Building User-Based Recommendation Model for Amazon .

November 9, 2022

DESCRIPTION

The dataset provided contains movie reviews given by Amazon customers. Reviews were given between May 1996 and July 2014.

Data Dictionary

UserID – 4848 customers who provided a rating for each movie

Movie 1 to Movie 206 – 206 movies for which ratings are provided by 4848 distinct users

Data Considerations

All the users have not watched all the movies and therefore, all movies are not rated. These missing values are represented by NA.

Ratings are on a scale of -1 to 10 where -1 is the least rating and 10 is the best.

Analysis Task

Exploratory Data Analysis:

Which movies have maximum views/ratings?

What is the average rating for each movie? Define the top 5 movies with the maximum ratings.

Define the top 5 movies with the least audience.

Recommendation Model: Some of the movies hadn't been watched and therefore, are not rated by the users. Netflix would like to take this as an opportunity and build a machine learning recommendation algorithm which provides the ratings for each of the users.

Divide the data into training and test data

Build a recommendation model on training data

Make predictions on the test data

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
ratings=pd.read_csv('Amazon - Movies and Tv Ratings.csv', index_col=0)
ratings.head()
```

[2]:
```
                Movie1  Movie2  Movie3  Movie4  Movie5  Movie6  Movie7  \
user_id
A3R5OBKS7OM2IR     5.0     5.0     NaN     NaN     NaN     NaN     NaN
AH3QC2PC1VTGP      NaN     NaN     2.0     NaN     NaN     NaN     NaN
A3LKP6WPMP9UKX     NaN     NaN     NaN     5.0     NaN     NaN     NaN
AVIY68KEPQ5ZD      NaN     NaN     NaN     5.0     NaN     NaN     NaN
A1CV1WROP5KTTW     NaN     NaN     NaN     NaN     5.0     NaN     NaN

                Movie8  Movie9  Movie10  …  Movie197  Movie198  Movie199  \
user_id                                  …
A3R5OBKS7OM2IR     NaN     NaN      NaN  …       NaN       NaN       NaN
AH3QC2PC1VTGP      NaN     NaN      NaN  …       NaN       NaN       NaN
A3LKP6WPMP9UKX     NaN     NaN      NaN  …       NaN       NaN       NaN
AVIY68KEPQ5ZD      NaN     NaN      NaN  …       NaN       NaN       NaN
A1CV1WROP5KTTW     NaN     NaN      NaN  …       NaN       NaN       NaN

                Movie200  Movie201  Movie202  Movie203  Movie204  Movie205  \
user_id
A3R5OBKS7OM2IR       NaN       NaN       NaN       NaN       NaN       NaN
AH3QC2PC1VTGP        NaN       NaN       NaN       NaN       NaN       NaN
A3LKP6WPMP9UKX       NaN       NaN       NaN       NaN       NaN       NaN
AVIY68KEPQ5ZD        NaN       NaN       NaN       NaN       NaN       NaN
A1CV1WROP5KTTW       NaN       NaN       NaN       NaN       NaN       NaN

                Movie206
user_id
A3R5OBKS7OM2IR       NaN
AH3QC2PC1VTGP        NaN
A3LKP6WPMP9UKX       NaN
AVIY68KEPQ5ZD        NaN
A1CV1WROP5KTTW       NaN

[5 rows x 206 columns]
```

[3]:
```python
ratings.tail()
```

[3]:
```
                Movie1  Movie2  Movie3  Movie4  Movie5  Movie6  Movie7  \
user_id
A1IMQ9WMFYKWH5     NaN     NaN     NaN     NaN     NaN     NaN     NaN
A1KLIKPUF5E88I     NaN     NaN     NaN     NaN     NaN     NaN     NaN
A5HG6WFZLO10D      NaN     NaN     NaN     NaN     NaN     NaN     NaN
A3UU690TWXCG1X     NaN     NaN     NaN     NaN     NaN     NaN     NaN
AI4J762YI6S06      NaN     NaN     NaN     NaN     NaN     NaN     NaN

                Movie8  Movie9  Movie10  …  Movie197  Movie198  Movie199  \
user_id                                  …
A1IMQ9WMFYKWH5     NaN     NaN      NaN  …       NaN       NaN       NaN
```

```
A1KLIKPUF5E88I     NaN     NaN     NaN  ...      NaN      NaN      NaN
A5HG6WFZLO10D      NaN     NaN     NaN  ...      NaN      NaN      NaN
A3UU690TWXCG1X     NaN     NaN     NaN  ...      NaN      NaN      NaN
AI4J762YI6S06      NaN     NaN     NaN  ...      NaN      NaN      NaN

                Movie200  Movie201  Movie202  Movie203  Movie204  Movie205  \
user_id
A1IMQ9WMFYKWH5       NaN       NaN       NaN       NaN       NaN       NaN
A1KLIKPUF5E88I       NaN       NaN       NaN       NaN       NaN       NaN
A5HG6WFZLO10D        NaN       NaN       NaN       NaN       NaN       NaN
A3UU690TWXCG1X       NaN       NaN       NaN       NaN       NaN       NaN
AI4J762YI6S06        NaN       NaN       NaN       NaN       NaN       NaN

                Movie206
user_id
A1IMQ9WMFYKWH5       5.0
A1KLIKPUF5E88I       5.0
A5HG6WFZLO10D        5.0
A3UU690TWXCG1X       5.0
AI4J762YI6S06        5.0

[5 rows x 206 columns]
```

[4]:
```python
ratings.describe()
```

[4]:
```
       Movie1  Movie2  Movie3  Movie4     Movie5  Movie6  Movie7  Movie8  \
count     1.0     1.0     1.0     2.0  29.000000     1.0     1.0     1.0
mean      5.0     5.0     2.0     5.0   4.103448     4.0     5.0     5.0
std       NaN     NaN     NaN     0.0   1.496301     NaN     NaN     NaN
min       5.0     5.0     2.0     5.0   1.000000     4.0     5.0     5.0
25%       5.0     5.0     2.0     5.0   4.000000     4.0     5.0     5.0
50%       5.0     5.0     2.0     5.0   5.000000     4.0     5.0     5.0
75%       5.0     5.0     2.0     5.0   5.000000     4.0     5.0     5.0
max       5.0     5.0     2.0     5.0   5.000000     4.0     5.0     5.0

       Movie9  Movie10  ...  Movie197  Movie198  Movie199  Movie200  Movie201  \
count     1.0      1.0  ...  5.000000       2.0       1.0  8.000000  3.000000
mean      5.0      5.0  ...  3.800000       5.0       5.0  4.625000  4.333333
std       NaN      NaN  ...  1.643168       0.0       NaN  0.517549  1.154701
min       5.0      5.0  ...  1.000000       5.0       5.0  4.000000  3.000000
25%       5.0      5.0  ...  4.000000       5.0       5.0  4.000000  4.000000
50%       5.0      5.0  ...  4.000000       5.0       5.0  5.000000  5.000000
75%       5.0      5.0  ...  5.000000       5.0       5.0  5.000000  5.000000
max       5.0      5.0  ...  5.000000       5.0       5.0  5.000000  5.000000

       Movie202  Movie203  Movie204   Movie205   Movie206
count  6.000000       1.0  8.000000  35.000000  13.000000
```

```
mean   4.333333      3.0  4.375000  4.628571  4.923077
std    1.632993      NaN  1.407886  0.910259  0.277350
min    1.000000      3.0  1.000000  1.000000  4.000000
25%    5.000000      3.0  4.750000  5.000000  5.000000
50%    5.000000      3.0  5.000000  5.000000  5.000000
75%    5.000000      3.0  5.000000  5.000000  5.000000
max    5.000000      3.0  5.000000  5.000000  5.000000

[8 rows x 206 columns]
```

[5]:
```
ratings.dtypes
```

[5]:
```
Movie1       float64
Movie2       float64
Movie3       float64
Movie4       float64
Movie5       float64
              …
Movie202     float64
Movie203     float64
Movie204     float64
Movie205     float64
Movie206     float64
Length: 206, dtype: object
```

[6]:
```
ratings.isna().sum()
```

[6]:
```
Movie1       4847
Movie2       4847
Movie3       4847
Movie4       4846
Movie5       4819
             …
Movie202     4842
Movie203     4847
Movie204     4840
Movie205     4813
Movie206     4835
Length: 206, dtype: int64
```

[7]:
```
ratings.fillna(0)
```

[7]:
```
                Movie1  Movie2  Movie3  Movie4  Movie5  Movie6  Movie7  \
user_id
A3R5OBKS7OM2IR     5.0     5.0     0.0     0.0     0.0     0.0     0.0
AH3QC2PC1VTGP      0.0     0.0     2.0     0.0     0.0     0.0     0.0
A3LKP6WPMP9UKX     0.0     0.0     0.0     5.0     0.0     0.0     0.0
```

```
AVIY68KEPQ5ZD      0.0     0.0     0.0     5.0     0.0     0.0     0.0
A1CV1WROP5KTTW     0.0     0.0     0.0     0.0     5.0     0.0     0.0
…                 …     …     …     …     …     …     …
A1IMQ9WMFYKWH5     0.0     0.0     0.0     0.0     0.0     0.0     0.0
A1KLIKPUF5E88I     0.0     0.0     0.0     0.0     0.0     0.0     0.0
A5HG6WFZLO1OD      0.0     0.0     0.0     0.0     0.0     0.0     0.0
A3UU690TWXCG1X     0.0     0.0     0.0     0.0     0.0     0.0     0.0
AI4J762YI6S06      0.0     0.0     0.0     0.0     0.0     0.0     0.0

                Movie8  Movie9  Movie10  …  Movie197  Movie198  Movie199  \
user_id                                  …
A3R5OBKS7OM2IR     0.0     0.0      0.0  …       0.0       0.0       0.0
AH3QC2PC1VTGP      0.0     0.0      0.0  …       0.0       0.0       0.0
A3LKP6WPMP9UKX     0.0     0.0      0.0  …       0.0       0.0       0.0
AVIY68KEPQ5ZD      0.0     0.0      0.0  …       0.0       0.0       0.0
A1CV1WROP5KTTW     0.0     0.0      0.0  …       0.0       0.0       0.0
…                 …     …       …  …        …       …       …
A1IMQ9WMFYKWH5     0.0     0.0      0.0  …       0.0       0.0       0.0
A1KLIKPUF5E88I     0.0     0.0      0.0  …       0.0       0.0       0.0
A5HG6WFZLO1OD      0.0     0.0      0.0  …       0.0       0.0       0.0
A3UU690TWXCG1X     0.0     0.0      0.0  …       0.0       0.0       0.0
AI4J762YI6S06      0.0     0.0      0.0  …       0.0       0.0       0.0

                Movie200  Movie201  Movie202  Movie203  Movie204  Movie205  \
user_id
A3R5OBKS7OM2IR       0.0       0.0       0.0       0.0       0.0       0.0
AH3QC2PC1VTGP        0.0       0.0       0.0       0.0       0.0       0.0
A3LKP6WPMP9UKX       0.0       0.0       0.0       0.0       0.0       0.0
AVIY68KEPQ5ZD        0.0       0.0       0.0       0.0       0.0       0.0
A1CV1WROP5KTTW       0.0       0.0       0.0       0.0       0.0       0.0
…                   …       …       …       …       …       …
A1IMQ9WMFYKWH5       0.0       0.0       0.0       0.0       0.0       0.0
A1KLIKPUF5E88I       0.0       0.0       0.0       0.0       0.0       0.0
A5HG6WFZLO1OD        0.0       0.0       0.0       0.0       0.0       0.0
A3UU690TWXCG1X       0.0       0.0       0.0       0.0       0.0       0.0
AI4J762YI6S06        0.0       0.0       0.0       0.0       0.0       0.0

                Movie206
user_id
A3R5OBKS7OM2IR       0.0
AH3QC2PC1VTGP        0.0
A3LKP6WPMP9UKX       0.0
AVIY68KEPQ5ZD        0.0
A1CV1WROP5KTTW       0.0
…                   …
A1IMQ9WMFYKWH5       5.0
A1KLIKPUF5E88I       5.0
```

```
A5HG6WFZLO10D       5.0
A3UU690TWXCG1X      5.0
AI4J762YI6S06       5.0

[4848 rows x 206 columns]
```

1 Exploratory Data Analysis:

[14]:
```python
rating_stack=ratings.stack().reset_index()
rating_stack
```

[14]:
```
             user_id   level_1    0
0     A3R5OBKS7OM2IR    Movie1  5.0
1     A3R5OBKS7OM2IR    Movie2  5.0
2      AH3QC2PC1VTGP    Movie3  2.0
3     A3LKP6WPMP9UKX    Movie4  5.0
4      AVIY68KEPQ5ZD    Movie4  5.0
…                  …        …   …
4995  A1IMQ9WMFYKWH5  Movie206  5.0
4996  A1KLIKPUF5E88I  Movie206  5.0
4997   A5HG6WFZLO10D  Movie206  5.0
4998  A3UU690TWXCG1X  Movie206  5.0
4999   AI4J762YI6S06  Movie206  5.0

[5000 rows x 3 columns]
```

[15]:
```python
rating_stack.columns = ['User_ID','Movie','Rating']
```

[16]:
```python
n_ratings = len(rating_stack)
```

[17]:
```python
n_movies = len(rating_stack['Movie'].unique())
n_users = len(rating_stack['User_ID'].unique())
```

[18]:
```python
print(f"Number of ratings: {n_ratings}")
print(f"Number of unique movieId's: {n_movies}")
print(f"Number of unique users: {n_users}")
print(f"Average ratings per user: {round(n_ratings/n_users, 2)}")
print(f"Average ratings per movie: {round(n_ratings/n_movies, 2)}")
```

```
Number of ratings: 5000
Number of unique movieId's: 206
Number of unique users: 4848
Average ratings per user: 1.03
Average ratings per movie: 24.27
```

[22]:
```python
rating_stack.head()
```

```
[22]:           User_ID   Movie  Rating
      0  A3R5OBKS7OM2IR  Movie1     5.0
      1  A3R5OBKS7OM2IR  Movie2     5.0
      2   AH3QC2PC1VTGP  Movie3     2.0
      3  A3LKP6WPMP9UKX  Movie4     5.0
      4   AVIY68KEPQ5ZD  Movie4     5.0
```

```
[25]: ratings_mean=rating_stack.groupby('Movie')['Rating'].describe()['mean']
      ratings_mean
```

```
[25]: Movie
      Movie1      5.000000
      Movie10     5.000000
      Movie100    4.000000
      Movie101    5.000000
      Movie102    4.000000
                    …
      Movie95     3.333333
      Movie96     5.000000
      Movie97     4.800000
      Movie98     5.000000
      Movie99     4.000000
      Name: mean, Length: 206, dtype: float64
```

```
[26]: ratings_count = rating_stack.groupby('Movie')['Rating' ].describe()['count']
      ratings_count
```

```
[26]: Movie
      Movie1      1.0
      Movie10     1.0
      Movie100    1.0
      Movie101    5.0
      Movie102    2.0
                 …
      Movie95     6.0
      Movie96     3.0
      Movie97     5.0
      Movie98     1.0
      Movie99     2.0
      Name: count, Length: 206, dtype: float64
```

```
[27]: ratings_concat = pd.concat([ratings_count, ratings_mean], axis = 1)
      ratings_concat
```

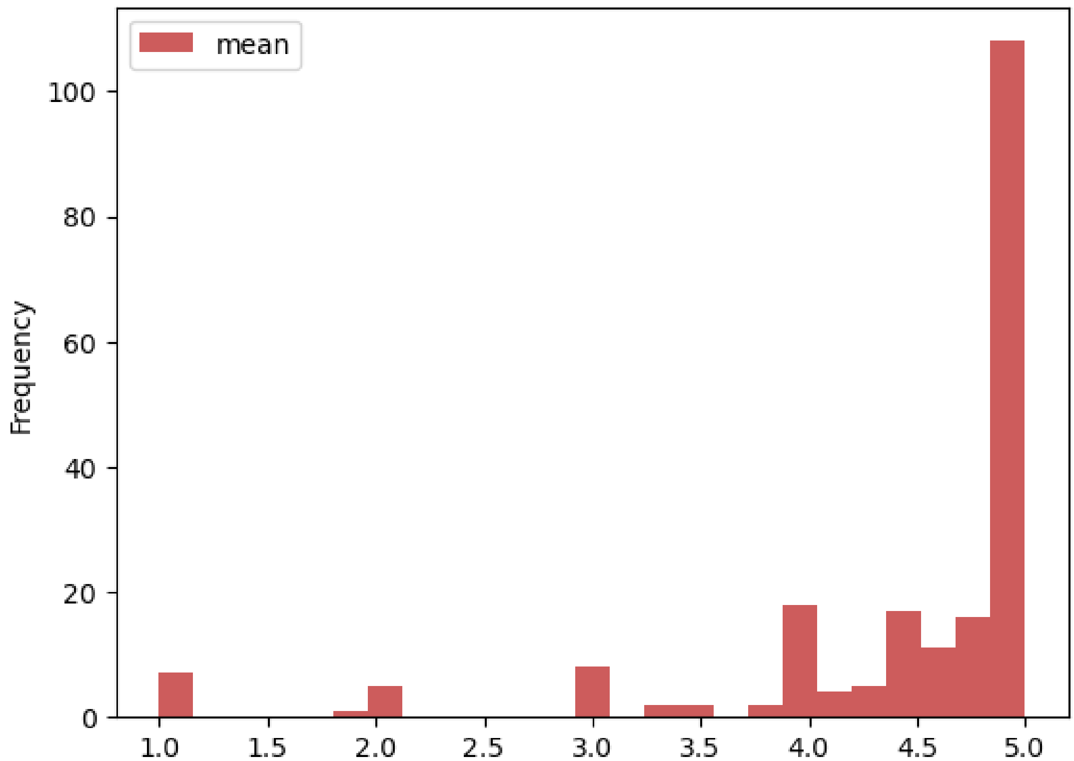
```
[27]:           count      mean
      Movie
      Movie1      1.0  5.000000
      Movie10     1.0  5.000000
```

```
Movie100     1.0  4.000000
Movie101     5.0  5.000000
Movie102     2.0  4.000000
…          …        …
Movie95      6.0  3.333333
Movie96      3.0  5.000000
Movie97      5.0  4.800000
Movie98      1.0  5.000000
Movie99      2.0  4.000000

[206 rows x 2 columns]
```

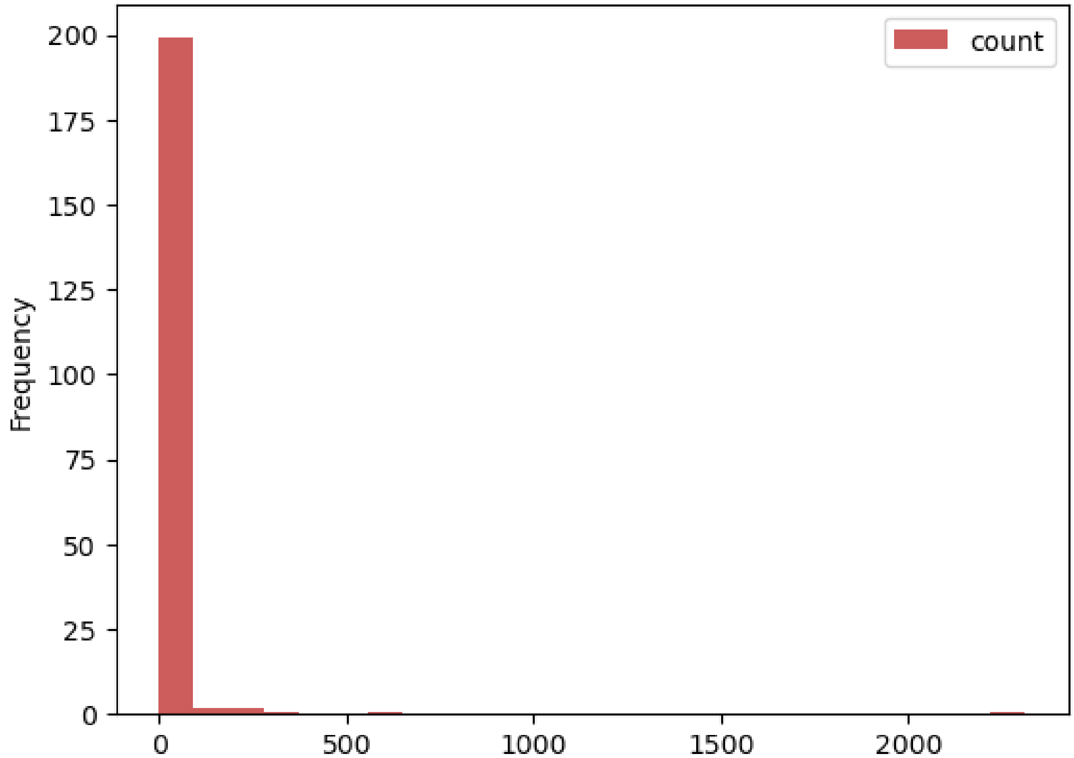
[28]:

```
ratings_concat['mean'].plot(bins=25, kind='hist', color = 'indianred')
plt.legend()
plt.show()
```



[29]:

```
ratings_concat['count'].plot(bins=25, kind='hist', color = 'indianred')
plt.legend()
plt.show()
```



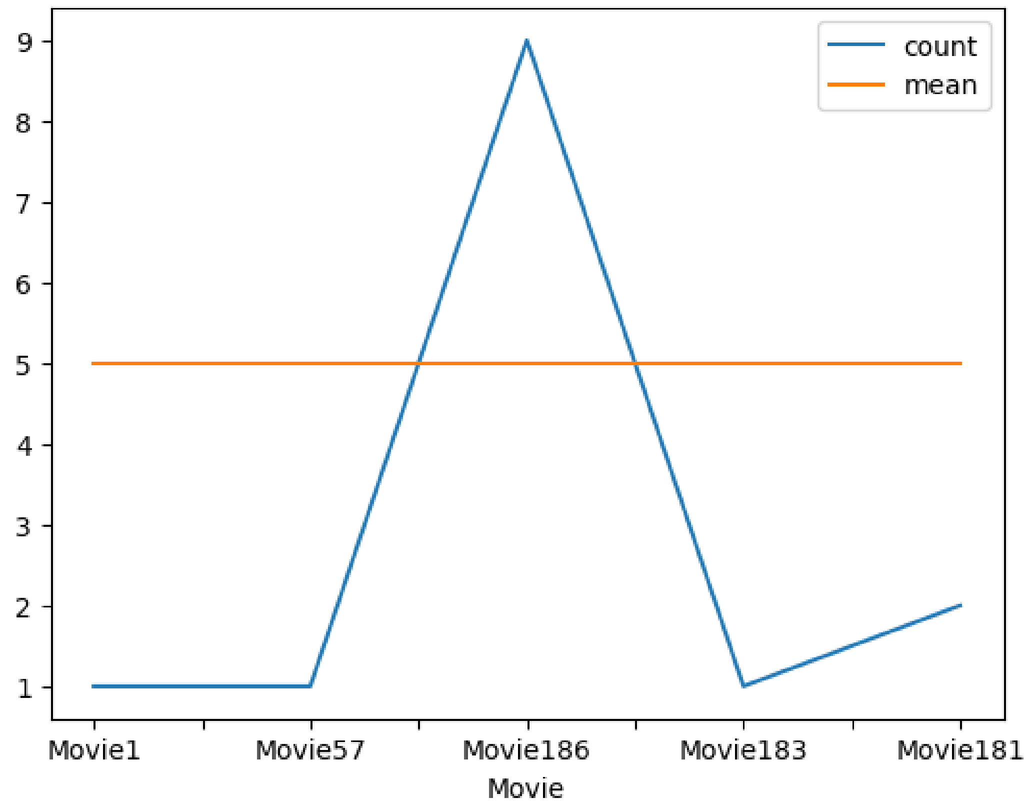
1.0.1 Which movies have maximum views/ratings?

[30]:
```
top=ratings_concat.sort_values('mean', ascending=False).head(5)
top
```

[30]:
```
           count  mean
Movie
Movie1       1.0   5.0
Movie57      1.0   5.0
Movie186     9.0   5.0
Movie183     1.0   5.0
Movie181     2.0   5.0
```

1.0.2 What is the average rating for each movie? Define the top 5 movies with the maximum ratings.

[31]:
```
top.plot()
plt.legend()
plt.show()
```



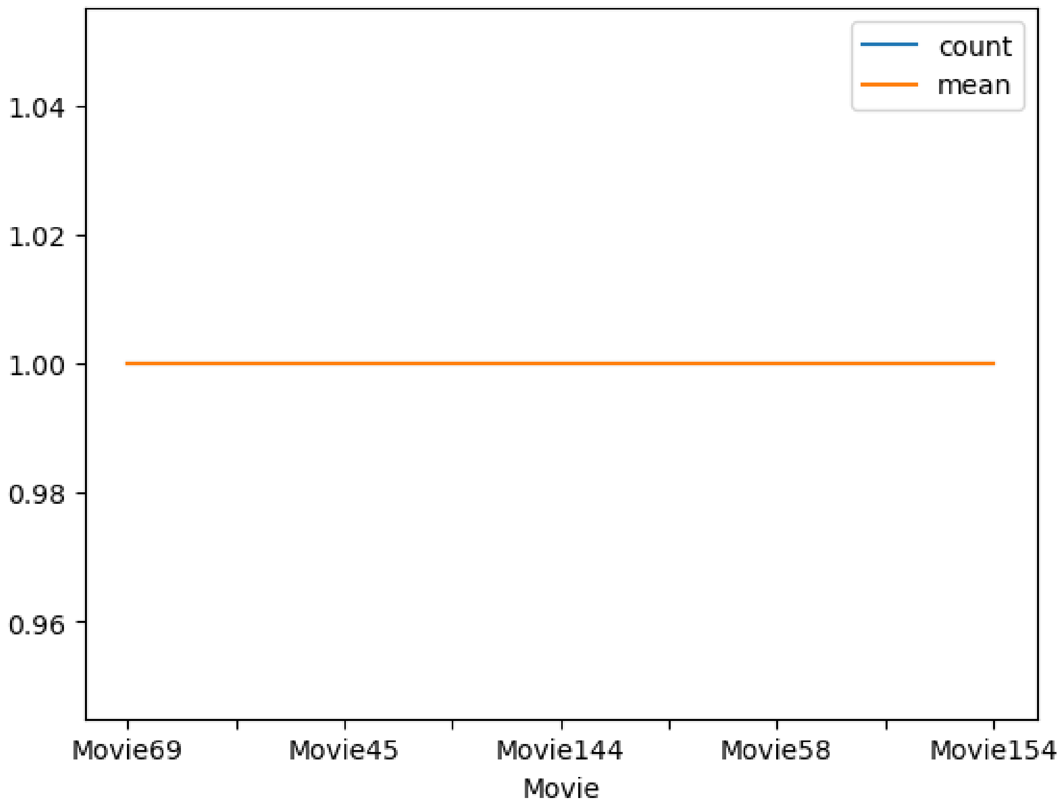
1.0.3 Define the top 5 movies with the least audience.

[32]:
```
least=ratings_concat.sort_values('mean', ascending = True).head(5)
```

[33]:
```
least
```

[33]:
```
          count  mean
Movie
Movie69     1.0   1.0
Movie45     1.0   1.0
Movie144    1.0   1.0
Movie58     1.0   1.0
Movie154    1.0   1.0
```

[34]:
```
least.plot()
plt.legend()
plt.show()
```



1.0.4 Divide the data into training and test data

[35]:
```
from collections import Counter
```

[36]:
```
rating_stack.columns
```

[36]:
```
Index(['User_ID', 'Movie', 'Rating'], dtype='object')
```

[37]:
```
df1=Counter(rating_stack['Movie'])
movie=pd.DataFrame.from_dict(df1,orient='index')
movie
```

[37]:
```
            0
Movie1      1
Movie2      1
Movie3      1
Movie4      2
Movie5     29
…          ..
Movie199    1
Movie203    1
```

```
Movie204   8
Movie205  35
Movie206  13

[206 rows x 1 columns]
```

[38]:
```
x=movie
x=x.head()
x
```

[38]:
```
         0
Movie1   1
Movie2   1
Movie3   1
Movie4   2
Movie5  29
```

[39]:
```
df2=Counter(rating_stack['Rating'])
ratings=pd.DataFrame.from_dict(df2,orient='index')
ratings
```

[39]:
```
        0
5.0  3659
2.0   185
1.0   363
4.0   521
3.0   272
```

[40]:
```
y=ratings
y
```

[40]:
```
        0
5.0  3659
2.0   185
1.0   363
4.0   521
3.0   272
```

[41]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

[42]:
```
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.
↪20,random_state=0)
```

[43]:
```
logreg=LogisticRegression()
```

[44]:
```
x_train.shape
```

[44]: (4, 1)

[45]:
```
x_test.shape
```

[45]: (1, 1)

[46]:
```
y_test.shape
```

[46]: (1, 1)

[47]:
```
y_train.shape
```

[47]: (4, 1)

2 Make predictions on the test data

[48]:
```
logreg.fit(x_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:993:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[48]: LogisticRegression()

[49]:
```
y_pred=logreg.predict(x_test)
y_pred
```

[49]: array([185], dtype=int64)

[50]:
```
y_pred.shape
```

[50]: (1,)

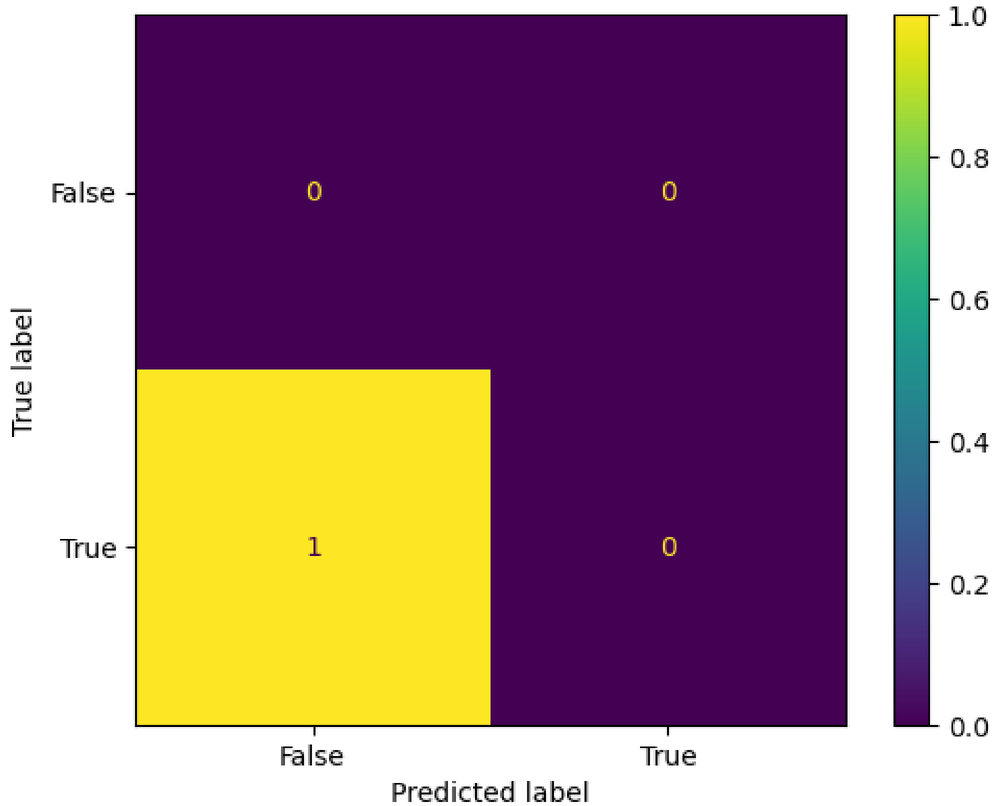
3 Build a recommendation model on training data

[51]:
```
from sklearn import metrics
```

[52]:
```
cmatrix=metrics.confusion_matrix(y_test, y_pred)
```

[53]:
```
cmatrix=metrics.ConfusionMatrixDisplay(confusion_matrix=cmatrix,
                                       display_labels=[False, True])
```

[54]:
```
cmatrix.plot()
plt.show()
```



[55]:
```
from sklearn.metrics import accuracy_score
```

[56]:
```
accuracy_score(y_test, y_pred)
```

[56]: 0.0

[57]:
```
from sklearn.linear_model import LinearRegression
```

[58]:
```
linreg=LinearRegression()
```

[59]:
```
linreg.fit(x_train, y_train)
```

[59]: LinearRegression()

[60]:
```
y_pred1=linreg.predict(x_test)
y_pred1=pd.DataFrame(y_pred1, columns=['Predicted'])
y_pred1
```

[60]:

	Predicted
0	1480.675076

[61]:
```python
y_pred1.shape
```

[61]: (1, 1)

[62]:
```python
from sklearn.metrics import mean_absolute_error,mean_squared_error, r2_score
```

[63]:
```python
mae=mean_absolute_error(y_test, y_pred1)
mae
```

[63]: 1117.675076120052

[64]:
```python
mse=mean_squared_error(y_test, y_pred1)
mse
```

[64]: 1249197.575779964

[65]:
```python
r2=r2_score(y_test, y_pred1)
r2
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py:796:
UndefinedMetricWarning: R^2 score is not well-defined with less than two
samples.
  warnings.warn(msg, UndefinedMetricWarning)
```

[65]: nan

[66]:
```python
from math import sqrt as sqrt
```

[67]:
```python
smse=sqrt(mean_squared_error(y_test, y_pred1))
smse
```

[67]: 1117.675076120052