CHAPTER 1

INTRODUCTION

1.1 Problem statement:

• Android Apps are freely available on Google Playstore, the official Android app store as well as third-party app stores for users to download. Due to its open source nature and popularity, malware writers are increasingly focusing on developing malicious applications for Android operating system. In spite of various attempts by Google Playstore to protect against malicious apps, they still find their way to mass market and cause harm to users by misusing personal information related to their phone book, mail accounts, GPS location information and others for misuse by third parties or else take control of the phones remotely. Therefore, there is need to perform malware analysis or reverse-engineering of such malicious applications which pose serious threat to Android platforms. Broadly speaking, Android Malware analysis is of two types: Static Analysis and Dynamic Analysis. Static analysis basically involves analyzing the code structure without executing it while dynamic analysis is examination of the runtime behavior of Android Apps in constrained environment. Given in to the ever-increasing variants of Android Malware posing zero-day threats, an efficient mechanism for detection of Android malwares is required. In contrast to signature-based approach which requires regular update of signature database.

1.2 Motivation:

In this paper author is using two machine learning algorithms such as SVM (Support Vector Machine) and NN (Neural Networks). App will contains more than 100 features and machine learning will take more time to build model so we need to optimized (reduce dataset columns size) features, to optimized features author is using genetic algorithm. Genetic algorithm will choose important features from dataset to train model and remove un-important features. Due to this process dataset size will be reduced and training model will be generated faster. In this paper comparison we are losing some accuracy after applying genetic algorithm but we are able to reduce model training execution time.

1.3 Objective:

• Android is an open source free operating system and it has support from Google to publish android application on its Play Store. Anybody can developed an android app and publish on play store free of

cost. This android feature attract cyber-criminals to developed and publish malware app on play store. If anybody install such malware app then it will steal information from phone and transfer to cyber-criminals or can give total phone control to criminal's hand. To protect users from such app in this paper author is using machine learning algorithm to detect malware from mobile app. To detect malware from app we need to extract all code from app using reverse engineering and then check whether app is doing any mischievous activity such as sending SMS or copying contact details without having proper permissions. If such activity given in code then we will detect that app as malicious app. In a single app there could be more than 100 permissions (examples of permissions are transact, API call signature, onServiceConnected, API call signature, bindService, API call signature, attachInterface, API call signature, ServiceConnection, API call signature, android.os.Binder, API call signature, SEND_SMS, Manifest Permission, Ljava.lang.Class.getCanonicalName, API call signature etc.) which we need to extract from code and then generate a features dataset, if app has proper permission then we will put value 1 in the features data and if not then we will value 0. Based on those features dataset app will be mark as malware or good ware.

1.3.1 ProposedSystem:

Two set of Android Apps or APKs: Malware/Goodware are reverse engineered to extract features such as permissions and count of App Components such as Activity, Services, Content Providers, etc. These features are used as featurevector with class labels as Malware and Goodware represented by 0 and 1 respectively in CSV format.

To reduce dimensionality of feature-set, the CSV is fed to Genetic Algorithm to select the most optimized set of features. The optimized set of features obtained is used for training two machine learning classifiers: Support Vector Machine and Neural Network.

In the proposed methodology, static features are obtained from AndroidManifest.xml which contains all the important information needed by any Android platform about the Apps. Androguard tool has been used for disassembling of the APKs and getting the static features.

1.3.2 Advantages of proposed system:

Security

Proposed a novel and efficient algorithm for feature selection to improve overall detection accuracy.

Machine-learning based approach in combination with static and dynamic analysis can be used to detect new variants of Android Malware posing zero-day threats.

CHAPTER- 2
TECHNOLOGIES LEARNT

2.1 What is Python :-

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

2.2 Advantages of Python :-

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it *contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more.* So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print **'Hello World'**. But in Python, just a print statement will do. It is also quite **easy to learn**, **understand**, and **code**. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory**. This further aids the readability of the code.

8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

9. Free and Open-Source

Like we said earlier, Python is **freely available**. But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.
Any doubts till now in the advantages of Python? Mention in the comment section.

2.3 Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

2.4 Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.
The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

2.5 History of Python : -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

2.6 What is Machine Learning : -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools

effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

2.7 Categories Of Machine Leaning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

2.8 Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction

- Stock market analysis and forecasting

- Speech synthesis

- Speech recognition

- Customer segmentation

- Object recognition

- Fraud detection

- Fraud prevention

- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term **"Machine Learning"** in 1959 and defined it as a **"Field of study that gives computers the capability to learn without being explicitly programmed".**

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of **$146,085** per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

*(*a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.
So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- **Model –** A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature –** A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label) –** A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training –** The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction –** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- **Supervised Learning –** This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning –** This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning –** This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning –** This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning :-

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and

reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps : -

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked.Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode.Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be   sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead Of Unicode Vs. 8-bit

Purpose :-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

14

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

2.9 Modules Used in Project :-

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent

interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

2.10 How to Install Python on Windows and Mac :

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System** Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

https://www.python.org

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Download the latest version for Windows

Download Python 3.7.4 ← **CLICK HERE**

Looking for Python with a different OS? Python for Windows,
Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases,
Docker images

Looking for Python 2.7? See below for specific releases

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.
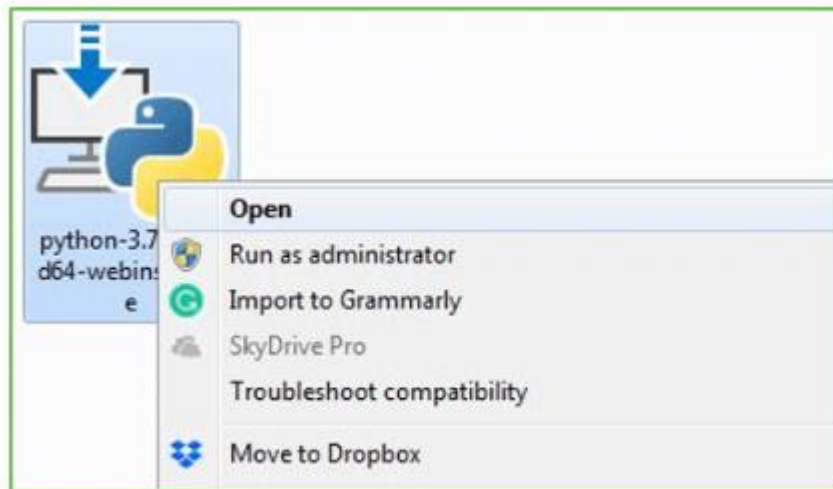
Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Step 3: Click on Install NOW After the installation is successful. Click on Close.

With these above three steps on python installation, you have successfully and correctly installed Python.

Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation
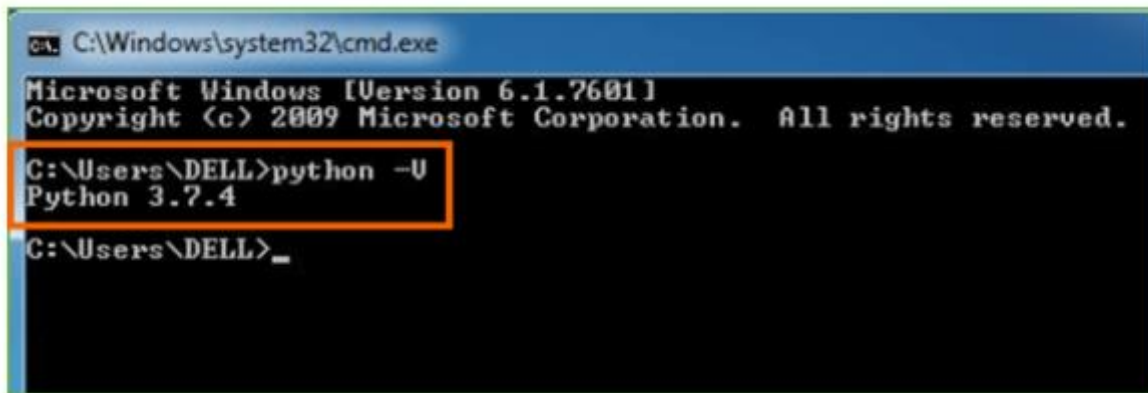
Step 1: Click on Start

Step 2: In the Windows Run Command, type "cmd"
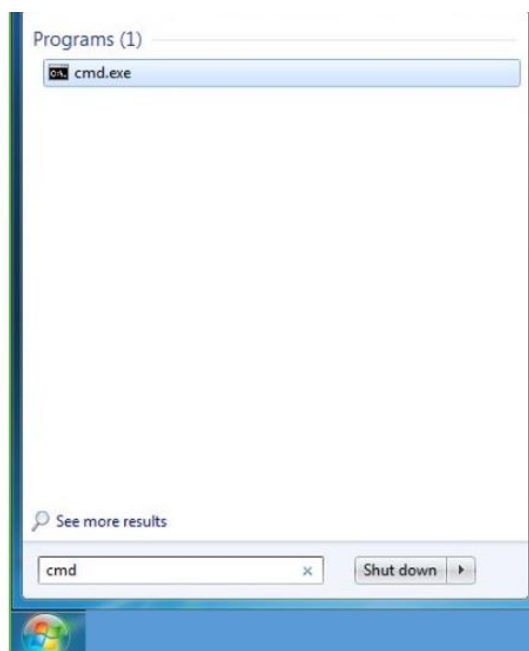
Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python –V** and press Enter.


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type "python idle"


```
Programs (2)
    IDLE (Python 3.7 64-bit)
    python idle
```
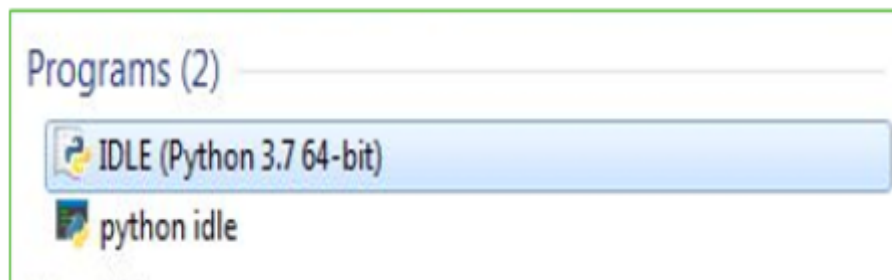
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**
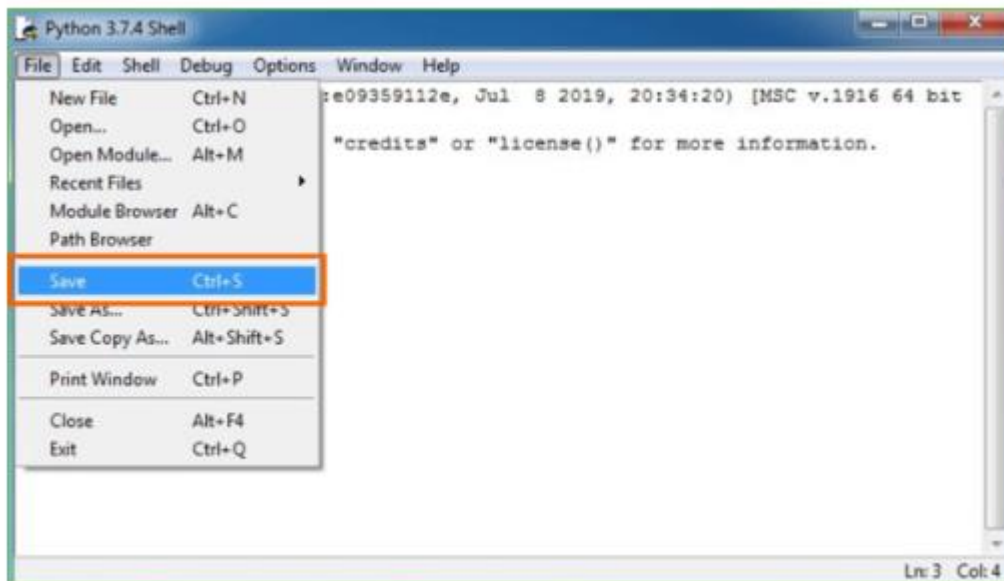
Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print ("Hey World")** and Press Enter.
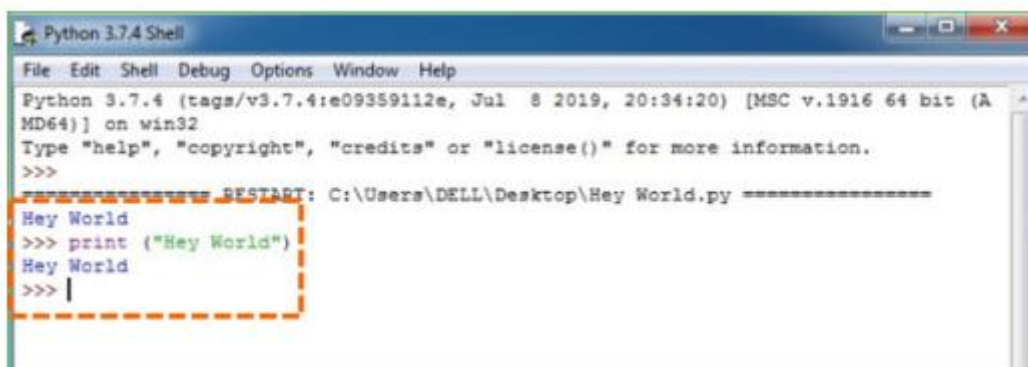
You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python doesn't need semicolons at the end of the statements otherwise it won't work. This stack that includes:

CHAPTER-3

SYSTEM DESIGN

3.1 Module Description

Feature selection is an important part in machine learning to reduce data dimensionality and extensive research carried out for a reliable feature selection method. For feature selection filter method and wrapper method have been used. In filter method, features are selected on the basis of their scores in various statistical tests that measure the relevance of features by their correlation with dependent variable or outcome variable. Wrapper method finds a subset of features by measuring the usefulness of a subset of feature with the dependent variable. Hence filter methods are independent of any machine learning algorithm whereas in wrapper method the best feature subset selected depends on the machine learning algorithm used to train the model. In wrapper method a subset evaluator uses all possible subsets and then uses a classification algorithm to convince classifiers from the features in each subset. The classifier consider the subset of feature with which the classification algorithm performs the best. To find the subset, the evaluator uses different search techniques like depth first search, random search, breadth first search or hybrid search. The filter method uses an attribute evaluator along with a ranker to rank all the features in the dataset. Here one feature is omitted at a time that has lower ranks and then sees the predictive accuracy of the classification algorithm. Weights or rank put by the ranker algorithms are different than those by the classification algorithm. Wrapper method is useful for machine learning test whereas filter method is suitable for data mining test because data mining has thousands of millions of features.

Algorithms used in this project :-

The steps involved in feature selection using Genetic Algorithm can be summarized as below:

STEP 1: Start the algorithm defining an initial set of population generated randomly

STEP 2: Assign a fitness score calculated by by the defined fitness function for genetic algorithm

STEP 3: Selection of parents: Chromosomes with good fitness scores are given preference over others to produce next generation off-springs

STEP 4: Perform crossover and mutation operations on the selected parents with the given probability of crossover and mutation

Repeat the steps 2 to 4 iteratively till the convergence is met and fittest chromosomes

from population, that is, the optimal feature subset is resulted.
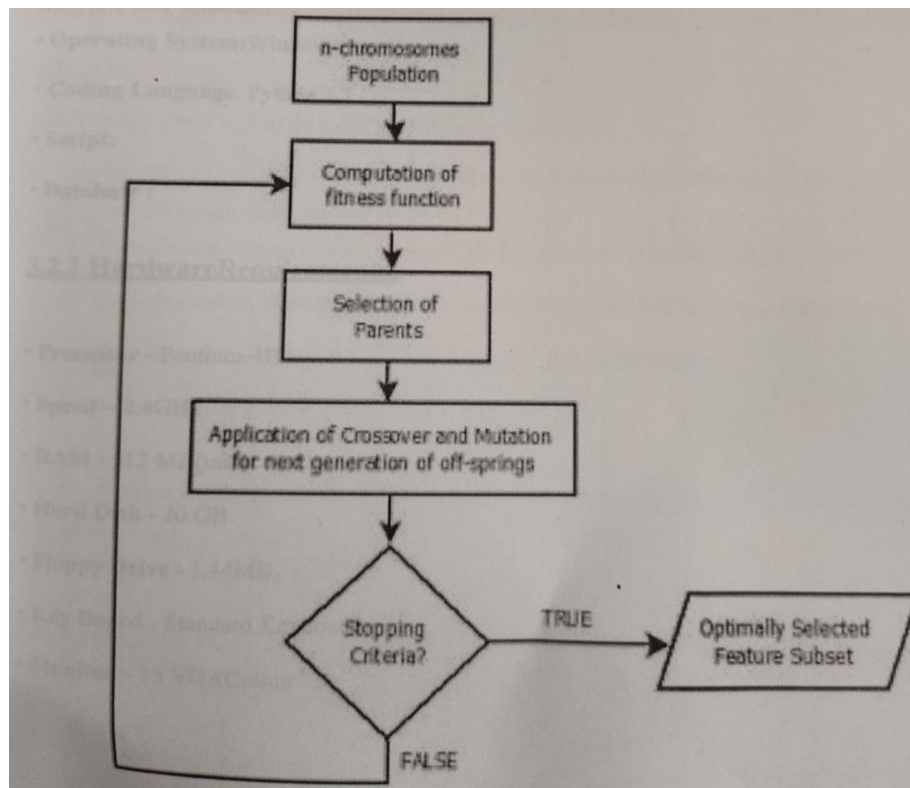
3.2 SystemSpecification

3.2.1 SoftwareRequirements

Functional requirements for a secure cloud storage service are straightforward:

1. The service should be able to store the user'sdata;

2. The data should be accessible through any devices connected to theInternet;

3. The service should be capable to synchronize the user's data between multiple devices (notebooks, smart phones,etc.);

4. The service should preserve all historical changes(versioning);

5. Data should be shareable with otherusers;

6. The service should support SSO;and

7. The service should be interoperable with other cloud storage services, enabling data migration from one CSP toanother.

• Operating System:Windows

• Coding Language: Python 3.7

• Script:

• Database :

3.2.2 HardwareRequirements:

• Processor - Pentium–III

• Speed – 2.4GHz

• RAM - 512 MB(min)

• Hard Disk - 20 GB

• Floppy Drive - 1.44MB

• Key Board - Standard Keyboard

• Monitor – 15 VGAColour

Cloud computing has three fundamental models, these are:

3.3 DetailedDesign

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques.

It is based on diagrammatic representations of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

UML was created as a result of the chaos revolving around software development and documentation. In the 1990s, there were several different ways to represent and document software systems. The need arose for a more unified way to visually represent those systems and as a result, in 1994-1996, the UML was developed by three software engineers working at Rational Software. It was later adopted as the standard in 1997 and has remained the standard ever since, receiving only a few updates.

GOALS:

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

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningfulmodels.

2. Provide extendibility and specialization mechanisms to extend the coreconcepts.

3. Be independent of particular programming languages and developmentprocess.

4. Provide a formal basis for understanding the modelinglanguage.

5. Encourage the growth of OO toolsmarket.

6 Support higher level development concepts such as collaborations, frameworks, patterns and components.

7   Integrate best practices.

i. USE CASEDIAGRAM:

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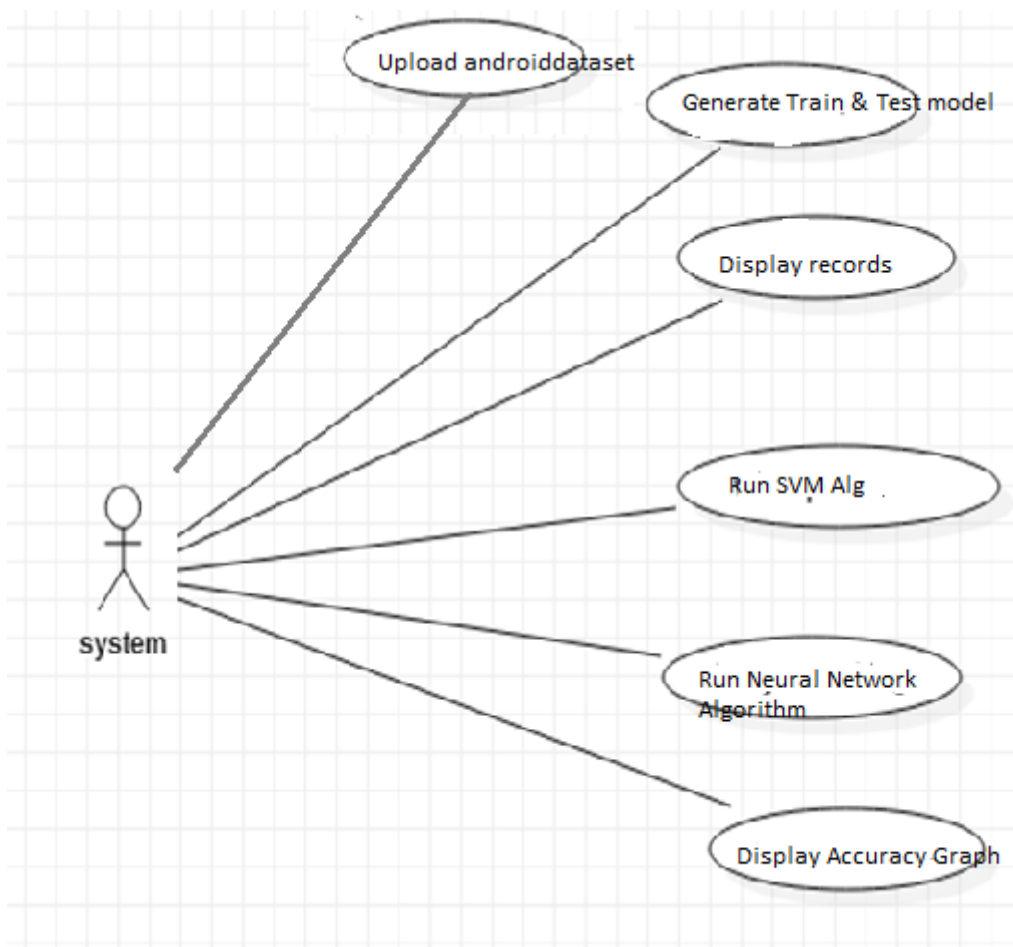
ii.   SEQUENCEDIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.
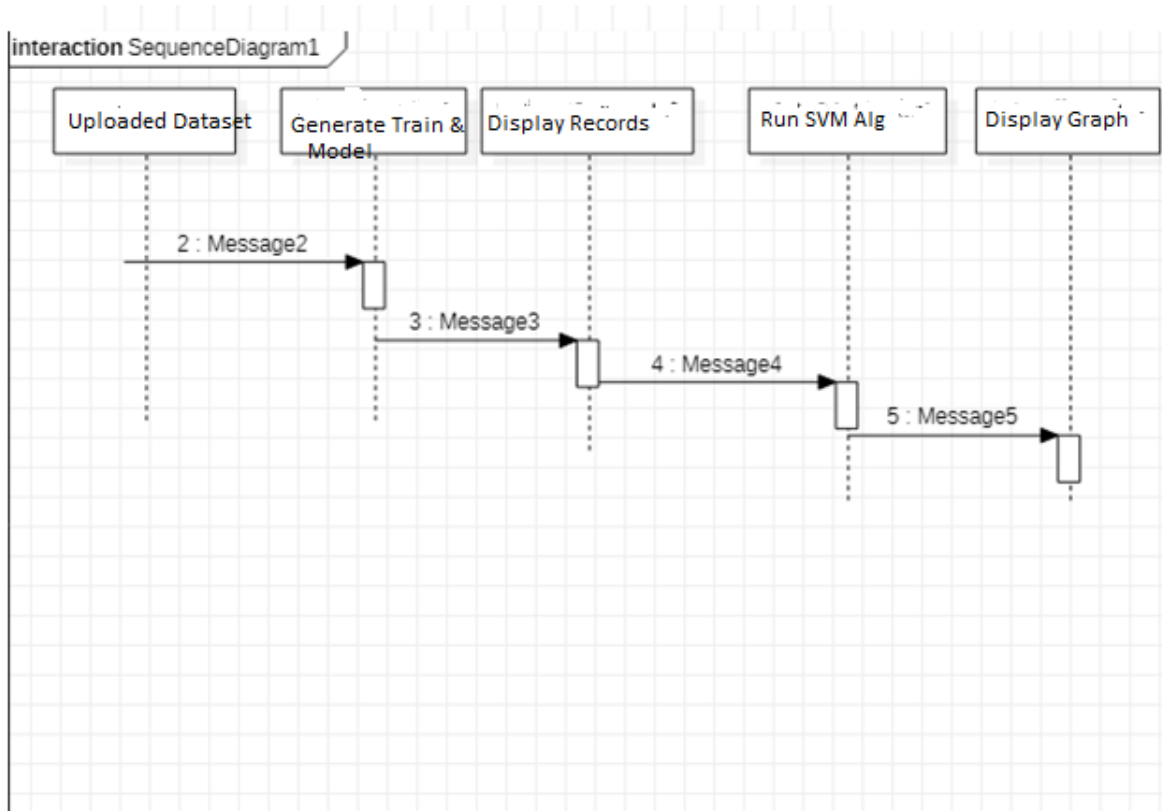
iii.   CLASSDIAGRAM:

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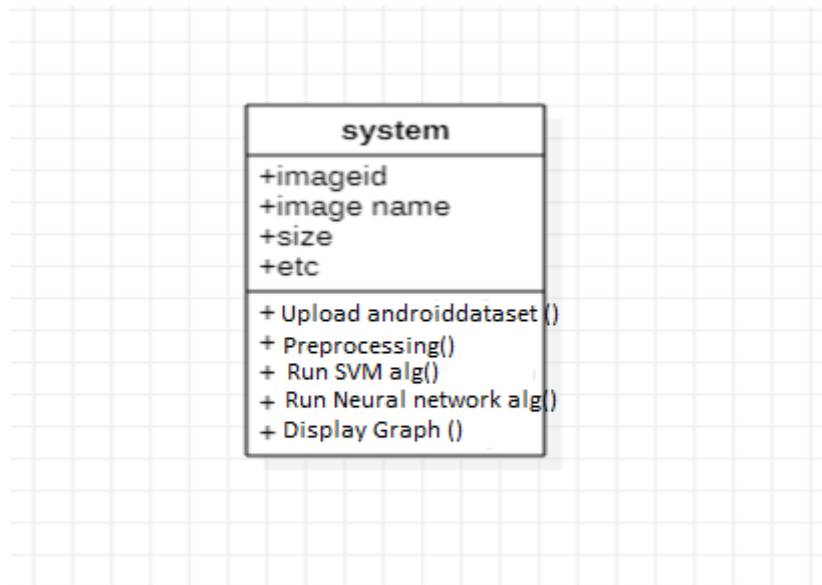
Component Diagram :-

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

UML Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.
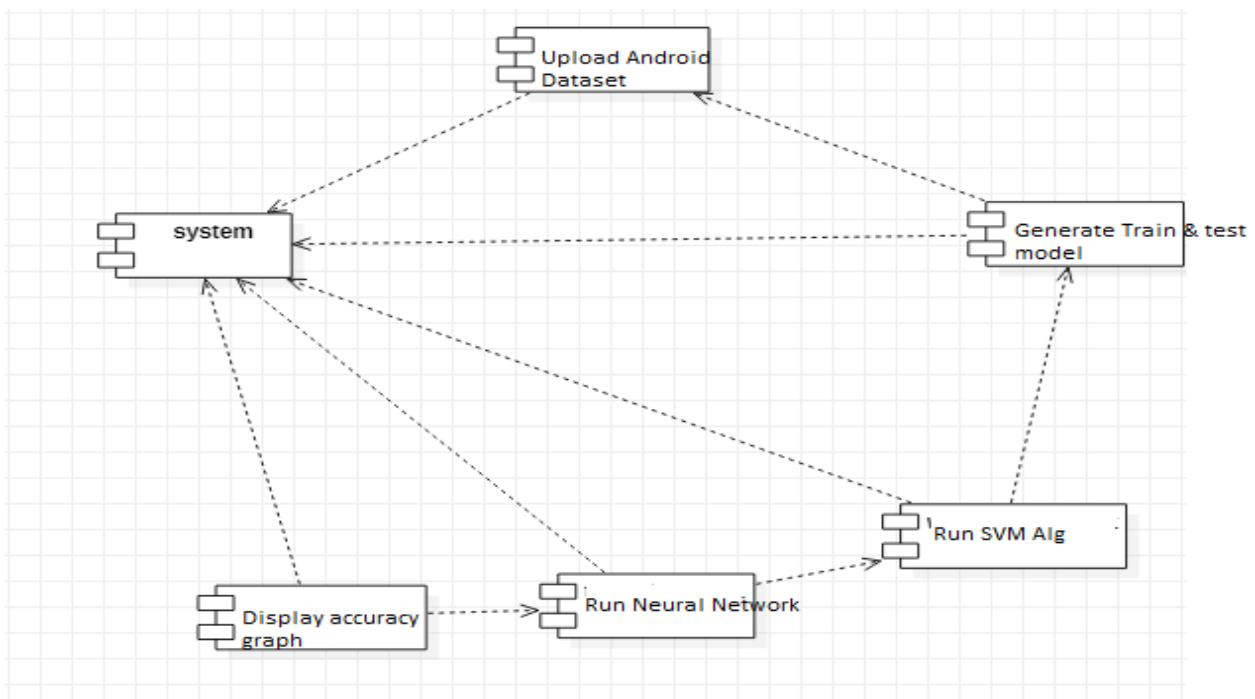
iv.   ACTIVITYDIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.
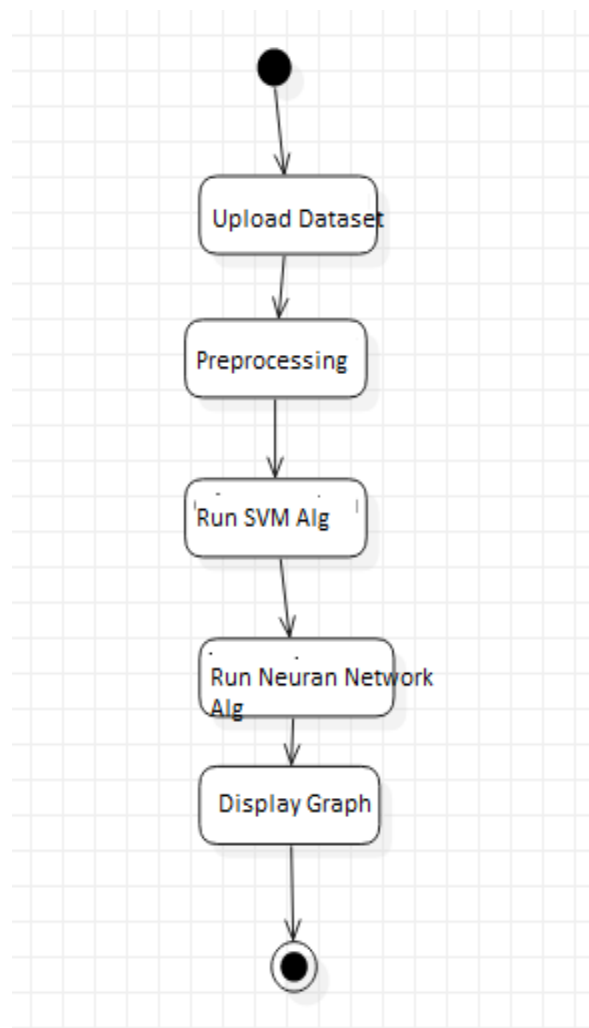
CHAPTER-4

IMPLEMENTATION

Index.html

```
from __future__ import absolute_import

from __future__ import division

from __future__ import print_function


import argparse

import collections

from datetime import datetime

import hashlib

import os.path

import random

import re

import sys

import tarfile


import numpy as np

from six.moves import urllib

import tensorflow as tf


from tensorflow.python.framework import graph_util

from tensorflow.python.framework import tensor_shape

from tensorflow.python.platform import gfile

from tensorflow.python.util import compat


FLAGS = None

MAX_NUM_IMAGES_PER_CLASS = 2 ** 27 - 1  # ~134M



def create_image_lists(image_dir, testing_percentage, validation_percentage):

if not gfile.Exists(image_dir):
```

```python
        tf.logging.error("Image directory '" + image_dir + "' not found.")
        return None
    result = collections.OrderedDict()
      sub_dirs = [
        os.path.join(image_dir,item)
        for item in gfile.ListDirectory(image_dir)]
      sub_dirs = sorted(item for item in sub_dirs
        if gfile.IsDirectory(item))
    for sub_dir in sub_dirs:
        extensions = ['jpg', 'jpeg', 'JPG', 'JPEG']
        file_list = []
        dir_name = os.path.basename(sub_dir)
    if dir_name == image_dir:
        continue
    tf.logging.info("Looking for images in '" + dir_name + "'")
    for extension in extensions:
        file_glob = os.path.join(image_dir, dir_name, '*.' + extension)
        file_list.extend(gfile.Glob(file_glob))
    if not file_list:
        tf.logging.warning('No files found')
        continue
    if len(file_list) < 20:
        tf.logging.warning(
            'WARNING: Folder has less than 20 images, which may cause issues.')
    elif len(file_list) > MAX_NUM_IMAGES_PER_CLASS:
        tf.logging.warning(
            'WARNING: Folder {} has more than {} images. Some images will '
            'never be selected.'.format(dir_name, MAX_NUM_IMAGES_PER_CLASS))
        label_name = re.sub(r'[^a-z0-9]+', ' ', dir_name.lower())
        training_images = []
        testing_images = []
        validation_images = []
    for file_name in file_list:
        base_name = os.path.basename(file_name)
```

```python
    hash_name = re.sub(r'_nohash_.*$', '', file_name)
    hash_name_hashed = hashlib.sha1(compat.as_bytes(hash_name)).hexdigest()
    percentage_hash = ((int(hash_name_hashed, 16) %
                (MAX_NUM_IMAGES_PER_CLASS + 1)) *
                (100.0 / MAX_NUM_IMAGES_PER_CLASS))
if percentage_hash < validation_percentage:
      validation_images.append(base_name)
elif percentage_hash < (testing_percentage + validation_percentage):
      testing_images.append(base_name)
else:
      training_images.append(base_name)
result[label_name] = {
      'dir': dir_name,
      'training': training_images,
      'testing': testing_images,
      'validation': validation_images,
   }
return result


def get_image_path(image_lists, label_name, index, image_dir, category):
if label_name not in image_lists:
tf.logging.fatal('Label does not exist %s.', label_name)
  label_lists = image_lists[label_name]
if category not in label_lists:
tf.logging.fatal('Category does not exist %s.', category)
  category_list = label_lists[category]
if not category_list:
tf.logging.fatal('Label %s has no images in the category %s.',
            label_name, category)
  mod_index = index % len(category_list)
  base_name = category_list[mod_index]
  sub_dir = label_lists['dir']
  full_path = os.path.join(image_dir, sub_dir, base_name)
```

```
return full_path


def get_bottleneck_path(image_lists, label_name, index, bottleneck_dir,
category, architecture):
return get_image_path(image_lists, label_name, index, bottleneck_dir,
category) + '_' + architecture + '.txt'


def create_model_graph(model_info):
with tf.Graph().as_default() as graph:
   model_path = os.path.join(FLAGS.model_dir, model_info['model_file_name'])
with gfile.FastGFile(model_path, 'rb') as f:
   graph_def = tf.GraphDef()
   graph_def.ParseFromString(f.read())
   bottleneck_tensor, resized_input_tensor = (tf.import_graph_def(
      graph_def,
name='',
      return_elements=[
         model_info['bottleneck_tensor_name'],
         model_info['resized_input_tensor_name'],
      ]))
return graph, bottleneck_tensor, resized_input_tensor


def run_bottleneck_on_image(sess, image_data, image_data_tensor,
               decoded_image_tensor, resized_input_tensor,
               bottleneck_tensor):
  resized_input_values = sess.run(decoded_image_tensor,
                     {image_data_tensor: image_data})
  bottleneck_values = sess.run(bottleneck_tensor,
                   {resized_input_tensor: resized_input_values})
  bottleneck_values = np.squeeze(bottleneck_values)
return bottleneck_values
```

```python
def maybe_download_and_extract(data_url):
  dest_directory = FLAGS.model_dir
if not os.path.exists(dest_directory):
os.makedirs(dest_directory)
filename = data_url.split('/')[-1]
filepath = os.path.join(dest_directory, filename)
if not os.path.exists(filepath):


def _progress(count, block_size, total_size):
sys.stdout.write('\r>> Downloading %s %.1f%%' %
             (filename,
float(count * block_size) / float(total_size) * 100.0))
sys.stdout.flush()


filepath, _ = urllib.request.urlretrieve(data_url, filepath, _progress)
print()
statinfo = os.stat(filepath)
tf.logging.info('Successfully downloaded', filename, statinfo.st_size,
           'bytes.')
tarfile.open(filepath, 'r:gz').extractall(dest_directory)



def ensure_dir_exists(dir_name):
if not os.path.exists(dir_name):
os.makedirs(dir_name)



bottleneck_path_2_bottleneck_values = {}



def create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                image_dir, category, sess, jpeg_data_tensor,
```

```
                  decoded_image_tensor, resized_input_tensor,
                  bottleneck_tensor):
tf.logging.info('Creating bottleneck at ' + bottleneck_path)
  image_path = get_image_path(image_lists, label_name, index,
                   image_dir, category)
if not gfile.Exists(image_path):
tf.logging.fatal('File does not exist %s', image_path)
  image_data = gfile.FastGFile(image_path, 'rb').read()
try:
    bottleneck_values = run_bottleneck_on_image(
sess, image_data, jpeg_data_tensor, decoded_image_tensor,
      resized_input_tensor, bottleneck_tensor)
except Exception as e:
raise RuntimeError('Error during processing file %s (%s)' % (image_path,
str(e)))
  bottleneck_string = ','.join(str(x) for x in bottleneck_values)
with open(bottleneck_path, 'w') as bottleneck_file:
    bottleneck_file.write(bottleneck_string)



def get_or_create_bottleneck(sess, image_lists, label_name, index, image_dir,
category, bottleneck_dir, jpeg_data_tensor,
                   decoded_image_tensor, resized_input_tensor,
                   bottleneck_tensor, architecture):
  label_lists = image_lists[label_name]
  sub_dir = label_lists['dir']
  sub_dir_path = os.path.join(bottleneck_dir, sub_dir)
  ensure_dir_exists(sub_dir_path)
  bottleneck_path = get_bottleneck_path(image_lists, label_name, index,
                       bottleneck_dir, category, architecture)
if not os.path.exists(bottleneck_path):
    create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                 image_dir, category, sess, jpeg_data_tensor,
                 decoded_image_tensor, resized_input_tensor,
```

38

```python
                    bottleneck_tensor)
with open(bottleneck_path, 'r') as bottleneck_file:
    bottleneck_string = bottleneck_file.read()
  did_hit_error = False
try:
    bottleneck_values = [float(x) for x in bottleneck_string.split(',')]
except ValueError:
tf.logging.warning('Invalid float found, recreating bottleneck')
    did_hit_error = True
if did_hit_error:
    create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                    image_dir, category, sess, jpeg_data_tensor,
                    decoded_image_tensor, resized_input_tensor,
                    bottleneck_tensor)
with open(bottleneck_path, 'r') as bottleneck_file:
      bottleneck_string = bottleneck_file.read()
    bottleneck_values = [float(x) for x in bottleneck_string.split(',')]
return bottleneck_values



def cache_bottlenecks(sess, image_lists, image_dir, bottleneck_dir,
                jpeg_data_tensor, decoded_image_tensor,
                resized_input_tensor, bottleneck_tensor, architecture):
  how_many_bottlenecks = 0
  ensure_dir_exists(bottleneck_dir)
for label_name, label_lists in image_lists.items():
for category in ['training', 'testing', 'validation']:
    category_list = label_lists[category]
for index, unused_base_name in enumerate(category_list):
      get_or_create_bottleneck(
sess, image_lists, label_name, index, image_dir, category,
        bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
        resized_input_tensor, bottleneck_tensor, architecture)
```

```python
        how_many_bottlenecks += 1
if how_many_bottlenecks % 100 == 0:
tf.logging.info(
str(how_many_bottlenecks) + ' bottleneck files created.')


def get_random_cached_bottlenecks(sess, image_lists, how_many, category,
                    bottleneck_dir, image_dir, jpeg_data_tensor,
                    decoded_image_tensor, resized_input_tensor,
                    bottleneck_tensor, architecture):
  class_count = len(image_lists.keys())
bottlenecks = []
  ground_truths = []
filenames = []
if how_many >= 0:
for unused_i in range(how_many):
    label_index = random.randrange(class_count)
    label_name = list(image_lists.keys())[label_index]
    image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
    image_name = get_image_path(image_lists, label_name, image_index,
                  image_dir, category)
bottleneck = get_or_create_bottleneck(
sess, image_lists, label_name, image_index, image_dir, category,
      bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
      resized_input_tensor, bottleneck_tensor, architecture)
    ground_truth = np.zeros(class_count, dtype=np.float32)
    ground_truth[label_index] = 1.0
bottlenecks.append(bottleneck)
    ground_truths.append(ground_truth)
filenames.append(image_name)
else:
for label_index, label_name in enumerate(image_lists.keys()):
for image_index, image_name in enumerate(
      image_lists[label_name][category]):
```

```python
        image_name = get_image_path(image_lists, label_name, image_index,
                        image_dir, category)
bottleneck = get_or_create_bottleneck(
sess, image_lists, label_name, image_index, image_dir, category,
        bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
        resized_input_tensor, bottleneck_tensor, architecture)
    ground_truth = np.zeros(class_count, dtype=np.float32)
    ground_truth[label_index] = 1.0
bottlenecks.append(bottleneck)
    ground_truths.append(ground_truth)
filenames.append(image_name)
return bottlenecks, ground_truths, filenames


def get_random_distorted_bottlenecks(
sess, image_lists, how_many, category, image_dir, input_jpeg_tensor,
    distorted_image, resized_input_tensor, bottleneck_tensor):
  class_count = len(image_lists.keys())
bottlenecks = []
  ground_truths = []
for unused_i in range(how_many):
    label_index = random.randrange(class_count)
    label_name = list(image_lists.keys())[label_index]
    image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
    image_path = get_image_path(image_lists, label_name, image_index, image_dir,
category)
if not gfile.Exists(image_path):
tf.logging.fatal('File does not exist %s', image_path)
    jpeg_data = gfile.FastGFile(image_path, 'rb').read()
    distorted_image_data = sess.run(distorted_image,
                        {input_jpeg_tensor: jpeg_data})
    bottleneck_values = sess.run(bottleneck_tensor,
                        {resized_input_tensor: distorted_image_data})
    bottleneck_values = np.squeeze(bottleneck_values)
```

```python
    ground_truth = np.zeros(class_count, dtype=np.float32)
    ground_truth[label_index] = 1.0
bottlenecks.append(bottleneck_values)
    ground_truths.append(ground_truth)
return bottlenecks, ground_truths


def should_distort_images(flip_left_right, random_crop, random_scale,
                 random_brightness):
return (flip_left_right or (random_crop != 0) or (random_scale != 0) or
        (random_brightness != 0))


def add_input_distortions(flip_left_right, random_crop, random_scale,
                 random_brightness, input_width, input_height,
                 input_depth, input_mean, input_std):
  jpeg_data = tf.placeholder(tf.string, name='DistortJPGInput')
  decoded_image = tf.image.decode_jpeg(jpeg_data, channels=input_depth)
  decoded_image_as_float = tf.cast(decoded_image, dtype=tf.float32)
  decoded_image_4d = tf.expand_dims(decoded_image_as_float, 0)
  margin_scale = 1.0 + (random_crop / 100.0)
  resize_scale = 1.0 + (random_scale / 100.0)
  margin_scale_value = tf.constant(margin_scale)
  resize_scale_value = tf.random_uniform(tensor_shape.scalar(),
minval=1.0,
maxval=resize_scale)
  scale_value = tf.multiply(margin_scale_value, resize_scale_value)
  precrop_width = tf.multiply(scale_value, input_width)
  precrop_height = tf.multiply(scale_value, input_height)
  precrop_shape = tf.stack([precrop_height, precrop_width])
  precrop_shape_as_int = tf.cast(precrop_shape, dtype=tf.int32)
  precropped_image = tf.image.resize_bilinear(decoded_image_4d,
                            precrop_shape_as_int)
  precropped_image_3d = tf.squeeze(precropped_image, squeeze_dims=[0])
```

```python
  cropped_image = tf.random_crop(precropped_image_3d,
                    [input_height, input_width, input_depth])
if flip_left_right:
  flipped_image = tf.image.random_flip_left_right(cropped_image)
else:
  flipped_image = cropped_image
brightness_min = 1.0 - (random_brightness / 100.0)
brightness_max = 1.0 + (random_brightness / 100.0)
brightness_value = tf.random_uniform(tensor_shape.scalar(),
minval=brightness_min,
maxval=brightness_max)
brightened_image = tf.multiply(flipped_image, brightness_value)
offset_image = tf.subtract(brightened_image, input_mean)
mul_image = tf.multiply(offset_image, 1.0 / input_std)
distort_result = tf.expand_dims(mul_image, 0, name='DistortResult')
return jpeg_data, distort_result


def variable_summaries(var):
with tf.name_scope('summaries'):
mean = tf.reduce_mean(var)
tf.summary.scalar('mean', mean)
with tf.name_scope('stddev'):
stddev = tf.sqrt(tf.reduce_mean(tf.square(var - mean)))
tf.summary.scalar('stddev', stddev)
tf.summary.scalar('max', tf.reduce_max(var))
tf.summary.scalar('min', tf.reduce_min(var))
tf.summary.histogram('histogram', var)


def add_final_training_ops(class_count, final_tensor_name, bottleneck_tensor,
              bottleneck_tensor_size):
with tf.name_scope('input'):
  bottleneck_input = tf.placeholder_with_default(
```

```python
      bottleneck_tensor,
shape=[None, bottleneck_tensor_size],
name='BottleneckInputPlaceholder')


    ground_truth_input = tf.placeholder(tf.float32,
                            [None, class_count],
name='GroundTruthInput')
  layer_name = 'final_training_ops'
with tf.name_scope(layer_name):
with tf.name_scope('weights'):
    initial_value = tf.truncated_normal(
       [bottleneck_tensor_size, class_count], stddev=0.001)


    layer_weights = tf.Variable(initial_value, name='final_weights')


    variable_summaries(layer_weights)
with tf.name_scope('biases'):
    layer_biases = tf.Variable(tf.zeros([class_count]), name='final_biases')
    variable_summaries(layer_biases)
with tf.name_scope('Wx_plus_b'):
logits = tf.matmul(bottleneck_input, layer_weights) + layer_biases
tf.summary.histogram('pre_activations', logits)


  final_tensor = tf.nn.softmax(logits, name=final_tensor_name)
tf.summary.histogram('activations', final_tensor)


with tf.name_scope('cross_entropy'):
  cross_entropy = tf.nn.softmax_cross_entropy_with_logits(
labels=ground_truth_input, logits=logits)
with tf.name_scope('total'):
    cross_entropy_mean = tf.reduce_mean(cross_entropy)
tf.summary.scalar('cross_entropy', cross_entropy_mean)


with tf.name_scope('train'):
```

```
optimizer = tf.train.GradientDescentOptimizer(FLAGS.learning_rate)
   train_step = optimizer.minimize(cross_entropy_mean)


return (train_step, cross_entropy_mean, bottleneck_input, ground_truth_input,
      final_tensor)




def add_evaluation_step(result_tensor, ground_truth_tensor):
with tf.name_scope('accuracy'):
with tf.name_scope('correct_prediction'):
prediction = tf.argmax(result_tensor, 1)
    correct_prediction = tf.equal(
prediction, tf.argmax(ground_truth_tensor, 1))
with tf.name_scope('accuracy'):
    evaluation_step = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
tf.summary.scalar('accuracy', evaluation_step)
return evaluation_step, prediction




def save_graph_to_file(sess, graph, graph_file_name):
  output_graph_def = graph_util.convert_variables_to_constants(
sess, graph.as_graph_def(), [FLAGS.final_tensor_name])
with gfile.FastGFile(graph_file_name, 'wb') as f:
f.write(output_graph_def.SerializeToString())
return




def prepare_file_system():
if tf.gfile.Exists(FLAGS.summaries_dir):
tf.gfile.DeleteRecursively(FLAGS.summaries_dir)
tf.gfile.MakeDirs(FLAGS.summaries_dir)
if FLAGS.intermediate_store_frequency > 0:
   ensure_dir_exists(FLAGS.intermediate_output_graphs_dir)
return
```

```python
def create_model_info(architecture):
architecture = architecture.lower()
if architecture == 'inception_v3':
    data_url = 'http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz'
    bottleneck_tensor_name = 'pool_3/_reshape:0'
    bottleneck_tensor_size = 2048
    input_width = 299
    input_height = 299
    input_depth = 3
    resized_input_tensor_name = 'Mul:0'
    model_file_name = 'classify_image_graph_def.pb'
    input_mean = 128
    input_std = 128
elif architecture.startswith('mobilenet_'):
parts = architecture.split('_')
if len(parts) != 3 and len(parts) != 4:
tf.logging.error("Couldn't understand architecture name '%s'",
architecture)
return None
    version_string = parts[1]
if (version_string != '1.0' and version_string != '0.75' and
      version_string != '0.50' and version_string != '0.25'):
tf.logging.error(
        """"The Mobilenet version should be '1.0', '0.75', '0.50', or '0.25',
but found '%s' for architecture '%s'""",
      version_string, architecture)
return None
    size_string = parts[2]
if (size_string != '224' and size_string != '192' and
      size_string != '160' and size_string != '128'):
tf.logging.error(
        """"The Mobilenet input size should be '224', '192', '160', or '128',
```

46

```
        but found '%s' for architecture '%s'""",
            size_string, architecture)
    return None
    if len(parts) == 3:
        is_quantized = False
    else:
        if parts[3] != 'quantized':
        tf.logging.error(
                "Couldn't understand architecture suffix '%s' for '%s'", parts[3],
    architecture)
    return None
            is_quantized = True
        data_url = 'http://download.tensorflow.org/models/mobilenet_v1_'
        data_url += version_string + '_' + size_string + '_frozen.tgz'
        bottleneck_tensor_name = 'MobilenetV1/Predictions/Reshape:0'
        bottleneck_tensor_size = 1001
        input_width = int(size_string)
        input_height = int(size_string)
        input_depth = 3
        resized_input_tensor_name = 'input:0'
    if is_quantized:
        model_base_name = 'quantized_graph.pb'
    else:
        model_base_name = 'frozen_graph.pb'
        model_dir_name = 'mobilenet_v1_' + version_string + '_' + size_string
        model_file_name = os.path.join(model_dir_name, model_base_name)
        input_mean = 127.5
        input_std = 127.5
    else:
    tf.logging.error("Couldn't understand architecture name '%s'", architecture)
    raise ValueError('Unknown architecture', architecture)

    return {
        'data_url': data_url,
```

```
    'bottleneck_tensor_name': bottleneck_tensor_name,
    'bottleneck_tensor_size': bottleneck_tensor_size,
    'input_width': input_width,
    'input_height': input_height,
    'input_depth': input_depth,
    'resized_input_tensor_name': resized_input_tensor_name,
    'model_file_name': model_file_name,
    'input_mean': input_mean,
    'input_std': input_std,
  }


def add_jpeg_decoding(input_width, input_height, input_depth, input_mean,
                input_std):
  jpeg_data = tf.placeholder(tf.string, name='DecodeJPGInput')
  decoded_image = tf.image.decode_jpeg(jpeg_data, channels=input_depth)
  decoded_image_as_float = tf.cast(decoded_image, dtype=tf.float32)
  decoded_image_4d = tf.expand_dims(decoded_image_as_float, 0)
  resize_shape = tf.stack([input_height, input_width])
  resize_shape_as_int = tf.cast(resize_shape, dtype=tf.int32)
  resized_image = tf.image.resize_bilinear(decoded_image_4d,
                           resize_shape_as_int)
  offset_image = tf.subtract(resized_image, input_mean)
  mul_image = tf.multiply(offset_image, 1.0 / input_std)
return jpeg_data, mul_image


def main(_):
  # Needed to make sure the logging output is visible.
  # See https://github.com/tensorflow/tensorflow/issues/3047
  tf.logging.set_verbosity(tf.logging.INFO)

  # Prepare necessary directories  that can be used during training
  prepare_file_system()
```

```python
  # Gather information about the model architecture we'll be using.
  model_info = create_model_info(FLAGS.architecture)
if not model_info:
tf.logging.error('Did not recognize architecture flag')
return -1

  # Set up the pre-trained graph.
  maybe_download_and_extract(model_info['data_url'])
graph, bottleneck_tensor, resized_image_tensor = (
    create_model_graph(model_info))

  # Look at the folder structure, and create lists of all the images.
  image_lists = create_image_lists(FLAGS.image_dir, FLAGS.testing_percentage,
                   FLAGS.validation_percentage)
  class_count = len(image_lists.keys())
if class_count == 0:
tf.logging.error('No valid folders of images found at ' + FLAGS.image_dir)
return -1
if class_count == 1:
tf.logging.error('Only one valid folder of images found at ' +
           FLAGS.image_dir +
           ' - multiple classes are needed for classification.')
return -1

  # See if the command-line flags mean we're applying any distortions.
  do_distort_images = should_distort_images(
    FLAGS.flip_left_right, FLAGS.random_crop, FLAGS.random_scale,
    FLAGS.random_brightness)

with tf.Session(graph=graph) as sess:
   # Set up the image decoding sub-graph.
   jpeg_data_tensor, decoded_image_tensor = add_jpeg_decoding(
     model_info['input_width'], model_info['input_height'],
```

```python
                model_info['input_depth'], model_info['input_mean'],
                model_info['input_std'])


if do_distort_images:
    # We will be applying distortions, so setup the operations we'll need.
    (distorted_jpeg_data_tensor,
     distorted_image_tensor) = add_input_distortions(
        FLAGS.flip_left_right, FLAGS.random_crop, FLAGS.random_scale,
        FLAGS.random_brightness, model_info['input_width'],
        model_info['input_height'], model_info['input_depth'],
        model_info['input_mean'], model_info['input_std'])
else:
    # We'll make sure we've calculated the 'bottleneck' image summaries and
    # cached them on disk.
    cache_bottlenecks(sess, image_lists, FLAGS.image_dir,
                FLAGS.bottleneck_dir, jpeg_data_tensor,
                decoded_image_tensor, resized_image_tensor,
                bottleneck_tensor, FLAGS.architecture)
```

CHAPTER – 5

TEST RESULTS

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.1 TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application

.it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expectedresults.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination ofcomponents.

Functional test

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

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
InvalidInput	: identified classes of invalid input must be rejected.
Functions	: identified functions must beexercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures: interfacing systems or procedures must beinvoked.	

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

System Test

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

White Box Testing

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

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated,

asablackbox.youcannot-see|intoit.Thetestprovidesinputsandrespondsto outputswithout considering how the softwareworks.

5.2 UnitTesting:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

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

Test objectives

- All field entries must workproperly.

- Pages must be activated from the identified link.

- The entry screen, messages and responses must not bedelayed.

Features to be tested

- Verify that the entries are of the correctformat

- No duplicate entries should beallowed

- All links should take the user to the correctpage.

5.3 IntegrationTesting

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.
The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

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

5.4AcceptanceTesting

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

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

CHAPTER 6

RESULTS
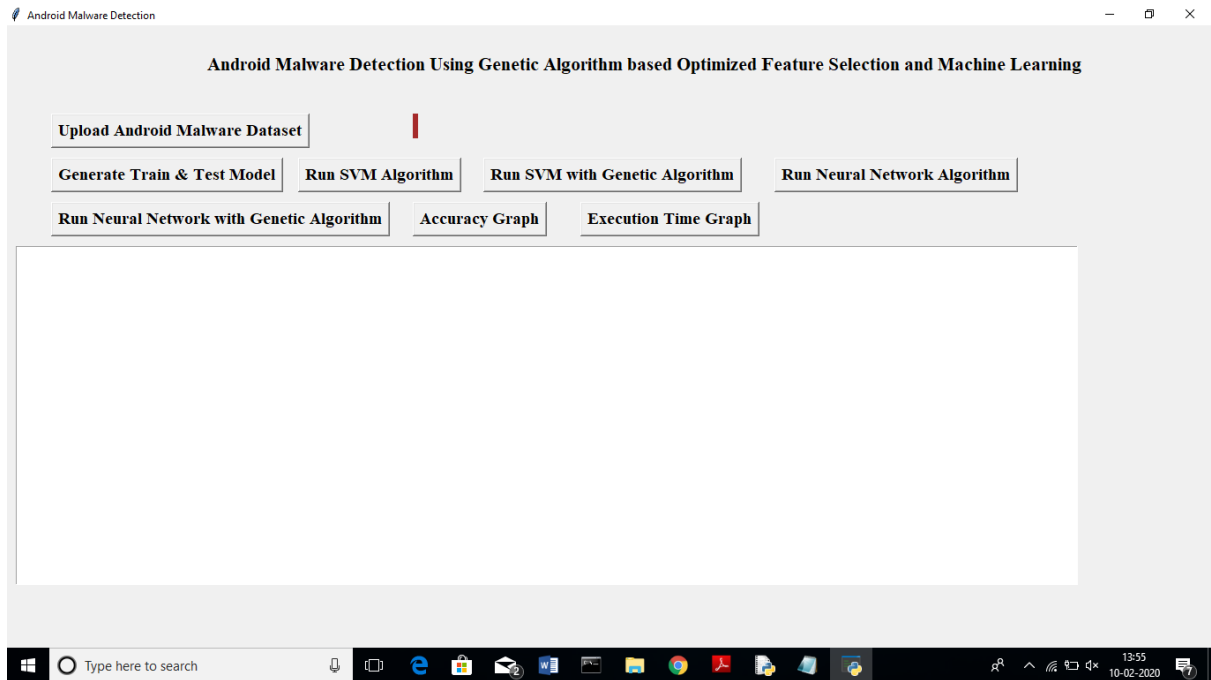
In above screen click on 'Upload Android Malware Dataset' button and upload dataset.
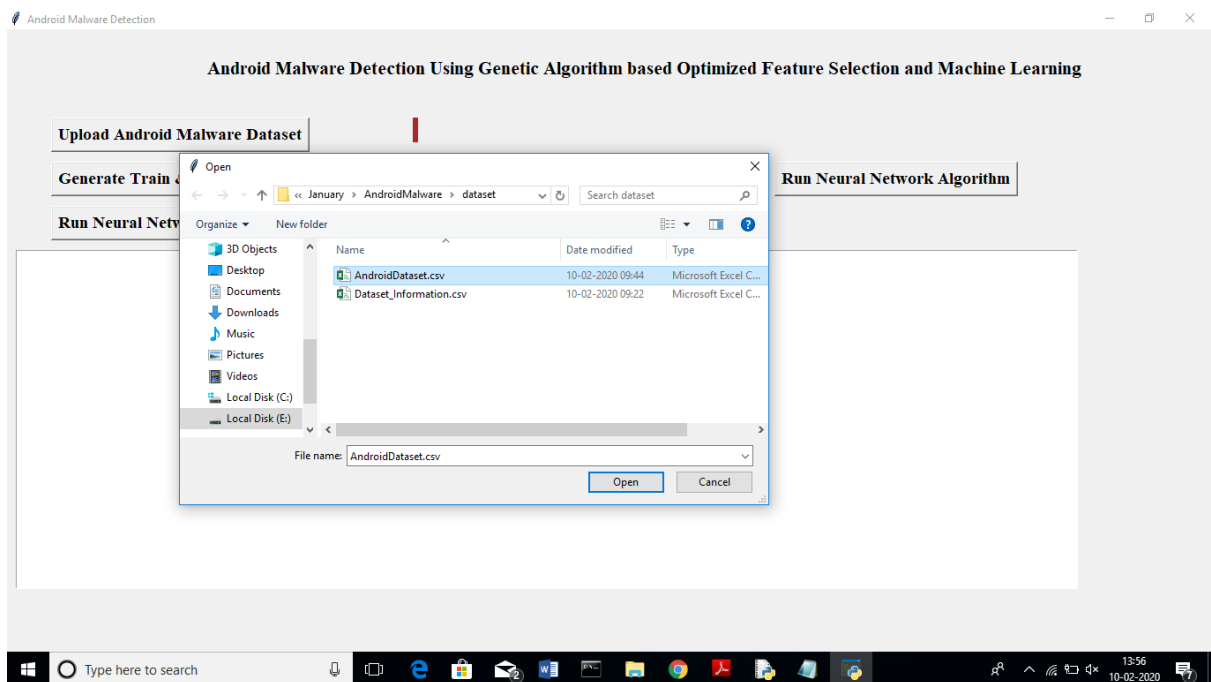
In above screen I am uploading 'AndroidDataset.csv' file and after upload will get below screen
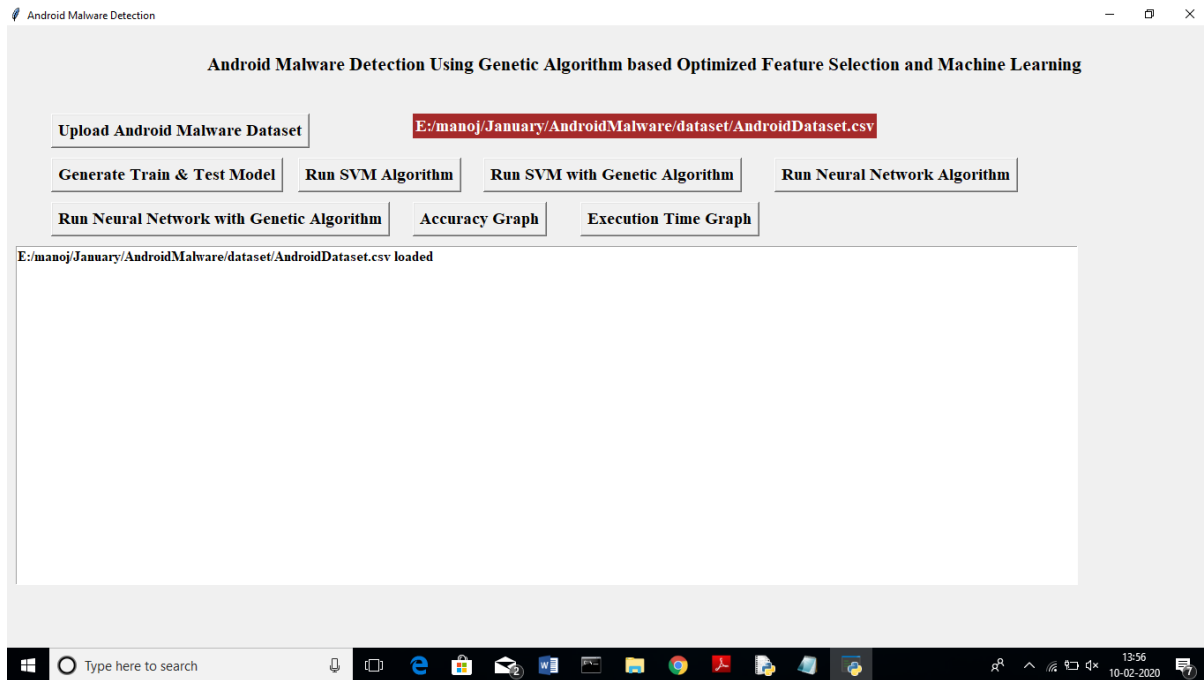
Now click on 'Generate Train & Test Model' button to split dataset into train and test part. All machine learning algorithms will take 80% dataset for training and 20% dataset to test accuracy of trained model. After clicking that button will get train and test model
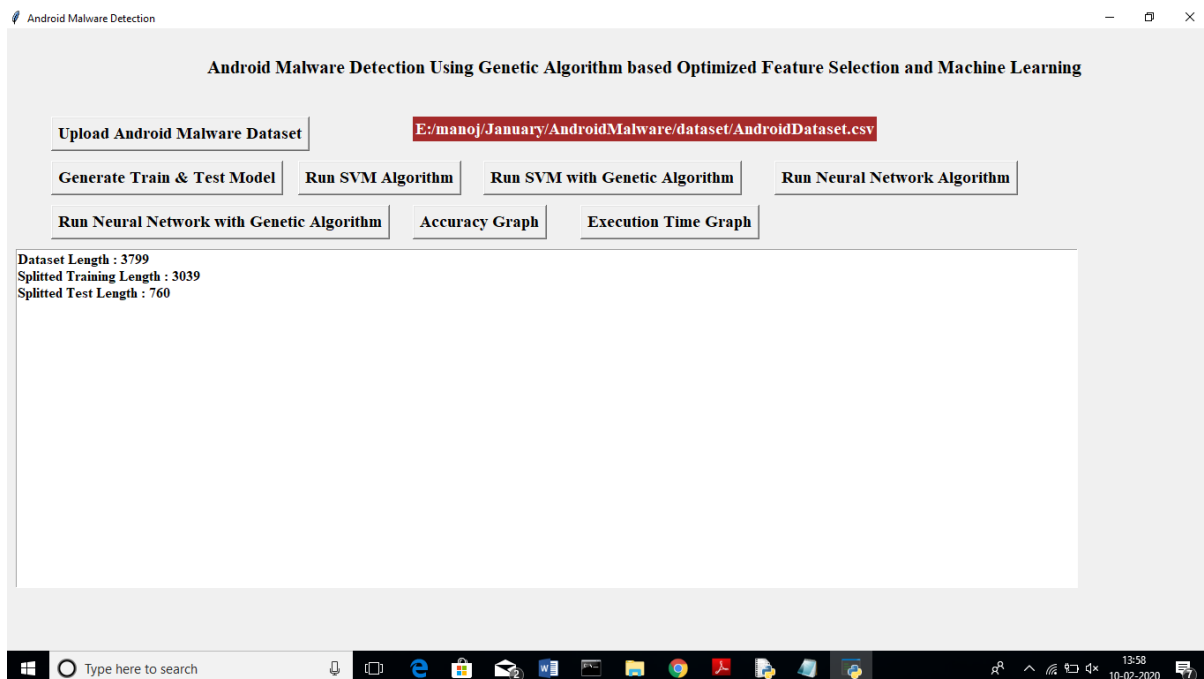
In above screen we can see there are total 3799 android app records are there and application using 3039 records for training and 760 records for testing. Now we have both train and test model and now click on 'Run SVM Algorithm' button to generate SVM model on train and test and get its accuracy
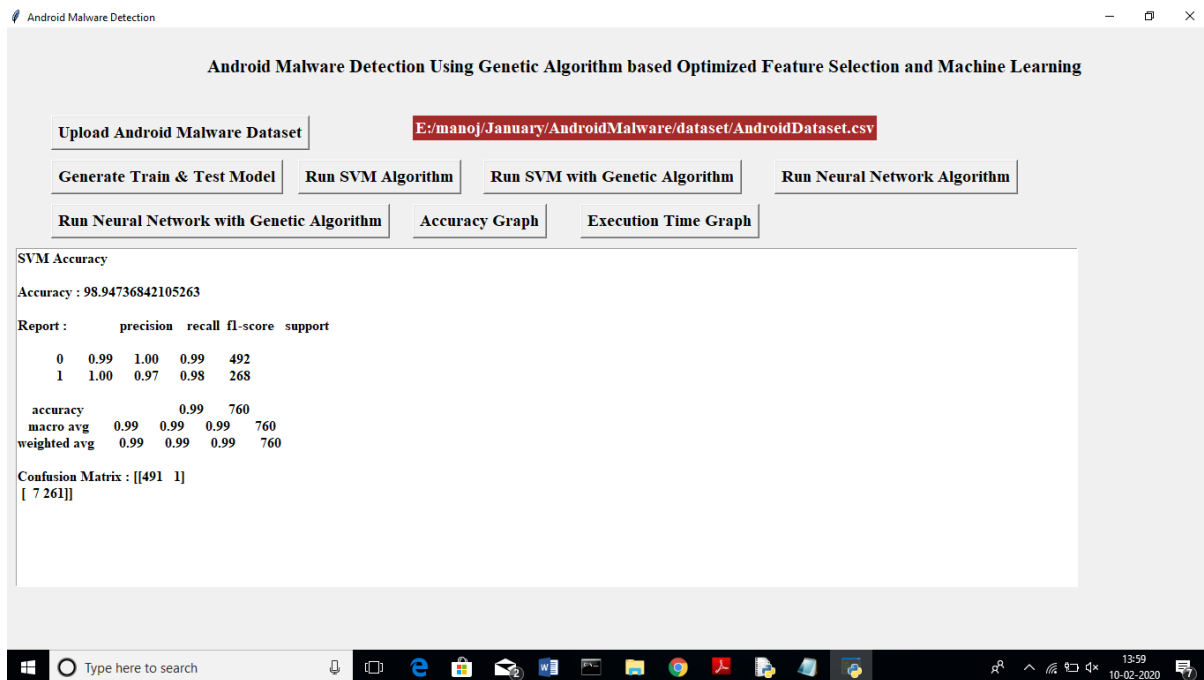
In above screen we got 98% accuracy for SVM and now click on 'Run SVM with Genetic Algorithm' button to choose optimize features and then run SVM on optimize features to get accuracy
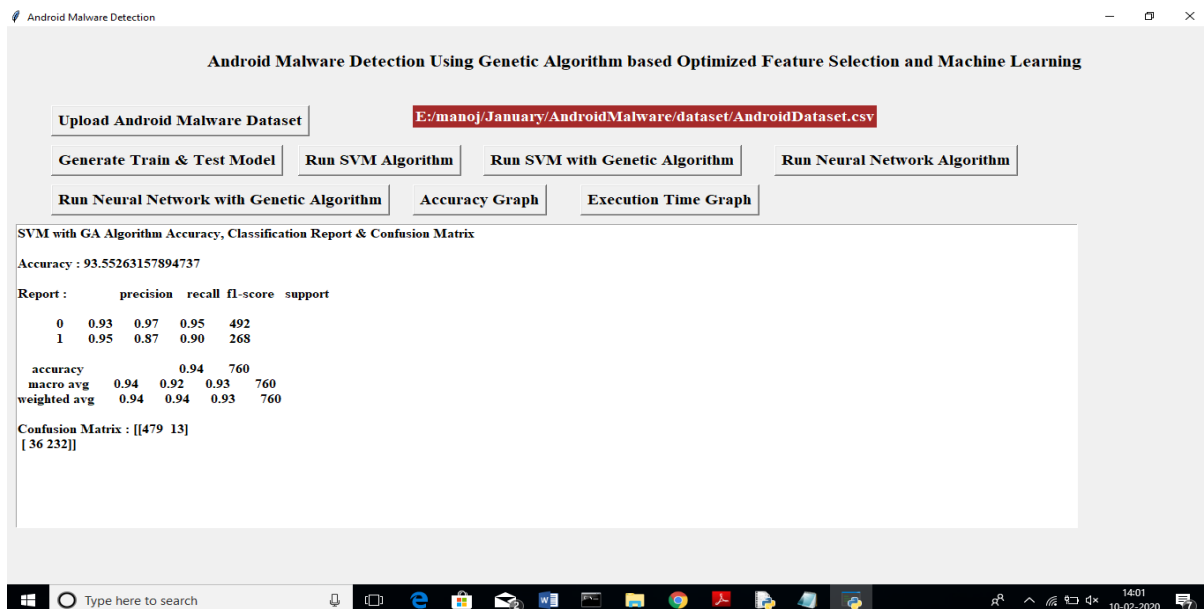
In above screen SVM with Genetic algorithm got 93% accuracy. Genetic with SVM accuracy is less but its execution time will be less which we can see at the time of comparison graph.

(Note: when u run genetic then 4 empty windows will open u just close all those 4 windows and let main window to run)

In above console we can see genetic algorithm chooses 40 features from all dataset features.

Now click on 'Run Neural Network Algorithm' button to test neural network accuracy.
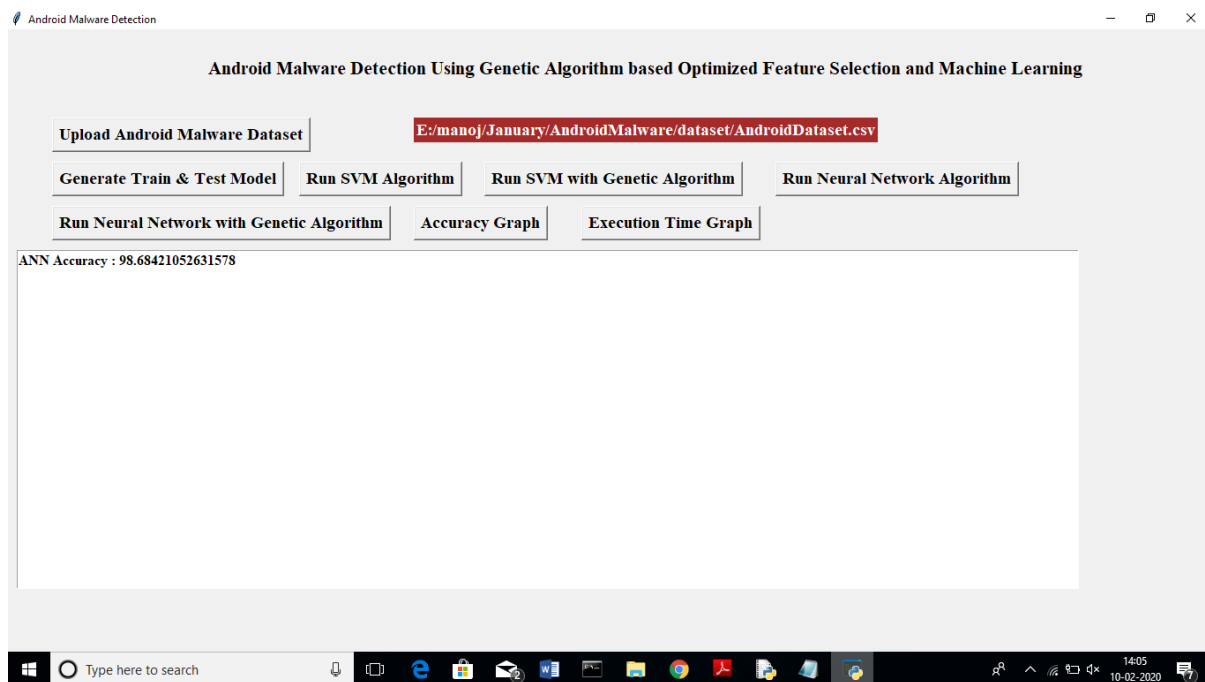
In above screen neural network also gave 98.64% accuracy. Now click on 'Run Neural Network with Genetic Algorithm' button to get NN accuracy with genetic algorithm
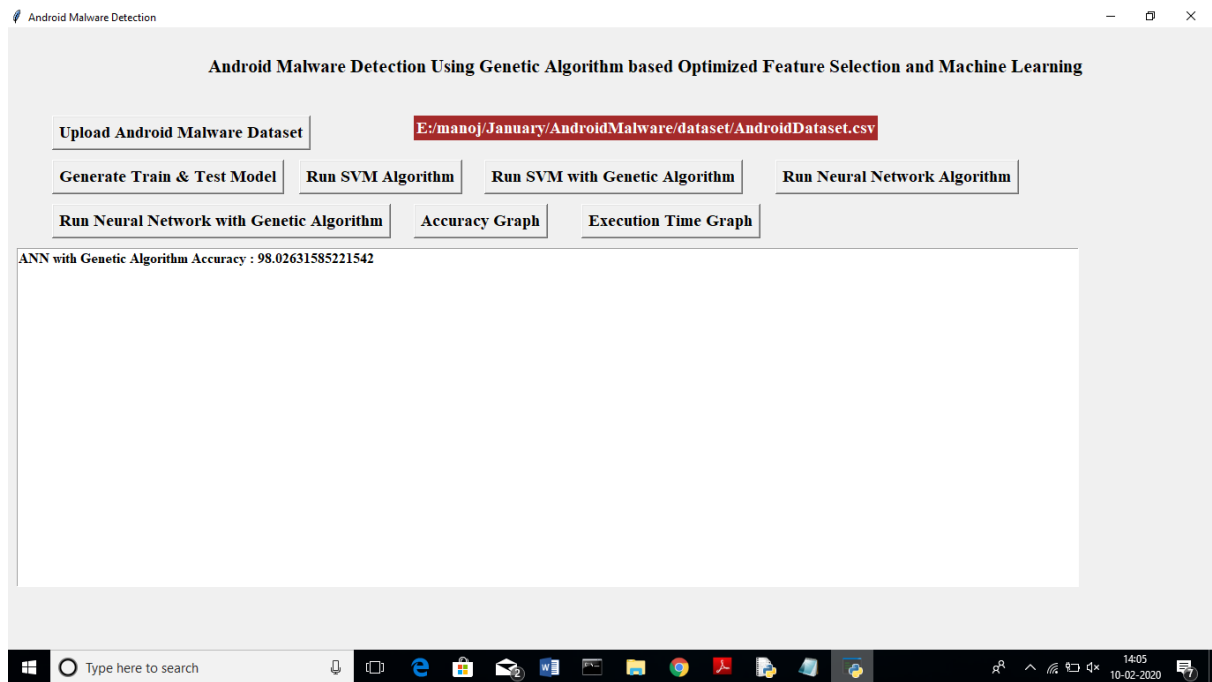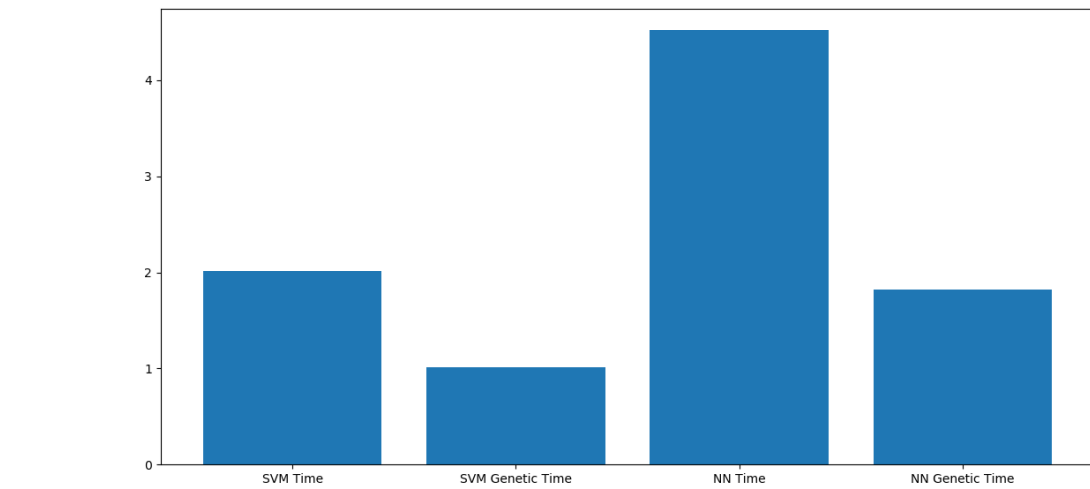
In above screen NN with genetic got 98.02% accuracy. Now click on 'Accuracy Graph' button to see all algorithms accuracy in graph

In above graph x-axis represents algorithm name and y-axis represents accuracy and in all SVM got high accuracy. Now click on 'Execution Time Graph' button to get execution time of all algorithm

In above graph x-axis represents algorithm name and y-axis represents execution time. From above graph we can conclude that with genetic algorithm machine learning algorithms taking less time to build model.

CHAPTER 7

CONCLUSION & FUTURE WORK

CONCLUSION

As the number of threats posed to Android platforms is increasing day to day, spreading mainly through malicious applications or malwares, therefore it is very important to design a framework which can detect such malwares with accurate results. Where signature-based approach fails to detect new variants of malware posing zero-day threats, machine learning based approaches are being used. The proposed methodology attempts to make use of evolutionary Genetic Algorithm to get most optimized feature subset which can be used to train machine learning algorithms in most efficient way.

Future Work

From experimentations, it can be seen that a decent classification accuracy of more than 94% is maintained using Support Vector Machine and Neural Network classifiers while working on lower dimension feature-set, thereby reducing the training complexity of the classifiers Further work can be enhanced using larger datasets for improved results and analyzing the effect on other machine learning algorithms when used in conjunction with Genetic Algorithm.

REFERENCES

D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," in Proceedings 2014 Network and Distributed System Security Symposium, 2014.

[2] N. Milosevic, A. Dehghantanha, and K. K. R. Choo, "Machine learning aided Android malware classification," Comput.Electr.Eng., vol. 61, pp. 266–274, 2017.

[3] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant Permission Identification for Machine-Learning-Based Android Malware Detection," IEEE Trans. Ind. Informatics, vol. 14, no. 7, pp. 3216–3225, 2018.

[4] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," IEEE Trans. Dependable Secur. Comput., vol. 15, no. 1, pp. 83–97, 2018.

[5] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song, and H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," IEEE Access, vol. 6, pp. 4321–4339, 2018.

-