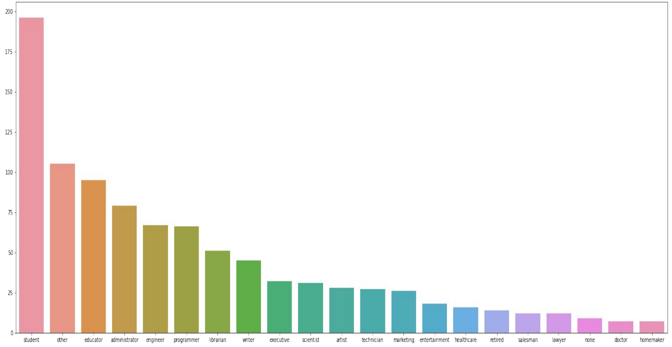
Recommendation System

MovieLens Dataset

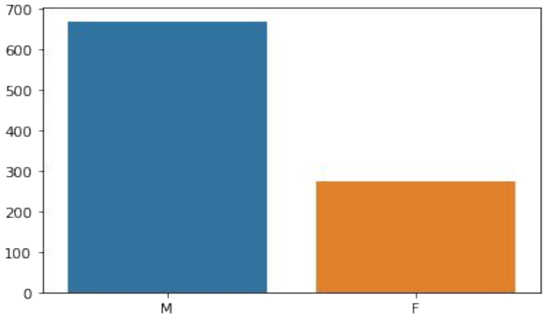
Exploratory Data Analysis

Users Data

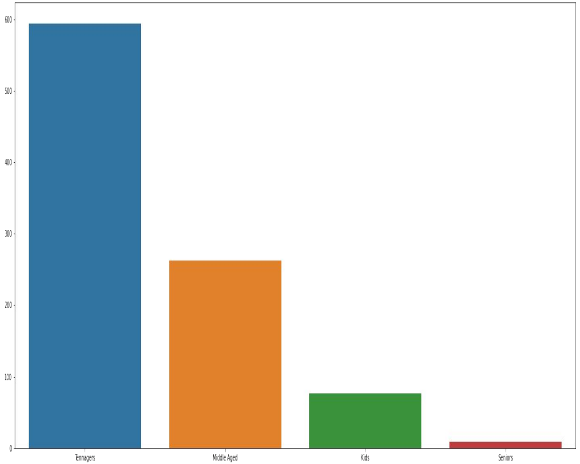
No. Of Users :- **943**



No. of users of the different profession



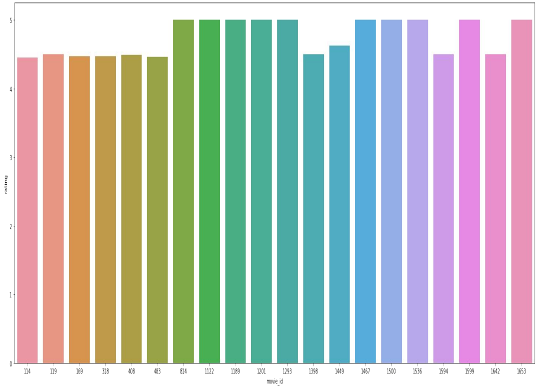
No of users of different generation



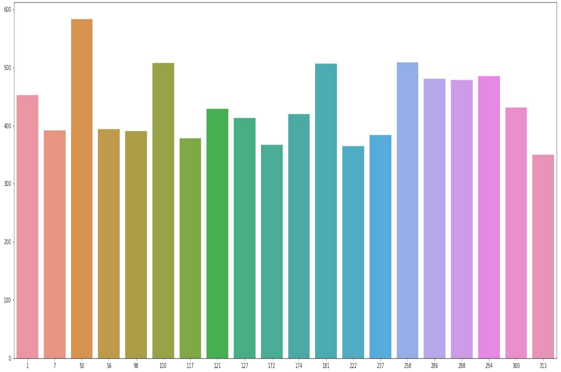
No. of users of different age groups

Ratings Data

No. Of Movies :- **1682**



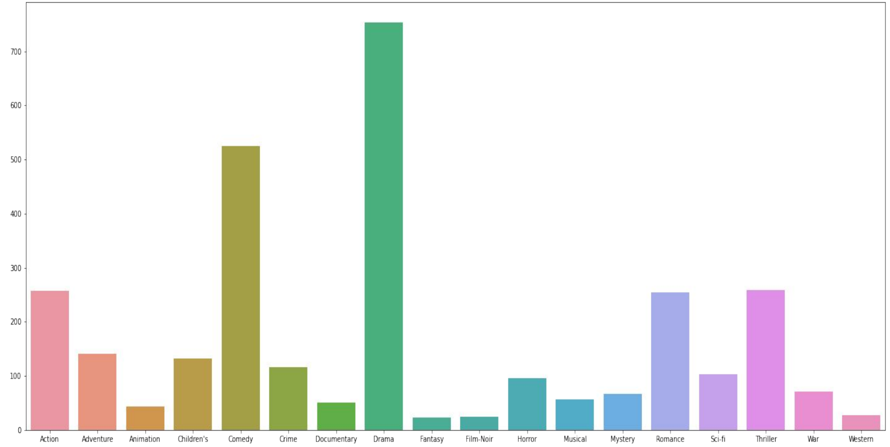
Top 20 rated movies



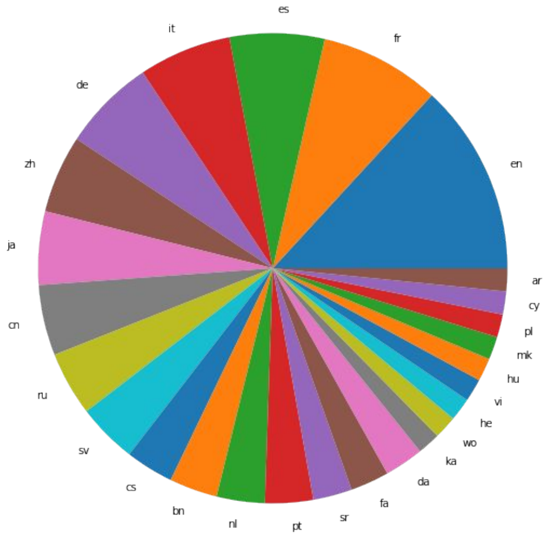
Top 20 most viewed movies

Items Dataset

No. of items :- **100000**



Number of movies in different genres



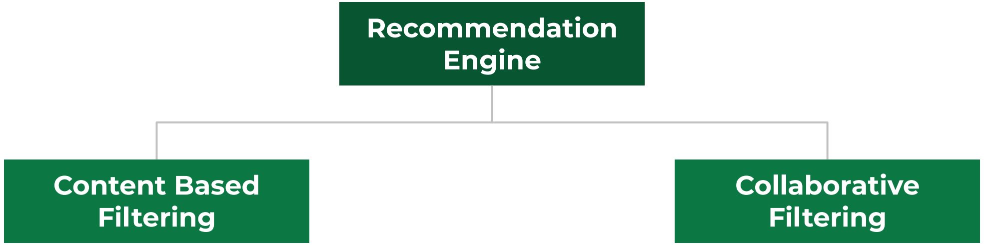
Number of movies in different languages

Model Explanation

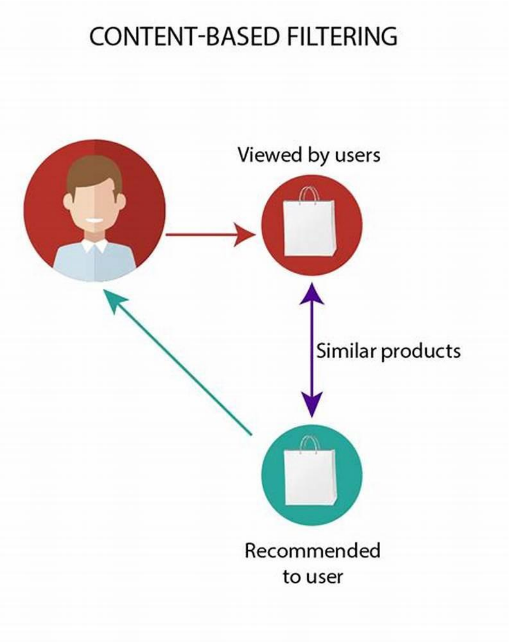
Train Test Split

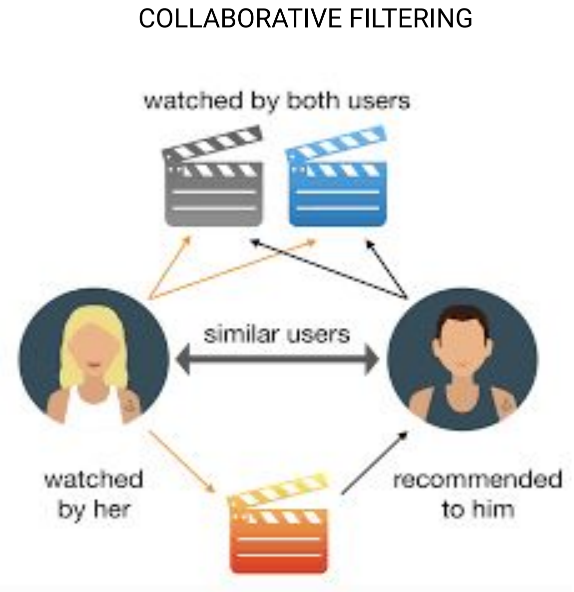
Train Data=60 %
Test Data=20%
Validation Data = 20%

Cadidate Generation



- ★ **Content-based filtering** :- Uses *similarity between items* to recommend items similar to what the user likes.
- ★ **Collaborative Filtering** :- Uses *similarities between queries and items simultaneously* to provide recommendations.





Main key points :-

- Embedding Space
- Similarity (Cosine , Dot product and Euclidean Distance)

Higher the similarity the more likely it is to recommend .

Collaborative Filtering

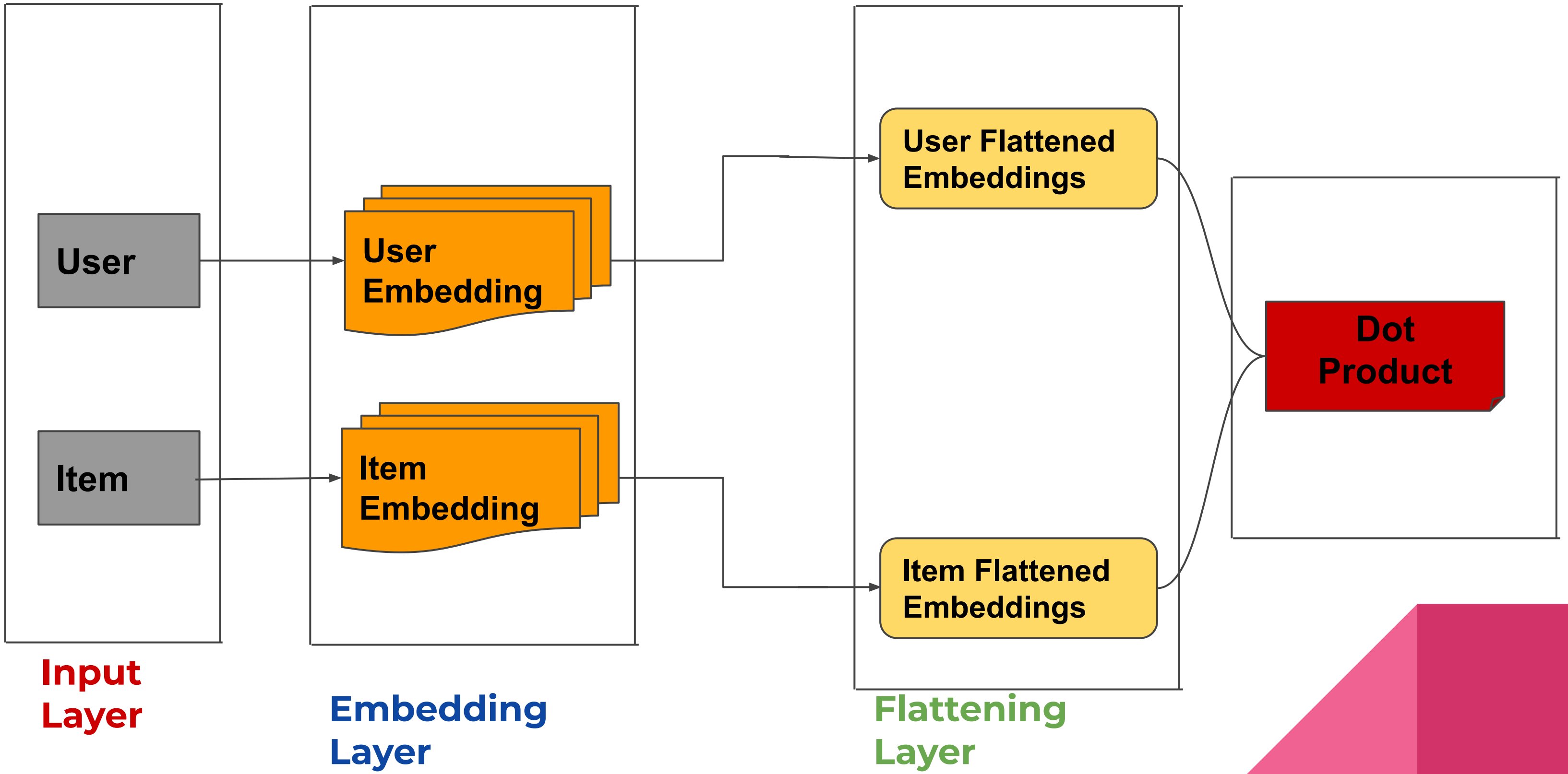
Objective Function :--
$$\min_{U \in \mathbb{R}^{m \times d},\, V \in \mathbb{R}^{n \times d}} \sum_{(i,j) \in \text{obs}} (A_{ij} - \langle U_i, V_j \rangle)^2 .$$

$U \in R^{m \times d}$ is the feature embedding matrix of the user.
$V \in R^{n \times d}$ is the feature embedding matrix of the item.
Embeddings are learned such that the product UV^T is a good approximation of A.

Loss Function :- The loss function used here is Mean Squared Loss

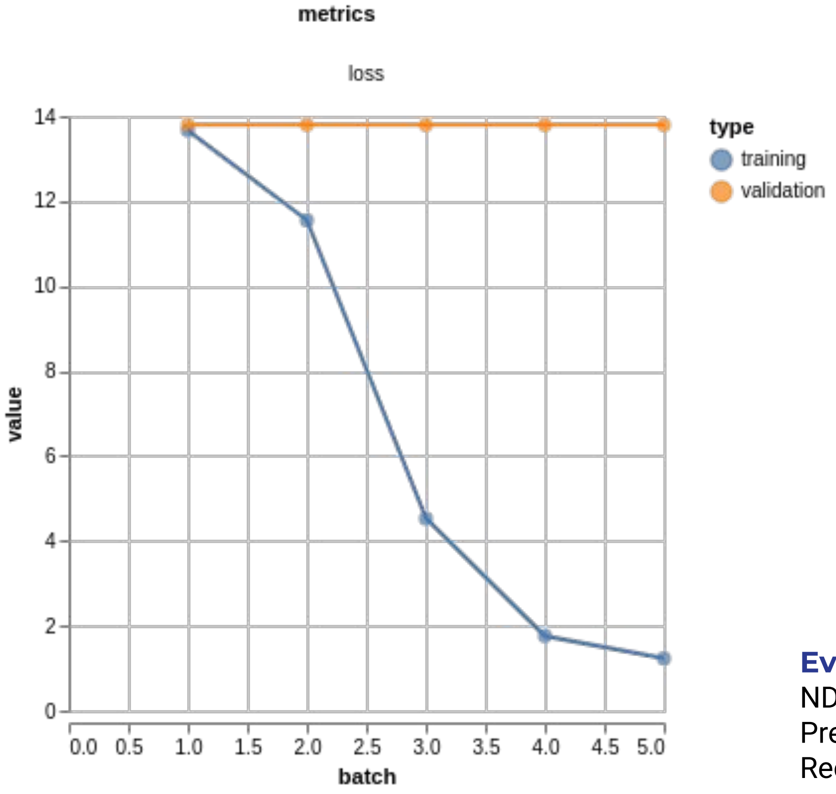
Minimization of the objective function is mainly done by :--
Stochastic Gradient Descent , Weighted Average Least Square or Adam.

MODEL ARCHITECTURE WITHOUT BIAS

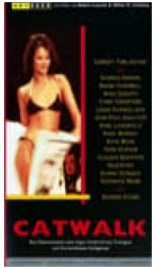


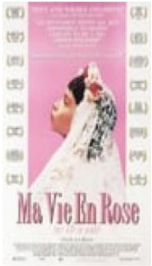
Evaluation And Output

Training Vs. Evaluation



Output





Evaluation:-

NDCG@K:	0.36639
Precision@K:	0.33193
Recall@K:	0.4252

Collaborative Filtering

Objective Function :--

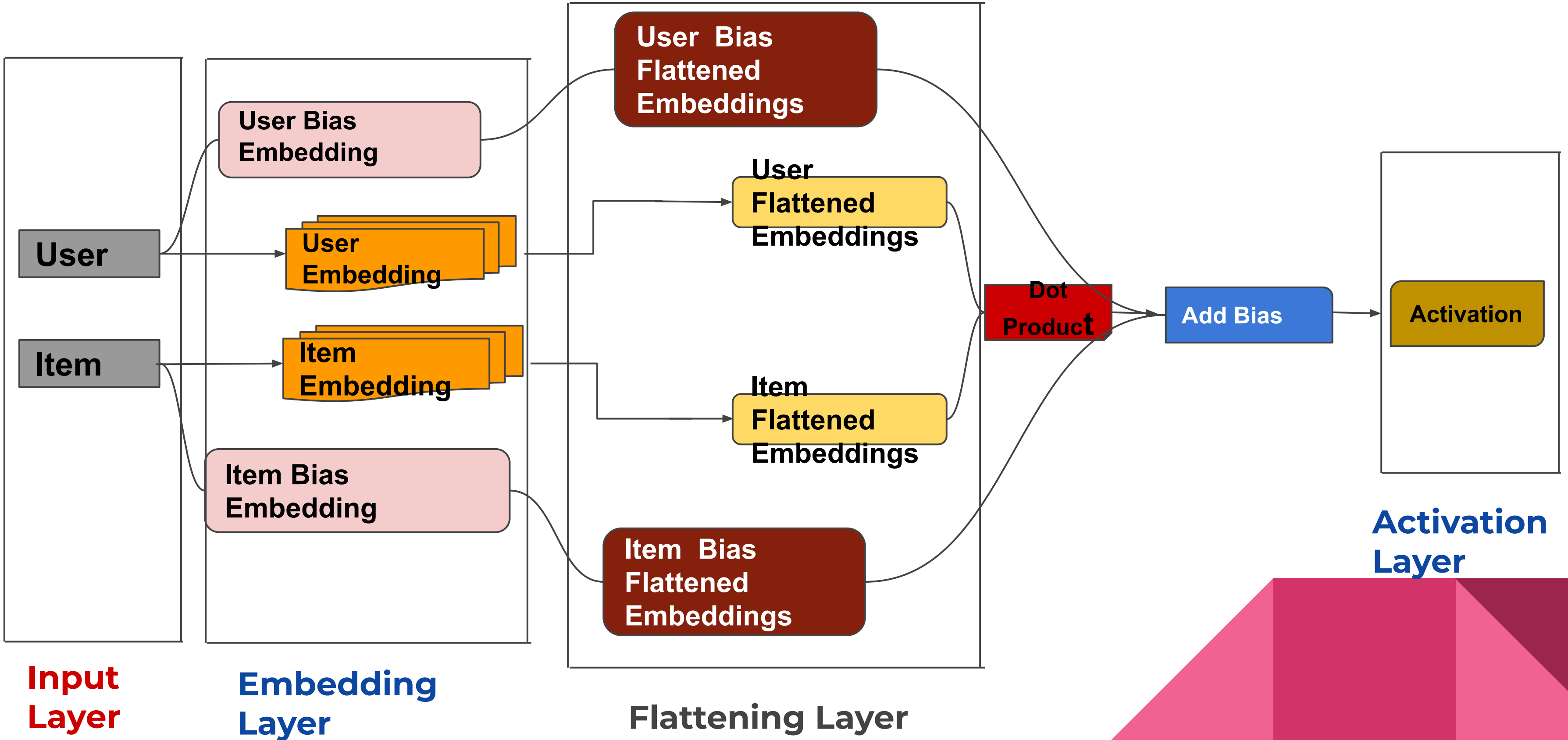
$$\min_{U\in\mathbb{R}^{m\times d},\, V\in\mathbb{R}^{n\times d}} \sum_{(i,j)\in\text{obs}} (A_{ij} - \langle U_i, V_j\rangle)^2 + w_0 \sum_{(i,j)\notin\text{obs}} (\langle U_i, V_j\rangle)^2$$

Here we will mainly use l2 regularisation techniques to add bias to the model .
W_0 is the hyperparameter that needs to be trained well.

Loss Function :- The loss function used here is Mean Squared Loss

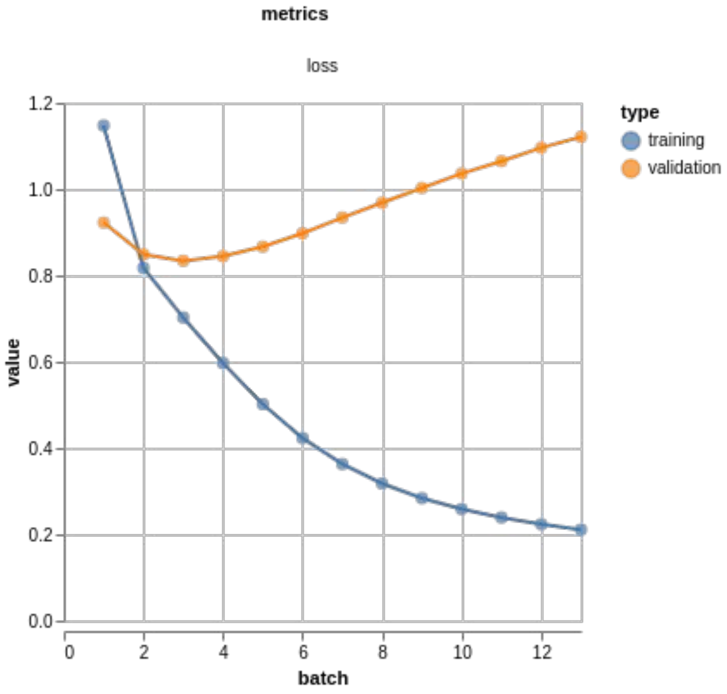
Minimization of the objective function is mainly done by :--
Stochastic Gradient Descent , Weighted Average Least Square or Adam.

MODEL ARCHITECTURE WITH BIAS



Evaluation And Output

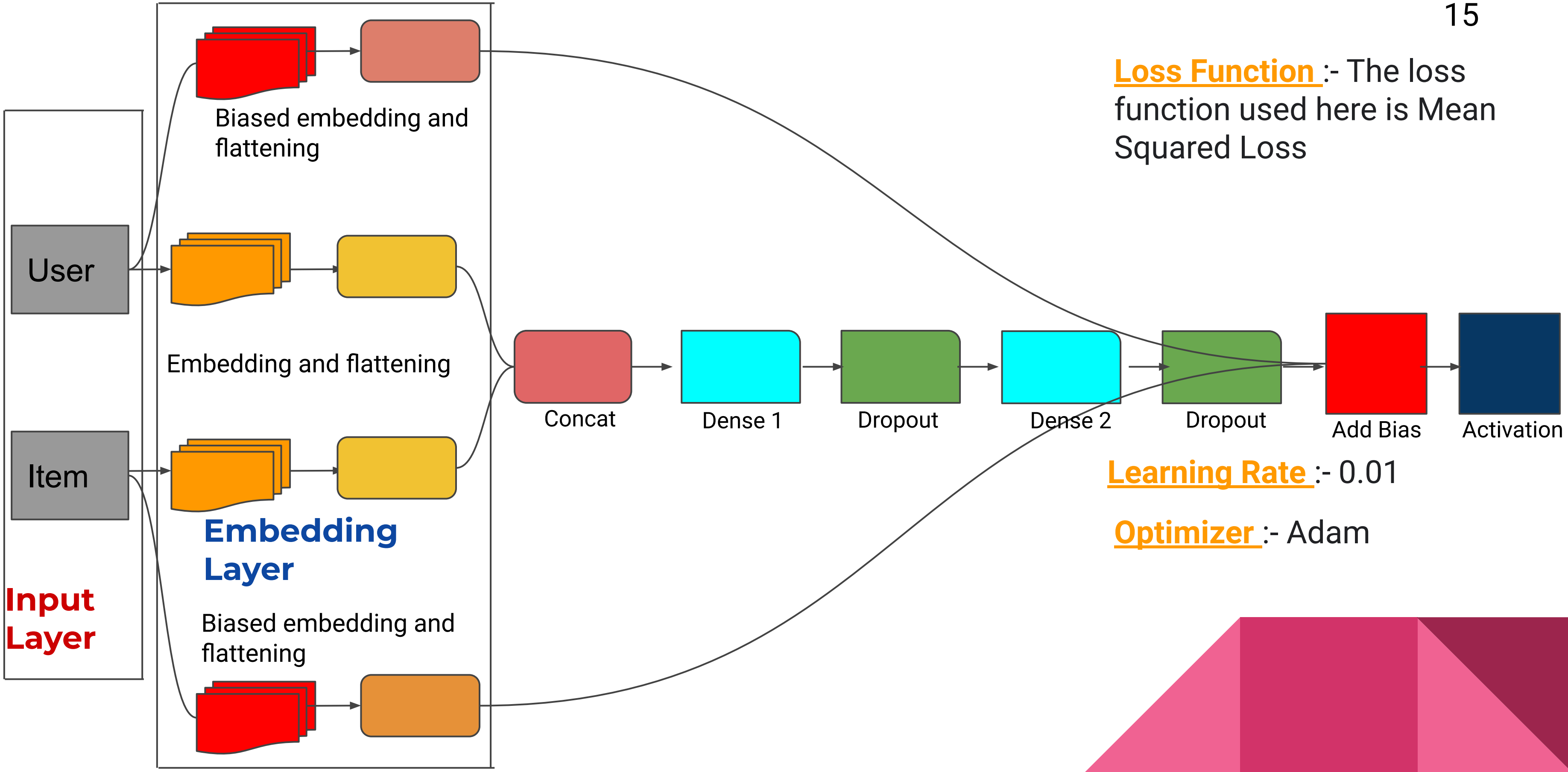
Training Vs. Evaluation



Output

Evaluation:-
NDCG@K: 0.50328
Precision@K: 0.42238
Recall@K: 0.47983

MODEL ARCHITECTURE DEEP MATRIX FACTORISATION



Loss Function :- The loss function used here is Mean Squared Loss

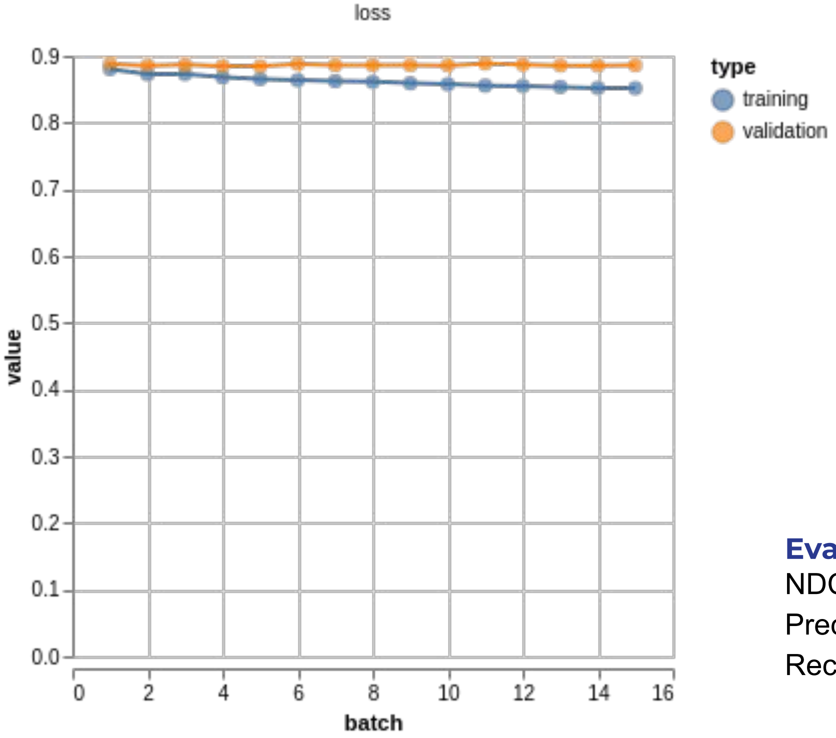
Learning Rate :- 0.01

Optimizer :- Adam

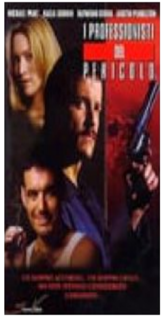
Evaluation And Output

Training Vs. Evaluation

metrics



Output



Evaluation:-

NDCG@K: 0.5873

Precision@K: 0.5635

Recall@K: 0.5810

Deep Neural Network Models

Softmax Model :- treats the problem as a multiclass prediction problem in which :

- Input -- User Query
- Output-- Probability vector with size equal to the number of items in the corpus

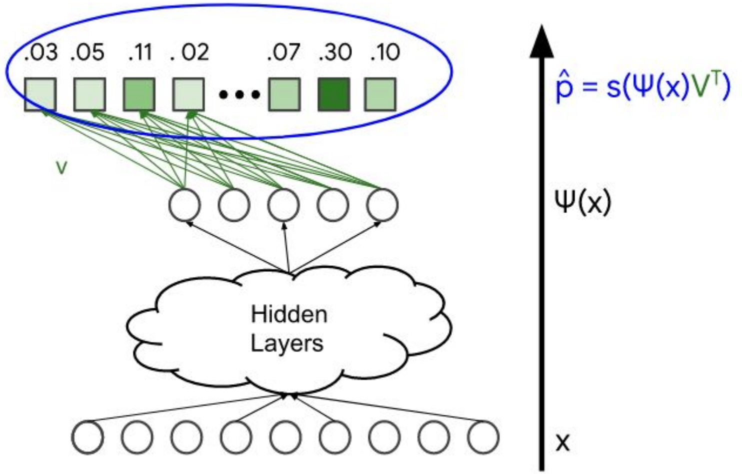
Input :-

- Dense Features
- Sparse Features

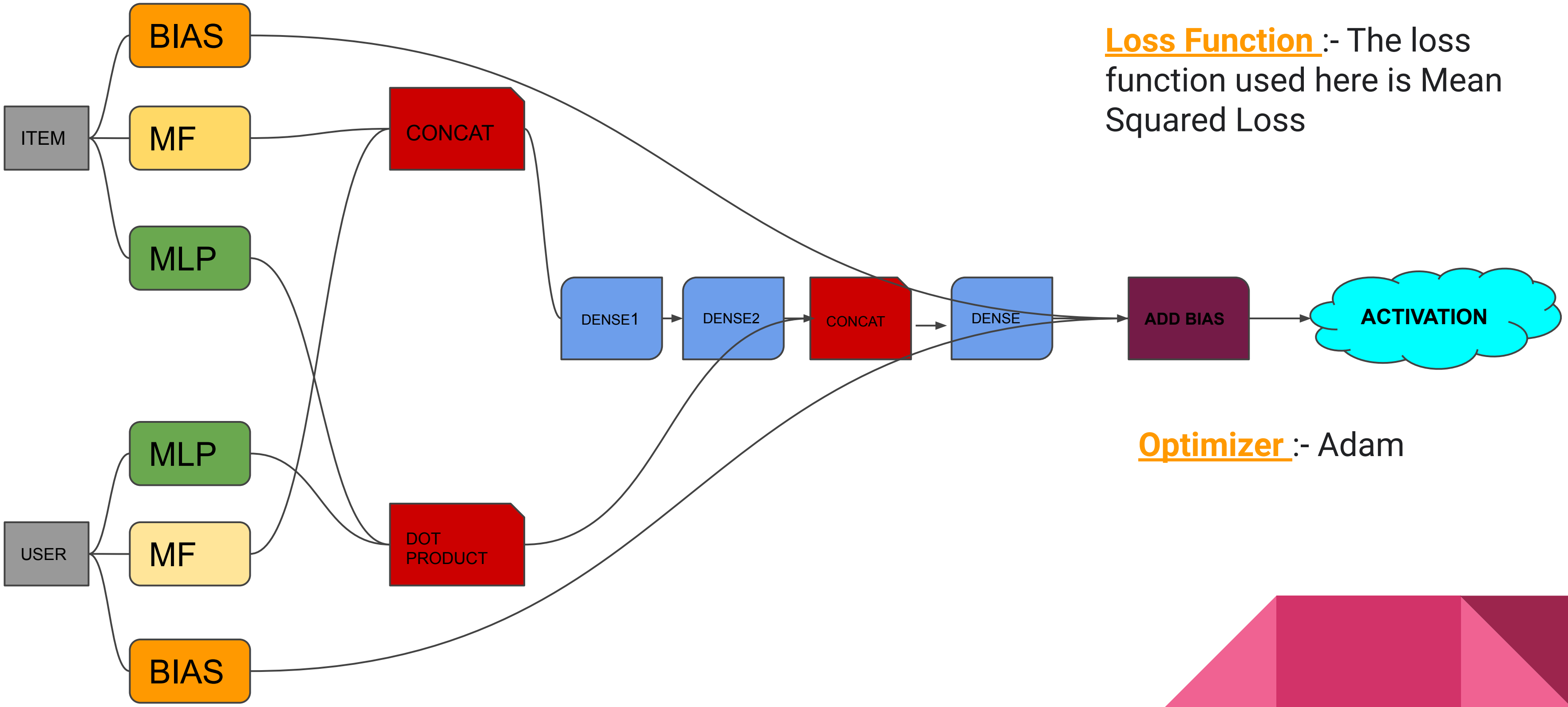
Unlike Matrix Factorisation we can also include side features .

Demo Of Mechanism :-

Main Catchpoint:- The Neural Network passes the last layer through a softmax layer to get the output in terms of probability.



MODEL ARCHITECTURE NEURAL NETWORK

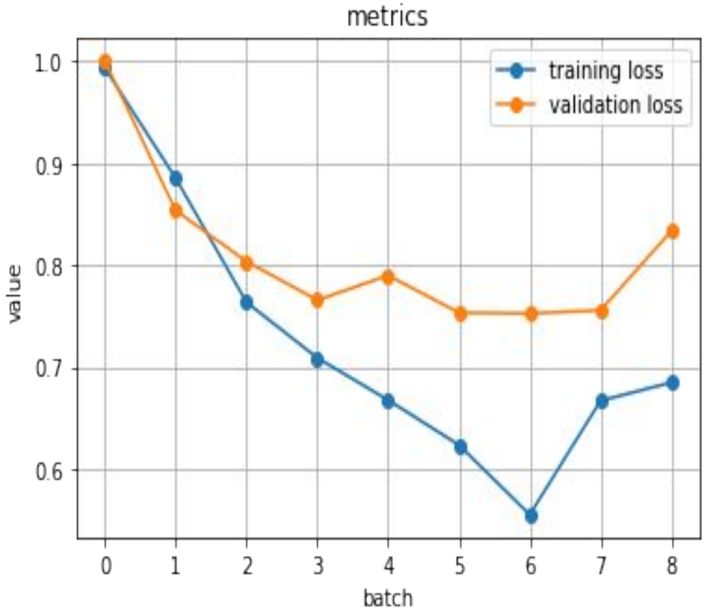


Loss Function :- The loss function used here is Mean Squared Loss

Optimizer :- Adam

Evaluation And Output

Training Vs. Evaluation



Output

Evaluation:-
NDCG@K: 0.65893
Precision@K: 0.6235
Recall@K: 0.50758

Thank You

What Is NDCG ?

Discounted cumulative gain is a measure of ranking quality. In information retrieval, it is often used to measure effectiveness of web search engine algorithms or related applications.

Search result lists vary in length depending on the query. Comparing a search engine's performance from one query to the next cannot be consistently achieved using DCG alone, so the cumulative gain at each position for a chosen value of should be normalized across queries. This is done by sorting all **relevant** documents in the corpus by their relative relevance, producing the maximum possible DCG through position , also called Ideal DCG (IDCG) through that position.

$$DCG_n = \sum_{i=1}^{n} \frac{rel_i}{\log_2^{i+1}},$$

$$NDCG_n = \frac{DCG_n}{IDCG_n},$$