Amazon_Movies_and_TV_Ratings

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.

Import Important librarys

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```
import warnings
warnings.filterwarnings("ignore")
```

```
# Read Data in the data frame
df=pd.read_csv('Amazon - Movies and TV Ratings.csv')
```

```
# Show data frame
df
```

```
             user_id  Movie1  Movie2  Movie3  Movie4  Movie5  Movie6
Movie7  \
0     A3R5OBKS7OM2IR     5.0     5.0     NaN     NaN     NaN     NaN
NaN
1      AH3QC2PC1VTGP     NaN     NaN     2.0     NaN     NaN     NaN
NaN
2     A3LKP6WPMP9UKX     NaN     NaN     NaN     5.0     NaN     NaN
NaN
3      AVIY68KEPQ5ZD     NaN     NaN     NaN     5.0     NaN     NaN
NaN
4     A1CV1WROP5KTTW     NaN     NaN     NaN     NaN     5.0     NaN
NaN
...              ...     ...     ...     ...     ...     ...     ...
...
```

```
4843  A1IMQ9WMFYKWH5     NaN     NaN     NaN     NaN     NaN     NaN
NaN
4844  A1KLIKPUF5E88I     NaN     NaN     NaN     NaN     NaN     NaN
NaN
4845   A5HG6WFZLO10D     NaN     NaN     NaN     NaN     NaN     NaN
NaN
4846  A3UU690TWXCG1X     NaN     NaN     NaN     NaN     NaN     NaN
NaN
4847   AI4J762YI6S06     NaN     NaN     NaN     NaN     NaN     NaN
NaN

      Movie8  Movie9  ...  Movie197  Movie198  Movie199  Movie200
Movie201  \
0        NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
1        NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
2        NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
3        NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4        NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
...      ...     ...  ...       ...       ...       ...       ...
...
4843     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4844     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4845     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4846     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4847     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN

      Movie202  Movie203  Movie204  Movie205  Movie206
0          NaN       NaN       NaN       NaN       NaN
1          NaN       NaN       NaN       NaN       NaN
2          NaN       NaN       NaN       NaN       NaN
3          NaN       NaN       NaN       NaN       NaN
4          NaN       NaN       NaN       NaN       NaN
...        ...       ...       ...       ...       ...
4843       NaN       NaN       NaN       NaN       5.0
4844       NaN       NaN       NaN       NaN       5.0
4845       NaN       NaN       NaN       NaN       5.0
4846       NaN       NaN       NaN       NaN       5.0
4847       NaN       NaN       NaN       NaN       5.0

[4848 rows x 207 columns]
```

```
# Check head top 5 rows
df.head()

          user_id  Movie1  Movie2  Movie3  Movie4  Movie5  Movie6  
Movie7  \
0  A3R5OBKS7OM2IR     5.0     5.0     NaN     NaN     NaN     NaN     
NaN   
1   AH3QC2PC1VTGP     NaN     NaN     2.0     NaN     NaN     NaN     
NaN   
2  A3LKP6WPMP9UKX     NaN     NaN     NaN     5.0     NaN     NaN     
NaN   
3   AVIY68KEPQ5ZD     NaN     NaN     NaN     5.0     NaN     NaN     
NaN   
4  A1CV1WROP5KTTW     NaN     NaN     NaN     NaN     5.0     NaN     
NaN   

   Movie8  Movie9  ...  Movie197  Movie198  Movie199  Movie200  
Movie201  \
0     NaN     NaN  ...       NaN       NaN       NaN       NaN       
NaN   
1     NaN     NaN  ...       NaN       NaN       NaN       NaN       
NaN   
2     NaN     NaN  ...       NaN       NaN       NaN       NaN       
NaN   
3     NaN     NaN  ...       NaN       NaN       NaN       NaN       
NaN   
4     NaN     NaN  ...       NaN       NaN       NaN       NaN       
NaN   

   Movie202  Movie203  Movie204  Movie205  Movie206  
0       NaN       NaN       NaN       NaN       NaN  
1       NaN       NaN       NaN       NaN       NaN  
2       NaN       NaN       NaN       NaN       NaN  
3       NaN       NaN       NaN       NaN       NaN  
4       NaN       NaN       NaN       NaN       NaN  

[5 rows x 207 columns]

# check tail 5 rows
df.tail()

             user_id  Movie1  Movie2  Movie3  Movie4  Movie5  Movie6  
Movie7  \
4843  A1IMQ9WMFYKWH5     NaN     NaN     NaN     NaN     NaN     NaN     
NaN   
4844  A1KLIKPUF5E88I     NaN     NaN     NaN     NaN     NaN     NaN     
NaN   
4845   A5HG6WFZLO10D     NaN     NaN     NaN     NaN     NaN     NaN     
NaN   
4846  A3UU690TWXCG1X     NaN     NaN     NaN     NaN     NaN     NaN     
```

```
NaN
4847   AI4J762YI6S06     NaN     NaN     NaN     NaN     NaN     NaN
NaN

      Movie8  Movie9  ...  Movie197  Movie198  Movie199  Movie200
Movie201  \
4843     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4844     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4845     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4846     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4847     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN

      Movie202  Movie203  Movie204  Movie205  Movie206
4843       NaN       NaN       NaN       NaN       5.0
4844       NaN       NaN       NaN       NaN       5.0
4845       NaN       NaN       NaN       NaN       5.0
4846       NaN       NaN       NaN       NaN       5.0
4847       NaN       NaN       NaN       NaN       5.0

[5 rows x 207 columns]
```

```
# check rows and columns
df.shape
```

```
(4848, 207)
```

```
df.describe()
```

```
       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]
```

```
df.isna().sum()
```

```
user_id        0
Movie1      4847
Movie2      4847
Movie3      4847
Movie4      4846
            ... 
Movie202    4842
Movie203    4847
Movie204    4840
Movie205    4813
Movie206    4835
Length: 207, dtype: int64
```

```
# Check data informations
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4848 entries, 0 to 4847
Columns: 207 entries, user_id to Movie206
dtypes: float64(206), object(1)
memory usage: 7.7+ MB
```

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.

```
df
```

```
             user_id  Movie1  Movie2  Movie3  Movie4  Movie5  Movie6
Movie7  \
0     A3R5OBKS7OM2IR     5.0     5.0     NaN     NaN     NaN     NaN
NaN
1      AH3QC2PC1VTGP     NaN     NaN     2.0     NaN     NaN     NaN
NaN
2     A3LKP6WPMP9UKX     NaN     NaN     NaN     5.0     NaN     NaN
NaN
3      AVIY68KEPQ5ZD     NaN     NaN     NaN     5.0     NaN     NaN
NaN
4     A1CV1WROP5KTTW     NaN     NaN     NaN     NaN     5.0     NaN
NaN
...              ...     ...     ...     ...     ...     ...     ...
...
4843  A1IMQ9WMFYKWH5     NaN     NaN     NaN     NaN     NaN     NaN
NaN
4844  A1KLIKPUF5E88I     NaN     NaN     NaN     NaN     NaN     NaN
NaN
4845   A5HG6WFZLO10D     NaN     NaN     NaN     NaN     NaN     NaN
NaN
4846  A3UU690TWXCG1X     NaN     NaN     NaN     NaN     NaN     NaN
NaN
4847   AI4J762YI6S06     NaN     NaN     NaN     NaN     NaN     NaN
NaN

      Movie8  Movie9  ...  Movie197  Movie198  Movie199  Movie200
Movie201  \
0        NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
1        NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
2        NaN     NaN  ...       NaN       NaN       NaN       NaN
```

```
NaN
3        NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4        NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
...      ...     ...  ...       ...       ...       ...       ...
...
4843     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4844     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4845     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4846     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN
4847     NaN     NaN  ...       NaN       NaN       NaN       NaN
NaN

      Movie202  Movie203  Movie204  Movie205  Movie206
0          NaN       NaN       NaN       NaN       NaN
1          NaN       NaN       NaN       NaN       NaN
2          NaN       NaN       NaN       NaN       NaN
3          NaN       NaN       NaN       NaN       NaN
4          NaN       NaN       NaN       NaN       NaN
...        ...       ...       ...       ...       ...
4843       NaN       NaN       NaN       NaN       5.0
4844       NaN       NaN       NaN       NaN       5.0
4845       NaN       NaN       NaN       NaN       5.0
4846       NaN       NaN       NaN       NaN       5.0
4847       NaN       NaN       NaN       NaN       5.0

[4848 rows x 207 columns]
```

```
revies = pd.DataFrame((df.apply(lambda x: x.count(),
axis=0)).sort_values(ascending = False)).reset_index()
revies.columns=['Movies Name','Views']
revies
```

```
    Movies Name  Views
0       user_id   4848
1      Movie127   2313
2      Movie140    578
3       Movie16    320
4      Movie103    272
..          ...    ...
202     Movie64      1
203     Movie65      1
204     Movie66      1
205      Movie3      1
206    Movie106      1
```

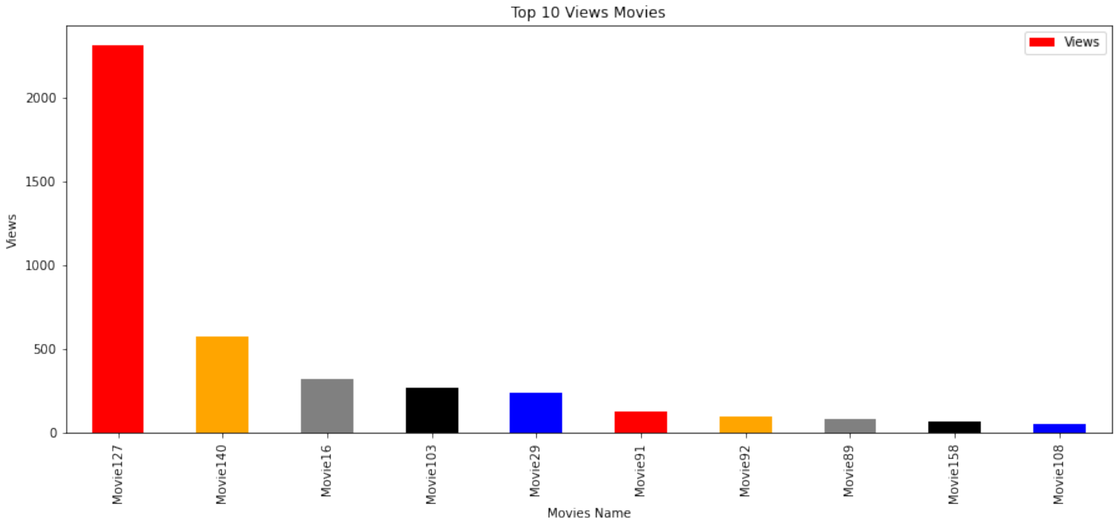
```
[207 rows x 2 columns]

# Drop first index because first row is not needed data
revies=revies.drop(index=[0])

# top 10 views movies
revies=revies.head(10)
revies

   Movies Name  Views
1     Movie127   2313
2     Movie140    578
3      Movie16    320
4     Movie103    272
5      Movie29    243
6      Movie91    128
7      Movie92    101
8      Movie89     83
9     Movie158     66
10    Movie108     54

ec = ['red', 'orange', 'gray', 'black', 'blue']
revies.plot(x='Movies Name',y='Views',kind='bar',title = 'Top 10 Views 
Movies',figsize=(15,6),color = ec)
plt.xlabel('Movies Name')
plt.ylabel('Views')
plt.show()
```



Results

Top 5 Movies are :-

- 1). Movie127
- 2). Movie140

- 3). Movie16
- 4). Movie103
- 5). Movie29

```
(df.drop('user_id',axis=1).sum().sort_values(ascending=False)).head(1)
```

```
Movie127    9511.0
dtype: float64
```

Results

Movie127 have maximum rating and views

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

```
#  the average rating for each movie is
# Pandas Series.to_frame() function is used to convert the given
series object to a dataframe.
average_rating=df.drop('user_id',axis=1).mean().sort_values(ascending=
False).to_frame()
average_rating
```

```
            0
Movie1    5.0
Movie66   5.0
Movie76   5.0
Movie75   5.0
Movie74   5.0
...       ...
Movie58   1.0
Movie60   1.0
Movie154  1.0
Movie45   1.0
Movie144  1.0

[206 rows x 1 columns]
```

```
# the top 5 movies with the maximum ratings
average_rating.head(5)
```

```
           0
Movie1   5.0
Movie66  5.0
Movie76  5.0
Movie75  5.0
Movie74  5.0
```

pandas.DataFrame.T() function

pandas.DataFrame.T property is used to transpose index and columns of the data frame.

The property T is somehow related to method transpose().

The main function of this property is to create a reflection of the data frame overs the main diagonal by making rows as columns and vice versa.

```
df.describe().T
```

```
          count      mean       std  min   25%  50%  75%  max
Movie1      1.0  5.000000       NaN  5.0  5.00  5.0  5.0  5.0
Movie2      1.0  5.000000       NaN  5.0  5.00  5.0  5.0  5.0
Movie3      1.0  2.000000       NaN  2.0  2.00  2.0  2.0  2.0
Movie4      2.0  5.000000  0.000000  5.0  5.00  5.0  5.0  5.0
Movie5     29.0  4.103448  1.496301  1.0  4.00  5.0  5.0  5.0
...         ...       ...       ...  ...   ...  ...  ...  ...
Movie202    6.0  4.333333  1.632993  1.0  5.00  5.0  5.0  5.0
Movie203    1.0  3.000000       NaN  3.0  3.00  3.0  3.0  3.0
Movie204    8.0  4.375000  1.407886  1.0  4.75  5.0  5.0  5.0
Movie205   35.0  4.628571  0.910259  1.0  5.00  5.0  5.0  5.0
Movie206   13.0  4.923077  0.277350  4.0  5.00  5.0  5.0  5.0

[206 rows x 8 columns]
```

```
df.describe().T['count'].sort_values(ascending=True)[:5].to_frame()
```

```
          count
Movie1      1.0
Movie71     1.0
Movie145    1.0
Movie69     1.0
Movie68     1.0
```

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

Import Recommendation Model related Librarys

```
from surprise import Reader
from surprise import accuracy
from surprise import Dataset
from surprise.model_selection import train_test_split
from surprise import SVD
from surprise.model_selection import cross_validate
from surprise.model_selection import GridSearchCV
```

melt() function is useful to message a DataFrame into a format where one or more columns are identifier variables, while all other columns, considered measured variables, are unpivoted to the row axis, leaving just two non-identifier columns, variable and value.

```
df.head(2)
```

```
          user_id  Movie1  Movie2  Movie3  Movie4  Movie5  Movie6  
Movie7  \
0  A3R5OBKS7OM2IR     5.0     5.0     NaN     NaN     NaN     NaN   
NaN   
1   AH3QC2PC1VTGP     NaN     NaN     2.0     NaN     NaN     NaN   
NaN   

   Movie8  Movie9  ...  Movie197  Movie198  Movie199  Movie200  
Movie201  \
0     NaN     NaN  ...       NaN       NaN       NaN       NaN   
NaN   
1     NaN     NaN  ...       NaN       NaN       NaN       NaN   
NaN   

   Movie202  Movie203  Movie204  Movie205  Movie206  
0       NaN       NaN       NaN       NaN       NaN  
1       NaN       NaN       NaN       NaN       NaN  

[2 rows x 207 columns]
```

```
df1=df.melt(id_vars=df.columns[0],value_vars=df.columns[1:],var_name='
Movies',value_name='Rating')
```

```
df1.head(5)
```

```
          user_id  Movies  Rating
0  A3R5OBKS7OM2IR  Movie1     5.0
1   AH3QC2PC1VTGP  Movie1     NaN
2  A3LKP6WPMP9UKX  Movie1     NaN
3   AVIY68KEPQ5ZD  Movie1     NaN
4  A1CV1WROP5KTTW  Movie1     NaN
```

```
read=Reader(rating_scale=(-1,10))
info=Dataset.load_from_df(df1.fillna(0),reader=read)
info
```

```
<surprise.dataset.DatasetAutoFolds at 0x249f45d4eb0>
```

```
trainset,testset=train_test_split(info,test_size=0.20)
```

Singular Value Decomposition (SVD) is one of the widely used methods for dimensionality reduction. SVD decomposes a matrix into three other matrices.

If we see matrices as something that causes a linear transformation in the space then with Singular Value Decomposition we decompose a single transformation in three movements.

```
#Usingn svd
svd = SVD()
```

```
svd.fit(trainset)
```

```
<surprise.prediction_algorithms.matrix_factorization.SVD at
0x249f45d4760>
```

```
predect = svd.test(testset)
```

```
accuracy.rmse(predect)
```

```
RMSE: 0.2790
```

```
0.2789579259730664
```

RMSE is an acronym for **Root Mean Square Error**, which is the square root of value obtained from Mean Square Error function.

Using RMSE, we can easily plot a difference between the estimated and actual values of a parameter of the model.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

```
accuracy.mae(predect)
```

```
MAE:  0.0410
```

```
0.04097044673118672
```

Mean Absolute Error (MAE) is calculated by taking the summation of the absolute difference between the actual and calculated values of each observation over the entire array and then dividing the sum obtained by the number of observations in the array.

(1/n) * Σ|yi – xi|

```
cross_validate(svd, info, measures = ['RMSE', 'MAE'], cv = 3, verbose
= True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.2869  0.2825  0.2767  0.2820  0.0042
```

```
MAE (testset)     0.0427  0.0430  0.0422  0.0426  0.0003  
Fit time          50.37   48.03   48.96   49.12   0.96    
Test time         3.77    4.27    4.52    4.18    0.31    
```

```
{'test_rmse': array([0.28687949, 0.28249254, 0.27665693]),
 'test_mae': array([0.04266601, 0.04300654, 0.04215707]),
 'fit_time': (50.36888766288757, 48.03077936172485,
48.96214151382446),
 'test_time': (3.7666263580322266, 4.2694337368011475,
4.515508413314819)}
```

```
def repeat(ml_type,dframe):
    rd = Reader()
    data = Dataset.load_from_df(dframe,reader=rd)
    print(cross_validate(ml_type, data, measures = ['RMSE', 'MAE'], cv
= 3, verbose = True))
    print("--"*10)
    usr_id = 'A3R5OBKS7OM2IR'
    mv = 'Movie1'
    r_u = 5.0
    print(ml_type.predict(usr_id,mv,r_ui = r_u,verbose=True))
    print("--"*10)
```

```
repeat(SVD(),df1.fillna(df1['Rating'].mean()))
```

```
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std     
RMSE (testset)    0.0837  0.0867  0.0878  0.0861  0.0017  
MAE (testset)     0.0098  0.0096  0.0099  0.0098  0.0001  
Fit time          50.93   48.49   48.09   49.17   1.25    
Test time         4.17    3.45    3.36    3.66    0.36    
{'test_rmse': array([0.08367766, 0.0867451 , 0.08779496]), 'test_mae':
array([0.00981442, 0.00964389, 0.00992693]), 'fit_time':
(50.92977452278137, 48.49196481704712, 48.091508626937866),
'test_time': (4.171383619308472, 3.4535999298095703,
3.3571794033050537)}
--------------------
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 4.40
{'was_impossible': False}
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 4.40
{'was_impossible': False}
--------------------
```

```
repeat(SVD(),df1.fillna(df1['Rating'].median()))
```

```
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std     
RMSE (testset)    0.0921  0.0930  0.0916  0.0922  0.0006  
MAE (testset)     0.0073  0.0071  0.0069  0.0071  0.0001  
```

```
Fit time          47.90   46.56   53.65   49.37   3.07
Test time         4.09    3.87    5.30    4.42    0.63
{'test_rmse': array([0.09213634, 0.09302467, 0.09156844]), 'test_mae':
array([0.00725849, 0.00709672, 0.00693507]), 'fit_time':
(47.90461802482605, 46.56060862541199, 53.64694809913635),
'test_time': (4.08527398109436, 3.874274969100952, 5.304906606674194)}
--------------------
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 5.00
{'was_impossible': False}
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 5.00
{'was_impossible': False}
--------------------
```

```
param_grid = {'n_epochs':[20,30],
             'lr_all':[0.005,0.001],
             'n_factors':[50,100]}
```

```
gs = GridSearchCV(SVD,param_grid,measures=['rmse','mae'],cv=3)
gs.fit(info)
```

```
data1 =
Dataset.load_from_df(df1.fillna(df1['Rating'].mean()),reader=read)
gs.fit(data1)
```

```
gs.best_score
```

```
{'rmse': 0.08479386856507552, 'mae': 0.009075366101835979}
```

```
print(gs.best_score["rmse"])
print(gs.best_params["rmse"])
```

```
0.08479386856507552
{'n_epochs': 30, 'lr_all': 0.001, 'n_factors': 50}
```