**An Intrusion Detection Feature Selection Method Based on Improved Mutual Information**

**ABSTRACT**

**ABSTRACT**

Feature selection methods for classification are crucial for intrusion detection techniques using machine learning. The high-dimensional features in intrusion detection data affect computational complexity, consume more used resources and more time for data analysis, and the irrelevant and redundant features among them often hinder the performance of classifiers and mislead the classification task. Therefore, it is challenging to select more relevant features from intrusion detection data containing many such features. Therefore, we proposes an efficient feature selection algorithm that considers the correlation between features and the redundancy of pairs of features with respect to class labels based on an improved Pearson correlation coefficient. The evaluation function is later improved based on conditional mutual information to obtain a final subset of features with the goal of improving the classification rate and accuracy. The proposed feature selection method is compared with three existing feature selection methods on the frequently studied public benchmark intrusion detection dataset NSL-KDD, and the experimental results show that the features selected by the proposed method lead to a significant reduction in execution time while resulting in higher classification accuracy.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.NO | CONTENT | PGNO |
| 1 | Introduction   * 1. Objective   2. Problem Statement   3. Software requirements   4. Hardware requirements |  |
| 2 | Feasibility study  2.1 Economic feasibility  2.2 Technical feasibility  2.3 Social feasibility |  |
| 3 | Literature survey |  |
| 4 | System analysis  4.1 Existing system  4.1.1 Disadvantages of existing system  4.2 Proposed system  4.2.1 Advantages of proposed system  4.3 Functional requirements  4.4 Non-Functional requirements |  |
| 5 | System design  5.1 System architecture  5.2 UML diagrams |  |
| 6 | Implementation  6.1 Modules  6.2 Sample code |  |
| 7 | Software environment |  |
| 8 | System testing  8.1 Testing strategies  8.2 Test cases |  |
| 9 | Screens |  |
| 10 | Conclusion |  |
| 11 | References |  |

**INTRODUCTION**

**1. INTRODUCTION**

In recent years, with the rapid development of network technology and the increasing richness of network applications, network traffic data has correspondingly increased in large amounts, tending to be complex and diverse. Therefore, reliable intrusion detection technologies are needed to prevent host and network attacks. Network Intrusion Detection System (IDS) is a common tool for detecting network intrusions by collecting data on the current operational state of the network and analyzing network traffic using system prebuilt algorithms and historical experience. For best results and accurate identification, machine learning (ML) is a popular and effective technique that is also widely used in the field of intrusion detection. However, there is a large amount of redundant and irrelevant data in network traffic data, which interferes with the classification process of IDS. Therefore, dimensionality reduction of the data is a very necessary step in the process of data mining, analysis and application. Due to the large amount of data processed by IDS, the shortcomings of ML-based IDS include high computational complexity, long computation time and excessive resource consumption, which are mainly caused by the redundant, incomplete and irrelevant features contained in the ML dataset. Feature selection techniques are now a very important step in intrusion detection research in order to be able to dig deeper into the critical information contained in network traffic data and to eliminate interfering redundant and irrelevant features.

* 1. **OBJECTIVE**

The main objective of this the application machine algorithms to Intrusion Detection by using different types of Machine learning and Deep learning Models.

* 1. **PROBLEM STATEMENT**

The proposed Project for Intrusion Detection including the dataset, pre-processing, feature extraction and feature selection, algorithms, framework, and evaluation metrics, is presented and discusses the evaluation results of the experiments performed, and finally concludes the project with framework predict of credit card fraud.

* 1. **SOFTWARE REQUIREMENTS**

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

**Platform –** In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer’s architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

**APIs and drivers –** Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

**Web browser –** Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

1. **Software : Anaconda**
2. **Primary Language : Python**
3. **Frontend Framework : Flask**
4. **Back-end Framework : Jupyter Notebook**
5. **Database : Sqlite3**
6. **Front-End Technologies : HTML,CSS,JavaScript and Bootstrap4**

**1.2 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 is often 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 aspects of hardware requirements.

**Architecture –** All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

**Processing power –** The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

**Memory –** All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

**Secondary storage –** Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

**Display adapter –** Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

**Peripherals –** Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

**1) Operating System : Windows Only**

**2) Processor : i5 and above**

**3) Ram : 8gb and above**

**4) Hard Disk : 25 GB in local drive**

**FEASIBILITY STUDY**

**2. 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

**2.1 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.

### **2.2 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.

**2.3 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.

**LITERATURE SURVEY**

**3. LITERATURE SURVEY**

**3.1 Machine learning approaches to network intrusion detection for contemporary internet traffic:**

[[Machine learning approaches to network intrusion detection for contemporary internet traffic | SpringerLink](https://link.springer.com/article/10.1007/s00607-021-01050-5)k](https://thenewstack.io/anomaly-detection-glimpse-into-the-future-of-iot-data/)

**ABSTRACT:** All organizations, be they businesses, governments, infrastructure or utility providers, depend on the availability and functioning of their computers, computer networks and data centers for all or part of their operations. Network intrusion detection systems are the first line of defense that protect computing infrastructure from external attacks. In this study we develop five different Machine Learning classifiers for a number of attacks. We used the CSE-CIC-IDS2018 dataset, developed in a collaborative effort between the Communications Security Establishment and the Canadian Institute for Cybersecurity. It is an extensive network traffic trace dataset that captures multiple attacks and has become available relatively recently. The previous major dataset used for the development of network intrusion detection systems is the KDD Cup’99 dataset, now going on 22 years, which predates mobile computing, Web 2.0/3.0, social media, streaming video and widespread use of SSL. These significant Internet trends of the last two decades demand a reevaluation and redevelopment of intrusion detectors. Prior studies that designed Machine Learning classifiers using the CSE-CIC-IDS2018 dataset use a large and rich set of features, of which at least one is not dataset-invariant. Almost none have explored the appropriateness of using all available features with datasets containing only a few hundred attack class samples. The classifiers developed in this study rely on a justifiable number of features and their performance is reviewed for stability and generalization by reporting not just average performance over 10 fold cross-validation but also the degree of variation from one fold to the next.

**3.2 Attack classification using feature selection techniques: a comparative study:**

[Attack classification using feature selection techniques: a comparative study | SpringerLink](https://link.springer.com/article/10.1007/s12652-020-02167-9)

**ABSTRACT:** The goal of securing a network is to protect the information flowing through the network and to ensure the security of intellectual as well as sensitive data for the underlying application. To accomplish this goal, security mechanism such as Intrusion Detection System (IDS) is used, that analyzes the network traffic and extract useful information for inspection. It identifies various patterns and signatures from the data and use them as features for attack detection and classification. Various Machine Learning (ML) techniques are used to design IDS for attack detection and classification. All the features captured from the network packets do not contribute in detecting or classifying attack. Therefore, the objective of our research work is to study the effect of various feature selection techniques on the performance of IDS. Feature selection techniques select relevant features and group them into subsets. This paper implements Chi-Square, Information Gain (IG), and Recursive Feature Elimination (RFE) feature selection techniques with ML classifiers namely Support Vector Machine, Naïve Bayes, Decision Tree Classifier, Random Forest Classifier, k-nearest neighbours, Logistic Regression, and Artificial Neural Networks. The methods are experimented on NSL-KDD dataset and comparative analysis of results is presented.

**3.3 Network Intrusion Detection Using Wrapper-based Decision Tree for Feature Selection:**

[Network Intrusion Detection Using Wrapper-based Decision Tree for Feature Selection | Proceedings of the 2020 International Conference on Internet Computing for Science and Engineering (acm.org)](https://dl.acm.org/doi/10.1145/3424311.3424330)

**ABSTRACT:** ne of the key challenges of the machine learning (ML) based intrusion detection system (IDS) is the expensive computation time which is largely caused by the redundant, incomplete, and unrelated features contain in the IDS datasets. To overcome such challenges and ensure building efficient and more accurate IDS models, many researchers utilize preprocessing techniques such as normalization and feature selection, and a hybrid modeling approach is typically used. In this work, we propose a hybrid IDS modeling approach with an algorithm for feature selection (FS) and another for building the IDS. The FS method is a wrapper-based FS with a decision tree as the feature evaluator. Five selected ML algorithms are individually used in combination with the proposed FS method to build five IDS models using the UNSW-NB15 dataset. As a baseline, five more IDS models are built, in a single modeling approach, using the full features of the datasets. We evaluate the effectiveness of our proposed method by comparing it with the baseline models and also with state-of-the-art works. Our method achieves the best DR of 97.95% and proved to be quite effective in comparison to state-of-the-art works. We, therefore, recommend its usage especially in IDS modeling with the UNSW-NB15 dataset.

**3.4 Using Ensemble Learning Algorithms and Feature Selection Method for Improved Intrusion Detection System:**

[Using Ensemble Learning Algorithms and Feature Selection Method for Improved Intrusion Detection System | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/9633821)

**ABSTRACT:** Intrusion Detection and Deduce Systems monitor network traffic for irregularities dependent on marks and heuristics that vary from one seller to another and from one execution to another. Host Intrusion Recognition System and Host Intrusion Prevention System relevant at endpoints where NIDDS applies to organize limits what's more, division focuses like the passages to the web or other untrusted networks. By surveying the traffic beyond a shadow of a doubt inconsistencies, a NIDDS can determine malevolent or other undesired or unexpected information. At the point when a match is discovered dependent on designs, marks, or different heuristics, the framework can log it, send a caution to the observing framework or to the worker, or even take activity like obstructing, diverting, or resetting the association relying upon the association. NIDDS is a malevolent interruption avoidance framework that utilizations freely delivered marks containing noxious or other questionable path, just as conventional path assembled from various enemy of infection records and catalogs with novel client identifiers, in which the course can be anything from a web index During this article, we have proposed a methodology based on disconnecting the dataset from the information in different subsets for each round. At that point, we developed a segment assertion strategy using the procurement channel for each subset. The game plan of ideal highlights is made by putting together the summary of the courses of action acquired for each round. The results of direct tests in the NSL-KDD educational file show that the proposed methodology to incorporate decision with less reflections improves plot accuracy and reduces multifaceted nature. Additionally, a similar report on the reasonableness of the frame is drawn for choosing highlights using a variety of mounting techniques. To reinvigorate the overall spectacle, another movement appears using Random Forest and PART to initiate a topographic structure.

**3.5 Feature Selection by Maximizing Independent Classification Information:**

[Feature Selection by Maximizing Independent Classification Information | IEEE Journals & Magazine | IEEE Xplore](https://ieeexplore.ieee.org/document/7812571)

**ABSTRACT:** Feature selection approaches based on mutual information can be roughly categorized into two groups. The first group minimizes the redundancy of features between each other. The second group maximizes the new classification information of features providing for the selected subset. A critical issue is that large new information does not signify little redundancy, and vice versa. Features with large new information but with high redundancy may be selected by the second group, and features with low redundancy but with little relevance with classes may be highly scored by the first group. Existing approaches fail to balance the importance of both terms. As such, a new information term denoted as Independent Classification Information is proposed in this paper. It assembles the newly provided information and the preserved information negatively correlated with the redundant information. Redundancy and new information are properly unified and equally treated in the new term. This strategy helps find the predictive features providing large new information and little redundancy. Moreover, independent classification information is proved as a loose upper bound of the total classification information of feature subset. Its maximization is conducive to achieve a high global discriminative performance. Comprehensive experiments demonstrate the effectiveness of the new approach.

**SYSTEM ANALYSIS**

**4. SYSTEM ANALYSIS**

**4.1 EXISTING SYSTEM:**

In literature they used machine learning (ML) to develop a Network Intrusion Detection System (NIDS) capable of detecting modern attack types with a high detection rate. They implemented and evaluated several ML algorithms and compared their effectiveness using a state-of-the-art dataset containing modern attack types. Their study shows that accurate prediction and a high detection rate of attacks can be achieved using ML, indicating that ML has the potential to create very effective NIDS systems. In another researchthey compare the effectiveness of two intrusion detection system (IDS) methods using data mining. The two methods compared are Support Vector Machines (SVM) and Deep Neural Networks (DNN). The accuracy, precision, and recall were calculated and compared using NSL-KDD training and validation data, which is widely used in intrusion detection to compare the detection effects of the two techniques.

**4.1.1 DISADVANTAGES OF EXISTING SYSTEM:**

1. The existing work seems to lack a detailed focus on feature selection. Without an optimized feature selection process, the performance of the intrusion detection system may be affected. Feature selection is crucial for reducing noise and irrelevant data, which can lead to more accurate results.
2. The existing work might not explicitly address the issue of feature redundancy, which can lead to inefficiencies and suboptimal results. Feature redundancy can negatively impact the model's ability to generalize well to new data.
3. Without a clear emphasis on feature importance or relevance, the existing work might not effectively identify which features contribute most to accurate detection. This can lead to overemphasis on less important features and potentially compromise the overall detection performance.
4. While SVM and DNN are both powerful techniques, the existing work might not explore the trade-offs between model complexity and performance. DNNs, in particular, can be computationally intensive and require careful tuning to avoid overfitt.

# 4.2 Proposed System:

We proposed an efficient feature selection algorithm CCMI (Feature Selection Based on Correlation and Conditional Mutual Information) for intrusion detection techniques using machine learning. The algorithm considers the correlation between features and the redundancy of pairs of features with respect to class labels based on an improved Pearson correlation coefficient. The evaluation function is later improved based on conditional mutual information to obtain a final subset of features with the goal of improving the classification rate and accuracy. The proposed feature selection method is compared with three existing feature selection methods on the frequently studied public benchmark intrusion detection dataset NSL-KDD, andThe intrusion detection dataset NSL-KDD is experimentally evaluated on four classifiers RF, XGBoost, KNN, and MLP, using the proposed feature selection method.The experiments are performed to show that the features selected by the proposed method lead to a significant reduction in execution time while resulting in higher classification accuracy.

# 4.2.1 Advantages of proposed system:

1. We proposes an efficient feature selection algorithm (CCMI) that explicitly considers feature correlations and redundancy. This targeted approach can lead to a more optimal subset of features, enhancing the model's performance..
2. We evaluates the proposed feature selection method using multiple classifiers (RF, XGBoost, KNN, MLP). This diversity provides a broader understanding of how the method interacts with different algorithms, offering insights into its versatility.
3. The CCMI algorithm improves Pearson correlation coefficient by addressing feature redundancy. This ensures that the selected features are not only relevant but also independent, contributing to a more robust intrusion detection system.
4. We emphasis on accuracy, classification rate, and execution time provides a more holistic view of the method's performance. This consideration of both accuracy and efficiency is crucial for practical deployment.

### **4.3 FUNCTIONAL REQUIREMENTS**

1. Data Collection

2. Data Pre-processing

3. Training and Testing

4. Modiling

5. Predicting

### **4.4 NON FUNCTIONAL REQUIREMENTS**

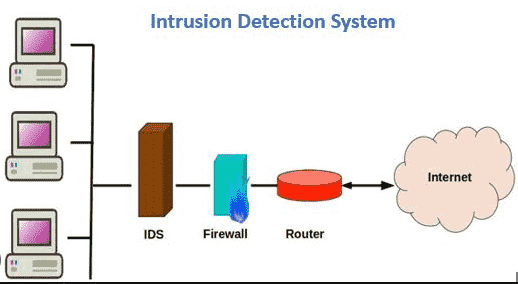
NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allow you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is > 10000. Description of non-functional requirements is just as critical as a functional requirement.

* Usability requirement
* Serviceability requirement
* Manageability requirement
* Recoverability requirement
* Security requirement
* Data Integrity requirement
* Capacity requirement
* Availability requirement
* Scalability requirement
* Interoperability requirement
* Reliability requirement
* Maintainability requirement
* Regulatory requirement
* Environmental requirement

**SYSTEM DESIGN**

**5. SYSTEM DESIGN**

**5.1 SYSTEM ARCHITECTURE:**

****

**Fig.5.1.1 System architecture**

**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

**Import libraries**

**VERIFY**

**NO PROCESS**

**Yes NO**

**Importing the dataset**

**Data processing**

**Feature Selection**

**Splitting the data into train & test**

**Building the model CCMI Features {Random Forest, XGBoost, KNN, MLP, Stacking Classifier (RF + ET with LightGBM), Voting Classifier (RF + AB)**

**Training the model**

**Signup & sign in**

**User input**

**Final outcome**

**End process**

**5.2 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 in 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.

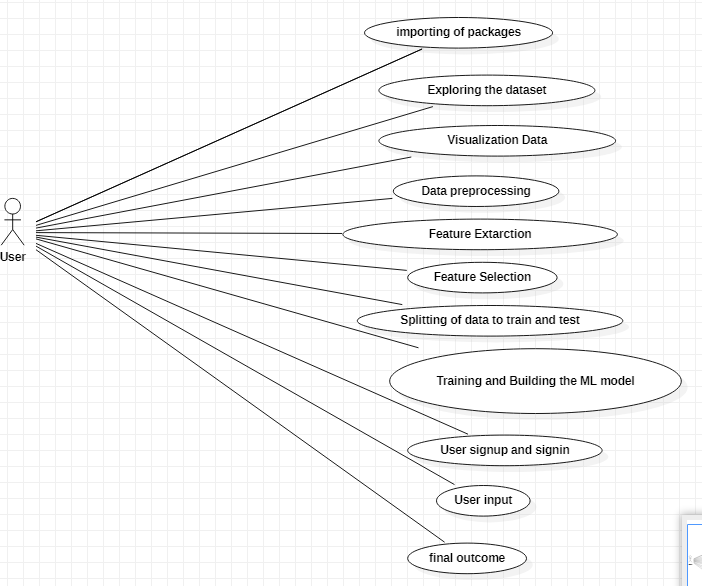
**GOALS:**

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.

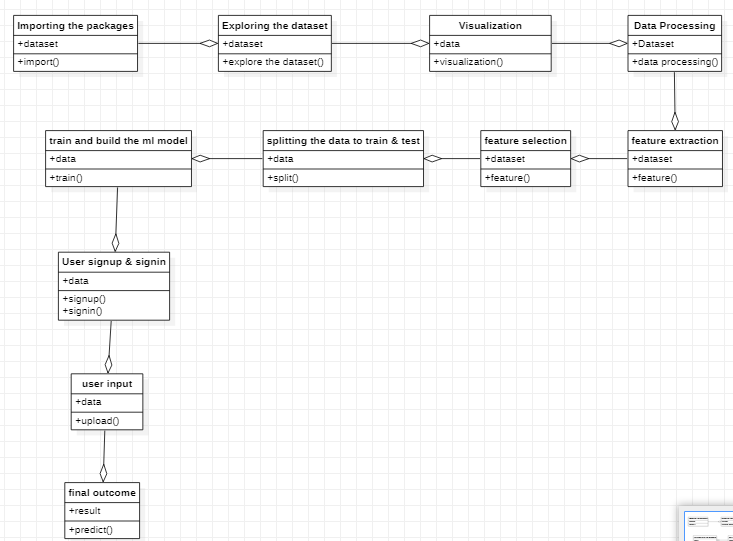
**Use case diagram:**

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.



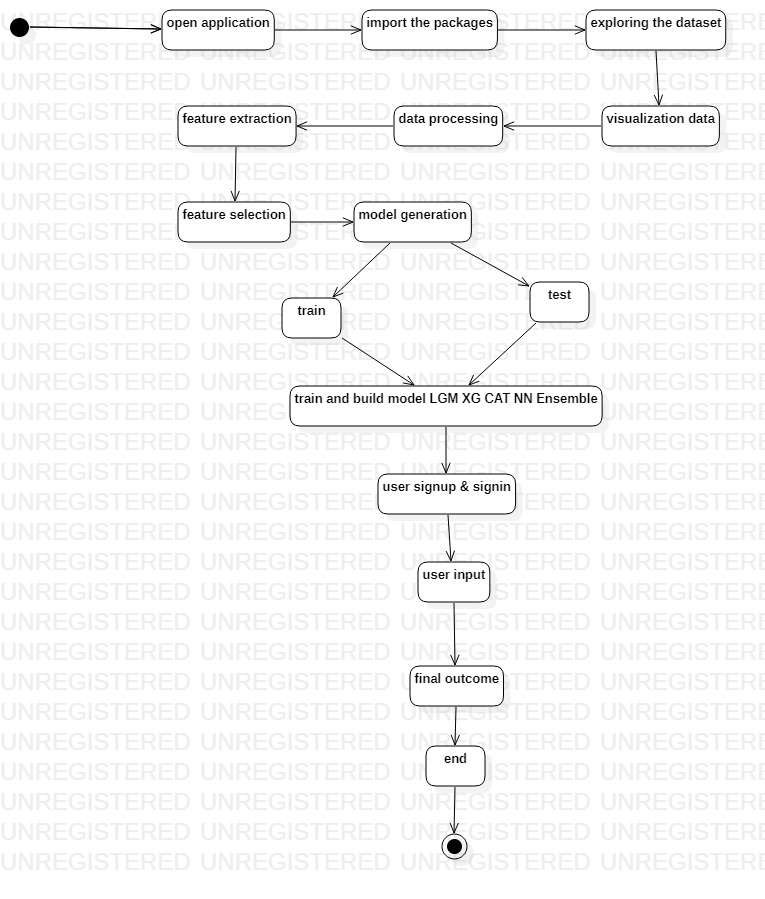
**Class diagram:**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

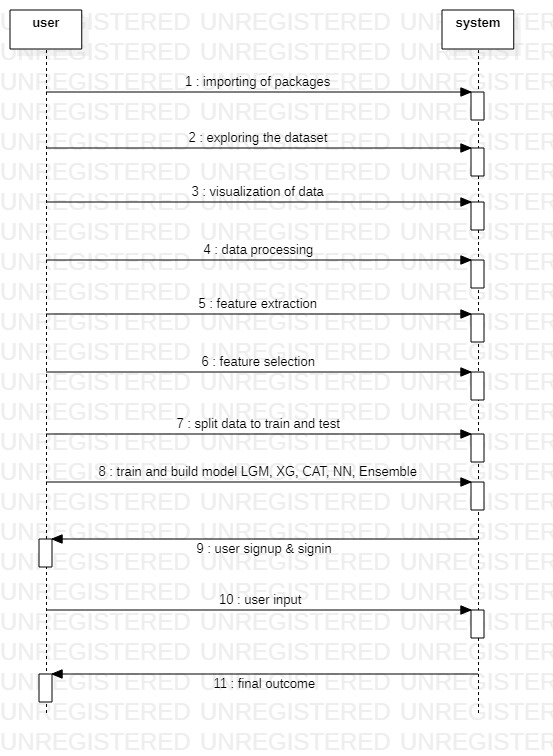


**Activity diagram:**

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

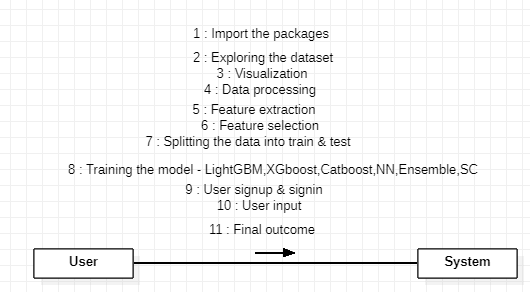
**Sequence diagram:**

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".



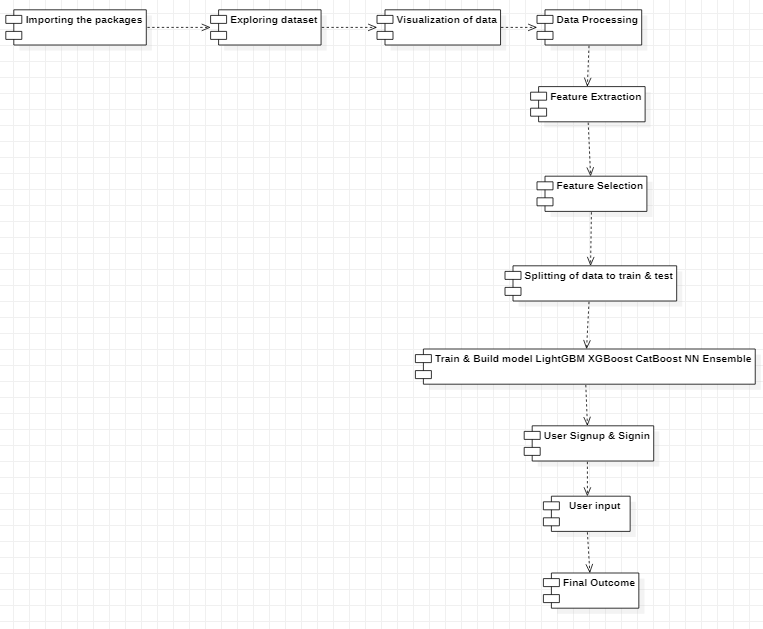
**Collaboration diagram:**

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

****

**Component diagram:**

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.



**Deployment diagram:**

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

****

**IMPLEMENTATION**

1. **IMPLEMENTATION**

MODULES:

* + Data exploration: using this module we will load data into system
  + Processing: Using the module we will read data for processing
  + Splitting data into train & test: using this module data will be divided into train & test
  + Model generation: Model building- Original dataset- CCMI Features {Random Forest, XGBoost, KNN, MLP, Stacking Classifier (RF + ET with LightGBM), Voting Classifier (RF + AB)
  + User signup & login: Using this module will get registration and login
  + User input: Using this module will give input for prediction
  + Prediction: final predicted displayed

**Note:** As an extension we applied an ensemble method combining the predictions of multiple individual models to produce a more robust and accurate final prediction.

However, we can further enhance the performance by exploring other ensemble techniques such as Voting Classifier RF + AB and Stacking Classifier using RF + ET with LightGBM which got 100% of accuracy.

**Algorithms:**

Random Forest: Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

XGBoost: XgBoost stands for Extreme Gradient Boosting, which was proposed by the researchers at the University of Washington. It is a library written in C++ which optimizes the training for Gradient Boosting.

KNN: k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

MLP: [multilayer perceptron (MLP)](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html) is a feedforward artificial neural network model that maps input data sets to a set of appropriate outputs. An MLP consists of multiple layers and each layer is fully connected to the following one. The nodes of the layers are neurons with nonlinear activation functions, except for the nodes of the input layer. Between the input and the output layer there may be one or more nonlinear hidden layers.

Stacking Classifier: Stacking is a way of ensembling classification or regression models it consists of two-layer estimators. The first layer consists of all the baseline models that are used to predict the outputs on the test datasets. The second layer consists of Meta-Classifier or Regressor which takes all the predictions of baseline models as an input and generate new predictions.

Voting Classifier: Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.

**6.2 SAMPLE CODE:**

# Necessary imports

import numpy as np

import pandas as pd

import tensorflow as tf

import matplotlib.pyplot as plt

import seaborn as sns

import time

import multiprocessing

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from collections import Counter

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.model\_selection import train\_test\_split

from imblearn.over\_sampling import SMOTE, ADASYN

from sklearn.metrics import confusion\_matrix, r2\_score, mean\_squared\_error

from sklearn.metrics import precision\_score, recall\_score, f1\_score, roc\_auc\_score, accuracy\_score, classification\_report, precision\_recall\_curve

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv("/kaggle/input/network-anamoly-detection/Train.txt",sep=",",names=["duration","protocoltype","service","flag","srcbytes","dstbytes","land", "wrongfragment","urgent","hot","numfailedlogins","loggedin", "numcompromised","rootshell","suattempted","numroot","numfilecreations", "numshells","numaccessfiles","numoutboundcmds","ishostlogin",

"isguestlogin","count","srvcount","serrorrate", "srvserrorrate",

"rerrorrate","srvrerrorrate","samesrvrate", "diffsrvrate", "srvdiffhostrate","dsthostcount","dsthostsrvcount","dsthostsamesrvrate", "dsthostdiffsrvrate","dsthostsamesrcportrate",

"dsthostsrvdiffhostrate","dsthostserrorrate","dsthostsrvserrorrate",

"dsthostrerrorrate","dsthostsrvrerrorrate","attack", "lastflag"])

df.head()

df.drop(['land','urgent','numfailedlogins','numoutboundcmds'],axis=1,inplace=True)

df.isna().sum()

df.select\_dtypes(exclude=[np.number])

df['attack'].loc[df['attack']!='normal']='attack'

le=LabelEncoder()

df['protocoltype']=le.fit\_transform(df['protocoltype'])

df['service']=le.fit\_transform(df['service'])

df['flag']=le.fit\_transform(df['flag'])

df['attack']=le.fit\_transform(df['attack'])

plt.figure(figsize=(20,15))

sns.heatmap(df.corr())

X=df.drop(['attack'],axis=1)

y=df['attack']

sns.countplot(df['attack'])

print("Class distribution: {}".format(Counter(y)))

scaler = StandardScaler()

scaler.fit(X)

X\_transformed = scaler.transform(X)

lr=LogisticRegression()

lr.fit(X\_transformed,y)

lr\_pred=lr.predict(X\_transformed)

lr\_df=pd.DataFrame()

lr\_df['actual']=y

lr\_df['pred']=lr\_pred

lr\_df.head()

print(accuracy\_score(y, lr\_pred))

print(classification\_report(y, lr\_pred))

rf=RandomForestClassifier()

rf.fit(X\_transformed,y)

rf\_pred=rf.predict(X\_transformed)

rf\_df=pd.DataFrame()

rf\_df['actual']=y

rf\_df['pred']=rf\_pred

rf\_df.head()

print(accuracy\_score(y, lr\_pred))

print(classification\_report(y, lr\_pred))

test\_df = pd.read\_csv("/kaggle/input/network-anamoly-detection/Test.txt",sep=",",names=["duration","protocoltype","service","flag","srcbytes","dstbytes","land", "wrongfragment","urgent","hot","numfailedlogins","loggedin", "numcompromised","rootshell","suattempted","numroot","numfilecreations", "numshells","numaccessfiles","numoutboundcmds","ishostlogin",

"isguestlogin","count","srvcount","serrorrate", "srvserrorrate",

"rerrorrate","srvrerrorrate","samesrvrate", "diffsrvrate", "srvdiffhostrate","dsthostcount","dsthostsrvcount","dsthostsamesrvrate", "dsthostdiffsrvrate","dsthostsamesrcportrate",

"dsthostsrvdiffhostrate","dsthostserrorrate","dsthostsrvserrorrate",

"dsthostrerrorrate","dsthostsrvrerrorrate","attack", "lastflag"])

test\_df.head()

test\_df.select\_dtypes(exclude=[np.number])

test\_df['attack'].loc[test\_df['attack']!='normal']='attack'

test\_df['protocoltype']=le.fit\_transform(test\_df['protocoltype'])

test\_df['service']=le.fit\_transform(test\_df['service'])

test\_df['flag']=le.fit\_transform(test\_df['flag'])

test\_df['attack']=le.fit\_transform(test\_df['attack'])

test\_df.drop(['land','urgent','numfailedlogins','numoutboundcmds'],axis=1,inplace=True)

X\_test=test\_df.drop(['attack'],axis=1)

y\_test=test\_df['attack']

sns.countplot(test\_df['attack'])

X\_test\_transformed = scaler.transform(X\_test)

test\_pred=rf.predict(X\_test\_transformed)

rf\_test\_df=pd.DataFrame()

rf\_test\_df['actual']=y\_test

rf\_test\_df['pred']=test\_pred

rf\_test\_df.head()

target\_names=["attack","normal"]

print(classification\_report(y\_test, test\_pred,target\_names=target\_names))

tn, fp, fn, tp = confusion\_matrix(y\_test, test\_pred).ravel()

print("True Negatives:",tn)

print("False Positives:",fp)

print("False Negatives:",fn)

print("True Positives:",tp)

**SOFTWARE ENVIRONMENT**

**7. SOFTWARE ENVIRONMENT**

**MACHINE LEARNING:**

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

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

## Challenges in Machines Learning:-

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

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

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

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

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

**Issue of over fitting & under fitting** − If the model is over fitting or under fitting, it cannot be represented well for the problem.

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

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

**DEEP LEARNING**

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don’t need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain’s biological neurons, and they are designed to learn from large amounts of data.

## What is Anaconda for Python?

Anaconda Python is a free, open-source platform that allows you to write and execute code in the programming language Python. It is by continuum.io, a company that specializes in Python development. The Anaconda platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over [thirty](https://www.anaconda.com/blog/10-years-of-data-science-innovation-anacondas-commitment-to-the-open-source-python-community)[million people](https://www.anaconda.com/blog/10-years-of-data-science-innovation-anacondas-commitment-to-the-open-source-python-community) worldwide and is available for Windows, macOS, and Linux.

People like using Anaconda Python because it simplifies package deployment and management. It also comes with a large number of libraries/packages that you can use for your projects. Since Anaconda Python is free and open-source, anyone can contribute to its development.

**What is Anaconda for Python?**

Anaconda software helps you create an environment for many different versions of Python and package versions. Anaconda is also used to install, remove, and upgrade packages in your project environments. Furthermore, you may use Anaconda to deploy any required project with a few mouse clicks. This is why it is perfect for beginners who want to learn Python.

Now that you know what Anaconda Python is, let's look at how to install it.

**How to install Anaconda for Python?**



To install Anaconda, just head to the Anaconda Documentation website and follow the instructions to download the installer for your operating system. Once the installer successfully downloads, double-click on it to start the installation process.

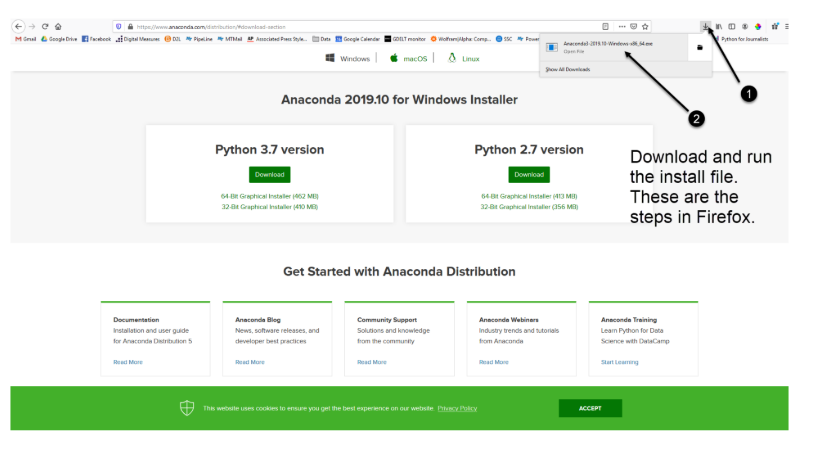
Follow the prompts and agree to the terms and conditions. When you are asked if you want to "add Anaconda to my PATH environment variable," make sure that you select "yes." This will ensure that Anaconda is added to your system's PATH, which is a list of directories that your operating system uses to find the files it needs.

Once the installation is complete, you will be asked if you want to "enable Anaconda as my default Python." We recommend selecting "yes" to use Anaconda as your default Python interpreter.

### **Python Anaconda Installation**

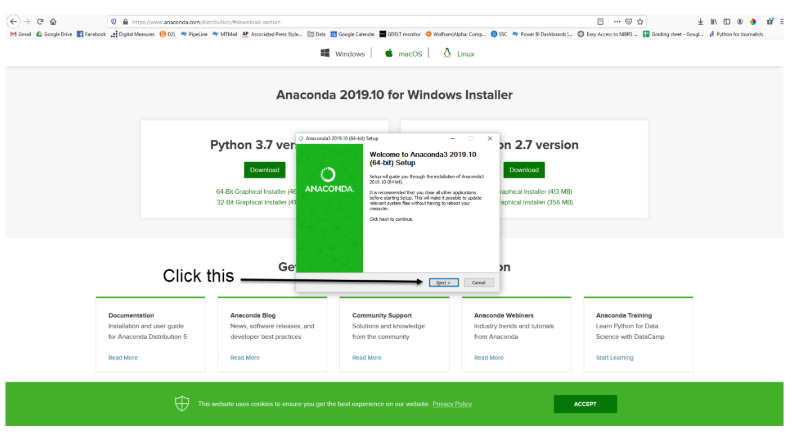
Next in the Python anaconda tutorial is its installation. The latest version of Anaconda at the time of writing is 2019.10. Follow these steps to download and install Anaconda on your machine:

1. Go to this link and download Anaconda for Windows, Mac, or Linux: – [Download anaconda](https://www.anaconda.com/distribution/)

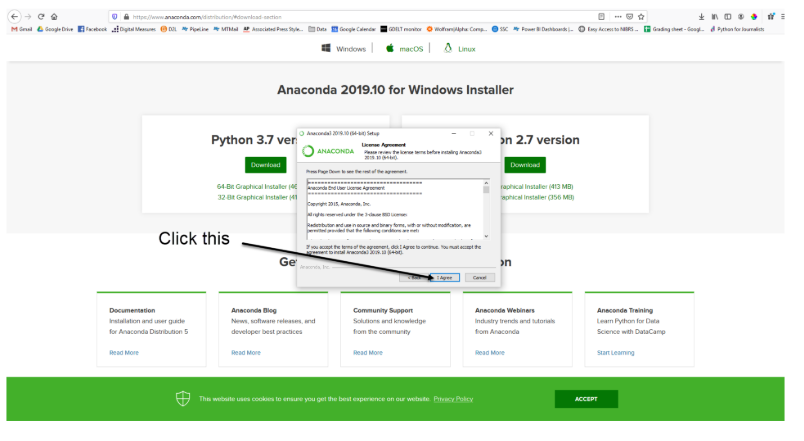


You can download the installer for Python 3.7 or for Python 2.7 (at the time of writing). And you can download it for a 32-bit or 64-bit machine.

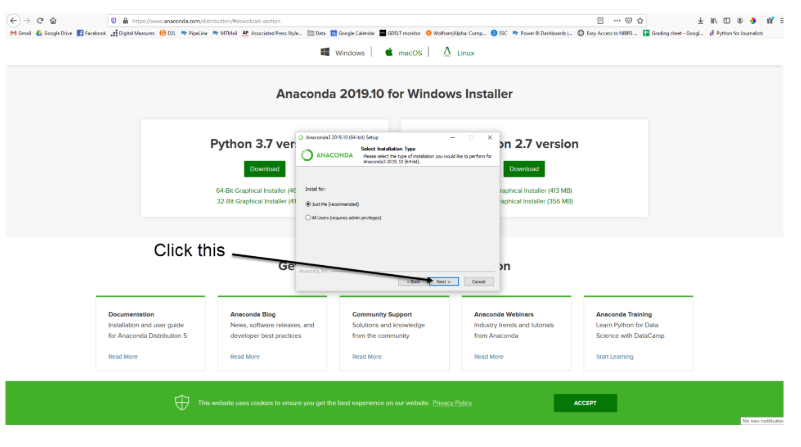
2. Click on the downloaded .exe to open it. This is the Anaconda setup. Click next.



3. Now, you’ll see the license agreement. Click on ‘I Agree’.



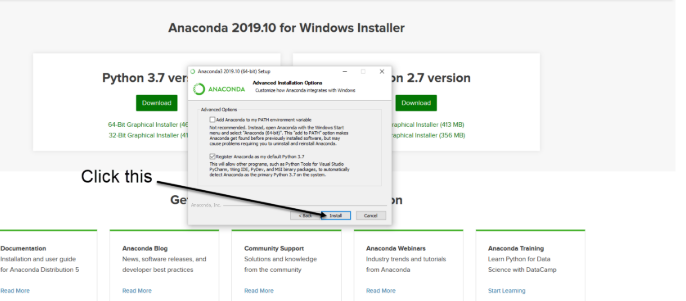
4. You can install it for all users or just for yourself. If you want to install it for all users, you need administrator privileges.



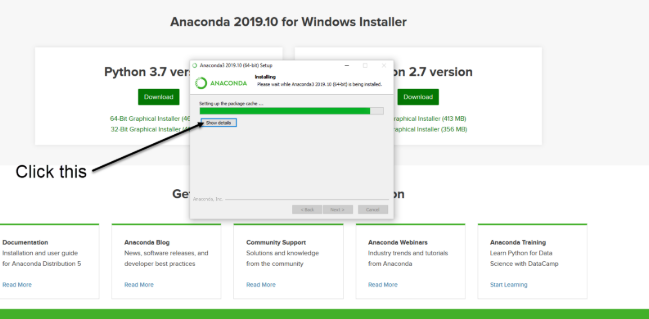
5. Choose where you want to install it. Here, you can see the available space and how much you need.



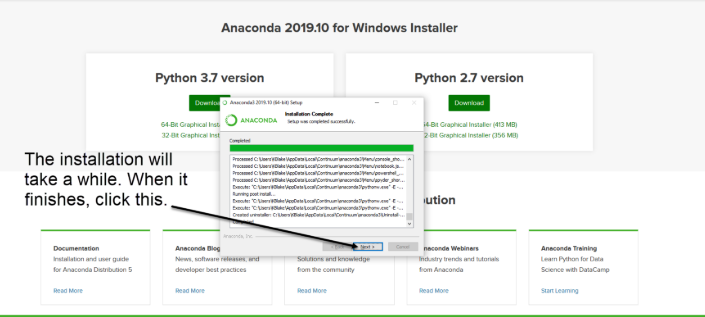
6. Now, you’ll get some advanced options. You can add Anaconda to your system’s PATH environment variable, and register it as the primary system Python 3.7. If you add it to PATH, it will be found before any other installation. Click on ‘Install’.



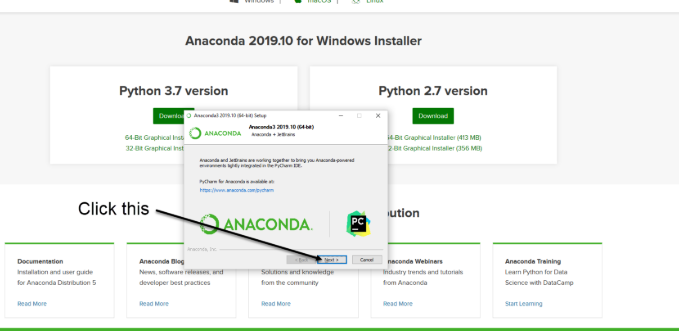
7. It will unpack some packages and extract some files on your machine. This will take a few minutes.



8. The installation is complete. Click Next.



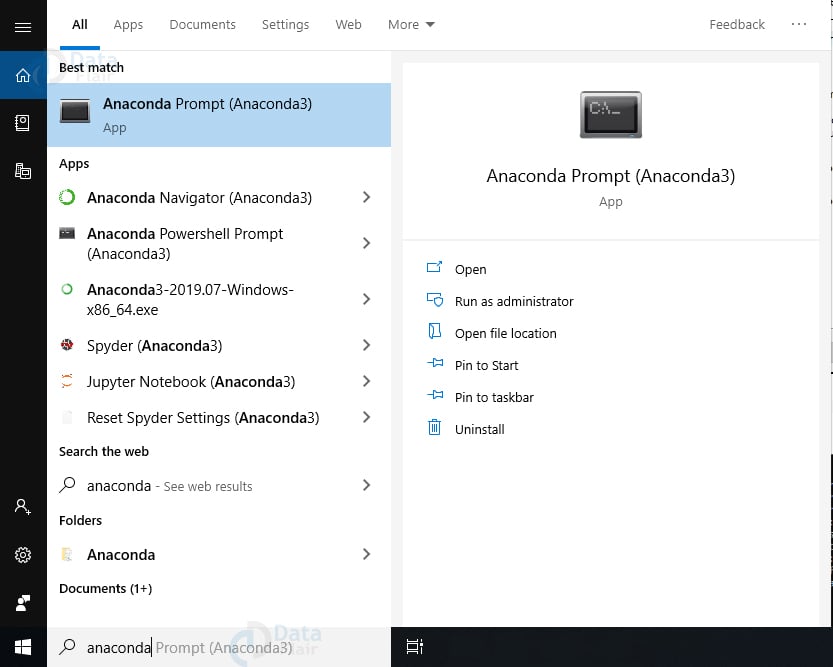
9. This screen will inform you about PyCharm. Click Next.



10. The installation is complete. You can choose to get more information about Anaconda cloud and how to get started with Anaconda. Click Finish.



11. If you search for Anaconda now, you will see the following options:



**PYTHON LANGUAGE:**

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language. For example, x = 10 Here, x can be anything such as String, int, etc.

## Features in Python:

There are many features in Python, some of which are discussed below as follows:

### **1. Free and Open Source**

[Python](https://www.geeksforgeeks.org/python-programming-language/)language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword. [Download Python](https://www.python.org/downloads/) Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

### **2. Easy to code**

Python is a [high-level programming language](https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/). Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

### 3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python’s syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

### **4. Object-Oriented Language**

One of the key features of [Python is Object-Oriented programming](https://www.geeksforgeeks.org/python-oops-concepts/). Python supports object-oriented language and concepts of classes, object encapsulation, etc.

### **5. GUI Programming Support**

Graphical User interfaces can be made using a module such as [PyQt5](https://www.geeksforgeeks.org/pyqt5-qaction/), PyQt4, wxPython, or [Tk in python](https://www.geeksforgeeks.org/python-gui-tkinter/). PyQt5 is the most popular option for creating graphical apps with Python.

### **6. High-Level Language**

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

### **7. Extensible feature**

Python is an **Extensible** language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

### 8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program’s issues once you understand how to [interpret](https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/)Python’s error traces. Simply by glancing at the code, you can determine what it is designed to perform.

### **9. Python is a Portable language**

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as [Linux](https://www.geeksforgeeks.org/introduction-to-linux-operating-system/), Unix, and Mac then we do not need to change it, we can run this code on any platform.

### **10. Python is an integrated language**

Python is also an integrated language because we can easily integrate Python with other languages like C, [C++](http://www.geeksforgeeks.org/c-plus-plus/), etc.

### **11. Interpreted Language:**

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, [Java](https://www.geeksforgeeks.org/java/), etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called **byte code**.

### **12. Large Standard Library**

Python has a large [standard library](https://www.geeksforgeeks.org/libraries-in-python/) that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as [regular expression](https://www.geeksforgeeks.org/regular-expression-python-examples-set-1/)s, [unit-testing](https://www.geeksforgeeks.org/unit-testing-software-testing/), web browsers, etc.

### **13. Dynamically Typed Language**

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

### **14. Frontend and backend development**

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like JavaScript. Backend is the strong forte of Python it’s extensively used for this work cause of its frameworks like [Django](https://www.geeksforgeeks.org/django-tutorial/)and [Flask](https://www.geeksforgeeks.org/flask-creating-first-simple-application/).

### 15. Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write int y = 18 if the integer value 15 is set to y. You may just type y=18.

**LIBRARIES/PACKGES:-**

**Tensor flow**

Tensor Flow is a [free](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library for dataflow and differentiable programming](https://en.wikipedia.org/wiki/Library_(computing)) across a range of tasks. It is a symbolic math library, and is also used for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications such as [neural networks](https://en.wikipedia.org/wiki/Neural_networks). It is used for both research and production at [Google](https://en.wikipedia.org/wiki/Google).‍

TensorFlow was developed by the [Google Brain](https://en.wikipedia.org/wiki/Google_Brain) team for internal Google use. It was released under the [Apache 2.0](https://en.wikipedia.org/wiki/Apache_License) [open-source license](https://en.wikipedia.org/wiki/Open-source_license) on November 9, 2015.

**Numpy**

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

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

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

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

**Pandas**

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

**Matplotlib**

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

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

**Scikit – learn**

Scikit-learn provide a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

**SYSTEM TESTING**

**8. SYSTEM TESTING**

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

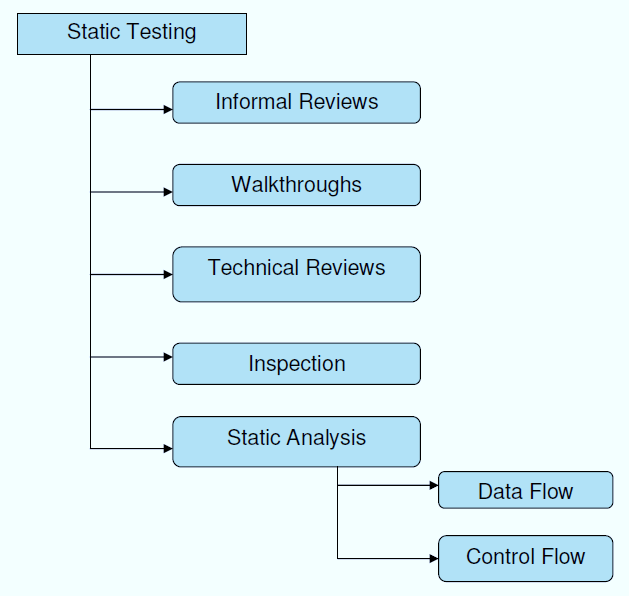
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

**8.1Software Testing Strategies:**

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

Static Testing:

The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.



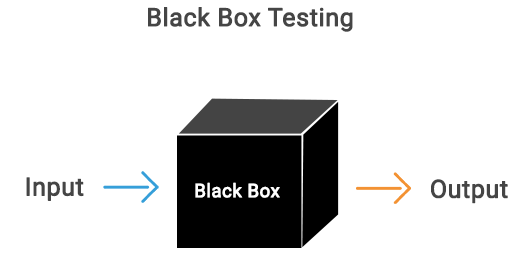
**Structural Testing:**

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software’s structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system’s behavior by comparing the tests’ outcomes with the results of previous iterations (control flow testing).



**Behavioral Testing:**

The final stage of testing focuses on the software’s reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user’s point of view. QA engineers usually have some specific information about a business or other purposes of the software (‘the black box’) to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.



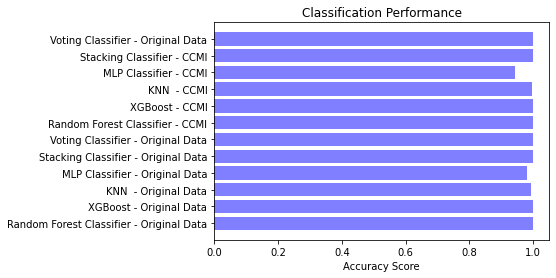
**8.2 TEST CASES:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **INPUT** | **If available** | **If not available** |
| 1 | User signup | User get registered into the application | There is no process |
| 2 | User sign in | User get login into the application | There is no process |
| 3 | Enter input for prediction | Prediction result displayed | There is no process |

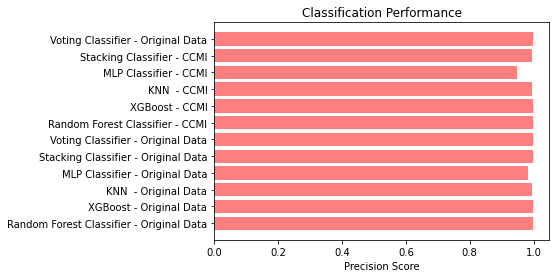
**SCREENS**

1. **SCREENSHOTS**

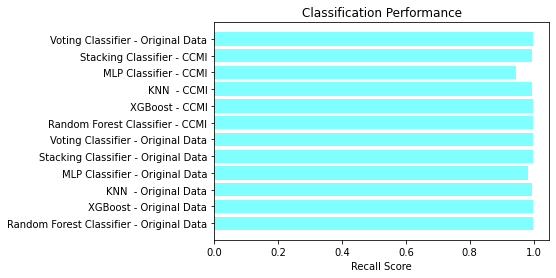
SCREENS:



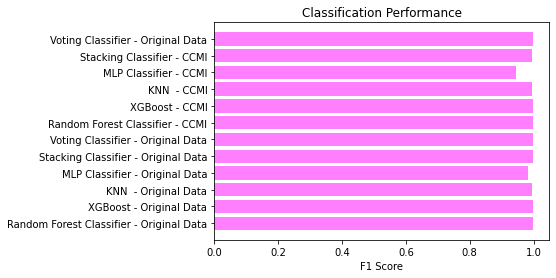
ACCURACY COMPARISON GRAPHS



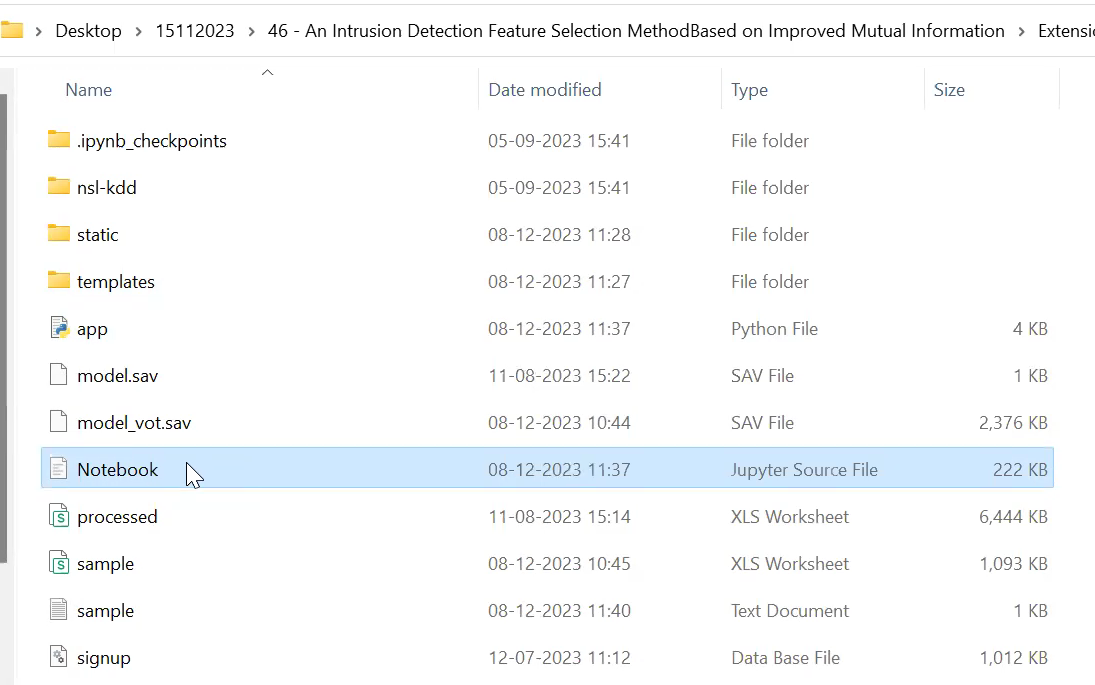
PRECISION COMPARISON GRAPHS



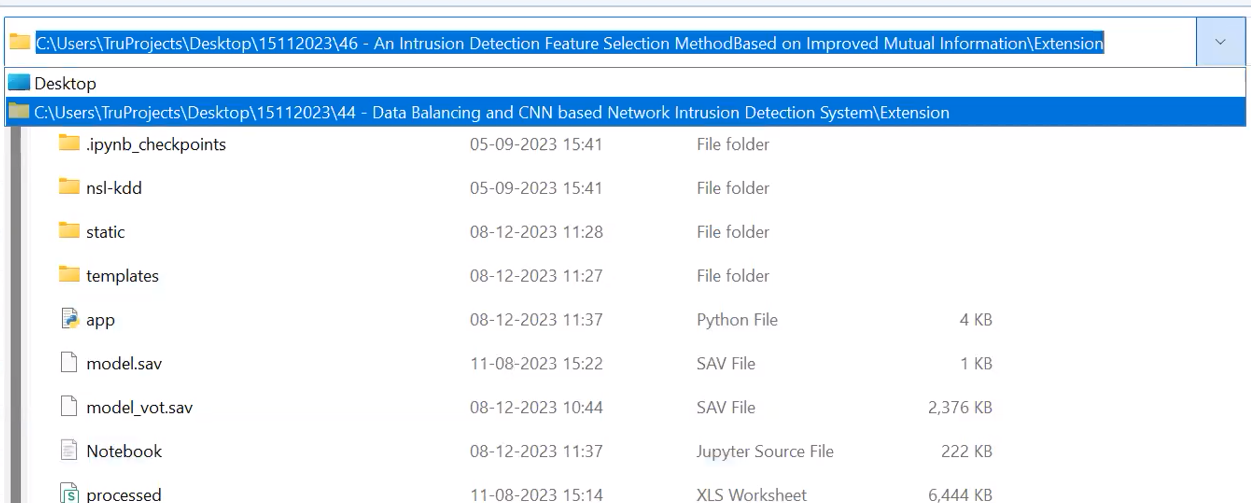
RECALL COMPARISON GRAPHS



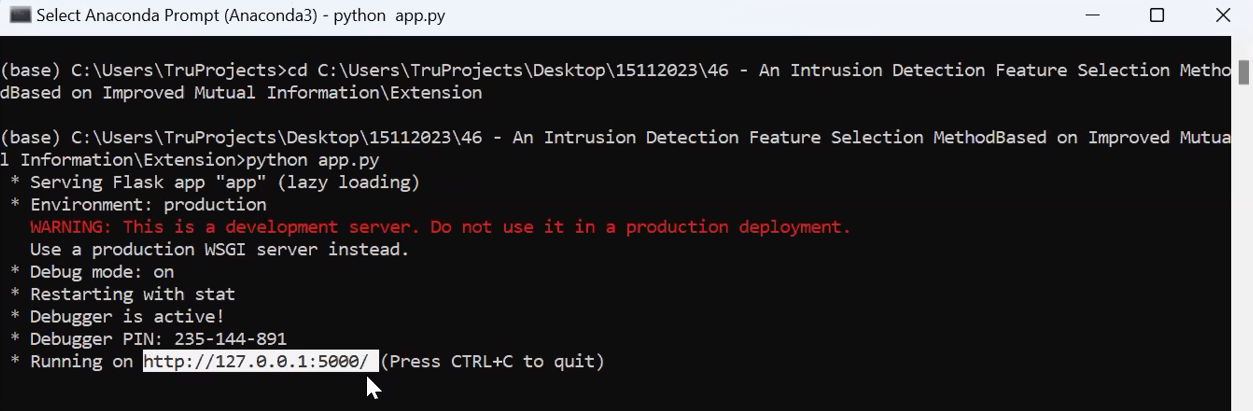
F1 SCORE COMPARISON GRAPHSSTEP 1



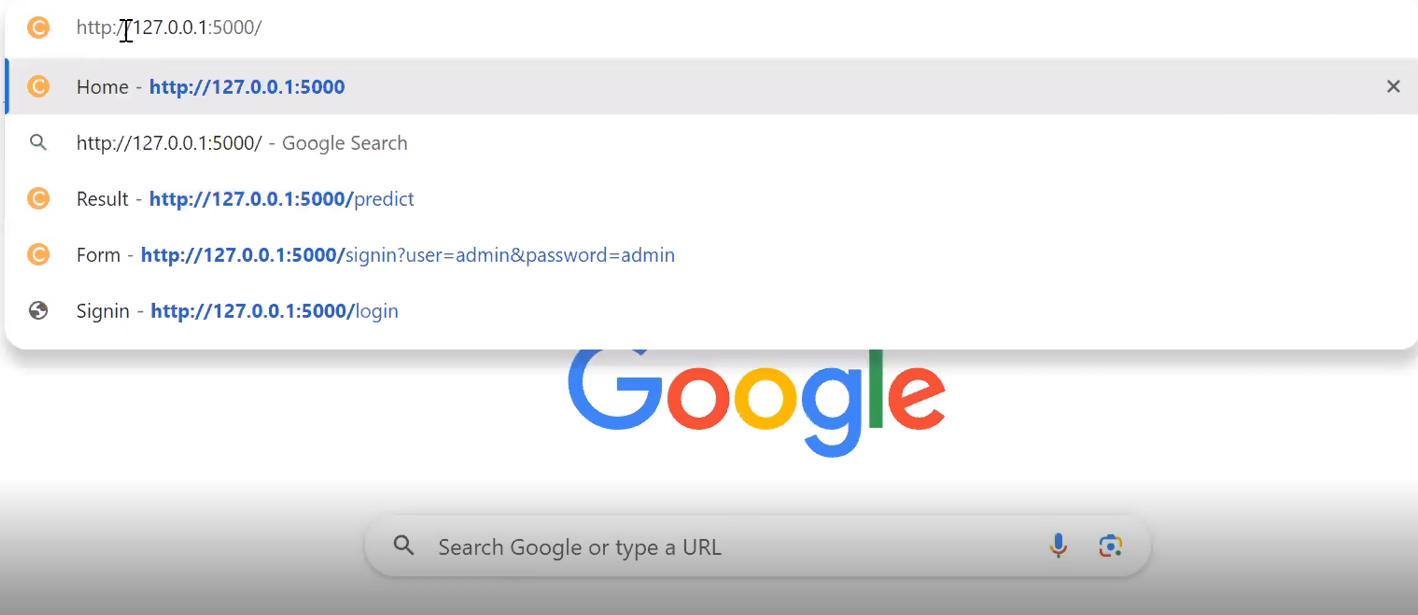
STEP 2



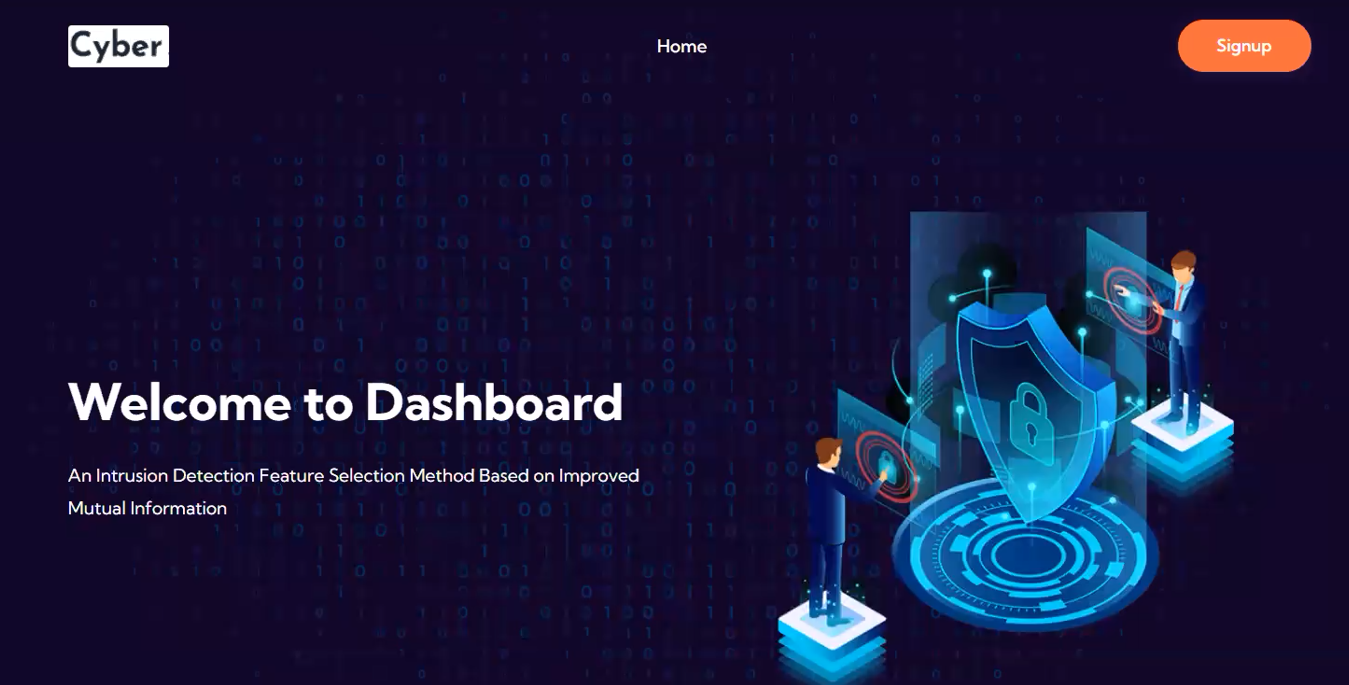
STEP 3



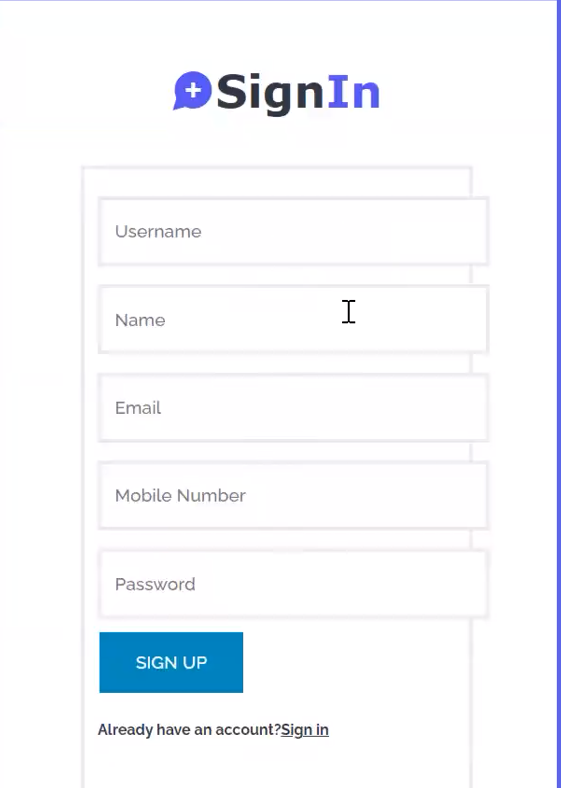
STEP 4



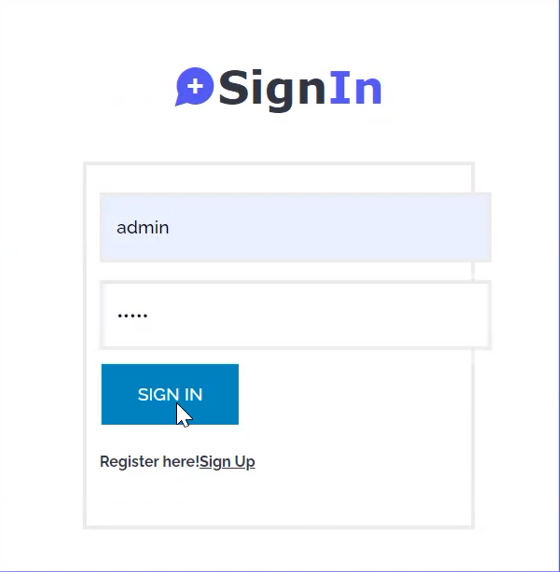
STEP 5



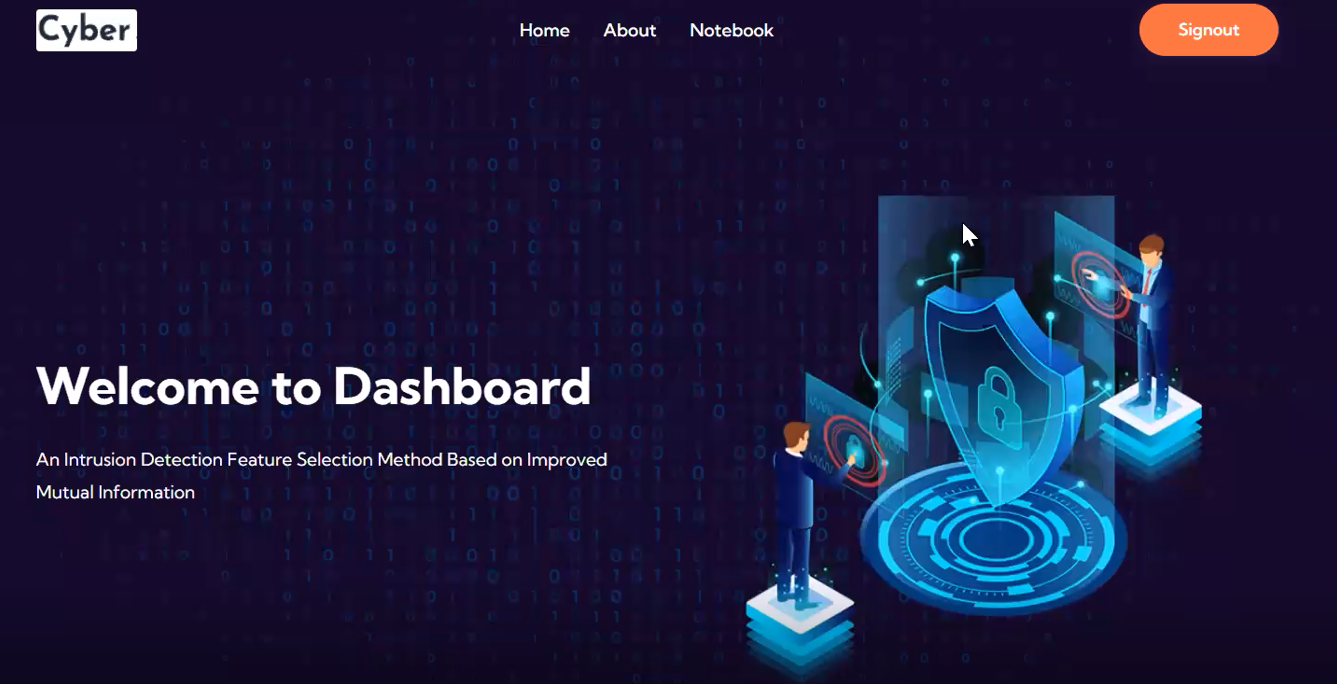
STEP 6



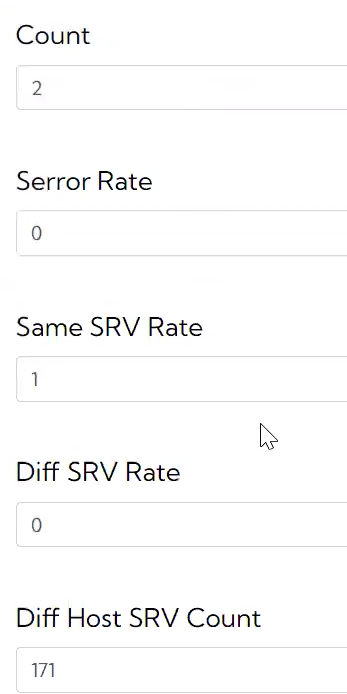
STEP 7



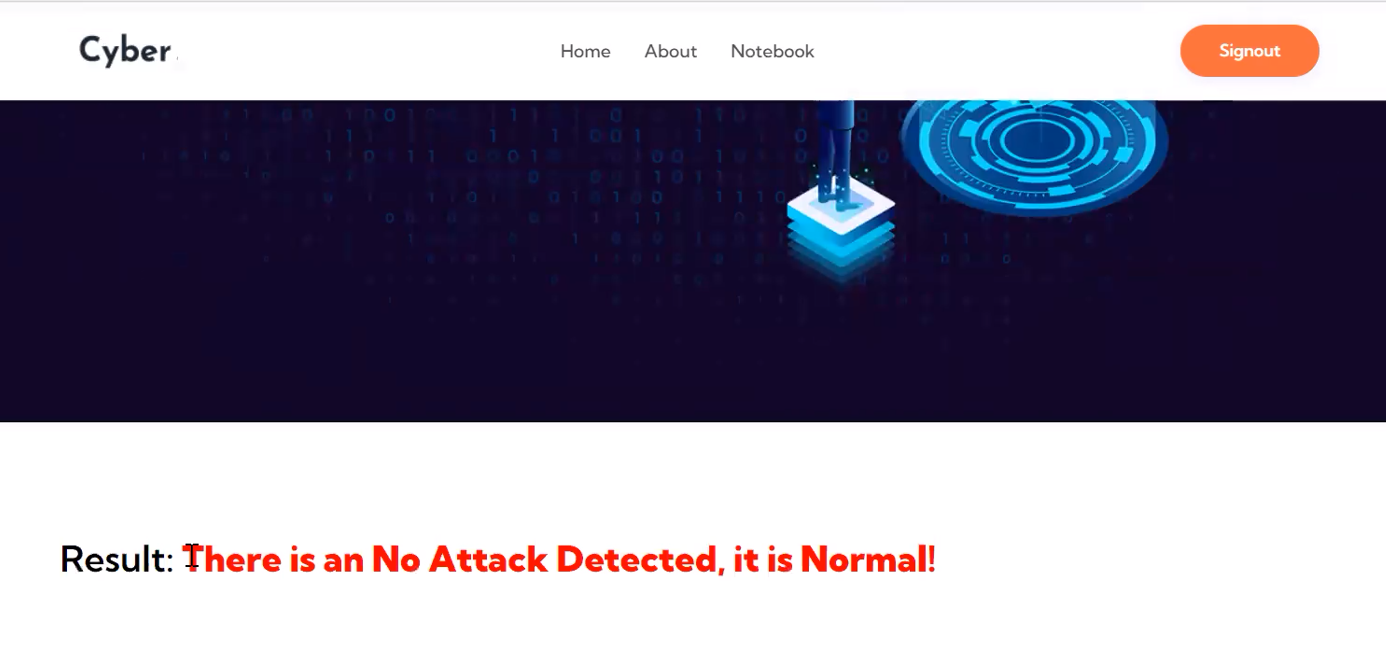
STEP 8



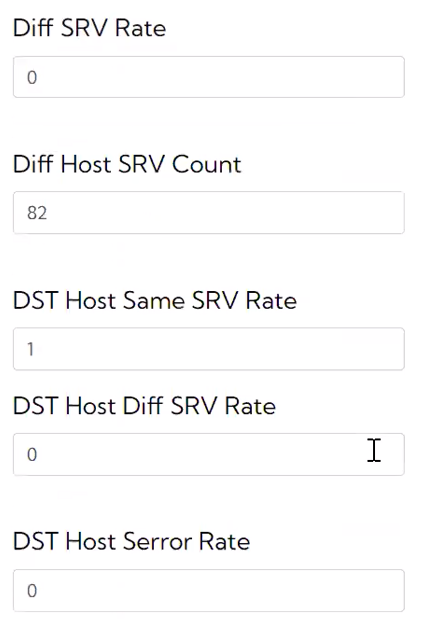
STEP 9



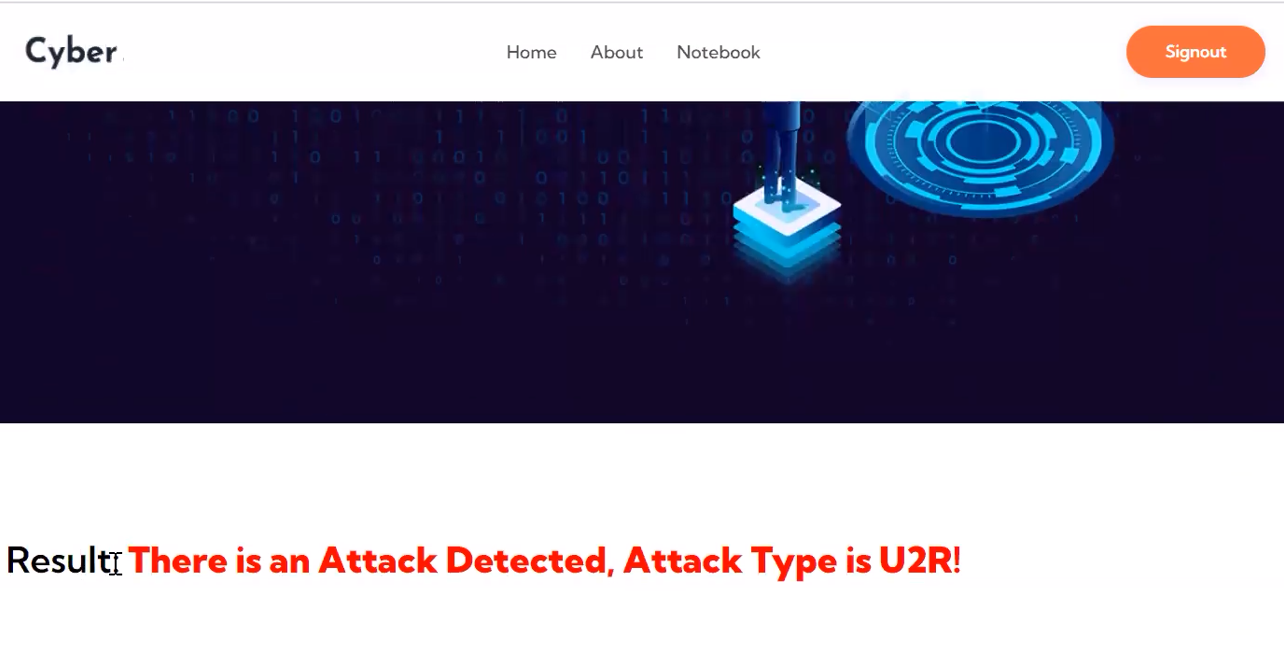
STEP 10



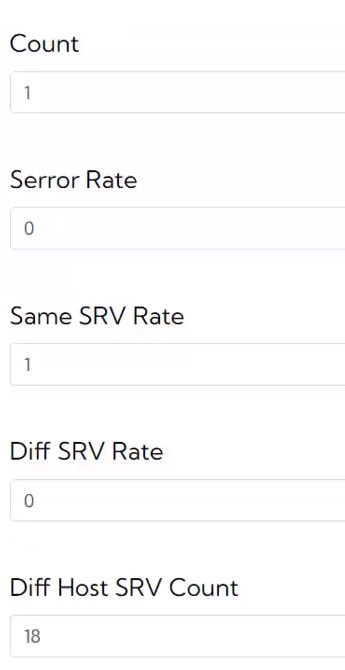
STEP 11



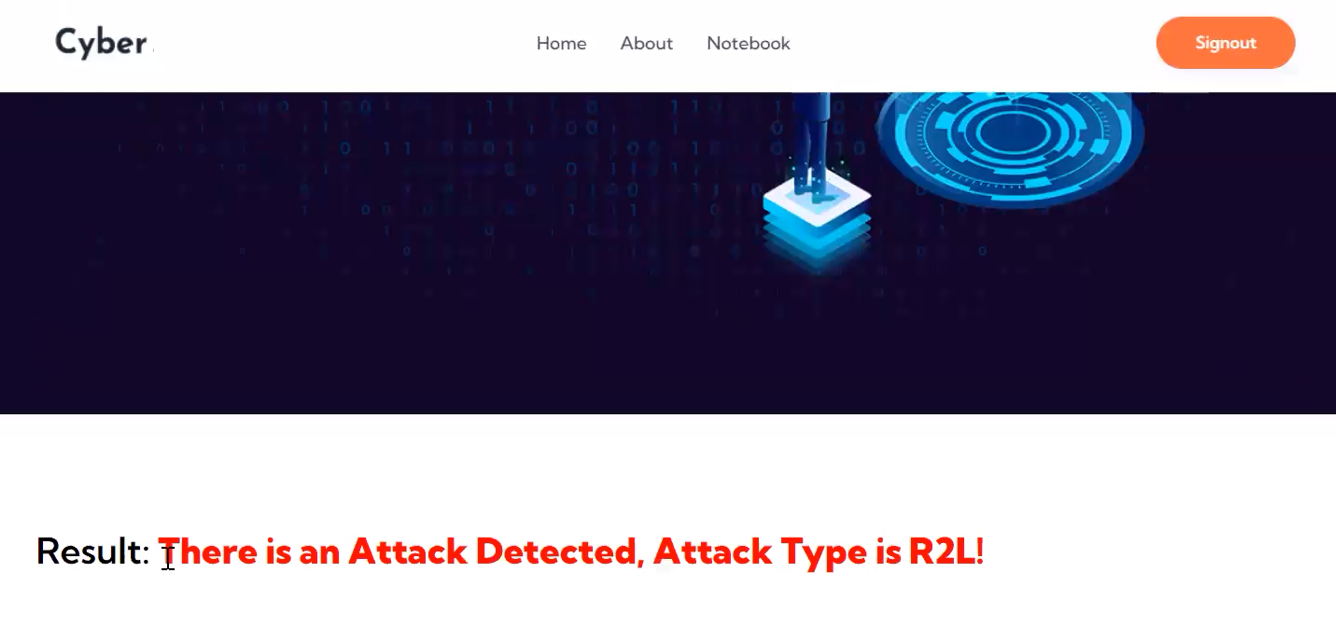
STEP 12



STEP 13



STEP 14



**CONCLUSION**

**10. CONCLUSION**

Many machine learning methods have been proposed to improve the efficiency and accuracy of intrusion detection systems (IDS), existing IDS still have some problems due to inappropriate feature selection. So, we proposed a new intrusion detection framework based on improved mutual information to cope with high-dimensional and unbalanced network traffic. We utilize the improved Pearson correlation coefficient, a normalized metric, to initially evaluate the redundancy of features.The proposed CCMI feature selection algorithm aims to select the optimal subset of features based on the correlation between features. The intrusion detection dataset NSL-KDD is experimentally evaluated on four classifiers RF, XGBoost, KNN, and MLP, using the proposed feature selection method. The experimental results show that the accuracy, precision, and MAE of the algorithm on the NSL-KDD dataset outperform the equivalent IDS without feature selection or using other feature selection methods, and also outperform the rest of the feature selection methods in terms of the number of required features and feature selection time.

**BIBILOGRAPHY**

**11. REFERENCES**

[1] Muna A L H, Moustafa N, Sitnikova E. Identification of malicious activities in industrial internet of things based on deep learning models[J]. Journal of information security and applications, 2018, 41: 1-11.

[2] Ilyas M U, Alharbi S A. Machine learning approaches to network intrusion detection for contemporary internet traffic[J]. Computing, 2022, 104(5): 1061- 1076.

[3] Umar M A, Zhanfang C, Liu Y. Network intrusion detection using wrapper-based decision tree for feature selection[C]//Proceedings of the 2020 International Conference on Internet Computing for Science and Engineering. 2020: 5-13.

[4] Hota H S, Shrivas A K. Decision tree techniques applied on NSLKDD data and its comparison with various feature selection techniques[M]//Advanced Computing, Networking and Informatics-Volume 1. Springer, Cham, 2014: 205-211.

[5] Abdullah M, Alshannaq A, Balamash A, et al. Enhanced intrusion detection system using feature selection method and ensemble learning algorithms[J]. International Journal of Computer Science and Information Security (IJCSIS), 2018, 16(2): 48-55.

[6] Zhou Y, Cheng G, Jiang S, et al. Building an efficient intrusion detection system based on feature selection and ensemble classifier[J]. Computer networks, 2020, 174: 107247.

[7] Thakkar A, Lohiya R (2021a) Attack classification using feature selection techniques: a comparative study. J. Ambient Intell Human Comput 12(1):1249–1266.

[8] Kullback S. Information theory and statistics[M]. Courier Corporation, 1997.

[9] Peng H, Long F, Ding C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and minredundancy[J]. IEEE Transactions on pattern analysis and machine intelligence, 2005, 27(8): 1226-1238.

[10] Jakulin A, Bratko I. Testing the significance of attribute interactions[C] //Proceedings of the twenty-first international conference on Machine learning. 2004: 52.

[11] Wang J, Wei J M, Yang Z, et al. Feature selection by maximizing independent classification information[J]. IEEE transactions on knowledge and data engineering, 2017, 29(4): 828-841.

[12] Wang Y, Li X, Ruiz R. Feature Selection With Maximal Relevance and Minimal Supervised Redundancy[J]. IEEE Transactions on Cybernetics, 2022.

[13] Tavallaee M, Bagheri E, Lu W, et al. A detailed analysis of the KDD CUP 99 data set[C]//2009 IEEE symposium on computational intelligence for security and defense applications. Ieee, 2009: 1-6.

[14] Lee W, Stolfo S J, Mok K W. A data mining framework for building intrusion detection models[C]//Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344). IEEE, 1999: 120-132