SUMMER TRAINING REPORT

**On**

**Movies Recommendation System**

**Submitted by**

**Suryansh Chaturvedi**

**(171500349)**

Department of Computer Engineering & Applications

**Institute of Engineering & Technology**



**GLA University**

**Mathura,U.P.-281406, INDIA**

**2019**

**Department of computer Engineering and Applications**

**GLA University, Mathura**

**17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,**

**Mathura – 281406**



**Declaration**

I hereby declare that the work which is being presented in the SummerTraining “**Movie Recommendation System”,**in partial fulfillment of the requirements for Summer Training viva voce, is an authentic record of my own work carried under the supervision of “IBM Machine Learning Specialist.”

Signature of Candidate:

Name of Candidate: Suryansh Chaturvedi

Roll. No.: 171500349

Course: B.tech. (Computer Science and Engineering)

Year: 3rd

Semester: 5th

****

**Summer Training Synopsis**

**B.tech.(CSE)-Batch 2017-2021**

**Student Information:**

|  |  |
| --- | --- |
| Name: Suryansh Chaturvedi | University Roll. No. 171500349 |
| Mobile: 8800186582 | Email: suryansh.chaturvedi\_cs17@gla.ac.in |

**Information about Industry/Organization:**

|  |  |
| --- | --- |
| Industry/Organization Name with full Address | Edx.org IBM |
| Contact Person | Name & Designation: Saeed Aghabozorgi(Senior Data Scientist  IBM)  Mobile/email: <https://ca.linkedin.com/in/saeedaghabozorgi> |

**Project Information:**

|  |  |
| --- | --- |
| Title Of Project/Training/Task | Movie Recommendation System |
| Role & Responsibility |  |
| Technical Details | Software Requirements: Python |
| Training Implementation Details | Partial Implemented |
| Training Period | Start Date: 21/05/2019  End Date: 06/07/2019  Duration Of Training (In Weeks): 06 |

**Summary of the Training Work:**

The project titled ”Movie Recommendation System” is use to recommend the movies to the user.First,the machine is trained using the ratings dataset given by user to the movie they watched.Then,by using these ratings providing the recommendations to the user.In this project we are giving 10 recommendations to the user.

**Acknowledgement**

I thank the almighty for giving me the courage and perseverance in completing the project.

This project itself is acknowledgements for all those people who have give us their heartfelt co-operation in making this project a grand success.

I extend my sincere thanks to Saeed Aghabozorgi, Senior Data Scientist at IBM for providing valuable guidance at every stage of this project work. I am profoundly grateful towards the unmatched services rendered by her.

Last but not least , we would like to express our deep sense of gratitude and earnest thanks giving to our dear parents for their moral support and heartfelt cooperation in doing the main project.

 **Certificate**

**Abstract**

Even though peoples’ tastes may vary, they generally follow patterns. I mean that there are similarities in the things that people tend to like or another way to look at it, is that people tend to like things in the same category or things that share the same characteristics. For example, if you’ve recently purchased a book on Machine Learning in Python and you’ve enjoyed reading it, it’s very likely that you’ll also enjoy reading a book on Data Visualization.People also tend to have similar tastes to those of the people they’re close to in their lives.Recommender systems try to capture these patternsand similar behaviors, to help predict what else you might like.

There are generally 2 main types of recommendation systems: Content-based and collaborative filtering.Content-based systems try to figure out what a user's favorite aspects of an item are, and then make recommendations on items that share those aspects.Collaborative filtering techniques find similar groups of users, and provide recommendations based on similar tastes within that group.In short, it assumes that a user might be interested in what similar users are interested in.We are using Collaborative filtering in this project using SVD algorithm.For this we work on movielens data(i.e. ratings) which are given by user.using this algorithm we can find the estimate rating of movies for that user and then recommend it accordingly.

Recommender systems have many applications that I’m sure you’re already familiar with. Indeed, Recommender systems are usually at play on many websites. For example, suggesting books on Amazon and movies on Netflix. In fact, everything on Netflix’s website is driven by customer selection. If a certain movie gets viewed frequently enough, Netflix’s recommender system ensures that that movie gets an increasing number of recommendations. Another example can be found in a daily-use mobile app, where a recommender engine is used to recommend anything from where to eat, or, what job to apply to.On social media, sites like Facebook or LinkedIn, regularly recommend friendships.

**Contents**

**Acknowlegment……………………………………………………………………..**04

**Certificate……………………………………………………………………………**05

**Abstract……………………………………………………………………………...**06

1. **Introduction……………….………………………………………………08**

Introduction to recommender system………..……………………...09

Types of recommender system………..…………………………….10

Pre requisites……………………..………………………………….08

1. **Software Requirement…………………………………………………...11**

Python

Introduction toSurprise……..……………………………………....11

Modules in Surprise and their functionality…..……………………..11

SVD……………………..…..…………………….………..12

Reader Class……….……….…..…………………….…….13

Dataset………………..……..………………….…………..14

Trainset Class……...……………...………………………...15

Introduction to Tkinter......................................................................16

**3.Dataset……………………………………………………………………...20**

Training Dataset……………………………………………………20

Testing Dataset………………………………………….................20

**4. Implementation and User Interface……..…………………………….....21**

**5.Conclusion……………….…….……………….…………………………..24**

**6.Bibliography..................................................................................................25**

**7.** **Appendices………………….….………………………………………….26**

**Chapter 1**

**Introduction**

**Introduction to recommender system**

During the last few decades, with the rise of Youtube, Amazon, Netflix and many other such web services, recommender systems have taken more and more place in our lives.From e-commerce (suggest to buyers articles that could interest them) to online advertisement (suggest to users the right contents, matching their preferences), recommender systems are today unavoidable in our daily online journeys.In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy or anything else depending on industries).

Recommender systems are really critical in some industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors. As a proof of the importance of recommender systems, we can mention that, a few years ago, Netflix organised a challenges (the “Netflix prize”) where the goal was to produce a recommender system that performs better than its own algorithm with a prize of 1 million dollars to win.

**Types of Recommender System**

There are generally 2 main types of recommendation systems: Content-based and collaborative filtering.

**Collaborative filtering:-** Collaborative methods for recommender systems are methods that are based solely on the past interactions recorded between users and items in order to produce new recommendations. These interactions are stored in the so-called “user-item interactions matrix”.

Fig 1.1:-Illustration of user item interactions matrix

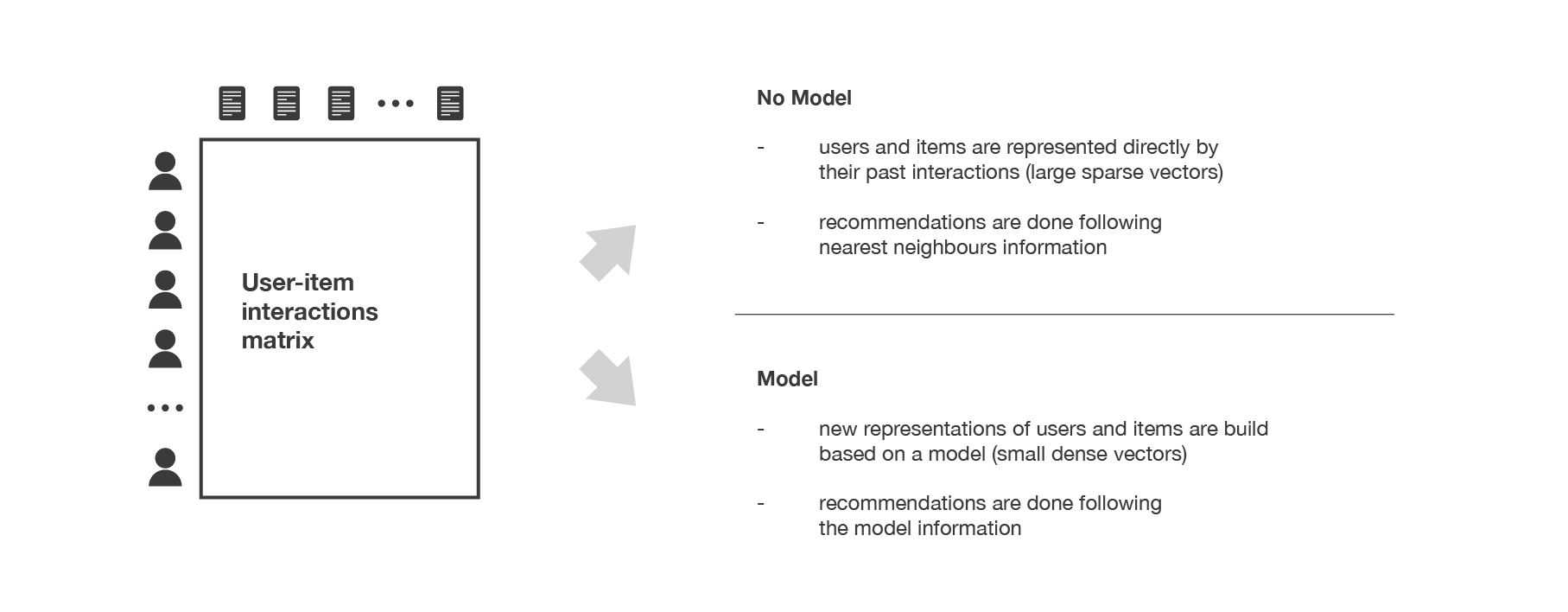
Then, the main idea that rules collaborative methods is that these past user-item interactions are sufficient to detect similar users and/or similar items and make predictions based on these estimated proximities.  
  
The class of collaborative filtering algorithms is divided into two sub-categories that are generally called memory based and model based approaches. Memory based approaches directly works with values of recorded interactions, assuming no model, and are essentially based on nearest neighbours search (for example, find the closest users from a user of interest and suggest the most popular items among these neighbours). Model based approaches assume an underlying “generative” model that explains the user-item interactions and try to discover it in order to make new predictions.

Fig 1.2:-Overview of collaborative filtering methods paradigm

**Content-based filtering:-** Unlike collaborative methods that only rely on the user-item interactions, content based approaches use additional information about users and/or items. If we consider the example of a movies recommender system, this additional information can be, for example, the age, the sex, the job or any other personal information for users as well as the category, the main actors, the duration or other characteristics for the movies (items).

Then, the idea of content based methods is to try to build a model, based on the available “features”, that explain the observed user-item interactions. Still considering users and movies, we will try, for example, to model the fact that young women tend to rate better some movies, that young men tend to rate better some other movies and so on. If we manage to get such model, then, making new predictions for a user is pretty easy: we just need to look at the profile (age, sex, …) of this user and, based on this information, to determine relevant movies to suggest.

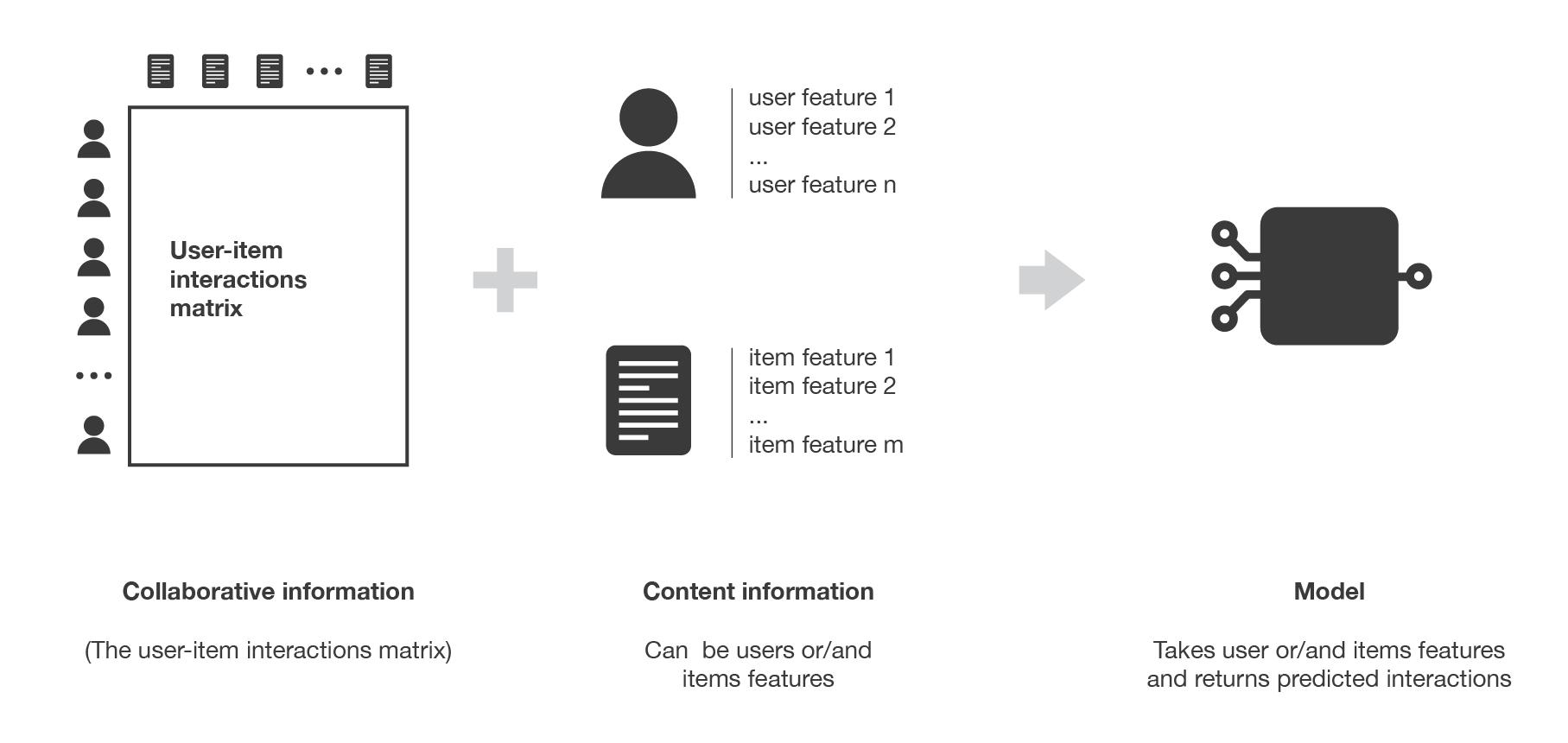


Fig 1.3:-Overview of content based filtering paradigm

**Pre-requisites**

Hands-on knowledge of tkinter(For User Interface) and surprise(SVD,Dataset,Reader) is essential before working on recommendation system. Make sure that you have the following packages installed and running before implementing it.

**Chapter 2**

**Software Requirement**

**Introduction to Surprise**

[Surprise](http://surpriselib.com/) is a Python [scikit](https://www.scipy.org/scikits.html) building and analyzing recommender systems that deal with explicit rating data.

[Surprise](http://surpriselib.com/) **was designed with the following purposes in mind:**

* Give users perfect control over their experiments. To this end, a strong emphasis is laid on [documentation](http://surprise.readthedocs.io/en/stable/index.html), which we have tried to make as clear and precise as possible by pointing out every detail of the algorithms.
* Alleviate the pain of [Dataset handling](http://surprise.readthedocs.io/en/stable/getting_started.html#load-a-custom-dataset). Users can use both built-in datasets ([Movielens](http://grouplens.org/datasets/movielens/), [Jester](http://eigentaste.berkeley.edu/dataset/)), and their own custom datasets.
* Provide various ready-to-use [prediction algorithms](http://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html) such as [baseline algorithms](http://surprise.readthedocs.io/en/stable/basic_algorithms.html), [neighborhood methods](http://surprise.readthedocs.io/en/stable/knn_inspired.html), matrix factorization-based ( [SVD](http://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVD), [PMF](http://surprise.readthedocs.io/en/stable/matrix_factorization.html#unbiased-note), [SVD++](http://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVDpp), [NMF](http://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.NMF)), and [many others](http://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html). Also, various [similarity measures](http://surprise.readthedocs.io/en/stable/similarities.html) (cosine, MSD, pearson…) are built-in.
* Make it easy to implement [new algorithm ideas](http://surprise.readthedocs.io/en/stable/building_custom_algo.html).
* Provide tools to [evaluate](http://surprise.readthedocs.io/en/stable/model_selection.html), [analyse](http://nbviewer.jupyter.org/github/NicolasHug/Surprise/tree/master/examples/notebooks/KNNBasic_analysis.ipynb/) and [compare](http://nbviewer.jupyter.org/github/NicolasHug/Surprise/blob/master/examples/notebooks/Compare.ipynb) the algorithms performance. Cross-validation procedures can be run very easily using powerful CV iterators (inspired by [scikit-learn](http://scikit-learn.org/) excellent tools), as well as [exhaustive search over a set of parameters](http://surprise.readthedocs.io/en/stable/getting_started.html#tune-algorithm-parameters-with-gridsearchcv).

The name SurPRISE roughly stands for Simple Python RecommendatIon System Engine.

**Installing surprise module**

To install surprise,we have to run “pip install scikit-surprise” in CMD or “conda install –c conda-forge scikit-surprise” while using Anaconda Prompt.

**Modules in Surprise and their functionality**

**SVD(Single Value Decomposition)**

SVD in the context of recommendation systems is used as a collaborative filtering algorithm. For those of you who don’t know, collaborative filtering is a method to predict a rating for a user item pair based on the history of ratings given by the user and given to the item. Most collaborative filtering algorithms are based on user-item rating matrix where each row represents a user, each column an item. The entries of this matrix are ratings given by users to items.

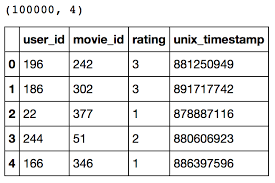
****

Fig 2.1:-Collaborative Filtering is applied on this type of datasets

****SVD is a matrix factorization technique.For the purpose of the recommendation systems however, we are only interested in the matrix factorization part keeping same dimensionality. The matrix factorization is done on the user-item ratings matrix. From a high level, matrix factorization can be thought of as finding 2 matrices whose product is the original matrix.

Fig 2.2:- User Product matrix

****

****

Fig 2.3 :-Factorised matrix for users and movies with 2 latent features

Now to predict rating of any movie for a user we can find the estimate rating by finding the product of latent features of user with that movie.

**Defining the SVD model**

SVDAlgo=SVD(n\_factors,random\_state)

Parameters:

* **n\_factors** – The number of factors. Default is 100.
* **random\_state** (int, RandomState instance from numpy, or None) – Determines the RNG that will be used for initialization. If int,random state will be used as a seed for a new RNG. This is useful to get the same initialization over multiple calls to fit(). If RandomState instance, this same instance is used as RNG. If None, the current RNG from numpy is used. Default is None.

Here,SVDAlgo stores the SVD model which can be trained using training dataset.

**Reader class**

The Reader class is used to parse a file containing ratings.

Such a file is assumed to specify only one rating per line, and each line needs to respect the following structure: (user , item , rating , [timestamp])

where the order of the fields and the separator (here ‘,’) may be arbitrarily defined (see below). brackets indicate that the timestamp field is optional.

reader = Reader(line\_format='user item rating timestamp', sep=',', skip\_lines=1)

reader = Reader(line\_format='user item rating timestamp', sep=',', skip\_lines=1)

Parameters:

* **line\_format (string)** – The fields names, in the order at which they are encountered on a line. Please note that line format  is always space-separated (use the sep parameter). Default is ‘user item rating’.
* **sep (*char*)** – the separator between fields. Example : ‘,’.
* **skip\_lines (int, optional)** – Number of lines to skip at the beginning of the file. Default is 0.

**Dataset module**

The dataset module defines the Dataset class and other subclasses which are used for managing datasets.

**Classmethod:load\_from\_file(**file\_path, reader)

Load a dataset from a (custom) file.

Use this if you want to use a custom dataset and all of the ratings are stored in one file. You will have to split your dataset using the split method.

Parameters:

* **file\_path** (string) – The path to the file containing ratings.
* **reader** (Reader) – A reader to read the file

**Classmethod:build\_full\_trainset()**

Do not split the dataset into folds and just return a trainset as is, built from the whole dataset.It returns TrainSet class object.

Ex:-

reader = Reader(line\_format='user item rating timestamp', sep=',', skip\_lines=1)

dataset=Dataset.load\_from\_file(‘C:\\Desktop\\ratings.csv’,reader=reader)

trainset=dataset.build\_full\_trainset()

**Trainset class**

A trainset contains all useful data that constitute a training set.

It is used by the fit() method of every prediction algorithm. You should not try to build such an object on your own but rather use the Dataset.folds() method or the Dataset.build\_full\_trainset() method.

Trainsets are different from Datasets. You can think of a Dataset as the raw data, and Trainsets as higher-level data where useful methods are defined. Also, a Dataset may be comprised of multiple Trainsets (e.g. when doing cross validation).

Trainset class contains:-

1. **ur**:- The users ratings. This is a dictionary containing lists of tuples of the form (item\_inner\_id, rating). The keys are user inner ids.
2. **ir**:- The items ratings. This is a dictionary containing lists of tuples of the form (user\_inner\_id, rating). The keys are item inner ids.
3. **global\_mean**:-The mean of all ratings.
4. **all\_items()**:-Generate function to iterate over all items.
5. **all\_ratings()**:-Generate function to iterate over all ratings.
6. **all\_users()**:-Generate function to iterate over all users.
7. **to\_inner\_iid(**riid**)**:-Convert an item raw id to an inner id.riid must be string.It returns inner id(int).
8. **to\_inner\_uid(**ruid**)**:-Convert an user raw id to an inner id.ruid must be string.It returns inner id(int).
9. **to\_raw\_iid(**iiid**)**:-Convert an item inner id to an raw id.iiid must be int.It returns raw id(str).
10. **to\_raw\_uid(**iuid**)**:-Convert an user inner id to a raw id.iuid must be int.It returns raw id(str).
11. **n\_users:-**Total number of users.
12. **n\_ratings:-**Total number of ratings.
13. **n\_items:-**Total number of items.

**Introduction to Tkinter**

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

There are two main methods used you the user need to remember while creating the Python application with GUI:-

1. Tk(): To create a main window, tkinter offers a method ‘Tk()’. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

m=tkinter.Tk() where m is the name of the main window object

1. mainloop(): There is a method known by the name mainloop() is used when you are ready for the application to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event till the window is not closed.

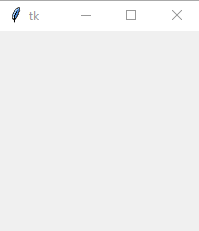
 m.mainloop()

Fig 2.4 Window appears on executing mainloop()

There are mainly three geometry manager classes class:-

1. **pack() method**:It organizes the widgets in blocks before placing in the parent widget.
2. **grid() method**:It organizes the widgets in grid (table-like structure) before placing in the parent widget.
3. **place() method**:It organizes the widgets by placing them on specific positions directed by the programmer.

There are a number of widgets which you can put in your tkinter application. Some of the major widgets are explained below:

1. **Button**:To add a button in your application, this widget is used.

The general syntax is:

w=Button(master, option=value)

master is the parameter used to represent the parent window.

Parameters:

* activebackground: to set the background color when button is under the cursor.
* activeforeground: to set the foreground color when button is under the cursor.
* bg: to set he normal background color.
* command: to call a function.
* font: to set the font on the button label.
* image: to set the image on the button.
* width: to set the width of the button.
* height: to set the height of the button.

1. **Entry**:It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used.The general syntax is:

w=Entry(master, option=value)

Parameters:

* bd: to set the border width in pixels.
* bg: to set the normal background color.
* cursor: to set the cursor used.
* width: to set the width of the button.
* height: to set the height of the button.

1. **Label**: It refers to the display box where you can put any text or image which can be updated any time as per the code.The general syntax is: w=Label(master, option=value)

Parameters:

* bg: to set he normal background color.
* bg to set he normal background color.
* font: to set the font on the button label.
* image: to set the image on the button.
* width: to set the width of the button.
* height: to set the height of the button.

1. **Text**: To edit a multi-line text and format the way it has to be displayed.The general syntax is: w =Text(master, option=value)

Parameters:

* highlightcolor: To set the color of the focus highlight when widget has to be focused.
* insertbackground: To set the background of the widget.
* bg: to set he normal background color.
* font: to set the font on the button label.
* image: to set the image on the widget.
* width: to set the width of the widget.
* height: to set the height of the widget.

1. **Listbox**: It offers a list to the user from which the user can accept any number of options.The general syntax is: w = Listbox(master, option=value)

Parameters:

* highlightcolor: To set the color of the focus highlight when widget has to be focused.
* bg: to set he normal background color.
* bd: to set the border width in pixels.
* font: to set the font on the button label.
* image: to set the image on the widget.
* width: to set the width of the widget.
* height: to set the height of the widget.

**Messagebox module:-** To show a minimalistic Tkinter message box, use the function showinfo() and showerror() where the parameters are the window title and text.

Ex:-

import tkinter

from tkinter import messagebox

root = tkinter.Tk()

root.withdraw()

messagebox.showerror("Error", "Error message")

messagebox.showwarning("Warning","Warning message")

messagebox.showinfo("Information","Informative message")

Fig 2.5 All 3 types of message boxes

**Chapter 3**

**Dataset**

We are using movielens dataset in this project which contains:-

* **ratings.csv:-**Contains ratings of 610 users for different movies they rated.It is arranged in the format userid,movieid,ratings(0.5-5),timestamp.This data is used to make the training data and testing data.We use this ratings data for collaborative filtering
* **movies.csv:-**Contains the info of 9742 movies.This dataset contains movieId,title,genres of the movie.This can be used to get moviename from movieid or movieid from moviename.

**Training Dataset**

We are building the training dataset by using build\_full\_trainset() in Dataset module of surprise package.For this,we require Dataset object to run this function.So, we use Dataset.load\_from\_file(file\_path,reader) to store ratings dataset into the dataset object.This training dataset includes the rating of user for all the movies and replacing the movie which user has not rated by \_.build\_full\_trainset() which we have used for making trainset returns Trainset class object.Trainset class attributes and methods are defined above.

**Testing Dataset**

When we are ready with the model trained on training dataset.We can find the predictions using this model by running this on testing dataset.For testing dataset,we find anti testset for each user which means this includes the movies that the user has not rated.and we will replace the ratings of this movies by global\_mean (i.e. the mean of all ratings in the dataset).

**Chapter 4**

**Implementation and User interface**

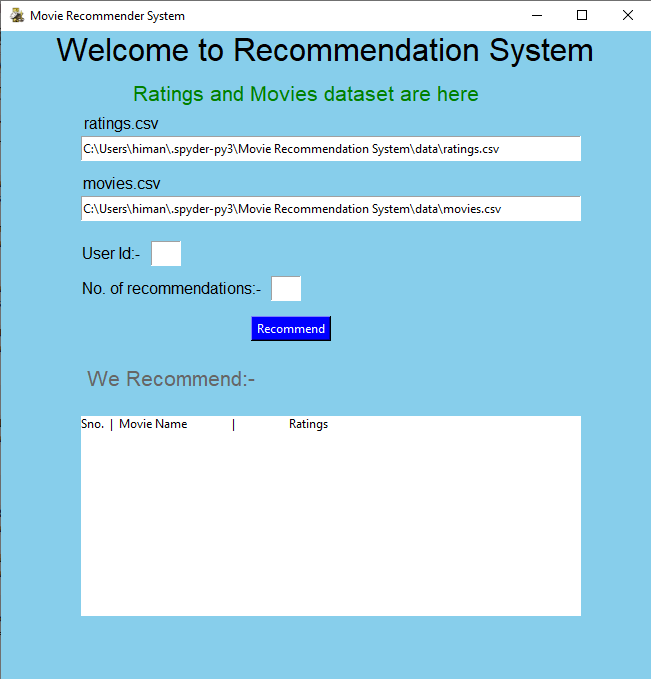
****

Fig 4.1

* When we run our project,this screen appears.
* First two entry’s in this labeled ratings.csv and movies.csv depicts where the data is stored in system.
* We have two empty fields UserId(For which user you want to compute recommendations) and No. of Recommendations.
* Then,there is Recommend button when we click it the recommendations are shown below the label We Recommend.
* Then,we have a Listbox where all the recommendations you want are shown.

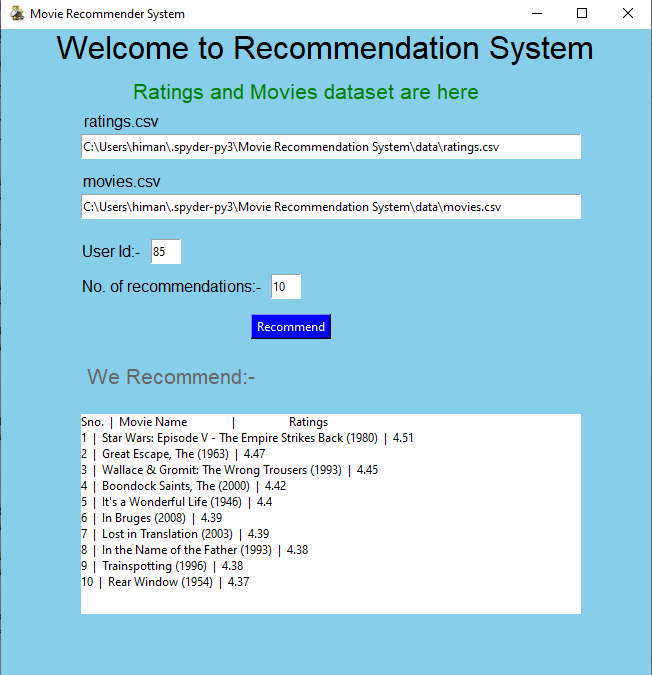
****

Fig 4.2

* In this,we have filled UserId=85 and No.of recommendations=10
* When we click on Recommend button,then 10 recommendations are shown below to the user with their estimated ratings.

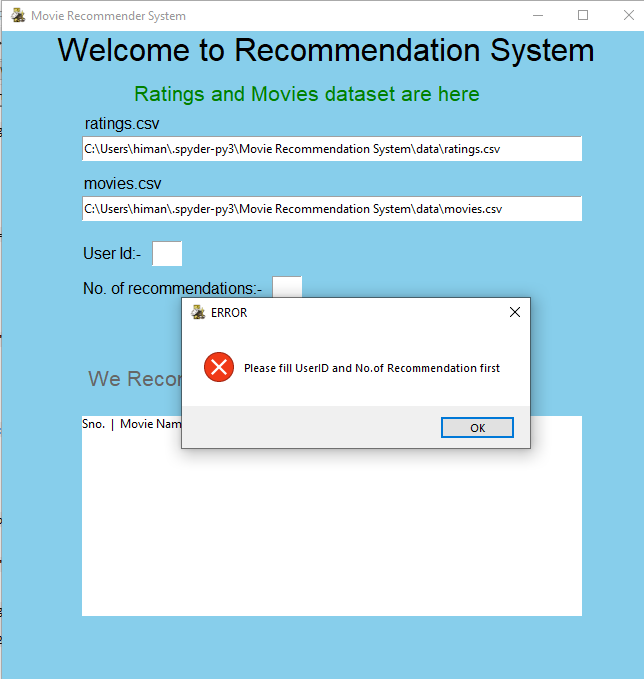
****

Fig 4.3

* If we click,on the Recommend button,without entering the UserId or No. of recommendations field,then this message box appears.
* This shows an “Error” and also tells you to Fill above fields.

**Chapter 5**

**Conclusion**

Movie Recommender systems are a powerful new technology for extracting additional value for movie websites like Netflix and Youtube from its user databases. These systems help users find movies they want to watch from a website. Movie Recommender systems benefit users by enabling them to find movies they like. Conversely, they help the websites by generating more users.Movie Recommender systems are rapidly becoming a crucial tool on the Web.Movie Recommender systems are being stressed by the huge volume of user ratings data in existing corporate databases, and will be stressed even more by the increasing volume of user ratings data available on the Web. New technologies are needed that can dramatically improve the scalability of recommender systems.

In this project,we have used Collaborative filtering approach using Single Value Decomposition(SVD) algorithm to find recommendations by simply making a training data from user ratings data and then built anti test set for user you want to find recommendations which include the movies that user not rated and ratings are set to global mean of training dataset.Then,we filter the top k recommendations that the user wanted.

In this paper we presented and experimentally evaluated a new algorithm for Collaborative Filtering-based movie recommender systems. Our approach can be further extended to other domains to recommend songs, video, venue, news, books, tourism and e-commerce sites, etc.These systems help users find items they want to buy from a business. Recommender systems benefit users by enabling them to find items they like. Conversely, they help the business by generating more sales. Recommender systems are rapidly becoming a crucial tool in E-commerce on the Web.

In the future, we plan to improve the effectiveness and performance by exploring a hybrid system(that uses Content and Collaborative both filtering) which will apply different algorithms on different user segments.

**Chapter 6**

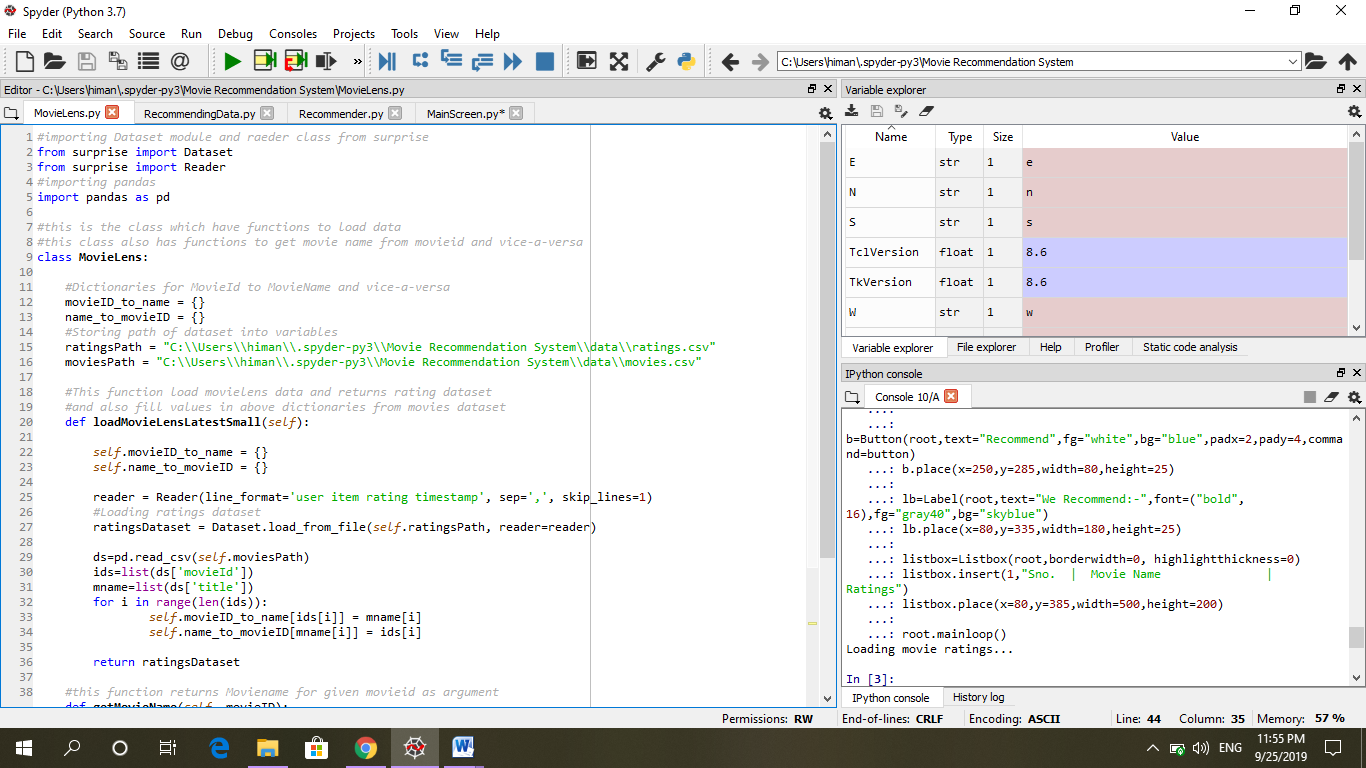
**Bibliography**

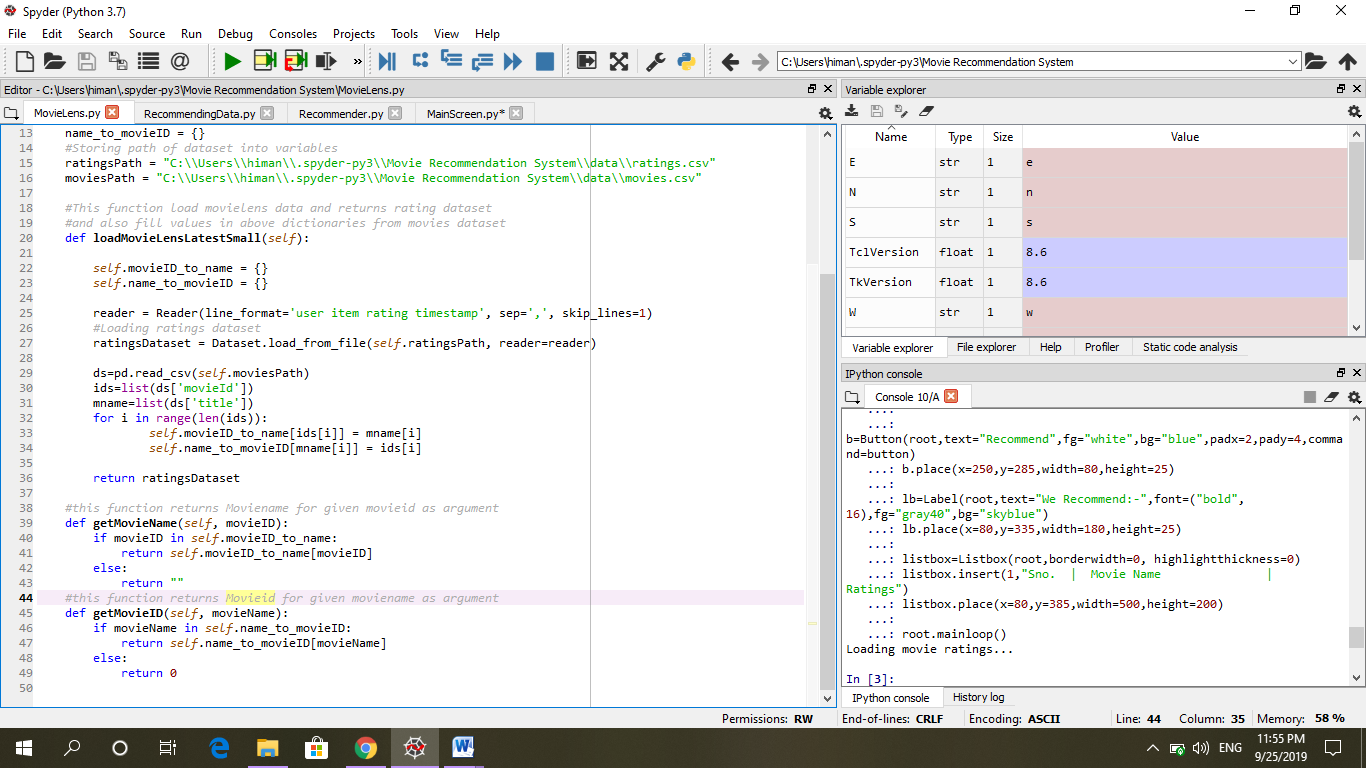
* [**geeksforgeeks.com**](http://www.geeksforgeeks.com)
* [**surprise.readthedocs.io**](http://www.surprise.readthedocs.io)
* [**towardsdatascience.com**](http://www.towardsdatascience.com)
* [**pythonspot.com**](http://www.pythonspot.com)
* **javatpoint.com**

**Chapter 7**

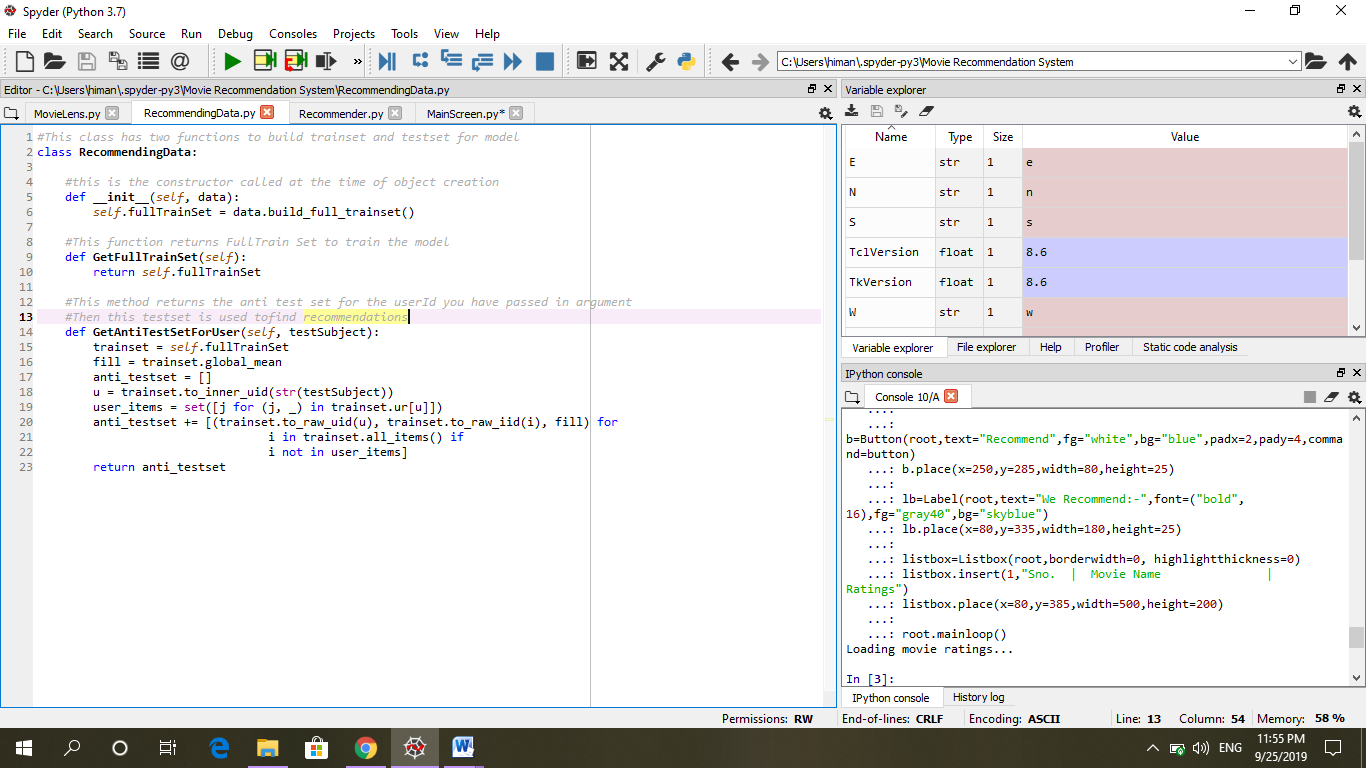
**Appendices**

**MovieLens.py**

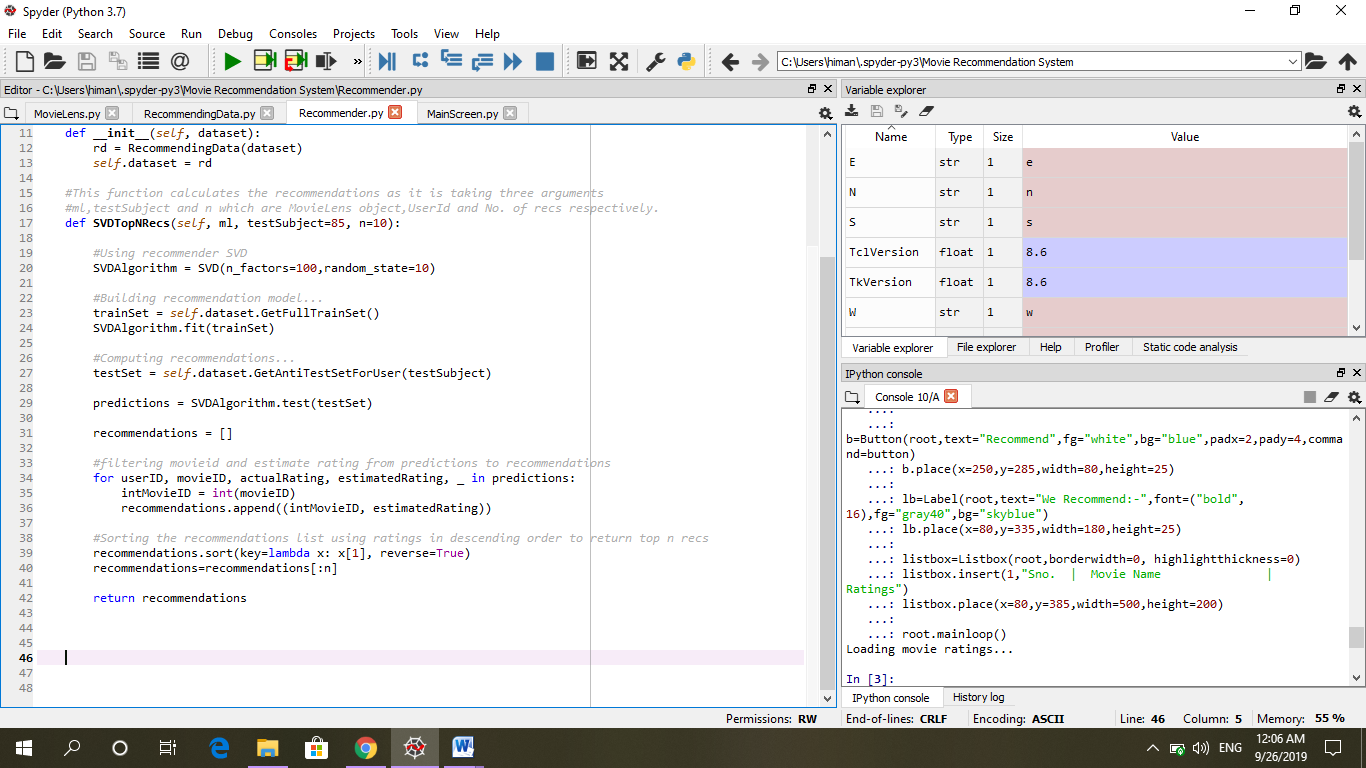
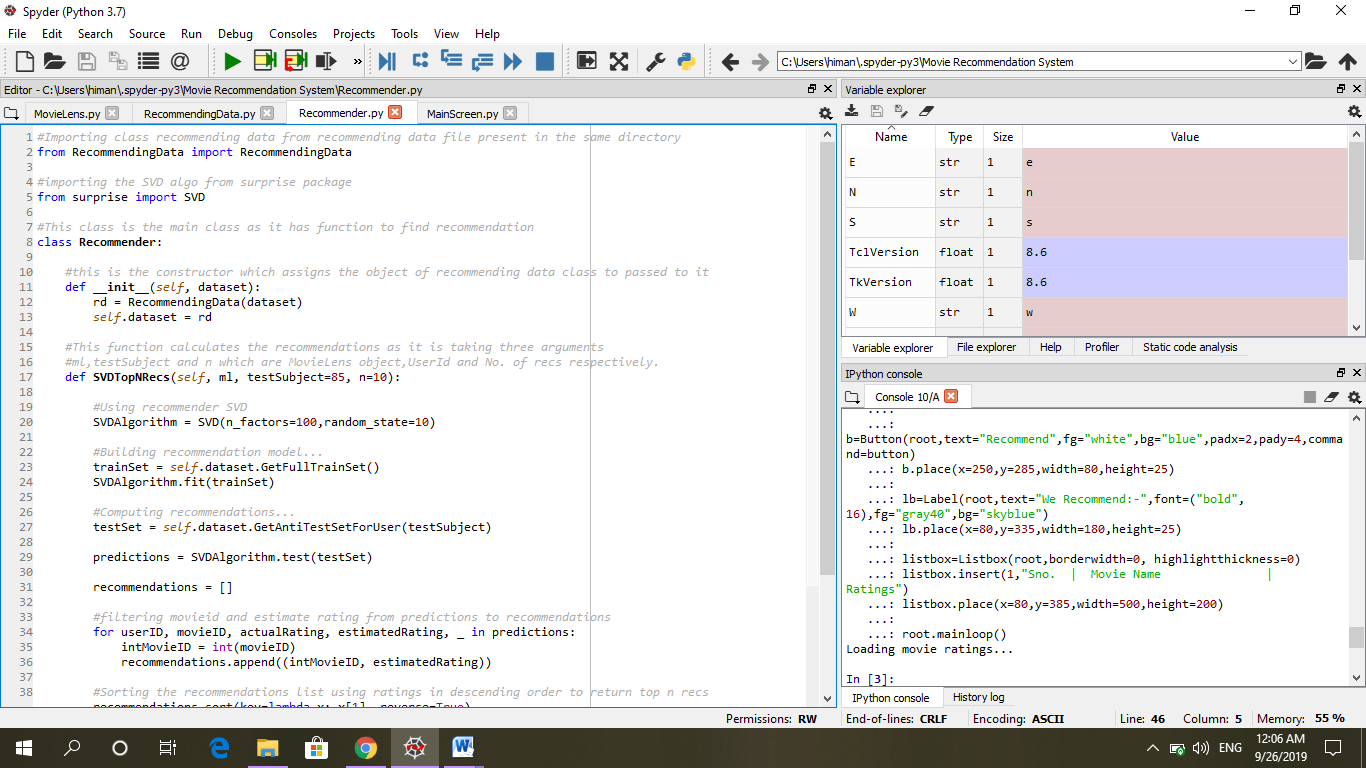
****

****

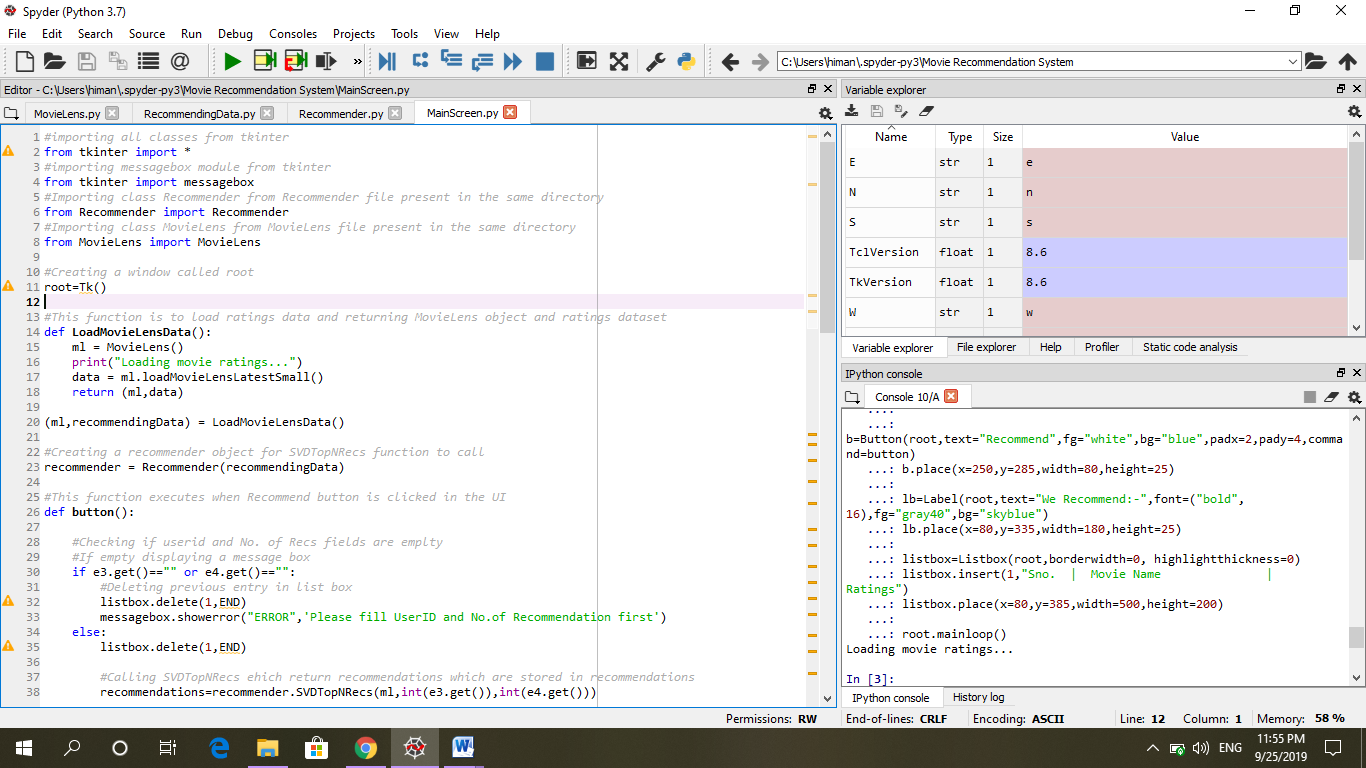
**RecommendingData.py**

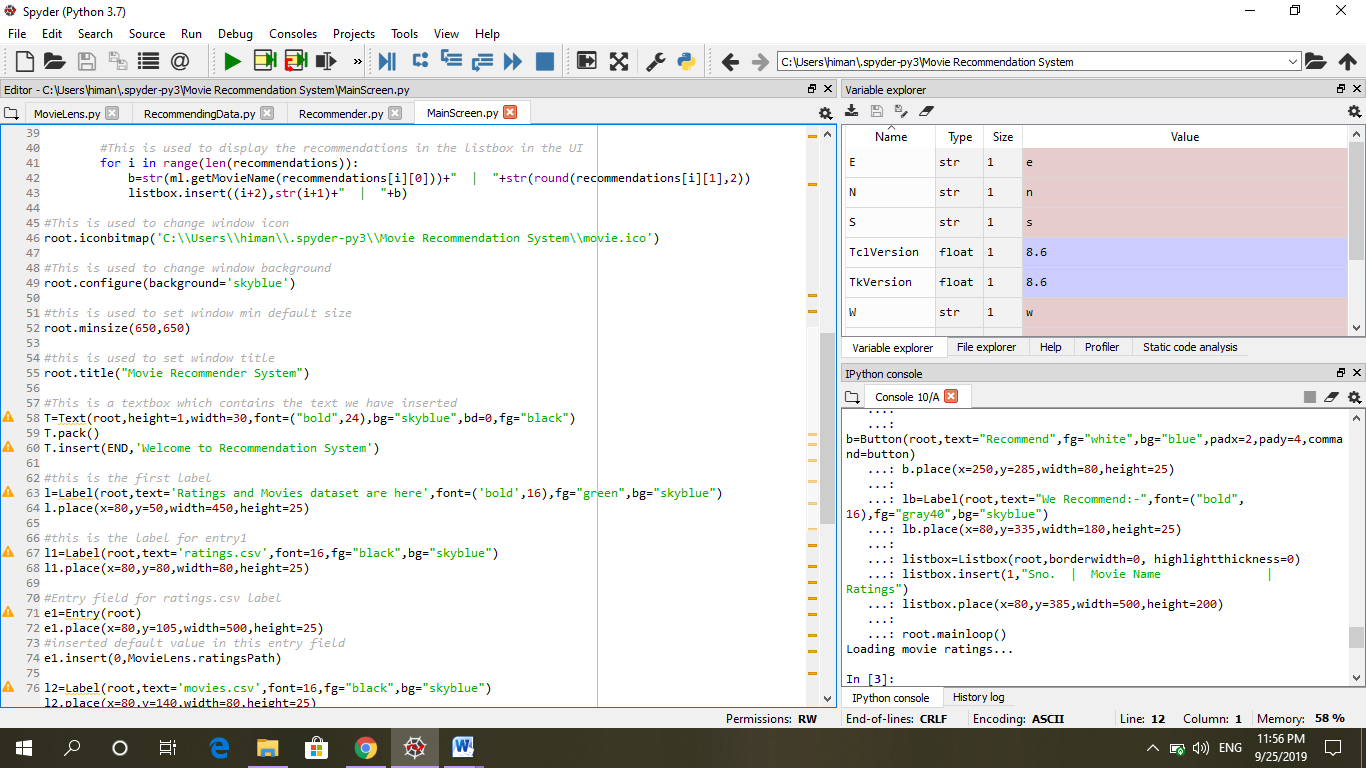
****

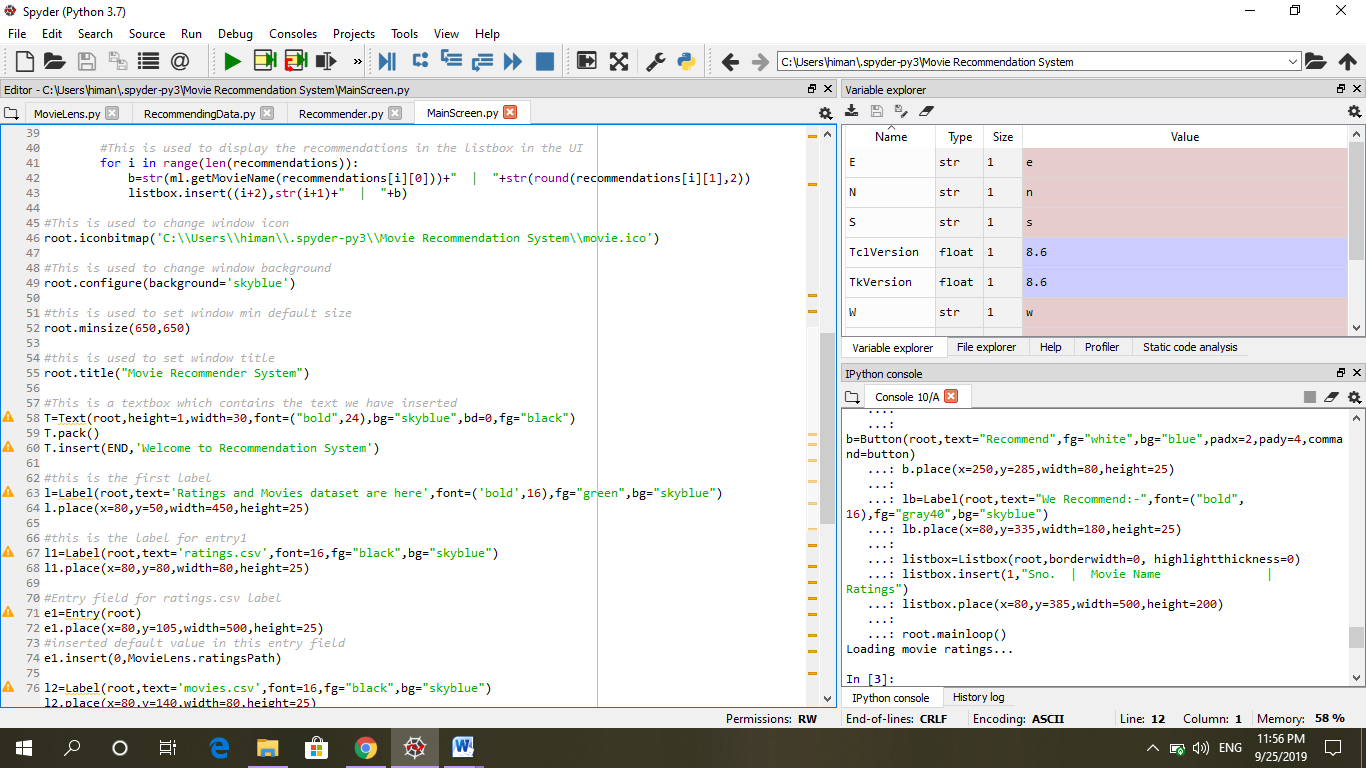
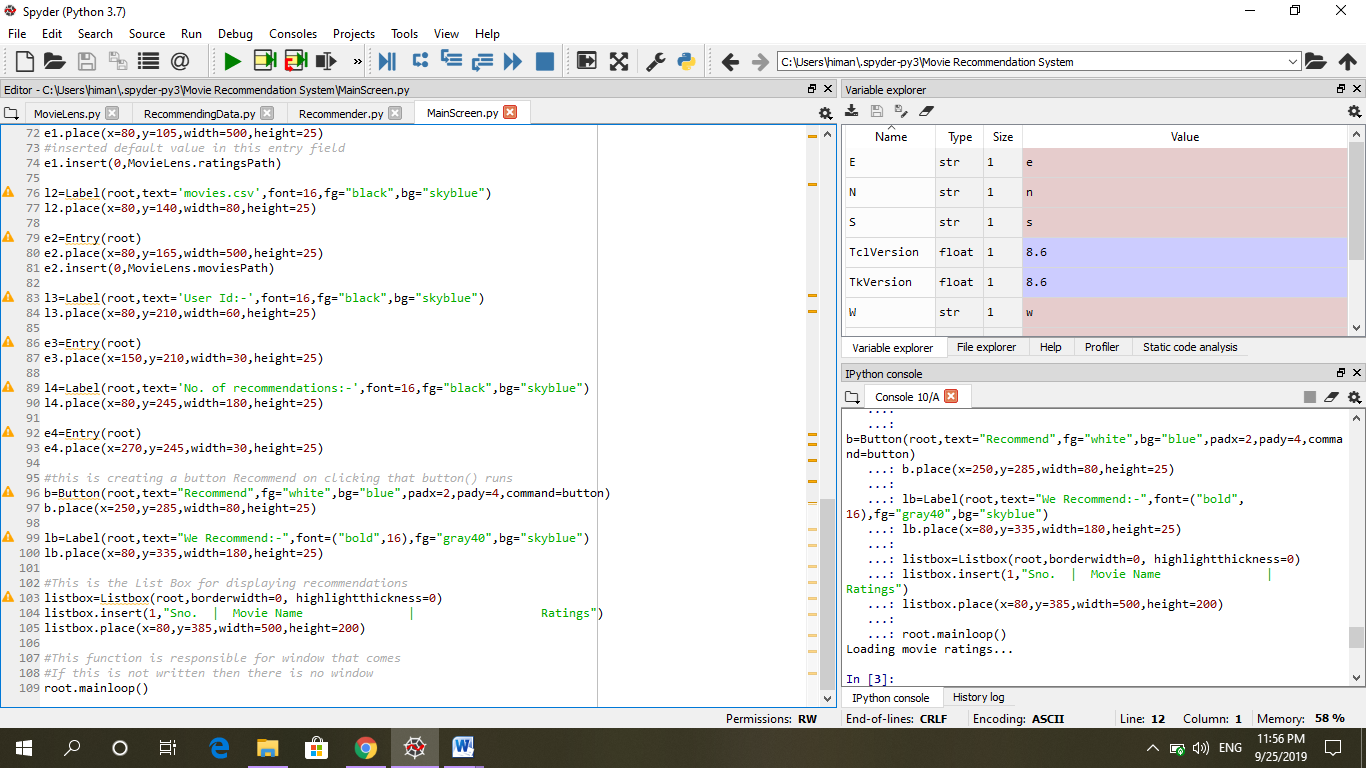
**Recommender.py**

****

**MainScreen.py**

****

****

****