CSCE 5222 : FEATURE ENGINEERING

Project Increment 1

ProjectTitle : Movie Recommendation System

Team Members:

1.Reshmi Chowdary Divi –11520230
2.Phani Sagar Kothapalli – 11555373
3.Harshith Reddy Nagireddy - 11517457

Github link : https://github.com/reshmi611/Movie-Recommendation-system

Introduction :

As a component of data-shifting frameworks, recommender systems are used to predict the preferences or evaluations a customer will generally have for a product. Due to their effectiveness, shared shifting procedures are one of the most common sorts of proposal draws near. These traditional frameworks for collaborative sifting can even be useful and provide typical solutions for pervasive problems. For anything that was influenced by their neighbor's preferences, Collaboration-based separation techniques favor some proposals over others. Although various approaches, such as those that are substance-based, encounter a lack of accuracy, adaptability, information abundance, and massive error forecast. We used a thing-based synergistic sifting approach to find these prospects. In this method for item-based collective separation, we first examine the user thing rating framework and we identify relationships between various factors, and then we use these relationships to develop suggestions for the client. We identify relationships between various factors, and then we use these relationships to develop suggestions for the client.

Everyone enjoys movies, regardless of age, sex, ethnicity, skin tone, or geographic location. Through this amazing medium, we are sort of connected to one another as a whole. However, what I find most fascinating is how unique our selections and combinations are in terms of our film preferences. Some groups prefer kind explicit movies, whether they are spine-tinglers, sentimental dramas, or science fiction, while others are more interested in the main performers and bosses. When all is said and done, it becomes incredibly difficult to, sum up a movie and declare that everybody could enjoy it. Having said that, it is nevertheless evident that a certain segment of the general population favors comparative films.

Personal recommendations are given via recommender systems based on user profiles and past activity. The Internet industry makes extensive use of recommender systems. Users of recommender systems frequently utilize websites like Amazon, Netflix, and YouTube. Users can find their favorite products with the aid of recommender systems, which can also assist online service providers make money.
The main goal of the recommendation system, a filtering program, is to predict a customer's "rating" or "inclination" toward a specific item or item. For our instance, the specific item in the space is a movie, hence the main goal of our proposal framework is to identify and predict only those movies that a customer would like given some information about the client themselves.

Goals and Objectives :

If KNN uses Euclidean distance in its intended work, its exhibition will suffer from the negative impacts of revile of dimensionality. Since all vectors are essentially equidistant from the inquiry question vector at high measurements (focus on the highlights of the film), Euclidean distance is useless. In light of everything, we shall use cosine proximity for nearest neighbor search.

The main goal of the recommendation system, a filtering program, is to predict a customer's "rating" or "inclination" toward a specific item or item in a given area. For our instance, the specific item in the space is a movie, hence the main goal of our proposal framework is to identify and predict only those movies that a customer would like given some information about the client themselves.

Motivation:

A kind of recommendation engine called collaborative filtering makes use of both user and item data. more specifically, user ratings on specific things submitted by specific individuals. In this method, products are collaboratively recommended based on user ratings. Users and things are the two axes of a utility matrix that can be used to represent this data. Since not every user has rated every item, collaborative filtering recommendation engines fill in the gaps in a utility matrix by outputting the highest-ranked, previously unrated things as suggestions.

Significance :

Normalizing the dataset is a considerable improvement in information preprocessing. This cycle changes the mean of the several information highlights to zero and also changes their difference to one. As a result of the different sizes of all information highlights, this ensures that there is no bias while building the model. If this isn't done, the neural organization may become confused and place a greater stress on the features that have a larger average incentive than others.
By including the Standard Scaler method from the sklearn preprocessing library, we actualize this evolution. With the Standard Scaler work, we start the variable sc. Then, using the X train and X test datasets, we implement these progressions using the fit transform approach.

Features :

Latent characteristics are simply an abstraction of all the features of the objects or users. You can multiply the matrices together to create a matrix as long as users and things both have the same number of latent attributes.

similar to your utility matrix's dimensions. You can fine-tune your model's hyperparameters by changing the amount of latent features. We can also observe, based on matrix multiplication, that the I3 row of the item matrix and the U1 column of the user matrix have an impact on the rating value of U1 for I3.
We must apply a different strategy because we can't decompose matrices with missing values. Machine learning can help with this. As noted earlier, We are now going to try to construct the utility matrix using our item matrix and our user matrix, as was previously described. Alternating Least Square is a gradient descent technique used in this.

Background:

Recommender systems are built on a variety of strategies, such as content-based collaborative procedures and hybrid approaches. Additionally, most collaborative filtering- and clustering-based movie recommendation systems. The user of movie recommender systems is prompted to rate the movies they have already seen. The user is then presented with recommendations for more movies based on these ratings and collaborative filtering, which is based on related ratings. Collaborative filtering has been widely embraced, and most recommender systems currently employ it.
Data sparsity and computing complexity are limitations of this strategy. The item-based collaborative filtering technique was one of many that attempted to address the issues with computational complexity and memory bottlenecks by computing relations between items for the neighborhood region surrounding

a target object. They showed that the item-based strategy can produce rationally accurate predictions while also taking computation time into account.

Dataset Description:

A set of movie ratings for this dataset were provided by the movie recommendation website Movie Lens. This dataset was created and maintained by the Group Lens research team at the University of Minnesota. "25m", "latest-small," "100k," "1," and "20m" are the five variations. Each dataset's movie data and ratings data are joined together via the movie Id. The 25m dataset, latest-small dataset, and 20m dataset only contain movie and rating data. The 1 million and 100,000 datasets include demographic data in addition to movie and rating data."25m": The Movie Lens dataset is currently at this stable version. It is advised for use in research.

"latest-small" refers to the most recent Movie Lens dataset's small subset. Group Lens alters and updates it over time.
The Movie Lens datasets "100k" version is the most recent. The dataset only contains a few demographic variables.
The largest Movie Lens dataset with demographic information is "1m."

Along with the 1m dataset, "20m" is one of the Movie Lens datasets that is most frequently used in academic works.
Config description: This dataset contains data of 62,423 movies rated in the 25m dataset.

Features :

All versions ending in "-ratings" come standard with the features listed below.

"movie id": a specific reference to the rated film

"movie title" is the name of the rated film, along with the year of its release, in parentheses.A list of the several genres to which the rated film belongs is called "movie genres."
"user id": a specific identification number for the user who submitted the ratingUser ratings are expressed as a rating out of five stars.
"Timestamp" refers to the ratings' timestamp, which is expressed in seconds since midnight. January 1, 1970, Coordinated Universal Time (UTC).

Analysis:
This filtration method depends on examining the client's behavior in relation to the behavior of other clients in the data collection. This estimate heavily relies on the historical context of each client. The key distinction between content-based separating and collective sifting is that while in the latter, the communication of all customers with the items affects the proposal calculation, only the information of the concerned client is taken into account in content-based separating.
There are several ways to carry out collective separating, but the key concept to grasp is that, in communitarian separating, diverse clients' information influences the outcome of the recommendation. additionally, doesn't base its arguments solely on the data of a single client.

Implementation:

The working rule is quite simple. We first check to see if the movie name entered is already in the database, and if it is, we use our suggestion system to find related movies, arrange them based on how similar they are, and then only produce the top 10 movies with strong points from the database film.
A type of suggestion framework known as "thing based cooperative sifting" hinges on the similarity of things as defined by the ratings that customers have given to objects. It tackles problems that client-based shared channels go into, such as when the framework includes many items but few things are valued. In addition, IBCF has a lower computational escalation than UBCF.

We channeled information using an item-based cooperative filtering technique. A table or network that illustrates how comparable one thing is to another will be produced by applying the modified cosine likeness criterion to evaluations of various things. To generate a suggestion, we should combine our grid of similitude with the prior evaluations made by clients. Applying IBCF's condition makes this simple.

By going over each customer and item again, this is the perfect time to determine the suggestion score for each client. In each cycle, we will first determine the client's average rating, which will be used to calculate the score. Then, we will determine whether the item has been evaluated by the client or not; if so, we will store a -1 because we only want to recommend items that the client has not loved or evaluated. We use the similitude network we previously established to grab the top 10 items that are similar to the provided thing, assuming it hasn't been analyzed. The client's appraisal for those comparable items is next presented. We will also determine the typical rating of comparable items in order to subtract it from the items' appraisal.

Finally, we will submit a list of comparable items, a list of comparable item client ratings, and a normal rating of those items to our "Cal Score" score job. The result will be included in the client's overall rating. This will result in scores that will show which recommendations to make for which clients.

Preliminary Results:

Examining the disorder network is a much better method to judge a classifier's performance. The general goal is to count the instances in which examples of class A are referred to as class B. Based on the results of the matrix, we will forecast the model's accuracy.
Project Management:

Work Completed :

In this project we involved content based filtering and Collaborative filtering.

There are several ways to carry out collective separating, but the key concept to grasp is that, in communitarian separating, diverse clients' information influences the outcome of the recommendation. additionally, doesn't base its arguments solely on the data of a single client.
For instance, if client 'A' like 'Batman Begins, 'Equity League,' and 'The Avengers,' while client 'B' enjoys 'Batman Begins, 'Equity League,' and 'Thor,' then they have comparative interests because we are aware that these films are appropriate for audiences who identify as superhuman. Accordingly, there is a good chance that client "A" would favor "Thor," while client "B" could prefer "The Avengers."
Computing missing values

We quickly transition to discussing similarity measurements. Common examples include: cosine similarity, Pearson correlation, Jaccard index, and Euclidean distance (a particular type of Minkowski distance). Pearson correlation has been shown to function best in experiments. This article's examples will make use of cosine similarity.

Responsibility :

Analyzing algorithms – Divi Reshmi – 40%

Implementation – Kothapalli Phani Sagar – 30%

Documentation and code analyzing – Nagireddy Harshith Reddy – 30%

Work to be Completed:

We need to implement some more filtration techniques other content-based filtering and collaborative filtering.
Need to improve the accuracy and In addition to accuracy, a variety of other parameters must be measured to make sure the algorithm generates correct predictions. The techniques may, for example, be quite precise but have too much logarithmic loss. Not all metrics are used to evaluate a model's performance effectiveness, including accuracy.

Responsibility :

Issues and concerns recovering – Divi Reshmi Chowdary – 30%

Implementing other algorithms - Kothapalli PhaniSagar – 40%

Improving efficiency of code – Nagireddy Harshith Reddy – 30%

Issues and conerns: We are involving to perform more algorithms to improve better accuracy

```
import seaborn as sns
```

```
movies_df = pd.read_csv('movies.csv')
ratings_df = pd.read_csv('ratings.csv')
```

```
movies_df.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

```
[ ] ratings_df.head()
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

```
[ ] def get_director(x):
        for i in x:
            if i["job"] == "Director":
                return i["name"]
        return np.nan
```

```
        return names
    return []
```

```
final_dataset = ratings.pivot(index='movieId',columns='userId',values='rating')
final_dataset.head()
```

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
movieId																					
1	4.0	NaN	NaN	NaN	4.0	NaN	4.5	NaN	NaN	NaN	...	4.0	NaN	4.0	3.0	4.0	2.5	4.0	2.5	3.0	5.0
2	NaN	NaN	NaN	NaN	NaN	4.0	NaN	4.0	NaN	NaN	...	NaN	4.0	NaN	5.0	3.5	NaN	NaN	2.0	NaN	NaN
3	4.0	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 610 columns

```
final_dataset.head()
```

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
movieId																					
1	4.0	0.0	0.0	0.0	4.0	0.0	4.5	0.0	0.0	0.0	...	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	3.0	5.0
2	0.0	0.0	0.0	0.0	0.0	4.0	0.0	4.0	0.0	0.0	...	0.0	4.0	0.0	5.0	3.5	0.0	0.0	2.0	0.0	0.0
3	4.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0
4	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	5.0	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

5 rows × 610 columns

```
[ ] no_user_voted = ratings.groupby('movieId')['rating'].agg('count')
    no_movies_voted = ratings.groupby('userId')['rating'].agg('count')
```

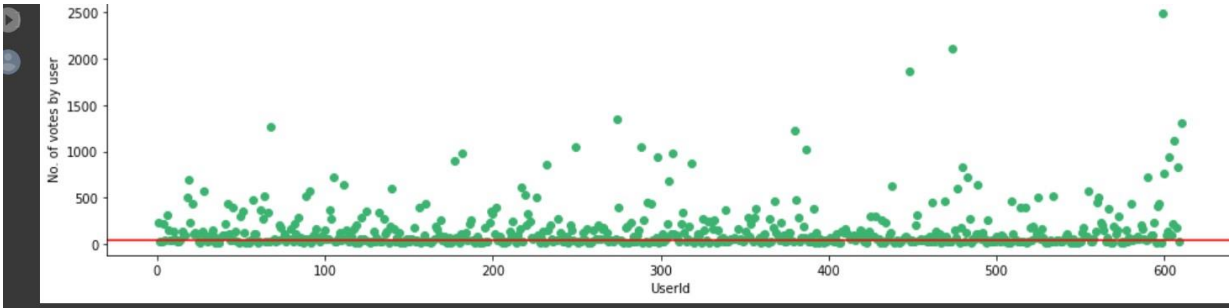
```
[ ]  sample = np.array([[0,0,3,0,0],[4,0,0,0,2],[0,0,0,0,1]])
     sparsity = 1.0 - ( np.count_nonzero(sample) / float(sample.size) )
     print(sparsity)
```

```
0.7333333333333334
```

```
[ ]  csr_sample = csr_matrix(sample)
     print(csr_sample)
```

```
  (0, 2)        3
  (1, 0)        4
  (1, 4)        2
  (2, 4)        1
```

```
[ ]  f,ax = plt.subplots(1,1,figsize=(16,4))
     # ratings['rating'].plot(kind='hist') plt.scatter(no_user_voted.index,no_user_voted,color='r
     plt.xlabel('MovieId')
     plt.ylabel('No. of users voted')
     plt.show()
```




```
]  final_dataset=final_dataset.loc[:,no_movies_voted[no_movies_voted>50].index]
   final_dataset

   sample =np.array([[0,0,3,0,0],[4,0,0,0,2],[0,0,0,0,1]])
   sparsity = 1.0 - ( np.count_nonzero(sample) / float(sample.size) )
   print(sparsity*100)

   csr_sample = csr_matrix(sample)
   print(csr_sample)
```

```
csr_sample = csr_matrix(sample)
print(csr_sample)

csr_data = csr_matrix(final_dataset.values)
final_dataset.reset_index(inplace=True)

knn=NearestNeighbors(metric='cosine',algorithm='brute',n_neighbors=20,n_jobs=-1)
knn.fit(csr_data)
NearestNeighbors(algorithm='brute', leaf_size=30, metric='cosine', metric_params=None, n_jobs=-1, n_
```

```
73.33333333333334
  (0, 2)        3
  (1, 0)        4
  (1, 4)        2
  (2, 4)        1
NearestNeighbors(algorithm='brute', metric='cosine', n_jobs=-1, n_neighbors=20)
```

```
def get_movie_recommendation(movie_name):
  n_movies_to_reccomend = 10
  movie_list = movies[movies['title'].str.contains(movie_name)]
```

References:

Ananya Agarwal, S. Srinivasan "Movie Recommendation System" (2020)

F. Furtado, A. singh, "Movie Recommendation System Using Machine Learning" (2019)

Nirav Raval, Vijayshri Khedar, "Collaborative Filtering System Based Movie Recommendation System" (2018)