```
!pip install surprise
```

```
Requirement already satisfied: surprise in /usr/local/lib/python3.6/dist-packages (0.
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.6/dist-packages
```

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
from sklearn import preprocessing
from surprise import SVD
from surprise import Reader
from surprise import Dataset
from surprise.model_selection import train_test_split
from surprise import accuracy
from sklearn.model_selection import train_test_split
from collections import defaultdict
```

```
import os
import numpy as np
import pandas as pd
%matplotlib inline
import pandas
from sklearn.model_selection import train_test_split
import numpy as np
import time
from sklearn.externals import joblib
import joblib
```

```
from google.colab import drive
drive.mount('/content/drive')
```



```
nt/drive; to attempt to forcibly remount, call drive.m
```

```
ratings_data = pd.read_csv("/content/drive/My Drive/24nov/ratings_Electronics.csv")
```

```
ratings_data.head()
```

	AKM1MP6P0OYPR	0132793040	5.0	1365811200
0	A2CX7LUOHB2NDG	0321732944	5.0	1341100800
1	A2NWSAGRHCP8N5	0439886341	1.0	1367193600
2	A2WNBOD3WNDNKT	0439886341	3.0	1374451200
3	A1GI0U4ZRJA8WN	0439886341	1.0	1334707200
4	A1QGNMC6O1VW39	0511189877	5.0	1397433600

```
ratings_data.columns = ['userid', 'productId', 'ratings', 'timestamp']
```

```
ratings_data.head()
```

	userid	productId	ratings	timestamp
0	A2CX7LUOHB2NDG	0321732944	5.0	1341100800
1	A2NWSAGRHCP8N5	0439886341	1.0	1367193600
2	A2WNBOD3WNDNKT	0439886341	3.0	1374451200
3	A1GI0U4ZRJA8WN	0439886341	1.0	1334707200
4	A1QGNMC6O1VW39	0511189877	5.0	1397433600

```
ratings_data.shape
```

```
(7824481, 4)
```

```
print('#ratings %d' % len(ratings_data.index))
```

```
#ratings 7824481
```

```
print('total unique users %d' % len(ratings_data['userid'].unique()))
```

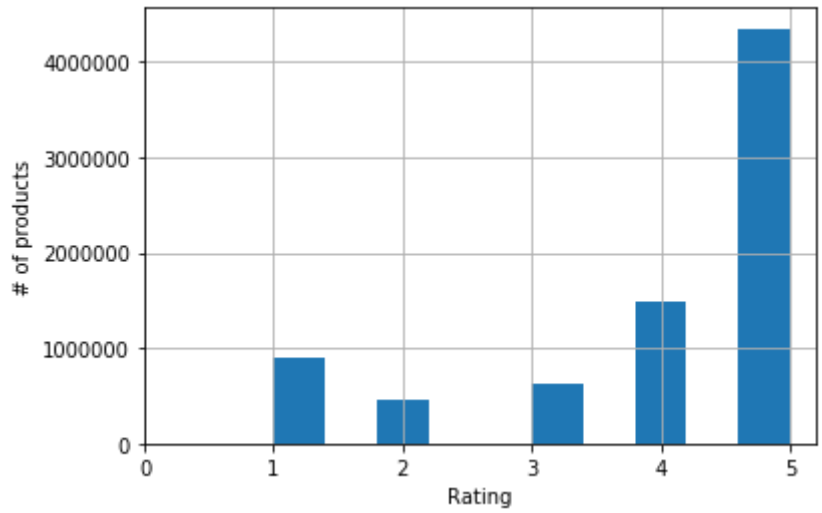
```
total unique users 4201696
```

```
# 5 point summary of ratings
print('min: %.1f' % np.min(ratings_data.ratings))
print('25 percentile: %.1f' % np.percentile(ratings_data.ratings, 25))
print('median: %.1f' % np.median(ratings_data['ratings']))
print('75 percentile: %.1f' % np.percentile(ratings_data.ratings, 75))
print('max: %.1f' % np.max(ratings_data.ratings))
```

```
min: 1.0
25 percentile: 3.0
```

Saved successfully!

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.hist(ratings_data.ratings)
plt.xticks([0, 1.0, 2.0, 3.0, 4.0, 5.0])
plt.xlabel('Rating')
plt.ylabel('# of products')
plt.grid()
plt.show()
```




```
ratings_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7824481 entries, 0 to 7824480
Data columns (total 4 columns):
userid       object
productId    object
ratings      float64
timestamp    int64
dtypes: float64(1), int64(1), object(2)
memory usage: 238.8+ MB
```

```
ratings_data.describe().transpose()
```

	count	mean	std	min	25%	5
ratings	7824481.0	4.012337e+00	1.380910e+00	1.0	3.000000e+00	5.000000e+
timestamp	7824481.0	1.338178e+09	6.900426e+07	912729600.0	1.315354e+09	1.361059e+

```
ratings_data.isnull().values.any()
```

Saved successfully!

```
ratings_data = ratings_data.drop('timestamp',axis=1)

# Users with max no of purchases
ratings_data["userid"].value_counts().head()
```

```
A5JLAU2ARJ0BO     520
ADLVFFE4VBT8      501
A3OXHLG6DIBRW8    498
A6FIAB28IS79      431
A680RUE1FDO8B     406
Name: userid, dtype: int64
```

```
ratings_data.groupby('productId')['ratings'].mean().head()
```

```
productId
0321732944    5.000000
0439886341    1.666667
0511189877    4.500000
0528881469    2.851852
0558835155    3.000000
Name: ratings, dtype: float64
```

```
ratings_data.groupby('productId')['ratings'].mean().sort_values(ascending=False).head()
```

```
productId
BT008V9J9U    5.0
B0058PRC0S    5.0
B0058ORBFU    5.0
B0058OQ9Q2    5.0
B0058OKSMS    5.0
Name: ratings, dtype: float64
```

```
ratings_data.groupby('productId')['ratings'].count().sort_values(ascending=False).head()
```

```
productId
B0074BW614    18244
B00DR0PDNE    16454
B007WTAJTO    14172
B0019EHU8G    12285
B006GWO5WK    12226
Name: ratings, dtype: int64
```

```
#Build Popularity Recommender model.
ratings_mean_count = pd.DataFrame(ratings_data.groupby('productId')['ratings'].mean())
```

```
ratings_mean_count
```

Saved successfully!

productId	ratings
0321732944	5.000000
0439886341	1.666667
0511189877	4.500000
0528881469	2.851852
0558835155	3.000000
...	...
BT008G3W52	5.000000
BT008SXQ4C	1.000000
BT008T2BGK	5.000000
BT008UKTMW	4.000000
BT008V9J9U	5.000000

476001 rows × 1 columns

```
ratings_data
```

	userid	productId	ratings
0	A2CX7LUOHB2NDG	0321732944	5.0
1	A2NWSAGRHCP8N5	0439886341	1.0
2	A2WNBOD3WNDNKT	0439886341	3.0
3	A1GI0U4ZRJA8WN	0439886341	1.0
4	A1QGNMC6O1VW39	0511189877	5.0
...	...	...	...
		BT008UKTMW	5.0
7824477	A322MDK0M89RHN	BT008UKTMW	5.0
7824478	A1MH90R0ADMIK0	BT008UKTMW	4.0
7824479	A10M2KEFPEQDHN	BT008UKTMW	4.0
7824480	A2G81TMIOIDEQQ	BT008V9J9U	5.0

7824481 rows × 3 columns

Saved successfully!

```
#Making Data Sparser
user_counts= ratings_data['userid'].value_counts()
user_counts.head()
```

```
A5JLAU2ARJ0BO     520
ADLVFFE4VBT8      501
A3OXHLG6DIBRW8    498
A6FIAB28IS79      431
A680RUE1FDO8B     406
Name: userid, dtype: int64
```

```
ratings_data.shape
```

```
(7824481, 3)
```

```
ratings_new=ratings_data[ratings_data['userid'].isin(user_counts[user_counts >=50].index)]
ratings_new.head()
```

	userid	productId	ratings
93	A3BY5KCNQZXV5U	0594451647	5.0
117	AT09WGFUM934H	0594481813	3.0
176	A32HSNCNPRUMTR	0970407998	1.0
177	A17HMM1M7T9PJ1	0970407998	4.0
491	A3CLWR1UUZT6TG	0972683275	5.0

```
ratings_new.shape
```

```
(125871, 3)
```

```
ratings_new1 = ratings_new.sample(frac=0.1, random_state=1)
ratings_new1.shape
```

```
(12587, 3)
```

```
from surprise import KNNWithMeans
from surprise import accuracy
                                    train_test_split
```

Saved successfully!

```
reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(ratings_new1[['userid', 'productId', 'ratings']], reader)
data
```

```
<surprise.dataset.DatasetAutoFolds at 0x7f39d6da2a58>
```

```
# Split data to train and test
from surprise.model_selection import train_test_split
trainset, testset = train_test_split(data, test_size=.30,random_state=123)

# Build Collaborative Filtering model.
user_records = trainset.ur
type(user_records)
for keys in user_records.keys():
    print(keys)
```

```
print(keys)
```

Saved successfully!

1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511

Saved successfully!

1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527

Saved successfully!

Saved successfully!

Saved successfully!