**HAZARD IDENTIFICATION AND DETECTION USING MACHINE LEARNING APPROACH**

**PROBLEM STATEMENT:**

Predicting if a given website is malicious or benign using machine learning classifiers are Decision Tree, XGBoost, KNN and Adaboost.

**ABSTRACT:**

Internet surfing has become a vital part of our daily life. So to catch the attention of the users' different browser vendors compete to set up the new functionality and advanced features that become the source of attacks for the intruder and the websites are put at hazard. However, the existing approaches are not adequate to protect the surfers which require an expeditious and precise model that can be able to distinguish between the benign or malicious webpages. In this research article, we design a new classification system to analyse and detect the malicious web pages using machine learning classifiers such as, random forest, support vector machine. naïve Bayes, logistic regression and Some special URL (Uniform Resource Locator) based on extricated features the classifiers are trained to predict the malicious web pages. The experimental results have shown that the performance of the random forest classifier achieves better accuracy of 95% in comparison to other machine learning classifiers.

**INTRODUCTION**

With the rapid development of the web, more and more services like internet banking, e-commerce, social networking, shopping, making a bill payment, e-learning, etc. are available to users and they are surfing the internet via browsers or web application. As the browsers are come up with different advanced features and functionalities which leads to risk by losing their personal and sensitive information.

As the naïve users are not aware of the different malware so they are easily trapped by the intruder by just a single click on the malicious web sites which allows the invaders to detect the vulnerabilities on the web page and inject the payloads to get remote access to victim’s web page. Therefore, the precise identification of web pages in an ever-growing web environment is very important. Blacklisting services were embedded in the browsers to face the challenges but it has several disadvantages like incorrect listing. In this article, we explore a self-learning approach to classify the web page based on a small feature set. We use four machine learning classifiers to classify the web site into two classes benign and malicious web pages.

“The rest of the research work is planned as follows: Section II presents related work, the methodology is discussed in section III, experimental result analysis is depicted in Section IV and Section V contains the conclusion of the research work and suggests some future work”.

II. RELATED WORK

To identify the web pages that are malicious, three different techniques i.e. blacklisting, static analysis, and dynamic analysis are suggested by research practitioners. Each approach has some objective to satisfy and we have discussed some of these techniques sequentially.

Tao et al. [1] presented a novel framework for detecting the web page as malicious or benign automatically using supervised machine learning approaches. The web pages were distinguished as malicious or not based upon features. Benign web pages were collected from dataset

Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2020) IEEE Explore Part Number: CFP20K74-ART; ISBN: 978-1-7281-4876-2

978-1-7281-4876-2/20/$31.00 ©2020 IEEE 1239

Authorized licensed use limited to: University of Exeter. Downloaded on July 02,2020 at 12:59:00 UTC from IEEE Explore. Restrictions apply.

Adware et al. [3] introduced a new lightweight self-learning approach to classifying the malicious web page based on the features categorized A MALURL framework was developed which used the Genetic Algorithm(GA) to train classifiers that are able to detect the malicious web pages. Dataset Alexa for Benign web site and Phis Tank for malicious web sites were considered. The average system precision was found out at 87%.

Hwang et al. [5] uses “Adaptive SVM(SVM) machine learning technique” The SVM can able to deal with new training data due to its adaptive capability. The objective of SVM is to reduce the probability of misclassification of new web pages.

Yue et al. [6] proposed a method for classifying malicious web pages using 30 features with the help of machine learning algorithm K-NN and SVM. The result of K-NN was better than SVM. Two classification models were used for detecting the malicious web pages and specific threat types. Yoo et al. [4] proposed two types of detection methods: misuse detection and anomaly detection for identifying known and unknown malicious web pages respectively. Though the detection rate was relatively high up to 98.9% it’s the false positive rate was high which is 30.5%. They have conducted their experiment in WEKA tool with dataset RafaBot.

Krishnaveni et al. [7] developed an interaction tool, Spider Net which was able to detect the malicious web page. The tool was implemented in MatLab. Two machine learning classifiers, multi-SVM, and ELM were implemented in the tool by taking three feature sets namely common features, redirect features, JavaScript features, and XSS attack features showing higher accuracy in ELM (96.62%) than multi-SVM(93.22%).

Sun et al. [8] build a framework, automatic blacklist generator(AutoBLG) which can be able to explore new and previously unknown and malicious URL. AutoBLG performed the task by URL Expansion, URL Filtration, and URL Verification methods. Wang et al. [10] proposed a hybrid approach to detecting malicious web pages. Static analysis extracted the static features of web pages and trained the classifier to predict whether the web page is malicious or not. Dynamic analysis executes the web pages in the browser engine and analyses the dynamic behaviour of the web page. The first one produced excessive false negative and the second one needed more computing resources. Kim et al. [11] developed a web page detector WebMon tool which was 7.6 times faster than the traditional tools. For malicious web page detection, a practical model was proposed which consists of WebKit-2, ML, and YARA based framework. Also, a call tree algorithm was presented to create a malicious redirection tree that was able to find the malicious path. Altay et al. [12] “developed a novel approach of context-sensitive and keyword density-based for classifying the web pages with the help of supervised machine learning algorithms (SVM, maximum entropy, ELM).”

Although there are several approaches have been proposed for detecting dangerous sites. The main disadvantage of these approaches is: to achieve their results they used tens and hundreds and thousands of samples, used no methodology to detect malicious URL redirection which is constantly changing, facing the difficulties in collecting all kinds of samples.

**LITERATURE SURVEY**

**[1] Altay, Betel, Tansel Dokeroglu, and Ahmet Cosar. "Context-sensitive and keyword density-based supervised machine learning techniques for malicious webpage detection." Soft Computing 23, no. 12 (2019): 4177-4191.**

Conventional malicious webpage detection methods use blacklists in order to decide whether a webpage is malicious or not. The blacklists are generally maintained by third-party organizations. However, keeping a list of all malicious Web sites and updating this list regularly is not an easy task for the frequently changing and rapidly growing number of webpages on the web. In this study, we propose a novel context-sensitive and keyword density-based method for the classification of webpages by using three supervised machine learning techniques, support vector machine, maximum entropy, and extreme learning machine. Features (words) of webpages are obtained from HTML contents and information is extracted by using feature extraction methods: existence of words, keyword frequencies, and keyword density techniques. The performance of proposed machine learning models is evaluated by using a benchmark data set which consists of one hundred thousand webpages. Experimental results show that the proposed method can detect malicious webpages with an accuracy of 98.24%, which is a significant improvement compared to state-of-the-art approaches.

**[2] Kim, Sungjin, Jinkook Kim, Seokwoo Nam, and Dohoon Kim. "WebMon: ML-and YARA-based malicious webpage detection." Computer Networks 137 (2018): 119-131. T**

Attackers use the openness of the Internet to facilitate the dissemination of [malware](https://www.sciencedirect.com/topics/engineering/malware). Their attempts to infect target systems via the Web have increased with time and are unlikely to abate. In response to this threat, we present an automated, low-interaction malicious webpage detector, WebMon, that identifies invasive roots in Web resources loaded from WebKit2-based browsers using machine learning and YARA signatures. WebMon effectively detects hidden exploit codes by tracing linked URLs to confirm whether the relevant websites are malicious. WebMon detects a variety of attacks by running 250 containers simultaneously. In this configuration, the proposed model yields a detection rate of 98%, and is 7.6 times faster (with a container) than previously proposed models. Most importantly, Web Môn’s focus on extracting malicious paths in a domain is a novel approach that has not been explored in previous studies.

**[3] Wang, Rong, Yan Zhu, Jiefan Tan, and Binbin Zhou. "Detection of malicious web pages based on hybrid analysis." Journal of Information Security and Applications 35 (2017): 68-74.74.**

Malicious web pages have become an increasingly serious threat to web security in recent years. In this paper, we propose a new detection method that consists of static and dynamic analyses for detecting malicious web pages. Static analysis utilizes [classification algorithms](https://www.sciencedirect.com/topics/computer-science/classification-algorithm) in machine learning to identify certain benign and malicious web pages. As a complement to static analysis, dynamic analysis mainly checks the unknown web pages to determine whether they have malicious shellcodes during their execution. Because of the combination of static and dynamic analyses, the proposed detection method achieves high performance, and it has a light weight and is simple to use.

**[4] Urcuqui, Christian, Andres Navarro, Jose Osorio, and Melisa Garcia. "Machine Learning Classifiers to Detect Malicious Websites." In SSN, pp. 14-17. 2017.).**

A risk that exists in Internet is the access of websites with malicious content, because they might be open doors for cybercrimes or be the mechanism to download files in order to affect organizations, persons and the environment. What is more, the attack registers through websites have been part of cyberattacks reports during the last years; this information includes attacks made by the currently risks found in new technologies, such as the IoT. Due the computer security complexity, studies have been working in to use machine learning algorithms to identify web malicious content. This article explores the application of a data analysis process through a framework that includes dynamic, static analysis, updated websites and a low interaction client honeypot in order to classify a website. Furthermore, it evaluates the capacity of the classification of four machine learning through the information analysed.

**[5] Yoo, Suyeon, Sehun Kim, Anil Choudhary, O. P. Roy, and T. Tuithung. "Two-phase malicious web page detection scheme using misuse and anomaly detection." International Journal of Reliable Information and Assurance 2, no. 1 (2014): 1-9**

Misuse detection method and anomaly detection method are widely used for the detection of malicious web pages. Both are based on machine learning. Misuse detection can detect known malicious web pages, but it cannot detect new ones. In contrast, anomaly detection can detect unknown malicious web pages, but it has a high false positive rate. In order to achieve a high detection rate through precisely detecting known and unknown malicious web pages, we propose a two-phase detection scheme. In the first phase, the misuse detection model is built based on the C4.5 decision tree algorithm, which allows known malicious web pages to be detected. In the second phase, the anomaly detection model with a one-class support vector machine is used to detect new types of malicious web pages. The experimental results show that our proposed method has significantly higher malicious web page detection rate than conventional ones with the expense of slightly higher false positive rate. Keywords: Anomaly detection, machine learning, malicious webpage, misuse detection

**SCOPE:**

In this application, we design a new classification system to analyse and detect the malicious web pages using machine learning classifiers. The classifiers are random forest, support vector machine, naïve Bayes and logistic regression. We use URL (Uniform Resource Locator) based on extricated features the classifiers are trained to predict the malicious web pages.

**EXISTING METHOD**

Existing methods typically detect malicious URLs of a single attack type. In this paper, we propose method using machine learning to detect malicious URLs of all the popular attack types and identify the nature of attack a malicious URL attempts.

**Disadvantages :**

1. Accuracy low

2. operating cost is high

3. difficult to handle

**PROPOSED METHOD**

Our method uses a variety of discriminative features including textual properties, link structures, webpage contents, DNS information, and network trafﬁc. Many of these features are novel and highly effective. Our experimental studies with 40,000 benign URLs and 32,000 malicious URLs obtained from real-life Internet sources show that our method delivers a superior performance: the accuracywasover98%indetectingmaliciousURLsandover 93%inidentifying attack types. Wealsoreport our studies on the effectiveness of each group of discriminative features, and discuss their evadability

**Advantages :**

1. accuracy is maximum

2. prediction is accurate

**HARDWARE & SOFTWARE REQUIREMENTS**

# H/W Configuration:

# Processor - I3/Intel Processor

* Hard Disk -160GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA
* RAM - 8Gb

**S/W Configuration:**

* Operating System : Windows 7/8/10
* Server side Script : Python, Anaconda
* IDE : PyCharm
* Libraries Used :sklearn, pandas, Scikit-learn
* Dataset : url dataset for benign and malicious web pages

Technology : Python 3.6

**ARCHITECTURE**

Import packages

View dataset

Download dataset

model performance

Train the dataset

Upload dataset

Decision Tree

XGBoost

KNN

Adaboost

Graphs

Predictions

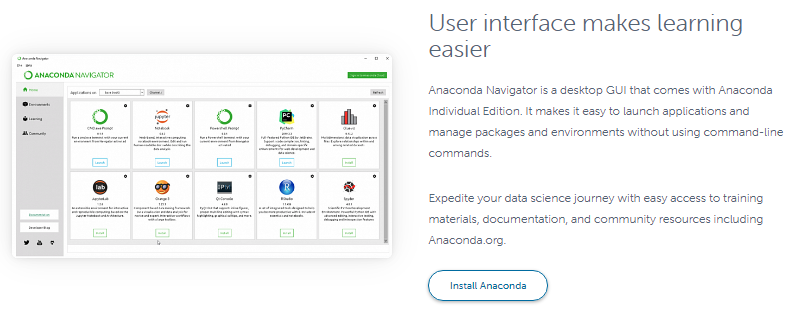
Benign

Malicious

#### **SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECT:**

#### 1. Visit the Anaconda downloads page.

Go to the following link: [Anaconda.com/downloads](https://www.anaconda.com/download/)



#### 2. Select Windows.

Select Windows where the three operating systems are listed.

#### 3. Download.

Choose Python 3.6 version, 64 bit graphical installer.

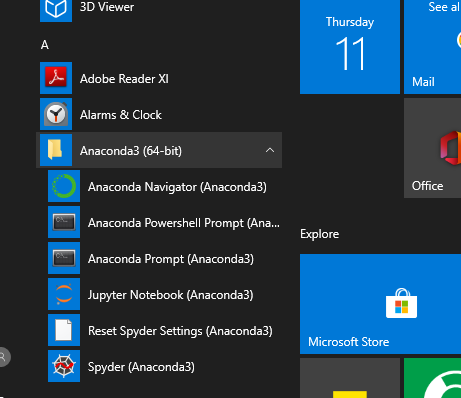
4. Let it download in an .exe format.

#### 5. Open and run the installer.

#### Once the download completes, open and run the .exe installer.

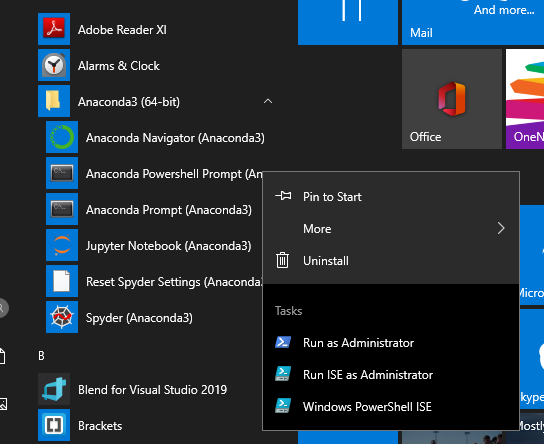
6. Click on next, I agree, install.

7. Completion of the installation, open your windows start menu and select the Anaconda navigator.



8. You need to install some packages to execute your project in a proper way.

9. Select windows start menu, right click on anaconda prompt, choose run as administrator.



11. Anaconda prompt will get open, with specified path, type “Pip install package name” which you want to install (like Jumpy, Pandas, seaborne, sickie learn, matplotlib, pilot)

Ex: pip install Numpy



12. You can also install required packages in Jupiter notebook directly by using the syntax as “! Pip install package name.

Ex: pip install Jumpy

**MODULES**

**System**

**User**

**1.System:**

**1.1 Train data set:**

System can give training to the data set

**1.2 model performance:**

The three main metrics used to evaluate a classification **model** are accuracy, precision, and recall. Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

**1.3 predictions:**

Using the machine leaning algorithms, we can predict the result

**2.User:**

**2.1 upload dataset**

The user uploads the dataset.

**2.2 view dataset**

The uploaded dataset is viewed by the user.

**2.2 viewing graphs**

Graphs can be generated by the system and the user can be view that graphs

**Algorithms:**

**XGBoost:**

XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.

Bagging: Now imagine instead of a single interviewer, now there is an interview panel where each interviewer has a vote. Bagging or bootstrap aggregating involves combining inputs from all interviewers for the final decision through a democratic voting process.

XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners (CARTs generally) using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

**Decision Trees:**

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal.

A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn’t split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.

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

So, what is actually going on in the background? Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, you will need to trim it down for it to look beautiful. Let’s start with a common technique used for splitting.

**K Nearest Neighbors:**

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.

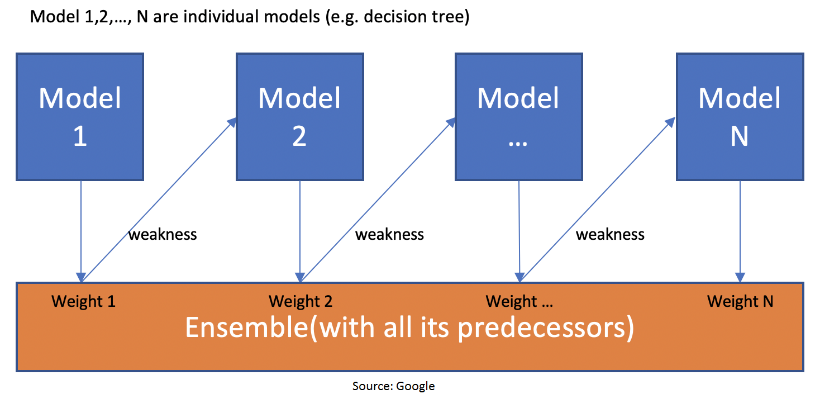
The K-NN working can be explained on the basis of the below algorithm:

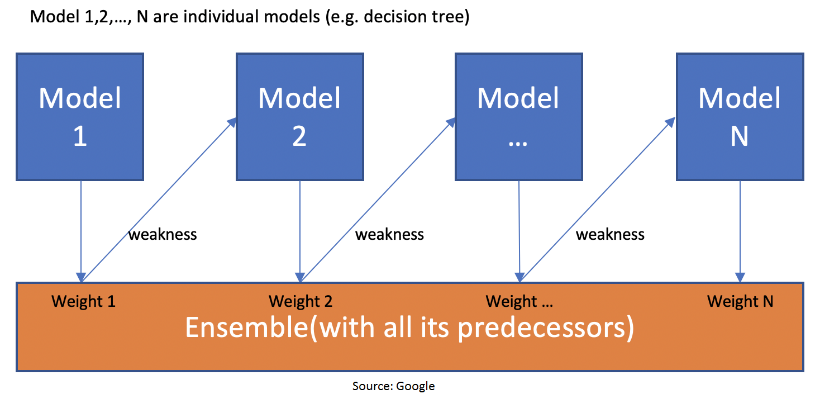
* Step-1: Select the number K of the neighbors
* Step-2: Calculate the Euclidean distance of K number of neighbors
* Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
* Step-4: Among these k neighbors, count the number of the data points in each category.
* Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.
* Step-6: Our model is ready.

**AdaBoost Algorithm:**

AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique that is used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights to incorrectly classified instances. Boosting is used to reduce bias as well as the variance for supervised learning. It works on the principle where learners are grown sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones. Adaboost algorithm also works on the same principle as boosting, but there is a slight difference in working.

It makes *n* number of decision trees during the training period of data. As the first decision tree/model is made, the record which is incorrectly classified during the first model is given more priority. Only these records are sent as input for the second model. The process will go on until we specify a number of base learners we want to create. Remember, the repetition of records is allowed with all boosting techniques.





This figure shows that when the first model is made and the errors from the first model are noted by the algorithm, the record which is incorrectly classified is given as the input for the next model. This process is repeated until the specified condition is met. As you can see in the figure, there are *n* number of models made by taking the errors from the previous model. This is how boosting works. The models 1,2, 3,…, N are individual models that can be known as decision trees. All types of boosting models work on the same principle.

**STEPS FOR EXECUTING THE PROJECTS**

1. Install the required packages

2. Load the datasets.

3. Pre-process the data.

4. Split the dataset into train and test.

5. Use the train dataset to train the ml models.

6. Use the test data to test the model for prediction and accuracy generation.

**SYSTEM DESIGN**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful inssss the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**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 meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**USE CASE DIAGRAM:**

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.



**CLASS DIAGRAM:**

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.



**SEQUENCE DIAGRAM:**

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.

****

**Collaboration Diagram:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.

****

**DEPLOYMENT DIAGRAM**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware’s used to deploy the application.



**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.



**Component diagram**,

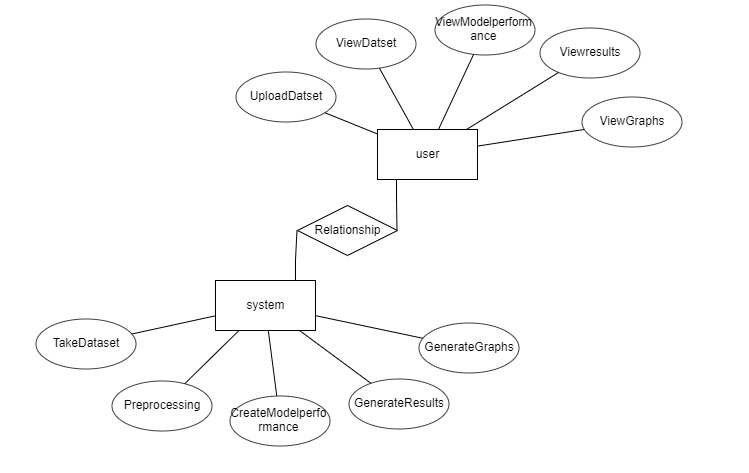
A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.



**ER Diagram:**

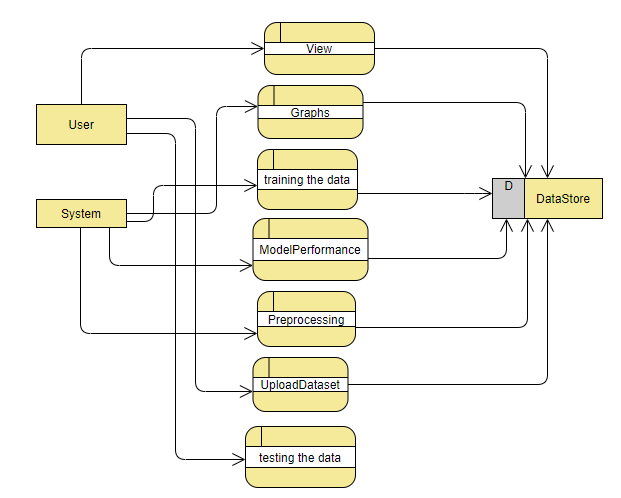
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

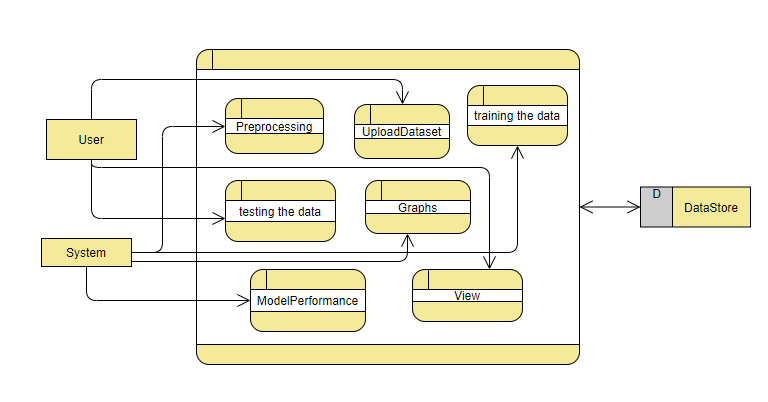
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.



**DFD Diagram:**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.





# **INTRODUCTION TO PYTHON**

* **Python**

### What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python.  This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

**Scripts are reusable**

Basically, a script is a text file containing the statements that comprise a Python program.  Once you have created the script, you can execute it over and over without having to retype it each time.

**Scripts are editable**

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor.  Then you can execute each of the individual versions.  In this way, it is easy to create different programs with a minimum amount of typing.

**You will need a text editor**

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad, Microsoft WordPad, Microsoft Word,*or just about any word processor if you want to.

**Difference between a script and a program**

**Script:**

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

**Program:**

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

**Python**

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

**Python concepts**

If you’re not interested in the the hows and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it’s a great one to start programming with.

• Open source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-is to interface with C++ (via SWIG)

• Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **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 is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Possum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and UNIX shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Possum still holds a vital role in directing its progress.

**Python Features**

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Dynamic vs. Static**

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

**Variables**

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

**Standard Data Types**

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

## Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

## Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

## Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

## Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as **read-only** lists.

## Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

**Different modes in python**

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .pie files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole 20 Python libraries

**1.** Requests. The most famous http library written by Kenneth remits. It’s a must have for every python developer.

**2.** Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won’t use any other.

**3.** Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.

**4.** Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

**5.** SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.

**6.** Beautiful Soup. I know it’s slow but this xml and html parsing library is very useful for beginners.

**7.** Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.

**8.** Numbly. How can we leave this very important library? It provides some advance math functionalities to python.

**9.** Skippy. When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

**10.** Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

**11.** Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.

**12.** Piglet. A 3d animation and game creation engine. This is the engine in which the famous [python port](https://github.com/fogleman/Minecraft) of mine craft was made

**13.** Pit. A GUI toolkit for python. It is my second choice after python for developing GUI’s for my python scripts.

**14.** Pit. Another python GUI library. It is the same library in which the famous Bit torrent client is created.

**15.** Scaly. A packet sniffer and analyzer for python made in python.

**16.** Pywin32. A python library which provides some useful methods and classes for interacting with windows.

**17.** Notch. Natural Language Toolkit – I realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

**18.** Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

**19.** Simply. Simply can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

**20.** I Python. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

**Jumpy**

Humpy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In numbly dimensions are called axes. The number of axes is rank.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

**Matplotlib**

• High quality plotting library.

**Python class and objects**

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

**Inheritance**

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by sub classing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components.

As we’ve seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python’s object model and the object interfaces are more consistent to Python’s built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python’s built-in operation methods

**Exceptions**

I’ve talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program’s flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn’t exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don’t occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a Key Error exception.

• Searching a list for a non−existent value will raise a Value Error exception

. • Calling a non−existent method will raise an Attribute Error exception.

• Referencing a non−existent variable will raise a Name Error exception.

• Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don’t need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don’t put any overhead on the code processing.

Because exceptions aren’t supposed to happen very often, they aren’t processed until they occur.

Exceptions can be thought of as a special form of the if/elf statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren’t processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won’t spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won’t have to do this very often but it’s nice to have the option when necessary.

However, before making your own exceptions, make sure there isn’t one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

**Python modules**

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

**Testing code**

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

His entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

## To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.

## There are three types of functions in python:

## Help (), min (), print ().

## Python Namespace

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a name spacing system from daily life, i.e. the naming of people in first name and family name (surname).

An example is a network: each network device (workstation, server, printer,) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames.   
Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

* **global names** of a module
* **local names** in a function or method invocation
* **built-in names**: this namespace contains built-in functions (e.g. abs(), camp(), ...) and built-in exception names

**Garbage Collection**

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

**Python XML Parser**

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX: Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

**Python Web Frameworks**

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

## Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format tinplating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

## Comparing web frameworks

There is also a repository called [compare-python-web-frameworks](https://github.com/mattmakai/compare-python-web-frameworks) where the same web application is being coded with varying Python web frameworks, tinplating engines and object.

## Web framework resources

* When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
* Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
* What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
* Jingo vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
* This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
* Python’s web frameworks benchmarks  is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
* What web frameworks do you use and why are they awesome? Is a language agnostic Reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
* This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?” The votes aren't as important as the list of the many frameworks that are available to Python developers.

## Web frameworks learning checklist

1. Choose a major Python web framework (Jingo or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the [deployment](https://www.fullstackpython.com/deployment.html) section to make it accessible on the web.

**SYSTEM STUDY**

**2.1 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

### **6. SYSTEM TESTING**

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, sub-assemblies, 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.

**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 expected results.

**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 of components.

**Functional test**

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

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

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 box level.

**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, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**6.1 Unit Testing:**

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 work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 6.2 Integration Testing

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.

**6.3 Acceptance Testing**

User 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 functional requirements.

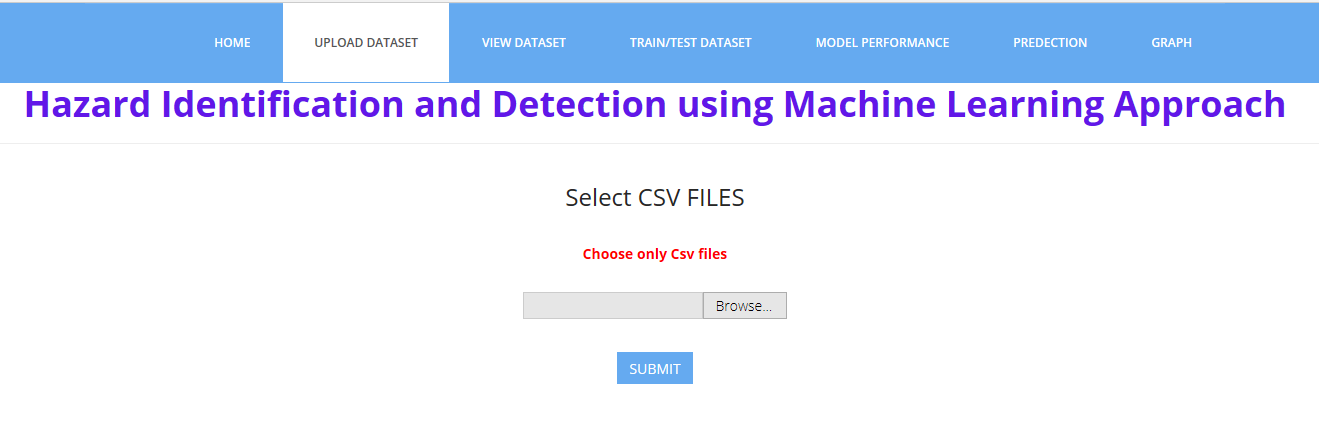
**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

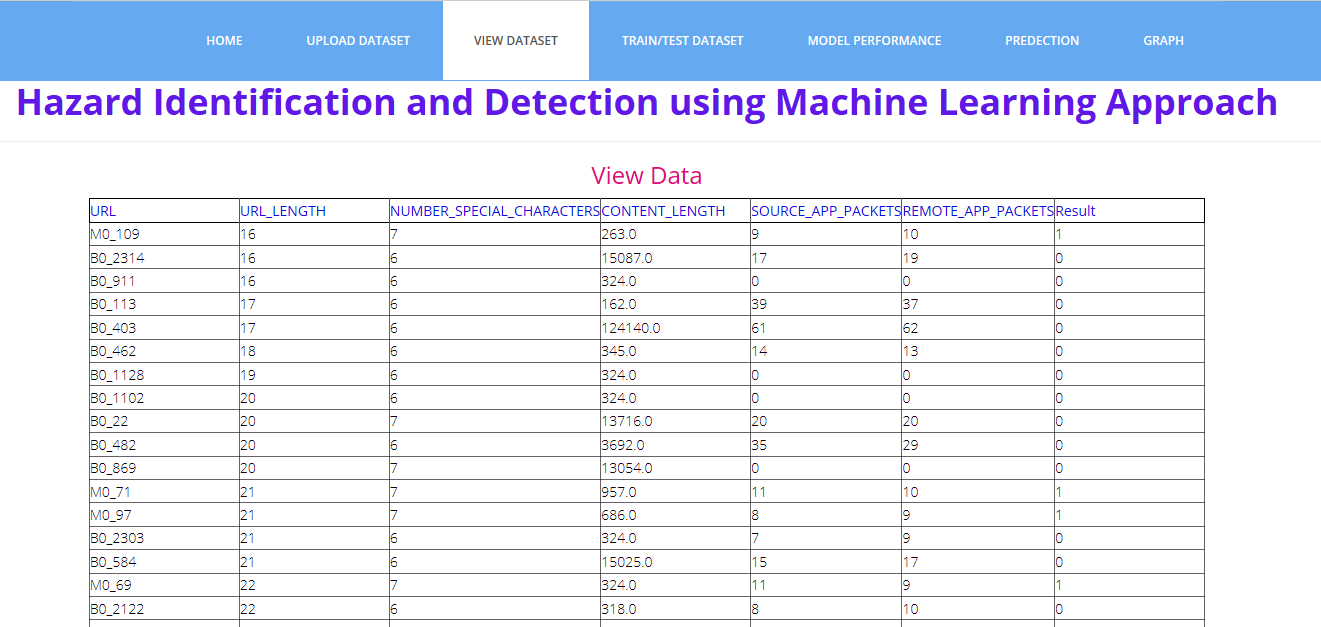
**RESULTS:**

Home page

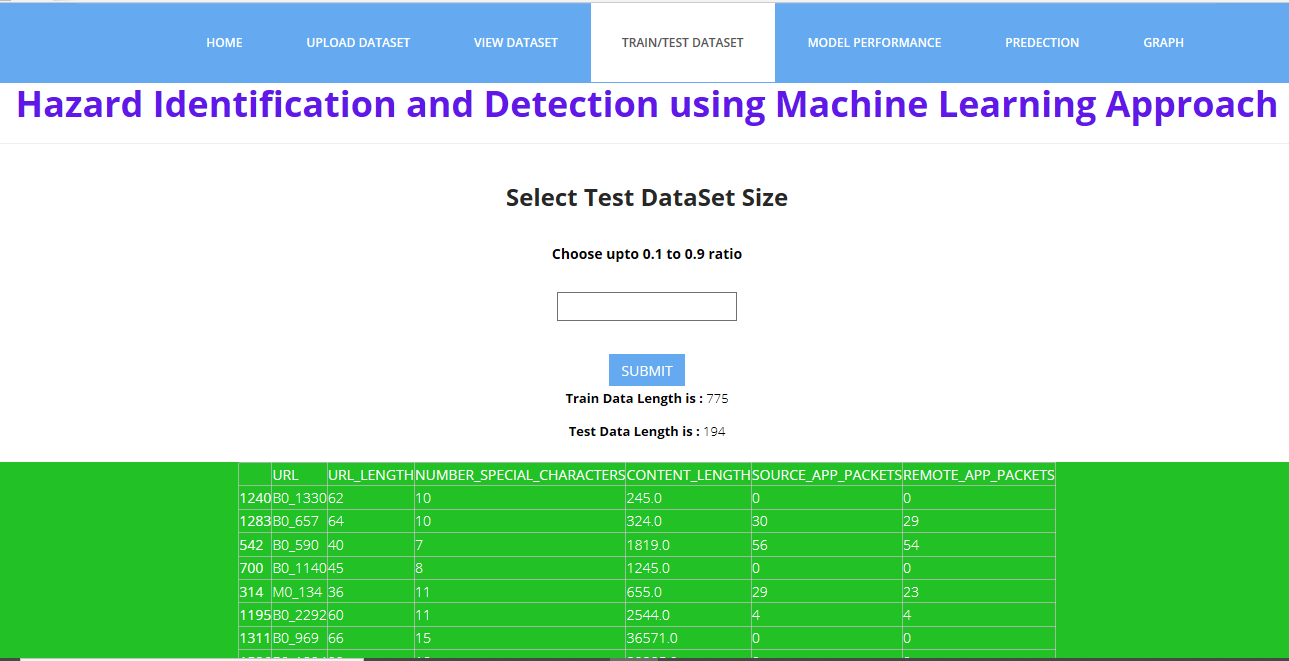


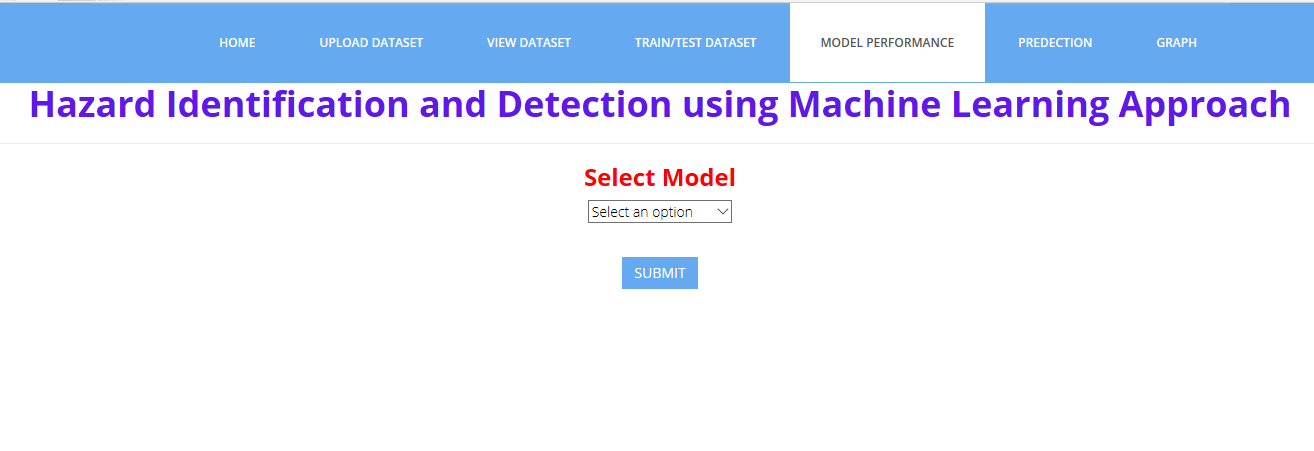
Upload dataset

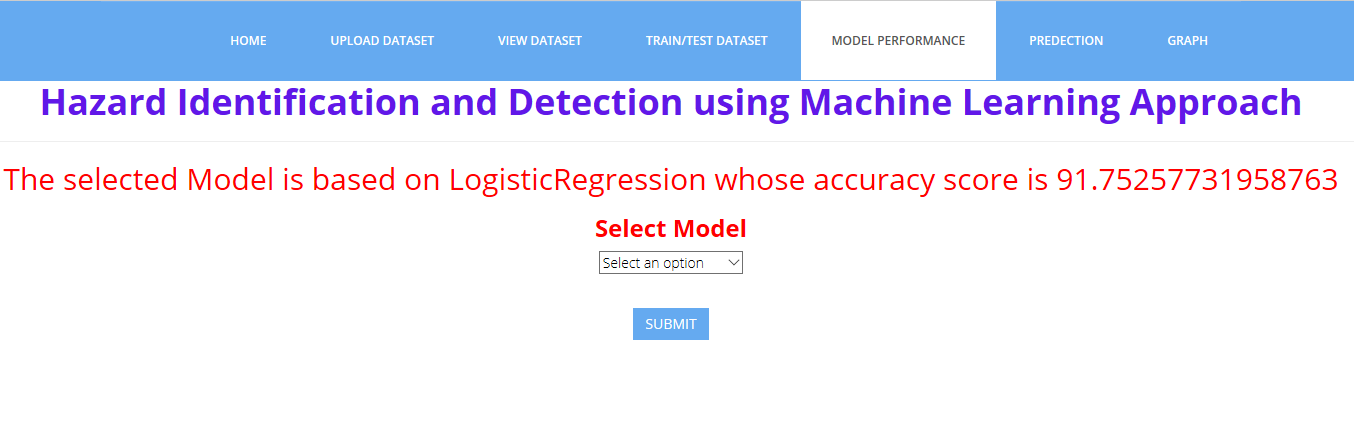


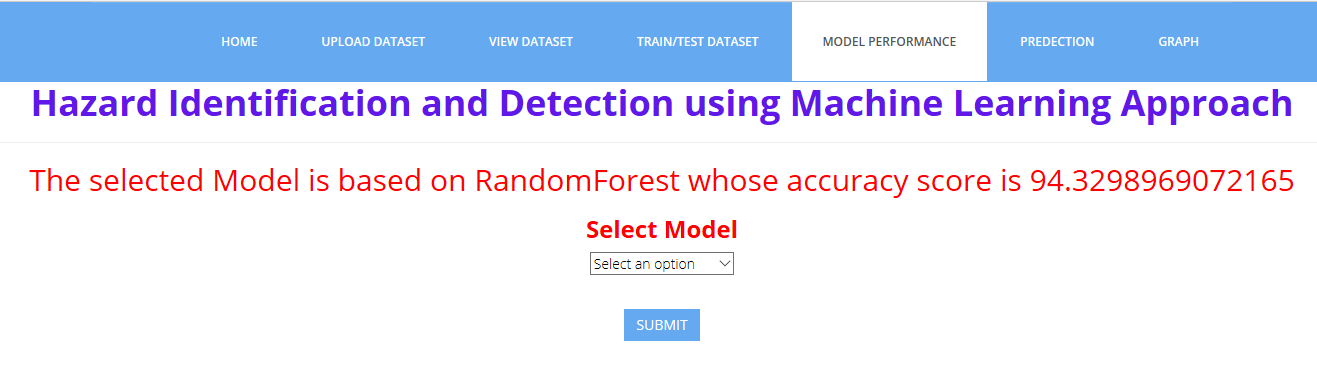


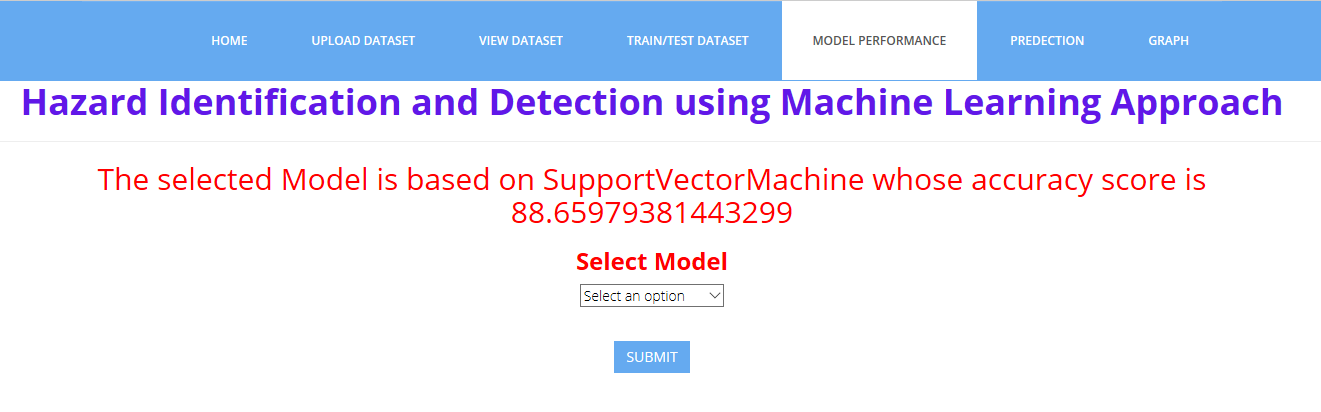
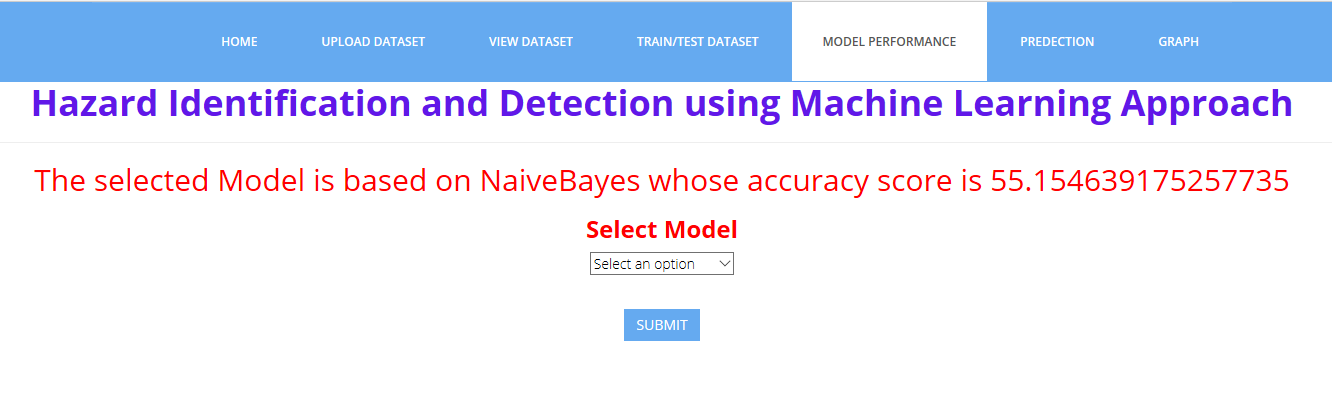
**Train/test dataset size:**



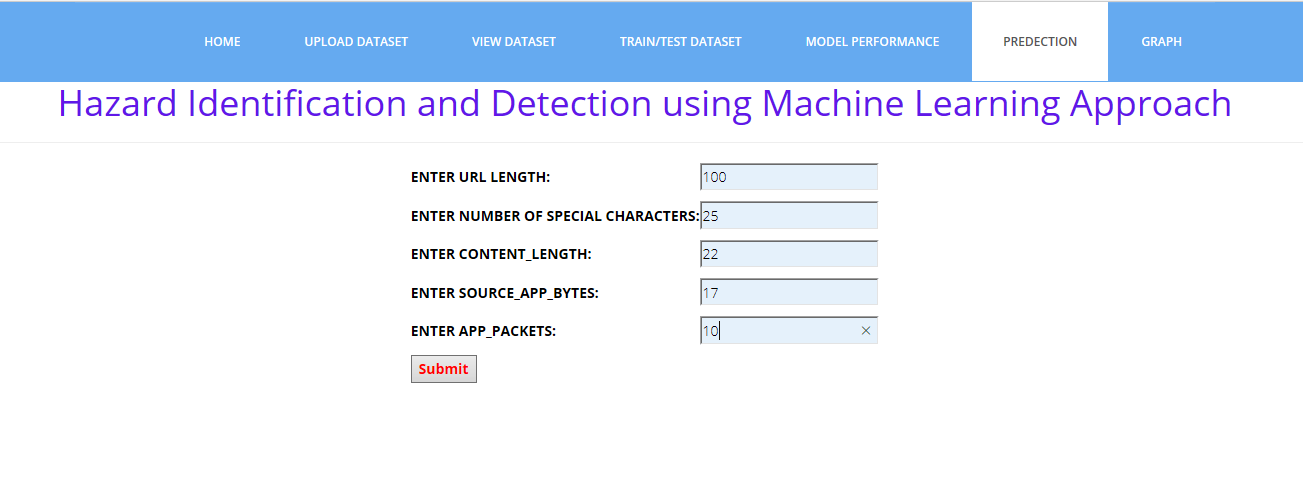
Model performance:

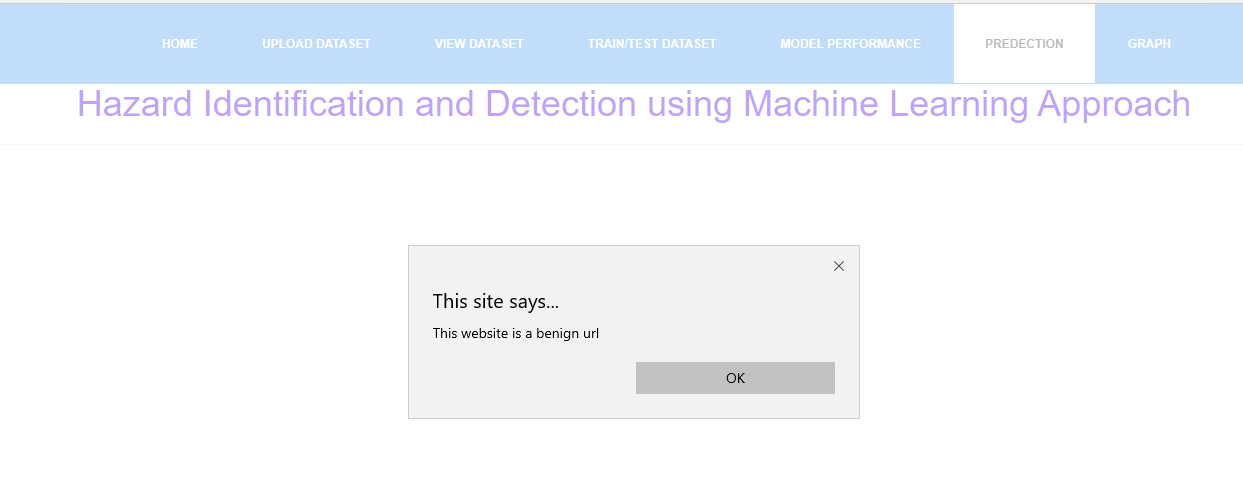


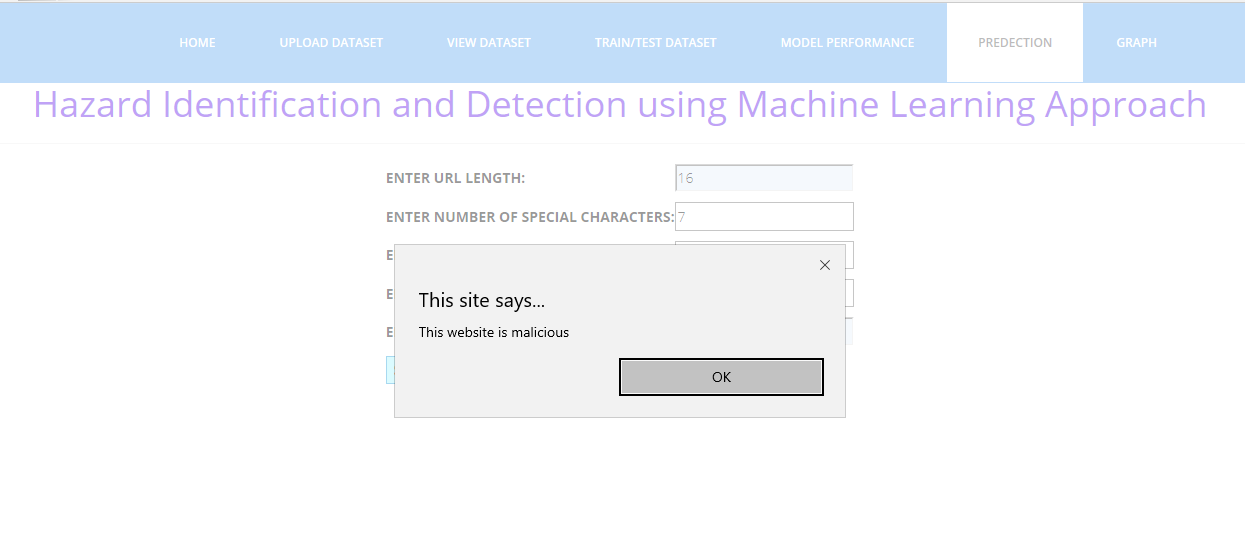




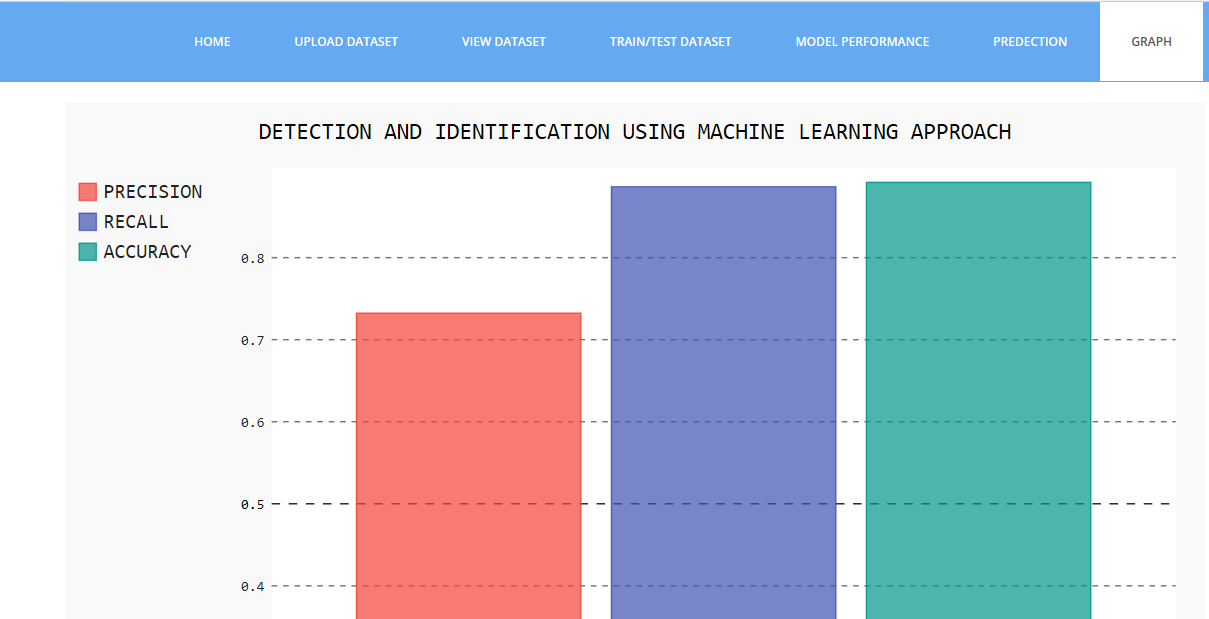
**Prediction:**







Graph:



**CONCLUSION:**

Malicious web page identification is an emerging topic in cybersecurity. Though several research studies have been performed relating to the issues of malicious web page detection these are very costly as they consume more time and resources. In this research article, we employed a new web site classification system based on URL features to predict the web pages as malicious or benign using machine learning algorithms. The machine learning classifiers Random Forest(RF) achieves a higher accuracy of 95%. The experimental results have shown that our method can perform effectively for detecting the malicious web page

**FUTURE SCOPE:**

In future work, it has been planned to expand the feature sets and analysis using various sources of data to enhance the classifier performance

**REFERENCES**

[1] Tao, Wang, Yu Shunzheng, and Xie Bailin. "A novel framework for learning to detect malicious web pages." In 2010 International Forum on Information Technology and Applications, vol. 2, pp. 353-357. Ieee, 2010.

[2] Eshete, Birhanu, Adolfo Villafiorita, and Komminist Weldemariam. "Malicious website detection: Effectiveness and efficiency issues." In 2011 First SysSec Workshop, pp. 123-126. IEEE, 2011..

[3] Aldwairi, Monther, and Rami Alsalman. "Malurls: A lightweight malicious website classification based on url features." Journal of Emerging Technologies in Web Intelligence 4, no. 2 (2012): 128-133..

[4] Yoo, Suyeon, Sehun Kim, Anil Choudhary, O. P. Roy, and T. Tuithung. "Two-phase malicious web page detection scheme using misuse and anomaly detection." International Journal of Reliable Information and Assurance 2, no. 1 (2014): 1-9.

[5] Hwang, Young Sup, Jin Baek Kwon, Jae Chan Moon, and Seong Je Cho. "Classifying malicious web pages by using an adaptive support vector machine." Journal of Information Processing Systems 9, no. 3 (2013): 395-404.

[6] Yue, Tao, Jianhua Sun, and Hao Chen. "Fine-grained mining and classification of malicious Web pages." In 2013 Fourth International Conference on Digital Manufacturing & Automation, pp. 616-619. IEEE, 2013..

[7] Krishnaveni, S., and K. Sathiyakumari. "SpiderNet: An interaction tool for predicting malicious web pages." In International Conference on Information Communication and Embedded Systems (ICICES2014), pp. 1-6. IEEE, 2014.

[8] Sun, Bo, Mitsuaki Akiyama, Takeshi Yagi, Mitsuhiro Hatada, and Tatsuya Mori. "Automating URL blacklist generation with similarity search approach." IEICE TRANSACTIONS on Information and Systems 99, no. 4 (2016): 873-882..

Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2020) IEEE Xplore Part Number:CFP20K74-ART; ISBN: 978-1-7281-4876-2

978-1-7281-4876-2/20/$31.00 ©2020 IEEE 1243

Authorized licensed use limited to: University of Exeter. Downloaded on July 02,2020 at 12:59:00 UTC from IEEE Xplore. Restrictions apply.

[9] Urcuqui, Christian, Andres Navarro, Jose Osorio, and Melisa García. "Machine Learning Classifiers to Detect Malicious Websites." In SSN, pp. 14-17. 2017.).

[10] Wang, Rong, Yan Zhu, Jiefan Tan, and Binbin Zhou. "Detection of malicious web pages based on hybrid analysis." Journal of Information Security and Applications 35 (2017): 68-74.74.

[11] Kim, Sungjin, Jinkook Kim, Seokwoo Nam, and Dohoon Kim. "WebMon: ML-and YARA-based malicious webpage detection." Computer Networks 137 (2018): 119-131.

[12] Altay, Betul, Tansel Dokeroglu, and Ahmet Cosar. "Context-sensitive and keyword density-based supervised machine learning techniques for malicious webpage detection." Soft Computing 23, no. 12 (2019): 4177-4191.

[13] website: http://jupyter.org/

[14] <https://archieve.ics.uci.edu/ml/dataset/>

[15] Ibrahim, M. Y. (2017). Real Time Xss Detection: A Machine Learning Approach.

[16] https://medium.com/thalus-ai/performance-metrics-forclassification-problems-in-machine-learning-part-ib085d432082b