Keshav_ML_Simplilearn_Project

October 1, 2022

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

[2]:
```
amz = pd.read_csv("Amazon.csv")
```

[3]:
```
amz.head()
```

[3]:
```
          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]
```

[4]:
```
amz.shape
```

[4]:
```
(4848, 207)
```

[5]:
```
amz.describe().T
```

[5]:

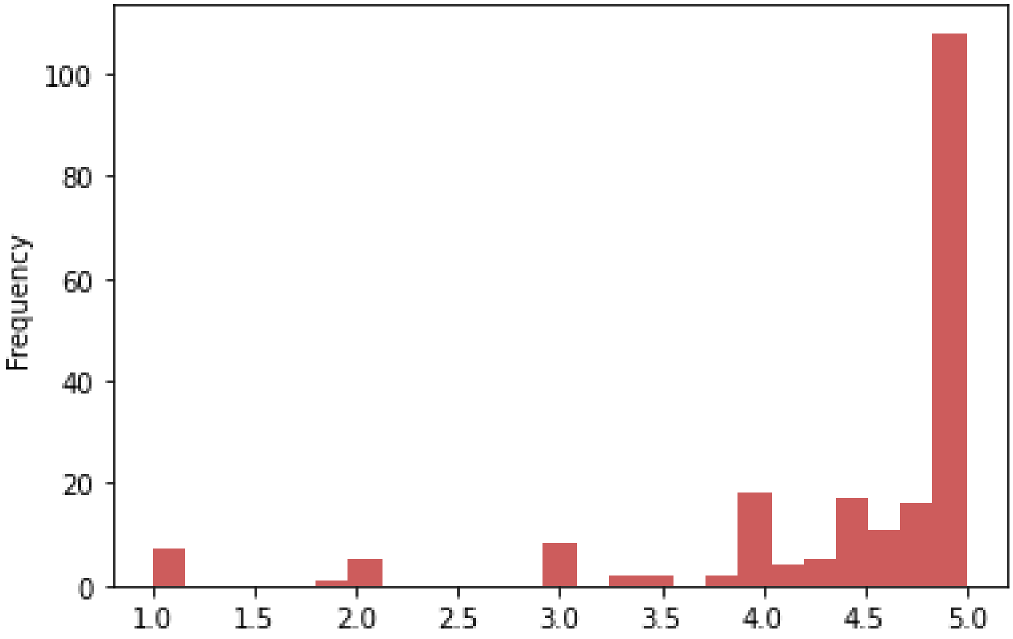
```
           count      mean       std  min   25%  50%  75%  max
Movie1       1.0  5.000000       NaN  5.0  5.00  5.0  5.0  5.0
Movie2       1.0  5.000000       NaN  5.0  5.00  5.0  5.0  5.0
Movie3       1.0  2.000000       NaN  2.0  2.00  2.0  2.0  2.0
Movie4       2.0  5.000000  0.000000  5.0  5.00  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]
```

[6]:

```
amz.describe().T['mean'].plot(bins=25, kind='hist', color = 'indianred')
```

[6]:

```
<AxesSubplot:ylabel='Frequency'>
```

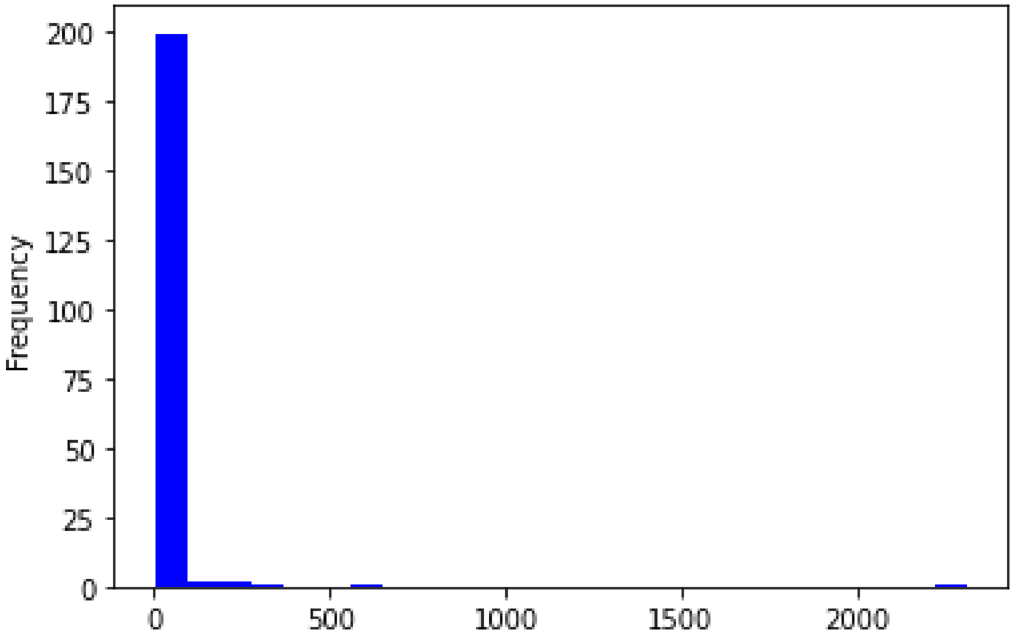


[7]:

```
amz.describe().T['count'].plot(bins=25, kind='hist', color = 'blue')
```

[7]:

```
<AxesSubplot:ylabel='Frequency'>
```



```
[8]: amz.describe().T['count'].sort_values(ascending=False)[:1].to_frame()
```

```
[8]:            count
     Movie127  2313.0
```

```
[9]: amz.drop('user_id',axis=1).sum().sort_values(ascending=False)[:1].to_frame()
```

```
[9]:                0
     Movie127  9511.0
```

```
[10]: amz.drop('user_id',axis=1).mean()
```

```
[10]: Movie1      5.000000
      Movie2      5.000000
      Movie3      2.000000
      Movie4      5.000000
      Movie5      4.103448
                    …
      Movie202    4.333333
      Movie203    3.000000
      Movie204    4.375000
      Movie205    4.628571
      Movie206    4.923077
      Length: 206, dtype: float64
```

```
[11]: amz.drop('user_id',axis=1).mean().sort_values(ascending=False)[0:5].to_frame()
```

```
[11]:             0
      Movie1    5.0
      Movie55   5.0
      Movie131  5.0
      Movie132  5.0
      Movie133  5.0
```

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

```
[12]:           count
      Movie1      1.0
      Movie71     1.0
      Movie145    1.0
      Movie69     1.0
      Movie68     1.0
```

```
[13]: #User based Model building
```

```
[14]: from surprise import Reader
      from surprise import Dataset
      from surprise import accuracy
      from surprise import SVD
      from surprise.model_selection import train_test_split
```

```
[15]: movie_data = amz.melt(id_vars = amz.columns[0],value_vars=amz.columns[1:
       ↪],var_name="Movies",value_name="Rating")
      movie_data
```

```
[15]:                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
      …                    …         …       …
      998683  A1IMQ9WMFYKWH5  Movie206     5.0
      998684  A1KLIKPUF5E88I  Movie206     5.0
      998685   A5HG6WFZLO10D  Movie206     5.0
      998686  A3UU690TWXCG1X  Movie206     5.0
      998687   AI4J762YI6S06  Movie206     5.0

      [998688 rows x 3 columns]
```

```
[16]: rd = Reader(rating_scale=(-1,10))
      data = Dataset.load_from_df(movie_data.fillna(0),reader=rd)
      data
```

```
[16]: <surprise.dataset.DatasetAutoFolds at 0x7fec6df95810>
```

```
[17]: train_data,test_data = train_test_split(data,test_size=0.20)
```

```
[18]: svd = SVD()
```

```
[19]: svd.fit(train_data)
```

```
[19]: <surprise.prediction_algorithms.matrix_factorization.SVD at 0x7fec71a64d10>
```

```
[20]: pred = svd.test(test_data)
```

```
[21]: accuracy.rmse(pred)
```

```
RMSE: 0.2778
```

```
[21]: 0.2777591242040622
```

```
[22]: accuracy.mae(pred)
```

```
MAE:  0.0401
```

```
[22]: 0.040060814814529916
```

```
[23]: u_id='AH3QC2PC1VTGP'
      mv = 'Movie206'
      r_id = 5.0
      svd.predict(u_id, mv, r_ui=r_id, verbose= True)
```

```
user: AH3QC2PC1VTGP item: Movie206   r_ui = 5.00   est = 0.00
{'was_impossible': False}
```

```
[23]: Prediction(uid='AH3QC2PC1VTGP', iid='Movie206', r_ui=5.0,
      est=0.0024259567644680166, details={'was_impossible': False})
```

```
[24]: from surprise.model_selection import cross_validate
```

```
[25]: cross_validate(svd, data, 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)    0.2804  0.2810  0.2855  0.2823  0.0023
MAE (testset)     0.0431  0.0426  0.0428  0.0428  0.0002
Fit time          36.21   36.60   36.57   36.46   0.17
Test time         3.35    3.73    3.61    3.56    0.16
```

[25]:
```
{'test_rmse': array([0.28041109, 0.28096431, 0.28553265]),
 'test_mae': array([0.04314298, 0.04255137, 0.04283306]),
 'fit_time': (36.21399760246277, 36.59962558746338, 36.567951917648315),
 'test_time': (3.3450958728790283, 3.7255845069885254, 3.609577178955078)}
```

[26]:
```
def repeat(ml_type,dframe,min_,max_):
    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)
    u_id = 'AH3QC2PC1VTGP'
    m_id = 'Movie206'
    ra_u = 5.0
    print(ml_type.predict(u_id,mv,r_ui=ra_u,verbose=True))
    print("#"*10)
    print()
```

[27]:
```
amz= amz.iloc[:3000, :50]
movie_data = amz.melt(id_vars = amz.columns[0],value_vars=amz.columns[1:
 ↪],var_name="Movies",value_name="Rating")
```

[28]:
```
repeat(SVD(),movie_data.fillna(0),-1,10)
repeat(SVD(),movie_data.fillna(movie_data.mean()),-1,10)
repeat(SVD(),movie_data.fillna(movie_data.median()),-1,10)
```

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

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    1.0270  1.0294  1.0308  1.0291  0.0016
MAE (testset)     1.0115  1.0126  1.0130  1.0124  0.0006
Fit time          5.21    5.25    5.24    5.23    0.01
Test time         0.36    0.35    0.66    0.46    0.14
{'test_rmse': array([1.0269672 , 1.02941764, 1.03084879]), 'test_mae':
array([1.01152851, 1.01259099, 1.01304796]), 'fit_time': (5.2148354053497314,
5.246253252029419, 5.2354795932769775), 'test_time': (0.3615226745605469,
0.3536815643310547, 0.6646759510040283)}
##########
user: AH3QC2PC1VTGP item: Movie206   r_ui = 5.00   est = 1.00
{'was_impossible': False}
user: AH3QC2PC1VTGP item: Movie206   r_ui = 5.00   est = 1.00
{'was_impossible': False}
##########

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

                  Fold 1  Fold 2  Fold 3  Mean    Std
```

```
RMSE (testset)    0.0571  0.0595  0.0543  0.0569  0.0021
MAE (testset)     0.0073  0.0072  0.0075  0.0073  0.0001
Fit time          5.22    5.24    5.25    5.24    0.01
Test time         0.36    0.68    0.35    0.46    0.15
{'test_rmse': array([0.057096  , 0.05945164, 0.05429029]), 'test_mae':
array([0.00729828, 0.00720864, 0.00749636]), 'fit_time': (5.219885349273682,
5.244813442230225, 5.2529613971710205), 'test_time': (0.3582289218902588,
0.6754562854766846, 0.34947729110717773)}
##########
user: AH3QC2PC1VTGP item: Movie206   r_ui = 5.00   est = 4.54
{'was_impossible': False}
user: AH3QC2PC1VTGP item: Movie206   r_ui = 5.00   est = 4.54
{'was_impossible': False}
##########

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

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.0655  0.0619  0.0595  0.0623  0.0024
MAE (testset)     0.0053  0.0047  0.0051  0.0050  0.0002
Fit time          5.18    5.25    5.25    5.23    0.03
Test time         0.66    0.34    0.65    0.55    0.15
{'test_rmse': array([0.06546078, 0.06192823, 0.05949946]), 'test_mae':
array([0.00528834, 0.00473081, 0.00507342]), 'fit_time': (5.18100643157959,
5.251812934875488, 5.246974468231201), 'test_time': (0.6601049900054932,
0.34183454513549805, 0.6488213539123535)}
##########
user: AH3QC2PC1VTGP item: Movie206   r_ui = 5.00   est = 4.90
{'was_impossible': False}
user: AH3QC2PC1VTGP item: Movie206   r_ui = 5.00   est = 4.90
{'was_impossible': False}
##########
```

[29]:

```
from surprise.model_selection import GridSearchCV
```

[30]:

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

[34]:

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

[35]:

```
gs.best_score
```

[35]:

```
{'rmse': 0.28035460861046646, 'mae': 0.04120152417259141}
```

[36]:

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

```
0.28035460861046646
{'n_epochs': 30, 'lr_all': 0.005, 'n_factors': 100}
```

[ ]:

```

```