ABSTRACT

An Early prediction of heart disease using Machine Learning algorithm using decision support systems. Dataset is collected from the UCI machine learning repository. Main aim of this project is to analyze the attributes and predict the presence of disease effectively. To provide more accuracy for presence of heart disease by using important attributes in order to reduce the number of tests which helps in achieving a cost-effective and time-efficient process. Considering Cleveland, Hungarian, Switzerland heart disease dataset with 14 attributes and 700+ instances for classification along closely related relationships in large datasets. Applying algorithms like Naive Bayes, Random forest and K-nearest neighbor and to compare and contrast them for more accuracy and for superior performance over conventional classification.

CHAPTER 1

INTRODUCTION

Dealing with the enormous amount of recruiting information on the Internet, a job seeker always spends hours to find useful ones. To reduce this laborious work, we design and implement a recommendation models for online job-hunting. Data mining techniques are used to gain the insights present in the datasets as a knowledge base. Data mining deals with digging through huge data to discover the hidden features and links between the attributes. It turns raw data into useful information, businesses can learn more and develop more effective marketing strategies. This includes the convolution between the discipline statistics, Artificial intelligence and machine learning.

The dataset for the jobs is generated by performing web scraping on the online job portal to retrieve the useful and required attributes from the job post and store it in the local system in the csv format. The characteristics of the datasets are summarized by performing exploratory data analysis.

1.1PROBLEM STATEMENT

On the Internet, where the number of choices is overwhelming, there is need to filter, prioritize and efficiently deliver relevant information in order to alleviate the problem of information overload, which has created a potential problem to many Internet users.

The most challenging and time-consuming task is to collate the information and find out most relevant user-job connection mapping according to the skills and preferences of a user. Thus a relevant job recommendations would ease the effort of diverse range of job seekers .This would reduce the time spent by each user in searching and applying for a potential and more relevant job online.

1.2TECHNOLOGIES

1.2.1 Machine Learning

Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data-such algorithms overcome following strictly static program instructions by making data driven predictions or decisions through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible.

Machine learning is closely related to computational statistics, which also focuses in prediction-making through the use of computers. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioural profiles for various entities and then used to find meaning anomalies.

Within the field of data analytics, machine learning is a method to devise complex models and algorithms that lend themselves to prediction also known as predictive analytics. Instead of hard coding software routines with specific instructions to accomplish a particular task, machine learning is a way of “training” an algorithm which involves feeding huge amounts of data to the algorithm and allowing the algorithm to adjust itself and improve. These analytical models allow researchers, data scientists, engineers, and analysts to “produce reliable, repeatable decisions and results” and uncover “hidden insights” through learning from historical relationships and trends in data.

1.2.2 Python

Python is a general-purpose interpreted, interactive, objective, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language. Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keyword frequently whereas other languages use punctuations.

Python is Interpreted- Python is processed at runtime by the interpreter. You do not need compiler your before executing it. This is similar to PERL and PHP.

Python is Interpreted- you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented-Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a beginner language- Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text to WWW browsers to games.

1.2.2.1 Python Features

Easy-to-learn- Python has few keywords, simple structure and a clearly defined syntax. This allows the student to pick up the language quickly.

Easy-to-read- Python code is more clearly defined and visible to the eyes.

Easy-to-maintain- Python and its source code is fairly easy-to-maintain.

A broad standard library- Python bulk of the library is very portable and cross-platform compatible on UNIX, Windows and Macintosh.

Interactive Mode- Python has a support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable- Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Extendable- You can add low-level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

Databases- Python provides interfaces to all major commercial databases.

GUI Programming- Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems such as Windows MFC, Macintosh and the X Window system of Unix.

Scalable- Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, python has a big list of good features, few are listed below.

It supports functional and structured programming methods as well as OOP.

It can be used as a scripting language or can be compiled byte-code for building large applications.

It provides very-high dynamic data types and supports dynamic type checking.

It supports automatic garbage collection.

It can be easily integrated with C, C++, COM, ActiveX, CORBA and Java.

1.2.3 Anaconda Tool

Anaconda is a FREE enterprise-ready Python distribution for data analytics, processing and scientific computing. Anaconda comes with Python 2.7 or Python 3.4 and 100+ cross-platforms tested and optimised Python packages. All of the usual Python ecosystem tools work with Anaconda.

Additionally, Anaconda can create custom environments that mix and match different Python versions (2.6, 2.7, 3.3, 3.4) and other packages into isolated environments and easily switch between them using CONDA, our innovative multi-platform package manager for Python and other languages.

1.2.3.1 Using Python in Anaconda

Many people write Python code using a text editor like Emacs or Vim. Others prefer to use an IDE Spyder, Wing IDE, PyCharm or Python Tools for Visual Studio.Jupyter Notebook is a free IDE included with Anaconda. To start Jupyter Notebook, type the name Jupyter Notebook in the Anaconda Prompt.

1.2.3.2 Jupyter Notebook

The Jupyter Notebook is an interactive computing environment that enables users to author notebook documents that include: - Live code - Interactive widgets – plots – Narrative text – Equations – Images – Videos

These documents provide a complete and self-contained record of a computation that can be converted to various formats and shared with others using email, Dropbox, version control systems (like git/GitHub) or nbviewer.jupyter.org.

CHAPTER 2

LITERATURE SURVEY

1.Context-Based Collaborative Filtering for Citation Recommendation

Authors: Haifeng Liu, Xiangjie Kong, Xiaomi Bai, Wei Wang, Teshome Megersa Bekele, Feng Xia

This paper suggests the reference papers for a particular research paper sice research papers are rapidly published in various conference and journals. Contest-based Collaborative filtering computes similarities between citing papers by comparing pairwise paper representations which are obtained from citation context. The rationale underlying this similarity calculation is that, citing papers are considered to be similar if they are co-occurred with the same citing papers. each citing paper, we randomly select cited papers into the test set at the ratio of 20%. We employed three commonly used evaluation metrics in our experiments that is Precision, is the ratio of the number of cited papers in the top-N recommendation list to the length of the same list. Recall, is the ratio of the number of cited papers in the top-N recommendation list to the total number of all cited articles.It shows the comparison results of Baseline and CCF (when *ts* equals 0.4) on HEP-PH dataset. To perform a parameter study the function of the threshold *ts* is to determine whether two citing papers are co-occurred or not. If their association degree is beyond *ts*, they are considered to be co-occurred, and vice versa.

2.Scraping and Clustering Techniques for Linkedin Jobs

Authors: Kais Dai ,Ana Fernández Vilas ,Rebeca P. Díaz Redondo

LinkedIn is the most popular OSN for professionals with more than 300 million subscribers as of the year 2014. The amount and quality of the data produced by this online social network is very large. The primary contribution of this paper is obtaining an anonymized dataset by scrapping the public profile of LinkedIn users subject to the terms of LinkedIn Privacy Policy. LinkedIn public profile refers to the user's information that appears in a Web Page when the browser is external to LinkedIn (not logged in). The main contributions of this paper can be summarized as obtaining a LinkedIn dataset, which does not exist to our knowledge and perform exploratory analysis to uncover educational categories and their relative appearance in LinkedIn and professional clusters in LinkedIn by grouping profiles according to the free summaries provided by LinkedIn users.

3. Exploratory data analysis

Authors: Chris Chatfield

The paper provides guidelines on from where to start the analysis. Start by assessing the structure of the data, as the analysis will depend crucially, not only on the number of observations, but also on the number of variables and whether they are continuous, discrete, qualitative, binary or whatever. The data quality should be checked particularly in regard to errors, outliers and missing observations. Summary statistics should be calculated for the data as a whole and for important subgroups usually including the mean and standard deviations and also plot the data appropriately.

4.Classifying online Job Advertisements through machine learning

Authors: RobertoBoselli , MirkoCesarini ,FabioMercorio ,MarioMezzanzanica

In this paper, we present our approach for automatically classifying million Web job vacancies on a standard taxonomy of occupations. It shows how the problem has been expressed in terms of text classification via machine learning and how the approach has been applied to certain real-life projects and we discuss the benefits provided to end users. an automatic text classifier is created by using an inductive process able to learn, from a set of pre-classified documents, the characteristics of the categories of interest.

5.Generating top-N items recommendation set using Collaborative,Content based filtering and rating variance

Authors: Anand Shanker Tewaria,*, Jyoti Prakash Singha , Asim Gopal Barman

Mostly content and collaborative filtering are widely used recommendation systems. Matrix factorization technique is also used by many recommendation systems and these techniques produce considerably bigger recommendation list. To overcome this issue the proposed approach generates smaller top-*n* item recommendations list by placing users' unseen items in recommendation list and thus attaining high precision value. The proposed recommendation system calculates the popularity of different item among other users, in the form of weight and which helps in generating high precision recommendations [6].It also finds the rating variance of all items. All these techniques collectively help in placing those items in smaller recommendation set that maybe preferred by the target user.

6.A Research of Job Recommendation System Based on Collaborative Filtering

Authors: Yingya Zhang ; Cheng Yang ; Zhixiang Ni

The paper aims to give an effective method of recommendation for online job hunting. It offers students a personalized service to find ideal jobs quickly even when the sparsity of user profile is obstructive. It also summarizes the filling of users' preference matrix with implicit behaviors. It explains various methods of calculating similarity scores.Cosine Similarity, uses two N-dimensional vector's cosine value to indicate the degree of similarity between them. Tanimoto coefficient measures similarity between finite sample sets. Log likelihood, method calculate similarity based on the common preference two users shared. City block distance is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes.

7.Machine learned job recommendation

Authors: Ioannis K. Paparrizos; Aristides Gioni

The implementation defines set of rules to extract job transitions based on employment dates. We represent transitions of an employee as a directed graph (I, T), which we refer to as a job transition graph. The set of nodes I represents institutions, and the set of edges T represents transitions between institutions. Given an individual who is currently employed in an institution, the system would predict the next institution where the individual will be employed. If the accuracy of predictions is sufficiently high, we may use our model to recommend institutions to employees who are seeking jobs. A predictive model is built by supervised machine learning. The results for the decision table/na ̈ive Bayes hybrid classifier achieves the highest accuracy.

8.A Dive into Web Scraper World

Authors: Deepak Kumar Mahto ; Lisha Singh

Web scraping is related to web indexing, whose task is to index information on the web with the help of a bot or web crawler. This journal paper gives an idea of how web scraping is carried out using python from a URL. The legal aspect, both positive and negative sides are taken into consideration.The Python language is used for the implementation due to the support extended by its library and also the beauty of coding style of python language, it is most suitable for Scraping data from Websites. The working of web scraping starts with the listing of the

URLs that is to be visited.The scraper visits the URLs and and processes all the webpage on the URL. Determine the content to be approach and find out the common pattern of CSS Selectors which is same on all other similar pages. To store the data in a well formatted manner check for the support of CSV into your programming language library.We have Data Extractor ready, do a recheck by extracting content on the temporary file so that CSV format gets a well formatted required output. Now we have required data, this data can be easily converted into any other format as CSV is so common to all.

9. Software refactoring at the package level using clustering techniques

Authors: A. Alkhalid ; M. Alshayeb ; S.A. Mahmoud

Software refactoring becomes quite challenging task as the software evolves .The papers implements clustering as a pattern recognition technique to assist in software refactoring activities at the package level. A comparative study is conducted applying three different clustering techniques on different software systems. In addition, the application of refactoring at the package level using an adaptive *k*-nearest neighbor (A-KNN) algorithm is introduced. The authors compared A-KNN technique with the other clustering techniques (viz. single linkage algorithm, complete linkage algorithm and weighted pair-group method using arithmetic averages).

10. A Sentiment-Enhanced Hybrid Recommender System for Movie Recommendation: A Big Data Analytics Framework

Authors: Yibo Wang ; Mingming Wang ; Wei Xu

The paper proposes a movie recommendation framework based on a hybrid recommendation model and sentiment analysis on Spark platform is proposed to improve the accuracy and timeliness of mobile movie recommender system. The approach first uses a hybrid recommendation method to generate a preliminary recommendation list. Then sentiment analysis is employed to optimize the list. Similarity between users are calculated by employing Euclidean distance to measure the similarity. The review is represented as a Vector Space Model (VSM).

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing recommendation system uses Collaborative filtering (CF) and its modifications as recommendation algorithm. Collaborative filtering can be implemented using two approaches that is user-based collaborative filtering and item-based collaborative filtering respectively. This recommendation engine mainly implements in two steps. Firstly, to find out how many users/items in the database are similar to the given user/item. Secondly, to assess other users/items to predict what grade you would give the user of his item, given the total weight of the users/items that are more similar to this one. Similarity is measured by using correlations between vectors of users/items.

3.2 SYSTEM REQUIREMENTS

3.2.1 HARDWARE REQUIREMENTS:

- 8GB RAM
- INTEL i5 Processor

3.2.2 SOFTWARE REQUIREMENTS

- Dataset in CSV format
- Windows OS/Linux

- Python-3.6.4-Anaconda

CHAPTER 4 SYSTEM DESIGN

4.1 PROPOSED SYSTEM

In the proposed system, we will use the job dataset obtained by performing web scraping from job boards and user dataset obtained from a survey conducted by a forum. The system performs exploratory analysis on the dataset to provide the insights on the data. A recommendation model is designed which uses hybrid filtering technique to overcome the cold start problem and sparsity encountered in the existing system. Hybrid algorithm trains the model for forming clusters and predicts the cluster to which a particular job can be assigned. Then based on user skills job cluster is determined and additionally finds the skills which the user can improve in future to have a wider range of job recommendations or newer career options.

4.2 ARCHITECTURE DIAGRAM

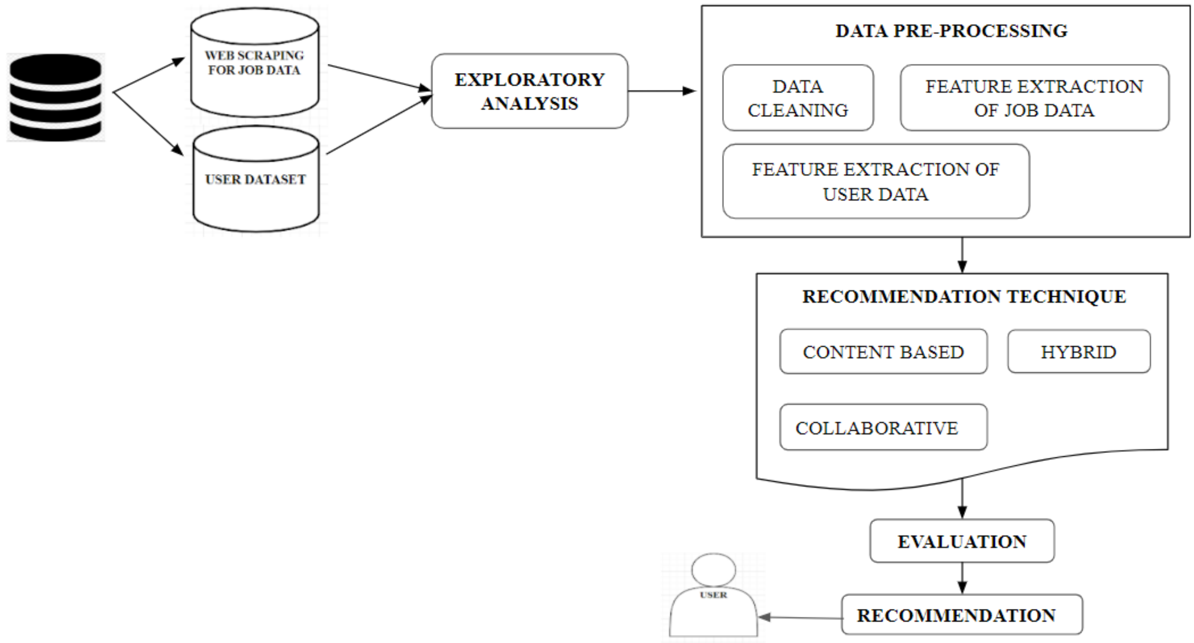


Figure 4.1 Architecture Diagram

4.3 USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

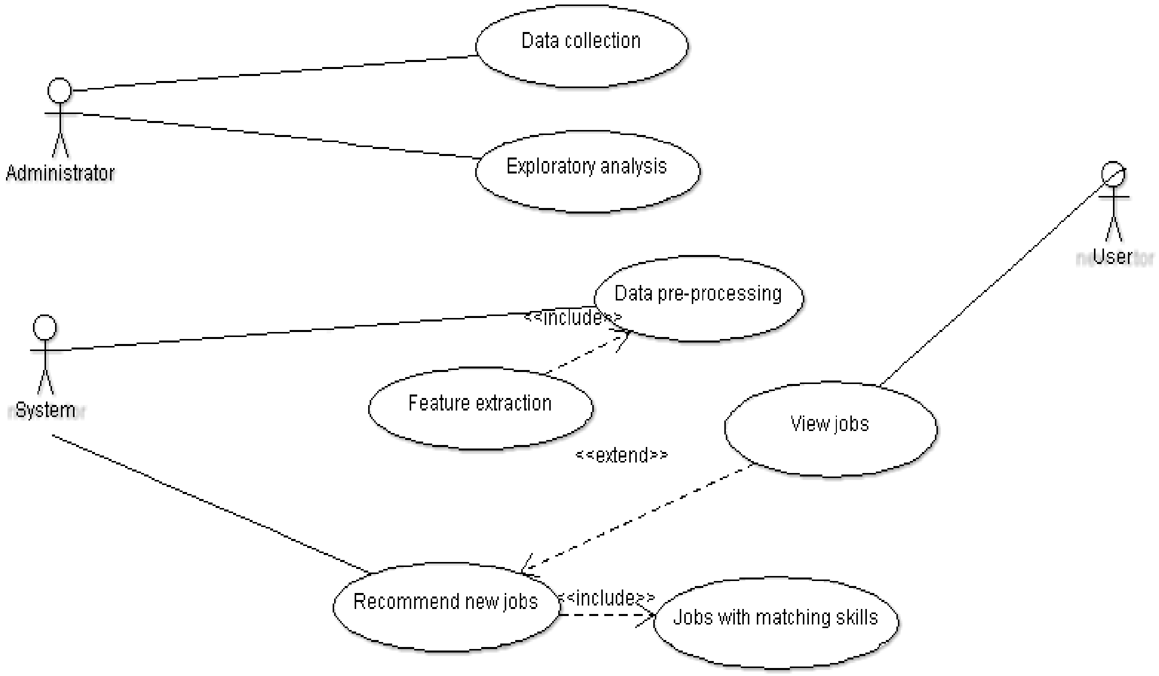


Figure 4.2 Use case diagram

4.4 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

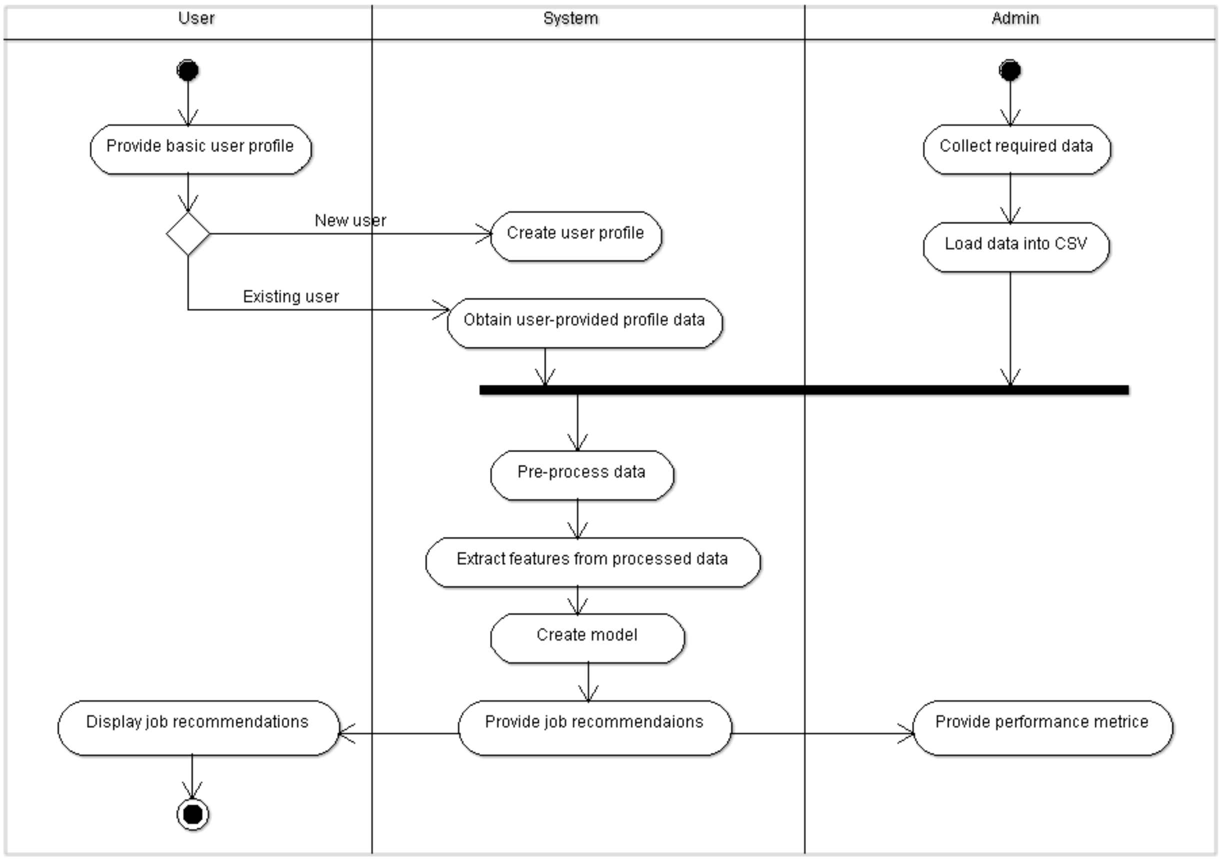
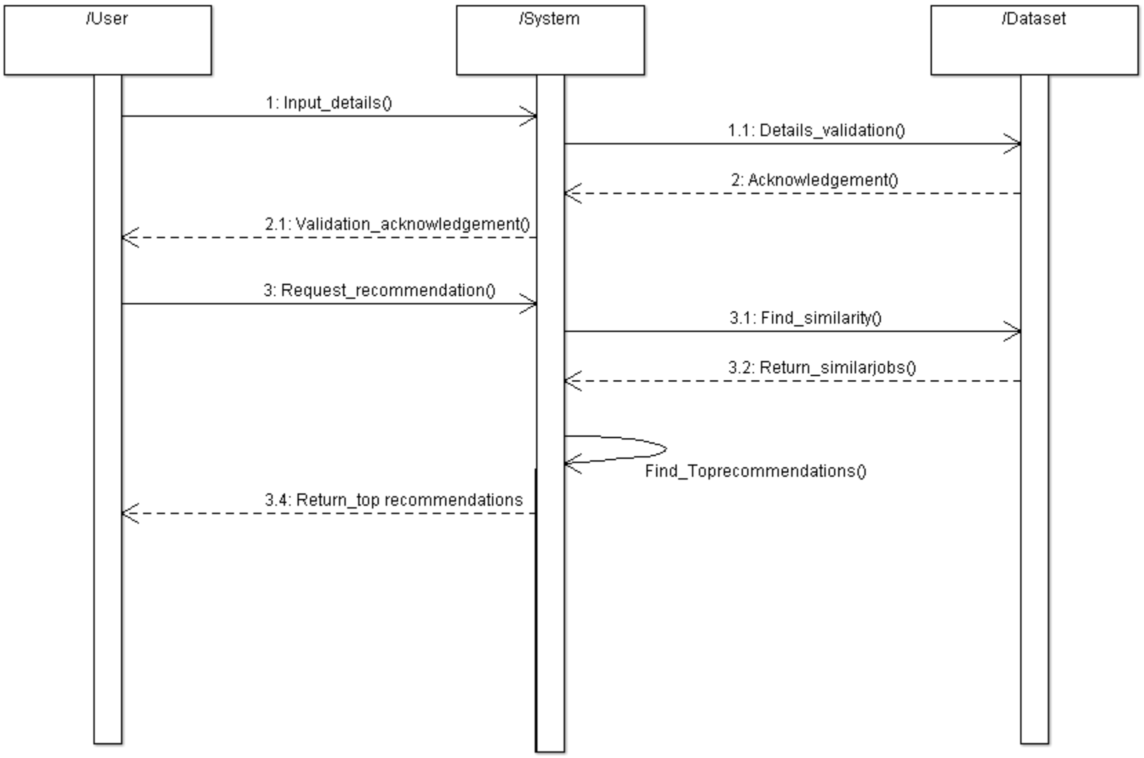

User
System
Admin
Provide basic user profile
New user
Create user profile
Existing user
Obtain user-provided profile data
Collect required data
Load data into CSV
Pre-process data
Extract features from processed data
Create model
Display job recommendations
Provide job recommendaions
Provide performance metrice

Figure 4.3 Activity diagram

4.5 SEQUENCE DIAGRAM

A sequence diagram depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.



/User
/System
/Dataset
1: Input_details()
1.1: Details_validation()
2: Acknowledgement()
2.1: Validation_acknowledgement()
3: Request_recommendation()
3.1: Find_similarity()
3.2: Return_similarjobs()
Find_Toprecommendations()
3.4: Return_top recommendations

Figure 4.4 Sequence Diagram

CHAPTER 5

IMPLEMENTATION

5.1 Data collection

Kaggle is an online community of data scientists and machine learners, owned by Google LLC. It is a public data platform. The data collection phase was divided into two phases, one is collection of job data containing the details of the jobs posted in websites and secondly collection of user data containing the details and descriptions of the users seeking job.

5.1.1 Data scraping for job dataset

The job data is nowadays available in the internet in vast amount through various websites. In order to perform the analysis on the latest data present on the jobs, the details were collected from the websites like the monsterindia and firstnaukri.

The web scraping technique was implemented to retrieve the necessary data from the list of URLs. This technique was implemented using the Python’s Beautiful Soup library. The raw data obtained as the result was converted to a CSV format to perform modifications on it. The attributes in the job dataset include the following:

Company_name-> Name of the company containing job opening

Employement_type->(full-time, part-time, Contract, Un-employeed)

Job_description->Textual explanation/expectations for the job_title

Skills->Features to be possessed by the candidate

Unique_id-> an identification for representing a job uniquely

Location-> State of a country

5.1.2 The user dataset for analysis

The user data depicts the information about the users of the system in other term a the candidates in search of jobs. The dataset was taken from the Kaggle data repository. The dataset is a survey dataset obtained by surveying a set of candidates and contained in the format of CSV file. It contains 80 thousand tuples and 120 attributes. Some of the important attributes in the dataset include the following:

Formal Education->(Bachelor's’ Degree, Master’s Degree, Associate Degree)

Undergraduate_Major-> The stream of undergraduate degree

Frameworks_workedin-> frameworks in which the user has worked

Platforms_workedin-> Platforms in which the user has worked

Years_of_experience->Number of years the user has been working in the industry

Job_Satisfaction-> The level of satisfaction the user has in the current job.

5.2 Exploratory data analysis

Exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods.The objectives of EDA is to suggest hypotheses about the causes of observed phenomena and support the selection of appropriate statistical tools and techniques.Multiple libraries are available in Python to perform EDA. Here we will be using the Pandas and the Matplotlib for performing EDA, Pandas for data manipulation and matplotlib for plotting graphs.Launch Jupyter Notebook by running command “jupyter-notebook”, an interface will get opened in the default browser.A new notebook is created and named. Visual representations are given by Bar-plots, Pie-chart, Box-plot, Scatter-plot, Histograms and so on.

Exploratory analysis is performed on the job dataset and various patterns have been recognised. The observed analysis includes the following findings:

- Number of jobs available by company
- Number of jobs available by job title
- Number of jobs available by each city

Secondly, the exploratory analysis on the user dataset produces the following findings:

- Top 10 databases used
- Top 10 languages used
- Educational level of the users
- Employment status of users

5.3 Data pre-processing

5.3.1 Data Cleaning of User Data and Job Data

Data cleaning was performed for job data and user data. It is generally the process of ensuring that your data is correct, consistent and useable by identifying any errors or corruptions in the data, correcting or deleting them, or manually processing them as needed to prevent the error from happening again.

In user data columns such as Hobbies , Contact Details , Time Fully Productive , Gender were dropped as they are not going to be of any value in our further process. The total number of attributes in the user dataset was reduced from 129 to 54.

Furthermore in our job data , the attributes such as job skills and job description were cleaned for stop words removal and numeric characters that occur very often were also removed. This was done using NLTK package.

5.3.2 Feature Extraction of User Data

Feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, leading to better human interpretations.It performs the dimensionality reduction process, where an initial set of raw variables is reduced to more manageable groups (features) for processing. The feature extraction is performed on the user dataset to find unique values in each attribute of the user dataset such as the attributes Frameworks_Worked_With , Development_Type, Languages_Worked_With, Database_Worked_With etc…

Making a list of all the unique values present in each feature.Converted the categorical variables such as employed / unemployed into numerical variable using LabelEncoder package.Extracted each skill feature such as domain , framework, language, platform by assigning true to the unique values from each feature of the user data.Created a separate csv file for each feature and stored the extracted features in it.

5.3.3 Feature Extraction of Job Data

Data pre-processing is performed on the job dataset. First step is to categorize the jobs. Predefined categories were used and word embedding was found using an advanced Natural Language Processing package called 'spacy'. The second step is to pull out the features from the job description using the features extracted from user data. Next using the extracted features from the jobs matchings for each job was found .

5.4 Recommendation Techniques

5.4.1 Content-based filtering

Recommendations in content-based filtering is based on a comparison between the content of the items and a user profile. The content of each item is represented as a set of descriptors or terms, typically the words that occur in a document. The user profile is represented with the same terms and built up by analyzing the content of items which have been seen by the user.

In the case of job recommender the users of the system is considered to be the term user and the jobs are considered to be the items in the filtering technique. The content of items is compared with the profile of users' skills. The content of an item is typically represented by a set of keywords extracted from a document. A user profile is represented by the same set of keywords and is generated by analyzing the content of jobs that matches user skills. Matching is done using cosine similarity between user and jobs.

Firstly a user_ID is given as an input to the system in case the user already exists in the system the recommendations will be generated , else for a new user the details such as the user_name and skills are collected and given as input to the system. Next the feature matrix is built for all the jobs in the job dataset and a feature matrix is built for the existing or new user . Comparison is performed between the job feature matrix and the user feature matrix and the jobs with closely matching skills are recommended. The recommendation provided is of the top 10 matching jobs.

5.4.1.1 Algorithm

```
INPUT     : user_id
OUTPUT : top 10 matching jobs


for job in jobs do
        read the features extracted & build a job feature matrix
end for
for i , j in jobs do
        similarity=cosine_simarility(i , j)
Sort similarity
for user in users
        recommend jobs based on highest total similarity measure
end for
```

5.4.1.2 Disadvantages of Content-based Filtering

- **Limited content analysis:** if the content does not contain enough information to discriminate the items precisely, the recommendation will be not precisely at the end.
- **Over-specialization:** content-based method provides a limit degree of novelty, since it has to match up the features of profile and items. A totally perfect content-based filtering may suggest nothing "surprised."
- **New user:** when there's not enough information to build a solid profile for a user, the recommendation could not be provided correctly.

5.4.2 Collaborative filtering

Collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources. Collaborative filtering encompasses techniques for matching people with similar interests and making recommendations on this basis. It recommends items based on similarity measures between users and/or items.

There are two basic approaches in Collaborative Filtering, user-based CF and item-based CF.

- User-based CF: find all other users with similar skills and recommend jobs that were recommended to them.
- Item-based CF: Instead of looking for the similar users, item-based CF finds all the similar items to the ones that the current user likes. The similarities between jobs are based on the applications received. Two jobs are similar when the same person apply to both of them.

  The approach used here in the system is User-based Collaborative Filtering. The various skills depicted in the user dataset were merged. A dictionary was built with the user_id as the unique key and skills as values. A user-user similarity matrix was built based on the skills of the user to find the users similar to the current user. The jobs recommended to the similar user were also recommended to the current user.

5.4.2.1 Algorithm

```
INPUT    : user_id
OUTPUT: jobs recommended to similar user

for user in users do
        user_skills= merge(user skills)
end do
Dict(user_id, user_skills)
for user in dict do
        find similarity of given user_id with all other user
        recommend jobs that were recommend to most similar user
end do
```

5.4.3 Hybrid filtering

Hybrid filtering technique combines different recommendation techniques in order to gain better system optimization to avoid some limitations and problems of pure recommendation systems. The idea behind hybrid techniques is that a combination of algorithms will provide more accurate and effective recommendations than a single algorithm as the disadvantages of one algorithm can be overcome by another algorithm. The combination of approaches can be done in any of the following ways: separate implementation of algorithms and combining the result, utilizing some content-based filtering in collaborative approach, utilizing some collaborative filtering in content-based approach.

5.4.3.1 Clustering Based Hybrid Filtering

In our approach we have implemented item-item Collaborative filtering using the concept of clustering .The feature matrix was built for the job dataset by vectorizing the skills and job description feature using TfidVecctorizer. Principal Component Analysis(PCA) was performed on job feature matrix for extracting important ones from large pool and to reduce overfitting. The resulting matrix was fed into the clustering algorithm. The obtained clusters are group of jobs with related skills. The model was trained for assigning a new user to a cluster.User resume was imported, vectorized and decomposed using PCA . Prediction was performed to determine the cluster in which the user’s skill fits in. Top suggestions from the predicted cluster was displayed. Also the most important skills the user is lacking from the closely related cluster was displayed.

5.4.3.2 Algorithm

```
INPUT     : resume
OUTPUT : prediction of cluster and recommendation of jobs and lacking skills
for job in jobs
        build job feature matrix
end do
```

```
PCA(job_feature_matrix)
clusters = AgglomerativeClustering(n_clusters=15)
for job in jobs
        Assign cluster number to each jobs
end do
Vectorize user's skills and job descriptions
Predict cluster for user
similarity = cosine_similarity(user_feature , cluster)
top 10  suggested jobs for the user's cluster
```

CHAPTER 6

PERFORMANCE EVALUATION

A performance metric measures an organization behaviour activities and performance. It should support a range of stakeholders need from customers, shareholders to employees. While traditionally many metrics are finance based, inwardly focussing on the performance of the organisation, metrics may also focus on the performance against customer requirements and value. In project management, performance metrics are used to assess the health of the project and consist of measuring of seven criteria that is safety, time, cost, resources, scope, quality, actions.

The quality of a recommendation algorithm can be evaluated using different types of measurement which can be accuracy or coverage. The type of metrics used depends on the type of filtering technique. Accuracy is the fraction of correct recommendations out of total possible recommendations while coverage measures the fraction of objects in the search space the system is able to provide recommendations for. The suitability of each metric depends on the features of the dataset and the type of tasks that the recommender system will do.

Developing performance metrics usually follows a process of:

- Establishing critical process/customer requirements
- Identifying specific, quantifiable outputs of work.
- Establishing targets against which results can be scored

Intra-list similarity is the average cosine similarity of all items in a list of recommendations. This calculation uses features of the recommended items (such as movie genre) to calculate the similarity. This metric can be used to measure diversity of the list of recommended items.

Parameters: recommendation- a list of top recommendations

feature_df - A dataframe containing the skill feature

Returns: The average intra-list similarity for recommendations

Recommendation model	Similarity accuracy
Content-based filtering	0.9170261355604602
Collaborative filtering	0.9343298097444914
Hybrid filtering	0.9458320948189590

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this project, analysis was performed on the real-time job data and recommendation model was generated that suggests suitable jobs for a job seeker based on skills . The model was created using three filtering techniques, content-based filtering, collaborative filtering and hybrid filtering. The exploratory analysis part of the project provided various insights of the job dataset and user dataset used. The hybrid filtering technique implemented as a combination of content-based filtering and item-item collaborative filtering using clustering, out performed the other two techniques implemented. Feature extraction performed on the job and user dataset to retrieve the unique value in each attribute was used to provide job suggestions based on the skills.

The proposed work could be further extended by performing web scraping dynamically to retrieve the job data from wider range of job board websites. The user clicks through the system could be monitored and used for increasing the accuracy of the recommendation model. The offline model created could be incorporated as an online model. The model could be run in a distributed system environment and improvise the architecture for scalability.

APPENDIX

APPENDIX 1

1.1 Web Scraping

```
#Importing the python packages
import requests
import csv
from datetime import datetime
from bs4 import BeautifulSoup
for i in range(1, 10):
        page=requests.get("https://www.monsterindia.com/jobs-in-chennai.html-
        %s" %i)
        soup = BeautifulSoup(page.content, 'html.parser')
data = soup.find_all("div",{"class":"jobwrap"})
item = data[0]
#Creating the csv file in write mode
filename="jobs.csv"
f=open(filename,"w")

headers="Job_title,Company\n"
f.write(headers)

#Extracting raw data
for item in data:
   job_title=item.find_all("span",{"class":"title_in"})
   job_company=item.find_all("div",{"class" : "jtxt orange"})
   job_loc=item.find_all("div",{"class":"jtxt jico ico1"})
   job_exp=item.find_all("div",{"class":"jtxt jico ico2"})
   job_id=item.find_all("div",{"class":"jtxt jico ico3"})
#Cleaning the data
   jobtitle=job_title[0].text.strip()
   industry="HR"
   company=job_company[0].text
   loc=job_loc[0].text
```

```
    exp=job_exp[0].text
    print(exp)
    f.write(title.replace(",","|") + ","+industry.replace(",","|")+"," +
company.replace(",","|") + ","+loc.replace(",","|")+ ","+exp+"\n")
f.close()
```

1.2 Exploratory Data Analysis

Finding jobs by title

```
plt.figure(figsize=(20,10))
sns.countplot(y=df['jobtitle'], order=df['jobtitle'].value_counts().index,
palette=mypalette)
plt.ylabel('Job Title', fontsize=14)
plt.xlabel('Number of Job postings', fontsize=14)
plt.title("Most seeked Jobs", fontsize=18)
plt.ylim(20.5,-0.5)
plt.show()
```

Finding jobs by company

```
mypalette = sns.color_palette('GnBu_d', 40)
plt.figure(figsize=(20,10))
sns.countplot(y=df['company'], order=df['company'].value_counts().index,
palette=mypalette)
plt.ylabel('Company Name', fontsize=14)
plt.xlabel('Number of Job postings', fontsize=14)
plt.title("Companies with most job postings", fontsize=18)
plt.ylim(20.5,-0.5)
plt.show()
```

Employment status of the users

```
temp = stack_data['Employment'].value_counts()
df = pd.DataFrame({'labels': temp.index,
                  'values': temp.values
                 })
df.iplot(kind='pie',labels='labels',values='values', title='Employment Status of
Developers', hole = 0.8, color =
['#8B7355','#FF6103','#8EE5EE','#458B00','#FFF8DC','#68228B'])
```

1.2 Data cleaning

```
attr_to_drop=['Hobby','NumberMonitors','CheckInCode','WakeTime','TimeFullyProductive','SkipMeals','SexualOrientation','HoursOutside','Exercise','EthicalImplications','EducationParents',’TimeAfterBootcamp’]

print(len(df.columns))

df=df[df.columns.difference(attr_to_drop)]

print(len(df.columns))
```

1.3 Feature Extraction of User Data

```
import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

#find the unique values in each attribute

Frameworknextyear=list()

for value in df1["FrameworkDesireNextYear"]:

    new=value.split(';')

    for i in new:

        Frameworknextyear.append(i)

sFNY=set(Frameworknextyear)

FrameworkWorkedWith=list()

for value in df1["FrameworkWorkedWith"]:

    new=value.split(';')

    for i in new:

        FrameworkWorkedWith.append(i)

sFW=set(FrameworkWorkedWith)
```

```
PlatformDesireNextYear=list()
for value in df1["PlatformDesireNextYear"]:
        new=value.split(';')
        for i in new:
            PlatformDesireNextYear.append(i)
sPDNY=set(PlatformDesireNextYear)
PlatformWorkedWith=list()
for value in df1["PlatformWorkedWith"]:
        new=value.split(';')
        for i in new:
            PlatformWorkedWith.append(i)
sPWW=set(PlatformWorkedWith)
LanguageDesireNextYear=list()
for value in df1["LanguageDesireNextYear"]:
        new=value.split(';')
        for i in new:
            LanguageDesireNextYear.append(i)
sLDNY=set(LanguageDesireNextYear)
LanguageWorkedWith=list()
for value in df1["LanguageWorkedWith"]:
        new=value.split(';')
        for i in new:
            LanguageWorkedWith.append(i)
sLWW=set(LanguageWorkedWith)

CommunicationTools=list()
```

```
for value in df1["CommunicationTools"]:
    new=value.split(';')
    for i in new:
        CommunicationTools.append(i)
sCT=set(CommunicationTools)

DatabaseWorkedWith=list()
for value in df1["DatabaseWorkedWith"]:
    new=value.split(';')
    for i in new:
        DatabaseWorkedWith.append(i)
sDWW=set(DatabaseWorkedWith)
DatabaseDesireNextYear=list()
for value in df1["DatabaseDesireNextYear"]:
    new=value.split(';')
    for i in new:
        DatabaseDesireNextYear.append(i)
sDNY=set(DatabaseDesireNextYear)
DevType=list()
for value in df1["DevType"]:
    new=value.split(';')
    for i in new:
        DevType.append(i)
```

1.4 Feature Extraction of Job Data

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from scipy import spatial
import spacy
import warnings; warnings.simplefilter('ignore')
class job_postings:
    def __init__(self,link):
        self.df2=pd.read_csv(link,encoding='cp850',engine='python')
        self.training_range=int(len(self.df2.loc[:,'uniq_id']))
    def check_threshold(self,threshold,ele):
        if(ele[0]!=threshold[0][0] and abs(ele[1]-threshold[0][1])<0.03):
            return True
        else:
            return False
    def categorize_jobs(self):
        #Predefined categories
        #Compare similarities of word embeddings
        nlp=spacy.load('en_core_web_sm')
        job_id=self.df2.loc[:,'uniq_id'].tolist()[:self.training_range]
        job_titles=self.df2.loc[:,'jobtitle'].tolist()[:self.training_range]
        job_descriptions=self.df2.loc[:,'jobdescription'].tolist()[:self.training_range]
        final_cat=pd.DataFrame(index=job_id)
        categories=['Network Engineer','Full stack','QA/Test Developer','Enterprise
application','DevOps','Mobile Developer','Back End','Database
Administrator(DBA)','Front End','Game developer','System Administrator','Data
Scientist','Business analyst','Sales professional','Product Manager','Information
Security','Software Developer/Java Developer','Web Developer','Cloud
Computing']

        for category in categories:

            final_cat[category]=np.nan

        for job_t_d in list(zip(job_id,job_titles,job_descriptions)):

            id_job=job_t_d[0]

            job_i=str(job_t_d[1])

            job_d=str(job_t_d[2])

            job_title=nlp(job_i.lower())

            job_description=nlp(job_d.lower())

            match_cat_title=dict()

            match_cat_description=dict()

            for category in categories:

                word=nlp(category.lower())

                match_cat_title[category]=job_title.similarity(word)

                match_cat_description[category]=job_description.similarity(word)

                    match_cat_title=sorted(match_cat_title.items(),key=lambda
          x:x[1],reverse=True)

match_cat_description=sorted(match_cat_description.items(),key=lambda
x:x[1],reverse=True)

            #a represents max

            a = match_cat_title[0]

            cat_description= lambda x: self.check_threshold(match_cat_title,x)

          match_cat_description=list(filter(cat_description,match_cat_description))

            l=len(match_cat_description)

            if(l!=0):
                final_cat.loc[id_job,a[0]]=1
                match_cat_description.extend([(match_cat_title[0][0],1)])
                sum_proportion=sum([x[1] for x in match_cat_description])
                for ele in match_cat_description:
                    final_cat.loc[id_job,ele[0]]=ele[1]/sum_proportion
            else:
                final_cat.loc[id_job,a[0]]=1
        return final_cat
    def clean_skills(self):
        extracted_skills=dict()
        job_skills=np.asarray(self.df2.loc[:,"skills"])
        for i in range(self.training_range):
            #Method 1: Manual pre-processing
            job_id=self.df2.iloc[i,-1]
            #Method 2:Using NLTK
            tokenizer=nltk.tokenize.RegexpTokenizer(r'\w+')
            #print(job_skills[i])
            if(pd.isnull(job_skills[i])):
                continue
            stopwords_list=stopwords.words("english")
            tokens=re.split("|".join([","," and","/"," AND"," or"," OR",";"]),job_skills[i])
            tokens=list(set(tokens))
            extracted_skills[job_id]=[]
            extracted_skills[job_id].extend(tokens)
        return extracted_skills
def extract_skills(self,extracted_skills):
```

```
nlp=spacy.load('en_core_web_sm')
df_languages=pd.read_csv("Documents/data/languages.csv")
df_frameworks=pd.read_csv("Documents/data/frameworks.csv")
df_database=pd.read_csv("Documents/data/database.csv")
df_os=pd.read_csv("Documents/data/operating_systems.csv")
df_plat=pd.read_csv("Documents/data/platforms.csv")
frameworks=str(df_frameworks.iloc[:,1].tolist())
frameworks=[x.lower().strip() for x in frameworks]
#frameworks=[str(x).split(",")[0] for x in df_frameworks.iloc[:,1]]
languages=str(df_languages.iloc[:,0].tolist())
languages=[x.lower().strip() for x in languages]
#frameworks=[x.lower().strip().split('\t')[0] for x in frameworks]
databases=str(df_database.iloc[:,0].tolist())
databases=[x.lower().strip() for x in databases]
op_systems=str(df_os.iloc[:,0].tolist())
op_systems=[x.lower().strip() for x in op_systems]
platforms=str(df_plat.iloc[:,1].tolist())
#print(platforms)
platforms=[x.lower().strip() for x in platforms]
#print(frameworks)
new_extracted=dict()
for ele in extracted_skills.keys():
    final_lang="
    final_frame="
    final_others="
    final_database="
```

```
final_plat="
final_os="
#print(extracted_skills[ele])
for skill in extracted_skills[ele]:
    skill_base=skill.lower().strip()
    #print(skill_base)
    if(skill_base in languages):
        if(final_lang=="):
            final_lang=skill_base
        else:
            final_lang=final_lang+","+skill_base
    elif(skill_base in frameworks):
        if(final_frame=="):
            final_frame=skill_base
        else:
            final_frame=final_frame+","+skill_base
    elif(skill_base in databases):
        if(final_database=="):
            final_database=skill_base
        else:
            final_database=final_database+","+skill_base
    elif(skill_base in op_systems):
        if(final_os=="):
            final_os=skill_base
        else:
            final_os=final_os+","+skill_base
        elif(skill_base in platforms):
            if(final_plat==''):
                final_plat=skill_base
            else:
                final_plat=final_plat+","+skill_base
        else:
            if(final_others==''):
                final_others=skill_base
            else:
                final_others=final_others+","+skill_base

new_extracted[ele]=[final_lang,final_frame,final_database,final_os,final_plat,final_others]
    for ele,describe in list(zip(self.df2.loc[:,'uniq_id'],self.df2.loc[:,'jobdescription'].tolist()))[:self.training_range]:
        doc=nlp(describe)
        final_lang=''
        final_frame=''
        final_others=''
        final_database=''
        final_plat=''
        final_os=''
        for ent in doc.ents:
            word=ent.text
            word=word.lower().strip()
            try:
```

```
            if(word in languages and word not in final_lang and word not in new_extracted[ele][0].split(",")):
                if(final_lang=="):
                    final_lang=word
                else:
                    final_lang=final_lang+","+word
            elif(word in frameworks and word not in final_frame and word not in new_extracted[ele][1].split(",")):
                if(final_frame=="):
                    final_frame=word
                else:
                    final_frame=final_frame+","+word
            elif(word in databases and word not in final_database and word not in new_extracted[ele][2].split(",")):
                if(final_database=="):
                    final_database=word
                else:
                    final_database=final_database+","+word
            elif(word in op_systems and word not in final_os and word not in new_extracted[ele][3].split(",")):
                if(final_os=="):
                    final_os=word
                else:
                    final_os=final_os+","+word
            elif(word in platforms and word not in final_plat and word not in new_extracted[ele][4].split(",")):
                if(final_plat=="):
                    final_plat=word
```

```
                        else:
                            final_plat=final_plat+","+word
                    else:
                        if(final_others=="):
                            final_others=word
                        else:
                            final_others=final_others+","+word
                except KeyError:
                    print("")
            try:
                if(final_lang!="):
                    new_extracted[ele][0]+=","+final_lang
                if(final_frame!="):
                    new_extracted[ele][1]+=","+final_frame
                if(final_database!="):
                    new_extracted[ele][2]+=","+final_database
                if(final_os!="):
                    new_extracted[ele][3]+=","+final_os
                if(final_plat!="):
                    new_extracted[ele][4]+=","+final_plat
                if(final_others!="):
                    new_extracted[ele][5]+=","+final_others
            except KeyError:
                print("")

new_extracted[ele]=[final_lang,final_frame,final_database,final_os,final_plat,fi
nal_others]
```

```
extracted_skills_df=pd.DataFrame.from_dict(new_extracted,orient='index',colu
mns=['Language','Framework','Database','OS','Platform','Others'])
    return extracted_skills_df
def create_job_profile(self,extracted_skills_df,domain_df):
    job_id=extracted_skills_df.index.tolist()
    languages_df=pd.DataFrame(index=job_id)
    platforms_df=pd.DataFrame(index=job_id)
    frameworks_df=pd.DataFrame(index=job_id)
    databases_df=pd.DataFrame(index=job_id)

    for job,lang,frame,plat,datab in
list(zip(job_id,extracted_skills_df.loc[:,'Language'].tolist(),extracted_skills_df.l
oc[:,'Framework'].tolist(),extracted_skills_df.loc[:,'Platform'].tolist(),extracted_s
kills_df.loc[:,'Database'].tolist())):
      #Languages
      l=lang.split(",")
      if(lang!=np.nan or lang!="):
        for ele in l:
          if(ele=="):
            continue
          if(ele not in languages_df.columns):
            #languages.append(ele)
            languages_df[ele]=np.nan
          languages_df.loc[job,ele]=1

      #Frameworks
      l=frame.split(",")
```

```
if(frame!=np.nan or frame!="):
    for ele in l:
        if(ele=="):
            continue
        if(ele not in frameworks_df.columns):
            #languages.append(ele)
            frameworks_df[ele]=np.nan
        frameworks_df.loc[job,ele]=1
#Platforms
l=plat.split(",")
if(plat!=np.nan or plat!="):
    for ele in l:
        if(ele=="):
            continue
        if(ele not in platforms_df.columns):
            #languages.append(ele)
            platforms_df[ele]=np.nan
        platforms_df.loc[job,ele]=1

#Databases
l=datab.split(",")
if(datab!=np.nan or datab!="):
    for ele in l:
        if(ele=="):
            continue
        if(ele not in databases_df.columns):
```

```
                        #languages.append(ele)
                        databases_df[ele]=np.nan
                    databases_df.loc[job,ele]=1
        languages_df=languages_df.reindex_axis(sorted(languages_df.columns), axis=1)

frameworks_df=frameworks_df.reindex_axis(sorted(frameworks_df.columns), axis=1)
        platforms_df=platforms_df.reindex_axis(sorted(platforms_df.columns), axis=1)
        databases_df=databases_df.reindex_axis(sorted(databases_df.columns), axis=1)
        domain_df=domain_df.reindex_axis(sorted(domain_df.columns), axis=1)


languages_df.index.name=frameworks_df.index.name=platforms_df.index.name=databases_df.index.name=domain_df.index.name='uniq_id'
        languages_df.to_csv("Documents/data/languages_job_profile.csv")
        frameworks_df.to_csv("Documents/data/frameworks_job_profile.csv")
        platforms_df.to_csv("Documents/data/platforms_job_profile.csv")
        databases_df.to_csv("Documents/data/databases_job_profile.csv")
        domain_df.to_csv("Documents/data/domain_job_profile.csv")
#Input is two dataframes
    def create_common_profile(self,flag=0):
        if(flag==0):
            #Domain

userprofile=pd.read_csv("Documents/data/DevType.csv",index_col='Respondent')
```

```
jobprofile=pd.read_csv("Documents/data/domain_job_profile.csv",index_col=False)
    print(jobprofile.index)
    jobprofile.drop('uniq_id', axis=1, inplace=True)
    jobprofile.index.name='uniq_id'
    print("index 2in domain")
    print(jobprofile.index)
    #print(jobprofile.loc[:,'uniq_id'])
    userprofile.rename(columns={'Product manager':'Product Manager','Back-end developer':'Back End','C-suite executive (CEO, CTO, etc.)':'C-suite executive','Data scientist or machine learning specialist':'Data Scientist','Database administrator':'Database Administrator(DBA)','Mobile developer':'Mobile Developer','Desktop or enterprise applications developer':'Enterprise application','DevOps specialist':'DevOps','Front-end developer':'Front End','Full-stack developer':'Full stack','Marketing or sales professional':'Sales professional','QA or test developer':'QA/Test Developer','System administrator':'System Administrator','Game or graphics developer':'Game developer'},inplace=True)
    jobprofile.rename(columns={'Business analyst':'Data or business analyst'},inplace=True)
    print(userprofile.columns)
    print(jobprofile.columns)
    print("index in domain")
    print(jobprofile.index)
    #Present in userprofile but not in jobprofile
    a=list(set(userprofile.columns)-set(jobprofile.columns))
    print(a)
    for i in a:
        if(i!='Respondent'):
            jobprofile[i]=0
    b=list(set(jobprofile.columns)-set(userprofile.columns))
    print(b)
    for i in b:
        if(i!='uniq_id'):
            userprofile[i]=0
    userprofile=userprofile[sorted(userprofile.columns.tolist())]
    jobprofile=jobprofile[sorted(jobprofile.columns.tolist())]
    #Exclude
    userprofile=userprofile[userprofile.columns.tolist()]
    jobprofile=jobprofile[jobprofile.columns.tolist()]
    userprofile.to_csv("Documents/data/domain_user_profile.csv")
    jobprofile.to_csv("Documents/data/domain_job_profile.csv")

    #Languages

df_user=pd.read_csv("Documents/data/LanguageWorkedWith.csv",index_col='Respondent')

df_job=pd.read_csv("Documents/data/languages_job_profile.csv",index_col=False)
    df_job.index.name='uniq_id'
    df_job.rename(columns={'visual basic .net':'vb.net'},inplace=True)
    df_user.columns=list(map(lambda x:x.lower(),df_user.columns))
    df_job.columns=list(map(lambda x:x.lower(),df_job.columns))
    columns_to_add=[]
    a=list(set(df_user.columns)-(set(df_job.columns)))
```

```
        print(a)
        for i in a:
            if(i!='Respondent'):
                df_job[i]=0
        b=list(set(df_job.columns)-set(df_user.columns))
        print(b)
        for i in b:
            if(i!='uniq_id'):
                df_user[i]=0
        #print(df_job.index)
        df_user=df_user[sorted(df_user.columns.tolist())]
        df_job=df_job[sorted(df_job.columns.tolist())]
        print("index 2")
        print(df_job.index)

print(len(set(df_user.columns).intersection(df_job.columns)),len(df_user.columns))
        df_user,df_job=self.clean_common_profile(df_user,df_job,'Language')
        df_user.to_csv("Documents/data/languages_profile_user.csv")
        df_job.to_csv("Documents/data/languages_profile_job.csv")


        #Frameworks

df_user=pd.read_csv("Documents/data/FrameworkWorkedWith.csv",index_col='Respondent')

df_job=pd.read_csv("Documents/data/frameworks_job_profile.csv",index_col=False)
```

```
df_job.index.name='uniq_id'
df_user.columns=list(map(lambda x:x.lower(),df_user.columns))
df_job.columns=list(map(lambda x:x.lower(),df_job.columns))

a=list(set(df_user.columns)-(set(df_job.columns)))
print(a)
for i in a:
    if(i!='Respondent'):
        df_job[i]=0
b=list(set(df_job.columns)-set(df_user.columns))
print(b)
for i in b:
    if(i!='uniq_id'):
        df_user[i]=0
df_user=df_user[sorted(df_user.columns.tolist())]
df_job=df_job[sorted(df_job.columns.tolist())]
print(len(set(df_user.columns).intersection(df_job.columns)),len(df_user.columns))
df_user,df_job=self.clean_common_profile(df_user,df_job,'Framework')
df_user.to_csv("Documents/data/frameworks_profile_user.csv")
df_job.to_csv("Documents/data/frameworks_profile_job.csv")
#Platforms
df_user=pd.read_csv("Documents/data/PlatformWorkedWith.csv",index_col='Respondent')
df_job=pd.read_csv("Documents/data/platforms_job_profile.csv",index_col=False)
print(df_user.columns)
```

```
            df_job.index.name='uniq_id'
            print(df_job.columns)
            df_user.columns=list(map(lambda x:x.lower(),df_user.columns))
            df_job.columns=list(map(lambda x:x.lower(),df_job.columns))

            a=list(set(df_user.columns)-(set(df_job.columns)))
            print(a)
            for i in a:
                if(i!='Respondent'):
                    df_job[i]=0
            b=list(set(df_job.columns)-set(df_user.columns))
            print(b)
            for i in b:
                if(i!='uniq_id'):
                    df_user[i]=0
            df_user=df_user[sorted(df_user.columns.tolist())]
            df_job=df_job[sorted(df_job.columns.tolist())]
print(len(set(df_user.columns).intersection(df_job.columns)),len(df_user.colum
ns))
            df_user,df_job=self.clean_common_profile(df_user,df_job,'Platform')
            df_user.to_csv("Documents/data/platforms_profile_user.csv")
            df_job.to_csv("Documents/data/platforms_profile_job.csv")
          #Databases
df_user=pd.read_csv("Documents/data/DatabaseWorkedWith.csv",index_col='
Respondent')
df_job=pd.read_csv("Documents/data/databases_job_profile.csv",index_col=Fa
lse)
            df_job.index.name='uniq_id'
```

```
        df_user.columns=list(map(lambda x:x.lower(),df_user.columns))
        df_job.columns=list(map(lambda x:x.lower(),df_job.columns))

        a=list(set(df_user.columns)-(set(df_job.columns)))
        print(a)
        for i in a:
            if(i!='Respondent'):
                df_job[i]=0
        b=list(set(df_job.columns)-set(df_user.columns))
        print(b)
        for i in b:
            if(i!='uniq_id'):
                df_user[i]=0
        df_user=df_user[sorted(df_user.columns.tolist())]
        df_job=df_job[sorted(df_job.columns.tolist())]        # print(len(set(df_user.columns).intersection(df_job.columns)),len(df_user.columns))
        df_user,df_job=self.clean_common_profile(df_user,df_job,'Database')
        df_user.to_csv("Documents/data/databases_profile_user.csv")
        df_job.to_csv("Documents/data/databases_profile_job.csv")
```

1.4 Content based filtering

```
def cosine_similarity(arr1,arr2):
    ans=1- spatial.distance.cosine(arr1,arr2)
    if(np.isnan(ans)):
        return 0
    else:
```

```
    return ans
def match_profile(self,input_path,user_id,flag=0):
    #Match a given user_id with all jobs in the database

    #Check if user id exists

df=pd.read_csv("Documents/data/domain_user_profile.csv",index_col='Respondent')
    #print(df.columns)
    matches=dict()
    if(flag==0):
      if(user_id in df.index):
        userdomain=df.loc[user_id,:]
        #print(userdomain)
        #If it does, retrieve the user profile from input_path

df=pd.read_csv("Documents/data/languages_profile_user.csv",index_col='Respondent')
        userlanguages=df.loc[user_id,:]


df=pd.read_csv("Documents/data/frameworks_profile_user.csv",index_col='Respondent')
        userframeworks=df.loc[user_id,:]


df=pd.read_csv("Documents/data/platforms_profile_user.csv",index_col='Respondent')
        userplatforms=df.loc[user_id,:]
```

```
df=pd.read_csv("Documents/data/databases_profile_user.csv",index_col='Respondent')
            userdatabases=df.loc[user_id,:]


            userdomain=np.asarray(userdomain.fillna(0))
            userlanguages=np.asarray(userlanguages.fillna(0))
            userframeworks=np.asarray(userframeworks.fillna(0))
            userplatforms=np.asarray(userplatforms.fillna(0))
            userdatabases=np.asarray(userdatabases.fillna(0))
            #print(userdomain)
        else:
            print("error! user id not in Dataset")
        #If it doesn't,take user profile as input
    else:


        print("New user!Enter details..")
        name=input("Enter full name")
        skills=input("Enter skills(comma separated). These are programming languages, frameworks,platforms or databases you have experience with").split(",")
        domains="
        flag=1
        while(1):
            print("Enter domain(s) of interest separated by commas(Names are case sensitive). Should be one of the following:")
            for i in df.columns:
```

```
            print(i,end=",")
        domains=input().split(",")
        for domain in domains:
            if(domain not in df.columns):
                flag=0
                break
        if(flag==1):
            break
        else:
            print("Please enter valid domain")



    #domains=list(map(lambda x:x.lower(),domains))
    skills=list(map(lambda x:x.lower(),skills))

    userdomain=pd.DataFrame(columns=df.columns)
    dictionary=dict()
    for domain in domains:
        dictionary[domain]=1.0
    userdomain=userdomain.append(dictionary,ignore_index=True)



df=pd.read_csv("Documents/data/languages_profile_user.csv",index_col='Respondent')
    userlanguages=pd.DataFrame(columns=df.columns)
    dictionary=dict()
    for skill in skills:
        if(skill in df.columns):
            dictionary[skill]=1.0
    userlanguages=userlanguages.append(dictionary,ignore_index=True)


df=pd.read_csv("Documents/data/frameworks_profile_user.csv",index_col='Respondent')
    userframeworks=pd.DataFrame(columns=df.columns)
    dictionary=dict()
    for skill in skills:
        if(skill in df.columns):
            dictionary[skill]=1.0

userframeworks=userframeworks.append(dictionary,ignore_index=True)


df=pd.read_csv("Documents/data/platforms_profile_user.csv",index_col='Respondent')
    userplatforms=pd.DataFrame(columns=df.columns)
    dictionary=dict()
    for skill in skills:
        if(skill in df.columns):
            dictionary[skill]=1.0
    userplatforms=userplatforms.append(dictionary,ignore_index=True)
```

```
df=pd.read_csv("Documents/data/databases_profile_user.csv",index_col='Respondent')
        userdatabases=pd.DataFrame(columns=df.columns)
        dictionary=dict()
        for skill in skills:
            if(skill in df.columns):
                dictionary[skill]=1.0
        userdatabases=userdatabases.append(dictionary,ignore_index=True)
        #print(userdomain)
        userdomain=np.asarray(userdomain.iloc[0,:].fillna(0))
        userlanguages=np.asarray(userlanguages.iloc[0,:].fillna(0))
        userframeworks=np.asarray(userframeworks.iloc[0,:].fillna(0))
        userplatforms=np.asarray(userplatforms.iloc[0,:].fillna(0))
        userdatabases=np.asarray(userdatabases.iloc[0,:].fillna(0))


jobdomain=pd.read_csv("Documents/data/domain_job_profile.csv",index_col='uniq_id')

joblanguages=pd.read_csv('Documents/data/languages_profile_job.csv',index_col='uniq_id')

jobframeworks=pd.read_csv('Documents/data/frameworks_profile_job.csv',index_col='uniq_id')

jobplatforms=pd.read_csv('Documents/data/platforms_profile_job.csv',index_col='uniq_id')
```

```
jobdatabases=pd.read_csv('Documents/data/databases_profile_job.csv',index_col='uniq_id')
        #print(len(jobdomain.index),len(joblanguages.index))
        start=time.time()
        for i in jobdomain.index:
            #print(i)
            domain=jobdomain.loc[i,:].fillna(0)
            language=joblanguages.loc[i,:].fillna(0)
            framework=jobframeworks.loc[i,:].fillna(0)
            platform=jobplatforms.loc[i,:].fillna(0)
            database=jobdatabases.loc[i,:].fillna(0)
            job_id=str(i)
            domain=np.asarray(domain)
            language=np.asarray(language)
            framework=np.asarray(framework)
            platform=np.asarray(platform)
            database=np.asarray(database)
            t=time.time()
            #print(len(domain),len(userdomain))

score=(0.7*cosine_similarity(domain,userdomain))+(0.3*(cosine_similarity(language,userlanguages)+cosine_similarity(framework,userframeworks)+cosine_similarity(platform,userplatforms)+cosine_similarity(database,userdatabases)))
            #print(score)
            matches[job_id]=score

score=(0.7*cosine_similarity(domain,userdomain))+(0.3*(cosine_similarity(lan
guage,userlanguages)+cosine_similarity(framework,userframeworks)+cosine_similarity(platform,userplatforms)+cosine_similarity(database,userdatabases)))
            #Initializing job profiles for later access
            #print(score)
            self.job_domain=domain
            self.job_language=language
            self.job_framework=framework
            self.job_platform=platform
            self.job_database=database

            self.user_domain=userdomain
            self.user_language=userlanguages
            self.user_framework=userframeworks
            self.user_platform=userplatforms
            self.user_database=userdatabases
        matches=sorted(matches.items(),key=lambda x:x[1],reverse=True)
        recommendations=matches[:10]
        #title=matches[0]
        score=matches[1]
        print("Recommendations are",'\n')
        print("The Job ID and Score",'\n',recommendations,'\n')
        rows=pd.DataFrame(columns=self.df2.columns)
        count=0
        title= []
        score= []
        for i in recommendations:
            title.append(i[0])
```

```
            score.append(i[1])
            row=self.df2[self.df2['uniq_id']==i[0]]
            #rows[count]=np.asarray(row.values.T.tolist()[0])
            rows=rows.append(row.iloc[:,0:9])
            count=count+1
            #print(row)
        end=time.time()
        print("total time for building similarity matrix ")
        print(end-start)
        acc=mean(score)
        print("Accuracy")
        print(acc)
        plt.plot(title,score, color='b')
        plt.xlabel('Title')
        plt.ylabel('Score')
        plt.title('Accuracy')
        plt.show()
        return rows
```

1.5 Collaborative Filtering

```
cbf = pd.DataFrame(df1['Respondent'])
import time
start=time.time()
for i in range(98855):
    r=random.randint(1,4291)
    cbf.loc[i,"company"]=l[r]
end=time.time()
```

```
print(end-start)
cbf.to_csv("Documents/data/colabdata.csv")
#merge all the dataset having skills from user profile.
dflang=pd.read_csv("Documents/data/LanguageWorkedWith.csv")
dfdata=pd.read_csv("Documents/data/DatabaseWorkedWith.csv")
dfplat=pd.read_csv("Documents/data/PlatformWorkedWith.csv")
dfframe=pd.read_csv("Documents/data/FrameworkWorkedWith.csv")
dfdev=pd.read_csv("Documents/data/DevType.csv")
dfmerge=pd.merge(dflang,dfdata,on="Respondent")
dfmerge=pd.merge(dfmerge,dfplat,on="Respondent")
dfmerge=pd.merge(dfmerge,dfframe,on="Respondent")
dfmerge=pd.merge(dfmerge,dfdev,on="Respondent")
#dfmerge.to_csv("Documents/data/userskills.csv")
import numpy as np
from numpy import array
dfuser=pd.read_csv("Documents/data/userskills.csv")
#remove duplicate columns during merge
dfuser = dfuser.loc[:,~dfuser.columns.duplicated()]
dfuser=dfuser.fillna(0)
dfuser.shape
dfuser.head()
#remove rows with less than 5 skills
dfuser=dfuser.dropna(thresh=5)
dfuser.shape
m=(np.asscalar(np.int32(max(dfuser["Respondent"]))))
print(m)
print(type(m))
#Build  a dictionary of respondent id's as keys  and thier skills as values
temp=[0]*m
vector=np.array(temp)
count=1
start=time.time()
d=dict()
for row in dfuser.iterrows():
    index,data=row
    l=list()
    #l=[data.values[0],list(data.values[1:])]
    s=np.asscalar(np.int32(data.values[0]))
    d[s]=np.array(list(data.values[1:]))
    #print(type(data))
end=time.time()
count1=1
count2=1
t=time.time()
for key, value in d.items():
    if(key<5000):
        #print(key)
        b=(np.linalg.norm(value))
        for key2,value2 in d.items():
            if(key2<5000):
                #print(key2)
                a=np.dot(d[key],d[key2])
```

```
                    ans=a/(np.linalg.norm(value2)*b)
                    sim[key][key2]=ans
                    #count2+=1
            #count1+=1
e=time.time()
print("total time for building similarity matrix ")
print(e-t)
#For buiding  collaborative filtering based on The Content based recommendations for the first 200 Respondents.
dfcont=pd.read_csv("Documents/data/recommendations.csv")
sim=list()
s=time.time()
for i in range(200):
        l=list()
        l=[0]*200
        sim.append(l)
e=time.time()
print(e-s)
t=time.time()
print("Building the similarity matrix....\n")
for key, value in d.items():
        if(key<200):
            #print(key)
            b=(np.linalg.norm(value))
            for key2,value2 in d.items():
                if(key2<200):
                    #print(key2)
        a=np.dot(d[key],d[key2])
        ans=a/(np.linalg.norm(value2)*b)
        sim[key][key2]=ans
        #count2+=1
    #count1+=1
e=time.time()
print("Respondent 3 was recommended job titles from content based filtering ")
print(dfcont.loc[dfcont.Respondent==3]["jobtitle"].values)
print("\n")
m1=max(sim[3][:3])
print(m1)
m2=max(sim[3][100:])
print(m2)
ma=max(m1,m2)
print(ma)
suser=sim[3].index(ma)
print(suser) #suser is very similar to user 3 and hence we can recommend suser job to user 3.
print("Respondent ",suser,"was most similiar to respondent 3\n")
print("Based on respondent ",suser," the jobs recommended to 3 are ")
print(dfcont.loc[dfcont.Respondent==suser]["company"].values)
print("The recommended job titles are ")
print(dfcont.loc[dfcont.Respondent==suser]["jobtitle"].values)
```

1.6 Hybrid Filtering

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.cluster import AgglomerativeClustering
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import AgglomerativeClustering
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from scipy.cluster.hierarchy import dendrogram, ward
from sklearn.feature_extraction import text
from sklearn.metrics.pairwise import cosine_similarity
from pylab import rcParams
rcParams['figure.figsize'] = 50, 20
import nltk
from nltk.corpus import stopwords
df = pd.read_csv("Documents/data/job_sample.csv",encoding='cp850')
mine = ['manager', 'amp', 'nbsp', 'responsibilities', 'used', 'skills', 'duties', 'work',
'worked', 'daily','services', 'job', 'using', 'com', 'end', 'prepare', 'prepared', 'lead',
'requirements','summary','Job Role','Position']
vec=TfidfVectorizer(analyzer='word',ngram_range=(1,2),token_pattern='[a-
zAz]{3,50}', max_df=0.2, min_df=2, max_features=10000,
stop_words=text.ENGLISH_STOP_WORDS.union(mine),
decode_error='ignore', vocabulary=None, binary=False)
df['skills']=df['skills']+df['jobdescription']+df['jobtitle']
description_matrix2 = vec.fit_transform(df['skills'].values.astype('U'))
description_matrix2 = pd.DataFrame(description_matrix2.todense())
description_matrix2.columns = vec.get_feature_names()
vec2 = TfidfVectorizer(vocabulary=voc, decode_error='ignore')
```

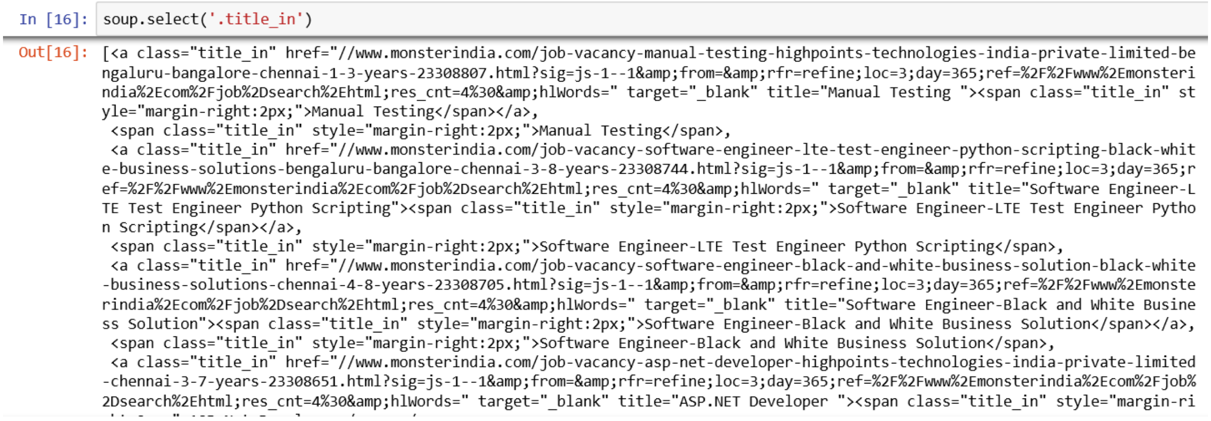
```
df['skills']=df['skills']+df['jobdescription']+df['jobtitle']
skills_matrix2 = vec2.fit_transform(df['skills'].values.astype('U'))
skills_matrix2 = pd.DataFrame(skills_matrix2.todense())
skills_matrix2.columns = vec2.get_feature_names()
jobtitle_matrix = pd.concat([skills_matrix2, description_matrix2], axis=1)
# Run PCA to reduce number of features
pca = PCA(n_components=1000, random_state=42)
comps = pca.fit_transform(jobtitle_matrix)
# Put the components into a dataframe
comps = pd.DataFrame(comps)
# Cluster job titles based on components derived from feature matrix
cltr = AgglomerativeClustering(n_clusters=8)
cltr.fit(comps)

# Add new column containing cluster number to sample, comps, and feature matrix dataframe
df['cluster_no'] = cltr.labels_
X = comps
y = df['cluster_no']
X_train, X_test, y_train, y_test = train_test_split(X,y, stratify=y, random_state=42)
lr = LogisticRegression(C=10, penalty='l2', multi_class='multinomial', solver='sag', max_iter=1000)
lr.fit(X_train, y_train)
score1=lr.score(X_test, y_test)
print(score1)
from sklearn.manifold import TSNE
```

```
tsne = TSNE()

g = pd.DataFrame(tsne.fit_transform(comps), columns=['one', 'two'])

g['cluster_no'] = cltr3.labels_

import matplotlib.cm as cm

plt.figure(figsize=(10,10))

plt.title('Clusters Using T-SNE Components', fontsize=20)

plt.xlabel('Component 1')

plt.ylabel('Component 2')

plt.scatter(g['one'], g['two'], c=g['cluster_no'], cmap=cm.jet, alpha=0.5)

plt.show()

# Assign cluster number to each job title in comps to pull particular cluster out for comparison

comps['cluster_no'] = y.values

comps.set_index('cluster_no', inplace=True)

def give_suggestions(resume_text):

    matches=dict()

    # Vectorize user's skills and job descriptions

    desc = pd.DataFrame(vec.transform([resume_text]).todense())

    desc.columns = vec.get_feature_names()

    skillz = pd.DataFrame(vec2.transform([resume_text]).todense())

    skillz.columns = vec2.get_feature_names()

    mat = pd.concat([skillz, desc], axis=1)

    # Tranform feature matrix with pca

    user_comps = pd.DataFrame(pca.transform(mat))


    # Predict cluster for user and print cluster number

    cluster = lr.predict(user_comps)[0]
    print ('CLUSTER NUMBER', cluster, '\n\n')

    # Calculate cosine similarity
    cos_sim =
pd.DataFrame(cosine_similarity(user_comps,comps[comps.index==cluster]))

    # Get job titles from df to associate cosine similarity scores with jobs
    samp_for_cluster = df[df['cluster_no']==cluster]
    cos_sim = cos_sim.T.set_index(samp_for_cluster['jobtitle'])
    cos_sim.columns = ['score']

    # Print the top ten suggested jobs for the user's cluster
    print ('Top ten suggested for your cluster', '\n', cos_sim.sort_values('score',
ascending=False)[:10], '\n\n')
   # print('Accuracy',)

    # Print the top five suggested jobs for each cluster
    mat = mat.T
    for i in range(8):
        cos_sim =
pd.DataFrame(cosine_similarity(user_comps,comps[comps.index==i]))
        samp_for_cluster = df[df['cluster_no']==i]
        cos_sim = cos_sim.T.set_index(samp_for_cluster['jobtitle'])
        cos_sim.columns = ['score']
        top_5 = cos_sim.sort_values('score', ascending=False)[:5]
        # Merge top_5 with sample2 to get skills and description
        merged_top_5 = top_5.merge(df, how='left', left_index=True, right_index=True)
        print ('---------Top five suggested in cluster', i,  '---------\n', top_5, '\n\n')
        # Vectorize to find skills needed for each job title
        for job in merged_top_5.index:

job_skills=pd.DataFrame(vec2.transform([merged_top_5.ix[job]['jobdescription']+merged_top_5.ix[job]['skills']]).todense())
            job_skills.columns = vec2.get_feature_names()
            job_skills = job_skills.T
            job_skills.columns = ['score']
            job_skills = job_skills[job_skills['score'] != 0].sort_values('score', ascending=False)
            mat.columns = ['score']
            mat = mat[mat['score'] != 0]
            needed_skills = []
            scorey = []
            for i in job_skills.index:
                if i not in mat.index:
                    needed_skills.append(i)
                    scorey.append(job_skills.ix[i][0])
            top_skills = pd.DataFrame(list(zip(needed_skills, scorey)), columns=['Skills', 'Importance'])
            print('To become a/an', job,',', '\n', 'these are the top ten skills you need:')
            print(top_skills[:5], '\n')
```

APPENDIX 2

2.1 Screenshots

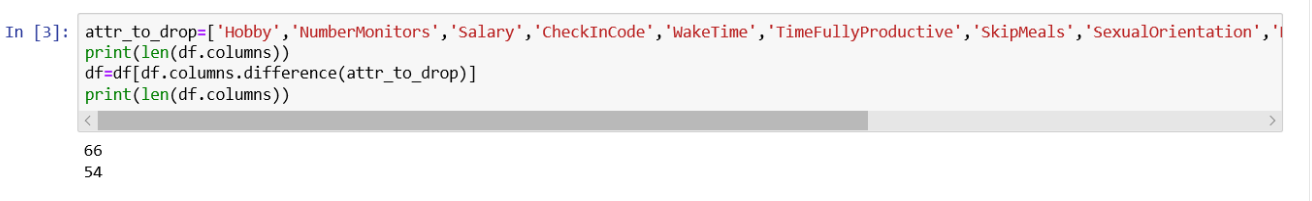
//webscarping



```
In [16]: soup.select('.title_in')
```

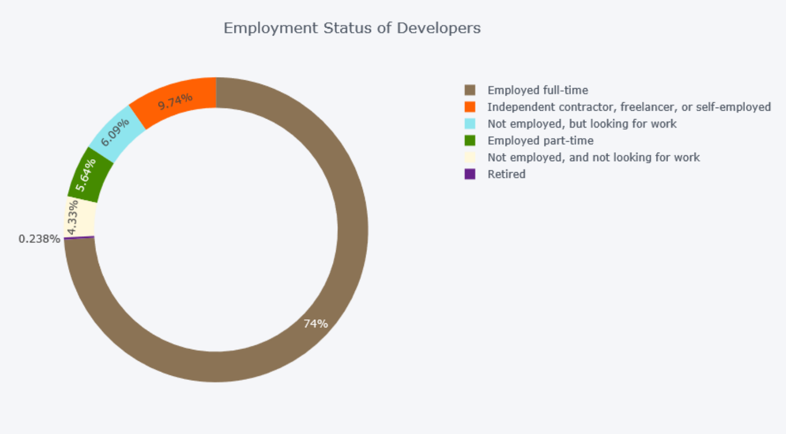
```
Out[16]: [<a class="title_in" href="//www.monsterindia.com/job-vacancy-manual-testing-highpoints-technologies-india-private-limited-be
ngaluru-bangalore-chennai-1-3-years-23308807.html?sig=js-1--1&amp;from=&amp;rfr=refine;loc=3;day=365;ref=%2F%2Fwww%2Emonsteri
ndia%2Ecom%2Fjob%2Dsearch%2Ehtml;res_cnt=4%30&amp;hlWords=" target="_blank" title="Manual Testing "><span class="title_in" st
yle="margin-right:2px;">Manual Testing</span></a>,
 <span class="title_in" style="margin-right:2px;">Manual Testing</span>,
 <a class="title_in" href="//www.monsterindia.com/job-vacancy-software-engineer-lte-test-engineer-python-scripting-black-whit
e-business-solutions-bengaluru-bangalore-chennai-3-8-years-23308744.html?sig=js-1--1&amp;from=&amp;rfr=refine;loc=3;day=365;r
ef=%2F%2Fwww%2Emonsterindia%2Ecom%2Fjob%2Dsearch%2Ehtml;res_cnt=4%30&amp;hlWords=" target="_blank" title="Software Engineer-L
TE Test Engineer Python Scripting"><span class="title_in" style="margin-right:2px;">Software Engineer-LTE Test Engineer Pytho
n Scripting</span></a>,
 <span class="title_in" style="margin-right:2px;">Software Engineer-LTE Test Engineer Python Scripting</span>,
 <a class="title_in" href="//www.monsterindia.com/job-vacancy-software-engineer-black-and-white-business-solution-black-white
-business-solutions-chennai-4-8-years-23308705.html?sig=js-1--1&amp;from=&amp;rfr=refine;loc=3;day=365;ref=%2F%2Fwww%2Emonste
rindia%2Ecom%2Fjob%2Dsearch%2Ehtml;res_cnt=4%30&amp;hlWords=" target="_blank" title="Software Engineer-Black and White Busine
ss Solution"><span class="title_in" style="margin-right:2px;">Software Engineer-Black and White Business Solution</span></a>,
 <span class="title_in" style="margin-right:2px;">Software Engineer-Black and White Business Solution</span>,
 <a class="title_in" href="//www.monsterindia.com/job-vacancy-asp-net-developer-highpoints-technologies-india-private-limited
-chennai-3-7-years-23308651.html?sig=js-1--1&amp;from=&amp;rfr=refine;loc=3;day=365;ref=%2F%2Fwww%2Emonsterindia%2Ecom%2Fjob%
2Dsearch%2Ehtml;res_cnt=4%30&amp;hlWords=" target="_blank" title="ASP.NET Developer "><span class="title_in" style="margin-ri
```

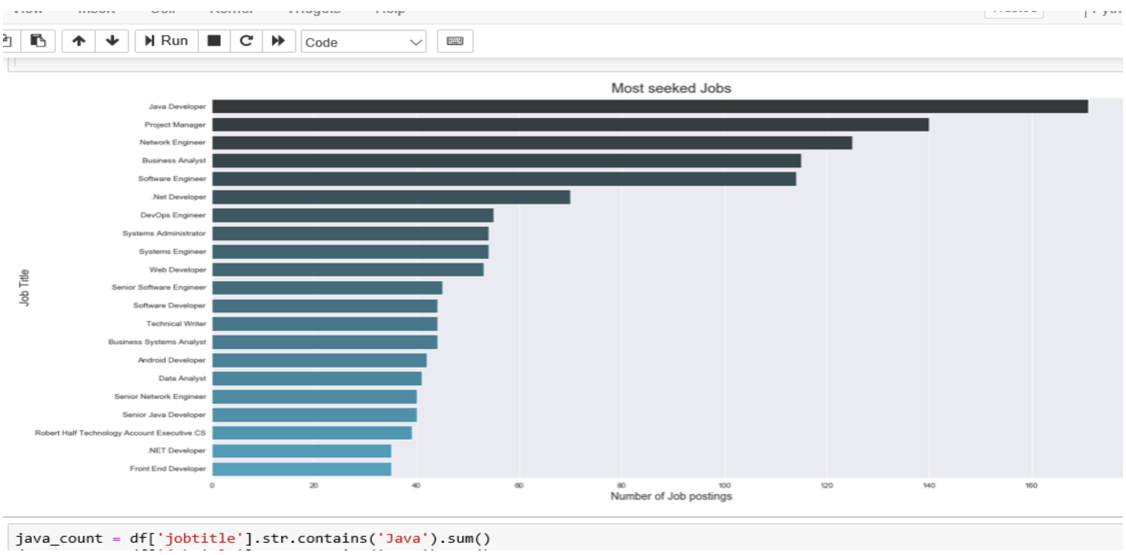
Figure 7.2.1 Output Screenshot 1



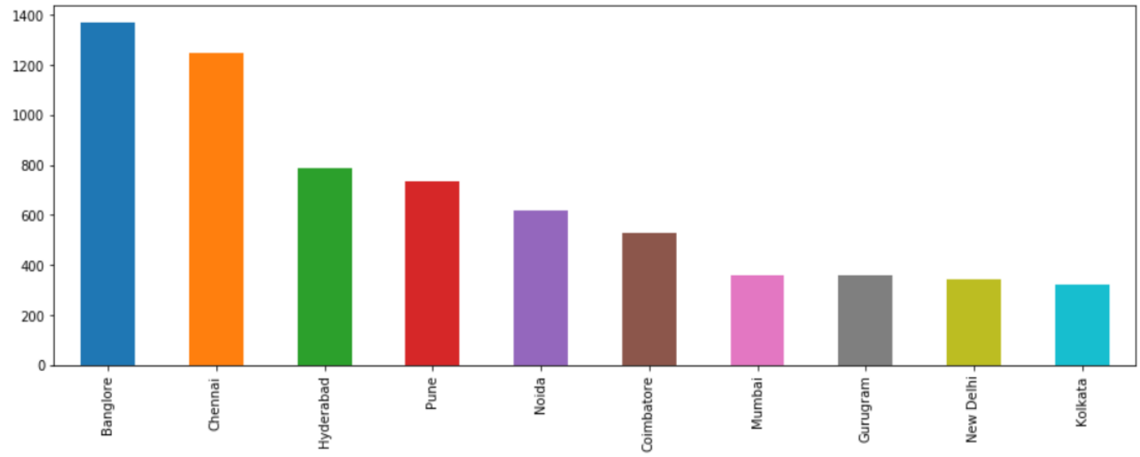
```
In [3]: attr_to_drop=['Hobby','NumberMonitors','Salary','CheckInCode','WakeTime','TimeFullyProductive','SkipMeals','SexualOrientation','
print(len(df.columns))
df=df[df.columns.difference(attr_to_drop)]
print(len(df.columns))
```

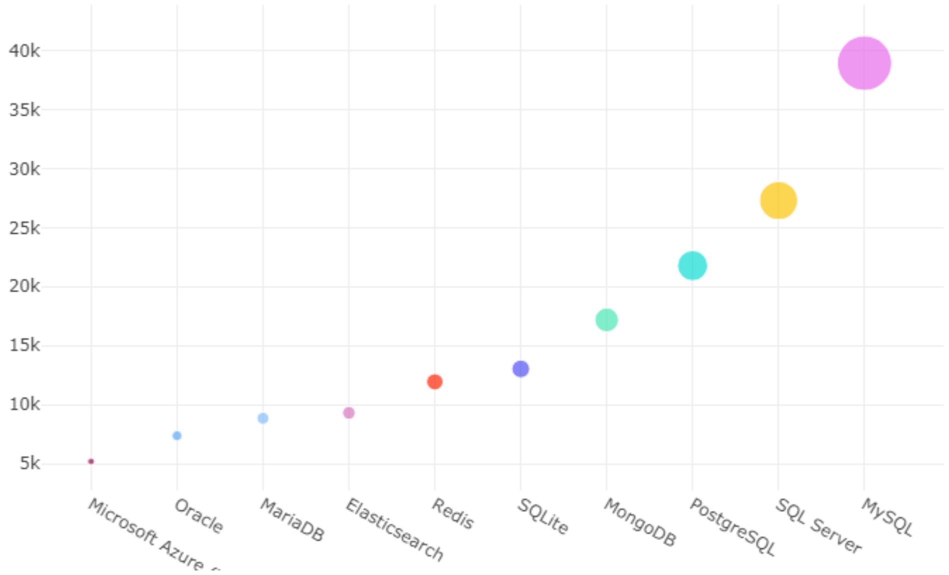
```
66
54
```





```
java_count = df['jobtitle'].str.contains('Java').sum()
```





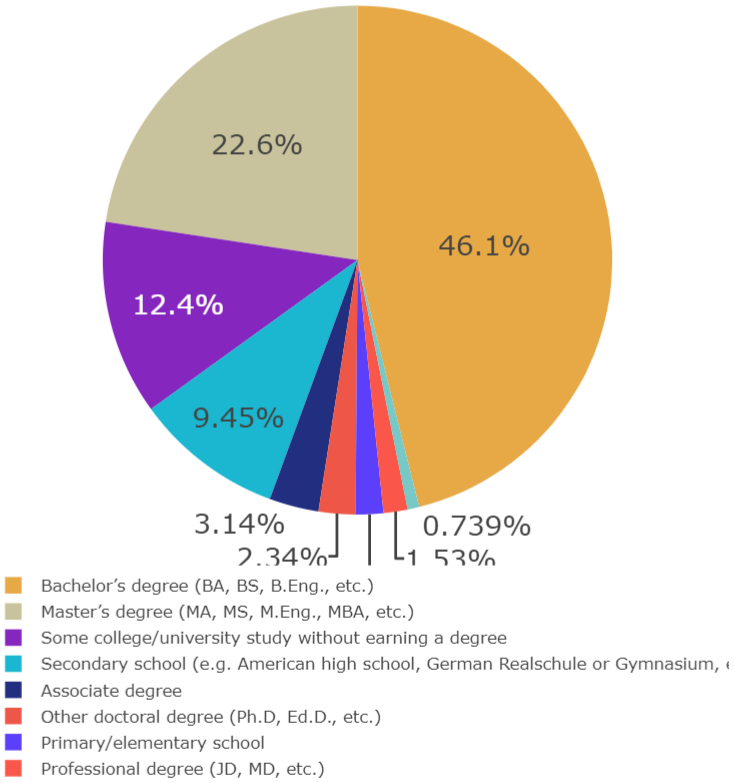
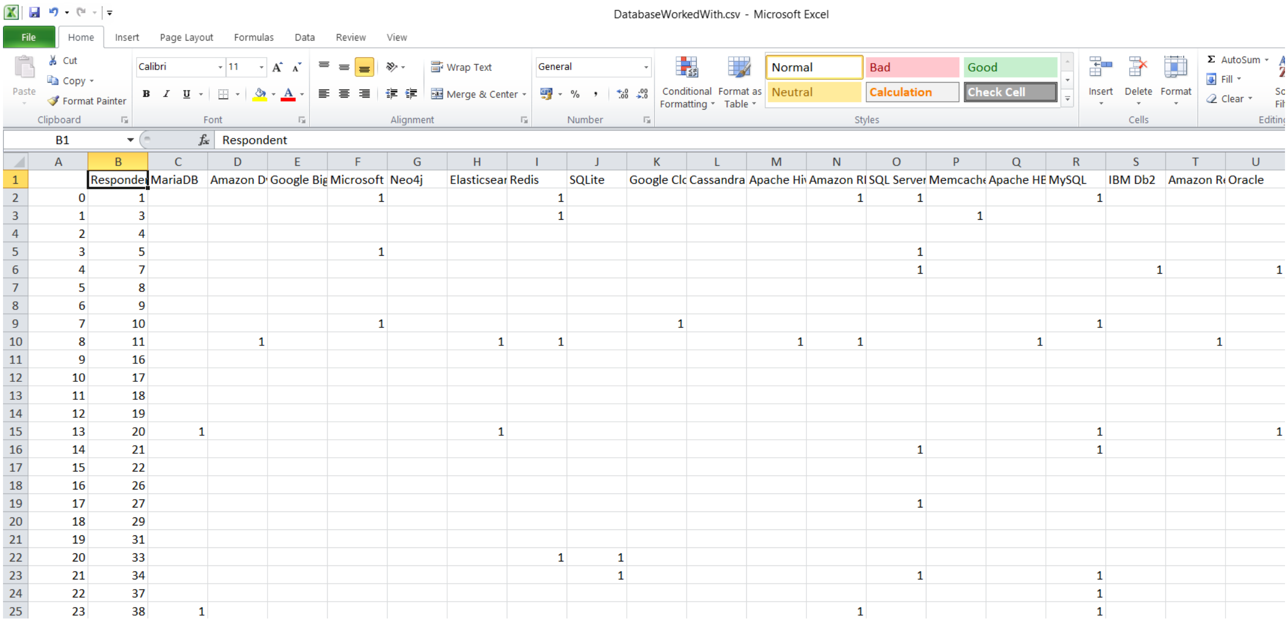
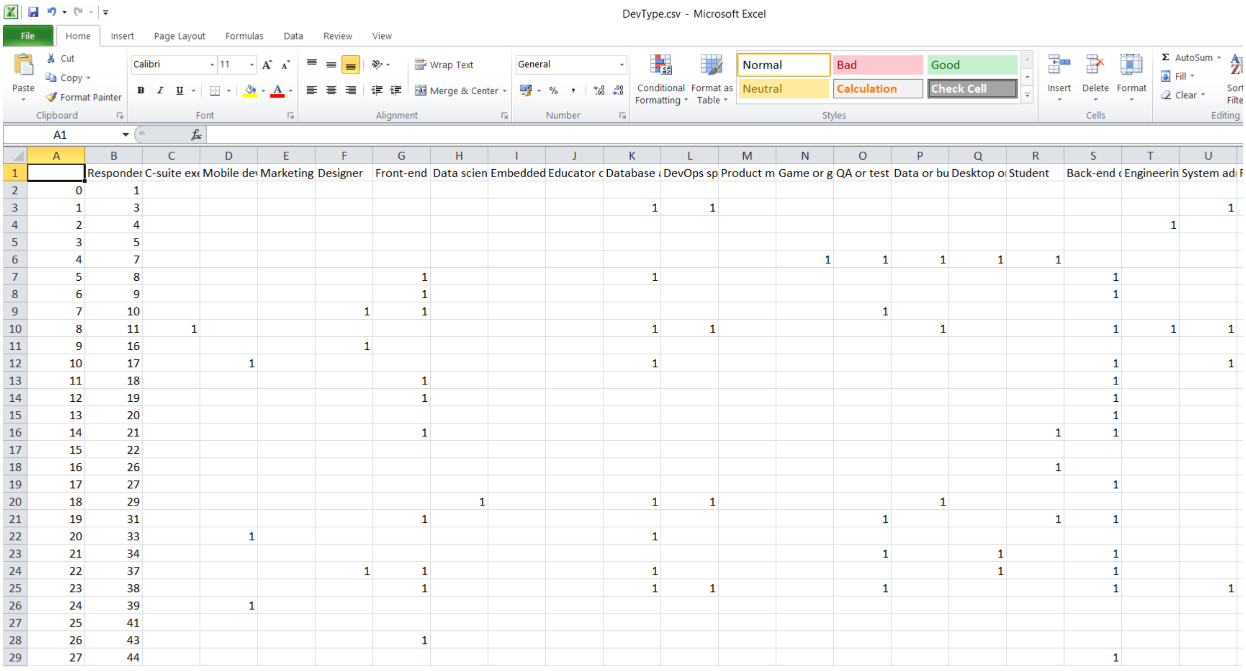


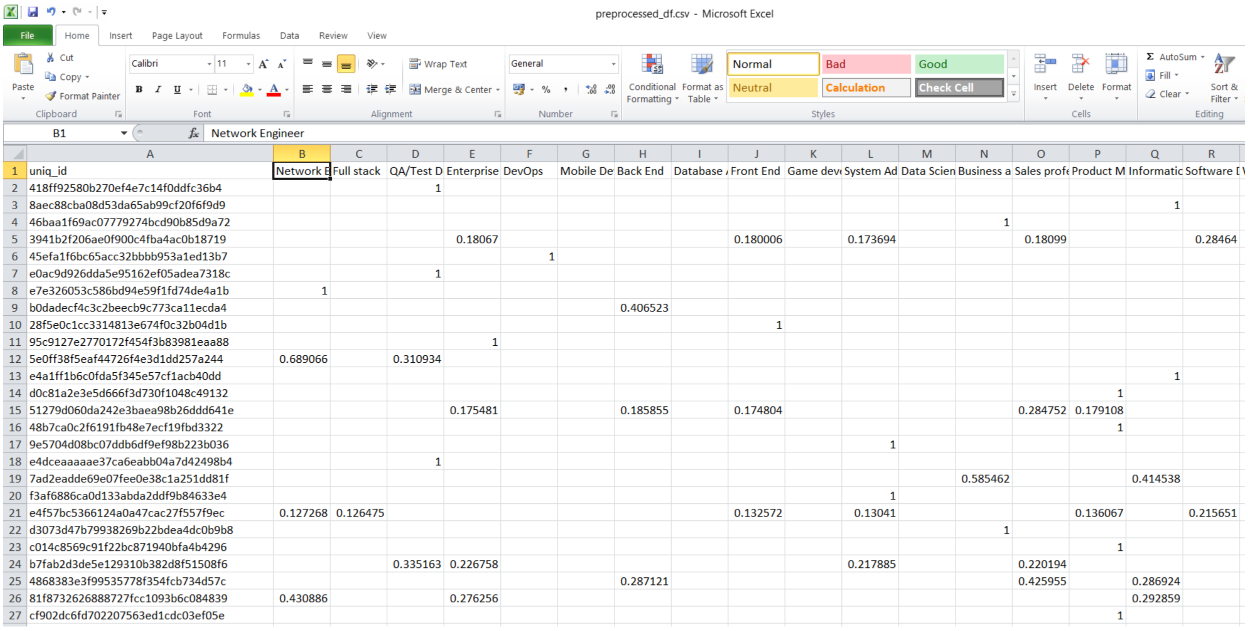
Figure 7.2.1 Output Screenshot 1



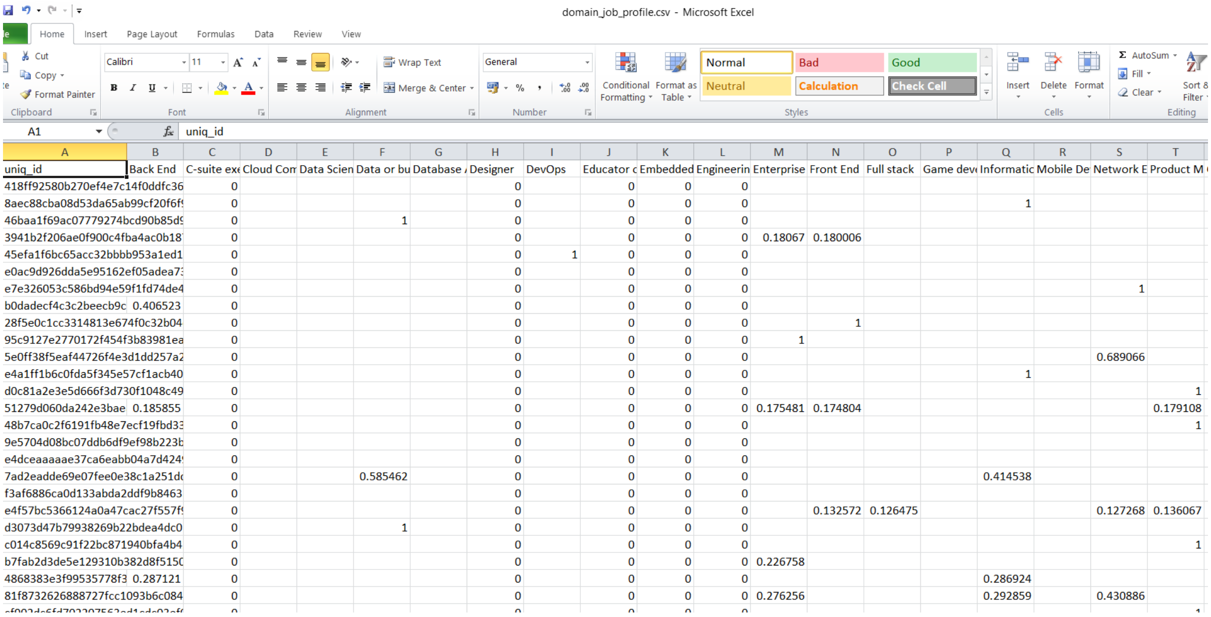
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		Responde	MariaDB	Amazon D	Google Big	Microsoft	Neo4j	Elasticsear	Redis	SQLite	Google Clc	Cassandra	Apache Hi	Amazon RI	SQL Server	Memcache	Apache HE	MySQL	IBM Db2	Amazon R	Oracle
2	0	1				1			1					1	1			1			
3	1	3							1							1					
4	2	4																			
5	3	5				1									1						
6	4	7													1				1		1
7	5	8																			
8	6	9																			
9	7	10				1					1							1			
10	8	11		1				1	1				1	1			1			1	
11	9	16																			
12	10	17																			
13	11	18																			
14	12	19																			
15	13	20	1					1										1			1
16	14	21													1			1			
17	15	22																			
18	16	26																			
19	17	27													1						
20	18	29																			
21	19	31																			
22	20	33							1	1											
23	21	34								1					1			1			
24	22	37																1			
25	23	38	1											1				1			



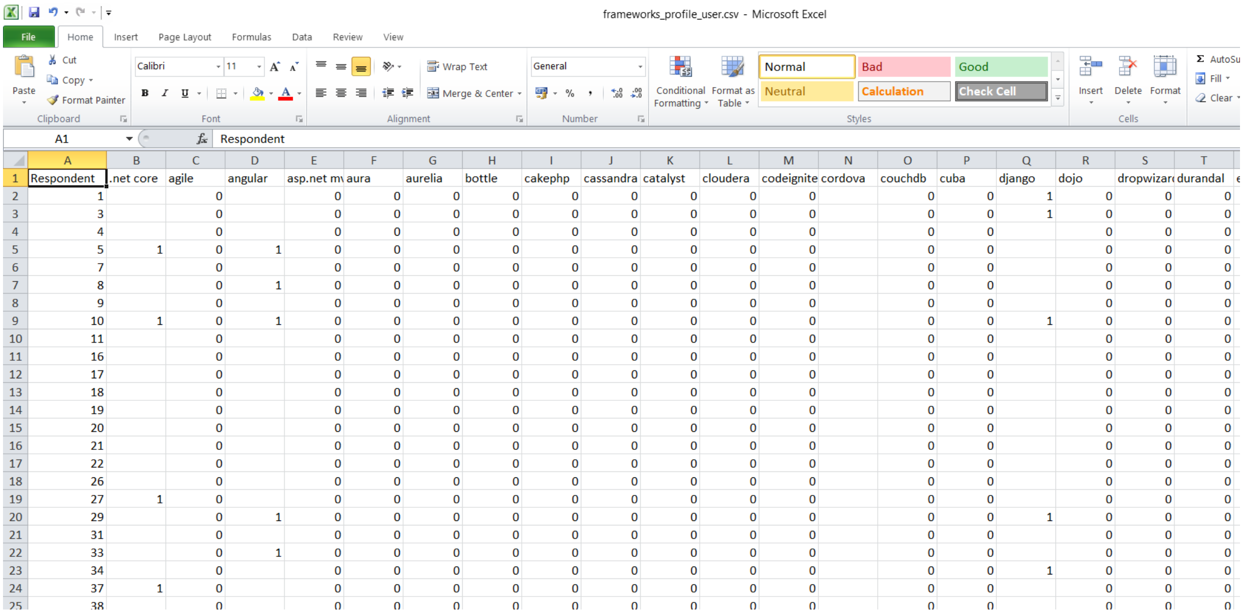
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		Responder	C-suite exe	Mobile de	Marketing	Designer	Front-end	Data scien	Embedded	Educator c	Database a	DevOps sp	Product m	Game or g	QA or test	Data or bu	Desktop o	Student	Back-end c	Engineerin	System adı
2	0	1																			
3	1	3									1	1									1
4	2	4																		1	
5	3	5																			
6	4	7												1	1	1	1	1			
7	5	8					1				1								1		
8	6	9					1												1		
9	7	10				1	1								1						
10	8	11	1								1	1				1			1	1	1
11	9	16				1															
12	10	17		1							1								1		1
13	11	18					1												1		
14	12	19					1												1		
15	13	20																	1		
16	14	21					1											1	1		
17	15	22																			
18	16	26																1			
19	17	27																	1		
20	18	29						1			1	1				1					
21	19	31					1								1			1	1		
22	20	33		1							1										
23	21	34													1		1		1		
24	22	37				1	1				1						1		1		
25	23	38					1				1	1			1				1		1
26	24	39		1																	
27	25	41																			
28	26	43					1														
29	27	44																	1		



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	uniq_id	Network E	Full stack	QA/Test D	Enterprise	DevOps	Mobile De	Back End	Database A	Front End	Game deve	System Ad	Data Scien	Business a	Sales profe	Product M	Informatic	Software D
2	418ff92580b270ef4e7c14f0ddfc36b4			1														
3	8aec88cba08d53da65ab99cf20f6f9d9																1	
4	46baa1f69ac07779274bcd90b85d9a72													1				
5	3941b2f206ae0f900c4fba4ac0b18719				0.18067					0.180006		0.173694			0.18099			0.28464
6	45efa1f6bc65acc32bbbb953a1ed13b7					1												
7	e0ac9d926dda5e95162ef05adea7318c			1														
8	e7e326053c586bd94e59f1fd74de4a1b	1																
9	b0dadecf4c3c2beecb9c773ca11ecda4							0.406523										
10	28f5e0c1cc3314813e674f0c32b04d1b									1								
11	95c9127e2770172f454f3b83981eaa88				1													
12	5e0ff38f5eaf44726f4e3d1dd257a244	0.689066		0.310934														
13	e4a1ff1b6c0fda5f345e57cf1acb40dd																1	
14	d0c81a2e3e5d666f3d730f1048c49132															1		
15	51279d060da242e3baea98b26ddd641e				0.175481			0.185855		0.174804					0.284752	0.179108		
16	48b7ca0c2f6191fb48e7ecf19fbd3322															1		
17	9e5704d08bc07ddb6df9ef98b223b036											1						
18	e4dceaaaaae37ca6eabb04a7d42498b4			1														
19	7ad2eadde69e07fee0e38c1a251dd81f													0.585462			0.414538	
20	f3af6886ca0d133abda2ddf9b84633e4											1						
21	e4f57bc5366124a0a47cac27f557f9ec	0.127268	0.126475							0.132572		0.13041				0.136067		0.215651
22	d3073d47b79938269b22bdea4dc0b9b8													1				
23	c014c8569c91f22bc871940bfa4b4296															1		
24	b7fab2d3de5e129310b382d8f51508f6			0.335163	0.226758							0.217885			0.220194			
25	4868383e3f99535778f354fcb734d57c							0.287121							0.425955		0.286924	
26	81f8732626888727fcc1093b6c084839	0.430886			0.276256												0.292859	
27	cf902dc6fd702207563ed1cdc03ef05e															1		

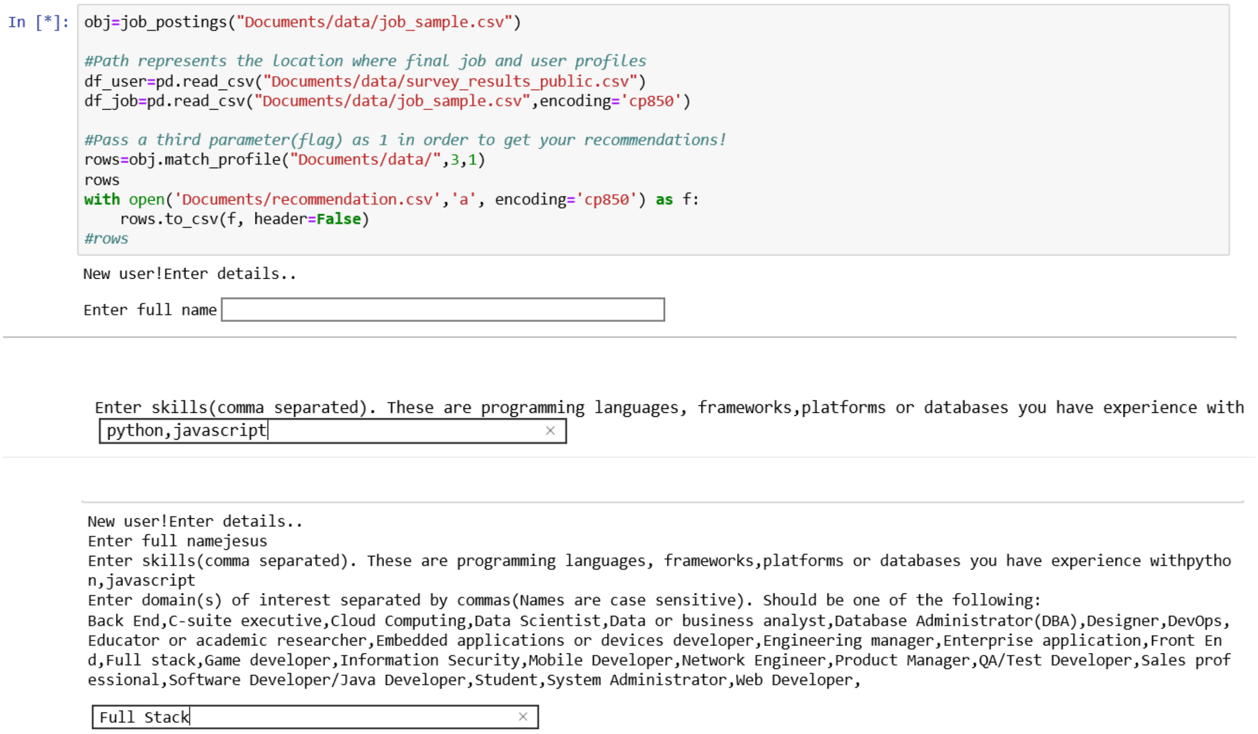


A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
uniq_id	Back End	C-suite exe	Cloud Con	Data Scien	Data or bu	Database /	Designer	DevOps	Educator c	Embedded	Engineerin	Enterprise	Front End	Full stack	Game dev	Informatic	Mobile De	Network E	Product M
418ff92580b270ef4e7c14f0ddfc36		0					0		0	0	0								
8aec88cba08d53da65ab99cf20f6f		0					0		0	0	0					1			
46baa1f69ac07779274bcd90b85d		0			1		0		0	0	0								
3941b2f206ae0f900c4fba4ac0b18		0					0		0	0	0	0.18067	0.180006						
45efa1f6bc65acc32bbbb953a1ed1		0					0	1	0	0	0								
e0ac9d926dda5e95162ef05adea7		0					0		0	0	0								
e7e326053c586bd94e59f1fd74de4		0					0		0	0	0							1	
b0dadecf4c3c2beecb9c	0.406523	0					0		0	0	0								
28f5e0c1cc3314813e674f0c32b04		0					0		0	0	0		1						
95c9127e2770172f454f3b83981ea		0					0		0	0	0	1							
5e0ff38f5eaf44726f4e3d1dd257a		0					0		0	0	0							0.689066	
e4a1ff1b6c0fda5f345e57cf1acb40		0					0		0	0	0					1			
d0c81a2e3e5d666f3d730f1048c49		0					0		0	0	0								1
51279d060da242e3bae	0.185855	0					0		0	0	0	0.175481	0.174804						0.179108
48b7ca0c2f6191fb48e7ecf19fbd3		0					0		0	0	0								1
9e5704d08bc07ddb6df9ef98b223b		0					0		0	0	0								
e4dceaaaaae37ca6eabb04a7d424		0					0		0	0	0								
7ad2eadde69e07fee0e38c1a251d		0			0.585462		0		0	0	0					0.414538			
f3af6886ca0d133abda2ddf9b8463		0					0		0	0	0								
e4f57bc5366124a0a47cac27f557f		0					0		0	0	0		0.132572	0.126475				0.127268	0.136067
d3073d47b79938269b22bdea4dc0		0			1		0		0	0	0								
c014c8569c91f22bc871940bfa4b4		0					0		0	0	0								1
b7fab2d3de5e129310b382d8f515		0					0		0	0	0	0.226758							
4868383e3f99535778f3	0.287121	0					0		0	0	0					0.286924			
81f8732626888727fcc1093b6c084		0					0		0	0	0	0.276256				0.292859		0.430886	



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Respondent	.net core	agile	angular	asp.net mv	aura	aurelia	bottle	cakephp	cassandra	catalyst	cloudera	codeignite	cordova	couchdb	cuba	django	dojo	dropwizar	durandal
2	1		0		0	0	0	0	0	0	0	0	0		0	0	1	0	0	0
3	3		0		0	0	0	0	0	0	0	0	0		0	0	1	0	0	0
4	4		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
5	5	1	0	1	0	0	0	0	0	0	0	0	0		0	0		0	0	0
6	7		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
7	8		0	1	0	0	0	0	0	0	0	0	0		0	0		0	0	0
8	9		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
9	10	1	0	1	0	0	0	0	0	0	0	0	0		0	0	1	0	0	0
10	11		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
11	16		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
12	17		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
13	18		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
14	19		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
15	20		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
16	21		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
17	22		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
18	26		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
19	27	1	0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
20	29		0	1	0	0	0	0	0	0	0	0	0		0	0	1	0	0	0
21	31		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
22	33		0	1	0	0	0	0	0	0	0	0	0		0	0		0	0	0
23	34		0		0	0	0	0	0	0	0	0	0		0	0	1	0	0	0
24	37	1	0		0	0	0	0	0	0	0	0	0		0	0		0	0	0
25	38		0		0	0	0	0	0	0	0	0	0		0	0		0	0	0

//content



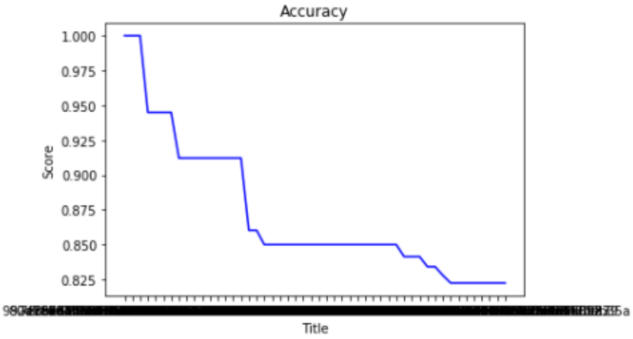
```
In [*]: obj=job_postings("Documents/data/job_sample.csv")

        #Path represents the location where final job and user profiles
        df_user=pd.read_csv("Documents/data/survey_results_public.csv")
        df_job=pd.read_csv("Documents/data/job_sample.csv",encoding='cp850')

        #Pass a third parameter(flag) as 1 in order to get your recommendations!
        rows=obj.match_profile("Documents/data/",3,1)
        rows
        with open('Documents/recommendation.csv','a', encoding='cp850') as f:
            rows.to_csv(f, header=False)
        #rows
```

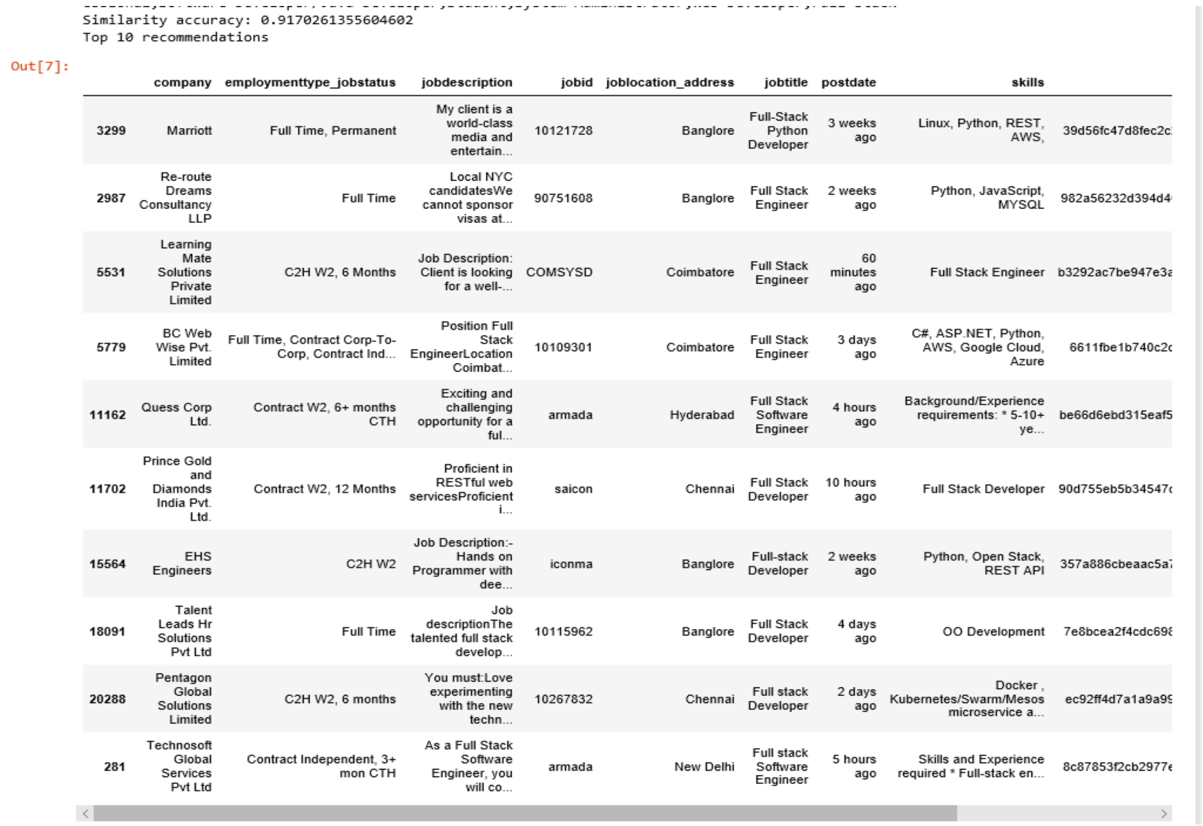
```
New user!Enter details..

Enter full name
```

```
Enter skills(comma separated). These are programming languages, frameworks,platforms or databases you have experience with
python,javascript
```

```
New user!Enter details..
Enter full namejesus
Enter skills(comma separated). These are programming languages, frameworks,platforms or databases you have experience withpytho
n,javascript
Enter domain(s) of interest separated by commas(Names are case sensitive). Should be one of the following:
Back End,C-suite executive,Cloud Computing,Data Scientist,Data or business analyst,Database Administrator(DBA),Designer,DevOps,
Educator or academic researcher,Embedded applications or devices developer,Engineering manager,Enterprise application,Front En
d,Full stack,Game developer,Information Security,Mobile Developer,Network Engineer,Product Manager,QA/Test Developer,Sales prof
essional,Software Developer/Java Developer,Student,System Administrator,Web Developer,

Full Stack
```

```
The Job ID and Score
 [('982a56232d394d4022fd854700f5b26e', 1.0), ('90d755eb5b34547c31978ce7a6eb519f', 1.0), ('7e8bcea2f4cdc698b6d614850ca397f2', 1.
0), ('8c87853f2cb2977e05df98497bcb7583', 0.9449489742783177), ('cc404090bcbd89fe3edeafd863880e73', 0.9449489742783177), ('9b134
ad623fad76e82ea501fa0b501b3', 0.9449489742783177), ('ed25e881bec334c636fc3edaba3272b2', 0.9449489742783177), ('c2158d3c93720eba
bbb2c3c59d4ebb9b', 0.9121320343559642), ('2f2a81f2128a30a806382049b03b5cbb', 0.9121320343559642), ('29ef164251d019ea20755d6243c
f839f', 0.9121320343559642), ('caadfe36bf38fe98525ee9b88e293794', 0.9121320343559642), ('39d56fc47d8fec2c2baa9e6e06142181', 0.9
121320343559642), ('236ad5bbac0eab634d9a27dbd4aa1bf2', 0.9121320343559642), ('5dd7bf5b767ca37a492e2a2d673befd0', 0.912132034355
9642), ('b0e8aa4357ca66ccc68d82c7febe639e', 0.9121320343559642), ('d34bcd54a2b8fb7c3cf539a4f24c2007', 0.9121320343559642), ('03
01e78babdcadd58319a570bec4e553', 0.8603567451474545), ('9e52951cfbef712f753153a6f5c2fff5', 0.8603567451474545), ('f273102f1405b
da8d043fac13e5c4ad1', 0.85), ('19efe1e35d5dd810857a855bf2fc238c', 0.85), ('5ae372b53b9dcf9cc23e553d3c79a700', 0.85), ('dd3204d1
be688ea884dbc134c232a6d6', 0.85), ('b3292ac7be947e3a89b0853f4a7e2577', 0.85), ('599df09a435ef034b31a2bcc789bc7a3', 0.85), ('661
1fbe1b740c2c29d4365ff9d60eefa', 0.85), ('a0132b7d5002d30a95f548d03081e66b', 0.85), ('be66d6ebd315eaf5c98c522375c8e1e5', 0.85),
('1469f44425bae8d7cc4cfa00fb9f9bf0', 0.85), ('7172c43d0d2642e432d3fd7319bcc946', 0.85), ('357a886cbeaac5a74c53a11665f6a45c', 0.
85), ('deb8ac91cb527574b00ae99f49cf1b3d', 0.85), ('ec92ff4d7a1a9a99e49f5792e1be2582', 0.85), ('27e2a223036c9fda11a06a93876b6a4
d', 0.85), ('d693200c176d779af4c5a2bf480ad11f', 0.85), ('f560a38109786ee7e79b8391cc0ec0f1', 0.85), ('56b2dd205154a1cb2eae101efc
d14567', 0.85), ('571a08dea5b08f0b89ba5a8c1766d468', 0.8414213562373094), ('9720546dece0d5ee07bfe88622c93052', 0.8414213562373O
94), ('7e71d56a0a0f62d3e4e062b84131da29', 0.8414213562373094), ('4b072aab5c96491c2d7f6731772b8825', 0.8341640786499873), ('8233
7c1b741dd2876fdb4d0137b42ab6', 0.8341640786499873), ('ef30e93383d71631c578bbf1496d1470', 0.8279204298133662), ('b9520049ff3e799
ec368ae0aa99ec5f5', 0.8224744871391588), ('9faf5a21b71de9c9dc205c9c3fbcc584', 0.8224744871391588), ('2bb34bd5d3bf132d865f39ed7c
f95de8', 0.8224744871391588), ('80b39037106ea5955c7487c7b3c832cc', 0.8224744871391588), ('7c5fd03c082311c03d9cc05f9be05092', 0.
8224744871391588), ('a0fe08d317ddaaa464e2fc0e4b1a50a7', 0.8224744871391588), ('d5c79c26122adbdcf4f213b36508b839', 0.82247448713
91588), ('24e316b6358999e1b5cd69864509b35a', 0.8224744871391588)]

total time for building similarity matrix
38.92397379875183
```



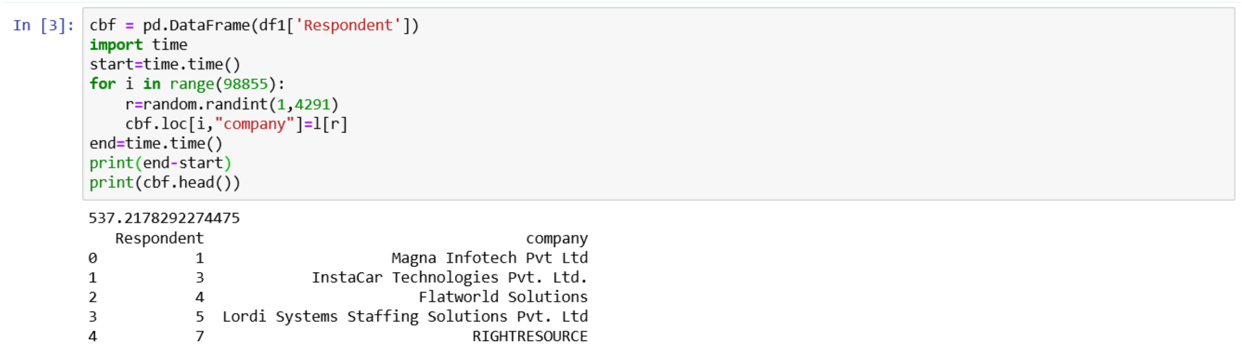


```
Similarity accuracy: 0.9170261355604602
Top 10 recommendations
```

Out[7]:

	company	employmenttype_jobstatus	jobdescription	jobid	joblocation_address	jobtitle	postdate	skills	
3299	Marriott	Full Time, Permanent	My client is a world-class media and entertain...	10121728	Banglore	Full-Stack Python Developer	3 weeks ago	Linux, Python, REST, AWS,	39d56fc47d8fec2c
2987	Re-route Dreams Consultancy LLP	Full Time	Local NYC candidatesWe cannot sponsor visas at...	90751608	Banglore	Full Stack Engineer	2 weeks ago	Python, JavaScript, MYSQL	982a56232d394d4
5531	Learning Mate Solutions Private Limited	C2H W2, 6 Months	Job Description: Client is looking for a well-...	COMSYSD	Coimbatore	Full Stack Engineer	60 minutes ago	Full Stack Engineer	b3292ac7be947e3a
5779	BC Web Wise Pvt. Limited	Full Time, Contract Corp-To-Corp, Contract Ind...	Position Full Stack EngineerLocation Coimbat...	10109301	Coimbatore	Full Stack Engineer	3 days ago	C#, ASP.NET, Python, AWS, Google Cloud, Azure	6611fbe1b740c2c
11162	Quess Corp Ltd.	Contract W2, 6+ months CTH	Exciting and challenging opportunity for a ful...	armada	Hyderabad	Full Stack Software Engineer	4 hours ago	Background/Experience requirements: * 5-10+ ye...	be66d6ebd315eaf5
11702	Prince Gold and Diamonds India Pvt. Ltd.	Contract W2, 12 Months	Proficient in RESTful web servicesProficient i...	saicon	Chennai	Full Stack Developer	10 hours ago	Full Stack Developer	90d755eb5b34547
15564	EHS Engineers	C2H W2	Job Description:- Hands on Programmer with dee...	iconma	Banglore	Full-stack Developer	2 weeks ago	Python, Open Stack, REST API	357a886cbeaac5a
18091	Talent Leads Hr Solutions Pvt Ltd	Full Time	Job descriptionThe talented full stack develop...	10115962	Banglore	Full Stack Developer	4 days ago	OO Development	7e8bcea2f4cdc69
20288	Pentagon Global Solutions Limited	C2H W2, 6 months	You must:Love experimenting with the new techn...	10267832	Chennai	Full stack Developer	2 days ago	Docker , Kubernetes/Swarm/Mesos microservice a...	ec92ff4d7a1a9a9
281	Technosoft Global Services Pvt Ltd	Contract Independent, 3+ mon CTH	As a Full Stack Software Engineer, you will co...	armada	New Delhi	Full stack Software Engineer	5 hours ago	Skills and Experience required * Full-stack en...	8c87853f2cb2977

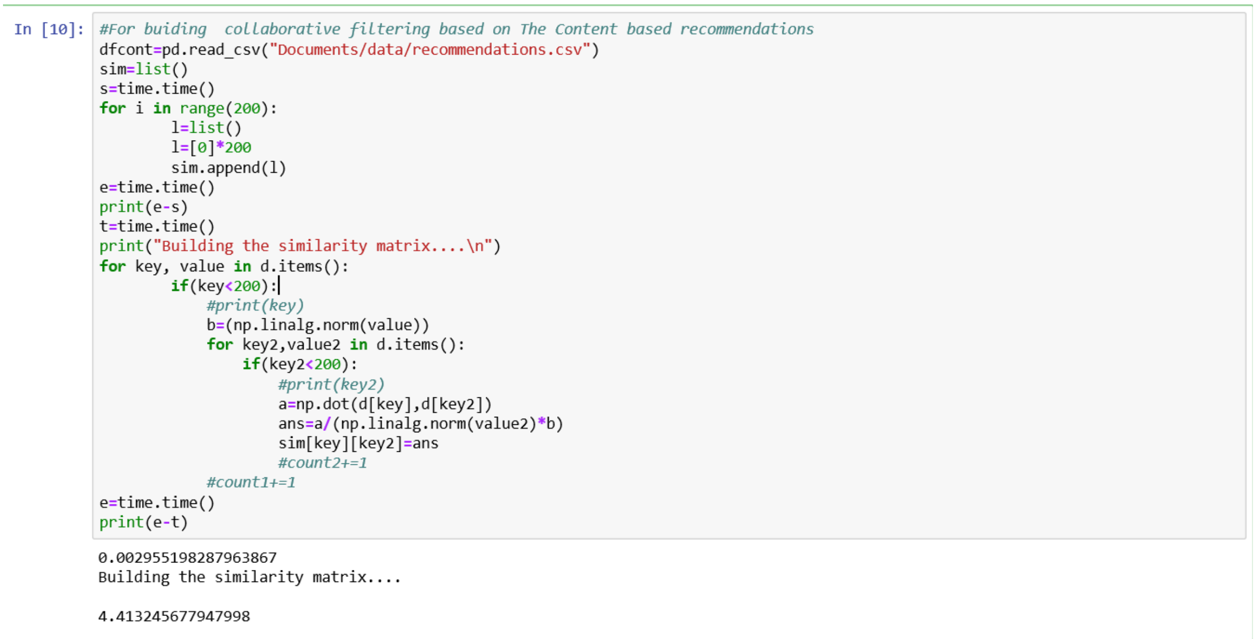
//collab



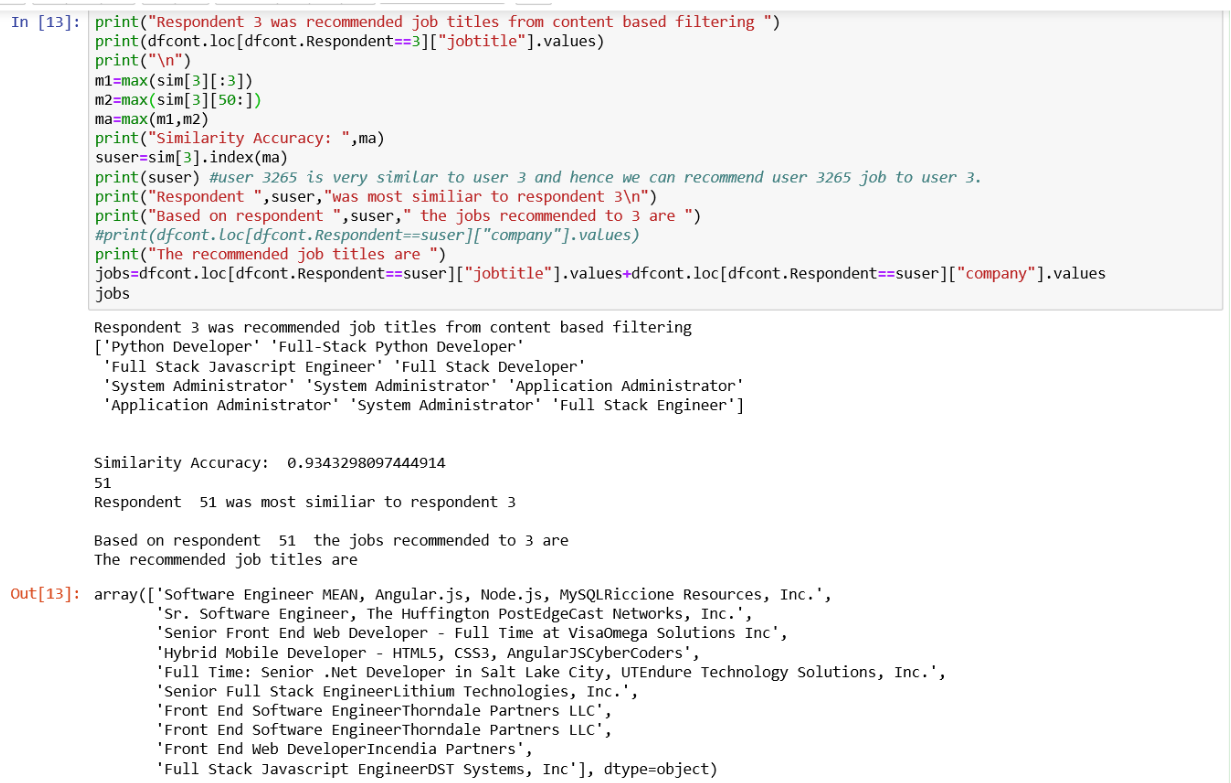
In [3]:

```python
cbf = pd.DataFrame(df1['Respondent'])
import time
start=time.time()
for i in range(98855):
    r=random.randint(1,4291)
    cbf.loc[i,"company"]=l[r]
end=time.time()
print(end-start)
print(cbf.head())
```

```
537.2178292274475
   Respondent                              company
0           1                   Magna Infotech Pvt Ltd
1           3          InstaCar Technologies Pvt. Ltd.
2           4                      Flatworld Solutions
3           5  Lordi Systems Staffing Solutions Pvt. Ltd
4           7                           RIGHTRESOURCE
```

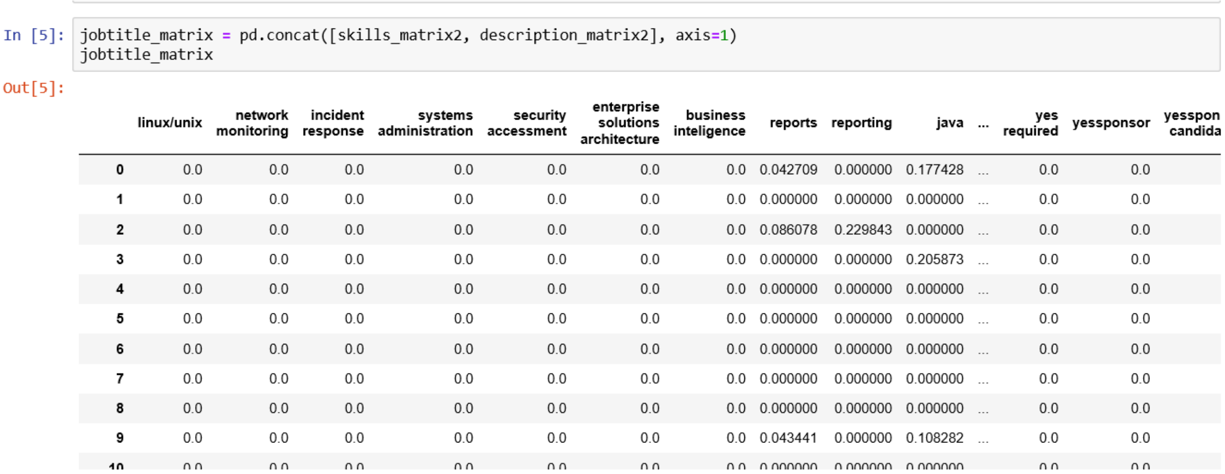


```
In [10]: #For buiding  collaborative filtering based on The Content based recomendations
         dfcont=pd.read_csv("Documents/data/recommendations.csv")
         sim=list()
         s=time.time()
         for i in range(200):
                 l=list()
                 l=[0]*200
                 sim.append(l)
         e=time.time()
         print(e-s)
         t=time.time()
         print("Building the similarity matrix....\n")
         for key, value in d.items():
                 if(key<200):
                     #print(key)
                     b=(np.linalg.norm(value))
                     for key2,value2 in d.items():
                         if(key2<200):
                             #print(key2)
                             a=np.dot(d[key],d[key2])
                             ans=a/(np.linalg.norm(value2)*b)
                             sim[key][key2]=ans
                             #count2+=1
                     #count1+=1
         e=time.time()
         print(e-t)

         0.002955198287963867
         Building the similarity matrix....

         4.413245677947998
```



```
In [13]: print("Respondent 3 was recommended job titles from content based filtering ")
         print(dfcont.loc[dfcont.Respondent==3]["jobtitle"].values)
         print("\n")
         m1=max(sim[3][:3])
         m2=max(sim[3][50:])
         ma=max(m1,m2)
         print("Similarity Accuracy: ",ma)
         suser=sim[3].index(ma)
         print(suser) #user 3265 is very similar to user 3 and hence we can recommend user 3265 job to user 3.
         print("Respondent ",suser,"was most similiar to respondent 3\n")
         print("Based on respondent ",suser," the jobs recommended to 3 are ")
         #print(dfcont.loc[dfcont.Respondent==suser]["company"].values)
         print("The recommended job titles are ")
         jobs=dfcont.loc[dfcont.Respondent==suser]["jobtitle"].values+dfcont.loc[dfcont.Respondent==suser]["company"].values
         jobs

         Respondent 3 was recommended job titles from content based filtering
         ['Python Developer' 'Full-Stack Python Developer'
          'Full Stack Javascript Engineer' 'Full Stack Developer'
          'System Administrator' 'System Administrator' 'Application Administrator'
          'Application Administrator' 'System Administrator' 'Full Stack Engineer']


         Similarity Accuracy:  0.9343298097444914
         51
         Respondent  51 was most similiar to respondent 3

         Based on respondent  51  the jobs recommended to 3 are
         The recommended job titles are

Out[13]: array(['Software Engineer MEAN, Angular.js, Node.js, MySQLRiccione Resources, Inc.',
                'Sr. Software Engineer, The Huffington PostEdgeCast Networks, Inc.',
                'Senior Front End Web Developer - Full Time at VisaOmega Solutions Inc',
                'Hybrid Mobile Developer - HTML5, CSS3, AngularJSCyberCoders',
                'Full Time: Senior .Net Developer in Salt Lake City, UTEndure Technology Solutions, Inc.',
                'Senior Full Stack EngineerLithium Technologies, Inc.',
                'Front End Software EngineerThorndale Partners LLC',
                'Front End Software EngineerThorndale Partners LLC',
                'Front End Web DeveloperIncendia Partners',
                'Full Stack Javascript EngineerDST Systems, Inc'], dtype=object)
```

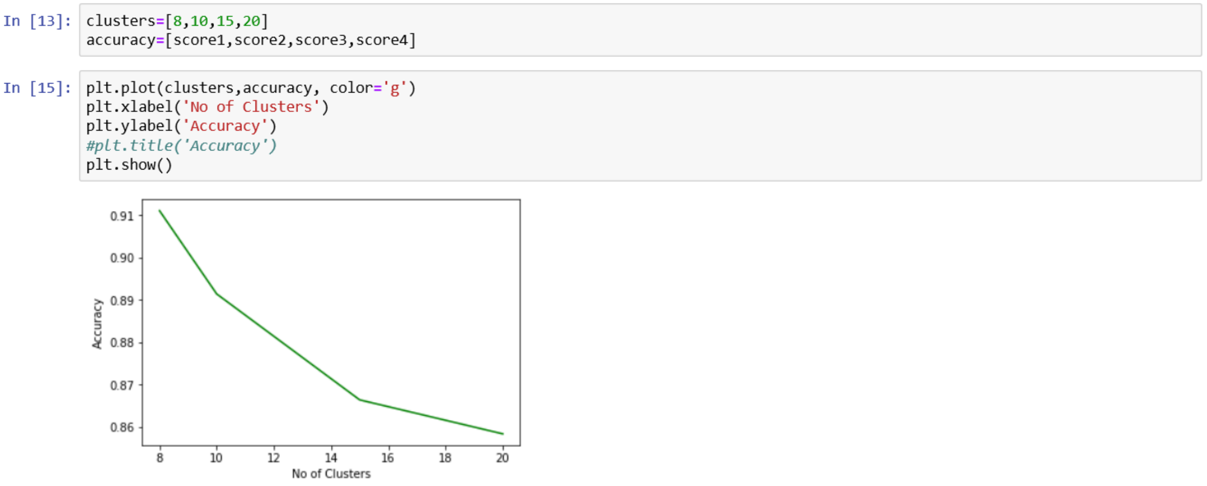
//hybrid



```
In [5]: jobtitle_matrix = pd.concat([skills_matrix2, description_matrix2], axis=1)
        jobtitle_matrix
```

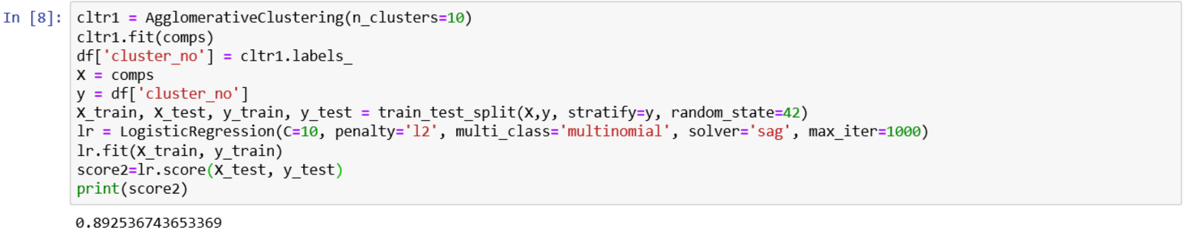
Out[5]:

	linux/unix	network monitoring	incident response	systems administration	security accessment	enterprise solutions architecture	business inteligence	reports	reporting	java	...	yes required	yessponsor	yesspon candida
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.042709	0.000000	0.177428	...	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	...	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.086078	0.229843	0.000000	...	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.205873	...	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	...	0.0	0.0	
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	...	0.0	0.0	
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	...	0.0	0.0	
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	...	0.0	0.0	
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	...	0.0	0.0	
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.043441	0.000000	0.108282	...	0.0	0.0	
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	...	0.0	0.0	



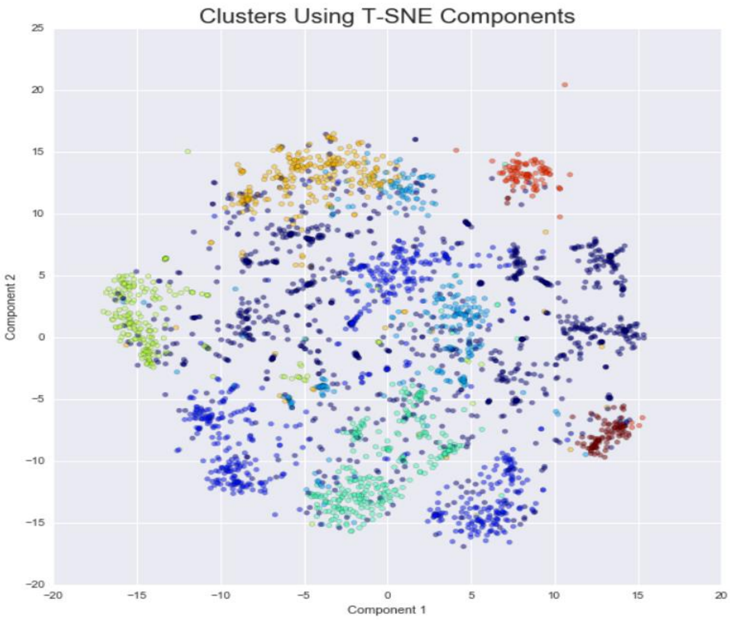
```
In [13]: clusters=[8,10,15,20]
         accuracy=[score1,score2,score3,score4]
```

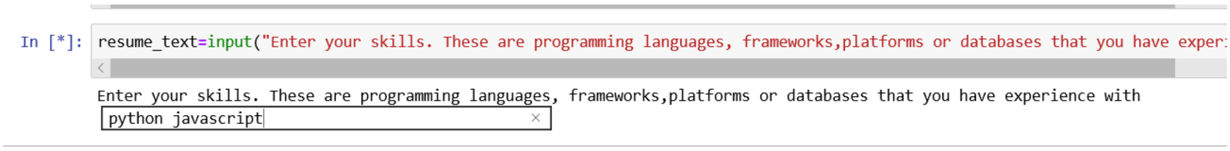
```
In [15]: plt.plot(clusters,accuracy, color='g')
         plt.xlabel('No of Clusters')
         plt.ylabel('Accuracy')
         #plt.title('Accuracy')
         plt.show()
```



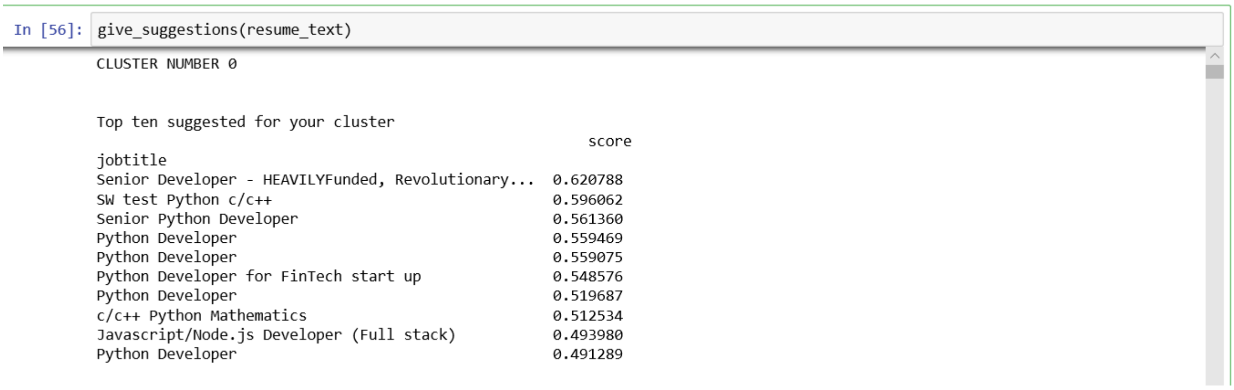
```
In [8]: cltr1 = AgglomerativeClustering(n_clusters=10)
        cltr1.fit(comps)
        df['cluster_no'] = cltr1.labels_
        X = comps
        y = df['cluster_no']
        X_train, X_test, y_train, y_test = train_test_split(X,y, stratify=y, random_state=42)
        lr = LogisticRegression(C=10, penalty='l2', multi_class='multinomial', solver='sag', max_iter=1000)
        lr.fit(X_train, y_train)
        score2=lr.score(X_test, y_test)
        print(score2)
```

```
0.892536743653369
```

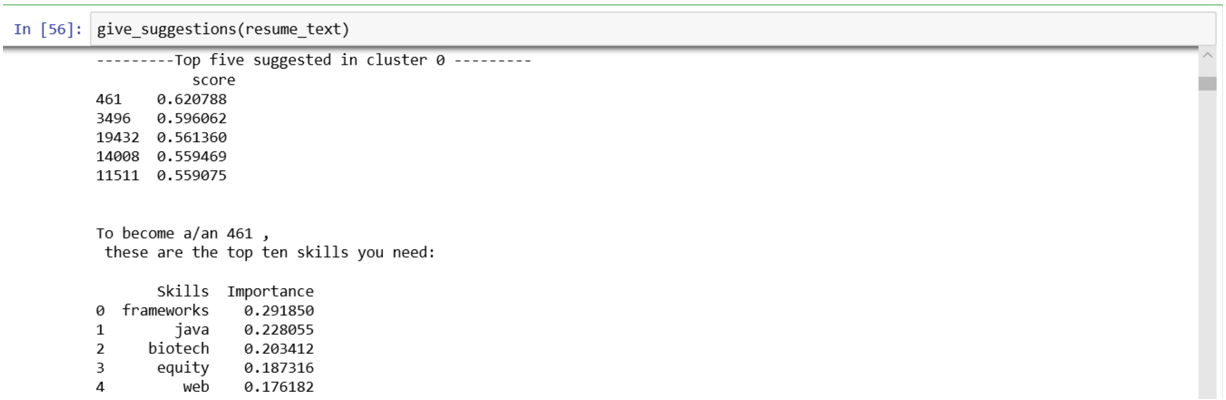




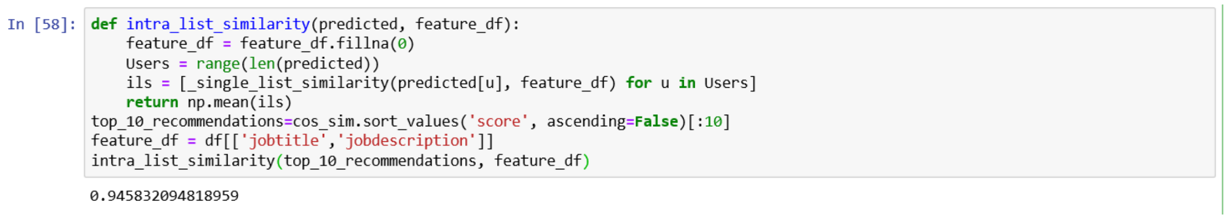
```
In [*]: resume_text=input("Enter your skills. These are programming languages, frameworks,platforms or databases that you have experi

        Enter your skills. These are programming languages, frameworks,platforms or databases that you have experience with
         python javascript
```



```
In [56]: give_suggestions(resume_text)

         CLUSTER NUMBER 0


         Top ten suggested for your cluster
                                                             score
         jobtitle
         Senior Developer - HEAVILYFunded, Revolutionary...  0.620788
         SW test Python c/c++                                0.596062
         Senior Python Developer                             0.561360
         Python Developer                                    0.559469
         Python Developer                                    0.559075
         Python Developer for FinTech start up               0.548576
         Python Developer                                    0.519687
         c/c++ Python Mathematics                            0.512534
         Javascript/Node.js Developer (Full stack)           0.493980
         Python Developer                                    0.491289
```



```
In [56]: give_suggestions(resume_text)

         ---------Top five suggested in cluster 0 ---------
                   score
         461    0.620788
         3496   0.596062
         19432  0.561360
         14008  0.559469
         11511  0.559075


         To become a/an 461 ,
          these are the top ten skills you need:

                Skills  Importance
         0  frameworks    0.291850
         1        java    0.228055
         2     biotech    0.203412
         3      equity    0.187316
         4         web    0.176182
```



```
In [58]: def intra_list_similarity(predicted, feature_df):
             feature_df = feature_df.fillna(0)
             Users = range(len(predicted))
             ils = [_single_list_similarity(predicted[u], feature_df) for u in Users]
             return np.mean(ils)
         top_10_recommendations=cos_sim.sort_values('score', ascending=False)[:10]
         feature_df = df[['jobtitle','jobdescription']]
         intra_list_similarity(top_10_recommendations, feature_df)

         0.945832094818959
```

REFERENCES

[1] “Context-Based Collaborative Filtering for Citation Recommendation”.IEEE (2015)

[2] “Scraping and Clustering Techniques for Linkedin Jobs”.Semanti Scholar (2016)

[3] “Classifying online Job Advertisements through machine learning”.IEEE (2018)

[4] “A Research of Job Recommendation System Based on Collaborative Filtering”.Science direct (2017)

[5] “ A Dive into Web Scraper World”. ieexplore:7724353 (2016)

[6] “Software refactoring at the package level using clustering techniques”. ieeexplore (2011)

[7] “ Generating top-N items recommendation set using Collaborative,Content based filtering and rating variance. ScienceDirect (2018)

[8] “ Exploratory data analysis. ScienceDirect:0377221 (1996)

[9] “ Web Scraping using Python (article) - DataCamp

[10] “ combine content based recommender system with K-means (Researchgate article) .