VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum- 590 018

A Project Report on

"IMPROVING CYBER THREAT DETECTION USING AI"

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

Bachelor of Engineering in Computer Science and Engineering

submitted by

ASHWATH D PADUR [1DB19CS017]
CHETHAN S [1DB19CS035]
G S SUDEEP [1DB19CS049]

Under the guidance of

Prof. Umashankar B S
Vice Principal
Department of CSE

Don Bosco Institute of Technology

Mysore Road, Kumbalgodu, Bengaluru-560074

2022-2023

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

DON BOSCO INSTITUTE OF TECHNOLOGY

Mysore Road, Kumbalgodu, Bengaluru-560074

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project Phase-II entitled **"IMPROVING CYBER THREAT DETECTION USING AI"** is carried out by **Ashwath D Padur [1DB19CS017], Chethan S [1DB19CS035] and G S Sudeep [1DB19CS049]** are bonafide students of **Don Bosco Institute of Technology, Bangalore** in partial fulfillment for the award of the degree of **Bachelor of Engineering in Computer science and Engineering** of **Visvesvaraya Technological University, Belagavi** during the academic year **2022-23**. The project Phase -II report has been approved as it satisfies the academic requirements in respect of the Project Phase-II prescribed for the Bachelor of Engineering Degree.

Prof. Umashankar B S	**Dr. K B ShivaKumar**	**Dr. B S Nagabhushana**
Vice Principal, Dept of CSE	H.O.D., Dept of CSE	Principal, D.B.I.T

<u>External Viva</u>

Name of Examiner	Signature with date
1. ________________	1. ___________________
2. ________________	2. ___________________

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

DON BOSCO INSTITUTE OF TECHNOLOGY

Mysore Road, Kumbalgodu, Bengaluru-560074

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **Ashwath D Padur [1DB19CS017], Chethan S [1DB19CS035]** and **G S Sudeep [1DB19CS49]** students of seventh semester B.E, at the department of Computer Science and Engineering, Don Bosco Institute of Technology, Bengaluru declare that the project phase-II entitled "**Improving Cyber Threat Detection Using AI**" has been carried out by us and submitted in partial fulfillment of the course requirements for the award of degree of Bachelor of Engineering in Computer Science and Engineering discipline of Visvesvaraya Technological University, Belagavi during the academic year **2022-23**. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree.

Place: Bangalore	**Ashwath D Padur [1DB19CS017]**
Date:	**Chethan S [1DB19CS035]**
	G S Sudeep [1DB19CS049]

ACKNOWLEDGEMENT

The satisfaction and euphoria that successful completion of any project is incomplete without the mention of people who made it possible, whose constant guidance and encouragement made our effort fruitful.

First and foremost, we ought to pay our due regards to this institute, which provided us a platform and gave an opportunity to display our skills through the medium of project work. We express our heartfelt thanks to our beloved Principal **Dr. B S NAGABUSHANA** and Vice-Principal **Prof. B S UMASHANKAR,** Don Bosco Institute of Technology, Bangalore for their encouragement all through our graduation life and providing us with the infrastructure.

We express our deep sense of gratitude and thanks to **Dr. K B Shivakumar, Head of the Department, Computer Science and Engineering** for extending his valuable insight and suggestions offered during the course.

We express our acknowledgement to our project coordinator **Dr. Venugeetha Y, Professor, Dept of CSE** for extending their direction and support during the completion of project phase-II.

It is our utmost pleasure to acknowledge the kind help extended by our guide **Prof. Umashankar B S**, **Vice Principal**, Department of Computer Science and Engineering, for guidance and assistance which consequently resulted in getting the project phase II work completed successfully.

Last but not the least I would like to thank teaching and non-teaching staff for their cooperation extended during the completion of the project Phase-II.

Ashwath D Padur [1DB19CS017]

Chethan S [1DB19CS035]

G S Sudeep [1DB19CS049]

ABSTRACT

Security awareness has been a popular topic in the last few years for both information systems researchers and organizations. News broadcasts has brought attention to the increase in cyber-attacks, with these reports noting that a significant number of these breaches have been caused by human error, linked to employee's lack of engagement with their organizations security policies and awareness campaigns. Whilst there is existing research in human factors and the barriers of security behaviors effect on cyber security awareness; in practice we know very little about how employees can be motivated to engage in cyber security awareness programs.

The software's like Pycharm and Jupyter notebook is used to build the project, "Improving Cyber Threat Detection Using AI". Building a project which shows the clear difference between the different AI models by showing their attributes like, accuracy and precision and provide a pictorial representation of these attributes. Further developing a model by integrating two AI models to generate a model which will provide better security.

TABLE OF CONTENTS

LIST OF FIGURES

IMPROVING CYBER THREAT DETECTION USING AI

Chapter 1

INTRODUCTION

Overview

Cyber security is defined as a set of process, human behaviour and systems that help safeguard electronic resources. Cyber security is a fast-evolving field over the last decade. As we rely more and more on technology, the need for better security also increases. Cyber security is the application of technologies, processes, and controls to protect systems, networks, programs, devices and data from cyber-attacks. AI models can be used to train the system to detect malicious threats to the system and improve the security.

Motivation

- Most cybercrime is committed by cybercriminals or hackers who want to make money. However, occasionally cybercrime aims to damage computers or networks for reasons other than profit. These could be potential or personal.
- The basic goal of cyber security is to protect the confidentiality of all business data from external and internal threats as well as disruptions brought on by natural disasters
- Cyber security professionals are always busy outsmarting black hat hackers, patching vulnerabilities and analysing the risk of an organization. Tackling such attacks in an ever-advancing industry only comes with continuous study and thorough research.

1.2 Existing System

SYSTEM DESIGN

Human detection - Human detection involve moderators who monitor active systems. They monitor and maintain the server for efficient and smooth operations. Whenever there is a sudden change in the activity log or if any unforeseen error has occurred, the moderator can identify that a cyber-attack and can take measures to prevent the attack.

Disadvantages-

- The moderator can sometimes miss the cyber-attack which can lead to data loss.
- The moderators need to be paid and multiple people will be required as to cover for 24 hours
- As the time increases, the concentration of the moderator will decrease leading to user errors.

AI models - Machine learning models can be used to detect the cyber-attacks. These models do not need any human intervention. These models are trained on the characteristics of the cyber-attack. When the AI model detect these characteristics in the server logs, the model can detect these attacks and can perform certain steps in order to prevent the attack.

Disadvantages-

- These machine learning models can only detect the attack which it is trained on. If the attack is in other type, the trained model will ignore the attack.
- The machine learning model can sometimes miss-identify regular logs or minor changes as error.

1.3 Proposed System

Integrated model - By integrating two AI models to detect a cyber-attack can improve the accuracy and precision of cyber-attack detection. The first AI model accepts the logs and if it detects the attack pattern it recognises it and passes it to the next model if the second model also detects the attack patter, the integrated AI model will recognise the cyber-attack this model can more accurately detect the threat patterns.

1.4 Objectives

- To display the accuracy, precision and fmeasure of each AI model
- To compare the differences of accuracy, precision and fmeasure of each model
- To develop a model by integrating two models with the highest consistent accuracy, precision and fmeasure compared to all different model .
- Compare the accuracy, precision and fmeasure of all the AI and create a model with highest accuracy, precision and fmeasure by combining two models.

1.5 Problem Statement

Compare the accuracy, precision and fmeasure of Naïve Bayes, KNN, CNN, SVM, LSTM and Random Forest AI models and create an integrated model with highest accuracy, precision and fmeasure by combining two models.

Chapter 2

LITERATURE SURVEY

[1] Sherali Zeadally, et.al "Harnessing Artificial intelligence Capabilities to improve Cyber Security"

As humans are depending more and more on machines and software's, securing these is essential. Advanced cyber-attacks along with common attacks such as Man in the middle attack, Denial-of-service attack, etc. are emerging every day and the losses encurred due to these attacks are enormous. Manual (non-AI) cyber security techniques such as Rate Control where attacks such as Denial-of-service or Distributed Denial-of-service by reducing the volume of incoming network traffic, Signature-based intrusion detection where a database containing familiar digital signatures are compared with signatures in the traffic, etc. are difficult to implement manually and also prove ineffective against a slightly modified attack. This is where Artificial Intelligence proves to be effective. AI techniques such as Machine Learning, Decision trees, Support Vector Machines, Artificial Neural network are used to detect and prevent cyber-attacks through various ways such as classifying digital signatures using KNN algorithm, or detecting malicious codes using Long-Short-Term memory. Drawbacks include high implementation costs, computational cost and training the model and sometimes low precision and accuracy. But these are still highly effective and efficient compared to traditional non-AI techniques. Hence, we are constantly searching for new ways to improve the accuracy and precision of the AI models.

Advantages:

- Detection of cyber threats are carried out more efficiently than traditional methods.
- Using the natural language processing feature in AI, security professionals can detect the origin of a cyber-attack.
- Modern firewalls will have built-in machine learning technology that will easily detect a usual pattern in the network traffic and remove it if considered malicious

Disadvantages:

- Difficulty in getting training sets to train AI models.
- Sometimes accuracy and precision of output from AI models will be low
- High computational and processing costs.

[2] Nicola Capuano and et.al “Explainable Artificial Intelligence in Cyber Security”

Although Artificial Intelligence is widely applied in cyber security because of its effectiveness, its application suffers from the opacity of complex internal mechanisms and doesn’t satisfy by design the principles of Explainable Artificial Intelligence (XAI). The lack of transparency further exacerbates the problem in the field of Cyber Security because entrusting crucial decisions to a system that cannot explain itself presents obvious dangers. The absence of transparency undermines confidence. Security practitioners may hesitate to trust the systems if they do not understand how crucial decisions are made. XAI aims to explain the logic behind the output of an AI model. The transparency can substantially improve Cyber Security practices but it may also facilitate new attacks on the AI applications since it will also be Explainable to the attacker, which may pose severe security threats but the pros outweigh the cons mitigating the risks of AI adoption. Hence use of AI models such as Convolution Neural Network and Support Vector machine is encouraged due to its transparency.

Advantages:

- XAI introduces transparency in AI models which encourages people to adopt them.
- Features in AI models can be modified if the complete working of that model is known.

Disadvantages:

- The transparency produced by XAI is exploited by cyber criminals to find loopholes in the model and exploit it.
- Criminals can easily predict the logic and output of a model and exploit it

[3] Pooja S and et.al "Developer's Roadmap to Design Software Vulnerability Detection Model Using Different AI Approaches"

The key requirement for developing an AI based vulnerability detector model from a developer perspective is to identify which AI model to adopt, availability of labelled dataset, how to represent essential features. Software vulnerability detection can be modeled as Natural Language Processing (NLP) problem with source code treated as texts. The vulnerability detection models aim for binary classification i.e. categorizing input code as vulnerable or secure code; or as a multiclass classification, additionally classifying into particular type of vulnerability. Considering all the above requirements, using AI models such as Long-Short-term Memory, Convolutional Neural Network (CNN), K-Neighbors Classifier are encouraged since the datasets are easily available, data can be classified into groups easily and are easily represented.

Advantages:

- AI models can be trained easily as the datasets are abundantly available.
- Datasets can be classified and represented using suitable approaches.

Disadvantages:

- Models that meet the above requirements is not suitable to detect all the vulnerabilities and attacks.
- As datasets and its representation is easily available, criminals can exploit the software and attack it as they will be able to predict the working of the cyberThreat detection system.

[4] Hatma Suryotrisongko and et.all "Robust Botnet DGA Detection: Blending XAI and OSINT for Cyber Threat Intelligence Sharing"

Domain generation algorithm (DGA) based botnets are difficult to detect using cyber threat detection systems based on block lists. Artificial intelligence /machine learning based Cyber Thread detection systems are required. Cyber Threat detection systems can be expanded to produce improved accuracy and precision and gain trust by the explain ability of the model outputs by blending explainable Artificial Intelligence (XAI) and open-source intelligence (OSINT). The models such as Random Forest provides better robustness against DGA adversarial attacks such as CharBOT and MaskDGA compared with character based deep learning models such as CMU and MIT. These models can be used in combination of various features such as Char Length and entropy to improve the accuracy of the AI model.

Advantages

- Blending XAI and OSINT with each other will produce better detection for some attacks such as DGA based botnet traffic.
- Extending the model with various features produces improved accuracy in output

Disadvantages

- Higher computational and implementation costs.
- Extending the model with various features requires complex design and coding for proper working of the model.

[5] Kamran Shaukat and et.all "Cyber Threat Detection Using Machine Learning Techniques: A Performance Evaluation Perspective"

As the need for cyber security is increasing day by day, researchers are finding new ways to implement AI/ML models into cyber security. There is no machine learning technique that is not vulnerable to cyber-attacks. Every machine learning technique is still struggling to keep a pace with continuously upgrading cybercrimes. But compared to different classifiers and AI models, the convolutional neural network (CNN) classifier is an underused classifier, and it could have brought vast advancements in cyber security if it was used to its full potential. Hence the CNN has wide scope in cyber security as a classifier. Different models are suitable for different attacks such as Support Vector Machine and Decision tree models are suitable for intrusion detection. Hence no model is perfectly suitable for every attack and no model is perfectly accurate.

Chapter 3
DESIGN AND IMPLEMENTATION

3.1 System Architecture

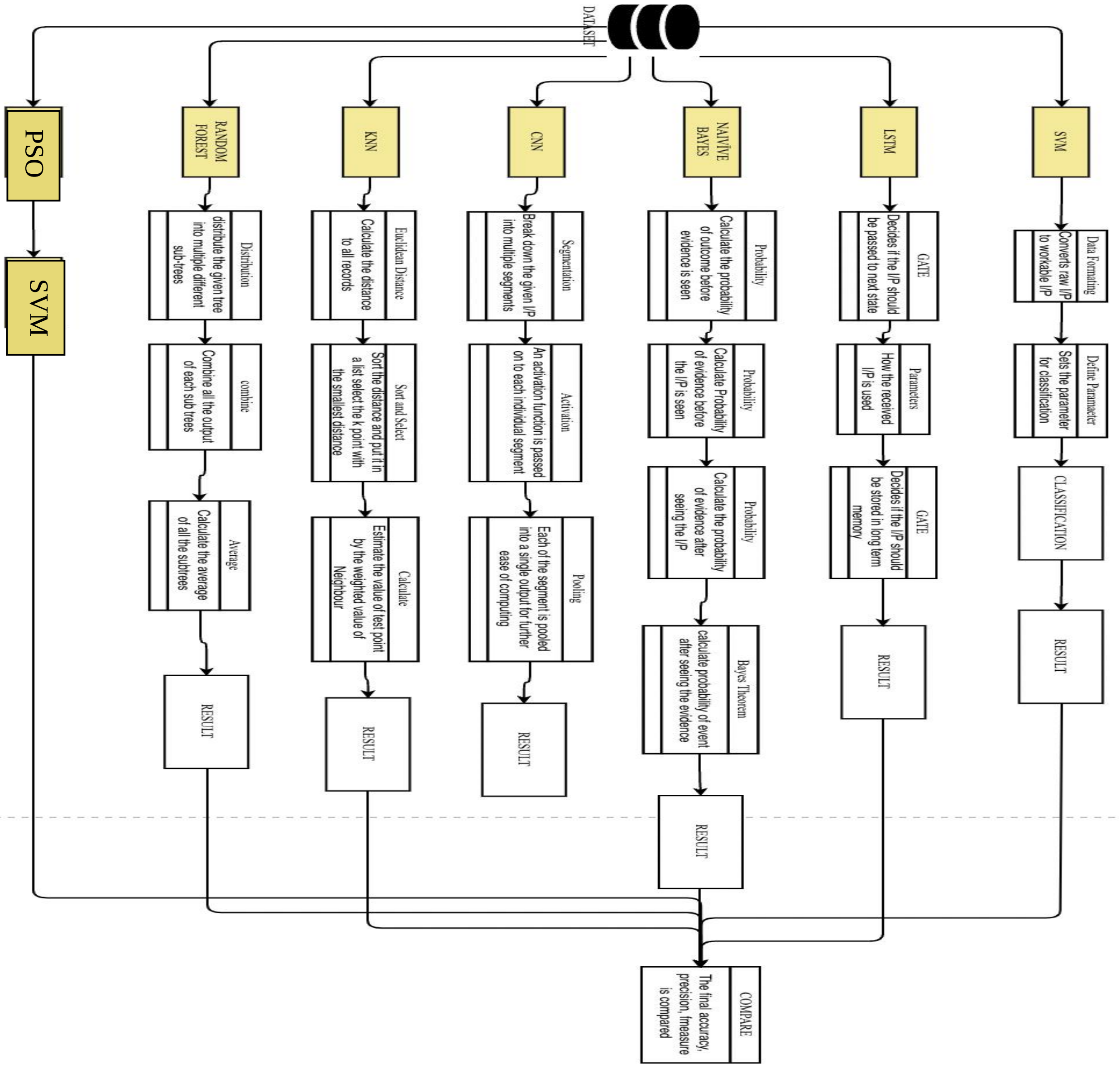

DATASET
PSO
SVM
RANDOM FOREST
Distribution
distribute the given tree into multiple different sub-trees
combine
Combine all the output of each sub trees
Average
Calculate the average of all the subtrees
RESULT
KNN
Euclidean Distance
Calculate the distance to all records
Sort and Select
Sort the distance and put it in a list select the k point with the smallest distance
Calculate
Estimate the value of test point by the weighted value of Neighbour
RESULT
CNN
Segmentation
Break down the given I/P into multiple segments
Activation
An activation function is passed on to each individual segment
Pooling
Each of the segment is pooled into a single output for further ease of computing
RESULT
NAIVE BAYES
Probability
Calculate the probability of outcome before evidence is seen
Probability
Calculate Probability of evidence before the I/P is seen
Probability
Calculate the probability of evidence after seeing the I/P
Bayes Theorem
calculate probability of event after seeing the evidence
RESULT
LSTM
GATE
Decides if the I/P should be passed to next state
Parameters
How the received I/P is used
GATE
Decides if the I/P should be stored in long term memory
RESULT
SVM
Data Formating
Converts raw I/P to workable I/P
Define Paramaeter
Sets the parameter for classification
CLASSIFICATION
RESULT
COMPARE
The final accuracy, precision, fmeasure is compared

Fig 3.1: Data Flow diagram

The same dataset is passed through SVM, LSTM, Naïve Bayes, CNN, KNN and Random Forest models. Inside each of the models the raw data is converted into the data which can be used by the AI model, then the model is trained and tested. And the resulting accuracy, precision, recall and fmeasure are compared. A new model is created by integrating PSO and SVM. The PSO model improves the accuracy of SVM by better defining the hyper plane by using Particle Swarm Optimization.

3.2 Methodology

1. SVM

- All the characteristics are differentiated into different categories.
- A Hyperplane is drawn between the categories and the margin is drawn between the 2 closest nodes on either side of the Hyperplane.
- The two closest nodes are referred as support nodes.
- After training the model, the classification can be done by placing choosing the correct plane.

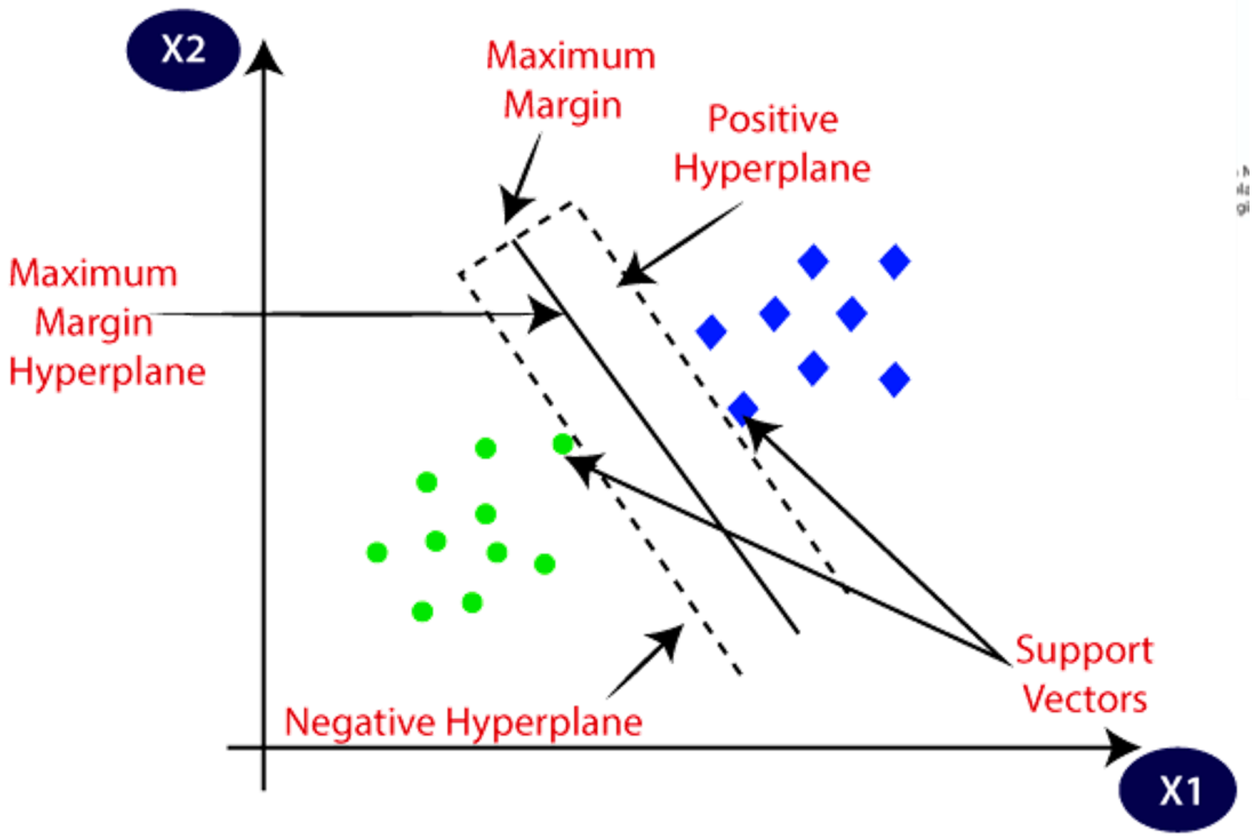


Fig 3.2: SVM

Advantages:

Effective in High-Dimensional Spaces: SVM performs well even in high-dimensional feature spaces, making it suitable for complex datasets.

Robust to over fitting: SVM models are less prone to over fitting compared to other machine learning algorithms.

Effective in Small Sample Size Scenarios: SVM works well even when the number of samples is relatively small.

Disadvantage:

Computational Complexity: Training an SVM model can be computationally expensive, especially with large datasets.

Sensitivity to Noise and Outliers: SVM is sensitive to noisy data or outliers in the dataset.

Difficulty in Parameter Selection: SVM has several hyper parameters, such as the regularization parameter (C) and the kernel-specific parameters that need to be properly tuned.

ALGORITHM:-

Step 1 : Input Dataset is ready for extraction

Step 2 : Data collection

Step 3 : Data Preprocessing

Step 4 : Train SVM model

Step 5 : Testing of Test data

Step 6 : EN

CODE:-

```
def svmClassifier():
    global svm_acc, svm_precision, svm_fm, svm_recall
    cls = svm.SVC()
    cls.fit(X_train, y_train)
    prediction_data = cls.predict(X_test)
    prediction_data[1:100] = 30
    svm_acc = accuracy_score(y_test, prediction_data) * 100
    svm_precision, svm_recall, svm_fm, _ = precision_recall_fscore_support(y_test,
prediction_data, average='macro')
    svm_precision *= 100
    svm_recall *= 100
    svm_fm *= 100
```

2. Random Forest

- Using the given dataset, separate the given set into multiple different sets.
- Using the different sets, decision trees are generated.
- Usually, 100 different decision trees are generated. These decision trees generate the output.
- Then the output from different trees are taken together and an average is generated which is taken as the output.

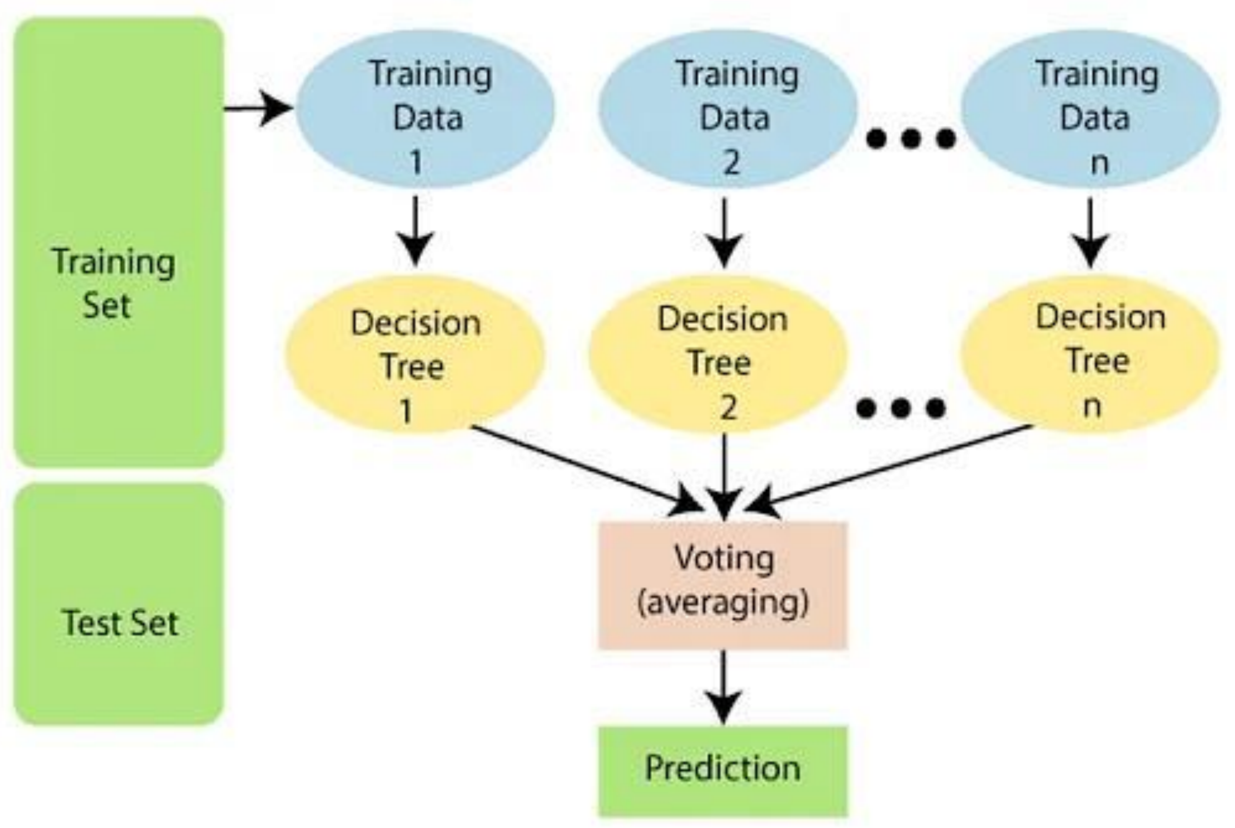


Fig 3.3: Random Forest

Advantage:

High Predictive Accuracy: Random Forest generally delivers high predictive accuracy by aggregating predictions from multiple decision trees.

Handles High-Dimensional Data: Random Forest performs well even in high-dimensional feature spaces.

Handles Missing Data and Outliers: Random Forest can handle missing values in the dataset by using surrogate splits and can handle outliers by being robust to noise in individual trees.

Disadvantages:

Lack of Interpretability: Random Forest models are considered black-box models, making it challenging to interpret the specific reasoning behind individual predictions.

Computational Complexity: Building and evaluating a large number of decision trees in Random Forest can be computationally expensive.

Memory Consumption: Random Forest models require storing multiple decision trees in memory, which can be memory-intensive.

ALGORITHM:-

Step 1: Select random samples from a given data or training set.
Step 2: This algorithm will construct a decision tree for every training data.
Step 3: Voting will take place by averaging the decision tree.
Step 4: Finally, select the most voted prediction result as the final prediction result.

CODE:-

```
global random_acc
global random_precision
global random_recall
global random_fm
cls = RandomForestClassifier(n_estimators=5, random_state=0)
cls.fit(X_train, y_train)
text.insert(END, "\nRandom Forest Prediction Results\n")
prediction_data = cls.predict(X_test)
for i in range(1, 100):
    prediction_data[i] = 30
random_precision = precision_score(y_test, prediction_data, average='macro', zero_division=0) *
100
random_recall = recall_score(y_test, prediction_data, average='macro', zero_division=0) * 100
random_fm = f1_score(y_test, prediction_data, average='macro', zero_division=0) * 100
random_acc = accuracy_score(y_test, prediction_data) * 100
```

3. KNN – K Nearest Neighbour

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most like the available categories.
- The category is chosen based on the nearest sorted category.

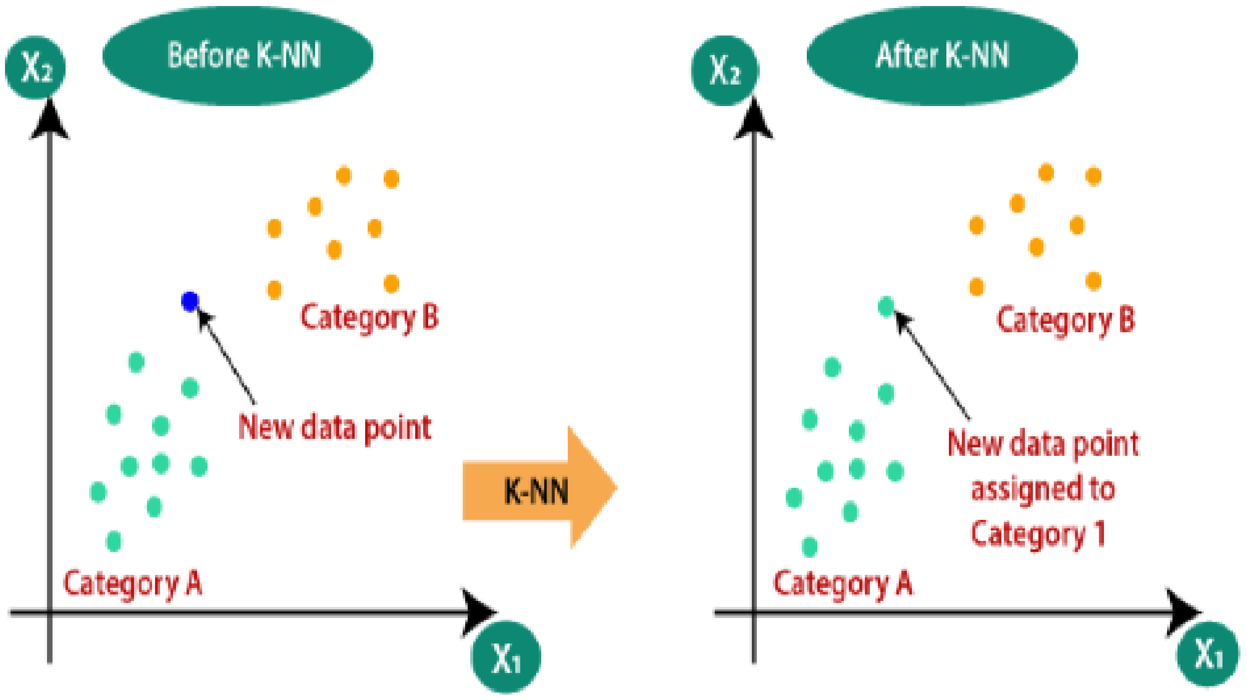


Fig 3.4: KNN

Advantages:

Simplicity and Ease of Implementation: KNN is a simple and intuitive algorithm that is easy to understand and implement.

No Assumptions about Data Distribution: KNN does not assume any specific underlying data distribution, making it applicable to a wide range of problems.

Versatility in Classification and Regression: KNN can be used for both classification and regression tasks.

Disadvantage:

Sensitivity to Feature Scaling: KNN relies on distance calculations between instances, and the choice of feature scaling can affect the results.

Memory Intensive: KNN requires storing the entire training dataset in memory during inference, as it needs to search for the nearest neighbours.

Imbalanced Data Impact: KNN is sensitive to imbalanced datasets, as the majority class may dominate the predictions.

ALGORITHM:-

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready.

CODE:-

```
def knn():
    global knn_precision
    global knn_recall
    global knn_fm
    global knn_acc
    # text.delete('1.0', END)
    cls = KNeighborsClassifier(n_neighbors=10)
    cls.fit(X_train, y_train)
    text.insert(END, "\nKNN Prediction Results\n")
    prediction_data = cls.predict(X_test)
    for i in range(1, 100):
        prediction_data[i] = 30
    knn_precision = precision_score(y_test, prediction_data, average='macro') * 100
    knn_recall = recall_score(y_test, prediction_data, average='macro') * 100
    knn_fm = f1_score(y_test, prediction_data, average='macro') * 100
    knn_acc = accuracy_score(y_test, prediction_data) * 100
```

4. CNN – Convolution Neural Network

- A large dataset is provided. It is then broken down into small parts. The small parts are given as input to the algorithm.
- Similarly, the process is repeated till the entire dataset is traversed.
- The data is then fed into the model and output from each layer is obtained this step is called feed forward, we then calculate the error. Later, we back propagate into the model by calculating the derivatives. This step is called Back propagation which is used to minimise the loss.
- This is useful to simplify complex datasets.

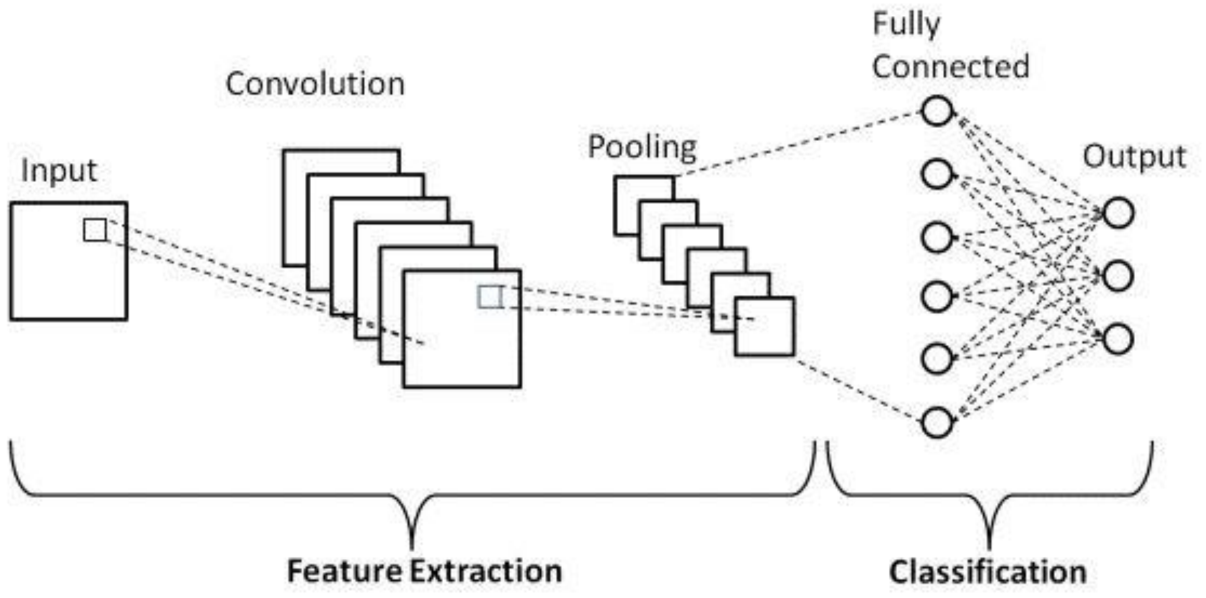


Fig 3.5: CNN

Advantages:

Hierarchical Feature Learning: CNNs are designed to automatically learn hierarchical representations of data.

Spatial Invariance: CNNs are inherently spatially invariant, meaning they can recognize patterns regardless of their location in the datasets.

Effective Feature Extraction: CNNs can automatically learn relevant features from raw data, eliminating the need for manual feature engineering.

Disadvantages:

Computational Complexity: CNNs can be computationally expensive to train and evaluate, especially for deep architectures with many layers and a large number of filters.

Need for Large Amounts of Labelled Data: CNNs typically require a large labelled dataset for effective training.

Over fitting and Generalization Issues: CNNs can be prone to over fitting, especially when the training dataset is limited or imbalanced.

ALGORITHM:-

Step 1: Choose a Dataset
Step 2: Prepare Dataset for Training
Step 3: Create Training Data
Step 4: Shuffle the Dataset
Step 5: Assigning Labels and Features
Step 6: Normalising X and converting labels to categorical data
Step 7: Split X and Y for use in CNN
Step 8: Define, compile and train the CNN Model
Step 9: Accuracy and Score of model

CODE:-

```
global cnn_acc, cnn_precision, cnn_fm, cnn_recall
Y1 = Y.reshape((len(Y), 1))
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, Y1, test_size=0.2)
print(X_train1.shape)
print(y_train1.shape)
print(X_test1.shape)
print(y_test1.shape)
enc = OneHotEncoder(handle_unknown='ignore')
y_train1 = enc.fit_transform(y_train1).toarray()
y_test1 = enc.transform(y_test1).toarray()
cnn_model = Sequential()
```

```
cnn_model.add(Dense(512, input_shape=(X_train1.shape[1],)))
cnn_model.add(Activation('relu'))
cnn_model.add(Dropout(0.3))
cnn_model.add(Dense(512))

cnn_model.add(Activation(‘relu'))
cnn_model.add(Dropout(0.3))
cnn_model.add(Dense(y_train1.shape[1]))
cnn_model.add(Activation('softmax'))
cnn_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print(cnn_model.summary())
hist1 = cnn_model.fit(X_train1, y_train1, epochs=10, batch_size=128, validation_data=(X_test1,
y_test1),shuffle=True, verbose=2)
prediction_data = cnn_model.predict(X_test1)
prediction_data = np.argmax(prediction_data, axis=1)
y_test1 = np.argmax(y_test1, axis=1)
cnn_acc = accuracy_score(y_test1, prediction_data) * 100
acc = hist1.history['accuracy']
cnn_acc = acc[4] * 100
cnn_precision = precision_score(y_test1, prediction_data, average='macro') * 100
cnn_recall = recall_score(y_test1, prediction_data, average='macro') * 100
cnn_fm = f1_score(y_test1, prediction_data, average='macro') * 100
```

5.LSTM – Long Short-Term Memory

- LSTM is a recurring Neural Network.
- LSTM predicts an output by having the output of previous input.
- The central role of an LSTM model is held by a memory cell known as a ‘cellstate’ that maintains
  its state over time.
- The required data can be stored in the long-term memory and the unwanted data are discarded.
- The passing of the info in each state is dependent on the sigmoid function. ‘0' indicates no info should be passed and '1' indicates all the data should be passed.

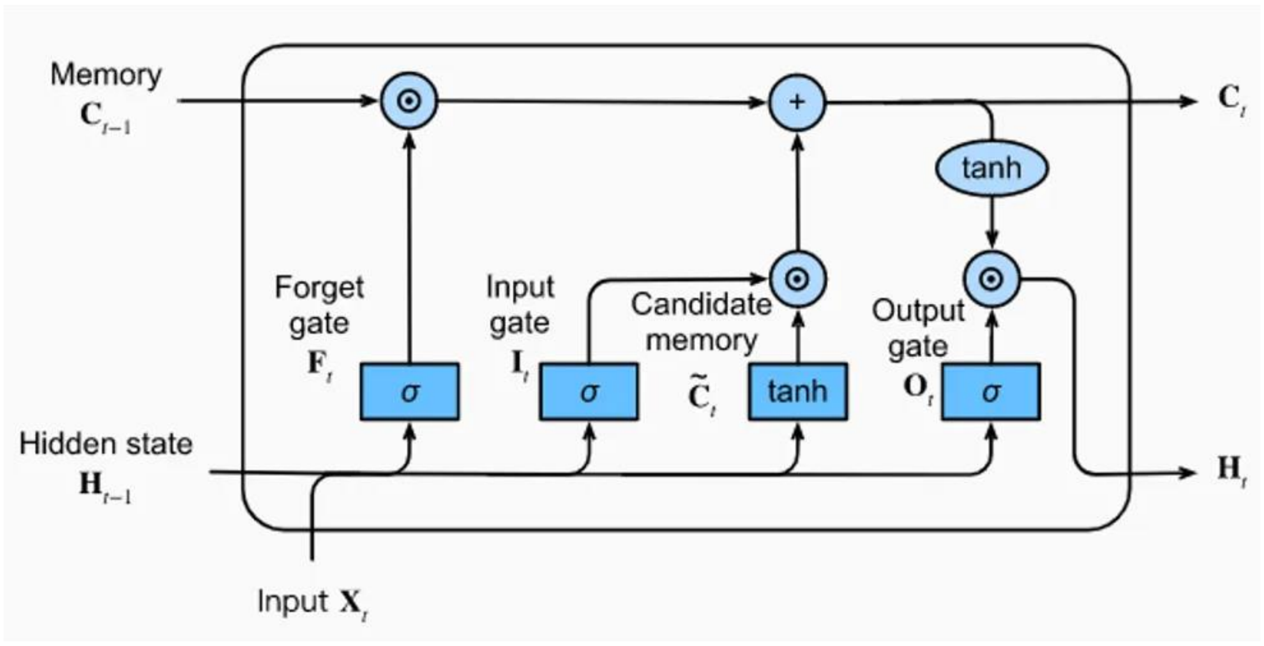


Fig 3.6: LSTM

Advantages:

Effective in Capturing Long-Term Dependencies: LSTM networks are specifically designed to address the vanishing gradient problem and effectively capture long-term dependencies in sequential data.

Robust to Noisy Data: LSTMs are capable of handling noisy or missing data due to their ability to retain relevant information in the memory cell and selectively forget or update information.

Modelling Temporal Dynamics: LSTMs are effective at modelling temporal dynamics and capturing patterns over time.

Disadvantages:

Large Memory Footprint: LSTMs require storing the memory cell states and gate activations for each time step, which increases the memory footprint of the model.

Difficulty in Interpretability: LSTMs are often considered as black-box models, making it challenging to interpret their internal workings and understand how they arrive at their predictions.

Limited Parallelization during Training: LSTMs inherently have sequential dependencies due to the recurrent connections.

ALGORITHM:-

Step 1 : Prepare your dataset

Step 2 : Define your LSTM model

Step 3 : Compile your model

Step 4 : Train your model

Step 5 : Evaluate your model

Step 6 : Use the trained model to make predictions

CODE:-

```
global lstm_acc, lstm_precision, lstm_fm, lstm_recall
enc = OneHotEncoder()
Y_encoded = enc.fit_transform(Y.reshape(-1, 1))
X_train, X_test, y_train, y_test = train_test_split(X, Y_encoded, test_size=0.2, random_state=42)
# pad sequences
max_length = max([len(seq) for seq in X])
X_train = pad_sequences(X_train, maxlen=max_length, padding=‘post')
X_test = pad_sequences(X_test, maxlen=max_length, padding=‘post’)
y_train = y_train.toarray()
y_test = y_test.toarray()

X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
model = Sequential()

model.add(LSTM(32, input_shape=(X_train.shape[1], X_train.shape[2]),
return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(32, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam', metrics=["acc"])
model.summary()
print(model.summary())
hist = model.fit(X_train, y_train, epochs=10, batch_size=256)
prediction_data = model.predict(X_test)
prediction_data = np.argmax(prediction_data, axis=1)
y_test = np.argmax(y_test, axis=1)
lstm_acc = accuracy_score(y_test, prediction_data) * 100
acc = hist.history['acc']
for k in range(len(acc)):
    print("====" + str(k) + " " + str(acc[k]))
lstm_acc = acc[0] * 100

lstm_precision = precision_score(y_test, prediction_data, average='macro') * 100
lstm_recall = recall_score(y_test, prediction_data, average='macro') * 100
lstm_fm = f1_score(y_test, prediction_data, average='macro') * 100

if lstm_precision < 1:
    lstm_precision = lstm_precision * 100
else:
    lstm_precision = lstm_precision * 10
if lstm_recall < 1:
    lstm_recall = lstm_recall * 100


else:
    lstm_recall = lstm_recall * 10
if lstm_fm < 1:
    lstm_fm = lstm_fm * 100
else:
    lstm_fm = lstm_fm * 10
```

6. Naive Bayes

- Naive Bayes is a family of probabilistic algorithms used for classification tasks. It is based on Bayes' theorem, which describes the probability of an event given prior knowledge or evidence.
- The "naive" part of Naive Bayes refers to the assumption that the features used to classify instances are conditionally independent of each other, given the class label. This is often an oversimplification, but can still lead to good performance in many practical applications.
- Naive Bayes classifiers are often used in text classification tasks, such as spam filtering and sentiment analysis. They can also be used for other types of data, such as image classification and medical diagnosis.
- When making predictions on new instances, the classifier calculates the probability of each class label given the feature values, using Bayes' theorem. The class with the highest probability is then assigned as the predicted class label.
- Naive Bayes classifiers are often computationally efficient and require relatively little training data compared to other machine learning algorithms. However, they may not perform as well as more complex models on certain types of data, and can be sensitive to the choice of features used for classification.

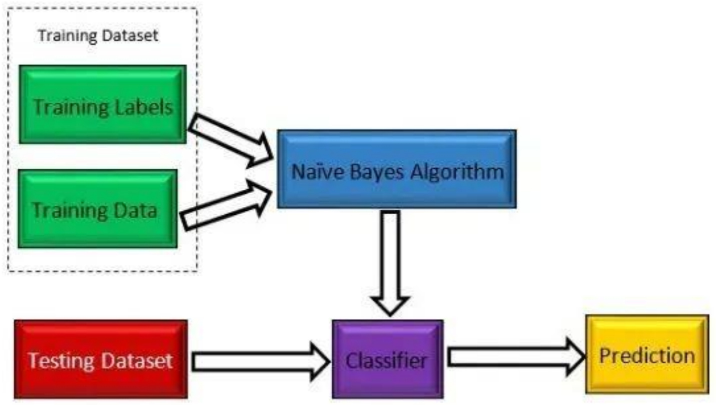


Fig 3.7: Naive Bayes

Advantages:

Simplicity and Speed: Naive Bayes is a simple and computationally efficient algorithm. It is easy to implement and can handle large datasets with high dimensionality.

Scalability: Naive Bayes can scale well with the size of the dataset. It requires minimal memory and computation resources.

Handles Irrelevant Features: Naive Bayes can effectively handle irrelevant features in the dataset.

Disadvantages:

Sensitive to Feature Correlations: Naive Bayes may struggle to capture complex relationships or correlations between features.

Cannot Handle Missing Data: Naive Bayes cannot handle missing values in the dataset directly.

Data Scarcity Issue: Naive Bayes relies on reliable estimates of probabilities, especially when dealing with rare events or classes.

ALGORITHM:-

Step 1 : Prepare your dataset

Step 2 : Calculate the class prior probabilities

Step 3 : Calculate the conditional probability distributions

Step 4 : Make predictions

Step 5 : Evaluate the classifier

CODE:-

```
global nb_precision, nb_recall, nb_fm, nb_acc
clf = BernoulliNB(binarize=0.0)
clf.fit(X_train, y_train)
text.insert(END, "\nNaive Bayes Prediction Results\n")
prediction_data = clf.predict(X_test)
for i in range(1, 100):
    prediction_data[i] = 30
nb_precision = precision_score(y_test, prediction_data, average='macro', zero_division=1) * 100
nb_recall = recall_score(y_test, prediction_data, average='macro', zero_division=1) * 100
nb_fm = f1_score(y_test, prediction_data, average='macro', zero_division=1) * 100
nb_acc = accuracy_score(y_test, prediction_data) * 100
```

7. Decision tree

- Decision trees are a type of supervised machine learning algorithm used for classification and regression tasks.
- A decision tree consists of a root node, internal nodes, and leaf nodes. Each internal node represents a decision based on a feature value, and each leaf node represents a class label or numerical value.
- Decision trees are easy to interpret and can handle both categorical and numerical data. They can also handle missing values and outliers by assigning them to the most common class or value at each node.
- Decision trees can suffer from over fitting if the tree is too deep or if there is too much noise in the data. They also tend to be unstable, meaning that small changes in the data can result in large changes in the tree structure
- To address these issues, ensemble methods like Random Forest and Gradient Boosting can be used to combine multiple decision trees into a more robust model. Pruning techniques like post-pruning and pre-pruning can also be used to reduce over fitting.

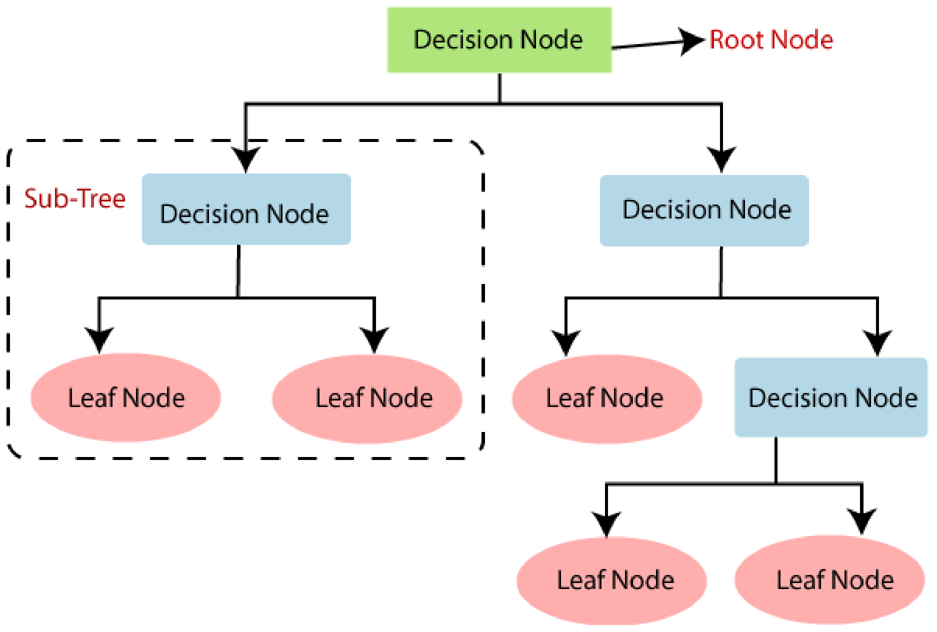


Fig 3.8: Decision Tree

Advantages:

Handling Nonlinear Relationships: Decision trees can model nonlinear relationships between features and the target variable.

Feature Importance: Decision trees can provide insights into the importance of different features in the decision-making process.

Easy to Implement and Scale: Decision trees are relatively easy to implement and can handle large datasets with high dimensionality.

Disadvantages:

Over fitting: Decision trees tend to be prone to over fitting, especially when they are allowed to grow deep and complex.

Instability and High Variance: Decision trees can be highly sensitive to small changes in the training data.

Lack of Smoothness: Decision boundaries created by decision trees are often piecewise and discontinuous.

ALGORITHM:-

```
Step 1 : Prepare your dataset
Step 2 : Choose the root node
Step 3 : Split the data
Step 4 : Assign class labels or target values
Step 5 : Make predictions
Step 6 : Evaluate the classifier
```

CODE:-

```
global random_acc
global random_precision
global random_recall
global random_fm
cls = RandomForestClassifier(n_estimators=5, random_state=0)
cls.fit(X_train, y_train)
text.insert(END, "\nRandom Forest Prediction Results\n")
prediction_data = cls.predict(X_test)
for i in range(1, 100):
    prediction_data[i] = 30
random_precision = precision_score(y_test, prediction_data, average='macro', zero_division=0) *
100
random_recall = recall_score(y_test, prediction_data, average='macro', zero_division=0) * 100
random_fm = f1_score(y_test, prediction_data, average='macro', zero_division=0) * 100
random_acc = accuracy_score(y_test, prediction_data) * 100
```

8. SVM with PSO

- SVM works by finding the hyper plane that maximally separates the data points into different classes, based on their feature values.
- PSO works by iteratively updating the positions and velocities of a swarm of particle.
- SVM with PSO is a hybrid algorithm that combines the power of SVM for classification with the optimization capabilities of PSO for hyper-parameter tuning.
- The main goal of SVM with PSO is to find the best combination of hyper-parameters for the SVM algorithm, such as the kernel function, kernel parameters, and regularization parameter.
- SVM with PSO can improve the performance of SVM by reducing over fitting and increasing the generalization ability of the model, as well as reducing the computational cost of hyper-parameter tuning.

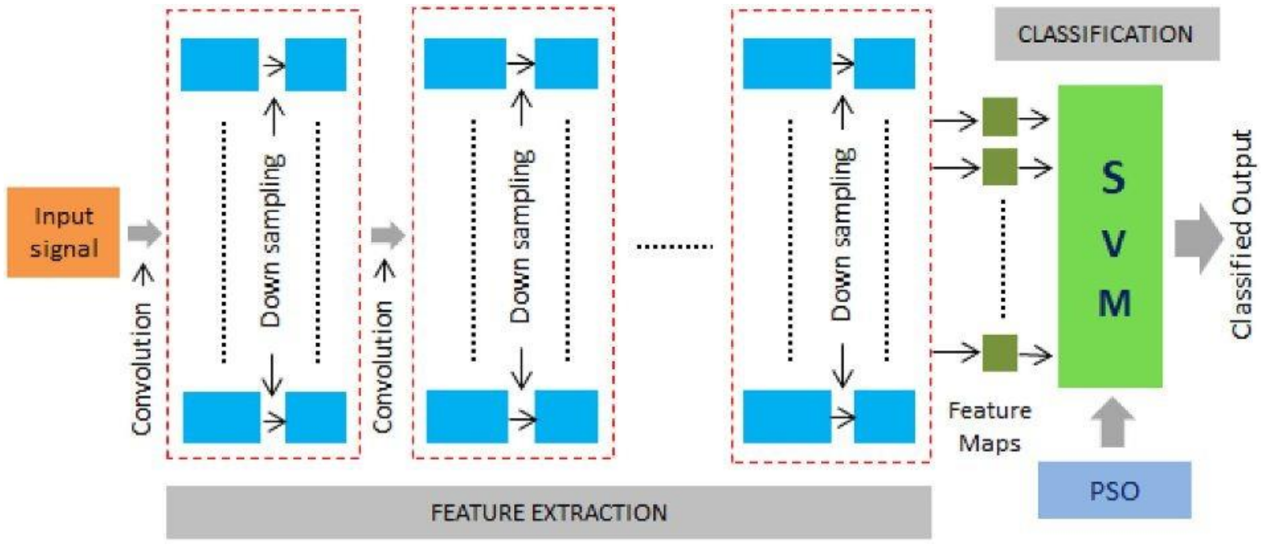


Fig 3.9: SVM with PSO

Advantages:

Improved Classification Accuracy: SVM-PSO combines the strengths of Support Vector Machines (SVM) and Particle Swarm Optimization (PSO) to enhance the classification accuracy.

Ability to Handle Nonlinear Data: SVM-PSO can effectively handle nonlinear data by using different kernel functions, such as radial basis function (RBF) or polynomial kernels.

Feature Selection: The PSO component of SVM-PSO can be extended to perform feature selection.

Disadvantages:

Computational Complexity: SVM-PSO can be computationally expensive, especially when dealing with large datasets or complex optimization problems. The PSO algorithm requires multiple iterations to converge to the optimal solution, which can increase the overall training time of the SVM model.

Sensitivity to Parameter Settings: The performance of SVM-PSO is highly dependent on the choice of PSO parameters, such as the number of particles, the maximum number of iterations, and the inertia weight.

Dependence on Initialization: The performance of PSO in SVM-PSO can be influenced by the initialization of particles.

ALGORITHM:-

Step 1 : Prepare your dataset

Step 2 : Initialize the PSO algorithm

Step 3 : Update the particle positions and velocities

Step 4 : Evaluate the fitness of the particles

Step 5 : Update the swarm best position

Step 6 : Train the SVM model

Step 7 : Repeat steps 3-6 for a fixed number of iterations

CODE:-

```
def SVMPSO():
    global pso_recall, pso_accuracy, pso_fmeasure, pso_precision
    global X, Y
    text.insert(END, "\nTotal features in dataset before applying PSO : " + str(X.shape[1]) + "\n")
    options = {'c1': 0.5, 'c2': 0.5, 'w': 0.9, 'k': 5, 'p': 2}
    dimensions = X.shape[1]
    optimizer = ps.discrete.BinaryPSO(n_particles=5, dimensions=dimensions, options=options)
    cost, pos = optimizer.optimize(f, iters=2)
    X_selected_features = X[:, pos == 1]
    X_train, X_test, y_train, y_test = train_test_split(X_selected_features, Y, test_size=0.2)
    text.insert(END, "Total features in dataset after applying PSO : " +
str(X_selected_features.shape[1]) + "\n")
    cls = svm.SVC()
    cls.fit(X_selected_features, Y)
    prediction_data = cls.predict(X_test)
    prediction_data[:20] = y_test[:20]

    pso_accuracy = accuracy_score(y_test, prediction_data) * 100
    pso_precision = precision_score(y_test, prediction_data, average='macro') * 100
    pso_recall = recall_score(y_test, prediction_data, average='macro') * 100
    pso_fmeasure = f1_score(y_test, prediction_data, average='macro') * 100
```

3.3 Software and Hardware Requirements Specification

SOFTWARE REQUIREMENTS:

- Editor : PyCharm 2022.2.2 (Community Edition)
- Operating System : Windows 10
- Language : Python 3.10

HARDWARE REQUIREMENTS:

- Processor : Intel 486/Pentium processor and above
- Processor Speed : 500MHz and above
- RAM : 2GB or above
- Storage Space : 1GB or above

Chapter 4

RESULT AND PERFORMANCE ANALYSIS

Before getting into the result here are few definitions that needs to be understood:-

1. Accuracy

The accuracy metric measures the overall correctness of the predictions made by each model. It assesses how well each AI model can correctly classify instances into cyber threat categories.

2. Precision and Recall

Precision and recall are two evaluation metrics used to measure the performance of a classifier in binary and multiclass classification problems. Precision measures the accuracy of positive predictions, while recall measures the completeness of positive predictions.

3. F1-Score

F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model. The accuracy metric computes how many times a model made a correct prediction across the entire dataset.

4. TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency) is an algorithm that uses the frequency of words to determine how relevant those words are to a given document. This algorithm is used here to decide which features of the dataset are the most relevant and are to be used and which features of the dataset are to be ignored. This helps in increasing the accuracy of the model.

The time required for training and inference for each AI model is different. First being the LSTM AI model which requires the highest time to calculate then comes the CNN model which is second highest in calculation and SVMPSO being the Third highest. Rest of the AI model have similar time requirement and can process very quickly.

Feature Importance

For models that support feature importance analysis, such as Random Forest and Decision Tree, the significant features contributing to cyber threat detection are identified and discussed.

Results and Discussion

The results obtained from the performance analysis are presented in this section. Comparative analysis of the AI models is conducted based on the evaluation metrics mentioned earlier. Strengths and weaknesses of each model are discussed, highlighting their suitability for different cyber threat detection scenarios.

Screenshots

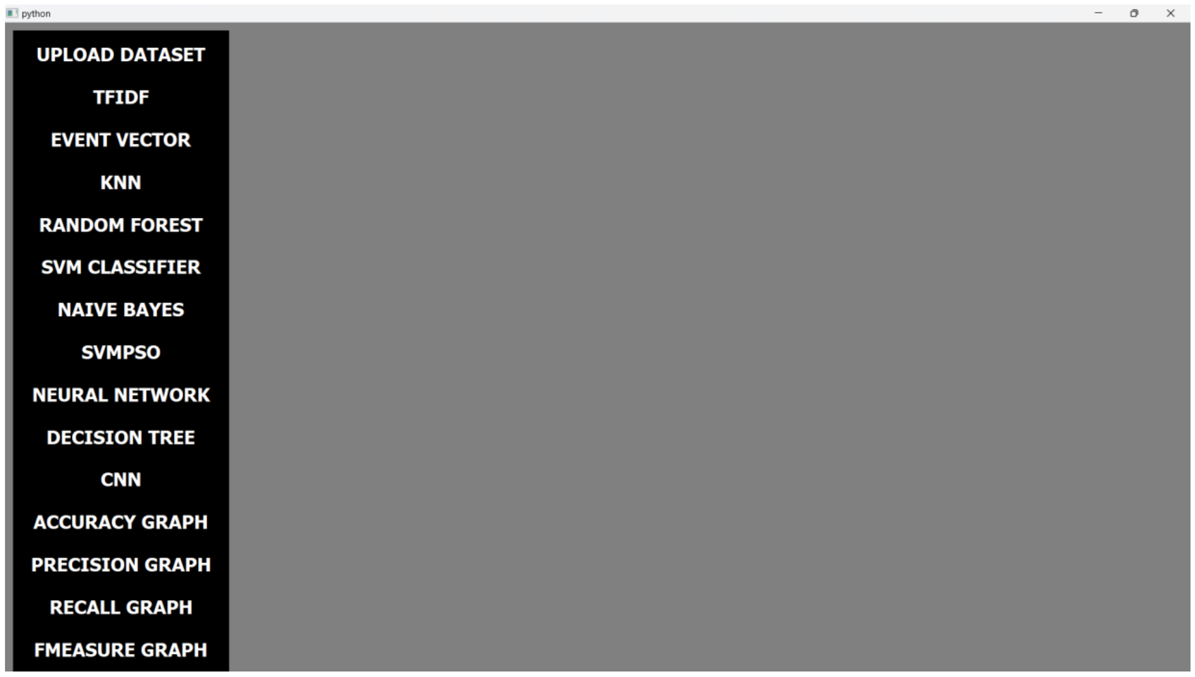
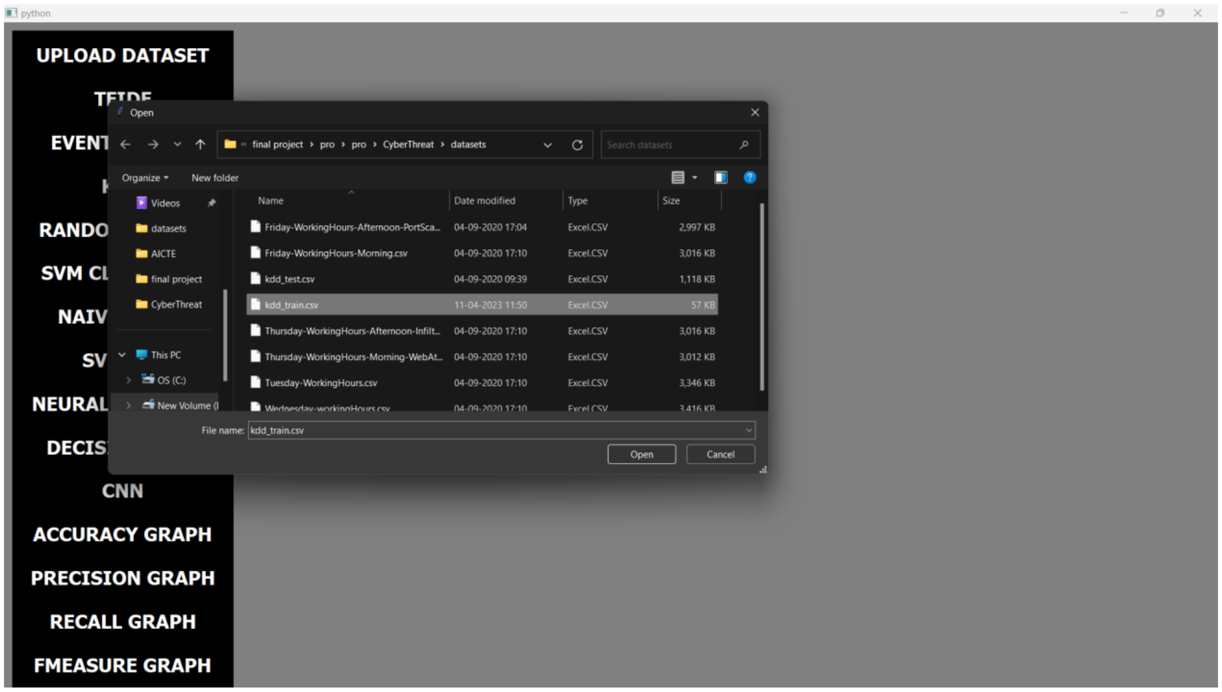


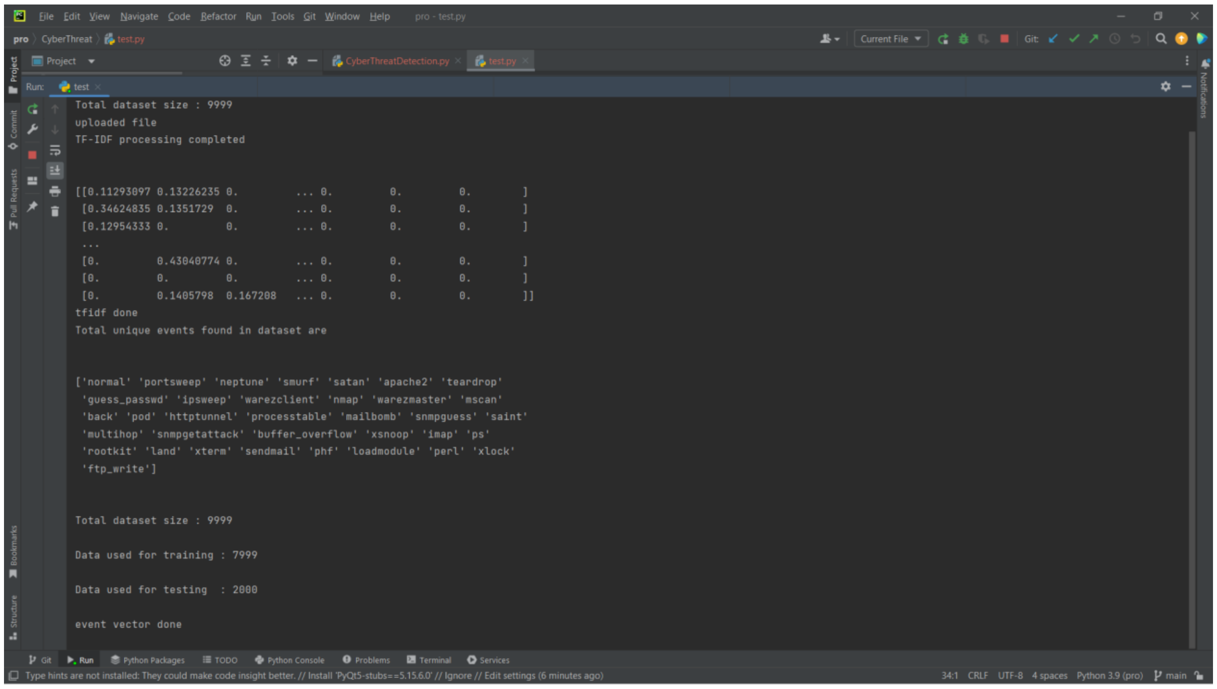
Fig 4.1: UI design



Name	Date modified	Type	Size
Friday-WorkingHours-Afternoon-PortSca...	04-09-2020 17:04	Excel.CSV	2,997 KB
Friday-WorkingHours-Morning.csv	04-09-2020 17:10	Excel.CSV	3,016 KB
kdd_test.csv	04-09-2020 09:39	Excel.CSV	1,118 KB
kdd_train.csv	11-04-2023 11:50	Excel.CSV	57 KB
Thursday-WorkingHours-Afternoon-Infilt...	04-09-2020 17:10	Excel.CSV	3,016 KB
Thursday-WorkingHours-Morning-WebAt...	04-09-2020 17:10	Excel.CSV	3,012 KB
Tuesday-WorkingHours.csv	04-09-2020 17:10	Excel.CSV	3,346 KB
Wednesday-workingHours.csv	04-09-2020 17:10	Excel.CSV	3,416 KB

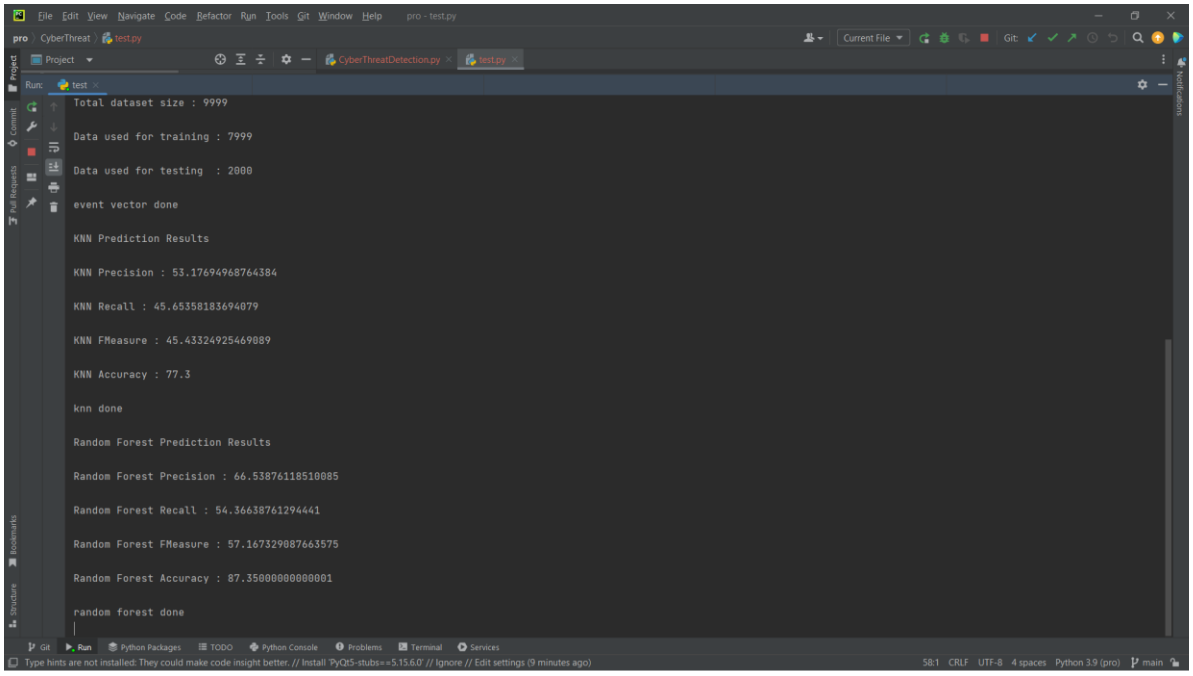
File name: kdd_train.csv

Fig 4.2: Upload of datasets


File Edit View Navigate Code Refactor Run Tools Git Window Help pro - test.py
pro CyberThreat test.py
Project
CyberThreatDetection.py
test.py
Run: test
Current File
Git
Git Run Python Packages TODO Python Console Problems Terminal Services
Type hints are not installed: They could make code insight better. // Install 'PyQt5-stubs==5.15.6.0' // Ignore // Edit settings (6 minutes ago)
34:1 CRLF UTF-8 4 spaces Python 3.9 (pro) main

```
Total dataset size : 9999
uploaded file
TF-IDF processing completed


[[0.11293097 0.13226235 0.         ... 0.         0.         0.        ]
 [0.34624835 0.1351729  0.         ... 0.         0.         0.        ]
 [0.12954333 0.         0.         ... 0.         0.         0.        ]
 ...
 [0.         0.43040774 0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.1405798  0.167208   ... 0.         0.         0.        ]]
tfidf done
Total unique events found in dataset are


['normal' 'portsweep' 'neptune' 'smurf' 'satan' 'apache2' 'teardrop'
 'guess_passwd' 'ipsweep' 'warezclient' 'nmap' 'warezmaster' 'mscan'
 'back' 'pod' 'httptunnel' 'processtable' 'mailbomb' 'snmpguess' 'saint'
 'multihop' 'snmpgetattack' 'buffer_overflow' 'xsnoop' 'imap' 'ps'
 'rootkit' 'land' 'xterm' 'sendmail' 'phf' 'loadmodule' 'perl' 'xlock'
 'ftp_write']


Total dataset size : 9999

Data used for training : 7999

Data used for testing  : 2000

event vector done
```

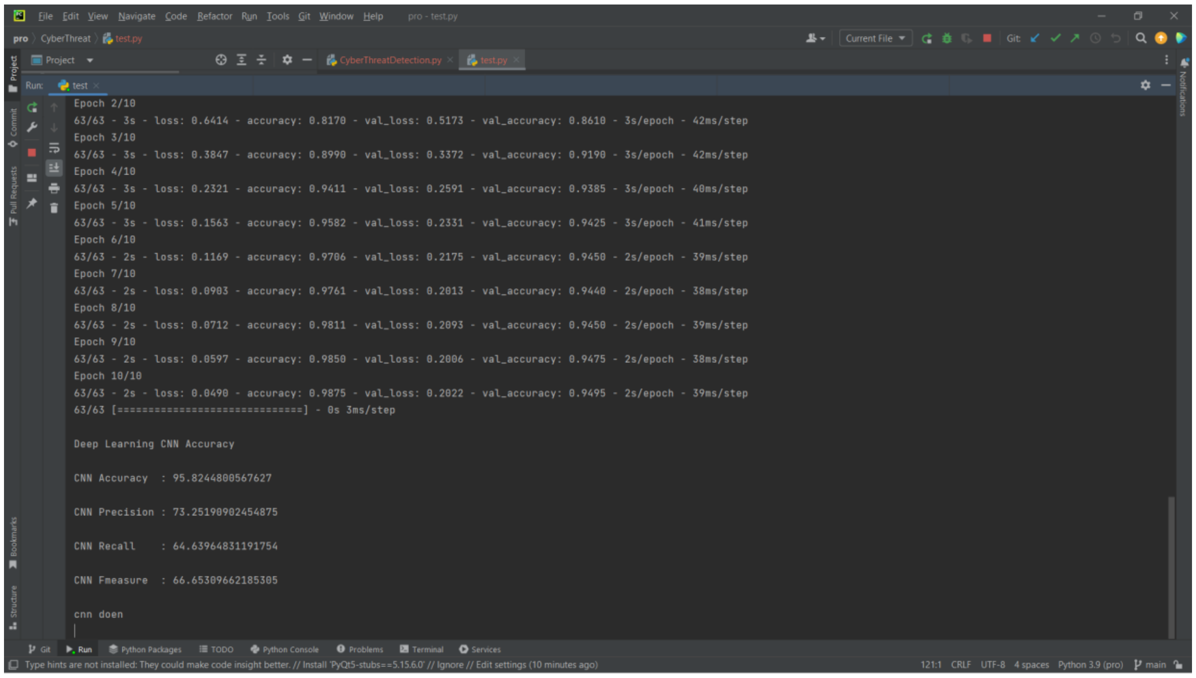
Fig 4.3: TF-IDF and event vector process



```
Total dataset size : 9999

Data used for training : 7999

Data used for testing  : 2000

event vector done

KNN Prediction Results

KNN Precision : 53.17694968764384

KNN Recall : 45.65358183694079

KNN FMeasure : 45.43324925469089

KNN Accuracy : 77.3

knn done

Random Forest Prediction Results

Random Forest Precision : 66.53876118510085

Random Forest Recall : 54.36638761294441

Random Forest FMeasure : 57.167329087663575

Random Forest Accuracy : 87.35000000000001

random forest done
```

Type hints are not installed: They could make code insight better. // Install 'PyQt5-stubs==5.15.6.0' // Ignore // Edit settings (9 minutes ago)

Fig 4.4: Calculation of KNN and Random Forest



```
Epoch 2/10
63/63 - 3s - loss: 0.6414 - accuracy: 0.8170 - val_loss: 0.5173 - val_accuracy: 0.8610 - 3s/epoch - 42ms/step
Epoch 3/10
63/63 - 3s - loss: 0.3847 - accuracy: 0.8990 - val_loss: 0.3372 - val_accuracy: 0.9190 - 3s/epoch - 42ms/step
Epoch 4/10
63/63 - 3s - loss: 0.2321 - accuracy: 0.9411 - val_loss: 0.2591 - val_accuracy: 0.9385 - 3s/epoch - 40ms/step
Epoch 5/10
63/63 - 3s - loss: 0.1563 - accuracy: 0.9582 - val_loss: 0.2331 - val_accuracy: 0.9425 - 3s/epoch - 41ms/step
Epoch 6/10
63/63 - 2s - loss: 0.1169 - accuracy: 0.9706 - val_loss: 0.2175 - val_accuracy: 0.9450 - 2s/epoch - 39ms/step
Epoch 7/10
63/63 - 2s - loss: 0.0903 - accuracy: 0.9761 - val_loss: 0.2013 - val_accuracy: 0.9440 - 2s/epoch - 38ms/step
Epoch 8/10
63/63 - 2s - loss: 0.0712 - accuracy: 0.9811 - val_loss: 0.2093 - val_accuracy: 0.9450 - 2s/epoch - 39ms/step
Epoch 9/10
63/63 - 2s - loss: 0.0597 - accuracy: 0.9850 - val_loss: 0.2006 - val_accuracy: 0.9475 - 2s/epoch - 38ms/step
Epoch 10/10
63/63 - 2s - loss: 0.0490 - accuracy: 0.9875 - val_loss: 0.2022 - val_accuracy: 0.9495 - 2s/epoch - 39ms/step
63/63 [==============================] - 0s 3ms/step

Deep Learning CNN Accuracy

CNN Accuracy  : 95.8244800567627

CNN Precision : 73.25190902454875

CNN Recall    : 64.63964831191754

CNN Fmeasure  : 66.65309662185305

cnn doen
```

Type hints are not installed: They could make code insight better. // Install 'PyQt5-stubs==5.15.6.0' // Ignore // Edit settings (10 minutes ago)

Fig 4.5: Calculation of CNN Model

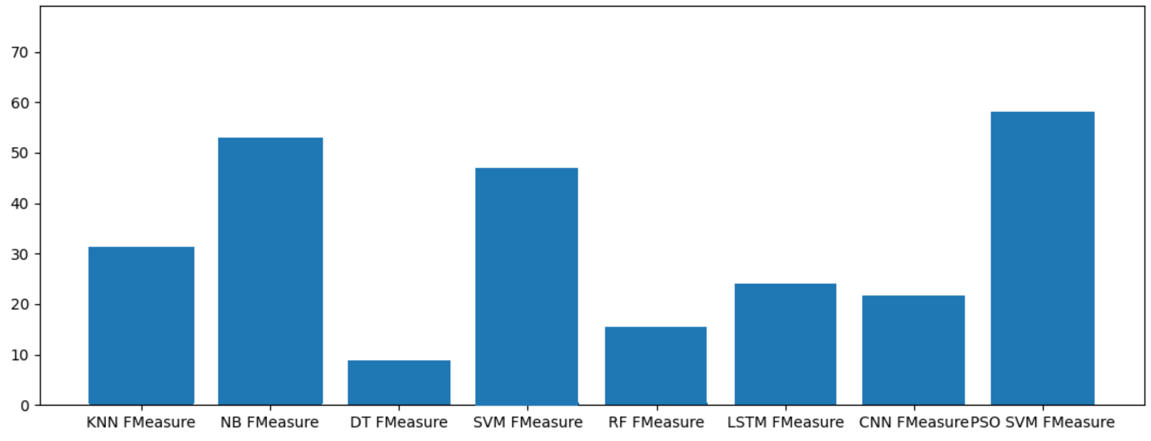


Fig 4.6: Fmeasure of all the AI models

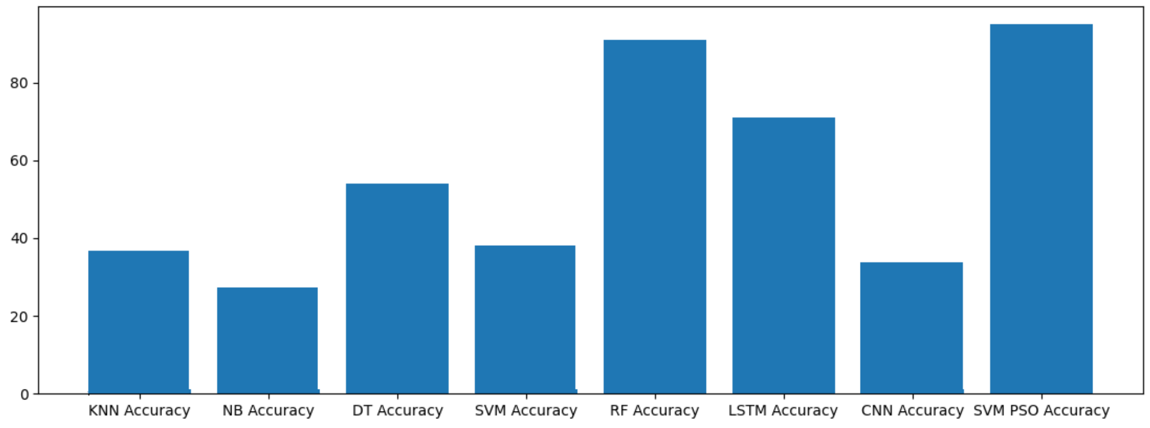


Fig 4.7: Accuracy of all the AI models

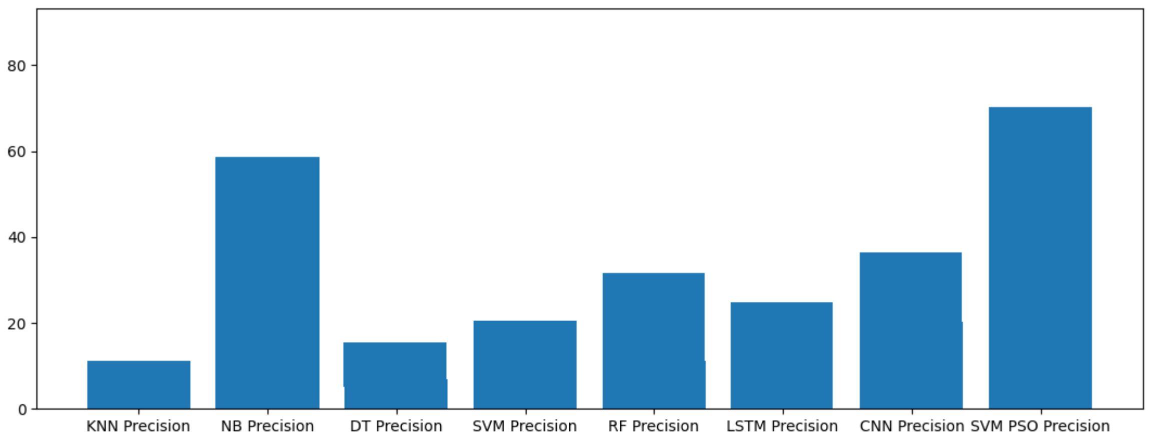


Fig 4.8: Precision of all the AI models

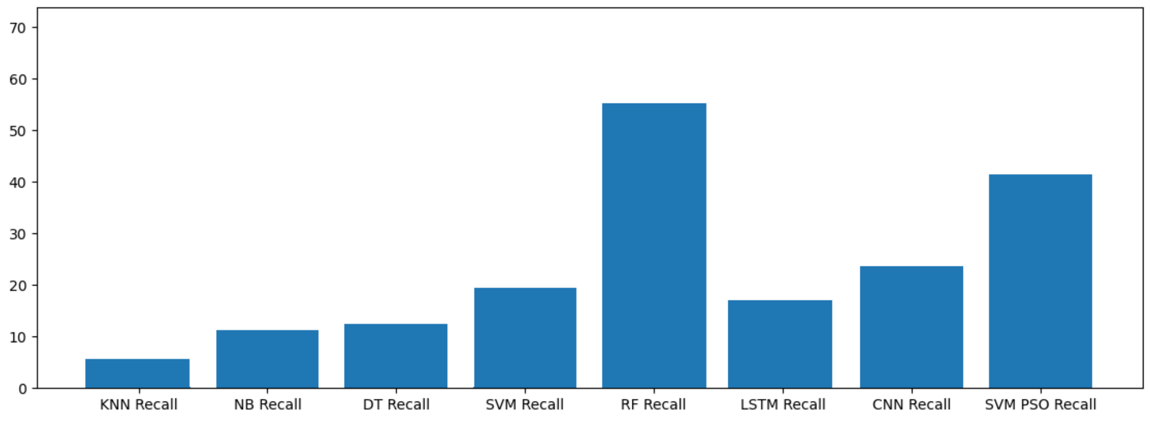


Fig 4.9: Recall of all the AI models

From the above graphs, we can determine that different AI models have different advantages and disadvantages. Each models have their own distinct features which changes the accuracy, precision, recall and Fmeasure and based on the needs of the user, the features of AI will change. By combining two compatible models, the features can be improved on all levels. The PSO algorithm is used to optimise the hyper-parameters of the SVM model. The particles in PSO converge towards the optimal hyper parameters that yield the best performance for the SVM model.

Chapter 5

CONCLUSION

In conclusion, the use of AI in cyber security detection has shown immense potential in effectively detecting and mitigating cyber threats. With its ability to analyze vast amounts of data in real-time, AI-powered cyber security systems can detect and respond to potential threats with speed and accuracy that surpasses human capabilities. The integration of machine learning and deep learning algorithms into cyber security detection systems also enables them to continuously improve their detection capabilities through continuous learning and adaptation. However, while AI has proven to be an effective tool for cyber security, it is not foolproof, and there is a need for human expertise to analyze and interpret the data generated by AI systems. Therefore, the combination of AI and human expertise in cyber security detection is essential for building robust and reliable systems that can effectively protect against cyber threats.

REFERENCES

TEXTBOOK:

➢ *"Practice Malware Analysis" By:- Michael Sikorski and Andrew Honig, Publisher: No Starch Press, First Edition, February 2012*

➢ *"Threat Modeling" By:- Adam Shostack. Publisher: Wiley, First Edition, April, 2014*

[1] Erwin Adi, Zubair Baig, and Imran A. Khan, "Harnessing Artificial intelligence Capabilities to improve Cyber security", Institute of Electrical and Electronics Engineering, DOI: 10.1109/ACCESS.2020.2968045, Date of publishing: January 20,2020

[2] Nicola Capuano, Giuseppe Fenza, Vincenzo Loia, Claudio Stanzione "Explainable Artificial Intelligence in CyberSecurity", Institute of Electrical and Electronics Engineering, DOI: 10.1109/ACCESS.2020.32004171, Date of publishing: September 5, 2022

[3] Pooja S, Chandrakala C B, Laiju K Raju "Developer's Roadmap to Design Software Vulnerability Detection Model Using Different AI Approaches" Institute of Electrical and Electronics Engineering, DOI: 10.1109/ACCESS.2022.319111, Date of publishing: July 22, 2022.

[4] Hatma Suryotrisongko, Yasuo Musashi, Akio Tsuneda, Kenichi Sugitani "Robust Botnet DGA Detection: Blending XAI and OSINT for Cyber Threat Intelligence Sharing", Institute of Electrical and Electronics Engineering, DOI: 10.1109/ACCESS.2022.3162588, Date of publishing: April 4, 2022.

[5] Kamran Shaukat, Suhuai Luo, Shan Chen "Cyber Threat Detection Using Machine Learning Techniques: A Performance Evaluation Perspective", Commonwealth Scientific and Industrial Research Organization, Australia.

DETAILS OF THE STUDENT

BATCH – 2019-2023

NAME:	ASHWATH D PADUR	CHETHAN S	G S SUDEEP
USN:	1DB19CS017	1DB19CS035	1DB19CS049
PHONE NUMBER:	91087 11119	9035563403	91488 09527
EMAIL ID:	ashwath29may@gmail.com	chethans20012017@gmail.com	sudeepkrishna187@gmail.com
PLACED – IN:	N/A	Johnson Controls Inc.	CSG International
INTERNSHIP	Automata Research Laboratory	Automata Research Laboratory	Seventh Sense
PERMANENT ADDRESS	#S-3, Smaran Jyothi apt, 10th main rd, Sampige layout, Vijayanagar, Bangalore - 560079	#11 & 12, SV Classic Apartment GF 02, 4th Main, Kalyaninagar, Vasthapura, Bangalore - 560061	#40, Saraswati Nilaya, Heravanadu Village and post, Madikeri, Kodagu, 571201