**INTRODUCTION**

**1. Introduction**

**a. Problem Definition:**

Lung cancer has become one of the most common causes of cancer in both men and women. A large number of people die every year due to lung cancer. The purpose of this project is to detect early signs of lung cancer and improve accuracy, sensitivity and specify.

The disease has different stages whereby it starts from the small tissue and spreads throughout the different areas of the lungs by a process called metastasis. It is the uncontrolled growth of unwanted cells in the lungs. It is estimated that around 12,203 individuals had lung cancer 2016, 7130 males and 5073 females deaths from lung cancer in 2016 were 8839

**b. Aim and Objective of the Project:**

**Aim:** To detect whether the given CT scan image is cancerous or not.

**Objective:**

* To improve the efficiency in detecting the Lung Cancer by performing image processing and using SVM.
* Sing Discrete Waveform Transform for image compression and extracting features using Grey Level Co-occurrence Matrix(GLCM).
* Using Support Vector Machine as classifier, separating hyper plane- a machine learning algorithm.

**c. Scope and Limitation of the Project:**

**Scope:**

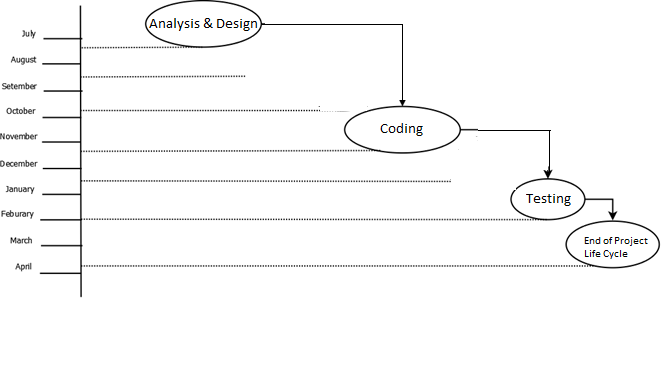
* This project works on social issue, Lung cancer which has large impact on the community throughout the world. 2/3rd of affected that get diagnosed at early stage which helped them to survive.
* This software used as a classifier to detect early lung cancer to improve the accuracy and sensitivity of the system in preference to other machine learning languages. In this software we are going to detect early signs of lung cancer. Whereas, we are not going to ensure to measure the size of nodule and the stages of lung cancer.

**Limitation:**

1. The system must classify the CT scan image as ‘Malignant’ or ‘Benign’.
2. The system does not mention the stages of the cancer detected.
3. The input must be a valid image.

**d.Timeline of the Project:**

We have used classic life cycle paradigm also called “Water-Fall Model”. For software engineering which is sequential approach for software development that begins at the system level and progress through analysis, design, coding, testing and maintenance.

****We completed the analysis and design phase in November 2018. The coding part was completed upto February 2019 and the testing part was completed in March 2019.

**e. Project Cost:**

**Hardware cost:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.no** | **Component** | **Quantity\*price** | **Price** |
| 1. | Computer System | 1\*32,000 | 32000 |
| **Total** | | | **32000** |

**Cost estimation based on COCOMO model**

In this project the Cost Estimation based on COCOMO (Constructive Cost Model) the formula for this Model is follows:

**Effort = Constant \* (Size) scale factor \* Effort Multiplier**

* Effort in terms of person-months
* Constant : 2.45 in 1998 based on Organic Mode
* Size : Estimated size in KLOC
* Scale Factor : Combined process factors
* Effort Multiplier (EM) : Combined effort factors

The basic COCOMO equations takes the form :

* Effort Applied (E) = ab (KLOC)bb [man-months]
* Development Time (D) =cb (Effort Applied)db [months]
* People required (P) = Effort Applied / Development Time [count]

Where, KLOC is the estimated number of delivered lines (expressed in thousands) of code for project,

The coefficients ab, bb,cb and db are given in the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Software Project** | **ab** | **bb** | **cb** | **db** |
| **Organic** | 2.4 | 1.05 | 2.5 | 0.38 |
| **Semi-detached** | 3.0 | 1.12 | 2.5 | 0.35 |
| **Embedded** | 3.6 | 1.20 | 2.5 | 0.32 |

**Organic Mode**

Effort Applied (E) = 3.0\*(0.264)1.12 = 0.68 Man-months

Development Time (D) = 2.5\*(0.68)0.35 = 2.2 months

People required (P) = 0.68/2.2 = 0.3 ~ 1 People

Productivity: = 338 =

**Literature Overview**

**2. Background study and literature overview**

**a. Literature overview:**

* Image processing technique proposed by Sangamithraa&Govindraju with a back propagation neural network algorithm yields the highest outcomes using an LIDC (Lung Image Database Consortium ) dataset with an accuracy of 90.65%.

* Vijaya, Suhasinni and Selvi proposed an image processing technique which produces an accuracy of 72.5%. They had used four core level where enhancement, segmentation, pixel value matching and feature extraction which became very lengthy and leads to less accuracy.

* In this paper (kaucha 2017), median filter and wiener filter is used in the preprocessing stage to enhance the quality of the image and increase the accuracy up to 95%. Different combinations of image processing techniques and algorithms have been proposed with different outcomes in terms of accuracy, sensitivity and specificity.

**Requirement Analysis**

**3. Requirement Analysis**

**Input:** CT scan images (.jpeg)

**Output:** Classification

1. Binary classification – Malignant or Benign

**Functional Requirement:**

1. **Preprocessing of image**

For the first stage of image processing the lung CT scan images are input from the database set. The database set contains lung CT images which are in the DICOM image format. The lung Ct scans are first converted to a grey scale image so that they can be easily processed using a digital processing technique and the quality of the image is improved by removing irregularities and noise present in the image.

1. **Segmentation of image**

For segmentation stage, the ROI is selected after background edge removal to enable segmentation processing. The Discrete Waveform Transform (DWT) technique is used in the ROI image. The multi resolution analysis is carried out by separating low and high level frequency portions of the image using filters. A 2 level DWT technique was applied to compress the extracted ROI image. The ROI based image is first decomposed into four sub bands LL, LH, HL and HH in the 1-level DWT technique. Further, the LL sub band is again decomposed into four additional sub bands using the 2-level DWT technique. The LL component has maximum information content and the other band contain edge in vertical, horizontal and diagonal directions

1. **Feature Extraction from image**

The GLCM (Gray Level Co-occurrence matrix) characterizes the texture of an image by calculating the value of pairs of pixels with specific value and the spatial relationship and extracts statistical measures from this matrix. It considers relation between the reference point and neighboring pixel. The GLCM has been used for the feature extraction. Feature such as entropy, correlation, energy, variance, contrast and dissimilarities are extracted from the sampled image obtained by means of the 2-level DWT technique for the purpose of the classification.

1. **Classification**  
   Classification is carried out using the SVM (Support Vector Machine), classifying whether the image is normal or tumor. Boyle (2011) identified the SVM as a classifier by separating hyper plane – a machine learning algorithm. For this algorithm, we plot data items in n dimensional space where n is the number of features with the value of the feature being equal to the value of the coordinate and then we perform classification by finding the hyper plane. They use optimum linear separating hyper planes which can be used for classification and regression

Software and Hardware Requirement

**Hardware Requirement:**

* Laptop

**Software Requirement:**

* Windows 7 / 8 / 10.
* Anaconda Navigator
* Jupyter Notebook
* OpenCV
* Numpy

Performance Requirements:

1. The number of simultaneous users to be supported: single user
2. Amount and type of information to be handled: We are taking (.jpeg) CT scan images of lung as an input to our system.

Software System Attributes

**1. Reliability**

The proposed software system is reliable and performs failure-free operation in a specified environment.

**2. Availability**

The proposed software system will work as required when required during the period of a mission. The software system is available most of the time.

**3. Security**

This should specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure.

**4. Maintainability**

The proposed software system is maintainable and capable of performing successful repair action in a stipulated time. The system can be restored to operational status after a failure occurs.

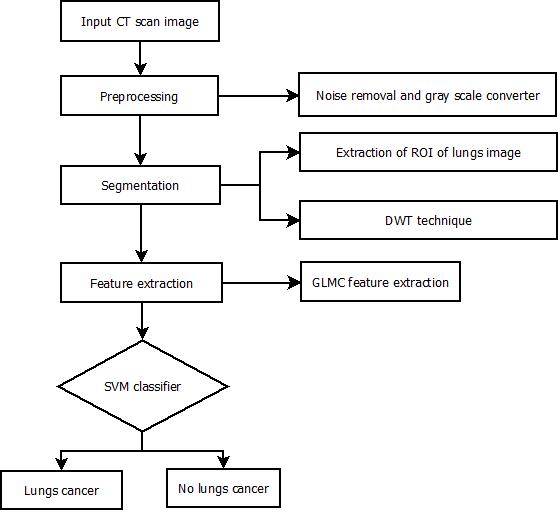
**5. Portability**

The proposed software system is portable and can be used in other environment other than the one in which it is created without requiring major network.

**System Design**

**4. System design**

**a. Architectural Design**



**b. Algorithmic Description of each Module**

**I. Pre-processing:**

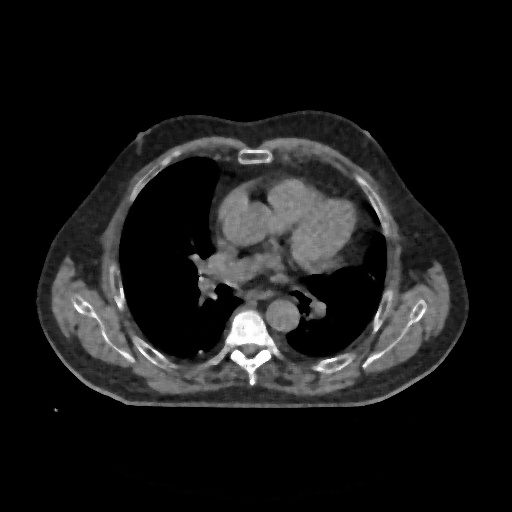
Preprocessing consists of applying two filters

1. Median filter
2. Wiener filter

**Median Filter**

1. Read the image.
2. Apply function cv2.medianBlur().
3. Save the image to the disk.

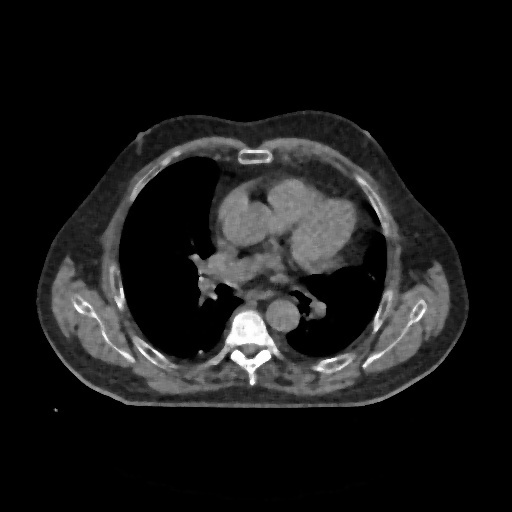
Output:



**Wiener Filter**

1. Read the saved image from median filter
2. Convert the image to gray
3. Calculate the PSF of the image using function defmotion\_process
4. Apply the function make\_blurred() and calculate value for blurred.
5. Calculate result by passing PSF, blurred to wiener () function.
6. Save the output of wiener filter to the disk.

Output:

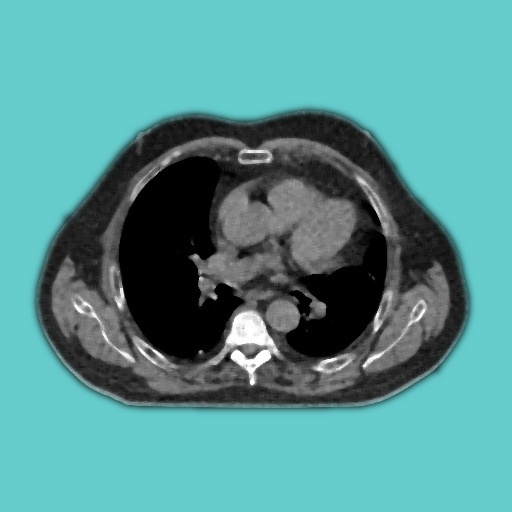


**II. Segmentation:**

Segmentation involves detecting the edge and applying threshold to the image.  
  
**Edge detection**

1. Read the image after wiener filter.
2. Apply Canny’s edge detection algorithm [cv2.Canny()] to calculate the edges
3. Calculate dilation of the edges using [cv2.dilate()]
4. Calculate erosion of the edges using [cv2.erode()]
5. Calculate the contours (i.e. continuous lines) using the extracted edges.
6. Calculate the masked image using the detected edges using [MASK\_COLOR = (0.8, 0.8, 0.4) # In BGR format] and save the image to the disk.

Output:



**Thresholding**

1. Read the image from disk saved from the previous process.
2. Apply adaptive threshold algorithm [cv2.adaptiveThreshold()]
3. Save the image to the disk.

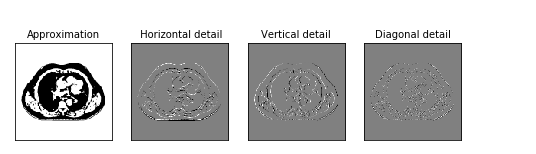
Output:



**Segmentation of image using DWT-2**

1. Read the saved image
2. Calculate the DWT-1 of the thresholded image using pywt.dwt2() and save the LL band .
3. Calculate DWT-2 by passing the LL band again to the function pywt.dwt2() which will output the LL1 band.
4. Save the LL1 band to the disk.

Output:



**III. Feature Extraction:**

1. Open thethe LL1 band saved to the disk.
2. Calculate the GLCM matrix for 0**°,**45**°,**90**°,**135**°** using the function greycomatrix().
3. Calculate the feature values using the function greycoprops() and store it in a file

Eg.dissimilarity = greycoprops(GLCM,’dissimilarity’)

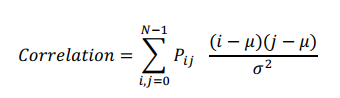
**Texture features**

1. Texture features computed mathematically, which are not evident to human eye and not easily extracted visually. Gray level co-occurrence matrix (GLCM) is used to compute texture

Feature; it is a statistical method that considers the spatial relationship of pixels in the gray-level co-occurrence matrix. Each element (I, J) in the resultant GLCM is simply the sum of the number of times that the pixel with value I occurred in the specified spatial relationship to a pixel with value J in the input image. Following are some examples of GLCM features: correlation, Homogeneity, variance, energy, contrast, entropy.

**Correlation :** Correlation returns a measure of how correlated a pixel is to its neighbor over the whole image. The range of correlation is [-1 1]. Correlation is 1 or -1 for a perfectly positively or negatively correlated image

Given an image composed of pixels each with an intensity (a specific gray level), the GLCM is a tabulation of how often different combinations of gray levels co-occur in an image or image section. Texture feature calculations use the contents of the GLCM to give a measure of the variation in intensity at the pixel of interest

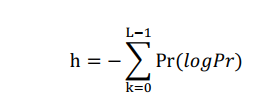


Where,

Pij = Element i,j of GLCM image

N= Number of gray levels in the image

**Entropy**: Entropy is a measure of information content. It measures the randomness of   
 intensity distribution. Entropy; h can also be used to describe the distribution variation in a   
 region. Entropy h can be calculated as:



**Homogeneity**: Homogeneity returns a value that measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal. The range of Homogeneity is [0 1]. Homogeneity is 1 for a diagonal GLCM. The Homogeneity is evaluated using the equation



**Energy:** Energy is defined based on a normalized histogram of the image. Energy shows how the gray levels are distributed. When the number of gray levels is low then energy is high

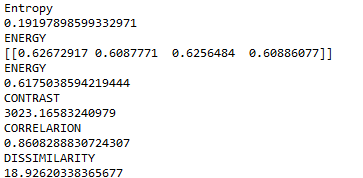
C:\Users\Sir\AppData\Local\Microsoft\Windows\INetCache\Content.Word\enrgy.png

**Contrast:**

A measure of the image contrast or the amount of local variations present in an image.



Here is some of the feature values computed from feature extraction module.

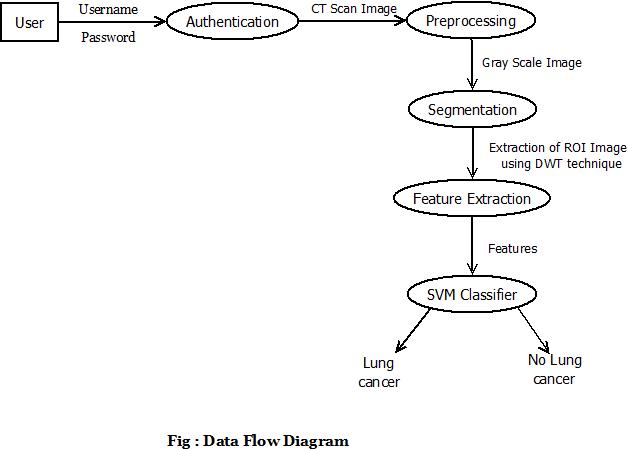


**IV. Classification:**

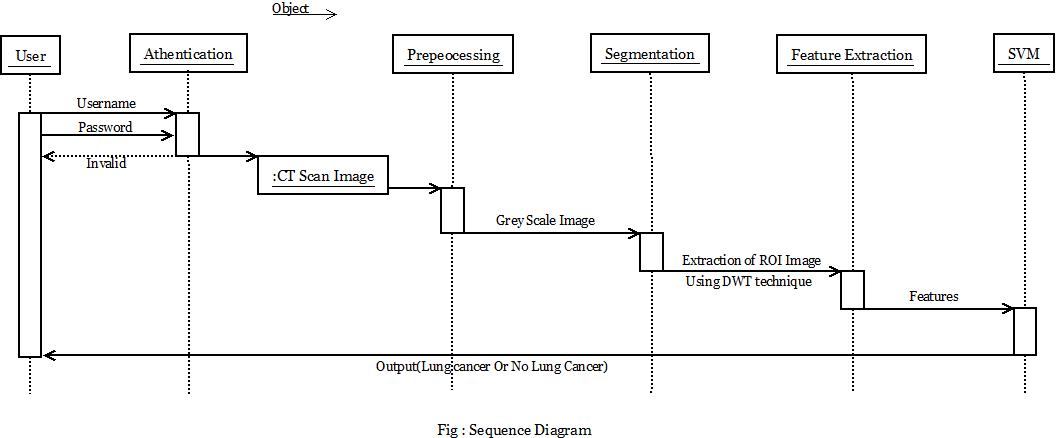
1. Pass the feature values to the clf.predict() function.
2. Check the output result by the function.

**c. System Modeling**

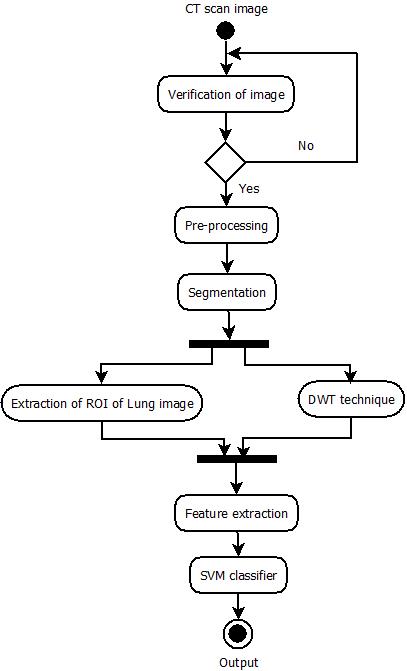
**1. Data Flow Diagram:**

****

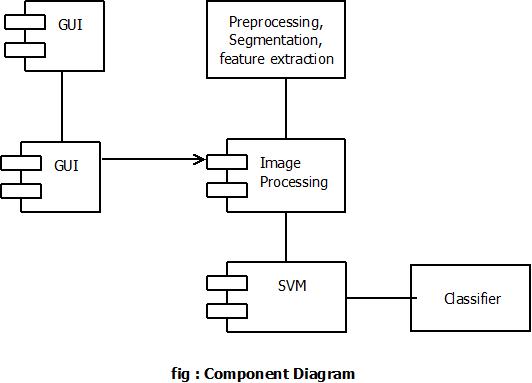
**2. Sequence Diagram :**

****

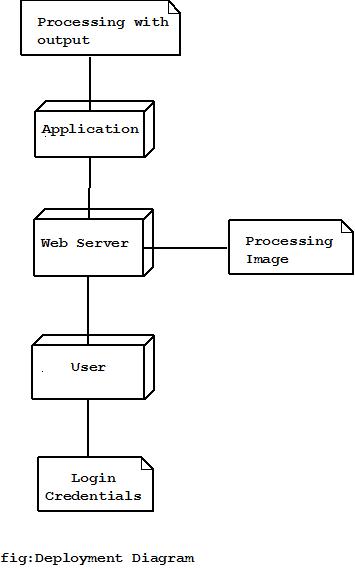
**3. Activity Diagram :**

****

**4. Component Diagram :**

****

**5. Deployment Diagram:**

****

**Implementation**

**5 . Implementation**

**Preprocessing the image**

1. To remove the noise

Applying meidan filter

**final = cv2.medianBlur(img, 5)**

The function **cv2.medianBlur()** computes the median of all the pixels under the kernel window and the central pixel is replaced with this median value. This is highly effective in removing salt-and-pepper noise.

1. Reconstructing the image using wiener filter

**PSF = motion\_process(30, data.shape)**

An out-of-focus image recovering algorithm consists of PSF generation

**blurred = numpy.abs(make\_blurred(data, PSF, 1e-3))**

**result = wiener(blurred, PSF, 1e-3)**

A function wiener() synthesizes the simplified Wiener filter Hw

Hw=H/|H|2+1/SNR

**To segment the image**

1. Applying Canny’s edge detection algorithm

**edges = cv2.Canny(img, CANNY\_THRESH\_1, CANNY\_THRESH\_2)**

**edges = cv2.dilate(edges, None)**

**edges = cv2.erode(edges, None)**Canny Edge Detection is a popular edge detection algorithm.In function cv2.Canny().First argument is our input image. Second and third arguments are our minVal and maxVal respectively. Third argument is aperture\_size. It is the size of Sobel kernel used for find image gradients.

1. Apply thresholding

**th = cv2.adaptiveThreshold(grayscaled, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 115, 1)**

If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black).

1. Applying DWT-2

**pywt.dwt2(original, 'db1')**

The function segments the image into four subbands LL,LH,HL,HH.

**coeffs2 = pywt.dwt2(LL, 'db1')**

The LL band is again passed to the same function where the LL subband is again segmented into four subbands LL1,LH1,HL1,HH1. Where the LL1 band contains the more information.

**Extracting the features**

1. Construct the GLCM matrix(greycomatrix)

**GLCM = GLCM = greycomatrix(im,[1],[0,np.pi/4,np.pi/2,3\*np.pi/4],levels=256,symmetric=False,normed=True)im,[1],[0,np.pi/4,np.pi/2,3\*np.pi/4],levels=256,symmetric=False,normed=True)**

Calculate the grey-level co-occurrence matrix. A grey level co-occurence matrix is a histogram of co-occuringgreyscale values at a given offset over an image.

For example :

**>>>**image=np.array([[0, 0, 1, 1],

**...**  [0, 0, 1, 1],

**...**  [0, 2, 2, 2],

**...**  [2, 2, 3, 3]], dtype=np.uint8)

**>>>**result=greycomatrix(image, [1], [0, np.pi/2], levels=4)

**>>>**result[:, :, 0, 0]

array([[2, 2, 1, 0],

[0, 2, 0, 0],

[0, 0, 3, 1],

[0, 0, 0, 1]], dtype=uint32)

**>>>**result[:, :, 0, 1]

array([[3, 0, 2, 0],

[0, 2, 2, 0],

[0, 0, 1, 2],

[0, 0, 0, 0]], dtype=uint32)

Extract the features from the matrix.

**cont = greycoprops(GLCM,'contrast')**

**coor = greycoprops(GLCM,'correlation')**

**diss = greycoprops(GLCM,'dissimilarity')**

**eng = greycoprops(GLCM, 'energy')**

**homo = greycoprops(GLCM, 'homogeneity')**

Extracts the features from calculated GLCM matrix.

**Classifying using Support Vector machine**

**from sklearn.model\_selection import train\_test\_split   
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)**

**from sklearn.svm import SVC**

**this\_C = 1.0  
 clf = SVC(kernel = 'linear', C=this\_C).fit(X\_train, y\_train)**

**clf.predict(a)**

The above code predicts weather the test image will be Malignant or benign and displays the corresponding array label.

**Integration and Testing**

**6. Testing performed:**

**Test Plan:**

1. Time:

This project should be used to predict that given input is cancerous or not. Different methods have been used in this project, so every module has different time constraint. The time has been measured in seconds with whom output has been shown on the display. Different times for different modules have been given in following table.

|  |  |
| --- | --- |
| **Methods** | **Time Required** |
| Preprocessing | 0.12 seconds |
| Segmentation | 0.25 second |
| Feature Extraction | 0.15 seconds |
| SVM | 0.2 seconds |

1. Output
   1. Early detection lung cancer:
2. As we are detecting cancer based on previous results it is necessary to pass the previous results while training the model. We observe that this method gives results that are more accurate

**Performance Analysis**

**7. Performance Requirement:**

The following shows the performance of each method:

|  |  |
| --- | --- |
| **Methods** | **Time Required** |
| Preprocessing | 0.12 seconds |
| Segmentation | 0.25 second |
| Feature Extraction | 0.15 seconds |
| SVM | 0.25 seconds |

As shown in above table, all the modules have different time constraint. Overall performance of system has been increased by many seconds. Also there are different aspects which affect the overall performance of the system which is as follows:

1. Accuracy

2. Sensitivity.

3. Productivity.

All these factors affect overall performance. In this project we got training accuracy upto 93% and testing accuracy of 95%. These aspects increase overall performance of the system.

**Applications**

**8. Applications:**

1. Can be used for detection of lung cancer at early stage.

2.Can be used in medical application as a single module.

**Installation Guide and User  
 Manual**

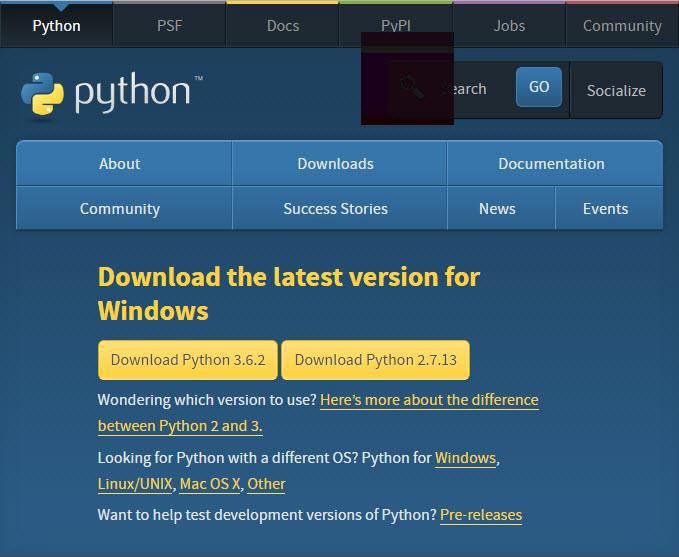
**9. Installation Guide and User Manual**

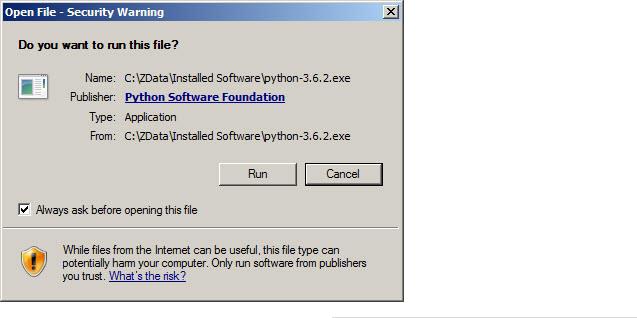
**1.** **Installing Python 3.6:**

Python 3.6 is most compatible version.

Go to: <https://www.python.org/downloads/release/python-360/>

Download the “[Windows x86 executable installer](https://www.python.org/ftp/python/3.6.0/python-3.6.0.exe)” for 32bit windows system.

Download the “[Windows x86-64 web-based installer](https://www.python.org/ftp/python/3.6.0/python-3.6.0-amd64-webinstall.exe)” for 64bit windows system

Double click the python icon to run

A **Python 3.6.2 (32-bit) Setup** pop-up window will appear.

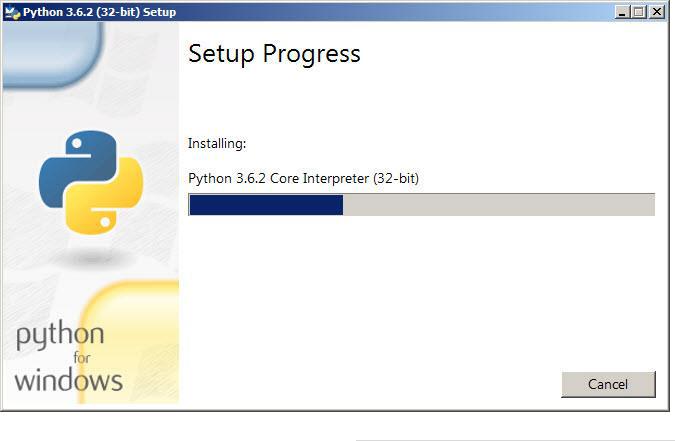
For 64 bit respective screen will be displayed.

Click the check box to add path for python in environment variables.

Click on “Install Now” to install.

****

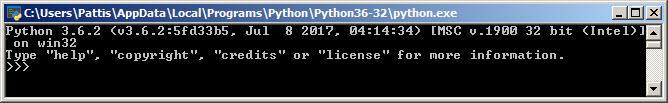
A new **Python 3.6.2 (32-bit) Setup** pop-up window will appear with a **Setup Progress** message and a progress bar

****

To try to verify installation,

1. Navigate to the directory **C:\Users\Pattis\AppData\Local\Programs\Python\Python36-**

**32** (or to whatever directory Python was installed: see the pop-up window for installing step 3).

1. Double-click the icon/file **python.exe**. The following pop-up window will appear.

2.**Installing Anaconda navigator**

1.[Download the Anaconda installer](https://www.anaconda.com/download/#windows).

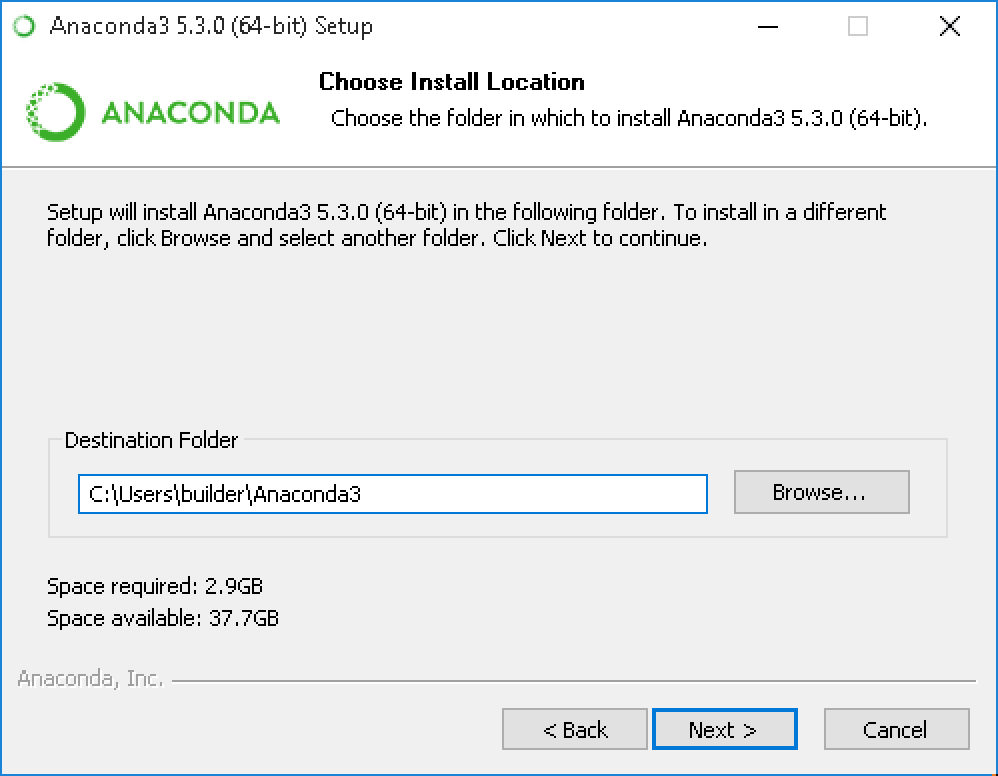
2. Double click the installer to launch.( To prevent permission errors, do not launch the installer from the [Favorites folder](https://docs.anaconda.com/anaconda/user-guide/troubleshooting/#distro-troubleshooting-favorites-folder).)

Note: If you encounter issues during installation, temporarily disable your anti-virus software during install, then re-enable it after the installation concludes. If you installed for all users, uninstall Anaconda and re-install it for your user only and try again.

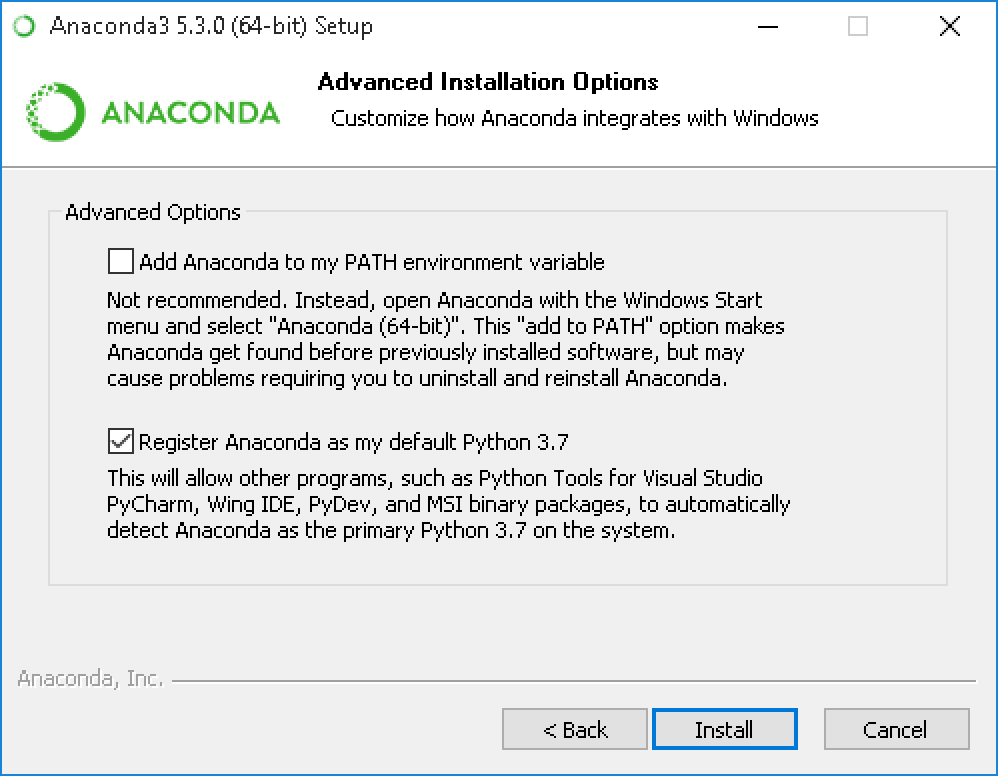
3. Click Next.

4. Read the licensing terms and click “I Agree”

5. Select an install for “Just Me” unless you’re installing for all users (which requires Windows Administrator privileges) and click Next

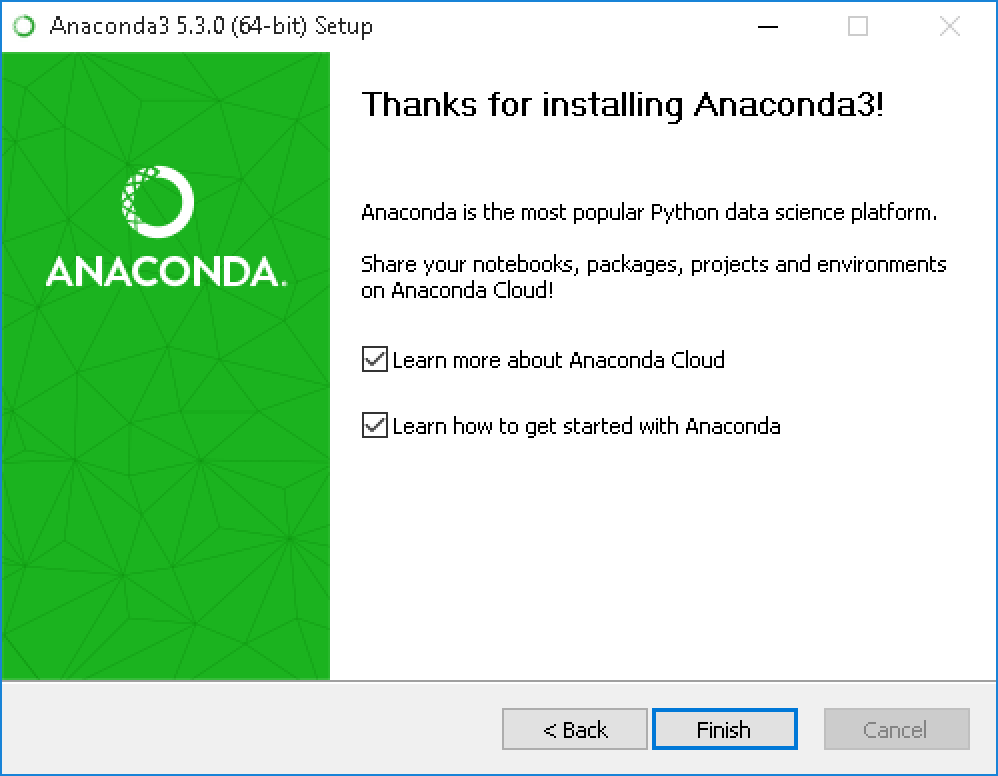


6. Select a destination folder to install Anaconda and click the Next button.  
7. Choose whether to add Anaconda to your PATH environment variable. We recommend not adding   
Anaconda to the PATH environment variable, since this can interfere with other software. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.

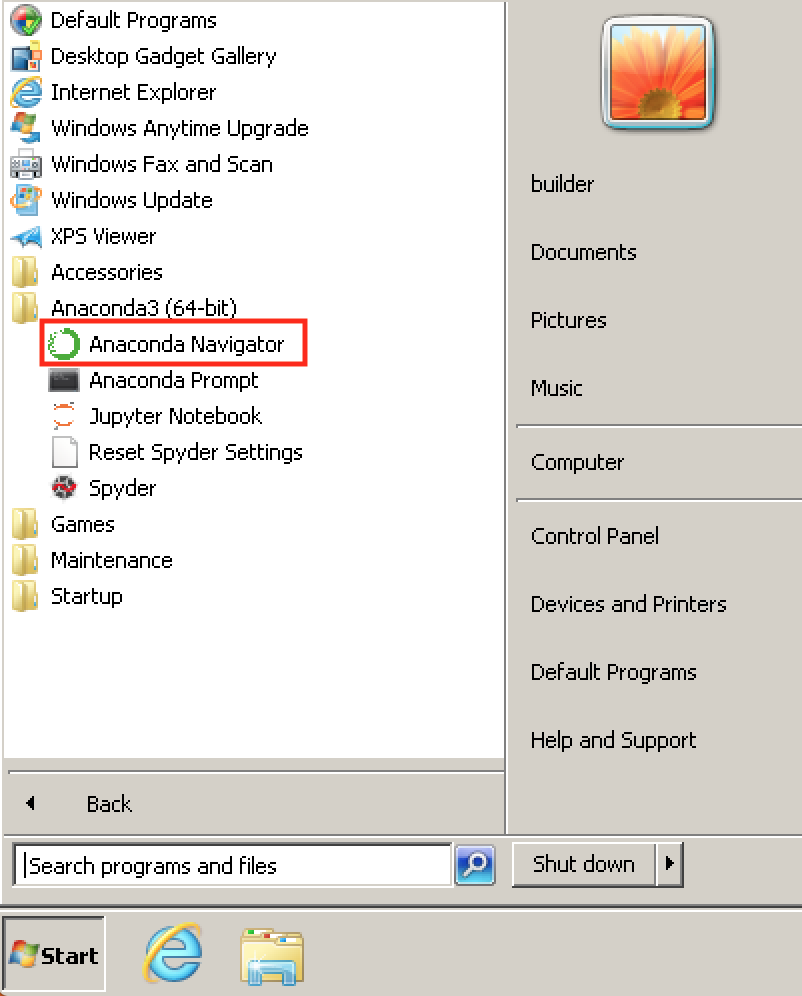


8. Choose whether to register Anaconda as your default Python. Unless you plan on installing and running multiple versions of Anaconda, or multiple versions of Python, accept the default and leave this box checked.

9. Click the Install button. If you want to watch the packages Anaconda is installing. Then click Next button.

10. After a successful installation you will see the “Thanks for installing Anaconda” dialog box  
  
 

11. After your install is complete, verify it by opening Anaconda Navigator, a program that is included with Anaconda: from your Windows Start menu, select the shortcut Anaconda Navigator. If Navigator opens, you have successfully installed Anaconda. If not, check that you completed each step above, then see our Help page.



**3.Installing dependencies required for this project**

* OpenCV  
  To install this package open Anaconda Prompt and type  
  **conda install -c conda-forge opencv**
* **Numpy**To install this package open Anaconda Prompt and type  
  **conda install -c anaconda numpy**
* **Matplotlib**To install this package open Anaconda Prompt and type  
  **conda install -c conda-forge matplotlib**
* **PIL**To install this package open Anaconda Prompt and type

**conda install -c anaconda pil**

* Scikit-imageTo install this package open Anaconda Prompt and type **pip install scikit-image**

**Ethics**

**Declaration of Ethics:**

As a computer Science & Engineering Student, I believe it is Unethical To,

1. Surf the internet for personal interest and non-class related purposes during classes.
2. Make a copy of software for personal or commercial use.
3. Make a copy of software for friend.
4. Loan CDs of Software to friends.
5. Download pirated software from the internet.
6. Distribute pirated software from the internet.
7. Buy software with a single user license and then install it on multiple Computers.
8. Share a pirated copy of software.
9. Install a pirated copy of software

**References**

**10. References:**

[1] G.Vijaya, A.Suhasini, & D.D.Selvi, “Lungs Tumor Detection Using Pixel Value Matching (PVM)Method,” AENSI, 2015.

[2] P.B. Sangmihara, & S. Govindaraju, “Lung Tumor Detection and classification using EK-Mean Clustering, ” Retreived from <https://ieeexplore.ieee.org/document/7566533>, 2016.

[3] Deep Prakash Koucha, P.W.C Prasad, AbeerAlsadoon “Early Detection of Lung Cancer using SVM Classifier in Binomial Image Processing” retrieved from <https://ieeexplore.iee.org/document/8392305> ,2017.