**A**

**MINI PROJECT REPORT**

**ON**

**MUSIC RECOMMENDER SYSTEM**

**SUBMITTED IN THE PARTIAL FULFILLMENT FOR COMPLETION OF**

**BE-VI SEMESTER**

**IN**

**INFORMATION TECHNOLOGY**

**BY**

**PAUL SAMPSON L. (160116737097)**

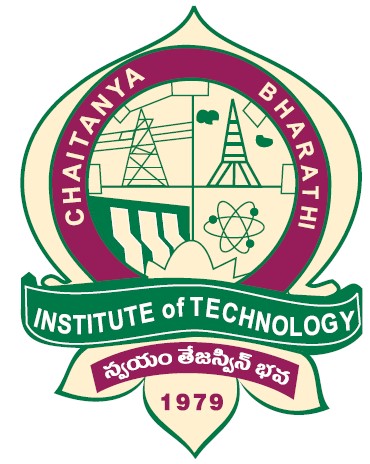
**UNDER THE GUIDANCE**

**OF**

**MR. P. VASANTH SENA**

**ASST. PROFESSOR,**

**DEPT. OF IT, CBIT.**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)**

**(AFFLIATED TO OSMANIA UNIVERSITY; AUTONOMOUS UNDER UGC, ACCREDITED BY NBA(AICTE) AND NAAC(UGC), ISO 9001:2015 CERTIFIED INSTITUTION)**

**GANDIPET, HYDERABAD – 500 075**

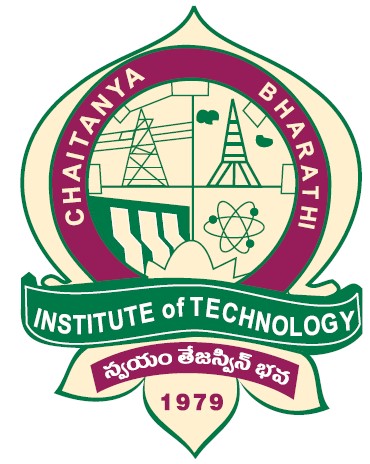
**2018-2019**

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**(Affiliated to Osmania University)**

**GANDIPET, HYDERABAD – 500 075**



**CERTIFICATE**

This is to certify that the project work entitled **“Music Recommender System”** submitted to **Chaitanya Bharathi Institute of Technology**, in partial fulfilment of the requirements for the award of degree of **B.E (Information Technology)** during the academic year 2018-2019 is a record of original work carried out by **Paul Sampson L (160116737097)** during the period of study in the Dept. of IT, CBIT, Hyderabad.

|  |  |
| --- | --- |
| Project Guide  **Mr. P Vasanth Sena**  Asst Professor,  Information Technology  CBIT, Hyd. | Head of the Department  **Dr. Suresh Pabboju**  Professor & Head of Dept.  Information Technology  CBIT, Hyd. |
|  |  |

**DECLARATION**

I declare that the project work entitled “**Music Recommender System”** is being submitted by me to the Department of Information Technology, Chaitanya Bharathi Institute of Technology (A), affiliated to Osmania University, Hyderabad is a record of bona-fide work carried out by us under the guidance and supervision of **Mr. P Vasanth Sena**, Assistant Professor, Dept. of IT, CBIT, Hyderabad.

No part of this work is copied from books/journals/internet and wherever a portion is taken, the same has been duly referred in the text. The report is based on the project work carried out entirely by us and not copied from any other source.

Paul Sampson L. (160116737097)

**ACKNOWLEDGEMENT**

I would like to express my gratitude to our guide **Mr. P Vasanth Sena**, Assistant Professor, Department of Information Technology, Chaitanya Bharathi Institute of Technology, for his kind co-operation and encouragement which help us in completion of this project.

I am highly indebted to **Dr. P. Ravinder Reddy**, The Principal, Chaitanya Bharathi Institute of Technology, Hyderabad and **Dr. Suresh Pabboju**, Head of Department, Information Technology, Chaitanya Bharathi Institute of Technology, for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I have taken efforts in this project. However, it would not have been possible without the kind support and help of Teaching Staff, Non-Teaching Staff and organizations. I would like to extend my sincere thanks to all of them.

I would like to express my special gratitude and thanks to industry persons for giving us such attention and time.

My thanks and appreciations also go to our Parents in developing the project and people who have willingly helped me out with their abilities.

Paul Sampson L. **(160116737097)**

# 

# ABSTRACT

When it comes to music, gone are the days you had to buy a record to listen to the favourite band you liked. Most people nowadays prefer to listen to their favourite music on music streaming services- Spotify, Amazon Prime Music, Apple Music, Pandora, etc. These different music services together cover various countries and regions throughout the world. Nowadays music streaming services have increasingly become common and can be accessed through a variety of devices like Windows, MacOS, Linux, iOS, Windows Phone, Android as well as in smart-home devices like smart TV’s, home consoles, smart-speakers, etc.

Taking into consideration the large number of users who stream music online, we can get loads of data on the music that people play on their devices. Tapping into this data and analysing it using the modern data mining techniques, gives a chance to provide a curated list of playlist preferences for a person. This can simply be done by running the history of the songs played by that person through the recommender model. The model comprises of two recommendation methods- one used as control and one is the recommendation model proposed. The control model gives the list of most popular songs, while the recommendation model using the collaborative filtering technique to recommend the songs in the playlist. I will be using the Jaccard Similarity Measure, which is useful for making effective predictions.

# CONTENTS

[ABSTRACT v](#_Toc4106618)

[CONTENTS vi](#_Toc4106619)

[1. Introduction 1](#_Toc4106620)

[1.1. Overview 1](#_Toc4106621)

[1.2. Motivation 1](#_Toc4106622)

[2. Technologies 2](#_Toc4106623)

[2.1 Python 2](#_Toc4106624)

[2.1.1. Introduction 2](#_Toc4106625)

[2.1.2. Variables 2](#_Toc4106626)

[2.1.3. Indentation 3](#_Toc4106627)

[2.1.4. User Defined Functions 3](#_Toc4106628)

[2.2. Pandas, Numpy, Matplotlib 5](#_Toc4106629)

[2.3.Item based Collaborative Filtering 8](#_Toc4106630)

[2.3.1.Similarity between items 8](#_Toc4106631)

[2.3.2. Similarity Measure 8](#_Toc4106632)

[3. Software Requirement Specification 10](#_Toc4106638)

[3.1. Introduction 10](#_Toc4106639)

[3.2. Software and Hardware Requirements 10](#_Toc4106640)

[3.2.1. Sublime Text 3: 10](#_Toc4106641)

[3.2.2. Spyder IDE: 11](#_Toc4106642)

[3.2.3. Jupyter Notebook: 12](#_Toc4106643)

[3.2.4. Google Colaboratory 13](#_Toc4106644)

[4. Implementation 14](#_Toc4106645)

[4.1. Introduction 14](#_Toc4106646)

[4.2. The Dataset 14](#_Toc4106647)

[4.3. Preprocessing 14](#_Toc4106648)

[4.4. Model 14](#_Toc4106649)

[5. Results 1](#_Toc4106650)5

[5.1.Introduction 15](#_Toc4106651)

[5.2.Output Screens 15](#_Toc4106652)

[6. Conclusion and Future Scope 1](#_Toc4106653)7

[Bibliography 18](#_Toc4106654)

**LIST OF FIGURES**

Fig 2.2.1. Pandas 5

Fig 2.2.2. Numpy 6

Fig.2.2.3. Matplotlib 7

Fig.2.3.1. Item-item similarity 8

Fig. 3.2.1.1 Sublime Text 3 11

Fig. 3.2.3.1 Jupyter Notebook 13

Fig: 5.2.1. Popularity based recommendation 15

Fig: 5.2.1. Creating Item-Similarity 15

Fig: 5.2.3. Item-Similarity ranking 16

Fig: 5.2.4. Comparing Popularity and Item- similarity models 16

## 1. Introduction

### 1.1. Overview

Music Recommendation in various streaming services is done by a popularity recommendation of the current trending songs. However, to improve the quality of service, it is essential to customize songs to the users’ preference. This can be done by a user-item similarity approach or an item-item similarity approach. This helps a streaming service to particularly note users’ preferences and customize songs to him/her for every mood.

### 1.2. Motivation

Emotions are best conveyed by symphony. This adage is proved by the fact that over the years, with improving technology people have become reliant on their favorite music as means of relaxing and escaping the stressful conditions of life. Music is not something that people listen to but feel. Getting the right music to an individual at any given time is tough for any algorithm to predict because of the behavioral patterns of human beings. This recommender system is by no means the greatest and the end of all recommender systems, but certainly a step in the right direction.

# 2. Technologies

## 2.1 **Python**

Python is a dynamic, interpreted (bytecode-compiled) language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and you lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs.

### 2.1.1. Introduction

Python source files use the ".py" extension and are called "modules." With a Python module hello.py, the easiest way to run it is with the shell command "python hello.py Alice" which calls the Python interpreter to execute the code in hello.py, passing it the command line argument "Alice".

#!/usr/bin/env python

# import modules used here -- sys is a very standard one

import sys

# Gather our code in a main() function

def main():

print 'Hello there', sys.argv[1]

# Command line args are in sys.argv[1], sys.argv[2] ...

# sys.argv[0] is the script name itself and can be ignored

# Standard boilerplate to call the main() function to begin

# the program.

if \_\_name\_\_ == '\_\_main\_\_': main()

### 2.1.2. Variables

Since Python variables don't have any type spelled out in the source code, it's extra helpful to give meaningful names to your variables to remind yourself of what's going on. So use "name" if it's a single name, and "names" if it's a list of names, and "tuples" if it's a list of tuples. Many basic Python errors result from forgetting what type of value is in each variable, so use your variable names (all you have really) to help keep things straight.

As far as actual naming goes, some languages prefer underscored\_parts for variable names made up of "more than one word," but other languages prefer camelCasing. In general, Python prefers the underscore method but guides developers to defer to camelCasing if integrating into existing Python code that already uses that style. Readability counts.

### 2.1.3. Indentation

One unusual Python feature is that the whitespace indentation of a piece of code affects its meaning. A logical block of statements such as the ones that make up a function should all have the same indentation, set in from the indentation of their parent function or "if" or whatever. If one of the lines in a group has a different indentation, it is flagged as a syntax error.Python's use of whitespace feels a little strange at first, but it's logical and I found I got used to it very quickly. Avoid using TABs as they greatly complicate the indentation scheme (not to mention TABs may mean different things on different platforms). Set your editor to insert spaces instead of TABs for Python code.

### 2.1.4. User Defined Functions

Functions in Python are defined like this:

# Defines a "repeat" function that takes 2 arguments.

def repeat(s, exclaim):

result = s + s + s # can also use "s \* 3" which is faster (Why?)

if exclaim:

result = result + '!!!'

return result

Notice also how the lines that make up the function or if-statement are grouped by all having the same level of indentation. We also presented 2 different ways to repeat strings, using the + operator which is more user-friendly, but \* also works because it's Python's "repeat" operator, meaning that '-' \* 10 gives '----------', a neat way to create an onscreen "line." In the code comment, we hinted that \* works faster than +, the reason being that \* calculates the size of the resulting object once whereas with +, that calculation is made each time + is called. Both + and \* are called "overloaded" operators because they mean different things for numbers vs. for strings (and other data types).

The def keyword defines the function with its parameters within parentheses and its code indented. The first line of a function can be a documentation string ("docstring") that describes what the function does. The docstring can be a single line, or a multi-line description as in the example above. (Yes, those are "triple quotes," a feature unique to Python!) Variables defined in the function are local to that function, so the "result" in the above function is separate from a "result" variable in another function. The return statement can take an argument, in which case that is the value returned to the caller.

Here is code that calls the above repeat() function, printing what it returns:

def main():

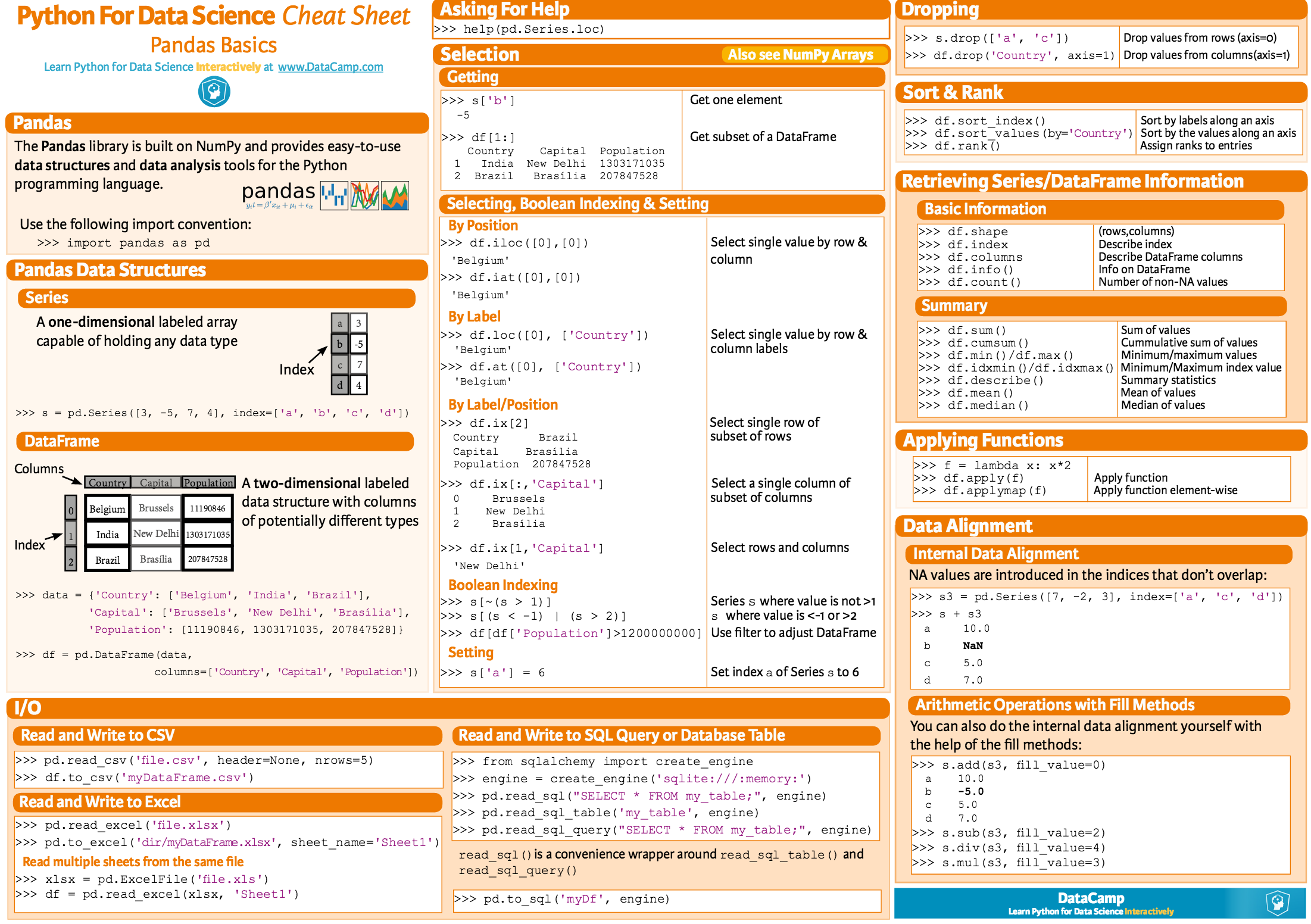
print repeat('Yay', False) ## YayYayYay

print repeat('Woo Hoo', True) ## Woo HooWoo HooWoo Hoo!!!

At run time, functions must be defined by the execution of a "def" before they are called. It's typical to def a main() function towards the bottom of the file with the functions it calls above it.

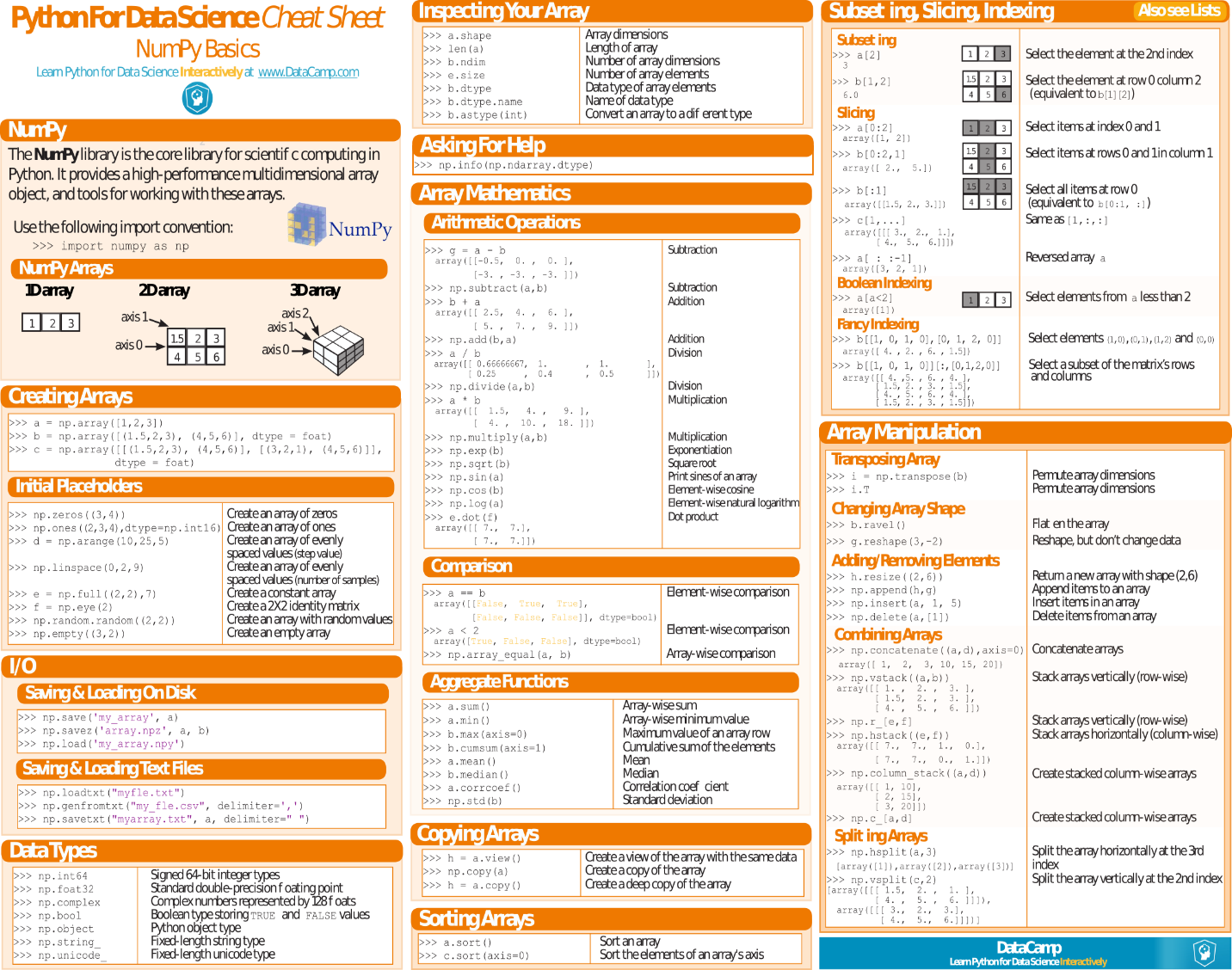
## 2.2. Pandas, Numpy, Matplotlib

**Pandas**: In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.



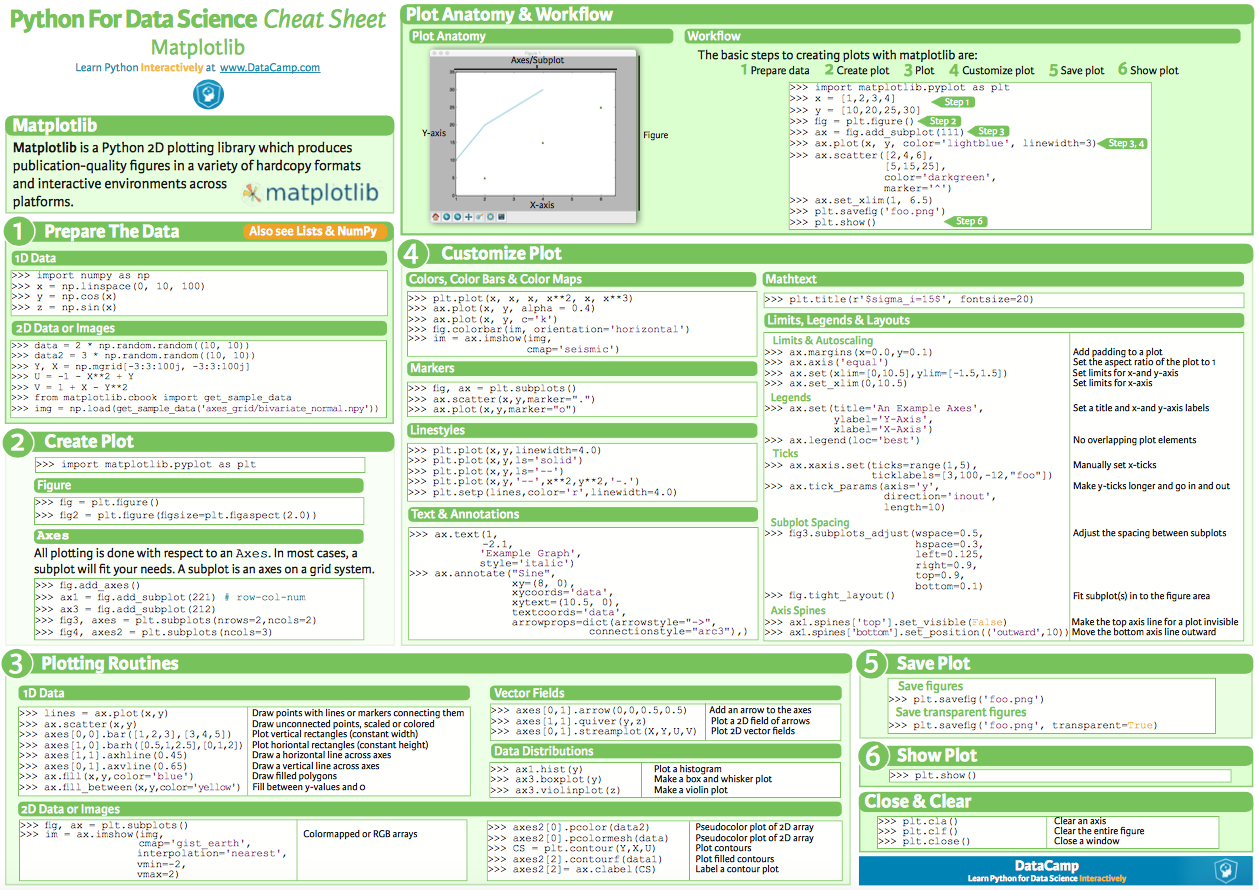
#### Fig 2.2.1. Pandas

**Numpy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays**.**



#### Fig 2.2.2. Numpy

**Matplotlib:** Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

.

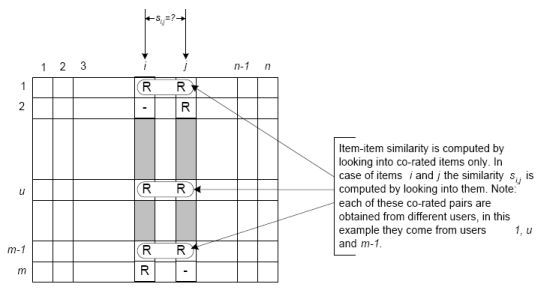
#### Fig.2.2.3. Matplotlib

## 2.3. Item Based Collaborative Filtering

**Item-based collaborative filtering** is a model-based algorithm for making recommendations. In the algorithm, the similarities between different items in the dataset are calculated by using one of several similarity measures, and then these similarity values are used to predict ratings for user-item pairs not present in the dataset.

### 2.3.1. Similarities between items

The similarity values between items are measured by observing **all the users who have rated both the items**. As shown in the diagram below, the similarity between two items is dependent upon the ratings given to the items by users who have rated both:



#### Fig.2.3.1. Item-item similarity

### 2.3.2. Similarity Measure

There are several different mathematical formulations that can be used to calculate the similarity between two items. In the project we’ve used the Jaccard Similarity measure:

**Jaccard similarity:** Jaccard Similarity also known as Jaccard index is a measure to find similarities between *sets*. So first, let’s learn the very basics of sets.

**Sets:**A set is (unordered) collection of objects {a,b,c}. we use the notation as elements separated by commas inside curly brackets { }. They are unordered so {a,b} = {b,a}.

**Cardinality:**The Cardinality of A (denoted by **|A|**) counts how many elements are in A.

**Intersection:**An intersection between two sets A and B is denoted**A ∩ B** and reveals all items which are in both sets A, B.

**Union:**Union between two sets A and B is denoted **A ∪ B** and reveals all items which are in either set.

The Jaccard similarity measures similarity between finite sample sets and is defined as the cardinality of the intersection of sets divided by the cardinality of the union of the sample sets. Suppose you want to find Jaccard similarity between two sets A and B it is the ratio of the cardinality of A ∩ B and A ∪ B.

Jaccard Index = (the number in both sets) / (the number in either set) \* 100

= |A∩B|/|A∪B|

# 3. Software Requirement Specification

## 3.1. Introduction

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirements analysis process it lists the requirements of a particular software system including functional, performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective. The purpose of software requirements specification is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the project and its functionality.

## 3.2. Software and Hardware Requirements

|  |  |
| --- | --- |
| Operating System | Windows, Unix |
| Programming Language | Python3 |
| Processor | Intel(R) i7 Quad Core CPU |
| RAM | 8 GB or more |
| GPU | Nvidia GPU with 4 GB or more VRAM  (1050Ti and above) |

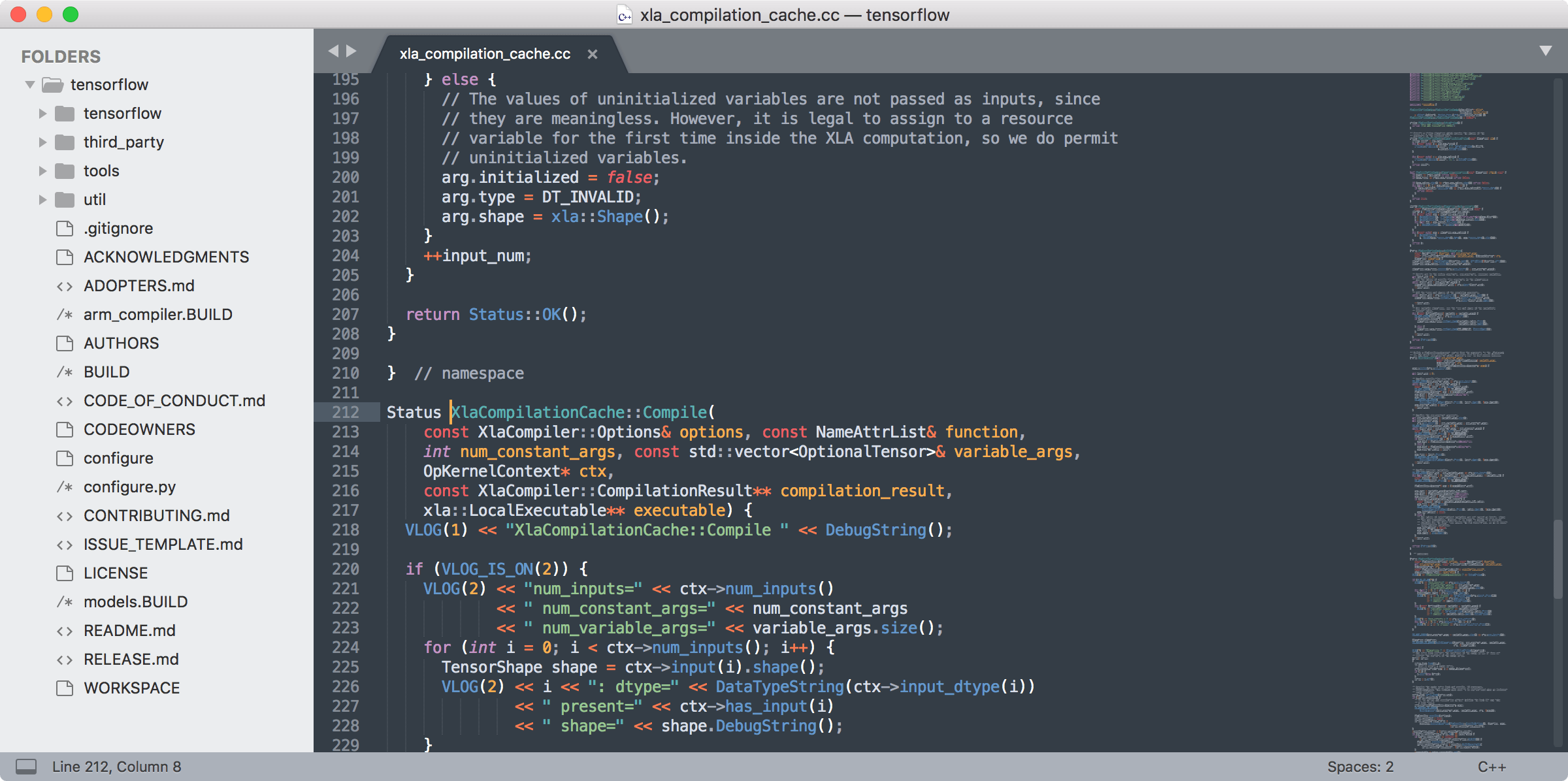
### 3.2.1. Sublime Text 3:

Sublime Text is a proprietary cross-platform source code editor with a Python application programming interface (API). It natively supports many programming languages and markup languages, and functions can be added by users with plugins, typically community-built and maintained under free-software licenses.

Features:

The following is a list of features of Sublime Text:

* "Command palette" uses adaptive matching for quick keyboard invocation of arbitrary commands
* Simultaneous editing: simultaneously make the same interactive changes to multiple selected areas
* Python-based plugin API
* Project-specific preferences
* Extensive customizability via JSON settings files, including project-specific and platform-specific settings
* Cross-platform (Windows, macOS, and Linux) and Supportive Plugins for cross-platform.
* Compatible with many language grammars from TextMate

****

#### Fig. 3.2.1.1 Sublime Text 3

### 3.2.2. Spyder IDE:

Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open source software. It is released under the MIT license.

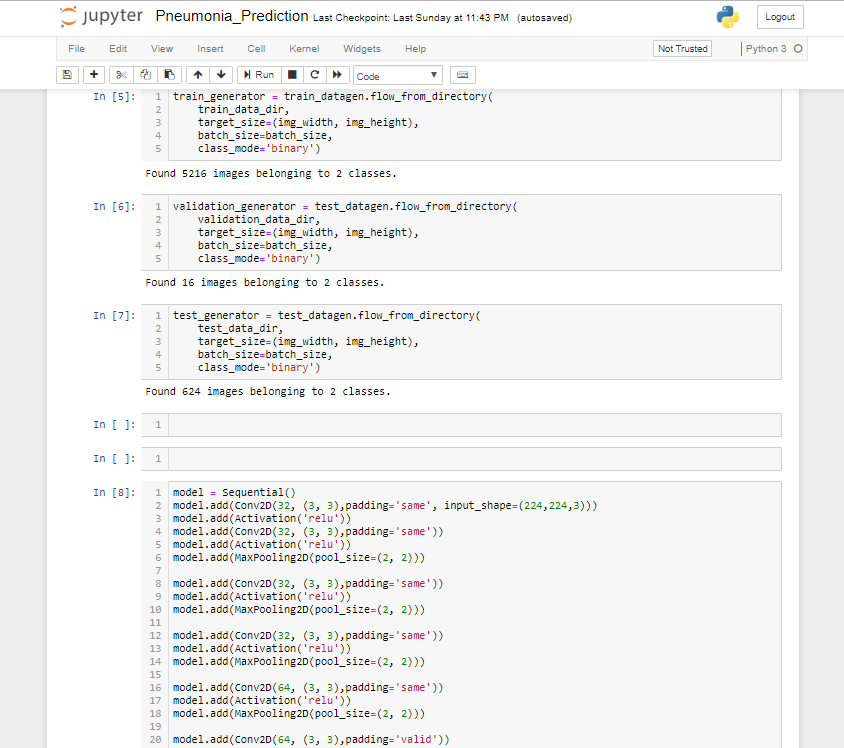
Features include:

* An editor with syntax highlighting, introspection, code completion
* Support for multiple IPython consoles
* The ability to explore and edit variables from a GUI
* A Help pane able to retrieve and render rich text documentation on functions, classes and methods automatically or on-demand
* Static code analysis, powered by Pylint
* A run-time Profiler, to benchmark code
* Project support, allowing work on multiple development efforts simultaneously
* A built-in file explorer, for interacting with the filesystem and managing projects
* A "Find in Files" feature, allowing full regular expression search over a specified scope
* An online help browser, allowing users to search and view Python and package documentation inside the IDE
* A history log, recording every user command entered in each console

### 3.2.3. Jupyter Notebook:

Jupyter Notebook (Formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebooks documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

Jupyter notebooks document can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through 'Download As' in the web interface, via the nbconvert library or 'jupyter nbconvert' command line interface in a shell.



#### Fig. 3.2.3.1 Jupyter Notebook

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the file and display it to the user.

## 3.2.4. Google Colaboratory

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.

Colab provides GPUs and TPUs to run and execute Deep learning models using google cloud services for free.

# 4. Implementation

## 4.1. Introduction

The success of the software product is determined only when it is successfully implemented according to the requirements. The analysis and the design of the proposed system provide a perfect platform to implement the idea using the specified technology in the desired environment. The implementation of our system is made user friendly.

Any software project is designed in modules and the project is said to be successfully implemented when each of the module is executed individually to obtain the expected result and, when all the modules are integrated and run together without any errors.

## 4.2. The Dataset

The dataset we used to train is called the Million Songs Dataset. It is taken from Labrosa Columbia (https://labrosa.ee.columbia.edu/millionsong/). It consists of various attributes relating to the different songs in the system. The second dataset is (https://static.turi.com/datasets/millionsong/10000.txt) gives the songs and the number of clicks by users.

## 4.3. Preprocessing

The songs from both the datasets are merged and each user’s songs and the number of clicks is stored in a matrix called cooccurrence matrix. This is used to perform the item similarity on it, to give the final recommendations.

## 4.4. Model

The Co-occurrence matrix is then subjected to the python set functions, union and intersection to update the values in the matrix. The songs are then given ranks and sorted in a list which is subsequently printed.

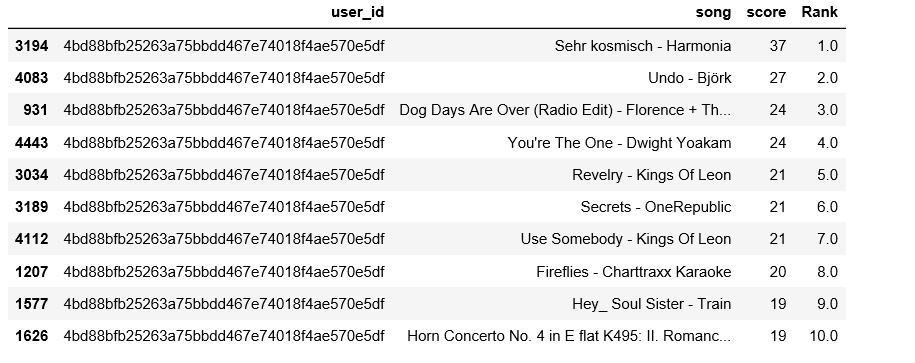
## 5. **Results**

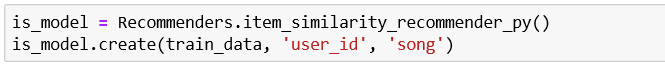
## 5.1. Introduction

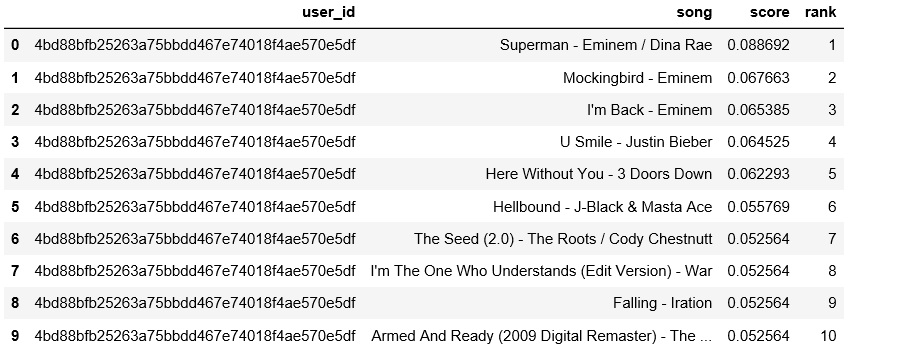
Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

## 5.2. Output Screens

Fig:5.2.1. Popularity based recommendation

Fig:5.2.2. Creating item similarity model

 Fig:5.2.3. Item-Similarity ranking

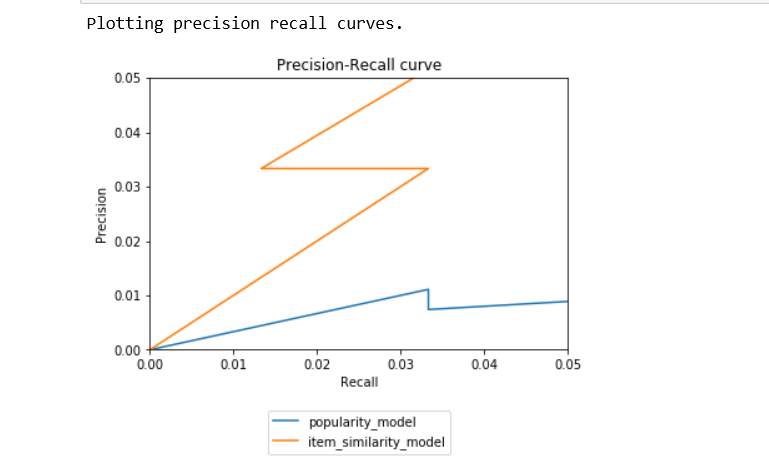


Fig:5.2.4. Comparing Popularity and Item- similarity models

# 6. Conclusion and Future Scope

The purpose of this project is to recommend a music playlist to a user. This Model can help streaming services to provide quality of service to the users by playing their favorite songs. The Model can be trained with more data to get more accurate results, thereby providing even better customized songs keeping up with the songs being released very day.

**FUTURE SCOPE**

This Model can be further developed in the following ways:

Can be used to make theme-based playlist which can cater to every moment and mood of the users.

Can be used with VR and newer technologies to create virtual music streaming with video that can provide an immersive live musical experience.

Can connect people with similar music preferences together in a community. This allows the music communities to grow and expand their tastes with new songs being released.

Record production companies, distributors to personalize music to the individual taste and preference, thus can directly advertise for their prospective communities who share the same musical song preferences.

Music preferences can be used in Behavioral Analysis, which can be used in state surveillance or advertisement analysis.

# Bibliography

# 

* <https://towardsdatascience.com/build-your-own-recommendation-engine-netflix-demystified-demo-code-550401d4885e>
* http://inpressco.com/wp-content/uploads/2014/09/Paper73131-3133.pdf
* https://www.statisticshowto.datasciencecentral.com/jaccard-index/
* https://beckernick.github.io/music\_recommender/
* <https://www.youtube.com/watch?v=h9gpufJFF-0>
* <https://www.numpy.in/documention/>
* <http://docs.python-requests.org/en/master/>