

A
MAJOR PROJECT
on
“PREDICTION OF THYROID DISEASE(HYPOTHYROID)”

Submitted to Jawaharlal Nehru Technological University, Hyderabad in partial fulfillment of the requirement for the award of the Under Graduate Degree.

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING



Submitted By

KOLA RAJU	20QA1A0566
KOTHA RAHUL	20QA1A0569
KONTHAM SHIVARAM	20QA1A0568
MOHD OZAIR	20QA1A0594

under guidance of

Mr. B VINOD KUMAR (M-TECH) Asst.Prof

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
BRILLIANT INSTITUTE OF ENGINEERING AND TECHNOLOGY
Abdullapurmet (V), Abdullapurmet (M), R.R.Dist, Pin-501505

Accredited by NAAC with 'A' Grade
(Affiliated to JNTU Hyderabad)
2023-2024

BRILLIANT INSTITUTE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTU, Hyderabad) Accredited
by NAAC with 'A' Grade

Abdullapurmet (V & M), R.R (Dist). Telangana -501505

Ph: 040- 23437788,Fax: 040-23437776 www.b-iet.ac.in/



CERTIFICATE

*This is to certify that the project report entitled “**PREDICTION OF THYROID DISEASE(HYPOTHYROID)**” is a bonafide work done by **Kola Raju(20QA1A0566)** , **Kotha Rahul(20QA1A0569)** , **Kontham Shivaram(20QA1A0568)**, **Mohd Ozair(20QA1A0594)** B.tech from Computer Science & Engineering is a record bonafide work carried out by us. The results embodied in this report have not been submitted to any other University for the award of any Degree.*

Internal Guide

*Head of the Department,
Dr J Reddeppa Reddy, Ph D
C.S.E*

Principal

Extenal Guide

ACKNOWLEDGEMENT

This report will certainly not be completed without due acknowledgements paid to all those who have helped us in doing our project work.

We thankful to our project internal guide **Mr. B VINOD KUMAR** M.Tech. Assistant professor for giving us moral support and conducting seminars smoothly to us throughout this project work.

It is our privilege to thank **Dr.J.REDDEPPA REDDY**, Ph.D, M.Tech, Assistant professor- Head of the Department,CSE. For this encouragement during the progress of this project work.

We thankful to both Teaching and non-teaching staff members of **CSE DEPARTMENT** for their kind cooperation and all sorts of help to bring out this project work successfully.

We derive great pleasure in expressing our sincere gratitude to our principal **Dr.SHAIK RUSTHUM** for his timely suggestions , which helped us to complete this work successfully.

We thankful to **BRILLIANT INSTITUTE OF ENGINEERING AND TECHNOLOGY** for providing required facilities during our project work.

We would like to thank our parents and our friends for being supportive all the time, and we are very much obliged to them.

KOLA RAJU	20QA1A0566
KOTHA RAHUL	20QA1A0569
KONTHAM SHIVARAM	20QA1A0568
MOHD OZAIR	20QA1A0594

DECLARATION

We are hereby declare that the project report entitled “**PREDICTION OF THYROID DISEASE(HYPOTHYROID)**” carried under guidance of **Mr. B VINOD KUMAR (M TECH)** Assistant Professor – CSE dept is being submitted in partial fulfillment of the academic requirements for the degree of Bachelor of Technology in Computer Science and Engineering.

This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project have not been submitted to any other university or institute for the award of any other degree.

KOLA RAJU	20QA1A0566
KOTHA RAHUL	20QA1A0569
KONTHAM SHIVARAM	20QA1A0568
MOHD OZAIR	20QA1A0594

ABSTRACT

Thyroid disease is one of the most common diseases among the female mass in Bangladesh. Hypothyroid is a common variation of thyroid disease. It is clearly visible that hypothyroid disease is mostly seen in female patients. Most people are not aware of that disease as a result of which, it is rapidly turning into a critical disease. It is very much important to detect it in the primary stage so that doctors can provide better medication to keep it from turning into a serious matter. Predicting disease in machine learning is a difficult task. Machine learning plays an important role in predicting diseases. Again distinct feature selection techniques have facilitated this process of prediction and assumption of diseases. There are two types of thyroid diseases namely 1. Hyperthyroid and 2. Hypothyroid. Here, in this paper, we have attempted to predict hypothyroid in the primary stage. To do so, we have mainly used three feature selection techniques along with diverse classification techniques. Feature selection techniques used by us are Recursive Feature Selection(RFE), Univariate Feature Selection(UFS) and Principal Component Analysis(PCA) along with classification algorithms named Support Vector Machine(SVM), Decision Tree(DT), Random Forest(RF), Logistic Regression(LR) and Naive Bayes(NB). By observing the results, we could extrapolate that the RFE feature selection technique helps us to provide constant 99.35% accuracy for all four classification algorithms. Thus it's deduced from our research that RFE helps each classifier to attain better accuracy than all the other feature selection methods used.

INDEX

S.NO	CONTENTS	PAGE NO
	ABSTRACT	1
	LIST OF FIGURES	4
	LIST OF SCREENSHOTS	5
1	INTRODUCTION	6
2	LITREATURE SURVEY	7
3	SYSTEM ANALYSIS	9
3.1	Existing system	9
3.2	Proposed system	9
4	FEASIBILITY STUDY	10
4.1	Economical Feasibility	10
4.2	Technical Feasibility	10
4.3	Social Feasibility	10
5	SYSTEM SPECIFICATIONS	11
5.1	Hardware Requirements	11
5.2	Software Requirements	11
6	SYSTEM DESIGN	12
6.1	System Architecture	12
6.2	Data flow diagram	12
6.3	UML diagram	13
6.4	Use Case Diagram	14
6.5	Class Diagram	14
6.6	Sequence Diagram	15
6.7	Activity Diagram	16
7	SOFTWARE ENVIRONMENT	19
7.1	Python	19
7.1.1	Python Identifiers	20
7.1.2	Reserved Words	21
7.1.3	Lines and Indentation	21
7.1.4	Command Line Arguments	26
7.1.5	Python Lists	26
7.1.6	Properties of Dictionary Keys	30
7.2	Django	32
7.2.1	Create a Project	33
7.2.2	Create an Application	36
8	SYSTEM IMPLEMENTATION	43
8.1	Modules	43
8.1.1	User	43
8.1.2	Admin	43
8.1.3	Data Preprocessing	43
8.1.4	Machine Learning	43

9	SYSTEM TESTING	44
9.1	Types of Testing	44
9.1.1	Unit Testing	44
9.1.2	Integration Testing	44
9.1.3	Functional Testing	44
9.1.4	System Test	45
9.1.5	White Box Testing	45
9.1.6	Black Box Testing	45
10	INPUT AND OUTPUT DESIGN	47
10.1	Input Design	47
10.2	Output Design	47
11	SOURCE CODE	49
12	SCREENSHOTS	69
13	SAMPLE TEST CASES	72
14	FURTHER ENCHANCEMENT	74
15	CONCLUSION	75
16	REFERENCES	76

LIST OF FIGURES

S.NO	Contents	Page No
Fig 1	System Architecture	12
Fig 2	Data Flow Diagram	13
Fig 3	Use Case Diagram	14
Fig 4	Class Diagram	15
Fig 5	Sequence Diagram	16
Fig 6.1	Activity Diagram 1	17
Fig 6.2	Activity Diagram 2	18
Fig 7.1	Data Base of Django	33
Fig 7.2	Flow chart of Django	33

LIST OF SCREENSHOTS

S.No	Contents	Page No
12.1	Home Page	65
12.2	User Register Form	65
12.3	Admin Login Form	66
12.4	Admin Home Page	66
12.5	View users and Activate	67
12.6	Admin View Results	67
12.7	User Login Form	68
12.8	User Home Page	68
12.9	User View Dataset	69
12.10	Server Side Results	69
12.11	User View Accuracy	70
12.12	Prediction Form	70
12.13	Prediction Results	71

CHAPTER-1

INTRODUCTION

At the current state, the thyroid is one of the most critical diseases of all and it has quite the potential to be transformed into a common disease among the female mass. In Bangladesh, according to experts, 50 million people suffer from thyroid disease. Among them, females are at 10 times more risk of being affected with thyroid disease. Though a vast majority of 50 million people are affected with thyroid disease, yet almost 30 million people among them are totally not aware of this condition. A study from the Bangladesh Endocrine Society(BES) depicts that around 20-30% of females are suffering from thyroid disease [14]. The thyroid is a gland that is situated in the middle of the neck in our body. It is butterfly-shaped and small in size. It secretes several hormones that are mixed with blood and travel across the body to control various activities. The thyroi hormone is responsible for conserving metabolism, sleep, growth, sexual function, and mood. Depending on the secretion of thyroid hormone we can feel tired or restless and also may have weight loss. There are two main thyroid hormones: Triiodothyronine (T3) and Thyroxin (T4). These two hormones are mainly responsible for maintaining the energy in our bodies. Thyroid Stimulating Hormone(TSH) is produced by the pituitary gland that helps the thyroid gland to release T3 and T4.

CHAPTER – 2

LITREATURE SURVEY

1) Early diagnosis of heart disease using classification and regression trees

AUTHORS: Amir Mohammad Amiri; Giuliano Armano

Early diagnosis of heart defects are very important for medical treatment. In this paper, we propose an automatic method to segment heart sounds, which applies classification and regression trees. The diagnostic system, designed and implemented for detecting and classifying heart diseases, has been validated with a representative dataset of 116 heart sound signals, taken from healthy and unhealthy medical cases. The ultimate goal of this research is to implement a heart sounds diagnostic system, to be used to help physicians in the auscultation of patients, with the goal of reducing the number of unnecessary echocardiograms and of preventing the release of newborns that are in fact affected by a heart disease. In this study, 99.14% accuracy, 100% sensitivity, and 98.28% specificity were obtained on the dataset used for experiments.

2) An Intelligent System for Thyroid Disease Classification and Diagnosis

AUTHORS: A K Aswathi; Anil Antony

Data mining Techniques play a vital role in healthcare organizations such as for decision making, diagnosing disease and giving better treatment to the patients. Thyroid gland plays a major role in maintaining the metabolism of human body. Data mining in health care industry provides a systematic use of the medical data. Thyroid diseases are most common today. Early changes in the thyroid gland will not affect the proper working of the gland. By the early identification of thyroid disorders, better treatment can be provided in the early stage thus can avoid thyroid replacement therapy and thyroid removal up to an extent. This paper proposes a method for the classification and diagnosis of thyroid disease that a user is suffering from along with disease description and healthy advices. Support Vector Machine is used for classification. To optimize SVM parameters Particle Swarm Optimization is applied. User is provided with a window to enter the details such as the values of TSH, T3, T4 etc. There may be some values missing while the user entering the values. K-Nearest Neighbor algorithm is used for approximating the missing values in the user input.

3) Prediction of thyroid Disease Using Data Mining Techniques

AUTHORS : Amina Begum; A Parkavi

Classification based Data mining plays important role in various healthcare services. In healthcare field, the important and challenging task is to diagnose health conditions and proper treatment of disease at the early stage. There are various diseases that can be diagnosed early and can be treated at the early stage. As for example, Thyroid diseases. the traditional ways of diagnosing thyroid diseases depends on clinical examination and many blood tests. The Main

task is to detect disease diagnosis at the early stages with higher accuracy. Data mining techniques plays an important role in healthcare field for making decision, disease diagnosis and providing better treatment for the patients at low cost. Thyroid disease Classification is an important task. The purpose of this study is predication of thyroid disease using different classification techniques and also to find the TSH, T3,T4 correlation towards hyperthyroidism and hypothyroidism and also to finding the TSH, T3,T4 correlation with gender towards hyperthyroidism and hypothyroidism.

4) Feature selection algorithms to improve thyroid disease diagnosis

AUTHORS : K. Pavva; B. Srinivasan

Correct and early diagnosis of diseases is important and mandatory in healthcare industry for correct and timely treatment. This fact is more important in diseases like thyroid, which is very difficult to detect as its symptoms coincide with several diseases. Usage of machine learning algorithms for thyroid disease diagnosis is prominent. A typical thyroid disease diagnosis system uses three main steps, namely, feature extraction, feature selection and classification. The main goal of this paper is to analyze the use of filter-based (F-Score) and wrapper-based (Recursive Feature Elimination) feature selection algorithms on its effect on disease identification and classification. The analysis is also performed with Principle Component Analysis dimensionality reduction algorithms. Performance evaluation was performed with three metrics, namely, accuracy, sensitivity and specificity. Four classifiers, namely, MultiLayer Perceptron, Back Propagation Neural Network, Support Vector Machine and Extreme Learning Machine were used to analyze the selected algorithms. Experimental results showed that while both F-Score and Recursive Feature Elimination improved the performance of thyroid disease diagnosis, the wrapper-based algorithm produced maximum efficiency and produced a maximum accuracy of 98.14% with ELM classifier.

5) Thyroid Disease Diagnosis Based on Genetic Algorithms Using PNN and SVM

AUTHORS: Fatemeh Saiti; Afsaneh Alavi Naini; Mahdi Aliyari Shoorehdeli; Mohammad Teshnehlab

Thyroid gland produces thyroid hormones to help the regulation of the body's metabolism. The abnormalities of producing thyroid hormones are divided into two categories. Hypothyroidism which is related to production of insufficient thyroid hormone and hyperthyroidism related to production of excessive thyroid hormone. Separating these two diseases is very important for thyroid diagnosis. Therefore support vector machines and probabilistic neural network are proposed to classification. These methods rely mostly on powerful classification algorithms to deal with redundant and irrelevant features. In this paper feature selection is argued as an important problem via diagnosis and demonstrate that GAs provide a simple, general and powerful framework for selecting good subsets of features leading to improved diagnosis rates. Thyroid disease datasets are taken from UCI machine learning dataset.

CHAPTER – 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

The thyroid is a gland that is situated in the middle of the neck in our body. It is butterfly-shaped and small in size. It secretes several hormones that are mixed with blood and travel across the body to control various activities. The thyroid hormone is responsible for conserving metabolism, sleep, growth, sexual function, and mood. At the current state, the thyroid is one of the most critical diseases of all and it has quite the potential to be transformed into a common disease among the female mass.

DISADVANTAGES OF EXISTING SYSTEM:

- Predicting disease in machine learning is a difficult task.
- It is not often known in advance which features will provide the best discrimination .

Algorithm:KNN, LINEAR Regression.

3.2 PROPOSED SYSTEM:

This paper constructed Classification of Hypothyroid Disorder using Optimized SVM Method. In this work, they proposed a method of detecting hypothyroid disorder level using classification machine learning techniques, namely KNN (K Nearest Neighbor), SVM (Support Vector Machines), LR (Logistic Regression), and NN (Artificial Neural Network). Logistic Regression method achieved 96.08% accuracy among the other three classifiers but SVM provides the highest accuracy of 99.08% after standardizing the data and parameter tuning.

ADVANTAGES OF PROPOSED SYSTEM:

- In this proposed work they use Machine Learning Algorithms such as SVM(99.63), K-NN(98.62), Decision Trees(75.76), ANN(97.5) were used to predict the estimated risk on a patient's chance of obtaining thyroid disease.
- selection technique helps us to provide constant 99.35% accuracy for all four classification

Algorithm: Support Vector Machine(SVM), Decision Tree(DT), Random Forest(RF), Logistic Regression(LR).

CHAPTER – 4

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ◆ **ECONOMICAL FEASIBILITY**
- ◆ **TECHNICAL FEASIBILITY**
- ◆ **SOCIAL FEASIBILITY**

4.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER – 5

SYSTEM SPECIFICATIONS

5.1 HARDWARE REQUIREMENTS:

- ❖ **System** : Intel i5.
- ❖ **SSD** : 512GB.
- ❖ **Monitor** : 15.6` Colour Monitor.
- ❖ **Mouse** : Optical Mouse.
- ❖ **Ram** : 8GB.

5.2 SOFTWARE REQUIREMENTS:

- ❖ **Operating system** : Windows 11.
- ❖ **Coding Language** : Python.
- ❖ **Front-End** : Html. CSS
- ❖ **Designing** : Html,css,javascript.
- ❖ **Data Base** : SQLite.

CHAPTER – 6

SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE:

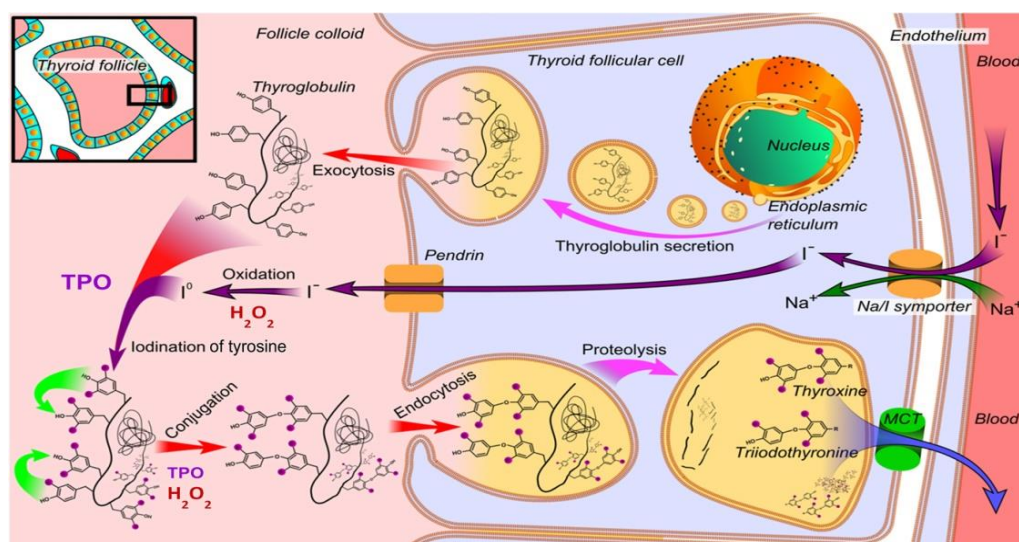


Fig 1 System Architecture

6.2 DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

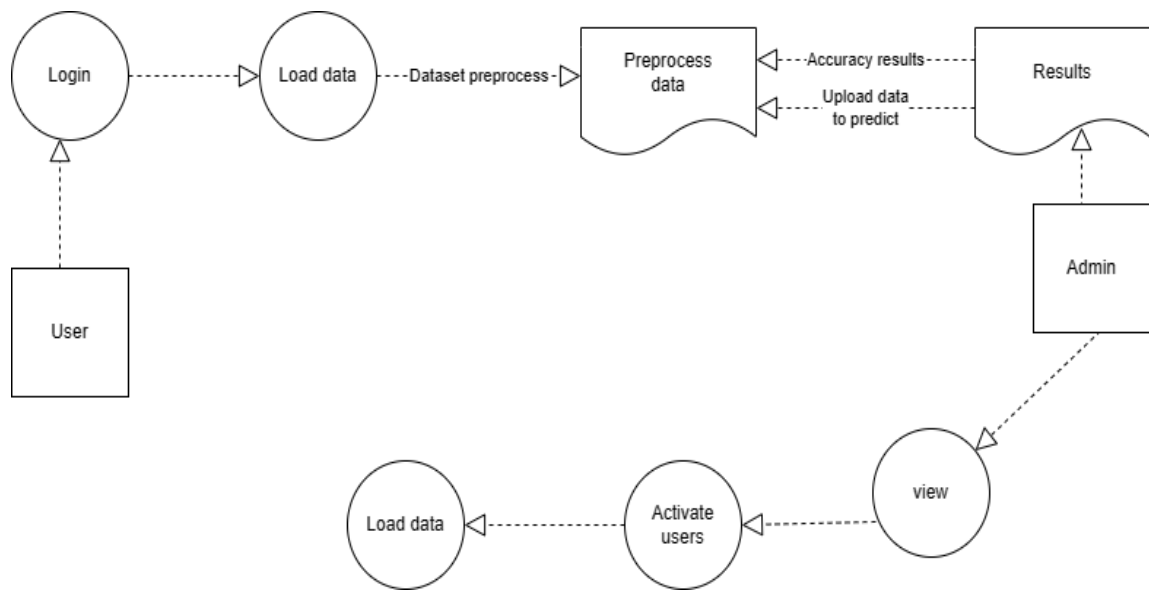


Fig 2 Data Flow Diagram

6.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.

7. Integrate best practices.

6.4 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

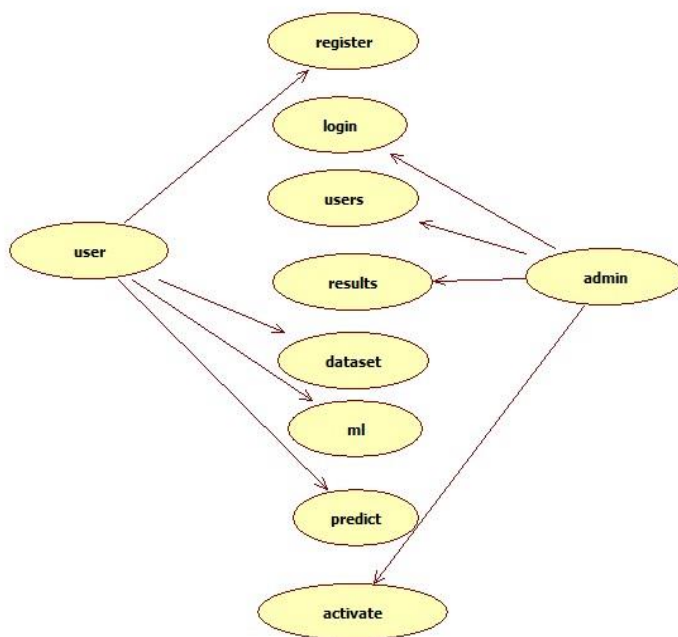


Fig 3 Use Case Diagram

6.5 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

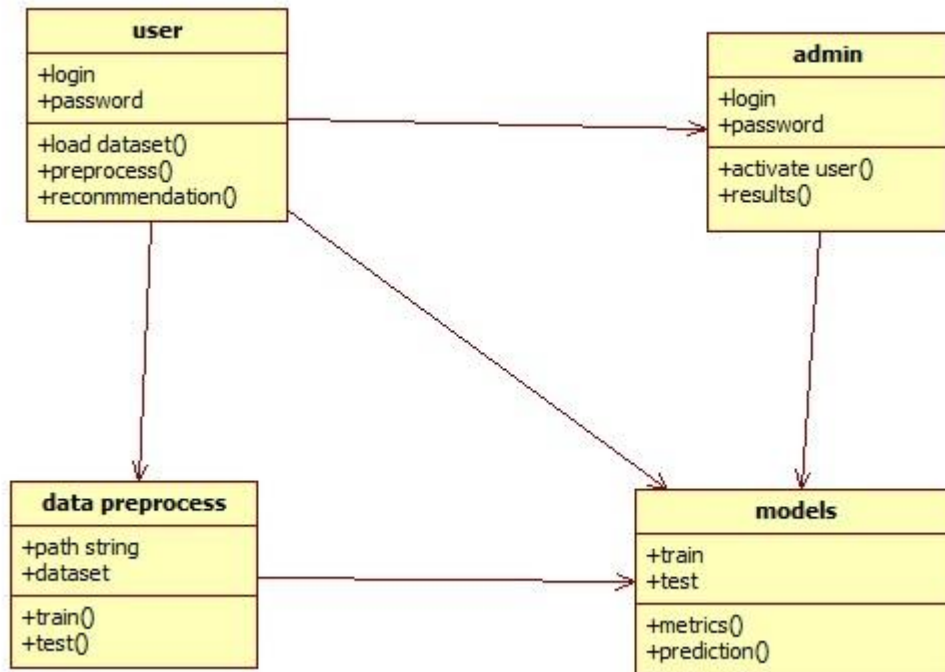


Fig 4 Class Diagram

6.6 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

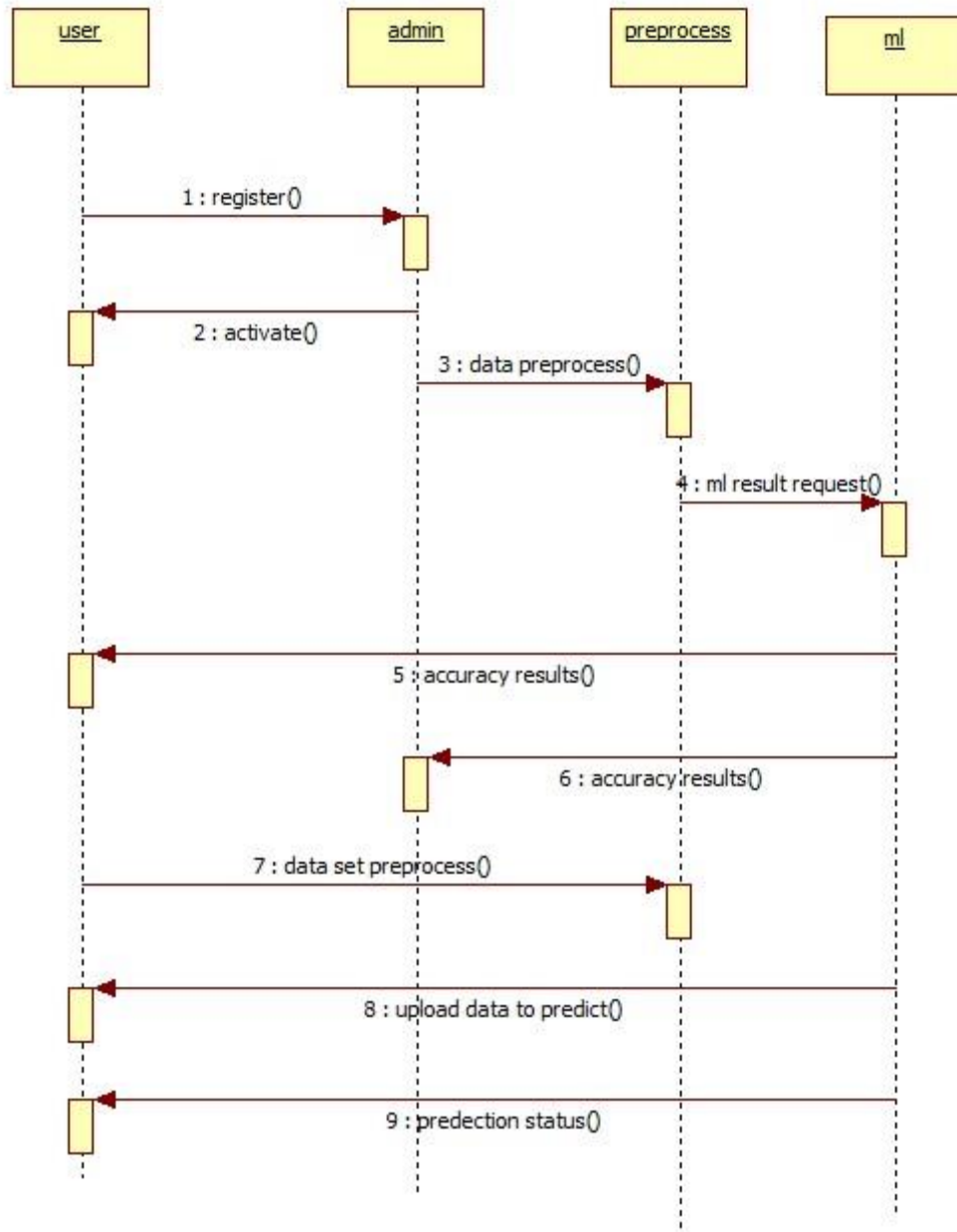


Fig 5 Sequence Diagram

6.7 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

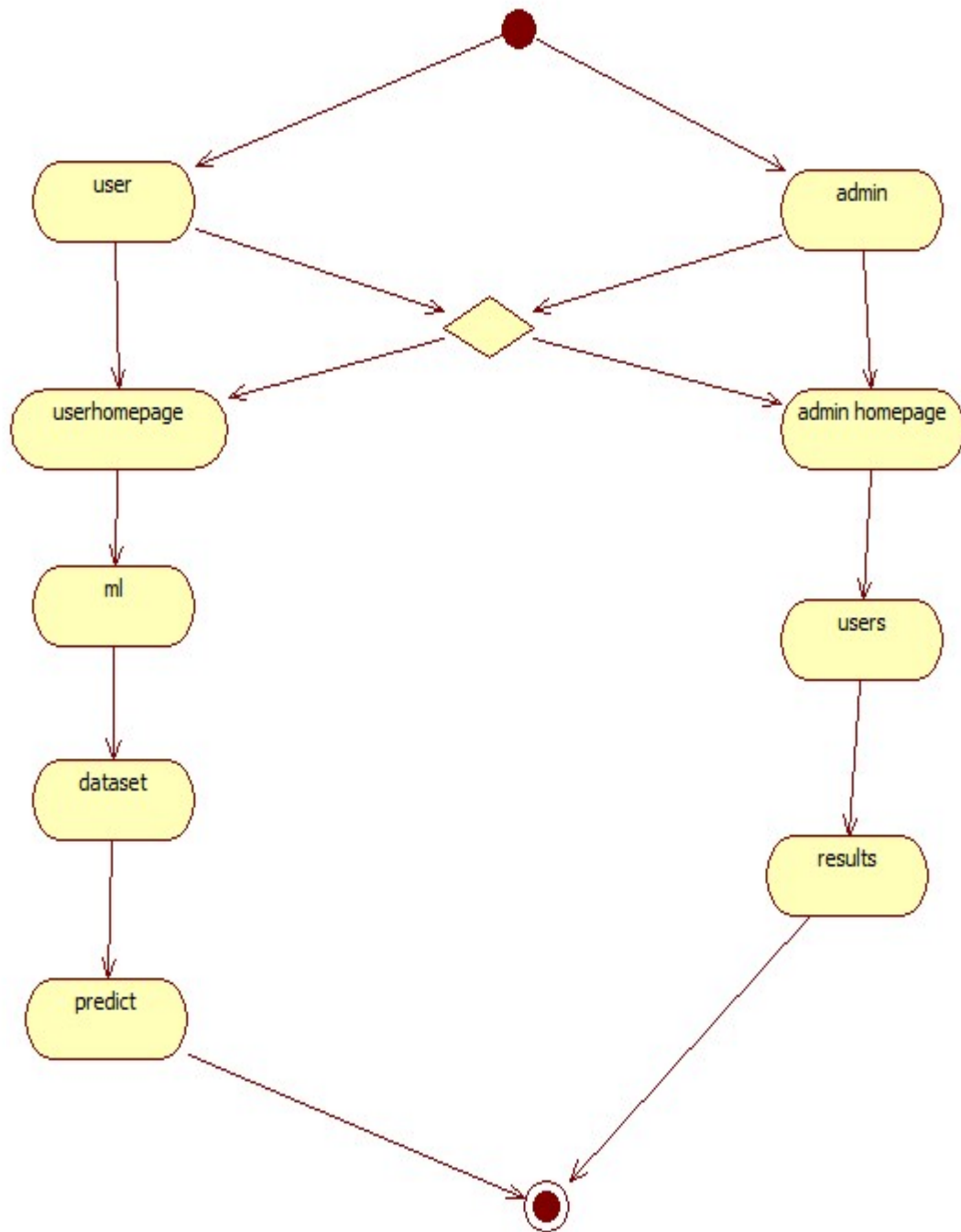


Fig 6.1 Activity Diagram 1

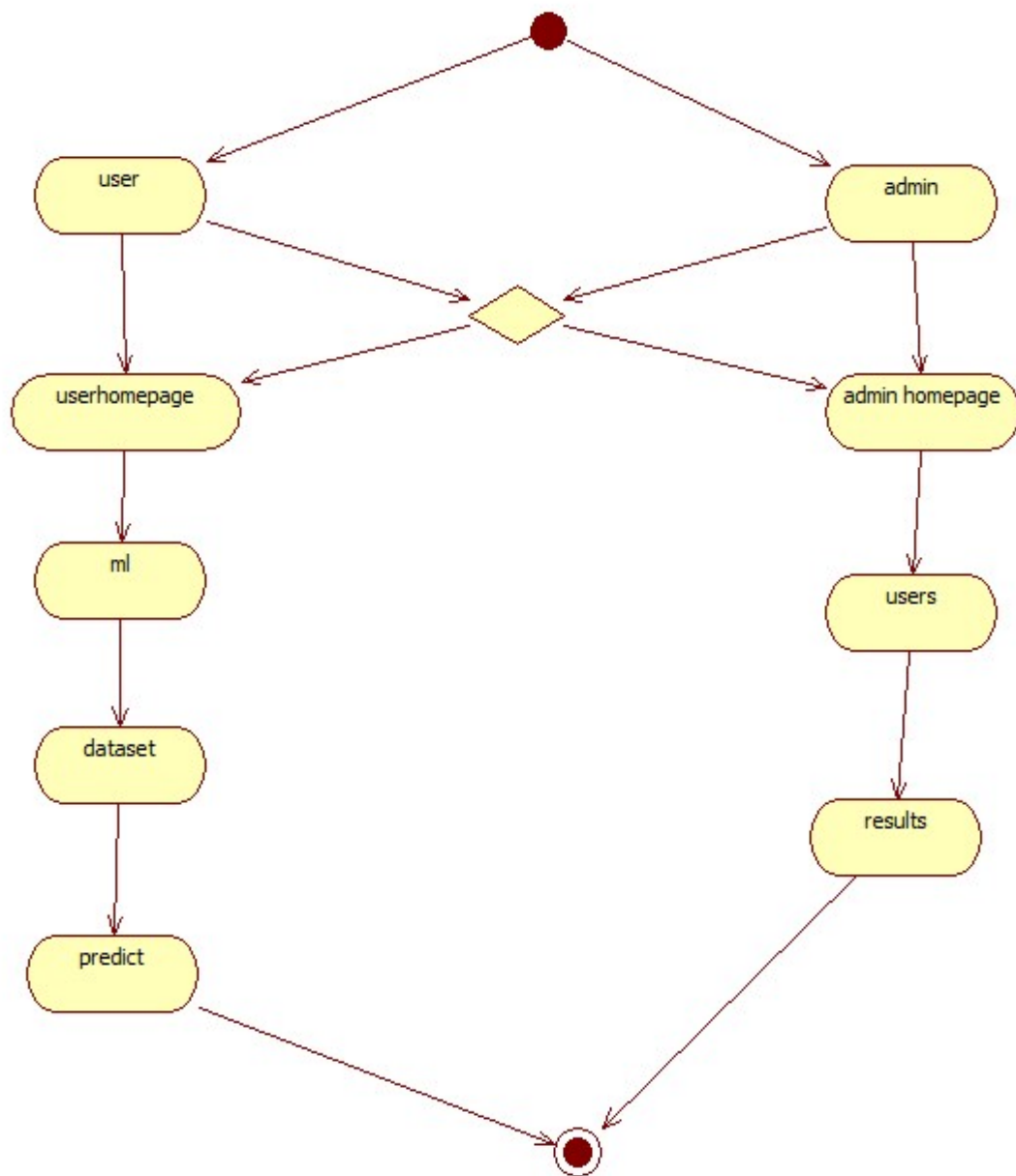


Fig 6.2 Activity Diagram 2

CHAPTER – 7

SOFTWARE ENVIRONMENT

7.1 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ python
```

```
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
```

```
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result –

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file –

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows –

```
$ python test.py
```

This produces the following result –

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file –

Live Demo

```
#!/usr/bin/python
```

```
print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows –

```
$ chmod +x test.py    # This is to make file executable
```

```
$/test.py
```

This produces the following result –

```
Hello, Python!
```

7.1.1 Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, \$, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers –

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

7.1.2 Reserved Words:

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and exec not

assert finally or

break for pass

class from print

continue global raise

def if return

del import try

elif in while

else is with

except lambda yield

7.1.3 Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

if True:

```
    print "True"
```

```
else:
```

```
    print "False"
```

However, the following block generates an error –

```
if True:
```

```
    print "Answer"
```

```
    print "True"
```

```
else:
```

```
    print "Answer"
```

```
    print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks –

Note – Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python
```

```
import sys
```

```
try:
```

```
    # open file stream
```

```
    file = open(file_name, "w")
```

```
except IOError:
```

```
    print "There was an error writing to", file_name
```

```
    sys.exit()
```

```
print "Enter '", file_finish,
```

```
print "' When finished"
```

```
while file_text != file_finish:
```

```

file_text = raw_input("Enter text: ")

if file_text == file_finish:

    # close the file

    file.close

    break

file.write(file_text)

file.write("\n")

file.close()

file_name = raw_input("Enter filename: ")

if len(file_name) == 0:

    print "Next time please enter something"

    sys.exit()

try:

    file = open(file_name, "r")

except IOError:

    print "There was an error reading file"

    sys.exit()

file_text = file.read()

file.close()

print file_text

```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example –

```
total = item_one + \
```

```
item_two + \
```

```
item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example –

```
days = ['Monday', 'Tuesday', 'Wednesday',  
        'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple ("" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal –

```
word = 'word'
```

```
sentence = "This is a sentence."
```

```
paragraph = """This is a paragraph. It is  
made up of multiple lines and sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python
```

```
# First comment
```

```
print "Hello, Python!" # second comment
```

This produces the following result –

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression –

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows –

```
# This is a comment.
```

```
# This is a comment, too.
```

```
# This is a comment, too.
```

```
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
"""This is a multiline comment."""
```

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action –

```
#!/usr/bin/python
```

```
raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example

if expression :

 suite

elif expression :

 suite

else :

 suite

7.1.4 Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h –

```
$ python -h
```

```
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
```

Options and arguments (and corresponding environment variables):

-c cmd : program passed in as string (terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

7.1.5 Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example –

```
list1 = ['physics', 'chemistry', 1997, 2000];
```

```
list2 = [1, 2, 3, 4, 5 ];
```

```
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000);
```

```
tup2 = (1, 2, 3, 4, 5 );
```

```
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing –

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value –

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example –

Live Demo

```
#!/usr/bin/python
```

```
tup1 = ('physics', 'chemistry', 1997, 2000);
```

```
tup2 = (1, 2, 3, 4, 5, 6, 7 );
```

```
print "tup1[0]: ", tup1[0];
```

```
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result –

```
tup1[0]: physics
```

```
tup2[1:5]: [2, 3, 4, 5]
```

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example –

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

```
print "dict['Name']: ", dict['Name']
```

```
print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result –

```
dict['Name']: Zara
```

```
dict['Age']: 7
```

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows –

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

```
print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result –

```
dict['Alice']:
```


Traceback (most recent call last):

File "test.py", line 4, in <module>

```
print "dict['Alice']: ", dict['Alice'];
```

KeyError: 'Alice'

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example –

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

```
dict['Age'] = 8; # update existing entry
```

```
dict['School'] = "DPS School"; # Add new entry
```

```
print "dict['Age']: ", dict['Age']
```

```
print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result –

```
dict['Age']: 8
```

```
dict['School']: DPS School
```

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example –

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

```
del dict['Name']; # remove entry with key 'Name'
```

```
dict.clear();    # remove all entries in dict
```

```
del dict ;      # delete entire dictionary
```

```
print "dict['Age']: ", dict['Age']
```

```
print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after `del dict` dictionary does not exist any more –

```
dict['Age']:
```

Traceback (most recent call last):

```
File "test.py", line 8, in <module>
```

```
    print "dict['Age']: ", dict['Age'];
```

TypeError: 'type' object is unsubscriptable

Note – `del()` method is discussed in subsequent section.

7.1.6 Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys –

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example –

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}
```

```
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result –

```
dict['Name']: Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example –

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7}

print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result –

Traceback (most recent call last):

```
File "test.py", line 3, in <module>

    dict = {'Name': 'Zara', 'Age': 7};
```

TypeError: unhashable type: 'list'

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates –

Live Demo

```
#!/usr/bin/python

tup1 = (12, 34.56);

tup2 = ('abc', 'xyz');

# Following action is not valid for tuples

# tup1[0] = 100;

# So let's create a new tuple as follows

tup3 = tup1 + tup2;

print tup3;
```

When the above code is executed, it produces the following result –

(12, 34.56, 'abc', 'xyz')

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example –

Live Demo

```
#!/usr/bin/python

tup = ('physics', 'chemistry', 1997, 2000);

print tup;

del tup;

print "After deleting tup : ";

print tup;
```

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more –

```
('physics', 'chemistry', 1997, 2000)
```

After deleting tup :

Traceback (most recent call last):

```
File "test.py", line 9, in <module>
```

```
    print tup;
```

NameError: name 'tup' is not defined

7.2 DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

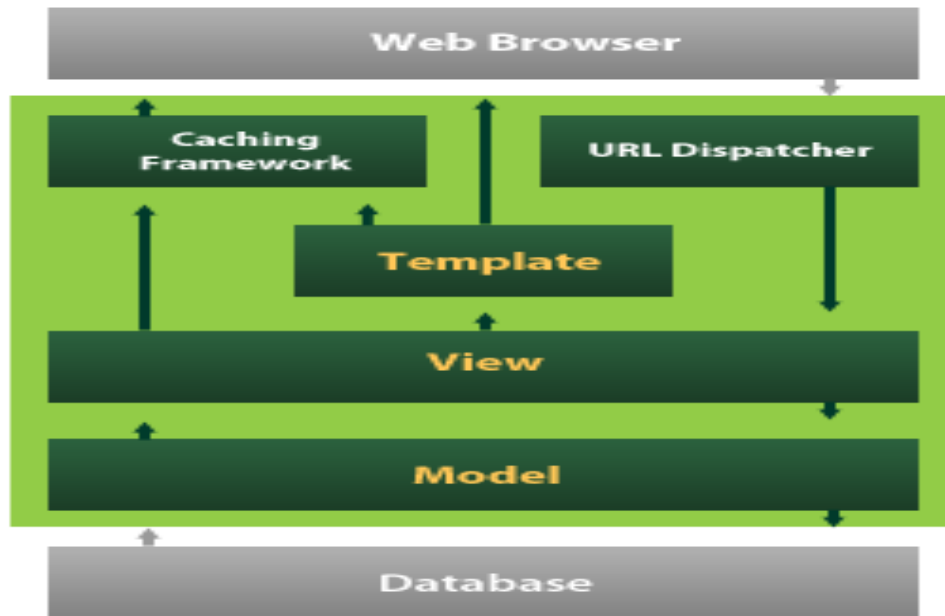


Fig 7.1 Data Base of Django

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

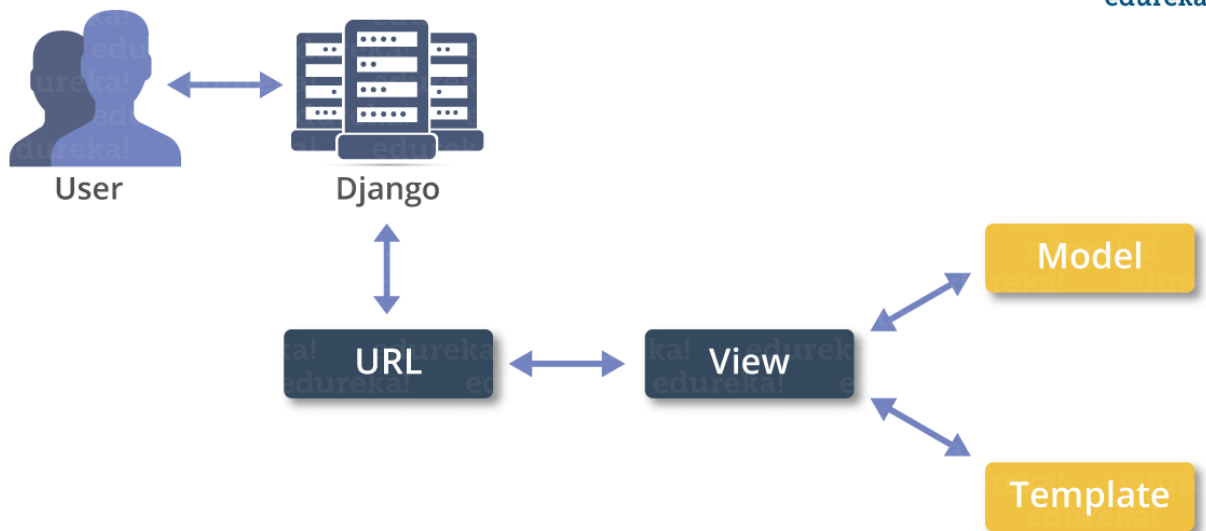


Fig 7.2 Flow chart of Django

7.2.1 Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code –

```
$ django-admin startproject myproject
```

This will create a "myproject" folder with the following structure –

```
myproject/
```

```
    manage.py
```

```
    myproject/
```

```
        __init__.py
```

```
        settings.py
```

```
        urls.py
```

```
        wsgi.py
```

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements –

manage.py – This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code –

```
$ python manage.py help
```

The “myproject” subfolder – This folder is the actual python package of your project. It contains four files –

__init__.py – Just for python, treat this folder as package.

settings.py – As the name indicates, your project settings.

urls.py – All links of your project and the function to call. A kind of ToC of your project.

wsgi.py – If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set –

```
DEBUG = True
```

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to 'True' for a live project. However, this has to be set to 'True' if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {  
  
    'default': {  
  
        'ENGINE': 'django.db.backends.sqlite3',  
  
        'NAME': 'database.sql',  
  
        'USER': '',  
  
        'PASSWORD': '',  
  
        'HOST': '',  
  
        'PORT': '',  
  
    }  
  
}
```

Database is set in the 'Database' dictionary. The example above is for SQLite engine. As stated earlier, Django also supports –

MySQL (django.db.backends.mysql)

PostgreSQL (django.db.backends.postgresql_psycopg2)

Oracle (django.db.backends.oracle) and NoSQL DB

MongoDB (django_mongodb_engine)

Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE...

Now that your project is created and configured make sure it's working –

```
$ python manage.py runserver
```

You will get something like the following on running the above code –

Validating models...

0 errors found

September 03, 2015 - 11:41:50

Django version 1.6.11, using settings 'myproject.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

7.2.2 Create an Application

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py –

```
$ python manage.py startapp myapp
```

You just created myapp application and like project, Django create a “myapp” folder with the application structure –

myapp/

 __init__.py

 admin.py

 models.py

 tests.py

 views.py

__init__.py – Just to make sure python handles this folder as a package.

admin.py – This file helps you make the app modifiable in the admin interface.

models.py – This is where all the application models are stored.

tests.py – This is where your unit tests are.

views.py – This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update `INSTALLED_APPS` tuple in the `settings.py` file of your project (add your app name) –

```
INSTALLED_APPS = (  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'myapp',  
)
```

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a `forms.py` file in `myapp` folder to contain our app forms. We will create a login form.

`myapp/forms.py`

```
#-*- coding: utf-8 -*-
```

```
from django import forms
```

```
class LoginForm(forms.Form):
```

```
    user = forms.CharField(max_length = 100)
```

```
    password = forms.CharField(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: `DateInput` for dates, `CheckboxInput` for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the `request.POST` dictionary.

Let's create a login view in our myapp/views.py –

```
#-*- coding: utf-8 -*-
```

```
from myapp.forms import LoginForm
```

```
def login(request):
```

```
    username = "not logged in"
```

```
    if request.method == "POST":
```

```
        #Get the posted form
```

```
        MyLoginForm = LoginForm(request.POST)
```

```
        if MyLoginForm.is_valid():
```

```
            username = MyLoginForm.cleaned_data['username']
```

```
    else:
```

```
        MyLoginForm = Loginform()
```

```
    return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

```
<html>
```

```
<body>
```

```
    <form name = "form" action = "{% url 'myapp.views.login' %}"
```

```
        method = "POST" >{% csrf_token %}
```

```
        <div style = "max-width:470px;">
```

```
            <center>
```

```
                <input type = "text" style = "margin-left:20%;"
```

```
                    placeholder = "Identifiant" name = "username" />
```

```
            </center>
```

```

</div>

<br>

<div style = "max-width:470px;">

    <center>

        <input type = "password" style = "margin-left:20%;"
            placeholder = "password" name = "password" />

    </center>

</div>

<br>

<div style = "max-width:470px;">

    <center>

        <button style = "border:0px; background-color:#4285F4; margin-top:8%;
            height:35px; width:80%;margin-left:19%;" type = "submit"
            value = "Login" >

            <strong>Login</strong>

        </button>

    </center>

</div>

</form>

</body>

</html>

```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

```
{% csrf_token %}
```

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

```
<html>

<body>

    You are : <strong>{{username}}</strong>

</body>

</html>
```

Now, we just need our pair of URLs to get started: myapp/urls.py

```
from django.conf.urls import patterns, url

from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',

    url(r'^connection/', TemplateView.as_view(template_name = 'login.html')),

    url(r'^login/', 'login', name = 'login'))
```

When accessing "/myapp/connection", we will get the following login.html template rendered –

Setting Up Sessions

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE_CLASSES and the INSTALLED_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE_CLASSES should have –

```
'django.contrib.sessions.middleware.SessionMiddleware'
```

And INSTALLED_APPS should have –

```
'django.contrib.sessions'
```

By default, Django saves session information in database (django_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won't see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first let's change our login view to save our username cookie server side –

```
def login(request):

    username = 'not logged in'

    if request.method == 'POST':

        MyLoginForm = LoginForm(request.POST)

        if MyLoginForm.is_valid():

            username = MyLoginForm.cleaned_data['username']

            request.session['username'] = username

        else:

            MyLoginForm = LoginForm()

    return render(request, 'loggedin.html', {"username" : username})
```

Then let us create formView view for the login form, where we won't display the form if cookie is set –

```
def formView(request):

    if request.session.has_key('username'):

        username = request.session['username']

        return render(request, 'loggedin.html', {"username" : username})

    else:

        return render(request, 'login.html', {})
```

Now let us change the url.py file to change the url so it pairs with our new view –

```
from django.conf.urls import patterns, url
```

```
from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',

    url(r'^connection/', 'formView', name = 'loginform'),

    url(r'^login/', 'login', name = 'login'))
```

When accessing /myapp/connection, you will get to see the following page

CHAPTER – 8

SYSTEM IMPLEMENTATION

8.1 MODULES:

- User
- Admin
- Data Preprocessing
- Machine Learning

MODULES DESCRIPTION:

8.1.1 User:

The User can register the first. While registering he required a valid user email and mobile for further communications. Once the user register then admin can activate the user. Once admin activated the user then user can login into our system. User can upload the dataset based on our dataset column matched. For algorithm execution data must be in float format. Here we took Dataset they used for this research is taken from the UCI Machine Learning Repository. for testing purpose. User can also add the new data for existing dataset based on our Django application. User can click the Classification in the web page so that the data calculated Accuracy based on the algorithms. User can click Prediction in the web page so that user can write the review after predict the review That will display results depends upon review like postive,negative or neutral.

8.1.2 Admin:

Admin can login with his login details. Admin can activate the registered users. Once he activate then only the user can login into our system. Admin can view the overall data in the browser. Admin can click the Results in the web page so calculated Accuracy based on the algorithms is displayed. All algorithms execution complete then admin can see the overall accuracy in web page.

8.1.3 Data Preprocessing:

A dataset can be viewed as a collection of data objects, which are often also called as a records, points, vectors, patterns, events, cases, samples, observations, or entities. Data objects are described by a number of features that capture the basic characteristics of an object, such as the mass of a physical object or the time at which an event occurred, etc. Features are often called as variables, characteristics, fields, attributes, or dimensions. The data preprocessing in this forecast uses techniques like removal of noise in the data, the expulsion of missing information, modifying default values if relevant and grouping of attributes for prediction at various levels.

8.1.4 Machine learning:

Based on the split criterion, the cleansed data is split into 60% training and 40% test, then the dataset is subjected to four machine learning classifiers such as Machine learning plays an important role in predicting diseases. algorithms named Support Vector Machine(SVM), Decision Tree(DT), Random Forest(RF), Logistic Regression(LR) and Naive Bayes(NB). The accuracy a of the classifiers was calculated and displayed in my results. The classifier which bags up the highest accuracy could be determined as the best classifier.

CHAPTER – 9

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

9.1 TYPES OF TESTS

9.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

9.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

9.1.3 Functional testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

9.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

9.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

9.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER – 10

INPUT AND OUTPUT DESIGN

10.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

10.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct

source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER-11

SOURCE CODE

```
User side views.py

from django.shortcuts import render, HttpResponseRedirect

from .forms import UserRegistrationForm

from django.contrib import messages

from .models import UserRegistrationModel

from django.conf import settings

import os

import pandas as pd

# Create your views here.

def UserRegisterActions(request):

    if request.method == 'POST':

        form = UserRegistrationForm(request.POST)

        if form.is_valid():

            print('Data is Valid')

            form.save()

            messages.success(request, 'You have been successfully registered')

            form = UserRegistrationForm()

            return render(request, 'UserRegistrations.html', {'form': form})

        else:

            messages.success(request, 'Email or Mobile Already Existed')

            print("Invalid form")

    else:
```

```

    form = UserRegistrationForm()

    return render(request, 'UserRegistrations.html', {'form': form})

def UserLoginCheck(request):

    if request.method == "POST":

        loginid = request.POST.get('loginid')

        pswd = request.POST.get('pswd')

        print("Login ID = ", loginid, ' Password = ', pswd)

        try:

            check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd)

            status = check.status

            print('Status is ', status)

            if status == "activated":

                request.session['id'] = check.id

                request.session['loggeduser'] = check.name

                request.session['loginid'] = loginid

                request.session['email'] = check.email

                print("User id At", check.id, status)

                return render(request, 'users/UserHome.html', {})

            else:

                messages.success(request, 'Your Account has not been activated by Admin.')

                return render(request, 'UserLogin.html')

        except Exception as e:

            print('Exception is ', str(e))

            pass

```

```

        messages.success(request, 'Invalid Login id and password')

    return render(request, 'UserLogin.html', {})

def UserHome(request):

    return render(request, 'users/UserHome.html', {})

def ViewDataSet(request):

    path = os.path.join(settings.MEDIA_ROOT, "full_dataset.csv")

    df = pd.read_csv(path)

    df = df.drop(

        ['on_thyroxine', 'query_on_thyroxine', 'on_antithyroid_medication', 'thyroid_surgery',
        'query_hypothyroid',

        'query_hyperthyroid'], axis=1)

    df = df.to_html

    return render(request, 'users/UsersViewData.html', {"data": df})

def UserMachineLearning(request):

    from .utility import ThyroidUtilities

    svc_accuracy,          svc_precision,          svc_recall,          svc_f1score          =
    ThyroidUtilities.calc_support_vector_classifier()

    dt_accuracy,          dt_precision,          dt_recall,          dt_f1score          =
    ThyroidUtilities.calc_decision_tree_classifier()

    rf_accuracy, rf_precision, rf_recall, rf_f1score = ThyroidUtilities.calc_random_forest()

    lg_accuracy, lg_precision, lg_recall, lg_f1score = ThyroidUtilities.calc_logistic_regression()

    nb_accuracy,          nb_precision,          nb_recall,          nb_f1score          =
    ThyroidUtilities.calc_naive_bayes_classifier()

    lg_dict = {'lg_accuracy': lg_accuracy, 'lg_precision': lg_precision, "lg_recall": lg_recall,

        'lg_f1score': lg_f1score}

    rf_dict = {'rf_accuracy': rf_accuracy, 'rf_precision': rf_precision, 'rf_recall': rf_recall,

```

```

        'rf_f1score': rf_f1score}

    svc_dict = {'svc_accuracy': svc_accuracy, 'svc_precision': svc_precision, 'svc_recall':
svc_recall,

        'svc_f1score': svc_f1score}

    nb_dict = {'nb_accuracy': nb_accuracy, 'nb_precision': nb_precision, 'nb_recall': nb_recall,

        'nb_f1score': nb_f1score}

    dt_dict = {'dt_accuracy': dt_accuracy, 'dt_precision': dt_precision, 'dt_recall': dt_recall,

        'dt_f1score': dt_f1score}

    return render(request, 'users/usermachinelearning.html',

        {'lg': lg_dict, 'rf': rf_dict, "svc": svc_dict, "gb": nb_dict, "dt": dt_dict, 'nb': nb_dict})

def user_predictions(request):

    if request.method == 'POST':

        age = int(request.POST.get('age'))

        sex = int(request.POST.get('sex'))

        tsh = float(request.POST.get('TSH'))

        t3 = float(request.POST.get('T3'))

        tt4 = float(request.POST.get('TT4'))

        t4u = float(request.POST.get('T4U'))

        fti = float(request.POST.get('FTI'))

        test_data = [age, sex, tsh, t3, tt4, t4u, fti]

        from .utility import ThyroidUtilities

        test_pred = ThyroidUtilities.test_user_date(test_data)

        if test_pred[0] == 0:

            rslt = False

```



```

else:

    rslt = True

    return render(request, "users/testform.html", {"test_data": test_data, "result": rslt})

else:

    return render(request, "users/testform.html", {})

```

ThyroidUtility.py

```

import pandas as pd

from sklearn.model_selection import train_test_split

from django.conf import settings

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score

from sklearn.metrics import accuracy_score

from sklearn.metrics import confusion_matrix

import os

path = os.path.join(settings.MEDIA_ROOT , "full_dataset.csv")

df = pd.read_csv(path)

df = df[['age','sex','TSH','T3','TT4','T4U','FTI','classes']]

X = df.iloc[:, :-1].values # independent variable

y = df.iloc[:, -1].values # Dependent variable

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.80, random_state=0)

def calc_logistic_regression():

    print("*"*25,"Logistic Regressions")

    from sklearn.linear_model import LogisticRegression

```

```

model = LogisticRegression()

model.fit(X_train, y_train) # Trained with 80% Data

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print('Accuracy:', accuracy)

cm = confusion_matrix(y_test, y_pred)

precision = precision_score(y_test, y_pred)

print('Precision Score:', precision)

recall = recall_score(y_test, y_pred)

print('Recall Score:', recall)

f1score = f1_score(y_test, y_pred)

print('F1-Score:', f1score)

return accuracy,precision,recall,f1score

def calc_random_forest():

    print("***25,Random Forest Classification")

    from sklearn.ensemble import RandomForestClassifier

    model = RandomForestClassifier()

    model.fit(X_train, y_train) # Trained with 80% Data

    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)

    print('RF Accuracy:', accuracy)

    cm = confusion_matrix(y_test, y_pred)

    precision = precision_score(y_test, y_pred)

    print('RF Precision Score:', precision)

```

```

recall = recall_score(y_test, y_pred)

print('RF Recall Score:', recall)

f1score = f1_score(y_test, y_pred)

print('RF F1-Score:', f1score)

return accuracy,precision,recall,f1score

def calc_support_vector_classifier():

    print("***25,\"SVM Classification\")

    from sklearn.svm import SVC

    model = SVC(kernel='rbf')

    model.fit(X_train, y_train) # Trained with 80% Data

    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)

    print('SVM Accuracy:', accuracy)

    cm = confusion_matrix(y_test, y_pred)

    precision = precision_score(y_test, y_pred)

    print('SVM Precision Score:', precision)

    recall = recall_score(y_test, y_pred)

    print('SVM Recall Score:', recall)

    f1score = f1_score(y_test, y_pred)

    print('SVM F1-Score:', f1score)

    return accuracy,precision,recall,f1score

def calc_decision_tree_classifier():

    print("***25,\"Decision Tree\")

    from sklearn.tree import DecisionTreeClassifier

```

```

model = DecisionTreeClassifier()

model.fit(X_train, y_train) # Trained with 80% Data

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print('GB Accuracy:', accuracy)

cm = confusion_matrix(y_test, y_pred)

precision = precision_score(y_test, y_pred)

print('GB Precision Score:', precision)

recall = recall_score(y_test, y_pred)

print('GB Recall Score:', recall)

f1score = f1_score(y_test, y_pred)

print('GB F1-Score:', f1score)

return accuracy,precision,recall,f1score

def calc_naive_bayes_classifier():

    print("***25,\"Naive Bayes\"")

    from sklearn.naive_bayes import GaussianNB

    model = GaussianNB()

    model.fit(X_train, y_train) # Trained with 80% Data

    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)

    print('NB Accuracy:', accuracy)

    cm = confusion_matrix(y_test, y_pred)

    precision = precision_score(y_test, y_pred)

    print('NB Precision Score:', precision)

```

```

recall = recall_score(y_test, y_pred)

print('NB Recall Score:', recall)

f1score = f1_score(y_test, y_pred)

print('NB F1-Score:', f1score)

return accuracy,precision,recall,f1score

def test_user_date(test_features):

    print(test_features)

    from sklearn.ensemble import RandomForestClassifier

    model = RandomForestClassifier()

    model.fit(X_train, y_train)

    test_pred = model.predict([test_features])

    return test_pred

base.html

{%load static %}

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>*Hypothyroid Classification*</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

<link href="{%static 'css/style.css'%}" rel="stylesheet" type="text/css" />

<link href="{%static 'css/menu.css'%}" rel="stylesheet" type="text/css" />

<!--[if lt IE 7]>

<style type="text/css" media="screen">

```

```

#menuh{float:none;}

body{behavior:url(csshover.htc); font-size:75%;}

#menuh ul li{float:left; width: 100%;}

#menuh a{height:1%;font:normal 1em/1.6em helvetica, "Trebuchet MS", arial, sans-serif;}

</style>

<![endif]-->

</head>

<body>

<div id="top">

    <h2><a href="#">Hypothyroid</a> | <a href="#">Classification</a></h2>

</div>

<div id="banner">

    <h1> <a href="#">Prediction Of Thyroid Disease(Hypothyroid) In Early Stage Using Feature
    Selection And Classification Techniques</a></h1>

</div>

<div id="menuh-container">

    <div id="menuh">

        <ul><li><a href="{%url 'index%}" class="top_parent">Homepage</a> </li></ul>

        <ul><li><a href="{%url 'UserLogin%}" class="top_parent">Users</a></li></ul>

        <ul><li><a href="{%url 'AdminLogin%}" class="top_parent">Admin</a></li></ul>

        <ul><li><a href="{%url 'UserRegister%}" class="top_parent">Register</a> </li></ul>

    </div>

</div>

{%block contents%}

```



```

<tr><td></td>

    <td>Password</td>

    <td>{{form.password}}</td>

</tr>

<tr><td></td>

    <td>Mobile</td>

    <td>{{form.mobile}}</td>

</tr>

<tr><td></td>

    <td>email</td>

    <td>{{form.email}}</td>

</tr>

<tr><td></td>

    <td>Locality</td>

    <td>{{form.locality}}</td>

</tr>

<tr><td></td>

    <td>Address</td>

    <td>{{form.address}}</td>

</tr>

<tr><td></td>

    <td>City</td>

    <td>{{form.city}}</td>

</tr>

```



```

<tr><td></td>

    <td>State</td>

    <td>{{ form.state }}</td>

</tr>

<tr><td></td>

    <td></td>

    <td>{{ form.status }}</td>

</tr>

<tr><td></td>

    <td></td>

    <td>
        <button class="btn btn-block btn-primary text-white py-3 px-5"
type="submit">Register </button></td>

</tr>

<tr>

    <td>

        <div class="form-group mt-3">

            <span >&nbsp;</span>

        </div>

    </td>

</tr>

{% if messages %}

{% for message in messages %}

<font color='GREEN'> {{ message }}</font>

{% endfor %}

```



```

        <th>Activate</th>

    </tr>

</thead>

<tbody>

{% for i in data %}

        <tr style="color: Black">

            <td>{{forloop.counter}}</td>

            <td>{{i.name}}</td>

            <td>{{i.loginid}}</td>

            <td>{{i.mobile}}</td>

            <td>{{i.email}}</td>

            <td>{{i.locality}}</td>

            <td>{{i.status}}</td>

            {% if i.status == 'waiting' %}

                <td><a class="btn-link" href="/AdminActivaUsers/?uid={{ i.id

            }}"

                style="color:DARKBLUE">Activate</a></td>

            {% else %}

                <td> Activated</td>

            {% endif %}

        </tr>

    {% endfor %}

</tbody>

</table>

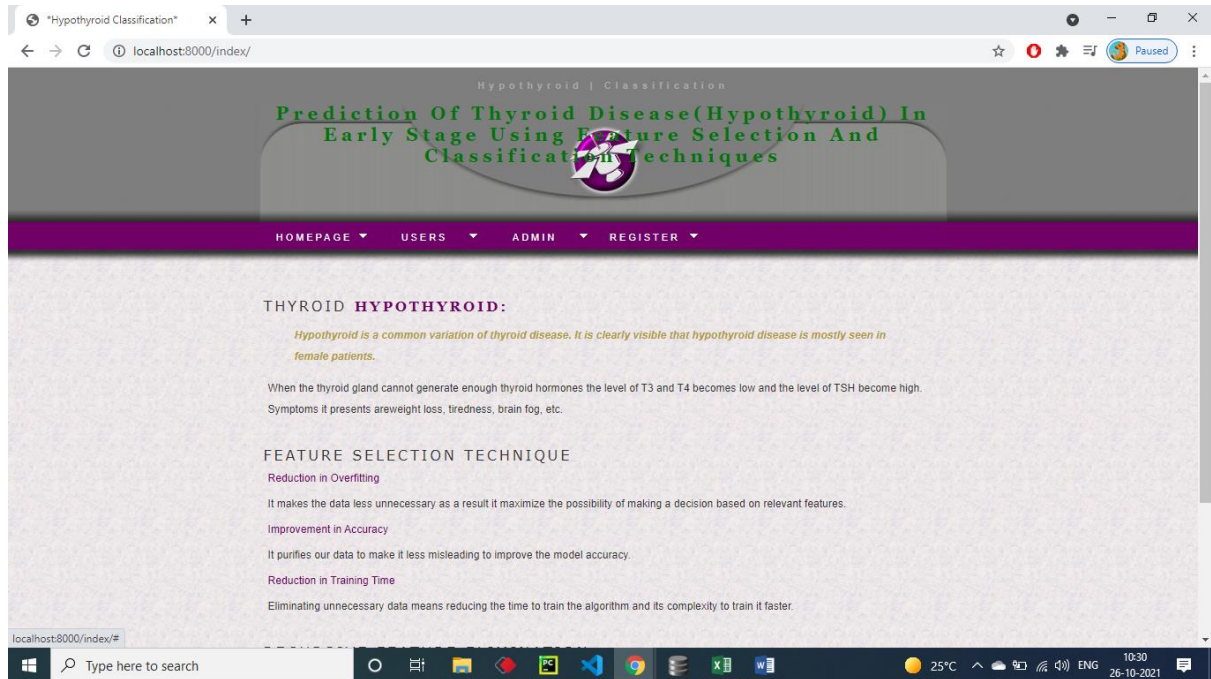
```

```
{%endblock%}
```

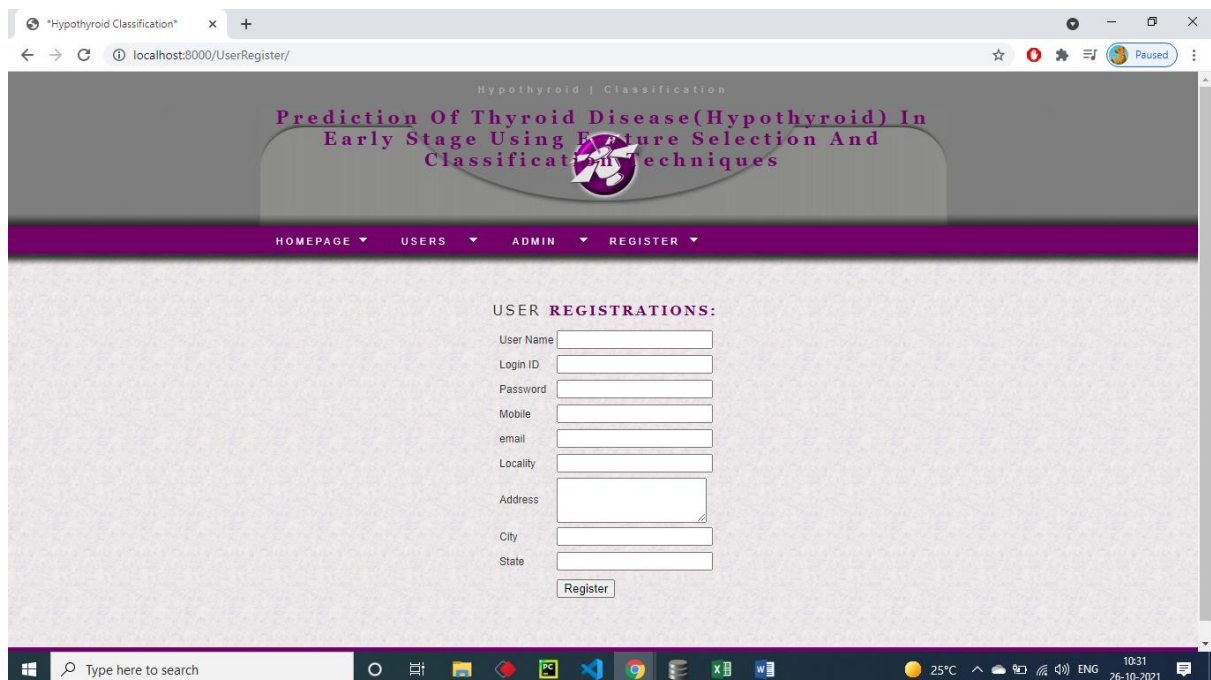
CHAPTER-12

SCREENSHOTS

12.1 Home Page



12.2 User Register Form



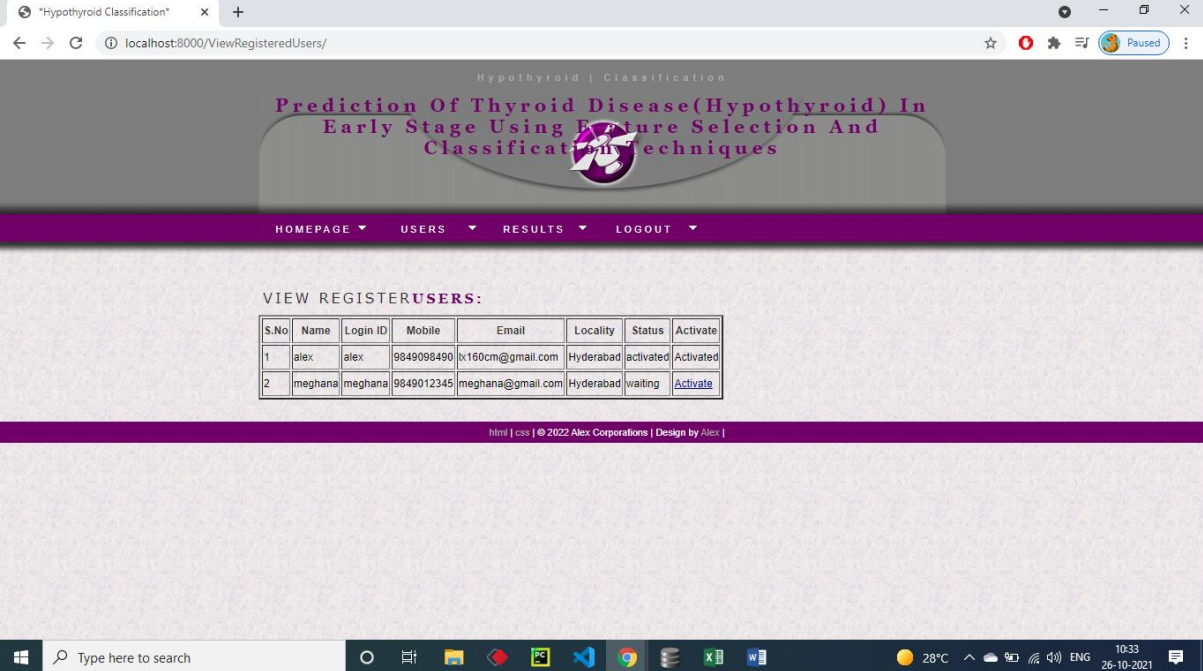
12.3 Admin Login Form

The screenshot shows a web browser window with the address bar displaying "localhost:8000/AdminLogin/". The page title is "Hypothyroid | Classification". The main heading is "Prediction Of Thyroid Disease(Hypothyroid) In Early Stage Using Feature Selection And Classification Techniques". Below the heading is a navigation bar with links: "HOMEPAGE", "USERS", "ADMIN", and "REGISTER". The main content area is titled "ADMIN LOGIN FORM:" and contains two input fields: "Enter Login Id" and "Enter password", followed by a "Login" button. At the bottom of the page, there is a footer that reads "html | css | © 2022 Alex Corporations | Design by Alex |". The Windows taskbar at the bottom shows the search bar, task view, and several application icons, along with system information like temperature (25°C) and time (10:31, 26-10-2021).

12.4 Admin Home Page

The screenshot shows a web browser window with the address bar displaying "localhost:8000/AdminLoginCheck/". The page title is "Hypothyroid | Classification". The main heading is "Prediction Of Thyroid Disease(Hypothyroid) In Early Stage Using Feature Selection And Classification Techniques". Below the heading is a navigation bar with links: "HOMEPAGE", "USERS", "RESULTS", and "LOGOUT". The main content area is titled "THYROID HYPOTHYROID:" and contains a paragraph of text: "Hypothyroid is a common variation of thyroid disease. It is clearly visible that hypothyroid disease is mostly seen in female patients." Below this, there is a section titled "FEATURE SELECTION TECHNIQUE" with three sub-points: "Reduction in Overfitting", "Improvement in Accuracy", and "Reduction in Training Time". The Windows taskbar at the bottom shows the search bar, task view, and several application icons, along with system information like temperature (28°C) and time (10:32, 26-10-2021).

12.5 View users and Activate



Prediction Of Thyroid Disease(Hypothyroid) In Early Stage Using Feature Selection And Classification Techniques

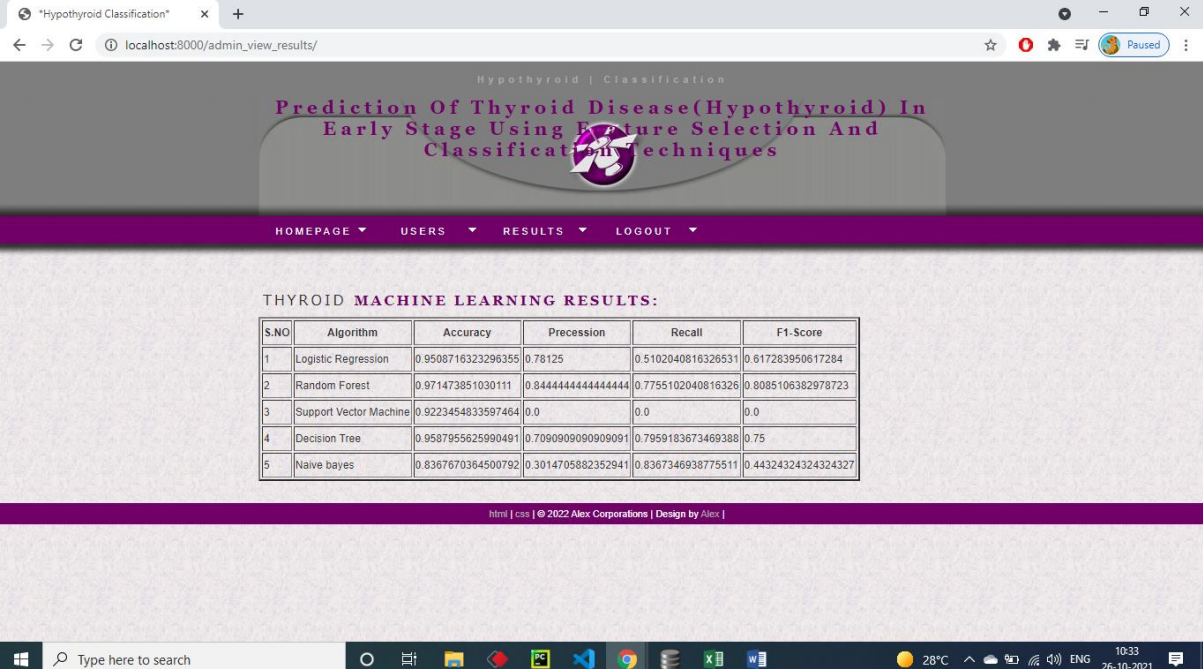
HOMEPAGE ▾ USERS ▾ RESULTS ▾ LOGOUT ▾

VIEW REGISTERED USERS:

S.No	Name	Login ID	Mobile	Email	Locality	Status	Activate
1	alex	alex	9849098490	alex160cm@gmail.com	Hyderabad	activated	Activated
2	meghana	meghana	9849012345	meghana@gmail.com	Hyderabad	waiting	Activate

html | css | © 2022 Alex Corporations | Design by Alex |

12.6 Admin view Results



Prediction Of Thyroid Disease(Hypothyroid) In Early Stage Using Feature Selection And Classification Techniques

HOMEPAGE ▾ USERS ▾ RESULTS ▾ LOGOUT ▾

THYROID MACHINE LEARNING RESULTS:

S.NO	Algorithm	Accuracy	Precision	Recall	F1-Score
1	Logistic Regression	0.9508716323296355	0.78125	0.5102040816326531	0.617283950617284
2	Random Forest	0.971473851030111	0.8444444444444444	0.7755102040816326	0.8085106382978723
3	Support Vector Machine	0.9223454833597464	0.0	0.0	0.0
4	Decision Tree	0.9587955625990491	0.7090909090909091	0.7959183673469388	0.75
5	Naive Bayes	0.8367670364500792	0.3014705882352941	0.8367346938775511	0.44324324324324327

html | css | © 2022 Alex Corporations | Design by Alex |

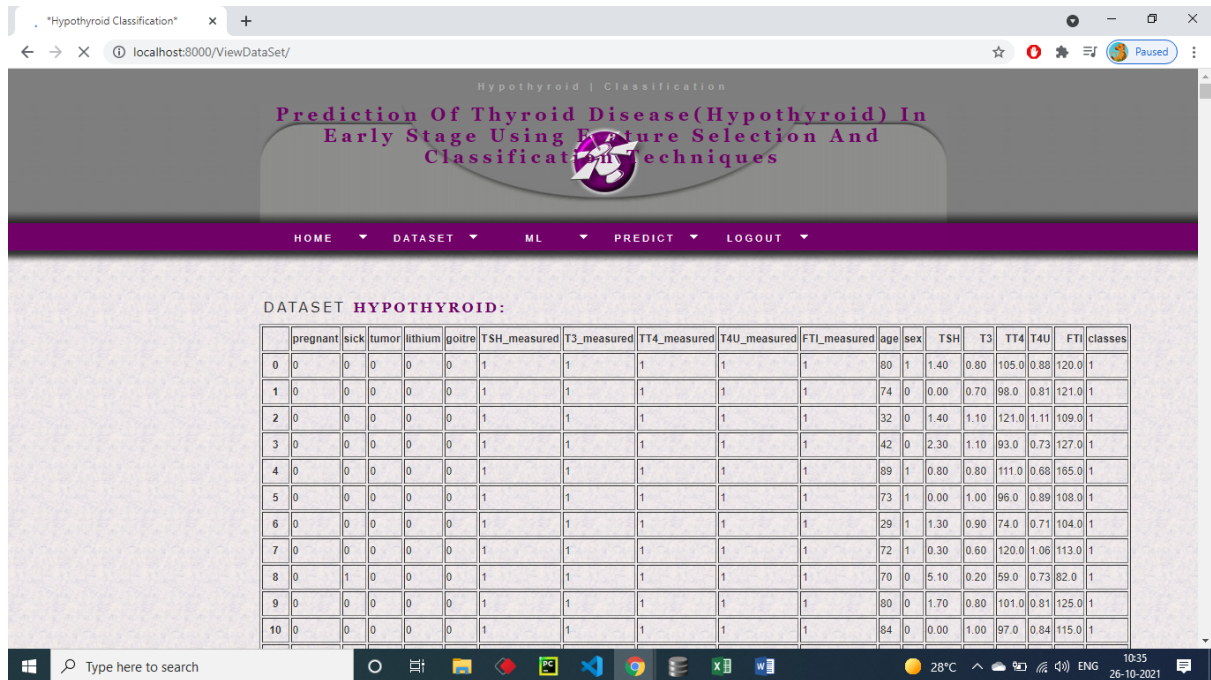
12.7 User Login Form

The screenshot shows a web browser window with the title "Hypothyroid Classification". The address bar displays "localhost:8000/UserLogin/". The page features a header with the title "Prediction Of Thyroid Disease(Hypothyroid) In Early Stage Using Feature Selection And Classification Techniques" and a navigation menu with links: HOME PAGE, USERS, ADMIN, and REGISTER. The main content area is titled "USER LOGIN FORM:" and contains two input fields: "Enter Login Id" and "Enter password", followed by a "Login" button. The footer of the page reads "html | css | © 2022 Alex Corporations | Design by Alex |". The Windows taskbar at the bottom shows the search bar and various application icons.

12.8 User Home Page

The screenshot shows a web browser window with the title "Hypothyroid Classification". The address bar displays "localhost:8000/UserLoginCheck/". The page features a header with the title "Prediction Of Thyroid Disease(Hypothyroid) In Early Stage Using Feature Selection And Classification Techniques" and a navigation menu with links: HOME, DATASET, ML, PREDICT, and LOGOUT. The main content area is titled "THYROID HYPOTHYROID:" and contains a paragraph of text: "Hypothyroid is a common variation of thyroid disease. It is clearly visible that hypothyroid disease is mostly seen in female patients." Below this, there is a section titled "FEATURE SELECTION TECHNIQUE" with three sub-sections: "Reduction in Overfitting", "Improvement in Accuracy", and "Reduction in Training Time". The Windows taskbar at the bottom shows the search bar and various application icons.

12.9 User View Dataset

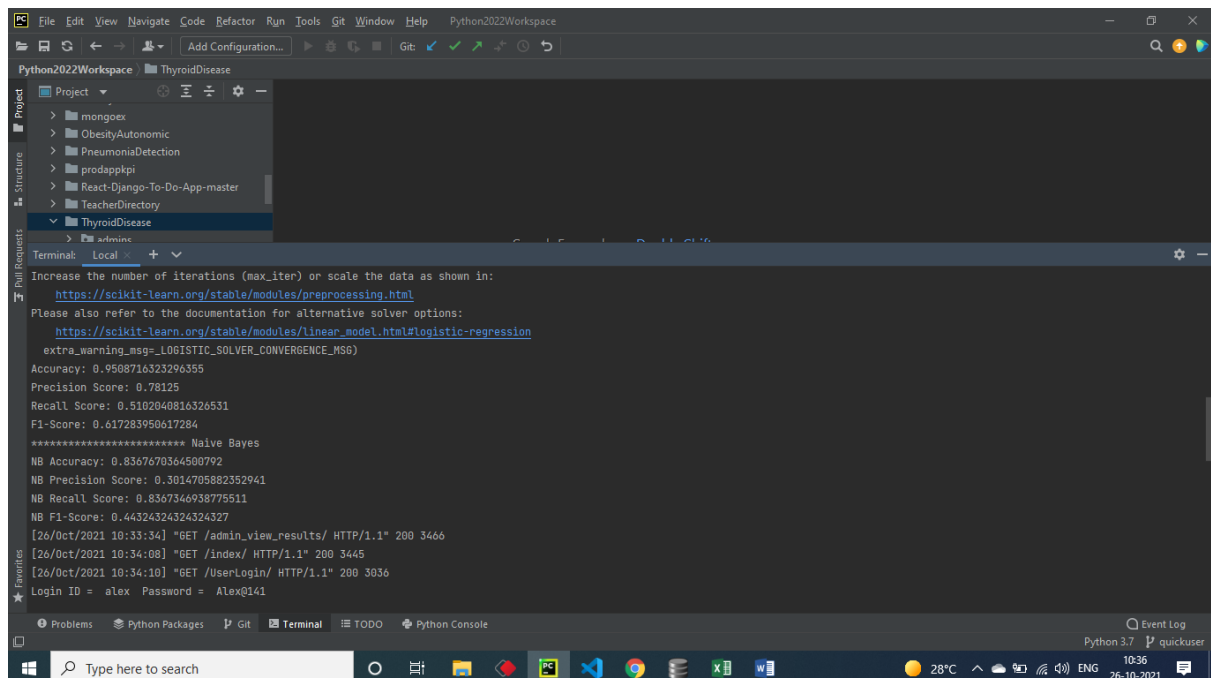


Prediction Of Thyroid Disease(Hypothyroid) In Early Stage Using Feature Selection And Classification Techniques

DATASET HYPOTHYROID:

	pregnant	sick	tumor	lithium	goitre	TSH_measured	T3_measured	TT4_measured	T4U_measured	FTI_measured	age	sex	TSH	T3	TT4	T4U	FTI	classes
0	0	0	0	0	0	1	1	1	1	1	80	1	1.40	0.80	105.0	0.88	120.0	1
1	0	0	0	0	0	1	1	1	1	1	74	0	0.00	0.70	98.0	0.81	121.0	1
2	0	0	0	0	0	1	1	1	1	1	32	0	1.40	1.10	121.0	1.11	109.0	1
3	0	0	0	0	0	1	1	1	1	1	42	0	2.30	1.10	93.0	0.73	127.0	1
4	0	0	0	0	0	1	1	1	1	1	89	1	0.80	0.80	111.0	0.68	165.0	1
5	0	0	0	0	0	1	1	1	1	1	73	1	0.00	1.00	96.0	0.89	108.0	1
6	0	0	0	0	0	1	1	1	1	1	29	1	1.30	0.90	74.0	0.71	104.0	1
7	0	0	0	0	0	1	1	1	1	1	72	1	0.30	0.60	120.0	1.06	113.0	1
8	0	1	0	0	0	1	1	1	1	1	70	0	5.10	0.20	59.0	0.73	82.0	1
9	0	0	0	0	0	1	1	1	1	1	80	0	1.70	0.80	101.0	0.81	125.0	1
10	0	0	0	0	0	1	1	1	1	1	84	0	0.00	1.00	97.0	0.84	115.0	1

12.10 Server Side Results



```
File Edit View Navigate Code Refactor Run Tools Git Window Help Python2022Workspace
Python2022Workspace / ThyroidDisease
Project
  > mongoex
  > ObesityAutonomic
  > PneumoniaDetection
  > prodappkpi
  > React-Django-To-Do-App-master
  > TeacherDirectory
  > ThyroidDisease
  > adminic
Terminal: Local x +
Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=LOGISTIC_SOLVER_CONVERGENCE_MSG)
Accuracy: 0.9508716323296355
Precision Score: 0.78125
Recall Score: 0.5102040816326531
F1-Score: 0.617283950617284
***** Naive Bayes
NB Accuracy: 0.8367670364500792
NB Precision Score: 0.3014705882352941
NB Recall Score: 0.8367346938775511
NB F1-Score: 0.44324324324324327
[26/Oct/2021 10:33:34] "GET /admin_view_results/ HTTP/1.1" 200 3466
[26/Oct/2021 10:34:08] "GET /index/ HTTP/1.1" 200 3445
[26/Oct/2021 10:34:10] "GET /UserLogin/ HTTP/1.1" 200 3036
Login ID = alex Password = Alex@141
Problems Python Packages Git Terminal TODO Python Console
Type here to search 28°C 10:36 26-10-2021
```

12.11 User View Accuracy

The screenshot shows a web browser window with the address bar displaying "localhost:8000/UserMachineLearning/". The page title is "Hypothyroid | Classification". The main heading is "Prediction Of Thyroid Disease(Hypothyroid) In Early Stage Using Feature Selection And Classification Techniques". Below the heading is a navigation bar with links: HOME, DATASET, ML, PREDICT, and LOGOUT. The main content area displays "THYROID MACHINE LEARNING RESULTS:" followed by a table with 5 columns: S.NO, Algorithm, Accuracy, Precession, Recall, and F1-Score. The table contains 5 rows of data for different machine learning algorithms. At the bottom of the page, there is a footer with the text "html | css | © 2022 Alex Corporations | Design by Alex |".

S.NO	Algorithm	Accuracy	Precession	Recall	F1-Score
1	Logistic Regression	0.9508716323296355	0.78125	0.5102040816326531	0.617283950617284
2	Random Forest	0.971473851030111	0.8444444444444444	0.7755102040816326	0.8085106382978723
3	Support Vector Machine	0.9223454833597464	0.0	0.0	0.0
4	Decision Tree	0.9572107765451664	0.72	0.7346938775510204	0.7272727272727272
5	Naive Bayes	0.8367670364500792	0.3014705882352941	0.8367346938775511	0.44324324324324327

12.12 Prediction Form

The screenshot shows a web browser window with the address bar displaying "localhost:8000/user_predictions/". The page title is "Hypothyroid | Classification". The main heading is "Prediction Of Thyroid Disease(Hypothyroid) In Early Stage Using Feature Selection And Classification Techniques". Below the heading is a navigation bar with links: HOME, DATASET, ML, PREDICT, and LOGOUT. The main content area displays "THYROID TEST FORM:" followed by "GIVEN DATA AND RESULT IS". Below this text are input fields for Age, Sex, TSH, T3, TT4, T4U, and FTI. A "Submit" button is located below the input fields. At the bottom of the page, there is a footer with the text "html | css | © 2022 Alex Corporations | Design by Alex |".

12.13 Prediction Results

Hypothyroid | Classification

Prediction Of Thyroid Disease(Hypothyroid) In Early Stage Using Feature Selection And Classification Techniques

HOME DATASET ML PREDICT LOGOUT

THYROID TEST FORM:

GIVEN DATA [49, 0, 1.25, 4.21, 2.5, 5.2, 0.25] AND RESULT IS FALSE

Age

Sex

TSH

T3

TT4

T4U

FTI

html | css | © 2022 Alex Corporations | Design by Alex |

CHAPTER – 13

SAMPLE TEST CASES

S.no	Test Case	Excepted Result	Result	Remarks(IF Fails)
1	User Register	If User registration successfully.	Pass	If already user email exist then it fails.
2	User Login	If Username and password is correct then it will getting valid page.	Pass	Un Register Users will not logged in.
3	Support vector machine(SVM)	The request will be accepted by the svm	Pass	The request will be accepted by svm other wise its failed
4	Random forest (rf)results	RF results calculated and displayed	Pass	Data is consider for testing
5	Decision tree results	Decision tree results calculated and displayed	Pass	<u>data is consider for testing</u>
6	Logistic regression(LR)	LR results calculated and displayed	pass	<u>Data is consider for testing</u>
7	Naïve bayes(NB)	NB results calculated and displayed	pass	<u>Data is consider for testing</u>
8	View dataset by user	Data set will be displayed by the user	Pass	Results not true failed
9	Calculate accuracy	Accuracy score calculated	Pass	Accuracy score not displayed failed
10	Prediction form	If Test results true	pass	If test results not true failed

11	Admin login	Admin can login with his login credential. If success he get his home page	Pass	Invalid login details will not allowed here
12	Admin can activate the register users	Admin can activate the register user id	Pass	If user id not found then it won't login.

CHAPTER – 14

FURTHER ENCHANCEMENT

Proposed method has not taken this data set for thyroid prediction; it will consider in future work and measure accuracy using decision tree and kNN. Hence, according to the data set which is used in this work, the accuracy obtained is satisfactory. The current scenario is of the developing of the model that help in the various sectors of life using the machine learning. The availability of data and its generation day by day increased a chance for the computer scientists to make prediction and analysis on such data sets that make the human life better and comfort. This study is concern with this motivation. The prediction and classification of any data depends on the data set itself and the various algorithms that are used.

CHAPTER – 15

CONCLUSION

We see that the feature selection technique RFE helps us to get better accuracy with all other classifiers. In our findings, we have seen that RFE significantly helps us to predict hypothyroid in the primary stage by using a real-time dataset. It is very difficult for us to collect data in this current pandemic situation. As a result, we have collected only 519 data. So, considering the situation and the constraint we couldn't study on a larger dataset. In our study, we have seen that there have not been done any work in thyroid based on Bangladesh before. We have a limitation of data to work with. So, in the future, we want to work with a larger dataset and we hope that more people from our country will show interest to work on this disease that will help us to find a better solution and able to predict disease in the primary stage with better accuracy. Hope that will help the people of our country to maintain a healthy society.

CHAPTER – 16

REFERENCES

- [1] A. M. Amiri, and G. Armano, “Early Diagnosis of Heart Disease Using Classification And Regression Trees”, In The 2013 International Joint Conference on Neural Networks, pp. 1-4, 09 January, 2014.
- [2] A. K. Aswathi, and A. Antony, “An Intelligent System for Thyroid Disease Classification and Diagnosis”, 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018), pp. 1261-1264, 27 September, 2018.
- [3] A. Begum, and A. Parkavi, “Prediction of thyroid Disease Using Data Mining Techniques”, 5th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 342-345, 06 June, 2019.
- [4] K. Pavya, and B. Srinivasan, “FEATURE SELECTION ALGORITHMS TO IMPROVE THYROID DISEASE DIAGNOSIS”, IEEE International Conference on Innovations in Green Energy and Healthcare Technologies(ICIGEHT’17), pp. 1-5, 02 November, 2017.
- [5] F. Saiti, A. A. Naini, M. A. Shoorehdeli, and M. Teshnehlab, “Thyroid Disease Diagnosis Based on Genetic Algorithms using PNN and SVM”, 3rd International Conference on Bioinformatics and Biomedical Engineering, pp. 1-4, 14 July, 2009.
- [6] Q. Pan, , Y. Zhang, M. Zuo, L. Xiang, and D. Chen, “Improved Ensemble Classification Method of Thyroid Disease Based on Random Forest”, 8th International Conference on Information Technology in Medicine and Education, pp 567-571, 13 July, 2017.
- [7] A. Tyagi, R. Mehra, and A. Saxena, “Interactive Thyroid Disease Prediction System Using Machine Learning Technique”, 5th IEEE International Conference on Parallel, Distributed and Grid Computing(PDGC-2018), pp 689-693, 27 June, 2019.
- [8] S. Dash, M. N. Das, and B. K. Mishra, “Implementation of an Optimized Classification Model for Prediction of Hypothyroid Disease Risks”, International Conference on Inventive Computation Technologies(ICICT) ,pp. 1-4, 19 January, 2017.
- [9] V. S. Vairale, and S. Shukla, “Classification of Hypothyroid Disorder using Optimized SVM Method”, Second International Conference on Smart Systems and Inventive Technology (ICSSIT 2019), pp. 258-263, 10 February, 2020.
- [10] K. Shankar, S. K. Lakshmanaprabu, D. Gupta, A. Maseleno, V. H. C.

D. Albuquerque, “Optimal feature-based multi-kernel SVM approach for thyroid disease classification”, Springer Science +Business Media, LLC, part of Springer Nature 2018, pp. 1128-1143, 2 July, 2018.

[11] M. R. N. Kousarrizi, F. Seiti, and M. Teshnehlab, “An Experimental Comparative Study on Thyroid Disease Diagnosis Based on Feature Subset Selection and classification”, International Journal of Electrical & Computer Sciences IJECS-IJENS, pp. 13-19, February, 2012.

[12] S. Bashir, Z. S. Khan, F. H. Khan, A. Anjum, and K. Bashir, “Improving Heart Disease Prediction Using Feature Selection Approaches”, 16th International Bhurban Conference on Applied Sciences & Technology (IBCAST), pp. 619-623, 18 March, 2019.

[13] P. Duggal, and S. Shukla, “Prediction Of Thyroid Disorders Using Advanced Machine Learning Techniques”, 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), pp. 670-675, 09 April 2020.

[14] Dhaka Tribune(2018), 50 million people suffer from thyroid disease in Bangladesh. Available: <https://www.dhakatribune.com/feature/healthwellness/2018/05/25/experts-50-million-people-suffer-from-thyroiddisease-in-bangladesh/>.