About Practice Problem: Is this joke funny?

Many online businesses rely on customer reviews and ratings. Explicit feedback is especially important in the entertainment and ecommerce industry where all customer engagements are impacted by these ratings. Netflix relies on such rating data to power its recommendation engine to provide best movie and TV series recommendations that are personalized and most relevant to the user.

This practice problem challenges the participants to predict the ratings for jokes given by the users provided the ratings provided by the same users for another set of jokes. This dataset is taken from the famous jester online Joke Recommender system dataset.

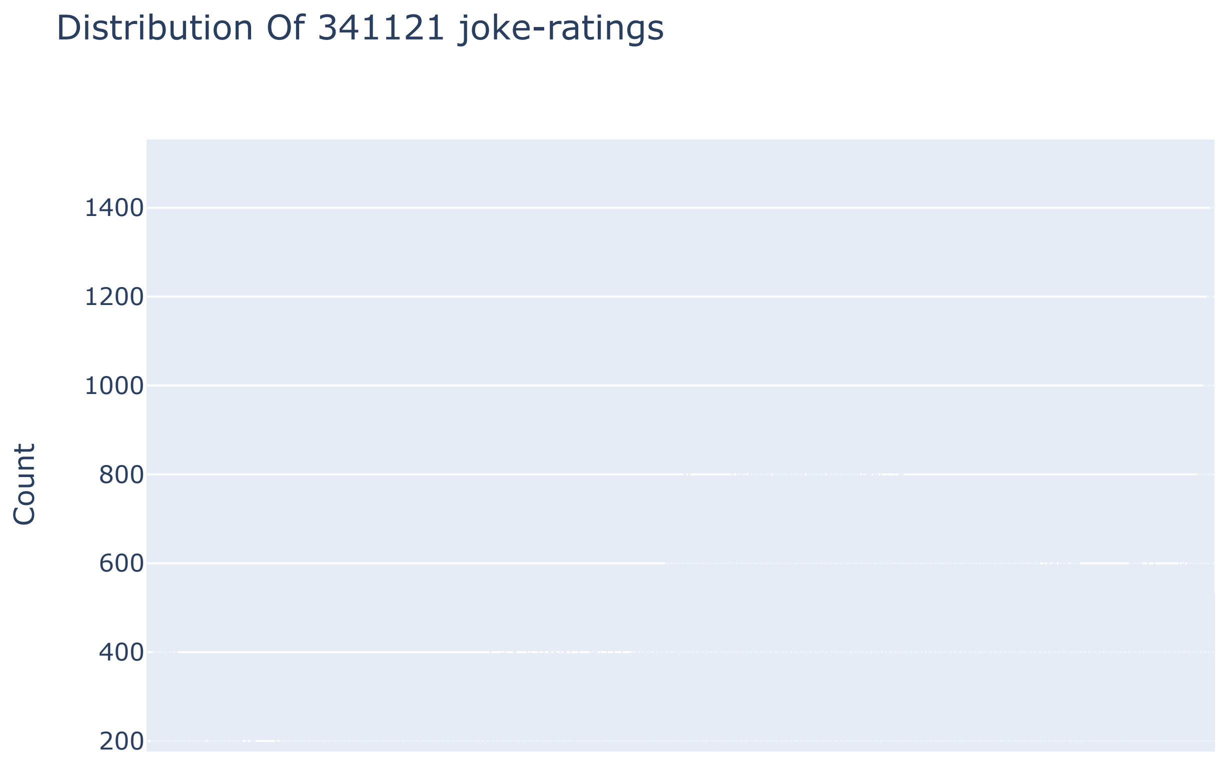
```
!pip install surprise
```

```
Collecting surprise
  Downloading https://files.pythonhosted.org/packages/61/de/e5cba8682201fcf9c3719a6f
Collecting scikit-surprise
  Downloading https://files.pythonhosted.org/packages/97/37/5d334adaf5ddd65da99fc65f
     |████████████████████████████████| 11.8MB 9.6MB/s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.1-cp36-cp36m-linux_
  Stored in directory: /root/.cache/pip/wheels/78/9c/3d/41b419c9d2aff5b6e2b4c0fc8d25
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.1 surprise-0.1
```

```
import pandas as pd
from surprise import Reader,Dataset
from surprise.model_selection import cross_validate,KFold,train_test_split
from surprise import KNNBasic
from surprise import KNNWithMeans,KNNWithZScore,KNNBaseline
from surprise import SVD,SVDpp
from surprise import BaselineOnly
from surprise import NMF,SlopeOne,CoClustering
from surprise import NormalPredictor
from surprise import accuracy
from surprise.accuracy import rmse
from plotly.offline import init_notebook_mode, plot, iplot
import plotly.graph_objs as go
init_notebook_mode(connected=True)
%matplotlib inline
```

```
train_data = pd.read_csv("train.csv")
```

```
jokes_data = pd.read_csv("jokes.csv")
test_data = pd.read_csv("test.csv")



df = pd.read_csv('train.csv')
reader = Reader(rating_scale=(0, 5))
data = Dataset.load_from_df(df[['user_id','joke_id','Rating']], reader)
trainingSet = data.build_full_trainset()


"""
Distribution of Ratings
"""
data = df['Rating'].value_counts().sort_index(ascending=False)
trace = go.Bar(x = data.index,
               text = ['{:.1f} %'.format(val) for val in (data.values / df.shape[0] * 100)
               textposition = 'auto',
               textfont = dict(color = '#000000'),
               y = data.values,
               )
# Create layout
layout = dict(title = 'Distribution Of {} joke-ratings'.format(df.shape[0]),
              xaxis = dict(title = 'Rating'),
              yaxis = dict(title = 'Count'))
# Create plot
fig = go.Figure(data=[trace], layout=layout)
#iplot(fig)
fig.show(renderer="colab")


"""
Rating Distribution by Jokes
"""
# Number of ratings per joke
data=df.groupby('joke_id')['Rating'].count().clip(upper=150)
trace=go.Histogram(x=data.values,
                   name='Ratings',
                   xbins=dict(start=0,
                              end=150,
                              size=2))
layout=dict(title ='Distribution of Rating per Jokes',
            xaxis=dict(title='No. of ratings per day'),
            yaxis=dict(title='count'),
            bargap=0.2)

figure=go.Figure(data=[trace],layout=layout)
fig.show(renderer='colab')
```



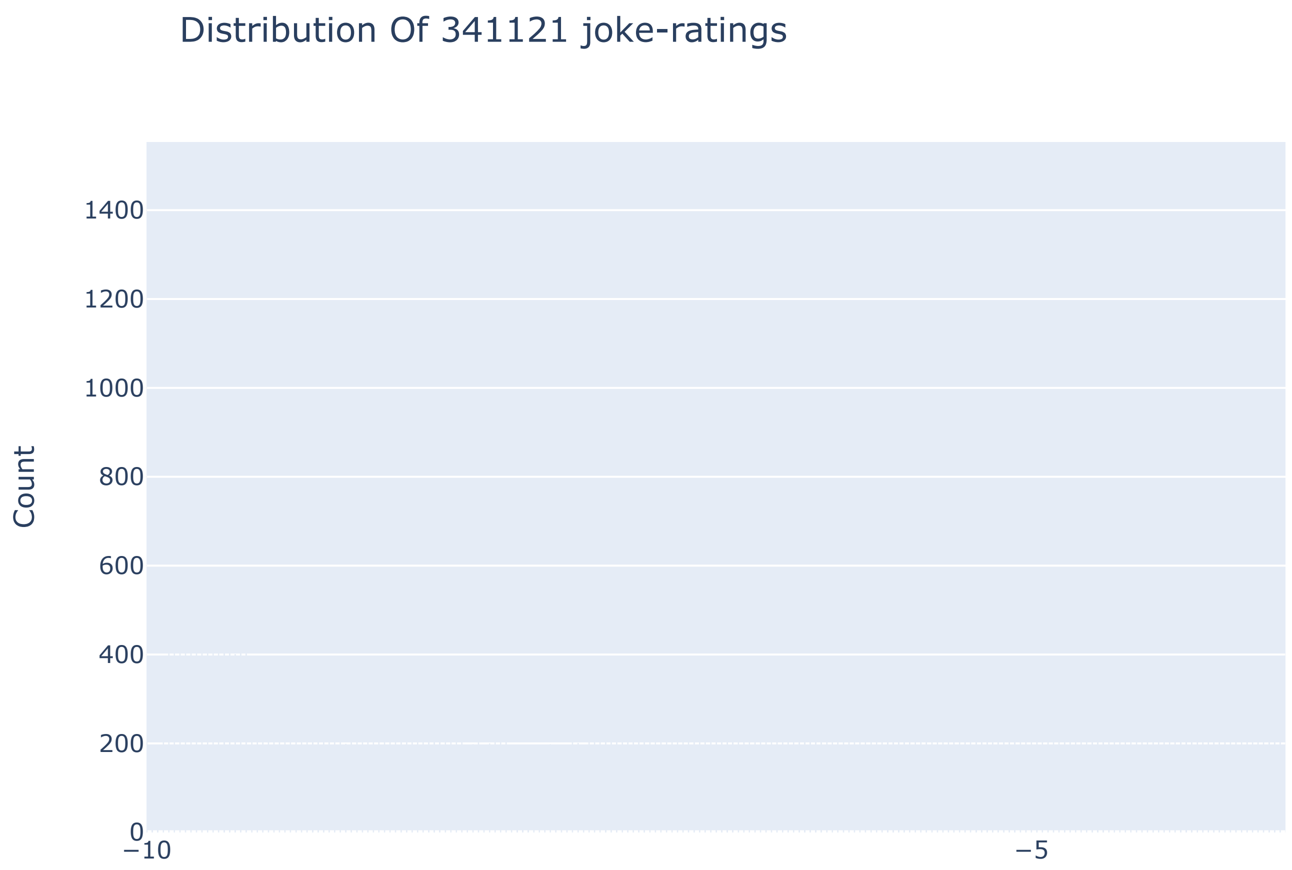
```
df.head(2)
```

	id	user_id	joke_id	Rating
0	31030_110	31030	110	2.750
1	16144_109	16144	109	5.094

```
df.groupby('joke_id')['Rating'].count().reset_index().sort_values('Rating',ascending=False
```

	joke_id	Rating
7	8	8689
3	4	8636
2	3	8600
4	5	8581
6	7	8556
1	2	8532
5	6	8525
8	9	8524
78	79	5339
103	104	5290

```
"""
Rating Distribution by User
"""
# Number of ratings per user
# Number of ratings per user

data=df.groupby('user_id')['Rating'].count().clip(upper=150)
trace=go.Histogram(x=data.values,
                   name='Ratings',
                   xbins=dict(start=0,
                              end=150,
                              size=2))
layout=dict(title ='Distribution Of Number of Ratings Per User (Clipped at 50)',
            xaxis=dict(title='ratings per user'),
            yaxis=dict(title='count'),
            bargap=0.2)

figure=go.Figure(data=[trace],layout=layout)
fig.show(renderer='colab')
```



```
df.groupby('user_id')['Rating'].count().reset_index().sort_values('Rating',ascending=False
```

	user_id	Rating
33500	34002	45
21159	21492	42
3061	3100	42
361	366	41
29914	30370	40

```
df.head()
```

	id	user_id	joke_id	Rating
0	31030_110	31030	110	2.750
1	16144_109	16144	109	5.094
2	23098_6	23098	6	-6.438
3	14273_86	14273	86	4.406
4	18419_134	18419	134	9.375

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 341121 entries, 0 to 341120
Data columns (total 4 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   id       341121 non-null  object 
 1   user_id  341121 non-null  int64  
 2   joke_id  341121 non-null  int64  
 3   Rating   341121 non-null  float64
dtypes: float64(1), int64(2), object(1)
memory usage: 10.4+ MB
```

```
df.describe()
```

	user_id	joke_id	Rating
count	341121.000000	341121.000000	341121.000000
mean	20700.840344	63.976601	1.752048
std	11808.463348	44.124420	5.232872
min	1.000000	1.000000	-10.000000
25%	10462.000000	22.000000	-1.750000
50%	21344.000000	62.000000	2.344000
75%	30771.000000	104.000000	5.781000
max	40863.000000	139.000000	10.000000

```
df.duplicated().sum()

     0


df.sort_values('Rating',ascending=False).head()


"""
Surprise Library
"""

"""
Building SVD Model
"""
svd=SVD(n_epochs=50,lr_all=0.01,reg_all=0.04,n_factors=250)
kf=KFold(n_splits=10,random_state=95)
for x,y in kf.split(data):
  svd.fit(trainingSet)
  pred=svd.test(y)
  rmse(pred,verbose=True)




     RMSE: 3.4889
     RMSE: 3.4813
     RMSE: 3.5047
     RMSE: 3.4886
     RMSE: 3.4929
     RMSE: 3.4728
     RMSE: 3.4762
     RMSE: 3.5059
     RMSE: 3.5116
     RMSE: 3.4920


trainsett=svd.trainset
print(svd.__class__.__name__)

     SVD


"""
Prediction on Test Data
"""

id=[]
user_id=[]
joke_id=[]
result=[]
result1=[]
for index,row in test_data.iterrows():
  print(index,row)
  id.append(str(row['id'])+'-'+str(row['joke_id'])+'-'+str(row['user_id']))
  result1.append(svd.predict(row['user_id'],row['joke_id']).est)
result=pd.DataFrame({'id':pd.Series(id),'rating':pd.Series(result1)})
result[['id','joke_id','user_id']] = result['id'].str.split('-',expand=True)
```

```
536679 id          334_46
user_id       334
joke_id        46
Name: 536679, dtype: object
536680 id         10782_43
user_id       10782
joke_id          43
Name: 536680, dtype: object
536681 id         24306_130
user_id        24306
joke_id          130
Name: 536681, dtype: object
536682 id         7015_79
user_id       7015
joke_id         79
Name: 536682, dtype: object
536683 id         8568_83
user_id       8568
joke_id         83
Name: 536683, dtype: object
536684 id         26708_85
user_id       26708
joke_id          85
Name: 536684, dtype: object
536685 id         25708_83
user_id       25708
joke_id          83
Name: 536685, dtype: object
536686 id         19207_84
user_id       19207
joke_id          84
Name: 536686, dtype: object
536687 id         13572_111
user_id        13572
joke_id          111
Name: 536687, dtype: object
536688 id         6158_58
user_id       6158
joke_id         58
Name: 536688, dtype: object
536689 id         39333_59
user_id       39333
joke_id          59
Name: 536689, dtype: object
536690 id         7403_98
user_id       7403
joke_id         98

Name: 536690, dtype: object
536691 id         10950_26
user_id       10950
joke_id          26
Name: 536691, dtype: object
536692 id         20161_104
user_id        20161
joke_id          104
Name: 536692, dtype: object
536693 id         3239_82
user_id       3239
joke_id         82
Name: 536693, dtype: object
```

```
result.head()
```

	id	rating	joke_id	user_id
0	6194_11	2.909905	11	6194
1	19356_3	0.000000	3	19356
2	23426_79	2.869830	79	23426
3	40030_3	0.000000	3	40030
4	19806_115	5.000000	115	19806

```
endResult = result.drop(['user_id','joke_id'],axis=1)
endResult.columns = ['id','Rating']
```

```
endResult.to_csv("brahm_jokes_submission1.csv",index=False)
```