

**MOVIE RECOMMENDATION USING TENSORFLOW**  
**RECOMMENDERS AND VISUALIZING PERFORMANCE**  
**USING TENSORBOARD**



Praveen Karuppasamy S

2022510301

B. Tech Artificial Intelligence and Data Science

Madras Institute of Technology

## Introduction:

This project explores the implementation of movie recommendation models using TensorFlow Recommenders (TFRS) and visualizes their performance using TensorBoard. Recommender systems are crucial in today's digital landscape for personalizing user experiences, particularly in streaming services like Netflix or Spotify. TensorFlow Recommenders provides a robust framework for building such systems, leveraging deep learning techniques to predict user preferences based on historical interactions.

The primary objectives of this project include:

1. **Model Implementation:** Implementing two distinct recommendation models using TensorFlow Recommenders. These models are designed to learn from user-item interactions in the MovieLens dataset, aiming to predict user preferences accurately.
2. **Training and Evaluation:** Training the models on the MovieLens dataset, evaluating their performance metrics such as accuracy and top-k categorical accuracy. The evaluation involves comparing how well each model predicts user preferences against known ratings.
3. **Visualization with TensorBoard:** Utilizing TensorBoard for visualizing the training process and comparing the performance of the recommendation models. TensorBoard provides insights into metrics like loss and accuracy over epochs, aiding in model optimization and selection.

By the end of this project, users will gain a comprehensive understanding of leveraging TensorFlow Recommenders for building robust recommendation systems and utilizing TensorBoard for effective performance visualization and comparison.

# Installation

To run the notebook and utilize the recommendation models and TensorBoard visualization, follow these steps:

## 1. Clone the repository:

```
git clone https://github.com/your-username/Movie-Recommendation-using-
Tensorflow-Recommendors-and-Visualising-performance-using-
tensorboard.git
cd Movie-Recommendation-using-Tensorflow-Recommendors-and-Visualising-
performance-using-tensorboard
```

## 2. Install Dependencies:

Ensure you have Python installed. Use pip to install the required packages:

```
pip install tensorflow tensorflow-recommenders tensorboard
```

This command installs TensorFlow, TensorFlow Recommenders, and TensorBoard, necessary for building models and visualizing results.

## 3. Open the Jupyter Notebook:

- **Local Environment:** Run Jupyter Notebook:

```
jupyter notebook
```

- **Google Colab:** Upload the notebook to Google Colab for cloud-based execution.

## 4. Run the Notebook:

- Follow the instructions provided in the notebook to load data, build models, train them, and visualize results using TensorBoard.

# Program Explanation and Working


## Components

1. **Loading Data:**
  - The project begins by loading the MovieLens dataset, which contains user ratings for movies.
  - Data preprocessing steps include filtering, splitting into training and validation sets, and preparing it for modeling.
2. **Building Models:**
  - Two types of recommendation models are defined:
    - **User-based Model:** Focuses on learning user preferences.
    - **Movie-based Model:** Focuses on learning movie characteristics.
  - Each model includes embedding layers and dense layers tailored for capturing user and movie interactions.
3. **Training Models:**
  - Models are compiled with appropriate optimizers and loss functions, such as categorical cross-entropy.
  - Training involves iterating over batches of data, optimizing the model parameters to minimize loss, and logging metrics using TensorBoard.
4. **Evaluating Models:**
  - Post-training, models are evaluated using retrieval tasks to predict user preferences for movies.
  - Metrics like top-k categorical accuracy are computed to measure how well the models predict user preferences against ground truth.
5. **Visualizing Results:**
  - TensorBoard is utilized to visualize various training metrics such as loss and accuracy over epochs.
  - Comparison between different models is facilitated through TensorBoard's graphical interface, aiding in identifying the most effective model.

# LOADING THE DATASET

Downloading and preparing dataset 4.70 MiB (download: 4.70 MiB, generated: 32.41 MiB, total: 37.10 MiB) to /root/tensorflow\_datasets/movielens/100k-ratings/0.1.1...

DI Completed...: 100%  1/1 [00:03<00:00, 3.35s/ url]

DI Size...: 100%  4/4 [00:03<00:00, 2.69s/ MiB]

Extraction completed...: 100%  23/23 [00:03<00:00, 3.46s/ file]

Dataset movielens downloaded and prepared to /root/tensorflow\_datasets/movielens/100k-ratings/0.1.1. Subsequent calls will reuse this data.

Downloading and preparing dataset 4.70 MiB (download: 4.70 MiB, generated: 150.35 KiB, total: 4.84 MiB) to /root/tensorflow\_datasets/movielens/100k-movies/0.1.1...

DI Completed...: 100%  1/1 [00:00<00:00, 12.24 url/s]

DI Size...: 100%  4924029/4924029 [00:00<00:00, 77513002.85 MiB/s]

Extraction completed...: 0/0 [00:00<?, ? file/s]

Dataset movielens downloaded and prepared to /root/tensorflow\_datasets/movielens/100k-movies/0.1.1. Subsequent calls will reuse this data.

---

## Viewing dataset:

Ratings Dataset:

User ID: b'138'

Movie Title: b"One Flew Over the Cuckoo's Nest (1975)"

---

User ID: b'92'

Movie Title: b'Strictly Ballroom (1992)'

---

User ID: b'301'

Movie Title: b'Very Brady Sequel, A (1996)'

---

User ID: b'60'

Movie Title: b'Pulp Fiction (1994)'

---

User ID: b'197'

Movie Title: b'Scream 2 (1997)'

---

---

# MODEL TRAINING :

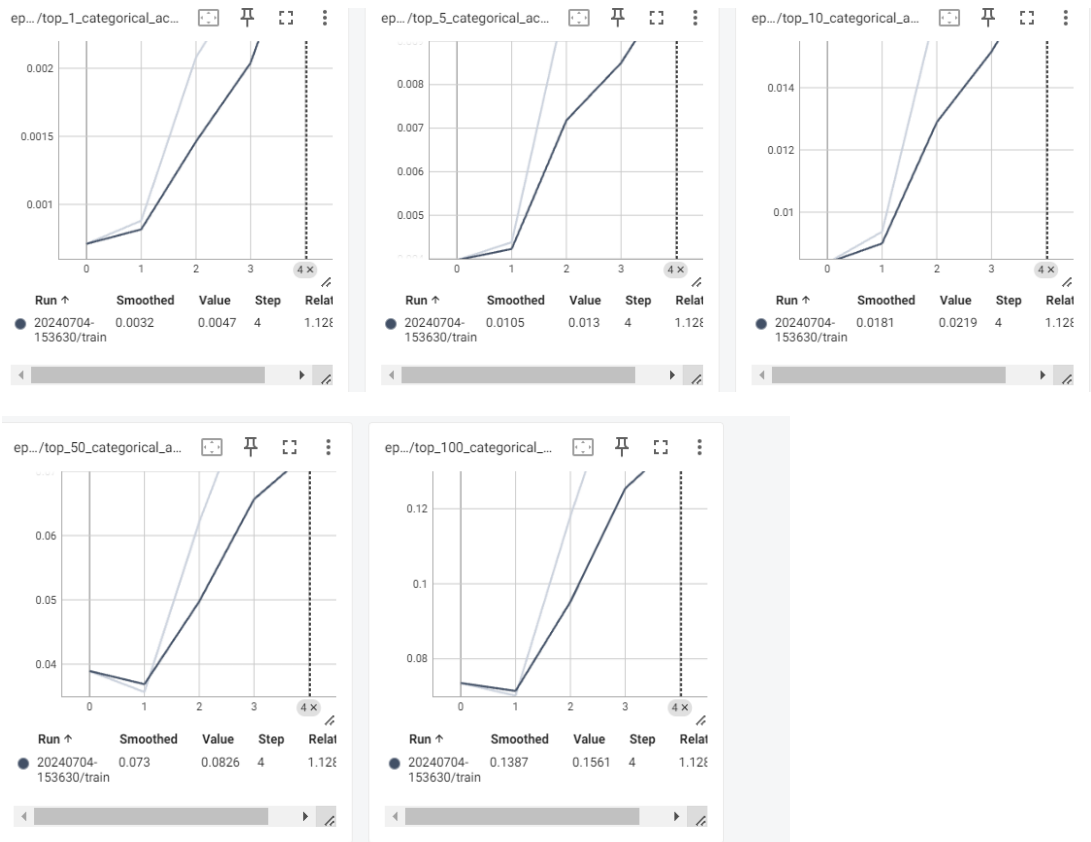
Epoch 1/5  
25/25 [=====] - 19s 706ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 3.5000e-04 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0022 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0063  
Epoch 2/5  
25/25 [=====] - 16s 640ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 3.1000e-04 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0022 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0063  
Epoch 3/5  
25/25 [=====] - 16s 640ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 2.8000e-04 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0022 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0063  
Epoch 4/5  
25/25 [=====] - 17s 691ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 3.6000e-04 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0023 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0063  
Epoch 5/5  
25/25 [=====] - 16s 642ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 4.0000e-04 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0023 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0063  
Epoch 1/5  
25/25 [=====] - 19s 698ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 0.0160 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0239 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0807  
Epoch 2/5  
25/25 [=====] - 16s 619ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 0.0012 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0063 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0239  
Epoch 3/5  
25/25 [=====] - 15s 615ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 0.0016 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0087 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0239  
Epoch 4/5  
25/25 [=====] - 16s 622ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 0.0018 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0094 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0239  
Epoch 5/5  
25/25 [=====] - 16s 618ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 0.0016 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0097 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0239  
(keras.src.callbacks.history at 0x7930a521fdf0)

# EVALUATING MODELS:

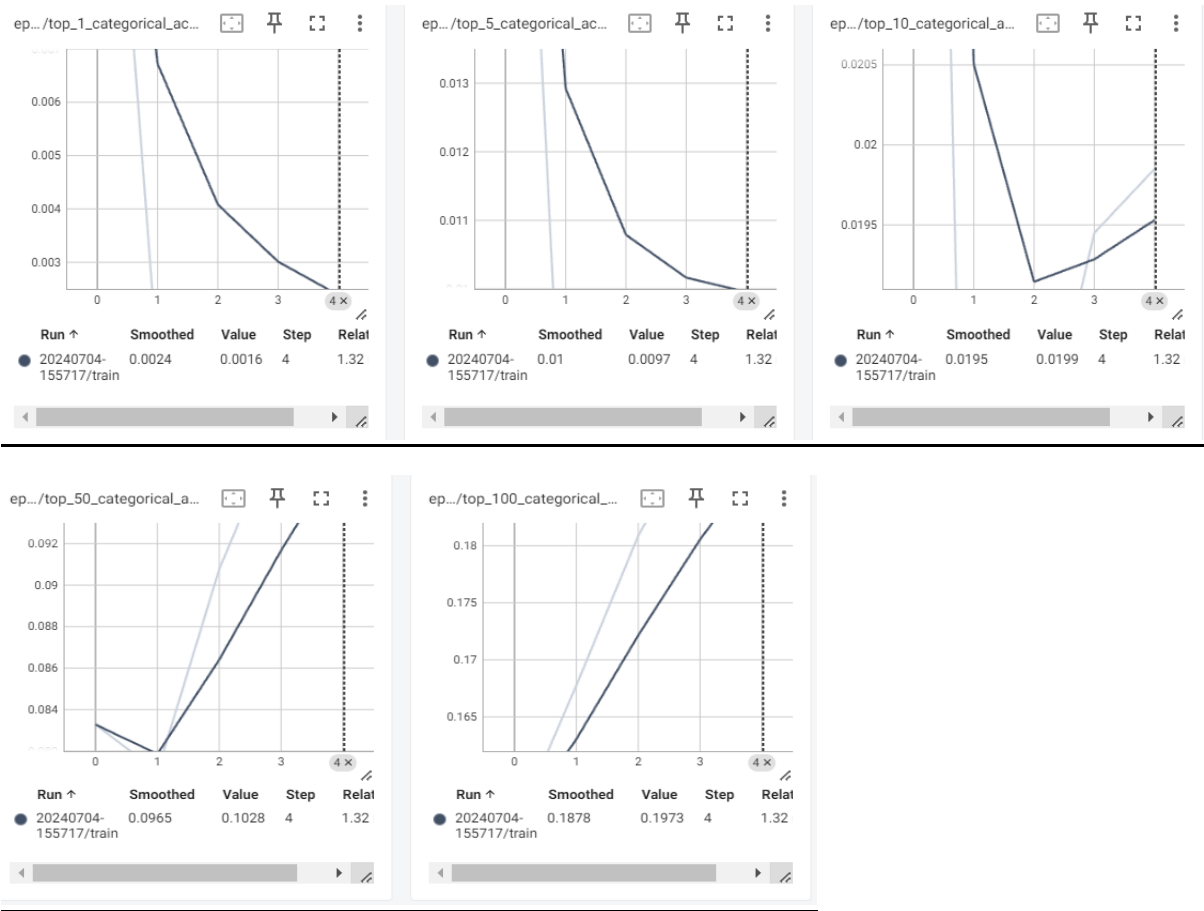
25/25 [=====] - 15s 593ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 4.0000e-04 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0023 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0063  
25/25 [=====] - 15s 576ms/step - factorized\_top\_k/top\_1\_categorical\_accuracy: 8.5000e-04 - factorized\_top\_k/top\_5\_categorical\_accuracy: 0.0063 - factorized\_top\_k/top\_10\_categorical\_accuracy: 0.0239  
First Model Evaluation: [0.00039999999999999999, 0.0023899999999999999, 0.005140000000000000, 0.02895000000000000, 0.05857999999999999, 0.1261150390625, 0, 12611.50390625]  
Second Model Evaluation: [0.0008500000000000000, 0.006279999999999999, 0.013319999999999999, 0.08040000000000000, 0.1706700000000000, 11964.9150390625, 0, 11964.9150390625]

# EPOCH FACTORIZED TOP K(5 CARDS) :

## [MODEL 1]:

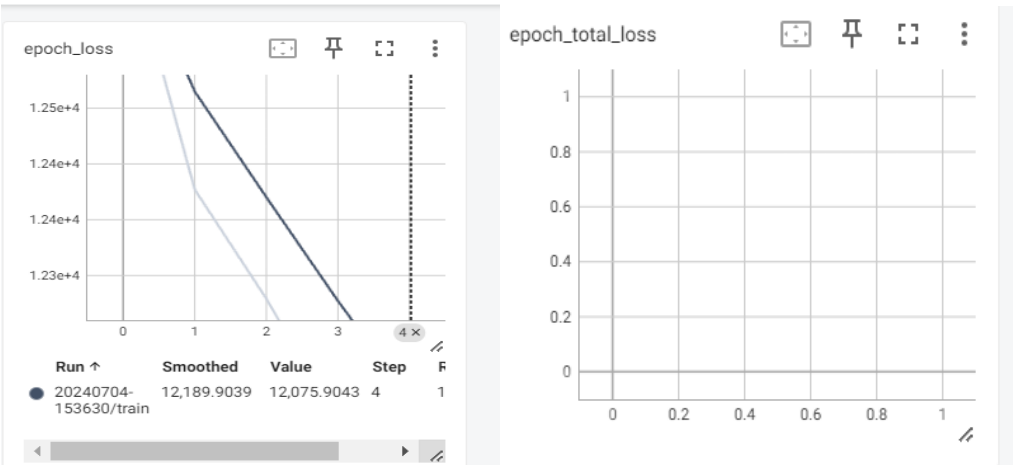


[MODEL 2]:

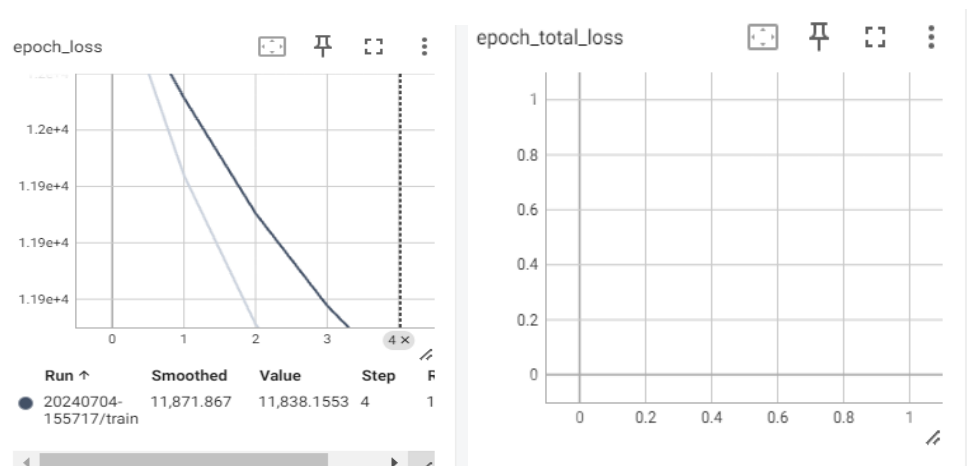


EPOCH LOSS:

[MODEL 1]



## [MODEL 2]



## Differences in Configurations and Outputs:

### Activation Functions:

Model 1: Uses relu activation functions in dense layers.

Model 2: Uses swish activation functions in dense layers, and an additional dense layer with softmax activation.

### Output Layer:

Model 1: Does not include a final dense layer with a softmax activation.

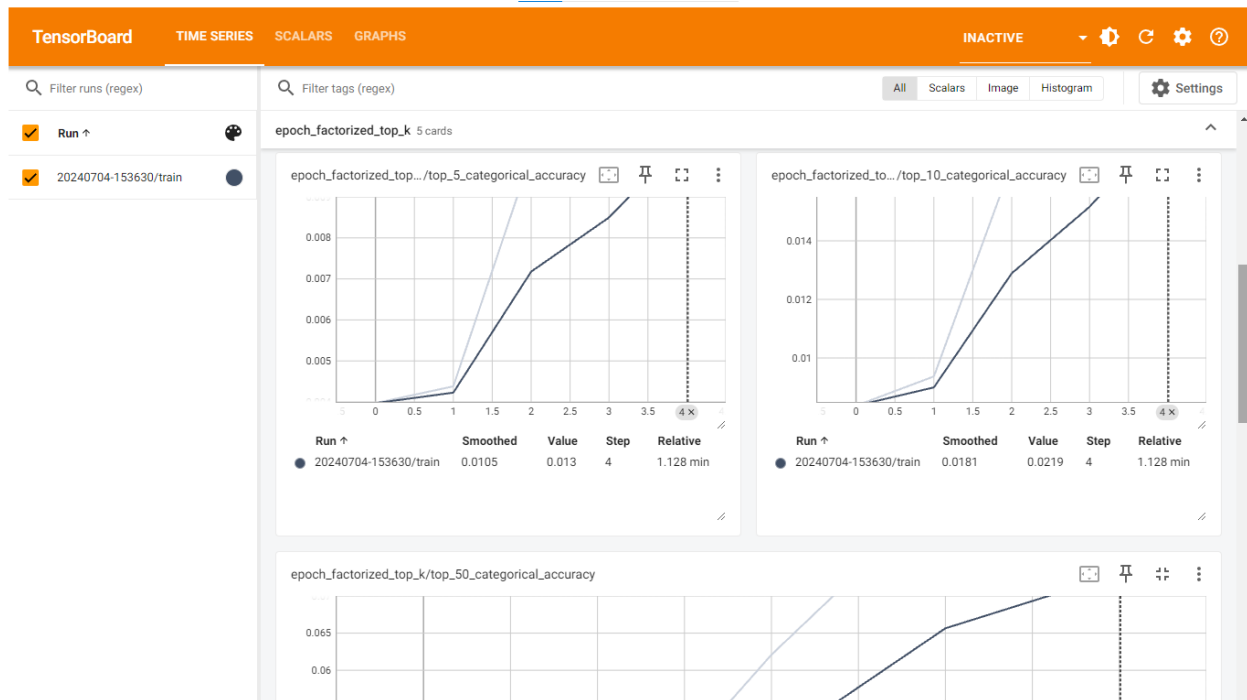
Model 2: Includes an additional dense layer with softmax activation, making it more complex.

**Impact on Outputs:** Activation Function Impact: The choice of activation functions (relu vs. swish) affects how the model learns and captures non-linear relationships. Swish activation can lead to smoother and potentially more accurate representations as it tends to perform better in deep learning models due to its non-monotonic nature.

**Output Layer Impact:** The inclusion of a softmax layer in Model 2 means the model outputs a probability distribution over classes. This can lead to a different approach in how the model interprets and ranks the movie features, affecting the final recommendations.



## Sample Screen Snip of TensorBoard



## Conclusion

In conclusion, the **Movie Recommendation using TensorFlow Recommenders and Visualizing Performance using TensorBoard** project underscores the efficacy of TensorFlow Recommenders for developing sophisticated recommendation systems. By harnessing the capabilities of TensorBoard, stakeholders can gain deeper insights into model behavior and performance, thereby making informed decisions for enhancing recommendation algorithms.

This project serves as a foundational guide for enthusiasts and practitioners looking to delve into recommendation systems, emphasizing the importance of both model construction and performance visualization through TensorBoard.