



**VIT<sup>®</sup>**  

---

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

# **DDoS Attack Detection Using Machine Learning**

*A report submitted for the J component final review of*

**INFORMATION SECURITY MANAGEMENT**  
**(CSE3502)**

**By**

**Abhiram Sreekantham - 19BCE2294**

**Submitted to**

**Prof. Vidhya Ramamoorthy**

## **Problem Statement**

### **Idea**

DDOS attacks or distributed denial of service attacks serve as one of the most common attacks in the applications that we use today. Most of the e-commerce and cloud based applications are vulnerable to these types of attacks and these also lead to destruction of businesses and net worth of individuals. We in this project plan to research about and implement machine learning algorithms. Machine learning is the branch of computer science that deals with the learning of computer systems. Hence we can also make systems capable of learning how to detect various types of DDOS attacks. We plan to use a combination of various algorithms or perform some improvements on existing algorithms to achieve optimized result and accuracy. We will first chose the dataset with relevant features and then analyze its features in a way which can be used most optimally (for example: packet transfer time and delay) and then compare the accuracies of different algorithms and then in the end come up with an optimized approach from these algorithm as an innovation in the detection of DDOS attacks.

### **Novelty**

There has been little research conducted on how machine learning algorithms can be applied to cyber attacks such as DDOS attacks, so we plan to extend machine learning applications to cyber security attacks as well. (some algorithms have been implemented, but not with optimised algorithms and solutions). Cyber security attacks have always been challenging to attack and have many a times sent false alarms to the incident response teams due to non proper implementation of inaccuracy of detection. To solve this problem, we have gone through multiple research papers and realized that such type of problems can be tackled by application of artificial intelligence. We will develop our model which scales to the greatest attacks and produces highly accurate results.

## **Literature Survey**

### **[1] Machine Learning DDoS Detection for Consumer Internet of Things Devices**

#### **Issues with previous technology:**

Many Internet of Things (IoT) devices are linking to the Internet, however many of these devices are essentially insecure, leaving the Internet vulnerable to a range of assaults. Botnets like Mirai have utilised vulnerable consumer IoT devices to attack key Internet infrastructure with distributed denial of service (DDoS) assaults.

#### **Methodology:**

For IoT traffic DDoS detection, this article employs a machine learning pipeline that includes data gathering, feature extraction, and binary classification. The features are built to take use of IoT-specific

network behaviours while also taking advantage of network flow parameters including packet length, inter-packet intervals, and protocol. Random forests, K-nearest neighbours, support vector machines, decision trees, and neural networks are among the classifiers we assess for attack detection.

#### **Applications:**

It's the first network anomaly detection framework that focuses on IoT characteristics, and also the first to apply anomaly detection exclusively to IoT botnets at the local network level, to our knowledge.

#### **Pros:**

The test set accuracy of all five methods was more than 0.99. These preliminary findings compel more investigation into machine learning anomaly detection in order to defend networks from vulnerable IoT devices.

#### **Cons:**

We'd also be able to observe how the volume and diversity of IoT traffic affects DoS detection accuracy if we had a larger dataset. We'd also like to try out some new features and more advanced machine learning approaches than those mentioned in this article.

## **[2] DDOS Detection Using Machine Learning Technique**

#### **Issues with previous technology:**

Network infrastructures are subjected to several assaults. Attacks on network availability, confidentiality, and integrity are among them. A distributed denial-of-service (DDoS) assault is a persistent attack that degrades network availability. To carry out such an assault, a command and control (C&C) mechanism is employed.

#### **Methodology:**

In this work, a DDoS assault was carried out with the ping of death approach and identified with the WEKA tool utilising machine learning techniques. This experiment uses the NSL-KDD dataset. The normal and assault samples were classified using the random forest technique. The samples were accurately categorised in 99.76 percent of the time.

#### **Applications:**

Using deep learning and machine learning approaches, a number of academics are focusing on detecting the most common DDoS attacks that have the most impact in the domain of social networking.

#### **Pros:**

Several active DDoS attack detection strategies are reviewed, with a focus on machine learning techniques. There is also a discussion about a list of publicly accessible DDoS tools. The DDoS assault was

carried out using a command-based ping of death approach. The model was trained using the random forest approach, which resulted in 99.76 percent of examples being properly identified.

**Cons:**

No deep learning techniques used for the classification of instances.

### **[3] A DDoS Attack Detection Method Based on Machine Learning**

**Issues with previous technology:**

A DDoS assault is currently one of the most prevalent network attacks. The threat of a DDoS assault is growing increasingly significant as computer and communication technology advances. Some relevant research work has been completed recently, and some progress has been made. However, owing to the wide variety of DDoS attack techniques and the changing amount of attack traffic, no detection solution with adequate detection accuracy has yet been developed.

**Methodology:**

This research provides a machine learning-based DDoS assault detection approach that incorporates two levels: feature extraction and model detection. The DDoS attack traffic attributes with a substantial proportion are recovered in the feature extraction step by comparing the data packages sorted according to rules. The recovered features are employed as machine learning input features in the model discovery stage, and the random forest approach is used to train the attack detection model.

**Applications:**

They employed the anomaly detection approach to model the network data stream based on the header attribute, and the naive Bayesian algorithm to evaluate every arriving data stream in order to assess the message's reasonableness.

**Pros:**

As background traffic grows, the detection model in this research still has a high detection accuracy for the DDOS attack detection findings of the three protocols, and is preferable to the SVM algorithm model.

**Cons:**

This paper uses the common DDoS attack tool to conduct local attacks.

### **[4] Modern Machine Learning for Cyber-defense and Distributed Denial of Service Attacks**

**Issues with previous technology:**

The nature of such DDoS attacks makes them difficult to detect and mitigate. When such DDoS attacks are distributed across a large part of the Internet, and involve hundreds or thousands of attackers, their mitigation often requires advanced algorithmic techniques. Differentiating legitimate user and malicious botnets requires use of complex algorithms.

#### **Methodology:**

This journal proposes the idea of using deep learning as it is an active area of research where new approaches are being developed on an almost daily basis. Having a measure of in-house expertise in such methods is important to be able to discern which algorithms and approaches are applicable in your particular context. In particular, the expertise to develop new algorithms in this domain would not be required, but the ability to faithfully assess the performance of algorithms is of prime importance.

#### **Applications:**

The journal suggests and discusses a variety of approaches that can be used to detect ddos attacks such as deep learning and neural networks and also also lays emphasis on how this can be done using data gathering, testing of approaches and choosing the correct tool for implementation. If we follow all these techniques we can be successful in detecting 99% of the DDOS attacks.

#### **Pros:**

The approach used is an accurate one and well researched one, hence it is optimal in detecting DDOS attacks.

#### **Cons:**

The paper does not talk much about how we can improve accuracy on existing methods or how efficiently other approaches work when compared to deep learning.

### **[5] A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning**

#### **Issues with existing technologies:**

Low-rate denial of service (LR-DDoS) attacks is one of the more challenging denial of service (DoS) attack types to detect, and these attacks are designed to exhaust computing resources on servers. Unlike high-rate distributed DoS (DDoS) attacks, an LR-DDoS attack does not flood the network with high traffic loads. Instead, it carefully triggers specific protocol mechanisms such as TCP's timeout retransmission and congestion control mechanisms, HTTP's keep alive mechanism, to deplete the target's computing resources.

#### **Methodology:**

This journal introduces a new versatile architecture for LR-DDoS attack detection and mitigation in SDN environments using machine learning techniques. Specifically, this architecture comprises an intrusion prevention system (IPS), which will forward the flows to the intrusion detection system (IDS) API. This will allow to determine whether the flow is malicious. The IDS API will identify the flow using one of several previously trained machine learning (ML) models. This API is programming language and framework independent, and hence different programming languages and frameworks can be used to implement and train the AI models. Once the IDS API returns the result, the IPS module running on the controller will process the flow accordingly to the mitigation strategy of the architecture if the flow is determined to be an attack.

#### **Applications:**

The modular architecture will allow system implementers to easily replace or enhance a module, API, or ML model without affecting the rest of the architecture.

System implementers can use any programming language and libraries as needed. Another application is that our IDS is capable of distinguishing between anomalous and normal traffic flows and determining the type of LR-DDoS attack being carried out.

#### **Pros:**

The approach used is innovative one and provides dual layer of protection and checking (both IDS and IPS) and hence is considered much safer than other traditional techniques. (it is also more flexible when compared to others)

#### **Cons:**

The approach has not been thoroughly tested and still needs further research for smooth and effective implementation (using the API).

## **[6] Machine Learning based DDOS Detection**

#### **Issues with Existing Technologies:**

Identifying the DoS attack becomes more complex issue since there are various types of DDos attack strategies exist. Some of the types of the DoS attacks are ICMP flood, SYN flood, IP packet flood etc. Due to the complexity involved and the various features to be analysed, it becomes very difficult to get results with high accuracy.

#### **Methodology:**

By using a Machine Learning way of approach, threat identification is easy and way faster than the traditional methods. This paper uses three ML algorithms Naive Bayesian, KNN and Random Forest. Choosing these three algorithms is that these three algorithms require less attributes and a low amount of training data to run the detection process compared to other machine learning algorithms. Choose the best two result from the three outputs for detection which increases the accuracy of the process and also it needs only a small amount of resource power.

**Applications:**

This project can be used in areas which require high degree of accuracy is needed such as detecting different DDOS techniques. As it combines the benefits of 3 algorithms of machine learning so it can be used in all areas these 3 algorithms are used.

**Pros:**

Accuracy is around 98.5%, which is very high when compared to traditional methods of detection.

**Cons:**

This approach uses 3 algorithms in combination and thus has increased complexity and analysis overhead when compared to other algorithms.

**[7] IoT DDoS Attack Detection Using Machine Learning****Issues with previous technology:**

Many security methods have already been offered to defend the Internet infrastructure against a variety of threats. However, Internet and application security remains an unsolved research problem. For developing a more reliable network, researchers are constantly working on novel network topologies such as HTTP as the narrow waist, Named Data Networking (NDN), programmable networks, and Software-Defined Networking (SDN).

**Methodology:**

The continuous ranked probability score (CRPS) statistical measure and the exponential smoothing (ES) scheme are used in this research to provide a reliable detection mechanism for DOS and DDOS attacks. The CRPS is used in this case to measure the difference between a new observation and the regular traffic distribution. For anomaly identification, the ES technique is used on CRPS readings, which is sensitive in detecting minor changes. Furthermore, in the CRPS-ES technique, abnormalities are detected using a nonparametric decision threshold obtained using kernel density estimation.

**Applications:**

An HTTP flood DDOS attack utilizes what appear to be legitimate HTTP GET or POST requests to attack a web server or application. These flooding DDOS attacks often rely on a botnet, which is a group of Internet-connected computers that have been maliciously appropriated through the use of malware such as a Trojan Horse.

**Pros and cons:**

With the increase in usage of cloud computing to provide computing resources on demand over the internet, so is the need to secure the computing resources to provide the reliable and secure services to its users by maintaining the confidentiality, integrity and authenticity constraints. Since the cloud uses

the internet to provide the services, it has become much more vulnerable to various types of attack. One of the major attack issues faced by the cloud platform is DDOS attack. It is a special type of DOS attack in which an attacker uses the network of infected computers which are actually bots to exhaust the resources of the target system.

## **[8] A Survey of DDos Attacks Using Machine Learning Techniques**

### **Issues with previous technology:**

Distributed denial of service (DDoS) attacks are some of the most devastating attacks against Web applications. A large number of these attacks aim to exhaust the network bandwidth of the server, and are called network layer DDoS attacks. Recently, DDoS attacks have started targeting the application layer. Unlike network layer attacks, these attacks can be carried out with a relatively low attack volume. They also utilize legitimate application layer requests, which makes it difficult for existing defense mechanisms to detect them.

### **Methodology:**

Explore hidden semi-Markov model to describe the browsing behaviors of Web users and apply it to implement the anomaly detection for the application layer DDoS attacks which simulate the Web request behaviors of browser and use HTTP requests to launch attacks.

### **Applications:**

Monitoring the Application-Layer DDoS Attacks for Popular Websites

### **Pros:**

When a DDoS attack is launched, the botnet will attack the target and deplete the application resources. A successful DDoS attack can prevent users from accessing a website or slow it down enough to increase bounce rate, resulting in financial losses and performance issues.

### **Cons:**

Accuracy and speed is slow when compared to traditional methods .

## **[9] DDoS attack Detection Using Machine Learning**

### **Issues with previous technology:**

Although there are now more DDoS protection solutions available than ever before, companies still tend to face a few major hurdles with DDoS defense. From attack complexity to a lack of granular control, the challenges that organizations are up against are as varied as they are difficult. Note that two of the three biggest challenges in 2018 had to do with the attacks themselves, while in 2017 all three had to do with DDoS protection solutions. Hence, we can glean that companies are now happier with their solutions than



they were in 2017 but are still struggling to keep up with the constant growth of DDoS attacks in terms of both size and frequency.

### **Methodology:**

For identifying malicious assaults on cyber systems, a monitoring technique is critical. One of the most difficult security challenges facing network technology is detecting denial of service (DOS) and distributed DOS (DDOS). The continuous ranked probability score (CRPS) statistical measure and the exponentially smoothing (ES) scheme are used in this research to provide a reliable detection mechanism for DOS and DDOS attacks. The CRPS is utilised in this case to measure the difference between a new observation and the regular traffic distribution.

### **Applications:**

Genuine users may be unable to access resources and hence find the information or do the activities they require. Businesses might be unable to complete time-sensitive tasks. It's possible that they'll be tarnished. When an organisation expects a large number of users to access its server or services, attackers frequently prepare attacks. To get the server back up and running as quickly as possible, companies will frequently have to call in employees outside of usual working hours or hire more workers.

### **Pros:**

It saves money to have a DDoS security system that is predictable, consistent, effective, and reliable. It saves money on operating and capital expenditures by not having to outsource its security to a third-party scrubbing centre, hire additional IT security personnel, or buy more bandwidth.

### **Cons:**

It can't entirely remove malicious traffic, and it can't tell the difference between harmful and genuine traffic from the same source.

## **[10] Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models**

### **Issues with previous tech:**

The process and communication capacity of the controller is overloaded when DDoS attacks occur against the SDN controller. Consequently, as a result of the unnecessary flow produced by the controller for the attack packets, the capacity of the switch flow table becomes full, leading the network performance to decline to a critical threshold. In this study, DDoS attacks in SDN were detected using machine learning-based models.

### **Methodology:**

Support Vector Machine (SVM). Naive Bayes Artificial Neural Network (ANN) K-Nearest Neighbors

**Pros:**

In this study, SDN-based detection systems developed for DDoS attacks were analyzed with machine learning systems. In the first proposed approach, by analyzing flow data, algorithms with 98.3% accuracy ensure the detection of attacks without discriminating the type of traffic.

**Cons:**

Among the proposed systems, the second approach proceeds by labeling DDoS attacks as normal traffic and attack traffic.

**Comparative Statement after analyzing research papers:**

We have referred to about 12 research papers, from which 10 have been mentioned above, while we see that traditional machine learning algorithms provide an accuracy of about 85%, hybrid machine learning models can provide an accuracy much greater than that (about 90%). Thus we will try hybrid models of different algorithms on our dataset and then test them against their individual and combined accuracies and then finally propose a hybrid model we find the best performance wise. This will not only increase the accuracy, but also combine the advantages of various algorithms in detecting ddos attacks.

**Dataset**

The dataset we have used for our project is "Ddos attack network logs" by jacob van steyn.

The dataset contains around 2,100,000 labelled network logs from various types of network attacks.

The types of network attacks logged are: UDP-Flood, Smurf, SIDDOS, HTTP-FLOOD, & Normal traffic

Link to the dataset:

<https://www.kaggle.com/datasets/jacobvs/ddos-attack-network-logs>

**Test Bed**

For the execution and testing of our project, we will be using jupyter notebook as a testing and execution environment. It provides us with numerous features such as explicit errors, and also a python compatible environment which makes it easy to implement and test the accuracy of the code used.

## **Expected Result**

We expect to propose a algorithm which provides higher accuracy when compared to traditional algorithm when we finish this project,we will be trying to get an accuracy as high as possible.We plan to use these hybrid algorithms on datasets and achieve better results and thus improve ddos attacks detection which are currently being detected at a rate of 85% accuracy.

## **Architecture**

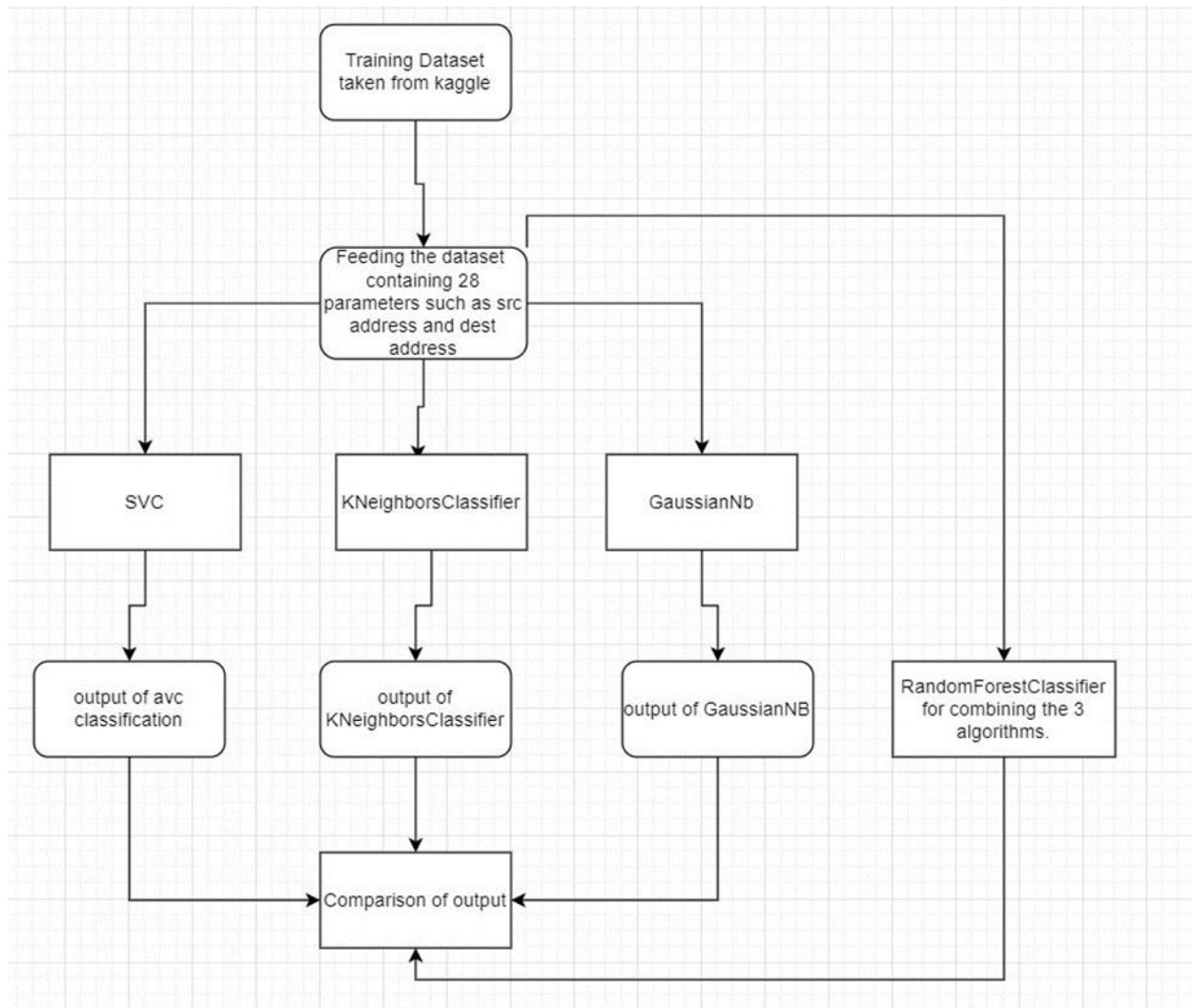
In our project,we have divided out project into 4 modules which can be seen evidently in the high level diagram below.The architecture consists of training,preprocessing and testing and then predicting the output and calculating the accuracy.

We aim to propose a model which is a combination of the three models(gaussian,KNeighborsClassifier,SVC) and then use it for classification of ddos attacks in a dataset which contains 3000 rows.We have also compared the output we got in all the cases.

We propose a system that we hope will be very efficient in detection of ddos attacks and will serve the purpose of defending websites like amazon and other ecommerce and banking websites which are prone to such attacks.

## **High Level Diagram**

It contains the higher level details of the project,such as modules and what are we comparing as an output.All the details have been shown in the form of flowchart/diagram below



## Explanation Of High Level Diagram

**1. Training dataset taken from kaggle:**The dataset containing 28 features is taken from kaggle and trained.

**2. Feeding the dataset containing 28 parameters such as src address and dest address:**The dataset contains 28 parameters such as src address,dest address,source node,destination node and other parameters can be analysed to predict ddos attacks.

**3. Training 3 different models,namely:**

a.SVC(Support Vector Classifier).

b.KNeighbors Classifier

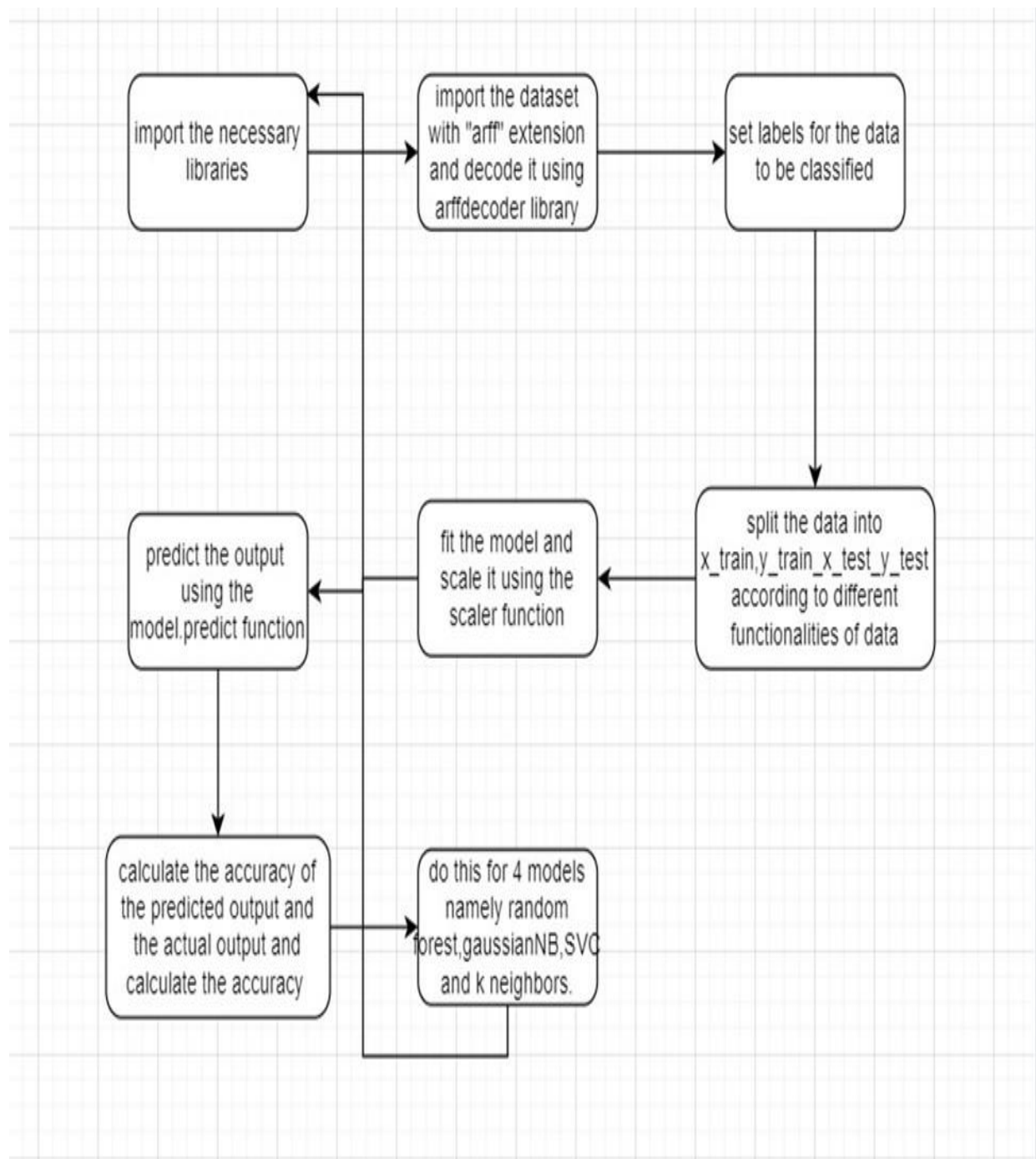
c. Gaussian Nb

**4. Comparing the individual output of these 3 models and calculating the accuracies:** We calculate the output accuracy for all the 3 models and calculate their respective accuracies.

**5. Using the random forest model to combine these 3 algorithms and produce the final output:** The random forest combines all the three above algorithms to give the final result and gives the accuracy of the combined model.

## **Low Level Diagram**

The low level description contains the details of each of the steps which each part of the project undergoes. All the details have been explained in the form of flowchart/diagram below.



### Explanation of Low Level Diagram

**1. Import necessary libraries:** import the necessary libraries such as sklearn and numpy which is needed to train and analyse machine learning data

**2. Import the dataset with "arff" extension and decode it using arffdecoder library:** We have a dataset of 28 features which needs to be imported and used using the arff type decoder.

**3. Set labels for the data to be classified:** Labels tell which class does the data belong to make it easier for the machine learning algorithms to learn how to classify data.

**4. Split the data into x\_train, y\_train, x\_test, y\_test according to different functionalities of data.**

1). X\_train - This includes your all independent variables, these will be used to train the model, also, for example, we have specified the test\_size = 0.4, this means 60% of observations from your complete data will be used to train/fit the model and rest 40% will be used to test the model.

2). X\_test - This is remaining 40% portion of the independent variables from the data which will not be used in the training phase and will be used to make predictions to test the accuracy of the model.

3). y\_train - This is your dependent variable which needs to be predicted by this model, this includes category labels against your independent variables, we need to specify our dependent variable while training/fitting the model.

4). y\_test - This data has category labels for your test data, these labels will be used to test the accuracy between actual and predicted categories.

**6. Fit the model and scale it using scaler function.**

This step comprises of training the model using the data, the model notices the patterns of data and output and learns how to classify data.

**7. Predict the output using the model predict function.**

The output of the input data is to be predicted at this stage for the performance to be analysed.

**8. Calculate the accuracy of the predicted output and the actual output and calculate the accuracy.**

To determine how our algorithms have performed over the given data, we compare the predicted output to the actual output and compare how many correct outputs have been determined by the algorithm.

**9. Do this for 4 models namely random forest, gaussianNB, SVC and k neighbors.**

The same step is repeated for all the 4 algorithms, once for SVC, once for k neighbors and once for gaussianNb and once for the combination of these (i.e., Random Forest).

## Implementation

### Algorithms followed

- **SVC**

SVC and LinearSVC are two functions which belong to the same module `svm` that may be used to generate a linear SVM model in `scikit-learn`. We intuitively decide to use this function since we want to generate an SVM model utilizing a linear kernel and the word `Linear` is also present in the title of the function `LinearSVC`. However, it appears that `SVC` may also be used with the `kernel='linear'` option.

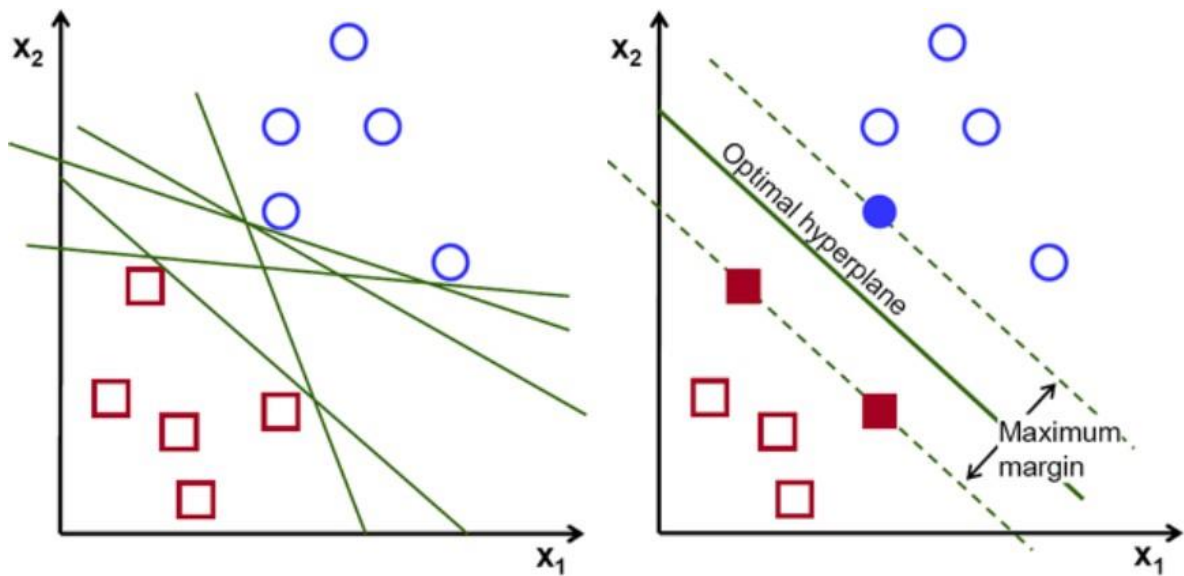
The Linear SVC (Support Vector Classifier) is designed to fit to the data you supply and provide a "best fit" hyperplane that divides or categorises your data. Following that, you may input some characteristics to your classifier to check just what "predicted" class is once you've obtained the hyperplane.

What is Support Vector Machine?

SVMs (support vector machines) are supervised machine learning techniques that may be used for both classification and regression. However, they are most commonly utilised in categorization difficulties. SVMs were initially presented in the 1960s, but they were enhanced around 1990. In comparison to certain other machine learning algorithms, SVMs feature a unique implementation method. They've recently gained a lot of traction due to their capacity to handle both numerical and categorical data.

The support vector machine algorithm's goal is to identify a hyperplane inside an  $N$ -dimensional space, where  $N$  is the number of features, that distinguishes between data points.





Possible Hyperplanes

In multidimensional space, an SVM model is essentially a representation of distinct classes in a hyperplane. SVM will generate the hyperplane in an incremental fashion in order to reduce the error. SVM's purpose is to partition datasets into classes such that maximum marginal hyperplane may be found (MMH).

important concepts in SVM

1. Support Vectors: Datapoints which are proximal to the hyperplane are referred to as support vectors. These data points will be used to define a separating line.
2. Hyperplane: Hyperplane would be a decision plane or space that is split between a group of objects with distinct classes, as seen in the picture above.
3. Margin: The space among two lines on the nearest data points of various classes is known as the margin. The perpendicular distance between the line and the support vectors may be computed. A large margin is seen as a good margin, whereas a small margin is regarded as a poor margin.

- **KneighborsClassifier**

The  $k$  nearest neighbours are represented by the  $K$  in the classifier's name, where  $k$  is an integer value given by the user. As the name implies, this classifier uses training depending on the  $k$  closest neighbours.

Both classification and regression predicting issues can be solved with KNN. However, in the industry, it is more commonly employed in categorization difficulties. We look at three key criteria while evaluating any technique:

1. The output is simple to comprehend.
2. Time to calculate
3. Predictive Ability

The K-Nearest Neighbour method is based on the Supervised Learning approach and is one of the most basic Machine Learning algorithms.

The K-NN method assumes that the new case/data and existing cases are comparable and places the new case in the classification that is most similar to the existing categories.

The K-NN method saves all data available and identifies a data point depending on its similarity to the existing data. This implies that fresh data may be quickly sorted into a well-defined category using the K-NN method.

The K-NN approach may be used for both regression and classification, however it is more commonly utilised for classification tasks.

The K-NN algorithm is a non-parametric algorithm, meaning it makes no assumptions about the underlying data.

It's also known as a lazy learner algorithm since it doesn't learn using the training set right away; instead, it saves the dataset and performs an action on it when it comes time to classify it.

During the training phase, the KNN algorithm simply stores the dataset, and when it receives new data, it classifies it into a classification that is quite equivalent to the original data.

- **GaussianNB**

The Gaussian Nave Bayes classifier, as the name implies, considers that the data for each label is obtained from a basic Gaussian distribution. To execute the Gaussian Nave Bayes method for classification, Scikit-learn includes `sklearn.naive_bayes.GaussianNB`.

A Gaussian Naive Bayes algorithm is a sort of NB algorithm that is unique. When the features contain continuous values, it's employed particularly. It's also expected that all of the characteristics have a gaussian distribution, or a normal distribution.

What is Naïve Bayes algorithm?

The simplest and fastest classification approach known, Naive Bayes, is particularly fit for dealing with massive volumes of data. The Naive Bayes Classification Algorithm has proven to be useful in a variety of applications, including phishing detection, text categorization, sentiment classification, and recommendation engines. The Bayes theory of probability is used to create predictions about unknown classes.

The Bayes Theorem is used to affect Naive Bayes, a simple yet powerful probabilistic classification algorithm in machine learning.

The Bayes theorem is a formula that calculates the conditional probability of any event A occurring if another event B has already occurred. The following is its mathematical formula: –

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

Where,

A and B are two separate occurrences.

The probability of occurrence A if event B has already occurred is  $P(A|B)$ .

The probability of occurrence B if event A has already occurred is  $P(B|A)$ .

The independent probability of A is  $P(A)$ .

The independent probability of B is  $P(B)$ .

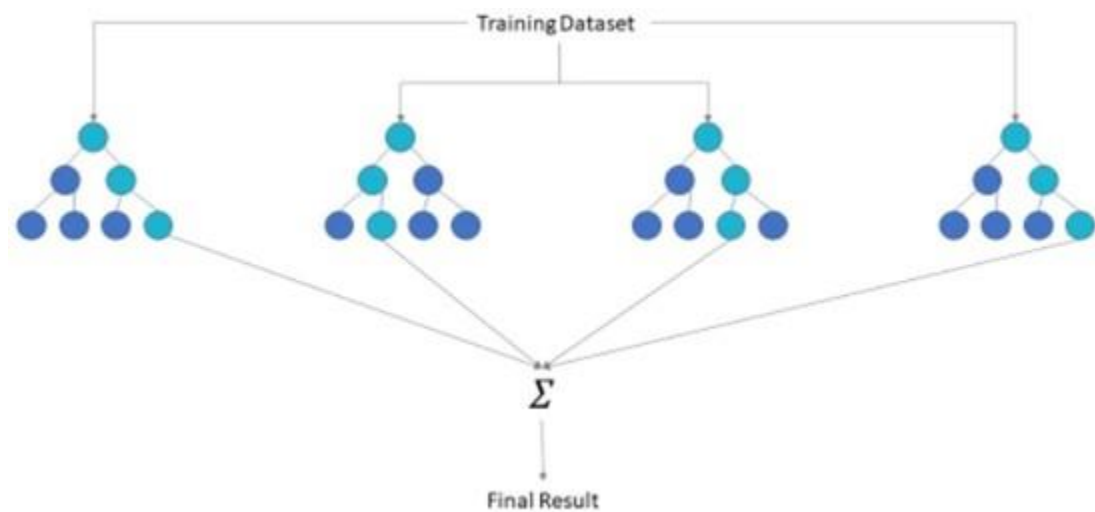
- **RandomForestClassifier**

Random forest is a machine learning technique developed by Leo Breiman and Adele Cutler that mixes the output of numerous decision trees to produce a single outcome. Its popularity is due to its ease of use and adaptability, since it can handle both classification and regression issues.

Because it uses both bagging and feature randomness to produce an uncorrelated forest of decision trees, the random forest technique is an extension of the bagging method. Feature randomization (also known as feature bagging or "the random subspace approach") provides a random selection of features, ensuring minimal correlation among decision trees. A significant distinction between decision trees and random forests is this. Random forests only evaluate a subset of the available feature splits, whereas decision trees consider all of them.

The three primary hyperparameters of random forest algorithms must be established before training. The size of the nodes, the number of trees, and the number of characteristics sampled are all factors to consider. The random forest classifier may then be used to tackle issues involving regression or classification.

The random forest algorithm is composed of a sample of decision trees, and every tree inside the ensemble is made up of a bootstrap sample, which is a data sample obtained from a training set by replacement. One-third of the training sample is set aside as test data, referred to it as the out-of-bag (oob) sample. Using feature bagging, additional instance of randomness will be injected into the dataset, increasing the dataset's variety and decreasing the correlation between decision trees. The forecast will be determined depending upon the type of difficulty.



### Mathematical Model Followed

We followed the Ensemble Mathematical Model,

Ensemble modeling is a useful method for improving your model's performance. It's typically a good idea to use ensemble learning in addition to any other models you're working on. It's a generic machine learning meta method that aims to improve predictive performance by integrating predictions from various models.

To fully comprehend what goes on behind the scenes of an ensemble model, we must first comprehend what causes the error.

The term "bias error" is used to describe how much the anticipated values depart from the real value on average. A large bias error indicates a model that is underperforming and consistently misses crucial trends.

Variance, however, measures how varied predictions built on the same observation are from one another. A model with a large variance will over-fit your training data and underperform on any observation other than training.

$$\text{Error} = (\text{Bias})^2 + \text{Variance} + \text{Irreducible error}$$

## STANDARD ENSEMBLE LEARNING STRATEGIES

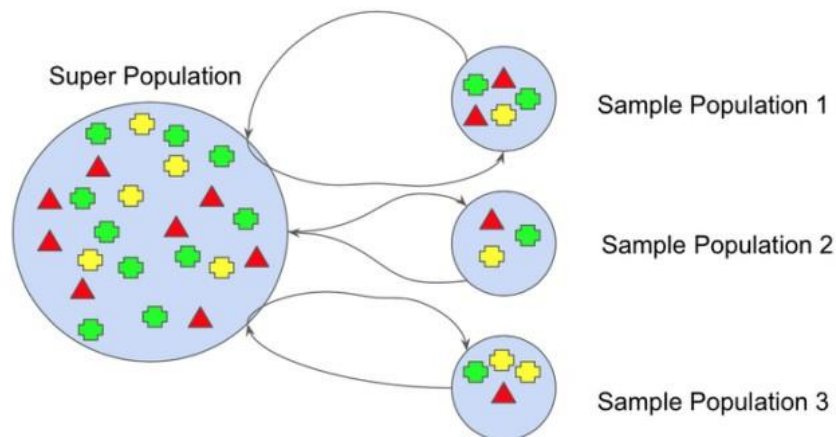
Even though there are practically infinite methods to do this, there are probably three kinds of ensemble learning approaches that are most frequently studied and employed in practice. Their appeal stems from their ease of use and ability to solve a huge spectrum of predictive modeling challenges.

They are known as "standard" ensemble learning procedures because of their widespread use; they are:

### 1. Bagging:

Bootstrap aggregation, or bagging for short, is an ensemble learning approach that varies the training data to find a varied collection of ensemble members.

This usually entails training each model on a distinct sample of the same training dataset using a single machine learning method, almost always an unpruned decision tree. The ensemble members' forecasts are then merged using simple statistics like voting or average.

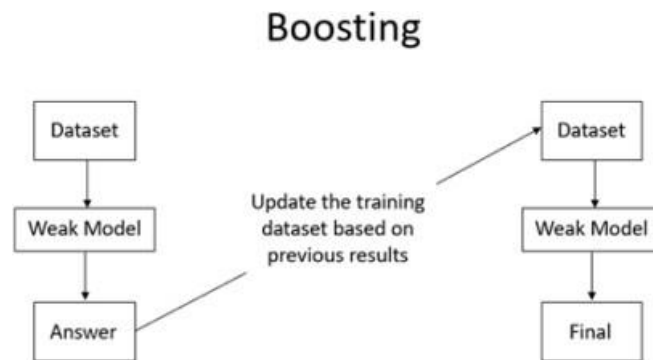


From the original dataset, many subsets are produced by replacing observations. On each of these subsets, a basic model (weak model) is generated. The models are independent of one another and run in parallel. The final forecasts are calculated by combining all of the models' predictions.

### 2. Boosting:

Boosting is a sequential procedure in which each successive model seeks to rectify the prior model's mistakes. The models that follow are reliant on the prior model. From the original dataset, a subset is produced. At first, all data points are given the same weighting. On this

subset, a foundation model is built. This model is applied to the entire dataset in order to create predictions.



The actual and expected numbers are used to determine the errors. Higher weights are assigned to observations that are mistakenly anticipated. (In this case, the three blue-plus points that were misclassified will be given larger weights.) On the dataset, a new model is developed, and predictions are produced. (This model attempts to remedy the prior model's flaws.)

Multiple models are built in the same way, each one fixing the flaws of the preceding model. The mean score of all the models is used to create the final model. As a result, the boosting method combines several weak learners into a single strong learner. Individual models may not perform well throughout the full dataset, but they do so for a portion of it. As a result, each model improves the ensemble's performance.

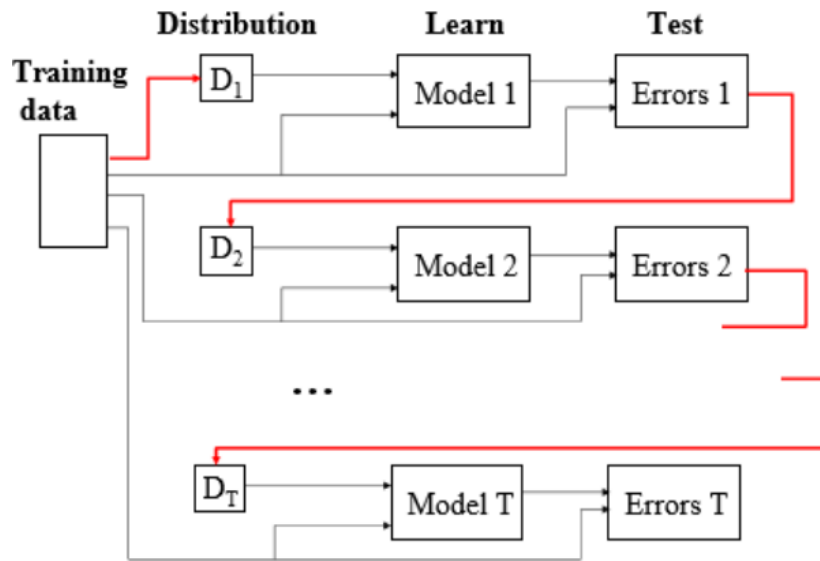
3. Stacking:

Stacked Generalization, or stacking, is an ensemble approach for finding a varied group of members by shifting the model types fit on the training data and combining predictions with a model.

Stacking has its own terminology, with level-0 models referring to ensemble members and level-1 models referring to the model which is used to integrate the forecasts. Although additional levels of models can be utilized, the most frequent strategy is a two-level hierarchy of models. Instead of a single level-1 model, we may have three or five level-1 models and a single level-2 model that integrates level-1 model predictions to generate a forecast.

4. Adaboost:

Adaboost picks a training subset at random at first. It trains the AdaBoost machine learning model repeatedly by picking the training set based on the last training's correct prediction.



It gives incorrectly categorized observations a larger weight so that they have a higher chance of being classified correctly in the following iteration. It also allocates weight to the trained classifier in each iteration based on the classifier's accuracy. The classifier with the highest accuracy will be given more weight. This procedure repeats until all of the training data fits perfectly or until the stated maximum number of estimators is achieved. Perform a "vote" across all of the learning algorithms created to categorize.

## Results and Discussion

### i) Implementation with Coding

```
In [1]: import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
import arff as arf
```

```
In [2]: file=open("final-dataset.arff")
decoder = arf.ArffDecoder()
data=decoder.decode(file,encode_nominal=True)
```

```
In [3]: vals=[val[0:-1] for val in data['data']]
labels=[lab[-1] for lab in data['data']]
```

```
In [5]: da=set(labels)
brac=600
templ=[]
tempd=[]
for i in da:
    coun=0
    while coun<brac:
        for j in range(len(labels)):
            if labels[j]==i:
                templ.append(labels[j])
                tempd.append(vals[j])
                coun+=1
            if coun==brac:
                break
vals=templ
labels=tempd
```

```
In [6]: l=len(vals)
print(l)

3000
```

```
In [7]: X_train,X_test,Y_train,Y_test=train_test_split(vals,labels,stratify=labels,test_size=0.2,random_state=0)
```

```
In [8]: scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
Y_train=np.array(Y_train)
Y_test=np.array(Y_test)
```

```
In [9]: model=SVC(kernel='sigmoid',gamma='auto')
model.fit(X_train,Y_train)
```

```
Out[9]: SVC(gamma='auto', kernel='sigmoid')
```

```
In [10]: y_pred=model.predict(X_test)
```

```
In [11]: print((accuracy_score(y_pred,Y_test))*100,"%")

88.66666666666667 %
```

```
In [12]: model1=KNeighborsClassifier(n_neighbors=5)
model1.fit(X_train,Y_train)
```

```
Out[12]: KNeighborsClassifier()
```

```
In [13]: y_pred1=model1.predict(X_test)
```

```
In [14]: print(accuracy_score(y_pred1,Y_test)*100,"%")

98.16666666666667 %
```

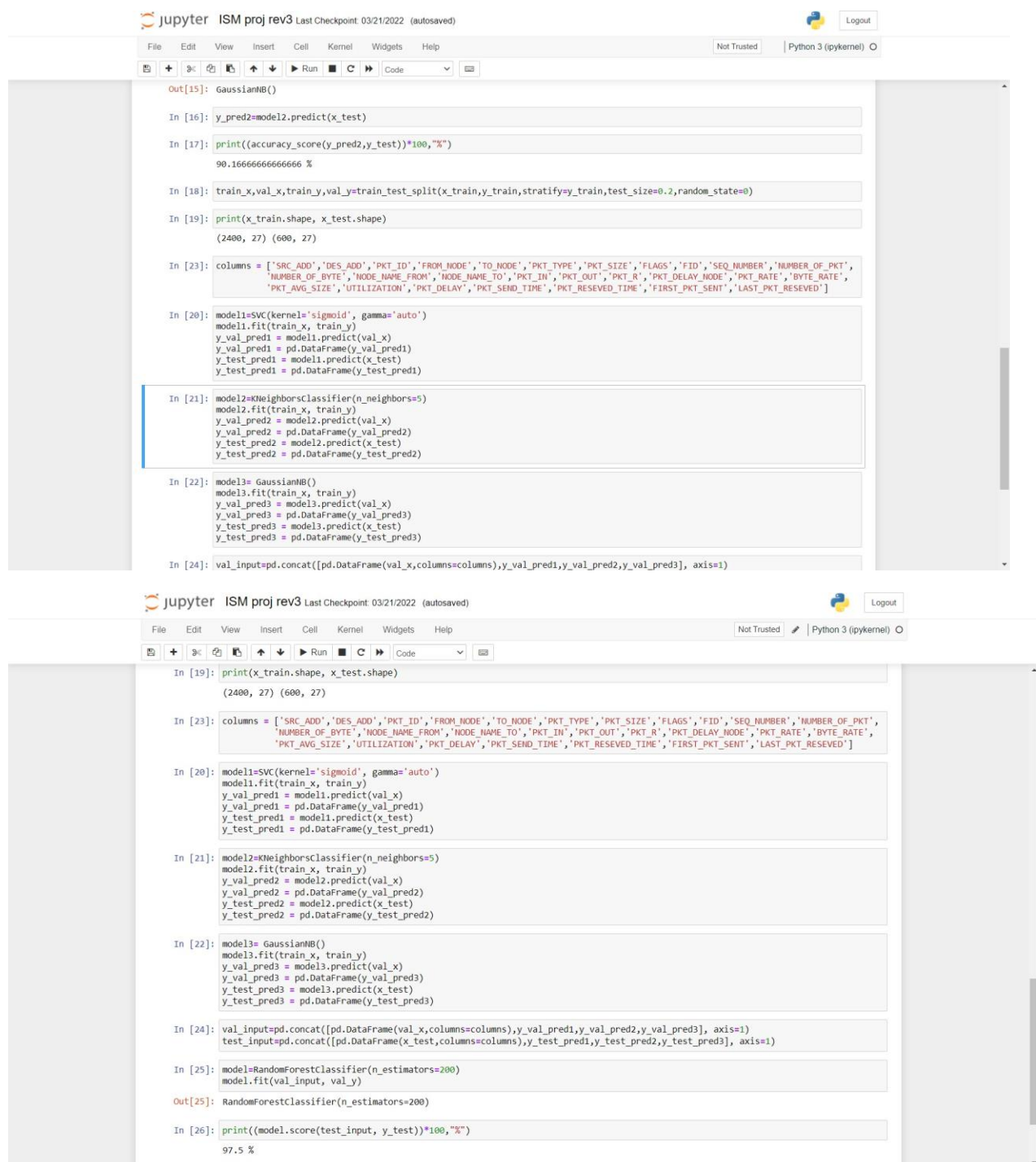
```
In [15]: model2=GaussianNB()
model2.fit(X_train,Y_train)
```

```
Out[15]: GaussianNB()
```

```
In [16]: y_pred2=model2.predict(X_test)
```

```
In [17]: print((accuracy_score(y_pred2,Y_test))*100,"%")
```





The image displays two screenshots of a Jupyter Notebook interface, showing a series of code cells and their outputs. The notebook is titled "ISM proj rev3" and shows the last checkpoint from 03/21/2022. The interface includes a top bar with "Jupyter" branding, a "Logout" button, and a "Python 3 (ipykernel)" indicator. Below the top bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar with icons for file operations, running, and code execution is also present.

The first screenshot shows the following code cells:

- Out[15]: GaussianNB()
- In [16]: `y_pred2=model2.predict(x_test)`
- In [17]: `print((accuracy_score(y_pred2,y_test))*100,"%")`  
Output: 98.16666666666666 %
- In [18]: `train_x,val_x,train_y,val_y=train_test_split(x_train,y_train,stratify=y_train,test_size=0.2,random_state=0)`
- In [19]: `print(x_train.shape, x_test.shape)`  
Output: (2400, 27) (600, 27)
- In [23]: `columns = ['SRC_ADD','DES_ADD','PKT_ID','FROM_NODE','TO_NODE','PKT_TYPE','PKT_SIZE','FLAGS','FID','SEQ_NUMBER','NUMBER_OF_PKT','NUMBER_OF_BYTE','NODE_NAME_FROM','NODE_NAME_TO','PKT_IN','PKT_OUT','PKT_R','PKT_DELAY_NODE','PKT_RATE','BYTE_RATE','PKT_AVG_SIZE','UTILIZATION','PKT_DELAY','PKT_SEND_TIME','PKT_RESEVED_TIME','FIRST_PKT_SENT','LAST_PKT_RESEVED']`
- In [20]: `model1=SVC(kernel='sigmoid', gamma='auto')`  
`model1.fit(train_x, train_y)`  
`y_val_pred1 = model1.predict(val_x)`  
`y_val_pred1 = pd.DataFrame(y_val_pred1)`  
`y_test_pred1 = model1.predict(x_test)`  
`y_test_pred1 = pd.DataFrame(y_test_pred1)`
- In [21]: `model2=KNeighborsClassifier(n_neighbors=5)`  
`model2.fit(train_x, train_y)`  
`y_val_pred2 = model2.predict(val_x)`  
`y_val_pred2 = pd.DataFrame(y_val_pred2)`  
`y_test_pred2 = model2.predict(x_test)`  
`y_test_pred2 = pd.DataFrame(y_test_pred2)`
- In [22]: `model3= GaussianNB()`  
`model3.fit(train_x, train_y)`  
`y_val_pred3 = model3.predict(val_x)`  
`y_val_pred3 = pd.DataFrame(y_val_pred3)`  
`y_test_pred3 = model3.predict(x_test)`  
`y_test_pred3 = pd.DataFrame(y_test_pred3)`
- In [24]: `val_input=pd.concat([pd.DataFrame(val_x,columns=columns),y_val_pred1,y_val_pred2,y_val_pred3], axis=1)`

The second screenshot shows the following code cells:

- In [19]: `print(x_train.shape, x_test.shape)`  
Output: (2400, 27) (600, 27)
- In [23]: `columns = ['SRC_ADD','DES_ADD','PKT_ID','FROM_NODE','TO_NODE','PKT_TYPE','PKT_SIZE','FLAGS','FID','SEQ_NUMBER','NUMBER_OF_PKT','NUMBER_OF_BYTE','NODE_NAME_FROM','NODE_NAME_TO','PKT_IN','PKT_OUT','PKT_R','PKT_DELAY_NODE','PKT_RATE','BYTE_RATE','PKT_AVG_SIZE','UTILIZATION','PKT_DELAY','PKT_SEND_TIME','PKT_RESEVED_TIME','FIRST_PKT_SENT','LAST_PKT_RESEVED']`
- In [20]: `model1=SVC(kernel='sigmoid', gamma='auto')`  
`model1.fit(train_x, train_y)`  
`y_val_pred1 = model1.predict(val_x)`  
`y_val_pred1 = pd.DataFrame(y_val_pred1)`  
`y_test_pred1 = model1.predict(x_test)`  
`y_test_pred1 = pd.DataFrame(y_test_pred1)`
- In [21]: `model2=KNeighborsClassifier(n_neighbors=5)`  
`model2.fit(train_x, train_y)`  
`y_val_pred2 = model2.predict(val_x)`  
`y_val_pred2 = pd.DataFrame(y_val_pred2)`  
`y_test_pred2 = model2.predict(x_test)`  
`y_test_pred2 = pd.DataFrame(y_test_pred2)`
- In [22]: `model3= GaussianNB()`  
`model3.fit(train_x, train_y)`  
`y_val_pred3 = model3.predict(val_x)`  
`y_val_pred3 = pd.DataFrame(y_val_pred3)`  
`y_test_pred3 = model3.predict(x_test)`  
`y_test_pred3 = pd.DataFrame(y_test_pred3)`
- In [24]: `val_input=pd.concat([pd.DataFrame(val_x,columns=columns),y_val_pred1,y_val_pred2,y_val_pred3], axis=1)`  
`test_input=pd.concat([pd.DataFrame(x_test,columns=columns),y_test_pred1,y_test_pred2,y_test_pred3], axis=1)`
- In [25]: `model=RandomForestClassifier(n_estimators=200)`  
`model.fit(val_input, val_y)`
- Out[25]: `RandomForestClassifier(n_estimators=200)`
- In [26]: `print((model.score(test_input, y_test))*100,"%")`  
Output: 97.5 %

## Results in Data:

- Number of Labeled Data Values:

```
In [6]: l=len(vals)
        print(l)
```

3000

- Accuracy of SVC model

```
In [9]: model=SVC(kernel='sigmoid',gamma='auto')
        model.fit(x_train,y_train)
```

Out[9]: SVC(gamma='auto', kernel='sigmoid')

```
In [10]: y_pred=model.predict(x_test)
```

```
In [11]: print((accuracy_score(y_pred,y_test))*100,"%")
```

88.66666666666667 %

- Accuracy of KNeighborsClassifier model

```
In [12]: model1=KNeighborsClassifier(n_neighbors=5)
        model1.fit(x_train,y_train)
```

Out[12]: KNeighborsClassifier()

```
In [13]: y_pred1=model1.predict(x_test)
```

```
In [14]: print(accuracy_score(y_pred1,y_test)*100,"%")
```

98.16666666666667 %

- Accuracy of GaussianNB model

```
In [15]: model2=GaussianNB()  
model2.fit(x_train,y_train)
```

```
Out[15]: GaussianNB()
```

```
In [16]: y_pred2=model2.predict(x_test)
```

```
In [17]: print((accuracy_score(y_pred2,y_test))*100,"%")  
  
90.16666666666666 %
```

- Accuracy of RandomForestClassifier model

```
In [25]: model=RandomForestClassifier(n_estimators=200)  
model.fit(val_input, val_y)
```

```
Out[25]: RandomForestClassifier(n_estimators=200)
```

```
In [26]: print((model.score(test_input, y_test))*100,"%")  
  
97.5 %
```

## Mapping Problem Statement with Existing System and Problem Statement

### Existing Methods

DDoS assaults may be detected in two ways: in-line packet inspection and out-of-band detection using traffic flow record analysis. On-premises or through cloud services, either strategy can be used. Basic in-line DDoS detection capabilities of network devices like load balancers, firewalls, and intrusion prevention systems may have been adequate when DDoS attacks were smaller, but high-volume attacks can overload these devices since they use memory-intensive stateful inspection methods.

In-line detection (and treatment) is currently accomplished mostly using dedicated DDoS mitigation equipment. In the face of larger volume threats, however, they might become expensive and have a limited life cycle. Although these appliances are still essential and important for mitigation since ASIC and network CPU power is required for deep packet inspection when scrubbing traffic, relocating detection out of mitigation devices has become the standard for cost-effectiveness and scale reasons. Out-of-band DDoS detection is carried out by a procedure that gathers flow data from NetFlow, J-Flow, sFlow, and IPFIX-enabled routers and switches, then analyses it to find assaults. The assaults are then mitigated, either manually or automatically, using routing or appliance-based approaches.

The original generation of out-of-band DDoS detection systems relied on single-server software, which was often deployed on rack-mounted server appliances.

Single servers, while better than nothing, lack the computing, memory, and storage capacity required to track large amounts of traffic data over a network. This is especially true when performing dynamic baselining, which necessitates scanning a large quantity of flow data to determine what is typical, then going back days or weeks to determine whether present conditions are abnormal. Single server DDoS detection, whether deployed on-premises or in the cloud, is insufficient to reliably detect today's assaults in a consistent and reliable manner.

**While these are the traditional methods of detection, there are machine learning models which are used to detect ddos attacks which have a accuracy of about 80%. We have proposed a hybrid model as a result which will give a accuracy of about 90% which is much higher when compared to the traditional machine learning models used to predict the attacks.**

## References

- [1] Doshi, R., Aphorpe, N., & Feamster, N. (2018, May). Machine learning ddos detection for consumer internet of things devices. In 2018 IEEE Security and Privacy Workshops (SPW) (pp. 29-35). IEEE.
- [2] Pande, S., Khamparia, A., Gupta, D., & Thanh, D. N. (2021). DDOS detection using machine learning technique. In *Recent Studies on Computational Intelligence* (pp. 59-68). Springer, Singapore.
- [3] Pei, J., Chen, Y., & Ji, W. (2019, June). A ddos attack detection method based on machine learning. In *Journal of Physics: Conference Series* (Vol. 1237, No. 3, p. 032040). IOP Publishing.
- [4] Paffenroth, R. C., & Zhou, C. (2019). Modern machine learning for cyber-defense and distributed denial-of-service attacks. *IEEE Engineering Management Review*, 47(4), 80-85.
- [5] Perez-Diaz, J. A., Valdovinos, I. A., Choo, K. K. R., & Zhu, D. (2020). A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. *IEEE Access*, 8, 155859-155872.
- [6] Priya, S. S., Sivaram, M., Yuvaraj, D., & Jayanthiladevi, A. (2020, March). Machine learning based DDoS detection. In *2020 International Conference on Emerging Smart Computing and Informatics (ESCI)* (pp. 234-237). IEEE.
- [7] Aysa, M. H., Ibrahim, A. A., & Mohammed, A. H. (2020, October). Iot ddos attack detection using machine learning. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)* (pp. 1-7). IEEE.
- [8] Arshi, M., Nasreen, M. D., & Madhavi, K. (2020). A survey of DDOS attacks using machine learning techniques. In *E3S Web of Conferences* (Vol. 184, p. 01052). EDP Sciences.

[9] Li, Q., Meng, L., Zhang, Y., & Yan, J. (2018, September). DDoS attacks detection using machine learning algorithms. In *International Forum on Digital TV and Wireless Multimedia Communications* (pp. 205-216). Springer, Singapore.

[10] Polat, H., Polat, O., & Cetin, A. (2020). Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models. *Sustainability*, 12(3), 1035.