

AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDYAPEETHAM

**BANGALORE 560 035**

**MODELLING DEMENTIA PROGRESSION USING GAUSSIAN HMM & MACHINE LEARNING**

A PROJECT REPORT

***Submitted by***

BL.EN.U4AIE21119 SIMHADRI TANYA

BL.EN.U4AIE21121 SRINIDHI KANNAN

BL.EN.U4AIE21144 VISHWASH SHARMA

For the subject

21BIO211 Intelligence of Biological Systems 4

Faculty Incharge

Mr. Shivakumar

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Content** | **Page Numbers** |
| 1 | OBJECTIVE | 3 |
| 2 | METHOD | 4-7 |
| 3 | RESULT |  |
| 4 | CONCLUSION |  |
| 5 | SUMMARY |  |

1. **OBJECTIVE**

The objective of the project is to track and understand the progress of dementia for earlier diagnosis and better treatment. Dementia is a neurodegenerative condition that refers to the loss or impairment of the daily functions of the human body like thinking, reasoning, remembering and behavior. Several diseases can be the cause of dementia. Dementia is usually associated with old age. The most common underlying cause for dementia is Alzheimer’s disease.

Since dementia is a progressive disease, that is, the symptoms are relatively mild at first and develop gradually over time, dementia is usually diagnosed at a later stage. Studying the disease can help in earlier diagnosis of the disease and developing better treatment methods.

Modelling and understanding the trajectory of dementia can help us diagnose and gain insights into the disease process at an earlier stage. So, we have developed a model to track the trajectory of the disease using a 2-state Gaussian Hidden Markov Model and Machine Learning.

To study the changes and transitions during dementia, patients with Mild Cognitive Impairment (MCI) are taken into consideration. Mild Cognitive Impairment is an early stage of memory loss or declining mental abilities like thinking and performing complex tasks.

Based on this data, a 2 state Gaussian Hidden Markov model is implemented. Since Hidden Markov model works on sequential data, it is suitable for tracking the progression and changes in a dementia patient.

But the HMM itself does not give desirable output as it cannot detect some of the patterns in the data as well as Machine Learning models do. Therefore, we have combined the HMM model with Machine Learning models to improve the accuracy of dementia prediction.

GUI and a Speech Recognition Tool have also been implemented to make it easier to use the model for dementia prediction by users. The GUI tells the user if he/she has dementia or not based on the data they enter into the GUI window. The Speech Recognition Tool further assists by advising the user on what is to be done if they are diagnosed with dementia.

‘

1. **METHOD**

To track the progression of dementia, we have combined two methods: a 2 state Gaussian Hidden Markov Model and Machine Learning models.

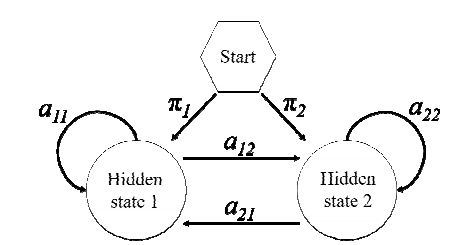
**2-State Gaussian Hidden Markov Model (HMM):**

***Markov model:*** A mathematical model that represents a chain of events in which the probability of transitioning to a state depends only on the probability of previous state and is independent of past history.

A markov model assumes that a patient is always in one of the states in the chain of events. Each event is the transition from one state to the next one.

***Hidden markov Model:***

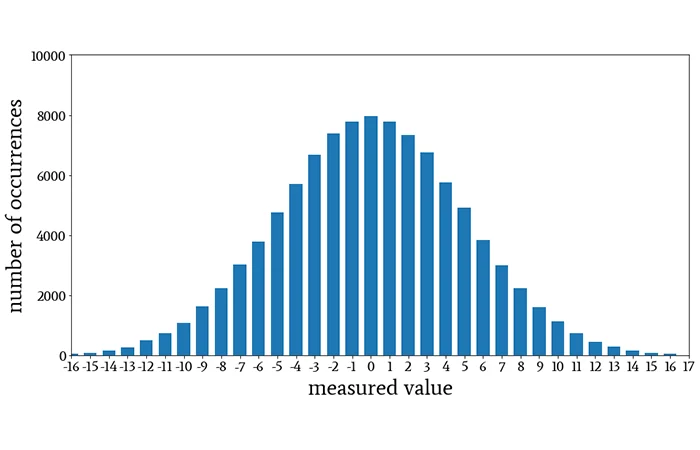
* It is an extension of the markovian model, where the state is hidden or unknown. Instead, the true state is to be calculated using observable symbols.
* Hidden Markov Models are particularly used in the bioinformatics field, since they are suitable for modelling sequential data.
* HMM is used to track disease progression when sequences of clinical parameters are taken from large dataset of patients.



Describes the 2 hidden states and the current state’s dependency on them

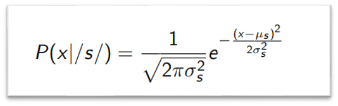
The data is obtained from MRI scans and cognitive tests for dementia. Some of the the parameters used in the dataset are: CDR (Clinical Dementia rating scale), Age, MMSE, estimated Total Intracranial Volume(eTIV) and Atlas Scaling Factor (ASF).

***Gaussian Distribution:***



* A probability distribution that tells us how data is spread or distributed around an average value. The curve obtained from this distribution is in form of a bell-shaped curve.
* A Gaussian distribution helps us understand how frequent the data is to around the mean/average and far from the average. The Probability Density Function (PDF) describes the relative likelihood of observing different values from the distribution.

The PDF function of a Gaussian Distribution is given by following equations:



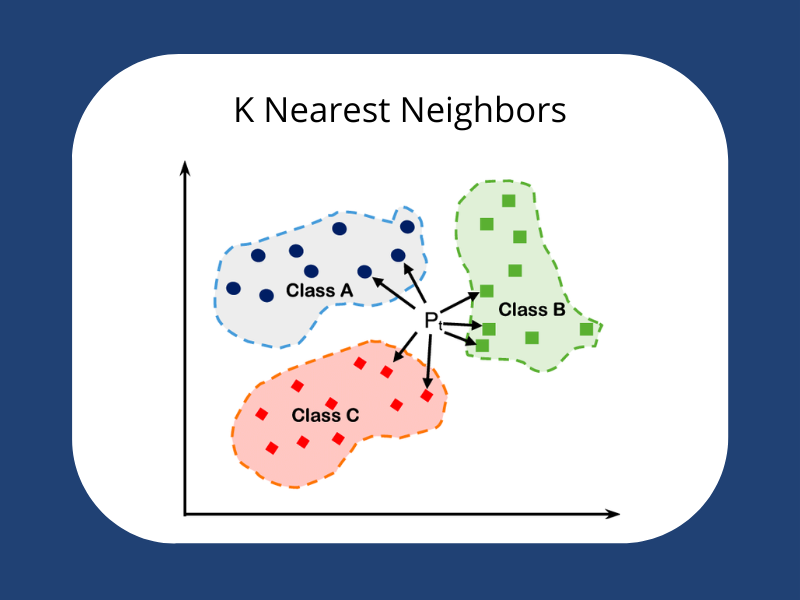
* The ***2-state Gaussian HMM*** consists of two hidden states, with each state representing an underlying stage. Based on the value in each hidden state the distribution of the observed variable, that is, the current state probability for that time step is described using Probability Density Function (PDF).
* The data is passed through this model and divided into several events and transitions to obtain the probability of final state. The probability of final state tells us whether the patient has dementia or not.
* Predictions are made by considering both the original features as well the extracted hidden features. This combined train and test data is stored as X\_train\_combined & X\_test\_combined.

Libraries used are: hmmlearn, sklearn, pandas

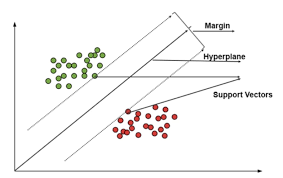
**Machine Learning Models:**

The machine learning models implemented in this project are:

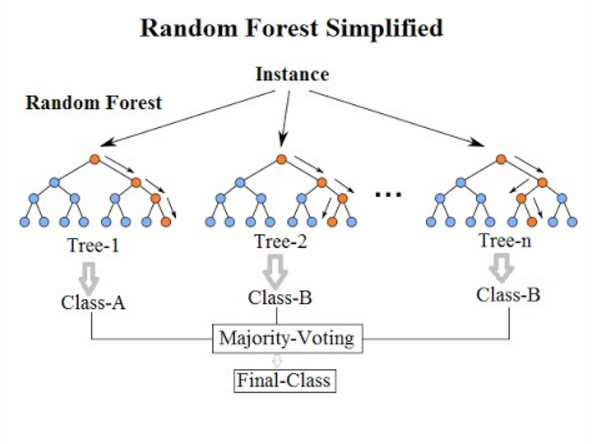
* K-Nearest Neighbour (KNN)
* Support Vector Machine (SVM)
* Decision Tree Classifier
* Random Forest Classifier
* ***K-Nearest Neighbour (KNN):*** It is a supervised machine learning algorithm that learns from the test samples and predicts output for the test sample. The deciding factor KNN model is the value of k. “K" represents the number of nearest neighbors to consider for making a prediction.



* ***Support Vector Machine (SVM):*** SVM is another supervised machine learning algorithm that makes use of a hyperplane to categorize data into groups. The hyperplane is chosen in such a way that the margin is maximum. This maximization provides SVM with good generalization ability and helps in dealing with noise.



* ***Decision Tree Classifier:*** Decision Tree has a hierarchical tree structure, consisting of root node, internal nodes and leaf nodes. It predicts the output by inferring rules for decision making for previous data and applying it on the test data. It is the best in dealing with hierarchical data.
* ***Random Forest Classifier:*** A supervised Machine learning model used for both regression and classification that fits a number of decision tree classifiers on various subsets of the dataset. It also uses averaging on the subsets to improve accuracy from Decision Tree Classifier.



The Machine learning algorithms are implemented using a Pipeline. It is a process in python that assembles several steps while setting different parameters. Pieline in python is imported from the built-in sklearn library.

The combined training and testing data obtained by finding the hidden variable data from 2-state Gaussian HMM is taken as input for the Machine Learning Models. After applying the following machine learning models, their accuracies are obtained.

Libraries used are:

* sklearn,
* pandas,
* numpy
* KNeighborsClassifier,
* SVC,
* DecisionTreeClassifier,
* RandomforestClassifier
* Pipeline

**3.CODE**

**import numpy as np**

**import pandas as pd**

**from hmmlearn import hmm**

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

**from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score**

**from sklearn.preprocessing import LabelEncoder**

**from sklearn.svm import SVC**

**from sklearn.neighbors import KNeighborsClassifier**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier**

**from sklearn.metrics import accuracy\_score**

**from sklearn.pipeline import Pipeline**

**from sklearn.preprocessing import StandardScaler**

**from tkinter import \***

**import speech\_recognition as sr**

**import openai**

**import pyttsx3**

**# Load the dataset**

**data = pd.read\_csv('C:/Users/sharm/OneDrive/Desktop/dementia\_dataset.csv')**

**# Replace zeros in 'MR Delay' and 'CDR' columns with NaN**

**data[['MR Delay', 'CDR']] = data[['MR Delay', 'CDR']].replace(0, np.nan)**

**# Drop rows with NaN values**

**data.dropna(inplace=True)**

**label\_encoder = LabelEncoder()**

**data['Group'] = label\_encoder.fit\_transform(data['Group'])**

**data['M/F'] = label\_encoder.fit\_transform(data['M/F'])**

**data['Hand'] = label\_encoder.fit\_transform(data['Hand'])**

**# Split the data into features (X) and target (y)**

**X = data.drop('Group', axis=1)**

**y = data['Group']**

**# Split the data into train and test sets**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**# Create and fit the Gaussian HMM model**

**n\_components = 2**

**hmm\_model = hmm.GaussianHMM(n\_components=n\_components, covariance\_type="diag", n\_iter=100)**

**hmm\_model.fit(X\_train)**

**# Perform predictions on the test set**

**predicted\_states = hmm\_model.predict(X\_test)**

**# Combine hidden states with original features**

**X\_train\_combined = np.column\_stack((X\_train, hmm\_model.predict(X\_train)))**

**X\_test\_combined = np.column\_stack((X\_test, predicted\_states))**

**# Create a Tkinter window**

**master = Tk()**

**master.title("Dementia Prediction")**

**label = Label(master, text="Dementia Prediction Using HMM & Machine Learning", bg="black", fg="white",**

**font=('times', 34)).grid(row=0, column=8000)**

**master.configure(bg='black')**

**master.geometry("2000x2000")**

**# Function to display the prediction result**

**def show\_prediction():**

**model = Pipeline([**

**('scaler', StandardScaler()),**

**('rf', RandomForestClassifier(max\_depth=5))**

**])**

**model.fit(X\_train\_combined, y\_train)**

**y\_pred = model.predict(X\_test\_combined)**

**if 0 in y\_pred:**

**result\_label.config(text="Non-Demented")**

**else:**

**result\_label.config(text="Demented")**

**# Function to process speech input**

**def process\_speech\_input():**

**with sr.Microphone() as source:**

**print("Listening...")**

**recognizer.adjust\_for\_ambient\_noise(source)**

**audio = recognizer.listen(source, timeout=5, phrase\_time\_limit=5)**

**try:**

**text = recognizer.recognize\_google(audio)**

**print("You said:", text)**

**if text.upper() == "BYE":**

**print("ChatGPT: Goodbye!")**

**engine.say("Goodbye!")**

**engine.runAndWait()**

**return True**

**# Pass the speech input to ChatGPT and get the model's response**

**model\_response = chat\_with\_gpt(text)**

**# Print the model's reply**

**print("ChatGPT:", model\_response)**

**engine.say(model\_response)**

**engine.runAndWait()**

**# Perform further processing or actions with the model\_response if needed**

**# ...**

**except sr.UnknownValueError:**

**print("Could not understand audio")**

**except sr.RequestError as e:**

**print("Error: {0}".format(e))**

**return False**

**# Function to interact with ChatGPT and get its response**

**def chat\_with\_gpt(message):**

**# Make an API call to ChatGPT**

**response = openai.Completion.create(**

**engine='text-davinci-003', # Replace with your desired ChatGPT model**

**prompt=message,**

**max\_tokens=50,**

**temperature=0.7**

**)**

**# Extract and return the generated response from ChatGPT**

**return response.choices[0].text.strip()**

**# Function to process speech input when the button is clicked**

**def process\_speech\_input\_button():**

**if process\_speech\_input():**

**master.quit()**

**# Create labels and entry fields in the Tkinter window**

**Label(master, text="Subject ID", font=('times', 22), bg="black", fg="white").grid(row=130, column=5000)**

**Label(master, text="MRI ID", font=('times', 22), bg="black", fg="white").grid(row=150, column=5000)**

**Label(master, text="Visit", font=('times', 22), bg="black", fg="white").grid(row=170, column=5000)**

**Label(master, text="MR Delay", font=('times', 22), bg="black", fg="white").grid(row=190, column=5000)**

**Label(master, text="M/F", font=('times', 22), bg="black", fg="white").grid(row=1110, column=5000)**

**Label(master, text="Hand", font=('times', 22), bg="black", fg="white").grid(row=1130, column=5000)**

**Label(master, text="Age", font=('times', 22), bg="black", fg="white").grid(row=1150, column=5000)**

**Label(master, text="EDUC", font=('times', 22), bg="black", fg="white").grid(row=1170, column=5000)**

**Label(master, text="SES", font=('times', 22), bg="black", fg="white").grid(row=1190, column=5000)**

**Label(master, text="MMSE", font=('times', 22), bg="black", fg="white").grid(row=1210, column=5000)**

**Label(master, text="CDR", font=('times', 22), bg="black", fg="white").grid(row=1230, column=5000)**

**Label(master, text="eTIV", font=('times', 22), bg="black", fg="white").grid(row=1250, column=5000)**

**Label(master, text="nWBV", font=('times', 22), bg="black", fg="white").grid(row=1270, column=5000)**

**Label(master, text="ASF", font=('times', 22), bg="black", fg="white").grid(row=1290, column=5000)**

**entry1 = Entry(master)**

**entry2 = Entry(master)**

**entry3 = Entry(master)**

**entry4 = Entry(master)**

**entry5 = Entry(master)**

**entry6 = Entry(master)**

**entry7 = Entry(master)**

**entry8 = Entry(master)**

**entry9 = Entry(master)**

**entry10 = Entry(master)**

**entry11 = Entry(master)**

**entry12 = Entry(master)**

**entry13 = Entry(master)**

**entry1.grid(row=130, column=9005)**

**entry2.grid(row=150, column=9005)**

**entry3.grid(row=170, column=9005)**

**entry4.grid(row=190, column=9005)**

**entry5.grid(row=1110, column=9005)**

**entry6.grid(row=1130, column=9005)**

**entry7.grid(row=1150, column=9005)**

**entry8.grid(row=1170, column=9005)**

**entry9.grid(row=1190, column=9005)**

**entry10.grid(row=1210, column=9005)**

**entry11.grid(row=1230, column=9005)**

**entry12.grid(row=1250, column=9005)**

**entry13.grid(row=1270, column=9005)**

**# Create a button to perform the prediction**

**predict\_button = Button(master, text="Predict", command=show\_prediction, font=('times', 22), bg="green",**

**fg="white").grid(row=1320, column=5000)**

**# Create a button to start speech recognition**

**speech\_recognition\_button = Button(master, text="Start Speech Recognition", command=process\_speech\_input\_button,**

**font=('times', 22), bg="blue", fg="white").grid(row=1360, column=5000)**

**# Create a label to display the prediction result**

**result\_label = Label(master, text="", font=('times', 22), bg="black", fg="white")**

**result\_label.grid(row=1400, column=9005)**

**# Initialize the speech recognizer and text-to-speech engine**

**recognizer = sr.Recognizer()**

**engine = pyttsx3.init()**

**# Start the Tkinter event loop**

**master.mainloop()**

**4.RESULT**

On giving the dataset as input, the given models are applied:

* The 2-state Gaussian Hidden Markov Model and
* the 4 Machine learning models: K-Nearest Neighbour, Support Vector Machine and Decision Tree Classifier on the data

The 2-state Gaussian HMM extracts the hidden variable data from the given data. This hidden data is combined with original factors and combined test and train dataset is obtained.

Machine learning models are applied on this combined data and accuracy for each one is stored.

To decide which model gives the most accurate results, a performance review has been conducted based on their accuracies.

We have also implemented a Graphical User Interface (GUI) using tkinter, so that the user can give input to the model easily. GUI automatically uses the model with best accuracy to predict the output for data entered by the user.

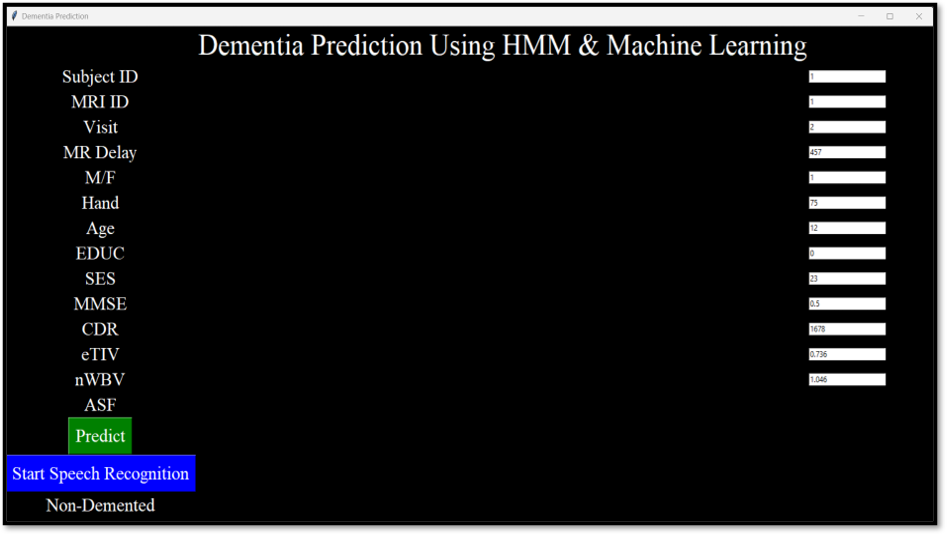
We have also included a Speech Recognition Tool, which assists the user if he/she has any queries regarding the disease.

If the user is diagnosed with dementia, he/she can use the speech recognition tool to seek advice on what is to be done next. The tool is linked with ChatGPT, an artificial intelligence chatbot that sends the answer in text format. This text is converted to speech and given as output to the user.

The GUI window includes: blanks for the user to fill his medical data in, a predict button (to obtain the prediction: demented or non-demented), a speech-recognition button to answer the user’s questions.

**OUTPUT**

Test case: 1, 1, 2, 457, 1, 1, 75, 12, 0, 23, 0.5, 1678, 0.736, 1.046



We have seen that random Forest Classifier gives the best accuracy in all of the models tested.

The output is printed at the bottom of the page: **Non-Demented**

**5.CONCLUSION**

Several methods are currently available for the diagnosis of dementia, like cognitive tests, Magnetic Resonance Imaging(MRI), genetic tests and blood tests. But these methods are not efficient enough to diagnose certain types of dementia at early stages.

Markov models have the ability to represent repetitive events and time dependencies of probability and utilities. This gives a more accurate representation of clinical settings for dementia.

But Markov models are unable to produce good accuracies. Hence, we have implemented Machine Learning into the existing 2-state Gaussian Hidden Markov Model. Implementing the methods together, we can leverage the strengths of both the models. HMM model for extraction of hidden features and Machine Learning to identify patterns in the original features.

Modelling of Dementia progression can be further improved by introduction of deep learning techniques, like recurrent neural networks. Obtaining a better dataset and applying several pre-processing methods can also improve the accuracy of the models.

**6. REFERENCES**

* C. Canavan, L. P. Maguire and M. Bucholc, "Development of a Two-State Gaussian Hidden Markov Model for Modelling Dementia Progression in Patients with Mild Cognitive Impairment," 2021 IEEE 9th International Conference on Healthcare Informatics (ICHI), Victoria, BC, Canada, 2021, pp. 113-119, doi: 10.1109/ICHI52183.2021.00028.
* Hidden Markov models for cancer classifification using gene expresion profiles. Thanh Nguyen ⇑ , Abbas Khosravi, Douglas Creighton, Saeid Nahavandi; Centre for Intelligent Systems Research (CISR), Deakin University, Waurn Ponds Campus, Victoria 3216, Australia