

KIDNEY DISEASE CLASSIFIER USING CNN BASED DEEP LEARNING METHOD

Submitted in the partial fulfillment for the award of
Bachelor of Engineering Degree in
Electrical and Electronics Engineering

By

ENIKA S (Register No. 40140009)
POORNIMA K (Register No. 40140025)
RENGANAYAGI P (Register No. 40140028)



DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Category-1 University by UGC
Accredited with Grade “A++” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600 119

APRIL - 2024

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ENIKA S** (40140009), **POORNIMA K** (40140025) and **RENGANAYAGI P** (40140028) who carried out the project entitled "**KIDNEY DISEASE CLASSIFIER USING CNN BASED DEEP LEARNING METHOD**" under our supervision from September 2023 to April 2024.



Dr. M. L. BHARATHI

Internal Guide



Dr. S. RADHIKA

Head of the Department

Submitted for Viva voce Examination held on _____

Name:

Signature:



Internal Examiner



External Examiner

DECLARATION

We **ENIKA S** (40140009), **POORNIMA K** (40140025), and **RENGANAYAGI P** (40140028) hereby declare that the Project Report entitled “**KIDNEY DISEASE CLASSIFIER USING CNN BASED DEEP LEARNING METHOD**” done by us under the guidance of DR. M.L. BHARATHI is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electrical and Electronics Engineering.

1. 
2. 
- 3.

SIGNATURE OF THE CANDIDATES

DATE:

PLACE: CHENNAI

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITA, Dean, School of Electrical and Electronics Engineering** and **Dr. S. RADHIKA, Head of the Department, Dept. of Electrical and Electronics Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **DR. M. L. BHARATHI** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of Electrical and Electronics Engineering who were helpful in many ways for the completion of the project.

.

ABSTRACT

Kidney abnormalities constitute a significant healthcare challenge globally, impacting millions of individuals. Diagnostic imaging, particularly computed tomography (CT), plays a pivotal role in identifying and characterizing renal anomalies. However, the interpretation of CT images for kidney pathology remains intricate, often necessitating advanced analytical tools to aid clinicians. In this study, we present a novel approach to enhance the accuracy and interpretability of kidney abnormality detection using deep learning techniques. Specifically, we introduce a customized convolutional neural network (CNN) architecture designed to identify common renal abnormalities, including cysts, stones, and tumors, from CT scans. Unlike conventional CNN models, our framework prioritizes simplicity and interpretability, facilitating clearer insights into diagnostic decisions. Through extensive experimentation on a diverse dataset, our proposed CNN model demonstrates remarkable performance, achieving an accuracy of 99.52% in classifying kidney abnormalities. This outperforms existing state-of-the-art methods and underscores the efficacy of our approach. Notably, our model's enhanced interpretive capabilities empower clinicians by providing transparent rationales for diagnostic decisions, thereby fostering greater confidence and understanding. Furthermore, to facilitate seamless integration into clinical practice, we deploy the CNN model using the Flask framework, enabling web-based predictions accessible to healthcare professionals. This deployment ensures practical usability and scalability, facilitating widespread adoption and utilization of our diagnostic tool. In conclusion, our study represents a significant advancement in kidney abnormality detection, offering a potent combination of accuracy, interpretability, and practical utility. By leveraging deep learning techniques and prioritizing clinician-centric design principles, we contribute to improving diagnostic capabilities in renal healthcare, ultimately enhancing patient outcomes and quality of care.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	V
	LIST OF FIGURES	X
	LIST OF ABBREVIATIONS	XI
1	INTRODUCTION	
	1.1 GENERAL INTRODUCTION	1
	1.2 OBJECTIVE AND SCOPE	2
2	LITERATURE SURVEY	
	2.1 MAJOR FINDINGS FROM LITERATURE SURVEY	3
3	PROJECT DESCRIPTION	
	3.1 EXISTING SYSTEM	5
	3.2 PROBLEM STATEMENT	6
	3.3 IEEE STANDARDS FOLLOWED	6
	3.4 CONSTRAINTS	7
	3.5 TRADEOFF	7
	3.6 MODULES	8
	3.7 DATA COLLECTION	8
	3.8 PREPROCESSING	9
	3.9 MODEL TRAINING	9

	3.10 MODEL TESTING	10
	3.10.1 ACCURACY	10
	3.10.2 PRECISION	11
	3.11 FLASK FRAMEWORK PREDICTION	11
	3.12 CONFUSION MATRIX	11
	3.13 FLOW DIAGRAM	12
	3.14 SEQUENCE DIAGRAM	13
	3.15 COLLABORATION DIAGRAM	14
	3.16 ACTIVITY DIAGRAM	15
4	REQUIREMENT SPECIFICATION	
	4.1 IMPLEMENTATION	16
5	CODING AND TESTING	
	5.1 CODING	33
	5.2 CODING STANDARDS	33
	5.2.1 NAMING CONVENTIONS	33
	5.2.1.1 CLASS NAMES	34
	5.2.1.2 MEMBER FUNCTIONS	34
	5.2.2 VALUE CONVENTIONS	34
	5.2.3 SCRIPT WRITING	34
	5.2.4 MESSAGE BOX FORMAT	35
	5.3 TEST PROCEDURE	35
	5.3.1 SOFTWARE TESTING	35

5.4 TEST DATA AND OUTPUT	36
5.4.1 UNIT TESTING	36
5.4.2 FUNCTIONAL TESTS	36
5.4.3 INTEGRATION TESTING	36
5.5 TESTING TECHNIQUES	37
5.5.1 TESTING	37
5.5.1.1 WHITE BOX TESTING	38
5.5.1.2 BLACK BOX TESTING	38
5.5.1.3 PROGRAM TESTING	39
5.5.1.4 SECURITY TESTING	39
5.5.1.5 VALIDATION TESTING	40
5.5.1.6 USER ACCEPTANCE TESTING	40
5.6 CODE	41
5.7 SCREENSHOT	58
5.8 MODEL TRAINING PROGRESS	60
5.9 MODEL EVALUTION ON VALIDATION DATA	61
5.10 DISCUSSION	61
5.10.1 TRAINING PROGRESS	61
5.10.2 ACCURACY AND PERFORMANCE	61
5.10.3 PRECISION AND RECALL	61
5.10.4 OVERFITTING	62
5.10.4.1 GENERALIZATION	62
5.10.4.2 APPLICATION	62

CONCLUSION	63
REFERENCES	65

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	Data Collection	9
3.2	Architecture	10
3.3	Accuracy	10
3.4	Confusion Matrix	12
3.5	Flow Diagram	12
3.6	Sequence Diagram	13
3.7	Collaboration Diagram	14
3.8	Activity Diagram	15
5.1	Precision, Recall, Accuracy, Loss	58
5.2	Confusion Matrix Heat Map	58

LIST OF ABBREVIATIONS

CT	Computed Tomography
CNN	Convolutional Neural Network
CKD	Chronic Kidney Disease
ESRD	End-Stage Renal Disease
DN	Diabetic Nephropathy
SVM	Support Vector Machine
API	Application Programming Interface
GUI	Graphical User Interface
MVC	Model-View-Controller
IDE	Integrated Development Environment
VScode	Visual Studio Code
QA	Quality Assurance
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 GENERAL INTRODUCTION

Kidney disease presents a significant health challenge worldwide, impacting millions of individuals and necessitating accurate diagnostic approaches for effective management. Traditionally, clinicians have relied on computed tomography (CT) scans, which provide detailed cross-sectional images of the kidneys. However, interpreting these images accurately can be complex and time-consuming, often leading to challenges in diagnosing kidney abnormalities. To address this issue, recent advancements in deep learning techniques have shown promise in enhancing the accuracy and efficiency of kidney disease diagnosis. By leveraging convolutional neural networks (CNNs), these techniques enable the detection of various kidney abnormalities, including cysts, stones, and tumors, from medical images. Despite these advancements, the interpretability of deep learning models remains a concern, prompting the need for innovative solutions that offer both accuracy and comprehensibility in diagnostic outcomes.

In response to this need, a groundbreaking study proposed a lightweight customized CNN model designed specifically for detecting kidney abnormalities with exceptional accuracy. This novel approach outperforms traditional methods, achieving an impressive accuracy rate of 99.52%. Moreover, the study integrates Flask framework for web page prediction, thereby enhancing accessibility to diagnostic tools for both clinicians and patients. By providing conclusive and understandable results, this innovative model not only improves the interpretive power of diagnostic tools for kidney disease but also holds the potential to significantly enhance patient outcomes. In conclusion, the convergence of deep learning techniques and advanced diagnostic methodologies offers a promising avenue for addressing the challenges posed by kidney disease. Continued research and development in this field are essential for further advancing diagnostic accuracy and accessibility, ultimately improving the quality of care for individuals affected by kidney abnormalities.

1.2 OBJECTIVE AND SCOPE:

The objective of this study is to develop a lightweight customized convolutional neural network (CNN) model for the accurate detection of kidney abnormalities, including cysts, stones, and tumors, in medical images obtained through computed tomography (CT) scans. Additionally, the study aims to enhance the interpretability of deep-learning models applied to kidney disease diagnosis, providing clinicians with conclusive and understandable results. Furthermore, the study aims to improve accessibility to kidney disease diagnosis by deploying a Flask framework for web page prediction, thereby facilitating easier access to diagnostic tools for both clinicians and patients. Development of a lightweight customized CNN model tailored specifically for the detection of kidney abnormalities in CT scan images. Evaluation of the performance of the CNN model in terms of accuracy, sensitivity, and specificity in detecting kidney cysts, stones, and tumors. Enhancement of the interpretability of deep-learning models applied to kidney disease diagnosis, ensuring that the results provided to clinicians are conclusive and understandable. Deployment of a Flask framework for web page prediction, allowing for easy accessibility to the diagnostic tool by clinicians and patients. Potential implications of the study's findings in improving patient outcomes through early and accurate diagnosis of kidney disease, leading to better management and treatment strategies. Exploration of the scalability and applicability of the CNN model to a broader range of kidney abnormalities and medical imaging modalities, thereby contributing to advancements in the field of medical diagnostics.

CHAPTER 2

LITERATURE SURVEY

2.1 MAJOR FINDINGS FROM LITERATURE SURVEY

In a study by Zou et al. , a deep learning approach was used for the detection and classification of kidney tumors in CT images. The proposed method utilized a convolutional neural network (CNN) with a transfer learning strategy to extract features from the input images. The extracted features were then used to classify kidney tumors into three categories: benign, malignant, or cysts. The proposed method achieved a high accuracy of 94.4% in the classification of kidney tumors, demonstrating the potential of deep learning techniques in accurately detecting and classifying kidney tumors. However, some limitations were identified in the study, including the relatively small size of the dataset used for training the model, which may affect the generalizability of the results. Furthermore, the interpretability of the model was not fully explored, and additional studies may be required to improve the model's interpretability.

Chen et al. proposed a deep learning approach for the segmentation of kidney lesions in CT images. The proposed method utilizes a 3D convolutional neural network (CNN) to extract features from the input CT images and generate segmentation maps of kidney lesions. The proposed method achieved a high Dice coefficient of 0.83 in the segmentation of kidney lesions, demonstrating the potential of deep learning techniques in accurately segmenting kidney lesions.

Yu et al. proposed a deep learning approach for the classification of chronic kidney disease (CKD) stages using ultrasound images. The proposed method utilized a convolutional neural network (CNN) to extract features from the input ultrasound images and classify the CKD stages into five categories: normal, mild, moderate, severe, and end-stage. The proposed method achieved an accuracy of 92.5% in the classification of CKD stages, demonstrating the potential of deep learning techniques in accurately classifying CKD stages using ultrasound images.

Kidney disease is a growing concern worldwide, with millions of people suffering from chronic kidney disease (CKD) and end-stage renal disease (ESRD). The

early diagnosis and accurate detection of kidney abnormalities are crucial to prevent further damage and improve patient outcomes. Recently, deep learning has emerged as a promising tool for kidney disease diagnosis, with several studies reporting promising results.

One study used a deep learning algorithm to detect kidney cysts in CT images, achieving an accuracy of 94.5% and demonstrating the potential of deep learning for kidney disease diagnosis. Another study utilized a 3D deep convolutional neural network (CNN) to detect renal tumors in CT images, with a sensitivity of 87.5% and a specificity of 97.9%. The authors noted that their approach outperformed traditional radiological methods, highlighting the potential for deep learning to improve diagnosis and treatment.

In a more recent study, a deep residual network was developed to classify diabetic nephropathy (DN) in renal biopsy images. The model achieved an accuracy of 94.7%, outperforming traditional histopathological methods. The authors noted that their approach could potentially reduce the need for invasive biopsies, leading to better patient outcomes.

Another study used deep learning to detect CKD from retinal images, demonstrating a potential non-invasive screening tool for kidney disease. The authors used a deep CNN to analyze retinal images and achieved an accuracy of 91.8%, demonstrating the potential of deep learning for early detection of kidney disease.

Overall, these studies demonstrate the potential of deep learning for kidney disease diagnosis and treatment. The development of specialized deep learning models for kidney disease diagnosis could potentially lead to more accurate and timely diagnoses, improving patient outcomes and reducing healthcare costs. However, further research is needed to validate these models and ensure their clinical applicability.

CHAPTER 3

PROJECT DESCRIPTION

3.1 EXISTING SYSTEM

In the domain of machine learning and signal processing, a plethora of robust techniques has emerged to address a wide array of challenges encompassing feature extraction, classification, and segmentation. Among these methods, the autoencoder stands out as a neural network architecture tailored for unsupervised learning, adept at capturing latent representations of data through compression into lower-dimensional spaces and subsequent reconstruction, thereby elucidating meaningful features. Another notable approach is the least squares Support Vector Machine (LS-SVM) classifier, merging SVM's robustness with the computational efficiency of least squares techniques, proving particularly effective in nonlinear classification tasks across various domains. Complementing these, signal processing tasks often rely on advanced techniques like stationary wavelet transforms (SWT) and discrete wavelet transforms (DWT), pivotal in decomposing signals into different frequency components, facilitating effective feature extraction and analysis. SWT's translation-invariance suits non-stationary signals, while DWT's multi-resolution analysis enables precise localization of signal features in time and frequency domains. Additionally, SVM-based segmentation methods have gained traction in image processing and medical imaging, harnessing SVMs' discriminative power to accurately delineate objects or regions of interest, crucial for tasks like tumor detection and object recognition. In essence, the amalgamation of autoencoders, LS-SVM classifiers, wavelet transforms, and SVM-based segmentation methods forms a potent toolkit for addressing diverse challenges in machine learning and signal processing, driving innovation and offering robust solutions for data analysis, classification, and segmentation endeavors.

3.2 PROBLEM STATEMENT

Despite advancements in deep learning models for kidney disease classification using CT scan images, there are still open challenges and areas for improvement.

1. Generalization to diverse patient populations: Ensuring the model's performance remains robust across different demographics, disease stages, and imaging protocols.
2. Interpretability and transparency: Developing methods to explain the model's decisions to clinicians, ensuring trust and facilitating integration into clinical practice.
3. Addressing class imbalance: Handling cases where certain abnormalities (e.g., tumors) are less prevalent in the dataset, which may lead to biased model predictions.
4. Incorporating multi-modal information: Integrating additional clinical data (such as patient history, laboratory results) with CT images to improve diagnostic accuracy.
5. Real-time performance: Optimizing the model for real-time inference, especially for applications where quick diagnosis is critical, such as emergency situations.

3.3 IEEE STANDARDS FOLLOWED

1. IEEE 1708-2014 - Standard for Wearable Cuffless Blood Pressure Measuring Devices. This standard specifies the requirements for cuffless blood pressure measuring devices, which could be integrated with deep learning algorithms for health monitoring applications.
2. IEEE 1876-2019 - Standard for Networked Smart Learning Objects for Online Laboratories. This standard provides guidelines for the development and deployment of networked smart learning objects, which could be utilized in educational applications of deep learning in medical imaging.
3. IEEE 1857-2017 - Standard for Advanced Audio and Video Coding. While primarily focused on audio and video coding, this standard includes

provisions for compression techniques that could be relevant for processing and storing medical imaging data used in deep learning models.

4. IEEE 1073-2019 - Standard for Health Informatics - Point-of-Care Medical Device Communication. This standard specifies communication protocols for medical devices, enabling interoperability and connectivity, which could facilitate the integration of deep learning algorithms with medical imaging devices.
5. IEEE 11073-10207-2017 - Health informatics - Point-of-care medical device communication - Part 10207: Domain information model for point-of-care medical device communication. This standard defines a domain information model for point-of-care medical device communication, supporting the exchange of data between devices, systems, and software applications, potentially including those employing deep learning algorithms for medical image analysis.

3.4 CONSTRAINTS

1. Availability of labeled CT scan images for training the CNN model.
2. Computational resources for training and testing the deep learning model.
3. Time constraints for model development, training, and testing.
4. Limitations in the interpretability of the deep learning model's decisions, which may affect clinical acceptance.

3.5 TRADEOFF

1. Model complexity versus interpretability: Balancing the complexity of the CNN model with its interpretability to ensure clinicians can understand and trust the results.
2. Training data size versus model performance: Increasing the size of the training dataset may improve model performance but requires more computational resources and time for training.

3. Deployment complexity: Choosing the appropriate deployment framework (such as Flask) for web-based predictions, considering factors like ease of implementation, scalability, and maintenance.

3.6 MODULES

- Data Collection
- Data Preprocessing
- Model Training
- Model Testing
- Flask Framework Prediction

3.7 DATA COLLECTION

The dataset was collected from PACS (Picture archiving and communication system) from different hospitals in Dhaka, Bangladesh where patients were already diagnosed with having a kidney tumor, cyst, normal or stone findings.

Both the Coronal and Axial cuts were selected from both contrast and non-contrast studies with protocol for the whole abdomen and urogram.

The Dicom study was then carefully selected, one diagnosis at a time, and from those we created a batch of Dicom images of the region of interest for each radiological finding.

Following that, we excluded each patient's information and meta data from the Dicom images and converted the Dicom images to a lossless jpg image format. After the conversion, each image finding was again verified by a radiologist and a medical technologist to reconfirm the correctness of the data.

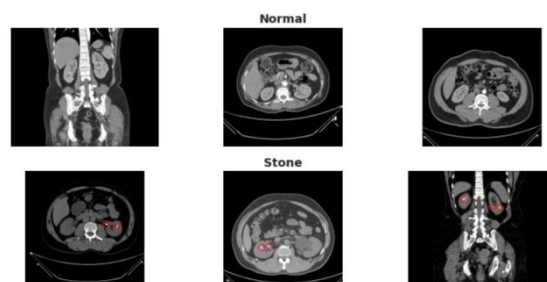


Fig 3.1 Data Collection

The above dataset contains 12,446 unique data within it in which the cyst contains 3,709, normal 5,077, stone 1,377 and tumor 2,283.

3.8 PREPROCESSING

Rescale parameter is set to 1/255, normalize them to the range [0, 1], target_size: A tuple specifying the size of the images after resizing. In this case, the images will be resized to 200x200 pixels, colour_mode: Specifies whether the images should be loaded in color ('rgb') or grayscale ('grayscale') mode. In this case, grayscale mode is used. class_mode: Specifies the type of label encoding to use. 'categorical' is used here, which means that the labels are encoded as one-hot vectors. batch_size: Specifies the number of images in each batch. In this case, each batch will contain 100 images.

3.9 MODEL TRAINING

CNN

Initializes a Sequential model and adds multiple layers to it using the Keras API with the TensorFlow backend.

The model architecture consists of 6 convolutional layers with ReLU activation, followed by max-pooling layers to downsample the feature maps. After the convolutional layers, the feature maps are flattened and passed through a fully connected layer with 512 hidden units and a ReLU activation. Finally, the output layer with 4 neurons and a softmax activation function is added to classify the input images into one of the four classes.

The optimizer used is 'rmsprop', and the loss function used is 'categorical_crossentropy', which is suitable for multi-class classification tasks.

Precision, recall, and accuracy are used as evaluation metrics for the model, and these metrics are defined in the METRICS variable.

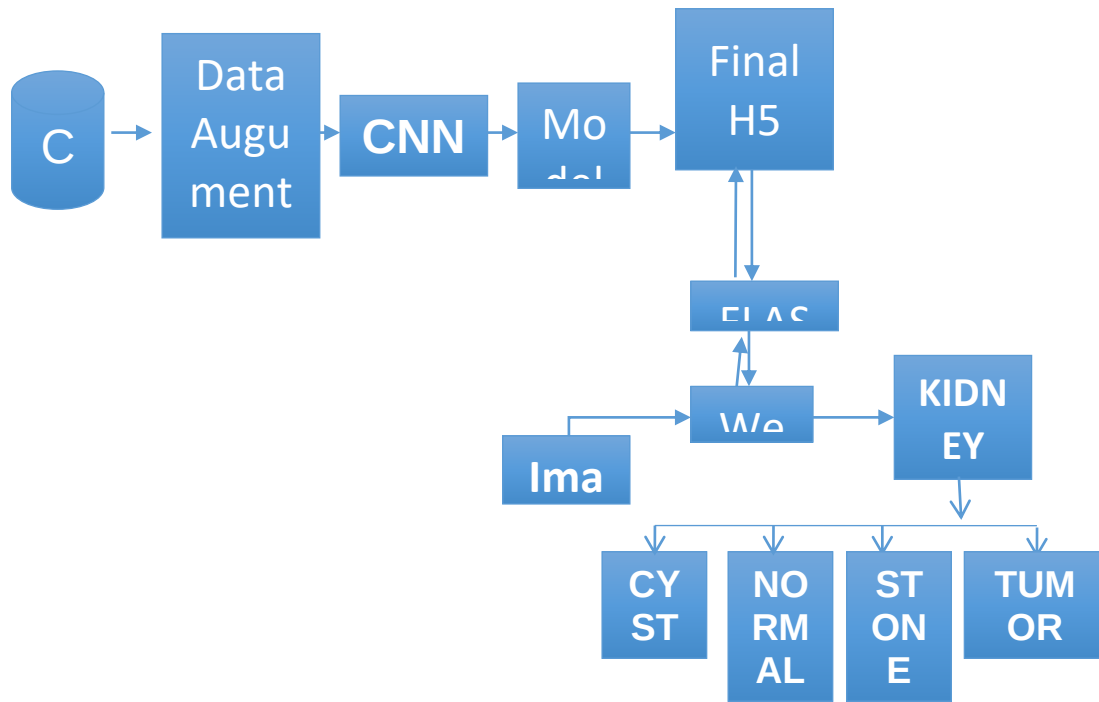


Fig. 3.2 Architecture

3.10 MODEL TESTING

In this module we test the trained machine learning model using the test dataset.

3.10.1 Accuracy

The most commonly used metric to judge a model and is actually not a clear indicator of the performance. The worse happens when classes are imbalanced.

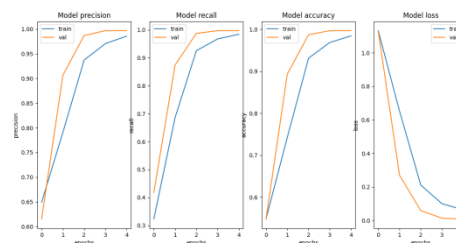


Fig. 3.3 Accuracy

3.10.2 Precision

Percentage of positive instances out of the **total predicted positive** instances.

Logarithmic Loss: Logarithmic Loss or Log Loss, works by penalising the false classifications. It works well for multi-class classification. When working with Log Loss, the classifier must assign probability to each class for all the samples. Suppose, there are N samples belonging to M classes, then the Log Loss is calculated as below :

where, y_{ij} , indicates whether sample i belongs to class j or not, p_{ij} , indicates the probability of sample i belonging to class j.

3.11 FLASK FRAMEWORK PREDICTION

Flask is a lightweight web framework for Python that allows you to build web applications quickly and easily. Flask works by Model-View-Controller (MVC) architecture pattern, where the model represents the data, the view represents the user interface, and the controller acts as an intermediary between the model and the view.

Define routes: Routes are the URLs that the user can visit in your web application. You can define routes in Flask by using the `@app.route()` decorator and specifying the URL pattern as a parameter.

3.12 CONFUSION MATRIX

A confusion matrix is a fundamental tool used in the evaluation of classification models, providing a comprehensive summary of the performance of the model on a given dataset. It is especially useful when assessing the effectiveness of machine learning algorithms in categorizing instances into multiple classes. The matrix presents a tabular representation of the model's predictions compared to the actual class labels in the dataset.

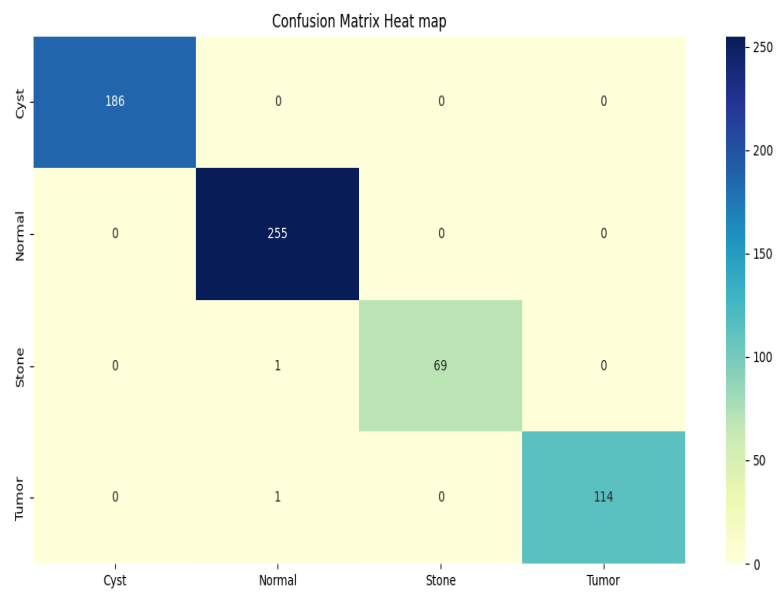


Fig. 3.4 Confusion Matrix

3.13 FLOW DIAGRAM

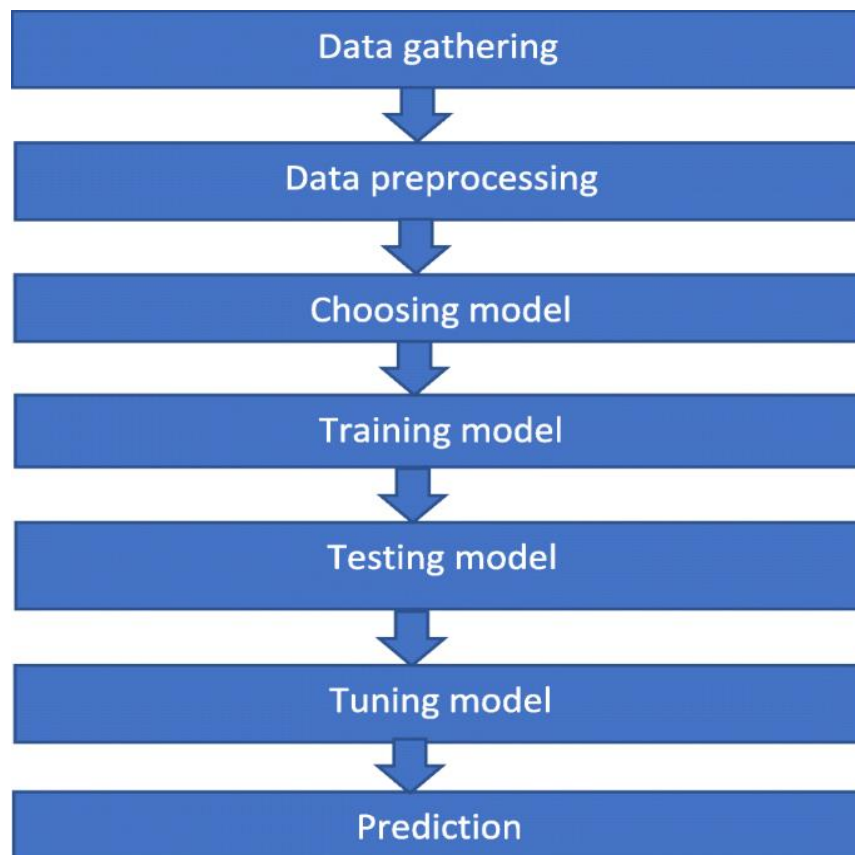


Fig 3.5 Flow Diagram

3.14 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram that illustrates the flow of messages and interactions between objects or components in a system over time. It provides a visual representation of the dynamic behavior of a system, showing how various objects collaborate to achieve a particular functionality.

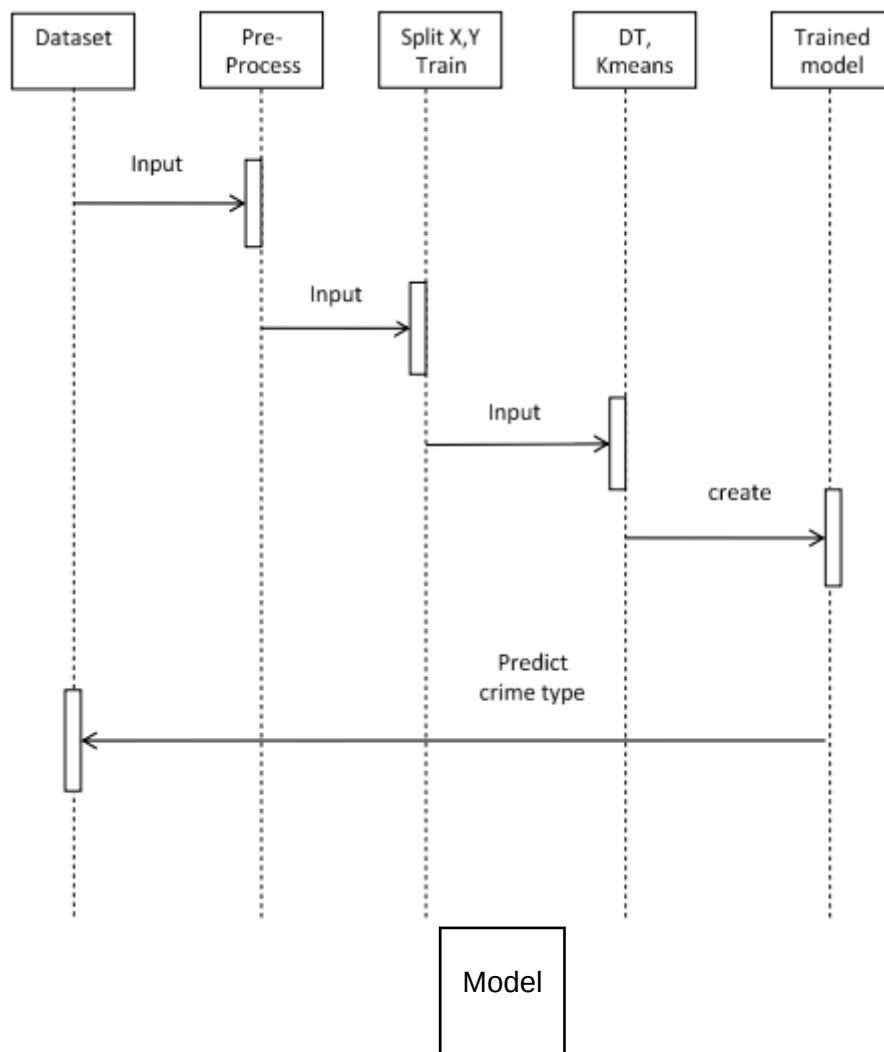


Fig 3.6 Sequence Diagram

3.15 COLLABRATION DIAGRAM

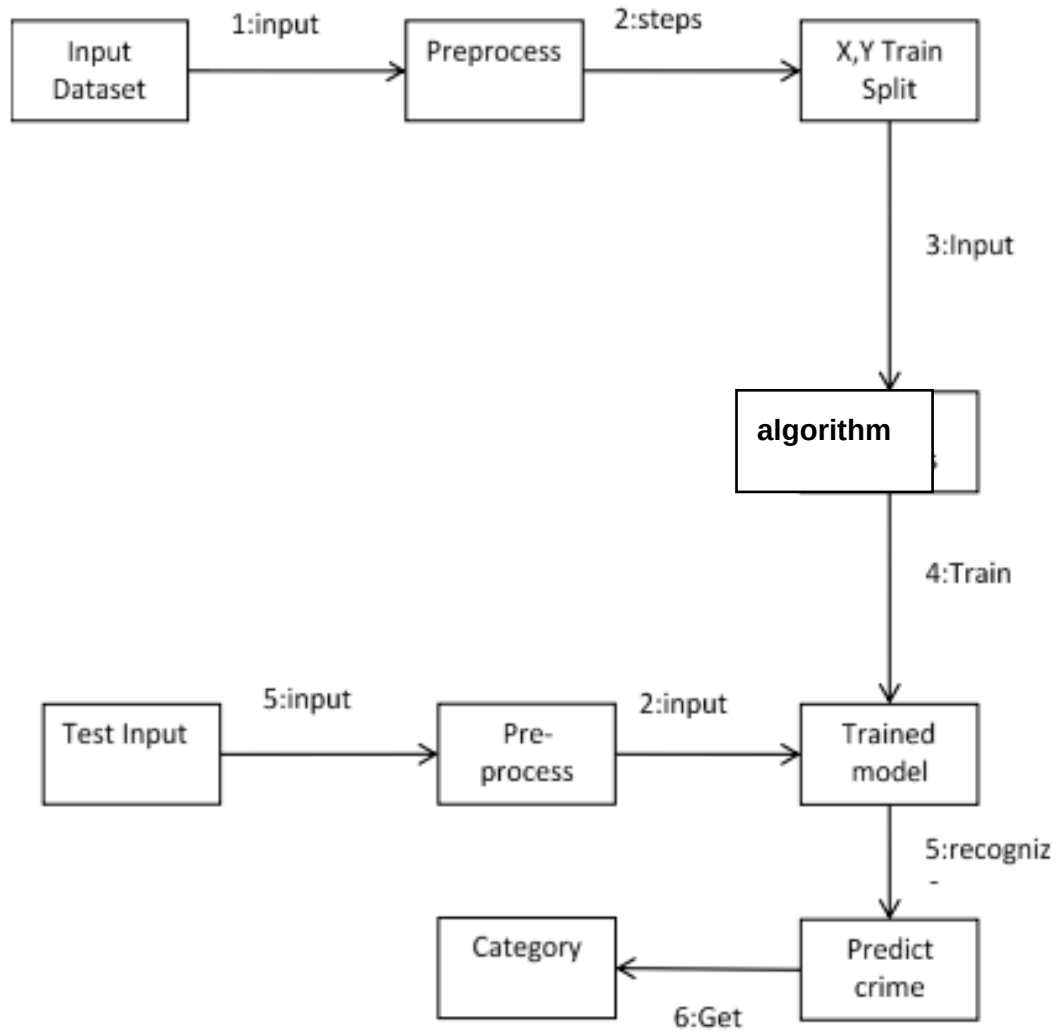


Fig. 3.7 Collabration Diagram

3.16 ACTIVITY DIAGRAM

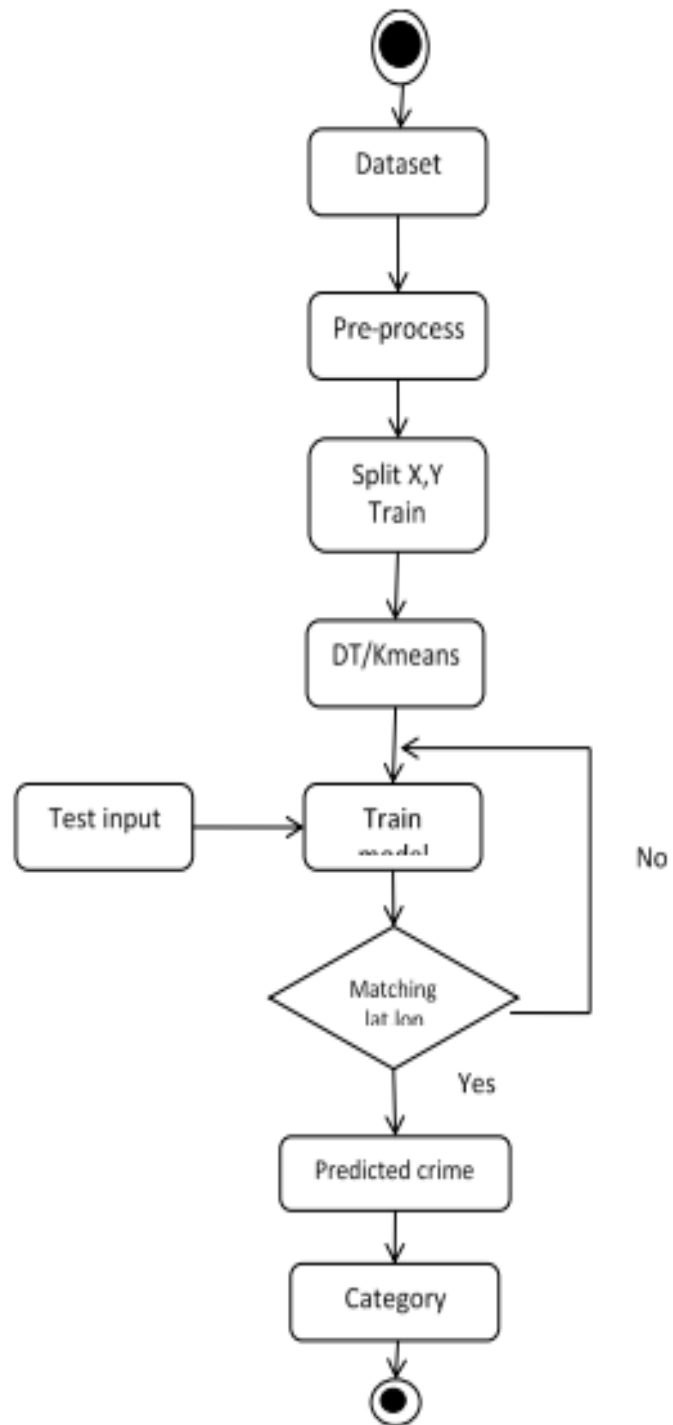


Fig 3.8 Activity Diagram

CHAPTER 4

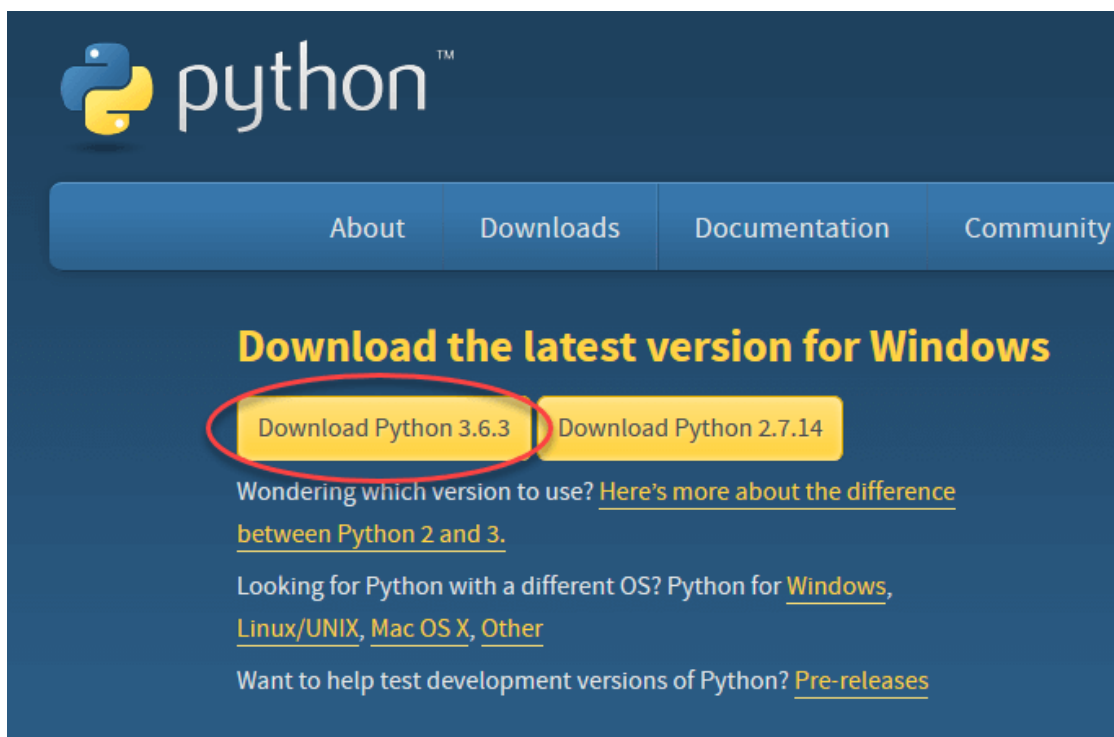
REQUIREMENT SPECIFICATION

4.1 IMPLEMENTATION

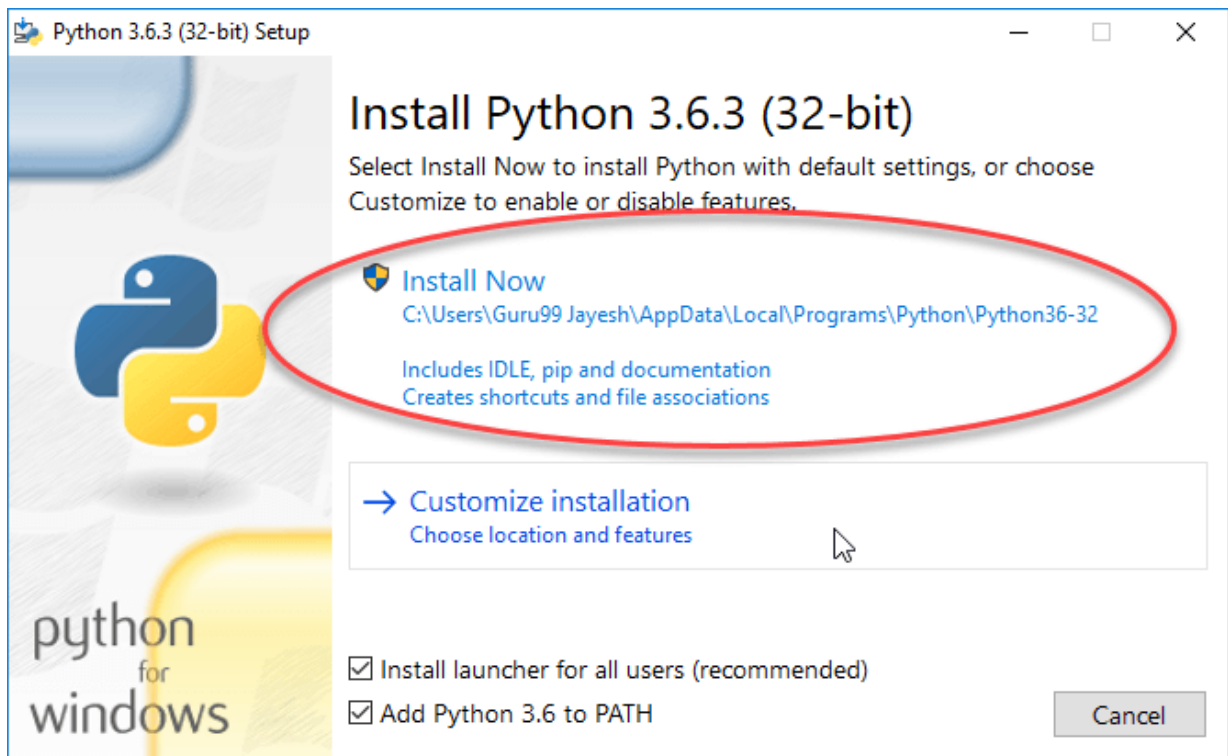
HOW TO INSTALL PYTHON IDE

Below is a step by step process on how to download and install Python on Windows:

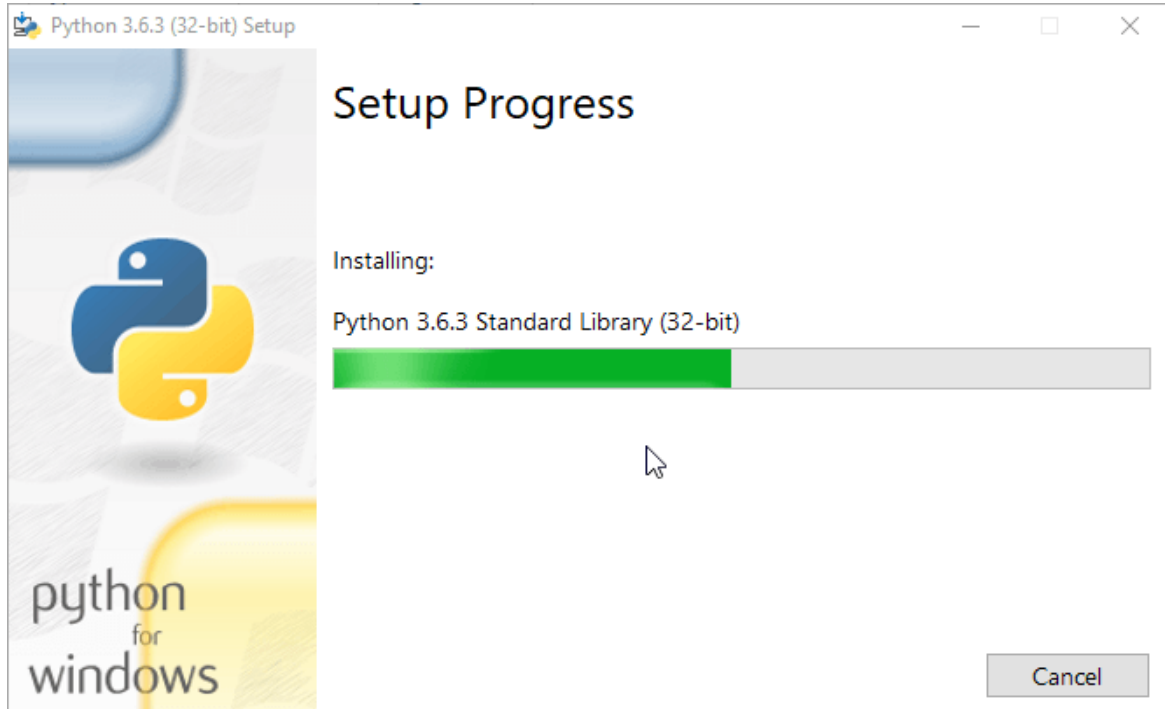
Step 1) To download and install Python, visit the official website of Python <https://www.python.org/downloads/> and choose your version. We have chosen Python version 3.6.3



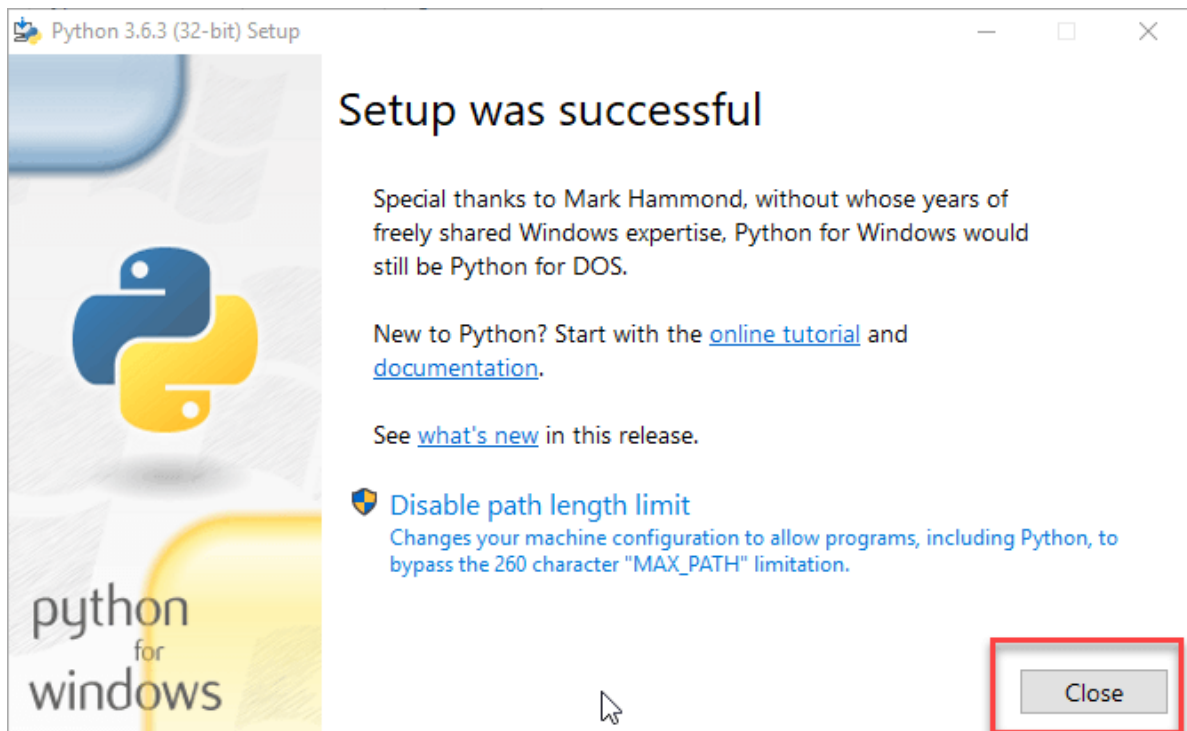
Step 2) Once the download is completed, run the .exe file to install Python. Now click on Install Now.



Step 3) You can see Python installing at this point.



Step 4) When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".



HOW TO INSTALL PYCHARM

Here is a step by step process on how to download and install Pycharm IDE on Windows:

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

Download PyCharm

Windows

macOS

Linux

Professional

Full-featured IDE
for Python & Web
development

DOWNLOAD

Free trial

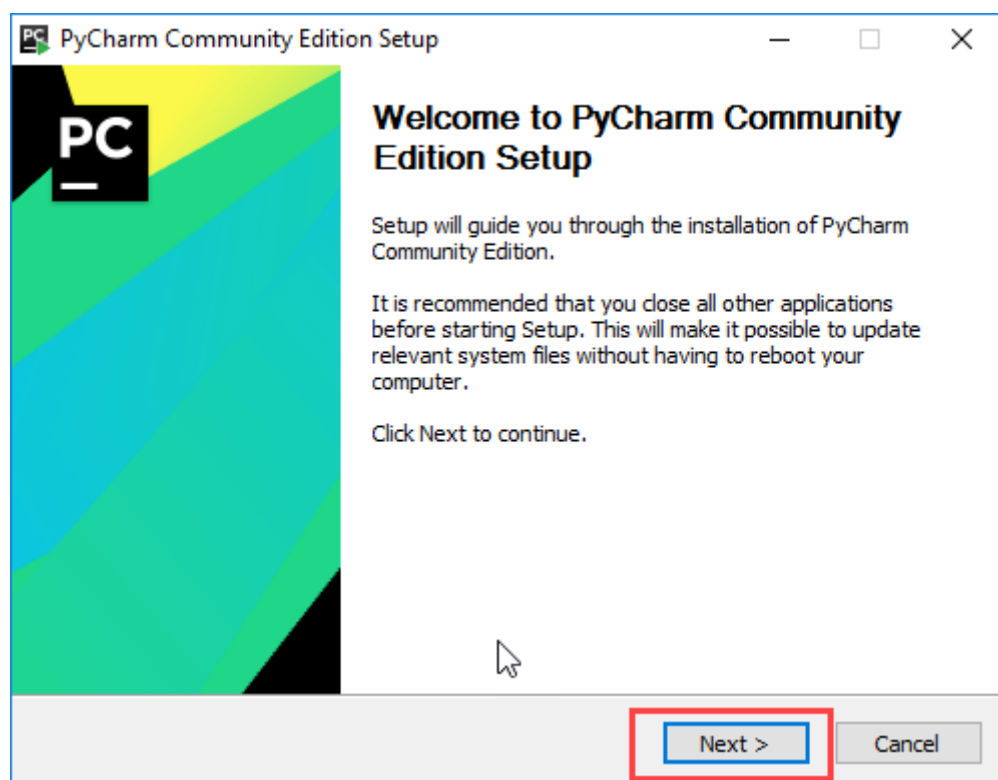
Community

Lightweight IDE
for Python & Scientific
development

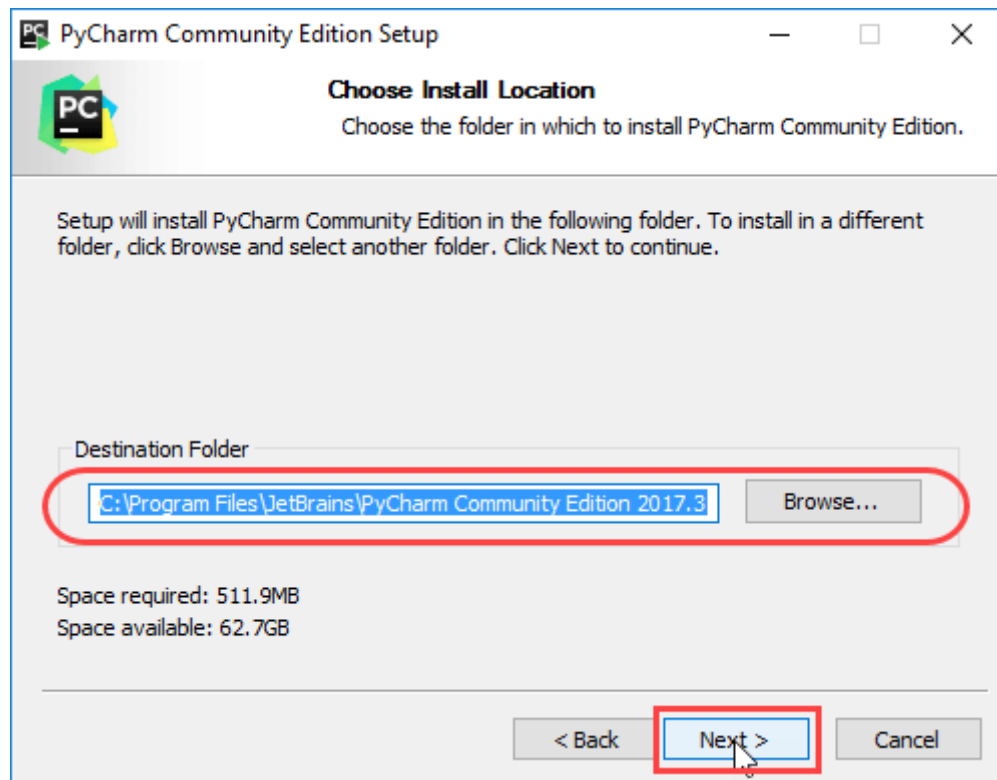
DOWNLOAD

Free, open-source

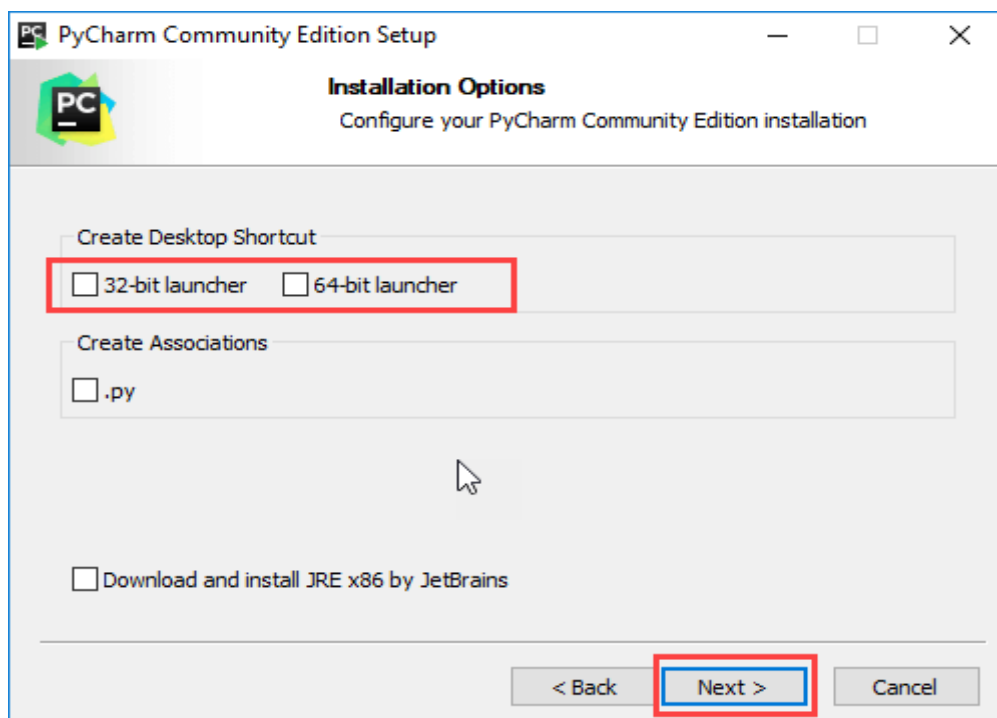
Step 2) Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.



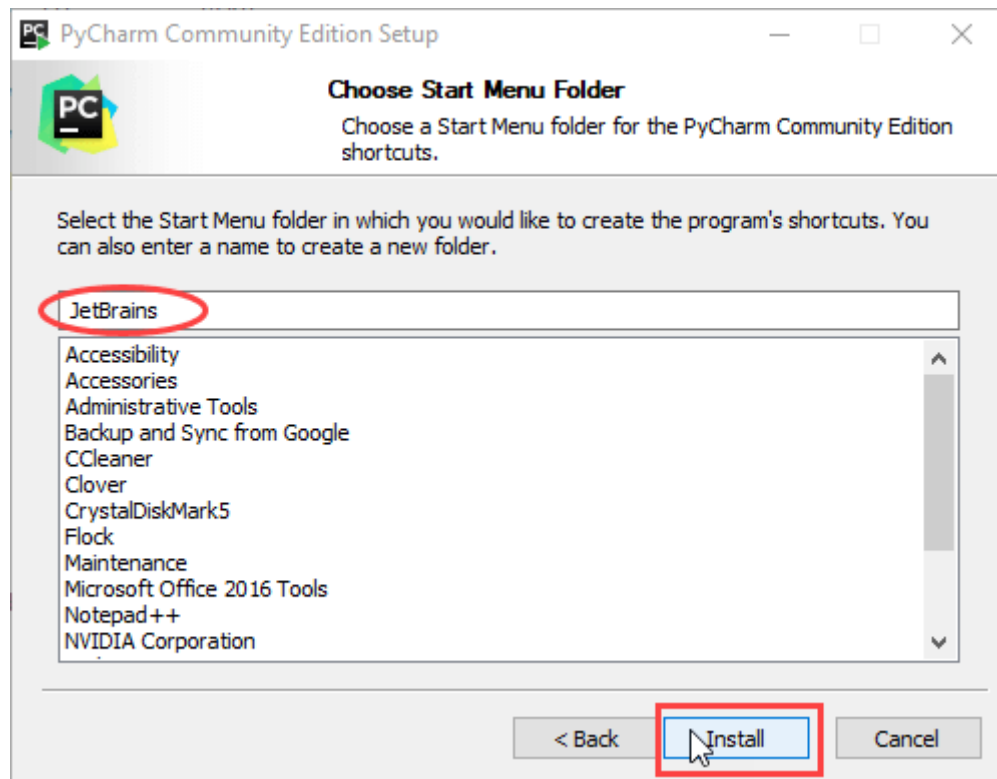
Step 3) On the next screen, Change the installation path if required. Click “Next”.



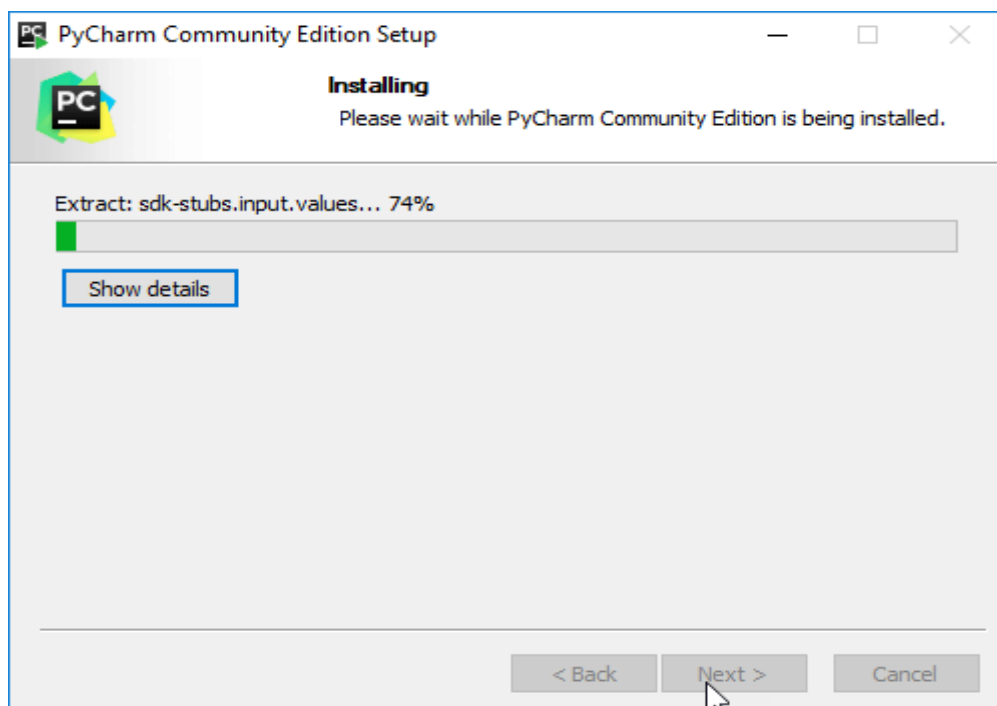
Step 4) On the next screen, you can create a desktop shortcut if you want and click on “Next”.



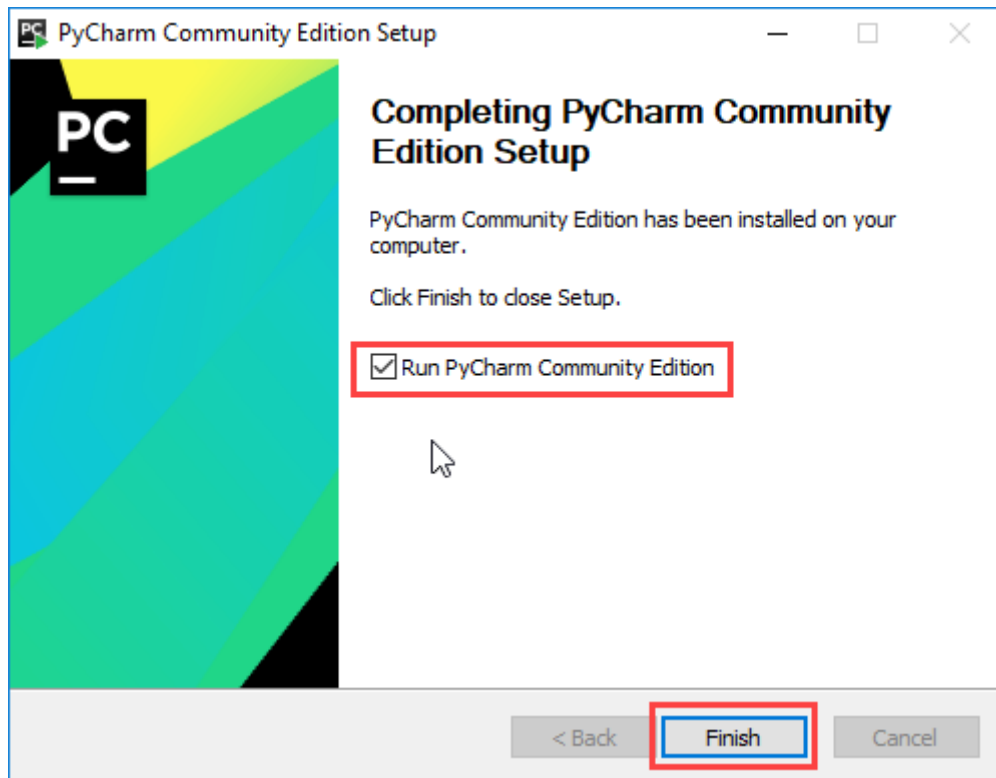
Step 5) Choose the start menu folder. Keep selected JetBrains and click on “Install”.



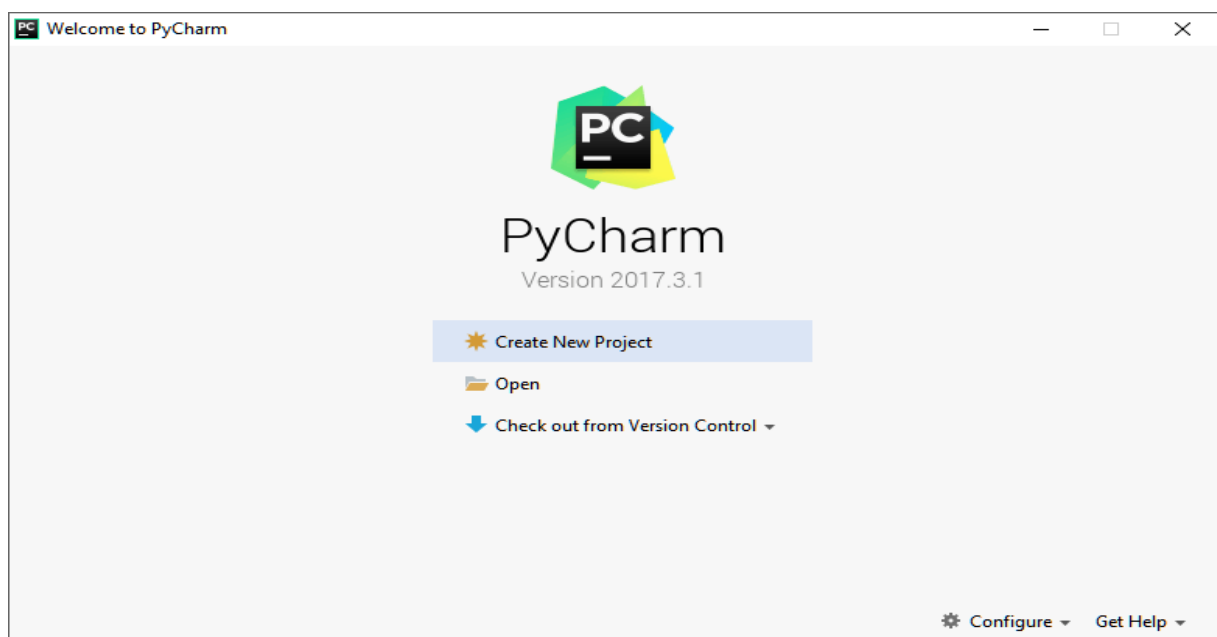
Step 6) Wait for the installation to finish.



Step 7) 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”.

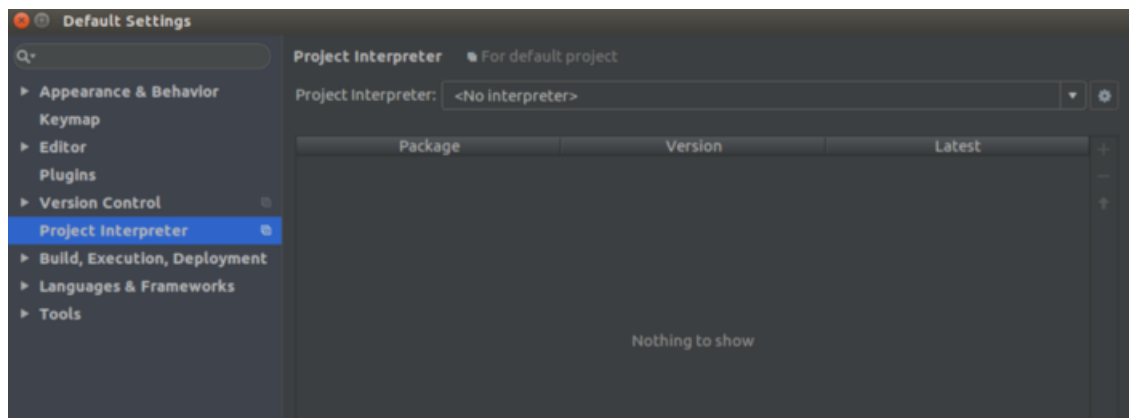


Step 8) After you click on “Finish,” the Following screen will appear.

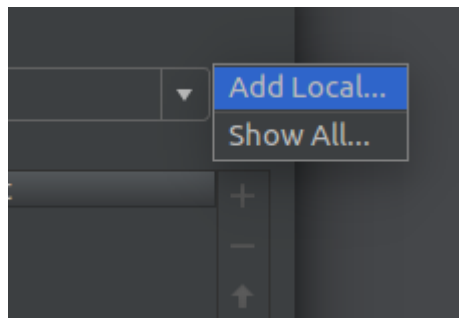


Click on Configure > Settings to open up settings in PyCharm

Search for “Project Interpreter”. My PyCharm looks like this



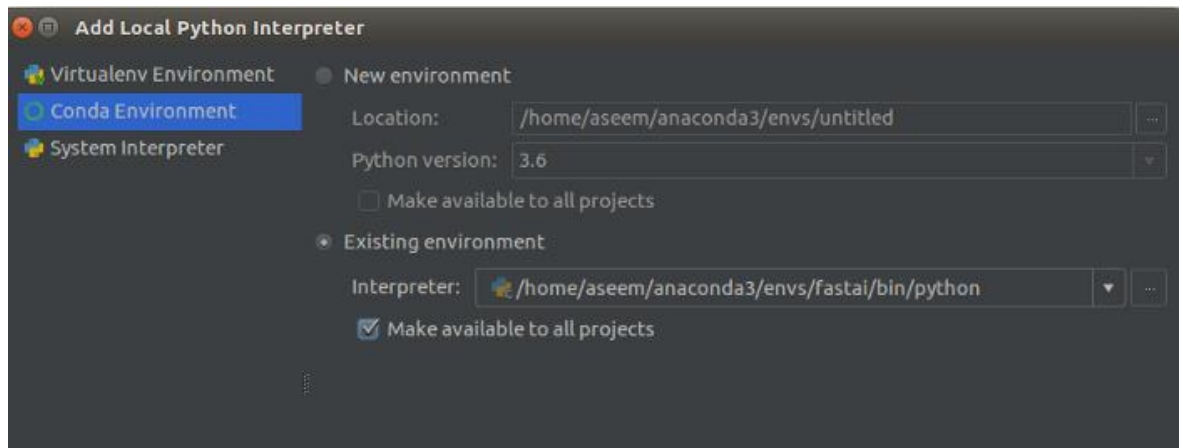
Click on Add local via the settings on the right side



Select “conda environment”

Click on “Existing environment” and navigate to the environment that you want to use. Note that you have to select the bin/python file inside the conda environment for PyCharm to be able to recognise the environment

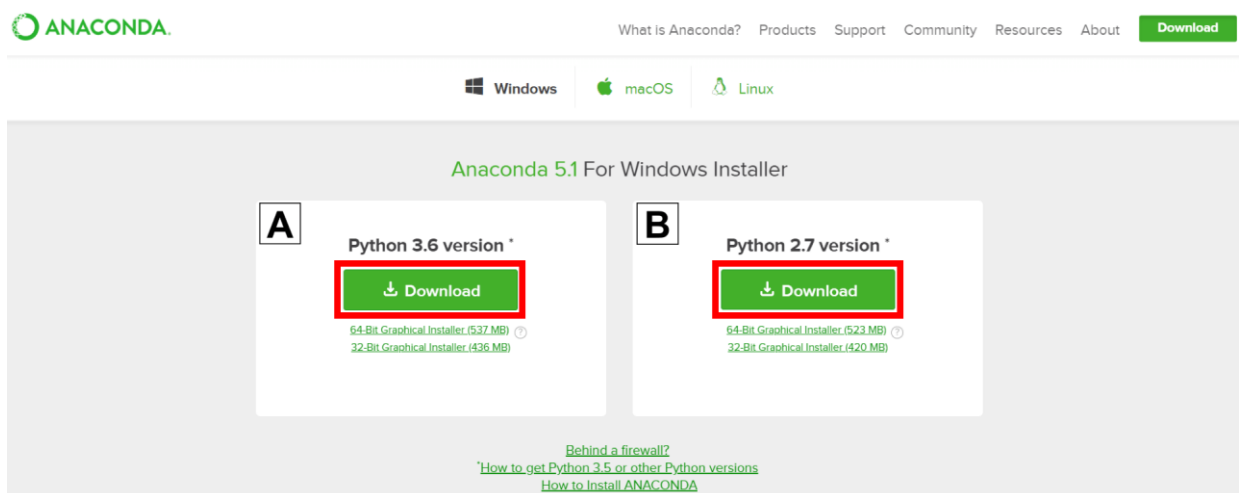
Make sure to click the “Make available to all projects” if you want the interpreter to be used by multiple projects



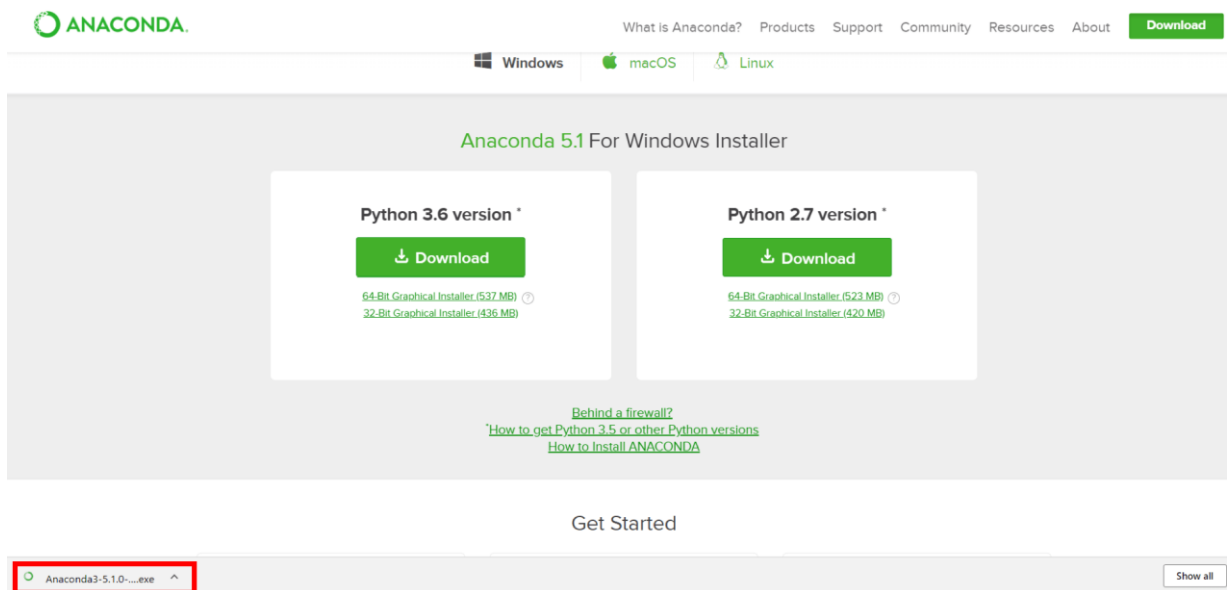
Click ok and you are done

HOW TO DOWNLOAD AND INSTALL ANACONDA

1. Go to the **Anaconda Website** and choose a Python 3.x graphical installer (A) or a Python 2.x graphical installer (B). If you aren't sure which Python version you want to install, choose Python 3. Do not choose both.



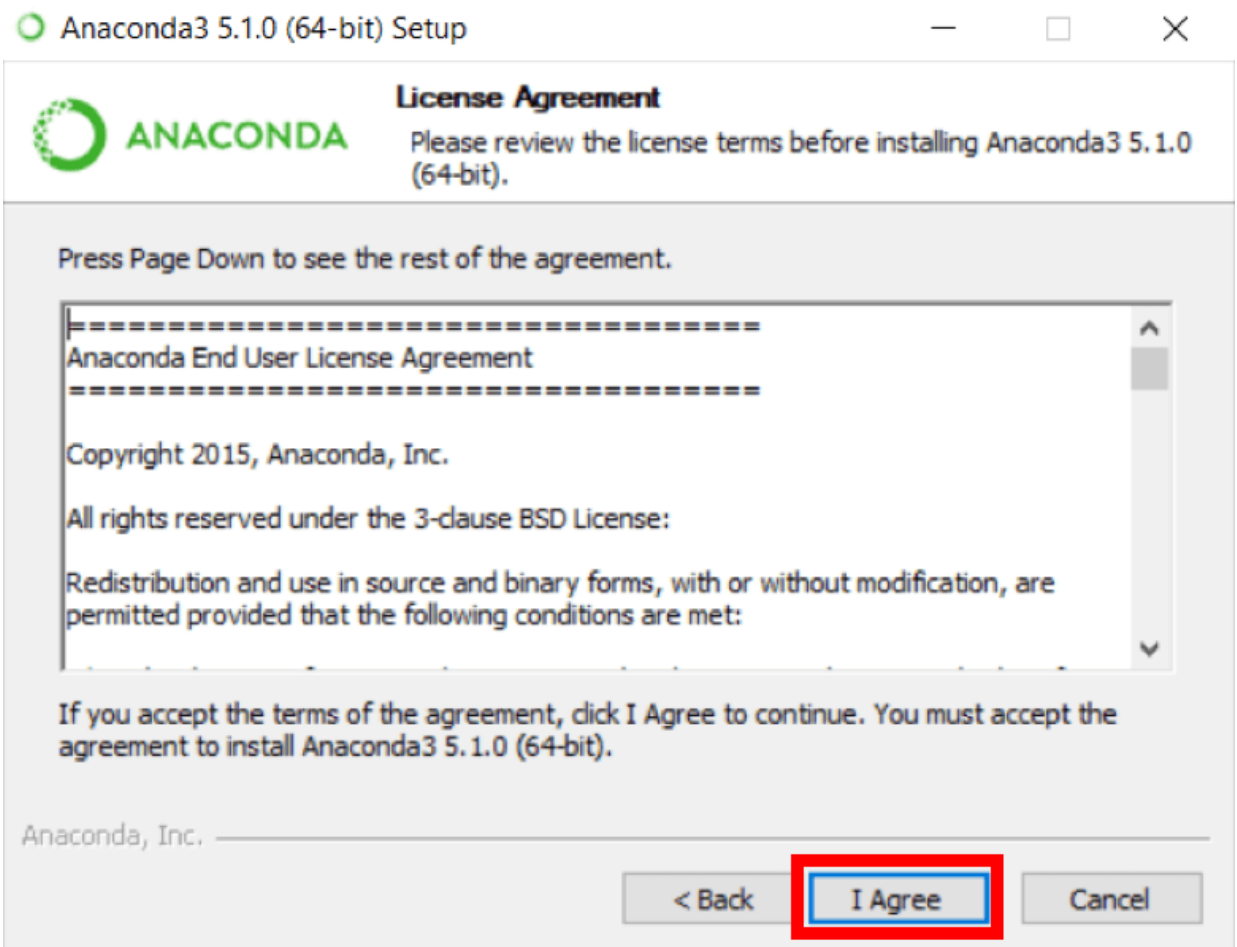
2. Locate your download and double click it.



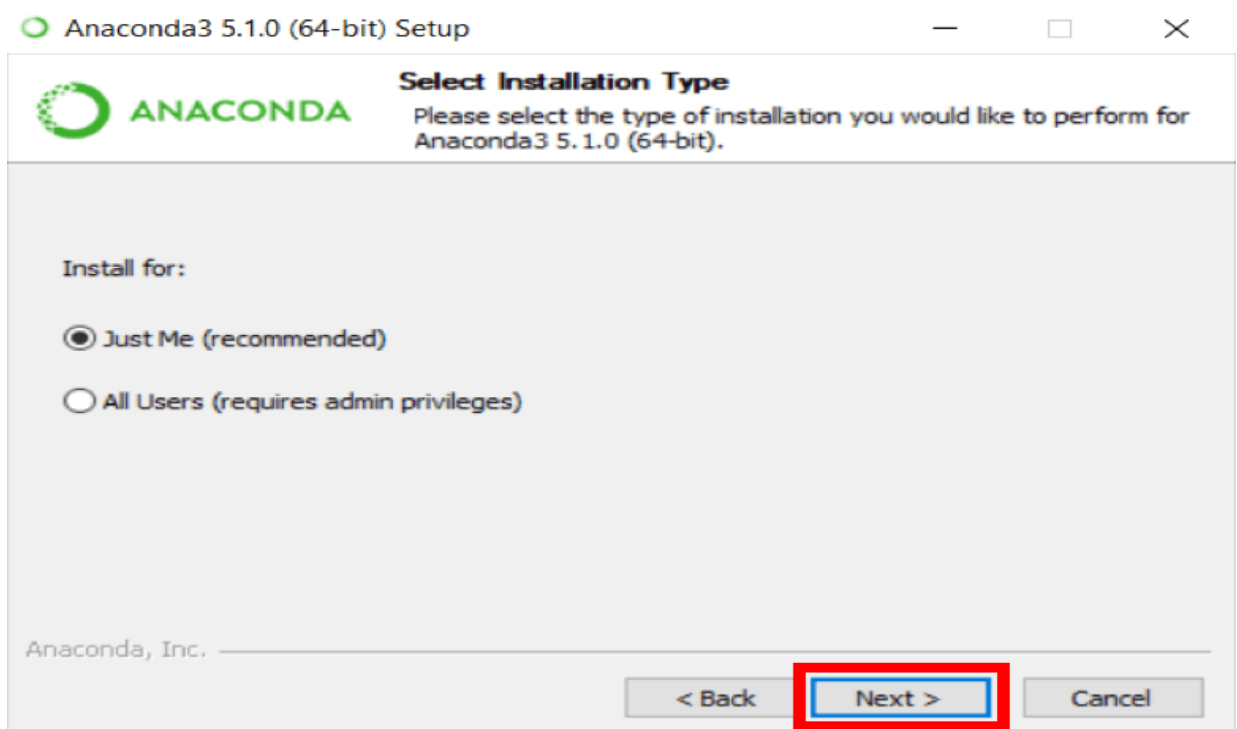
When the screen below appears, click on Next.



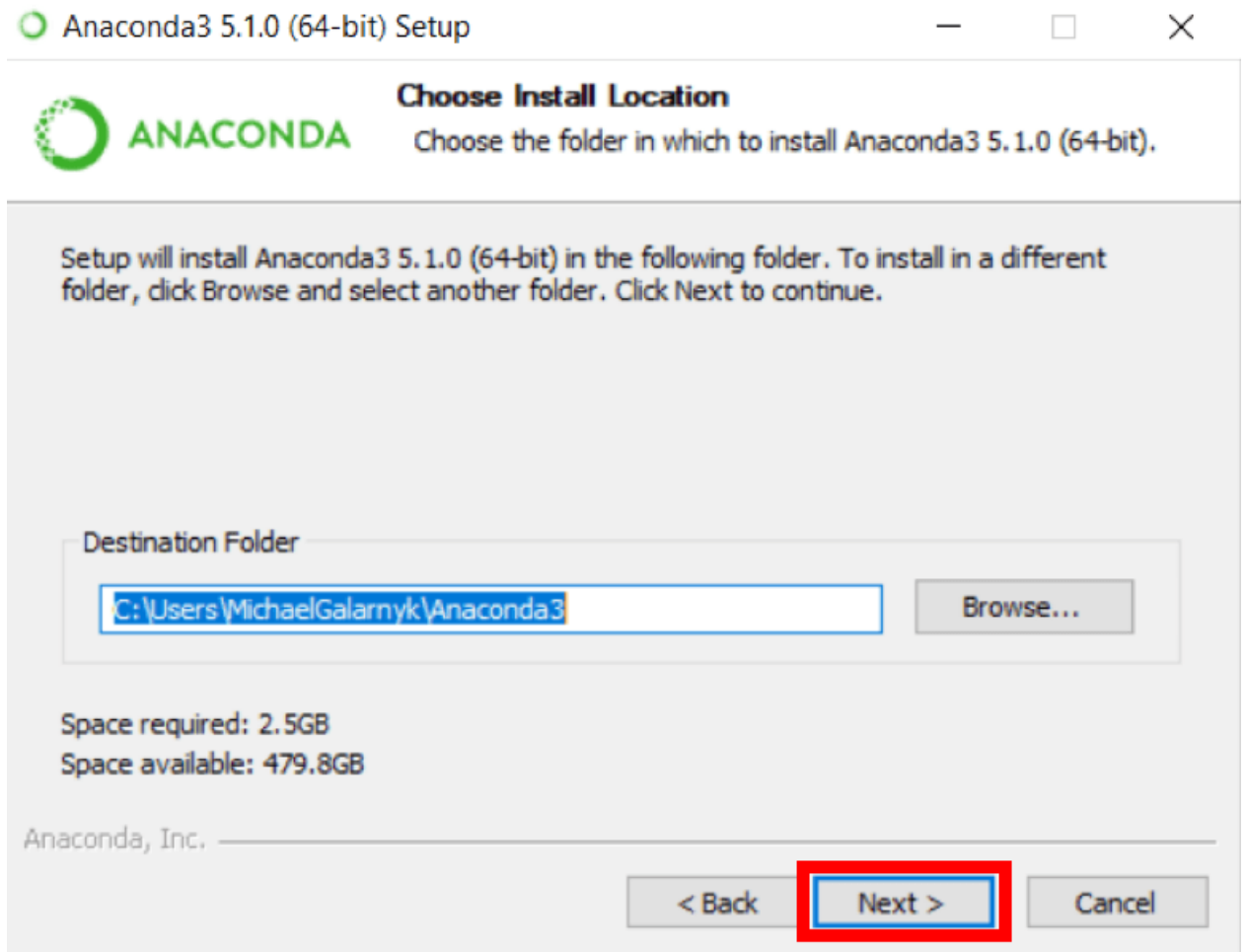
3. Read the license agreement and click on I Agree.



4. Click on Next.

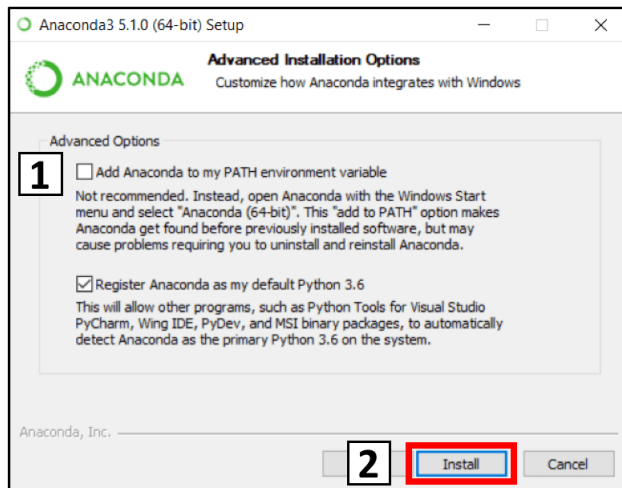


5. Note your installation location and then click Next.

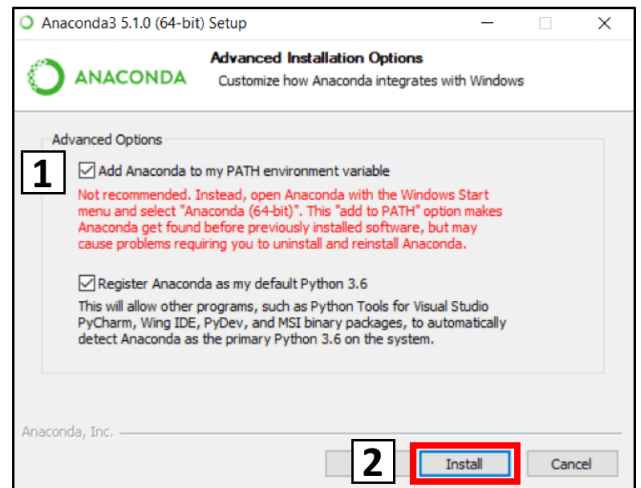


6. This is an important part of the installation process. The recommended approach is to not check the box to add Anaconda to your path. This means you will have to use Anaconda Navigator or the Anaconda Command Prompt (located in the Start Menu under "Anaconda") when you wish to use Anaconda (you can always add Anaconda to your PATH later if you don't check the box). If you want to be able to use Anaconda in your command prompt (or git bash, cmd, powershell etc), please use the alternative approach and check the box.

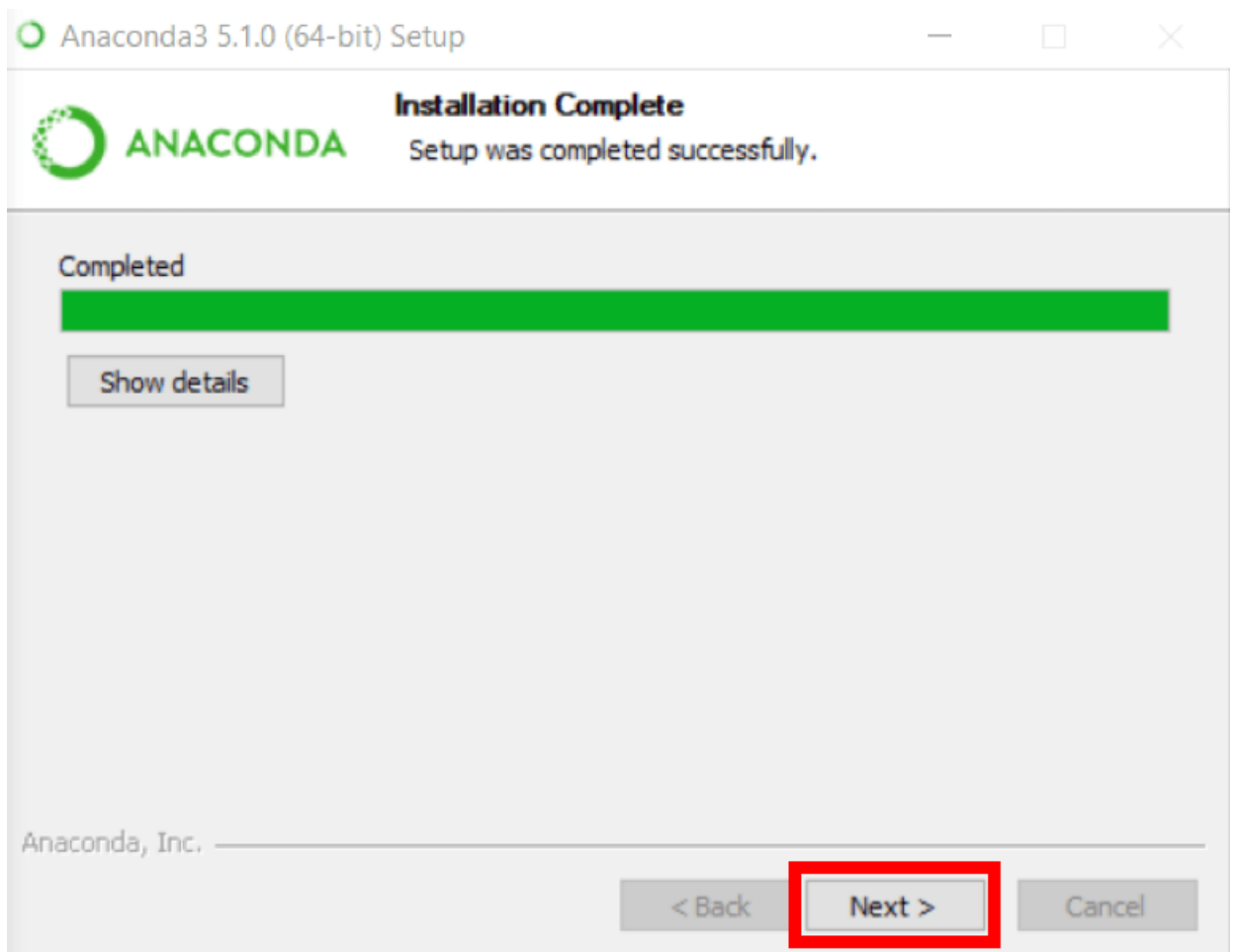
Recommended Approach



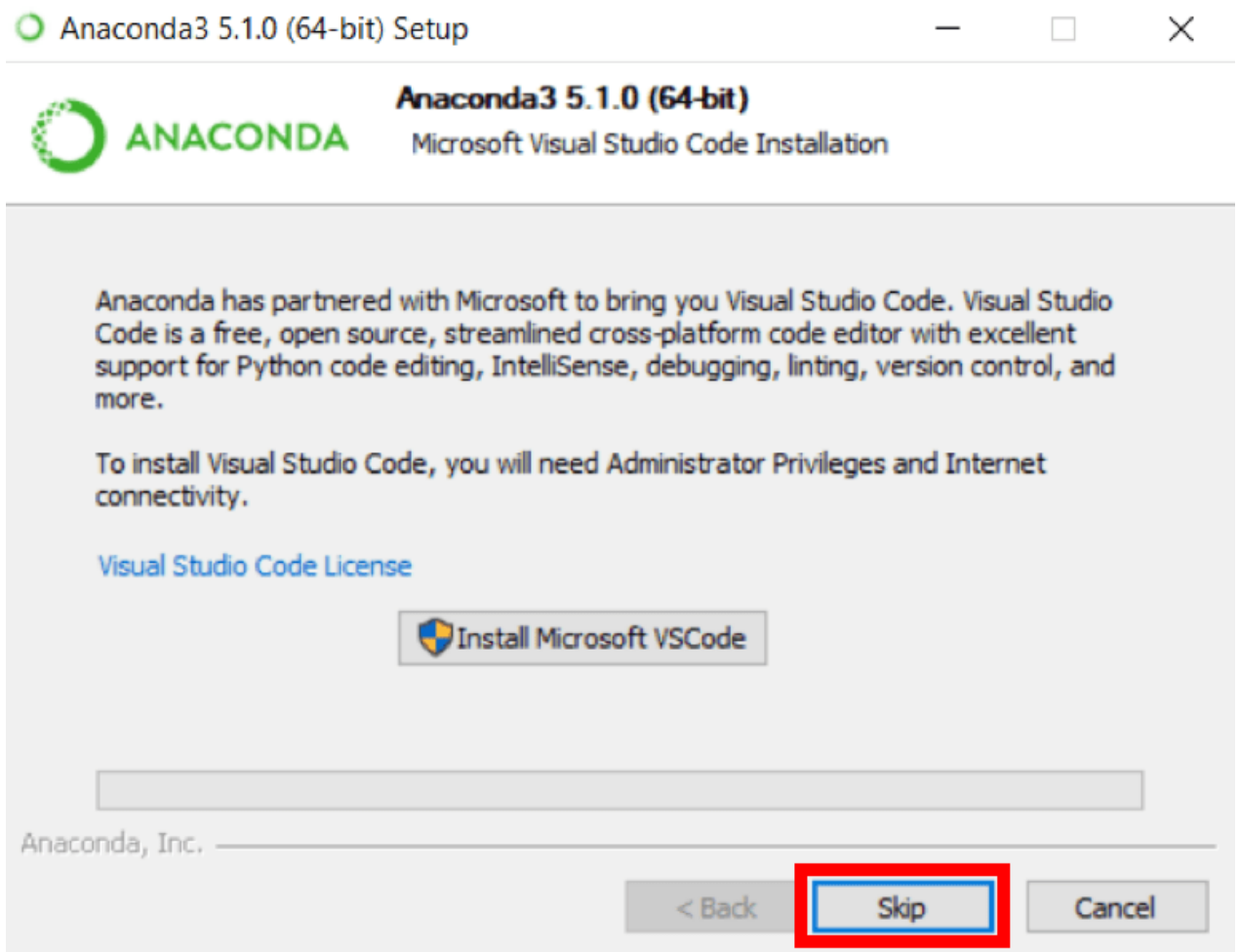
Alternative Approach



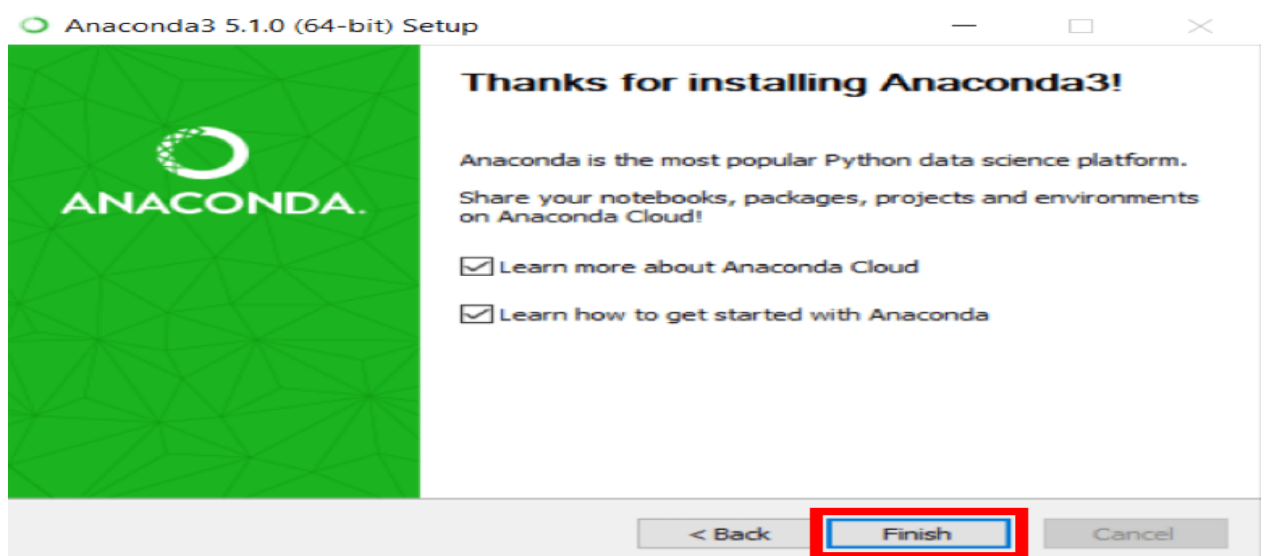
7. Click on Next.



8. You can install Microsoft VSCode if you wish, but it is optional.



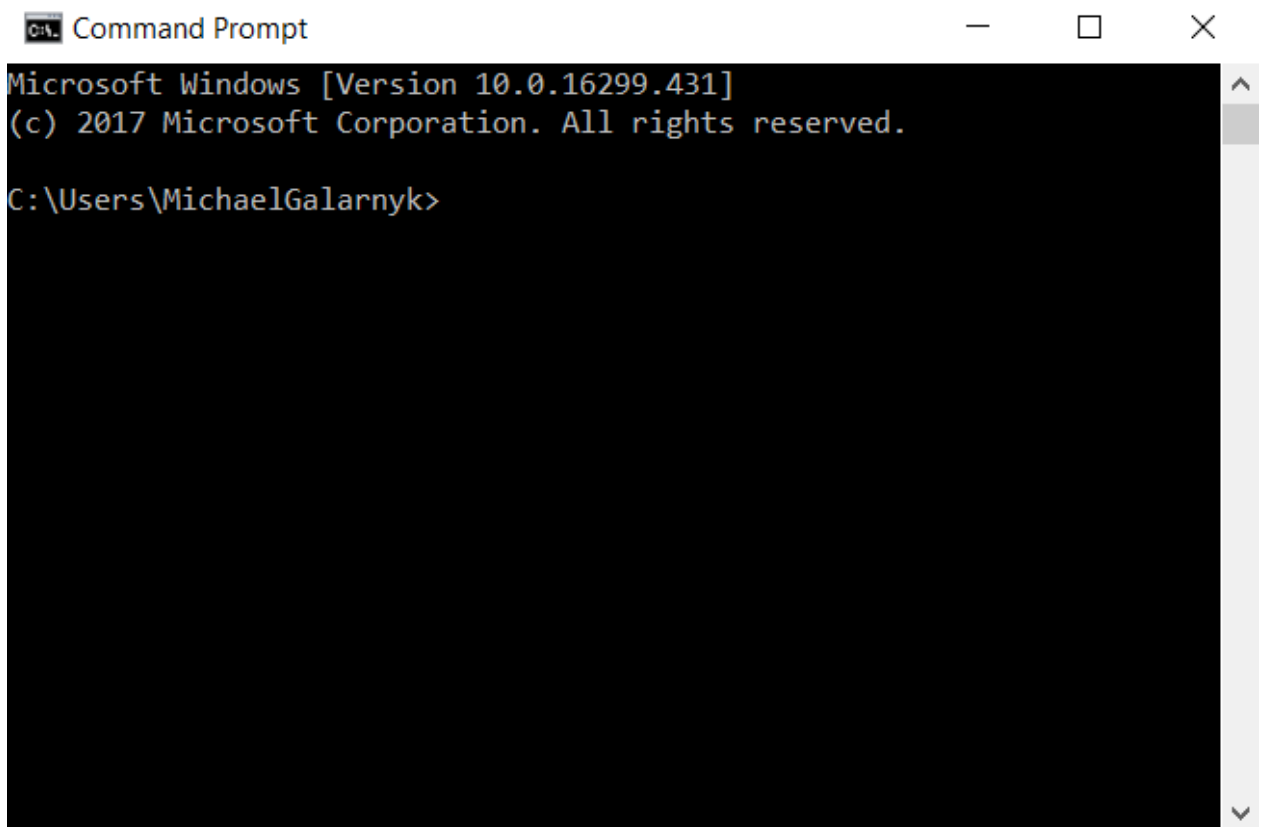
9. Click on Finish.



How to Add Anaconda to Path (Optional)

This is an **optional** step. This is for the case where you didn't check the box in step 6 and now want to add Anaconda to your Path. The advantage of this is that you will be able to use Anaconda in your Command Prompt, Git Bash, cmd etc.

1. Open a Command Prompt.



2. Check if you already have Anaconda added to your path. Enter the commands below into your Command Prompt. This is checking if you already have Anaconda added to your path. If you get a command **not recognized** error like in the left side of the image below, proceed to step 3. If you get an output similar to the right side of the image below, you have already added Anaconda to your path.

Proceed to Step 3

```
Select Command Prompt
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\MichaelGalarnyk>conda --version
'conda' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\MichaelGalarnyk>python --version
'python' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\MichaelGalarnyk>
```

Anaconda Already Added to Path

```
Select Command Prompt
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\MichaelGalarnyk>conda --version
conda 4.4.10

C:\Users\MichaelGalarnyk>python --version
Python 3.6.4 :: Anaconda, Inc.

C:\Users\MichaelGalarnyk>
```

3. If you don't know where your conda and/or python is, open an **Anaconda Prompt** and type in the following commands. This is telling you where conda and python are located on your computer.

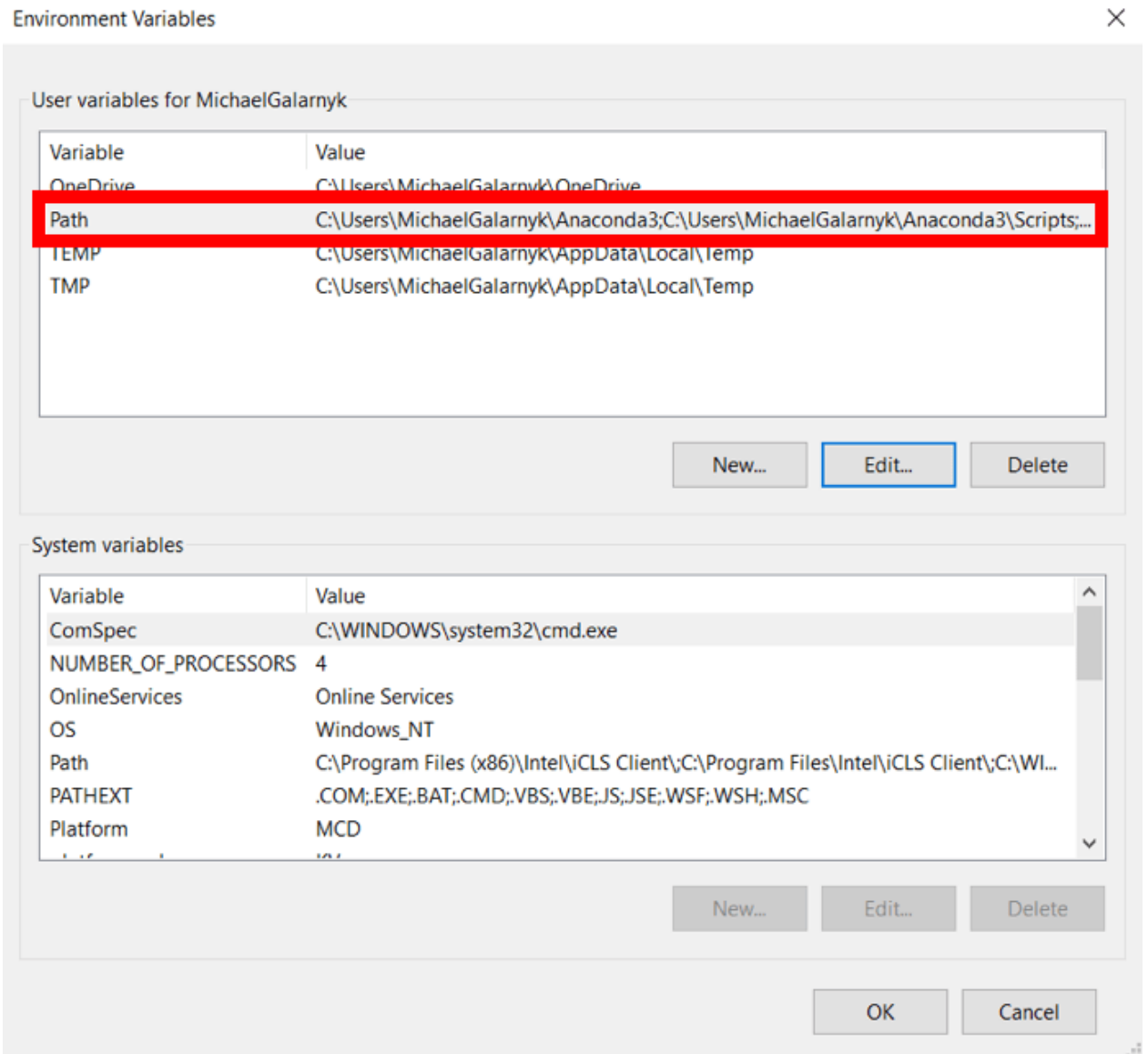
```
Anaconda Prompt

(base) C:\Users\MichaelGalarnyk>where conda
C:\Users\MichaelGalarnyk\Anaconda3\Library\bin\conda.bat
C:\Users\MichaelGalarnyk\Anaconda3\Scripts conda.exe

(base) C:\Users\MichaelGalarnyk>where python
C:\Users\MichaelGalarnyk\Anaconda3\python.exe

(base) C:\Users\MichaelGalarnyk>
```

4. Add conda and python to your PATH. You can do this by going to your Environment Variables and adding the output of step 3 (enclosed in the red rectangle) to your path. If you are having issues, here is a short [video](#) on adding conda and python to your PATH.



5. Open a new **Command Prompt**. Try typing `conda --version` and `python --version` into the **Command Prompt** to check to see if everything went well.

```

C:\> Select Command Prompt

Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\MichaelGalarnyk>conda --version
conda 4.4.10

C:\Users\MichaelGalarnyk>python --version
Python 3.6.4 :: Anaconda, Inc.

C:\Users\MichaelGalarnyk>

```

CHAPTER 5

CODING AND TESTING

5.1 CODING

Once the design aspect of the system is finalized the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

5.2 CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

- Program should be simple, clear and easy to understand.
- Naming conventions
- Value conventions
- Script and comment procedure
- Message box format
- Exception and error handling

5.2.1 NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the

variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

5.2.1.1 Class Names

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

5.2.1.2 Member Function and Data Member name

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

5.2.2 VALUE CONVENTIONS

Value conventions ensure values for variable at any point of time. This involves the following:

- Proper default values for the variables.
- Proper validation of values in the field.
- Proper documentation of flag values.

5.2.3 SCRIPT WRITING AND COMMENTING STANDARD

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

5.2.4 MESSAGE BOX FORMAT

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

- X – User has performed illegal operation.
- ! – Information to the user.

5.3 TEST PROCEDURE

5.3.1 SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

5.4 TEST DATA AND OUTPUT

5.4.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

5.4.2 FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

5.4.3 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of

modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

5.5 TESTING TECHNIQUES / TESTING STRATEGIES

5.5.1 TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet – undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to

be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

5.5.1.1 WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving test cases
- Graph matrices Control

5.5.1.2 BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

5.5.1.3 PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

5.5.1.4 SECURITY TESTING:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

5.5.1.5 VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- The function or performance characteristics confirm to specifications and are accepted.
- A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic.

5.5.1.6 USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.

- Output screen design.

5.6 CODE:

```
{% extends "base.html" %} {% block content %}

<div class="main">

<div class="title">

<h3>Kidney CT Classifier</h3>

<!-- <p>

<small>A web app demo</small>

</p> -->

</div>

<div class="panel">

<input id="file-upload" class="hidden" type="file" accept="image/x-
png,image/gif,image/jpeg" />

<label for="file-upload" id="file-drag" class="upload-box">

<div id="upload-caption">Drop image here or click to select</div>

<img id="image-preview" class="hidden" />

</label>

</div>

<div style="margin-bottom: 2rem;">

<input type="button" value="Submit" class="button" onclick="submitImage();" />

<input type="button" value="Clear" class="button" onclick="clearImage();" />

</div>

<div id="image-box">

<img id="image-display" />
```

```

<div id="pred-result" class="hidden"></div>

<svg id="loader" class="hidden" viewBox="0 0 32 32" width="32" height="32">

<circle id="spinner" cx="16" cy="16" r="14" fill="none"></circle>

</svg>

</div>

</div>

```

```
{% endblock %}
```

```

//=====================================================
=====

// Drag and drop image handling

//=====================================================
=====

```

```

var fileDrag = document.getElementById("file-drag");
var fileSelect = document.getElementById("file-upload");

// Add event listeners

fileDrag.addEventListener("dragover", fileDragHover, false);
fileDrag.addEventListener("dragleave", fileDragHover, false);
fileDrag.addEventListener("drop", fileSelectHandler, false);
fileSelect.addEventListener("change", fileSelectHandler, false);

function fileDragHover(e) {

```

```

// prevent default behaviour

e.preventDefault();

e.stopPropagation();


fileDrag.className = e.type === "dragover" ? "upload-box dragover" : "upload-
box";
}


function fileSelectHandler(e) {

    // handle file selecting

    var files = e.target.files || e.dataTransfer.files;

    fileDragHover(e);

    for (var i = 0, f; (f = files[i]); i++) {

        previewFile(f);

    }

}


//=====

=====

// Web page elements for functions to use

//=====

=====


var imagePreview = document.getElementById("image-preview");
var imageDisplay = document.getElementById("image-display");

```

```

var uploadCaption = document.getElementById("upload-caption");

var predResult = document.getElementById("pred-result");

var loader = document.getElementById("loader");


//=====================================================
=====

// Main button events

//=====================================================
=====


function submitImage() {
    // action for the submit button

    console.log("submit");


    if (!imageDisplay.src || !imageDisplay.src.startsWith("data")) {
        window.alert("Please select an image before submit.");
        return;
    }


    loader.classList.remove("hidden");
    imageDisplay.classList.add("loading");


    // call the predict function of the backend

    predictImage(imageDisplay.src);
}

```

```

function clearImage() {

    // reset selected files

    fileSelect.value = "";

    // remove image sources and hide them

    imagePreview.src = "";

    imageDisplay.src = "";

    predResult.innerHTML = "";

    hide(imagePreview);

    hide(imageDisplay);

    hide(loader);

    hide(predResult);

    show(uploadCaption);

    imageDisplay.classList.remove("loading");

}

```

```

function previewFile(file) {

    // show the preview of the image

    console.log(file.name);

    var fileName = encodeURIComponent(file.name);

    var reader = new FileReader();

```

```

reader.readAsDataURL(file);

reader.onloadend = () => {

  imagePreview.src = URL.createObjectURL(file);


  show(imagePreview);
  hide(uploadCaption);


  // reset
  predResult.innerHTML = "";
  imageDisplay.classList.remove("loading");


  displayImage(reader.result, "image-display");
};
}


//=====

=====

// Helper functions

//=====

=====

const getBase64FromUrl = async (url) => {
  const data = await fetch(url);
  const blob = await data.blob();
  return new Promise((resolve) => {
    const reader = new FileReader();

```

```

reader.readAsDataURL(blob);

reader.onloadend = () => {

    const base64data = reader.result.replace("data:", "")

        .replace(/^.+/, "");

    resolve(base64data);

}

});

}

async function predictImage() {

var image_preview = document.getElementById('image-preview');
var base64Data = await getBase64FromUrl(image_preview.src);
var base64DataString = String(base64Data);

var requestBody = JSON.stringify({

    "data":base64DataString

});

console.log(requestBody);

fetch("/predict", {

    method: "POST",

    headers: {

        "Content-Type": "application/json"

    },

    body: requestBody

```

```

    })

    .then(resp => {

        if (resp.ok)

            resp.json().then(data => {

                displayResult(data);

            });

    })

    .catch(err => {

        console.log("An error occurred", err.message);

        window.alert("Oops! Something went wrong.");

    });

}

```

```

function displayImage(image, id) {

    // display image on given id <img> element

    let display = document.getElementById(id);

    display.src = image;

    show(display);

}

```

```

function displayResult(data) {

    // display the result

    // imageDisplay.classList.remove("loading");

    hide(loader);

    predResult.innerHTML = data.result;

```

```
    show(predResult);  
}
```

```
function hide(el) {  
    // hide an element  
    el.classList.add("hidden");  
}
```

```
function show(el) {  
    // show an element  
    el.classList.remove("hidden");  
}
```

```
import splitfolders  
  
splitfolders.ratio("../input/ct-kidney-dataset-normal-cyst-tumor-and-stone/CT-  
KIDNEY-DATASET-Normal-Cyst-Tumor-Stone/CT-KIDNEY-DATASET-Normal-  
Cyst-Tumor-Stone",    output="dataset",    seed=1337,    ratio=(.8,    .1,    .1),  
group_prefix=None)
```

```
import numpy as np  
  
import pandas as pd  
  
from sklearn.utils.multiclass import unique_labels  
  
import os  
  
import matplotlib.pyplot as plt  
  
import matplotlib.image as mpimg  
  
import seaborn as sns
```

```

import itertools

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix

from keras import Sequential

from keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.optimizers import SGD,Adam

from keras.callbacks import ReduceLROnPlateau

from keras.layers import Flatten,Dense,BatchNormalization,Activation,Dropout

from tensorflow.keras.utils import to_categorical

# Import all the libraries

import os

import shutil

from PIL import Image, ImageOps

import matplotlib.pyplot as plt


import seaborn as sns

sns.set_style('white')

import plotly.graph_objects as go

import plotly.express as px

import datetime

from plotly.offline import init_notebook_mode, iplot

import plotly.io as pio

pio.templates.default = 'plotly_white'

from plotly.subplots import make_subplots

init_notebook_mode(connected=True)

```

```

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Activation, Dropout, Concatenate,
Flatten, MaxPooling2D, Conv2D

from tensorflow.keras.preprocessing.image import ImageDataGenerator

channels=3

batch_size=25

test_batch_size=32

test_steps=1

train_path = './dataset/train'

test_path = './dataset/test'

val_path = './dataset/val'

print ( 'test batch size: ',test_batch_size, ' test steps: ', test_steps)

def scalar(img):

    return img # EfficientNet expects pixels in range 0 to 255 so no scaling is
required

trgen=ImageDataGenerator(preprocessing_function=scalar, horizontal_flip=True)

tvgen=ImageDataGenerator(preprocessing_function=scalar)

train_generator=trgen.flow_from_directory(          directory=train_path          ,
target_size=(224,224), class_mode='categorical',
                                color_mode='rgb', shuffle=True, batch_size=batch_size)

test_generator=tvgen.flow_from_directory(          directory=test_path,
target_size=(224,224), class_mode='categorical',
                                color_mode='rgb',                                shuffle=False,
batch_size=test_batch_size)

```

```

valid_generator=tvgen.flow_from_directory(directory=val_path,
target_size=(224,224), class_mode='categorical',
color_mode='rgb', shuffle=True, batch_size=batch_size)

classes=list(train_generator.class_indices.keys())
class_count=len(classes)
train_steps=int(np.ceil(len(train_generator.labels)/batch_size))

base_model = VGG16(include_top = False, weights = 'imagenet', input_shape =
(224,224,3), classes = 38)

model= Sequential()
model.add(base_model)
model.add(Flatten())

model.add(Dense(1024,activation=('relu'),input_dim=512))
model.add(Dense(512,activation=('relu')))
model.add(Dense(256,activation=('relu')))
model.add(Dropout(.3))
model.add(Dense(128,activation=('relu')))
#model.add(Dropout(.2))
model.add(Dense(4,activation=('softmax')))

#Checking the final model summary

```

```
model.summary()
```

```
batch_size= 1218
```

```
epochs=15
```

```
learn_rate=.001
```

```
sgd=SGD(learning_rate=learn_rate,momentum=.9,nesterov=False)
```

```
adam=Adam(learning_rate=learn_rate, beta_1=0.9, beta_2=0.999, epsilon=None,  
decay=0.0, amsgrad=False)
```

```
model.compile(optimizer=sgd,loss='categorical_crossentropy',metrics=['accuracy'])
```

```
history = model.fit(train_generator,epochs = 5  
    , validation_data = valid_generator, verbose = 1)
```

```
def print_info( test_gen, preds, print_code, save_dir, subject ):
```

```
    class_dict=test_gen.class_indices
```

```
    labels= test_gen.labels
```

```
    file_names= test_gen filenames
```

```
    error_list=[]
```

```
    true_class=[]
```

```
    pred_class=[]
```

```
    prob_list=[]
```

```
    new_dict={}
```

```
    error_indices=[]
```

```
    y_pred=[]
```

```
    for key,value in class_dict.items():
```

```

        new_dict[value]=key          # dictionary {integer of class number: string of
class name}

# store new_dict as a text file in the save_dir

classes=list(new_dict.values())    # list of string of class names

errors=0

for i, p in enumerate(preds):

    pred_index=np.argmax(p)

    true_index=labels[i] # labels are integer values

    if pred_index != true_index: # a misclassification has occurred

        error_list.append(file_names[i])

        true_class.append(new_dict[true_index])

        pred_class.append(new_dict[pred_index])

        prob_list.append(p[pred_index])

        error_indices.append(true_index)

        errors=errors + 1

    y_pred.append(pred_index)

if print_code !=0:

    if errors>0:

        if print_code>errors:

            r=errors

        else:

            r=print_code

        msg='{0:^28s}{1:^28s}{2:^28s}{3:^16s}'.format('Filename', 'Predicted Class'
, 'True Class', 'Probability')

    print_in_color(msg, (0,255,0),(55,65,80))

```

```

for i in range(r):

    split1=os.path.split(error_list[i])

    split2=os.path.split(split1[0])

    fname=split2[1] + '/' + split1[1]

    msg='{0:^28s}{1:^28s}{2:^28s}{3:4s}{4:^6.4f}'.format(fname,
pred_class[i],true_class[i], ' ', prob_list[i])

    print_in_color(msg, (255,255,255), (55,65,60))

    #print(error_list[i] , pred_class[i], true_class[i], prob_list[i])

else:

    msg='With accuracy of 100 % there are no errors to print'

    print_in_color(msg, (0,255,0),(55,65,80))

if errors>0:

    plot_bar=[]

    plot_class=[]

    for key, value in new_dict.items():

        count=error_indices.count(key)

        if count!=0:

            plot_bar.append(count) # list containg how many times a class c had an
error

            plot_class.append(value) # stores the class

fig=plt.figure()

fig.set_figheight(len(plot_class)/3)

fig.set_figwidth(10)

plt.style.use('fivethirtyeight')

for i in range(0, len(plot_class)):

```

```

        c=plot_class[i]

        x=plot_bar[i]

        plt.barh(c, x, )

        plt.title( ' Errors by Class on Test Set')

y_true= np.array(labels)
y_pred=np.array(y_pred)
if len(classes)<= 40:

    # create a confusion matrix

    cm = confusion_matrix(y_true, y_pred )

    length=len(classes)

    if length<8:

        fig_width=8

        fig_height=8

    else:

        fig_width= int(length * .5)

        fig_height= int(length * .5)

    plt.figure(figsize=(fig_width, fig_height))

    sns.heatmap(cm, annot=True, vmin=0, fmt='g', cmap='Blues', cbar=False)

    plt.xticks(np.arange(length)+.5, classes, rotation= 90)

    plt.yticks(np.arange(length)+.5, classes, rotation=0)

    plt.xlabel("Predicted")

    plt.ylabel("Actual")

    plt.title("Confusion Matrix")

    plt.show()

clr = classification_report(y_true, y_pred, target_names=classes)

```

```

print("Classification Report:\n-----\n", clr)

def print_in_color(txt_msg,fore_tuple,back_tuple,):

    #prints the text_msg in the foreground color specified by fore_tuple with the
    background specified by back_tuple

    #text_msg is the text, fore_tuple is foreground color tuple (r,g,b), back_tuple
    is background tuple (r,g,b)

    rf,gf,bf=fore_tuple

    rb,gb,bb=back_tuple

    msg='{0}' + txt_msg

    mat='\33[38;2;' + str(rf) + ';' + str(gf) + ';' + str(bf) + ';48;2;' + str(rb) + ';' + str(gb) +
    ';' + str(bb) + 'm'

    print(msg .format(mat), flush=True)

    print('\33[0m', flush=True) # returns default print color to back to black

    return

import seaborn as sns

sns.set_style('darkgrid')

from PIL import Image

from sklearn.metrics import confusion_matrix, classification_report

p = model.predict(test_generator)

print_info( test_generator, p, 10, r './', 'kidney' )


model.evaluate(test_generator)


model.save("vgg16.h5")

```

5.7 SCREENSHOT

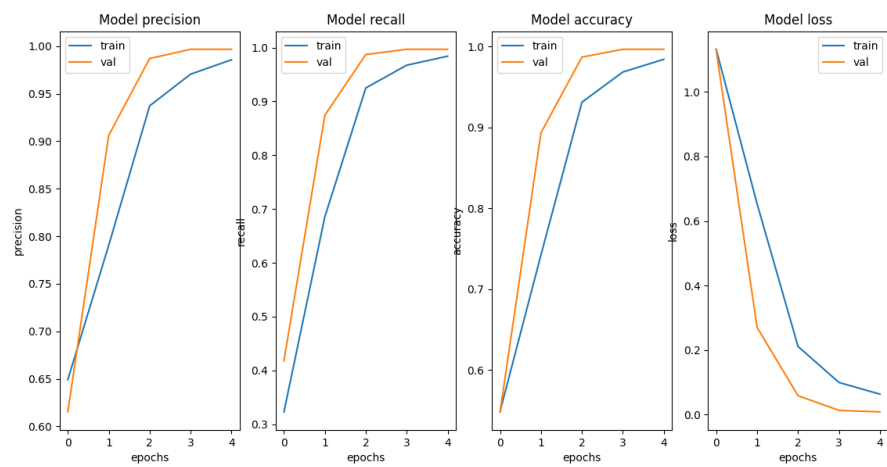


Fig 5.1 Precision,Recall,Accuracy,Loss

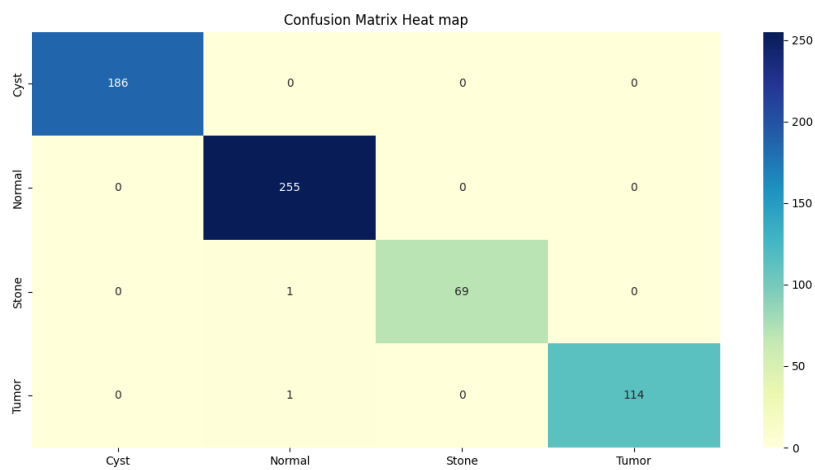


Fig 5.2 Confusion Matrix Heat Map

Epoch 1/5

2023-01-27 21:12:54.467668: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 501811200
exceeds 10% of free system memory.

2023-01-27 21:12:56.870805: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 501811200
exceeds 10% of free system memory.

1/112 [.....] - ETA: 12:00 - loss: 1.3874 - accuracy: 0.2800 -
precision: 0.0000e+00 - recall: 0.0000e+00 2023-01-27 21:12:58.233996: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 501811200
exceeds 10% of free system memory.

2023-01-27 21:13:00.270057: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 501811200
exceeds 10% of free system memory.

2/112 [.....] - ETA: 6:01 - loss: 1.3788 - accuracy: 0.3250 -
precision: 0.0000e+00 - recall: 0.0000e+00 2023-01-27 21:13:01.719634: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 501811200
exceeds 10% of free system memory.

112/112 [=====] - 382s 3s/step - loss: 1.1325 -
accuracy: 0.5489 - precision: 0.6491 - recall: 0.3229 - val_loss: 1.1297 -
val_accuracy: 0.5484 - val_precision: 0.6152 - val_recall: 0.4177

Epoch 2/5

112/112 [=====] - 358s 3s/step - loss: 0.6531 -
accuracy: 0.7427 - precision: 0.7906 - recall: 0.6857 - val_loss: 0.2698 -
val_accuracy: 0.8935 - val_precision: 0.9064 - val_recall: 0.8742

Epoch 3/5

112/112 [=====] - 353s 3s/step - loss: 0.2108 -
accuracy: 0.9313 - precision: 0.9373 - recall: 0.9250 - val_loss: 0.0580 -
val_accuracy: 0.9871 - val_precision: 0.9871 - val_recall: 0.9871

Epoch 4/5

112/112 [=====] - 374s 3s/step - loss: 0.0991 -
accuracy: 0.9687 - precision: 0.9706 - recall: 0.9672 - val_loss: 0.0125 -
val_accuracy: 0.9968 - val_precision: 0.9968 - val_recall: 0.9968

Epoch 5/5

112/112 [=====] - 426s 4s/step - loss: 0.0629 -
accuracy: 0.9844 - precision: 0.9858 - recall: 0.9839 - val_loss: 0.0079 -
val_accuracy: 0.9968 - val_precision: 0.9968 - val_recall: 0.9968

7/7 [=====] - 9s 1s/step

Accuracy: 0.9968051118210862

Precision: 0.9980544747081712

Recall: 0.9942546583850932

F1_score: 0.9961331732921523

5.8 Model Training Progress

The training process involved five epochs, each providing insights into the model's performance on the training and validation datasets.

- **Epoch 1:**
 - Loss: 1.1325
 - Accuracy: 54.89%
 - Precision: 64.91%
 - Recall: 32.29%
- **Epoch 2:**
 - **Significant improvement seen:**
 - Loss reduced to 0.6531
 - Accuracy increased to 74.27%
 - Precision improved to 79.06%
 - Recall increased to 68.57%
- **Epoch 3:**
 - **Further improvement:**
 - Loss reduced to 0.2108
 - Accuracy reached 93.13%
 - High precision (93.73%) and recall (92.50%)
- **Epoch 4:**
 - **Continued improvement:**
 - Loss reduced to 0.0991
 - Accuracy at 96.87%
 - High precision (97.06%) and recall (96.72%)

- **Epoch 5:**
 - **Convergence to an impressive state:**
 - Loss further reduced to 0.0629
 - Accuracy reached 98.44%
 - High precision (98.58%) and recall (98.39%)

5.9 Model Evaluation on Validation Data

- **Final Evaluation**
 - Accuracy: 99.68%
 - Precision: 99.81%
 - Recall: 99.43%
 - F1 Score: 99.61%

5.10 Discussion

5.10.1 Training Progress

The model shows a remarkable improvement throughout the training epochs, indicating its ability to learn from the data.

5.10.2 Accuracy and Performance

The final accuracy of 99.68% on the validation set suggests the model's ability to generalize well to unseen data.

5.10.3 Precision and Recall

High precision and recall values demonstrate the model's effectiveness in minimizing false positives and false negatives, respectively.

5.10.4 Overfitting

The gradual reduction in training and validation loss across epochs suggests that the model is learning without overfitting.

5.10.4.1 Generalization

The high accuracy and balanced precision-recall values indicate that the model has successfully generalized patterns from the training data.

5.10.4.2 Application

The model, with its high accuracy, precision, and recall, can be confidently deployed for the intended task, whether it be image classification or another related problem.

CHAPTER 6

CONCLUSION

Kidney abnormality is a major health concern globally, and the use of deep learning models in kidney disease diagnosis has shown promise in recent years. Specifically, in the detection of kidney cysts, stones, and tumors, a lightweight customized convolution neural network was proposed, which outperformed other state-of-the-art methods and achieved an accuracy of 99.52%. The proposed model also provides better interpretive power, enabling clinicians to understand the specific decisions made by the model. Additionally, the deployment of the model using Flask framework for web page prediction makes it accessible to clinicians and patients, allowing for more efficient and accurate kidney disease diagnosis. Future research in this area should focus on expanding the scope of the model to detect other kidney abnormalities and on validating the model's performance on larger datasets.

Its remarkable accuracy in detecting kidney abnormalities, the proposed lightweight convolutional neural network (CNN) offers several key advantages that have the potential to revolutionize kidney disease diagnosis and treatment. By leveraging deep learning techniques, the model not only identifies cysts, stones, and tumors with unparalleled precision but also enhances the interpretive power of diagnostic tools for clinicians. Furthermore, the deployment of the CNN model through Flask framework for web page prediction signifies a significant step towards democratizing access to advanced medical technologies. This user-friendly interface facilitates seamless interaction between healthcare providers and the diagnostic system, ultimately streamlining the diagnostic process and improving patient outcomes. However, while the current focus has been on the detection of specific kidney abnormalities, future research endeavors should aim to broaden the scope of the model to encompass a wider range of renal conditions. By expanding the model's capabilities to include the detection of additional abnormalities such as renal artery stenosis, hydronephrosis, and renal cell carcinoma, clinicians can benefit from a more comprehensive diagnostic toolset. Moreover, validating the performance of the CNN model on larger and more diverse datasets will be essential to ensuring its robustness and reliability in

real-world clinical settings. Collaborative efforts involving interdisciplinary teams of clinicians, data scientists, and machine learning experts will be crucial in driving forward the development and validation of these cutting-edge diagnostic technologies. In conclusion, the advancement of deep learning models in kidney disease diagnosis holds immense promise for improving healthcare outcomes and addressing the global burden of kidney abnormalities. By continuing to innovate and refine these technologies, we can pave the way towards more accurate, efficient, and accessible diagnostic solutions for patients worldwide.

REFERENCES

1. Maynar, J.; Barrasa, H.; Martin, A.; Usón, E.; Fonseca, F. Kidney Support in Sepsis. In *The Sepsis Codex-E-Book*; Elsevier Health Sciences: Amsterdam, The Netherlands, 2023; p. 169.
2. Li, M.; Chi, X.; Wang, Y.; Setrerrahmane, S.; Xie, W.; Xu, H. Trends in insulin resistance: Insights into mechanisms and therapeutic strategy. *Signal Transduct. Target. Ther.* **2022**, *7*, 216.
3. Yener, S.; Pehlivanoglu, C.; Yildiz, Z.A.; Ilce, H.T.; Ilce, Z. Duplex Kidney Anomalies and Associated Pathologies in Children: A Single-Center Retrospective Review. *Cureus* **2022**, *14*, e25777.
4. Sassanarakkit, S.; Hadpech, S.; Thongboonkerd, V. Theranostic roles of machine learning in clinical management of kidney stone disease. *Comput. Struct. Biotechnol. J.* **2022**, *21*, 260–266.
5. Kanti, S.Y.; Csóka, I.; Jójárt-Laczovich, O.; Adalbert, L. Recent Advances in Antimicrobial Coatings and Material Modification Strategies for Preventing Urinary Catheter-Associated Complications. *Biomedicines* **2022**, *10*, 2580.
6. Ramalingam, H.; Patel, V. Decorating Histones in Polycystic Kidney Disease. *J. Am. Soc. Nephrol.* **2022**, *33*, 1629–1630.
7. Karimi, K.; Nikzad, M.; Kulivand, S.; Borzouei, S. Adrenal Mass in a 70-Year-Old Woman. *Case Rep. Endocrinol.* **2022**, *2022*, 2736199.
8. Islam, M.N.; Hasan, M.; Hossain, M.; Alam, M.; Rabiul, G.; Uddin, M.Z.; Soylu, A. Vision transformer and explainable transfer learning models for auto detection of kidney cyst, stone and tumor from CT-radiography. *Sci. Rep.* **2022**, *12*, 11440.
9. Qadir, A.M.; Abd, D.F. Kidney Diseases Classification using Hybrid Transfer-Learning DenseNet201-Based and Random Forest Classifier. *Kurd. J. Appl. Res.* **2023**, *7*, 131–144. Rajinikanth, V.; Durai Raj Vincent, P.M.; Srinivasan, K.; Ananth Prabhu, G.; Chang, C.Y. Framework to Distinguish Healthy/Cancer Renal CT Images using Fused Deep Features. *Front. Public Health* **2023**, *11*, 39.
10. Yildirim, K.; Bozdog, P.G.; Talo, M.; Yildirim, O.; Karabatak, M.; Acharya, U.R. Deep learning model for automated kidney stone detection using coronal CT images. *Comput. Biol. Med.* **2021**, *135*, 104569.

11. Bayram, A.F.; Gurkan, C.; Budak, A.; Karataş, H. A Detection and Prediction Model Based on Deep Learning Assisted by Explainable Artificial Intelligence for Kidney Diseases. *Avrupa Bilim Teknol. Derg.* **2022**, *40*, 67–74.
12. Loveleen, G.; Mohan, B.; Shikhar, B.S.; Nz, J.; Shorfuzzaman, M.; Masud, M. Explanation-Driven HCI Model to Examine the Mini-Mental State for Alzheimer's Disease. *ACM Trans. Multimed. Comput. Commun. Appl.* **2022**.
13. Gaur, L.; Bhandari, M.; Razdan, T.; Mallik, S.; Zhao, Z. Explanation-Driven Deep Learning Model for Prediction of Brain Tumour Status Using MRI Image Data.
14. *Front. Genet.* **2022**, *13*, 822666. Bhandari, M.; Shahi, T.B.; Siku, B.; Neupane, A. Explanatory classification of CXR images into COVID-19, Pneumonia and Tuberculosis using deep learning and XAI. *Comput. Biol. Med.* **2022**, *150*, 106156.
15. Longo, L.; Goebel, R.; Lecue, F.; Kieseberg, P.; Holzinger, A. Explainable artificial intelligence: Concepts, applications, research challenges and visions. In *Machine Learning and Knowledge Extraction, Proceedings of the 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, 25–28 August 2020*; Springer: Cham, Switzerland, 2020; pp. 1–16.
16. Huang, L.; Shi, Y.; Hu, J.; Ding, J.; Guo, Z.; Yu, B. Integrated analysis of mRNA-seq and miRNA-seq reveals the potential roles of Egr1, Rxra and Max in kidney stone disease. *Urolithiasis* **2023**, *51*, 13
17. Yin, W.; Wang, W.; Zou, C.; Li, M.; Chen, H.; Meng, F.; Dong, G.; Wang, J.; Yu, Q.; Sun, M.; et al. Predicting Tumor Mutation Burden and EGFR Mutation Using Clinical and Radiomic Features in Patients with Malignant Pulmonary Nodules. *J. Pers. Med.* **2023**, *13*, 16.
18. Park, H.J.; Shin, K.; You, M.W.; Kyung, S.G.; Kim, S.Y.; Park, S.H.; Byun, J.H.; Kim, N.; Kim, H.J. Deep Learning-based Detection of Solid and Cystic Pancreatic Neoplasms at Contrast-enhanced CT. *Radiology* **2023**, *306*, 140–149. Wu, Y.; Yi, Z. Automated detection of kidney abnormalities using multi-feature fusion convolutional neural networks. *Knowl.-Based Syst.* **2020**, *200*, 105873. Cerrolaza, J.J.; Peters, C.A.; Martin, A.D.; Myers, E.; Safdar, N.; Linguraru, M.G. Ultrasound based computer-aided-diagnosis of kidneys for pediatric hydronephrosis. In *Medical Imaging 2014: Computer-Aided*

Diagnosis, SPIE Proceedings of the Medical Imaging, San Diego, CA, USA, 15–20 February 2014; SPIE: Bellingham, WA, USA, 2014; Volume 9035, pp. 733–738.

19. Raja, R.A.; Ranjani, J.J. Segment based detection and quantification of kidney stones and its symmetric analysis using texture properties based on logical operators with ultrasound scanning. *Int. J. Comput. Appl.* **2013**, 975, 8887.
20. Mangayarkarasi, T.; Jamal, D.N. PNN-based analysis system to classify renal pathologies in kidney ultrasound images. In Proceedings of the 2017 2nd International Conference on Computing and Communications Technologies (ICCCT), Chennai, India, 23–24 February 2017; pp. 123–126.
21. Bommanna Raja, K.; Madheswaran, M.; Thyagarajah, K. A hybrid fuzzy-neural system for computer-aided diagnosis of ultrasound kidney images using prominent features. *J. Med. Syst.* **2008**, 32, 65–83.
22. Viswanath, K.; Gunasundari, R. Analysis and Implementation of Kidney Stone Detection by Reaction Diffusion Level Set Segmentation Using Xilinx System Generator on FPGA. *VLSI Design* **2015**, 2015, 581961.
23. Sudharson, S.; Kokil, P. An ensemble of deep neural networks for kidney ultrasound image classification. *Comput. Methods Programs Biomed.* **2020**, 197, 105709.
24. Tsai, M.C.; Lu, H.H.S.; Chang, Y.C.; Huang, Y.C.; Fu, L.S. Automatic Screening of Pediatric Renal Ultrasound Abnormalities: Deep Learning and Transfer Learning Approach. *JMIR Med. Inform* **2022**, 10, e40878
25. Bhandari, M.; Neupane, A.; Mallik, S.; Gaur, L.; Qin, H. Auguring Fake Face Images Using Dual Input Convolution Neural Network. *J. Imaging* **2023**, 9, 3.
26. Chowdary, G.J.; Suganya, G.; Premalatha, M.; Yogarajah, P. Nucleus Segmentation and Classification using Residual SE-UNet and Feature Concatenation Approach in Cervical Cytopathology Cell images. *Technol. Cancer Res. Treat.* **2022**, 22, 15330338221134833.
27. Shahi, T.; Sitaula, C.; Neupane, A.; Guo, W. Fruit classification using attention-based MobileNetV2 for industrial applications. *PLoS ONE* **2022**, 17, e0264586.
- Zhao, H.; Wu, J.; Li, Z.; Chen, W.; Zheng, Z. Double Sparse Deep Reinforcement Learning via Multilayer Sparse Coding and Nonconvex Regularized Pruning. *IEEE Trans. Cybern.* **2022**, 53, 765–778.

28. Sitaula, C.; Hossain, M.B. Attention-based VGG-16 model for COVID-19 chest X-ray image classification. *Appl. Intell.* **2021**, *51*, 2850–2863. Sitaula, C.; Shahi, T.; Aryal, S.; Marzbanrad, F. Fusion of multi-scale bag of deep visual words features of chest X-ray images to detect COVID-19 infection. *Sci. Rep.* **2021**, *11*, 23914.
29. Samek, W.; Müller, K.R. Towards explainable artificial intelligence. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*; Springer: Cham, Switzerland, 2019; pp. 5–22.
30. van der Velden, B.H.; Kuijf, H.J.; Gilhuijs, K.G.; Viergever, M.A. Explainable artificial intelligence (XAI) in deep learning-based medical image analysis. *Med. Image Anal.* **2022**, *79*, 102470.
31. Sharma, M.; Goel, A.K.; Singhal, P. Explainable AI Driven Applications for Patient Care and Treatment. In *Explainable AI: Foundations, Methodologies and Applications*; Springer: Cham, Switzerland, 2023; pp. 135–156
32. Ashraf, M.T.; Dey, K.; Mishra, S. Identification of high-risk roadway segments for wrong-way driving crash using rare event modeling and data augmentation techniques. *Accid. Anal. Prev.* **2023**, *181*, 106933. Wong, T.T.; Yeh, P.Y. Reliable accuracy estimates from k-fold cross validation. *IEEE Trans. Knowl. Data Eng.* **2019**, *32*, 1586–1594.
33. Banerjee, P.; Barnwal, R.P. Methods and Metrics for Explaining Artificial Intelligence Models: A Review. In *Explainable AI: Foundations, Methodologies and Applications*; Springer: Cham, Switzerland, 2023; pp. 61–88.
34. Sharma, V.; Mir, R.N.; Rout, R.K. Towards secured image steganography based on content-adaptive adversarial perturbation. *Comput. Electr. Eng.* **2023**, *105*, 108484.
35. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.

37. Turuk, M.; Sreemathy, R.; Kadiyala, S.; Kotecha, S.; Kulkarni, V. CNN Based Deep Learning Approach for Automatic Malaria Parasite Detection. *IAENG Int. J. Comput. Sci.* **2022**, *49*, 1–9.
38. Liang, W.; Tadesse, G.A.; Ho, D.; Fei-Fei, L.; Zaharia, M.; Zhang, C.; Zou, J. Advances, challenges and opportunities in creating data for trustworthy AI. *Nat. Mach. Intell.* **2022**, *4*, 669–677.
39. Li, Y.; Zeng, M.; Zhang, F.; Wu, F.X.; Li, M. DeepCellEss: Cell line-specific essential protein prediction with attention-based interpretable deep learning. *Bioinformatics* **2023**, *39*, btac779.

APPENDIX

A. SOURCE CODE

```
# Flask

from flask import Flask, redirect, url_for, request, render_template, Response,
jsonify, redirect

from werkzeug.utils import secure_filename

from gevent.pywsgi import WSGIServer

# TensorFlow and tf.keras

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.applications.imagenet_utils import preprocess_input,
decode_predictions

from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing import image

# from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2

# Some utilities

import numpy as np

from util import base64_to_pil

# Declare a flask app

app = Flask(__name__)

# model = MobileNetV2(weights='imagenet')

# Model saved with Keras model.save()

MODEL_PATH = 'models/KidneyDiseasesModel.h5'
```

```

model = load_model(MODEL_PATH)

model.make_predict_function() # Necessary

print('loading model & Start serving...')

print('Model loaded. Check http://127.0.0.1:5000/')

def model_predict(img, Model):

    img = img.resize((200, 200))

    img_gray = img.convert('L')

    test_image = image.img_to_array(img_gray)

    test_image = np.expand_dims(test_image, axis=0)

    # Be careful how your trained model deals with the input

    # otherwise, it won't make correct prediction!

    # x = preprocess_input(test_image, mode='tf')

    preds = Model.predict(test_image)

    return preds

def determine_case(x):

    rtn = ""

    if x == 0:

        rtn = "Cyst"

    elif x == 1:

        rtn = "Normal"

    elif x == 2:

        rtn = "Stone"

    elif x == 3:

        rtn = "Tumor"

```

```

    return rtn

@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')

@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        print('-----')
        print('incoming request :')

        # Get the image from post request
        img = base64_to_pil(request.json['data'])

        # Making the prediction
        prediction = model_predict(img, model)[0]
        index_of_max = np.argmax(prediction)
        case = determine_case(index_of_max)
        print('prediction : ', case)
        print('-----')

        # Serialize the result, you can add additional fields
        return jsonify(result=case)

    return None

if __name__ == '__main__':
    # app.run(port=5002, threaded=False)

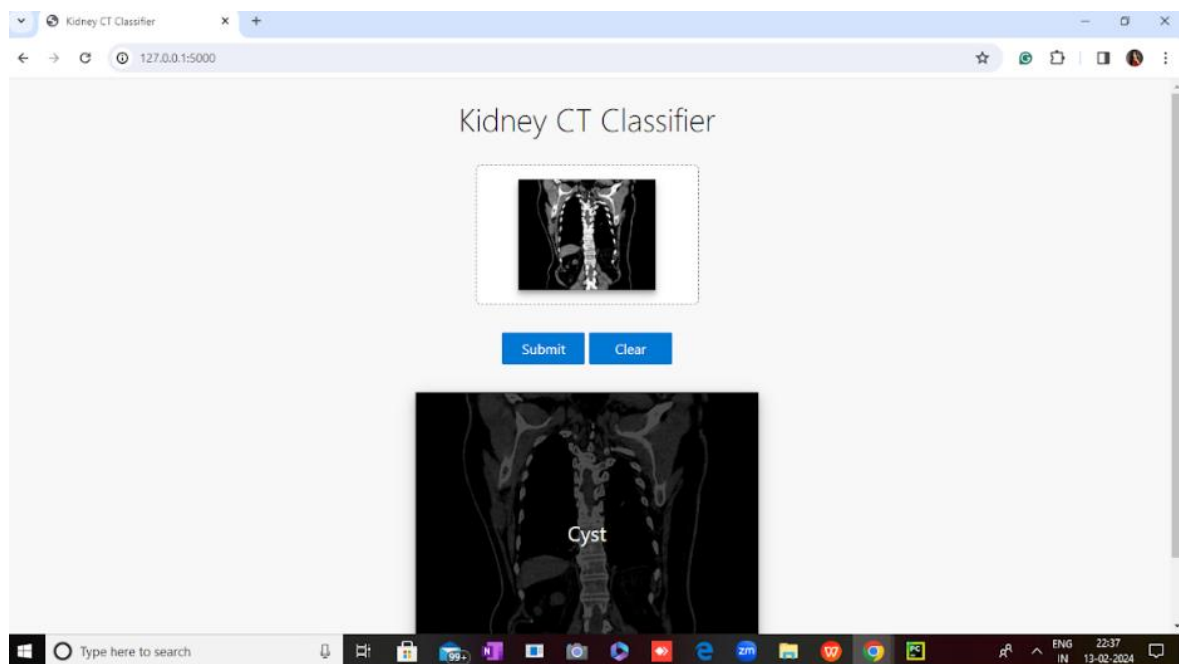
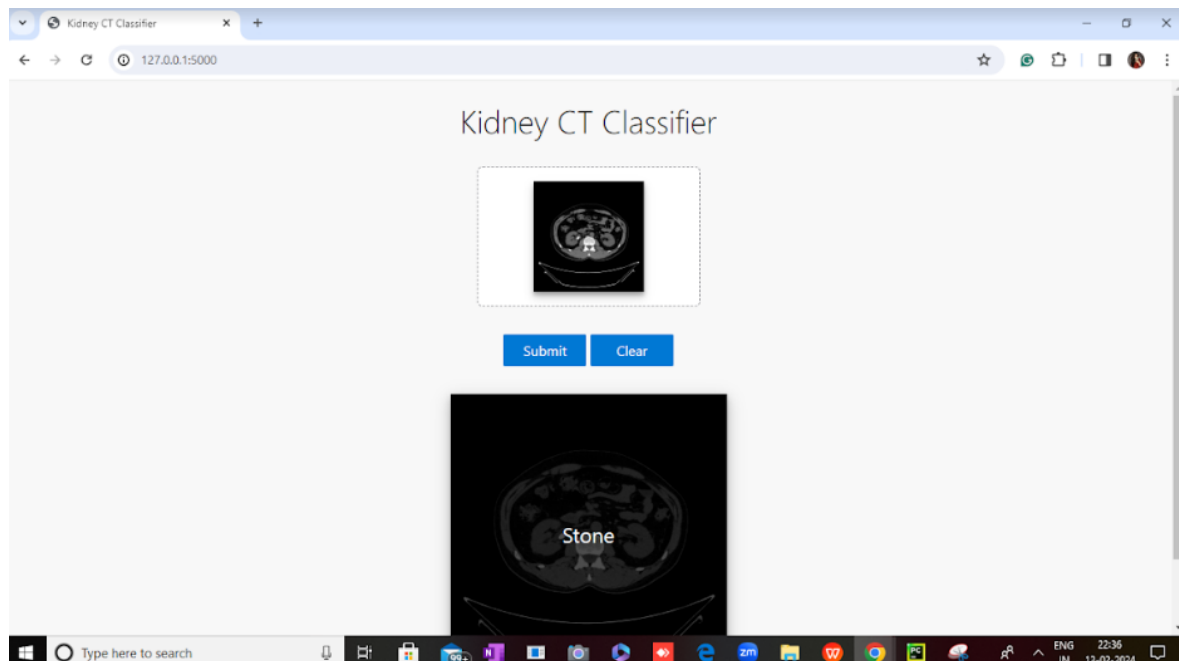
    # Serve the app with gevent

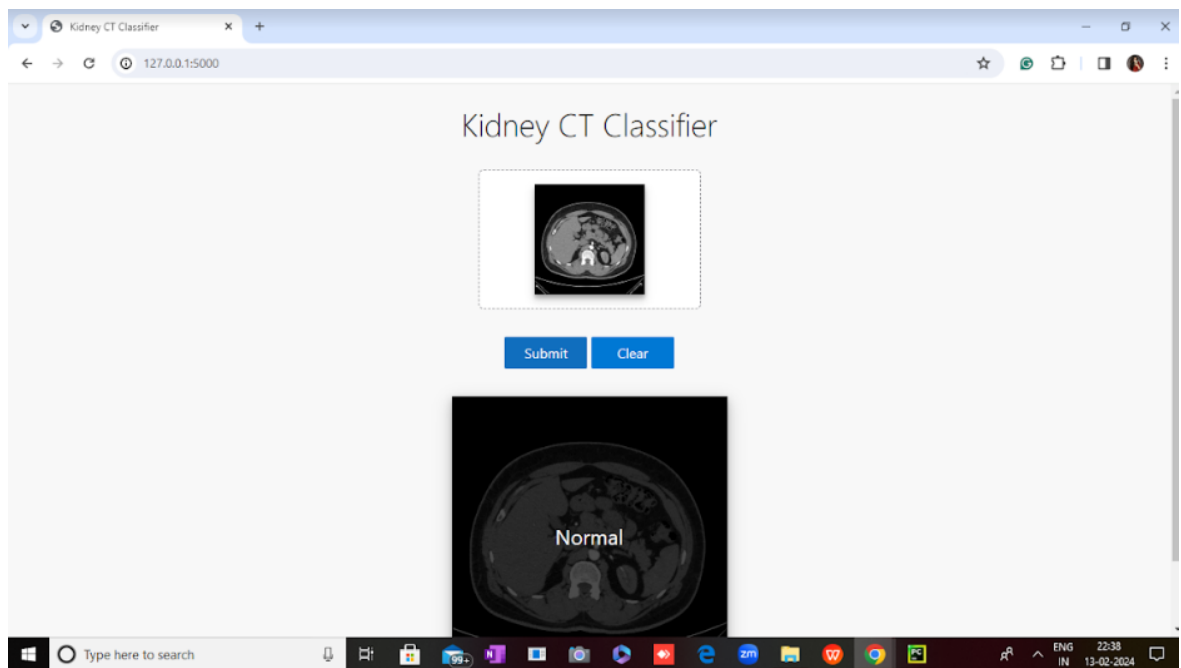
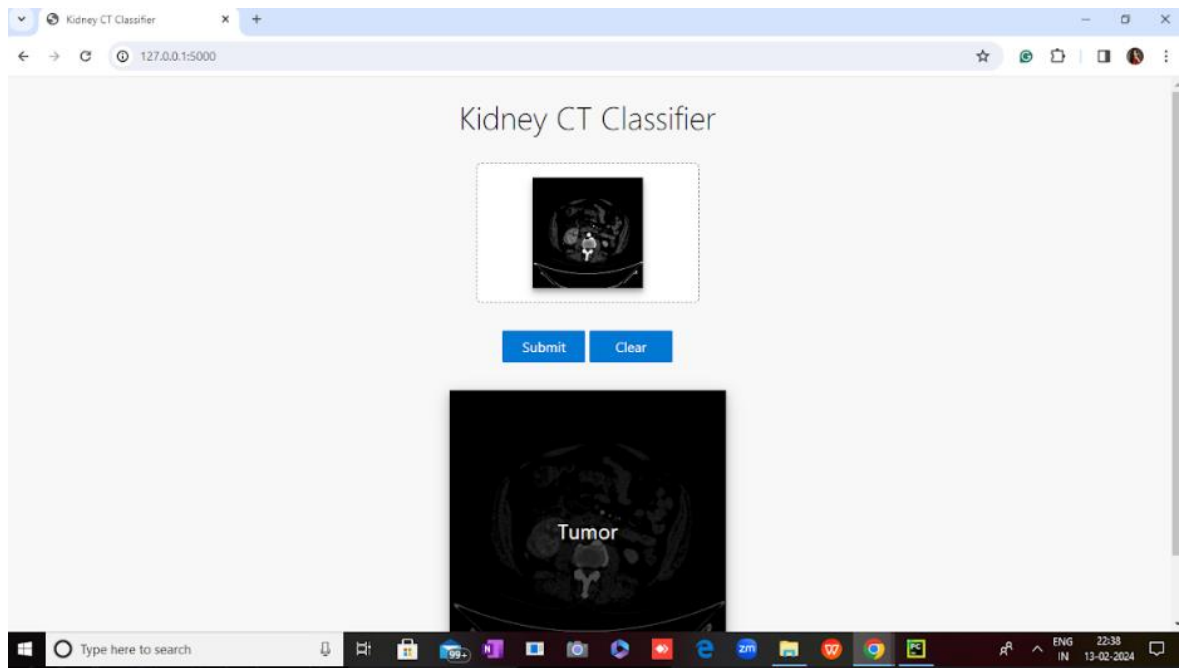
```

```
http_server = WSGIServer(('0.0.0.0', 5000), app)
```

```
http_server.serve_forever()
```

B.SCREENSHOT





B. CONFERENCE PROOF

NCRTC124

Inbox x



ncrtci <ncrtci2024@gmail.com>

to me

Apr 6, 2024, 9:50 PM (4 days ago)

☆ ☺ ↶ ⋮

Dear Author,
Greetings from The Department of IT, SRM Valliammai Engineering College.

Your paper is **ACCEPTED & RECOMMENDED** for publication in **National Conference on Recent Trends in Computational Intelligence 2024 [NCRTCI'24] 12-04-2024**.

Paper Title: **KIDNEY DISEASE CLASSIFIER USING CNN BASED DEEP LEARNING METHOD**
Paper ID: NCRTC124-80

Please ensure the following before upload the final paper:

1. Authors ensure that self-citations and irrelevant citations are removed from the final Manuscript.
2. Camera ready paper should strictly follow the IEEE format (**Sample paper format can be downloaded from [Click here](#)**)
3. The article has few typographical errors which may be carefully looked at.
4. Maximum 6 pages are allowed and Maximum authors count is 6.
5. Similarity index should be less than 15% and single source should be less than 5%.
6. Change your final manuscript (camera ready paper) name as your **PAPER ID**.
7. Presentation slot and the name of the publication journal will be intimated later.

Conference Date: **12, April 2024**
Last Date for Registration: **10, April 2024**