

VEGETABLE DISEASE PREDICTION USING DEEP LEARNING

Submitted in partial fulfilment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

P S B S V M KALYAN (37110201)

D S G V SYAM KUMAR (37110607)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600 119**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **P S B S V M KALYAN (37110201) and D S G V SYAM KUMAR (37110607)**

who carried out the project entitled "**VEGETABLE DISEASE PREDICTION USING DEEP LEARNING**" under my supervision from **August 2020 to March 2021**.

Internal Guide

Dr. L. Lakshmanan, M.E., Ph.D.,

Head of the Department

Dr. L. Lakshmanan, M.E., Ph.D.,
Dr. S. Vigneswari, M.E., Ph.D

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **P S B S V M Kalyan** hereby declare that the Project Report entitled “**Vegetable Disease Prediction Using Deep Learning**” is done by me under the guidance of **Dr. L. Lakshmanan, M.E., Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. L. Lakshmanan**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Agriculture is backbone of our country. As a farming country we need produce and transfer the vegetables. Manual sorting requires labours and it is time consuming. We need to identify the vegetables (or) agriculture products without damaging. The latest generation of convolutional neural networks (CNNs) has achieved impressive results in the field of image classification. This project is concerned with a new approach to the development of vegetable disease recognition model, based on leaf image classification, by the use of deep convolutional networks. All essential steps required for implementing this disease recognition model are fully described throughout the document starting from gathering images in order to create a database, assessed by agricultural experts. The experimental results on the developed model achieved precision between 81% and 90%, for separate class tests, on average 86.6%.

TABLE OF CONTENTS

	ABSTRACT	V
	LIST OF FIGURES	Viii
CHAPTER No.	TITLE	PAGE I .
1.	INTRODUCTION	1
	1.1 OVERVIEW	1
	1.2 DOMAIN OVERVIEW	1
2.	LITERATURE SURVEY	8
	2.1 LITERATURE SURVEY 1	8
	2.2 LITERATURE SURVEY 2	9
	2.3 LITERATURE SURVEY 3	10
	2.4 LITERATURE SURVEY 4	12
3.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	13
	3.1 AIM OF THE PROJECT	13
	3.1.1 OBJECTIVE	13
	3.2 SCOPE OF THE PROJECT	13
4.	SYSTEM ANALYSIS AND DESIGN AND IMPLEMENTATION METHODOLOGY	14
	4.1 EXISTING SYSTEM	14
	4.2 DRAWBACK OF EXISTING SYSTEM	14
	4.3 PROPOSED SYSTEM	14
	4.4 ADVANTAGES OF PROPOSED SYSTEM	14
	4.5 HARDWARE AND SOFTWARE SPECIFICATIONS	15
	4.5.1 HARDWARE SPECIFICATIONS	15
	4.5.2 SOFTWARE SPECIFICATIONS	15
	4.6 SOFTWARE DESCRIPTION	15

4.6.1 ABOUT PYTHON	15
4.6.2 PANDAS	17
4.6.3 NUMPY	19
4.6.4 MATHPLOTLIB	21
4.6.5 ABOUT IDE	22
4.7 SYSTEM DESIGN	23
4.7.1 ARCHITECTURE DIAGRAM	24
4.8 LIST OF MODULES	25
4.8.1 MODULE1- EXPLORATORY DATA EVALUATION	25
4.8.2 MODULE2- PRE-PROCESSING	26
4.8.3 MODULE3- FEATURE ENGINEERING	27
4.8.4 MODULE4- PREDICTION	27
5. RESULTS AND DISCUSSIONS	29
6. CONCLUSION AND FUTURE ENHANCEMENT	34
6.1 SUMMARY & CONCLUSION	34
6.2 FUTURE ENHANCEMENT	35
REFERENCES	36
APPENDIX	37
SOURCE CODE	37

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
4.1	PYTHON INSTALLATION ON SEVERAL OS	16
4.2	PYTHON LIBRARIES	16
4.3	ARCHITECTURE DIAGRAM	25
5.1	CONVERTING IMAGE TO ARRAYS	29
5.2	UNIQUE DETAILS OF DISEASES	29
5.3	DISEASED LEAFS	30
5.4	CONVERTING THE IMAGE LABELS TO BINARY USING SCIKIT- LEARN'S LABEL BINARIZER	30
5.5	SEQUENTIAL MODELS	31
5.6	TRAINING NETWORK	32
5.75.7	TRAINING AND VALIDATION ACCURACY	32
5.85.8	TRAINING AND VALIDATION LOSS & TEST ACCURACY	33

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Digital images are profoundly successful in passing on specific feature that assistance in specific assessments. In food science the way toward distinguishing the defect in a vegetable acts an essential job. So we propose a vegetable recognition algorithm to recognize the vegetables with the help of digital image processing technique.

1.2 DOMAIN OVERVIEW

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support

About Pandas

Pandas is a popular Python package for data science, and with good reason it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures. Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1. Series: a pandas Series is a one dimensional data structure ("a one dimensional ndarray") that can store values — and for every value it holds a unique index, too.
2. DataFrame: a pandas DataFrame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows

do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame
- a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.
- a NumPy ndarray, which can be a record or structured
- a two-dimensional ndarray
- dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very fast and efficient.

- There are tools available for loading data of different file formats into in-memory data objects.
- It is easy to perform data alignment and integrated handling of missing data in Python Pandas.
- It is very simple to perform pivoting and reshaping of data sets in Pandas.
- It also provides indexing, label-based slicing, and sub-setting of large data sets.
- We can easily insert and delete columns from a data structure.
- Data aggregation and transformations can be done using group by.
- High-performance merging and joining of data can be done using Pandas.
- It also provides time series functionality.

- Inserting and deleting columns in data structures.
- Merging and joining data sets.
- Reshaping and pivoting data sets.
- Aligning data and dealing with missing data.
- Manipulating data using integrated indexing for DataFrame objects.
- Performing split-apply-combine on data sets using the group by engine.
- Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
- Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
- Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
- Reading from files with CSV, XLSX, TXT, among other formats.
- Arranging data in an order ascending or descending.
- Filtering data around a condition.
- Analyzing time series.
- Iterating over a data set.

Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension. NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library

provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2- dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to

be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

Matplotlib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an

object oriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

CHAPTER2

LITERATURE SURVEY

2.1 Literature Survey 1

Title	Survey on Diseased Leaf Using Segmentation
Authors	Jothiaruna.N Joseph Abraham Sundar. K
Published Year	2019
Efficiency	Farmers were mainly focused on low cost and accuracy. Image processing is used to solve alFarmers were mainly focused on low cost and accuracy. Image processing is used to solve all those.
Drawbacks	problems with image segmentation, Color space Transformation, Morphological operation and many more techniques are used.
Description	Agriculture plays a vital role in all living things but it has many challenges to cultivate something like plants and vegetables from diseases. Disease affecting in the plant or leaf is quite a natural, if proper care is taken or not taken plant will surely affect by the disease because of nature, climate, soil condition and also by water. Here, we mainly focus on the detection of a diseased leaf. To solve those challenges many techniques are used like Color space transformation, Neural Network,

Preprocessing, Classification, Segmentation, and Extraction techniques are used. Any techniques can be used to detect disease from the leaf, but finding which can give the better solution from those techniques is hard because some techniques can give better result some may give low accuracy. In this paper, we mainly focus on Color Space transformation, Neural Network, Segmentation has many algorithms like region growing method and Morphological operation.

2.2 Literature Survey 2

Title	Leaves Diseases Detection of Tomato Using Image Processing
Authors	Tahmina Tashrif Mim
Published Year	2019
Efficiency	The angle rotates in counterclockwise direction which is controlled by the shear range.
Drawbacks	Adam algorithm has been used for reducing errors and value loss.
Description	Today's era is an era of Scientific Development. Where, technologies and new ways of solving real-life problems are being invented every day. With the increasing population of the world, basic need of food is increasing

parallelly. That's why agriculture plays an important role all over the world. Throughout the year different crops, vegetables, fruits, fishes, animals are cultivated to fulfill the need of people as well as to gain profit for the people involving in those cultivation. But due to lack of proper cultivating knowledge, experience and sense of disease prediction, sometimes those cultivating crops and grains get damaged partially or even completely. Of course, that ends up with a huge loss for the farmers as well as for the economic growth of the country. So, this research paper tends to merge or combine a part of agricultural sector with science and technology to reduce the loss caused by insect's attack and diseases of plant leaves. More specifically, this research happens to combine agricultural sector with computer science. Since, agriculture is a vast sector to work on, to simplify the work, we are detecting vegetable plant diseases using Artificial Intelligence and computer science.

2.3 Literature Survey 3

Title	Weed growth and intrusion detection by multi angle images of vegetable plants using modified SVD
Authors	K.Ramesh

Published Year	2017
Efficiency	a desired advantage in real time image based monitoring of plants, weeds and intrusion in the plants.
Drawbacks	the framer having trouble expelling parrots, Peacocks ,grasshoppers, hare and wild boar from plants like maize,chilli,radish and Tapioca.
Description	The requirement of earlier identification of weed growth and plant disease is one of the most important procedures in precision agriculture. In this research work we identified the best method of positioning the optical sensors to get the best possible images. To address this purpose the variant of SVD is used for feature construction. Our proposed approach optimize and reiterate our similar earlier findings in identification of plants. The threshold augmentation technique is used to find the inter and intra average differences for both aerial and portrait angle images for comparison. The intrusion detection software matlab is created and tested for the Portrait images and compared with aerial images. The Portrait images to find the intrusion are found better than aerial images in number of trails 15% more than the aerial images.

2.4 Literature Survey 4

Title	Molecular Diagnosis of Gray Mould Pathogen On Fruits And Vegetables.
Authors	Yun-Feng
Published Year	2010
Efficiency	The Bayesian statistical procedure used to create the gene expression-based gray mould pathogen predictor has been described
Drawbacks	Currently, no molecular tests are routinely available for the diagnosis of gray mould pathogen.
Description	Molecular methods used directly on clinical material play an important role in the diagnosis and gray mould pathogen. Molecular diagnostic tests that allow timely and accurate detection of pathogen are already implemented in many laboratories. The intent of this review is that the reader will better understand the variety of technical approaches to the molecular genetic testing for the gray mould pathogen. The early molecular diagnosis of gray mould pathogen should significantly aid in the development of therapies tailored to this clinically and pathogenetically distinctive subgroup of bio-prevention.

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 AIM:

The Aim of this project is to develop an efficient vegetable recognition algorithm to recognize the vegetables with the help of digital image processing technique.

Vegetable disease recognition in the complex environment mainly focuses on three aspects: disease leaves image segmentation, feature extraction, and disease identification.

3.1.1. OBJECTIVE

The main purpose of doing this project is:

- Building user-friendly application where user can easier interact easily.
- User can easily identify the diseased vegetable from the predefined data set
- It is also used to build a real time project.

3.2 SCOPE OF THE PROJECT:

The main contributions of this project therefore are:

- Data Analysis
- Dataset Preprocessing
- Training the Model
- Testing of Dataset

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN AND IMPLEMENTATION METHODOLOGY

4.1 EXISTING SYSTEM

Existing system use the spectral features of high spatial resolution data combined with multitemporal characteristics and some auxiliary features (such as elevation information and textural information) to improve classification accuracy

4.2 DRAWBACKS OF EXISTING SYSTEM

- High-resolution data source requirements for crop identification.
- Overfitting problem in multidimensional feature data classification.
- Not suitable for large regions.
- Time consuming

4.3 PROPOSED SYSTEM

Bacterial illnesses of vegetables are most extreme and unhelpful infections impacting in field crops. Under most conditions they can cause confined pandemics impacting young rising vegetable.

The exact detection of vegetable illness is request undertaking task to specialists. This work presents vegetable disease detection using image processing techniques to monitor diseases. The objective of this project is to detect the disease in vegetables efficiently

4.4 ADVANTAGES OF PROPOSED SYSTEM

- Accuracy is more than 90%
- More reliable
- Low computational power is enough
- We provide solution for disease

4.5 HARDWARE & SOFTWARE SPECIFICATION

4.5.1 Hardware Requirements:

Hardware Minimum Requirements

Disk Space	32 GB or more,10 GB or more for Foundation Edition
Processor	1.4 GHz 64 bit
Memory	512 MB
Display	(800 × 600) Capable video adapter and monitor

4.5.2 Software Specification

Backend Technologies

Python

Numpy

Sci-learn

Eclipse IDE

Frontend Technologies

Web Technologies

Bootstrap

4.6 SOFTWARE DESCRIPTION

4.6.1 About Python

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems.

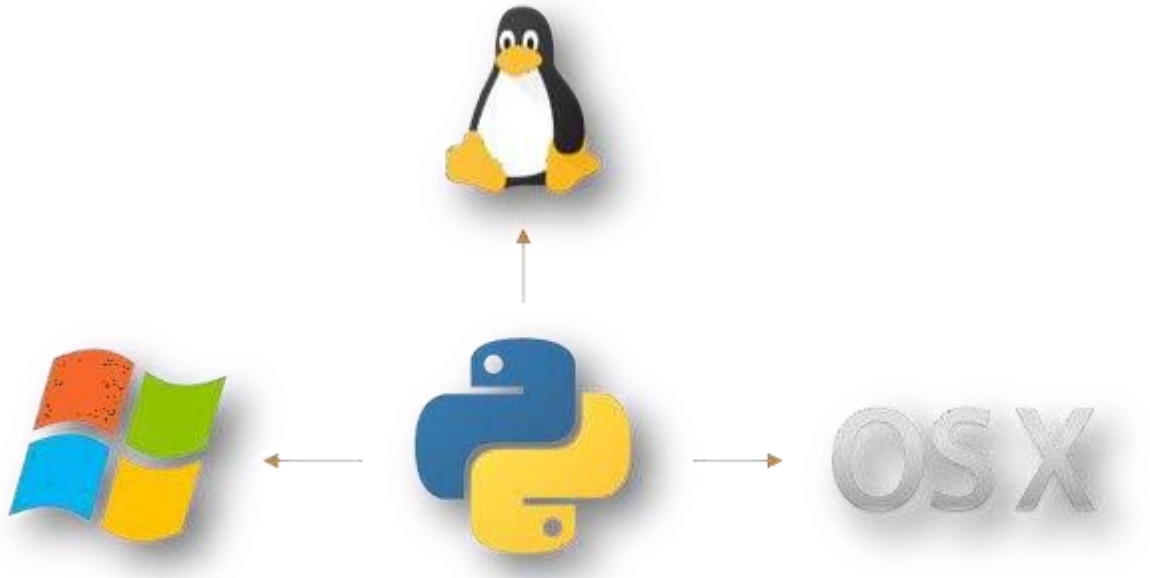


Fig 4.1. Python Installation on Several OS

Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.



Fig 4.2 Python Libraries

In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

Advantages of Python

- Universal Language Construct

- Support both High Level and Low Level Programming
- Language Interoperability
- Fastest Development life cycle therefore more productive coding environment
- Less memory used because a single container hold multiple data types and each type doesn't require its own function
- Learning Ease and open source development
- Speed and user-friendly data structure
- Extensive and extensible libraries.
- Simple & support IoT
- And many more

4.6.2 Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to

visualize pandas' DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1. Series: a pandas Series is a one dimensional data structure (“a one dimensional ndarray”) that can store values — and for every value it holds a unique index, too.
2. DataFrame: a pandas DataFrame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame
- a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.
- a NumPy ndarray, which can be a record or structured
- a two-dimensional ndarray

- dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very fast and efficient.

- There are tools available for loading data of different file formats into in-memory data objects.
- It is easy to perform data alignment and integrated handling of missing data in Python Pandas.
- It is very simple to perform pivoting and reshaping of data sets in Pandas.
- It also provides indexing, label-based slicing, and sub-setting of large data sets.
- We can easily insert and delete columns from a data structure.

4.6.3 Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to

be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

4.6.4 *Mathplotlib*

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms

4.6.5 About IDE

IDE – Integrated Development Environment. It is software suite. It contains different instruments for web application software engineers to compose and test applications.

Programming Web application software engineers utilize wide scope of devices habitually all through application source code creation, building and application testing. Programming Development apparatuses frequently incorporate word processors for source code creation, code libraries for reuse, compilers and test stages.

An Integrated Development Environment (IDE) is a product application that gives a programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a familiarity with the earth, and they cooperate to display a consistent improvement set for the designer.

An IDE typically has an editor and a compiler window. There, we can write our code, run it, and compile it. Hence, it becomes much easier to develop applications using an IDE. The major reasons for using an IDE are given below:

- ▶ We can easily write our code in its text editor window.
- ▶ To store the resource files of a project, we have a project editor window.
- ▶ It makes debugging of the code easier.
- ▶ An IDE's autosuggest feature helps us in writing efficient code.
- ▶ We can import other projects for integrating with the existing one.

Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration. The incorporated toolset is intended to streamline programming advancement and can distinguish and limit coding missteps and grammatical errors.

- ✓ Komodo
- ✓ NetBeans
- ✓ PyCharm
- ✓ Python for Visual Studio.NET
- ✓ PyDev
- ✓ PyStudio
- ✓ LiClipse (Eclipse)

4.7 SYSTEM DESIGN

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Mainly, UML has been used as a general-purpose modeling language in the field

of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency.

4.7.1 Architecture Diagram

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components.

What is a diagram? What are the types of diagrams for architecture? The Dragon1 open EA Method makes it very clear: if a diagram does not show a concept, principle or part of a principle, it is NOT an architecture diagram, because it does not show (a part of) the architecture.

There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc.

For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at the system planning phase helping partners understand the architecture, discuss changes, and communicate intentions clearly.

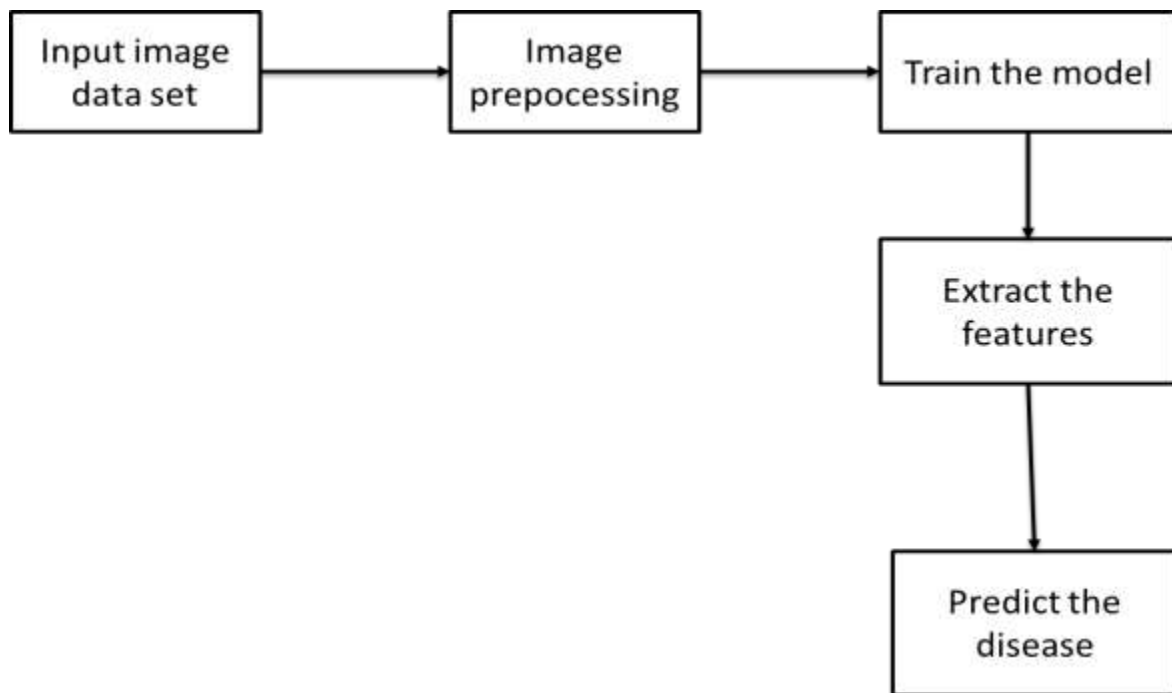


Fig 4.3. Architecture Diagram

4.8 LIST OF MODULES

Module 1: Exploratory Data Evaluation

Module 2: Pre-processing

Module 3: Feature Engineering

Module 4: Prediction

4.8.1 Module 1-Exploratory Data Evaluation

Exploratory Data Analysis (EDA) is the first step in your data analysis process. Here, you make sense of the data you have and then figure out what questions you want to ask and how to frame them, as well as how best to manipulate your available data sources to get the answers you need. You do this by taking a broad look at patterns, trends, outliers, unexpected results and so on in your existing data, using visual and quantitative methods to get a sense of the story this tells.

Exploratory Data Analysis is valuable to data science projects since it allows to get closer to the certainty that the future results will be valid, correctly interpreted, and

applicable to the desired business contexts. Such level of certainty can be achieved only after raw data is validated and checked for anomalies, ensuring that the data set was collected without errors. EDA also helps to find insights that were not evident or worth investigating to business stakeholders and data scientists but can be very informative about a particular business.

EDA is performed in order to define and refine the selection of feature variables that will be used for machine learning. Once data scientists become familiar with the data set, they often have to return to feature engineering step, since the initial features may turn out not to be serving their intended purpose. Once the EDA stage is complete, data scientists get a firm feature set they need for supervised and unsupervised machine learning.

4.8.2 Module 2-Pre-processing

Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common idea to handle the problem is to take a mean of all the values of the same column and have it to replace the missing data.

The library that we are going to use for the task is called Scikit Learn preprocessing. It contains a class called Imputer which will help us take care of the missing data.

Sometimes our data is in qualitative form, that is we have texts as our data. We can find categories in text form. Now it gets complicated for machines to understand texts and process them, rather than numbers, since the models are based on mathematical equations and calculations. Therefore, we have to encode the categorical data.

Now we need to split our dataset into two sets — a Training set and a Test set. We will train our machine learning models on our training set, i.e our machine learning models will try to understand any correlations in our training set and then we will test the models on our test set to check how accurately it can predict. A general rule of the thumb is to allocate 80% of the dataset to training set and the remaining 20% to test set. For this task, we will import `test_train_split` from `model_selection` library of `scikit`.

4.8.3 Module 3 : Feature Engineering

Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. The correlation is a subjective term here. For basic guidance, you can refer to the following table for defining correlation co-efficients.

Pearson's Correlation: It is used as a measure for quantifying linear dependence between two continuous variables X and Y. Its value varies from -1 to +1.

LDA: Linear discriminant analysis is used to find a linear combination of features that characterizes or separates two or more classes (or levels) of a categorical variable.

ANOVA: ANOVA stands for Analysis of variance. It is similar to LDA except for the fact that it is operated using one or more categorical independent features and one continuous dependent feature. It provides a statistical test of whether the means of several groups are equal or not.

Chi-Square: It is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution

4.8.4 Module 4-Prediction

Once training is complete, it's time to see if the model is any good, using Evaluation. This is where that dataset that we set aside earlier comes into play. Evaluation allows us to test our model against data that has never been used for training. This metric allows us to see how the model might perform against data that it has not yet seen. This is meant to be representative of how the model might perform in the real world.

A good rule of thumb I use for a training-evaluation split somewhere on the order of 80/20 or 70/30. Much of this depends on the size of the original source dataset. If you have a lot of data, perhaps you don't need as big of a fraction for the evaluation dataset.

Once you've done evaluation, it's possible that you want to see if you can further improve your training in any way. We can do this by tuning our parameters. There were a few parameters we implicitly assumed when we did our training, and now is a good time to go back and test those assumptions and try other values.

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn't split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.

Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can't ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or died. Regression trees are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

CHAPTER 5

RESULT AND DISCUSSIONS

we picked just 200 images from each folder but you can choose to add more we converted each image to an array using the function below. After converting each image to an array using this same function, you should have something similar to what we have in the image for each image.

```
def convert_image_to_array(image_dir):
    try:
        image = cv2.imread(image_dir)
        if image is not None :
            image = cv2.resize(image, default_image_size)
            return img_to_array(image)
        else :
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None
```

```
[[ 97.  94. 109.]
 [ 91.  88. 103.]
 [122. 119. 134.]
 ...
 [149. 148. 164.]
 [154. 153. 169.]
 [164. 163. 179.]]

[[126. 123. 138.]
 [101.  98. 113.]
 [118. 115. 130.]
 ...
 [149. 148. 164.]
 [148. 147. 163.]
 [151. 150. 166.]]

[[135. 132. 147.]
 [ 87.  84.  99.]
 [142. 139. 154.]
 ...
 [153. 152. 168.]
 [147. 146. 162.]
 [145. 144. 160.]]
...
```

Fig 5.1 Converting image to arrays

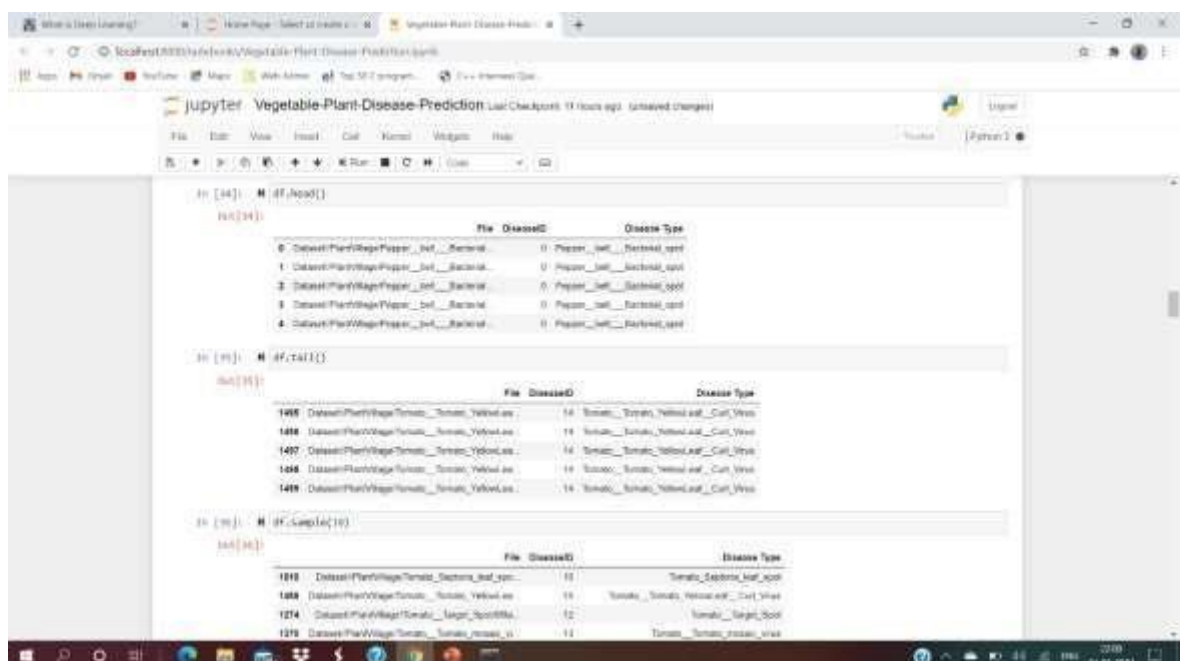


Fig 5.2 Unique details of diseases

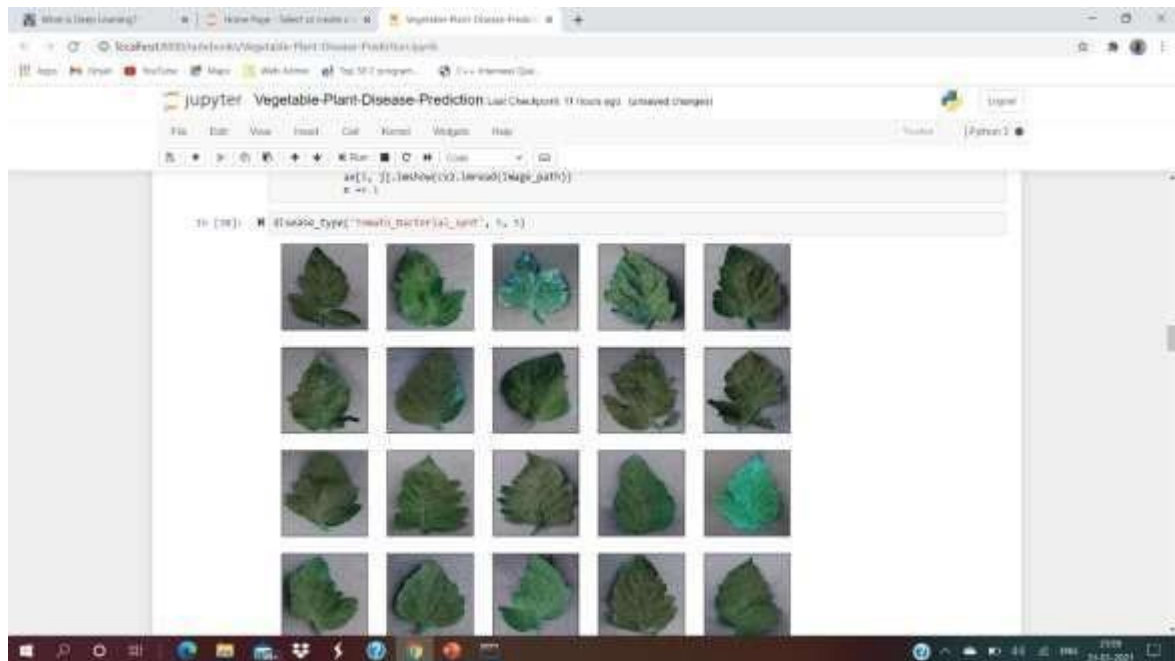


Fig 5.3 Diseased Leaf

Using Scikit-learn's Label Binarizer , we converted each image label to binary levels. Then we saved the label binarizer instance using pickle after which we printed the classes from the label binarizer.

```
In [47]: print(label_binarizer.classes_)

['Pepper_bell_Bacterial_spot' 'Pepper_bell_healthy'
 'Potato_Early_blight' 'Potato_Late_blight' 'Potato_healthy'
 'Tomato_Bacterial_spot' 'Tomato_Early_blight' 'Tomato_Late_blight'
 'Tomato_Leaf_Mold' 'Tomato_Septoria_leaf_spot'
 'Tomato_Spider_mites_Two_spotted_spider_mite' 'Tomato_Target_Spot'
 'Tomato_Tomato_YellowLeaf_Curl_Virus' 'Tomato_Tomato_mosaic_virus'
 'Tomato_healthy']
```

Fig 5.4 Converting the image labels to binary using Scikit-learn's Label Binarizer

we further pre-process the input data by scaling the data points from [0, 255] (the minimum and maximum RGB values of the image) to the range [0, 1]. we then performed a training/testing split on the data using 80% of the images for training and 20% for testing. we created an image generator object which performs random rotations, shifts, flips, crops, and sheers on our image dataset. This allows us to

use a smaller dataset and still achieve high results. Next we created our model. In the model we are defaulting to “channel_last” architecture but also creating a switch for backends that support “channel_first” on the fourth line. Then we created the first CONV => RELU => POOL. Our CONV layer has 32 filters with a 3 x 3 kernel and RELU activation (Rectified Linear Unit). We apply batch normalization, max pooling, and 25% (0.25) dropout.

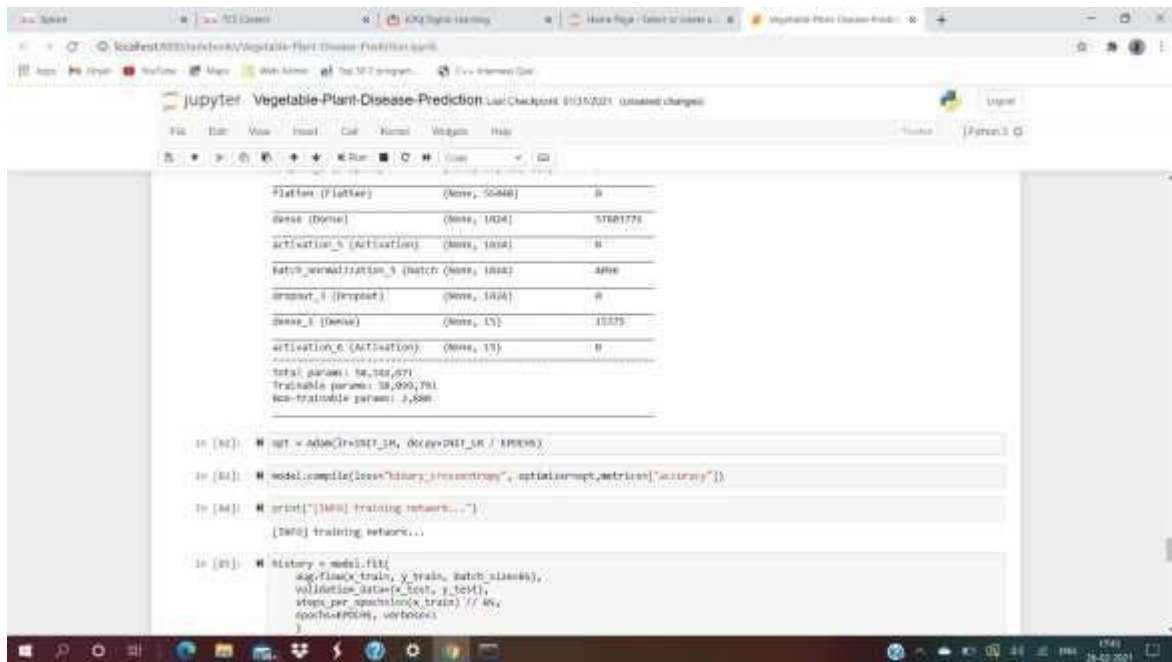


Fig 5.5 Sequential Models

we used the Keras Adam Optimizer for my model. Training our network, where we call `model.fit_generator`, supplying our data augmentation object, training/testing data, and the number of epochs we wish to train for. I used an epochs value of 5 in the below picture.

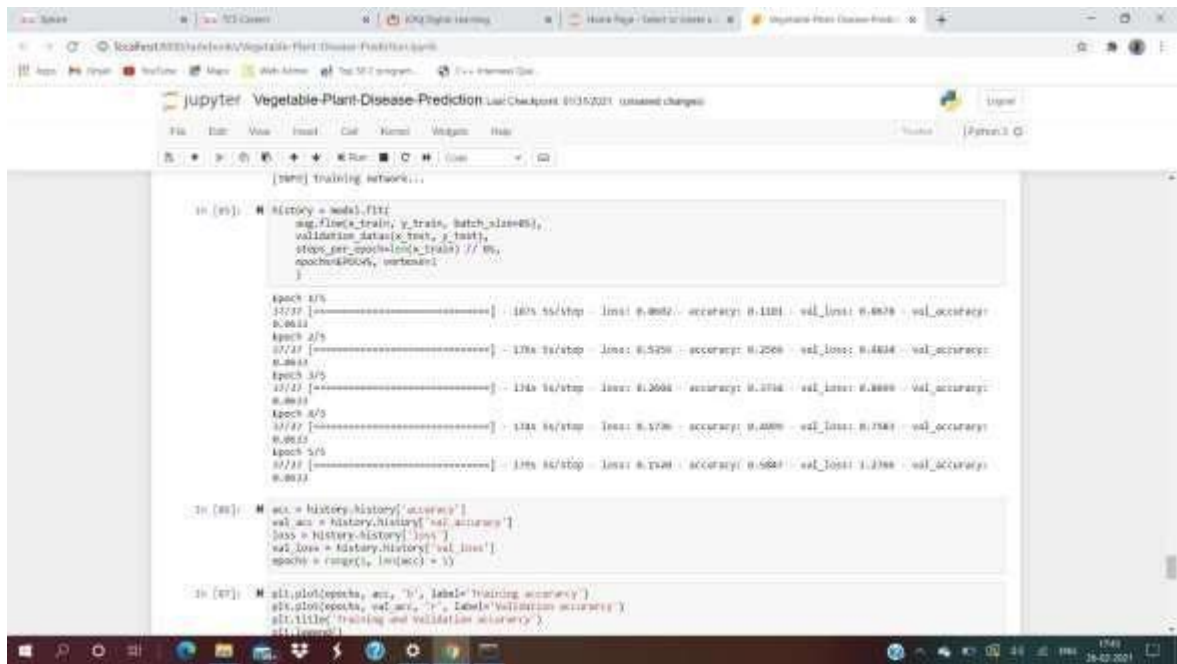


Fig 5.6 Training Network

Graph Plotting — Using matplotlib, we plotted a graph for Training and Validation accuracy and Training and Validation loss

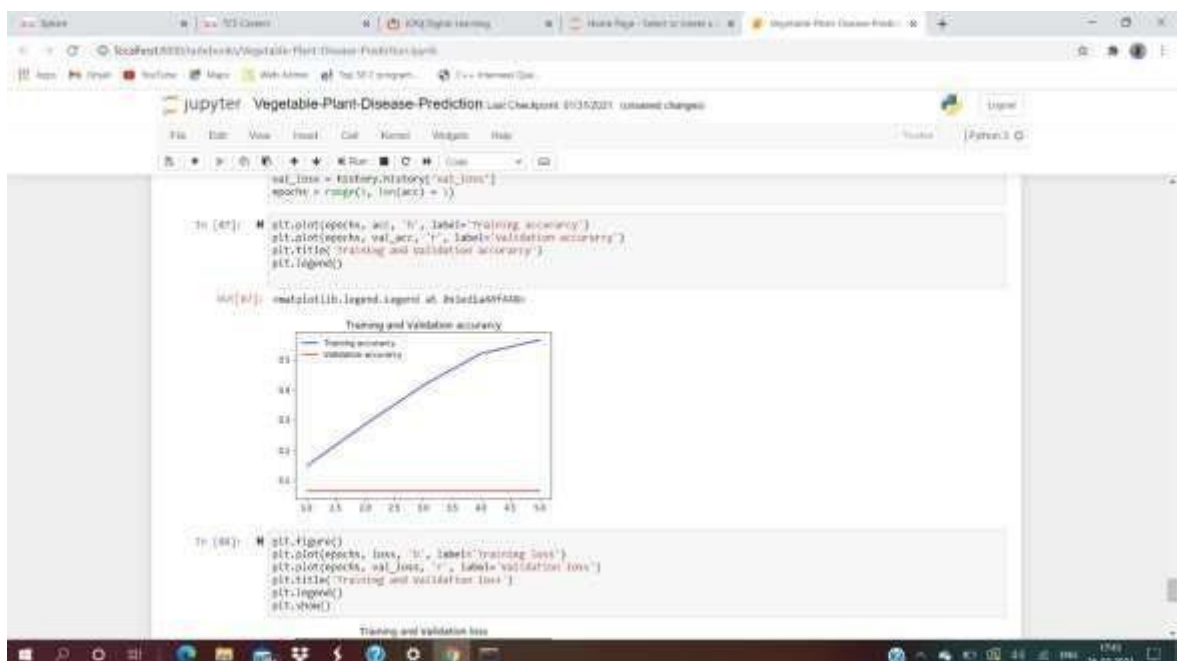


Fig 5.7 Training and validation accuracy

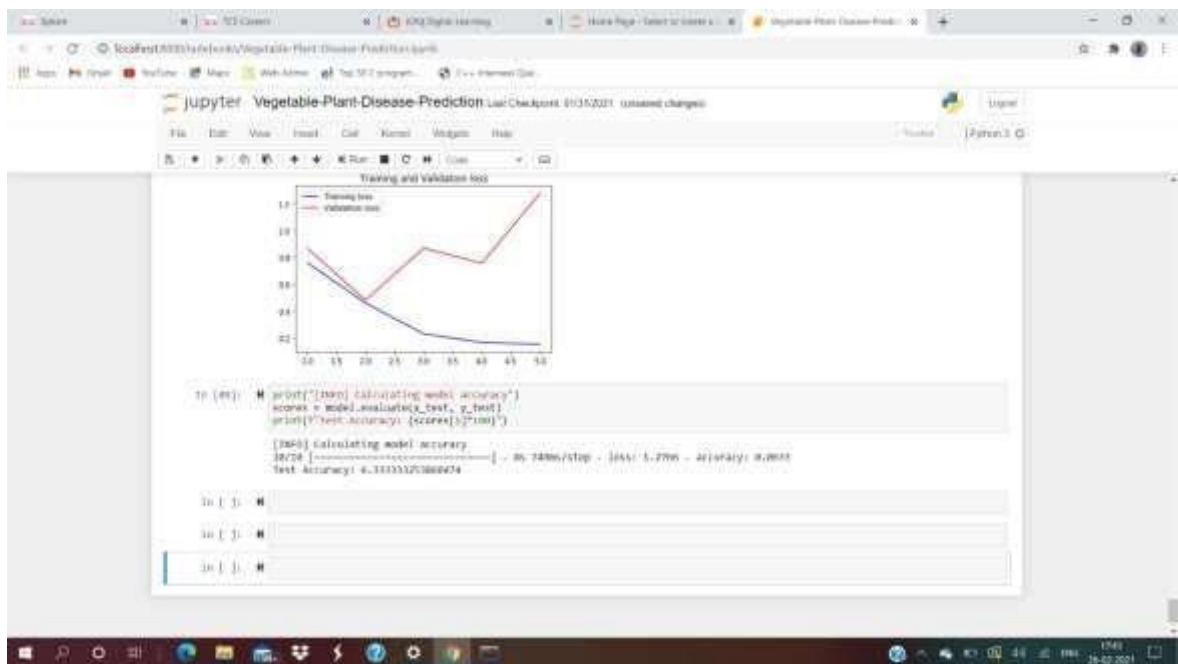


Fig 5.8 Training and validation loss & test accuracy

Lastly we calculated the accuracy of our model using the validation data (x_test and y_test) created earlier. we got an accuracy score as per the epoch value . And we saved the trained model for further use using Pickle.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 SUMMARY & CONCLUSION

There are many methods in automated or computer vision plant disease detection and classification process, but still, this research field is lacking. In addition, there are still no commercial solutions on the market, except those dealing with plant species recognition based on the leaves images.

In this project, a new approach of using deep learning method was explored in order to automatically classify and detect plant diseases from leaf images. The developed model was able to detect leaf presence and distinguish between healthy leaves and 13 different diseases, which can be visually diagnosed. The complete procedure was described, respectively, from collecting the images used for training and validation to image preprocessing and augmentation and finally the procedure of training the deep CNN and fine-tuning. Different tests were performed in order to check the performance of newly created model.

New plant disease image database was created, containing more than 3,000 original images taken from the available Internet sources and extended to more than 30,000 using appropriate transformations. The experimental results achieved precision between 81% and 90%, for separate class tests. The final overall accuracy of the trained model was 86.6%. Fine-tuning has not shown significant changes in the overall accuracy, but augmentation process had greater influence to achieve respectable results. An extension of this study will be on gathering images for enriching the database and improving accuracy of the model using different techniques of fine-tuning and augmentation.

The main goal for the future work will be developing a complete system consisting of server side components containing a trained model and an application for smart mobile devices with features such as displaying recognized diseases in fruits, vegetables, and other plants, based on leaf images captured by the mobile phone camera. This application will serve as an aid to farmers (regardless of the level of experience), enabling fast and efficient recognition of plant diseases and facilitating the decision-making process when it comes to the use of chemical

pesticides.

Furthermore, future work will involve spreading the usage of the model by training it for plant disease recognition on wider land areas, combining aerial photos of orchards and vineyards captured by drones and convolution neural networks for object detection. By extending this research, the authors hope to achieve a valuable impact on sustainable development, affecting crop quality for future generations

6.2 FUTURE ENHANCEMENT

In future, we like to implement real time Vegetable Disease prediction using IoT

REFERENCES

1. K. A. Garrett, S. P. Dendy, E. E. Frank, M. N. Rouse, and S. E. Travers, "Climate change effects on plant disease: genomes to ecosystems," *Annual Review of Phytopathology*, vol. 44, pp. 489–509, 2006.
2. S. M. Coakley, H. Scherm, and S. Chakraborty, "Climate change and plant disease management," *Annual Review of Phytopathology*, vol. 37, no. 1, pp. 399–426, 1999.
3. S. Chakraborty, A. V. Tiedemann, and P. S. Teng, "Climate change: potential impact on plant diseases," *Environmental Pollution*, vol. 108, no. 3, pp. 317–326, 2000.
4. A. J. Tatem, D. J. Rogers, and S. I. Hay, "Global transport networks and infectious disease spread," *Advances in Parasitology*, vol. 62, pp. 293–343, 2006.
5. J. R. Rohr, T. R. Raffel, J. M. Romansic, H. McCallum, and P. J. Hudson, "Evaluating the links between climate, disease spread, and amphibian declines," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 45, pp. 17436–17441, 2008.
6. T. Van der Zwet, "Present worldwide distribution of fire blight," in *Proceedings of the 9th International Workshop on Fire Blight*, vol. 590, Napier, New Zealand, October 2001.
7. P. R. Reddy, S. N. Divya, and R. Vijayalakshmi, "Plant disease detection techniquetool—a theoretical approach," *International Journal of Innovative Technology and Research*, pp. 91–93, 2015.
8. A.-K. Mahlein, T. Rumpf, P. Welke et al., "Development of spectral indices for detecting and identifying plant diseases," *Remote Sensing of Environment*, vol. 128, pp. 21–30, 2013.

APPENDIX – SOURCE CODING

```
import numpy as np
import pandas as pd
import pickle
import cv2
import os
from os import listdir
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Activation, Flatten, Dropout, Dense
from tensorflow.keras import backend as K
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
EPOCHS = 2
INIT_LR = 1e-3
BS = 32
default_image_size = tuple((256, 256))
image_size = 0
directory_root = 'Dataset/'
width=256
height=256
depth=3
def convert_image_to_array(image_dir):
```

```

try:
    image = cv2.imread(image_dir)
    if image is not None :
        image = cv2.resize(image, default_image_size)
        return img_to_array(image)
    else :
        return np.array([])
except Exception as e:
    print(f"Error : {e}")
    return None

image_list, label_list = [], []
eda_data = []
disease_type = []

try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
    for directory in root_dir :
        if directory == ".DS_Store" :
            root_dir.remove(directory)

    for plant_folder in root_dir :
        plant_disease_folder_list = listdir(f"{directory_root}/{plant_folder}")

        for disease_folder in plant_disease_folder_list :
            # remove .DS_Store from list
            if disease_folder == ".DS_Store" :
                plant_disease_folder_list.remove(disease_folder)
        for idx, plant_disease_folder in enumerate(plant_disease_folder_list):
            print(f"[INFO] Processing {plant_disease_folder} ...")
            plant_disease_image_list =
listdir(f"{directory_root}/{plant_folder}/{plant_disease_folder}")

            for single_plant_disease_image in plant_disease_image_list :

```

```

        if single_plant_disease_image == ".DS_Store" :
            plant_disease_image_list.remove(single_plant_disease_image)
        for image in plant_disease_image_list[:100]:
            disease_type.append(plant_disease_folder)
            image_directory =
f'{directory_root}/{plant_folder}/{plant_disease_folder}/{image}'
            if image_directory.endswith(".jpg") == True or
image_directory.endswith(".JPG") == True:
                eda_data.append([image_directory, idx, plant_disease_folder])
                image_list.append(convert_image_to_array(image_directory))
                label_list.append(plant_disease_folder)
            print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")
df = pd.DataFrame(eda_data, columns=['File', 'DiseaseID', 'Disease Type'])
df.head()
df.tail()
df.sample(10)
def disease_type(disease_type, rows, cols):
    fig, ax = plt.subplots(rows, cols, figsize=(12,12))
    disease_type = df['File'][df['Disease Type'] == disease_type].values
    n = 0
    for i in range(rows):
        for j in range(cols):
            image_path = os.path.join("", disease_type[n])
            ax[i, j].set_xticks([])
            ax[i, j].set_yticks([])
            ax[i, j].imshow(cv2.imread(image_path))
            n += 1
disease_type('Tomato_Bacterial_spot', 5, 5)
disease_type('Tomato_healthy', 5, 5)
disease_type('Potato___Late_blight', 5, 5)
disease_type('Pepper_bell_Bacterial_spot', 5, 5)
image_size = len(image_list)

```

```

print(image_size)
label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)
pickle.dump(label_binarizer,open('label_transform.pkl', 'wb'))
n_classes = len(label_binarizer.classes_)
print(label_binarizer.classes_)
np_image_list = np.array(image_list, dtype=np.float16) / 255.0
print("[INFO] Splitting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels,
test_size=0.2, random_state = 42)
aug = ImageDataGenerator(
    rotation_range=25, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2,
    zoom_range=0.2, horizontal_flip=True,
    fill_mode="nearest")
model = Sequential()
inputShape = (height, width, depth)
chanDim = -1
if K.image_data_format() == "channels_first":
    inputShape = (depth, height, width)
    chanDim = 1
model.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

```

```

model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))
model.summary()
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])
print("[INFO] training network...")
history = model.fit(
    aug.flow(x_train, y_train, batch_size=BS),
    validation_data=(x_test, y_test),
    steps_per_epoch=len(x_train) // BS,
    epochs=EPOCHS, verbose=1
)
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'b', label='Training accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and Validation accuracy')
plt.legend()

```

```
plt.figure()
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
print("[INFO] Calculating model accuracy")
scores = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")
```