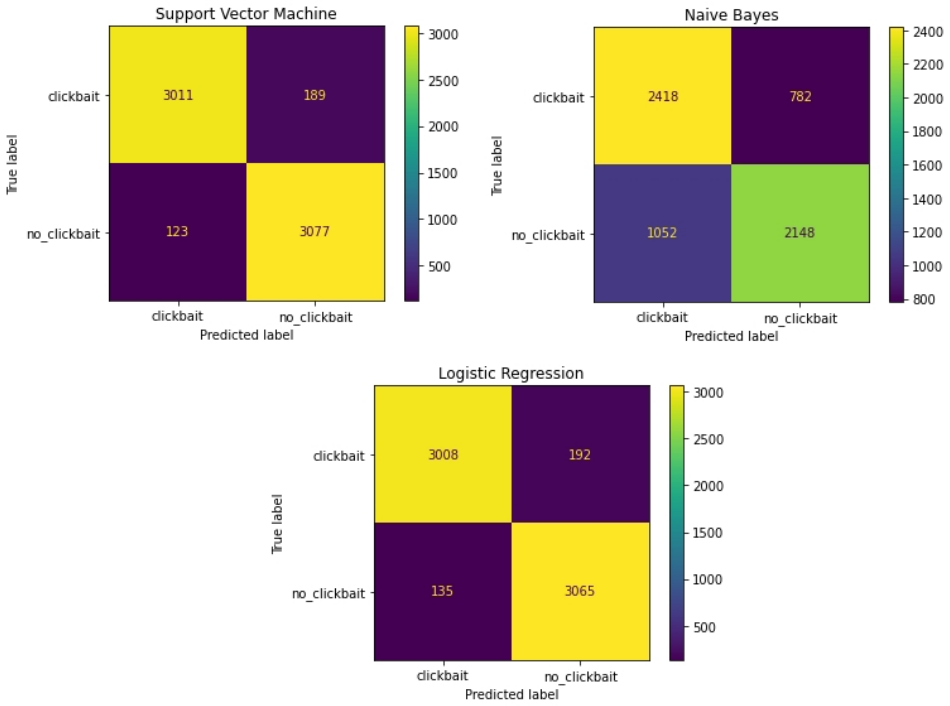

Figura 4.4: Matrici di confusione

Anche dalle confusion matrix si può notare come i modelli più performanti sembrano essere SVM e LogisticRegression, dal momento che presentano un maggior numero di istanze classificate correttamente rispetto a GaussianNB. Tuttavia, per un confronto più specifico e basato su più misure, si presenta una serie di tabelle riassuntive dei valori ottenuti in relazione alle misure di performance. La prima tabella (Tabella 4.1) si riferisce alle misure di performance inerenti alla classe clickbait:

	Precision	Recall	F1-measure	Accuracy
SVM	0.96	0.94	0.95	0.95
GaussianNB	0.70	0.76	0.73	0.71
LogisticReg	0.96	0.94	0.95	0.95

Tabella 4.1: Risultati per classe clickbait

Si presenta una tabella di valori anche per la classe no-clickbait (Tabella 4.2):

	Precision	Recall	F1-measure	Accuracy
SVM	0.94	0.96	0.95	0.95
GaussianNB	0.73	0.67	0.70	0.71
LogisticReg	0.94	0.96	0.95	0.95

Tabella 4.2: Risultati per classe no-clickbait

Dalle Tabelle 4.1 e 4.2 si evince che i modelli utilizzati non presentano risultati significativamente differenti passando dalla classe clickbait alla classe no-clickbait, risultando

performanti approssimativamente alla stessa maniera su entrambe le classi. Una volta di più si evince la minore efficienza dell'algoritmo probabilistico Gaussian Naive Bayes nella classificazione.

Si propone infine un confronto finale basato sulla media delle misure di performance per le due classi (Tabella 4.3):

	Precision	Recall	F1-measure	Accuracy
SVM	0.94	0.95	0.95	0.95
GaussianNB	0.72	0.71	0.72	0.71
LogisticReg	0.95	0.94	0.95	0.95

Tabella 4.3: Risultati medi

Vengono confermati dunque i risultati ottenuti precedentemente. In conclusione, si può affermare che il modello probabilistico della famiglia Naive Bayes, applicato ai dati, risulta essere meno performante rispetto al modello di separazione Support Vector Machine ed al modello di regressione Logistic Regression. Inoltre, dalle misure di performance utilizzate per valutare gli algoritmi non si riesce a valutare quale sia il migliore tra SVM e Logistic. Tuttavia, in fase di sviluppo si è andati incontro a tempi computazionali decisamente più elevati con SVM rispetto al modello di regressione, per cui dovendo selezionare uno dei due modelli, si opterebbe per Logistic Regression.

5 Conclusioni

In conclusione, per quanto riguarda la task di Clustering, si può affermare che, nonostante gli indici di valutazione segnalino la presenza di clusters non chiaramente definiti e overlapping, analizzando il contenuto degli elementi dei 9 clusters, emerge la presenza sia di "topic" afferenti alle singole classi originarie (clickbait vs no_clickbait) sia di "topic" comuni ad entrambe.
Per quanto riguarda invece le analisi relative alla classificazione, si seleziona come algoritmo migliore il modello di regressione Logistic Regression, anche in virtù dei minori tempi di computazione richiesti per allenare il classificatore. L'algoritmo di separazione Support Vector Machine presenta misure di performance molto simili a quelle del modello di regressione logistica, presentando tuttavia tempi di computazione decisamente più elevanti in fase di training. Risulta invece significativamente meno performante il modello probabilistico Gaussian Naive Bayes.

Riferimenti bibliografici

[1] Abhijnan Chakraborty et al. «Stop Clickbait: Detecting and preventing clickbaits in online news media». In: *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*. IEEE. 2016, pp. 9–16.

[2] Daniel Jurafsky e James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Vol. 2. Feb. 2008.

[3] G. H. Golub e C. Reinsch. «Singular Value Decomposition and Least Squares Solutions». In: *Numer. Math.* 14.5 (apr. 1970), pp. 403–420. ISSN: 0029-599X. DOI: 10.1007/BF02163027.

[4] Rocha L.M Wall M.E. Rechtsteiner A. «Singular Value Decomposition and Principal Component Analysis». In: (2003). ISSN: 0029-599X. DOI: https://doi.org/10.1007/0-306-47815-3_5.

[5] Aristidis Likas, Nikos Vlassis e Jakob J. Verbeek. «The global k-means clustering algorithm». In: *Pattern Recognition* 36.2 (2003). Biometrics, pp. 451–461. ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(02)00060-2.

[6] U. Maulik e S. Bandyopadhyay. «Performance evaluation of some clustering algorithms and validity indices». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.12 (2002), pp. 1650–1654. DOI: 10.1109/TPAMI.2002.1114856.

[7] Xu Wang e Yusheng Xu. «An improved index for clustering validation based on Silhouette index and Calinski-Harabasz index». In: *IOP Conference Series: Materials Science and Engineering* 569 (ago. 2019), p. 052024. DOI: 10.1088/1757-899x/569/5/052024.

[8] Corinna Cortes e Vladimir Vapnik. «Support-vector networks». In: *Machine learning* 20.3 (1995), pp. 273–297.

[9] Shuo Xu. «Bayesian Naïve Bayes classifiers to text classification». In: *Journal of Information Science* 44.1 (2018), pp. 48–59. DOI: 10.1177/0165551516677946.

[10] Georgiana Ifrim, Gökhan Bakir e Gerhard Weikum. «Fast logistic regression for text categorization with variable-length n-grams». In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, pp. 354–362.

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