# *Recommendation*
# Engine

A case study to implement a recommendation engine by using historical data for a video sharing platform.

***Bidhya Pokharel***

***November 2021***

# CONTENT

- ***Project's*** *possible Solutions*
- *What might the* ***ACTUAL DATA (Database design)*** *look like?*
- *Our* ***SIMPLIFIED*** *model*
- ***CONCLUSION*** *&* ***NEXT STEP***
- ***FUTURE WORK*** *and* ***POSSIBLE SITUATIONS*** *with* ***SOLUTIONS*** *&* ***ARCHITECTURE***

# POSSIBLE SOLUTIONS FOR PROJECT.

APPROACHES I

*The approached project can be actually solved using different approaches. Let me list 3 of them:*

*1. Python Correlation (Easy)*
*2. ML Algorithm (Intermediate)*
*3. Deep Neural Network (Advanced and Complex)*

# Selected Approach For Instance.

APPROACHES II

*For respective project, Considering following conditions, I've decided to go through ML Algorithm:*

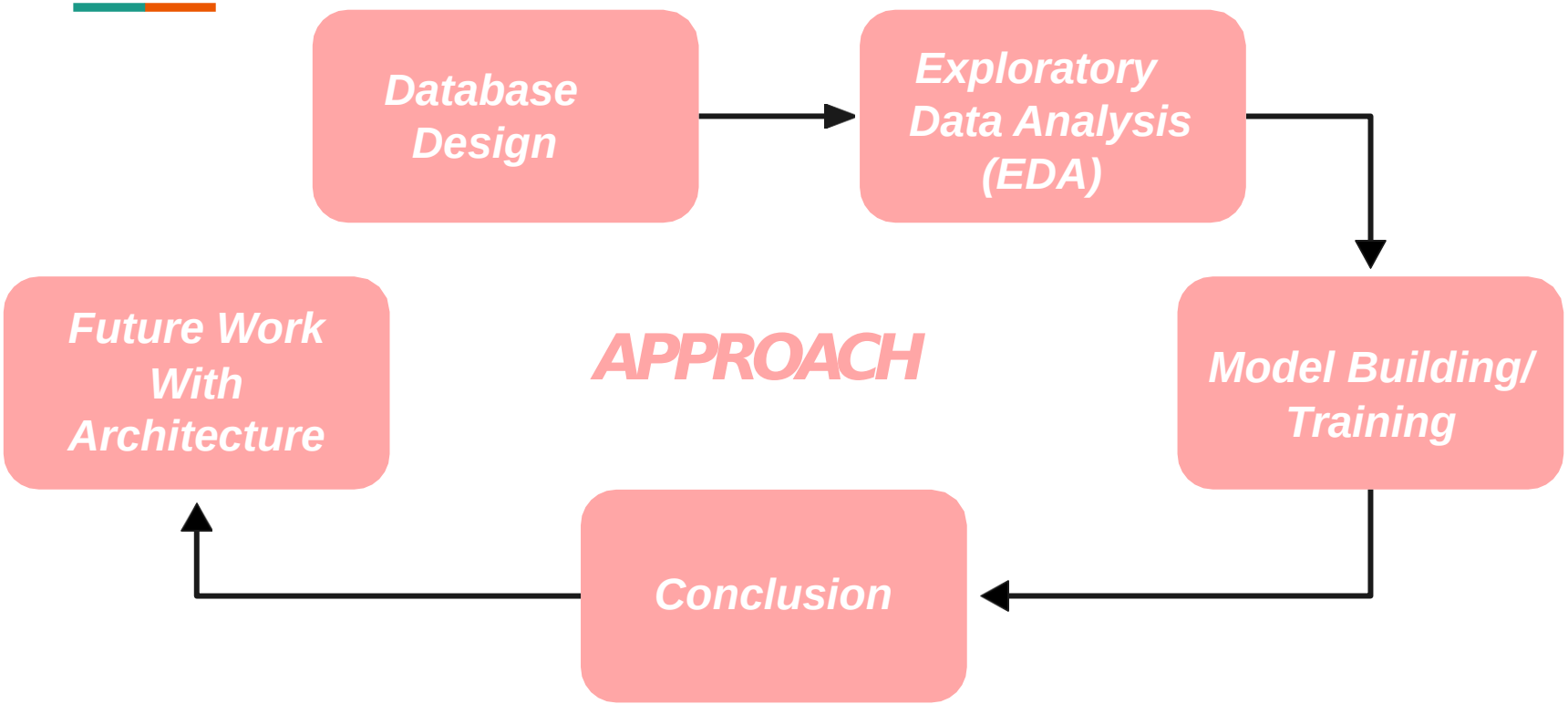
*1. We've collected ratings.*
*2. We've limited amount of ratings and items(videos).*

*( If we do not match expected conditions then we'll go through advanced approach.)*

# ML Algorithm, Why and Which?

- Just because the architecture is advanced does not mean the solution is optimal. So, if we can meet our requirement with ML Algorithm then why not?

- We'll use Nearest Neighbor algorithm based on collaborative filtering because it calculates cosine_similarity to predict and recommend the similar videos as like in Collaborative filtering (Best solution one has ever come to, for recommendation engine project).


Database Design
Exploratory Data Analysis (EDA)
Model Building/ Training
Conclusion
Future Work With Architecture
APPROACH

# Database Design

- I’ve created the database considering we’ll ask the user to rate the videos, so for now I’ll be preparing database as per that.

- But again I came up with different conditions, like what if user watched the video but didn’t rate/interact with videos. So, I’ve prepared the database for these sort of imaginative conditions in future work.

# Data set and Features.

Database Tables

*We'll have two table:*

*1. Videos:*
- *video_id*
- *title*
- *type/genre*

*2. Ratings:*
- *user_id*
- *video_id*
- *rating*
- *timestamp*

# Data Pre processing

I’ve collected the dummy data from internet but though Let’s check how much messy is our data.

# Data Analysis On Merged Features.

Database Analysis I

*Data set merged to do further analysis:*
*There are different merged table, let me keep one as example:*

| | userId | movieId | rating | title | totalRatingCount |
|---|---|---|---|---|---|
| **0** | 1 | 1 | 4.0 | Toy Story (1995) | 215 |
| **1** | 5 | 1 | 4.0 | Toy Story (1995) | 215 |
| **2** | 7 | 1 | 4.5 | Toy Story (1995) | 215 |
| **3** | 15 | 1 | 2.5 | Toy Story (1995) | 215 |
| **4** | 17 | 1 | 4.5 | Toy Story (1995) | 215 |

# Data Analysis On Merged Features.

Database Analysis I

*Since the data set contains dummy value. We've a very clean data set:*

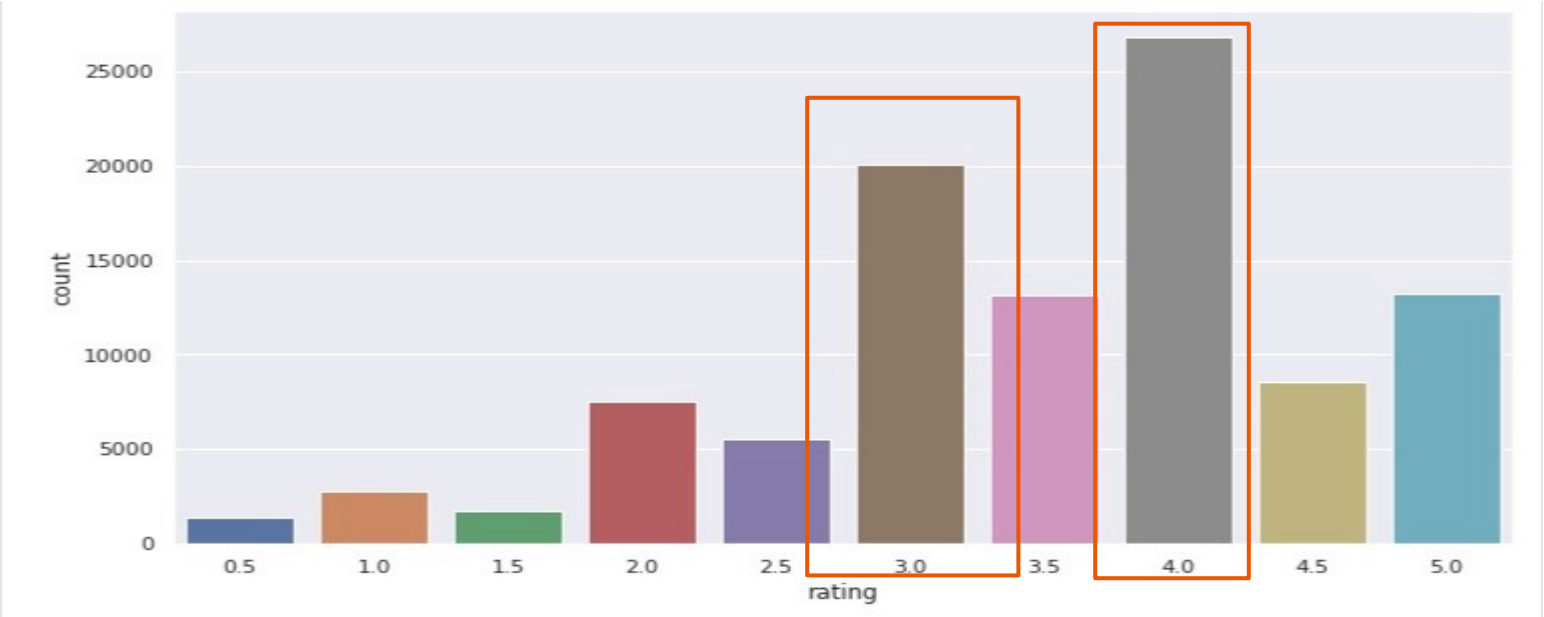
1. *No null values*
2. *No duplicate values*
3. *No strings / messy contents to work with.*
4. *No need to create any target values.*
5. *We've little bit of outliers. Its' very little so we won't deal with it for now.*

# Exploratory Data Analysis

What the data told us? Lets’ go for an EDA on the data set.

# Summary of Bar plot

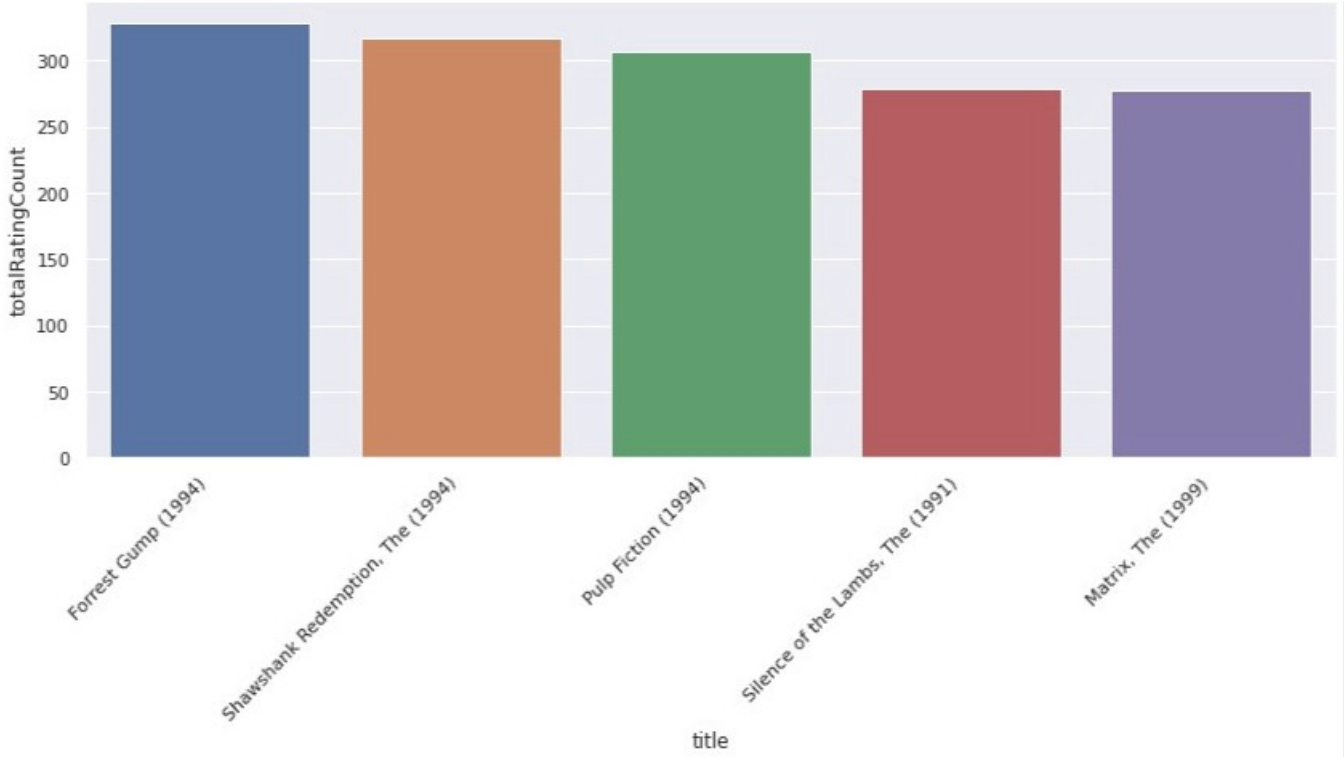
- *We can see most of the videos has got Rating '3' and '4'.*
- *Low rating are really less where as High ratings are quite neutral..*



# MOST POPULAR VIDEOS AMONG USER.

SUMMARY

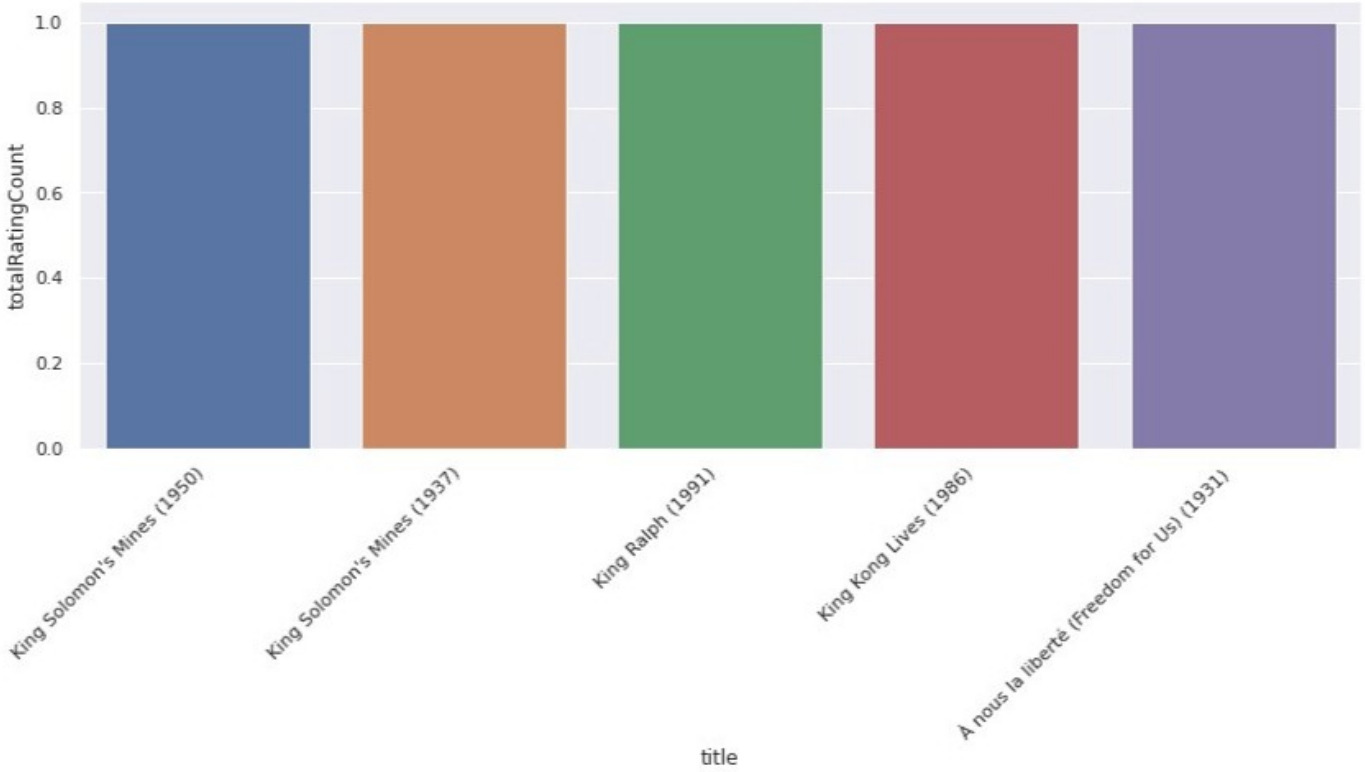
Let’s visualize the most popular videos.



# MOST LEAST INTERACTED VIDEOS AMONG USER.

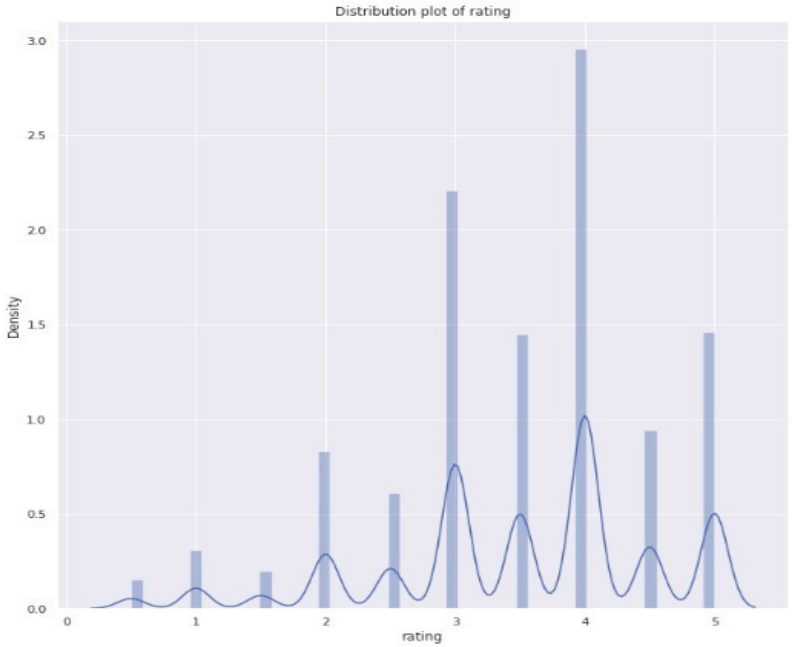
SUMMARY

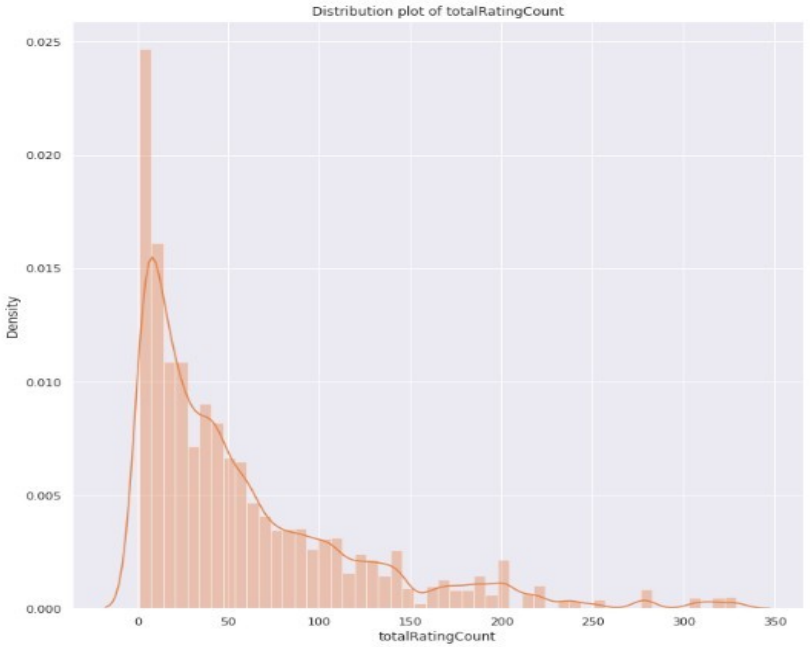
Let’s visualize the lest interacted videos.



# Distribution Plot

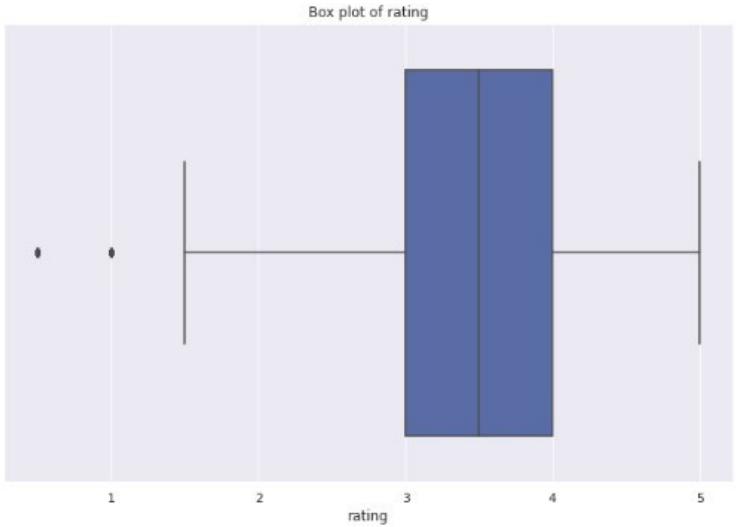
- *It’s understanding why we got such plot for Rating.*
- *But for total_Rating, it seems the distribution is not Gaussian but negatively skewed.*

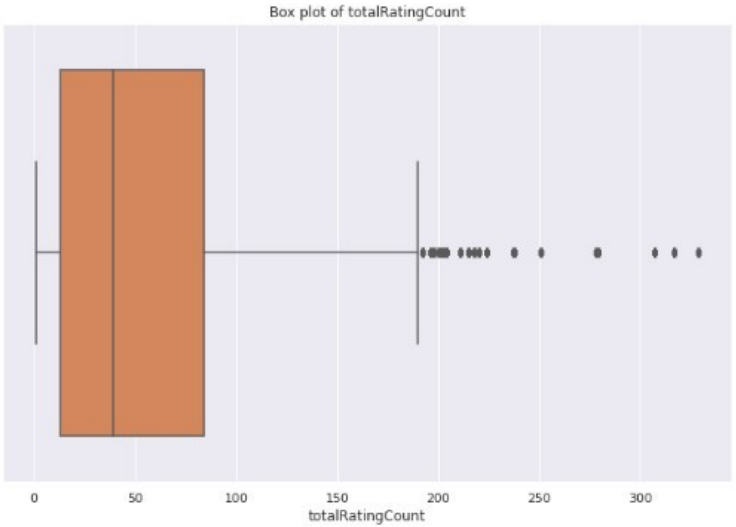


# Box Plot :

- *Seems like rating has few outliers.*
- *We can treat the outliers using different methods like IQR.*



# PREDICTION MODEL

Using Nearest Neighbour: It is the unsupervised machine learning algorithms which calculates the cosine_similarity among the (items,items) and (user,user) to get the nearest one just like collaborative filtering.

# MODEL Building/Training

*Nearest Neighbour was selected for a model. Lets' draw a threshold line as Some movies might be rated by very few persons like 1 or 2.*

- *So, because of this the mean of rating is highly going to be biased, so lets get the rating of only those videos which has received more than certain number.*
- *If it will pass specific threshold, then we'll take the video into consideration.*

```
popularity_threshold = 50
rating_popular_video= data_totalRatingCount.query('totalRatingCount >= @popularity_threshold')
rating_popular_video.head()
```

*So, After filtering the data, we can go for algorithm:*

```
from sklearn.neighbors import NearestNeighbors


model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model_knn.fit(video_features_matrix)
```

# PERFORMANCE EVALUATION

As for Hyper parameter Tuning/ Evaluation metrics, We are not doing any for instance.

# Output

- We are considering user is watching the certain video by passing a value using random function of python.
- After getting that specific value, the model predicts and shortlists the videos users might like.
- For now, we have asked to return 5 videos only. So, that is how we're getting the list of 5 videos only.

```
Recommendations for Tron (1982):

1: RoboCop (1987), with distance of 0.4333110059193197:
2: Total Recall (1990), with distance of 0.5267681600937928:
3: Star Wars: Episode I - The Phantom Menace (1999), with distance of 0.5270073286035528:
4: Predator (1987), with distance of 0.5352344047252271:
5: Blade (1998), with distance of 0.5359050654742769:
```

# *Model Deployment*

Instantly, I’ve not done deployment.  But Flask can be used for deployment.

# MODEL DEPLOYMENT

## Tools:

Flask, HTML,
CSS, JS

## Input

video_id (Recently being watched)

**Summary**

*Let's me explain little about how the deployment part is going to work:*

*1. We'll pickle the trained model.*

*2. From Front end, we'll track and send the id of the video which user is currently watching.*

*3. We'll pass the id to back end (Flask framework), where we'll also unpickle the trained model.*

*4. So, after getting the respective video_id,model will send the list of videos to be recommended which we can finally display in front end.*

# *Conclusion*

Hence, this way we can predict the user preferences using historical data of videos and items.

# FUTURE WORK & NEXT STEP

# *Future Work:*

## *1. Database Structure:*

I. I’ve built the data set considering we’ll take the rating from users. But Let’s consider we might not want ratings but instead, comment section and like button only like of You tube. Then, during this scenario:
- We might take the count of like buttons.
- And perform sentiment analysis in the content of comment section.

II. But what if the user is watches the video, but does not interact with the video, for example: he/she does not like/dislike, comment, rate and so on, so for this we’ve have to now track the user activities.

# *Future Work:*

## *1. Database Structure:*

So, as mentioned before we're going to create 7 implicit features, tracking the user and item history. Like:

1. **Watch History** (Videos Watched recently)
2. **Search History** ( Queries typed in the search bar)
3. **Location** ( From where exactly is user using the video sharing platform, might come handy if we've international users/customers)
4. **Device type** (If the video sharing platform has different solution, this might be useful. )
5. **Gender**
6. **Age** (Of user)
7. **Video Freshness** (Age of the video)

# *Future Work:*

## *2. Approach:*

Well, I've selected ML algorithm for now considering we won't have very large amount of data sets whether it's item or users. But if we'll have excessive amount of users/items then compiling time is going to be quite long. So, ML algorithm won't be able to give its' best.

So, we might consider using **Deep Neural network** by compressing the items for any particular user with candidate generation.

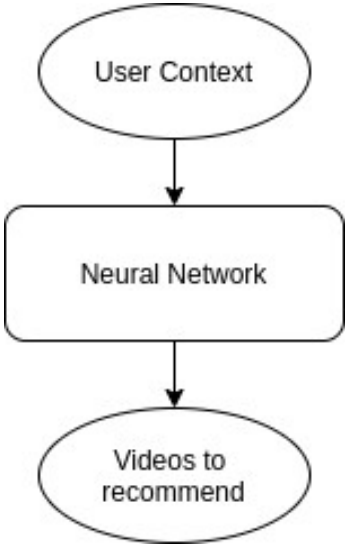



Fig. Next Approaches

# *Related Works:*

Many platforms use recommendation system, to attract more users or to let their customers spend more time in their platforms.

From Amazon to Netflix to Youtube, every platform uses one of the approaches from the mentioned architecture.

As a solution of one problem, new approach is invented but then again it brings the new cons, so on the way of overcoming the cons of one to another, recommendation system has become way too vast now.

*Source: Article published by Google regarding Recommendation engine, 1997.*

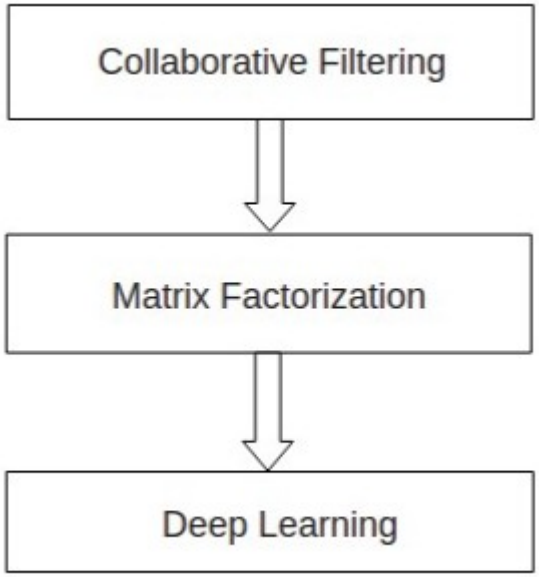



Fig. Recommendation System Approaches

# Recommendation Engine:

Summing up all the points into consideration, our final final recommendation engine architecture will look somethings like the mentioned image.

**1. Candidate Generation:**

We'll use the implicit features mentioned above to filter the images whether to submit it for ranking or not.

$$P(s_i = S \mid \dot{x}_i) = \prod_{Yes} Q(Y|\dot{x}_i) \prod_{Yes} (1 - Q(Y|\dot{x}_i))$$

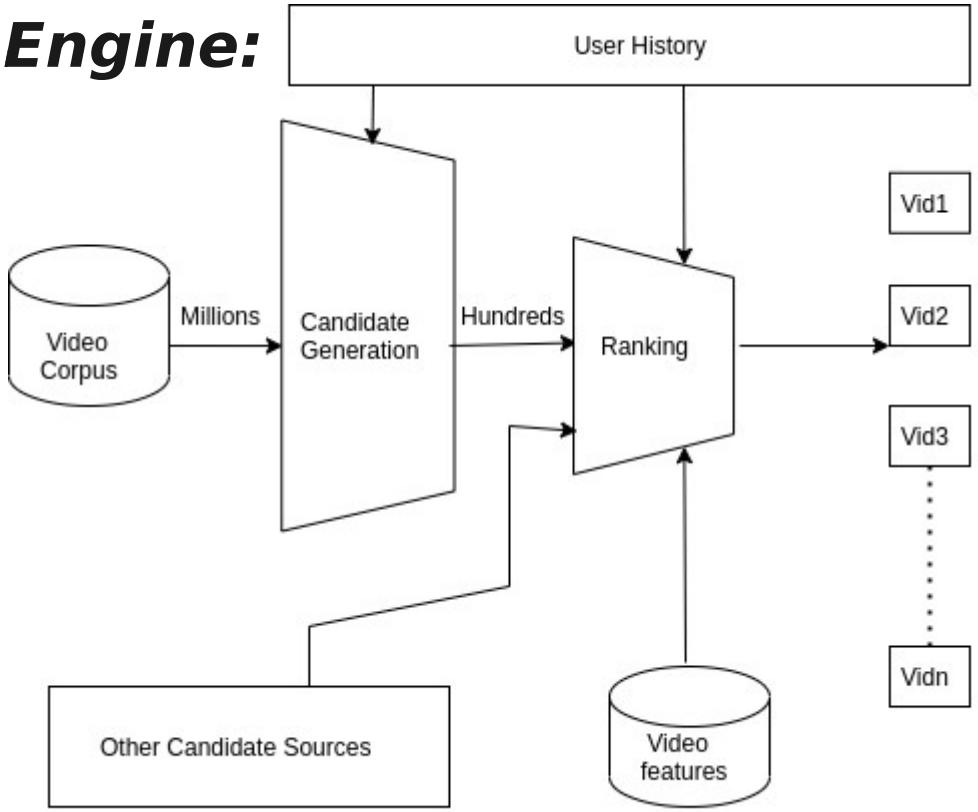



Fig. Recommendation Engine Architecture

# Deep Neural Network Architectute:

As I already showed the recommendation engine architecture in previous slide, now here the math/algorithm behind that architecture.

**2. Ranking:**

We'll perform candidate ranking Before actually sending for ranking using weighted logistic regression, where we calculate the relevance score for each video, where:

Score = Expected Watch time of video

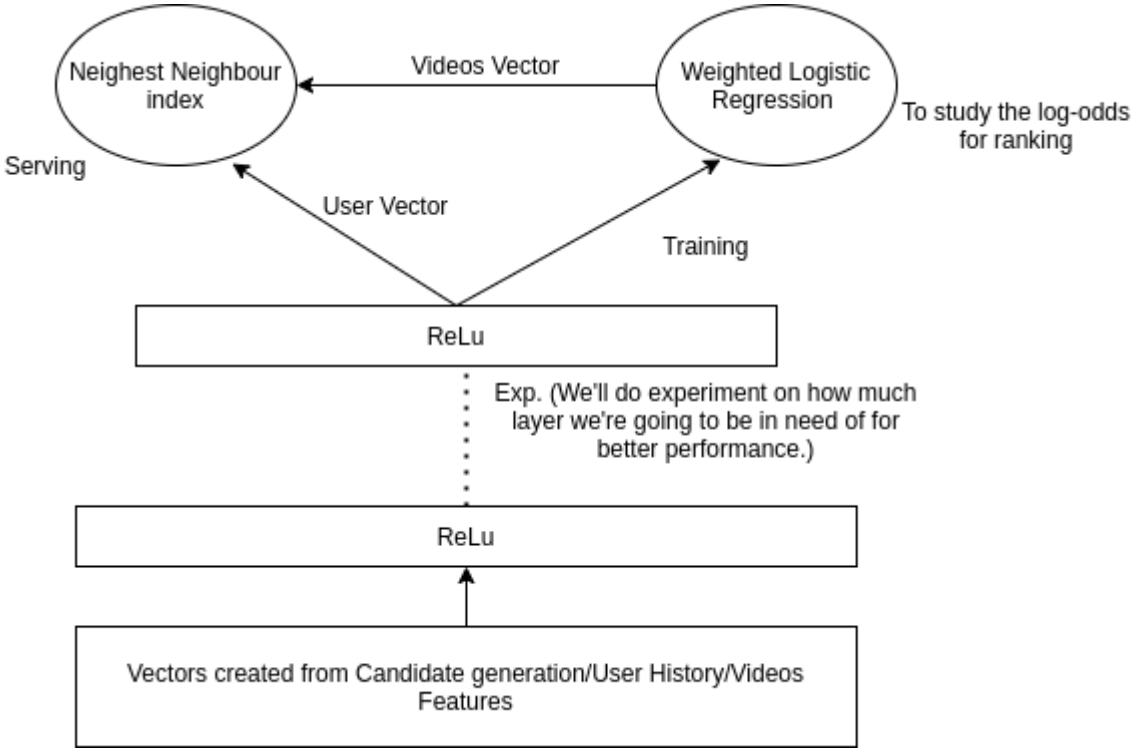



Fig. Recommendation Engine Architecture

# Conclusion

Explaining the whole idea for better understanding with relevant architecture/picture and mathematical expressions might be very lengthy. So, I’m summarizing it here.

Hope, it provides a clear picture about what we’re planing to do for recommendation engine.

Fig. Recommendation Engine Architecture

# END