Titolo: Addestramento del Modello di Raccomandazione TransE sul Dataset LFM1B Campionato con Augmentation Demografica degli Artisti

Modello di Raccomandazione TransE:

Il modello di raccomandazione TransE si basa su una rappresentazione vettoriale degli elementi del dataset, con l'obiettivo di catturare le relazioni semantiche tra di essi. La sua architettura si compone di un livello di embedding per gli utenti e un livello di embedding per i brani musicali. Durante l'addestramento, il modello apprende a rappresentare gli elementi in modo che le relazioni tra di essi siano rispettate nello spazio vettoriale.

Knowledge Graph:

I knowledge graph rappresentano una potente struttura dati che organizza le conoscenze in forma di grafo, dove i nodi rappresentano entità del mondo reale e gli archi denotano le relazioni tra di esse. Questa rappresentazione consente di catturare e modellare in modo esaustivo le complesse interconnessioni tra le informazioni, consentendo ai sistemi di elaborare le conoscenze in modo più semantico e contestuale. Grazie alla loro flessibilità e scalabilità, i knowledge graph sono ampiamente utilizzati in una varietà di settori, tra cui il web semantico, l'intelligenza artificiale e i motori di ricerca, dove contribuiscono a migliorare la comprensione e l'organizzazione delle informazioni. Inoltre, i knowledge graph giocano un ruolo cruciale nel supportare applicazioni avanzate come i sistemi di raccomandazione e l'elaborazione del linguaggio naturale, consentendo di ottenere risultati più accurati e contestualizzati. Per questo esperimento, il KG che è stato impiegato, è stato quello adattato al dataset LFM1M, che contiene svariate relazioni identificate univocamente da un id. Tutti i file necessari al suo utilizzo sono situati all'interno di data/lfm1m.

Dataset LFM1B campionato:

Il dataset LFM1B rappresenta una raccolta significativa di dati di ascolto musicale, comprendente milioni di tracce musicali, gli utenti che le hanno ascoltate e gli artisti che le hanno prodotte. Ogni campione del dataset contiene informazioni cruciali, tra cui l'identificatore dell'utente, l'identificatore del brano e il timestamp dell'interazione. La diversità di generi musicali e la vastità del dataset lo rendono un ambiente di test ideale per modelli di raccomandazione, offrendo sfide e opportunità uniche per l'addestramento e la valutazione.
Per questo esperimento, è stato utilizzato un campione specifico di questo dataset, ottenuto prelevando dall'originale le tracce contenute anche nel KG di LFM1M.

Per farlo, è stato impiegato lo script *extract_sample.ipynb*, che scansiona dal dataset originale tutte le tracce, dopodiché salva in un file in output solo le tracce con i track id coincidenti.

extract_sample.ipynb | augmentation.ipynb | graphs_metrics.ipynb | products.csv

Code

```
[ ]: !pip install pandas
```

```
[3]: import pandas as pd

# Funzione che legge il file che possiede informazioni del dataset delle tracce quali user id, track id e timestamp e
# salva un campione di tracce uguale a quelle di LFM1M, in modo da usufruire del suo kg
def leggi_e_salva_file(percorso_file_input, percorso_file_output, nrows):
    # Leggi il file di input
    df = pd.read_csv(percorso_file_input, delimiter='\t', header=None, names=['eid', 'pid', 'name', 'entity'])

    # Carica i nomi delle tracce da i2kg_map.txt
    tracks_info = pd.read_csv('data/lfm1m/preprocessed/i2kg_map.txt', sep='\t', header=None, names=['users_id', 'tracks_id', 'timestamp'])

    # Seleziona solo le righe in cui l'id della traccia coincide con gli id in tracks_id, per avere come campione le stesse tracce del kg
    selected_rows = df[df['pid'].isin(tracks_info[1])]

    # Salva le prime nrows righe nel file di output
    selected_rows.head(nrows).to_csv(percorso_file_output, index=False)

# Applicazione delle funzioni definite
if __name__ == "__main__":
    percorso_file_input = 'data/lfm1b/tracks.inter'
    percorso_file_output = "data/lfm1m/preprocessed/products.txt"
    nrows = 12492 # Valore che coincide col numero di tuple del campione desiderato
    leggi_e_salva_file(percorso_file_input, percorso_file_output, nrows)

# Fatto ciò, potremo eseguire gli esperimenti sul dataset modellato in base al KG di riferimento.
```

Augmentation Demografica degli Artisti:

L'augmentation demografica del dataset è stata eseguita integrando informazioni demografiche specifiche degli artisti. Le informazioni, quali genere, data di nascita e luogo di nascita, sono state estratte da Wikidata. Questo arricchimento consente al modello di raccomandazione di comprendere meglio il contesto demografico degli artisti, consentendo raccomandazioni più personalizzate basate su caratteristiche individuali degli artisti stessi. Le motivazioni dietro questa scelta risiedono nella volontà di migliorare la precisione del modello e di fornire esperienze di raccomandazione più raffinate agli utenti.
Per effettuarla, è stato utilizzato lo script *augmentation.ipynb*.

extract_sample.ipynb | augmentation.ipynb | graphs_metrics.ipynb | products.csv

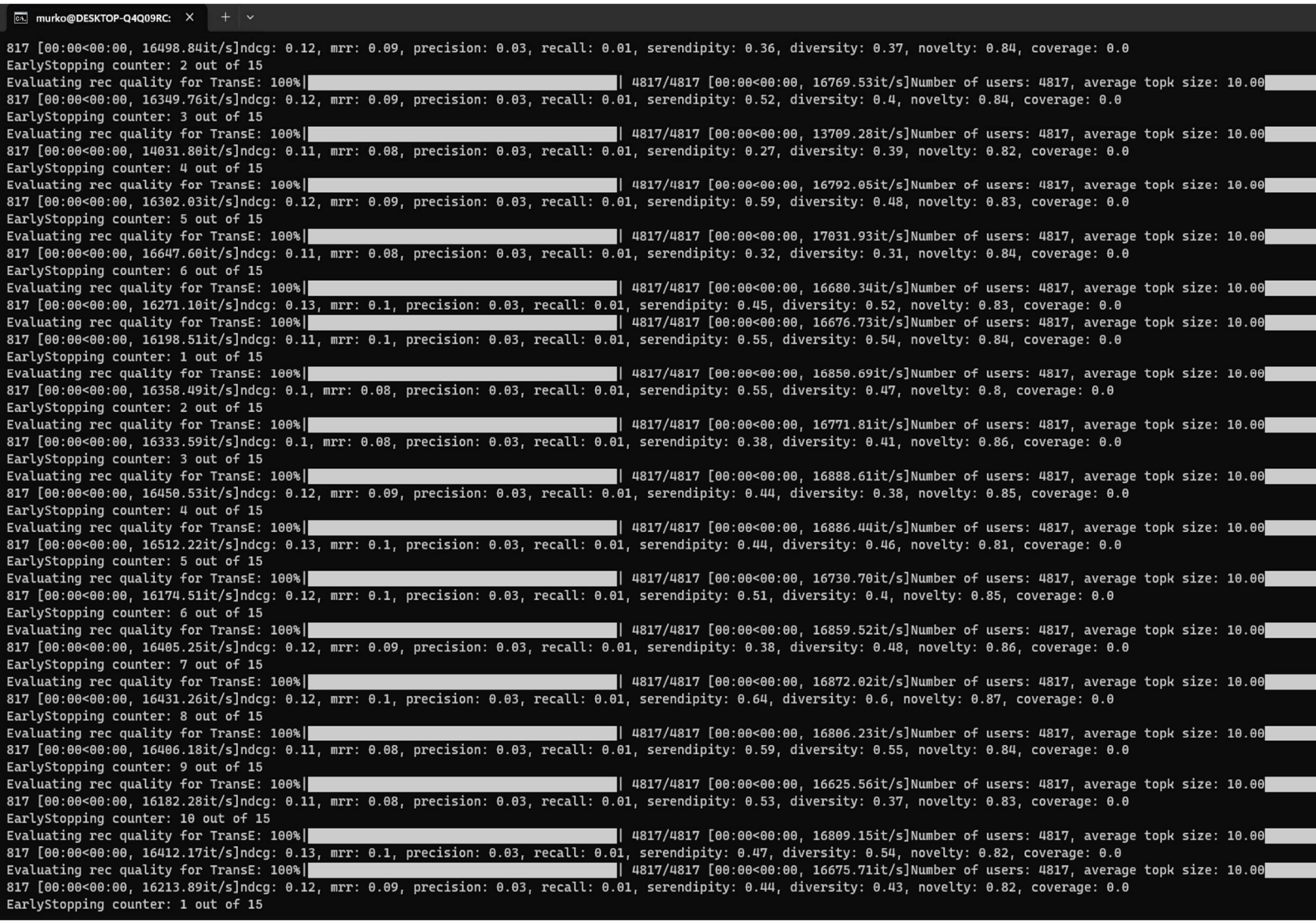
Code

```
•[3]: # Con questo script, si effettuerà una query SPARQL per eseguire l'augmentation coi dati demografici quali genere
      # (mappato con tre possibili valori), data di nascita e luogo di nascita
      # dell'artista. Si lasciano vuoti i campi in caso di dati non trovati
      # (avviene nel caso l'artista sia un gruppo musicale e non una singola persona). Infine, si produce in
      # output un file con le stesse colonne del file in input, con le tre colonne aggiuntive dei dati demografici.

      import pandas as pd
      import requests
      import re
      import time
      from json.decoder import JSONDecodeError

      # Mappatura del genere coi possibili valori su Wikidata
      genre_mapping = {
          'Q6581097': 'Male',
          'Q6581072': 'Female',
          'Q1097630': 'Non specificato'
      }

      # Funzione che si dedicherà ad effettuare la query SPARQL su Wikidata per cercare le informazioni di genere, data di nascita e luogo di nascita per ogni artista
      def get_demographic_data(artist_name):
          # Controlla se artist_name è una stringa, altrimenti lascia i campi vuoti
          if not isinstance(artist_name, str):
              return {
                  'genre': None,
                  'birth_date': None,
                  'birth_place': None
              }

          # Gestisce il caso di nomi contenenti simboli che non permettono di esser trovati su Wikidata.
          cleaned_artist_name = re.sub(r'[^\w\s&,]+', '', artist_name)
          if '&' in cleaned_artist_name:
              cleaned_artist_name = cleaned_artist_name.split('&')[0].strip()
          elif 'and' in cleaned_artist_name:
              cleaned_artist_name = cleaned_artist_name.split('and')[0].strip()

          sparql_query = f"""
          SELECT ?artist ?genre ?birthDate ?birthPlaceLabel
          WHERE {{
            ?artist rdfs:label "{cleaned_artist_name}"@en.
            ?artist wdt:P21 ?genre;
                    wdt:P569 ?birthDate;
                    wdt:P19 ?birthPlace.

            SERVICE wikibase:label {{ bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }}
          }}
          LIMIT 1
          """

          # Verifica se ci sono risultati nella lista. Wikidata restituisce i risultati delle query in formato JSON. Quando si effettua una query, i risultati sono organizzati in una struttura gerarchica,
          # dove i dati rilevanti sono contenuti all'interno di varie chiavi e sottochiavi. Per esempio, supponiamo che la query SPARQL abbia restituito informazioni su un artista, quali il genere,
          # la data di nascita e il luogo di nascita. Queste informazioni saranno contenute all'interno di data['results']['bindings']. Si assegnano così ai campi delle colonne aggiuntive nel file in output

          if data and 'results' in data and 'bindings' in data['results'] and data['results']['bindings']:
              genre_url = data['results']['bindings'][0]['genre']['value'] if 'genre' in data['results']['bindings'][0] else None
              birth_date = data['results']['bindings'][0]['birthDate']['value'] if 'birthDate' in data['results']['bindings'][0] else None
              birth_place = data['results']['bindings'][0]['birthPlaceLabel']['value'] if 'birthPlaceLabel' in data['results']['bindings'][0] else None
              genre_id = genre_url.split('/')[-1] if genre_url else None
              genre = genre_mapping.get(genre_id, None)

              return {
                  'genre': genre,
                  'birth_date': birth_date,
                  'birth_place': birth_place
              }
          else:
              return {
                  'genre': None,
                  'birth_date': None,
                  'birth_place': None
              }

      # Caricamento del campione iniziale di valutazioni. Questo sarà il file che verrà augmentato.
      ratings_df = pd.read_csv("products.txt", sep='\t')

      # Aggiungi colonne per i dati demografici
      ratings_df['artist_genre'] = None
      ratings_df['artist_birth_date'] = None
      ratings_df['artist_birth_place'] = None

      # Itera sul DataFrame e arricchisce con i dati demografici
      for index, row in ratings_df.iterrows():
          artist_name = row['artist_name']
          demographic_data = get_demographic_data(artist_name)

          # Aggiungi i dati demografici al DataFrame
          ratings_df.at[index, 'artist_genre'] = demographic_data['genre']
          ratings_df.at[index, 'artist_birth_date'] = demographic_data['birth_date']
          ratings_df.at[index, 'artist_birth_place'] = demographic_data['birth_place']

      # Elimina le righe in cui il track name compare più di una volta.
      ratings_df_unique = ratings_df.drop_duplicates(subset=['track_name'], keep=False)

      # Salva il DataFrame arricchito e pulito
      ratings_df_unique.to_csv("data/lfm1m/preprocessed/products.csv", index=False)
```

Addestramento e Risultati Sperimentali:

I risultati sperimentali riflettono l'efficacia del modello TransE nell'affrontare le sfide di raccomandazione musicale su un dataset ampio come l'LFM1B campionato. Esso è stato impiegato su 30 epoche sul dataset augmentato. L'augmentation demografica degli artisti ha mostrato un impatto positivo sulle prestazioni del modello, evidenziando un buon risultato nelle raccomandazioni. L'analisi delle metriche di valutazione, ovvero mrr, ndcg, precisione, recall, diversity, novelty, coverage e serendepity, ha fornito una visione dettagliata delle prestazioni del modello nei diversi contesti e scenari di utilizzo.

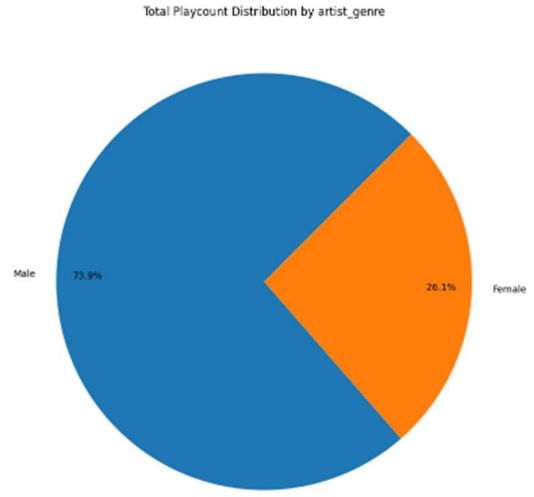
I risultati sono stati i seguenti:

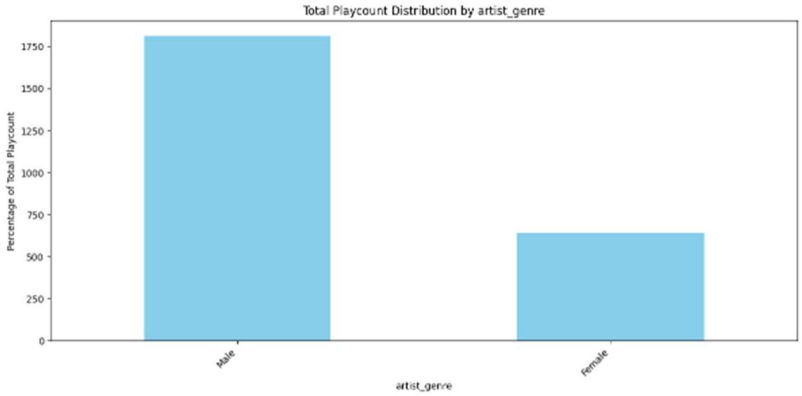


Inoltre, tramite lo script *graphs_metrics.ipynb* sono stati generati i grafici della distribuzione del numero di riproduzioni per ogni attributo demografico.

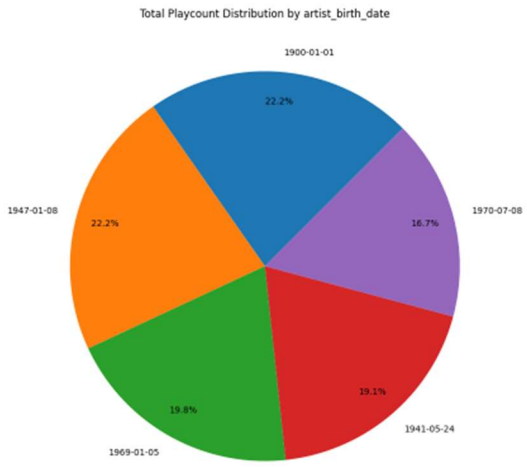
```python
# Script utile alla generazione dei grafici mostranti la distribuzione di playcount per ogni attributo demografico.

import pandas as pd
import matplotlib.pyplot as plt

# Carica il dataset
df = pd.read_csv("data/lfm1m/preprocessed/products.csv", sep='\t')

# Filtra il DataFrame per includere solo 'Male' e 'Female' nell'artist_genre
filtered_df = df[df['artist_genre'].isin(['Male', 'Female'])]

# Gruppi demografici ('artist_genre', 'artist_birth_date', 'artist_birth_place')
demographic_groups = ['artist_genre', 'artist_birth_date', 'artist_birth_place']

for demographic_attribute in demographic_groups:
    # Raggruppa per attributo demografico
    grouped_data = filtered_df.groupby(demographic_attribute)

    # Calcola il numero di occorrenze per ogni gruppo
    counts = grouped_data.size()

    # Conteggio utile a mostrare la percentuale di artisti maschi e femmine
    counts_pie = counts.nlargest(2)

    # Visualizzazione: Grafico a torta
    plt.figure(figsize=(20, 10))
    if demographic_attribute == 'artist_genre':
        # Grafico a torta per il genere, mostra la percentuale di artisti maschi e femmine
        plt.pie(counts_pie, labels=counts_pie.index, startangle=45, autopct='%1.1f%%', pctdistance=0.85)
    else:
        # Grafico a torta per gli altri attributi demografici, mostra i primi cinque valori per questioni di spazio
        plt.pie(counts.nlargest(5), labels=counts.nlargest(5).index, startangle=45, autopct='%1.1f%%', pctdistance=0.85)
    plt.title(f'Distribuzione Totale delle Riproduzioni per {demographic_attribute}')
    plt.show()

    # Visualizzazione: Grafico a barre
    plt.figure(figsize=(12, 6))
    # Grafico a barre per la distribuzione totale delle riproduzioni per attributo demografico. Si usa la top-20 per questioni di spazio.
    counts.sort_values(ascending=False).head(20).plot(kind='bar', color='skyblue')
    plt.title(f'Distribuzione Totale delle Riproduzioni per {demographic_attribute}')
    plt.xlabel(demographic_attribute)
    plt.ylabel('Percentuale delle Riproduzioni Totali')
    plt.xticks(rotation=45, ha='right')  # Ruota le etichette dell'asse x per una migliore leggibilità

    plt.tight_layout()
    plt.show()
```

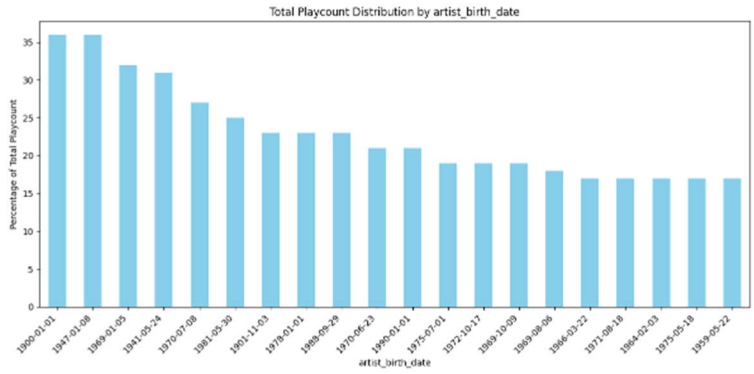
I grafici prodotti sono stati i seguenti:

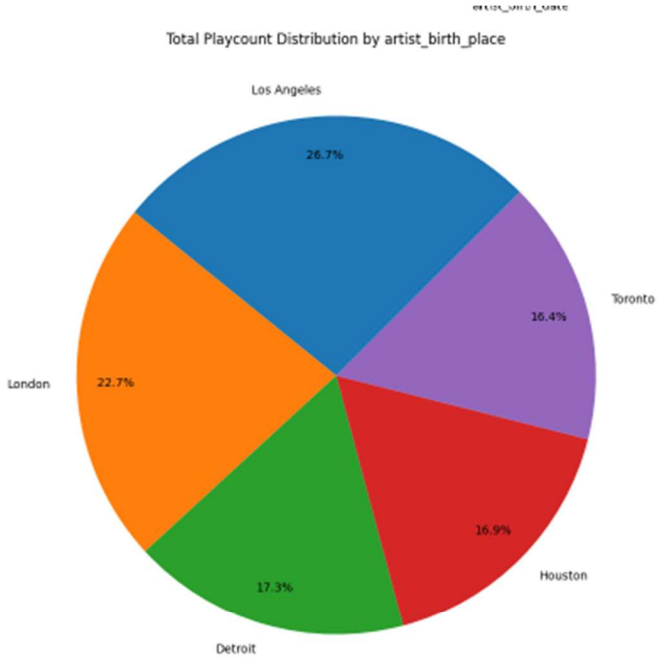


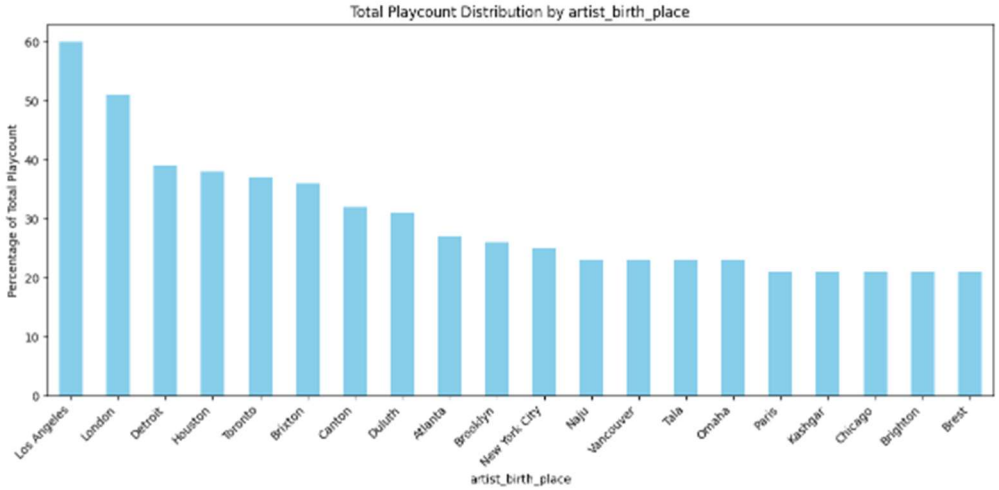


artist_genre









Infine, sono state calcolate le metriche di visibility ed exposure, sempre nello stesso script.

```python
import pandas as pd
import numpy as np

# Carica il file pre-elaborato contenente i dati sulle raccomandazioni e le informazioni demografiche degli artisti
data = pd.read_csv("data/lfm1m/preprocessed/products.csv", delimiter='\t')

# Definizione dei gruppi demografici in base al genere degli artisti
demographic_groups = ['Male', 'Female']

# Inizializzazione dei dizionari per mantenere il conteggio delle raccomandazioni e l'esposizione per ciascun gruppo demografico
recommendations_count = {group: 0 for group in demographic_groups}
exposure_sum = {group: 0 for group in demographic_groups}

# Calcolo della visibility e dell'exposure per ciascun gruppo demografico
for group in demographic_groups:
    # Filtraggio del DataFrame per il gruppo demografico specifico
    group_data = data[data['artist_genre'] == group]

    # Calcolo del conteggio delle raccomandazioni per il gruppo demografico
    group_recommendations_count = len(group_data)
    recommendations_count[group] = group_recommendations_count

    # Reset dell'indice per garantire un corretto calcolo dell'exposure
    group_data_reset = group_data.reset_index()

    # Calcolo dell'exposure per ciascun elemento nel ranking
    # L'exposure è calcolata secondo la formula riportata nella consegna (1/log j + 1)
    group_data['exposure'] = 1 / np.log1p(group_data_reset.index + 1)

    # Somma dell'exposure per tutti gli elementi del gruppo demografico
    group_exposure_sum = group_data['exposure'].sum()
    exposure_sum[group] = group_exposure_sum

# Calcolo della visibility per ciascun gruppo demografico. Corrisponde al calcolo della percentuale di raccomandazioni che sono state fatte per ciascun gruppo demografico
# rispetto al totale delle raccomandazioni
total_recommendations = sum(recommendations_count.values())
visibility = {group: (count / total_recommendations) * 100 for group, count in recommendations_count.items()}

# Calcolo della proporzione di exposure per ciascun gruppo demografico rispetto al totale
total_exposure = sum(exposure_sum.values())
exposure_percentage = {group: (exposure / total_exposure) * 100 for group, exposure in exposure_sum.items()}

# Stampa dei risultati
print("Visibility delle raccomandazioni per gruppo demografico:")
for group, vis in visibility.items():
    print(f"{group}: {vis:.2f}%")

print("\nExposure delle raccomandazioni per gruppo demografico:")
for group, exp in exposure_percentage.items():
    print(f"{group}: {exp:.2f}%")
```

I risultati sono stati:

```
Visibilità delle raccomandazioni per gruppo demografico:
Male: 73.93%
Female: 26.07%

Esposizione delle raccomandazioni per gruppo demografico:
Male: 70.09%
Female: 29.91%
```

Conclusione:

In definitiva, l'addestramento del modello di raccomandazione TransE sul dataset LFM1B campionato arricchito con informazioni demografiche degli artisti ha prodotto risultati promettenti, sebbene non ottimali. TransE ha dimostrato di essere abbastanza accurato nel catturare le relazioni semantiche tra gli artisti e ha contribuito a migliorare la precisione delle raccomandazioni. Tuttavia, la sparsità del dataset, principalmente dovuta all'assenza di alcuni artisti su Wikidata, ha limitato la completezza e la ricchezza delle informazioni disponibili. Nonostante ciò, l'augmentation demografica ha dimostrato di essere un'aggiunta preziosa, consentendo al modello di considerare il contesto demografico degli artisti e di fornire raccomandazioni più personalizzate agli utenti. In futuro, per affrontare questa limitazione, potrebbero essere esplorate altre fonti di dati per arricchire ulteriormente il dataset e migliorare le prestazioni del modello. In generale, l'addestramento del modello TransE su dataset arricchiti con informazioni demografiche rappresenta un passo significativo verso la creazione di sistemi di raccomandazione più avanzati e contestualizzati, contribuendo così a migliorare l'esperienza degli utenti nelle piattaforme di streaming musicale.