

**TRIBHUVAN UNIVERSITY**

**Institute of Science and Technology**

Final Year Project Report

On

**Doctor Recommendation using Naïve Bayes Classification and Analytic Hierarchy Process**

**Under the Supervision of**

Mr. Bikash Balami

**Submitted To:**

Department of Computer Science and Information Technology

Asian School of Management and Technology

Gongabu, Kathmandu

***In the partial fulfillment of the requirement for the Bachelor’s Degree in Computer Science and Information Technology***

**Submitted By:**

Sagar Khanal [8512/072]

Manoj Phuyal [7944/072]

Bishal Lamsal [7932/072]

Sandip Paudel [7956/072]

**ACKNOWLEDGEMENT**

It is our immense pleasure to have the opportunity to extend our heartfelt gratitude to all those who helped us throughout this course of this project. We wish to express our gratitude to our project supervisor Mr. Bikash Balami. Lecturer at Central Department of Computer Science and Information Technology, for his invaluable guidance, encouragement and help for completing this

work despite his busy schedule.

We are additionally thankful to Mr. Surya Bam, for the underlying guidance in our project and spurred us to put more exertion in this project by providing a flexible working environment. We are likewise thankful to every one of our respected educators for their motivation and encouragement that made ready towards the completion of this project.

Likewise, sincere thanks to our respected principal Er. Anil Lal Amatya and all the other faculty members for their support and encouragement. Also, we would like to thank our friends and other people who helped us directly or indirectly during this project.

Sagar Khanal [8512/072]

Manoj Phuyal [7944/072]

Bishal Lamsal [7932/072]

Sandip Paudel [7956/072]

**ABSTRACT**

Today the world Wide Web provides users with a vast array of information and such activity on the Web has increased to the point where hundreds of new companies are adding web pages daily. This has led to the problem of information overload. Recommender systems have been developed to overcome this problem by providing recommendations that help individual users identify content of interest by using the opinions of a community of users and/or the user’s preferences. The aim of this project is to help patients to get the best doctors. To achieve this goal, First, we analyze the weight of all the doctors based on provided feedbacks and then quality model of all the doctors is made. Feedbacks are processed through clustering and sentiment analysis using Bayes Classification. Clustering is used to group the best doctors out of the given list of doctors. Sentiment mining will be performed on the comments given by the patients’ in the feedback form. Then the final part is all about providing exact results based on the parameters provided. The algorithm used in this, most of all, it ensures the accuracy and diversity of the recommended result.

**TABLE OF CONTENT**

**ACKNOWLEDGEMENT................................................................................... i**

**ABSTRACT.......................................................................................................... ii**

**LIST OF FIGURES..............................................................................................iii**

**LIST OF TABLES.................................................................................................iv**

**ABBREVIATIONS................................................................................................ v**

**LIST OF FIGURE**

Figure 2.1 Use case Diagram of Sentiment Analysis………………….

Figure 2.2 Class Diagram of system…………………………………..

Figure 3.1 Architecture Of Sentiment Analysis………………………

Figure 3.2 Three Class sentiment analysis…………………………..

Figure 3.3 Flow Chart of the classifier System……………………..

Figure 3.4 User Interface of system…………………………………

Figure 3.5 Input Output Design………………………………………

Figure 3.6 General process architecture………………………………

Figure 4.1 Implementation model for Sentiment Analysis…………...

**LIST OF TABLE**

Table 4.1 Unit Testing……………………………........................................

Table 4.2. Accuracy Measurement…………………………………..……...

**ABBREVIATIONS**

**TF** (Term Frequency)

**IDF**(Inverse Document Frequency)

**SA**(Sentiment Analysis)

**Chapter One**

**INTRODUCTION**

**1.1. Introduction**

It’s been known that there are various studies and applications in many sectors, and health-care service is one of these areas that will continue to be vital and show no sign of maturing. With the popularization and rapid development of Internet, the internet services have pervaded every aspect of people’s lives [1]. Most of the people have access to smartphone with the internet connection but the less people got the knowledge on possibilities with the smartphone. Also, the massive amount of digital information is present on the internet. This information is unevenly distributed. So, there is need for intelligent recommender system more than ever before.

Talking about medicals and hospitals, there is limited resources for the reservation, and most of the patients prefer to make an appointment with the few famous doctors which causes a phenomenon that most patients fail to reserve while there are many other doctors available, and then led to low appointment rate [1]. In such situations, people make decisions based on other people recommendation, internet and advertisements. This is causing unnecessary waste of money and time. To solve these problems, it is urgent to present a valid doctor recommendation method.

This recommendation system will overcome the problems faced by people in finding the most relevantdoctors. The best doctors are the one who understand the patients’ problems, care for them, respect them regardless of who they are and treat the disease properly [2]. Thus, we propose a recommendation system that consists of two parts.

First, we analyze the weight of all the doctors based on provided feedbacks and then quality model of all the doctors is made. Feedbacks are processed through sentiment analysis using Naïve Bayes Classification. Sentiment mining will be performed on the comments given by the patients’ in the feedback form. Then the final part is all about providing exact results based on the parameters provided. The algorithm used in this overall ensures the accuracy and diversity of the recommended result.

**1.2. Problem Statements**

The appointment of potential patients with a doctor are performed manually and it is a very time-consuming procedure. The opportunity to communicate with on a one-to-one basis is highly valued. However, with many hundreds of applications each year, one-to-one conversations are not feasible in most cases. The communication requires a member of academic staff to expend several hours to recommend the best doctor for the patient. It would be useful to reduce his/her costs and time if recommendation system is made. One can simply do the task by his/her own.

The problem would be partially solved if the patient could find a best doctor as per recommendation, able to easily consult with them and do the treatment of his disease.

**1.3. Objectives**

The main objective for this project on doctor recommendation using Naïve Bayes Classification and Analytic Hierarchy Process are as follows:

* AHP is used to build the ranking model of the doctor based on their degree and working experience years.
* Naïve Bayes Classification algorithm is used to check the sentiment of the feedbacks provided to the system based on classification as Positive or Negative.
* To provide a platform to the users where they can find the relevant doctor based on doctor performance.

**1.4. Scope and Limitation**

One key reason why we need a recommender system in modern society is that people have too many options too use from due to prevalence of Internet. The project will design and evaluate a Doctor Recommendation System that is based on the Naïve Bayes Classification and Analytic Hierarchy Process. The application is targeted towards a people in need of relevant doctor for the specific treatment of disease and as of present it can only be used as a mobile application.

Looking the current scenario, most of the patients wants to consult with best doctor as possible which means no one would want to consult with some random doctors. The application is supposed to be a good companion to all the patients who are seeking the best doctor to consult. It contains all the information about doctors and is represented based on rank.

There are certain limitations in our system, they are:

* Accuracy of the system depends on the provided datasets and generated training model.
* Ranking of the doctor model is generated using AHP. Pair wise comparison for the criterions on AHP are done manually for now.

**1.5. Report Organization**

The report is organized as follows: Chapter 1 introduces about our system and its scope and limitations. Chapter 2 reviews requirement analysis and feasibility analysis. Chapter 3 specifies the system design of the project. Chapter 4 introduces an approach of implementation used, the tools used for the project and the necessary testing done to test the project. And finally, in Chapter 5 Conclusion and Recommendation are discussed.

**CHAPTER TWO**

**REQUIREMENT AND FEASIBILITY ANALYSIS**

**2.1. Literature Review**

For nearly 20 years, a major social problem in China is the difficulty and high cost of getting medical treatment, which is also a concern of people's livelihood. Although many new health care policies continue being proposed, the problem has not been solved fundamentally. A survey found that the most prominent difficulty of getting medical treatment for patients is to find suitable doctors. For safety and security, patients tend to pay high fees only to select big hospitals, regardless of their illnesses. But when all patients choose big hospitals, a series of problems emerge, including long queuing time, medical shortage, high fees and bad service attitudes of medical staff, etc. Personalized doctor recommendation can recommend the most suitable doctor according to each patient's interests and needs. On one hand, it will not delay the illness; on the other hand, it can solve the problems caused by too many people choosing limited number of big hospitals. For example, resource in big hospital is often in shortage while grass-roots medical facilities are idle. Compared with the other algorithms, recommendation system has a brighter future in development and application in e-commerce, which makes people feel more interested in it and start trying to find its rational uses in other areas.

Traditional methods of recommendation system include recommendation based on association rules [3], demographic [4], content [5], collaborative filtering [6] and on network [7], etc. The method based on association rules tries to mine the internal linkages between the items selected by the user find the items that are selected together and then use them in next recommendation. The method based on demographic and content finds similar users and projects to complete the recommendation, respectively. The method based on collaborative filtering considers users’ preference and finds the correlation between users and projects, besides the items we mentioned above. The method based on network completes the recommendation by building network models. However, only one single recommendation method has limitations inevitably, which is not suitable for complex medical diagnosis.

Baofu Yu [8] proposed the doctor recommendation algorithm based on BP neural network and SOM neural network. The former one uses the idea of the method based on content and classifies the information by subject; putting the information pages on the same subject in the same group. The latter one uses the idea of recommendation algorithm based on collaborative filtering, which matches the users interests model with different groups of pages, and then recommends the patients with those pages which have greater values than threshold.

Peng Li and Zhongxin Yu [9] used network models for doctor recommendation, which explored new ideas for the research. They proposed building a heterogeneous doctor information network model to describe different parts in a recommend process, such as the relationship among patients, doctors, diseases and treatment. Then classify the data and rank them so the same category of doctors and treatment method will be effective on patients. When a new patient comes, the recommendation process can be achieved by calculating the similarity between the new patient and the one in the network models.

Here, we try to use the most widely used analytic hierarchy process to design a patient-oriented doctor’s recommendation model. Inspired by the doctor rating system in the paper of Chonglin Sun [10], we try to design a novel rating algorithm. First, cluster doctors according to the illness that they are professional on, then we can ger the basic score using analytic hierarchy process. The recommendation in original doctor rating system may be inaccurate and unreasonable since the same doctors are recommended all the time, and the proposed algorithm can best overcome these shortcomings. Besides, we also improve the efficiency of the algorithm.

**2.2. Requirement Analysis**

**2.2.1. Functional Requirement**

System must manage to implement the query in appropriate manner. It should be able to classify the provided feedback passed to the system as query. System should be able to perform the dynamic operation to the database as the feedback query is passed to the system. System should generate the rank model of the criterions in the implementation of AHP.

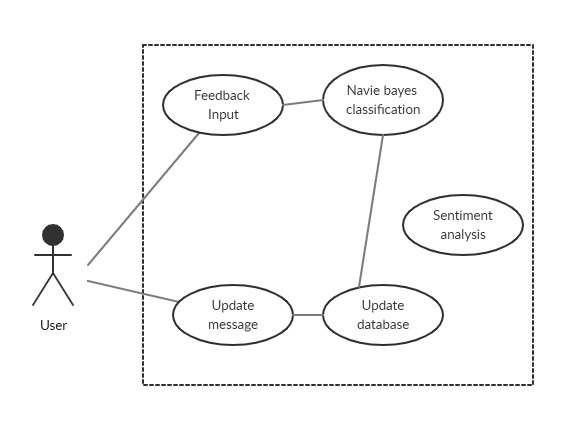


Figure 2.2.1.1 Use case diagram for Naïve Bayes Classification

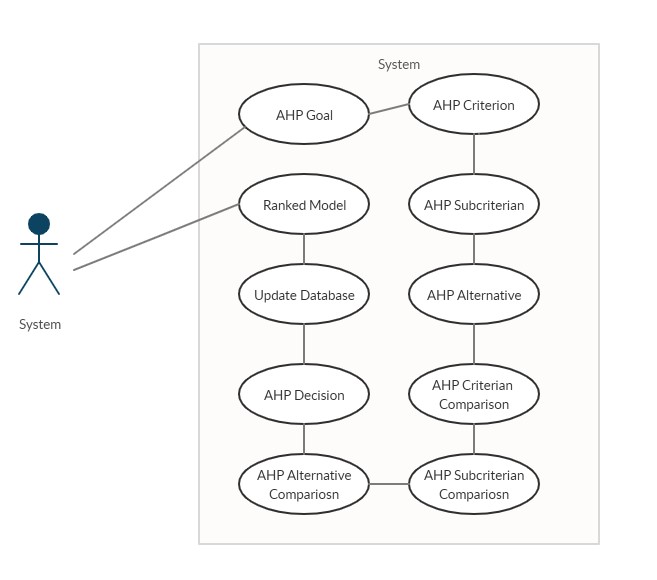
****

Figure 2.2.2.2 Use case diagram for Analytic Hierarchy Process

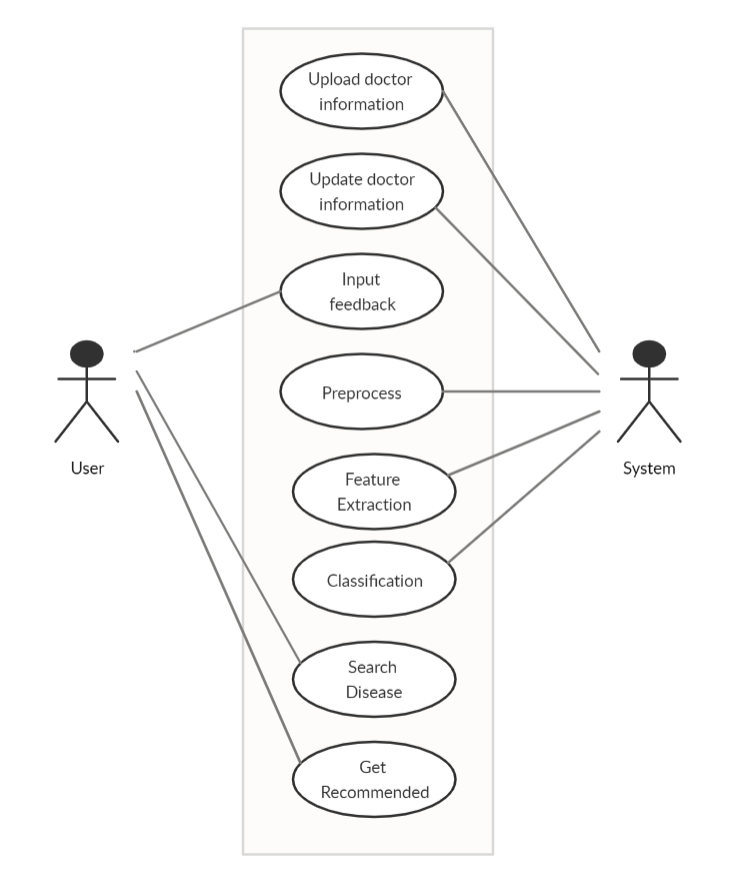


Figure 2.2.2.3. Use case diagram for system-user interaction

**2.2.2. Non-Functional Requirements**

The system should maintain the user transparency level while querying the data. The system should perform the needed operation in time. System should be available all the time besides the maintenance schedule. The system should provide reliable data as output. It should also provide the security measures to maintain the privacy of user data.

**2.3. Feasibility Analysis**

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats as presented by the environment, the resources required to carry for optimum analysis. The feasibility study is necessary to determine if creating a new or improved system is friendly with the cost, benefits, operation, technology and time. Following feasibility study is given as below:

**2.3.1. Technical Feasibility**

Technical analysis is concerned with determining how feasible a system is from a technical perspective. The project is developed for recommended system. System is developed using Python programming language used for sentiment analysis and hierarchical, .net programming language used for backend and react native used for the frontend design. Current tools are more than enough for system development. In future, if we want to change platform for this it is possible. Hence, the system satisfies the needs of technical feasibility. The following points were considered for the project’s technical feasibility.

•         The system will analyze the given views and opinions.

•         After analyzing the public opinions, it will categorize them under respective polarity,

•         The system can classify comments into negative, positive or neutral class.

•         If any error is occurred system will handle it with case.

**2.3.2. Economic Feasibility**

The purpose of economic feasibility is to determine the positive economic benefits. Since, this project work just providing a platform where user can get the best recommended doctor on the basis of specialty and user feedback. It requires just the hardware, software, and storage support as required by other existing application. It requires internet to run this site. As this system does not require any specific software hence, it does not require enormous budget while developing nor it expects any return from the system, so it satisfies the economic feasibility.

**2.3.3. Operational Feasibility**

System solves the problem well or it serves the user well. User can use this system simply as they use other application like Hello doctor, Health Tracker. There is not any restriction from government regulation for this system. This system will be socially acceptable because anyone with simple knowledge can use this system.

**2.3.4. Schedule Feasibility**

We have had the following schedule of activities during our project preparation.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Activities | 1 W | 1 W | 1 W | 1 W | 1 W | 1 W | 1 W | 1 W | 1 W | 1 W |
| Study & Analysis |  |  | 4W |  |  |  |  |  |  |  |
| Data Collection |  |  |  | 3W |  |  |  |  |  |  |
| Implementation |  |  |  |  |  |  | 3W |  |  |  |
| Testing |  |  |  |  |  |  |  | 3W |  |  |
| Documentation |  |  |  |  |  |  | 4W |  |  |  |
| Review |  |  |  |  |  |  |  |  |  | 1W |
| Presentation |  |  |  |  |  |  |  |  |  | \* |

**2.4.1. Data Modeling**

For the data modeling purposes, we chose ER-Diagram. Our system consists of seven entities that are; doctor, specialty, hospital, comment, disease, ahp and doctor degree. Ever entities contain three or more attributes. Data modeling is represented in ER Diagram as follows.

**Chapter Three**

**SYSTEM DESIGN**

**3.1. Schema Design**

The database consists of two main tables that are: Doctor table, Hospital table and Doctor Comment table. Doctor includes all the doctor data. The specialties and hospital taken from Specialty table and Hospital table. However, each of the table uses foreign key Doctor\_ID from Doctor.

Another table “Doctor Comment’’ consists all the comment provided for doctor after reviewing by any user identifying with their user email. The system consists of two strong entities; Doctor table and Doctor Comment table and two weak entities; Specialties table and Hospital table. The comment that the user has posted by the multiple attribute which is represented by the foreign key Doctor\_ID.

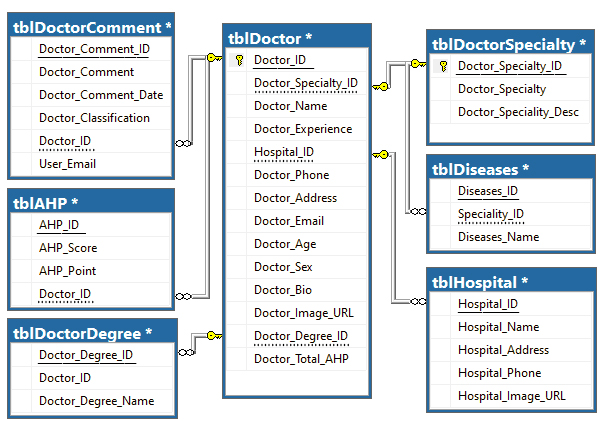


Figure 3.1: Schema Diagram

**3.2. Interface Design**

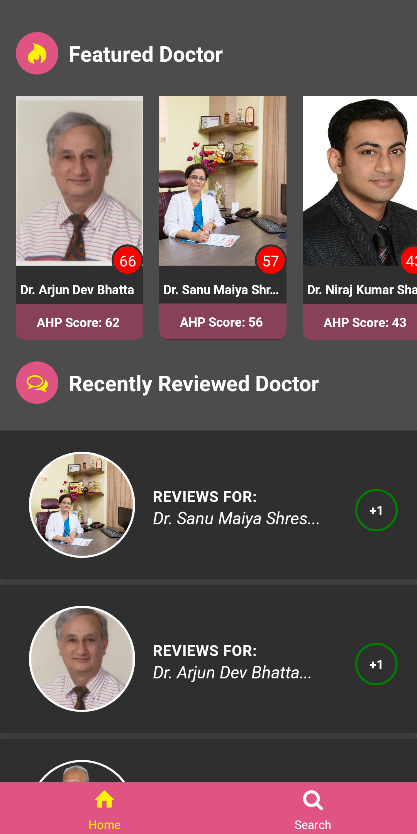
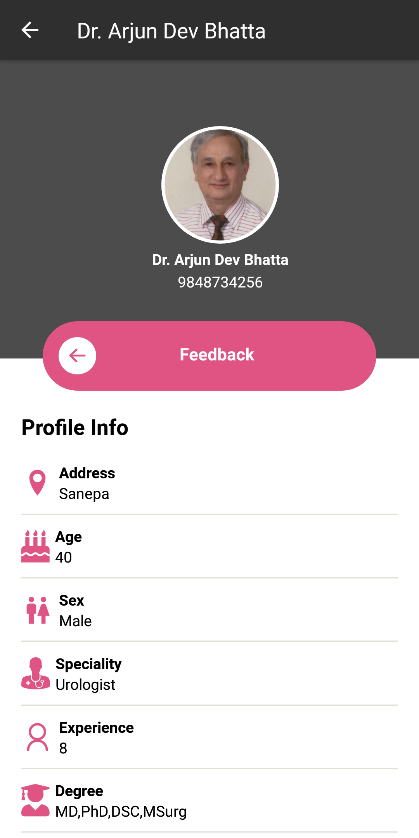
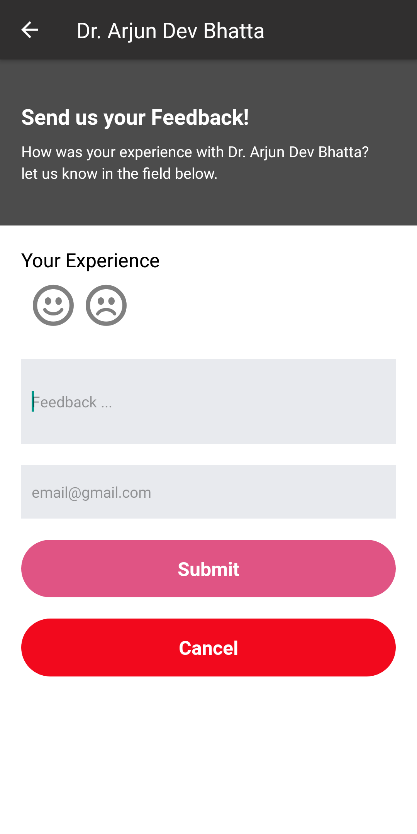
  

Figure 1 Figure 2 Figure 3

Figure 1 describes the home page with two section. First one is Featured Doctor and second one is Recently Reviewed Doctor. In Featured Doctor, list of doctors along with their AHP score is shown. Doctor having highest AHP score comes at the beginning and the list of doctors is shown based on AHP score arranged in descending order. In Recently Reviewed Doctor, comment given to doctor recently is shown. Positive comment is indicated as +1 and negative comment is indicated as -1.

Figure 2 contains the part where after clicking on the doctor image, it opens the page which contain doctor’s profile along with the feedback button.

Figure 3 contains the part where after clicking the feedback button, it opens the page where patient can review for their doctor and submit it. It contains the input text for feedback and email address.

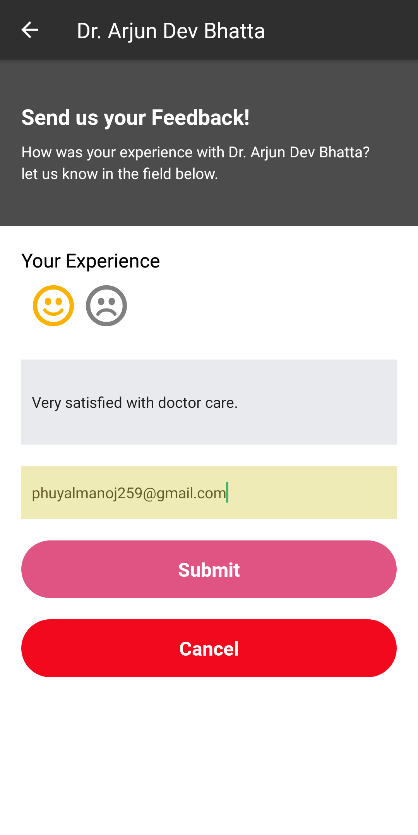
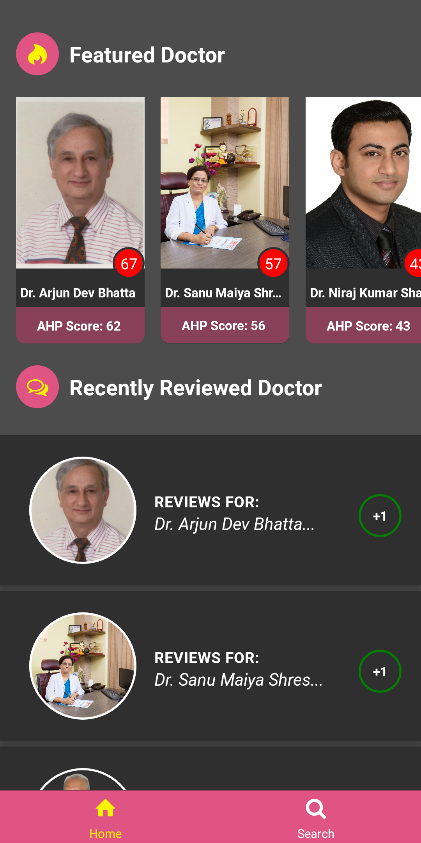
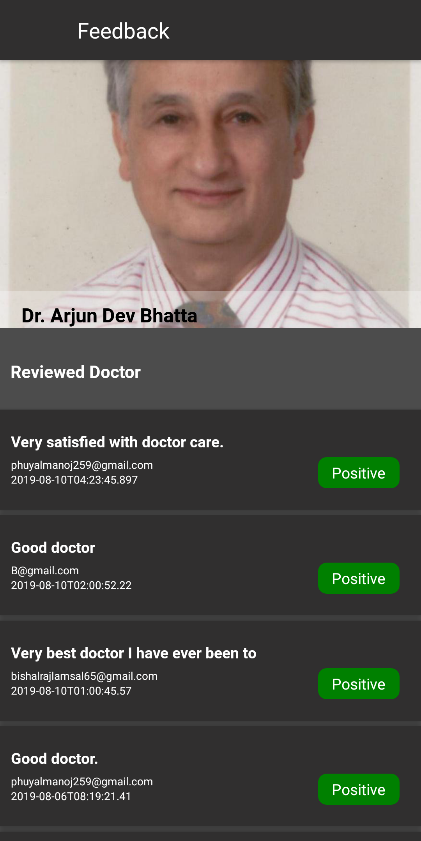
  

Figure 4 Figure 5 Figure 6

Figure 4 contains the part where when the patient gives positive feedback, first smiley is marked indicating the positive symbol. Comment and email are provided and submitted.

Figure 5 contains the part where after submitting the positive feedback, the AHP point of doctor increases from 66 to 67 and is shown in the page.

Figure 6 contains the part where It displays the page which contains the list of comment given to the doctor after clicking on the review tab in recently reviewed doctor section.

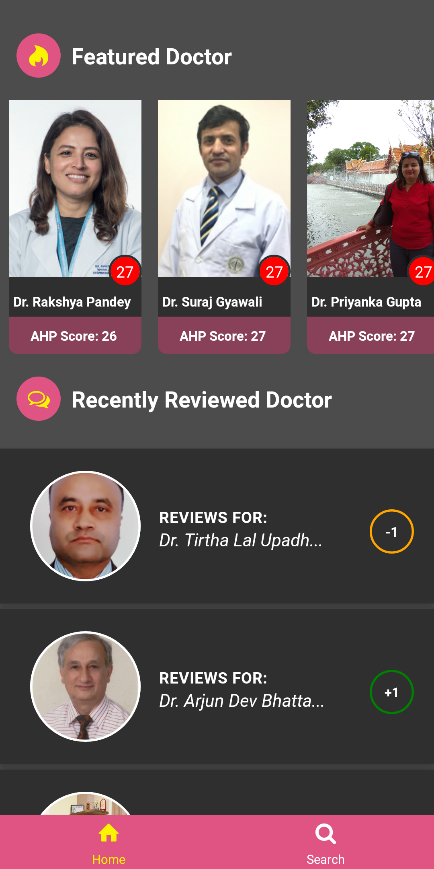
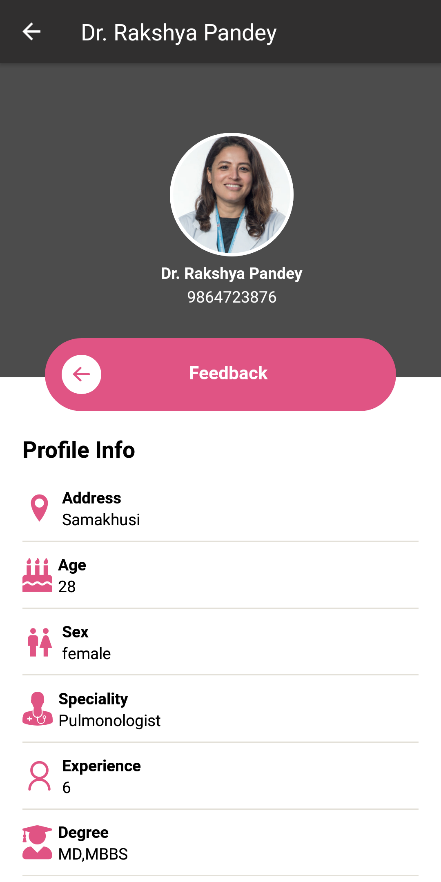
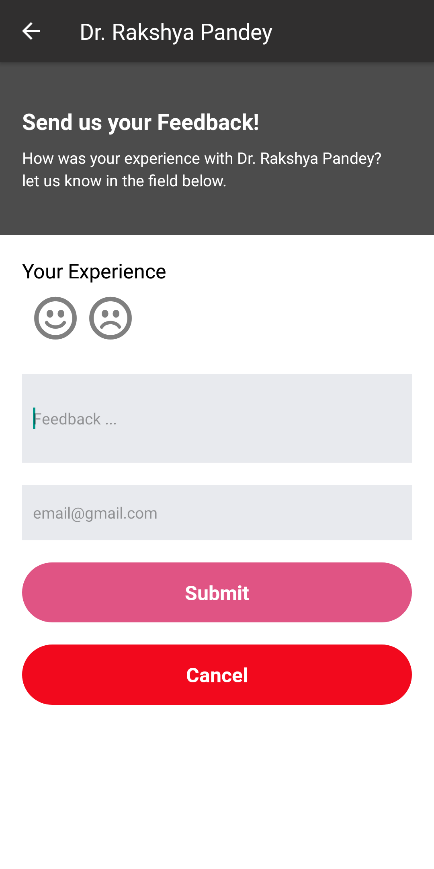
  

Figure 7 Figure 8 Figure 9

Figure 7 contains the part where it displays the home page with list of doctors having same AHP point.

Figure 8 contains the part where after clicking on the doctor image, it opens the page which contain doctor’s profile along with the feedback button.

Figure 9 contains the part after clicking the feedback button, it opens the page where patient can review for their doctor and submit it. It contains the input text for feedback and email address.

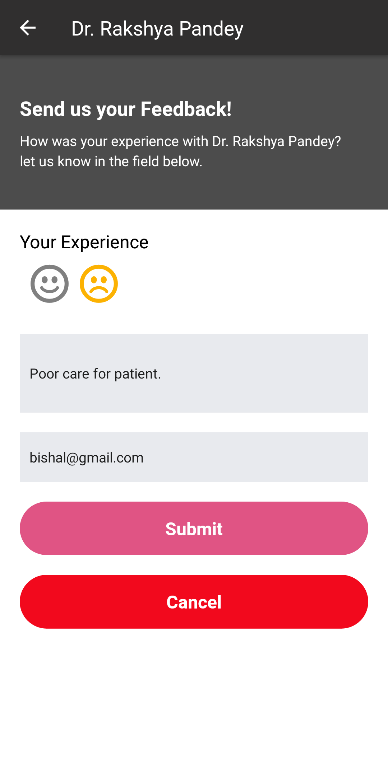
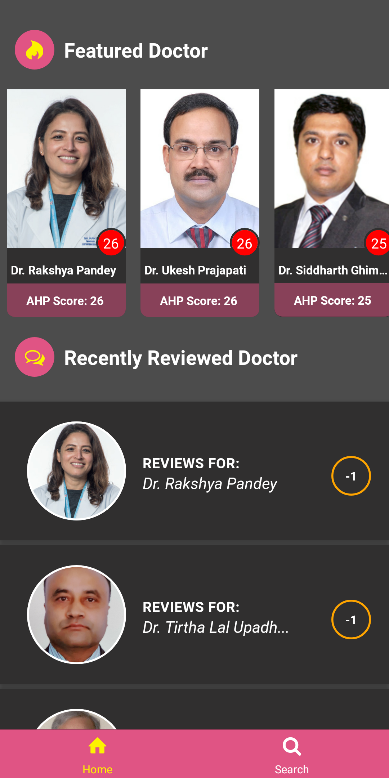
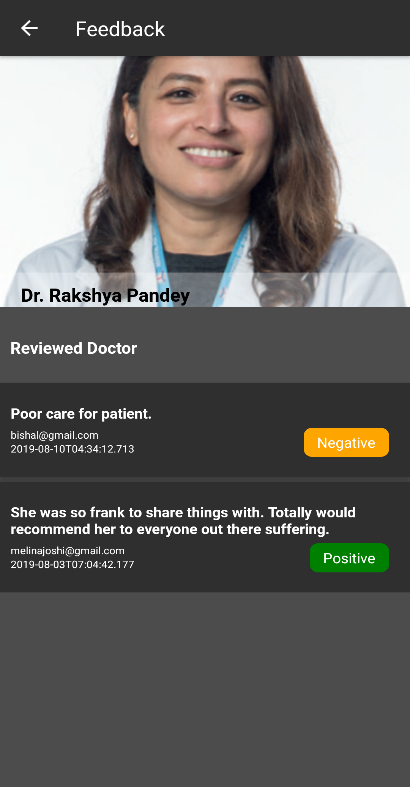
  

Figure 10 Figure 11 Figure 12

Figure 10 contains the part where when the patient provides negative feedback, second smiley is marked indicating the negative symbol. Comment and email are provided and submitted.

Figure 11 contains the part where after submitting the negative feedback, the AHP point of doctor decreases from 27 to 26 and is shown in the home page.

Figure 12 contains the part where it displays the page which contains the list of comment given to the doctor after clicking on the review tab in recently reviewed doctor section.

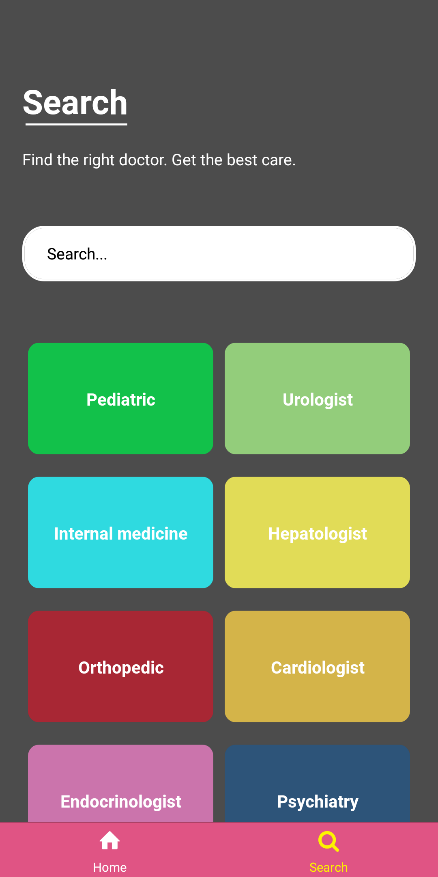
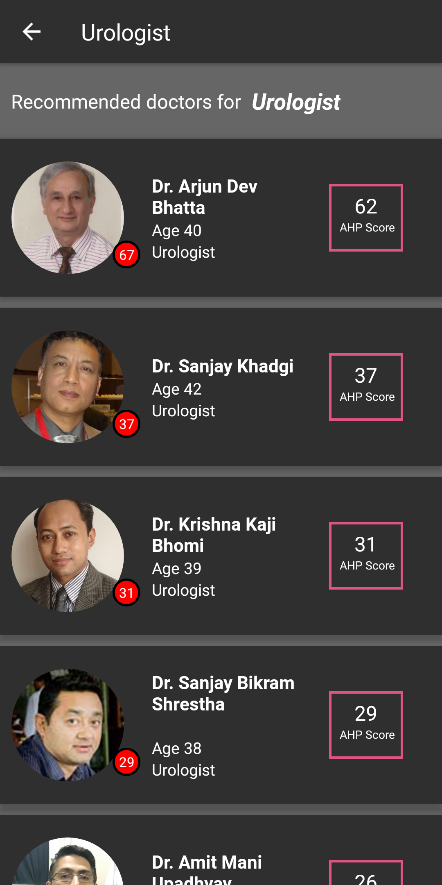
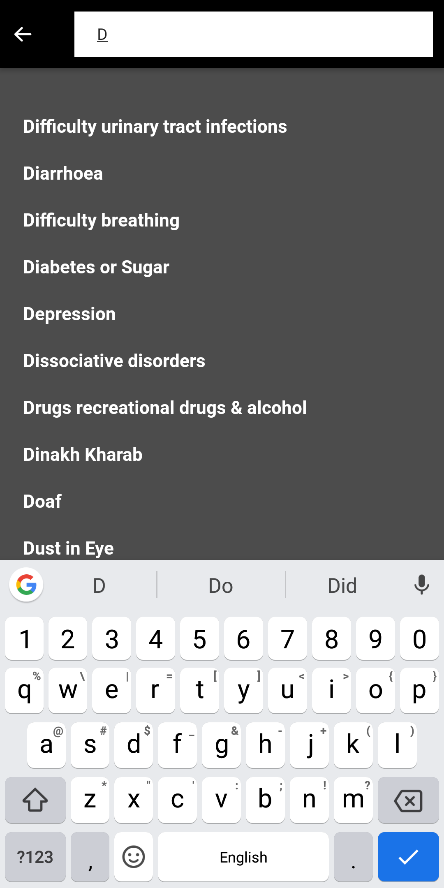
  

Figure 13 Figure 14 Figure 15

Figure 13 contains the part where after clicking the search tab, page is opened which contain the input text named search and the list of buttons based on doctor specialty.

Figure 14 contains the part where after clicking on the button, list of doctors based on specialty is displayed along with their AHP score. List of doctors is shown according to the AHP score arranged in descending order.

Figure 15 contains the part where It displays the search option given in figure 13. List of disease is shown based on given keyword. It displays the disease matching the keyword.

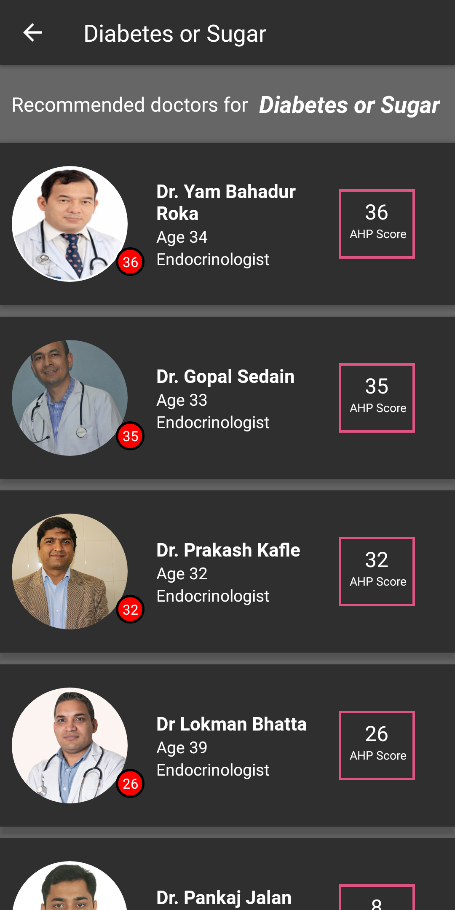


Figure 16

Figure 16 contains the list of recommended doctors which is obtained after clicking on the disease that is displayed matching the keyword.

**3.3. Process Design**

The user input can either be a form input or a user’s search query. When user passes the query, it will be parsed and separated as a token and stop words are removed and retrieval process is done.

Main Window

Plot Summary

Parse input word

Raw Input

Form Data

Search Query

Processing

(Tokenization, Removal of stop words, lowercase)

Processing

(Tokenization, Removal of stop words, lowercase)

Document TF-IDF score

Query TF-IDF score

Cosine similarly between Document and query

Sort and select Top K relevant document with large cosine similarly values

Classification

# Chapter 4

# IMPLEMENTATION AND TESTING

**4.1. Implementation**

This system follows a certain process for document retrieval. This process can be divided into five main phases; Document preprocessing, Document weighing based on query term, and Ranking and Recommendation. Ranking and Recommendation can also be divided into three main sub-phases; Preprocessing, Weighing and Scoring. But, before any of that can be done, we require enough dataset to work on.

**4.1.1. Data Collection**

Along with the study of the system, data collection was carried out following the analysis of working structure. We collected more than 300+ feedback data from different hospital websites and forums and added some manually. We grouped those feedback based on sentiment i.e. positive or negative.

In order to be able to differentiate the effect of algorithm in the retrieved set, we need to know the term frequency for a certain term to be used as an example. This was done by predefining the number of the term frequency in each classified model.

For example, let’s consider the term ‘good’, ‘reliable’, ‘experienced’. Here these are mostly mentioned in positive model i.e. Term frequency of positive model with those terms are high. In such a way, negative model is also classified based on the terms they contain. These are first manually classified for the accurate analysis of the result that the system provides.

**4.1.2. Document Preprocessing**

After user provides the feedback to the system, the system needs to search through all the documents that are indexed with the category. Before taking the documents through all the calculations involved in text classification algorithm, there are two major operations that need to be performed in each document which are:

* Tokenization
* Stop words removal

**Tokenization**

Tokenization occurs on the word level. So here, the stream of text from the documents are parsed and broken up into words, phrases, symbols, or other meaningful elements called token. However, it is difficult to define what is meant by a “word”. This system uses white spaces to differentiate words to a token. These tokens are kept as a comma separated tokens as list data types which are then ready to be searched.

**Stop words removal**

The tokenized form of documents consists of all the words that appear in the document. This document consists of mainly of common words like a, the, an, is, are, and, for, and so on. Inclusion of these words results in misinterpretation of the document relevancy.

For example,

User feedback: Dipesh is very experienced doctor.

Doc 1: Dipesh (1 time) is (14 times) very (2 times) experienced (10 times) doctor (2 times).

Doc 2: Dipesh (2 time) is (32 times) very (13 times) experienced (2 times) doctor (3 times).

Doc 3: Dipesh (0 time) is (10 times) very (7 times) experienced (0 times) doctor (4 times).

Here, the TF calculation in system will identify Doc 2 as most relevant document as the term frequency of ‘is’, ‘very’, is very high as compared to others. This results in ignorance of important terms like ‘experienced’, ’doctor’. Thus, stop words must be removed in order to retrieve the meaningful and logically relevant document.

**4.1.3. Query Processing**

When the user enters the feedback, the keywords provided by them are considered as a query. Then TF-IDF is calculated for the query. Now, according to the query term, TF-IDF in train data are used to calculate cosine similarity for retrieving relevant classification.

**4.1.4. Document Weighting**

We use TF (Term Frequency) and IDF (Inverse Document Frequency) for document weighting.

**Term Frequency (TF)** is a measure of how often a term is found in a collection of documents. It is a reasonable scoring mechanism that computes a score for each query term that matches with the document terms. It counts the frequency of the terms that matches between the query terms and the document terms list and is denoted by TF(t,d) [7]

Term frequency suffers from a critical problem that all terms are considered equally important. In fact, certain terms have little or no selective power in determining relevance. For example: a collection of documents of the “Doctor” is likely to have the term “Doctor” in almost every document. Terms which appear very few in numbers may have higher probability of being relevant. So, we must scale down the term weight of term with high collection frequency. Thus, **Inverse Document Frequency (IDF)** is used to find the rarity of a term and helps to show how important a word is. IDF is calculated as

IDF(t) = loge(D/Dt)

Where, D = Total number of documents,

Dt = Number of documents with query term “t”

Now, weighing of TF and IDF is combined to obtain those sets of documents that contains higher frequency of important words.

**4.1.5 Classification**

**Multinomial Naïve Bayes**

The multinomial Naïve Bayes classifier is suitable for classification with discrete features. For example: word counts for text classification. The multinomial distribution normally requires integer feature counts. However, in this, as well as in practice, fraction counts such as TFIDF works. After calculation of TFIDF, Multinomial NB is used for the classification.

Patient Feedback

Model Training

Evaluation

Preprocessing

Data Partition

Feature Extraction

Test Set

Train Set

Model

Feature Matrix of Test Set

Feature Matrix of Train Set

Fig 4.2 Implementation of Patient Feedback Sentiment Analysis

**4.1.6 Analytic Hierarchical Process**

Analytic Hierarchy Process is a technique for decision making in complex environments in which many variables or criteria are considered in the prioritization and selection of alternatives or projects.

**Data Collection**

To create the ranking model of doctor, data collection on doctors is also carried out along with their specialties and information from different hospital websites.

**Decision Hierarchy**

Decision hierarchy is structured considering the goal of study and determine the criteria and sub-criteria.

Doctor Ranking

Degree

Experience

1 year

PhD

2 year

MBBS

3 year

M.D

5 year +

MPhil

MSurg

**Pair Comparison Matrix**

|  |  |
| --- | --- |
| **Relative Importance** | **Definition** |
| 1 | Equal Importance |
| 3 | Weak Importance |
| 5 | Strong Importance |
| 7 | Demonstrated Importance over the other |
| 9 | Absolute Importance |

Table 1: Scale of pair-wise comparison for AHP

A set of all judgements in the comparison matrix in which the set of all elements is compared to itself by using the fundamental scale of pair-wise comparison shown in Table 1.

|  |  |  |
| --- | --- | --- |
|  | Degree | Experience |
| Degree | 1 | 1/5 |
| Experience | 5 | 1 |

Table 2: Pairwise Comparison Matrix

**Consistency Check**

Then the consistency of judgements across the Consistency Index (CI) and the Consistency Raito (CR).

where λmax is the Eigen value corresponding to the matrix of pair-wise comparisons and n is the number of elements being compared.

Consistency Ratio (CR) is given by:

where, (RCI) is a random consistency index defined i.e. 0 for 2 criteria

A value of CR less than 0.1 is generally acceptable; otherwise the pair-wise comparisons should be revised to reduce incoherence

**Priorities**

Priorities are obtained from the following steps

* Sum of each column of the reciprocal matrix are calculated

|  |  |  |
| --- | --- | --- |
|  | Degree | Experience |
| Degree | 1 | 1/5 |
| Experience | 5 | 1 |
| Sum | 6 | 6/5 |

* Each element of the matrix is divided with the sum of its column, relative weight is normalized. Sum of each column is 1.

|  |  |  |
| --- | --- | --- |
|  | Degree | Experience |
| Degree | 1/6 | 1/6 |
| Experience | 5/6 | 5/6 |
| Sum | 1 | 1 |

* Finally normalized principal Eigen vector is obtained by averaging across the rows

|  |  |
| --- | --- |
|  | Weight |
| Degree | 1/2(1/6+1/6) | 0.166 |
| Experience | 1/2(5/6+5/6) | 0.833 |

Ranks are set according to the highest percentage

|  |  |  |
| --- | --- | --- |
| Criteria | Priority | Rank |
| Degree | 16.7 % | 2 |
| Experience | 83.3 % | 1 |

**Scoring**

This is the manual process of assigning points to each of the alternatives i.e. to the doctors according to their rank.

|  |  |  |
| --- | --- | --- |
| Criteria | Rank | AHP Score |
| Degree | 2 | 20 |
| Experience | 1 | 80 |

**Integration of AHP and Naïve Bayes Classification for the Recommendation System**

AHP is implemented by which the rank model for all the doctors are created. Scoring is done to provide ‘AHP Score’ to each doctor according to their rank. Every time the doctor gets feedback from the patients, their scores are changed dynamically depending upon the sentiment of feedback i.e. +1 for positive feedback and -1 for negative feedback. Finally, on the basis on rank of doctors for each specialty, doctors are recommended to the patients.

**4.2. Tools Used**

**4.2.1. Python 3.7**

We have used Python programming language for the development of recommendation system. The main reason behind using Python for coding is because it supports List data type. We use list for parsing through the token in database model. Python also supports various inbuilt system functions which minimizes the line of code to be written making it similar to use.

**4.2.1.1. Sklearn**

Sklearn is a simple and efficient tool for data mining and data analysis. Sklearn consists of the class that can be used for text classification. Word Count Vectorizer, TFIDF Transformer, Multinomial NB, Metrics are used for the whole classification process. Word Count Vectorizer converts a collection of text documents to a matrix of token counts. TFIDF Transformer transforms a count matrix to a normalized TF or TFIDF representation. Finally, Multinomial NB is used for the classification based on the feature from the trained model.

**4.2.1.2. Flask**

Flask is used to create API of user’s data classifications i.e. Positive classification and Negative classification. That classification points i.e. 0(zero) for negative and 1(one) for positive are stored in database of respective doctor’s profile.

**4.2.1.3. Paralleldots API**

Paralleldots have artificial intelligence powered text and visual intelligence APIs. Those APIs are implemented in our system to improve the accuracy of classification. Behind the scenes, user feedbacks are analyzed through this API and are added specifically to the existing training model of the system, thus increasing the accuracy rate of our model.

**4.2.2. Visual Studio 2019**

Visual Studio .NET is a Microsoft-integrated development environment (IDE) that can be used for developing consoles, graphical user interfaces (GUIs), Windows Forms, Web services and Web applications. We have used .NET MVC and C# programming language for data processing between algorithms and database which help in parsing, retrieving and implementation of data model in the form of API.

**4.2.3. Microsoft SQL Server Management Studio**

We used MSQL for database management in which we created tables, generated stored procedure query and OOP database programming.

**4.2.4. Android Studio**

We used Android Studio for developing React Native App. React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. React Native also exposes JavaScript interfaces for platform APIs, so our React Native apps can access platform features like the phone camera, or the user's location. Here, we used React Native for main UI and front in of our system.

**4.2.5. Adobe Experience Design**

Adobe XD was used for designing UX/UI prototypes. It provides wide varieties of designing tools and can simulate the navigation of the application pages and actions of the system.

**4.2.6. Visual Code**

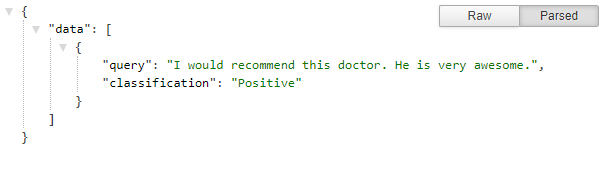
We used VS Code for scripting with HTML, CSS and JS. It comes with lot of shortcut that makes coding easy and time efficient.

**4.3 Testing**

**4.3.1. Unit Testing**

**4.3.1.1. Testing the Algorithm through a query**

For the classification of the feedbacks provided to the system, they are classified as either positive or negative according to the training model created. To check whether the feedback gets classified, we provided the positive term “I would recommend this doctor. He is very awesome.” to the system as a search query. For testing the algorithm, we passed the documents through the algorithm. For the test to be successful, the system should classify the feedback as positive. Any irrelevancy in the result would refer inaccuracy of the algorithm.

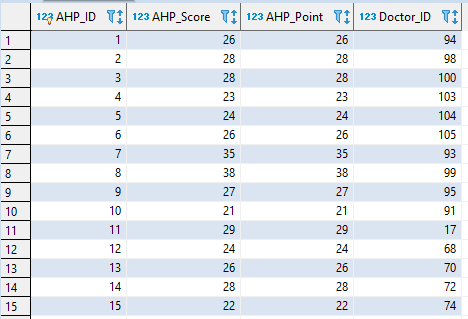


As we can see when we passed query to the API of the algorithm, it classifies the query as “Positive” which is accurate.

**4.3.1.2. Database Testing**

The following tests were conducted to maintain the consistency of the database design.

* Retrieving, updating, deleting and inserting data on the database.
* Insertion of ahp score and point details into their respective tables in the database.



**4.3.2 User Interface Testing**

Finally, the user interface was tested on different devices based on different resolution i.e. mobile phones and tablet devices.

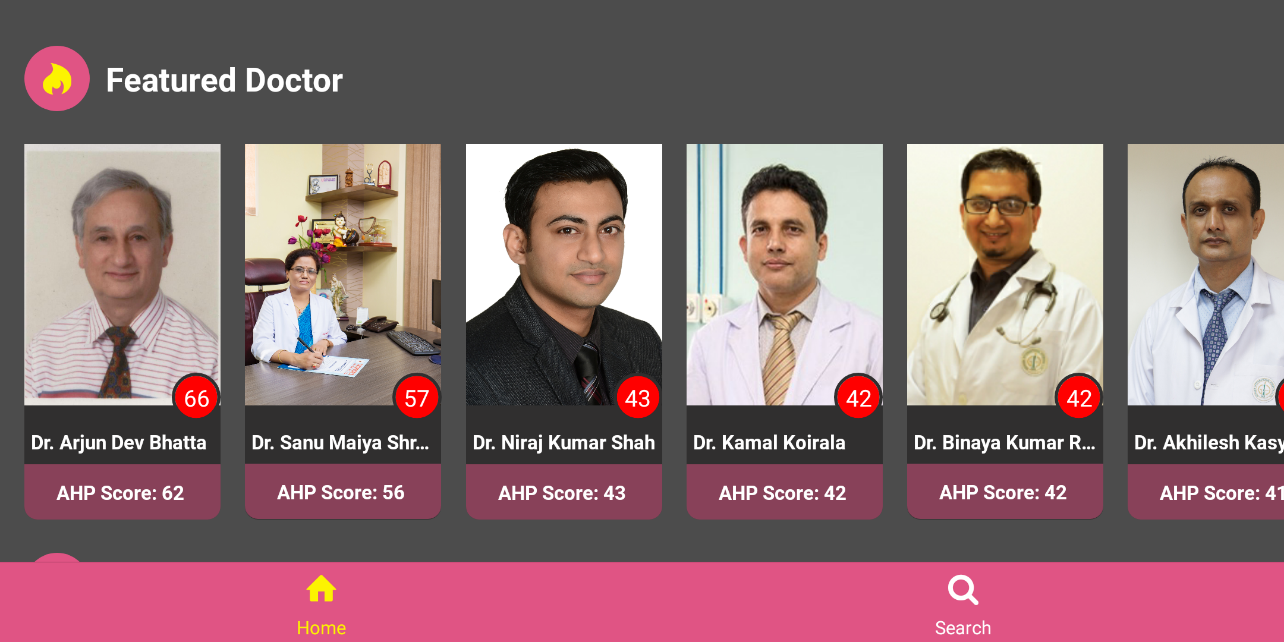


Figure 4.3.2.1 User Interface on Tablet Devices

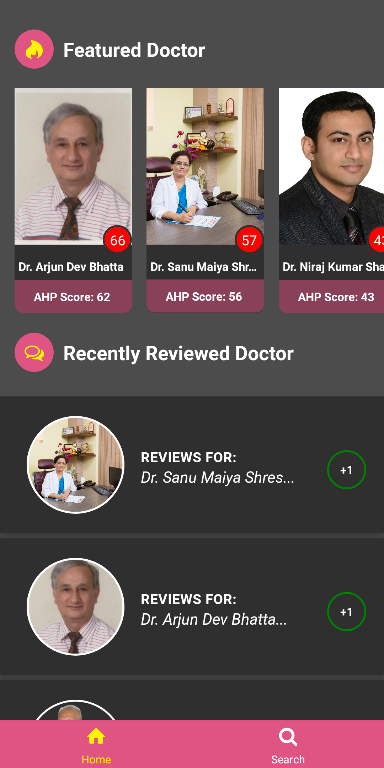


Figure 4.3.2.2. User Interface on Mobile Devices

**4.3. Integration Testing**

Integration testing is an approach where modules are developed and testing of modules always starts at the finest level of the programming hierarchy and continues towards the lower levels. It’s the extension of unit testing. Integration testing takes a smaller unit of unit testing and tests their behavior as the whole.

During the integration test, the system with feedback classification and database were integrated to specify whether the output obtained is relevant or not for the users.

**Chapter 5**

**CONCLUSION**

**5.1. Conclusion**

The first goal of the current study was the use of Naïve Bayes Classification algorithm and Analytic Hierarchy Process technique for the purpose of doctor recommendation. The project study involved great deal of work in the area of information retrieval and data mining. We mainly focused on methods and problems on the relevant doctor retrieval.

Naive Bayes algorithm and Analytic Hierarchy Process is a powerful algorithm and technique respectively. Our system has been based on this algorithm and technique. We presented the system for finding the best doctor based on user search query. The real-world application of this project study would help people to find the best doctor that they are searching for. And this algorithm is being used into the system to check the user feedback given to the doctor whether it is positive or negative. This helps to rank  the doctor on the basis of AHP score i.e. doctor having more positive feedback has greater AHP score than the doctor having least positive score. Doctor is listed based on Analytic Hierarchy Process.

**5.2. Future Works**

Rank model of doctors are built only with pair comparison of two criterions degree and experience while implementing AHP. In future, more criterions or the factors about doctors will also be added. For the implementation of Naïve Bayes Classification, training model can be made much better by adding more datasets to the system. In the future, profile system will also be added including their previous check up records and track records of their health. Not to forget about appointment, such feature will also be added later in future to maintain the connection between patient and doctor. Also, the system can also be extended to be more user-friendly improving user experience and of user’s convenience by adding the location mapping of the patient and showing relevant doctors which are nearest to them. Overall more precision development works for the system can be done in future.

**REFERENCES**

[1] YONG-FENG HUANG I, PENG LIU I, QIAO PAN I, JING-SHENG LIN. A Doctor Recommendation Algorithm based on doctor performances and patient preferences,2012:92-93.

[2] Qiwei Han, Mengxin Ji, Inigo Martinez de Rituerto de Troya, Manas Gaur, Leid Zejnilovic Universidade Nova de Lisboa, School of Business and Economics. A Hybrid Recommender System for Patient-Doctor Matchmaking in Primary Care,2018: 1-2

[3] Zhenguo Ping, Jing Chen. A personalized recommendation system based on association rules[J]. Computer integrated system, 2003,9(10):1–2.

[4] Chen L, Nayak R, Xu Y. A Recommendation Method for Online Dating Networks Based on Social Relations and Demographic Information[C]. In Proceedings of ASONAM, 2011: 407–411.

[5] Kompan M, Bieliková M. Content-Based News Recommendation [J/OL]. In Proceedings of EC-Web, 2010: 61–72.

[6] Ailin Deng, Yangyong Zhu, Bole Shi. Collaborative filtering recommendation algorithm based on item score prediction[J]. Journal of software, 2003, 14(9):1621–1628.

[7] Kehan Chen, Panpan Han, Jian Wu. Recommendation algorithm for heterogeneous social networks based on user clustering[J]. Journal of Computer Science, 2013: 36(2):2–5.

[8] Baofu Yu. Research and implementation of personalized recommendation system for medical information[D]. Zhejiang: Zhejiang University, 2012: 46–50.

[9] Peng Li, Zhongxin Yu, Ning Li. Research on intelligent medical recommendation system based on heterogeneous network analysis[N]. Chinese Journal of health information management, 2013-12-10(6): 1–6.

[10] Chonglin Sun. The research and implementation of doctor recommendation system based on multi label classification and collaborative filtering[D]. Liaoning: Liaoning University, 2015:35-36, 65–66.

**APPENDIX**

**Generate the rank model of the doctor and save it to the csv file**

import numpy as np

import pandas as pd

print("=============================")

print("Analytic Hierarchy Process")

print("=============================")

# Objective

objective = input("What's the main objective?: ")

#Criterias

no\_of\_criterias = int(input("No of Criterias?: "))

criterias = []

for i in range(no\_of\_criterias):

criterias\_input = input("Enter Criteria "+str(i+1)+": ")

criterias.append(criterias\_input)

rank\_collection\_one = []

rank\_collection\_two = []

factor\_one = []

factor\_two = []

#SubCriterias

print("=============================")

No\_Factors\_One = 0

No\_Factors\_Two = 0

No\_Factors = [No\_Factors\_One,No\_Factors\_Two]

for i in range(no\_of\_criterias):

times = int(input("How many "+str(criterias[i]).lower()+' values are there?: '))

No\_Factors[i] = int(times)

# For First Criteria

def rankModel(no\_of\_factors,s):

n = 0

factors = {}

com\_board = [[" " for i in range(no\_of\_factors + 1)] for z in range(no\_of\_factors + 1)]

norm\_factors = [[" " for i in range(no\_of\_factors)] for z in range(no\_of\_factors)]

while n < no\_of\_factors:

factor =input("Enter factor No. {} name: ".format(n + 1))

factors[n] = factor

global factor\_one

global factor\_two

if(s==0):

factor\_one.append(factor)

elif(s==1):

factor\_two.append(factor)

com\_board[0][n + 1] = factor

com\_board[n + 1][0] = factor

com\_board[n + 1][n + 1] = 1

n += 1

# print(com\_board)

# inserting the User Weights for each factors pair

def weights():

#Pair Comparison

print("Pair Comarison Importance Scale: Pick One Value")

print("============================================")

print("============================================")

print("1 - Equal Importance")

print("2 - Equal to Moderately Importance")

print("3 - Moderately Importance")

print("4 - Moderately to Strongly Importance")

print("5 - Strongly Importance")

print("6 - Strong to very Strongly Importance")

print("7 - Very Strongly Importance")

print("8 - Very Strongly to Extremely Importance")

print("9 - Extremely Importance")

print("============================================")

print("============================================")

x = 0

while x <= no\_of\_factors \*\* 2:

z = x + 1

while z < no\_of\_factors:

com\_board[x + 1][z + 1] = float(input("The Importance of Factor {} {} compared to Factor {} {} on a Scale of (1-9): ".format(x + 1,factors[x],z + 1,factors[z])))

com\_board[z + 1][x + 1] = 1 / (com\_board[x + 1][z + 1])

z += 1

x += 1

weights()

# Normalization & weight determination

x = 1

tot\_factors = [0 for i in range(no\_of\_factors)]

print("Total values for Each Column")

while x <= no\_of\_factors:

j = 1

while j <= no\_of\_factors:

tot\_factors[(x - 1)] = float(tot\_factors[(x - 1)]) + com\_board[j][x]

j += 1

print(tot\_factors[(x - 1)], end=" ")

print()

x += 1

print()

print("=================================")

print("Normalized values for Each Factor")

print("=================================")

x = 1

while x <= no\_of\_factors:

j = 1

while j <= no\_of\_factors:

norm\_factors[j - 1][x - 1] = float(com\_board[j][x]) / tot\_factors[x - 1]

j += 1

x += 1

x = 1

while x <= no\_of\_factors:

j = 1

while j <= no\_of\_factors:

print(norm\_factors[(x - 1)][j - 1], end=" ")

j += 1

print()

x += 1

# Normalization & weight determination

print()

print("=================================")

print("Priority vector or weight Values")

print("=================================")

x = 0

priority\_vector = [0 for i in range(no\_of\_factors)]

while x < no\_of\_factors:

priority\_vector[x] = sum(norm\_factors[x]) / no\_of\_factors

print(factors[x], ": ", priority\_vector[(x)])

global rank\_collection\_one

global rank\_collection\_two

if(s==0):

rank\_collection\_one.append(priority\_vector[(x)])

elif(s==1):

rank\_collection\_two.append(priority\_vector[(x)])

x += 1

for i in range(no\_of\_criterias):

print("==================================")

print("Ranking for Sub-Criterian :"+str(criterias[i]))

print("==================================")

rankModel(No\_Factors[i],i)

if(i==0):

dataframeOne = pd.DataFrame(data =rank\_collection\_one,columns=['Weight'] )

for j in range(len(factor\_one)):

dataframeOne.rename(index={j:factor\_one[j]},inplace=True)

dataframeOne['Rank'] = dataframeOne['Weight'].rank(ascending=0)

elif(i==1):

dataframeTwo = pd.DataFrame(data =rank\_collection\_two,columns=['Weight'] )

for j in range(len(factor\_two)):

dataframeTwo.rename(index={j:factor\_two[j]},inplace=True)

dataframeTwo['Rank'] = dataframeTwo['Weight'].rank(ascending=0)

# Scoring

score\_one = []

total\_one = len(rank\_collection\_one)

for i in range(total\_one):

score\_one.append(total\_one - dataframeOne['Rank'][i] + 1)

dataframeOne['AHP Score']= score\_one

dataframeOne.to\_csv(r'DegreePriority.csv')

score\_two = []

total\_two = len(rank\_collection\_two)

for i in range(total\_two):

score\_two.append(total\_two - dataframeTwo['Rank'][i] + 1)

dataframeTwo['AHP Score']= score\_two

dataframeTwo.to\_csv(r'ExperiencePriority.csv')

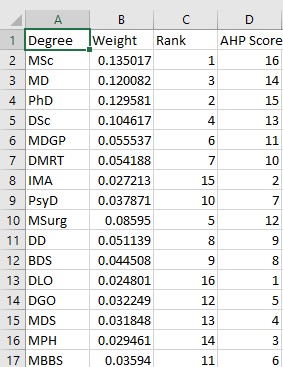


Table 1. AHP Score generated for Degree Criterion

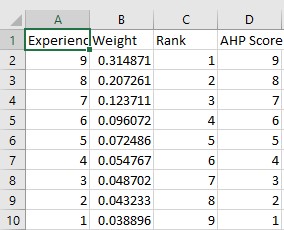


Table 2. AHP Score generated for Experience Criterion

**Fetching data from database and generating a new csv file for rank model**

import pandas as pd

import numpy as np

#Degree Priority

degreeFrame = pd.read\_csv("DegreePriority.csv")

# print(degreeFrame.head(10))

print()

degreeScore = np.array(degreeFrame['AHP Score']).astype(int)

print(degreeScore)

#Experience Priority

experienceFrame = pd.read\_csv("ExperiencePriority.csv")

# print(experienceFrame.head(10))

print()

experienceScore = np.array(degreeFrame['AHP Score']).astype(int)

#Fetching Doctor Information

import json

import urllib.request

doctorInfo = urllib.request.urlopen("http://manojphuyal-001-site1.atempurl.com/api/GetDoctor")

doctorEducationInfo = urllib.request.urlopen("http://manojphuyal-001-site1.atempurl.com/api/GetDoctorDegree")

doctorInfoData = json.loads(doctorInfo.read().decode())

doctorEducationInfoData = json.loads(doctorEducationInfo.read().decode())

doctor\_id = []

doctor\_name = []

doctor\_experience = []

for i in range(len(doctorInfoData)):

doctor\_id.append(doctorInfoData[i]['Doctor\_ID'])

doctor\_name.append(doctorInfoData[i]['Doctor\_Name'])

doctor\_experience.append(doctorInfoData[i]['Doctor\_Experience'])

# Building a Table DataFrame

doctor\_degree = []

for i in range(len(doctor\_id)):

doctorEducationInfoArray = urllib.request.urlopen("http://manojphuyal-001-site1.atempurl.com/api/GetDoctorDegree?id="+str(doctor\_id[i]))

doctorEducationInfoDataArray = json.loads(doctorEducationInfoArray.read().decode())

tempArray=[]

for j in range(len(doctorEducationInfoDataArray)):

tempArray.append(doctorEducationInfoDataArray[j]['Doctor\_Degree\_Name'])

doctor\_degree.append(tempArray)

totalData = [doctor\_id,doctor\_name,doctor\_degree,doctor\_experience]

totalDataArray = np.array(totalData).transpose()

doctorFrame = pd.DataFrame(data=totalDataArray,columns=['ID','Doctor Name','Degree','Experience'])

doctorFrame.to\_csv(r'NewScoreLabelData.csv')

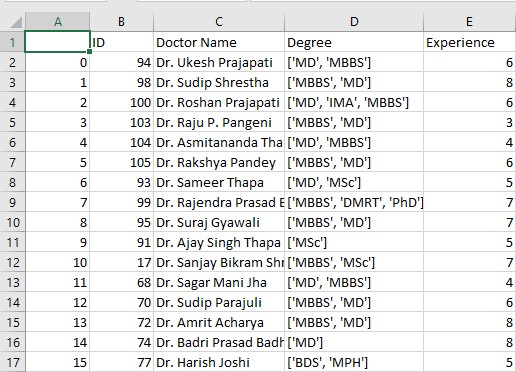


Table 3. Information on specific doctor on their degree and experience years

**Generating final rank model and posting AHP Score table to the database**

# Calculating Score and Points for Doctors

import pandas as pd

import numpy as np

import ast

#Importing Degree lables

scoreFrame = pd.read\_csv("NewScoreLabelData.csv",converters={'Degree':rm\_quote})

degreeFrame = pd.read\_csv("DegreePriority.csv")

experienceFrame = pd.read\_csv("ExperiencePriority.csv")

print(scoreFrame)

#Experience Years Priority

experienceYears = experienceFrame['Experience'].values

#Score for each Experience

experienceScore = experienceFrame['AHP Score'].values

#Score Value Generation i.e Score for Each Degree

score = degreeFrame['AHP Score']

scoreValue = []

for i in range(len(score)):

scoreValue.append(int(score[i]))

#Degree Value Generation from DegreePriorityCSV

degreeValue = degreeFrame['Degree']

#Generating Dictionary for each

dictionaryForDegree = dict(zip(degreeValue,scoreValue))

dictionaryForExperience = dict(zip(experienceYears,experienceScore))

# print(dictionaryForDegree)

# print(dictionaryForExperience)

#Degree in Score Format Calculation

degreeData = scoreFrame['Degree'].values

degreeData = [ast.literal\_eval(i) for i in degreeData]

print(degreeData)

pointsForDegree = []

for i in range(len(degreeData)):

for j in range(len(degreeData[i])):

temp = [dictionaryForDegree[j] for j in degreeData[i]]

pointsForDegree.append(temp)

# Summing Degree Score Inside Array

pointsForDegreeNormalized = []

for i in range(len(pointsForDegree)):

temp=sum(pointsForDegree[i])

pointsForDegreeNormalized.append(temp)

#Experience in Score Format Calculation

experienceData = scoreFrame['Experience'].values

pointsForExperience = []

pointsForExperience = [dictionaryForExperience[i] for i in experienceData]

#Totalling Degree Score and Experience Score

finalAHPScore = [sum(i) for i in zip(pointsForDegreeNormalized,pointsForExperience)]

finalAHPPoint = finalAHPScore

#Generating Score Table To New DataFrame

doctorData = scoreFrame['Doctor Name'].values

totalData = [doctor\_id,doctorData,finalAHPScore,finalAHPPoint]

totalDataArray = np.array(totalData).transpose()

scoreTable = pd.DataFrame(data =totalDataArray,columns=['Doctor ID','Doctor Name','AHP Score','AHP Point'] )

scoreTable.to\_csv(r'FinalScoreTable.csv')

#Posting Datas To Database

doctor\_id = scoreFrame['ID'].values

print(len(doctor\_id))

import json

import urllib.request

import requests

ahpList = urllib.request.urlopen("http://manojphuyal-001-site1.atempurl.com/api/GetDoctorAHP")

ahpListData = json.loads(ahpList.read().decode())

ahpPostURL = "http://manojphuyal-001-site1.atempurl.com/api/PostDoctorAHP"

def send\_data():

for j in range(len(doctor\_id)):

data = {

'AHP\_Score':finalAHPScore[j],

'AHP\_Point':finalAHPPoint[j],

'Doctor\_ID':doctor\_id[j]

}

requests.post(ahpPostURL,data=data)

send\_data()

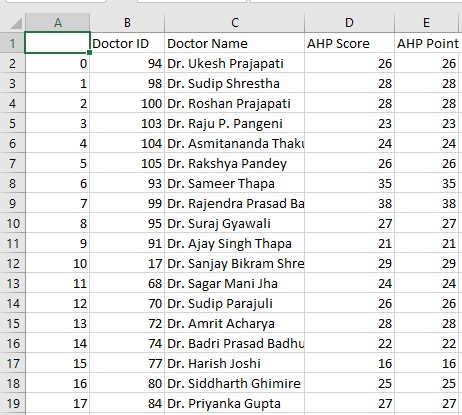
****

Table 4. AHP Score and AHP Point calculated for each doctors

**Naïve Bayes Classification for the feedback**

#Importing Datasets Offline

import sklearn.datasets as skd

# Positive and Negative Category

categories = ['com.positive','com.negative']

# Training Datas of Feedback

feedback\_train = skd.load\_files('feedback/train',categories=categories,encoding='ISO-8859-1')

#Testing Datas of Feedback

feedback\_test = skd.load\_files('feedback/test',categories=categories,encoding='ISO-8859-1')

# Train and Test are formed in dict files

## feedback\_test.keys()

## feedback\_train.target\_names

# => ['com.negative','com.positive']

#Word Count Vectorizer

from sklearn.feature\_extraction.text import CountVectorizer

count\_vect = CountVectorizer()

X\_train\_tf = count\_vect.fit\_transform(feedback\_train.data)

#Tfidf Transformer (Term Frequency)

from sklearn.feature\_extraction.text import TfidfTransformer

tfidf\_transformer = TfidfTransformer()

X\_train\_tfidf = tfidf\_transformer.fit\_transform(X\_train\_tf)

#X\_train\_tfidf.shape

# MultinomialNB used for the features with discrete values like word count 1,2,3.

from sklearn.naive\_bayes import MultinomialNB

clf = MultinomialNB().fit(X\_train\_tfidf,feedback\_train.target)

#Count Vectorizer and TfiDF for Test data

X\_test\_tf = count\_vect.transform(feedback\_test.data)

X\_test\_idf = tfidf\_transformer.transform(X\_test\_tf)

#prediction value for test data

predicted = clf.predict(X\_test\_idf)

#Checking Accuracy by comparing test target and predicted value

from sklearn import metrics

from sklearn.metrics import accuracy\_score

# print("Accuracy:",accuracy\_score(feedback\_test.target,predicted))

# Classification for New Feedbacks

import requests

# classified\_value

classified\_value = ''

def onRun(grabbedFeedback):

new\_feedback = [grabbedFeedback]

#Count Vectorization

X\_new\_counts = count\_vect.transform(new\_feedback)

#TFIDF (Term Frequency)

X\_new\_tfidf= tfidf\_transformer.transform(X\_new\_counts)

predicted\_new=clf.predict(X\_new\_tfidf)

#Gives the array value of predicted classification

global classified\_value

if(1 in predicted\_new):

classified\_value="Positive"

else:

classified\_value="Negative"

#START OF API

from flask import Flask

from flask\_restful import Resource

# from json import dumps

from flask import jsonify

from urllib.request import urlopen

import json

app = Flask(\_\_name\_\_)

@app.route('/<feedback>',methods=['GET','POST'])

def index(feedback):

some\_json = str(feedback)

onRun(some\_json)

result={'data':[{'classification':classified\_value,"api\_classification":"api exceeded"}]}

return jsonify(result), 201

@app.route('/',methods=['GET'])

def default():

return jsonify({'Message':'Pass Feedback at the end of the link'})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port='5002')

#END OF API