ROUND – 2
ENVISION DOCUMENTATION

Problem statement: Movie Rating Prediction

Description: Using a dataset of user ratings, genres, and movie details, predict the rating a user might give to a movie they haven't watched yet. Implement collaborative or content-based filtering.

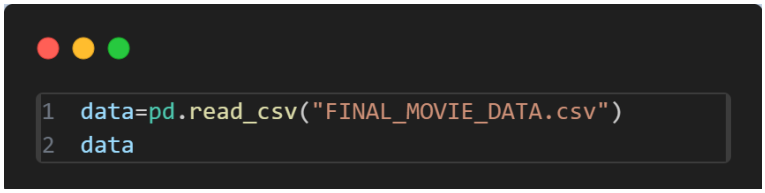
Dataset: The dataset that we have chose is taken from the website https://www.imdb.com/ which is a renowned website that is used for taking ratings and reviews from the audience. This website helped us to make a raw dataset of 2045 rows x 10 columns.

These 10 columns included variables like

1) Movie title 2)URL of that movie 3)Title type 4) IMDb ratings 5) Runtime (mins)
6) Year of release 7)Genres 8)No. of votes 9) Release dates 10) Name of directors

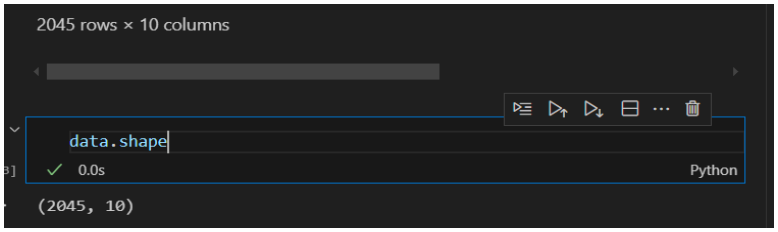
Problem faced in raw dataset: The dataset contained of issues like **Null values**, **Duplicate values**, **Unscaled values** which has to be solved to create an accurate machine learning model.

Solution for above problem: To solve the problem we made use of **python** programming language. As our raw data contained 2045 rows x 10 columns



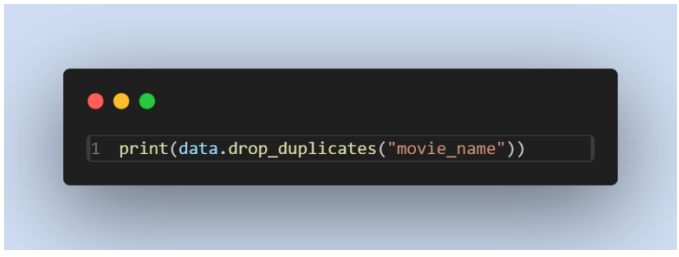
This LOC helped us to import the raw data file so that we can perform operation on it.

.



Shape of our raw dataset

Code that we used to remove Null values, Duplicate data, and Unscaled values are as follow:

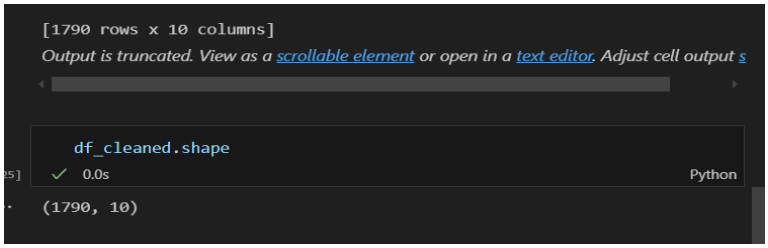


We used this to find the duplicates in our raw data.

We first calculate the mean of Runtime and substituted the null values with the mean value.

```
df_cleaned = data.drop_duplicates(subset=[
'movie_name'])
print(df_cleaned)
```

Deleted all duplicates and cleaned the data .



The shape of data of cleaned data is 1790 rows x 10 columns.

DETAIL SOLUTION PLAN OF ML MODEL

Step 1: Problem Understanding and Data Exploration

1. Understand the Problem:

The task here involves predicting the ratings a user provides for a movie, to which the person hasn't as yet rated, basing its estimation on past or historical data. This involves, in a nutshell, recommending movies such that the predicated ratings obtained are as close as possible toward the real or actual rating the user himself would have bestowed.

2. Data Collection Overview:

The dataset will more often contain only user ratings or movie details (like title, genre, release year), and user information (demographics, preferences). Common sources for this data include movie rating databases such as Movie Lens or IMDB.

no Key variables:

 User data: User ID, demographics, location.

 Movie data: Movie ID, title, genre, director, release year.

 : User ID, Movie ID, Rating (typically on a scale of 1-5).

3. Data Preprocessing:

no Handle missing values in the ratings matrix.

no Encode categorical variables like genres (using one-hot

encoding or embedding layers if using deep learning).

Normalize ratings if necessary.

Split data into training and test datasets for evaluation

purposes.

Step 2: Content-Based Filtering Approach

1. Movie Features:

Use the metadata of the movies (genres, directors, cast,

etc.) to build a feature matrix. This can be done using

one-hot encoding for categorical features (genres,

director) or embedding methods.

2. User Preferences:

Build a user profile based on the movies they have rated

very. This may be achieved by computing the average ratings

of genres they like, or by using TF-IDF (Term FrequencyInverse Document Frequency) to represent movies

based on keywords, plot summaries, etc.

3. Recommendation Engine:

o For a given movie, compute the similarity between that

movie and other movies based on the user's preferences

and movie features. Predict the ratings by looking at the

similarity between movies.

Step 3: Evaluation Metrics

1. Evaluation:

o Root Mean Squared Error (RMSE): Measures the

difference between predicted and actual ratings.

o Mean Absolute Error (MAE): The average of the

absolute errors between predicted and actual ratings.

o Precision and Recall: Useful especially if we want to

recommend top-k movies.

2. Cross-Validation:

o Use k-fold cross-validation to evaluate the performance of the

model and to prevent overfitting.

Step 4: Model Optimization

1. Hyperparameter Tuning:

o For collaborative filtering, tune parameters such as the number of neighbors or latent factors (in case of matrix factorization).

o Hyperparameter tuning for content-based filtering related to feature extraction, for example, the number of features in the TF-IDF or embedding layers.

2. Regularization:

o Implement regularization techniques to avoid overfitting especially for matrix factorization methods.

Step 5: Deployment

1. Recommendation System Deployment:

o Develop an API or a web interface that accepts a user's ID as input and returns a list of recommended movies along with predicted ratings.

o Update the model continuously with new user ratings and movie data to improve the prediction over time.

2. Real-Time Predictions:

o Implement real-time prediction functionality so that the system can suggest movies as soon as a user interacts with it.

Tools and Technologies

Programming Languages: Python (as preferred for its large ecosystem in data science).

Libraries/Frameworks:

o Collaborative Filtering: surprise, implicit.

o Content-Based Filtering: scikit-learn (for vectorization, similarity)

o Hybrid Models: TensorFlow, PyTorch (for neural networks), or traditional ML libraries like XGBoost.

o Evaluation: scikit-learn, pandas, numpy.