A Major Project Report

On

**Data Analysis To Explore Demographic Information For Identifying COPD Patients**

Submitted in partial fulfillment of the

Requirements for the award of degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

by

**ISRAR AHMED KHAN**

(17H61A0517)

**NARAPONGU SRAVAN KUMAR**

(17H61A0531)

**V S S HARIKA KOUNDINYA**

(17H61A0556)

**Under the Guidance of**

**Mrs. B JYOTHI**

**Assistant Professor, Dept. of CSE**



**Department of Computer Science and Engineering**

**ANURAG GROUP OF INSTITUTIONS**

**(Formerly CVSR College of Engineering)**

**(An Autonomous Institution, Approved by AICTE and NBA Accredited)**

**Venkatapur (V), Ghatkesar (M), Medchal(D)., T.S-500088** **(2017-2021)**

****

**CERTIFICATE**

This is to certify that the project entitled **“DATA ANALYSIS TO EXPLORE DEMOGRAPHIC INFORMATION FOR IDENTIFYING COPD PATIENTS”**  being submitted by **Israr Ahmed Khan** bearing the Hall Ticket number **17H61A0517**, **Narapongu Sravan Kumar** bearing the Hall Ticket number **17H61A0531** and **V S S Harika Koundinya** bearing the Hall Ticket number **17H61A0556** in partial fulfillment of the requirements for the award of the degree of the **Bachelor of Technology** in **Computer Science and Engineering** to **Anurag Group of Institutions** **(Formerly** **CVSR College of Engineering)** is a record of bonafide work carried out by them under my guidance and supervision from April 2021 to July 2021.

The results presented in this project have been verified and found to be satisfactory. The results embodied in this project report have not been submitted to any other University for the award of any other degree or diploma.

**Internal Guide External Examinar**

**Mrs. B Jyothi**

**(Assistant Professor, Dept. of CSE)**

**Dr.G.Vishnu Murthy,**

**Professor and Head, Dept. of CSE**

**ACKNOWLEDGEMENT**

It is our privilege and pleasure to express profound sense of respect, gratitude and indebtedness to our guide **Mrs. B Jyothi, Assistant Professor**, Dept. of Computer Science and Engineering, Anurag Group of Institutions (Formerly CVSR College of Engineering), for his/her indefatigable inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout this dissertation work.

We express our sincere gratitude to **Dr. G.Vishnu Murthy, Professor & Head**, Department of Computer Science and Engineering, Anurag Group of Institutions (Formerly CVSR College of Engineering), for his suggestions, motivations and co-operation for the successful completion of the work.

We extend our sincere thanks to **Dr. V.Vijaya Kumar**, **Dean,** Research and Development, Anurag Group of Institutions, for his encouragement and constant help.

We extend our sincere thanks to **Dr. K.S.Rao, Director**, Anurag Group of Institutions for his encouragement.

Israr Ahmed Khan (17H61A0517)

Narapongu Sravan Kumar (17H61A0531)

V S S Harika Koundinya (17H61A0556)

**DECLARATION**

We hereby declare that the project work entitled “**Data Analysis To Explore Demographic Information For Identifying COPD Patients**” submitted to the **Anurag Group of Institutions(Formerly CVSR College of Engineering)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology (B.Tech)** in Computer Science and Engineering is a record of an original work done by us under the guidance of **Mrs. B Jyothi, Assistant Professor** and this project work have not been submitted to any other university for the award of any other degree or diploma.

Israr Ahmed Khan (17H61A0517)

Narapongu Sravan Kumar (17H61A0531)

V S S Harika Koundinya (17H61A0556)

Date:

**ABSTRACT**

Chronic Obstructive Pulmonary Disease (COPD) is a life-threatening lung disease and a major cause of morbidity and mortality worldwide. Monitoring of biomarkers that reflect the disease progression plays a pivotal role for the effective management of COPD. Hence, the accurate examination of respiratory tract fluids like saliva is a promising approach for staging disease and predicting its upcoming exacerbations in a Point-of-Care (PoC) environment. Here we propose a data analysis pipeline to analyze the saliva-metric & demographic data and use it for identification of COPD patients. The pipeline includes two phase data analysis i.e., descriptive, and predictive analysis. Descriptive analysis focus on statistically describing and visualizing the dataset. It helps in selecting the model for prediction that is useful for doctors.

|  |  |  |
| --- | --- | --- |
|  | **CONTENT** |  |
| **S.NO** | | **PAGE NO** |
|  | 1. Introduction | 1 |
|  | 1.1. Motivation | 1 |
|  | 1.2. Problem Definition | 1 |
|  | 1.3. Objective of the Project | 2 |
|  | 1. Literature Survey | 3 |
|  | 1. Analysis | 5 |
|  | 3.1. Existing System | 5 |
|  | 3.2. Proposed System | 5 |
|  | 3.3. Software Requirement Specification | 6 |
|  | 3.3.1 Purpose | 6 |
|  | 3.3.2 Scope | 7 |
|  | 3.3.3 Overall Description | 7 |
|  | 1. Design | 11 |
|  | 4.1. UML diagrams | 11 |
|  | 1. Implementation | 16 |
|  | 5.1. Modules | 16 |
|  | 5.2. Module description | 16 |
|  | 5.3. Introduction of technologies used | 17 |
|  | 5.4. Sample Code | 19 |
|  | 1. Test cases | 50 |
|  | 1. Screenshots | 52 |
|  | 1. Conclusion | 57 |
|  | 1. Future Enhancement | 58 |
|  | 10.       Bibliography | 59 |

**1. INTRODUCTION**

Chronic obstructive pulmonary disease (COPD) is a pulmonary disorder that causes respiratory problems in patients by restricting airflow to the lungs. It is a disorder that worsens over time. It develops slowly over time, and the symptoms often worsen. The chronic disorder is a leading causes of morality worldwide, affecting a large number of people and putting a tremendous financial strain on the healthcare system. The main reason for the pulmonary disorder is the prolonged exposure to cigarette smoke (active second-hand smokers) or other respiratory irritants such as toxins, industrial dust, or chemical pollutants. Alpha-1 antitrypsin deficiency is a genetic condition which causes can also lead to lung damage and subsequently COPD in rare cases. The most common sign of COPD is shortness of breath, persistent cough, wheezing, chest tension, and abnormal mucus development. While lung disease are irreversible, early intervention has been proven to be critical in the successful management of COPD.

**1.1. MOTIVATION**

COPD is the most prevalent lung disease in the world. Most of the times it is not diagnosed in first step. COPD treatment necessitates are recurrent visit to the medical practioners and sometimes prolonged stay in medical care facility. This has a significant impact on patient financial status and the quality of life. However, the proportion of person with COPD is less compared to cases with temporary lung disorder which has similar characteristics to COPD. So, a preliminary diagnostics system with saliva metrics and demographic data might be helpful in scrutinizing initial cases of COPD as opposed to other temporary disorder. This motivated us to design a prediction model for initial identification of COPD.

**1.2. PROBLEM DEFINITION**

Diagnosing COPD, A life-threatening lung disease caused in second-hand smokers (people who stop smoking after a while) and people who are exposed to industrial pollution in early stages has been detected as a major problem and the root cause for the increasing death rate in COPD patients. In the survey conducted by WHO, it is found out that the disease shows common symptoms like shortness of breath, wheezing, chest tightness, A chronic cough that may produce mucus (sputum) that may be clear, white, yellow, or greenish, frequent respiratory problems, lack of energy and swelling in ankles or legs. People usually don’t consult the doctor in such conditions either due to financial tensions or due to negligence which ultimately results in progression of the disease with time. In order to diagnose this problem in early stages a machine model is proposed that takes saliva bio-sensing values to identify if the person is subjected to COPD or not.

**1.3. OBJECTIVES OF THE PROJECT**

The objectives of the project can be described as:

* The key aim is to use saliva metric and demographic data to predict COPD.
* To detect the disease in early stages to minimize the effect on the subjected patients.
* Besides that, the complete technique can be used to know the degree of the disease caused in the patient that can be used to inform the criticality of the situation.

**2. LITERATURE SURVEY**

**1. Diagnosis of COPD based on knowledge graph and integrated model**

Chronic obstructive pulmonary disease (COPD) is a persistent lung infection that causes a reformist decrease in respiratory capacity. Diagnosing COPD in the early treatable stages is vital also, may even save the existence of a patient. In this paper, we present a coordinated model for diagnosing COPD in light of an information chart.

To start with, we build an information diagram of COPD to dissect the relationship between include subsets and further find the information on inferred infections from the information. Second, we propose a calculation for arranging highlights and a versatile element subset determination calculation CMFS-η, which chooses an ideal subset of highlights from the first high-dimensional set. At long last, the DSA-SVM incorporated model is recommended to assemble the classifier for the finding and forecast of COPD.

We performed broad tests on the dataset from the emergency clinic outpatient electronic clinical record data set. The order exactness of our strategy was 91.1%.

**2. Detection of Chronic Obstructive Pulmonary Disease in Computer**

**Aided Diagnosis System with CNN Classification**

The Chronic obstructive pulmonary disease (COPD) is a sort of noticeable ongoing illness. The principle factors that cause COPD sickness are breathing in dust, windedness, climate contamination, weariness and regular respiratory diseases.

COPD is represented by airshaft impediment coming about because of ongoing incendiary reactions in the aviation routes and harmful particles or gases. Airshaft obstacle is a normal part in Chronic obstructive pulmonary disease (COPD). Area of impediment and its assessing is fundamental.

PC Supported Analysis Framework helps the experts for clarification of clinical pictures. The PC Helped Analysis Framework has been proposed to conclusion the COPD by using CT pictures. Processed Tomography (CT) pictures are by and large picked because of less mutilation, less time utilization and negligible exertion.

The proposed work of electronic based analysis framework for a Chronic obstructive pulmonary disease (COPD) is to analysis the sickness with assurance utilizing convolutional neural network (CNN). CNN classifier to group the CT pictures and it will be assessed utilizing execution measurements. The PC Helped Finding Framework for COPD is made out of pre-processing, highlight extraction, division and arrangement. The pre-processing is to improve the nature of picture like eliminating commotion and confining locale of interest. The component extraction is a technique of catching visual substance of pictures for ordering and recovery. The division partitions the CT picture into various districts. The CNN classifier is to portray the divided CT pictures for upgrading the assurance of bunching under clamour.

**3. ANALYSIS**

**3.1. EXISTING SYSTEMS**

COPD is a lung discomfort in which the ventilation to the lungs is limited, making it impossible for patients to breathe. It's a slow-moving condition with signs that sometimes escalate over time. COPD is one of the leading causes of death globally, afflicting a large number of individuals and putting a tremendous financial strain on healthcare systems. Long-term exposure to tobacco smoke or other respiratory irritants like toxins, chemical vapours, or factory dust is the most common cause of COPD. Shortness of breath, persistent cough, wheezing, chest tension, and abnormal mucus development are the most common signs of COPD. Early detection can help in successful control of COPD.

* **Spirometer lung function tests** are the most simple and standardized means of diagnosing COPD, out of all the screening and diagnostic tools available. Spirometry is a procedure tests that tests the patient's lung volume. However as the sensitivity of spirometry test is limited to 64.5%-79.9% COPD is not diagnosed in number of vehicles. These methods have shown results in collecting traffic data.

**3.1.1. DRAWBACKS OF EXISTING SYSTEMS**

* Since the spirometer's intensity ranges from 64.5 to 79.9%, it is often misguided.
* It has a major drawback where COPD can’t be diagnosed in initial phases.

**3.2. PROPOSED SYSTEM**

The proposed system has three components:

* Data Cleaning
* Descriptive Analysis
* Predictive Analysis
* Data Cleaning includes missing value imputation.
* Descriptive Analysis is used to provide:
  + Class based statistical distribution of patterns.
  + Correlation analysis of saliva sample and demographic features with the class samples.
  + Exploratory analysis will focus on selection of predictive models.
  + Prediction analysis will focus on predicting COPD Patients.

Fig. 3.2.1 Block Diagram of proposed system

Descriptive Data Analysis

Model Construction

Predictive Data Analysis

Exploratory Result

Saliva Data

Training/ Validation

Testing

Model Selection

Data cleaning

* + 1. **ADVANTAGES OF PROPOSED SYSTEM**
* Advantageous technique over such as the use of complex sensor-based system and using any additional devices.
* Grayscale image has superior signal to noise ratio compared to RGB image.
* Only required object information is extracted using edge detection.

**3.3. SOFTWARE REQUIREMENT SPECIFICATION**

**TECHNOLOGY**

* KNN Algorithm
* Decision tree algorithm
* Gradient Boosting
* Python Language

**TOOLS**

* Anaconda: Spyder, Jupyter
* Numpy
* Pandas
* Matplotlib
* Sklearn

**3.3.1. PURPOSE**

As there is no proper system that diagnoses the lung disease in initial stages many people are battling with their lives. This creates tension among the family and disturbs people mentally which effects in the overall degradation of the lives of the people. Spirometry test and the other physical diagnosis systems are only effective when the disease has affected the person up to 65-70% and treating the disease in that scenario is vital and tough. The results of the person being cured is unpredictable and to minimize this tension and to give an intuition to the person about his/her health being affected through COPD is the prime purpose of the project.

**3.3.2. SCOPE**

A system that diagnoses COPD in initial stages is proposed. It needs saliva metrics of the person for which he/she must visit the nearest possible hospital and get the data which may take up to 1-2 days. The future scope of the project will be to collect the saliva sample images which will then be sent for image processing and through that the saliva metrics can be taken and then the COPD analysis and prediction can be done using the proposed algorithm.

**3.3.3. OVERALL DESCRIPTION**

**Dataset:**

Biosensor outputs for dielectric characterization of saliva samples for COPD, HC, and asthma are included in the data collection. Other details such as gender, age, and smoking status(smokers, former smokers and non-smokers) is also included in the data collection.

**Data cleaning:**

Data cleaning includes missing values imputation, deletion of duplicate rows and removing the outlays from the dataset.

**Data Imputation:**

Imputation is the primary task involved in our project. Imputation is done using Bayes’ Theorem.

**Bayes Theorem:**

The Bayesian hypothesis is a formula for calculating conditional probability, It calculates the likelihood of an event occurring if another event has already happened. Conditional probabilities have an additional requirement, which allows for more precise outcomes. To make correct predictions and probabilities.

The following actions are included in Bayes theorem implementation:

•  Consider the given example suppose X is a vector that has 'n' attributes where **X={x1,x2,x3,.....,xn}.**

•  Consider we have 'm’ classes**{C1,C2, .....,Cm}**. Our model has to identify X belong to a certain class. The class which gives highest posterior probability will be considered as the best class. The classifier would predict for class Ci iff **P(Ci | X) > P(Cj | X)** using Bayes Theorem

•  **P(Ci | X) = [ P(X | Ci) . P(Ci)] / P(X)**

•  P(X), is conditional-independent, it is constant for every class. To increase P(Ci | X), we have to increase **[P(X | Ci) . P(Ci)].** By assuming every class is equally probable, we have

**P(C1) = P(C2) = P(C3) . . . = P(Cn)**.

•  So therefore, we have to maximize P(X | Ci).

•  Since several common data sets are likely to have several attributes, performing a P(X|Ci) operation on each one is computationally expensive. The conditional independence of the class simplifies function and lowers accounting costs in this case. According to conditional class independence, attribute values are mutually exclusive.

•  **P(Xi | C) = P(x1 | C) . P(x2 | C) . . . . . P(xn | C)**

•  Now it is very easy to calculate the smaller probabilities. Note: Since xk belongs to each attribute, we have to confirm whether the the attribute we are using is categorical or continuous.

•  Everything is simpler if you have certain properties. You can count the number of instances of class Ci containing the value xk for attribute k, and then divide it by the number of instances of class Ci.

•  If any of the attribute is a continuous attribute we can us normal distribution function, we can apply the following formula, prior to applying the formula we have to calculate mean and the standard deviation.

•  We have all the values in that are used in Bayes Theorem for each class Ci. Our expected class will now have the best chance of succeeding ie. Highest probability

**P(X | Ci) \* P (Ci)**

**Descriptive Analysis:**  Descriptive Analysis is used to provide

•  Class based statistical distribution of patterns.

•  Correlation analysis of saliva sample and demographic features with the class samples.

•  Exploratory Analysis will focus on selection of predictive models.

**Prediction analysis:** Prediction analysis will focus on predicting COPD Patients.

**4. DESIGN**

**4.1. UML DIAGRAMS**

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

**What is UML?**

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it’s one of the most popular [business process modeling techniques](http://tallyfy.com/business-process-modeling-techniques).

It is based on diagrammatic representations of software components. As the old proverb says: “a picture is worth a thousand words”. By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

**What is the use of UML?**

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several [business processes](http://tallyfy.com/business-process) or [workflows](https://tallyfy.com/what-is-a-workflow/). For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency.

**Types of UML Diagrams**

There are several types of UML diagrams and each one of them serves a different purpose regardless of whether it is being designed before the implementation or after (as part of documentation).

**4.1.1. CLASS DIAGRAM**

Class UML diagram is the most common diagram type for software documentation. Since most software being created nowadays is still based on the [Object-Oriented Programming paradigm](https://en.wikipedia.org/wiki/Object-oriented_programming), using class diagrams to document the software turns out to be a common-sense solution. This happens because OOP is based on classes and the relations between them.

In a nutshell, class diagrams contain classes, alongside with their attributes (also referred to as data fields) and their behaviors (also referred to as member functions). More specifically, each class has 3 fields: the class name at the top, the class attributes right below the name, the class operations/behaviors at the bottom. The relation between different classes (represented by a connecting line), makes up a class diagram.

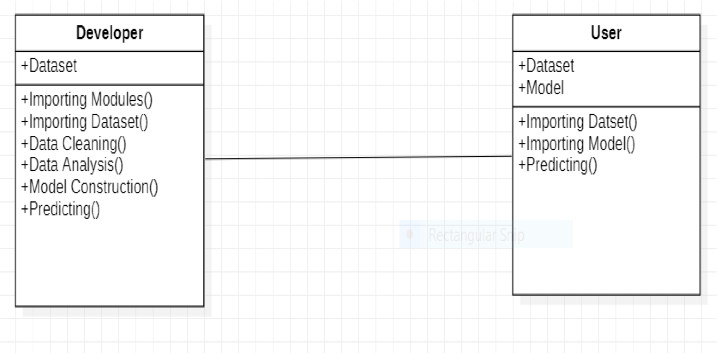
****

Fig. 4.1.1.1 Class Diagram

**4.1.2. USECASE DIAGRAM**

A cornerstone part of the system is the [functional requirements](https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/) that the system fulfills. Use Case diagrams are used to analyze the system’s [high-level requirements](http://www.testablerequirements.com/testablerequirements/ident_hlrs.htm). These requirements are expressed through different use cases. We notice three main components of this UML diagram:

* Functional requirements – represented as use cases; a verb describing an action
* Actors – they interact with the system; an actor can be a human being, an organization or an internal or external application
* Relationships between actors and use cases – represented using straight arrows

**Diagram

Description automatically generated**

Fig. 4.1.2.1 Use Case Diagram

**4.1.3. ACTIVITY DIAGRAM**

Activity diagrams are probably the most important UML diagrams for doing [business process modeling](https://tallyfy.com/business-process-modeling/). In software development, it is generally used to describe the flow of different activities and actions. These can be both sequential and in parallel. They describe the objects used, consumed or produced by an activity and the relationship between the different activities. All the above are essential in business process modeling.

**Diagram

Description automatically generated**

Fig. 4.1.3.1 Activity Diagram

**4.1.4. SEQUENCE DIAGRAM**

Sequence diagrams are probably the most important UML diagrams among not only the computer science community but also as design-level models for business application development. Lately, they have become popular in depicting business processes, because of their visually self-explanatory nature.

As the name suggests, sequence diagrams describe the sequence of messages and interactions that happen between actors and objects. Actors or objects can be active only when needed or when another object wants to communicate with them. All communication is represented in a chronological manner. To get a better idea, check the example of a UML sequence diagram below.

As the name suggests, structural diagrams are used to depict the structure of a system. More specifically, it is used in software development to represent the architecture of the system and how the different components are interconnected (not how they behave or communicate, simply where they stand).

**Chart, diagram, box and whisker chart

Description automatically generated**

Fig. 4.1.4.1 Sequence Diagram

**5.** **IMPLEMENTATION**

**5.1. MODULES**

* User
* Tkinter
* Matplotlib
* Numpy
* Gradient Boosting

**5.2. MODULE DESCRIPTION**

**User**

In this project, the first module is for users. They are made to upload image, start the process of detection. They can view the allocate time according the image uploaded.

**Tkinter**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

**Matplotlib**

The main purpose of Matplotlib is toascertain the results in the required graphical representations. There are various functions available in this module that allows us to plot different kinds of graphs. In this project, we have used this library for the sake of visualization of edge detection to get the accurate results.

**Numpy**

NumPy or Numerical Python is an open-source Python library that makes it easy to complex numerical operations. Working with machine learning and deep learning applications involve complex numerical operations with large datasets. NumPy makes implementing these operations relatively simple and effective when compared to their pure Python implementation.

**Gradient boosting**

Gradient boosting is a machine learning technique for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

**5.3. INTRODUCTION TO TECHNOLOGY USED**

**Python**

Python is certainly one of the best languages when working with Machine Learning and AI models as it has many built-in libraries which can be used directly without much implementation and code.

**Python Programming Language**

Python is a high-level, interpreted, interactive, and object-oriented scripting language. Python is designed to be highly readable. It uses English words frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

**Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

**Python is Interactive** − You can sit at a Python prompt and interact with the interpreter directly to write your programs.

**Python is Object-Oriented** − Python supports an Object-Oriented style or technique of programming that encapsulates code within objects.

**Python is a Beginner's Language** − Python is a great language for beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Gradient boosting**

Gradient boosting is a machine learning technique for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

**KNN Algorithm**

A supervised machine learning algorithm (as opposed to an unsupervised machine learning algorithm) is one that relies on labelled input data to learn a function that produces an appropriate output when given new unlabelled data.

**Decision Tree Algorithm**  
Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record’s attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

**5.4.** **SAMPLE CODE**

#Main.py

from \_\_future\_\_ import division

from PIL import ImageTk,Image

import pandas as pd

import numpy as np

import pandas as pd

from sklearn import \*

from datetime import datetime

from sklearn.svm import SVC

import seaborn as sns

import numpy as np

import seaborn as sns

import math

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from sklearn import svm

import os

import numpy as np

import pickle

import sys

import pyttsx3

from termcolor import colored

import datetime

import time

from datetime import timedelta

import csv

from tkinter import \*

from gtts import gTTS

import math

from tkinter import \*

from gtts import gTTS

# create tkinter window

root = Tk()

# styling the frame which helps to

# make our background stylish

frame1 = Frame(root,

bg = "#1f0036",

height = "150")

# plcae the widget in gui window

frame1.pack(fill = X)

frame2 = Frame(root,

bg = "#1f0036",

height = "750")

frame2.pack(fill=X)

# styling the label which show the text

# in our tkinter window

label = Label(frame1, text = "COPD Analysis System",

font = "bold, 30",

bg = "#f9f7cb")

label.place(x = 180, y = 70)

# entry is used to enter the text

name\_entry = Entry(frame2, width = 45,

bd = 4, font = 14)

name\_entry.place(x = 130, y = 52)

name\_entry.insert(0, "")

engine = pyttsx3.init()

def pyttsx3(text):

# obtain voice property

voices = engine.getProperty('voices')

# voice id 1 is for female and 0 for male

engine.setProperty('voice', voices[0].id)

# convert to audio and play

engine.say(text)

engine.runAndWait()

def submit():

name=name\_entry.get()

if name=="hi":

pyttsx3("Welcome to COPD Analysis System! Please wait to get the results")

play()

else:

exit(0)

def play():

import matplotlib.pyplot as plt

df=pd.read\_csv('data.csv')

df.columns = ['Diagnosis','Imaginary\_min','Imaginary\_avg','Real\_min','Real\_avg','Gender','Age','Smoking']

df.head() #first 5 rows

df[df['Diagnosis'] == 'COPD'].isnull().sum()

df[df['Diagnosis'] == 'HC'].isnull().sum()

df[df['Diagnosis'] == 'Infected'].isnull().sum()

df[df['Diagnosis'] == 'Asthma'].isnull().sum()

df.isnull().sum()

df.duplicated()

plt.figure(figsize=(14,14))

sns.heatmap(df.corr(),annot=True)

dfn=pd.notnull(df['Imaginary\_avg'])

dfnt=df[dfn]

dfnt

G0=df[df['Gender']==0].count().Gender

G1=df[df['Gender']==1].count().Gender

s1=df[df['Smoking']==1].count().Smoking

s2=df[df['Smoking']==2].count().Smoking

s3=df[df['Smoking']==3].count().Smoking

df1=df[df['Diagnosis'] == 'COPD']

y=df1.count().Diagnosis

print(y)

x=df.count().Diagnosis

print(x)

pcopd=y/x

ng0c=df1[df1['Gender']==0].count().Gender

ng1c=df1[df1['Gender']==1].count().Gender

ns1c=df1[df1['Smoking']==1].count().Smoking

ns2c=df1[df1['Smoking']==2].count().Smoking

ns3c=df1[df1['Smoking']==3].count().Smoking

pg=ng0c/G0

print(pg)

pg=ng1c/G1

print(pg)

ps=ns1c/s1

print(ps)

ps=ns2c/s2

print(ps)

ps=ns3c/s3

print(ps)

df2=df[df['Diagnosis'] == 'HC']

y1=df2.count().Diagnosis

print(y1)

x1=df.count().Diagnosis

print(x1)

ng0h=df2[df2['Gender']==0].count().Gender

ng1h=df2[df2['Gender']==1].count().Gender

ns1h=df2[df2['Smoking']==1].count().Smoking

ns2h=df2[df2['Smoking']==2].count().Smoking

ns3h=df2[df2['Smoking']==3].count().Smoking

pgh=ng0h/G0

print(pgh)

pgh=ng1h/G1

print(pgh)

psh=ns1h/s1

print(psh)

psh=ns2h/s2

print(psh)

psh=ns3h/s3

print(psh)

phc=y1/x1

df2=df[df['Diagnosis'] == 'Asthma']

y2=df2.count().Diagnosis

print(y2)

x2=df.count().Diagnosis

print(x2)

ng0a=df2[df2['Gender']==0].count().Gender

ng1a=df2[df2['Gender']==1].count().Gender

ns1a=df2[df2['Smoking']==1].count().Smoking

ns2a=df2[df2['Smoking']==2].count().Smoking

ns3a=df2[df2['Smoking']==3].count().Smoking

pga=ng0a/G0

print(pga)

pga=ng1a/G1

print(pga)

psa=ns1a/s1

print(psa)

psa=ns2a/s2

print(psa)

psa=ns3a/s3

print(psa)

p\_asthama=y2/x2

df3=df[df['Diagnosis'] == 'Infected']

y3=df3.count().Diagnosis

print(y3)

x3=df.count().Diagnosis

print(x3)

ng0i=df3[df3['Gender']==0].count().Gender

ng1i=df3[df3['Gender']==1].count().Gender

ns1i=df3[df3['Smoking']==1].count().Smoking

ns2i=df3[df3['Smoking']==2].count().Smoking

ns3i=df3[df3['Smoking']==3].count().Smoking

pgi=ng0i/G0

print(pgi)

pgi=ng1i/G1

print(pgi)

psi=ns1i/s1

print(psi)

psi=ns2i/s2

print(psi)

psi=ns3i/s3

print(psi)

p\_infected=y3/x3

dfcopd=dfnt[dfnt['Diagnosis']=='COPD']

mean\_age=dfcopd.Age.mean()

std\_age=dfcopd.Age.std()

mean\_img\_min=dfcopd.Imaginary\_min.mean()

std\_img\_min=dfcopd.Imaginary\_min.std()

mean\_img\_avg=dfcopd.Imaginary\_avg.mean()

std\_img\_avg=dfcopd.Imaginary\_avg.std()

mean\_real\_min=dfcopd.Real\_min.mean()

std\_real\_min=dfcopd.Real\_min.std()

mean\_real\_avg=dfcopd.Real\_avg.mean()

std\_real\_avg=dfcopd.Real\_avg.std()

dfhc=dfnt[dfnt['Diagnosis']=='HC']

mean\_age\_hc=dfhc.Age.mean()

std\_age\_hc=dfhc.Age.std()

mean\_img\_min\_hc=dfhc.Imaginary\_min.mean()

std\_img\_min\_hc=dfhc.Imaginary\_min.std()

mean\_img\_avg\_hc=dfhc.Imaginary\_avg.mean()

std\_img\_avg\_hc=dfhc.Imaginary\_avg.std()

mean\_real\_min\_hc=dfhc.Real\_min.mean()

std\_real\_min\_hc=dfhc.Real\_min.std()

mean\_real\_avg\_hc=dfhc.Real\_avg.mean()

std\_real\_avg\_hc=dfhc.Real\_avg.std()

dfasthama=dfnt[dfnt['Diagnosis']=='Asthma']

mean\_age\_asthama=dfasthama.Age.mean()

std\_age\_asthama=dfasthama.Age.std()

mean\_img\_min\_asthama=dfasthama.Imaginary\_min.mean()

std\_img\_min\_asthama=dfasthama.Imaginary\_min.std()

mean\_img\_avg\_asthama=dfasthama.Imaginary\_avg.mean()

std\_img\_avg\_asthama=dfasthama.Imaginary\_avg.std()

mean\_real\_min\_asthama=dfasthama.Real\_min.mean()

std\_real\_min\_asthama=dfasthama.Real\_min.std()

mean\_real\_avg\_asthama=dfasthama.Real\_avg.mean()

std\_real\_avg\_asthama=dfasthama.Real\_avg.std()

dfinfected=dfnt[dfnt['Diagnosis']=='Infected']

mean\_age\_infected=dfinfected.Age.mean()

std\_age\_infected=dfinfected.Age.std()

mean\_img\_min\_infected=dfinfected.Imaginary\_min.mean()

std\_img\_min\_infected=dfinfected.Imaginary\_min.std()

mean\_img\_avg\_infected=dfinfected.Imaginary\_avg.mean()

std\_img\_avg\_infected=dfinfected.Imaginary\_avg.std()

mean\_real\_min\_infected=dfinfected.Real\_min.mean()

std\_real\_min\_infected=dfinfected.Real\_min.std()

mean\_real\_avg\_infected=dfinfected.Real\_avg.mean()

std\_real\_avg\_infected=dfinfected.Real\_avg.std()

def normpdf(x, mean, sd):

var = float(sd)\*\*2

denom = (2\*math.pi\*var)\*\*.5

num = math.exp(-(float(x)-float(mean))\*(float(x)-float(mean))/(2\*var))

return num/denom

p=[]

for i in range(len(dfcopd)):

#print(dfcopd.iloc[i, 0], dfcopd.iloc[i, 5])

p\_img\_min=normpdf(dfcopd.iloc[i,1],mean\_img\_min,std\_img\_min)

p\_img\_avg=normpdf(dfcopd.iloc[i,2],mean\_img\_avg,std\_img\_avg)

p\_real\_min=normpdf(dfcopd.iloc[i,3],mean\_real\_min,std\_real\_min)

p\_real\_avg=normpdf(dfcopd.iloc[i,4],mean\_real\_avg,std\_real\_avg)

if(dfcopd.iloc[i,5]==0):

pg=ng0c/G0

if(dfcopd.iloc[i,5]==1):

pg=ng1c/G1

p\_age=normpdf(dfcopd.iloc[i,6],mean\_age,std\_age)

if(dfcopd.iloc[i,7]==1):

ps=ns1c/s1

if(dfcopd.iloc[i,7]==2):

ps=ns2c/s2

if(dfcopd.iloc[i,7]==3):

ps=ns3c/s3

px=float(p\_img\_min\*p\_img\_avg\*p\_real\_min\*p\_real\_avg\*pg\*p\_age\*ps\*pcopd)

#print(px)

p.append(px)

x=np.mean(p)

print(x)

p1=[]

for i in range(len(dfhc)):

p\_img\_min\_hc=normpdf(dfhc.iloc[i,1],mean\_img\_min\_hc,std\_img\_min\_hc)

p\_img\_avg\_hc=normpdf(dfhc.iloc[i,2],mean\_img\_avg\_hc,std\_img\_avg\_hc)

p\_real\_min\_hc=normpdf(dfhc.iloc[i,3],mean\_real\_min\_hc,std\_real\_min\_hc)

p\_real\_avg\_hc=normpdf(dfhc.iloc[i,4],mean\_real\_avg\_hc,std\_real\_avg\_hc)

if(dfhc.iloc[i,5]==0):

pg\_hc=ng0h/G0

if(dfhc.iloc[i,5]==1):

pg\_hc=ng1h/G1

p\_age\_hc=normpdf(dfhc.iloc[i,6],mean\_age\_hc,std\_age\_hc)

if(dfhc.iloc[i,7]==1):

ps\_hc=ns1h/s1

if(dfhc.iloc[i,7]==2):

ps\_hc=ns2h/s2

if(dfhc.iloc[i,7]==3):

ps\_hc=ns3h/s3

px\_hc=float(p\_img\_min\_hc\*p\_img\_avg\_hc\*p\_real\_min\_hc\*p\_real\_avg\_hc\*pg\_hc\*p\_age\_hc\*ps\_hc\*phc)

#print(px\_hc)

p1.append(px\_hc)

x\_hc=np.mean(p1)

print(x\_hc)

p2=[]

for i in range(len(dfasthama)):

p\_img\_min\_asthama=normpdf(dfasthama.iloc[i,1],mean\_img\_min\_asthama,std\_img\_min\_asthama)

p\_img\_avg\_asthama=normpdf(dfasthama.iloc[i,2],mean\_img\_avg\_asthama,std\_img\_avg\_asthama)

p\_real\_min\_asthama=normpdf(dfasthama.iloc[i,3],mean\_real\_min\_asthama,std\_real\_min\_asthama)

p\_real\_avg\_asthama=normpdf(dfasthama.iloc[i,4],mean\_real\_avg\_asthama,std\_real\_avg\_asthama)

if(dfasthama.iloc[i,5]==0):

pg\_asthama=ng0a/G0

if(dfasthama.iloc[i,5]==1):

pg\_asthama=ng1a/G1

p\_age\_asthama=normpdf(dfasthama.iloc[i,6],mean\_age\_asthama,std\_age\_asthama)

if(dfasthama.iloc[i,7]==1):

ps\_asthama=ns1a/s1

if(dfasthama.iloc[i,7]==2):

ps\_asthama=ns2a/s2

if(dfasthama.iloc[i,7]==3):

ps\_asthama=ns3a/s3

px\_asthama=float(p\_img\_min\_asthama\*p\_img\_avg\_asthama\*p\_real\_min\_asthama\*p\_real\_avg\_asthama\*pg\_asthama\*p\_age\_asthama\*ps\_asthama\*p\_asthama)

p2.append(px\_asthama)

x\_asthama=np.mean(p2)

print(x\_asthama)

p3=[]

for i in range(len(dfinfected)):

p\_img\_min\_infected=normpdf(dfinfected.iloc[i,1],mean\_img\_min\_infected,std\_img\_min\_infected)

p\_img\_avg\_infected=normpdf(dfinfected.iloc[i,2],mean\_img\_avg\_infected,std\_img\_avg\_infected)

p\_real\_min\_infected=normpdf(dfinfected.iloc[i,3],mean\_real\_min\_infected,std\_real\_min\_infected)

p\_real\_avg\_infected=normpdf(dfinfected.iloc[i,4],mean\_real\_avg\_infected,std\_real\_avg\_infected)

if(dfinfected.iloc[i,5]==0):

pg\_infected=ng0i/G0

if(dfinfected.iloc[i,5]==1):

pg\_infected=ng1i/G1

p\_age\_infected=normpdf(dfinfected.iloc[i,6],mean\_age\_infected,std\_age\_infected)

if(dfinfected.iloc[i,7]==1):

ps\_infected=ns1i/s1

if(dfinfected.iloc[i,7]==2):

ps\_infected=ns2i/s2

if(dfinfected.iloc[i,7]==3):

ps\_infected=ns3i/s3

px\_infected=float(p\_img\_min\_infected\*p\_img\_avg\_infected\*p\_real\_min\_infected\*p\_real\_avg\_infected\*pg\_infected\*p\_age\_infected\*ps\_infected\*p\_infected)

p3.append(px\_infected)

x\_infected=np.mean(p3)

print(x\_infected)

df.shape #rows and columns

from scipy.stats import invgauss

df=df.replace(np.nan,0)

count=0

for i in range(len(df)):

if(df.iloc[i,0]=='COPD'):

if(df.iloc[i,1]==0):

if(df.iloc[i,5]==0):

pg=ng0c/G0

if(df.iloc[i,5]==1):

pg=ng1c/G1

if(df.iloc[i,7]==1):

ps=ns1c/s1

if(df.iloc[i,7]==2):

ps=ns2c/s2

if(df.iloc[i,7]==3):

ps=ns3c/s3

m=math.log(x)-math.log(pcopd)-(math.log(normpdf(df.iloc[i,6],mean\_age,std\_age))+math.log(ps)+math.log(pg))

n=math.exp(m)

df.iloc[i,1]=invgauss.rvs(n,mean\_img\_min,std\_img\_min)

df.iloc[i,2]=invgauss.rvs(n,mean\_img\_avg,std\_img\_avg)

df.iloc[i,3]=invgauss.rvs(n,mean\_real\_min,std\_real\_min)

df.iloc[i,4]=invgauss.rvs(n,mean\_real\_avg,std\_real\_avg)

if(df.iloc[i,0]=='HC'):

if(df.iloc[i,1]==0):

if(df.iloc[i,5]==0):

pg\_hc=ng0h/G0

if(df.iloc[i,5]==1):

pg\_hc=ng1h/G1

if(df.iloc[i,7]==1):

ps\_hc=ns1h/s1

if(df.iloc[i,7]==2):

ps\_hc=ns2h/s2

if(df.iloc[i,7]==3):

ps\_hc=ns3h/s3

m\_hc=math.log(x\_hc)-math.log(phc)-(math.log(normpdf(df.iloc[i,6],mean\_age\_hc,std\_age\_hc))+math.log(ps\_hc)+math.log(pg\_hc))

n\_hc=math.exp(m\_hc)

df.iloc[i,1]=invgauss.rvs(n\_hc,mean\_img\_min\_hc,std\_img\_min\_hc)

df.iloc[i,2]=invgauss.rvs(n\_hc,mean\_img\_avg\_hc,std\_img\_avg\_hc)

df.iloc[i,3]=invgauss.rvs(n\_hc,mean\_real\_min\_hc,std\_real\_min\_hc)

df.iloc[i,4]=invgauss.rvs(n\_hc,mean\_real\_avg\_hc,std\_real\_avg\_hc)

if(df.iloc[i,0]=='Asthma'):

if(df.iloc[i,1]==0):

if(df.iloc[i,5]==0):

pg\_asthama=ng0a/G0

if(df.iloc[i,5]==1):

pg\_asthama=ng1a/G1

if(df.iloc[i,7]==1):

ps\_asthama=ns1a/s1

if(df.iloc[i,7]==2):

ps\_asthama=ns2a/s2

if(df.iloc[i,7]==3):

ps\_asthama=ns3a/s3

m\_asthama=math.log(x\_asthama)-math.log(p\_asthama)-(math.log(normpdf(df.iloc[i,6],mean\_age\_asthama,std\_age\_asthama))+math.log(ps\_asthama)+math.log(pg\_asthama))

n\_asthama=math.exp(m\_asthama)

df.iloc[i,1]=invgauss.rvs(n\_asthama,mean\_img\_min\_asthama,std\_img\_min\_asthama)

df.iloc[i,2]=invgauss.rvs(n\_asthama,mean\_img\_avg\_asthama,std\_img\_avg\_asthama)

df.iloc[i,3]=invgauss.rvs(n\_asthama,mean\_real\_min\_asthama,std\_real\_min\_asthama)

df.iloc[i,4]=invgauss.rvs(n\_asthama,mean\_real\_avg\_asthama,std\_real\_avg\_asthama)

if(df.iloc[i,0]=='Infected'):

if(df.iloc[i,1]==0):

if(df.iloc[i,5]==0):

pg\_infected=ng0i/G0

if(df.iloc[i,5]==1):

pg\_infected=ng1i/G1

if(df.iloc[i,7]==1):

ps\_infected=ns1i/s1

if(df.iloc[i,7]==2):

ps\_infected=ns2i/s2

if(df.iloc[i,7]==3):

ps\_infected=ns3i/s3

m\_infected=math.log(x\_infected)-math.log(p\_infected)-(math.log(normpdf(df.iloc[i,6],mean\_age\_infected,std\_age\_infected))+math.log(ps\_infected)+math.log(pg\_infected))

n\_infected=math.exp(m\_infected)

df.iloc[i,1]=invgauss.rvs(n\_infected,mean\_img\_min\_infected,std\_img\_min\_infected)

df.iloc[i,2]=invgauss.rvs(n\_infected,mean\_img\_avg\_infected,std\_img\_avg\_infected)

df.iloc[i,3]=invgauss.rvs(n\_infected,mean\_real\_min\_infected,std\_real\_min\_infected)

df.iloc[i,4]=invgauss.rvs(n\_infected,mean\_real\_avg\_infected,std\_real\_avg\_infected)

df.isnull().sum()

fig = plt.figure()

fig.patch.set\_facecolor('grey')

plt.pie(df['Diagnosis'].value\_counts(),colors=['red','black','blue','green'],labels=['HC','Asthama','Infected','COPD'])

plt.show()

column\_names = ['Imaginary\_min','Imaginary\_avg','Real\_min','Real\_avg','Gender','Age','Smoking','Diagnosis']

training\_data = df.reindex(columns=column\_names)

sns.countplot(x = "Gender",hue="Diagnosis", data = df)

df.head()

outputs = training\_data.iloc[: , -1]

inputs = training\_data.iloc[: , :-1]

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test = train\_test\_split(inputs,outputs,test\_size=0.30,random\_state=2)

print(X\_train.shape)

print(X\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

from sklearn.neighbors import KNeighborsClassifier

Knn=KNeighborsClassifier(n\_neighbors=6,metric='euclidean')

## apply the knn object on the dataset(training phase)

Knn.fit(X\_train, y\_train)

y\_train\_pred=Knn.predict(X\_train)

y\_train\_pred

from sklearn.metrics import accuracy\_score

scores=[]

for k in range(1,20):

knn\_model = KNeighborsClassifier(n\_neighbors=k)

knn\_model.fit(X\_train, y\_train)

pred\_test = knn\_model.predict(X\_test)

scores.append(accuracy\_score(y\_test, pred\_test))

scores

print("For k = {} accuracy is {}".format(scores.index(max(scores))+1,max(scores)))

final\_model=KNeighborsClassifier(n\_neighbors=1,metric='euclidean')

final\_model.fit(X\_train,y\_train)

final\_train\_pred=final\_model.predict(X\_train)

final\_train\_pred

knn\_train=accuracy\_score(y\_train,final\_train\_pred)

print(knn\_train)

final\_test\_pred=final\_model.predict(X\_test)

final\_test\_pred

knn\_test=accuracy\_score(y\_test,final\_test\_pred)

print(knn\_test)

#Import svm model

from sklearn import svm

#Create a svm Classifier

clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets

clf.fit(X\_train, y\_train)

clf.fit(X\_test,y\_test)

##syntax:objname.ppredict(input\_values)

y\_pred\_train=clf.predict(X\_train)

svm\_train=accuracy\_score(y\_train,y\_train\_pred)

print(svm\_train)

##syntax:objname.ppredict(input\_values)

y\_pred\_test=clf.predict(X\_test)

svm\_test=accuracy\_score(y\_test,y\_pred\_test)

print(svm\_test)

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()

dt.fit(X\_train,y\_train)

pred = dt.predict(X\_train)

dt\_train=accuracy\_score(y\_train,pred)

print(dt\_train)

test\_pred=dt.predict(X\_test)

print(test\_pred)

dt\_test=accuracy\_score(y\_test,test\_pred)

print(dt\_test)

# Fitting Naive Bayes to the Training set

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

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

pred = classifier.predict(X\_train)

nb\_train=accuracy\_score(y\_train,pred)

print(nb\_train)

test\_pred=dt.predict(X\_test)

nb\_test=accuracy\_score(y\_test,test\_pred)

print(nb\_test)

from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier()

gbc.fit(X\_train, y\_train)

pred=gbc.predict(X\_train)

gb\_train=accuracy\_score(y\_train,pred)

print(gb\_train)

test\_pred=gbc.predict(X\_test)

gb\_test=accuracy\_score(y\_test,test\_pred)

print(gb\_test)

acc=[]

acc.append(knn\_train\*100)

acc.append(knn\_test\*100)

acc.append(svm\_train\*100)

acc.append(svm\_test\*100)

acc.append(nb\_train\*100)

acc.append(nb\_test\*100)

acc.append(dt\_train\*100)

acc.append(dt\_test\*100)

acc.append(gb\_train\*100)

acc.append(gb\_test\*100)

print(' knn traning data accuracy = {}\n knn testing data accuracy = {}\n svm traning data accuracy = {} \n svm testing data accuracy = {}\n NB training data accuracy = {} \n NB testing data accuracy = {} \n DT training data accuracy = {} \n DT testing data accuracy = {} \n GB training data accuracy = {} \n GB testing data accuracy = {}'.format(acc[0],acc[1],acc[2],acc[3],acc[4],acc[5],acc[6],acc[7],acc[8],acc[9]))

import matplotlib.pyplot as plt

fig = plt.figure(figsize = (10,5))

algo=['KNN\_train','KNN\_test','svm\_train','svm\_test','nb\_train','nb\_test','dt\_train','dt\_test','gb\_train','gb\_test']

plt.bar(algo,acc,color=['red','green','red','green','red','green','red','green','red','green'])

plt.xlabel('Algorithms')

plt.ylabel('Accuracy')

plt.show()

testSet = [[-311,-305,-422,-445,1,67,1]]

test = pd.DataFrame(testSet)

result1=gbc.predict(test)

print('The final prediction on the random test set 1 is',result1)

pyttsx3("The final prediction on the random test set 1 is"+str(result1))

testSet2 = [[-323,-301,-405,-466,0,92,2]]

test = pd.DataFrame(testSet2)

result2=gbc.predict(test)

print('knn Prediction on the second random test set 2 is:',result2)

pyttsx3("The final prediction on the random test set 2 is"+str(result2))

btn = Button(frame2, text = "SUBMIT",

width = "15", pady = 10,

font = "bold, 15",

command = submit, bg='#f9f7cb')

btn.place(x = 250,

y = 130

)

# give a title

root.title("COPD Analysis System")

# we can not change the size

# if you want you can change

root.geometry("650x550+350+200")

# start the gui

root.mainloop()

**6.** **TESTING**

Testing is the process of evaluating a system or its components with the intent to find whether it satisfies the specified requirements or not. This activity results in the actual, expected and difference between their results i.e. testing is executing a system to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.

**6.1.** **TEST CASES**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Functionality | Initial State | Input | Expected Output | Actual Output | Status |
| Initialize system | Pause state | “Hi” | Receive Input | Receive Input | Success |
| Upload Saliva metrics | Pause state | Receiving the saliva metrics | Analysis of the model | Model Analysed | Success |
| Saliva metrics analysis | Load the appropriate model | Receiving the data and model | Receiving the data and model. | Receiving the data and model. | Success |
| Prediction | Pause state | The saliva metrics and the model are taken to analyse the patients data. | The prediction is done based on the saliva metrics comparison. | The prediction is done based on the saliva metrics comparison. | Success |

Table. 6.1.1 Test cases

**7.** **SCREENSHOTS**

**7.1. SCREENSHOTS OF OUTPUT**

**A screenshot of a computer

Description automatically generated with medium confidence**

Photo. 7.1.1 In above screen click hi and submit button to initialize the model



Photo. 7.1.2 In above graph the relation between numerical data is found out.

**Graphical user interface, chart, application, pie chart

Description automatically generated**

Photo. 7.1.3 In above screen the data set distribution is analyzed.

**A screenshot of a computer

Description automatically generated**

Photo. 7.1.4 In above screen the graph depicts the models used to process the system and the model selection is done based on the graph model vs accuracy.

**A screenshot of a computer

Description automatically generated**

Photo. 7.1.5 For the uploaded saliva metrics test set 1, test set 2 the prediction is made.

**7.2. ACCURACY OVERVIEW**

**Result Analysis**

In this section, the performance of the system is checked with respect to White Point Count, calculation of the Traffic Density. The Time Allocation for various images is accurate and it helps in performing the traffic control smoothly suggests the proposed system to be an efficient solution. The proposed system helps in reducing manpower required to operate the traffic signals. It reduces the need for additional hardware that might incur extra cost. The following table suggests the calculation of white point count, calculation of traffic density related to the white point count and time allocation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Models** | **Train** | **Test** | **Overall Accuracy** |
| KNN | 100% | 88% | 90% |
| Decision tree | 100% | 90% | 95% |
| Naive Bayes | 88.8% | 90% | 89% |
| SVM | 87% | 91% | 89% |

Table. 7.2.1 Models and their accuracy on the data used for the system.

**8. CONCLUSION**

The COPD prediction system for predicting COPD is successfully designed and implemented. The data was visualized using different plots which are computed to get a clear understanding of the dataset. For disease prediction a machine learning models was trained and tested successfully with significantly good metrics. The model is employed to make predictions on newer data. The study thus demonstrated the testing accuracies of different classifiers models on the dataset where Gradient Boosting Model got highest accuracy of 92.5 thus we can use this model for predicting COPD.

.

**9. FUTURE ENHANCEMENT**

In future of the project, we can implement other algorithms which give better performance metrics values. We can make the project available in the real time by integrating this model with the biosensor we can predict COPD instantly, the images of the saliva can be uploaded or taken instantly and they can be used for the further processing of the prediction.

**10.** **BIBLIOGRAPHY**

[[1] P. J. Barnes, ‘‘Mechanisms in COPD: Differences from asthma,’’ J. Chest, vol. 117, no. 2, p. 10S–14S, 2000.

[2] C. D. Mathers and D. Loncar, ‘‘Projections of global mortality and burden of disease from 2002 to 2030,’’ PLoS Med., vol. 3, no. 11, p. e442, Nov. 2006.

[3] N. G. Csikesz and E. J. Gartman, ‘‘New developments in the assessment of COPD: Early diagnosis is key,’’ Int. J. Chronic Obstructive Pulmonary Disease, vol. 9, pp. 277–286, 2014.

[4] T. Dong, S. Santos, Z. Yang, S. Yang, and N. E. Kirkhus, ‘‘Sputum and salivary protein biomarkers and point-of-care biosensors for the management of COPD,’’ Analyst, vol. 145, no. 5, pp. 1583–1604, 2020.

[5] S. Mirza, R. D. Clay, M. A. Koslow, and P. D. Scanlon, ‘‘COPD guidelines: A review of the 2018 GOLD report,’’ Mayo Clinic Proc., vol. 93, no. 10, pp. 1488–1502, Oct. 2018.

[6] S. Chiappin, G. Antonelli, R. Gatti, and F. Elio, ‘‘Saliva specimen: A new laboratory tool for diagnostic and basic investigation,’’ Clinica Chim. acta, vol. 383, no. 1, pp. 30–40, 2007.

[7] Bayes Theorem in Machine Learning: Bayes Theorem in Machine Learning: Introduction, How to Apply & Example | upGrad blog

[8] Gradient Boosting Algorithm: Gradient boosting - Wikipedia

[9] SVM Algorithm: SVM Algorithm | Working & Pros of Support Vector Machine Algorithm (educba.com).

[10] KNN Algorithm: KNN Algorithm - Finding Nearest Neighbors - Tutorialspoint

[11] D. Price, A. Crockett, M. Arne, B. Garbe, R. Jones, A. Kaplan, A. Langhammer, S. Williams, and B. Yawn, ‘‘Spirometry in primary care case-identification, diagnosis and management of COPD,’’ Primary Care Respiratory J., vol. 18, no. 3, pp. 216–223, Aug. 2009.

[12] S. Haroon, R. Jordan, Y. Takwoingi, and P. Adab, ‘‘Diagnostic accuracy of screening tests for COPD: A systematic review and meta-analysis,’’ BMJ Open, vol. 5, no. 10, Oct. 2015, Art. no. e008133.