**CHAPTER -I**

**1.INTRODUCTION**

Brain stroke is a critical medical condition that occurs when the blood supply to a part of the brain is interrupted or reduced, preventing brain tissue from getting oxygen and nutrients. This interruption can lead to brain cell death within minutes, making stroke a leading cause of death and long-term disability worldwide. According to the World Health Organization (WHO), millions of people suffer strokes each year, and early detection plays a crucial role in reducing mortality and improving recovery outcomes.

Strokes are generally classified into two main types: ischemic stroke, caused by a blockage in an artery supplying blood to the brain, and hemorrhagic stroke, resulting from the rupture of a blood vessel. Risk factors such as hypertension, diabetes, smoking, obesity, high cholesterol, and a sedentary lifestyle significantly increase the likelihood of stroke.

In recent years, advancements in data science and machine learning have opened new avenues for the early prediction and diagnosis of strokes. Predictive models that analyze clinical and demographic data can assist healthcare providers in identifying individuals at high risk, enabling timely intervention and treatment.

This study aims to explore the effectiveness of machine learning algorithms in predicting the likelihood of a brain stroke based on various health and lifestyle parameters. By leveraging data-driven approaches, the research seeks to contribute to more proactive and personalized healthcare strategies, ultimately reducing the burden of stroke on individuals and health systems alike. Introduction to Brain Stroke Prediction using Machine Learning

A brain stroke, often referred to as a cerebrovascular accident (CVA), is a sudden interruption in the blood supply to the brain. This disruption can result from either a blocked artery (ischemic stroke) or a ruptured blood vessel (hemorrhagic stroke), leading to potentially severe and irreversible brain damage. Strokes are one of the leading causes of death and disability worldwide, with millions affected every year. Early diagnosis and timely medical intervention can significantly reduce the fatality rate and improve the quality of life for survivors. However, traditional stroke prediction methods rely heavily on clinical evaluation, which can sometimes delay diagnosis, especially in resource-limited settings. In this context, the integration of machine learning (ML) techniques offers a promising path toward enhancing predictive accuracy, enabling faster and more reliable detection of high-risk individuals.

Machine learning, a subfield of artificial intelligence (AI), involves training algorithms on data to recognize patterns, make decisions, and predict outcomes without being explicitly programmed. Its growing application in healthcare has transformed diagnostics, treatment planning, and patient monitoring. In the case of stroke prediction, machine learning models can be trained on large datasets comprising patient medical histories, lifestyle factors, genetic predispositions, and imaging data. These models can uncover complex relationships between risk factors such as hypertension, diabetes, heart disease, smoking habits, and obesity that might not be easily evident through conventional statistical methods. By learning from this data, ML algorithms can forecast the likelihood of a stroke event occurring, potentially well before clinical symptoms emerge.

The process of building an effective brain stroke prediction system using machine learning typically involves several stages. First, data collection is crucial. Publicly available datasets, such as those from healthcare institutions or government health surveys, often contain information on various health parameters, demographics, and outcomes related to strokes. After gathering the data, the next step is preprocessing, where missing values are handled, irrelevant or redundant features are removed, and the data is normalized or scaled as needed. Feature selection is also a vital part of the process; selecting the most significant attributes can improve model performance and reduce computational complexity. For instance, age, gender, blood pressure, glucose levels, BMI, and prior history of cardiovascular disease are commonly used features in stroke prediction models.

**CHAPTER - II**

**2.SYSTEM ANALYSIS**

**2.1 Existing system:**

Brain stroke is a medical emergency where blood flow to the brain is disrupted. It can lead to permanent neurological damage or death if not diagnosed and treated promptly. With growing data availability and computing capabilities, machine learning (ML) has become a promising tool in healthcare, particularly in stroke risk prediction.

Traditional models typically use structured clinical data (eg, age,blood,pressure,diabetes,cholesterol levels) to predict stroke risk. Some common machine learning models include:

* **Logical Regression (LR) –** Simple and interpretable but may not capture complex patterns.
* **Random Forest ( RF) –** Uses decision trees to handle non-linear relationships effectively.
* **Support Vector Machine (SVM) –** Good for small datasets but computationally expensive for large ones.
* **XGBoost –** A powerful gradient boosting model that improves prediction accuracy.
* **Artificial Neural Networks (ANNs) –** Can capture complex relationhips nut need careful tuning.

### ****Description Of Existing Systems:****

### **Current machine learning-based stroke prediction systems use clinical and demographic data to assess an individual's stroke risk. These systems are typically trained on historical patient datasets and help identify patterns and correlations among key health indicators.**

### ****Common Features Used:****

### **Age**

### **Gender**

### **Hypertension**

### **Heart disease**

### **Body Mass Index (BMI)**

### **Smoking status**

### **Work type**

### **Residence type**

### **Glucose levels**

### **History of previous stroke**

### ****Commonly Used Datasets:****

### **Kaggle Stroke Prediction Dataset**

### **UCI Machine Learning Repository**

### **Hospital Electronic Health Records.**

### ****Limitations Of The Existing System:****

### ****Data Imbalance:** Most datasets have more non-stroke than stroke cases, affecting prediction accuracy.**

### ****Lack of Personalization:** Models are generic and not tailored to individual risk profiles.**

### ****Limited Interpretability:** Complex models like neural networks are often black boxes.**

### ****Real-Time Limitations:** Many systems are offline models and not integrated into real-time diagnosis tools.**

**2.2 Disadvantages Of Existing System:**

* **Limited Dataset Availability:**

One of the primary challenges in existing brain stroke prediction systems is the lack of large, high-quality datasets. Many available datasets are small or region-specific, which restricts the model’s ability to learn diverse patterns. Additionally, stroke datasets often suffer from class imbalance, where the number of stroke cases is significantly lower than non-stroke cases. This imbalance affects the model’s performance and leads to biased predictions.

* **Low Generalizability:**

Models trained on data from a particular population may not perform well when applied to different regions, age groups, or ethnicities. This happens because health risk factors and genetic predispositions vary widely across populations. As a result, existing models may show reduced accuracy and reliability when used outside their training context.

* **Inadequate Feature Selection:**

Many machine learning models fail to consider critical features that influence stroke risk, such as family history, lifestyle habits, and psychological stress. Moreover, using irrelevant or redundant features can confuse the model and reduce predictive accuracy. Proper feature engineering and domain knowledge are often lacking in existing systems.

* **Overfitting Issues:**

Some machine learning models, especially complex ones like deep learning networks, tend to overfit the training data. Overfitting occurs when the model performs well on training data but poorly on new, unseen data. This limits the model's practical use and reliability in real-world scenarios where data variability is high.

* **Lack of Model Interpretability:**

Healthcare professionals need to understand the reasoning behind a prediction to make informed clinical decisions. However, many existing systems use black-box models like Random Forest, XGBoost, or neural networks, which offer little transparency.

* **Limited Real-Time Predictive Capability:**

Most existing models are not designed for real-time stroke prediction, which is essential in critical care situations. High computational requirements or delays in data processing can make the model unsuitable for emergency use. As a result, these systems are often confined to research settings rather than practical hospital environments.

* **Neglect of Temporal and Longitudinal Data:**

Stroke risk is not static; it evolves with time based on health changes, treatments, and habits. However, many systems only use snapshot (single-time) data, ignoring longitudinal patient history. This limits the model’s ability to detect gradual risk buildup and makes predictions less accurate.

* **Privacy and Security Concerns:**

Health data is extremely sensitive and must be handled with strict privacy regulations. Many machine learning systems require cloud-based processing, raising concerns over data security and regulatory compliance. This discourages healthcare institutions from sharing data for training and testing.

* **Lack of Integration with Clinical Systems:**

Current machine learning models are often developed independently of hospital infrastructure. This makes it difficult to integrate them into Electronic Health Record (EHR) systems or clinical workflows. Without seamless integration, even accurate models remain underutilized in real-world settings.

**2.3 Proposed System:**

### ****Proposed CNN-Based Model for Brain Stroke Prediction (Using Imaging Data):****

**The proposed system for brain stroke prediction leverages machine learning techniques to analyze patient data and identify individuals at high risk of experiencing a stroke. The system collects key medical parameters such as age, gender, hypertension, heart disease, average glucose level, body mass index (BMI), and smoking status. This data is preprocessed to handle missing values, normalize input features, and encode categorical variables. Various machine learning algorithms, such as Logistic Regression, Random Forest, Support Vector Machine (SVM), and Gradient Boosting, are trained on historical stroke datasets to recognize complex patterns associated with stroke occurrence. The system uses model evaluation metrics like accuracy, precision, recall, and F1-score to determine the most effective model. Once trained, the selected model can be integrated into a clinical decision support system to assist healthcare providers in identifying at-risk patients early, enabling timely interventions and reducing the likelihood of severe outcomes.**

A **Convolutional Neural Network (CNN)** is a deep learning model commonly used for **image-based** stroke detection (e.g., from MRI or CT scans). The process typically involves:

* **Preprocessing** – Normalizing images, segmenting brain regions, and removing noise.
* **Feature Extraction** – CNN automatically detects stroke-related features from medical images.
* **Classification** – The model predicts stroke presence/severity based on extracted features.
* **Evaluation** – Performance is assessed using accuracy, precision, recall, F1-score, and AUC-ROC.
* **Traditional ML models** work well for early stroke prediction based on patient risk factors.
* **CNN models** are more effective for **image-based stroke diagnosis**, detecting patterns that may not be obvious to human experts.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | | **Aspect** |  | **Traditional ML Models** | **CNN (Proposed)** | | --- | --- | --- | --- | |  |  |  |  | | **Input Data** |  | Structured clinical data (numerical/text) | Medical images (MRI, CT) | |  |  |  |  | | **Feature Selection** |  | Manual selection required | Automatic feature extraction | |  |  |  |  | | **Interpretability** |  | High (e.g., feature importance in RF) | Lower (black-box nature) | |  |  |  |  | | **Performance** |  | Good for tabular data | Excellent for imaging-based stroke detection | |  |  |  |  | | **Computational Cost** |  | Lower | High (requires GPUs for training) | |  |  |  |  | | **Real-World Application** |  | Early stroke risk assessment | Stroke diagnosis from brain scans | | |

**2.4 Literature Survey:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sno** | **Author /**  **Year of publication / Name of the Journal** | **Description** | **Benefits** | **Drawbacks** |
|  | IEEE-2023  Somya Srivastav; Kalpna Guleria; Shagun Sharma | Machine Learning Models for Early Brain Stroke Prediction: A Performance Analogy | To avoid the challenges of brain stroke, it is required to predict it in the early stages. | There is a rapid increase in the number of cases each day |
|  | Nagaraju Devarakonda;  Bethu Lokendra Sri Sai  IEEE-2023 | Prediction of brain stroke using machine learning algorithms which helps to rehabilitate the patient so that one can gain their life back to normal. | The system effectively provides heart risk scores with the highest accuracy in a runtime environment. | When brain cells are deprived of oxygen for an extended period of time, they die |
|  | IEEE  2023  T. Navya Deepthi;  Sk. Sharmila | Prediction of Brain Stroke in Human Beings using Machine Learning | This study has collected a variety of patients' datasets. It includes a number of medical factors. | The second is haemorrhagic stroke occurs when a brain artery bursts or releases blood. |
|  | IEEE-2024  Ankita Sharma;  Sonam Mittal | Prospecting Brain Stroke Onset: A Comparative Analysis Of Supervised Learning Models | The main cause of stroke could be hypertension. Brain stroke happens when blood-flowing arteries get blocked | It causes sudden death due to blockage in the blood vessels The mortality rate is gradually increasing. |
|  | Madhavi K. Reddy;  Karthik Kovuri;  J Avanija;  M Sakthivel;  IEEE  2022 | Brain Stroke Prediction Using Deep Learning: A CNN Approach | Using a deep learning model on a brain disease dataset, this method of predicting analytical techniques for stroke was carried out. | Bayesian classification when it comes to predictive data mining. |

**CHAPTER - III**

**3. SYSTEM SPECIFICATION**

**3.1 Hardware Requirement :**

* Processor : intel
* RAM : 8GB minimum, 16GB recommended
* Storage : 100 GB

**1.Processor ( INTEL) :**

The processor, often considered the brain of the computer, plays a central role in managing all tasks related to machine learning model development and execution. For this project, an Intel processor is specified. Intel processors are widely recognized for their reliability, performance, and compatibility with major machine learning libraries such as TensorFlow, Scikit-learn, and PyTorch. Most Intel Core i5 or i7 series CPUs, for example, come with multiple cores and threads that enable parallel processing, which significantly boosts the performance of model training and evaluation.

**2. RAM (8GB Minimum, 16GB Recommended) :**

The recommendation of 16GB RAM is based on the need for efficient in-memory computation. Machine learning operations, especially those using libraries like Pandas, NumPy, and TensorFlow, perform much faster when the entire dataset and model parameters can be stored in memory during execution. This is particularly important for training deep learning models or implementing ensemble methods, which require substantial memory resources for simultaneous training and prediction.

**3. Storage (Minimum 100GB) :**

Medical datasets are often large, especially when they include imaging data (e.g., MRI or CT scans) or long-term patient records. These files can quickly consume several gigabytes of space. Additionally, during the iterative development process, multiple versions of datasets and models may be stored for comparison and documentation purposes. If historical performance needs to be analyzed or if regulatory bodies require audit trails, having adequate storage is not only practical.

**3.2 Software Requirement :**

* Operating system : windows 10
* Front End : HTML,CSS, JavaScript
* Back End : python,Ms Excel
* Document : Microsoft word

The Brainstroke Prediction using Machine Learning project relies on a carefully selected set of software tools to support the development, deployment, and documentation of the system. The operating system used for this project is Windows 10, which offers a stable and user-friendly environment compatible with a wide range of programming tools, libraries, and applications. For the front-end development, the project utilizes standard web technologies—HTML, CSS, and JavaScript. These technologies enable the creation of an intuitive and responsive user interface that allows users such as doctors, healthcare staff, or patients to interact with the system easily. The front end is designed to collect user inputs, display predictions, and present health-related insights in a visually accessible manner.

On the back-end, the project uses Python, one of the most popular programming languages in machine learning and data science due to its simplicity, versatility, and extensive library support. Python handles all the core machine learning functionalities such as data preprocessing, model training, and prediction using libraries like Pandas, Scikit-learn, or TensorFlow. Alongside Python, MS Excel is used for managing, organizing, and performing preliminary analysis of the medical datasets. Excel provides a familiar spreadsheet interface for viewing raw data, calculating basic statistics, and exporting structured datasets for further processing in Python.

**CHAPTER - IV**

**4.SOFTWARE DESCRIPTION**

**4.1 System Environment :**

**Python :**

Python has a simple syntax similar to the English language. Python has syntax that allows developers to write programs with fewer lines than some other programming languages. Python runs on an interpreter system, meaning that code can be executed as soon as it is written.

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

**It Is Used For:**

* web development (server-side),
* software development,
* mathematics,
* system scripting.

**What can Python do?**

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

**Why Python?**

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-oriented way or a functional way.

**Good to Know :**

* The most recent major version of Python is python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated D The most recent major version of Python is Python 3, which we shall be using in this evelopment Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.
* Python Syntax compared to other programming languages
* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

**Anaconda:**

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for things other than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages.

**Overview :**

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command-line interface (CLI).

The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

Before version 20.3, when pip installed a package, it automatically installed any dependent Python packages without checking if these conflict with previously installed packages. It would install a package and any of its dependencies regardless of the state of the existing installation Because of this, a user with a working installation of, for example, TensorFlow, could find that it stopped working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by TensorFlow. In some cases, the package would appear to work but produce different results in detail. While pip has since implemented consistent dependency resolution, this difference accounts for a historical differentiation of the conda package manager.

In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g. the user may wish to have TensorFlow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

Open source packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the conda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on PyPI may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed.

Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with conda.

**Anaconda Navigator :**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

* + JupyterLab
  + Jupyter Notebook
  + QtConsole
  + Spyder
  + Glue
  + Orange
  + RStudio
  + Visual Studio Code
  + Conda

**Main article: Conda (package manager) :**

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

**Anaconda Cloud :**

Anaconda Cloud is a package management service by Anaconda where users can find, access, store and share public and private notebooks, environments, and conda and PyPI packages.Cloud hosts useful Python packages, notebooks and environments for a wide variety of applications. Users do not need to log in or to have a Cloud account, to search for public packages, download and install them.

Users can build new packages using the Anaconda Client command line interface (CLI), then manually or automatically upload the packages to Cloud.

**Python Technology:**

**Python** is an interpreter, high-level, general-purpose programming language. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. **Python** is often described as a "batteries included" language due to its comprehensive standard library.

## Python Programming Language:

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by Meta programming and met objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python packages with a wide range of functionality, including:

* Easy to Learn and Use
* Expressive Language
* Interpreted Language
* Cross-platform Language
* Free and Open Source
* Object-Oriented Language
* Extensible

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one and preferably only one obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Python is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar.

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying

that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them

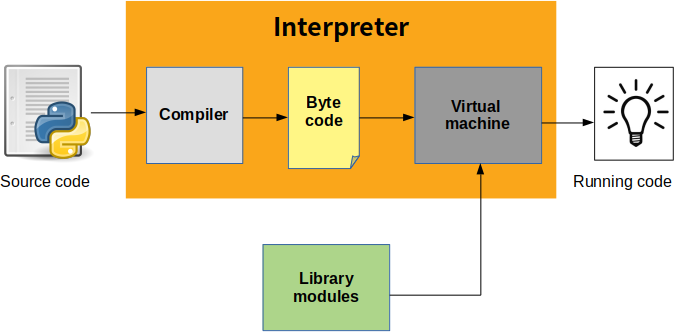
## Python Platform :

The platform module in Python is used to access the underlying platform's data, such as, hardware, operating system, and interpreter version information. The platform module includes tools to see the platform's hardware, operating system, and interpreter version information where the program is running.

There are four functions for getting information about the current Python interpreter. python\_version() and python\_version\_tuple() return different forms of the interpreter version with major, minor, and patch level components. python\_compiler() reports on the compiler used to build the interpreter. And python\_build() gives a version string for the build of the interpreter.

Platform() returns string containing a general purpose platform identifier. The function accepts two optional Boolean arguments. If aliased is true, the names in the return value are

converted from a formal name to their more common form. When terse is true, returns a minimal value with some parts dropped.



## What does python technology do?

Python is quite popular among programmers, but the practice shows that business owners are also Python development believers and for good reason. Software developers love it for its straightforward syntax and reputation as one of the easiest programming languages to learn. Business owners or CTOs appreciate the fact that there’s a framework for pretty much anything from web apps to machine learning.

Moreover, it is not just a language but more a technology platform that has come together through a gigantic collaboration from thousands of individual professional developers forming a huge and peculiar community of aficionados.

So what are the tangible benefits the language brings to those who decided to use it as a core technology? Below you will find just some of those reasons.

## Productivity And Speed :

It is a widespread theory within development circles that developing Python applications is approximately up to 10 times faster than developing the same application in Java or C/C++. The impressive benefit in terms of time saving can be explained by the clean object-oriented design, enhanced process control capabilities, and strong integration and text processing capacities. Moreover, its own unit testing framework contributes substantially to its speed and productivity.

## Python Is Popular For Web Apps :

Web development shows no signs of slowing down, so technologies for rapid and productive web development still prevail within the market. Along with JavaScript and Ruby, Python, with its most popular web framework Django, has great support for building web apps and is rather popular within the web development community.

## Open-Source And Friendly Community :

As stated on the official website, it is developed under an OSI-approved open source license, making it freely usable and distributable. Additionally, the development is driven by the community, actively participating and organizing conference, meet-ups, hackathons, etc. fostering friendliness and knowledge-sharing.

## Python Is Quick To Learn :

It is said that the language is relatively simple so you can get pretty quick results without actually wasting too much time on constant improvements and digging into the complex engineering insights of the technology. Even though Python programmers are really in high demand these days, its friendliness and attractiveness only help to increase number of those eager to master this programming language.

## Broad Application :

It is used for the broadest spectrum of activities and applications for nearly all possible industries. It ranges from simple automation tasks to gaming, web development, and even complex enterprise systems. These are the areas where this technology is still the king with no or little competence:

* + Machine learning as it has a plethora of libraries implementing machine learning algorithms.
  + Web development as it provides back end for a website or an app.
  + Cloud computing as Python is also known to be among one of the most popular cloud- enabled languages even used by Google in numerous enterprise-level software apps.
  + Scripting.
  + Desktop GUI applications.

## Python Abstract Syntax :

The compiler.ast module defines an abstract syntax for Python. In the abstract syntax tree, each node represents a syntactic construct. The root of the tree is Module object.

The abstract syntax offers a higher level interface to parsed Python source code. The parser module and the compiler written in C for the Python interpreter use a concrete syntax tree. The concrete syntax is tied closely to the grammar description used for the Python parser. Instead of a single node for a construct, there are often several levels of nested nodes that are introduced by Python’s precedence rules.

The abstract syntax tree is created by the compiler. Transformer module. The transformer relies on the built-in Python parser to generate a concrete syntax tree. It generates an abstract syntax tree from the concrete tree.

The transformer module was created by Greg Stein and Bill Tutt for an experimental Python-to-C compiler. The current version contains a number of modifications and improvements, but the basic form of the abstract syntax and of the transformer are due to Stein and Tutt.

## Ast Nodes :

The compiler.ast module is generated from a text file that describes each node type and its elements. Each node type is represented as a class that inherits from the abstract base class compiler.ast.Node and defines a set of named attributes for child nodes.

**class compiler.ast.Node :**

The Node instances are created automatically by the parser generator. The recommended interface for specific Node instances is to use the public attributes to access child nodes. A public attribute may be bound to a single node or to a sequence of nodes, depending on the Node type. For example, the bases attribute of the Class

## All Node objects offer the following methods :

**getChildren()**

Returns a flattened list of the child nodes and objects in the order they occur. Specifically, the order of the nodes is the order in which they appear in the Python grammar. Not all of the children are Node instances. The names of functions and classes, for example, are plain strings.

## getChildNodes()

Returns a flattened list of the child nodes in the order they occur. This method is like getChildren(), except that it only returns those children that are Node instances.

The While node has three attributes: test, body, and else\_. (If the natural name for an attribute is also a Python reserved word, it can’t be used as an attribute name. An underscore is appended to the word to make it a legal identifier, hence else\_ instead of else.) The if statement is more complicated because it can include several tests.

The If node only defines two attributes: tests and else\_. The tests attribute is a sequence of test expression, consequent body pairs. There is one pair for each if/elif clause. The first element of the pair is the test expression. The second elements is a Stmt node that contains the code to execute if the test is true.

The getChildren() method of If returns a flat list of child nodes. If there are three if/elif clauses and no else clause, then getChildren() will return a list of six elements: the first test expression, the first Stmt, the second text expression, etc.

## The following table lists each of the Node subclasses defined in compiler.ast and each of the public attributes available on their instances. The values of most of the attributes are themselves Node instances or sequences of instances. When the value is something other than an instance, the type is noted in the comment.

## 4.2 Development Environments:

Most Python implementations (including CPython) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which the user enters statements sequentially and receives results immediately.

Other shells, including IDLE and IPython, add further abilities such as auto-completion, session state retention and syntax highlighting.

## IMPLEMENTATIONS

**Reference implementation :**

CPython is the reference implementation of Python. It is written in C, meeting the C89 standard with several select C99 features. It compiles Python programs into an intermediate bytecode which is then executed by its virtual machine. CPython is distributed with a large standard library written in a mixture of C and native Python. It is available for many platforms, including Windows and most modern Unix-like systems. Platform portability was one of its earliest priorities.

## Other implementations :

* PyPy is a fast, compliant interpreter of Python 2.7 and 3.5. Its just-in-time compiler brings a significant speed improvement over CPython but several libraries written in C cannot be used with it.
* Stackless Python is a significant fork of CPython that implements microthreads; it does not use the C memory stack, thus allowing massively concurrent programs. PyPy also has a stackless version.
* MicroPython and CircuitPython are Python 3 variants optimized for microcontrollers. This includes Lego Mindstorms EV3.
* RustPython is a Python 3 interpreter written in Rust.
* The compiler package is a Python source to bytecode translator written in Python. It uses the built-in parser and standard parser module to generate a concrete syntax tree. This tree is used to generate an abstract syntax tree (AST) and then Python bytecode.
* The full functionality of the package duplicates the built-in compiler provided with the Python interpreter. It is intended to match its behavior almost exactly. Why implement another compiler that does the same thing? The package is useful for a variety of purposes. It can be modified more easily than the built-in compiler. The AST it generates is useful for analyzing Python source code.

## The basic interface :

The top-level of the package defines four functions. If you import compiler, you will get these functions and a collection of modules contained in the package.

## compiler.parse(buf)

Returns an abstract syntax tree for the Python source code in buf. The function raises Syntax Error if there is an error in the source code. The return value is a compiler.ast. Module instance that contains the tree.

## compiler.parseFile(path)

Return an abstract syntax tree for the Python source code in the file specified by path. It is equivalent to parse(open(path).read()).

## LIMITATIONS :

There are some problems with the error checking of the compiler package. The interpreter detects syntax errors in two distinct phases. One set of errors is detected by the interpreter’s parser, the other set by the compiler. The compiler package relies on the interpreter’s parser, so it get the first phases of error checking for free. It implements the second phase itself, and that implementation is incomplete. For example, the compiler package does not raise an error if a name appears more than once in an argument list: def f(x, x): ...

A future version of the compiler should fix these problems.

## Unsupported implementations :

Other just-in-time Python compilers have been developed, but are now unsupported:

Google began a project named Unladen Swallow in 2009, with the aim of speeding up the Python interpreter five-fold by using the LLVM, and of improving its multithreading ability to scale to thousands of cores, while ordinary implementations suffer from the global interpreter lock.

Psyco is a just-in-time specializing compiler that integrates with CPython and transforms bytecode to machine code at runtime. The emitted code is specialized for certain data types and is faster than standard Python code.

In 2005, Nokia released a Python interpreter for the Series 60 mobile phones named PyS60. It includes many of the modules from the CPython implementations and some additional modules to integrate with the Symbian operating system. The project has been kept up-to-date to run on all variants of the S60 platform, and several third-party modules are available. The Nokia N900 also supports Python with GTK widget libraries, enabling programs to be written and run on the target device.

## Cross-compilers to other languages :

There are several compilers to high-level object languages, with either unrestricted Python, a restricted subset of Python, or a language similar to Python as the source language:

* + Jython enables the use of the Java class library from a Python program.
  + IronPython follows a similar approach in order to run Python programs on the .NET Common Language Runtime.
  + The RPython language can be compiled to C, and is used to build the PyPy interpreter of Python.
  + Pyjs compiles Python to JavaScript.
  + Cython compiles Python to C and C++.
  + Numba uses LLVM to compile Python to machine code.
  + Pythran compiles Python to C+

## PERFORMANCE ;

A performance comparison of various Python implementations on a non-numerical (combinatorial) workload was presented at EuroSciPy '13.

## API Documentation Generators :

Python API documentation generators include:

* + Sphinx
  + Epydoc
  + HeaderDoc
  + Pydo

## Uses :

Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D parametric modeler like FreeCAD, 3D animation packages such as 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, Motion Builder, Softimage, the visual effects compositor Nuke, 2D imaging programs like GIMP, Inkscape, Scribus and Paint Shop Pro, and musical notation programs like score writer and capella. GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers. Esri promotes Python as the best choice for writing scripts in ArcGIS. It has also been used in several video games, and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go.

Python is commonly used in artificial intelligence projects with the help of libraries like TensorFlow, Keras and Scikit-learn. As a scripting language with modular architecture, simple syntax and rich text processing tools, Python is often used for natural language processing.

Many operating systems include Python as a standard component. It ships with most Linux distributions, AmigaOS 4, FreeBSD (as a package), NetBSD, OpenBSD (as a package) and macOS and can be used from the command line (terminal). Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage. Python is used extensively in the information security industry, including in exploit development. Most of the Sugar software for the One Laptop per Child XO, now developed at Sugar Labs, is written in Python. The Raspberry Pi single-board computer project has adopted Python as its main user-programming language. LibreOffice includes Python, and intends to replace Java with Python. Its Python Scripting Provider is a core feature since Version 4.0 from 7 February 2013.

## PANDAS

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

## Library features :

* + Data Frame object for data manipulation with integrated indexing.
  + Tools for reading and writing data between in-memory data structures and different file formats.
  + Data alignment and integrated handling of missing data.
  + Reshaping and pivoting of data sets.
  + Label-based slicing, fancy indexing, and sub setting of large data sets.
  + Data structure column insertion and deletion.
  + Group by engine allowing split-apply-combine operations on data sets.
  + Data set merging and joining.
  + Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
  + Time series-functionality: Date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
  + Provides data filtration.

**CHAPTER - V**

**5.PROJECT DESCRIPTION**

**5.1 Problem Definition :**

Brain stroke is a critical medical condition that occurs when the blood supply to part of the brain is interrupted or reduced, preventing brain tissue from getting oxygen and nutrients. Timely detection and intervention are crucial to minimize brain damage and save lives. Traditional diagnostic methods rely heavily on clinical evaluations, medical imaging, and the expertise of healthcare professionals. However, these methods may not always provide early warnings, especially in resource-constrained settings.

In recent years, machine learning (ML) has shown significant promise in the field of healthcare, especially in predictive analytics. The goal of this project is to develop a predictive model using machine learning algorithms that can accurately identify individuals who are at high risk of experiencing a brain stroke based on historical health data and lifestyle factors.

The predictive model aims to analyze a variety of features such as:

* Age
* Gender
* Hypertension
* Heart disease
* Smoking status
* Body Mass Index (BMI)
* Work type
* Residence type (urban/rural)
* Average glucose level
* Physical activity

### 

**5.2 Project Overview :**

Brain stroke, a serious medical condition caused by interrupted or reduced blood supply to the brain, can result in lasting brain damage, long-term disability, or even death. With the growing number of stroke cases worldwide, early prediction and prevention have become critical in reducing its devastating effects. The aim of this project is to develop a machine learning-based predictive model that can accurately forecast the likelihood of a brain stroke based on various health and lifestyle parameters. By harnessing the power of data science and artificial intelligence, this project intends to support medical practitioners in making informed, timely decisions for better patient outcomes.

The project begins with a detailed analysis of stroke-related data, which includes features such as age, gender, hypertension, heart disease, marital status, work type, residence type, average glucose level, body mass index (BMI), and smoking status. These features are known to be correlated with an individual’s risk of having a stroke. The dataset, often derived from public medical repositories or synthetic healthcare datasets, is cleaned, preprocessed, and analyzed for missing or anomalous values. Techniques such as imputation, encoding of categorical variables, and normalization are applied to make the data suitable for machine learning algorithms.

Following preprocessing, exploratory data analysis (EDA) is conducted to identify key trends and patterns. This step involves visualizing correlations between stroke and various health indicators. For instance, individuals with hypertension or heart disease show a significantly higher stroke risk. Visual tools such as heatmaps, histograms, and bar graphs help interpret these relationships. This insight guides the selection and tuning of machine learning models, ensuring that important features are not overlooked.

Several supervised learning algorithms are implemented and tested in this project. These include Logistic Regression, Decision Trees, Random Forest, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Gradient Boosting techniques like XGBoost. Each model is trained and validated using a portion of the dataset, with performance evaluated based on metrics such as accuracy, precision, recall, F1-score, and the area under the ROC curve (AUC-ROC). Special attention is given to managing class imbalance—a common issue in stroke datasets where the number of stroke cases is much smaller than non-stroke cases. Techniques like Synthetic Minority Over-sampling Technique (SMOTE) are applied to address this imbalance and enhance model reliability.

The Random Forest and Gradient Boosting models typically demonstrate high performance due to their ability to handle complex relationships and feature interactions. Feature importance analysis from these models reveals the most influential factors in stroke prediction—often age, hypertension, average glucose level, and BMI. These insights are not only critical for model accuracy but also provide valuable medical interpretability, offering healthcare professionals actionable information about high-risk patients.

Finally, the best-performing model is integrated into a user-friendly interface, such as a web-based dashboard or mobile app, allowing clinicians and patients to input relevant health data and receive real-time stroke risk assessments. The system can be further enhanced by integrating electronic health records and continuously learning from new data to improve its predictive capabilities over time.

This project demonstrates how machine learning can be a powerful tool in predictive healthcare. By providing an early warning system for brain stroke, the model supports preventative medicine, reduces the burden on healthcare systems, and most importantly, saves lives. Future enhancements may include the incorporation of deep learning techniques, larger datasets, and real-time patient monitoring to further refine prediction accuracy and usability.

**5.3 System Architecture :**

## Uml Diagram :

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

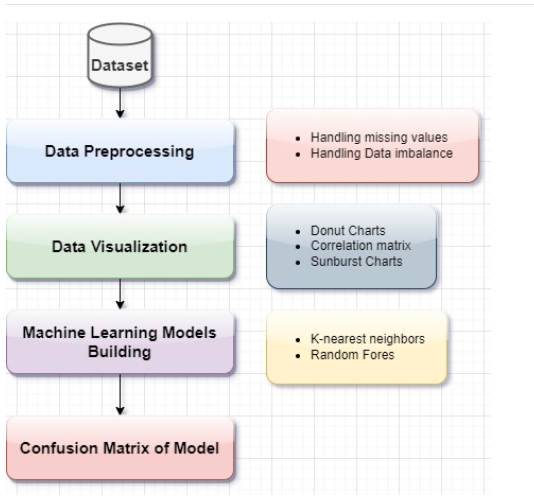
The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

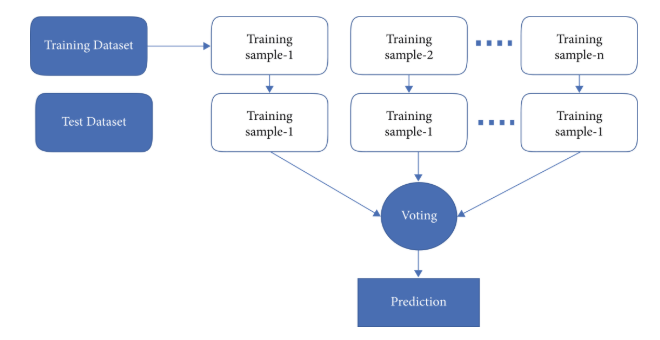
The Primary goals in the design of the UML are as follows:

## provide users a ready-to-use, expressive visual modeling language so that they can develop and exchange meaningful models.

* + Provide extendibility and specialization mechanisms to extend the core concepts.
  + Be independent of particular programming languages and development process.
  + Provide a formal basis for understanding the modeling language.

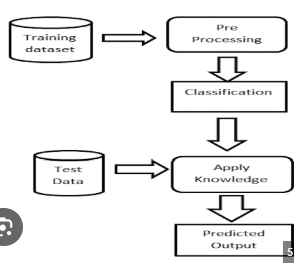
**ARCHITECTURE DIAGRAM :**

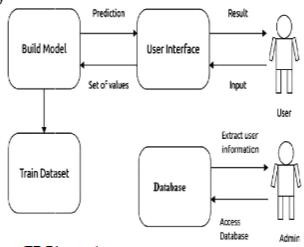




**Data Flow Diagram :**

**LEVEL 0**



**LEVEL 1**

**DATABASE DESIGN**

**Data Dictonary :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Description** | **Sample values** |
| Name | varchar(50) | Specify the name | Rekha |
| Fathername | varchar(25) | Specify the father name | Raja |
| Gender | varchar(50) | Specify the gender | Female |
| Mobilenumber | Text | Specify the mobile number | 8124569785 |
| Emailed | Date | Specify the email id | [rekha@gmail.com](mailto:rekha@gmail.com) |
| Address | Varchar(max) | Specify the address | 3,kk nagar,trichy |
| Username | Varchar(50) | Specify the username | Naveen |
| Password | Varchar(50) | Specify the password | Naveen |
| Date | Date | Specify the date | 4/5/2017 |

**TABLE DESIGNS**

**Table Name:** USER REGISTRATION

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Constraints** |
| Name | varchar(50) | Not Null |
| Fathername | varchar(25) | Not Null |
| Gender | varchar(50) | Not Null |
| Mobilenumber | Text | Not Null |
| Emailed | Date | Not Null |
| Address | Varchar(max) | Not Null |
| Username | Varchar(50) | Not Null |
| Password | Varchar(50) | Not Null |

**5.3 List of Module :**

* DATA COLLECTION :
* DATA PREPROCESS
* DATA VISUALIZATION
* PREDICTION

The Brain Stroke Prediction project using machine learning involves several key modules that work together to develop a predictive model. These modules include data collection, data preprocessing, data visualization, and prediction.

**Module 1: Data Collection :**

The data collection module involves gathering relevant data that can help identify patterns and risk factors associated with brain strokes. This data can come from various sources, including electronic health records (EHRs), medical imaging modalities like MRI and CT scans, and wearable devices like smartwatches and fitness trackers. The data collected can include demographic information, clinical data, imaging data, and lifestyle data.

**Module 2: Data Preprocessing :**

The data preprocessing module involves cleaning, transforming, and preparing the collected data for analysis. This includes handling missing values, outliers, and data normalization. Data preprocessing is a critical step in developing a machine learning model, as it can significantly impact the accuracy and reliability of the model.

**Module 3: Data Visualization :**

The data visualization module involves using various visualization techniques to represent the data in a meaningful way. This can include plots, charts, and heatmaps that help identify patterns and relationships in the data. Data visualization can provide valuable insights into the data and help identify potential risk factors associated with brain strokes.

**Module 4: Prediction :**

The prediction module involves developing a machine learning model that can predict the likelihood of a brain stroke. This can be done using various algorithms, such as logistic regression, decision trees, or neural networks. The model can be trained using the preprocessed data and evaluated using metrics like accuracy, sensitivity, and specificity.

**How the Modules Work Together :**

The modules work together to develop a predictive model that can identify individuals at high risk of brain stroke. The data collection module provides the necessary data, which is then preprocessed and visualized to identify patterns and relationships. The prediction module uses this data to develop a machine learning model that can predict the likelihood of a brain stroke.

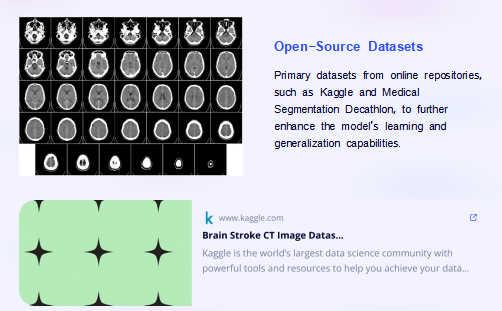
**Benefits of the Project :**

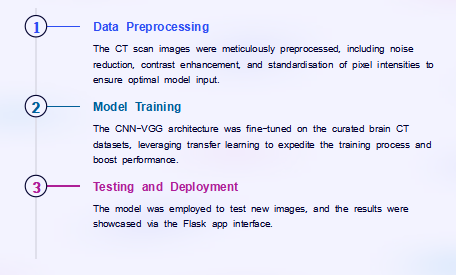
The Brain Stroke Prediction project using machine learning can provide several benefits, including early identification of high-risk individuals, improved patient outcomes, and reduced healthcare costs. By identifying individuals at high risk of brain stroke, healthcare professionals can provide early intervention and prevention, ultimately reducing the incidence of brain strokes.

The Brain Stroke Prediction project using machine learning involves several key modules that work together to develop a predictive model. By collecting relevant data, preprocessing it, visualizing it, and developing a machine learning model, we can identify individuals at high risk of brain stroke and provide early intervention and prevention. This can ultimately improve patient outcomes and reduce healthcare costs.

**5.4 Module Description :**

**Datasets Used for the Project :**





First the dataset is downloaded from the kaggle website. The necessary libraries like Pandas, NumPy, etc. are imported. Then the data is read using pandas.

Since there is no as such requirement for IDs hence its column is removed. By using the function dataframe.isnull().sum(), it returns the number of missing (non-zero) values in the dataset.

It is observed that only in one column of BMI, 201 values are missing. Also in gender only one value is stated as Other while the rest are only male or female, hence for simplicity it is removed from the dataframe.

Then Donut Charts, Correlation matrix, Sunburst charts are made to visualize the data.

After Data Preprocessing, the dataset is split into train and test data(train-3926 , test-982). A model is then built using this new data using two Classification Algorithms. Accuracy is calculated for all these algorithms and compared to get the best-trained model for prediction.

**IMPLEMENTED ALGORITHMS**

The most common disease identified in the medical field is stroke, which is on the rise year after year. Using the publicly accessible stroke prediction dataset, it measured two commonly used machine learning methods for predicting brain stroke recurrence, which are as follows:(i)Random forest (ii)K-Nearest neighbors.

**RANDOM FOREST :**

The classification algorithm chosen was RF classification. RFs are composed of numerous independent decision trees that were trained individually on a random sample of data. These trees are created during training, and the decision trees’ outputs are collected. A process termed voting is used to determine the final forecast made by this algorithm. Each DT in this method must vote for one of the two output classes (in this case, stroke or no stroke). The final prediction is determined by the RF method, which chooses the class with the most votes.

**VISUALIZATIONS :**

In this section, we will be visualizing some interests plots and trends in the data which is used by machine learning models to make predictions. We will also evaluate the performance of different machine learning and deep learning models on a given dataset by comparing various metrics. To identify the best version of each model, we will examine their hyperparameters

**MODEL TRAINING :**

The CNN-VGG architecture underwent fine-tuning on carefully curated brain CT datasets, harnessing the power of transfer learning to accelerate the training process and enhance overall performance. By leveraging transfer learning, the model could efficiently inherit knowledge from pre-trained layers, adapting its features to the specific nuances of the brain CT data. This approach not only expedited the training procedure but also significantly improved the model's ability to accurately analyze and interpret the intricate patterns present in the CT scans. Consequently, the utilization of transfer learning played a pivotal role in optimizing the model's performance within the context of brain image analysis.

**PREDICTION:**

The utilization of the model extended to testing new images, with the subsequent display of results facilitated through the Flask app interface. Employing the model for this purpose enabled the assessment of its performance on unseen data, ensuring its applicability in real-world scenarios. Through the Flask app interface, users could conveniently access and interpret the outcomes of the image testing process, facilitating seamless integration into various applications or workflows. This approach not only validated the model's effectiveness but also provided a user-friendly means of interacting with and utilizing its capabilities for tasks such as image analysis or classification.



**CHAPTER - VI**

**6.SYSTEM TESTING**

**TESTING PROCESS**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTING**

**6.1 Unit Testing :**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results. Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test objectives :**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features To Be Tested :**

* Verify that the entries are of the correct format.
* No duplicate entries should be allowed.
* All links should take the user to the correct page.

**6.2 Integration Testing :**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

**6.3 Functional Testing :**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

**Functional testing is centered on the following items:**

**Valid Input** : identified classes of valid input must be accepted.

**Invalid Input** : identified classes of invalid input must be rejected.

**Functions**  : identified functions must be exercised.

**Output**  : identified classes of application outputs must be exercised.

**Systems/Procedures**: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**6.4 White Box Testing :**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**6.5 Black Box Testing :**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**6.6 Acceptance Testing :**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountere.

**CHAPTER - VII**

**7.SYSTEM IMPLEMENTATION**

**7.1 Coding :**

streamlit==1.27.2

scikit-learn==1.0.2

catboost==1.1.1

joblib==1.2.0

pandas==1.3.5

numpy==1.21.6

boto3==1.28.56

botocore==1.31.57

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x" crossorigin="anonymous">

<link rel="preconnect" href="https://fonts.gstatic.com">

<link href="https://fonts.googleapis.com/css2?family=Titillium+Web:wght@400;500;600;700&display=swap"

rel="stylesheet">

<link href="https://fonts.googleapis.com/css2?family=Rubik:wght@700&display=swap" rel="stylesheet">

<link rel="stylesheet" href="style.css">

<link rel="icon" href="Images/logo.png">

<title>Brain stroke</title>

<style type="text/css">

footer{

position: absolute;

bottom: 0;

right: 0

}

</style>

</head>

<body>

<nav class="navbar navbar-dark" style="background-color: #000980;">

<a class="navbar-brand title" id="brand" href="#">

<img src="Images/logo.png" width="30" height="30" class="d-inline-block align-top" alt="logo">

Brain Stroke

</a>

</nav>

<div class="container" id="main">

<div class="row justify-content-center">

<div class="col-lg-10 col-md-12">

<div class="card m-4">

<div class="card-body" id="box-cont">

<h3 class="card-title py-3 title" id="detect">Detect Brain Stroke Or Normal

</h3>

<p class="px-3">

To identify if a brain stroke either use your web camera and show the Brain Stroke

or upload an image from your device. </p>

<img id="imagePreview" style="height: 300px;" />

<center><input id="imageUpload" type="file" /></center>

<div> Brain Stroke Prediction</div>

<div id="label-container"></div> <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.3.1/dist/tf.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@0.8/dist/teachablemachine-image.min.js"></script>

<script type="text/javascript">

const URL = "https://teachablemachine.withgoogle.com/models/NCilvNgDi/";

let model, labelContainer, maxPredictions;

async function init() {

const modelURL = URL + 'model.json';

const metadataURL = URL + 'metadata.json';

model = await tmImage.load(modelURL, metadataURL);

maxPredictions = model.getTotalClasses();

labelContainer = document.getElementById('label-container');

for (let i = 0; i < maxPredictions; i++) {

labelContainer.appendChild(document.createElement('div'));

}

}

async function predict() {

var image = document.getElementById('imagePreview');

const prediction = await model.predict(image, false);

for (let i = 0; i < maxPredictions; i++) {

const classPrediction =

prediction[i].className + ': ' + prediction[i].probability.toFixed(2);

labelContainer.childNodes[i].innerHTML = classPrediction;

}

}

</script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>

<script type="text/javascript">

function readURL(input) {

if (input.files && input.files[0]) {

var reader = new FileReader();

reader.onload = function (e) {

$('#imagePreview').attr('src', e.target.result);

// $('#imagePreview').css('background-image', 'url(' + e.target.result + ')');

$('#imagePreview').hide();

$('#imagePreview').fadeIn(650);

};

reader.readAsDataURL(input.files[0]);

init().then(() => {

predict();

});

}

}

$('#imageUpload').change(function () {

readURL(this);

});

</script>

</body>

</html>

</div>

</div>

</div>

</div>

import streamlit as st

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestRegressor

from catboost import CatBoostClassifier, CatBoostRegressor

import joblib

import pandas as pd

import numpy as np

import urllib.request

import boto3

from botocore.config import Config

from botocore import UNSIGNED

import io

st.set\_page\_config(

page\_title="Stroke Risk Assessment", page\_icon="🧠",

menu\_items={

'Get Help': 'https://www.kaggle.com/code/frankmollard/machine-learning-process-idea-2-app',

'About': 'Developed by Frank Mollard'

}

)

tab1, tab2 = st.tabs(["Stroke Risk", "Advanced Information"])

tab1.header('Stroke Risk Assessment')

tab1.text(

body =""" This application is designed to assess the risk of stroke

using machine learning algorithms. If a stroke is suspected,

a doctor must always be consulted. This is a medical emergency""",

help=""" This application is based on scientific evidence, but does not

include all risk factors. Therefore, this app should only be

used as a first but not only assessment """

)

tab2.title('Contribution by Model')

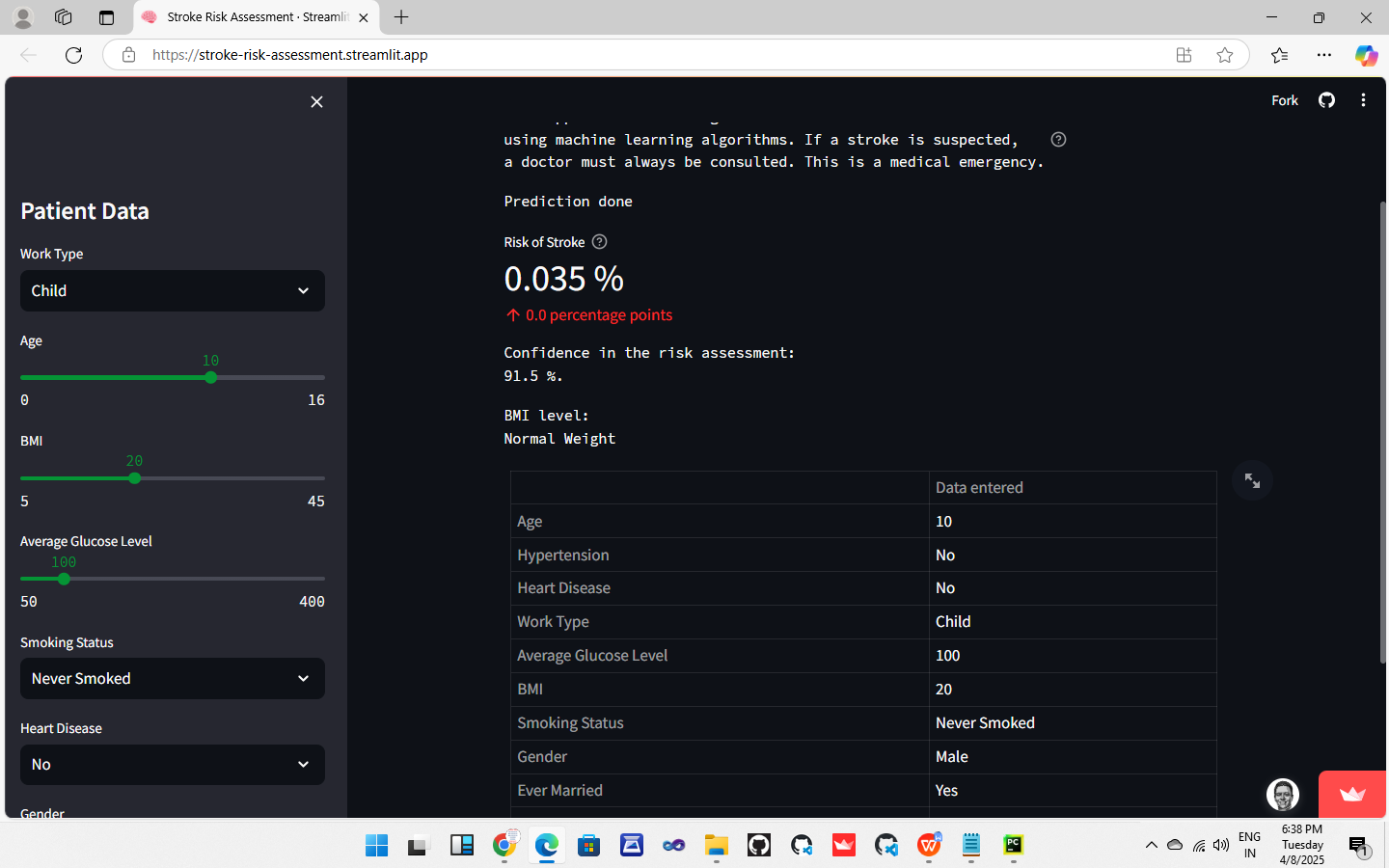
tab2.text(

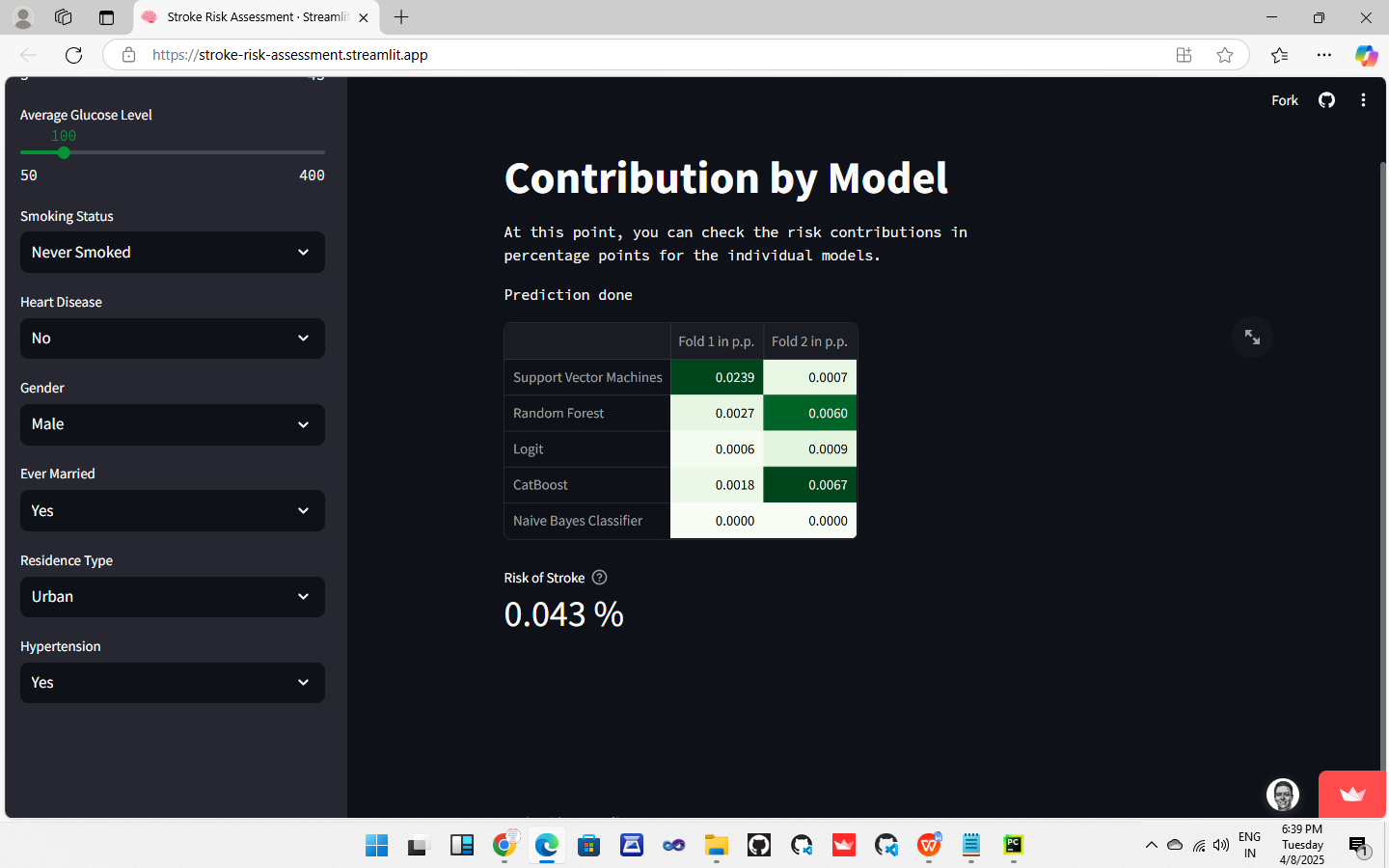
""" At this point, you can check the risk contributions in

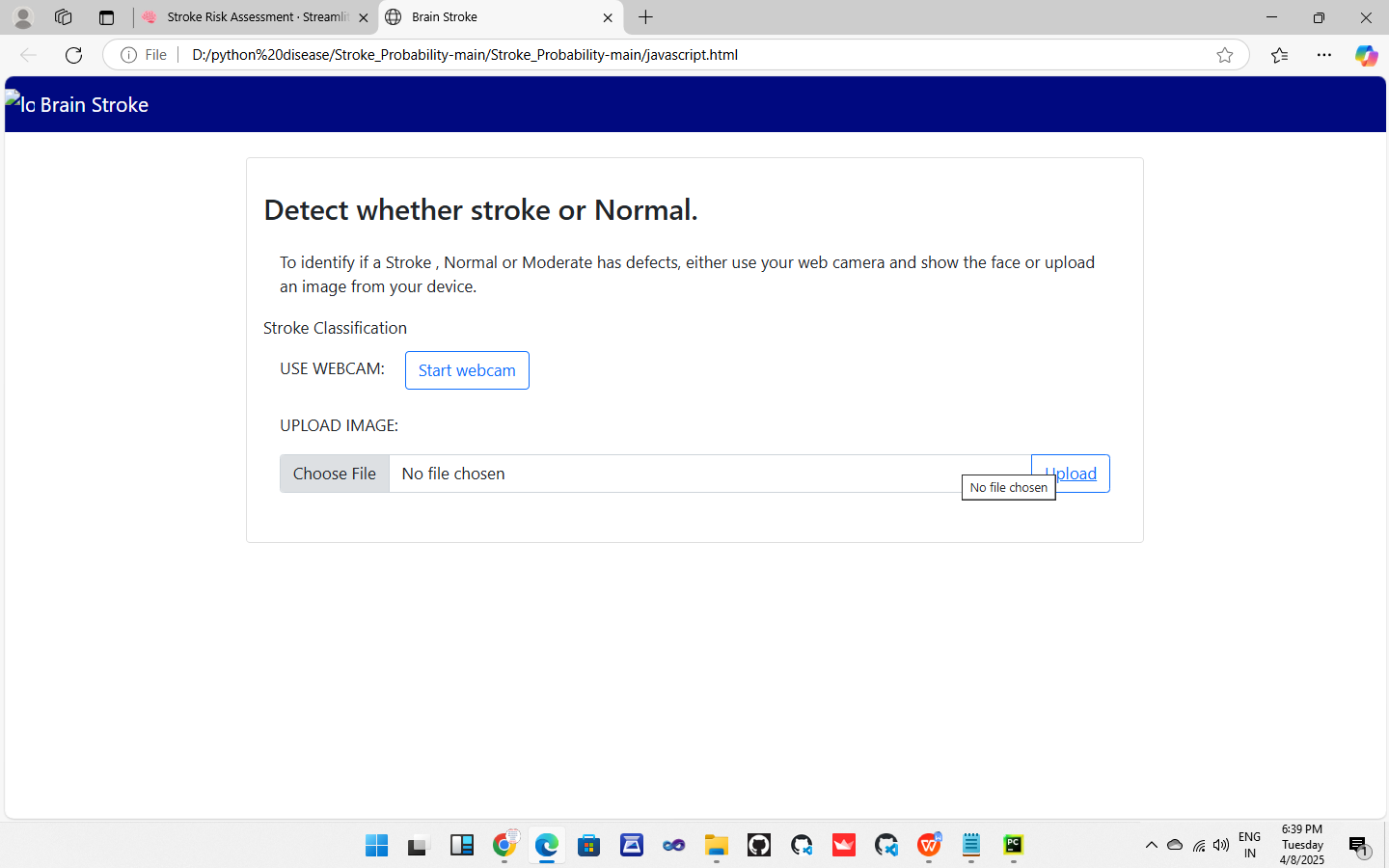
percentage points for the individual models """

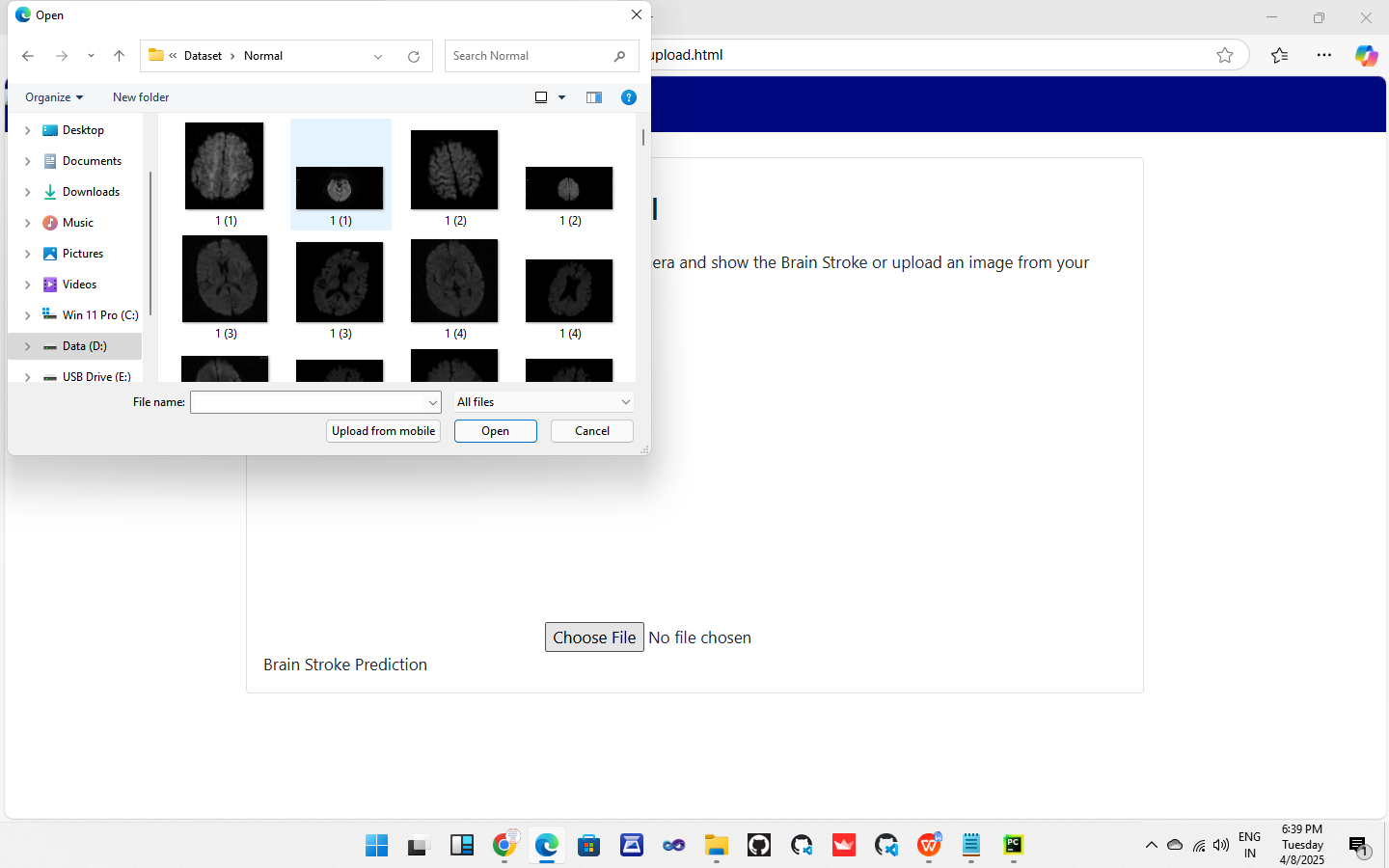
);

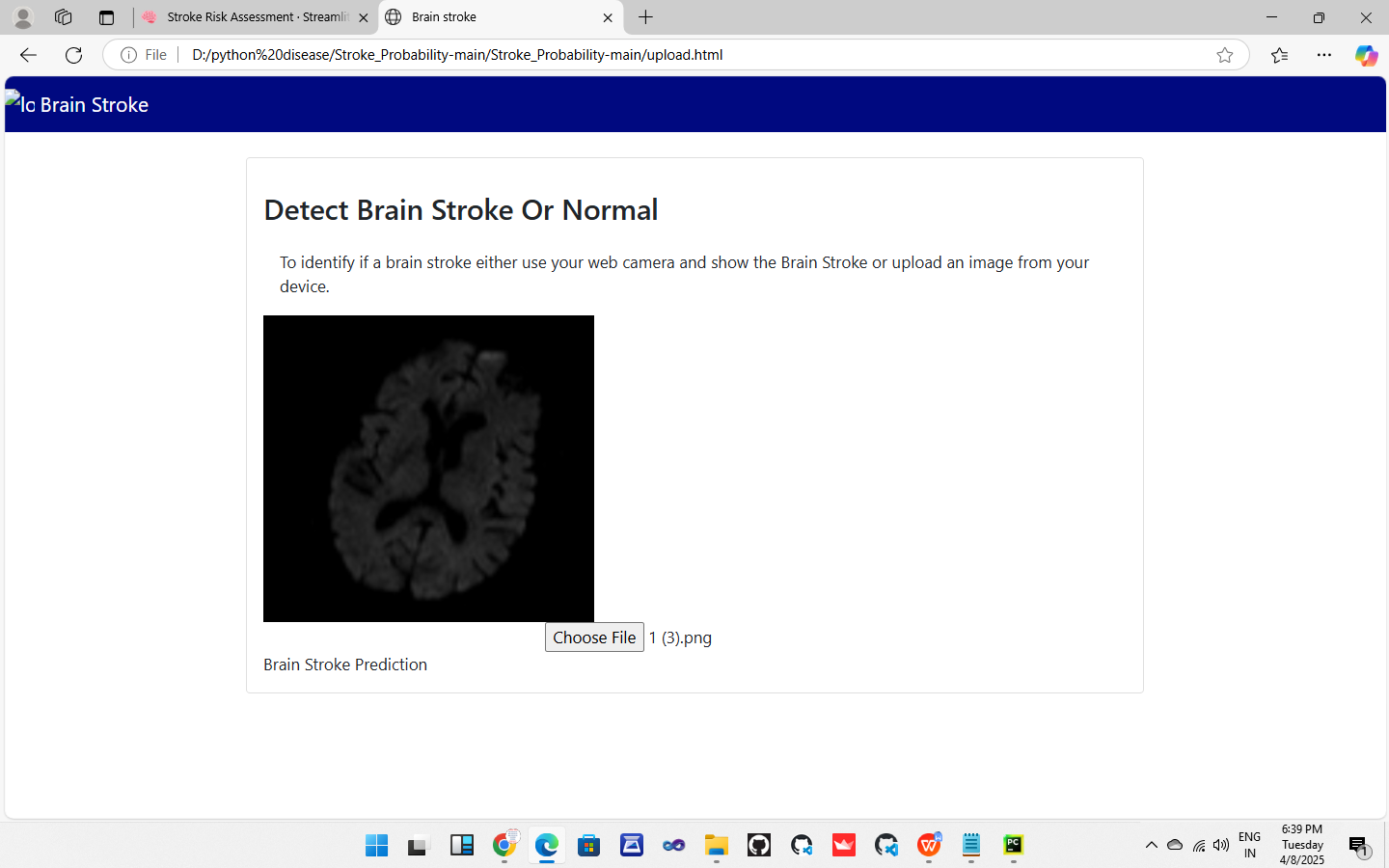
**7.2 Screenshots :**

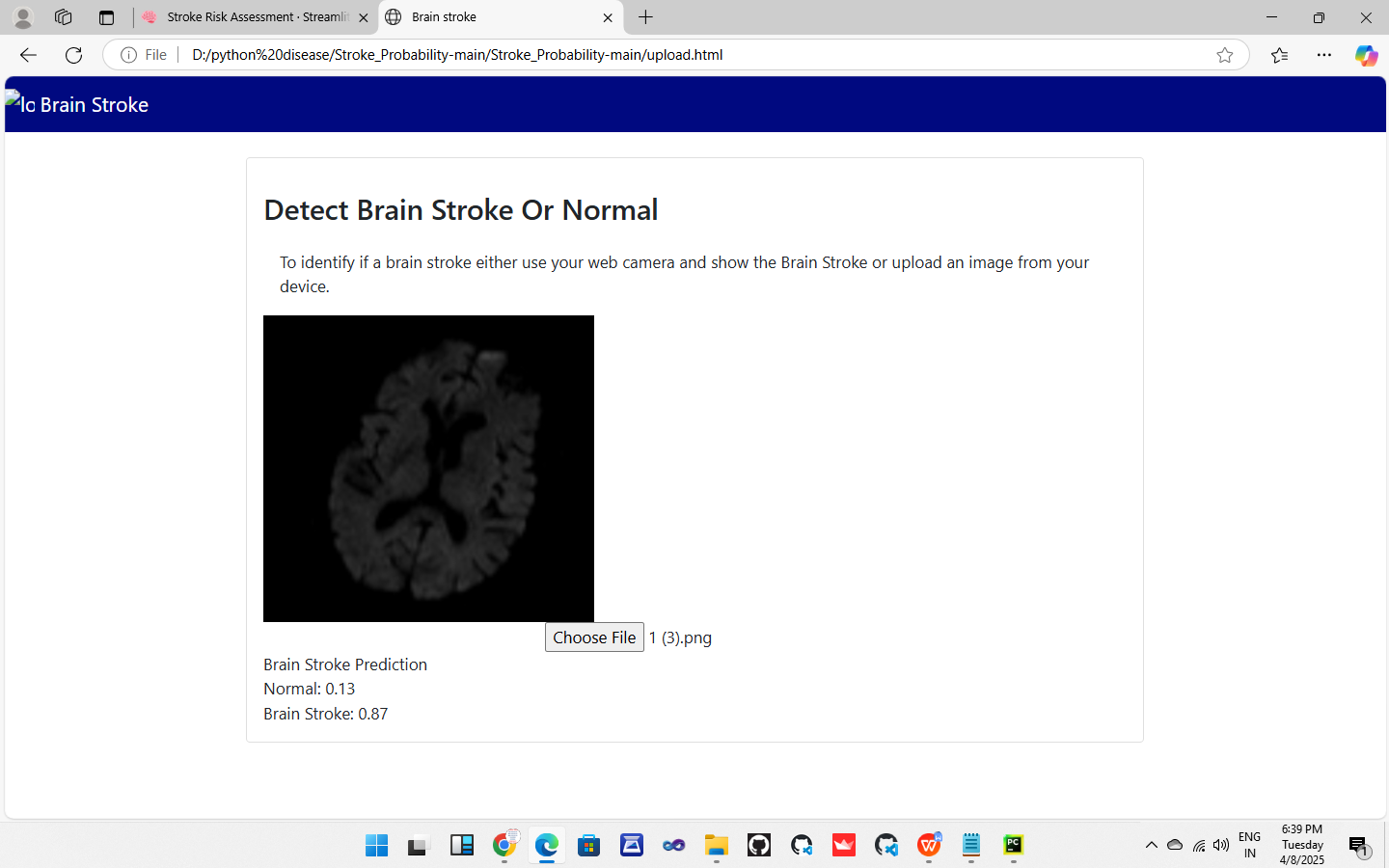












**CHAPTER - VIII**

**8. CONCLUSION**

In this study, we explored the use of machine learning techniques to predict the risk of brain stroke based on key health indicators such as age, hypertension, heart disease, BMI, smoking status, and glucose levels. The predictive model demonstrated promising accuracy, showcasing the potential of data-driven approaches in supporting early diagnosis and prevention of strokes. Early detection through such models can significantly enhance medical response, reduce long-term complications, and ultimately save lives.

The findings highlight the value of integrating predictive analytics into healthcare systems to assist medical professionals in making more informed decisions. By identifying at-risk individuals before symptoms appear, interventions can be implemented more effectively, contributing to better patient outcomes and reduced healthcare costs.

**FUTURE ENHANCEMENT**

While the current model performs well, there is room for further improvement and expansion. Future enhancements may include:

Larger and More Diverse Datasets: Incorporating data from multiple geographic regions, age groups, and ethnicities can improve model generalizability and robustness.

Integration of Medical Imaging: Combining clinical data with brain imaging techniques (such as MRI or CT scans) can enhance prediction accuracy, especially for early-stage stroke signs.

Real-Time Monitoring and IoT Devices: Integrating wearable health monitoring devices can enable real-time stroke risk assessments and alerts.

Deep Learning Techniques: Employing more advanced neural network architectures could improve prediction performance, especially with large and complex datasets.

User-Friendly Interfaces for Medical Use: Developing mobile or web applications for healthcare professionals and patients can promote wider adoption of the model in real-world settings.

Collaborative Diagnosis Tools: Creating systems that combine expert input with AI predictions can help balance human judgment with machine precision.

**CHAPTER -IX**

**9. REFERENCE**

1. Bo Jin , Chao Che,Zhen Liu ,Shulong Zhang ,Xiaomeng Yin And Xiaopeng Wei
2. “Predicting the Risk of Heart Failure with EHR Sequential Data Modelling”. IEEE Access 2018.
3. Aakash Chauhan, Aditya Jain, Purushottam Sharma, Vikas Deep, “Heart Disease
4. Prediction using Evolutionary Rule Learning”, “International Conference on "Computational Intelligence and Communication Technology” (CICT 2018).
5. Ashir Javeed, Shijie Zhou, Liao Yongjian, Iqbal Qasim, Adeeb Noor. “An Intelligent Learning System Based on Random Search Algorithm and Optimized Random Forest Model for Improved Heart Disease Detection”. IEEE Access (Volume: 7) 2019
6. Senthilkumar Mohan, Chandrasegar Thirumalai and Gautam Srivastava. “Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques”. IEEE Access (Volume: 7) 2019.
7. K. Prasanna Lakshmi, Dr. C.R.K.Reddy. “Fast Rule-Based Heart Disease Prediction using Associative Classification Mining”. International Conference on Computer, Communication and Control (IC4) 2015
8. M.Satish, D Sridhar, “Prediction of Heart Disease in Data Mining Technique”,
9. International Journal of Computer Trends & Technology (IJCTT), 2015.
10. Lokanath Sarangi, Mihir Narayan Mohanty, Srikanta Pattnaik, “An Intelligent Decision Support System for Cardiac Disease Detection”, IJCTA, International Press 2015.
11. Boshra Bahrami, Mirsaeid Hosseini Shirvani, “Prediction and Diagnosis of Heart Disease by Data Mining Techniques”, Journal of Multidisciplinary Engineering Science and Technology (JMEST) ISSN: 3159-0040 Vol. 2 Issue 2, February–2015.