**BRAIN DISEASE CLASSIFICATION ALONG WITH AGE ESTIMATION FROM MRI**

**ABSTRACT**

Chronological age of healthy people is able to be predicted accurately using deep neural networks from neuroimaging data, and the predicted brain age could serve as a biomarker for detecting aging-related diseases. Hence, in this proposed method we are using the cascade network of the deep learning which is a SVM,CNN and also the one of the machine learning algorithm named MobileNet. These algorithms are been used to train the brain MRI images which considered in the three classes as the Normal which is not effected with any disease and the other classes which were effected with Alzheimer’s disease (AD) and Mild Cognitive Impairment (MCI). From which we can also find out the ages from the classified images. CNN,SVM and MOBILENET are mainly used for the training of the MRI image dataset, upon where the classification will be performed along with the age estimation.

**Keywords:** MRI, Age estimation, Alzheimer’s disease (AD) and Mild Cognitive Impairment, CNN, SVM, deep learning, MOBILENET, machine learning.

**CHAPTER 1**

**INTRODUCTION**

Alzheimer's disease (AD) is a neurodegenerative disease that usually starts slowly and progressively worsens. It is the cause of 60–70% of cases of dementia. The most common early symptom is difficulty in remembering recent events. As the disease advances, symptoms can include problems with language, disorientation (including easily getting lost), mood swings, loss of motivation, self-neglect, and behavioral issues. As a person's condition declines, they often withdraw from family and society. Gradually, bodily functions are lost, ultimately leading to death. Although the speed of progression can vary, the typical life expectancy following diagnosis is three to nine years.

The cause of Alzheimer's disease is poorly understood. There are many environmental and genetic risk factors associated with its development. The strongest genetic risk factor is from an allele of APOE. Other risk factors include a history of head injury, clinical depression, and high blood pressure. The disease process is largely associated with amyloid plaques, neurofibrillary tangles, and loss of neuronal connections in the brain. A probable diagnosis is based on the history of the illness and cognitive testing with medical imaging and blood tests to rule out other possible causes. Initial symptoms are often mistaken for normal aging. Examination of brain tissue is needed for a definite diagnosis, but this can only take place after death. Good nutrition, physical activity, and engaging socially are known to be of benefit generally in aging, and these may help in reducing the risk of cognitive decline and Alzheimer's; in 2019 clinical trials were underway to look at these possibilities. There are no medications or supplements that have been shown to decrease risk.

No treatments stop or reverse its progression, though some may temporarily improve symptoms. Affected people increasingly rely on others for assistance, often placing a burden on the caregiver. The pressures can include social, psychological, physical, and economic elements. Exercise programs may be beneficial with respect to activities of daily living and can potentially improve outcomes. Behavioral problems or psychosis due to dementia are often treated with antipsychotics, but this is not usually recommended, as there is little benefit and an increased risk of early death.

As of 2015, there were approximately 29.8 million people worldwide with AD with about 50 million of all forms of dementia as of 2020. It most often begins in people over 65 years of age, although up to 10 per cent of cases are early-onset affecting those in their 30's to mid-60. Women get sick more often than men. It affects about 6% of people 65 years and older. In 2015, all forms of dementia resulted in about 1.9 million deaths. The disease is named after German psychiatrist and pathologist Alois Alzheimer, who first described it in 1906. Alzheimer's financial burden on society is large, on par with the costs of cancer and heart disease, with a 2013 study estimating an annual cost of $200 billion (equivalent to $222 billion in 2020) in the US alone

The first symptoms are often mistakenly attributed to aging or stress. Detailed neuropsychological testing can reveal mild cognitive difficulties up to eight years before a person fulfills the clinical criteria for diagnosis of Alzheimer's disease. These early symptoms can affect the most complex activities of daily living. The most noticeable deficit is short term memory loss, which shows up as difficulty in remembering recently learned facts and inability to acquire new information.

Subtle problems with the executive functions of attentiveness, planning, flexibility, and abstract thinking, or impairments in semantic memory (memory of meanings, and concept relationships) can also be symptomatic of the early stages of Alzheimer's disease. Apathy and depression can be seen at this stage, with apathy remaining as the most persistent symptom throughout the course of the disease. The preclinical stage of the disease has also been termed mild cognitive impairment (MCI). This is often found to be a transitional stage between normal aging and dementia. MCI can present with a variety of symptoms, and when memory loss is the predominant symptom, it is termed amnestic MCI and is frequently seen as a prodromal stage of Alzheimer's disease. Amnestic MCI has a greater than 90% likelihood of being associated with Alzheimer's.

In people with Alzheimer's disease, the increasing impairment of learning and memory eventually leads to a definitive diagnosis. In a small percentage, difficulties with language, executive functions, perception (agnosia), or execution of movements (apraxia) are more prominent than memory problems. Alzheimer's disease does not affect all memory capacities equally. Older memories of the person's life (episodic memory), facts learned (semantic memory), and implicit memory (the memory of the body on how to do things, such as using a fork to eat or how to drink from a glass) are affected to a lesser degree than new facts or memories.

Progressive deterioration eventually hinders independence, with subjects being unable to perform most common activities of daily living. Speech difficulties become evident due to an inability to recall vocabulary, which leads to frequent incorrect word substitutions (par aphasias). Reading and writing skills are also progressively lost. Complex motor sequences become less coordinated as time passes and Alzheimer's disease progresses, so the risk of falling increases. During this phase, memory problems worsen, and the person may fail to recognize close relatives. Long-term memory, which was previously intact, becomes impaired.

During the final stage, known as the late-stage or severe stage, the patient is completely dependent upon caregivers. Language is reduced to simple phrases or even single words, eventually leading to complete loss of speech. Despite the loss of verbal language abilities, people can often understand and return emotional signals. Although aggressiveness can still be present, extreme apathy and exhaustion are much more common symptoms. People with Alzheimer's disease will ultimately not be able to perform even the simplest tasks independently; muscle mass and mobility deteriorates to the point where they are bedridden and unable to feed themselves. The cause of death is usually an external factor, such as infection of pressure ulcers or pneumonia, not the disease itself.

**CHAPTER 2**

**LITERATURE REVIEW**

**[1] ] C. R. Jack Jr:** Currently available evidence strongly supports the position that the initiating event in Alzheimer's disease (AD) is related to abnormal processing of β-amyloid (Aβ) peptide, ultimately leading to formation of Aβ plaques in the brain. This process occurs while individuals are still cognitively normal. Biomarkers of brain β-amyloidosis are reductions in CSF Aβ42 and increased amyloid PET tracer retention. After a lag period, which varies from patient to patient, neuronal dysfunction and neurodegeneration become the dominant pathological processes. Biomarkers of neuronal injury and neurodegeneration are increased CSF tau and structural MRI measures of cerebral atrophy. Neurodegeneration is accompanied by synaptic dysfunction, which is indicated by decreased fluorodeoxy glucose uptake on PET. Here a model is proposed that relates disease stage to AD biomarkers in which Aβ biomarkers become abnormal first, before neurodegenerative biomarkers and cognitive symptoms, and neurodegenerative biomarkers become abnormal later, and correlate with clinical symptom severity.

**Summary:** Here a model is proposed that relates disease stage to AD biomarkers in which Aβ biomarkers become abnormal first, before neurodegenerative biomarkers and cognitive symptoms, and neurodegenerative biomarkers become abnormal later, and correlate with clinical symptom severity.

**[2] G. Spulber:** For both clinical and research reasons, it is essential to identify which mild cognitive impairment (MCI) subjects subsequently progress to Alzheimer's disease (AD). The prediction may be facilitated by accelerated whole brain atrophy exhibited by AD subjects. Iterative principal component analysis (IPCA) was used to characterize whole brain atrophy rates using sequential MRI scans for 102 MCI subjects from the Kuopio University Hospital. We modelled the likelihood of progression to probable AD, and found that each additional percent of annualized whole brain atrophy rate was associated with a higher odds ratio (OR) of progression (OR=1.30, p=0.01, 95% CI=1.05-1.60). Our study demonstrates an association between whole brain atrophy rate and subsequent rate of clinical progression from MCI to AD. These findings suggest that IPCA could be an effective brain-imaging marker of progression to AD and useful tool for the evaluation of disease-modifying treatments.

**Summary:** Our study demonstrates an association between whole brain atrophy rate and subsequent rate of clinical progression from MCI to AD. These findings suggest that IPCA could be an effective brain-imaging marker of progression to AD and useful tool for the evaluation of disease-modifying treatments.

**[3] J. Dukart, M. L. Schroeter, and K. Mueller:** Here, we propose a simple method to control for possible effects of confounding variables such as age prior to statistical evaluation of magnetic resonance imaging (MRI) data using support vector machine classification (MOBILENET) or voxel-based morphometry (VBM). We compare MOBILENET results for the classification of 80 AD patients and 79 healthy control subjects based on MRI data with and without prior age correction. Additionally, we compare VBM results for the comparison of three different groups of AD patients differing in age with the same group of control subjects obtained without including age as covariate, with age as covariate or with prior age correction using the proposed method. The results suggest that the approach proposed in this work is generally suited to control for confounding variables such as age in MOBILENET or VBM analyses.

**Summary:** MOBILENET classification using the proposed method resulted in higher between-group classification accuracy compared to uncorrected data. Further, applying the proposed age correction substantially improved univariate detection of disease-related grey matter atrophy using VBM in AD patients differing in age from control subjects.

**[4] K. Franke, G. Ziegler, S. Kloppel, and C. Gaser:** The early identification of brain anatomy deviating from the normal pattern of growth and atrophy, such as in Alzheimer's disease (AD), has the potential to improve clinical outcomes through early intervention. Here, we introduce a framework for automatically and efficiently estimating the age of healthy subjects from their T (1)-weighted MRI scans using a kernel method for regression. This method was tested on over 650 healthy subjects, aged 19-86 years, and collected from four different scanners. The framework proved to be a reliable, scanner-independent, and efficient method for age estimation in healthy subjects, yielding a correlation of r=0.92 between the estimated and the real age in the test samples and a mean absolute error of 5 years. The results indicated favorable performance of the RVM and identified the number of training samples as the critical factor for prediction accuracy. Applying the framework to people with mild AD resulted in a mean brain age gap estimate (Brain AGE) score of +10 years.

**Summary:** Here, we introduce a framework for automatically and efficiently estimating the age of healthy subjects from their T (1)-weighted MRI scans using a kernel method for regression. This method was tested on over 650 healthy subjects, aged 19-86 years, and collected from four different scanners. The framework proved to be a reliable, scanner-independent, and efficient method for age estimation in healthy subjects, yielding a correlation of r=0.92 between the estimated and the real age in the test samples and a mean absolute error of 5 years

**[5] J. H. Cole:** Machine learning analysis of neuroimaging data can accurately predict chronological age in healthy people. Deviations from healthy brain ageing have been associated with cognitive impairment and disease. Here we sought to further establish the credentials of 'brain-predicted age' as a biomarker of individual differences in the brain ageing process, using a predictive modelling approach based on deep learning, and specifically convolutional neural networks (RESNET), and applied to both pre-processed and raw T1-weighted MRI data. Firstly, we aimed to demonstrate the accuracy of RESNET brain-predicted age using a large dataset of healthy adults (N = 2001). Next, we sought to establish the heritability of brain-predicted age using a sample of monozygotic and dizygotic female twins (N = 62). Thirdly, we examined the test-retest and multi-center reliability of brain-predicted age using two samples. RESNET brain-predicted ages were generated and compared to a Gaussian Process Regression (GPR) approach, on all datasets. Input data were grey matter (GM) or white matter (WM) volumetric maps generated by Statistical Parametric Mapping (SPM) or raw data.

**Summary:** Here we sought to further establish the credentials of 'brain-predicted age' as a biomarker of individual differences in the brain ageing process, using a predictive modelling approach based on deep learning, and specifically convolutional neural networks (RESNET), and applied to both pre-processed and raw T1-weighted MRI data.

**[6] B. A. Jonsson:** Machine learning algorithms can be trained to estimate age from brain structural MRI. The difference between an individual’s predicted and chronological age, predicted age difference (PAD), is a phenotype of relevance to aging and brain disease. Here, a new deep learning approach is developed to predict brain age from a T1-weighted MRI. The method was trained on a dataset of healthy Icelanders and tested on two datasets, IXI and UK Biobank, utilizing transfer learning to improve accuracy on new sites. A genome-wide association study (GWAS) of PAD in the UK Biobank data (discovery set: N=12378, replication set: N=4456) yielded two sequence variants, rs1452628-T (β=−0.08, P=1.15×10−9) and rs2435204-G (β=0.102, P=9.73×10−12). The former is near KCNK2 and correlates with reduced sulcal width, whereas the latter correlates with reduced white matter surface area and tags a well-known inversion.

**Summary:** Here, a new deep learning approach is developed to predict brain age from a T1-weighted MRI. The method was trained on a dataset of healthy Icelanders and tested on two datasets, IXI and UK Biobank, utilizing transfer learning to improve accuracy on new sites.

**SYSTEM ANALYSIS & FEASIBILITY STUDY**

**Existing Method:**

This model emphasizes an existing method that which was designed using the some of the algorithms of machine learning. That which are used to classify the brain images which are effected with the some types of tumour or not. Here the process does not performed accurately with the algorithms that were used and classification is not up to the expected levels accuracy.

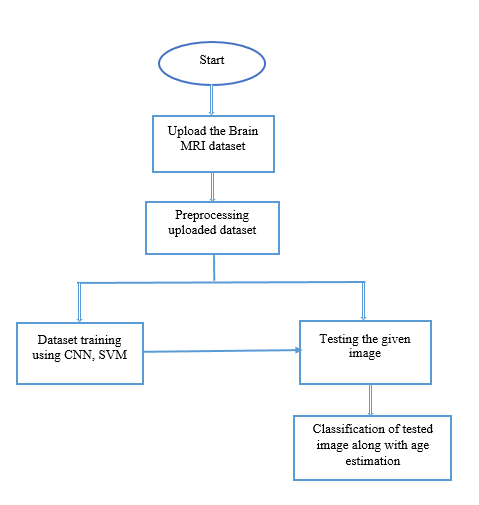
**Disadvantages:**

* Less feature compatibility
* Low accuracy
* Can consider only normal images

**Proposed System:**

In our proposed method we are using the cascade network of the deep learning which is a CNN,SVM and also the one of the machine learning algorithm named MobileNet. These algorithms are been used to train the brain MRI images which considered in the three classes as the Normal which is not effected with any disease and the other classes which were effected with Alzheimer’s disease (AD) and Mild Cognitive Impairment (MCI). From which we can also find out the ages from the classified images. CNN and MOBILENET are mainly used for the training of the MRI image dataset, upon where the classification will be performed along with the age estimation.

**Block Diagram:**



**Fig 1. Block diagram of proposed method**

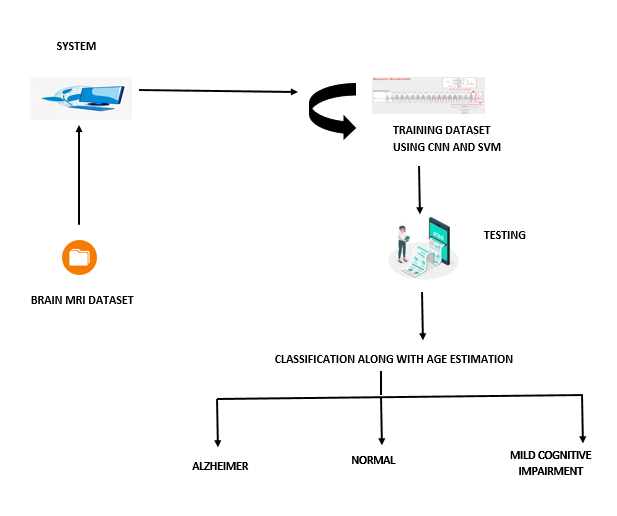
**Advantages**:

• Accurate classification

• Less complexity

• High performance

**ARCHITECTURE**

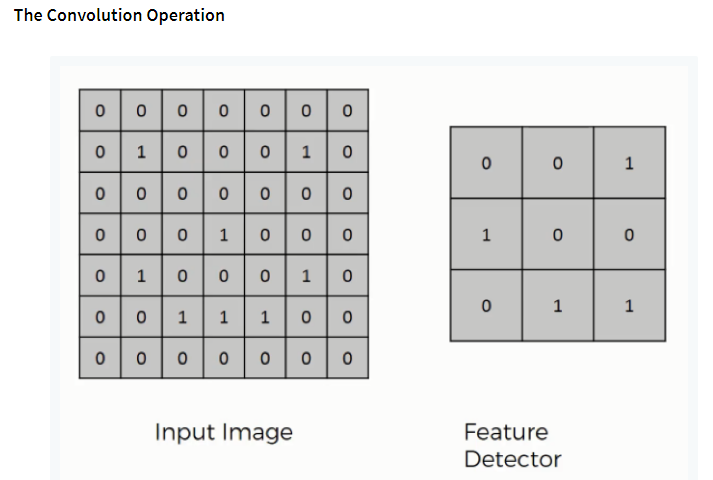


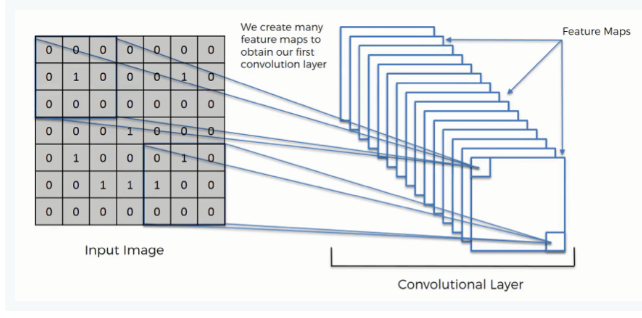
**METHODOLOGY AND ALGORITHMS:**

**1. Convolutional Neural Network**

**Step1: convolutional operation**

The first building block in our plan of attack is convolution operation. In this step, we will touch on feature detectors, which basically serve as the neural network's filters. We will also discuss feature maps, learning the parameters of such maps, how patterns are detected, the layers of detection, and how the findings are mapped out.

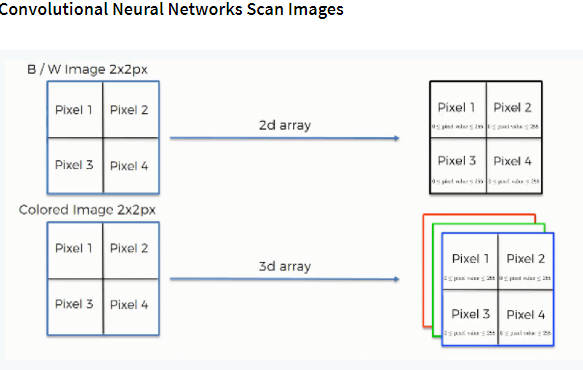




**Step (1b): ReLU Layer**

The second part of this step will involve the Rectified Linear Unit or Relook. We will cover Relook layers and explore how linearity functions in the context of Convolutional Neural Networks.

Not necessary for understanding CNN's, but there's no harm in a quick lesson to improve your skills.



**Step 2: Conv2D**

Keras Conv2D is 2D Convolution Layer; this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs.

Kernel: In image processing kernel is a convolution matrix or masks which can be used for blurring, sharpening, embossing, edge detection, and more by doing a convolution between a kernel and an image

**Step 3: Flattening**

This will be a brief breakdown of the flattening process and how we move from pooled to flattened layers when working with Convolutional Neural Networks.

**Step 4: Full Connection**

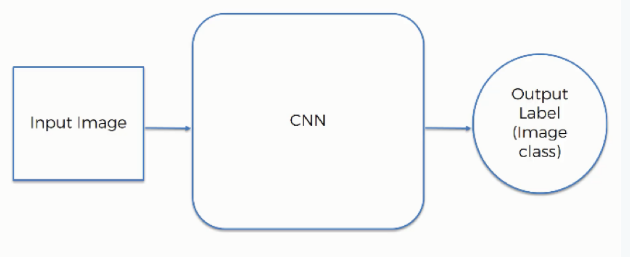
In this part, everything that we covered throughout the section will be merged together. By learning this, you'll get to envision a fuller picture of how Convolutional Neural Networks operate and how the "neurons" that are finally produced learn the classification of images.

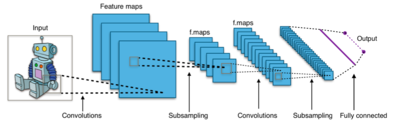
**Summary**

In the end, we'll wrap everything up and give a quick recap of the concept covered in the section. If you feel like it will do you any benefit (and it probably will), you should check out the extra tutorial in which Softmax and Cross-Entropy are covered. It's not mandatory for the course, but you will likely come across these concepts when working with Convolutional Neural Networks and it will do you a lot of good to be familiar with them.

**Convolutional neural network (CNN):**

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution.





**Fig 2. CNN Architecture**

**2.MobileNet:**

As the name applied, the MobileNet model is designed to be used in mobile applications, and it is Tensor Flow’s first mobile computer vision model.

MobileNet uses **depthwise separable** **convolutions.**It significantly **reduces the number of parameters** when compared to the network with regular convolutions with the same depth in the nets. This results in lightweight deep neural networks.

A depthwise separable convolution is made from two operations.

1. **Depthwise convolution.**
2. **Pointwise convolution**.

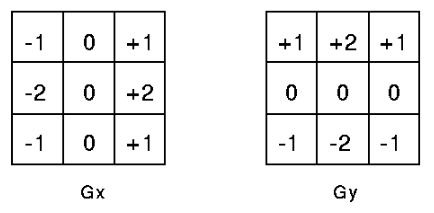
MobileNet is a class of CNN that was open-sourced by Google, and therefore, this gives us an excellent starting point for training our classifiers that are insanely small and insanely fast.



The speed and power consumption of the network is proportional to the number of MACs (Multiply-Accumulates) which is a measure of the number of fused Multiplication and Addition operations.

Depth wise Convolution:

This convolution originated from the idea that a filter’s depth and spatial dimension can be separated- thus, the name separable. Let us take the example of Sobel filter, used in image processing to detect edges.



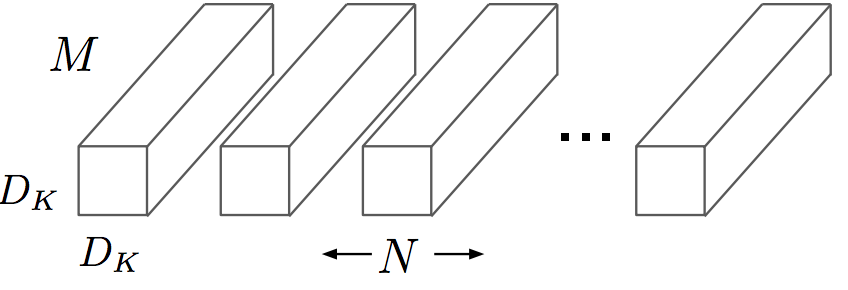
You can separate the height and width dimensions of these filters. Gx filter can be viewed as a matrix product of [1 2 1] transpose with [-1 0 1].

We notice that the filter had disguised itself. It shows it had nine parameters, but it has 6. This has been possible because of the separation of its height and width dimensions.

**Pointwise convolution:**

Convolution with a kernel size of 1x1 that simply combines the features created by the depthwise convolution. Its computational cost is

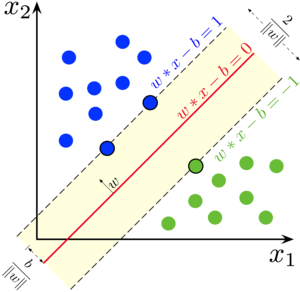
**M \* N \* Df²**.



The main difference between MobileNet architecture and a traditional CNN instead of a single 3x3 convolution layer followed by the batch norm and ReLU. Mobile Nets split the convolution into a 3x3 depth-wise conv and a 1x1 pointwise conv, as shown in the figure.

**3.Support vector machine (SVM):**

Generally, Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a Hyperplane in multidimensional space to separate different classes. SVM generates optimal Hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal Hyperplane (MMH) that best divides the dataset into classes.



**Support Vectors**

Support vectors are the data points, which are closest to the Hyperplane. These points will define the separating line better by calculating margins. These points are more relevant to the construction of the classifier.

**Hyperplane**

A Hyperplane is a decision plane which separates between a set of objects having different class memberships.

**Margin**

A margin is a gap between the two lines on the closest class points. This is calculated as the perpendicular distance from the line to support vectors or closest points. If the margin is larger in between the classes, then it is considered a good margin, a smaller margin is a bad margin.

The main objective is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin. The objective is to select a Hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal Hyperplane in the following steps:

Generate Hyperplane which segregates the classes in the best way. Left-hand side figure showing three Hyperplane black, blue and orange. Here, the blue and orange have higher classification error, but the black is separating the two classes correctly.

Select the right Hyperplane with the maximum segregation from the either nearest data points as shown in the right-hand side figure.



**SVM Kernels**

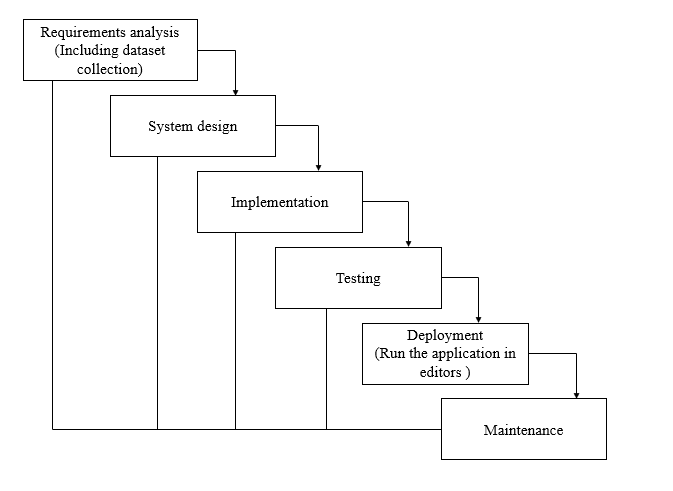
The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space. In other words, you can say that it converts non separable problem to separable problems by adding more dimension to it. It is most useful in non-linear separation problem. Kernel trick helps you to build a more accurate classifier.

The main advantage of SVM is that it can be used for both classification and regression problems. SVM draws a decision boundary which is a Hyperplane between any two classes in order to separate them or classify them. SVM also used in Object Detection and image classification.

**Polynomial Kernel**: A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.

**SOFTWARE DEVELOPMENT LIFE CYCLE – SDLC:**

In our project we use waterfall model as our software development cycle because of its step-by-step procedure while implementing.



**Fig 3**: Waterfall Model

* **Requirement Gathering and analysis** − all possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − the requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − with inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

**FEASIBILITY STUDY**

The feasibility of the project is analysed 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
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**Economic feasibility:**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### Technical feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**Social feasibility:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**SYSTEM REQUIREMENTS SPECIFICATION**

**Functional and non-functional requirements:**

Requirement’s analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

**Functional Requirements**: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

1. Authentication of user whenever he/she logs into the system
2. System shutdown in case of a cyber-attack
3. A verification email is sent to user whenever he/she register for the first time on some software system.

**Non-functional requirements**: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.  
They basically deal with issues like:

* Portability
* Security
* Maintainability
* Reliability
* Scalability
* Performance
* Reusability
* Flexibility

Examples of non-functional requirements:

1. Emails should be sent with a latency of no greater than 12 hours from such an activity.
2. The processing of each request should be done within 10 seconds
3. The site should load in 3 seconds whenever of simultaneous users are > 10000

**SYSTEM SPECIFICATIONS:**

# **H/W Specifications:**

# Processor : I3/Intel Processor

# RAM : 8GB (min)

* Hard Disk : 128 GB

**S/W Specifications:**

* Operating System : Windows 10
* Server-side Script : Python 3.6
* IDE : Jupyter Notebook
* Libraries Used : Numpy, IO, OS, keras.

**SYSTEM DESIGN:**

## **Input Design:**

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties −

* It should serve specific purpose effectively such as storing, recording, and retrieving the information.
* It ensures proper completion with accuracy.
* It should be easy to fill and straightforward.
* It should focus on user’s attention, consistency, and simplicity.
* All these objectives are obtained using the knowledge of basic design principles regarding −
  + What are the inputs needed for the system?
  + How end users respond to different elements of forms and screens.

### Objectives for Input Design:

The objectives of input design are

* To design data entry and input procedures
* To reduce input volume
* To design source documents for data capture or devise other data capture methods
* To design input data records, data entry screens, user interface screens, etc.
* To use validation checks and develop effective input controls.

**Output Design:**

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

### Objectives of Output Design:

The objectives of input design are:

* To develop output design that serves the intended purpose and eliminates the production of unwanted output.
* To develop the output design that meets the end user’s requirements.
* To deliver the appropriate quantity of output.
* To form the output in appropriate format and direct it to the right person.
* To make the output available on time for making good decisions.

**MODULES:**

System

User

1. System:

1.1 Create Dataset:

The dataset containing images of the brain MRI images with the three different classes which are to be classified is split into training and testing dataset with the test size of 30-20%.

1.2 Pre-processing:

Resizing and reshaping the images into appropriate format to train our model.

1.3 Training:

Use the pre-processed training dataset is used to train our model using SVM, CNN and MobileNet algorithm.

1.4 Classification:

The results of our model is display of classified images with their species name as either it is as the MRI report is normal or effected with the disease

**2. User:**

2.1 View training accuracy

After training, user can view the training accuracy

2.2 Upload testing images

The images will be uploaded that are to be classified

2.3 View Results

The classified image results are viewed by user.

**UML DIAGRAMS**

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling 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 Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling 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.

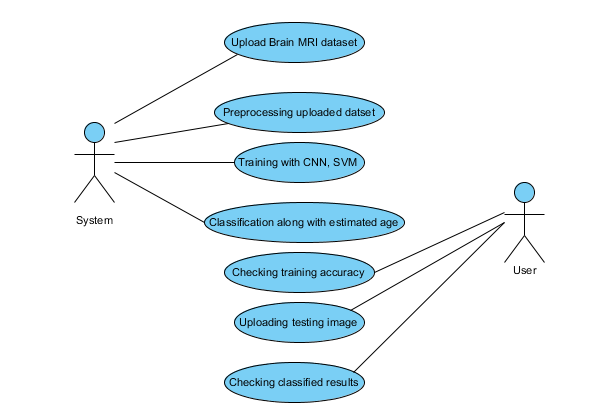
**GOALS:**

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

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

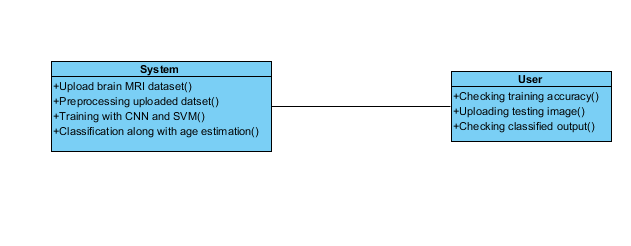
**USE CASE DIAGRAM**

* A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.
* Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.
* The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



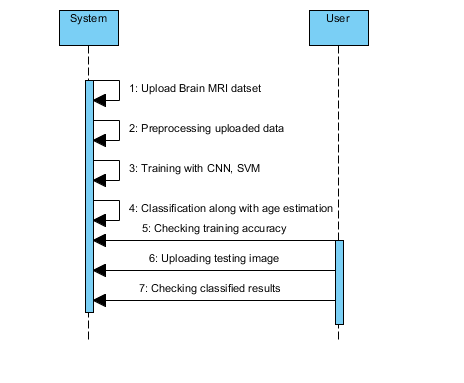
**CLASS DIAGRAM**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information



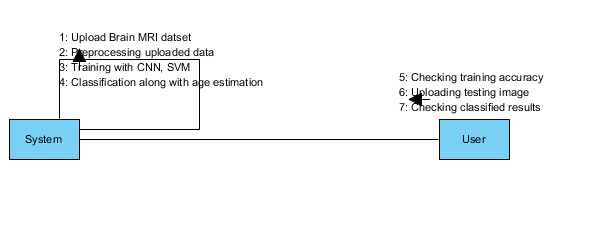
**SEQUENCE DIAGRAM**

* A sequence diagram in Unified Modeling Language (UML) 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. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams



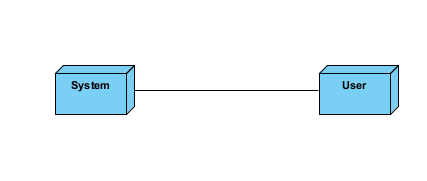
**COLLABORATION DIAGRAM:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



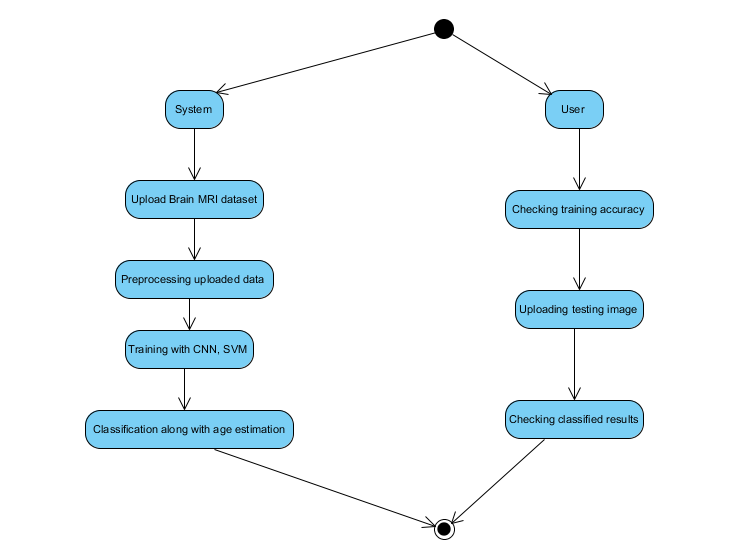
**DEPLOYMENT DIAGRAM**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware’s used to deploy the application.



**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**COMPONENT DIAGRAM**:

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.



**ER DIAGRAM:**

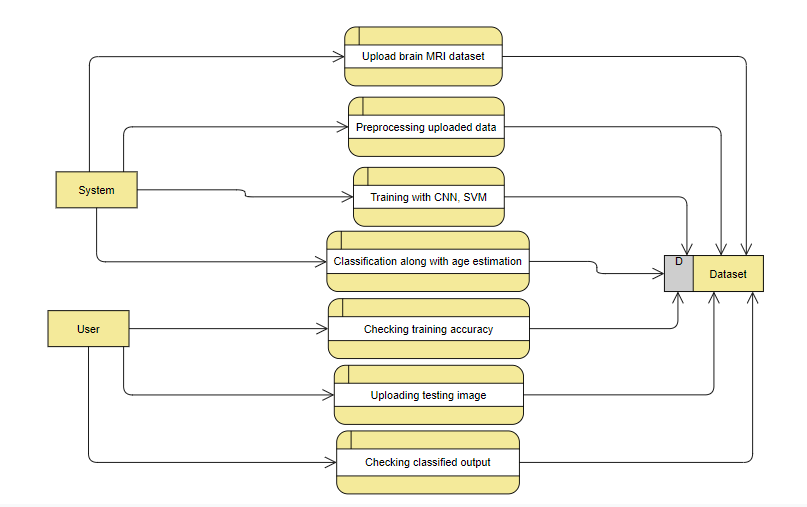
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

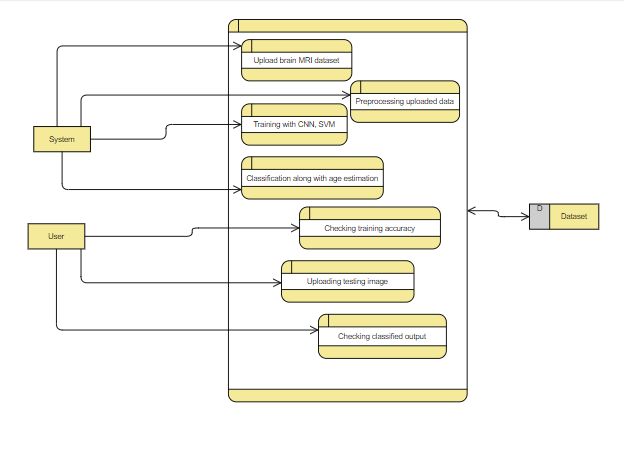
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.

# 

**DFD DIAGRAM:**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.





**OUTPUT SCREEN SHOTS WITH DESCRIPTION.**

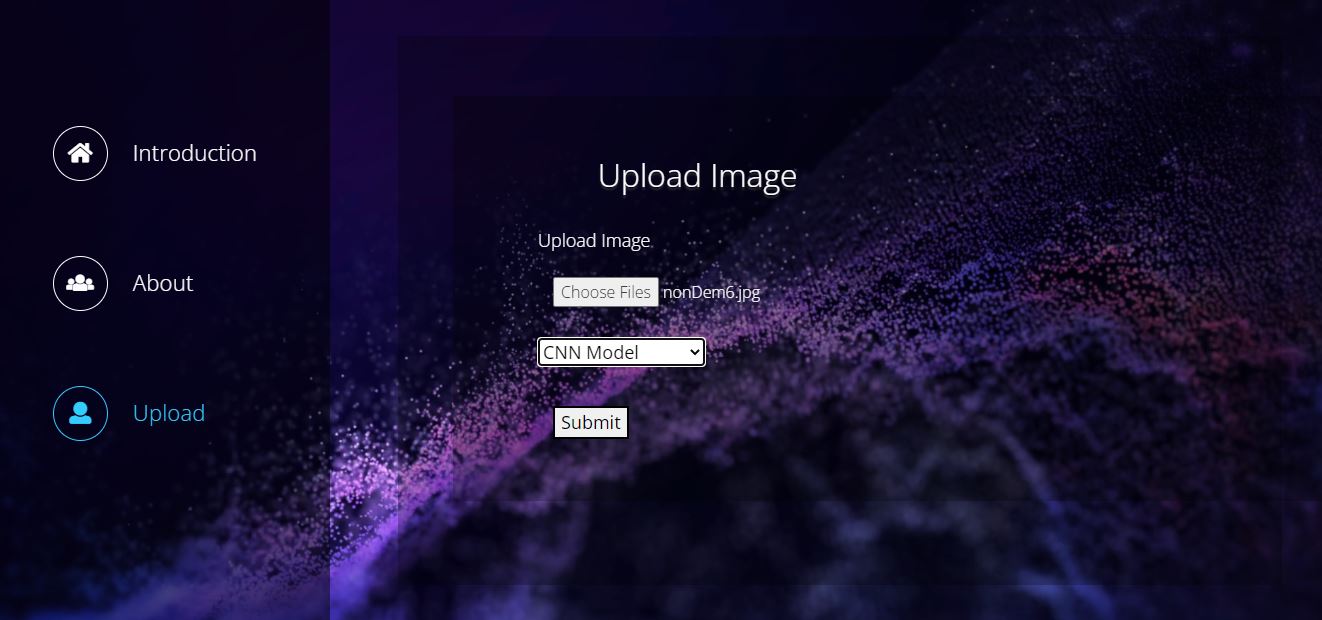
**Home:**



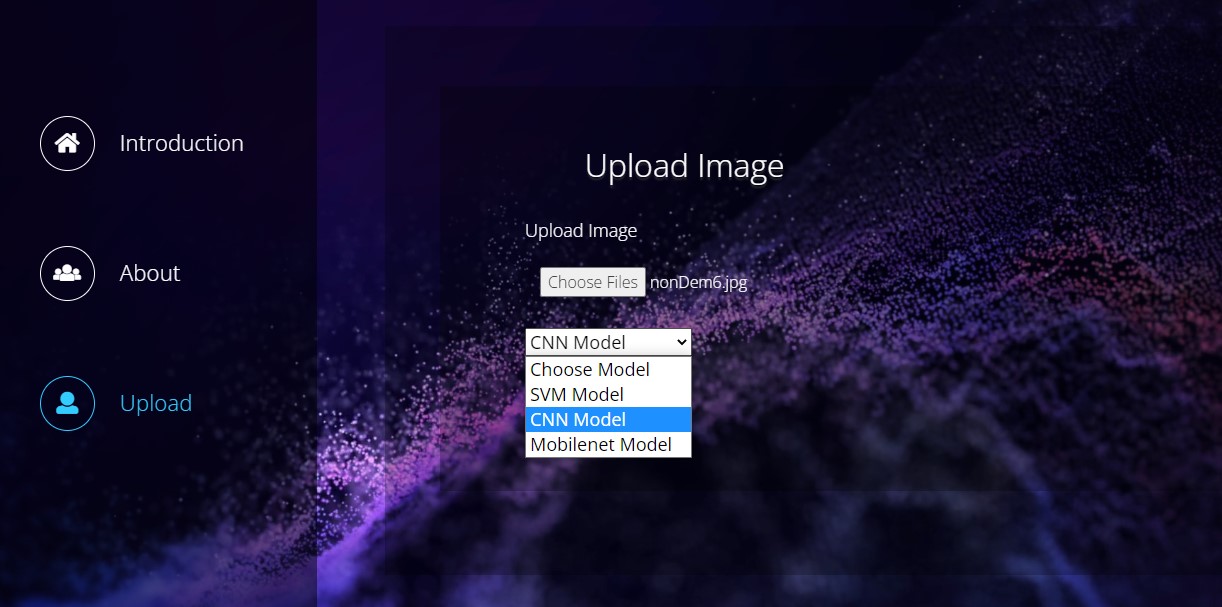
**About:**



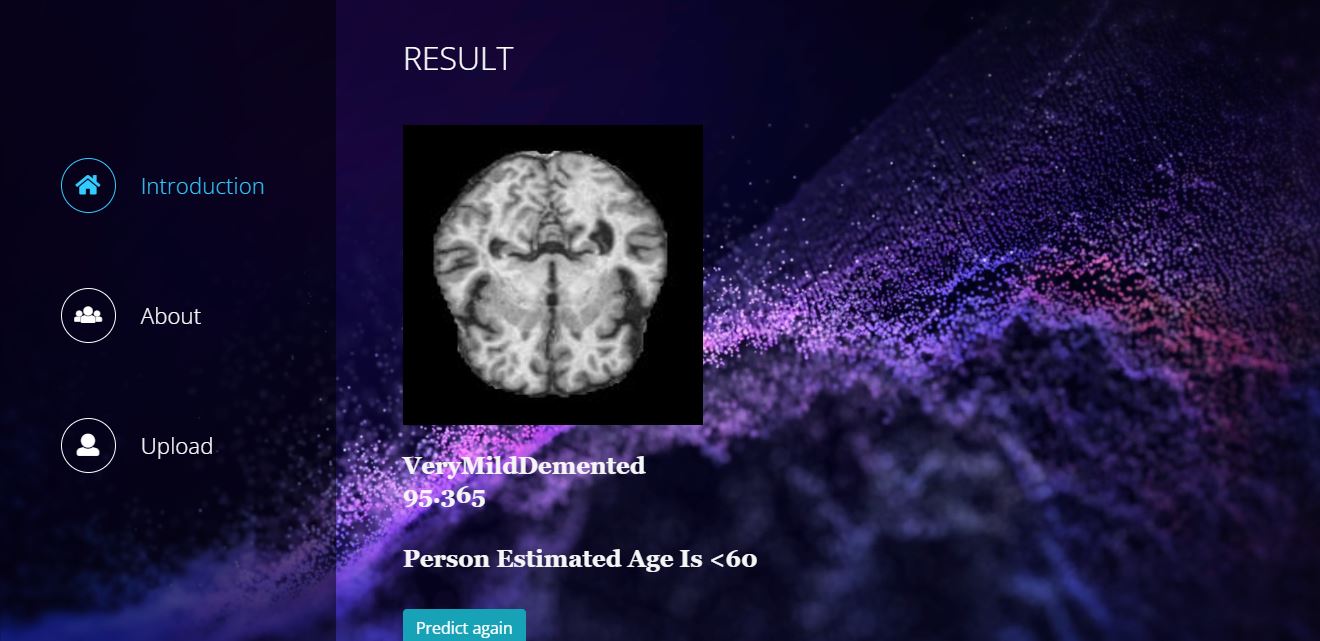
**Upload:**



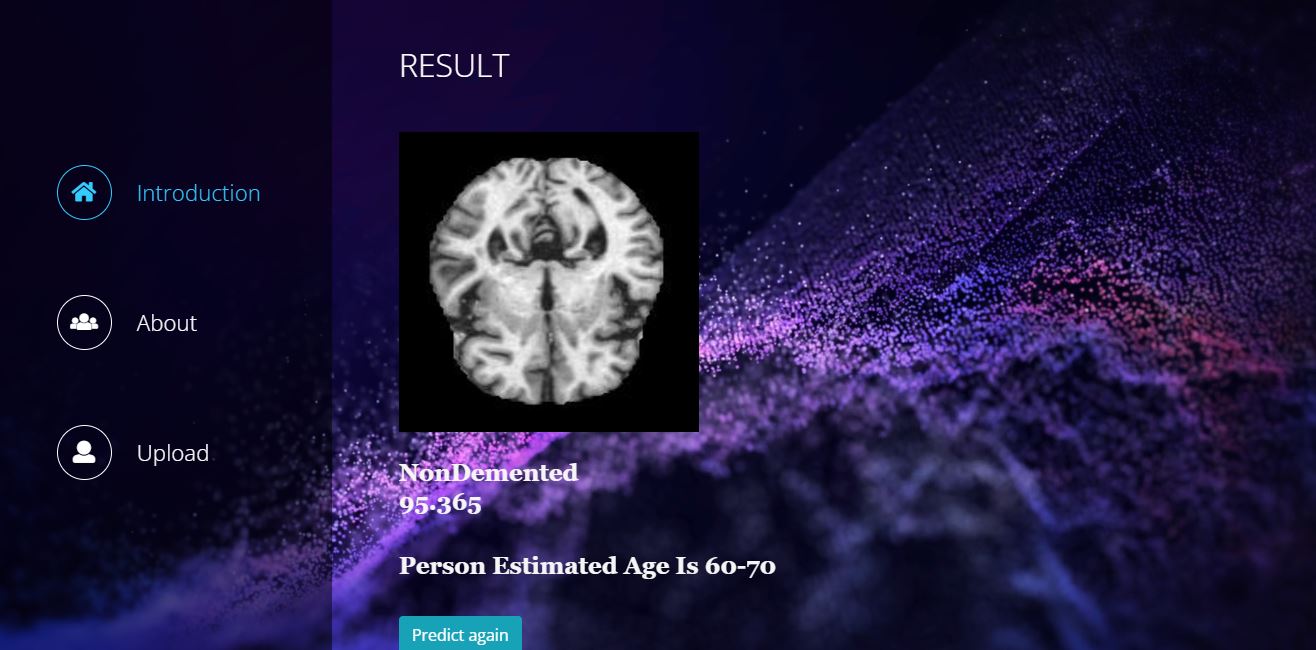
**Model Selection:**



**Result:**



**Result:**



**TEST CASES:**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Result** |
| Input image | Tested for the classified output for brain MRI images | Success |
| Age Estimation | Along with the classification age should be estimated | Success |

**Test cases Model building:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **Test cases** | **I/O** | **Expected O/T** | **Actual O/T** | **P/F** |
| 1 | Read the dataset. | Dataset path. | Dataset need to read successfully. | Dataset fetched successfully. | P |
| 2 | Performing pre-processing on the dataset | Pre-processing part takes place | Pre-processing should be performed on dataset | Pre-processing successfully completed. | P |
| 3 | Model Building | Model Building for the clean data | Need to create model using required algorithms | Model Created Successfully. | P |
| 4 | Classification | Input image provided. | Output should be the either pneumonia affected or normal classification | Model classified successfully | P |
| 5 | Age Estimation | Input Image provided | Output should be the estimated age for the classified output | Successfully estimated | P |

**CONCLUSION:**

In this project we have successfully classified the brain MRI scanned images are normal or effected with the Alzheimer’s disease or Mild Cognitive Impairment. Along with the classification, the age is also estimated based on the classified output. This process is performed using the deep learning based RESNET and machine learning based MOBILENET algorithms.

**FUTURE SCOPE:**

This proposed process can be extended further and can be used for the age estimations of the persons if their information is absent and the process for diagnosis can be suggested based ion their estimated age and the disease that was effected.

**REFERENCES:**

[1] C. Lopez-Ot ´ ´ın, M. A. Blasco, L. Partridge, M. Serrano, and G. Kroemer, “The hallmarks of aging,” Cell, vol. 153, no. 6, pp. 1194–1217, 2013.

[2] J. Bijsterbosch, “How old is your brain?” Nature neuroscience, vol. 22, no. 10, pp. 1611–1612, 2019.

[3] C. R. Jack Jr, et al., “Hypothetical model of dynamic biomarkers of the Alzheimer’s pathological cascade,” The Lancet Neurology, vol. 9, no. 1, pp. 119–128, 2010.

[4] I. Driscoll, et al., “Longitudinal pattern of regional brain volume change differentiates normal aging from MCI,” Neurology, vol. 72, no. 22, pp. 1906–1913, 2009.

[5] G. Spulber, et al., “Whole Brain Atrophy Rate Predicts Progression From MCI to Alzheimer’s Disease,” Neurobiology of aging, vol. 31, no. 9, pp. 1601–1605, 2010.

[6] J. Dukart, M. L. Schroeter, K. Mueller, The Alzheimer’s Disease Neuroimaging Initiative, “Age correction in dementia–matching to a healthy brain,” PloS one, vol. 6, no. 7, 2011.

[7] R. Brookmeyer, E. Johnson, K. Ziegler-Graham, and H. M. Arrighi, “Forecasting The Global Burden of Alzheimer’s Disease,” Alzheimer’s & dementia, vol. 3, no. 3, pp. 186–191, 2007.

[8] M. Liu, J. Zhang, E. Adeli, and D. Shen, “Deep multi-task multi-channel learning for joint classification and regression of brain status,” in International conference on medical image computing and computer assisted intervention. Springer, 2017, pp. 3–11.

[9] K. Franke, G. Ziegler, S. Kloppel, C. Gaser, The Alzheimer’s Disease ¨ Neuroimaging Initiative, “Estimating the age of healthy subjects from T1-weighted MRI scans using kernel methods: exploring the influence of various parameters,” Neuroimaging, vol. 50, no. 3, pp. 883–892, 2010.

[10] N. U. Dosenbach, et al., “Prediction of individual brain maturity using fMRI,” Science, vol. 329, no. 5997, pp. 1358–1361, 2010.

[11] J. H. Cole, et al., “Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker,” Neuro Image, vol. 163, no. March, pp. 115–124, 2017.

[12] J. H. Cole, “Multi-modality neuroimaging brain-age in UK Biobank: relationship to biomedical, lifestyle and cognitive factors,” Neurobiology of Aging, 2020.

[13] J. H. Cole and K. Franke, “Predicting Age Using Neuroimaging: Innovative Brain Ageing Biomarkers,” Trends in Neurosciences, vol. 40, no. 12, pp. 681–690, 2017.

[14] J. H. Cole, et al., “Brain age predicts mortality,” Molecular Psychiatry, vol. 23, no. 5, pp. 1385–1392, 2018.

[15] B. A. Jonsson, et al., “Brain age prediction using deep learning uncovers associated sequence variants,” Nature Communications, vol. 10, no. 1, pp. 1–10, 2019.

**BIBLIOGRAPHY:**

**SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS:**

**Installing Python:**

1. To download and install Python visit the official website of Python <https://www.python.org/downloads/> and choose your version.



1. Once the download is complete, run the exe for install Python. Now click on Install Now.
2. You can see Python installing at this point.
3. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

**Installing PyCharm:**

1. To download PyCharm visit the website <https://www.jetbrains.com/pycharm/download/> and click the "DOWNLOAD" link under the Community Section.



1. Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.
2. On the next screen, Change the installation path if required. Click “Next”.
3. On the next screen, you can create a desktop shortcut if you want and click on “Next”.
4. Choose the start menu folder. Keep selected Jet Brains and click on “Install”.
5. Wait for the installation to finish.
6. Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.
7. After you click on "Finish," the Following screen will appear.



9. You need to install some packages to execute your project in a proper way.

10. Open the command prompt/ anaconda prompt or terminal as administrator.

11. The prompt will get open, with specified path, type “pip install package name” which you want to install (like NumPy, pandas, sea born, scikit-learn, Matplotlib, Pyplot)

Ex: Pip install NumPy



# **INTRODUCTION TO PYTHON**

* Python

### What Is a Script?

Up to this point, I have concentrated on the interactive programming capability of Python.  This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program.  Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor.  Then you can execute each of the individual versions.  In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad, Microsoft WordPad, Microsoft Word,*or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

Python concepts

If you’re not interested in the how’s and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it’s a great one to start programming with.

• Open-source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-is to interface with C++ (via SWIG)

• Great interactive environment

• Great interactive environment

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

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

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

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

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

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

Python Features

Python's features include −

* Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* Easy-to-read − Python code is more clearly defined and visible to the eyes.
* Easy-to-maintain − Python's source code is fairly easy-to-maintained.
* A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* Extendable − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* Databases − Python provides interfaces to all major commercial databases.
* GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* Scalable − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs. Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

## Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

## Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

## Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

## Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

## Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .pie files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

# 20 Python libraries

1. Requests. The most famous http library written by Kenneth remits. It’s a must have for every python developer.

2. Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won’t use any other.

3. Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.

4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

5. SQL Alchemy. A database library. Many love it and many hate it. The choice is yours.

6. Beautiful Soup. I know it’s slow but this xml and html parsing library is very useful for beginners.

7. Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.

8. Numbly. How can we leave this very important library? It provides some advance math functionalities to python.

9. Skippy. When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

10. Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

11. Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.

12. Piglet. A 3d animation and game creation engine. This is the engine in which the famous [python port](https://github.com/fogleman/Minecraft) of mine craft was made

13. Pit. A GUI toolkit for python. It is my second choice after python for developing GUI’s for my python scripts.

14. Pit. Another python GUI library. It is the same library in which the famous Bit torrent client is created.

15. Scaly. A packet sniffer and analyser for python made in python.

16. Pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. Notch. Natural Language Toolkit – I realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. Simply. Simply can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. I Python. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

Numpy

Humpy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In numbly dimensions are called axes. The number of axes is rank.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

Matplotlib

• High quality plotting library.

Python class and objects

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by sub classing a class, you can change the behaviour of the program by simply adding new components to it rather than rewriting the existing components.

As we’ve seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python’s object model and the object interfaces are more consistent to Python’s built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python’s built-in operation methods

Exceptions

I’ve talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program’s flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn’t exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don’t occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a Key Error exception.

• Searching a list for a non−existent value will raise a Value Error exception

. • Calling a non−existent method will raise an Attribute Error exception.

• Referencing a non−existent variable will raise a Name Error exception.

• Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don’t need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don’t put any overhead on the code processing.

Because exceptions aren’t supposed to happen very often, they aren’t processed until they occur.

Exceptions can be thought of as a special form of the if/elf statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren’t processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won’t spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won’t have to do this very often but it’s nice to have the option when necessary.

However, before making your own exceptions, make sure there isn’t one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

Testing code

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

His entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

## To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.

## There are three types of functions in python:

## Help (), min (), print ().

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

* global names of a module
* local names in a function or method invocation
* built-in names: this namespace contains built-in functions (e.g. abs(), camp(), ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX: Here, you register call-backs for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

## Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format tinplating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

## Comparing web frameworks

There is also a repository called [compare-python-web-frameworks](https://github.com/mattmakai/compare-python-web-frameworks) where the same web application is being coded with varying Python web frameworks, tinplating engines and object.

## Web framework resources

* When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
* Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
* What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
* Jingo vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
* This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
* Python’s web frameworks benchmarks  is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
* What web frameworks do you use and why are they awesome? Is a language agnostic Reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
* This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?” The votes aren't as important as the list of the many frameworks that are available to Python developers.

## Web frameworks learning checklist

1. Choose a major Python web framework (Jingo or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the [deployment](https://www.fullstackpython.com/deployment.html) section to make it accessible on the web.

2. SYSTEM STUDY

### SYSTEM TESTING

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 TESTS

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.

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.

Functional test

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.

SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

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.

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.

6.1 Unit Testing:

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 strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

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

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.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

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

6.3 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 encountered.