

INTRUSION DETECTION PREDICTION USING DATA SCIENCE TECHNIQUE

A PROJECT REPORT

Submitted by

KARTHICK ARAVIND B [211419104318]

T. PENIEL BRANHAM [211419104321]

BALAMURUGAN S [211419104036]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

BONAFIDE CERTIFICATE

Certified that this project report “**INTRUSION DETECTION PREDICTION USING DATA SCIENCE TECHNIQUE**” is the bonafide work of “**Karthick Aravind B [211419104318], T. Peniel Branham [211419104321], Balamurugan S [211419104036]**” who carried out the project work under my supervision.

SIGNATURE

**Dr.L.JABASHEELA,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT**

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

**Mr.THYAGARAJAN.C M.E.,(Ph.D),
SUPERVISOR
ASSISTANT PROFESSOR**

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION BY THE STUDENT

We **Karthick Aravind B [211419104318]**, **T. Peniel Branham [211419104321]**, **Balamurugan S [211419104036]** hereby declare that this project report titled **“INTRUSION DETECTION PREDICTION USING DATA SCIENCE TECHNIQUE”**, under the guidance of **Mr.THYAGARAJAN.C M.E.,(Ph.D.)**, is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

KARTHICK ARAVIND B

T. PENIEL BRANHAM

BALAMURUGAN S

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHI KUMAR, M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L.JABASHEELA, M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank our parents, friends, project Guide **Mr.THYAGARAJAN.C M.E.,(Ph.D.),** and coordinator **Dr.N.PUGHAZENDI M.E., Ph.D.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

KARTHICK ARAVIND B

T. PENIEL BRANHAM

BALAMURUGAN S

ABSTRACT

Intrusion Detection Systems are designed to safeguard the security needs of enterprise networks against cyber-attacks. However, networks suffer from several limitations, such as generating a high volume of low-quality alerts. The study has reviewed the state-of-the-art cyber-attack prediction based on Intrusion Alert, its models, and limitations. The ever-increasing frequency and intensity of intrusion attacks on computer networks worldwide intense research efforts towards the design of attack detection and prediction mechanisms. While there are a variety of intrusion *detection* solutions available, the *prediction* of network intrusion events is still under active investigation. Over the past, statistical methods have dominated the design of attack prediction methods. The analysis of dataset by supervised machine learning technique (SMLT) to capture several information's like, variable identification, univariate analysis, bivariate and multivariate analysis, missing value treatments etc. A comparative study between machine learning algorithms had been carried out in order to determine which algorithm is the most accurate in predicting the type cyberattacks. The results show that the effectiveness of the proposed machine learning algorithm technique can be compared with best accuracy, precision, Recall, F1 Score, Sensitivity, and Specificity.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF TABLES	viii
	LIST OF FIGURES	ix
	LIST OF SYMBOLS, ABBREVIATIONS	xi
1.	INTRODUCTION	1
	1.1 Overview	2
	1.2 Problem Definition	3
2.	LITERATURE SURVEY	4
3.	SYSTEM ANALYSIS	8
	3.1 Existing System	9
	3.1.1 Disadvantages of Existing System	9
	3.2 Proposed system	10
	3.2.1 Advantages of Proposed System	10
	3.3 Development Environment	11
4.	SYSTEM DESIGN	12
	4.1 UML Diagrams	13
	4.1.1 Use Case Diagram	13
	4.1.2 Class Diagram	14
	4.1.3 Sequence Diagram	15
	4.1.4 Activity Diagram	16

CHAPTER NO.	TITLE	PAGE NO.
	4.2 Entity Relationship Diagram (ERD)	17
	4.3 Work flow Diagram	18
5.	SYSTEM ARCHITECTURE	19
	5.1 Architecture Diagram	20
	5.2 Module Description	21
	5.2.1 Algorithm implementation	21
	5.2.2 SVM	23
	5.2.3 Adaboost Classifier	25
	5.2.4 Random Forest Classifier	26
	5.2.5 Voting Classifier	28
6.	SYSTEM IMPLEMENTATION	30
	6.1 Data Pre-processing	31
	6.2 Data visualization	34
7.	SYSTEM TESTING	36
	7.1 Test Cases	37
8.	CONCLUSION & FUTURE ENHANCEMENT	39
	8.1 Conclusion	40
	8.2 Future Enhancements	40
	APPENDICES	42
	A.1 Coding	43
	A.2 Sample Screens	54
	REFERENCES	58

LIST OF TABLES

TABLE NO.	TABLE DESCRIPTION	PAGE NO.
7.1.1	Test Case for User/Admin Signup	37
7.1.2	Test Case for User/Admin Login	37
7.1.3	Test Case for Getting Inputs from users/Admin	38

LIST OF FIGURES

FIGURE NO.	FIGURE DESCRIPTION	PAGE NO.
4.1.1	Use case diagram	13
4.1.2	Class diagram	14
4.1.3	Sequence diagram	15
4.1.4	Activity diagram	16
4.2	Entity Relationship Diagram	17
4.3	Work flow Diagram	18
5.1	Architecture diagram	20
5.2.1	Support Vector Machine	24
5.2.2	Module diagram for Implementing Support Vector Machine Algorithm	24
5.2.3	Adaboost Classifier	25
5.2.4	Module diagram for Implementing Adaboost Classifier Algorithm	26
5.2.5	Random Forest Classifier	27
5.2.6	Module diagram for Implementing Random Forest Classifier Algorithm	27
5.2.7	Voting Classifier	29
5.2.8	Module diagram for Implementing Voting Classifier Algorithm	29
6.1	Data Pre-processing	33
6.2	Module diagram for Data Pre-processing	33
6.3	Data visualization	35
6.4	Module diagram for Data visualization	35
A.2.1	Deploying the Program	54

A.2.2	Initializing the server	54
A.2.3	User Login Page	55
A.2.4	User Input Page	55
A.2.5	User Detail Submission	56
A.2.6	Officer Login Page	56
A.2.7	Data submissions	57
A.2.8	User feedback	57

LIST OF SYMBOLS, ABBREVIATIONS

IDS	Intrusion Detection System
SOC	Security Operations Center
DNNs	Deep Neural Networks
ML	Region Of Interest
NIDS	Network Intrusion Detection Systems
FP	False Positives
FN	False Negatives
TP	True Positives
TN	True Negatives
TPR	True Positive Rate
FPR	False Positive rate
SVM	Support Vector Machine
IOT	Internet of Things

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 Overview

The main objective of this project is to develop a machine learning model for intrusion detection Prediction, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm. The scope of the project is that an IPS prevents attacks by dropping malicious packets, blocking offending IPs and alerting security personnel to potential threats. Such a system usually uses a pre-existing database for signature recognition and can be programmed to recognize attacks based on traffic and behavioral anomalies. An Intrusion Detection System (IDS) is a monitoring system that detects suspicious activities and generates alerts when they are detected. Based upon these alerts, it is necessary to know the attack type. So, the security operations center (SOC) analyst or incident responder can investigate the issue and take the appropriate actions to remediate the threat.

Although intrusion detection systems monitor networks for potentially malicious activity, they are also disposed to false alarms. Hence, organizations need to fine-tune their IDS products when they first install them. It means properly setting up the intrusion detection systems to recognize what normal traffic on the network looks like as compared to malicious activity.

1.2 Problem Definition

The problem definition for the proposed model to build a machine learning model for anomaly detection is to develop a reliable and efficient system that can accurately identify and detect fraudulent activities, suspicious activities, network intrusions, and other abnormal events that are challenging to detect. The model should apply appropriate data science techniques such as variable identification, data preprocessing, and visualization to build a model based on a previous dataset where the algorithm learns and gets trained using various algorithms for better comparisons. The goal is to evaluate the performance metrics of the model and compare them to select the most appropriate algorithm for anomaly detection. The proposed model aims to contribute to the development of an efficient and effective anomaly detection system that can help organizations prevent security breaches and protect their critical assets and data from potential threats.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

Title : An Intrusion Detection Method Based on Machine Learning and State Observer for Train-Ground Communication Systems

Author: Bing Gao, Bing Bu, Wei Zhang, Xiang Li

Year : 2022

This paper proposes an intrusion detection method to ensure the information security of the train-ground communication system in urban rail transit. The communication-based train control system (CBTC) uses wireless communication protocols to transmit control commands, but faces potential security risks. The proposed method uses machine learning and state observer to detect and recognize various attacks, including anomalies of wireless network data and train physical states. The detection system includes two layers: one to detect and identify wireless network attacks using machine learning algorithms, and another to detect abnormal physical state of train operation. The results of both layers are combined to give a comprehensive intrusion detection result. Simulation results show that the proposed method is effective and practical.

Title : Early Detection of Network Intrusions Using a GAN-Based One-Class Classifier

Author: Taehoon Kim, Wooguil Pak

Year : 2022

Detecting network intrusions early is crucial for network security. Most current methods use features for full sessions, making early detection difficult. The proposed

method uses packet data as features to determine malicious traffic. This can lead to false detections, so the proposed method learns patterns of unhelpful packets to improve classification. A new training dataset for Generative Adversarial Network (GAN) is created using misclassified data from an original training dataset by the LSTM-DNN model. The GAN can determine if a packet can be classified accurately and cancels detection if not. The proposed algorithm can accurately perform network intrusion detection in real time without session termination or delay time. Experiments confirm early detection before session end while maintaining similar detection performance to existing methods.

Title : Adversarial Examples: Attacks and Defenses for Deep Learning

Author: Xiaoyong Yuan , Pan He, Qile Zhu

Year : 2019

With rapid progress and significant successes in a wide spectrum of applications, deep learning is being applied in many safety-critical environments. However, deep neural networks (DNNs) have been recently found vulnerable to well-designed input samples called adversarial examples. Adversarial perturbations are imperceptible to human but can easily fool DNNs in the testing/deploying stage. The vulnerability to adversarial examples becomes one of the major risks for applying DNNs in safety-critical environments. Therefore, attacks and defenses on adversarial examples draw great attention. In this paper, we review recent findings on adversarial examples for DNNs, summarize the methods for generating adversarial examples, and propose a taxonomy of these methods. Under the taxonomy, applications for adversarial examples are investigated. We further elaborate on countermeasures for adversarial examples. In addition, three major challenges in adversarial examples and the potential solutions are discussed

Title : Detecting Zero-Day Intrusion Attacks Using Semi-Supervised Machine Learning Approaches

Author: Mbona, Jan H. P. Eloff

Year : 2022

Zero-day attacks are a frequent cybersecurity threat that can exploit network system vulnerabilities for prolonged periods. Network traffic analysis (NTA) plays a crucial role in supporting machine learning (ML) based network intrusion detection systems (NIDS). However, most existing ML models for NIDS employ redundant features that negatively impact performance. Benford's law is a viable technique that can effectively identify significant network features indicative of anomalous behavior and detect zero-day attacks. Semi-supervised ML approaches are effective for detecting zero-day attacks if significant features are optimally chosen. Experimental results demonstrate that one-class support vector machines achieved the best results for detecting zero-day network attacks (Matthews correlation coefficient of 74% and F1 score of 85%).

CHAPTER 3

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Enhancing Network Intrusion Detection Systems (NIDS) with supervised Machine Learning (ML) is tough. ML-NIDS must be trained and evaluated, operations requiring data where benign and malicious samples are clearly labelled. Such labels demand costly expert knowledge, resulting in a lack of real deployments, as well as on papers always relying on the same outdated data. The situation improved recently, as some efforts disclosed their labelled datasets. However, most past works used such datasets just as a ‘yet another’ test bed, overlooking the added potential provided by such availability. Despite many successes, the integration of supervised Machine Learning (ML) methods in Network Intrusion Detection Systems (NIDS) is still at an early stage. This is due to the difficulty in obtaining comprehensive sets of labelled data for training and evaluating an ML-NIDS. The recent release of labelled datasets for ML-NIDS was appreciated by the research community; however, few works noticed the opportunity that such availability provides to the state-of-the-art.

3.1.1 Disadvantages of Existing System:

1. They are not predicting the classification of attack types.
2. They are not mentioning the accuracy.
3. They are using machine learning technique only for analyzing purpose.

3.2 PROPOSED SYSTEM

The proposed model is to build a machine learning model for anomaly detection. Anomaly detection is an important technique for recognizing fraud activities, suspicious activities, network intrusion, and other abnormal events that may have great significance but are difficult to detect. The machine learning model is built by applying proper data science techniques like variable identification which is the dependent and independent variables. Then the pre-processing and visualization of the data is done. The model is build based on the previous dataset where the algorithm learn data and get trained different algorithms are used for better comparisons. The performance metrics are calculated and compared.

3.2.1 Advantages of Proposed System:

1. We are implementing the machine learning algorithm for classification purpose.
2. More than two machine learning algorithms are used for comparison of getting the best accuracy.
3. Deployment is done for getting result.

3.3 DEVELOPMENT ENVIRONMENT

1. Software Requirements:

Operating System : Windows 10 or later

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Intel i3

Hard disk : minimum 80 GB

RAM : minimum 2 GB

CHAPTER 4

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 UML DIAGRAMS

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

4.1.1 Use case diagram

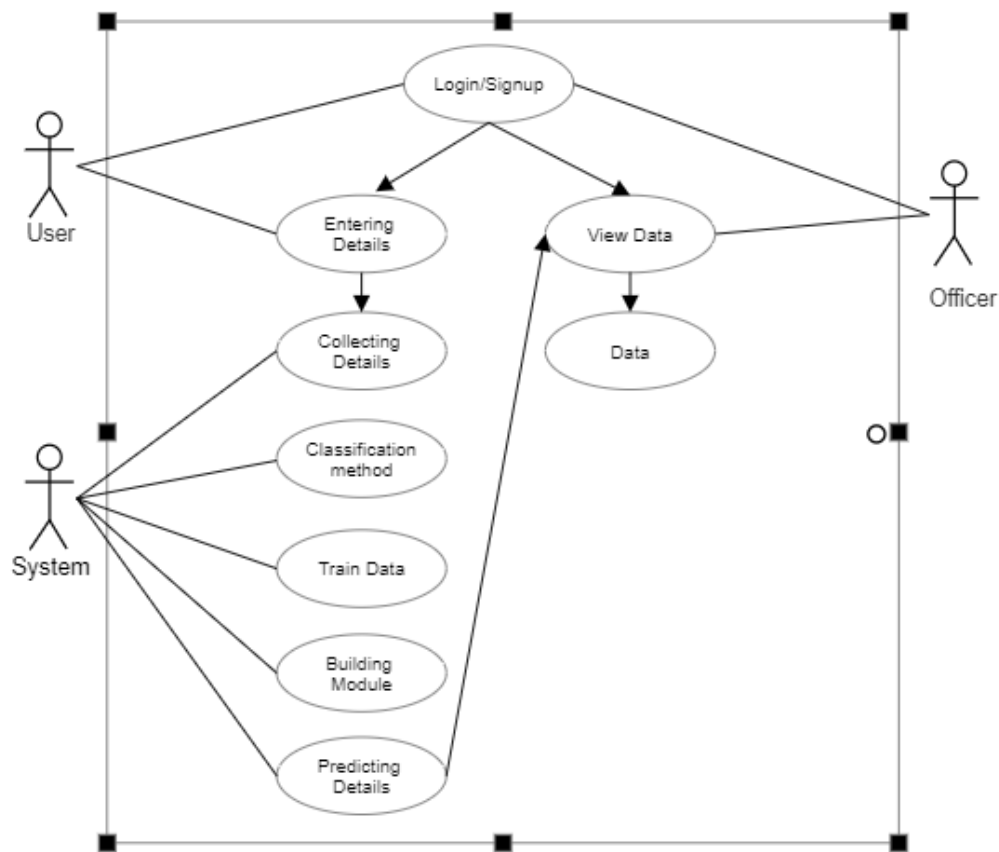


Fig 4.1.1 Use case diagram for Intrusion Detection Prediction using Data Science Technique

4.1.2 Class diagram

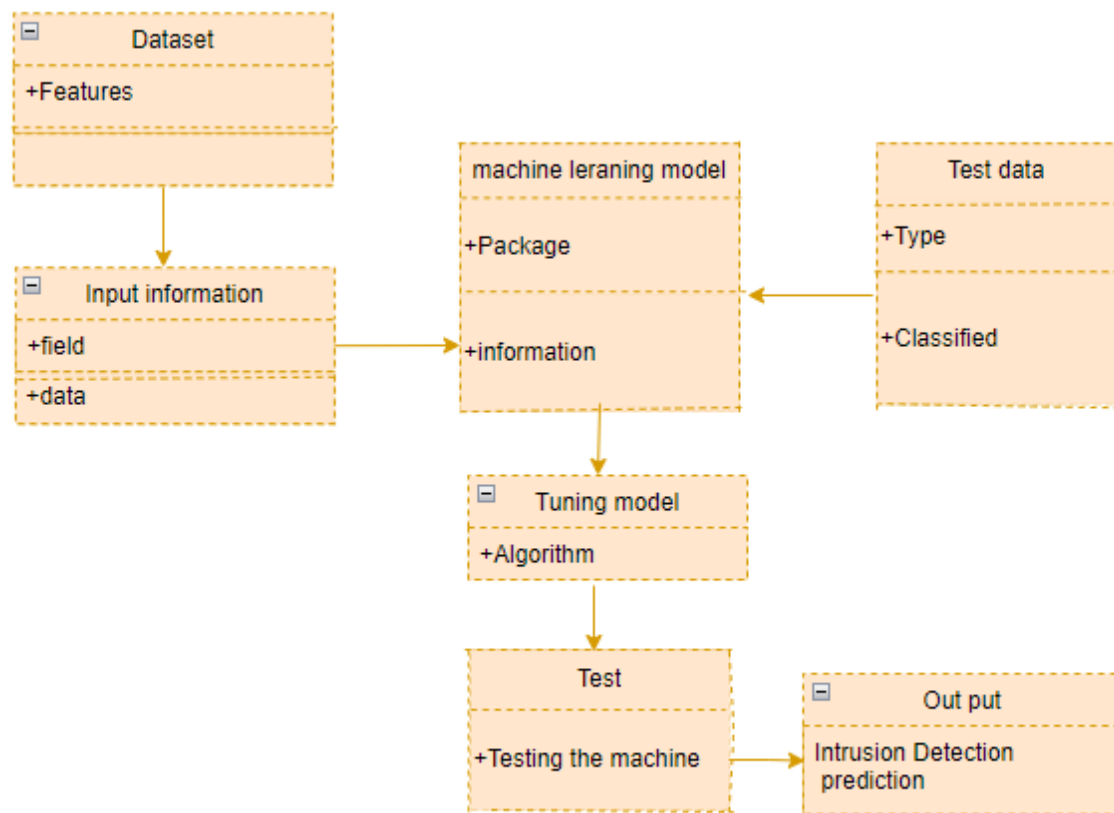


Fig 4.1.2 Class diagram for Intrusion Detection Prediction using Data Science Technique

4.1.3 Sequence diagram

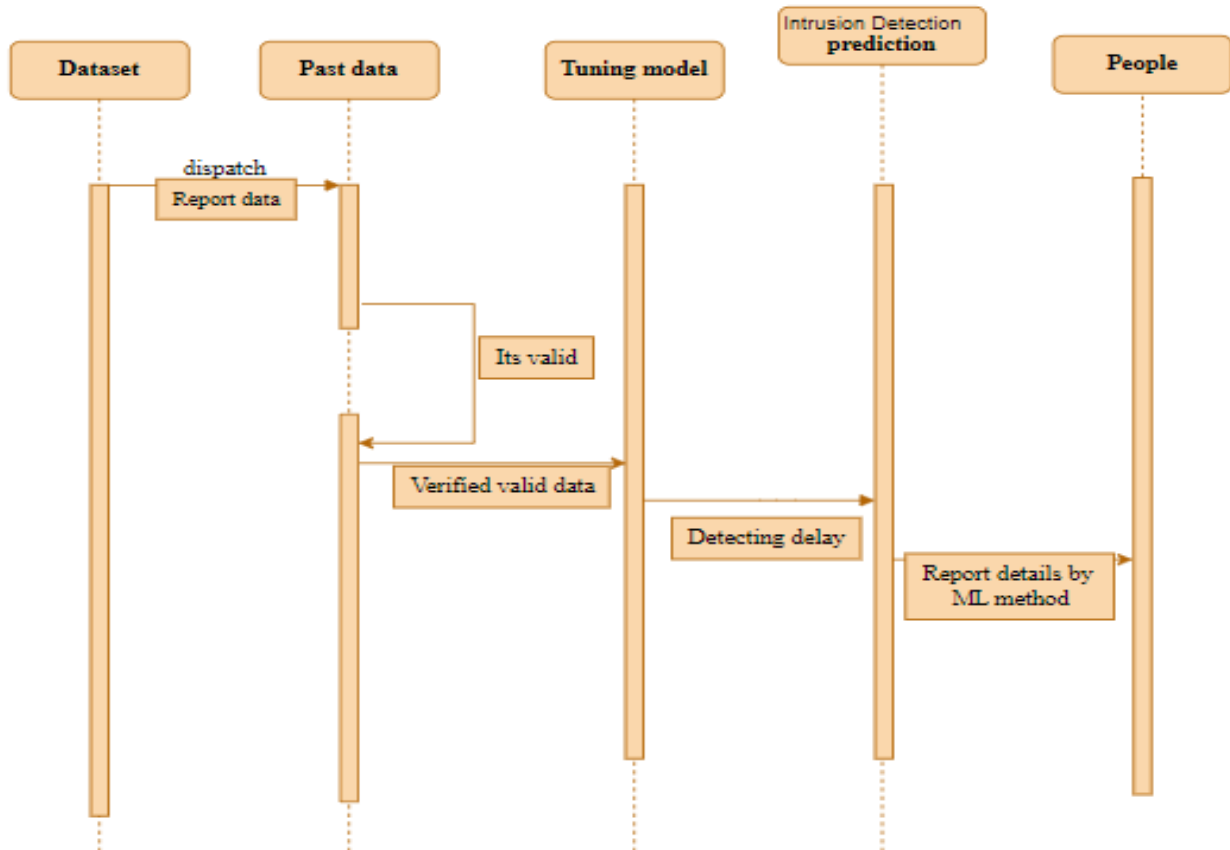


Fig 4.1.3 Sequence diagram for Intrusion Detection Prediction using Data Science Technique

4.1.4 Activity diagram

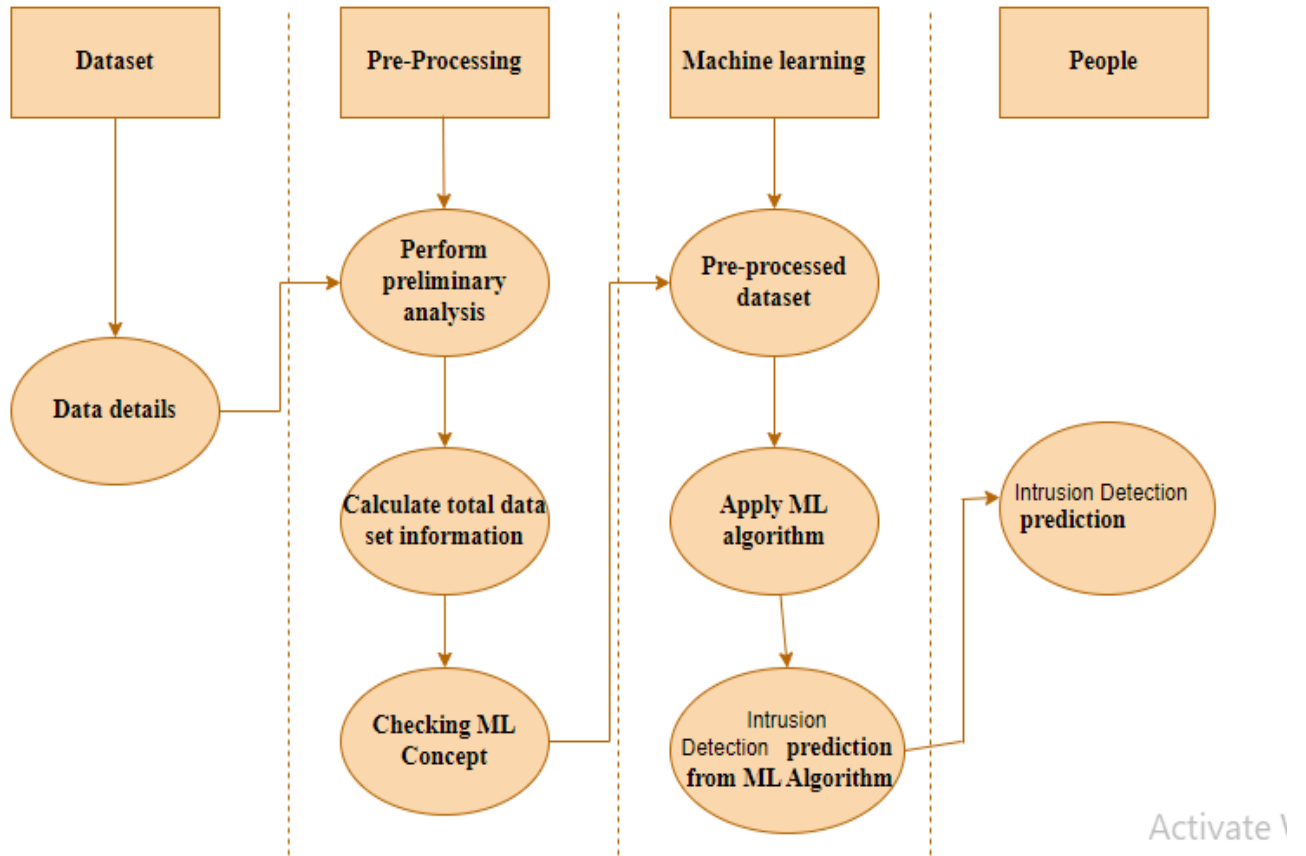


Fig 4.1.4 Activity diagram for Intrusion Detection Prediction using Data Science Technique

4.2 Entity Relationship Diagram (ERD):

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

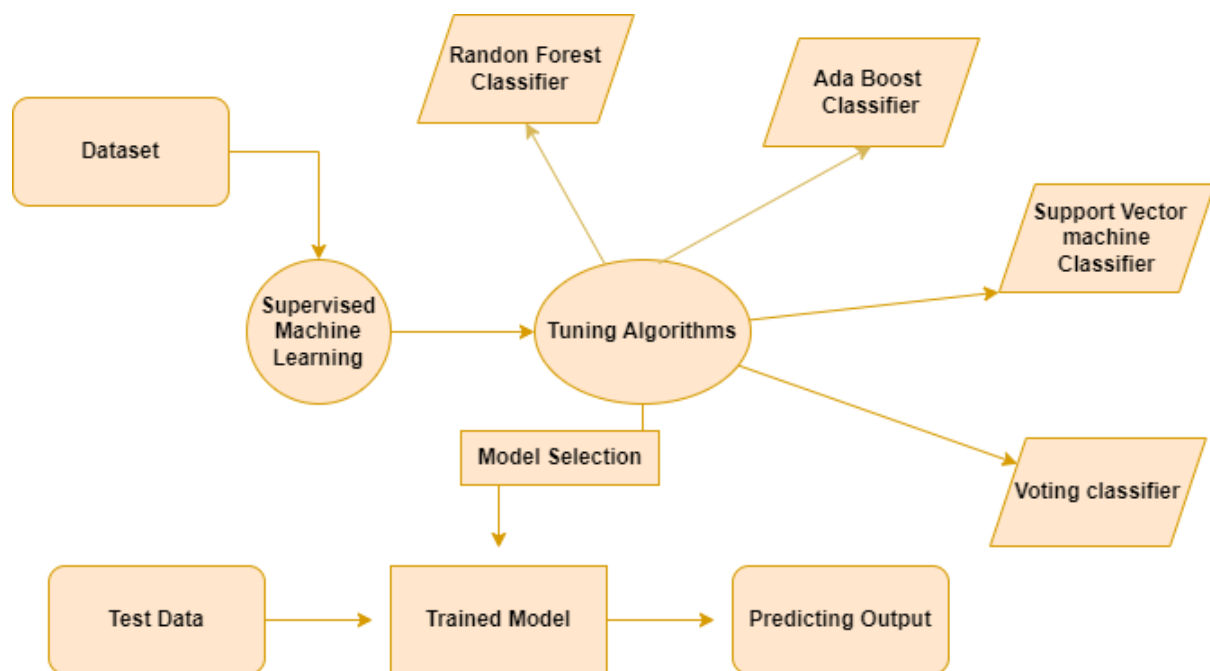


Fig 4.2 Entity Relationship Diagram for Intrusion Detection Prediction using Data Science Technique

4.3 Work flow Diagram

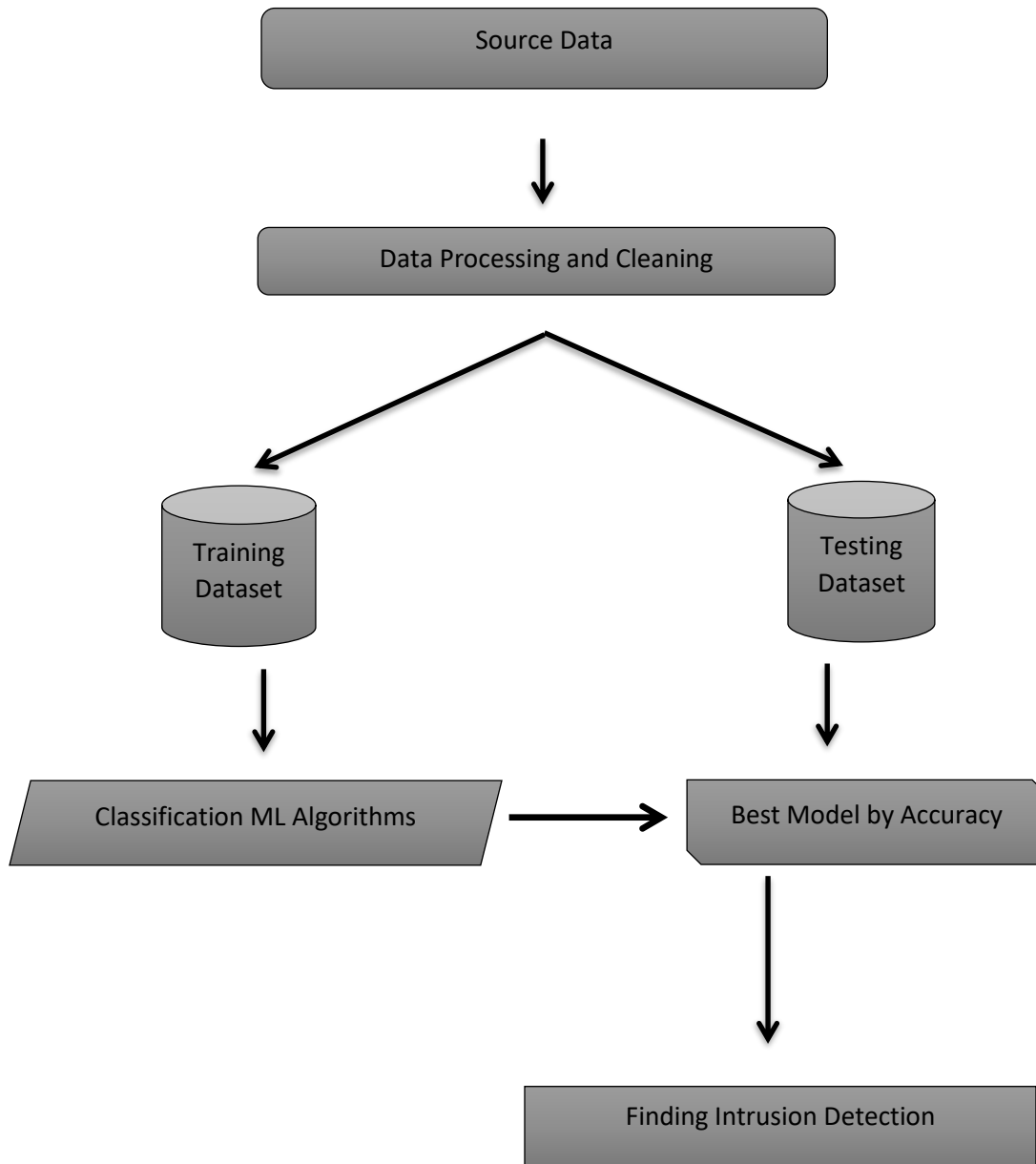


Fig 4.3 Work flow diagram for Intrusion Detection Prediction using Data Science Technique

CHAPTER 5

SYSTEM

ARCHITECTURE

5. SYSTEM ARCHITECTURE

5.1 Architecture Diagram:

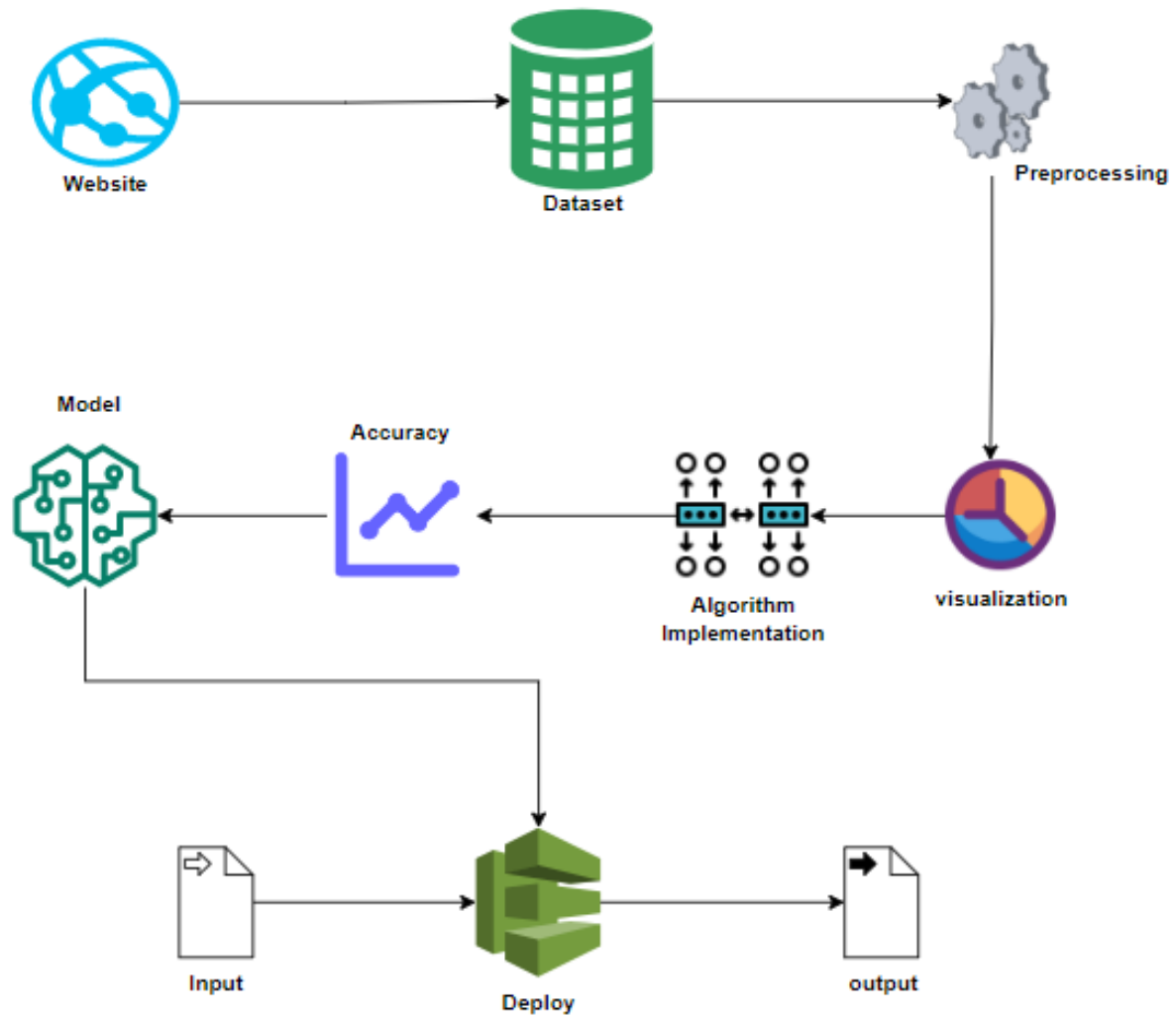


Fig 5.1 Architecture diagram for Intrusion Detection Prediction using Data Science Technique

5.2 MODULE DESCRIPTION:

5.2.1 Algorithm implementation:

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

Performance Metrics to calculate:

False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

$$\text{True Positive Rate(TPR)} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{False Positive rate(FPR)} = \text{FP} / (\text{FP} + \text{TN})$$

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

General Formula:

$$F\text{- Measure} = 2TP / (2TP + FP + FN)$$

F1-Score Formula:

$$F1 \text{ Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

The below 4 different algorithms are compared:

- SVM
- RandomForest
- Voting
- AdaBoost

5.2.2 SVM:

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.

SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. A separator between the categories is found, then the data are transformed in such a way that the separator could be drawn as a hyperplane.

SVMs are used in applications like handwriting recognition, intrusion detection, face detection, email classification, gene classification, and in web pages.

This is one of the reasons we use SVMs in machine learning. It can handle both classification and regression on linear and non-linear data.

```
s = SVC()

s.fit(X_train,y_train)

predictS = s.predict(X_test)
```

```
print("")
print("Accuracy Result of Support Vector Machine is:",accuracy.mean() * 100)
svc=accuracy.mean() * 100
```

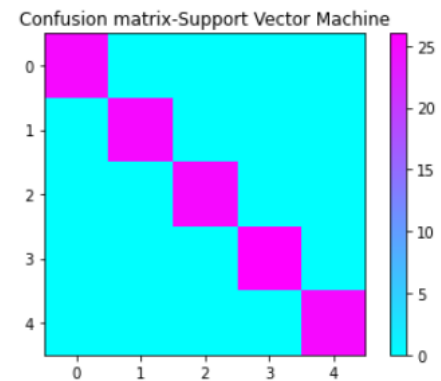


Fig 5.2.1 Support Vector Machine

MODULE DIAGRAM

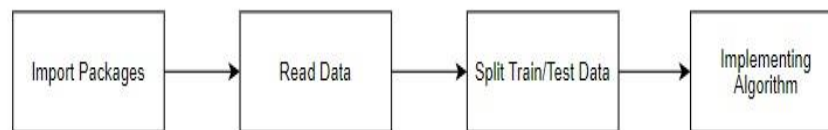


Fig 5.2.2 Module diagram for Implementing Support Vector Machine Algorithm

GIVEN INPUT EXPECTED OUTPUT

Input : Data

Output : Getting accuracy

5.2.3 Adaboost Classifier:

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem. The most suited and therefore most common algorithm used with AdaBoost are decision trees with one level. How does the AdaBoost algorithm work explain?

It works on the principle of learners growing sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones. The AdaBoost algorithm works on the same principle as boosting with a slight difference.

```
ADC = AdaBoostClassifier()
ADC.fit(X_train,y_train)
predictRF = ADC.predict(X_test)

accuracy = cross_val_score(ADC, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)

print("")
print("Accuracy Result of AdaBoostClassifier is:",accuracy.mean() * 100)
adc=accuracy.mean() * 100
```

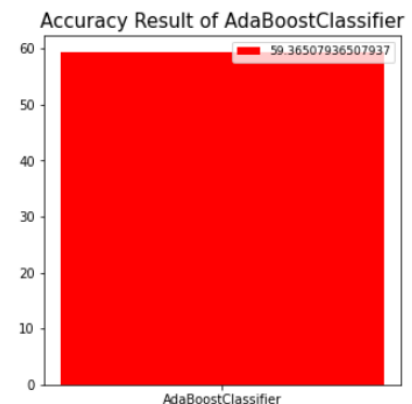


Fig 5.2.3 Adaboost Classifier

MODULE DIAGRAM

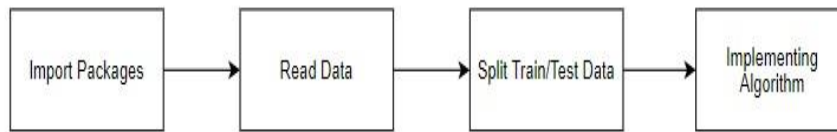


Fig 5.2.4 Module diagram for Implementing Adaboost Classifier Algorithm

GIVEN INPUT EXPECTED OUTPUT

Input : Data

Output : Getting accuracy

5.2.4 Random Forest Classifier:

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.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

```
In [17]: import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["y_predict"] = y_predict
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["y_predict"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
plt.show()
```

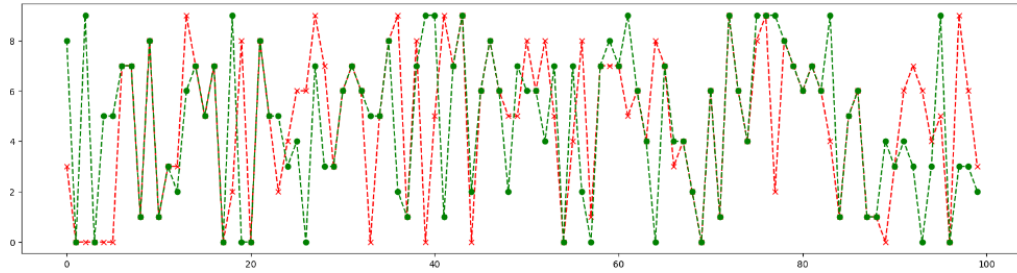


Fig 5.2.5 Random Forest Classifier

MODULE DIAGRAM

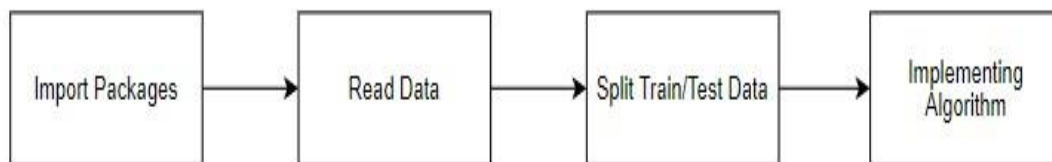


Fig 5.2.6 Module diagram for Implementing Random Forest Classifier Algorithm

GIVEN INPUT EXPECTED OUTPUT

Input : Data

Output : Getting accuracy

5.2.5 Voting Classifier:

A 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.

It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class.

Voting Classifier supports two types of voting's.

1. **Hard Voting:** In hard voting, the predicted output class is a class with the highest majority of votes i.e. the class which had the highest probability of being predicted by each of the classifiers. Suppose three classifiers predicted the output class (A, A, B), so here the majority predicted A as output. Hence A will be the final prediction.
2. **Soft Voting:** In soft voting, the output class is the prediction based on the average of probability given to that class. Suppose given some input to three models, the prediction probability for class A = (0.30, 0.47, 0.53) and B = (0.20, 0.32, 0.40). So, the average for class A is 0.4333 and B is 0.3067, the winner is clearly class A because it had the highest probability averaged by each classifier.

```
In [17]: import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["y_predict"] = y_predict
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["y_predict"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
plt.show()
```

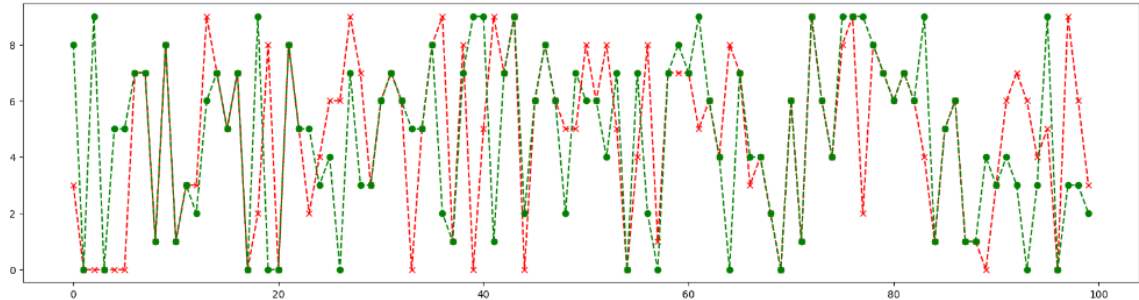


Fig 5.2.7 Voting Classifier

MODULE DIAGRAM

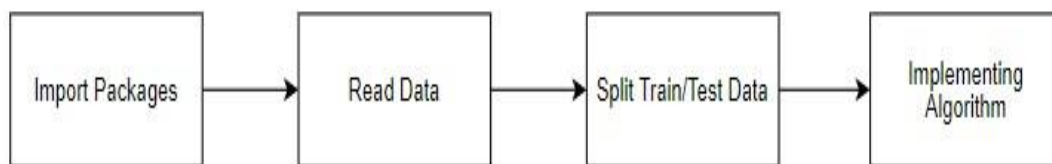


Fig 5.2.8 Module diagram for Implementing Voting Classifier Algorithm

GIVEN INPUT EXPECTED OUTPUT

Input : Data

Output : Getting accuracy

CHAPTER 6

SYSTEM

IMPLEMENTATION

6. SYSTEM IMPLEMENTATION

6.1 Data Pre-processing:

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

- import libraries for access and functional purpose and read the given dataset
- General Properties of Analyzing the given dataset
- Display the given dataset in the form of data frame
- show columns
- shape of the data frame
- To describe the data frame
- Checking data type and information about dataset
- Checking for duplicate data
- Checking Missing values of data frame
- Checking unique values of data frame
- Checking count values of data frame
- Rename and drop the given data frame
- To specify the type of values
- To create extra columns

In [26]:	df.corr()											
Out[26]:	S.No	Crime	Gender	Age	Income	Job	Maritalstatus	Education	Harm	Attack	AttackMethod	
	S.No	1.000000	0.268073	-0.107670	-0.073293	-0.034236	-0.029119	0.088998	-0.050873	0.070474	0.177265	0.056369
	Crime	0.268073	1.000000	-0.237805	-0.008191	-0.063181	-0.102847	0.050194	-0.040659	-0.231643	0.051459	0.330578
	Gender	-0.107670	-0.237805	1.000000	0.142829	0.032175	0.176279	-0.284636	0.057562	0.080345	0.013549	-0.049627
	Age	-0.073293	-0.008191	0.142829	1.000000	0.110287	-0.108884	-0.450486	0.042025	0.052463	-0.014358	0.082491
	Income	-0.034236	-0.063181	0.032175	0.110287	1.000000	-0.092378	-0.108141	-0.162012	0.001779	-0.049182	-0.005811
	Job	-0.029119	-0.102847	0.176279	-0.108884	-0.092378	1.000000	0.086361	0.239153	-0.029589	-0.024657	0.004522
	Maritalstatus	0.088998	0.050194	-0.284636	-0.450486	-0.108141	0.086361	1.000000	0.012036	-0.043787	0.005369	-0.082279
	Education	-0.050873	-0.040659	0.057562	0.042025	-0.162012	0.239153	0.012036	1.000000	-0.008143	-0.034823	0.027717
	Harm	0.070474	-0.231643	0.080345	0.052463	0.001779	-0.029589	-0.043787	-0.008143	1.000000	0.304497	-0.166878
	Attack	0.177265	0.051459	0.013549	-0.014358	-0.049182	-0.024657	0.005369	-0.034823	0.304497	1.000000	0.045194
	AttackMethod	0.056369	0.330578	-0.049627	0.082491	-0.005811	0.004522	-0.082279	0.027717	-0.166878	0.045194	1.000000

In [23]:	df.head()											
Out[23]:	S.No	Crime	Gender	Age	Income	Job	Maritalstatus	Education	Harm	Attack	AttackMethod	
	0	0	Misuse of Debit Cards or Credit Cards	Male	Between 38 and 50	Middle	Technical	Married	Primary Education	Fraud	Hacking Social Media Accounts	Creating a Fake Shopping Site
	1	1	Misuse of Debit Cards or Credit Cards	Female	27 and Under	Low	Student	Single	High School	Internet Shopping Out of Knowledge	Hacking Social Media Accounts	Social Engineering
	2	2	Misuse of Debit Cards or Credit Cards	Male	Between 38 and 50	Low	Retired	Married	Primary Education	Internet Shopping Out of Knowledge	Hacking Social Media Accounts	Social Engineering
	3	3	Misuse of Debit Cards or Credit Cards	Male	Between 38 and 50	Low	Technical	Married	High School	Internet Shopping by Introducing Himself as a B...	Hacking Social Media Accounts	Social Engineering
	4	4	Misuse of Debit Cards or Credit Cards	Male	Between 38 and 50	Low	Others	Married	High School	Internet Shopping Out of Knowledge	Hacking Social Media Accounts	Social Engineering

Fig 6.1 Data Pre-processing

MODULE DIAGRAM

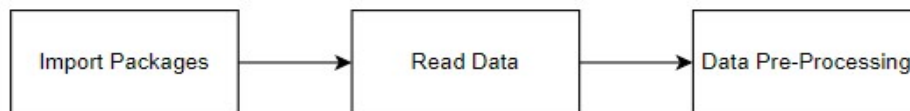


Fig 6.2 Module diagram for Data Pre-processing

GIVEN INPUT EXPECTED OUTPUT

Input : Data

Output : Removing noisy data

6.2 Data visualization:

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- How to chart time series data with line plots and categorical quantities with bar charts.
- How to summarize data distributions with histograms and box plots.

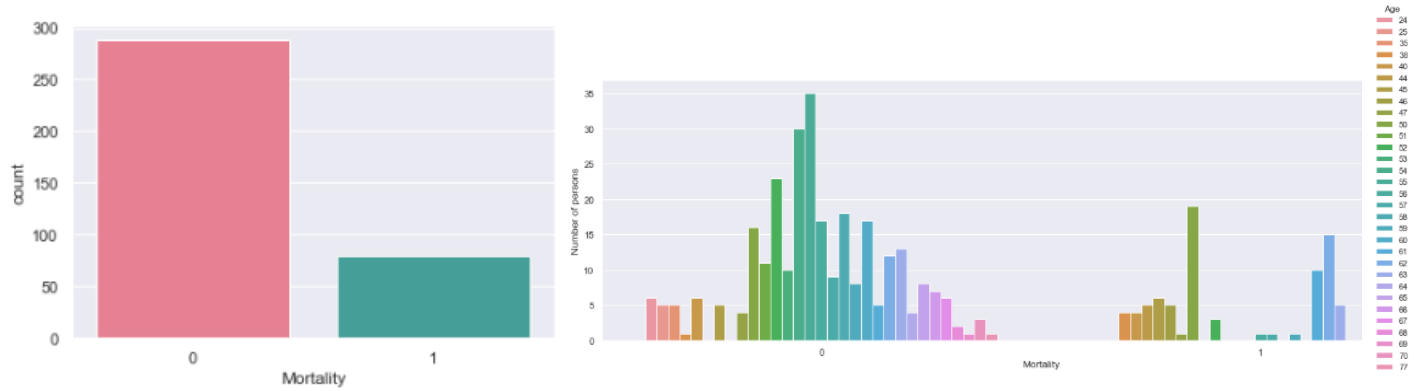


Fig 6.3 Data visualization

MODULE DIAGRAM

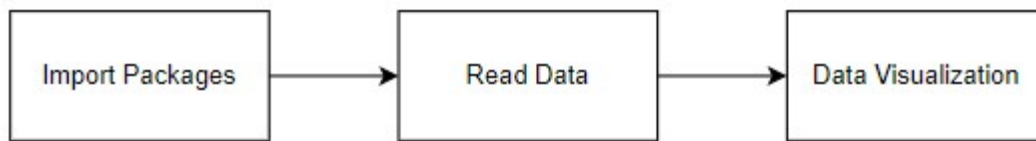


Fig 6.4 Module diagram for Data visualization

GIVEN INPUT EXPECTED OUTPUT

Input : Data

Output : Visualized data

CHAPTER 7

SYSTEM TESTING

7.1 TEST CASES

TEST REPORT: 01

USECASE : User/Admin Signup

TEST CASE ID	TESTCASE/ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	PASS/ FAIL
1	Enter a Name & Password in the Text Box	User Created Successful	User Created Successful	Pass
2	Enter a Name & Password in the Text Box	Enter Valid Input	Enter Valid Input	Pass

Table-7.1.1 Test Case for User/Admin Signup

TEST REPORT: 02

USECASE : User/Admin Login

TEST CASE ID	TESTCASE/ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	PASS/ FAIL
1	Enter a Name & Password in the Text Box	Login Successfully	Login Successfully	Pass
2	Enter a Name & Password in the Text Box	Enter Valid Input	Enter Valid Input	Pass

Table-7.1.2 Test Case for User/Admin Login

TEST REPORT: 03**USECASE** : Getting Inputs from users/Admin

TEST CASE ID	TESTCASE/ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	PASS/ FAIL
1	Selecting Type of Crime from Dropdown list Box	Option Selected	Option Selected	Pass
2	Selecting Type of Crime from Dropdown list Box	Select Valid Input	Select Valid Input	Pass
3	Clicking Submit Button	Submitted	Submitted	Pass
4	Clicking User Logout Button	Logout Successfully	Logout Successfully	Pass
5	Show the Latest Report	Data Displayed	Data Displayed	Pass
6	Show All Report	Data Displayed	Data Displayed	Pass
7	Clicking User Logout Button	Logout Successfully	Logout Successfully	Pass

Table-7.1.3 Test Case for Getting Inputs from users/Admin

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENT

8.1 CONCLUSION

In conclusion, intrusion detection prediction using data science techniques is a crucial aspect of cybersecurity. Anomaly detection using machine learning models can help organizations identify and prevent potential security breaches, network intrusions, and other abnormal activities that may pose a significant risk to their critical assets and data. The proposed methodology involves various steps, including data collection, preprocessing, exploratory data analysis, feature selection, model selection, training, evaluation, optimization, and deployment. The successful implementation of the proposed model can contribute to the development of reliable and efficient intrusion detection systems that can help organizations protect their assets from potential threats. The continuous monitoring and updating of the model can help organizations stay ahead of the evolving threat landscape and ensure the security of their networks and systems. The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set of higher accuracy score algorithm will be find out. The founded one is used in the application which can help to find the type of intrusions.

8.2 FUTURE ENHANCEMENT

One potential future enhancement for project is to deploy the system in the cloud. Cloud deployment has several advantages, including scalability, reliability, and accessibility. By deploying the system in the cloud, users can access it from anywhere with an internet connection, and the system can easily scale up or down to handle varying levels of traffic.

Another potential future enhancement for the project is to optimize the system to implement in an IoT system. This would involve modifying the system to work with IoT devices and sensors, allowing for real-time monitoring and detection of network intrusions.

APPENDICES

A.1 Coding:

```
#import library packages
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
# Load given dataset
data = pd.read_csv("data.csv")
df = data.dropna()
del df['S.No']
df.columns
df.head()

from sklearn.preprocessing import LabelEncoder
var_mod = ['Crime', 'Gender', 'Age', 'Income', 'Job', 'Maritalstatus', 'Education',
'Harm', 'Attack']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
df.head()

#preprocessing, split test and dataset, split response variable
X = df.drop(labels='AttackMethod', axis=1)
#Response variable
y = df.loc[:, 'AttackMethod']
#Splitting for train and test

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=42, stratify=y)
```

```

print("Number of training dataset: ", len(X_train))
print("Number of test dataset: ", len(X_test))
print("Total number of dataset: ", len(X_train)+len(X_test))

Implementing SVC Algo
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, plot_confusion_matrix

Training

svc = SVC()
svc.fit(X_train,y_train)
predicted = svc.predict(X_test)

Finding Accuracy

accuracy = accuracy_score(y_test,predicted)
print('Accuracy of Support Vector Classifier',accuracy*100)

Finding Classification Report

cr = classification_report(y_test,predicted)
print('Classification report\n\n',cr)

Finding Confusion matrix

cm = confusion_matrix(y_test,predicted)
print('Confusion matrix\n\n',cm)
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10,10))
plot_confusion_matrix(svc, X_test, y_test, ax=ax)
plt.title('Confusion matrix of SVC')
plt.show()

df2 = pd.DataFrame()

```

```

df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
plt.show()

#import library packages
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

# Load given dataset
data = pd.read_csv("data.csv")
df = data.dropna()
del df['S.No']
df.columns
df.head()

from sklearn.preprocessing import LabelEncoder
var_mod = ['Crime', 'Gender', 'Age', 'Income', 'Job', 'Maritalstatus', 'Education',
'Harm', 'Attack']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
df.head()

#preprocessing, split test and dataset, split response variable
X = df.drop(labels='AttackMethod', axis=1)

#Response variable

```



```

y = df.loc[:, 'AttackMethod']
#Splitting for train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=42, stratify=y)
print("Number of training dataset: ", len(X_train))
print("Number of test dataset: ", len(X_test))
print("Total number of dataset: ", len(X_train)+len(X_test))

Implementing RF Algo
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, plot_confusion_matrix

Training

rf = RandomForestClassifier()
rf.fit(X_train,y_train)
predicted = rf.predict(X_test)

Finding Accuracy

accuracy = accuracy_score(y_test,predicted)
print('Accuracy of Random Forest Classifier',accuracy*100)

Finding Classification Report

cr = classification_report(y_test,predicted)
print('Classification report\n\n',cr)

Finding Confusion matrix

cm = confusion_matrix(y_test,predicted)
print('Confusion matrix\n\n',cm)
import matplotlib.pyplot as plt

```

```

fig, ax = plt.subplots(figsize=(10,10))
plot_confusion_matrix(rf, X_test, y_test, ax=ax)
plt.title('Confusion matrix of Random Forest')
plt.show()

df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
plt.show()

from joblib import dump
dump(rf, 'RF.pkl')

from sklearn.preprocessing import LabelEncoder
var_mod = ['Crime', 'Gender', 'Age', 'Income', 'Job', 'Maritalstatus', 'Education',
'Harm', 'Attack']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
df.head()

#preprocessing, split test and dataset, split response variable
X = df.drop(labels='AttackMethod', axis=1)
#Response variable
y = df.loc[:, 'AttackMethod']
#Splitting for train and test
from sklearn.model_selection import train_test_split

```

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=42, stratify=y)
print("Number of training dataset: ", len(X_train))
print("Number of test dataset: ", len(X_test))
print("Total number of dataset: ", len(X_train)+len(X_test))

```

Implementing AdaBoost Algo

```

from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, plot_confusion_matrix

```

Training

```

ab = AdaBoostClassifier()
ab.fit(X_train,y_train)
predicted = ab.predict(X_test)

```

Finding Accuracy

```

accuracy = accuracy_score(y_test,predicted)
print('Accuracy of AdaBoost Classifier',accuracy*100)

```

Finding Classification Report

```

cr = classification_report(y_test,predicted)
print('Classification report\n\n',cr)

```

Finding Confusion matrix

```

cm = confusion_matrix(y_test,predicted)
print('Confusion matrix\n\n',cm)
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10,10))
plot_confusion_matrix(ab, X_test, y_test, ax=ax)
plt.title('Confusion matrix of AdaBoost')

```

```

plt.show()
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
plt.show()

from django.shortcuts import render, redirect
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth import login, authenticate, logout
from . import forms
from . import models
import numpy as np
import joblib

model = joblib.load('C:/Users/SPIRO15/Desktop/ITPML01
(INTRUSION)/Deploy/new/RF.pkl')

# Create your views here.
def home_view(request):
    if request.method == "POST":
        username = request.POST['username']
        #print(username)
        password = request.POST['password']

```

```

#print(password)
name = request.POST['user']
if name == "user":
    user = authenticate(request, username=username, password=password)
    #print(user)
    if user is not None:
        login(request, user)
        return render(request, 'new/index.html')
    else:
        msg = 'Invalid Credentials'
        form = AuthenticationForm(request.POST)
        return render(request, 'new/user_login.html', {'form': form, 'msg': msg})
else:
    user = authenticate(request, username=username, password=password)
    #print(user)
    if user is not None:
        login(request, user)
        model = models.UserPredictDataModel.objects.latest('id')
        form = forms.DoctorFeedForm(request.POST)
        #print(model)
        return render(request, 'new/last.html', {'model':model,'form':form})
    else:
        msg = 'Invalid Credentials'
        form = AuthenticationForm(request.POST)
        return render(request, 'new/user_login.html', {'form': form, 'msg': msg})
else:
    form = AuthenticationForm()

```

```

    return render(request, 'new/user_login.html', {'form': form})
def doctor_login(request):
    form = AuthenticationForm()
    return render(request, 'new/login.html', {'form': form})
def user_register(request):
    if request.method == "POST":
        form = UserCreationForm(request.POST)
        if form.is_valid():
            #print('saving')
            form.save()
            return render(request, 'new/user_signup.html', {'msg':"Successfully
registered",'form':form})
        else:
            form = UserCreationForm()
            return render(request, 'new/user_signup.html',{'form':form})
def doctor_register(request):
    if request.method == "POST":
        form = UserCreationForm(request.POST)
        if form.is_valid():
            #print('saving')
            form.save()
            return render(request, 'new/user_signup.html', {'msg':"Successfully
registered",'form':form})
        else:
            form = UserCreationForm()
            return render(request, 'new/signup.html',{'form':form})
def predict_view(request):

```

```

if request.method == 'POST':
    print('IF')
    fieldss = ['Crime', 'Gender', 'Age', 'Income', 'Job', 'Maritalstatus', 'Education',
'Harm', 'Attack']
    form = forms.UserPredictDataForm(request.POST)
    features = []
    for i in fieldss:
        info = request.POST[i]
        features.append(info)
    final_features = [np.array(features)]
    #print(final_features)
    prediction = model.predict(final_features)
    #print(prediction)
    output = prediction[0]
    print(features)
    print(output)
    if form.is_valid():
        print('saving')
        form.save()
    ob = models.UserPredictDataModel.objects.latest('id')
    ob.AttackMethod = output
    ob.save()
    return render(request, 'new/index.html', {'prediction_text':output,'form':form})
else:
    print('ELSE')
    form = forms.UserPredictDataForm(request.POST)
    return render(request, 'new/index.html', {'form':form})

```

```

def doctor_patient_details_view_all(request):
    model = models.UserPredictDataModel.objects.all()
    #print(model)
    return render(request, 'new/all.html', {'model':model})

def doctor_patient_details_view_last(request):
    if request.method == "POST":
        form = forms.DoctorFeedForm(request.POST)
        #print('form',form)
        if form.is_valid():
            form.save()
            model = models.UserPredictDataModel.objects.latest('id')
            #print(model)
            return render(request, 'new/last.html', {'model':model,'msg':'Feedback
sent'})

```


A.2 Sample Screens

In Fig A.2.1, For deploying and run the entire program, just call and run the file `manage.py` in a server.

```
Microsoft Windows [Version 10.0.22624.1537]
(c) Microsoft Corporation. All rights reserved.

(base) C:\Users\tpeni\OneDrive\Desktop\final yr project\CODE\CODE\Deploy>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator DecisionTreeClassifier from version 1.1.2 when using version 1.0.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
```

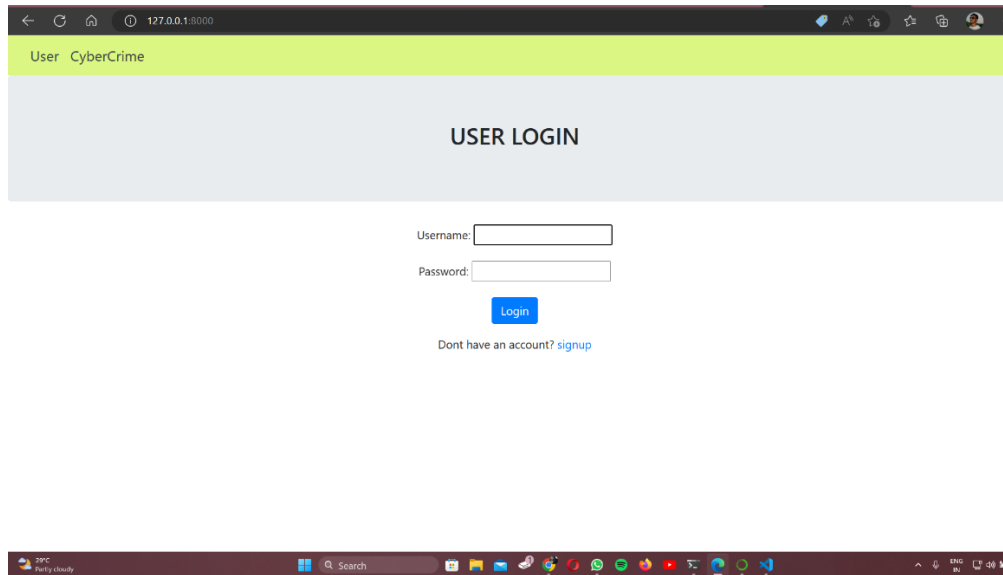
Fig A.2.1 Deploying the Program

In Fig A.2.2, After initializing the server press CTRL and right click the IP Address to run the program in a separate server.

```
https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-limitations
warnings.warn(
System check identified no issues (0 silenced).
April 06, 2023 - 10:46:43
Django version 4.1.7, using settings 'latest.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[06/Apr/2023 10:46:48] "GET / HTTP/1.1" 200 2745
Not Found: /favicon.ico
[06/Apr/2023 10:46:49] "GET /favicon.ico HTTP/1.1" 404 4031
█
```

Fig A.2.2 Initializing the server

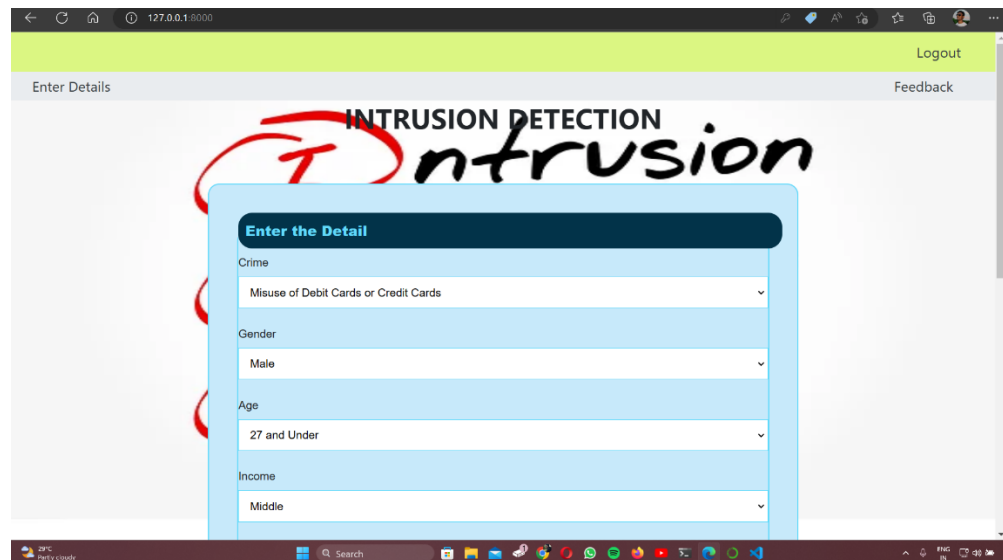
In Fig A.2.3, Enter user Login Details for new user or already existed user in Login Page.



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000'. The page has a light green header with the text 'User CyberCrime'. Below the header, the title 'USER LOGIN' is centered. The login form consists of two input fields: 'Username:' and 'Password:'. A blue 'Login' button is positioned below the password field. At the bottom of the form, there is a link that says 'Dont have an account? [signup](#)'. The browser's taskbar at the bottom shows the Windows Start button, a search bar, and several application icons.

Fig A.2.3 User Login Page

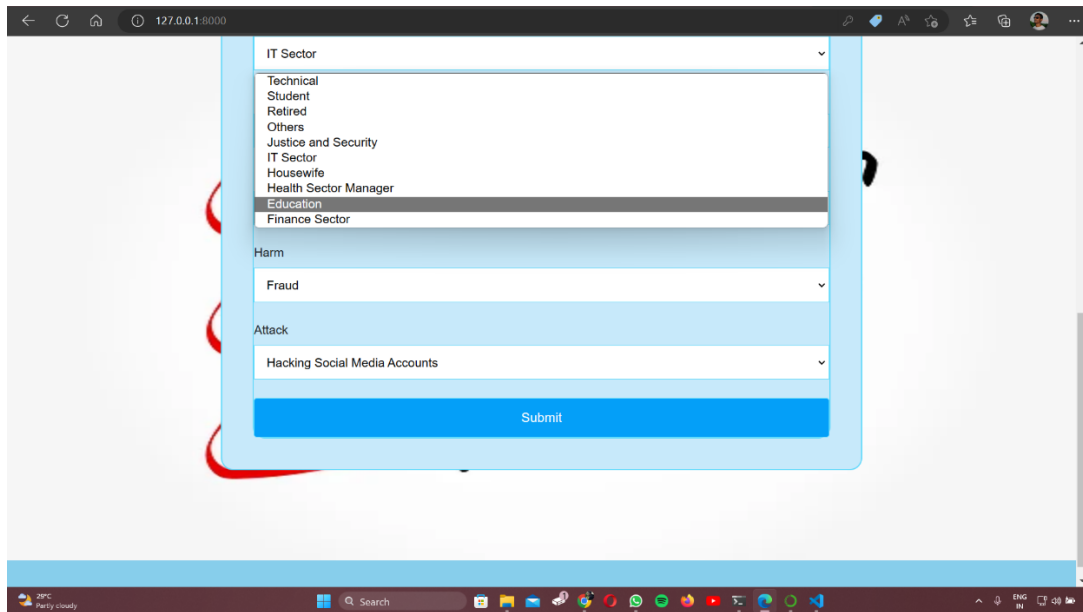
In Fig A.2.4, Enter all the details regarding the experienced attack in the drop-down list box.



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000'. The page has a light green header with the text 'Logout' on the right. Below the header, the title 'Enter Details' is on the left and 'Feedback' is on the right. The main content area features a large, stylized graphic with the text 'INTRUSION DETECTION' and 'Intrusion'. Below the graphic, there is a form titled 'Enter the Detail' with four drop-down menus: 'Crime' (selected: 'Misuse of Debit Cards or Credit Cards'), 'Gender' (selected: 'Male'), 'Age' (selected: '27 and Under'), and 'Income' (selected: 'Middle'). The browser's taskbar at the bottom shows the Windows Start button, a search bar, and several application icons.

Fig A.2.4 User Input Page

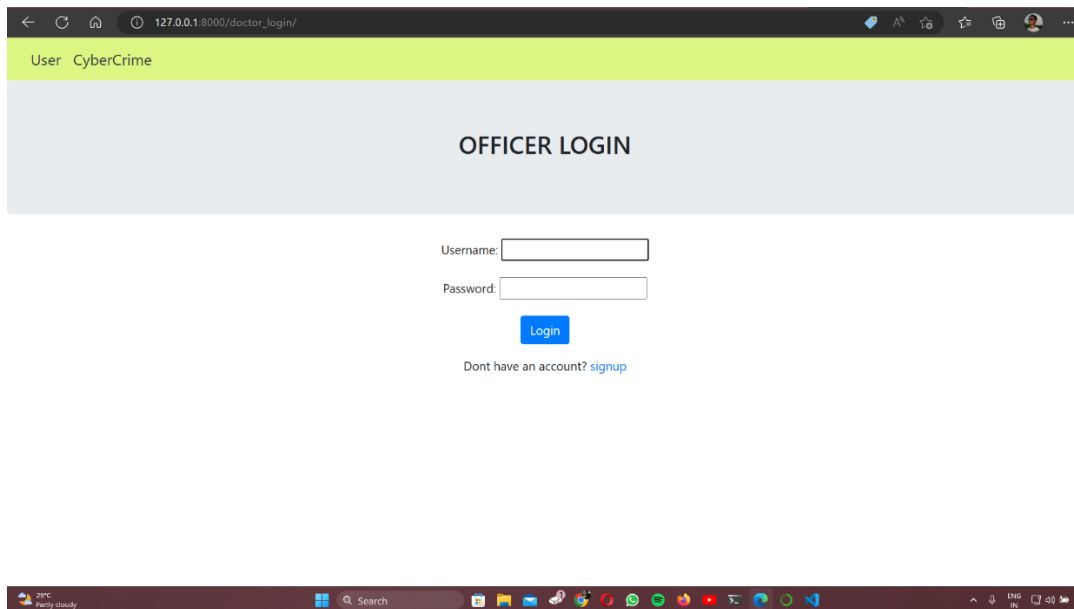
In Fig A.2.5, Now submit all the details.



The screenshot shows a web browser window with a URL bar displaying "127.0.0.1:8000". The main content area contains a form with several dropdown menus and a "Submit" button. The dropdown menus are labeled "IT Sector", "Harm", "Attack", and "Hacking Social Media Accounts". The "IT Sector" dropdown is open, showing a list of options: Technical, Student, Retired, Others, Justice and Security, IT Sector, Housewife, Health Sector Manager, Education, and Finance Sector. The "Harm" dropdown is open, showing "Fraud". The "Attack" dropdown is open, showing "Hacking Social Media Accounts". The "Submit" button is a blue button with white text.

Fig A.2.5 User Detail Submission

In Fig A.2.6, Enter user Login Details for new user or already existed user in cybercrime page



The screenshot shows a web browser window with a URL bar displaying "127.0.0.1:8000/doctor_login/". The main content area has a light green header with the text "User CyberCrime". Below the header, the text "OFFICER LOGIN" is displayed in a large, bold, black font. Underneath, there are two input fields labeled "Username:" and "Password:". Below the "Password:" field is a blue "Login" button. At the bottom, there is a link that says "Dont have an account? [signup](#)".

Fig A.2.6 Officer Login Page

In Fig A.2.7, The Officer can check on all the data submitted.

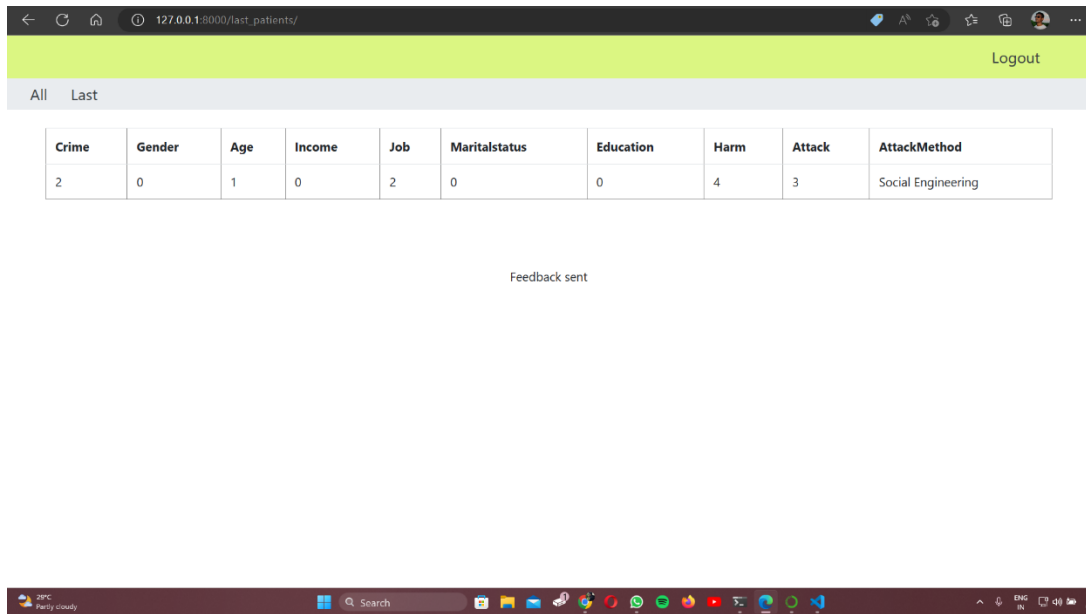


Fig A.2.7 Data submissions

In Fig A.2.8, Send feedback to the user's report.

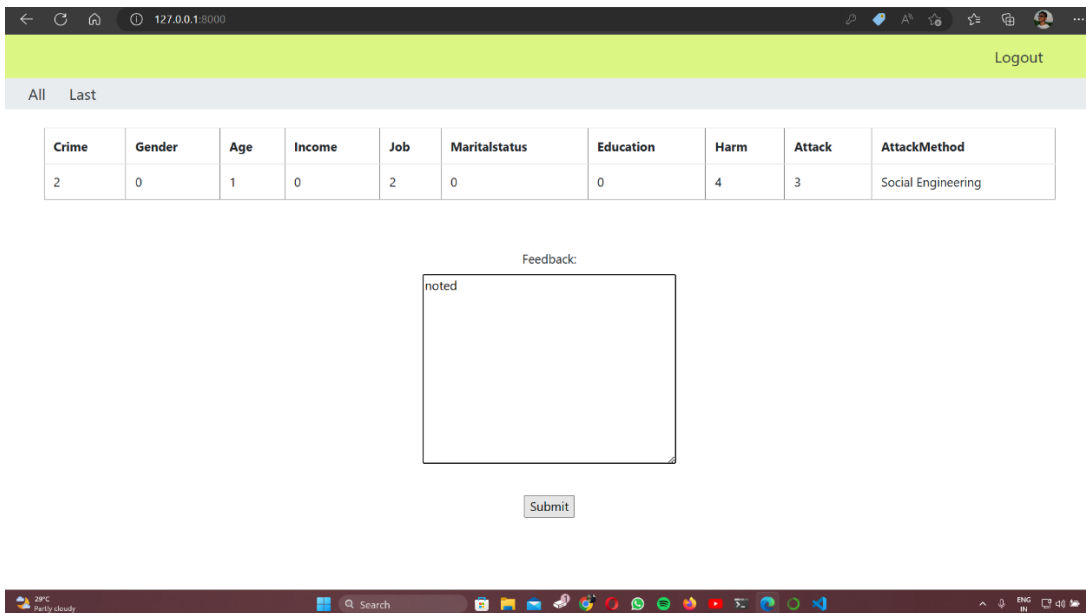


Fig A.2.8 User feedback

REFERENCES

REFERENCES

- [1] Tchakoucht TA, Ezziyyani M. Building a fast intrusion detection system for high-speed-networks: probe and DoS attacks detection. *Procedia Comput Sci.* 2018;127:521–30.
- [2] Zuech R, Khoshgoftaar TM, Wald R. Intrusion detection and big heterogeneous data: a survey. *J Big Data.* 2015;2:3.
- [3] Debar H. An introduction to intrusion-detection systems. In: *Proceedings of Connect*, 2000. 2000.
- [4] Ferhat K, Sevcan A. Big Data: controlling fraud by using machine learning libraries on Spark. *Int J Appl Math Electron Comput.* 2018;6(1):1–5.
- [5] Manzoor MA, Morgan Y. Real-time support vector machine based network intrusion detection system using Apache Storm. In: *IEEE 7th annual information technology, electronics and mobile communication conference (IEMCON)*, 2016. Piscataway: IEEE. 2016; p. 1–5.
- [6] Vimalkumar K, Radhika N. A big data framework for intrusion detection in smart grids using Apache Spark. In: *International conference on advances in computing, communications and informatics (ICACCI)*, 2017. Piscataway: IEEE; 2017. p. 198–204.
- [7] Dahiya P, Srivastava DK. Network intrusion detection in big dataset using Spark. *Procedia Comput Sci.* 2018;132:253–62.
- [8] Vapnik, "The Nature of Statistical Learning Theory", Springer-Verlag, New York, 1995.
- [9] Bremner D, Demaine E, Erickson J, Iacono J, Langerman S, Morin P, Toussaint G (2005). "Output-sensitive algorithms for computing nearest-neighbor decision boundaries". *Discrete and Computational Geometry* 33 (4): 593-604.
- [10] Domingos, Pedro & Michael Pazzani (1997) "On the optimality of the simple Bayesian classifier under zero-one loss". *Machine Learning*, 29:103-137
- [11] Bradley AP. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognit.* 1997;30(7):1145–59.

- [12] Rung-Ching Chen, Kai-Fan Cheng and Chia-Fen Hsieh, "Using Rough Set and Support Vector Machine for Network Intrusion Detection", International Journal of Network Security & Its Applications (IJNSA), Vol 1, No 1, 2009.
- [13] Enache A-C, Sgârciu V. Enhanced intrusion detection system based on bat algorithm-support vector machine. In: 11th international conference on security and cryptography (SECRYPT), 2014. Piscataway: IEEE; 2014. p. 1–6.
- [14] Bhavsar H, Ganatra A. A comparative study of training algorithms for supervised machine learning. Int J Soft Comput Eng (IJSCE). 2012;2(4):2231–307.
- [15] Kulariya M. et al. Performance analysis of network intrusion detection schemes using Apache Spark. In: International conference on communication and signal processing (ICCSP), 2016. Piscataway: IEEE; 2016. p. 1973–1977.
- [16] C. Leckie and R. Kotagiri. A Probabilistic Approach to Network Scan Detection. In Proceedings of the 8th