



Università degli Studi di Salerno

Documentazione progetto

Fondamenti di Intelligenza Artificiale
a.a. 2022/2023 prof. Fabio Palomba

Repository GitHub

<i>Autore</i>	<i>Matricola</i>
Costante Luigina	0512110457
Lo Conte Simona	0512110922
Napolillo Marta	0512109836

Indice

1	Introduzione	2
2	Business Understanding	3
2.1	Obiettivi di business	3
2.2	PEAS	3
2.3	Proprietà dell'ambiente	3
2.4	Analisi del problema	3
3	Data Understanding	5
3.1	Acquisizione dei dataset	5
3.2	Analisi dei dataset	5
4	Data Preparation	6
4.1	Data Cleaning	6
4.2	Feature Scaling	6
4.3	Feature Selection	7
4.4	Data Balancing	8
5	Modeling	9
5.1	Scelta dell'algoritmo da utilizzare	9
5.1.1	K-Means	9
5.1.2	DBScan	9
6	Evaluation	11
6.1	K-Means Evaluation	11
6.1.1	Elbow point	11
6.1.2	Silhouette coefficient	12
6.2	DB-Scan Evaluation	12
6.2.1	Silhouette Score	12
7	Integrazione con il sistema	14
8	Deployment	16
9	Glossario	17

1 Introduzione

Con l'avvento del digitale è in costante crescita il numero di persone - di ogni fascia di età - che scelgono di guardare un film nel proprio tempo libero. Di conseguenza, aumenta la voglia di scoprire sempre nuovi contenuti in base alle proprie preferenze e rimanere costantemente aggiornati sulle ultime novità. La maggior parte degli utenti, però, è spesso indecisa su quale film scegliere e passa gran parte del tempo a navigare tra i contenuti disponibili. A tal proposito *iLike*, oltre a realizzare una piattaforma unificata che consente di recensire contenuti, offre la possibilità di interagire con un Conversational Agent, il cui ruolo principale è fornire all'utente indicazioni circa possibili film da vedere, sulla base delle sue preferenze. Il Conversational Agent è di fondamentale importanza poichè permette agli utenti di ottenere consigli attraverso una conversazione simil-umana.

2 Business Understanding

2.1 Obiettivi di business

L'obiettivo principale di *iLike* è la realizzazione di un Conversational Agent, che permetterà all'utente di interagire per richiedere consigli su nuovi film da guardare. Lo scopo è quello di consentire anche agli utenti più indecisi di ricevere consigli personalizzati su film che potrebbero essere di interesse personale. Tutto ciò sarà possibile analizzando le diverse liste degli iscritti, nelle quali quest'ultimi potranno decidere di inserire contenuti diversi in base alle preferenze personali.

2.2 PEAS

Specifica PEAS dell'ambiente.

Performance	Capacità dell'agente di suggerire all'utente film che rispecchiano i suoi gusti.
Environment	L'ambiente in cui l'agente opera è rappresentato da iLike, un'applicazione in cui gli utenti possono scrivere recensioni ed esprimere preferenze sui contenuti che si trovano all'interno di essa.
Actuators	Risposta del Conversational Agent.
Sensors	Utterances (messaggi in linguaggio naturale dati in input al CA da un utente umano).

2.3 Proprietà dell'ambiente

L'ambiente possiede le seguenti proprietà:


- **Completamente osservabile:** l'agente ha accesso all'elenco dei contenuti presenti nell'applicazione e alle preferenze degli utenti in qualsiasi momento;
- **Stocastico:** lo stato dell'ambiente varia indipendentemente dall'azione intrapresa dall'agente;
- **Sequenziale:** le decisioni prese dall'agente dipendono dalle azioni passate dell'utente;
- **Statico:** nel momento in cui l'agente sta elaborando la sua decisione l'utente non può modificare le sue preferenze;
- **Discreto:** i suggerimenti dati dall'agente dipendono dalla combinazione di contenuti preferiti di cui l'utente dispone o da un genere stabilito ed esistono un numero limitato di possibili combinazioni;
- **Singolo-agente:** esiste un unico agente che opera nell'ambiente.

2.4 Analisi del problema

Il problema che l'agente intelligente dovrà risolvere consiste nel suggerire film da vedere in base ai contenuti presenti nei dataset dell'applicazione e soprattutto in merito alle preferenze espresse dagli utenti (in base ai contenuti della lista preferiti dell'iscritto).

Il problema in esame può essere risolto con un algoritmo di apprendimento, in quanto consiste nel migliorare l'esecuzione di un task (T = fornire suggerimenti personalizzati) rispetto ad una misura di prestazione (P = numero di suggerimenti accettati dall'utente) e sulla base dell'esperienza (E = dataset di film). Inoltre l'algoritmo di apprendimento in questione è di tipo non supervisionato in quanto l'agente dovrà essere capace di apprendere il valore reale della variabile dipendente sulla base delle conoscenze di cui dispone. Nello specifico il problema in esame può essere risolto tramite l'utilizzo di un algoritmo di clustering. Una volta che l'utente ha espresso le sue preferenze riguardanti contenuti presenti nell'applicazione, l'algoritmo creerà, in base ad una misura di similarità (definita in fase di modelling), dei cluster contenenti film che

hanno un certo grado di omogeneità. Procederà quindi a consigliarne nuovi in base alla clusterizzazione effettuata.

I suggerimenti verranno dati solo qualora l'utente ne faccia richiesta ed il tutto avviene in maniera automatica tramite l'utilizzo di un Conversational Agent. L'iscritto potrà interagire con quest'ultimo ogni qualvolta lo ritenga opportuno, mediante  presente nel footer delle varie activity dell'applicazione. Una volta premuto tale pulsante l'utente potrà scrivere messaggi in base ai quali riceverà una risposta opportuna. Nel momento in cui richiederà un consiglio per un film, il Conversational Agent chiederà l'algoritmo che si vuole utilizzare e successivamente provvederà ad analizzare l'insieme di cluster di cui si dispone e "calcolare" la relativa risposta.

Si è ritenuto non opportuno utilizzare un algoritmo di classificazione in quanto non disponiamo di un'insieme di osservazioni per cui la variabile target è nota, di conseguenza non sarebbe possibile stimare il valore di una nuova variabile categorica. Un'analoga considerazione è stata fatta in merito all'utilizzo di un algoritmo di regressione, anche se in questo caso occorreva stimare il valore di una variabile numerica.

3 Data Understanding

3.1 Acquisizione dei dataset

Durante la scelta dei dati da fornire al machine learning le possibili scelte da seguire erano sostanzialmente due:

- Creare un dataset contenente gli utenti di iLike ed analizzare il loro comportamento, al fine di creare cluster di utenti i quali hanno preferenze simili;
- Cercare dataset con le informazioni relative ai film e creare cluster di film.

I pro e i contro delle alternative sopra elencate sono:

- La disponibilità di dati era maggiore nei dataset già esistenti;
- Ogni utente ha gusti differenti, quindi la similarità tra utenti, rappresentata come il numero di contenuti uguali appartenente alle proprie liste, può non essere sempre veritiera;
- Individuare dataset con un numero ottimale di istanze e le giuste informazioni sui film richiede un'accurata analisi.

Al seguito di un trade-off tra le due alternative abbiamo preferito utilizzare dataset già esistenti relativi ai film, poichè la disponibilità di dati e la giusta similarità di elementi in un cluster agevola le prestazioni dell'algoritmo di machine learning.

3.2 Analisi dei dataset

Il dataset utilizzato riguardo i Film è reperibile sulla piattaforma Kaggle.

4 Data Preparation

In fase di Data Understanding abbiamo scelto l'utilizzo di un dataset già esistente, quindi è opportuno effettuare Data Preparation. Lo scopo di questa fase è preparare i dati al fine di passarli correttamente all'algoritmo di Machine Learning.

4.1 Data Cleaning

Lo scopo del Data Cleaning è gestire dati rumorosi e/o nulli. A tal proposito, è stata effettuata inizialmente un'analisi dei dati al fine di evidenziare dati rumorosi. Si è notato che le colonne 'voto_medio' e 'voti_totali' rispecchiano esattamente la media e la somma delle colonne 'voto_critica' e 'voto_pubblico'. Dunque abbiamo deciso di eliminare le colonne 'voto_critica' e 'voto_pubblico' e non eliminare le colonne 'voto_medio' e 'voti_totali', rimandando tale scelta nel Feature Selection, per valutare quale delle due ha una maggiore correlazione tra le altre variabili.

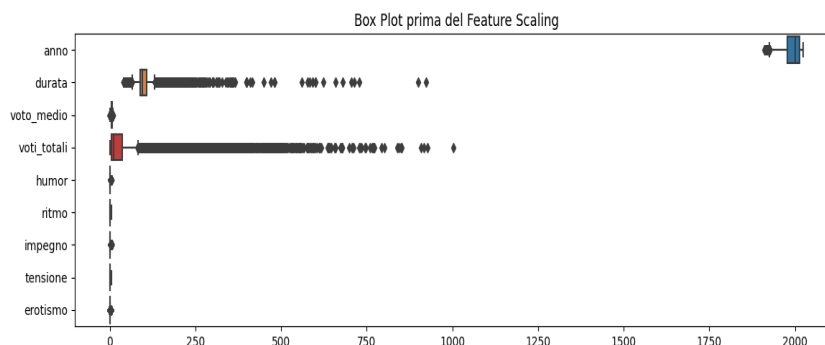
In seguito è riportata una tabella contenente il nome della colonna, il numero di elementi mancanti e la scelta di Data Imputation effettuata.

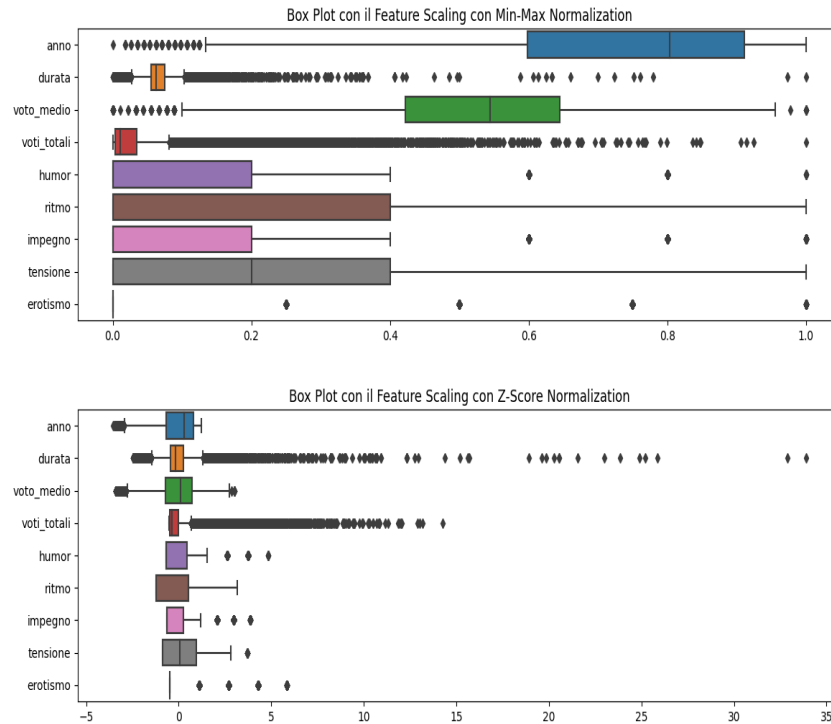
Nome Colonne	Numero Elementi	Scelta Data Imputation
genere	95	Eliminazione Riga
paese	11	Eliminazione Riga
registi	33	Eliminazione Riga
attori	2.027	Eliminazione Colonna
descrizione	1.449	Eliminazione Riga
note	21.628	Eliminazione Colonna

Le scelte sopra riportate sono state effettuate per evitare di eliminare un eccessivo numero di righe. Si fa eccezione per la colonna descrizione in quanto tale colonna è necessaria all'interno dell'applicazione iLike, per altre funzionalità non riguardanti il modulo di Intelligenza Artificiale.

4.2 Feature Scaling

Lo scopo del Feature Scaling è normalizzare i valori numerici del dataset, portandoli tutti nello stesso range. I grafici box plot sottoriportati confrontano i valori iniziali e i valori ottenuti prima con la tecnica del Min-Max Normalization e in seguito con lo Z-Score Normalization.



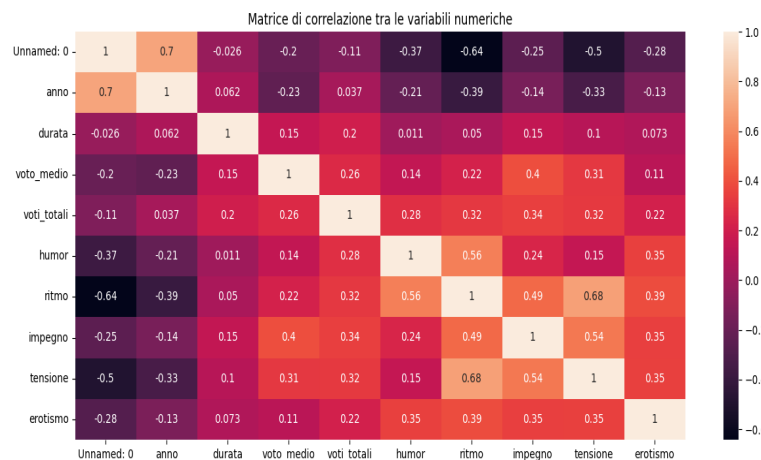


Come evidenziato nei grafici, la tecnica dello Z-Score Normalization è meno tendente ai valori estremi. Dunque, abbiamo preferito tale tecnica al fine di effettuare una corretta normalizzazione dei dati.

4.3 Feature Selection

Lo scopo del Feature Selection è selezionare le caratteristiche più correlate al problema in esame. Le feature dal nostro dataset sono di tipo string o int. Le feature di tipo string rappresentano solo informazioni descrittive dei film. Per tal motivo, le feature individuate sono di tipo int. In questa fase, le tecniche utilizzate sono:

- Eliminazione di feature con bassa varianza. Con questa tecnica vengono eliminate le feature con varianza uguale a zero. Nel nostro caso, nessuna feature a noi disponibile ha varianza uguale a zero;
- Eliminazione univariata di feature. Con questa tecnica si analizzano le correlazioni tra le variabili. Il grafico sotto riportato offre un'analisi riassuntiva delle correlazioni con le variabili.



Come si può notare la feature 'anno' è inversamente correlata con quasi tutte le variabili, e la variabile 'voto_medio' ha meno correlazione tra le variabili rispetto a 'voto_totale'. Dunque abbiamo eliminato le feature 'anno' e 'voto_medio'.

4.4 Data Balancing

Lo scopo del Data Balancing è bilanciare i dati, al fine di avere lo stesso numero di istanze in tutte le classi. Nel nostro dataset le classi possono essere rappresentate dalla feature 'genere', ma avendo optato in fase di Business Understanding per un algoritmo di apprendimento non supervisionato, tale bilanciamento non è necessario in fase di Data Preparation.

5 Modeling

5.1 Scelta dell'algoritmo da utilizzare

Esiste un gran numero di algoritmi di clustering, pertanto bisogna effettuare una scelta dell'algoritmo da considerare per effettuare il clustering tra gli elementi. Prima di effettuare la scelta dell'algoritmo da utilizzare abbiamo fatto un confronto tra due algoritmi di clustering molto usati, ossia il K-Means e il DBScan.

5.1.1 K-Means

Per quanto riguarda l'algoritmo di clustering K-Means, questo viene realizzato tramite la funzione `KMeans()` della libreria `sklearn.cluster`. Tale funzione permette di definire alcuni parametri di input, pertanto abbiamo valutato l'algoritmo di K-Means con diversi valori per due di questi parametri.

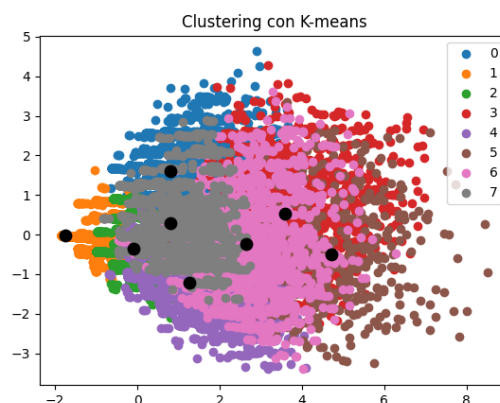
Per il parametro che permette di definire il metodo per l'inizializzazione abbiamo valutato due opzioni:

- `random`, che sceglie n cluster di osservazioni (righe) a caso dai dati per i centroidi iniziali;
- `k-means++`, che garantisce un'inizializzazione più intelligente dei centroidi e migliora la qualità del clustering.

Dopo aver valutato i due metodi di inizializzazione sul nostro dataset, abbiamo scelto il metodo `k-means++`, in quanto più adatto a dividere il nostro dataset in cluster distinti. Per quanto riguarda il parametro che specifica l'algoritmo che utilizza il K-Means abbiamo valutato le seguenti opzioni:

- `lloyd`, che è quello utilizzato di default;
- `elkan`, che usa la disuguaglianza triangolare.

A seguito di un confronto abbiamo preferito utilizzare l'algoritmo di `elkan`, in quanto il numero di iterazioni effettuate per arrivare ad ottenere i cluster era nettamente minore al numero di iterazioni effettuate scegliendo l'algoritmo di `lloyd`. Il grafico riportato sotto mostra la divisione del dataset in cluster, effettuata utilizzando l'algoritmo di K-Means.

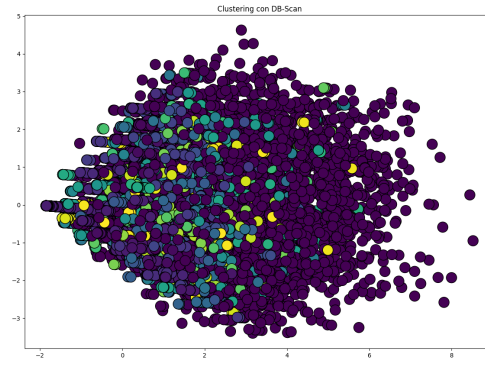


5.1.2 DBScan

L'algoritmo di clustering DBScan viene realizzato tramite la funzione `DBSCAN()` della libreria `sklearn.cluster`. Anche questa funzione permette di definire alcuni parametri di input, che sono:

- `eps`, che definisce un intorno circolare di ciascun punto dai suoi vicini;
- `min samples`, che stabilisce il numero minimo di punti per considerare un intorno di un punto denso.

La scelta dei valori effettuata è stata `eps=1` e `min samples=5`. Di seguito è riportato il clustering ottenuto con DBScan con i valori dei parametri definiti sopra.



Tutte le considerazioni e valutazioni relative ai valori dei parametri scelti saranno effettuate nella fase successiva che è quella di Evaluation.

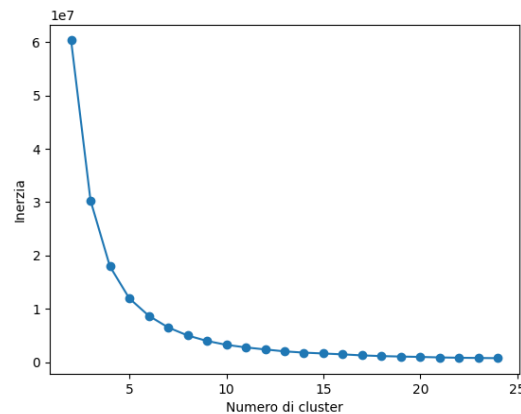
6 Evaluation

6.1 K-Means Evaluation

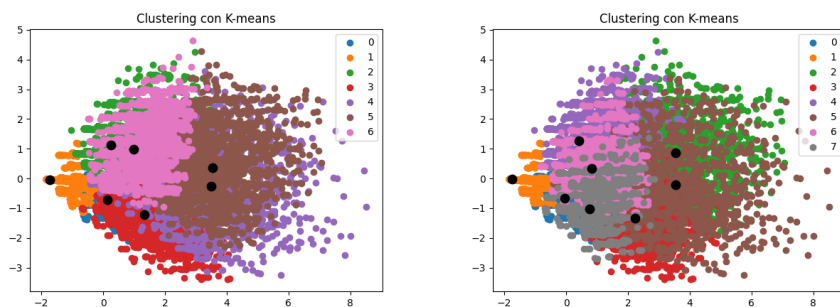
6.1.1 Elbow point

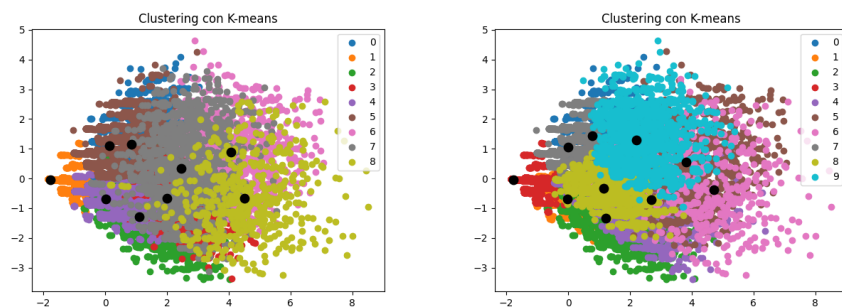
Il primo metodo di valutazione dei risultati di cui si intende usufruire è l'elbow point, con l'ausilio della libreria Python *scikit-learn*. Dopo aver caricato i valori presenti nel dataset e inizializzato una nuova lista (*inertias[]*), abbiamo provveduto ad eseguire il k-means su un numero crescente di cluster, memorizzando ad ogni iterazione il valore di inerzia prodotto. Quest'ultimo è una misura della qualità dei cluster generati, in quanto si riferisce alla misura di somiglianza ottenuta all'interno dei cluster. Più bassa è l'inerzia, più simili sono i punti all'interno di un cluster. L'inerzia di un cluster è calcolata come la somma delle distanze quadratiche tra ciascun punto del cluster e il centroide del cluster stesso. Il punto in cui si osserva una diminuzione significativa dell'inerzia indica il numero ottimale di cluster, ovviamente si cerca di scegliere il numero di cluster che massimizzi la differenziazione tra cluster e minimizzi l'inerzia all'interno dei cluster stessi.

Il grafico ottenuto è il seguente:



Come è possibile notare leggendo il grafico, si nota una diminuzione dell'inerzia con l'utilizzo di un k da 7 in su. Sebbene l'errore venga minimizzato quando i cluster aumentano, avere un numero eccessivo di cluster implica avere tanti gruppi formati da pochi elementi. Valutando i grafici ottenuti con l'esecuzione dell'algoritmo k-means al variare della variabile k da 7 a 10 (mostrati di seguito), abbiamo notato che, al crescere del valore della variabile k, alcuni cluster si mantengono abbastanza stabili mentre altri tendono ad essere fin troppo piccoli. Per questo motivo, risultano migliori i cluster ottenuti con l'utilizzo della variabile k pari ad 8 o a 9. Abbiamo ritenuto opportuno demandare tale scelta alla fase successiva, con l'ausilio del silhouette coefficient.





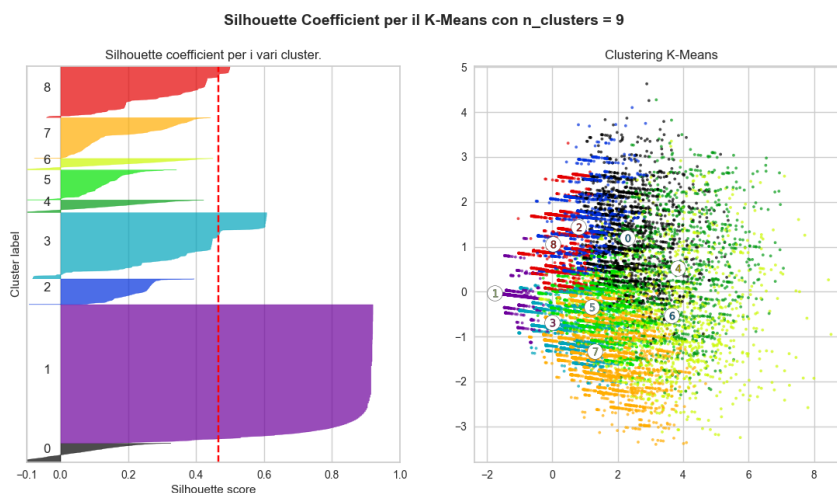
6.1.2 Silhouette coefficient

Un altro metodo di valutazione dei risultati utilizzato è il silhouette coefficient, che permette di calcolare quanto i dati siano ben disposti nei cluster generati. Dal metodo di valutazione dell'elbow point sopra citato, emerge che il numero di cluster ottimale è un valore per $k=7$ in su. Pertanto abbiamo valutato il silhouette coefficient eseguendo il K-Means con un numero di cluster che variava nel range da 7 a 10. Quindi, utilizzando la libreria di Python sklearn, per ogni valore di $7 \leq k \leq 10$, abbiamo eseguito il k-means sul nostro dataset. Successivamente, tramite la funzione `silhouette_score()`, abbiamo calcolato il silhouette score, ossia una misura della coesione e separazione tra i dati. Siamo passati poi a calcolare il silhouette score per ogni campione dei cluster utilizzando l'apposita funzione `silhouette_samples()`.

Vengono riportati di seguito i valori di silhouette score relativi al numero di cluster indicato

```
n_clusters = 7 , Silhouette Score: 0.4543560055647245
n_clusters = 8 , Silhouette Score: 0.4546889416677923
n_clusters = 9 , Silhouette Score: 0.4650567318077946
n_clusters = 10 , Silhouette Score: 0.4404451887726646
```

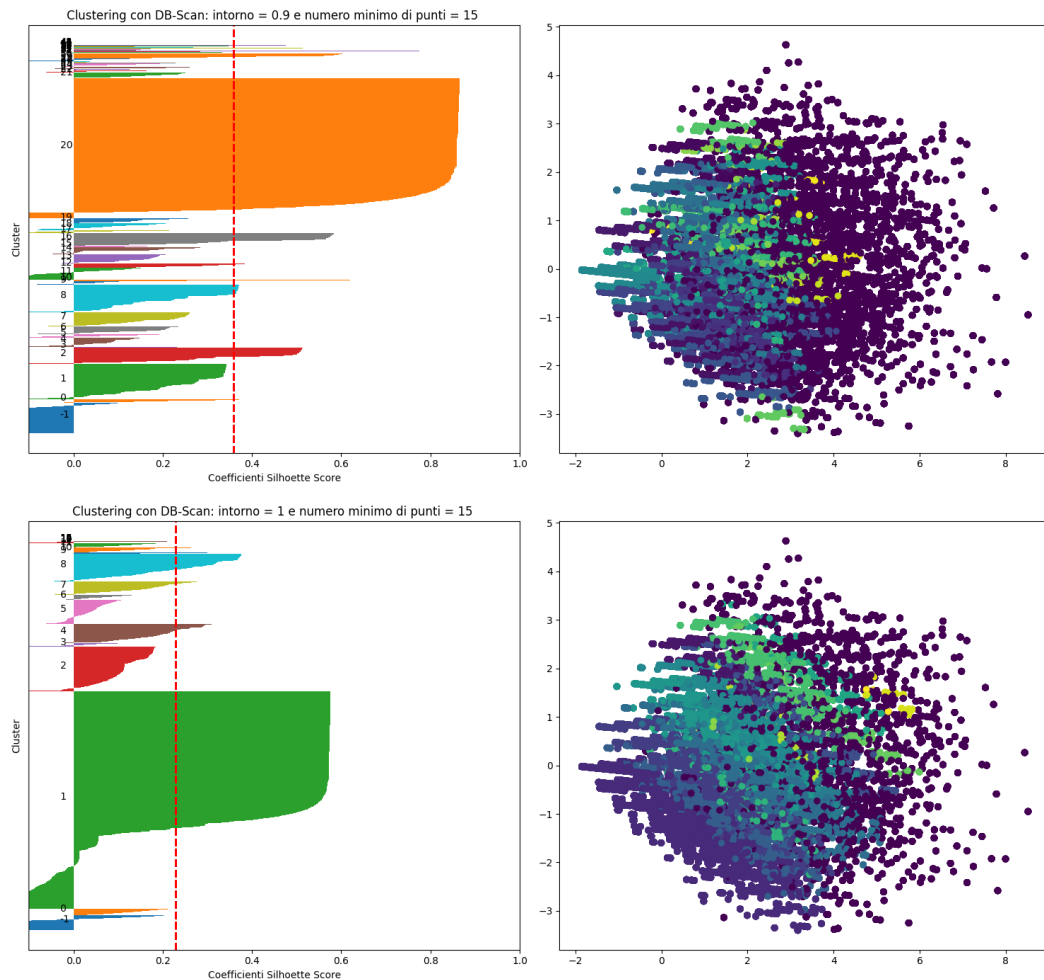
Come mostrato nell'immagine soprastante, il valore di k ottimale è $k=9$. Quindi, considerando anche le valutazioni prodotte con il metodo di Elbow point, il clustering con l'algoritmo k-means è ottimale impostando il parametro `n_cluster = 9`. Questo grafico mostra, per `n_cluster=9`, il valore del silhouette score e inoltre la divisione dei dati nei vari cluster ottenuti.



6.2 DB-Scan Evaluation

6.2.1 Silhouette Score

Il metodo di valutazione, utilizzato per valutare i risultati prodotti dall'algoritmo DB-Scan, è il Silhouette Score fornito dalla libreria Python *scikit-learn*. Dopo aver eseguito il DB-Scan con diversi valori sia per l'intorno circolare, sia per il minimo numero di punti per considerare un intorno denso, le esecuzioni più rilevanti sono ripostate in seguito. I grafici sottoriportati evidenziano il valore del silhouette score e la divisione dei dati nei vari cluster ottenuti. La linea verticale tratteggiata è rappresenta la media delle Silhouette Score calcolate in precenza.



Come si può notare nessun grafico risulta essere un clustering ottimale. Nel secondo grafico si può notare una suddivisione sub-ottima in quanto:

- il numero di cluster è minore nel secondo grafico, ciò consente una migliore suddivisione dei cluster;
- la media delle Silhouette Score è simile tra le due;
- il numero di outline (rappresentato dal cluster -1) è minore nel secondo grafico.

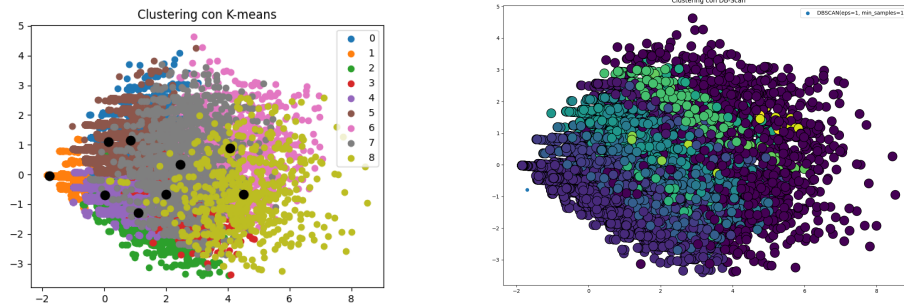
Gli altri risultati, non riportati, hanno gli stessi risultati dei precedenti con una media inferiore e una suddivisione non ideale.

Di conseguenza, i valori sub-ottimi sono rappresentati da $\text{eps} = 1$ e $\text{min samples} = 15$.

Si può notare che l'algoritmo K-Means effettua un clustering migliore, ma il giudizio sul clustering usato sarà offerto dagli utenti finali, pertanto la scelta dell'algoritmo migliore tra il K-Means e il DBScan sarà rimandata in fase di utilizzo. Questo sarà possibile in quanto il Conversational Agent permetterà ai beta tester di scegliere l'algoritmo di clustering da utilizzare, chiedendo direttamente di effettuare tale scelta prima di iniziare il clustering.

7 Integrazione con il sistema

Dopo aver scelto l'algoritmo di clustering da utilizzare e le conseguenti valutazioni su di esso, va definito come integrarlo ed impiegarlo all'interno del nostro sistema software. Gli algoritmi adoperati ci consentono di calcolare, a partire dal dataset di film di cui si dispone, un certo numero di cluster contenenti film con caratteristiche simili, come visibile anche dai grafici seguenti.



Una volta ottenuto questo modello, l'obiettivo da raggiungere era partire da una lista di film preferiti di un iscritto e consigliarne nuovi che potessero rispecchiare i suoi gusti e le sue preferenze espresse tramite l'applicazione. Ad esempio, ipotizziamo che la lista di preferiti dell'utente contenga i seguenti film:

- Storie irlandesi
- Diva!
- La grande bellezza
- Adults in the Room
- Ciao amore, vado a combattere
- Il piccolo ladro
- Polvere di stelle
- Home
- 120 battiti al minuto
- Tutti fratelli nel West... per parte di padre

Sulla base degli elementi della lista dell'utente, abbiamo costruito un dizionario formato da coppie chiave-valore rispettivamente contenenti il titolo e l'anno di un film (chiave) e il cluster di appartenenza (valore). Successivamente abbiamo individuato il cluster contenente il maggior numero di elementi della lista di preferiti dell'iscritto (*tableCluster*) e salvato all'interno di una nuova lista (*newList*) i film che appartenevano a questa intersezione.

```
NUMERO DI FILM DELLA LISTA PREFERITI DELL'ISCRITTO PER OGNI CLUSTER  
[1, 5, 2, 0, 0, 1, 0, 1]  
VALORE MASSIMO CONTENUTO IN list: 5 --> cluster 1
```

Abbiamo poi calcolato le distanze metriche medie tra gli elementi di *newList* e tutti i restanti elementi del cluster *tableCluster*.

```
DISTANZE CALCOLATE  
[[2.55964593 0.9205035 0.90439705 ... 0.89715624 0.89715624 0.89628666]  
 [2.56603778 0.93813074 0.92233217 ... 0.91523326 0.91523326 0.91438087]  
 [2.55745048 0.91438087 0.89816464 ... 0.89087317 0.89087317 0.88999746]  
 [2.55745048 0.91438087 0.89816464 ... 0.89087317 0.89087317 0.88999746]  
 [2.69358084 1.24535825 1.23350091 ... 1.22820186 1.22820186 1.22756681]]  
DISTANZA MINIMA --> (0.9855702164588142, 2)
```

Ottenuta questa informazione abbiamo individuato l'elemento con distanza metrica inferiore. Quest'ultimo rappresenta proprio il film più vicino alle preferenze espresse dall'utente e quindi il film che verrà consigliato all'iscritto tramite il Conversational Agent.

```
FILM SELEZIONATO --> Alba di fuoco
voti_totali          -0.363923
humor                -0.661062
ritmo                -1.220368
impegno              -0.625122
tensione              -0.865012
erotismo              -0.485327
titolo_italiano      Alba di fuoco
```






In merito all'addestramento del Conversational Agent abbiamo usufruito della libreria *Natural Language Toolkit* offerta dal linguaggio di programmazione *Python*. Abbiamo inizializzato un array (*pairs*) di coppie domande-risposte, in particolare abbiamo ipotizzato una serie di pattern corrispondenti ad espressioni regolari che esprimono statement o domande che l'utente può inviare al sistema ed in base ad essi abbiamo definito delle risposte che il nostro CA dovrà fornire all'utente.

In questo modo andiamo ad istruire il bot su come rispondere alle domande o alle frasi dell'utente. Per diverse domande sono state previste risposte multiple, in modo da rendere il bot meno monotono. *Pairs* contiene domande e risposte specifiche per la nostra applicazione e per lo scopo principale della creazione del bot, ovvero il suggerimento di film in base ai gusti degli iscritti. Nel momento in cui l'iscritto digita un messaggio e lo invia, se questo avrà un match con uno dei pattern previsti allora verrà visualizzata la risposta appropriata, altrimenti verrà visualizzato un messaggio che invita l'iscritto ad esprimere diversamente la sua richiesta.

Grazie all'utilizzo della libreria *Natural Language Toolkit* i vari passaggi previsti dal Natural Language Processing sono eseguiti in automatico e risultano per questo motivo "nascosti".

Nel caso specifico in cui l'utente richiede un consiglio la risposta verrà data sulla base del risultato della chiamata all'algoritmo di clustering per dare un suggerimento mirato, basato sui gusti dell'utente in questione espressi tramite l'aggiunta dei film presenti nell'applicazione ad una lista di preferiti.

8 Deployment

Sarà possibile usufruire del Conversational Agent premendo sull'apposito pulsante  presente nel footer dell'applicazione. All'inizio l'utente visualizza un messaggio di benvenuto del bot e successivamente è libero di scrivere le sue richieste nell'apposita area visibile in fondo alla schermata. Una volta scritto il messaggio, l'utente dovrà premere il pulsante   e dopo pochi secondi la risposta del bot sarà visibile. A questo punto sarà possibile inviare un ulteriore messaggio a cui il bot risponderà e così via fino a quando l'utente non deciderà di uscire dalla pagina in questione tramite l'apposito pulsante   o tramite il tasto del dispositivo mobile.

Nel caso in cui, l'utente non ha liste a lui associate oppure ha liste ma non ha contenuti all'interno delle liste, il Conversational Agent inviterà l'utente a creare ed aggiungere nelle liste i suoi film preferiti.

9 Glossario

Conversational Agent/CA	Bot che interpreta e risponde alle dichiarazioni fatte dagli utenti in un linguaggio naturale, attraverso la generazione di una conversazione simil-umana.
Lista di contenuti	Sottoinsieme di contenuti offerti da iLike, scelti dagli utenti secondo i loro gusti e inseriti nelle proprie liste disponibili sul proprio profilo personale.
Cluster	Sottoinsieme di contenuti con caratteristiche simili.
Machine Learning	Branca dell'Intelligenza Artificiale che include tutti gli algoritmi che possano imparare dai dati e sulla base di questi fare previsioni.
Data Imputation	Insieme di tecniche che permettono di stimare il valore di dati mancanti sulla base dei dati disponibili oppure mitigare il problema dei dati mancanti.
Blox Plot	Rappresentazione grafica utilizzata per descrivere la distribuzione di un campione tramite la media e deviazione standard.
Utente	Iscritto, cioè persona registrata ad iLike.