ClaRa News

Progetto per il corso “Ingegneria della Conoscenza”

Membri del gruppo:

- Marino Giuseppe:
  697617
  g.marino45@studenti.uniba.it
- Mauro Alessandro:
  698464
  a.mauro11@studenti.uniba.it

Link Repository: https://github.com/the-stonecutters/ClaRa

INTRODUZIONE

Il progetto consiste in un programma basato su apprendimento, supervisionato e non, in grado di:

- Valutare diversi algoritmi di classificazione su un dataset di notizie prese da ANSA.it (apprendimento supervisionato)
- Usare gli algoritmi di apprendimento supervisionato per classificare notizie prese da link inseriti dall’utente o digitando il testo di un articolo (anche creato da un utente)
- Raccomandare le 10 notizie più rilevanti (tra quelle presenti nel dataset) attraverso algoritmi di apprendimento supervisionato e non a un determinato utente che ha espresso delle preferenze

PROGRAMMI UTILIZZATI

Il programma è stato sviluppato in **Python,** usando **PyCharm** come IDE. Come librerie per gli algoritmi di apprendimento è stato fatto uso di **Scikit,** per la creazione dei grafici invece la libreria scelta è stata **Matplotlib**, per la gestione dei dataset la libreria **Pandas** e infine **NLTK** per il Natural Language Processing (NLP).

DATASET

Il dataset di apprendimento, utilizzato dal programma, è stato generato da noi effettuando scraping sul sito ANSA.it con le notizie delle principali categorie:

- Mondo
- Cronaca
- Politica
- Sport
- Tecnologia
- Economia
- Cultura

Il dataset contenente le preferenze degli utenti è stato creato chiedendo a diversi utenti di valutare notizie casuali secondo i loro interessi, su una piattaforma appositamente creata.

L’ultimo dataset contiene le notizie dell’ultima ora (topnews) prese da ANSA.it negli ultimi 2 anni.

LAVORARE SU DOCUMENTI DI TESTO

La rappresentazione dei dati sotto forma di documento testuale non è idonea, poiché gli algoritmi di apprendimento per il loro funzionamento richiedono una rappresentazione vettoriale di dimensione costante.
Perciò, prima di elaborare i documenti, questi sono stati convertiti in una rappresentazione vettoriale di tipo TF-IDF.

Il TF-IDF (Term Frequency – Inverse Document Frequency) , dato un documento j e una keyword i, misura il prodotto tra la Term Frequency (ovvero il numero di volte in cui la keyword i appare nel documento j) e l'Inverse Document Frequency misurata come

$$IDF(i) = log \frac{N}{n(i)}$$

con:

- N: numero di tutti i documenti
- n(i): il numero di documenti in cui appare la keyword i

in formula il TF-IDF si calcola:

$$TF\text{-}IDF(i,j) = TF(i,j) * IDF(i)$$

Questa rappresentazione dà più importanza ai termini utilizzati meno frequentemente nel corpus, questo perché le parole di uso più frequente portano meno informazioni sul contenuto del documento (i.e. parole specifiche per un argomento tendono a non comparire in documenti che non trattato di quello specifico argomento, parole generiche tendono a ripetersi più volte in diversi documenti non correlati tra loro)

APPRENDIMENTO SUPERVISIONATO

1. Algoritmi di Apprendimento Supervisionato
Per predire la categoria di nuovi articoli (siano essi inseriti tramite link o scritti dall'utente), sono stati valutati vari modelli di classificazione basati su apprendimento supervisionato presenti nella libreria python sklearn. La decisione di tenere in considerazione un maggior numero di algoritmi è stata presa per verificare il diverso funzionamento e le diverse perfomance (il cui confronto verrà spiegato nella sezione riguardante la Model Selection). Di seguito sono elencati gli algoritmi valutati:

- Linear Models:

  - **Ridge Classifier**: Metodo basato sulla Ridge Regression, Il classificatore converte i valori in un range compreso tra -1 e 1, successivamente gestisce il problema come un task di regressione
  - **Stochastic Gradient Descent (SGD)**: Metodo iterativo per ricercare i minimi di una funzione; nel caso della Discesa di Gradiente Stocastica l'aggiornamento incrementale avviene con esempi selezionati in maniera causale
  - **Logistic Regression**: Modello statistico utilizzato per ottenere la probabilità di appartenenza ad una determinata classe. Si basa sulla funzione logistica della sigmoide per convertire i valori reali in un range compreso tra 0 e 1

- Probabilistic Classifier:

  - **Complement Naive Bayes:** Algoritmo di classificazione basato sul teorema di Bayes

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

    Questa versione dell'algoritmo è particolarmente indicata per dataset sbilanciati: questo infatti utilizza le statistiche dei complementi di ciascuna classe per calcolare la pesatura dei modelli. Inoltre è anche consigliato per operazioni di text classification.

- Decision Tree:

  - **Decision Tree Classifier**: L'albero di decisione (o classificazione) è costituito da due tipologie di nodi:

    -Nodi Interni (non foglia): nodi etichettati con delle condizioni (funzioni booleane) applicabili agli esempi e con due figli (radici di sottoalberi) etichettati rispettivamente con "True" e "False"

    -Nodi Foglia: nodi etichettati con una stima puntuale della classe.
    La Classificazione avviene partendo dalla radice e, per ogni condizione incontrata, si effettua una valutazione e si segue l'arco corrispondente al risultato; raggiunta una foglia si assegna la classe corrispondente.

- Case Based Reasoning:

- **K-Nearest Neighbors**: algoritmo utilizzato nel riconoscimento di pattern per la classificazione di oggetti basandosi sulle caratteristiche dei *k* oggetti più vicini a quello considerato. Un oggetto è classificato in base alla maggioranza dei voti dei suoi *k* vicini. *k* è un intero positivo tipicamente non molto grande. La scelta di *k* dipende dalle caratteristiche dei dati. Generalmente, all'aumentare di *k*, si riduce il rumore che compromette la classificazione. Al fine dell'apprendimento, lo spazio multidimensionale viene partizionato in regioni in base alle posizioni e alle caratteristiche degli oggetti di apprendimento, rappresentati come vettori. Un oggetto è assegnato alla classe C se questa è la più frequente fra i *k* esempi più vicini all'oggetto sotto esame, la vicinanza si misura in base alla distanza fra punti. I vicini sono presi da un insieme di oggetti per cui è nota la classificazione corretta.

- Ensemble Learning:

- **Random Forest:** Modello ottenuto mediando su più decision trees; l'idea alla base è quella di addestrare un certo numero di alberi su una parte del dataset, ogni albero farà una predizione per ogni esempio da classificare e le predizioni verranno aggregate per ottenere la predizione finale per l'esempio
- **Extra Trees**: Modello simile al precedente, la differenza è data dalla scelta degli alberi che avviene in maniera totalmente casuale.

- Support Vector Machine

- **SVC (Support-Vector Classification)**: Modello di apprendimento per la regressione e la classificazione. Dato un insieme di esempi per l'addestramento, ognuno dei quali etichettato con la classe di appartenenza fra le due possibili classi; un algoritmo di addestramento per le SVC costruisce un modello che assegna i nuovi esempi a una delle due classi, ottenendo quindi un classificatore lineare binario non probabilistico.

2. Ottimizzazione degli Iperparametri

Per migliorare le performance degli algoritmi di classificazione presi in esame è stata effettuata l'ottimizzazione degli iperparametri.
Gli iperparametri non sono altro che i parametri impostati all'algoritmo di classificazione durante la sua creazione. Tuttavia, poiché gli algoritmi presenti nella libreria sono impostati con valori di default, i quali spesso non gli permettono di performare al meglio, è opportuno ricercare i valori corretti da sostituire per ottenere il massimo da ogni algoritmo. Per ricercare i parametri migliori da passare ai classificatori ci siamo affidati alle tecniche di ricerca note come "Validation Curve" e "Grid Search".

2.0.1 K-Fold Cross Validation

La K-Fold Cross Validation (CV) è un metodo usato per valutare modelli di apprendimento con un dataset di piccole dimensioni.
Questo metodo consiste nel dividere il dataset in k set delle stessa dimensione (nel nostro caso k = 5) chiamati "fold" e si addestra il modello k volte, escludendo un "fold" distinto che verrà usato per la validazione dell'accuratezza; i valori di ogni parametro vengono quindi

ottimizzati in base all'errore su ogni esempio, infine si restituisce il modello con le migliori impostazioni dei parametri.

2.1 Validation Curve

Tale metodo consente di avere un feedback visivo su quali valori garantiscano i risultati migliori.
Per il suo utilizzo vengono inseriti nel programma una serie di valori per ogni parametro da ottimizzare (i valori di ogni parametro sono stati scelti inizialmente ampliando il range indicato sulla documentazione e, nel caso in cui il grafico dovesse mostrare valori in continua salita, la procedura è stata ripetuta adattando il range in base ai risultati ottenuti precedentemente) e tramite una CV vengono calcolati i risultati, i quali sono mostrati all'utente tramite un grafico in cui vengono confrontati il "Training Score" e il "Cross Validation Score", in base ai valori di massimo del grafico si può dedurre quali valori garantiscano perfomance migliori.

2.1.1 Esempio Utilizzo Validation curve

Per generare la validation curve abbiamo ricercato i parametri da ottimizzare nella documentazione di scikit: per ogni algoritmo di apprendimento supervisionato sono riportati nella pagina specifica tutti i parametri che possono essere modificati, con un range di valori consigliati e una spiegazione sul funzionamento.

Come esempio useremo l'algoritmo "SVC", il quale si è successivamente dimostrato tra i migliori a livello di performance, ma lo stesso procedimento è stato svolto per ogni altro algoritmo descritto in precedenza e tutti i grafici sono presenti nella repo su github.

Per l'algoritmo preso ora in esame, la documentazione è presente al seguente link:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

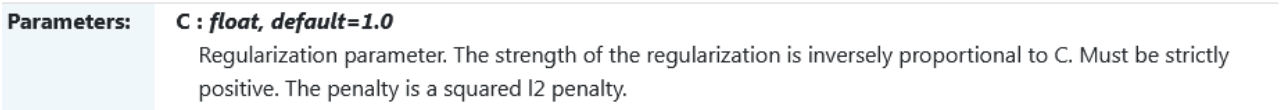


Figura 1: Esempio del parametro "C" presente nella documentazione

Con i parametri numerici presi dalla documentazione è stato scritto il codice per ricercare i parametri migliori (per ogni algoritmo):

```
from sklearn.svm import SVC
import numpy as np

from iperparametri.validation_curve import do_validation_curve
import matplotlib.pyplot as plt


for param_name, param_range in (
    ('tol', np.logspace(-7, 0, 7)),
    ('C', np.arange(0, 2, 0.1)),
):
    do_validation_curve(SVC(), param_name, param_range)
plt.show()
```

Figura 2: Codice per la Validation Curve dell'algoritmo SVC

Da questo codice si ottengono due curve; la prima del parametro “tol” e la seconda del parametro “C”.

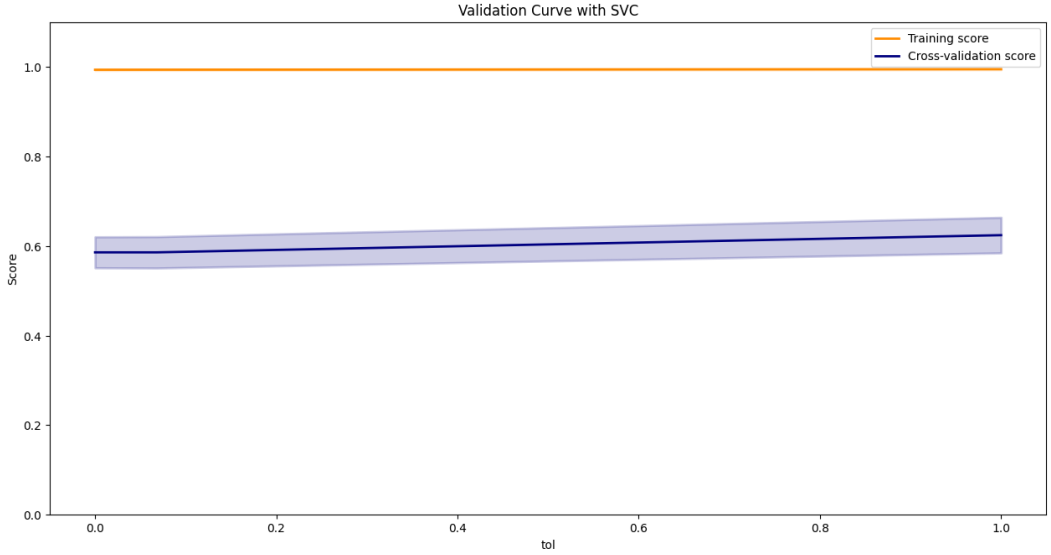


Figura 3: Curva del parametro "tol"

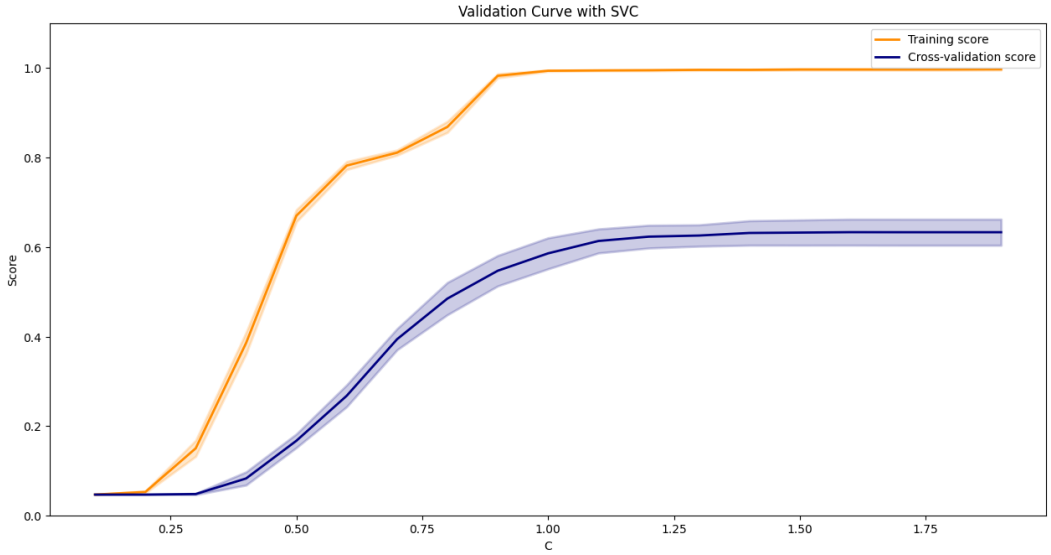


Figura 4: Curva del parametro "C"

In questo caso, nel lato destro del grafico di entrambe le curve, si vede come la curva del “Cross Validation score” sia orizzontale e non sembra salire, possiamo quindi vedere in quale punto (o anche più di un punto) il grafico abbia altezza maggiore e scegliere quei valori per impostare gli iperparametri. Nel caso in cui il grafico avesse mostrato una curva ascendente il test sarebbe stato eseguito nuovamente ampliando il range, fino a trovarci in una situazione in cui la linea sia piatta oppure discendente.

Dai grafici possiamo quindi prendere come valori per gli iperparametri:

tol = 1.0
C = 1.75

Tuttavia, per avere maggiore sicurezza e risultati più precisi, con i valori qui ottenuti andremo ad effettuare una “grid search”, in modo da poter anche confrontare i parametri non numerici che non avevamo potuto calcolare in precedenza.

2.2 Grid Search

Questo metodo di ottimizzazione genera i possibili iperparametri candidati tramite una griglia di valori, specificata in maniera opportuna dal parametro “param_grid”, il quale contiene un range di valori inseriti dall’utente. I parametri sono stati scelti in base ai valori ottenuti in precedenza dalla Validation Curve: questo ci ha permesso di velocizzare il processo, avendo già una base di valori su cui partire e quindi confrontare solo i valori che si erano già dimostrati come i candidati migliori ad una prima analisi.
In maniera automatica vengono valutate le possibili combinazioni di iperparametri, effettuando una CV, fino a trovare quella migliore che sarà mantenuta.
A fine processo vengono mostrati gli iperparametri migliori per il modello di classificazione scelto.

2.2.1 Esempio Utilizzo “Grid Search”

Come per la “validation curve”, i parametri sono presi dalla documentazione della libreria utilizzata.

In questo caso andremo a valutare anche parametri non numerici

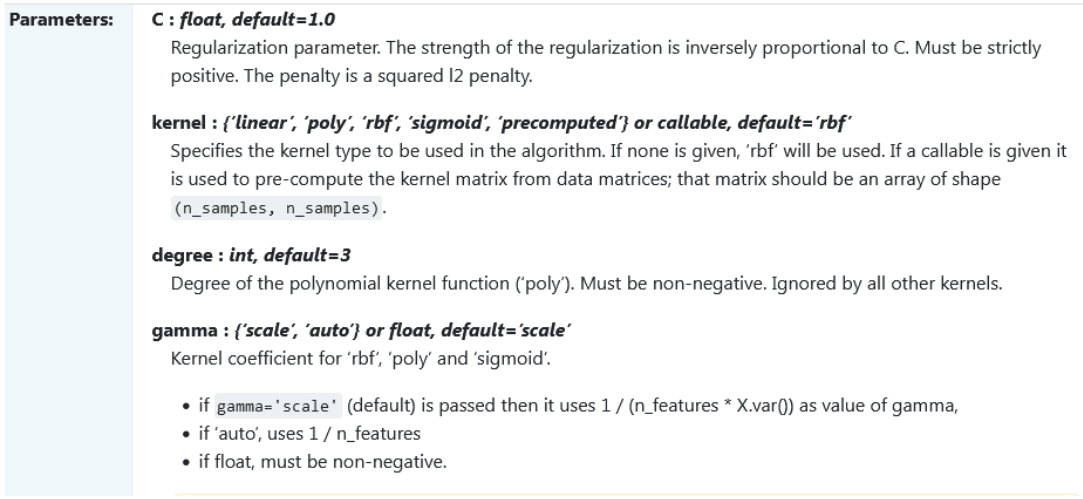


Figura 5: Esempio di altri parametri (anche non numerici) presenti nella documentazione di SVC

Sempre in maniera analoga alla “Validation Curve”, con i parametri presenti abbiamo generato il codice per effettuare la “grid search” (anche qui il codice è specifico per algoritmo)

```
parameters = {
    'tol': [0.6, 0.8, 1.0, 1.2],
    'C': [1.0, 1.25, 1.50, 1.75, 2.0],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'gamma': ['scale', 'auto'],
    'decision_function_shape': ['ovo', 'ovr']
}
```

Figura 6: In questo caso la lista dei parametri risulta più ampia e i range dei parametri calcolati con validation curve sono stati leggermente ampliati per avere una maggiore sicurezza

Verrà effettuata una Cross Validation molto più corposa. Avendo molte più combinazioni da testare, questo potrebbe richiedere molto tempo per ottenere il risultato (anche diverse ore).

Una volta terminata l’esecuzione, verrà stampata a schermo la lista contenente i parametri migliori per ogni opzione.

Questa è la lista di iperparametri migliori per l’algoritmo SVC:

```
{'C': 1.75, 'decision_function_shape': 'ovo', 'gamma': 'scale',
'kernel': 'sigmoid', 'tol': 1.2}
```

3. Metriche di Valutazione degli Algoritmi

Dopo aver impostato i migliori iperparametri per ogni algoritmo, si è passato alla fase di valutazione.
Ogni algoritmo è stato valutato in base alle metriche di “Accuracy”, “Precision”, “Recall”, “F1-Score”.
Il significato delle metriche è il seguente:

- **Accuracy**: corrisponde al numero di elementi correttamente classificati sul numero di tutti gli elementi da classificare.

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

- **Precision**: è il numero di elementi classificati correttamente rispetto sul numero di tutti gli elementi classificati

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

- **Recall**: chiamato anche “true positive rate”, è dato dal rapporto tra il numero di elementi predetti correttamente e il numero di elementi corretti sommati agli elementi predetti erroneamente

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

- **F1-Score**: metrica che prende in considerazione quelle viste precedentemente e si calcola col rapporto tra il doppio prodotto tra precision e recall e la loro somma

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

3.2 Valutazione e Analisi Risultati

Per la valutazione abbiamo optato per una 5-Fold Cross Validation e, per ogni algoritmo valutato, stampato a schermo i risultati ottenuti, calcolandone media e scostamento, generato un grafico per facilitare la lettura dei risultati.

Con i parametri ottenuti tramite “Grid Search“ abbiamo effettuato la valutazione di tutti gli algoritmi di apprendimento supervisionato in precedenza.

La valutazione è stata effettuata con i seguenti parametri:

```
(RidgeClassifier(alpha=0.01, tol=0.1, solver='sparse_cg', random_state=42), 'Ridge Classifier'),
(ComplementNB(alpha=0.2136), 'Naive Bayes'),
(SGDClassifier(alpha=1e-06, tol=0.1, loss='hinge', penalty='l2', random_state=42), 'SGD Classifier'),
(KNeighborsClassifier(algorithm='brute', leaf_size=5, n_neighbors=10, p=1, weights='distance', metric='cosine'),
 'Neirest Neighbors'),
(ExtraTreesClassifier(criterion='gini', max_depth=150, max_features='sqrt', min_samples_leaf=1,
                      min_samples_split=20, n_estimators=150, random_state=42), 'Extra Trees'),
(RandomForestClassifier(criterion='gini', max_depth=100, max_features='sqrt', max_samples=600,
                        min_samples_leaf=2, min_samples_split=4, n_estimators=220, random_state=42),
 'Random Forest'),
(DecisionTreeClassifier(criterion='gini', max_features='sqrt', min_samples_leaf=1, min_samples_split=3,
                        random_state=42), 'Decision Tree'),
(SVC(C=1.75, decision_function_shape='ovo', gamma='scale', kernel='sigmoid', tol=1.2, random_state=42),
 'SVC Classifier'),
(LogisticRegression(C=3.5, penalty='l1', solver='liblinear', tol=1e-07, random_state=42), 'Logistic Regression')
```

Figura 7: Codice del K-Fold con i parametri impostati per ogni classificatore

Dopo aver effettuato una CV sui diversi algoritmi, abbiamo generato due grafici per ogni algoritmo: il primo con i risultati della Cross Validation, il secondo con la media dei valori di “Accuracy”, “Precision”, “Recall” e “F1-Score”.

Di seguito sono riportati i grafici con le medie dei risultati, per l’algoritmo migliore il grafico con i valori della Cross Validation. Tutti i valori ottenuti dalle Cross Validation eseguite sono presenti nel file “kfold.txt” e in maniera strutturata nel file “results.csv”, entrambi presente nella repo insieme ai vari grafici.

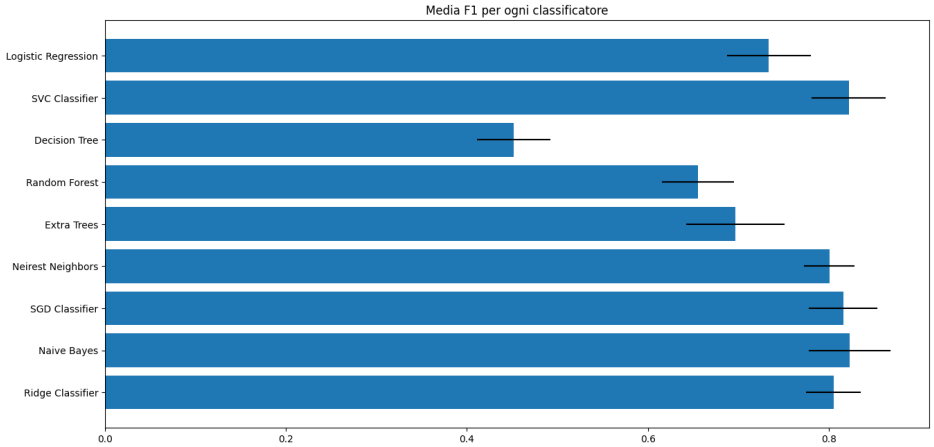
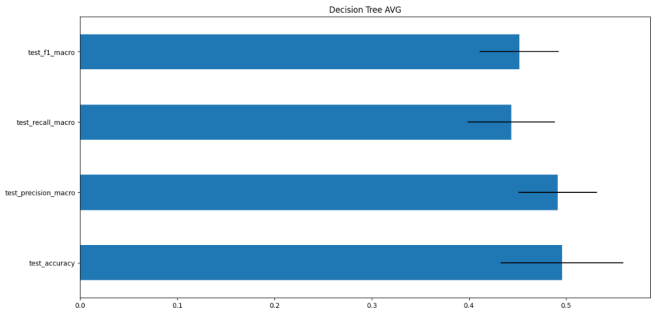
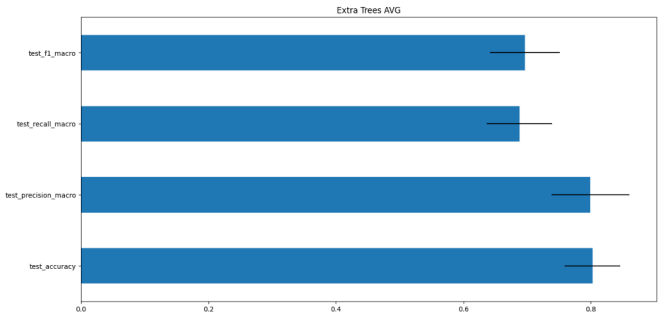
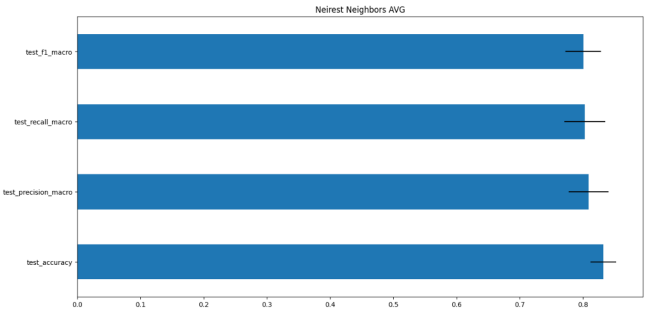


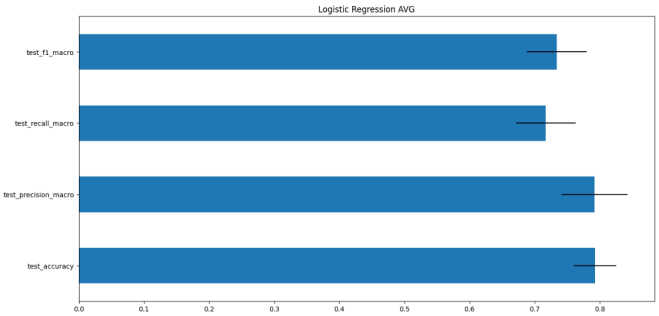
Figura 8: Grafico di confronto tra il valore medio dello score F1 degli algoritmi valutati

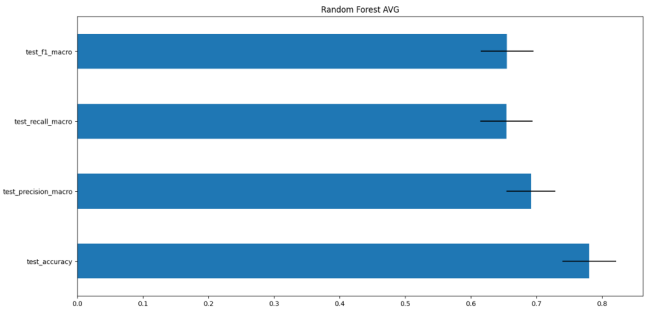
Di seguito sono riportati tutti i risultati delle Cross Validation:

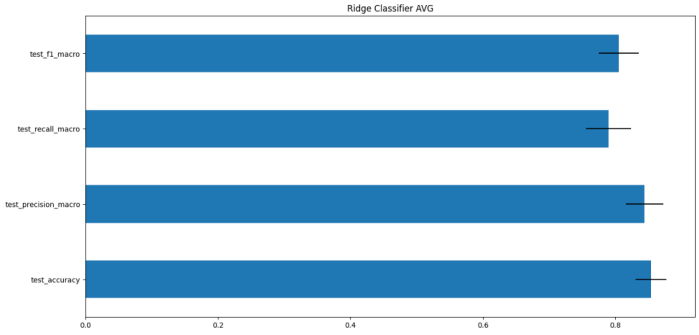


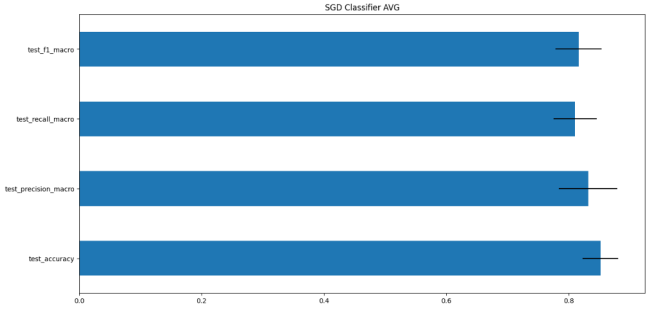












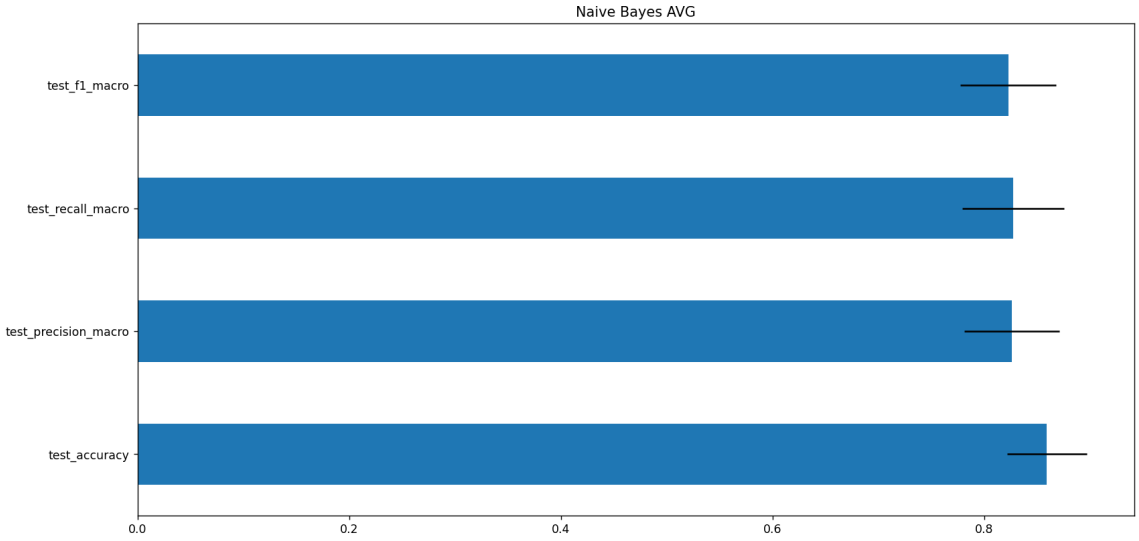


Figura 9: Grafico dei valori medi ottenuti dal K-fold effettuato su Naive Bayes

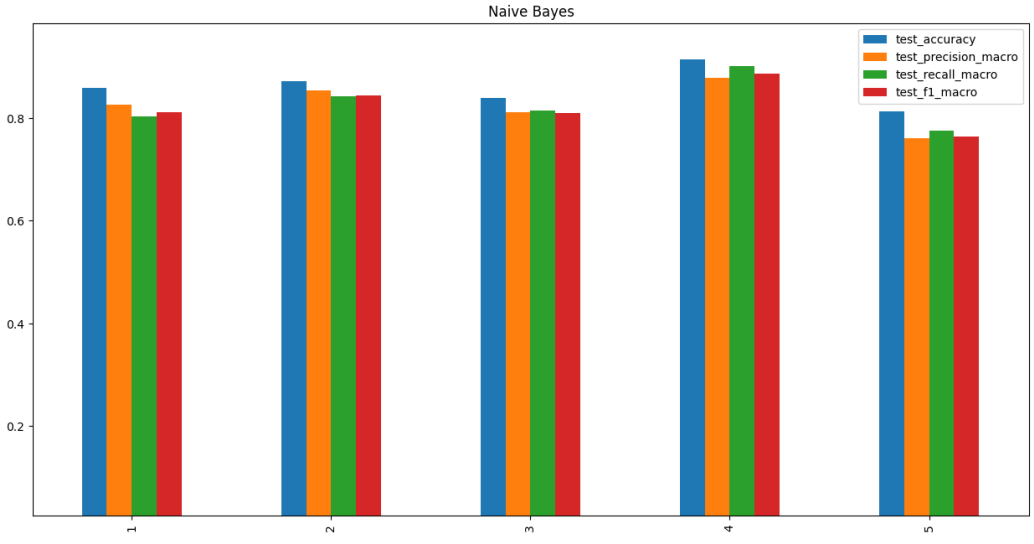


Figura 10: Grafico dei valori ottenuti dal K-fold effettuato su SVC

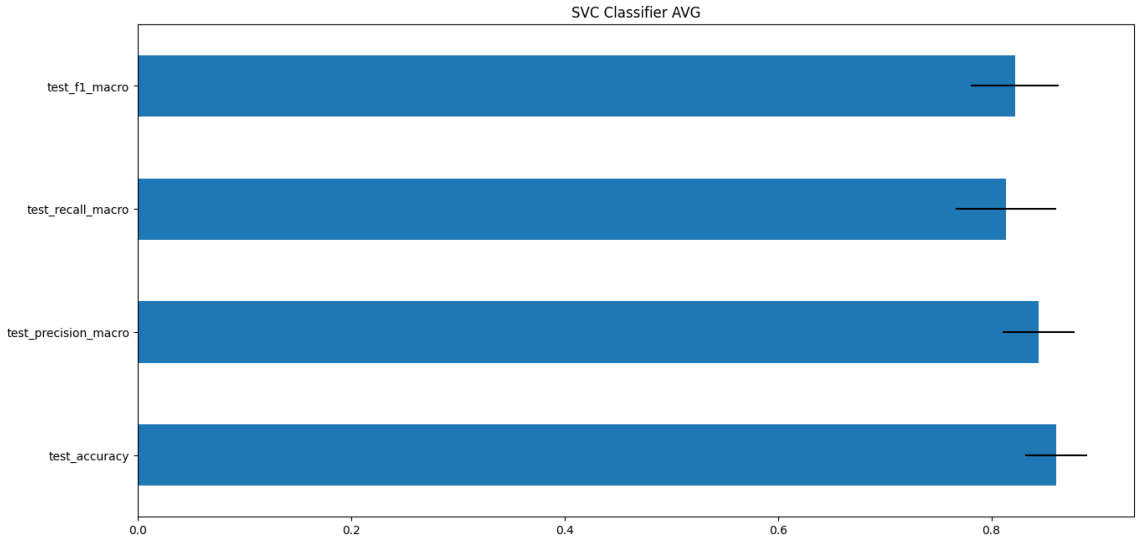


Figura 11: Grafico dei valori medi ottenuti dal K-fold effettuato su SVC

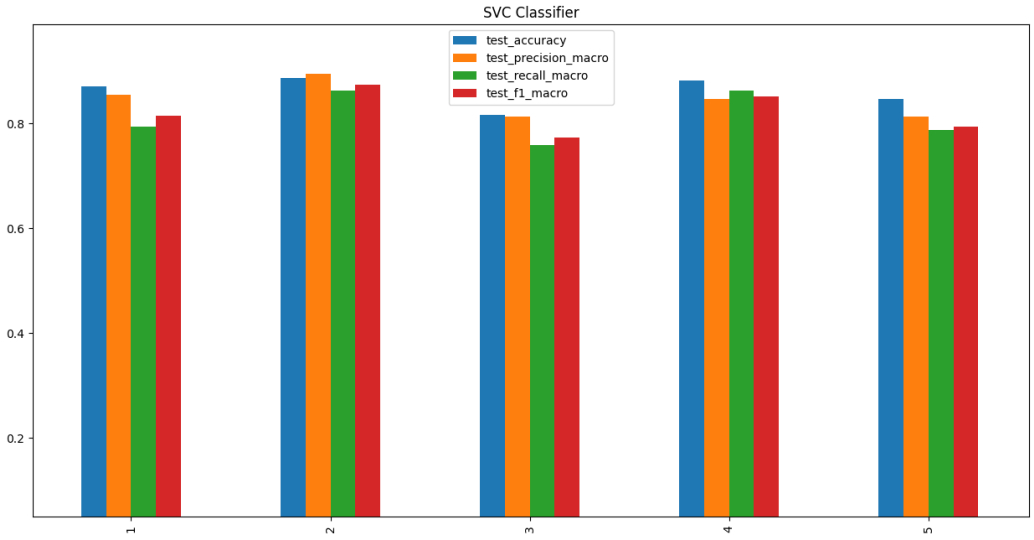


Figura 12: Grafico dei valori medi ottenuti dal K-fold effettuato su SVC

Dai risultati è possibile vedere come “SVC” e “Complement Naive Bayes” siano molto vicini a livello di prestazioni, entrambi gli algoritmi hanno infatti uno score F1 di 0.82 e uno scostamento di 0.04; inoltre entrambi gli algoritmi hanno una “accuracy” molto simile:

Complement NB
test_accuracy: 0.859438 σ(0.037802)

SVC
test_accuracy: 0.860583 σ(0.028935)

Si è deciso di usare come algoritmo il “Complement Naive Bayes”, non solo per il suo ottimo funzionamento con task di text classification, ma anche per il minor tempo di training richiesto e di conseguenza il minor tempo di esecuzione del task.

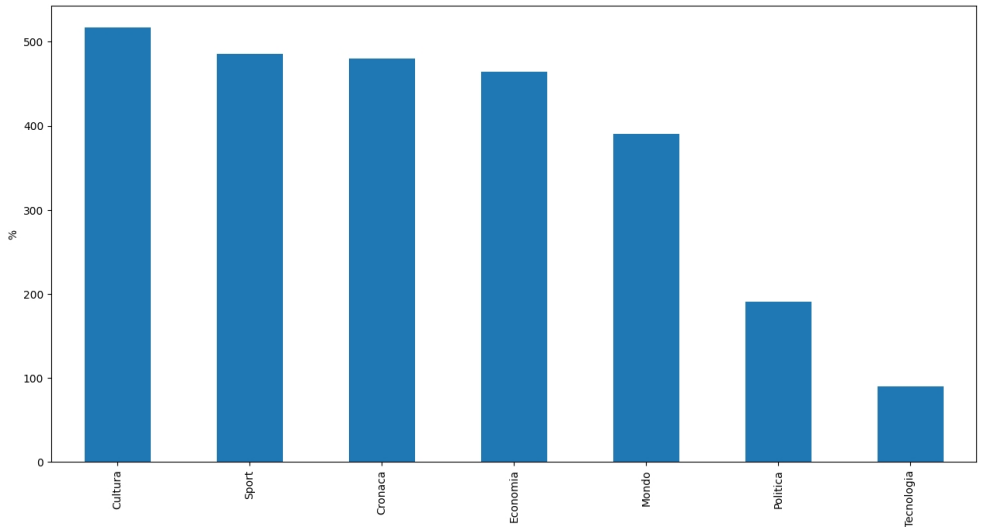
DIMOSTRAZIONE CLASSIFICATORE

All’avvio del programma ci viene presentata la schermata iniziale in cui è possibile scegliere tra 4 opzioni:

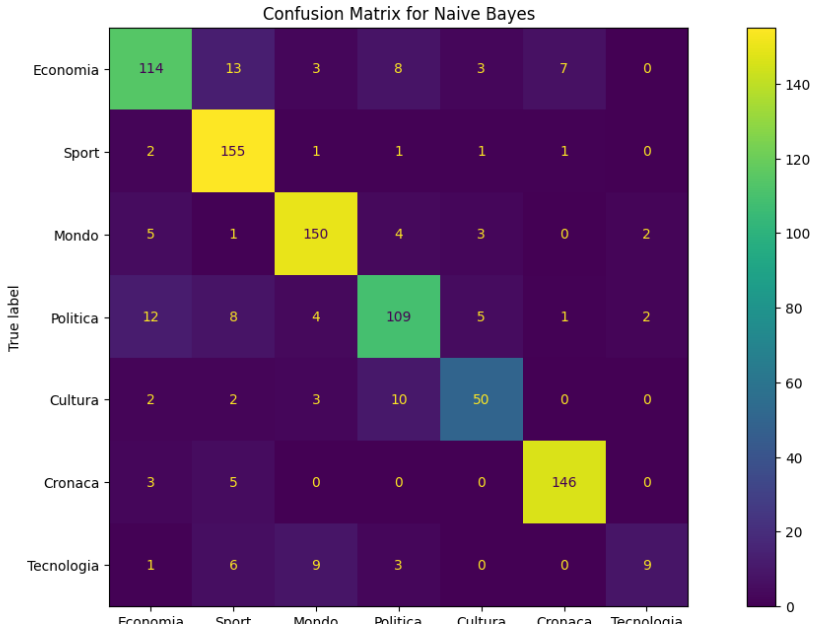
```
Loading...
Ready
Cosa vuoi fare?
1) Valuta classificatore
2) Classifica una notizia (LINK)
3) Classifica il dataset di notizie d'ultim'ora di ANSA
4) Classifica una stringa data in input
0) Esci
```

La Prima opzione permette di visualizzare la distribuzione del dataset di notizie e la matrice di confusione del classificatore scelto:

Inizialmente viene mostrata la distribuzione delle notizie:



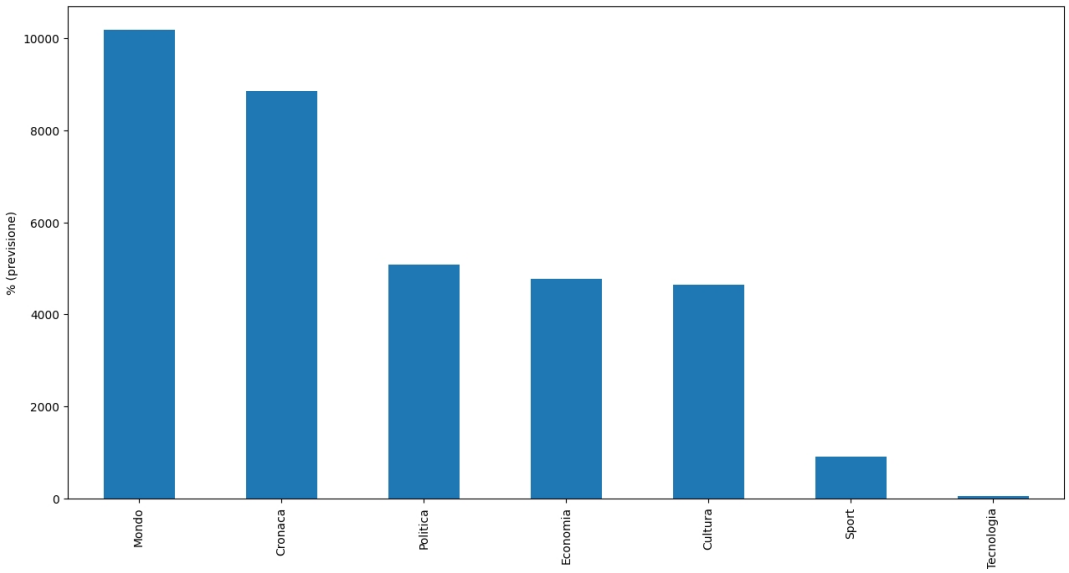
Successivamente per l’algoritmo scelto verrà mostrata la “Matrice di Confusione”, contenente la distribuzione di notizie nelle varie categorie (tutte le matrici degli altri algoritmi di classificazione sono presenti nella repo su github):



La Seconda opzione permette di classificare un link dato in input dall'utente

```
2
Inserisci il link della notizia da classificare: https://www.open.online/2023/01/13/ucraina-invio-tank-occidentali-spot-video/
Mondo
```

La Terza opzione permette di classificare il dataset delle notizie dell'ultima ora (topnews) prese da ANSA, mostrando un grafico contenente la distribuzione delle varie notizie.



Successivamente sarà possibile ricercare tramite *id* una notizia specifica per visualizzare il titolo, il link e la categoria in cui è stata classificata.

```
3
Inserisci l'id della notizia da visualizzare, 0 per uscire per uscire: 35
Staatsballett Berlino caccia ballerina nera, Ã¨ 'razzismo'
http://www.ansa.it/sito/notizie/topnews/2020/11/30/staatsballett-berlino-caccia-ballerina-nera-e-razzismo_5449ce07-cb3d-4e37-9872-c7383beb7f5e.html
Cultura
Inserisci l'id della notizia da visualizzare, 0 per uscire per uscire:
```

Infine la Quarta ed ultima opzione permette di digitare un testo a scelta da parte dell'utente che potrà essere classificato

```
4
Scrivi testo da classificare: Crollo della borsa di Milano
['Economia']
```

RECOMMENDER SYSTEM

Per la raccomandazione delle notizie abbiamo sperimentato due tecniche di raccomandazioni “Content Based”, ossia che vanno ad analizzare il contenuto delle notizie presenti nel dataset e le correlano alle notizie a cui un determinato utente ha espresso una preferenza. Le tecniche sperimentate fanno uso rispettivamente di algoritmi di clustering e di classificazione.

4.0 K-Means

Questo algoritmo va ad inizializzare in maniera casuale un dato numero di centroidi pari al numero di cluster inserito, per poi fare una prima assegnazione degli esempi del dataset. A questo punto, per ciascun centroide, verrà calcolata la media dei valori delle features, che verranno utilizzati per il successivo ricalcolo delle assegnazioni: il k-means, infatti, ha come condizione per la convergenza la minimizzazione della sommatoria degli errori quadratici.
Il numero migliore di clusters viene determinato attraverso il **“metodo del gomito”**, il quale permette di individuare il numero di clusters minimo per poter diminuire significativamente l’errore associato al modello.

4.1 Raccolta Preferenze utente

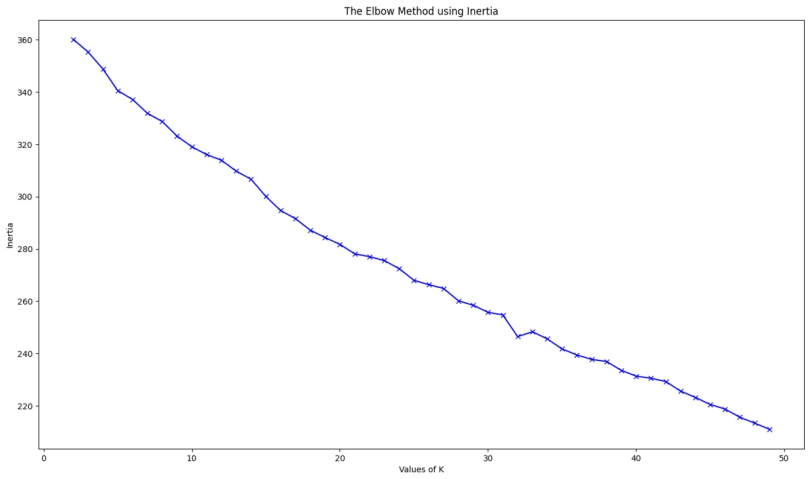
Per raccogliere i dati sui gusti dei diversi utenti è stato creato un portale, accessibile tramite telegram, in cui ogni utente è in grado di esprimere la propria preferenza su un articolo mostrato in maniera casuale dal dataset di ANSA. Le informazioni così raccolte verranno usate successivamente per raccomandare le notizie ad uno specifico utente.

4.2 Recommender System

Il sistema di raccomandazione ordina le notizie presenti nel dataset in base a una metrica “score” calcolata a partire dalle notizie valutate da un determinato utente.
Questa metrica è basata sulla “similarità del coseno”, scelta fatta considerando sia la popolarità di questa metrica nei task di text categorization, sia in quanto risulta la metrica migliore nei classificatori case-based.

4.2.1 Raccomandazione tramite uso di cluster di notizie.

Per raccomandare le notizie, abbiamo calcolato una metrica di score basata sulla media ponderata della distanza di una notizia presente nel dataset con i centroidi dei clusters.
In particolare, per poter calcolare lo score, le notizie presenti sono state suddivise in n clusters (n è un numero variabile da 2 a 50 calcolato tramite metodo del gomito).



In particolare, per una notizia i, lo score è calcolato come

$$S[i] = \sum_{n=1}^{N} cosim(C_n, i) * K_n / |P|$$

Dove

S[i] è lo score per la notizia i
N è il numero di cluster
C_n è il centroide del cluster n
P è l'insieme di notizie a cui l'utente ha espresso giudizio positivo
K_n è il numero di notizie in P che risiedono nel cluster n

Una volta calcolato lo score per ogni notizia non ancora valutata dall'utente, vengono stampate le prime 10 in ordine decrescente.

```
metodo kmeans
ricerca numero cluster ottimale
<kneed.knee_locator.KneeLocator object at 0x000000001B9BF160>
21
21
Raccomandazioni:
183475  - Champions, Allegri: "Cinque ko su sei, dobbiamo avere rabbia" - https://www.ansa.it/sito/notizie/sport/calcio/2022/11/02/champions-allegri-cinque-ko-su-sei-dobbiamo-av
164446  - Kiev: violento attacco russo, 80% degli abitanti senza acqua - https://www.ansa.it/sito/notizie/mondo/2022/10/31/kiev-alcuni-quartieri-senza-elettricita-dopo-attacchi-
128416  - Serie A: Inter-Sampdoria DIRETTA - https://www.ansa.it/sito/notizie/sport/2022/10/29/serie-a-inter-sampdoria-diretta_eb183ae2-0ebc-408c-a6e4-3664188f6ced.html
182840  - Champions: il Milan spera, Napoli e Inter già qualificate - https://www.ansa.it/sito/notizie/sport/calcio/2022/10/31/champions-il-milan-spera-napoli-e-inter-gia-qualif
182911  - Milan rinnova Pioli fino al 2025 'proseguiamo progetto ambizioso' - https://www.ansa.it/sito/notizie/sport/2022/10/31/milan-rinnova-pioli-fino-al-2025-prosequiamo-prog
74420  - Champions: Pioli, Milan sta bene e sa importanza partita - https://www.ansa.it/sito/notizie/sport/calcio/2022/10/24/champions-pioli-milan-sta-bene-e-sa-importanza-parti
28978  - Champions, fiducia Allegri: non siamo ancora fuori - https://www.ansa.it/sito/notizie/sport/2022/10/24/champions-fiducia-allegri-non-siamo-ancora-fuori_e12e2d24-ecff-4e
128833  - Serie A: Inter-Sampdoria 0-0 DIRETTA - https://www.ansa.it/sito/notizie/sport/2022/10/29/serie-a-inter-sampdoria-0-0-diretta_eb183ae2-0ebc-408c-a6e4-3664188f6ced.html
129252  - Serie A: Inter-Sampdoria 1-0 DIRETTA - https://www.ansa.it/sito/notizie/sport/2022/10/29/serie-a-inter-sampdoria-1-0-diretta_eb183ae2-0ebc-408c-a6e4-3664188f6ced.html
129673  - Serie A: Inter-Sampdoria 2-0 DIRETTA - https://www.ansa.it/sito/notizie/sport/2022/10/29/serie-a-inter-sampdoria-2-0-diretta_eb183ae2-0ebc-408c-a6e4-3664188f6ced.html
```

4.2.2 Raccomandazione tramite algoritmo di "Rocchio"
Il secondo metodo di raccomandazione sfrutta un algoritmo noto come "algoritmo di Rocchio" per calcolare lo score da assegnare alle notizie da raccomandare.

In particolare, viene creato un classificatore di Rocchio (noto anche come NearestCentroid) sfruttando come dataset le notizie a cui l'utente ha espresso una preferenza e come classi il giudizio (positivo o negativo).

Lo score, in questo caso, terrà conto sia dei giudizi positivi che di quelli negativi ed è calcolato come segue

$$S[i] = cosim(C_p, i) * \beta - cosim(C_n, i) * \gamma$$

Dove

C_p è il centroide delle notizie a cui l'utente ha espresso giudizio positivo
C_n è il centroide delle notizie a cui l'utente ha espresso giudizio negativo

β e γ sono i coefficienti di pesatura rispettivamente dei centroidi C_p e C_n. β e γ sono dei numeri reali compresi tra 0 e 1 e tali che β + γ = 1

In altre parole, lo score può essere penalizzato dalle preferenze negative e γ è il coefficiente di penalizzazione.

Nel nostro caso, abbiamo preso β = 0.8 e γ = 0.2