# A SYNOPSIS REPORT

**ON**

**Disease Prediction using Machine Learning and Django and Online Consultation.**

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in “**Information Technology**”

|  |  |
| --- | --- |
|  | By: |
| **Name of Students** | **ID No** |
| Shubham Awhad | VU4F1718119 |
| Yash Tambe | VU4F1718108 |
| Akash Keny | VU4F1718106 |
|  |  |

Supervisor:

Prof. MEDHA KULKARNI

### Department of Information Technology

**Vasantdada Patil Pratishthan's College of Engineering & Visual Arts**

Vasantdada Patil Educational Complex, Eastern Express Highway, Near Everard Nagar, Sion- Chunabhatti, Mumbai – 400022.

### UNIVERSITY OF MUMBAI

**October, 2021**

# CERTIFICATE

This is to certify that the project entitled **“Disease Prediction using Machine Learning and Django and Online Consultation.”** is a bonafide work of **“Shubham Awhad” (VU4F1718119), “Yash Tambe” (VU4F1718108), “Akash Keny” (VU4F1718106)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in **“Information Technology”**.

Prof. MEDHA KULKARNI GUIDE

|  |  |
| --- | --- |
| Prof. Dr. RAHUL KHOKLE | Dr. ALAM N. SHAIKH |
| Head of Department | Principal and Campus Director |

## Project Report Approval for B. E.

This project report entitled “Disease Prediction using Machine Learning and Django and Online Consultation.” by (Shubham Awhad, Yash Tambe, Akash Keny) is approved for the degree of **“Bachelor of Engineering”** in **“Information Technology”**.

Examiners

1.

2.

Date:

Place: Mumbai - 22

## ABSTRACT

It might have happened so many times that you or someone yours need doctors help immediately, but they are not available due to some reason. The Health Prediction system is an end user support and online consultation project. Here we propose a system that allows users to get instant guidance on their health issues through an intelligent health care system online.

The system is fed with various symptoms and the disease/illness associated with those systems. The system allows users to share their symptoms and issues. It then processes the user's symptoms to check for various illnesses that could be associated with it. Here we use some intelligent data mining techniques to guess the most accurate illness that could be associated with a patient's symptoms.

In the doctor module when a doctor login to the system doctor can view his patient details and the report of that patient. Doctors can view details about the patient search and what the patient searched for according to their prediction. Doctor can view his personal details. Admin can add new disease details by specifying the type and symptoms of the disease into the database. Based on the name of the disease and symptom the data mining algorithm works. Admin can view various diseases and symptoms stored in the database. This system will provide proper guidance when the user specifies the symptoms of his illness.

Submitted by:

|  |  |
| --- | --- |
| **Name of Students** | **ID No** |
| SHUBHAM AWHAD | VU4F1718119 |
| YASH TAMBE | VU4F1718108 |
| AKASH KENY | VU4F1718106 |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TABLE OF CONTENTS** |  | |
| **Caption** | | **Page No** | |
| **List of Figures** | | | |
| **Chapter 1** | | | |
| **Introduction** | | **8-9** | |
| 1.1 | Introduction | 8 | |
|  | 1.1.1 Prediction | 9 | |
| 1.2 | Aim and Objectives | 9 | |
| 1.3 | Motivation for the work | 10 | |
| 1.4 | Scope of Project | 10 | |
| **Chapter 2** | | | |
| **Literature Survey and Problem Specification** | | **11-13** | |
| 2.1 | Introduction | 11 | |
| 2.2 | Existing System | 11 | |
|  | 2.2.1 Overall procedure | 12 | |
|  | 2.2.2 Advantages | 12 | |
|  |  |  | |
| 2.3 Need of New System | | 13 | |
| 2.4 | Problem Definition | 13 | |
| **Chapter** **3** | | | |
| **System Requirements** | | | **14** |
| 3.1 | Hardware Requirements |  | |
| **Chapter 4** | | | |
| **Design and Implementation** | | **15-19** | |
| 4.1 | Proposed System | 15 | |
|  | 4.1.1 Software Architectural Designs | 15 | |
| 4.2 | Design | 17 | |
|  | 4.2.1 use case Diagrams | 17 | |
| 4.3 | Algorithms | 18 | |
| **Chapter 5** | | | |
| **Screenshots** | | | **20-24** |
| 5.1 | Screenshots | 20 | |

|  |  |  |
| --- | --- | --- |
| **Chapter 6** | | |
| **Code** | | **25-30** |
| 6.1 | Code | 25 |
| 6.2 | User Interface Code | 28 |
| **Chapter 7** | | |
| **Conclusion** | | **31-31** |
| 7.1 | Conclusion | 31 |
| **Chapter 8** | | |
| **Reference** | | **32-32** |
| 8.1 | Reference | 32 |

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Caption** | **Page No.** |
| 3.2.1 | Architecture of System | 15 |
| 3.2.2 | FLOW CHART | 16 |
| 3.2.3 | K-MEANS ALGORITHM | 17 |

# CHAPTER 1

* 1. **Introduction**

**Introduction**

In today’s situation where there’s a covid outbreak and people having been rushing to healthcare institutions, Hospitals, Clinics have been overwhelmed. Doctor’s and medical professionals have been busy and some have no time to meet their patient’s. Patients who have not been infected are not been able to get diagnosed as COVID patients are on priority. This poses a problem because some symptoms maybe become fatal quickly. Hence, we propose to build a system which can predict disease based on symptoms given by patient and then the patient can consult the doctor online without leaving the comfort of their home.

### Prediction

Prediction can be used to automatically sort the cv. Prediction is used in situations where there are multiple possibilities to a single problem. There are many types of algorithms that can be usedk-means, agglomerative algorithm

* + - 1. Classification. When the data are being used to predict a category, supervised learning is also called classification. When there are only two choices, it's called two-class or binomial classification. That is clustering of the nearest approximate value.
      2. Regression. When a value is being predicted, as with stock prices, supervised learning is called regression.

### Aim and Objectives of the project

* + 1. **Aim**

Our aim is to build a system which can predict disease based on symptoms given by patient and then the patient can consult the doctor online without leaving the comfort of their home.

### Objective

1. To consult doctor without personally visiting doctor in clinic/hospital.
2. To provide correct disease as per the symptoms given by patients.

### Motivation for the work

1. Due to pandemic situation, it is rather difficult to visit doctors in their clinic for treatment.
2. There is a lot of paper trail if you visit the doctor frequently.
3. Doctors intend to get confused if any case paper is missing and might miss out on some important symptoms.
4. The patient can select the symptoms and get an idea about what exactly they are going through and contact the doctor in emergency.
5. The doctor will keep the record of the patient’s medical history in the database which can accessed and changed by the doctor only.

### Scope of the Project

### This project aims to provide a web platform to predict the occurrences of disease on the basis of various symptoms. The user can select various symptoms and can find the diseases with their probabilistic figures.

### 

# CHAPTER 2

## Literature Survey

### 1.1 Introduction:

In today’s situation where there’s a covid outbreak and people having been rushing to healthcare institutions, Hospitals, Clinics have been overwhelmed.

Doctors and medical professionals have been busy and some have no time to meet their patient’s.

Patients who have not been infected are not been able to get diagnosed as COVID patients are on priority. This poses a problem because some symptoms maybe become fatal quickly.

Hence, we propose to build a system which can predict disease based on symptoms given by patient and then the patient can consult the doctor online without leaving the comfort of their home.

### Existing System

### Overall procedure

* In current situation doctors and medical institutions have been overwhelmed and patients are not been able to get diagnosed due to larger number of people getting infected.
* The other problem is that the medical records of patient are written on paper and patient needs to take care of the record themselves.
* In case the record gets lost the doctor might lose the track of symptoms and not been able to diagnose a critical illness.
* So, to conclude there are two problems, one the doctors are busy and patients are missing out diagnosis other thing is that the medical records are not being stored.

.

### Advantages

* The patient can get immediate symptoms checked ad the system gives a confidence score.
* The patient can consult the doctor without having to leave the home.
* The system keeps track of patient’s history and diagnosis eliminating the need to maintain a file.
* The doctor and patient can keep a record of patient’s medical history and symptoms being developed.
* The patient can consult the doctor no matter where in the world the doctor is.
* The doctor and attend multiple patients at one time.
* An entire healthcare institution can be run virtually without even having a need to buy a property, construct a hospital/Clinic.
* The system can be useful wherein there is a lack of physical clinic or hospital.

### Need of New System

Limitations of Existing System:

The first problem is due to pandemic that doctors have been busy.

Patient’s record is stored in paper format and hence patient have to maintain a file.

In case the patient loses a paper, patient might need to do the diagnosis again.

The other problem is doctor is not able to keep a track of symptoms and hence might miss out a critical symptom.

Doctor is not able to consult multiple patients at a time.

### Problems Definition

The major objective is designing a system which can predict disease based on symptoms given by patient. and then the patient can consult the doctor online without leaving their home.

# CHAPTER 3

# System Requirements

# 3.1 Hardware Requirements

# Processor (CPU): Intel Core i5(sixth generation or newer) or any equivalent.

# Operating System: Microsoft Windows 10

# Memory: 8 GB RAM

# Storage: 500 GB internal storage drive.

# Monitor/Display: 14’’ LCD monitor, resolution of 1600 x 900 or better.

# Network Adapter: 802.11ac 2.4/5 GHz wireless Adapter

# CHAPTER 4

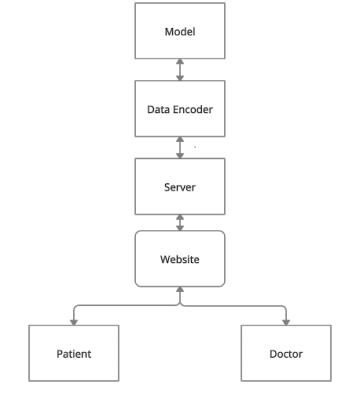
## Design and Implementation

### Proposed System

We propose to build a system in which patient gives symptoms and the disease is predicted. We plan to use test and train split technique of Machine Learning to train the model, we will download dataset of diseases from Kaggle and spilt it and use to first train the model and then test it. After successful prediction of the disease the patient is suggested to visit a doctor and given an option to consult the doctor with related specialty. Upon selecting the doctor, the patient can chat with the doctor and get diagnosed. The symptoms of patient are shown on the screen and doctor can consult accordingly. The consultation history is stored in patient profile as well as doctors’ profile.

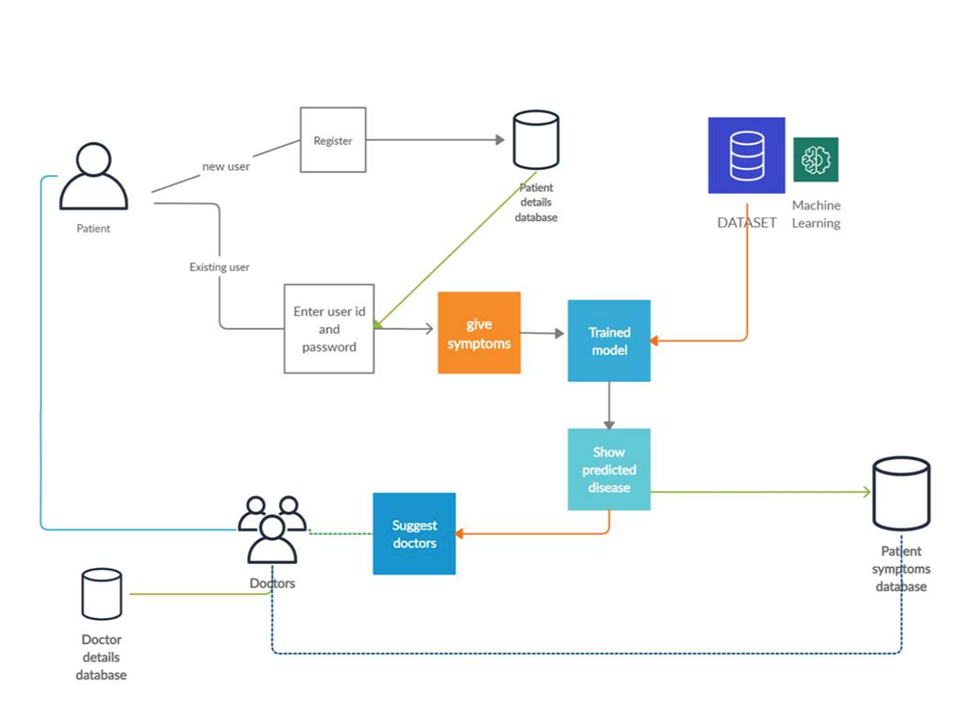
* + 1. Software Architectural Designs:

The software architecture follows an encoded path from model to hosted website on the user side, where the user has access to patient portal information and the doctors can see the database for all the previously evaluated symptoms and predictions, this in turn gives the doctor administrator rights to check and consult with the patient depending on the severity.



### Design

**4.2.1**

****

.

### Figure 4.2.1 Architecture of System

* 1. **Algorithms**

**Random Forest Algorithm**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

It takes less training time as compared to other algorithms.

It predicts output with high accuracy, even for the large dataset it runs efficiently.

It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

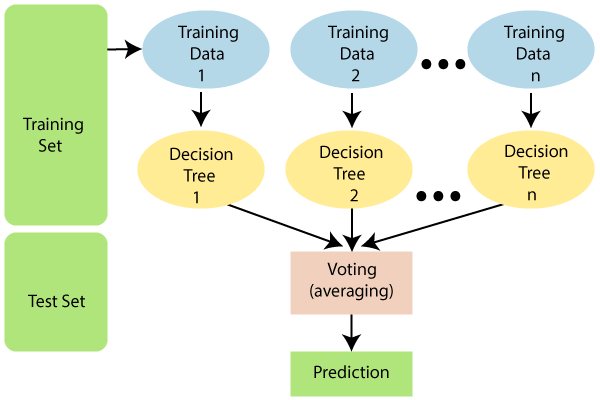
Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Advantages of Random Forest

Random Forest is capable of performing both Classification and Regression tasks.

It is capable of handling large datasets with high dimensionality.

It enhances the accuracy of the model and prevents the overfitting issue.



**DECISION TREE ALGORITHM**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

Why use Decision Trees?

Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.

The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Steps for Decision tree

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

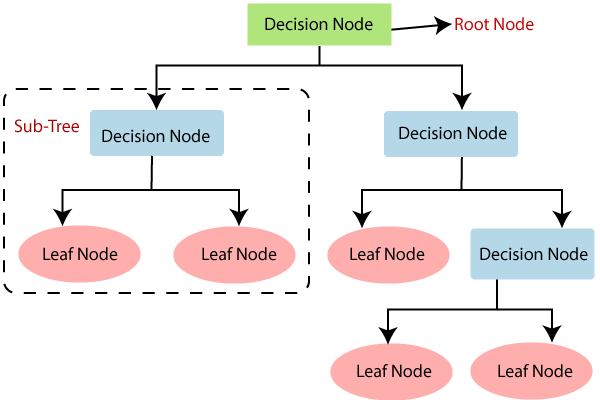
Advantages of Decision tree algo:

It is simple to understand as it follows the same process which a human follow while making any decision in real-life.

It can be very useful for solving decision-related problems.

It helps to think about all the possible outcomes for a problem.

There is less requirement of data cleaning compared to other algorithms.



# CHAPTER 5

## Screenshots

### 5.1 Screenshots

### 

### 

### 

### 

### 

### 

# CHAPTER 6

## Code

### 6.1 Code

### This is the configuration code and the main driver code to create and save the model to be used

### for the final prediction.

# This file is for training and saving model files # Import Dependencies

import yaml

from joblib import dump, load import pandas as pd

from sklearn.model\_selection import train\_test\_split from sklearn.metrics import accuracy\_score

# Trees Approach

from sklearn.tree import DecisionTreeClassifier

# Ensemble Approach

from sklearn.ensemble import RandomForestClassifier

class DiseasePrediction:

# Initialize and Load the Config File def init (self, model\_name=None):

# Load Config File try:

with open('./config.yaml', 'r') as f: self.config = yaml.safe\_load(f)

except Exception as e:

print("Error reading Config file...")

# Load Training Data

self.train\_features, self.train\_labels, self.train\_df = self.\_load\_train\_dataset()

# Load Test Data

self.test\_features, self.test\_labels, self.test\_df = self.\_load\_test\_dataset()

# Model Definition

self.model\_name = model\_name # Model Save Path

self.model\_save\_path = self.config['model\_save\_path']

# Function to Load Train Dataset def \_load\_train\_dataset(self):

df\_train = pd.read\_csv(self.config['dataset']['training\_data\_path']) cols = df\_train.columns

cols = cols[:-2]

train\_features = df\_train[cols]

train\_labels = df\_train['prognosis']

# Check for data sanity

assert (len(train\_features.iloc[0]) == 132)

assert (len(train\_labels) == train\_features.shape[0]) return train\_features, train\_labels, df\_train

# Function to Load Test Dataset def \_load\_test\_dataset(self):

df\_test = pd.read\_csv(self.config['dataset']['test\_data\_path']) cols = df\_test.columns

cols = cols[:-1]

test\_features = df\_test[cols]

test\_labels = df\_test['prognosis']

# Check for data sanity

assert (len(test\_features.iloc[0]) == 132)

assert (len(test\_labels) == test\_features.shape[0]) return test\_features, test\_labels, df\_test

# Dataset Train Validation Split def \_train\_val\_split(self):

X\_train, X\_val, y\_train, y\_val = train\_test\_split(self.train\_features,

self.train\_labels,

test\_size=self.config['dataset']['validation\_size'], random\_state=self.config['random\_state'])

return X\_train, y\_train, X\_val, y\_val

# Model Selection

def select\_model(self):

if self.model\_name == 'decision\_tree': self.clf =

DecisionTreeClassifier(criterion=self.config['model']['decision\_tree']['criteri

on'])

elif self.model\_name == 'random\_forest': self.clf =

RandomForestClassifier(n\_estimators=self.config['model']['random\_forest']['n\_es timators'])

return self.clf # ML Model

def train\_model(self):

# Get the Data

X\_train, y\_train, X\_val, y\_val = self.\_train\_val\_split()

classifier = self.select\_model() # Training the Model

classifier = classifier.fit(X\_train, y\_train)

# Trained Model Evaluation on Validation Dataset confidence = classifier.score(X\_val, y\_val)

# Validation Data Prediction

y\_pred = classifier.predict(X\_val) # Model Validation Accuracy

accuracy = accuracy\_score(y\_val, y\_pred) # Model Classification Report

clf\_report = classification\_report(y\_val, y\_pred)

print('\nTraining Accuracy: ', confidence) print('\nValidation Prediction: ', y\_pred) print('\nValidation Accuracy: ', accuracy)

print('\nClassification Report: \n', clf\_report)

# Save Trained Model

dump(classifier, str(self.model\_save\_path + self.model\_name + ".joblib"))

# Function to Make Predictions on Test Data

def make\_prediction(self, saved\_model\_name=None, test\_data=None): try:

".joblib"))

# Load Trained Model

clf = load(str(self.model\_save\_path + saved\_model\_name +

except Exception as e:

print("Model not found...")

if test\_data is not None:

result = clf.predict(test\_data) return result

else:

result = clf.predict(self.test\_features)

accuracy = accuracy\_score(self.test\_labels, result)

clf\_report = classification\_report(self.test\_labels, result) return accuracy, clf\_report

if name == " main ":

# Model Currently Training

current\_model\_name = 'random\_f' # Instantiate the Class

dp = DiseasePrediction(model\_name=current\_model\_name) # Train the Model

dp.train\_model()

# Get Model Performance on Test Data

test\_accuracy, classification\_report =

dp.make\_prediction(saved\_model\_name=current\_model\_name) print("Model Test Accuracy: ", test\_accuracy)

print("Test Data Classification Report: \n", classification\_report)

## 6.2 User Interface Code

The user interface is made using Streamlit, a fast lightweight data dashboard generating framework created using python.

# importing necessary modules import streamlit as st

import joblib

import pandas as pd import time

# loading model and list of symptoms

model = joblib.load("saved\_model/random\_f.joblib")

symptoms\_list = ['itching', 'skin\_rash', 'nodal\_skin\_eruptions',

'continuous\_sneezing', 'shivering', 'chills', 'joint\_pain', 'stomach\_pain', 'acidity', 'ulcers\_on\_tongue', 'muscle\_wasting', 'vomiting',

'burning\_micturition', 'spotting\_ urination', 'fatigue', 'weight\_gain', 'anxiety', 'cold\_hands\_and\_feets', 'mood\_swings', 'weight\_loss',

'restlessness', 'lethargy', 'patches\_in\_throat', 'irregular\_sugar\_level', 'cough', 'high\_fever', 'sunken\_eyes', 'breathlessness', 'sweating',

'dehydration', 'indigestion', 'headache', 'yellowish\_skin', 'dark\_urine', 'nausea', 'loss\_of\_appetite', 'pain\_behind\_the\_eyes', 'back\_pain',

'constipation', 'abdominal\_pain', 'diarrhoea', 'mild\_fever', 'yellow\_urine', 'yellowing\_of\_eyes', 'acute\_liver\_failure', 'fluid\_overload',

'swelling\_of\_stomach', 'swelled\_lymph\_nodes', 'malaise',

'blurred\_and\_distorted\_vision', 'phlegm', 'throat\_irritation',

'redness\_of\_eyes', 'sinus\_pressure', 'runny\_nose', 'congestion', 'chest\_pain', 'weakness\_in\_limbs', 'fast\_heart\_rate', 'pain\_during\_bowel\_movements',

'pain\_in\_anal\_region', 'bloody\_stool', 'irritation\_in\_anus', 'neck\_pain', 'dizziness', 'cramps', 'bruising', 'obesity', 'swollen\_legs',

'swollen\_blood\_vessels', 'puffy\_face\_and\_eyes', 'enlarged\_thyroid', 'brittle\_nails', 'swollen\_extremeties', 'excessive\_hunger',

'extra\_marital\_contacts', 'drying\_and\_tingling\_lips', 'slurred\_speech', 'knee\_pain', 'hip\_joint\_pain', 'muscle\_weakness', 'stiff\_neck',

'swelling\_joints', 'movement\_stiffness', 'spinning\_movements', 'loss\_of\_balance', 'unsteadiness', 'weakness\_of\_one\_body\_side', 'loss\_of\_smell', 'bladder\_discomfort', 'foul\_smell\_of urine',

'continuous\_feel\_of\_urine', 'passage\_of\_gases', 'internal\_itching', 'toxic\_look\_(typhos)', 'depression', 'irritability', 'muscle\_pain', 'altered\_sensorium', 'red\_spots\_over\_body', 'belly\_pain',

'abnormal\_menstruation', 'dischromic \_patches', 'watering\_from\_eyes',

'increased\_appetite', 'polyuria', 'family\_history', 'mucoid\_sputum', 'rusty\_sputum', 'lack\_of\_concentration', 'visual\_disturbances',

### 'receiving\_blood\_transfusion', 'receiving\_unsterile\_injections', 'coma',

'stomach\_bleeding', 'distention\_of\_abdomen', 'history\_of\_alcohol\_consumption', 'fluid\_overload.1', 'blood\_in\_sputum', 'prominent\_veins\_on\_calf',

'palpitations', 'painful\_walking', 'pus\_filled\_pimples', 'blackheads',

'scurring', 'skin\_peeling', 'silver\_like\_dusting', 'small\_dents\_in\_nails',

'inflammatory\_nails', 'blister', 'red\_sore\_around\_nose', 'yellow\_crust\_ooze']

# start of streamlit UI

st.title("Predico - The Smart health predictor") st.header("Please enter your symptoms ")

symptoms = st.multiselect('Enter your symptoms so that we can get you a primary diagnosis:',[\*symptoms\_list],key='symptoms')

# creating dataframe for accepting testing values prediction\_value = [0 for i in range(132)]

for sym in symptoms:

index = symptoms\_list.index(sym)

# assigning encoded value to testing frame prediction\_value[index] = 1

# convert list to Pandas dataframe and transpose it for model evaluation query = pd.DataFrame(prediction\_value).T

prediction = model.predict(query)

# evaluation and confirmation if st.button("Evaluate"):

with st.spinner('Predicting output...'): time.sleep(1)

if symptoms:

st.success("Prediction complete!")

st.write("The diagnosis we have reached is: ") st.error(\*prediction)

st.write("Please consult your nearest health administrator soon, take care! 🏥")

else:

st.info("Please enter at least one symptom before clicking

### evaluate)

# CHAPTER 7

## CONCLUSION

### 7.1 CONCLUSION

The methodology of maintaining medical record is quite primitive it’s update is long overdue.

With the implementation of this system healthcare industry will become safer and more efficient.

The fatality rate due to wrong diagnosis and tampered medical records will drastically fall down.

The patient can get diagnosed at home and from the best doctors no matter the distance between doctor and patient.

# CHAPTER 8

## References

### References

* + 1. Django Website for Disease Prediction using Machine Learning Journal of University of Shanghai for Science and Technology  
       Authors: Sayali Kulkarni, Isha Sawant, Megha Shinde, Vishal Sonar, Vaishali Latke
    2. Disease Prediction and Doctor Recommendation System using Machine Learning Approaches. International Journal for Research in Applied Science & Engineering Technology (IJRASET). Volume 9 Issue VII July 2021  
       Authors: Anand Kumar, Ganesh Kumar Sharma, U.M. Prakash
    3. Disease Prediction by Machine Learning Over Big Data FromHealthcare Communities. April 2017  
       Authors: Min Chen, Yixue Hao, Kai Wang, Lu Wang
    4. Intelligent Heart Disease Prediction on Physical and Mental Parameters: A ML Based IoT and Big Data Application and Analysis. 10 March 2020  
       Authors: Rohit Rastogi, D. K. Chaturvedi, Santosh Satya, Navneet Arora