# Progetto di Machine Learning e Sistemi Intelligenti per Internet

# Sentimental Analysis

Giulia Gaglione e Sara Marrapesa

A.A. 2024/2025

## 1 Introduzione

Link del git: `https://github.com/saramarrapesa/progetto-ml-e-sii.git`

L'analisi del sentiment delle recensioni è uno dei compiti più comuni nella classificazione dei testi. Le opinioni degli utenti, sia online che offline, riguardo a un prodotto, rappresentano una piattaforma ideale per raccogliere un grande volume di dati destinati all'analisi del sentiment.
Pertanto, le recensioni complessive degli utenti su un prodotto sono il compito principale per l'analisi del sentiment, che può essere suddiviso in due categorie: positiva e negativa.

### 1.1 Cosa è un Sentimental Analysis?

Con *Sentimental Analysis* si definisce un processo che estrae informazioni su un'entità e identifica automaticamente le eventuali soggettività di quella entità. L'obiettivo è determinare se il testo generato dagli utenti esprime opinioni positive, negative o neutrali.
Per individuare le informazioni soggettive all'interno del testo vengono utilizzate delle tecniche di Natural Language Processing (NLP) e Text Analysis, che consentono ai computer di apprendere, leggere e regolare il linguaggio umano.

## 2 Approccio utilizzato

Attualmente, esistono tre approcci per affrontare il problema dell'analisi del sentiment: tecniche basate su lessico, tecniche basate su machine learning e approcci ibridi.
In questo progetto sono state impiegate tecniche di machine learning, focalizzandosi in particolare su modelli tradizionali come il classificatore *Naive Bayes* e la *Regressione Logistica*, con l'obiettivo di confrontarne le prestazioni.

## 3 Obiettivo

L'obiettivo del progetto è addestrare un modello sui dati al fine di identificare il sentiment nascosto nelle recensioni, classificandolo come positivo, negativo o neutrale.
Successivamente, verranno analizzate le prestazioni del modello utilizzando parametri come l'accuratezza e la matrice di confusione.

## 4 Dataset

Il dataset utilizzato è presente al link: `https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews`.
Il dataset contiene circa 500.000 recensioni scritte nell'arco di 10 anni, fino all'ottobre del 2012.
Le recensioni riguardano cibi raffinati di Amazon, includendo informazioni su prodotti e utenti, valutazioni e una recensione in testo normale.
Nel dettaglio si hanno:

- Recensioni da ottobre 1999 a ottobre 2012;
- 568.454 recensioni;
- 256.059 utenti;
- 74.258 prodotti;
- 260 utenti con più di 50 recensioni.

La struttura del dataset è così gestita:

- *ProductId*: identificatore univoco del prodotto;

- *UserId*: identificatore univoco dell'utente;
- *ProfileName*: nome del cliente;
- *HelpfulnessNumerator*: numero di utenti che hanno trovato utile la recensione;
- *HelpfulnessDenominator*: numero di utenti che hanno indicato se la recensione è stata utile o meno;
- *Score*: valutazione compresa tra 1 e 5;
- *Time*: timestamp della recensione;
- *Summary*: breve riassunto della recensione;
- *Text*: testo della recensione.

# 5 Struttura del codice

## 5.1 Import delle librerie

Il primo passo per eseguire il progetto consiste nell'importazione delle librerie necessarie.
In primo luogo, *Keras* fornisce gli strumenti per costruire e addestrare modelli di deep learning. Nel contesto del machine learning tradizionale, strumenti come la *LogisticRegression* e *MultinomialNB* , di *scikit-learn*, vengono usati per classificare dati.
Per la pre-elaborazione del testo, *Stopword* rimuove le parole comuni che non aggiungono valore all'analisi, e *Wordcloud* permette di visualizzare in modo interessante le parole più frequenti in un corpus.
Infine per la valutazione, *accuracy score* e *confusion matrix* permettono di misurare la precisione del modello e analizzare gli errori in dettaglio.

## 5.2 Preparazione dei dati

Viene poi caricato il file .csv contenente il dataset da utilizzare.

## 5.3 Data cleaning

La Sentiment Analysis richiede che i dati di addestramento del testo vengano puliti prima di essere utilizzati per costruire il modello di classificazione.
Si procede rimuovendo le righe con i valori mancanti, le righe duplicate e le righe con valori incoerenti di "Helpfulness", ossia le righe il cui valore di HelpfulnessNumerator è maggiore del valore di HelpfulnessDenominator.
Si procede poi con la creazione di una nuova colonna *Target* che viene inizializzata con l'output della funzione *create_target*. Questa funzione prende in input il campo Score e in base al suo valore classifica il record in *Positive*, se maggiore di 3, *Neutral*, se uguale a 3, e *Negative*, se minore di 3.

## 5.4 Gestione dello squilibrio delle classi

Successivamente, si affronta il problema dello sbilanciamento del dataset che si ha quando questo ha una distribuzione delle etichette sbilanciata.
Dopo aver visualizzato la distribuzione delle etichette prima del bilanciamento, visualizzata tramite un grafico a barre orizzontali, come riportato nella Figura 1, si prosegue con il *downsampling*.

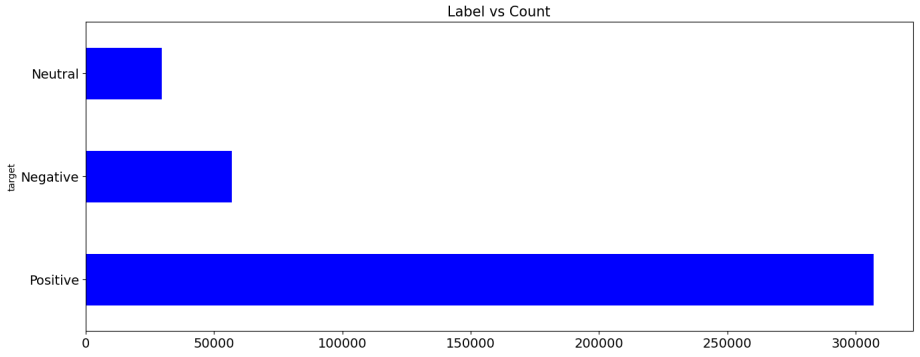


Figure 1: Grafico a barre orizzontali che riporta lo sbilanciamento delle etichette.

Il downsampling è una tecnica utilizzata per ridurre la quantità di dati in un dataset. In generale, consiste nel ridurre la risoluzione o la frequenza dei dati, mantenendo solo una parte rappresentativa o significativa di essi.
Nel progetto in analisi, vengono selezionati solo 50.000 campioni per le classi Positive e Negative, ottenendo così la seguente distribuzione, illustrata della

Figura 2.

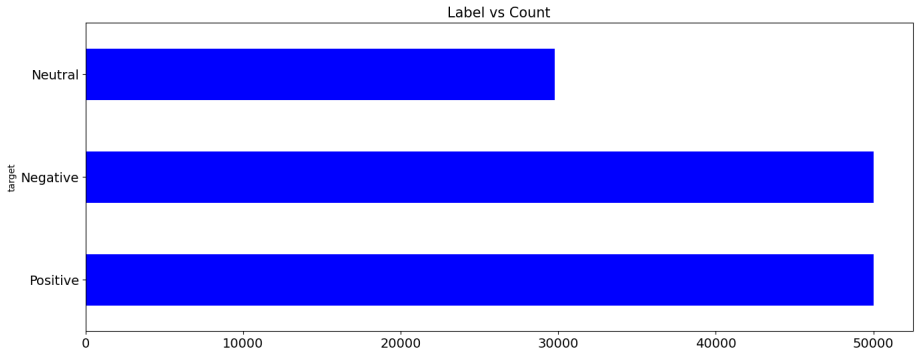


Figure 2: Grafico a barre orizzontali che riporta il numero di campioni post downsampling.

## 5.5 Data pre-processing

Il pre-processing è un passaggio molto importante per la classificazione del testo. Non è possibile inserire una sequenza di simboli in questi algoritmi, poiché dovrebbero avere valori numerici e non una sequenza di simboli con lunghezza variabile.
Si inizia il processo eliminando dal testo tutte le stopwords del linguaggio delle recensioni, ossia l'inglese. Tra le stopwords troviamo parole di uso comune, come "the", "and", ma che hanno poco significato.
Bisogna notare che tra le stopwords da eliminare vengono però preservate, per ottenere un'analisi dei sentimenti corretta, le parole con negazioni, come "no", "n't".
Oltre alla rimozione delle parole comuni di poco peso per l'analisi, si procede a eliminare anche hashtag, punteggiatura, numeri e a convertire tutte le lettere in minuscolo.
Dopo aver eliminato dal testo le parole non importanti per l'obiettivo del progetto in questone, si prosegue con la manipolazione del testo andando a ridurre alla radice tutte le parole tramite lo stemming, eliminando così eventuali suffissi e desinenze: ad esempio la parola "studies" diventa "study".

## 5.6 Word cloud

In questa sezione, si generano delle word cloud, ossia una rappresentazione visiva delle parole più frequenti in un testo dove le parole più usate appaiono più grandi, mentre quelle meno usate appaiono più piccole.
Di seguito si possono osservare le word cloud per le recensioni con target positivo, negativo e neutrale.

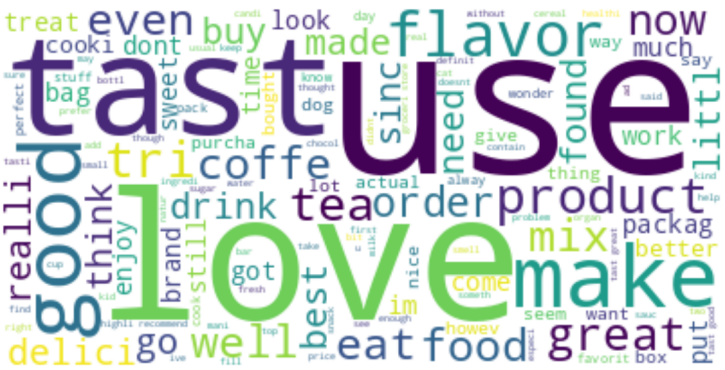


Figure 3: Word cloud delle recensioni positive.

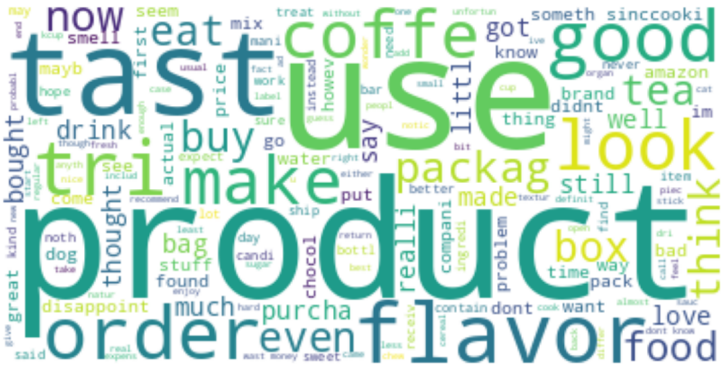


Figure 4: Word cloud delle recensioni negative.


Figure 5: Word cloud delle recensioni neutrali.

## 5.7 Train and test split

Dopo aver pulito il dataset, si può passare all'addestramento del modello.
Il codice divide il dataset in due parti:

- Training set (80%) per addestrare il modello;
- Test set (20%) per valutarne le prestazioni.

## 5.8 Vectorization

Il lavoro prosegue trasformando il testo in un formato adatto per la lavorazione tramite i modelli di machine learning.
Vengono utilizzati la libreria *CountVectorizer*, per applicare la tecnica di *Bag of Words* (BoW) che converte il testo in una matrice di conteggi di parole, e la libreria *TfidfVectorizer*, che converte il testo in una matrice TF-IDF (Term Frequency - Inverse Document Frequency), e valuta l'importanza di una parola in un documento rispetto a un insieme di documenti.

## 5.9 Model training

Giusti a questo punto, si inizia ad addestrare e valutare i modelli utilizzati.
Vengono addestrati e valutati due modelli di classificazione del testo:

- Regressione Logistica;
- Naive Bayes Multinomiale.

L'algoritmo Naive Bayes può essere facilmente scritto nel codice, fornendo previsioni del modello in pochissimo tempo. Naive Bayes è semplice e veloce nel prevedere la classe di un set di informazioni di esempio, oltre a funzionare bene nella previsione multiclasse.
La Logistic Regression utilizza l'equazione della regressione lineare per generare output binari discreti, ma a differenza della regressione lineare, la sua funzione di costo è la funzione Sigmoide.L'ipotesi di questo algoritmo tende a limitare la funzione logistica tra 0 e 1.
Questi due modelli vengono poi addestrati usando le due rappresentazioni di testo prima sviluppate, Bag of Words e TF-IDF.
Si confrontano diversi iperparametri per migliorare il modello, valutando l'accuracy sul training set e sul test set.
Questo aiuta a scegliere il miglior modello e la migliore rappresentazione del testo per un task di classificazione del sentiment.

### 5.9.1 Logistic Regression with Bag of Words

Si provano diversi valori dell'iperparametro $c$ per il modello di *Logistic Regression*, utilizzando la vettorizzazione del Bag of Words.
L'iperparametro $c$ controlla la regolarizzazione ed evita l'overfitting:

- $c$ piccolo (ad esempio, 0.001, 0.01): si ottiene maggiore regolarizzazione, quindi il modello è più semplice e tende a generalizzare meglio, ma potrebbe sottoprevedere i dati, ottenendo alto bias;
- $c$ grande (ad esempio, 1, 10): si ottiene minore regolarizzazione, il modello si adatta meglio ai dati di addestramento ma rischia di overfittare, con una minore capacità di generalizzazione, ottenendo alta varianza.

In questo caso, i diversi valori di $c$ sono testati per trovare il miglior compromesso tra accuratezza sul training set e sul test set. Osservando i risultati per $c = 0.01$ e $c = 0.1$ sembrano dare il miglior equilibrio tra train accuracy e test accuracy. Di seguito i risultati:

```
LogisticRegression(C=0.001, max_iter=500, random_state=1)
Train accuracy score : 0.7009035216151653
Test accuracy score : 0.6901826308083533

 ----------------------------------------

LogisticRegression(C=0.01, max_iter=500, random_state=1)
Train accuracy score : 0.7390864606611698
Test accuracy score : 0.7148031132002775

 ----------------------------------------

LogisticRegression(C=0.1, max_iter=500, random_state=1)
Train accuracy score : 0.7679259459042922
Test accuracy score : 0.7160360638051938

 ----------------------------------------

LogisticRegression(C=1, max_iter=500, random_state=1)
Train accuracy score : 0.7883755875780226
Test accuracy score : 0.7023965477383063

 ----------------------------------------

LogisticRegression(C=10, max_iter=500, random_state=1)
Train accuracy score : 0.7936830546351237
Test accuracy score : 0.6906449872851969
```

Figure 6: Risultati ottenuti per diversi valori dell'iperparametro *c*.

### 5.9.2 Naive Bayes Multinomiale with Bag of Words

Si provano diversi valori del parametro *alpha* per il modello di *Naive Bayes Multinomiale*, utilizzando la vettorizzazione del Bag of Words.
*Alpha* è il parametro di smussamento di Laplace ed evita le probabilità nulle.

- *Alpha* = 0: non si ottiene nessuna regolarizzazione. Il modello si basa esclusivamente sui dati osservati, senza alcuna correzione per termini non visti. Questo può portare a problemi se una parola non compare in una classe, perché la probabilità diventa zero;
- *Alpha* > 0 (es. 0.2, 0.6, 0.8, 1): si ha l'applicazione della smoothing. Il modello aggiunge un valore piccolo a tutte le probabilità per evitare zeri, rendendolo più robusto a dati scarsi o squilibrati tra le classi. Per valori bassi di alpha si ha uno smoothing più leggero, mentre per valori

più alti di alpha si ottiene maggiore regolarizzazione, migliorando così la generalizzazione.

Di seguito i risultati:

```
MultinomialNB(alpha=0)
Train accuracy score : 0.7114991908761655
Test accuracy score : 0.6804731447946367

 ---------------------------------------

MultinomialNB(alpha=0.2)
Train accuracy score : 0.7105841103490791
Test accuracy score : 0.6816675656931495

 ---------------------------------------

MultinomialNB(alpha=0.6)
Train accuracy score : 0.7101410187254373
Test accuracy score : 0.6823996301148185

 ---------------------------------------

MultinomialNB(alpha=0.8)
Train accuracy score : 0.7100832241658318
Test accuracy score : 0.6823611004084149

 ---------------------------------------

MultinomialNB(alpha=1)
Train accuracy score : 0.709967635046621
Test accuracy score : 0.6823996301148185
```

Figure 7: Risultati ottenuti per diversi valori dell'iperparametro *alpha*.

### 5.9.3 Regressione Logistica with TF-IDF

Come nel caso precedente, si provano diversi valori dell'iperparametro $c$ per il modello di *Logistic Regression*, utilizzando la vettorizzazione del TF-IDF.
Di seguito i risultati:

```
LogisticRegression(C=0.001, max_iter=500, random_state=1)
Train accuracy score : 0.6454689065269322
Test accuracy score : 0.6400554827772212

 ----------------------------------------

LogisticRegression(C=0.01, max_iter=500, random_state=1)
Train accuracy score : 0.679586961547353
Test accuracy score : 0.6730369114587347

 ----------------------------------------

LogisticRegression(C=0.1, max_iter=500, random_state=1)
Train accuracy score : 0.7335959774986515
Test accuracy score : 0.7143407567234338

 ----------------------------------------

LogisticRegression(C=1, max_iter=500, random_state=1)
Train accuracy score : 0.7655948986668721
Test accuracy score : 0.7223934653617939

 ----------------------------------------

LogisticRegression(C=10, max_iter=500, random_state=1)
Train accuracy score : 0.7868632966016799
Test accuracy score : 0.7080604145796409
```

Figure 8: Risultati ottenuti per diversi valori dell'iperparametro $c$.

### 5.9.4 Naive Bayes Multinomiale with TF-IDF

Come nel caso precedente, si provano diversi valori del parametro *alpha* per il modello di *Naive Bayes Multinomiale*, utilizzando la vettorizzazione del TF-IDF.
Di seguito i risultati:

```
MultinomialNB(alpha=0)
Train accuracy score : 0.7177410033135547
Test accuracy score : 0.6788163674192803

 ---------------------------------------

MultinomialNB(alpha=0.2)
Train accuracy score : 0.7159493719657857
Test accuracy score : 0.6813593280419203

 ---------------------------------------

MultinomialNB(alpha=0.6)
Train accuracy score : 0.7133389843569392
Test accuracy score : 0.6815519765739385

 ---------------------------------------

MultinomialNB(alpha=0.8)
Train accuracy score : 0.7119133852200046
Test accuracy score : 0.6814363874547276

 ---------------------------------------

MultinomialNB(alpha=1)
Train accuracy score : 0.710815288587501
Test accuracy score : 0.680974030977884
```

Figure 9: Risultati ottenuti per diversi valori dell'iperparametro *alpha*.

## 5.10 Model Evaluation

Dopo aver addestrato i due modelli, si può identificare il modello migliore da scegliere, ossia LogisticRegression(C=1, max_iter=500, random_state=1), e lo si addestra, lasciandolo fare previsioni sul training set e sul test set, per poi calcolarne l'accuracy.

```
Train accuracy score : 0.7655948986668721
Test accuracy score : 0.7223934653617939
```

Figure 10: Risultati dell'accuracy del training set e del test set.

Successivamente, se ne visualizza la matrice di confusione, che aiuta a capire quali classi sono più difficili da distinguere.

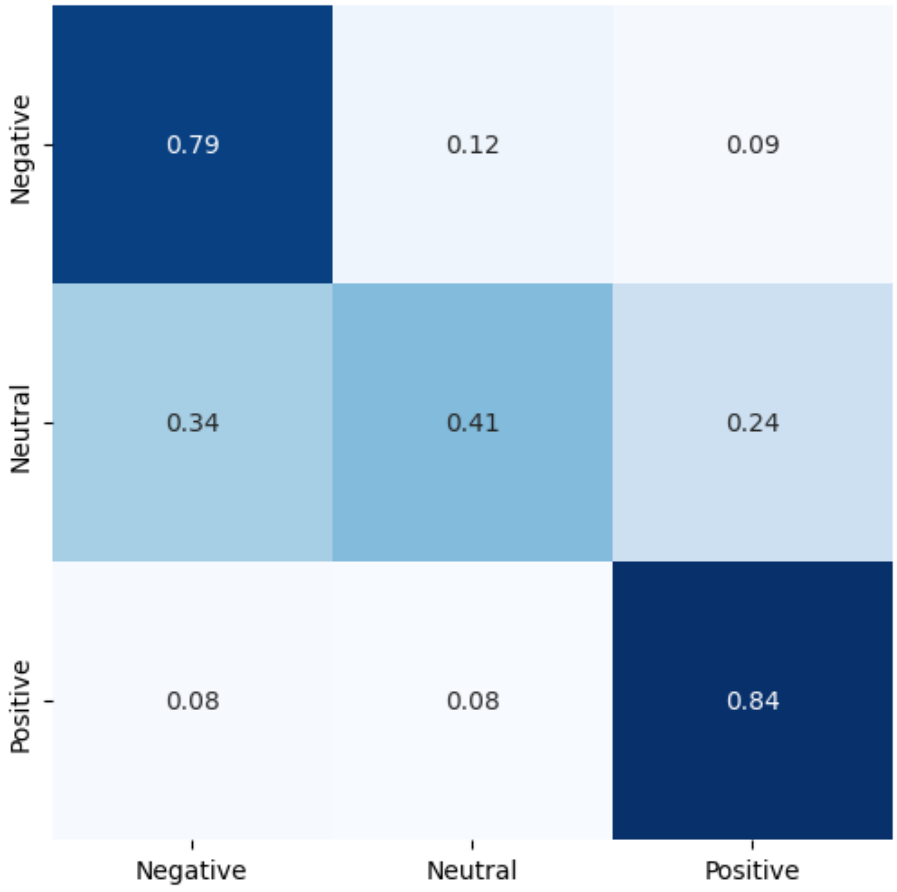


Figure 11: Matrice di confusione.

### 5.11 Deployment

Nell'ultima sezione del codice, si salva il modello addestrato e il TF-IDF vectorizer sul file usando il modulo *pickle*, per poter permettere di riutilizzare il modello senza doverlo riaddestrare ogni volta.
Si definisce la funzione *get_sentiment()* e si chiama la funzione su due recensioni di test.

```
# recensione positiva
review = "This chips packet is very tasty. I highly recommend this!"
print(f"This is a {get_sentiment(review)} review!")
```

```
This is a Positive review!
```

```
# recesione negativa
review = "This product is a waste of money. Don't buy this!!"
print(f"This is a {get_sentiment(review)} review!")
```

```
This is a Negative review!
```

Figure 12: Codice e output dei test svolti.

## 6 Risultati

I risultati ottenuti dal sistema di classificazione del sentiment mostrano una buona capacità di distinzione tra le recensioni positive, negative e neutre da parte del classificatore Logistic Regression con TF-IDF.
A differenza di Naïve Bayes, la Logistic Regression non assume che le parole siano indipendenti tra loro, riuscendo così a catturare meglio le relazioni semantiche tra i termini presenti nei testi. Questo aspetto è fondamentale nell'analisi del linguaggio naturale, dove il significato di una parola può variare in base al contesto.
Inoltre, la Logistic Regression è un modello discriminativo, il che significa che si concentra direttamente sulla distinzione tra le classi, piuttosto che stimare la probabilità congiunta come fa il modello generativo di Naïve Bayes. Questo approccio spesso si traduce in una maggiore accuratezza nella classificazione testuale.

Un altro fattore determinante è l'uso della rappresentazione TF-IDF, che assegna un peso maggiore alle parole più rilevanti per un documento e riduce l'importanza di quelle troppo frequenti nel corpus. Questo aiuta la Logistic Regression a individuare meglio le caratteristiche distintive di ogni classe, migliorando la separabilità dei dati.

# 7 Sviluppi futuri

Possibili sviluppi futuri del presente progetto potrebbero essere:

- Includere l'integrazione di tecniche di deep learning, come le Reti Neurali Convoluzionali (CNN) o le Long Short-Term Memory (LSTM), per migliorare ulteriormente la comprensione del sentiment, soprattutto con testi complessi e/o ambigui.
- Utilizzare word embeddings che potrebbe arricchire la rappresentazione semantica delle parole, portando a una maggiore precisione nelle previsioni;
- Espandere la classificazione del sentiment a livelli più granulosi, ad esempio con l'introduzione di categorie intermedie tra positivo e negativo, per una maggiore sofisticazione dell'analisi;
- Implementare una interfaccia utente che permetta di caricare recensioni in tempo reale e ottenere risposte immediate.

# 8 Conclusioni

Il codice sviluppato ha creato un sistema per analizzare il sentiment delle recensioni di prodotti utilizzando tecniche di NLP e machine learning.
Dopo aver pulito e trasformato i dati con il preprocessing, sono stati applicati il TF-IDF vectorizer e modelli di classificazione come Regressione Logistica e Naive Bayes. I risultati, valutati tramite accuratezza e matrici di confusione, mostrano l'importanza della scelta dei parametri. La word cloud ha offerto un'interessante visualizzazione delle parole più frequenti per ciascun sentiment.
Infine, grazie al salvataggio dei modelli e del vettorizzatore con pickle, il sistema è facilmente riutilizzabile in altri contesti, rendendolo utile per applicazioni come l'analisi automatica delle recensioni online.