Final Capstone Presentation | January 2022

Movie Recommendation using TensorFlow Recommenders

Mohamed Ziane Data Science Track – Machine Learning Specialty

Mentor: Dipanjan Sarkar

January 2022

Outline

- Problem Statement
- Dataset
- Data Wrangling
- Exploration Data Analysis
- Machine Learning Application using TensorFlow Recommenders (TFRS)
- Future Work

Problem Statement

- This project aims to:
    - Design a recommendation engine
    - Differentiate between implicit and explicit feedback
    - Build a movie recommendation system with TFRS

- Success Criteria: The ability to build a movie recommendation engine yielding the highest possible retrieval accuracy (predicting movies) coupled with the lowest Loss/Ranking RMSE (ranking movies)

- Constraints: TFRS is a relatively new package

- Dataset Origin: The MovieLens website was the main source for the datasets fetched:
    - movie_lens/100k-ratings
    - movie_lens/100k-movies
    - Metadata_movies/credit

Dataset

- 100k Movielens from TensorFlow is our main dataset for this project. Also, we used both datasets from the Movielens website: movies metadata & credits

- Description:
  - 100,000 anonymous ratings of approximately 1,682 movies made by 943 MovieLens users who joined MovieLens.
  - Ratings are in whole-star increments.
  - Demographic data of users in addition to data on movies and ratings.

- Main Features:
  - "user_gender": gender of the user who made the rating; a true value corresponds to male.
  - "bucketized_user_age": bucketized age values of the user who made the rating.
  - "movie_genres": The Genres of the movies are classified into 21 different classes.
  - "user_occupation_label": the occupation of the user who made the rating represented by an integer-encoded label; labels are preprocessed to be consistent across different versions.
  - "user_occupation_text": the occupation of the user who made the rating in the original string; different versions can have different set of raw text labels.
  - "user_zip_code": the zip code of the user who made the rating.
  - "release_date": This is the movie release date, in unix epoch (UTC - units of seconds) (int64).
  - "director": This is the director of the movie.
  - "start": This is the main star of the movie.

Data Wrangling

- The Data wrangling step focuses on collecting or converting the data, organizing it, and making sure it's well defined.

- We'll focus on:

    - Cleaning NANs (If any), duplicate values (If any), wrong values and removing insignificant columns.
    - Removing any special characters.
    - Renaming some Column labels.
    - Correcting some datatypes.

- Objectives:

    - Changing the user_gender from booleans "Female" or "Male" to the following association: True:"Male", False:"Female"
    - Removing the symbols: (b), (') and (").
    - Dropping the following columns: "user_occupation_label" and "movie_genres".
    - Changing "timestamp" which is in the unix epoch (units of seconds) to a datetime64 type.
    - Fixing any wrong values in "user_zip_code" (removing any zipcode >5 characters & zip codes made out of letters)

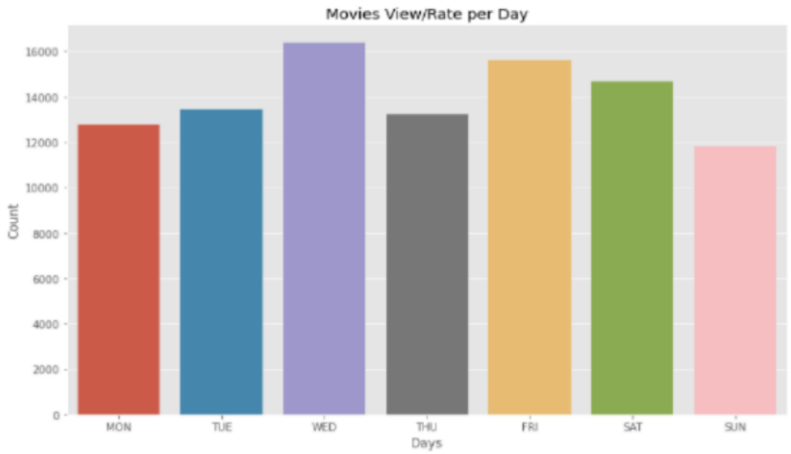
Exploration Data Analysis (EDA)

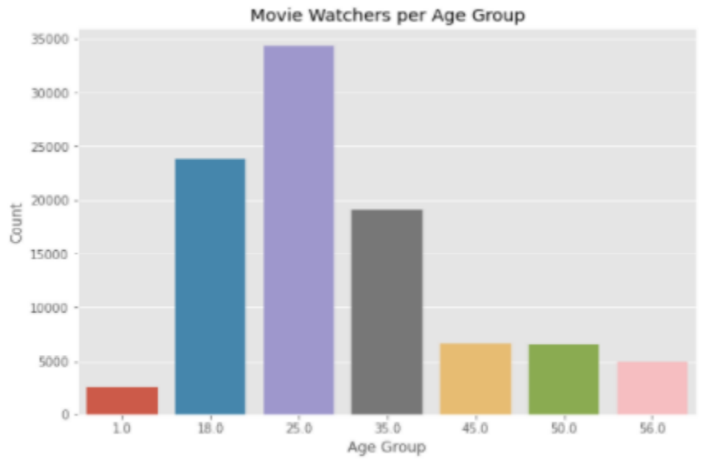
- The Exploratory Data Analysis (EDA) Step will focus on:
    - To get familiar with the features in our dataset.
    - Generally, understand the core characteristics of our cleaned dataset.
    - Explore the data relationships of all the features and understand how the features compare to the response variable.
    - We will think about interesting figures and all the plots that can be created to help deepen our understanding of the data.
    - We will be creating one feature that give us the year when the movie was released and will call it "movie_year_release".

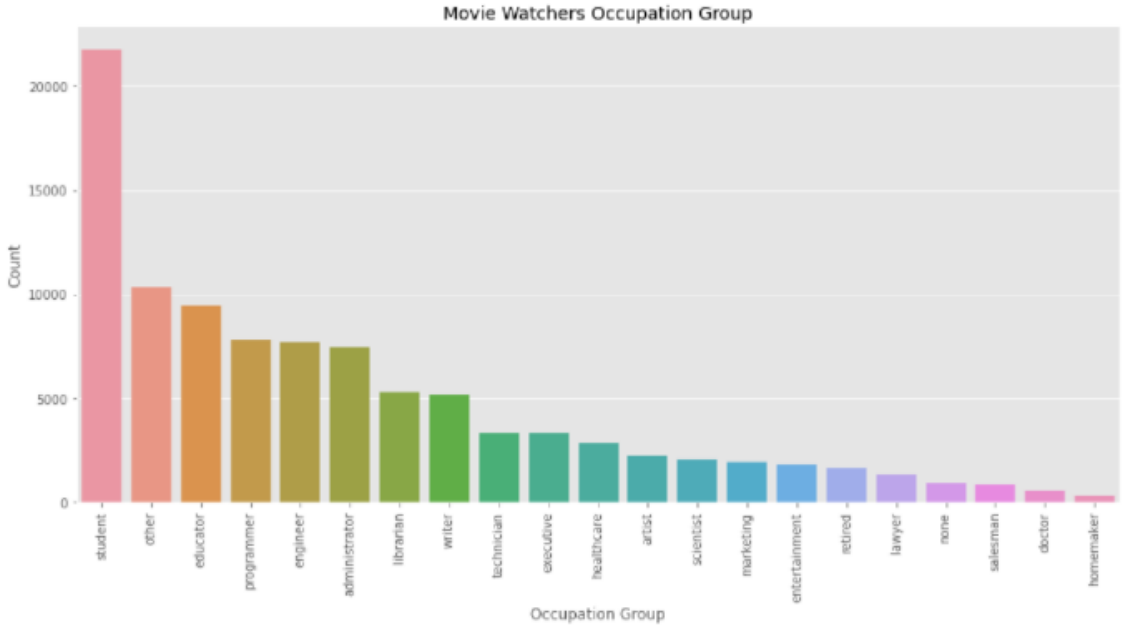
[418]:

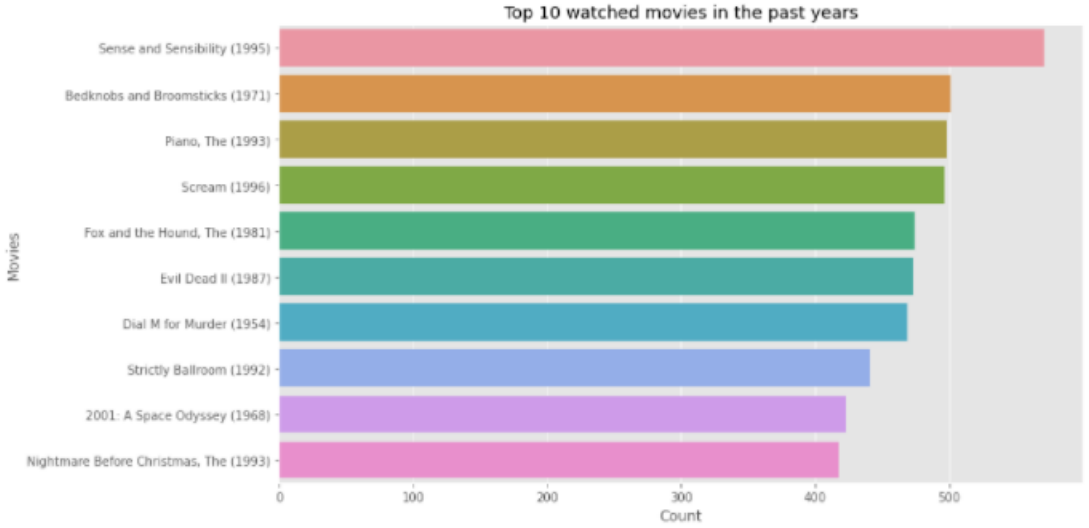
	bucketized_user_age	movie_id	raw_user_age	user_id	user_occupation_label	user_rating	user_zip_code
count	97914.0	97914.0	97914.0	97914.0	97914.0	97914.0	97914.0
mean	29.2	425.0	33.0	461.5	11.4	3.5	52179.2
std	12.1	330.4	11.6	265.6	6.5	1.1	30976.4
min	1.0	1.0	7.0	1.0	0.0	1.0	0.0
25%	18.0	175.0	24.0	256.0	6.0	3.0	22902.0
50%	25.0	321.0	30.0	445.0	12.0	4.0	55106.0
75%	35.0	630.8	40.0	682.0	17.0	4.0	80913.0
max	56.0	1681.0	73.0	943.0	21.0	5.0	99835.0

Exploration Data Analysis (EDA)

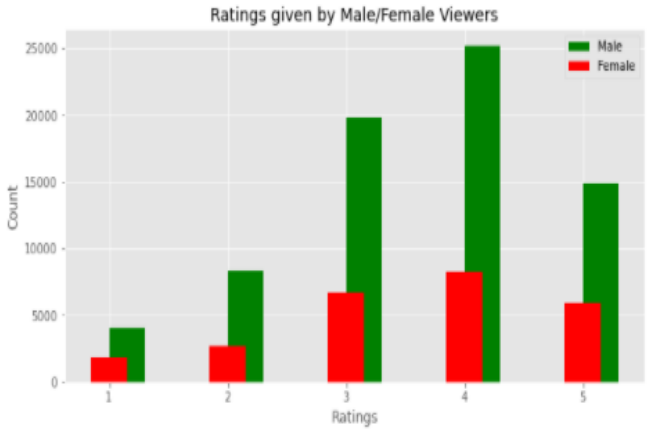


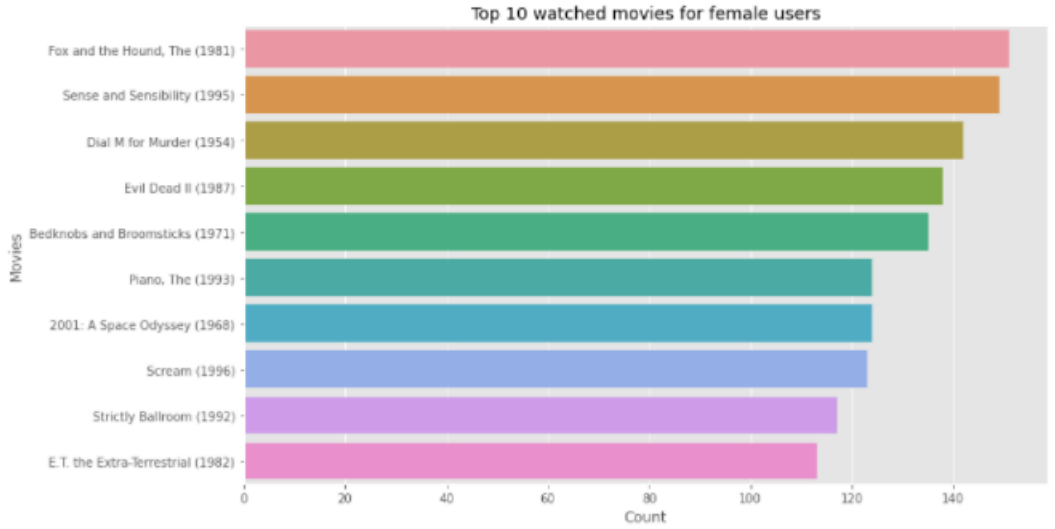


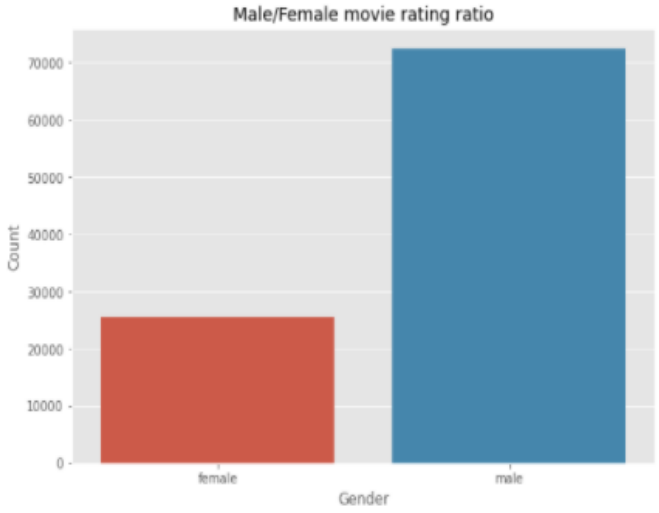


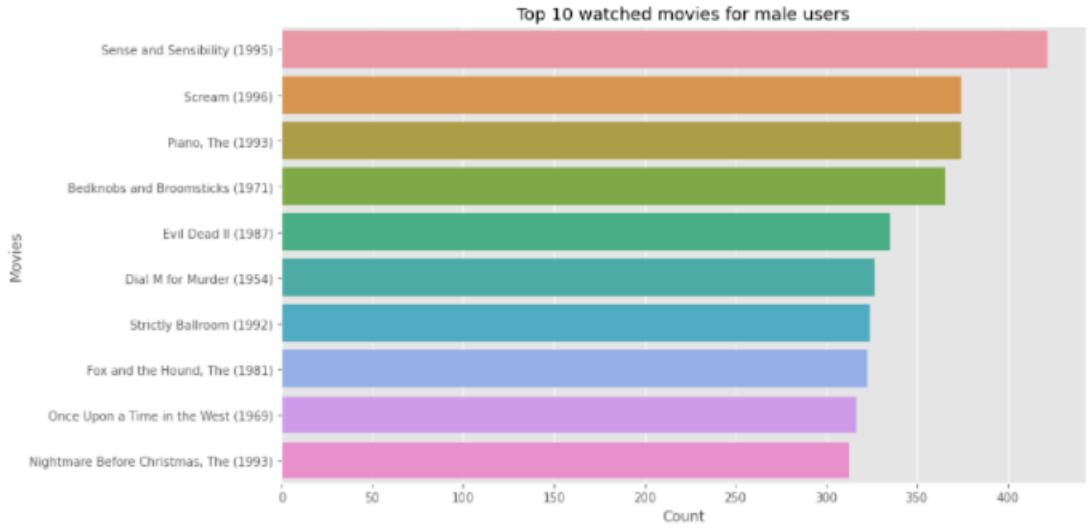


Exploration Data Analysis (EDA)

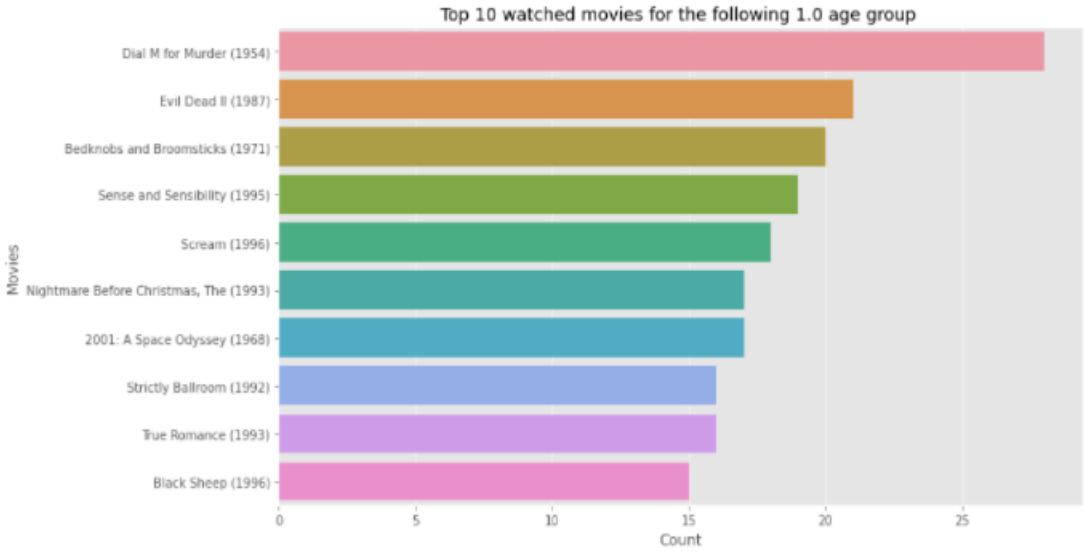


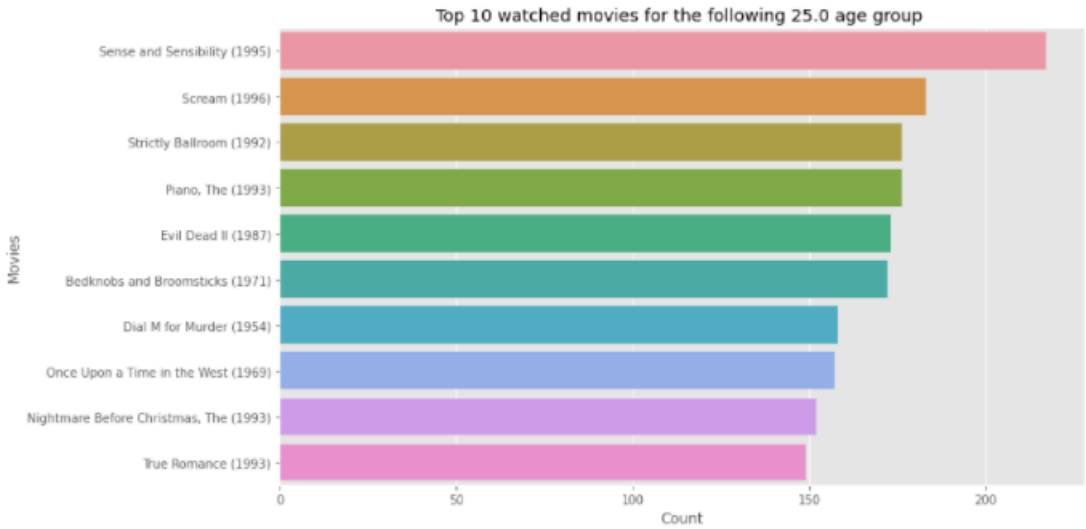


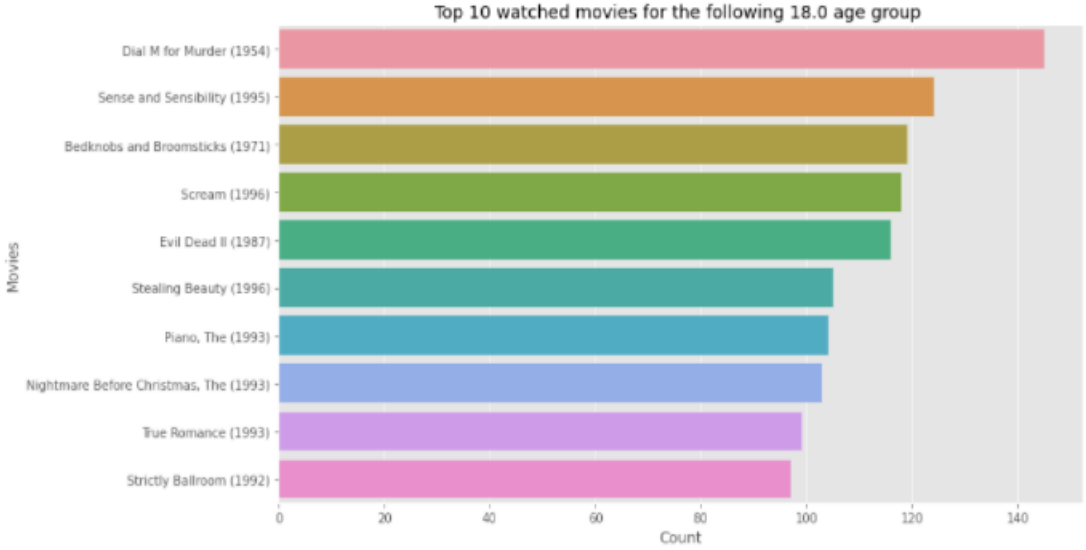


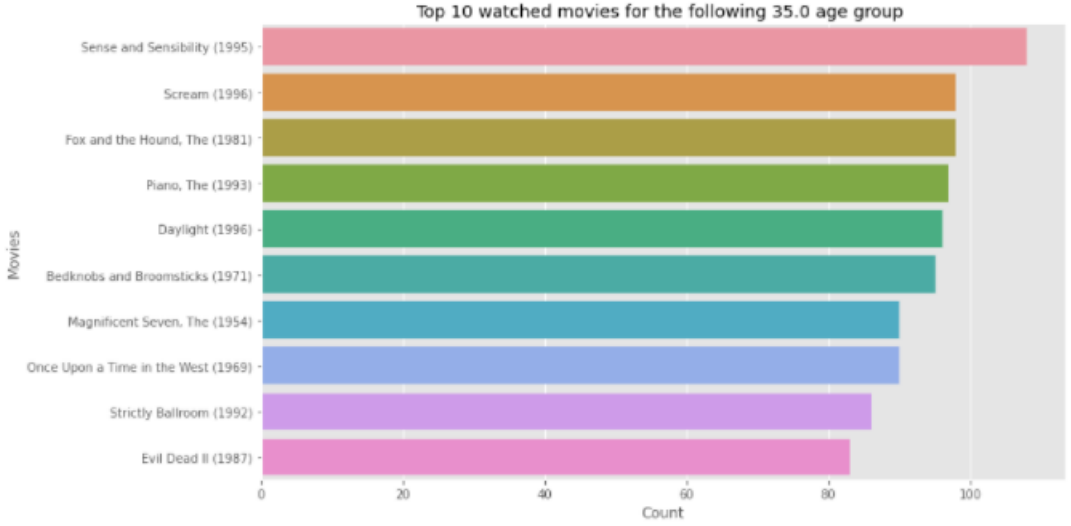


Exploration Data Analysis (EDA)

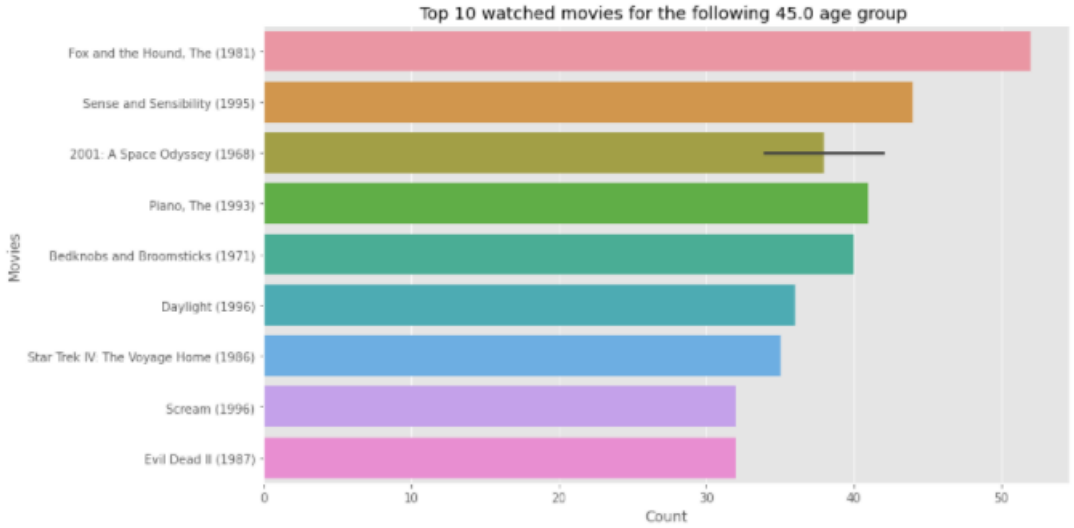


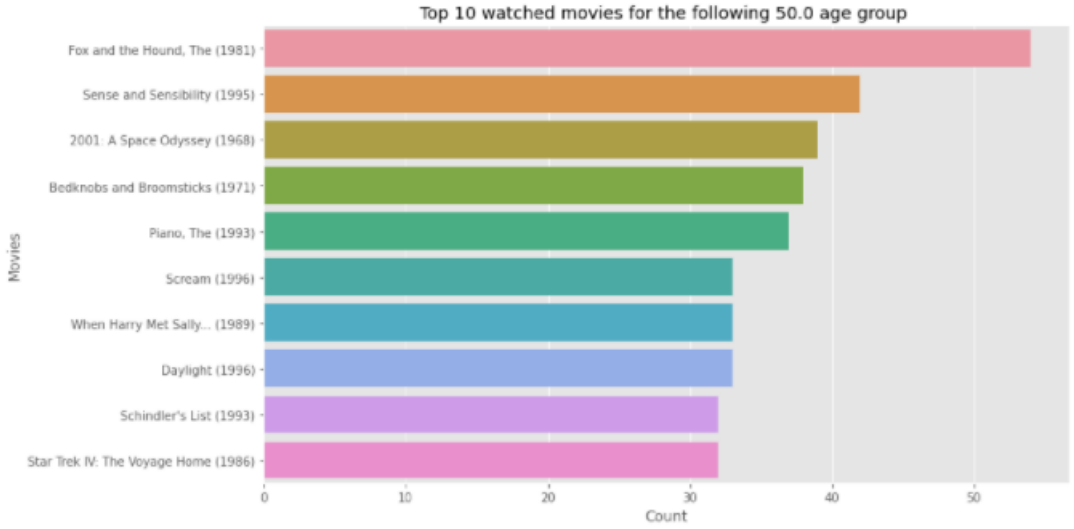


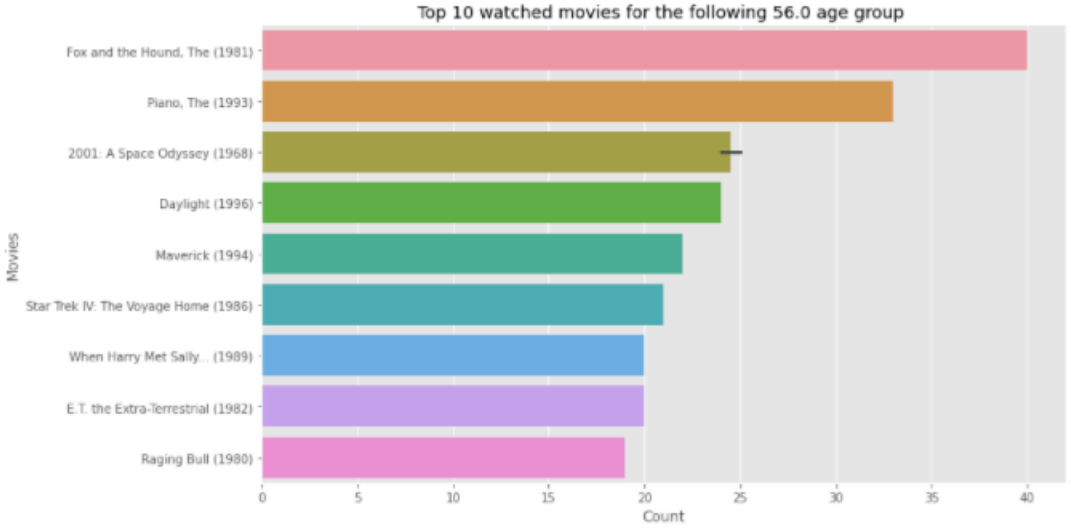




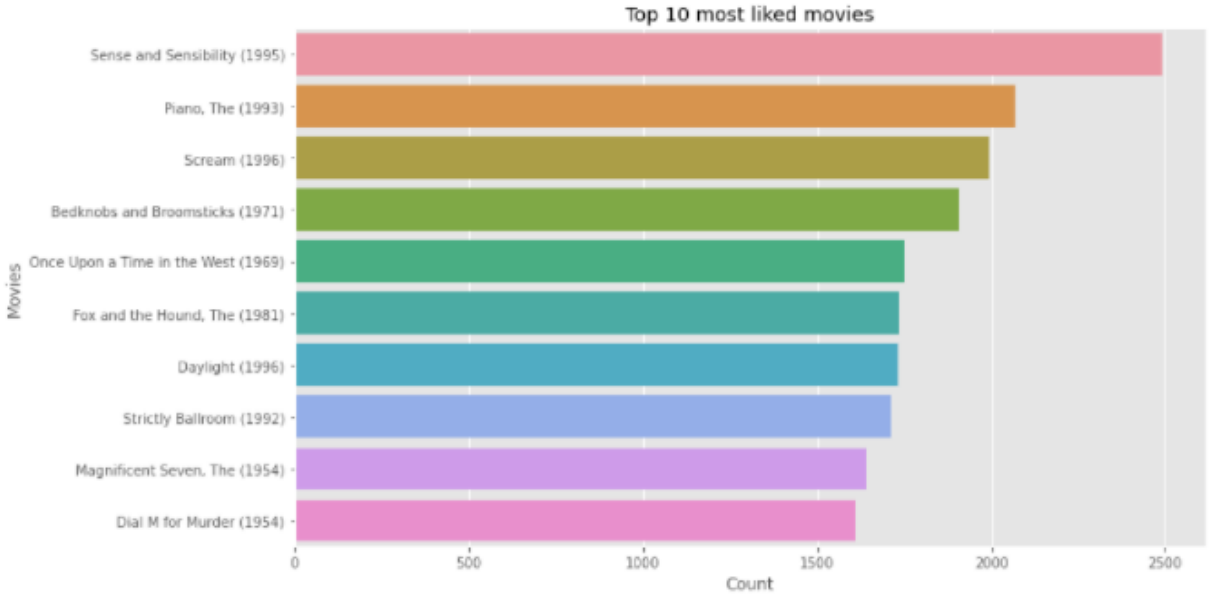
Exploration Data Analysis (EDA)



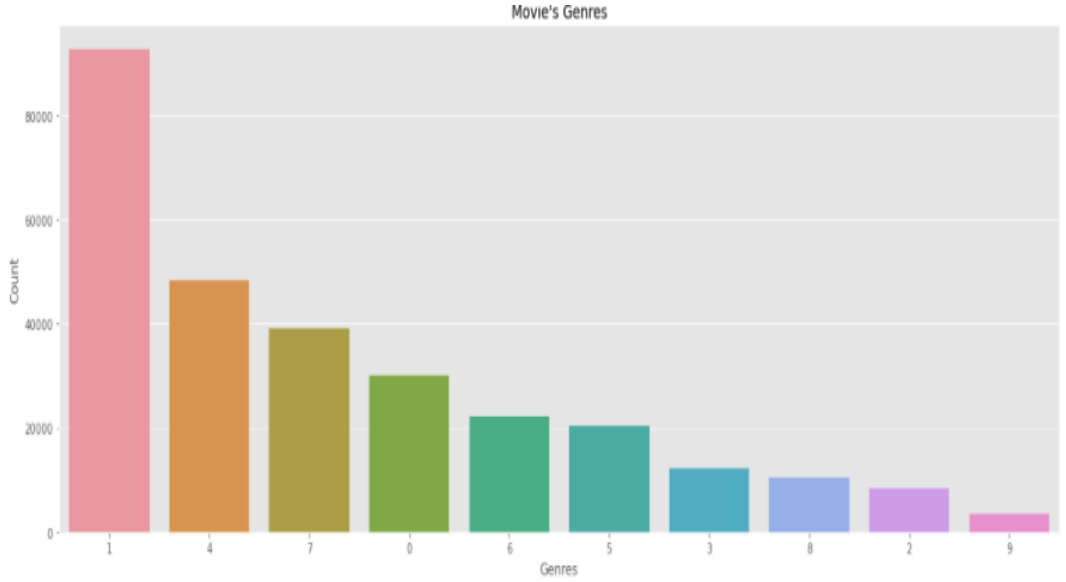


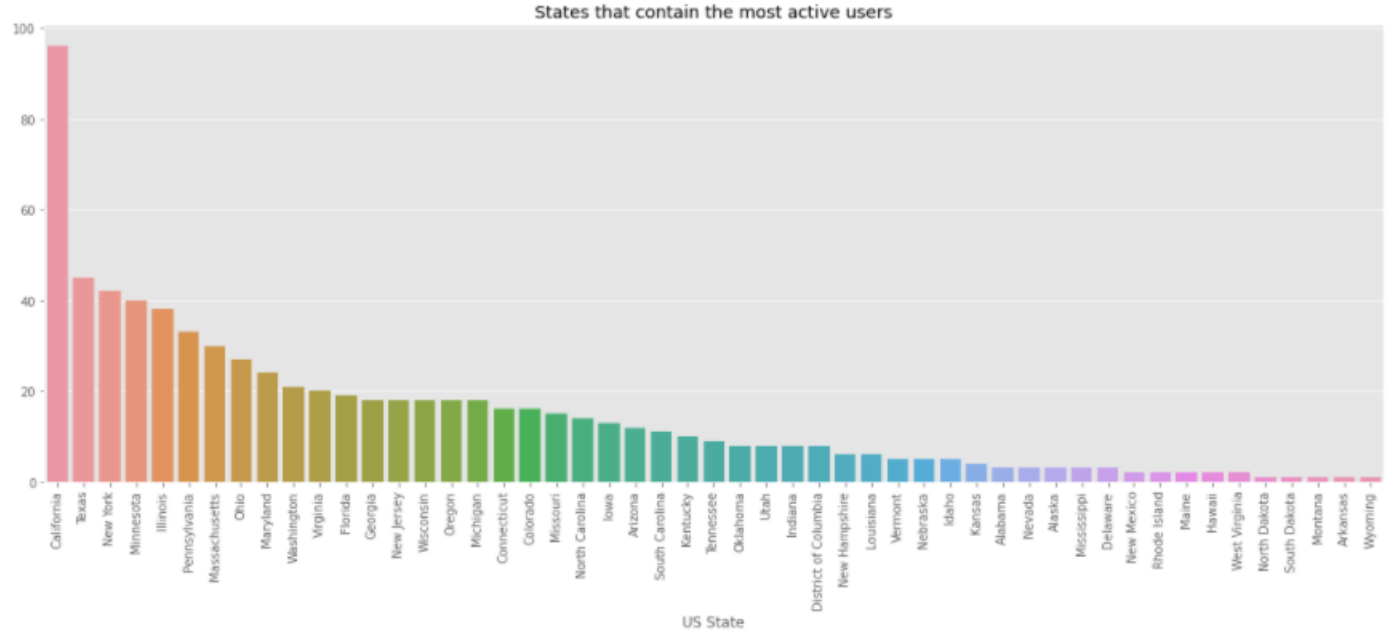


Exploration Data Analysis (EDA)



What are the worst movies per rating?





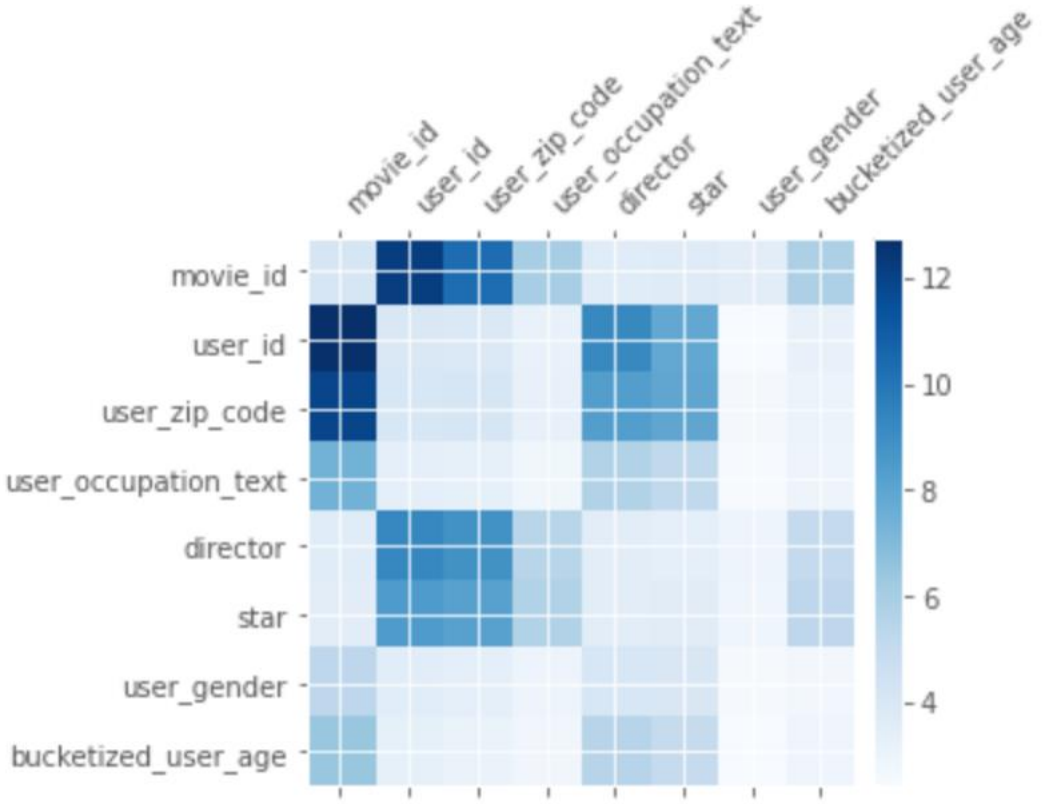
Machine Learning Application using TensorFlow Recommenders (TFRS)
Deep and cross network (DCN)

Deep and cross network (DCN) came out of Google Research, and is designed to learn explicit and bounded-degree cross features effectively:

- Large and sparse feature space is extremely hard to train.

- Oftentimes, we needed to do a lot of manual feature engineering, including designing cross features, which is very challenging and less effective.

- Whilst possible to use additional neural networks under such circumstances, it's not the most efficient approach. DCN is specifically designed to tackle all of the above challenges.



The feature cross of user ID and movie ID are of great importance.

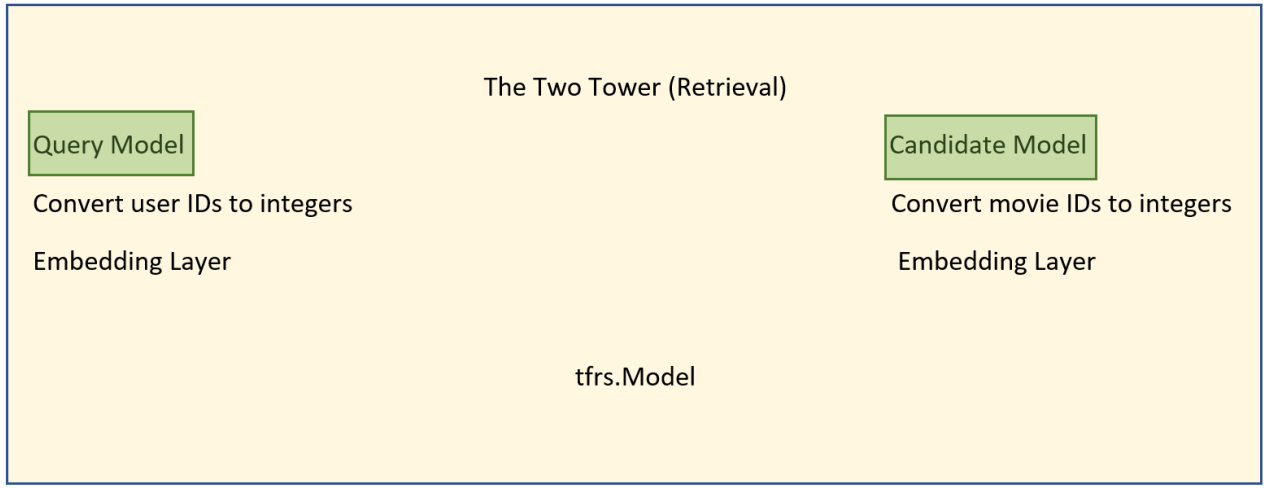
Machine Learning Application using TensorFlow Recommenders (TFRS)
The Two-Tower & Ranking Models: Baseline

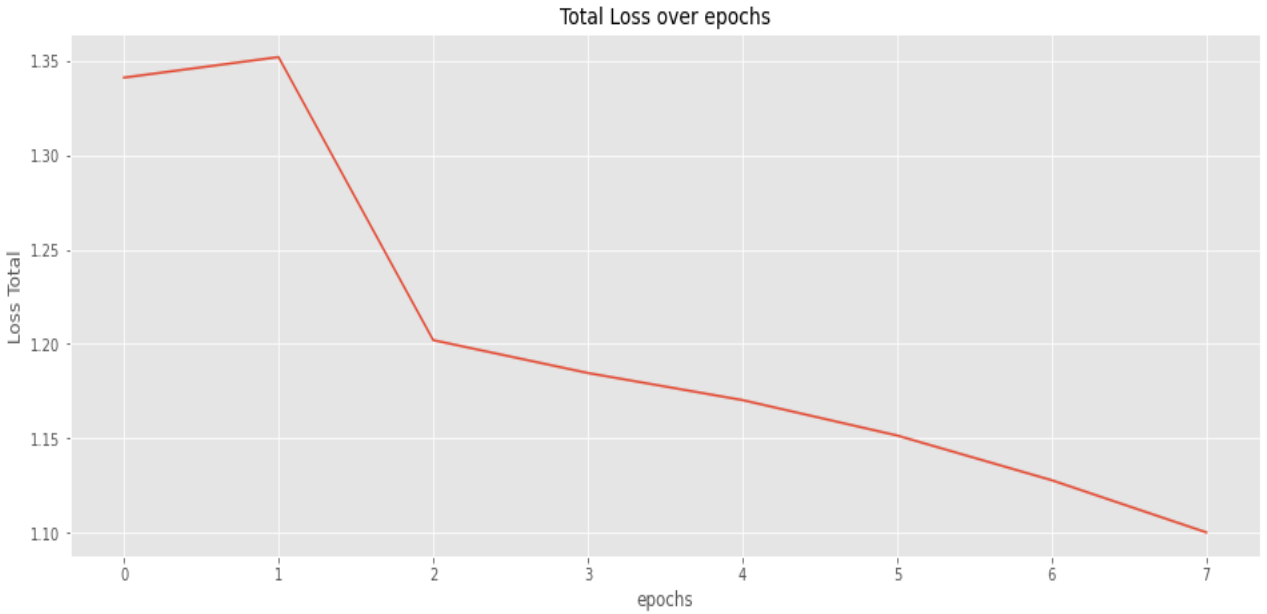
The retrieval stage (Selects recommendation candidates): is responsible for selecting an initial set of hundreds of candidates from all possible candidates.

- The main objective of this model is to efficiently weed out all candidates that the user is not interested in. Because the retrieval model may be dealing with millions of candidates, it has to be computationally efficient.

The ranking stage (Selects the best candidates and rank them): takes the outputs of the retrieval model and finetunes them to select the best possible handful of recommendations.

- Its task is to narrow down the set of items the user may be interested in to a shortlist of likely candidates

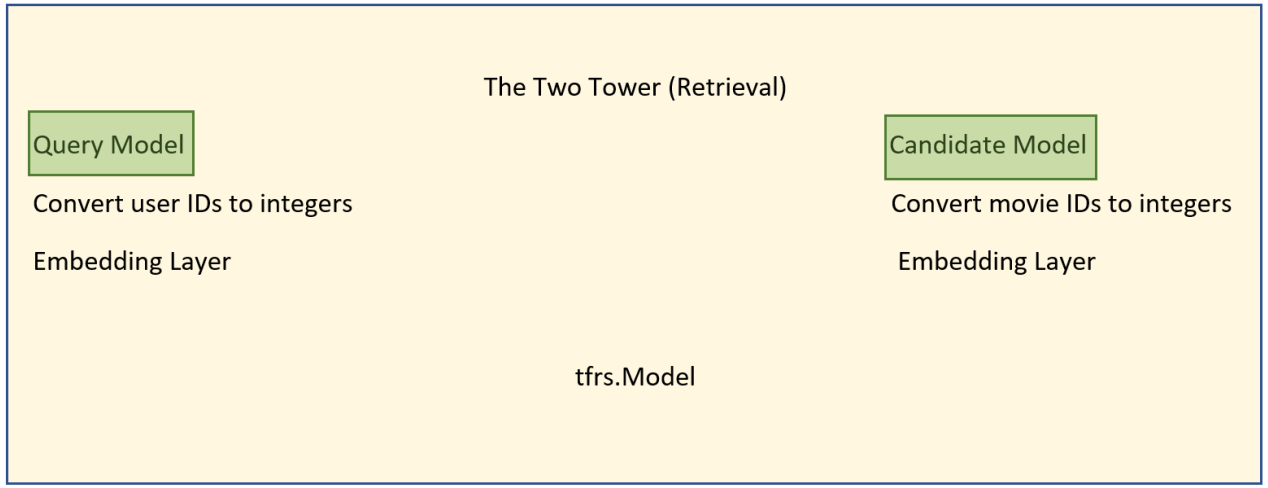




Machine Learning Application using TensorFlow Recommenders (TFRS)
The Two-Tower & Ranking Models: Baseline

The joint model seems to provide an overall better prediction than the other independent models What can we improve upon the joint model?

- Adding more features
- Optimize Embedding
- embedding_dimension
- epochs= 8 to 16
- Learning rate



Models	Retrieval top-10 accuracy	Retrieval top-100 accuracy	Ranking RMSE
Model: Rating-specialized model	0.1153	0.236	1.071
Model: Retrieval-specialized model	0.0013	0.058	3.691
Model: Joint model: Baseline	0.1474	0.285	1.087

Machine Learning Application using TensorFlow Recommenders (TFRS)
The Two-Tower & Ranking Models: Baseline vs. Tuned Model

Those following classes were re-configured to accommodate the new embedded design due to new features, having deeper neural networks and adding regularization to help overfitting:

- class UserModel
- class QueryModel
- class MovieModel
- class MovieModel
- class CandidateModel
- class MovielensModel

Models	Retrieval top-10 accuracy	Retrieval top-100 accuracy	Ranking RMSE
Model: Rating-specialized model	0.1153	0.236	1.071
Model: Retrieval-specialized model	0.0013	0.058	3.691
Model: Joint model: Baseline	0.1474	0.285	1.087
Model: Tuned Joint model (Best Model for teh Recommendation)	0.9544	0.96	1.11

Future Work

- Further optimizing existing Tensorflow Recommenders Classes to accommodate extra features.
- Additional fine-tuning needed to enhance retrieval prediction and reduce RMSE Ranking from the Joint Baseline Model (Multi-Task)
- Building an application.