Practical Journal

# APPLIED ARTIFICIAL INTELLIGENCE

# MACHINE LEARNING

# SECURITY OPERATION CENTRE

**A Practical Report**

Submitted in partial fulfilment of the
Requirements for the award of the Degree of

**MASTER OF SCIENCE (INFORMATION TECHNOLOGY)**
**Part II – SEM III**

## Submitted by


## VIRAJ JOSHI

## Department of Information Technology

**Shri Vile Parle Kelavani Mandal's**

**USHA PRAVIN GANDHI COLLEGE OF ARTS, SCIENCE AND COMMERCE**

**NAAC Reaccredited 'A+' Grade**

*(Autonomous Affiliated to University of Mumbai)*

**MUMBAI, 400056**


## 2024-25

Practical Journal

# APPLIED ARTIFICIAL INTELLIGENCE

# MACHINE LEARNING

# SECURITY OPERATION CENTRE

**A Practical Report**

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

**MASTER OF SCIENCE (INFORMATION TECHNOLOGY)**

**Part II – SEM III**

## Submitted by


## VIRAJ JOSHI - 53004230033

## Department of Information Technology

**Shri Vile Parle Kelavani Mandal's**

**USHA PRAVIN GANDHI COLLEGE OF ARTS, SCIENCE AND COMMERCE**

**NAAC Reaccredited 'A+' Grade**

*(Autonomous Affiliated to University of Mumbai)*

**MUMBAI, 400056**


## 2024-25

# INDEX

# Practical 1

Aim:- Design an Expert system using AIML.

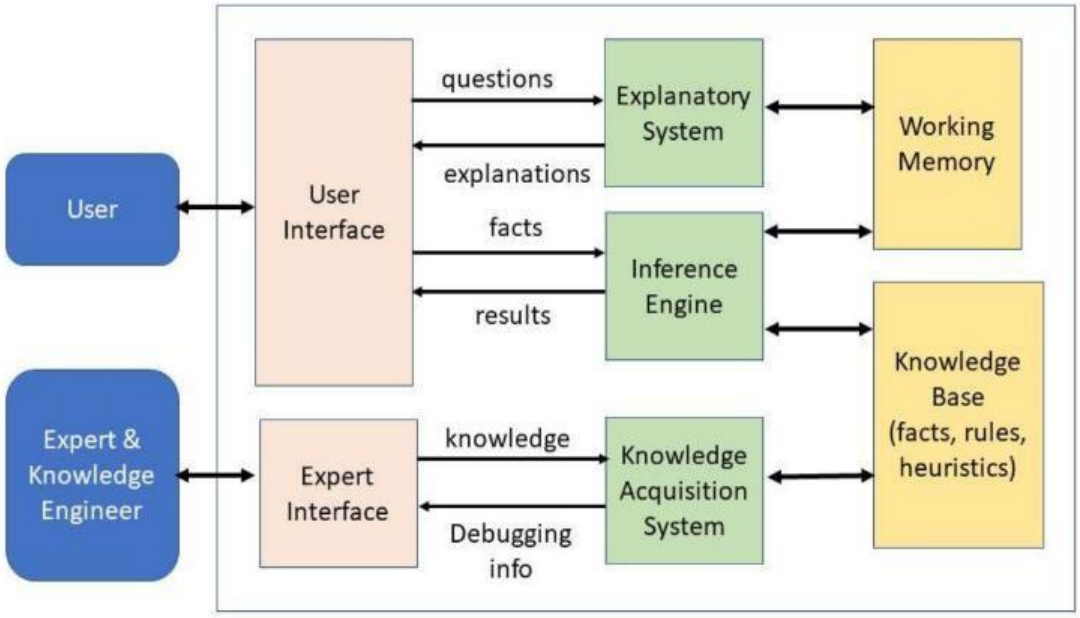
Theory:-

In an expert system there are three main components: User Interface, Inference Engine and Knowledge Base

**User Interface**: Uses various user interfaces, such as menus, graphics, and dashboards, or Natural Language Processing (NLP) to communicate with the user.

**Expert System:** A software program that makes decisions or provides advice based on databases of expert knowledge in various contexts, such as medical diagnosis. An expert system is a computer program that solves problems in a specialized field that typically calls for human expertise using techniques from artificial intelligence. A well-organized collection of data about the system's domain is called a knowledge base.

The **knowledge base's** facts are interpreted and assessed by the inference engine, which then outputs the desired results or an answer.




Code:-

Defining Flu.aiml

```
<aiml version="1.0.1" encoding="UTF-8">
  <category>
    <pattern>WHAT ARE THE SYMPTOMS OF FLU </pattern>
    <template>
```

Flu symptoms usually include fever, chills, muscle aches, cough, congestion, runny nose, headaches, and fatigue.

</template>

</category>

<category>

<pattern>I HAVE FEVER AND COUGH</pattern>

<template>

These symptoms could be associated with the flu. However, I recommend visiting a healthcare professional for an accurate diagnosis.

</template>

</category>

<category>

<pattern>IS FLU CONTAGIOUS</pattern>

<template>

Yes, flu is highly contagious and can spread easily from person to person.

</template>

</category>

<category>

<pattern>HOW CAN I PREVENT FLU</pattern>

<template>

The best way to prevent the flu is by getting a flu vaccine each year. Additionally, wash your hands frequently, avoid close contact with sick people, and maintain a healthy lifestyle.

</template>

</category>

<category>

<pattern>THANK YOU</pattern>

<template>

```
        You're welcome! Take care and stay healthy.
      </template>
    </category>

    <category>
      <pattern>BYE</pattern>
      <template>
        Goodbye! Feel free to reach out if you have more questions.
      </template>
    </category>

    <category>
      <pattern>FLU*</pattern>
      <template>
        Could you please provide more details about your symptoms so that I can assist you better?
      </template>
    </category>
</aiml>
```

Jupyter Code:-

```
import aiml

kernel = aiml.Kernel()

kernel.learn("flu.aiml")

print("Expert System for Identifying Flu Symptoms")

print("Type 'bye' to exit the conversation.")

while True:

    user_input = input("You: ")

    if user_input.lower() == "bye":

        print("System: Goodbye! Stay healthy.")

        break

    response = kernel.respond(user_input.upper())

    print(f"System: {response}")
```

Output:-

```
Loading flu.aiml...done (0.00 seconds)
Expert System for Identifying Flu Symptoms
Type 'bye' to exit the conversation.
You: What are the symptoms of Flu?
System: Flu symptoms usually include fever, chills, muscle aches, cough, congestion, runny nose, headaches, and fatigue.
You: Is flu contagious?
System: Yes, flu is highly contagious and can spread easily from person to person.
You: Thank You
System: You're welcome! Take care and stay healthy.
You: bye
System: Goodbye! Stay healthy.
Viraj Joshi - 53004230033
```

# Practical 2

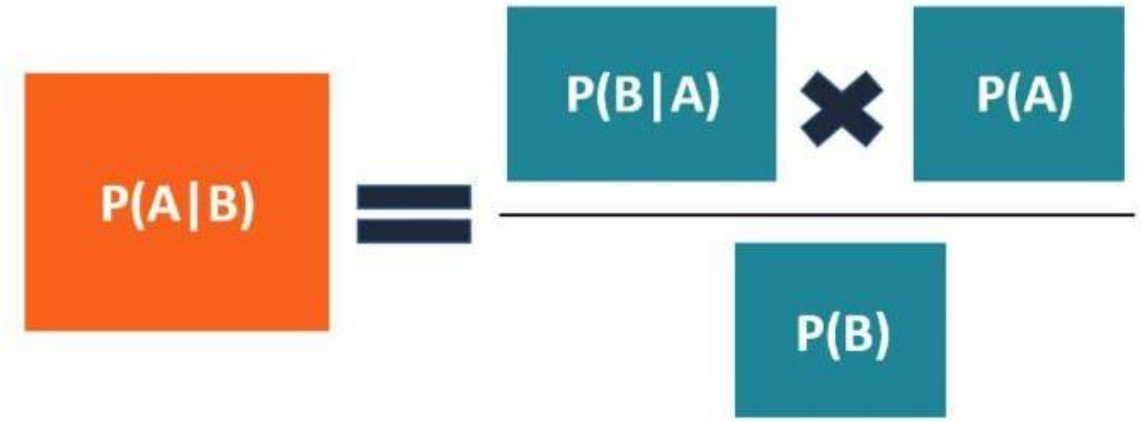
Aim:- Implement Bayes Theorem using Python.

Theory:-

Bayes' Theorem is a fundamental concept in probability theory that describes how to update the probability of a hypothesis based on new evidence. It's widely used in various fields such as medicine, finance, and machine learning.

Bayes' Theorem Formula

P(A|B) – the probability of event A occurring, given event B has occurred. P(B|A) – the probability of event B occurring, given event A has occurred. P(A) – the probability of event A.




**Events:**

A: Flower is Setosa

B: sepal length being greater than 5.0 cm

**Steps Breakdown:**

1. **Prior Probability P(A)**: The probability of the flower being setosa (without any conditions).
2. **Likelihood P(B|A)**: The probability of the sepal length being greater than 5.0 cm, given the flower is setosa.
3. **Marginal Probability P(B)**: The probability of the sepal length being greater than 5.0 cm, across the whole dataset.
4. **Posterior Probability P(A|B)**: The probability that a flower is setosa, given that its sepal length is greater than 5.0 cm.

Code:-

```
import pandas as pd

def bayes_theorem(prior_A, likelihood_B_given_A, marginal_B):
    """
    Calculate the posterior probability using Bayes' Theorem.

    :param prior_A: P(A) - Prior probability of A
    :param likelihood_B_given_A: P(B|A) - Likelihood of B given A
    :param marginal_B: P(B) - Marginal probability of B
    :return: P(A|B) - Posterior probability of A given B
    """
    return (likelihood_B_given_A * prior_A) / marginal_B


# Load the Iris dataset
def load_iris_dataset(file_path):
    return pd.read_csv(file_path)


# Calculate prior probability P(A)
def calculate_prior(data, class_col, class_value):
    return len(data[data[class_col] == class_value]) / len(data)

# Calculate likelihood P(B|A)
def calculate_likelihood(data, class_col, class_value, feature_col,
feature_condition):
    subset = data[data[class_col] == class_value]
    return len(subset[subset[feature_col] > feature_condition]) /
len(subset)

# Calculate marginal probability P(B)
def calculate_marginal(data, feature_col, feature_condition):
    return len(data[data[feature_col] > feature_condition]) / len(data)

# Apply Bayes' Theorem on the Iris dataset
def apply_bayes_to_iris(file_path, class_col, class_value, feature_col,
feature_condition):
    # Load dataset
    data = load_iris_dataset(file_path)


    # Calculate prior P(A)
```

```
    prior_A = calculate_prior(data, class_col, class_value)


    # Calculate likelihood P(B|A)
    likelihood_B_given_A = calculate_likelihood(data, class_col,
class_value, feature_col, feature_condition)


    # Calculate marginal probability P(B)
    marginal_B = calculate_marginal(data, feature_col, feature_condition)


    # Apply Bayes' Theorem
    posterior_A_given_B = bayes_theorem(prior_A, likelihood_B_given_A,
marginal_B)


    return posterior_A_given_B


# Example usage:
# Assume we want to calculate the probability P(Class='setosa' |
SepalLength > 5.0)
file_path = 'iris.csv'  # Path to the iris dataset file
class_col = 'species'  # The column representing the class (A)
class_value = 'virginica'  # The class value we're interested in (A)
feature_col = 'sepal_length'  # The feature we're using (B)
feature_condition = 5.0  # The condition on the feature (B > 5.0)


# Calculate posterior probability P(setosa|sepal_length > 5.0)
posterior_probability = apply_bayes_to_iris(file_path, class_col,
class_value, feature_col, feature_condition)
print(f"P({class_value} | {feature_col} > {feature_condition}) =
{posterior_probability:.4f}")
print()
print('Viraj Joshi - 53004230033')
```

Output:-

```
P(virginica | sepal_length > 5.0) = 0.4153

Viraj Joshi - 53004230033
```

# Practical 3

Aim:-Implement Conditional Probability and joint probability using Python.

Theory:-

Conditional Probability (P(A|B)): Conditional probability is a measure of the likelihood of event A occurring given that event B has occurred. In mathematical terms, it's defined as:

$P(A|B) = P(A \cap B) / P(B)$

where $P(A \cap B)$ represents the joint probability of events A and B occurring together.

Joint Probability ($P(A \cap B)$): The joint probability of two events A and B is a measure of how likely it is that both events will occur simultaneously. In mathematical terms, it's defined as:

$P(A \cap B) = P(A) \times P(B|A)$

Code Explanation:

Loading Penguins Dataset: The code loads the penguins dataset from a CSV file using Pandas.

Calculating Joint Probability ($P(Species \cap Island)$): The joint probability is calculated by creating a pivot table that represents the number of penguins in each species category across different island categories.

```
pivot_table = pd.crosstab(df['species'], df['island'], normalize=True)
```

This code calculates the joint probability as:

$P(Adelie \cap Dream) = P(Adelie) \times P(Dream|Adelie)$

Calculating Conditional Probability (P(Species|Island)): The conditional probability is calculated by dividing each cell in the pivot table by the sum of all cells across a specific island category.

```
conditional_probability = pivot_table.div(pivot_table.sum(axis=0), axis=1)
```

This code calculates the conditional probability as:

$P(Adelie|Dream) = P(Adelie \cap Dream) / P(Dream)$

Key Features and Assumptions:

The code assumes:

Independence: The events (species and island) are assumed to be independent, meaning that the likelihood of one event does not influence the other.

Mutual Exclusivity: The code implicitly assumes that each species category is mutually exclusive across different island categories.

Code:-

```
import pandas as pd

df = pd.read_csv('penguins.csv')

print('Viraj Joshi - 53004230033')

print("Data Preview:")

print(df.head())

pivot_table = pd.crosstab(df['species'], df['island'], normalize=True)

print("\nJoint Probability is represented in the pivot table (Species vs
Island):")

print(pivot_table)

p_adelie_given_dream = pivot_table.loc['Adelie', 'Dream']

print()

print('The joint probability of an Adelie penguin being found on Dream
Island.')

print(f"\nP(Adelie | Dream) = {p_adelie_given_dream:.4f}")


conditional_probability = pivot_table.div(pivot_table.sum(axis=0), axis=1)

print("\nConditional Probability of Species given Island:")

print(conditional_probability)

normalized=True

p_adelie_given_dream = conditional_probability.loc['Adelie', 'Dream']

print()

print('The conditional probability of an Adelie penguin being found on
Dream Island.')

print(f"\nP(Adelie | Dream) = {p_adelie_given_dream:.4f}")
```

Output:-

```
Viraj Joshi - 53004230033
Data Preview:
  species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen            39.1           18.7              181.0
1  Adelie  Torgersen            39.5           17.4              186.0
2  Adelie  Torgersen            40.3           18.0              195.0
3  Adelie  Torgersen             NaN            NaN                NaN
4  Adelie  Torgersen            36.7           19.3              193.0

   body_mass_g     sex
0       3750.0    MALE
1       3800.0  FEMALE
2       3250.0  FEMALE
3          NaN     NaN
4       3450.0  FEMALE
```

Joint Probability is represented in the pivot table (Species vs Island):

| island | Biscoe | Dream | Torgersen |
|---|---|---|---|
| species | | | |
| Adelie | 0.127907 | 0.162791 | 0.151163 |
| Chinstrap | 0.000000 | 0.197674 | 0.000000 |
| Gentoo | 0.360465 | 0.000000 | 0.000000 |

The joint probability of an Adelie penguin being found on Dream Island.

P(Adelie | Dream) = 0.1628

Conditional Probability of Species given Island:

| island | Biscoe | Dream | Torgersen |
|---|---|---|---|
| species | | | |
| Adelie | 0.261905 | 0.451613 | 1.0 |
| Chinstrap | 0.000000 | 0.548387 | 0.0 |
| Gentoo | 0.738095 | 0.000000 | 0.0 |

The conditional probability of an Adelie penguin being found on Dream Island.

P(Adelie | Dream) = 0.4516

# Practical 4

Aim:- Create a simple rule-based system in Prolog for diagnosing a common illness based on symptoms.

Theroy:-

Prolog expressions are comprised of the following truth-functional symbols, which have the same interpretation as in the predicate calculus.

Code:-

```
%Facts:Define symptoms
symptom(fever).
symptom(cough).
symptom(sore_throat).
symptom(body_aches).
symptom(runny_nose).
symptom(headache).
symptom(fatigue).
%Facts:Define possible illnesses
condition(cold).
condition(flu).
condition(strep_throat).
%Rules: Diagnosing based on the presence of symptoms
diagnose(cold):-
                symptom(runny_nose),
                symptom(cough),
                symptom(sore_throat),
                \+ symptom(fever). %Absence of fever
diagnose(flu):-
                symptom(fever),
                symptom(cough),
                symptom(body_aches),
                symptom(headache),
                symptom(fatigue).
diagnose(sterp_throat):-
                symptom(sore_throat),
                symptom(fever),
```

```
                \+symptom(cough). %Absence of cough


%Alternative:Diagnosing based on rule covering all possible symptoms

diagnose(unknown):-

                \+diagnose(cold),

                \+diagnose(flu),

                \+diagnose(strep_throat).

%You can ask Prolog:

?-diagnose(Condition).

% Viraj Joshi - 53004230033
```

Output:-

```
%Facts:Define symptoms
%symptom(fever).
symptom(cough).
symptom(sore_throat).
%symptom(body_aches).
symptom(runny_nose).
%symptom(headache).
%symptom(fatigue).

%Facts:Define possible illnesses
condition(cold).
condition(flu).
condition(strep_throat).

%Rules: Diagnosing based on the presence of symptoms
diagnose(cold):-
                symptom(runny_nose),
                symptom(cough),
                symptom(sore_throat),
                \+ symptom(fever). %Absence of fever
diagnose(flu):-
                symptom(fever),
                symptom(cough),
                symptom(body_aches),
                symptom(headache),
                symptom(fatigue).
diagnose(sterp_throat):-
                symptom(sore_throat),
                symptom(fever),
                \+symptom(cough). %Absence of cough

%Alternative:Diagnosing based on rule covering all possible symptoms
diagnose(unknown):-
                \+diagnose(cold),
                \+diagnose(flu),
                \+diagnose(strep_throat).

% Viraj Joshi - 53004230033
```

```
SWI-Prolog (AMD64, Multi-threaded, version 9.2.7)
File  Edit  Settings  Run  Debug  Help
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.7)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- diagnose(Condition).
Condition = cold
```

# Practical 5

Aim:- Design a Fuzzy based application using Python.

Theory:-

A fuzzy logic system is a mathematical framework that handles reasoning with uncertainty and imprecision. Unlike traditional binary logic (where variables are either true or false, i.e., 0 or 1), fuzzy logic allows for degrees of truth, where values can range between 0 and 1. This makes fuzzy logic particularly useful in systems that involve human-like reasoning, where decisions are not strictly binary but involve some level of vagueness.

Example: Consider an air conditioning system:

- Inputs: Temperature and humidity (both can be fuzzy).
- Outputs: Fan speed (also fuzzy).
- Fuzzy rules might say: "If the temperature is high and the humidity is low, then set fan speed to high."

Applications:

- Control systems: Air conditioning, washing machines, and automatic transmission in cars.
- Decision-making systems: Medical diagnosis, stock market prediction, etc.

Fuzzy logic is well-suited for systems where precise data is unavailable, and human-like reasoning is required to make decisions under uncertainty.

Code:-

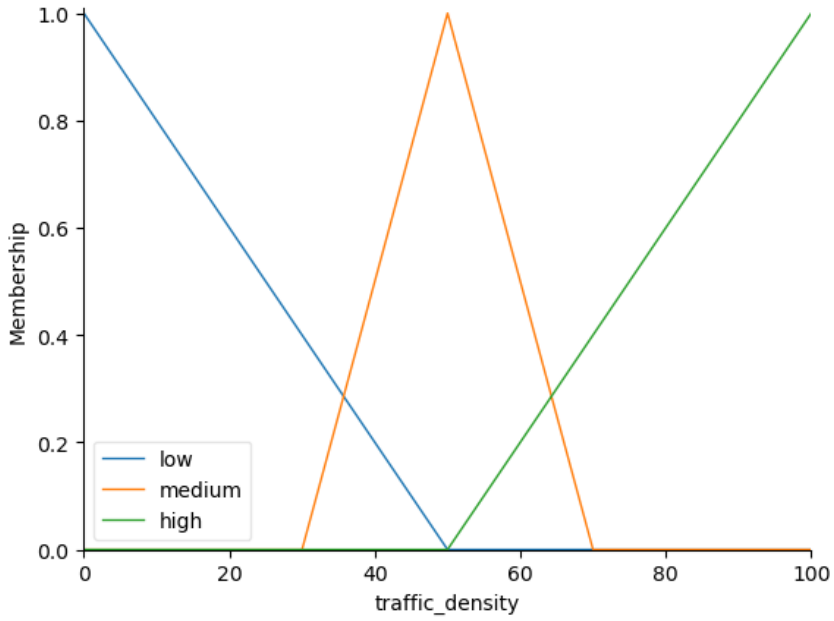
```
import numpy as np

import skfuzzy as fuzz

from skfuzzy import control as ctrl

import matplotlib.pyplot as plt


traffic_density = ctrl.Antecedent(np.arange(0, 101, 1), 'traffic_density')

time_of_day = ctrl.Antecedent(np.arange(0, 25, 1), 'time_of_day')

green_light_duration = ctrl.Consequent(np.arange(0, 61, 1),
'green_light_duration')


traffic_density['low'] = fuzz.trimf(traffic_density.universe, [0, 0, 50])

traffic_density['medium'] = fuzz.trimf(traffic_density.universe, [30, 50,
70])

traffic_density['high'] = fuzz.trimf(traffic_density.universe, [50, 100,
100])
```

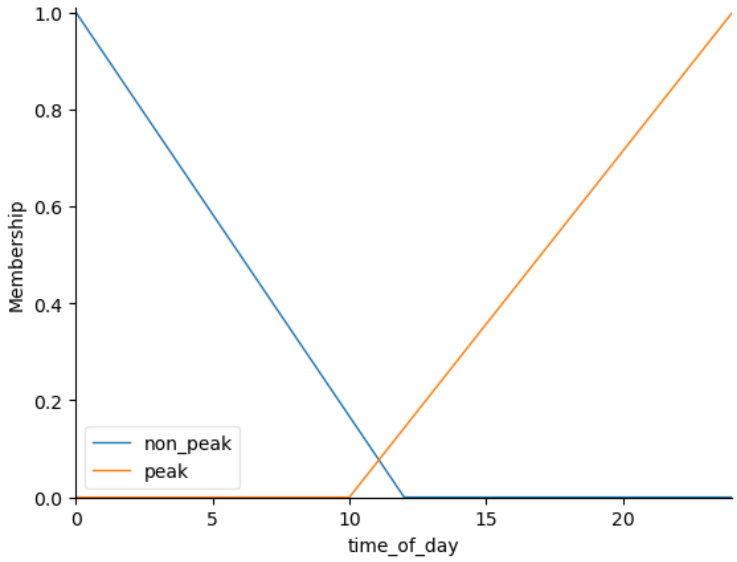
```
time_of_day['non_peak'] = fuzz.trimf(time_of_day.universe, [0, 0, 12])

time_of_day['peak'] = fuzz.trimf(time_of_day.universe, [10, 24, 24])


green_light_duration['short'] = fuzz.trimf(green_light_duration.universe,
[0, 0, 20])

green_light_duration['moderate'] =
fuzz.trimf(green_light_duration.universe, [15, 30, 45])

green_light_duration['long'] = fuzz.trimf(green_light_duration.universe,
[40, 60, 60])


traffic_density.view()

time_of_day.view()

green_light_duration.view()


rule1 = ctrl.Rule(traffic_density['low'] & time_of_day['non_peak'],
green_light_duration['short'])

rule2 = ctrl.Rule(traffic_density['low'] & time_of_day['peak'],
green_light_duration['moderate'])

rule3 = ctrl.Rule(traffic_density['medium'] & time_of_day['non_peak'],
green_light_duration['moderate'])

rule4 = ctrl.Rule(traffic_density['medium'] & time_of_day['peak'],
green_light_duration['long'])

rule5 = ctrl.Rule(traffic_density['high'] & time_of_day['non_peak'],
green_light_duration['long'])

rule6 = ctrl.Rule(traffic_density['high'] & time_of_day['peak'],
green_light_duration['long'])


green_light_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5,
rule6])

green_light_sim = ctrl.ControlSystemSimulation(green_light_ctrl)


green_light_sim.input['traffic_density'] = 75  # High traffic

green_light_sim.input['time_of_day'] = 18      # Peak hours

green_light_sim.compute()

print('Viraj Joshi - 53004230033')

print(f"Recommended Green Light Duration:
{green_light_sim.output['green_light_duration']} seconds")

green_light_duration.view(sim=green_light_sim)
```

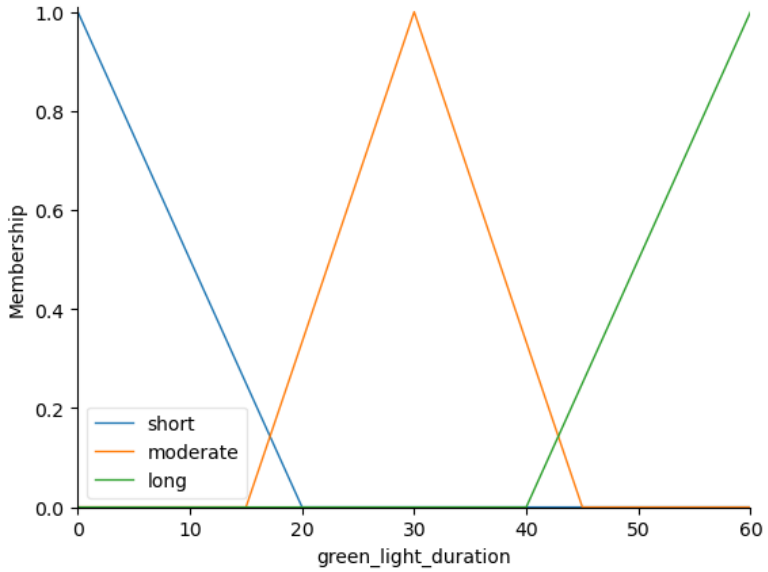
```
plt.show()
```

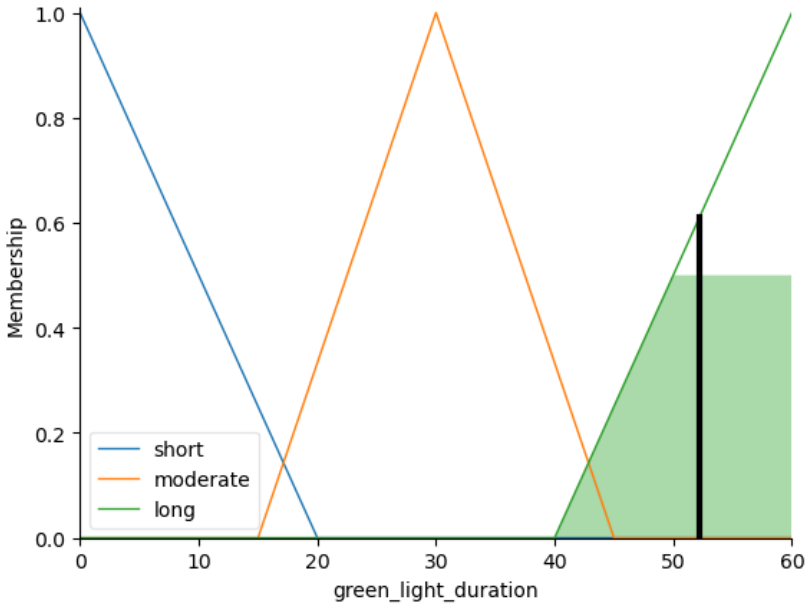
Output:-

```
Viraj Joshi - 53004230033
Recommended Green Light Duration: 52.22222222222222 seconds
```




1.0
0.8
0.6
0.4
0.2
0.0
Membership
short
moderate
long
0
10
20
30
40
50
60
green_light_duration


1.0
0.8
0.6
0.4
0.2
0.0
Membership
short
moderate
long
0
10
20
30
40
50
60
green_light_duration

# Practical 6

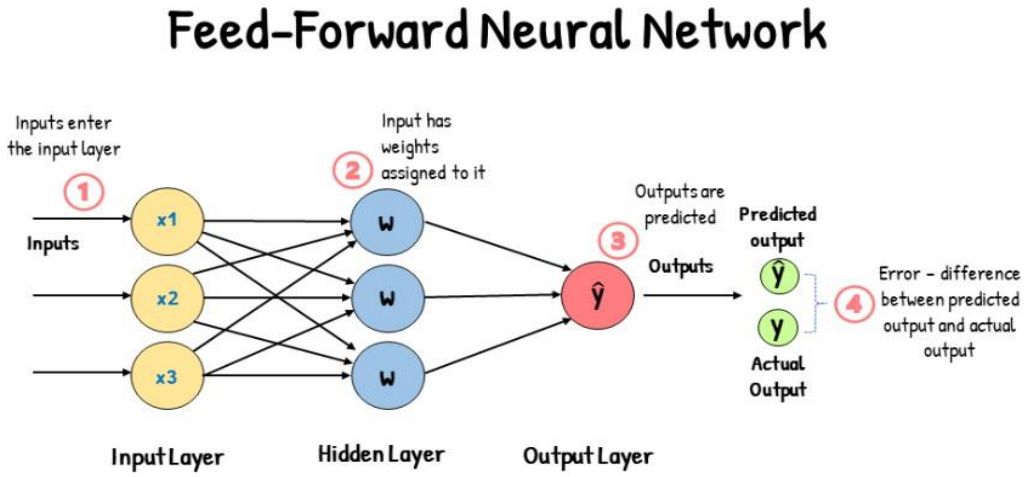
Aim:- Simulate artificial neural network model with both feedforward and backpropagation approach.

Theory:-

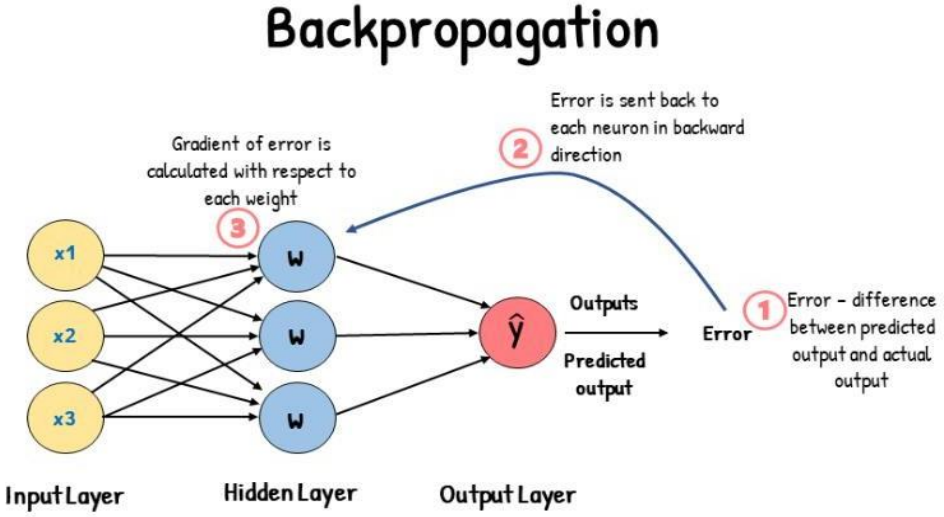
Feedforward Neural Network:

In a Feedforward Neural Network, information moves in one direction only: from the input layer through the hidden layers to the output layer. There are no loops or cycles in the network.

How it works: Input data is passed through the network, with each neuron applying a weighted sum of its inputs and an activation function to produce an output. This continues layer by layer until the final output is obtained.




Backpropagation:Backpropagation is the learning algorithm used to train an ANN. It adjusts




the weights of the network to minimize the error between the predicted output and the actual target.

Code:-

```
import numpy as np

# Sigmoid Activation Function

def sigmoid(x):

    return 1 / (1 + np.exp(-x))


# Derivative of the Sigmoid Function for backpropagation

def sigmoid_derivative(x):

    return x * (1 - x)


# ANN class to simulate feedforward and backpropagation

class ArtificialNeuralNetwork:

    def __init__(self, input_size, hidden_size, output_size,
learning_rate=0.5):

        # Initialize weights randomly

        self.weights_input_hidden = np.random.rand(input_size, hidden_size)

        self.weights_hidden_output = np.random.rand(hidden_size,
output_size)


        # Initialize biases randomly

        self.bias_hidden = np.random.rand(1, hidden_size)

        self.bias_output = np.random.rand(1, output_size)


        # Set the learning rate

        self.learning_rate = learning_rate


    # Feedforward process

    def feedforward(self, X):

        # Hidden layer activation

        self.hidden_input = np.dot(X, self.weights_input_hidden) +
self.bias_hidden

        self.hidden_output = sigmoid(self.hidden_input)


        # Output layer activation

        self.output_input = np.dot(self.hidden_output,
self.weights_hidden_output) + self.bias_output
```

```
        self.output = sigmoid(self.output_input)

        return self.output

    # Backpropagation process
    def backpropagation(self, X, y):
        # Error at the output layer
        output_error = y - self.output
        output_delta = output_error * sigmoid_derivative(self.output)

        # Error at the hidden layer
        hidden_error = output_delta.dot(self.weights_hidden_output.T)
        hidden_delta = hidden_error *
sigmoid_derivative(self.hidden_output)

        # Update the weights and biases using the deltas
        self.weights_hidden_output +=
self.hidden_output.T.dot(output_delta) * self.learning_rate
        self.weights_input_hidden += X.T.dot(hidden_delta) *
self.learning_rate
        self.bias_output += np.sum(output_delta, axis=0, keepdims=True) *
self.learning_rate
        self.bias_hidden += np.sum(hidden_delta, axis=0, keepdims=True) *
self.learning_rate

    # Train the neural network
    def train(self, X, y, epochs):
        for epoch in range(epochs):
            # Feedforward
            self.feedforward(X)
            # Backpropagation
            self.backpropagation(X, y)
            # Print loss every 100 epochs
            if (epoch + 1) % 100 == 0:
                loss = np.mean(np.square(y - self.output))
                print(f'Epoch {epoch + 1}/{epochs}, Loss: {loss:.6f}')
```

```
# Example usage
if __name__ == "__main__":
    # Input dataset (XOR problem)
    X = np.array([[0, 0],
                  [0, 1],
                  [1, 0],
                  [1, 1]])

    # Output dataset (XOR output)
    y = np.array([[0],
                  [1],
                  [1],
                  [0]])

    # Parameters
    input_size = X.shape[1]  # 2 features in input
    hidden_size = 2          # 2 neurons in hidden layer
    output_size = 1          # 1 output neuron (binary classification)

    # Create the neural network
    ann = ArtificialNeuralNetwork(input_size, hidden_size, output_size,
learning_rate=0.5)

    # Train the neural network
    ann.train(X, y, epochs=10000)

    # Test the neural network
    output = ann.feedforward(X)
    print("\nPredicted Output after training:")
    print(output)
    print('Viraj Joshi - 53004230033')
```

Output:-

```
Epoch 100/10000, Loss: 0.250062
Epoch 200/10000, Loss: 0.250009
Epoch 300/10000, Loss: 0.249959
Epoch 400/10000, Loss: 0.249892
Epoch 500/10000, Loss: 0.249783
Epoch 600/10000, Loss: 0.249575
Epoch 700/10000, Loss: 0.249131
Epoch 800/10000, Loss: 0.248059
Epoch 900/10000, Loss: 0.245082
Epoch 1000/10000, Loss: 0.235938
Epoch 1100/10000, Loss: 0.212820
Epoch 1200/10000, Loss: 0.181673
Epoch 1300/10000, Loss: 0.159208
Epoch 1400/10000, Loss: 0.147136
Epoch 1500/10000, Loss: 0.140660
Epoch 1600/10000, Loss: 0.136868
Epoch 1700/10000, Loss: 0.134449
Epoch 1800/10000, Loss: 0.132798
Epoch 1900/10000, Loss: 0.131610
```

```
Epoch 8900/10000, Loss: 0.125487
Epoch 9000/10000, Loss: 0.125480
Epoch 9100/10000, Loss: 0.125474
Epoch 9200/10000, Loss: 0.125467
Epoch 9300/10000, Loss: 0.125461
Epoch 9400/10000, Loss: 0.125455
Epoch 9500/10000, Loss: 0.125449
Epoch 9600/10000, Loss: 0.125444
Epoch 9700/10000, Loss: 0.125438
Epoch 9800/10000, Loss: 0.125433
Epoch 9900/10000, Loss: 0.125428
Epoch 10000/10000, Loss: 0.125422

Predicted Output after training:
[[0.01754588]
 [0.98369851]
 [0.49937487]
 [0.50048993]]
Viraj Joshi - 53004230033
```

# Practical 7

Aim:- Simulate genetic algorithm with suitable example using Python any other platform.

Theroy:-

Genetic Algorithms are a type of optimization technique inspired by the process of natural selection. They're particularly useful for solving complex problems that involve multiple parameters or variables.

In the given code, we are using a genetic algorithm to find a sequence of characters (target_string = "HELLO") with a certain fitness level. The fitness function here checks if the generated string is equal to the target string. If so, it means we've found the optimal solution.

Genetic Algorithms work by:

Initializing a population of candidate solutions (in this case, strings).

Evaluating each candidate's fitness based on a predefined metric.

Selecting parents from the current generation to reproduce and create offspring.

Applying crossover and mutation operators to generate new candidates.

Repeating steps 3-4 until a termination condition is met.

Code:-

```
import random

import string

target_string = "HELLO"

population_size = 50

mutation_rate = 0.01

generations = 200


def fitness(individual):

    return sum(1 for a, b in zip(individual, target_string) if a == b)


def create_population(size):

    return [''.join(random.choices(string.ascii_uppercase,
k=len(target_string))) for _ in range(size)]


def select_parents(population):

    tournament = random.sample(population, 5)

    return max(tournament, key=fitness)
```

```
def crossover(parent1, parent2):
    crossover_point = random.randint(1, len(parent1) - 1)
    return parent1[:crossover_point] + parent2[crossover_point:]


def mutate(individual):
    individual = list(individual)
    for i in range(len(individual)):
        if random.random() < mutation_rate:
            individual[i] = random.choice(string.ascii_uppercase)
    return ''.join(individual)


population = create_population(population_size)
for generation in range(generations):
    best_individual = max(population, key=fitness)
    print(f"Generation {generation}: Best individual: {best_individual},
Fitness: {fitness(best_individual)}")


    if fitness(best_individual) == len(target_string):
          break


    new_population = []
    for _ in range(population_size):
        parent1 = select_parents(population)
        parent2 = select_parents(population)
        child = crossover(parent1, parent2)
        child = mutate(child)
        new_population.append(child)


    population = new_population


best_individual = max(population, key=fitness)
print(f"Best individual: {best_individual}, Fitness:
{fitness(best_individual)}")
```

Output:-

```
Generation 0: Best individual: HMFLW, Fitness: 2
Generation 1: Best individual: HRELO, Fitness: 3
Generation 2: Best individual: HEYMO, Fitness: 3
Generation 3: Best individual: HMLLO, Fitness: 4
Generation 4: Best individual: HALLO, Fitness: 4
Generation 5: Best individual: HALLO, Fitness: 4
Generation 6: Best individual: HULLO, Fitness: 4
Generation 7: Best individual: HMLLO, Fitness: 4
Generation 8: Best individual: HALLO, Fitness: 4
Generation 9: Best individual: HEYLO, Fitness: 4
Generation 10: Best individual: HALLO, Fitness: 4
Generation 11: Best individual: HMLLO, Fitness: 4
Generation 12: Best individual: HULLO, Fitness: 4
Generation 13: Best individual: HALLO, Fitness: 4
Generation 14: Best individual: HALLO, Fitness: 4
Generation 15: Best individual: HALLO, Fitness: 4
Generation 16: Best individual: HULLO, Fitness: 4
Generation 17: Best individual: HULLO, Fitness: 4
Generation 18: Best individual: HZLLO, Fitness: 4
Generation 19: Best individual: HALLO, Fitness: 4
Generation 20: Best individual: HALLO, Fitness: 4
Generation 21: Best individual: HALLO, Fitness: 4
Generation 22: Best individual: HZLLO, Fitness: 4
Generation 23: Best individual: HMLLO, Fitness: 4
Generation 24: Best individual: HULLO, Fitness: 4
Generation 25: Best individual: HMLLO, Fitness: 4
Generation 26: Best individual: HULLO, Fitness: 4
Generation 27: Best individual: HZLLO, Fitness: 4
Generation 28: Best individual: HALLO, Fitness: 4

Generation 29: Best individual: HALLO, Fitness: 4
Generation 30: Best individual: HALLO, Fitness: 4
Generation 31: Best individual: HALLO, Fitness: 4
Generation 32: Best individual: HZLLO, Fitness: 4
Generation 33: Best individual: HALLO, Fitness: 4
Generation 34: Best individual: HELLO, Fitness: 5
Best individual: HELLO, Fitness: 5
Viraj Joshi - 53004230033
```

# Practical 8

Aim:- Design intelligent agent using any AI algorithm. design expert tutoring system

Theory:-

Artificial Neural Network (ANN)

Artificial Neural Networks are computational models inspired by the structure and function of biological neural networks. They're powerful tools for classification, regression, and other machine learning tasks.

The given code implements an ANN with a single hidden layer to classify inputs into binary outputs. The sigmoid activation function is used in both the hidden and output layers.

ANNs work by:

Initializing weights and biases randomly.

Forward propagating input data through the network, using activation functions (e.g., sigmoid) to introduce non-linearity.

Computing the error between predicted outputs and actual labels.

Backpropagating this error through the network to adjust weights and biases.

Repeating steps 2-4 until convergence or a termination condition is met.

Code:-

```
class MathTutor:
    def __init__(self):
        self.operations = {
            '+': lambda a, b: a + b,
            '-': lambda a, b: a - b,
            '*': lambda a, b: a * b,
            '/': lambda a, b: a/b,
        }
    def explain_operation(self, operator):
        explanation = {
            '+': "Addition adds two numbers together.",
            '-': "Subtraction subtracts the second number from the first.",
            '*': "Multiplication gives the product of two numbers.",
            '': "Division divides the first number by the second.",
        }
        return explanation.get(operator, "Invalid operation.")
    def perform_operation(self, operator, a, b):
```

```
        if operator in self.operations:
            return self.operations[operator](a, b)
        else:
            return None
if __name__ == "__main__":
    tutor = MathTutor()
    # Example usage:
    operator = '+'
    a, b = 24, 8
    print(tutor.explain_operation(operator))
    result = tutor.perform_operation(operator, a, b)
    print(f"Result of {a} {operator} {b} = {result}")
print('Viraj Joshi - 53004230033')
```

Output:-

```
Addition adds two numbers together.
Result of 24 + 8 = 32
Viraj Joshi - 53004230033

Subtraction subtracts the second number from the first.
Result of 24 - 8 = 16
Viraj Joshi - 53004230033

Multiplication gives the product of two numbers.
Result of 24 * 8 = 192
Viraj Joshi - 53004230033

Division divides the first number by the second.
Result of 24 / 8 = 3.0
Viraj Joshi - 53004230033
```

# Practical 9

Aim: Design an application to simulate language parser.

Theory:-

Knowledge Representation is a field in artificial intelligence (AI) that deals with how to formally capture and organize information about the world in a way that a machine can understand and reason with.

KR systems use structured formats (e.g., logic, semantic networks, ontologies) to represent facts, concepts, and relationships between objects. This allows AI systems to process complex data, reason about it, and draw conclusions.

In short, KR provides a framework for storing and manipulating knowledge in a machine-readable format, enabling intelligent decision-making and problem-solving.

Code:-

```
class SimpleParser:
    def __init__(self, expr):
        self.tokens = expr.replace('(', ' ( ').replace(')', ' ) ').split()
        self.pos = 0

    def parse(self):
        return self.expr()

    def advance(self):
        self.pos += 1

    def current_token(self):
        return self.tokens[self.pos] if self.pos < len(self.tokens) else
None

    def expr(self):
        result = self.term()
        while self.current_token() in ('+', '-'):
            if self.current_token() == '+':
                self.advance()
                result += self.term()
            elif self.current_token() == '-':
                self.advance()
```

```
                result -= self.term()
        return result

    def term(self):
        result = self.factor()
        while self.current_token() in ('*', ''):
            if self.current_token() == '*':
                self.advance()
                result *= self.factor()
            elif self.current_token() == '':
                self.advance()
                result = self.factor()
        return result

    def factor(self):
        token = self.current_token()
        if token.isdigit():
            self.advance()
            return int(token)
        elif token == '(':
            self.advance()
            result = self.expr()
            self.advance()  # skip ')'
            return result
        raise ValueError("Invalid syntax")

if __name__ == "__main__":
    expr = "(10 + 5) * 2"
    parser = SimpleParser(expr)
    result = parser.parse()
    print(f"Result of '{expr}' is {result}")
print('Viraj Joshi - 53004230033')
```

Output:-

```
Result of '(10 + 5) * 2' is 30
Viraj Joshi - 53004230033

Result of '(15 - 30) * 4' is -60
Viraj Joshi - 53004230033
```

# Practical 10

Aim: Develop the semantic net using python.

Theory:-

Semantic Network:

A semantic network is a data structure used to represent knowledge in the form of concepts (nodes) and their interrelationships (edges or links). It is a graphical model that depicts how different concepts in a particular domain are connected and how they relate to each other semantically.

Example: Consider a semantic network for animals:

- Concepts (Nodes): "Dog", "Cat", "Animal", "Mammal".
- Relationships (Edges): "Dog is a Mammal", "Mammal is a Animal", "**Dog** has **Fur**", "**Dog** can **Bark**".

In this network:

- Dog is connected to Mammal by an "is a" relationship.
- Dog is connected to Bark by a "can" relationship.

Code:-

```
class SemanticNetwork:
    def __init__(self):
        self.network = {}

    def add_concept(self, concept):
        if concept not in self.network:
            self.network[concept] = {'is_a': [], 'has_a': []}

    def add_relation(self, relation, concept1, concept2):
        self.add_concept(concept1)
        self.add_concept(concept2)
        self.network[concept1][relation].append(concept2)

    def get_relations(self, concept):
        return self.network.get(concept, {})

    def display_network(self):
```

```
        for concept, relations in self.network.items():
            print(f"Concept: {concept}")
            for relation, related_concepts in relations.items():
                for related_concept in related_concepts:
                    print(f"  {relation} -> {related_concept}")

if __name__ == "__main__":
    sn = SemanticNetwork()

    # Adding concepts and relations
    sn.add_concept("Animal")
    sn.add_concept("Bird")
    sn.add_concept("Mammal")
    sn.add_concept("Penguin")
    sn.add_concept("Canary")

    sn.add_relation("is_a", "Bird", "Animal")
    sn.add_relation("is_a", "Mammal", "Animal")
    sn.add_relation("is_a", "Penguin", "Bird")
    sn.add_relation("is_a", "Canary", "Bird")
    sn.add_relation("has_a", "Bird", "Wings")
    sn.add_relation("has_a", "Canary", "Yellow_Feathers")
    # Displaying the network
    sn.display_network()
print()
print('Viraj Joshi - 53004230033')
```

Output:-

```
Concept: Animal
Concept: Bird
  is_a -> Animal
  has_a -> Wings
Concept: Mammal
  is_a -> Animal
Concept: Penguin
  is_a -> Bird
Concept: Canary
  is_a -> Bird
  has_a -> Yellow_Feathers
Concept: Wings
Concept: Yellow_Feathers

Viraj Joshi - 53004230033
```

# INDEX

# Practical 1

Aim:- Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

Theory:-

Naïve Bayes is a family of supervised learning algorithms based on the Bayes' theorem. It's used for classification and regression tasks, particularly when dealing with categorical or numerical features.

Bayes' Theorem

The Bayes' theorem states that the posterior probability of a hypothesis (H) given some evidence (E) can be calculated using the following formula:

$P(H|E) = P(E|H) * P(H) / P(E)$

where:

P(H) is the prior probability of the hypothesis

P(E|H) is the likelihood of the evidence given the hypothesis

P(E) is the marginal likelihood of the evidence

Naïve Bayes Assumptions

The Naïve Bayes algorithm makes several assumptions to simplify the calculation:

Independence : The features are assumed to be independent of each other, i.e., the presence or absence of one feature does not affect the others.

Normality : The data is normally distributed for continuous features and has a specific distribution (e.g., Bernoulli) for categorical features.

Equal Prior Probabilities : The prior probabilities of all classes are assumed to be equal.

Types of Naïve Bayes

There are several types of Naïve Bayes algorithms, including:

Gaussian Naïve Bayes (GNB) : Assumes a normal distribution for continuous features.

Multinomial Naïve Bayes : Assumes a multinomial distribution for categorical features.

Bernoulli Naïve Bayes : Assumes a Bernoulli distribution for binary features.

Code:-

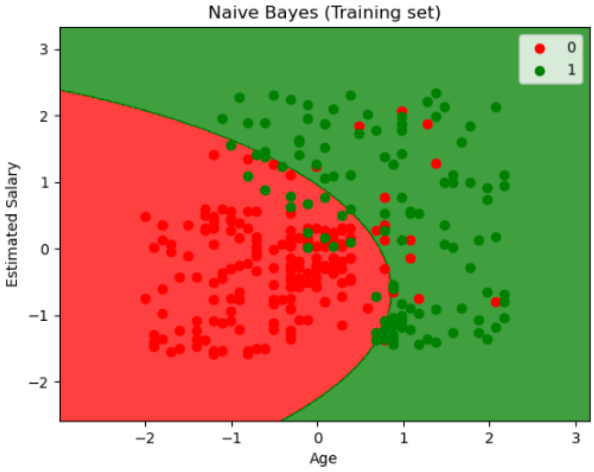
```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd
```

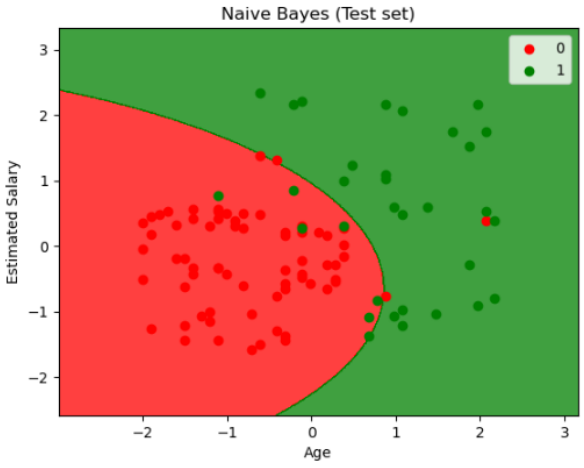
```
# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting classifier to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
```

```
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Naive Bayes (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
print('Viraj Joshi-53004230033')
plt.show()


# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Naive Bayes (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Output:-

```
Viraj Joshi-53004230033
```



# Practical 2

Aim:- Write a program to implement Decision Tree and Random forest with Prediction, Test Score and Confusion Matrix.

Theory:-

Random Forest is a popular ensemble learning algorithm that combines multiple decision trees to improve the accuracy and robustness of predictions. Here's an overview of the theory behind Random Forest:

Key Concepts

Ensemble Learning : A machine learning paradigm where multiple models are combined to produce a single output.

Decision Trees : A type of supervised learning model that splits data into subsets based on feature values.

Random Forest Algorithm

The Random Forest algorithm consists of the following steps:

Bootstrap Sampling : Create multiple training sets by randomly sampling from the original dataset with replacement.

Decision Tree Construction : Build a decision tree on each bootstrap sample using a random subset of features.

Feature Importance : Calculate the importance of each feature in the decision trees using metrics like Gini impurity or variance reduction.

Prediction : Combine the predictions from all decision trees to produce a final output.

Key Components

Multiple Decision Trees : Random Forest combines multiple decision trees, which helps to:

Reduce overfitting: By averaging the predictions of multiple trees, we reduce the impact of individual tree errors.

Improve robustness: Combining multiple trees with different feature subsets and bootstrap samples increases the stability of the model.

Random Feature Subset : At each node, Random Forest selects a random subset of features to consider for splitting. This helps to:

Reduce overfitting: By using only a portion of available features, we avoid overly complex models that fit noise in the data.

Improve generalization: The random feature selection process helps to capture different patterns and relationships in the data.

Code:-

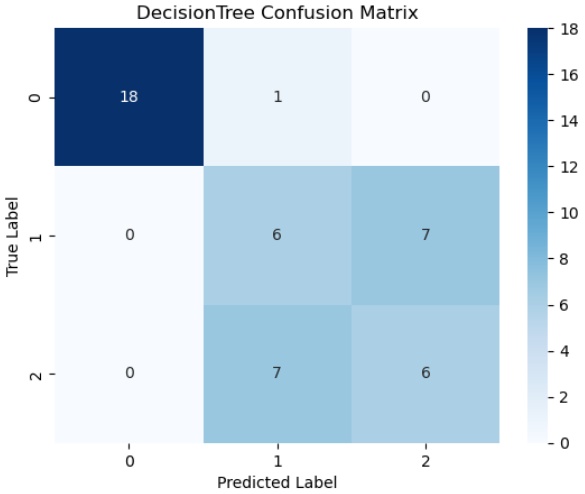
```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import load_iris


iris = load_iris()

X = pd.DataFrame(iris.data, columns=iris.feature_names).iloc[:,:2]

y = pd.DataFrame(iris.target, columns=['species'])


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)


def plot_decision_boundary(clf, X, y, title):

    x_min, x_max = X.iloc[:,0].min()-1, X.iloc[:,0].max()+1

    y_min, y_max = X.iloc[:,1].min()-1, X.iloc[:,1].max()+1

    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),

                         np.arange(y_min, y_max, 0.01))


    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)


    plt.contour(xx, yy, Z, alpha=0.4, cmap=plt.cm.RdYlBu)

    plt.scatter(X.iloc[:,0], X.iloc[:,1], c=y.values.ravel(), s=40,
edgecolor='k', cmap=plt.cm.RdYlBu)

    plt.title(title)

    plt.xlabel(iris.feature_names[0])

    plt.ylabel(iris.feature_names[1])

    plt.show()
```

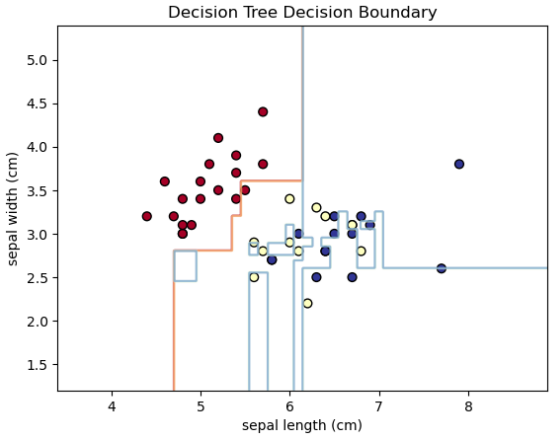
```
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

dt_predictions = dt_model.predict(X_test)

dt_accuracy = accuracy_score(y_test, dt_predictions)
dt_confusion_matrix = confusion_matrix(y_test, dt_predictions)

print(f'Decision Tree Accuracy: {dt_accuracy}')
print('Decision Tree Classification Report:')
print(classification_report(y_test, dt_predictions))

sns.heatmap(dt_confusion_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('DecisionTree Confusion Matrix')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()

plot_decision_boundary(dt_model, X_test, y_test, 'Decision Tree Decision
Boundary')

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train.values.ravel())

rf_predictions = rf_model.predict(X_test)

rf_accuracy = accuracy_score(y_test, rf_predictions)
rf_confusion_matrix = confusion_matrix(y_test, rf_predictions)

print(f'Random Forest Accuracy: {rf_accuracy}')
print('Random Forest Classification Report:')
print(classification_report(y_test, rf_predictions))

sns.heatmap(rf_confusion_matrix, annot=True, fmt='d', cmap='Greens')
plt.title('Random Forest Confusion Matrix')
```

```
plt.ylabel('True Label')

plt.xlabel('Predicted Label')

plt.show()

plot_decision_boundary(rf_model, X_test, y_test, 'Random Forest Decision
Boundary')

print('Viraj Joshi- 53004230033')
```

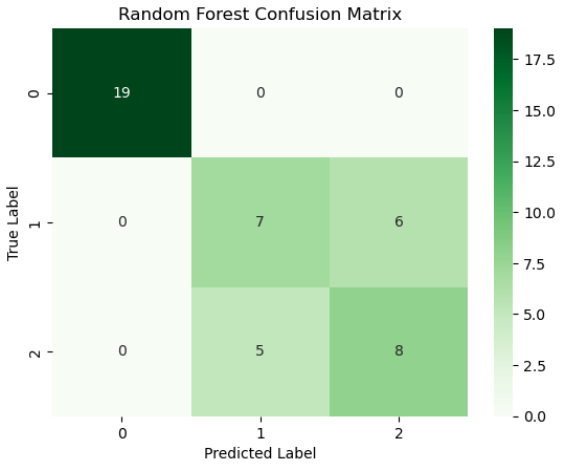
Output:-

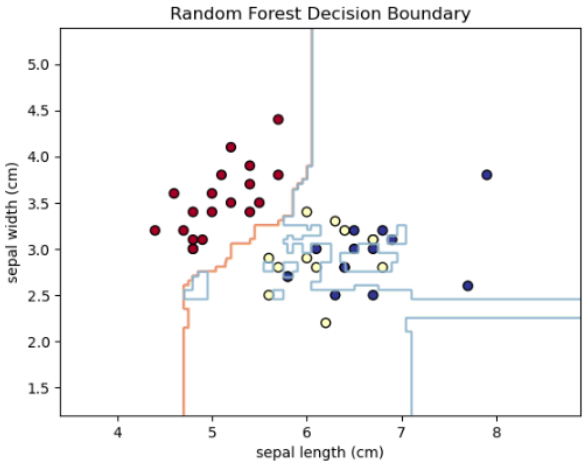
```
Decision Tree Accuracy: 0.6666666666666666
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        19
           1       0.43      0.46      0.44        13
           2       0.46      0.46      0.46        13

    accuracy                           0.67        45
   macro avg       0.63      0.62      0.63        45
weighted avg       0.68      0.67      0.67        45
```



```
Random Forest Accuracy: 0.7555555555555555
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.58      0.54      0.56        13
           2       0.57      0.62      0.59        13

    accuracy                           0.76        45
   macro avg       0.72      0.72      0.72        45
weighted avg       0.76      0.76      0.76        45
```




Viraj Joshi- 53004230033

# Practical 3

Aim:- For a given set of training data examples stored in a .CSV file implement Least Square Regression algorithm.

Thoery:-

Least Squares Regression is a fundamental approach in linear regression, widely used to model the relationship between a dependent variable (target) and one or more independent variables (features). The primary goal is to find the line (or hyperplane in higher dimensions) that best fits the observed data points by minimizing the sum of the squared differences between the actual target values and the predicted values from the model.

**Mathematical Foundation:**

Given a dataset with nnn data points, each consisting of mmm features, the relationship between the features XXX and the target yyy can be represented as:

y=Xβ+ϵy = X\beta + \epsilony=Xβ+ϵ

- XXX is an n×mn \times mn×m matrix where each row corresponds to a feature vector of a data point.
- β\betaβ is the vector of coefficients we want to estimate (with a length of mmm).
- yyy is the n×1n \times 1n×1 vector of the observed target values.
- ϵ\epsilonϵ represents the error term, capturing the difference between the actual and predicted values.

The Least Squares Regression model seeks to minimize the **Sum of Squared Errors (SSE)**, which is the sum of the squared residuals (the differences between actual and predicted values).

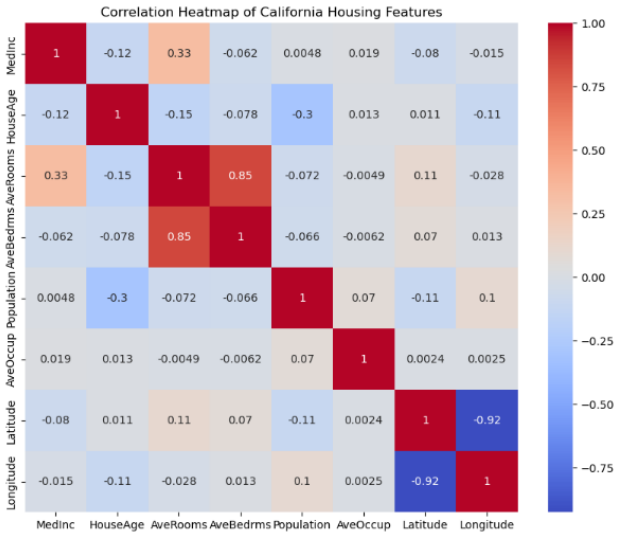
Code:-

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

from sklearn.datasets import fetch_california_housing


housing = fetch_california_housing()

X = pd.DataFrame(housing.data, columns = housing.feature_names)

y = pd.DataFrame(housing.target, columns = ['MEDV'])
```

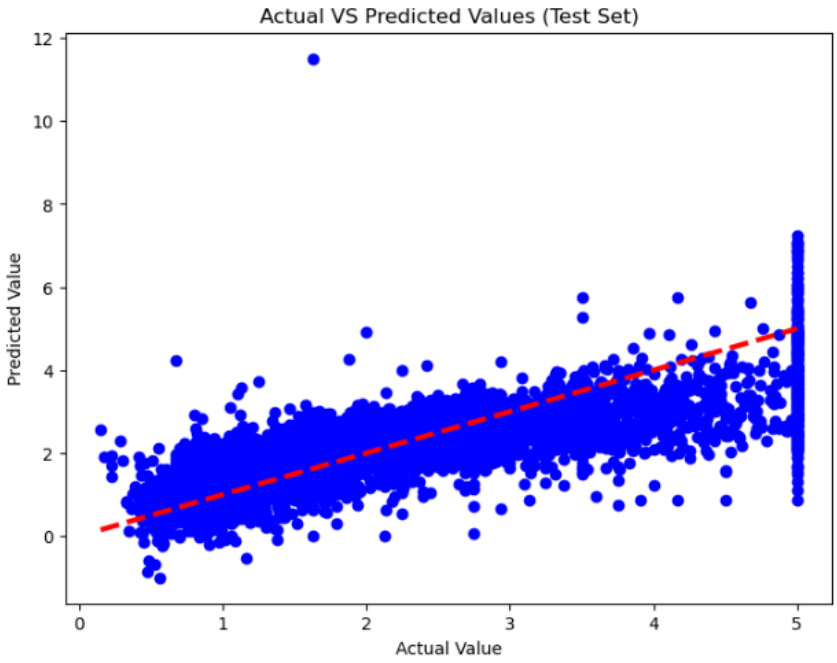
```
plt.figure(figsize=(10,8))

sns.heatmap(X.corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap of California Housing Features")

plt.show()


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)


reg_model = LinearRegression()

reg_model.fit(X_train, y_train)


y_train_pred = reg_model.predict(X_train)

y_test_pred = reg_model.predict(X_test)


train_mse = mean_squared_error(y_train, y_train_pred)

test_mse = mean_squared_error(y_test, y_test_pred)

train_r2 = r2_score(y_train, y_train_pred)

test_r2 = r2_score(y_test, y_test_pred)


print(f'Training Mean Squared Error: {train_mse}')

print(f'Test Mean Squared Error: {test_mse}')

print(f'Training R^2 Score: {train_r2}')

print(f'Test R^2 Score: {test_r2}')


coefficients = pd.DataFrame(reg_model.coef_.T, X.columns,
columns=['Coefficients'])

print(coefficients)


plt.figure(figsize=(8,6))

plt.scatter(y_test, y_test_pred, c='blue')

plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--r',
lw =3)

plt.xlabel('Actual Value')

plt.ylabel('Predicted Value')

plt.title('Actual VS Predicted Values (Test Set)')
```

```
print('Viraj Joshi- 53004230033')
```

Output:-




```
Training Mean Squared Error: 0.5233576288267755
Test Mean Squared Error: 0.5305677824766757
Training R^2 Score: 0.6093459727972159
Test R^2 Score: 0.595770232606166
              Coefficients
MedInc       4.458226e-01
HouseAge     9.681868e-03
AveRooms    -1.220951e-01
AveBedrms    7.785996e-01
Population  -7.757404e-07
AveOccup    -3.370027e-03
Latitude    -4.185367e-01
Longitude   -4.336880e-01

Viraj Joshi- 53004230033
```



# Practical 4

Aim:- For a given set of training data examples stored in a .CSV file implement Logistic Regression algorithm.

Theory:-

Logistic regression is a popular supervised learning algorithm used for binary classification problems, where the output variable is categorical (0/1 or yes/no). Here's an overview of the theory behind logistic regression:

Key Concepts

Binary Response Variable : The output variable we're trying to predict has only two possible values: 0 and 1.

Logistic Function : A sigmoid function that maps any real-valued number to a probability between 0 and 1.

Logistic Regression Equation

The logistic regression equation is:

$p = 1 / (1 + \exp(-z))$

where p is the predicted probability of the positive class, exp is the exponential function, and z is a linear combination of the input features (X) and weights (β):

$z = \beta 0 + \beta 1x1 + \beta 2x2 + \ldots + \beta nxn$

Maximum Likelihood Estimation

The goal of logistic regression is to find the optimal values for the weights (β) that maximize the likelihood of observing the training data. This is achieved through maximum likelihood estimation (MLE), which involves minimizing the negative log-likelihood function:

$L = -\sum[i=1]n [yilog(p\_i) + (1-y\_i)log(1-p\_i)]$

Code:-

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

from sklearn.datasets import load_breast_cancer
```

```
cancer_data = load_breast_cancer()

X = pd.DataFrame(cancer_data.data, columns= cancer_data.feature_names)

y = pd.DataFrame(cancer_data.target, columns= ['target'])


print('Dataset Head:')

print(X.head())


print('Target Distribution:')

print(y['target'].value_counts())


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state = 42)


logreg = LogisticRegression(max_iter=10000, random_state=42)

logreg.fit(X_train, y_train.values.ravel())


y_pred = logreg.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)

class_report = classification_report(y_test, y_pred)


print(f'Accuracy: {accuracy}')

print()

print('Confusion matrix:')

print(conf_matrix)

print()

print('Classification report:')

print(class_report)


plt.figure(figsize=(6,4))

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')
```

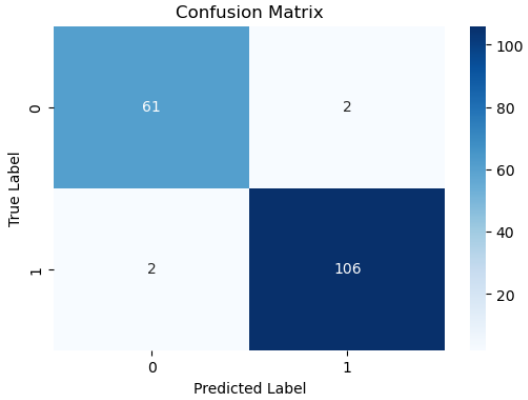
```
plt.show()


new_input = np.array([X.mean().values])
print(f'New input for prediction: {new_input}')


new_prediction = logreg.predict(new_input)
predicted_class = 'benign' if new_prediction == 1 else 'maligant'
print(f'Predicted class for the new input: {predicted_class}')
plt.figure(figsize=(6,4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion matrix - Test set')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
print('Viraj Joshi- 53004230033')
```

Output:-

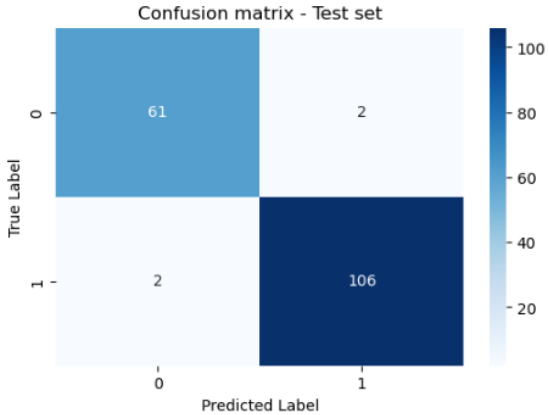
```
Dataset Head:
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809


    mean fractal dimension  ...  worst radius  worst texture  worst perimeter  \
 0                 0.07871  ...         25.38          17.33           184.60
 1                 0.05667  ...         24.99          23.41           158.80
 2                 0.05999  ...         23.57          25.53           152.50
 3                 0.09744  ...         14.91          26.50            98.87
 4                 0.05883  ...         22.54          16.67           152.20

    worst area  worst smoothness  worst compactness  worst concavity  \
 0      2019.0            0.1622             0.6656           0.7119
 1      1956.0            0.1238             0.1866           0.2416
 2      1709.0            0.1444             0.4245           0.4504
 3       567.7            0.2098             0.8663           0.6869
 4      1575.0            0.1374             0.2050           0.4000
```

```
   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
Target Distribution:
target
1    357
0    212
Name: count, dtype: int64
Accuracy: 0.9766081871345029

Confusion matrix:
[[ 61   2]
 [  2 106]]
```

```
Classification report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        63
           1       0.98      0.98      0.98       108

    accuracy                           0.98       171
   macro avg       0.97      0.97      0.97       171
weighted avg       0.98      0.98      0.98       171
```




```
New input for prediction: [[1.41272917e+01 1.92896485e+01 9.19690334e+01 6.54889104e+02
  9.63602812e-02 1.04340984e-01 8.87993158e-02 4.89191459e-02
  1.81161863e-01 6.27976098e-02 4.05172056e-01 1.21685343e+00
  2.86605923e+00 4.03370791e+01 7.04097891e-03 2.54781388e-02
  3.18937163e-02 1.17961371e-02 2.05422988e-02 3.79490387e-03
  1.62691898e+01 2.56772232e+01 1.07261213e+02 8.80583128e+02
  1.32368594e-01 2.54265044e-01 2.72188483e-01 1.14606223e-01
  2.90075571e-01 8.39458172e-02]]
Predicted class for the new input: maligant
```




```
Viraj Joshi- 53004230033
```

# Practical 5

Aim:- Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Theory:-

A Decision Tree is a type of machine learning algorithm that splits data into subsets based on feature values. The goal is to find the best split that maximizes the purity or impurity of the data.

Gini Impurity

In this example, we're using Gini Impurity as the criterion for splitting the data. Gini Impurity measures the probability of an instance being misclassified by a decision tree. It's calculated as:

$1 - \sum (p_i)^2$

where p_i is the proportion of instances in node i.

The goal is to minimize Gini Impurity at each split.

Decision Tree Construction

Here's a high-level overview of how the Decision Tree construction works:

Root Node : Start with the root node, which contains all instances.

Splitting : Select the best feature and value for splitting the data into two subsets. This is done by minimizing Gini Impurity at each split.

Node Creation : Create a new node for each subset created in step 2.

Recursion : Repeat steps 1-3 until all instances are classified or a stopping criterion is met (e.g., maximum depth reached).

Feature Selection

The feature selection process involves selecting the best feature and value for splitting the data at each node. In this example, we're using all features (df_encoded.iloc[:, :-1]) except the target variable ('Buys Computer').

Code:-

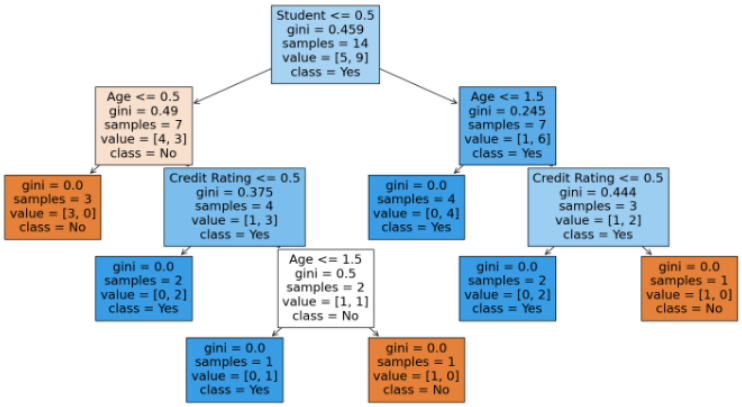
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import tree as sk_tree
```

```
# Step 1: Parse the dataset

data = {

    'Age': ['<=30', '<=30', '31-40', '>40', '>40', '>40', '31-40', '<=30',
'<=30', '>40', '<=30', '31-40', '31-40', '>40'],

    'Income': ['High', 'High', 'High', 'Medium', 'Low', 'Low', 'Low',
'Medium', 'Low', 'Medium', 'Medium', 'Medium', 'High', 'Medium'],

    'Student': ['No', 'No', 'No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes',
'Yes', 'Yes', 'No', 'Yes', 'No'],

'Credit Rating': ['Fair', 'Excellent', 'Fair', 'Fair', 'Fair', 'Excellent',
'Excellent', 'Fair', 'Fair', 'Fair', 'Excellent', 'Excellent', 'Fair',
'Excellent'],

    'Buys Computer': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No',
'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']

}


df = pd.DataFrame(data)


# Encode the categorical variables

df_encoded = df.apply(lambda x: pd.factorize(x)[0])

# Fit the decision tree classifier using Gini impurity

clf_gini = sk_tree.DecisionTreeClassifier(criterion='gini')

clf_gini = clf_gini.fit(df_encoded.iloc[:, :-1], df_encoded['Buys
Computer'])


# Convert the feature names from Index to list

feature_names = df.columns[:-1].tolist()


# Convert the class names to a list

class_names = df['Buys Computer'].unique().tolist()


# Plot the decision tree

plt.figure(figsize=(20,10))

sk_tree.plot_tree(clf_gini, feature_names=feature_names,
class_names=class_names, filled=True)

plt.show()


# Plot the decision tree

plt.figure(figsize=(20,10))
```

```
sk_tree.plot_tree(clf_gini, feature_names=feature_names,
class_names=class_names, filled=True)

plt.show()

# Function to print Gini impurity and chosen attribute at each split

def print_gini_and_splits(tree, feature_names):

    tree_ = tree.tree_

    feature_name = [

        feature_names[i] if i != sk_tree._tree.TREE_UNDEFINED else
"undefined!"

        for i in tree_.feature

    ]


    print("Decision tree splits and Gini impurities:")

    for i in range(tree_.node_count):

        if tree_.children_left[i] != sk_tree._tree.TREE_LEAF:

            print(f"Node {i} (Gini: {tree_.impurity[i]:.4f}): split on
feature '{feature_name[i]}'")

        else:

            print(f"Node {i} (Gini: {tree_.impurity[i]:.4f}): leaf node")


print_gini_and_splits(clf_gini, feature_names)


# Example test sample

test_sample = {

    'Age': '<=30',

    'Income': 'Medium',

    'Student': 'Yes',

    'Credit Rating': 'Fair'

}


# Encode the test sample

encoded_sample = pd.DataFrame([test_sample]).apply(lambda x:
pd.factorize(df[x.name])[0][df[x.name].tolist().index(x[0])])


# Predict using sklearn decision tree

sklearn_prediction = clf_gini.predict([encoded_sample])
```

```
decoded_prediction = pd.factorize(df['Buys
Computer'])[1][sklearn_prediction[0]]

print("Prediction for sklearn decision tree:", decoded_prediction)

print()

print('Viraj Joshi- 53004230033')
```

Output:-




```
Decision tree splits and Gini impurities:
Node 0 (Gini: 0.4592): split on feature 'Student'
Node 1 (Gini: 0.4898): split on feature 'Age'
Node 2 (Gini: 0.0000): leaf node
Node 3 (Gini: 0.3750): split on feature 'Age'
Node 4 (Gini: 0.0000): leaf node
Node 5 (Gini: 0.5000): split on feature 'Credit Rating'
Node 6 (Gini: 0.0000): leaf node
Node 7 (Gini: 0.0000): leaf node
Node 8 (Gini: 0.2449): split on feature 'Credit Rating'
Node 9 (Gini: 0.0000): leaf node
Node 10 (Gini: 0.4444): split on feature 'Age'
Node 11 (Gini: 0.0000): leaf node
Node 12 (Gini: 0.0000): leaf node
Prediction for sklearn decision tree: Yes

Viraj Joshi- 53004230033
```

# Practical 6

Aim:- Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set.

Theory:-

K-Nearest Neighbors (KNN) is a supervised learning algorithm that uses the distance between data points in a feature space to make predictions. It's a simple yet effective method for classification and regression tasks.

Key Concepts

Similarity : KNN relies on measuring the similarity between new, unseen instances and existing training data.

Neighbors : For each instance, find its k nearest neighbors (k-NN) in the feature space.

Distance Metrics : Use a distance metric (e.g., Euclidean, Manhattan, or cosine similarity) to compute the similarity between instances.

Theory Behind KNN

KNN is based on the idea that similar data points tend to have similar labels. By examining the k nearest neighbors of an instance, we can infer its label based on the patterns learned from the training data.

Instance-based Learning : KNN is an instance-based learning algorithm, meaning it makes predictions based on individual instances rather than features.

Supervised Learning : KNN is a supervised learning algorithm since it relies on labeled training data to make predictions.

Non-parametric Model : KNN is a non-parametric model, which means it doesn't assume any underlying distribution for the data.

Code:-

```
# Step 1: Import necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

from mpl_toolkits.mplot3d import Axes3D
```

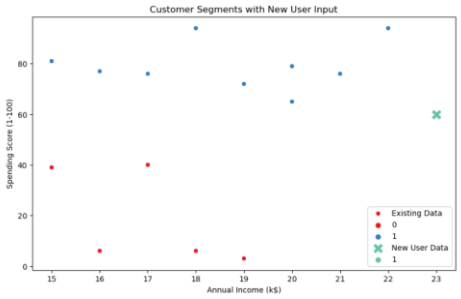
```
# Step 2: Load and display the sample data

data = {

    'Age': [19, 21, 20, 23, 31, 22, 35, 25, 23, 64, 30, 67, 35, 58, 24],

    'Annual Income (k$)': [15, 15, 16, 16, 17, 17, 18, 18, 19, 19, 20, 20,
21, 21, 22],

    'Spending Score (1-100)': [39, 81, 6, 77, 40, 76, 6, 94, 3, 72, 79, 65,
76, 76, 94],

    'Segment': [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1]  # 0: Low-
value, 1: High-value

}


df = pd.DataFrame(data)

print("Sample Data:")

print(df.head())


# Step 3: Data Preprocessing

X = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]

y = df['Segment']


scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)


# Step 4: Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)


# Step 5: Apply KNN Algorithm

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)


# Step 6: Evaluation

print("\nConfusion Matrix:")

print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")

print(classification_report(y_test, y_pred))
```

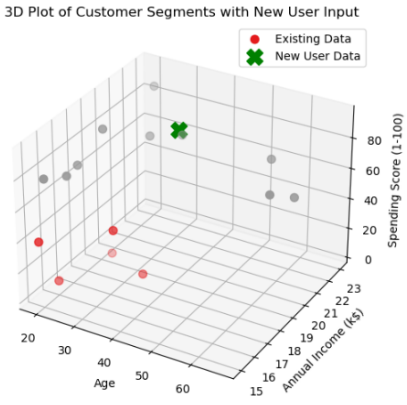
```
print("\nAccuracy Score:")

print(accuracy_score(y_test, y_pred))


# Step 7: Classify new user input

new_user_data = {'Age': [27], 'Annual Income (k$)': [23], 'Spending Score
(1-100)': [60]}

new_user_df = pd.DataFrame(new_user_data)

new_user_scaled = scaler.transform(new_user_df)


new_user_segment = knn.predict(new_user_scaled)

new_user_df['Segment'] = new_user_segment

print("\nNew User Data Prediction:")

print(new_user_df)


# Visualization: Scatter plot of the customer segments

plt.figure(figsize=(10, 6))

sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
hue='Segment', data=df, palette='Set1', marker='o', label='Existing Data')

sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
hue='Segment', data=new_user_df, palette='Set2', marker='X', s=200,
label='New User Data')

plt.title('Customer Segments with New User Input')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()


# Visualization: 3D plot for KNN decision boundaries and customer segments
including new user input

fig = plt.figure(figsize=(10, 6))

ax = fig.add_subplot(111, projection='3d')


# Plot the existing data with original values

ax.scatter(X['Age'], X['Annual Income (k$)'], X['Spending Score (1-100)'],
c=y, cmap='Set1', s=50, label='Existing Data')


# Plot the new user input with original values
```

```
ax.scatter(new_user_df['Age'], new_user_df['Annual Income (k$)'],
new_user_df['Spending Score (1-100)'], c='green', marker='X', s=200,
label='New User Data')

ax.set_xlabel('Age')

ax.set_ylabel('Annual Income (k$)')

ax.set_zlabel('Spending Score (1-100)')

plt.title('3D Plot of Customer Segments with New User Input')

ax.legend()

plt.show()

print('Viraj Joshi- 53004230033')
```

Output:-

```
Sample Data:
   Age  Annual Income (k$)  Spending Score (1-100)  Segment
0   19                  15                      39        0
1   21                  15                      81        1
2   20                  16                       6        0
3   23                  16                      77        1
4   31                  17                      40        0

Confusion Matrix:
[[1 0]
 [0 2]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3


Accuracy Score:
1.0

New User Data Prediction:
   Age  Annual Income (k$)  Spending Score (1-100)  Segment
0   27                  23                      60        1
```




Viraj Joshi- 53004230033

# Practical 7

Aim:- Implement the different Distance methods (Euclidean) with Prediction, Test Score and Confusion Matrix.

Theory:-

Distance metrics, also known as distance functions or proximity measures, are used in machine learning and statistics to calculate the similarity between two points or data samples. In this code snippet, we use three common distance metrics: Euclidean, Manhattan, and Chebyshev.

1. Euclidean Distance (L2 Norm)

The Euclidean distance is the straight-line distance between two points in n-dimensional space. It's calculated as the square root of the sum of the squared differences between corresponding features. Mathematically, it can be represented as:

$d(x, y) = \sqrt{((x1 - y1)^2 + (x2 - y2)^2 + ... + (xn - yn)^2)}$

2. Manhattan Distance (L1 Norm)

The Manhattan distance, also known as the Taxicab geometry or L1 distance, is the sum of the absolute differences between corresponding features. It's calculated as:

$d(x, y) = |x1 - y1| + |x2 - y2| + ... + |xn - yn|$

3. Chebyshev Distance (L∞ Norm)

The Chebyshev distance is the maximum absolute difference between corresponding features. It's calculated as:

$d(x, y) = \max(|x1 - y1|, |x2 - y2|, ..., |xn - yn|)$

Code:-

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification_report,confusion_matrix
```

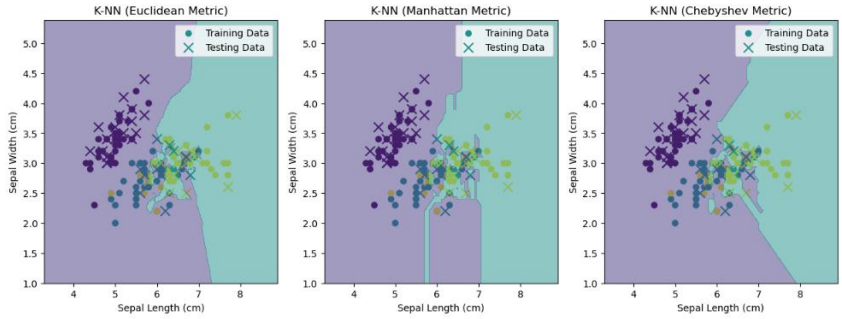
```
# Load the Iris dataset

iris = load_iris()

X = iris.data[:, :2]  # Select only the first two features (sepal length
and sepal width)

y = iris.target


# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Initialize k-NN classifier with different distance metrics

k = 3


# List of distance metrics to test

distance_metrics = ['euclidean', 'manhattan', 'chebyshev']


# Create subplots for each distance metric

fig, axes = plt.subplots(1, len(distance_metrics), figsize=(15, 5))

for i, metric in enumerate(distance_metrics):

    knn_classifier = KNeighborsClassifier(n_neighbors=k, metric=metric)


 # Fit the classifier to the training data

    knn_classifier.fit(X_train, y_train)

    # Make predictions on the test data

    y_pred = knn_classifier.predict(X_test)

    # Evaluate the classifier's performance

    print(f"Distance Metric: {metric}")

    print("Confusion Matrix:")

    print(confusion_matrix(y_test, y_pred))

    print("\nClassification Report:")

    print(classification_report(y_test, y_pred))

    print("\n")


    # Visualize the dataset and decision boundaries for the current metric

    ax = axes[i]
```

```
# Plot the training data points

    ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap='viridis',
label='Training Data')


    # Plot the testing data points

    ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap='viridis',
marker='x', s=100, label='Testing Data')

# Plot decision boundaries using the current metric

    knn_classifier = KNeighborsClassifier(n_neighbors=k, metric=metric)

    knn_classifier.fit(X, y)

    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1

    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min,
y_max, 0.01))

    Z = knn_classifier.predict(np.c_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)

    ax.contourf(xx, yy, Z, cmap='viridis', alpha=0.5, levels=range(4))

    ax.set_title(f'K-NN ({metric.capitalize()} Metric)')

    ax.set_xlabel('Sepal Length (cm)')

    ax.set_ylabel('Sepal Width (cm)')

    ax.legend()


plt.show()

print('Viraj Joshi- 53004230033')
```

Output:-

```
Distance Metric: euclidean
Confusion Matrix:
[[19  0  0]
 [ 0  7  6]
 [ 0  5  8]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.58      0.54      0.56        13
           2       0.57      0.62      0.59        13

    accuracy                           0.76        45
   macro avg       0.72      0.72      0.72        45
weighted avg       0.76      0.76      0.76        45
```

```
Distance Metric: manhattan
Confusion Matrix:
[[19  0  0]
 [ 0  7  6]
 [ 0  5  8]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.58      0.54      0.56        13
           2       0.57      0.62      0.59        13

    accuracy                           0.76        45
   macro avg       0.72      0.72      0.72        45
weighted avg       0.76      0.76      0.76        45

Distance Metric: chebyshev
Confusion Matrix:
[[19  0  0]
 [ 0  8  5]
 [ 0  7  6]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.53      0.62      0.57        13
           2       0.55      0.46      0.50        13

    accuracy                           0.73        45
   macro avg       0.69      0.69      0.69        45
weighted avg       0.73      0.73      0.73        45
```




Viraj Joshi- 53004230033

# Practical 8

Aim:- Implement the classification model using clustering for the following techniques with K means clustering with Prediction, Test Score and Confusion Matrix.

Theory:-

KMeans clustering is a type of unsupervised learning algorithm that partitions the data into K clusters based on their similarities. The main goal of KMeans clustering is to group similar data points together.

The KMeans clustering process works as follows:

Randomly Initialize K-Means Centroids : Initialize K centroids randomly from the dataset.

Assign Data Points to Their Closest Cluster : Assign each data point to its closest centroid based on a distance metric (e.g., Euclidean distance).

Update Cluster Centers : Update the cluster centers by calculating the mean of all data points assigned to that cluster.

Repeat Steps 2 and 3 Until Convergence : Repeat steps 2 and 3 until the cluster centers converge or reach a specified maximum number of iterations.

KMeans clustering is based on the following theoretical aspects:

Euclidean Distance : The Euclidean distance metric is used to assign data points to their closest centroid.

Centroid Update Rule : The centroid update rule is used to update the cluster centers by calculating the mean of all data points assigned to that cluster.

KMeans clustering is used as a pre-processing step for supervised learning. The idea behind this approach is to assign class labels to clusters based on the most frequent class label in each cluster. This can help improve the performance of the classifier by reducing the impact of irrelevant features.

Code:-

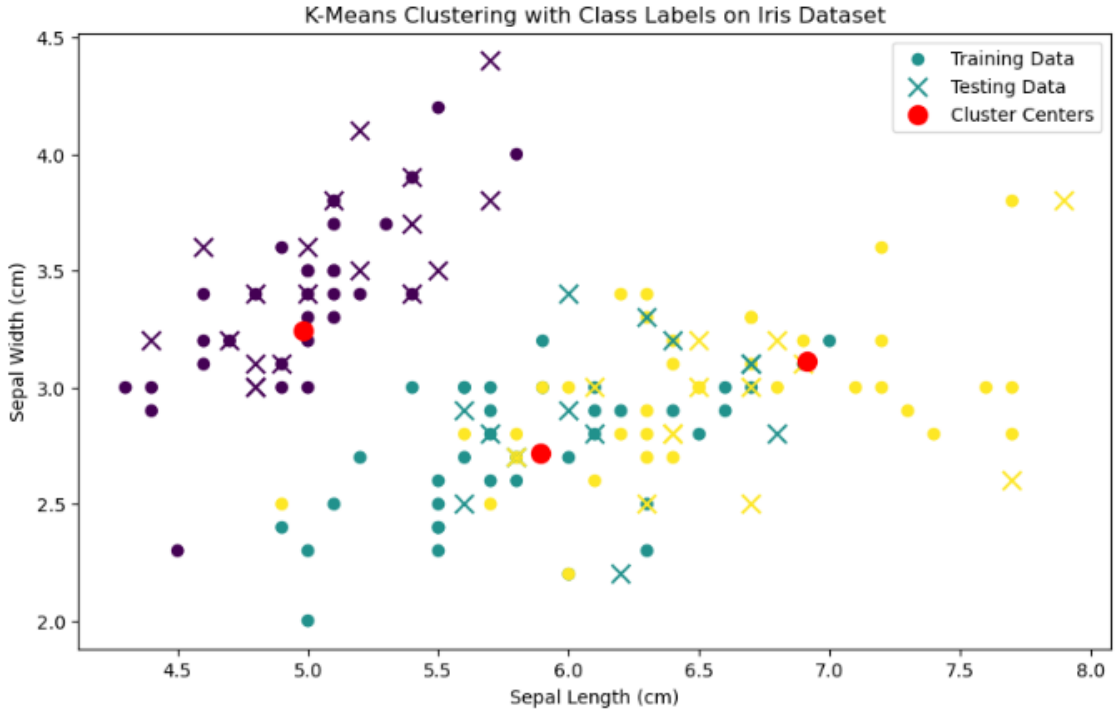
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.cluster import KMeans

from sklearn.metrics import classification_report, confusion_matrix


#Load the Iris dataset

iris = load_iris()
```

```
X = iris.data[:, :2] #Select only the features (sepal lengthy and sepal
width)

y = iris.target


#Split database into traini9ng and testing

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)


#Initalize K-Means clustering with the number of clusters equal to the
number of classes

n_clusters = len(np.unique(y))

kmeans = KMeans(n_clusters=n_clusters, random_state=42)


#Fit K-Means clustering to the training data

kmeans.fit(X_train)


#Assign cluster labels to data points in test set

cluster_labels = kmeans.predict(X_test)


#Assign class labels to clusters based on thge most frequent class label in
each cluster

cluster_class_labels = []

for i in range(n_clusters):

    cluster_indices = np.where(cluster_labels ==i)[0]


cluster_class_labels.append(np.bincount(y_test[cluster_indices]).argmax())


#Assign cluster class labels to data points in the test set

y_pred = np.array([cluster_class_labels[cluster_labels[i]] for i in
range(len(X_test))])


#Evaluate the classifier's performance

print("Confusion Matrix:")

print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")

print(classification_report(y_test, y_pred))
```

```
#Visualize the dataset and cluster cemters

plt.figure(figsize=(10, 6))


#Plot the training data points

plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap='viridis',
label='Training Data')


#Plot testing data

plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap='viridis',
marker='x', s=100, label='Testing Data')


#plt cluster centers

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
c='red', marker='o', s=100, label='Cluster Centers')


plt.xlabel('Sepal Length (cm)')

plt.ylabel('Sepal Width (cm)')

plt.title('K-Means Clustering with Class Labels on Iris Dataset')

plt.legend()

plt.show()

print("Viraj Joshi- 53004230033")
```

Output:-

```
Confusion Matrix:
[[19  0  0]
 [ 0  8  5]
 [ 0  5  8]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.62      0.62      0.62        13
           2       0.62      0.62      0.62        13

    accuracy                           0.78        45
   macro avg       0.74      0.74      0.74        45
weighted avg       0.78      0.78      0.78        45
```




Viraj Joshi- 53004230033

# Practical 9

Aim:- Implement the classification model using clustering for the following techniques with hierarchical clustering with Prediction, Test Score and Confusion Matrix.

Theory:-

Hierarchical clustering is a type of unsupervised learning algorithm that groups similar data points into clusters. It's also known as agglomerative clustering.

How Hierarchical Clustering Works

The hierarchical clustering process works as follows:

Start with Each Data Point as a Separate Cluster : Initially, each data point is placed in its own cluster.

Calculate the Distance Between All Pairs of Clusters : Calculate the distance between all pairs of clusters using a linkage method (e.g., single, complete, or ward).

Merge the Two Closest Clusters into One : Merge the two closest clusters into one new cluster.

Repeat Steps 2 and 3 Until Only One Cluster Remains : Repeat steps 2 and 3 until only one cluster remains.

Code:-

```
import pandas as pd

import numpy as np

from sklearn.cluster import AgglomerativeClustering

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

from sklearn.datasets import load_iris

import matplotlib.pyplot as plt

from scipy.cluster.hierarchy import dendrogram, linkage


#Load the Iris dataset

iris = load_iris()

X = iris.data

y = iris.target


#Step 1: Hierarchical Clustering with different Linkage Methods and Draw
denograms
```

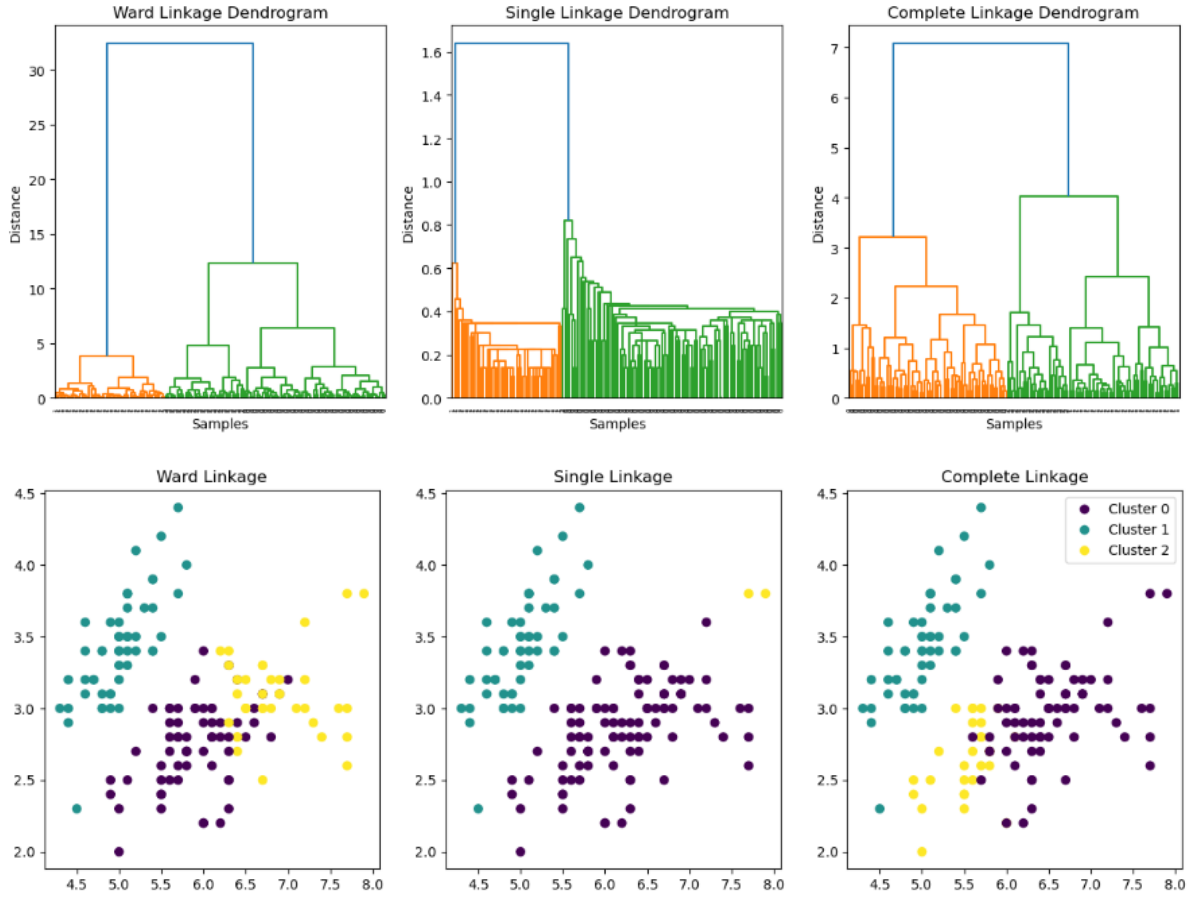
```
n_clusters = 3 # Number of clusters

linkage_methods = ['ward', 'single', 'complete'] # Different Linkage
methods

cluster_labels = []


#Define figure and axes for dendrograms

plt.figure(figsize=(15, 5))

dendrogram_axes = []


for i, linkage_method in enumerate(linkage_methods):

    labels = AgglomerativeClustering(n_clusters=n_clusters,
linkage=linkage_method).fit_predict(X)

    cluster_labels.append(labels)


#Create  a dendrgram for the current linkage method

    dendrogram_data = linkage(X, method=linkage_method)

    dendrogram_axes.append(plt.subplot(1, len(linkage_methods), i+1))

    dendrogram(dendrogram_data, orientation='top', labels=labels)

    plt.title(f"{linkage_method.capitalize()} Linkage Dendrogram")

    plt.xlabel('Samples')

    plt.ylabel('Distance')


#Plot clustering results for different linkage methods

plt.figure(figsize=(15, 5))

for i, linkage_method in enumerate(linkage_methods):

    plt.subplot(1, len(linkage_methods), i + 1)

    scatter = plt.scatter(X[:, 0], X[:, 1], c=cluster_labels[i],
cmap='viridis',

                          label=f'Clusters ({linkage_method.capitalize()}
Linkage)')

    plt.title(f"{linkage_method.capitalize()} Linkage")

#Add legend to scatter plots

plt.legend(handles=scatter.legend_elements()[0], labels=[f'Cluster {i}' for
i in range(n_clusters)])


#sTEP 2 :fEATURE ENGINEERING (uSING CLUSTER ASSIGNMENT AS A feature)
```

```
X_with_cluster = np.column_stack((X, cluster_labels[-1])) # using complete
linkage


#Step 3: Classification

X_train, X_test, y_train, y_test = train_test_split(X_with_cluster, y,
test_size=0.2, random_state=42)

classifier = RandomForestClassifier(n_estimators=100, random_state=42)

classifier.fit(X_train, y_train)


#Step 4: Prediction

y_pred = classifier.predict(X_test)


#Step 5 : Test Score and Confusion Matrix

accuracy = accuracy_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)


#Genrate classification report with zero_division parametrs

classification_rep = classification_report(y_test, y_pred, zero_division=0)


#Print cluster description

cluster_descriptions = {

    'ward': 'Clusters based on Ward linkage interpretation.',

    'single': 'Cluster based on Single linkage interpretation.',

    'complete': 'Clusters based on Complete linkage interpretation.'

}

for method in linkage_methods:

    print(f"Cluster Descriptions ({method.capitalize()} Linkage):")

    print(cluster_descriptions[method.lower()])  # Convert to lowercase for
dictionary access


# Print accuracy, confusion matrix, and classification report

print("Accuracy:", accuracy)

print("Confusion Matrix:\n", conf_matrix)

print("Classification Report:\n", classification_rep)


plt.show()
```

```
print("Viraj Joshi- 53004230033")
```

Output:-

```
Cluster Descriptions (Ward Linkage):
Clusters based on Ward linkage interpretation.
Cluster Descriptions (Single Linkage):
Cluster based on Single linkage interpretation.
Cluster Descriptions (Complete Linkage):
Clusters based on Complete linkage interpretation.
Accuracy: 1.0
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```




```
Viraj Joshi- 53004230033
```

# Practical 10

Aim:- Implement the Rule based method and test the same.

Theory:-

Rule-based methods are a type of decision-making approach that relies on predefined rules or conditions to make predictions.

The rule-based classifier function uses a series of if-elif-else statements to evaluate the input feature vector against a set of predefined conditions. The function returns the class assignment based on which condition is met.

If feature 2 < 2.0 : If the value of feature 2 (petal length) is less than 2.0, the function assigns the input data point to Class 0.

elif feature 3 > 1.5 : If the value of feature 3 (petal width) is greater than 1.5 and the value of feature 2 is greater than or equal to 2.0, the function assigns the input data point to Class 2.

else : In all other cases, the function assigns the input data point to Class 1.

Code:-

```
import numpy as np

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report


#Load the Iris dataset

iris = load_iris()

X = iris.data

y = iris.target


#Split the data for testing

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


#Define a simple rule-based classifier function

def rule_based_classifier(x):

    if x[2] < 2.0:

        rule = "If feature 2 < 2.0, assign to Classd 0"

        return 0 # Class 0

    elif x[3] > 1.5:
```

```
        rule = "If feature 2 >= 2.0 and feature 3 > 1.5, assign to Class 2"
        return 2 # Class 2
    else:
        rule = "If feature 2 >= 2.0 and feature 3 <=1.5, assign to Class 1"
        return 1 # Class 1
    print("Rule:", rule)

# Apply the rule-based classifier to make predictions on the test set
y_pred = [rule_based_classifier(x) for x in X_test]

# Calculate accuracy, confusion matrix, and classification report
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred,
target_names=iris.target_names)

# Print the results
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", classification_rep)
print('Viraj Joshi- 53004230033')
```

Output:-

```
Accuracy: 0.9666666666666667
Confusion Matrix:
 [[10  0  0]
 [ 0  8  1]
 [ 0  0 11]]
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      0.89      0.94         9
   virginica       0.92      1.00      0.96        11

    accuracy                           0.97        30
   macro avg       0.97      0.96      0.97        30
weighted avg       0.97      0.97      0.97        30

Viraj Joshi- 53004230033
```

# Practical 11

Aim:- Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.

Theory:-

Bayesian networks are a powerful tool for modeling complex relationships between variables in a probabilistic framework. Here's an overview of the theory behind Bayesian networks:

Key Concepts

Probabilistic Graphical Models (PGMs) : PGMs provide a unified framework for representing and reasoning about uncertain information.

Conditional Independence : The core idea behind Bayesian networks is that variables are conditionally independent given their parents in the network.

Bayesian Network Structure

A Bayesian network consists of:

Nodes : Representing the variables of interest (e.g., Age, Gender, ChestPain).

Edges : Indicating the relationships between variables.

Conditional Probability Tables (CPDs) : Describing the conditional probability distributions for each variable given its parents.

Key Components

Directed Acyclic Graphs (DAGs) : Bayesian networks are represented as DAGs, ensuring that there is no directed cycle in the graph.

Parents and Children : Each node has a set of parent nodes, which influence it, and child nodes, which are influenced by it.

Conditional Independence : Given the parents of a node, its children are conditionally independent.

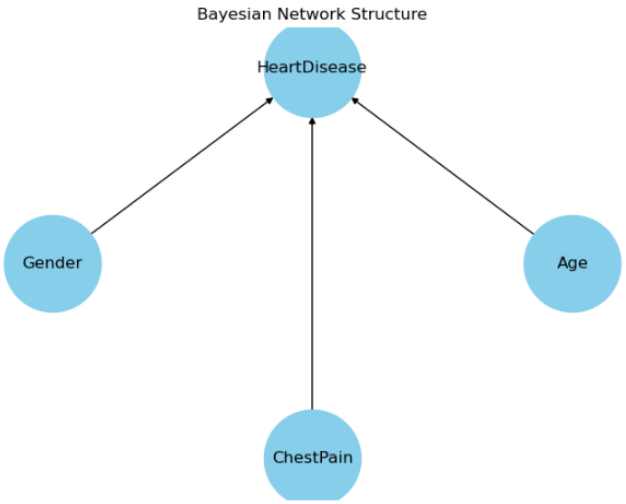
Code:-

```
import numpy as np

import pandas as pd

from pgmpy.models import BayesianNetwork

from pgmpy.estimators import ParameterEstimator, MaximumLikelihoodEstimator

from pgmpy.inference import VariableElimination

import networkx as nx

import matplotlib.pyplot as plt


data = pd.DataFrame (data={'Age': [30, 40, 50, 60, 70],
```

```
                            'Gender': ['Male', 'Female', 'Male', 'Female',
'Male'],
                            'ChestPain': ['Typical', 'Atypical', 'Typical',
'Atypical', 'Typical'],
                            'HeartDisease': ['Yes', 'No', 'Yes', 'No',
'Yes']})
model = BayesianNetwork([('Age', 'HeartDisease'),
                         ('Gender', 'HeartDisease'),
                         ('ChestPain', 'HeartDisease')])


model.fit(data, estimator=MaximumLikelihoodEstimator)


pos = nx.circular_layout(model)
nx.draw(model, pos, with_labels=True, node_size=5000, node_color="skyblue",
font_size=12, font_color="black")
print('Viraj Joshi- 53004230033')
plt.title("Bayesian Network Structure")
plt.show()


for cpd in model.get_cpds():
    print("CPD of", cpd.variable)
    print(cpd)


inference = VariableElimination(model)
query = inference.query(variables=['HeartDisease'], evidence={'Age':50,
'Gender': 'Male', 'ChestPain': 'Typical'})
print(query)
```

Output:-

```
Viraj Joshi- 53004230033
```




```
CPD of Age
+---------+-----+
| Age(30) | 0.2 |
+---------+-----+
| Age(40) | 0.2 |
+---------+-----+
| Age(50) | 0.2 |
+---------+-----+
| Age(60) | 0.2 |
+---------+-----+
| Age(70) | 0.2 |
+---------+-----+
CPD of HeartDisease
+------------------+--------------------+-----+--------------------+--------------------+
| Age              | Age(30)            | ... | Age(70)            | Age(70)            |
+------------------+--------------------+-----+--------------------+--------------------+
| ChestPain        | ChestPain(Atypical)| ... | ChestPain(Typical) | ChestPain(Typical) |
+------------------+--------------------+-----+--------------------+--------------------+
| Gender           | Gender(Female)     | ... | Gender(Female)     | Gender(Male)       |
+------------------+--------------------+-----+--------------------+--------------------+
| HeartDisease(No) | 0.5                | ... | 0.5                | 0.0                |
+------------------+--------------------+-----+--------------------+--------------------+
| HeartDisease(Yes)| 0.5                | ... | 0.5                | 1.0                |
+------------------+--------------------+-----+--------------------+--------------------+
```

```
CPD of Gender
+----------------+-----+
| Gender(Female) | 0.4 |
+----------------+-----+
| Gender(Male)   | 0.6 |
+----------------+-----+
CPD of ChestPain
+---------------------+-----+
| ChestPain(Atypical) | 0.4 |
+---------------------+-----+
| ChestPain(Typical)  | 0.6 |
+---------------------+-----+
+------------------+---------------------+
| HeartDisease     |   phi(HeartDisease) |
+==================+=====================+
| HeartDisease(No) |              0.0000 |
+------------------+---------------------+
| HeartDisease(Yes)|              1.0000 |
+------------------+---------------------+
```

# Practical 12

Aim:- Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

Theory:-

Locally Weighted Regression is a non-parametric regression technique that generalizes the traditional linear regression model by considering weighted data points. Here's an overview of the theory behind LWR:

Key Concepts

Non-Parametric Models : Unlike parametric models, which assume a specific functional form for the relationship between variables, non-parametric models do not make such assumptions.

Weighted Least Squares (WLS) : WLS is an extension of traditional least squares regression that assigns different weights to each data point based on their proximity to the prediction point.

Locally Weighted Regression Algorithm

The LWR algorithm consists of the following steps:

Calculate Weights : At a given query point, calculate the weights for all data points using a kernel function (e.g., Gaussian or Epanechnikov). The weights measure the proximity of each data point to the query point.

Weighted Least Squares : Perform weighted least squares regression on the weighted data points to estimate the model parameters.

Prediction : Use the estimated model parameters and the weights to predict the value at the query point.

Key Components

Kernel Function : The kernel function is used to calculate the weights for each data point. Commonly used kernels include:

Gaussian: exp(-((x - x0) / (2 * sigma))^2)

Epanechnikov: (1 - ((x - x0) / (sigma))^2) for -1 <= (x - x0) / (sigma)^2 <= 1, otherwise, 0

Weighted Least Squares : The weighted least squares regression estimates the model parameters using a weighted sum of squared residuals.

Model Complexity : LWR allows for flexible modeling of complex relationships by considering data points with varying weights.

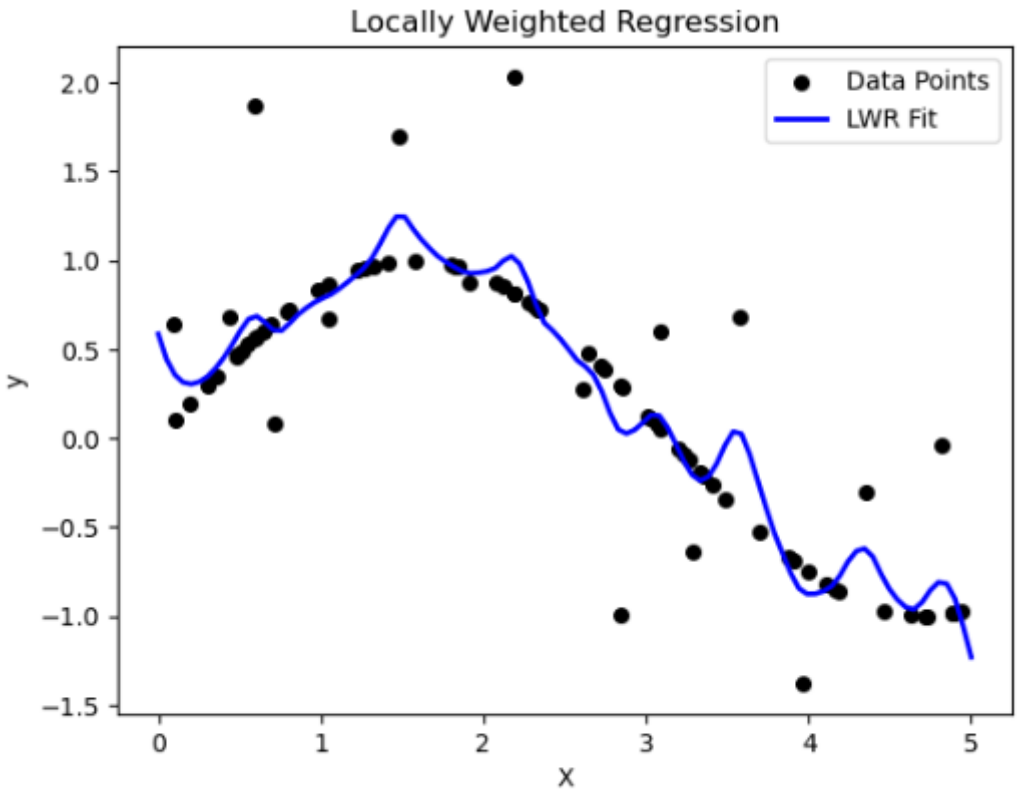
Code:-

```
import numpy as np

import matplotlib.pyplot as plt
```

```
# Seed for reproducibility
np.random.seed(0)

# Generate random dataset
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - np.random.rand(16))

# Locally Weighted Regression function
def locally_weighted_regression(query_point, X, y, tau=0.1):
    m = X.shape[0]
    # Calculate weights
    weights = np.exp(-((X - query_point) * 2).sum(axis=1) / (2 * tau * 2))
    W = np.diag(weights)

    # Add bias term to X
    X_bias = np.c_[np.ones((m, 1)), X]

    # Calculate theta using weighted least squares
    theta =
np.linalg.inv(X_bias.T.dot(W).dot(X_bias)).dot(X_bias.T).dot(W).dot(y)

    # Predict for query_point
    x_query = np.array([1, query_point])
    prediction = x_query.dot(theta)
    return prediction

# Generate test points
X_test = np.linspace(0, 5, 100)

# Predict using locally weighted regression
predictions = [locally_weighted_regression(query_point, X, y, tau=0.1) for
query_point in X_test]

# Plot results
```

```
plt.scatter(X, y, color='black', s=30, marker='o', label='Data Points')
plt.plot(X_test, predictions, color='blue', linewidth=2, label='LWR Fit')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Locally Weighted Regression')
plt.legend()
plt.show()
print('Viraj Joshi- 53004230033')
```

Output:-




```
Viraj Joshi- 53004230033
```

# Practical 13

Aim:- Implement ANN to solve the XOR problem using forward/ backward propagation and sigmoid activation function.

Theory:-

Artificial Neural Networks (ANNs) is a mathematical model inspired by the structure and function of biological neural networks. It consists of interconnected nodes or "neurons" that process information in parallel.

Basic Components

A basic ANN consists of three types of layers:

Input Layer : Receives input data from the environment.

Hidden Layer(s) : Processes complex representations of the input data through nonlinear transformations.

Output Layer : Generates the final output based on the processed data.

Activation Functions

To introduce nonlinearity and enable the network to learn complex patterns, activation functions are applied at each layer:

Sigmoid (Logistic) Function : Maps any real-valued number to a value between 0 and 1.

ReLU (Rectified Linear Unit) : Maps all negative values to 0 and leaves positive values unchanged.

Backpropagation Algorithm

The backpropagation algorithm is an efficient way to train ANNs by minimizing the difference between predicted outputs and actual outputs. It involves:

Forward Pass : The network receives input data, propagates it through the hidden layers, and generates output.

Error Computation : The difference between actual output and predicted output is computed for each example in the dataset.

Backward Pass (Propagation) : The error is propagated backward through the network to adjust the weights of the connections between neurons.

Code:-

```
import numpy as np

import matplotlib.pyplot as plt

def sigmoid(x):

    return 1/(1+np.exp(-x))
```

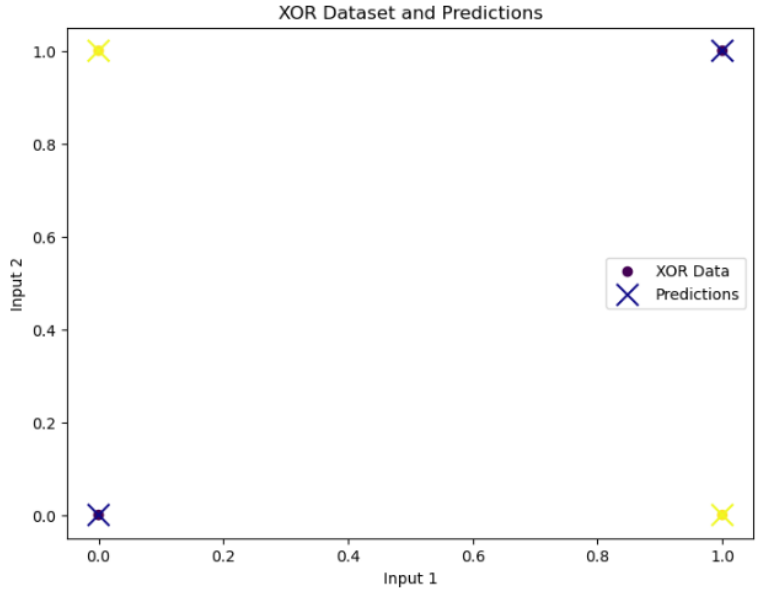
```
def sigmoid_derivative(x):
    return x*(1-x)


class NeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size):
        self.weights_input_hidden = np.random.uniform(size=(input_size,
hidden_size))
        self.weights_hidden_output = np.random.uniform(size=(hidden_size,
output_size))



    def forward(self, X):
        self.hidden_input = np.dot(X, self.weights_input_hidden)
        self.hidden_output = sigmoid(self.hidden_input)
        self.output = sigmoid(np.dot(self.hidden_output,
self.weights_hidden_output))
        return self.output



    def backward(self, X, y, learning_rate):
        error_output = y-self.output
        delta_output = error_output*sigmoid_derivative(self.output)


        error_hidden = delta_output.dot(self.weights_hidden_output.T)
        delta_hidden = error_hidden*sigmoid_derivative(self.hidden_output)


        self.weights_hidden_output +=
self.hidden_output.T.dot(delta_output)*learning_rate
        self.weights_input_hidden += X.T.dot(delta_hidden)*learning_rate



    def train(self, X, y, learning_rate, epochs):
        self.loss_history = []
        for _ in range(epochs):
            output = self.forward(X)
            error = y-output
```

```
            self.loss_history.append(np.mean(error**2))
            self.backward(X,y,learning_rate)


    def predict(self, X):
        return self.forward(X)


X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([[0], [1], [1], [0]])

input_size = 2
hidden_size = 4
output_size = 1
learning_rate = 0.1
epochs = 10000

nn = NeuralNetwork(input_size, hidden_size, output_size)
nn.train(X, y, learning_rate, epochs)

predictions = nn.predict(X)

plt.figure(figsize=(8, 6))
plt.scatter(X[:,0], X[:,1], c=y, cmap='viridis', label='XOR Data')
plt.scatter(X[:,0], X[:,1], c=np.round(predictions), cmap='plasma',
marker='x', s=200, label='Predictions')
plt.title('XOR Dataset and Predictions')
plt.xlabel('Input 1')
plt.ylabel('Input 2')
plt.legend()

for i in range(len(X)):
    print(f"Input: {X[i]}, Actual: {y[i]}, Predicted:
{np.round(predictions[i])}")
```

```
plt.show()

print('Viraj Joshi- 53004230033')
```

Output:-

```
Input: [0 0], Actual: [0], Predicted: [0.]
Input: [0 1], Actual: [1], Predicted: [1.]
Input: [1 0], Actual: [1], Predicted: [1.]
Input: [1 1], Actual: [0], Predicted: [0.]
```




```
Viraj Joshi- 53004230033
```

# INDEX

## Introduction to Security Operations and Center

Overview of Security Operations Center (SOC):- A Security Operations Center (SOC) is a centralized unit within an organization that focuses on monitoring, analysing, and responding to cybersecurity incidents. It functions as a command center for managing and defending against cyber threats in real-time. The primary purpose of a SOC is to identify, analyse, and mitigate security incidents using a combination of technology, processes, and skilled security personnel.

Key features of a SOC include:

1. **Monitoring and Detection**: The SOC continuously monitors an organization's network, systems, and applications to detect potential security threats or vulnerabilities.
2. **Incident Response**: When a potential threat is detected, the SOC team assesses the severity of the incident and coordinates an appropriate response, which may include containment, eradication, and recovery efforts.
3. **Threat Intelligence**: SOC teams use threat intelligence feeds and advanced analytics tools to stay informed of the latest cyber threats and attack techniques.
4. **Forensics and Analysis**: In case of an attack, the SOC performs detailed analysis and digital forensics to determine the scope of the breach, the method used, and how to prevent future incidents.
5. **Continuous Improvement**: The SOC constantly evaluates and enhances security protocols, ensuring they adapt to evolving threats and vulnerabilities.

A SOC typically uses tools such as Security Information and Event Management (SIEM) systems, Intrusion Detection Systems (IDS), and endpoint detection solutions to keep an organization’s digital assets safe.

Role and Responsibilities of SOC Analysts:-

SOC analysts are responsible for monitoring, detecting, and responding to cybersecurity threats. Key duties include:

1. **Monitoring**: Continuously track security alerts and events.
2. **Incident Detection**: Investigate potential security breaches.
3. **Incident Response**: Contain and mitigate threats during security incidents.
4. **Threat Hunting**: Proactively search for vulnerabilities and suspicious activities.
5. **Vulnerability Management**: Identify and help fix weaknesses.
6. **Threat Intelligence**: Stay updated on emerging threats.
7. **Forensics**: Analyse breaches to understand attack methods and impact.
8. **Reporting**: Document incidents and responses for future reference.

SOC Architecture and Components:- The architecture of a Security Operations Center (SOC) includes various components and systems designed to monitor, detect, and respond to cyber

threats. It's structured to ensure real-time threat detection, efficient incident response, and ongoing security management.

**Key Components of SOC Architecture:**

**1. Security Information and Event Management (SIEM)**

- Centralizes log collection and analysis.
- Correlates data from various systems to detect security incidents in real-time.

**2. Intrusion Detection/Prevention Systems (IDS/IPS)**

- Monitors network traffic to detect and block malicious activities.

**3. Firewalls**

- Act as the first line of defense, controlling incoming and outgoing traffic based on security rules.

**4. Endpoint Detection and Response (EDR)**

- Monitors and protects endpoints (computers, mobile devices) from threats.
- Provides real-time detection and automated response to threats on endpoints.

**5. Threat Intelligence Platform (TIP)**

- Gathers and provides insights on emerging threats, vulnerabilities, and attack vectors.
- Helps SOC analysts make informed decisions on threat mitigation.

**6. Security Orchestration, Automation, and Response (SOAR)**

- Automates routine tasks, incident response, and workflow processes.
- Integrates different security tools for efficient incident handling.

**7. Vulnerability Management Tools**

- Scans the network and systems for vulnerabilities and potential exploits.

**8. Forensic Tools**

- Helps in post-incident investigation to determine the cause and extent of breaches.

**9. Network Monitoring Tools**

- Continuously monitors traffic for abnormal patterns or potential security risks.

**10. User and Entity Behavior Analytics (UEBA)**

- Detects unusual behavior from users or entities within the system, helping to identify insider threats or compromised accounts.

**11. Incident Response Platform (IRP)**

- Centralizes incident management and tracks the lifecycle of incidents from detection to resolution.

These components work together to form a comprehensive system for proactive cybersecurity management

Incident Response Frameworks (e.g., NIST, SANS):-

Incident Response Frameworks provide structured approaches for handling cybersecurity incidents. Two widely used frameworks are:

**1. NIST (National Institute of Standards and Technology):**

- **Preparation**: Establish incident response capabilities.
- **Detection & Analysis**: Identify and assess incidents.
- **Containment, Eradication & Recovery**: Limit damage, remove threats, and restore systems.
- **Post-Incident Activity**: Review and improve processes.

**2. SANS:**

- **Preparation**: Plan and build response capabilities.
- **Identification**: Detect and confirm incidents.
- **Containment**: Isolate affected systems.
- **Eradication**: Remove the threat.
- **Recovery**: Restore operations.
- **Lessons Learned**: Analyze and improve for future incidents.

Both frameworks emphasize preparation, quick containment, and continuous improvement.

Introduction to Security Information and Event Management (SIEM) Systems:- Security Information and Event Management (SIEM) systems are tools that collect, analyze, and correlate security data from various sources across an organization. They provide real-time threat detection, incident response, and compliance reporting.

**Key Functions:**

- **Data Aggregation**: Collects logs and event data from networks, devices, and applications.
- **Correlation**: Identifies patterns and anomalies by analyzing data.
- **Alerting**: Generates alerts for suspicious activities.
- **Incident Response**: Helps in investigating and responding to security incidents.
- **Compliance**: Assists in meeting regulatory requirements by tracking security events.

SIEM helps organizations maintain visibility, detect threats, and respond to incidents effectively.

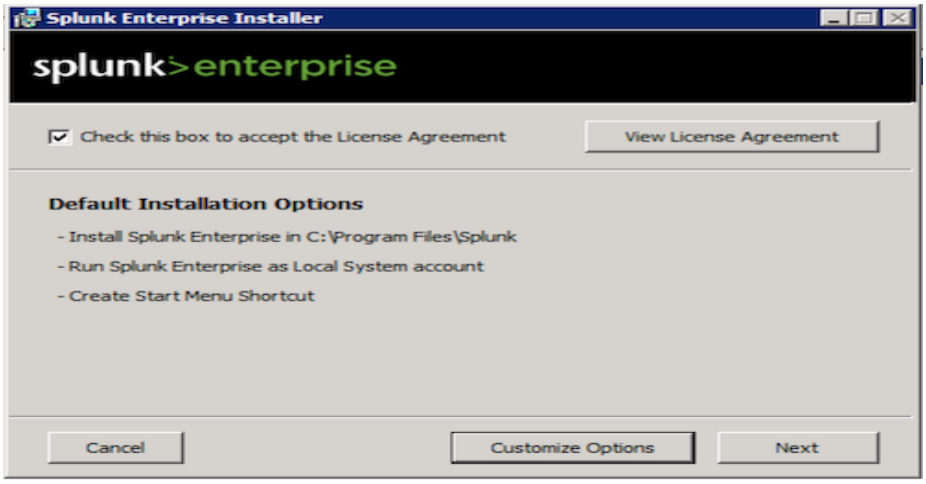
Practical 1

Aim:- To install Splunk enterprise via the GUI installer.

The Windows installer is an MSI file.

**Begin the installation**

1. Download the Splunk installer from the Splunk download page.
2. To start the installer, double-click the splunk.msi file. The installer runs and displays the **Splunk Enterprise Installer** panel.




3. To continue the installation, check the "Check this box to accept the License Agreement" checkbox. This activates the "Customize Installation" and "Next" buttons.
4. (Optional) If you want to view the license agreement, click **View License Agreement**.

**Installation Options**

The Windows installer gives you two choices: Install with the default installation settings, or configure all settings prior to installing.

When you choose to install with the default settings, the installer does the following:

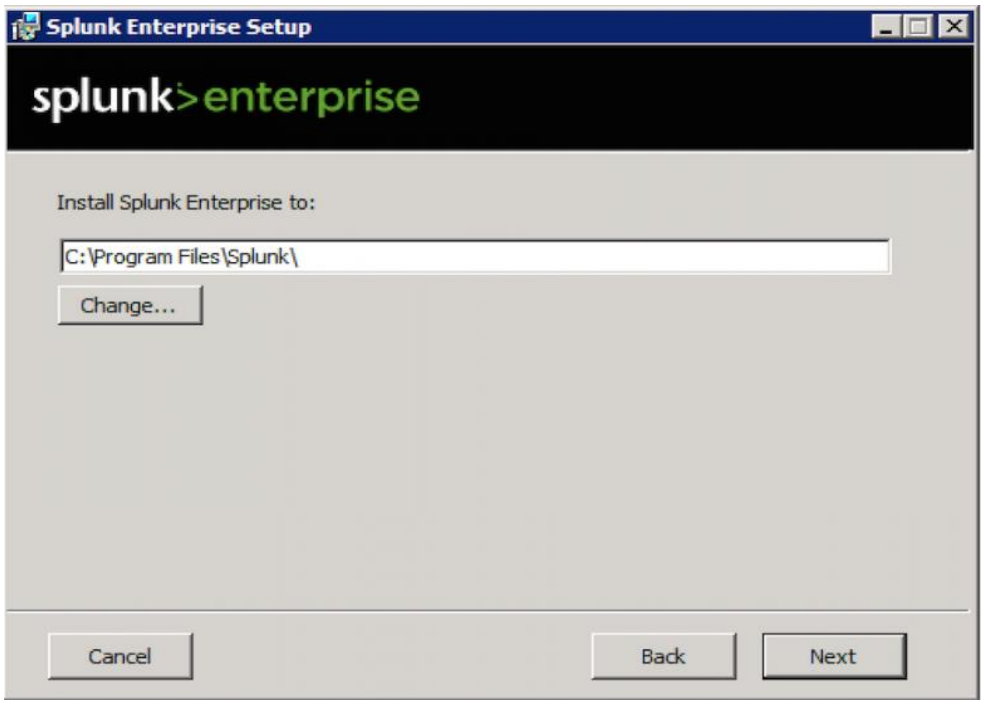
- Installs Splunk Enterprise in \Program Files\Splunk on the drive that booted your Windows machine.
- Installs Splunk Enterprise with the default management and Web network ports.
- Configures Splunk Enterprise to run as the Local System user.
- Prompts you to create a Splunk administrator password. You must do this before installation can continue.
- Creates a Start Menu shortcut for the software.

If you want to change any of these default installation settings, click **Customize Options** and proceed with the instructions in "Customize Options" in this topic.

Otherwise, click **Next**. You will be prompted for a password for the Splunk admin user. After you supply a password, installation begins and you can continue with the "Complete the install" instructions later in this topic.

**Customize options during the installation**

You can customize several options during the installation. When you choose to customize options, the installer displays the "Install Splunk Enterprise to" panel.




By default, the installer puts Splunk Enterprise into \Program Files\Splunk on the system drive. This documentation set refers to the Splunk Enterprise installation directory as $SPLUNK_HOME or %SPLUNK_HOME%.

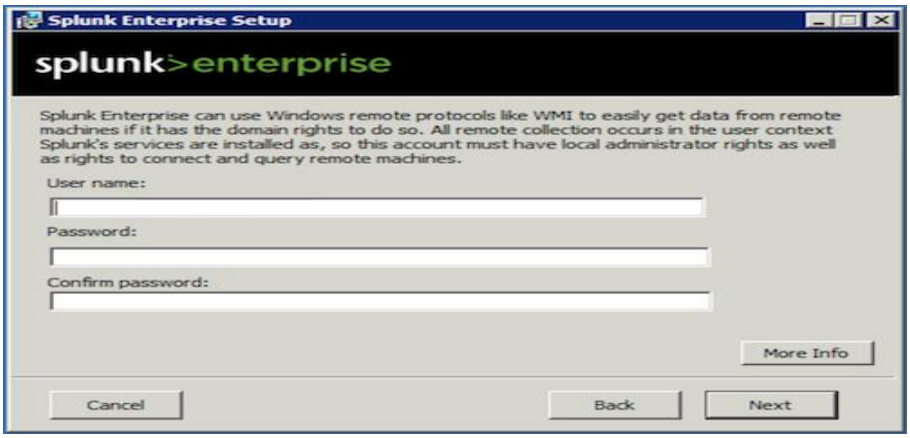
Splunk Enterprise installs and runs two Windows services, splunkd and splunkweb. The splunkd service handles all Splunk Enterprise operations, and the splunkweb service installs to run only in legacy mode.

These services install and run as the user you specify on the "Choose the user Splunk Enterprise should run as" panel. You can choose to run Splunk Enterprise as the Local System user, or another user.

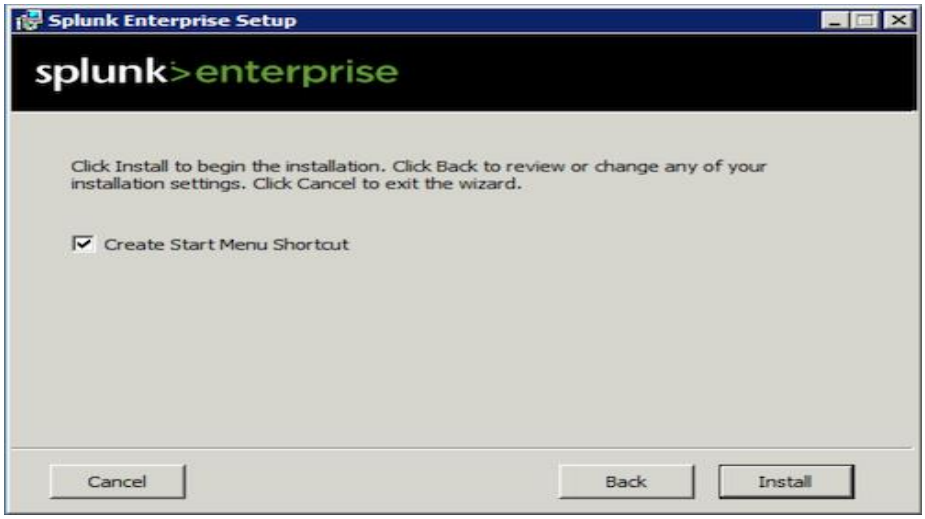
When the installer asks you the user that you want to install Splunk Enterprise as, you must specify the user name in domain\username format. The user must be a valid user in your security context, and must be an active member of an Active Directory domain. Splunk Enterprise must run under either the Local System account or a valid user account with a valid password and local administrator privileges. Failure to include the domain name with the user will cause the installation to fail.

1. Click **Change…** to specify a different location to install Splunk Enterprise, or click **Next** to accept the default value. The installer displays the "Choose the user Splunk Enterprise should run as" panel.


2. Select a user type and click **Next**.
3. If you selected the Local System user, proceed to Step 5. Otherwise, the installer displays the **Logon Information: specify a username and password** panel.



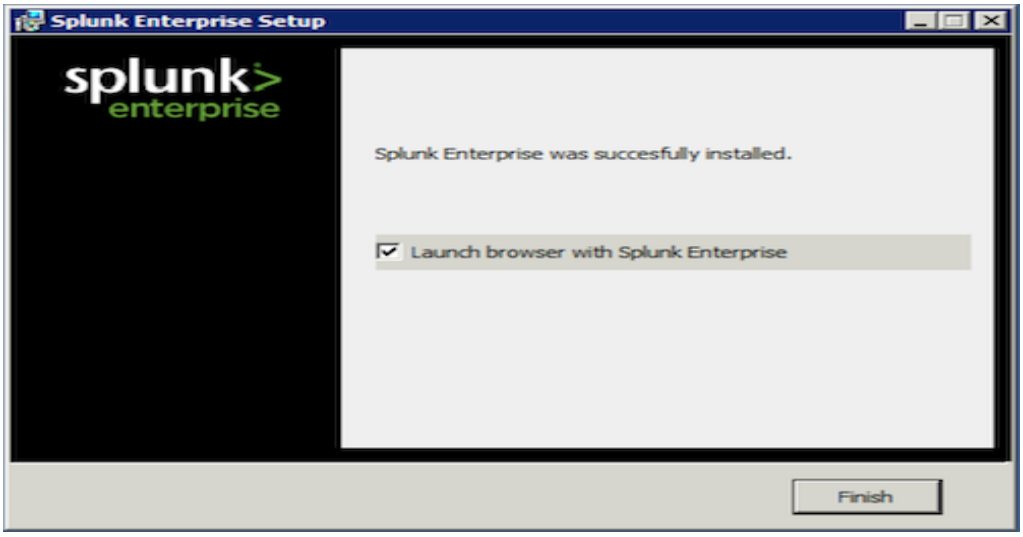

4. Enter the Windows credentials that Splunk Enterprise uses to run on the machine and click **Next**.
5. Create credentials for the Splunk administrator user by entering a username and password that meets the minimum eligibility requirements as shown in the panel and click **Next**.
6. The installer displays the installation summary panel.




7. Click "Install" to proceed with the installation.

**Complete the installation**

The installer runs, installs the software, and displays the **Installation Complete** panel.




Click **Finish**. The installation completes, Splunk Enterprise starts and launches in a supported browser.

Practical 2

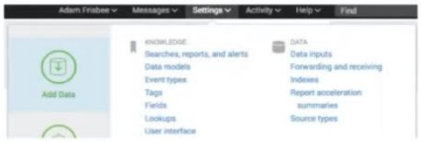
Aim:- To upload data on splunk and learn how it consumes data.

## How Splunk Consumes Data >

- Virtual Machines
- Physical Machines
- Servers
- IoT
- Communications


- Logs
- Configurations
- Scripts
- Tickets
- Alerts

## How Splunk Consumes Data >




**upload**
files from my computer

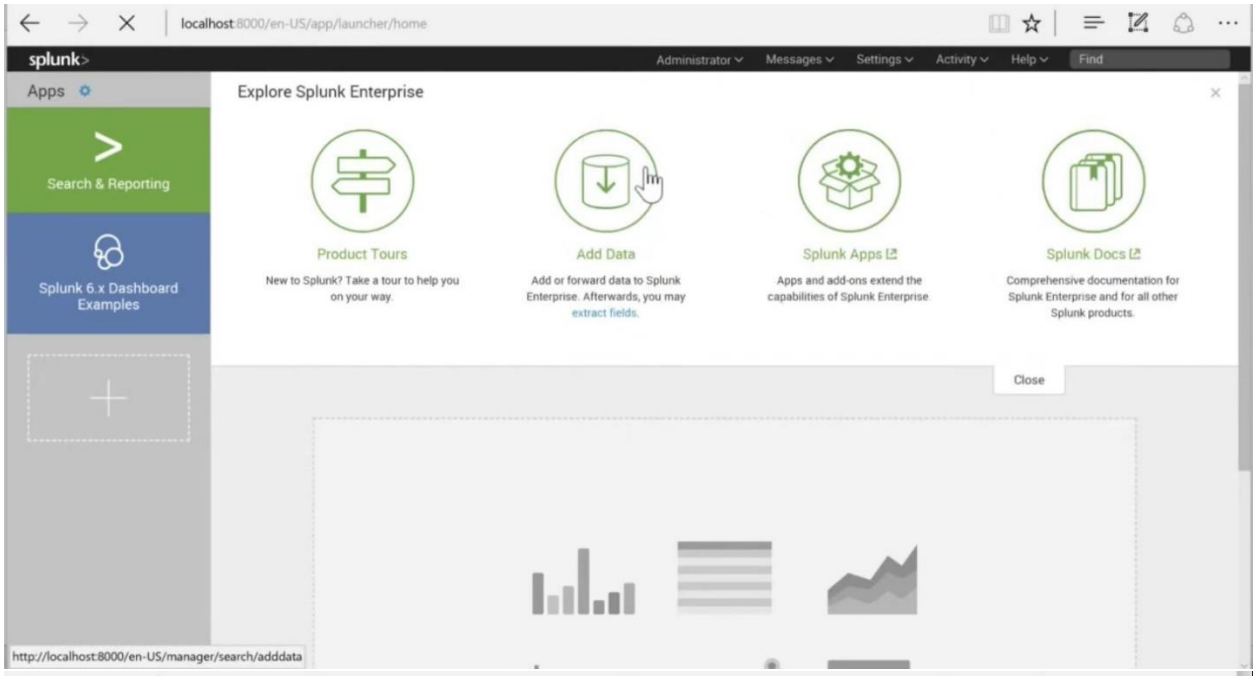
Local log files
Local structured files (e.g. CSV)
Tutorial for adding data

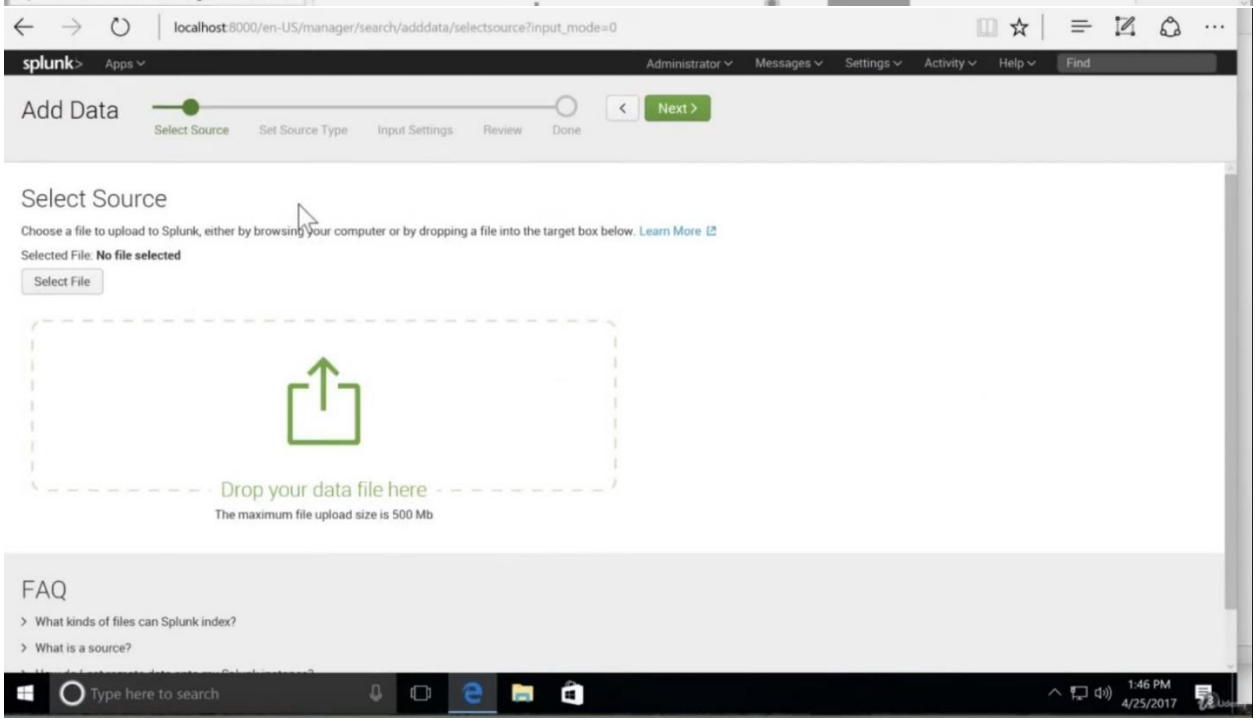
**monitor**
files and ports on this Splunk indexer

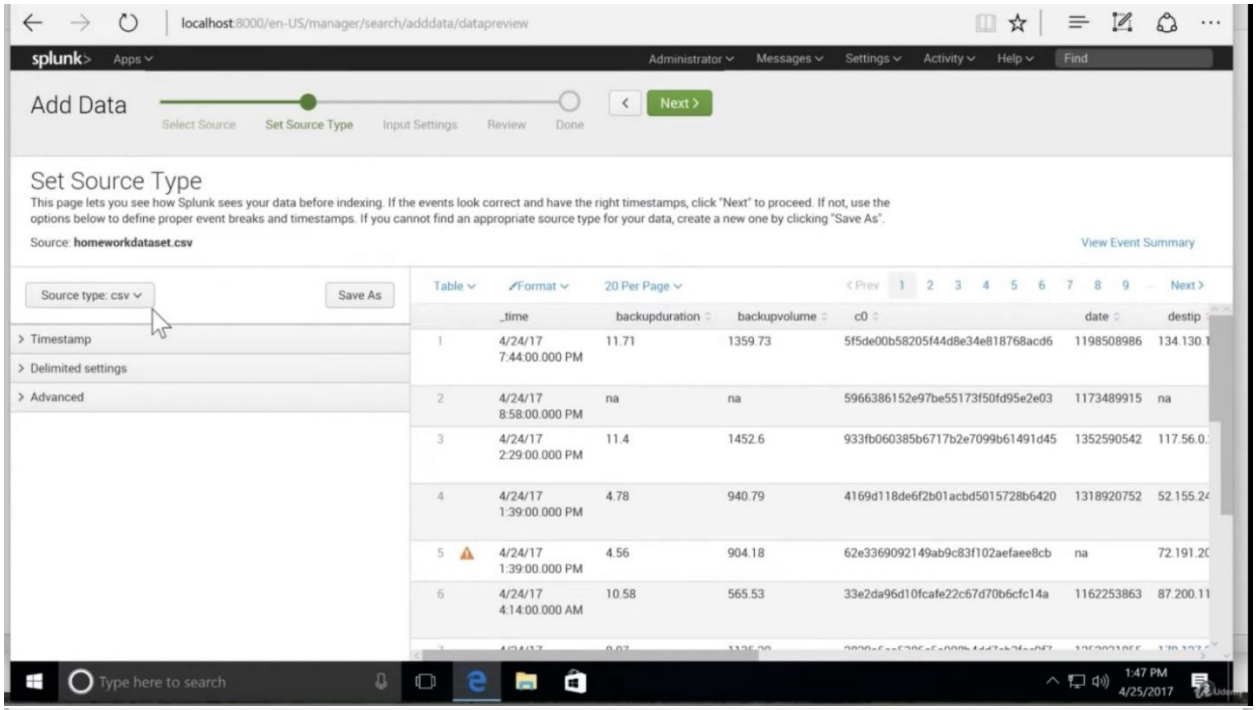
Files - WMI - TCP/UDP - Scripts
Modular inputs for external data sources

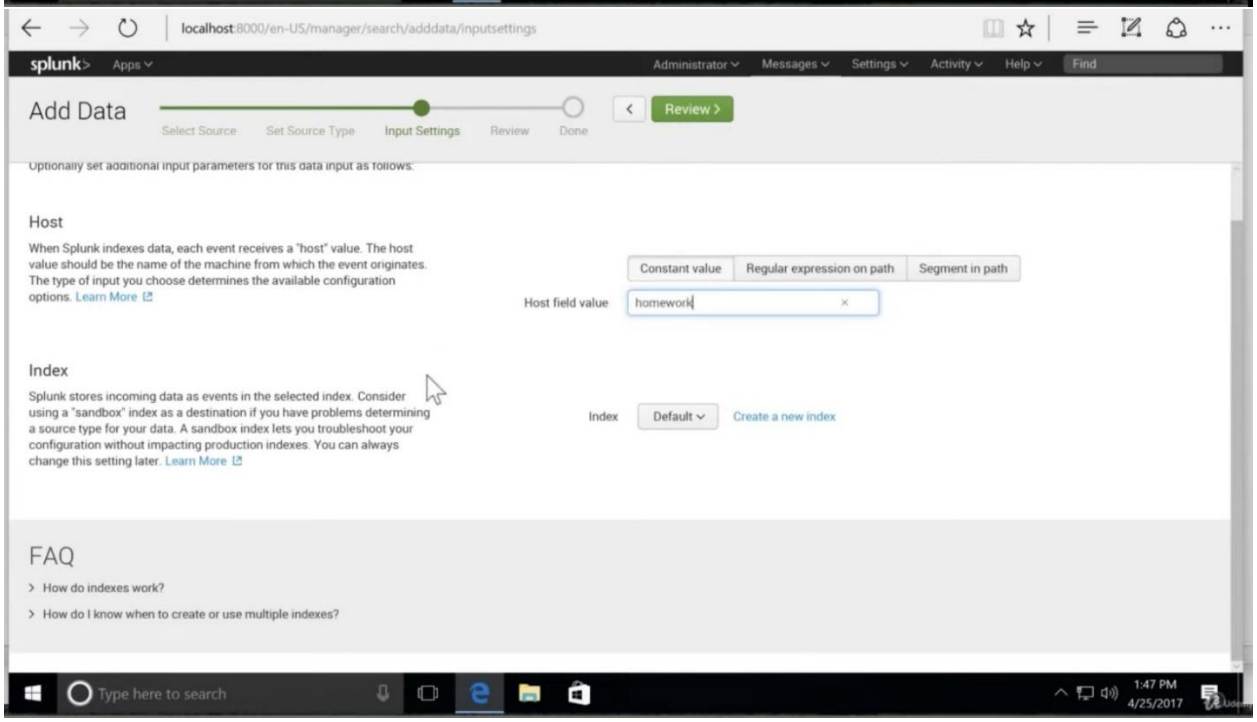
**forward**
data from Splunk forwarder

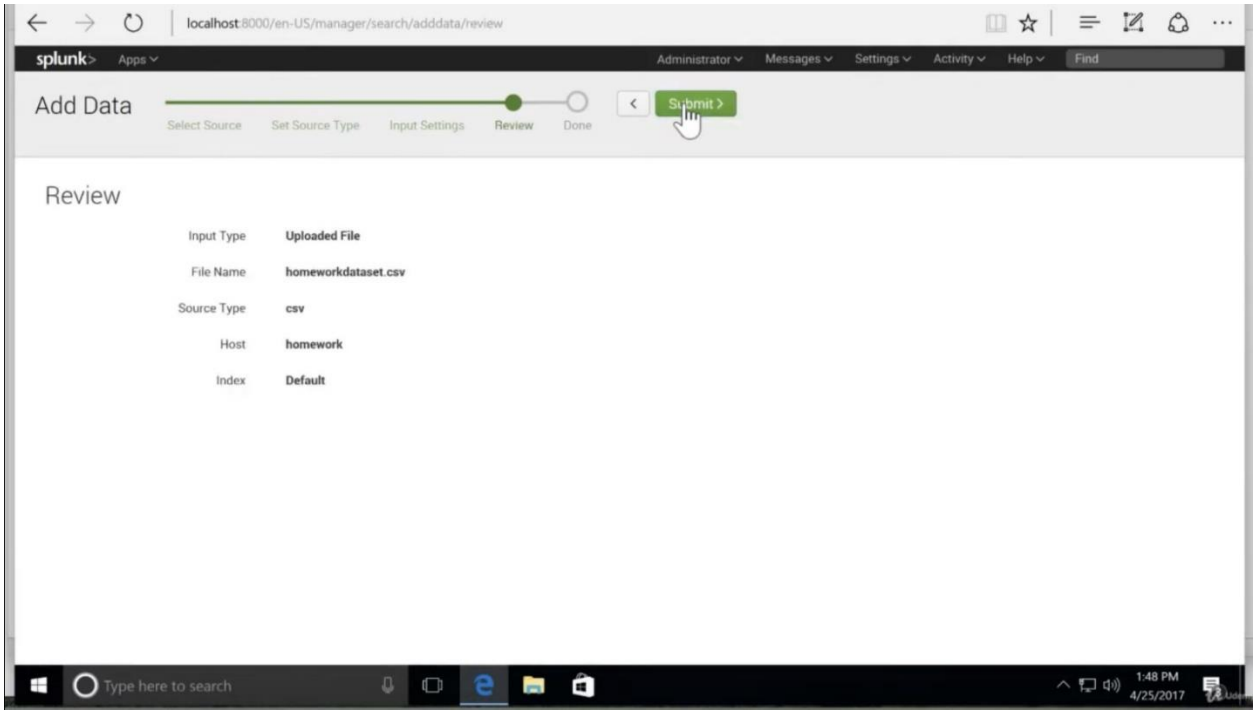
Files - TCP/UDP - Scripts

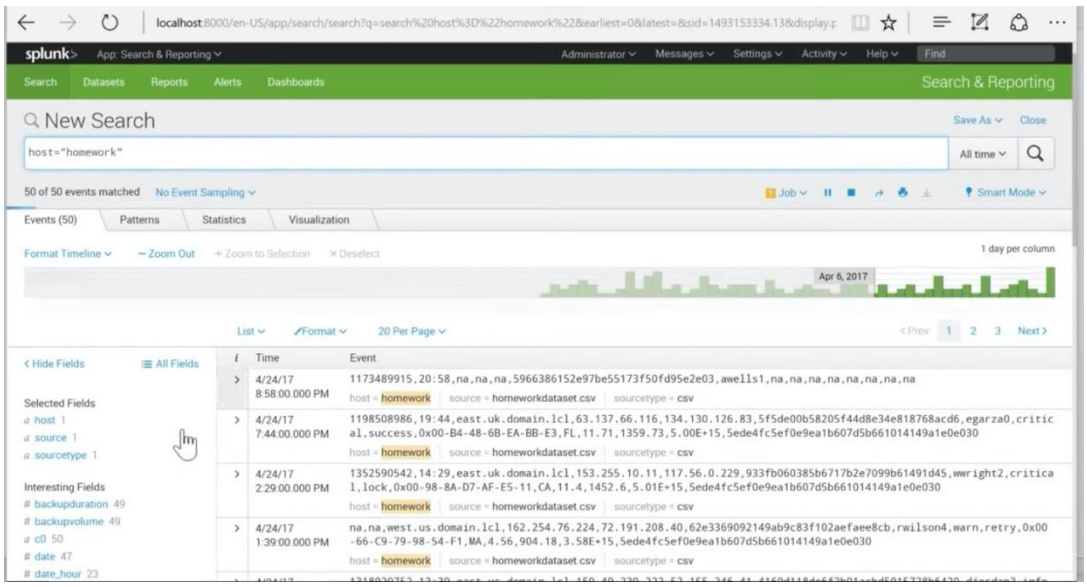

localhost:8000/en-US/app/launcher/home
splunk>
Administrator
Messages
Settings
Activity
Help
Find
Apps
Search & Reporting
Splunk 6.x Dashboard Examples
Explore Splunk Enterprise
Product Tours
New to Splunk? Take a tour to help you on your way.
Add Data
Add or forward data to Splunk Enterprise. Afterwards, you may extract fields.
Splunk Apps
Apps and add-ons extend the capabilities of Splunk Enterprise
Splunk Docs
Comprehensive documentation for Splunk Enterprise and for all other Splunk products.
Close
http://localhost:8000/en-US/manager/search/adddata

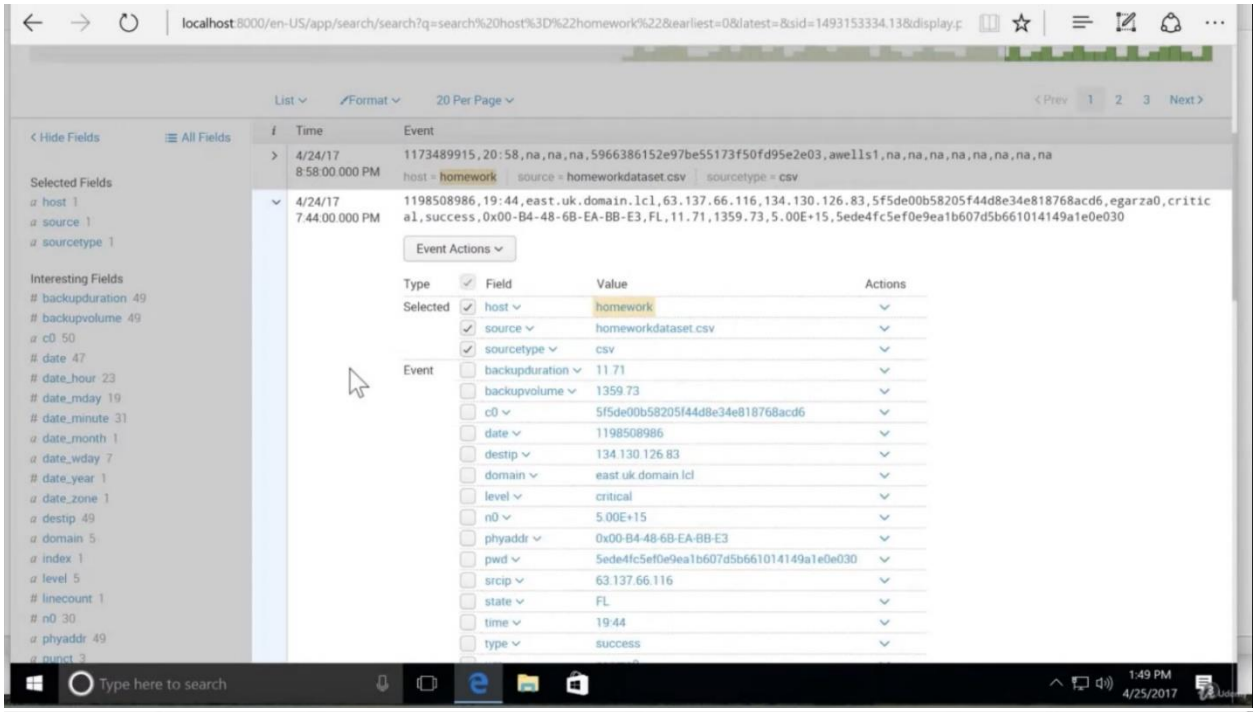

localhost:8000/en-US/manager/search/adddata/selectsource?input_mode=0
splunk>
Apps
Administrator
Messages
Settings
Activity
Help
Find
Add Data
Select Source
Set Source Type
Input Settings
Review
Done
Next >
Select Source
Choose a file to upload to Splunk, either by browsing your computer or by dropping a file into the target box below. Learn More
Selected File: No file selected
Select File
Drop your data file here
The maximum file upload size is 500 Mb
FAQ
What kinds of files can Splunk index?
What is a source?
Type here to search
1:46 PM
4/25/2017

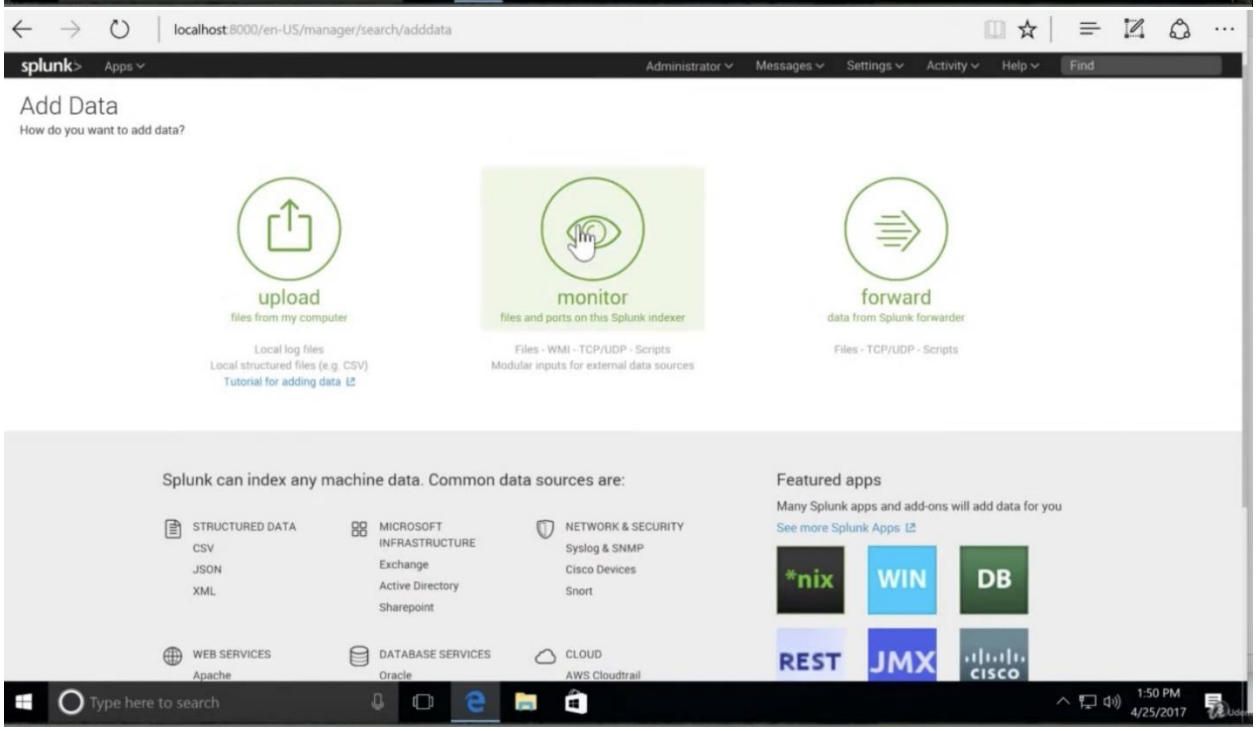

localhost:8000/en-US/manager/search/adddata/datapreview
splunk> Apps
Administrator Messages Settings Activity Help Find
Add Data
Select Source Set Source Type Input Settings Review Done
Next >
Set Source Type
This page lets you see how Splunk sees your data before indexing. If the events look correct and have the right timestamps, click "Next" to proceed. If not, use the options below to define proper event breaks and timestamps. If you cannot find an appropriate source type for your data, create a new one by clicking "Save As".
Source: homeworkdataset.csv
View Event Summary
Source type: csv
Save As
Timestamp
Delimited settings
Advanced
Table Format 20 Per Page
Prev 1 2 3 4 5 6 7 8 9 Next >
_time backupduration backupvolume c0 date destip
1 4/24/17 7:44:00.000 PM 11.71 1359.73 5f5de00b58205f44d8e34e818768acd6 1198508986
2 4/24/17 8:58:00.000 PM na na 5966386152e97be55173f50fd95e2e03 1173489915 na
3 4/24/17 2:29:00.000 PM 11.4 1452.6 933fb060385b6717b2e7099b61491d45 1352590542
4 4/24/17 1:39:00.000 PM 4.78 940.79 4169d118de6f2b01acbd5015728b6420 1318920752
5 4/24/17 1:39:00.000 PM 4.56 904.18 62e3369092149ab9c83f102aefaee8cb na
6 4/24/17 4:14:00.000 AM 10.58 565.53 33e2da96d10fcafe22c67d70b6cfc14a 1162253863
Type here to search
1:47 PM 4/25/2017

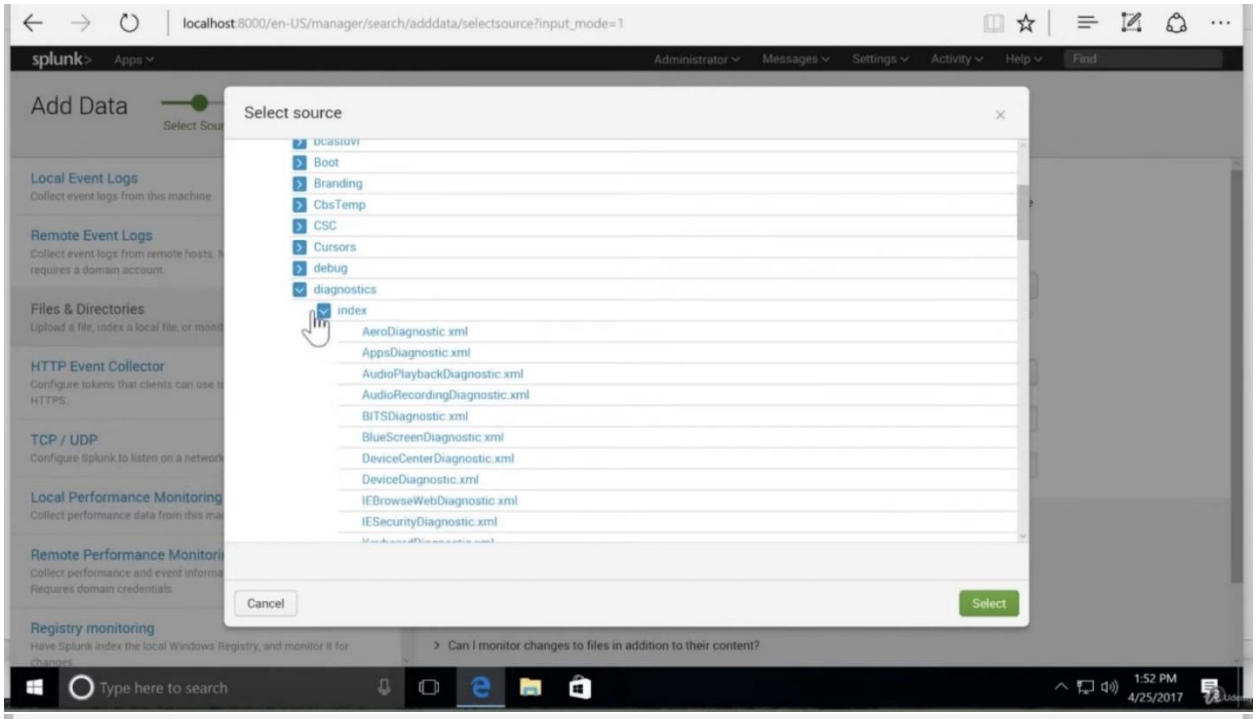

localhost:8000/en-US/manager/search/adddata/inputsettings
splunk> Apps
Administrator Messages Settings Activity Help Find
Add Data
Select Source Set Source Type Input Settings Review Done
Review >
Optionally set additional input parameters for this data input as follows:
Host
When Splunk indexes data, each event receives a "host" value. The host value should be the name of the machine from which the event originates. The type of input you choose determines the available configuration options. Learn More
Constant value Regular expression on path Segment in path
Host field value homework
Index
Splunk stores incoming data as events in the selected index. Consider using a "sandbox" index as a destination if you have problems determining a source type for your data. A sandbox index lets you troubleshoot your configuration without impacting production indexes. You can always change this setting later. Learn More
Index Default Create a new index
FAQ
How do indexes work?
How do I know when to create or use multiple indexes?
Type here to search
1:47 PM 4/25/2017

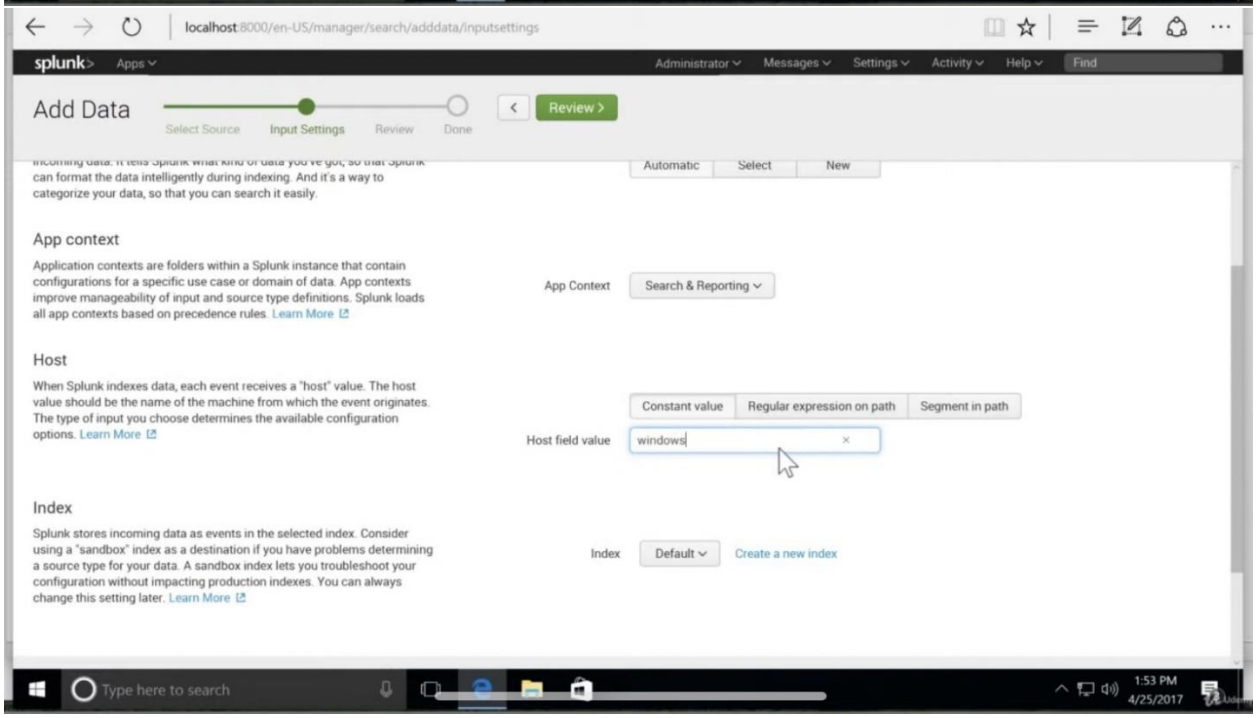

localhost:8000/en-US/manager/search/adddata/review
splunk> Apps
Administrator Messages Settings Activity Help Find
Add Data
Select Source
Set Source Type
Input Settings
Review
Done
Submit >
Review
Input Type Uploaded File
File Name homeworkdataset.csv
Source Type csv
Host homework
Index Default
Type here to search
1:48 PM
4/25/2017

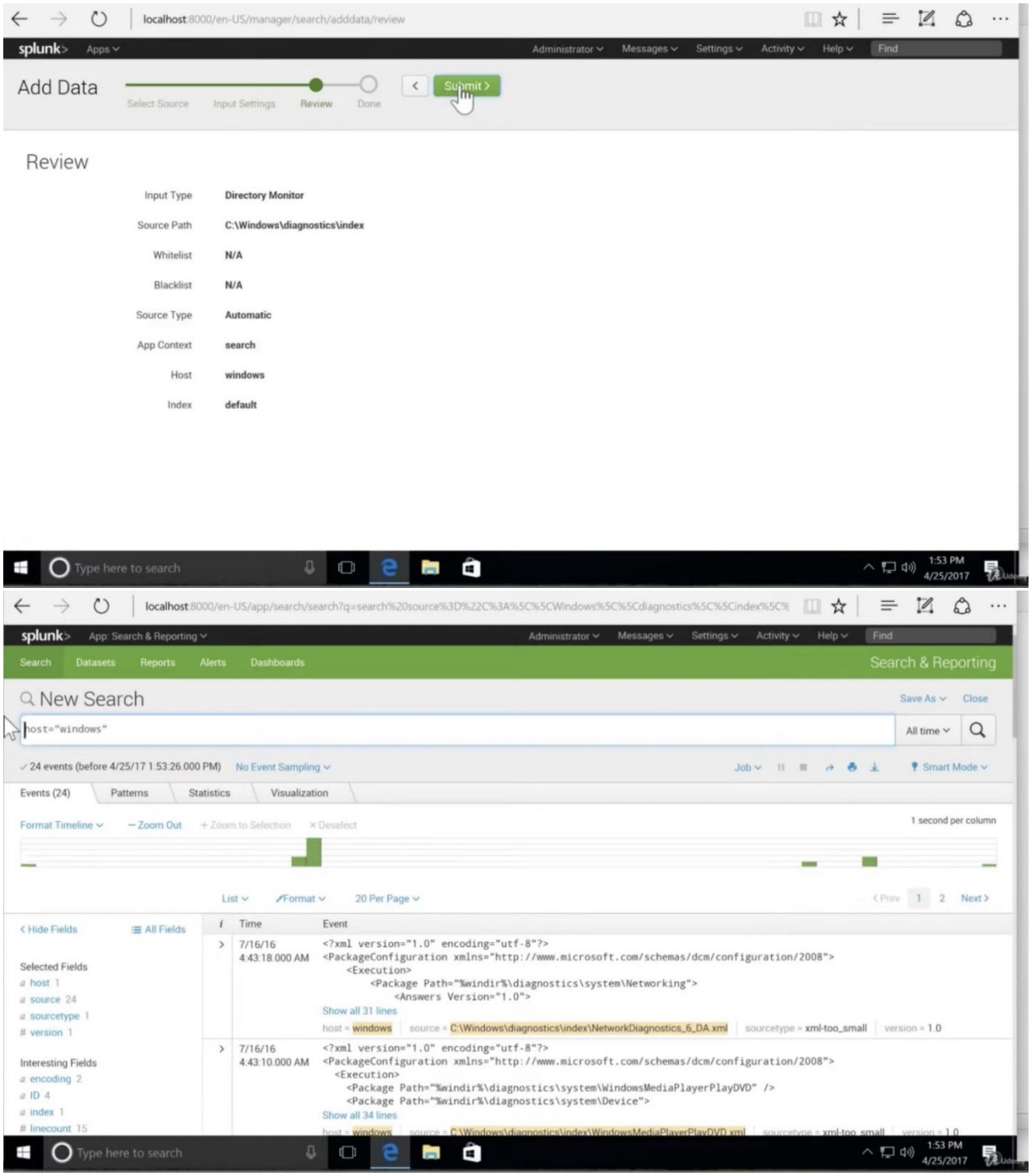

splunk> App: Search & Reporting
Search Datasets Reports Alerts Dashboards
Search & Reporting
New Search
host="homework"
50 of 50 events matched
Events (50) Patterns Statistics Visualization
Apr 6, 2017
Hide Fields All Fields
Selected Fields
host 1
source 1
sourcetype 1
Interesting Fields
backupduration 49
backupvolume 49
c0 50
date 47
date_hour 23
Time Event
4/24/17 8:58:00.000 PM
4/24/17 7:44:00.000 PM
4/24/17 2:29:00.000 PM
4/24/17 1:39:00.000 PM


localhost:8000/en-US/app/search/search?q=search%20host%3D%22homework%22&earliest=0&latest=&sid=1493153334.13&display.p
List
Format
20 Per Page
Prev 1 2 3 Next
Hide Fields
All Fields
Time
Event
4/24/17 8:58:00.000 PM
1173489915,20:58,na,na,na,5966386152e97be55173f50fd95e2e03,awells1,na,na,na,na,na,na,na,na
host = homework
source = homeworkdataset.csv
sourcetype = csv
4/24/17 7:44:00.000 PM
1198508986,19:44,east.uk.domain.lcl,63.137.66.116,134.130.126.83,5f5de00b58205f44d8e34e818768acd6,egarza0,critical,success,0x00-B4-48-6B-EA-BB-E3,FL,11.71,1359.73,5.00E+15,5ede4fc5ef0e9ea1b607d5b661014149a1e0e030
Event Actions
Type
Field
Value
Actions
Selected
host
homework
source
homeworkdataset.csv
sourcetype
csv
Event
backupduration
11.71
backupvolume
1359.73
c0
5f5de00b58205f44d8e34e818768acd6
date
1198508986
destip
134.130.126.83
domain
east.uk.domain.lcl
level
critical
n0
5.00E+15
phyaddr
0x00-B4-48-6B-EA-BB-E3
pwd
5ede4fc5ef0e9ea1b607d5b661014149a1e0e030
srcip
63.137.66.116
state
FL
time
19:44
type
success
Selected Fields
host 1
source 1
sourcetype 1
Interesting Fields
backupduration 49
backupvolume 49
c0 50
date 47
date_hour 23
date_mday 19
date_minute 31
date_month 1
date_wday 7
date_year 1
date_zone 1
destip 49
domain 5
index 1
level 5
linecount 1
n0 30
phyaddr 49
Type here to search
1:49 PM
4/25/2017


localhost:8000/en-US/manager/search/adddata
splunk>
Apps
Administrator
Messages
Settings
Activity
Help
Find
Add Data
How do you want to add data?
upload
files from my computer
Local log files
Local structured files (e.g. CSV)
Tutorial for adding data
monitor
files and ports on this Splunk indexer
Files - WMI - TCP/UDP - Scripts
Modular inputs for external data sources
forward
data from Splunk forwarder
Files - TCP/UDP - Scripts
Splunk can index any machine data. Common data sources are:
STRUCTURED DATA
CSV
JSON
XML
MICROSOFT INFRASTRUCTURE
Exchange
Active Directory
Sharepoint
NETWORK & SECURITY
Syslog & SNMP
Cisco Devices
Snort
WEB SERVICES
Apache
DATABASE SERVICES
Oracle
CLOUD
AWS Cloudtrail
Featured apps
Many Splunk apps and add-ons will add data for you
See more Splunk Apps
*nix
WIN
DB
REST
JMX
cisco
Type here to search
1:50 PM
4/25/2017


localhost:8000/en-US/manager/search/adddata/selectsource?input_mode=1
splunk> Apps
Administrator Messages Settings Activity Help Find
Add Data
Select Source
Local Event Logs
Collect event logs from this machine
Remote Event Logs
Collect event logs from remote hosts
requires a domain account
Files & Directories
Upload a file, index a local file, or
HTTP Event Collector
Configure tokens that clients can use
HTTPS.
TCP / UDP
Configure Splunk to listen on a network
Local Performance Monitoring
Collect performance data from this
Remote Performance Monitori
Collect performance and event informa
Requires domain credentials
Registry monitoring
Have Splunk index the local Windows Registry, and monitor it for
changes
Select source
Boot
Branding
CbsTemp
CSC
Cursors
debug
diagnostics
index
AeroDiagnostic.xml
AppsDiagnostic.xml
AudioPlaybackDiagnostic.xml
AudioRecordingDiagnostic.xml
BITSDiagnostic.xml
BlueScreenDiagnostic.xml
DeviceCenterDiagnostic.xml
DeviceDiagnostic.xml
IEBrowseWebDiagnostic.xml
IESecurityDiagnostic.xml
Cancel
Select
> Can I monitor changes to files in addition to their content?
Type here to search
1:52 PM
4/25/2017


localhost:8000/en-US/manager/search/adddata/inputsettings
splunk> Apps
Administrator Messages Settings Activity Help Find
Add Data
Select Source
Input Settings
Review
Done
Review >
can format the data intelligently during indexing. And it's a way to
categorize your data, so that you can search it easily.
Automatic Select New
App context
Application contexts are folders within a Splunk instance that contain
configurations for a specific use case or domain of data. App contexts
improve manageability of input and source type definitions. Splunk loads
all app contexts based on precedence rules. Learn More
App Context
Search & Reporting
Host
When Splunk indexes data, each event receives a "host" value. The host
value should be the name of the machine from which the event originates.
The type of input you choose determines the available configuration
options. Learn More
Constant value
Regular expression on path
Segment in path
Host field value
windows
Index
Splunk stores incoming data as events in the selected index. Consider
using a "sandbox" index as a destination if you have problems determining
a source type for your data. A sandbox index lets you troubleshoot your
configuration without impacting production indexes. You can always
change this setting later. Learn More
Index
Default
Create a new index
Type here to search
1:53 PM
4/25/2017


localhost:8000/en-US/manager/search/adddata/review
splunk> Apps
Administrator Messages Settings Activity Help Find
Add Data
Select Source Input Settings Review Done
Submit >
Review
Input Type Directory Monitor
Source Path C:\Windows\diagnostics\index
Whitelist N/A
Blacklist N/A
Source Type Automatic
App Context search
Host windows
Index default
Type here to search
1:53 PM
4/25/2017
localhost:8000/en-US/app/search/search?q=search%20source%3D%22C%3A%5C%5CWindows%5C%5Cdiagnostics%5C%5Cindex%5C%
splunk> App: Search & Reporting
Search Datasets Reports Alerts Dashboards
Search & Reporting
New Search
Save As Close
host="windows"
All time
24 events (before 4/25/17 1:53:26.000 PM) No Event Sampling
Job Smart Mode
Events (24) Patterns Statistics Visualization
Format Timeline Zoom Out Zoom to Selection Deselect
1 second per column
List Format 20 Per Page
Prev 1 2 Next
Hide Fields All Fields
Selected Fields
host 1
source 24
sourcetype 1
version 1
Interesting Fields
encoding 2
ID 4
index 1
linecount 15
Time Event
7/16/16 4:43:18.000 AM
<?xml version="1.0" encoding="utf-8"?>
<PackageConfiguration xmlns="http://www.microsoft.com/schemas/dcm/configuration/2008">
<Execution>
<Package Path="%windir%\diagnostics\system\Networking">
<Answers Version="1.0">
Show all 31 lines
host = windows source = C:\Windows\diagnostics\index\NetworkDiagnostics_6_DA.xml sourcetype = xml-too_small version = 1.0
7/16/16 4:43:10.000 AM
<?xml version="1.0" encoding="utf-8"?>
<PackageConfiguration xmlns="http://www.microsoft.com/schemas/dcm/configuration/2008">
<Execution>
<Package Path="%windir%\diagnostics\system\WindowsMediaPlayerPlayDVD" />
<Package Path="%windir%\diagnostics\system\Device">
Show all 34 lines
Type here to search
1:53 PM
4/25/2017

Practical 3

Aim:- To install and work on universal forwarder in Splunk.

## Forwarding & Receiving >

Types of forwarders

- Universal Forwarder
  - Most popular
  - Installs easily in Windows, Linux, Unix (AIX and HP-UX), Solaris, and FreeBSD
  - Installed at the local machine, and can be configured using a deployment server
  - Default ports: **8089** for management and **9997** for indexing

## Forwarding & Receiving >

Types of forwarders

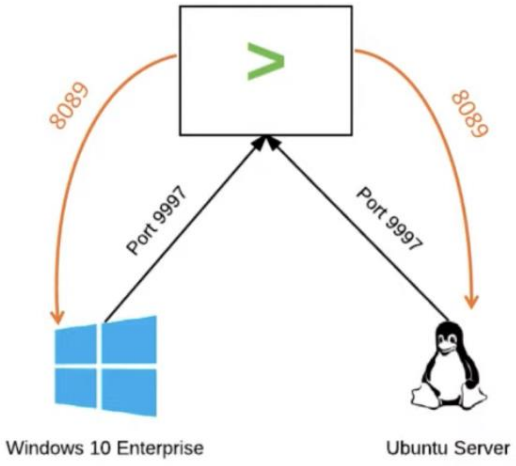
- Heavy Forwarder
  - A complete installation of the Splunk software, but with a forwarder license applied
  - Does much of the "heavy lifting" at the source
  - Can be configured at the source, and through a deployment server

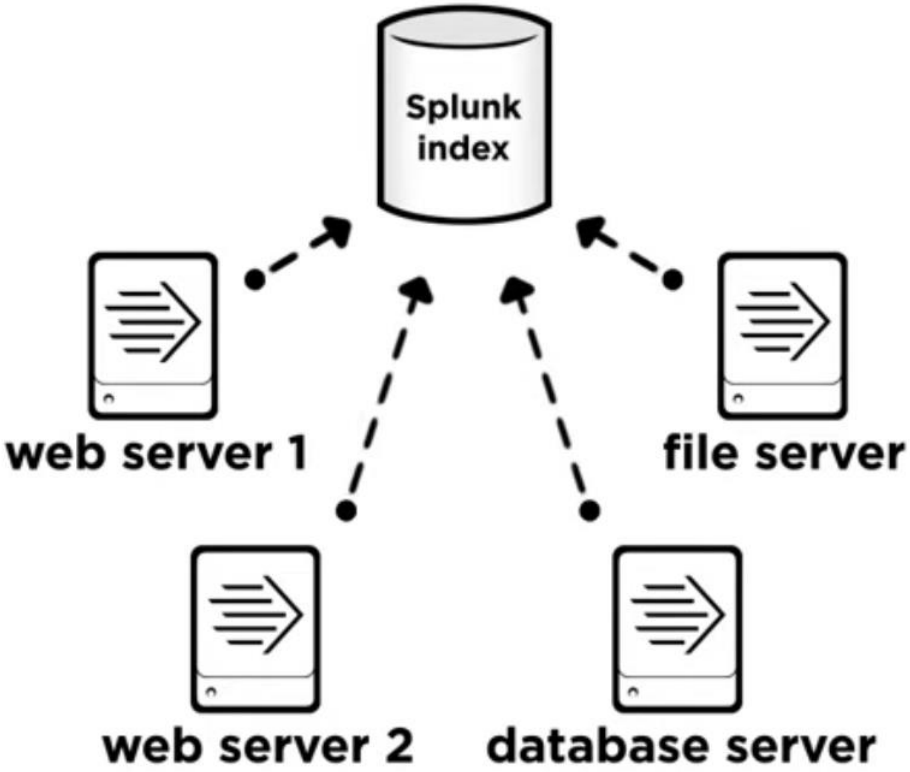
# Forwarding & Receiving >

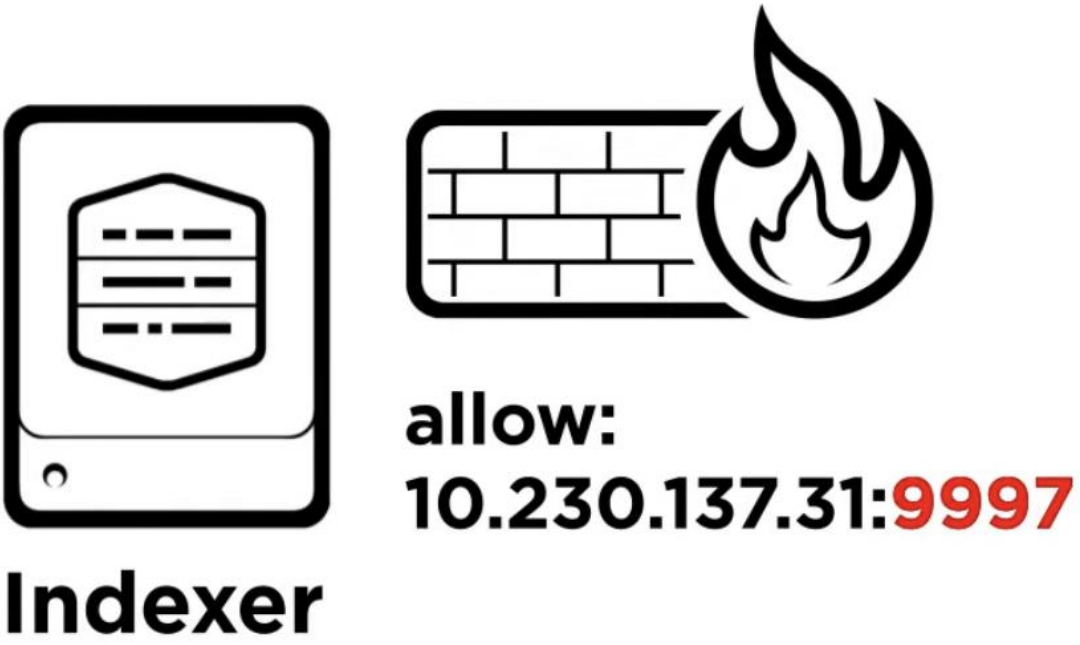
Configure receiving

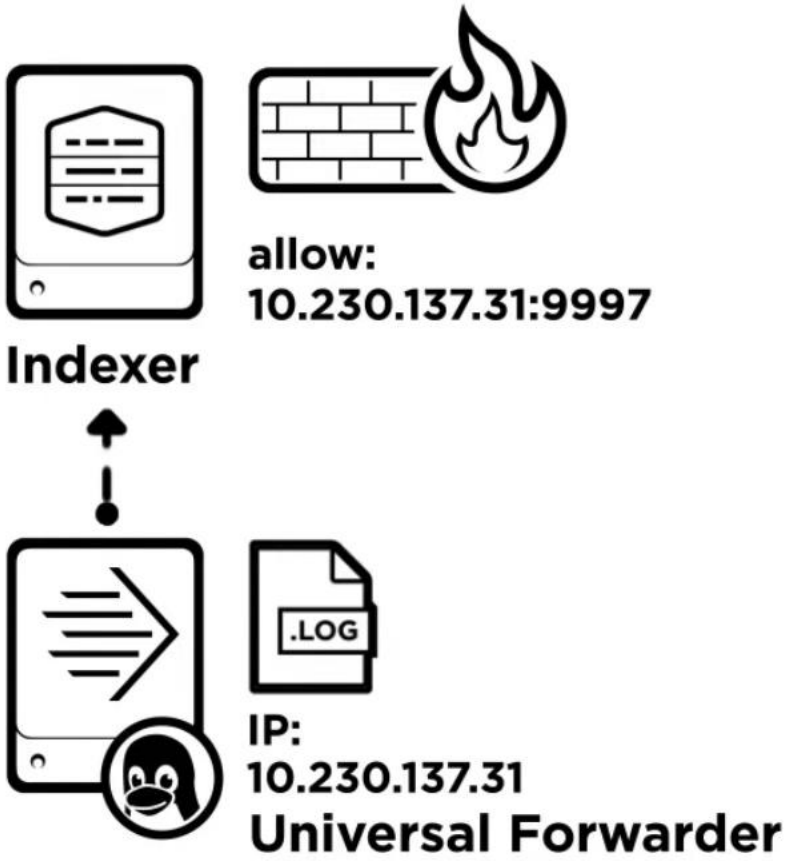
- Forwarding data to a Splunk indexer or search head won't work unless you configure that indexer or search head to receive data!
- Settings > Forwarding and Receiving > Add New

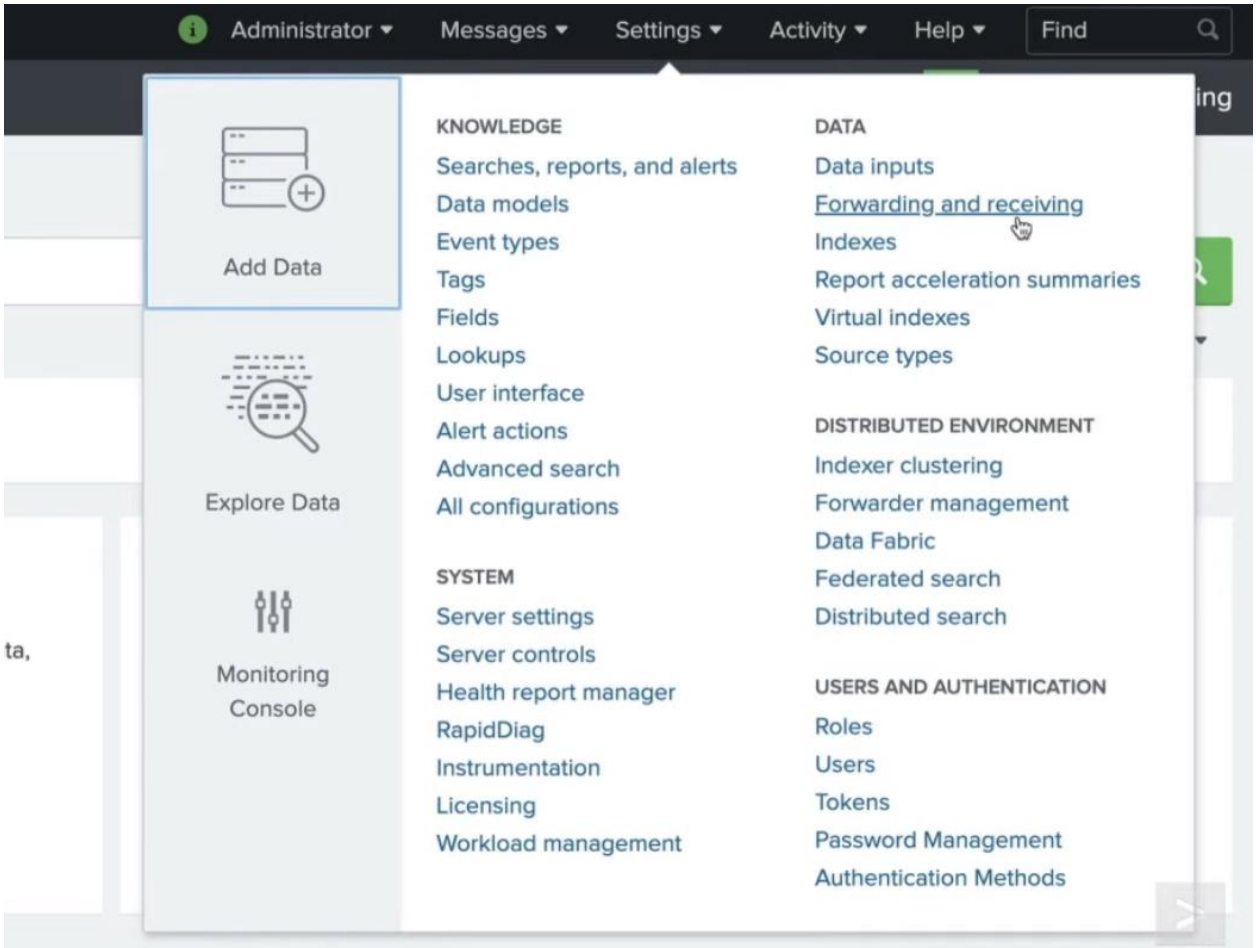
# Forwarding & Receiving >

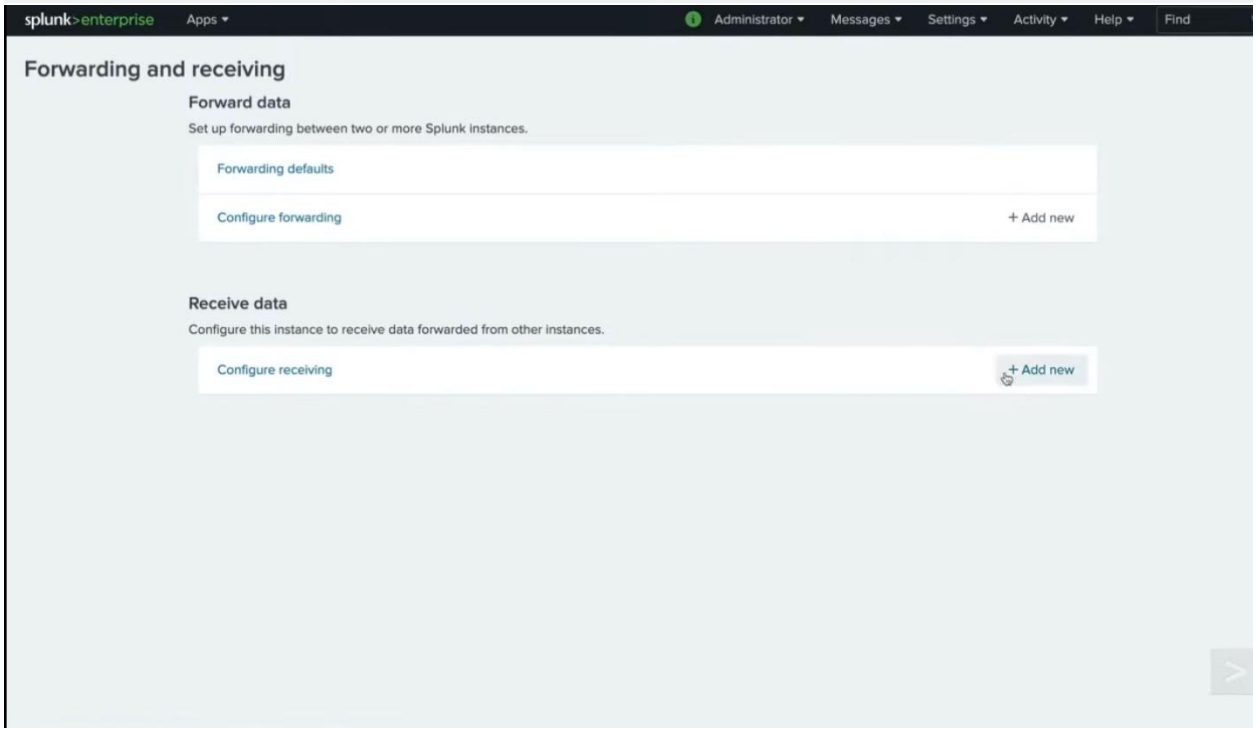


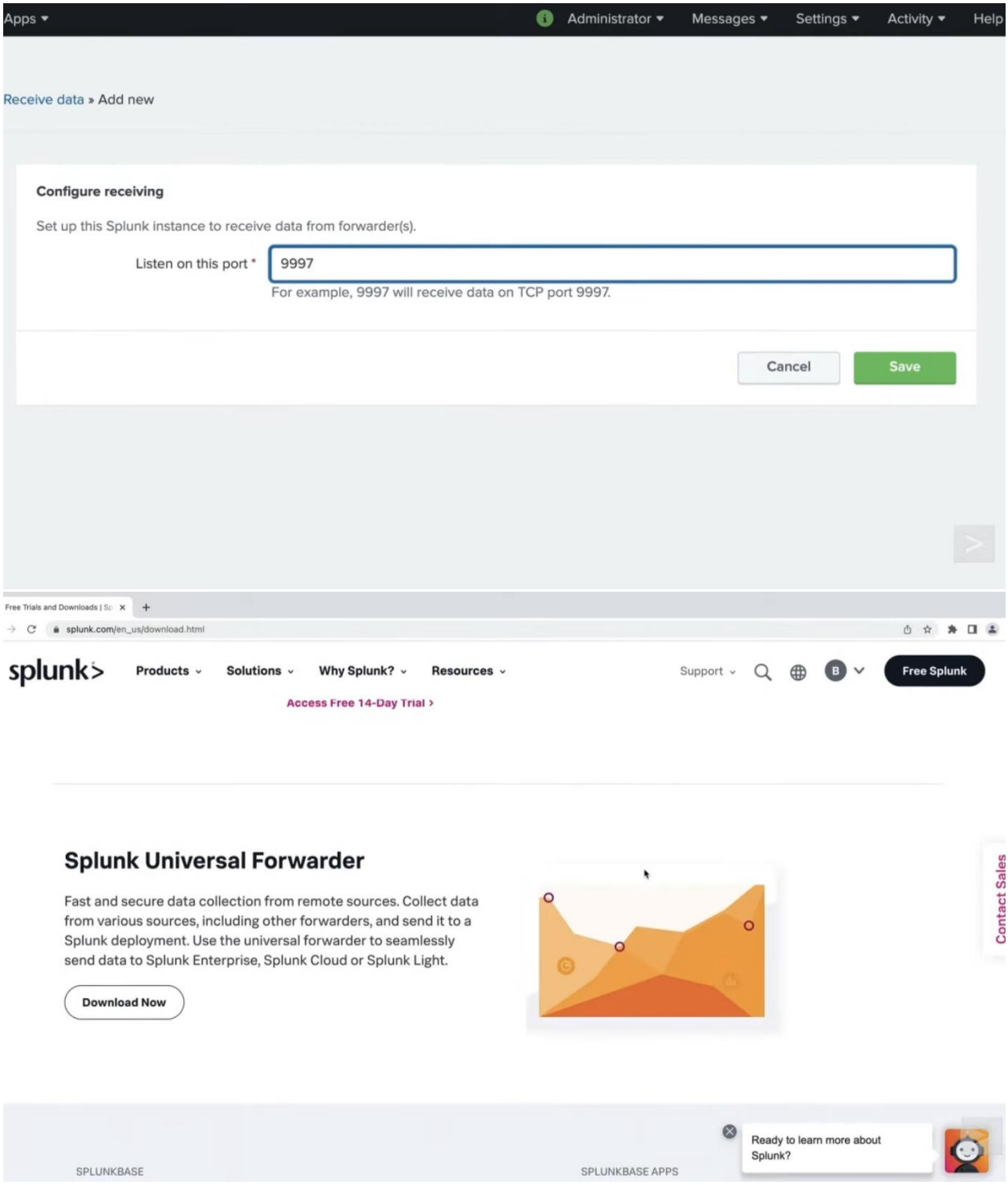

Splunk
index
web server 1
file server
web server 2
database server


allow:
10.230.137.31:9997
Indexer


allow:
10.230.137.31:9997
Indexer
.LOG
IP:
10.230.137.31
Universal Forwarder


Administrator
Messages
Settings
Activity
Help
Find
Add Data
Explore Data
Monitoring Console
KNOWLEDGE
Searches, reports, and alerts
Data models
Event types
Tags
Fields
Lookups
User interface
Alert actions
Advanced search
All configurations
SYSTEM
Server settings
Server controls
Health report manager
RapidDiag
Instrumentation
Licensing
Workload management
DATA
Data inputs
Forwarding and receiving
Indexes
Report acceleration summaries
Virtual indexes
Source types
DISTRIBUTED ENVIRONMENT
Indexer clustering
Forwarder management
Data Fabric
Federated search
Distributed search
USERS AND AUTHENTICATION
Roles
Users
Tokens
Password Management
Authentication Methods

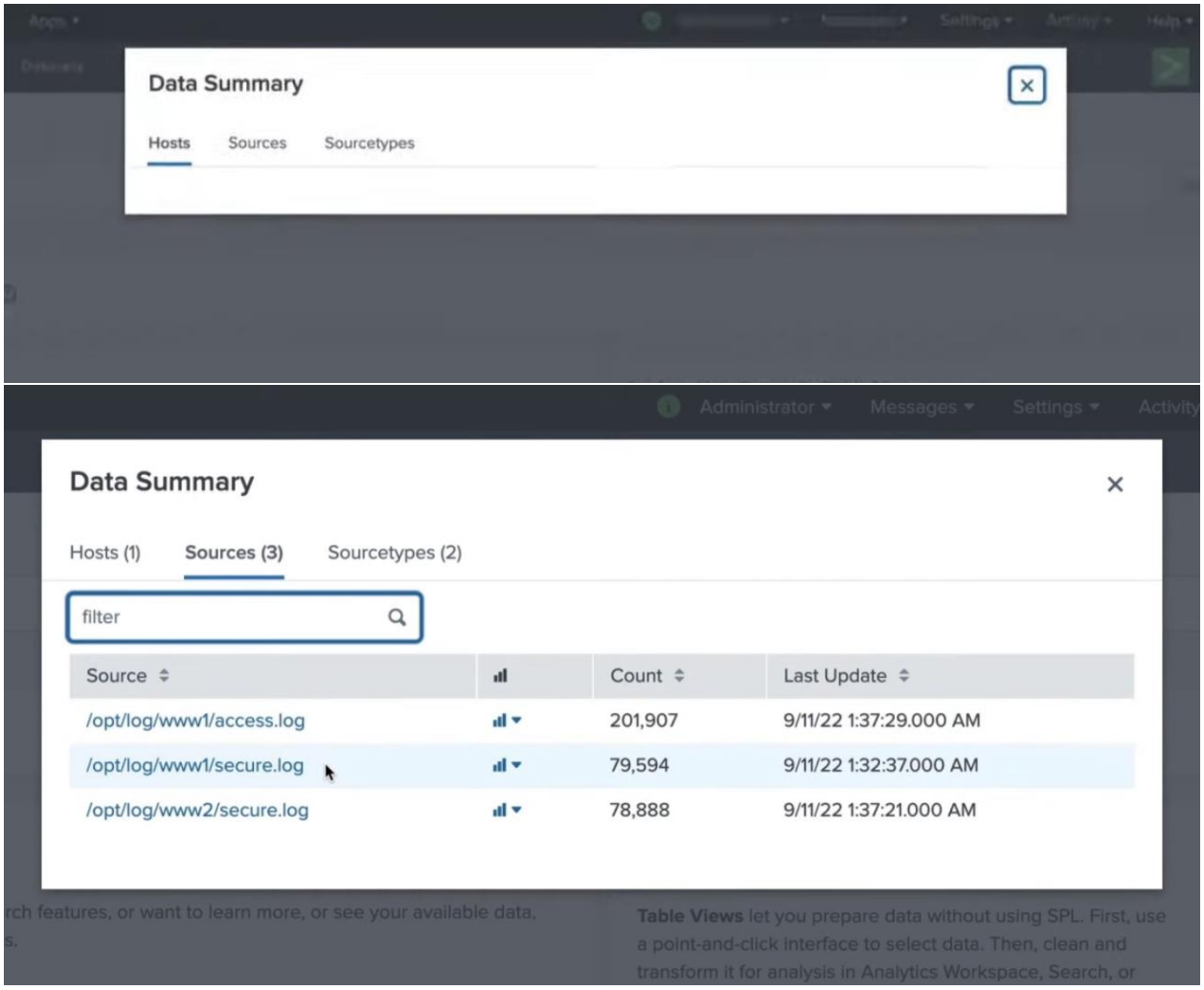

splunk>enterprise
Apps
Administrator
Messages
Settings
Activity
Help
Find
Forwarding and receiving
Forward data
Set up forwarding between two or more Splunk instances.
Forwarding defaults
Configure forwarding
+ Add new
Receive data
Configure this instance to receive data forwarded from other instances.
Configure receiving
+ Add new


Apps
Administrator
Messages
Settings
Activity
Help
Receive data » Add new
Configure receiving
Set up this Splunk instance to receive data from forwarder(s).
Listen on this port *
9997
For example, 9997 will receive data on TCP port 9997.
Cancel
Save
splunk.com/en_us/download.html
splunk>
Products
Solutions
Why Splunk?
Resources
Support
Free Splunk
Access Free 14-Day Trial ›
Splunk Universal Forwarder
Fast and secure data collection from remote sources. Collect data from various sources, including other forwarders, and send it to a Splunk deployment. Use the universal forwarder to seamlessly send data to Splunk Enterprise, Splunk Cloud or Splunk Light.
Download Now
Contact Sales
SPLUNKBASE
SPLUNKBASE APPS
Ready to learn more about Splunk?

```
[splunker@ip-10-0-0-100 opt]$ cd /opt/splunkforwarder/bin
[splunker@ip-10-0-0-100 bin]$ ./splunk start --accept-license

This appears to be your first time running this version of Splunk.

Splunk software must create an administrator account during startup. Otherwise, you cannot log in.
Create credentials for the administrator account.
Characters do not appear on the screen when you type in credentials.

Please enter an administrator username: admin
Password must contain at least:
   * 8 total printable ASCII character(s).
Please enter a new password:
Please confirm new password:
```

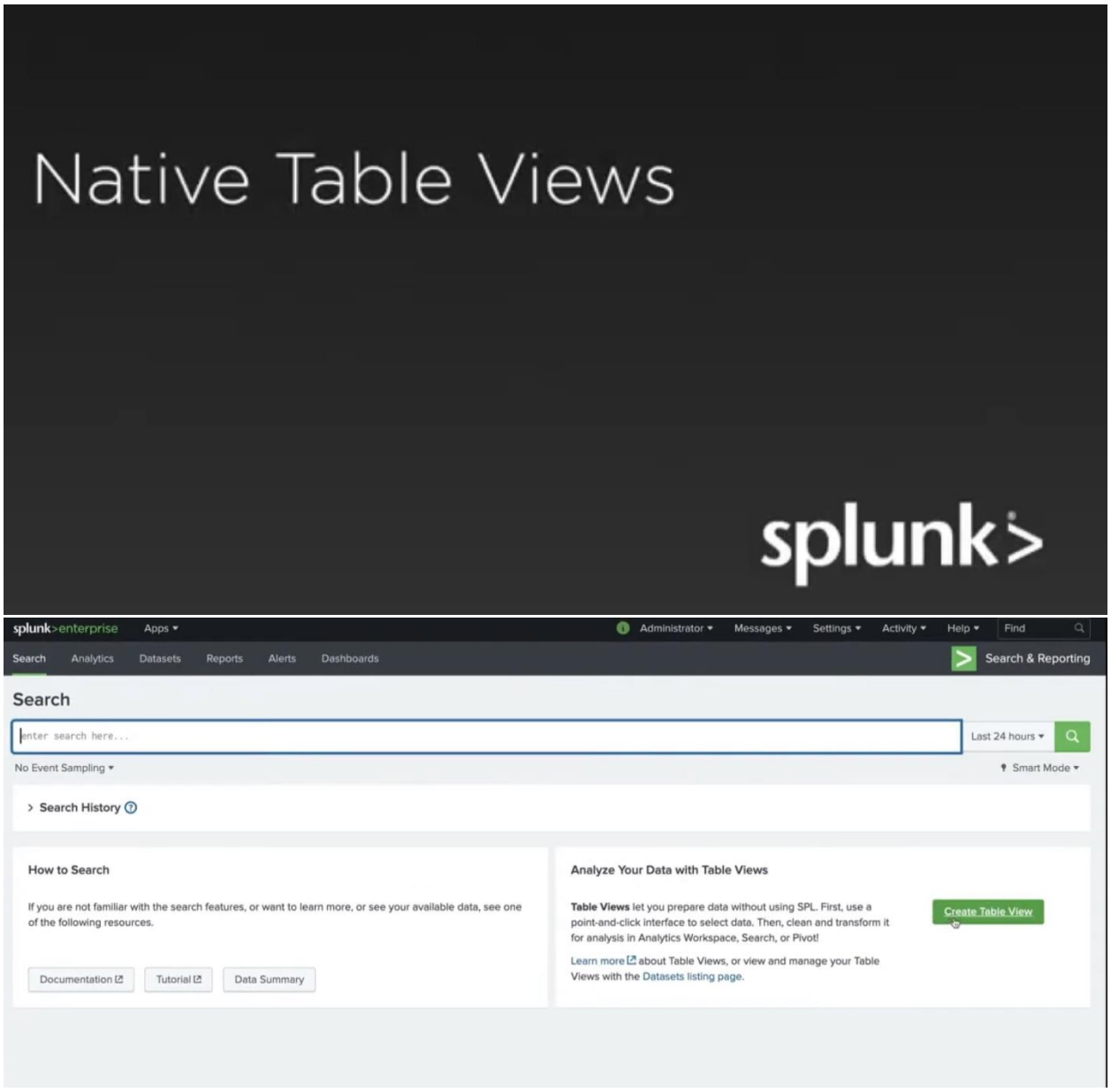
```
[splunker@ip-10-0-0-100 bin]$ sudo su
[root@ip-10-0-0-100 bin]# ./splunk enable boot-start -user splunker
Init script installed at /etc/init.d/splunk.
Init script is configured to run at boot.
[root@ip-10-0-0-100 bin]# exit
exit
[splunker@ip-10-0-0-100 bin]$
```

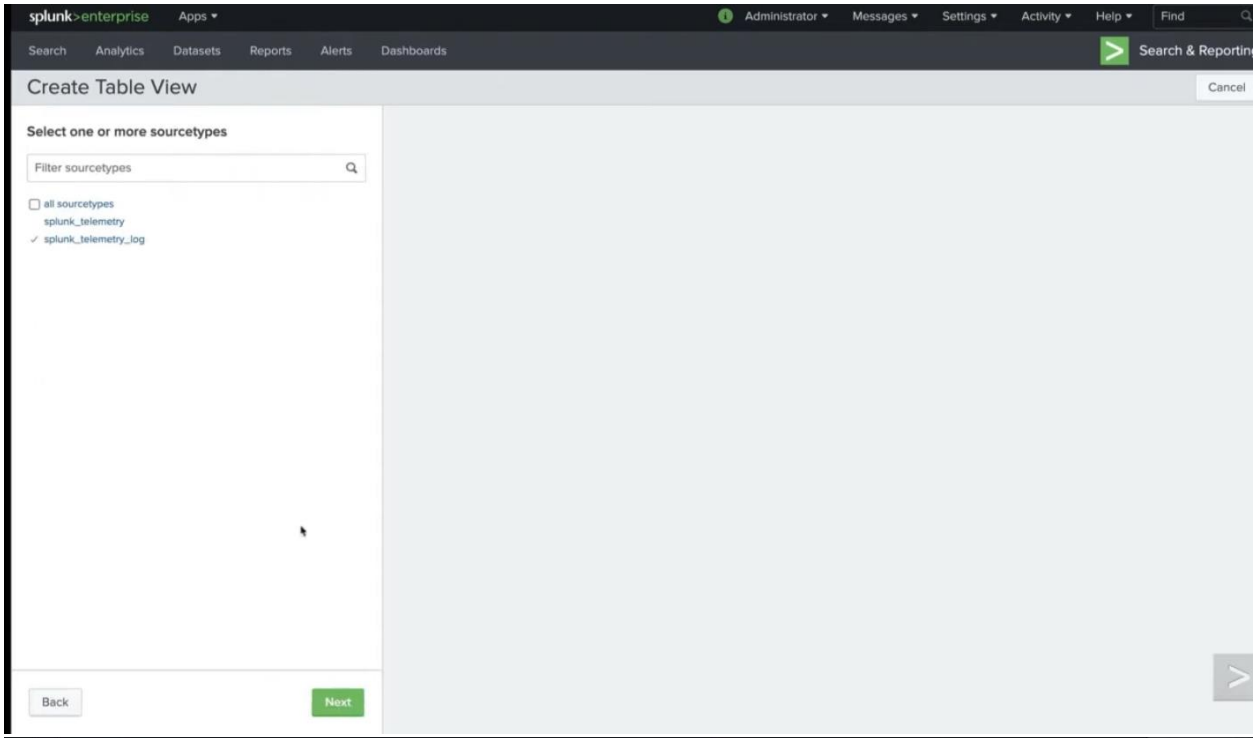
```
[splunker@ip-10-0-0-100 bin]$ ./splunk add forward-server 10.0.0.201:9997
Splunk username: admin
Password:
Added forwarding to: 10.0.0.201:9997.
[splunker@ip-10-0-0-100 bin]$ ./splunk add monitor -auth admin:goodPassword /opt/log/www1
Added monitor of '/opt/log/www1'.
[splunker@ip-10-0-0-100 bin]$
```

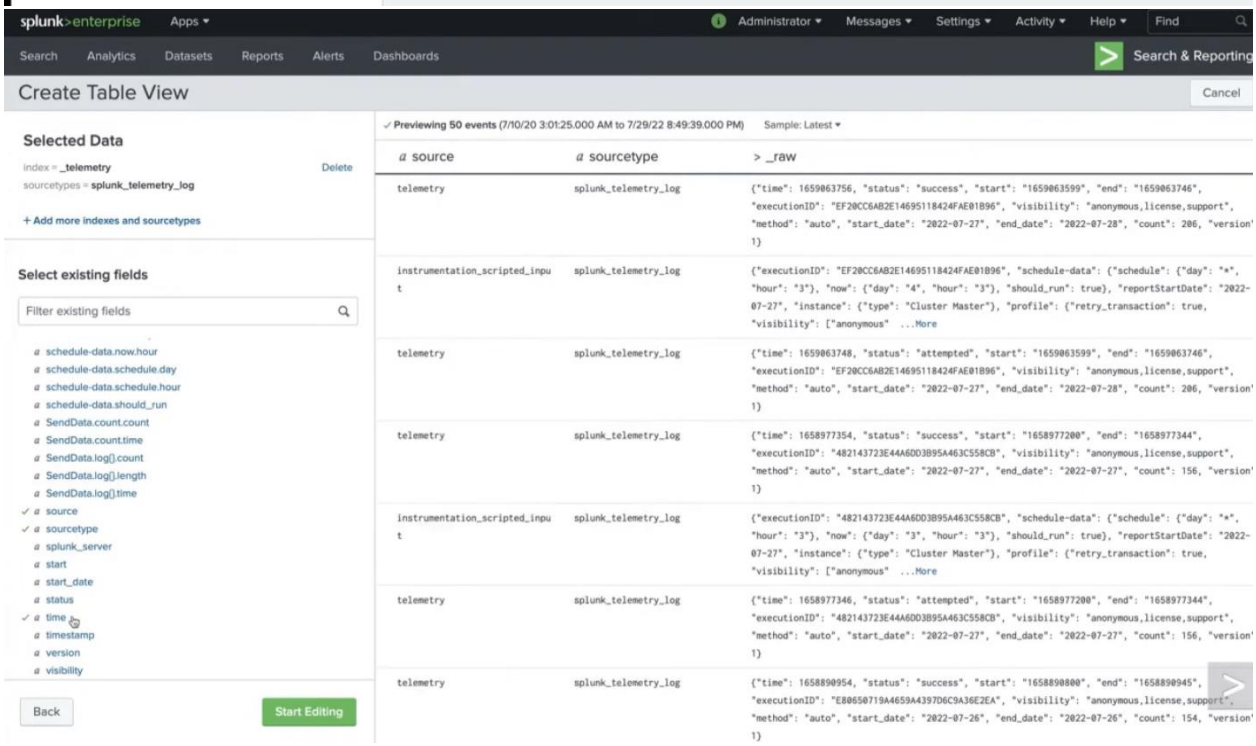

Data Summary
Hosts
Sources
Sourcetypes
Administrator
Messages
Settings
Activity
Data Summary
Hosts (1)
Sources (3)
Sourcetypes (2)
filter
Source
Count
Last Update
/opt/log/www1/access.log
201,907
9/11/22 1:37:29.000 AM
/opt/log/www1/secure.log
79,594
9/11/22 1:32:37.000 AM
/opt/log/www2/secure.log
78,888
9/11/22 1:37:21.000 AM
Table Views let you prepare data without using SPL. First, use a point-and-click interface to select data. Then, clean and transform it for analysis in Analytics Workspace, Search, or

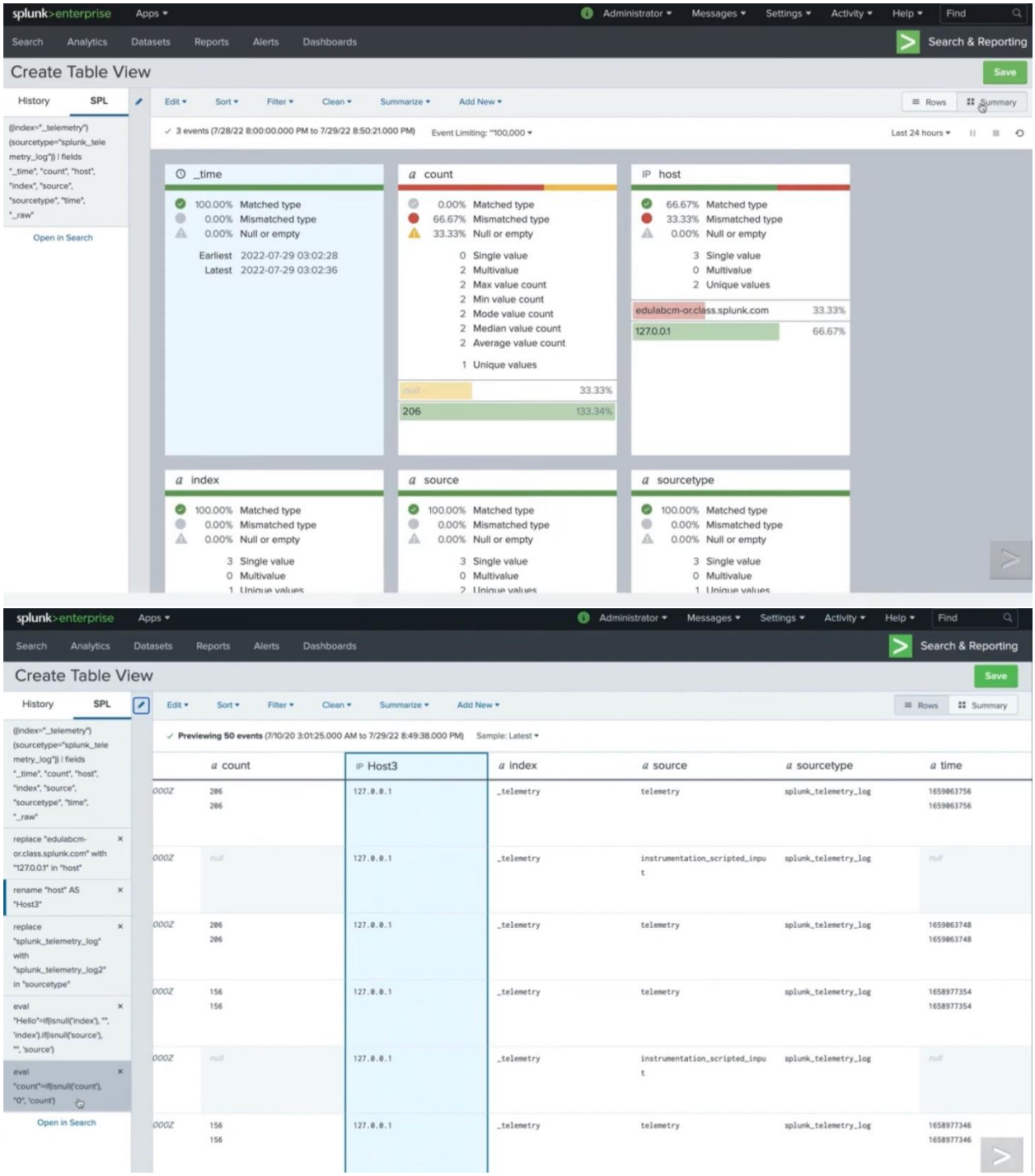
Practical 4

Aim:- To create and view native tables in Splunk.




splunk>enterprise
Apps
Administrator
Messages
Settings
Activity
Help
Find
Search
Analytics
Datasets
Reports
Alerts
Dashboards
Search & Reportin
Create Table View
Cancel
Select one or more sourcetypes
Filter sourcetypes
all sourcetypes
splunk_telemetry
splunk_telemetry_log
Back
Next


splunk>enterprise
Apps
Administrator
Messages
Settings
Activity
Help
Find
Search
Analytics
Datasets
Reports
Alerts
Dashboards
Search & Reporting
Create Table View
Cancel
Selected Data
index = _telemetry
sourcetypes = splunk_telemetry_log
Delete
+ Add more indexes and sourcetypes
Select existing fields
Filter existing fields
schedule-data.now.hour
schedule-data.schedule.day
schedule-data.schedule.hour
schedule-data.should_run
SendData.count.count
SendData.count.time
SendData.log{}.count
SendData.log{}.length
SendData.log{}.time
source
sourcetype
splunk_server
start
start_date
status
time
timestamp
version
visibility
Back
Start Editing
Previewing 50 events (7/10/20 3:01:25.000 AM to 7/29/22 8:49:39.000 PM)
Sample: Latest
source
sourcetype
_raw
telemetry
splunk_telemetry_log
{"time": 1659063756, "status": "success", "start": "1659063599", "end": "1659063746", "executionID": "EF20CC6AB2E14695118424FAE01B96", "visibility": "anonymous,license,support", "method": "auto", "start_date": "2022-07-27", "end_date": "2022-07-28", "count": 206, "version": 1}
instrumentation_scripted_input
splunk_telemetry_log
{"executionID": "EF20CC6AB2E14695118424FAE01B96", "schedule-data": {"schedule": {"day": "*", "hour": "3"}, "now": {"day": "4", "hour": "3"}, "should_run": true}, "reportStartDate": "2022-07-27", "instance": {"type": "Cluster Master"}, "profile": {"retry_transaction": true, "visibility": ["anonymous" ...More
telemetry
splunk_telemetry_log
{"time": 1659063748, "status": "attempted", "start": "1659063599", "end": "1659063746", "executionID": "EF20CC6AB2E14695118424FAE01B96", "visibility": "anonymous,license,support", "method": "auto", "start_date": "2022-07-27", "end_date": "2022-07-28", "count": 206, "version": 1}
telemetry
splunk_telemetry_log
{"time": 1658977354, "status": "success", "start": "1658977200", "end": "1658977344", "executionID": "482143723E44A6DD3B95A463C558CB", "visibility": "anonymous,license,support", "method": "auto", "start_date": "2022-07-27", "end_date": "2022-07-27", "count": 156, "version": 1}
instrumentation_scripted_input
splunk_telemetry_log
{"executionID": "482143723E44A6DD3B95A463C558CB", "schedule-data": {"schedule": {"day": "*", "hour": "3"}, "now": {"day": "3", "hour": "3"}, "should_run": true}, "reportStartDate": "2022-07-27", "instance": {"type": "Cluster Master"}, "profile": {"retry_transaction": true, "visibility": ["anonymous" ...More
telemetry
splunk_telemetry_log
{"time": 1658977346, "status": "attempted", "start": "1658977200", "end": "1658977344", "executionID": "482143723E44A6DD3B95A463C558CB", "visibility": "anonymous,license,support", "method": "auto", "start_date": "2022-07-27", "end_date": "2022-07-27", "count": 156, "version": 1}
telemetry
splunk_telemetry_log
{"time": 1658890954, "status": "success", "start": "1658890800", "end": "1658890945", "executionID": "E80650719A4659A4397D6C9A36E2EA", "visibility": "anonymous,license,support", "method": "auto", "start_date": "2022-07-26", "end_date": "2022-07-26", "count": 154, "version": 1}


splunk>enterprise
Apps
Administrator
Messages
Settings
Activity
Help
Find
Search
Analytics
Datasets
Reports
Alerts
Dashboards
Search & Reporting
Create Table View
Save
History
SPL
Edit
Sort
Filter
Clean
Summarize
Add New
Rows
Summary
((index="_telemetry") (sourcetype="splunk_telemetry_log")) | fields "_time", "count", "host", "index", "source", "sourcetype", "time", "_raw"
Open in Search
3 events (7/28/22 8:00:00.000 PM to 7/29/22 8:50:21.000 PM)
Event Limiting: 100,000
Last 24 hours
_time
100.00% Matched type
0.00% Mismatched type
0.00% Null or empty
Earliest 2022-07-29 03:02:28
Latest 2022-07-29 03:02:36
count
0.00% Matched type
66.67% Mismatched type
33.33% Null or empty
0 Single value
2 Multivalue
2 Max value count
2 Min value count
2 Mode value count
2 Median value count
2 Average value count
1 Unique values
null 33.33%
206 133.34%
host
66.67% Matched type
33.33% Mismatched type
0.00% Null or empty
3 Single value
0 Multivalue
2 Unique values
edulabcm-or.class.splunk.com 33.33%
127.0.0.1 66.67%
index
100.00% Matched type
0.00% Mismatched type
0.00% Null or empty
3 Single value
0 Multivalue
source
100.00% Matched type
0.00% Mismatched type
0.00% Null or empty
3 Single value
0 Multivalue
sourcetype
100.00% Matched type
0.00% Mismatched type
0.00% Null or empty
3 Single value
0 Multivalue

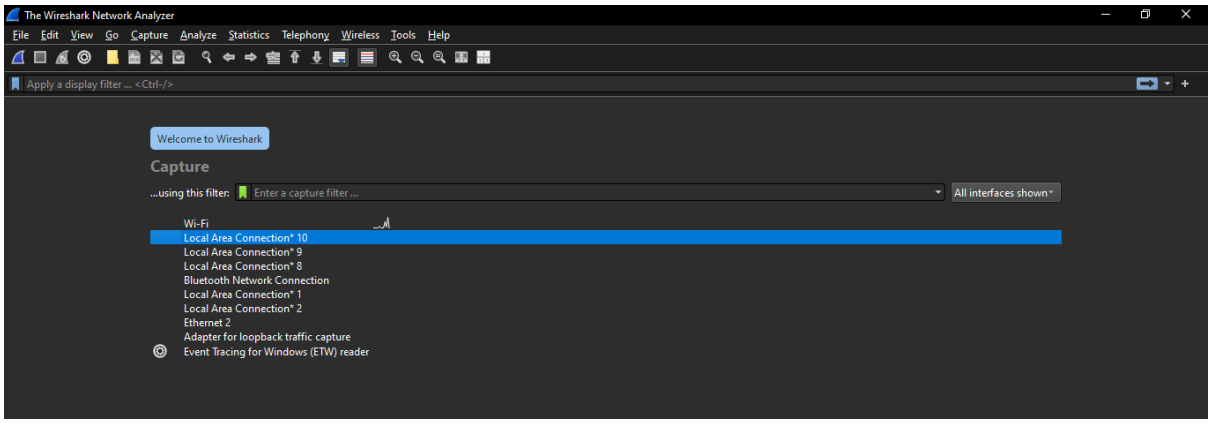

splunk>enterprise
Apps
Administrator
Messages
Settings
Activity
Help
Find
Search
Analytics
Datasets
Reports
Alerts
Dashboards
Search & Reporting
Create Table View
Save
History
SPL
Edit
Sort
Filter
Clean
Summarize
Add New
Rows
Summary
((index="_telemetry") (sourcetype="splunk_telemetry_log")) | fields "_time", "count", "host", "index", "source", "sourcetype", "time", "_raw"
replace "edulabcm-or.class.splunk.com" with "127.0.0.1" in "host"
rename "host" AS "Host3"
replace "splunk_telemetry_log" with "splunk_telemetry_log2" in "sourcetype"
eval "Hello"=if(isnull('index'), "", 'index').if(isnull('source'), "", 'source')
eval "count"=if(isnull('count'), "0", 'count')
Open in Search
Previewing 50 events (7/10/20 3:01:25.000 AM to 7/29/22 8:49:38.000 PM)
Sample: Latest
count
Host3
index
source
sourcetype
time
000Z 206 206 127.0.0.1 _telemetry telemetry splunk_telemetry_log 1659063756 1659063756
000Z null 127.0.0.1 _telemetry instrumentation_scripted_input splunk_telemetry_log null
000Z 206 206 127.0.0.1 _telemetry telemetry splunk_telemetry_log 1659063748 1659063748
000Z 156 156 127.0.0.1 _telemetry telemetry splunk_telemetry_log 1658977354 1658977354
000Z null 127.0.0.1 _telemetry instrumentation_scripted_input splunk_telemetry_log null
000Z 156 156 127.0.0.1 _telemetry telemetry splunk_telemetry_log 1658977346 1658977346

Practical 5

Aim:- To Collect network traffic logs with WireShark and Analyse for potential security incidents.

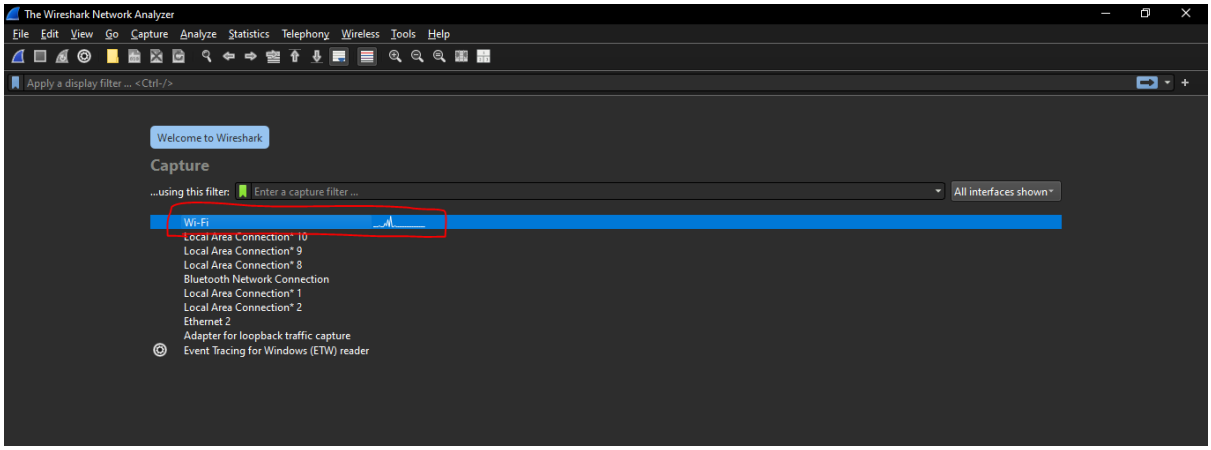
**Step 1: Open the Wireshark application**

First, open the Wireshark application by clicking on the icon.



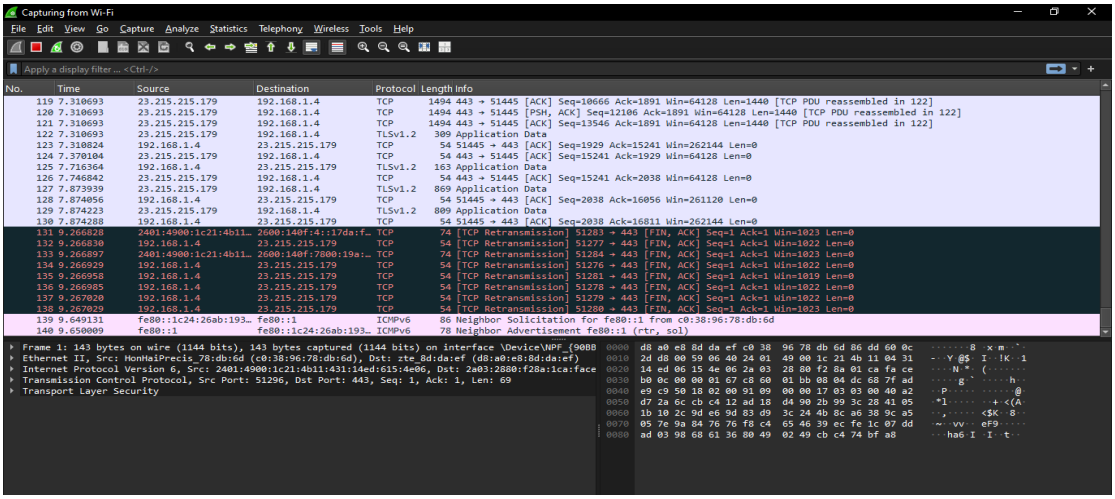

**Step 2: Select a network interface to capture traffic on**

From the welcome screen, select which of your network interfaces you want to capture traffic on. The line to the right of the interface indicates the network traffic flow passing through that interface.

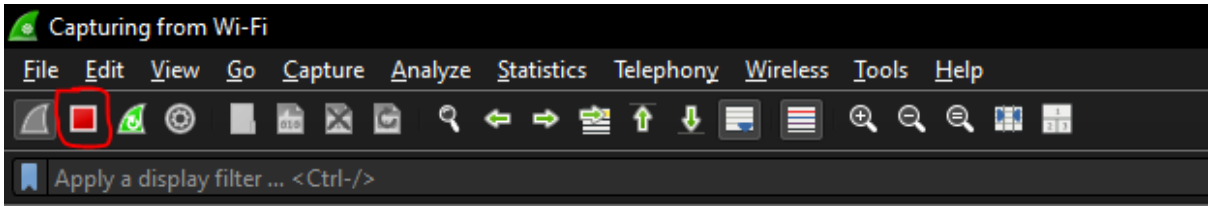



**Step 3: Stopping the packet capture**

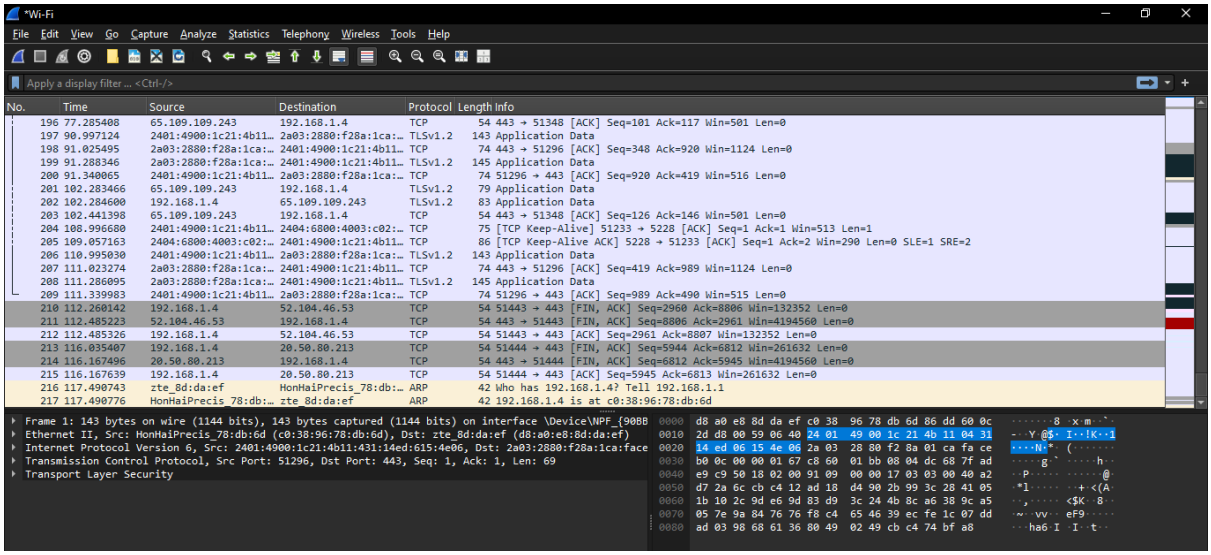
Double-clicking on the capture interface you want to capture traffic to start capturing packets.



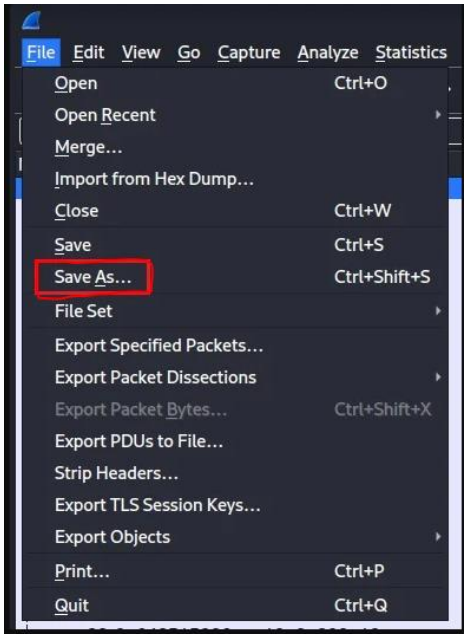

To stop capture packets, click on the red Stop Capture button.

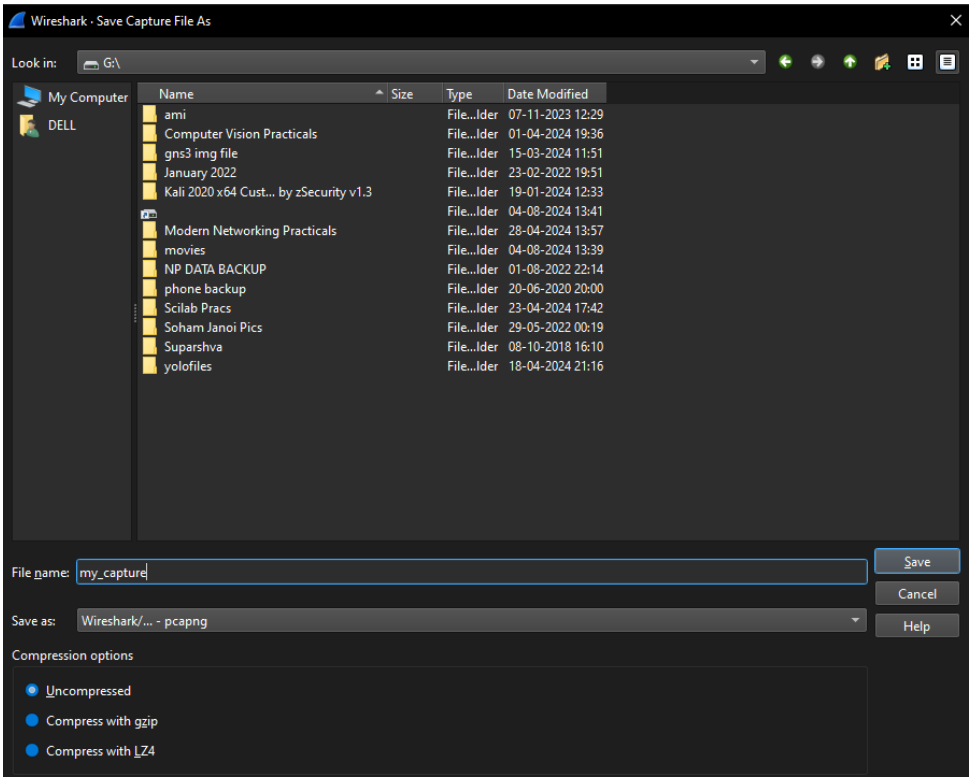



This will save the packet capture into a temporary file and allow you to perform your analysis.




To save your data into a specific file. Select File > Save As, then enter the location and name of the file under which you want to save this data.



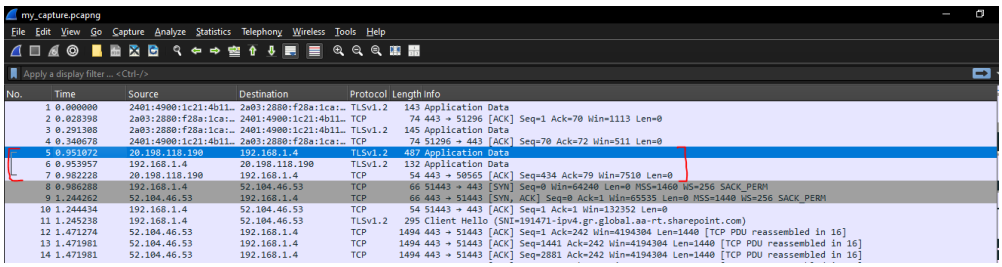




The network traffic logs will be collected.

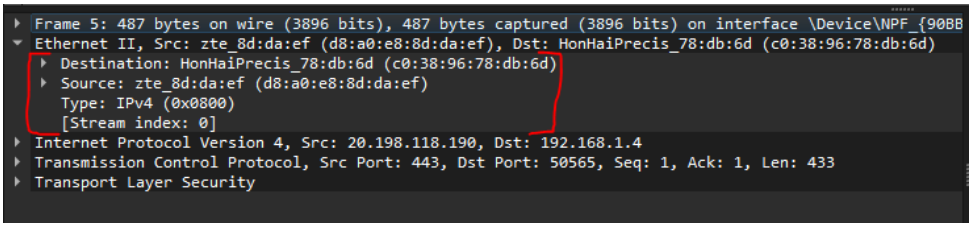
Practical 6

Aim:- To examine packets using WireShark.

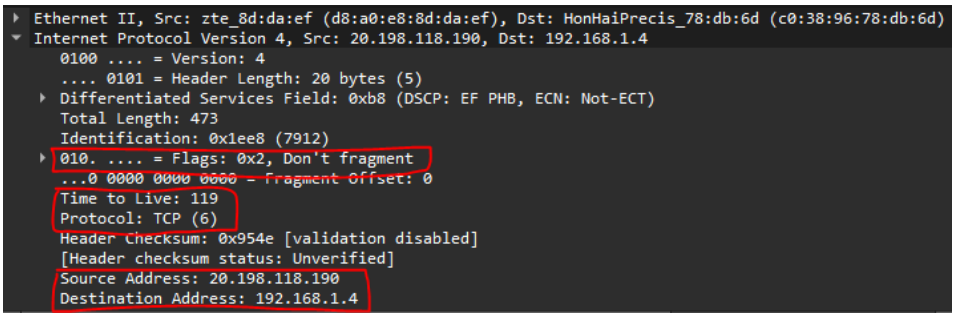
To examine a packet, select it within the packet list pane. This will populate the packet details pane with that packet's network information.



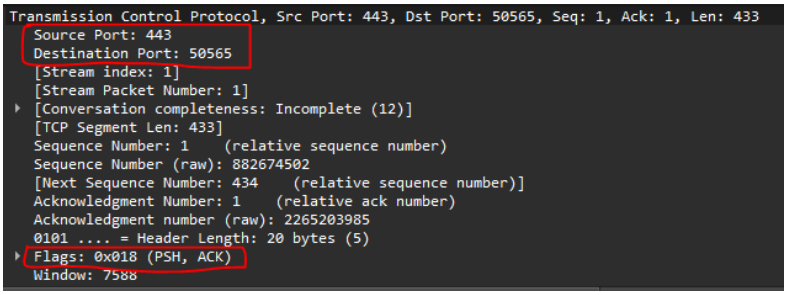

To start, you can analyze the network information at the **link layer**. This includes the source and destination MAC addresses of the two communicating devices and the type of Internet protocol used (IPv4 or IPv6).




You can then move on to the information included within the **Internet layer**. This example contains information specific to the IPv4 protocol, such as IP flags, the source and destination IP address, Time to Live (TTL), the transport protocol encapsulated within this packet (e.g., TCP), and other header information.



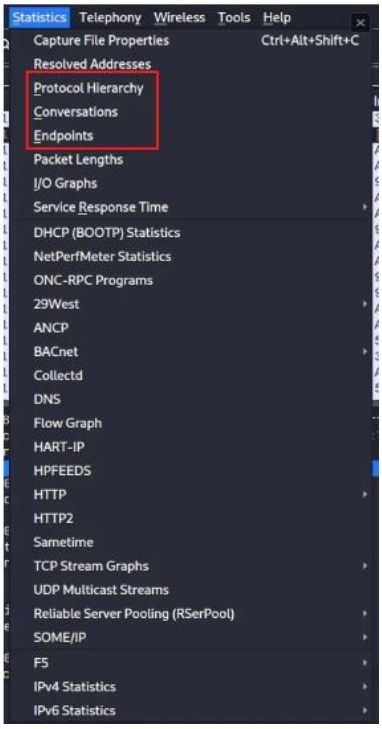

Following the IP layer is the **transport layer**. This network protocol is responsible for the end-to-end data delivery between hosts and will be either UDP or TCP. It includes the source and destination port of the segment, TCP-specific header flags, the size of the encapsulated application message, and other header information.



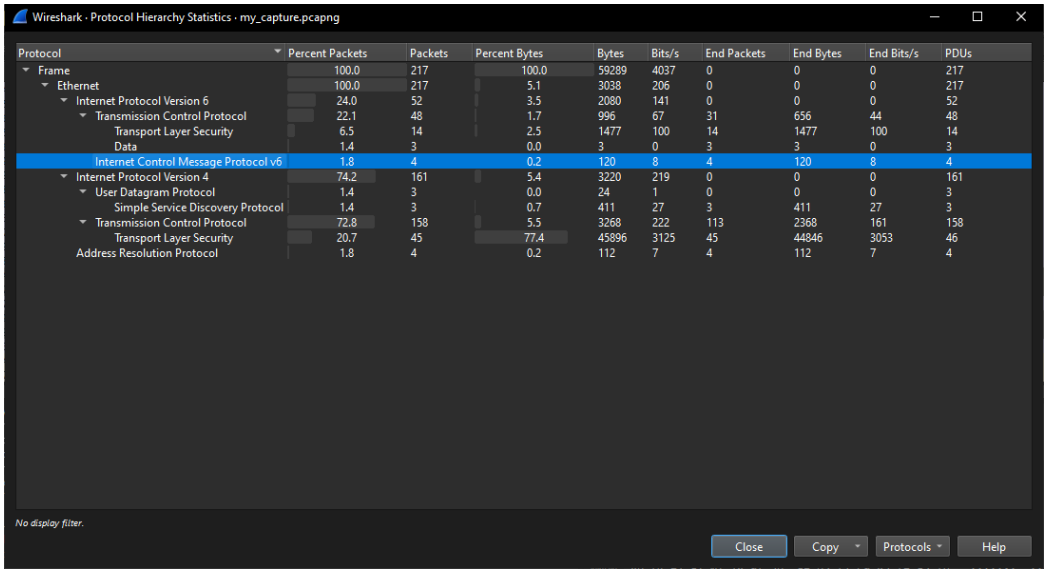

**Performing statistical analysis**

Aside from analyzing individual network packets, Wireshark also has a powerful statistical analysis feature that lets you quickly summarise your network traffic.

Select the Statistic menu from Wireshark's main menu bar. This will provide options for showing summary information about the protocols being used, the endpoints communicating, and the network conversations between those endpoints.

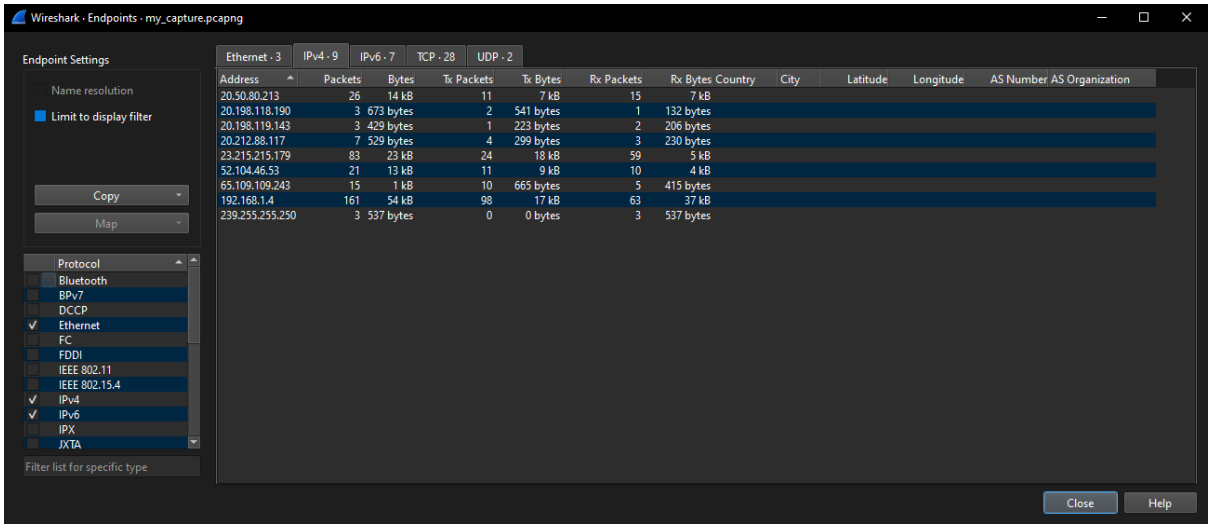



Select the Protocol Hierarchy option to discover the protocols used within your captured network traffic. This will generate a window summarizing the network protocols used at each TCP/IP stack layer.

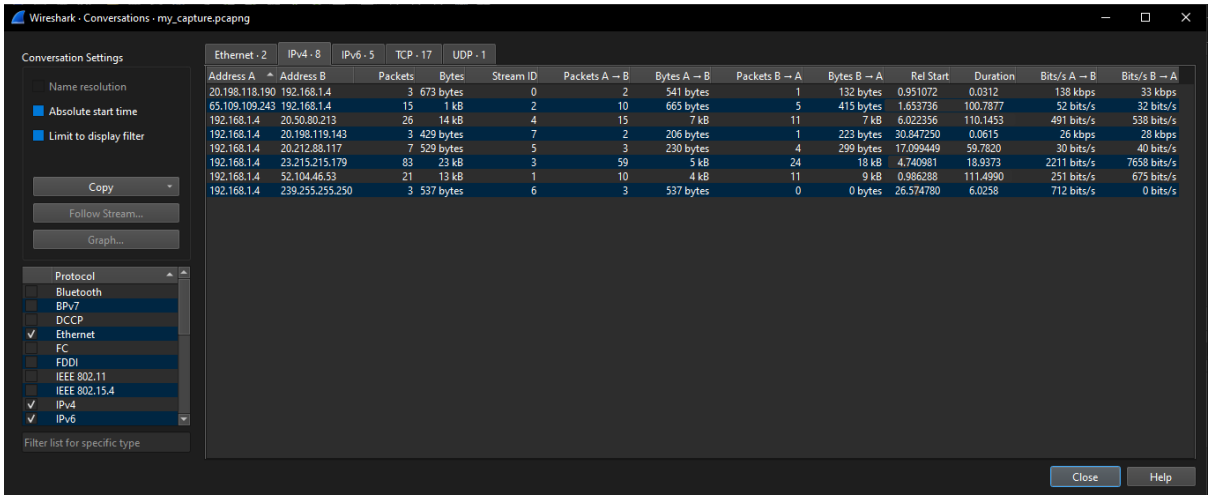



Select the Endpoints option to find out what endpoints are present in your network traffic. This will list all the unique endpoint devices communicating within the network packets you have captured based on Ethernet address, IP address, and TCP/UDP port.

In the example below, the IPv4 tab has been selected. This shows that nine devices are communicating in this packet capture, with four being the primary exchangers of data.




Select the Conversations option to delve into the network conversations within your packet capture. This will show you the network traffic traveling between endpoints.

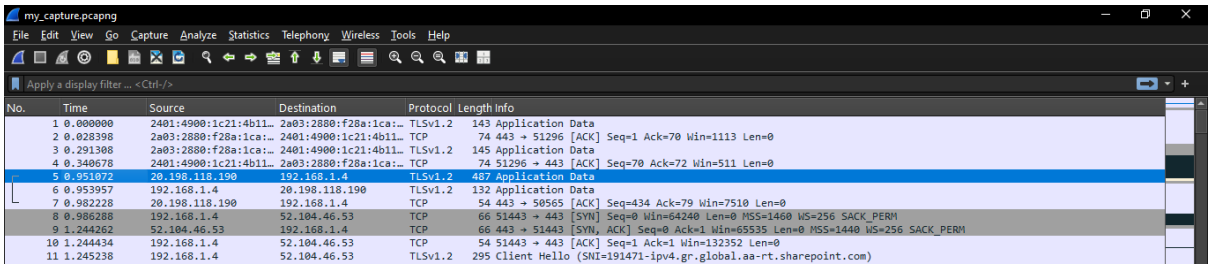

Wireshark · Conversations · my_capture.pcapng
Conversation Settings
Name resolution
Absolute start time
Limit to display filter
Copy
Follow Stream...
Graph...
Protocol
Bluetooth
BPv7
DCCP
Ethernet
FC
FDDI
IEEE 802.11
IEEE 802.15.4
IPv4
IPv6
Filter list for specific type
Ethernet · 2
IPv4 · 8
IPv6 · 5
TCP · 17
UDP · 1
Address A Address B Packets Bytes Stream ID Packets A → B Bytes A → B Packets B → A Bytes B → A Rel Start Duration Bits/s A → B Bits/s B → A
20.198.118.190 192.168.1.4 3 673 bytes 0 2 541 bytes 1 132 bytes 0.951072 0.0312 138 kbps 33 kbps
65.109.109.243 192.168.1.4 15 1 kB 2 10 665 bytes 5 415 bytes 1.653736 100.7877 52 bits/s 32 bits/s
192.168.1.4 20.50.80.213 26 14 kB 4 15 7 kB 11 7 kB 6.022356 110.1453 491 bits/s 538 bits/s
192.168.1.4 20.198.119.143 3 429 bytes 7 2 206 bytes 1 223 bytes 30.847250 0.0615 26 kbps 28 kbps
192.168.1.4 20.212.88.117 7 529 bytes 5 3 230 bytes 4 299 bytes 17.099449 59.7820 30 bits/s 40 bits/s
192.168.1.4 23.215.215.179 83 23 kB 3 59 5 kB 24 18 kB 4.740981 18.9373 2211 bits/s 7658 bits/s
192.168.1.4 52.104.46.53 21 13 kB 1 10 4 kB 11 9 kB 0.986288 111.4990 251 bits/s 675 bits/s
192.168.1.4 239.255.255.250 3 537 bytes 6 3 537 bytes 0 0 bytes 26.574780 6.0258 712 bits/s 0 bits/s
Close
Help

Practical 7

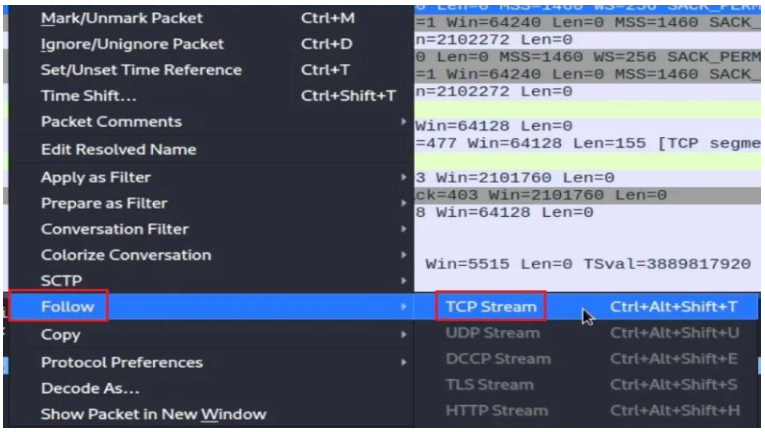
Aim:- To follow TCP streams using WireShark.

One of the most common use cases for Wireshark is following TCP streams. By following this stream, you can see the conversation between the two devices and the data they exchanged. This is useful for troubleshooting network issues or discovering hidden information.

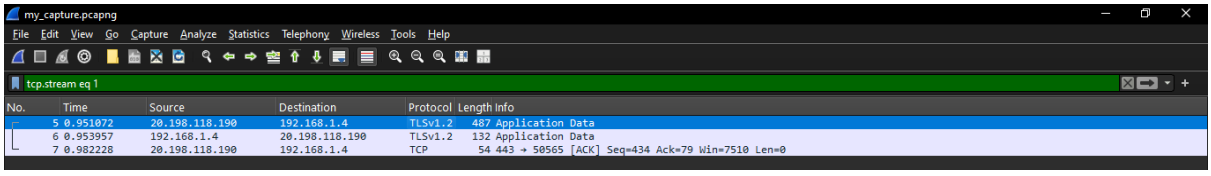
To follow a TCP stream in Wireshark, right-click on a packet whose TCP stream you want to follow.



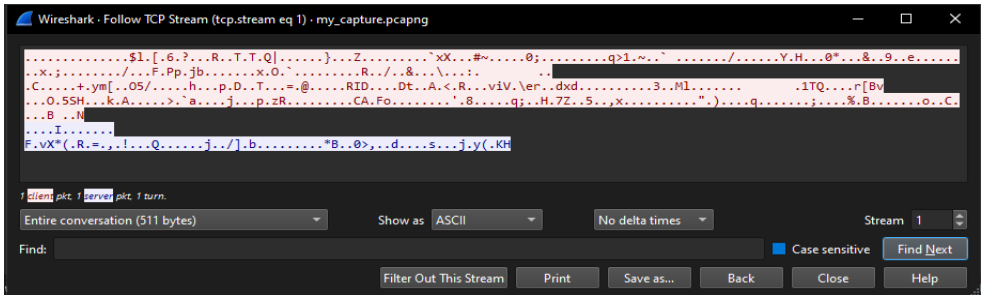

Here packet number 5 has been selected. A drop-down arrow on the left of the packet list pane indicates the entire network conversation down to packet number 7. To follow this TCP stream, right-click on packet number 5 and select Follow > TCP Stream.




Wireshark will automatically apply a display filter that only shows packets from this TCP stream.



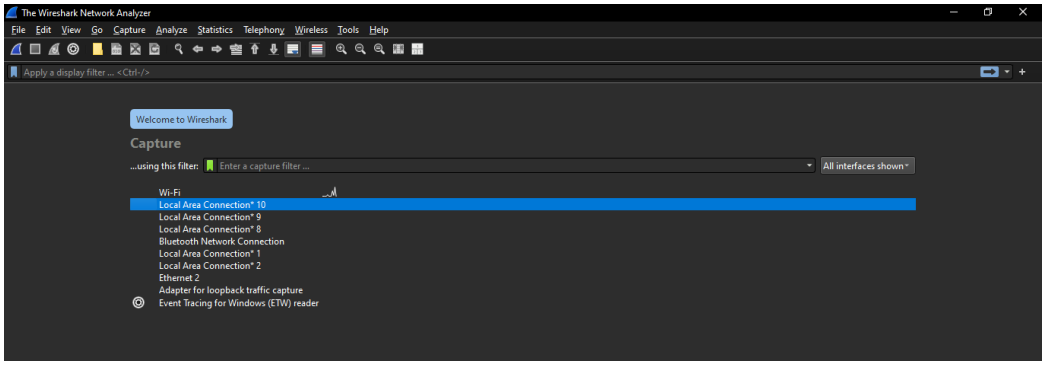

Wireshark will also generate a summary view of the TCP stream in a new window.

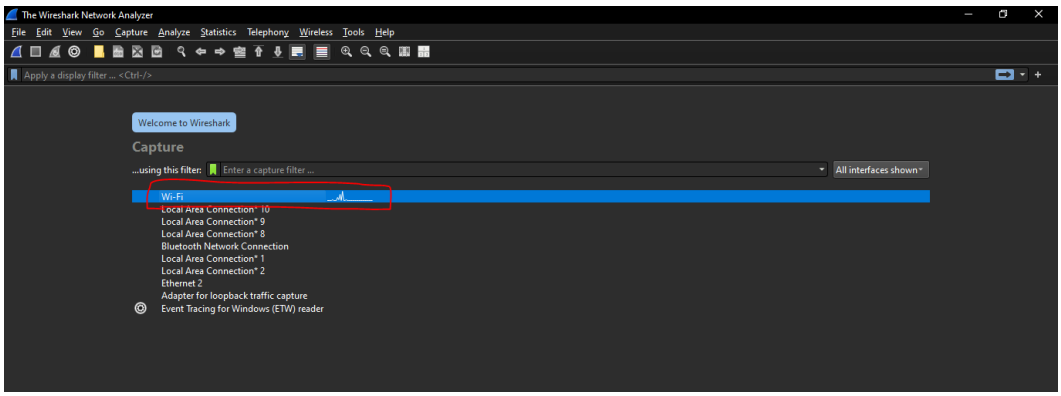


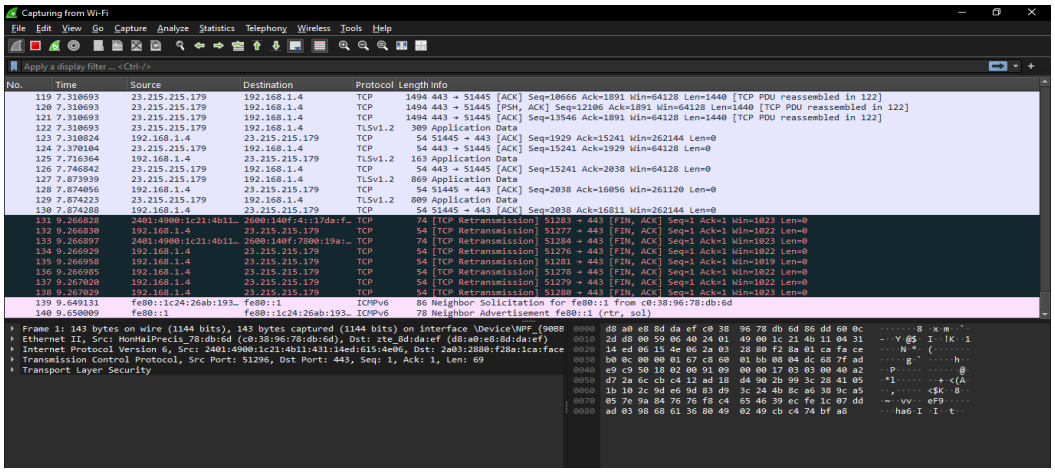

## Capture DNS Traffic

**To capture DNS traffic:**

1. Start a Wireshark capture.

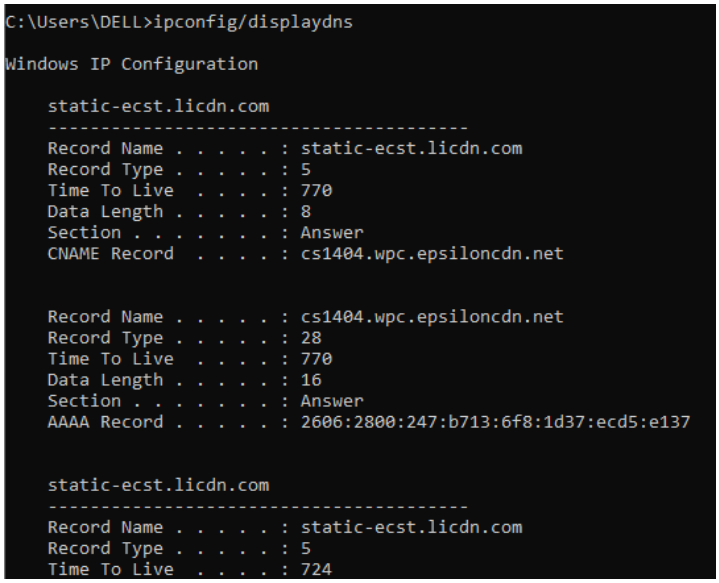



2. Open a command prompt.

3. Type **ipconfig /displaydns** and press **Enter** to display the DNS cache.
4. Observe the results. Notice the only records currently displayed come from the hosts file.



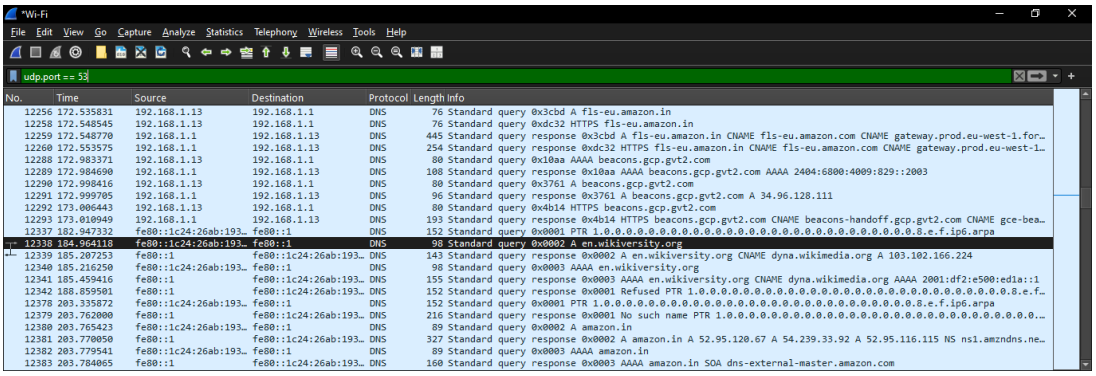

5. Type **nslookup en.wikiversity.org** and press **Enter**.


6. Observe the result. Notice there is an entry in the cache for *en.wikiversity.org*.
7. Close the command prompt.
8. Stop the Wireshark capture.

Practical 8

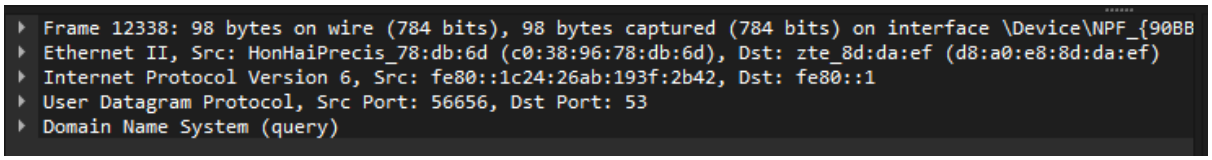
Aim:- To analyse DNS query traffic using WireShark.

1. Observe the traffic captured in the top Wireshark packet list pane. To view only DNS traffic, type **udp.port == 53** (lower case) in the Filter box and press **Enter.**



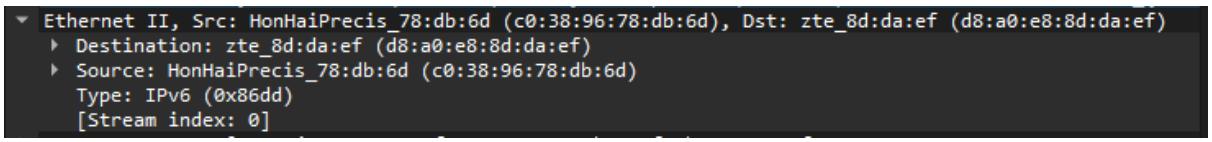

2. Select the DNS packet labeled **Standard query A en.wikiversity.org**.

3. Observe the packet details in the middle Wireshark packet details pane. Notice that it is an Ethernet II / Internet Protocol Version 4 / User Datagram Protocol / Domain Name System (query) frame.




4. Expand Ethernet II to view Ethernet details.

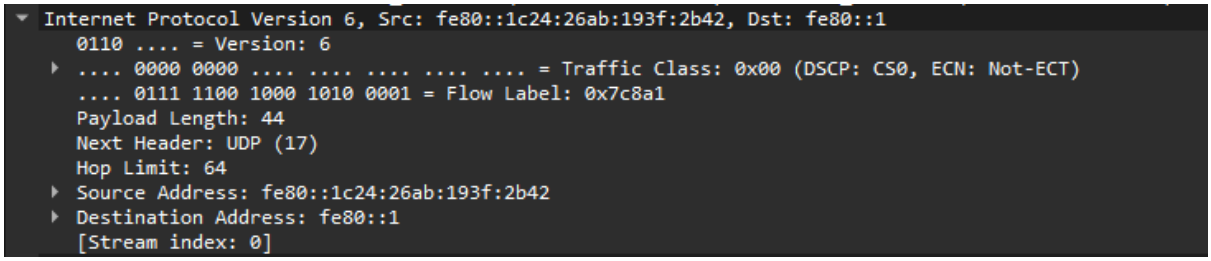
5. Observe the Destination and Source fields. The destination should be either your local DNS server's MAC address or your default gateway's MAC address and the source should be your MAC address. You can use ipconfig /all and arp -a to confirm.



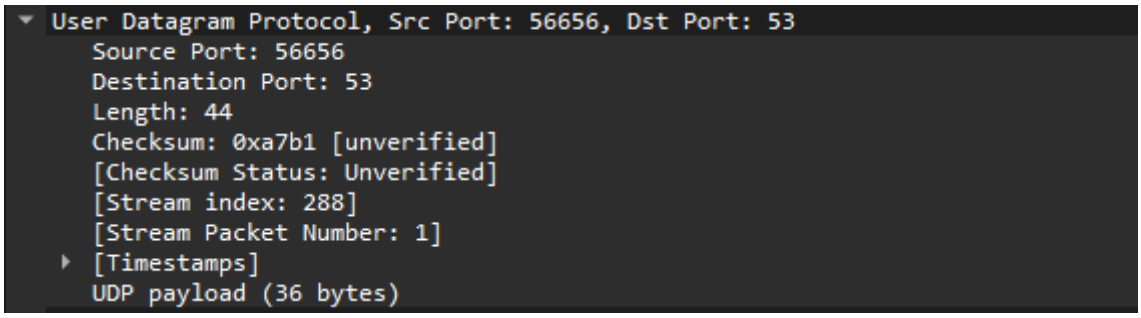

6. Expand Internet Protocol Version 4 to view IP details.

7. Observe the Source address. Notice that the source address is your IP address.

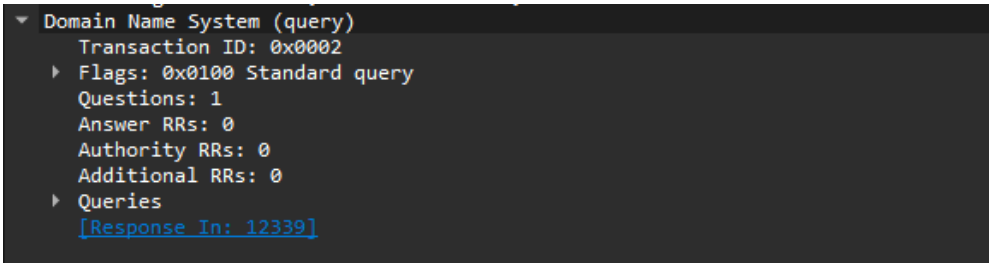
8. Observe the Destination address. Notice that the destination address is the IP address of the DNS server.



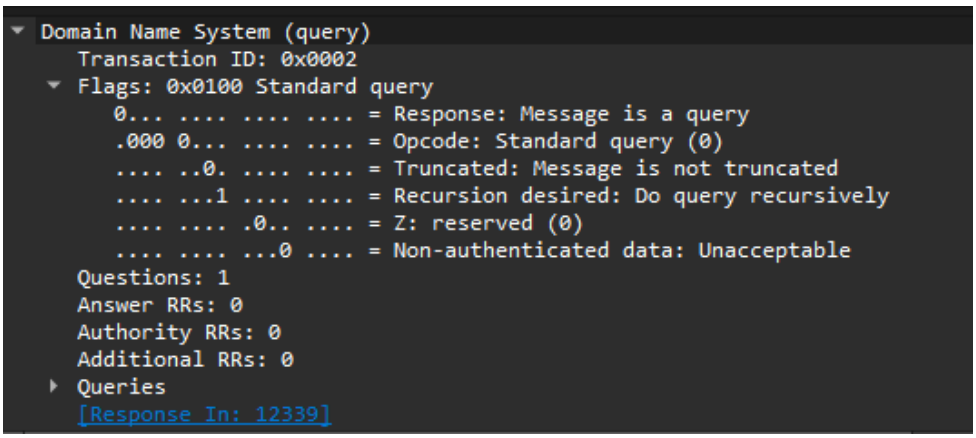

9. Expand User Datagram Protocol to view UDP details.
10. Observe the Source port. Notice that it is a dynamic port selected for this DNS query.
11. Observe the Destination port. Notice that it is domain (53), the DNS server port.




12. Expand Domain Name System (query) to view DNS details.

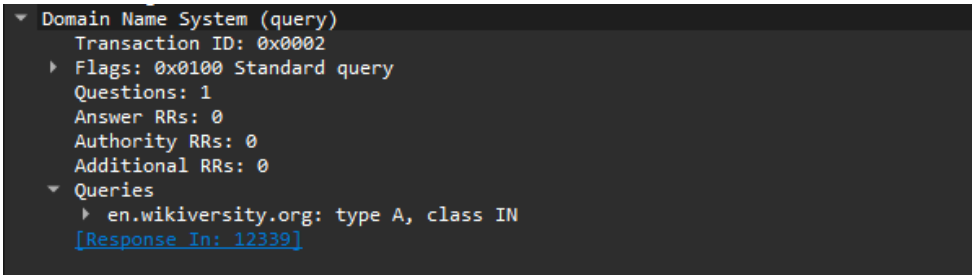



13. Expand Flags to view flags details.




14. Observe the Recursion desired field. Notice that a recursive query is requested.

15. Expand Queries to view query details.

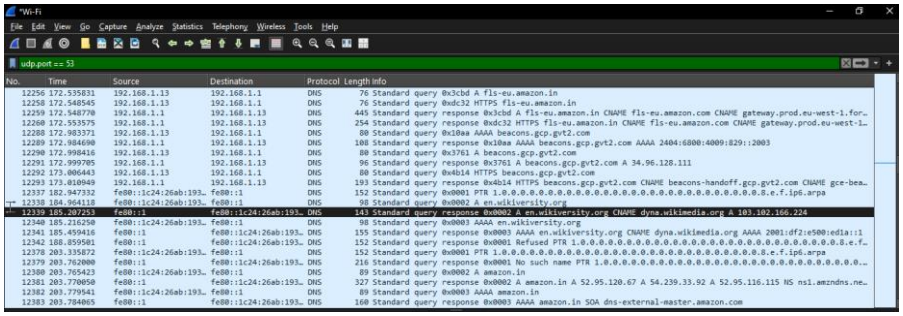



16. Observe the query for en.wikiversity.org.

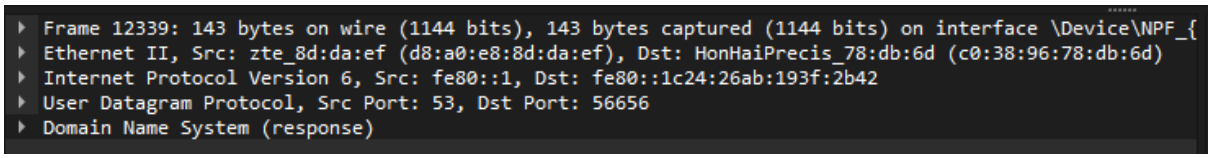
Practical 9

Aim:- To analyse DNS response traffic using WireShark.

1. In the top Wireshark packet list pane, select the next DNS packet, labeled **Standard query response CNAME wikiversity....**

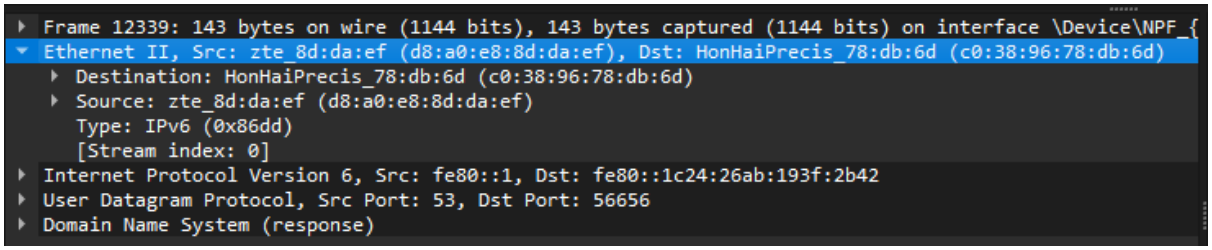



2. Observe the packet details in the middle Wireshark packet details pane. Notice that it is an Ethernet II / Internet Protocol Version 4 / User Datagram Protocol / Domain Name System (response) frame.




3. Expand Ethernet II to view Ethernet details.

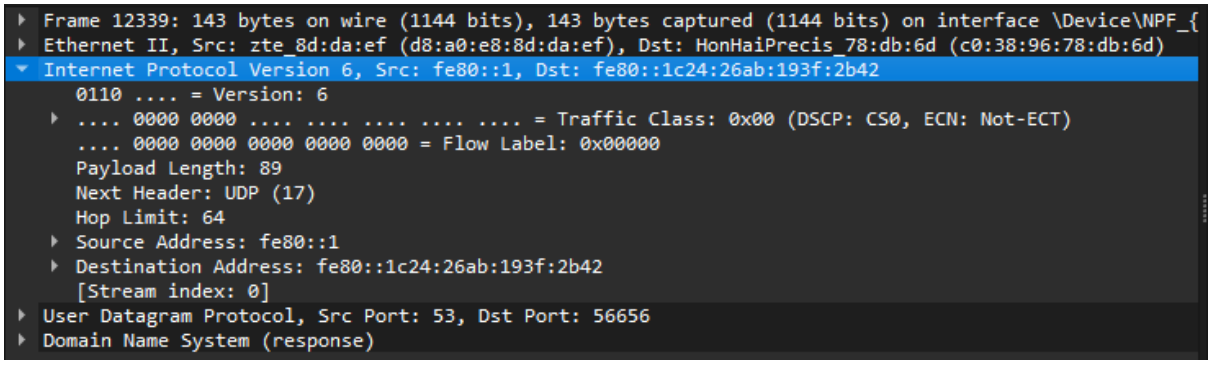
4. Observe the Destination and Source fields. The destination should be your MAC address and the source should be your local DNS server's MAC address or your default gateway's MAC address.



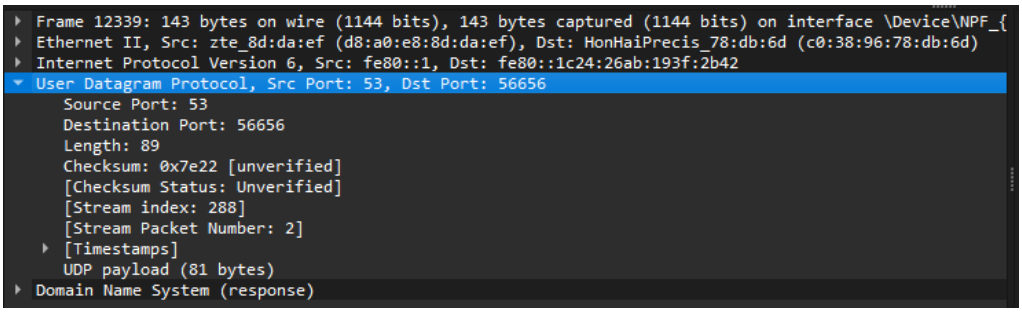

5. Expand Internet Protocol Version 4 to view IP details.

6. Observe the Source address. Notice that the source address is the DNS server IP address.

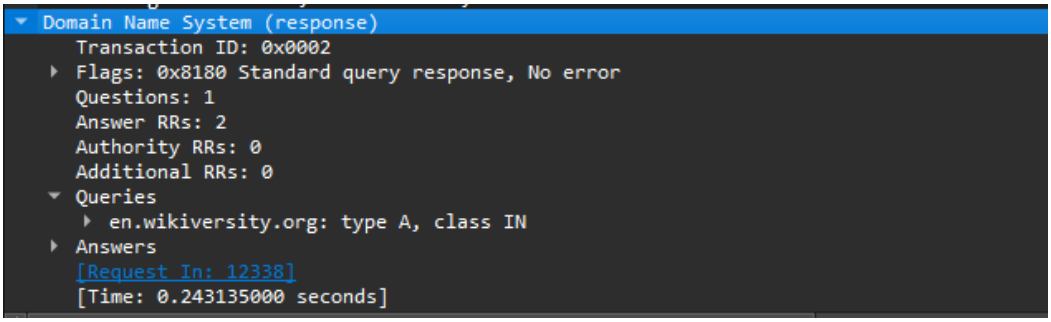
7. Observe the Destination address. Notice that the destination address is your IP address.



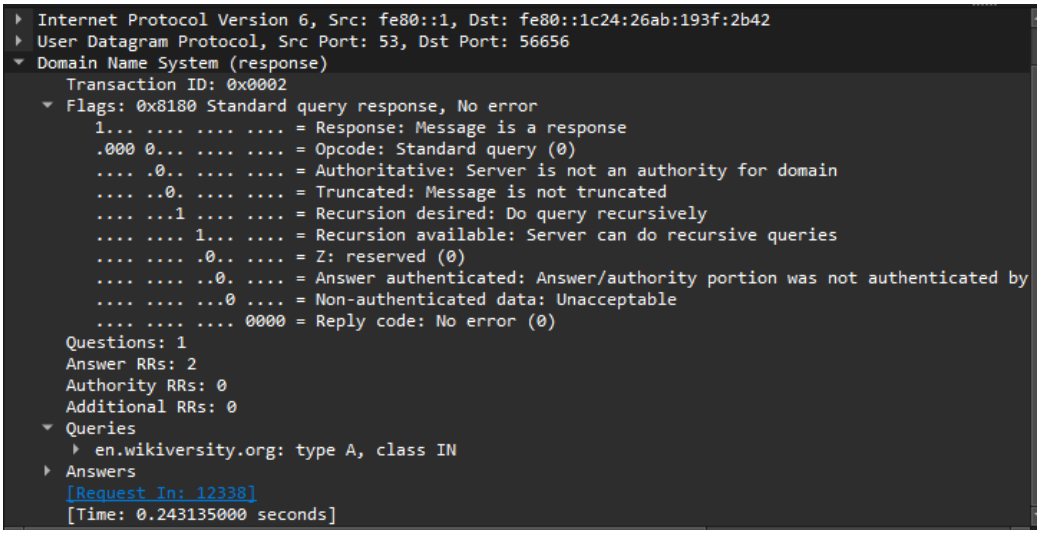

8. Expand User Datagram Protocol to view UDP details.
9. Observe the Source port. Notice that it is domain (53), the DNS server port.
10. Observe the Destination port. Notice that it is the same dynamic port used to make the DNS query in the first packet.



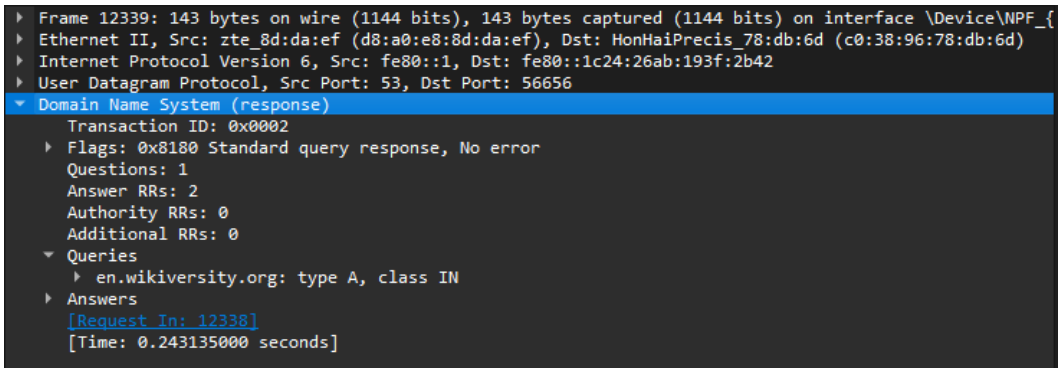

11. Expand Domain Name System (query) to view DNS details.




12. Expand Flags to view flags details.
13. Observe the flags. Notice that this is a recursive response.

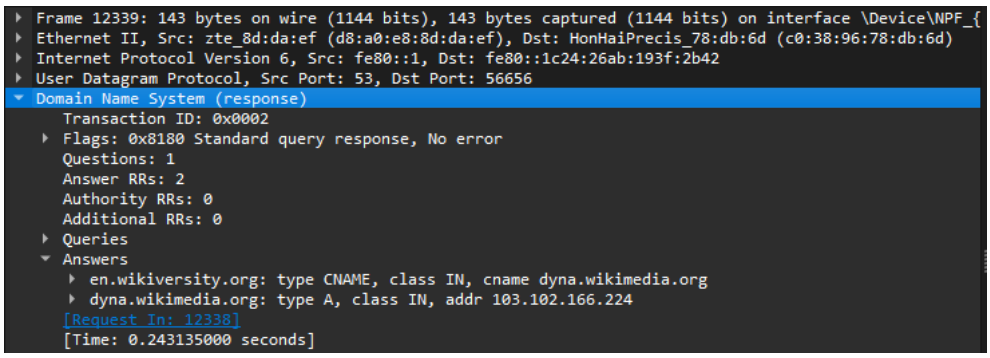



14. Expand Queries to view query details.




15. Observe the query for en.wikiversity.org.

16. Expand Answers to view answer details.




17. Observe the CNAME and A records returned in response to this DNS query.

18. Close Wireshark to complete this activity. **Quit without Saving** to discard the captured traffic.

Practical 10

Aim:- Splunk Case Study for Real Time Network

Splunk is a software technology that uses the data generated by the computer to track, scan, analyze, and visualize it in real-time. It tracks and read store data as indexer events and various types of log files. It enables us to view data in different Dashboard formats.

Splunk is a program that enables the search and analysis of computer data. It analyzes semi-structured data and logs generated by various processes with proper data modeling as per the need of the IT companies. The user produces the data by means of any device like- web apps, sensors, or computers. It has built-in functionality for defining data types, field separators, and search process optimization. For the searched result, it also provides visualization of data.

Students, IT developers, and experts in IT infrastructure management who want to develop a strong understanding of basic Splunk concepts must-read study this tutorial. You can attain intermediate expertise in Splunk after completing this tutorial, and quickly draw on your skills to solve more difficult problems.

The reader should be familiar with the language of querying, like SQL. General awareness of standard operations would be particularly useful when using computer applications such as data storage and retrieval and reading computer programs generated logs.

There is a variety of benefits that are offered by the Splunk, as follows:

- Real-time screen visibility.
- Splunk offers Better Interface.
- By offering instant results, it reduces troubleshooting and time-solving.
- It is the most effective method for the study of root causes.
- Splunk permits the generation of graphs, warnings, and dashboards.
- Similar findings can be quickly checked and analyzed using Splunk.
- It enables us to troubleshoot any failure state to improve performance.
- Helps you to track and make educated decisions on every company measure.
- Splunk allows Artificial Intelligence to be incorporated into the data strategy.
- Helps you to gather useful Operational Intelligence from your system data
- Splunk allows us to recognize any data type such as **.csv, json, log** formats, etc.
- Provides the most powerful search and visualization tools to enable all types of users.
- Allows us to establish a central server, where Splunk data can be searched from various sources.

Splunk has some essential features:

- It accelerates the Development & Testing.
- The building of Real-time Data Applications.
- Generate ROI faster
- Agile figures and Real-time architecture documentation.
- Splunk also provides search, analysis, and visualization capabilities to empower users.

Splunk Versions

There are three different versions of Splunk

- Splunk Enterprise
- Splunk Light
- Splunk Cloud

**Splunk Enterprise**

Big IT enterprise uses the Splunk Enterprise Version. With the help of the Splunk tool, we can collect and analyze the data from mobile phones, websites, and applications, etc.

Features of Splunk

We are going to tell you all the features of the Business version of the Spunk.

**Data Ingestion**

In Splunk, we can import or insert the date from different data formats like - JSON, XML, and weblogs and application logs that have unstructured system data. The unstructured data can be modeled as the consumer wants in a data structure.

**Data Indexing**

plunk indexes the ingested data for speedier search and query on different conditions.

**Data Searching**

Splunk analysis involves using the indexed data to establish graphs, to forecast future trends, and to find patterns in the data

**Using Alerts**

Used to trigger emails or RSS feeds when a certain requirement is identified in the data that is being analyzed.

**Dashboards**

When we searched anything, the search result is displayed in the dashboard in the form of maps, reports, pivots, etc.

**Data Model**

The indexed data may be modeled into one or more data sets based on domain expertise. It leads to more straightforward navigation by end-users who evaluate the business cases without understanding the language techniques used by Splunk to process information.

Important Blocks of Splunk useful for SOC:

- Splunk Attack Analyzer automatically navigates complex attack chains to detect credential phishing and malware threats, generating actionable insights and reducing the friction of repetitive manual tasks typically associated with investigating threats.
- Splunk Asset and Risk Intelligence is a security application that can help you discover assets on your network and shorten your investigations.
- Splunk Enterprise Security provides prebuilt content and searches to help focus security analysts on answering root-cause questions in real-time about malicious and anomalous events in the IT infrastructure.
- Splunk User Behavior Analytics detects unknown and hidden threats using unsupervised machine learning, reducing the need for manual baselining, rule creation, and customization.
- Splunk Security for SAP solutions provides powerful, data driven security functionality for threat hunting, risk reduction, and vulnerability mitigation. The solutions include the Splunk Security Add-on for SAP solutions and the Splunk Security App for SAP solutions.
- Splunk Infrastructure Monitoring, It Gain insights into and perform powerful, capable analytics on your infrastructure and resources across hybrid and multi-cloud environments with Splunk Infrastructure Monitoring. Infrastructure Monitoring offers support for a broad range of integrations for collecting all kinds of data, from system metrics for infrastructure components to custom data from your applications.

Important Links for case study:

1. https://www.splunk.com/en_us/customers/success-stories/splunk-at-tesco.html
2. https://www.splunk.com/en_us/customers/success-stories/splunk-centurylink.html
3. https://www.splunk.com/en_us/customers/success-stories/tokyo-stock-exchange.html
4. https://www.splunk.com/en_us/mclaren.html