PACKET FILTERING USING MACHINE LEARNING

Project Report

Submitted in partial fulfilment for the award of degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Submitted By

K. V. S. P. AKHILESH (18L31A1201)

T. SAI LAKSHMI (18L31A1231)

T. SRAVANTHI (18L31A1234)

G. O. S. MANIKANTA (18L31A1248)

S. SANDEEP RAHUL (18L31A1273)

Under the esteemed guidance of

Mr. Ch. Srinivasa Reddy

Assistant Professor

VIGNAN's INSTITUTE OF INFORMATION TECHNOLOGY
(AUTONOMOUS)
(Approved by AICTE-New Delhi & Affiliated to JNTUK, Kakinada)
Beside VSEZ, Duvvada, Vadlapudi Post, Gajuwaka, Visakhapatnam - 530 049.

May 2022

VIGNAN's INSTITUTE OF INFORMATION TECHNOLOGY
(AUTONOMOUS)
(Approved by AICTE-New Delhi & Affiliated to JNTUK, Kakinada)
Beside VSEZ, Duvvada, Vadlapudi Post, Gajuwaka, Visakhapatnam - 530 049.

Department of Information Technology

CERTIFICATE

This is to certify that the Project report entitled **"PACKET FILTERING USING MACHINE LEARNING"** is a bonafide record of project work carried out under my supervision by **K.V.S.P. AKHILESH** bearing Regd.No.**18L31A1201**, **T. SAI LAKSHMI** bearing Regd.No.**18l31A1231**, **T. SRAVANTHI** bearing the Regd.No.**18L31A1234**, **G.O.S. MANIKANTA** bearing the Regd.No.**18L31A1248**, **S. SANDEEP RAHUL** bearing the Regd.No.**18L31A1273** in partial fulfilment of the degree of **Bachelor of Technology in Information Technology of** Vignan's Institute of Information Technology(A) affiliated to Jawaharlal Nehru Technology University Kakinada, during the academic year 2018-2022.

Signature	Signature
Mr. Ch. Srinivasa Reddy	**Dr. G. Rajendra Kumar (HOD)**

DECLARATION

We here by declare that this project report entitled **"PACKET FILTERING USING MACHINE LEARNING"** has undertaken by us for the fulfilment of Bachelor of Technology in Information Technology. We declare that this project report has not been submitted anywhere in the part of fulfilment for any degree of any other University.

PLACE: Visakhapatnam

DATE:

K. V. S. P. Akhilesh	(18L31A1201)
T. Sai Lakshmi	(18L31A1231)
T. Sravanthi	(18L31A1234)
G.O.S. Manikanta	(18L31A1248)
S. Sandeep Rahul	(18L31A1273)

ACKNOWLEDGEMENT

An endeavor over a long period can be successfully with the advice and support of many well-wishers. I take this opportunity to express our gratitude and appreciation to all of them.

I express my sincere gratitude to my internal guide, **Mr. Ch. Srinivasa Reddy** for his encouragement and cooperation in completion of my project. I am very fortunate in getting the generous help and constant encouragement from him/her.

I would be very grateful to our project coordinator, **Mrs. A. Sirisha** for the continuous monitoring of my project work. I truly appreciate for her time and effort spent helping me

I would like to thank our Head of the Department, **Dr. G. Rajendra Kumar** and all other teaching and non-teaching staff of the department for their cooperation and guidance during my project.

I sincerely thank to **Dr. B. Arundhati,** Principal of VIGNAN’S INSTITUTE OF INFORMATION TECHNOLOGY (A) for her inspiration to undergo this project.

I wanted to convey my sincere gratitude to **Dr. V. Madhusudan Rao**, Rector of VIGNAN’S INSTITUTE OF INFORMATION TECHNOLOGY (A) for allocating the required resources and for the knowledge sharing during my project work.

I extended my grateful thanks to our honorable Chairman **Dr. L. Rathaiah** for giving me an opportunity to study in his esteemed institution.

K. V. S. P. Akhilesh	(18L31A1201)
T. Sai Lakshmi	(18L31A1231)
T. Sravanthi	(18L31A1234)
G.O.S. Manikanta	(18L31A1248)
S. Sandeep Rahul	(18L31A1273)

VIGNAN's INSTITUTE OF INFORMATION TECHNOLOGY
(AUTONOMOUS)
(Approved by AICTE-New Delhi & Affiliated to JNTUK, Kakinada)
Beside VSEZ, Duvvada, Vadlapudi Post, Gajuwaka, Visakhapatnam - 530 049.

INFORMATION TECHNOLOGY

VISION:

To be a centre of excellence in high quality education and research producing globally competent IT professionals with ethical /human values to meet the needs of Information Technology sector and related services.

MISSION:

- To impart high quality of education through innovative teaching –learning practices resulting in strong software and hardware knowledge and skills to enable students to meet the challenges of IT profession

- To facilitate faculty and students to carry out research work by providing necessary latest facilities and a conductive environment

- To mould the students into effective professionals with necessary communication skills, team spirit, leadership qualities, managerial skills, integrity, social and environmental responsibility and lifelong learning ability with professional ethics and human values.

Vision and Mission of the Institute:

Vision of the Institute (VIIT):

We envision to be recognized leader in technical education and shall aim at national excellence by creating competent and socially conscious technical manpower for the current and future industrial requirements and development of the nation.

Mission of the Institute (VIIT):

We intend to fulfil our stated vision by

- Introducing innovative practices of Teaching-Learning
- Undertaking research and development in thrust areas
- Continuously collaborating with industry
- Promoting strong set of ethical values
- Serving the surrounding region and nation at large

Program Educational Objectives (PEOs):

PEO1: To work in core IT companies/allied industries, educational institutions, research organizations and/or be entrepreneurs.

PEO2: To pursue higher education/ research in the field of Information Technology.

PEO3: To demonstrate communication skills, team spirit, leadership qualities, managerial skills, integrity, social & environmental responsibility and lifelong learning ability, professional ethics and human values in profession/career.

PSOs: Program Specific outcomes:

PSO1: Analyse and design the solutions for data storage & computing systems.

PSO2: Implement the solutions for network and communication problems of Information Technology.

PROGRAM OUTCOMES	
PO1	**Engineering Knowledge:** Apply the knowledge of mathematics science engineering fundamentals and mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems engineering problems.
PO2	**Problem Analysis:** Identify, formulate, review research Literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, and natural sciences, and engineering sciences.
PO3	**Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate

	consideration for the public health and safety, and the cultural societal, and environmental considerations.
PO4	**Conduct Investigations of Complex Problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
PO6	**The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and Team Work:** Function effectively as an individual and as a member or leader in diverse teams and individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12	**Life-long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The Intrusion detection System (IDS) is the first line of defense for any network against any cyberattacks. The rapid increase in internet-connected devices due to the implementation of IoT and Industry 4.0 poses a significant challenge for the IDS to process a massive amount of network traffic to segregate the cyberattacks from benign traffic. One of the severe issues in the present generation is the malware or malicious data. The increase in internet-connected devices due to the implementation of IoT and Industry 4.0 causes a significant challenge for the Intrusion Detection System to process a massive amount of network traffic to segregate the cyberattacks from nonthreatening traffic. This project has proposed a machine learning based Packet Filtering System that can detect Intrusion by analyzing network traffic behavior and stop the attacks. To prepare a machine learning model that can check the malicious data, the information is gathered from the validated internet source. This consists of collection of data sets from KDD-cup data to train a model for filtering of packets and to check the which model gives the highest accuracy to use for filtering of packets.

INDEX

List of Figures

1.INTRODUCTION

1.1 Introduction to information technology:

Computer is a discipline that spans theory and practice. It requires thinking both in abstract terms and concrete terms. The practical side of computing can be seen everywhere. Nowadays, practically everyone is a computer user, and many people are even computer programmers. Computer scientists must be adept at modelling and analysing problems.

Problem-solving requires precision, creativity, and careful reasoning. Computer science also has strong connections to other disciplines, many problems in science, engineering, healthcare, business, and other areas can be solved effectively with computers, but finding a solution requires both computer science expertise and knowledge of the particular application domain.

Finally, Computer science has a wide range of specialties. These include computer architecture software systems, graphics, AI, computational science, and software engineering. Engineering provides the techniques for building hardware and software.

1.2 Introduction to Machine Learning:

The term 'machine learning' is often, incorrectly, interchanged with Artificial Intelligence [JB1], but machine learning is a sub-field/type of AI. Machine learning is also often referred to as predictive analytics, or predictive modelling.

Coined by American computer by scientist Arthur Samuel in 1959. At its most basic, machine learning uses programmed algorithms that receive and analyse input data to predict output values within an acceptable range.

As new data is fed to these algorithms, they learn and optimize their operations to improve performance, developing 'intelligence' over time. There are four types of machine learning algorithms: **Supervised, Semi-supervised, Unsupervised, and Reinforcement.**

1.2.1 Supervised learning:

In supervised learning, the machine is taught by example. The operator provides the machine learning algorithm with a known dataset that includes desired inputs and outputs, and the algorithm must find a method to determine how to arrive at those inputs and outputs. While the operator knows the correct answers to the problem, the algorithm identifies patterns in data, learns from observations and makes predictions. The algorithm makes predictions and is corrected by the operator – and this process continues until the algorithm achieves a high level of accuracy/performance. Under the umbrella of supervised learning: Classification, Regression and Forecasting.

Classification:

In classification tasks, the machine learning program must conclude observed values and determine to what category new observations belong. For example, when filtering emails as 'spam' or 'not spam', the program must look at existing observational data and filter the emails accordingly.

Regression:

In regression tasks, the machine learning program must estimate – and understand – the relationships among variables. Regression analysis focuses on one dependent variable and a series of other changing variables –making by Predicting and forecasting.

Forecasting:

Forecasting is the process of making predictions about the future based on the past and present data and is commonly used to analyse trends.

1.2.2 Semi-Supervised learning:

Semi-supervised learning is like supervised learning but instead uses both labelled and unlabelled data. Labelled data is essentially information that has meaningful tags so that the algorithm can understand the data, whilst unlabelled data lacks that information. By using this combination, machine learning algorithms can learn to label unlabelled data.

1.2.3 Unsupervised learning:

Here, the machine learning algorithm studies data to identify patterns. There is no answer key or human operator to provide instruction. Instead, the machine determines the correlations and relationships by analysing available data. In an unsupervised learning process, the machine learning algorithm is left to interpret large data sets and address that data accordingly.

Clustering:

Clustering involves grouping sets of similar data (based on defined criteria). It’s useful for segmenting data into several groups and performing analysis on each data set to find the problems.

Dimension reduction:

Dimension reduction reduces the number of variables being considered to find the exact information required.

1.2.4 Reinforcement learning:

Reinforcement learning focuses on regimented learning processes, where a machine learning algorithm is provided with a set of actions, parameters and end values. By defining the rules, the machine learning algorithm then tries to explore different options and possibilities, monitoring and evaluating each result to determine which one is optimal. Reinforcement learning teaches the machine trial and error. It learns from past experiences and begins to adapt its approach in response to the situation to achieve the best possible result.

1.2.5 Classification and Regression :

The above scenario is an example of a classification machine learning problem. A classification algorithm will give a probability score of a person having the disease, or, more broadly, the probability of event happening vs. not. Healthcare. AI has already implemented some of the simplest algorithms to answer questions like:

1. What is the likelihood that a patient will develop a central line infection?
2. What is the likelihood that a COPD patient will be readmitted within 90 days of discharge?
3. What is the likelihood that a person will no-show for their appointment?

These questions are posed with a lot of example data and the expectation that a model will give a probability from 0 (low) to 1 (high). It's up to us to draw the line of what we call a positive prediction and what we call a negative prediction. On the other hand, a regression algorithm will predict a continuous value. Here are some examples of questions that we have or plan to tackle using regression algorithms:

1. How many days will a patient need to stay in the hospital?
2. How many people do we need on staff in the ED on a given night?
3. How much money will a patient cost the health system over the next year?

There are a lot of exciting questions that can be answered with very basic machine learning! As we've said before, we are focused on trying to answer the questions that will make the most impact right now. Luckily for us, there is still a lot of low-hanging fruit in healthcare for our team to address. If there is good example data, ML could help answer a huge range of questions.

1.2.6 Supervised vs. Unsupervised:

All the problems that this post has discussed so far are supervised machine learning problems. For each of these there is a ground truth associated with every patient example being used to train the model. The other major type of machine learning is unsupervised. There is no ground truth associated with the data. The algorithms in this category are largely related to identifying patterns and similarity and using them to group or stratify data into different categories. This type of functionality is a high priority capability that we are working on implementing in healthcare.ai. In the very near future, we hope to be able to use clustering methods and anomaly detection to answer questions like:

Does this patient (who hasn't been associated with diabetes) belong in a diabetes registry? Or a heart disease registry? And with what likelihood?

How similar are my high-utilizing patients? Do they fall into clusters? What can we learn about the characteristics of these separate clusters?

These questions are typically more nebulous than supervised learning problems, but useful insights can still be gathered. For example, there would be value in labelling a non-diabetic patient as a person to watch and intervene before they ever develop diabetes. This is great information to have, and ML will absolutely make an impact by answering such questions.

1.3 About project:

Due to the explosion in the internet connectivity and the mainstream use of broadband and mobile technologies nowadays there is a massive increase in the computer systems and storage devices to a network. Due to this our assets, confidential data and intellectual property are more suspectable to misuse of information or cyber-attacks more than ever before.

To this changing threat landscape, packet filtering using machine learning model was developed to provide protection beyond that firewalls and intrusion detection system. Packet filtering makes analysis of packets that were used to send from one to another and drops that packet if any malicious data or corrupted data found in that packets, if it’s found then it drops that packet and blocks that IP address.

This packet filtering was trained by using the data set which was taken from the KDD-cup data set, which consists of **4898431 rows × 43 columns** initially. After performing data cleaning, we have taken into consideration of 4898431 rows × 33columns.

This data set of **4898431 rows × 33 columns** divided into two data sets one is for training data set and another for testing purpose. That is for training data set we have taken **3281948 rows x 31 columns** and for testing purpose we taken **1616483 rows x 31 columns** with a split factor of **0.33**. with this training data set we got Random Forest is the best model to implement the packet filtering.

1.4 Algorithms Used

1.4.1 Naive Bayes Classifier (Supervised Learning - Classification) :

The Naive Bayes classifier is based on Bayes’ theorem and classifies every value as independent of any other value. It allows us to predict a class/category, based on a given set of features, using probability. Despite its simplicity, the classifier does surprisingly well and is often used due to the fact it outperforms more sophisticated classification methods.

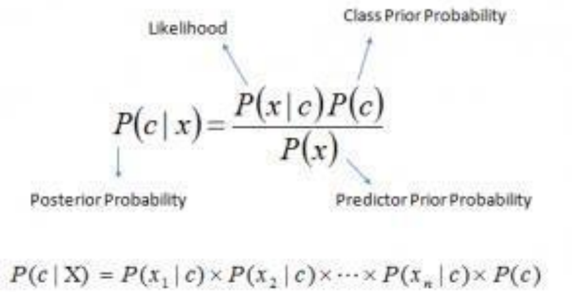

Likelihood
Class Prior Probability
$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$
Posterior Probability
Predictor Prior Probability
$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$

Fig 1.4.1: Hyper plane Classifying two classes.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), Above,

1.4.1.1 *P*(*c*|*x*) is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).

1.4.1.2 *P*(*c*) is the prior probability of *class*.

1.4.1.3 *P*(*x*|*c*) is the likelihood which is the probability of *predictor* given *class*.

1.4.1.4 *P*(*x*) is the prior probability of *predictor*.

1.4.2 Logistic Regression (Supervised Learning – Classification)

Logistic regression focuses on estimating the probability of an event occurring based on the previous data provided. It is used to cover a binary dependent variable, that is where only two values, 0 and 1, represent outcomes. Logistic regression models the data using the sigmoid function.

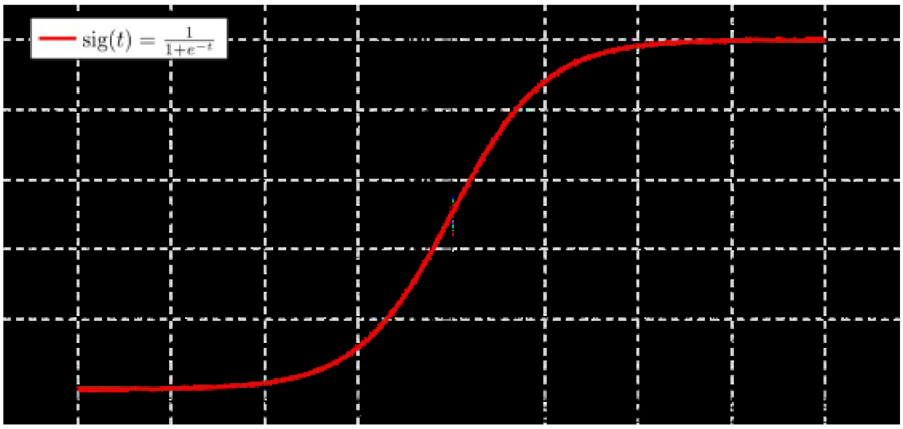


Fig 1.4.2: Sigmoid function Graph

1.4.3 Decision Trees (Supervised Learning–classification/Regression)

A decision tree is a flow-chart-like tree structure that uses a branching method to illustrate every possible outcome of a decision. Each node within the tree represents a test on a specific variable – and each branch is the outcome of that test.

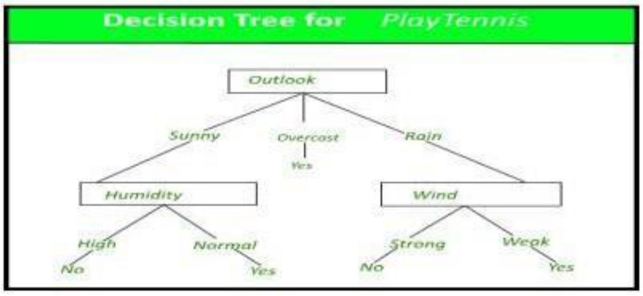


Fig 1.4.3: Representation of decision Tree

1.4.4 Random Forest (Supervised Learning – Classification/Regression)

Random forests or 'random decision forests' is an ensemble learning method, combining multiple algorithms to generate better results for classification, regression and other tasks. Each classifier is weak, but when combined with others, can produce excellent results. The algorithm starts with a 'decision tree' (a tree-like graph or model of decisions) and an input is entered at the top. It then travels down the tree, with data being segmented into smaller and smaller sets, based on specific variables.

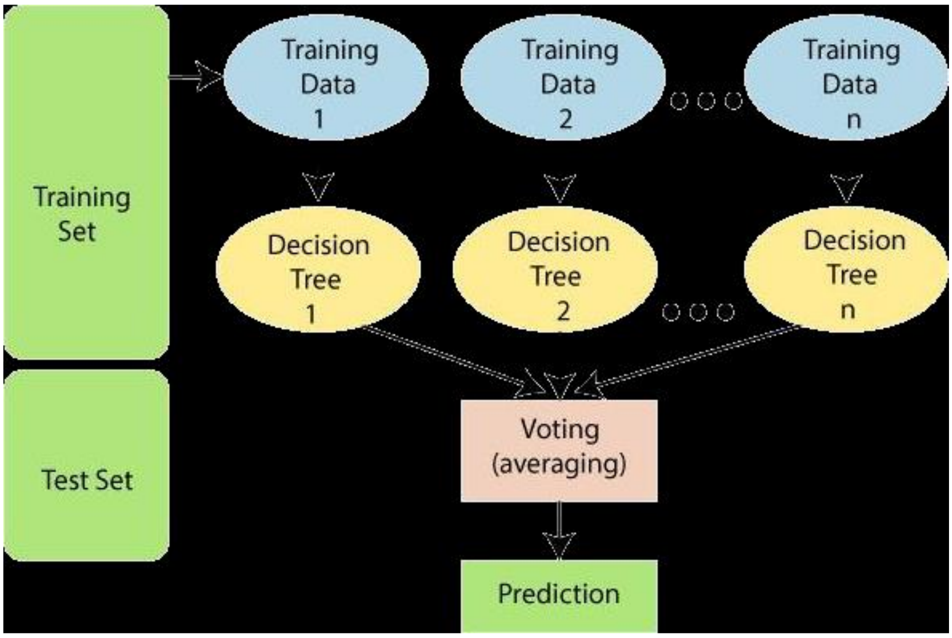


Fig 1.4.4: Prediction process in Random Forest

1.5 Software Development Cycle

SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time possible. SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific
software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC:

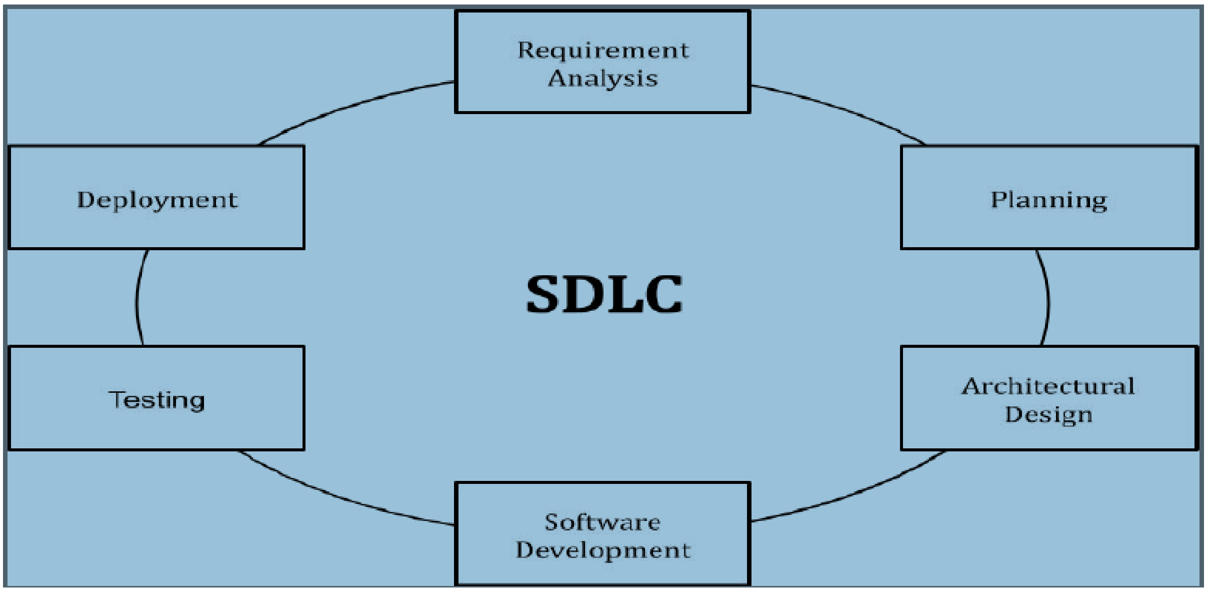


Fig 1.5: Representation of SDLC

Stage 1: Requirement Gathering and Analysis:

During this all phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only.

Business analyst and Project Manager set up a meeting with the customer to gather all the Information like what the customer wants to build, who will be the end-user, what is the purpose of the product. Before building a product a core understanding or knowledge of the product is very important.

Once the requirement gathering is done, an analysis is done to check the feasibility of the development of a product. In case of any ambiguity, a call is set up for further discussion. Once the requirement is clearly understood, the SRS (Software Requirement Specification) document is created.

This document should be thoroughly understood by the developers and also should be reviewed by the customer for future reference.

Stage 2: Design:

In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

Stage 3: Implementation or coding:

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase.

Stage 4: Testing :

Testing starts once the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed. Retesting, regression testing is done until the point at which the software is as per the customer's expectation. Testers refer SRS document to make sure that the software is as per the customer's standard**.**

Stage 5: Deployment :

Once the product is tested, it is deployed in the production environment or first UAT (User Acceptance testing) is done depending on the customer expectation. In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

Stage 6: Maintenance:

After the deployment of a product on the production environment, maintenance of the product i.e., if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

1.6 Workflow of a machine learning project:

Workflow can mean different things to different people, but in the case of ML it is the series of various steps through which a ML project goes on. It means that passing each and every stage Detection of degenerative disorder using Machine learning (stacking)under the workflow to complete the project successfully and in time.

We will follow the general Machine Learning workflow steps:

1. Gathering data
2. Data per-processing
3. Splitting the data for training and testing the model
4. Researching the model that will be best for the type of data
5. Training and testing the model
6. Evaluation

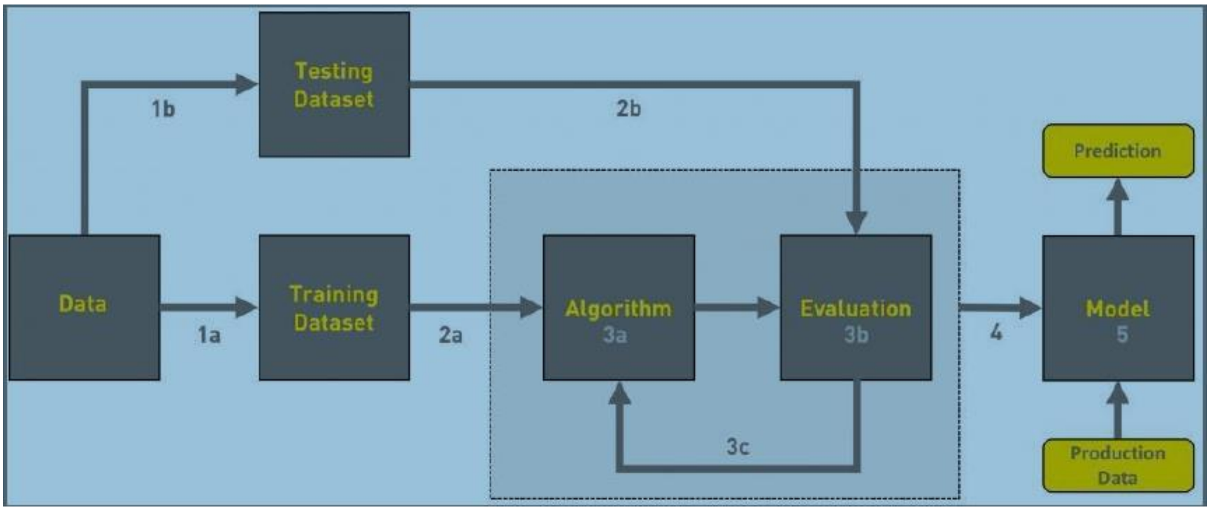


Figure 1.6: **Workflow of machine learning Project.**

What is the machine learning Model?

The machine learning model is nothing but a piece of code; an engineer or data scientist makes it smart through training with data. So, if you give garbage to the model, you will get garbage
in return, i.e., the trained model will provide false or wrong predictions

1. Gathering Data

The process of gathering data depends on the type of project we desire to make, if we want to
make an ML project that uses real-time data, then we can build an IoT system that using different sensors data. The data set can be collected from various sources such as a file, database, sensor and many other such sources but the collected data cannot be used directly for performing the analysis process as there might be a lot of missing data, extremely large values, unorganized text data or noisy data. Therefore, to solve this problem Data Preparation is done.

2. Data pre-processing

Data pre-processing is a process of cleaning the raw data i.e., the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn't feasible for the analysis. Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing.

Why do we need it?

As we know that data pre-processing is a process of cleaning the raw data into clean data, so that can be used to train the model. So, we need data pre-processing to achieve good results from the applied model in machine learning and deep learning projects. Most of the real-world data is messy, some of these types of data are:

Missing Data: Missing data can be found when it is not continuously created or due to technical issues in the application (IOT system).

Noisy Data: This type of data is also called outliners; this can occur due to human errors(human manually gathering the data) or some technical problem of the device at the time of collection of data.

Inconsistent Data: This type of data might be collected due to human errors (mistakes with the name or values) or duplication of data.

3. Splitting the data for training and testing the model:

For training a model we initially split the model into 3 three sections which are '**Training data**' and '**Testing data**'.

Training Data

The training set is the material through which the computer learns how to process information.

Machine learning uses algorithms to perform the training part. A set of data used for learning, that is to fit the parameters of the classifier.

Test Data

A set of unseen data used only to assess the performance of a fully specified classifier.

4. Researching the model that will be best for the type of data:

Moving on forward to the next step, we have to choose the best suited model. Models are compared based on the accuracy score that they generate. One way to choose the best model is to train each model and take the results of that model that is showing the best results out of them (obviously, a time taking process, but quite interesting if we get familiar). This step also includes training the data set and fitting our data in the model and then testing it to predict and get the accuracy score.

5. Training and testing the model on data:

You train the classifier using '**training data set**', tune the parameters using '**validation set**' and then test the performance of your classifier on unseen '**test data set**'. An important point to note is that during training the classifier only the training and/or validation set is available.

The test data set must not be used during training the classifier. The test set will only be available during testing the classifier. In a data set, a training set is implemented to build up a model, while a test (or validation) set is to validate the model built. Data points in the training set are excluded from the test (validation) set. Usually, a data set is divided into a training set, a validation set (some people use 'test set' instead) in each iteration, or divided into a training set, a validation set and a test set in each iteration.

6. Evaluation

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future.

1.7 Software Description:

1.7.1 About Python

Python Programming Language is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. The syntax in Python helps the programmers to do coding in fewer steps as compared to Java or C++. The language founded Detection of degenerative disorder using Machine learning (stacking) in the year 1991 by the developer Guido Van Rossum has the programming easy and fun to do. The Python is widely used in bigger organizations because of its multiple programming paradigms. They usually involve imperative and object-oriented functional programming. It has a comprehensive and large standard library that has automatic memory management and dynamic features.

1.7.2 Why Python?

Python is a general purpose and high-level programming language. You can use Python for developing desktop GUI applications, websites and web applications. Also, Python, as a high-level programming language, allows you to focus on core functionality of the application by taking care of common programming tasks. The simple syntax rules of the programming language further make it easier for you to keep the code base readable and application maintainable. There are also a number of reasons why you should prefer Python to other programming languages.

Reasons Why You Must Consider Writing Software Applications in Python:

1) Readable and Maintainable Code
2) Multiple Programming Paradigms
3) Compatible with Major Platforms and Systems
4) Robust Standard Library
5) Many Open-Source Frameworks and Tools
6) Simplify Complex Software Development
7) Adopt Test Driven Development

1.7.3 Advantages and Limitations of Python Advantages of Python

The Python language has diversified application in the software development companies such as in gaming, web frameworks and applications, language development, prototyping, graphic design applications, etc. This provides the language a higher plethora over other programming languages used in the industry. Some of its advantages are-

1. Extensive Support Libraries

It provides large standard libraries that include areas like string operations, Internet, web service Detection of degenerative disorder using Machine learning (stacking) tools, operating system interfaces, and protocols. Most of the highly used programming tasks are already scripted into it that limits the length of the codes to be written in Python.

2. Integration Feature

Python integrates the Enterprise Application Integration that makes it easy to develop Web services by invoking COM or COBRA components. It has powerful control capabilities as it calls directly through C, C++, or Java via Jython. Python also processes XML and other mark- up languages as it can run on all modern operating systems through same byte code.

3. Improved Programmer's Productivity

The language has extensive support libraries and clean object-oriented designs that increase two to tenfold of programmer's productivity while using languages like Java, VB, Perl, C, C++, and C#.

4. Productivity

With its strong process integration features, unit testing framework and enhanced control capabilities contribute towards the increased speed for most applications and productivity of applications. It is a great option for building scalable multi-protocol network applications.

Limitations of Python

Python has varied advantageous features, and programmers prefer this language to other programming languages because it is easy to learn and code too.
However, this language has still not made its place in some computing arenas that includes Enterprise Development Shops. Therefore, this language may not solve some of the enterprise solutions, and limitations include-

1. Difficulty in Using Other Languages

The Python lovers become so accustomed to its features and its extensive libraries, so they face problem in learning or working on other programming languages. Python experts may see the declaring of cast "values" or variable "types", syntactic requirements of adding curly braces or semicolons as an onerous task. Detection of degenerative disorder using Machine learning (stacking)

1. Live code

2. Interactive widgets

3. Plots

4. Narrative text

5. Videos

6. Images

2. Weak in Mobile Computing

Python has made its presence on many desktop and server platforms, but it is seen as a weak language for mobile computing. This is the reason very few mobile applications are built in it like Carbon Nelle.

3. Gets Slow in Speed

Python executes with the help of an interpreter instead of the compiler, which causes it to slow down because compilation and execution help it to work normally. On the other hand, it is fast for many web applications too.

4. Run-time Errors

The Python language is dynamically typed so it has many design restrictions that are reported by some Python developers. It is even seen that it requires more testing time, and the errors show up when the applications are finally run.

5. Underdeveloped Database Access Layers

As compared to popular technologies like JDBC and ODBC, Python's database access layer is found to be bit underdeveloped and primitive. However, it cannot be applied in the enterprises that need smooth interaction of complex legacy data.

1.7.4 About Jupyter Notebook

The Jupyter Notebook is an interactive computing environment that enables user to author documents that include:

Jupyter Notebook is great for the following use cases:

- learn and try out Python
- data processing / transformation
- numeric simulation
- statistical modelling
- machine learning

Components of Jupyter Notebook

The Jupyter Notebook combines three components:

• **The Notebook Web Application:** An interactive web application for writing code and running code interactively and authoring notebook documents.

• **Kernels:** Separate processes started by the notebook web application that run's users code in a given language and returns output back to the notebook web application. The kernel also handles things like computation for interactive widgets, tab completion and introspection.

• **Notebook Documents:** Self-contained documents that contain a representation of all content visible in the notebook web application, including inputs and outputs of the computations etc. Each notebook document has its own kernel.

1.7.5 Installing Jupyter Notebook

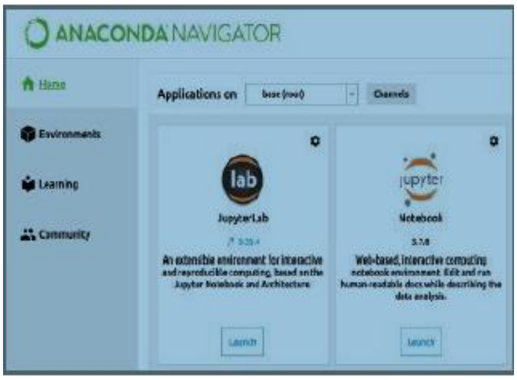
As of late 2019, there are two major environments that you can use to run Jupyter Notebooks: Jupyter Notebook (not to be confused with the Jupyter notebook files themselves, which have an .ipynb extension), and the newer Jupyter Lab. Jupyter Notebook is widely used and well- documented and provides a simple file browser along with the environment for creating, editing, and running the notebooks. Jupyter Lab is more complex, with a user environment more reminiscent of an Integrated Development Environment (discussed in previous Programming Historian tutorials for Windows, Mac, and Linux). While Jupyter Lab is meant to eventually replace Jupyter Notebook, there is no indication that Jupyter Notebook will stop being supported anytime soon. Because of its comparative simplicity and ease of use for beginners, this tutorial uses Jupyter Notebook as the software for running notebook files.

1.7.6 LAUNCHING JUPYTER NOTEBOOK:

Assuming you've already installed Anaconda as described above, you can launch Anaconda Navigator like any other software application. (You can close the prompt about creating an Anaconda Cloud account; you don't need an account to work with Anaconda.) On the home screen, you should see a set of icons and brief blurbs about each application included with Anaconda.

Click on the "Launch" button under the *Jupyter Notebook* icon.

If you prefer to use the command line instead of Anaconda Navigator, once you have Anaconda installed, you should be able to open a new Terminal window (Mac) or Command Prompt (Win) and run jupyter notebook to launch the web browser with the Jupyter Notebook application. If you are using the command line to launch Jupyter Notebook, pay attention to the directory you are in when you launch it. That folder becomes the home directory that will appear immediately in the Jupyter Notebook interface, as described below.



An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Fig. 1.7.6

User Interface of Jupyter Notebook

If you create a new notebook or open an existing one, you will be taken to the notebook user interface (UI). This UI allows you to run code and author notebook documents interactively. The notebook UI has the following main areas:

- Menu
- Toolbar
- Notebook area and cells

Creating A New Notebook

Inside the *notebooks* folder, create a new Jupyter notebook to use to convert the dates for your research project. Click the “New” button in the upper right of the Jupyter Notebook file browser interface. If you’ve just installed Anaconda as described above, your only option will be to create a Jupyter notebook using the Python 3 *kernel* (the backend component that actually runs the code written in the notebook), but we’ll discuss below how to add kernels for other languages. Click on “Python 3”, and Jupyter Notebook will open a new tab with the interface for Jupyter notebooks themselves. By default, the notebook will be named “Untitled”; you can click on that text at the top of the screen to rename it.

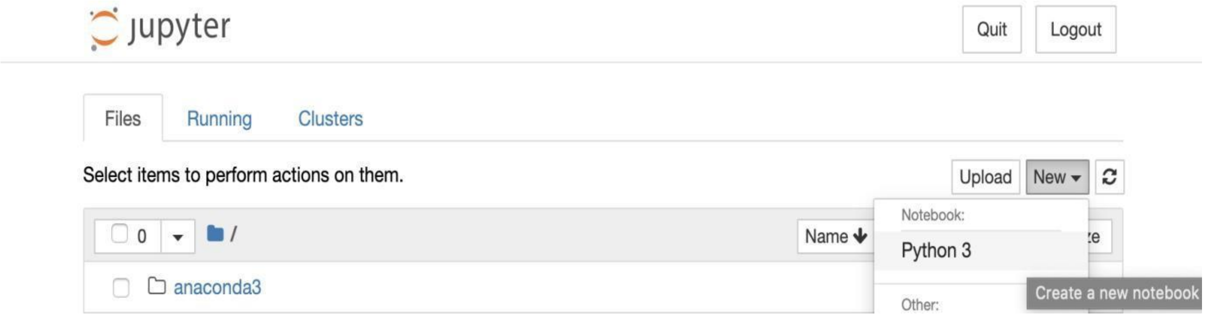


Fig. 1.7.6.1

Working with The Notebook

The notebook itself consists of cells. A first empty cell is already available after having created the new notebook.

This cell is of type “Code” and you can start typing in Python code directly. Executing code in this cell can be done by either clicking on the run cell button or hitting Shift + Return keys.

The resulting output becomes visible right underneath the cell.

The next empty code cell is created automatically, and you can continue to add further code to that cell.

You can change the cell type from Code to Markdown to include explanatory text in your notebook.

To change the type, you can use the dropdown input control.

Once switched the type to Markdown you can start typing in markdown code.

After having entered the markdown code you can compile the cell by hitting Shift + Return once again. The markdown editor cell is then replaced with the output.

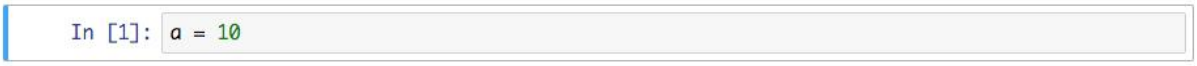
If you want to change the markdown code again you can simply click into the compiled result and the editor mode opens again.

Edit and Command Mode

If a cell is active two modes distinguished:

command mode:

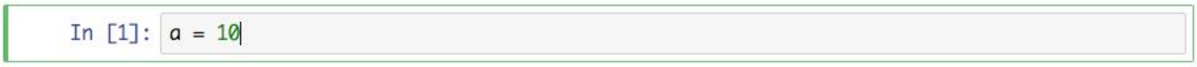
If you just click in one cell the cell is opened in command mode which is indicated by a blue border on the left.



```
In [1]: a = 10
```

Edit mode:

The edit mode is entered if you click into the code area of that cell. This mode is indicated by a green border on the left side of the cell.



```
In [1]: a = 10
```

Checkpoints

Another cool function of Jupyter Notebook is the ability to create checkpoint. By creating a checkpoint you’re storing the current state of the notebook so that you can later on go back to this checkpoint and revert changes which have been made to the notebook in the meantime.

To create a new checkpoint for your notebook, select menu item Save and Checkpoint from the

File menu. The checkpoint is created, and the notebook file is saved. If you want to go back to that checkpoint at a later point in time you need to select the corresponding checkpoint entry from menu File → Revert to Checkpoint.

Exporting the Notebook

Jupyter Notebook gives you several options to export your notebook. Those options can be found in menu File → Download as.

2.Literature survey

At present many researching focussing on using machine learning to detect anomalies or intrusions in a network and thereby filter the malicious packets. This paper is about choosing a machine learning algorithm that best suits for the detection of the various types of intrusions. This paper contains the training results of various models which gives an idea to choose the best suitable algorithm for Packet Filtering.

[1] T. Saranyaa, S. Sridevib, C. Deisyc, Tran Duc Chungd, M.K.A. Ahamed Khane, Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review, Elsevier (CoCoNet'19).

[2] Sudhakar, Sushil Kumar, Botnet Detection Techniques and Research Challenges, ResearchGate 2019.

[3] Yadu Singh, Future Correlation Matters a Lot…, LinkedIn 2021

[4]SH Kok, Azween Abdullah, NZ Jhanjhi, Mahadevan Supramaniam, A Review of Intrusion Detection System using Machine Learning Approach, ResearchGate 2019.

[5] Mrutyunjaya Panda, Ajith Abraham, Swagatam Das, Manas Ranjan Patra, Network Intrusion Detection System: A Machine Learning Approach, ResearchGate 2011.

[6] U. S. Musa, S. Chakraborty, M. M. Abdullahi and T. Maini, "A Review on Intrusion Detection System using Machine Learning Techniques," 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2021, pp. 541-549, doi: 10.1109/ICCCIS51004.2021.9397121.

[7] M. K. Yadav and K. P. Sharma, "Intrusion Detection System using Machine Learning Algorithms: A Comparative Study," 2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC), 2021, pp. 415-420, doi: 10.1109/ICSCCC51823.2021.9478086.

[8] Mohammad Almseidin, Maen Alzubi, Szilveszter Kovacs and Mouhammd Alkasassbeh, Evaluation of Machine Learning Algorithms for Intrusion Detection System, arxiv.

[9] C. Fleizach and S. Fukushima, A naive bayes classifier on 1998 kdd cup, 1998.

[10] A. Cutler and G. Zhao, Pert-perfect random tree ensembles, Computing Science and Statistics, vol. 33, pp. 490–497, 2001.

[11]L. Breiman, Random forests, Machine learning, vol. 45, no. 1, pp. 5–32, 2001.

[12] N. Friedman, D. Geiger, and M. Goldszmidt, “Bayesian network classifiers,” Machine learning, vol. 29, no. 2-3, pp. 131–163, 1997.

3.SYSTEM ANALYSIS

3.1 Existing system:

- A Packet Filtering works by actively scanning forwarded network traffic for malicious activities and known attack patterns.
- The Filtering engine analyzes network traffic and continuously compares the bitstream with its internal signature database for known attack patterns.
- An IPS might drop a packet determined to be malicious and follow up this action by blocking all future traffic from the attacker’s IP address or port.
- Legitimate traffic can continue without any perceived disruption in service.
- An IPS will typically record information related to observed events, notify security administrators, and produce reports. To help secure a network, an IPS can automatically receive Prevention and security updates in order to continuously monitor and block emerging Internet threats.
- Intrusion Prevention Systems can also perform more complicated observation and analysis, such as watching and reacting to suspicious traffic patterns or packets. Detection mechanisms can include:
    - Address matching
    - HTTP string and substring matching
    - Generic pattern matching
    - TCP connection analysis
    - Packet anomaly detection
    - Traffic anomaly detection
    - TCP/UDP port matching

3.2 Proposed system:

- Although there are many methods to detect and prevent the security attacks on a network, the attacks are also becoming sophisticated and it has been very difficult to detect them using pre-defined rules.
- Several machine learning algorithms can be used to effectively detect a network intrusion and thus preventing them.
- A dataset that contains data packets with 43 different types of features with more than 40000 records with different types of attacks is used for training.
- This data is trained using various Machine Learning Algorithms and an effective algorithm is used to generate the final model.
- This model is used predict the intrusion level on a network on every packet that is coming into the network.
- If a packet is detected as a packet that is causing intrusion, that packet is dropped and further the ip address is blocked from sending or receiving the packets.

In this way the Intrusion Prevention can be taken to the next level.

3.3 Feasibility Study:

A feasibility study is used to determine the feasibility of an idea, for example to ensure that a project is legally and technically feasible as well as economically justifiable. The feasibility study is generally conducted before undertaking any initiative concerning a project, including planning. It is one of the fundamental factors, if not the most important, which determine whether the project should be carried out or not.

Three Key Considerations involved in the feasibility analysis are:

- Economic feasibility
- Technical feasibility
- Operational feasibility

3.3.1Economic Feasibility:

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

At least 7 to 10 million people in the world are affecting with degenerative disorder. This is one disease which affects the brain cells and that indirectly affects the movements and voice. This disease will affect in 5 stages. People affected with this disease are identified late. So, the cost for treatment of the disease is more. This is a chronic and no cure yet. So, early detection of this disease will increase the life span by starting the treatment at the early stage. If the treatment is started early the cost can be affordable by all the people and the life span can be increased. So, our project is economically feasible.

1. All the necessary technology exists to develop the system.
2. The existing resources are capable and can hold all necessary data.
3. The system is too flexible, and it can be expanded further.
4. The system can give guarantees of accuracy, ease of use, reliability, and the data security.

3.3.2 Technical Feasibility:

A large part of determining resources has to do with assessing technical feasibility. It considers the technical requirements of the proposed project. The technical requirements are then compared with technical capability of the organization. The systems project is considered technically feasible if the internal technical capability is sufficient to support the project requirements.

The system can be technically feasible on the following grounds.

3.3.3 Operational Feasibility:

Operational feasibility is a measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

The present system can be justified “Detection of Degenerative Disorder Using Machine Learning” as operationally feasible based on the following grounds:

- The proposed system is approximately 94% accurate so, the error rate will be low.
- The interface is easy to learn, and the system needs minimum effort on user’s part.

4.System specifications

4.1 Functional requirements:

Functional requirements are product features or functions those developers must implement to enable users to accomplish their tasks. So, it’s important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behaviour under specific conditions.

For instance: A search feature allows a user to hunt among various invoices if they want to credit an issued invoice. Here’s another simple example: As a guest, I want a sofa that I can sleep on overnight. Requirements are usually written in text, especially for Agile-driven projects. However, they may also be visuals.

Here are the most common formats and documents:

- Software requirements specification document
- Use cases
- User stories
- Work Breakdown Structure (WBS) (functional decomposition)
- Prototypes
- Models and diagrams

4.2 Non-functional requirements:

Non-functional requirements describe user-visible aspects of the system that are not directly related to functionally of the system. These requirements define what qualities is exhibited by the system.

The following are the non-function requirements of the system:

1. Performance (arrives at solution faster than hard computing approach)
2. Usability
3. Scalability

4.Maintainability

5. Interoperability

4.3 Hardware requirements:

Processor: Intel Pentium -IV Processor machine

Speed: 1.1Ghz

RAM: 1GB

Hard Disk: 110GB

Input Devices: Keyboard, Mouse

Output Devices: Monitor

4.4 Software Requirements:

- Language – Python (3.8 and above)
- Modules
    - Pandas - for data manipulation and analysis
    - NumPy - to process multi-dimensional arrays
    - Matplotlib – plotting library for python
    - Seaborn – for statistical graphs in python
    - Time – for calculating the training and testing times
- Dataset - KDD Cup 1999
    - Properties
        - Size (before preprocessing) - 4898431 rows × 43 columns
        - Size (after preprocessing) - 4898431 rows × 33 columns
        - Train Data Size – 3281948 rows x 31 columns
        - Test Data Size – 1616483 rows x 31 columns
        - Split factor - 0.33

5.SYSTEM DESIGN

5.1 System Architecture:

This system architecture shows how the internal or external process will be taken place. Describes how it is going to be implemented and how it executed. This shows the step-by-step procedure for implementation.

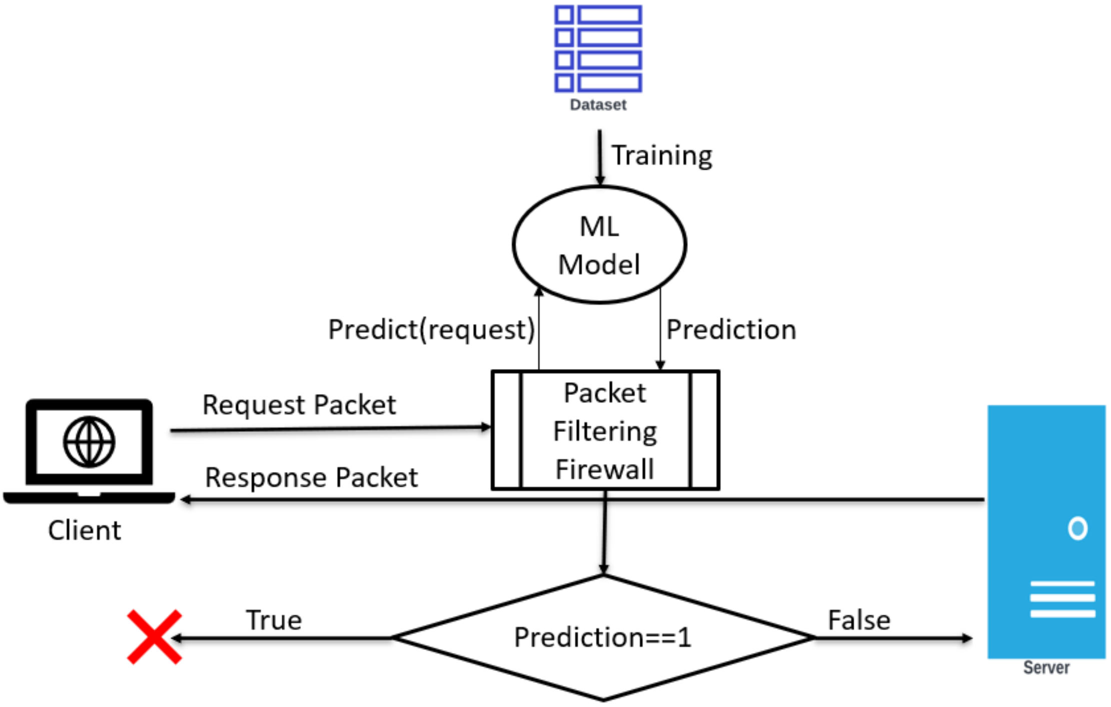


Fig. 5.1 Architecture

5.2 Dataset Description

Abstract

This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between ``bad'' connections, called intrusions or attacks, and ``good'' normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment.

Information files:

- task description. This is the original task description given to competition participants.

Data files:

- kddcup.names A list of features.
- kddcup.data.gz The full data set (18M; 743M Uncompressed)
- kddcup.data_10_percent.gz A 10% subset. (2.1M; 75M Uncompressed)
- kddcup.newtestdata_10_percent_unlabeled.gz (1.4M; 45M Uncompressed)
- kddcup.testdata.unlabeled.gz (11.2M; 430M Uncompressed)
- kddcup.testdata.unlabeled_10_percent.gz (1.4M;45M Uncompressed)
- corrected.gz Test data with corrected labels.
- training_attack_types A list of intrusion types.
- typo-correction.txt A brief note on a typo in the data set that has been corrected (6/26/07)

5.3 UML Diagrams

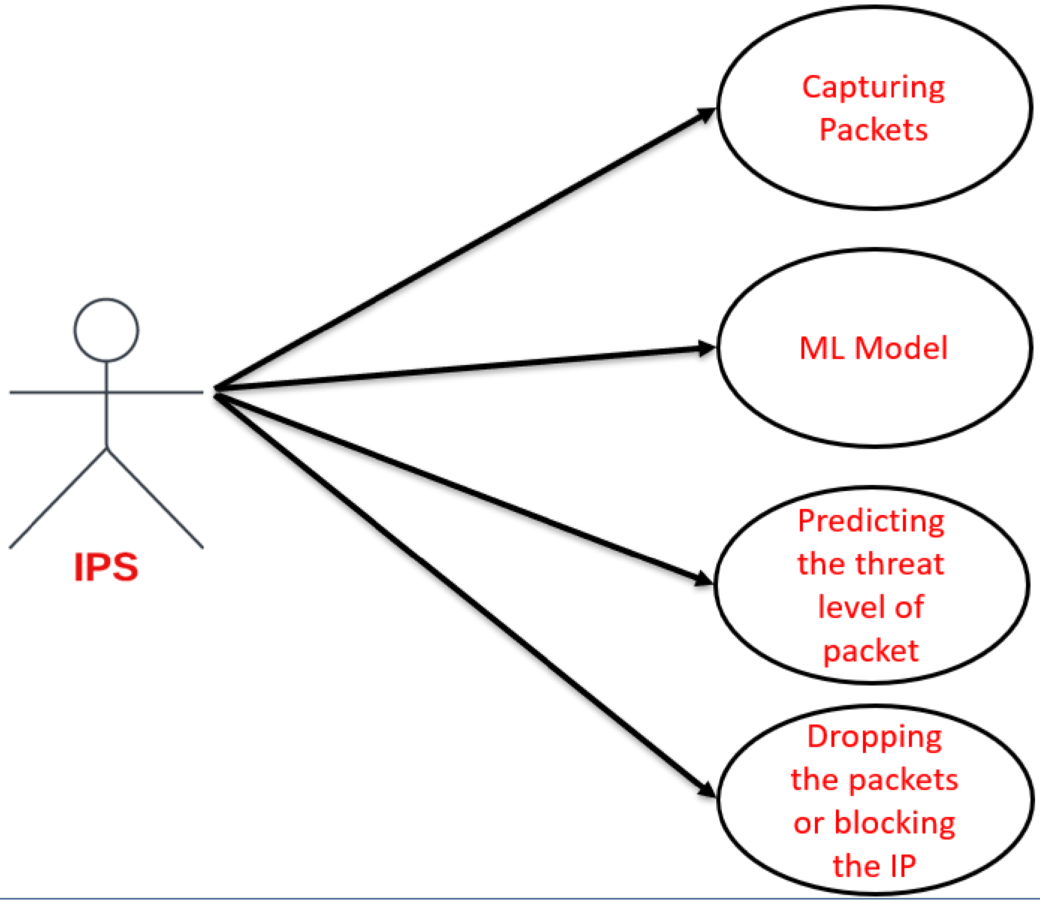

Capturing Packets
ML Model
Predicting the threat level of packet
Dropping the packets or blocking the IP
IPS

Fig. 5.3.1 USE-CASE DIAGRAM

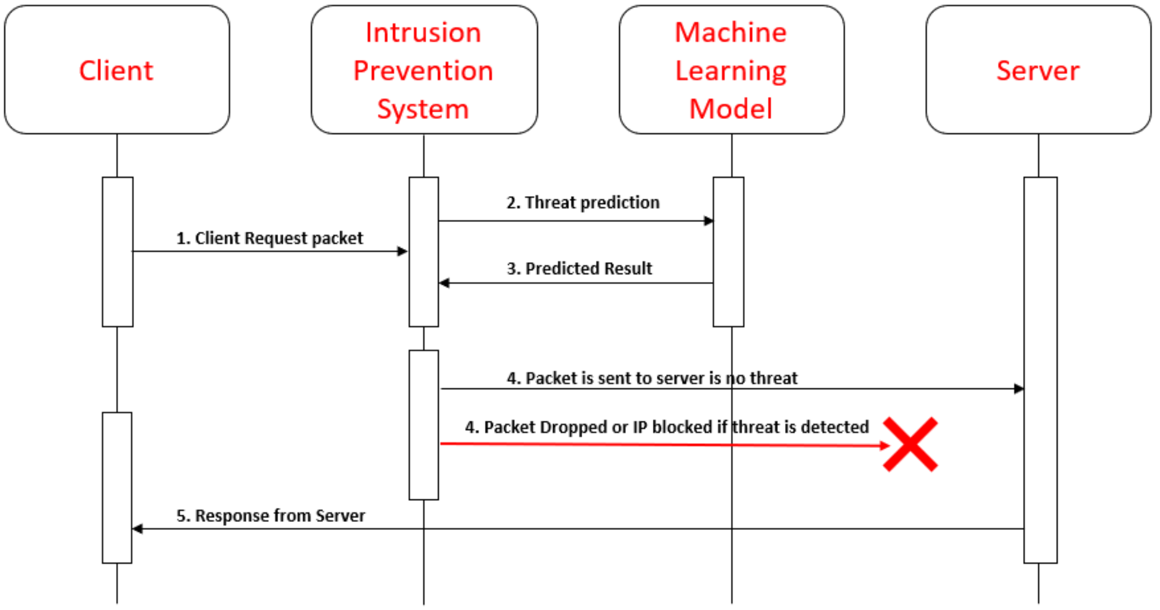

Client
Intrusion Prevention System
Machine Learning Model
Server
1. Client Request packet
2. Threat prediction
3. Predicted Result
4. Packet is sent to server is no threat
4. Packet Dropped or IP blocked if threat is detected
5. Response from Server

Fig. 5.3.2 SEQUENCE DIAGRAM

la

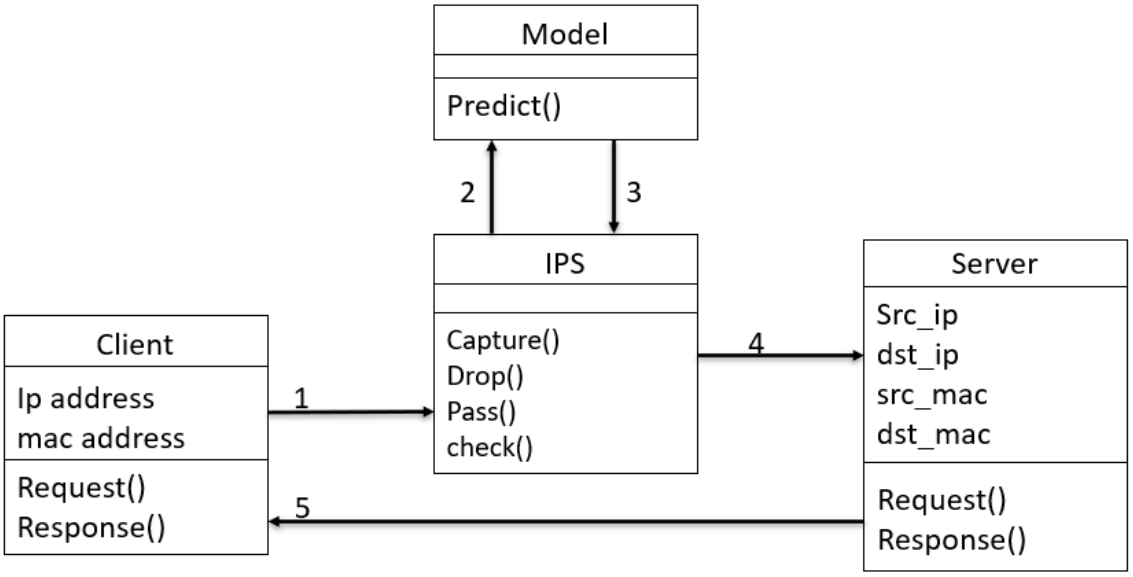
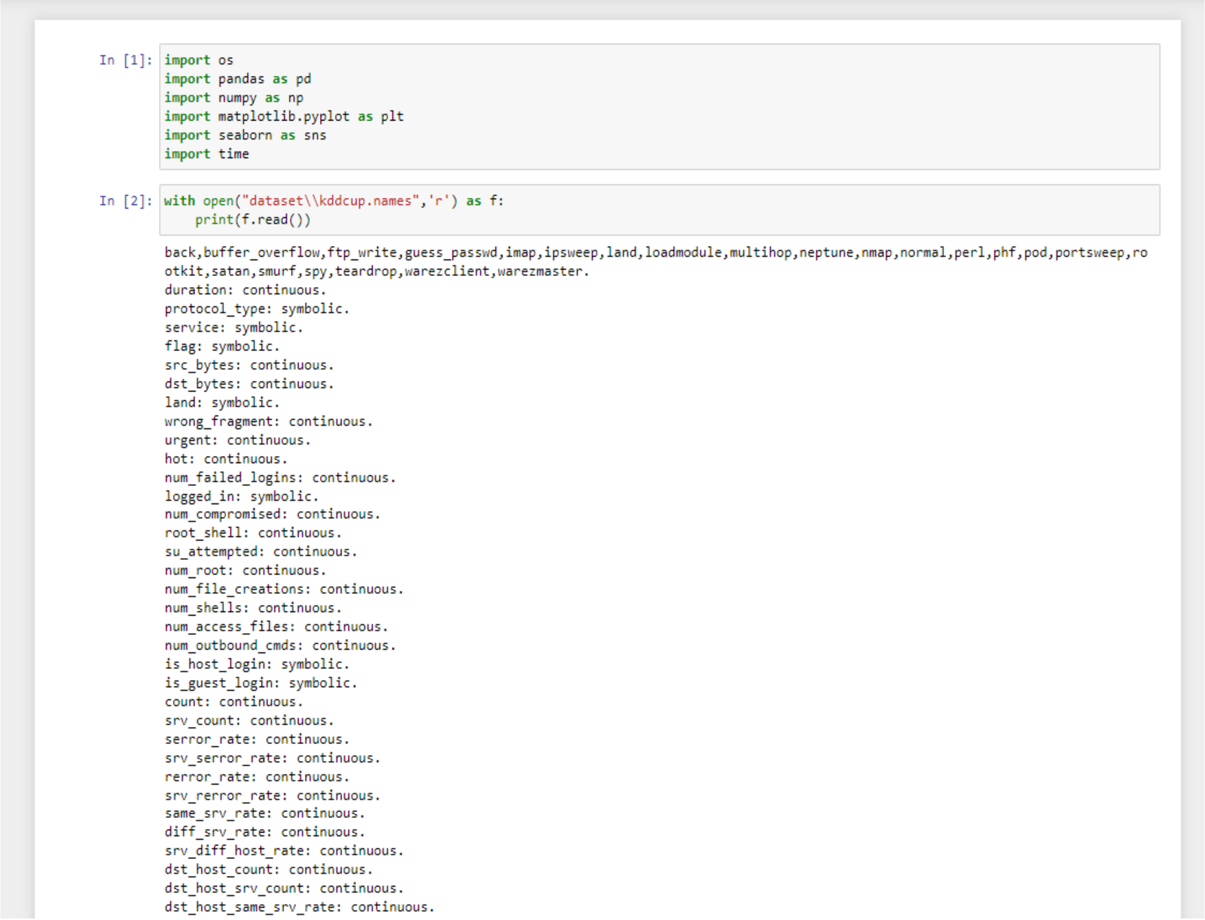


Fig. 5.3.3 CLASS DIAGRAM

6. SYSTEM IMPLEMENTATION

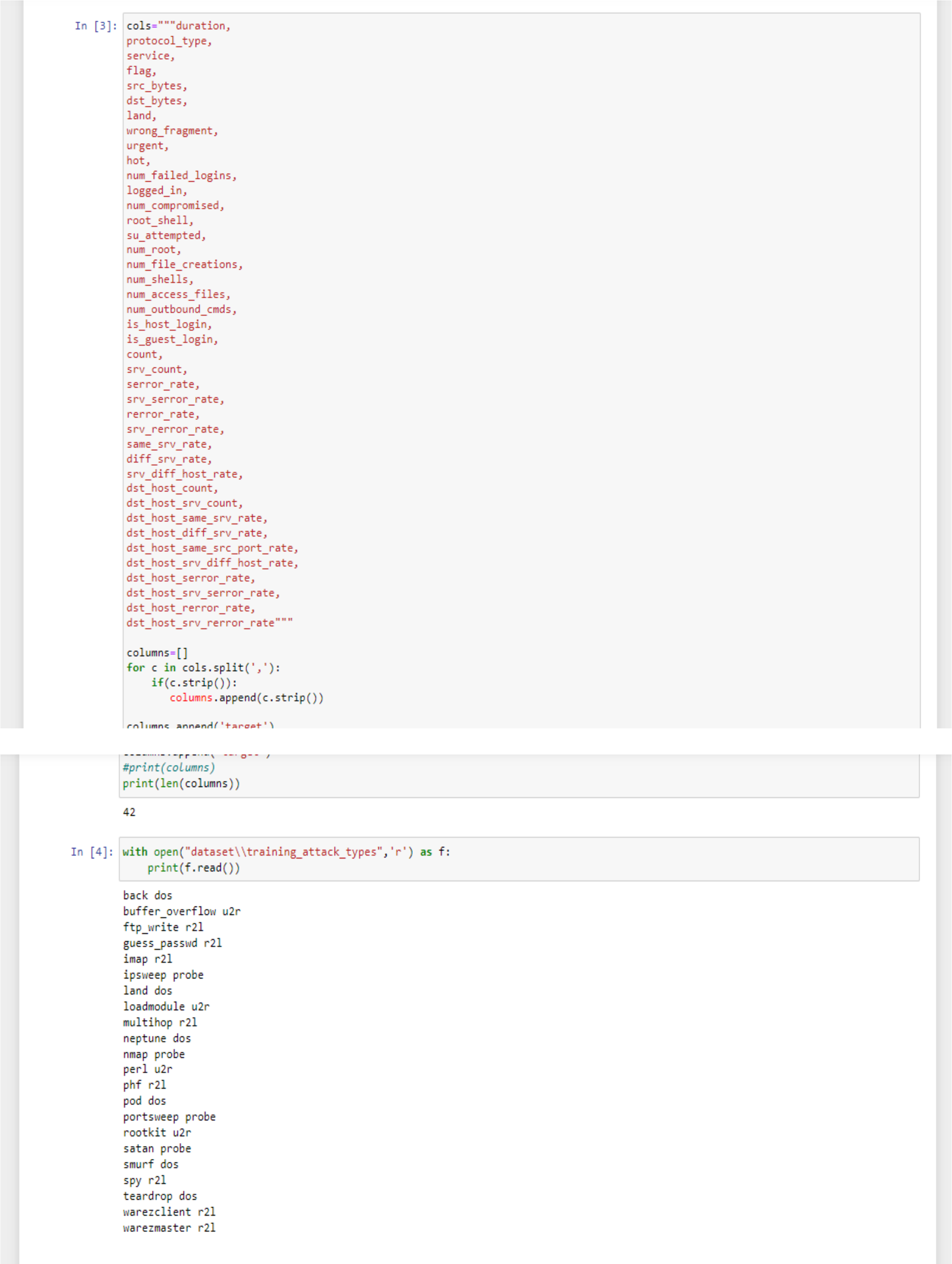
6.1 MACHINE LEARNING MODEL TRAINING CODE



```
In [1]: import os
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import time
```

```
In [2]: with open("dataset\\kddcup.names",'r') as f:
            print(f.read())
```

```
back,buffer_overflow,ftp_write,guess_passwd,imap,ipsweep,land,loadmodule,multihop,neptune,nmap,normal,perl,phf,pod,portsweep,ro
otkit,satan,smurf,spy,teardrop,warezclient,warezmaster.
duration: continuous.
protocol_type: symbolic.
service: symbolic.
flag: symbolic.
src_bytes: continuous.
dst_bytes: continuous.
land: symbolic.
wrong_fragment: continuous.
urgent: continuous.
hot: continuous.
num_failed_logins: continuous.
logged_in: symbolic.
num_compromised: continuous.
root_shell: continuous.
su_attempted: continuous.
num_root: continuous.
num_file_creations: continuous.
num_shells: continuous.
num_access_files: continuous.
num_outbound_cmds: continuous.
is_host_login: symbolic.
is_guest_login: symbolic.
count: continuous.
srv_count: continuous.
serror_rate: continuous.
srv_serror_rate: continuous.
rerror_rate: continuous.
srv_rerror_rate: continuous.
same_srv_rate: continuous.
diff_srv_rate: continuous.
srv_diff_host_rate: continuous.
dst_host_count: continuous.
dst_host_srv_count: continuous.
dst_host_same_srv_rate: continuous.
```

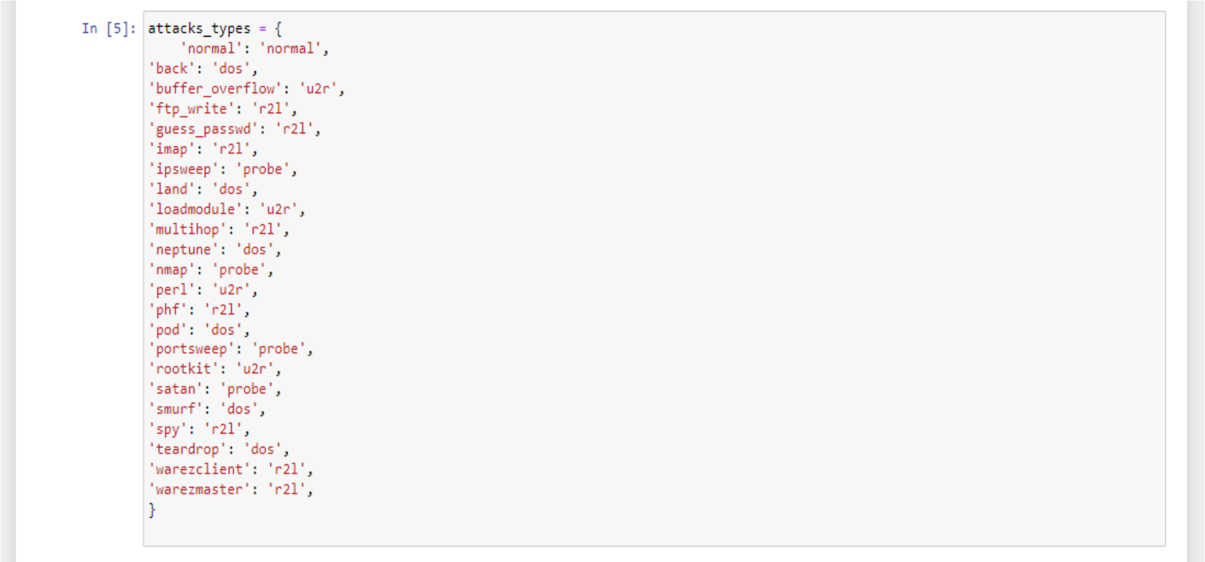


In [3]:
```
cols="""duration,
protocol_type,
service,
flag,
src_bytes,
dst_bytes,
land,
wrong_fragment,
urgent,
hot,
num_failed_logins,
logged_in,
num_compromised,
root_shell,
su_attempted,
num_root,
num_file_creations,
num_shells,
num_access_files,
num_outbound_cmds,
is_host_login,
is_guest_login,
count,
srv_count,
serror_rate,
srv_serror_rate,
rerror_rate,
srv_rerror_rate,
same_srv_rate,
diff_srv_rate,
srv_diff_host_rate,
dst_host_count,
dst_host_srv_count,
dst_host_same_srv_rate,
dst_host_diff_srv_rate,
dst_host_same_src_port_rate,
dst_host_srv_diff_host_rate,
dst_host_serror_rate,
dst_host_srv_serror_rate,
dst_host_rerror_rate,
dst_host_srv_rerror_rate"""

columns=[]
for c in cols.split(','):
    if(c.strip()):
        columns.append(c.strip())

columns.append('target')
#print(columns)
print(len(columns))
```

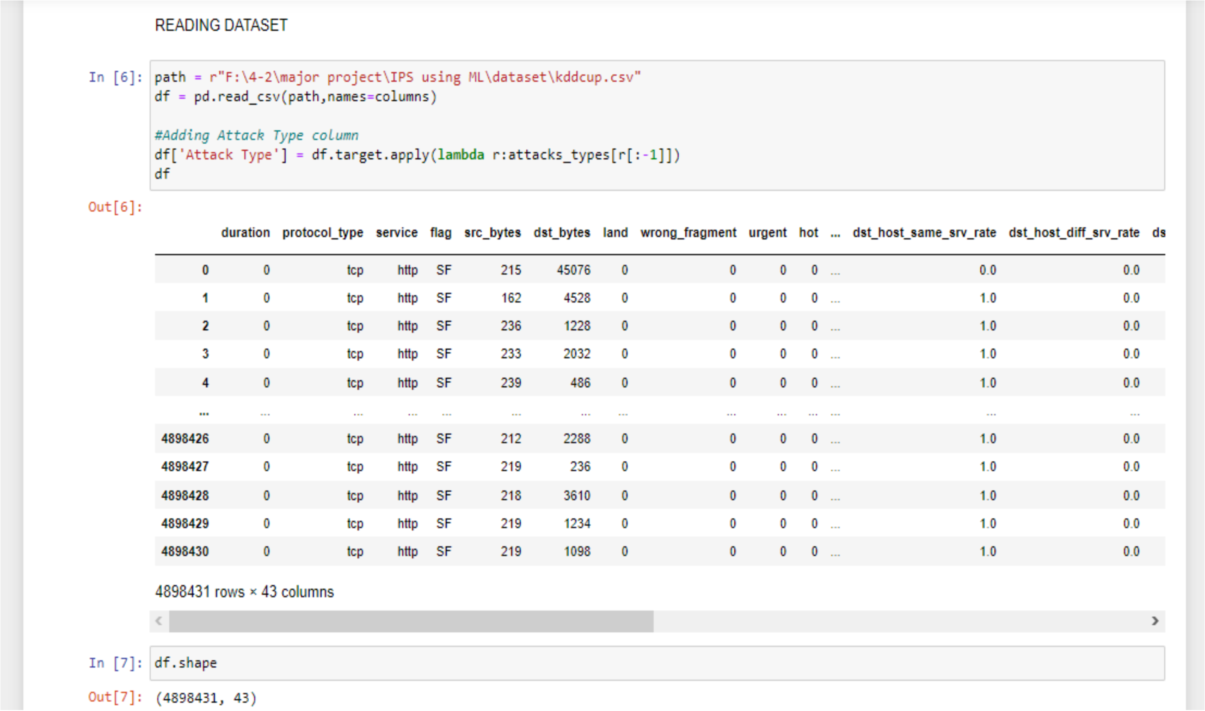
```
42
```

In [4]:
```
with open("dataset\\training_attack_types",'r') as f:
    print(f.read())
```

```
back dos
buffer_overflow u2r
ftp_write r2l
guess_passwd r2l
imap r2l
ipsweep probe
land dos
loadmodule u2r
multihop r2l
neptune dos
nmap probe
perl u2r
phf r2l
pod dos
portsweep probe
rootkit u2r
satan probe
smurf dos
spy r2l
teardrop dos
warezclient r2l
warezmaster r2l
```



```
In [5]: attacks_types = {
            'normal': 'normal',
        'back': 'dos',
        'buffer_overflow': 'u2r',
        'ftp_write': 'r2l',
        'guess_passwd': 'r2l',
        'imap': 'r2l',
        'ipsweep': 'probe',
        'land': 'dos',
        'loadmodule': 'u2r',
        'multihop': 'r2l',
        'neptune': 'dos',
        'nmap': 'probe',
        'perl': 'u2r',
        'phf': 'r2l',
        'pod': 'dos',
        'portsweep': 'probe',
        'rootkit': 'u2r',
        'satan': 'probe',
        'smurf': 'dos',
        'spy': 'r2l',
        'teardrop': 'dos',
        'warezclient': 'r2l',
        'warezmaster': 'r2l',
        }
```



READING DATASET

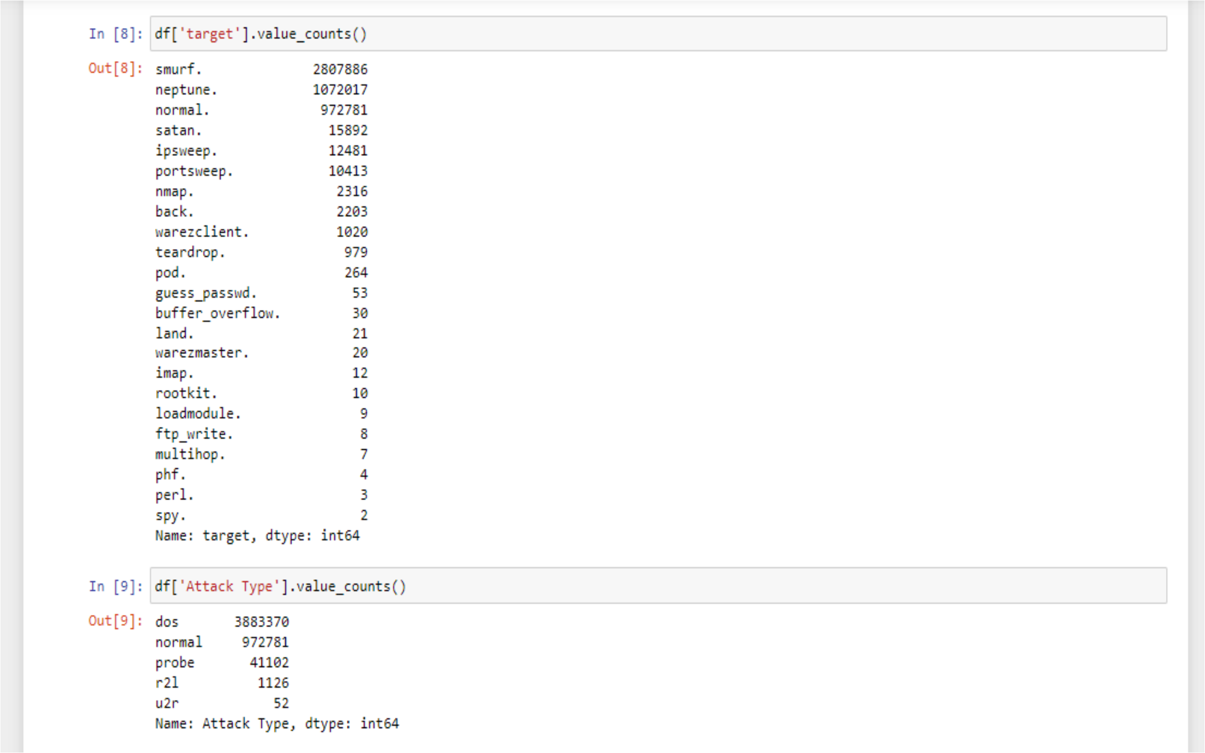
```
In [6]: path = r"F:\4-2\major project\IPS using ML\dataset\kddcup.csv"
        df = pd.read_csv(path,names=columns)

        #Adding Attack Type column
        df['Attack Type'] = df.target.apply(lambda r:attacks_types[r[:-1]])
        df
```

Out[6]:

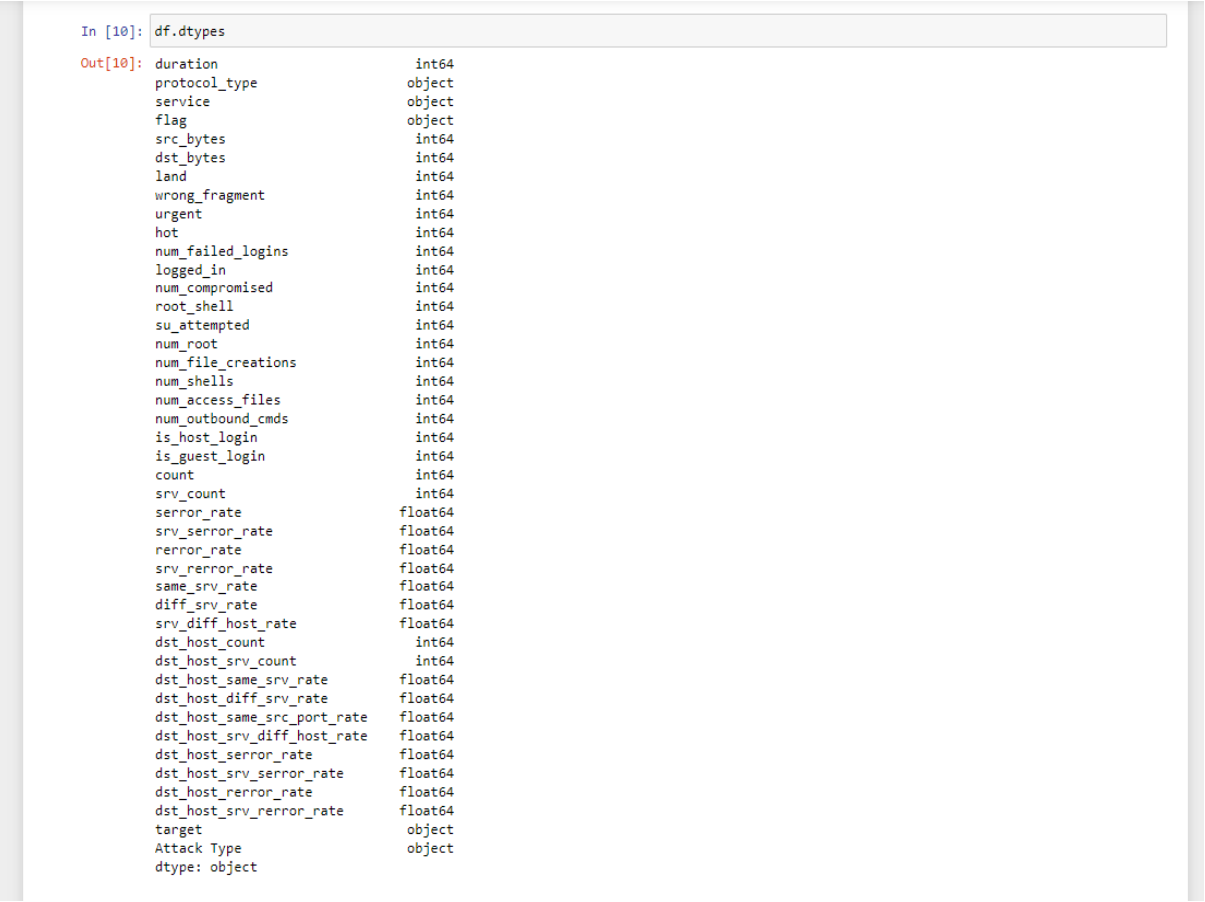
	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	ds
0	0	tcp	http	SF	215	45076	0	0	0	0	...	0.0	0.0	
1	0	tcp	http	SF	162	4528	0	0	0	0	...	1.0	0.0	
2	0	tcp	http	SF	236	1228	0	0	0	0	...	1.0	0.0	
3	0	tcp	http	SF	233	2032	0	0	0	0	...	1.0	0.0	
4	0	tcp	http	SF	239	486	0	0	0	0	...	1.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4898426	0	tcp	http	SF	212	2288	0	0	0	0	...	1.0	0.0	
4898427	0	tcp	http	SF	219	236	0	0	0	0	...	1.0	0.0	
4898428	0	tcp	http	SF	218	3610	0	0	0	0	...	1.0	0.0	
4898429	0	tcp	http	SF	219	1234	0	0	0	0	...	1.0	0.0	
4898430	0	tcp	http	SF	219	1098	0	0	0	0	...	1.0	0.0	

4898431 rows × 43 columns

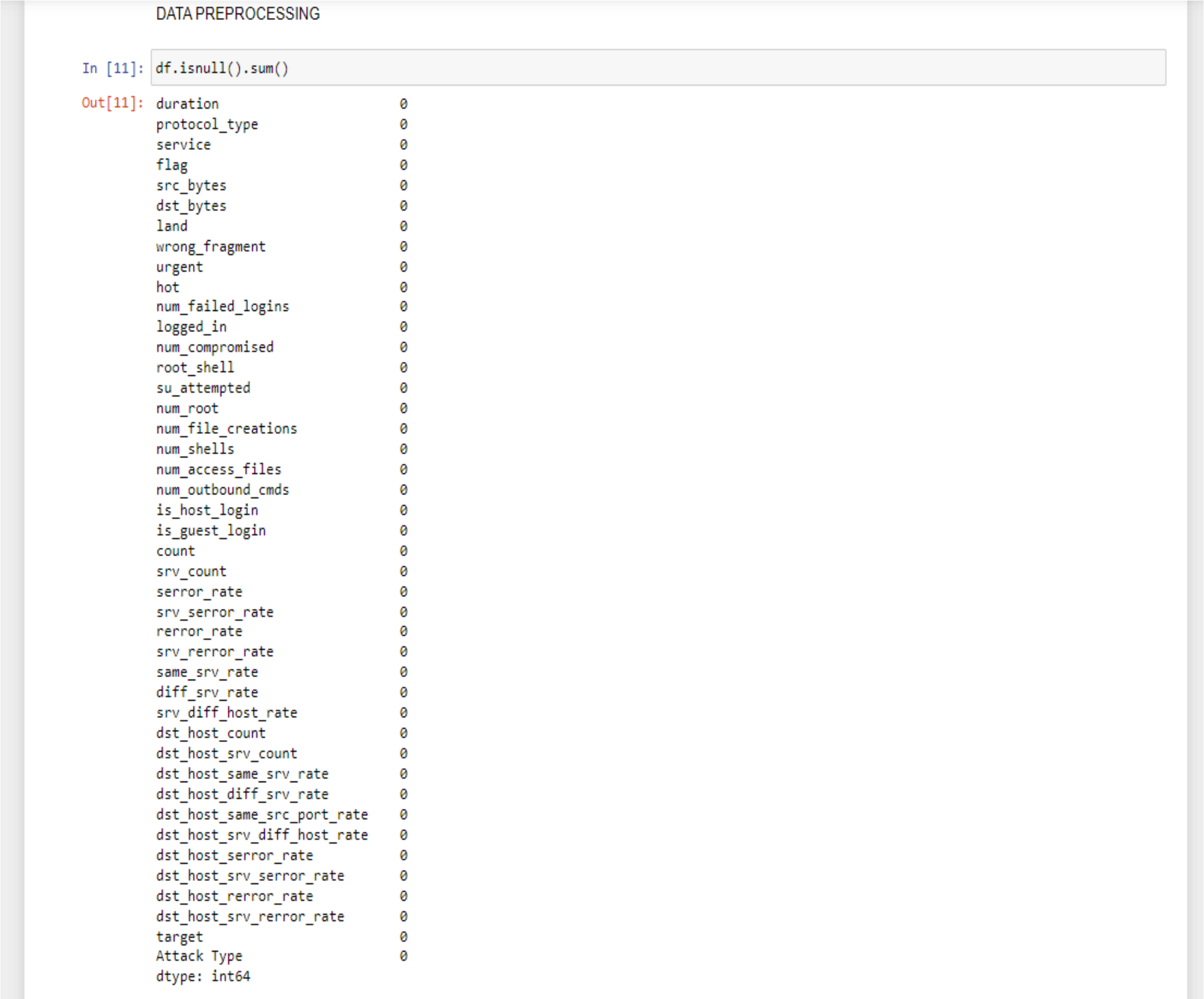
```
In [7]: df.shape

Out[7]: (4898431, 43)
```



```
In [8]: df['target'].value_counts()

Out[8]: smurf.              2807886
        neptune.            1072017
        normal.              972781
        satan.                15892
        ipsweep.              12481
        portsweep.            10413
        nmap.                  2316
        back.                  2203
        warezclient.           1020
        teardrop.               979
        pod.                    264
        guess_passwd.            53
        buffer_overflow.         30
        land.                    21
        warezmaster.             20
        imap.                    12
        rootkit.                 10
        loadmodule.               9
        ftp_write.                8
        multihop.                 7
        phf.                      4
        perl.                     3
        spy.                      2
        Name: target, dtype: int64

In [9]: df['Attack Type'].value_counts()

Out[9]: dos       3883370
        normal     972781
        probe       41102
        r2l          1126
        u2r            52
        Name: Attack Type, dtype: int64
```



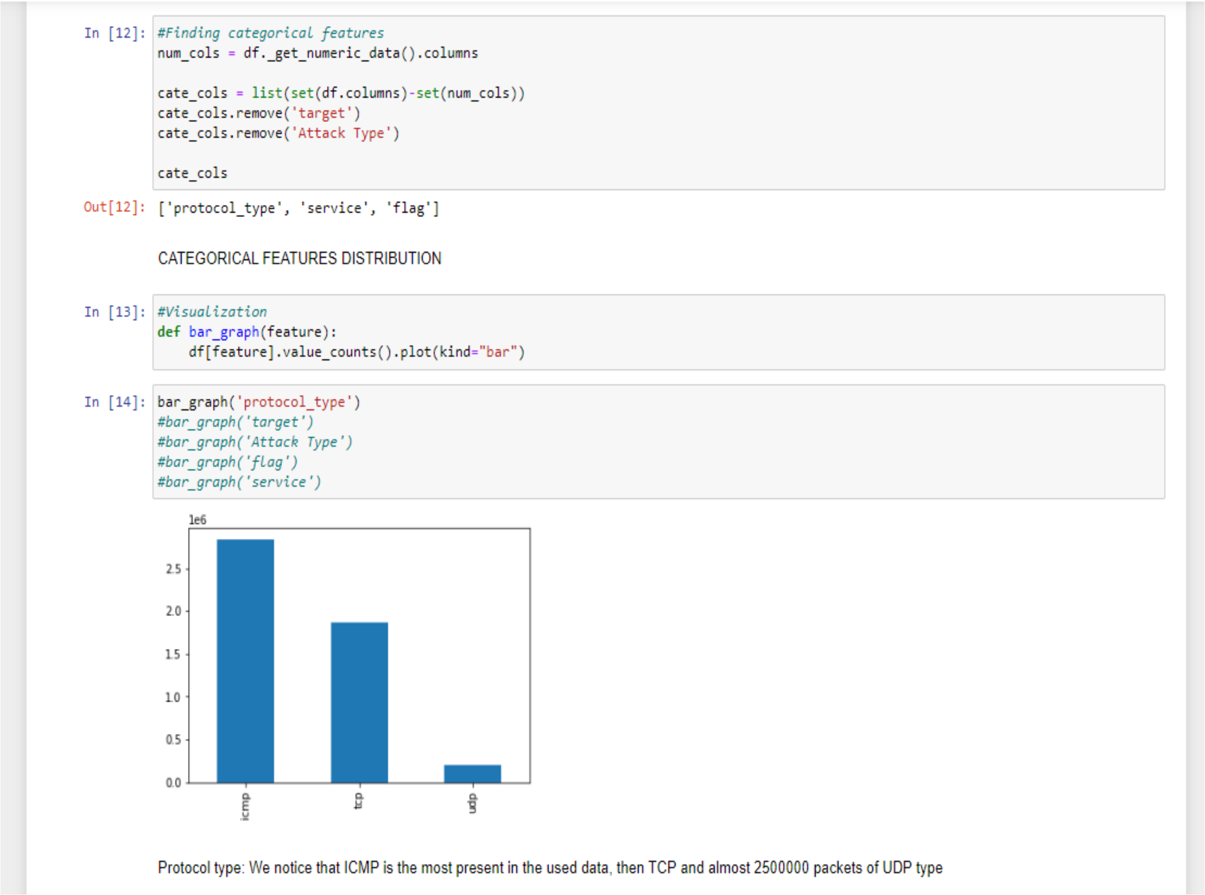
```
In [10]: df.dtypes

Out[10]: duration                          int64
         protocol_type                    object
         service                          object
         flag                             object
         src_bytes                         int64
         dst_bytes                         int64
         land                              int64
         wrong_fragment                    int64
         urgent                            int64
         hot                               int64
         num_failed_logins                 int64
         logged_in                         int64
         num_compromised                   int64
         root_shell                        int64
         su_attempted                      int64
         num_root                          int64
         num_file_creations                int64
         num_shells                        int64
         num_access_files                  int64
         num_outbound_cmds                 int64
         is_host_login                     int64
         is_guest_login                    int64
         count                             int64
         srv_count                         int64
         serror_rate                     float64
         srv_serror_rate                 float64
         rerror_rate                     float64
         srv_rerror_rate                 float64
         same_srv_rate                   float64
         diff_srv_rate                   float64
         srv_diff_host_rate              float64
         dst_host_count                    int64
         dst_host_srv_count                int64
         dst_host_same_srv_rate          float64
         dst_host_diff_srv_rate          float64
         dst_host_same_src_port_rate     float64
         dst_host_srv_diff_host_rate     float64
         dst_host_serror_rate            float64
         dst_host_srv_serror_rate        float64
         dst_host_rerror_rate            float64
         dst_host_srv_rerror_rate        float64
         target                           object
         Attack Type                      object
         dtype: object
```



DATA PREPROCESSING

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
duration                       0
protocol_type                  0
service                        0
flag                           0
src_bytes                      0
dst_bytes                      0
land                           0
wrong_fragment                 0
urgent                         0
hot                            0
num_failed_logins              0
logged_in                      0
num_compromised                0
root_shell                     0
su_attempted                   0
num_root                       0
num_file_creations             0
num_shells                     0
num_access_files               0
num_outbound_cmds              0
is_host_login                  0
is_guest_login                 0
count                          0
srv_count                      0
serror_rate                    0
srv_serror_rate                0
rerror_rate                    0
srv_rerror_rate                0
same_srv_rate                  0
diff_srv_rate                  0
srv_diff_host_rate             0
dst_host_count                 0
dst_host_srv_count             0
dst_host_same_srv_rate         0
dst_host_diff_srv_rate         0
dst_host_same_src_port_rate    0
dst_host_srv_diff_host_rate    0
dst_host_serror_rate           0
dst_host_srv_serror_rate       0
dst_host_rerror_rate           0
dst_host_srv_rerror_rate       0
target                         0
Attack Type                    0
dtype: int64
```



In [12]:
```
#Finding categorical features
num_cols = df._get_numeric_data().columns

cate_cols = list(set(df.columns)-set(num_cols))
cate_cols.remove('target')
cate_cols.remove('Attack Type')

cate_cols
```

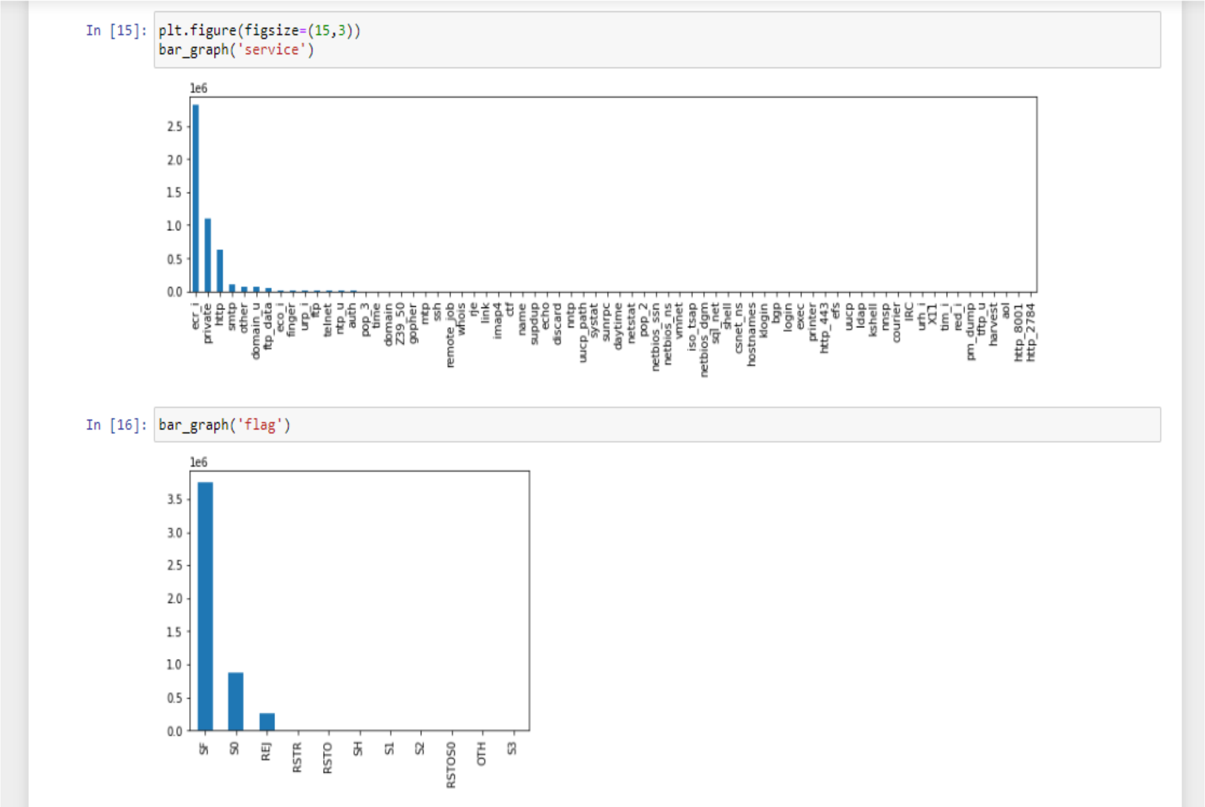
Out[12]:
```
['protocol_type', 'service', 'flag']
```

CATEGORICAL FEATURES DISTRIBUTION

In [13]:
```
#Visualization
def bar_graph(feature):
    df[feature].value_counts().plot(kind="bar")
```

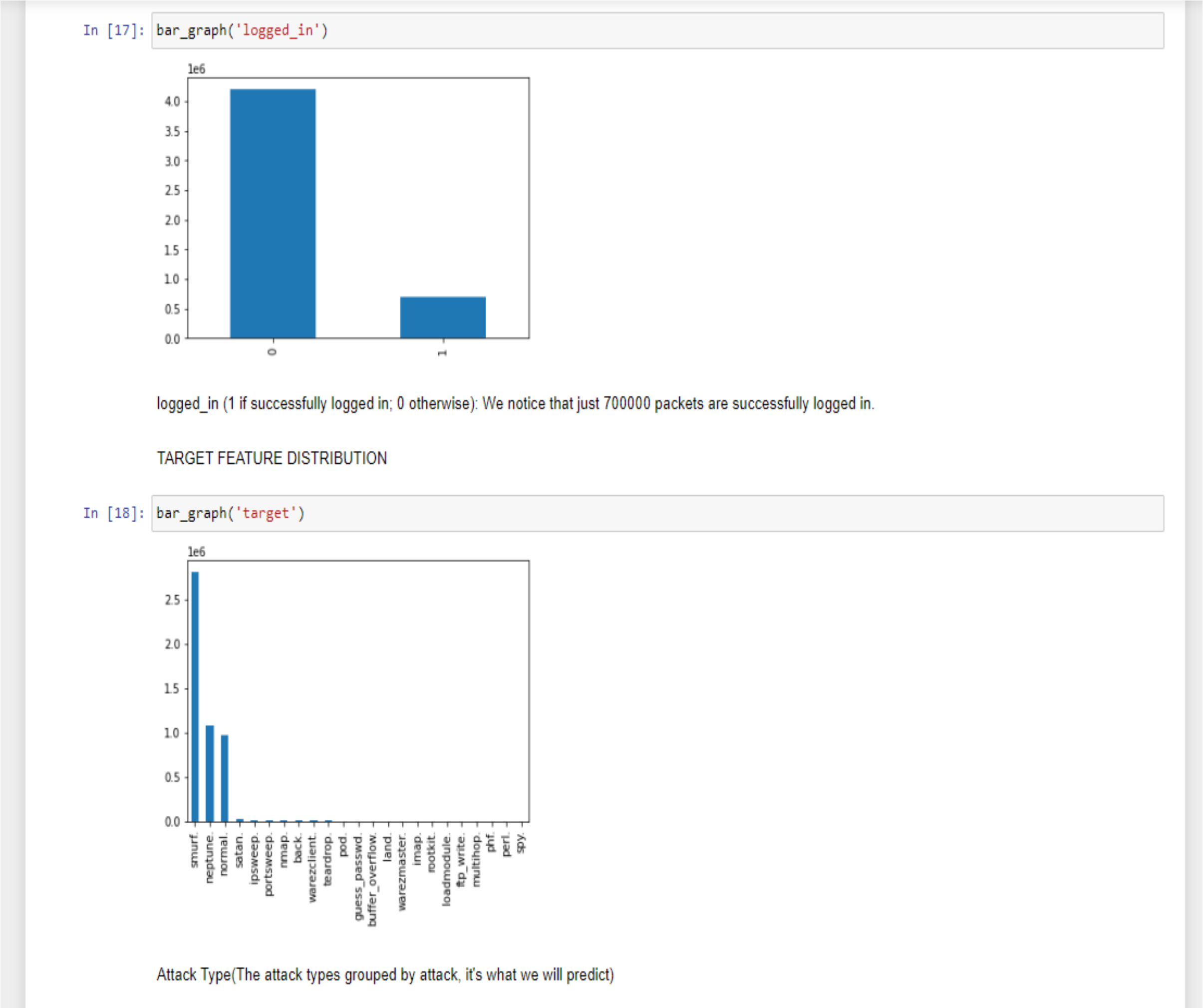
In [14]:
```
bar_graph('protocol_type')
#bar_graph('target')
#bar_graph('Attack Type')
#bar_graph('flag')
#bar_graph('service')
```

Protocol type: We notice that ICMP is the most present in the used data, then TCP and almost 2500000 packets of UDP type



In [15]:
```
plt.figure(figsize=(15,3))
bar_graph('service')
```

In [16]:
```
bar_graph('flag')
```



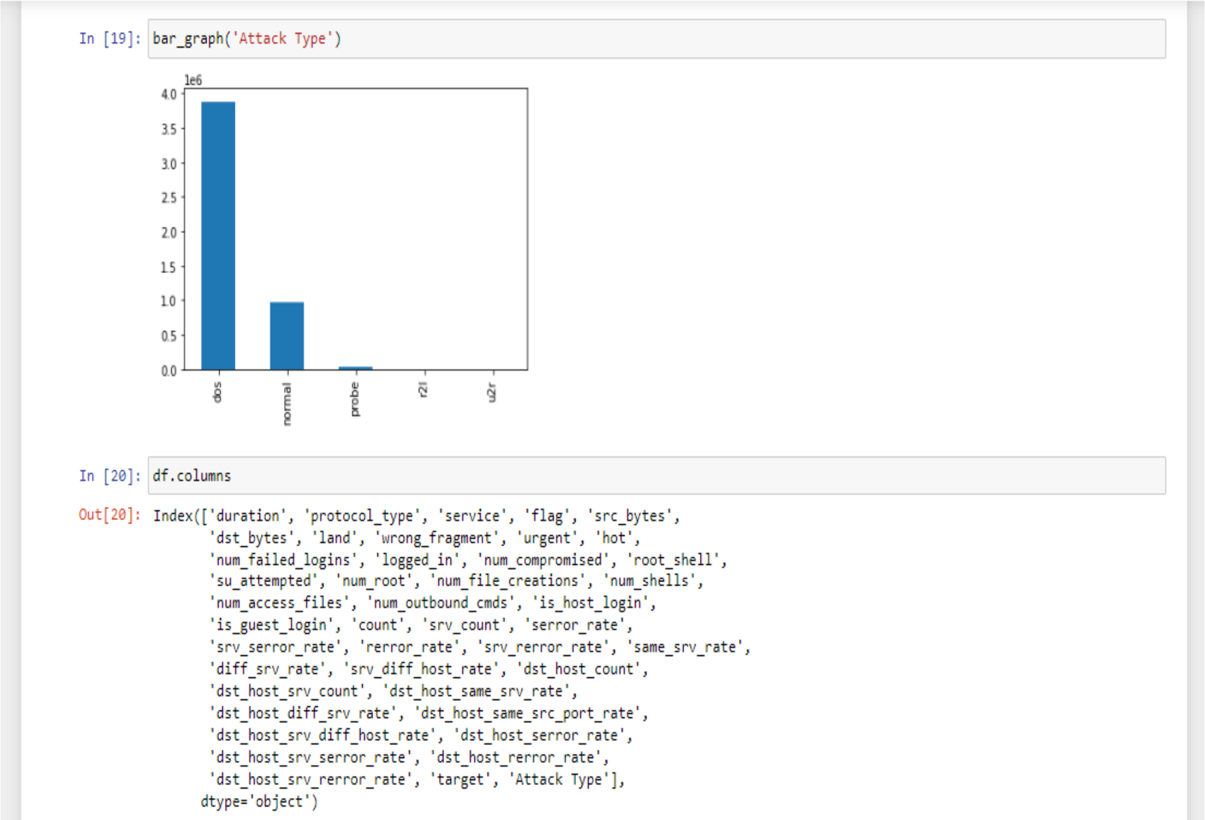
```
In [17]: bar_graph('logged_in')
```

logged_in (1 if successfully logged in; 0 otherwise): We notice that just 700000 packets are successfully logged in.

TARGET FEATURE DISTRIBUTION

```
In [18]: bar_graph('target')
```

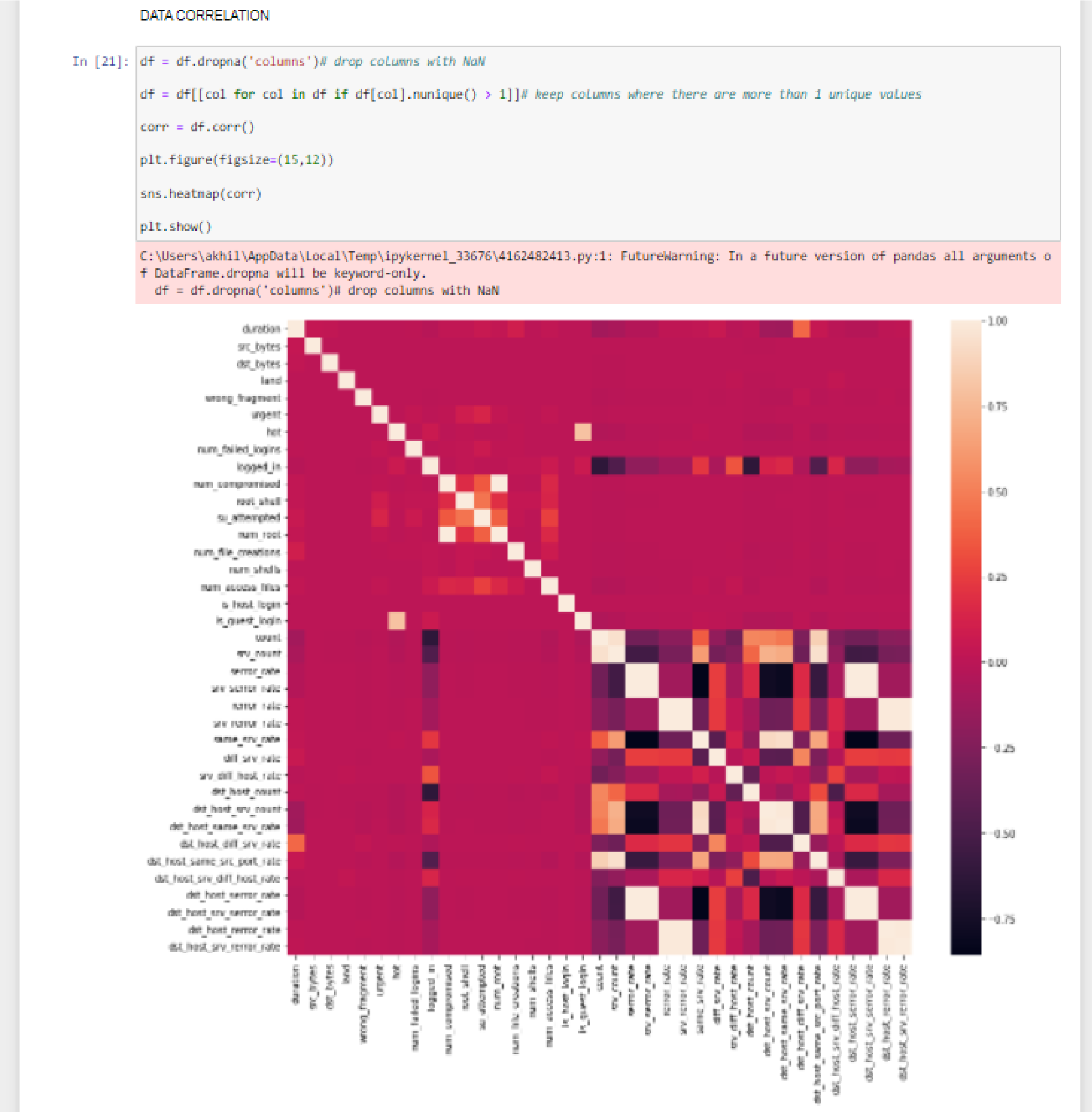
Attack Type(The attack types grouped by attack, it's what we will predict)



```
In [19]: bar_graph('Attack Type')
```

```
In [20]: df.columns
```

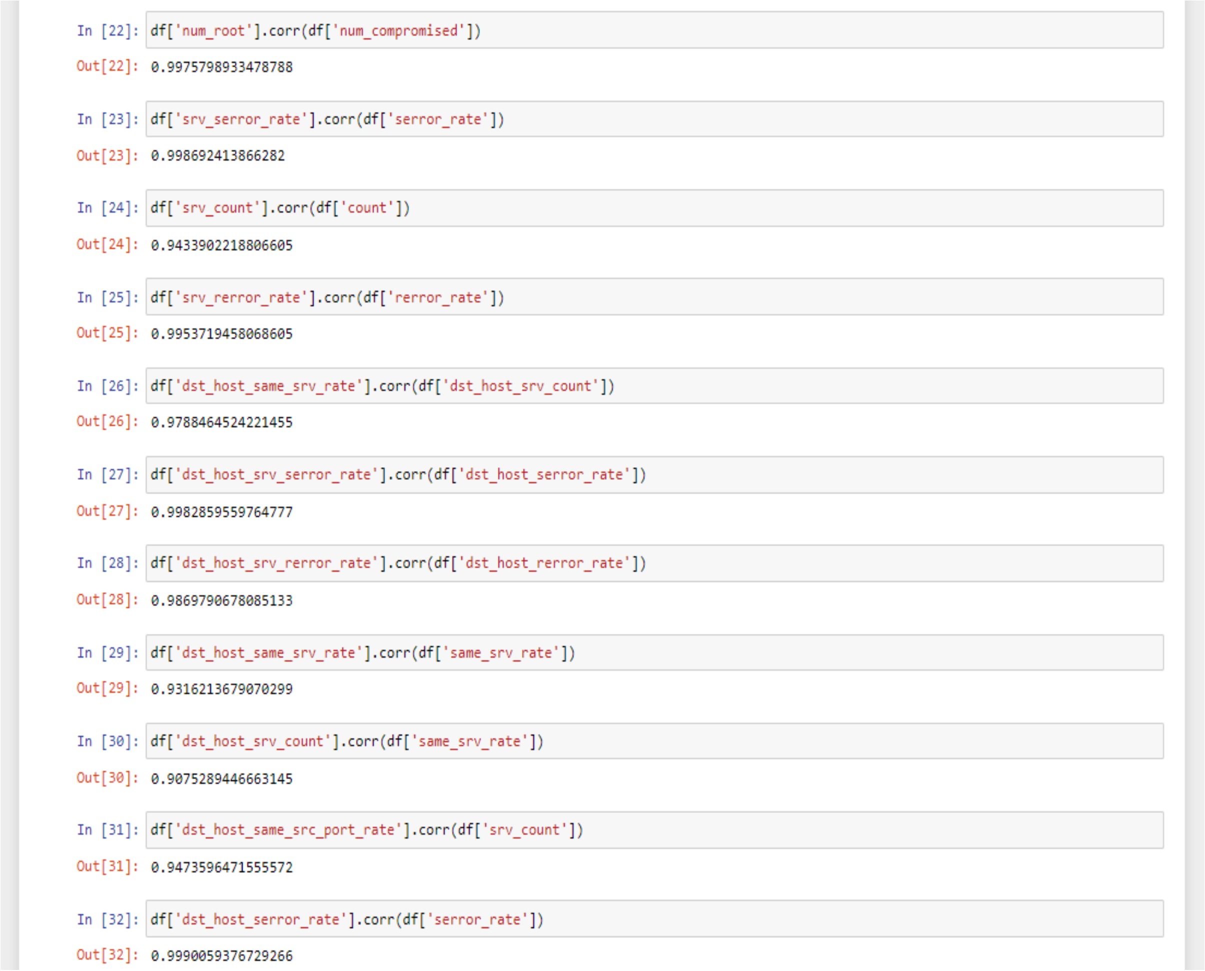
```
Out[20]: Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
                'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
                'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
                'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
                'num_access_files', 'num_outbound_cmds', 'is_host_login',
                'is_guest_login', 'count', 'srv_count', 'serror_rate',
                'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
                'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
                'dst_host_srv_count', 'dst_host_same_srv_rate',
                'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
                'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
                'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
                'dst_host_srv_rerror_rate', 'target', 'Attack Type'],
               dtype='object')
```



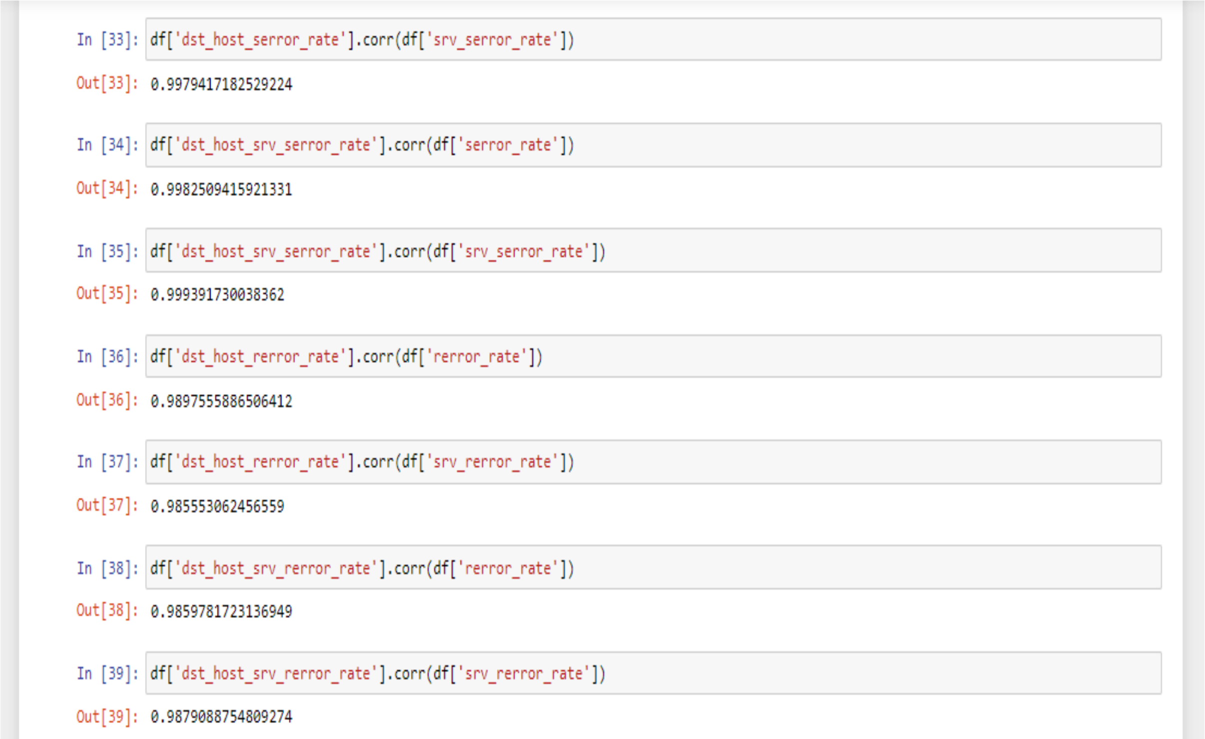
DATA CORRELATION

```
In [21]: df = df.dropna('columns')# drop columns with NaN

         df = df[[col for col in df if df[col].nunique() > 1]]# keep columns where there are more than 1 unique values

         corr = df.corr()

         plt.figure(figsize=(15,12))

         sns.heatmap(corr)

         plt.show()
```

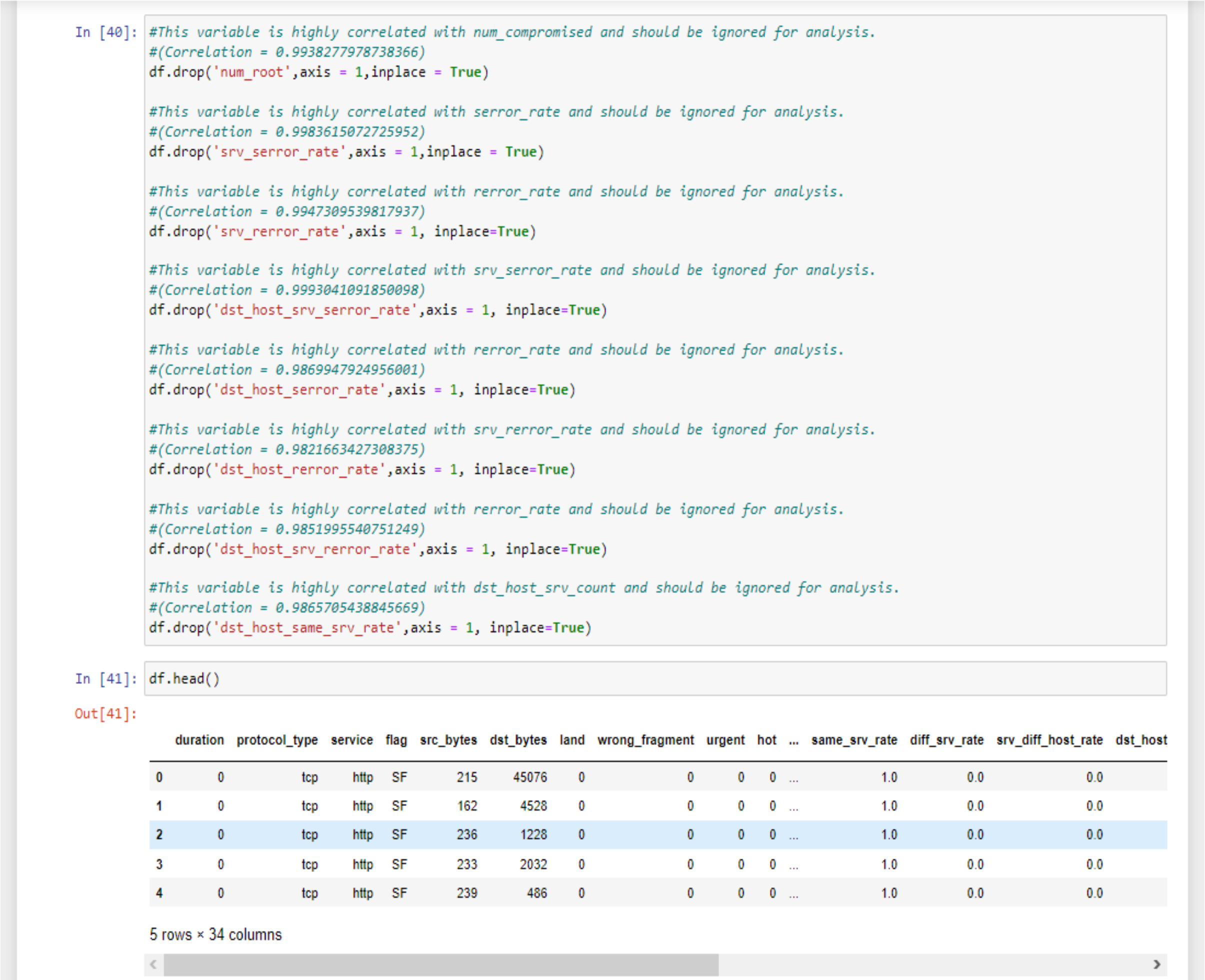
```
C:\Users\akhil\AppData\Local\Temp\ipykernel_33676\4162482413.py:1: FutureWarning: In a future version of pandas all arguments o
f DataFrame.dropna will be keyword-only.
  df = df.dropna('columns')# drop columns with NaN
```



```
In [22]: df['num_root'].corr(df['num_compromised'])

Out[22]: 0.9975798933478788

In [23]: df['srv_serror_rate'].corr(df['serror_rate'])

Out[23]: 0.998692413866282

In [24]: df['srv_count'].corr(df['count'])

Out[24]: 0.9433902218806605

In [25]: df['srv_rerror_rate'].corr(df['rerror_rate'])

Out[25]: 0.9953719458068605

In [26]: df['dst_host_same_srv_rate'].corr(df['dst_host_srv_count'])

Out[26]: 0.9788464524221455

In [27]: df['dst_host_srv_serror_rate'].corr(df['dst_host_serror_rate'])

Out[27]: 0.9982859559764777

In [28]: df['dst_host_srv_rerror_rate'].corr(df['dst_host_rerror_rate'])

Out[28]: 0.9869790678085133

In [29]: df['dst_host_same_srv_rate'].corr(df['same_srv_rate'])

Out[29]: 0.9316213679070299

In [30]: df['dst_host_srv_count'].corr(df['same_srv_rate'])

Out[30]: 0.9075289446663145

In [31]: df['dst_host_same_src_port_rate'].corr(df['srv_count'])

Out[31]: 0.9473596471555572

In [32]: df['dst_host_serror_rate'].corr(df['serror_rate'])

Out[32]: 0.9990059376729266
```



```
In [33]: df['dst_host_serror_rate'].corr(df['srv_serror_rate'])

Out[33]: 0.9979417182529224

In [34]: df['dst_host_srv_serror_rate'].corr(df['serror_rate'])

Out[34]: 0.9982509415921331

In [35]: df['dst_host_srv_serror_rate'].corr(df['srv_serror_rate'])

Out[35]: 0.999391730038362

In [36]: df['dst_host_rerror_rate'].corr(df['rerror_rate'])

Out[36]: 0.9897555886506412

In [37]: df['dst_host_rerror_rate'].corr(df['srv_rerror_rate'])

Out[37]: 0.985553062456559

In [38]: df['dst_host_srv_rerror_rate'].corr(df['rerror_rate'])

Out[38]: 0.9859781723136949

In [39]: df['dst_host_srv_rerror_rate'].corr(df['srv_rerror_rate'])

Out[39]: 0.9879088754809274
```



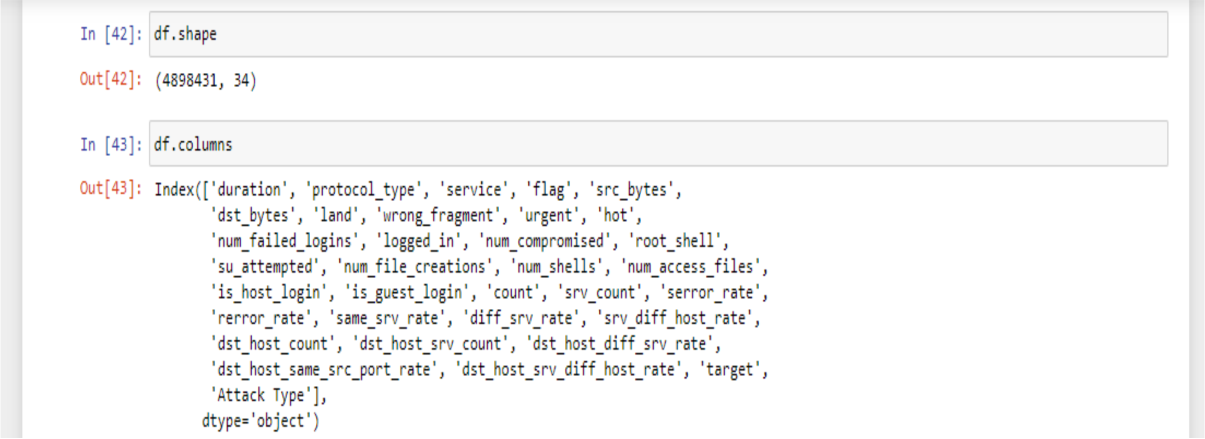
In [40]:
```python
#This variable is highly correlated with num_compromised and should be ignored for analysis.
#(Correlation = 0.9938277978738366)
df.drop('num_root',axis = 1,inplace = True)

#This variable is highly correlated with serror_rate and should be ignored for analysis.
#(Correlation = 0.9983615072725952)
df.drop('srv_serror_rate',axis = 1,inplace = True)

#This variable is highly correlated with rerror_rate and should be ignored for analysis.
#(Correlation = 0.9947309539817937)
df.drop('srv_rerror_rate',axis = 1, inplace=True)

#This variable is highly correlated with srv_serror_rate and should be ignored for analysis.
#(Correlation = 0.9993041091850098)
df.drop('dst_host_srv_serror_rate',axis = 1, inplace=True)

#This variable is highly correlated with rerror_rate and should be ignored for analysis.
#(Correlation = 0.9869947924956001)
df.drop('dst_host_serror_rate',axis = 1, inplace=True)

#This variable is highly correlated with srv_rerror_rate and should be ignored for analysis.
#(Correlation = 0.9821663427308375)
df.drop('dst_host_rerror_rate',axis = 1, inplace=True)

#This variable is highly correlated with rerror_rate and should be ignored for analysis.
#(Correlation = 0.9851995540751249)
df.drop('dst_host_srv_rerror_rate',axis = 1, inplace=True)

#This variable is highly correlated with dst_host_srv_count and should be ignored for analysis.
#(Correlation = 0.9865705438845669)
df.drop('dst_host_same_srv_rate',axis = 1, inplace=True)
```

In [41]:
```python
df.head()
```

Out[41]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	same_srv_rate	diff_srv_rate	srv_diff_host_rate	dst_host
0	0	tcp	http	SF	215	45076	0	0	0	0	...	1.0	0.0	0.0	
1	0	tcp	http	SF	162	4528	0	0	0	0	...	1.0	0.0	0.0	
2	0	tcp	http	SF	236	1228	0	0	0	0	...	1.0	0.0	0.0	
3	0	tcp	http	SF	233	2032	0	0	0	0	...	1.0	0.0	0.0	
4	0	tcp	http	SF	239	486	0	0	0	0	...	1.0	0.0	0.0	

5 rows × 34 columns

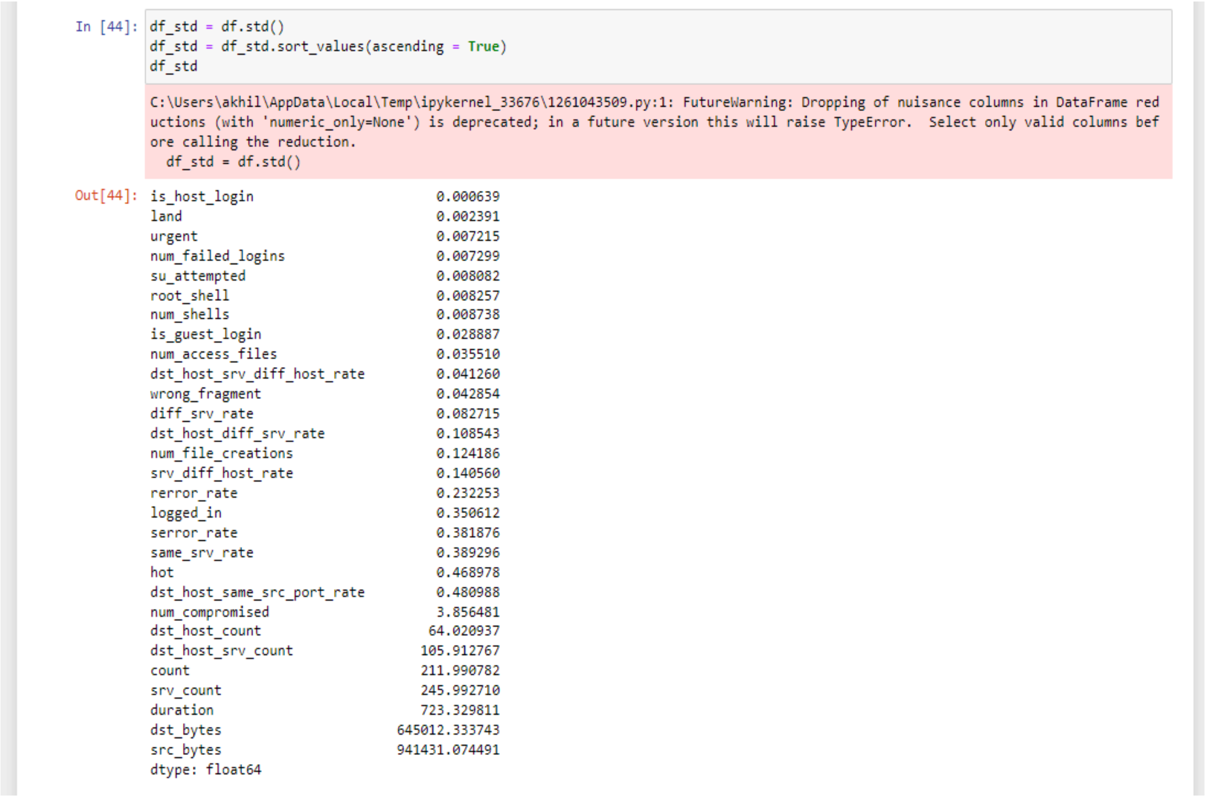


In [42]:
```python
df.shape
```

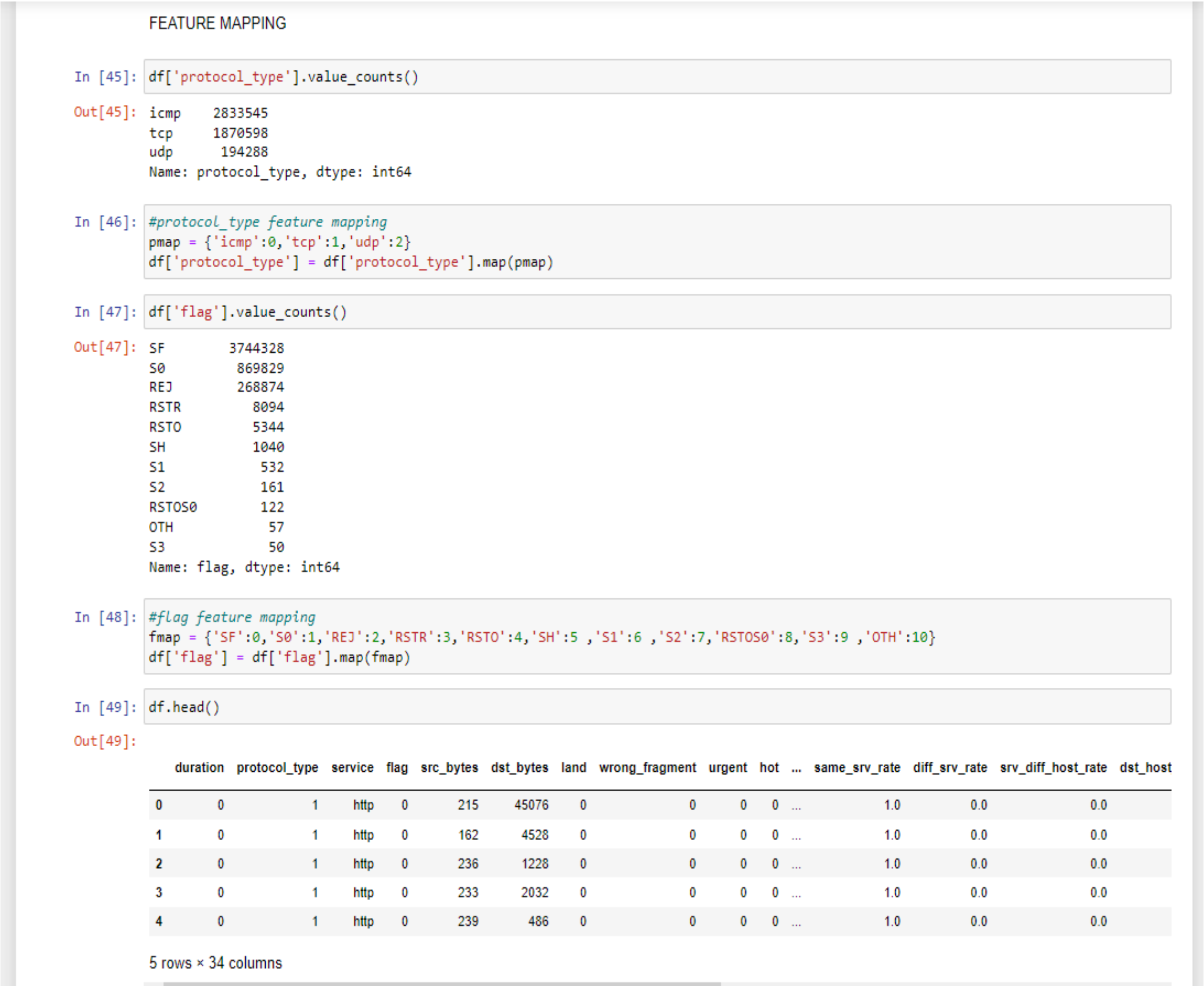
Out[42]:
```
(4898431, 34)
```

In [43]:
```python
df.columns
```

Out[43]:
```
Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
       'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
       'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
       'su_attempted', 'num_file_creations', 'num_shells', 'num_access_files',
       'is_host_login', 'is_guest_login', 'count', 'srv_count', 'serror_rate',
       'rerror_rate', 'same_srv_rate', 'diff_srv_rate', 'srv_diff_host_rate',
       'dst_host_count', 'dst_host_srv_count', 'dst_host_diff_srv_rate',
       'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'target',
       'Attack Type'],
      dtype='object')
```



```
In [44]: df_std = df.std()
         df_std = df_std.sort_values(ascending = True)
         df_std

C:\Users\akhil\AppData\Local\Temp\ipykernel_33676\1261043509.py:1: FutureWarning: Dropping of nuisance columns in DataFrame red
uctions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns bef
ore calling the reduction.
  df_std = df.std()

Out[44]: is_host_login                   0.000639
         land                            0.002391
         urgent                          0.007215
         num_failed_logins               0.007299
         su_attempted                    0.008082
         root_shell                      0.008257
         num_shells                      0.008738
         is_guest_login                  0.028887
         num_access_files                0.035510
         dst_host_srv_diff_host_rate     0.041260
         wrong_fragment                  0.042854
         diff_srv_rate                   0.082715
         dst_host_diff_srv_rate          0.108543
         num_file_creations              0.124186
         srv_diff_host_rate              0.140560
         rerror_rate                     0.232253
         logged_in                       0.350612
         serror_rate                     0.381876
         same_srv_rate                   0.389296
         hot                             0.468978
         dst_host_same_src_port_rate     0.480988
         num_compromised                 3.856481
         dst_host_count                 64.020937
         dst_host_srv_count            105.912767
         count                         211.990782
         srv_count                     245.992710
         duration                      723.329811
         dst_bytes                  645012.333743
         src_bytes                  941431.074491
         dtype: float64
```

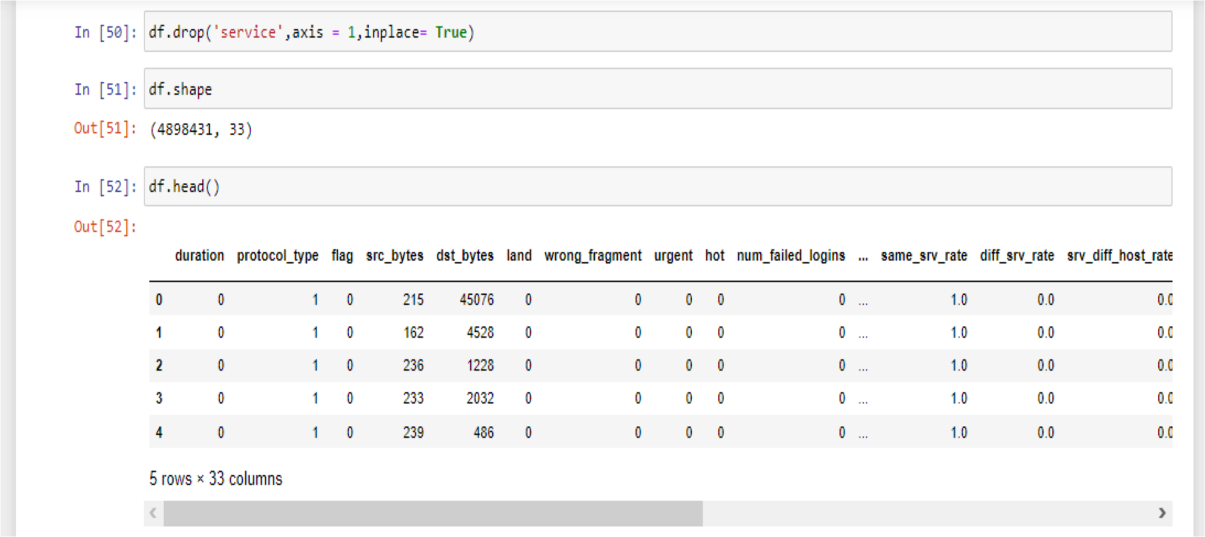


FEATURE MAPPING

```
In [45]: df['protocol_type'].value_counts()

Out[45]: icmp    2833545
         tcp     1870598
         udp      194288
         Name: protocol_type, dtype: int64

In [46]: #protocol_type feature mapping
         pmap = {'icmp':0,'tcp':1,'udp':2}
         df['protocol_type'] = df['protocol_type'].map(pmap)

In [47]: df['flag'].value_counts()

Out[47]: SF        3744328
         S0         869829
         REJ        268874
         RSTR         8094
         RSTO         5344
         SH           1040
         S1            532
         S2            161
         RSTOS0        122
         OTH            57
         S3             50
         Name: flag, dtype: int64

In [48]: #flag feature mapping
         fmap = {'SF':0,'S0':1,'REJ':2,'RSTR':3,'RSTO':4,'SH':5 ,'S1':6 ,'S2':7,'RSTOS0':8,'S3':9 ,'OTH':10}
         df['flag'] = df['flag'].map(fmap)

In [49]: df.head()

Out[49]:
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	same_srv_rate	diff_srv_rate	srv_diff_host_rate	dst_host
0	0	1	http	0	215	45076	0	0	0	0	...	1.0	0.0	0.0	
1	0	1	http	0	162	4528	0	0	0	0	...	1.0	0.0	0.0	
2	0	1	http	0	236	1228	0	0	0	0	...	1.0	0.0	0.0	
3	0	1	http	0	233	2032	0	0	0	0	...	1.0	0.0	0.0	
4	0	1	http	0	239	486	0	0	0	0	...	1.0	0.0	0.0	

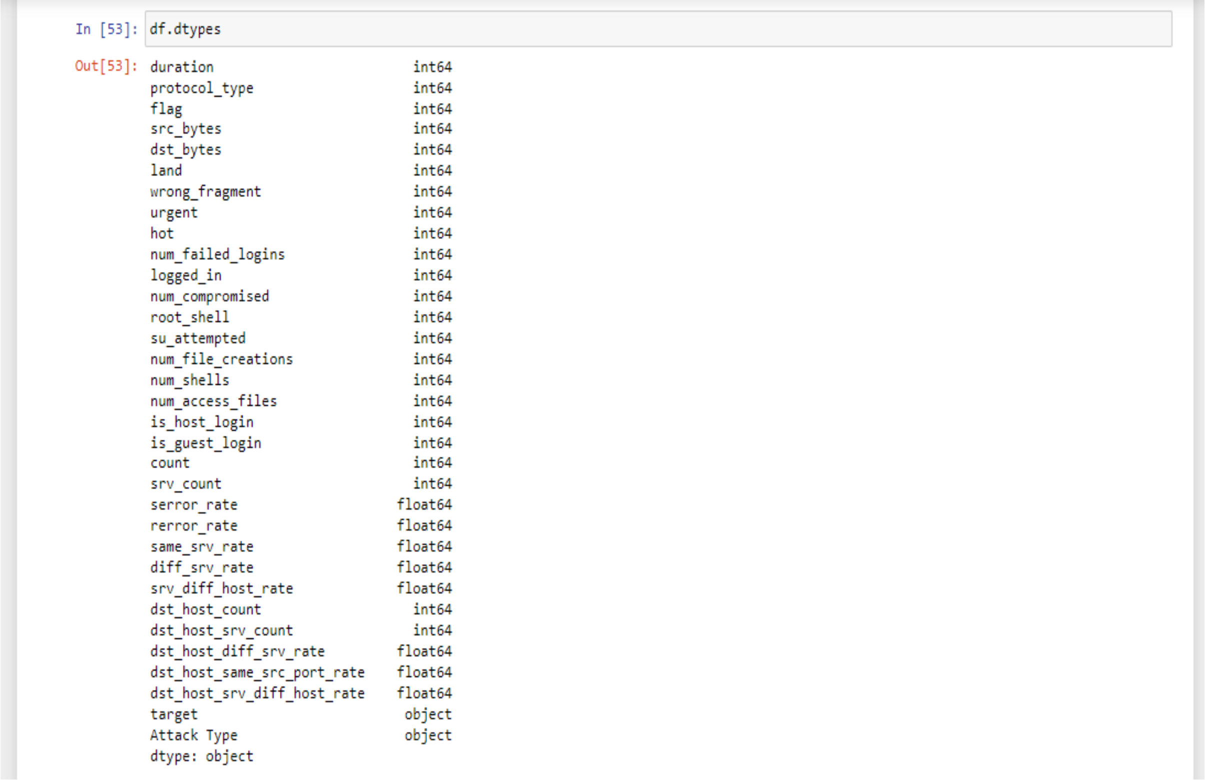
5 rows × 34 columns



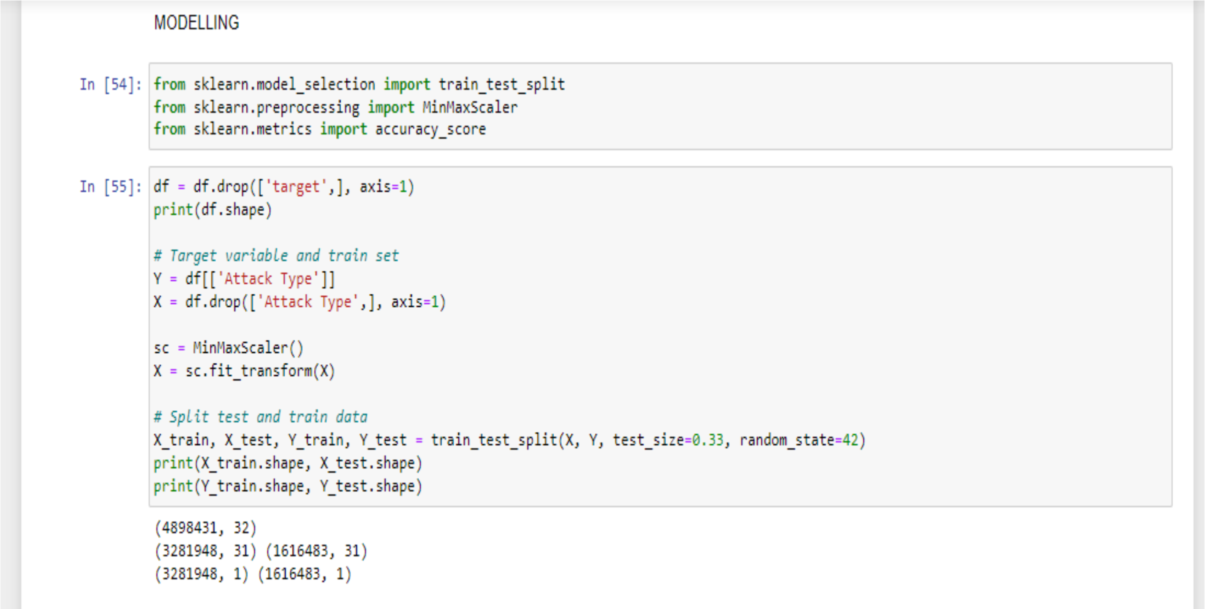
```
In [50]: df.drop('service',axis = 1,inplace= True)

In [51]: df.shape

Out[51]: (4898431, 33)

In [52]: df.head()

Out[52]:
```

	duration	protocol_type	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	...	same_srv_rate	diff_srv_rate	srv_diff_host_rate
0	0	1	0	215	45076	0	0	0	0	0	...	1.0	0.0	0.0
1	0	1	0	162	4528	0	0	0	0	0	...	1.0	0.0	0.0
2	0	1	0	236	1228	0	0	0	0	0	...	1.0	0.0	0.0
3	0	1	0	233	2032	0	0	0	0	0	...	1.0	0.0	0.0
4	0	1	0	239	486	0	0	0	0	0	...	1.0	0.0	0.0

5 rows × 33 columns

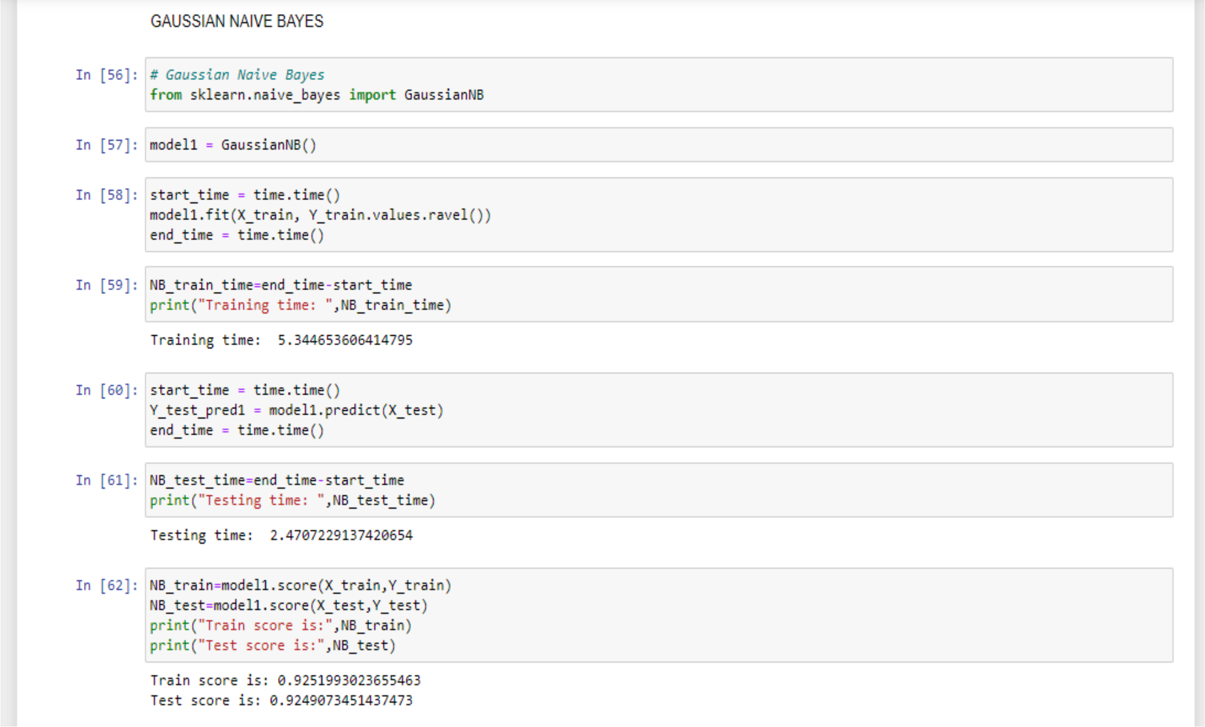


```
In [53]: df.dtypes

Out[53]: duration                         int64
         protocol_type                    int64
         flag                             int64
         src_bytes                        int64
         dst_bytes                        int64
         land                             int64
         wrong_fragment                   int64
         urgent                           int64
         hot                              int64
         num_failed_logins                int64
         logged_in                        int64
         num_compromised                  int64
         root_shell                       int64
         su_attempted                     int64
         num_file_creations               int64
         num_shells                       int64
         num_access_files                 int64
         is_host_login                    int64
         is_guest_login                   int64
         count                            int64
         srv_count                        int64
         serror_rate                    float64
         rerror_rate                    float64
         same_srv_rate                  float64
         diff_srv_rate                  float64
         srv_diff_host_rate             float64
         dst_host_count                   int64
         dst_host_srv_count               int64
         dst_host_diff_srv_rate         float64
         dst_host_same_src_port_rate    float64
         dst_host_srv_diff_host_rate    float64
         target                          object
         Attack Type                     object
         dtype: object
```



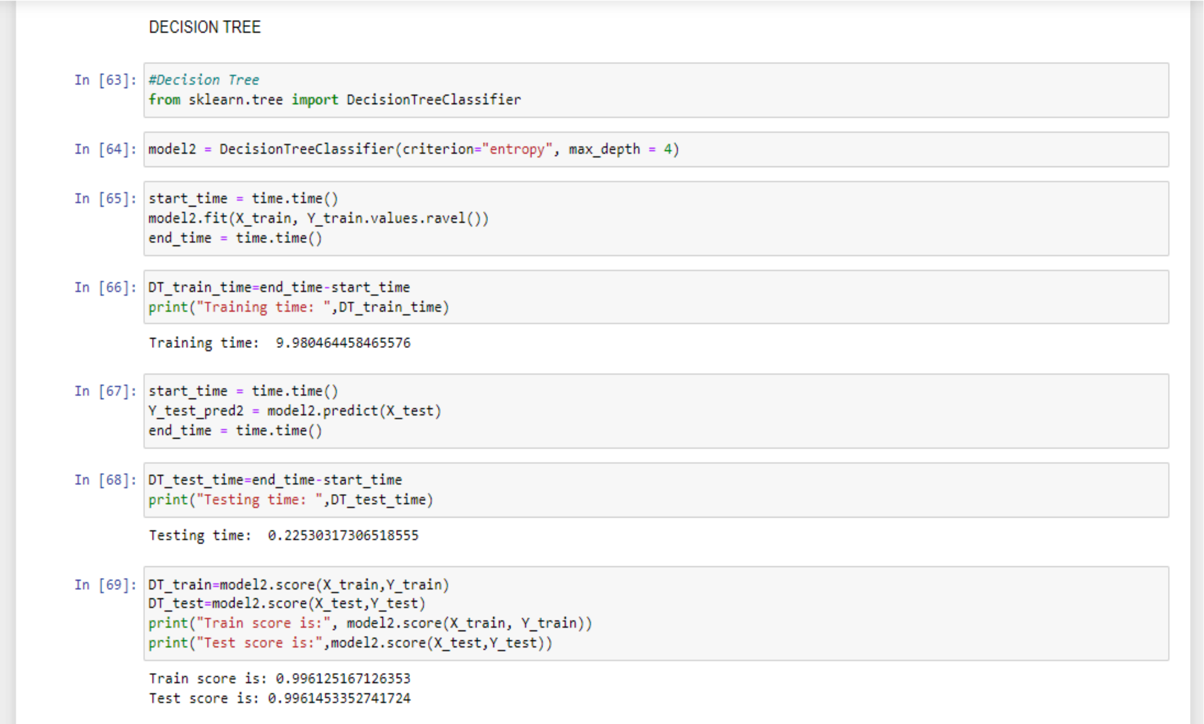
MODELLING

```
In [54]: from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import MinMaxScaler
         from sklearn.metrics import accuracy_score

In [55]: df = df.drop(['target',], axis=1)
         print(df.shape)

         # Target variable and train set
         Y = df[['Attack Type']]
         X = df.drop(['Attack Type',], axis=1)

         sc = MinMaxScaler()
         X = sc.fit_transform(X)

         # Split test and train data
         X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_state=42)
         print(X_train.shape, X_test.shape)
         print(Y_train.shape, Y_test.shape)

         (4898431, 32)
         (3281948, 31) (1616483, 31)
         (3281948, 1) (1616483, 1)
```



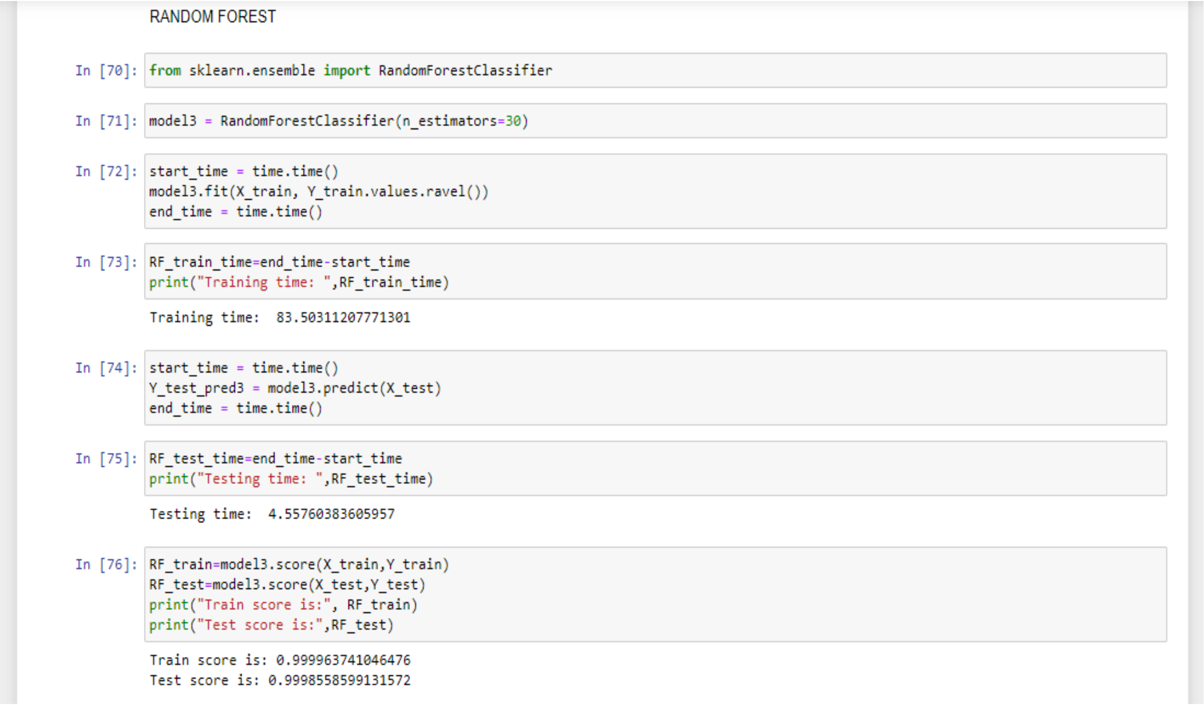
GAUSSIAN NAIVE BAYES

```
In [56]: # Gaussian Naive Bayes
         from sklearn.naive_bayes import GaussianNB

In [57]: model1 = GaussianNB()

In [58]: start_time = time.time()
         model1.fit(X_train, Y_train.values.ravel())
         end_time = time.time()

In [59]: NB_train_time=end_time-start_time
         print("Training time: ",NB_train_time)

         Training time:  5.344653606414795

In [60]: start_time = time.time()
         Y_test_pred1 = model1.predict(X_test)
         end_time = time.time()

In [61]: NB_test_time=end_time-start_time
         print("Testing time: ",NB_test_time)

         Testing time:  2.4707229137420654

In [62]: NB_train=model1.score(X_train,Y_train)
         NB_test=model1.score(X_test,Y_test)
         print("Train score is:",NB_train)
         print("Test score is:",NB_test)

         Train score is: 0.9251993023655463
         Test score is: 0.9249073451437473
```



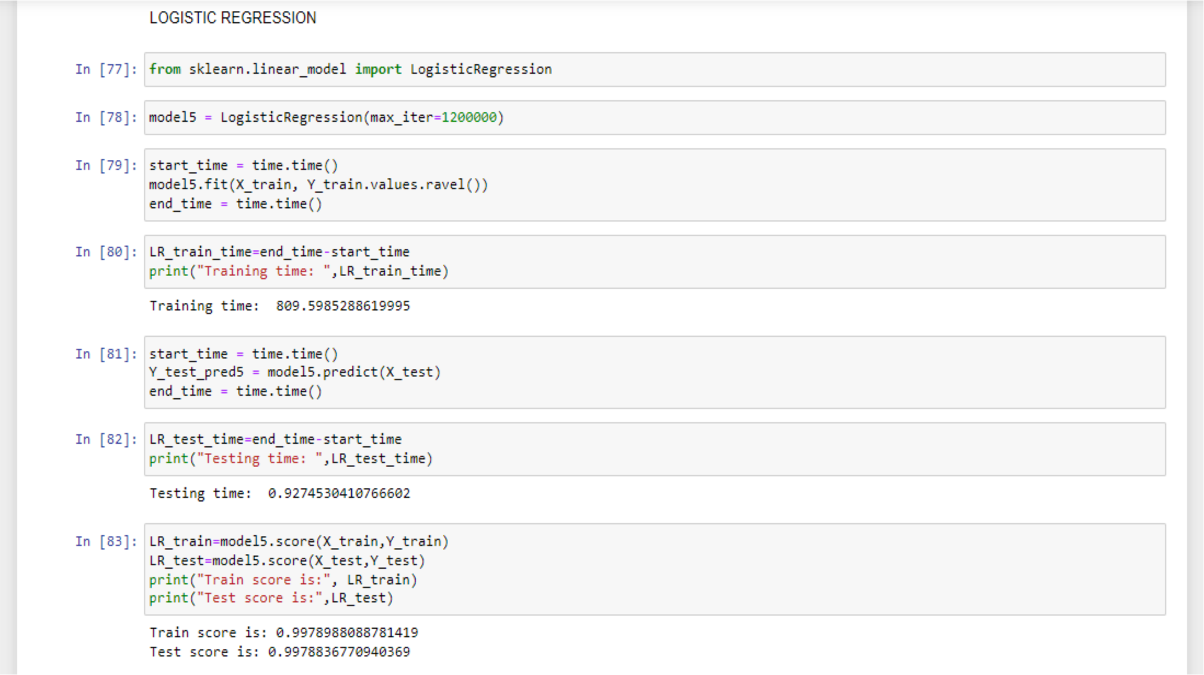
DECISION TREE

```
In [63]: #Decision Tree
         from sklearn.tree import DecisionTreeClassifier

In [64]: model2 = DecisionTreeClassifier(criterion="entropy", max_depth = 4)

In [65]: start_time = time.time()
         model2.fit(X_train, Y_train.values.ravel())
         end_time = time.time()

In [66]: DT_train_time=end_time-start_time
         print("Training time: ",DT_train_time)

         Training time:  9.980464458465576

In [67]: start_time = time.time()
         Y_test_pred2 = model2.predict(X_test)
         end_time = time.time()

In [68]: DT_test_time=end_time-start_time
         print("Testing time: ",DT_test_time)

         Testing time:  0.22530317306518555

In [69]: DT_train=model2.score(X_train,Y_train)
         DT_test=model2.score(X_test,Y_test)
         print("Train score is:", model2.score(X_train, Y_train))
         print("Test score is:",model2.score(X_test,Y_test))

         Train score is: 0.996125167126353
         Test score is: 0.9961453352741724
```



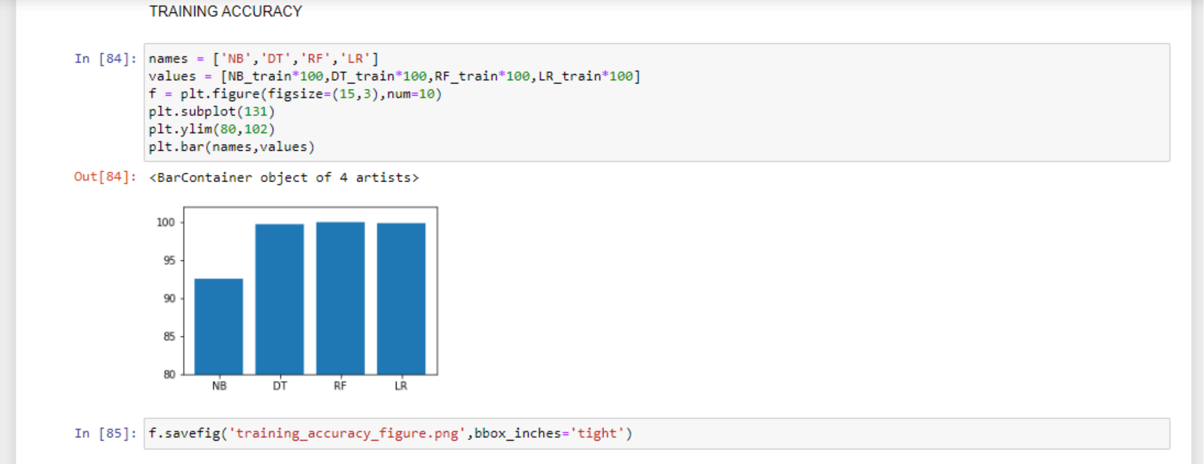
RANDOM FOREST

```
In [70]: from sklearn.ensemble import RandomForestClassifier

In [71]: model3 = RandomForestClassifier(n_estimators=30)

In [72]: start_time = time.time()
         model3.fit(X_train, Y_train.values.ravel())
         end_time = time.time()

In [73]: RF_train_time=end_time-start_time
         print("Training time: ",RF_train_time)

         Training time:  83.50311207771301

In [74]: start_time = time.time()
         Y_test_pred3 = model3.predict(X_test)
         end_time = time.time()

In [75]: RF_test_time=end_time-start_time
         print("Testing time: ",RF_test_time)

         Testing time:  4.55760383605957

In [76]: RF_train=model3.score(X_train,Y_train)
         RF_test=model3.score(X_test,Y_test)
         print("Train score is:", RF_train)
         print("Test score is:",RF_test)

         Train score is: 0.999963741046476
         Test score is: 0.9998558599131572
```



LOGISTIC REGRESSION

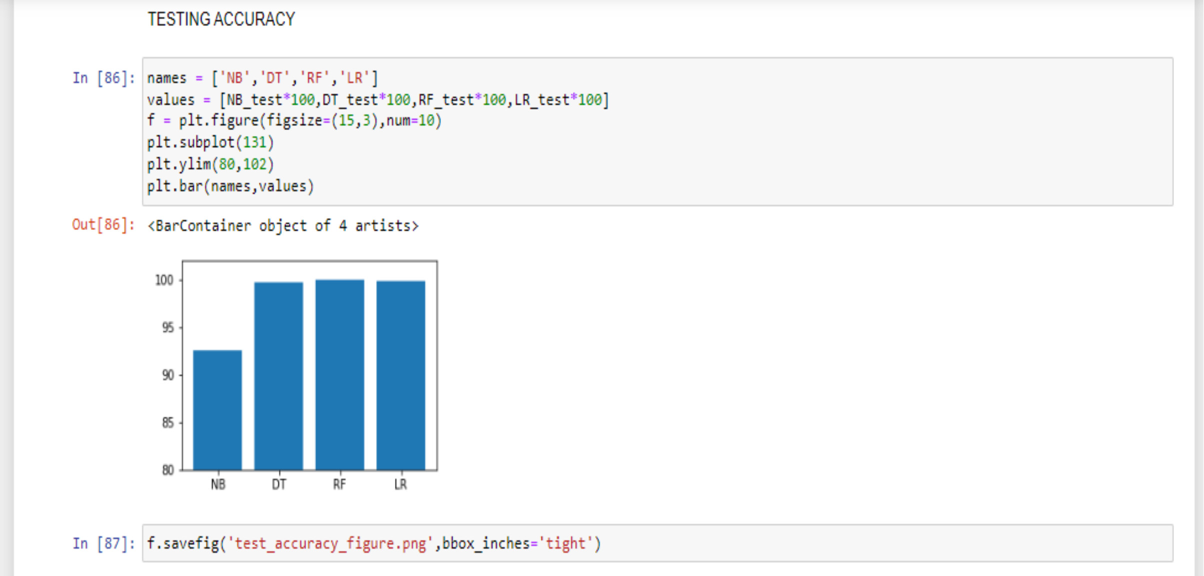
```
In [77]: from sklearn.linear_model import LogisticRegression

In [78]: model5 = LogisticRegression(max_iter=1200000)

In [79]: start_time = time.time()
         model5.fit(X_train, Y_train.values.ravel())
         end_time = time.time()

In [80]: LR_train_time=end_time-start_time
         print("Training time: ",LR_train_time)

         Training time:  809.5985288619995

In [81]: start_time = time.time()
         Y_test_pred5 = model5.predict(X_test)
         end_time = time.time()

In [82]: LR_test_time=end_time-start_time
         print("Testing time: ",LR_test_time)

         Testing time:  0.9274530410766602

In [83]: LR_train=model5.score(X_train,Y_train)
         LR_test=model5.score(X_test,Y_test)
         print("Train score is:", LR_train)
         print("Test score is:",LR_test)

         Train score is: 0.9978988088781419
         Test score is: 0.9978836770940369
```



TRAINING ACCURACY

```
In [84]: names = ['NB','DT','RF','LR']
         values = [NB_train*100,DT_train*100,RF_train*100,LR_train*100]
         f = plt.figure(figsize=(15,3),num=10)
         plt.subplot(131)
         plt.ylim(80,102)
         plt.bar(names,values)

Out[84]: <BarContainer object of 4 artists>
```

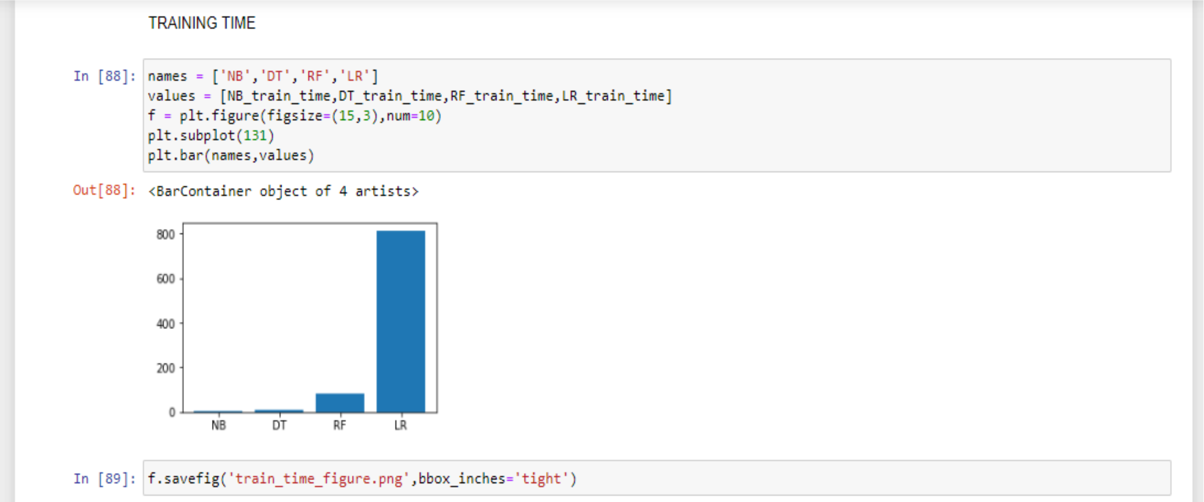
```
In [85]: f.savefig('training_accuracy_figure.png',bbox_inches='tight')
```



TESTING ACCURACY

```
In [86]: names = ['NB','DT','RF','LR']
         values = [NB_test*100,DT_test*100,RF_test*100,LR_test*100]
         f = plt.figure(figsize=(15,3),num=10)
         plt.subplot(131)
         plt.ylim(80,102)
         plt.bar(names,values)

Out[86]: <BarContainer object of 4 artists>
```

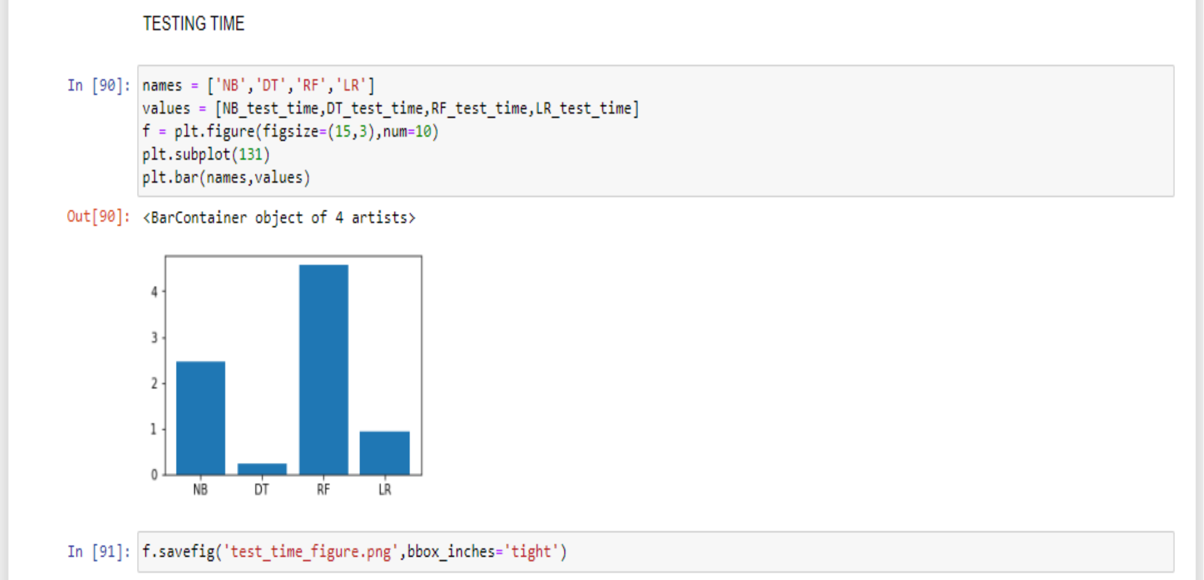
```
In [87]: f.savefig('test_accuracy_figure.png',bbox_inches='tight')
```



TRAINING TIME

```
In [88]: names = ['NB','DT','RF','LR']
         values = [NB_train_time,DT_train_time,RF_train_time,LR_train_time]
         f = plt.figure(figsize=(15,3),num=10)
         plt.subplot(131)
         plt.bar(names,values)

Out[88]: <BarContainer object of 4 artists>
```

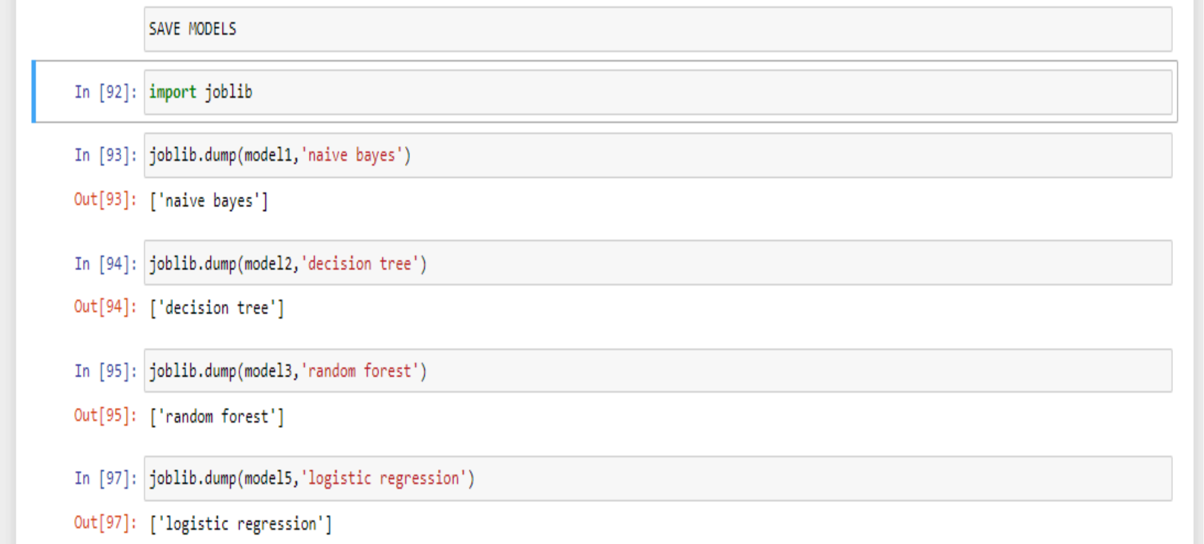
```
In [89]: f.savefig('train_time_figure.png',bbox_inches='tight')
```



TESTING TIME

```
In [90]: names = ['NB','DT','RF','LR']
         values = [NB_test_time,DT_test_time,RF_test_time,LR_test_time]
         f = plt.figure(figsize=(15,3),num=10)
         plt.subplot(131)
         plt.bar(names,values)

Out[90]: <BarContainer object of 4 artists>
```

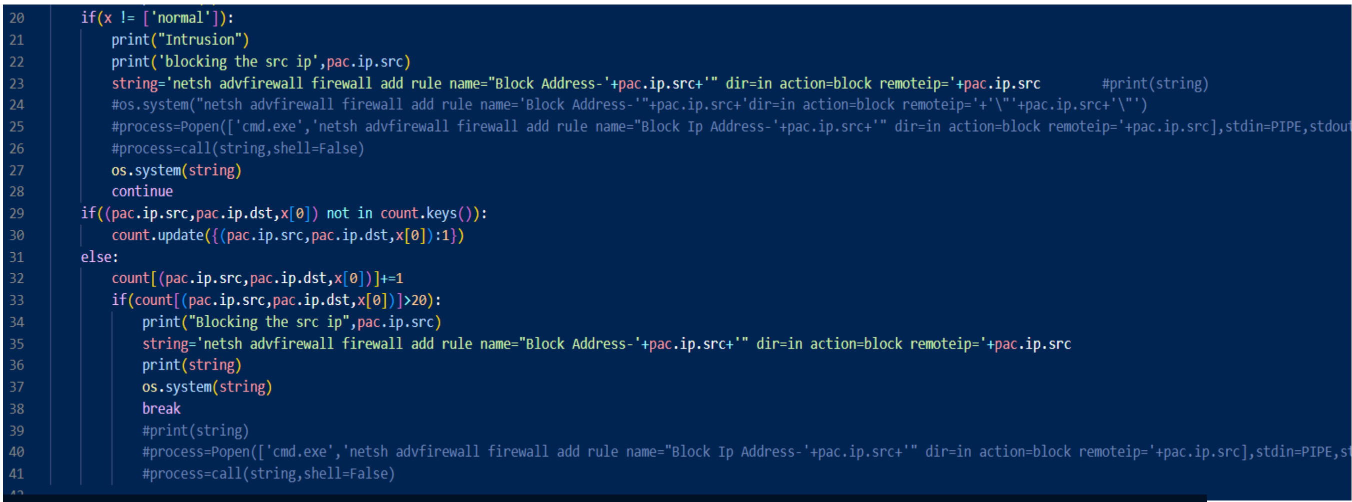
```
In [91]: f.savefig('test_time_figure.png',bbox_inches='tight')
```



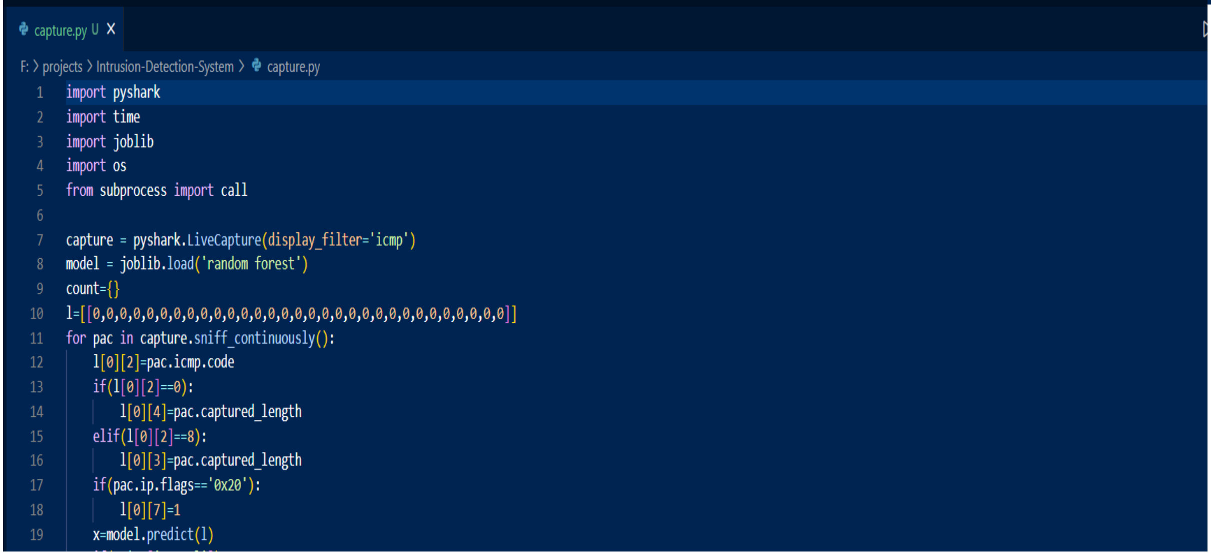
SAVE MODELS

```
In [92]: import joblib

In [93]: joblib.dump(model1,'naive bayes')

Out[93]: ['naive bayes']

In [94]: joblib.dump(model2,'decision tree')

Out[94]: ['decision tree']

In [95]: joblib.dump(model3,'random forest')

Out[95]: ['random forest']

In [97]: joblib.dump(model5,'logistic regression')

Out[97]: ['logistic regression']
```

6.2 MODEL DEPLOYMENT CODE



```
    if(x != ['normal']):
        print("Intrusion")
        print('blocking the src ip',pac.ip.src)
        string='netsh advfirewall firewall add rule name="Block Address-'+pac.ip.src+'" dir=in action=block remoteip='+pac.ip.src        #print(string)
        #os.system("netsh advfirewall firewall add rule name='Block Address-'"+pac.ip.src+'dir=in action=block remoteip='+'\"'+pac.ip.src+'\"')
        #process=Popen(['cmd.exe','netsh advfirewall firewall add rule name="Block Ip Address-'+pac.ip.src+'" dir=in action=block remoteip='+pac.ip.src],stdin=PIPE,stdout
        #process=call(string,shell=False)
        os.system(string)
        continue
    if((pac.ip.src,pac.ip.dst,x[0]) not in count.keys()):
        count.update({(pac.ip.src,pac.ip.dst,x[0]):1})
    else:
        count[(pac.ip.src,pac.ip.dst,x[0])]+=1
        if(count[(pac.ip.src,pac.ip.dst,x[0])]>20):
            print("Blocking the src ip",pac.ip.src)
            string='netsh advfirewall firewall add rule name="Block Address-'+pac.ip.src+'" dir=in action=block remoteip='+pac.ip.src
            print(string)
            os.system(string)
            break
            #print(string)
            #process=Popen(['cmd.exe','netsh advfirewall firewall add rule name="Block Ip Address-'+pac.ip.src+'" dir=in action=block remoteip='+pac.ip.src],stdin=PIPE,st
            #process=call(string,shell=False)
```



```
import pyshark
import time
import joblib
import os
from subprocess import call

capture = pyshark.LiveCapture(display_filter='icmp')
model = joblib.load('random forest')
count={}
l=[[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]]
for pac in capture.sniff_continuously():
    l[0][2]=pac.icmp.code
    if(l[0][2]==0):
        l[0][4]=pac.captured_length
    elif(l[0][2]==8):
        l[0][3]=pac.captured_length
    if(pac.ip.flags=='0x20'):
        l[0][7]=1
    x=model.predict(l)
```

7. SYSTEM TESTING

7.1 Testing

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce the quality product. In simple terms, Software Testing means Verification of Application Under Test (AUT).

There are two types of software testing:

1. Manual Testing
2. Automation Testing

7.1.1 Manual Testing

Manual testing is the process of testing software by hand to learn more about it, to find what is and isn't working. This usually includes verifying all the features specified in requirements documents, but often also includes the testers trying the software with the perspective of their end users in mind. Manual test plans vary from fully scripted test cases, giving testers detailed steps and expected results, through to high-level guides that steer exploratory testing sessions. There are lots of sophisticated tools on the market to help with manual testing, but if you want a simple and flexible place to start, take a look at testpad.com.

7.1.2 Automation Testing

Automation testing is the process of testing the software using an automation tool to find the defects. In this process, testers execute the test scripts and generate the test results automatically by using automation tools. Some of the famous automation testing tools for functional testing are QTP/UFT and selenium.

7.2 Testing Methods

There are two types of testing methods:

1. Static Testing
2. Dynamic Testing

7.2.1 Static Testing

It is also known as Verification in Software Testing. Verification is a static method of checking documents and files. Verification is the process, to ensure that whether we are building the product right i.e., to verify the requirements which we have and to verify whether we are developing the product accordingly or not.

7.2.2 Dynamic Testing

It is also known as Validation in Software Testing. Validation is a dynamic process of testing the real product. Validation is the process, whether we are building the right product i.e., to validate the product which we have developed is right or not.

7.3 Testing Approaches

There are three types of testing approaches:

1.Black Box Testing
2.White Box Testing
3.Grey Box Testing

7.3.1 Black Box Testing

It is also called as Behavioural/Specification-Based/Input-Output Testing. **BLACK BOX TESTING** is defined as a testing technique in which functionality of the Application Under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications. In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

Input
Black Box
Output

Fig. 7.3.1

7.3.2 White Box Testing

WHITE BOX TESTING is testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. It is usually performed by developers.

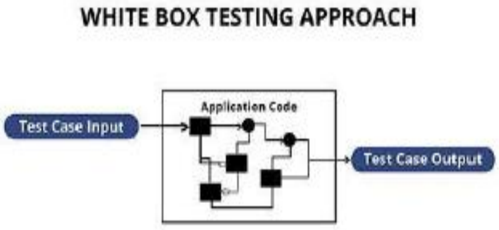


Fig. 7.3.2

7.3.3 Grey Box Testing

Grey box is the combination of both White Box and Black Box Testing. The tester who works on this type of testing needs to have access to design documents. This helps to create better test cases in this process.

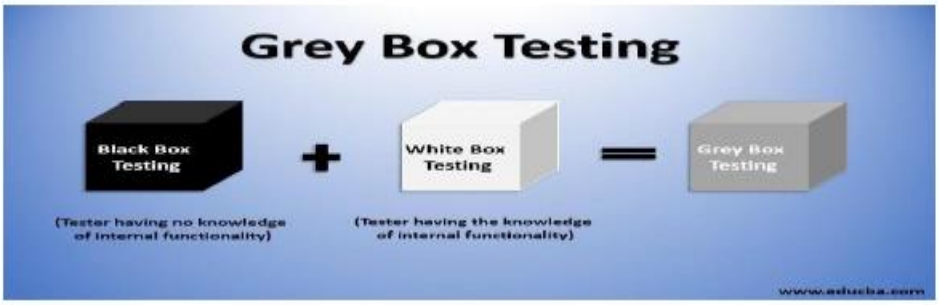

Grey Box Testing
Black Box Testing
(Tester having no knowledge of internal functionality)
+
White Box Testing
(Tester having the knowledge of internal functionality)
=
Grey Box Testing
www.educba.com

Fig. 7.3.3

7.4 Types of Testing

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing

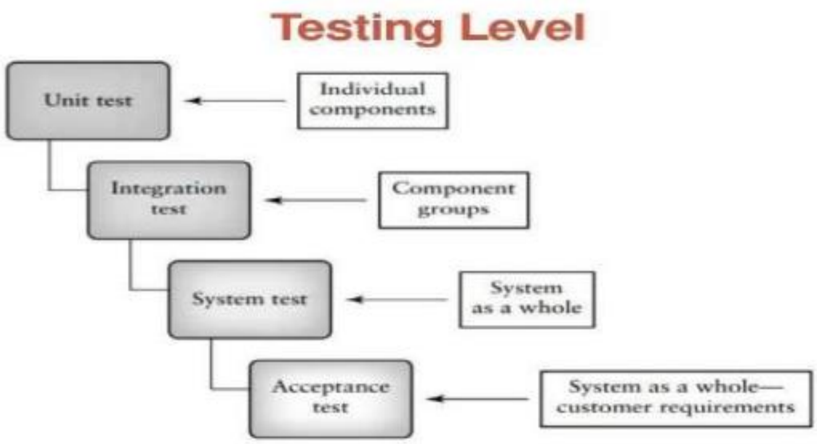


Fig. 7.4

7.4.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches an internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic test at component level and test a specific business process, application, and system configuration. Unit test ensures that each unit path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.4.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concern with the basic outcome of screens of fields. Integration test demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.4.3 System Testing

System testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

7.4.4 Acceptance Testing

Acceptance testing is a test conducted to find if the requirements of a specification or contract are met as per its delivery. Acceptance testing is basically done by the user or customer. However, other stockholders can be involved in this process.

7.5 Test Procedures

Specific knowledge of the application's code, internal structure and programming knowledge in general is not required. The tester is aware of *what* the software is supposed to do but is not aware of *how* it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of *how* the software produces the output in the first place.

7.6 Test Cases

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test Strategy and Approach

Field testing will be performed manually and functional tests will be written in detail.

Test Objectives

1. All field entries must work properly.
2. Pages must be activated from the identical link.
3. The entry screen, messages and responses must not be delayed.

Features to be Tested

1. Verify that the entries are of the correct format.
2. Verify that the entries are of within the range.

7.7 Test Results

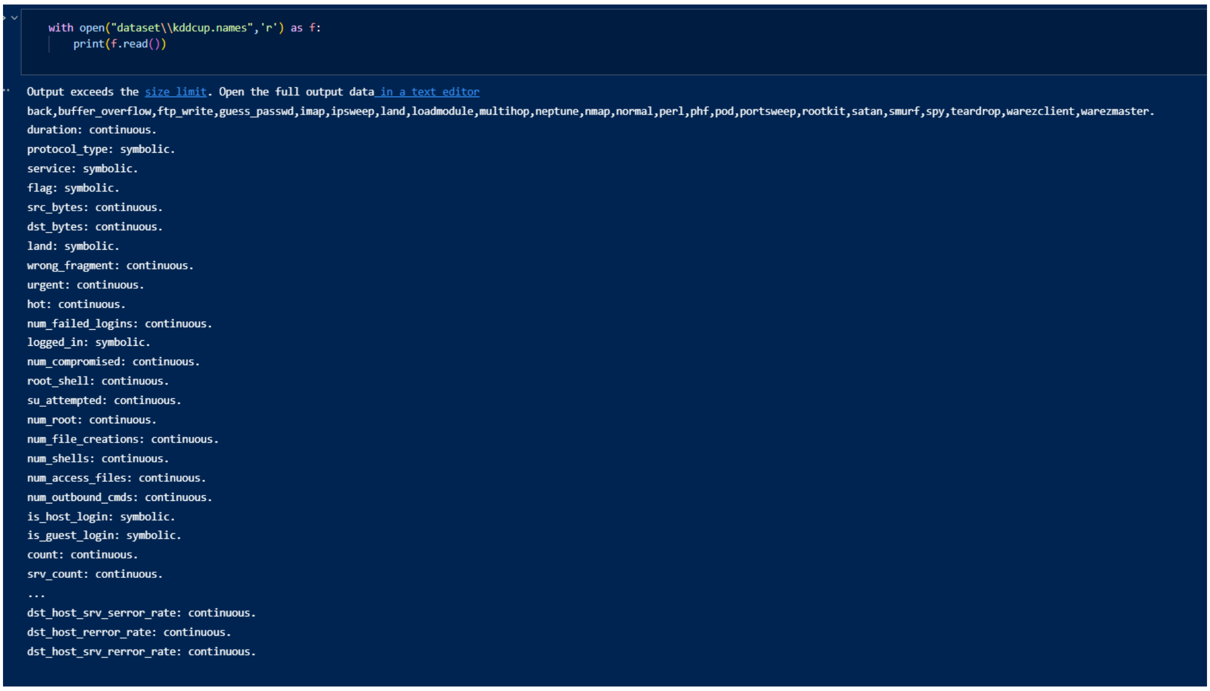
Unit Test Case 1

S. No of the Test Case	1
Name of the Test	Check if the attack is detected or not
Sample Input	Duration: 0 Protocol Type: 0

	Flag:8 Src_bytes: 305 Dst_bytes: 0 land: 4545 Wrong_fragment:0 urgent:0 Hot:0 Num_failed_logins:0 Logged_in:0 num_compromised:0 Root_shell:0 Su_attempted:0 Num_file_creations:0 Num_shells:0 Num_access_files:0 Is_host_login:0 Is_guest_login:0 Count:0 Srv_count:0 Serror_rate:0 Same_srv_rate:0
Expected Output	Detected
Actual Output	Detected
Remarks	Pass

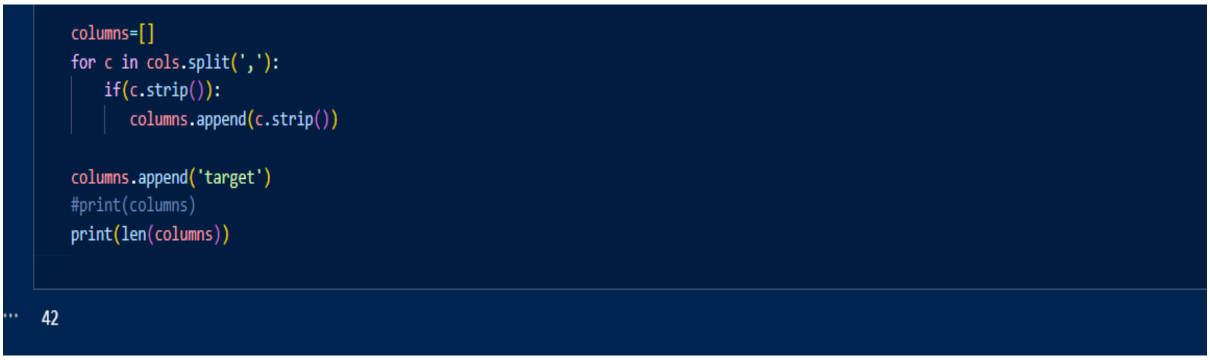
8. EXPERIMENTAL RESULTS

8.1 Output Screens



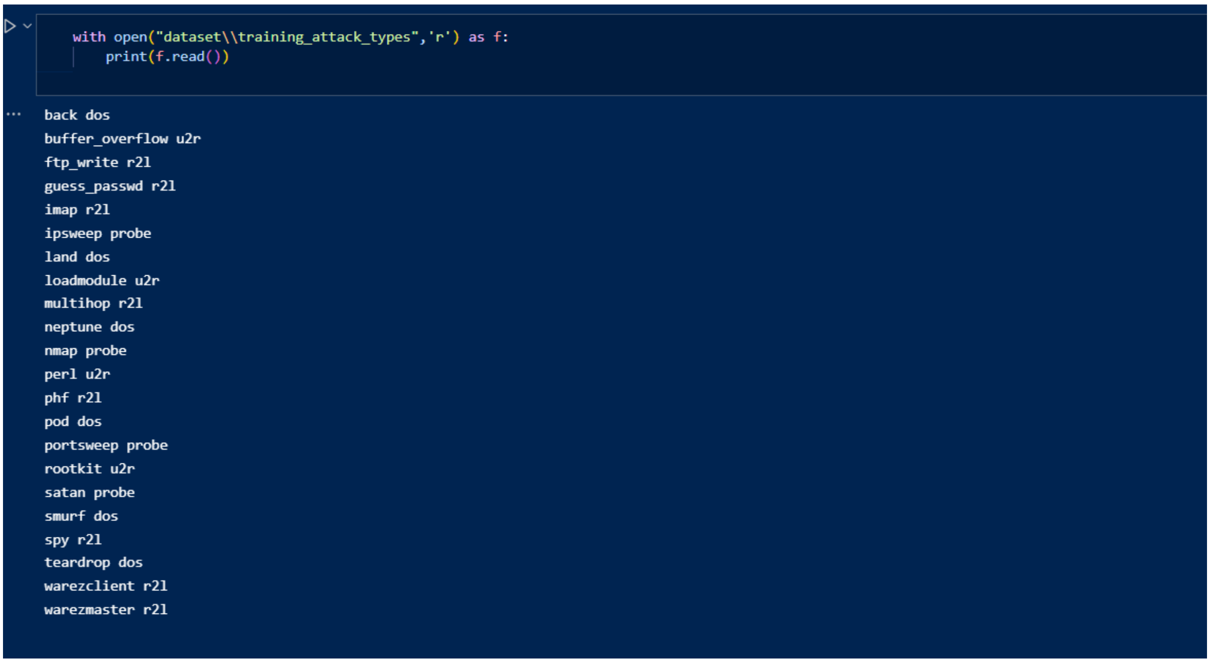
```
with open("dataset\\kddcup.names",'r') as f:
    print(f.read())
```

```
Output exceeds the size limit. Open the full output data in a text editor
back,buffer_overflow,ftp_write,guess_passwd,imap,ipsweep,land,loadmodule,multihop,neptune,nmap,normal,perl,phf,pod,portsweep,rootkit,satan,smurf,spy,teardrop,warezclient,warezmaster.
duration: continuous.
protocol_type: symbolic.
service: symbolic.
flag: symbolic.
src_bytes: continuous.
dst_bytes: continuous.
land: symbolic.
wrong_fragment: continuous.
urgent: continuous.
hot: continuous.
num_failed_logins: continuous.
logged_in: symbolic.
num_compromised: continuous.
root_shell: continuous.
su_attempted: continuous.
num_root: continuous.
num_file_creations: continuous.
num_shells: continuous.
num_access_files: continuous.
num_outbound_cmds: continuous.
is_host_login: symbolic.
is_guest_login: symbolic.
count: continuous.
srv_count: continuous.
...
dst_host_srv_serror_rate: continuous.
dst_host_rerror_rate: continuous.
dst_host_srv_rerror_rate: continuous.
```



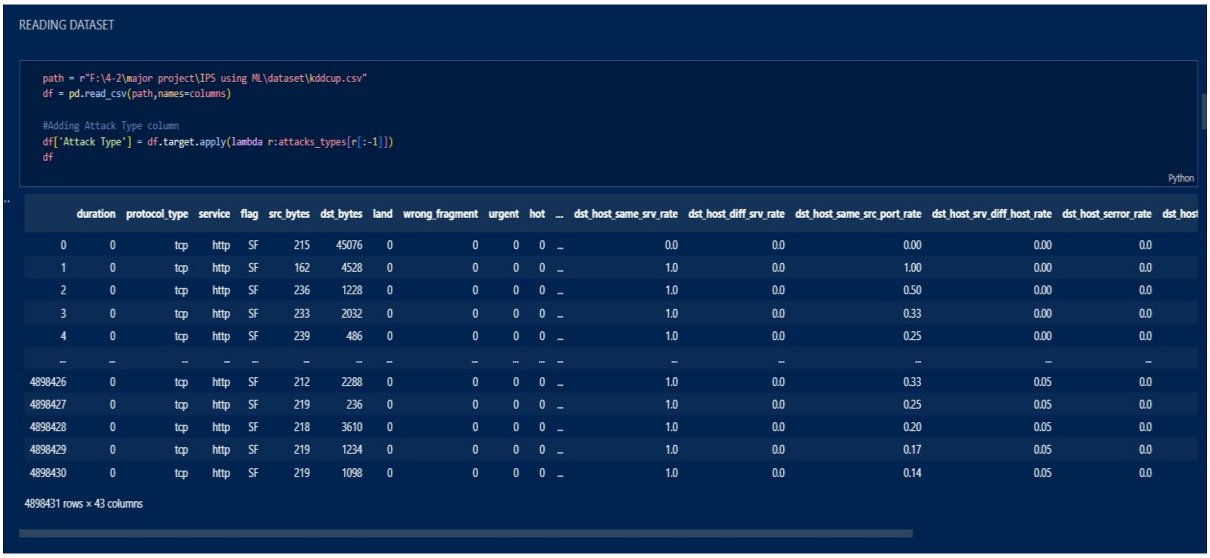
```
columns=[]
for c in cols.split(','):
    if(c.strip()):
        columns.append(c.strip())

columns.append('target')
#print(columns)
print(len(columns))
```

```
42
```



```
with open("dataset\\training_attack_types",'r') as f:
    print(f.read())
```

```
back dos
buffer_overflow u2r
ftp_write r2l
guess_passwd r2l
imap r2l
ipsweep probe
land dos
loadmodule u2r
multihop r2l
neptune dos
nmap probe
perl u2r
phf r2l
pod dos
portsweep probe
rootkit u2r
satan probe
smurf dos
spy r2l
teardrop dos
warezclient r2l
warezmaster r2l
```



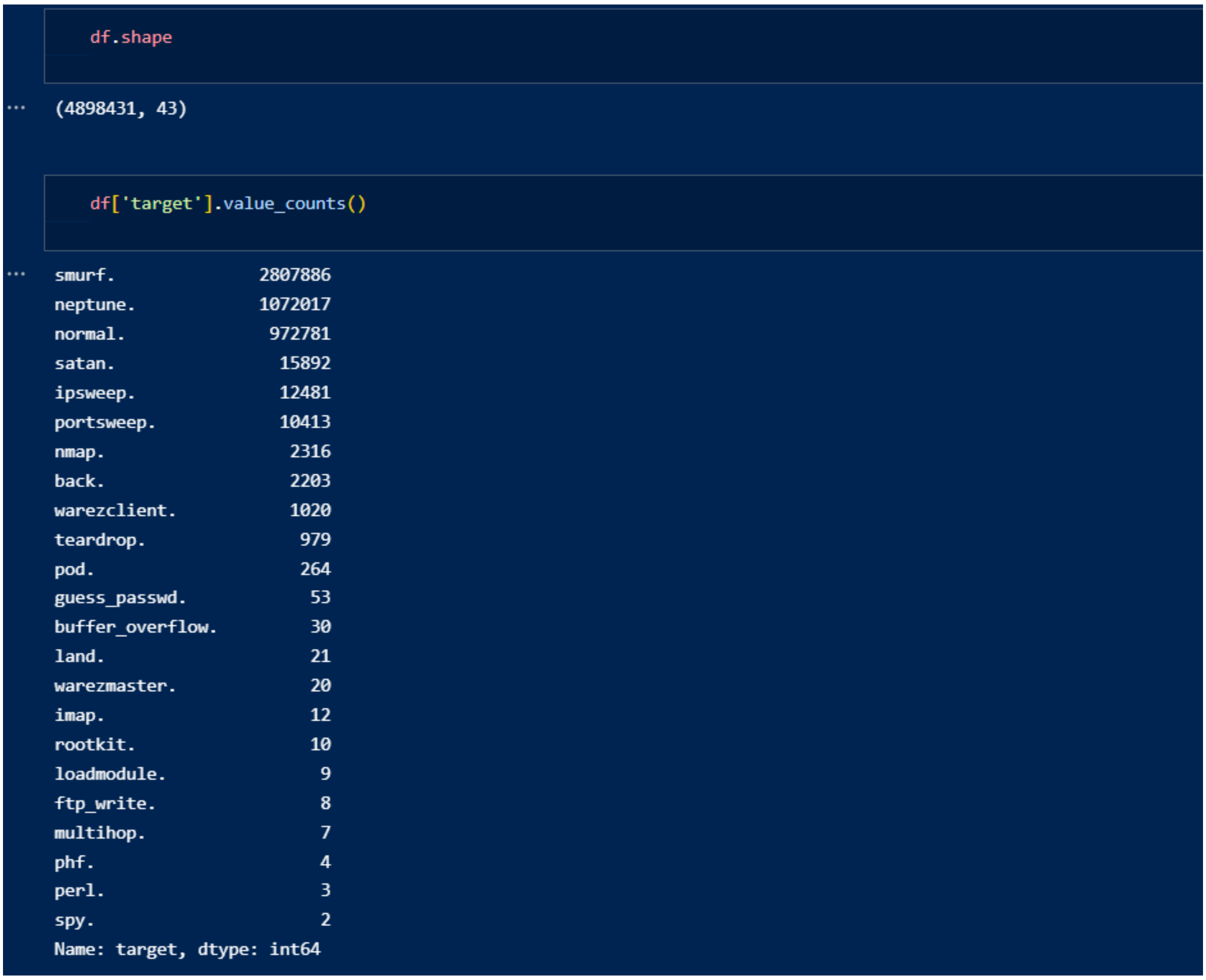
READING DATASET

```
path = r"F:\4-2\major project\IPS using ML\dataset\kddcup.csv"
df = pd.read_csv(path,names=columns)

#Adding Attack Type column
df['Attack Type'] = df.target.apply(lambda r:attacks_types[r[:-1]])
df
```

Python

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host
0	0	tcp	http	SF	215	45076	0	0	0	0	...	0.0	0.0	0.00	0.00	0.0	
1	0	tcp	http	SF	162	4528	0	0	0	0	...	1.0	0.0	1.00	0.00	0.0	
2	0	tcp	http	SF	236	1228	0	0	0	0	...	1.0	0.0	0.50	0.00	0.0	
3	0	tcp	http	SF	233	2032	0	0	0	0	...	1.0	0.0	0.33	0.00	0.0	
4	0	tcp	http	SF	239	486	0	0	0	0	...	1.0	0.0	0.25	0.00	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4898426	0	tcp	http	SF	212	2288	0	0	0	0	...	1.0	0.0	0.33	0.05	0.0	
4898427	0	tcp	http	SF	219	236	0	0	0	0	...	1.0	0.0	0.25	0.05	0.0	
4898428	0	tcp	http	SF	218	3610	0	0	0	0	...	1.0	0.0	0.20	0.05	0.0	
4898429	0	tcp	http	SF	219	1234	0	0	0	0	...	1.0	0.0	0.17	0.05	0.0	
4898430	0	tcp	http	SF	219	1098	0	0	0	0	...	1.0	0.0	0.14	0.05	0.0	

4898431 rows × 43 columns

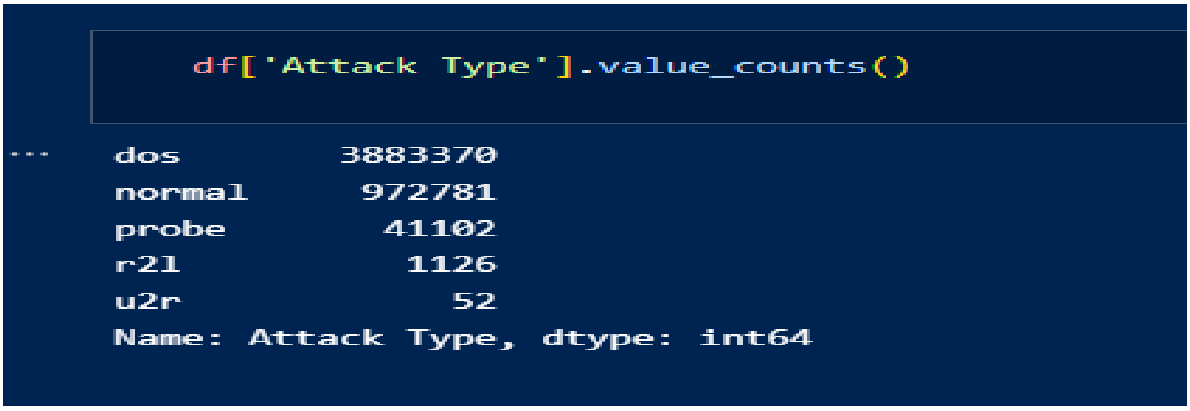


```
df.shape
```

```
(4898431, 43)
```

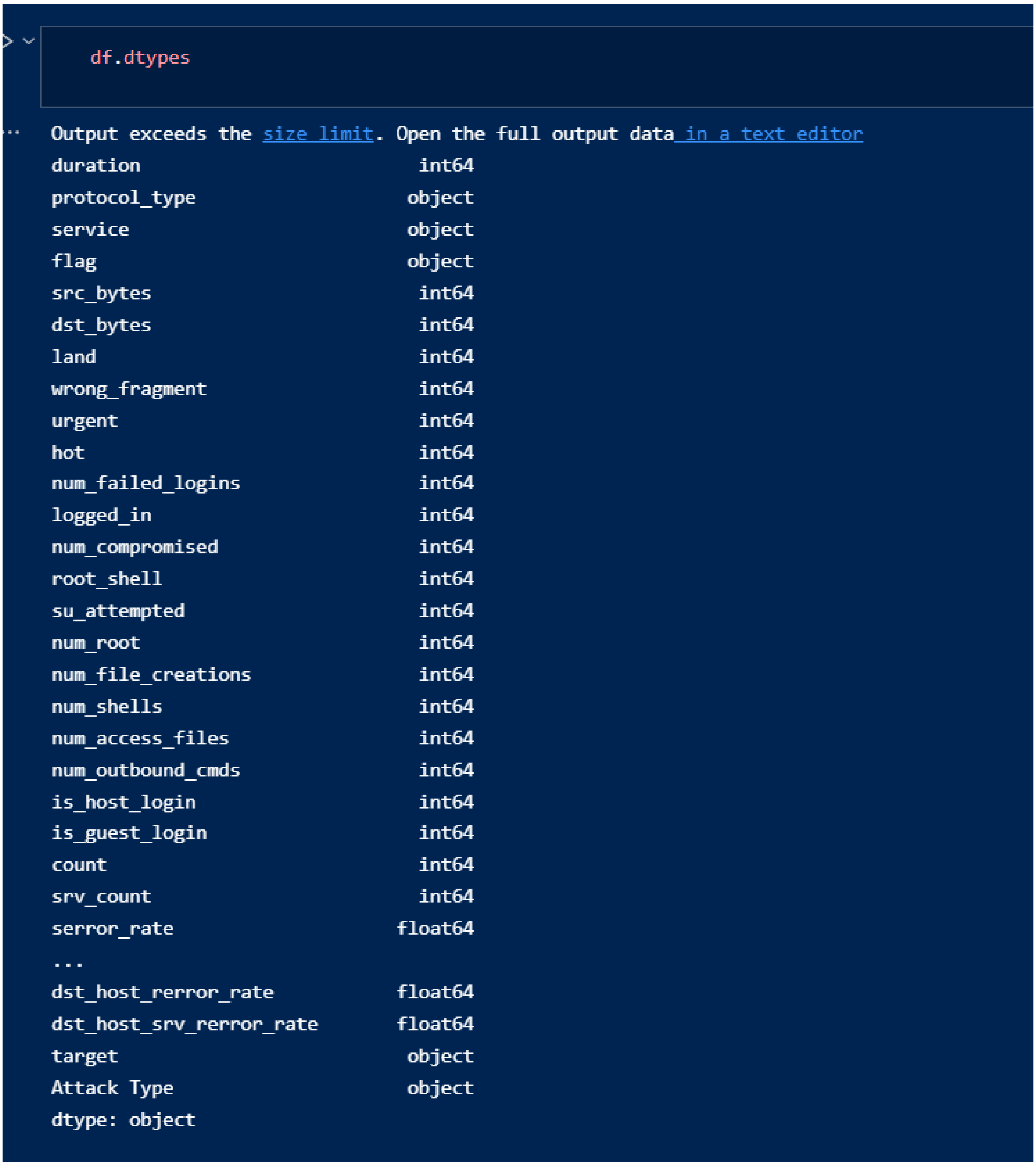
```
df['target'].value_counts()
```

```
smurf.              2807886
neptune.            1072017
normal.              972781
satan.                15892
ipsweep.              12481
portsweep.            10413
nmap.                  2316
back.                  2203
warezclient.           1020
teardrop.               979
pod.                    264
guess_passwd.            53
buffer_overflow.         30
land.                    21
warezmaster.             20
imap.                    12
rootkit.                 10
loadmodule.               9
ftp_write.                8
multihop.                 7
phf.                      4
perl.                     3
spy.                      2
Name: target, dtype: int64
```



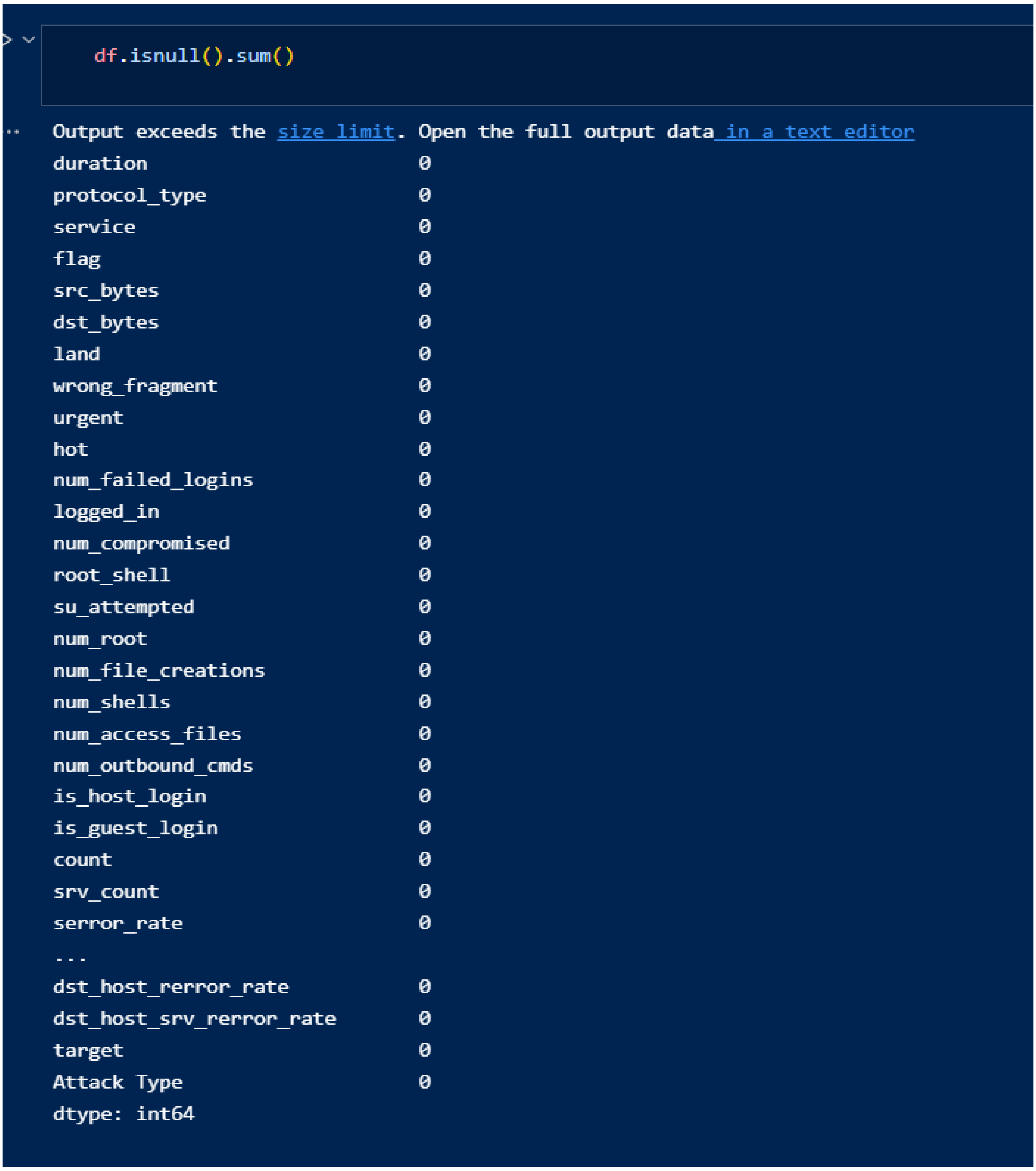
```
df['Attack Type'].value_counts()
```

```
dos        3883370
normal      972781
probe        41102
r2l           1126
u2r             52
Name: Attack Type, dtype: int64
```



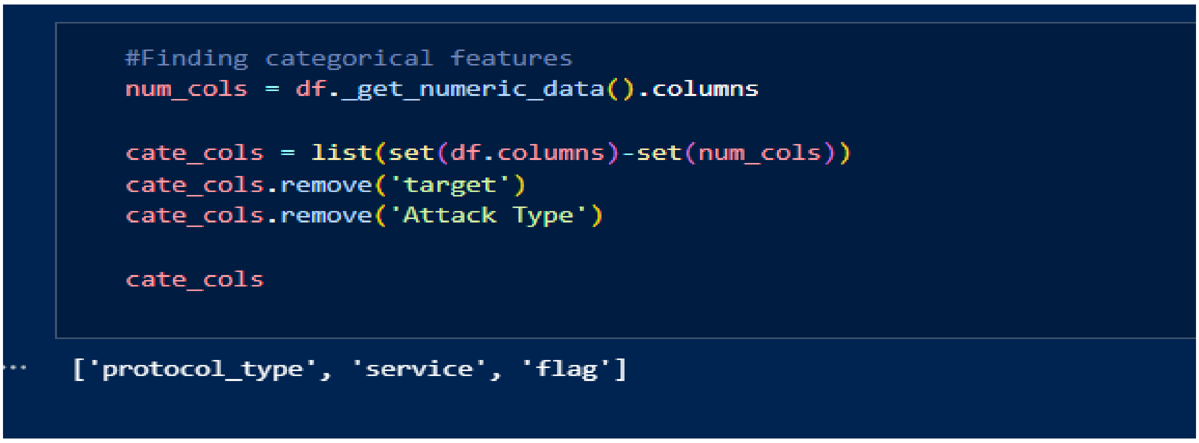
```
df.dtypes
```

```
Output exceeds the size limit. Open the full output data in a text editor
duration                          int64
protocol_type                    object
service                          object
flag                             object
src_bytes                         int64
dst_bytes                         int64
land                              int64
wrong_fragment                    int64
urgent                            int64
hot                               int64
num_failed_logins                 int64
logged_in                         int64
num_compromised                   int64
root_shell                        int64
su_attempted                      int64
num_root                          int64
num_file_creations                int64
num_shells                        int64
num_access_files                  int64
num_outbound_cmds                 int64
is_host_login                     int64
is_guest_login                    int64
count                             int64
srv_count                         int64
serror_rate                     float64
...
dst_host_rerror_rate            float64
dst_host_srv_rerror_rate        float64
target                           object
Attack Type                      object
dtype: object
```



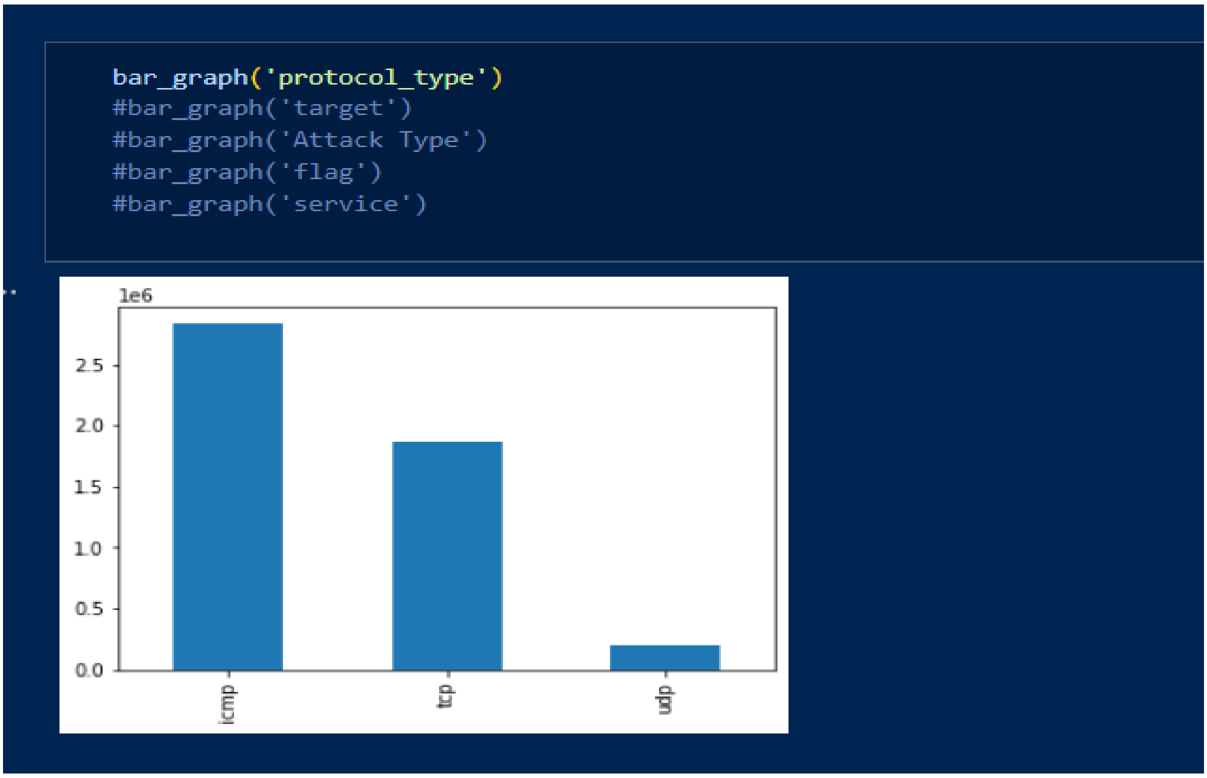
```
df.isnull().sum()
```

```
Output exceeds the size limit. Open the full output data in a text editor
duration                       0
protocol_type                  0
service                        0
flag                           0
src_bytes                      0
dst_bytes                      0
land                           0
wrong_fragment                 0
urgent                         0
hot                            0
num_failed_logins              0
logged_in                      0
num_compromised                0
root_shell                     0
su_attempted                   0
num_root                       0
num_file_creations             0
num_shells                     0
num_access_files               0
num_outbound_cmds              0
is_host_login                  0
is_guest_login                 0
count                          0
srv_count                      0
serror_rate                    0
...
dst_host_rerror_rate           0
dst_host_srv_rerror_rate       0
target                         0
Attack Type                    0
dtype: int64
```

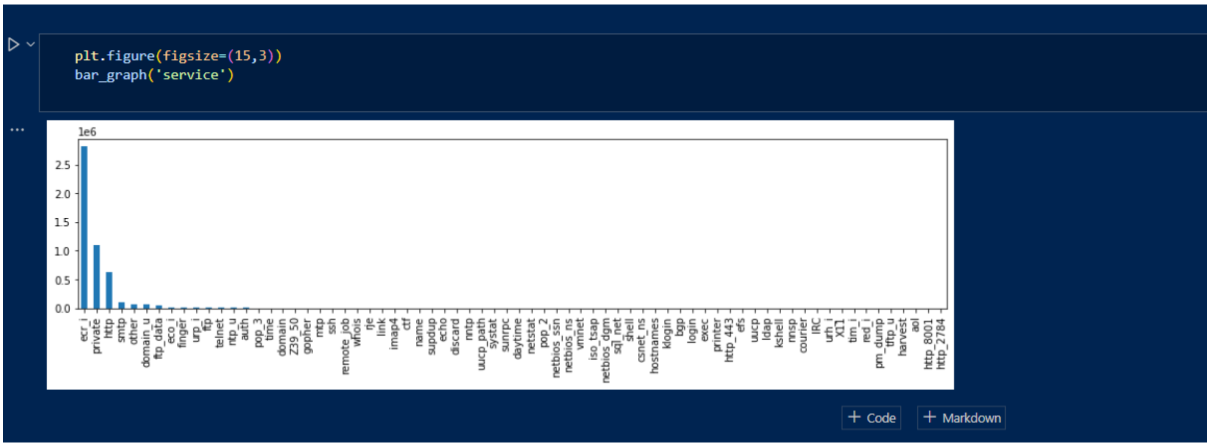


```
#Finding categorical features
num_cols = df._get_numeric_data().columns

cate_cols = list(set(df.columns)-set(num_cols))
cate_cols.remove('target')
cate_cols.remove('Attack Type')

cate_cols
```

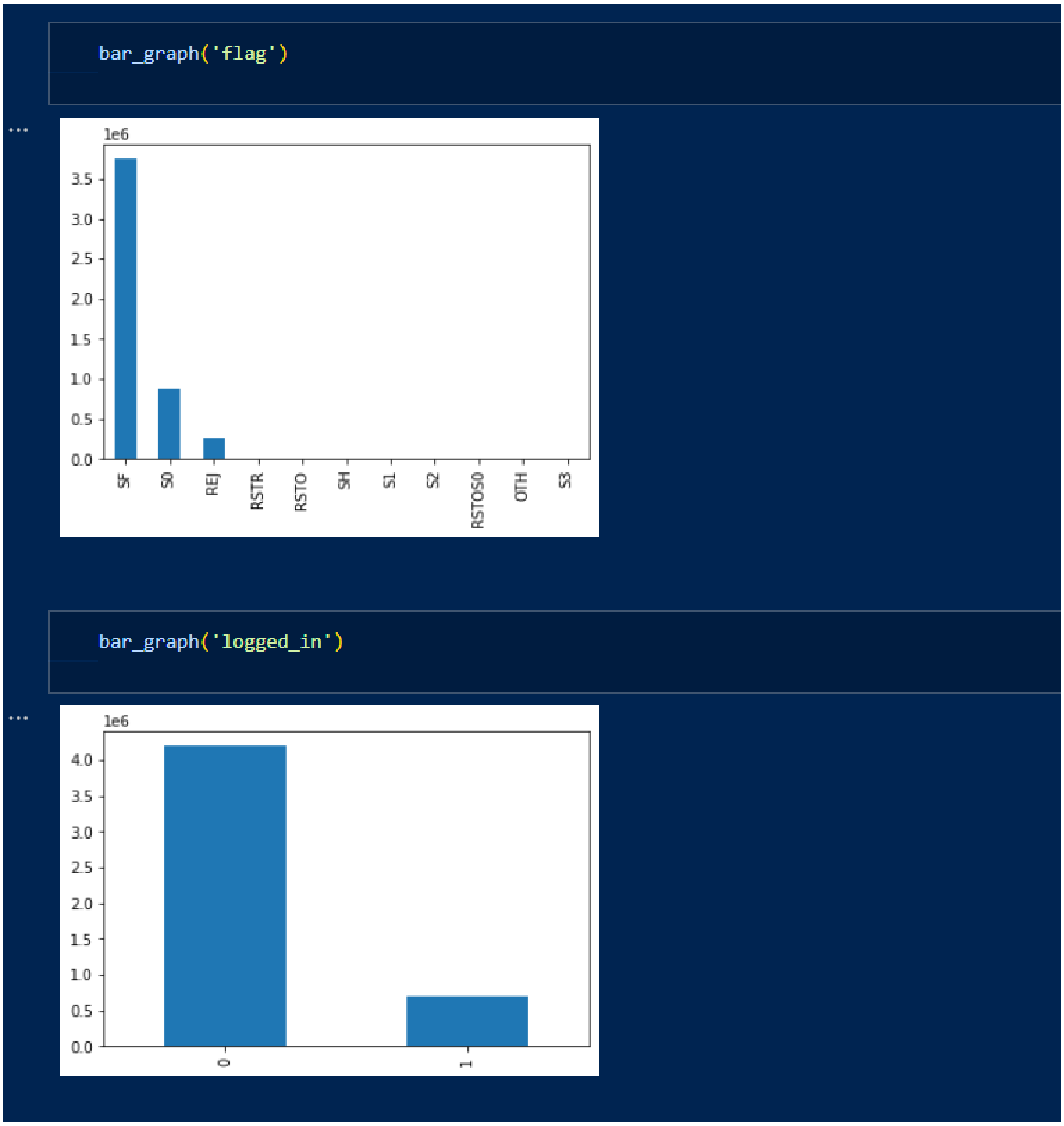
```
['protocol_type', 'service', 'flag']
```



```
bar_graph('protocol_type')
#bar_graph('target')
#bar_graph('Attack Type')
#bar_graph('flag')
#bar_graph('service')
```

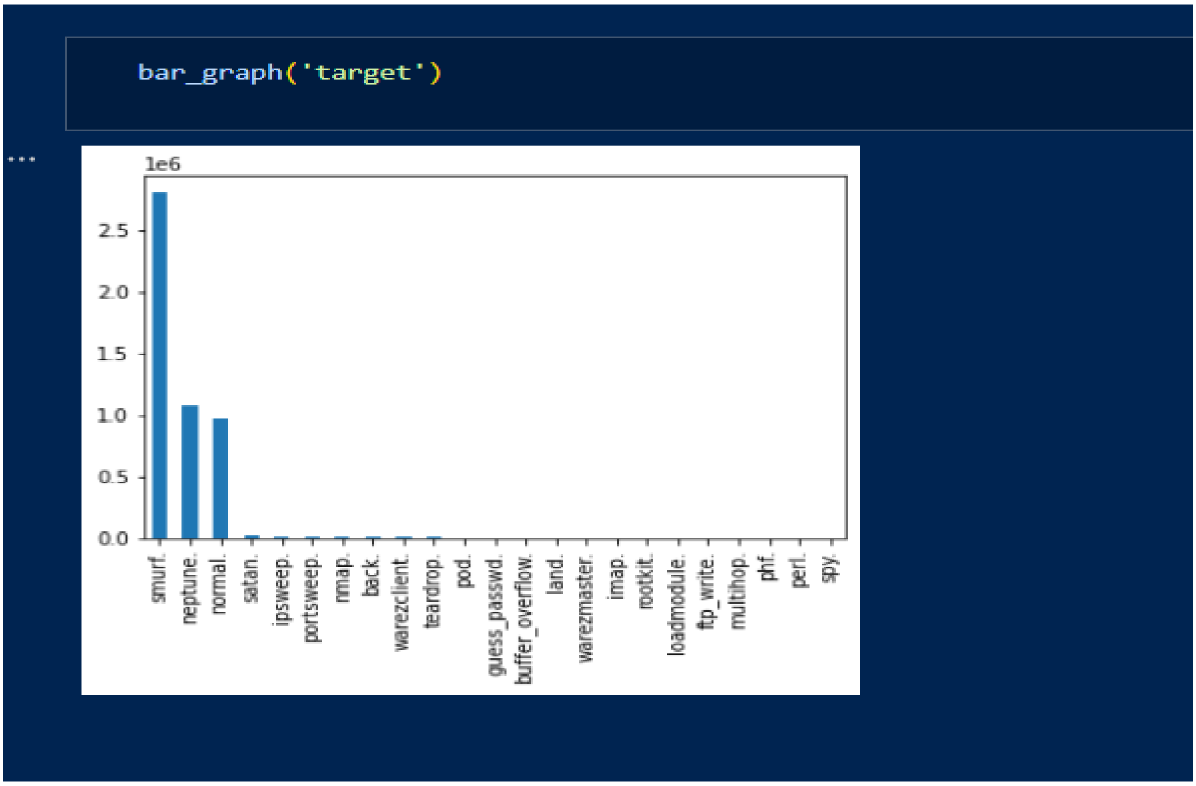


```
plt.figure(figsize=(15,3))
bar_graph('service')
```

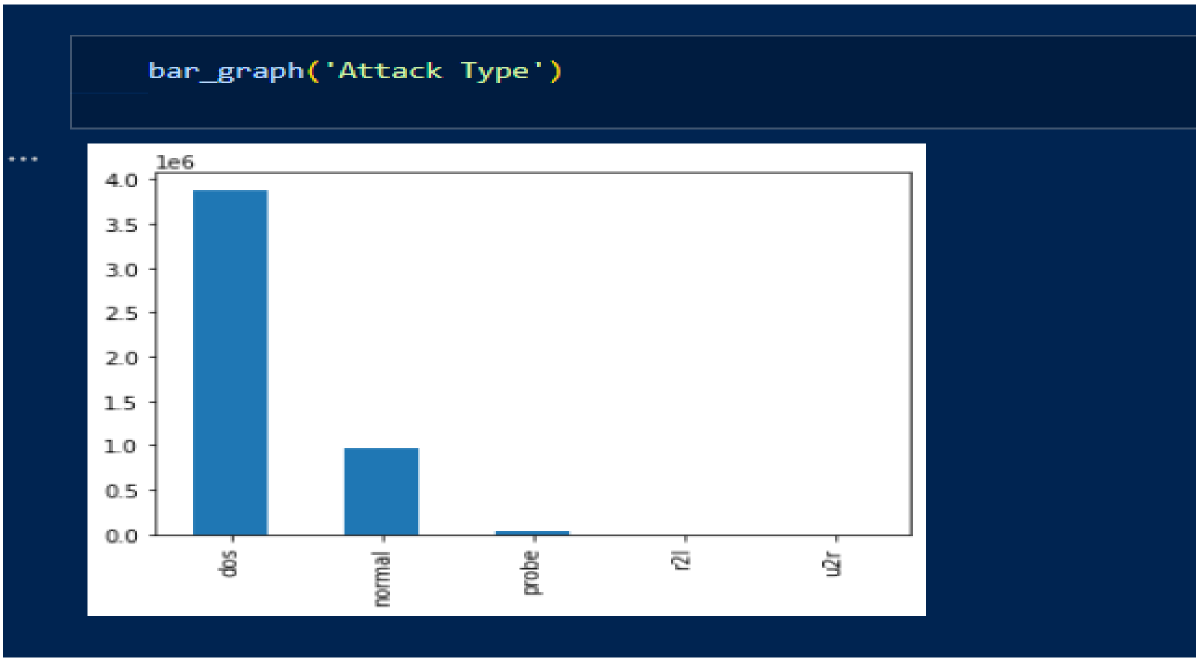


```
bar_graph('flag')
```

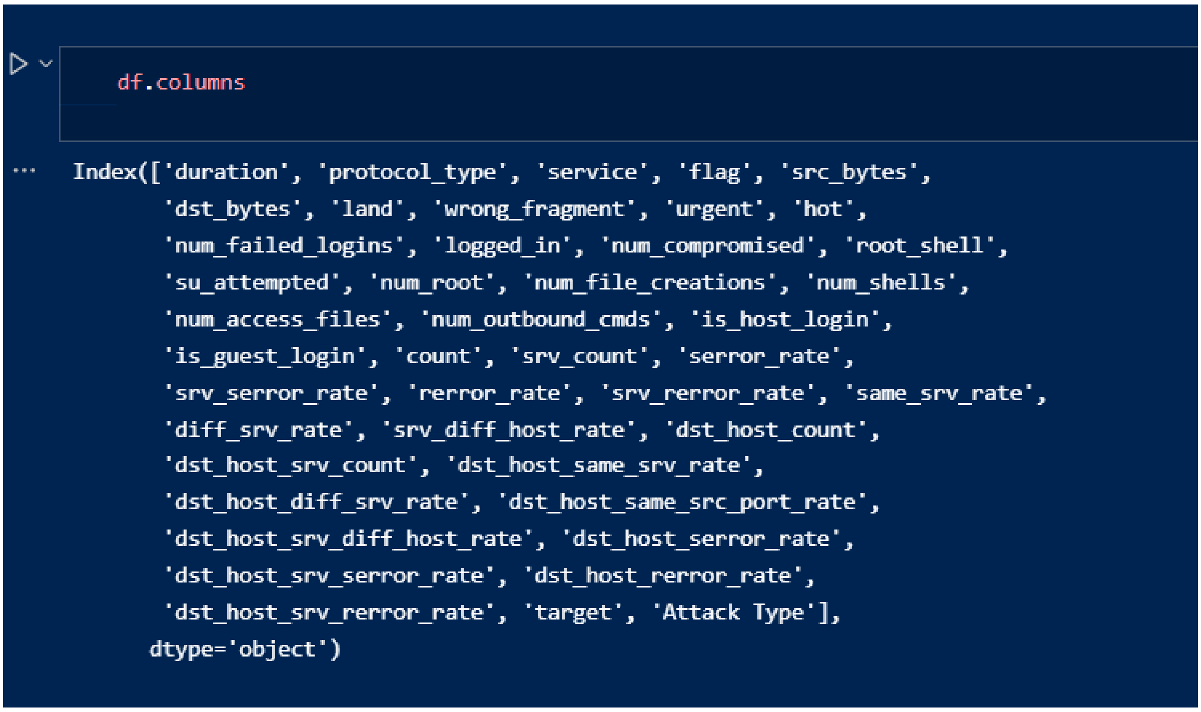
```
bar_graph('logged_in')
```



```
bar_graph('target')
```

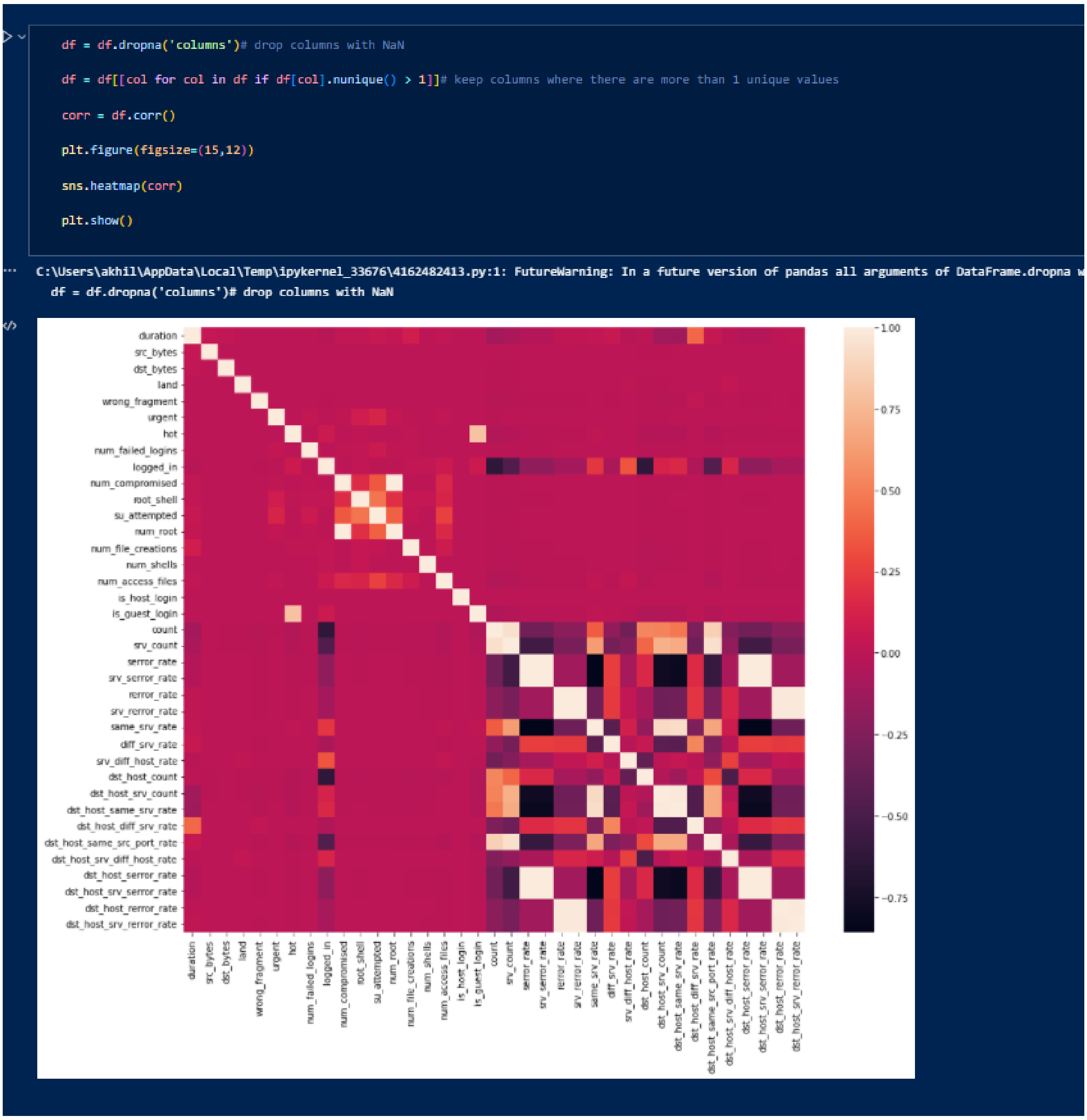


```
bar_graph('Attack Type')
```



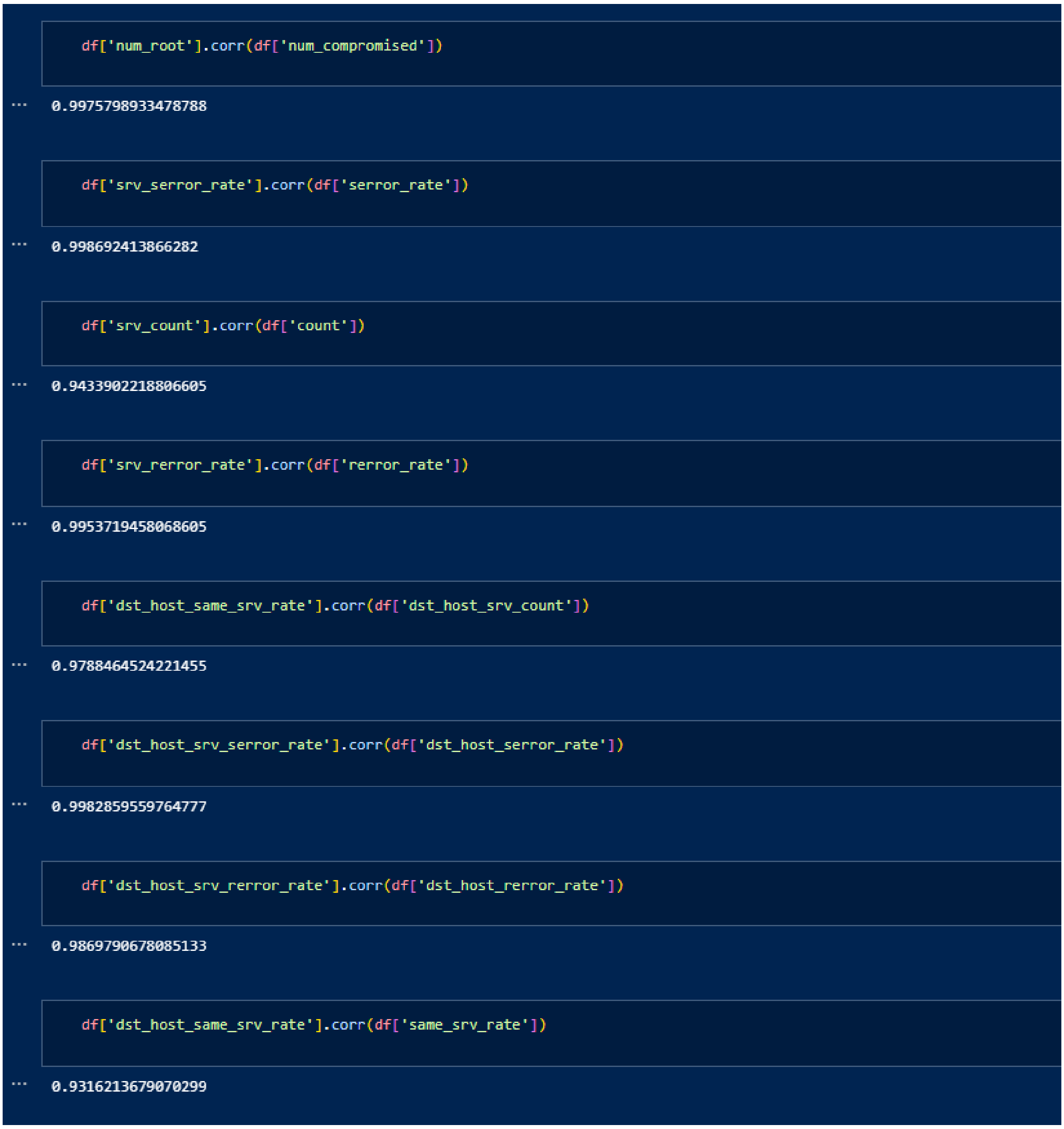
```
df.columns
```

```
Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
       'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
       'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
       'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
       'num_access_files', 'num_outbound_cmds', 'is_host_login',
       'is_guest_login', 'count', 'srv_count', 'serror_rate',
       'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
       'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
       'dst_host_srv_count', 'dst_host_same_srv_rate',
       'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
       'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
       'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
       'dst_host_srv_rerror_rate', 'target', 'Attack Type'],
      dtype='object')
```



```
df = df.dropna('columns')# drop columns with NaN

df = df[[col for col in df if df[col].nunique() > 1]]# keep columns where there are more than 1 unique values

corr = df.corr()

plt.figure(figsize=(15,12))

sns.heatmap(corr)

plt.show()
```

```
C:\Users\akhil\AppData\Local\Temp\ipykernel_33676\4162482413.py:1: FutureWarning: In a future version of pandas all arguments of DataFrame.dropna w
  df = df.dropna('columns')# drop columns with NaN
```



```
df['num_root'].corr(df['num_compromised'])
```

```
0.9975798933478788
```

```
df['srv_serror_rate'].corr(df['serror_rate'])
```

```
0.998692413866282
```

```
df['srv_count'].corr(df['count'])
```

```
0.9433902218806605
```

```
df['srv_rerror_rate'].corr(df['rerror_rate'])
```

```
0.9953719458068605
```

```
df['dst_host_same_srv_rate'].corr(df['dst_host_srv_count'])
```

```
0.9788464524221455
```

```
df['dst_host_srv_serror_rate'].corr(df['dst_host_serror_rate'])
```

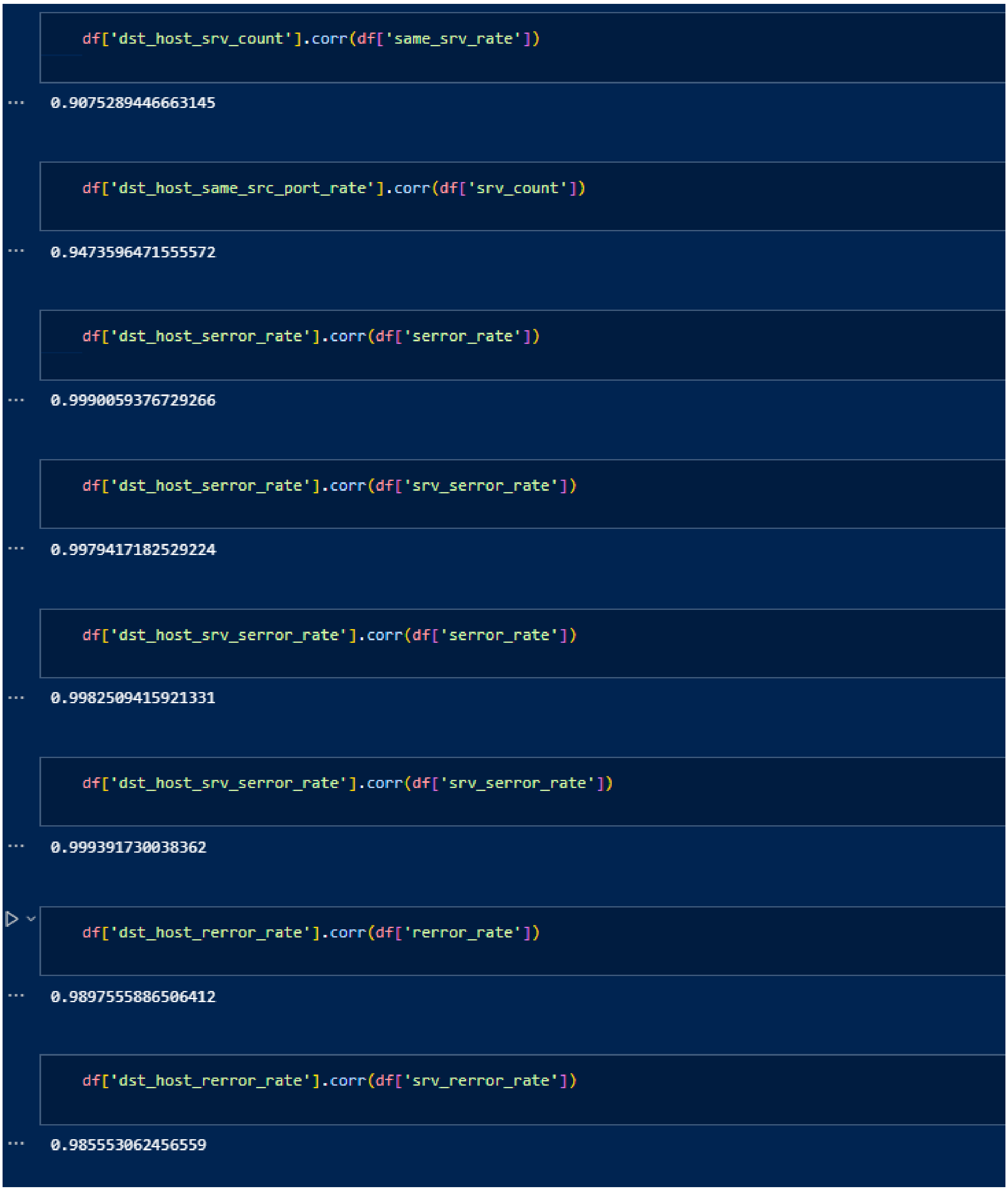
```
0.9982859559764777
```

```
df['dst_host_srv_rerror_rate'].corr(df['dst_host_rerror_rate'])
```

```
0.9869790678085133
```

```
df['dst_host_same_srv_rate'].corr(df['same_srv_rate'])
```

```
0.9316213679070299
```



```
df['dst_host_srv_count'].corr(df['same_srv_rate'])
```

```
0.9075289446663145
```

```
df['dst_host_same_src_port_rate'].corr(df['srv_count'])
```

```
0.9473596471555572
```

```
df['dst_host_serror_rate'].corr(df['serror_rate'])
```

```
0.9990059376729266
```

```
df['dst_host_serror_rate'].corr(df['srv_serror_rate'])
```

```
0.9979417182529224
```

```
df['dst_host_srv_serror_rate'].corr(df['serror_rate'])
```

```
0.9982509415921331
```

```
df['dst_host_srv_serror_rate'].corr(df['srv_serror_rate'])
```

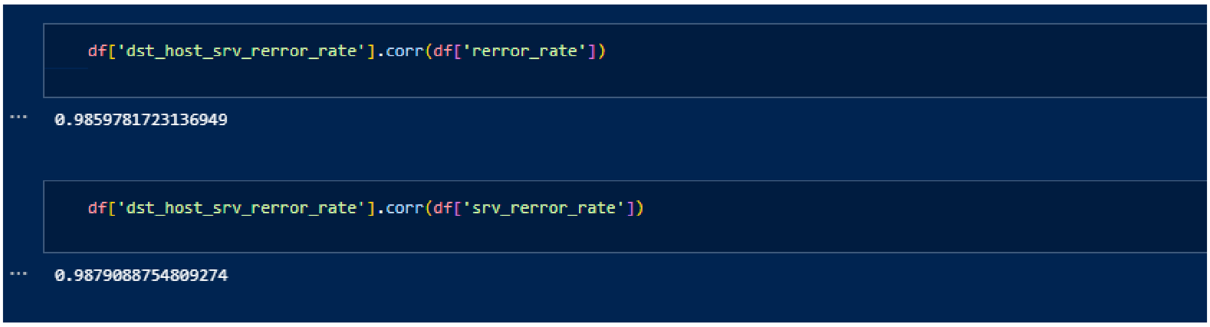
```
0.999391730038362
```

```
df['dst_host_rerror_rate'].corr(df['rerror_rate'])
```

```
0.9897555886506412
```

```
df['dst_host_rerror_rate'].corr(df['srv_rerror_rate'])
```

```
0.985553062456559
```

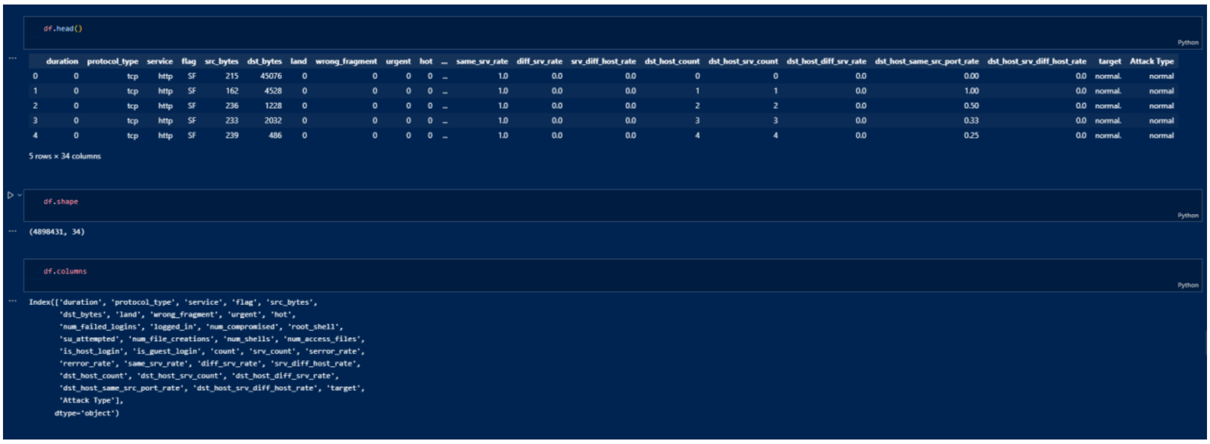


```
df['dst_host_srv_rerror_rate'].corr(df['rerror_rate'])
```

```
0.9859781723136949
```

```
df['dst_host_srv_rerror_rate'].corr(df['srv_rerror_rate'])
```

```
0.9879088754809274
```



```
df.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	same_srv_rate	diff_srv_rate	srv_diff_host_rate	dst_host_count	dst_host_srv_count	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	target	Attack Type
0	0	tcp	http	SF	215	45076	0	0	0	0	...	1.0	0.0	0.0	0	0	0.0	0.00	0.0	normal.	normal
1	0	tcp	http	SF	162	4528	0	0	0	0	...	1.0	0.0	0.0	1	1	0.0	1.00	0.0	normal.	normal
2	0	tcp	http	SF	236	1228	0	0	0	0	...	1.0	0.0	0.0	2	2	0.0	0.50	0.0	normal.	normal
3	0	tcp	http	SF	233	2032	0	0	0	0	...	1.0	0.0	0.0	3	3	0.0	0.33	0.0	normal.	normal
4	0	tcp	http	SF	239	486	0	0	0	0	...	1.0	0.0	0.0	4	4	0.0	0.25	0.0	normal.	normal

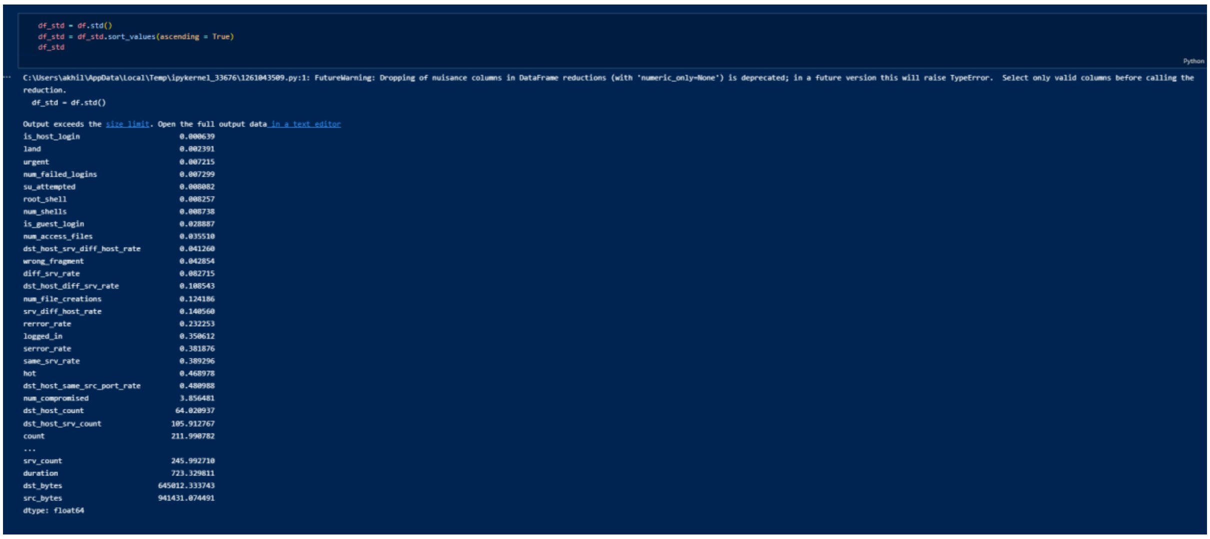
5 rows × 34 columns

```
df.shape
```

```
(4898431, 34)
```

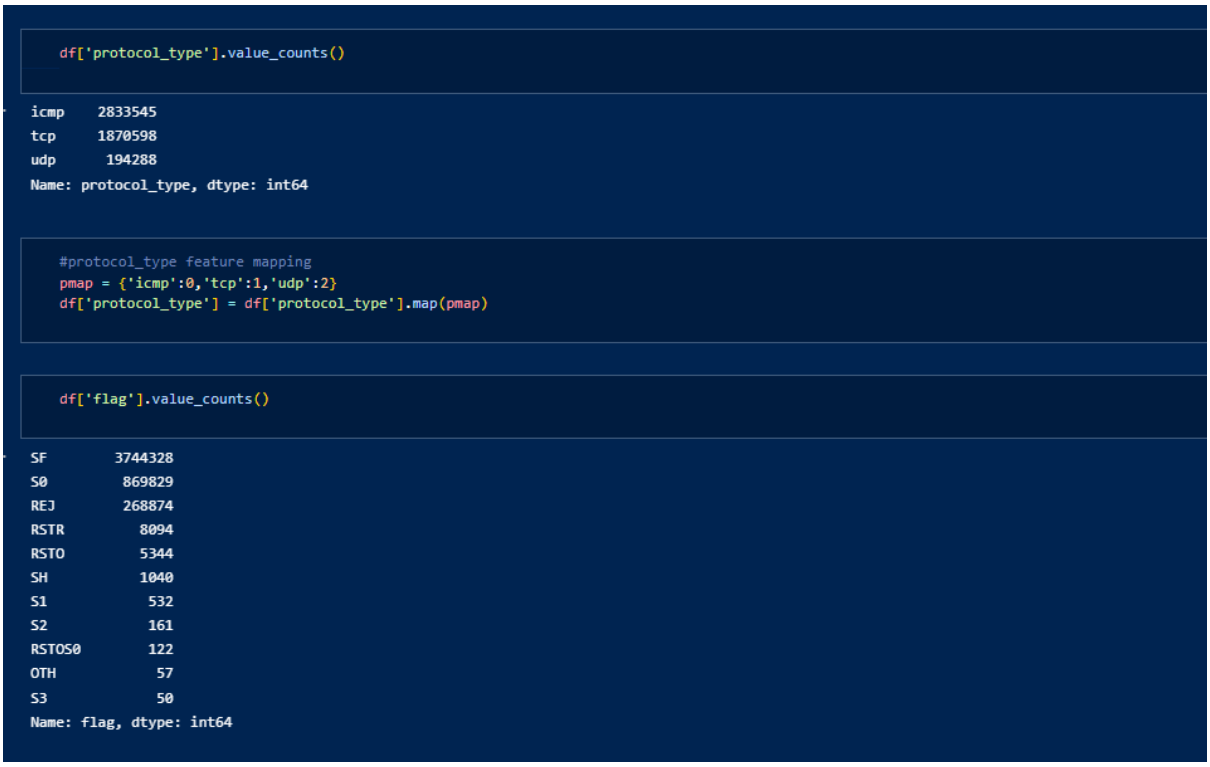
```
df.columns
```

```
Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
       'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
       'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
       'su_attempted', 'num_file_creations', 'num_shells', 'num_access_files',
       'is_host_login', 'is_guest_login', 'count', 'srv_count', 'serror_rate',
       'rerror_rate', 'same_srv_rate', 'diff_srv_rate', 'srv_diff_host_rate',
       'dst_host_count', 'dst_host_srv_count', 'dst_host_diff_srv_rate',
       'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'target',
       'Attack Type'],
      dtype='object')
```



```
df_std = df.std()
df_std = df_std.sort_values(ascending = True)
df_std
```

```
C:\Users\akhil\AppData\Local\Temp\ipykernel_33676\1261043509.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df_std = df.std()

Output exceeds the size limit. Open the full output data in a text editor
is_host_login                       0.000639
land                                0.002391
urgent                              0.007215
num_failed_logins                   0.007299
su_attempted                        0.008082
root_shell                          0.008257
num_shells                          0.008738
is_guest_login                      0.028887
num_access_files                    0.035510
dst_host_srv_diff_host_rate         0.041260
wrong_fragment                      0.042854
diff_srv_rate                       0.082715
dst_host_diff_srv_rate              0.108543
num_file_creations                  0.124186
srv_diff_host_rate                  0.140560
rerror_rate                         0.232253
logged_in                           0.350612
serror_rate                         0.381876
same_srv_rate                       0.389296
hot                                 0.468978
dst_host_same_src_port_rate         0.480988
num_compromised                     3.856481
dst_host_count                     64.020937
dst_host_srv_count                105.912767
count                             211.990782
...
srv_count                         245.992710
duration                          723.329811
dst_bytes                      645012.333743
src_bytes                      941431.074491
dtype: float64
```



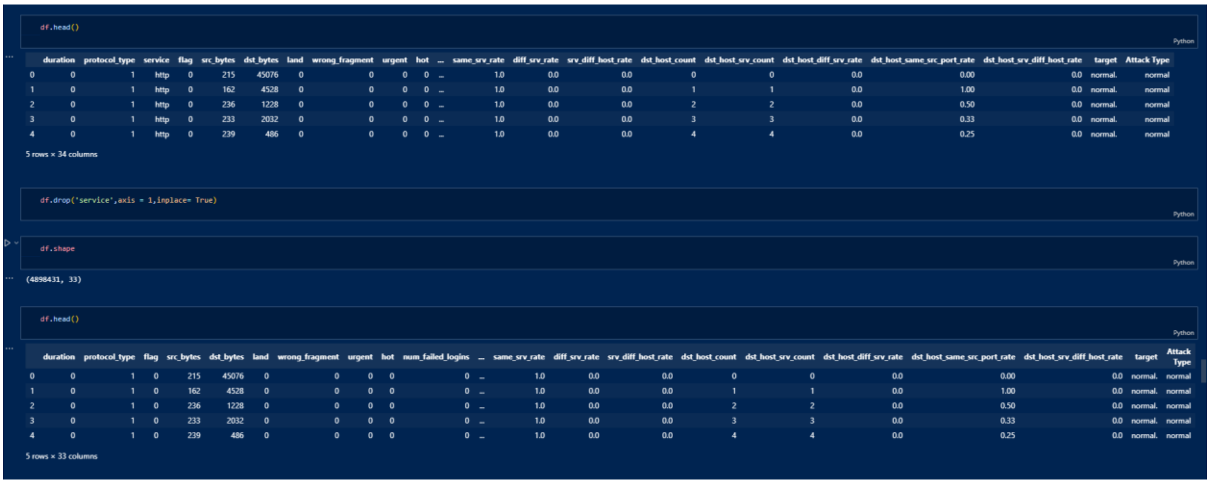
```
df['protocol_type'].value_counts()
```

```
icmp    2833545
tcp     1870598
udp      194288
Name: protocol_type, dtype: int64
```

```
#protocol_type feature mapping
pmap = {'icmp':0,'tcp':1,'udp':2}
df['protocol_type'] = df['protocol_type'].map(pmap)
```

```
df['flag'].value_counts()
```

```
SF        3744328
S0         869829
REJ        268874
RSTR         8094
RSTO         5344
SH           1040
S1            532
S2            161
RSTOS0        122
OTH            57
S3             50
Name: flag, dtype: int64
```



```
df.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	same_srv_rate	diff_srv_rate	srv_diff_host_rate	dst_host_count	dst_host_srv_count	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	target	Attack Type
0	0	1	http	0	215	45076	0	0	0	0	...	1.0	0.0	0.0	0	0	0.0	0.00	0.0	normal.	normal
1	0	1	http	0	162	4528	0	0	0	0	...	1.0	0.0	0.0	1	1	0.0	1.00	0.0	normal.	normal
2	0	1	http	0	236	1228	0	0	0	0	...	1.0	0.0	0.0	2	2	0.0	0.50	0.0	normal.	normal
3	0	1	http	0	233	2032	0	0	0	0	...	1.0	0.0	0.0	3	3	0.0	0.33	0.0	normal.	normal
4	0	1	http	0	239	486	0	0	0	0	...	1.0	0.0	0.0	4	4	0.0	0.25	0.0	normal.	normal

5 rows × 34 columns

```
df.drop('service',axis = 1,inplace= True)
```

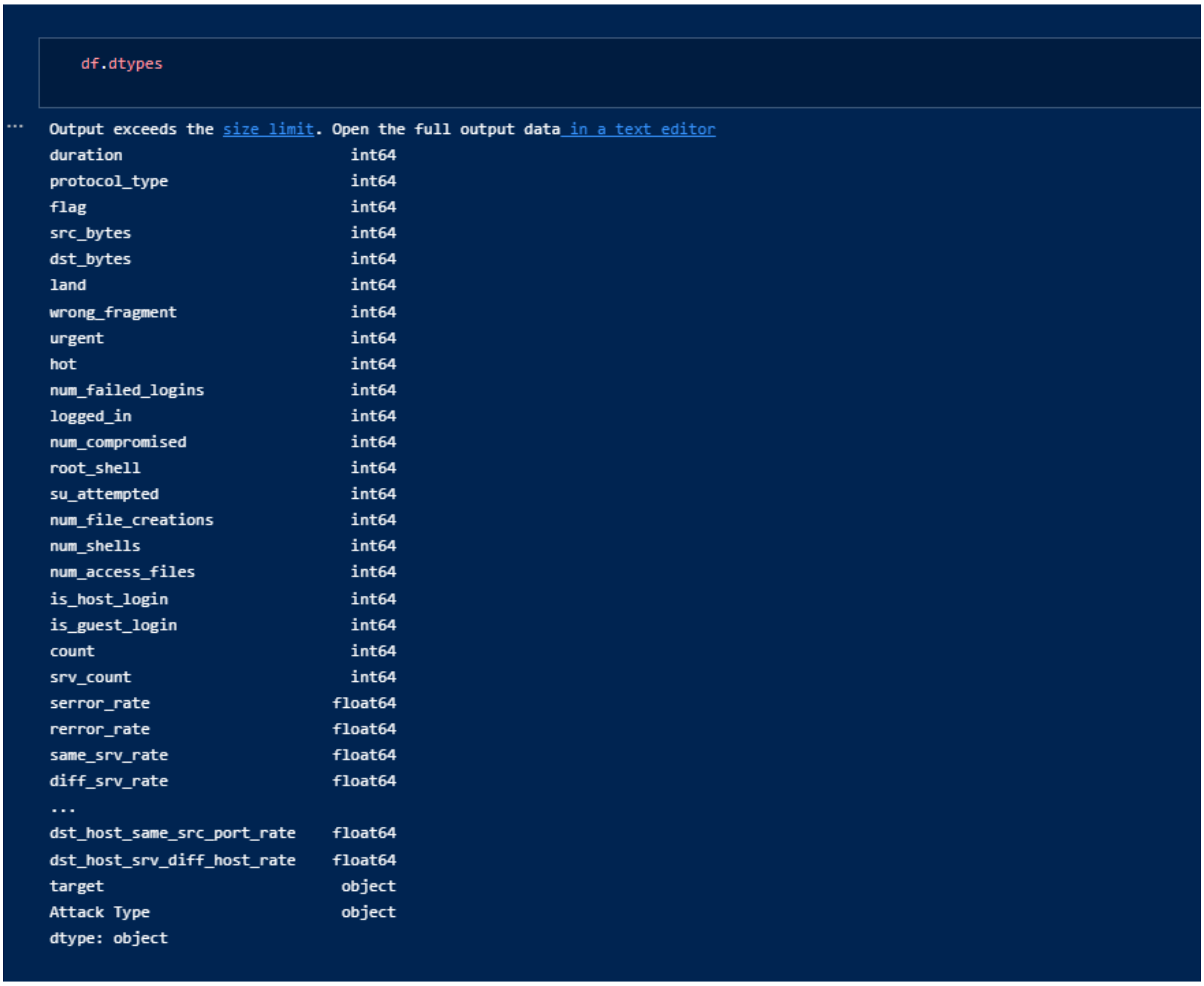
```
df.shape
```

```
(4898431, 33)
```

```
df.head()
```

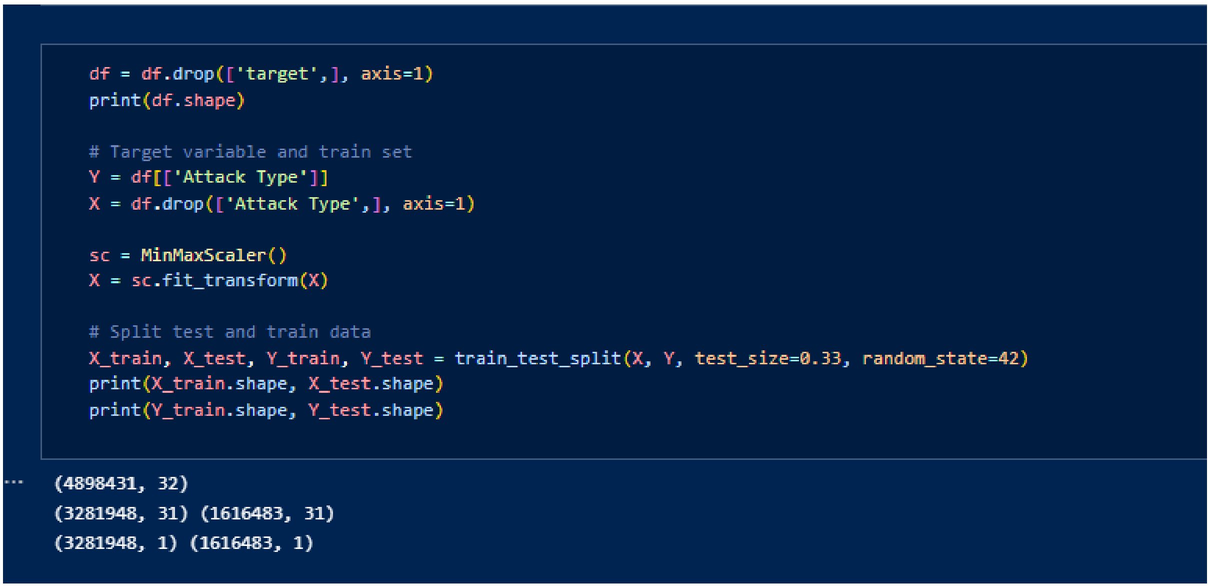
	duration	protocol_type	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	...	same_srv_rate	diff_srv_rate	srv_diff_host_rate	dst_host_count	dst_host_srv_count	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	target	Attack Type
0	0	1	0	215	45076	0	0	0	0	0	...	1.0	0.0	0.0	0	0	0.0	0.00	0.0	normal.	normal
1	0	1	0	162	4528	0	0	0	0	0	...	1.0	0.0	0.0	1	1	0.0	1.00	0.0	normal.	normal
2	0	1	0	236	1228	0	0	0	0	0	...	1.0	0.0	0.0	2	2	0.0	0.50	0.0	normal.	normal
3	0	1	0	233	2032	0	0	0	0	0	...	1.0	0.0	0.0	3	3	0.0	0.33	0.0	normal.	normal
4	0	1	0	239	486	0	0	0	0	0	...	1.0	0.0	0.0	4	4	0.0	0.25	0.0	normal.	normal

5 rows × 33 columns



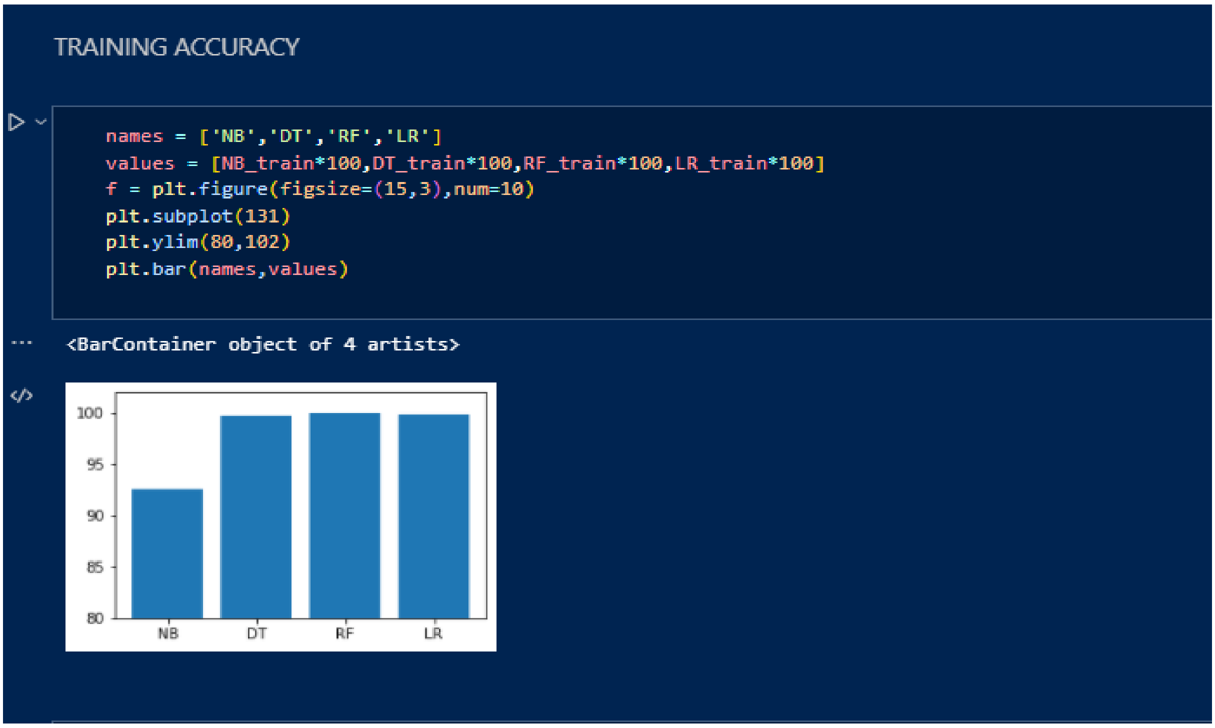
```
df.dtypes
```

```
Output exceeds the size limit. Open the full output data in a text editor
duration                         int64
protocol_type                    int64
flag                             int64
src_bytes                        int64
dst_bytes                        int64
land                             int64
wrong_fragment                   int64
urgent                           int64
hot                              int64
num_failed_logins                int64
logged_in                        int64
num_compromised                  int64
root_shell                       int64
su_attempted                     int64
num_file_creations               int64
num_shells                       int64
num_access_files                 int64
is_host_login                    int64
is_guest_login                   int64
count                            int64
srv_count                        int64
serror_rate                    float64
rerror_rate                    float64
same_srv_rate                  float64
diff_srv_rate                  float64
...
dst_host_same_src_port_rate    float64
dst_host_srv_diff_host_rate    float64
target                          object
Attack Type                     object
dtype: object
```



```
df = df.drop(['target',], axis=1)
print(df.shape)

# Target variable and train set
Y = df[['Attack Type']]
X = df.drop(['Attack Type',], axis=1)

sc = MinMaxScaler()
X = sc.fit_transform(X)

# Split test and train data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_state=42)
print(X_train.shape, X_test.shape)
print(Y_train.shape, Y_test.shape)
```

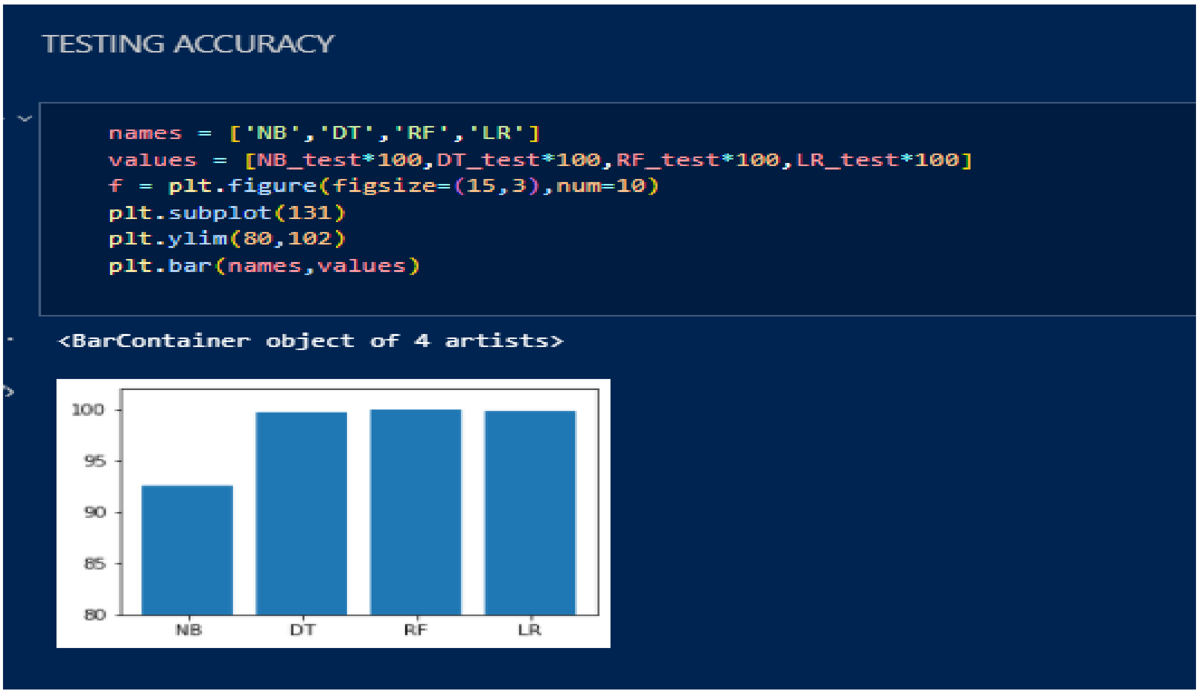
```
(4898431, 32)
(3281948, 31) (1616483, 31)
(3281948, 1) (1616483, 1)
```



TRAINING ACCURACY

```
names = ['NB','DT','RF','LR']
values = [NB_train*100,DT_train*100,RF_train*100,LR_train*100]
f = plt.figure(figsize=(15,3),num=10)
plt.subplot(131)
plt.ylim(80,102)
plt.bar(names,values)
```

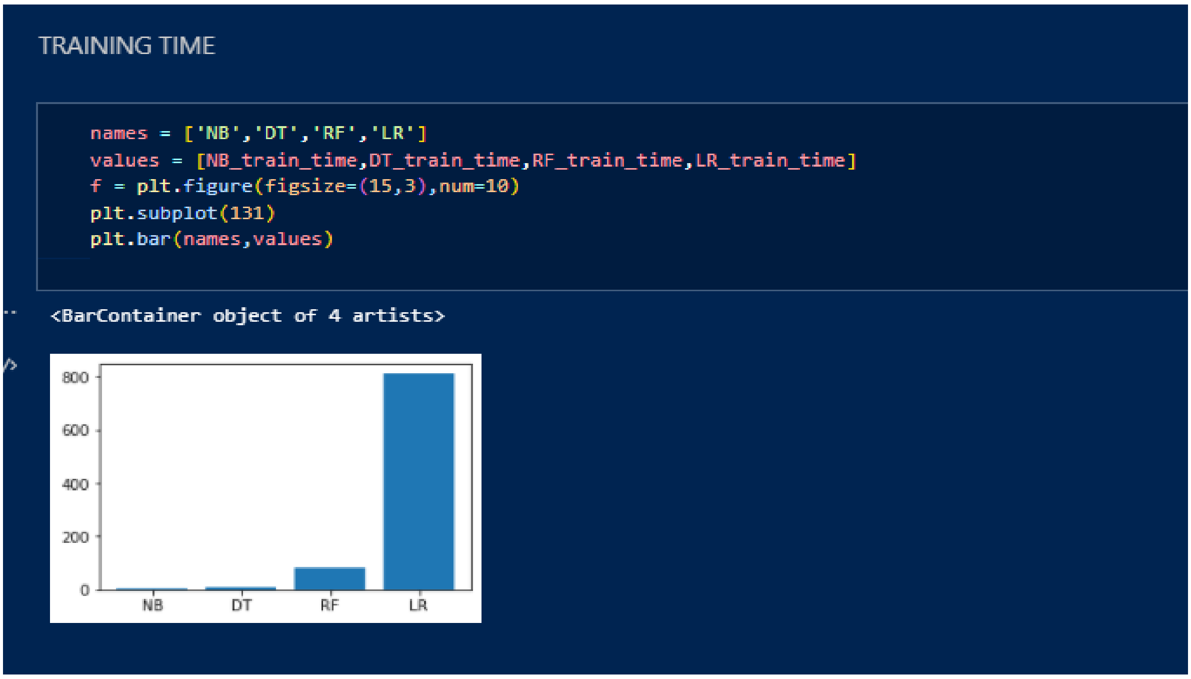
```
<BarContainer object of 4 artists>
```



TESTING ACCURACY

```
names = ['NB','DT','RF','LR']
values = [NB_test*100,DT_test*100,RF_test*100,LR_test*100]
f = plt.figure(figsize=(15,3),num=10)
plt.subplot(131)
plt.ylim(80,102)
plt.bar(names,values)
```

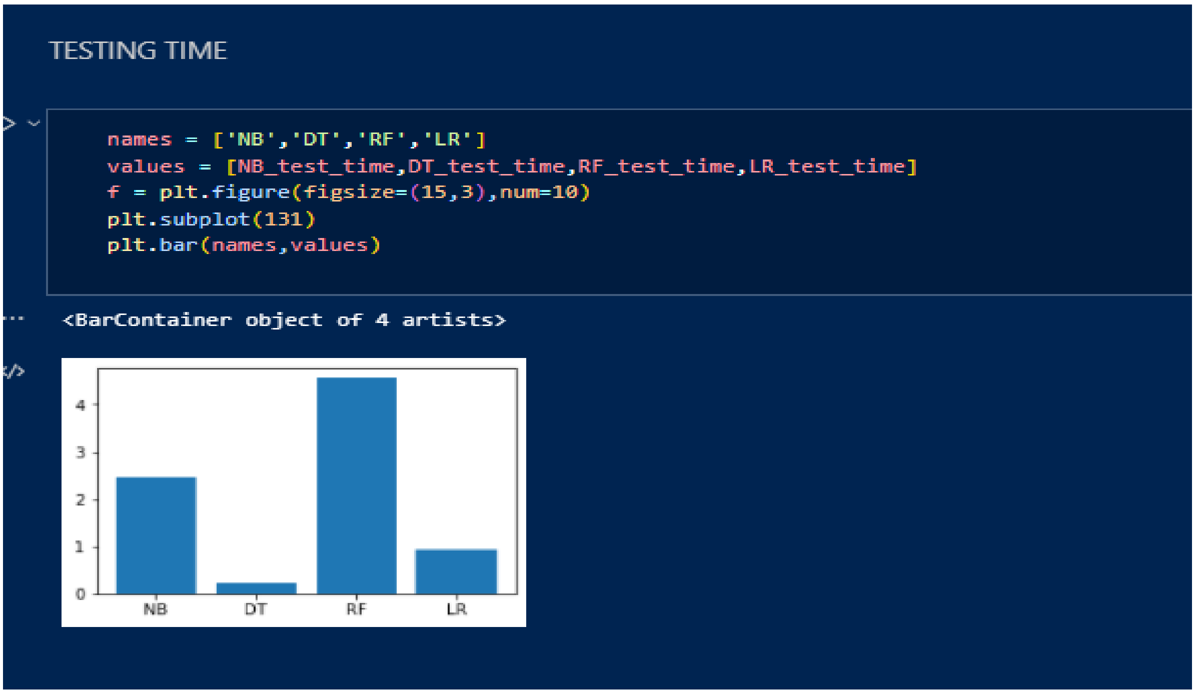
```
<BarContainer object of 4 artists>
```



TRAINING TIME

```
names = ['NB','DT','RF','LR']
values = [NB_train_time,DT_train_time,RF_train_time,LR_train_time]
f = plt.figure(figsize=(15,3),num=10)
plt.subplot(131)
plt.bar(names,values)
```

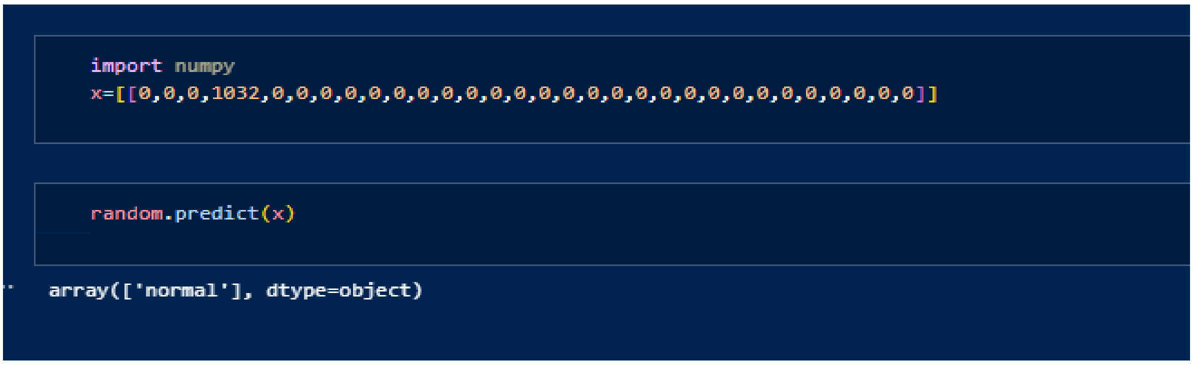
```
<BarContainer object of 4 artists>
```



TESTING TIME

```
names = ['NB','DT','RF','LR']
values = [NB_test_time,DT_test_time,RF_test_time,LR_test_time]
f = plt.figure(figsize=(15,3),num=10)
plt.subplot(131)
plt.bar(names,values)
```

```
<BarContainer object of 4 artists>
```



```
import numpy
x=[[0,0,0,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]]
```

```
random.predict(x)
```

```
array(['normal'], dtype=object)
```

8.2 Results

Algorithm	Training Accuracy	Testing Accuracy
Naïve Bayes	92.51	92.49
Decision Tree	99.61	99.61
Random Forest	99.99	99.98
Logistic Regression	99.78	99.78

9. CONCLUSION AND FUTURE WORK

9.1 Conclusion and Future Work

In a world with over usage of internet and the increasing in the number of attacks on the data available in a network, we believe our project can help us conclude on using machine learning algorithms to detect attacks and filter malicious packets on a network. Many organizations are already developing programs that can detect and report the cyber-attacks. There are many advances and upgrades happening in them. But we are in a world where there is a new type of attack evolving every new day. So, we need to make sure the programs which we are developing can upgrade themselves to these changes and work accurately and efficiently in real time.

Using this project, we can efficiently detect and filter packets from the attacks on which the model is trained. There are multiple models in this project and we can choose according to our requirement the kind of model we want, deploy it in the server and can detect all the attacks accurately.

In future, this program or software can be very useful to everyone and we will try to use a more realistic dataset and implement this model in a network in real-world. Also, we will make sure the software can be able to update itself to the new upcoming attacks.

10. REFERENCES

10.1 References

[1]T.Saranyaa, S.Sridevib, C.Deisyc, Tran Duc Chungd, M.K.A.Ahamed Khane, Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review, Elsevier (CoCoNet'19).

[2]Sudhakar, Sushil Kumar, Botnet Detection Techniques and Research Challenges, ResearchGate 2019.

[3]Yadu Singh, Future Correlation Matters a Lot…, LinkedIn 2021

[4]SH Kok, Azween Abdullah, NZ Jhanjhi, Mahadevan Supramaniam, A Review of Intrusion Detection System using Machine Learning Approach, ResearchGate 2019.

[5]Mrutyunjaya Panda, Ajith Abraham, Swagatam Das, Manas Ranjan Patra, Network Intrusion Detection System: A Machine Learning Approach, ResearchGate 2011.

[6]U. S. Musa, S. Chakraborty, M. M. Abdullahi and T. Maini, "A Review on Intrusion Detection System using Machine Learning Techniques," 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2021, pp. 541-549, doi: 10.1109/ICCCIS51004.2021.9397121.

[7]M. K. Yadav and K. P. Sharma, "Intrusion Detection System using Machine Learning Algorithms: A Comparative Study," 2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC), 2021, pp. 415-420, doi: 10.1109/ICSCCC51823.2021.9478086.

[8]Mohammad Almseidin, Maen Alzubi, Szilveszter Kovacs and Mouhammd Alkasassbeh, Evaluation of Machine Learning Algorithms for Intrusion Detection System, arxiv.

[9]C. Fleizach and S. Fukushima, A naive bayes classifier on 1998 kdd cup, 1998.

[10]A. Cutler and G. Zhao, Pert-perfect random tree ensembles, Computing Science and Statistics, vol. 33, pp. 490–497, 2001.

[11]L. Breiman, Random forests, Machine learning, vol. 45, no. 1, pp. 5–32, 2001.

[12]N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," Machine learning, vol. 29, no. 2-3, pp. 131–163, 1997.