MALICIOUS URL PREDICTION

A PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE ENGINEERING

Under the guidance of

MAHENDRA DATTA

BY

AYAN DAS

TAMOGHNA GHOSH

MOUMI DAS

ABHISEK MAITI

FUTURE INSTITUTE OF ENGINEERING AND MANAGEMENT

In association with

(ISO 9001:2015)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

1. Title of the Project: **MALICIOUS URL PREDICTION**

2. Project Members: **AYAN DAS**
TAMOGHNA GHOSH
MOUMI DAS
ABHISEK MAITI

3. Name of the guide: **Mr. MAHENDRA DUTTA**

4. Address: Ardent Computech Pvt. Ltd
(An ISO 9001:2015 Certified)
SDF Building, Module #132, Ground Floor,
Salt Lake City, GP Block, Sector V,
Kolkata, West Bengal, 700091

Project Version Control History

Version	Primary Author	Description of Version	Date Completed
Final	AYAN DAS TAMOGHNA GHOSH MOUMI DAS ABHISEK MAITI	Project Report	30.01.2022

Ayan Das

Tamoghna Ghosh

Moumi Das

Abhisek Maiti

Signature of Team Member

Date: 28/01/2022

Mahendra Datta

Signature of Approver

Date: 28/01/2022

For Office Use Only

Approved	Not Approved

MR. MAHENDRA DATTA

Project Proposal Evaluator

DECLARATION

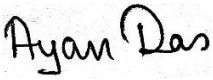
We hereby declare that the project work being presented in the project proposal entitled "**MALICIOUS URL PREDICTION**" in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY at ARDENT COMPUTECH PVT. LTD, SALTLAKE, KOLKATA, WEST BENGAL,** is an authentic work carried out under the guidance of **MR. MAHENDRA DATTA.** The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: 30.01.2022

Name of the Students:

AYAN DAS, TAMOGHNA GHOSH, MOUMI DAS, ABHISEK MAITI

Signature of the students:



Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

CERTIFICATE

This is to certify that this proposal of minor project entitled

"MALICIOUS URL PREDICTION" is a record of bonafide work, carried out by AYAN DAS, TAMOGHNA GHOSH, MOUMI DAS, ABHISEK MAITI under my guidance at **ARDENT COMPUTECH PVT LTD**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT®.** To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor

MR. MAHENDRA DATTA

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to **Mr. MAHENDRA DATTA**, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

CONTENTS

- Overview
- History of Python
- Environment Setup
- Basic Syntax
- Variable Types
- Functions
- Modules
- Packages
- Artificial Intelligence
- Machine Learning
    - Supervised, Semi-Supervised and Unsupervised Learning
    - NumPy
    - Pandas
    - Scikit-learn
    - Regression Analysis
    - K-Neighbour
    - Gaussian Naive Bayes
    - Decision Tree
    - Random Forest
    - Matplotlib

- MALICIOUS URL PREDICTION

INTRODUCTION TO PYTHON

Python is an interpreted high-level general-purpose programming language. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

In Python, we don't need to declare the type of variable because it is a dynamically typed language. Python has gained popularity because of its simplicity, easy syntax and user-friendly environment.
Some of its features are mentioned below:

- **Object-Oriented Language:**
  In Python, we don't need to declare the type of variable because it is a dynamically typed language.

- **High-Level Language:**
  Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

- **Large Standard Library:**
  Python has a large standard library which provides a rich set of module and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.

- **Embeddable:**
  The code of the other programming language can use in the Python source code. We can use Python source code in another programming language as well. It can embed other language into our code.

- **Portable:** Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

HISTORY OF PYTHON

Python was initially designed by Guido van Rossum and he began working on Python in the late 1980s. He started his application-based work in December of 1989 at Centrum Wiskunde & Informatica (CWI) situated in Netherland. The first ever version of Python (Python 1.0) was introduced in 1991.

ABC programming language is said to be the predecessor of Python language, which was capable of Exception Handling and interfacing with the Amoeba Operating System. Python's main objective is to provide code readability and advanced developer productivity.

It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. The inspiration for the name came from BBC's TV Show – 'Monty Python's Flying Circus'.

ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- Palm OS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX

Type the following text at the Python prompt and press the Enter –

>>> print "Hello, User!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, User!");

However, in Python version 2.4.3, this produces the following result –

Hello, User!

Python Keywords:

Python has a set of keywords that are reserved words that cannot be used as variable names, function names, or any other identifiers. Keywords are also called as reserved words. They are as follows:

and- A logical operator*, as*- To create an alias

break- To break out of a loop*, class*- To define a class

except- Used with exceptions*, False*- Boolean value*.*

Class, from, print continue, global, raise def, if, return
del, import, try elif, in, while else, is, with, lambda, yield are some more examples.

Python Identifiers:

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language.

Lines & Indentation:

Indentation is the leading whitespace (spaces and tabs) before any statement in python. It is a mandatory concept that should be followed when writing a python code, otherwise, Indentation Error is thrown by the python interpreter. Python treats the statements that have the same indentation level (statements that have equal no of whitespaces before them) as a single block of code.

Example:

Case 1 with Wrong Indentation (Error):

```
if( 1 == 1):
 print("This is test code")
    print("This is test code1")
```

Case 2 with Correct Indentation:

```
if( 1 == 1):
 print("This is test code")
 print("This is test code1")
```

Command Line Arguments in Python:

The arguments that are given after the name of the program in the command line shell of the operating system are known as Command Line Arguments. Python enables you to do this with -h

− $ python-h

usage: python [option]...[-c cmd|-m mod | file |-][arg]...

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string(terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit [etc.]

VARIABLE TYPES

Variables are reserved memory locations to store values, when assigned. This means that when we create a variable the computer reserves some space in its memory. Some important points to remember when assigning a variable are:

- In variable name, no special characters allowed other than underscore (_)
- Variables are case sensitive.
- Variable name can have numbers but not at the beginning.
- Variable name should not be a Python keyword.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when we assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
Counter = 10       # An integer
avg = 10.12        # A floating point
name = "World"     # A string
```

Multiple Assignment

Python allows us to assign a single value to several variables simultaneously.

For example:
a = b = c = d=1
a, b, c, d = 1, 2, "Hello", “World!”

Standard Data Types

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.
The type() function is used to determine the type of data type.

Following are the standard or built-in data type of Python:

- Numeric
  - Integers
  - Float
  - Complex Numbers

- Sequence Type
  - String
  - List
  - Tuple

- Boolean
  - Boolean objects that are equal to True are truthy (true), and those equal to False are falsy (false). But non-Boolean objects can be evaluated in Boolean context as well and determined to be true or false. It is denoted by the class bool.

    True and False with capital 'T' and 'F' are valid booleans Other-wise python will throw an error.
- Set
  - It is an unordered collection of data type that is iterable, mutable and has no duplicate elements.

    Sets can be created by using the built-in set() function with an iterable object or a sequence by placing the sequence inside curly braces, separated by 'comma'.

- Dictionary

  - It is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key: value pair. a Dictionary can be created by placing a sequence of elements within curly {} braces, separated by 'comma'.

DATA TYPE CONVERSION

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.
There are several built-in functions to perform conversion from one data type to another. Some of them are mentioned below:

Sr.No.	Function & Description
1	**int(x [,base])** Converts x to an integer. base specifies the base if x is a string.
2	**long(x [,base])** Converts x to a long integer. base specifies the base if x is a string.
3	**float(x)** Converts x to a floating-point number.
4	**complex(real [,imag])** Creates a complex number.
5	**str(x)** Converts object x to a string representation.
6	**eval(str)** Evaluates a string and returns an object.
7	**tuple(s)** Converts s to a tuple.
8	**list(s)** Converts s to a list.

FUNCTIONS IN PYTHON

A function is a block of code which only runs when it is called. Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable. It also makes the code reusable. We can pass data, known as parameters, into a function. A function can return data as a result. Functions are basically of two types:

i. Built-in functions - Functions that are built into Python.
ii. User-defined functions - Functions defined by the users themselves.

In Python a function is defined using the def keyword:

```
def my_function():
    print(“Welcome!”)
```

To call a function, use the function name followed by parenthesis:

```
def my_function():
    print("Hello from a function")
my_function()
```

Arguments of a Function

Arguments are the values passed inside the parenthesis of the function.
A function can have any number of arguments separated by a comma.
Example: To check whether the number passed as an argument to the function is even or odd.

```
def evenOdd(x):
    if (x % 2 == 0):
        print("even")
    else:
        print("odd")
 #Calling the function
evenOdd(2)
evenOdd(3)
```

Pass by reference

When we pass something by reference any change, we make to the variable inside the function then those changes are reflected to the outside value as well.

Example:

```
student = {'Jim': 12, 'Anna': 14, 'Preet': 10}
def test(student):
   new = {'Sam':20, 'Steve':21}
   student.update(new)
   print("Inside the function", student)
   return
test(student)
print("Outside the function:", student)
```

Pass by value

When we pass something by value then the changes made to the function or copying of the variable are not reflected back to the calling function.

Example:

```
student = {'Jim': 12, 'Anna': 14, 'Preet': 10}
def test(student):
   student = {'Sam':20, 'Steve':21}
   print("Inside the function", student)
   return
test(student)
print("Outside the function:", student)
```

MODULES

In Python, Modules are simply files with the ".py" extension containing Python code that can be imported inside another Python Program.

In simple terms, we can consider a module to be the same as a code library or a file that contains a set of functions that you want to include in your application. With the help of modules, we can organize related functions, classes, or any code block in the same file.

The module contains the following components:

- Definitions and implementation of classes,
- Variables
- Functions that can be used inside another program.

To create a module:

- Save the code with a name of your choice in a file with the file extension ".py".
- The name of the Python file becomes the name of the module.

Example: The code is saved in a file named my_ownmodule.py

```
def welcome(name):
    print("Hello, " + name +". All the best for your new ventures.")
```

To incorporate the module into our program, we will use the import keyword. When we are using a function from a module, then we use the following syntax:

```
module_name.function_name
```

In this example, we will Import the module named mymodule, and then call the welcome function with a given argument:

```
import my_ownmodule
my_ownmodule.welcome("Ayan Das")
```

Output:

```
Hello, Ayan Das. All the best for your new ventures.
```

PACKAGES

A Python package usually consists of several modules. Its is basically a directory with Python files and a file with the name __init__.py. This means that every directory inside of the Python path, which contains a file named __init__.py, will be treated as a package by Python. It's possible to put several modules into a Package.

Consider a file Pots.py available in Phone directory. This file has following line of source code −

```
def Pots ():

print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

- Phone/Isdn.py file having function Isdn ()
- Phone/G3.py file having function G3 ()

Now, create one more file __init__.py in Phone directory −

- Phone/__init__.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in __init__.py as follows −

```
from Pots import Pots

from Isdn import Isdn

from G3 import
```

ARTIFICIAL INTELLIGENCE

[Image source: Internet]

Introduction:

Artificial intelligence (AI), is the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. AI can be achieved by reading the behaviour of humans and using the results to develop intelligent systems. For example, they learn, make decisions and act in certain situations. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience.

Aim:

- The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception, and the ability to move and manipulate objects.
- General intelligence: The ability to solve an arbitrary problem is among the field's long-term goals.
- An intelligent agent that can plan makes a representation of the state of the world, makes predictions about how their actions will change it and makes choices that maximize the utility (or "value") of the available choices.
- Social Intelligence: Effective computing is the study and development of systems that can detect, interpret, process, and simulate human.

Applications:

AI applications were at the heart of the most commercially successful areas of computing, and have become a ubiquitous feature of daily life.

- **Search engines** (such as Google Search), targeting online advertisements, recommendation systems (offered by Netflix, YouTube or Amazon) are some important applications.

- **Virtual assistants** (such as Siri or Alexa), automatic language translation (Microsoft Translator, Google Translate)

- **Facial recognition** (Apple's Face ID or Microsoft's DeepFace)

- **Healthcare:** The AI algorithm in a smartwatch monitors a person's vital activity to detect heart problems and even alerts emergency services. Additionally, AI has helped increase the speed and accuracy of drugs.

- **Financial Services**: It is useful for bookkeeping management, stock forecasting, and even fraud prevention in banking.

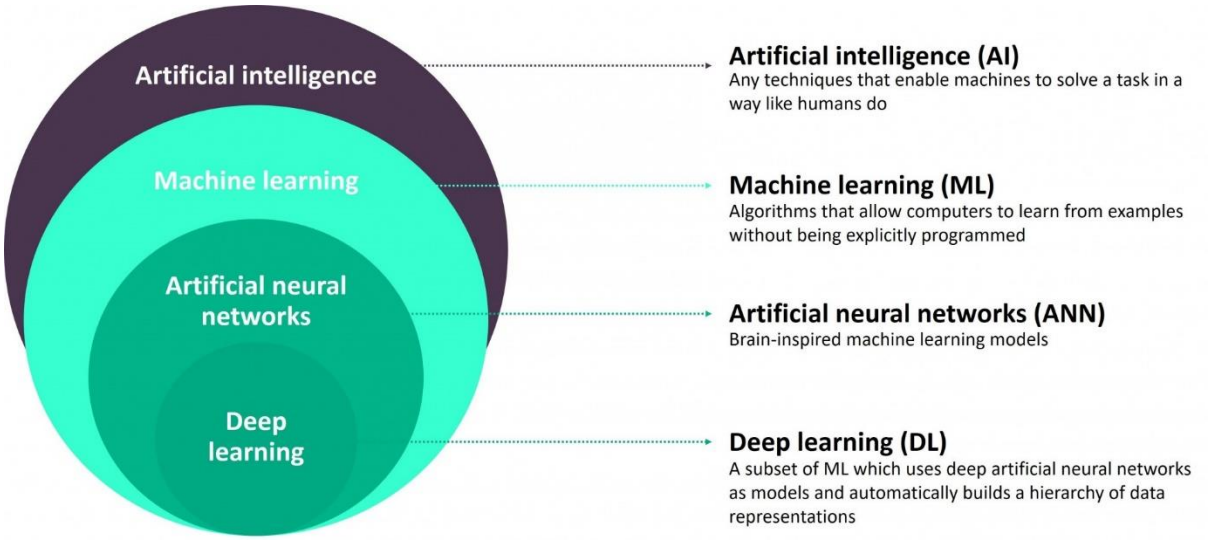
- **Retail Sector**: In the retail business, AI algorithms can combat supply-chain problems by managing inventory. Apart from predicting future trends in the apparel business, AI can also help forecast demand and enhance customer experience through real-time data analysis.

- **Automobile Industry**: An autonomous or self-driving car is the latest research area. Several carmakers have already used AI features such as voice-control, lane-switch, collision-detection, and improved driver safety.

- **Gaming**: AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

- **Robots**: AI is arguably the most exciting field in robotics. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence.

APPLICATION OF AI ON DIFFERENT LEARNING FIELDS:



[Image source: Internet]

Machine learning (ML):

It is the study of computer algorithms that can improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so.

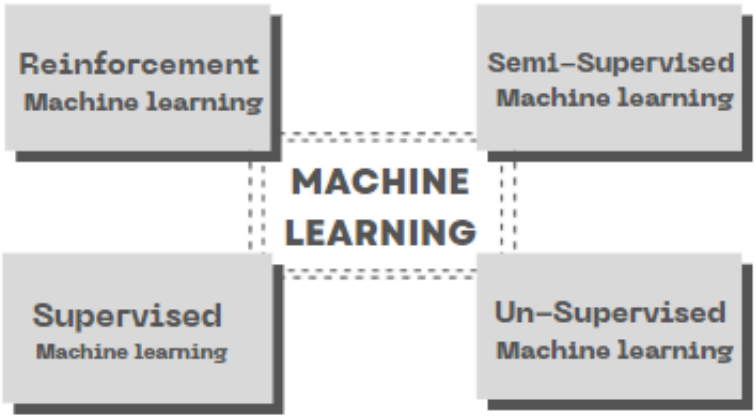
Artificial neural networks (ANN):

An artificial neural network (ANN) is a computational model that mimics the way nerve cells work in the human brain. It is designed to simulate the way the human brain analyzes and processes information.

Deep learning:

It is also known as deep structured learning and is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher-level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

INTRODUCTION TO MACHINE LEARNING



[Image source: Internet]

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly.

- **Supervised Learning:** Supervised learning is an approach in creating artificial intelligence (AI), where a computer algorithm is trained on input data that has been labelled for a particular output. The model is trained until it can detect the underlying patterns and relationships between the input data and the output labels, enabling it to yield accurate labelling results when presented with never-before-seen data.

  During its training phase, the system is fed with labelled data sets, which instruct the system what output is related to each specific input value. The trained model is then presented with test data: This is data that has been labelled, but the labels have not been revealed to the algorithm. The aim of the testing data is to measure how accurately the algorithm will perform on unlabelled data.

- **Un-Supervised Learning:** Unsupervised learning refers to the use of artificial intelligence (AI) algorithms to identify patterns in data sets containing data points that are neither classified nor labeled. Thus, the algorithm classify, label and group the data points contained within the data sets without having any external guidance in performing that task.

  Unsupervised learning can be more unpredictable than a supervised learning model. While an unsupervised learning AI system might, for example, figure out on its own how to sort A from B, it might also add unforeseen and undesired categories to deal with unusual breeds, creating clutter instead of order.

- **Semi-Supervised Learning:** Semi-supervised machine learning is a combination of supervised and unsupervised learning. It uses a small amount of labeled data and a large amount of unlabeled data, which provides the benefits of both unsupervised and supervised learning while avoiding the challenges of finding a large amount of labeled data. That means you can train a model to label data without having to use as much labeled training data.

- **Reinforcement Learning:** Reinforcement learning is the training of machine learning models to make a sequence of decision. The agent learns to achieve a goal in an uncertain, potentially complex environment. In this learning, an artificial intelligence faces a game-like situation. The computer employs trial and error to come up with a solution to the problem. To get the machine to do what the programmer wants, the artificial intelligence gets either rewards or penalties for the actions it performs. Its goal is to maximize the total reward.

NUMPY

NumPy is a Python package that stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object. The reason to choose python NumPy array is that it occupies less memory as compared to list. It is fast in terms of execution and at the same time, it is very convenient to work with NumPy. These are some major advantages that Python NumPy array has over list. Using NumPy, mathematical and logical operations on arrays can be performed.

- **NumPy Array:** A NumPy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along with its each dimension.

  A NumPy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along with its each dimension.

- **Indexing/Slicing:** Similar to Python lists, NumPy arrays also can be sliced. Since arrays may be multidimensional, you must specify a slice for each dimension of the array.

  Example: Slice elements from index 1 to index 5 from the following

  array:

  ```
  import numpy as np

  arr = np.array([1, 2, 3, 4, 5, 6, 7])

  print(arr[1:5])
  ```

  (The result includes the start index, but excludes the end index.)

PANDAS

Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package name Numpy, which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other data Science modules inside the Python ecosystem.

Following are some features of Pandas:

- **Handling of data:**
  The Pandas library provides a really fast and efficient way to manage and explore data. It does that by providing us with Series and Data Frames, which help us not only to represent data efficiently but also manipulate it in various ways.

- **Alignment and indexing:**
  Having data is useless if you don't know where it belongs and what it tells us about. Therefore, labeling of data is of utmost importance. Another important factor is an organization, without which data would be impossible to read. These two needs: Organization and labeling of data are taken care of by the methods of alignment and indexing, which can be found within Pandas.

- **Handling missing data:**
  Sometimes data are very crude in nature and one of the many problems associated with data is the occurrence of missing data or value. Therefore, it is pertinent to handle the missing values properly so that they do not adulterate our study results. Some Pandas features have you covered on this end because handling missing values is integrated within the library.

- **Cleaning up data:**
  Some Data are messy, so much so that performing any analysis over such data would lead to severely wrong results. Thus, it is of extreme importance that we clean our data up, and this Pandas feature is easily provided. They help a lot to not only make the code clean but also tidies up the data so that even the normal eye can decipher parts of the data. The cleaner the data, the better the result.

- **Input and output tools:**
  Pandas provide a wide array of built-in tools for the purpose of reading and writing data. While analysing we need to read and write data into data structures, web service, databases, etc. This has been made extremely simple with the help of Panda's inbuilt tools. In other languages, it would probably take a lot of code to generate the same results, which would only slow down the process of analysing.

- **Multiple file formats supported:**
  Data these days can be found in so many different file formats, that it becomes crucial that libraries used for data analysis can read various file formats. Pandas aces this sector with a huge scope of file formats supported. Whether it is a JSON or CSV, Pandas can support it all, including Excel and HDF5. This can be considered as one of the most appealing Python Pandas features.

- **Merging and joining of datasets:**
  While analysing data we constantly need to merge and join multiple datasets to create a final dataset to be able to properly analyse it. This is important because if the datasets aren't merged or joined properly, then it is going to affect the results adversely and we do not want that. Pandas can help to merge various datasets, with extreme efficiency so that we don't face any problems while analysing the data.

- **A lot of time series:**
  These Pandas features won't make sense to beginners right away, but they will be of great use in the future. These features include the likes of moving window statistics and frequency conversion. So, as we go deeper into learning Pandas we will see how essential and useful these features are, for a data scientist.

- **Optimized performance:**
  Pandas is said to have a really optimized performance, which makes it really fast and suitable for data science. The critical code for Pandas is written in C or Python, which makes it extremely responsive and fast.

- **Python support:**
  This feature of Pandas is the deal closer. Huge number of helpful libraries at our disposal Python has become one of the most sought-after programming languages for data analysis. Thus, Pandas being a part of Python and allowing us to access the other libraries like NumPy and MatPlotLib.

- **Visualize:**
  Visualizing the data is an important part of data science. It is what make the results of the study understandable by human eyes. Pandas have an in-built ability to help you plot your data and see the various kinds of graphs formed. Without visualization, data analysis would make no sense to most of the population.

SCIKIT-LEARN

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. Some of the most popular groups of models provided by Sklearn are as follows:

- **Supervised Learning algorithms**:

  Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.

- **Unsupervised Learning algorithms**:

  On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

- **Clustering**:

  This model is used for grouping unlabelled data.

- **Cross Validation**:

  It is used to check the accuracy of supervised models on unseen data.

- **Dimensionality Reduction**:

  It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.

- **Ensemble methods:**

  As name suggest, it is used for combining the predictions of multiple supervised models.

- **Feature extraction:**

  It is used to extract the features from data to define the attributes in image and text data.

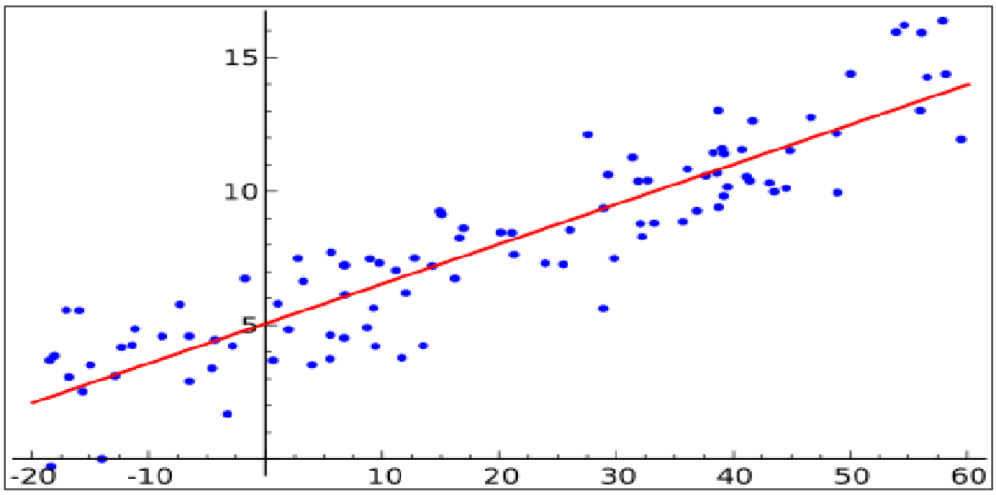
- **Feature selection:**

  It is used to identify useful attributes to create supervised models.

- **Open Source:**

  It is open-source library and also commercially usable under BSD license.

REGRESSION ANALYSIS



- In statistical modelling, regression analysis is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors').

- More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

- Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. It is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships.

LINEAR REGRESSION

- Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression.

- In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

LOGISTIC-REGRESSION

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. It is used for solving the classification problems.

- It predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. But instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
  Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

POLYNOMIAL REGRESSION

- Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an n^{th} degree polynomial in x.

- Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y, denoted E(y | x), and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.

K-NEIGHBOUR

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).

Some of its key points are mentioned below:

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- It stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

GAUSSIAN NAIVE BAYES

As the name suggest, Gaussian Naïve Bayes classifier assumes that the data from each label is drawn from a simple Gaussian distribution.

The Scikit-learn provides sklearn.naive_bayes. GaussianNB to implement the Gaussian Naive Bayes algorithm for classification.

Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data.

This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all what is needed to define such a distribution.

DECISION TREE CLASSIFIER ALGORITHM

• Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

• In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. The decisions or the test are performed on the basis of features of the given dataset. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure

A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

RANDOM FOREST CLASSIFIER

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Each individual tree in the random forest spits out a class prediction and the class with the most votes become our model's prediction. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

Uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. For classification tasks, the output of the random forest is the class selected by most trees. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. It is capable of handling large datasets with high dimensionality.

MATPLOTLIB

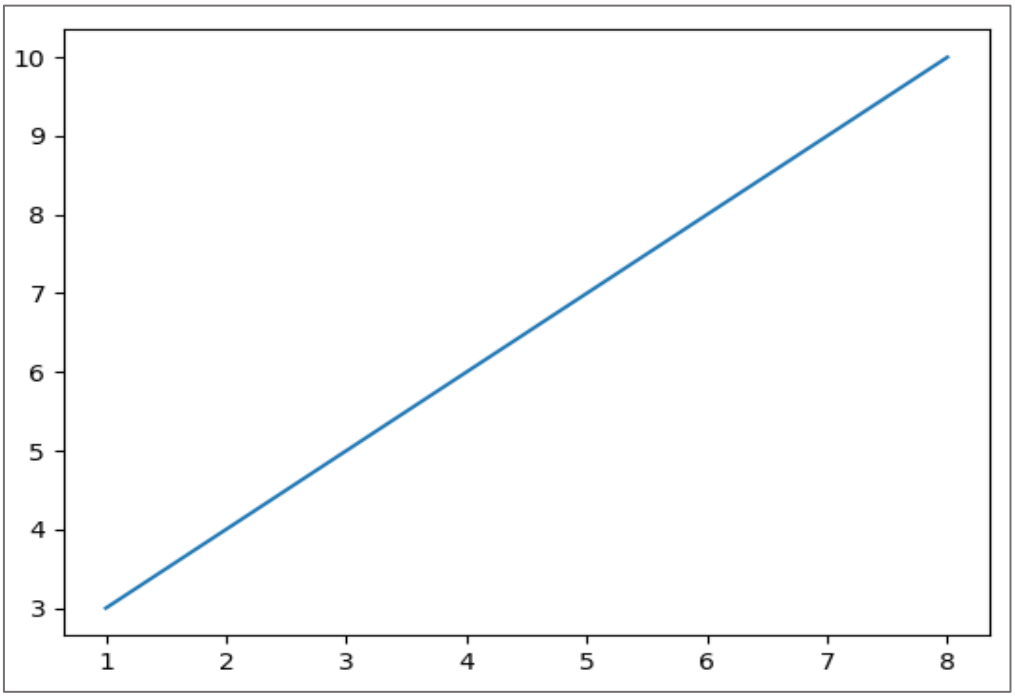
Matplotlib is a visualization library in Python for 2D plots of arrays. It is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Below are some examples:

- **LINE PLOT**

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints, ypoints)
plt.show()
```

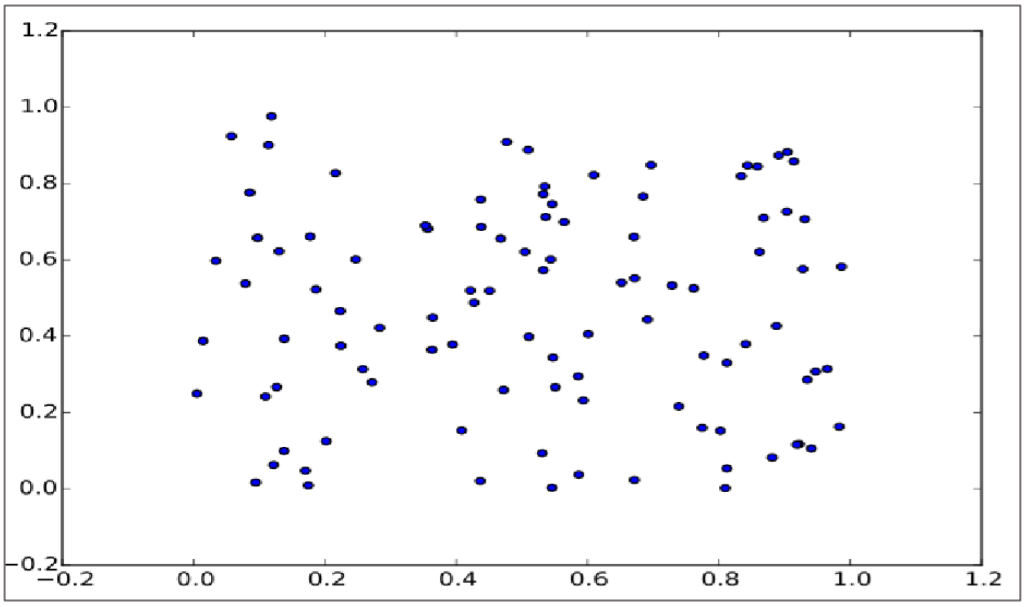
OUTPUT:



- **SCATTER PLOT**

```
>>> import matplotlib.pyplot as plt
>>> from numpy.random import rand
>>> a = rand(100)
>>> b = rand(100)
>>> plt.scatter(a, b)
>>> plt.show ()
```

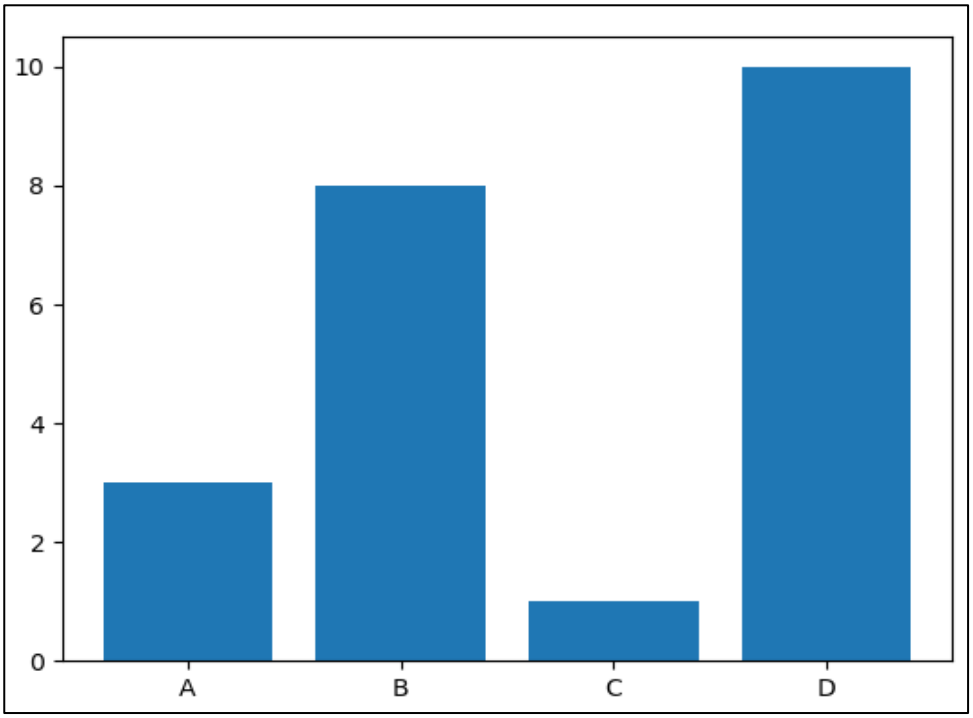
OUTPUT:



BAR CHART

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.bar(x,y)
plt.show()
```

OUTPUT:



MALICIOUS URL PREDICTION

1) Explanation: Importing the *numpy, pandas, matplotlib, label-encoder, seaborn* and different packages and modules of Python. We are using read_csv() and head() functions from pandas to read and display the dataset respectively.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from tld import get_tld
from urllib.parse import urlparse
import re
import seaborn as sns
```

Python

```python
url_data = pd.read_csv('./malicious.csv')
url_data.head()
```

Python

	url	type
0	br-icloud.com.br	phishing
1	mp3raid.com/music/krizz_kaliko.html	benign
2	bopsecrets.org/rexroth/cr/1.htm	benign
3	http://www.garage-pirenne.be/index.php?option=...	defacement
4	http://adventure-nicaragua.net/index.php?optio...	defacement

2) Explanation: The code below gives us the shape and information of the dataset, i.e., it has 651191 rows and 2 columns, all non-null.

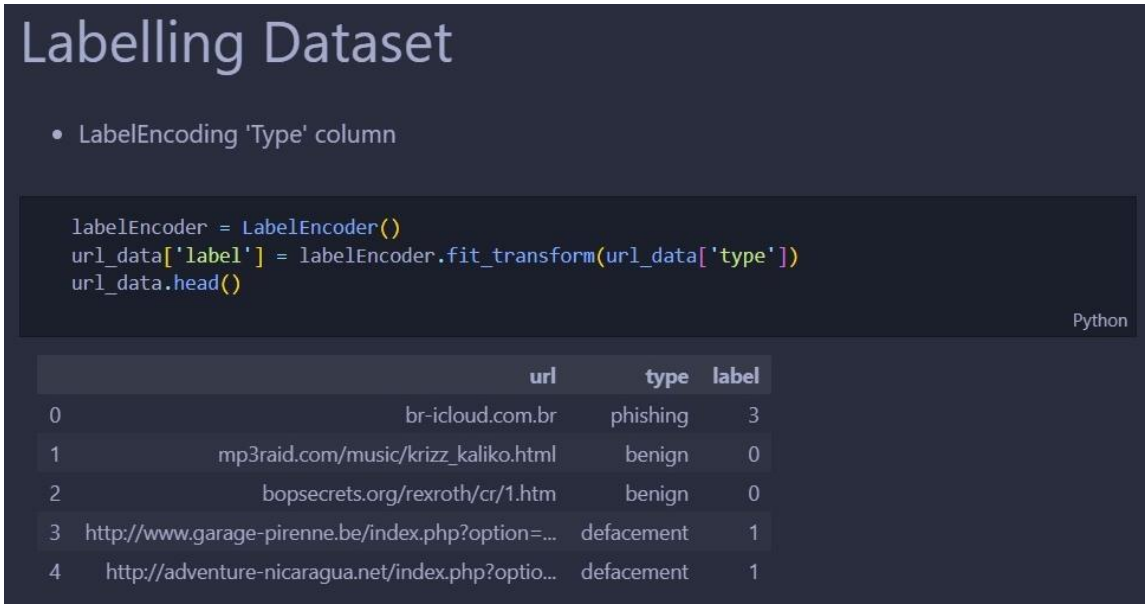
Dataset Information

```python
print("Shape of dataset: ", url_data.shape, "\n")
url_data.info()
print("\nTotal no. of Null Values: \n", url_data.isnull().sum())
```

Python

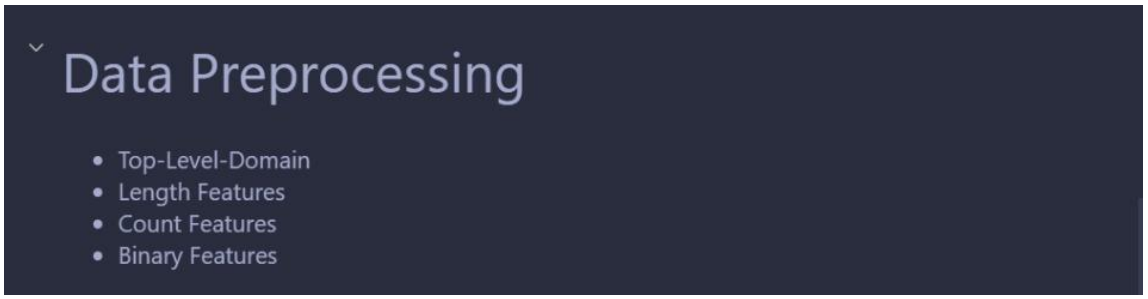
```
Shape of dataset:  (651191, 2)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 651191 entries, 0 to 651190
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   url     651191 non-null  object
 1   type    651191 non-null  object
dtypes: object(2)
memory usage: 9.9+ MB

Total no. of Null Values:
 url     0
type    0
dtype: int64
```

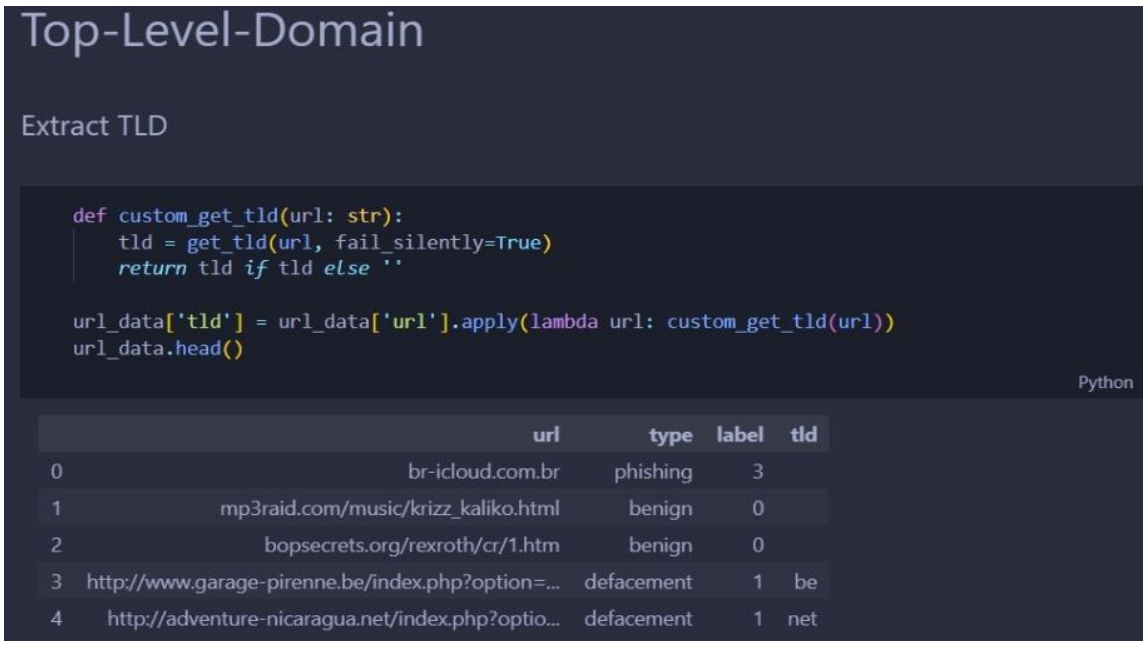
3) <u>Explanation:</u> The different types of URLs input have been labelled with different numbers. For example- defacement is labelled as 1, benign as 0.



	url	type	label
0	br-icloud.com.br	phishing	3
1	mp3raid.com/music/krizz_kaliko.html	benign	0
2	bopsecrets.org/rexroth/cr/1.htm	benign	0
3	http://www.garage-pirenne.be/index.php?option=...	defacement	1
4	http://adventure-nicaragua.net/index.php?optio...	defacement	1

4) <u>Explanation:</u> Data preprocessing(feature extraction) by extracting the below mentioned features. We are extracting a column for the top-level-domain of the URL.





	url	type	label	tld
0	br-icloud.com.br	phishing	3	
1	mp3raid.com/music/krizz_kaliko.html	benign	0	
2	bopsecrets.org/rexroth/cr/1.htm	benign	0	
3	http://www.garage-pirenne.be/index.php?option=...	defacement	1	be
4	http://adventure-nicaragua.net/index.php?optio...	defacement	1	net

5) <u>Explanation:</u> After extracting the TLD coloumn, we are Label Encoding it in the coloumn named 'tld_vector'.

Label Encoding TLD column

```
tldEncoder = LabelEncoder()
url_data['tld_vector'] = tldEncoder.fit_transform(url_data['tld'])
url_data.head()
```

Python

	url	type	label	tld	tld_vector
0	br-icloud.com.br	phishing	3		0
1	mp3raid.com/music/krizz_kaliko.html	benign	0		0
2	bopsecrets.org/rexroth/cr/1.htm	benign	0		0
3	http://www.garage-pirenne.be/index.php?option=...	defacement	1	be	35
4	http://adventure-nicaragua.net/index.php?optio...	defacement	1	net	353

6) <u>Explanation:</u> Extracting the length features of the URLs. For example- length of the TLD, URL, Host Name, Path and First Directory.

Length Features

- Length of TLD
- Length of URL
- Length of HostName
- Length of Path
- Length of First Directory

Length of Top-Level-Domain

```
url_data['tld_len'] = url_data['tld'].apply(lambda tld: len(tld))
```

Python

```
# Dropping TLD column
url_data.drop(["tld"], axis = 1, inplace=True)
```

Python

Length of URL

```
url_data['url_len'] = url_data['url'].apply(lambda url: len(str(url)))
```

Python

Length of Hostname

```
url_data['host_len'] = url_data['url'].apply(lambda url: len(urlparse(url).netloc))
```

Python

Length of Path

```
url_data['path_len'] = url_data['url'].apply(lambda url: len(urlparse(url).path))
```

Python

Length of First Directory

```
def fd_length(url):
    urlpath= urlparse(url).path
    try:
        return len(urlpath.split('/')[1])
    except:
        return 0

url_data['fd_len'] = url_data['url'].apply(lambda url: fd_length(url))
```

Python

7) <u>Explanation:</u> The .head() is implemented to display the url data set which now comprises several features of length.

```
url_data.head()
```

	url	type	label	tld_vector	tld_len	url_len	host_len	path_len	fd_len
0	br-icloud.com.br	phishing	3	0	0	16	0	16	0
1	mp3raid.com/music/krizz_kaliko.html	benign	0	0	0	35	0	35	5
2	bopsecrets.org/rexroth/cr/1.htm	benign	0	0	0	31	0	31	7
3	http://www.garage-pirenne.be/index.php?option=...	defacement	1	35	2	88	21	10	9
4	http://adventure-nicaragua.net/index.php?optio...	defacement	1	353	3	235	23	10	9

8) <u>Explanation:</u> Extracting the different types of count features from the URLs. For example- Count of ‘www’, ‘http’, no. of directories and so on.

Count Features

- Count of 'www'
- Count of 'https'
- Number of directories
- Count of Digits
- count of Letters
- Count of Special Characters ('$&+_:=?@-.!<>#%^~')

Count 'www' & 'https'

```
url_data['www_count'] = url_data['url'].apply(lambda url: url.count('www'))
url_data['http_count'] = url_data['url'].apply(lambda url: url.count('http'))
```

Python

Number of Directories

```python
def count_dir(url):
    return urlparse(url).path.count('/')

url_data['dir_count']= url_data['url'].apply(lambda url: count_dir(url))
```

Python

Count Letters

```python
def countLetters(url: str):
    c = 0
    for i in url:
        if i.isalpha(): c=c+1
    return c

url_data['letters_count'] = url_data['url'].apply(lambda url: countLetters(url))
```

Python

Count Digits

```python
def countDigits(url: str):
    c = 0
    for i in url:
        if i.isnumeric(): c=c+1
    return c

url_data['digits_count'] = url_data['url'].apply(lambda url: countDigits(url))
```

Python

Count Special Characters ($&+_:=?@-.!<>#%^~)

```python
def countSpclChars(url: str, charset: str):
    c = 0
    for i in url:
        if i in charset: c = c+1
    return c

url_data['spec_chars_count'] = url_data['url'].apply(lambda url: countSpclChars(url, '$&+_:=?@-.!<>#%^~'))
```

Python

Count Specific Special Characters('&.?-') each in the url

```python
url_data['count&'] = url_data['url'].apply(lambda url: url.count('&'))
url_data['count.'] = url_data['url'].apply(lambda url: url.count('.'))
url_data['count?'] = url_data['url'].apply(lambda url: url.count('?'))
url_data['count-'] = url_data['url'].apply(lambda url: url.count('-'))
```

Python

9) <u>Explanation:</u> The .head() is implemented to display the url data set which now comprises several features of length and count.

```
url_data.head()
```

	url	type	label	tld_vector	tld_len	url_len	host_len	path_len	fd_len
0	br-icloud.com.br	phishing	3	0	0	16	0	16	0
1	mp3raid.com/music/krizz_kaliko.html	benign	0	0	0	35	0	35	5
2	bopsecrets.org/rexroth/cr/1.htm	benign	0	0	0	31	0	31	7
3	http://www.garage-pirenne.be/index.php?option=...	defacement	1	35	2	88	21	10	9
4	http://adventure-nicaragua.net/index.php?optio...	defacement	1	353	3	235	23	10	9

www_count	http_count	dir_count	letters_count	digits_count	spec_chars_count	count&	count.	count?	count-
0	0	0	13	0	3	0	2	0	1
0	0	2	29	1	3	0	2	0	0
0	0	3	25	1	2	0	2	0	0
1	1	1	63	7	15	3	3	1	1
0	1	1	199	22	11	2	2	1	1

10) <u>Explanation:</u> In the code given below, we are checking the URL is an IP address or not and creating a new column 'is_ip'. 0 means the URL is not an IP address and 1 means it is.

Binary Features

- url domain is ip address or not

```
def isIP(url: str):
    match = re.match('^((25[0-5]|(2[0-4]|1\d|[1-9]|)\d)(\.(?!$)|$)){4}$', url)
    return 1 if match else 0

url_data['is_ip'] = url_data['url'].apply(lambda url: isIP(str(urlparse(url).hostname)))
```

```
url_data.head(5)
```

	url	type	label	tld_vector	tld_len	url_len	host_len	path_len	fd_len
0	br-icloud.com.br	phishing	3	0	0	16	0	16	0
1	mp3raid.com/music/krizz_kaliko.html	benign	0	0	0	35	0	35	5
2	bopsecrets.org/rexroth/cr/1.htm	benign	0	0	0	31	0	31	7
3	http://www.garage-pirenne.be/index.php?option=...	defacement	1	35	2	88	21	10	9

www_count	http_count	dir_count	letters_count	digits_count	spec_chars_count	count&	count.	count?	count-	is_ip
0	0	0	13	0	3	0	2	0	1	0
0	0	2	29	1	3	0	2	0	0	0
0	0	3	25	1	2	0	2	0	0	0
1	1	1	63	7	15	3	3	1	1	0

11) <u>Explanation:</u> Data Visualization is an important step in creating a machine learning model. For example- The first chart provided below, gives the viewers a better understanding of the data set by plotting the types of URL vs the number of URLs.

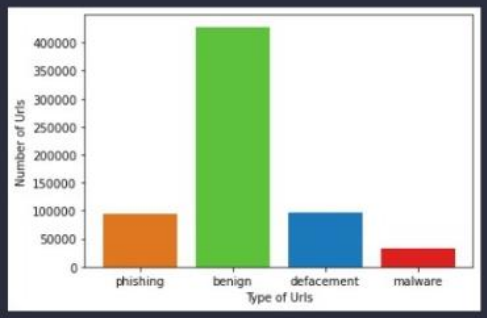
Data Visualization

Bar chart: Type of URLs vs Number of URLs

```
red_color = '#ff0000'
orange_color = '#ff7300'
green_color = '#52d726'
blue_color = '#007ed6'
color_list = [orange_color, green_color, blue_color, red_color]

sns.countplot(x='type', data=url_data, palette=color_list)
plt.xlabel('Type of Urls')
plt.ylabel('Number of Urls')
plt.plot()
```

Python

[]



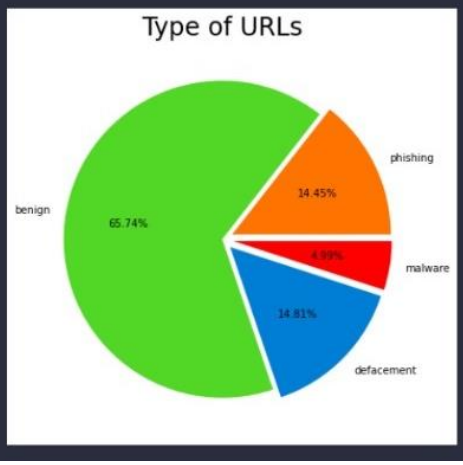
Pie chart: Type of URLs

```
type_list = url_data['type'].unique()
num_of_type = [len(url_data[url_data['type'] == type]) for type in type_list]

plt.figure(figsize=(7,7), facecolor='#ffffff')
plt.pie(num_of_type, labels=type_list, colors=color_list, autopct='%0.2f%%', explode=(0.07, 0, 0.07, 0.07))
plt.title('Type of URLs', fontdict={'color': '#000000', 'size': 24})
plt.plot()
```

Python

... []

</>

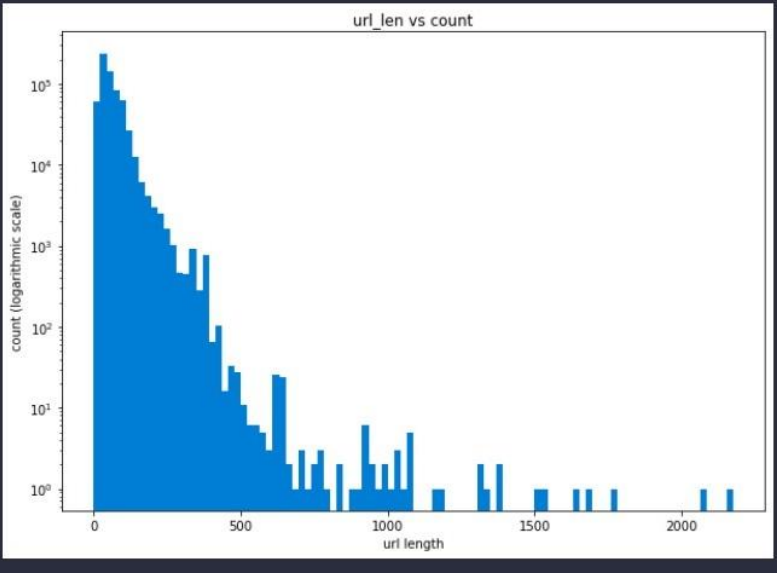


URL length vs Number of URL histogram

```
plt.figure(figsize=(10,7))
plt.hist(url_data['url_len'], color=blue_color, bins=100)

plt.title('url_len vs count')
plt.xlabel('url length')
plt.ylabel('count (logarithmic scale)')
plt.yscale('log')
plt.plot()
```

Python

[]

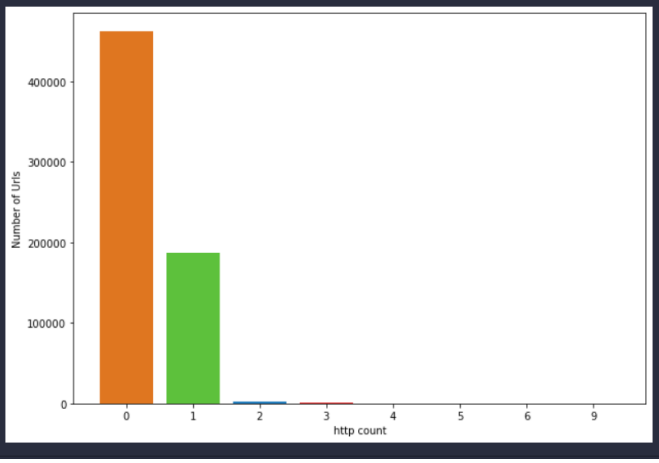


Http-count vs Number of URL histogram

```
plt.figure(figsize=(10,7))
sns.countplot(x='http_count', data=url_data, palette=color_list)
plt.xlabel('http count')
plt.ylabel('Number of Urls')
plt.plot()
```

[26] ✓ 0.5s

... []

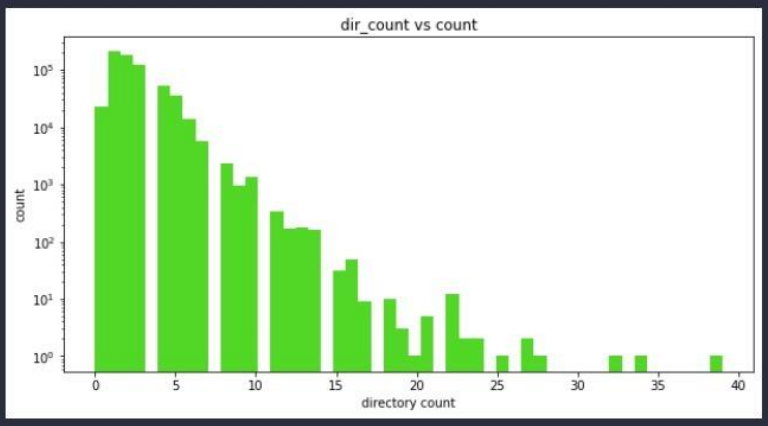


Directory-count vs Number of URL histogram

```
plt.figure(figsize=(10,5))
plt.hist(url_data['dir_count'], color=green_color, bins=50)

plt.title('dir_count vs count')
plt.xlabel('directory count')
plt.ylabel('count')
plt.yscale('log')
plt.plot()
```

Python

... []

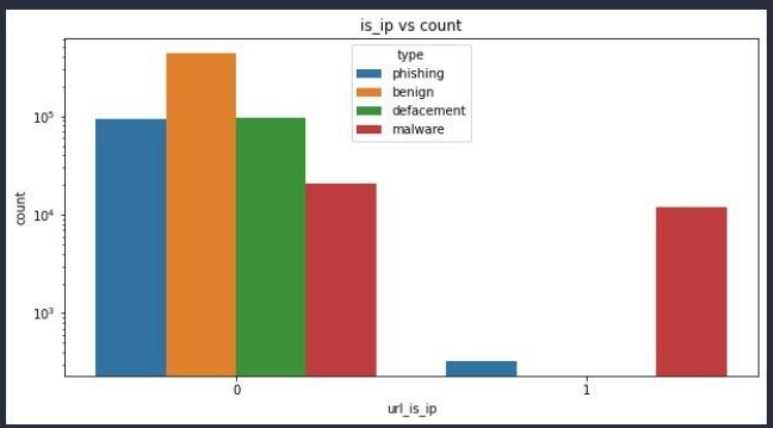


IP or not vs Number of URLs

```
plt.figure(figsize=(10,5))
sns.countplot(x = 'is_ip', data=url_data, hue='type')

plt.title('is_ip vs count')
plt.xlabel('url_is_ip')
plt.ylabel('count')
plt.yscale('log')
plt.plot()
```

Python

... []

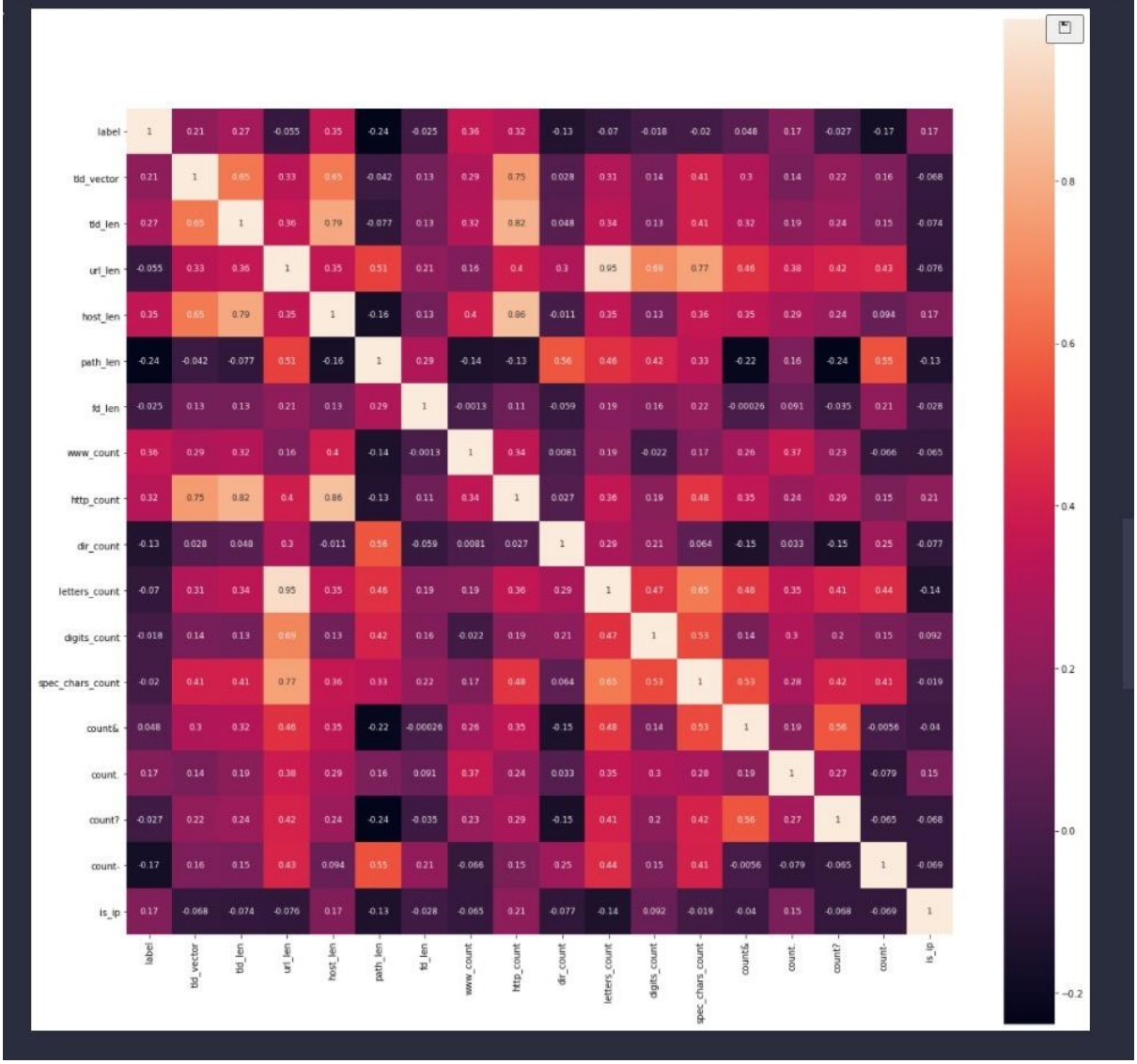


12) <u>Explanation:</u> When we implement a heap map, the model can understand the corelation between the features of the data set and thus it gets the saturation points accordingly.

Heat map of correlation matrix

```
corr_matrix = url_data.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corr_matrix, square=True, annot=True, annot_kws={'size': 9})
plt.show()
```

Python



13) <u>Explanation:</u> Calculating Mutual Information gives us the compatibility values between the features. When having a big dataset with a big range of features, mutual information helps the model to select a subset of those features in order to discard the irrelevant ones.

Mutual Information of Features vs Label

Calculating Mutual Information

```python
from sklearn.feature_selection import mutual_info_classif
mutual_info = mutual_info_classif(url_data.iloc[:, 3:], url_data['label'])
mutual_info
```

Python

```
array([0.36733965, 0.31088215, 0.10695204, 0.39292116, 0.22572274,
       0.17587208, 0.18075016, 0.32749959, 0.11406061, 0.11077246,
       0.08341143, 0.13326579, 0.10248462, 0.1504659 , 0.06449394,
       0.03454078, 0.05698507])
```

```python
mutual_info = pd.Series(mutual_info)
mutual_info.index = url_data.iloc[:, 3:].columns
mutual_info = mutual_info.sort_values(ascending = False)
print(mutual_info)
```

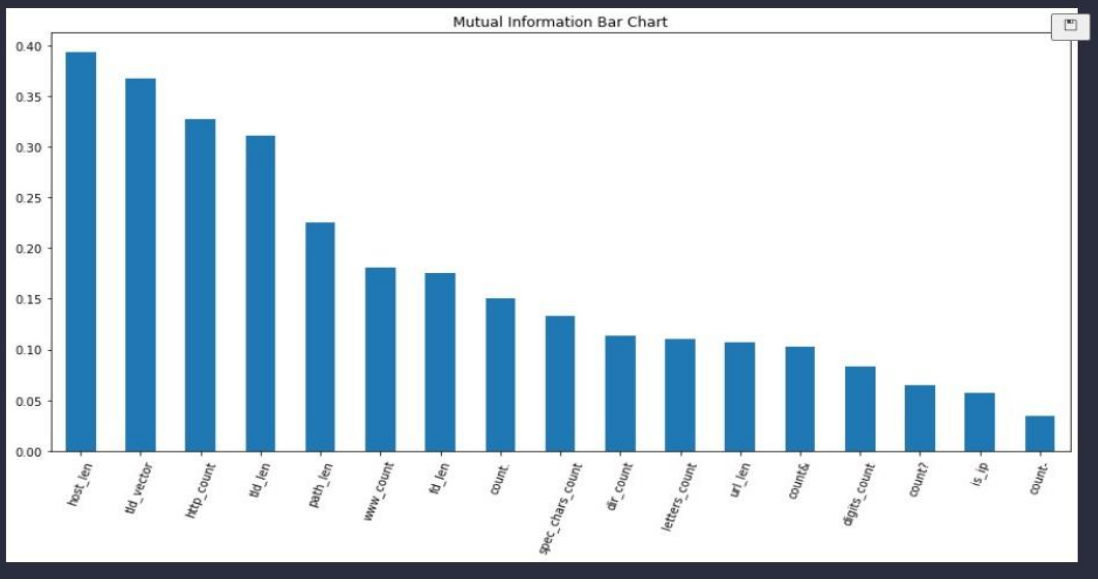
Python

```
host_len            0.392921
tld_vector          0.367340
http_count          0.327500
tld_len             0.310882
path_len            0.225723
www_count           0.180750
fd_len              0.175872
count.              0.150466
spec_chars_count    0.133266
dir_count           0.114061
letters_count       0.110772
url_len             0.106952
count&              0.102485
digits_count        0.083411
count?              0.064494
is_ip               0.056985
count-              0.034541
dtype: float64
```

Mutual Information Bar Chart

```python
plt.figure(figsize=(15,7))
mutual_info.plot.bar()
plt.title("Mutual Information Bar Chart")
plt.xticks(rotation = 70)
plt.show()
```

Python



14) <u>Explanation:</u> In the code given below, we have imported five different types of machine learning algorithms. The algorithms are *KNeighbor Classifier, Logistic Regression, GaussianNB, Decision Tree Classifier and Random Forest Classifier*. By using different machine learning algorithms, we can compare the accuracy or error between them.

Machine Learning Models: Training Testing

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, mean_squared_error

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

#965023
```

Python

15) <u>Explanation:</u> The Dataset is split into four parts for training and testing each having features dataset and output dataset. Training dataset size is given 70% of the original dataset.

Train-Test Splitting the data

```python
X = url_data.iloc[:, 3:]
y = url_data['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=42)
```

Python

16) <u>Explanation:</u> The training and testing features data set are implemented with two different feature scaling methods.

Feature Scaling

- Standardization
- Normalization

```python
sc = StandardScaler()
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)

mm = MinMaxScaler()
X_train_nml = mm.fit_transform(X_train)
X_test_nml = mm.transform(X_test)
```

Python

17) <u>Explanation:</u> After feature scaling of the data sets, the Logistic Regression algorithm is fitted into the training data set and the testing data set is then used for prediction.

LogisticRegression

```python
lr = LogisticRegression(max_iter=1000)

print("-> fit start")
lr.fit(X_train_nml, y_train)
print("-> fit complete")

print("-> prediction start")
y_pred = lr.predict(X_test_nml)
print("-> prediction complete")

lr_accuracy_score = accuracy_score(y_test, y_pred)
lr_precision_score = precision_score(y_test, y_pred, average='weighted')
lr_recall_score = recall_score(y_test, y_pred, average='weighted')
lr_f1_score = 2 * (lr_precision_score * lr_recall_score) / (lr_precision_score + lr_recall_score)

lr_rms_error = np.sqrt(mean_squared_error(y_test, y_pred))

print("\nAccuracy score with LogisticRegression:", lr_accuracy_score)
print("Precision score with LogisticRegression:", lr_precision_score)
print("Recall score with LogisticRegression:", lr_recall_score)
print("F1 score with LogisticRegression:", lr_f1_score)
print("\nconfusion Matrix: ", confusion_matrix(y_test, y_pred), sep="\n")
print("RMS Error:", lr_rms_error)
```

Python

```
... -> fit start
-> fit complete
-> prediction start
-> prediction complete
```

18) <u>Explanation:</u> After the prediction, the model returned an accuracy score of 0.852. The confusion matrix is used to represent the number of matched and unmatched cases i.e., it's a table that compares predicted values with actual values. It is important because it gives analytics like accuracy, precision and recall.

```
Accuracy score with LogisticRegression: 0.8528701153779216
Precision score with LogisticRegression: 0.8395672164714 66
Recall score with LogisticRegression: 0.8528701153779216
F1 score with LogisticRegression: 0.846171211756902

confusion Matrix:
[[124793   1336    171   2433]
 [  1327  25820    323   1222]
 [   822   1524   4906   2447]
 [ 13272   3433    433  11096]]
RMS Error: 0.9364774083897649
```

19) <u>Explanation:</u> Fitted the KNN classifier to the training data set and then the testing data set is used for the prediction.

KNeighborsClassifier

```
knn = KNeighborsClassifier( n_neighbors=1, n_jobs=-1, weights='distance')

print("-> fit start")
knn.fit(X_train_nml, y_train)
print("-> fit complete")

print("-> prediction start")
y_pred= knn.predict(X_test_nml)
print("-> prediction complete")

knn_accuracy_score = accuracy_score(y_test, y_pred)
knn_precision_score = precision_score(y_test, y_pred, average='weighted')
knn_recall_score = recall_score(y_test, y_pred, average='weighted')
knn_f1_score = 2 * (knn_precision_score * knn_recall_score) / (knn_precision_score + knn_recall_score)

knn_rms_error = np.sqrt(mean_squared_error(y_test, y_pred))

print("\nAccuracy score with KNeighborsClassifier:", knn_accuracy_score)
print("Precision score with KNeighborsClassifier:", knn_precision_score)
print("Recall score with KNeighborsClassifier:", knn_recall_score)
print("F1 score with KNeighborsClassifier:", knn_f1_score)
print("\nconfusion Matrix: ", confusion_matrix(y_test, y_pred), sep="\n")
print("RMS Error:", knn_rms_error)
```

Python

```
... -> fit start
    -> fit complete
    -> prediction start
    -> prediction complete
```

20) <u>Explanation:</u> After the prediction, the model returned an accuracy score of 0.951.

```
Accuracy score with KNeighborsClassifier: 0.9510744376989937
Precision score with KNeighborsClassifier: 0.950802268176922
Recall score with KNeighborsClassifier: 0.9510744376989937
F1 score with KNeighborsClassifier: 0.950938333463444

confusion Matrix:
[[124932     39     75   3687]
 [    36  28307     50    299]
 [    55     71   9176    397]
 [  3546    741    562  23385]]
RMS Error: 0.6025672204767119
```

21) <u>Explanation:</u> Fitted the Gaussian Naive Bayes classifier to the training data set and then the testing data set is used for the prediction.

GaussianNB

```python
nb = GaussianNB()

print("-> fit start")
nb.fit(X_train_std, y_train)
print("-> fit complete")

print("-> prediction start")
y_pred= nb.predict(X_test_std)
print("-> prediction complete")

nb_accuracy_score = accuracy_score(y_test, y_pred)
nb_precision_score = precision_score(y_test, y_pred, average='weighted')
nb_recall_score = recall_score(y_test, y_pred, average='weighted')
nb_f1_score = 2 * (nb_precision_score * nb_recall_score) / (nb_precision_score + nb_recall_score)

nb_rms_error = np.sqrt(mean_squared_error(y_test, y_pred))

print("\nAccuracy score with GaussianNB:", nb_accuracy_score)
print("Precision score with GaussianNB:", nb_precision_score)
print("Recall score with GaussianNB:", nb_recall_score)
print("F1 score with GaussianNB:", nb_f1_score)
print("\nconfusion Matrix: ", confusion_matrix(y_test, y_pred), sep="\n")
print("RMS Error:", nb_rms_error)
```

Python

```
... -> fit start
-> fit complete
-> prediction start
-> prediction complete
```

22) <u>Explanation:</u> After the prediction, the model returned an accuracy score of 0.786.

```
Accuracy score with GaussianNB: 0.786320498776605
Precision score with GaussianNB: 0.7609214095643261
Recall score with GaussianNB: 0.786320498776605
F1 score with GaussianNB: 0.7734124820080535

confusion Matrix:
[[119434   8078    368    853]
 [    60  28566     41     25]
 [   458   4440   4740     61]
 [ 20960   6201    199    874]]
RMS Error: 1.1023726144526722
```

23) <u>Explanation:</u> In the code given below the Decision Tree classifier is fitted into the training data set and then used the testing data set for the prediction.

DecisionTreeClassifier

```
dt = DecisionTreeClassifier(random_state=42)

print("-> fit start")
dt.fit(X_train_nml, y_train)
print("-> fit complete")

print("-> prediction start")
y_pred= dt.predict(X_test_nml)
print("-> prediction complete")

dt_accuracy_score = accuracy_score(y_test, y_pred)
dt_precision_score = precision_score(y_test, y_pred, average='weighted')
dt_recall_score = recall_score(y_test, y_pred, average='weighted')
dt_f1_score = 2 * (dt_precision_score * dt_recall_score) / (dt_precision_score + dt_recall_score)

dt_rms_error = np.sqrt(mean_squared_error(y_test, y_pred))

print("\nAccuracy score with DesicionTreeClassifier:", dt_accuracy_score)
print("Precision score with DesicionTreeClassifier:", dt_precision_score)
print("Recall score with DesicionTreeClassifier:", dt_recall_score)
print("F1 score with DesicionTreeClassifier:", dt_f1_score)
print("\nconfusion Matrix: ", confusion_matrix(y_test, y_pred), sep="\n")
print("RMS Error:", dt_rms_error)
```

Python

```
... -> fit start
    -> fit complete
    -> prediction start
    -> prediction complete
```

24) <u>Explanation:</u> After the prediction, the model returned an accuracy score of 0.956.

```
Accuracy score with DesicionTreeClassifier: 0.9563672846773615
Precision score with DesicionTreeClassifier: 0.9558238490440104
Recall score with DesicionTreeClassifier: 0.9563672846773615
F1 score with DesicionTreeClassifier: 0.9560954896397736

confusion Matrix:
[[125929     35     60   2709]
 [    40  28143     89    420]
 [    59     76   9114    450]
 [  3676    601    309  23648]]
RMS Error: 0.5679856735026977
```

25) <u>Explanation:</u> Fitted the Random Forest Classifier to the training data set and then the testing data set is used for the prediction.

RandomForestClassifier

```
rf = RandomForestClassifier(n_estimators = 700)

print("-> fit start")
rf.fit(X_train_nml, y_train)
print("-> fit complete")

print("-> prediction start")
y_pred= rf.predict(X_test_nml)
print("-> prediction complete")

rf_accuracy_score = accuracy_score(y_test, y_pred)
rf_precision_score = precision_score(y_test, y_pred, average='weighted')
rf_recall_score = recall_score(y_test, y_pred, average='weighted')
rf_f1_score = 2 * (rf_precision_score * rf_recall_score) / (rf_precision_score + rf_recall_score)

rf_rms_error = np.sqrt(mean_squared_error(y_test, y_pred))

print("\nAccuracy score with RandomForestClassifier:", rf_accuracy_score)
print("Precision score with RandomForestClassifier:", rf_precision_score)
print("Recall score with RandomForestClassifier:", rf_recall_score)
print("F1 score with RandomForestClassifier:", rf_f1_score)
print("\nconfusion Matrix: ", confusion_matrix(y_test, y_pred), sep="\n")
print("RMS Error:", rf_rms_error)
```

Python

```
-> fit start
-> fit complete
-> prediction start
-> prediction complete
```

26) <u>Explanation:</u> After the prediction, the model returned an accuracy score of 0.965.

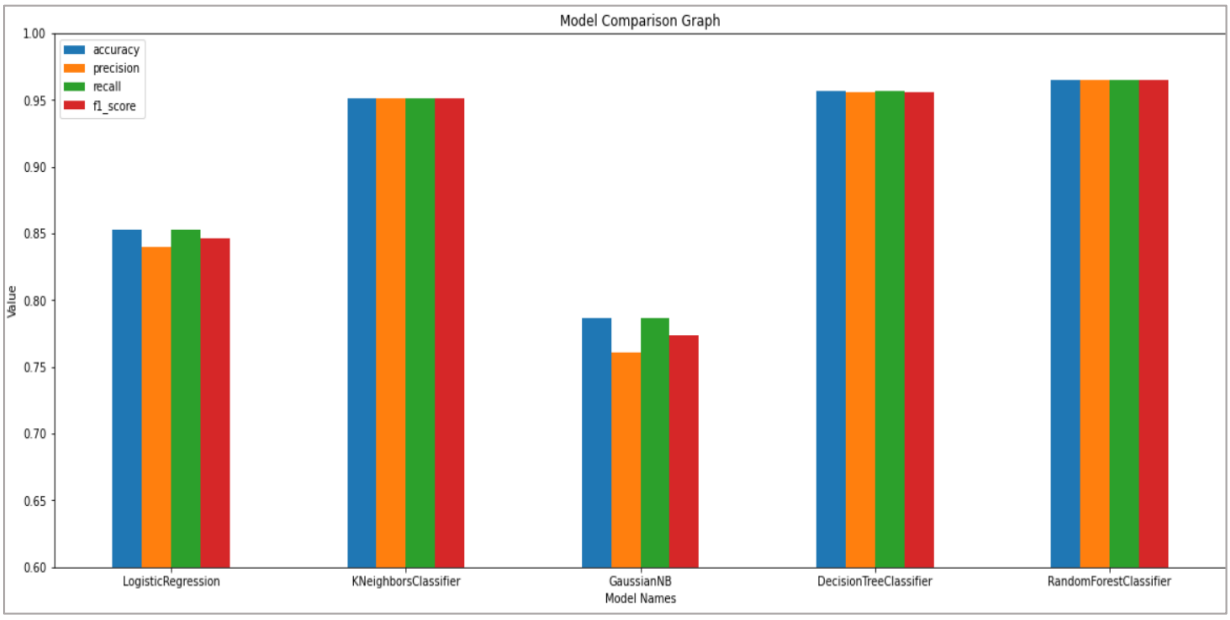
```
Accuracy score with RandomForestClassifier: 0.9651921088463231
Precision score with RandomForestClassifier: 0.9647421463643828
Recall score with RandomForestClassifier: 0.9651921088463231
F1 score with RandomForestClassifier: 0.9649670751511736

confusion Matrix:
[[126652      9     23   2049]
 [    36  28487     12    157]
 [    57     71   9120    451]
 [  3327    531     77  24299]]
RMS Error: 0.5164797165598152
```

27) <u>Explanation:</u> Calculated the Accuracy, Precision, Recall and F1_Score rate among the five different algorithms used above.

Accuracy, Precision, Recall, F1_Score Comparison Graph

```
modelData = pd.DataFrame({
    'model_name': ['LogisticRegression', 'KNeighborsClassifier', 'GaussianNB', 'DecisionTreeClassifier', 'RandomForestClassifier'],
    'accuracy': [lr_accuracy_score,knn_accuracy_score, nb_accuracy_score, dt_accuracy_score, rf_accuracy_score],
    'precision': [lr_precision_score, knn_precision_score, nb_precision_score, dt_precision_score, rf_precision_score],
    'recall': [lr_recall_score, knn_recall_score, nb_recall_score, dt_recall_score, rf_recall_score],
    'f1_score': [lr_f1_score, knn_f1_score, nb_f1_score, dt_f1_score, rf_f1_score],
    'rms_error': [lr_rms_error, knn_rms_error, nb_rms_error, dt_rms_error, rf_rms_error]
})
print(modelData.set_index('model_name', inplace=True))

modelData.iloc[:, 0:-1].plot(kind='bar', figsize=(20,8))
plt.title('Model Comparison Graph')
plt.xlabel('Model Names')
plt.ylabel('Value')
plt.ylim(bottom = 0.6, top = 1.0)
plt.xticks(rotation=0)
plt.legend(loc="upper left")
plt.show()
```

28) <u>Explanation:</u> The Model Comparison Graph is plotted on a bar chart which shows the Random Forest Classifier gives the highest accuracy rate.



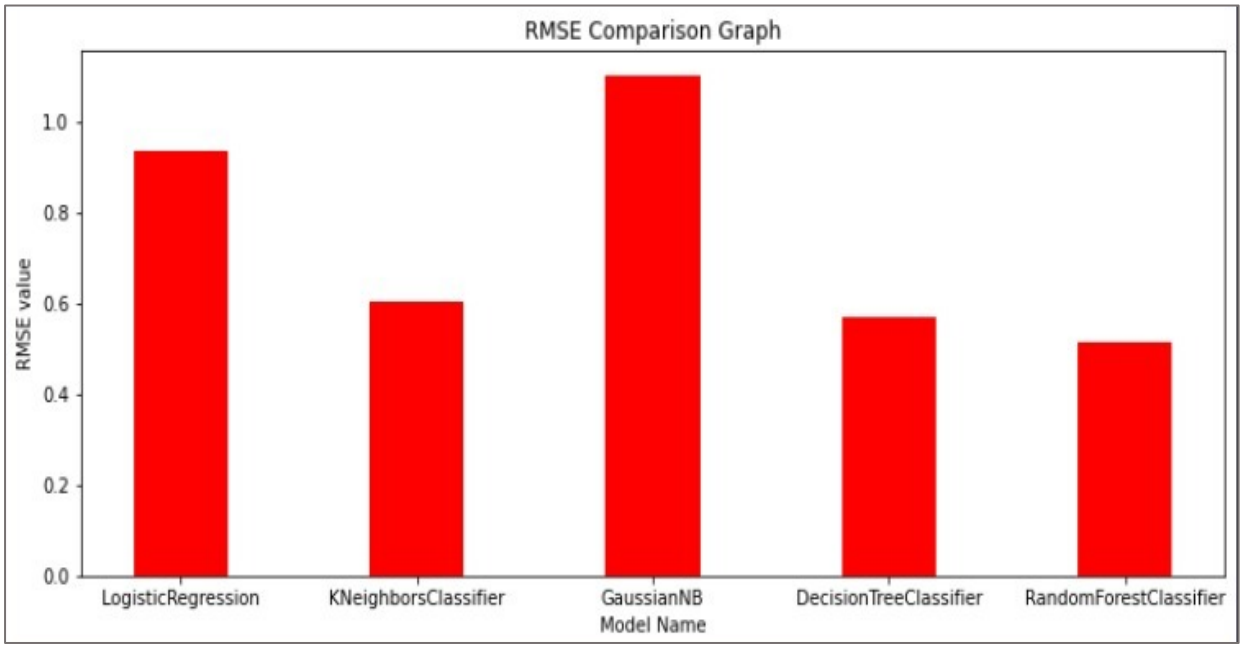
28) <u>Explanation:</u> Comparing the RMSE values of different machine learning algorithms used above.

RMS Error Comparison Graph

```
plt.figure(figsize=(12, 5))
plt.bar(modelData.index.values, modelData['rms_error'].values, color=red_color, width = 0.4)
plt.title('RMSE Comparison Graph')
plt.xlabel('Model Name')
plt.ylabel('RMSE value')
plt.show()
```

Python

29) <u>Explanation:</u> The Root Mean Square Error Comparison Graph is plotted on a bar chart which shows the GaussianNB algorithm gives the highest RMSE value and the Random Forest Classifier gives the lowest RMSE value.



CONCLUSION

In this project, we conclude that the model can predict the type of an URL on the basis of its features. The dataset on which we worked consisted different types of URL, for example- Benign, Phishing, Defacement and Malware. These predictions are done by various Machine learning algorithms which we performed using Python. Among all the algorithms we used, the Gaussian Naive Bayes returned the lowest accuracy value and highest RMSE. Logistic Regression also had comparatively low accuracy score and high RMSE value as this is a classification problem in the first place and don't go well with regression algorithms.

It was found that the Random Forest Classifier predicted the Highest accuracy rate, 0.965 which also means it had the lowest RMSE value, which makes it the most desired classification algorithm to work upon this huge data set.

FUTURE SCOPE

The Malicious URL Prediction model's efficiency can also increase if we implement some more data into the data set and use different data pre-processing techniques, as the working of a model depends largely upon the quality of the data set that is used while training the model. As the technology is improving, with every passing day it is taking cyber-attacks a step forward. The model's improved and updated version can be implemented in various browsers and applications which can make the users aware, if found any suspected malicious URL.