A Thesis/Project/Dissertation Report

on

MY DOCTOR : THE DISEASE PREDICTION SYSTEM

Submitted in partial fulfillment of the

requirement for the award of the degree of

B. Tech – (CSE)

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Mr. Abhay Kumar
Associate Professor

Submitted By--

ANIKET TEWARI

19SCSE1010615, 19021011772

ATUL YADAV

19SCSE1010495, 19021011669

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING /
DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA

DECEMBER – 2021.

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled "**MY DOCTOR : THE DISEASE PREDICTION SYSTEM**" in partial fulfillment of the requirements for the award of the **Bachelor of Technology (B. Tech)** submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of December, Year 2021, under the supervision of Mr. Abhay Kumar Designation, Associate Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

ANIKET TEWARI – 19SCSE1010615

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr. Abhay Kumar

Associate Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **ANIKET TEWARI – 19SCSE1010615** has been held on December, 2021 and his/her work is recommended for the award of Bachelor of Technology (B. Tech – CSE).

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021

Place: Greater Noida

STATEMENT OF PROJECT REPORT PREPARATION

1. Thesis Title— My Doctor : The Disease Prediction System.
2. Degree for which the report is submitted— Bachelor of Technology (B.Tech – CSE).
3. Project Supervisor was referred to for preparing the report— Ms. Abhay Kumar
4. Specifications regarding the thesis / report format have been closely followed.
5. The contents of the thesis have been organised based on the guidelines provided.
6. The report has been prepared without resorting to the plagiarism.
7. All the resources used have been cited appropriately.
8. The report has not been submitted elsewhere for any consideration / a degree.

Signature of the Student—

Aniket Tewari

Name—Aniket Tewari
Admission No.—19SCSE1010615
Enrollment No.—19021011772.

ACKNOWLEDGEMENT

I would like to express my special thank you to my Guide
“Mr. Abhay Kumar Sir” *and my*
Vice Chancellor ***“Mrs. Preeti Bajaj Mam”*** *who gave me*
the golden opportunity to work on this wonderful project on the topic ***“My Doctor: The Disease Prediction System”****, which helped me in doing a lot of Research in this field and I*
came to know about various new things and facts. I am really thankful to them.

Secondly, but not the least I would like to thank my parents and as well as my friends who helped me a lot in
finalizing this superb project within the limited time frame.

Date—21/12/2021

Name—*Aniket Tewari*
Admission No.—19SCSE1010615*
Enrollment No.—19021011772*
Sem—5*
Section—*CSE-1.*

Abstract

This Project is about developing a Machine Learning (ML) Model that predicts the type of disease that a person has. This model is completely based on Machine Learning and is built using Machine Learning Algorithms only.

In this fast growing and busy life, the person has to go to doctor if he / she feels any problem. May be the person is busy, he / she is not able to get appointment of the doctor, the clinic is very far away, etc. are some of the reasons that stuck the person in his / her meeting with the doctor. In order to solve this problem, this ML Model is of great use.

This ML Model works in such a way that the person enters all his details or simply chooses the accurate value and feeds it into the model, and based on the input, the model predicts the output, i.e., the type of disease from which the person is suffering. This solves all the problems which are mentioned above and the patient gets the 100% accurate result.

This model is built using Machine Learning Algorithms. This is a Supervised Learning Model as we will train the model using the training dataset. It mainly uses classification approach as it predicts only the class in which the disease is falling.
So, in a way this model is completely built in Python Programming Language as Machine Learning Algorithm.

The model gives result as the type of disease from which the person is suffering based on the inputs. Suppose, the person enters his Blood Pressure value, and all that, the model predicts whether the person has BP or not.

So, in a way, we can say that this model is very useful and is of great use to all the patients who are health conscious. One can get his / her results by simply sitting in homes without going anywhere.
In future, many other functionalities will also be added that makes the model pretty good as well.

Table of Contents

List of Figures

Acronyms

ML	Machine Learning
SML	Supervised Machine Learning
UML	Unsupervised Machine Learning
ER	Entity – Relationship Diagram
LR	Logistic Regression
AD	Activity Diagram
UCD	Use – Case Diagram.

CHAPTER-1

Introduction

This Project is about developing a Machine Learning Model which predicts the type of disease that a person has.

Problem Statement--

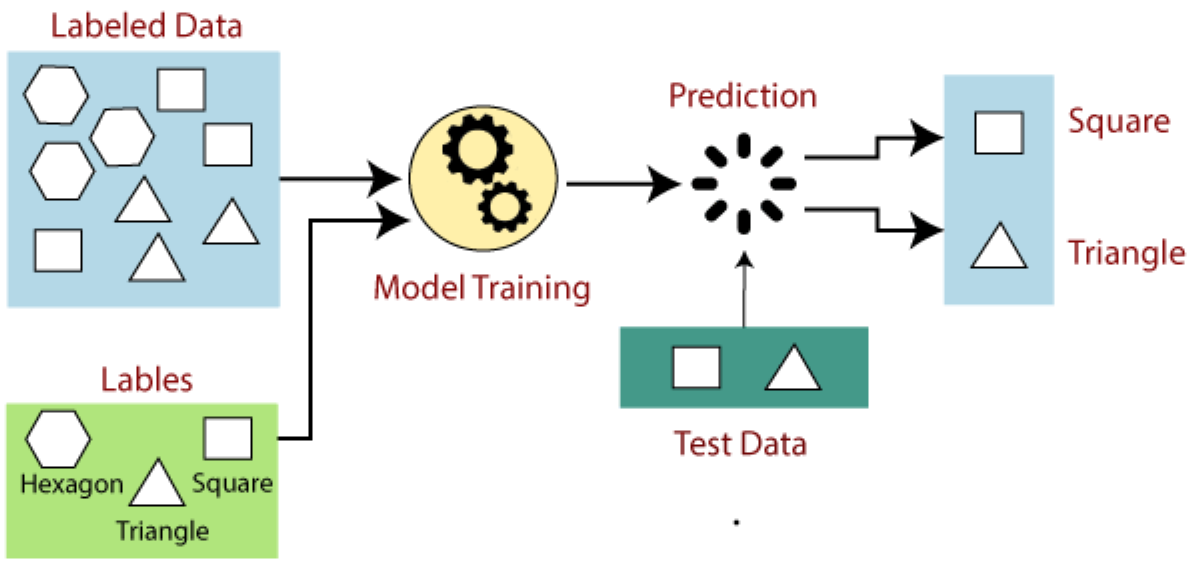
In this busy life, people may not get the time in order to meet their health requirements. There may be several reasons for this—

1. The person is not getting the appointment.
2. The Doctor is not available.
3. The clinic is too far, etc. and many more.

In such a case this model is very helpful in getting the accurate results. The user simply enters his / her details, and based on the input provided the model predicts the output. It gives very accurate results.

Tools and Technology Used--

In order to successfully build this model, we use the Machine Learning Algorithms. The main algorithm is the Supervised Machine Learning Algorithm in which we train the model using the well labeled dataset. The dataset includes the values ranging from a starting value to the highest value, the symptoms and the type of the disease. We will divide the dataset into three parts namely, train, test and validation sets. We will train the model using the training dataset, and then validate it using validation dataset, and finally test the model using the test dataset to check whether the model is giving the accurate results or not. This model basically follows the classification approach as it predicts the class of the disease. It tells from which disease, the person is suffering.



<u>Figure-1</u>—Supervised Machine Learning.

After successfully building the model, we will deploy it and then the user enters his / her details and get the predictions and the accurate results regarding his / her disease.

<u>Functions—</u>

The model has various functionalities in which some important one are listed below—

1. The model predicts the class of the disease from which a person is suffering.
2. The model takes input as the values of health parameters like blood pressure level, sugar level, etc.
3. The model predicts the accurate results.
4. The model is trained using well – labeled dataset which is very large and is tested also using the test dataset.

Requirements—

In order to successfully build this model, we use the Machine Learning Algorithms. The main algorithm is the Supervised Machine Learning Algorithm in which we train the model using the well labeled dataset. The dataset includes the values ranging from a starting value to the highest value, the symptoms and the type of the disease. We will divide the dataset into three parts namely, train, test and validation sets. We will train the model using the training dataset, and then validate it using validation dataset, and finally test the model using the test dataset to check whether the model is giving the accurate results or not. This model basically follows the classification approach as it predicts the class of the disease. It tells from which disease, the person is suffering.

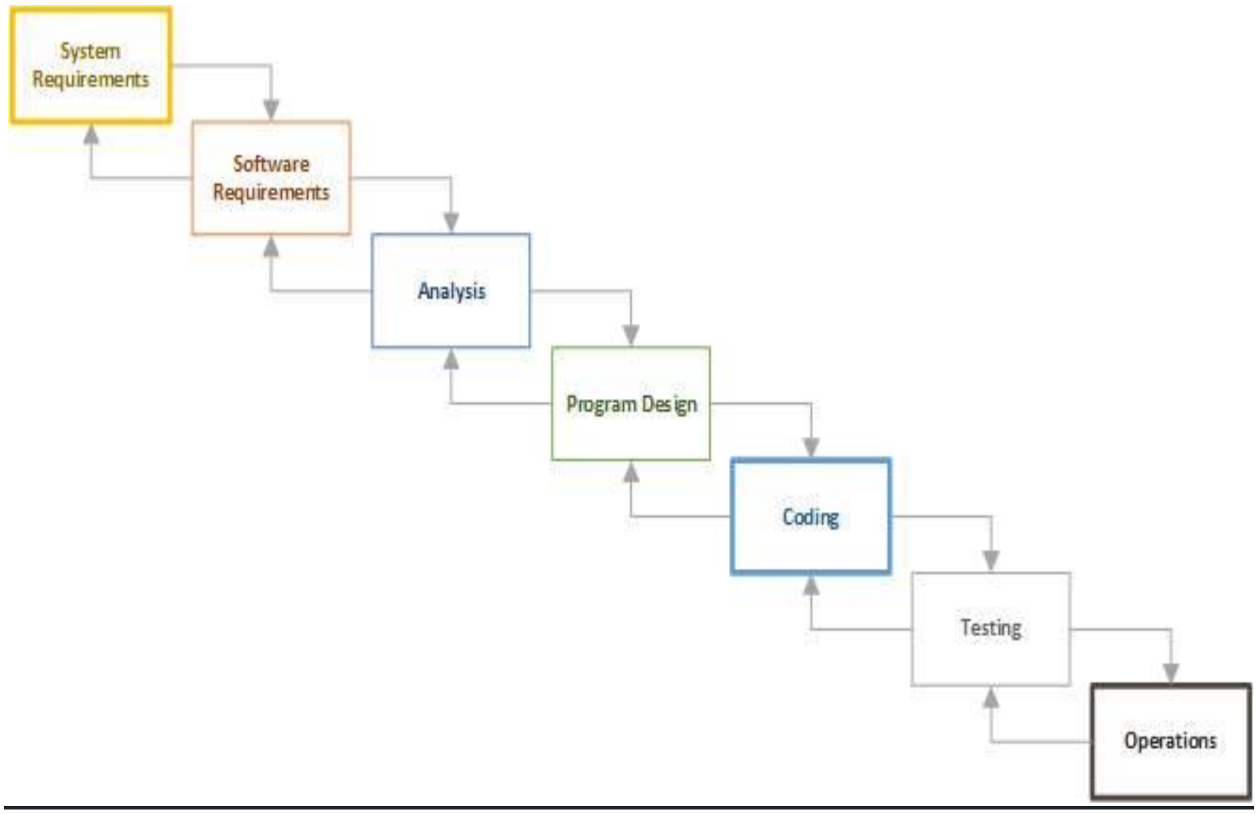


Figure-2:-- Iterative Waterfall Model.

Advantages—

The model has various advantages. It is useful for people in all aspects. Some of the advantages are as follows—

1. The model gives the accurate results and it predicts the class of the disease.
2. It solves the problem of physical meet with the doctors.
3. The patient gets the result by simply sitting in the homes rather than going anywhere.

It predicts almost all the type of diseases ranging from asthma to blood pressure, etc.

CHAPTER-2

Literature Survey / Project Design

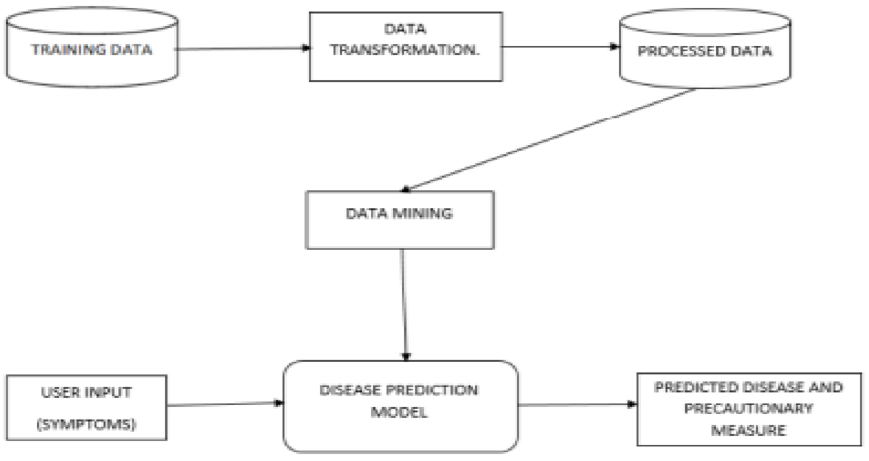
This Model predicts the class of the disease from which a person is suffering. Yet, there are several other models, apps available in the market that perform the same task to some extent. There are also several research papers related to the project in which some of them with its main idea are listed below--

1. A Research Paper proposed Effective Heart Disease Prediction Using Machine Learning Techniques in which strategy that objective is to find the critical includes by applying Machine Learning bringing about improving the exactness in the expectation of cardiovascular malady. The expectation model is created with various blends of highlights and a few known arrangement strategies. The model has a precision level of 88.7% through the prediction model for heart disease with hybrid random forest with a linear model (HRFLM) they likewise educated about Diverse data mining approaches and expectation techniques, Such as, KNN, LR, SVM, NN, and Vote have been fairly famous of late to distinguish and predict heart disease.

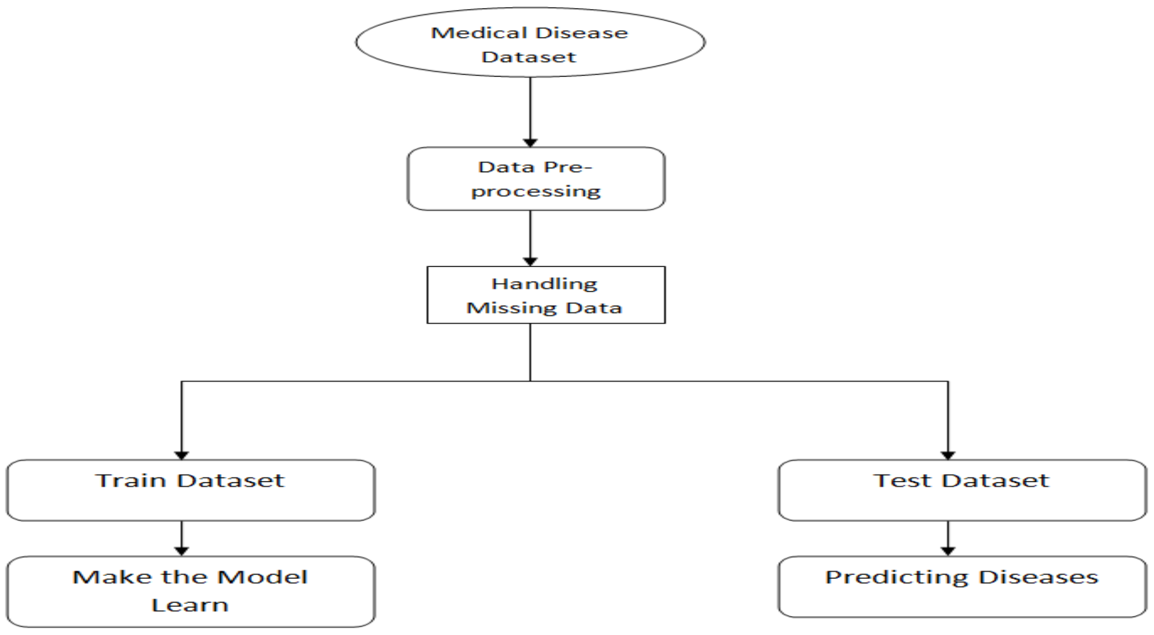
2. Another Research Paper has built up which was titled as Prediction of Disease Using Machine Learning Algorithms. This exploration plans to give a point by point portrayal using the Naive Bayes Algorithm and the decision tree classifier that are applied especially in the prediction of Disease. Some analysis has been led to think about the maintenance and the execution of prescient strategy on the equivalent dataset, and the result uncovers that Decision Tree beats over Bayesian classification system.

Project Related Diagram Models—

The models which easily depict the Disease Prediction System are shown below—



<u>Figure-3.1</u>—Block Diagram of the Disease Prediction System.



<u>Figure-3.2—</u>Framework of the Disease Prediction System.

<u>Feasibility Study—</u>

Feasibility study is disbursed when there's a fancy problem or opportunity. it's considered because the primary investigation which emphasises on "Look before

You Work" approach to any project .A Feasibility study is undertaken to see the likelihood of either improving the present system or developing a very new system.

We are visiting develop the new system which is possible as our application is incredibly user friendly and simple to apply/understand.

Technical Feasibility—

In this kind of study the present technology in employed in a corporation is checked like the prevailing software, hardware, and private staff to work out whether it'll work for the proposed system. The technology that was important in developing a brand new system like development tools, back-end database system were available from within the organization. The proposed system is capable of handling large storage of information. The back-end and front-end technology has greater important for providing an accurate, error-free, frequencies of knowledge to be used.
• My this project is technically feasible in all aspects. This project will provide latest platform like android technology.

Economical Feasibility—

For proving that system developed is economical, the economical feasibility study takes place to test the value of developing a system against the advantages that it provides. If the price are less and benefits are over we will define our system to be economically developed. User saves time in looking for a specific product to be purchased by simply few clicks. The registration process is very much faster than the manual registration. The all data is stored computerized so it saves paper. The record is of freed from human errors as there's less chance of mistakes. The above benefits are in terms of saving time, minimize errors and supply efficiency in work done.
• In terms of economical feasibility our application is incredibly reasonable in cost. So application is economically feasible.

Operational Feasibility—

The operational feasibility is anxious with the operability of the system after it's been installed. That is, some programmer might not like changes in their routine

method of labor or has fear that they'll lose their contemporaries .The following areas will have the operational feasibility within the proposed project.
• The organization has approved this method as their working system.
• The user of the system has accepted the proposed system as their new working system and realized the advantages of it.
• The system will add a correct way after it's been installed and therefore the installation process is straightforward to use.

In Short—

Feasibility Analysis—

This app is very useful for all persons in every aspect.
“Feasibility Study is the evaluation of the proposed system regarding its workability, effectiveness, and its usefulness for any person / organisation.”

(i) Technical Feasibility--

a. This software can be implemented with the existing technical resources.
b. Required Hardware and software tools are easily available and easy to use.
c. Hence this software is Technically Feasible.

(ii) Economic Feasibility –

This proposed system is financially feasible, because of the following reasons—

a. Installation of the system can be made with very low cost.
b. The hardware requirements for this app is minimum and can be easily affordable, making it economically feasible.

(iii) Operational Feasibility—

a. The operational feasibility of this app is user friendly with good GUI (Graphical User Interface).
b. Information updating is done easily by the administrator.
c. As in present day situation technology is known for making changes and robustness in performing tasks.

(iv) **<u>Behavioral Feasibility—</u>**

People often do not change the devices they are using. So, a prior information has to be collected on how strong a reaction the user having regarding this app.

a. User friendly.

b. Knowledge of programming is required.

c. This will save a lot of time and reduce the workload.

Hence my this app is behaviorally feasible in the organization.

(v) **<u>Schedule feasibility—</u>**

The time schedule required for the development of the project is very important since more development time effects machine time, costs and delay time in the development of other systems.

Chapter – 3

Functionality / Working of the Project

This is a Machine Learning Project in which we are making the predictions in order to get the output. The Machine Learning model is about making predictions of the disease that a person is having. The model uses Machine Learning algorithms to predict the disease that a person is having. The process is carried out by training the model and after training making the successful predictions from it.

The first step is to choose the appropriate dataset for the model in order to train it. The dataset must contain all the required attributes like the type of disease, symptoms, value range, etc. As the model is based on supervised machine learning, so we provide a well – labeled data to train the model.

After chosing the dataset, we divide the dataset into three parts namely—

1. Training Dataset
2. Validation Dataset
3. Test Dataset

1. **Training Dataset—**This dataset is used for the training process. This dataset contains all the necessary attributes for the training purpose of the model.
2. **Validation Dataset—**The training dataset is split into two parts namely training and validation sets. The validation dataset is used for the validation process of the model. After successfully training the model, we validate our model using the validation dataset. The ratio of the split is generally, 80:20 ratio. The 80% part contributes to the training dataset and the 20% part contributes to the validation dataset. Sometimes, this ratio can got to the 90:10 ratio.
3. **Test Dataset—**The test dataset is mainly used for the testing purpose. After successful training and validation, we test our model using the test dataset. This dataset is used to check whether our model is making the accurate predictions or not. The ratio of the train, test and the validation set is generally 70:20:10, where, 70% comes from training dataset, 20% comes from the validation dataset and the 10% part comes from the test dataset.

In order to successfully build this model, we use the Machine Learning Algorithms. The main algorithm is the Supervised Machine Learning Algorithm in which we train the model using the well labeled dataset. The dataset includes the values ranging from a starting value to the highest value, the symptoms and the type of the disease. We will divide the dataset into three parts namely, train, test and validation sets. We will train the model using the training dataset, and then validate it using validation dataset, and finally test the model using the test dataset to check whether the model is giving the accurate results or not. This model basically follows the classification approach as it predicts the class of the disease. It tells from which disease, the person is suffering.
The dataset is split into three parts namely—

1. **Training Dataset**—This dataset is responsible for training the model. With this part, we basically train our model for making accurate predictions.

This comprises of about 70% contribution.

2. **Validation Dataset**—This dataset is responsible for validating our model. This dataset makes a contribution of 20%.
3. **Testing Dataset**—This dataset is responsible for testing our model. This dataset makes result whether the model is predicting accurate results or not. This comprises of 10% contribution.

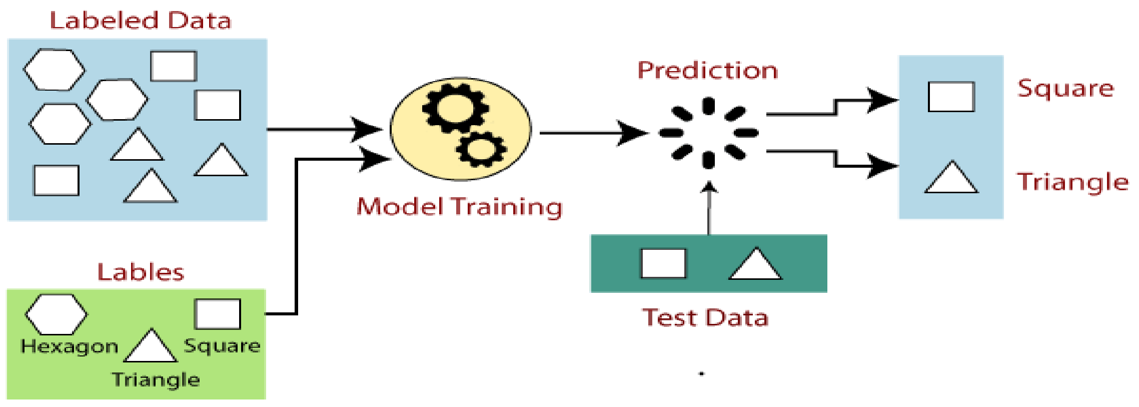


Figure-4—Supervised Machine Learning.

After successfully building the model, we will deploy it and then the user enters his / her details and get the predictions and the accurate results regarding his / her disease.

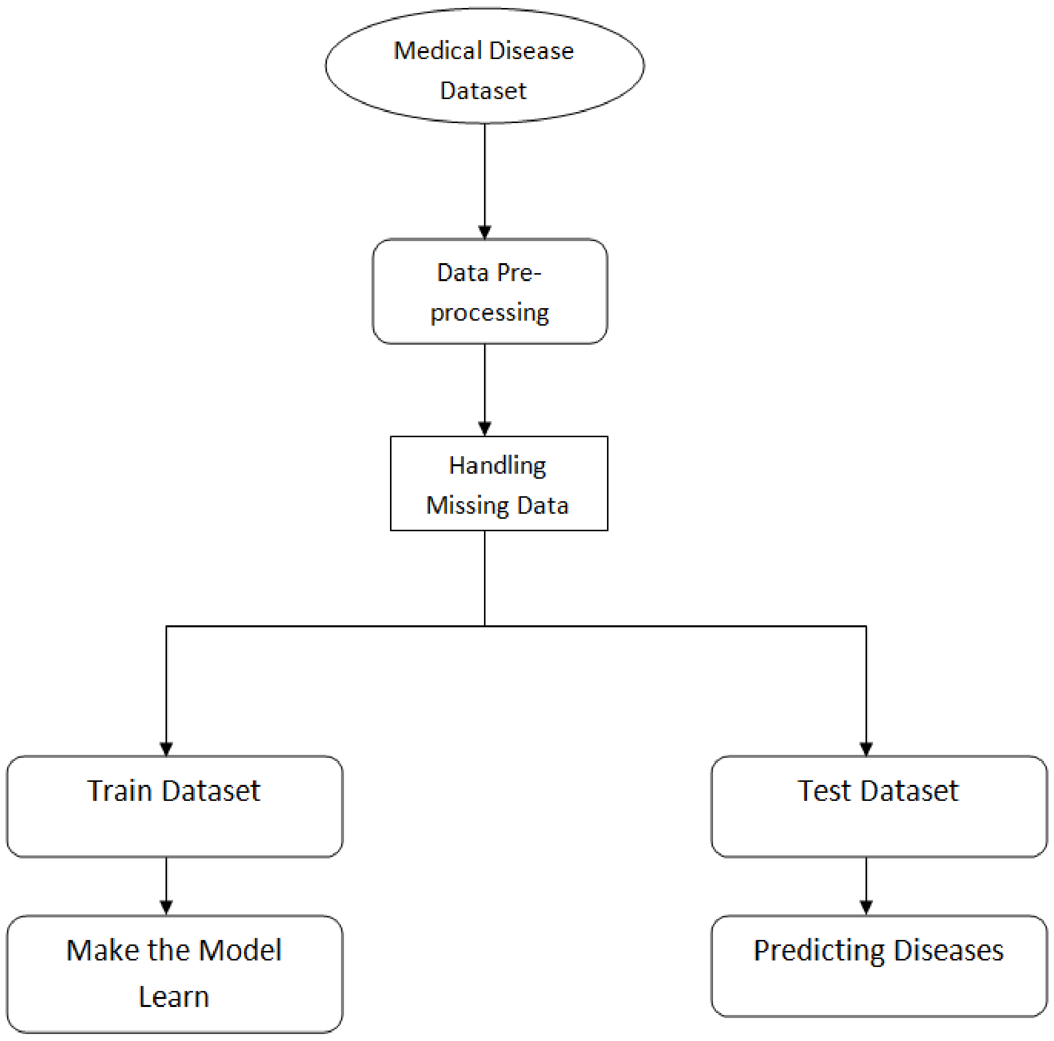


Fig.5- Block Diagram of the Proposed System.

This block diagram shows the actual way followed in this model. As we can see that the first step in building this model is to chose a dataset which is very large and contains all the objects like disease symptoms, disease causes, disease levels, and so on. After chosing the dataset, we pre-process it, like removing the null values, adding some new values, and like many other also. Then, this dataset is divided into training and testing parts using the appropriate functions and of fixed test size.

We train our model using this training dataset. The model learns from this dataset only. As it is supervised machine learning model, we provide a well-labelled data to our model for training purpose. The data is labeled in such a way that the model can easily learn from this dataset only. Finally after training our model, we generate the predictions from our model.

The other diagram is ER-diagram of the proposed system. It shows how different attributes are related to various entities and so on.

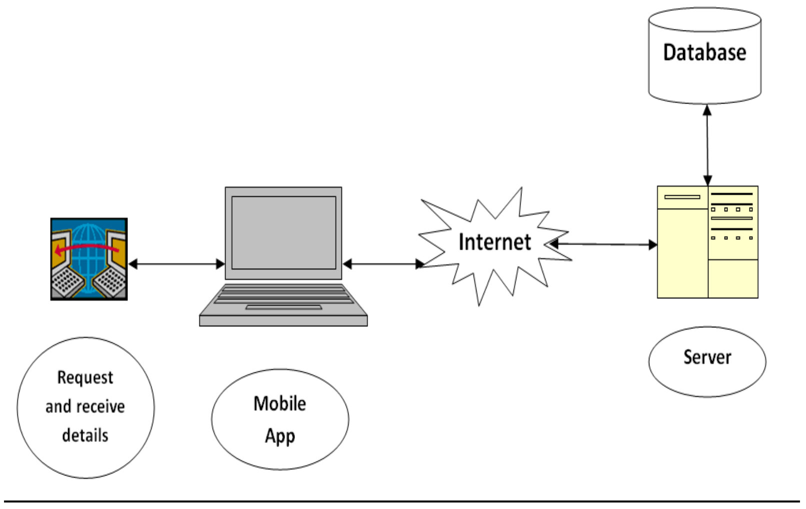


Fig.6- Architectural Diagram of the Proposed System for the App.

This image shows the architectural design of the model app. We can see that model/app is connected to the database(training set) where all the information like type of disease, cause of disease, etc. are stored.

Hardware and Software Requirements--

Hardware:--

- **Processor:--** Intel Pentium IV and above
- **RAM: --** 1GB or more
- Hard disk 250 GB and more

Software:--

- **Operating System:--**
    1. Microsoft Windows XP and above
    2. Smart Phone

1. Marshmallow/ Lolly Pop Version
2. 4 GB RAM or more.

- **Front End:--**
    1. Jupyter Notebook
    2. Kaggle Software
- **Back End:--**
    1. SQL Server
    2. Wamp Version

Others:--

- **Web Browser :--**

  Internet Explorer, Google Chrome, Mozilla Firefox.

Documentation:--

1. Microsoft Word (MS Word) 2010 and above
2. Microsoft Excel (MS Excel) 2010 and above
3. Microsoft Vision (MS) 2010 and above
4. Edraw Max.

Diagrams—

E-R Diagram—

An entity-relationship diagram (ERD) is a graphical representation of an information system that shows the relationship between entities, attributes, people, objects, places, concepts or events within that system. An ERD is basically a data modeling technique that is used as the foundation for a relation database.

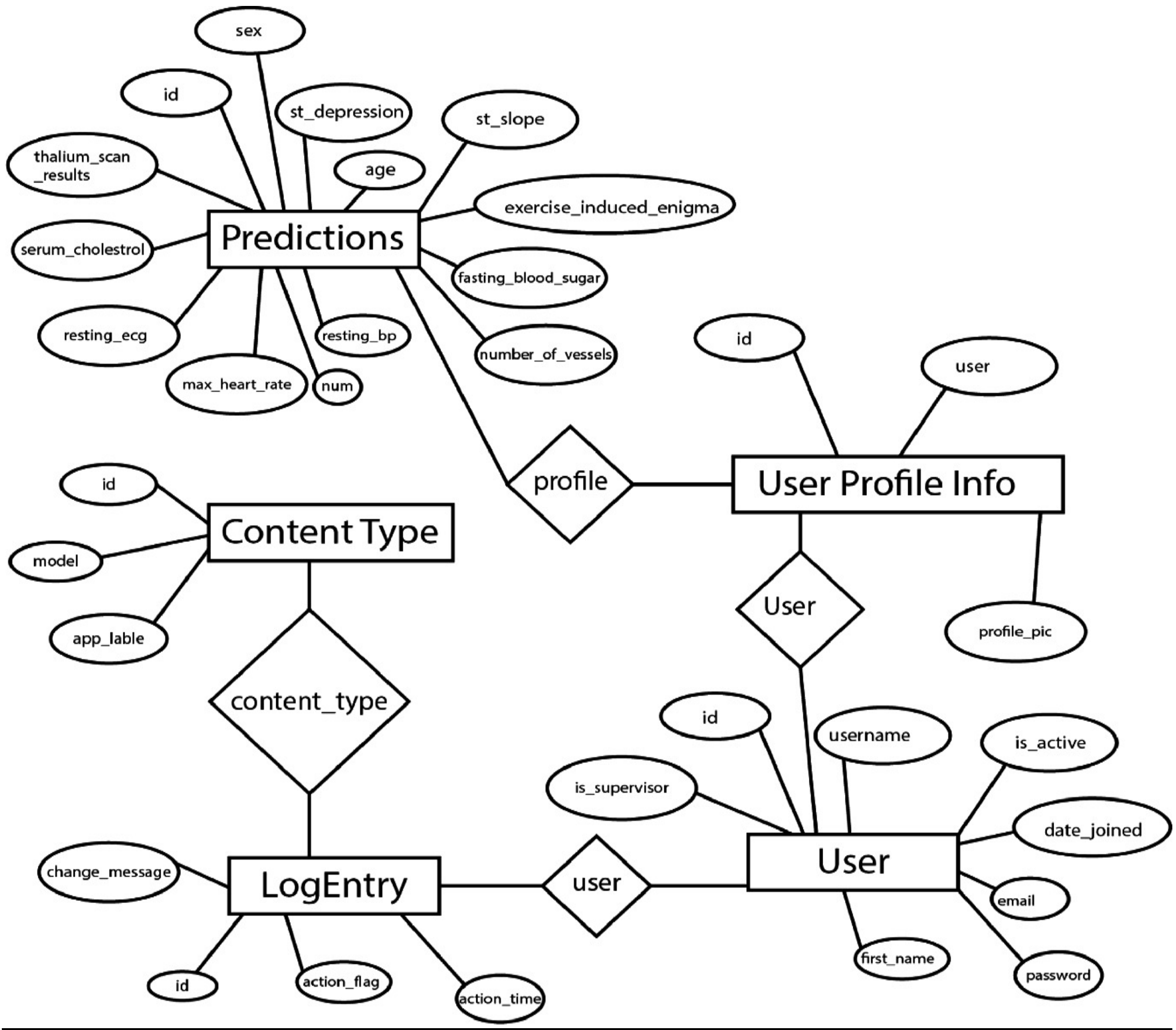


Figure-7—ER - Diagram of the Disease Prediction System.

Activity Diagram—

The Activity diagram tells the flow of control in a system. It consists of the activities and links between them. The type of flow can be sequential, concurrent or branched.

Activities are basically the functions of a system. Number of activity diagrams are prepared to explain the entire flow in a system.

Activity diagrams are used to visualize the flow of controls in a proposed system. It is made to have an idea of how the system works when it is executed.

(i). Activity Diagram—

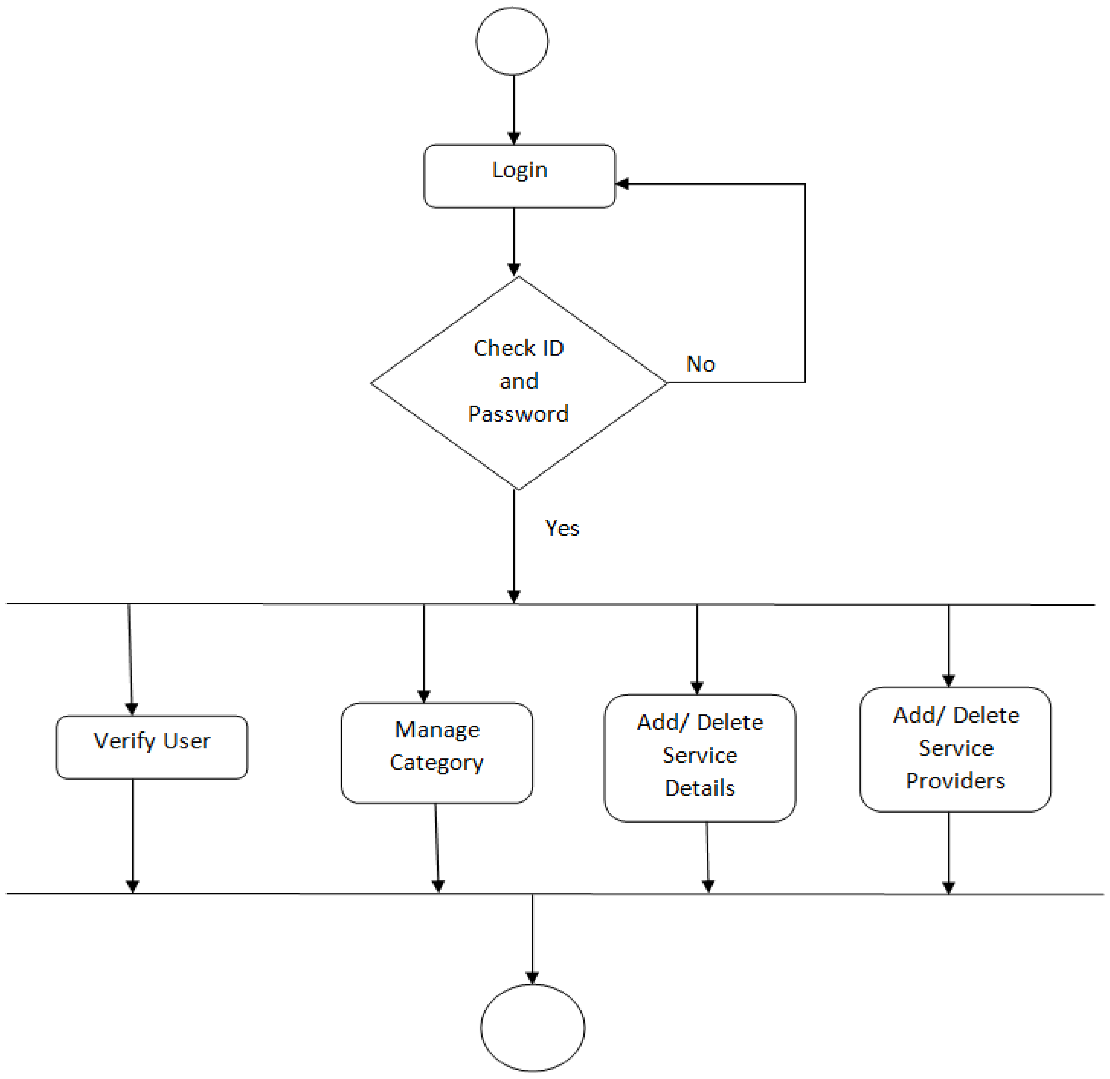


Figure-8—Activity Diagram.

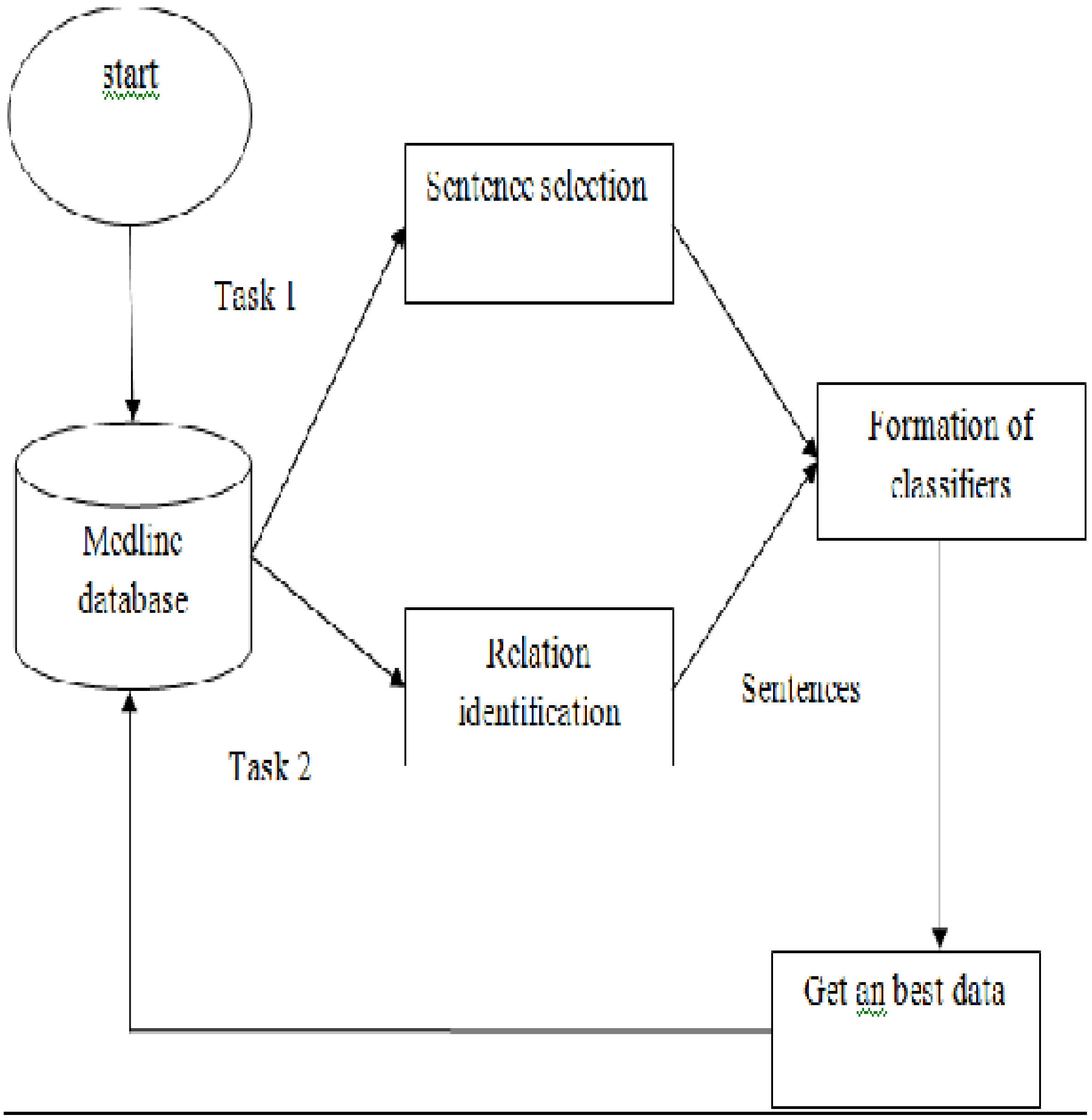


Fig.-9—Model showing relationship between various tasks.

This shows how one task is related to the other tasks. How we can use our database to get the details and relate it with the other factors.

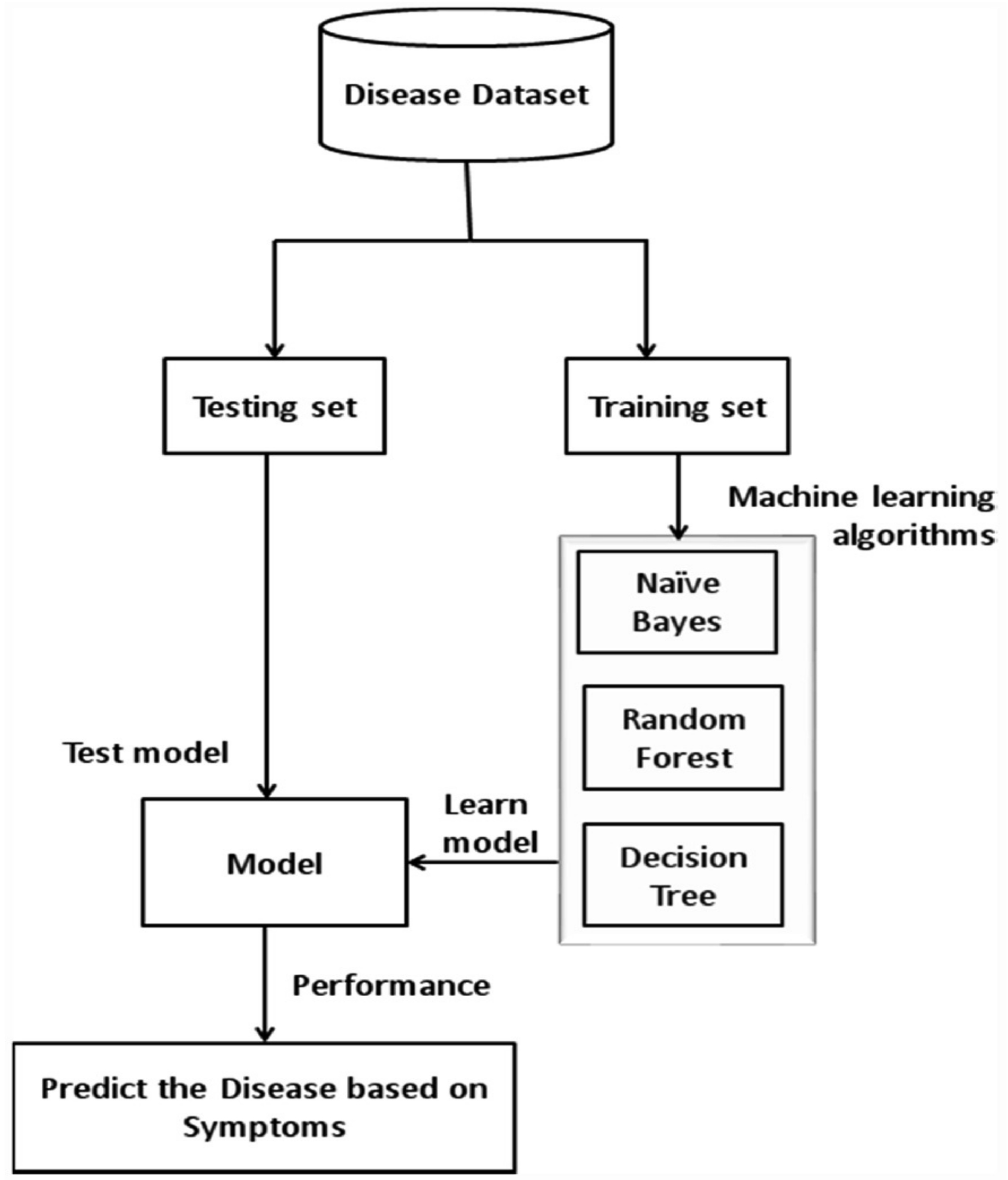


Fig.-10—Diagram depicting various algorithms used on the Dataset.

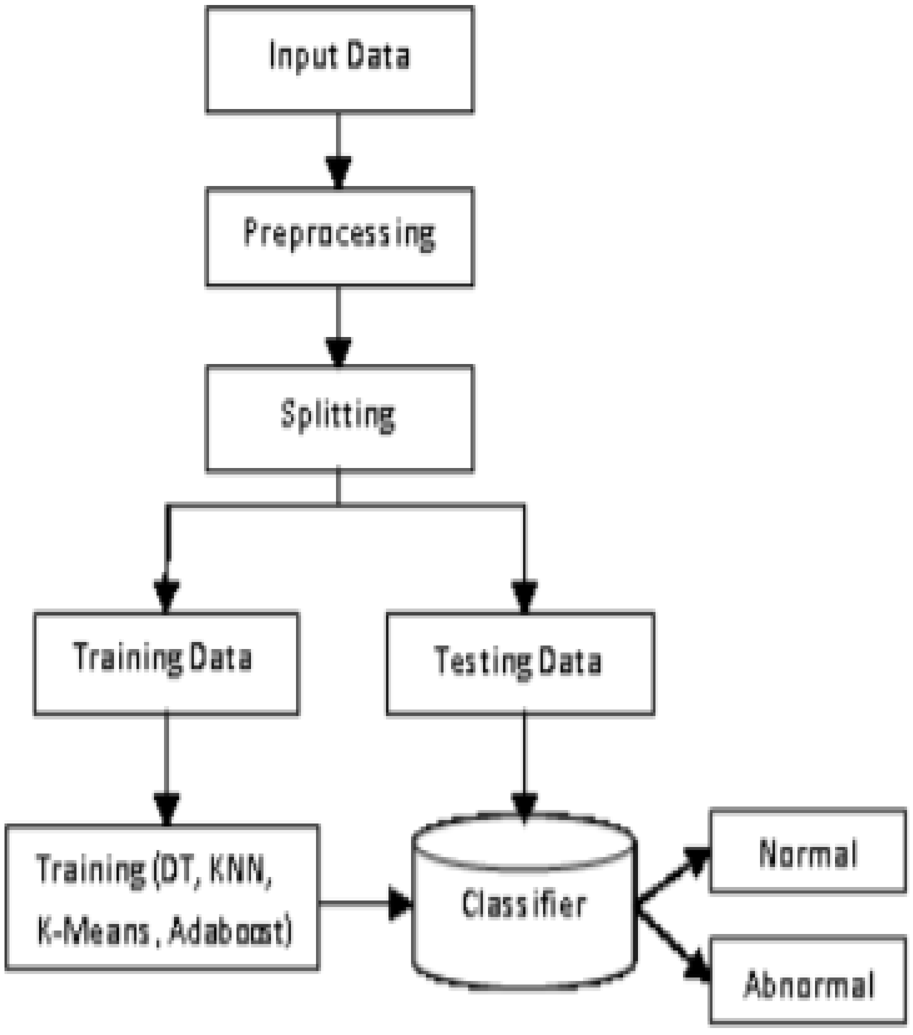
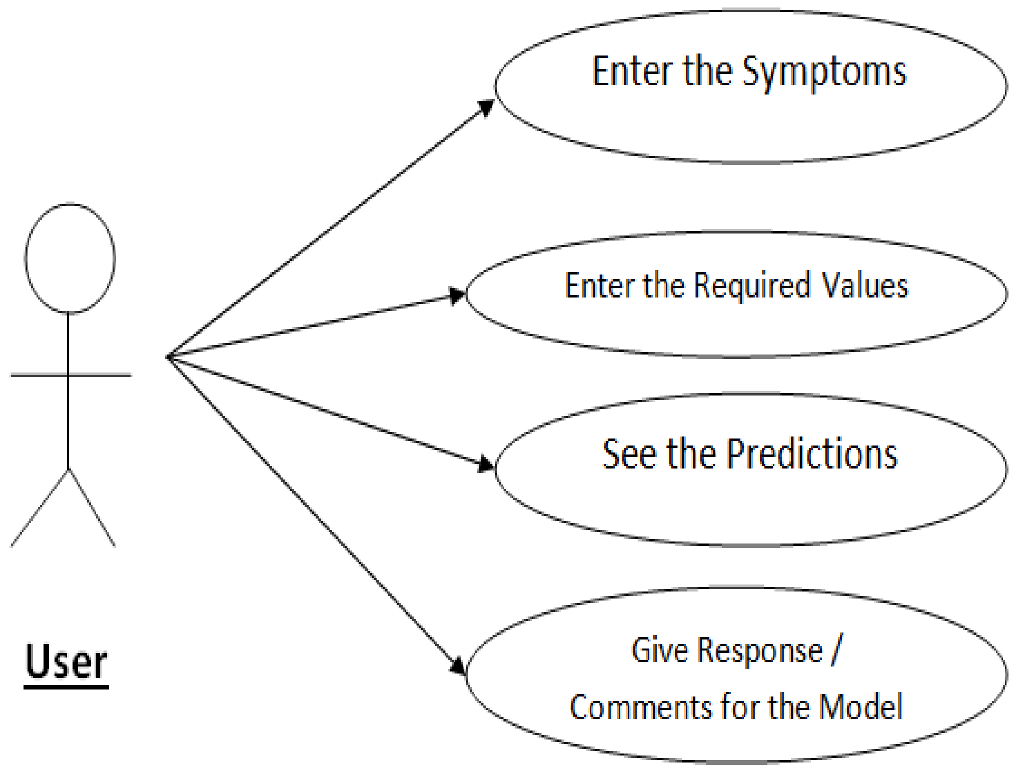


Fig.-11—Proposed Data Classification System.

Use Case Diagram—

Use case diagrams are a set of use cases, and their relationships. They represent the use case view of the proposed system. A use case represents a particular functionality of a the proposed app system. So use case diagrams are used to describe the relationships between the functionalities and their controllers. These controllers are also known as actors.


Enter the Symptoms
Enter the Required Values
See the Predictions
Give Response / Comments for the Model
User

<u>Figure-12</u>—Use-Case Diagram of the Proposed System.

This diagram depicts the various options that are available to the user. As we can see, the user can enter his details and symptoms and based on this given input value the output is generated which is very accurate and correct.
As we can see, the user can—

1. Enter the symptoms (like cough, cold, fever, etc.).
2. Enter all the required values that are necessary for making the predictions.
3. See the predictions generated by the model.
4. Give response or comments for the model seeing the accuracy of it.

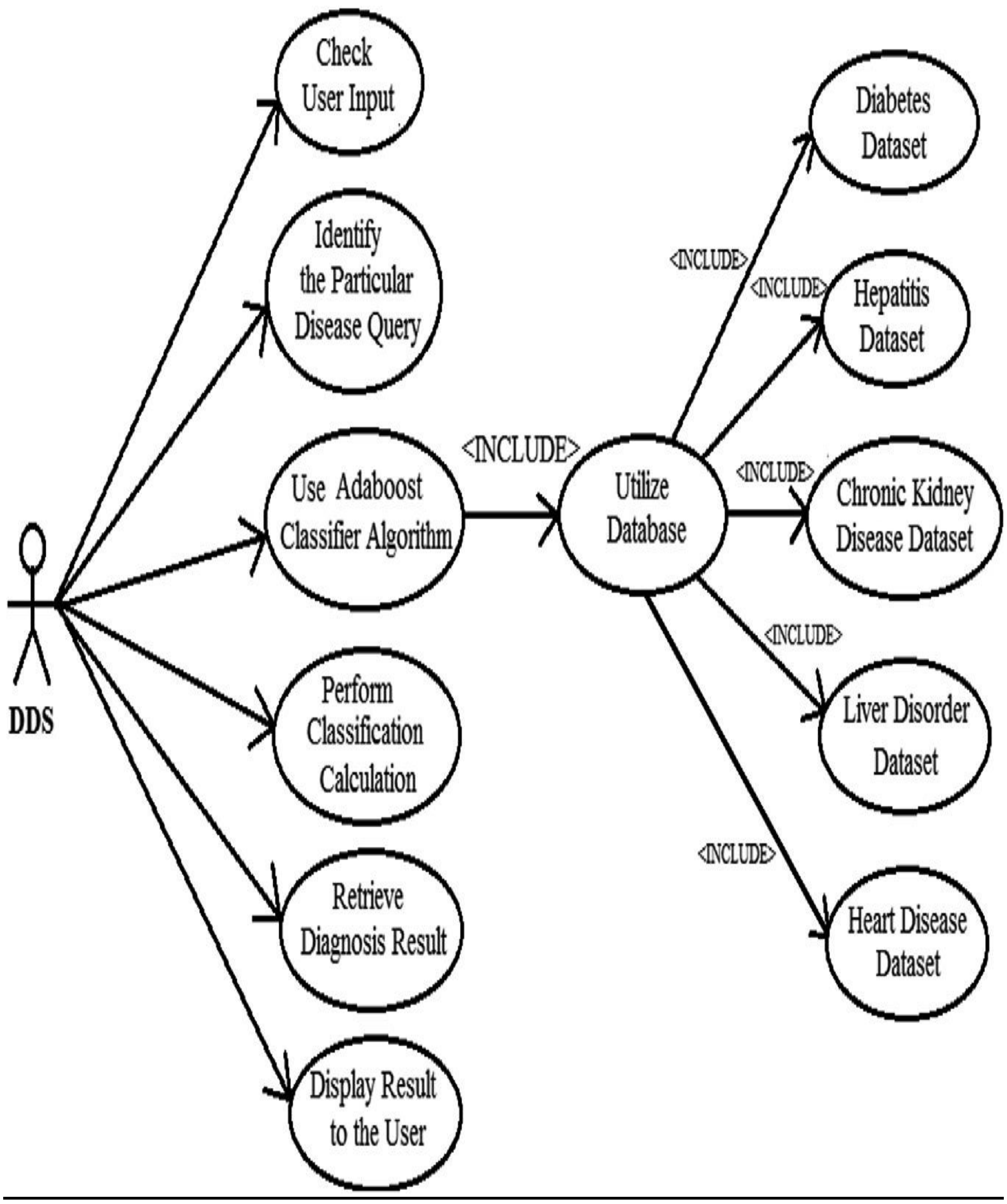
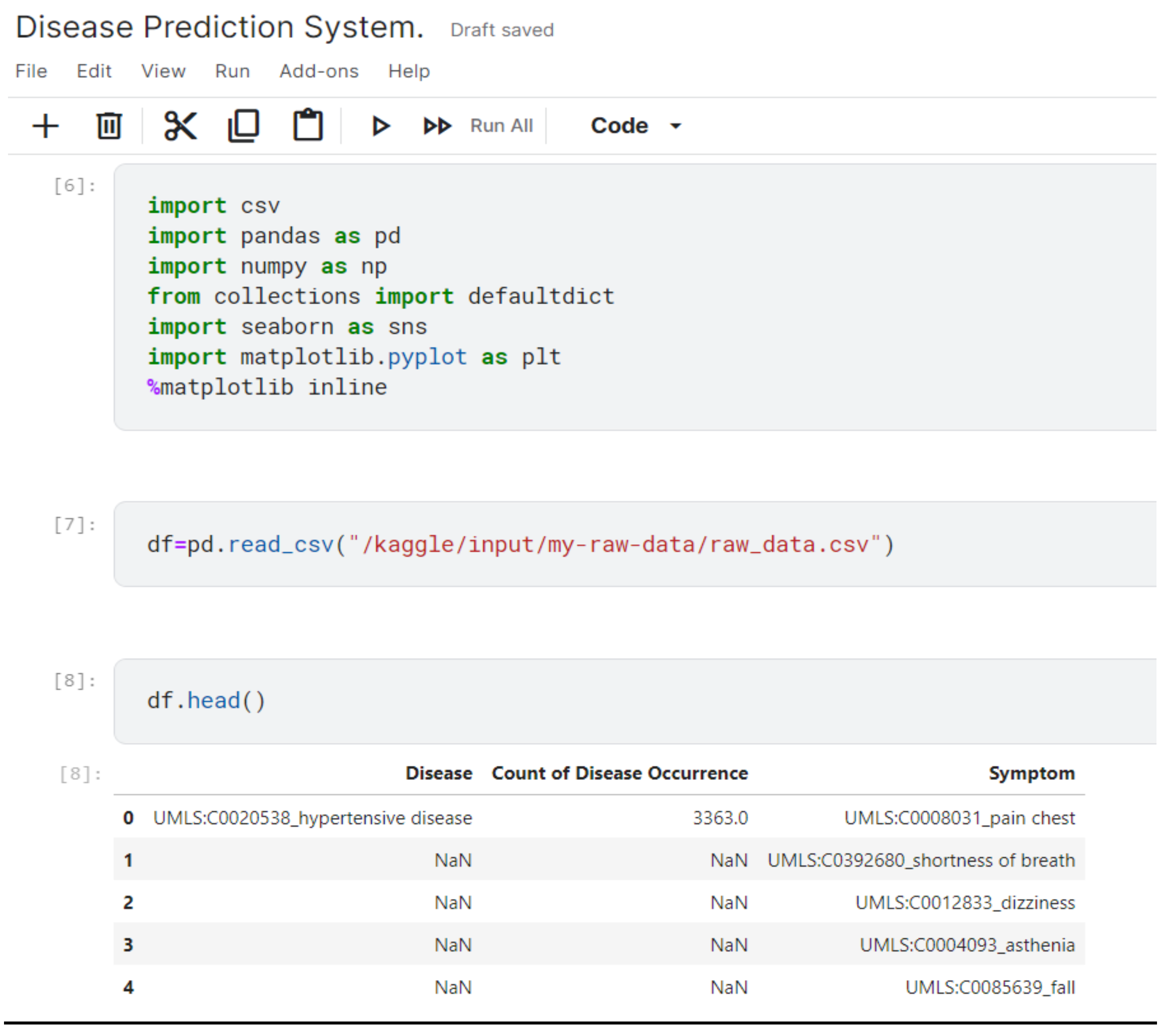

Check User Input
Identify the Particular Disease Query
Use Adaboost Classifier Algorithm
Perform Classification Calculation
Retrieve Diagnosis Result
Display Result to the User
DDS
<INCLUDE>
Utilize Database
<INCLUDE>
<INCLUDE>
<INCLUDE>
<INCLUDE>
<INCLUDE>
Diabetes Dataset
Hepatitis Dataset
Chronic Kidney Disease Dataset
Liver Disorder Dataset
Heart Disease Dataset

Fig.-13—Disease Detection Technique using the DDS System.

Chapter—4

Results and Discussions


Disease Prediction System. Draft saved
File Edit View Run Add-ons Help
Run All Code

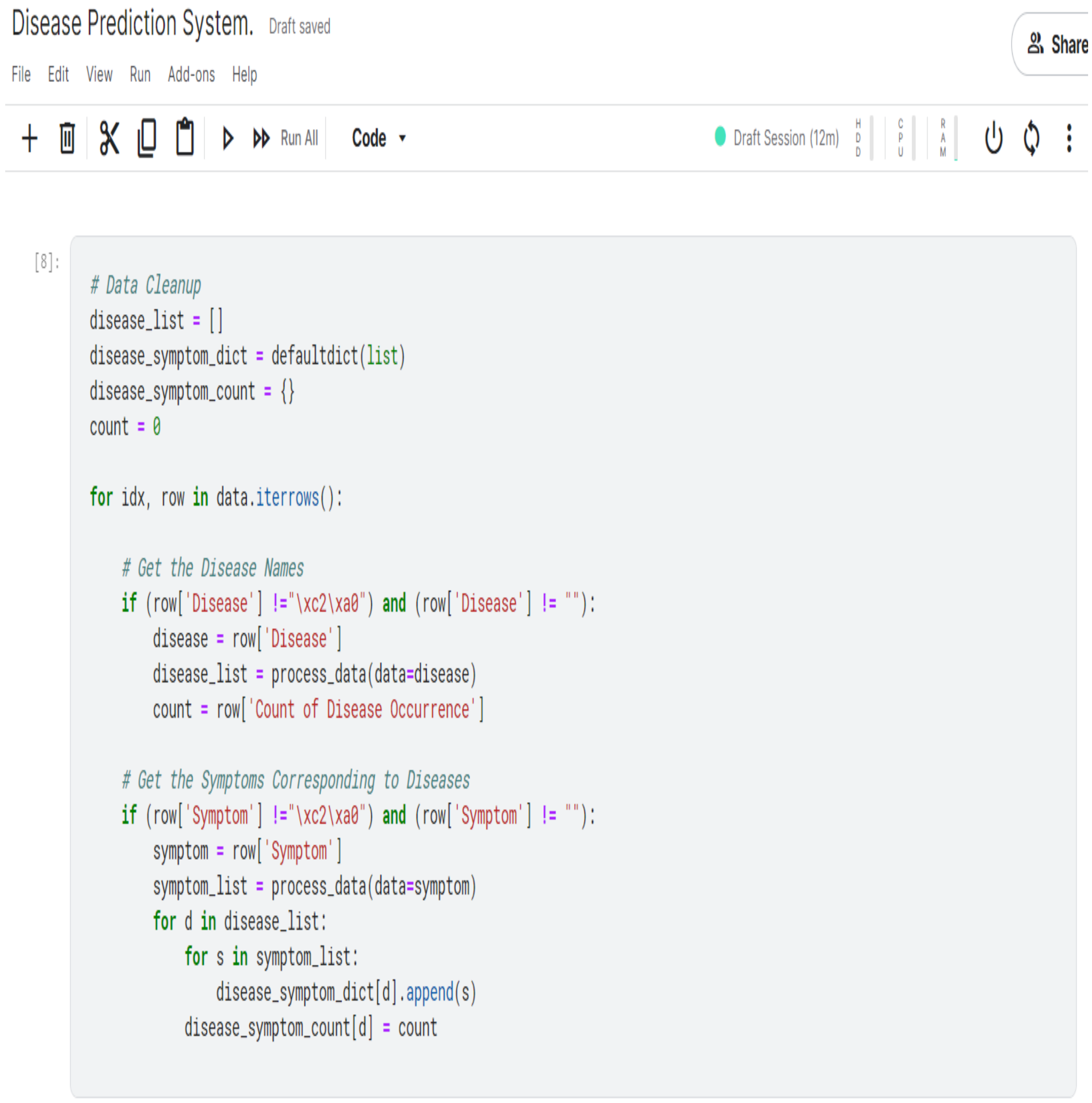
```
[6]:  import csv
      import pandas as pd
      import numpy as np
      from collections import defaultdict
      import seaborn as sns
      import matplotlib.pyplot as plt
      %matplotlib inline

[7]:  df=pd.read_csv("/kaggle/input/my-raw-data/raw_data.csv")

[8]:  df.head()

[8]:
                            Disease  Count of Disease Occurrence                          Symptom
0  UMLS:C0020538_hypertensive disease                       3363.0          UMLS:C0008031_pain chest
1                                NaN                          NaN  UMLS:C0392680_shortness of breath
2                                NaN                          NaN           UMLS:C0012833_dizziness
3                                NaN                          NaN            UMLS:C0004093_asthenia
4                                NaN                          NaN                UMLS:C0085639_fall
```

Figure-14- Reading the Dataset.

The screenshot shows the reading of the dataset by the model. We see that the dataset has been successfully uploaded to the model and the model will make predictions based on the training on this dataset only.

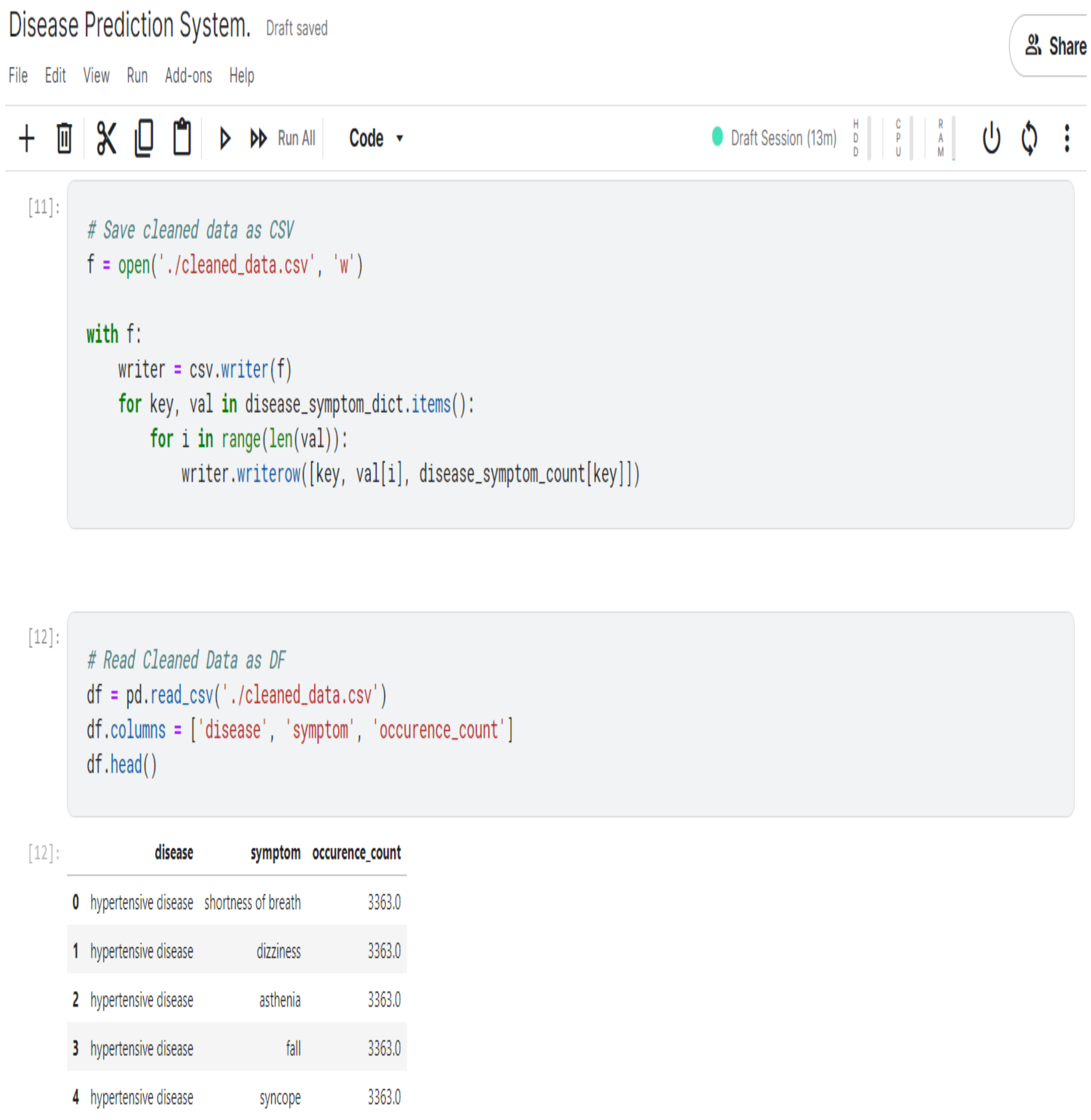
The dataset is also printed using the print() command below.

Firstly, the csv file of the dataset is uploaded using the read.csv() command and then all the operations are performed using this dataset only.


Disease Prediction System. Draft saved
File Edit View Run Add-ons Help
Run All
Code
Draft Session (12m)
HDD CPU RAM
Share
[8]:

```python
# Data Cleanup
disease_list = []
disease_symptom_dict = defaultdict(list)
disease_symptom_count = {}
count = 0

for idx, row in data.iterrows():

    # Get the Disease Names
    if (row['Disease'] !="\xc2\xa0") and (row['Disease'] != ""):
        disease = row['Disease']
        disease_list = process_data(data=disease)
        count = row['Count of Disease Occurrence']

    # Get the Symptoms Corresponding to Diseases
    if (row['Symptom'] !="\xc2\xa0") and (row['Symptom'] != ""):
        symptom = row['Symptom']
        symptom_list = process_data(data=symptom)
        for d in disease_list:
            for s in symptom_list:
                disease_symptom_dict[d].append(s)
            disease_symptom_count[d] = count
```

This shows the data pre-processing step. As we do some modifications in the dataset as removing null values, adding some new features, i.e., cleaning-up the data also. So, we perform all the required operations on the dataset for training our model.

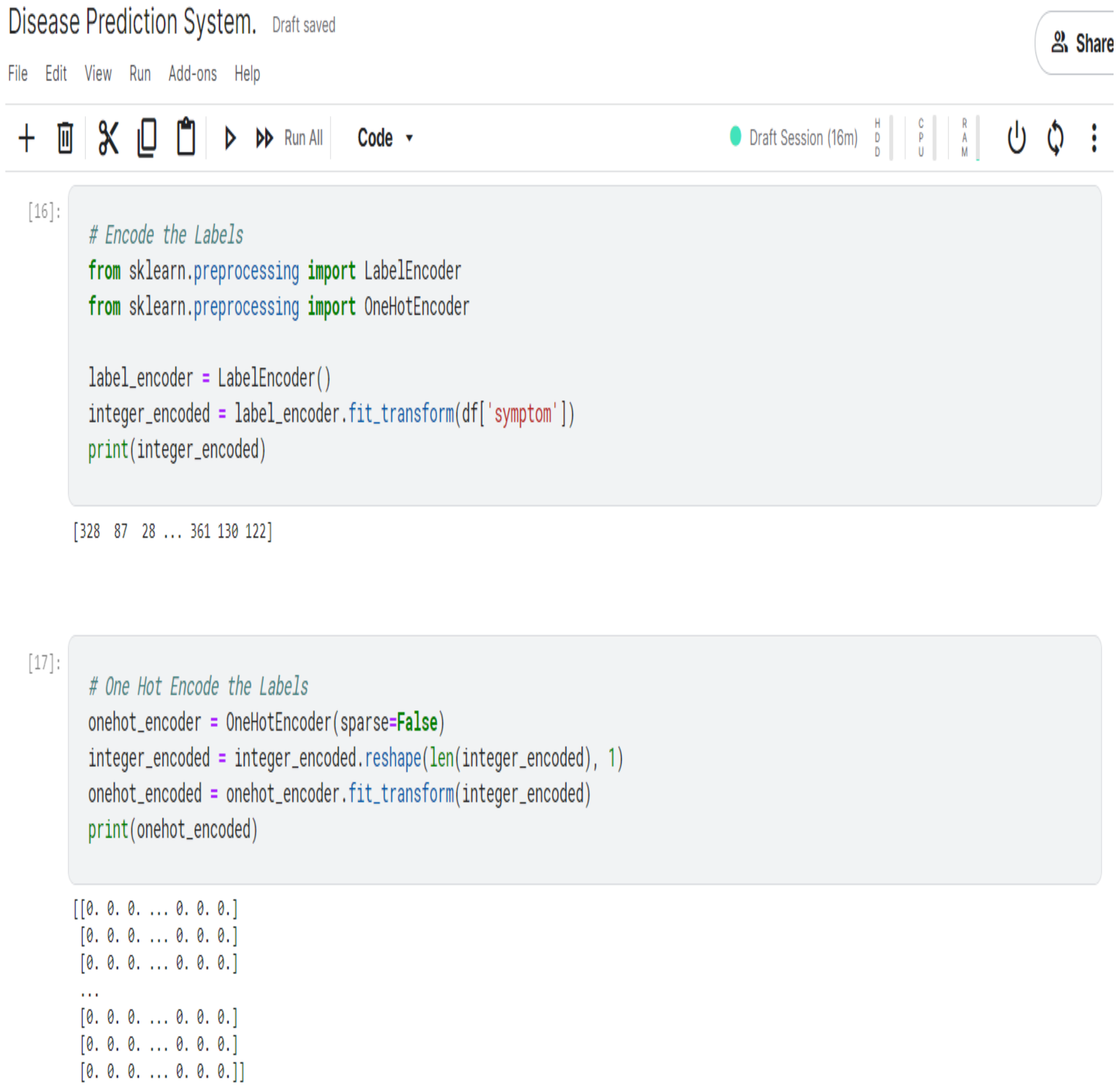

Disease Prediction System. Draft saved
Share
File Edit View Run Add-ons Help
Run All
Code
Draft Session (13m)
HDD
CPU
RAM

```
[11]:
# Save cleaned data as CSV
f = open('./cleaned_data.csv', 'w')

with f:
    writer = csv.writer(f)
    for key, val in disease_symptom_dict.items():
        for i in range(len(val)):
            writer.writerow([key, val[i], disease_symptom_count[key]])
```

```
[12]:
# Read Cleaned Data as DF
df = pd.read_csv('./cleaned_data.csv')
df.columns = ['disease', 'symptom', 'occurence_count']
df.head()
```

```
[12]:
              disease            symptom  occurence_count
0  hypertensive disease  shortness of breath           3363.0
1  hypertensive disease            dizziness           3363.0
2  hypertensive disease             asthenia           3363.0
3  hypertensive disease                 fall           3363.0
4  hypertensive disease              syncope           3363.0
```

After pre-processing the data, we save our data in a local csv file as “cleaned_data.csv”. After that we perform all the operations by using this file only.


Disease Prediction System. Draft saved
File Edit View Run Add-ons Help
Run All
Code
Draft Session (16m)
Share

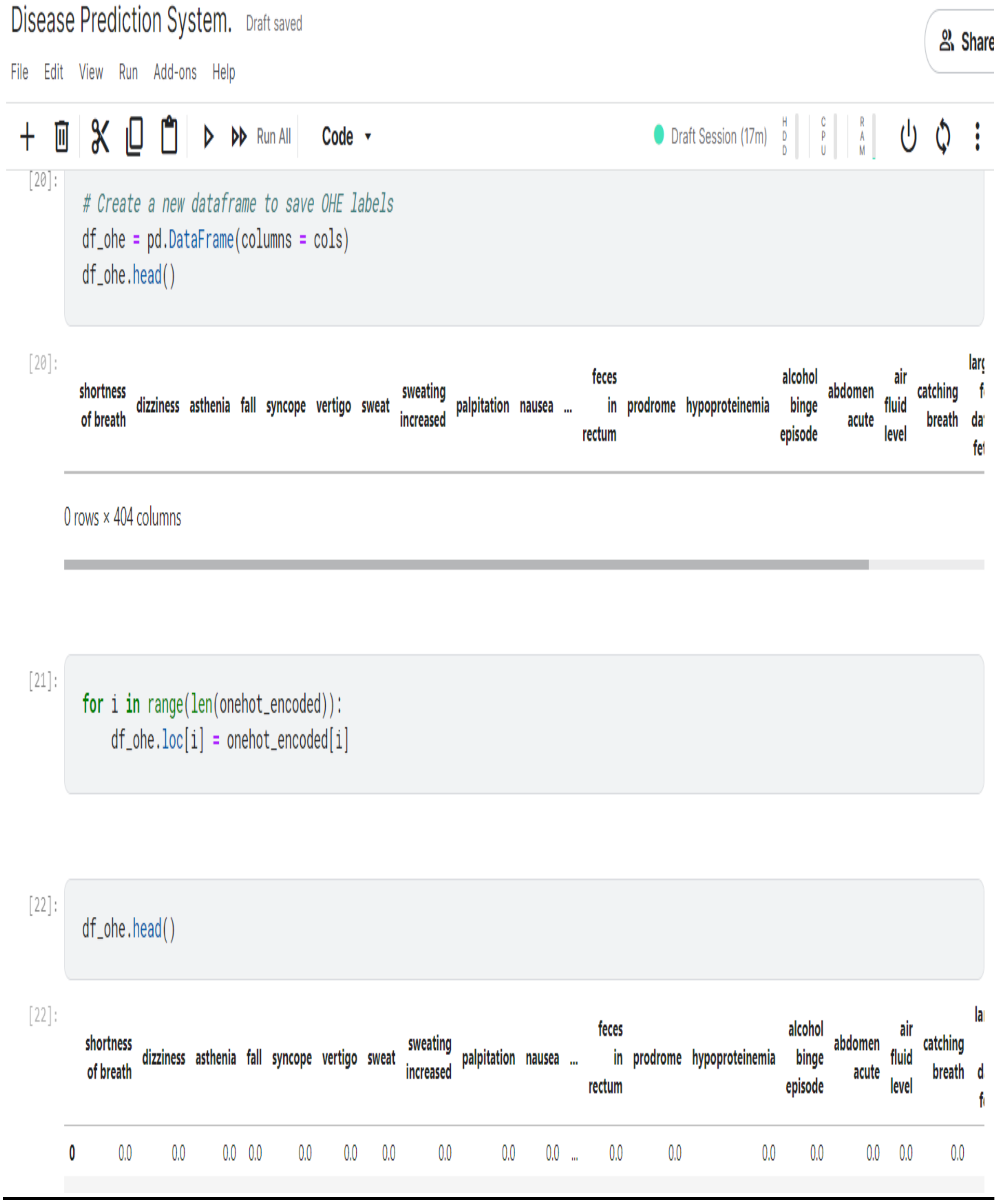
```
[16]:
# Encode the Labels
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(df['symptom'])
print(integer_encoded)
```

```
[328  87  28 ... 361 130 122]
```

```
[17]:
# One Hot Encode the Labels
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
print(onehot_encoded)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

This shows the encoding of the data. The data is encoded using certain keywords and after that its encoded form is also printed.


Disease Prediction System. Draft saved
Share
File Edit View Run Add-ons Help
Run All
Code
Draft Session (17m)
HDD
CPU
RAM

```
[20]:
# Create a new dataframe to save OHE labels
df_ohe = pd.DataFrame(columns = cols)
df_ohe.head()
```

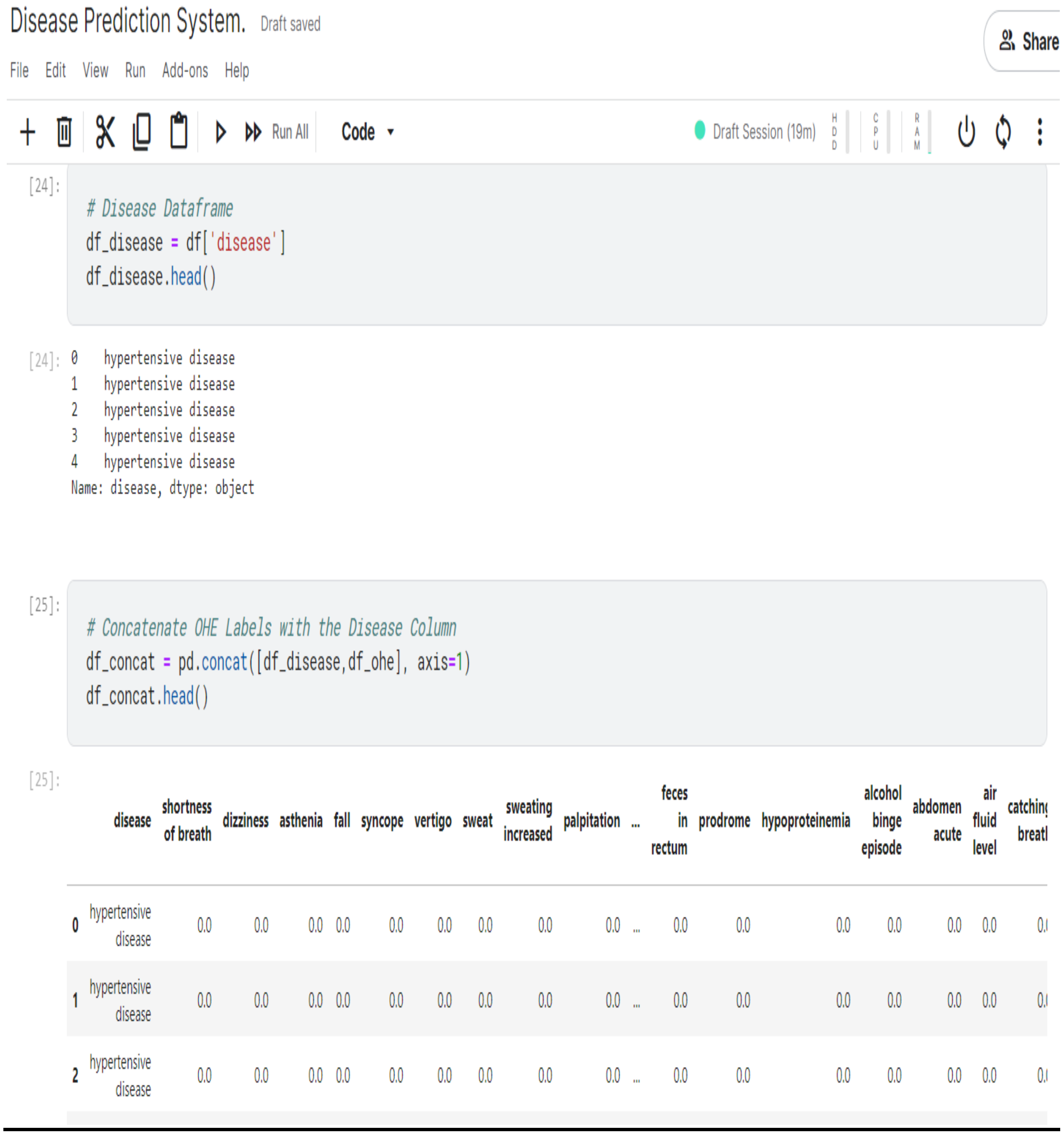
```
[20]:
   shortness  dizziness asthenia fall syncope vertigo sweat sweating  palpitation nausea ... feces    prodrome hypoproteinemia alcohol abdomen air   catching larg
   of breath                                                 increased                        in                             binge   acute   fluid breath   f
                                                                                              rectum                         episode         level          da
                                                                                                                                                             fe

0 rows × 404 columns
```

```
[21]:
for i in range(len(onehot_encoded)):
    df_ohe.loc[i] = onehot_encoded[i]
```

```
[22]:
df_ohe.head()
```

```
[22]:
   shortness  dizziness asthenia fall syncope vertigo sweat sweating  palpitation nausea ... feces    prodrome hypoproteinemia alcohol abdomen air   catching la
   of breath                                                 increased                        in                             binge   acute   fluid breath   d
                                                                                              rectum                         episode         level          f
0        0.0        0.0      0.0  0.0     0.0     0.0   0.0       0.0          0.0    0.0 ...    0.0       0.0             0.0     0.0     0.0   0.0      0.0
```

After all these steps, we create a new dataframe for our model which initially has no rows. After modifications, we add up all the data in this new dataframe only.


Disease Prediction System. Draft saved
Share
File Edit View Run Add-ons Help
Run All
Code
Draft Session (19m)
HDD
CPU
RAM

[24]:
```
# Disease Dataframe
df_disease = df['disease']
df_disease.head()
```

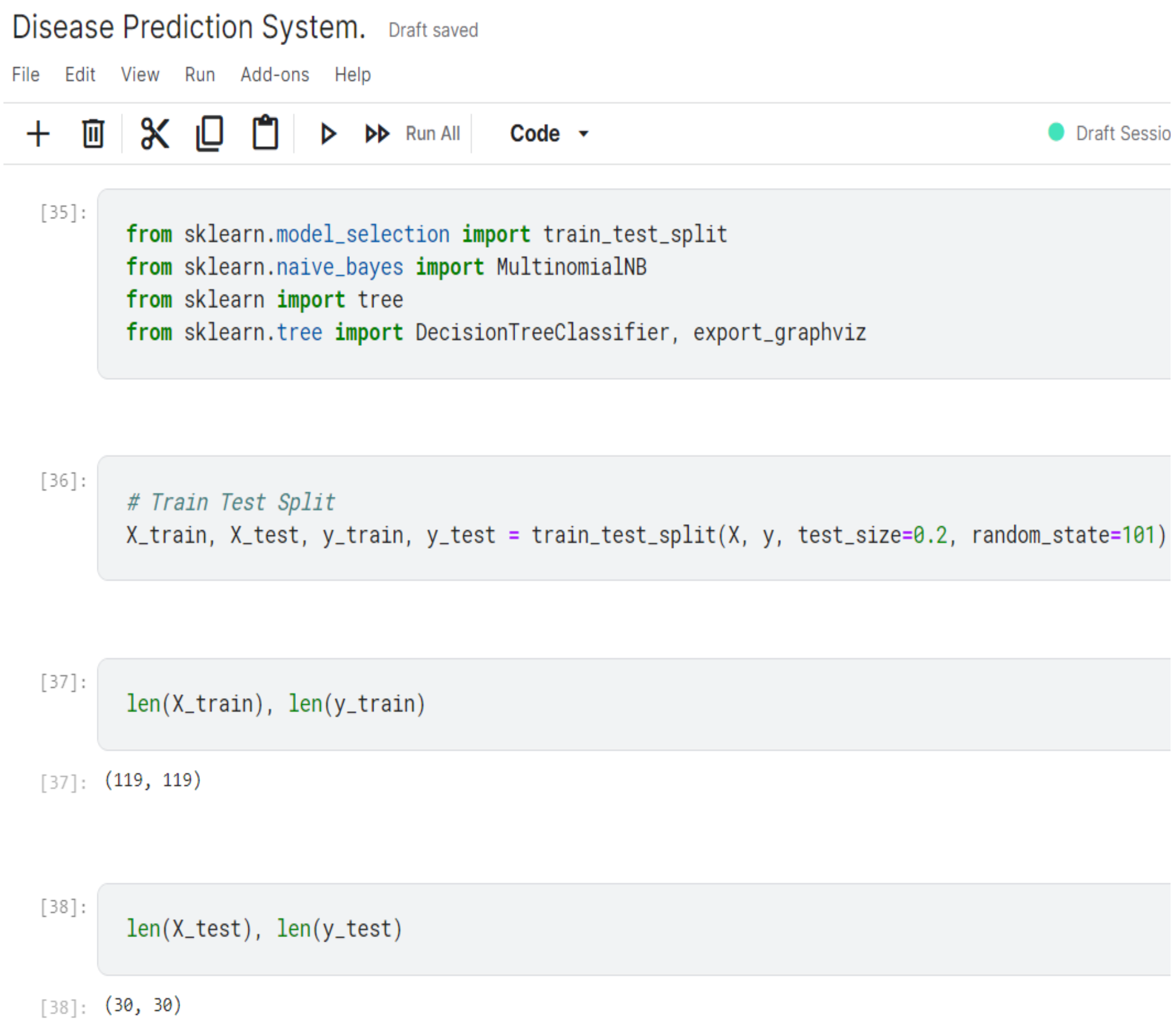
[24]:
```
0    hypertensive disease
1    hypertensive disease
2    hypertensive disease
3    hypertensive disease
4    hypertensive disease
Name: disease, dtype: object
```

[25]:
```
# Concatenate OHE Labels with the Disease Column
df_concat = pd.concat([df_disease,df_ohe], axis=1)
df_concat.head()
```

[25]:

	disease	shortness of breath	dizziness	asthenia	fall	syncope	vertigo	sweat	sweating increased	palpitation	...	feces in rectum	prodrome	hypoproteinemia	alcohol binge episode	abdomen acute	air fluid level	catching breat
0	hypertensive disease	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.
1	hypertensive disease	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.
2	hypertensive disease	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.

We print the dataframe of disease which contains the names of all the types of disease and our model will make predictions from this dataframe only.


Disease Prediction System. Draft saved
File Edit View Run Add-ons Help
Run All Code
Draft Sessio

```
[35]:
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, export_graphviz
```

```
[36]:
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
```

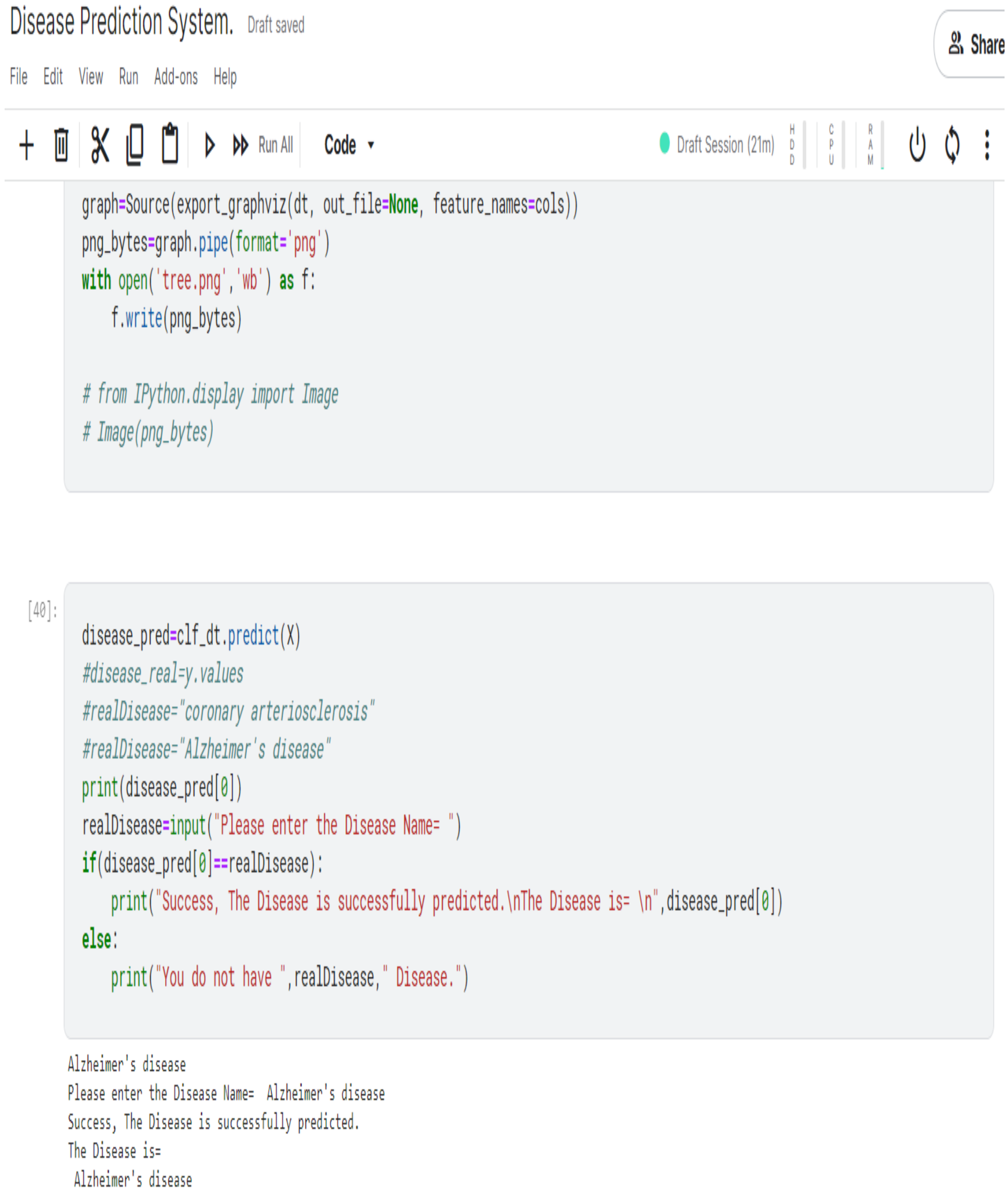
```
[37]:
len(X_train), len(y_train)
[37]: (119, 119)
```

```
[38]:
len(X_test), len(y_test)
[38]: (30, 30)
```

<u>Figure-15- Splitting the Dataset.</u>

This screenshot shows the splitting of the dataset into training and testing parts. We have divided the dataset into training and testing parts using the train_test_split() command with the test size of 0.2.

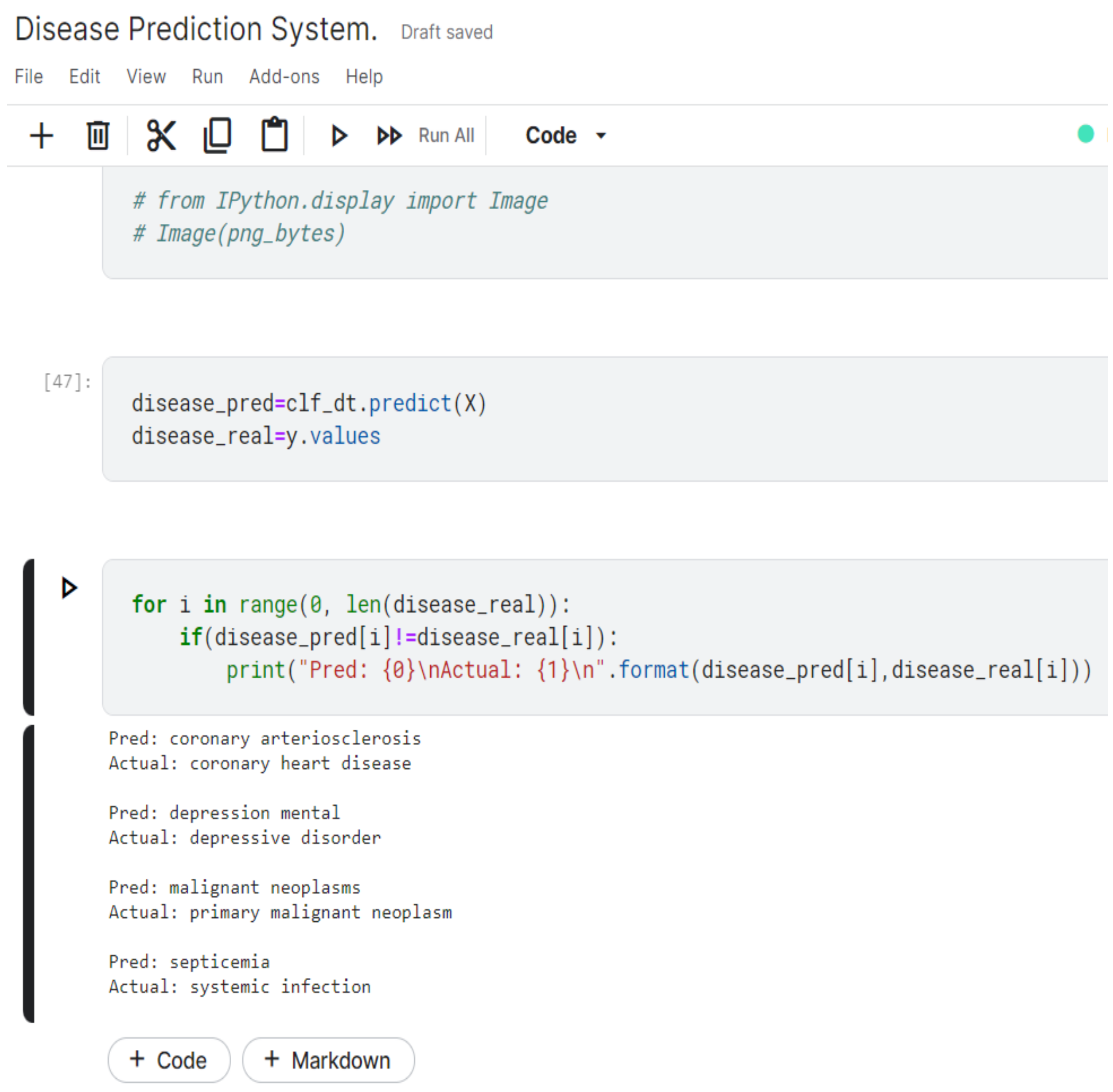
The length of the training datasets (X_train and Y-train) and the testing datasets (X_test and Y_test) are also printed in the below cells.


Disease Prediction System. Draft saved
File Edit View Run Add-ons Help
Share
Run All
Code
Draft Session (21m)
HDD
CPU
RAM
[40]:

```
graph=Source(export_graphviz(dt, out_file=None, feature_names=cols))
png_bytes=graph.pipe(format='png')
with open('tree.png','wb') as f:
    f.write(png_bytes)

# from IPython.display import Image
# Image(png_bytes)
```

```
disease_pred=clf_dt.predict(X)
#disease_real=y.values
#realDisease="coronary arteriosclerosis"
#realDisease="Alzheimer's disease"
print(disease_pred[0])
realDisease=input("Please enter the Disease Name= ")
if(disease_pred[0]==realDisease):
    print("Success, The Disease is successfully predicted.\nThe Disease is= \n",disease_pred[0])
else:
    print("You do not have ",realDisease," Disease.")
```

```
Alzheimer's disease
Please enter the Disease Name=  Alzheimer's disease
Success, The Disease is successfully predicted.
The Disease is=
 Alzheimer's disease
```


Disease Prediction System. Draft saved
File Edit View Run Add-ons Help
Run All Code
+ Code + Markdown

```
# from IPython.display import Image
# Image(png_bytes)
```

```
[47]:
disease_pred=clf_dt.predict(X)
disease_real=y.values
```

```
for i in range(0, len(disease_real)):
    if(disease_pred[i]!=disease_real[i]):
        print("Pred: {0}\nActual: {1}\n".format(disease_pred[i],disease_real[i]))
```

```
Pred: coronary arteriosclerosis
Actual: coronary heart disease

Pred: depression mental
Actual: depressive disorder

Pred: malignant neoplasms
Actual: primary malignant neoplasm

Pred: septicemia
Actual: systemic infection
```

<u>Figure-16- Predicting the Result from the Model.</u>

This image shows the predictions generated from the model. As we trained our model using a large dataset, and performed testing also. So, we are predicting the type of disease, cause of the disease, etc. from the model.

<u>Chapter-- 5</u>

<u>Implementation and Testing</u>

<u>Testing:--</u>

Software Testing is the most important part for the development of any type of software. It involves checking the system, whether the software is responding well to the requests or not. It checks whether the outputs for the corresponding inputs are well or not. It also tells whether the system is responding well to the requests or not.

In my this model. Testing is done by checking the Login credentials whether it is responding well or not. Testing is also done by checking the user details and predictions generated, whether the model is able to make successful predictions or not. Other than that there are also various types of testings done to check the functionalities of the model.

The model is tested for validation, its implementation and its navigation.

1. <u>Validation Testing:--</u>

The user must login to the system with his/her unique login name and password and must enter all the mandatory fields. If the same does not happens, then a warning message is displayed.

The user accesses his / her account with the credentials. If the credentials are correct, then, the login is successful else login is unsuccessful.

We check this in the Validation Testing.

2. <u>Functional Testing:--</u>

The entire system is divided into sub - modules. Here, in this testing, addition or updation of the user data / information in the database is performed.

In this type of testing, we try to add something new data / information or we try to update the prior information. If all the things goes well, then this testing gives positive result else not.

3. **Navigational Testing:--**

The system is tested so that all the pages of the model are properly accessible with their respective links and the data.

We check whether all the navigations in the model are proper or not, i.e., whether we are successfully navigating to other screens of the model well or not.

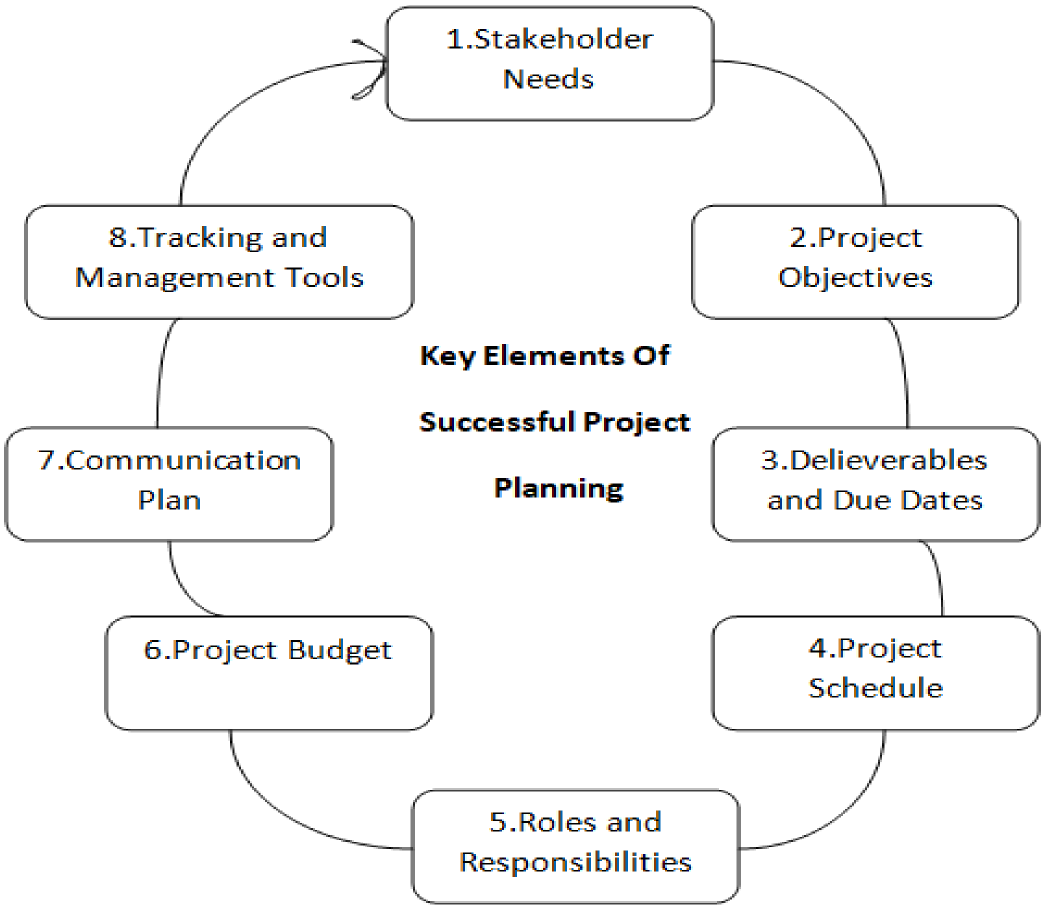

1.Stakeholder Needs
2.Project Objectives
3.Delieverables and Due Dates
4.Project Schedule
5.Roles and Responsibilities
6.Project Budget
7.Communication Plan
8.Tracking and Management Tools
Key Elements Of Successful Project Planning

Figure-17—Key Elements of Successful Project Planning.

To uncover the errors in the system, I have done testing as follows:--

1. Input Checking
2. Condition Testing
3. Loop Testing
4. Output Testing
5. Acceptance Testing

Testing Approaches—

There are three types of software testing approaches which are as follows--

1. White Box Testing
2. Black Box Testing
3. Grey Box Testing

Testing Levels—

There are three types of software testing levels which are as follows--

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing

Types of Black Box Testing—

1. Functionality Testing

2. Non-functionality Testing.

Software Debugging—

Software Debugging is the process of removing bugs, and errors from the software in order to make it work efficiently.

It may possible that software may contain some errors and bugs, so we need to remove them.

After removing these errors, software works efficiently.

Source Code—

```
import csv
import pandas as pd
import numpy as np
from collections import defaultdict
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
df=pd.read_csv("/kaggle/input/my-raw-data/raw_data.csv")
df.head()
# Fill all NaN with the values above
data = df.fillna(method='ffill')
# Process Disease and Symptom Names
def process_data(data):
    data_list = []
    data_name = data.replace('^','_').split('_')
    n = 1
    for names in data_name:
        if (n % 2 == 0):
            data_list.append(names)
        n += 1
    return data_list
# Data Cleanup
disease_list = []
disease_symptom_dict = defaultdict(list)
disease_symptom_count = {}
count = 0

for idx, row in data.iterrows():

    # Get the Disease Names
    if (row['Disease'] !="\xc2\xa0") and (row['Disease'] != ""):
        disease = row['Disease']
        disease_list = process_data(data=disease)
        count = row['Count of Disease Occurrence']

    # Get the Symptoms Corresponding to Diseases
    if (row['Symptom'] !="\xc2\xa0") and (row['Symptom'] != ""):
        symptom = row['Symptom']
        symptom_list = process_data(data=symptom)
        for d in disease_list:
            for s in symptom_list:
                disease_symptom_dict[d].append(s)
            disease_symptom_count[d] = count’
# Save cleaned data as CSV
f = open('./cleaned_data.csv', 'w')

with f:
    writer = csv.writer(f)
    for key, val in disease_symptom_dict.items():
        for i in range(len(val)):
            writer.writerow([key, val[i], disease_symptom_count[key]])
# Read Cleaned Data as DF
df = pd.read_csv('./cleaned_data.csv')
df.columns = ['disease', 'symptom', 'occurence_count']
df.head()
# Remove any rows with empty values
df.replace(float('nan'), np.nan, inplace=True)
df.dropna(inplace=True)
from sklearn import preprocessing
n_unique = len(df['symptom'].unique())
n_unique
df.dtypes
# Encode the Labels
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
```

```
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(df['symptom'])
print(integer_encoded)
# One Hot Encode the Labels
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
print(onehot_encoded)
cols = np.asarray(df['symptom'].unique())
cols
# Create a new dataframe to save OHE labels
df_ohe = pd.DataFrame(columns = cols)
df_ohe.head()
for i in range(len(onehot_encoded)):
    df_ohe.loc[i] = onehot_encoded[i]
# Disease Dataframe
df_disease = df['disease']
df_disease.head()
# Concatenate OHE Labels with the Disease Column
df_concat = pd.concat([df_disease,df_ohe], axis=1)
df_concat.head()
df_concat.drop_duplicates(keep='first',inplace=True)
df_concat.head()
cols = df_concat.columns
cols
cols=cols[1:]
```

```
# Since, every disease has multiple symptoms, combine all symptoms per disease
per row
df_concat = df_concat.groupby('disease').sum()
df_concat = df_concat.reset_index()
df_concat[:5]
df_concat.to_csv("./training_dataset.csv", index=False)
# One Hot Encoded Features
X = df_concat[cols]

# Labels
y = df_concat['disease']
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, export_graphviz
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=101)
len(X_train), len(y_train)
len(X_test), len(y_test)
dt = DecisionTreeClassifier()
clf_dt=dt.fit(X,y)
clf_dt.score(X, y)
export_graphviz(dt, out_file='./tree.dot',feature_names=cols)
!pip install graphviz
from graphviz import Source
from sklearn import tree
graph=Source(export_graphviz(dt, out_file=None, feature_names=cols))
png_bytes=graph.pipe(format='png')
with open('tree.png','wb') as f:
    f.write(png_bytes)

# from IPython.display import Image
# Image(png_bytes)
disease_pred=clf_dt.predict(X)
#disease_real=y.values
#realDisease="coronary arteriosclerosis"
#realDisease="Alzheimer's disease"
print(disease_pred[0])
realDisease=input("Please enter the Disease Name= ")
if(disease_pred[0]==realDisease):
    print("Success, The Disease is successfully predicted.\nThe Disease is=
\n",disease_pred[0])
else:
    print("You do not have ",realDisease," Disease.")
```

Chapter—6

Conclusion and Future Works

So, in a way, this model is very useful and is of great use to all the patients who are health conscious. One can get his / her results by simply sitting in homes without going anywhere. It easily predicts the class of the disease from which a person is suffering.
So, this model is very useful to all the persons and in all aspects. It provides all the medical services at a reasonable and affordable cost and with much ease.

Future Works--

The following features can make this model very effective --

In future, the main focus is to add new features to the model. Rather than predicting the type of disease, it would give solutions to that particular disease as well which will solve the problem of physical meet with the doctors.

The model will also be implemented in Android App also. The Android app will work same as this model which will have many additional features.

Refrences—

Websites--

[1]. https://wikipedia.org

[2]. https://www.softwaredevelopment.in

[3]. https://mlappdevelop.com

[4]. https://MLAppworld.com

[5]. https://en.wikipedia.org/wiki/

[6]. https://developer.org

[7]. http://diseaseprediction.com

[8]. www.help.com

[9]. www.justdial.com

[10]. https://patents.google.com.

Books--

[1]. Software Engineering – Beginners to Advanced.

[2]. Machine Learning Model Development

[3]. PL/SQL- by P. Steven.

[4]. Machine Learning Engineering.

Other References—

[1] Dr. A. K. Singh, " Machine Learning Model", IEEE Conference, Management System, Vol no.-2, PP-125-128, ISN- 187-165, ISO- 234:261, 08/01/2005.

[2] D. Kumar, "Research on ML Model Development", Scopus Journal, Manag. Services, Vol no.- 15, PP-105-108, 05/08/2016.

[3] Dr. Ashok Talwar, "Research Paper on Machine Learning Development", IEEE Conference, Vol no-18, ISSN-245-228, ISO- 125:162, 25/02/2012.

[4] Tom Alice, Bob Parker, " How to create an ML Model?".

[5] Peter Bob, "Research Paper Journal on Machine Learning" Scopus Conf., ISSN-223-252, ISO- 276:278 08/12/2008.

[6] Pradeep Singh, “Disease Predictor”, Service Booking System, Dream Press, ISN- 187-145, Aug, 2015.

[7] Harsh Kothari “A knowledge based system for Disease Prediction” IEEE Conf. on Android App and Computing Facilities, 16/12/2003.

[8] Ved Singh, “Design and Implementation of ML Model”, IEEE Conf., Java Handling Android Studio, 26/04/2013.

[9] Karan, “Research Paper on Medical Services”, IEEE International Conference on Information Technology, ISSN-145-128, ISO- 225:262, 07/02/2016.

[10] Dr. Chetrepal, “ML Model Development from beginning”, Scopus Journal, Vol no-6, PP- 345-365, ISN- 123-156, ISO- 345:386, 09/02/2018.

[11] B. L. Rao, “ Jupyter Notebook Programming from Beginning”, 19/03/2009.

[12] Naveen Pandit, “Research Paper Journal for Machine Learning Projects”,Suscom, Mang. Household Services, PP- 345-187, Vol no.-23, 18/01/2020.

[13] Machine Learning Projects for Beginners Step by Step Guide.

[14] Pradeep Kant- “Disease Prediction System”, IEEE Conference on Computer and Information Technology, ISSN-245-228, ISO- 125:162, Vol no-25, 05/01/2015.

[15] Nirmal Rao “ Research on the development of Machine Learning Projects”, IEEE Conf, ISSN-245-228, ISO- 125:162, Vol no.-18, PP- 245-254, 03/02/2015.

Thank You!!