**Project on Machine Learning**

Name of the project:

**Building user-based recommendation model for Amazon.**

**Done by: Mrinmay Saha**

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

**Source Code with graphs an insights:**

#Importing Libaries

import pandas as pd

import numpy as np

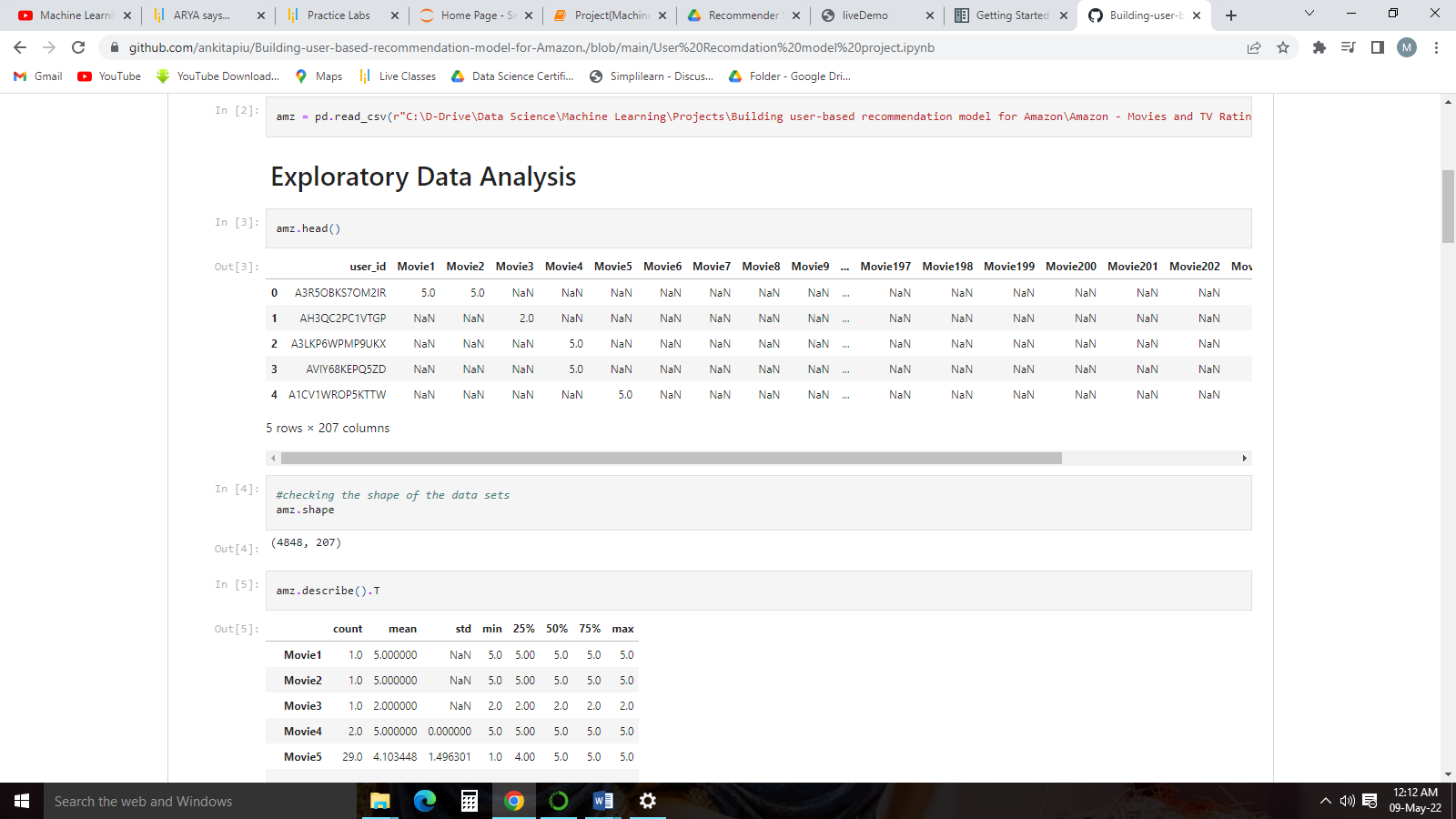
import matplotlib.pyplot as plt

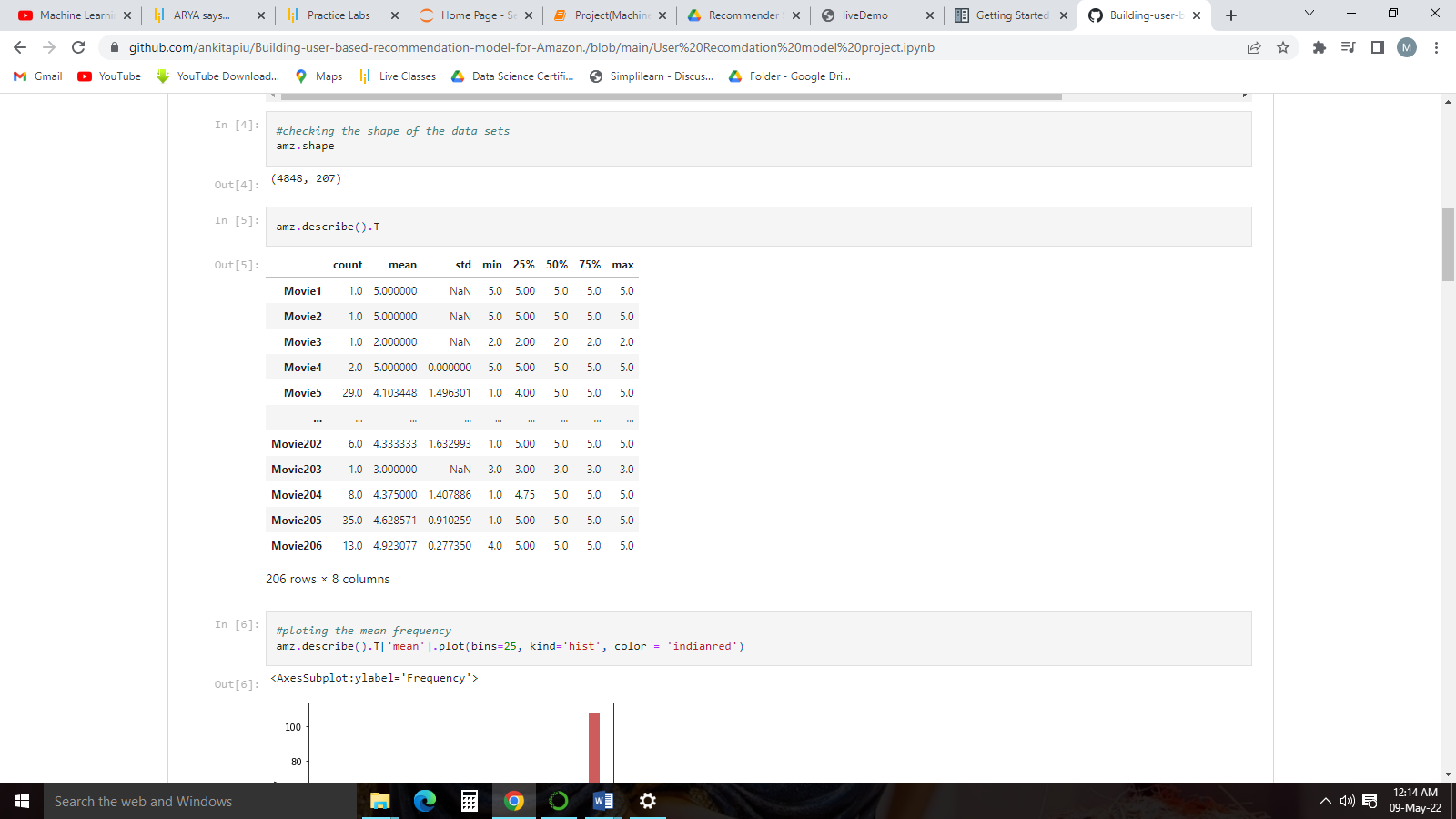
%matplotlib inline

#Loading Data

df=pd.read\_csv('Amazon\_-\_Movies\_and\_TV\_Ratings[1].csv')

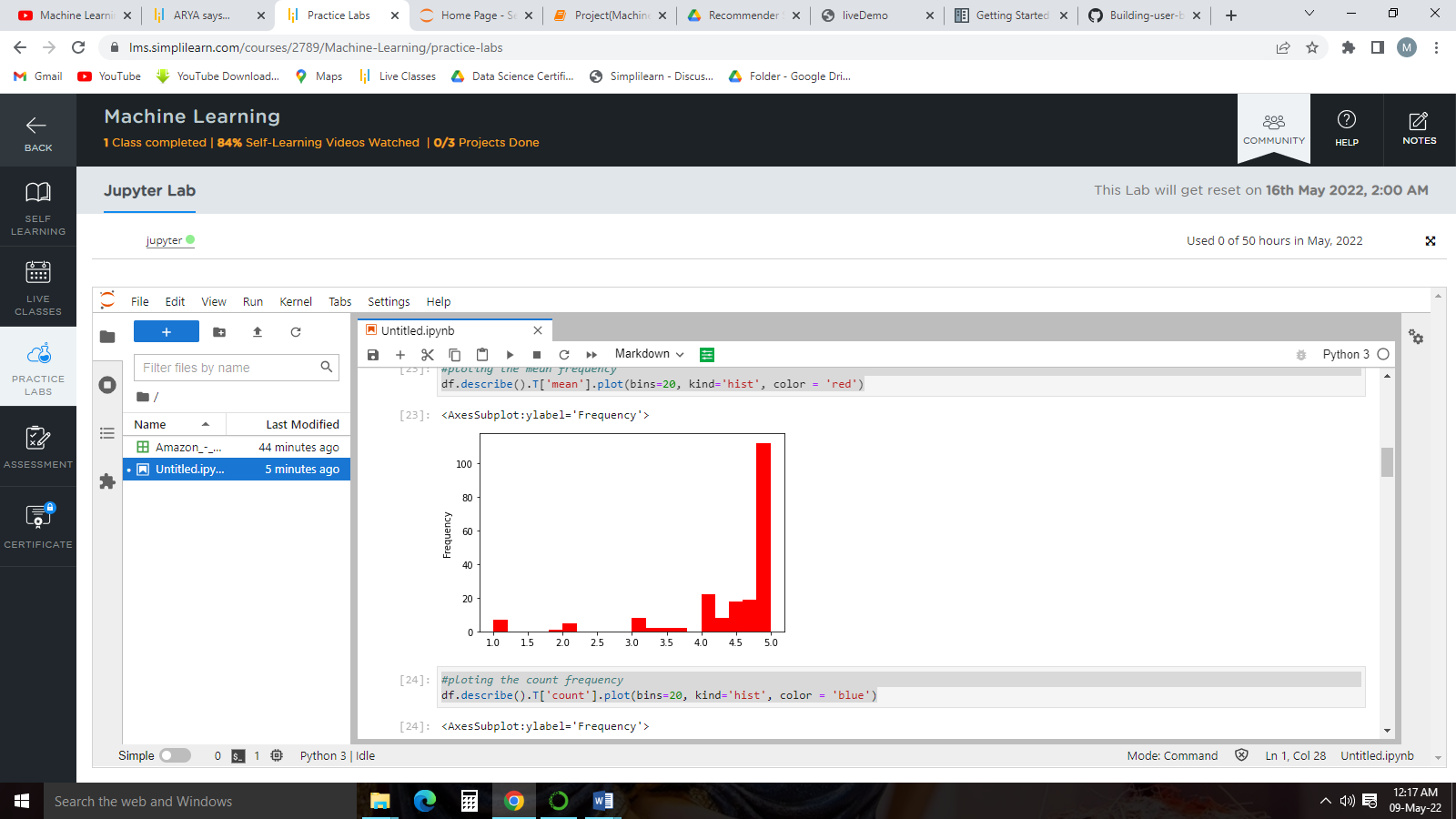
#checking first 5 data points

df.head()

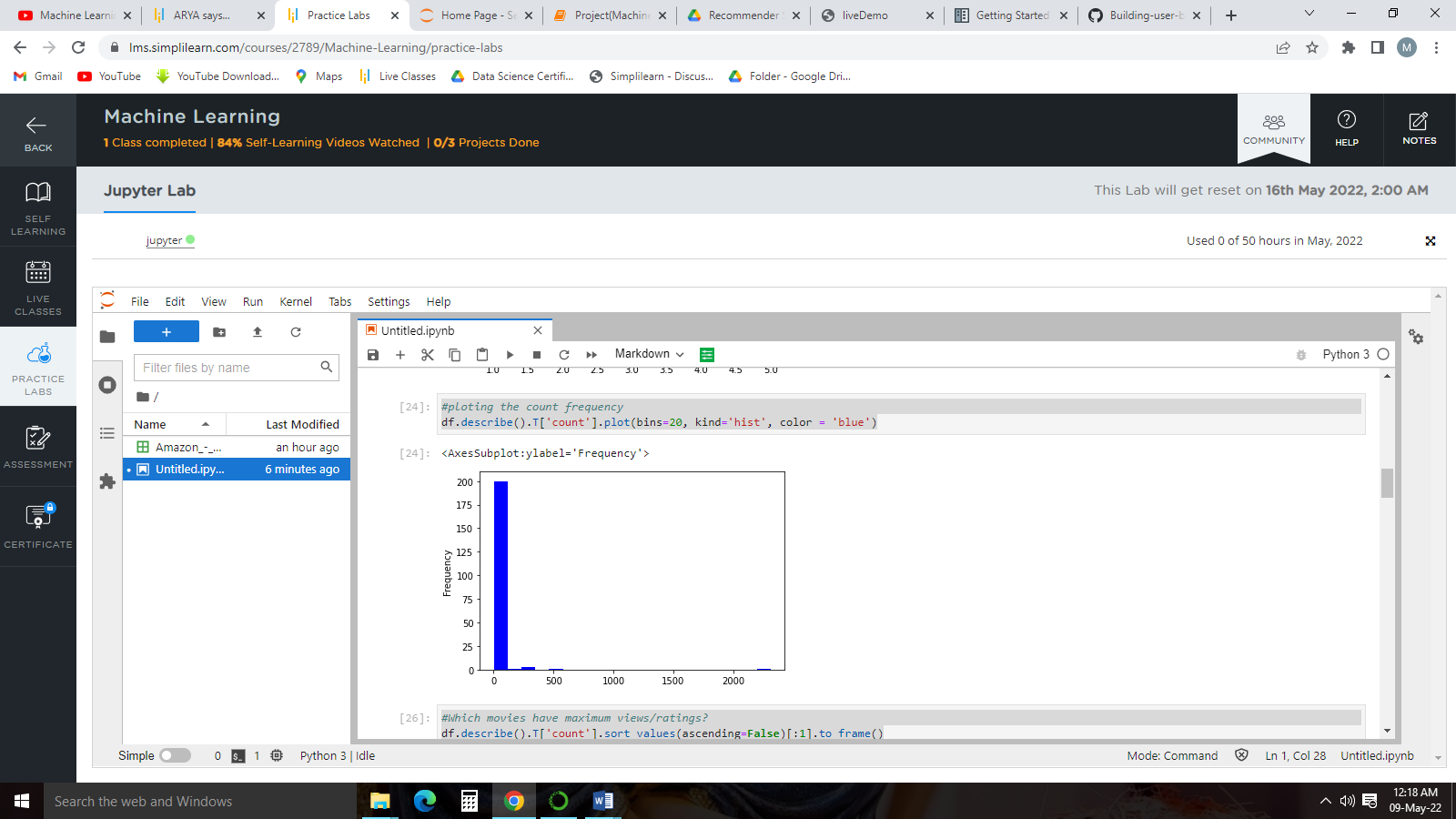
df.describe().T

#ploting the mean frequency

df.describe().T['mean'].plot(bins=20, kind='hist', color = 'red')

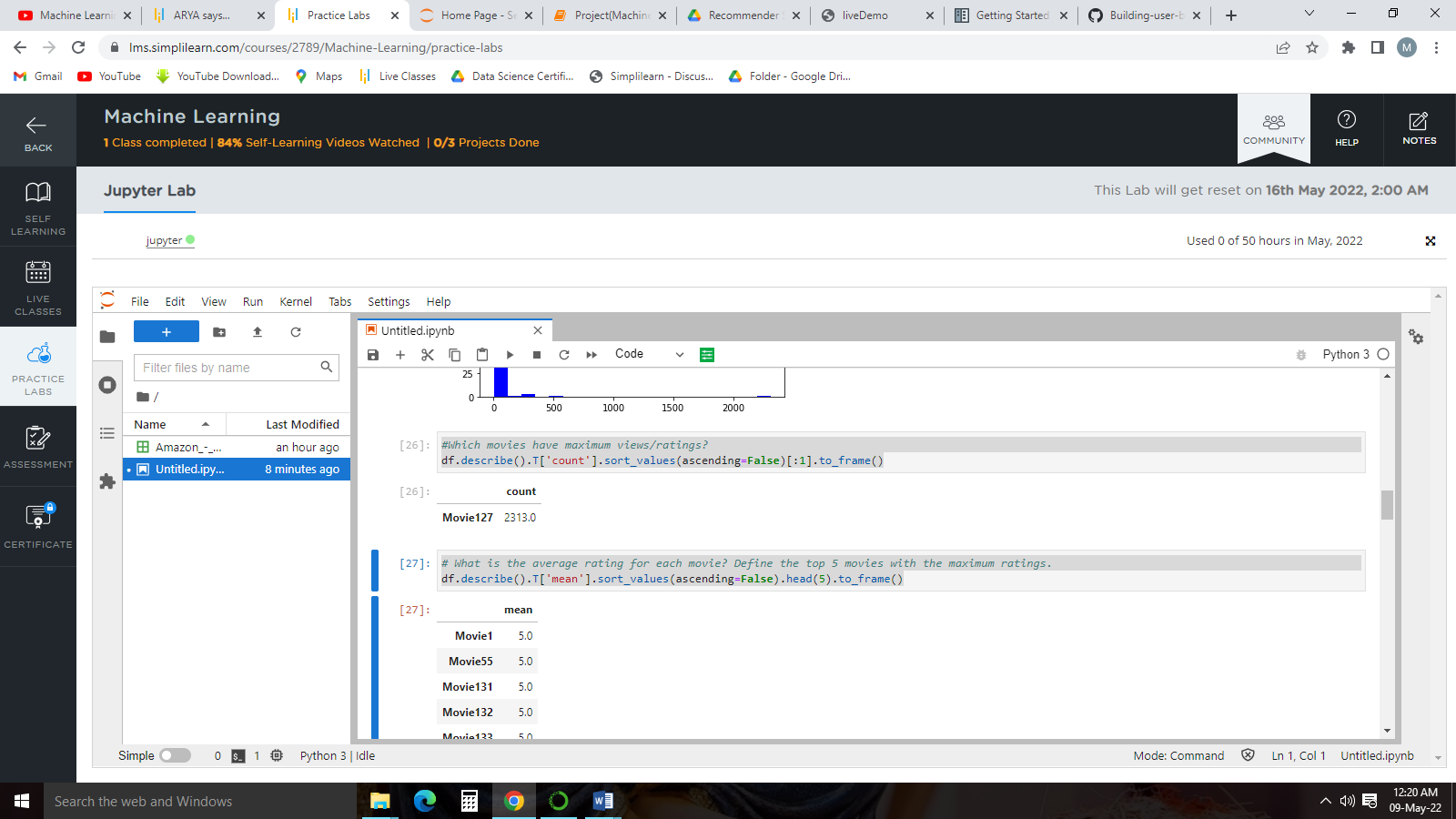


#ploting the count frequency

df.describe().T['count'].plot(bins=20, kind='hist', color = 'blue')

#Which movies have maximum views/ratings?

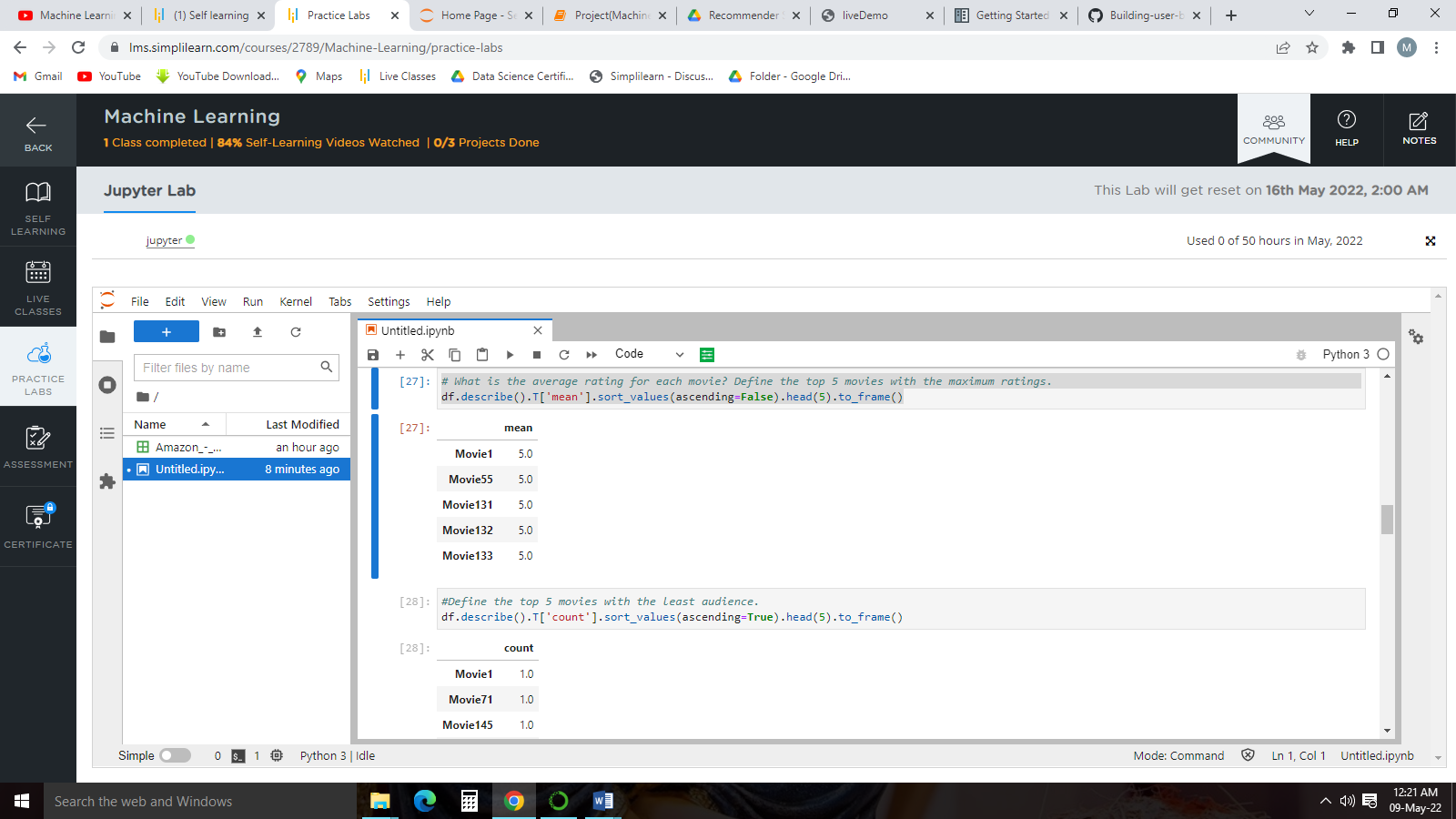
df.describe().T['count'].sort\_values(ascending=False)[:1].to\_frame()

Output:

From the above analysis we can conclude that Movie127 has the highest views/ratings.

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

df.describe().T['mean'].sort\_values(ascending=False).head(5).to\_frame()

Ouput:

From the above analysis we can conclude that Movie1,Movie55,Movie131,Movie132,Movie133 are the movies with the highest ratings.

**User Based Model Building**

*#importing libiraies for model building*

**from** surprise **import** Reader

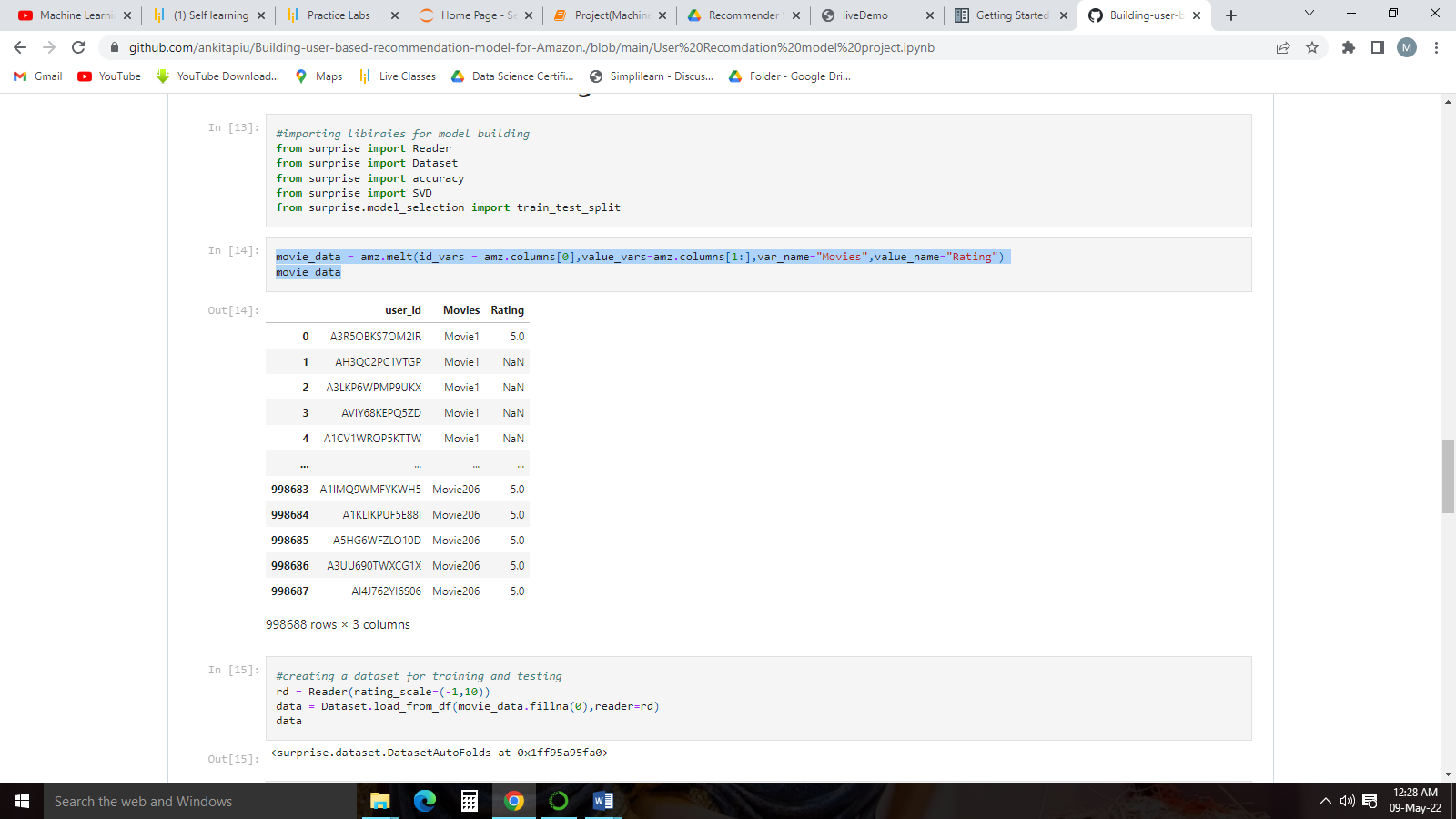
**from** surprise **import** Dataset

**from** surprise **import** accuracy

**from** surprise **import** SVD

**from** surprise.model\_selection **import** train\_test\_split

df1=df.melt(id\_vars=df.columns[0],value\_vars=df.columns[1:],var\_name="Movies",value\_name="Rating").fillna(0)

df1

reader = Reader(rating\_scale=(-1,10))

data = Dataset.load\_from\_df(df1.fillna(0), reader=reader)

trainset, testset = train\_test\_split(data, test\_size=0.25)

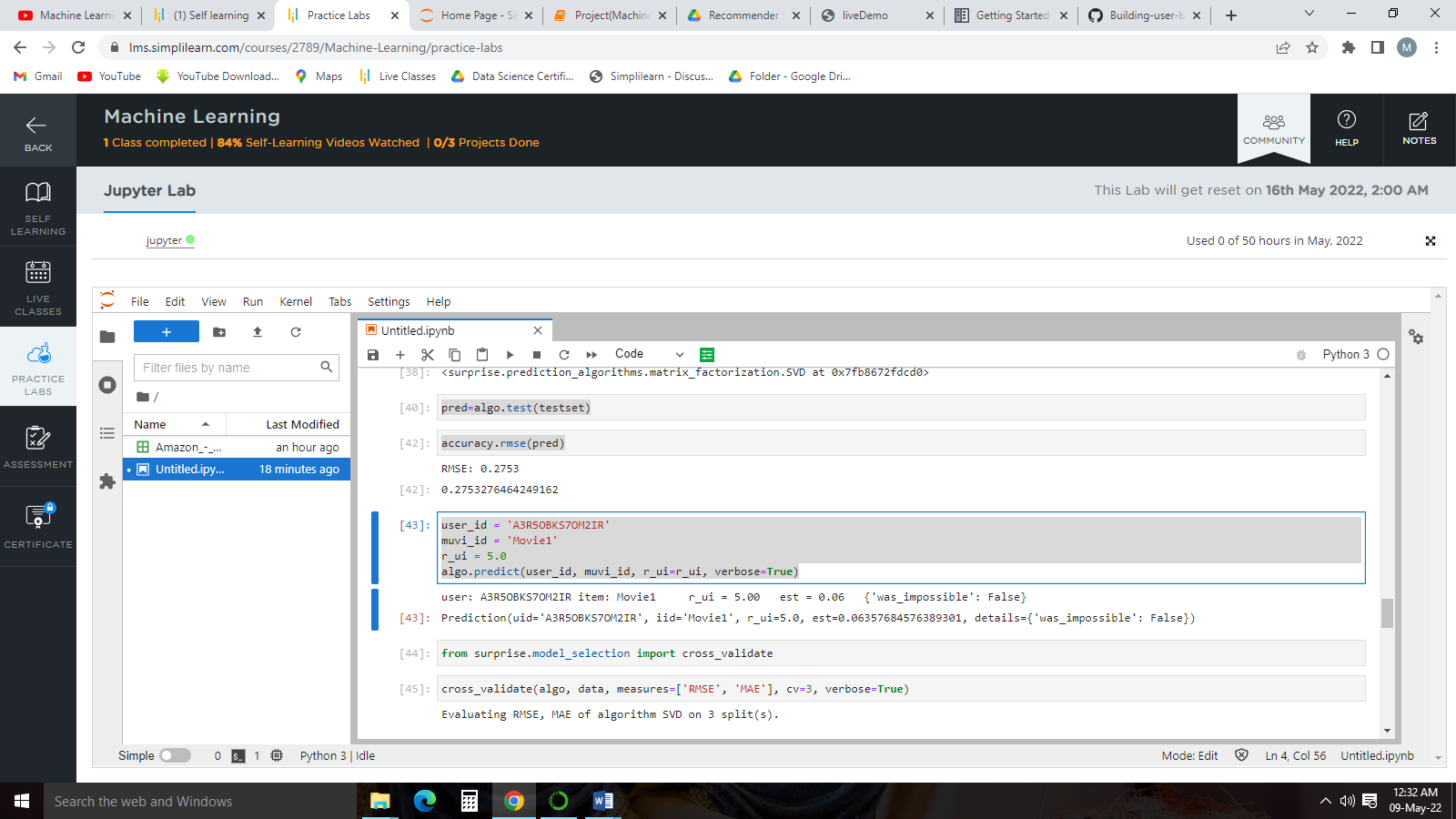
algo=SVD()

algo.fit(trainset)

pred=algo.test(testset)

accuracy.rmse(pred)

Ouput:



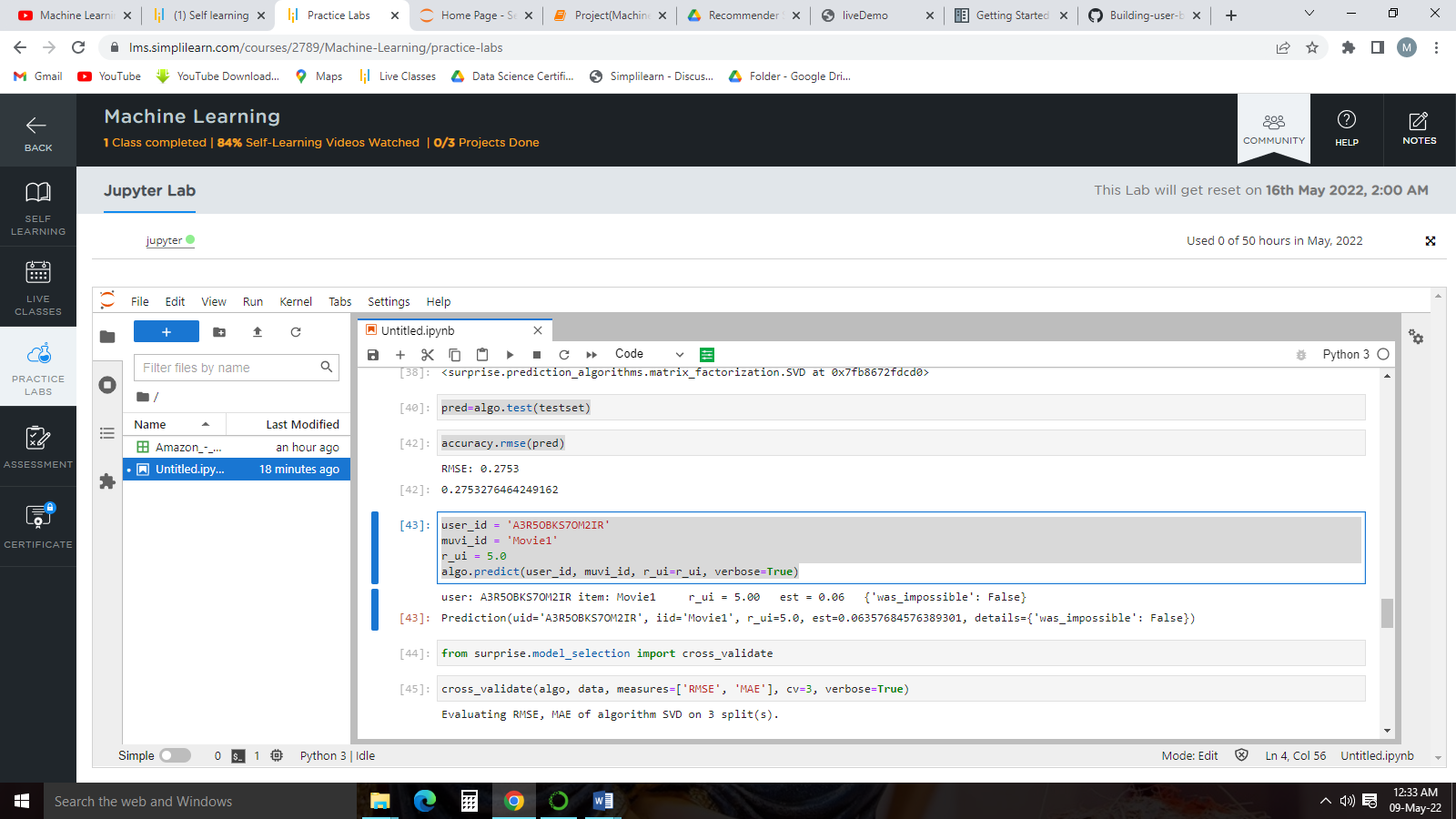
#Doing Predication

user\_id = 'A3R5OBKS7OM2IR'

muvi\_id = 'Movie1'

r\_ui = 5.0

algo.predict(user\_id, muvi\_id, r\_ui=r\_ui, verbose=True)

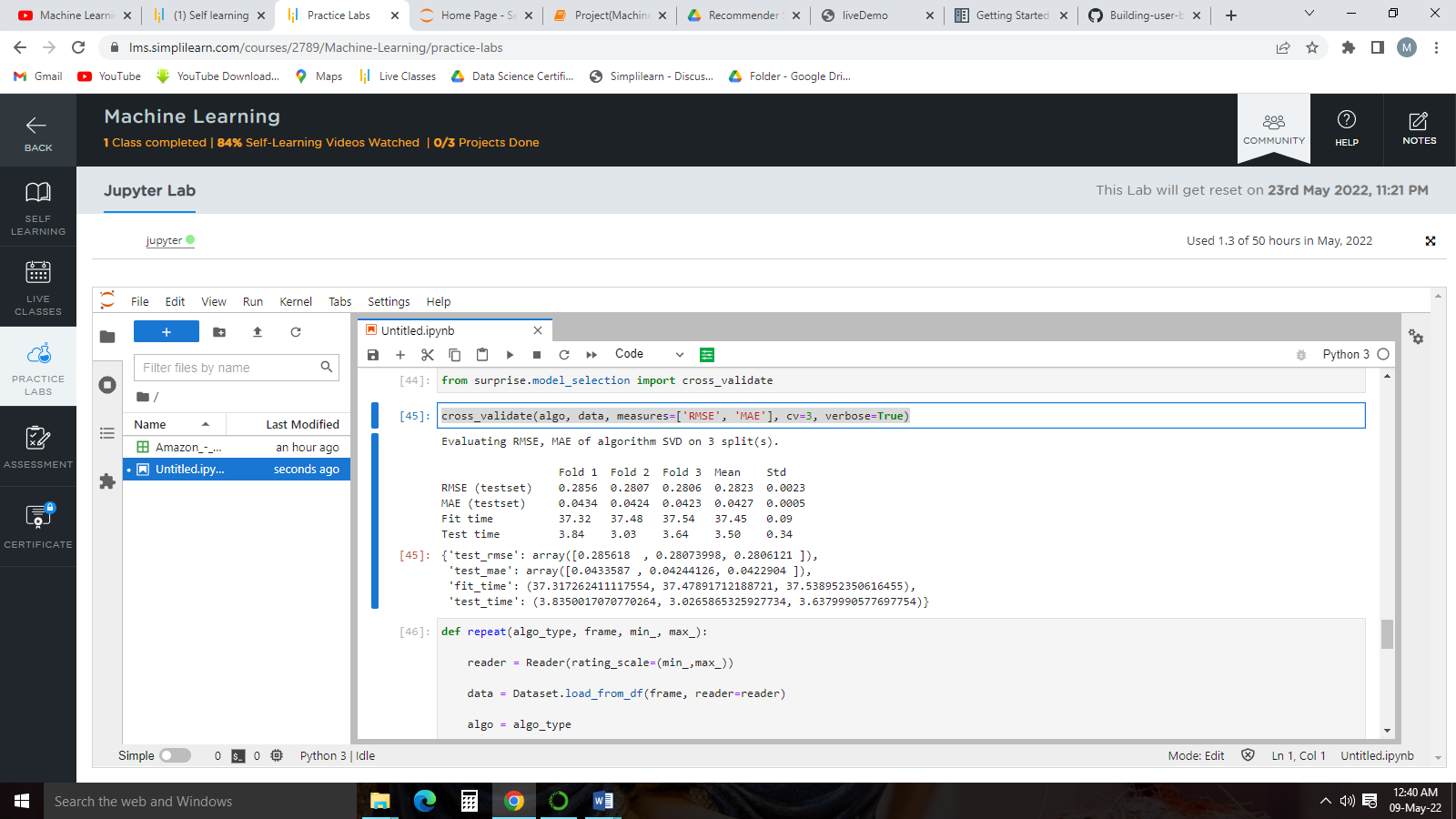
Output:

From the above prediction we can see that the predication made by the model is not accurate and very far from the original value.

#using cross-validation to tune the model

cross\_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)

Output:



#Checking the model with different values in the place of null values

def repeat(algo\_type, frame, min\_, max\_)

reader = Reader(rating\_scale=(min\_,max\_))

data = Dataset.load\_from\_df(frame, reader=reader)

algo = algo\_type

print(cross\_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True))

print("#"\*10)

user\_id = 'A3R5OBKS7OM2IR'

muvi\_id = 'Movie1'

r\_ui = 5.0

print(algo.predict(user\_id, muvi\_id, r\_ui=r\_ui, verbose=True))

print("#"\*10)

print()

df = df.iloc[:2500, :50]

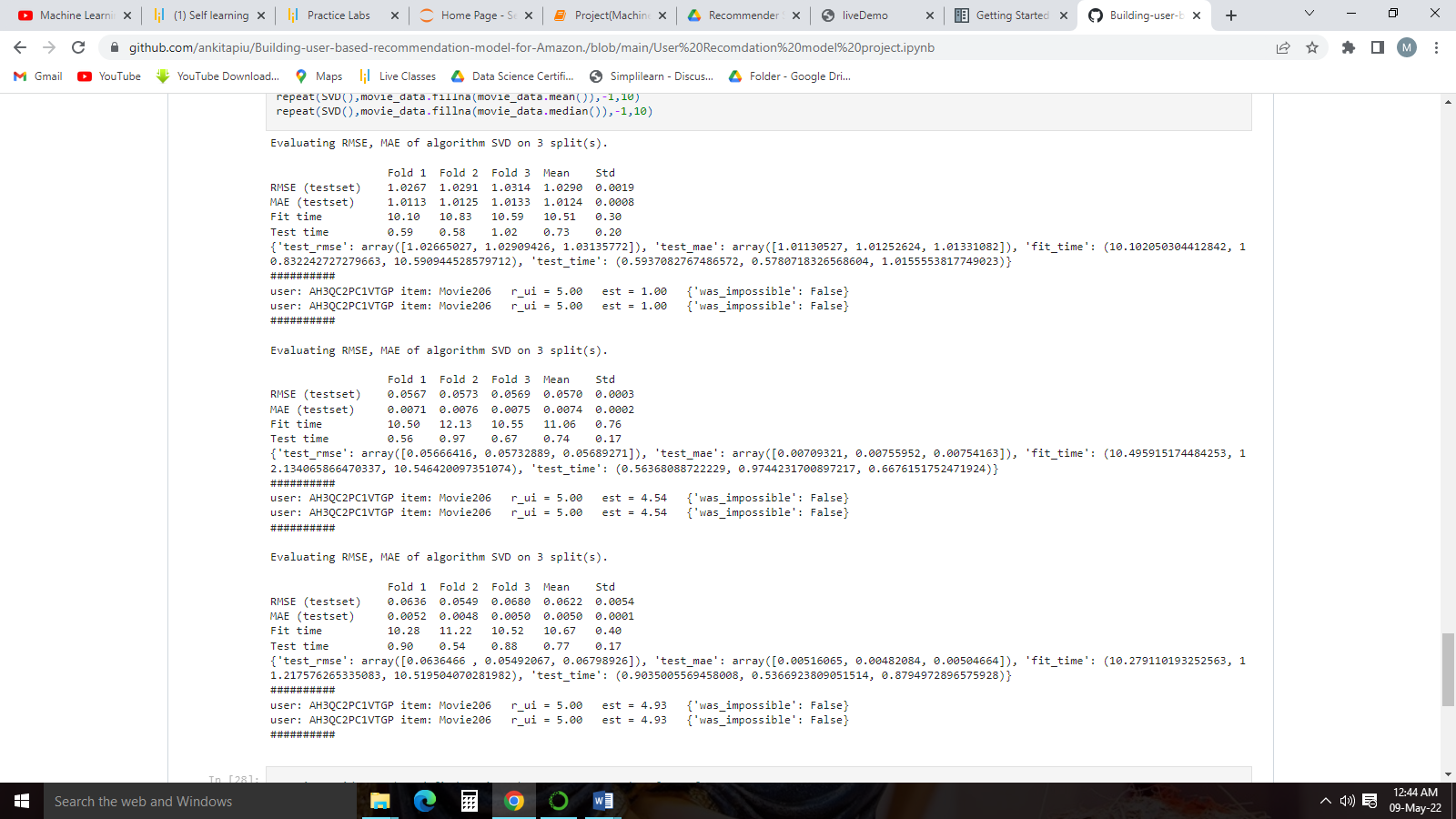
df1 = df.melt(id\_vars = df.columns[0], value\_vars= df.columns[1:], var\_name="movie\_name", value\_name="rating")

repeat(SVD(), df1.fillna(0), -1, 10)

repeat(SVD(), df1.fillna(df1.mean()), -1, 10)

repeat(SVD(), df1.fillna(df1.median()), -1, 10)

Output:



From the above analysis we can conclude that by filling the null values with mean or median the prediction of the model is improved drastically.

#trying grid search and find optimum hyperparameter value for n\_factors

from surprise.model\_selection import GridSearchCV

param\_grid = {'n\_epochs':[20,30],

'lr\_all':[0.005,0.001],

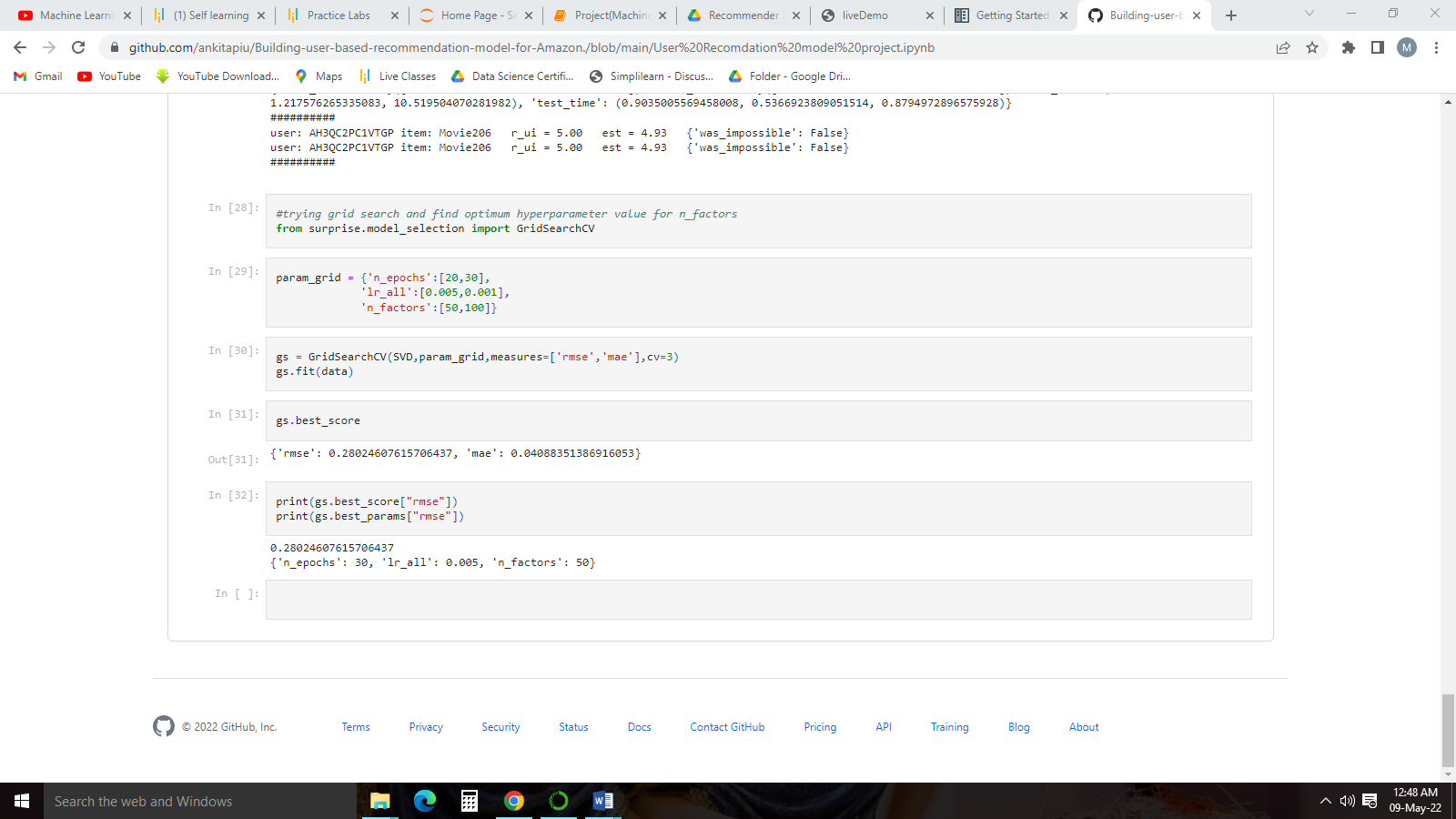
'n\_factors':[50,100]}

gs = GridSearchCV(SVD,param\_grid,measures=['rmse','mae'],cv=3)

gs.fit(data)

gs.best\_score()

Output:



print(gs.best\_score["rmse"])

print(gs.best\_params["rmse"])

