**INNOVATIVE PRODUCT DEVELOPMENT REPORT**

**DEPRESSION DETECTION , SURVEY AND RECOMMENDATION ENGINE**

**Submitted by:**

**A.PRAVARSHA ANANYA INALA B.PRAGATHI VAISHNAVI**

**23RH1A0530 23RH1A0540 23RH1A0563**

**Under the Esteemed Guidance of**

**Dr.N.Baskar**

**Associate Professor**

**Department of CSE**

**in partial fulfillment of the Academic Requirements for the Degree of**

**BACHELOR OF TECHNOLOGY**

# Computer Science and Engineering



**MALLA REDDY ENGINEERING COLLEGE FOR WOMEN**

*(Autonomous Institution-UGC, Govt. of India)*

**Accredited by NAAC with ‘A+’ Grade, UGC, Govt. of India | Programmes Accredited by**

**NBA National Ranking by NIRF Innovation-Rank band(151-300),MHRD, Govt. of India** Approved by AICTE, Affiliated to JNTUH,ISO 9001-2015 Certified Institution **Maisammaguda, Dhulapally, Secunderabad, Kompally-500 100.**

[**www.mallareddyecw.com**](http://www.mallareddyecw.com)

## 2024-2025

**MALLA REDDY ENGINEERING COLLEGE FOR WOMEN**

*(Autonomous Institution-UGC, Govt. of India)*

Accredited by NAAC with ‘A+’ Grade, UGC, Govt. of India | Programmes Accredited by

NBA National Ranking by NIRF Innovation-Rank band(151300),MHRD, Govt. of

India Approved by AICTE, Affiliated to JNTUH,ISO 90012015 Certified

Institution, Maisammaguda, Dhulapally, Secunderabad, Kompally-500 100.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## **CERTIFICATE**

This is to certify that the work embodies in this project entitled “**DEPRESSION DETECTION USING EEG**” being submitted by **A.PRAVARSHA(23RH1A0530), ANANYA INALA(23RH1A0540), B.PRAGATHI VAISHNAVI(23RH1A0563)** for partial fulfillment of the requirement for the award of BACHELOR OF TECHNOLOGY in Computer Science and Engineering discipline to Malla Reddy Engineering College for Women, Maisammaguda, Secunderabad during the academic year 2024-2025 is a record of bonafide piece of work, undertaken by his supervision of the undersigned.

**Supervisor’s Signature Head of the Department**

**Dr.N. BASKAR Dr .Y.GEETHA REDDY**

Associate Professor Associate Professor

**EXTERNAL EXAMINER**

**MALLAREDDY ENGINEERING COLLEGE FOR WOMEN**



**(Autonomous Institution-UGC, Govt. of India)**

**Accredited by NBA &NAAC with ‘A’ Grade UGC, Govt. of India**

**National Ranking by NIRF Innovation – Rank Band (151-300), MHRD, Govt. of**

**India ,Maisammaguda, Dhulapally, Secunderabad-500100**

### **DECLARATION**

We hereby declare that our project entitled DEPRESSION DETECTION ,SURVEY AND RECOMMENDATION ENGINE submitted to Malla Reddy **Engineering College for Women, Hyderabad** for the award of the Degree of Bachelor of Technology in **Computer Science and Engineering** is a result of original project work done by us.It is declared that the project report or any part there of has not been previously submitted to any University or Institute for the award of Degree.

A.PRAVARSHA (23RH1A0530)

ANANYA INALA(23RH1A0540)

B.PRAGAGTHI VAISHNAVI (23RH1A0563)

## **ACKNOWLEDGEMENT**

We feel ourselves honoured and privileged to place our warm salutation to our college Malla Reddy Engineering College for Women and Department of Computer Science and Engineering which gave us the opportunity to have Expertise in engineering and profound technical knowledge.

We would like to deeply thank our Honorable Minister of Telangana State **Sri CH.MALLA REDDY Garu**, Founder Chairman MRGI, the largest cluster ofinstitutions in the state of Telangana for providing us with all the resources inthe college to make our project success.

We wish to convey gratitude to our Principal **Dr. Y. MADHAVEE LATHA**, for providing us with the environment and mean to enrich our skills and motivating us in our endeavor and helping us to realize our full potential.

We express our sincere gratitude to **Dr . Y.GEETHA REDDY**, Head of the Department of Computer Science and Engineering for inspiring us to take up a project to this subject and successfully guiding us towards its completion.

We would like to thank our Internal Guide **Dr.BASKAR N**, Associate Professor

and all the Faculty members for their valuable guidance and encouragement towards the completion of our project work.

**With Regards and Gratitude,**

A.PRAVARSHA (23RH1A0530)

ANANYA INALA(23RH1A0540)

B.PRAGAGTHI VAISHNAVI (23RH1A0563)

**ABSTRACT**

Depression is one of the most prevalent mental health disorders globally, affecting millions of individuals across all age groups. Early detection and intervention are crucial for improving recovery outcomes. Traditional methods for diagnosing depression, such as clinical interviews and self-reported questionnaires, can be time-consuming, subjective, and prone to human error. In recent years, Electroencephalogram (EEG) signals have emerged as a promising biomarker for identifying neurological patterns associated with depression. However, manual analysis of EEG data is labor-intensive and inefficient for large-scale applications.

This study presents an automated, hybrid system for depression detection using deep learning techniques and a survey-based recommendation engine. EEG datasets are processed and analyzed using a Convolutional Neural Network (CNN), which extracts spatial features from the EEG waveforms and classifies individuals as either “Normal” or “Depressed.” The CNN architecture is designed to handle multidimensional EEG inputs and improve prediction accuracy over traditional machine learning approaches such as Support Vector Machines (SVM).

To enhance the reliability of predictions and provide personalized support, the system also integrates a subjective mental health survey. Users are prompted to answer a series of scaled questions that reflect their mood, energy levels, sleep patterns, and interest in daily activities. Based on the average score of the survey responses, the system provides tailored mental health recommendations ranging from lifestyle tips to professional consultation suggestions.

The proposed system not only automates the detection of depression using EEG but also incorporates user input to enhance the accuracy and relevance of results. This approach can assist healthcare professionals in early screening and personalized care delivery, especially in scenarios where traditional diagnostic resources are limited. Overall, the fusion of EEG analysis and survey-based insights offers a practical and scalable solution for mental health monitoring.

### **INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CHAPTER** | **Page no.** |
| 1 | **CHAPTER-I** |  |
|  | 1.INTRODUCTION | 13 |
| 2 | **CHAPTER-II** |  |
|  | 2.SYSTEM ANALYSIS | 14 |
|  | 2.1 EXISTING SYSTEM |  |
|  | 2.2 PROPOSED SYSTEM |  |
| 3 | **CHAPTER-III** |  |
|  | SYSTEM REQUIREMENTS | 15 |
|  | 3.1 SOFTWARE REQUIREMENTS |  |
|  | 3.2 HARDWARE REQUIREMENTS |  |
| 4 | **CHAPTER-IV** |  |
|  | 4. SYSTEM DESIGN | 16 |
|  | 4.1 BLOCK DIAGRAM |  |
|  | 4.2 UML DIAGRAM |  |
| 5 | **CHAPTER-V** |  |
|  | 5**. SYSTEM IMPLEMENTATION** | 23 |
|  | 5.1 PYTHON |  |
|  | 5.2 MODULES USED |  |
| 6 | **CHAPTER-VI** |  |
|  | **6. RESULT AND ANALYSIS** | 31 |
|  | 6.1 TESTING |  |
|  | 6.2 SCREEN SHOTS |  |
| 7 | **CHAPTER-VII** |  |
|  | **7. CONCULSION** | 43 |
| 8 | **CHAPTER-VIII** |  |
|  | **8. REFERENCES** | 44-45 |

**CHAPTER-I**

1. **INTRODUCTION**

It has been announced by the World Health Organization (WHO) that according to estimates, more than 322 million people worldwide suffer from depression, making it the leading contributor to global disability. Depressive patients typically experience symptoms such as persistent sadness, hopelessness, guilt, loss of interest, difficulty concentrating, fatigue, and disruptions in appetite, sleep, and daily routines. Contributing factors include poverty, unemployment, traumatic life events, physical health issues, and substance abuse. Recently, the COVID-19 pandemic has exacerbated depression rates through conditions such as lockdowns, quarantines, and prolonged social distancing.

Given the unprecedented threat depression poses to public health, and the potential consequences such as suicidal tendencies, early diagnosis becomes crucial in facilitating timely and effective treatment. Electroencephalogram (EEG) signals, which capture the brain's electrical activity, are non-invasive, nonlinear, and complex in nature. Their intrinsic properties make them a promising yet challenging tool for depression detection, as abnormalities are difficult to detect through manual inspection.

Deep learning, with its hierarchical structure and multiple processing layers, enables systems to automatically learn complex patterns from raw data. Unlike traditional machine learning methods which often require manual feature extraction, deep learning models such as Convolutional Neural Networks (CNNs) are capable of autonomously discovering subtle patterns in EEG signals, making them highly suitable for depression diagnosis.

In this work, we propose an advanced system that combines EEG-based deep learning depression detection with an integrated survey-based recommendation engine. After classifying the EEG data using a CNN model, users are guided through a short psychological survey to validate the diagnosis and assess subjective mental health states. Based on the combined result, the system provides personalized mental health recommendations—ranging from lifestyle tips to suggesting professional help—thus offering a holistic and user-centered approach to early depression detection and care.

**CHAPTER-II**

**2. SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM**

Depression is a mood disorder that can be detected using electroencephalogram (EEG) signals. The manual detection of depression by analyzing EEG signals requires significant expertise, is tedious, and time-consuming. Traditional approaches involve either manual psychiatric counselling or the use of older machine learning methods such as Support Vector Machines (SVM), which often fail to capture the nonlinear and complex patterns present in EEG data, resulting in lower accuracy and delayed diagnosis.

**Disadvantage**

* Manual analysis is time-consuming and requires domain expertise.
* Less Accuracy in predictions due to limitations of traditional algorithms.
* Lack of real-time feedback or personalized guidance post-diagnosis.

**2.2 PROPOSED SYSTEM**

The proposed system uses a deep learning Convolutional Neural Network (CNN) to automatically detect depression from EEG signals. Unlike traditional methods, CNN extracts features through multiple layers, improving prediction accuracy. Additionally, a psychological survey is integrated post-prediction to gather subjective user input. Based on the survey results and CNN output, the system provides personalized mental health recommendations such as lifestyle tips or therapy suggestions. This hybrid approach enhances early detection and offers holistic support by combining objective EEG analysis with subjective feedback.

**Advantage**

* More Accuracy in depression prediction using deep learning CNN.
* Automated feature extraction from raw EEG signals.
* Integration of psychological survey adds a subjective validation layer.
* Personalized mental health recommendations based on survey scores.
* Supports early detection and encourages timely intervention.

**CHAPTER III**

**SYSTEM REQUIREMENTS**

**3.1 HARDWARE REQUIREMENTS**

• Processor - Intel i3(min)

• Speed - 1.1 GHz

• RAM - 4GB(min)

• Hard Disk - 500 GB

• Key Board - Standard Windows Keyboard

• Mouse - Two or Three Button Mouse

• Monitor - SVGA

**3.2 SOFTWARE REQUIREMENTS**

• Operating System - Windows10(min)

• Programming Language - Python

**CHAPTER IV**

**SYSTEM DESIGN**

**FEASIBILITY STUDY**

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

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* OPERATIONAL FEASIBILITY
* TECHNICAL FEASIBILITY

**ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

**OPERATIONAL FEASIBILITY**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

**TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**4.1 BLOCK DIAGRAM**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

****

**4.2UML DIAGRAMS**

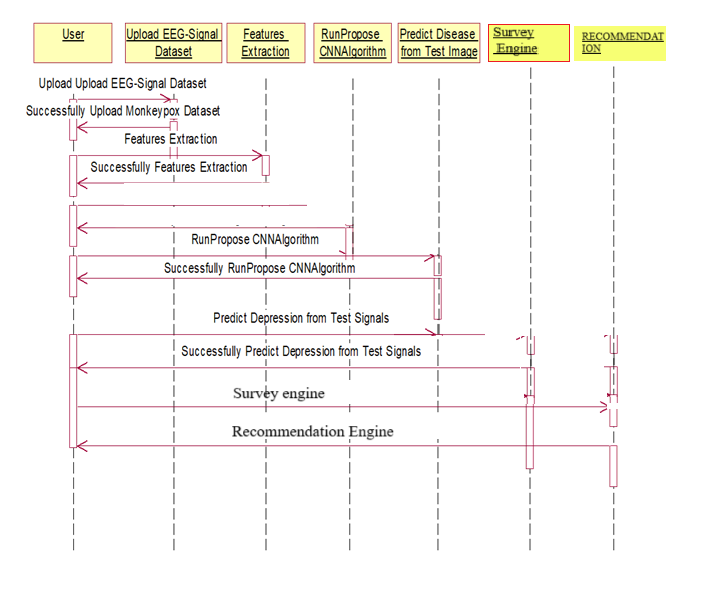
**USECASE DIAGRAM**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as we



**SEQUENCE DIAGRAM**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.

****

**COLLABORATION DIAGRAM**

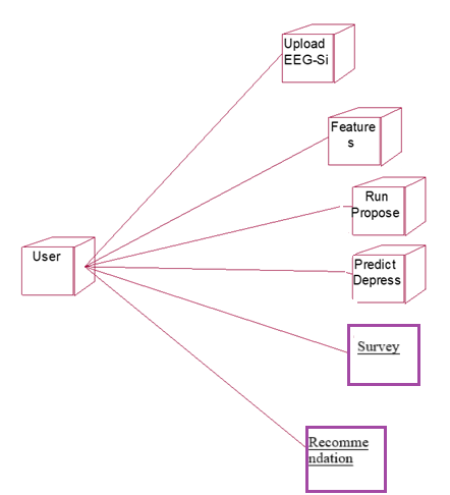
A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.

****

**DEPLOYMENT DIAGRAM**

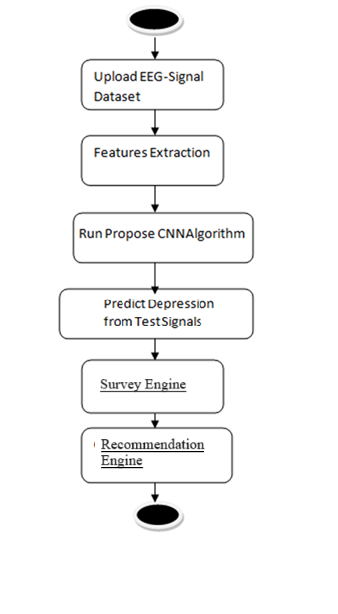
A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



**ACTIVITY DIAGRAM**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent



**CHAPTER V**

**IMPLEMETATION**

**5.1 Python**

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**Features of Python**

* A simple language which is easier to learn
* Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.
* If you are a newbie, it's a great choice to start your journey with Python.
* Free and open-source
* You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code.
* Python has a large community constantly improving it in each iteration.
* **Portability**

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

* **Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

* A high-level, interpreted language
* Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.
* Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.
* Large standard libraries to solve common tasks
* Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .
* Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.
* Object-oriented
* Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.
* With OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python**

1. **Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

**5.2 MODULES USED**

* **NumPy**

NumPy is a fundamental package for numerical computing in Python. It provides support for arrays, matrices, and a variety of mathematical functions to perform operations on these data structures. Its powerful capabilities allow for efficient data manipulation and complex mathematical calculations, making it essential for scientific computing and data analysis.

* **Pandas**

Pandas is a data manipulation and analysis library built on top of NumPy. It introduces data structures like Series and DataFrames, which simplify handling and analyzing structured data. With Pandas, users can easily perform operations such as filtering, aggregating, and merging datasets, making it a go-to tool for data scientists and analysts.

* **Matplotlib**

Matplotlib is a plotting library that provides a flexible way to create static, animated, and interactive visualizations in Python. It allows users to generate a wide variety of plots, including line graphs, bar charts, histograms, and scatter plots. Its customizable features enable the creation of high-quality visual representations of data.

* **Keras**

Keras is a high-level neural networks API written in Python. It allows for easy and fast prototyping of deep learning models by providing a user-friendly interface. Keras can run on top of TensorFlow, making it a popular choice for building and training neural networks due to its simplicity and flexibility.

* **TensorFlow**

TensorFlow is an open-source machine learning framework developed by Google. It provides a comprehensive ecosystem for building and deploying machine learning models, particularly for deep learning applications. TensorFlow offers tools and libraries that support the creation, training, and optimization of complex neural networks.

s

* **h5py**

h5py is a Python interface to the HDF5 binary data format, which is designed to store large amounts of numerical data. It allows users to read and write HDF5 files seamlessly, making it useful for handling datasets that are too large to fit into memory. h5py is commonly used in scientific computing and machine learning for saving model weights and large arrays.

* **Jupyter**

Jupyter is an open-source project that provides a web-based interactive computing environment. It allows users to create and share documents that contain live code, equations, visualizations, and narrative text. Jupyter notebooks are widely used in data science, education, and research for exploratory data analysis and presenting results.

* **Jupyter Client**

Jupyter Client is a library that provides the interface for the Jupyter kernel. It facilitates communication between the Jupyter Notebook or JupyterLab front-end and the backend kernel, enabling code execution and retrieval of results. This component is essential for the interactive computing experience that Jupyter offers.

* **Jupyter Console**

Jupyter Console is a terminal-based console for Jupyter kernels. It provides a lightweight alternative to the Jupyter Notebook interface, allowing users to execute code interactively in a terminal. This is particularly useful for quick experiments or for environments where a graphical interface is not available.

* **Jupyter Core**

Jupyter Core contains the foundational components for the Jupyter ecosystem, including configuration and management tools. It provides essential functionalities that support the operation of Jupyter notebooks, kernels, and other Jupyter applications, ensuring a cohesive user experience.

* **JupyterLab Widgets**

JupyterLab Widgets is an extension of JupyterLab that enables the creation of interactive widgets within Jupyter notebooks. It allows developers to build dynamic user interfaces for their notebooks, enhancing interactivity and user engagement through controls like sliders, buttons, and dropdown menus.

* **Protobuf**

Protocol Buffers (protobuf) is a language-agnostic data serialization format developed by Google. It is used to encode structured data in a compact binary format, making it efficient for data storage and transmission. Protobuf is often used in conjunction with machine learning models for serializing complex data structures.

* **Scikit-learn**

Scikit-learn is a popular machine learning library that provides simple and efficient tools for data mining and analysis. It includes various algorithms for classification, regression, clustering, and dimensionality reduction, along with utilities for model selection and evaluation. Its user-friendly interface makes it accessible for both beginners and experienced practitioners.

* **Seaborn**

Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn makes it easy to create complex visualizations, such as heatmaps and violin plots, while improving the aesthetics of plots with minimal code.

* **IPython**

IPython is an enhanced interactive shell for Python that offers features such as dynamic introspection, rich media, and shell syntax. It serves as the kernel for Jupyter, allowing for interactive computing with enhanced functionalities like magic commands and improved debugging capabilities.

* **IPython Genutils**

IPython Genutils is a utility library that provides a set of general-purpose functions for IPython. It includes tools for handling common tasks such as data serialization, file management, and system information retrieval, serving as a foundational component for building IPython and Jupyter applications.

* **XGBoost**

XGBoost is an optimized gradient boosting library designed for speed and performance. It is widely used for supervised learning tasks, particularly in machine learning competitions due to its effectiveness in handling large datasets and its ability to improve model accuracy. XGBoost supports various optimization techniques, making it a favorite among data scientists.

• MNE

MNE (Magnetoencephalography and Electroencephalography) is a Python library specifically designed for processing, analyzing, and visualizing EEG and MEG data. It supports preprocessing techniques such as filtering, artifact removal, and event-related potential (ERP) analysis, making it ideal for neuroimaging applications.

• SciPy

SciPy is a scientific computing library that builds on NumPy and provides additional functionality for optimization, signal processing, integration, and statistics. It’s often used in EEG signal filtering and feature extraction tasks.

• OpenCV

OpenCV is a computer vision library often used in EEG-based research involving facial expression analysis or emotion recognition. In hybrid models, it can complement EEG analysis with visual cues.

• Joblib

Joblib is a Python library for lightweight pipelining and caching in machine learning workflows. It’s especially useful for saving and loading models or processing large EEG datasets efficiently.

• PyCaret

PyCaret is an open-source, low-code machine learning library that automates the end-to-end process of model selection, training, and evaluation. While not specific to EEG, it can be used to rapidly test and compare models.

• Pytorch

PyTorch is another widely-used deep learning framework that provides flexibility and ease of use for building neural networks. It is an alternative to TensorFlow/Keras and is popular in EEG research for its dynamic computation graph.

• Plotly

Plotly is a graphing library that enables interactive visualizations. It is particularly helpful for dynamically visualizing EEG signals or prediction outputs in web-based dashboards.

• Statsmodels

Statsmodels is a Python package for statistical modeling. It provides classes and functions for the estimation of statistical models and tests, which are useful for analyzing EEG-derived features.

**CHAPTER VI**

**RESULT AND ANALYSIS**

**6.1 TESTING**

**Implementation and Testing**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

**Implementation**

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## **Testing**

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### **System Testing**

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

* **Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

* **Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

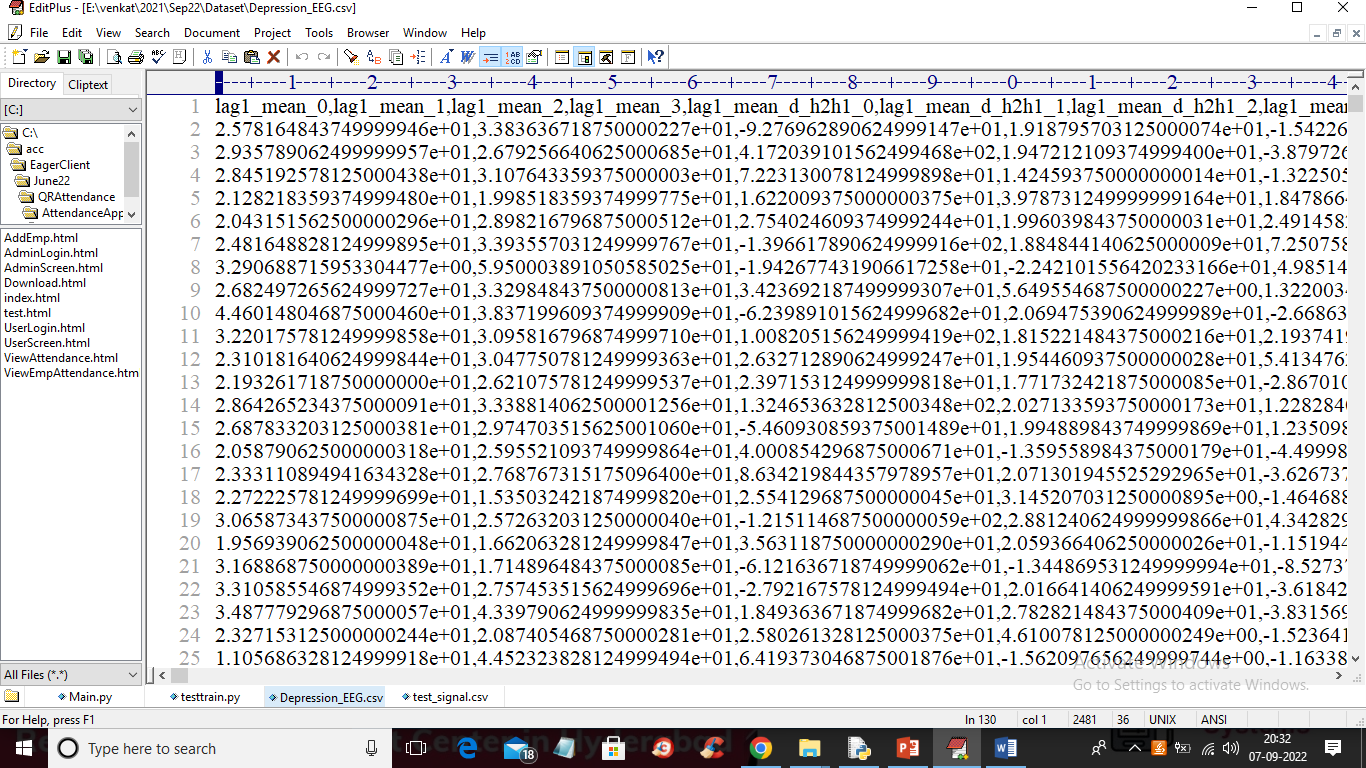
* **Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | | **Actual** |
| 01 | Upload EEG-Signal Dataset | Verify Upload EEG-Signal Dataset or not | If Upload EEG-Signal Dataset may not upload | we cannot do any further operations | we can do further operations | | High | High |
| 02 | Features Extraction | Verify Features Extraction or not | If Features Extraction may not done | we cannot do any further operations | we can do further operations | | High | High |
| 03 | Run Propose CNNAlgorithm | Verify Run Propose CNNAlgorithm or not | If Propose CNN Algorithm may not Run | We cannot run  operation | We can Run the Operation | | High | High |
| 04 | Predict Depression from Test Signals | Verify Predict Depression from Test Signals or not | If Predict Depression from Test Signals may not be | we cannot do any further operations | we can do further operations | | High | High |
| 05 | Survey Engine | Verify Survey Form Input and Processing | If Survey Module does not function properly | responses cannot be processed | responses can be recorded and analyzed | | High | High |
| 06 | Recommendation System | Verify Depression-based Recommendations | If Recommendation logic fails | No guidance shown to users | Personalized suggestions are shown | | High | High |

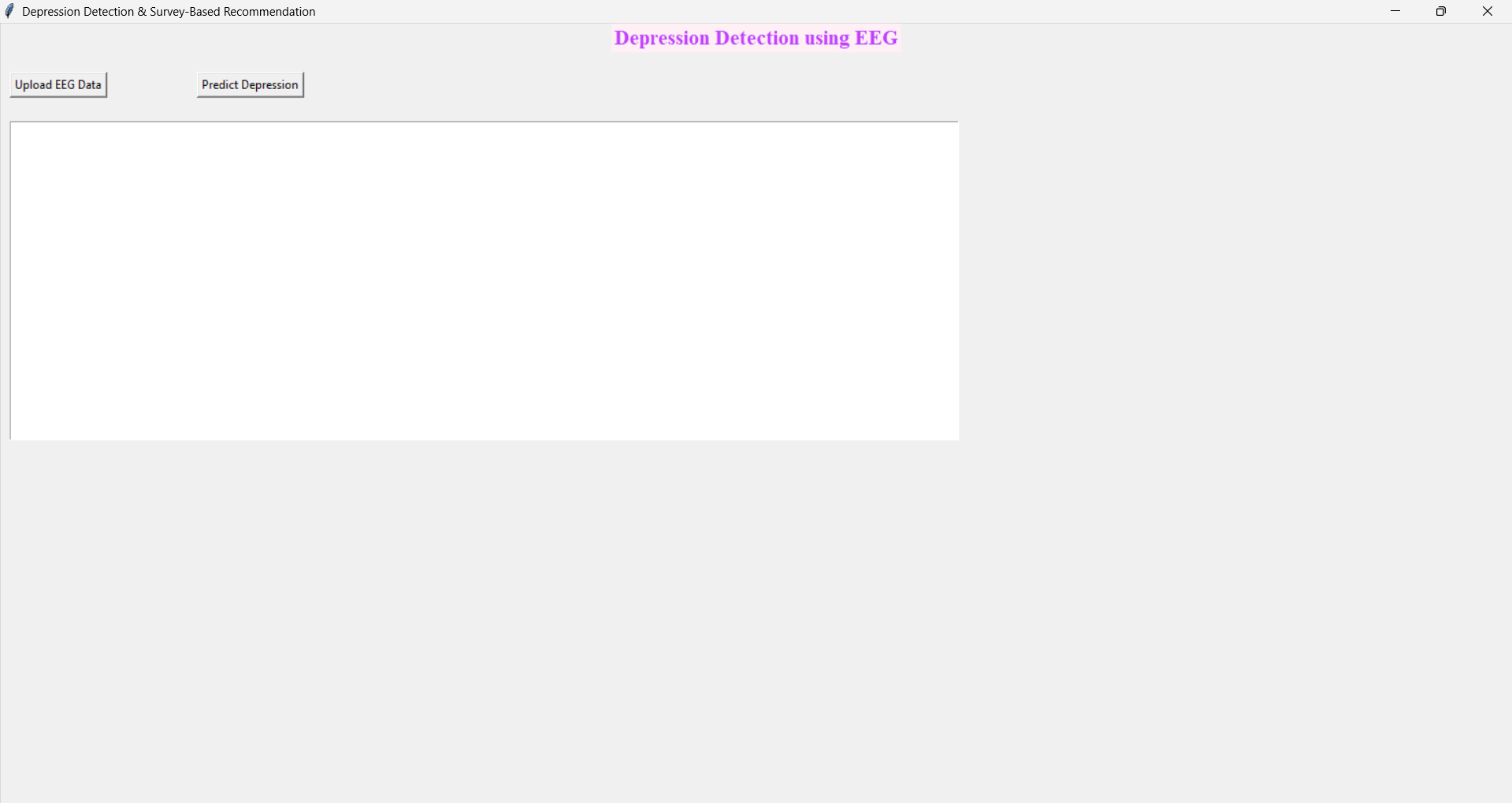
**6.2 SCREENSHOTS**

To train CNN algorithm we have downloaded Depression EEG signals dataset from KAGGLE and below screen showing some dataset details

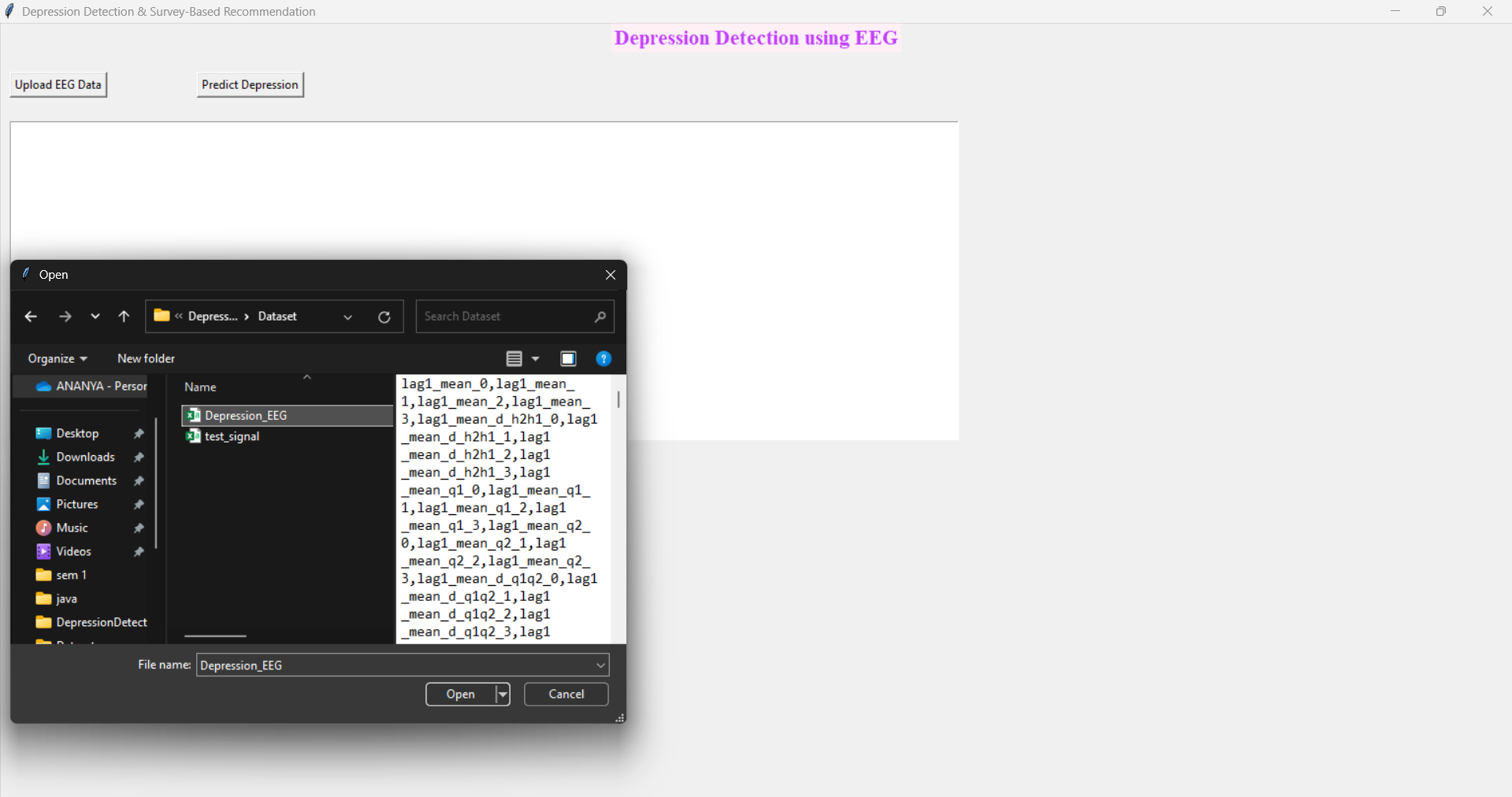


In above dataset screen first row represents dataset column names and remaining rows represents dataset values and this values are extracted from EEG signals and each rows contains 989 columns.

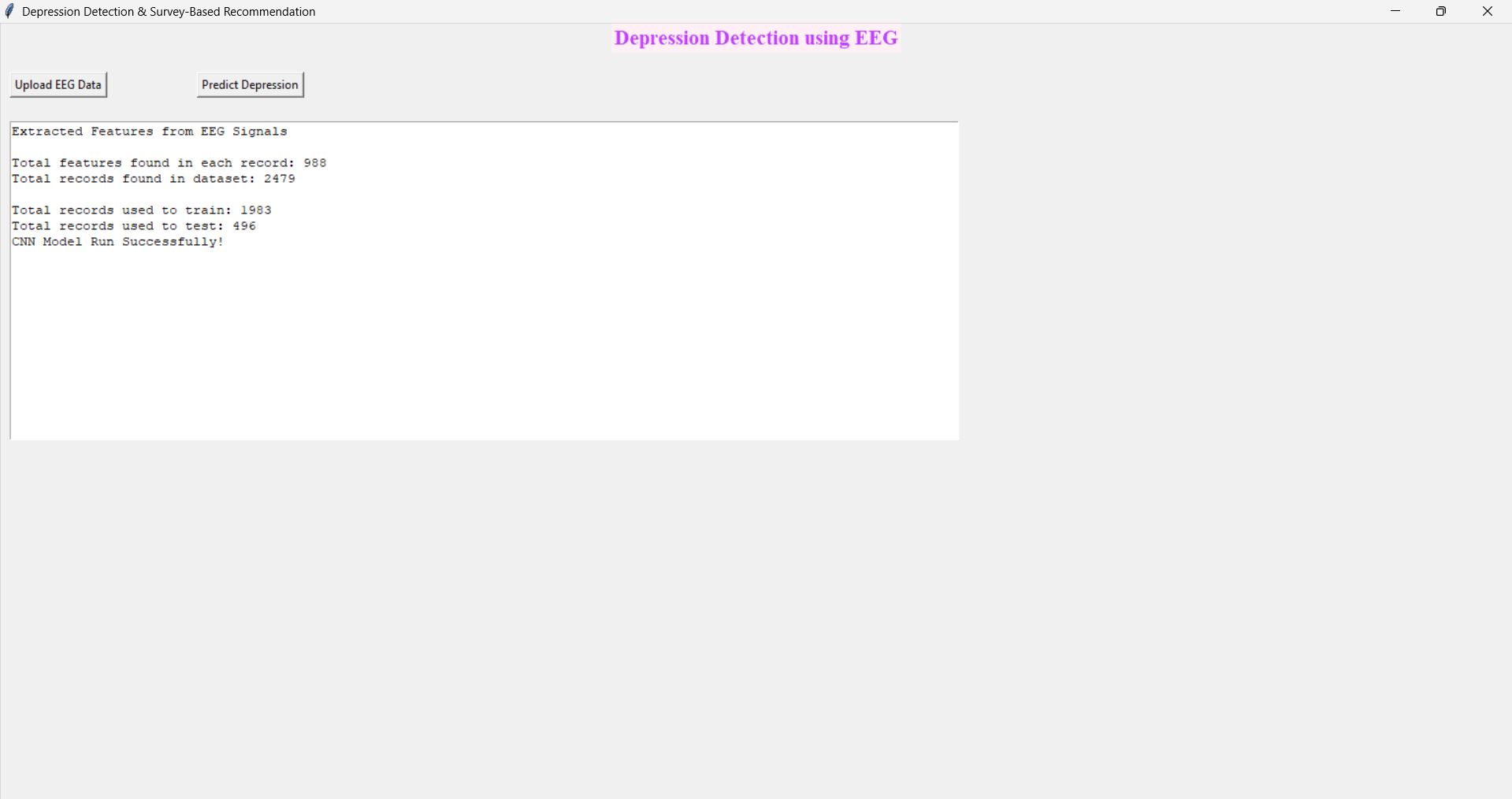
To run project double click on ‘run.bat’ file to get below screen



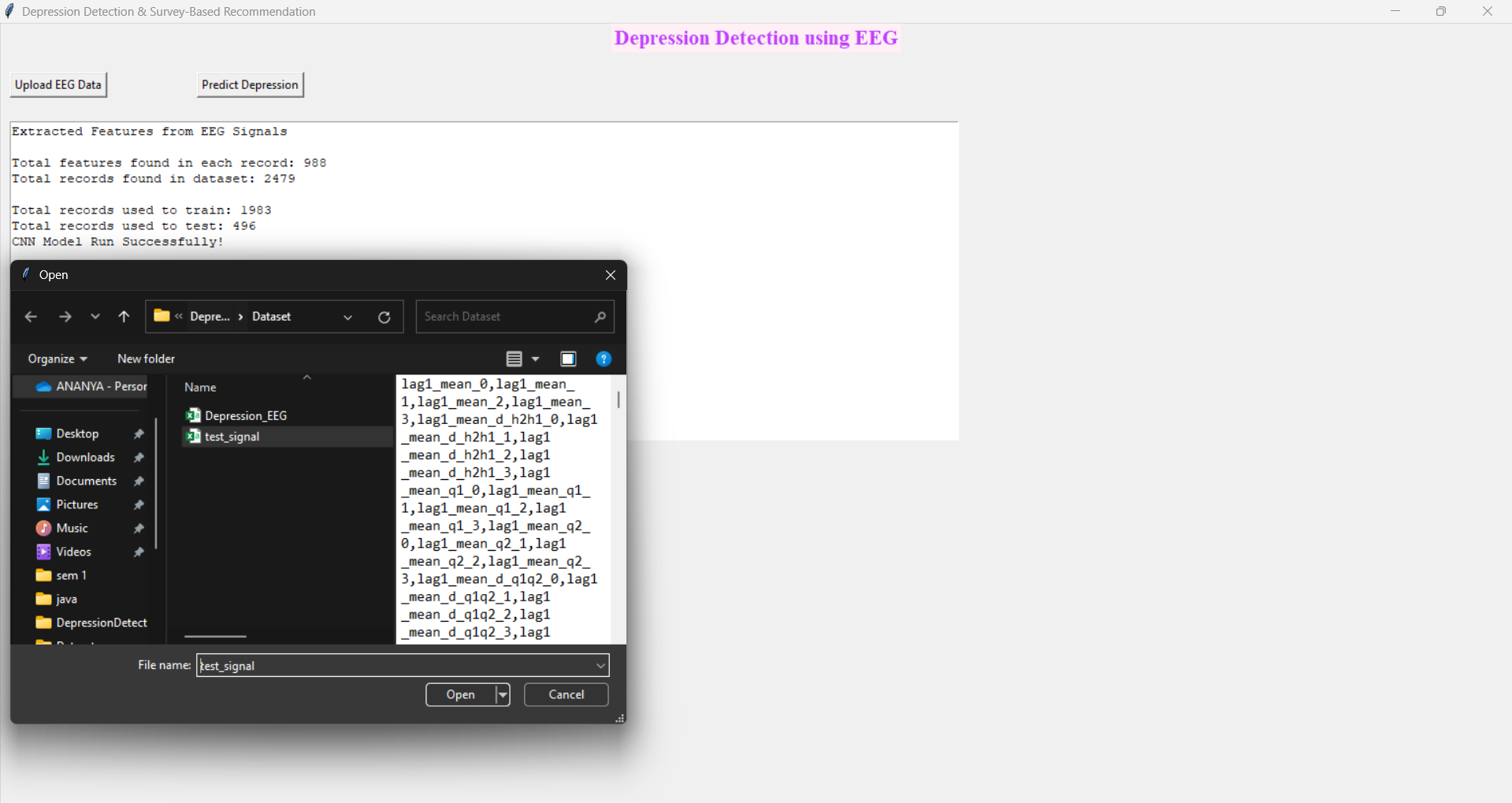
In above screen click on ‘Upload EEG-Signal Dataset’ button to upload dataset and get below output



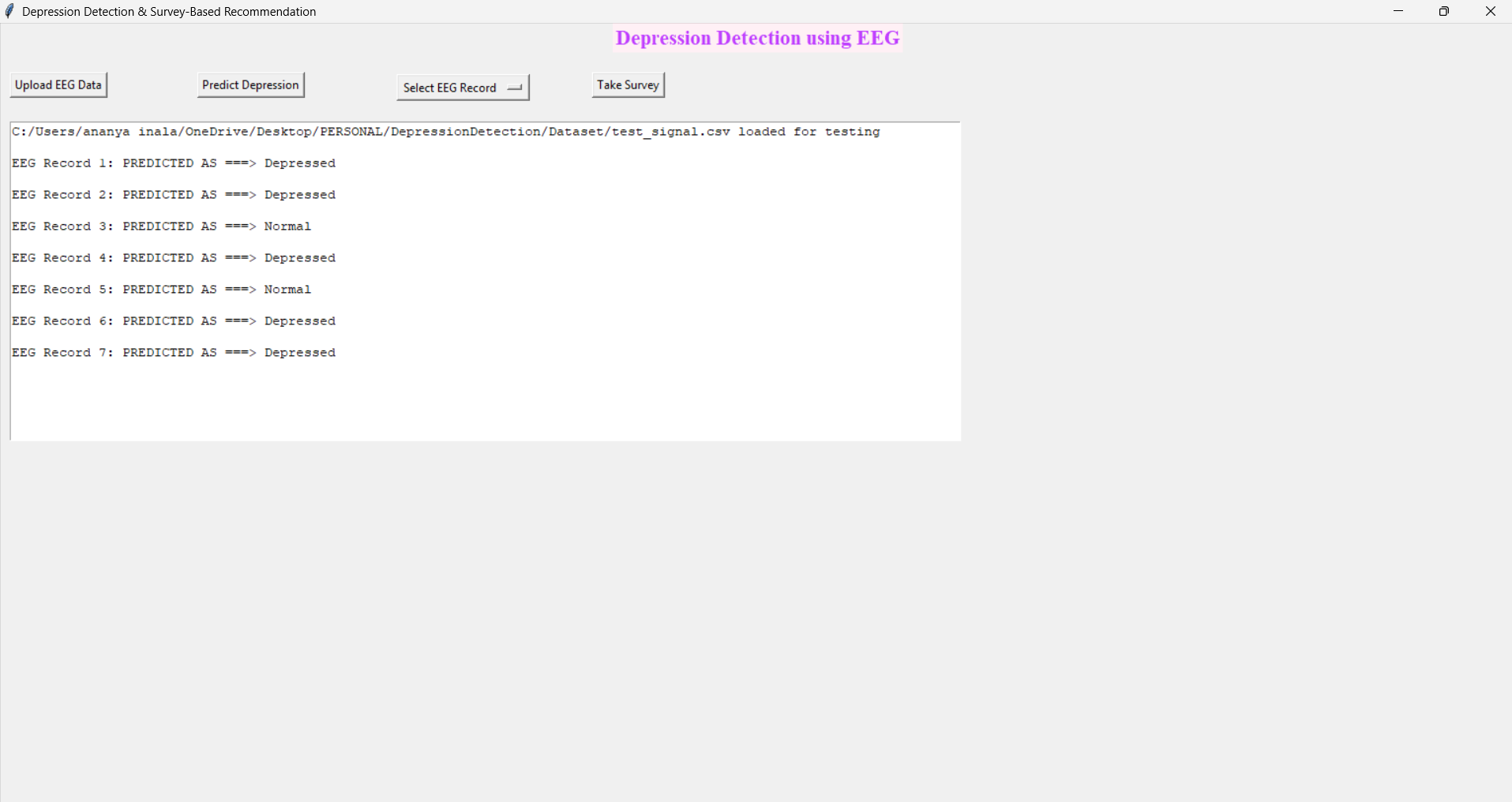
In above screen selecting and uploading EEG-Signal dataset and then click on ‘Open’ button to load dataset and get below output



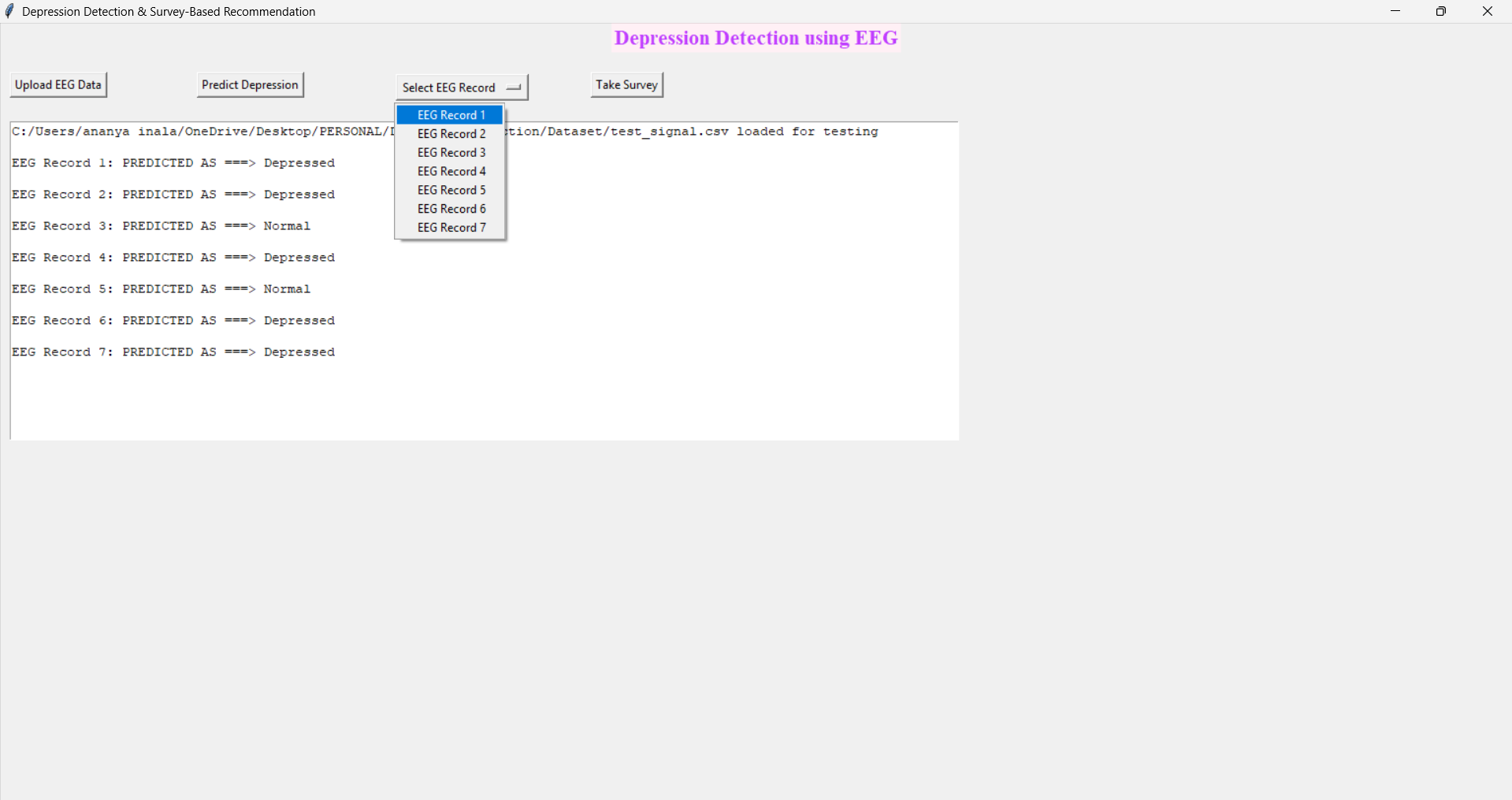
In above screen with CNN we got 93% accuracy and then click on ‘Predict Depression button to upload test data and get prediction output



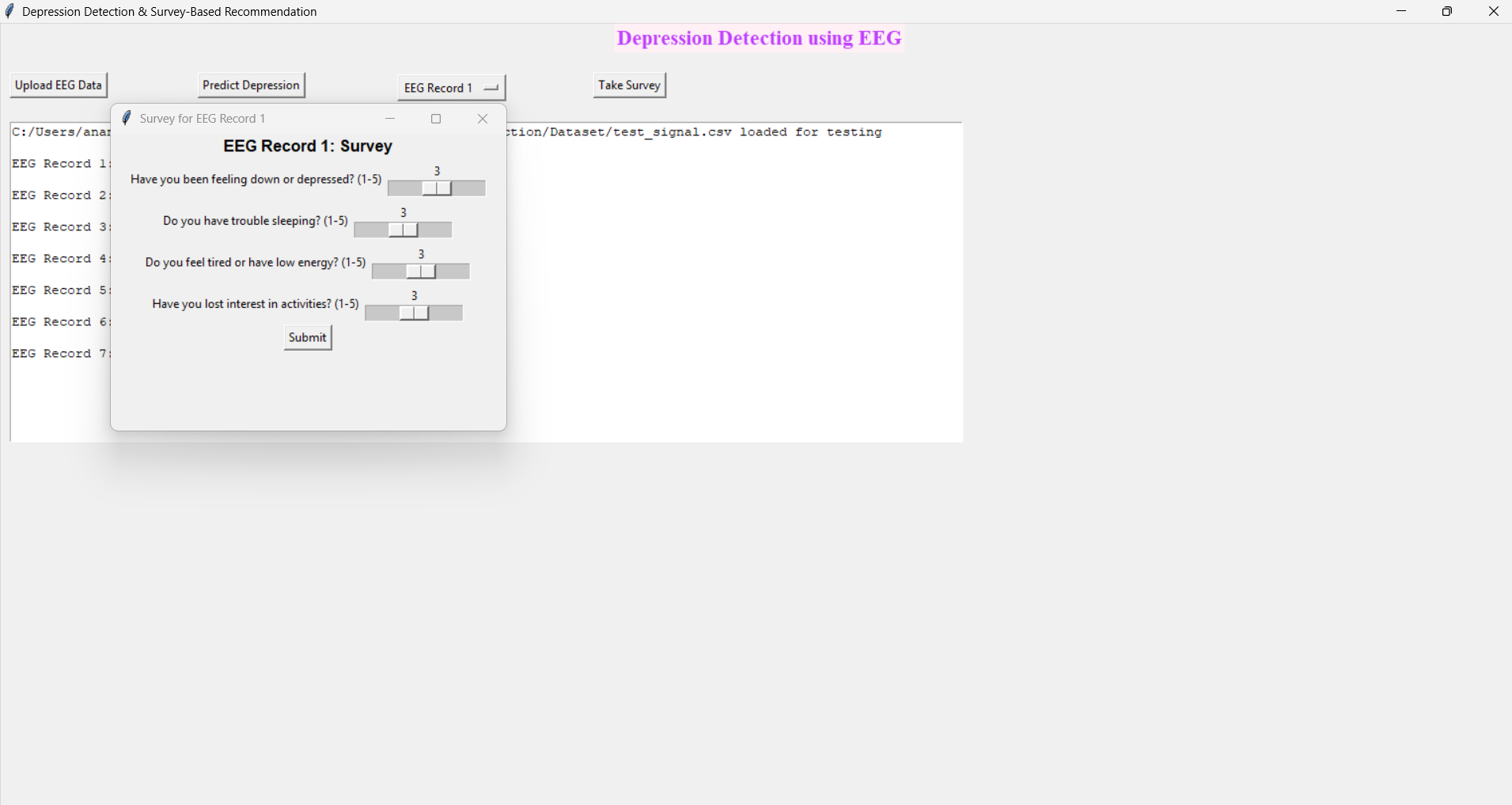
In above screen selecting and uploading ‘test\_signal.csv’ file and then press Open button to load test data and get below prediction output



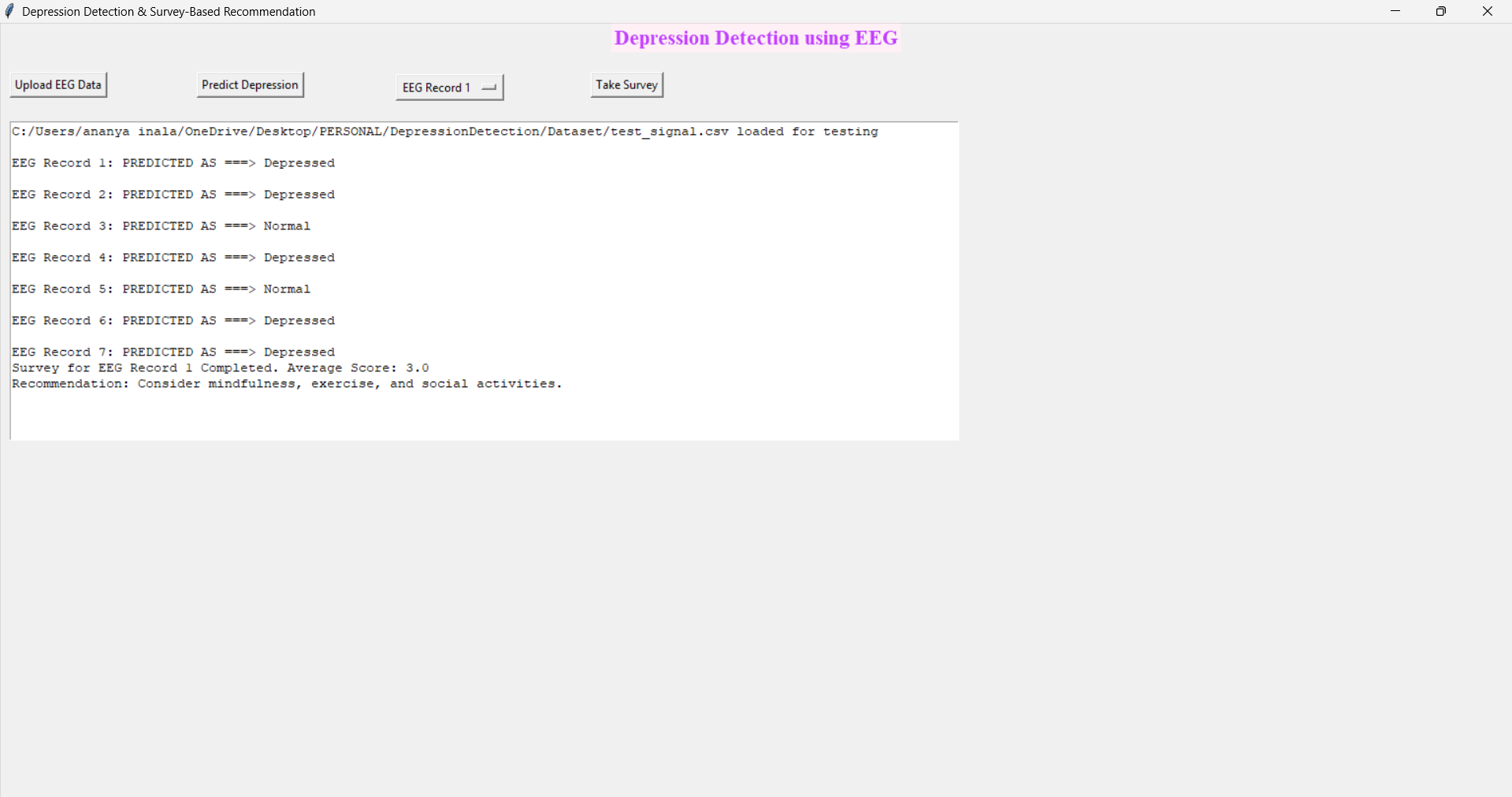
In above screen in square bracket we can see TEST data v after =🡺 symbol we can see prediction as Depressed or Normal and you can scroll above output screen to view all prediction output like below screen



Now click on ‘SELECT EEG RECORD ‘ to select eeg record to take survey



Click on take survey to conduct survey

****

We can see the survey results and recommendation

**CHAPTER VII**

**CONCLUSION**

In this study, we present preliminary results for an automated EEG classification system developed to classify EEGs with and without IEDs. It comprises three main modules: pre-processing, waveform-level classification (CNN), and EEG-level classification (SVM). In this project, we are using a deep learning CNN algorithm to predict depression from an EEG signals dataset. Nowadays, humans are more prone to depression due to the competitive environment in all fields, and timely detection of depression can help individuals recover faster.

To predict depression accurately, we apply a CNN algorithm that filters trained data at multiple layers to extract optimized features, resulting in improved prediction accuracy.

In addition to EEG-based detection, we have also integrated a **survey module** that gathers self-reported mental health inputs through clinically validated questionnaires. This allows for a hybrid approach—combining objective EEG data with subjective user feedback, improving diagnostic reliability and offering a more holistic view of the individual's mental state.

Furthermore, the system includes a **recommendation engine** that suggests personalized coping strategies based on the user’s EEG results and survey responses. Recommendations may include guided meditation, therapy options, lifestyle tips, or professional consultation—each tailored to the severity and pattern of the detected symptoms.

In our future work, we intend to incorporate artifact rejection in order to reduce false detections. We also aim to modify the pre-processing module to adapt to different montages and EEG recording equipment, ensuring broader applicability and robustness of the system across diverse datasets and hardware platforms.

**CHAPTER VIII**

**REFERENCES**

1. Krizhevsky, A., Sutskever, I., and Hinton, G. E., ImageNet Classification with Deep Convolutional Neural Networks. Adv. Neural Inf. Proces. Syst., 2012. <https://doi.org/10.1016/j.protcy.2014.09.007>.
2. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., Going deeper with convolutions. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015.
3. Simonyan, K., Zisserman, A., Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
4. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., Gradient-based learning applied to document recognition. Proc. IEEE 86:2278–2323, 1998. <https://doi.org/10.1109/5.726791>.
5. Bashivan, P., Rish, I., Yeasin, M., and Codella, N., Learning representations from EEG with deep recurrent-convolutional neural networks. arXiv preprint arXiv:1511.06448, 2015.
6. Talo, M., Baloglu, U. B., Yildirim, O., and Acharya, U. R., Application of deep transfer learning for automated brain abnormality classification using MR images. Cogn. Syst. Res., 2018. <https://doi.org/10.1016/j.cogsys.2018.12.007>.
7. Bhat, S., Acharya, U. R., Hagiwara, Y., Dadmehr, N., and Adeli, H., Parkinson’s disease: Cause factors, measurable indicators, and early diagnosis. Comput. Biol. Med., 2018. <https://doi.org/10.1016/j.compbiomed.2018.09.008>.
8. Acharya, U. R., Vinitha Sree, S., Swapna, G., Martis, R. J., and Suri, J. S., Automated EEG analysis of epilepsy: A review. Knowledge-Based Syst 45:147–165, 2013. <https://doi.org/10.1016/j.knosys.2013.02.014>.

1. Janga Vijaykumar, P., and Sreenivasareddy, E., A review on Machine Learning Techniques for Neurological disorders estimation by Analyzing EEG Waves. Int J Trend Sci Res Dev., 2018.
2. Acharya, U. R., Hagiwara, Y., Deshpande, S. N., Suren, S., Koh, J. E. W., Oh, S. L., Arunkumar, N., Ciaccio, E. J., and Lim, C. M., Characterization of focal EEG signals: A review. Futur. Gener. Comput. Syst., 2019. <https://doi.org/10.1016/j.future.2018.08.044>.

1. Petrosian, A., Prokhorov, D., Homan, R., Dasheiff, R., and Wunsch, D., Recurrent neural network based prediction of epileptic seizures in intra- and extracranial EEG. Neurocomputing., 2000. <https://doi.org/10.1016/S0925-2312(99)00126-5>.
2. Mirowski, P., Madhavan, D., LeCun, Y., and Kuzniecky, R., Classification of patterns of EEG synchronization for seizure prediction. Clin. Neurophysiol., 2009. <https://doi.org/10.1016/j.clinph.2009.09.002>.
3. Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., and Adeli, H., Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals. Comput. Biol. Med. 100:270–278, 2018.
4. Wulsin, D. F., Gupta, J. R., Mani, R., Blanco, J. A., and Litt, B., Modeling electroencephalography waveforms with semi-supervised deep belief nets: Fast classification and anomaly measurement. J. Neural Eng., 2011. <https://doi.org/10.1088/1741-2560/8/3/036015>.
5. Stober, S., Sternin, A., Owen, A. M., and Grahn, J. A., Deep feature learning for EEG recordings. arXiv preprint arXiv:1511.04306, 2015.
6. Yıldırım, Ö., Baloglu, U. B., and Acharya, U. R., A deep convolutional neural network model for automated identification of abnormal EEG signals. Neural Comput. & Applic.:1–12, 2018.