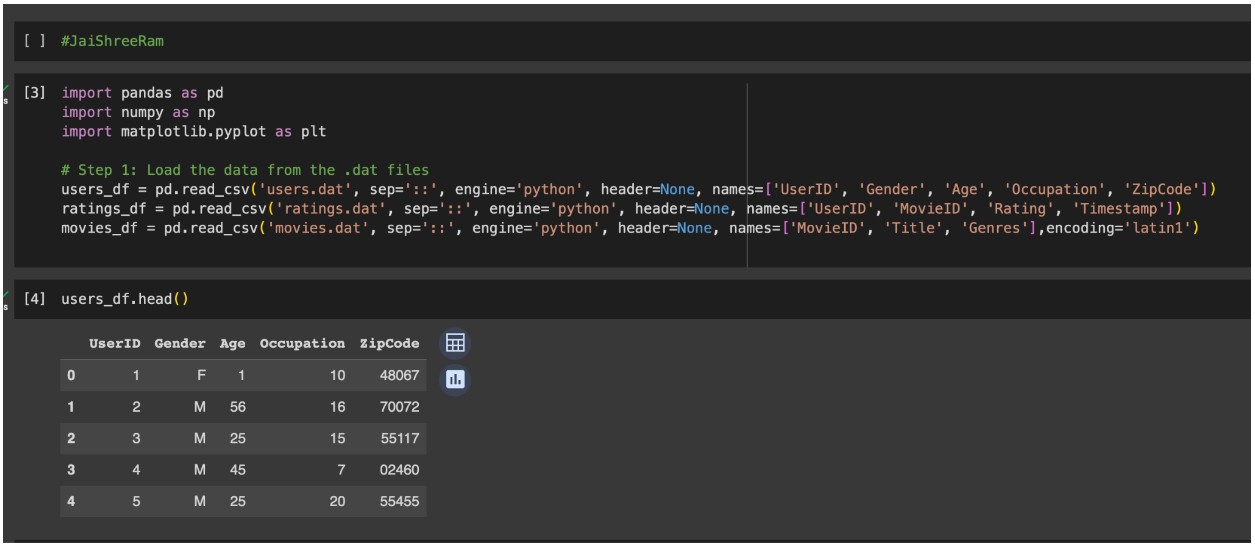
Movie Rating Prediction

1. Load the data from .dat files.

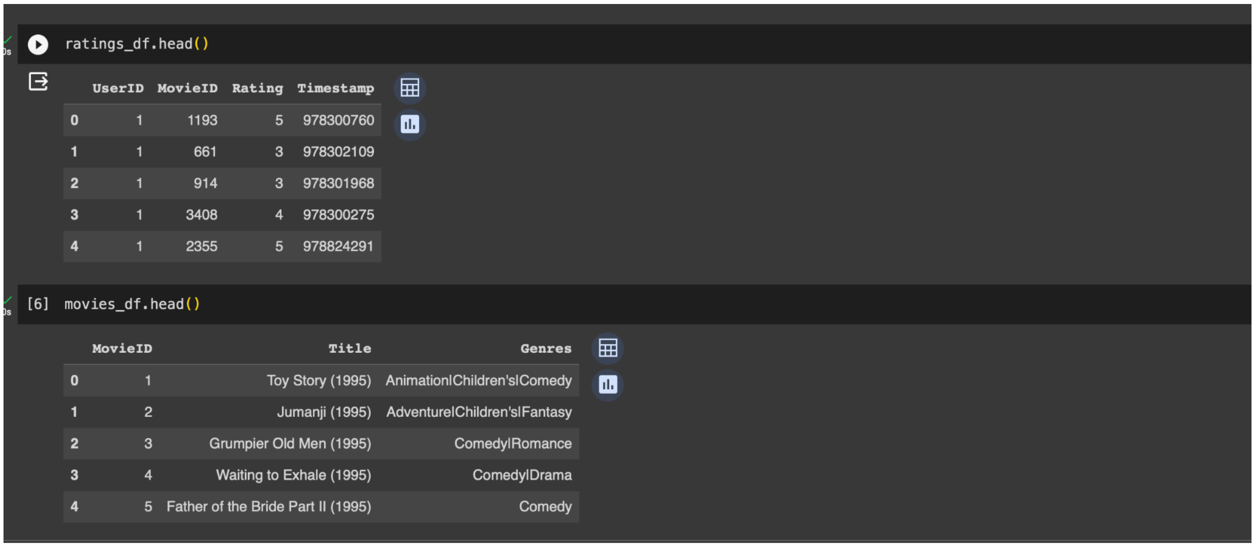


```
[ ]  #JaiShreeRam
```

```
[3]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     # Step 1: Load the data from the .dat files
     users_df = pd.read_csv('users.dat', sep='::', engine='python', header=None, names=['UserID', 'Gender', 'Age', 'Occupation', 'ZipCode'])
     ratings_df = pd.read_csv('ratings.dat', sep='::', engine='python', header=None, names=['UserID', 'MovieID', 'Rating', 'Timestamp'])
     movies_df = pd.read_csv('movies.dat', sep='::', engine='python', header=None, names=['MovieID', 'Title', 'Genres'],encoding='latin1')
```

```
[4]  users_df.head()
```

```
   UserID Gender  Age  Occupation  ZipCode
0       1      F    1          10    48067
1       2      M   56          16    70072
2       3      M   25          15    55117
3       4      M   45           7    02460
4       5      M   25          20    55455
```



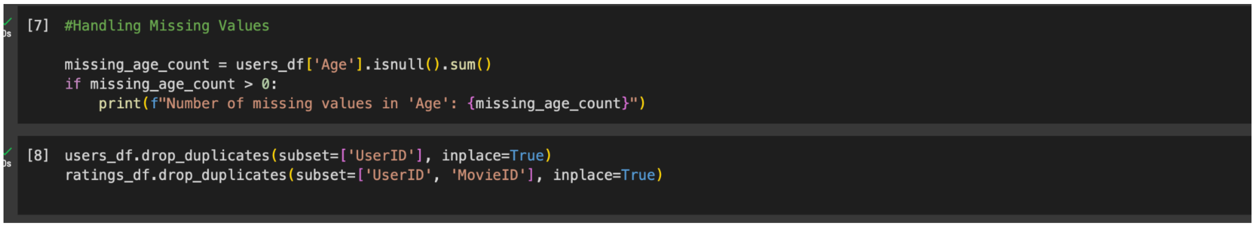
```
ratings_df.head()
```

```
   UserID  MovieID  Rating  Timestamp
0       1     1193       5  978300760
1       1      661       3  978302109
2       1      914       3  978301968
3       1     3408       4  978300275
4       1     2355       5  978824291
```

```
[6]  movies_df.head()
```

```
   MovieID                               Title                        Genres
0        1                    Toy Story (1995)   Animation|Children's|Comedy
1        2                      Jumanji (1995)  Adventure|Children's|Fantasy
2        3             Grumpier Old Men (1995)                Comedy|Romance
3        4            Waiting to Exhale (1995)                  Comedy|Drama
4        5  Father of the Bride Part II (1995)                        Comedy
```

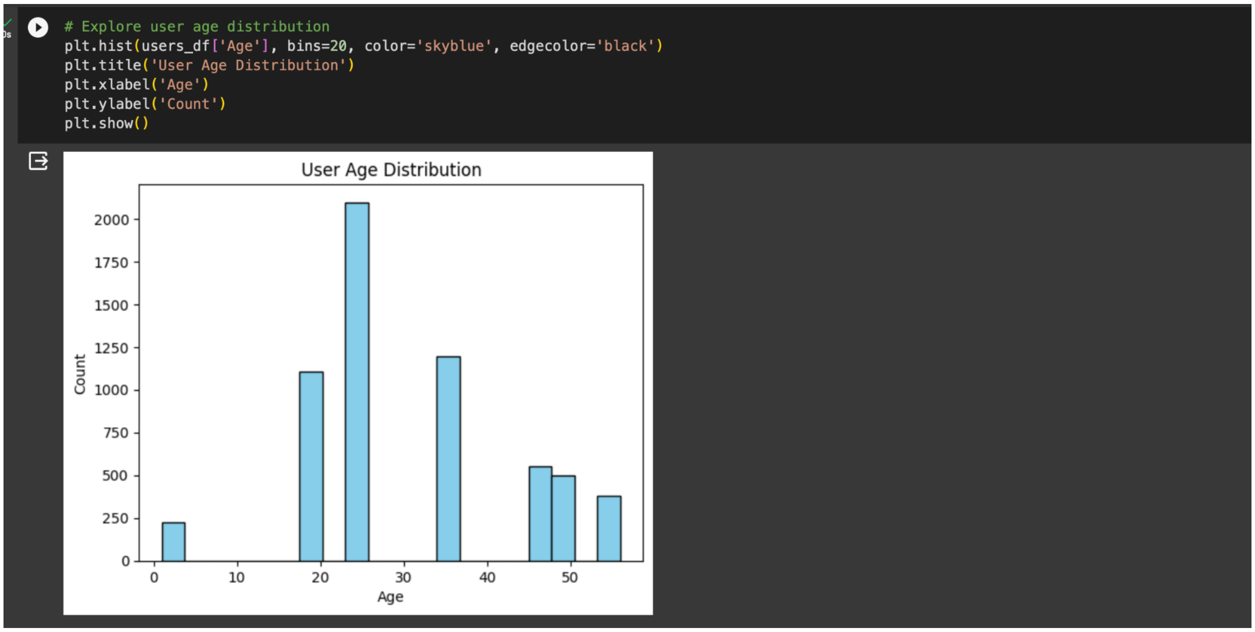
2. Data cleaning



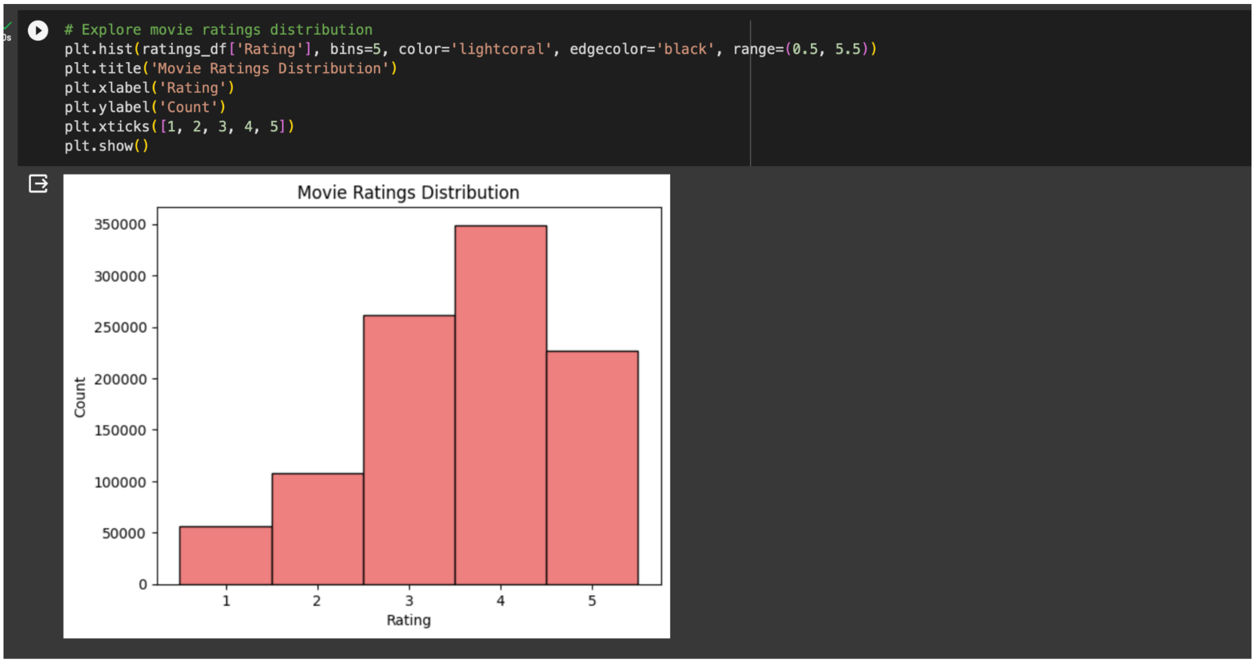
```
[7]  #Handling Missing Values

     missing_age_count = users_df['Age'].isnull().sum()
     if missing_age_count > 0:
         print(f"Number of missing values in 'Age': {missing_age_count}")
```

```
[8]  users_df.drop_duplicates(subset=['UserID'], inplace=True)
     ratings_df.drop_duplicates(subset=['UserID', 'MovieID'], inplace=True)
```

3. Explore and visualize the data

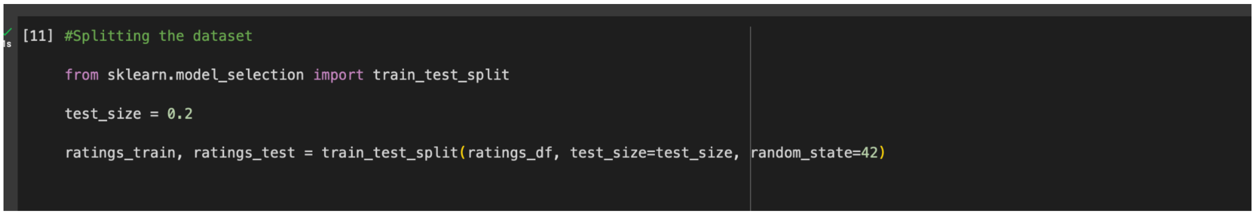


```
# Explore user age distribution
plt.hist(users_df['Age'], bins=20, color='skyblue', edgecolor='black')
plt.title('User Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



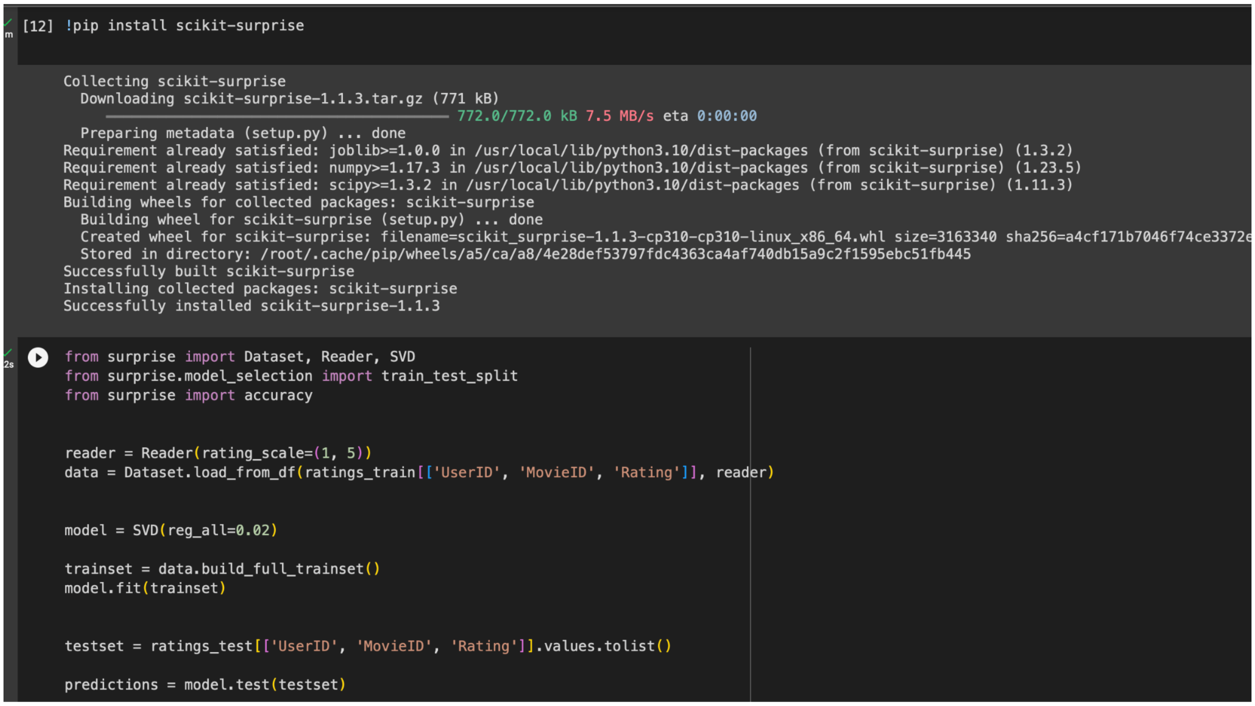
```
# Explore movie ratings distribution
plt.hist(ratings_df['Rating'], bins=5, color='lightcoral', edgecolor='black', range=(0.5, 5.5))
plt.title('Movie Ratings Distribution')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.xticks([1, 2, 3, 4, 5])
plt.show()
```

4. Split the dataset.



```
[11] #Splitting the dataset

     from sklearn.model_selection import train_test_split

     test_size = 0.2

     ratings_train, ratings_test = train_test_split(ratings_df, test_size=test_size, random_state=42)
```

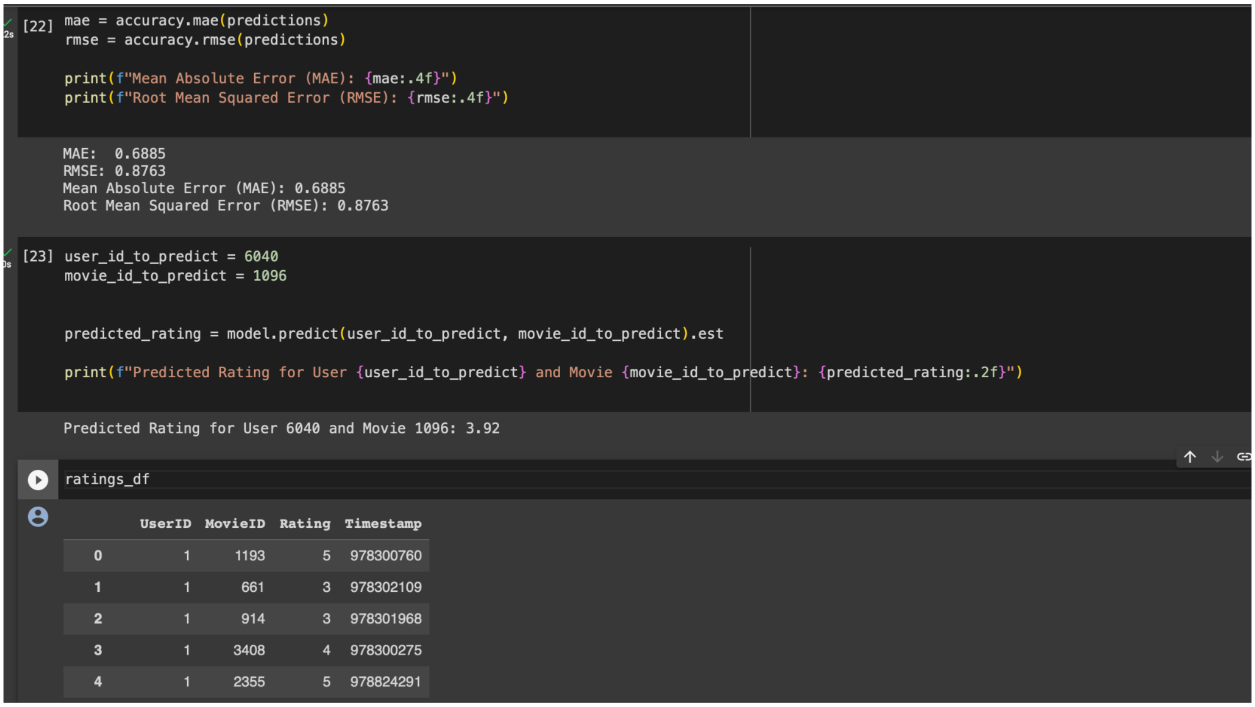
5. Model training and testing.



```
[12] !pip install scikit-surprise
```

```
Collecting scikit-surprise
  Downloading scikit-surprise-1.1.3.tar.gz (771 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 772.0/772.0 kB 7.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.3.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.11.3)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.3-cp310-cp310-linux_x86_64.whl size=3163340 sha256=a4cf171b7046f74ce3372e
  Stored in directory: /root/.cache/pip/wheels/a5/ca/a8/4e28def53797fdc4363ca4af740db15a9c2f1595ebc51fb445
Successfully built scikit-surprise
Installing collected packages: scikit-surprise
Successfully installed scikit-surprise-1.1.3
```

```python
from surprise import Dataset, Reader, SVD
from surprise.model_selection import train_test_split
from surprise import accuracy


reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(ratings_train[['UserID', 'MovieID', 'Rating']], reader)


model = SVD(reg_all=0.02)

trainset = data.build_full_trainset()
model.fit(trainset)


testset = ratings_test[['UserID', 'MovieID', 'Rating']].values.tolist()

predictions = model.test(testset)
```



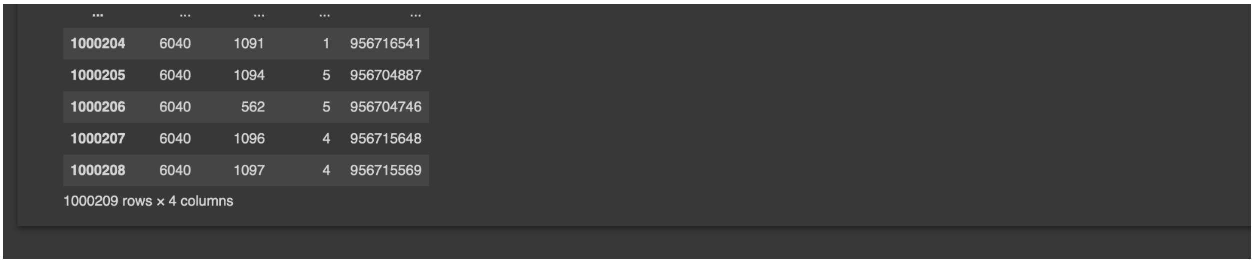
```python
[22] mae = accuracy.mae(predictions)
     rmse = accuracy.rmse(predictions)

     print(f"Mean Absolute Error (MAE): {mae:.4f}")
     print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
```

```
MAE:  0.6885
RMSE: 0.8763
Mean Absolute Error (MAE): 0.6885
Root Mean Squared Error (RMSE): 0.8763
```

```python
[23] user_id_to_predict = 6040
     movie_id_to_predict = 1096


     predicted_rating = model.predict(user_id_to_predict, movie_id_to_predict).est

     print(f"Predicted Rating for User {user_id_to_predict} and Movie {movie_id_to_predict}: {predicted_rating:.2f}")
```

```
Predicted Rating for User 6040 and Movie 1096: 3.92
```

```python
ratings_df
```



	UserID	MovieID	Rating	Timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291
...	...	...	...	...
1000204	6040	1091	1	956716541
1000205	6040	1094	5	956704887
1000206	6040	562	5	956704746
1000207	6040	1096	4	956715648
1000208	6040	1097	4	956715569

1000209 rows × 4 columns

The predicted rating is **3.9** and the actual rating is **4**.