MOVIE ROULETTE

A HYBRID CHATBOT RECOMMENDER SYSTEM FOR RECOMMENDING MOVIES

“Movie Roulette” is an intelligent chatbot that recommends movies to the users using Machine Learning techniques. It takes some basic information from the user like which movies the user finds interesting, user’s favourite and disliked genres to recommend new movies to the user. The chatbot is trained in such a way that it understands the phrases and sentences (i.e.) the natural language of the user.

PROBLEM FORMULATION:

“Serendipity” – this is the major concern when it comes to most of the movie recommender systems. Users are often shocked to see movies which they dislike (preferably genres) being recommended to them. To overcome these issues, most recommender systems use a hybrid technique in which they use both collaborative and content-based filtering, and they are mostly based on user ratings and reviews. But we have gone another step forward to introduce another filtering technique called genre-based filtering. Thus, Movie Roulette is a hybrid recommender system that makes use of Collaborative-based, content-based, and genre-based filtering techniques to recommend movies.

Our recommender system revolves around the two significant hypotheses.

Hypothesis 1: “Do not recommend the same movies which the user has mentioned to be interesting.”

Explanation:

If the user has mentioned certain movies to be interesting, it means that user might have already watched them. So, there is no point in recommending these same movies and it might cause frustration to the user.

Hypothesis 2: “Do not recommend any movie which belongs to the disliked genre of the user.”

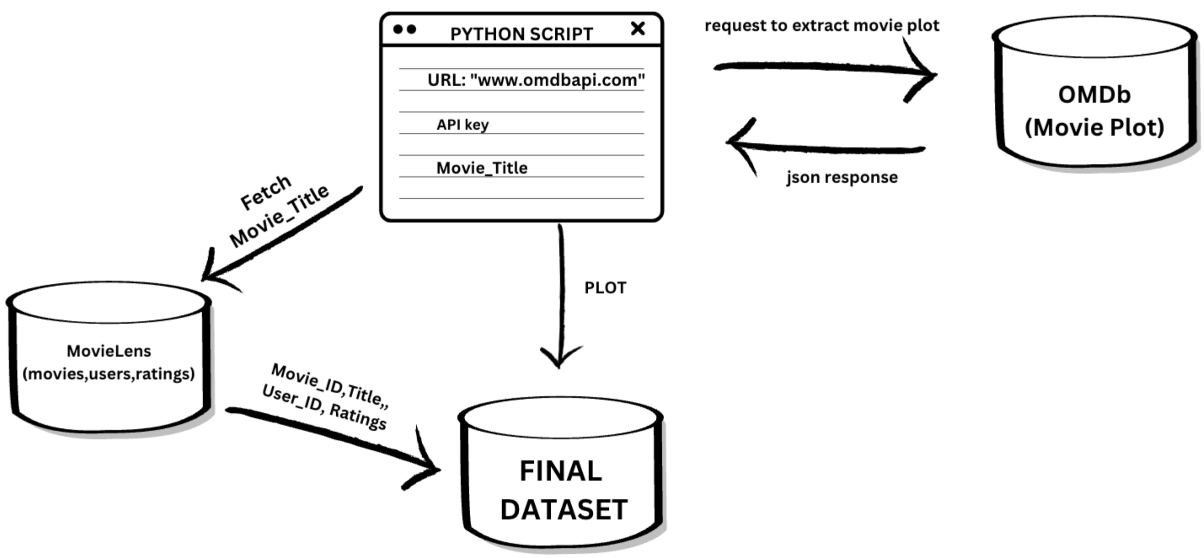
Explanation:

Say, the user likes animation and dislikes horror. The recommender should not recommend any film that belongs to the horror category though the movie falls under the animation genre.

DATA PREPARATION:

To build our recommendation system, we required a very sophisticated dataset that should be suitable to perform all our filtering techniques. Thus, we chose the 'MovieLens' dataset which contains personalized ratings of various movies from many users. The dataset was perfectly suitable to perform the collaborative filtering method as it also contained many user ratings for the same movie. But the MovieLens dataset contained only the basic information of the movie(such as movie name, ID, title, genre) but we required more information such as the plot of the movie to perform content-based filtering.

How did we extract the plot of the movie that was missing in the dataset?



We performed web scrapping to extract the plot summaries of the movies. To achieve this, we purchased the API key of the OMDb dataset. So, we sent a request to the OMDb API with the movie title as a parameter and the response was received in a JSON format which was then appended to the dataset for the corresponding movie titles.

This was necessary because we wanted to give more innovative recommendations based on plots of the movies to the users and incorporating the storyline of each movie was a crucial step in this project.

DATA CLEANING:

We have two dataframes:

```
df1 = pd.read_csv("Movies with Plots.csv")
df2 = pd.read_csv("ratings.csv")
```

The initial dataframe looks like this:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9742 entries, 0 to 9741
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   9742 non-null   int64
 1   movieId      9742 non-null   int64
 2   title        9742 non-null   object
 3   genres       9742 non-null   object
 4   movie_name   9742 non-null   object
 5   Plot         7855 non-null   object
dtypes: int64(2), object(4)
memory usage: 456.8+ KB
```

We found that there were some unnamed columns and thereby we dropped them.

```
df.drop(columns={'Unnamed: 0', 'timestamp'})
```

Also it was found that some plots contained NaN values.

	movieId	title	genres	movie_name	Plot	userId	rating
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	Toy Story	A little boy named Andy loves to be in his roo...	1	4.0
1	3	Grumpier Old Men (1995)	Comedy|Romance	Grumpier Old Men	Things don't seem to change much in Wabasha Co...	1	4.0
2	6	Heat (1995)	Action|Crime|Thriller	Heat	Hunters and their prey--Neil and his professio...	1	4.0
3	47	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller	Seven	A veteran samurai, who has fallen on hard time...	1	5.0
4	50	Usual Suspects, The (1995)	Crime|Mystery|Thriller	Usual Suspects, The	NaN	1	5.0

And hence we dropped those rows as well.

```
df.dropna(subset=['Plot'], inplace=True)
```

And inorder to merge these two dataframes into a single dataframe we used left join on movie ID.

```
df = pd.merge(df1, df2, on='movieId', how='right')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100836 entries, 0 to 100835
Data columns (total 9 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Unnamed: 0   100836 non-null  int64
 1   movieId      100836 non-null  int64
 2   title        100836 non-null  object
 3   genres       100836 non-null  object
 4   movie_name   100836 non-null  object
 5   Plot         83447 non-null   object
 6   userId       100836 non-null  int64
 7   rating       100836 non-null  float64
 8   timestamp    100836 non-null  int64
dtypes: float64(1), int64(4), object(4)
memory usage: 7.7+ MB
```

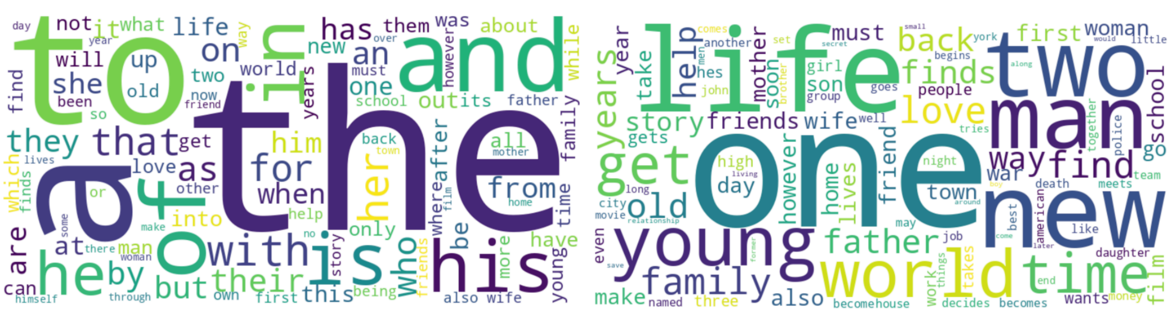
In our project we have created a function called ‘clean_text’ which does some basic cleaning on any text that is given as a parameter to it. The function removes backslash, apostrophes, whitespaces, and all other special characters from the text.

```
# function for text cleaning
def clean_text(text):
    # remove backslash-apostrophe
    text = re.sub("\'", "", text)
    # remove everything except alphabets
    text = re.sub("[^a-zA-Z]"," ",text)
    # remove whitespaces
    text = ' '.join(text.split())
    # convert text to lowercase
    text = text.lower()

    return text
```

And to remove stop words we have used another function.

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

# function to remove stopwords
def remove_stopwords(text):
    no_stopword_text = [w for w in text.split() if not w in stop_words]
    return ' '.join(no_stopword_text)

df['clean_plot'] = df['clean_plot'].apply(lambda x: remove_stopwords(x))
```

This function is used to remove the stop words such as to, is, the, etc., which does not contain much meaning.

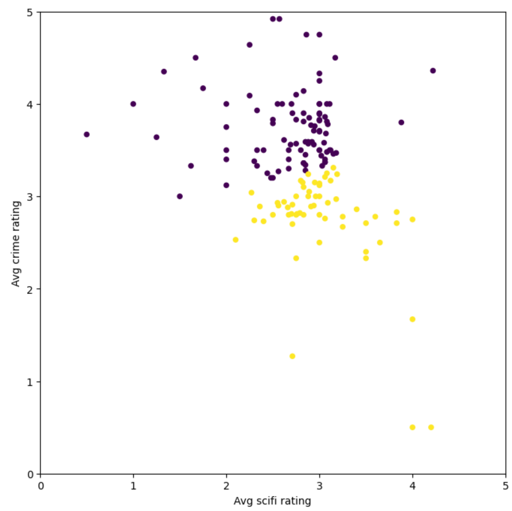


GETTING SOME INSIGHTS ON OUR DATA:

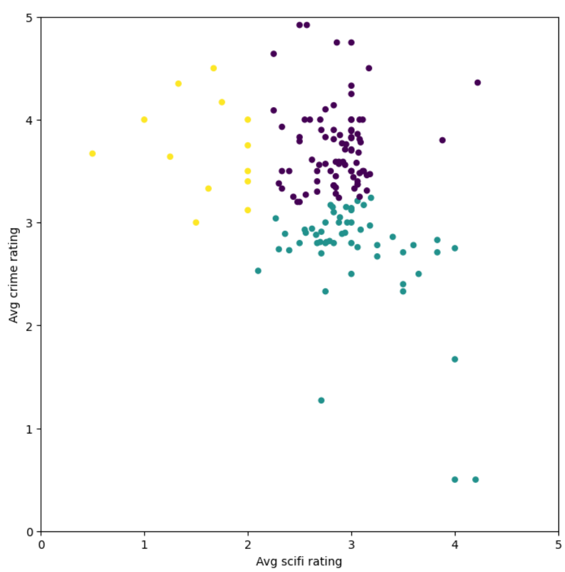
CLUSTERING:

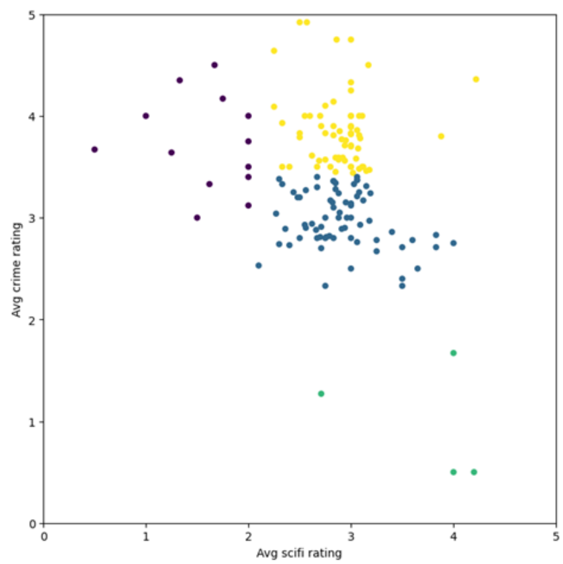
To perform clustering, we utilised the K means clustering and used the Silhouette scores to evaluate the best K value (Number of Clusters)

We utilised the Average crime rating and average sci-fi rating in this case.

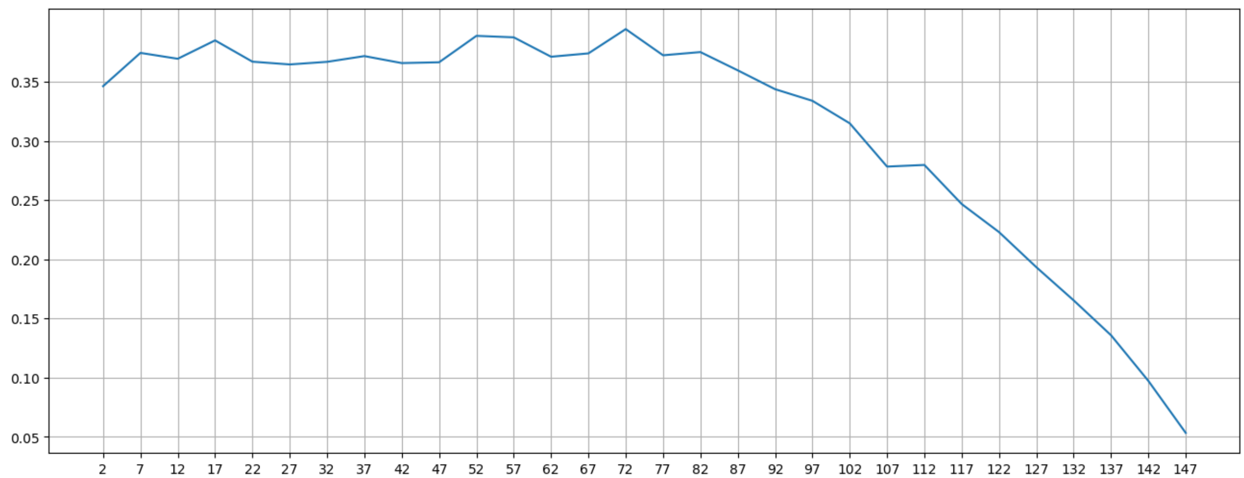


We can see the clustering for K=2, similarly we performed the same for K=3,4

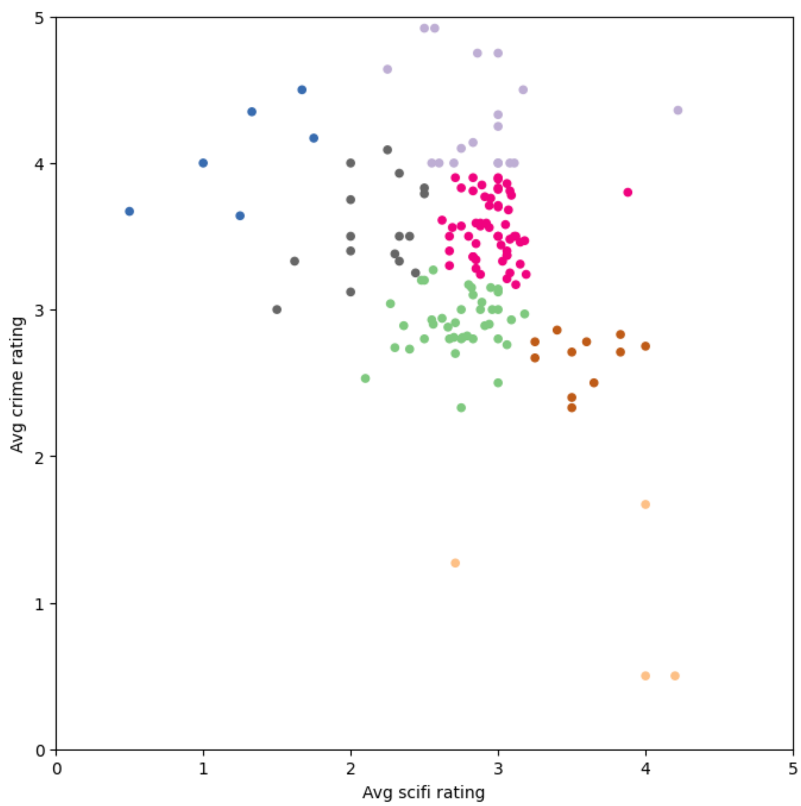




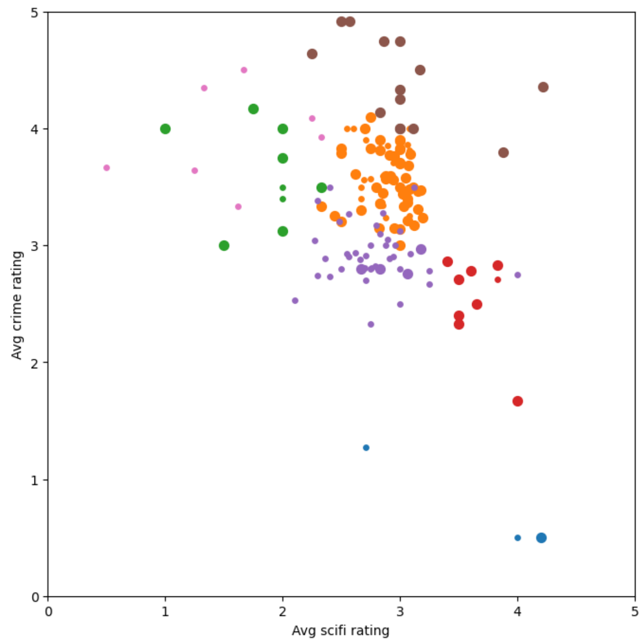
After that we plotted K values with their corresponding Silhouette scores, the graph is shown below.



Looking at this graph, the good choice for k is 7 as it will be easier to visualize, other good values are 17,57 and 77 which are difficult to visualize.
So, this is the visualization with the number of clusters i.e. K value as 7.



And then as the plots were made only for Sci-Fi and Crime movies, we added Action into the mixture by keeping the size of the dots representing the Action movies.



CLASSIFICATION:

For the classification, we made use of the plots to determine the genre of the movies. We made use of Logistic Regression, Decision Tree Classifier and Support Vector Machine with OneVsRestClassifier.

For the genres, we made use of MultiLabelBinarizer. For the plots we pre-processed the data, removed stop words and made use of Bag of words and tf-idf to make the prediction of genres.

Below are the F1 scores for Logistic Regression in Bag of words and Tf-idf

```python
lr = LogisticRegression()
clf = OneVsRestClassifier(lr)
# fit model on train data
clf.fit(xtrain_bow, ytrain)
# make predictions for validation set
y_pred = clf.predict(xval_bow)
LRSCORE_BOW=f1_score(yval, y_pred, average="micro")
print(LRSCORE_BOW)
```

```
0.48722016569716203
```

```python
lr = LogisticRegression()
clf = OneVsRestClassifier(lr)
# fit model on train data
clf.fit(xtrain_tfidf, ytrain)
# make predictions for validation set
y_pred = clf.predict(xval_tfidf)
```

```python
LRSCORE=f1_score(yval, y_pred, average="micro")
print(LRSCORE)
```

```
0.3539166484835261
```

For SVM these were the F1 scores in BOW and TF-IDF

```python
s= SVC(kernel='linear', random_state=0)
clf = OneVsRestClassifier(s)
# fit model on train data
clf.fit(xtrain_bow, ytrain)
# make predictions for validation set
y_pred = clf.predict(xval_bow)
SVMSCORE_BOW=f1_score(yval, y_pred, average="micro")
print(SVMSCORE_BOW)
```

```
0.48686514886164617
```

```python
from sklearn.svm import SVC
s= SVC(kernel='linear', random_state=0)
clf = OneVsRestClassifier(s)
# fit model on train data
clf.fit(xtrain_tfidf, ytrain)
# make predictions for validation set
y_pred = clf.predict(xval_tfidf)
SVMSCORE=f1_score(yval, y_pred, average="micro")
print(SVMSCORE)
```

```
0.4610616040294459
```

For the Decision Tree the f1 scores of BOW and TF - IDF are given below

```python
c = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
clf = OneVsRestClassifier(c)
clf.fit(xtrain_bow, ytrain)
# make predictions for validation set
y_pred = clf.predict(xval_bow)
DTCSCORE_BOW=f1_score(yval, y_pred, average="micro")
print(DTCSCORE_BOW)
```

```
0.38907103825136613
```

```python
from sklearn.tree import DecisionTreeClassifier
c = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
clf = OneVsRestClassifier(c)
# fit model on train data
clf.fit(xtrain_tfidf, ytrain)
# make predictions for validation set
y_pred = clf.predict(xval_tfidf)
```

```python
DTCSCORE=f1_score(yval, y_pred, average="micro")
print(DTCSCORE)
```

```
0.38027952695438483
```

We further generated a classification report for one of the models, which is shown below.

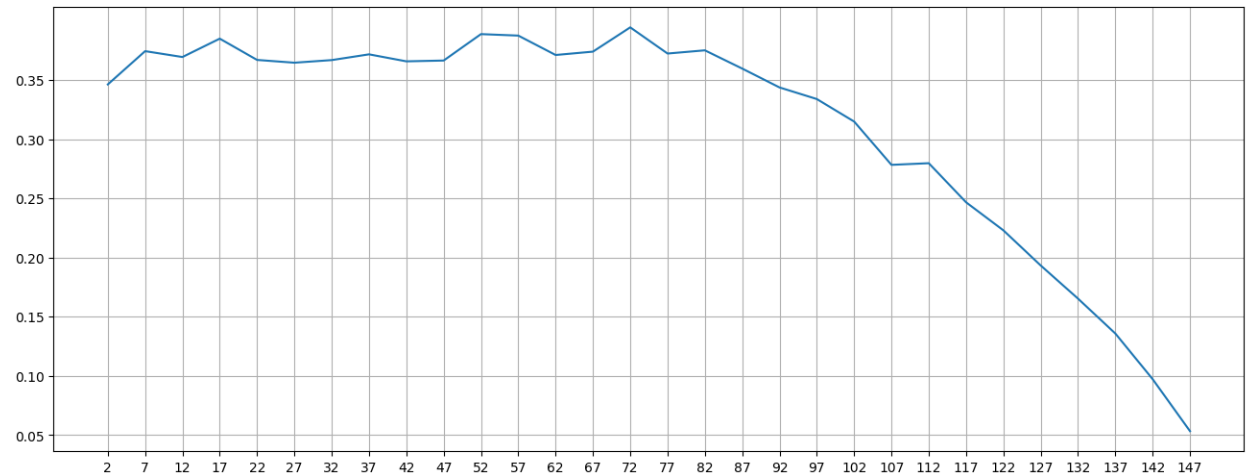
```python
from sklearn.metrics import classification_report
cr=classification_report(yval, y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.70      0.19      0.30       298
           2       0.73      0.06      0.10       199
           3       0.00      0.00      0.00        92
           4       0.00      0.00      0.00        79
           5       0.68      0.47      0.56       589
           6       0.67      0.05      0.10       194
           7       0.00      0.00      0.00        71
           8       0.75      0.54      0.63       707
           9       0.00      0.00      0.00       117
          10       0.00      0.00      0.00        13
          11       0.83      0.04      0.07       141
          12       0.00      0.00      0.00        32
          13       0.00      0.00      0.00        49
          14       0.00      0.00      0.00        94
          15       0.88      0.09      0.16       253
          16       0.92      0.14      0.24       162
          17       0.75      0.09      0.15       281
          18       0.00      0.00      0.00        64
          19       0.00      0.00      0.00        26

   micro avg       0.73      0.23      0.35      3469
   macro avg       0.35      0.08      0.12      3469
weighted avg       0.61      0.23      0.30      3469
 samples avg       0.45      0.29      0.33      3469
```

It can be seen that Bag of Words performed better than TF-IDF in all three cases.

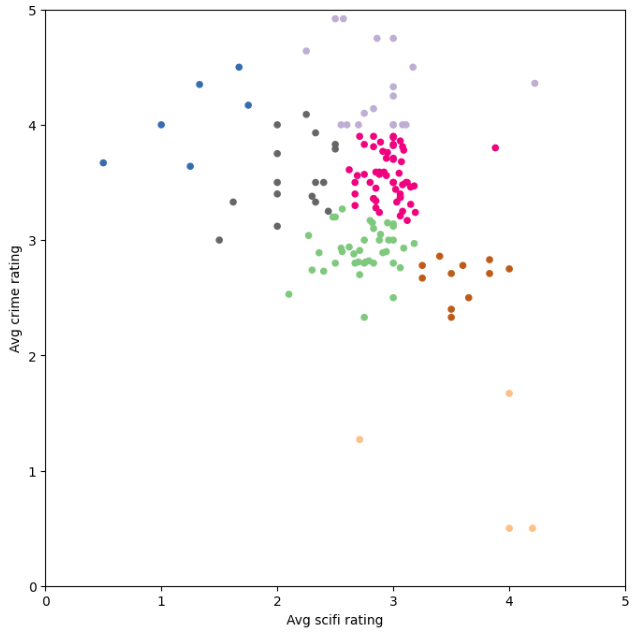
EVALUATION OF ML RESULTS:

When we performed clustering we made use of K-means clustering and by using the silhouette score we found the optimal value for the number of clusters in our dataset to be 7.

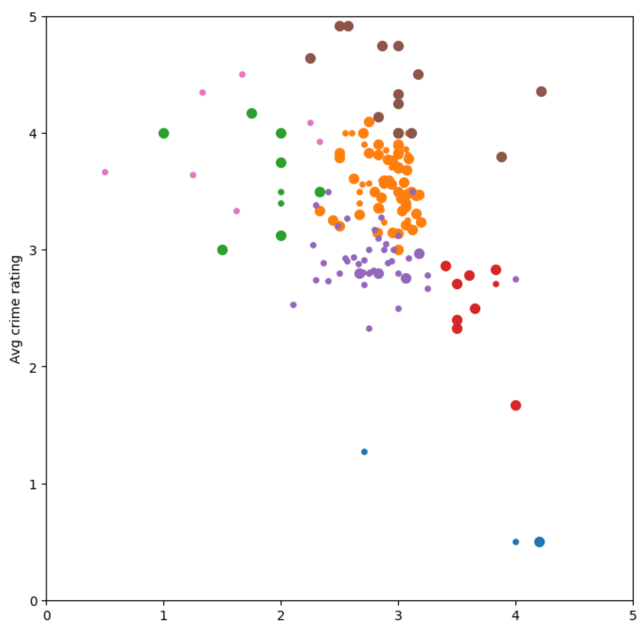


The X axis shows the number of clusters, and the y axis shows the Silhouette Scores.

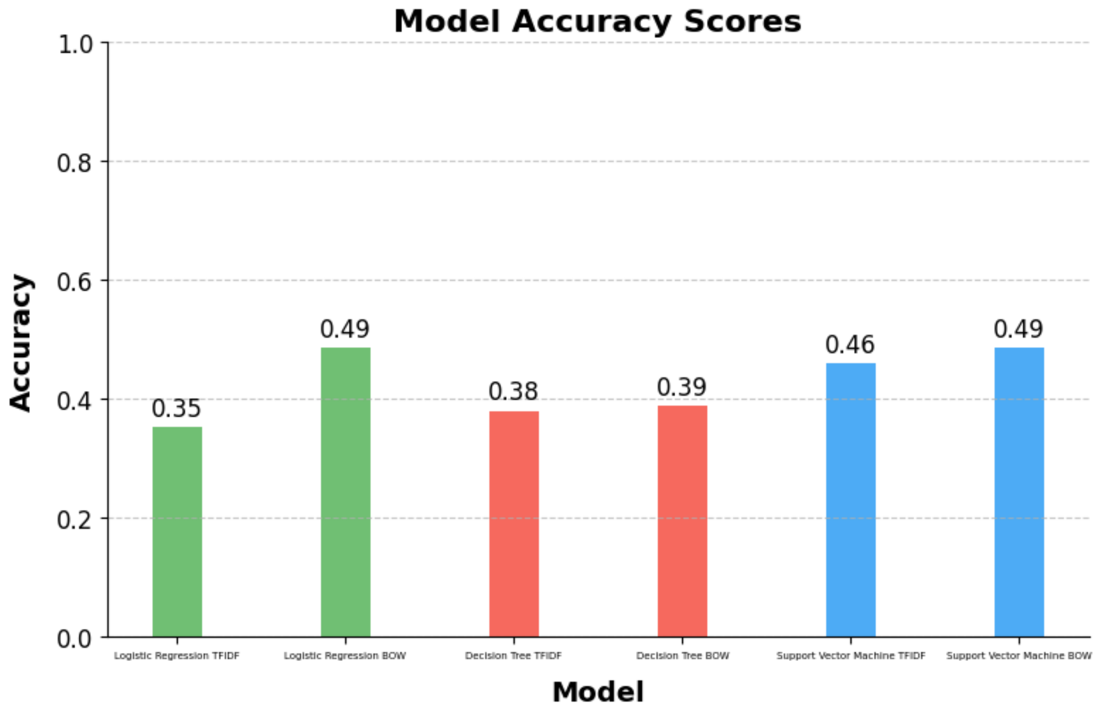
The below result shows the graphical representation of clustering for Crime movies and Sci-fi movies with the number of clusters as 7.



The graph below shows the same along with Action movies which are represented by the size of the dots.



Since we had performed 3 types of classification techniques – LOGISTIC REGRESSION, DECISION TREE CLASSIFIER AND SUPPORT VECTOR MACHINE along with Bag of words and Tf-idf, we plotted the F1 scores of all 6 models. The results are shown below.



From this, it is clearly seen that Bag of Words performed better in all three cases than TF-IDF and Support Vector Machine performed better in both TF-IDF and BOW when compared to the other two models.

In conclusion, we can say that the Support Vector Machine model along with Bag of Words is our best model out of all the six experiments.

Since Bag of Words outperformed TF-IDF when we used it for the plot, we will use Bag of Words for performing our recommendation of movies as well.

TEXT FEATURE ENGINEERING:

From our classification techniques, it is found that bag of words gave the best results. Therefore, in our content-based filtering we have used BOW to perform text extraction on the plot summaries.

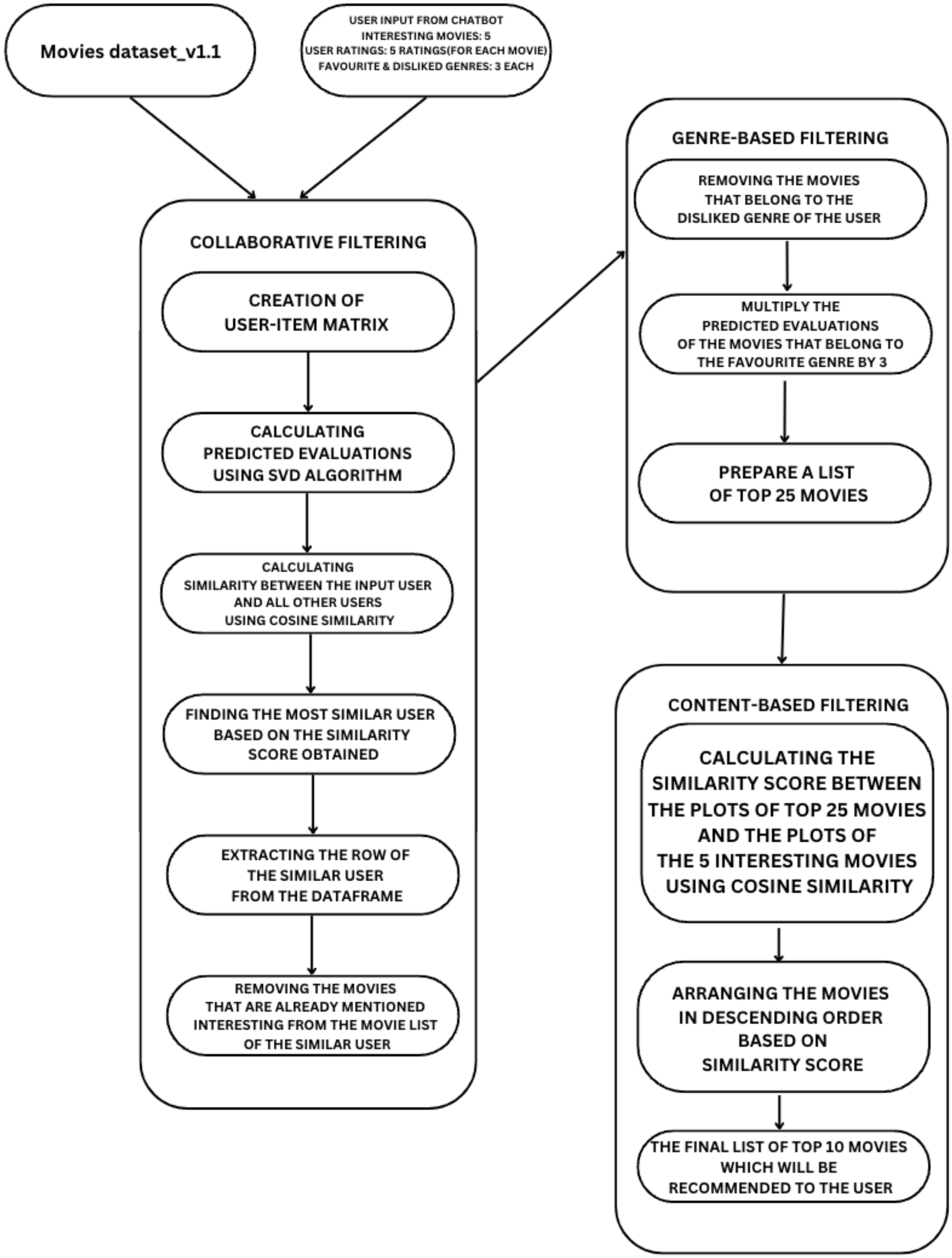
PLOTS BEFORE PERFORMING STEMMING USING BOW:

	movie_name	predicted_evaluation	genres	Plot
1	Pulp Fiction	1.161653	Comedy|Crime|Drama|Thriller	Jules Winnfield (Samuel L. Jackson) and Vincen...
2	The Matrix	1.142187	Action|Sci-Fi|Thriller	Before Neo and Morpheus became a household nam...
3	Star Wars: Episode IV - A New Hope	0.963070	Action|Adventure|Sci-Fi	The Imperial Forces, under orders from cruel D...
4	Terminator 2: Judgment Day	0.922919	Action|Sci-Fi	Over 10 years have passed since the first mach...
5	Fight Club	0.888329	Action|Crime|Drama|Thriller	A nameless first person narrator (Edward Norto...
6	Star Wars: Episode V - The Empire Strikes Back	0.855110	Action|Adventure|Sci-Fi	Luke Skywalker, Han Solo, Princess Leia and Ch...
7	Seven	0.824994	Mystery|Thriller	A veteran samurai, who has fallen on hard time...
8	Jurassic Park	0.819862	Action|Adventure|Sci-Fi|Thriller	Huge advancements in scientific technology hav...
9	Saving Private Ryan	0.808547	Action|Drama|War	Opening with the Allied invasion of Normandy o...
10	Schindler's List	0.756412	Drama|War	Oskar Schindler is a vain and greedy German bu...

PLOTS AFTER PERFORMING STEMMING USING BOW:

	movie_name	predicted_evaluation	genres	Plot
1	Pulp Fiction	1.161653	Comedy|Crime|Drama|Thriller	Jules Winnfield (Samuel L. Jackson) and Vincen...
2	The Matrix	1.142187	Action|Sci-Fi|Thriller	Before Neo and Morpheus became a household nam...
3	Star Wars: Episode IV - A New Hope	0.963070	Action|Adventure|Sci-Fi	The Imperial Forces, under orders from cruel D...
4	Terminator 2: Judgment Day	0.922919	Action|Sci-Fi	Over 10 years have passed since the first mach...
5	Fight Club	0.888329	Action|Crime|Drama|Thriller	A nameless first person narrator (Edward Norto...
6	Star Wars: Episode V - The Empire Strikes Back	0.855110	Action|Adventure|Sci-Fi	Luke Skywalker, Han Solo, Princess Leia and Ch...
7	Seven	0.824994	Mystery|Thriller	A veteran samurai, who has fallen on hard time...
8	Jurassic Park	0.819862	Action|Adventure|Sci-Fi|Thriller	Huge advancements in scientific technology hav...
9	Saving Private Ryan	0.808547	Action|Drama|War	Opening with the Allied invasion of Normandy o...
10	Schindler's List	0.756412	Drama|War	Oskar Schindler is a vain and greedy German bu...

SYSTEM ARCHITECTURE:



METHODOLOGY:

The recommender system consists of:

- Collaborative filtering
- Genre-based filtering
- Content-based filtering

COLLABORATIVE FILTERING TECHNIQUE:

The collaborative filtering is the main module of the entire recommender system. The first step in this filtering technique is forming the user-item matrix. The matrix consists of the ratings for the various movies of each user. For example, the following matrix has the ratings of the movies for the corresponding user.

The empty cells means that the user has not watched that specific film. Thus, those empty cells are added with zeros and the complete user-item matrix would look like this:

```
# Printing the input matrix
print(svd_input_matrix)
```

```
movieId  1       2       3       4       5       6       7       8       \
userId
1          4.0     0.0     4.0     0.0     0.0     4.0     0.0       0
2          0.0     0.0     0.0     0.0     0.0     0.0     0.0       0
3          0.0     0.0     0.0     0.0     0.0     0.0     0.0       0
4          0.0     0.0     0.0     0.0     0.0     0.0     0.0       0
5          4.0     0.0     0.0     0.0     0.0     0.0     0.0       0
...        ...     ...     ...     ...     ...     ...     ...     ...
606        2.5     0.0     0.0     0.0     0.0     0.0     2.5       0
607        4.0     0.0     0.0     0.0     0.0     0.0     0.0       0
608        2.5     2.0     2.0     0.0     0.0     0.0     0.0       0
609        3.0     0.0     0.0     0.0     0.0     0.0     0.0       0
610        5.0     0.0     0.0     0.0     0.0     5.0     0.0       0
```

Then, this user-item matrix is used to calculate predicted evaluations. Predicted evaluations is a term that is used to represent the output of the SVD algorithm. The user-item matrix containing the predicted evaluations looks like this:

movieId	1	2	3	4	5	6	7	\
userId								
1	2.516296	0.986196	0.822255	0.024764	0.195323	2.357841	0.320027	
2	0.144916	0.049374	-0.109209	-0.012633	-0.048288	0.033861	-0.097100	
3	0.019042	0.006661	0.029940	-0.003271	-0.010466	0.074984	-0.009047	
4	1.179820	0.033772	0.204816	0.028291	0.067272	0.751367	0.292991	
5	1.397266	0.926419	0.440084	0.089430	0.487908	0.580794	0.541642	
...	...	...	...	...	...	...	...	
607	2.238138	1.053949	0.693760	0.057515	0.336404	1.775303	0.419750	
608	4.326357	2.974011	1.560603	0.075227	1.015587	2.619117	1.081629	
609	0.964011	0.629514	0.283927	0.054603	0.295351	0.481129	0.308408	
610	2.279684	1.648709	-0.102771	-0.287299	-0.579850	2.111336	-1.204137	
611	0.126369	0.073707	0.015336	0.001702	0.013917	0.084287	0.003479	

Now, the user is compared with the other users who exhibit the same behaviour with respect to ratings and one most similar user is identified and the entire user information(row) is extracted. This is similarity is calculated using cosine similarity.

As per our hypothesis 1, we should not recommend any movie that is already mentioned by the user as interesting. So, to satisfy this hypothesis these interesting movies are removed from the list of movies of the similar user.

GENRE-BASED FILTERING:

After performing collaborative filtering, genre-based filtering is done. The second hypothesis of our recommender system is to not recommend any movie that belong to the disliked genre of the user. Thus, all the movies which come under the disliked genre category is removed.

Also, in the beginning we obtained the favourite genres of the user which is helpful in this filtering technique. From the list of movies, the movies which belong to the favourite genre are taken. Now, the predicted evaluation score is increased so that these movies are pushed to the top as they are very relevant to the user. To increase the predicted evaluation value, it is multiplied by 3.

Favourite_Genre_Predicted_Evaluation = Predicted_Evaluation * 3

Finally, a list of top 25 movies is prepared by arranging the final list in descending order.

Before Genre-based filtering:-

	movie_name
0	Forrest Gump
1	Pulp Fiction
2	The Matrix
3	Star Wars: Episode IV - A New Hope
4	Terminator 2: Judgment Day
5	Fight Club
6	Star Wars: Episode V - The Empire Strikes Back
7	Seven
8	Jurassic Park
9	Saving Private Ryan
10	Schindler's List
11	Raiders of the Lost Ark
12	The Godfather
13	The Dark Knight
14	Gladiator
15	Inception
16	Toy Story
17	The Fugitive
18	Apollo 13
19	American Beauty
20	The Sixth Sense
21	Independence Day
22	Back to the Future
23	Twelve Monkeys
24	Fargo

After Genre based filtering:

	movie_name	predicted_evaluation \
1	Pulp Fiction	1.161653
2	The Matrix	1.142187
3	Star Wars: Episode IV - A New Hope	0.963070
4	Terminator 2: Judgment Day	0.922919
5	Fight Club	0.888329
6	Star Wars: Episode V - The Empire Strikes Back	0.855110
7	Seven	0.824994
8	Jurassic Park	0.819862
9	Saving Private Ryan	0.808547
10	Schindler's List	0.756412
11	Raiders of the Lost Ark	0.747043
12	The Godfather	0.711990
13	The Dark Knight	0.704223
14	Gladiator	0.680839
15	Inception	0.671633
17	The Fugitive	0.637388
18	Apollo 13	0.634150
20	The Sixth Sense	0.629467
21	Independence Day	0.626634
22	Back to the Future	0.585571
23	Twelve Monkeys	0.583721
24	Fargo	0.581521
25	Indiana Jones and the Last Crusade	0.573532
26	Memento	0.552086
29	Léon: The Professional	0.531207

CONTENT-BASED FILTERING:

Finally, to enhance our recommender system, we have also performed content-based filtering. For this we have used the plot summaries of the movies. A similarity score is calculated for the final list of 25 movies based on the similarity between their plots and the plots of the interesting movies (input from user) that was initially obtained from the user, using cosine similarity. The movies are again arranged in descending order of the similarity scores and recommended to the user in this order.

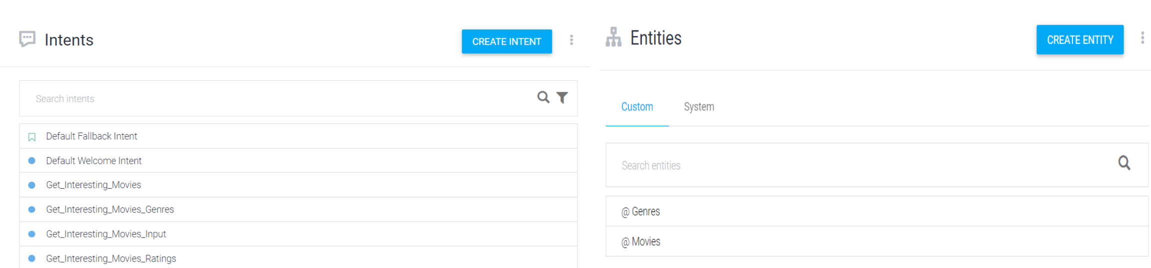
FINAL LIST:

```
                                       movie_name  Similarity  \
3            Star Wars: Episode IV - A New Hope    0.136522
6    Star Wars: Episode V - The Empire Strikes Back    0.124883
1                                   Pulp Fiction    0.110158
15                                     Inception    0.107001
4                     Terminator 2: Judgment Day    0.098347
5                                     Fight Club    0.097901
24                                         Fargo    0.085310
14                                     Gladiator    0.083727
8                                  Jurassic Park    0.082763
9                            Saving Private Ryan    0.077101

    predicted_evaluation                                         genres
3               0.963070                      Action|Adventure|Sci-Fi
6               0.855110                      Action|Adventure|Sci-Fi
1               1.161653                  Comedy|Crime|Drama|Thriller
15              0.671633  Action|Crime|Drama|Mystery|Sci-Fi|Thriller|IMAX
4               0.922919                                Action|Sci-Fi
5               0.888329                  Action|Crime|Drama|Thriller
24              0.581521                  Comedy|Crime|Drama|Thriller
14              0.680839                                 Action|Drama
8               0.819862             Action|Adventure|Sci-Fi|Thriller
9               0.808547                             Action|Drama|War
```

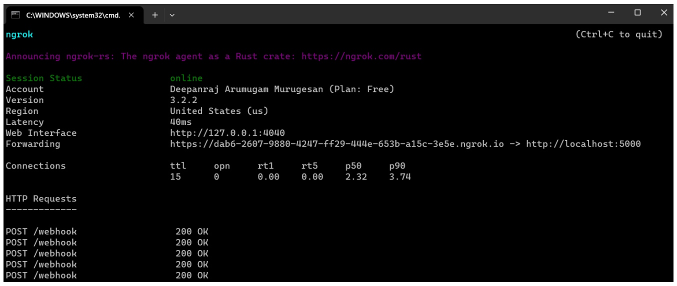
RESULTS OF THE DESIGNED SYSTEM/MODEL:

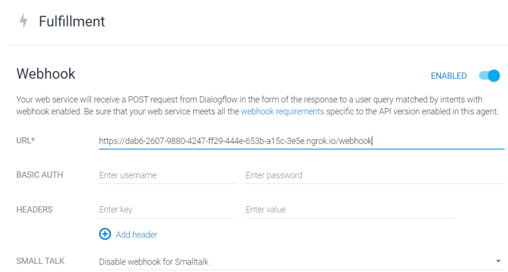
The user interface is built using Dialogflow as a chatbot and is also integrated with telegram.

For the chatbot integration, we have trained our chatbot by creating 4 intents. Also, we have two entities(movies, genres).

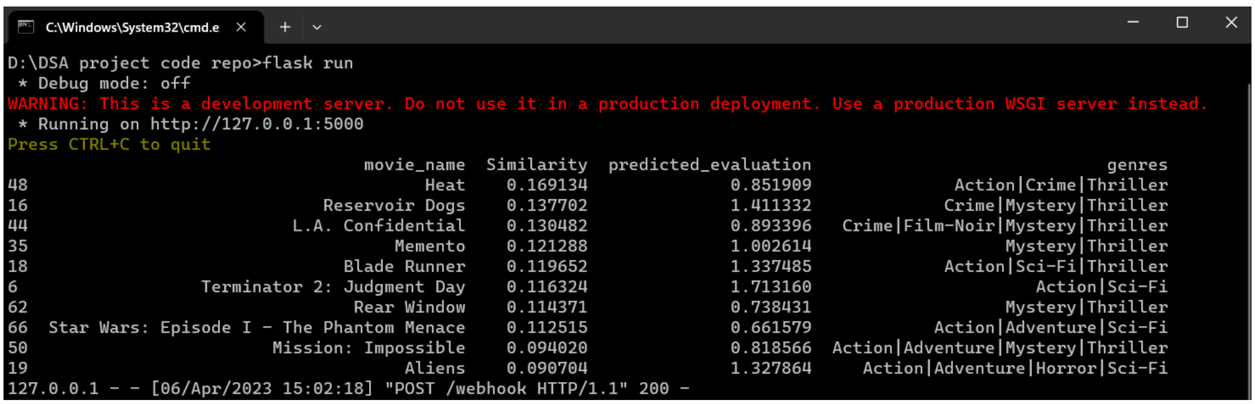


We have used ngrok to obtain the https url as dialogflow accepts only secured links and we have used this in our fulfilment section.





As the final step we now run the flask server to interact with our chatbot.



This is an example showing the interactions with the chatbot that is integrated on Telegram.

ERROR ANALYSIS:

For the designed model, we had to come up with a way to find the most similar user from the SVD output matrix, and this was possible once we applied the cosine similarity algorithm between the newly added user row in the matrix and all the other users.

Also, we have an input constraint where the user should give at least one movie that's present in our dataset to give out recommendations; otherwise, the model will give out unrelated movie recommendations. We have fine-tuned our model to accept user input, wherein at least one movie input is enough for a recommendation, but of course, the performance won't be the same as considering the input for five movies. The major drawback in our model would be the small dataset we have, as we only have 7840 unique movies, which weakens the model's performance. Due to our own computation constraints, we had to stick with the smallest dataset from MovieLens, as it took around 45 minutes for us to complete web scraping of plot summaries in execution itself with the smallest dataset.

INNOVATIVENESS:

Most of the movie recommender systems used ratings or semantic analysis of movie reviews to recommend movies to the users. Also, the systems mostly used only collaborative or content based filtering techniques. But our system uses ratings, genres and plots to give the best and most comprehensive recommendations to the users.

REFERENCES:

- Walek, Bogdan, and Vladimir Fojtik. "A Hybrid Recommender System for Recommending Relevant Movies Using an Expert System." *Expert Systems with Applications*, vol. 158, Nov. 2020, p. 113452, https://doi.org/10.1016/j.eswa.2020.113452. Accessed 28 June 2020.
- Falk, K. (2019). *Practical recommender systems* (1st ed.). Shelter Island: Manning ISBN 1-61729-270-2.
- Barragáns-Martínez, B., Costa-Montenegro, E., & Juncal-Martínez, J. (2015). Developing a recommender system in a consumer electronic device. *Expert Systems with Applications, 42*(9), 4216–4228.
- Aggarwal, C. C. (2016). *Recommender systems*: 1. Cham: Springer International PubLishing.