Darknet Traffic Classification Using Machine Learning Techniques

A STUDENT PROJECT REPORT

for

Center for Cyber Intelligence

by

Sahithi Kasim

BTech(Computer Science& Engineering)

G. Narayanamma Institute Of Technology And Science

Under the Guidance of

Prof. Dr Padmavathi Ganapathi

Avinashilingam Institute for Home Science and Higher Education for Women

Acknowledgement

I would like to express my sincere thanks to Avinashilingam Institute for Home Science and Higher Education for Women for providing the working facilities to do the project.

My sincere thanks and gratitude to **Dr. Padmavathi Ganapathi, Dean-School of Physical Sciences and Computational Sciences-Professor**, Avinashilingam Institute for Home Science and Higher Education for Women for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to **Ms. A. Roshini** for her constant guidance, encouragement and moral support throughout the project.

Finally, I would also like to thank all the faculty and staff of Avinashilingam who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Abstract

Darknet traffic classification is playing an important to categorize real-time applications it is an unused address space used in the internet. Analyzing darknet traffic aids in early detection of malware and early monitoring of malware before it outbreaks.To identify Darknet traffic, we used machine learning methods. To provide a better visual representation of the results, a ROC curve is used and a feature selection analysis is used for the better classifier results. The experiments were carried out on the CIC-Darknet2020 dataset. Traffic is divided into two categories: "Benign" and "Darknet,"where"Tor" and "VPN"are considered into"Darknet" category and "Non Tor" and "Non VPN"are considered into"Benign" category. Using several supervised machine learning approaches, like Logistic Regression, Support Vector Machine, Naive Bayes, K-Nearest Neighbors and Decision Tree Classifier an average prediction accuracy of over 99% was achieved.

Table Of Contents

1. Introduction

1.1 Darknet

A darknet is an Internet overlay network that can only be accessed with specific software, configurations, or authorization, and frequently uses a custom communication protocol. “Darknet” this term was originated in 1970’s to describe the networks that are not connected to APRANET (Advanced Research Projects Agency Network) for security reasons. Darknet addresses could connect to ARPANET and receive data, but they didn't show up in network listings and didn't respond to pings or other enquiries. Despite receiving communications from APRANET, they did not respond or recognize them, to remain invisible. It is also known as network telescopes, sinkholes, or black-holes. Social networks (often used for file hosting over a peer-to-peer connection) and anonymity proxy networks (Tor) are mainly two darknet types. Darknet can be used for various reasons, such as:-

- ✓ To protect privacy rights of citizens
- ✓ Computer crime
- ✓ Defending political dissidents against retaliation.
- ✓ News leaks and whistle blowing.
- ✓ Bypassing tight firewall policies or circumventing network censorship and content-filtering technologies.
- ✓ File Sharing.
- ✓ Selling of banned goods on darknet.

The act of spreading or providing access to digital media, such as computer programs, multimedia (audio, pictures, and video), documents, or electronic books, is known as File Sharing. File sharing can be done in a variety of ways. Manual sharing using portable media, centralized servers on computer networks, World Wide Web-based hyperlinked papers, and distributed peer-to-peer networking are all common means of storage, transfer, and dispersion.

P2P computing, often known as peer-to-peer networking, is a distributed application architecture that divides jobs or workloads among peers. Peers are equal-opportunity, equally capable participants in the application. They are supposed to build a node network that is peer-to-peer. Without the requirement for central coordination by servers or reliable hosts, peers make a portion of their resources, such as processing power, disc storage, or network bandwidth, directly available to other network participants. Peers are both resource suppliers and consumers, in contrast to the typical client–server model, which divides resource consumption and supply.

An anonymizer, often known as an anonymous proxy, is a tool that tries to hide your online activity. It's a proxy server computer that functions as a middleman and a privacy shield between a client computer and the rest of the Internet. It connects to the Internet on the user's behalf, masking the client computer's identifying information and protecting the user's personal information. Anonymizers are useful for a variety of reasons, including reducing risk, preventing identity theft, and concealing search records from public disclosure.

TOR and VPN provide users with encrypted entry points and pathways to the darknet. Due to this layered encryption mechanism, darknet users' identities and locations remain anonymous and cannot be monitored. Users' data is routed through a large number of intermediate servers using darknet encryption technology, which conceals users' identities and ensures anonymity. Only a following node in the scheme, which leads to the exit node, may decode the transferred data. The sophisticated mechanism makes duplicating the node path and decrypting the information layer by layer nearly impossible. Websites cannot trace the geo-location and IP addresses of their users because to the high level of encryption, and users are unable to obtain this information about the host. As a result, darknet users' communication is highly encrypted, allowing them to converse, blog, and share confidential files.

1.2 VPN

A virtual private network (VPN) connects a private network to a public network, allowing users to send and receive data as if their computers were directly connected to the private network. The functionality, security, and management of a VPN may be beneficial to applications operating over it. It allows telecommuting employees access to resources that are not available on the public network. Encryption is often used, but it is not a requirement for a VPN connection. Dedicated circuits or tunneling techniques are used to build a virtual point-to-point connection over existing networks, resulting in a VPN. Some of the benefits of a wide area network can be obtained using a VPN accessible via the public Internet (WAN). The resources provided within the private network can be accessed remotely from the user's perspective. VPN provides confidentiality, authentication and integrity to the transmitted messages. It is mainly classified into 3 categories:-

- ✓ **Remote access:-** Connecting a PC to a local area network is equivalent to a host-to-network configuration. This type allows users to connect to a corporate network, such as an intranet. This could be used by telecommuting workers who require access to private resources, or by mobile workers who need access to critical technologies without exposing them to the public Internet.
- ✓ **Site-to-Site:-** Two networks are connected through a site-to-site arrangement. This setup connects a network to a data centre installation through geographically dispersed offices or a group of offices. A distinct intermediary network, such as two IPv6 networks connected across an IPv4 network, could be used for the interconnecting link.
- ✓ **Extra-net-Based Site-to-Site:-** The terms intranet and extra-net are used to define two separate use cases in site-to-site deployments. An intranet site-to-site VPN connects sites that are all part of the same organization, whereas an extranet site-to-site VPN connects sites that are all part of different companies.

1.3 TOR

The Onion Router (TOR) is a free and open-source technology that allows users to communicate anonymously. It hides a user's location and usage from anyone doing network surveillance or traffic analysis by routing Internet traffic over a free, global volunteer overlay network with over 6,000 relays. Tor makes it more difficult to track an individual's online behavior. Tor's purpose is to safeguard its users' personal privacy, as well as their freedom and capacity to communicate in confidence, by preventing their Internet activity from being monitored.

Tor allows its users to access the Internet, communicate, and send instant messages while remaining anonymous, and it is used by a wide range of people for both licit and illegal objectives. Tor isn't intended to be a perfect solution to the problem of online anonymity. Tor isn't meant to entirely wipe your tracks; rather, it's meant to make it harder for websites to track your actions and data back to you. Tor is also used for nefarious purposes. Privacy protection or censorship evasion, as well as the spread of child abuse content, drug transactions, or malware distribution, are all examples. "Overall, on an average country/day, 6.7 percent of Tor network users connect to Onion/Hidden Services that are disproportionately used for unlawful activities," according to one assessment. It is implemented in many programming languages in different ways like:-

- ✓ Tor Browser
- ✓ Firefox/Tor browser attack
- ✓ Tor Messenger
- ✓ Third-party applications
- ✓ Security-focused operating systems

Tor browser has three levels of security, which can be found under the Security Level (the small grey shield at the top-right of the screen) icon > Advanced Security Settings, depending on the demands of the user. Several extra layers of protection are available to a user, in addition to encrypting data and constantly changing an IP address over a virtual circuit comprised of successive, randomly selected Tor relays:-

- ➢ Standard (default) - all browser features are enabled at this security level.
  - ✧ This level offers the most useful experience while also offering the least security.
- ➢ Safer – the following changes apply at this security level:
  - ✧ On non-HTTPS sites, JavaScript is disabled.
  - ✧ Performance optimizations are disabled for sites that use JavaScript. Some websites' scripts may take longer to load.
  - ✧ Some arithmetic equation display techniques are disabled.
  - ✧ Click-to-play audio and video (HTML5 media), as well as WebGL.
- ➢ Safest - At this level of security, the following additional changes apply:
  - ✧ On all sites, JavaScript is turned off by default.
  - ✧ The use of several fonts, icons, math symbols, and images is restricted.
  - ✧ Click-to-play audio and video (HTML5 media), as well as WebGL.

1.4 Importance of machine learning in cyber attacks detection

The ongoing tracking and correlation of millions of external and internal data points across an organization's infrastructure and users is required by the cyber threat landscape. It is just not possible to manage this volume of data with with a small group of individuals. Machine learning excels in this area because it can discover patterns and forecast dangers in large data sets at machine speed. Cyber teams can quickly discover threats and isolate instances that require further human study by automating the analysis.

Machine learning identifies vulnerabilities by continuously monitoring network behaviour for anomalies. To detect major occurrences, machine learning engines process huge volumes of data in near real time. Insider threats, undiscovered malware, and policy infractions can all be detected using these methods. Machine learning can help users avoid connecting to harmful websites by predicting "bad neighborhoods" online. Machine learning examines Internet behaviour to detect attack infrastructures that are ready to respond to existing and emerging threats. Algorithms can detect malware that has never been seen before and is attempting to run on endpoints. It detects new hazardous files and activity based on known malware's features and behavior. To identify dangers and risks in cloud apps and platforms, machine learning can analyse suspicious cloud app login activity, detect location-based abnormalities, and undertake IP reputation analysis. By examining encrypted traffic data pieces in common network telemetry, machine learning can detect malware in encrypted traffic. Machine learning algorithms, rather than decrypting, identify dangerous patterns to uncover risks buried behind encryption.

1.5 Supervised Learning

The machine learning task of learning a function that transforms an input to an output based on example input-output pairs is known as supervised learning. It uses labeled training data and a set of training examples to infer a function. Each example in supervised learning is made up of an input object (usually a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm examines the training data and generates an inferred function that can be applied to fresh cases. The algorithm will be able to accurately determine the class labels for unseen examples in the best case scenario. This necessitates a "reasonable" generalization of the training data to unknown scenarios by the learning algorithm. The generalization error is a statistical metric for determining an algorithm's statistical quality.

Steps involved in supervised learning:-

- ✧ Determine the type of training dataset you'll be using.
- ✧ Gather the training data that has been labeled.
- ✧ Divide the data into three sections: training, testing, and validation.
- ✧ Determine the training dataset's input characteristics, which should contain enough information for the model to accurately predict the output.
- ✧ Choose an appropriate algorithm for the model, such as a support vector machine or a decision tree.
- ✧ Use the training dataset to run the algorithm. Validation sets, which are a subset of training datasets, are sometimes required as control parameters.
- ✧ By giving the test set, you may assess the model's correctness. If the model correctly predicts the outcome, then our model is accurate.

It is of two types:-

1) Classification
2) Regression

Challenges involved in supervised learning:-

- ✧ Irrelevant input features
- ✧ Data preparation and pre-processing
- ✧ When impossible, improbable, and incomplete values are used as training data, accuracy falls.

Advantages:-

- ✧ In Machine Learning, supervised learning allows you to collect data or generate a data output based on prior experience.
- ✧ Using experience, it assists you in optimizing performance criteria.
- ✧ You can use supervised machine learning to solve a variety of real-world compute challenges.

Disadvantages:-

- ✧ If your training set lacks the types of examples you'd like to see in a class, your decision boundary may be over trained.

- While training the classifier, you must select a large number of good samples from each class.
- It might be difficult to categorize large amounts of data.
- Computation time is required for supervised learning training.

1.5.1 Classification

The technique of predicting the class of given data points is known as classification. Targets, labels, and categories are all terms used to describe classes. The task of approximating a mapping function (f) from input variables (X) to discrete output variables is known as classification predictive modelling (y). Classification is a type of supervised learning in which the input data is also delivered to the objectives. Classification has numerous uses in a variety of fields, including credit approval, medical diagnosis, and target marketing.

It is divided into two categories:

i. Lazy learners:-

Lazy learners simply save the training data and wait for the testing data. When this happens, classification is performed using the most closely related data from the stored training data. It has less training time but more predicting time.

Ex. K-Nearest Neighbors

ii. Eager learners:-

Before receiving data for classification, eager learners develop a classification model based on the available training data. It must be able to commit to a single hypothesis that encompasses all possible instances. Eager learners take a long time to train and a short time to predict due to the model's structure.

Ex. Decision Tree, Naive Bayes

Some of the classification methods are:-

- ✓ Support Vector Machine (SVM)
- ✓ Naive Bayes
- ✓ Decision Trees
- ✓ K-Nearest Neighbor
- ✓ Random Forest

1.5.2 Regression

The technique of assessing the relationship between a dependent variable and independent factors is known as regression analysis, i.e. fitting a function from a selected family of functions to the sampled data while accounting for some inaccuracy. Regression analysis is one of the most basic strategies for prediction in the field of machine learning. You fit a function to the available data and try to forecast the outcome in the future or for hold-out data points using regression. This function-fitting is beneficial in two ways.

- ➢ Within your data range, you can estimate missing data (Interpolation).
- ➢ Outside of your data range, you can make educated guesses about future data (Extrapolation).

Some of the regression methods are:-

- ✓ Linear Regression
- ✓ Logistical Regression
- ✓ Polynomial Regression

1.6 Unsupervised Learning

Unsupervised learning is a machine learning technique in which models are not supervised using a training dataset, as the name suggests. Models, on the other hand, use the data to uncover hidden patterns and insights. It is comparable to the learning that occurs in the human brain while learning new things. Unsupervised learning can be defined as "a sort of machine learning in which models are taught using unlabeled datasets and then allowed to act on that data without supervision."

Because, unlike supervised learning, we have the input data but no corresponding output data, unsupervised learning cannot be immediately applied to a regression or classification task. Unsupervised learning aims to uncover a dataset's underlying structure, categorize data based on similarities, and display the dataset in a compact fashion.

Uses of Unsupervised Learning:-

- ➢ Unsupervised learning is beneficial for extracting relevant information from data.

- Unsupervised learning is analogous to how a human learns to think via their own experiences, bringing it closer to true AI.
- Because unsupervised learning works with unlabeled and uncategorized data, it is more important.
- In the real world, we don't always have input data that corresponds to output data, hence we require unsupervised learning to handle these problems.

Working Of Unsupervised Learning:-

We used unlabeled input data, which means it wasn't categorized and didn't have any associated outputs. Now, the machine learning model is fed this unlabeled input data in order to train it. It will first analyse the raw data in order to uncover hidden patterns, and then use appropriate algorithms such as k-means clustering, Decision tree, and so on.

After applying the appropriate method, the algorithm splits the data objects into groups based on their similarities and differences.

It is of two types:-

1. Clustering: Clustering is a way of organizing things into clusters so that those with the most similarities stay in one group while those with less or no similarities stay in another. Cluster analysis identifies commonalities among data objects and classifies them according to the presence or absence of such commonalities.
2. An association rule is an unsupervised learning strategy that is used to discover links between variables in a large database. It identifies the group of items that appear in the dataset together. The association rule improves the effectiveness of marketing strategies. People who buy X item are more likely to buy Y item as well.
   Ex:- Market Basket Analysis

Some of the unsupervised learning methods are:-

- ✓ K-means clustering
- ✓ KNN (k-nearest neighbors)
- ✓ Hierarchical clustering

- ✓ Anomaly detection
- ✓ Neural Networks
- ✓ Principle Component Analysis
- ✓ Independent Component Analysis
- ✓ Apriori algorithm
- ✓ Singular value decomposition

Advantages:-

- ➢ Unsupervised learning is utilized for more complex problems than supervised learning because there is no labeled input data in unsupervised learning.
- ➢ Unsupervised learning is preferred because unlabeled data is easier to obtain than labeled data.

Disadvantages:-

- ➢ Because it lacks a comparable output, unsupervised learning is inherently more challenging than supervised learning.
- ➢ Because the input data is not labeled and algorithms do not know the exact output in advance, the result of an unsupervised learning method may be less accurate.

Challenges:-

- ➢ Due to the large amount of training data, there is a significant level of computational complexity.
- ➢ Longer periods of training
- ➢ There is a greater chance of getting erroneous results.
- ➢ Validation of output variables requires human intervention.
- ➢ The basis on which data was grouped was not transparent.

1.7 Supervised Learning Vs Unsupervised Learning

The usage of labeled datasets is the key difference between the two methodologies. Simply put, supervised learning algorithms use labeled input and output data, whereas unsupervised learning algorithms do not.

The algorithm “learns” from the training dataset by iteratively making predictions on the data and adjusting for the correct answer in supervised learning. While supervised learning models are more accurate than unsupervised learning models, they do necessitate human interaction to properly identify the data. A supervised learning model, for example, can forecast the length of your commute based on the time of day, weather conditions, and other factors.But first, you'll have to teach it that driving in rainy weather takes longer.

Unsupervised learning models, on the other hand, function independently to uncover the structure of unlabeled data. It's worth noting that validating output variables still necessitates human intervention.

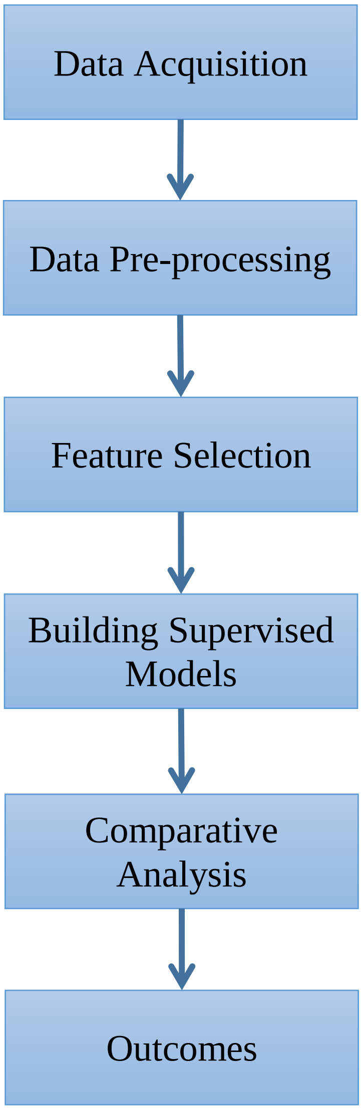
Goals:- The purpose of supervised learning is to predict the results of fresh data. You know exactly what to expect from the start. The purpose of an unsupervised learning algorithm is to derive insights from enormous amounts of new data. What is unusual or interesting from the dataset is determined by machine learning.

Applications:- Spam detection, sentiment analysis, weather forecasting, and pricing forecasts are just a few of the applications for supervised learning models. Unsupervised learning, on the other hand, is well suited to anomaly detection, recommendation engines, customer personas, and medical imaging.

Complexity:- Supervised learning is a straightforward machine learning method that is usually calculated using languages like R or Python. You'll need robust tools for working with vast amounts of unclassified data in unsupervised learning. Because they require a large training set to obtain the desired results, unsupervised learning models are computationally complex.

Drawbacks:- Training supervised learning models takes time, and the labels for input and output variables require knowledge. Meanwhile, unless human intervention is used to evaluate the output variables, unsupervised learning algorithms might produce radically erroneous findings.

2. Methodology



2.1 Stage 1

2.1.1 Data Acquisition

The term "data acquisition" refers to the process of gathering data from relevant sources before it is stored, cleaned, preprocessed, and used in other methods. It is the process of gathering important business data, translating it into the appropriate business form, and loading it into the appropriate system.

It is categorized into 3 main segments:-

1. Data Exploration:- Data discovery is the initial step in acquiring data. When indexing, distributing, and searching for new datasets available on the web and incorporating data lakes, this is a critical step. There are two parts to it: searching and sharing. To begin, the data must be categorized or indexed, then published for sharing via one of the various collaborative platforms available.
2. Data Augmentation:- Data augmentation is the next step in the data collecting process. We are essentially enriching the existing data by adding more external data in the context of data acquisition. Augment is to make something greater by adding to it, therefore we are essentially enriching the existing data by adding more external data. Using pre-trained models and embeddings to expand the number of features to train on is frequent in deep and machine learning.
3. Data Generation:- The data is created, as the name implies. If we don't have enough and don't have access to any external data, we can create the datasets manually or automatically. Crowd-sourcing is a common method for manually collecting data, in which people are assigned tasks to collect the information needed to create a dataset. Automatic methods for creating synthetic datasets are also available. Also, where there is data accessible but missing values that need to be imputed, the data production process can be viewed as data augmentation.

2.1.2 Collection Of Datasets

VPN and Tor applications are combined in the CICDarknet2020 dataset to detect and characterize the genuine representation of darknet traffic by combining two public datasets, namely ISCXTor2016 and ISCXVPN2016, to build a complete darknet dataset encompassing Tor and VPN traffic, respectively. At the first layer, a two-layered method is employed to generate benign and darknet traffic. The second layer of the darknet traffic is made up of Audio-Stream, Browsing, Chat, Email, P2P, Transfer, Video-Stream, and VOIP. Table 1 lists the different types of darknet traffic and the applications that are used to generate it.

Table 1: Darknet Network Traffic Details

Traffic Category	Applications used
Audio-Stream	Vimeo and Youtube
Browsing	Firefox and Chrome
Chat	ICQ, AIM, Skype, Facebook and Hangouts
Email	SMTPS, POP3S and IMAPS
P2P	uTorrent and Transmission (BitTorrent)
Transfer	Skype, FTP over SSH (SFTP) and FTP over SSL (FTPS) using Filezilla and an external service
Video-Stream	Vimeo and Youtube
VOIP	Facebook, Skype and Hangouts voice calls

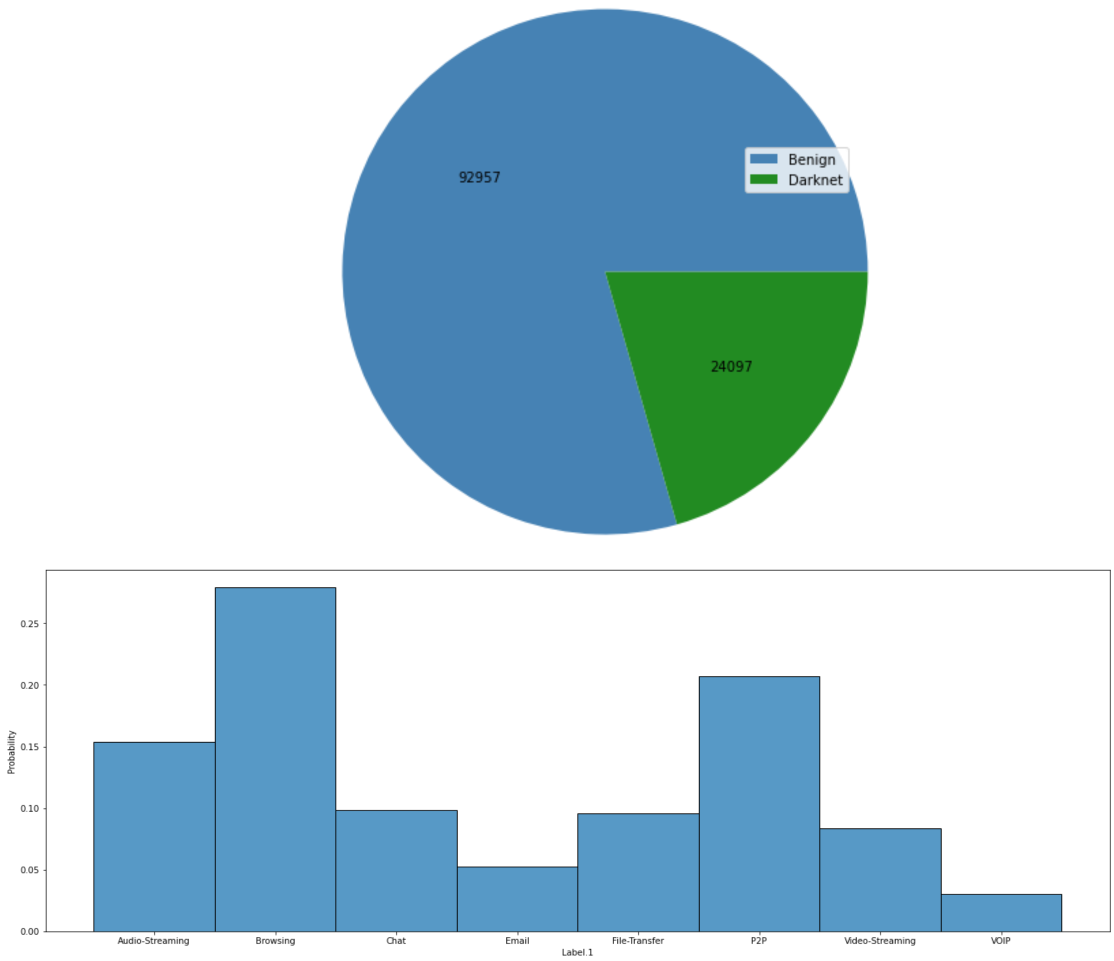


Figure 1:- Dataset details

Table 2 : Dataset Description

S.No	Attribute	Description
1.	Flow ID	It is a combination of Src IP, Dst IP, Src Port, Dst Port and Protocol.
2.	Src IP	It is the IP packet field, which provides the IP address of the workstation it came from.
3.	Src Port	The source port identifies the process that sent the data and are contained in the first header word of each TCP segment and UDP packet.
4.	Dst IP	It is the IP address of the workstation to which it is addressed is provided in the IP packet field.
5.	Dst Port	The destination port identifies the process that is to receive the data and are contained in the first header word of each TCP segment and UDP packet.
6.	Protocol	It is a port number, it is a way to identify a specific process to which an internet or other network message is to be forwarded when it arrives at a server.
7.	Timestamp	It is a sequence of characters or encoded information identifying when a certain event occurred, usually giving date and time of day, sometimes accurate to a small fraction of a second.
8.	Flow Duration	It is the basic time unit used in preparing a flow-duration curve will greatly affect its appearance.
9.	Total Fwd Packet	It tells us about the total number of packets were sent from source port.
10.	Total Bwd Packet	It tells us about the total number of packets were received to destination port.

11.	Total Length of Fwd Packet	Total length of all forward packets
12.	Total Length of Bwd Packet	Total length of all backward packets
13.	Fwd Packet Length Max	Maximum of forward packets length
14.	Fwd Packet Length Min	Minimum of forward packets length
15.	Fwd Packet Length Mean	Mean of forward packets length
16.	Fwd Packet Length Std	Standard deviation of forward packets length
17.	Bwd Packet Length Max	Maximum of backward packets length
18.	Bwd Packet Length Min	Minimum of backward packets length
19.	Bwd Packet Length Mean	Mean of backward packets length
20.	Fwd Packet Length Std	Standard deviation of backward packets length
21.	Flow Bytes/s	Number of bytes sent per second
22.	Flow Packets/s	Number of packets sent per second
23.	Flow IAT Mean	Mean of packets flow inter arrival time
24.	Flow IAT Std	Standard deviation of packets flow inter arrival time
25.	Flow IAT Min	Minimum of packets flow inter arrival time
26.	Fwd IAT Total	Total forward inter arrival time
27.	Fwd IAT Mean	Mean of forward inter arrival time
28.	Fwd IAT Std	Standard deviation of forward inter arrival time
29.	Fwd IAT Max	Maximum of forward inter arrival time
30.	Fwd IAT Min	Minimum of forward inter arrival time
31.	Bwd IAT Total	Total backward inter arrival time
32.	Bwd IAT Mean	Mean of backward inter arrival time
33.	Bwd IAT Std	Standard deviation of backward inter arrival time
34.	Bwd IAT Max	Minimum of backward inter arrival time
35.	Bwd IAT Min	Minimum of backward inter arrival time
36.	Fwd PSH Flags	If the Fwd PSH Flag is set to 1 then all data in the buffer will be pushed to receiver
37.	Bwd PSH Flags	If the Bwd PSH Flag is set to 1 then all data

		in the buffer will be pushed to sender
38.	Fwd URG Flags	If the Fwd URG Flag is set to 1 then all urgent data in the buffer will be sent to receiver
39.	Bwd URG Flags	If the Bwd URG Flag is set to 1 then all urgent data in the buffer will be sent to sender
40.	Fwd Header Length	Header length of forward packet
41.	Bwd Header Length	Header length of backward packet
42.	Fwd Packets/s	Number of forward packets per second
43.	Bwd Packets/s	Number of backward packets per second
44.	Packet Length Min	Minimum of packet lengths
45.	Packet Length Max	Maximum of packet lengths
46.	Packet Length Mean	Mean of packet lengths
47.	Packet Length Std	Standard deviation of packet lengths
48.	Packet Length Variance	Variance of packet lengths
49.	FIN Flag Count	The FIN flag indicates the end of data transmission to finish a TCP connection. It is the count of all FIN flags.
50.	SYN Flag Count	To start a TCP connection, the SYN flag synchronizes sequence numbers. It is the count of all SYN flags.
51.	RST Flag Count	When a segment comes that does not fit the criteria for a referenced connection, the RST flag is set. It is the count of all the RST flags.
52.	PSH Flag Count	It is the count which tells the receiver to process these packets immediately rather than buffering them.
53.	ACK Flag Count	It's the count which used to acknowledge packets that the host has successfully received.
54.	URG Flag Count	It is the count which is used to inform a

		receiving station that certain data within a segment is urgent and should be prioritized.
55.	CWE Flag Count	It's the count of CWE flag
56.	ECE Flag Count	It is the count which is used as a signal by sender to reduce the transmission rate.
57.	Down/Up Ratio	It is the ratio of down and up
58.	Average Packet Size	It is the average size of all the packets
59.	Fwd Segment Size Avg	Average size of the forward segment bits
60.	Bwd Segment Size Avg	Average size of the backward segment bits
61.	Fwd Bytes/Bulk Avg	It is the ratio of Forward bytes by average bulk size
62.	Fwd Packet/Bulk Avg	It is the ratio of Forward packets by average bulk size
63.	Fwd Bulk Rate Avg	It is the average of forward bulk rate
64.	Bwd Bytes/Bulk Avg	It is the ratio of Backward bytes by average bulk size
65.	Bwd Packet/Bulk Avg	It is the ratio of Backward packets by average bulk size
66.	Bwd Bulk Rate Avg	It is the average of backward bulk rate
67.	Subflow Fwd Packets	It is the subflow of forward packets
68.	Subflow Fwd Bytes	It is the subflow of forward bytes
69.	Subflow Bwd Packets	It is the subflow of backward packets
70.	Subflow Bwd Bytes	It is the subflow of backward bytes
71.	Fwd Init Win Bytes	It is the forward init win bytes
72.	Bwd Init Win Bytes	It is the backward init win bytes
73.	Fwd Act Data Pkts	It is the active data packets which are forwarded
74.	Fwd Seg Size Min	It is the minimum forward segment size
75.	Active Mean	Mean of seconds in which the flow has been active
76.	Active Std	Standard deviation of seconds in which the flow has been active
77.	Active Max	Maximum of seconds in which the flow has

		been active
78.	Active Min	Minimum of seconds in which the flow has been active
79.	Idle Mean	Mean of seconds in which the flow has been idle
80.	Idle Std	Standard Deviation of seconds in which the flow has been idle
81.	Idle Max	Maximum of seconds in which the flow has been idle
82.	Idle Min	Minimum of seconds in which the flow has been idle
83.	Label	It represents the networks they have used like Tor, Non-Tor, VPN and NonVPN.
84.	Label.1	It represents the network's type of traffic like Browsing, Email, Chat, Audio-Streaming, Video-Streaming, File Transfer, Voice Over.

2.2 Stage 2

2.2.1 Data Pre-processing

The transformations we apply to our data before feeding it to the algorithm are referred to as pre-processing. Data preprocessing is a method for converting unclean data into a clean data set, i.e anytime data is received from various sources, it is collected in raw format, which makes analysis impossible.

Need of Data Preprocessing

- The data must be formatted properly in order to achieve better outcomes from the used model in Machine Learning applications.
- The data set should be organized in such a way that it can run many Machine Learning and Deep Learning algorithms in parallel and choose the best one.

2.3 Stage 3

2.3.1 Feature Selection

Feature selection is the process of selecting the features that contribute the most to the prediction variable or output that you are interested in, either automatically or manually. The presence of irrelevant characteristics in your data can reduce model accuracy and cause your model to train based on irrelevant features.

Benefits of performing feature selection

- ✓ Models are simplified to make them easier to interpret for researchers and users.
- ✓ a shorter training period
- ✓ to avoid the dimensionality curse
- ✓ to make data more compatible with a learning model class
- ✓ Inherent symmetries in the input space are encoded.

Some of the feature selection techniques are:-

1. Filter Method:- This method employs the variable ranking strategy to choose the variables for ordering, and the features chosen are unaffected by the classifiers utilized. When we talk about ranking, we're talking about how valuable and crucial each attribute is for classification. As a pre-processing step, it basically picks subsets of variables independent of the predictor. Prior to classification, the ranking approach can be used to filter out the less important features in filtering. It performs feature selection as a pre-processing phase that does not use an induction approach.

   Some of the filter methods are:-

   a. Chi-Square Test:- This method is used to test the independence of two events in general. We can acquire the observed count and the predicted count from a dataset for two occurrences, and this test assesses how much both counts are derivate from one other.
   b. Variance Threshold:- This method of feature selection eliminates any features whose variance falls below a certain threshold. In general, it eliminates all zero-variance characteristics, which are those that have the same value across all samples.

c. Information Gain:- Information gain (IG) refers to the amount of information a feature provides about a class. As a result, we can figure out which attribute in a set of training features is the most useful for distinguishing between the classes to be lean.

2. Wrapper Method:- The learning machine of interest is used as a black box in this method to score subsets of variables based on their prediction capability. The induction technique is illustrated with a collection of training cases in the above picture, where each instance is described by a vector of feature values and a class label in supervised machine learning. The induction algorithm, often known as the black box, is used to create a classifier that may be used to classify data. The feature subset selection technique is used as a wrapper around the induction process in the wrapper approach. One of the most significant disadvantages of this method is the large number of computations necessary to produce the feature subset.

   Some of the wrapper methods are:-

   a. Genetic Algorithms:- A subset of features can be found using this technique. CHCGA is a modified version of this algorithm that converges faster and produces a more effective search by preserving population diversity and avoiding population stagnation.

   b. Recursive Feature Elimination:- RFE is a feature selection method that fits a model and removes the weakest feature (or features) until the desired amount of features is reached. The model's coef_ or feature importances_ attributes rank features, and RFE seeks to minimize dependencies and collinearity in the model by recursively deleting a small number of features per loop. RFE requires that a certain number of features be kept, although the amount of valid features is frequently unknown in advance. Cross-validation is used with RFE to score several feature subsets and pick the top scoring collection of features to determine the optimal amount of features.

   c. Sequential Feature Selection:- This naive method starts with a null set, then adds one feature to the first step that represents the maximum value for the objective function, and then adds the remaining features individually to the current subset from the second step onwards, resulting in the new subset being assessed. This approach is continued until all of the essential features have been included.

3. Embedded Method:- This method seeks to combine the efficiency of both preceding methods and performs variable selection throughout the training process. It is usually particular to specific learning machines. This method essentially determines which attribute contributes the most to the model's accuracy.

Some of the embedded method techniques are:-

a. L1 Stabilization:- LASSO (Least Absolute Shrinkage and Selection Operator) is a linear model that estimates sparse coefficients and is effective in specific situations since it prefers solutions with fewer parameter values.
b. Ridge Regression:- The L2 Regularization, also known as Ridge Regression or Tikhonov Regularization, solves a regression model with a linear least squares function as the loss function and regularization.
c. Elastic Net:- This linear regression model is trained with L1 and L2 as regularizes, allowing it to learn a sparse model with few non-zero weights, similar to Lasso, but yet keeping Ridge's regularization qualities.

2.4 Stage 4

2.4.1 Building Supervised Models

2.4.1.1 Naive Bayes

The Bayes theorem inspired Naive Bayes, a probabilistic classifier that works under the basic assumption that the qualities are conditionally independent. With the above assumption applied to Bayes theory, the classification is done by obtaining the maximum posterior, which is the maximal P(Ci|X). By merely counting the class distribution, this assumption drastically minimizes the computational cost. Despite the fact that the assumption is not valid in most circumstances because the qualities are dependent, Naive Bayes has been able to perform well. Naive Bayes is a very simple algorithm to develop, and it has produced good results in the majority of applications. Because it requires linear time rather than the expensive iterative approximation employed by many other types of classifiers, it can quickly scale to larger datasets. The zero probability problem can be a difficulty with naive Bayes. When the conditional probability for a given property is zero, the prediction is invalid. Using a Laplacian estimator, this must be addressed explicitly.

$$P(\mathbf{X} \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \ldots \times P(x_n \mid C_i)$$

There are three types of Naive Bayes classifiers:-

- ✓ Multinomial Naïve Bayes
- ✓ Bernoulli Naïve Bayes
- ✓ Gaussian Naïve Bayes

Advantages:-

- ✓ This technique is extremely quick and can accurately predict the test dataset's class.
- ✓ It can be used to address multi-class prediction issues because it works well with them.
- ✓ If the premise of feature independence holds true, the Naive Bayes classifier outperforms alternative models with less training data.
- ✓ In compared to numerical data, the Naive Bayes method works remarkably well with categorical input variables.

Disadvantages:-

- ✓ If your test data set contains a categorical variable of a category that was not present in the training data set, the Naive Bayes model will give it zero probability and will be unable to make any predictions. This issue is known as 'Zero Frequency,' and to solve it, you'll need to employ a smoothing approach.
- ✓ This approach is also known for being a bad estimator. As a result, you shouldn't take ‘predict proba"s probability outputs too seriously.
- ✓ It is assumed that all of the features are self-contained. While this sounds fantastic in theory, you'll rarely encounter a set of separate features in practise.

Applications:-

- ✓ You can use this technique to make real-time forecasts because it is fast and efficient.
- ✓ For multi-class predictions, this approach is widely used. Using this approach, you can quickly determine the probability of numerous target classes.
- ✓ This algorithm is used by email services to determine whether or not an email is spam. This algorithm is fantastic at detecting spam.
- ✓ It's ideal for Sentiment Analysis because of its feature independence assumption and effectiveness in handling multi-class problems. Sentiment Analysis is the process of determining if a target group's sentiments are good or negative.
- ✓ To create recommendation systems, Collaborative Filtering and the Naive Bayes algorithm operate together. These systems employ data mining and machine learning to forecast whether or not a user will enjoy a specific content.

2.4.1.2 Support Vector Machine (SVM)

The SVM algorithm's purpose is to find the best line or decision boundary that can divide n-dimensional space into classes so that fresh data points can be readily placed in the correct category in the future. A hyperplane denotes the optimal choice boundary. SVM selects the hyperplane helping extreme points/vectors. Support vectors are the extreme situations, and the Support Vector Machine algorithm is named after them. It is of 2 types:-

i. Linear SVM:- It is a classifier that is used for linearly separable data, which implies that if a dataset can be classified into two classes using a single straight line, it is called linearly separable data, and the classifier is named Linear SVM.

ii. Non-linear SVM:- It is used for non-linearly separated data, which implies that if a dataset can't be classified using a straight line, it's non-linear data, and the classifier employed is called Non-linear SVM.

Advantages:-

- ✓ Regularization capabilities: SVM contains a capability called L2 Regularization. As a result, it has high generalization capabilities and avoids over-fitting.
- ✓ Effectively handles non-linear data: Using the Kernel technique, SVM can effectively handle non-linear data.
- ✓ Can be utilized to address both classification and regression problems: SVM can solve both classification and regression problems. For classification difficulties, SVM is employed, while for regression problems, SVR (Support Vector Regression) is use.
- ✓ Stability: A little change in the data has little impact on the hyperplane and, as a result, the SVM. As a result, the SVM model is reliable.

Disadvantages:-

- ✓ It's challenging to pick the right Kernel function: It's not easy to pick the right Kernel function (to deal with non-linear data). It could be difficult and complicated. If you use a high-dimensional Kernel, you can end up with too many support vectors, slowing down the training process significantly.
- ✓ Extensive memory requirement: SVM has a high level of algorithmic complexity and memory needs. You'll need a lot of memory since you'll need to store all of the support vectors in memory, which increases dramatically as the size of the training dataset grows.
- ✓ Feature Scaling is Required: Before using SVM, the variables must be feature scaled.
- ✓ Long training time: On large datasets, SVM takes a long time to train.
- ✓ In comparison to Decision Trees, the SVM model is difficult to understand and interpret for humans.

Applications:-

- ✓ Face detection:- SVMc classifies sections of the picture as face or non-face and draws a square around the face.
- ✓ Text and hypertext categorization:- Both inductive and transductive models can use SVMs for text and hypertext classification. They classify articles into multiple groups using training data. It categorizes based on the generated score and compares it to a threshold number.

- ✓ Image classification:- The use of SVMs improves image classification search accuracy. It outperforms typical query-based searching techniques in terms of accuracy.
- ✓ Protein classification and cancer classification are examples of bio-informatics. SVM is used to detect gene classification, patient classification based on genes, and other medical difficulties.
- ✓ Protein fold and distant homology detection:- SVM methods are used to detect protein remote homology.
- ✓ Handwriting recognition:- SVMs are used to recognize commonly used handwritten characters.
- ✓ Generalized predictive control (GPC):- Control chaotic dynamics using usable parameters using SVM-based GPC.

2.4.1.3 K-Nearest Neighbor (KNN)

The k-Nearest Neighbor algorithm is a lazy learning algorithm that stores all instances in n-dimensional space that correspond to training data points. When an unknown discrete data is received, it examines the nearest k number of saved instances (nearest neighbors) and returns the most common class as the prediction, whereas real-valued data returns the mean of k nearest neighbors. The distance-weighted nearest neighbour method uses the following query to weight the contributions of each of the k neighbors based on their distance, providing greater weight to the closest neighbors.

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

Advantages:-

- ✓ KNN is known as the Lazy Learner since there is no training period. During the training phase, it does not learn anything. The training data isn't used to derive any discriminating functions. In other words, it does not require any training. It saves the training dataset and uses it only when making real-time predictions to learn from it. This makes the KNN method much faster than other training-based algorithms.

- ✓ Because the KNN algorithm does not require any training before making predictions, new data can be supplied without affecting the system's accuracy.
- ✓ KNN is a simple algorithm to use. The distance function and the value of K are the only two parameters necessary to implement KNN.

Disadvantages:-

- ✓ Does not function well with large datasets: When working with large datasets, the cost of computing the distance between a new point and an old point becomes prohibitively expensive, lowering the algorithm's performance.
- ✓ The KNN technique does not perform well with high-dimensional data because it becomes harder for the algorithm to calculate the distance in each dimension when the number of dimensions increases.
- ✓ Before applying the KNN method to any dataset, we must first perform feature scaling. If we don't, KNN may make incorrect predictions.
- ✓ KNN is sensitive to noise in the dataset, as well as missing values and outliers. Missing values must be manually imputed, and outliers must be removed.

Applications:-

- ✓ Text mining.
- ✓ Agriculture.
- ✓ Finance.
- ✓ Medical.
- ✓ Facial recognition.

2.4.1.4 Decision Trees

Decision tree builds classification or regression models in the form of a tree structure. It uses a mutually exclusive and exhaustive collection of if-then rules to classify data. The rules are learned one at a time, one by one, from the training data. The tuples covered by a rule are eliminated each time it is learned. On the training set, this process is repeated until a termination condition is satisfied. Top-down recursive divide-and-conquer is used to build the tree. All of the characteristics must be categorical. They should be discretized ahead of time if not. The information gain concept is used to identify attributes at the top of the tree that have a greater impact on classification. A decision tree can easily be over-fitted, resulting in an excessive number of branches, which can reveal anomalies due to noise or outliers. The performance of an over-fitted model on unseen data is bad, despite its great performance on training data. Pre-pruning, which stops tree growth early, or post-pruning, which removes branches from a fully grown tree, can both help to avoid this.

Advantages:-

- ✓ Clear Visualization: Because the idea is commonly utilized in our daily lives, the algorithm is simple to grasp, interpret, and visualize. Humans can easily interpret the output of a Decision Tree.
- ✓ Simple and straightforward: Decision Trees appear to be basic if-else statements that are straightforward to comprehend.
- ✓ Both classification and regression problems can be solved using a Decision Tree.
- ✓ Both continuous and categorical variables can be handled by a Decision Tree.
- ✓ There's no need to scale the features: Because the Decision Tree uses a rule-based approach rather than distance calculation, no feature scaling (standardization and normalization) is necessary.
- ✓ Non-linear parameters have no effect on the performance of a Decision Tree, unlike curve-based algorithms. As a result, if the independent variables are highly nonlinear, Decision Trees may outperform alternative curve-based methods.
- ✓ Missing values can be handled automatically using Decision Tree.
- ✓ Outliers are frequently tolerated by Decision Trees, which can handle them automatically.
- ✓ Less Training Duration: When compared to Random Forest, the training period is shorter since it generates only one tree instead of a forest of trees in Random Forest.

Disadvantages:-

- ✓ The fundamental issue with the Decision Tree is over-fitting. It usually results in data over-fitting, which leads to incorrect predictions. It continues to generate new nodes in order to fit the data (even noisy data), and the tree eventually gets too complex to interpret. It loses its ability to generalize in this way. On taught data, it functions admirably, but on unseen data, it begins to make several errors.

- ✓ High variance: Decision Trees are notorious for over-fitting data. over-fitting causes a lot of variance in the output, which leads to a lot of inaccuracies in the final estimation and shows a lot of inaccuracy in the findings. It leads to excessive variance in order to attain zero bias (over-fitting).
- ✓ Unstable: Adding a new data point can cause the overall tree to regenerate, requiring all nodes to be recalculated and regenerated.
- ✓ Noise affects it: Even a small amount of noise can make it unstable, resulting in incorrect predictions.
- ✓ Not appropriate for large datasets: If the data set is huge, a single tree can become complex and over-fitting can occur. As a result, rather than using a single Decision Tree, we should employ Random Forest in this scenario.

Applications:-

- ✓ One of the uses of decision trees is to assess future growth potential for organizations based on historical data. Historical sales data can be utilized in decision trees to help businesses expand and grow by allowing them to make significant adjustments in their strategy.
- ✓ Another usage of decision trees is in the identification of potential clients using demographic data. They can assist in streamlining a marketing budget and making informed selections about the business's target market. Without decision trees, the company may spend its marketing budget without considering a certain demography, which will have an impact on overall sales.
- ✓ Lenders can also use decision trees to estimate the likelihood of a customer defaulting on a loan by generating predictive models based on the client's previous data. To avoid losses, lenders can utilize a decision tree assistance tool to evaluate a customer's creditworthiness.
- ✓ In operations research, decision trees can be used to plan logistics and strategic management. They can assist in finding the best tactics to help a firm reach its objectives. Engineering, education, law, business,

healthcare, and finance are some of the other sectors where decision trees can be used.

2.4.1.5 Random Forest

Random forest is a supervised machine learning technique that can be used for classification and regression. The "forest" refers to a group of uncorrelated decision trees that are then combined to reduce variation and generate accurate data predictions. Any of the individual constituent models will outperform a large number of reasonably uncorrelated models (trees) working as a committee. The key is the low correlation between models.

Advantages:-

- ✓ Random Forest is a technique that uses Ensemble Learning and is based on the bagging algorithm. It grows as many trees as possible on a subset of the data and then merges the results of all of the trees. As a result, the over-fitting problem in decision trees is reduced, as is the variance, which increases accuracy.
- ✓ Random Forest can be used to address problems in both classification and regression.
- ✓ Both categorical and continuous variables function well with Random Forest.
- ✓ Missing values can be handled automatically using Random Forest.
- ✓ There's no need to scale the features: Random Forest does not require feature scaling (standardization and normalization) because it use a rule-based method rather than distance calculations.
- ✓ Effectively handles non-linear parameters: Unlike curve-based algorithms, non linear parameters have no effect on the performance of a Random Forest. As a result, if the independent variables are highly nonlinear, Random Forest may outperform conventional curve-based methods.
- ✓ Missing values can be handled automatically using Random Forest.
- ✓ Outliers are frequently tolerated well by Random Forest, which can handle them automatically.
- ✓ The Random Forest method has a high level of consistency. Even if a new data point is added to the dataset, the overall method remains

unaffected because the new data may have an impact on one tree, but it is extremely unlikely to have an impact on all trees.

- ✓ Random Forest, on the other hand, is less affected by noise.

Disadvantages:-

- ✓ Complexity: The Random Forest algorithm generates a large number of trees and then aggregates their outputs. This approach necessitates a significant increase in processing power and resources. On the other hand, a decision tree is straightforward and does not necessitate a large amount of computer power.
- ✓ Random Forest takes much longer to train than decision trees since it generates a large number of trees and makes decisions based on the majority of votes.

Applications:-

- ✓ Banking Industry
  - ■Credit Card Fraud Detection
  - ■Customer Segmentation
  - ■Predicting Loan Defaults
- ✓ Healthcare and Medicine
  - ■Cardiovascular Disease Prediction
  - ■Diabetes Prediction
  - ■Breast Cancer Prediction
- ✓ Stock Market
  - ■Stock Market Prediction
  - ■Stock Market Sentiment Analysis
  - ■Bitcoin Price Detection
- ✓ E-Commerce
  - ■Product Recommendation
  - ■Price Optimization
  - ■ Search Ranking

2.4.1.6 Logistic Regression

Whenever the dependent variable is categorical, like when it has binary outputs, such as "true" and "false" or "yes" and "no," logistic regression is used.

Despite the fact, that regression models aim to understand correlations between data inputs, logistic regression is mostly utilized to solve binary classification problems. The Sigmoid function is used to convert predicted values to probabilities. The logistic regression hypothesis suggests that the cost function be limited to a value between 0 and 1.

$$f(x) = \frac{1}{1 + \mathrm{e}^{-(x)}}$$

Advantages:-

- ✓ When the dataset is linearly separable, Logistic Regression works well.
- ✓ over-fitting is less likely with logistic regression, but it can happen in high-dimensional datasets. In these cases, regularization (L1 and L2) approaches should be used to minimize over-fitting.
- ✓ Logistic regression provides not only a measure of a predictor's relevance (coefficient size), but also the direction of connection (positive or negative).
- ✓ Logistic regression is more straightforward to apply, analyse, and train.

Disadvantages:-

- ✓ The assumption of linearity between the dependent and independent variables is the main restriction of Logistic Regression. Data is rarely linearly separable in the actual world. The majority of the time, data is a jumbled mess.
- ✓ Logistic Regression should not be used if the number of observations is less than the number of features, since it may result in over-fitting.
- ✓ Only discrete functions may be predicted using logistic regression. As a result, Logistic Regression's dependent variable is limited to the discrete number set. This limitation is troublesome in and of itself, as it makes continuous data prediction impossible.

Applications:-

- ✓ Text analytics
- ✓ Chi-square automatic interaction detection (CHAID)
- ✓ Conjoint analysis
- ✓ Bootstrapping statistics

- ✓ Nonlinear regression
- ✓ Cluster statistics and cluster analysis software
- ✓ Monte Carlo simulation
- ✓ Descriptive statistics

2.4.2 Evaluation Parameters

A classification algorithm's success is determined by its overall accuracy, recall, precision, F-measure, and false positive rate (FPR). Overall accuracy, detection rate, and false positive rate are all used to evaluate IDS performance.

		Test Result	
		0	1
Ground Truth	0	TN (True Negative)	FP (False Positive)
	1	FN (False Negative)	TP (True Positive)

- ➢ Accuracy is defined as the percentage of total records properly identified as positive or negative.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

- ➢ Recall Rate: The percentage of positive records accurately identified compared to the total number of negative records correctly classified as positive or wrongly classified as negative.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- ➢ The True Positive Rate (TPR) is a measurement of how many positive records are correctly identified.

$$TruePositiveRate = \frac{TruePositive}{FalseNegative + TruePositive}$$

- ➢ The False Positive Rate (FPR) is a measurement of how many negative records are correctly identified.

$$FalsePositiveRate = \frac{FalsePositive}{TrueNegative + FalsePositive}$$

- ➢ The True Negative Rate (TNR) is a measurement of how many positive records are wrongly identified.

$$TrueNegativeRate = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

- The False Negative Rate (FNR) is a measurement of how many negative records are wrongly identified.
- Precision is the ratio of accurately classified positive records to the total number of records labeled as positive.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

- The F-Measure is the harmonic mean of precision and recall, and it provides a better indicator of an unbalanced dataset's performance.

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

2.4.3 Receiver Operating Characteristics Curve (ROC Curve)

It's a graphical representation of a binary classifier system's diagnostic capabilities as its discriminating threshold is changed. Starting in 1941, the approach was created for operators of military radar receivers, hence the name. Plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold levels yields the ROC curve. It's also known as a plot of the power as a function of the decision rule's Type I Error. As a result, the ROC curve represents sensitivity or recall as a function of fall-out. In general, the ROC curve can be created by plotting the probability distributions for both detection and false alarm.

2.5 Stage 5

2.5.1 Comparative Analysis

For the training data sets of darknet network traffic, the five models (DT, KNN, LR, NB, and SVM) were compared in terms of accuracy, precision, sensitivity, and specificity metrics. The average accuracy, precision, sensitivity, and specificity across all categories are shown in Table 3.

Table 3: Comparative Analysis

	Before Feature Selection				After Feature Selection			
Algorithm Used	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
GaussianNB	0.585	0.812	0.585	0.622	0.914	0.918	0.914	0.915
Decision Tree Classifier	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999
Support Vector Machine	0.993	0.993	0.993	0.993	0.994	0.994	0.994	0.994
Logistic Regression	0.943	0.943	0.943	0.943	0.930	0.930	0.930	0.930
K Nearest Neighbors	0.992	0.992	0.992	0.992	0.996	0.996	0.996	0.996

2.6 Stage 6

2.6.1 Outcomes

Table 3 shows that the accuracy metrics for Logistic Regression, and GaussianNB models were reasonably constant across categories, while the other metrics (precision, sensitivity, and specificity) were inconsistent. Two models have (relatively) low metrics, indicating an under-fitting issue that isn't worth optimizing further. The Decision Tree, Support Vector Machine, and KNN all performed well on all measures (an average of 99 percent accuracy for both models). These models are quite comparable and perform admirable in other areas.

3.Results & Discussions

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        %matplotlib inline
```

Loading The Dataset

```
In [2]: df= pd.read_csv("Darknet_Data.csv")
        pd.set_option('display.max_columns', None)
        df.shape

Out[2]: (141530, 85)

In [3]: df.columns

Out[3]: Index(['Flow ID', 'Src IP', 'Src Port', 'Dst IP', 'Dst Port', 'Protocol',
               'Timestamp', 'Flow Duration', 'Total Fwd Packet', 'Total Bwd packets',
               'Total Length of Fwd Packet', 'Total Length of Bwd Packet',
               'Fwd Packet Length Max', 'Fwd Packet Length Min',
               'Fwd Packet Length Mean', 'Fwd Packet Length Std',
               'Bwd Packet Length Max', 'Bwd Packet Length Min',
               'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Bytes/s',
               'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max',
               'Flow IAT Min', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std',
               'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean',
               'Bwd IAT Std', 'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags',
               'Bwd PSH Flags', 'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Header Length',
               'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/s',
               'Packet Length Min', 'Packet Length Max', 'Packet Length Mean',
               'Packet Length Std', 'Packet Length Variance', 'FIN Flag Count',
               'SYN Flag Count', 'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count',
```

Figure 2 : Importing Packages and loading dataset

Cleaning Data

```
In [4]: df = df.replace([np.inf, -np.inf], np.nan)
        print(df)

                                          Flow ID          Src IP  Src Port  \
0        10.152.152.11-216.58.220.99-57158-443-6   10.152.152.11     57158
1        10.152.152.11-216.58.220.99-57159-443-6   10.152.152.11     57159
2        10.152.152.11-216.58.220.99-57160-443-6   10.152.152.11     57160
3       10.152.152.11-74.125.136.120-49134-443-6   10.152.152.11     49134
4     10.152.152.11-173.194.65.127-34697-19305-6   10.152.152.11     34697
...                                           ...             ...       ...
141525       10.8.8.246-224.0.0.252-55219-5355-17      10.8.8.246     55219
141526       10.8.8.246-224.0.0.252-64207-5355-17      10.8.8.246     64207
141527       10.8.8.246-224.0.0.252-61115-5355-17      10.8.8.246     61115
141528       10.8.8.246-224.0.0.252-64790-5355-17      10.8.8.246     64790
141529   80.239.235.110-10.8.8.246-11666-60245-17  80.239.235.110     11666

                Dst IP  Dst Port  Protocol         Timestamp  Flow Duration  \
0        216.58.220.99       443         6  24-07-2015 16:09            229
1        216.58.220.99       443         6  24-07-2015 16:09            407
2        216.58.220.99       443         6  24-07-2015 16:09            431
3       74.125.136.120       443         6  24-07-2015 16:09            359
4       173.194.65.127     19305         6  24-07-2015 16:09       10778451

In [5]: pd.set_option("max_rows", None)
        df.isnull().sum()

Out[5]: Flow ID                        0
        Src IP                         0
        Src Port                       0
```

Figure 3 : Cleaning imported data

```
        Total Fwd Packet               0
        Total Bwd packets              0
        Total Length of Fwd Packet     0
        Total Length of Bwd Packet     0
        Fwd Packet Length Max          0
        Fwd Packet Length Min          0
        Fwd Packet Length Mean         0
        Fwd Packet Length Std          0
        Bwd Packet Length Max          0
        Bwd Packet Length Min          0
        Bwd Packet Length Mean         0

In [6]: df['Flow Bytes/s'].fillna((df['Flow Bytes/s'].median()),inplace=True)
        df['Flow Packets/s'].fillna((df['Flow Packets/s'].median()),inplace=True)
        df.isnull().sum()

Out[6]: Flow ID                        0
        Src IP                         0
        Src Port                       0
        Dst IP                         0
        Dst Port                       0
        Protocol                       0
        Timestamp                      0
        Flow Duration                  0
        Total Fwd Packet               0
        Total Bwd packets              0
        Total Length of Fwd Packet     0
        Total Length of Bwd Packet     0
        Fwd Packet Length Max          0
        Fwd Packet Length Min          0
        Fwd Packet Length Mean         0
        Fwd Packet Length Std          0
        Bwd Packet Length Max          0
```

Figure 4 : Replacing null values with median

```
In [7]:  df = df.drop_duplicates()
          df.shape

Out[7]:  (117056, 85)

In [8]:  df.describe(include='all')

Out[8]:
```

	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Total Fwd Packet	Total Bwd packets	Total
count	117056	117056	117056.000000	117056	117056.000000	117056.000000	117056	1.170560e+05	117056.000000	117056.000000	1.1705
unique	77568	3914	NaN	7197	NaN	NaN	2464	NaN	NaN	NaN	
top	8.6.0.1-8.0.6.4-0-0-0	10.152.152.11	NaN	10.152.152.10	NaN	NaN	23-02-2016 14:09	NaN	NaN	NaN	
freq	430	59273	NaN	11865	NaN	NaN	1017	NaN	NaN	NaN	
mean	NaN	NaN	38336.817797	NaN	14193.148749	10.810638	NaN	1.999155e+07	161.712992	163.723508	1.1758
std	NaN	NaN	19492.571528	NaN	20779.156799	5.520079	NaN	3.781548e+07	2530.910936	3714.022418	3.5016
min	NaN	NaN	0.000000	NaN	0.000000	0.000000	NaN	0.000000e+00	1.000000	0.000000	0.0000
25%	NaN	NaN	30480.500000	NaN	53.000000	6.000000	NaN	9.150000e+02	1.000000	0.000000	0.0000
50%	NaN	NaN	43533.000000	NaN	443.000000	6.000000	NaN	4.111780e+05	2.000000	1.000000	4.4000
75%	NaN	NaN	53574.250000	NaN	31586.000000	17.000000	NaN	9.896463e+06	4.000000	3.000000	1.9600
max	NaN	NaN	65534.000000	NaN	65535.000000	17.000000	NaN	1.200000e+08	238161.000000	470862.000000	7.6930

Figure 5 : Dropping duplicates from dataset

Unique Values

```
In [9]:  for col in df:
              print('Number of unique values in',col,' is ',df[col].nunique())

Number of unique values in Bwd Packet Length Max  is  1477
Number of unique values in Bwd Packet Length Min  is  396
Number of unique values in Bwd Packet Length Mean  is  15272
Number of unique values in Bwd Packet Length Std  is  18540
Number of unique values in Flow Bytes/s  is  73854
Number of unique values in Flow Packets/s  is  82669
Number of unique values in Flow IAT Mean  is  82686
Number of unique values in Flow IAT Std  is  58063
Number of unique values in Flow IAT Max  is  79396
Number of unique values in Flow IAT Min  is  34375
Number of unique values in Fwd IAT Total  is  61186
Number of unique values in Fwd IAT Mean  is  61531
Number of unique values in Fwd IAT Std  is  45630
Number of unique values in Fwd IAT Max  is  58428
Number of unique values in Fwd IAT Min  is  27164
Number of unique values in Bwd IAT Total  is  39204
Number of unique values in Bwd IAT Mean  is  39217
Number of unique values in Bwd IAT Std  is  32414
Number of unique values in Bwd IAT Max  is  37218
Number of unique values in Bwd IAT Min  is  15446

In [10]: count = df['Label'].value_counts()
         print(count)

Non-Tor    69112
NonVPN     23846
VPN        22919
Tor         1179
```

Figure 6 : Number of unique values in each column

```
Non-Tor    69112
NonVPN     23846
VPN        22919
Tor         1179
Name: Label, dtype: int64

In [11]: count = df['Label.1'].value_counts()
         print(count)

Browsing           32714
P2P                24260
Audio-Streaming    16477
Chat               11477
File-Transfer      11098
Video-Streaming     9486
Email               6129
VOIP                3566
AUDIO-STREAMING     1484
Video-streaming      281
File-transfer         84
Name: Label.1, dtype: int64

In [12]: df['Label.1'].loc[df['Label.1'] == 'AUDIO-STREAMING'] = 'Audio-Streaming'
         df['Label.1'].loc[df['Label.1'] == 'File-transfer'] = 'File-Transfer'
         df['Label.1'].loc[df['Label.1'] == 'Video-streaming'] = 'Video-Streaming'

C:\Users\sahit\anaconda3\lib\site-packages\pandas\core\indexing.py:670: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  iloc._setitem_with_indexer(indexer, value)

In [13]: count = df['Label.1'].value_counts()
```

Figure 7 : Unique values and their count in Label and Label.1

```
In [13]:  count = df['Label.1'].value_counts()
          print(count)

          Browsing          32714
          P2P               24260
          Audio-Streaming   17961
          Chat              11477
          File-Transfer     11182
          Video-Streaming    9767
          Email              6129
          VOIP               3566
          Name: Label.1, dtype: int64

In [14]:  samples = df.copy()

In [15]:  samples['Label'].loc[(samples['Label'] == 'Non-Tor') | (samples['Label'] == 'NonVPN')] = 'Benign'
          samples['Label'].loc[(samples['Label'] == 'Tor') | (samples['Label'] == 'VPN')] = 'Darknet'

          traffic_labels = samples["Label"].unique()
          print(traffic_labels)

          ['Benign' 'Darknet']

In [16]:  d1 = samples.groupby("Label").count()
          fig, ax = plt.subplots(figsize=(7,7))
          d1.head()
          plt.pie(d1["Flow ID"], shadow=False,
                  autopct=lambda x: int(d1["Label.1"].values.sum()*x/100), colors=["#4682B4","#228B22"])
          plt.legend(["Benign", "Darknet"], bbox_to_anchor=(0.43,0.3,0.45,0.4),framealpha=0.8)
          plt.tight_layout()
          plt.show()
```

Figure 8 : Classifying data to Benign and Darknet.

```
d1.head()
plt.pie(d1["Flow ID"], shadow=False,
        autopct=lambda x: int(d1["Label.1"].values.sum()*x/100), colors=["#4682B4","#228B22"])
plt.legend(["Benign", "Darknet"], bbox_to_anchor=(0.43,0.3,0.45,0.4),framealpha=0.8)
plt.tight_layout()
plt.show()
```

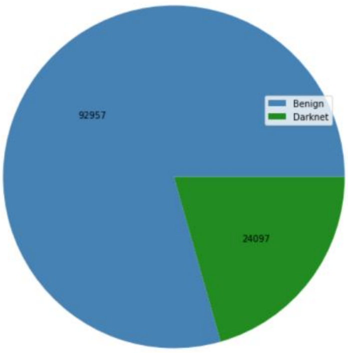


Figure 9 : Pictorial representation of Benign and Darknet.

```
In [17]:  samples.describe(include='all')
Out[17]:
```

	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Total Fwd Packet	Total Bwd packets	Total
count	117056	117056	117056.000000	117056	117056.000000	117056.000000	117056	1.170560e+05	117056.000000	117056.000000	1.1705
unique	77568	3914	NaN	7197	NaN	NaN	2464	NaN	NaN	NaN	
top	8.6.0.1-8.0.6.4-0-0-0	10.152.152.11	NaN	10.152.152.10	NaN	NaN	23-02-2016 14:09	NaN	NaN	NaN	
freq	430	59273	NaN	11865	NaN	NaN	1017	NaN	NaN	NaN	
mean	NaN	NaN	38336.817797	NaN	14193.148749	10.810638	NaN	1.999155e+07	161.712992	163.723508	1.1758
std	NaN	NaN	19492.571528	NaN	20779.156799	5.520079	NaN	3.781548e+07	2530.910936	3714.022418	3.5016
min	NaN	NaN	0.000000	NaN	0.000000	0.000000	NaN	0.000000e+00	1.000000	0.000000	0.0000
25%	NaN	NaN	30480.500000	NaN	53.000000	6.000000	NaN	9.150000e+02	1.000000	0.000000	0.0000
50%	NaN	NaN	43533.000000	NaN	443.000000	6.000000	NaN	4.111780e+05	2.000000	1.000000	4.4000
75%	NaN	NaN	53574.250000	NaN	31586.000000	17.000000	NaN	9.896463e+06	4.000000	3.000000	1.9600
max	NaN	NaN	65534.000000	NaN	65535.000000	17.000000	NaN	1.200000e+08	238161.000000	470862.000000	7.6930

```
In [18]:  hours = []
          for timestamp in samples['Timestamp']:
```

Figure 10 : Description of all attributes in the dataset

```
In [18]:  hours = []
          for timestamp in samples['Timestamp']:
              hora = int(timestamp.split()[1].split(':')[0])
              hours.append(hora)
          samples['hour'] = hours
          print(samples[['Timestamp', 'hour']][:5])

                   Timestamp  hour
          0  24-07-2015 16:09    16
          1  24-07-2015 16:09    16
          2  24-07-2015 16:09    16
          3  24-07-2015 16:09    16
          4  24-07-2015 16:09    16

In [19]:  df.info()

          <class 'pandas.core.frame.DataFrame'>
          Int64Index: 117056 entries, 0 to 141529
          Data columns (total 85 columns):
           #   Column                       Non-Null Count   Dtype
          ---  ------                       --------------   -----
           0   Flow ID                      117056 non-null  object
           1   Src IP                       117056 non-null  object
           2   Src Port                     117056 non-null  int64
           3   Dst IP                       117056 non-null  object
           4   Dst Port                     117056 non-null  int64
           5   Protocol                     117056 non-null  int64
           6   Timestamp                    117056 non-null  object
           7   Flow Duration                117056 non-null  int64
           8   Total Fwd Packet             117056 non-null  int64
           9   Total Bwd packets            117056 non-null  int64
           10  Total Length of Fwd Packet   117056 non-null  int64
           11  Total Length of Bwd Packet   117056 non-null  int64
```

Figure 11 : Deriving hours from timestamp

```
In [20]:  da=samples.copy()
```

Preprocessing Data

```
In [21]:  from sklearn.preprocessing import LabelEncoder
          number = LabelEncoder()
          da['Flow ID']=number.fit_transform(da['Flow ID'])
          da['Src IP']=number.fit_transform(da['Src IP'])
          da['Dst IP']=number.fit_transform(da['Dst IP'])
          da['Timestamp']=number.fit_transform(da['Timestamp'])
          da['Label']=number.fit_transform(da['Label'])
          da['Label.1']=number.fit_transform(da['Label.1'])
          da.info()

          <class 'pandas.core.frame.DataFrame'>
          Int64Index: 117056 entries, 0 to 141529
          Data columns (total 86 columns):
           #   Column                       Non-Null Count   Dtype
          ---  ------                       --------------   -----
           0   Flow ID                      117056 non-null  int32
           1   Src IP                       117056 non-null  int32
           2   Src Port                     117056 non-null  int64
           3   Dst IP                       117056 non-null  int32
           4   Dst Port                     117056 non-null  int64
           5   Protocol                     117056 non-null  int64
           6   Timestamp                    117056 non-null  int32
           7   Flow Duration                117056 non-null  int64
           8   Total Fwd Packet             117056 non-null  int64
           9   Total Bwd packets            117056 non-null  int64
           10  Total Length of Fwd Packet   117056 non-null  int64
           11  Total Length of Bwd Packet   117056 non-null  int64
```

Figure 12 : Normalizing the data

```
In [22]:  float_col = df.select_dtypes(include=['float64'])
          for col in float_col.columns.values:
              df[col] = df[col].astype('int64')

In [23]:  da.info()

          <class 'pandas.core.frame.DataFrame'>
          Int64Index: 117056 entries, 0 to 141529
          Data columns (total 86 columns):
           #   Column                       Non-Null Count   Dtype
          ---  ------                       --------------   -----
           0   Flow ID                      117056 non-null  int32
           1   Src IP                       117056 non-null  int32
           2   Src Port                     117056 non-null  int64
           3   Dst IP                       117056 non-null  int32
           4   Dst Port                     117056 non-null  int64
           5   Protocol                     117056 non-null  int64
           6   Timestamp                    117056 non-null  int32
           7   Flow Duration                117056 non-null  int64
           8   Total Fwd Packet             117056 non-null  int64
           9   Total Bwd packets            117056 non-null  int64
           10  Total Length of Fwd Packet   117056 non-null  int64
           11  Total Length of Bwd Packet   117056 non-null  int64
           12  Fwd Packet Length Max        117056 non-null  int64
           13  Fwd Packet Length Min        117056 non-null  int64
           14  Fwd Packet Length Mean       117056 non-null  float64
           15  Fwd Packet Length Std        117056 non-null  float64
           16  Bwd Packet Length Max        117056 non-null  int64
           17  Bwd Packet Length Min        117056 non-null  int64
           18  Bwd Packet Length Mean       117056 non-null  float64
           19  Bwd Packet Length Std        117056 non-null  float64
           20  Flow Bytes/s                 117056 non-null  float64
```

Figure 13 : Converting float values to int

Plotting Graphs

In [24]:
```
import seaborn as sns
```

In [25]:
```
fig, ax = plt.subplots(figsize=(10,5))
sns.histplot(data=samples, x="Label", stat="probability")
plt.show()
```

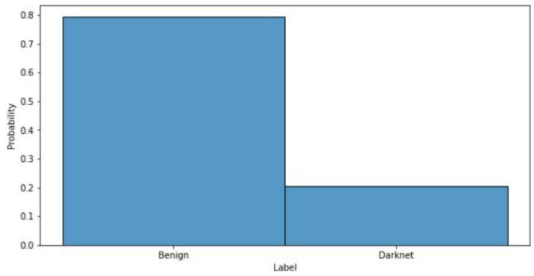


Figure 14 : Histogram representation of Label data

In [26]:
```
fig, ax = plt.subplots(figsize=(22,8))
sns.histplot(data=samples, x="Label.1", stat="probability")
plt.show()
```

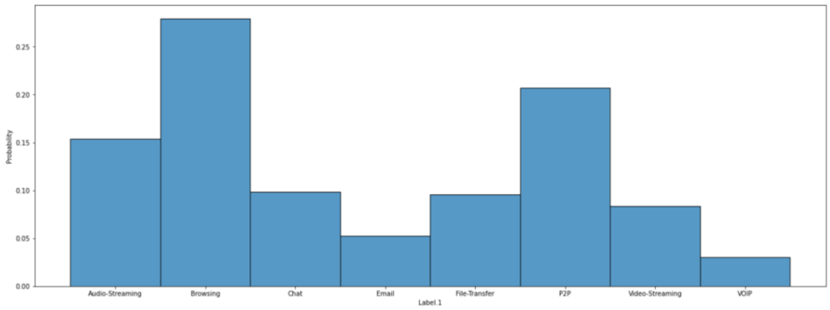


Figure 15 : Histogram representation of Label.1 data

In [27]:
```
fig, ax = plt.subplots(figsize=(24,8))
sns.histplot(data=samples[samples['Label'] == "Benign"], x="Label.1", stat="probability")
ax.set_title( "Benign Network Traffic Probability" , size = 24 )
plt.show()
```

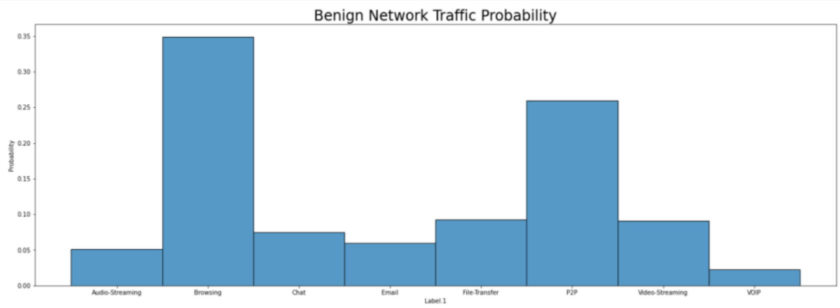


Figure 16 : Histogram representation of Benign Network Traffic Probability

```
In [28]:  fig, ax = plt.subplots(figsize=(24,8))
          sns.histplot(data=samples[samples['Label'] == "Darknet"], x="Label.1", stat="probability")
          ax.set_title("Darknet Network Traffic Probability", size = 24 )
          plt.show()
```

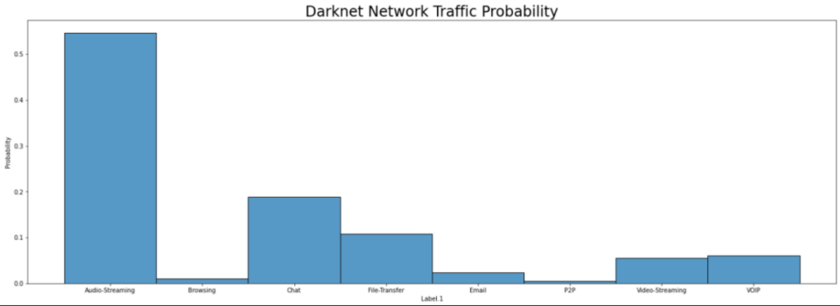


Figure 17 : Histogram representation of Darknet Network Traffic Probability

```
fig,ax=plt.subplots(figsize=(28,10))
sns.histplot(data=samples[samples['Label'] == "Benign"], x="Label.1", stat="probability",shrink=0.8 ,label="Benign", color='b
sns.histplot(data=samples[samples['Label'] == "Darknet"], x="Label.1", stat="probability",shrink=0.6, label="Darknet", color=
plt.xlabel("Traffic Type Labels")
plt.ylabel("Probability")
ax.set_title( "Traffic in Network" , size = 24 )
plt.legend()
plt.xticks(rotation=-10)
plt.show()
```

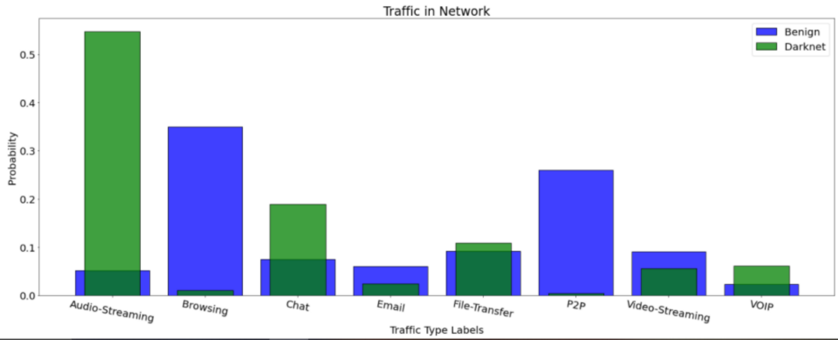


Figure 18 : Histogram representation of Traffic in Network

```
In [30]:  fig, ax = plt.subplots(figsize=(12,6))
          sns.histplot(data=samples[samples['Label'] == "Darknet"], x="hour", stat="probability")
          ax.set_title("Darknet Network identified in hours" , size = 24)
          plt.show()
```

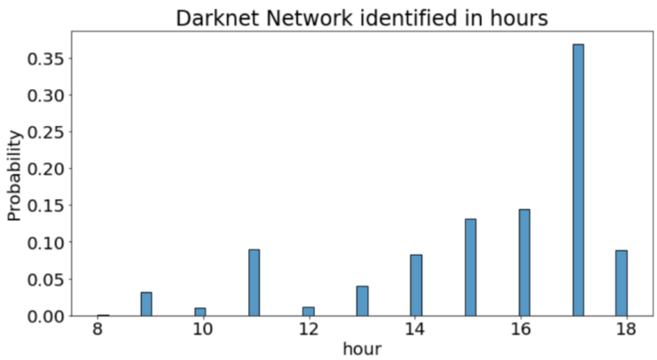


Figure 19 : Histogram representation of Darknet Network identified in hours

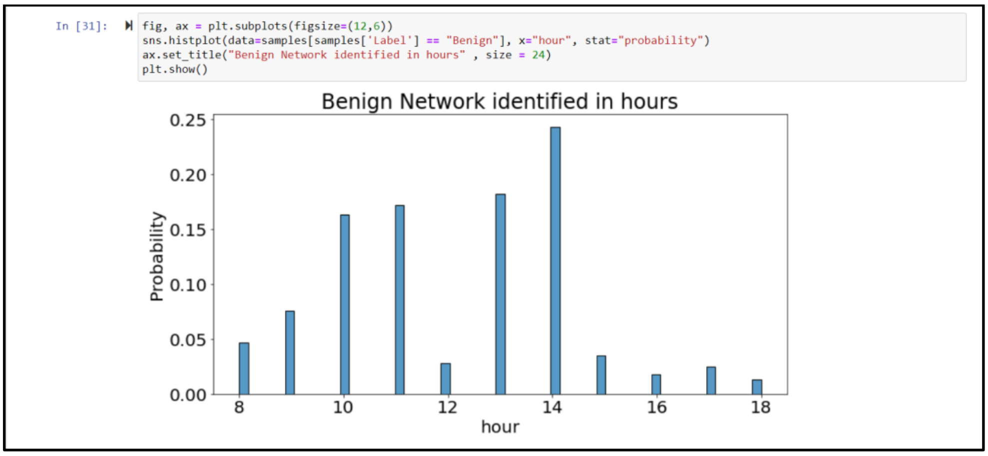


Figure 20 : Histogram representation of Benign Network identified in hours

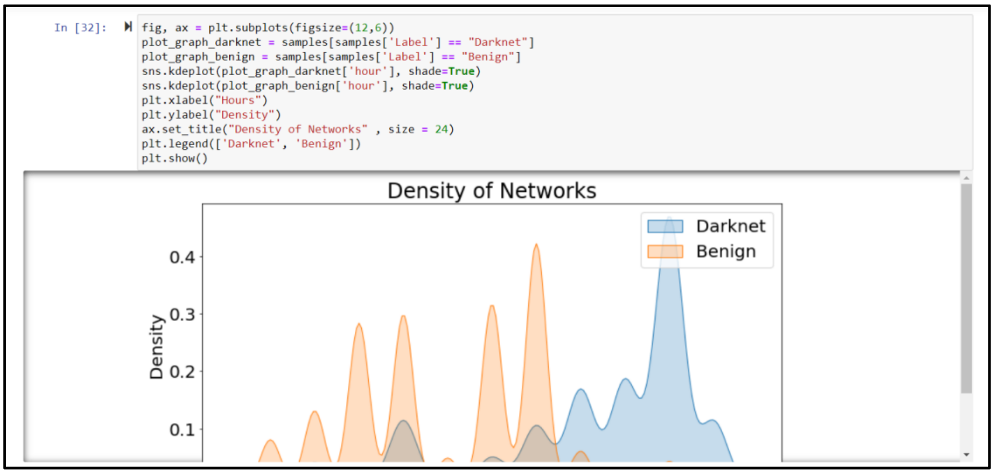


Figure 21 : Density Of Networks

Figure 22 : Loading Dataset

Training & Testing Data

```
In [3]:  features=['Flow ID', 'Src IP', 'Src Port', 'Dst IP', 'Dst Port', 'Protocol',
                 'Timestamp', 'Flow Duration', 'Total Fwd Packet', 'Total Bwd packets',
                 'Total Length of Fwd Packet', 'Total Length of Bwd Packet',
                 'Fwd Packet Length Max', 'Fwd Packet Length Min',
                 'Fwd Packet Length Mean', 'Fwd Packet Length Std',
                 'Bwd Packet Length Max', 'Bwd Packet Length Min',
                 'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Bytes/s',
                 'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max',
                 'Flow IAT Min', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std',
                 'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean',
                 'Bwd IAT Std', 'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags',
                 'Bwd PSH Flags', 'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Header Length',
                 'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/s',
                 'Packet Length Min', 'Packet Length Max', 'Packet Length Mean',
                 'Packet Length Std', 'Packet Length Variance', 'FIN Flag Count',
                 'SYN Flag Count', 'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count',
                 'URG Flag Count', 'CWE Flag Count', 'ECE Flag Count', 'Down/Up Ratio',
                 'Average Packet Size', 'Fwd Segment Size Avg', 'Bwd Segment Size Avg',
                 'Fwd Bytes/Bulk Avg', 'Fwd Packet/Bulk Avg', 'Fwd Bulk Rate Avg',
                 'Bwd Bytes/Bulk Avg', 'Bwd Packet/Bulk Avg', 'Bwd Bulk Rate Avg',
                 'Subflow Fwd Packets', 'Subflow Fwd Bytes', 'Subflow Bwd Packets',
                 'Subflow Bwd Bytes', 'FWD Init Win Bytes', 'Bwd Init Win Bytes',
                 'Fwd Act Data Pkts', 'Fwd Seg Size Min', 'Active Mean', 'Active Std',
                 'Active Max', 'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max',
                 'Idle Min', 'Label.1']
         X = da[features] # Features
         y = da['Label']
```

Figure 23 : Preparing Data

```
In [4]:  from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33,random_state =1)
         from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         X_train = sc.fit_transform(X_train)
         X_test = sc.transform(X_test)
```

Figure 24 : Splitting the data

Feature Selection with Recursive Feature Elimination

```
In [5]:  from sklearn.ensemble import RandomForestClassifier
         from sklearn.feature_selection import RFE
         X_train10, X_test10, y_train10, y_test10 = train_test_split(X, y, test_size = 0.33,random_state =1)

In [6]:  sel = RFE(RandomForestClassifier(n_estimators = 100),n_features_to_select=20)
         sel.fit(X_train10, y_train10)

Out[6]:  RFE(estimator=RandomForestClassifier(), n_features_to_select=20)

In [7]:  sel.get_support()

Out[7]:  array([ True,  True, False,  True, False, False,  True,  True, False,
                False, False,  True, False, False, False, False,  True,  True,
                 True, False,  True, False,  True, False,  True,  True, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False,  True, False,  True, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False,  True, False, False,
                False, False, False, False, False, False, False, False,  True,
                 True, False,  True, False, False, False, False, False, False,
                False, False,  True])
```

Figure 25 : Feature Selection with Recursive Elimination

```
In [8]:  selected_feat= X_train10.columns[(sel.get_support())]
         len(selected_feat)

Out[8]:  20

In [9]:  print(selected_feat)

         Index(['Flow ID', 'Src IP', 'Dst IP', 'Timestamp', 'Flow Duration',
                'Total Length of Bwd Packet', 'Bwd Packet Length Max',
                'Bwd Packet Length Min', 'Bwd Packet Length Mean', 'Flow Bytes/s',
                'Flow IAT Mean', 'Flow IAT Max', 'Flow IAT Min', 'Fwd Header Length',
                'Fwd Packets/s', 'Bwd Segment Size Avg', 'FWD Init Win Bytes',
                'Bwd Init Win Bytes', 'Fwd Seg Size Min', 'Label.1'],
               dtype='object')

In [10]: X1 = da[selected_feat] # Features

In [11]: X_train1, X_test1, y_train1, y_test1= train_test_split(X1, y, test_size = 0.33,random_state =1)
         X_train1 = sc.fit_transform(X_train1)
         X_test1 = sc.transform(X_test1)
```

Figure 26 : Selecting top features

Applying GaussianNB before Feature Selection

```
In [12]:  from sklearn.naive_bayes import GaussianNB
          classifier = GaussianNB()
          classifier.fit(X_train, y_train)

Out[12]:  GaussianNB()

In [13]:  y_pred = classifier.predict(X_test)

In [14]:  from sklearn.metrics import confusion_matrix
          from sklearn import metrics
          cm = metrics.confusion_matrix(y_test, y_pred)
          from sklearn.metrics import accuracy_score
          print ("Accuracy for Naive Bayes: ", accuracy_score(y_test, y_pred))
          print("Confusion Matrix for Naive Bayes")
          print(cm)

          Accuracy for Naive Bayes:  0.5851821170623107
          Confusion Matrix for Naive Bayes
          [[15732 15022]
           [ 1002  6873]]

In [15]:  print(metrics.classification_report(y_test, y_pred))

                        precision    recall  f1-score   support

                     0       0.94      0.51      0.66     30754
                     1       0.31      0.87      0.46      7875

              accuracy                           0.59     38629
             macro avg       0.63      0.69      0.56     38629
```

Figure 27 : Applying GaussianNB before feature selection

```
In [16]:  from sklearn.metrics import precision_score
          print ("Precision for Naive Bayes: ", precision_score(y_test, y_pred,average='weighted'))
          from sklearn.metrics import recall_score
          print ("Recall score for Naive Bayes: ", recall_score(y_test, y_pred,average='weighted'))
          from sklearn.metrics import f1_score
          print ("F1 score for Naive Bayes: ", f1_score(y_test, y_pred,average='weighted'))

          Precision for Naive Bayes:  0.8124603021204716
          Recall score for Naive Bayes:  0.5851821170623107
          F1 score for Naive Bayes:  0.621626237409026
```

Applying GaussianNB after Feature Selection

```
In [17]:  classifier1 = GaussianNB()
          classifier1.fit(X_train1, y_train1)

Out[17]:  GaussianNB()

In [18]:  y_p= classifier1.predict(X_test1)

In [19]:  m = metrics.confusion_matrix(y_test1, y_p)
          print ("Accuracy for Naive Bayes: ", accuracy_score(y_test1, y_p))
          print("Confusion Matrix for Naive Bayes")
          print(m)

          Accuracy for Naive Bayes:  0.9139765461182013
          Confusion Matrix for Naive Bayes
          [[28692  2062]
           [ 1261  6614]]
```

Figure 28 : Applying GaussianNB after feature selection

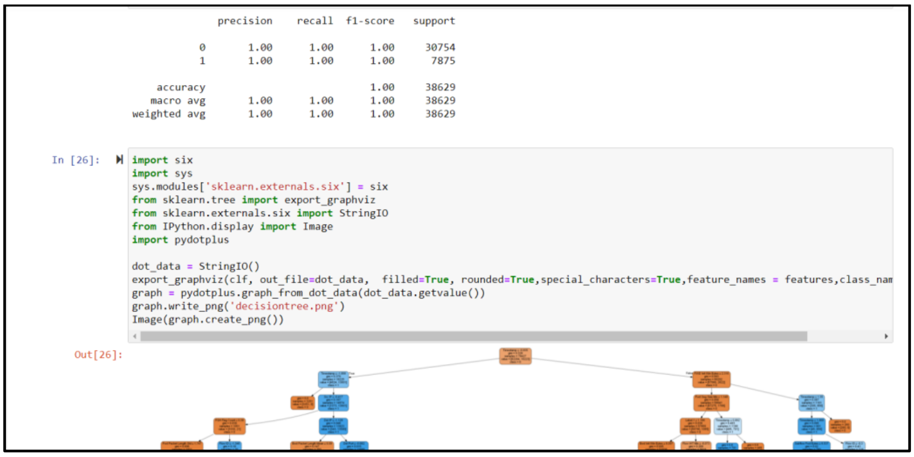
```
In [20]:  print(metrics.classification_report(y_test1, y_p))

                        precision    recall  f1-score   support

                     0       0.96      0.93      0.95     30754
                     1       0.76      0.84      0.80      7875

              accuracy                           0.91     38629
             macro avg       0.86      0.89      0.87     38629
          weighted avg       0.92      0.91      0.92     38629

In [21]:  print ("Precision for Naive Bayes: ", precision_score(y_test1, y_p,average='weighted'))
          print ("Recall score for Naive Bayes: ", recall_score(y_test1, y_p,average='weighted'))
          print ("F1 score for Naive Bayes: ", f1_score(y_test1, y_p,average='weighted'))

          Precision for Naive Bayes:  0.9180317854176133
          Recall score for Naive Bayes:  0.9139765461182013
          F1 score for Naive Bayes:  0.9154906166626585
```

Figure 29 : Evaluation Metric values for GaussianNB

Applying Decision Tree Classifier before Feature Selection

```
In [22]:  from sklearn.tree import DecisionTreeClassifier
          from sklearn.tree import export_text
          clf = DecisionTreeClassifier()

          # Train Decision Tree Classifer
          clf = clf.fit(X_train,y_train)

          #Predict the response for test dataset
          y_pred1 = clf.predict(X_test)

In [23]:  cm5 = metrics.confusion_matrix(y_test, y_pred1)
          print("Confusion Matrix for Decision Tree:")
          print(cm5)

          Confusion Matrix for Decision Tree:
          [[30750     4]
           [    4  7871]]

In [24]:  print("Accuracy:",metrics.accuracy_score(y_test, y_pred1))
          print ("Precision: ", precision_score(y_test, y_pred1,average='weighted'))
          print ("Recall score:  ", recall_score(y_test, y_pred1,average='weighted'))
          print ("F1 score: ", f1_score(y_test, y_pred1,average='weighted'))

          Accuracy: 0.9997929017059722
          Precision:  0.9997929017059722
          Recall score:   0.9997929017059722
          F1 score:  0.9997929017059722
```

Figure 30 : Applying Decision tree classifier before feature selection



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     30754
           1       1.00      1.00      1.00      7875

    accuracy                           1.00     38629
   macro avg       1.00      1.00      1.00     38629
weighted avg       1.00      1.00      1.00     38629
```

In [26]:

```
import six
import sys
sys.modules['sklearn.externals.six'] = six
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,  filled=True, rounded=True,special_characters=True,feature_names = features,class_nam
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('decisiontree.png')
Image(graph.create_png())
```

Out[26]:

Figure 31 : Printing decision tree

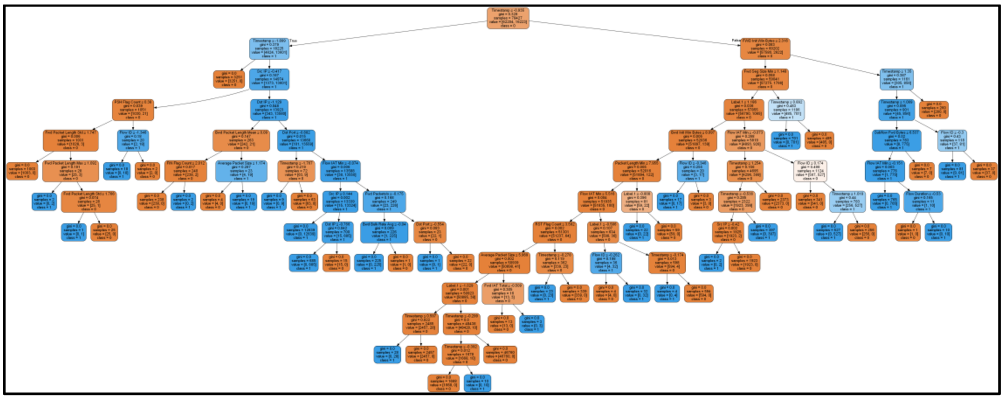


Figure 32 : Decision tree before feature selection

Applying Decision Tree Classifier after Feature Selection

In [27]:

```
clf1 = DecisionTreeClassifier()
clf1= clf1.fit(X_train1,y_train1)

#Predict the response for test dataset
y_pred11 = clf1.predict(X_test1)
```

In [28]:

```
cm51 = metrics.confusion_matrix(y_test1, y_pred11)
print("Confusion Matrix for Decision Tree:")
print(cm51)
```

```
Confusion Matrix for Decision Tree:
[[30752     2]
 [    1  7874]]
```

In [29]:

```
print("Accuracy:",metrics.accuracy_score(y_test1, y_pred11))
print ("Precision: ", precision_score(y_test1, y_pred11,average='weighted'))
print ("Recall score:  ", recall_score(y_test1, y_pred11,average='weighted'))
print ("F1 score: ", f1_score(y_test1, y_pred11,average='weighted'))
```

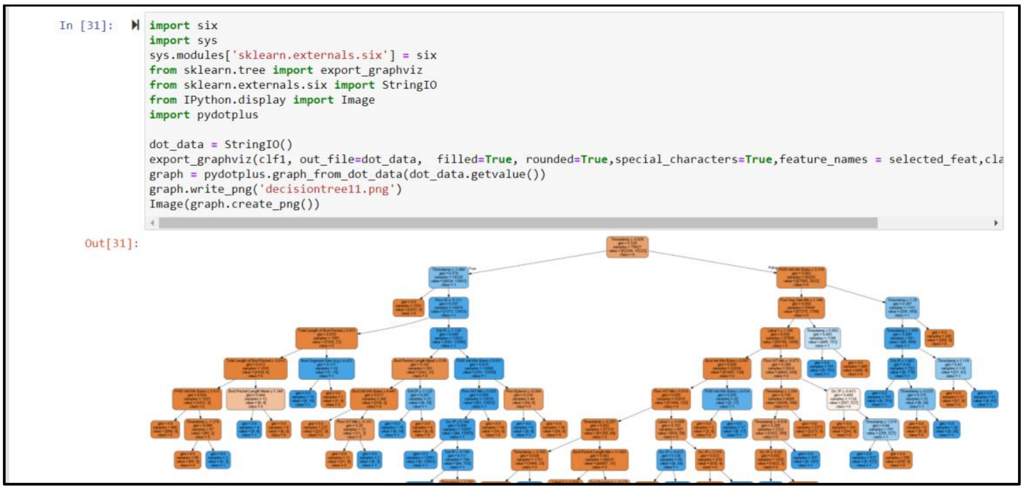
```
Accuracy: 0.9999223381397395
Precision:  0.999922343871673
Recall score:   0.9999223381397395
F1 score:  0.9999223399737135
```

In [30]:

```
print(metrics.classification_report(y_test1, y_pred11))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     30754
           1       1.00      1.00      1.00      7875
```

Figure 33 : Applying decision tree classifier after feature selection



```
In [31]: import six
         import sys
         sys.modules['sklearn.externals.six'] = six
         from sklearn.tree import export_graphviz
         from sklearn.externals.six import StringIO
         from IPython.display import Image
         import pydotplus

         dot_data = StringIO()
         export_graphviz(clf1, out_file=dot_data,  filled=True, rounded=True,special_characters=True,feature_names = selected_feat,cla
         graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
         graph.write_png('decisiontree11.png')
         Image(graph.create_png())
Out[31]:
```

Figure 34 : Printing decision tree

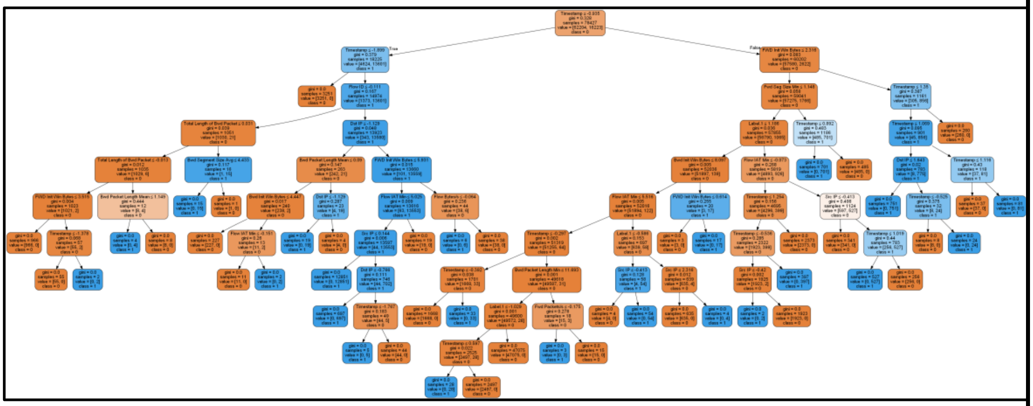
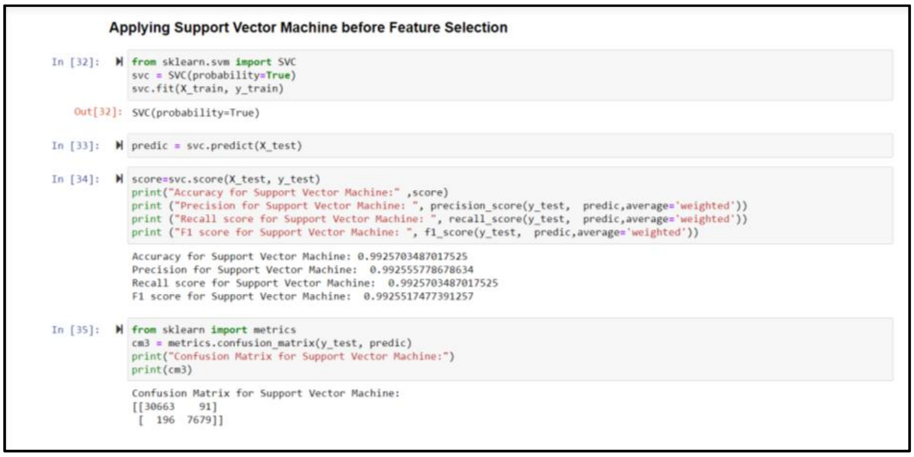


Figure 35 : Decision tree after feature selection



Applying Support Vector Machine before Feature Selection

```
In [32]: from sklearn.svm import SVC
         svc = SVC(probability=True)
         svc.fit(X_train, y_train)

Out[32]: SVC(probability=True)

In [33]: predic = svc.predict(X_test)

In [34]: score=svc.score(X_test, y_test)
         print("Accuracy for Support Vector Machine:" ,score)
         print ("Precision for Support Vector Machine: ", precision_score(y_test,  predic,average='weighted'))
         print ("Recall score for Support Vector Machine: ", recall_score(y_test,  predic,average='weighted'))
         print ("F1 score for Support Vector Machine: ", f1_score(y_test,  predic,average='weighted'))

         Accuracy for Support Vector Machine: 0.9925703487017525
         Precision for Support Vector Machine:  0.992555778678634
         Recall score for Support Vector Machine:  0.9925703487017525
         F1 score for Support Vector Machine:  0.9925517477391257

In [35]: from sklearn import metrics
         cm3 = metrics.confusion_matrix(y_test, predic)
         print("Confusion Matrix for Support Vector Machine:")
         print(cm3)

         Confusion Matrix for Support Vector Machine:
         [[30663    91]
          [  196  7679]]
```

Figure 36 : Applying Support Vector Machine before feature selection

Applying Support Vector Machine after Feature Selection

```
In [36]:  svc1 = SVC(probability=True)
          svc1.fit(X_train1, y_train1)
          predic1 = svc1.predict(X_test1)

In [37]:  score=svc1.score(X_test1, y_test1)
          print("Accuracy for Support Vector Machine:" ,score)
          print ("Precision for Support Vector Machine: ", precision_score(y_test1,  predic1,average='weighted'))
          print ("Recall score for Support Vector Machine: ", recall_score(y_test1,  predic1,average='weighted'))
          print ("F1 score for Support Vector Machine: ", f1_score(y_test1,  predic1,average='weighted'))

          Accuracy for Support Vector Machine: 0.9938647130394264
          Precision for Support Vector Machine:  0.9938582846859035
          Recall score for Support Vector Machine:  0.9938647130394264
          F1 score for Support Vector Machine:  0.9938478744183149

In [38]:  cm31 = metrics.confusion_matrix(y_test1, predic)
          print("Confusion Matrix for Support Vector Machine:")
          print(cm31)

          Confusion Matrix for Support Vector Machine:
          [[30663    91]
           [  196  7679]]
```

Figure 37 : Applying support vector machine after feature selection

Applying Logistic Regression before Feature Selection

```
In [39]:  from sklearn.linear_model import LogisticRegression
          logisticRegr = LogisticRegression()
          logisticRegr.fit(X_train, y_train)

          C:\Users\sahit\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed to conver
          ge (status=1):
          STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

          Increase the number of iterations (max_iter) or scale the data as shown in:
              https://scikit-learn.org/stable/modules/preprocessing.html
          Please also refer to the documentation for alternative solver options:
              https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
            n_iter_i = _check_optimize_result(

Out[39]:  LogisticRegression()

In [40]:  logisticRegr.predict(X_test[0:10])
          predictions = logisticRegr.predict(X_test)
          cm10 = metrics.confusion_matrix(y_test, predictions)
          print("Confusion Matrix for Logistic Regression:")
          print(cm10)

          Confusion Matrix for Logistic Regression:
          [[29697  1057]
           [ 1133  6742]]
```

Figure 38 : Applying Logistic Regression before feature selection

```
In [41]:  score = logisticRegr.score(X_test, y_test)
          print("Accuracy for Logistic Regression:" ,score)
          print ("Precision for Logistic Regression: ", precision_score(y_test,  predictions ,average='weighted'))
          print ("Recall score for Logistic Regression: ", recall_score(y_test,  predictions ,average='weighted'))
          print ("F1 score for Logistic Regression: ", f1_score(y_test,  predictions ,average='weighted'))

          Accuracy for Logistic Regression: 0.9433068420098889
          Precision for Logistic Regression:  0.9431124983542822
          Recall score for Logistic Regression:  0.9433068420098889
          F1 score for Logistic Regression:  0.9432043774208704

In [42]:  print(metrics.classification_report(y_test, predictions))

                        precision    recall  f1-score   support

                     0       0.96      0.97      0.96     30754
                     1       0.86      0.86      0.86      7875

              accuracy                           0.94     38629
             macro avg       0.91      0.91      0.91     38629
          weighted avg       0.94      0.94      0.94     38629
```

Figure 39 : Evaluation metric values for logistic regression

Applying Logistic Regression after Feature Selection

```
In [43]:  logisticRegr1= LogisticRegression()
          logisticRegr1.fit(X_train1, y_train1)

Out[43]:  LogisticRegression()

In [44]:  logisticRegr1.predict(X_test1[0:10])
          predictions = logisticRegr1.predict(X_test1)
          cm10 = metrics.confusion_matrix(y_test1, predictions)
          print("Confusion Matrix for Logistic Regression:")
          print(cm10)

          Confusion Matrix for Logistic Regression:
          [[29490  1264]
           [ 1436  6439]]

In [45]:  score = logisticRegr1.score(X_test1, y_test1)
          print("Accuracy for Logistic Regression:" ,score)
          print ("Precision for Logistic Regression: ", precision_score(y_test1,  predictions ,average='weighted'))
          print ("Recall score for Logistic Regression: ", recall_score(y_test1,  predictions ,average='weighted'))
          print ("F1 score for Logistic Regression: ", f1_score(y_test1,  predictions ,average='weighted'))

          Accuracy for Logistic Regression: 0.9301043257656165
          Precision for Logistic Regression:  0.929580438003303
          Recall score for Logistic Regression:  0.9301043257656165
          F1 score for Logistic Regression:  0.9298159143463754
```

Figure 40 : Applying logistic regression after feature selection

```
In [46]:  print(metrics.classification_report(y_test1, predictions))

              precision    recall  f1-score   support

           0       0.95      0.96      0.96     30754
           1       0.84      0.82      0.83      7875

    accuracy                           0.93     38629
   macro avg       0.89      0.89      0.89     38629
weighted avg       0.93      0.93      0.93     38629
```

Figure 41 : Classification report for logistic regression

Applying KNN before Feature Selection

```
In [47]:  from sklearn.neighbors import KNeighborsClassifier
          classi = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
          classi.fit(X_train, y_train)

Out[47]:  KNeighborsClassifier()

In [48]:  y_pre = classi.predict(X_test)
          cm2 = metrics.confusion_matrix(y_test, y_pre)
          print("Confusion Matrix for KNN:")
          print(cm2)

          Confusion Matrix for KNN:
          [[30616   138]
           [  189  7686]]

In [49]:  print(metrics.classification_report(y_test, y_pre))

              precision    recall  f1-score   support

           0       0.99      1.00      0.99     30754
           1       0.98      0.98      0.98      7875

    accuracy                           0.99     38629
   macro avg       0.99      0.99      0.99     38629
weighted avg       0.99      0.99      0.99     38629

In [50]:  print ("Accuracy for KNN : ", accuracy_score(y_test, y_pre))
          print ("Precision for KNN: ", precision_score(y_test,  y_pre ,average='weighted'))
          print ("Recall score for KNN: ", recall_score(y_test,  y_pre ,average='weighted'))
```

Figure 42 : Applying KNN before feature selection

Applying KNN after Feature Selection

```
In [51]:  classi1 = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
          classi1.fit(X_train1, y_train1)

Out[51]:  KNeighborsClassifier()

In [52]:  y_pre = classi1.predict(X_test1)
          cm2 = metrics.confusion_matrix(y_test1, y_pre)
          print("Confusion Matrix for KNN:")
          print(cm2)

          Confusion Matrix for KNN:
          [[30686    68]
           [   97  7778]]

In [53]:  print(metrics.classification_report(y_test1, y_pre))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     30754
           1       0.99      0.99      0.99      7875

    accuracy                           1.00     38629
   macro avg       0.99      0.99      0.99     38629
weighted avg       1.00      1.00      1.00     38629

In [54]:  print ("Accuracy for KNN : ", accuracy_score(y_test1, y_pre))
          print ("Precision for KNN: ", precision_score(y_test1,  y_pre ,average='weighted'))
          print ("Recall score for KNN: ", recall_score(y_test1,  y_pre ,average='weighted'))
          print ("F1 score for KNN: ", f1_score(y_test1,  y_pre ,average='weighted'))
```

Figure 43 : Applying KNN after feature selection

ROC Curve before Feature Selection

```
In [55]:  from sklearn.metrics import roc_curve
          predp1 = logisticRegr .predict_proba(X_test)
          predp2 = classi.predict_proba(X_test)
          predp3=classifier.predict_proba(X_test)
          predp4=clf.predict_proba(X_test)
          y_proba = svc.predict_proba(X_test)

In [56]:  fpr1, tpr1, thresh1 = roc_curve(y_test, predp1[:,1], pos_label=1)
          fpr2, tpr2, thresh2 = roc_curve(y_test, predp2[:,1], pos_label=1)
          fpr3, tpr3, thresh3 = roc_curve(y_test, predp3[:,1], pos_label=1)
          fpr4, tpr4, thresh4 = roc_curve(y_test, predp4[:,1], pos_label=1)
          fpr5, tpr5, thresh5 = roc_curve(y_test, y_proba[:,1], pos_label=1)
          # roc curve for tpr = fpr
          random_probs = [0 for i in range(len(y_test))]
          p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)

In [57]:  print("Logistic Regression TPR : ",tpr1)
          print(" FPR : ",fpr1)
          print("KNN TPR : ",tpr2)
          print(" FPR : ",fpr2)
          print("Naive Bayes TPR : ",tpr3)
          print(" FPR : ",fpr3)
          print("Decision Tree TPR : ",tpr4)
          print(" FPR : ",fpr4)
          print("SVM TPR : ",tpr5)
          print(" FPR : ",fpr5)

          9.87555556e-01 9.87555556e-01 9.87809524e-01 9.87809524e-01
          9.87936508e-01 9.87936508e-01 9.88063492e-01 9.88063492e-01
```

Figure 44 : ROC Curve before feature selection

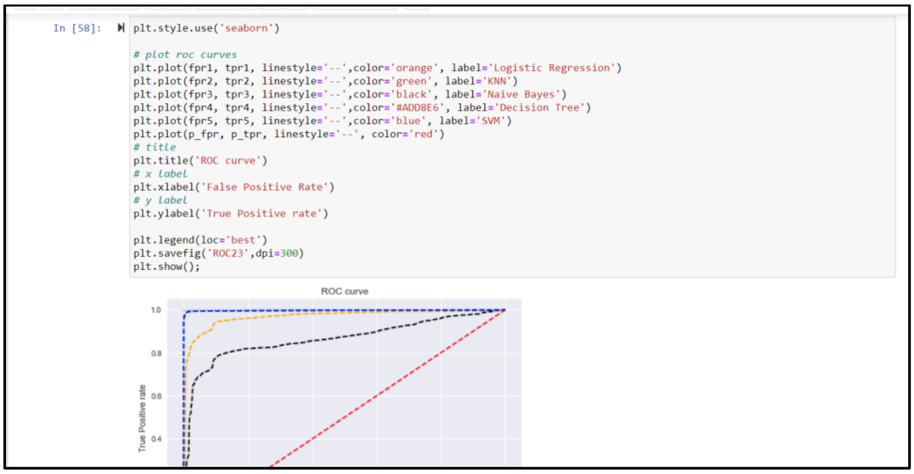


Figure 45 : Plotting ROC curve

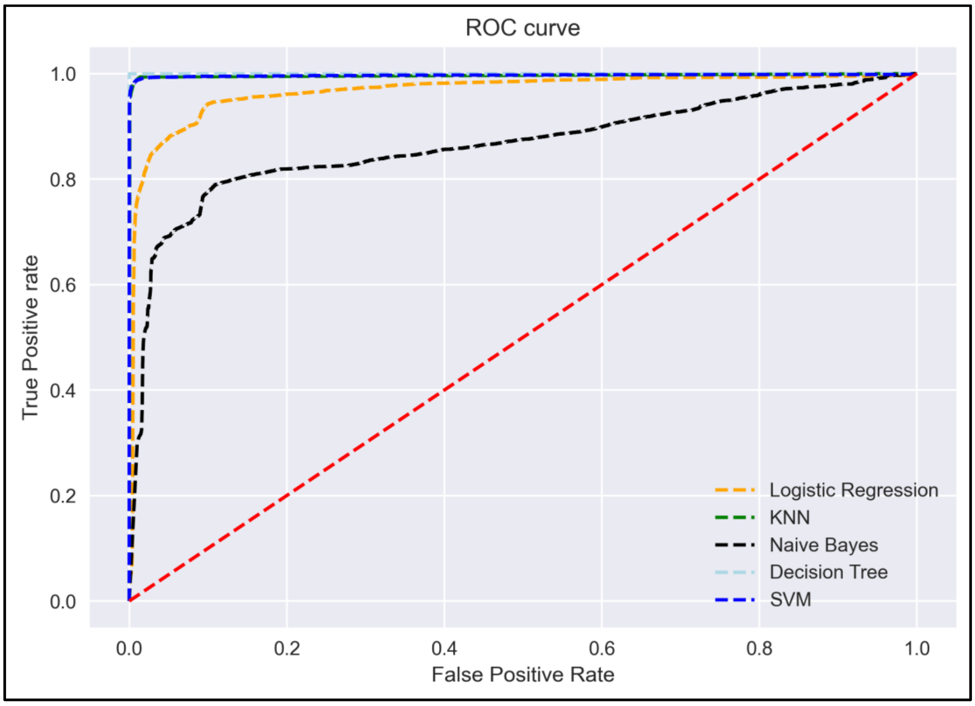


Figure 46 : ROC curve before feature selection

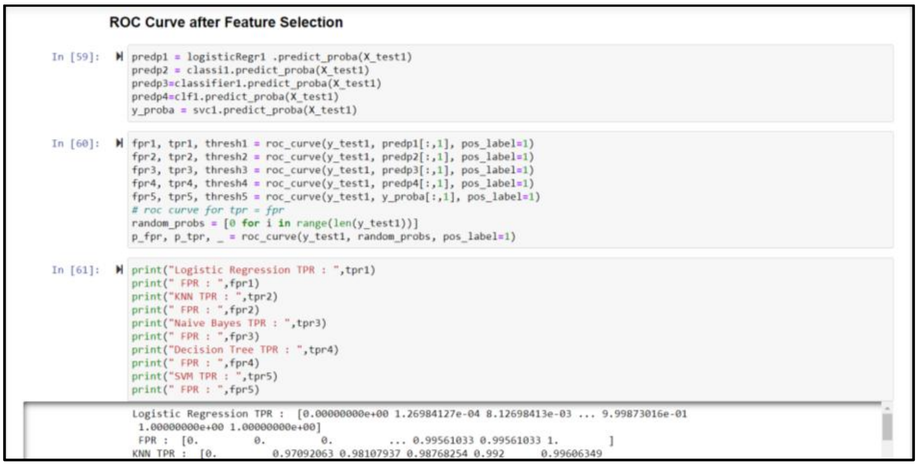


Figure 47 : ROC Curve after feature selection

```
In [62]: plt.style.use('seaborn')

         # plot roc curves
         plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='Logistic Regression')
         plt.plot(fpr2, tpr2, linestyle='--',color='red', label='KNN')
         plt.plot(fpr3, tpr3, linestyle='--',color='green', label='Naive Bayes')
         plt.plot(fpr4, tpr4, linestyle='--',color='#ADD8E6', label='Decision Tree')
         plt.plot(fpr5, tpr5, linestyle='--',color='blue', label='SVM')
         plt.plot(p_fpr, p_tpr, linestyle='--', color='yellow')
         # title
         plt.title('ROC curve after Feature Selection')
         # x label
         plt.xlabel('False Positive Rate')
         # y label
         plt.ylabel('True Positive rate')

         plt.legend(loc='best')
         plt.savefig('ROC10',dpi=300)
         plt.show();
```


Figure 48 : Plotting ROC Curve

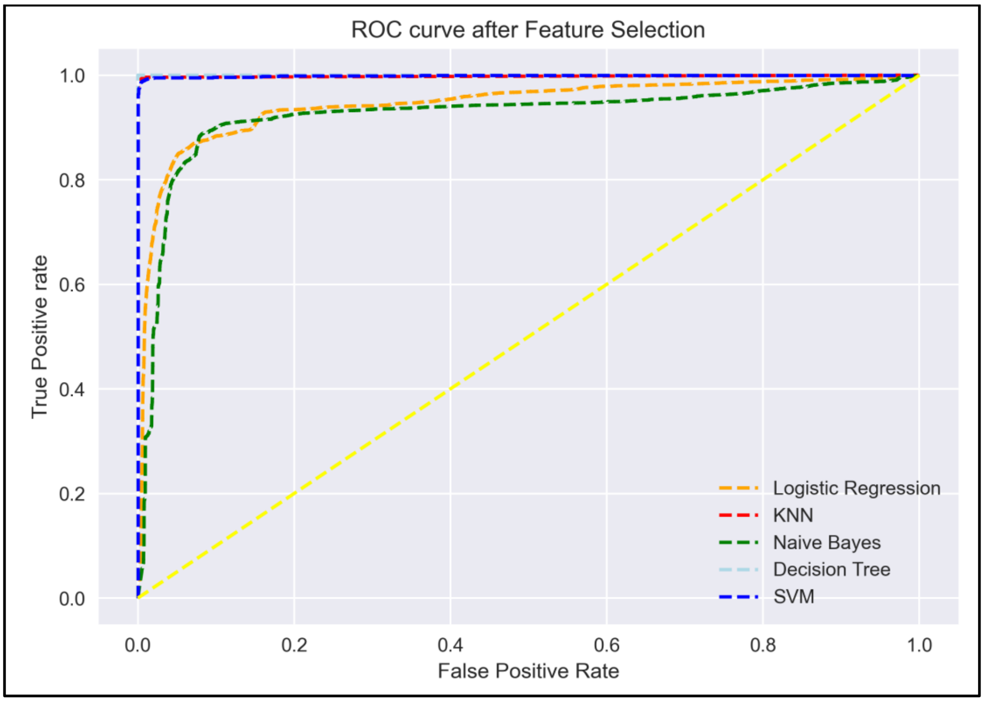


Figure 49 : ROC Curve after feature selection

4. Conclusion

Traffic classification using machine learning algorithms was not previously considered relevant. With the rise of traffic encryption and anonymity services like Tor and VPN in the darknet, machine learning techniques for encrypted traffic classification should be viewed as one of the most important methods for identifying this traffic. We gave an outline of machine learning categorization for darknet traffic networks in this research. We'll start with some technical background on the darknet and machine learning. To summarize, there are still many aspects of the machine learning classification process that could be researched and improved in order to reveal the truth about network privacy protection.

5. References

1) http://205.174.165.80/CICDataset/CICDarknet2020/Dataset/
2) https://www.unb.ca/cic/datasets/darknet2020.html
3) https://en.wikipedia.org/wiki/Machine_learning
4) https://www.ibm.com/cloud/learn/machine-learning#:~:text=Machine%20learning%20is%20a%20branch,learn%2C%20gradually%20improving%20its%20accuracy.&text=Machine%20learning%20is%20an%20important,growing%20field%20of%20data%20science.
5) https://en.wikipedia.org/wiki/Darknet
6) https://en.wikipedia.org/wiki/Virtual_private_network
7) https://en.wikipedia.org/wiki/Tor_(anonymity_network)
8) https://en.wikipedia.org/wiki/Supervised_learning
9) https://en.wikipedia.org/wiki/Unsupervised_learning
10) https://www.analytixlabs.co.in/blog/data-acquisition/
11) https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825
12) https://en.wikipedia.org/wiki/Decision_tree_learning
13) https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
14) https://en.wikipedia.org/wiki/Logistic_regression
15) https://en.wikipedia.org/wiki/Support-vector_machine
16) https://en.wikipedia.org/wiki/Naive_Bayes_classifier
17) https://machinelearningmastery.com/rfe-feature-selection-in-python/
18) https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623
19) https://builtin.com/data-science/regression-machine-learning#:~:text=Regression%20is%20a%20supervised%20machine,are%20variance%2C%20bias%20and%20error.
20) https://en.wikipedia.org/wiki/Feature_selection
21) https://www.upgrad.com/blog/naive-bayes-classifier/#Advantages_of_Naive_Bayes
22) https://en.wikipedia.org/wiki/Random_forest
23) https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226
24) https://en.wikipedia.org/wiki/Receiver_operating_characteristic