A Project Report

on

Malicious Application Detection Using Machine Learning

Submitted in partial fulfilment of the requirements for the award of degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

17WH1A0506	Ms. B CHINMAI
17WH1A0558	Ms. A MEGHANA
17WH1A0560	Ms. K KRISHNA SIRI

Under the esteemed guidance of
DR. GANTI NAGA SATISH
Professor

Department of Computer Science and Engineering

BVRIT HYDERABAD College of Engineering for Women
(NBA Accredited EEE, ECE, CSE, IT B.Tech. Courses,
Accredited by NAAC with 'A' Grade)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

May 2021

BVRIT HYDERABAD College of Engineering for Women
(NBA Accredited EEE, ECE, CSE, IT B.Tech. Courses,
Accredited by NAAC with ‘A’ Grade)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project work report entitled “**Malicious Application Detection Using Machine Learning**” is a bonafide work carried out by **Ms. B. Chinmai (17WH1A0506), Ms. A. Meghana (17WH1A0558), Ms. K. Krishna Siri (17WH1A0560)** in partial fulfillment for the award of B.Tech degree in **Computer Science & Engineering** , **BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide
Dr. Ganti Naga Satish
Professor, CSE

Head of the Department
Dr. K. Srinivasa Reddy
Professor & HOD,CSE

External Examiner

DECLARATION

We hereby declare that the work presented in this project work entitled **"Malicious Application Detection Using Machine Learning"** submitted towards completion of Project work in IV Year of B.Tech of CSE at **BVRIT HYDERABAD College of Engineering for Women,** Hyderabad is an authentic record of our original work carried out under the guidance of **Dr. Ganti Naga Satish, Professor, Department of Computer Science and Engineering.**

Roll No	**Name**	**Signature**
17WH1A0506	Ms. B Chinmai	
17WH1A0558	Ms. A Meghana	
17WH1A0560	Ms. K Krishna Siri	

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. K. V. N. Sunitha**, **Principal, BVRIT HYDERABAD College of Engineering for Women,** for her support by providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. Srinivasa Reddy Konda,** Head of the Department of CSE, BVRIT HYDERABAD College of Engineering for Women, for all timely support and valuable suggestions during the period of our project.

We are extremely thankful to our Internal Guide, **Dr. Ganti Naga Satish, Professor, CSE,** BVRIT HYDERABAD College of Engineering for Women**,** for her constant guidance and encouragement throughout the project.

We would like to record my sincere thankfulness to the major project coordinator **Dr. Ganti Naga Satish, Professor, CSE,** for his valuable guidence and skillful management.

17WH1A0506	**Ms. B Chinmai**
17WH1A0558	**Ms. A Meghana**
17WH1A0560	**Ms. K Krishna Siri**

TABLES OF CONTENTS

ABSTRACT

Android plays a vital role in the today's market. According to recent survey placed nearly 84.4% of people stick to android which explosively become popular for personal or business purposes. It is no doubt that the application is extremely familiar in the market for their amazing features and the wonderful benefits of android applications makes the users to fall for it. Android imparts significant responsibility to application developers for designing the application with understanding the risk of security issues. When concerned about security, malware protection is a major issue in which android has been a major target of malicious applications. In android based applications, permission control is one of the major security mechanisms. In this project, the permission induced risk in application, and the fundamentals of the android security architecture are explored, and it also focuses on the security ranking algorithms that are unique to specific applications. Hence, we propose the system providing the detection of malware analysis based on permission and steps to mitigate from accessing unwanted permission (limits the permission). It is also designed to reduce the probability of vulnerable attacks.

LIST OF FIGURES

1. INTRODUCTION

In recent years, the usages of smart phones are increasing steadily and also growth of Android application users are increasing. Due to growth of Android application user, some intruder are creating malicious android application as tool to steal the sensitive data and identity theft / fraud mobile bank, mobile wallets. There are so many malicious applications detection tools and software are available.

But an effectively and efficiently malicious applications detection tools needed to tackle and handle new complex malicious apps created by intruder or hackers. In this paper we came up with idea of using machine learning approaches for detecting the malicious android application. First we have to gather dataset of past malicious apps as training set and with the help of Support vector machine algorithm and decision tree algorithm make up comparision with training dataset and trained datasetwe can predict the malware android apps

1.1 Objective:

The ultimate aim of the project is to improve permission for detecting the malicious android mobileapplication using machine learning algorithms.

1.2 Existing System:

Traditionally Numerous malware detection tools have been developed, but some tools are may not able to detect newly created malware application and unknown malware application infected by various Trojan, worms, spyware. Detecting of large number of malicious application over millions of android application is still a challenging task using traditional way. In existing, Non machine learning way of detecting the malicious application based on characteristics, properties, behavioral.

1.2.1 Disadvantages:

- ✓ Identification of newly updated or created malicious application is hard to find out.
- ✓ Non Machine learning approaches are not reliable and efficient
- ✓ In Existing approaches covers only 30 permissions out of 300 app permissions, due to thislimited apps permissions different types of attacks can occur.

1.3 Proposed System:

In proposed paper, we implement SIGPID, Significant Permission Identification (SIGPID). The goal of the sigpid is to improve the apps permissions effectively and efficiently. This SIGID system improves the accuracy and efficient detection of malware application. With help machine learning algorithms such as SVM and Decision Tree algorithms make a comparison between training and trained datasets.

Support vector machine algorithms act as a classifier which is used to classify malicious application and benign app**.**

1.3.1 Advantages

- ✓ Improves the percentages of detection malicious application.
- ✓ Machine learning is better efficient than Non machine learning algorithm.
- ✓ Able to detect new malware android applications.
- ✓ We only need to consider 22 out of 135 permissions to improve the runtime performance by85.6.

1.4 Methodology

To classify malicious application from benign application a decent dataset is required.The dataset can be downloaded from debrin dataset. We construct massive experiments, including 516 benign applications and 528 malicious applications. In this section the methodology followed is discussed in detail.

1.5 Dataset

Proper and large dataset is required for all classification research during the training and the testing phase. The dataset for the experiment is downloaded from the debrin data set, which contains different android application permissions and their values. It contains a collection of android permissions with their names.

1.6 Permission

Permission characterize existing Android malware from various aspects, including the permissions requested. They identified individually the permissions that are widely requested in both malicious and benign apps.

According to this work, malicious apps clearly tend to request more frequently on the SMS-related permissions, such as ‘READ SMS’, ‘WRITE SMS’, ‘RECEIVE SMS’, and ‘SEND SMS’. we found that malicious apps tend to request more permissions than benign ones. we found no strong correlation between applications categories and requested permissions, and introduce a method to visualize permissions usage in different app categories.

The aim of their work is to classify Android applications into several categories such as entertainment, society, tools, and productivity, multimedia and video, communication, puzzle and brain games. Mentions a method that analyses manifest files in Android application by extracting four types of keyword lists:

(1) Permission

(2) Intent filter(action)

(3) Intent filter(category)

(4) Process name.

This approach determines the malignancy score by classifying individually permissions as malicious orbenign.

1.7 Combination of Permission

A high-level contextual analysis and an exploration of Android applications based on their implementation of permission-based security models by applying network visualization techniques and clustering algorithms. From that, they discovered new potentials in permission-based security models that may provide additional security to the users. This method on network classification helps to define irregular permission combinations requested by abnormal applications. The nature, sources and implications of sensitive data on Android devices in enterprise settings. They characterized malicious apps and the risks they pose to enterprises. Finally, they have proposed several approaches for dealing with security risks for enterprise. From the analysis of third-party applications, Permission additions dominate the evolution of third-party apps, of which Dangerous permissions tend to account for most of the changes. a method for detecting malware based on three metrics, which evaluate: the occurrences of a specific subset of system calls, a weighted sum of a subset of permission that the application required, and a set of combinations of permissions.

1.8 Feature Extraction

A new method to detect malicious Android applications through machine learning techniques by analyzing the extracted permissions from the application itself. Features used to classify are the presence of tags uses-permission and uses-feature into the manifest as well as the number of permissions of each application. These features are the permission requested individually and the «uses- feature» tag the possibility of detection malicious Android applications based on permissions and 20 features from Android application packages.

1.9 Classification

by combining results from various classifiers, it can be a quick filter to identify more suspicious applications. And propose a framework that intends to develop a machine learning-based malware detection system on Android to detect malware applications and to enhance security and privacy of smart-phone users. This system monitors various permission-based features and events obtained from the android applications, and analyses these features by using machine learning classifiers to classify whether the application is benign or malware. Once, the Support Vector Machine trained online on a dedicated system and only it is transferred the learned model to the smart-phone for detecting malicious applications.

2 REQUIREMENTS

2.1 Software Requirements

Software requirements is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software. The IEEE Standard Glossary of Software Engineering Terminology defines a requirements:

- A condition or capability needed by a user to solve a problem or achieve anobjective.
- A condition or capability that must be met or possessed by a system or system componentto satisfy a contract, standard, specification, or other formally imposed document.
- A documented representation of a condition or capability as in 1or 2

The activities related to working with software requirements can broadly be broken down into elicitation, analysis, specification, and management.

The software requirements specify the use of all required software products like data management system.

The required software product specifies the numbers and version. Each interface specifiesthe purpose of the interfacing software as related to this software product.

A software requirement can be of 3 types:

- Functional requirements: These are the requirements that the end user specifically Demands as basic facilities that the system should offer.
- Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract.
- Domain requirements: These are the requirements which are characteristic of a particular category or domain of projects

This project requirements are:

- **Programming Language –** Python Language.
- **Editor** – Anaconda Navigator.
- **Packages** – numpy, pandas, matplotlib, sklearn, seaborn, flask.

2.2 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A **hardware requirements** list isoften accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspectsof hardware requirements.

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

Operating system - windows 7

Processor - Intel core i5

Memory - RAM 4GB

Storage - 1TB

2.3 Technology Description

2.3.1 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. Pythonis Interactive − you can actually sit at a Python prompt and interact with the interpreterdirectly 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.

Maintain ability also ties into this may be an all but useless metric, but it does say something abouthow 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 canpick 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.

2.3.2 Html

HTML stands for Hypertext Markup Language, and it is the most widely used language to write WebPages.

- Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus,the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a Markup Language which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings,paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

Now, HTML is being widely used to format web pages with the help of different tags available inHTML language**.**

2.3.3 CSS

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify theprocess of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, and variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTMLdocument. Most commonly, CSS is combined with the markup languages HTML or XHTML.

JavaScript

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as Live Script, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape
in 1995 with the name Live Script. The general-purpose core of the language has been embeddedin Netscape, Internet Explorer, and other web browsers.

The ECMA-262 Specification defined a standard version of the core JavaScript language.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.

- Complementary to and integrated with Java.
- Complementary to and integral integrated with HTML.
- Open and cross-platform.

2.3.4 Pandas

Pandas is quite a game changer when it comes to analyzing data with Python and it is one of the most preferred and widely used tools in data wrangling if not the most used one. Pandas is an opensource.
What’s cool about Pandas is that it takes data (like a CSV or TSV file, or a SQL database) and creates aPython object with rows and columns called data frame that looks very similar to table in a statistical software (think Excel or SPSS for example. People who are familiar with R would see similarities to R too). This is so much easier to work within comparison to working with lists and/or dictionaries throughfor loops or list comprehension.

2.3.5 Numpy

Numpy is general-purpose array-processing package. It provides a high-performance multidimensionalarray object, and tools for working with these arrays.
It is the fundamental package for scientific computing with Python. It contains various featuresincluding 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.

2.3.6 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 simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with Python.

2.3.7 Sklearn

In python, scikit-learn library has a pre-built functionality under sklearn. Preprocessing.

Next thing is to do feature extraction Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally, our models are trained using Classifier algorithm. We use nltk. classify module on Natural Language Toolkit library on Python. We use the labeled dataset gathered. The rest of our labeled data will be used to evaluate the models. Some machine learning algorithms were used to classify preprocessed data. The chosen classifiers were Decision tree, Support Vector Machines and Random forest. These algorithms are very popular in text classification tasks.

2.3.8 Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Data visualization is the discipline of trying to understand data by placing it in a visual context, so that patterns, trends and correlations that might not otherwise be detected can be exposed. Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has a excellent library for you.

2.3.9 Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

2.3.10 Flask

Flask is a micro web framework written in python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common frame work related tools.

Flask follows MVC Architecture

MVC Architecture

The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used in industry-standard web development framework to create scalable and extensible projects.

MVC Components

Following are the components of MVC −

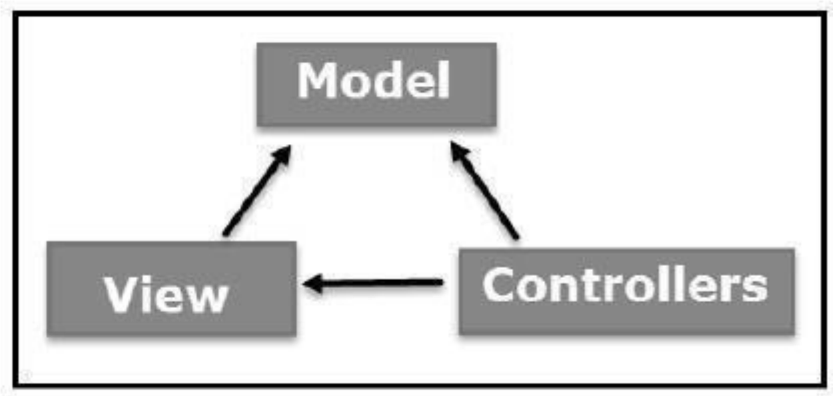


Fig 2.3.1 MVC Components

Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

View

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

2.4 Classification Algorithms

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog,** etc. Classes can be called as targets/labels or categories. Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a Supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output. In classification algorithm, a discrete output function(y) is mapped to input variable(x).

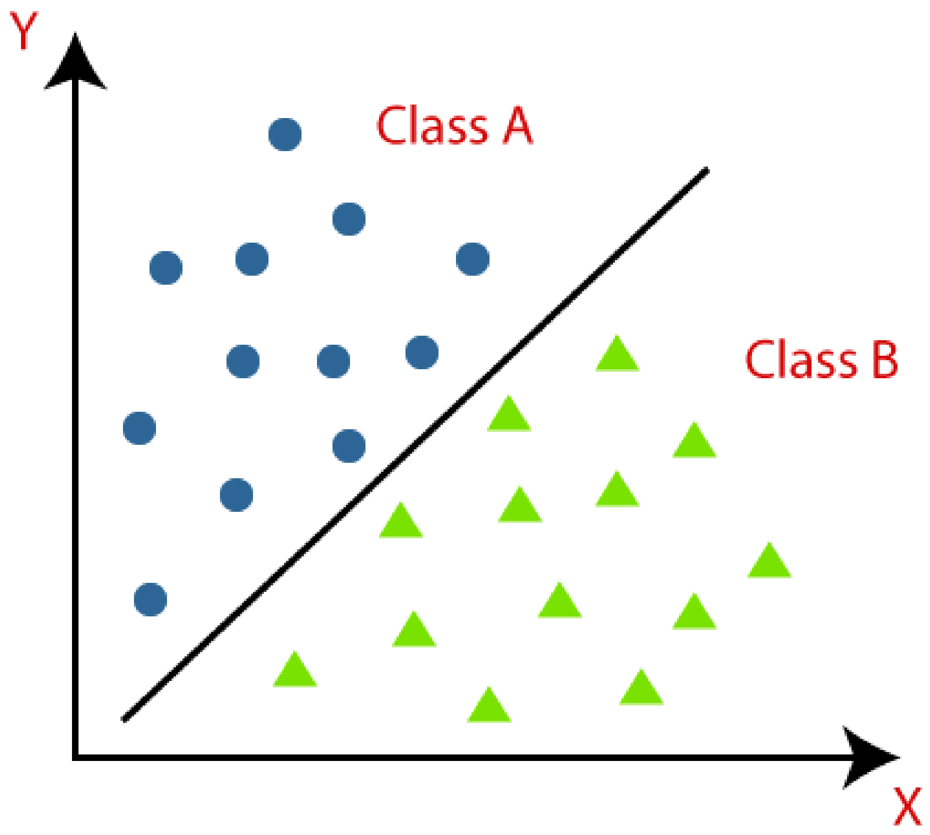


Fig. 2.4.1 Classification Example

In the proposed System, Svm, Decision tree and Naive Bayes classification algorithms are utilized to find the algorithm that best classifies the malicious applications from benign applications.

2.4.1 Support Vector Machine Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n- dimensional space into classes so that we can easily put the new data point in the correct category in thefuture. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases arecalled as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundaryor hyperplane

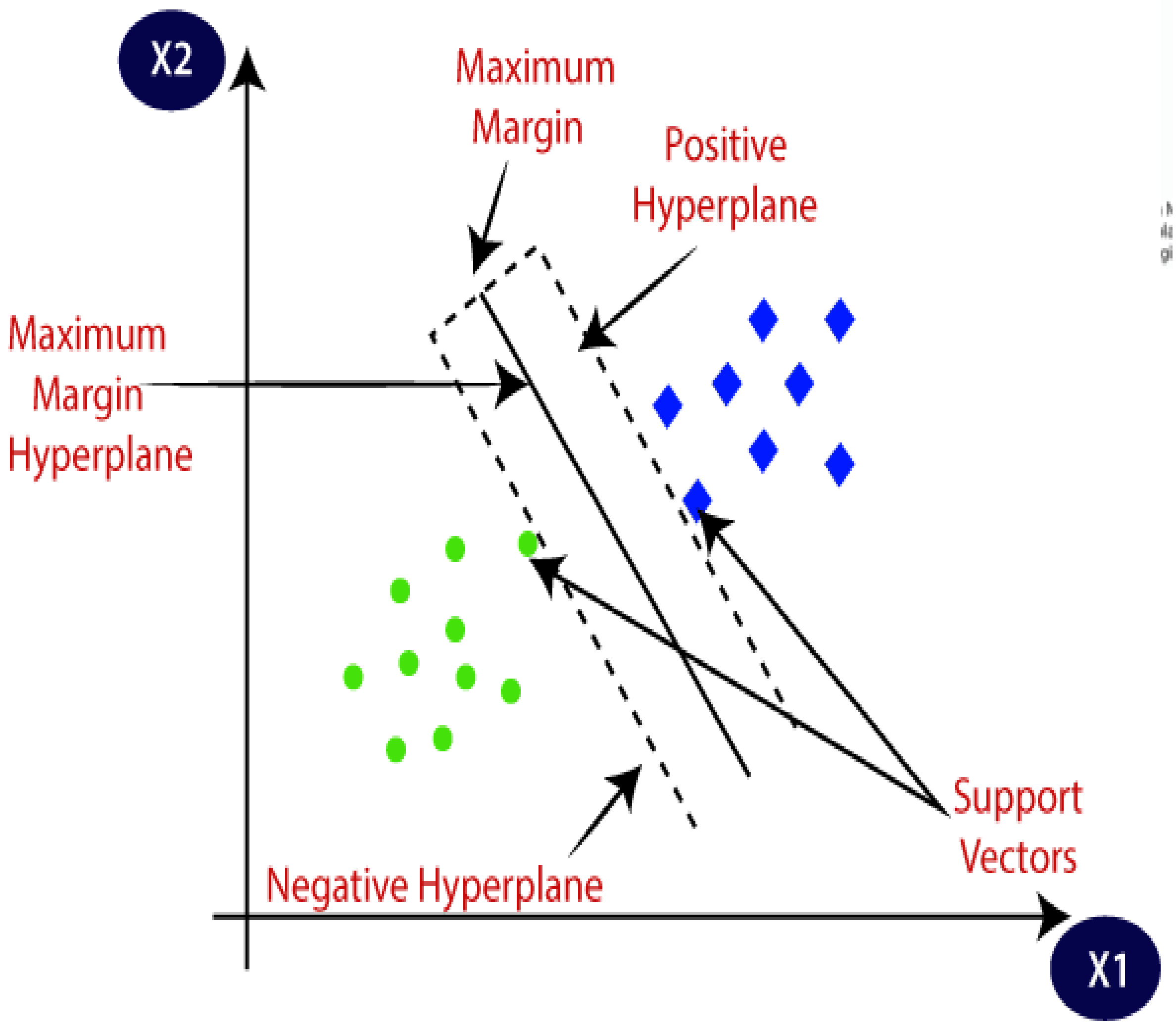


Fig 2.4.2 Support Vector Machine Algorithm

Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:

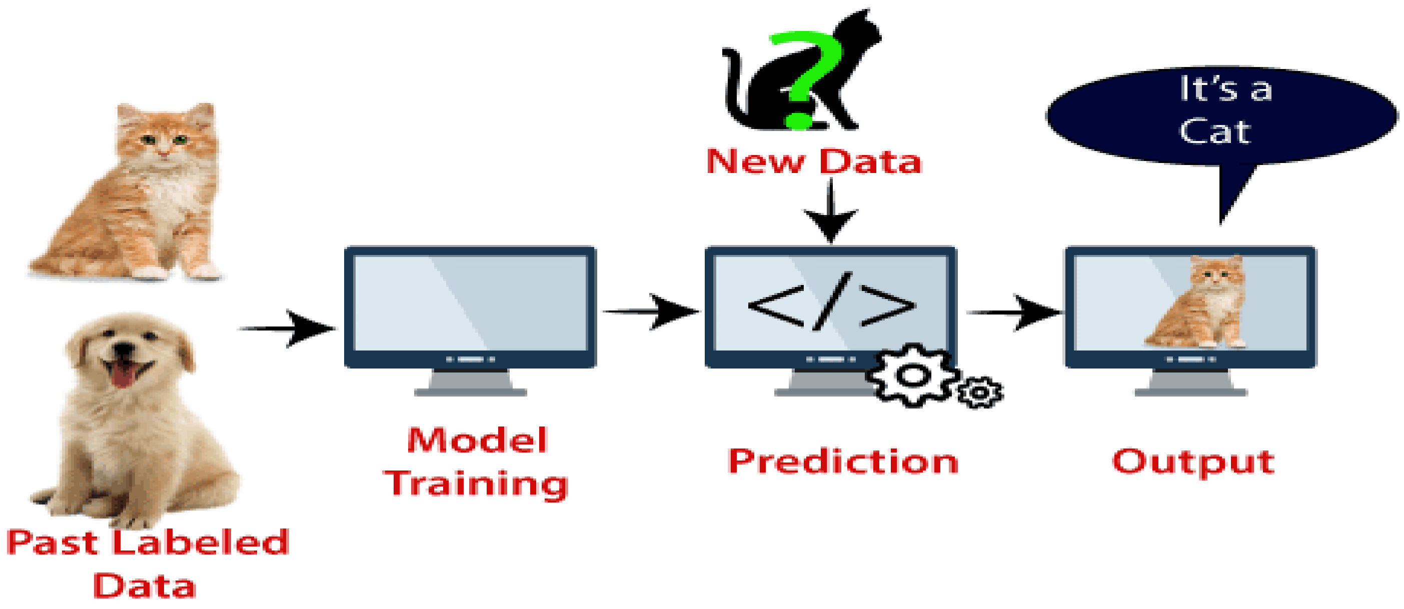


Fig 2.4.2 Support Vector Machine Algorithm Example

SVM algorithm can be used for **Face detection, image classification, text categorization**.

2.4.2 Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem andused for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms whichhelps in building the fast machine learning models that can make quick predictions.

- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.**
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis,and classifying articles**.**

Bayes' Theorem:

- Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesisis true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Convert the given dataset into frequency tables.

Generate Likelihood table by finding the probabilities of given features.Now, use Bayes theorem to calculate the posterior probability.

2.4.3 Decision Tree Algorithm

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display analgorithm that only contains conditional control statements. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning. It is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), eachbranch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

In decision analysis, a decision tree and the closely related influence diagram are used as a visual andanalytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated.

A decision tree consists of three types of nodes:

1. Decision nodes – typically represented by squares
2. Chance nodes – typically represented by circles
3. End nodes – typically represented by triangles

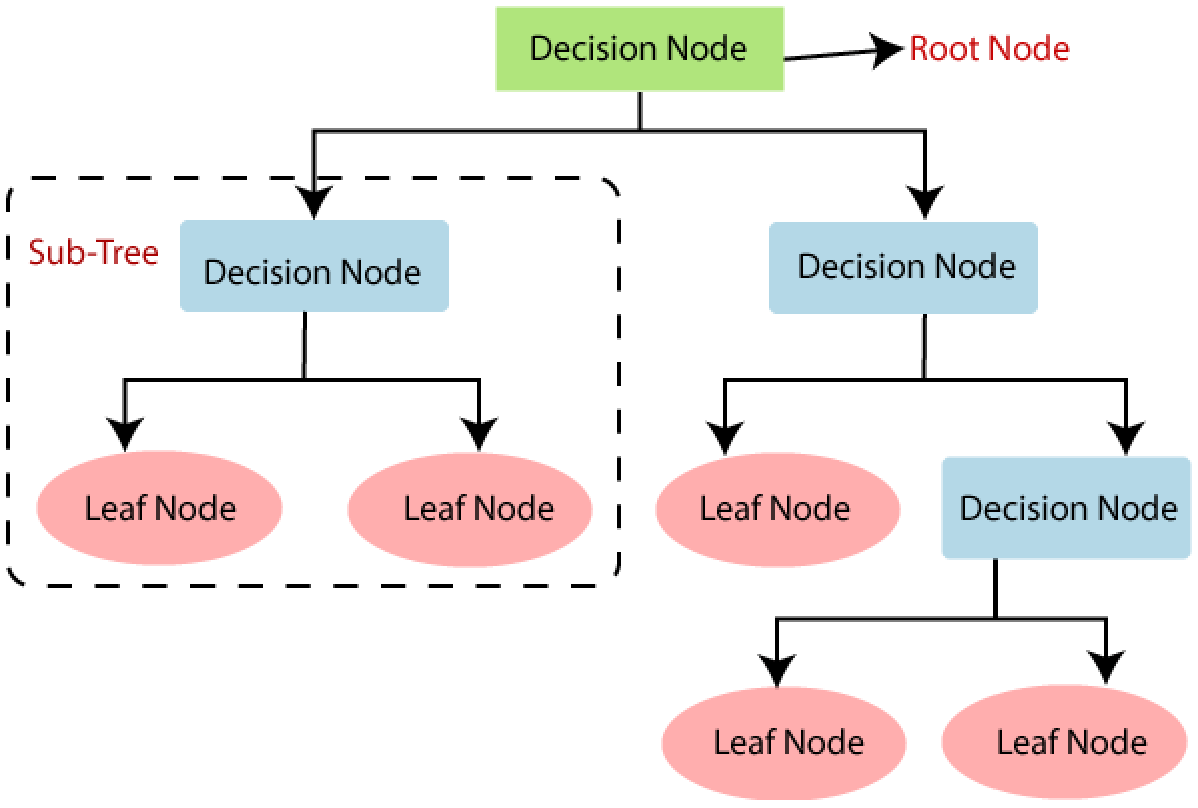


Fig 2.4.3 Decision Tree Algorithm

Working

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure(ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step - 3.Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

3. DESIGN

3.1 Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

3.2 Architecture Diagram

Web applications are by nature distributed applications, meaning that they are programs that run on more than one computer and communicate through network or server. Specifically, web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. For the enterprise, software on potentially thousands of client computers is a key reason for their popularity. Web applications are used for web mail, online retail sales, discussion boards, weblogs, online banking, and more. One web application can be accessed and used by millions of people.

Like desktop applications, web applications are made up of many parts and often contain mini programs and some of which have user interfaces. In addition, web applications frequently require an additional markup or scripting language, such as HTML, CSS, or JavaScript programming language. Also, many applications use only the Python programming language, which is ideal because of its versatility.

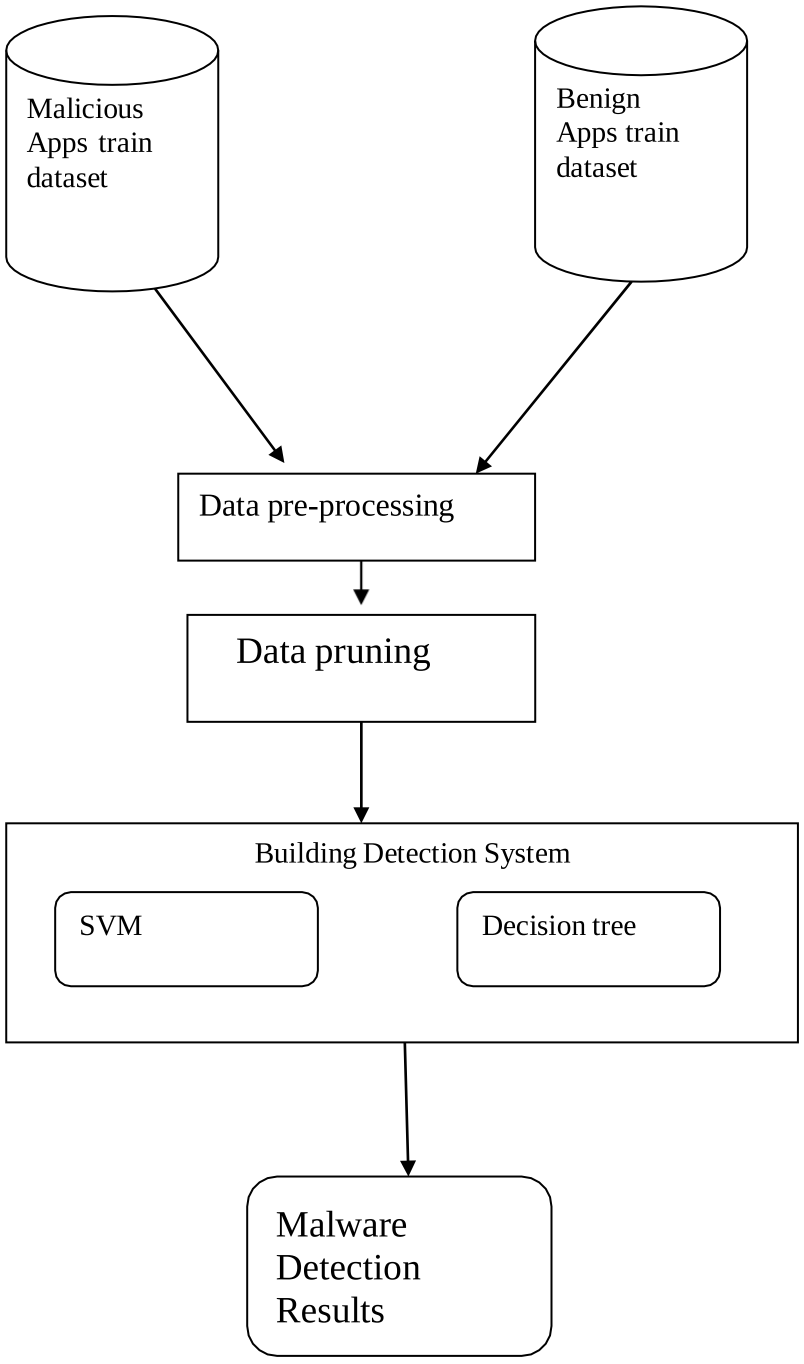

Malicious Apps train dataset
Benign Apps train dataset
Data pre-processing
Data pruning
Building Detection System
SVM
Decision tree
Malware Detection Results

Fig 3.2.1 Architecture Diagram

3.3 UML Diagrams

3.3.1 Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction. These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Hence to model the entire system, a number of use case diagrams are used

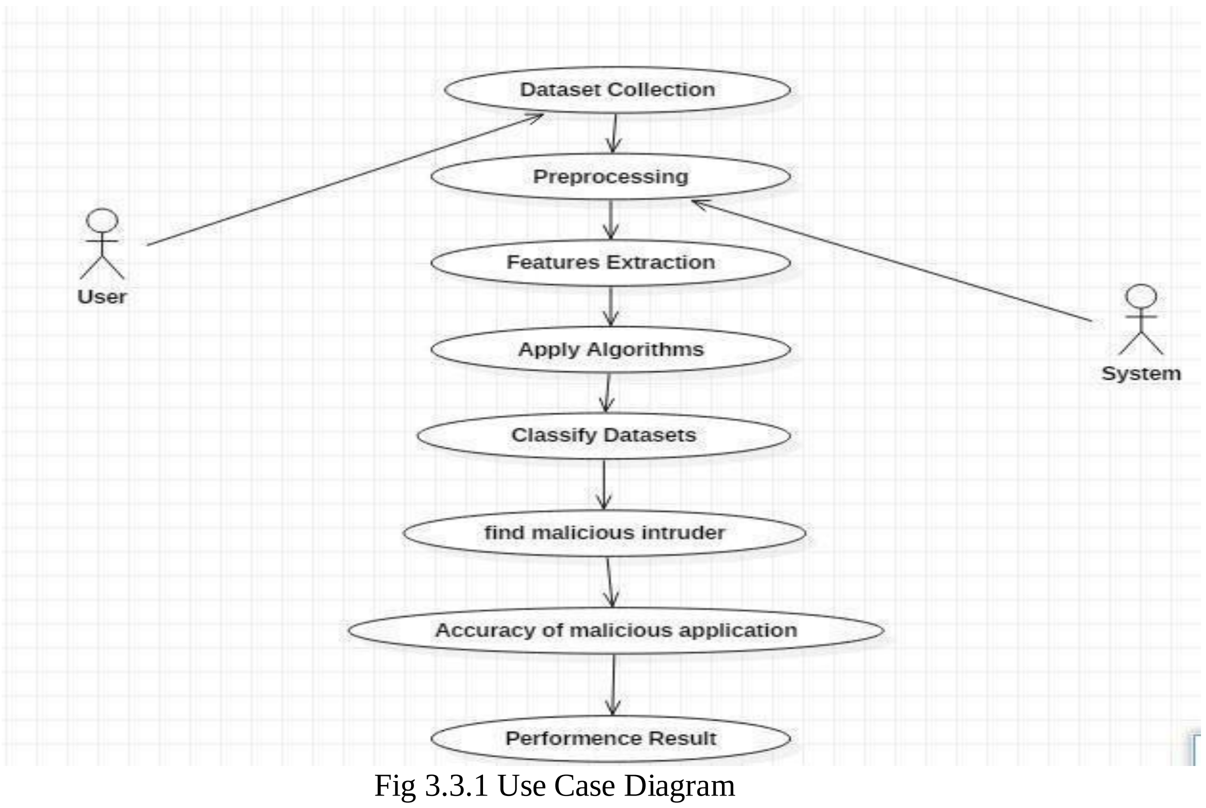


Fig 3.3.1 Use Case Diagram

3.3.2 Sequence Diagram

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors.

Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

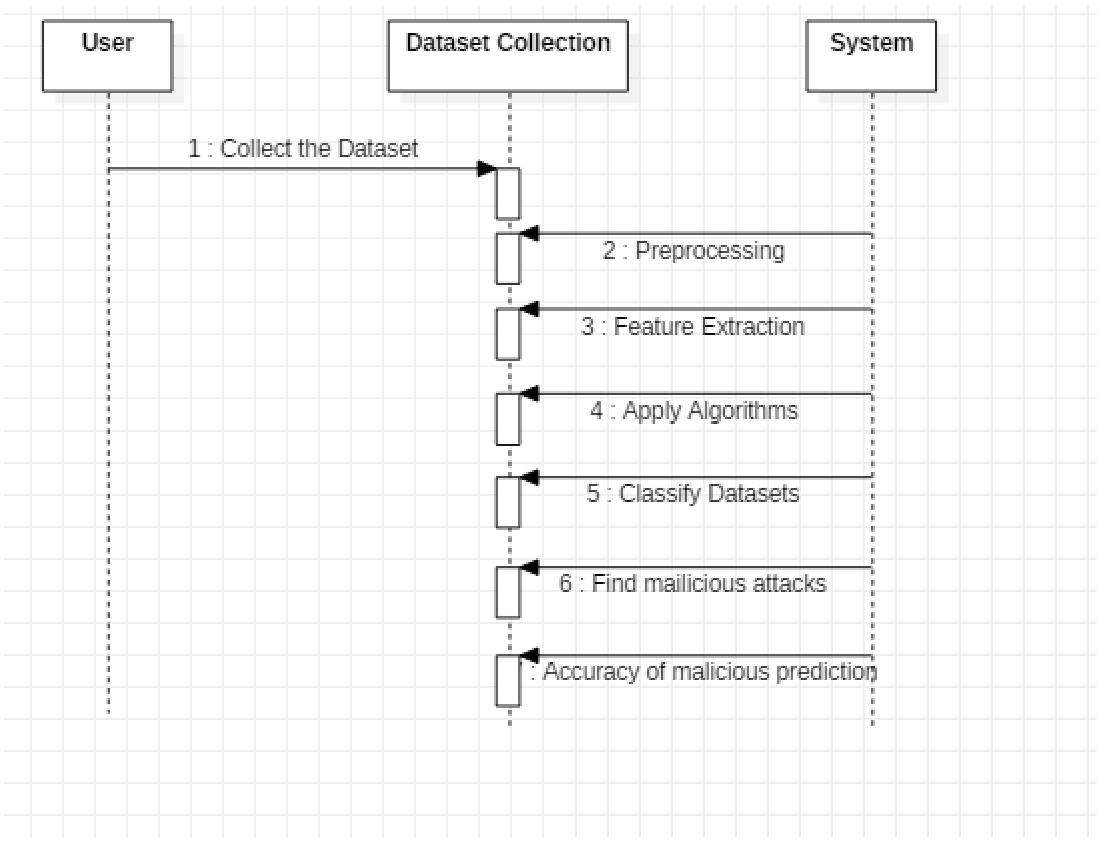


Fig 2.3.2 Sequence Diagram

3.3.3 Activity Diagram

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

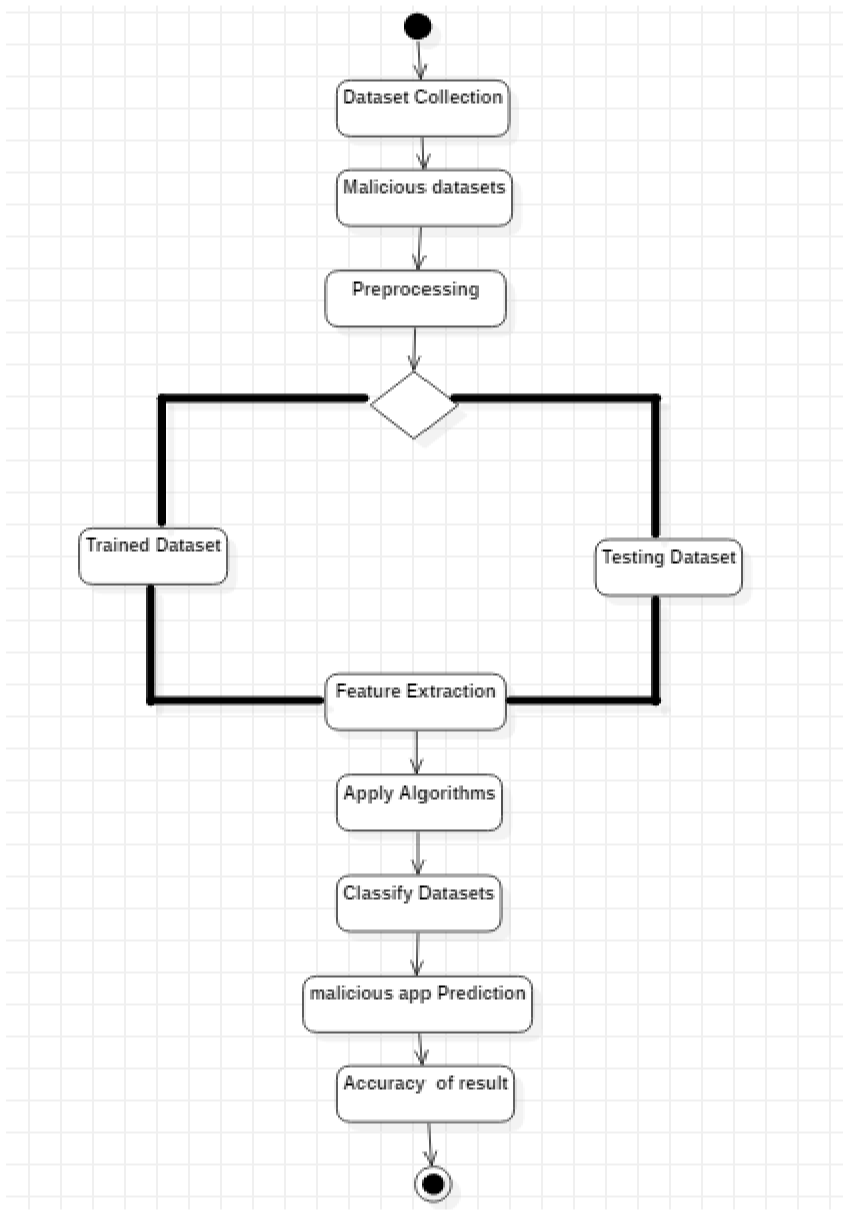


Fig 3.3.3 Activity Diagram

3.3.4 Class Diagram

The class diagram is the main building block of object-oriented modelling. It is used for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

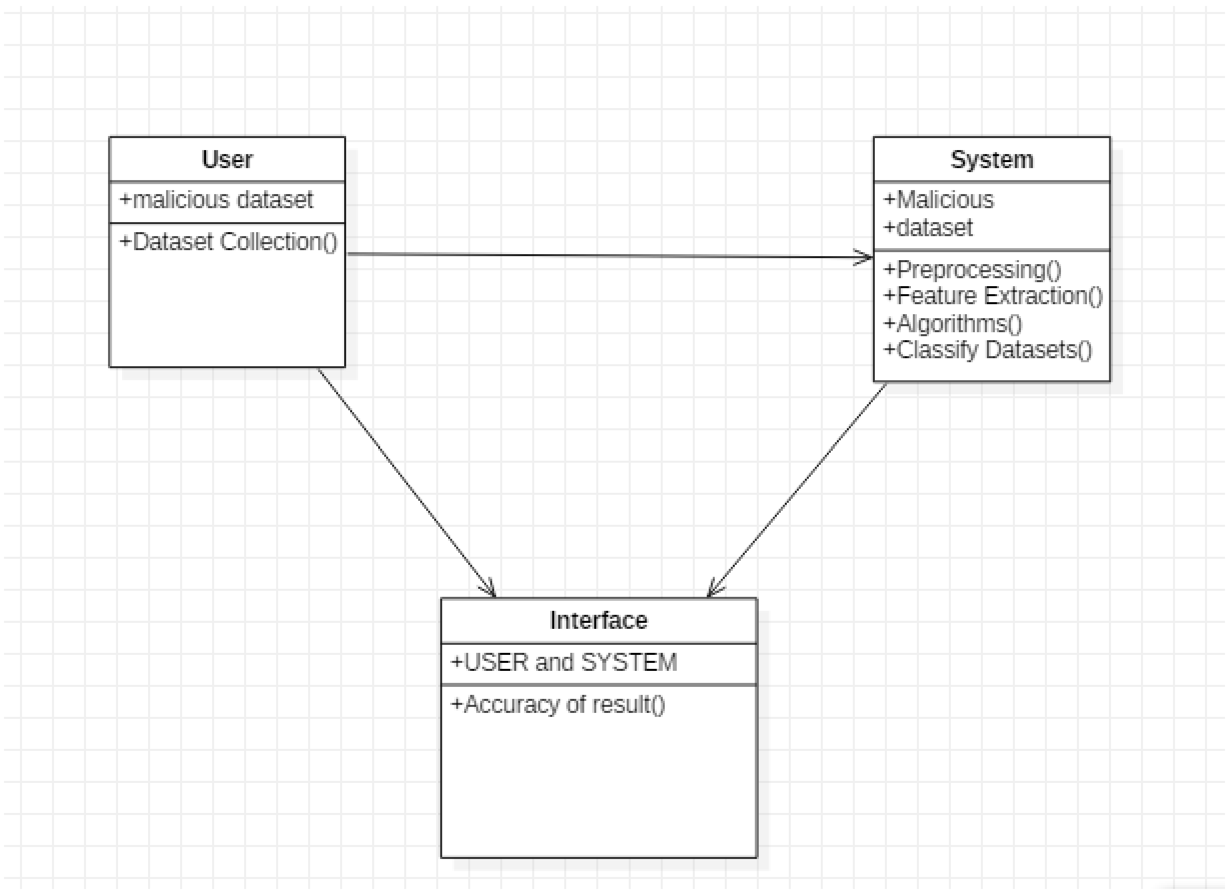


Fig 3.3.4 Class Diagram

4. IMPLEMENTATION

4.1 Coding

app.py

```
import numpy as np

import pandas as pd

from flask import Flask, request, jsonify, render

template import pickle


app = Flask(name)

model = pickle.load(open('model.pkl', 'rb'))


dataset = pd.read_csv('android.csv')


dataset_X = dataset.iloc[:,[1,2,3,4, 5,6,7,8,9, 10,11,12,13,14,15]].values



@app.route('/')

def home():

return render_template('index.html')


@app.route('/predict',methods=['POST'])

def predict():

'''

For rendering results on HTML GUI'''

float_features = [float(x) for x in request.form.values()]

final_features = [np.array(float_features)]
```

```
    prediction = model.predict( final_features )


    if prediction == 1:
        pred = "Malicious Application"
    elif prediction == 0:
        pred = "Benign Application."
    output = pred


    return render_template('index.html', prediction_text='{}'.format(output))


ifname_______== "main":
    app.run(debug=True)
```

android_malware.py

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd#cv
import numpy as np
import seaborn as sns
from sklearn.svm import SVC
from sklearn.model_selection import KFold
from sklearn import preprocessing
import matplotlib.pyplot as plt

data=pd.read_csv('android.csv')


data


data.shape


data = data.sample(frac=1).reset_index(drop=True)


data.head()


import seaborn as sns


sns.countplot(x='malware',data=data)

target_count = data.malware.value_counts()
print('Class 0:',target_count[0])
print('Class 1:',target_count[1])



count_class_0, count_class_1 = data.malware.value_counts()
```

```
 df_class_0 = data[data['malware'] == 0]
df_class_1 = data[data['malware'] == 1]

df_class_1_over = df_class_1.sample(count_class_0, replace=True)
df_test_over = pd.concat([df_class_0, df_class_1_over], axis=0)

df_test_over.shape

sns.countplot(x='malware',data=df_test_over)

X=df_test_over.iloc[:,df_test_over.columns !='malware']

Y=df_test_over.iloc[:,df_test_over.columns =="malware"]


X.head()

Y.head()

from sklearn.utils import shuffle

X, Y=shuffle(X, Y)

X.head()

X=X.drop(columns='name')
X.head()

Y.head()

from sklearn.feature_selection import
SelectKBestfrom sklearn.feature_selection import
chi2
```

```
bestfeatures = SelectKBest(score_func=chi2, k=10)

fit = bestfeatures.fit(X,Y)

dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)


featureScores = pd.concat([dfcolumns,dfscores],axis=1)

featureScores.columns = ['Specs','Score']

featureScores.nlargest(10,'Score')


from sklearn.ensemble import ExtraTreesClassifier

import matplotlib.pyplot as plt

model = ExtraTreesClassifier()

model.fit(X,Y)

print(model.feature_importances_) #use inbuilt class feature_importances of tree based

classifiers#plot graph of feature importances for better visualization

feat_importances = pd.Series(model.feature_importances_,

index=X.columns)feat_importances.nlargest(10).plot(kind='barh')

plt.show()


from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.2, random_state=0)


X_train.shape


X_train.head()


y_train.head()


from sklearn import metrics

from sklearn.metrics import confusion_matrix


from sklearn import svm

support = svm.LinearSVC(random_state=20)


support.fit(X_train,y_train)
```

```
y_pred = support.predict(X_test)
y_pred


model1=metrics.accuracy_score(y_test,y_pred)
print(model1)


cnf_matrix = confusion_matrix(y_test,y_pred)


labels = ['Good','Malware']
sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
plt.show()


from sklearn.tree import DecisionTreeClassifier


tree = DecisionTreeClassifier()


tree.fit(X_train,y_train)


y_pred = tree.predict(X_test)
y_pred


model2=metrics.accuracy_score(y_test,y_pred)
print(model2)


cnf_matrix = confusion_matrix(y_test,y_pred)


labels = [0,1]
sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
plt.show()


from sklearn.naive_bayes import GaussianNB
```

```
  clf = GaussianNB()

  clf.fit(X, Y)

  y_pred = clf.predict(X_test)
  y_pred

  model3=metrics.accuracy_score(y_test,y_pred)
  print(model3)

  cnf_matrix = confusion_matrix(y_test,y_pred)

  labels = [0,1]
  sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
  plt.show()

import matplotlib.pyplot as plt; plt.rcdefaults()
import numpy as np
import matplotlib.pyplot as plt

objects = ('SVM','DecisionTreeClassifier','GaussianNB')
y_pos = np.arange(len(objects))
performance = [model1,model2,model3]

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy Score')
plt.title('SVM vs DecisionTreeClassifier vs Naive Bayes')

plt.show()
```

4.2 Data Collection and Balancing data

We collected data comparing the true and false positive rate of our classifier, shown below.

```
import seaborn as sns
```

```
sns.countplot(x='malware',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xf6ffa70>
```

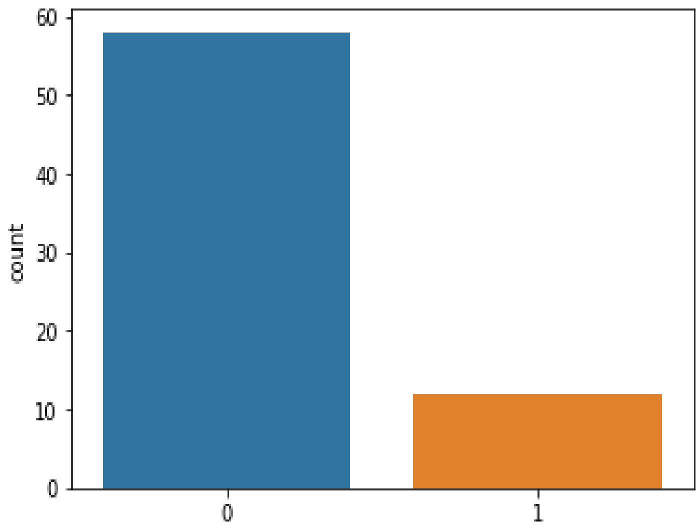


Fig 4.2.1 Data Collection and Balancing

The data is unbalanced , Therefore we balanced data

```
target_count = data.malware.value_counts()
print('Class 0:', target_count[0])
print('Class 1:', target_count[1])
```

```
Class 0: 58
Class 1: 12
```

```
count_class_0, count_class_1 = data.malware.value_counts()
```

```
df_class_0 = data[data['malware'] == 0]
df_class_1 = data[data['malware'] == 1]
```

```
df_class_1_over = df_class_1.sample(count_class_0, replace=True)
df_test_over = pd.concat([df_class_0, df_class_1_over], axis=0)
```

```
df_test_over.shape
```

```
sns.countplot(x='malware',data=df_test_over)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x103021f0>
```

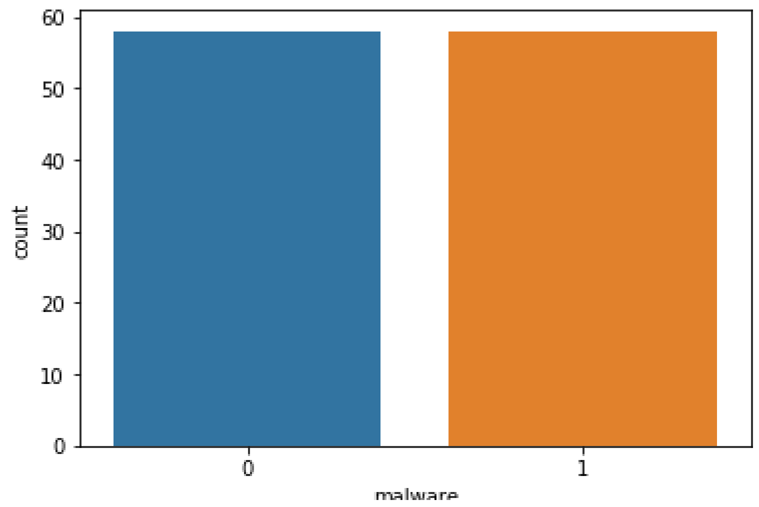
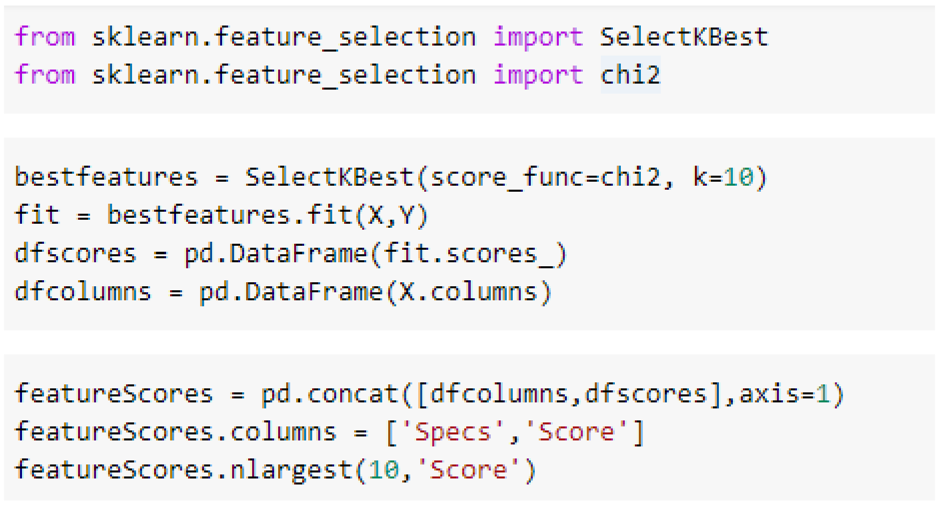


Fig 4.2.2 Balanced Data

4.3 Feature Selection

using selectkBest we have selected the best attributes/ permissions/ features from the given attributes

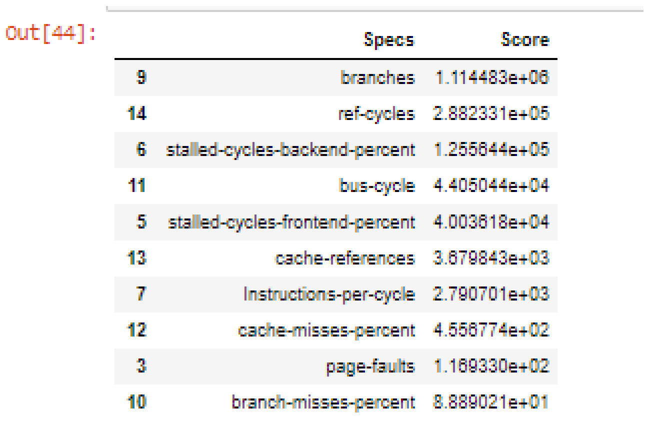


```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

```
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,Y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
```

```
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']
featureScores.nlargest(10,'Score')
```

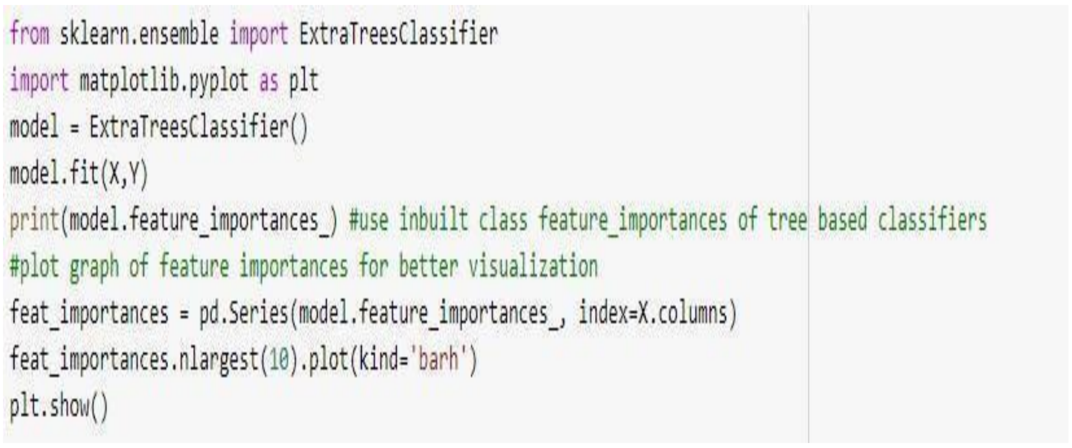
Fig 4.3.1 Feature Selection



Out[44]:

	Specs	Score
9	branches	1.114483e+06
14	ref-cycles	2.882331e+05
6	stalled-cycles-backend-percent	1.255644e+05
11	bus-cycle	4.405044e+04
5	stalled-cycles-frontend-percent	4.003618e+04
13	cache-references	3.679843e+03
7	Instructions-per-cycle	2.790701e+03
12	cache-misses-percent	4.556774e+02
3	page-faults	1.169330e+02
10	branch-misses-percent	8.889021e+01

Fig 4.3.2 Feature Selection output



```
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,Y)
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```

```
[0.06391337 0.04524854 0.01839569 0.06943939 0.02815262 0.01992447
 0.03237499 0.01862666 0.2106383  0.10583143 0.12077583 0.08900648
 0.08174854 0.04222666 0.05369705]
```

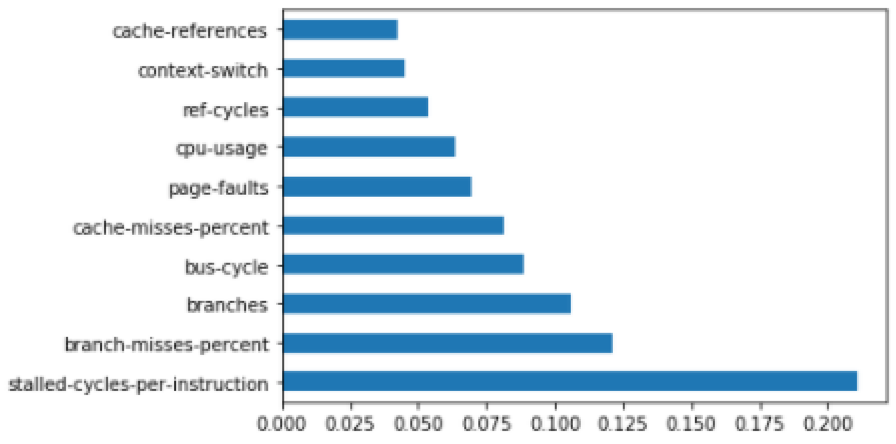
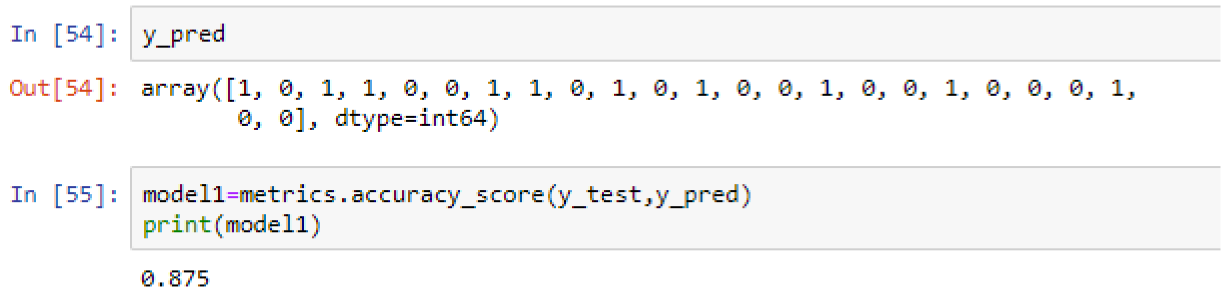


Fig 4.3.3 Feature Selection Graph

4.4 Building SVM Model

After building the svm model, we have trained and tested the data using the built svm model and obtained a accuracy of 87.5



```
In [54]: y_pred

Out[54]: array([1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
                0, 0], dtype=int64)

In [55]: model1=metrics.accuracy_score(y_test,y_pred)
         print(model1)

         0.875
```

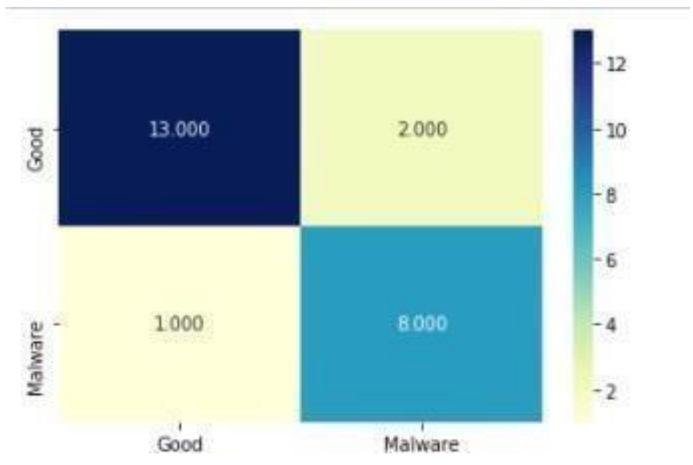


Fig 4.4.1 Building SVM Model

4.5 Building Decision Tree Classification Model

After building the Decision tree model, we have trained and tested the data using the built decision tree model and obtained a accuracy of 95.8

```
In [61]: y_pred = tree.predict(X_test)
         y_pred

Out[61]: array([1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
                1, 0], dtype=int64)

In [62]: model2=metrics.accuracy_score(y_test,y_pred)
         print(model2)

         0.9583333333333334

In [63]: cnf_matrix = confusion_matrix(y_test,y_pred)

In [64]: labels = [0,1]
         sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
         plt.show()
```

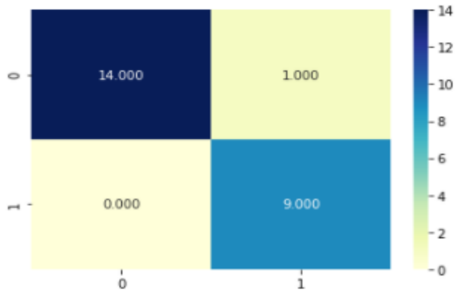


Fig 4.5.1 Building Decision Tree Classification Model

4.6 Building Naive Bayes classification Model

After building the Bayesian model, we have trained and tested the data using the built Bayesian model and obtained a accuracy of 54.1

```
In [68]: y_pred = clf.predict(X_test)
         y_pred

Out[68]: array([1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
                1, 0], dtype=int64)

In [69]: model3=metrics.accuracy_score(y_test,y_pred)
         print(model3)

         0.5416666666666666

In [70]: cnf_matrix = confusion_matrix(y_test,y_pred)

In [71]: labels = [0,1]
         sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
         plt.show()
```

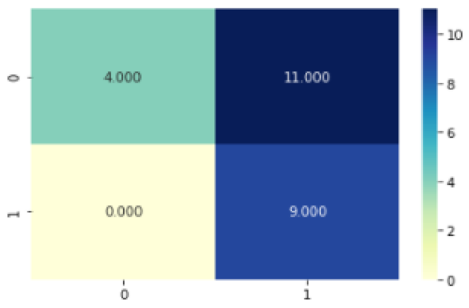


Fig 4.6.1 Building Naïve Bayes Classification Model

4.7 SVM vs Decision Tree vs Naive Bayes

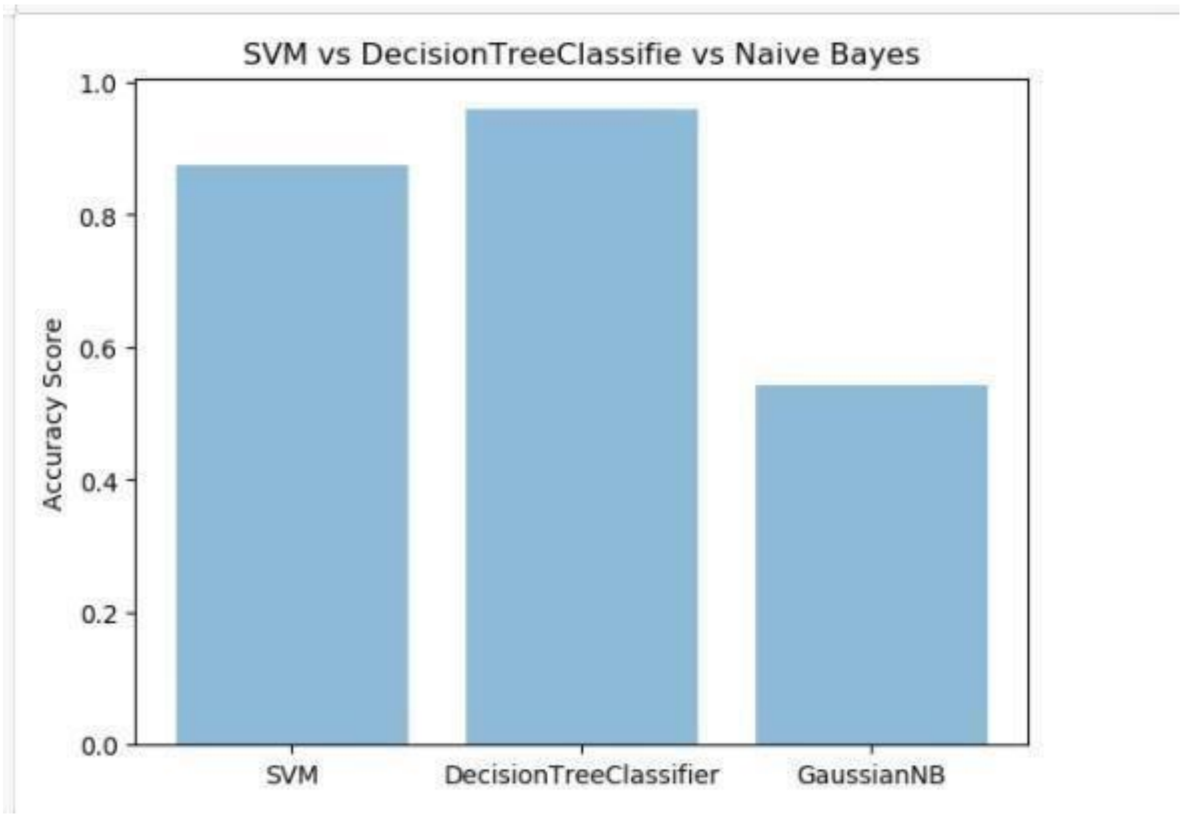


Fig 4.7.1 SVM vs Decision Tree vs Naïve Bayes

5.Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.
There of basically two types of testing approaches.

One is Black-Box testing – the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operated.
The other is White-Box testing – knowing the internal workings of the product, tests canbe conducted to ensure that the internal operation of the product performs according to specifications and all internal components have been adequately exercised.

White box and Black box testing methods have been used to test this package. The entireloop constructs have been tested for their boundary and intermediate conditions. The test data was designed with a view to check for all the conditions and logical decisions. Error handling has been taken care of by the use of exception handlers.

5.1 Testing Strategies

Testing is a set of activities that can be planned in advanced and conducted systematically. A strategy for software testing must accommodation low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements as specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

The main objective of software is testing to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed. Each test step is accomplished through a series of system static test technique that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened.

Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase. This is an activity to be performed in parallel with the software effort and one that consists of its own phases of analysis, design, implementation, execution and maintenance.

5.1.1 Unit Testing

This testing method considers a module as single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level. Here the module will be treated as a black box, which will take some input and generate output. Outputs for a given set of input combination are pre-calculated and are generated by the module.

5.1.2 System Testing

Here all the pre tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchronous with each other. This testing methodology helps in making sure that all modules which are running perfectly when checked individually are also running in cohesion with other modules. For this testing we create test cases to check all modules once and then generated test combinations of test paths throughout the system to make sure that no path is making its way into chaos.

5.1.3 Integrated Testing

Testing is a major quality control measure employed during software development. Its basic function is to detect errors. Sub functions when combined may not produce than it is desired. Global data structures can represent the problems. Integrated testing is a systematic technique for constructing the program structure while conducting the tests. To uncover errors that are associated with interfacing the objective is to make unit test modules and built a program structure that has been detected by design. In a non - incremental integration all the modules are combined in advance and the program is tested as a whole. Here errors will appear in an endless loop function. In incremental testing the program is constructed and tested in small segments where the errors are isolated and corrected.

Different incremental integration strategies are top – down integration, bottom – up integration, regression testing.

5.1.4 Regression Testing

Each time a new module is added as a part of integration as the software changes. Regression testing is an actually that helps to ensure changes that do not introduce unintended behavior as additional errors.

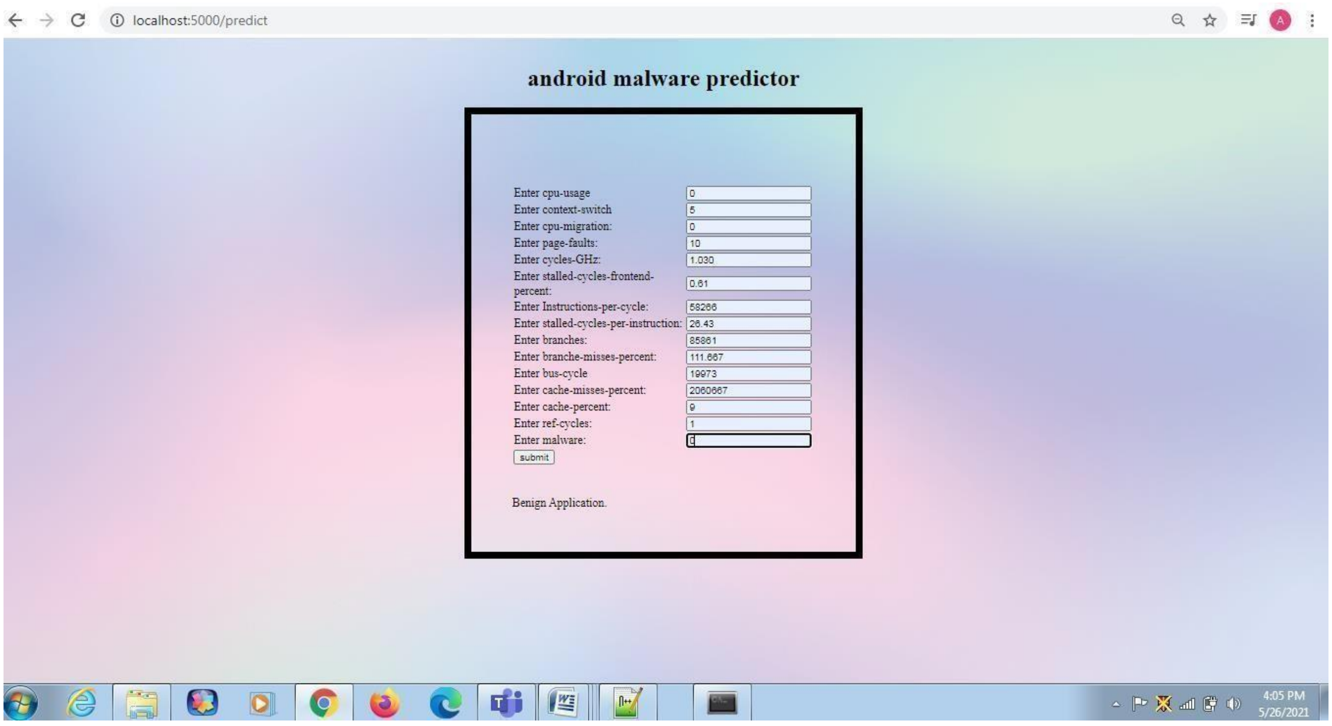
Regression testing maybe conducted manually by executing a subset of all test cases or using automated capture play back tools enables the software engineer to capture the test case and results for subsequent playback and compression. The regression suit contains different classes of test cases.

A representative sample to tests that will exercise all software functions.

Additional tests that focus on software functions that are likely to be affected by the change.

5.2 Test Cases

Integrated and regression testing strategies are used in this application for testing.



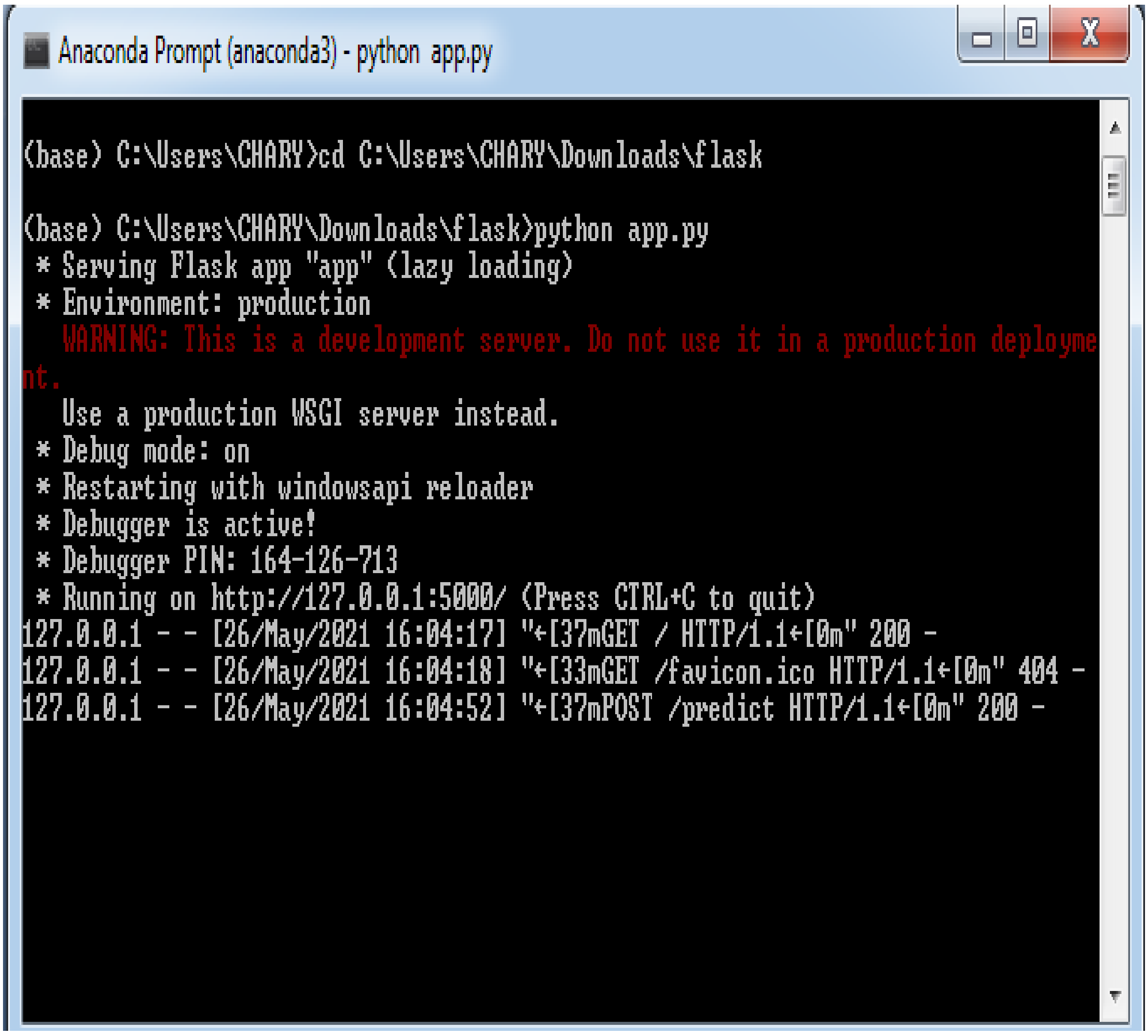
android malware predictor

Enter cpu-usage	0
Enter context-switch	5
Enter cpu-migration:	0
Enter page-faults:	10
Enter cycles-GHz:	1.030
Enter stalled-cycles-frontend-percent:	0.61
Enter Instructions-per-cycle:	58266
Enter stalled-cycles-per-instruction:	26.43
Enter branches:	85861
Enter branche-misses-percent:	111.667
Enter bus-cycle	19973
Enter cache-misses-percent:	2060667
Enter cache-percent:	9
Enter ref-cycles:	1
Enter malware:	0

Benign Application.

Fig 5.3.1 Test Cases

5.3 Execution Screenshots



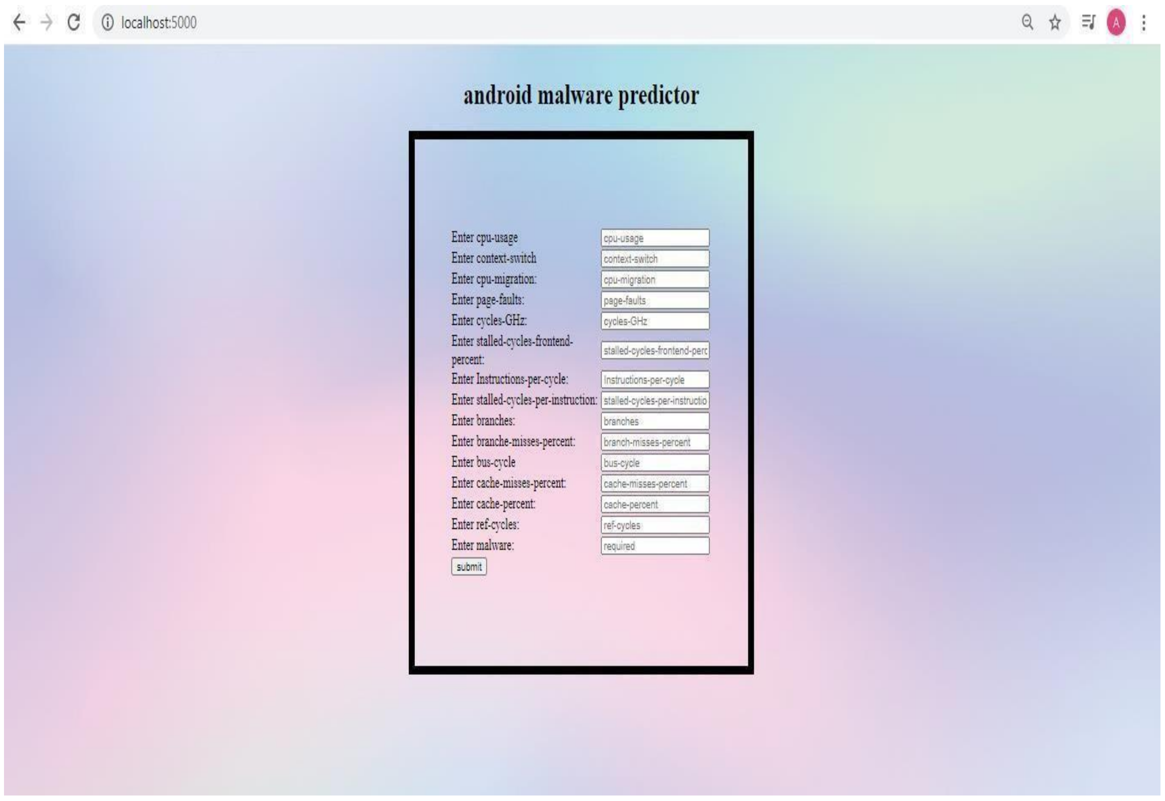
Anaconda Prompt (anaconda3) - python app.py

```
(base) C:\Users\CHARY>cd C:\Users\CHARY\Downloads\flask

(base) C:\Users\CHARY\Downloads\flask>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 164-126-713
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [26/May/2021 16:04:17] "←[37mGET / HTTP/1.1←[0m" 200 -
127.0.0.1 - - [26/May/2021 16:04:18] "←[33mGET /favicon.ico HTTP/1.1←[0m" 404 -
127.0.0.1 - - [26/May/2021 16:04:52] "←[37mPOST /predict HTTP/1.1←[0m" 200 -
```

Fig 5.4.1 Execution Screenshots

5.3.1 Dashboard

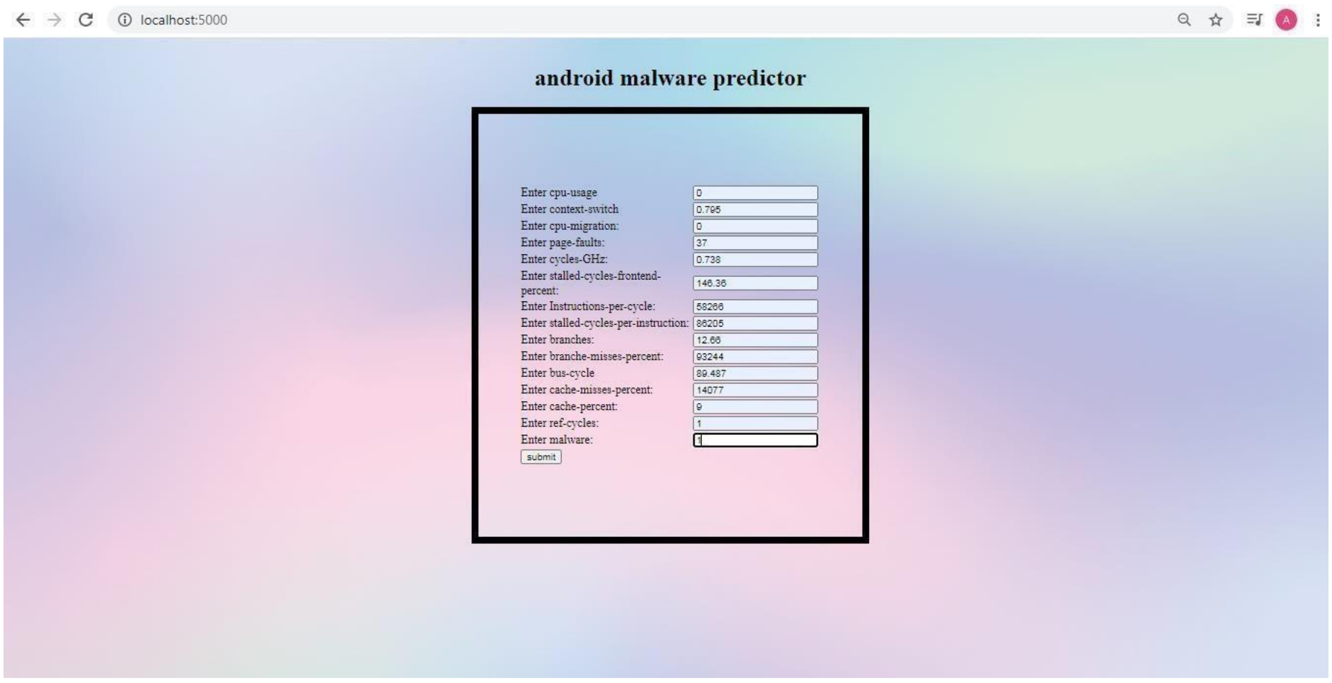

localhost:5000

android malware predictor

Enter cpu-usage	cpu-usage
Enter context-switch	context-switch
Enter cpu-migration:	cpu-migration
Enter page-faults:	page-faults
Enter cycles-GHz:	cycles-GHz
Enter stalled-cycles-frontend-percent:	stalled-cycles-frontend-perc
Enter Instructions-per-cycle:	Instructions-per-cycle
Enter stalled-cycles-per-instruction:	stalled-cycles-per-instructio
Enter branches:	branches
Enter branche-misses-percent:	branch-misses-percent
Enter bus-cycle	bus-cycle
Enter cache-misses-percent:	cache-misses-percent
Enter cache-percent:	cache-percent
Enter ref-cycles:	ref-cycles
Enter malware:	required

Fig 5.4.2 Dashboard

5.3.2 INPUT:

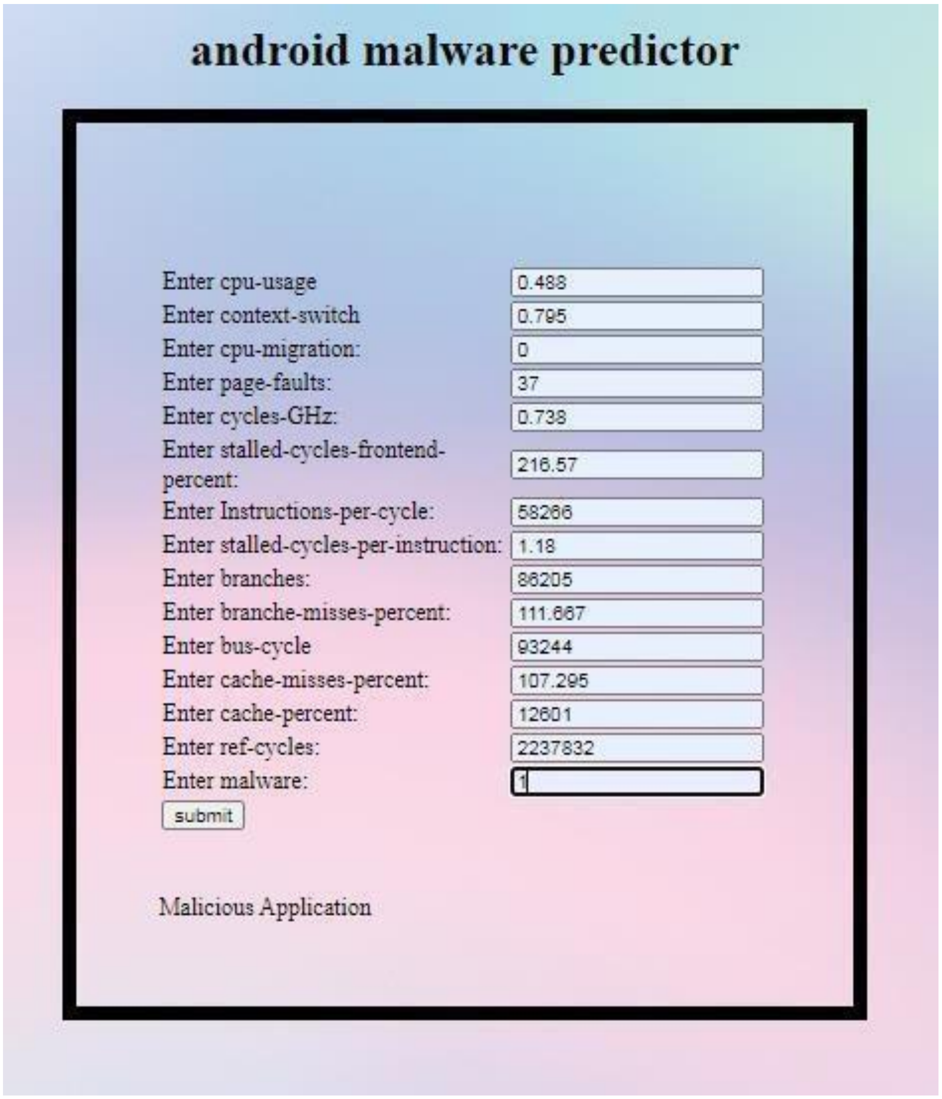


android malware predictor

Enter cpu-usage	0
Enter context-switch	0.795
Enter cpu-migration:	0
Enter page-faults:	37
Enter cycles-GHz:	0.738
Enter stalled-cycles-frontend-percent:	146.36
Enter Instructions-per-cycle:	58266
Enter stalled-cycles-per-instruction:	86205
Enter branches:	12.66
Enter branche-misses-percent:	93244
Enter bus-cycle	89.487
Enter cache-misses-percent:	14077
Enter cache-percent:	9
Enter ref-cycles:	1
Enter malware:	1

Fig 5.4.3 Input

5.3.4 OUTPUT:



android malware predictor

Enter cpu-usage	0.488
Enter context-switch	0.795
Enter cpu-migration:	0
Enter page-faults:	37
Enter cycles-GHz:	0.738
Enter stalled-cycles-frontend-percent:	216.57
Enter Instructions-per-cycle:	58266
Enter stalled-cycles-per-instruction:	1.18
Enter branches:	86205
Enter branche-misses-percent:	111.667
Enter bus-cycle	93244
Enter cache-misses-percent:	107.295
Enter cache-percent:	12601
Enter ref-cycles:	2237832
Enter malware:	1

Malicious Application

Fig 5.4.4 Output

Conclusion and Future Work

In conclusion, our project can identify, with moderate success, applications that pose a potential threat based on the permissions that they request. Our application can scan applications on a phone at any time, and alerts the user to do so when an installation or app update occurs. We believe that this is an important step in preventing Android malware, because this application brings to the user's attention all the possibly dangerous applications, allowing them to scrutinize the applications that they trust more carefully. This in turn will help users become more security-conscious overall.

Even so, this is only a first step. Future work for this project will include increasing the accuracy of the classifier, migrating the Python portions of this project to Java, and integrating more advanced methods of detecting malicious behavior such as looking at API calls (this follows a "defense in depth" strategy). One benefit of the decision tree classifier is its speed. It can serve as a preliminary screen for more advanced but slower methods, to focus the applications they will inspect. Lastly, taking into account application categories such as being a game or email-client would also help detect suspicious permissions and behaviors.

References

1. A. P. Felt, K. Greenwood, and D. Wagner, “The effectiveness of install-time permission systems forthird-partyapplications”,2010.
2. B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, “Android permissions: aperspective combining risks and benefits,” 2012.
3. Y. Zhou and X. Jiang, “Dissecting android malware: Characterization andevolution,2012.
4. V. Rastogi, Y. Chen, and X. Jiang, “Droidchameleon: evaluating android anti-malware againsttransformation attacks, 2013.
5. G. Canfora, F. Mercaldo, and C. A. Visaggio, “A classifier of malicious android applications,”2013.
6. B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. A´ lvarez, “Puma:Permission usage to detect malware in android,”,2013.
7. C.-Y. Huang, Y.-T. Tsai, and C.-H. Hsu, “Performance Evaluation on Permission-Based Detectionfor Android Malware,”2013.
8. Franklin Tchakount´, Computers & Security “Permission-based Malware Detection Mechanisms onAndroid: Analysis and Perspectives”,2014.
9. Z. Fang, W. Han, and Y. Li, “Permission-based Android security: Issues and counter measures,”
10. W. Enck, M. Ongtang, P. McDaniel, "Understanding Android Security"
11. Aung, Zarni, and Win Zaw. "Permission-based android malware detection."International Journal ofScientific and Technology Research2.3 (2013): 228-234.
12. Urcuqui, C., and A. Cadavid. "Machine learning classifiers for Android malware analysis."Proceedings of the IEEE Colombian Conference on Communications and Computing.2016.
13.