

# **AUTOMATED REMEDY SYSTEM FOR FEVER USING MACHINE LEARNING**

## **A PROJECT REPORT**

*Submitted by*

**SOMNATH ABHISHEK [Reg No: RA1411008010023]**

**IYAPPAN S. [Reg No: RA1411008010026]**

**UDIT GARG [Reg No: RA1411008010164]**

*Under the guidance of*

**Mr. L.N.B. SRINIVAS**

(Assistant Professor (S.G.), Department of Information Technology)

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

in

**INFORMATION TECHNOLOGY**

of

**FACULTY OF ENGINEERING AND TECHNOLOGY**



S.R.M. Nagar, Kattankulathur, Kancheepuram District

**MAY 2018**

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be **University** u/s 3 of UGC Act, 1956)

## BONAFIDE CERTIFICATE

Certified that this project report titled “**AUTOMATED REMEDY SYSTEM FOR FEVER USING MACHINE LEARNING**” is the bonafide work of “ **SOMNATH ABHISHEK [Reg No: RA1411008010023], IYAPPAN S. [Reg No: RA1411008010026], UDIT GARG [Reg No: RA1411008010164]**”, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

Mr. L.N.B. SRINIVAS  
**GUIDE**  
Assistant Professor (S.G.)  
Dept. of Information Technology

Signature of the Internal Examiner

**SIGNATURE**

Dr. G. Vadivu (Professor, Ph.D.)  
**HEAD OF THE DEPARTMENT**  
Dept. of Information Technology

Signature of the External Examiner

## **ABSTRACT**

This project aims at providing the first step of recovery that can be administered to the patient until the patient can visit the doctor. This is achieved by the use of machine learning to develop a Decision Support System which works by mimicking doctor's intuition on analyzing symptoms and inferring on the type of fever.

The application records symptoms of an experienced by a person that will be matched with a set of symptoms, which can result to a certain fever. The data will be provided by appropriate medical practitioner. This application has two interfaces. At the first sign of symptoms, a patient may enter their interface and specify the appropriate symptoms, which will be recorded by a questionnaire.

Based on the given information the system will identify the type of fever and suggest a medicinal drug or natural remedy for immediate relief as well as fix an appointment with a doctor closest to the locality of the patient. The prescriptions will be administered with proper dosage which the patient may take or consult doctor before taking them.

Existing interfaces like Web MD that can identify symptoms and find the doctor as well as find the prices of prescriptions, but has not been implemented in India.

## **ACKNOWLEDGEMENTS**

The success and the outcome of this project required guidance and assistance from different sources and we feel extremely fortunate to have this all along the completion of our project. Whatever we have done is largely due to such guidance and assistance and we would not forget to thank them.

We express our sincere thanks to the Head of Department, Department of Information Technology, Dr.G.Vadivu (Professor, Ph.D.), for all the help and infrastructure provided to us to complete this project successfully and for her valuable guidance.

We would like to express our heartiest gratitude to our guide, Mr.L.N.B.Srinivas (Assistant Professor), for his guidance, support, consistent encouragement, caring attitude and providing us with an excellent environment to complete the project. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to us for completing this research work.

We are thankful and fortunate enough to get constant encouragement and support from all the Teaching and Non-Teaching staff of the Department of Information Technology who have helped us in every little way in completing our Project Work.

**SOMNATH ABHISHEK (RA1411008010023)**

**IYAPPAN S. (RA1411008010026)**

**UDIT GARG (RA1411008010164)**

# TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>iv</b>
<b>LIST OF TABLES</b>	<b>viii</b>
<b>LIST OF FIGURES</b>	<b>x</b>
<b>ABBREVIATIONS</b>	<b>xi</b>
<b>LIST OF SYMBOLS</b>	<b>xii</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Health-care Analytics . . . . .	2
1.1.1 Machine Learning in Health Care: . . . . .	2
1.2 Automated Remedy System for Fever using Machine Learning . . . .	3
1.3 Project Outline . . . . .	5
<b>2 LITERATURE REVIEW</b>	<b>6</b>
2.1 Existing Systems . . . . .	6
2.1.1 Web MD . . . . .	6
2.1.2 Practo . . . . .	7
2.2 Related Works . . . . .	8
<b>3 DATA ANALYSIS AND MODELLING</b>	<b>9</b>
3.1 Overview . . . . .	9
3.2 Doctor's Intuition in Fever Diagnosis . . . . .	9
3.3 Symptoms Listing . . . . .	10
3.4 Type of Fever Classes . . . . .	11
3.5 Web interface for Data Collection . . . . .	12

3.6	Data Overview . . . . .	14
3.7	Data Preprocessing . . . . .	14
3.8	Exploratory Data Analysis and Key Insights . . . . .	18
3.8.1	Stratified Shuffle Split . . . . .	20
3.9	Machine Learning Models and Validation Methods . . . . .	21
3.9.1	Machine Learning Models: . . . . .	23
3.9.2	Validation Metrics: . . . . .	27
3.10	Model Deployment Methods . . . . .	29
<b>4</b>	<b>SYSTEM DESIGN</b>	<b>31</b>
4.1	USE-CASE DIAGRAM . . . . .	31
4.2	ARCHITECTURE DIAGRAM . . . . .	32
4.3	DATA FLOW DIAGRAM . . . . .	32
4.4	SEQUENCE DIAGRAM . . . . .	33
<b>5</b>	<b>CODING AND TESTING</b>	<b>34</b>
5.1	Overview . . . . .	34
5.2	Django Framework . . . . .	34
5.3	Code . . . . .	35
5.4	Testing . . . . .	41
5.4.1	Home-Page . . . . .	41
5.4.2	Testing Prediction . . . . .	42
5.4.3	Testing Appointments . . . . .	44
5.5	Test Cases: . . . . .	46
<b>6</b>	<b>RESULTS AND ANALYSIS</b>	<b>47</b>
6.1	RANDOM FOREST CLASSIFIER: . . . . .	47
6.2	ADAPTIVE BOOSTING CLASSIFIER: . . . . .	48
6.3	BERNOULLI'S NAIVE BAYES CLASSIFIER: . . . . .	49
6.4	RESULTS OF K-FOLD CROSS VALIDATION: . . . . .	49
6.5	Inferences and Insights . . . . .	51
<b>7</b>	<b>CONCLUSION</b>	<b>53</b>

<b>8</b>	<b>FUTURE ENHANCEMENTS</b>	<b>54</b>
<b>A</b>	<b>DJANGO SET-UP</b>	<b>55</b>
A.1	Python Installation: . . . . .	55
A.2	Installing Django Package: . . . . .	56
A.3	Creating a Basic Django Project: . . . . .	57
A.4	Setting up Custom Hostname for Django Apps: . . . . .	58
<b>B</b>	<b>JINJA TEMPLATING</b>	<b>60</b>

## LIST OF TABLES

3.1	<b>Centers (Clinics and Health Centers) visited for Data Collection</b>	13
3.2	<b>Sampling Comparison . . . . .</b>	21
3.3	<b>Summary of Models . . . . .</b>	27
5.1	<b>Test Cases . . . . .</b>	46



## LIST OF FIGURES

2.1	Find a Doctor - Web MD . . . . .	7
2.2	Practo in India . . . . .	8
3.1	Data Collection Through Shiny Web interface (a) . . . . .	13
3.2	Data Collection Through Shiny Web interface (b) . . . . .	13
3.3	Data Dictionary Size: 1016 Rows i.e. Patients . . . . .	14
3.4	Data Labelling . . . . .	16
3.5	Data Labelling with Names . . . . .	17
3.6	Labelled Data . . . . .	17
3.7	Exploratory Data Analysis . . . . .	19
3.8	Preparing Stratified Shuffle Split . . . . .	20
3.9	Stratified Shuffle Split inference . . . . .	21
3.10	Example: Binary Classification Scenario . . . . .	28
3.11	Saving Model as Pickle file . . . . .	30
4.1	Use Case Diagram . . . . .	31
4.2	Component Diagram . . . . .	32
4.3	Data Flow Diagram . . . . .	33
4.4	Sequence Diagram . . . . .	33
5.1	Django Architecture . . . . .	34
5.2	Django Forms for Symptoms Input . . . . .	35
5.3	Views in Prediction (a) . . . . .	36
5.4	Views in Prediction (b) . . . . .	36
5.5	URLs in Prediction . . . . .	37
5.6	Models for Prediction . . . . .	37
5.7	ML Module using symptoms . . . . .	38
5.8	Prescriptions Module (a) . . . . .	38

5.9	Appointments Views File, delegating appointments . . . . .	39
5.10	Appointment URLs . . . . .	39
5.11	Part of Appointment Data . . . . .	40
5.12	Django Server Running successfully . . . . .	40
5.13	ARS Health Home . . . . .	41
5.14	On Clicking Prediction . . . . .	42
5.15	Fill the Details . . . . .	42
5.16	Non Allergic Symptoms . . . . .	43
5.17	Results from symptoms . . . . .	43
5.18	Consulting after predictions . . . . .	44
5.19	Consultation form . . . . .	45
5.20	Appointment Booked . . . . .	45
6.1	Random Forest Classification . . . . .	47
6.2	Random Forest Classification - Feature Importance . . . . .	47
6.3	AdaBoost . . . . .	48
6.4	AdaBoost Classification - Feature Importance . . . . .	48
6.5	Naive Bayes Classification . . . . .	49
6.6	K-Fold Validations . . . . .	50

## ABBREVIATIONS

<b>PaaS</b>	Cloud Platform as a Service
<b>API</b>	Application Programming Interface
<b>CSV</b>	Comma Separated Values
<b>EDA</b>	Exploratory Data Analysis
<b>GPS</b>	Global Positioning Satellite
<b>MVC</b>	Model-View-Controller
<b>NB</b>	Naive Bayes
<b>SQL</b>	Structured Query Language
<b>SVM</b>	Support Vector Machines
<b>TB</b>	Tuberculosis
<b>UI</b>	User Interface

## LIST OF SYMBOLS

$\Sigma$       Summation of Series

# CHAPTER 1

## INTRODUCTION

Health is one of the major aspect which defines aesthetic needs of our life. In this fast moving world, where the nutrition of the people and working hours have changed drastically in the last two decades, we have seen dreadful impact of new complicated diseases, illness and other problems affecting the general working population and their lifestyle.

People are also inclined to quick cures with comes along with different side-effects, instead of looking at the longer run. Time and money has played a diverse role in shrinking the diagnosis period and also it has been distancing the quality of diagnosis from the people by assigning a weight of higher cost since industrialization and rapid development came into picture.

Expert and immediate guidance of doctors is far beyond reach if the patients do not have enough money or time to get themselves up in the mid-night at the end of the month and start walking towards a global hospital.

In India, people settle for getting diagnosed by newly graduated M.B.B.S professionals due to fact that the pricing is less. The probability of a patient getting cured by the medicine prescribed by a newly graduated professional is around the centre of the Normal curve and it is for a fact that these newly graduated professionals lack experience and lack knowledge of industrial standard medicines and new combinations.

Thus the probability of people going for second opinion and diagnosis is increasing exponentially these days. This leads to objectifying the patient's body like a guinea-pig where different medicines and drugs are given to find out the right cure.

Now with everything in the form of a Mobile App starting from ordering food till finding your life partner and along with age of artificial intelligence and machine learning, it is possible to mimic the diagnosis of expert doctors and create the first-aid for the people to trust and follow in case of emergency situations and as well as in situations where people do not trust the medical services available in their locality.

## 1.1 Health-care Analytics

health-care Analytics is the process of deriving insights from patterns and correlations in data that can be used to make better health-care decisions. health-care Analytics moves beyond data management to find meaning in real-time or historical data, and making predictions about the future to improve the probability of success.

Health systems can also use health analytics to gain insight into numerous areas of health-care, including patient and physician engagement, revenue, risk, and population health initiatives. Using health analytics helps health-care organizations get a clear picture of current operations and make data-backed decisions about how to improve outcomes in the future.

Harnessing the power of all the data being collected and generated by health-care organizations is extremely important and synthesizing and analyzing health-care data can influence patient outcomes, create diversification, and drive revenue growth.

While health-care data management and analysis exist as two different stages of the health analytics process, the truth is that analytics can't exist without data. Before analytics can begin, health systems need to acquire, capture, cleanse, and integrate data from multiple sources with the help of a centralized health-care analytics platform.

In that sense, there is an interlacing at the point where data preparation ends and where analytics begins. With valid and accurate data in place, health-care enterprises can begin the process of interpreting the data to inform future interactions with patients, prospects, and populations.

### 1.1.1 Machine Learning in Health Care:

*Machine learning* is the new age technology in the field of computer science that uses statistical techniques which gives the machine the ability to learn and predict the results in the future without explicitly specifying each condition through complex and long codes.

The algorithms learn from the past data, understand patterns available in the data

and are able to predict the scenarios and values given a set of features or independent variables. Usage of machine learning in data analytics predominantly occupies the space of predictive modelling and predictive analytics.

In the field of health care, there are immense use-cases and problems which can be solved using Machine learning and deep neural network algorithms. As the domain itself revolves around associations, probability, identification based upon the past data, the patterns and rules formulated over past researches done in the field of medicine, machine learning algorithms can be made to learn the patterns, associations and with the help of technology, a certain level of automation can be done to assist the medical practitioners in diagnosis.

Algorithms such as Deep neural networks for image classification for tumour detection in cancer patients perform really well under the world famous IBM Watson. Apart from the Neural Networks, algorithms such as Decision trees, apriori, Naive Bayes classifiers are among the few to be used for different classification scenarios that arise in predictive analytics in health-care.

## **1.2 Automated Remedy System for Fever using Machine Learning**

In the field of Medicine and Diagnosis for Fever, the general approach followed by the doctors has been to first identify the particular type of fever after interrogation with the patients about the symptoms and it is followed by suggestion of medicines in the form prescription to reduce the effects of symptoms as well as combat the cause of the fever.

The First Health Consultation with the doctor makes efforts to reduce the effect of symptoms. But this is not always possible due to the following facts:

- Lack of Availability of Doctor
- Lack of Availability of Appointment to visit a Doctor
- Lack of experienced Doctors near the locality
- Emergency consultancy needed at odd times of the day or night

With the availability of technologies such as Machine Learning, Artificial Intelligence and Big data and along with the concepts of Distributed App Development and Frameworks we can address the above problems by automating the whole process of First Health Consultation and deliver it to the end users i.e. Patients, to use in emergency situations.

The Machine learning layer can be introduced to predict the type of fever from the given set of symptoms from the user/patient and prescribe medicines for reducing the symptoms. Robust non-linear and probabilistic models such as Random Forest, Adaptive Boosting Classifier (Ensemble Techniques) and Bayesian classifiers can mimic the doctor's intuition and experience in identifying the type of fever the patient experiences.

By collecting data of the past diagnosis done by experienced doctors and training the machine learning model in regular intervals can increase the accuracy and precision of the model leading to very accurate identifications of fever. With the power of Big Data, we can leverage the dynamic training and improvisation of models based on the locality and associated cases in the locality. Thus by giving the service of expert health consultation to all people who are in need at any time of the day.

The proposed system will be able to prescribe appropriate general medicines and natural remedies after identification of the Fever class. The patient has to go to the doctor for detailed examination in cases the symptoms does not subside within a day.

The web application will also be able to book appointments with nearby doctors by accessing the Global Positioning Satellite (GPS) Location of the patients.



## 1.3 Project Outline

The results of the project carried out under the title **Automated Remedy System for Fever using Machine Learning** have been presented in following chapters:

**Chapter 1** gives an introduction about the scope of health-care analytics and its importance in the field of medicine. It also discusses about the usage of machine learning and Deep learning algorithms and how are they used to create Decision Support Systems for doctors to perform faster and accurate diagnosis.

**Chapter 2** gives a brief introduction of how a doctor diagnoses a patient and the intuition behind this diagnosis. It also gives a brief of previous works done in fever classification ,the symptoms that are taken into consideration and the number of fever classes which will be classified by the machine learning algorithm along with giving a picture of the web platforms which are into health analytics.

**Chapter 3** tells about the procedures and methodology used to data collection, the list of clinics that were used to collect data. It also takes a deep dive in the data overview, exploratory analysis, inferential statistics and the data pre-processing template that has been used. It also gives the list of narrowed down machine learning models in accordance with intuition of the doctor in identifying a fever category.

**Chapter 4** gives a detailed view of the Web Platform that was built by diagrammatic representations of Architecture, Data flow model, sequence model and Use case model

**Chapter 5** gives the code snippets of the User interface, Back-end data flow and Machine learning modules. It also gives the snapshots of the working Web application during the testing phase of the system

**Chapter 6** gives a detailed comparative study of all the results obtained in different Machine learning models.

**Chapter 7** is the conclusion that summarizes the research conducted, its outcome and its limitations. In addition, the future scope of research has been envisioned that will enhance the efficiency of Automated Remedy System. The emerging technologies are highlighted as how it could be helpful for further development of the application.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Existing Systems**

The concept of online doctors started during the early-2000s where the health-care community started to offer their service on the internet. Many health-care applications such as Web MD, DocOnline, Practo, etc. have been used by people in recent times. Most of them offer consultation with doctors either free or with some subscription based charge. The Doctors collaborate with them to get a chat-like interaction with patient and understand their ailments and suggest them a treatment. Although, it's debatable that the acceptance of online health-care applications have not been widely accepted. This is due to the applications do not provide the proper diagnosis procedures to arrive at relevant conclusions. Also certain doctors prescribe some drugs which can harm patients, to promote pharmaceutical companies.

##### **2.1.1 Web MD**

It is one of the first websites which provided Medical Support and Consultancy in US. Founded in 1996, it's still the most popular website for Health-care and Pharmaceutical information. It provides a symptom-based analysis of a disease. It describes what causes the symptoms and offers to suggest relevant treatment procedures. These information is contributed by doctors from around the world. The website also provides information on a certain drug, its variants and its usage. It warns the user if the drug has any harmful effects or previous cases of health damage. Hence it can be also used a search-engine as it has a huge database of diseases, pharmacy and health-care information. Web MD has a latest feature of Symptom Checker which takes in symptoms of the user and matches it with their vast repository of known-Diseases and ranks the closest set of diseases. Then it provides information for the cure of the symptoms or related diseases.

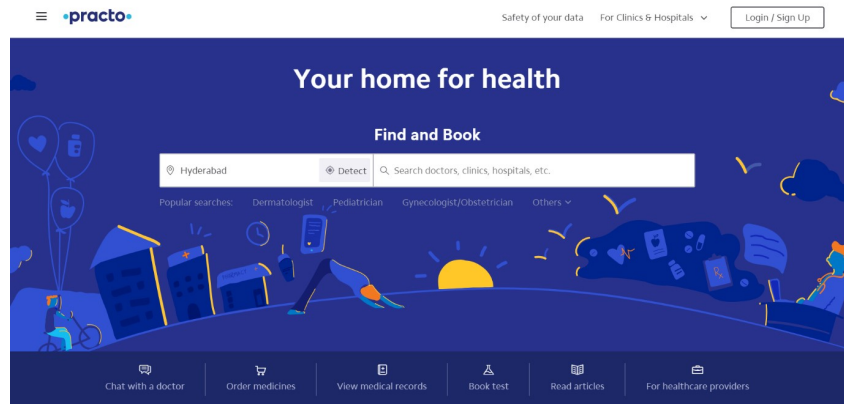
**Figure 2.1:** Find a Doctor - Web MD

The *Find a Doctor* 2.1 feature enables a user to find a Physician closest to their location. This however works for US as it is designed for US consumers at focus. Overall, the website has been widely successful as it offers loosely-coupled set of features which gives the end-user to explore the world of health-care.

### 2.1.2 Practo

Practo is one of the websites gaining popularity in India for booking appointments with doctors and getting online consultations. The main features include chatting with doctors, booking appointments, order medicines and booking diagnostic tests. The system works on profile based services and provides subscriptions for certain premium services.

Practo 2.2 follows the traditional approach Online Doctor paradigm which provides health services by collaborating with doctors and clinics around the country. The site also provides Health information on various diseases as a reference for the user which can help them find the right doctor for their needs.



**Figure 2.2:** Practo in India

## 2.2 Related Works

Oguntimilehin et al. (2013) states the idea of diagnosis of Typhoid Fever and its classification. Machine learning has been implemented over a labelled dataset of different severity levels. The method used by them was Rough Set Theory and was implemented using Visual Basic and MySQL Database.

In some research, Fever has been classified as and remedial measures has been administered using machine learning. This has been explained by (Oo et al., 2016). The condition is by analyzing a set of symptoms which are fed into a Decision Tree which determines the type of fever. It recommends then domestic measures and remedies to cure the fever. Similar work has been done as a predictive analysis of undifferentiated fevers where human body temperature of patients have been used with Support Vector Machines (SVM) algorithm (Dakappa et al., 2017).

Fatima and Pasha (2017) give an insight on variety of machine learning algorithms that can be used to determine the type of diseases. It compares the usages of such algorithms with the success rates of other researches.

## **CHAPTER 3**

### **DATA ANALYSIS AND MODELLING**

#### **3.1 Overview**

This chapter talks about Doctor's Intuition of Diagnosis of a Disease and the data which we collected and its variations. It will also explain the symptoms of fever and methods of collection, preprocessing and cleaning of data before we build our machine learning model.

#### **3.2 Doctor's Intuition in Fever Diagnosis**

One of the exceptional challenges was to understand the hybrid methodologies doctors use to diagnose patients. The objective was to find Machine learning algorithms that work partially or similar to a doctor's intuition in diagnosing a patient in the case of fever and other general symptoms and prescribe medicines for it.

After hours of discussion with expert doctors, the following inference came out which they use to handle cases with fever during the first consultation:

- Associate certain symptoms which mainly contribute to a type of fever
- Give a weightage to the recent type of fevers that is spreading in particular locality and during a particular season
- First level of diagnosis is symptomatic one if in case they identify the fever to be of Viral type after examining and questioning the patient.
- In case of Bacterial, the dosage of the drugs and medicines are increased and a continuous monitoring is done after first consultation

From the above discussions and inferences in understanding of the doctor's intuition in diagnosing a patient in the first consultation, the main points that were narrowed down while choosing machine learning models are as follows:

- The model must be able to do associations and understand patterns from the past data
- (Associate Rule mappings: Symptom -> Fever)
- The model must also give a weightage to the recent type of fever spreading in locality i.e. confidence and Support of combination of symptoms
- The model should be a hybrid between having a trade-off of information gain (entropy) and Posterior Probability which mimics the doctor's intuition

### 3.3 Symptoms Listing

The various symptoms as suggested by the doctor to identify the type of fever of the patient are:

- **Body Temperature:** The temperature of the body ranging from 97-degree F to 104-degree F
- **Type of Temperature Rise:**
  - \* **Continuous:** The phenomenon of having a constant body temperature throughout the day during fever
  - \* **Irregular Temperature Rise/ Night Rise:** The phenomenon of having random rise of body temperature in the night and day during fever
  - \* **Chills:** The phenomenon of having sudden shivering during fever
- **Type of Body Pain:**
  - \* **Weakness and Fatigue:** General Weakness and Fatigue experienced during fever
  - \* **Joint Pain:** Specific pain in the joints (Shoulders, knee and elbow)
  - \* **Chest Pain:** Pain in the chest and suffocation in breathing
  - \* **Stiffness:** Stiffness of the body increases followed by uneasiness in muscles and joints
- **Nasal Symptoms:**
  - \* **Running Nose:** Flow of mucus from the nasal cavities

- \* **Nasal Blockage:** Difficulty in breathing through the nasal cavities
- **Throat Symptoms:**
  - \* **Dry Cough:** Excess Cough without Mucus
  - \* **Sore Throat:** Itchiness in the throat
  - \* **Cough with Sputum:** Cough originating from Chest with Mucus
- **Vomiting**
- **Bowel Movement**
- **Urinary Infection**

The Symptoms listed above are framed in accordance with ease of user's (patients) interpretation and understanding ability.

These are the symptoms which are necessary for classifying a particular type of fever and moving for further diagnosis. The further intrinsic interrogation with the patient by the doctor to identify other specific symptoms is done after the first step of classification or identification of the type of fever of the patient.

### 3.4 Type of Fever Classes

According to our research and consultation with experts, the inference comes out as follows:

- Fevers can be broadly categorized into 3 categories:
  - \* Viral
  - \* Bacterial
  - \* Protozoan
- Protozoan have lot of sub categorical fevers which can be identified only using laboratory tests.
- Viral Fever encompasses Common Flu, Viral Fever, Influenza, Dengue, Malaria, Respiratory Tract Infection, Urinary Tract infection and Chikungunya.
- Bacterial Fever encompasses typhoid enteritis Fever, Appendicular abscess, Pneumonia, Tuberculosis (TB).

- Protozoan Fever encompasses Severe infections (of lungs; of bones; of brain; inside abdomen), connective tissue disease (Arthritis), infection in the heart(bacterial endocarditis), Osteomyelitis (bone infection).
- The most common protozoan fever that tend to occur is Malaria and its variants.

The suggestion from consultation panel was to classify fever into 3 categories:

- Viral
- Bacterial
- Malarial

Due to the fact that the application simulates emergency and first â€š level diagnosis and the above mentioned categories are the first step of identification a Medical practitioner does before prescribing medicines or going for further interrogation.

### 3.5 Web interface for Data Collection

Our Data Collection for analysis of the said methods and intuition was made possible by a series of steps. This section talks about the method we used to collect raw data from Patients.

1. A Global Hospital and three clinics where selected for data collection of patients over a month to get at least a diverse sample of cases and a minimum of 1000 data tuple.
2. A Web User Interface (UI) 3.2 was developed for data collection in Shiny Framework in R and was deployed to Cloud Platform as a Service (PaaS) platform named Shinyapps.io. The URL was handed over to the clinics and the respective doctors.
3. After post-diagnosis, the patients were asked to input their symptoms before leaving the clinic.
4. The doctor would input the type of fever he/she has identified based on the symptoms of the patient. The machine learning algorithm has to mimic the intuition of the doctor and predict the fever class in the same method i.e. Learning from recent past and also association rules (Support and lift ratio of the symptoms)
5. The data was collected in the Structured Query Language (SQL) database in the back-end which was further processed as a Comma Separated Values (CSV) file for data analysis and modelling



MED SURVEY

READ ME  
SYMPTOMS INPUT

## DISCLAIMER

The data collected through the app is solely used for **Research purposes** only.

**No personal information** of the patient are recorded.

The symptoms are to be filled by the **patient** and the Fever class has to be filled by the **doctor** and will be updated.

Please **DO NOT** type any other additional information (phone numbers, name, email etc) if told by anyone !!

**Figure 3.1:** Data Collection Through Shiny Web interface (a)

MED SURVEY

READ ME  
SYMPTOMS INPUT

### GENERAL DETAILS

**GENDER**

MALE

**Do you have allergies to general medications ?**

Yes

**What was your body Temperature?**

0

**What was your Age at that Time?**

0

**To be filled by the patient**

**Did you have Cold or Nasal Congestion?**

Running Nose

**Did you have throat related symptoms?**

Sore Throat

**Did you any irritation while passing urine?**

No

**Any other specific symptoms?**

**To be filled by the patient**

**How was your body temperature?**

Continous

**How was your body pain?**

Stiffness and Swells

**Did you have Vomitting?**

No

**Did you have Loose motion**

No

**To be filled by the DOCTOR**

**What is the Fever Class ?**

Viral

**Update** **Refresh**

**Figure 3.2:** Data Collection Through Shiny Web interface (b)

**Table 3.1: Centers (Clinics and Health Centers) visited for Data Collection**

CENTER	ADDRESS	DOCTOR
<b>SREE BALAJI MEDICAL HOSPITAL</b>	7, Works Road, Chromepet, Chennai, Tamil Nadu 600044	Dr. V. Padma (M.B.B.S, M.D)
<b>GEM CLINIC</b>	19, Agaramthen Road, Mappedu post , Chennai, Tamil Nadu 600126	Dr. Rini Rajan (M.B.B.S, M.D)
<b>REVATHY CLINIC</b>	19, Agaramthen Road, Balaji Nagar, Chennai, Tamil Nadu 600074	Dr. Revathy (M.B.B.S)

## 3.6 Data Overview

The following is the excerpts of the CSV file which has been processed from the Cloud Database. As mentioned before, the data has been collected from the centers and it is stored in SQL database in the back end of the Web application made for data collection. This is the data of 1016 patients collected over a month in the mentioned centers. 3.3

	Gender	Allergy	Age	Btemp	Tof	Bpain	Vomitting	Lmotion	Cold	Cough	UI	Fever_Class
0	Male	No	21	103.7	Frequent Triggers	Weakness & Fatigue	No	No	No	Sore throat	No	BACTERIAL
1	Male	No	45	102	Continuous	Weakness & Fatigue	No	No	Nose blockage	Cough with sputum	No	VIRAL
2	Female	No	20	103	Continuous	Weakness & Fatigue	No	No	No	None	No	MALARIAL
3	Male	No	21	104	Frequent Triggers	Stiffness and swells	No	No	Running Nose	Dry cough	No	MALARIAL
4	Male	No	19	100	Night Rise	Joint Pain & Lower back pain	Yes	Yes	Running Nose	Sore throat	No	MALARIAL
5	Male	No	21	102	Night Rise	Joint Pain & Lower back pain	No	No	Nose blockage	Sore throat	No	BACTERIAL
6	Male	Yes	20	102	Continuous	Weakness & Fatigue	No	No	No	None	No	VIRAL
7	Male	No	21	101	Frequent Triggers	Weakness & Fatigue	No	No	Nose blockage	Dry cough	No	BACTERIAL
8	Male	No	26	101	Continuous	Weakness & Fatigue	No	No	Nose blockage	Dry cough	No	BACTERIAL
9	Male	No	32	104	Continuous	Weakness & Fatigue	No	No	No	None	No	MALARIAL
10	Male	No	38	101	Frequent Triggers	Weakness & Fatigue	No	No	No	Cough with sputum	No	MALARIAL
11	Male	No	32	101	Continuous	Stiffness and swells	No	No	Nose blockage	Sore throat	No	BACTERIAL
12	Male	No	39	104	Night Rise	Weakness & Fatigue	Yes	No	Nose blockage	Dry cough	No	BACTERIAL
13	Male	No	33	102	Night Rise	Weakness & Fatigue	No	No	No	Dry cough	No	MALARIAL
14	Female	No	26	104	Night Rise	Weakness & Fatigue	No	No	Running Nose	None	No	BACTERIAL
15	Male	No	20	103	Continuous	Weakness & Fatigue	No	No	No	Dry cough	No	BACTERIAL
16	Female	No	21	102.5	Night Rise	Stiffness and swells	Yes	No	Nose blockage	Sore throat	No	MALARIAL
17	Female	No	20	102	Night Rise	Weakness & Fatigue	No	No	Nose blockage	Dry cough	No	VIRAL
18	Male	No	18	103	Night Rise	Weakness & Fatigue	No	No	Nose blockage	Dry cough	No	MALARIAL
19	Male	No	20	103	Night Rise	Weakness & Fatigue	No	No	Nose blockage	Dry cough	No	VIRAL
20	Female	No	41	101	Continuous	Weakness & Fatigue	No	No	Running Nose	Sore throat	No	VIRAL
21	Male	No	17	102	Continuous	Weakness & Fatigue	No	No	No	None	No	VIRAL

**Figure 3.3:** Data Dictionary Size: 1016 Rows i.e. Patients

## 3.7 Data Preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. This is done in order to make the data into a understandable format for the machine learning algorithms for training.

These are the following major types of processing:

- **Cleaning:** Removing outliers, filling missing NA values
- **Transformation:** Normalising or standard scaling the data
- **Reduction:** Binning, dummy variable reduction

The following concepts of data preprocessing will be used to make this dataset ready for modelling:

- **Dummy variables: Data Transformation:**

The process of splitting the variables with multiple categories into individual mutually exclusive features to ensure the smoothness of the model performance.

E.g.: Cough has 3 categories, The Feature Cough is transformed into Dry Cough, Sore Throat and Cough with Sputum as Individual columns (Features) Representing (1/0) scenario.

- **Encoding: Data Reduction:**

This is done to convert a string representation of (Yes/No) to 1/0 for Features like Allergy, Urinary Infection etc.

The following are the snippets of the data preprocessing done in Python in the Jupyter Notebooks. [in the figures 3.4, 3.5 ]

## LABELLING THE FEATURES

- 1.All the categories are in a feature vector are given a number.
- 2.This encoding is done for the machine learning algorithms to function. The algorithms would not function with features encoded in strings.

The following are the encoding functions.

```
1 def gen(g):
2     if g == "Male":
3         return 1
4     else :
5         return 0
6
7
8 def allergy(a):
9     if a == "No":
10        return 0
11    else:
12        return 1
13
14
15 def tof(t):
16     if t == "Continuous":
17         return 0
18     elif t == "Night Rise":
19         return 2
20     else:
21         return 1 #chills
22
23
24 def Bpain(bp):
25     if bp == "Weakness & Fatigue":
26         return 0
27     elif bp == "Joint Pain & Lower back pain":
28         return 1
29     elif bp == "Chest Pain":
30         return 2
31     else :
32         return 3 #stiffness and swells
33
34
35 def Vomit(v):
36
37     if v == "No":
38         return 0
39     else :
40         return 1
41
42
43 def lmo (l):
44
45     if l=="No":
46         return 0
47     else :
48         return 1
49
50 def cold(c):
51
52     if c == "No":
53         return 0
54     elif c == "Running Nose":
55         return 1
56     else :
57         return 2
58
59 def cough(co):
60
61     if co == "Dry cough":
62         return 1
63     elif co == "Sore throat":
64         return 2
65     elif co == "Cough with sputum":
66         return 3
67     else:
68         return 0
69
70 def ui(u):
71     if u == "Yes":
72         return 1
73     else :
74         return 0
75
```

Figure 3.4: Data Labelling

```

In [17]: 1 Tof = pd.get_dummies(df['Tof'])

In [18]: 1 Bpain= pd.get_dummies(df['Bpain'])

In [19]: 1 cold = pd.get_dummies(df['Cold'])

In [20]: 1 cough = pd.get_dummies(df['Cough'])

In [21]: 1 Tof.columns = ["Continuous", "Chills", "Night Rise"]

In [22]: 1 Bpain.columns = ["Weakness & Fatigue", "Joint Pain", "Chest Pain", "Stiffness"]

In [23]: 1 cold.columns = ['No_cold', 'Running Nose', 'Nose Blockage']

In [24]: 1 cough.columns = ['No_cough', 'Dry Cough', 'Sore Throat', 'Cough with Sputum']

In [25]: 1 df.drop(['Tof', 'Bpain', 'Cough', 'Cold'],axis = 1 , inplace = True)

In [26]: 1 df = pd.concat([df,Tof,Bpain,cold,cough],axis = 1)[['Btemp','Continuous',
2         'Night Rise', 'Chills', 'Weakness & Fatigue', 'Joint Pain', 'Chest Pain',
3         'Stiffness', 'No_cold', 'Running Nose', 'Nose Blockage', 'No_cough',
4         'Dry Cough', 'Sore Throat', 'Cough with Sputum', 'Vomitting', 'Lmotion', 'UI', 'Fever_Class']]

```

We are defining creating dummy variables due to reduce the collinearity and smoothen the machine learning model

Reason

1. Cold column has 3 categories represented in 3 integers (0,1,2) and similarly cough (0,1,2,3).
2. Even though we know for a fact that these numbers represent the occurrence of different events i.e. Symptoms, but the numbers represent the magnitude resulting in high correlation.
3. Highy correlation will bring down the performance of the machine learning algorithm.
4. Hence we are using the dummy variables to remove the categorical variables.

**Figure 3.5: Data Labelling with Names**

```

In [27]: 1 df.head(10)

```

Out[27]:

	Btemp	Continuous	Night Rise	Chills	Weakness & Fatigue	Joint Pain	Chest Pain	Stiffness	No_cold	Running Nose	Nose Blockage	No_cough	Dry Cough	Sore Throat	Cough with Sputum	Vomitting	Lmotion	UI
0	101.0	0	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0
1	100.0	0	0	1	0	1	0	0	0	1	0	0	0	0	1	1	1	0
2	102.0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0
3	100.0	0	0	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0
4	100.0	0	0	1	1	0	0	0	0	0	1	0	1	0	0	1	0	0
5	101.0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
6	100.0	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	1
7	100.0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0
8	102.0	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0
9	104.0	0	0	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0

**Figure 3.6: Labelled Data**

After doing the above mentioned pre-processing steps the data set has changed its shape to 18 X 1016 Matrix. 3.6

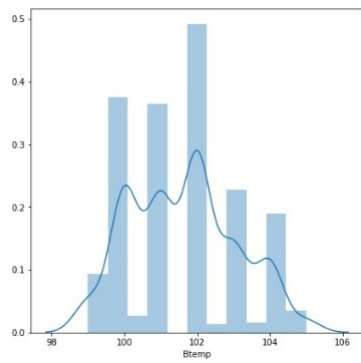
### **3.8 Exploratory Data Analysis and Key Insights**

Exploratory Data Analysis (EDA) is done in order to understand the distribution, nature and patterns present in the given data. It is the most important step in the analytics due to the fact that it gives the idea about the further data transformations that have to be done for the machine learning algorithms to give high results.

The objective of the EDA in the current problem statement is to examine the following cases:

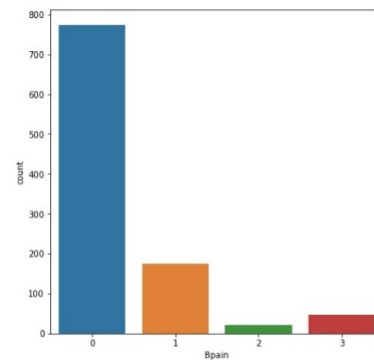
- Class imbalance Ratios of Categorical Variables.
- The homogeneity of the dataset (Strata)
- The distribution of the Body Temperature to check outliers
- Skewness of the distribution of Body Temperature
- Descriptive Statistics
- The mean of the Body Temperature is 101.6 Fahrenheit. 3.7(a)
- This is the ideal Fever temperature for any patient
- There is no Skewness depicted in the graph which further gives us insights about the presence of outliers; they are very minimal.
- There are more 2 features which suffer a huge class imbalance ratio. 3.7(b)
- The proportion of the instances in the test and train will differ drastically if the Train and Test is done by the Random Shuffling method.
- Due to this imbalance, the model may be more biased towards a certain set of combinations of fever. 3.7(c)
- To Reduce the Bias of the machine learning model which is more susceptible to happen from above inferences taken, Stratified Shuffled Sampling method must be done while splitting the dataset into Test and Train.

```
In [7]: 1 sns.distplot(df['Btemp'])
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x113ba3ac8>
```



(a) Distribution of Body temperature

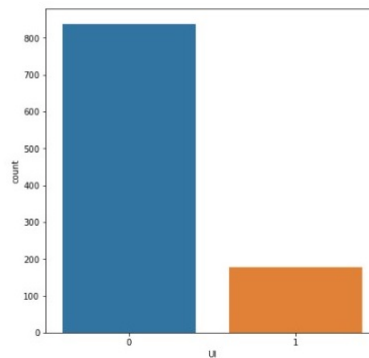
```
In [11]: 1 sns.countplot(x = "Bpain", data = df)
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x113eee8d0>
```



The mode of distribution centered around 0 : General Weakness and Fatigue.  
There is huge difference the ratios of the other classes.  
This may pull down the accuracy of the classification

(b) Distribution of Body Pain Vector

```
In [15]: 1 sns.countplot(x="UI", data= df)
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1142c5550>
```



(c) Urine Infection Count

**Figure 3.7: Exploratory Data Analysis**

## 1. Preparing Stratified Shuffled Split

```
In [4]: 1 from sklearn.model_selection import StratifiedShuffleSplit

In [5]: 1 split = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=0)
        2
        3 for train_index, test_index in split.split(df, df['Fever_Class']):
        4     train = df.iloc[train_index]
        5     test = df.iloc[test_index]

In [6]: 1 len(train)
Out[6]: 710

In [7]: 1 len(test)
Out[7]: 305

In [8]: 1 pd.value_counts(train['Fever_Class'])
Out[8]: VIRAL      287
        BACTERIAL  212
        MALARIAL   211
        Name: Fever_Class, dtype: int64

In [9]: 1 pd.value_counts(test['Fever_Class'])
Out[9]: VIRAL      123
        BACTERIAL   91
        MALARIAL    91
        Name: Fever_Class, dtype: int64
```

**Figure 3.8:** Preparing Stratified Shuffle Split

### 3.8.1 Stratified Shuffle Split

Stratified Shuffled Split (, SciKitLearn) is sampling method which maintains the homogeneity of the data sample which is also termed as “Strat”. The population is divided into homogeneous subgroups called Strata, and the right number of instances of each sample is sampled and split in the Train and Test samples respectively.

E.g. If your classifier variables has two classes Yes/No and from EDA you infer that the total number of instances of Yes is 80% of the total population and No is 20% (ideal class imbalance ratio), then the ideal way to data for Training and Testing is using Stratified Shuffled Split.

This is an example of binary classification problem, the values and ratios can change w.r.t to the dataset. But the ratios between the classes are similar in Train and Testing samples which reduces the probability of a Machine learning modelling being biased



**Train set : Classes Ratio**

Viral : Bacterial : Malarial  
41.83 : 29.85 : 29.71

**Test set : Classes Ratio**

Viral : Bacterial : Malarial  
40.32 : 29.83 : 29.83

**Inference :** Stratified split enables in maintaining the ratio of classes in both training and test sets similar. This is done in order to avoid overfitting and being biased towards a class during training.

```
1 Btemp', 'Continous', 'Night Rise', 'Chills', 'Weakness & Fatigue', 'Joint Pain', 'Chest Pain', 'Stiffness', 'Running
1 y_train = train['Fever_Class']
1 x_test = test[['Btemp', 'Continous', 'Night Rise', 'Chills', 'Weakness & Fatigue', 'Joint Pain', 'Chest Pain', 'Stif
1 y_test = test['Fever_Class']
1 = df[['Btemp', 'Continous', 'Night Rise', 'Chills', 'Weakness & Fatigue', 'Joint Pain', 'Chest Pain', 'Stiffness', 'F
2 = df['Fever_Class']
```

**Figure 3.9: Stratified Shuffle Split inference**

towards predicting a particular class.

**Table 3.2: Sampling Comparison**

Sampling Method	Ratio of Yes-No between Test and Train (approx.)
Stratified Sampling	<b>Train :</b> Yes: 75% No: 25% <b>Test:</b> Yes: 75% No: 25%
Random Sampling	<b>Train :</b> Yes: 75% No: 25% <b>Test:</b> Yes: 95% No: 5%

### 3.9 Machine Learning Models and Validation Methods

The objective is to predict the fever category using Machine Learning algorithm. The models used must be able to work like the intuition of the doctors diagnosing the patients.

The following methodology was used to do data modelling and validate machine learning algorithms:

1. Split the data using Stratified Shuffled split to maintain the ratio of the classes in the train and test samples.
2. Define a clear set evaluation metrics which are suitable for multi-class problems.
3. Machine Learning Models to be used: Random Forest, Adaptive Boosting and Bayesian Classifier with Laplace smoothing.

4. Validate the results using K-Fold Cross Validation with different metrics to have better insights of the prediction.
5. Select the best Performing Model get it product ready for deployment.

### 3.9.1 Machine Learning Models:

#### Random Forest:

Random Forest is a supervised algorithm which builds weak learners aggregates them to form a strong learner. This method of learning is called Ensemble learning. It uses a aggregating known as “Bootstrap Aggregating or Bagging”.

#### Bagging or Bootstrap Aggregating:

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set  $X = x_1, \dots, x_n$  with responses  $Y = y_1, \dots, y_n$ , bagging repeatedly ( $B$  times) selects a random sample with replacement of the training set and fits trees to these samples: For  $b = 1, \dots, B$ : Sample, with replacement,  $n$  training examples from  $X, Y$ ; call these  $X_b, Y_b$ . Train a classification or regression tree  $f_b$  on  $X_b, Y_b$ .

After training, predictions for unseen samples  $x'$  can be made by averaging the predictions from all the individual regression trees on  $x'$  or by taking the majority vote in the case of classification trees.

$$\hat{f} = \frac{1}{B} \sum_{b=1}^B f_b(x')$$
(3.1)

#### Node Splitting Criterion- Information Gain

Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes.

For each node of the tree, the information value “represents the expected amount of information that would be needed to specify whether a new instance should be classified

yes or no, given that the example reached that node.”

$$\overbrace{IG(T, a)}^{\text{Information Gain}} = \overbrace{H(T)}^{\text{Entropy(parent)}} - \overbrace{H(T|a)}^{\text{Weighted Sum of Entropy(Children)}} \quad (3.2)$$

Where Entropy is given as:

$$H(T) = I_E(p_1, p_2, \dots, p_J) = - \sum_{i=1}^J p_i \log_2 p_i \quad (3.3)$$

where  $p_1, p_2, \dots$  are fractions that add up to 1 and represent the percentage of each class present in the child node that results from a split in the tree.

The Random Forest functions in two stages:

### **Stage 1: Classifier Building**

The algorithm randomly selects “N” Features from total features available in the dataset Among the the selected features, the node split is done (Information gain: Entropy split) Multiple trees are created till maximum split is reached. The forest is built by aggregating all the built trees.

### **Stage 2: Prediction using Majority Voting**

Each tree predicts the outcome class using the test feature vectors. The algorithm uses votes for each predicted target and the majority voted target is the final prediction that is returned by the algorithm.

### Advantages of using Random Forest:

- Avoids over-fitting in classification problems in machine learning due to the presence of optimal Decision trees
- It also calculates the Feature importance of each feature during training
- Can be scalable and deployed easily

### Adaptive Boosting Classifier:

Adaptive Boosting Classifier (AdaBoost) is a supervised machine learning algorithm. It uses boosting to tweak weak learners to combat class imbalance problem.

Boosting first initializes equal weights to all the learners at start. During the training it increases the weight of learners which predicts incorrect results. This process repeats when a threshold of accuracy is reached.

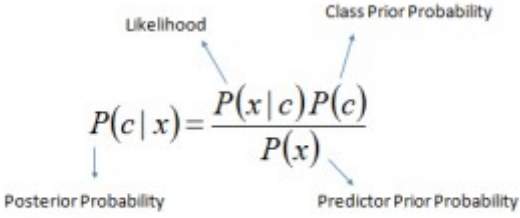
### Advantages:

- It can achieve accurate predictions with datasets having class imbalance problems
- The boosting algorithm can be mapped with any Machine Learning algorithm for training Weak Learners and Strong Learners

### Bernoulli's Naive Bayes (NB) classifier with Laplace Smoothing:

This is a supervised machine learning algorithm which is built over Bayesian probability. Bayes theorem revolves around posterior probability.

Bayes theorem provides a way of calculating posterior probability  $P(c|x)$  from  $P(c)$ ,  $P(x)$  and  $P(x|c)$ . Look at the equation below:


$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$
$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c) \quad (3.4)$$

- $P(c|x)$  is the posterior probability of class (c, target) given predictor (x, attributes).
- $P(c)$  is the prior probability of class.
- $P(x|c)$  is the likelihood which is the probability of predictor given class.
- $P(x)$  is the prior probability of predictor.

#### **Advantages:**

- Suitable for large dataset
- Suitable for multi class scenarios
- Performs well if most of the features are categorical variables

#### **Disadvantages:**

- The algorithm assumes all the given feature vectors are independent which is not possible in real-life. There is always dependencies.
- If a category is not in the test data or a pattern or observation the model will assign zero to the posterior probability calculated. The model will be unable to make a prediction. This phenomenon is known as “Zero Frequency error”.

Bernoulli NB implements algorithms for the data whose distributions of the feature vectors are according to the multivariate Bernoulli Distributions i.e. each feature must be represented in binary.

The decision rule for Bernoulli naive Bayes is based on:

$$P(x_i | y) = P(i | y)x_i + (1 - P(i | y))(1 - x_i) \quad (3.5)$$

*Note: The dataset has features in the form of Binary values. The number of categorical variables in the dataset are 14 out of 15.*

#### **Zero Frequency and Laplace (Additive) Smoothing:**

- There are 14 categorical variables which are represent themselves in the form 1 and zeroes indicating their occurrence (Running nose, Dry Cough etc...)
- The 14! (14 Factorial) possible combinations possible, out which a minimum of 8! Combinations will be valid in accordance to the field of medicine.

- The total unique permutations available in the dataset is 236 which is very less than 8 factorial.
- If we train the Naive Bayes Classifier which treats each feature individually, then the model will be able to have a non-zero probability of the incoming cases which were already in the training data. i.e. 236 unique cases.
- The posterior probability of new permutation with the fever class will be zero.
- This scenario is called the Zero Frequency.

In order to handle this scenario we use Laplace Smoothing also called as Additive Smoothing. Laplace Smoothing is a technique used to smooth categorical data. Given an observation  $x = (x_1, \dots, x_d)$  from a multinomial distribution with  $N$  trials, a “smoothed” version of the data gives the estimator.

In simple terms, if the value of the smoothing parameter has been set to 1, then it will assume each permutation of the categorical features have occurred at least once in the dataset which is going to avoid a zero probability.

Table 3.3: **Summary of Models**

Model Class	Machine Learning Model	Intuition
Ensemble	Random Forest	To reduce the bias and overfitting of the model. Classify fever based upon Information gain.
Ensemble	Adaptive Boosting Classifier	Avoid Class imbalance problem
Bayesian	Bernoulli's NB Classifier with Laplace Smoothing	Posterior Probability which mimics the doctors intuition.

### 3.9.2 Validation Metrics:

Validation is the most important setup in data modelling to measure the performance of the model. It is also important to look in to different metrics to gain better insights of the performance of the model. The following metrics where looked into to understand how the above mentioned models performed in the test dataset:

n=165		Predicted: NO	Predicted: YES	
Actual: NO		TN = 50	FP = 10	60
Actual: YES		FN = 5	TP = 100	105
		55	110	

**Figure 3.10:** Example: Binary Classification Scenario

### Confusion Matrix:

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.

**True positives (TP):** These are cases in which the model predicted yes and the actual value is yes.

**True negatives (TN):** The model predicted No and the actual value is No.

**False positives (FP):** The model predicted yes but they actual value is No.

**False negatives (FN):** The model predicted no, but the actually value is Yes.

### Accuracy:

Overall, how often is the classifier correct?

$(TP+TN)/total = (100+50)/165 = 0.91$  Accuracy gives the overall performance of the algorithm, which will act as the base of comparison between multiple models.



**Precision:**

When it predicts yes, how often is it correct?

$TP/predicted\ yes = 100/110 = 0.91$  This metric gives us the percentage of True positives from the Total number of classifications made as positive. The less Precision, the more False positives and in the field of health care the threshold of False positives must be less than 10% in accordance with the Risk factors.

**Recall:**

Also known as **True Positive Rate / Sensitivity** . When it's actually yes, how often does it predict yes?

$TP/actual\ yes = 100/105 = 0.95$  This metric gives us the insight of the how much percentage of the total positive classes is predicted by the model.

The results of the above mentioned algorithms are explained in detail the Results and Analysis Section.

### 3.10 Model Deployment Methods

- After the identification of the best machine learning model which is production ready both in terms of deployment and Doctor's intuition, the model will be trained in the whole dataset.
- The trained model can be store as a serialized file using the Python Package "Pickle"
- The Serialized object file stored in the form of ".sav" extension can be loaded into the web application (Views Layer ) and can be used for prediction. 3.11

```

import pandas as pd
import numpy as np
import pickle
from sklearn.naive_bayes import BernoulliNB

df = pd.read_csv("Prep_data.csv")
X = df[['Btemp', 'Continous', 'Night Rise', 'Chills', 'Weakness & Fatigue', 'Join
Y = df['Fever_Class']

#training the model

bnb = BernoulliNB(alpha=2.0) #setting the laplacing smoothing to 2 instances
bnb.fit(X,Y)

filename = 'trainedmodel.sav'

#saving the object file
pickle.dump(bnb,open(filename,'wb'))

```

**Figure 3.11:** Saving Model as Pickle file

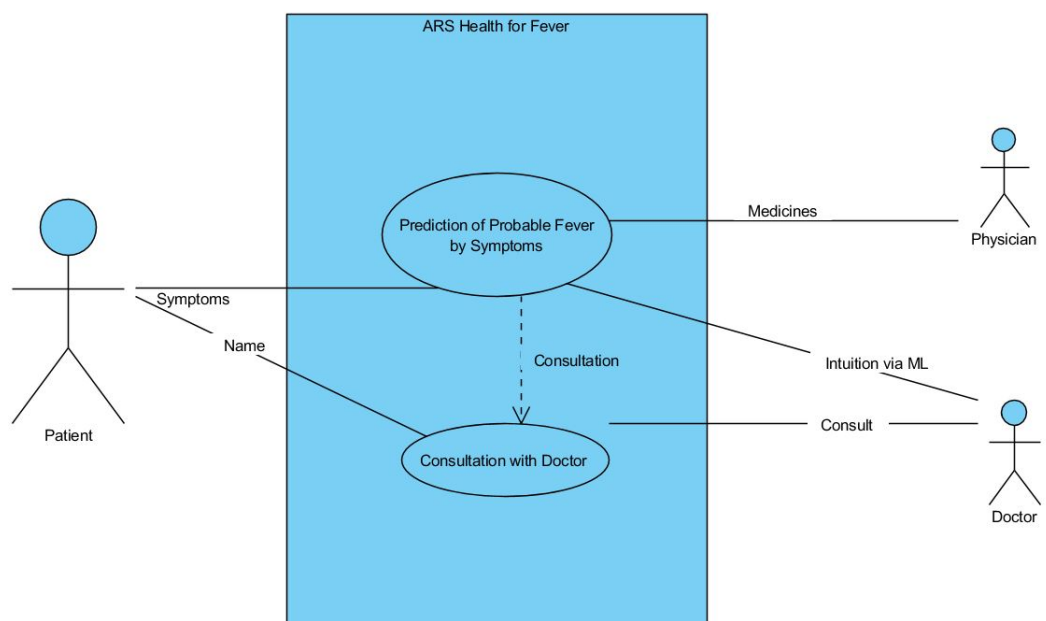
# CHAPTER 4

## SYSTEM DESIGN

This Chapter discusses about the general architecture of the Automated Remedy System for fever using Machine Learning and its components.

### 4.1 USE-CASE DIAGRAM

Use Case Diagram relates the application to the needs of the end-users. In our application, we are connecting the actors( Patients, Doctors and Physicians) to the need of consultation. (diagram 4.1)



**Figure 4.1:** Use Case Diagram

## 4.2 ARCHITECTURE DIAGRAM

This explains the core structure of the application and how it interacts with the various components of the Application such as Database and web interface. Here, the major components are Web Views, Database and Machine Learning Module. All of it is brought together using Django Framework in Python. The User(Patient) can perform the actions of Prediction, Book Appointments and Review them. Predefined Prescription Mappings has been created based on individual symptoms or combination of inter-related symptoms and stored in the MySQL Database. Similarly the Appointments are stored in the database for review by the Doctor during consultation. (diagram 4.2)

### AUTOMATIC REMEDY SYSTEM FOR FEVER USING MACHINE LEARNING

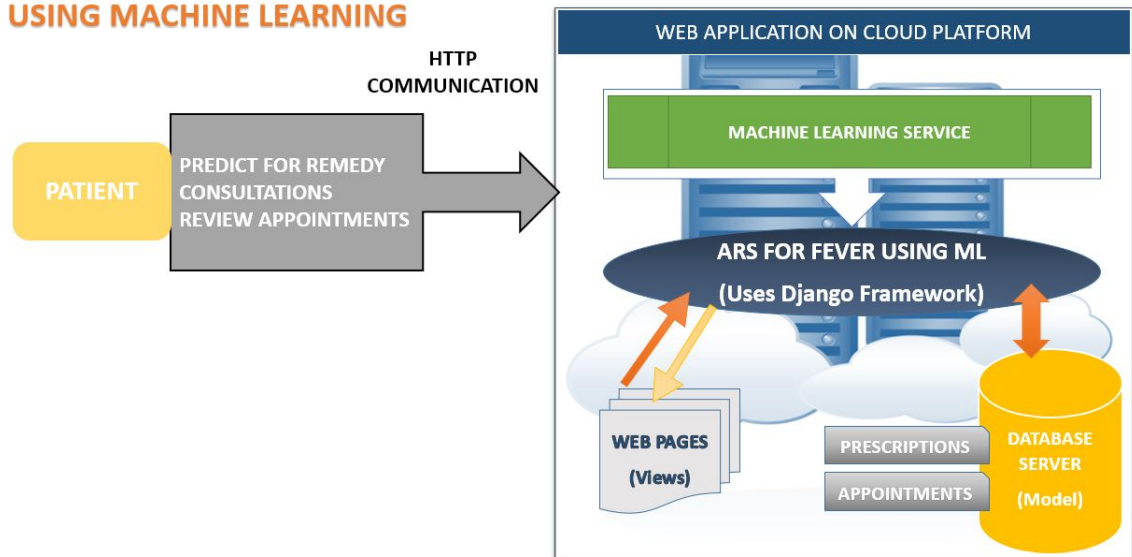


Figure 4.2: Component Diagram

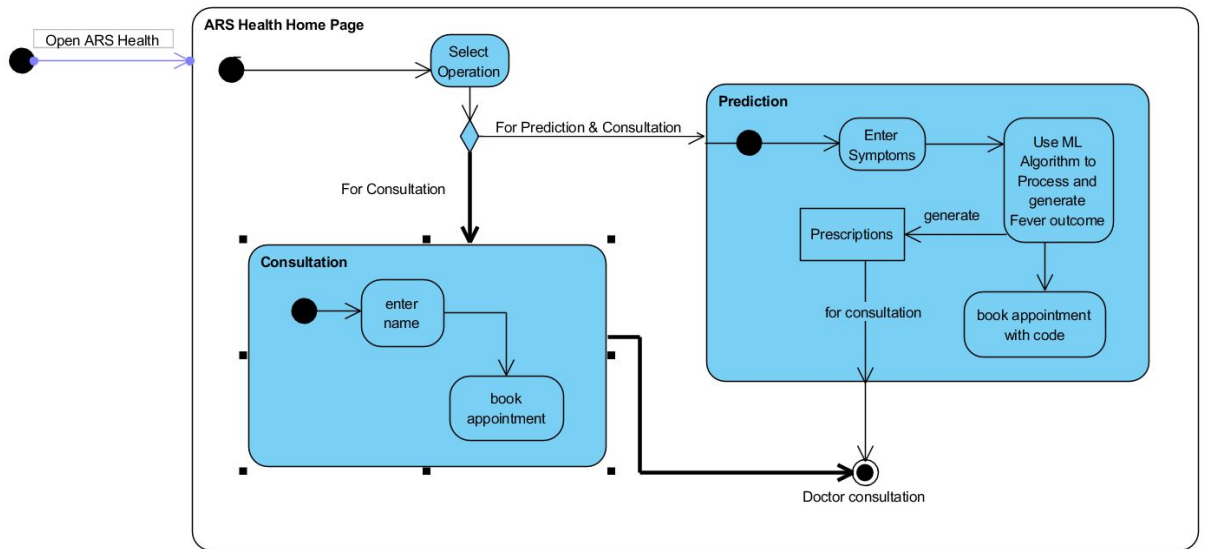
## 4.3 DATA FLOW DIAGRAM

Data Flow Diagram describes how the data is transformed by various components. The data is defined and moved around the system. Here the operations comprise of Prediction and Consultation.

Prediction will result in the type of fever and generate a list of Prescriptions.

Consultation will fix an appointment with the doctor in a clinic closest to the patient.

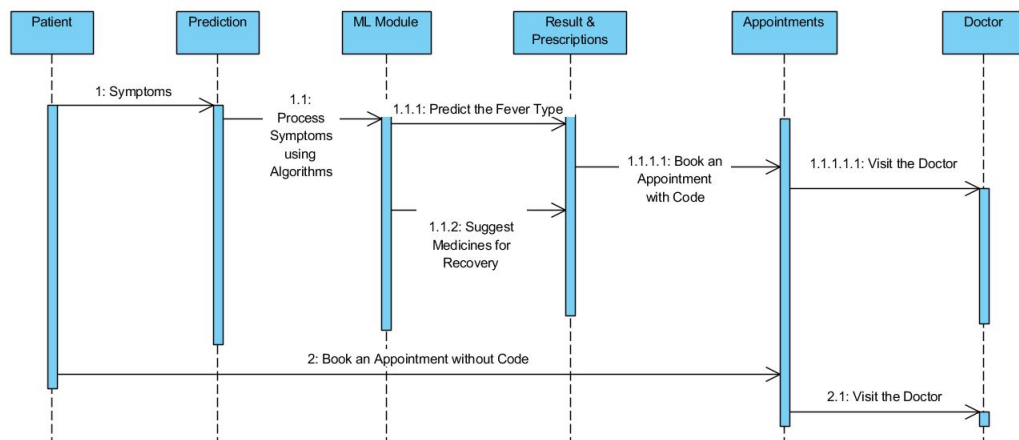
(diagram 4.3)



**Figure 4.3:** Data Flow Diagram

## 4.4 SEQUENCE DIAGRAM

A Sequence Diagram explains the sequence of actions a user(patient) may take in order to achieve the goal. The Patient here can perform two basic operations which are Prediction and Consultation, as explained in the previous section. Here we can see how Patient can perform the operations in a detailed manner and how it results into the output. (diagram 4.4)



**Figure 4.4:** Sequence Diagram

# CHAPTER 5

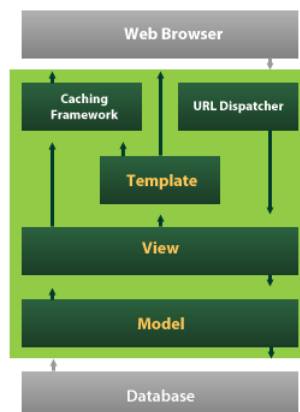
## CODING AND TESTING

### 5.1 Overview

This chapter describes the Coding Architecture followed to design and implement the Web Application as Automated Remedy System for Fever. The application was developed on Ubuntu 16.04 using Django Framework. The Ubuntu OS was launched as a virtual machine acting like a stand-alone server. It was accessible by the Host Machine using the URL(ubuntuserver.it).

### 5.2 Django Framework

Django Web Framework is based on Python, which is used to build Web Applications rapidly. The framework facilitates an architecture which is open for all. It is highly modular, such that Django Apps can even be plugged-in from one project to another. That also removes the dependencies on the OS. IF we have Python installed, we can build and launch Django Apps in minutes.



**Figure 5.1:** Django Architecture

Django follows the Model-View-Controller (MVC) architecture. Models can be created which define the Database. They need to be bound with the Views to serve the

HTTP requests. Django also has a efficient Template Engine. In latest versions, we use Jinja for templating.

Templating reduces the time to design and create web-pages. It consolidates the redundant content in templates which then are filled with the dynamic content. The URL dispatcher of the framework makes sure we arrive at the right page using the view layer objects and RegEx.

## 5.3 Code

As the part of the questionnaire, We need to build a form for getting the User Data( i.e. Symptoms of the Patient). So using framework's form class we defined the our Form class for input. Here is a excerpt of the Code.

```
1 from django import forms
2
3 class InputParameters(forms.Form):
4     YES = 1
5     NO = 0
6
7     YES_NO_CHOICES = (
8         (NO, 'No'),
9         (YES, 'Yes'),
10    )
11    name = forms.CharField(label='Your name: ', max_length=100)
12    body_t = forms.DecimalField(label = "Enter your Body Temperature in Fahrenheit ", required = True)
13
14    allergy = forms.ChoiceField(label="Any type of Allergy or Rashes? ",
15                                widget=forms.Select,
16                                choices=YES_NO_CHOICES)
17    ToF = forms.ChoiceField(label='How are you getting Fever? ',
18                            widget=forms.Select,
19                            choices = ((0,'Continuous'),
20                                     (1,'Chills'),
21                                     (2,'Night Rise/Irregular Rise'),)
22    )
23    BP = forms.ChoiceField(label = 'Do you have any Pain in your body? ',
24                            widget = forms.Select,
25                            choices = ((0,'Weakness & Fatigue'),
26                                     (1,'Joint Pain & Lower Back Pain'),
27                                     (2,'Chest-Pain'),
28                                     (3,'Stiffness & Swells'),)
29    )
30    vomit = forms.ChoiceField(label = "Are you experiencing any vomiting? ",
31                              widget = forms.Select,
32                              choices = YES_NO_CHOICES)
33
34    lo_mo = forms.ChoiceField(label = "Any episode of Loose Motion? ",
35                              widget = forms.Select,
36                              choices = YES_NO_CHOICES)
37
38    cold = forms.ChoiceField(label="Do you have any Cold? ",
39                              widget=forms.Select,
40                              choices = ((0,'No Cold'),
41                                       (1,'Running Nose'),
42                                       (2,'Nose Blockage'),)
43    )
44    cough = forms.ChoiceField(label = 'Are you Coughing? What is it like? ',
45                              widget = forms.Select,
46                              choices = ((0,'No Cough'),
47                                       (1,'Dry Cough'),
48                                       (2,'Sore Throat'),
49                                       (3,'Cough with Sputtum'),)
50    )
51    ui = forms.ChoiceField(label = "Are you experiencing any Urine Infection/Burning Sensation during Urinating? ",
52                            widget = forms.Select,
```

**Figure 5.2:** Django Forms for Symptoms Input

We have a Views.py module in python which defines functions which render the appropriate View layer objects( web pages). We use this to handle requests which are dispatched by the URL dispatcher.

Here we are handling the View layer code for the Prediction App.

```
1 from django.shortcuts import render
2 from django.http import HttpResponse, Http404
3 from django import forms
4 from .forms import InputParameters
5 from .MLmodule import Machine_learning
6 from .prescrip import Pres, Med
7 from django.utils.six.moves import cPickle as pickle
8 import datetime
9 from .models import Prescription
10 from .models import Appointment
11
12
13 def predict(request):
14     return render(request, "prediction/pred_home.html")
15
16 def input(request):
17     form = InputParameters()
18     return render(request, "prediction/pred_input.html", {'form': form})
19
20 def output(request):
21     form = InputParameters(request.POST)
22     if form.is_valid():
23         allergy = str(form.cleaned_data.get('allergy'))
24         temp = str(form.cleaned_data.get('body_t'))
25         tof = str(form.cleaned_data.get('ToF'))
26         bpain = str(form.cleaned_data.get('BP'))
27         vomit = str(form.cleaned_data.get('vomit'))
28         lm = str(form.cleaned_data.get('lo_mo'))
29         cough = str(form.cleaned_data.get('cough'))
30         ui = str(form.cleaned_data.get('ui'))
31         cold = str(form.cleaned_data.get('cold'))
32         name = str(form.cleaned_data.get('name'))
33
34         ml = Machine_learning()
35         ml.setValues(temp, tof, bpain, vomit, lm, cough, cold, ui)
36         answer = ml.predictValues()
37
38         med = []
39         if int(allergy) != 1:
40             pr = Pres()
41             pr.setValues(temp, tof, bpain, vomit, lm, cough, cold, ui)
42             med = pr.gen_pres()
43         else:
44             answer = "ALLERGY"
45             med.append(Med("Allergy", "Cannot Prescribe Medicine", "Make Appointment and consult Doctor immediately!"))
46
```

Figure 5.3: Views in Prediction (a)

```
47 doctor_clinic = "Dr.(Prof).V. Padma, Internal Medicine Specialist, Sree Balaji Medical College and Hospital, #7, Works Road, Chrompet, Chennai, Tamil Nadu."
48 #print(answer)
49 #pr = Prescription(pres=pickle.dumps(med))
50 #pr.save()
51
52 app = Appointment(p_name = name, disease = answer, pres = pickle.dumps(med), doctor = doctor_clinic)
53
54 app.save()
55 #appid = Appointment.objects.latest('id')
56 #request.session['id1'] = appid
57
58
59 return render(request, "prediction/pred_output.html", {'result': answer, 'prescriptions': med})
60 else:
61     return HttpResponse("Insufficient Data, go Back and enter your Symptoms well")
62
```

Figure 5.4: Views in Prediction (b)

URLs are handled by a dispatcher module. They reference the Views.py file and call the functions which need to be referenced to render appropriate web page. The url patterns is a list of url objects which maps patterns using regex code.



```

1 from django.conf.urls import url, include
2 from . import views
3 urlpatterns = [
4     url(r'^$', views.predict, name='prediction'),
5     url(r'^input$', views.input, name='prediction'),
6     url(r'^output$', views.output, name='prediction'),
7 ]
8

```

**Figure 5.5:** URLs in Prediction

To store content in database, we configure the models.py File. This defines classes which can create the appropriate records in the Database. Database can be configured using settings.py file in Django Admin App. We use the following Models.py file for Prediction data.

```

models.py
1 from django.db import models
2
3 class Med(models.Model):
4     symp = models.CharField(max_length=140)
5     name = models.CharField(max_length=140)
6     dose = models.CharField(max_length=140)
7
8 class Prescription(models.Model):
9     date = models.DateField(auto_now_add=True)
10    pres = models.BinaryField()
11
12 class Appointment(models.Model):
13    date = models.DateField(auto_now_add=True)
14    p_name = models.CharField(max_length=140)
15    disease = models.CharField(max_length=100)
16    pres = models.BinaryField()
17    doctor = models.CharField(max_length=500)
18

```

**Figure 5.6:** Models for Prediction

Here we have defined classes for Medicine Table(Class: Med), Prescriptions(Class: Prescription), Appointments(Class: Appointment).

We can have custom python modules which render us help with some algorithms to handle business logic. They can be integrated easily with other django modules as the base language is Python. We have created certain modules like for determining Prescriptions as well as for processing the Machine Learning Model from a Pickle object.

```

1 import numpy as np
2 import pickle
3
4 class Machine_learning:
5
6     filename = 'prediction/trainedmodel.sav'
7     loaded_model = pickle.load(open(filename, 'rb'))
8
9     def __init__(self):
10         self.btemp = 0.0
11         self.continuous = 0
12         self.Night_Rise = 0
13         self.Chills = 0
14         self.wf = 0 # weakness and Fatigue
15         self.jp = 0 #joint pain
16         self.cp = 0 #chest Pain
17         self.stiffness = 0
18         self.RN = 0 #Runningnose
19         self.NB = 0 #Noseblockage
20         self.DC = 0 #dry Cough
21         self.ST = 0 #sorethroat
22         self.CSP = 0 #coughwithsputum
23         self.Vomitting = 0
24         self.Lmotion = 0
25         self.UI = 0
26
27     def setValues(self,Temp,TOF,bpain,VOM,LM,COU,COLD,Ui):
28         TOF = int(TOF)
29         bpain = int(bpain)
30         COU = int(COU)
31         COLD = int(COLD)
32         # setting the labels
33         #-----setting Type of fever
34         self.btemp = float(Temp)
35         if TOF == 0:
36             self.continuous = 1
37             self.Night_Rise = 0
38             self.Chills = 0
39
40         elif TOF == 1:
41             self.continuous = 0
42             self.Night_Rise = 0
43             self.Chills = 1

```

**Figure 5.7:** ML Module using symptoms

ML Module takes symptoms as input, cleans them and feeds to predict method. This returns the type of fever.

```

50     #For Loose Motion
51     if self.Lmotion == 1:
52         p.append(Med("Loose Motion (Light)","Loperamide 2 mg","1 - 0 - 1"))
53         p.append(Med("Loose Motion (Heavy)","Doxycycline 100mg","1 - 0 - 1"))
54     #For Cold
55     if self.Cold == 1:
56         p.append(Med("Running Nose","Cetirizine","1 - 1 - 1"))
57     elif self.Cold == 2:
58         p.append(Med("Nose Blockage","Karvol Plus (Steam Inhalation)","1 - 1 - 1"))
59         p.append(Med("Nose Blockage","Sinarest","1 - 1 - 1"))
60     #For Cough
61     if self.Cough == 1:
62         p.append(Med("Dry Cough","Levocetirizine 10 mg","1 - 0 - 1"))
63     elif self.Cough == 2:
64         p.append(Med("Sore Throat","Betadine Gargle Solution","1 - 1 - 1"))
65     elif self.Cough == 3:
66         p.append(Med("Cough with Sputum","Ascoril Syrup","1 - 0 - 1"))
67
68     #For Urine Infection
69     if self.ui == 1:
70         p.append(Med("Urine Infection","Drink Alcosol Solution 10ml with 200ml water ","1 - 0 - 1"))
71         p.append(Med("Urine Infection","May take any previously prescribed medicines ","N/A"))
72
73     return p
74

```

**Figure 5.8:** Prescriptions Module (a)

Prescriptions takes symptoms as input and returns a list of Med objects(from model) to Views.

For Appointments App, we have designed a similar interface which takes hold of Prescription objects and returns you with appointment ID for consultation with the doctor. As of now, we have a static doctor information for every patient.

```

1 from django.shortcuts import render
2 from django.http import HttpResponse
3 from prediction.models import Appointment
4 from prediction.prescrip import Pres, Med
5 from django.utils.six.moves import cPickle as pickle
6 from .forms import InputParameters
7 import datetime
8
9 def appointment(request):
10
11     app = Appointment.objects.last()
12     #session-cache-Test
13     #print(name)
14     print(app.doctor)
15     pr = pickle.loads(app.pres)
16     return render(request, "appointments/app.html", {'doctor': str(app.doctor), 'appid': str(app.id)+str(app.p_name), 'prescriptions': pr})
17
18 def getName(request):
19     form = InputParameters()
20     return render(request, 'appointments/name_input.html', {'form': form})
21
22 def output(request):
23     form = InputParameters(request.POST)
24     if form.is_valid():
25         name= str(form.cleaned_data.get('name'))
26         n = str(name) + '-EXT'
27         doc = doctor_clinic = "Dr.(Prof).V. Padma, Internal Medicine Specialist, Sree Balaji Medical College and Hospital,#7, Works Road, Chrompet, Chennai, Tamil Nadu."
28         return render(request, 'appointments/output.html', {'name':n, 'doctor': doc})
29

```

**Figure 5.9:** Appointments Views File, delegating appointments

Similar to URL dispatcher of Prediction app, we have Appointments Urls.py. This is to facilitate the option to Patient to got for prediction or direct consultation with the doctor.

```

1 from django.conf.urls import url, include
2 from . import views
3 urlpatterns = [
4     url(r'^$', views.appointment, name='appointment'),
5     url(r'^ext$', views.getName, name='getName'),
6     url(r'^output$', views.output, name='output'),
7
8 ]
9

```

**Figure 5.10:** Appointment URLs

The results are stored in the database in pickle format(Serialized). This prevents data to be visible explicitly.

```

q)q)q)q)q)q) doseqX 1 - 1 - 1qX nameqX Dolo 650mg (Paracetamol)qX sympX Continuous Feverq ubh)q)
}q)
(h)X 1 - 0 - 1q hX# Zerodol P (Pain Killer) with Pan 40q
X Chest Painqubh)q)q)q)h)h)h)
Cetirizineq)X
Running Noseq)be.
Sree Balaji Medical College and Hospital,#7, Works Road, Chrompet, Chennai, Tamil Nadu. | Dr.(Prof).V. Padma, Internal Medicine Specialist,
| 9 | 2018-04-09 | Somnath | VIRAL | q)q)q) (cprediction.prescrip
Med
q)q)q)q)q)q) doseqX 1 - 1 - 1qX nameqX Dolo 650mg (Paracetamol)qX sympX Continuous Feverq ubh)q)q)
}q)
(h)X 1 - 0 - 1q hX# Zerodol P (Pain Killer) with Pan 40q
X Chest Painqubh)q)q)q)h)h)h)
Cetirizineq)X
Running Noseq)be.
Sree Balaji Medical College and Hospital,#7, Works Road, Chrompet, Chennai, Tamil Nadu. | Dr.(Prof).V. Padma, Internal Medicine Specialist,
| 10 | 2018-04-09 | Soms | MALARIAL | q)q)q) (cprediction.prescrip
Med
q)q)q)q)q)q) doseqX 1 - 0 - 1qX nameqX Crocin 500mg (Paracetamol)qX sympX General Body Painq ubh)q)q)
}q)
(h)X Eat Wellq
X Weakness & Fatigue(mild)qube.
.(Prof).V. Padma, Internal Medicine Specialist, Sree Balaji Medical College and Hospital,#7, Works Road, Chrompet, Chennai, Tamil Nadu. | Dr
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
6 rows in set (0.01 sec)

```

**Figure 5.11:** Part of Appointment Data

If all works well, the server runs successfully and the application goes live.

```

root@server-network-vb:/home/server/web/ARSHealth# python3 manage.py runserver ubuntuserver.it:8000
Performing system checks...
System check identified no issues (0 silenced).
April 09, 2018 - 23:53:35
Django version 2.0.1, using settings 'ARSHealth.settings'
Starting development server at http://ubuntuserver.it:8000/
Quit the server with CONTROL-C.
create mode 100644 ARSHealth/prediction/migrations/_pycache_/0002_auto_20180409_0959.cpython-35.pyc
create mode 100644 ARSHealth/prediction/migrations/_pycache_/0003_auto_20180409_2136.cpython-35.pyc
create mode 100644 ARSHealth/prediction/migrations/_pycache_/0004_auto_20180409_2202.cpython-35.pyc
create mode 100644 ARSHealth/prediction/migrations/_pycache_/0005_delete_appointment.cpython-35.pyc
create mode 100644 ARSHealth/prediction/migrations/_pycache_/0006_appointment.cpython-35.pyc
create mode 100644 ARSHealth/prediction/migrations/_pycache_/0007_delete_appointment.cpython-35.pyc
create mode 100644 ARSHealth/prediction/migrations/_pycache_/0008_appointment.cpython-35.pyc
root@server-network-vb:/home/server/web/WebMLProj-Automated-Remedy-System# git commit -m "With Appointments Module"
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
nothing to commit, working directory clean
root@server-network-vb:/home/server/web/WebMLProj-Automated-Remedy-System# git commit -m "With Prescriptions Integrated to

```

**Figure 5.12:** Django Server Running successfully

## 5.4 Testing

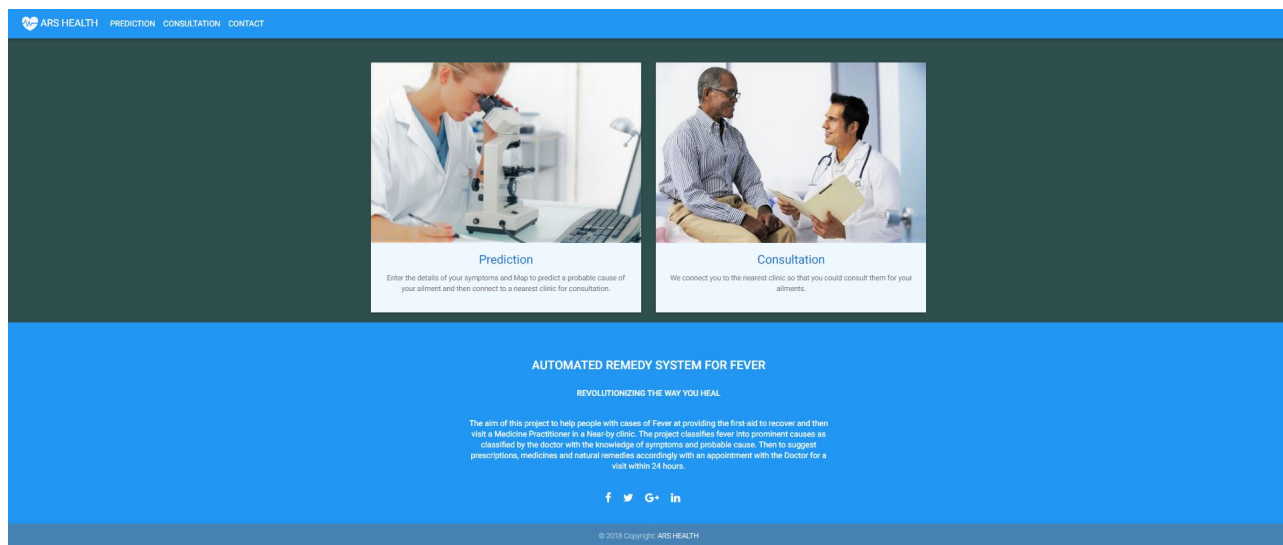
Testing is an integral part of any development activity. Testing exposes any fault in the system and helps to continuously improve the performance and security of the application. Unit testing performed in each and every stage and any unexpected outputs are corrected then and there to develop a stable system.

### 5.4.1 Home-Page

Launching the web-app on the browser we get to see the homepage. The application shows the two basic functionalities.

Prediction will enable user to launch the Fever Class Predicting app.

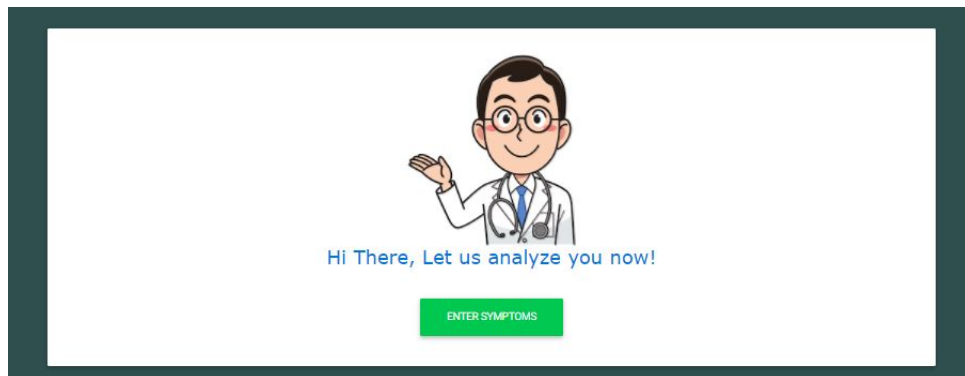
Consultation will enable user to simply book an Appointment with a Doctor.



**Figure 5.13: ARS Health Home**

## 5.4.2 Testing Prediction

Clicking Prediction will ask the user to confirm and then launch a form to fill. The form includes various parameters like Body Temperature, Allergic indications, Type of Fever, etc. which will help the Machine learning module to determine the class of the Fever.

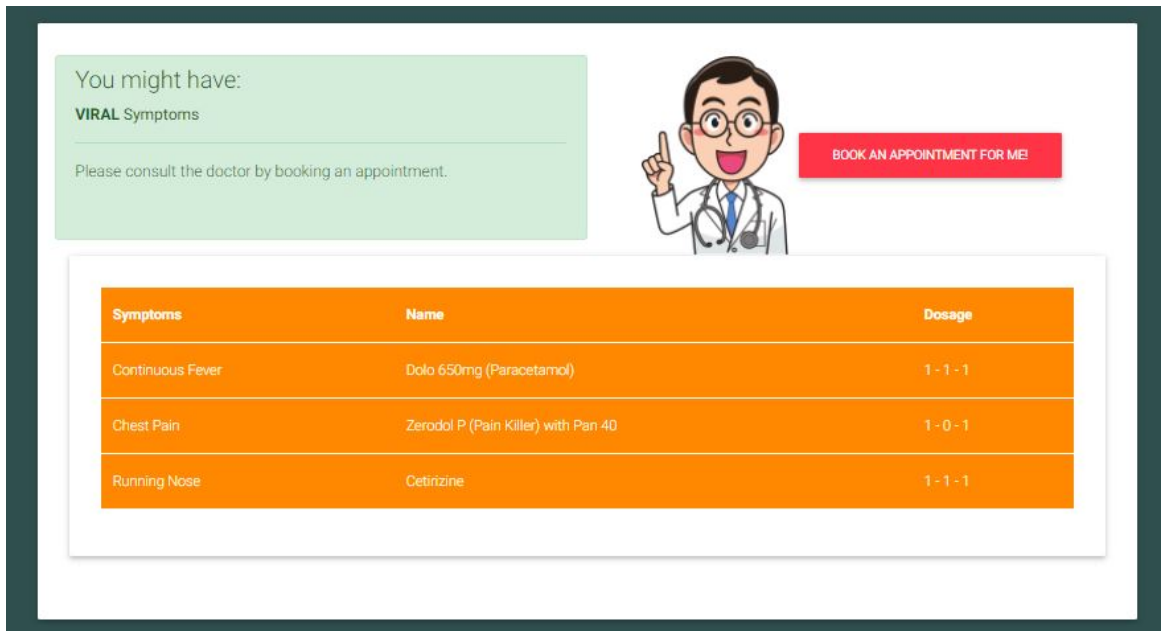


**Figure 5.14:** On Clicking Prediction

A screenshot of a web application form titled "Please answer these Queries...". The form contains several input fields and dropdown menus. The first field is "Your name: :", with the text "Somnath" entered. The second field is "Enter your Body Temperature in Fahrenheit :", with the value "102.3" entered. The third field is "Any type of Allergy or Rashes? :", with a dropdown menu showing "Yes". The fourth field is "How are you getting Fever? :", with a dropdown menu showing "Continuous". The fifth field is "Do you have any Pain in your body? :", with a dropdown menu showing "Chest-Pain". The sixth field is "Are you experiencing any vomiting? :", with a dropdown menu showing "No". The seventh field is "Any episode of Loose Motion? :", with a dropdown menu showing "No". The eighth field is "Do you have any Cold? :", with a dropdown menu showing "Running Nose". The ninth field is "Are you Coughing? What is it like? :", with a dropdown menu showing "No Cough". The tenth field is "Are you experiencing any Urine Infection/Burning Sensation during Urinating? :", with a dropdown menu showing "No". At the bottom center of the form is a green rectangular button with the text "PREDICT MY CONDITION!" in white, uppercase letters.

**Figure 5.15:** Fill the Details


Non-Allergic conditions may get user the prescription list like this. Here the symptoms have successfully predicted the class to be of **Viral Fever**. The appropriate medications are suggested.



You might have:

**VIRAL** Symptoms

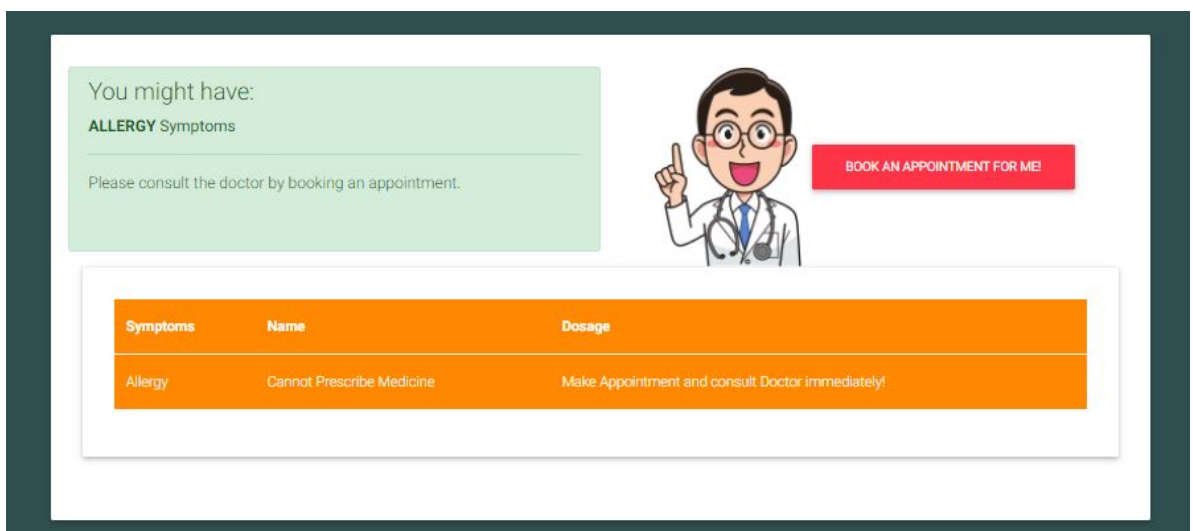
Please consult the doctor by booking an appointment.

 [BOOK AN APPOINTMENT FOR ME!](#)

Symptoms	Name	Dosage
Continuous Fever	Dolo 650mg (Paracetamol)	1 - 1 - 1
Chest Pain	Zerodol P (Pain Killer) with Pan 40	1 - 0 - 1
Running Nose	Cetirizine	1 - 1 - 1

**Figure 5.16:** Non Allergic Symptoms


Having Allergic symptoms will require user to consult doctor immediately. It does not prescribe any medicines as it may cause harm to the user.



You might have:

**ALLERGY** Symptoms

Please consult the doctor by booking an appointment.

 [BOOK AN APPOINTMENT FOR ME!](#)

Symptoms	Name	Dosage
Allergy	Cannot Prescribe Medicine	Make Appointment and consult Doctor immediately!

**Figure 5.17:** Results from symptoms



### 5.4.3 Testing Appointments

After prediction we proceed to Appointments app. This happens after the predictions, where the user clicks on “ Book an Appointment for Me ” button.

On successfully booking an appointment, the user can see this screen.

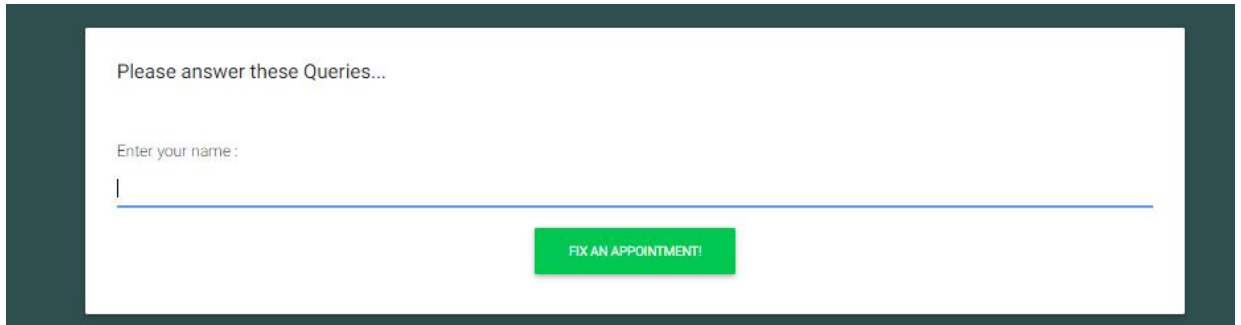


**Figure 5.18:** Consulting after predictions

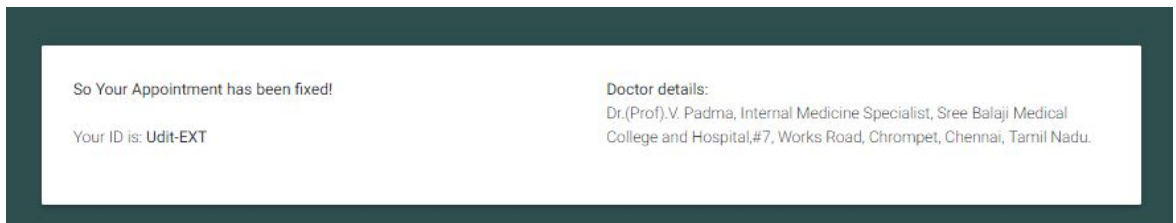
It auto-generates the appointment ID for the patient. It also extracts the Prescription from the database successfully and displays them in the Appointments app. The prescriptions can help the doctor to evaluate the patient in a better way and administer a future treatment.



For stand-alone consultations, name is taken as input for processing the appointment ID. It also gives the location of the clinic to be visited for consultation. As we can see this won't be displaying the Prescriptions as no predictions were conducted. The doctor needs to evaluate the patient from the start.

A screenshot of a web form titled "Please answer these Queries...". It contains a text input field labeled "Enter your name :" with a blue underline. Below the input field is a green button with the text "FIX AN APPOINTMENT!". The form is set against a dark teal background.

**Figure 5.19:** Consultation form

A screenshot of a confirmation screen with a dark teal background. It contains two columns of text. The left column says "So Your Appointment has been fixed!" and "Your ID is: Udit-EXT". The right column is titled "Doctor details:" and lists "Dr.(Prof).V. Padma, Internal Medicine Specialist, Sree Balaji Medical College and Hospital,#7, Works Road, Chrompet, Chennai, Tamil Nadu."

**Figure 5.20:** Appointment Booked

Once the Appointment is booked, user can visit the assigned doctor within 24 hours of the appointment. This will help them to get further tests and get better care.

## 5.5 Test Cases:

Table 5.1: Test Cases

Test Case	Steps to execute the Test Case	Expected Result	Actual Result	Test Result (Met / Not-Met)
Fever Prediction	Enter your symptoms in the form then Click the button <b>“Predict my Condition”</b>	The Fever Class must be predicted by the back-end machine learning module and must be displayed in the front-end.	The Fever Class for the given symptoms of the patient is predicted and displayed.	Met
Prescription Generation	Enter your symptoms in the form then Click the button <b>“Predict my Condition”</b>	The prescription of medicines must be generated for each symptom and must be displayed	The prescription has been generated for the input of symptoms given by the patient.	Met
Appointment Booking	Click on the <b>“Book an Appointment”</b> button.	The appointment must be booked and displayed with unique code for the patient.	The appointment has been generated with Unique code for the Patient	Met

# CHAPTER 6

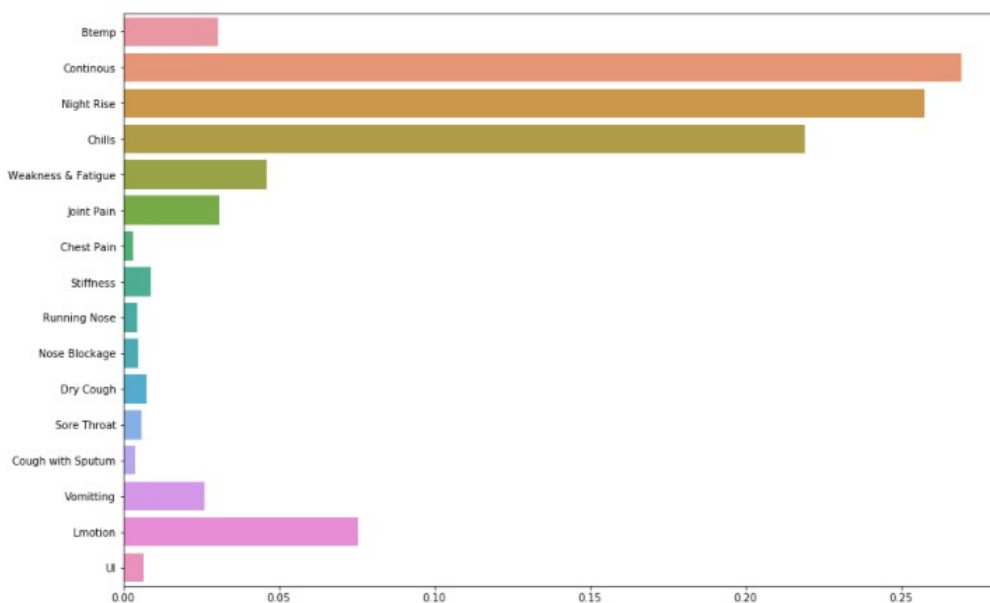
## RESULTS AND ANALYSIS

The following are the results of the models applied over the dataset and validated over the previously mentioned metrics.

### 6.1 RANDOM FOREST CLASSIFIER:

```
Classification Report :  
  
              precision    recall  f1-score   support  
  
   BACTERIAL      0.99      1.00      0.99        91  
   MALARIAL      0.96      1.00      0.98        91  
   VIRAL         1.00      0.96      0.98       123  
  
 avg / total      0.98      0.98      0.98       305  
  
Precision MICRO: 98.3606557377  
Recall MICRO: 98.3606557377  
F1 MICRO : 98.3606557377  
K-FOLD CROSS VALIDATION RESULTS : Number of Folds : 5
```

**Figure 6.1:** Random Forest Classification



**Figure 6.2:** Random Forest Classification - Feature Importance

## 6.2 ADAPTIVE BOOSTING CLASSIFIER:

```
Confusion Matrix
[[ 91   0   0]
 [  0  15  76]
 [  2   0 121]]

Classification Report :

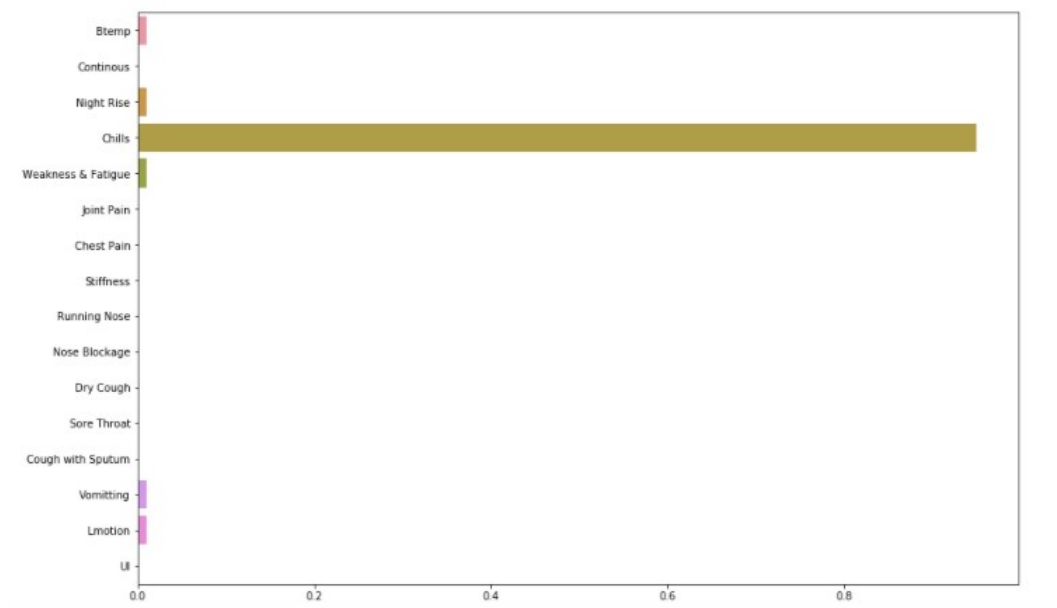
              precision    recall  f1-score   support

 BACTERIAL       0.98         1.00         0.99         91
  MALARIAL       1.00         0.16         0.28         91
   VIRAL         0.61         0.98         0.76        123
 avg / total       0.84         0.74         0.68        305

Precision MICRO:  74.4262295082
Recall MICRO:    74.4262295082
F1 MICRO :      74.4262295082

K-FOLD CROSS VALIDATION RESULTS : Number of Folds : 5
```

**Figure 6.3:** AdaBoost



**Figure 6.4:** AdaBoost Classification - Feature Importance

## 6.3 BERNOULLI'S NAIVE BAYES CLASSIFIER:

```
Confusion Matrix
[[ 86   0   5]
 [  0  88   3]
 [  0  11 112]]

Classification Report :

              precision    recall  f1-score   support

 BACTERIAL       1.00        0.95        0.97         91
  MALARIAL       0.89        0.97        0.93         91
   VIRAL         0.93        0.91        0.92        123

 avg / total       0.94        0.94        0.94        305

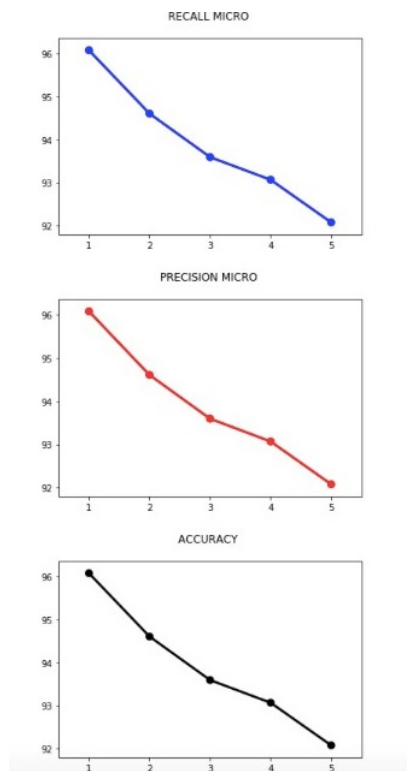
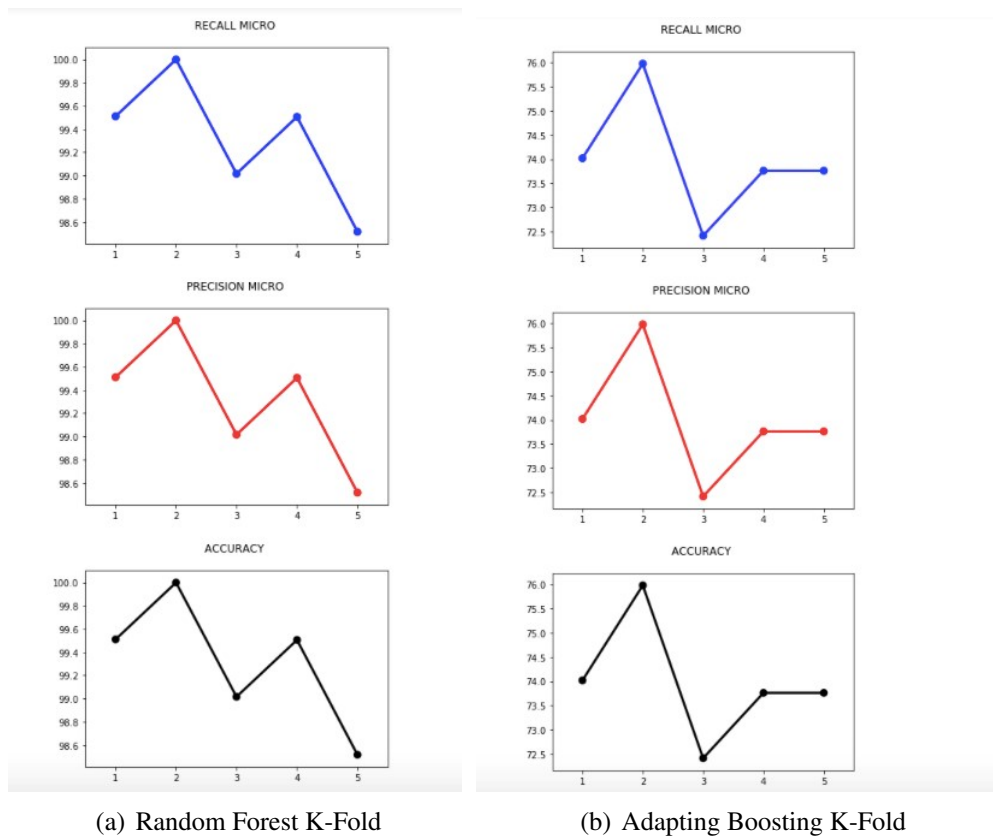
Precision MICRO:  93.7704918033
Recall MICRO:    93.7704918033
F1 MICRO :       93.7704918033

K-FOLD CROSS VALIDATION RESULTS : Number of Folds : 5
```

**Figure 6.5:** Naive Bayes Classification

## 6.4 RESULTS OF K-FOLD CROSS VALIDATION:

NUMBER OF FOLDS = 5



(c) Bernoulli's Naive Bayes K-Fold

**Figure 6.6: K-Fold Validations**

## 6.5 Inferences and Insights

The inferences mentioned below were drawn from the insights given by the validation metrics when applied to evaluate different aspects of performance of the machine learning models:

Adaptive boosting algorithm gave the lowest score in all the metrics in comparison with other two model. Further, the algorithm did not include all the features to attain the accuracy and precision. The model was predominantly biased towards “Chills” feature vector

Random forest gave the best score in all the metrics in comparison with Bernoulli’s Naive Bayes classifier. The precision and recall of the individual fever classes were also above 98%.

But, the algorithm is also highly biased over the feature vectors such as “Chills, Continuous, Night Rise, Lmotion”.

This is due to the fact that the number of unique cases in the dataset is very less and moreover the information gain on other combinations is comparatively less and close to zero. Due to this the algorithm will not be able to predict the fever category if a new permutation of the dataset is given as input i.e. the data points which are not present in the dataset. The Bernoulli’s Naive Bayes Classifier has the right balance between over-fitting and predicting the fever class of new cases whose symptoms are not listed in the training dataset. The solution is given by Laplace smoothing. The algorithms also depict an accuracy of 93% and the precision and recall of individual fever classes are above 92%.

Thus, Bernoulli’s Naive Bayes is a better performing algorithm. For the initial prototype and testing, the Automated Remedy System will use Bernoulli’s Naive Bayes.

The Bernoulli’s Naive Bayes classifier has the following benefits:

- Easily Scalable and Fast Training
- No complex hypothetical situations
- Mimics the intuition of the doctor of understanding the recent cases of the past

- Performs well with unseen cases
- Treats symptoms as independent features

Thus for the Prototype 1.0 of Automated Remedy System for Fever, Bernoulli's Naive Bayes will be production ready Machine learning model which be deployed in the Web Application.



## **CHAPTER 7**

### **CONCLUSION**

Health analytics is one the most important and useful applications of analytics. With new age technologies such as Artificial Intelligence, Machine learning and Distributed Application development, the industry is able to create highly efficient support systems in giving accurate diagnosis to the patients.

So accordingly we developed a user i.e. patient, personalized Health Consultation service that help identifies the type of fever from which they are suffering and also prescribes medicines/natural remedies to reduce the effects of the symptoms faced by the respective patients.

The system receives inputs of the symptoms from the users via a Front end. The list of symptoms is sent to the back-end framework developed in Django to handle multiple requests and sessions of the users. Once the request is received by the server at the Application Programming Interface (API) endpoint, it predicts the type of fever with the help of trained machine learning algorithms and returns back the response along with list of prescriptions of medicine. The whole process of the First health consultation is automated for the users to access at any part of the day or night.

The web-application is also capable of booking appointments with the nearby experienced doctors and also generates an appointment ID.

## **CHAPTER 8**

### **FUTURE ENHANCEMENTS**

The current machine learning model uses a Bernoulli's Naive Bayes classifier to identify the type of fever with a given set of symptoms. In order to combine information gain, rule based mappings and probability of recent cases in a locality or an area, a hybrid ensemble model which predicts the type of fever using Majority voting between Random Forest, Apriori algorithms and Naive Bayes classifier can be built and dynamic training within intervals of seasons can be automated.

The web-application runs on single server - multi-client model. The architecture of the model will be scaled up to run as a multi-server - multi-client model using distributed application frameworks such as Redis, where recent and most frequent data can be cached, making the application perform faster. The system will also be able to track past prescriptions of patients and maintain Electronic health records and appointments taken by the users.

# APPENDIX A

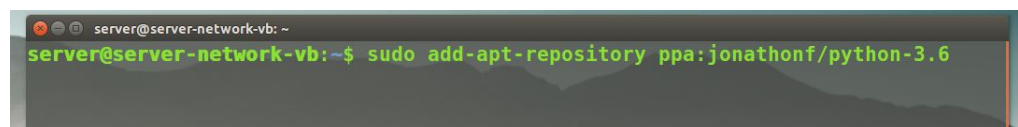
## DJANGO SET-UP

For the testing machine, we have used Ubuntu 16.04 LTS. Installation of Django requires Python as it is a package in it.

### A.1 Python Installation:

To install python we use:

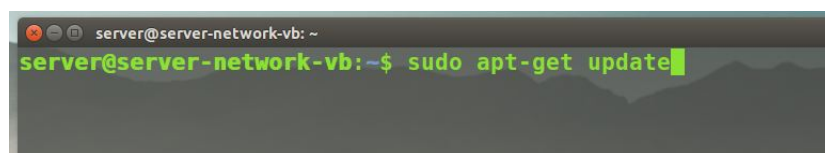
1. Open the Terminal by pressing Ctrl + Alt + T.
2. Type the following command, to download the repository for Python3. We install version 3.6 on our machine.

A terminal window with a dark background and light green text. The prompt is 'server@server-network-vb: ~'. The command entered is 'sudo add-apt-repository ppa:jonathonf/python-3.6'.

```
server@server-network-vb: ~  
server@server-network-vb:~$ sudo add-apt-repository ppa:jonathonf/python-3.6
```

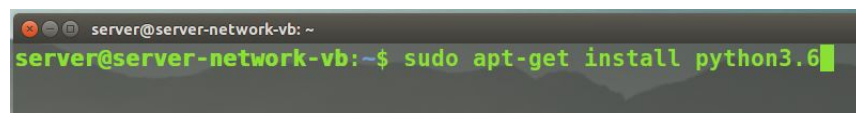
(A.1)

3. Then we update the repository and install the python.

A terminal window with a dark background and light green text. The prompt is 'server@server-network-vb: ~'. The command entered is 'sudo apt-get update'.

```
server@server-network-vb: ~  
server@server-network-vb:~$ sudo apt-get update
```

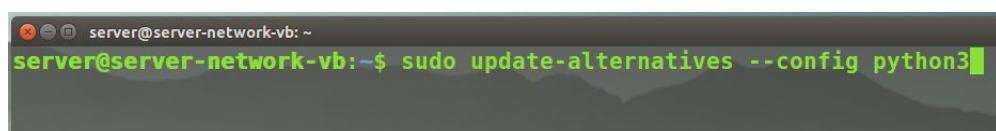
(A.2)

A terminal window with a dark background and light green text. The prompt is 'server@server-network-vb: ~'. The command entered is 'sudo apt-get install python3.6'.

```
server@server-network-vb: ~  
server@server-network-vb:~$ sudo apt-get install python3.6
```

(A.3)

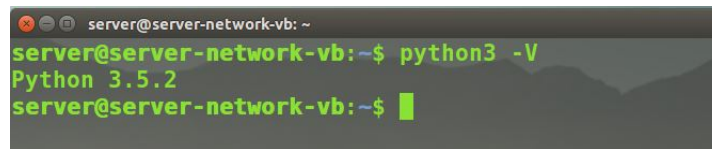
4. if you have alternate versions of Python on the system, type this command to select the version of python.

A terminal window with a dark background and light green text. The prompt is 'server@server-network-vb: ~'. The command entered is 'sudo update-alternatives --config python3'.

```
server@server-network-vb: ~  
server@server-network-vb:~$ sudo update-alternatives --config python3
```

(A.4)

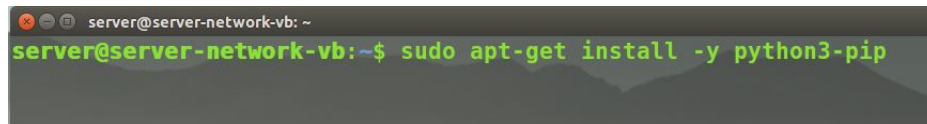
5. Then we can see the Python version installed by putting this command.



```
server@server-network-vb: ~  
server@server-network-vb:~$ python3 -V  
Python 3.5.2  
server@server-network-vb:~$
```

(A.5)

6. To install packages in Python, install pip.



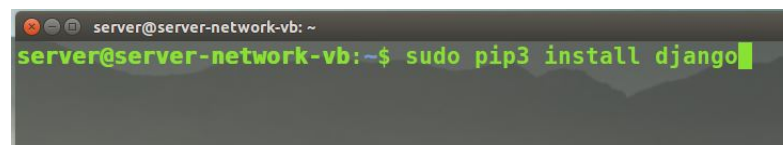
```
server@server-network-vb: ~  
server@server-network-vb:~$ sudo apt-get install -y python3-pip
```

(A.6)

## A.2 Installing Django Package:

Once Python3 is installed we use pip3 command to install django packages.

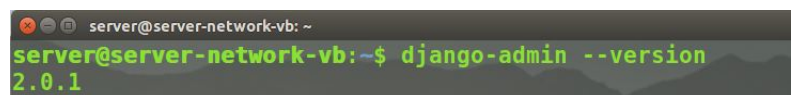
1. Use this command to download and install django package.



```
server@server-network-vb: ~  
server@server-network-vb:~$ sudo pip3 install django
```

(A.7)

2. To confirm the version and installation, type this.



```
server@server-network-vb: ~  
server@server-network-vb:~$ django-admin --version  
2.0.1
```

(A.8)

## A.3 Creating a Basic Django Project:

We can create a basic starter project in Django in this way.

1. First type the following commands to create a Test project. This will create folder Test which has manage.py. This python file can be used to start the project. Once set, run the server. By Default it runs on localhost.

```
server@server-network-vb: ~/Test
server@server-network-vb:~$ django-admin startproject Test
server@server-network-vb:~$ cd Test
server@server-network-vb:~/Test$ ls
manage.py  Test
server@server-network-vb:~/Test$ python3 manage.py runserver
Performing system checks...

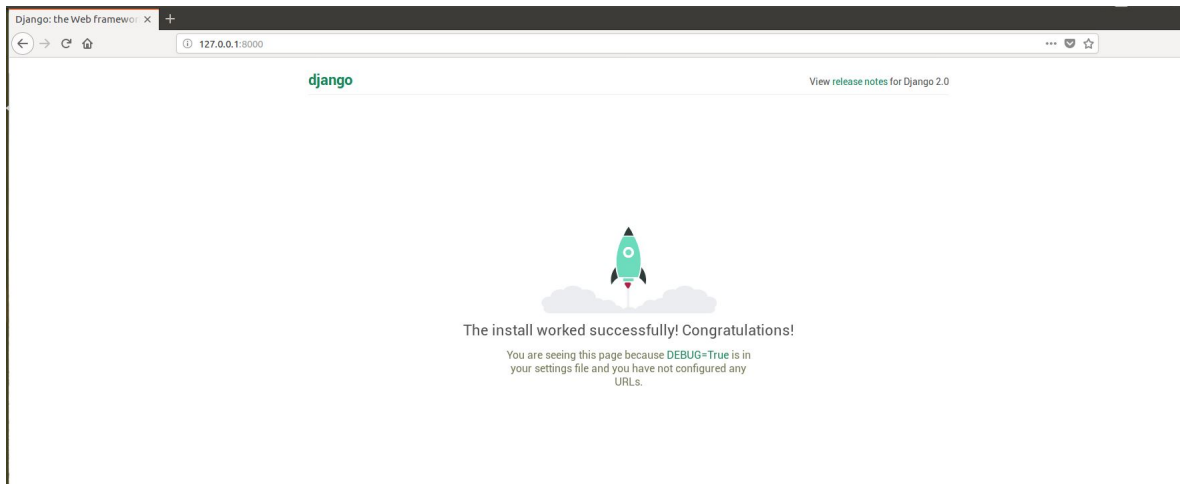
System check identified no issues (0 silenced).

You have 14 unapplied migration(s). Your project may not work properly until you
apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

April 27, 2018 - 06:43:04
Django version 2.0.1, using settings 'Test.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

(A.9)

2. Launching the address <http://127.0.0.1:8000/> we get this on our Browser.

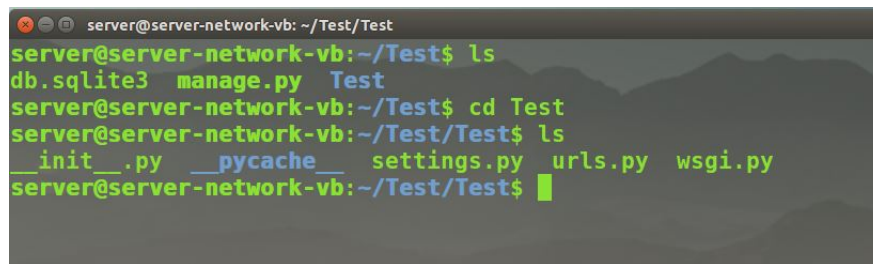


(A.10)

## A.4 Setting up Custom Hostname for Django Apps:

For real-time applications it's not feasible to run them on localhost. Hence we need to add the hosts in the Django such that it recognizes it.

The Test project has a default App called Test. This app contains all the necessary configurations for the Project. This should not be modified or deleted without due care.



```
server@server-network-vb: ~/Test/Test
server@server-network-vb:~/Test$ ls
db.sqlite3  manage.py  Test
server@server-network-vb:~/Test$ cd Test
server@server-network-vb:~/Test/Test$ ls
__init__.py  __pycache__  settings.py  urls.py  wsgi.py
server@server-network-vb:~/Test/Test$
```

(A.11)

The file settings.py is very essential for all the configurations.

```
Django settings for Test project.
Generated by 'django-admin startproject' using Django 2.0.1.
For more information on this file, see
https://docs.djangoproject.com/en/2.0/topics/settings/
For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.0/ref/settings/
"""
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 't%vb*p7zy0y5pz82*xl^(=lzu#3^5)v6e_pggw_s0l5y^jepsi'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

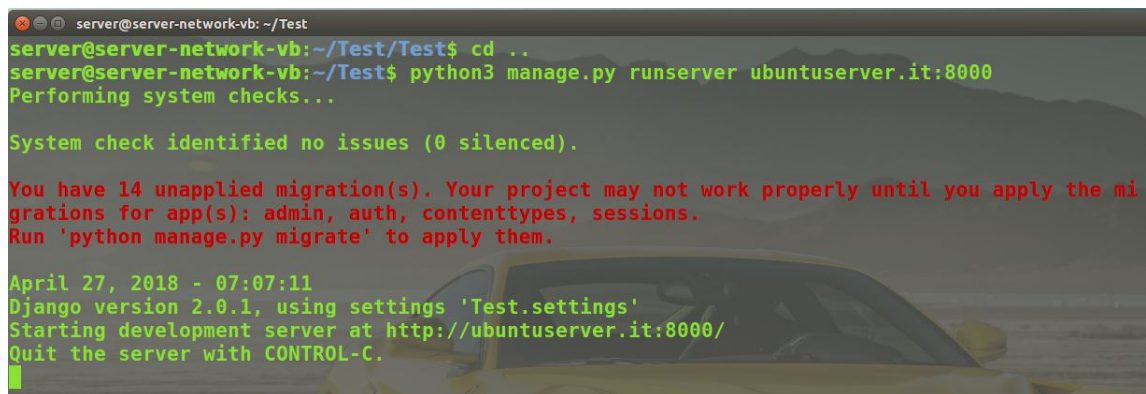
(A.12)

We add our deployment host configurations in ALLOWED\_HOSTS list and save the file.

```
| ALLOWED_HOSTS = ['ubuntuserver.it', '192.168.56.101', '127.0.0.1', 'localhost']|
```

(A.13)

Here ubuntuserver.it is a custom hostname which is setup for use. We can test it by applying our new host name.

A terminal window titled 'server@server-network-vb: ~/Test' showing the execution of 'python3 manage.py runserver ubuntuserver.it:8000'. The output includes 'Performing system checks...', 'System check identified no issues (0 silenced).', a warning about 14 unapplied migrations, the date and time 'April 27, 2018 - 07:07:11', 'Django version 2.0.1, using settings \'Test.settings\'', and 'Starting development server at http://ubuntuserver.it:8000/'. The prompt 'Quit the server with CONTROL-C.' is visible at the bottom.

```
server@server-network-vb: ~/Test
server@server-network-vb:~/Test/Test$ cd ..
server@server-network-vb:~/Test$ python3 manage.py runserver ubuntuserver.it:8000
Performing system checks...

System check identified no issues (0 silenced).

You have 14 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

April 27, 2018 - 07:07:11
Django version 2.0.1, using settings 'Test.settings'
Starting development server at http://ubuntuserver.it:8000/
Quit the server with CONTROL-C.
```

(A.14)

## APPENDIX B

### JINJA TEMPLATING

Jinja2 is a template engine for use in Python. It is fully supported with Unicode, an optional integrated sand-boxed execution environment, and widely used in other environments. It has been used with our project to load dynamic content in the web-pages.

he following is an excerpt of how Med objects are rendered using Jinja Templating:

```
{% for med in prescriptions %}
<tr>
  <td>{{ med.symp }}</td>
  <td>{{ med.name }}</td>
  <td>{{ med.dose }}</td>
</tr>
{% endfor %}
```

(B.1)

“{{ }}” defines Interpolation which retrieves the objects definition and processes them as a normal code do, in the HTML.

Features of Jinja include:

- Sand-boxed execution mode. Makes use of categorizing the templates which are safe or unsafe of use
- Efficient auto HTML escaping-system for XSS prevention.
- You can inherit templated definition
- Just-in-time compilation to Python bytecode for best run-time performance.
- Optional ahead-of-time compilation
- debugging is easy as errors are put into the standard Python trace-back system.
- Configurable syntax. Can be moulded for others.



## REFERENCES

1. Dakappa, P. H., Prasad, K., Rao, S. B., Bolumbu, G., Bhat, G. K., and Mahabala, C. (2017). “A predictive model to classify undifferentiated fever cases based on twenty-four-hour continuous tympanic temperature recording.” *Journal of Healthcare Engineering*, 2017.
2. Fatima, M. and Pasha, M. (2017). “Survey of machine learning algorithms for disease diagnostic.” *Journal of Intelligent Learning Systems and Applications*, 9(01), 1–16.
3. Oguntimilehin, A., Adetunmbi, A., and Abiola, O. (2013). “A machine learning approach to clinical diagnosis of typhoid fever.” *A Machine Learning Approach to Clinical Diagnosis of Typhoid Fever*, 2(4), 671–676.
4. Oo, H. N. et al. (2016). “Fever classification and remedial recommendation system.” *International Journal of Open Information Technologies*, 4(12).
5. SciKitLearn. “Stratified shuffled split.” SKLearn, <[http://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.StratifiedShuffleSplit.html](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html)>.

# AUTOMATED REMEDY SYSTEM FOR FEVER USING MACHINE LEARNING

## ORIGINALITY REPORT

6%

SIMILARITY INDEX

4%

INTERNET SOURCES

2%

PUBLICATIONS

6%

STUDENT PAPERS

## PRIMARY SOURCES

1

Submitted to SRM University

Student Paper

2%

2

Submitted to B.S. Abdur Rahman University

Student Paper

1%

3

[www.analyticsvidhya.com](http://www.analyticsvidhya.com)

Internet Source

1%

4

Submitted to iGroup

Student Paper

1%

5

Submitted to Engineers Australia

Student Paper

<1%

6

Talko B. Dijkhuis, Frank J. Blaauw, Miriam W. van Ittersum, Hugo Velthuijsen, Marco Aiello. "Personalized Physical Activity Coaching: A Machine Learning Approach", Sensors, 2018

Publication

<1%

7

"The International Conference on Advanced Machine Learning Technologies and Applications (AMLT A2018)", Springer Nature,

<1%

2018

Publication

8

Kella Bhanu Jyothi, K. Hima Bindu. "Chapter 17  
A Case Study in R to Recognize Human  
Activity Using Smartphones", Springer Nature,  
2018

Publication

<1%

9

Submitted to HELP UNIVERSITY

Student Paper

<1%

10

Submitted to De La Salle University - Manila

Student Paper

<1%

Exclude quotes On

Exclude bibliography On

Exclude matches < 10 words