

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

June , 2022

**J Component report**

**Programme : MCA**

**Course Title : Data mining And Business Intelligence**

**Course Code : ITA5007**

**Slot : A2+TA2**

**Title: Business Recommendation System**

**Team Members: Aravind V R / 21MCA1023**

**Pavan Kumar C R/ 21MCA1066**

**Faculty: Dr Punitha K**  **Sign :**

**Date :**

**DECLARATION**

We, Aravind, Pavan Kumar hereby declare that the thesis entitled “**Business Recommendation System**” submitted by us, for the completion of the course, Data mining and Business Intelligence (ITA5007) is a record of Bonafide work carried out by us under the supervision of Dr Punitha K, our course instructor. We further declare that the work reported in this document has not been submitted and will not be submitted, either in part or in full, for any other courses in this institute or any other institute or university.

Place: Chennai Signature of the Candidates:

Date: 05.06.2022

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

CERTIFICATE

This is to certify that the report entitled **“BUSINESS RECOMMENDATION SYSTEM”** is prepared and submitted by **Aravind V R(21MCA1023), Pavan Kumar C R(21MCA1066)** to Vellore Institute of Technology, Chennai, in partial fulfilments of the requirement for the course, **Data Mining and Business Intelligence (ITA5007)**, is a bonafide record carried out under our guidance. The project fulfills the requirements as per the regulations of this University and in our opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for any other course and the same is certified.

Name: Dr. Punitha K

Date: 05.06.2022

**ABSTRACT**

Business recommendation is becoming one of the most needed services to the world. Business providers around the world is trying to get their customers as more as possible and there is a fair competition in the field as well. Here in this project, we were given a data set of 2,038,130 known ratings by Yelp users for various businesses and asked to build a recommender system to predict unknown ratings for a list of 158,024 user and business combinations, using algorithms and implementations of our choosing. We have chosen 6 algorithms to predict the ratings for each business entity out of 5. This allows us to understand which business of the same stream has the better ratings and out of these ratings we can easily select the one we need. Business recommendation has a wide scope in all the fields where there is selling and buying involved. The vast size of the training dataset hampered the process of identifying an appropriate strategy to predict test ratings. Most recommendation system libraries and packages are unable to handle this volume of data, resulting in runtime memory issues. The primary reason behind this is due to the fact that most recommendation algorithms require a dense matrix filled with data. Even if the initial training data fits in memory, with all the ratings (both known and anticipated), The matrix with all ratings generated at the end of the algorithm does not. We started by attempting to implement matrix factorization in Python using a tutorial. When this method was run, it resulted in a memory error and took a lengthy time to complete. After that, we tried utilising the scikit-learn Python package and its Nonnegative Matrix Factorization class, but the memory problem persisted. We used a sparse matrix from the SciPy Python module to try both of these approaches, but they didn't work. Then we came upon Surprise, a Python module designed to construct recommendation systems. Single Value Decomposition (SVD), Matrix Factorization, and Co-Clustering are among the algorithms included in Surprise.

**Keywords**: Data Mining, Business Prediction, Predicting Business Ratings, Classification

**ACKNOWLEDGEMENT**

In successfully completing this project, many people have helped me. I would like to thank all those who are related to this project.

I would like to thank my teacher **(Dr. Punitha K)** who gave me this opportunity to work on this project and also for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and implementation of the project. I got to learn a lot from this project about how to search for the best datasets for the projects, how to make your project more effective, which will be very helpful in solving some real world problems and many other things. I would also like to thank our school HOD ma’am **(Dr Sivagami M).**

At last, I would like to extend my heartfelt thanks to my parents because without their help this project would not have been successful. Finally, I would like to thank my dear friends and my fellow students for many helpful discussions and good ideas along the way who have been with me all the time.

## Table of Contents

**CONTENTS…………………………………………III**

**LIST OF FIGURES…………………………………IV**

**LIST OF ACRONYMS……………………………...V**

1. **INTRODUCTION**
   1. Introduction…………………………………………..1
   2. Relevance of Project…………………………………3
   3. Problem statement…………………………………...4
   4. Dataset……………………………………………….4
   5. Objectives……………………………………………4
   6. Scope of project……………………………………...5
   7. Literature Review……………………………………5
2. **PACKAGE USED**

* 1. Packages…………………………………………….10

**3. ALGORITHMS USED**

* 1. 3.1 Singular Value Decomposition …………………10

3.1.1 Implementation of Singular Value Decomposition

* 1. Singular Value Decomposition Plus Plus………..14

3.2.1 Implementation of Singular Value Decomposition Plus Plus

* 1. Probabilistic Matrix Factorization…………………17
     1. Implementation of Probabilistic Matrix Factorization

* 1. Non-negative Matrix Factorization ……………….20
     1. Implementation of Non-negative Matrix Factorization
  2. Alternating Least Squares…………………………24
     1. Implementation of Alternating Least Squares
  3. Co-Clustering……………………………………..27
     1. Implementation of Co-Clustering

1. Conclusion……………………………………………...31

5.References…………………………………………..32

**List Of Figures**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **Figures Name** | **Page No.** |
| 1 | Fig1. Output of SVD implementation in pycharm | 12 |
| 2 | Fig.2 Result csv file of SVD implementation | 13 |
| 3 | Fig.3 Output of SVD++ implementation in pycharm | 16 |
| 4 | Fig.4 Result csv file of SVD++ implementation | 17 |
| 5 | Fig.5 Output of PMF implementation in pycharm | 19 |
| 6 | Fig.6 Result csv file of PMF implementation | 19 |
| 7 | Fig.7 Output of NMF implementation in pycharm | 22 |
| 8 | Fig.8 Result csv file of NMF implementation | 23 |
| 9 | Fig.9 Output of ALS implementation in pycharm | 26 |
| 10 | Fig.10 Result csv file of ALS implementation | 26 |
| 11 | Fig.11 Output of CO-CLUSTERING implementation in pycharm | 29 |
| 12 | Fig.12 Result csv file of CO-CLUSTERING implementation | 30 |

**LIST OF ABBREVIATIONS**

**Abbreviations Full Forms**

**SVD Singular Value Decomposition**

**SVD++ Singular Value Decomposition Plus Plus**

**PMF Probabilistic Matrix Factorization**

**NMF Non-negative Matrix Factorization**

**ALS Alternating Least Squares**

**1. INTRODUCTION**

Business is the wheel of the world. We will come across many businesses in a single day of our lives. It is so important for an individual as well for an organization to fall for the right business. We thought of using user ratings to understand which ones are better among a set of businesses. We use ratings from Yelp for this which we got through Kaggle. We decided that using numerous algorithms to construct an ensemble forecast was the best course of action after identifying and testing these diverse techniques. We chose the algorithms that performed well on the huge training set and ran them many times, each with slightly varied hyperparameter values, to generate various prediction files. The average of all the predictions was then calculated. Ensemble predictions were chosen because they have been found to create a set of predictions with lower variance than any of the originals, resulting in superior results.

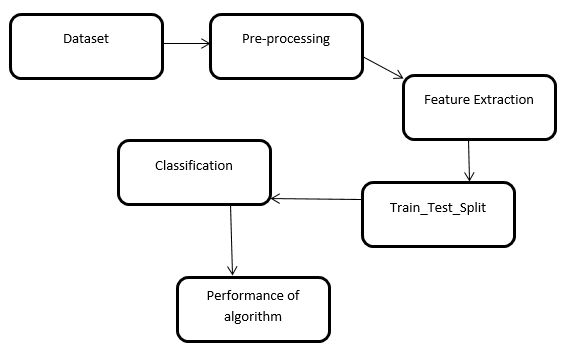


Fig 1.1: Data Flow Diagram

**Steps involved in this Proposed System are**

1. **Pre-process-**Data preprocessing is a step in the data mining and data analysis process that takes raw data and transforms it into a format that can be understood and analyzed by computers and machine learning.
2. **Feature Extraction-**Feature Extraction aims to reduce the number of features in a dataset by creating new features from the existing ones (and then discarding the original features). These new reduced set of features should then be able to summarize most of the information contained in the original set of features.
3. **Train\_Test\_split-**The train-test split is a technique for evaluating the performance of a machinelearning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets.
4. **Classification**- is a data mining function that assigns items in a collection to targetcategories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.

The pandemic has presented us with numerous obstacles. Many of us were forced to alter our business practices. Nothing completely novel, in my opinion, was invented. We were, however, compelled to use, update, and adapt current technologies and business processes at a faster rate than we'd ever had to before. Furthermore, leadership acknowledged that in order to stay up with the demanding business environment, long-term strategic plans had to be set aside and substituted with spur-of-the-moment decisions.

Customers are becoming more aware of what constitutes excellent customer service and CX with each passing year. They are being trained by firms like Amazon, Target, Walmart, and other well-known brands that excel at providing excellent customer service. Your clients no longer compare you to your competition, but to the best service they've ever received from any brand, regardless of the type of business you're in or how big or little it is.

**1.2 Relevance of the Project**

Data mining and machine learning techniques can provide an efficient prevention approach where data is associated with business ratings. We now have more data than ever before. Customers generate many types of data every second as a result of their various interactions. For a while, having large data was a fascinating advantage for enterprises, but the knowledge buried behind it and leveraging it to make future decisions have become serious concerns.

Predictive Analytics is a data driven technology and statistical techniques which examine large data sets to discover patterns, uncover new information and predict failure points and outcome for future.  Big data can be a huge benefit to any organization when used with predictive analytics which enables customers of businesses to make really quick strategic decisions. It is basically a road map to choose a better business.

The relevance of this study is helpful for all the customers other than business providers and it helps the customers in choosing the best out of a number of business proposals they are having.

**1.3 Problem Statement**

To predict the business ratings for a number of businesses and that helps the customers to select to go with one of them.

**1.4 Dataset**

We were given a data collection of 2,038,130 known Yelp user ratings for various companies and requested to create a recommender system to forecast unknown ratings for a list of 158,024 user and business combinations using techniques and implementations of our choosing for this project.

**1.5 Objectives**

● To predict business ratings

● It helps to choose the business from customer side

● It business data taken from Kaggle site, we train the models using these ratings data and then without the help of the ratings data the test process is done and then the results are generated

● It uses 6 algorithms as a whole

● It uses SVD (Singular Value Decomposition), SVD plus, PMF, NMF, Co-Clustering and ALS (Alternating Least Squares)

● To create a user awareness for the businesses that someone going to deal with and to help him to find the best one out of a number of options someone is having

**1.6 Scope of the Project**

The goal of this project is to predict forest fires using machine learning algorithms. This project uses Random Forest to provide better accuracy. It also uses sub-decision trees for high speed computation. The dataset provided by IMD (Indian Meteorological Department) will be divided into training data and test data. The training data is fed into the model for supervised learning. This model should now predict forest fires based on the test data given as input. A user interface is developed for testing different test data to predict the occurrence of forest fires.

* 1. **Literature Review**

Literature Review of the papers dealing with business prediction is done here. We have used 6 papers for this as well.

1. A hybrid recommendation algorithm–based intelligent business recommendation system:

Fan Yang

To address the present challenges with e–commerce recommendation systems, such as low accuracy, inflexibility, and lack of personalization, a solution based on a hybrid recommendation algorithm is offered, with the goal of creating a personalised e–commerce recommendation system. To compensate for the lack of a single suggestion.

We use the hybrid recommendation method, which combines three algorithms: depending on content item-based collaborative filtering recommendation algorithm, and demography-based recommendation algorithm for making recommendations To broaden the scope of our recommendations, we use categorization techniques to harvest the history data of objects and users, as well as clustering After that, we conduct a performance appraisal and a performance review.

We propose the design and implementation of a hybrid recommendation algorithm-based intelligent business recommendation system that serves as a model for e–commerce recommendation systems. The hybrid recommendation algorithm lies at the heart of the entire architecture of this recommendation system. The recommendation module and the recommendation dimensions update module are included in this system. The optimization and implementation of content-based recommendation algorithms, item-based collaborative filtering recommendation algorithms, and demography-based recommendation algorithms are covered in the recommendation module. We introduce mining recommendation dimensions from user data and commodities data using classification and clustering methods in the recommendation dimensions update module. In the end, crucial measurements show that this intelligent recommendation system is more efficient than other single recommendation algorithms.

1. A Framework of Hybrid Recommendation System for Government-to-Business Personalized e-Services :

Qusai Shambour, Jie Lu

One of the issues that e-governments face is how to provide businesses with services and information that are tailored to their needs rather than a homogenous mass of people, to their needs information. One way to accomplish this is to use recommendation systems, create and develop personalised government eservices. In order to accomplish this, a personalised hybrid recommender is presented in this paper a framework for handling customised data

In particular, advice in G2B e-services E-services for matching business partners. The suggested hybrid trust-based multi-criteria framework is used in this framework as a model for making recommendations that incorporates the concepts of With the multi-criteria CF, trust-based filtering is possible. The proposed approach can be utilised to save time, money, and effort. Businesses that enter overseas markets face a number of risks. As a result, the quality of G2B e-services will improve.

This study provides a tailored hybrid RS framework to assist exporters looking for G2B e-services partners. The framework uses a hybrid trust-based multi-criteria recommendation model that combines trust-based filtering and multi-criteria CF approaches. This model will be able to manage potential recommendation roadblocks, such as sparsity and cold start user issues, and will thus outperform traditional CF recommendation algorithms. Furthermore, the proposed system framework has the ability to work within a wide range of related G2B e-services, such as international trade exhibition and export market recommendations, as well as other related multi-criteria based recommendation applications in various domains (e.g. e-commerce and elearning). Experimenting with the proposed hybrid trust-based multi-criteria recommendation model will be the subject of future research.

1. recommendation system development for fashion retail e-commerce :

Hyunwoo Hwangbo, Yang Sok Kim ⇑ , Kyung Jin Cha

This paper describes a real-world collaborative filtering recommendation system that was implemented in a large Korean fashion firm that sells fashion products both online and in physical stores. The following aspects distinguish the company's recommendation environment: For starters, the company's online and physical stores both sell the same items. Second, most fashion items are only available for a limited time.As a result, clients' preferences vary depending on the season. Finally, customers frequently make purchases to replace previously favoured items or to add to the collection of items already purchased they offer K-RecSys, a new system that extends the traditional item-based collaborative filtering algorithm by taking into account the aforesaid domain properties. K-RecSys integrates online product click data with customer feedback.

Data on offline goods sales has been weighted to suit customers' online and offline preferences.

They suggested a new approach of recommending fashion products to clients in this study by extending the previous collaborative filtering method to include fashion product features. First, we took into account the fact that fashion items are sold.Preferences for fashion products can be found both online and offline.Using online click data and offline purchase data, both online and offline to make suggestions based on data Second, there is the issue of client preference.Fashion products have a tendency to depreciate over time. To We've proposed a recommendation approach that takes this into account a preference decay function that reduces the intensity of preference over time

1. Designing Business Processes with a Recommendation-Based Editor:

Agnes Koschmider

Building "excellent" process models frequently takes more than a basic understanding of modelling language syntax. To use a modelling language in practise, you'll need a lot of modelling experience. Users with no modelling experience have low productivity, and consequently the quality of the modelling result may be unsatisfactory if appropriate modelling tool assistance is lacking. We introduce a recommendation-based editor for process modelling in this chapter, which can assist in solving this problem by lowering the requirement for the user to master the modelling notation and, as a result, directing her attention on the model content. Early reviews show that our strategy, which goes beyond traditional modelling assistance for business processes, is beneficial.

In contrast to traditional control flow-driven workflows, this chapter provides a broad framework for the design and implementation of resource-driven workflows. Resources serve as an organising factor in a resource-driven workflow.

Rather than an explicit, pre-set control flow, the actions in a process are completed in the correct order based on the availability of resources such as data documents, human or system roles, physical space, and equipment. We propose that when multiple, dynamic, and perhaps competing resources are involved, it is impossible to design a business process just on the basis of the control flow; rather, it develops from the interaction of resources that are required for each task in the process. 438 A. Kumar and J. Wang

We demonstrated how resource-driven workflows are particularly beneficial.

2.**PACKAGES USED**

We have implemented our algorithms with the help of Python. We have made use of in-built python libraries and packages to implement our classification algorithms. We have made use of the following libraries and packages:

NumPy

Pandas

Scikit-learn

Surprise

1. **ALGORITHMS USED**

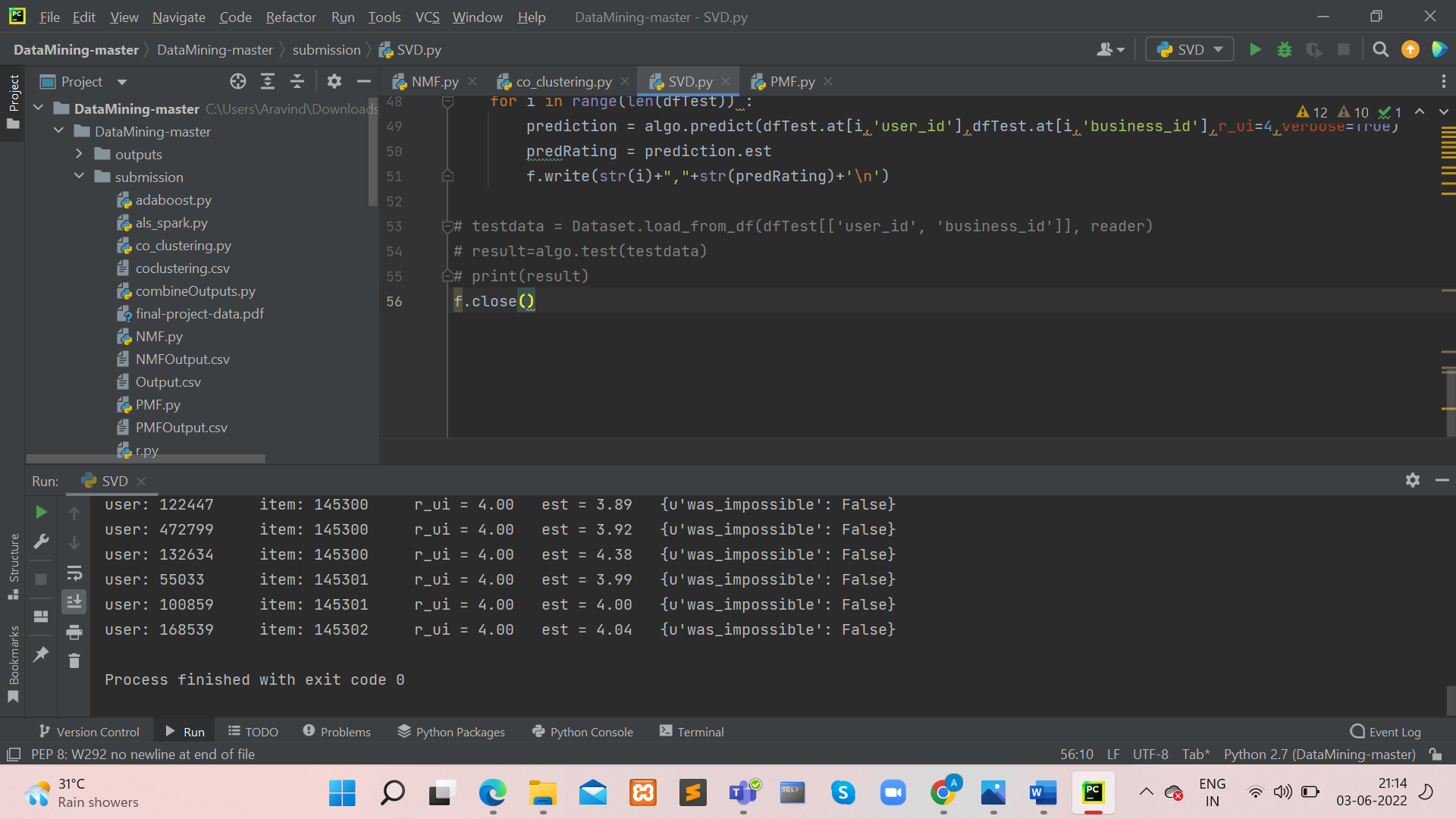
**3.1** **Singular Value Decomposition (SVD)**

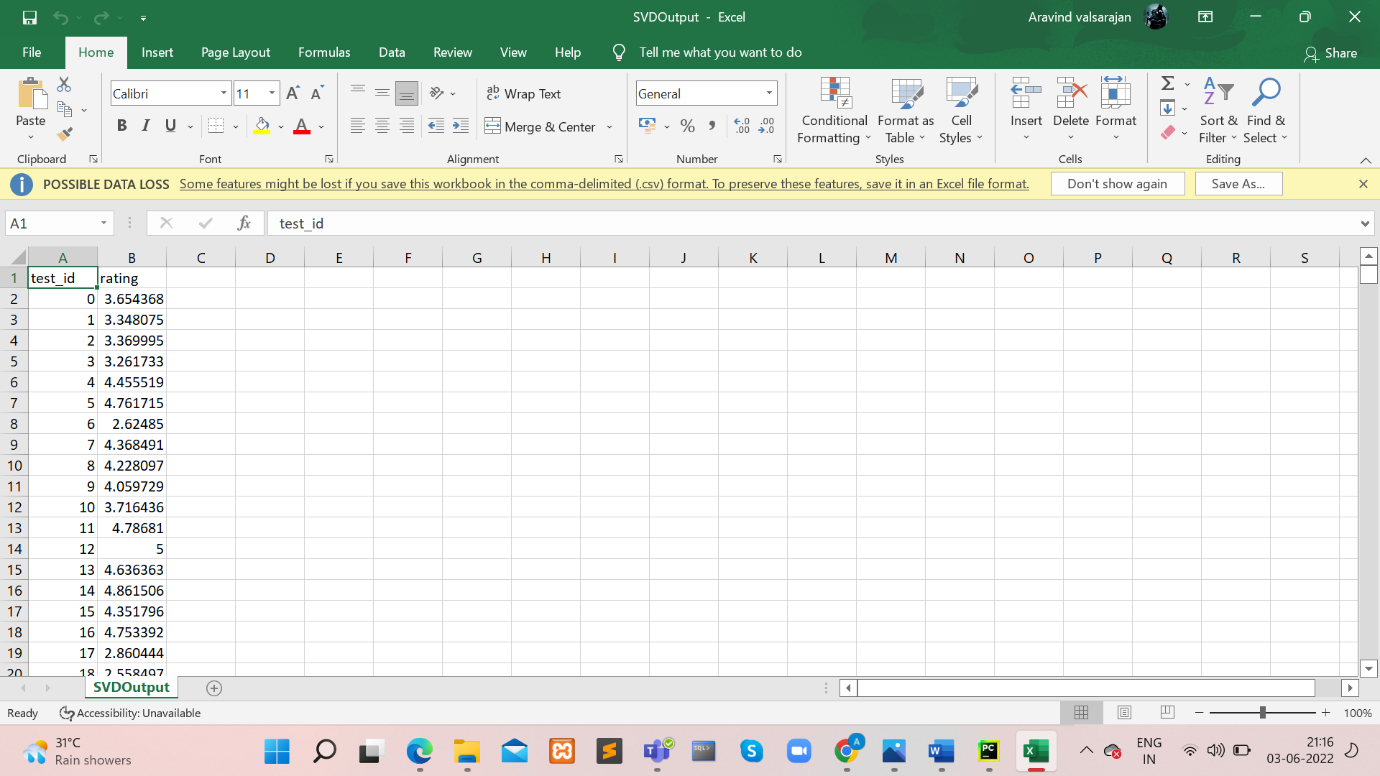
The SVD algorithm provided by Surprise is matrix factorization with user and item biases taken into account. The predicted rating for user u and item i is given by rˆui = µ + bu + bi + q T i pu (1) where µ is the average known rating for all items, bu and bi are the user and item biases, and qi and pu are the latent factor vectors. The values for bu, bi , qi and pu are found by minimizing the regularized squared error, given in equation 2, using stochastic gradient descent, iterating over all known ratings until convergence: X rui∈Train (rui − rˆui) 2 + λ(b 2 i + b 2 u + ||qi ||2 + ||pu||2 ) (2) The hyperparameters that were tuned for this algorithm were the number of latent features used, number of epochs and learning rate. The tests uploaded to Kaggle were the best out of the different cross-validation combinations. They all used 50 epochs. One was done with the default parameters of the algorithm (20 epochs, 0.005 learning rate and 100 features), another with 50 epochs, 50 features and 0.005 learning rate and another with 50 epochs, 50 features and 0.01 learning rate. The best results were obtained with the default parameters but with 50 features instead of 100 and obtained a Kaggle score of 1.31023

**3.1.1 Implementation of Singular Value Decomposition.**

import pandas as pd  
from scipy.sparse import lil\_matrix  
from surprise import NMF  
from surprise import SVD  
from surprise import SVDpp  
from surprise import Dataset  
from surprise.model\_selection import cross\_validate  
from surprise import Reader  
  
  
numUsers = 693208  
numItems = 145302  
  
# Main program  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 df = pd.read\_csv("train\_rating.txt", sep=",")  
  
 # Delete unused columns  
 del df['date']  
 # del df['train\_id']  
 #  
 # del df['test\_id']  
  
  
 # Set the rating scale and create the data for Surprise to use  
 reader = Reader(rating\_scale=(1, 5))  
 data = Dataset.load\_from\_df(df[['user\_id', 'business\_id', 'rating']], reader)  
  
 factors = 50  
  
 train\_set = data.build\_full\_trainset()  
  
 # Use SVD with surprise  
 algo = SVD(n\_factors=factors)  
 algo.fit(train\_set)  
  
 f = open('SVDOutput.csv','w')  
 f.write("test\_id,rating\n")  
 dfTest = pd.read\_csv("test\_rating.txt", sep=",")  
 for i in range(len(dfTest)) :  
 prediction = algo.predict(dfTest.at[i,'user\_id'],dfTest.at[i,'business\_id'],r\_ui=4,verbose=True)  
 predRating = prediction.est  
 f.write(str(i)+","+str(predRating)+'\n')  
  
# testdata = Dataset.load\_from\_df(dfTest[['user\_id', 'business\_id']], reader)  
# result=algo.test(testdata)  
# print(result)  
f.close()

**OUTPUT:**



****

**3.2 Singular Value Decomposition Plus Plus (SVD++)**

This is an extended version of SVD that takes into account implicit ratings. We decided to use this one because, once again, SVD is one of the most popular algorithms for solving this kind of problem, and we wanted to see the effect taking the implicit ratings into account would have on this data set. We used the implementation provided by Surprise, where the predicted rating is given by

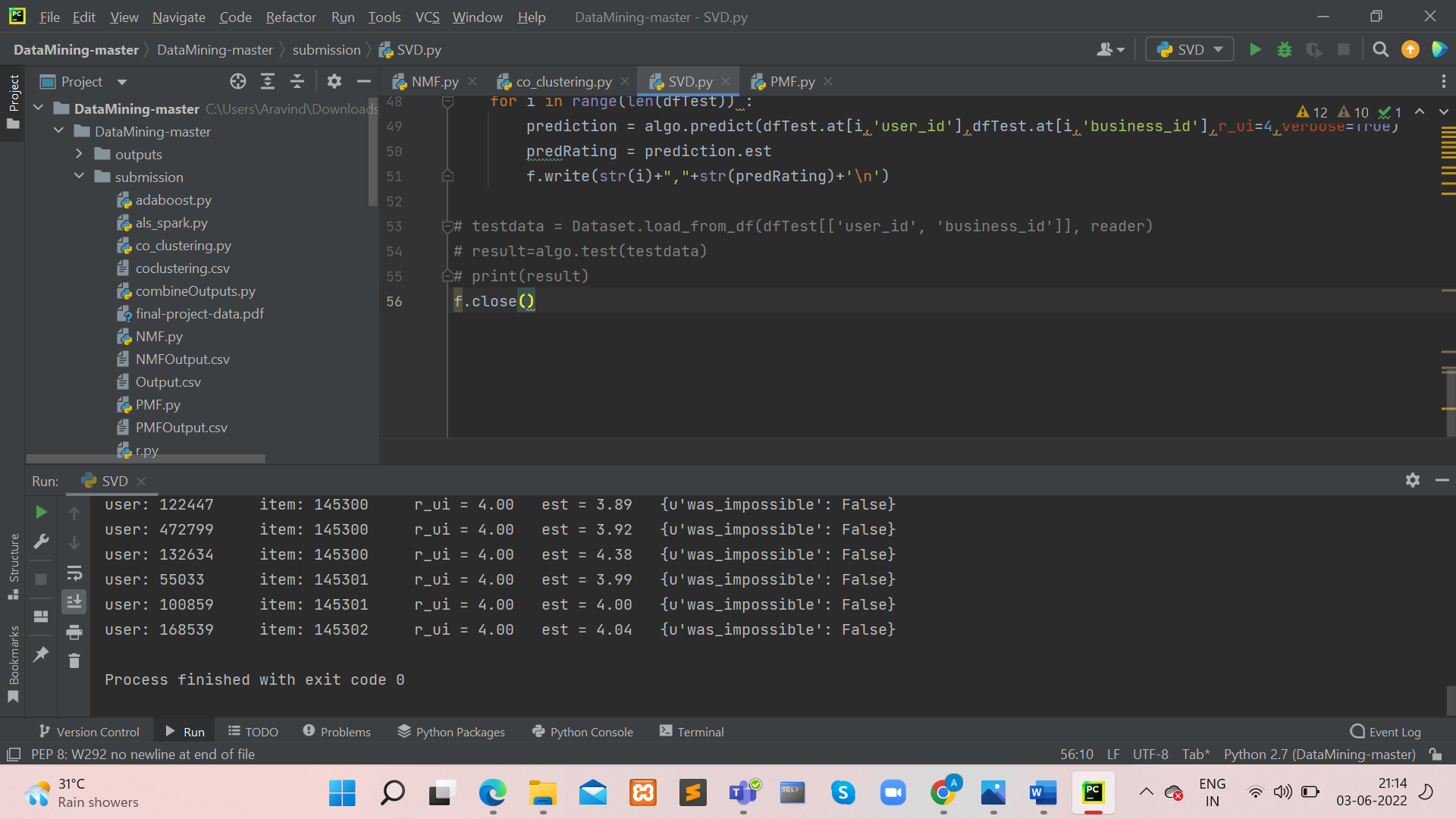
rˆui = µ + bu + bi + q T i pu + |Iu| − 1 2 X j∈Iu yj ! (3)

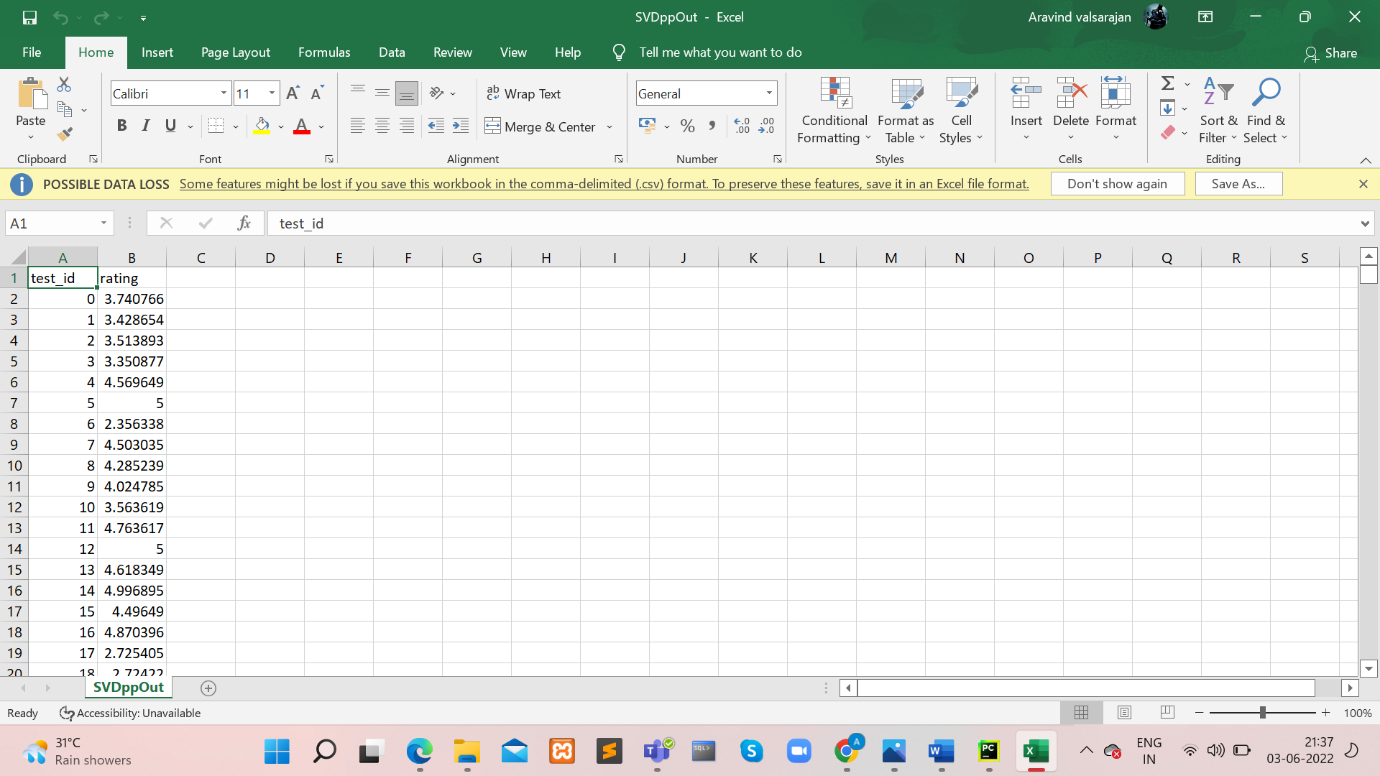
where yj are a new set of item factors that capture the fact that user u rated item j, regardless of the rating value, and Iu is the set of users who rated item j [4]. The hyperparameters tuned for this algorithm were the same as in SVD. The best results obtained and the ones that were uploaded to Kaggle, were one with default parameters (20 features, 20 epochs, 0.007 learning rate and 0.02 regularization term), another with 50 factors, 30 epochs, 0.01 learning rate and 0.02 regularization term, and another with 50 factors, 50 epochs, 0.05 learning rate and 0.02 regularization term. The one that obtained the best score was the first one, using default settings with a Kaggle score of 1.31466

**3.2.1 Implementation of Singular Value Decomposition Plus Plus**

import sys  
import operator  
import re, string  
import csv  
import math  
import numpy as np  
from scipy import sparse  
import pandas as pd  
from scipy.sparse import lil\_matrix  
from surprise import NMF  
from surprise import SVD  
from surprise import SVDpp  
from surprise import Dataset  
from surprise import evaluate, print\_perf  
from surprise import Reader  
  
  
numUsers = 693208  
numItems = 145302  
  
# Main program   
if \_\_name\_\_ == '\_\_main\_\_':  
  
 # Read csv into a pandas dataframe  
 dfRatings = pd.read\_csv(sys.argv[1])  
 dfTest = pd.read\_csv(sys.argv[2])  
  
 # Delete unused columns  
 del dfRatings['date']  
 del dfRatings['train\_id']  
 del dfTest['date']  
 del dfTest['test\_id']  
  
  
 # Set the rating scale and create the data for Surprise to use  
 reader = Reader(rating\_scale=(1, 5))  
 data = Dataset.load\_from\_df(dfRatings[['user\_id', 'business\_id', 'rating']], reader)  
  
 train\_set = data.build\_full\_trainset()   
  
 # Use SVD with surprise  
 algo = SVDpp()  
 algo.train(train\_set)  
  
 f = open('SVDOutput.csv','w')  
 f.write("test\_id,rating\n")  
 for i in range(len(dfTest)) :  
 prediction = algo.predict(dfTest.at[i,'user\_id'],dfTest.at[i,'business\_id'],r\_ui=4,verbose=True)  
 predRating = prediction.est  
 f.write(str(i)+","+str(predRating)+'\n')  
  
 f.close()

**OUTPUT:**



****

* 1. **Probabilistic Matrix Factorization (PMF)**

PMF was presented by Ruslan Salakhutdinov and Andriy Mnih also in the context of the Netflix contest. This method scales linearly with observations and, thus, can better handle large and sparse datasets [7]. To implement an algorithm equivalent to Probabilistic Matrix Factorization, 3 we again used Surprise's SVD algorithm, but this time setting the biased parameter to False [4]. This gives a predicted rating of

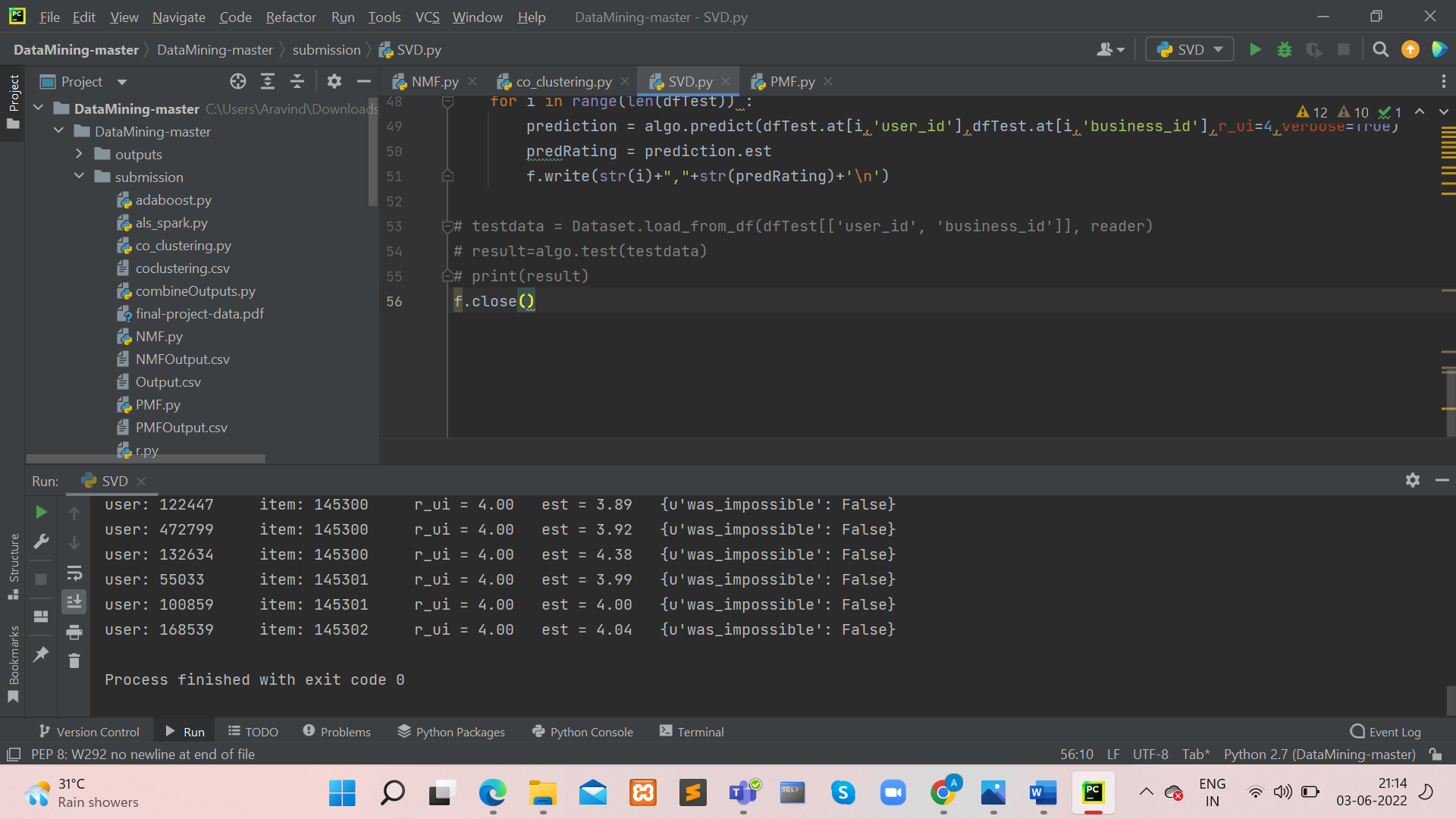
rˆui = q T i pu (4)

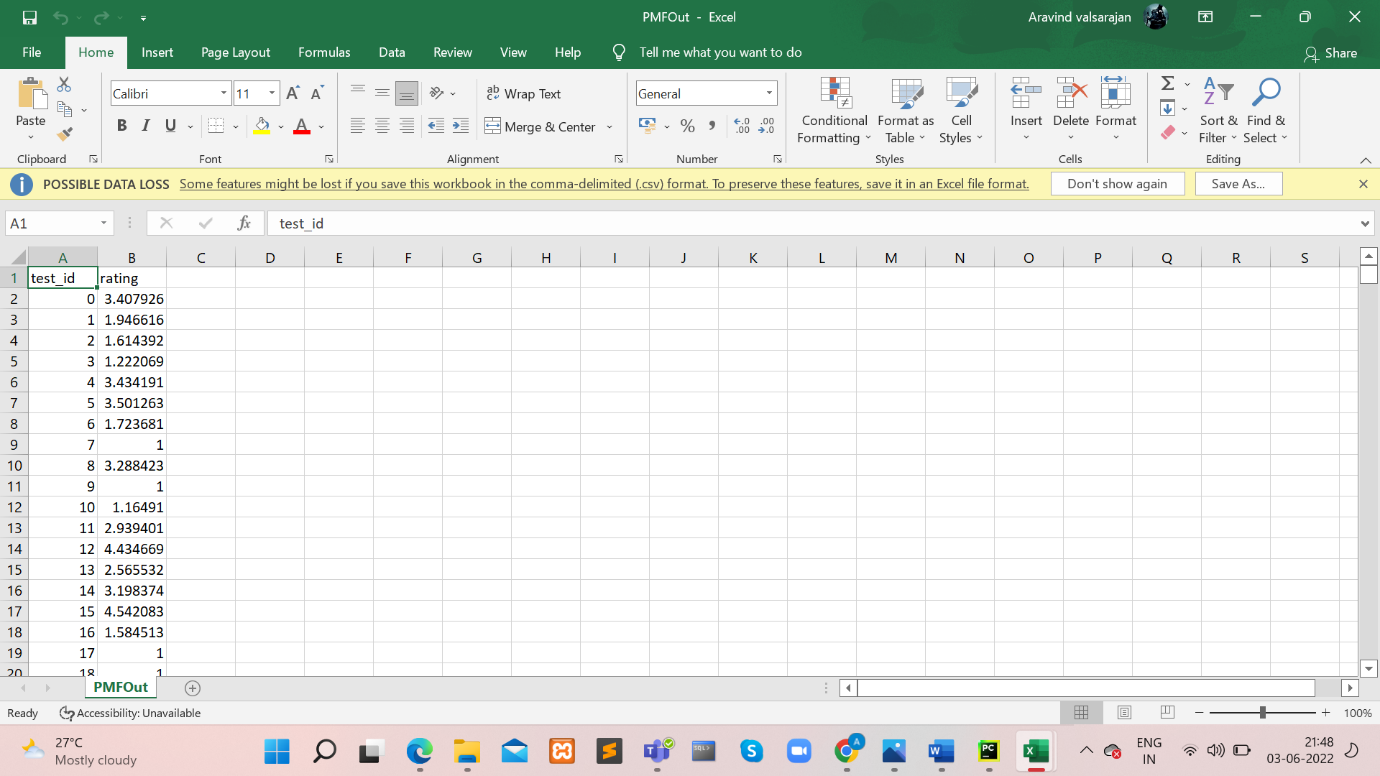
The hyperparameters that achieved the best results through cross-validation were 100 features, 20 epochs, 0.005 learning rate and 0.02 regularization term. With these hyperparameters we achieved a Kaggle score of 1.56279.

**3.3.1 Implementation of Probabilistic Matrix Factorization**

import pandas as pd  
import sys  
from surprise import Dataset  
from surprise import SVD  
from surprise import Reader  
  
  
# Main program   
if \_\_name\_\_ == '\_\_main\_\_':  
  
 # Read csv into a pandas dataframe  
 df = pd.read\_csv("train\_rating.txt", sep=",")  
  
 # Delete unused columns  
 del df['date']  
  
 # Set the rating scale and create the data for Surprise to use  
 reader = Reader(rating\_scale=(1, 5))  
 data = Dataset.load\_from\_df(df[['user\_id', 'business\_id', 'rating']], reader)  
  
 # Cross validation for tuning  
 # Split in 5 folds  
 # data.split(5)  
  
 # This part is to use all the data to train and get the output  
 train\_set = data.build\_full\_trainset()  
  
 # Use PMF with surprise. To use PMF you use SVD with the parameter biased = False  
 algo = SVD(biased = False)  
 algo.fit(train\_set)  
  
 f = open('PMFOutput.csv','w')  
 f.write("test\_id,rating\n")  
 dfTest = pd.read\_csv("test\_rating.txt", sep=",")  
 for i in range(len(dfTest)):  
 prediction = algo.predict(dfTest.at[i,'user\_id'],dfTest.at[i,'business\_id'],r\_ui=4,verbose=True)  
 predRating = prediction.est  
 f.write(str(i)+","+str(predRating)+'\n')  
  
 f.close()

**OUTPUT:**



****

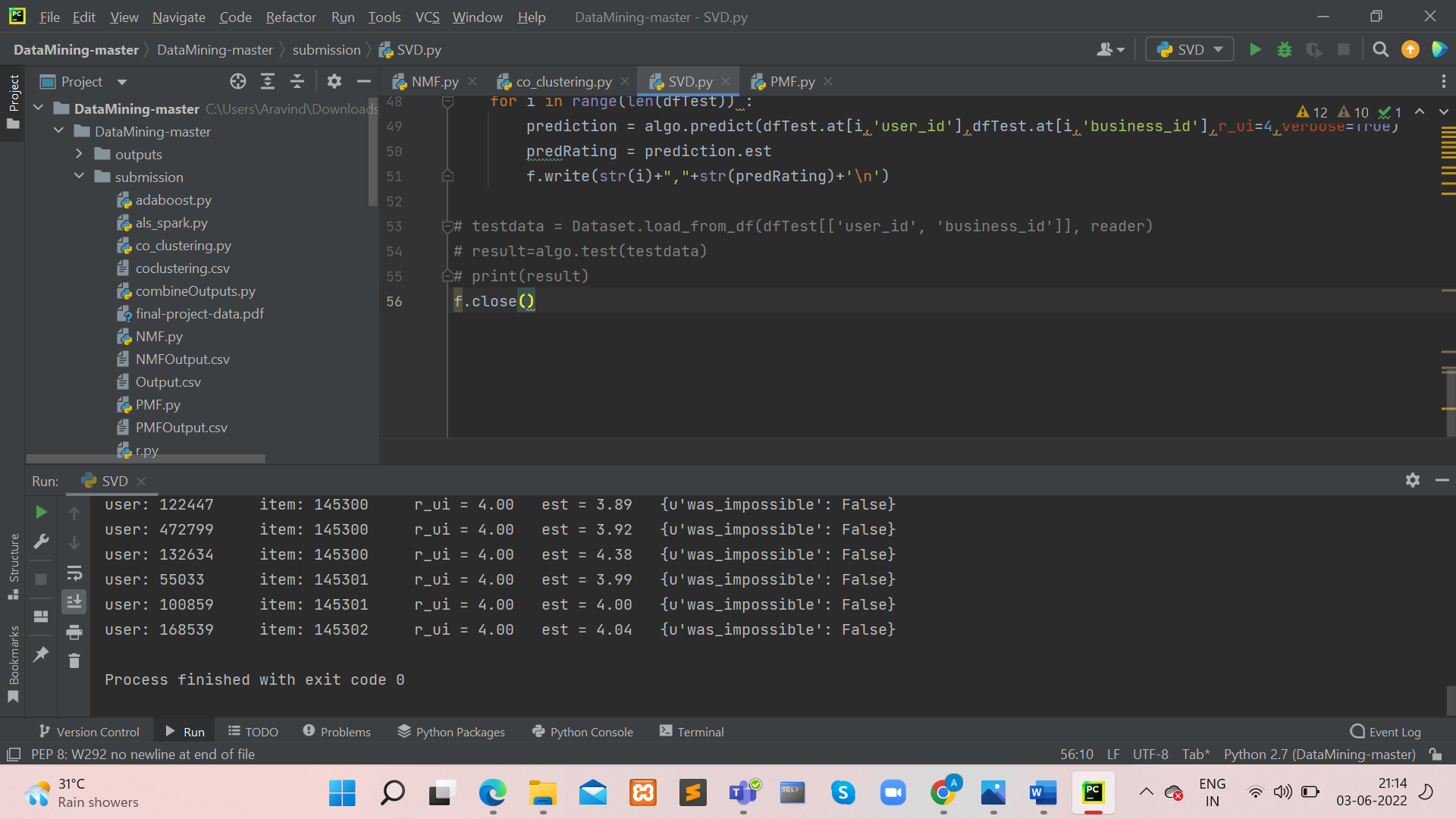
**3.4 Non-negative Matrix Factorization (NMF)**

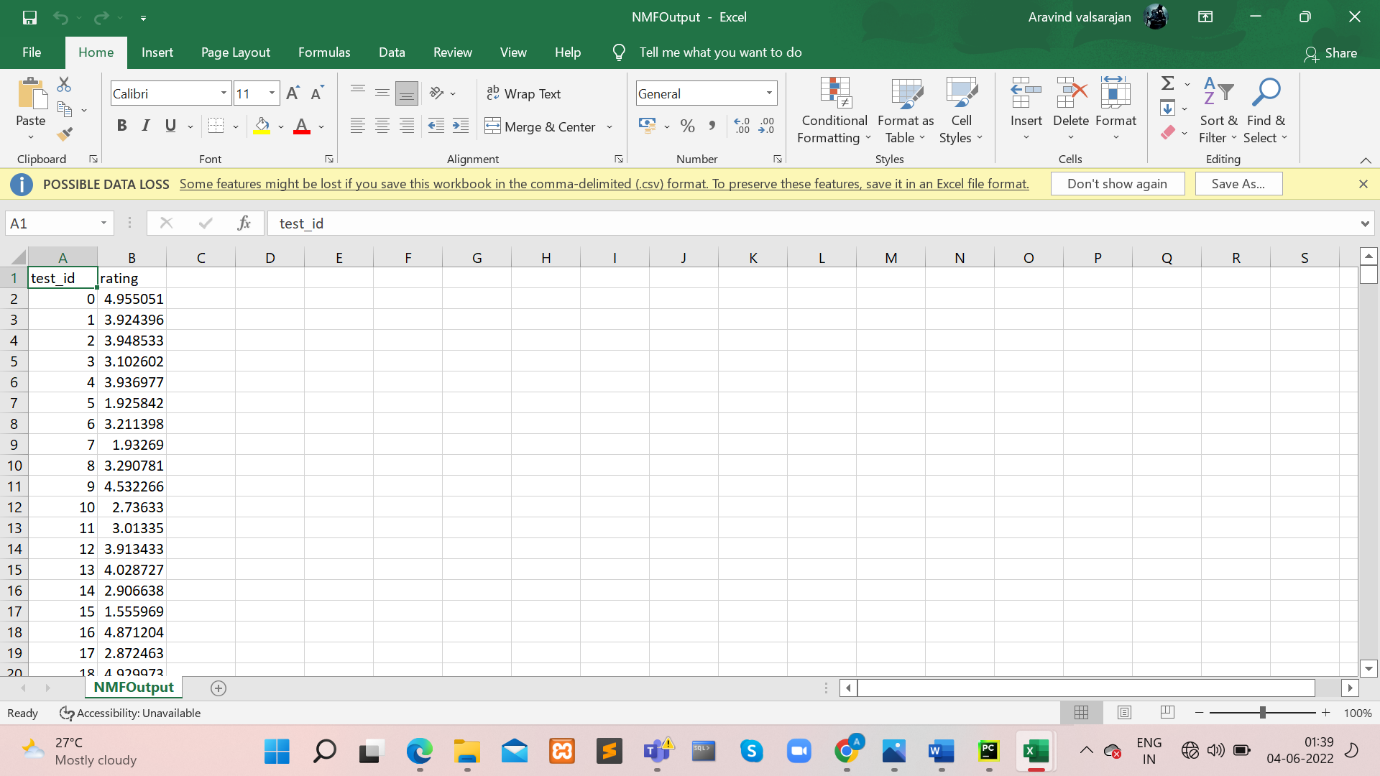
This algorithm is very similar to SVD. The basic difference is that it keeps the user and item factors always positive [4]. It does this by using an adaptive learning rate that is adjusted during the update phase on each iteration to account for any negative components [6]. A cross-validation search for the best hyperparameter settings gave the best results with 15 features, 50 epochs, 0.005 learning rate and 0.06 regularization term for users and items. These settings achieved a Kaggle score of 1.52098

* + 1. **Implementation of Non-negative Matrix Factorization**

import pandas as pd  
import sys  
from surprise import Dataset  
from surprise import NMF  
from surprise import Reader  
  
  
# Main program  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 # Read csv into a pandas dataframe  
 df = pd.read\_csv("train\_rating.txt", sep="," )  
  
 # Delete unused columns  
  
 del df['date']  
  
  
 # Set the rating scale and create the data for Surprise to use  
 reader = Reader(rating\_scale=(1, 5))  
 data = Dataset.load\_from\_df(df[['user\_id', 'business\_id', 'rating']], reader)  
  
 # Cross validation for tuning  
 # Split in 5 folds  
 # data.split(5)  
  
 # This part is to use all the data to train and get the output  
 train\_set = data.build\_full\_trainset()  
  
 # Use NMF with surprise  
 algo = NMF()  
 algo.fit(train\_set)  
  
 f = open('NMFOutput.csv','w')  
 f.write("test\_id,rating\n")  
 dfTest = pd.read\_csv("test\_rating.txt", sep=",")  
 for i in range(len(dfTest)) :  
 prediction = algo.predict(df.at[i,'user\_id'],df.at[i,'business\_id'],r\_ui=4,verbose=True)  
 predRating = prediction.est  
 f.write(str(i)+","+str(predRating)+'\n')  
  
 f.close()

**OUTPUT:**



****

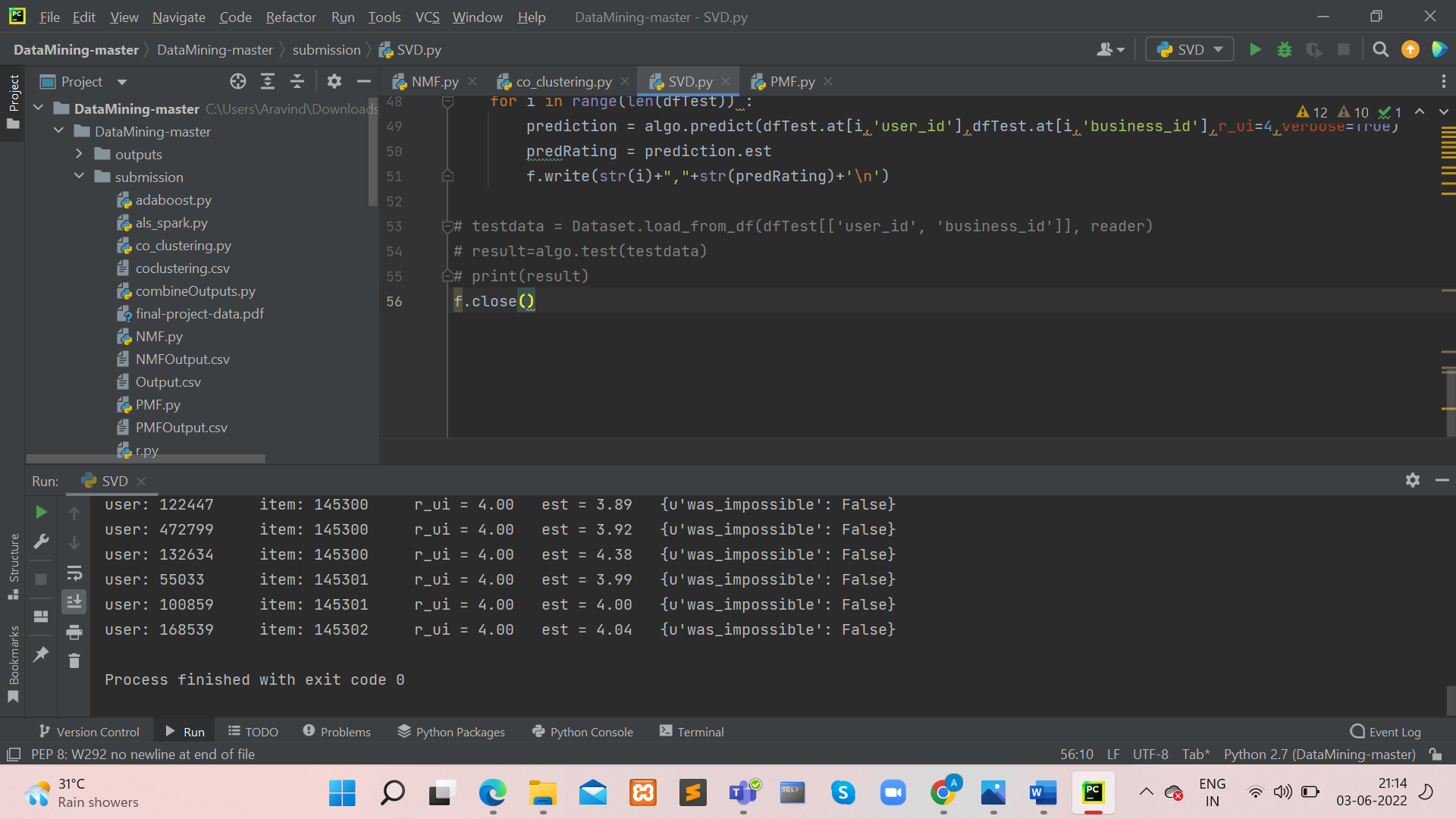
* 1. **Alternating Least Squares (ALS)**

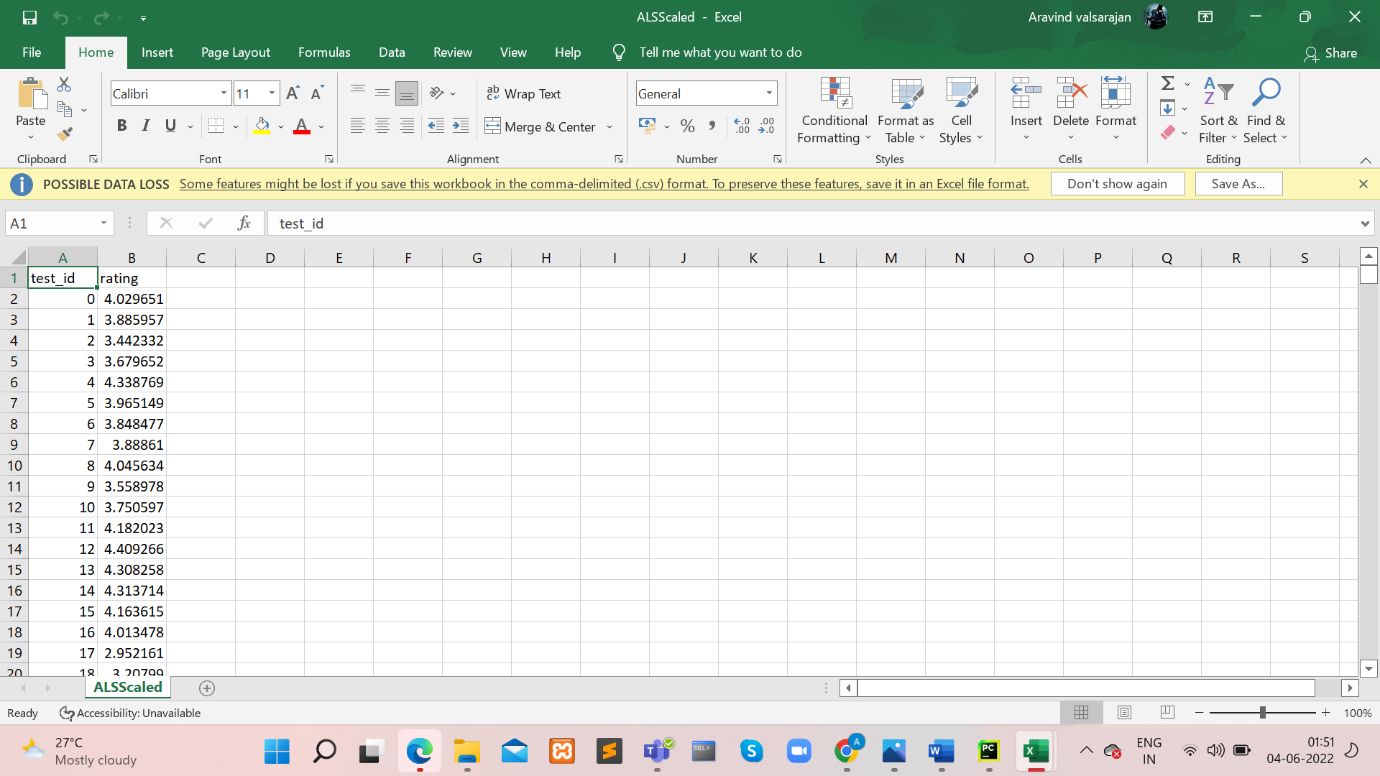
The ALS technique tries to learn the factor vectors pu and qi by minimizing the regularized squared error on the set of known ratings: minp ∗,q∗ = X (u,i)∈κ (rui − q T i pu) 2 + λ(||qi ||2 + ||pu||2 ) Since both qi and pu are unknowns, this equation is not convex. However, if we fix one of the unknowns, the optimization problem becomes quadratic and can be solved optimally. Thus, ALS techniques alternate between fixing the qi and fixing the pu. When all pu are fixed, the system recomputes the qi by solving a least-squares problem, and vice versa. This ensures that each step decreases the equation until convergence [5]. We used the implementation of ALS in the Apache Spark MLlib library [1]. We used maximum number of iterations as 10, regularization parameter as 0.1, and the predictions made by this model scored 1.41254 on Kaggle.

* + 1. **Implementation of Alternating Least Squares**

from pyspark.ml.recommendation import ALS  
from pyspark.sql import SparkSession  
import pandas as pd  
import sys  
  
spark = SparkSession.builder.appName('ensemble average').getOrCreate()  
  
# normalize a value to the range 1-5  
def normalize(x):  
 return 4 \* (x - min\_value) / (max\_value - min\_value) + 1  
  
train\_input = sys.argv[1]  
test\_input = sys.argv[2]  
output = sys.argv[3]  
  
# Read train and test data to Pandas dataframes  
train\_raw = pd.read\_csv(train\_input)  
test\_raw = pd.read\_csv(test\_input)  
  
# Delete unused columns  
test\_raw = test\_raw.drop(['date'],axis=1)  
train\_raw = train\_raw.drop(['train\_id','date'],axis=1)  
  
# Create Spark Dataframes  
train = spark.createDataFrame(train\_raw)  
test = spark.createDataFrame(test\_raw)  
  
# Create ALS predictor, fit the model and generate the predictions  
als = ALS(userCol="user\_id", itemCol="business\_id", ratingCol="rating")  
model = als.fit(train)  
predictions = model.transform(test)  
  
# Store result in a pandas dataframe  
predict\_df = predictions.select('test\_id','prediction').coalesce(1).orderBy('test\_id').toPandas()  
  
### Scaling the model to range 1-5  
max\_value = predict\_df.prediction.max()  
min\_value = predict\_df.prediction.min()  
predict\_df['rating'] = predict\_df.prediction.apply(normalize)  
  
predict\_df = predict\_df[['test\_id','rating']]  
  
# Save the predictions in a file to submit to Kaggle  
predict\_df.sort\_values('test\_id').to\_csv(output,index=False)

**OUTPUT:**





* 1. **Co-Clustering**

Co-clustering is a collaborative filtering technique for recommendations whereby the user-item matrix, M, is co-clustered into user-clusters and item-clusters. It is the only non-matrix factorization algorithm we had any success with. The predicted recommendation for user u and item i is

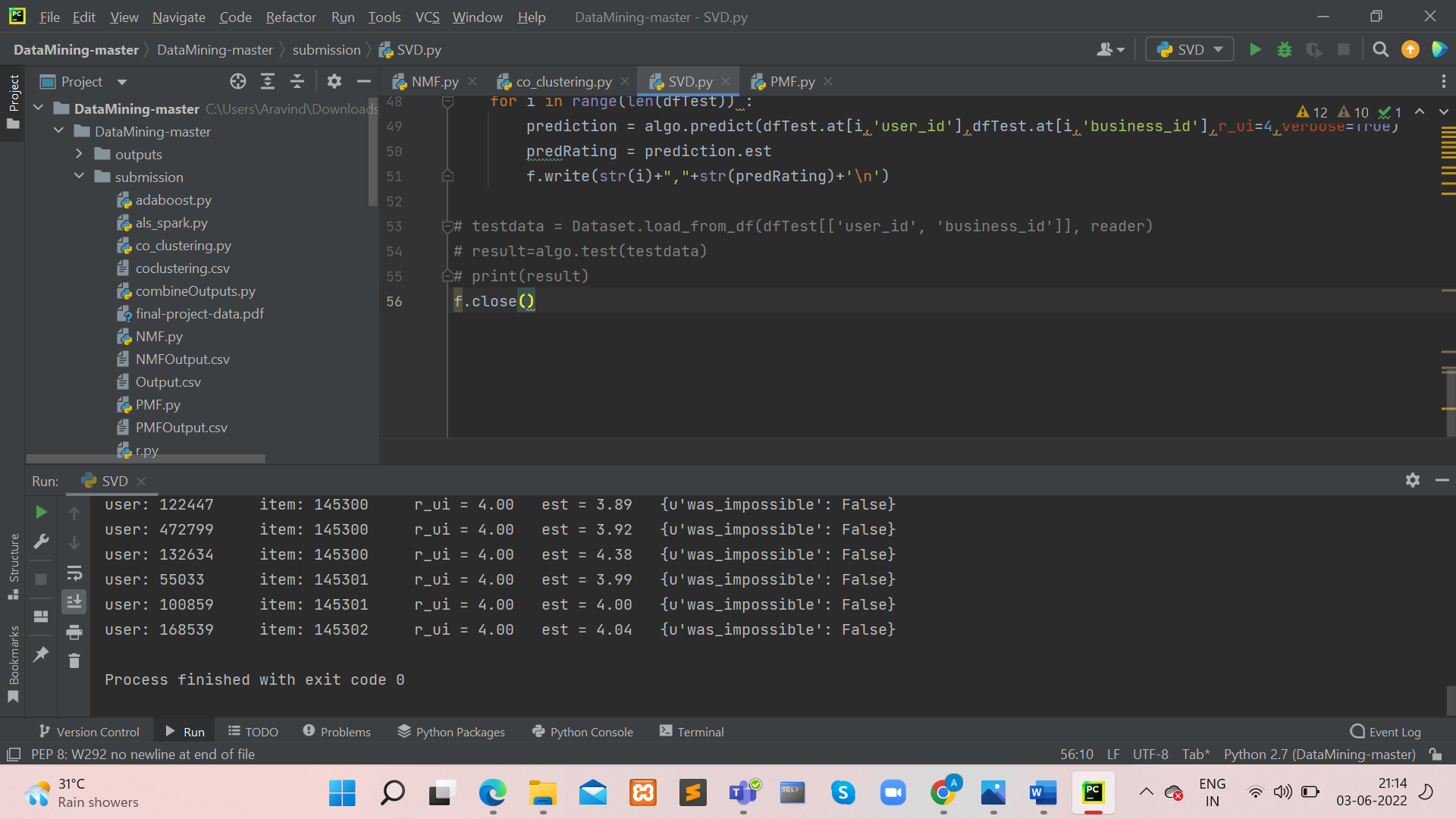
rˆui = Cui + (µu − Cu) + (µi − Ci)

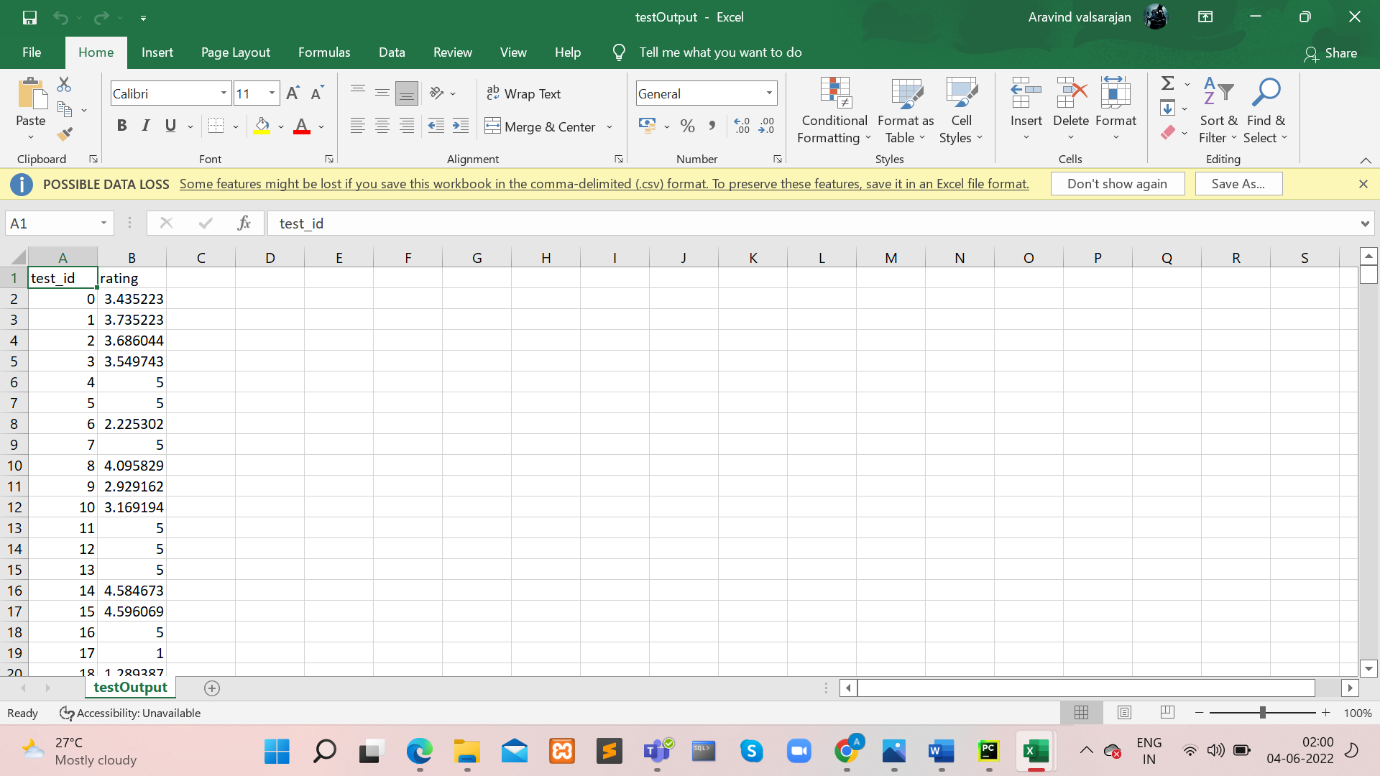
where Cui is the average rating of the particular user-item co-cluster, µu and µi are the average ratings of user u and item i, and Cu and Ci are the average ratings of the user-cluster and item cluster. George and Merugu showed co-clustering to be comparable to SVD and NMF in terms of accuracy, but its particular strength is its support for dynamic addition of new users and items[3]. While not relevant to this project, this is a big strength for online systems where a computationally intensive retraining of the recommendation model every time a new user or item is added would 4 be undesirable. We used the implementation of Co-clustering provided by Surprise. Hyperparameters tuned were the number of user clusters, the number of item clusters, and the number of training epochs. Through a cross-validation grid-search of parameters, we determined the optimal settings to be 3 user-clusters, 3 item-clusters and 100 epochs. The predictions made by this model with these parameters scored 1.46421 on Kaggle.

* + 1. **Implementation of Co-Clustering**

import pandas as pd  
from scipy.sparse import lil\_matrix  
from surprise import NMF  
from surprise import SVD  
from surprise import SVDpp  
from surprise import Dataset  
from surprise.model\_selection import cross\_validate  
from surprise import Reader  
from surprise import CoClustering,Dataset, Reader  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 df = pd.read\_csv("train\_rating.txt", sep=",")  
  
 # Delete unused columns  
 del df['date']  
  
reader = Reader(rating\_scale=(1, 5))  
data = Dataset.load\_from\_df(df[['user\_id', 'business\_id', 'rating']], reader)  
  
# create a trainset object   
# reader = Reader()  
# data = Dataset.load\_from\_df(dftrain, reader)  
trainset = data.build\_full\_trainset()  
  
  
#   
  
  
# create a co-clustering algorithm  
algo = CoClustering(n\_cltr\_u=3, n\_cltr\_i=3, n\_epochs=100)  
algo.fit(trainset)  
  
  
# use the trained algorithm to predict ratings for every user in the test set  
f = open('testOutput.csv','w')  
f.write("test\_id,rating\n")  
dfTest = pd.read\_csv("test\_rating.txt", sep=",")  
for i in range(len(dfTest)) :  
 prediction = algo.predict(dfTest.at[i,'user\_id'],dfTest.at[i,'business\_id'],r\_ui=4,verbose=True)  
 predRating = prediction.est  
 f.write(str(i)+","+str(predRating)+'\n')  
  
f.close()

**OUTPUT:**





**4.CONCLUSION**

In this project our objective was to classify and recommend a rating for each business entry in the input data. The data contained ratings from yelp is used to train each model, after training we used business id and user id attributes of data to predict the ratings for businesses.

**5.REFERENCES**

[1] Collaborative filtering - rdd-based api. https://spark.apache.org/docs/2.2.0/ mllib-collaborative-filtering.html.

[2] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In Data Mining, Fifth IEEE international conference on, pages 4–pp. IEEE, 2005. [4] Nicolas Hug. Matrix factorization-based algorithms. http://surprise.readthedocs.io/en/ stable/matrix\_factorization.html, 2015

[3] Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. An efficient non-negative matrix factorization-based approach to collaborative filtering for recommender systems. IEEE Transactions on Industrial Informatics, 10(2):1273–1284, 2014.

[4] Ariel Bar, Lior Rokach, Guy Shani, Bracha Shapira, and Alon Scholar. Improving simple collaborative filtering models using ensemble methods. In International Workshop on Multiple Classifier Systems, pages 1–12. Springer, 2013.

[5] Albert Au Yeung. Matrix factorization: A simple tutorial and implementation in python. www.quuxlabs.com/blog/2010/09

[6] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. IEEE Computer Society, pages 42–49, 2009.

[7] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In Advances in neural information processing systems, pages 1257–1264, 2008.