Detection of DDoS Attack in Software Defined Networking Using KNN

Submitted by -

D.VENKATESH TAMILSELVAM

Detection of DDoS attack in software defined networking using KNN

Abstract

Software-Defined Networking (SDN) is the networking architecture defined by the software program. In SDN, network traffic is controlled by the software which directs the traffic between the hosts. This is unlike the current network architecture where the traffic control is hardware-based which is defined by the switches, routers and, other network infrastructure. The centralized SDN architecture is designed in such a way that SDN switch is comprised of only the data plane and the control plane is moved to another entity known as the controller.

The controller acts as the brain of the network and it is a network wide centralized entity where all the switches in the network abide by the decision made by the controller.

SDN helps network engineers to monitor the network expeditely, control the network from a central point, identify malicious traffic and link failure in easy and efficient manner. Besides such flexibility provided by SDN, it is also vulnerable to attacks such as DDoS which can halt the complete network.

To mitigate this attack, in this project we proposes to classify the benign traffic from DDoS attack traffic by using Machine learning technique. We use KNN for classification process and hence we can able to get an accuracy of above 95% during training and Testing

INTRODUCTION

1.1 DDoS ATTACK

A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt the normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic.

DDoS attacks achieve effectiveness by utilizing multiple compromised computer systems as sources of attack traffic. Exploited machines can include computers and other networked resources such as IoT devices.

From a high level, a DDoS attack is like an unexpected traffic jam clogging up the highway, preventing regular traffic from arriving at its destination.

How does a DDoS attack work?

DDoS attacks are carried out with networks of Internet-connected machines.

These networks consist of computers and other devices (such as IoT devices)which have been infected with malware, allowing them to be controlled remotely by an attacker. These individual devices are referred to as bots (or zombies), and a group of bots is called a botnet.

Once a botnet has been established, the attacker is able to direct an attack by sending remote instructions to each bot.

When a victim's server or network is targeted by the botnet, each bot sends requests to the target's IP address, potentially causing the server or network to become overwhelmed, resulting in a denial-of-service to normal traffic.

Because each bot is a legitimate Internet device, separating the attack traffic from normal traffic can be difficult.

How to identify a DDoS attack

The most obvious symptom of a DDoS attack is a site or service suddenly becoming slow or unavailable. But since a number of causes — such a legitimate spike in traffic — can create similar performance issues, further investigation is usually required. Traffic analytics tools can help you spot some of these telltale signs of a DDoS attack:

- Suspicious amounts of traffic originating from a single IP address or IP range
- A flood of traffic from users who share a single behavioral profile, such as device type, geolocation, or web browser version
- An unexplained surge in requests to a single page or endpoint
- Odd traffic patterns such as spikes at odd hours of the day or patterns that appear to be unnatural (e.g. a spike every 10 minutes)

There are other, more specific signs of DDoS attack that can vary depending on the type of attack.

What are some common types of DDoS attacks?

Different types of DDoS attacks target varying components of a network connection. In order to understand how different DDoS attacks work, it is necessary to know how a network connection is made.

A network connection on the Internet is composed of many different components or “layers”. Like building a house from the ground up, each layer in the model has a different purpose.

The OSI model, shown below, is a conceptual framework used to describe network connectivity in 7 distinct layers.

While nearly all DDoS attacks involve overwhelming a target device or network with traffic, attacks can be divided into three categories. An attacker may use one or more different attack vectors, or cycle attack vectors in response to counter measures taken by the target.

Application layer attacks

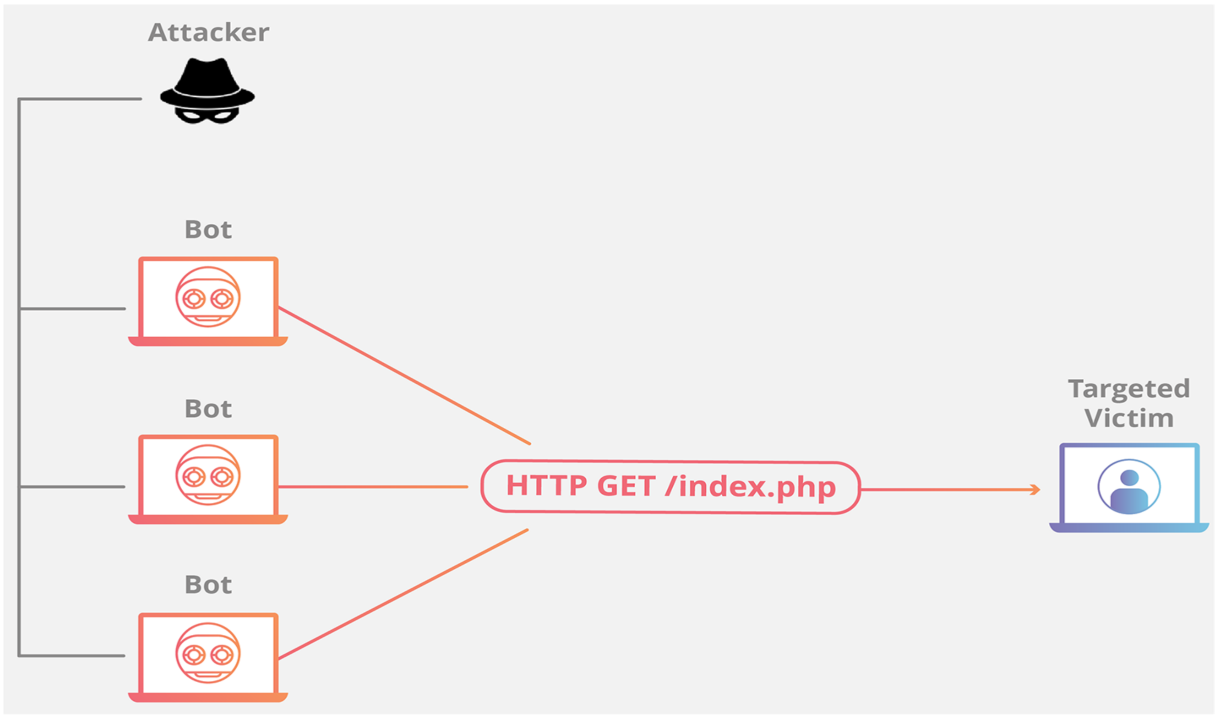
The goal of the attack:

Sometimes referred to as a layer 7 DDoS attack (in reference to the 7th layer of the OSI model), the goal of these attacks is to exhaust the target's resources to create a denial-of-service.

The attacks target the layer where web pages are generated on the server and delivered in response to HTTP requests. A single HTTP request is computationally cheap to execute on the client side, but it can be expensive for the target server to respond to, as the server often loads multiple files and runs database queries in order to create a web page.

Layer 7 attacks are difficult to defend against, since it can be hard to differentiate malicious traffic from legitimate traffic.

Application layer attack example:



HTTP flood

This attack is similar to pressing refresh in a web browser over and over on many different computers at once – large numbers of HTTP requests flood the server, resulting in denial-of-service.

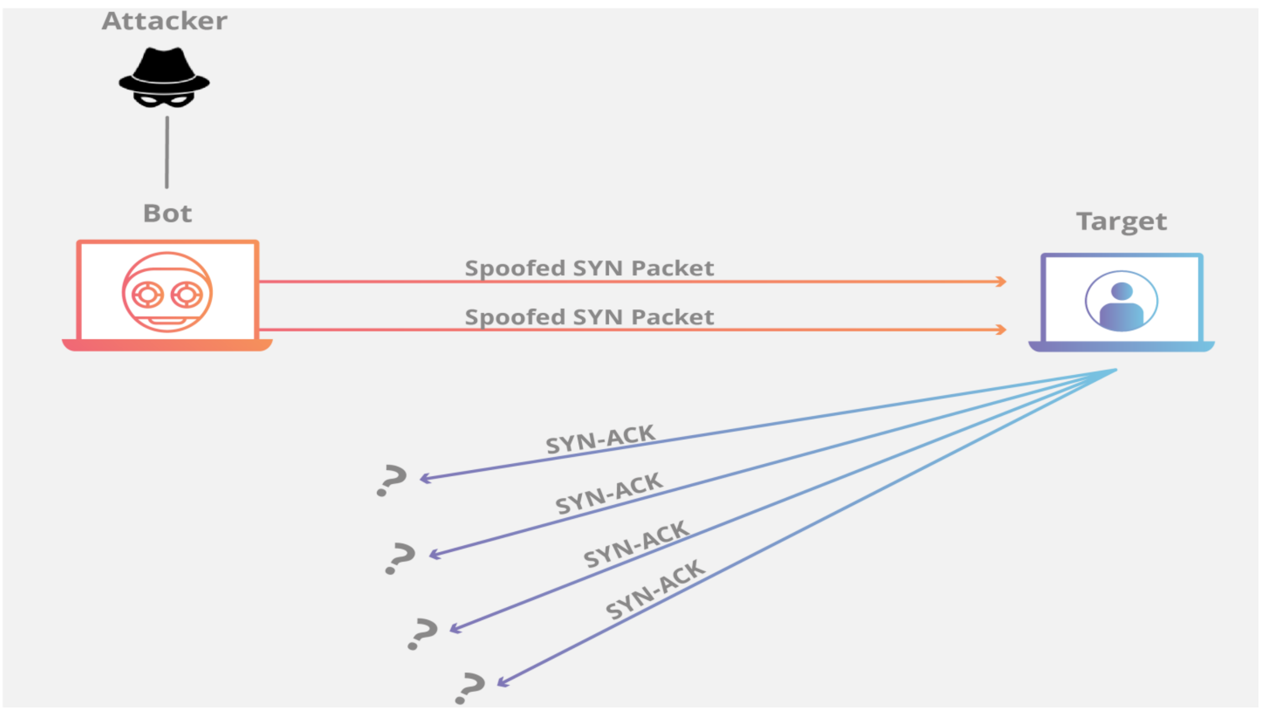
This type of attack ranges from simple to complex.

Simpler implementations may access one URL with the same range of attacking IP addresses, referrers and user agents. Complex versions may use a large number of attacking IP addresses, and target random urls using random referrers and user agents.

Protocol attacks

The goal of the attack*:*
Protocol attacks, also known as a state-exhaustion attacks, cause a service disruption by over-consuming server resources and/or the resources of network equipment like firewalls and load balancers.
Protocol attacks utilize weaknesses in layer 3 and layer 4 of the protocol stack to render the target inaccessible.

Protocol attack example:



SYN flood

A SYN Flood is analogous to a worker in a supply room receiving requests from the front of the store.

The worker receives a request, goes and gets the package, and waits for confirmation before bringing the package out front. The worker then gets many more package requests without confirmation until they can't carry any more packages, become overwhelmed, and requests start going unanswered.

This attack exploits the TCP handshake — the sequence of communications by which two computers initiate a network connection — by sending a target a large number of TCP "Initial Connection Request" SYN packets with spoofed source IP addresses.

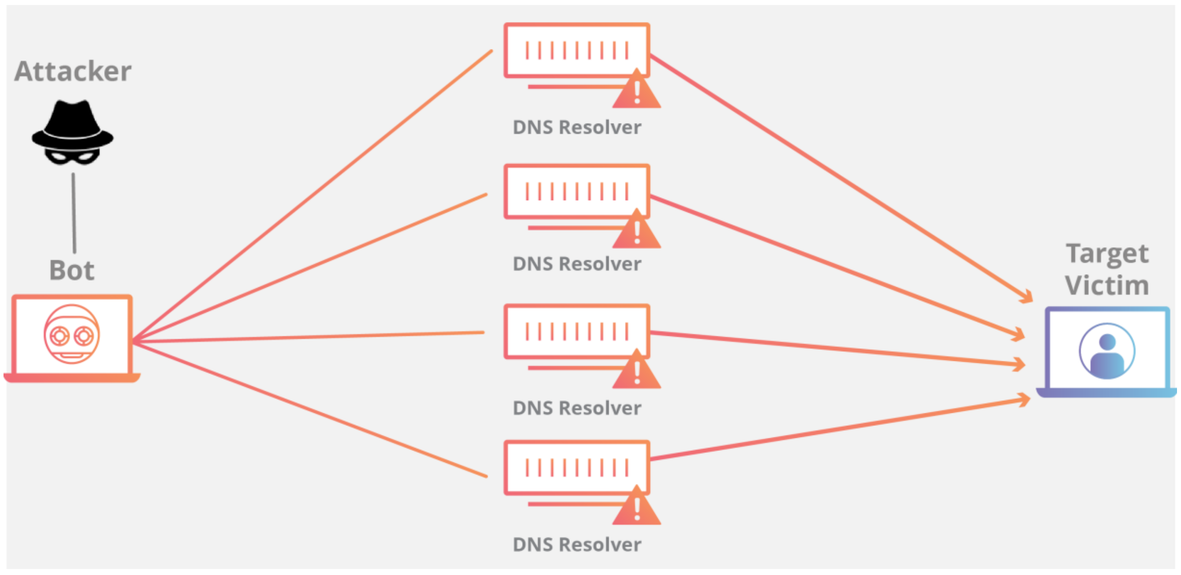
The target machine responds to each connection request and then waits for the final step in the handshake, which never occurs, exhausting the target's resources in the process.

Volumetric attacks

The goal of the attack:

This category of attacks attempts to create congestion by consuming all available bandwidth between the target and the larger Internet. Large amounts of data are sent to a target by using a form of amplification or another means of creating massive traffic, such as requests from a botnet.

Amplification example:



DNS Amplification

A DNS amplification is like if someone were to call a restaurant and say “I’ll have one of everything, please call me back and repeat my whole order,” where the callback number actually belongs to the victim. With very little effort, a long response is generated and sent to the victim.

By making a request to an open DNS server with a spoofed IP address (the IP address of the victim), the target IP address then receives a response from the server.

What is the process for mitigating a DDoS attack?

The key concern in mitigating a DDoS attack is differentiating between attack traffic and normal traffic.

For example, if a product release has a company’s website swamped with eager customers, cutting off all traffic is a mistake. If that company suddenly has a surge in traffic from known attackers, efforts to alleviate an attack are probably necessary.

The difficulty lies in telling the real customers apart from the attack traffic.

In the modern Internet, DDoS traffic comes in many forms. The traffic can vary in design from un-spoofed single source attacks to complex and adaptive multi-vector attacks.

A multi-vector DDoS attack uses multiple attack pathways in order to overwhelm a target in different ways, potentially distracting mitigation efforts on any one trajectory.

An attack that targets multiple layers of the protocol stack at the same time, such as a DNS amplification (targeting layers 3/4) coupled with an HTTP flood (targeting layer 7) is an example of multi-vector DDoS.

Mitigating a multi-vector DDoS attack requires a variety of strategies in order to counter different trajectories.

Generally speaking, the more complex the attack, the more likely it is that the attack traffic will be difficult to separate from normal traffic - the goal of the attacker is to blend in as much as possible, making mitigation efforts as inefficient as possible.

Mitigation attempts that involve dropping or limiting traffic indiscriminately may throw good traffic out with the bad, and the attack may also modify and adapt to circumvent countermeasures. In order to overcome a complex attempt at disruption, a layered solution will give the greatest benefit.

Blackhole routing

One solution available to virtually all network admins is to create a blackhole route and funnel traffic into that route. In its simplest form, when blackhole filtering is implemented without specific restriction criteria, both legitimate and malicious network traffic is routed to a null route, or blackhole, and dropped from the network.

If an Internet property is experiencing a DDoS attack, the property's Internet service provider (ISP) may send all the site's traffic into a blackhole as a defense. This is not an ideal solution, as it effectively gives the attacker their desired goal: it makes the network inaccessible.

Rate limiting

Limiting the number of requests a server will accept over a certain time window is also a way of mitigating denial-of-service attacks.

While rate limiting is useful in slowing web scrapers from stealing content and for mitigating brute force login attempts, it alone will likely be insufficient to handle a complex DDoS attack effectively.

Nevertheless, rate limiting is a useful component in an effective DDoS mitigation strategy. Learn about Cloudflare's rate limiting

Web application firewall

A Web Application Firewall (WAF) is a tool that can assist in mitigating a layer 7 DDoS attack. By putting a WAF between the Internet and an origin server, the WAF may act as a reverse proxy, protecting the targeted server from certain types of malicious traffic.

By filtering requests based on a series of rules used to identify DDoS tools, layer 7 attacks can be impeded. One key value of an effective WAF is the ability to quickly implement custom rules in response to an attack. Learn about Cloudflare's WAF.

Anycast network diffusion

This mitigation approach uses an Anycast network to scatter the attack traffic across a network of distributed servers to the point where the traffic is absorbed by the network.

Like channeling a rushing river down separate smaller channels, this approach spreads the impact of the distributed attack traffic to the point where it becomes manageable, diffusing any disruptive capability.

The reliability of an Anycast network to mitigate a DDoS attack is dependent on the size of the attack and the size and efficiency of the network. An important part of the DDoS mitigation implemented by Cloudflare is the use of an Anycast distributed network.

Cloudflare has a 100 Tbps network, which is an order of magnitude greater than the largest DDoS attack recorded.

If you are currently under attack, there are steps you can take to get out from under the pressure. If you are on Cloudflare already, you can follow these steps to mitigate your attack.

The DDoS protection that we implement at Cloudflare is multifaceted in order to mitigate the many possible attack vectors. Learn more about Cloudflare's DDoS protection and how it works.

1.2 Prediction

“Prediction” refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome, such as whether or not a customer will churn in 30 days. The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be.

The word “prediction” can be misleading. In some cases, it really does mean that you are predicting a future outcome, such as when you’re using machine learning to determine the next best action in a marketing campaign. Other times, though, the “prediction” has to do with, for example, whether or not a transaction that already occurred was fraudulent. In that case, the transaction already happened, but you’re making an educated guess about whether or not it was legitimate, allowing you to take the appropriate action.

Why are Predictions Important?

Machine learning model predictions allow businesses to make highly accurate guesses as to the likely outcomes of a question based on historical data, which can be about all kinds of things – customer churn likelihood, possible fraudulent activity, and more. These provide the business with insights that result in tangible business value. For example, if a model predicts a customer is likely to churn, the business can target them with specific communications and outreach that will prevent the loss of that customer.

What are Prediction Explanations in Machine Learning?

Prediction explanations in machine learning explain how or why an AI platform arrived at an outcome. In the past, these models did not explain how a decision was made. This lack of clarity causes the "black box" syndrome, where you have predictions but are unsure of which feature variables affect a model's outcomes.

Machine learning models learn how to make decisions based on rules that it created while analyzing the training dataset. Prediction explanations allow us to understand these rules and how they are applied to new data, as well as what features are the most valuable considerations in determining the output.

The prediction explanation assigns each input value a measure of importance, and if you take the sum of all of them, it will add up to 100%.

For decision tree models, the prediction explanation follows the prediction path. Still, for others like regression, it is calculated based on aggregating the results of many predictions that use random variations of input data.

Why are Prediction Explanations Helpful?

Understanding how your AI platforms interpret feature variables to arrive at predictions is essential to making data-driven decisions. By learning what inputs most contribute to the outcomes, you can gain insights into what is driving things like consumer trends or revenue growth.

Prediction explanations are required by regulations when machine learning models are used in industries like finance. Loan issuers must explain how they arrived at a credit decision so that they can prove it was fair – and provide support for why they declined a loan application.

They are also crucial in healthcare, as medical professionals must provide explanations for diagnosis. This is the only way that doctors can be confident enough to make a life-or-death decision, and so that they can convey that decision path to the patient and their family.

SYSTEM IMPLEMENTATION

3.1 EXISITING SYSTEM

- The In the existing system they did not used Machine learning algorithm, they just done comparison searching and sorting.
- And hence their accuracy to predict the network change is very low.

3.2 PROPOSED SYSREM

In the proposed system we use a Machine learning model which is called as K-Nearest Neighbour which is suitable for Accurate Prediction. By using KNN, we can predict the DDoS attack with more than 96% of accuracy.

3.3 DATA COLLECTION

We collected the DDoS datasets

Feature	**Description**
PkSeqID	Row Identifier
Stime	Record start time
Flgs	Flow state flags seen in transactions
flgs_number	Numerical representation of feature flags
Proto	Textual representation of transaction protocols present in network flow
proto_number	Numerical representation of feature proto
Saddr	Source IP address
Sport	Source port number
Daddr	Destination IP address
Dport	Destination port number
Pkts	Total count of packets in transaction
Bytes	Totan number of bytes in transaction
State	Transaction state

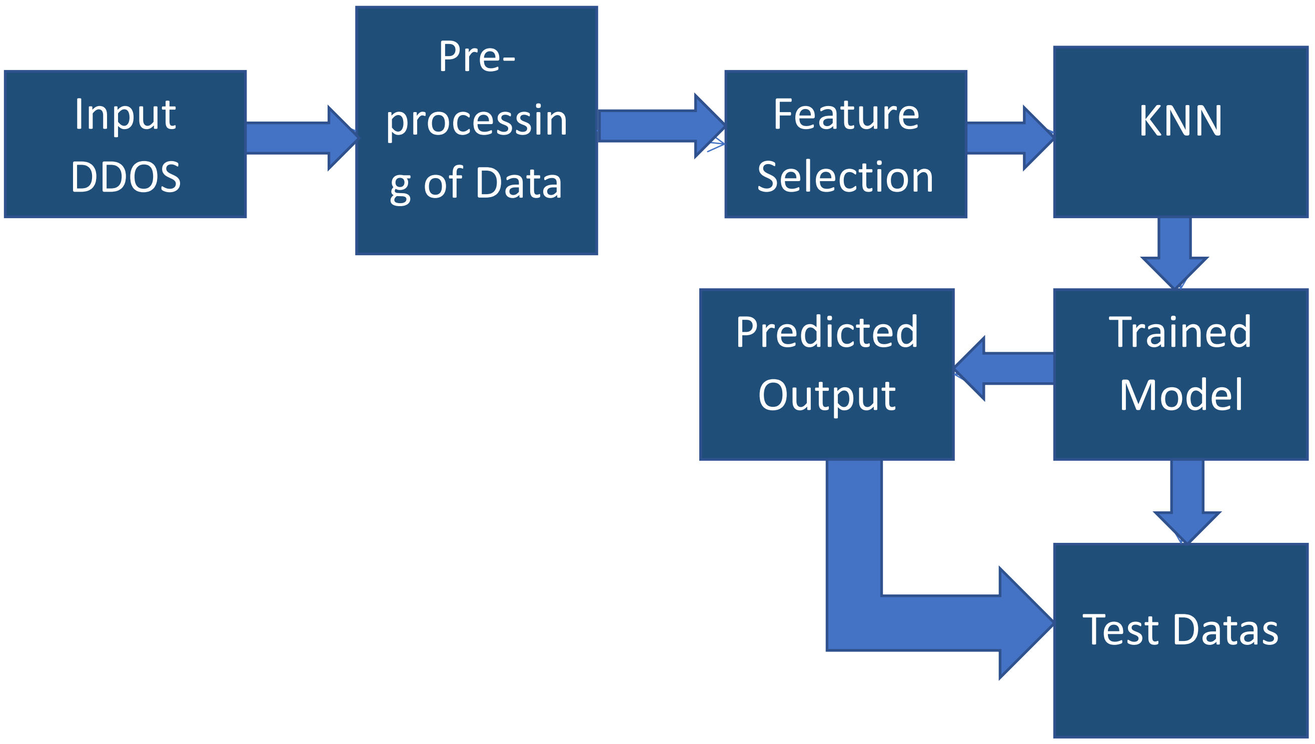
state_number	Numerical representation of feature state
Ltime	Record last time
Seq	Argus sequence number
Dur	Record total duration
Mean	Average duration of aggregated records
Stddev	Standard deviation of aggregated records
Sum	Total duration of aggregated records
Min	Minimum duration of aggregated records
Max	Maximum duration of aggregated records
Spkts	Source-to-destination packet count
Dpkts	Destination-to-source packet count
Sbytes	Source-to-destination byte count
Dbytes	Destination-to-source byte count
Rate	Total packets per second in transaction
Srate	Source-to-destination packets per second
Drate	Destination-to-source packets per second
TnBPSrcIP	Total Number of bytes per source IP
TnBPDstIP	Total Number of bytes per Destination IP.
TnP_PSrcIP	Total Number of packets per source IP.
TnP_PDstIP	Total Number of packets per Destination IP.
TnP_PerProto	Total Number of packets per protocol.
TnP_Per_Dport	Total Number of packets per dport
AR_P_Proto_P_SrcIP	Average rate per protocol per Source IP. (calculated by pkts/dur)

AR_P_Proto_P_DstIP	Average rate per protocol per Destination IP.
N_IN_Conn_P_SrcIP	Number of inbound connections per source IP.
N_IN_Conn_P_DstIP	Number of inbound connections per destination IP.
AR_P_Proto_P_Sport	Average rate per protocol per sport
AR_P_Proto_P_Dport	Average rate per protocol per dport
Pkts_P_State_P_Protocol_P_DestIP	Number of packets grouped by state of flows and protocols per destination IP.
Pkts_P_State_P_Protocol_P_SrcIP	Number of packets grouped by state of flows and protocols per source IP.
Attack	Class label: 0 for Normal traffic, 1 for Attack Traffic
Category	Traffic category
Subcategory	Traffic subcategory

3.4 DATA PRE-PROCESSING

At first the dataset is fetched by using pandas library and then we save the datas inside a pandas dataframe, At first this dataset consists of lots of null values, then we drop all the null values, because our Machine learning model cannot able to process null values

Proposed system Architecture



Module Description

- At first the datas are getted from the *input dataset* by using pandas library.
- The *Pre-processing* the data by droping null values,
- we make *feature selection* by selecting input features for feeding it in the *KNN module,* we design a KNN model which can able to give high accuracy,
- the extracted features are inserted in to the KNN model and the machine gets trained. After training we predict *DDos attack datas by feeding test datas* into the model.

SIMULATION RESULTS& DISCUSSION

4.1 SOFTWARE DESCRIPTION

The Python language had a humble beginning in the late 1980s when a Dutchman Guido Von Rossum started working on a fun project, which would be a successor to ABC language with better exception handling and capability to interface with OS Amoeba at Centrum Wiskunde and Informatica. It first appeared in 1991. Python 2.0 was released in the year 2000 and Python 3.0 was released in the year 2008. The language was named Python after the famous British television comedy show Monty Python's Flying Circus, which was one of Guido's favorite television programmes. Here we will see why Python has suddenly influenced our lives and the various applications that use Python and its implementations.

In this chapter, you will be learning the basic installation steps that are required to perform on different platforms (that is Windows, Linux, and Mac), about environment variables, setting up of environment variables, file formats, Python interactive shell, basic syntaxes and finally printing out formatted output.

4.1.1 Why Python?

Now you might be suddenly bogged with the question, why Python? According to Institute of Electrical and Electronics Engineers (IEEE) 2016 ranking Python ranked third after C and Java. As per Indeed.com's data of 2016, the Python job market search ranked fifth. Clearly, all the data points to the ever rising demand in the job market for Python. It’s a cool language if you want to learn just for fun or if you want to build your career around Python, you will adore the language. At school level, many schools have started including Python programming for kids. With new technologies taking the market by surprise Python has been playing a dominant role. Whether it is cloud platform, mobile app development, Big Data, IoT with Raspberry Pi, or the new Block chain technology, Python is being seen as a niche language platform to develop and deliver a scalable and robust applications.

Some key features of the language are:

- Python programs can run on any platform, you can carry code created in Windows machine and run it on Mac or Linux
- Python has inbuilt large library with prebuilt and portable functionality, also known as the standard library
- Python is an expressive language
- Python is free and open source

- Python code is about one third of the size of equivalent C++ and Java code
- Python can be both dynamically and strongly typed--dynamically typed means it is a type of variable that is interpreted at runtime, which means, in Python, there is no need to define the type (int or float) of the variable

Python applications

One of the most famous platforms where Python is extensively used is YouTube. The other places where you will find Python being extensively used are the special effects in Hollywood movies, drug evolution and discovery, traffic control systems, ERP systems, cloud hosting, e-commerce platform, CRM systems, and whatever field you can think of.

Versions

At the time of writing this book, two main versions of the Python programming language were available in the market, which are Python 2.x and Python 3.x. The stable release as of writing the book were Python 2.7.13 and Python 3.6.0.

Implementations of Python

Major implementations include CPython, Jython, IronPython, MicroPython, and PyPy.

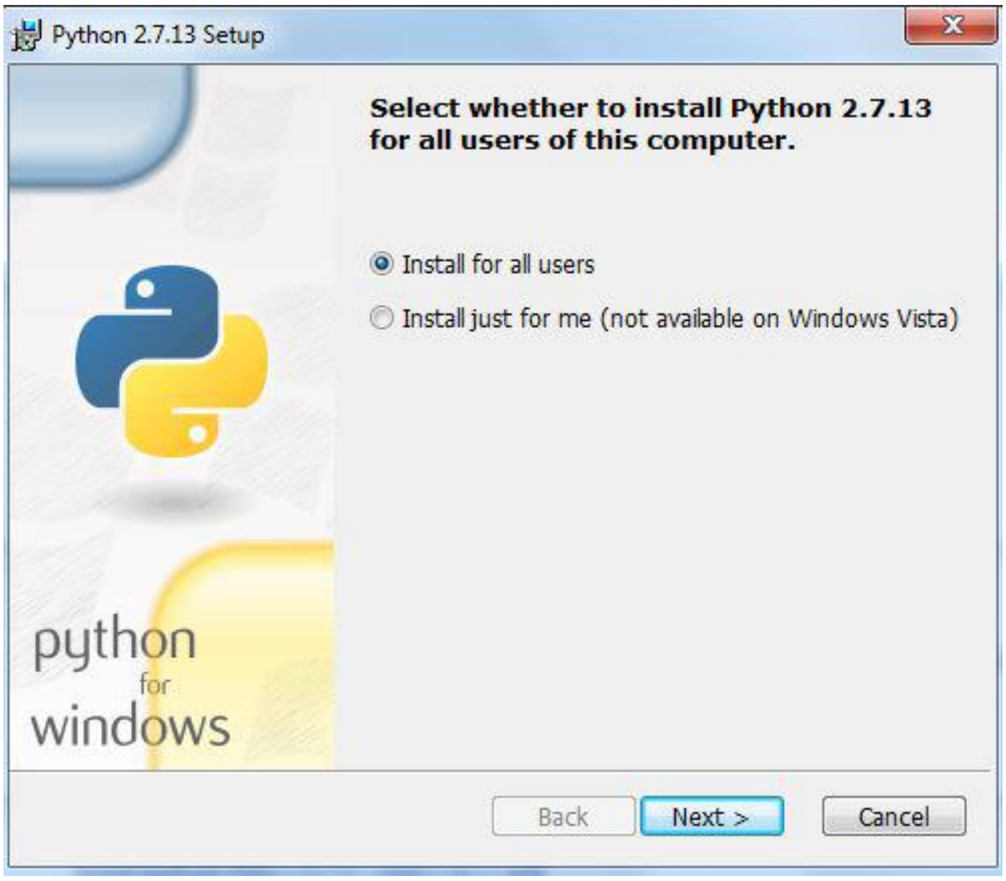
4.1.2 Installation

Here we will look forward to the installation of Python on three different OS platforms, namely, Windows, Linux, and Mac OS. Let's begin with the Windows platform.

Installation on Windows platform

Python 2.x can be downloaded from h t t p s ://w ww . p y t h o n . o r g /d o w n l o a d s . The installer is simple and easy to install. Perform the following steps to install the setup:

1. Once you click on setup installer, you will get a small window on your desktop screen as shown here; click on **Next**:

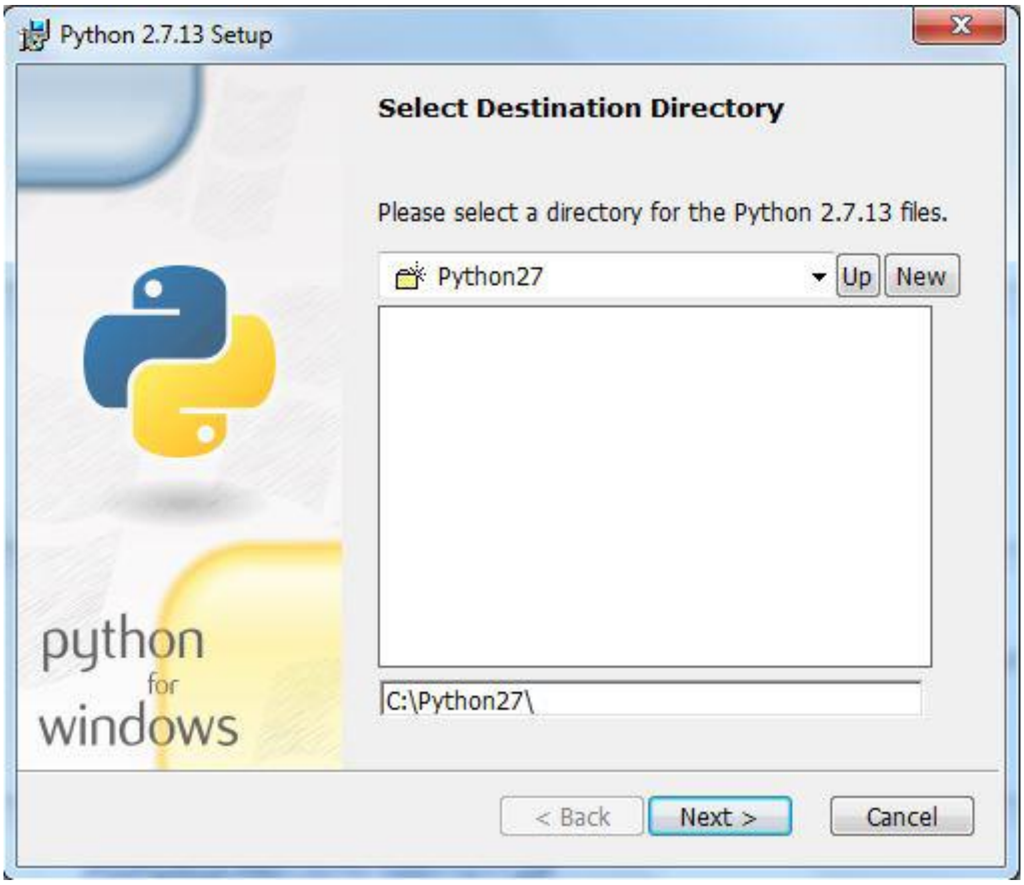

Python 2.7.13 Setup

Select whether to install Python 2.7.13 for all users of this computer.

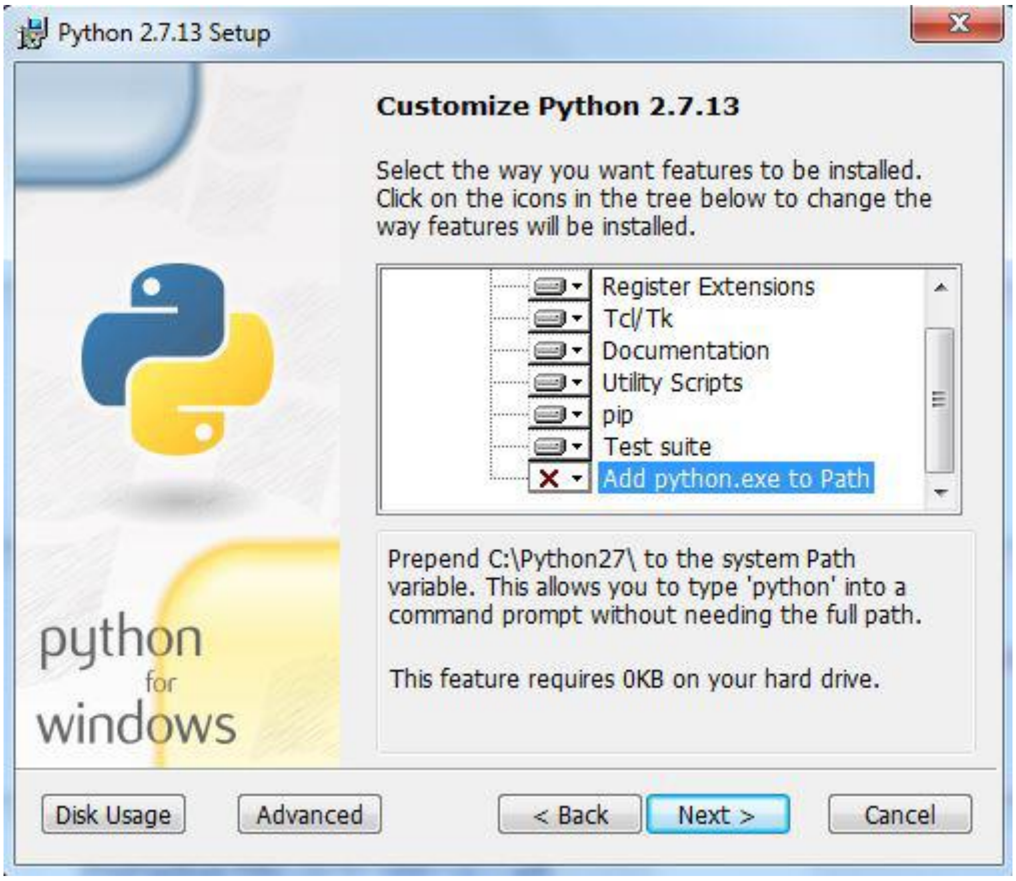
(●) Install for all users

( ) Install just for me (not available on Windows Vista)

2. Provide a suitable installation folder to install Python. If you don't provide the installation folder, then the installer will automatically create an installation folder for you, as shown in the following screenshot. Click on **Next**:


Python 2.7.13 Setup
Select Destination Directory
Please select a directory for the Python 2.7.13 files.
Python27
Up
New
C:\Python27\
python for windows
< Back
Next >
Cancel

3. After completion of step 2, you will get a window to customize Python as shown in the preceding screenshot. Notice that the **Add python.exe to Path** option has been marked **x**. Select this option to add it to system path variable (which will be explained later in the chapter), and click on **Next**:


Python 2.7.13 Setup

Customize Python 2.7.13

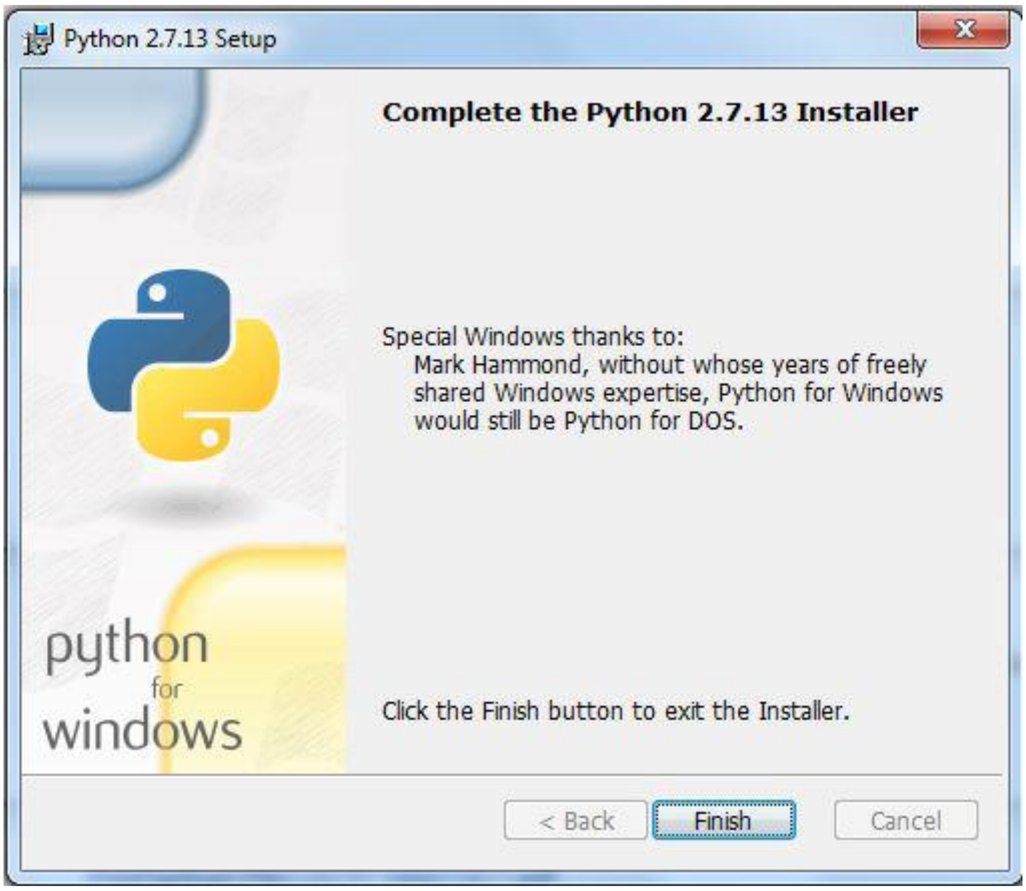
Select the way you want features to be installed.
Click on the icons in the tree below to change the way features will be installed.

- Register Extensions
- Tcl/Tk
- Documentation
- Utility Scripts
- pip
- Test suite
- ✗ Add python.exe to Path

Prepend C:\Python27\ to the system Path variable. This allows you to type 'python' into a command prompt without needing the full path.

This feature requires 0KB on your hard drive.

4. Finally, click on **Finish** to complete the installation:


Python 2.7.13 Setup

Complete the Python 2.7.13 Installer

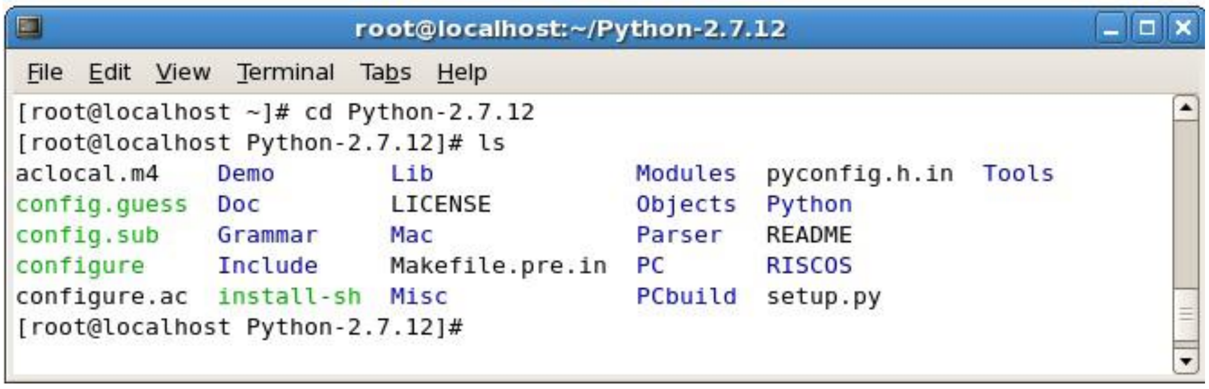
Special Windows thanks to:
Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

Click the Finish button to exit the Installer.

Installation on Linux platform

These days most of the Linux-based systems come preloaded with Python, so in most cases, you do not need to install it separately. However, if you do not find your desired version of Python on the Linux platform, you can download your desired version for a particular Linux platform from the site h t t p s ://w ww . p y t h o n . o r g /d o w n l o a d s /s o u r c e /. Perform the following steps:

1. Extract the compressed file using the tar –xvzfpython_versionx.x command.
2. Browse the directory of the compressed file as shown in the screenshot:


root@localhost:~/Python-2.7.12
File Edit View Terminal Tabs Help

```
[root@localhost ~]# cd Python-2.7.12
[root@localhost Python-2.7.12]# ls
aclocal.m4     Demo        Lib              Modules   pyconfig.h.in  Tools
config.guess   Doc         LICENSE          Objects   Python
config.sub     Grammar     Mac              Parser    README
configure      Include     Makefile.pre.in  PC        RISCOS
configure.ac   install-sh  Misc             PCbuild   setup.py
[root@localhost Python-2.7.12]#
```

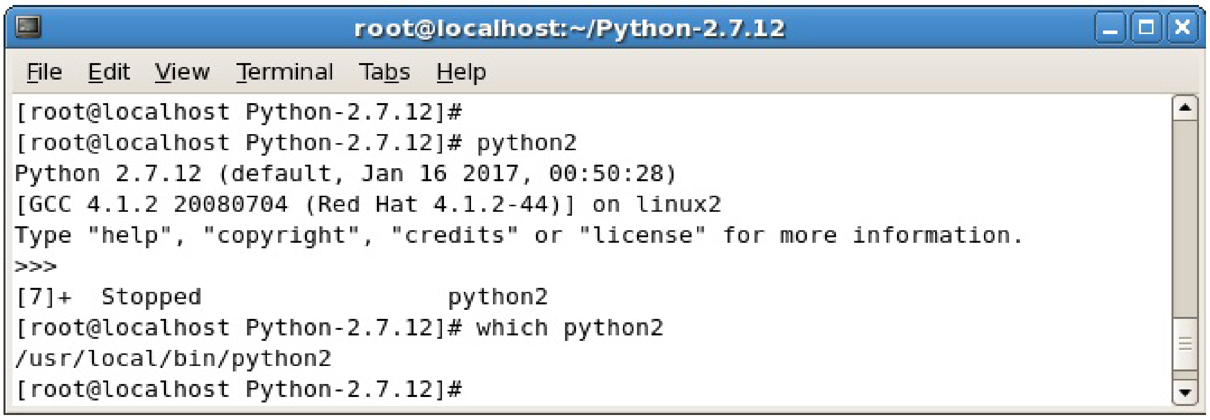
3. Run the following commands:

[root@localhost Python-2.7.12]# ./configure

[root@localhost Python-2.7.12]# make

[root@localhost Python-2.7.12]# make install

4. Use the command as shown in screenshot to ensure that Python is running:


root@localhost:~/Python-2.7.12
File Edit View Terminal Tabs Help

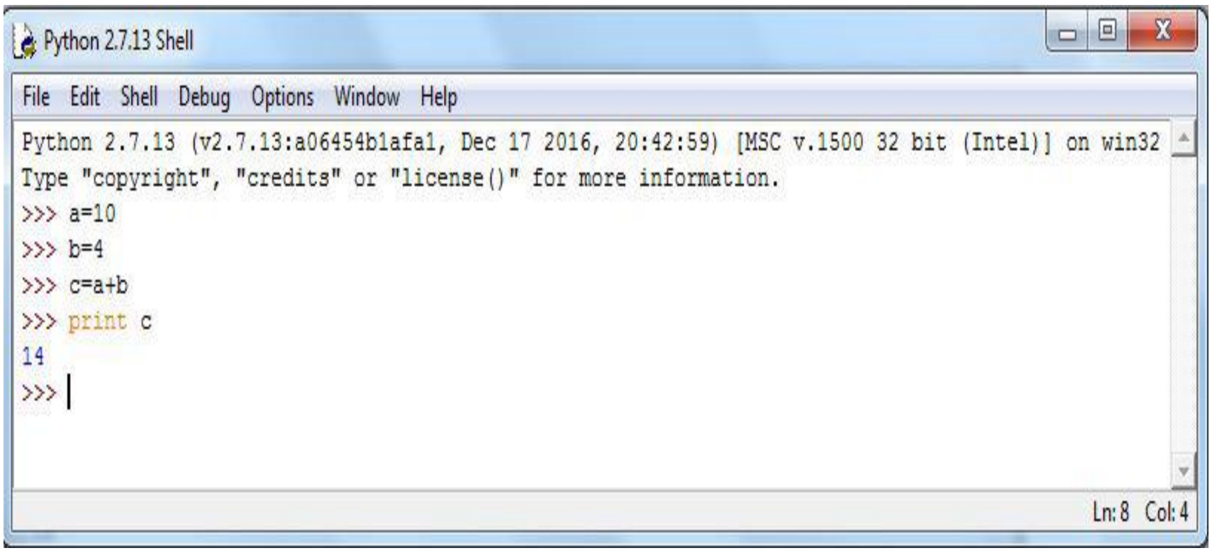
```
[root@localhost Python-2.7.12]#
[root@localhost Python-2.7.12]# python2
Python 2.7.12 (default, Jan 16 2017, 00:50:28)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-44)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
[7]+  Stopped                 python2
[root@localhost Python-2.7.12]# which python2
/usr/local/bin/python2
[root@localhost Python-2.7.12]#
```

4.1.3 Python file formats

Every language understands a file format, for example, like the C language file extension is .c likewise java language has a file extension .java. The Python file extension is .py while bytecode file extension is .pyc.

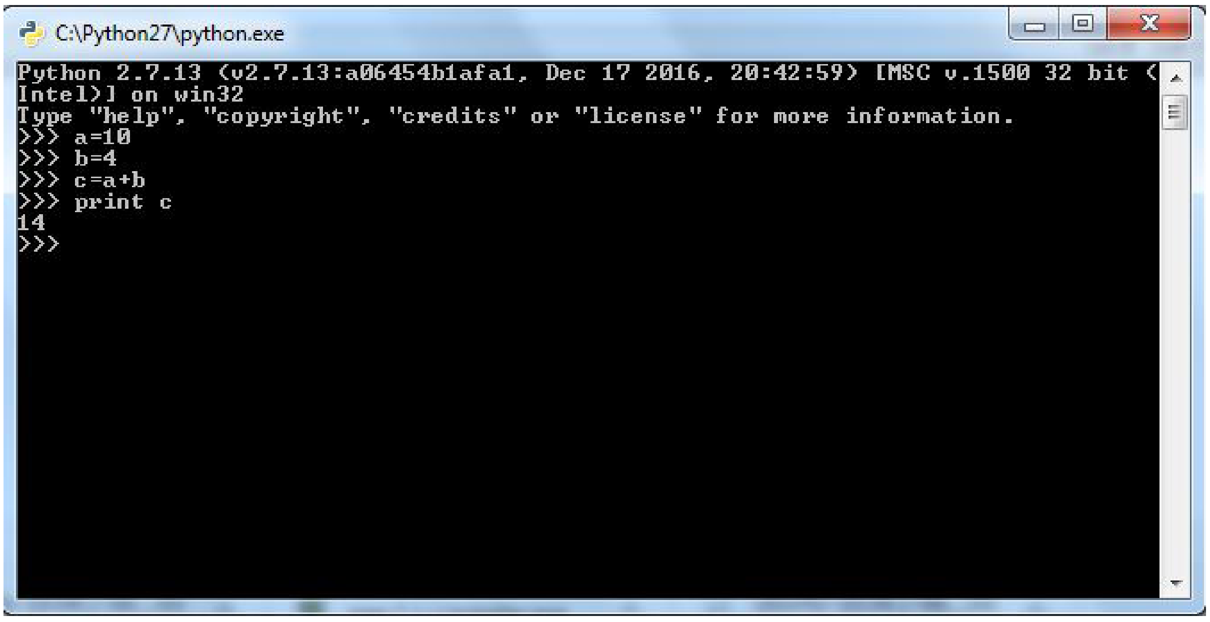
Python interactive shell

Python interactive shell is also known as **Integrated Development Environment** (**IDLE**). With the Python installer, two interactive shells are provided: one is IDLE (Python GUI) and the other is Python (command line). Both can be used for running simple programs. For complex programs and executing large files, the windows command prompt is used, where after the system variables are set automatically, large files are recognized and executed by the system.



```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a=10
>>> b=4
>>> c=a+b
>>> print c
14
>>> 
```

The preceding screenshot is what we call Python IDLE, which comes bundled with the Python installation. The next screenshot is of the command line that also comes bundled with the Python installation, or we can simply launch the Python command through the windows command line and get Python command line. For most of our programming instructions, we will be using the Python command line:



```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (
Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a=10
>>> b=4
>>> c=a+b
>>> print c
14
>>>
```

4.1.4 Syntax and semantics

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation doesn't have any semantic meaning.

Statements and control flow

Python's statements include (among others):

- The **assignment** statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language. Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object. Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However at a given time a name will be bound to some object, which will have a type; thus there is dynamic typing.

- The **if** statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The **for** statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The **while** statement, which executes a block of code as long as its condition is true.
- The **try** statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The **raise** statement, used to raise a specified exception or re-raise a caught exception.
- The **class** statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The **def** statement, which defines a function or method.
- The **with** statement, from Python 2.5 released in September 2006, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing Resource Acquisition Is Initialization (RAII)-like behavior and replaces a common try/finally idiom.
- The **break** statement, exits from the loop.

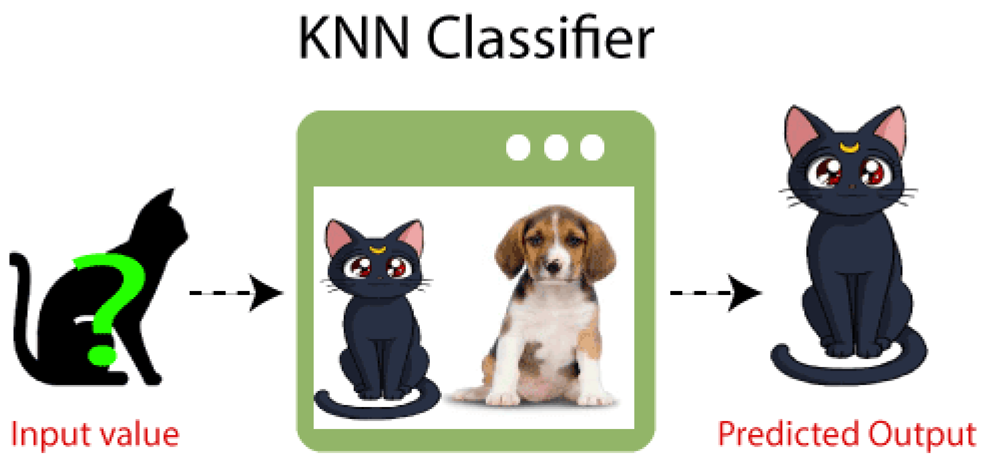
- The **continue** statement, skips this iteration and continues with the next item.
- The **pass** statement, which serves as a NOP. It is syntactically needed to create an empty code block.
- The **assert** statement, used during debugging to check for conditions that ought to apply.
- The **yield** statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.
- The **import** statement, which is used to import modules whose functions or variables can be used in the current program. There are three ways of using import: import <module name> [as <alias>] or from <module name> import * or from <module name> import <definition 1> [as <alias 1>], <definition 2> [as <alias 2>], ...
- The **print** statement was changed to the print() function in Python 3.

MACHINE LEARNING

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. IBM has a rich history with machine learning. One of its own, Arthur Samuel, is credited for coining the term, "machine learning" with his research (PDF, 481 KB) (link resides outside IBM) around the game of checkers. Robert Nealey, the self-proclaimed checkers master, played the game on an IBM 7094 computer in 1962, and he lost to the computer. Compared to what can be done today, this feat almost seems trivial, but it's considered a major milestone within the field of artificial intelligence. Over the next couple of decades, the technological developments around storage and processing power will enable some innovative products that we know and love today, such as Netflix's recommendation engine or self-driving cars. Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

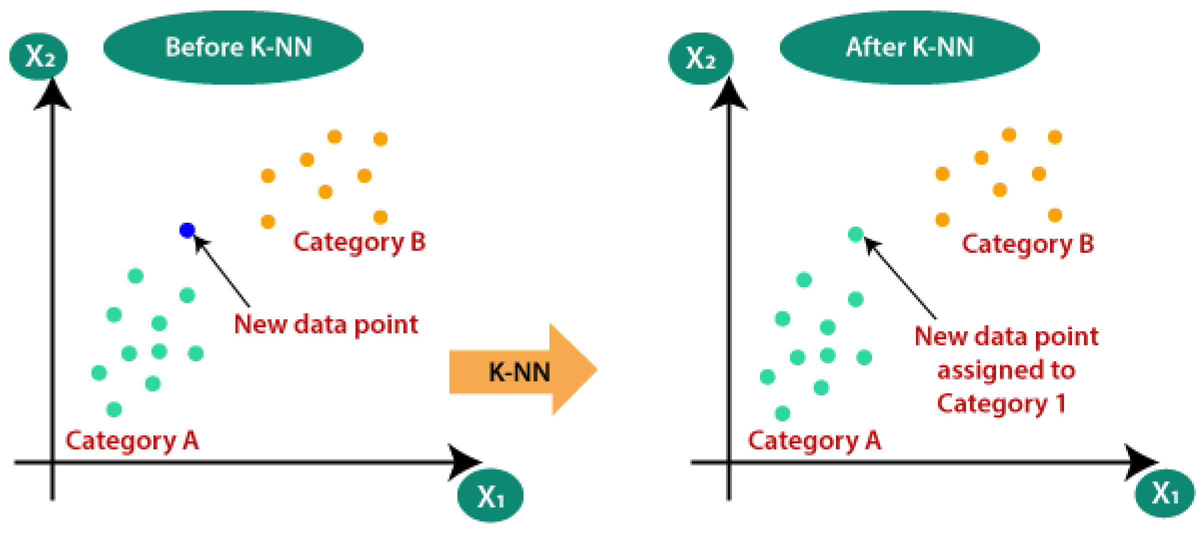
K-NEAREST NEIGHBOUR

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



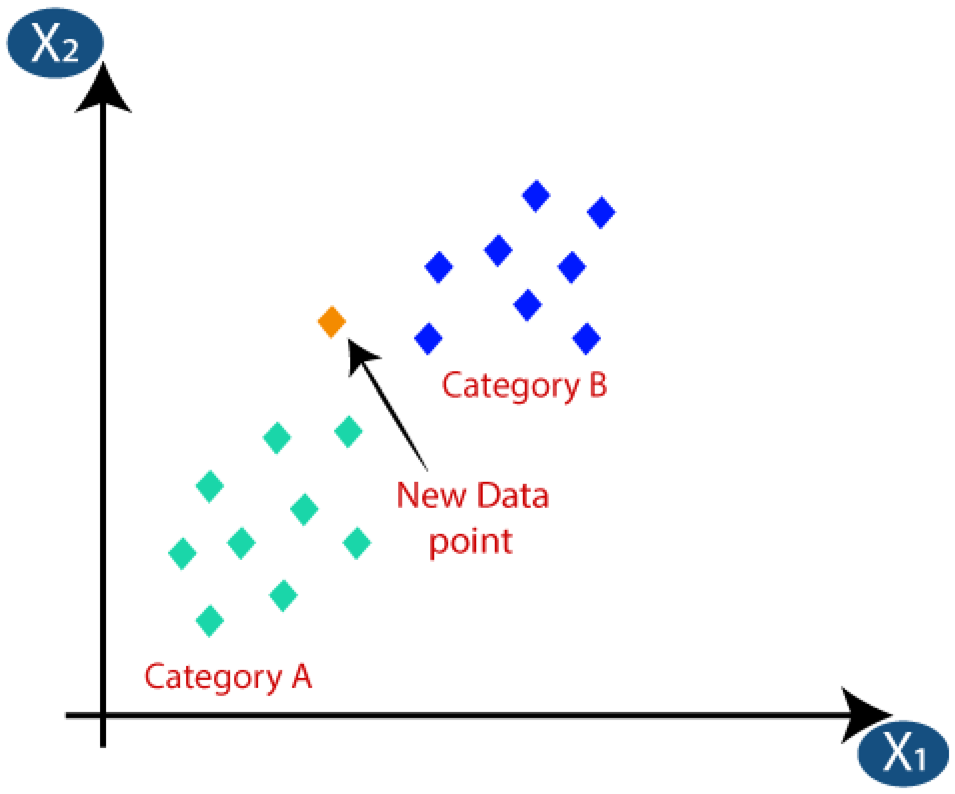
How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

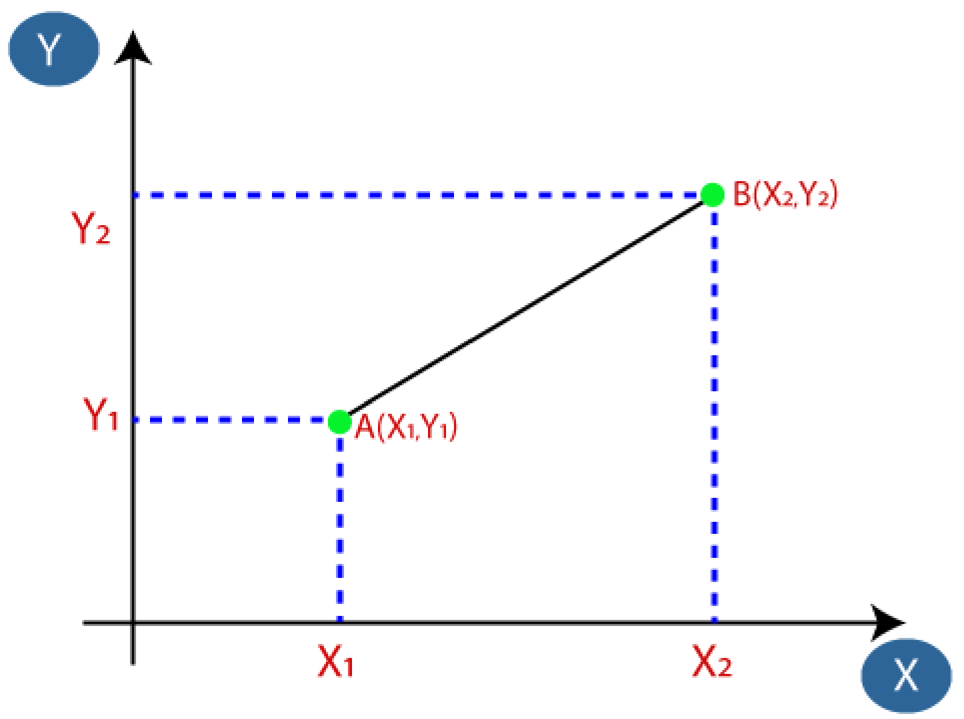
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**

- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

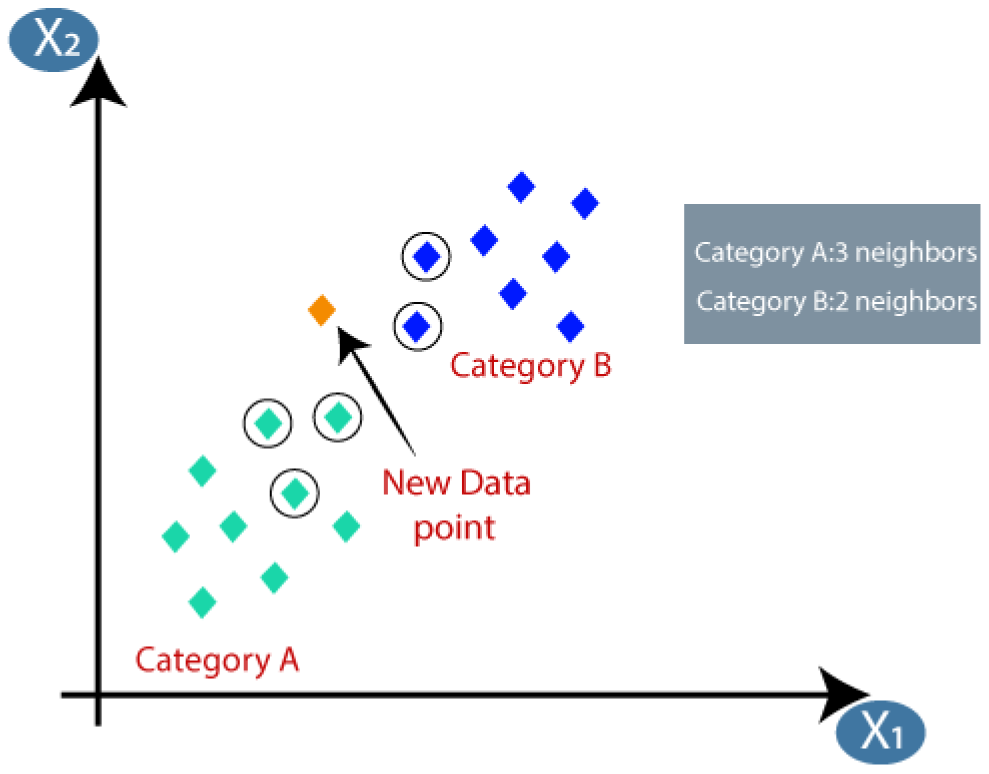


- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

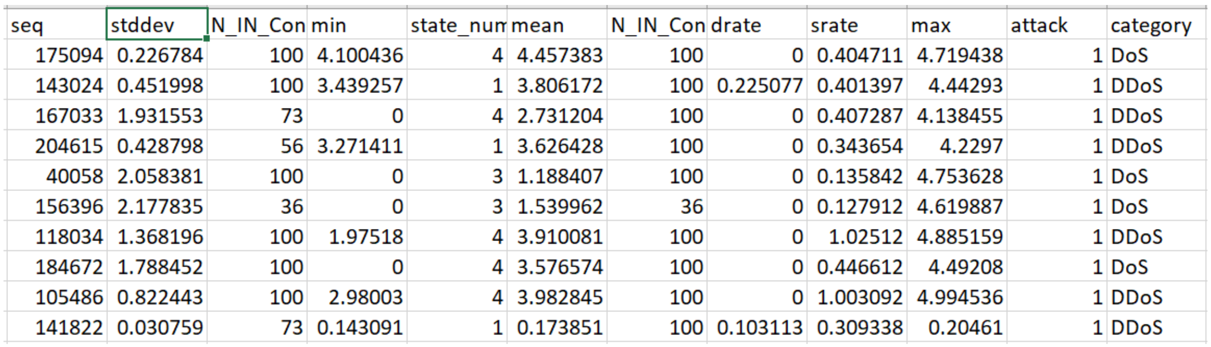
- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

5.SOURCE CODE

DATASETS



seq	stddev	N_IN_Con	min	state_num	mean	N_IN_Con	drate	srate	max	attack	category
175094	0.226784	100	4.100436	4	4.457383	100	0	0.404711	4.719438	1	DoS
143024	0.451998	100	3.439257	1	3.806172	100	0.225077	0.401397	4.44293	1	DDoS
167033	1.931553	73	0	4	2.731204	100	0	0.407287	4.138455	1	DDoS
204615	0.428798	56	3.271411	1	3.626428	100	0	0.343654	4.2297	1	DDoS
40058	2.058381	100	0	3	1.188407	100	0	0.135842	4.753628	1	DoS
156396	2.177835	36	0	3	1.539962	36	0	0.127912	4.619887	1	DoS
118034	1.368196	100	1.97518	4	3.910081	100	0	1.02512	4.885159	1	DDoS
184672	1.788452	100	0	4	3.576574	100	0	0.446612	4.49208	1	DoS
105486	0.822443	100	2.98003	4	3.982845	100	0	1.003092	4.994536	1	DDoS
141822	0.030759	73	0.143091	1	0.173851	100	0.103113	0.309338	0.20461	1	DDoS

TRAINING.PY

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn.metrics as metrics
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import (
    precision_score, recall_score, f1_score,
    cohen_kappa_score, accuracy_score, roc_curve, auc,
    confusion_matrix, ConfusionMatrixDisplay
)
import pickle
import warnings
# Ignore future warnings
warnings.filterwarnings("ignore", category=FutureWarning)
# Load and preprocess the dataset
dataset = pd.read_csv('ddos_dataset.csv')
dataset = dataset.dropna(how="any")
# Display the dataset and its info
print(dataset)
print(dataset.info())
# Plot a histogram of the 'attack' feature
plt.figure(figsize=(10, 8))
plt.title("Histogram of attack")
plt.hist(dataset['attack'], rwidth=0.9)
plt.show()
# Plot a bar graph of 'N_IN_Conn_P_SrcIP' against 'attack'
m = dataset['attack']
n = dataset['N_IN_Conn_P_SrcIP']
plt.figure(figsize=(4, 4))
plt.title("Bar plot graph", fontsize=20)
plt.xlabel("attack", fontsize=15)
plt.ylabel("N_IN_Conn_P_SrcIP", fontsize=15)
plt.bar(m, n, label="bar plot", color="orange", width=0.1)
plt.legend(loc='best')
plt.show()
# Select features and target
X = dataset.iloc[:, 6:16].values
y = dataset.iloc[:, 16].values
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=121)
# Display the training data
print(X_train)
print(" ")
print(y_train)
# Train the KNN model
print("Training Started")
print("Processing")
classifier = KNeighborsClassifier(n_neighbors=2, weights='distance', algorithm='brute', p=1,
leaf_size=1)
classifier.fit(X_train, y_train)
# Save the trained model to disk
with open('knnpickle_file', 'wb') as knnPickle:
    pickle.dump(classifier, knnPickle)
print("Training Completed")
# Predict the test set results
y_pred = classifier.predict(X_test)
```

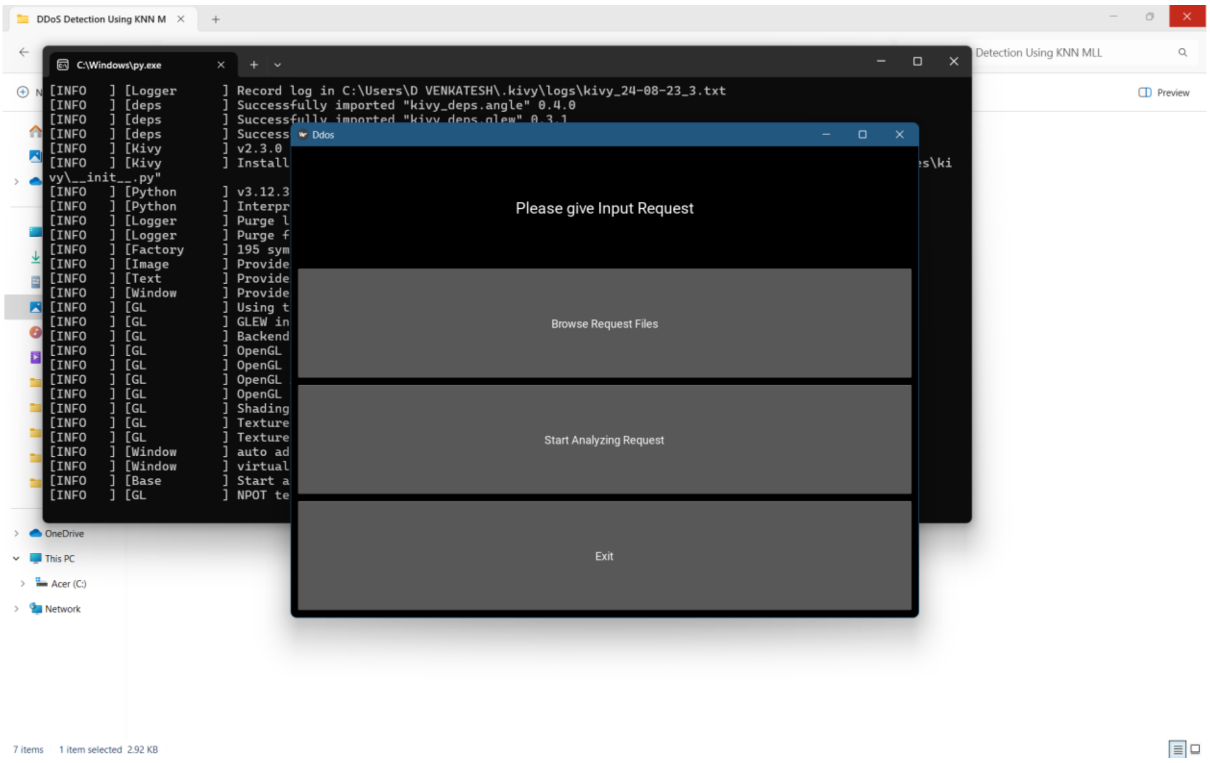
```
# Round the predictions
y_pred = y_pred.round()
print(y_pred)
# Calculate ROC curve and AUC score
fpr1, tpr1, _ = roc_curve(y_test, y_pred)
auc_score1 = auc(fpr1, tpr1)
# Compute and display the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix")
print(cm)
# Plot the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
# Print evaluation metrics
print("KNN ALGORITHM")
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy Score: ', accuracy)
precision = precision_score(y_test, y_pred)
print('Precision: %f' % precision)
recall = recall_score(y_test, y_pred)
print('Recall: %f' % recall)
f1 = f1_score(y_test, y_pred)
print('F1 score: %f' % f1)
kappa = cohen_kappa_score(y_test, y_pred)
print('Cohen\'s kappa: %f' % kappa)
# Plot the ROC curve
plt.figure(figsize=(7, 6))
plt.plot(fpr1, tpr1, color='blue', label='ROC (KNN AUC = %0.4f)' % auc_score1)
plt.legend(loc='lower right')
plt.title("ROC Curve")
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.show()
```

APPLICATION.PY

```
import numpy as np
import pandas as pd
import pickle
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
from kivy.uix.button import Button
from kivy.uix.filechooser import FileChooserIconView
from kivy.uix.popup import Popup
from kivy.uix.widget import Widget
from kivy.uix.screenmanager import ScreenManager, Screen
class MainScreen(Screen):
    def __init__(self, **kwargs):
        super(MainScreen, self).__init__(**kwargs)
        self.layout = BoxLayout(orientation='vertical', padding=10, spacing=10)
        self.label = Label(text="Please give Input Request", font_size='20sp', size_hint=(1, 0.2))
        self.layout.add_widget(self.label)
        self.btn_browse = Button(text="Browse Request Files", size_hint=(1, 0.2), on_press=self.browse_files)
        self.layout.add_widget(self.btn_browse)
        self.btn_start = Button(text="Start Analyzing Request", size_hint=(1, 0.2), on_press=self.start_analysis)
        self.layout.add_widget(self.btn_start)
        self.btn_exit = Button(text="Exit", size_hint=(1, 0.2), on_press=App.get_running_app().stop)
        self.layout.add_widget(self.btn_exit)
        self.add_widget(self.layout)
    def browse_files(self, instance):
        content = FileChooserIconView()
        content.bind(on_submit=self.file_selected)
        self.popup = Popup(title="Select a CSV File",
                           content=content,
                           size_hint=(0.9, 0.9))
        self.popup.open()
    def file_selected(self, filechooser, selection, touch):
        if selection:
            self.filepath = selection[0]
            self.label.text = f"File Opened: {self.filepath}"
            self.popup.dismiss()
    def start_analysis(self, instance):
        if hasattr(self, 'filepath'):
            print("Process Started")
            dataset = pd.read_csv(self.filepath)
            dataset = dataset.dropna(how="any")
            print(dataset.info())
            X = dataset.iloc[:, 6:16].values
            # Load the model from disk
            model = pickle.load(open('knnpickle_file', 'rb'))
            ypred = model.predict(X)
            ypred = ypred.round()
            print(ypred)
            if ypred[0] == 0:
                self.label.text = "There is Everything Normal, Nothing to Worry !!"
                self.label.color = [0, 1, 0, 1]  # Green color
            else:
                self.label.text = "Possibility of DDOS Attack, Better take the Precautions!!"
                self.label.color = [1, 0, 0, 1]  # Red color
        else:
            self.label.text = "Please select a file first."
            self.label.color = [1, 1, 0, 1]  # Yellow color for warning
class DdosApp(App):
    def build(self):
        sm = ScreenManager()
        sm.add_widget(MainScreen(name='main'))
        return sm
if __name__ == '__main__':
    DdosApp().run()
```

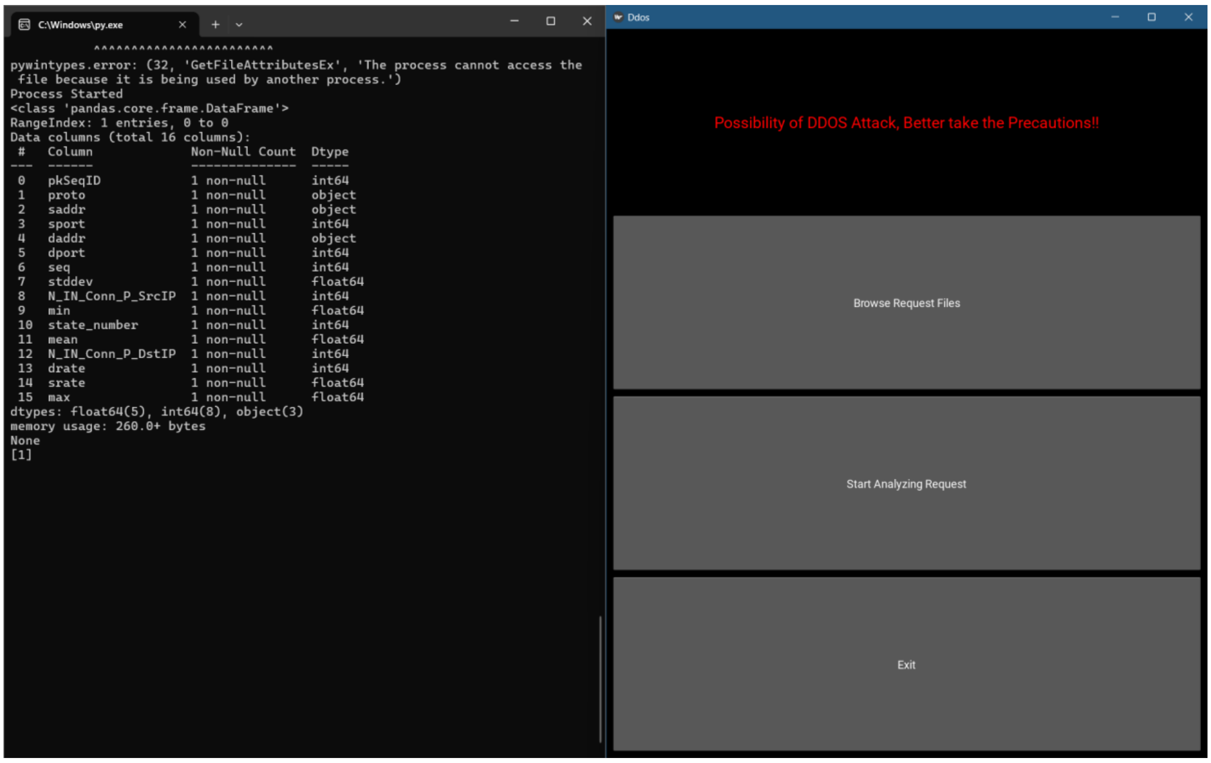
6.SAMPLE OUTPUT

USER INTERFACE



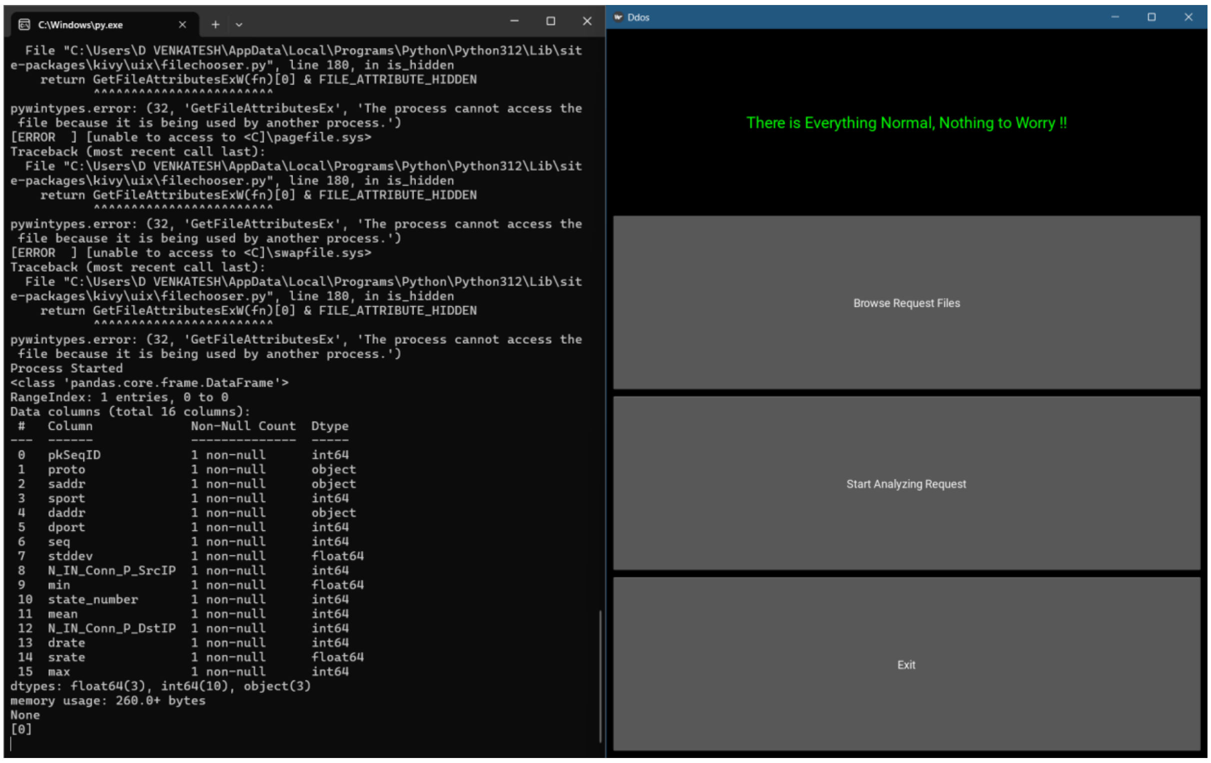
```
[INFO   ] [Logger      ] Record log in C:\Users\D VENKATESH\.kivy\logs\kivy_24-08-23_3.txt
[INFO   ] [deps        ] Successfully imported "kivy_deps.angle" 0.4.0
[INFO   ] [deps        ] Successfully imported "kivy_deps.glew" 0.3.1
[INFO   ] [deps        ] Success
[INFO   ] [Kivy        ] v2.3.0
[INFO   ] [Kivy        ] Install                                                      s\ki
vy\__init__.py"
[INFO   ] [Python      ] v3.12.3
[INFO   ] [Python      ] Interpr
[INFO   ] [Logger      ] Purge l
[INFO   ] [Logger      ] Purge f
[INFO   ] [Factory     ] 195 sym
[INFO   ] [Image       ] Provide
[INFO   ] [Text        ] Provide
[INFO   ] [Window      ] Provide
[INFO   ] [GL          ] Using t
[INFO   ] [GL          ] GLEW in
[INFO   ] [GL          ] Backend
[INFO   ] [GL          ] OpenGL
[INFO   ] [GL          ] OpenGL
[INFO   ] [GL          ] OpenGL
[INFO   ] [GL          ] OpenGL
[INFO   ] [GL          ] Shading
[INFO   ] [GL          ] Texture
[INFO   ] [GL          ] Texture
[INFO   ] [Window      ] auto ad
[INFO   ] [Window      ] virtual
[INFO   ] [Base        ] Start a
[INFO   ] [GL          ] NPOT te
```

DETECTION OF DDOS



```
                  ^^^^^^^^^^^^^^^^^^^^^^^^
pywintypes.error: (32, 'GetFileAttributesEx', 'The process cannot access the
 file because it is being used by another process.')
Process Started
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   pkSeqID            1 non-null      int64
 1   proto              1 non-null      object
 2   saddr              1 non-null      object
 3   sport              1 non-null      int64
 4   daddr              1 non-null      object
 5   dport              1 non-null      int64
 6   seq                1 non-null      int64
 7   stddev             1 non-null      float64
 8   N_IN_Conn_P_SrcIP  1 non-null      int64
 9   min                1 non-null      float64
 10  state_number       1 non-null      int64
 11  mean               1 non-null      float64
 12  N_IN_Conn_P_DstIP  1 non-null      int64
 13  drate              1 non-null      int64
 14  srate              1 non-null      float64
 15  max                1 non-null      float64
dtypes: float64(5), int64(8), object(3)
memory usage: 260.0+ bytes
None
[1]
```

Possibility of DDOS Attack, Better take the Precautions!!



```
  File "C:\Users\D VENKATESH\AppData\Local\Programs\Python\Python312\Lib\sit
e-packages\kivy\uix\filechooser.py", line 180, in is_hidden
    return GetFileAttributesExW(fn)[0] & FILE_ATTRIBUTE_HIDDEN
                  ^^^^^^^^^^^^^^^^^^^^^^^^
pywintypes.error: (32, 'GetFileAttributesEx', 'The process cannot access the
 file because it is being used by another process.')
[ERROR  ] [unable to access to <C]\pagefile.sys>
Traceback (most recent call last):
  File "C:\Users\D VENKATESH\AppData\Local\Programs\Python\Python312\Lib\sit
e-packages\kivy\uix\filechooser.py", line 180, in is_hidden
    return GetFileAttributesExW(fn)[0] & FILE_ATTRIBUTE_HIDDEN
                  ^^^^^^^^^^^^^^^^^^^^^^^^
pywintypes.error: (32, 'GetFileAttributesEx', 'The process cannot access the
 file because it is being used by another process.')
[ERROR  ] [unable to access to <C]\swapfile.sys>
Traceback (most recent call last):
  File "C:\Users\D VENKATESH\AppData\Local\Programs\Python\Python312\Lib\sit
e-packages\kivy\uix\filechooser.py", line 180, in is_hidden
    return GetFileAttributesExW(fn)[0] & FILE_ATTRIBUTE_HIDDEN
                  ^^^^^^^^^^^^^^^^^^^^^^^^
pywintypes.error: (32, 'GetFileAttributesEx', 'The process cannot access the
 file because it is being used by another process.')
Process Started
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   pkSeqID            1 non-null      int64
 1   proto              1 non-null      object
 2   saddr              1 non-null      object
 3   sport              1 non-null      int64
 4   daddr              1 non-null      object
 5   dport              1 non-null      int64
 6   seq                1 non-null      int64
 7   stddev             1 non-null      float64
 8   N_IN_Conn_P_SrcIP  1 non-null      int64
 9   min                1 non-null      float64
 10  state_number       1 non-null      int64
 11  mean               1 non-null      int64
 12  N_IN_Conn_P_DstIP  1 non-null      int64
 13  drate              1 non-null      int64
 14  srate              1 non-null      float64
 15  max                1 non-null      int64
dtypes: float64(3), int64(10), object(3)
memory usage: 260.0+ bytes
None
[0]
```

There is Everything Normal, Nothing to Worry !!

CONCLUSION

In our project we used K Nearest Neighbour algorithm for predicting DDoS Attack. It is a one of the Machine learning technique which can able to train and predict DDoS based upon input data. We used DDoS attack dataset which holds over 1000 data's for training purpose. After training we predict DDoS attack by using test data's using KNN. Our Model Archives more than 98% accuracy during testing and training. The predicted results by KNN is accurate and stable, its patterns are also matched with the existing dataset patterns. And hence our model is perfectly trained and it can able to predict the DDoS attack with high stability. Our future work is to improve the accuracy by using Deep learning techniques.