JNAN VIKAS MANDAL'S

JVM'S DEGREE COLLEGE OF INFORMATION TECHNOLOGY, AIROLI,
NAAC RE-ACCREDITED Grade 'A+' (CGPA-3.31) NAVI MUMBAI – 400708

DEPARTMENT OF INFORMATION TECHNOLOGY

Practical Work Submitted in partial fulfilment of the

Requirements for the award of the Degree of

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

By

SONALI RUPCHAND THAKUR

Seat No: 1313626

ADVANCED AI (602)

MACHINE LEARNING (604)

SECURITY OPERATIONS CENTER (606b)

Under the esteemed guidance of

Mrs.Janhavi Kshirsagar

Mrs Bhagyashree Karnekar

Dr.Sanjivani Nalkar

ARTIFICIAL INTELLIGENCE

JNAN VIKAS MANDAL'S

Mohanlal Raichand Mehta College of Commerce
Diwali Maa College of Science
Amritlal Raichand Mehta College of Arts
Dr. R.T. Doshi College of Computer Science
NAAC Re-Accredited Grade 'A+' (CGPA : 3.31) (3rd Cycle)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that Sonali Rupchand Thakur bearing seat no. 1313626 has done the project work/journal work in the subject of Artificial intelligence of semester 3 Practical Examination during the academic year 2025-26 under the guidance of Mrs. Janhavi Kshirsagar being the partial requirement for the fulfilment of the curriculum of Degree in Master of Science in Information Technology under University of Mumbai.

Place: Airoli Date:

Sign of Subject in- Charge Sign of External Examiner

Sign of Coordinator

TABLE OF CONTENTS

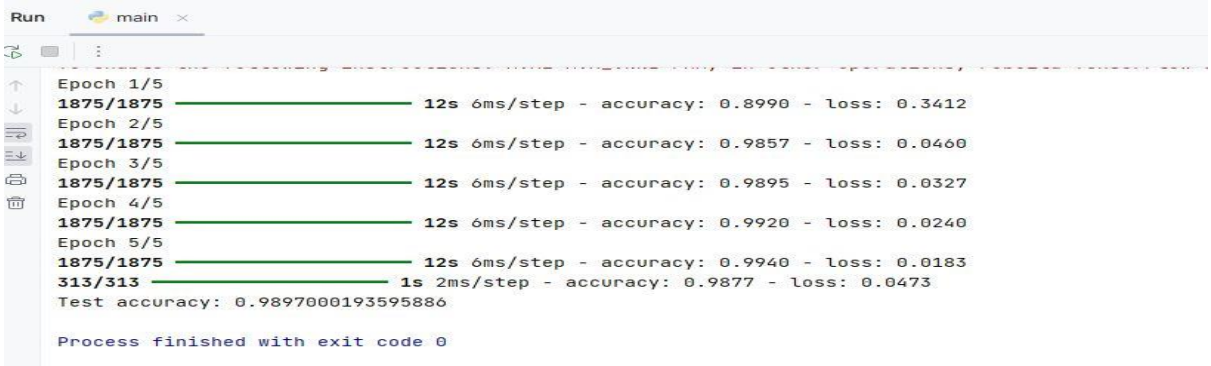
Practical No	Practicals	Date	Sign
1.	Implementing advanced deep learning algorithms such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs) using Python libraries like TensorFlow or PyTorch.		
2.	.Building a natural language processing (NLP) model for sentiment analysis or text classification.		
3.	Creating a chatbot using advanced techniques like transformer models.		
4.	Developing a recommendation system using collaborative filtering or deep learning approaches.		
5.	Implementing a computer vision project, such as object detection or image segmentation.		
6.	Training a generative adversarial network (GAN) for generating realistic images.		
7.	Applying reinforcement learning algorithms to solve complex decision-making problems.		
8.	Utilizing transfer learning to improve model performance on limited datasets.		
9.	Building a deep learning model for time series forecasting or anomaly detection.		
10.	Implementing a machine learning pipeline for automated feature engineering and model selection.		
11.	Using advanced optimization techniques like evolutionary algorithms or Bayesian optimization for hyperparameter tuning.		
12.	Use Python libraries such as GPT-2 or textgenrnn to train generative models on a corpus of text data and generate new text based on the patterns it has learned.		

Practical 1

Aim- Implementing advanced deep learning algorithms such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs) using Python libraries like TensorFlow or PyTorch.

```
import tensorflow as tf

from tensorflow.keras import layers, models

(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()

train_images = train_images.reshape((train_images.shape[0], 28, 28, 1))

test_images = test_images.reshape((test_images.shape[0], 28, 28, 1))

train_images, test_images = train_images / 255.0, test_images / 255.0

model = models.Sequential([

layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),

layers.MaxPooling2D((2, 2)),layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D((2, 2)),layers.Conv2D(64, (3, 3), activation='relu'),layers.Flatten(),

layers.Dense(64, activation='relu'), layers.Dense(10, activation='softmax')])

model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy']
) model.fit(train_images, train_labels, epochs=5)

test_loss, test_acc = model.evaluate(test_images, test_labels)

print(f'Test accuracy: {test_acc}')
```

OUTPUT:



Practical 2

Aim- Building a natural language processing (NLP) model for sentiment analysis or text classification.

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, classification_report

data = { 'text': [

    'I love this product, it is amazing!',

    'This is the worst purchase I have ever made.',

    'I am very happy with the service.',

    'I will never buy from this store again.',

    'The quality is excellent!',

    'Terrible experience, very disappointed.',

    'Absolutely fantastic! Highly recommend it.',

    'Not good at all, I want a refund.',

    'Very satisfied with my order.',

    'The item broke after one use, awful.'   ],   'label': ['positive', 'negative', 'positive', 'negative', 'positive', 'negative', 'positive', 'negative', 'positive', 'negative']

}df = pd.DataFrame(data)

X_train, X_test, y_train, y_test = train_test_split( df['text'], df['label'], test_size=0.2, random_state=42, stratify=df['label']

)vectorizer = CountVectorizer()

X_train_vectors = vectorizer.fit_transform(X_train)

X_test_vectors = vectorizer.transform(X_test)

model = MultinomialNB()model.fit(X_train_vectors, y_train)
```

```
y_pred = model.predict(X_test_vectors)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print('Classification Report:')
print(classification_report(y_test, y_pred, zero_division=0))
custom_reviews = [
    "This product exceeded my expectations!",
    "Worst service ever, completely dissatisfied.",
    "The experience was okay, not great, not terrible.",
    "Highly recommend this to everyone!"
]custom_review_vectors = vectorizer.transform(custom_reviews)
custom_predictions = model.predict(custom_review_vectors)
for review, sentiment in zip(custom_reviews, custom_predictions):
 print(f"Review: \"{review}\" -> Sentiment: {sentiment}")
```

OUTPUT:

```
============== RESTART: D:/Janu_mscIT_P2/AAI-practs/(NLP) model.py =============
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

    negative       1.00      1.00      1.00         1
    positive       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2

Review: "This product exceeded my expectations!" -> Sentiment: positive
Review: "Worst service ever, completely dissatisfied." -> Sentiment: negative
Review: "The experience was okay, not great, not terrible." -> Sentiment: negati
ve
Review: "Highly recommend this to everyone!" -> Sentiment: positive
```

Practical 3

Aim- Creating a chatbot using advanced techniques like transformer models.

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer,import torch,import os

os.environ["HF_HOME"] = "./hf_cache"

model_name = "microsoft/DialoGPT-medium"

try:

    model = GPT2LMHeadModel.from_pretrained(model_name)

    tokenizer = GPT2Tokenizer.from_pretrained(model_name)

except Exception as e:

    print(f"Error loading model: {e}")

    exit()

def chat_with_bot(input_text):

    try:

        inputs = tokenizer.encode(input_text + tokenizer.eos_token, return_tensors="pt")

        attention_mask = torch.ones(inputs.shape, dtype=torch.long)outputs = model.generate(
inputs,max_length=100,pad_token_id=tokenizer.eos_token_id,

attention_mask=attention_mask,              no_repeat_ngram_size=2,

  temperature=0.7,              top_p=0.9,  top_k=50, do_sample=True,          )

Decode the response back into text and filter it

        response = tokenizer.decode(outputs[:, inputs.shape[-1]:][0], skip_special_tokens=True)

        return filter_response(response)

    except Exception as e:

        return f"Error during response generation: {e}"

def filter_response(response):

    if len(response.strip()) < 5:  # Adjusted to allow shorter valid responses

        return "Could you say that again? I'm not sure I understood."

elif "?" in response:  # If response is a question, it might be a valid conversational turn
```

```
return response return response

if __name__ == "__main__":

    print("Chatbot: Hello! Type 'exit' to end the conversation.")

  while True:  user_input = input("You: ")

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

            print("Chatbot: Goodbye!") break

        response = chat_with_bot(user_input)

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

OUTPUT:

```
Chatbot: Hello! Type 'exit' to end the conversation.
You: Hello,how are you?
Chatbot: Hi, I'm good thanks!
You: i am also fine
Chatbot: okay im in the room now
```

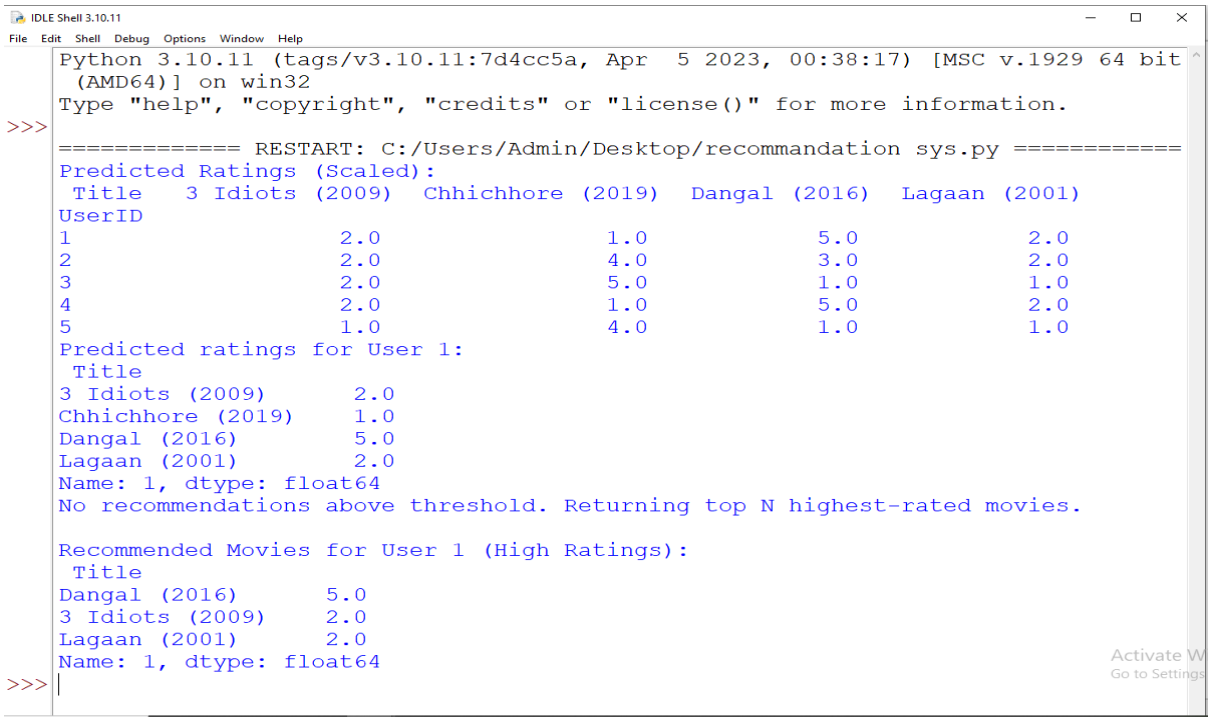
Practical 4

Aim- Developing a recommendation system using collaborative filtering or deep learning approaches.

```
import pandas as pd

import numpy as np

from sklearn.decomposition import TruncatedSVD

data = {

    "UserID": [1, 1, 2, 2, 3, 3, 4, 4, 5, 5],

    "MovieID": [101, 102, 101, 103, 102, 103, 101, 104, 103, 104],

    "Rating": [5, 4, 3, 4, 2, 5, 5, 3, 4, 2]}movies = {

    "MovieID": [101, 102, 103, 104],

    "Title": ["Dangal (2016)", "3 Idiots (2009)", "Chhichhore (2019)", "Lagaan (2001)"]}

ratings = pd.DataFrame(data)

movies = pd.DataFrame(movies)

ratings = pd.merge(ratings, movies, on="MovieID")

user_item_matrix = ratings.pivot_table(index="UserID", columns="Title",
values="Rating").fillna(0)

svd = TruncatedSVD(n_components=2)  # Use a smaller number of components for
simplicity

user_factors = svd.fit_transform(user_item_matrix)

item_factors = svd.components_

predicted_ratings = np.dot(user_factors, item_factors)

min_rating, max_rating = predicted_ratings.min(), predicted_ratings.max()

predicted_ratings = 1 + 4 * (predicted_ratings - min_rating) / (max_rating - min_rating)

predicted_df = pd.DataFrame(predicted_ratings, index=user_item_matrix.index,
columns=user_item_matrix.columns)

predicted_df = predicted_df.round()

print("Predicted Ratings (Scaled):\n", predicted_df)
```

```
def recommend_movies(user_id, user_item_matrix, predicted_df, top_n=3, threshold=2):
    user_ratings = predicted_df.loc[user_id]
    rated_movies = user_item_matrix.loc[user_id] > 0
    recommendations = user_ratings[~rated_movies]
    print(f"Predicted ratings for User {user_id}:\n", user_ratings)
    recommendations = recommendations[recommendations > threshold].sort_values(ascending=False)
  if recommendations.empty:
        print("No recommendations above threshold. Returning top N highest-rated movies.")
        recommendations = user_ratings.sort_values(ascending=False)
 return recommendations.head(top_n)
recommendations = recommend_movies(1, user_item_matrix, predicted_df, top_n=3, threshold=2)
print("\nRecommended Movies for User 1 (High Ratings):\n", recommendations)
```

OUTPUT:

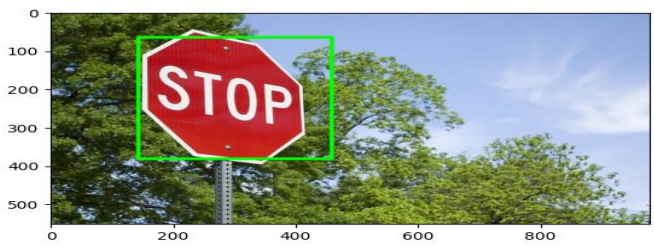


Practical 5

Aim- Implementing a computer vision project, such as object detection or image segmentation.

```
import cv2
from matplotlib import pyplot as plt
image_path = r"C:\Users\Janhavi\Downloads\stop.jpg"
cascade_path = r"C:\Users\Janhavi\Downloads\stop_data.xml"
img = cv2.imread(image_path)
if img is None:
   raise FileNotFoundError("Image file could not be loaded. Check the path.")
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
stop_data = cv2.CascadeClassifier(cascade_path)
if stop_data.empty():
    raise FileNotFoundError("Cascade file could not be loaded. Check the path.")
for (x, y, w, h) in stop_data.detectMultiScale(img_gray, minSize=(20, 20)):
 cv2.rectangle(img_rgb, (x, y), (x + h, y + w), (0, 255, 0), 5)
plt.imshow(img_rgb)
plt.show()
```

OUTPUT:



Practical 6

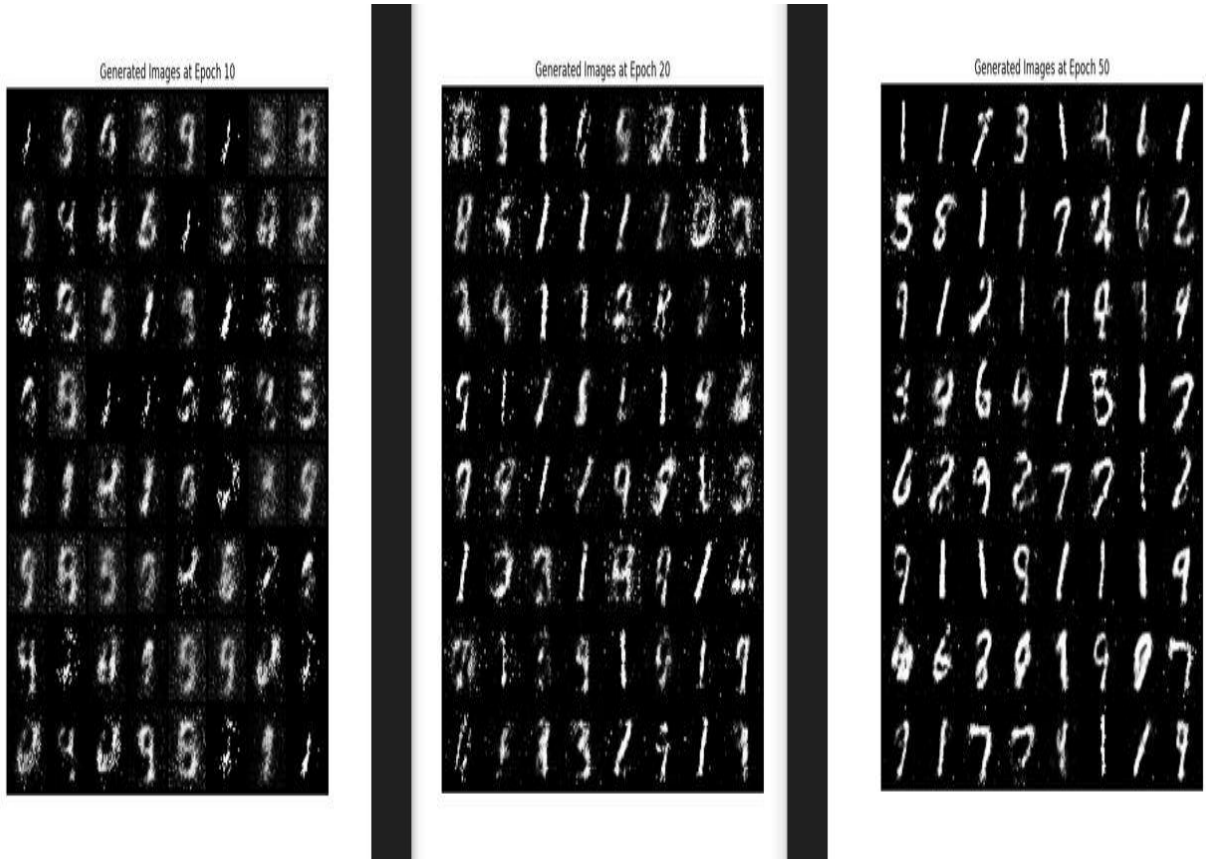
Aim- Training a generative adversarial network (GAN) for generating realistic images.

```
import torch

import torch.nn as nn

import torch.optim as optim

from torchvision import datasets, transforms

from torch.utils.data import DataLoader

import torchvision.utils as vutils

import matplotlib.pyplot as plt

class Generator(nn.Module):

    def __init__(self, input_size, output_size):

        super(Generator, self).__init__() self.model = nn.Sequential(nn.Linear(input_size, 128),

            nn.ReLU(),nn.Linear(128, 256),nn.BatchNorm1d(256), nn.ReLU(),nn.Linear(256,
512), nn.BatchNorm1d(512), nn.ReLU(),nn.Linear(512, output_size), nn.Tanh()        )

    def forward(self, x):

        return self.model(x)

class Discriminator(nn.Module):

    def __init__(self, input_size):

        super(Discriminator, self).__init__() self.model = nn.Sequential( nn.Linear(input_size,
512), nn.LeakyReLU(0.2), nn.Linear(512, 256), nn.LeakyReLU(0.2), nn.Linear(256, 1),
nn.Sigmoid()                    )

 def forward(self, x):

  return self.model(x)latent_size = 100 image_size = 28 * 28  batch_size = 64

epochs = 50 lr = 0.0002

transform = transforms.Compose([

    transforms.ToTensor(),

    transforms.Normalize(mean=[0.5], std=[0.5])])

dataset = datasets.MNIST(root="./data", train=True, transform=transform, download=True)
```

```
data_loader = DataLoader(dataset, batch_size=batch_size, shuffle=True)

generator = Generator(input_size=latent_size, output_size=image_size)

discriminator = Discriminator(input_size=image_size)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

generator.to(device) discriminator.to(device)

criterion = nn.BCELoss()

optimizer_g = optim.Adam(generator.parameters(), lr=lr)

optimizer_d = optim.Adam(discriminator.parameters(), lr=lr)

for epoch in range(epochs):

    for batch_idx, (real_images, _) in enumerate(data_loader):

        real_images = real_images.view(-1, image_size).to(device)  batch_size =
real_images.size(0) real_labels = torch.ones(batch_size, 1).to(device)fake_labels =
torch.zeros(batch_size, 1).to(device)outputs = discriminator(real_images)d_loss_real =
criterion(outputs, real_labels)z = torch.randn(batch_size, latent_size).to(device)fake_images
= generator(z)outputs = discriminator(fake_images.detach())d_loss_fake = criterion(outputs,
fake_labels)d_loss = d_loss_real + d_loss_fake
optimizer_d.zero_grad()d_loss.backward()optimizer_d.step()

  z = torch.randn(batch_size, latent_size).to(device)fake_images = generator(z)

  outputs = discriminator(fake_images) g_loss = criterion(outputs, real_labels)

optimizer_g.zero_grad()g_loss.backward()optimizer_g.step()

print(f"Epoch [{epoch + 1}/{epochs}]  D Loss: {d_loss.item():.4f}  G Loss:
{g_loss.item():.4f}")

    if (epoch + 1) % 10 == 0:

        with torch.no_grad():

            z = torch.randn(64, latent_size).to(device)

            fake_images = generator(z).view(-1, 1, 28, 28).cpu()

            print(f"Epoch {epoch + 1}: Generated Images Mean Pixel Value =
{fake_images.mean().item():.4f}")

 grid = vutils.make_grid(fake_images, nrow=8, normalize=True)

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

```
        plt.imshow(grid.permute(1, 2, 0).numpy(), cmap="gray")
        plt.title(f"Generated Images at Epoch {epoch + 1}")
        plt.axis("off")
        filename = f"generated_images_epoch_{epoch + 1}.png"
        plt.savefig(filename)
        plt.close()
print(f"Images saved to {filename}")
```

OUTPUT:



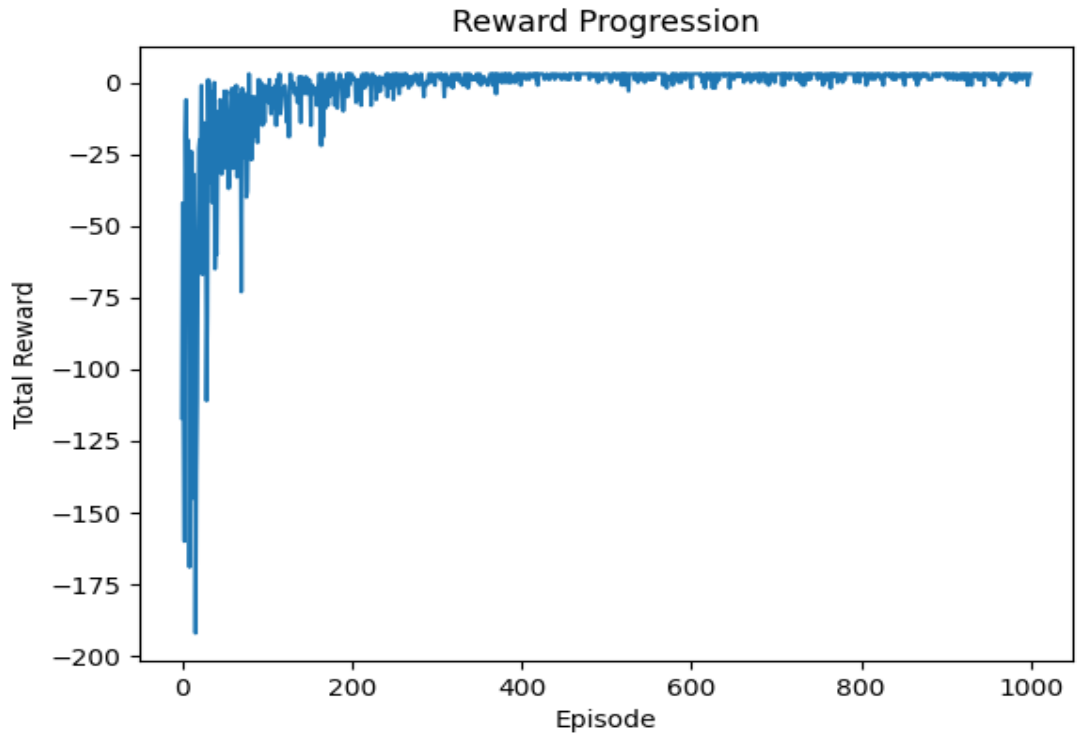
Practical 7

Aim- Applying reinforcement learning algorithms to solve complex decision-making problems.

```
import numpy as np
import matplotlib.pyplot as plt
class GridWorld:
    def __init__(self, size, start, goal, obstacles):
        self.size = size self.start = start self.goal = goal self.obstacles = obstacles self.reset()
   def reset(self):
        self.state = self.start  return self.state def step(self, action):
        x, y = self.state
        if action == 0:  # Up
            next_state = (max(x - 1, 0), y)
        elif action == 1:  # Down
            next_state = (min(x + 1, self.size - 1), y)
        elif action == 2:  # Left
            next_state = (x, max(y - 1, 0))
        elif action == 3:  # Right
            next_state = (x, min(y + 1, self.size - 1))
        if next_state in self.obstacles:
            next_state = self.state  reward = -1  done = False
 if next_state == self.goal:
            reward = 10   done = True
self.state = next_state
        return next_state, reward, done
start = (0, 0)  goal = (4, 4)  obstacles = [(2, 2), (3, 2)]
env = GridWorld(size, start, goal, obstacles)
```

```
actions = 4  q_table = np.zeros((size, size, actions))  episodes = 1000 alpha = 0.1

gamma = 0.9   epsilon = 1.0   epsilon_decay = 0.995 min_epsilon = 0.1

rewards = []

for episode in range(episodes): state = env.reset()      total_reward = 0 while not done:

      x, y = state if np.random.uniform(0, 1) < epsilon: action = np.random.choice(actions)  # Explore          else: action = np.argmax(q_table[x, y])

 next_state, reward, done = env.step(action)  next_x, next_y = next_state

plt.plot(rewards)  plt.xlabel('Episode')  plt.ylabel('Total Reward')

plt.title('Reward Progression') plt.show()

print("Learned Q-values:")

for i in range(size):

   for j in range(size):

      print(f"({i}, {j}): {q_table[i, j]}")
```

OUTPUT:



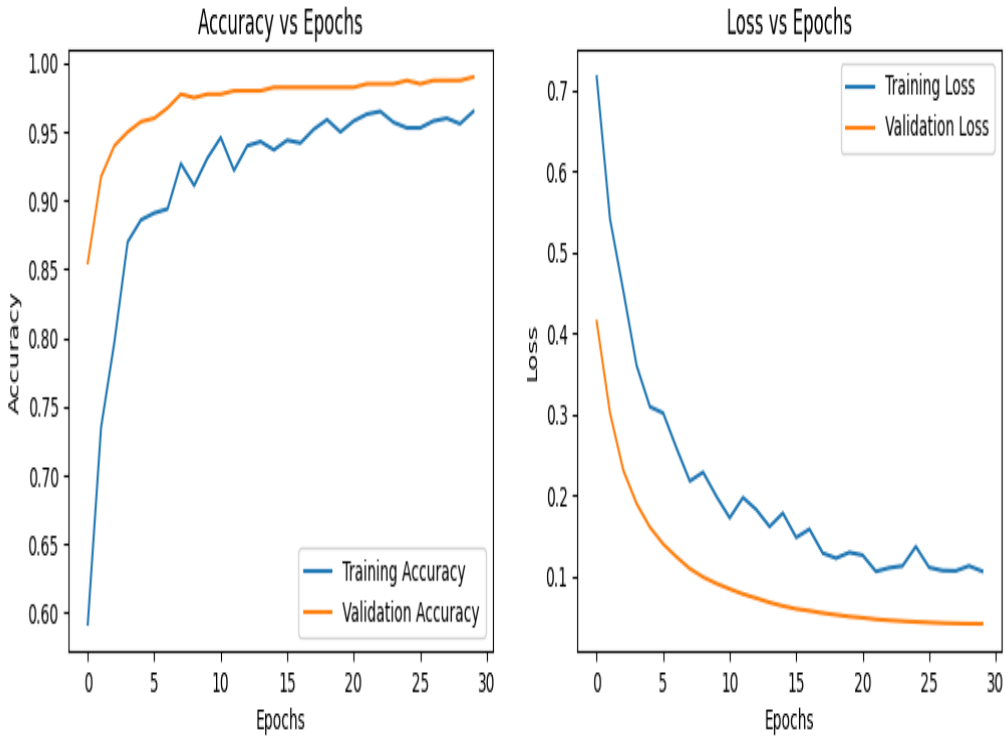
Practical 8

Aim- Utilizing transfer learning to improve model performance on limited datasets.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from PIL import Image
import matplotlib.pyplot as plt
train_dir = r'C:\Users\Janhavi\Downloads\cats_dogs_light\train'
validation_dir = r'C:\Users\Janhavi\Downloads\cats_dogs_light\test'
print(f"Train directory exists: {os.path.isdir(train_dir)}")
print(f"Validation directory exists: {os.path.isdir(validation_dir)}")
print(f"Found {len(os.listdir(train_dir))} subdirectories in the train directory.")
print(f"Found {len(os.listdir(validation_dir))} subdirectories in the validation directory.")
def check_images(directory):
    print(f"Checking files in {directory}...")
    for subdir in os.listdir(directory):
        subdir_path = os.path.join(directory, subdir)
        if os.path.isdir(subdir_path):
            for file_name in os.listdir(subdir_path):
                file_path = os.path.join(subdir_path, file_name)
for layer in base_model.layers[:-10]:  # Unfreeze all but the last 10 layers
    layer.trainable = False model = Sequential([  base_model,
GlobalAveragePooling2D(),Dense(64, activation='relu'),  # Reduced neurons for simplicity
    Dropout(0.5),  # Dropout to prevent overfitting
    Dense(1, activation='sigmoid')  # Binary classification])
model.compile(
```

```
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),  # Use lower learning rate
    loss='binary_crossentropy',    metrics=['accuracy'] )
train_generator = train_datagen.flow_from_directory(train_dir, target_size=(224, 224),
batch_size=32, class_mode='binary'  # Binary classification (cats vs dogs)history = model.fit(
    train_generator,    validation_data=validation_generator, epochs=30,
callbacks=[early_stopping, lr_scheduler]plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1) plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy') plt.ylabel('Accuracy')
plt.legend()plt.title('Accuracy vs Epochs') plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss') plt.xlabel('Epochs')
plt.ylabel('Loss') plt.legend()
plt.title('Loss vs Epochs')
plt.show()
```

OUTPUT:



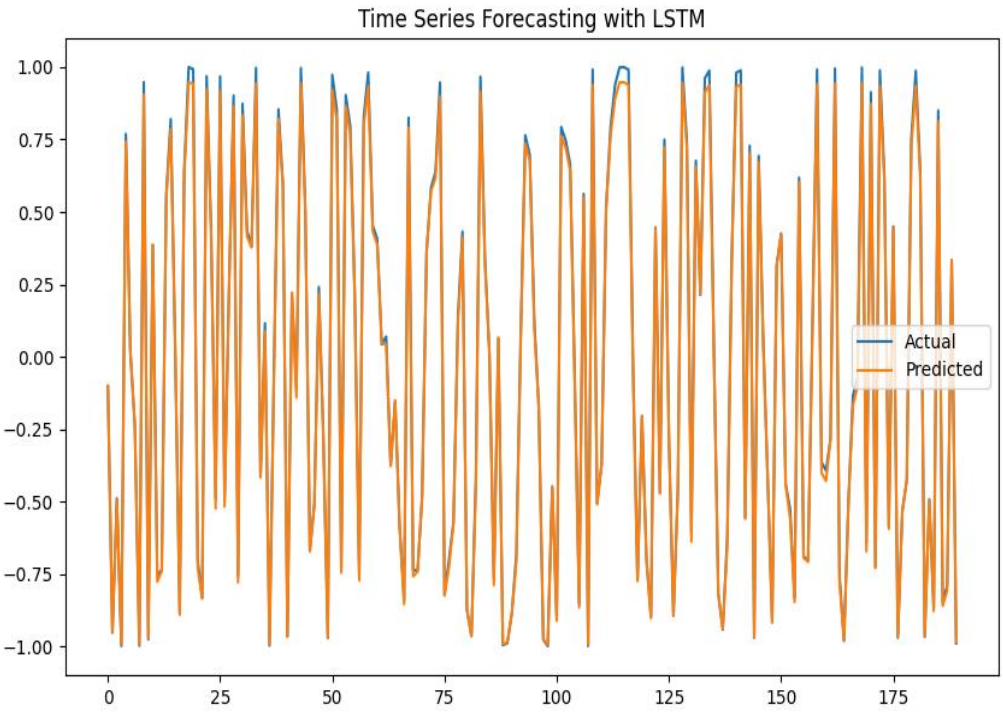
Practical 9

Aim- Building a deep learning model for time series forecasting or anomaly detection.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Dropout

from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split

def generate_sine_wave(length):

    x = np.linspace(0, 50, length)  y = np.sin(x) return y

def create_dataset(data, look_back):

    X, y = [], [] for i in range(len(data) - look_back):X.append(data[i:i + look_back])
y.append(data[i + look_back]) return np.array(X), np.array(y)

TIME_STEPS = 50  # Look-back window size EPOCHS = 20 BATCH_SIZE = 32

data = generate_sine_wave(1000) scaler = MinMaxScaler(feature_range=(0, 1))

scaled_data = scaler.fit_transform(data.reshape(-1, 1)).flatten()

X, y = create_dataset(scaled_data, TIME_STEPS)

X = np.expand_dims(X, axis=-1)  # Add a channel dimension for LSTM

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

model = Sequential([LSTM(50, return_sequences=True, input_shape=(TIME_STEPS, 1)),

Dropout(0.2), LSTM(50), Dropout(0.2), Dense(1)])model.compile(optimizer='adam',
loss='mean_squared_error')

history = model.fit(X_train, y_train, epochs=EPOCHS, batch_size=BATCH_SIZE,
validation_data=(X_test, y_test))

train_loss = model.evaluate(X_train, y_train, verbose=0)

test_loss = model.evaluate(X_test, y_test, verbose=0)

print(f"Train Loss: {train_loss:.4f}, Test Loss: {test_loss:.4f}")
```

```
predicted = model.predict(X_test)

predicted = scaler.inverse_transform(predicted)

y_test_actual = scaler.inverse_transform(y_test.reshape(-1, 1))

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

plt.plot(y_test_actual, label='Actual')

plt.plot(predicted, label='Predicted')

plt.legend()

plt.title("Time Series Forecasting with LSTM")

plt.show()
```

OUTPUT:



Practical 10

Aim- Implementing a machine learning pipeline for automated feature engineering and model selection.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
import lightgbm as lgb
import matplotlib.pyplot as plt
url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"
data = pd.read_csv(url)
data = data.drop(columns=['Name', 'Ticket', 'Cabin'])  # Drop irrelevant columns
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})  # Encode 'Sex' feature
data = pd.get_dummies(data, drop_first=True)  # One-hot encoding for 'Embarked'
X = data.drop(columns='Survived')
y = data['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
numeric_features = X.select_dtypes(include=['int64', 'float64']).columns
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),  # Handle missing values
    ('scaler', StandardScaler())  # Feature scaling])
categorical_features = X.select_dtypes(include=['object']).columns
    ('encoder', 'passthrough')  # Use 'passthrough' to handle categorical features in pd.get_dummies outside the pipeline
])preprocessor = ColumnTransformer( transformers=[('num', numeric_transformer, numeric_features), ('cat', categorical_transformer, categorical_features)])
```

```
models = [
    ('Logistic Regression', LogisticRegression(max_iter=1000)),
    ('Random Forest', RandomForestClassifier()),
    ('SVM', SVC()),    ('XGBoost', xgb.XGBClassifier()),    ('LightGBM',
lgb.LGBMClassifier())]
    cv_score = cross_val_score(pipeline, X_train, y_train, cv=5, scoring='accuracy')
    mean_score = np.mean(cv_score)
if hasattr(best_pipeline.named_steps['classifier'], 'feature_importances_'):
    importances = best_pipeline.named_steps['classifier'].feature_importances_
    indices = np.argsort(importances)[::-1]
    plt.figure(figsize=(10, 6))
    plt.title(f"Feature Importances for {best_model}")
    plt.bar(range(X_train.shape[1]), importances[indices], align='center')
    plt.xticks(range(X_train.shape[1]), X_train.columns[indices], rotation=90)
    plt.show()
```

OUTPUT:

```
Training Logistic Regression...
Logistic Regression Mean Cross-Validation Score: 0.7913
Training Random Forest...
Random Forest Mean Cross-Validation Score: 0.8058
Training SVM...
SVM Mean Cross-Validation Score: 0.8170
Training XGBoost...
XGBoost Mean Cross-Validation Score: 0.7978
Training LightGBM...
[LightGBM] [Info] Number of positive: 184, number of negative: 314
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000175 seconds.
```

Practical 11

Aim- Using advanced optimization techniques like evolutionary algorithms or Bayesian optimization for hyperparameter tuning.

```
import os
os.environ["OPTUNA_NO_COLOR"] = "1"  # Disable colored output
import optuna
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
data = load_iris()
X, y = data.data, data.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
def objective(trial):
    # Define hyperparameter search space
    n_estimators = trial.suggest_int("n_estimators", 10, 200)
    max_depth = trial.suggest_int("max_depth", 2, 20)
    min_samples_split = trial.suggest_int("min_samples_split", 2, 20)
    min_samples_leaf = trial.suggest_int("min_samples_leaf", 1, 10)
    clf = RandomForestClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth,
        min_samples_split=min_samples_split,
        min_samples_leaf=min_samples_leaf,
        random_state=42, )
    score = cross_val_score(clf, X_train, y_train, cv=5, scoring="accuracy").mean()
    return score
```

```
study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=50)
print("Best hyperparameters:", study.best_params)
best_params = study.best_params
best_model = RandomForestClassifier(**best_params, random_state=42)
best_model.fit(X_train, y_train)
accuracy = best_model.score(X_test, y_test)
print(f"Accuracy with best hyperparameters: {accuracy:.2f}")
```

OUTPUT:

```
meters: {'n_estimators': 154, 'max_depth': 6, 'min_samples_split': 5, 'min_sampl
es_leaf': 3}. Best is trial 1 with value: 0.9583333333333334.[0m
[32m[I 2024-11-22 19:24:57,858][0m Trial 49 finished with value: 0.94166666666
66667 and parameters: {'n_estimators': 144, 'max_depth': 9, 'min_samples_split':
 6, 'min_samples_leaf': 4}. Best is trial 1 with value: 0.9583333333333334.[0m
Best hyperparameters: {'n_estimators': 92, 'max_depth': 20, 'min_samples_split':
 15, 'min_samples_leaf': 3}
Accuracy with best hyperparameters: 1.00
>>>
```

Practical 12

Aim- Use Python libraries such as GPT-2 or textgenrnn to train generative models on a corpus of text data and generate new text based on the patterns it has learned.

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer

model_name = "gpt2"  # You can change to a larger version like 'gpt2-medium' if needed

tokenizer = GPT2Tokenizer.from_pretrained(model_name)

model = GPT2LMHeadModel.from_pretrained(model_name)

tokenizer.pad_token = tokenizer.eos_token

prompt = "Once upon a time in a faraway land"

inputs = tokenizer.encode(prompt, return_tensors="pt")

outputs = model.generate(inputs, max_length=50,

    num_return_sequences=5, do_sample=True,    top_k=50,  top_p=0.95,

     attention_mask=None,   pad_token_id=tokenizer.eos_token_id)  # Set pad_token_id to eos_token_id

print("\nGenerated Texts:") for i, output in enumerate(outputs, start=1): print(f"{i}: {tokenizer.decode(output, skip_special_tokens=True)}")
```

OUTPUT:

```
Generated Texts:
1: Once upon a time in a faraway land, when in my travels the wind and fire are
both so keen they break down to pieces, yet in a vast place a stone is broken, t
hough it cannot break. When in a faraway land,
2: Once upon a time in a faraway land there is a mighty earthquake, and all that
 can be seen is that there is an enormous and massive earth that is completely c
overed with molten lava, and there is still a great amount of energy left over,
3: Once upon a time in a faraway land in the South, he stood on a ledge on which
 his body was resting. He saw in its depths a woman with no name or name of any
kind with his wife. And he took her out to
4: Once upon a time in a faraway land, a spirit spoke to the inhabitants and tol
d them to stop. The inhabitants thought they had been tricked by the spirit and
were prepared to fight it for their lives with the hope of having the spirit out
 of
5: Once upon a time in a faraway land, there existed a civilization of humans an
d elves. One of their basic needs was to produce food, so that humans could get
by without relying on agriculture for their livelihood. Many of the human races
of the
>>>
```

MACHINE LEARNING

JNAN VIKAS MANDAL'S

Mohanlal Raichand Mehta College of Commerce
Diwali Maa College of Science
Amritlal Raichand Mehta College of Arts
Dr. R.T. Doshi College of Computer Science
NAAC Re-Accredited Grade 'A+' (CGPA : 3.31) (3rd Cycle)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that Sonali Rupchand Thakur bearing seat no. 1313626 has done the project work/journal work in the subject of Machine Learning of semester III Practical Examination during the academic year 2025-26 under the guidance of **Mrs Bhagyashree Karnekar** being the partial requirement for the fulfilment of the curriculum of Degree in Master of Science in Information Technology under University of Mumbai

Place: Airoli Date:

Sign of Subject in- Charge Sign of External Examiner

Sign of Coordinator

Index

Sr.No	Practical	Date	Sign
1.	a)Load a CSV dataset. Handle missing values, inconsistent formatting, and outliers. b) Load a dataset, calculate descriptive summary statistics, create visualizations using different graphs, and identify potential features and target variables c) Create or Explore datasets to use all pre-processing routines like label encoding,scaling, and binarization.		
2.	Implement and demonstrate the Find-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a. CSV file and generate the final specific hypothesis. (Create your dataset)		
3.	a)Fit a linear regression model on a dataset. Interpret coefficients, make predictions, and evaluate performance using metrics like R-squared and MSE b) Extend linear regression to multiple features. Handle feature selection and potential multicollinearity. c)Implement regression variants like LASSO and Ridge on any generated dataset.		
4.	A) Perform binary classification using logistic regression. Calculate accuracy,precision, recall, and understand the ROC curve. B) Implement and demonstrate k-nearest Neighbor algorithm. Read the training data from a .CSV file and build the model to classify a test sample. Print both correct and wrong predictions. C)Build a decision tree classifier or regressor. Control hyperparameters like tree depth to avoid overfitting. Visualize the tree. D)Implement a Support Vector Machine for any relevant dataset. E) Train a random forest ensemble. Experiment with the number of trees and feature sampling. Compare performance to a single decision tree. F)Implement a gradient boos ng machine (e.g., XGBoost). Tune hyperparameters and explore feature importance.		
5.	A)Implement and demonstrate the working of a Naive Bayesian classifier using a sample data set. Build the model to classify a test sample B) Implement Hidden Markov Models using hmmlearn		
6.	A) Implement Bayesian Linear Regression to explore prior and posterior distribution B) Implement Gaussian Mixture Models for density es ma on and unsupervised clustering		
7.	A) Implement cross-validation techniques (k-fold, stratified, etc.) for robust model Evaluation B) Systematically explore combinations of hyperparameters to optimize model performance. (Use grid and randomized search)		
8.	Implement Bayesian Learning using inferences		

PRACTICAL NO 1

Aim-Load a CSV dataset. Handle missing values, inconsistent formatting, and outliers.

```
import pandas as pd
import numpy as np
# Step 1: Load the dataset
file_path = 'C:/Users/admin/Downloads/enjoysport.csv'
df = pd.read_csv(file_path)
print("Original Dataset:\n", df.head())
# Step 2: Handle inconsistent formatting for string columns only
for col in df.select_dtypes(include=['object', 'string']):
    df[col] = df[col].map(lambda x: x.strip().lower() if isinstance(x, str) else x)
for col in df.columns:
    if df[col].dtype == 'object' or df[col].dtype.name == 'string':
        df[col] = df[col].fillna(df[col].mode(dropna=True)[0])  # Fill categorical with mode
    else:
        df[col] = df[col].fillna(df[col].mean())
print("\nCleaned Dataset:\n", df.head())
```

OUTPUT

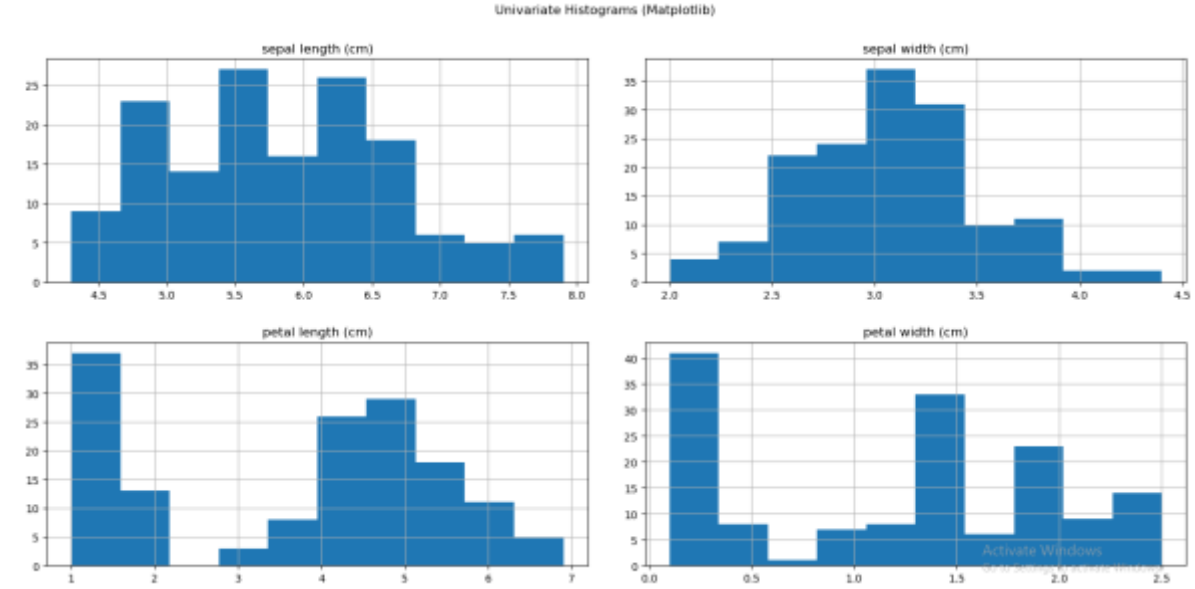
```
(base) C:\Users\admin>python Desktop/MLPRAC/prac2.py
Original Dataset:
      sky airtemp humidity    wind water forcast enjoysport
0  sunny    warm   normal  strong  warm    same        yes
1  sunny    warm     high  strong  warm    same        yes
2  rainy    cold     high  strong  warm  change         no
3  sunny    warm     high  strong  cool  change        yes

Cleaned Dataset:
      sky airtemp humidity    wind water forcast enjoysport
0  sunny    warm   normal  strong  warm    same        yes
1  sunny    warm     high  strong  warm    same        yes
2  rainy    cold     high  strong  warm  change         no
3  sunny    warm     high  strong  cool  change        yes
```

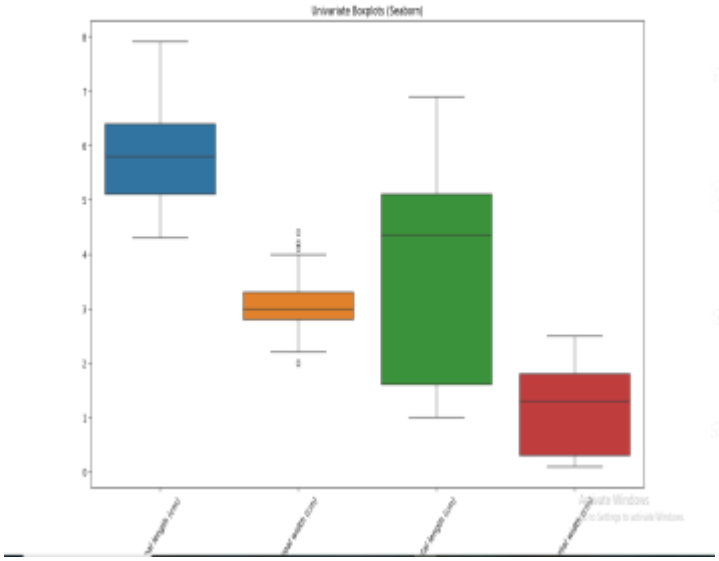
B)Load a dataset, calculate descriptive summary statistics, create visualizations using different graphs, and identify potential features and target variables
Note: Explore Univariate and Bivariate graphs (Matplotlib) and Seaborn for visualization.

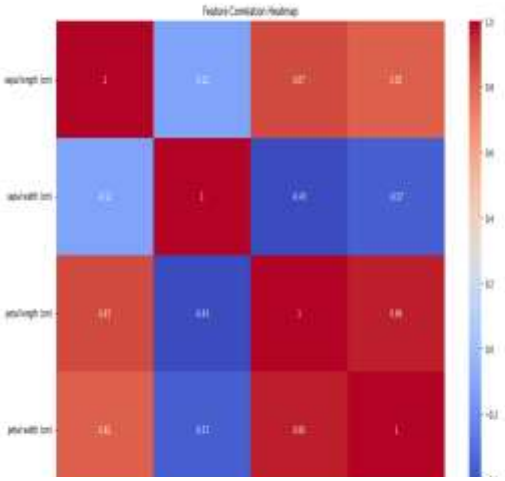
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)
df.hist(figsize=(10, 6), edgecolor='black')
plt.suptitle("Univariate Histograms (Matplotlib)", fontsize=14)
plt.tight_layout()
plt.show()
plt.figure(figsize=(10, 6))
sns.boxplot(data=df.drop(columns='species'))
plt.title("Univariate Boxplots (Seaborn)", fontsize=14)
plt.xticks(rotation=45)
plt.show()
sns.pairplot(df, hue="species", corner=True, diag_kind='hist')
plt.suptitle("Bivariate Scatter Plots (Pairplot)", y=1.02, fontsize=14)
plt.show()
plt.figure(figsize=(8, 5))
sns.heatmap(df.drop('species', axis=1).corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Feature Correlation Heatmap", fontsize=14)
plt.show()
features = df.columns[:-1]
target = 'species'
print("\nFeatures:", list(features))
print("Target:", target)
```

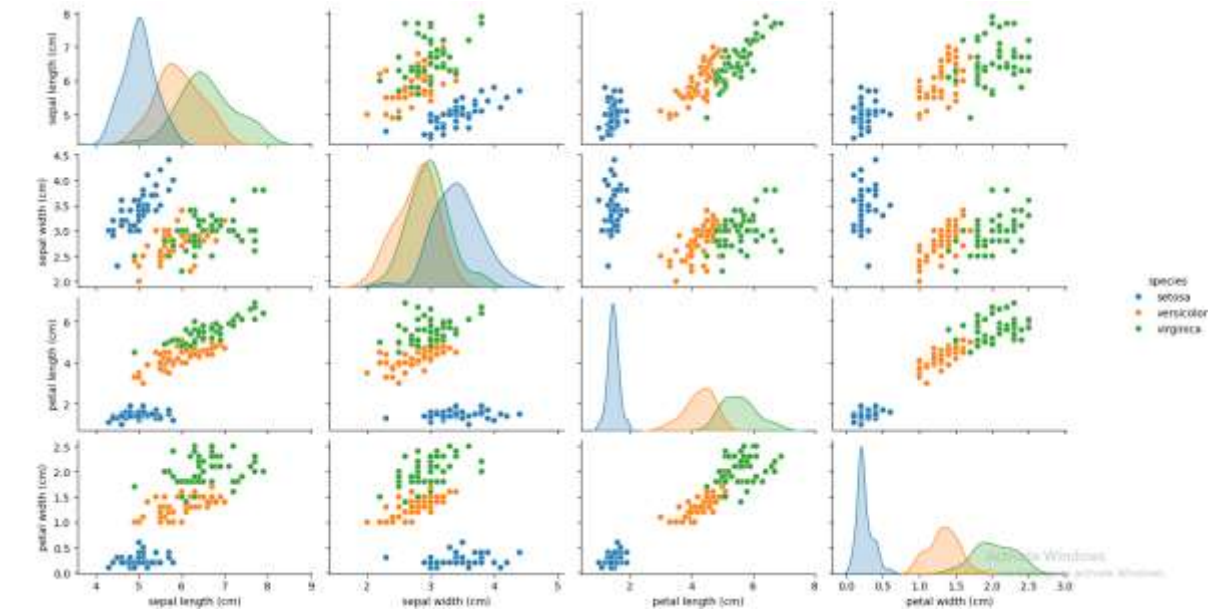
OUTPUT

```
(base) C:\Users\admin>python Desktop/MLPRAC/prac1b.py
Descriptive Statistics:

       sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
count         150.000000        150.000000         150.000000        150.000000
mean            5.843333          3.057333           3.758000          1.199333
std             0.828066          0.435866           1.765298          0.762238
min             4.300000          2.000000           1.000000          0.100000
25%             5.100000          2.800000           1.600000          0.300000
50%             5.800000          3.000000           4.350000          1.300000
75%             6.400000          3.300000           5.100000          1.800000
max             7.900000          4.400000           6.900000          2.500000
```


Univariate Histograms (Matplotlib)
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)


Univariate Boxplots (Seaborn)


Feature Correlation Heatmap


sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
species
setosa
versicolor
virginica

C)Create or Explore datasets to use all pre-processing routines like label encoding, scaling, and binarization.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, MinMaxScaler, StandardScaler,
Binarizer
data = {
    'City': ['Mumbai', 'Delhi', 'Chennai', 'Delhi', 'Mumbai'],
    'Temperature': [33, 45, 30, 40, 35],
    'Humidity': [70, 60, 80, 75, 72],
    'Rainy': ['Yes', 'No', 'Yes', 'No', 'Yes']
}

df = pd.DataFrame(data)
print("Original Data:\n", df)
le = LabelEncoder()
df['City_Label'] = le.fit_transform(df['City'])
df['Rainy_Label'] = le.fit_transform(df['Rainy'])
ohe = OneHotEncoder(sparse_output=False)
city_encoded = ohe.fit_transform(df[['City']])
city_df = pd.DataFrame(city_encoded, columns=ohe.get_feature_names_out(['City']))
df = pd.concat([df, city_df], axis=1)
scaler_minmax = MinMaxScaler()
df[['Temperature_MinMax', 'Humidity_MinMax']] = scaler_minmax.fit_transform(df[['Temperature',
'Humidity']])
scaler_std = StandardScaler()
df[['Temperature_Std', 'Humidity_Std']] = scaler_std.fit_transform(df[['Temperature', 'Humidity']])
binarizer = Binarizer(threshold=70)
df['Humidity_Binarized'] = binarizer.fit_transform(df[['Humidity']])
print("\nProcessed Data:\n", df)
```

```
(base) C:\Users\admin>python Desktop/MLPRAC/prac1c.py
Original Data:
      City  Temperature  Humidity Rainy
0   Mumbai           33        70   Yes
1    Delhi           45        60    No
2  Chennai           30        80   Yes
3    Delhi           40        75    No
4   Mumbai           35        72   Yes

Processed Data:
      City  Temperature  Humidity Rainy  ...  Humidity_MinMax  Temperature_Std  Humidity_Std  Humidity_Binarized
0   Mumbai           33        70   Yes  ...             0.50        -0.677430     -0.211443                   0
1    Delhi           45        60    No  ...             0.00         1.580691     -1.723748                   0
2  Chennai           30        80   Yes  ...             1.00        -1.241971      1.298863                   1
3    Delhi           40        75    No  ...             0.75         0.639803      0.543710                   1
4   Mumbai           35        72   Yes  ...             0.60        -0.301084      0.090618                   1

[5 rows x 14 columns]
```

Practical no 2

Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a. CSV file and generate the final specific hypothesis. (Create your dataset)

```
import pandas as pd
data = {
    'Sky': ['Sunny', 'Sunny', 'Rainy', 'Sunny', 'Sunny'],
    'AirTemp': ['Warm', 'Warm', 'Cold', 'Warm', 'Warm'],
    'Humidity': ['Normal', 'High', 'High', 'High', 'Normal'],
    'Wind': ['Strong', 'Strong', 'Strong', 'Strong', 'Strong'],
    'Water': ['Warm', 'Warm', 'Warm', 'Cool', 'Warm'],
    'Forecast': ['Same', 'Same', 'Change', 'Change', 'Same'],
    'EnjoySport': ['Yes', 'Yes', 'No', 'Yes', 'Yes']
}
df = pd.DataFrame(data)
df.to_csv("C:/Users/admin/Downloads/enjoysport.csv", index=False)
# Step 2: Load dataset
dataset = pd.read_csv("C:/Users/admin/Downloads/enjoysport.csv")
# Step 3: FIND-S algorithm
def find_s_algorithm(dataframe):
    # Extract only positive examples
    positive_examples = dataframe[dataframe['EnjoySport'] == 'Yes'].iloc[:, :-1].values
    hypothesis = positive_examples[0].copy()
    for example in positive_examples[1:]:
        for i in range(len(hypothesis)):
            if hypothesis[i] != example[i]:
                hypothesis[i] = '?'
    return hypothesis
final_hypothesis = find_s_algorithm(dataset)
print("Final Specific Hypothesis:", final_hypothesis)
```

OUTPUT

```
(base) C:\Users\admin>python Desktop/MLPRAC/prac2.py
Final Specific Hypothesis: ['Sunny' 'Warm' '?' 'Strong' '?' '?']
```

Practical no 3

A)Fit a linear regression model on a dataset. Interpret coefficients, make predictions, and evaluate performance using metrics like R-squared and MSE

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
data = pd.read_csv("C:/Users/admin/Downloads/winequality_red.csv")
print(data.head())
data = data.dropna()
X = data[['fixed acidity', 'volatile acidity', 'citric acid']]  # update as needed
y = data['quality']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
print("Intercept:", model.intercept_)
print("Coefficients:")
for feature, coef in zip(X.columns, model.coef_):
    print(f"{feature}: {coef}")
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)
print("R-squared (R²):", r2)
```

```
(base) C:\Users\admin>python Desktop/ML/prac3.py
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  ...  density    pH  sulphates  alcohol  quality
0            7.4              0.70         0.00             1.9      0.076  ...   0.9978  3.51       0.56      9.4        5
1            7.8              0.88         0.00             2.6      0.098  ...   0.9968  3.20       0.68      9.8        5
2            7.8              0.76         0.04             2.3      0.092  ...   0.9970  3.26       0.65      9.8        5
3           11.2              0.28         0.56             1.9      0.075  ...   0.9980  3.16       0.58      9.8        6
4            7.4              0.70         0.00             1.9      0.076  ...   0.9978  3.51       0.56      9.4        5

[5 rows x 12 columns]
Intercept: 6.378972118987667
Coefficients:
fixed acidity: 0.02053646472220784
volatile acidity: -1.7035172962776297
citric acid: -0.08136606250766007
Mean Squared Error (MSE): 0.533975815696693
R-squared (R²): 0.18290584845348323
```

B) Extend linear regression to multiple features. Handle feature selection and potential multicollinearity.

```
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.feature_selection import RFE
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
import pandas as pd
import numpy as np
df = pd.read_csv("C:/Users/admin/Downloads/winequality_red.csv")
X = df.drop(columns='quality')  # target is 'quality'
y = df['quality']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
vif_data = pd.DataFrame()
vif_data['Feature'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X_scaled, i) for i in range(X.shape[1])]
print(vif_data)
model = LinearRegression()
rfe = RFE(model, n_features_to_select=5)
X_rfe = rfe.fit_transform(X_scaled, y)
selected_features = X.columns[rfe.support_]
print("Selected features:", list(selected_features))
ridge = Ridge(alpha=1.0)
ridge.fit(X_scaled, y)
```

Output

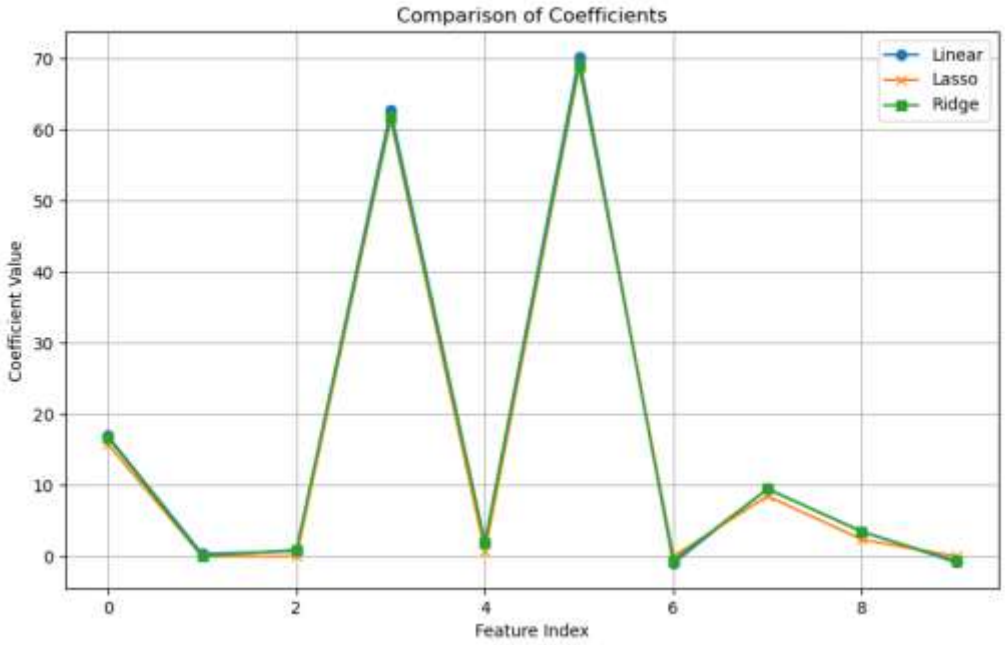
```
(base) C:\Users\admin>python Desktop/ML/prac3b.py
                Feature       VIF
0         fixed acidity  7.767512
1      volatile acidity  1.789390
2           citric acid  3.128022
3        residual sugar  1.702588
4             chlorides  1.481932
5   free sulfur dioxide  1.963019
6  total sulfur dioxide  2.186813
7               density  6.343760
8                    pH  3.329732
9             sulphates  1.429434
10              alcohol  3.031160
Selected features: ['volatile acidity', 'chlorides', 'total sulfur dioxide', 'sulphates', 'alcohol']
```

C)Implement regression variants like LASSO and Ridge on any generated dataset.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

X, y, coef = make_regression(
    n_samples=100,
    n_features=10,
    n_informative=5,
    noise=10,
    coef=True,
    random_state=42
)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
lr = LinearRegression()
lasso = Lasso(alpha=1.0)
ridge = Ridge(alpha=1.0)
lr.fit(X_train, y_train)
lasso.fit(X_train, y_train)
ridge.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
y_pred_lasso = lasso.predict(X_test)
y_pred_ridge = ridge.predict(X_test)
print("Linear Regression MSE:", mean_squared_error(y_test, y_pred_lr))
print("Lasso Regression MSE:", mean_squared_error(y_test, y_pred_lasso))
print("Ridge Regression MSE:", mean_squared_error(y_test, y_pred_ridge))
plt.figure(figsize=(10,6))
plt.plot(lr.coef_, label='Linear', marker='o')
plt.plot(lasso.coef_, label='Lasso', marker='x')
plt.plot(ridge.coef_, label='Ridge', marker='s')
plt.title("Comparison of Coefficients")
plt.xlabel("Feature Index")
plt.ylabel("Coefficient Value")
plt.legend()
plt.grid(True)
plt.show()
```

Output

```
(base) C:\Users\admin>python Desktop/ML/prac3c.py
Linear Regression MSE: 126.39524754125014
Lasso Regression MSE: 154.83952646022993
Ridge Regression MSE: 141.31120885254484
```



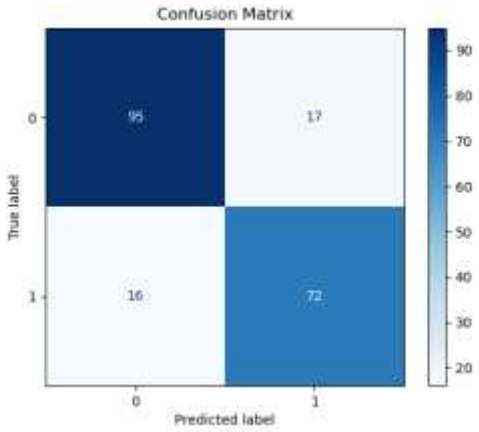
Practical no 4

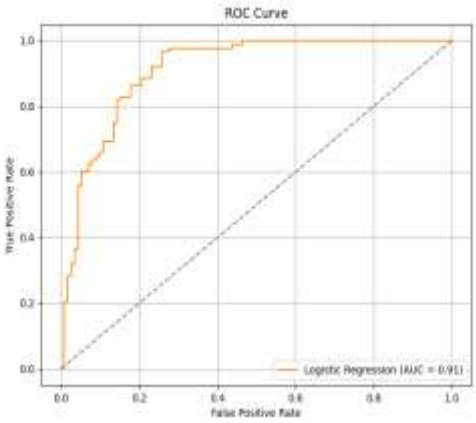
A) Perform binary classification using logistic regression. Calculate accuracy, precision, recall, and understand the ROC curve.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score,
    roc_curve, roc_auc_score, confusion_matrix
)
from sklearn.model_selection import train_test_split
# Create binary classification dataset
X, y = make_classification(
    n_samples=1000,
    n_features=10,
    n_informative=5,
    n_redundant=2,
    n_classes=2,
    random_state=42
)

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict probabilities and classes
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]  # probability of class 1
# Accuracy, Precision, Recall
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print(f"Accuracy : {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall   : {recall:.2f}")
# Compute ROC curve and AUC
```

```
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc_score = roc_auc_score(y_test, y_prob)

# Plot ROC
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {auc_score:.2f})', color='darkorange')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')  # baseline
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.grid(True)
plt.show()
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_estimator(model, X_test, y_test, cmap='Blues')
plt.title("Confusion Matrix")
plt.show()
```

Output





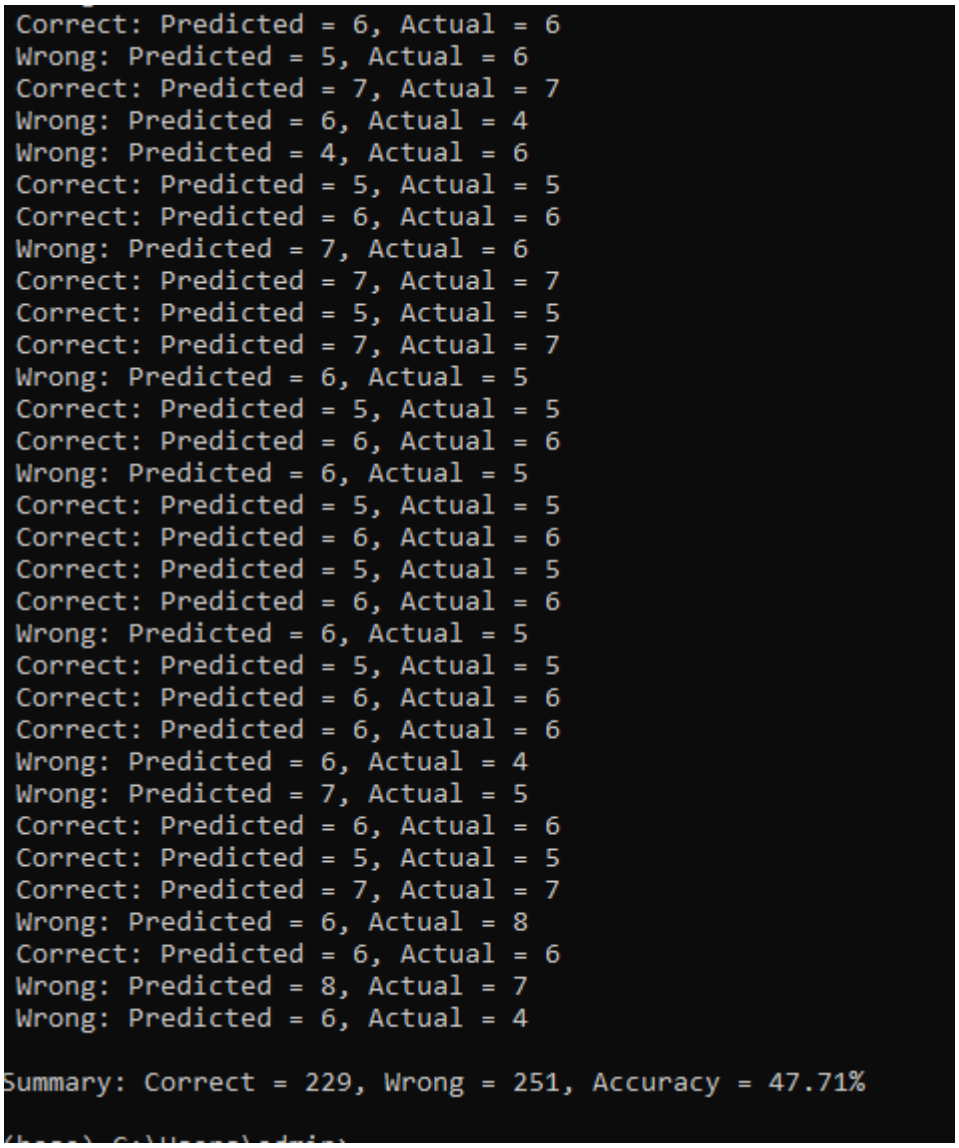

```
(base) C:\Users\admin>python Desktop/ML/prac4.py
Accuracy : 0.83
Precision: 0.81
Recall   : 0.82
```

B) Implement and demonstrate k-nearest Neighbor algorithm. Read the training data from a .CSV file and build the model to classify a test sample. Print both correct and wrong predictions.

```
import csv
import math
from collections import Counter
from sklearn.model_selection import train_test_split

# Function to read CSV data
def read_csv(filename):
    data = []
    with open(filename, 'r') as file:
        csv_reader = csv.reader(file)
        header = next(csv_reader)  # skip header
        for row in csv_reader:
            features = list(map(float, row[:-1]))  # convert features to float
            label = row[-1]
            data.append((features, label))
    return data

# Euclidean distance
def euclidean_distance(a, b):
    return math.sqrt(sum((x - y) ** 2 for x, y in zip(a, b)))

# k-NN algorithm
def knn_predict(train_data, test_sample, k=3):
    distances = []
    for features, label in train_data:
        distance = euclidean_distance(features, test_sample)
        distances.append((distance, label))
    distances.sort(key=lambda x: x[0])
    k_nearest = [label for _, label in distances[:k]]
    most_common = Counter(k_nearest).most_common(1)
    return most_common[0][0]

# Evaluation function
def evaluate_knn(train_data, test_data, k=3):
    correct = 0
    wrong = 0
    for features, actual_label in test_data:
        predicted = knn_predict(train_data, features, k)
        if predicted == actual_label:
```

```
            correct += 1
            print(f" Correct: Predicted = {predicted}, Actual = {actual_label}")
        else:
            wrong += 1
            print(f" Wrong: Predicted = {predicted}, Actual = {actual_label}")
    print(f"\nSummary: Correct = {correct}, Wrong = {wrong}, Accuracy = {correct / (correct + wrong) *
100:.2f}%")

# Main execution
if __name__ == "__main__":
    filename = "C:/Users/admin/Downloads/winequality_red.csv"  # Replace with your file
    dataset = read_csv(filename)
    train_set, test_set = train_test_split(dataset, test_size=0.3, random_state=42)
    evaluate_knn(train_set, test_set, k=3)
```

Output-

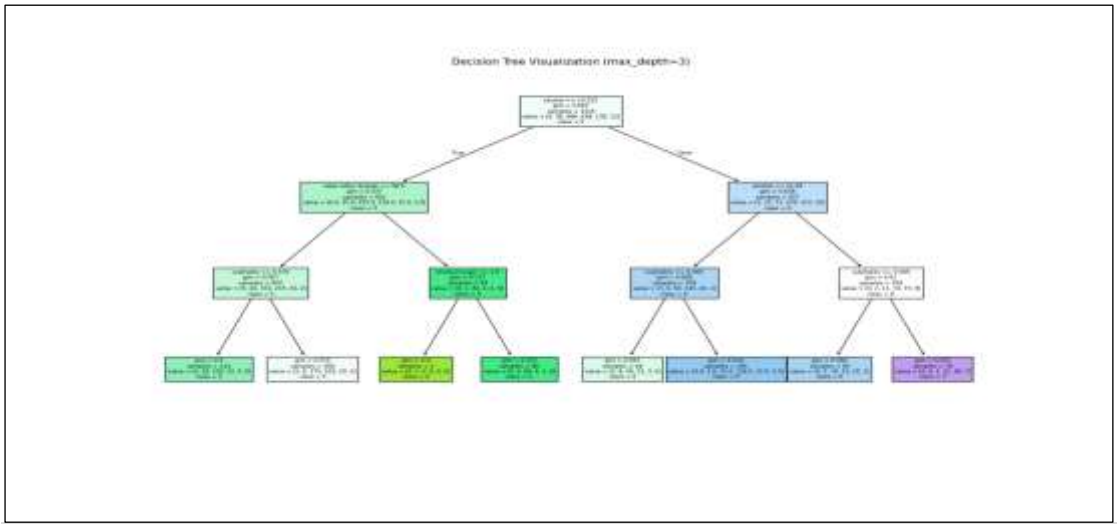


C)Build a decision tree classifier or regressor. Control hyperparameters like tree depth to avoid overfitting. Visualize the tree.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
data = pd.read_csv("C:/Users/admin/Downloads/winequality_red.csv")
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
clf = DecisionTreeClassifier(max_depth=3, random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred, zero_division=0))
plt.figure(figsize=(20, 10))
plot_tree(clf, feature_names=X.columns, class_names=[str(c) for c in clf.classes_], filled=True)
plt.title("Decision Tree Visualization (max_depth=3)")
plt.show()
```

Output

```
(base) C:\Users\admin>python Desktop/ML/prac4c.py

Accuracy: 0.5166666666666667

Classification Report:
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00        17
           5       0.53      0.88      0.66       195
           6       0.52      0.30      0.38       200
           7       0.42      0.25      0.31        61
           8       0.00      0.00      0.00         6

    accuracy                           0.52       480
   macro avg       0.24      0.24      0.23       480
weighted avg       0.48      0.52      0.47       480

(base) C:\Users\admin>
```

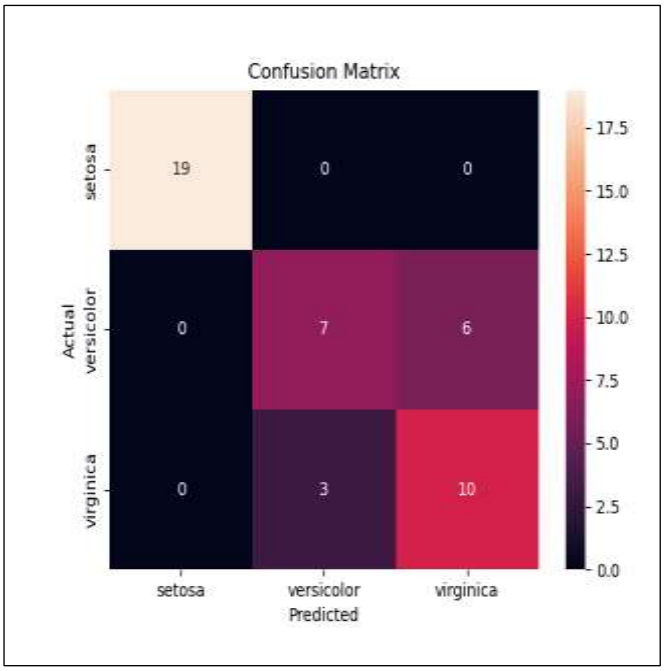


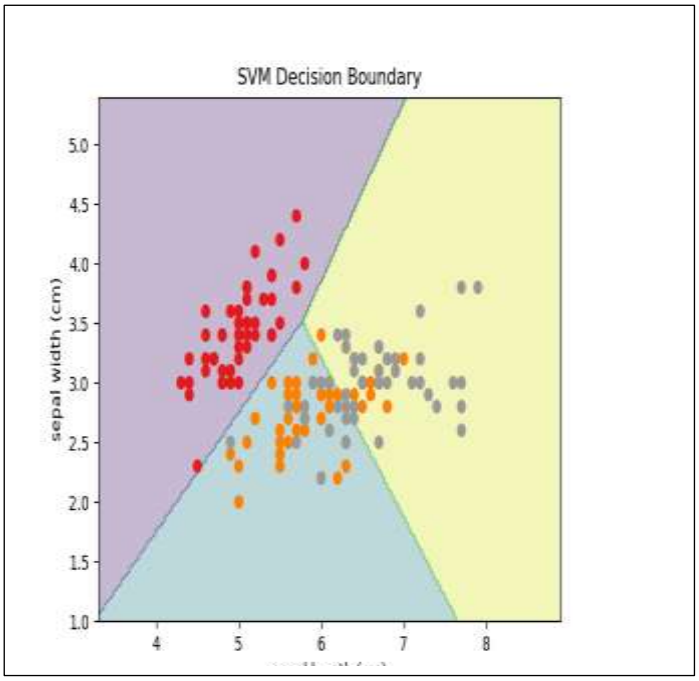
D)Implement a Support Vector Machine for any relevant dataset.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
iris = datasets.load_iris()
X = iris.data[:, :2]  # use only first 2 features for visualization
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
model = SVC(kernel='linear', C=1.0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
def plot_decision_boundary(X, y, model):
    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.linspace(x_min, x_max, 200),
                         np.linspace(y_min, y_max, 200))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.3)
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1)
    plt.xlabel(iris.feature_names[0])
    plt.ylabel(iris.feature_names[1])
    plt.title("SVM Decision Boundary")
    plt.show()
plot_decision_boundary(X, y, model)
```

Output

```
(base) C:\Users\admin>python Desktop/ML/prac4d.py
Accuracy: 0.8
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.70      0.54      0.61        13
           2       0.62      0.77      0.69        13

    accuracy                           0.80        45
   macro avg       0.78      0.77      0.77        45
weighted avg       0.81      0.80      0.80        45
```



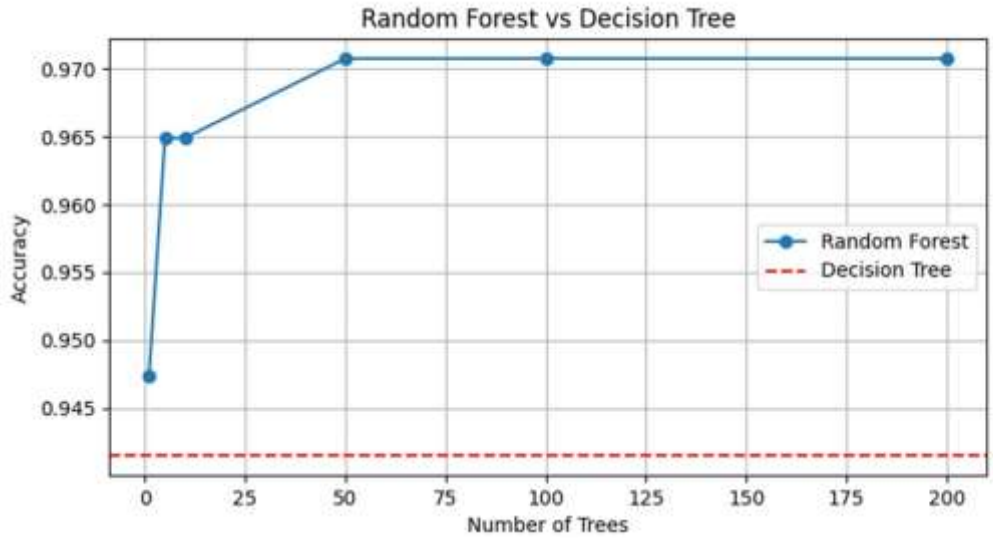


E) Train a random forest ensemble. Experiment with the number of trees and feature sampling. Compare performance to a single decision tree.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
data = load_breast_cancer()
X = data.data
y = data.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)
tree_preds = tree_clf.predict(X_test)
tree_acc = accuracy_score(y_test, tree_preds)
n_trees_list = [1, 5, 10, 50, 100, 200]
results = []
for n_trees in n_trees_list:
    rf_clf = RandomForestClassifier(n_estimators=n_trees, max_features='sqrt', random_state=42)
    rf_clf.fit(X_train, y_train)
    rf_preds = rf_clf.predict(X_test)
    rf_acc = accuracy_score(y_test, rf_preds)
    results.append((n_trees, rf_acc))
print(f"Single Decision Tree Accuracy: {tree_acc:.4f}")
print("Random Forest Accuracies:")
for n_trees, acc in results:
    print(f" Trees: {n_trees}, Accuracy: {acc:.4f}")
n_vals, acc_vals = zip(*results)
plt.figure(figsize=(8, 4))
plt.plot(n_vals, acc_vals, marker='o', label='Random Forest')
plt.axhline(y=tree_acc, color='r', linestyle='--', label='Decision Tree')
plt.xlabel("Number of Trees")
plt.ylabel("Accuracy")
plt.title("Random Forest vs Decision Tree")
plt.legend()
plt.grid(True)
plt.show()
```

Output

```
Single Decision Tree Accuracy: 0.9415
Random Forest Accuracies:
 Trees: 1, Accuracy: 0.9474
 Trees: 5, Accuracy: 0.9649
 Trees: 10, Accuracy: 0.9649
 Trees: 50, Accuracy: 0.9708
 Trees: 100, Accuracy: 0.9708
 Trees: 200, Accuracy: 0.9708
```

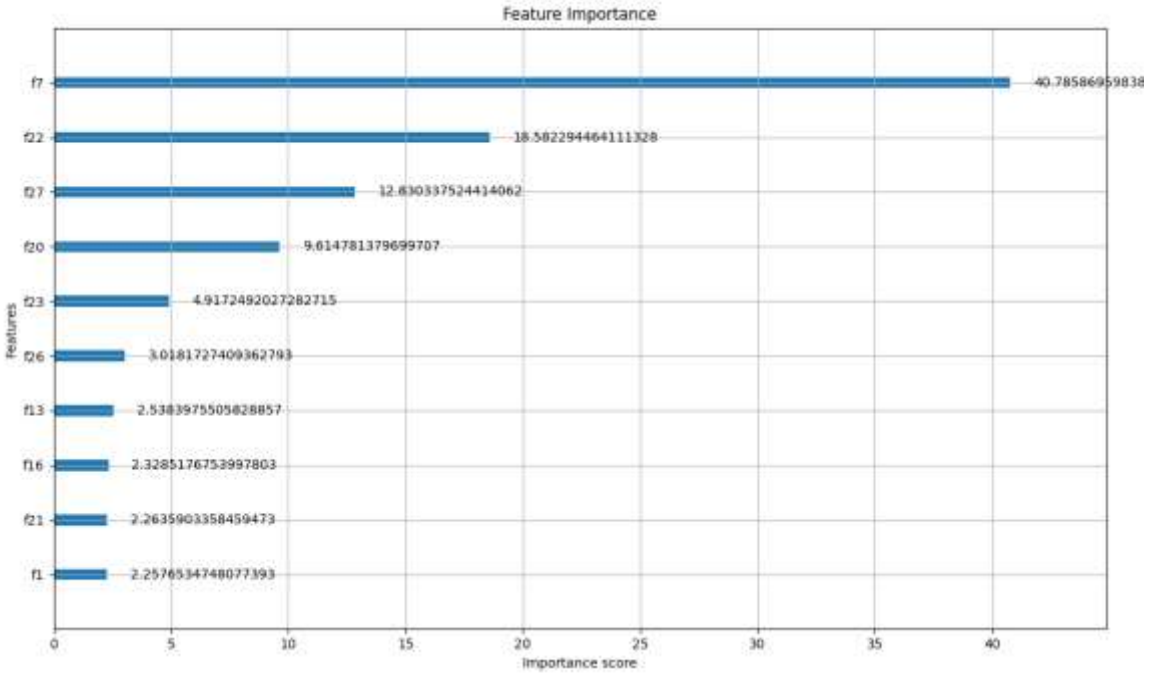


F) Implement a gradient boos ng machine (e.g., XGBoost). Tune hyperparameters and explore feature importance.

```
import xgboost as xgb
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
data = load_breast_cancer()
X = data.data
y = data.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
xgb_model = xgb.XGBClassifier(eval_metric='logloss', use_label_encoder=False, random_state=42)
xgb_model.fit(X_train, y_train)
y_pred = xgb_model.predict(X_test)
print("Basic XGBoost Accuracy:", accuracy_score(y_test, y_pred))
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [3, 5],
    'learning_rate': [0.01, 0.1],
    'subsample': [0.8, 1.0]
}
grid = GridSearchCV(
    xgb.XGBClassifier(eval_metric='logloss', use_label_encoder=False, random_state=42),
    param_grid,
    cv=3,
    scoring='accuracy',
    verbose=1
)
```

```
grid.fit(X_train, y_train)
best_model = grid.best_estimator_
y_best_pred = best_model.predict(X_test)
print("\nBest Parameters:", grid.best_params_)
print("Tuned XGBoost Accuracy:", accuracy_score(y_test, y_best_pred))
print("\nClassification Report:\n", classification_report(y_test, y_best_pred))
xgb.plot_importance(best_model, importance_type='gain', title='Feature Importance',
max_num_features=10)
plt.tight_layout()
plt.show()
```

```
Best Parameters: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100, 'subsample': 0.8}
Tuned XGBoost Accuracy: 0.9649122807017544

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        63
           1       0.97      0.97      0.97       108

    accuracy                           0.96       171
   macro avg       0.96      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171
```



Practical no 5

Implement and demonstrate the working of a Naive Bayesian classifier using a sample data set. Build the model to classify a test sample.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import CategoricalNB
data = {
    'Outlook': [
        'Sunny', 'Sunny', 'Overcast', 'Rain', 'Rain', 'Rain', 'Overcast',
        'Sunny', 'Sunny', 'Rain', 'Sunny', 'Overcast', 'Overcast', 'Rain'
    ],
    'Temp': [
        'Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool',
        'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild'
    ],
    'Humidity': [
        'High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal',
        'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'High'
    ],
    'Wind': [
        'Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong',
        'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Strong'
    ],
    'Play': [
        'No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes',
        'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No'
    ]
}
```

```
df = pd.DataFrame(data)
encoders = {}
for col in df.columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    encoders[col] = le  # Store encoder for future decoding
X = df[['Outlook', 'Temp', 'Humidity', 'Wind']]
y = df['Play']
model = CategoricalNB()
model.fit(X, y)
test_sample_dict = {
    'Outlook': encoders['Outlook'].transform(['Sunny'])[0],
    'Temp': encoders['Temp'].transform(['Cool'])[0],
    'Humidity': encoders['Humidity'].transform(['High'])[0],
    'Wind': encoders['Wind'].transform(['Strong'])[0]
}
test_sample = pd.DataFrame([test_sample_dict])
prediction = model.predict(test_sample)
decoded_prediction = encoders['Play'].inverse_transform(prediction)
print("Prediction:", decoded_prediction[0])
```

Output

```
base) C:\Users\admin>python Desktop/MLPRAC/prac5a.py
rediction: No
```

B) Implement Hidden Markov Models using hmmlearn

```
import numpy as np
from hmmlearn import hmm
states = ["Rainy", "Sunny"]
n_states = len(states)
observations = ["Walk", "Shop", "Clean"]
n_observations = len(observations)
transition_prob = np.array([
   [0.7, 0.3],  # Rainy -> (Rainy, Sunny)
   [0.4, 0.6]   # Sunny -> (Rainy, Sunny)
])
emission_prob = np.array([
   [0.1, 0.4, 0.5],  # Rainy -> (Walk, Shop, Clean)
   [0.6, 0.3, 0.1]   # Sunny -> (Walk, Shop, Clean)
])
start_prob = np.array([0.6, 0.4])
model = hmm.CategoricalHMM(n_components=n_states, n_iter=100, init_params="")
model.startprob_ = start_prob
model.transmat_ = transition_prob
model.emissionprob_ = emission_prob
obs_sequence = np.array([[0], [1], [2]])  # Walk -> Shop -> Clean
logprob, hidden_states = model.decode(obs_sequence, algorithm="viterbi")
print("Observation sequence:", [observations[i[0]] for i in obs_sequence])
print("Most likely hidden states:", [states[i] for i in hidden_states])
```

Output

```
(base) C:\Users\admin>python Desktop/MLPRAC/prac5b.py
Observation sequence: ['Walk', 'Shop', 'Clean']
Most likely hidden states: ['Sunny', 'Rainy', 'Rainy']
```

Practical no 6

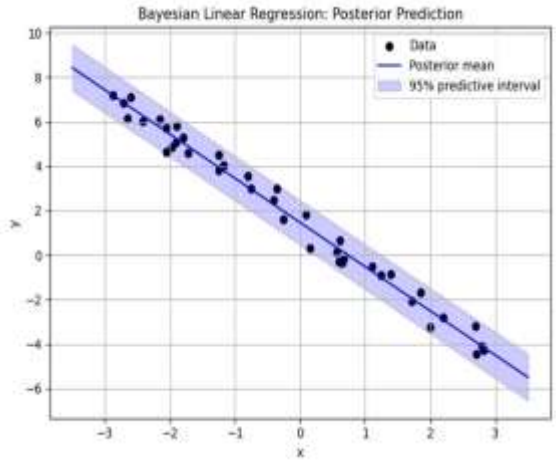
A) Implement Bayesian Linear Regression to explore prior and posterior distribution

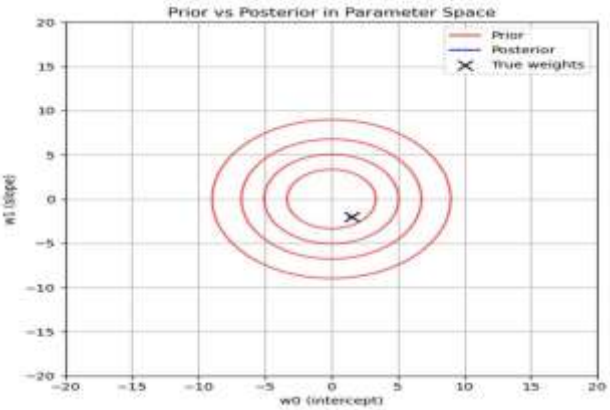
```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv
np.random.seed(42)
n = 40
x = np.random.uniform(-3, 3, size=n)
X = np.column_stack([np.ones(n), x])  # Add intercept term
true_w = np.array([1.5, -2.0])
sigma = 0.5  # Known noise standard deviation
y = X @ true_w + np.random.normal(0, sigma, size=n)
m0 = np.array([0.0, 0.0])  # Prior mean
S0 = np.diag([5.0**2, 5.0**2])  # Prior covariance (broad)
SN = inv(inv(S0) + (X.T @ X) / (sigma**2))
mN = SN @ (inv(S0) @ m0 + (X.T @ y) / (sigma**2))
print("True weights:", true_w)
print("Prior mean:", m0)
print("Posterior mean:", mN)
print("Posterior covariance:\n", SN)
x_grid = np.linspace(-3.5, 3.5, 200)
Xg = np.column_stack([np.ones_like(x_grid), x_grid])
pred_mean = Xg @ mN
pred_var = sigma**2 + np.einsum('ij,jk,ik->i', Xg, SN, Xg)  # Predictive variance
pred_sd = np.sqrt(pred_var)
plt.figure(figsize=(7, 5))
plt.scatter(x, y, label="Data", color="black")
plt.plot(x_grid, pred_mean, label="Posterior mean", color="blue")
plt.fill_between(
    x_grid,
    pred_mean - 1.96 * pred_sd,
    pred_mean + 1.96 * pred_sd,
    color="blue", alpha=0.2,
    label="95% predictive interval"
)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Bayesian Linear Regression: Posterior Prediction")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
def gaussian_contour(ax, mean, cov, color, label):
    w0 = np.linspace(mean[0] - 4 * np.sqrt(cov[0, 0]), mean[0] + 4 * np.sqrt(cov[0, 0]), 200)
    w1 = np.linspace(mean[1] - 4 * np.sqrt(cov[1, 1]), mean[1] + 4 * np.sqrt(cov[1, 1]), 200)
    W0, W1 = np.meshgrid(w0, w1)
    W = np.stack([W0 - mean[0], W1 - mean[1]], axis=-1)
    icov = inv(cov)
    M = (W[..., 0]*icov[0, 0] + W[..., 1]*icov[1, 0])*W[..., 0] + \
        (W[..., 0]*icov[0, 1] + W[..., 1]*icov[1, 1])*W[..., 1]
    Z = np.exp(-0.5 * M)
    ax.contour(W0, W1, Z, levels=5, colors=color, alpha=0.7)
    ax.plot([], [], color=color, label=label)  # For legend
plt.figure(figsize=(6, 6))
ax = plt.gca()
gaussian_contour(ax, m0, S0, "red", "Prior")
gaussian_contour(ax, mN, SN, "blue", "Posterior")
ax.scatter(*true_w, marker="x", s=100, color="black", label="True weights")
ax.set_xlabel("w0 (intercept)")
ax.set_ylabel("w1 (slope)")
ax.set_title("Prior vs Posterior in Parameter Space")
ax.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Output

```
(base) C:\Users\admin>python Desktop/MLPRAC/ml6a.py
True weights: [ 1.5 -2. ]
Prior mean: [0. 0.]
Posterior mean: [ 1.4633892  -1.98871086]
Posterior covariance:
 [[0.00638859 0.00053425]
 [0.00053425 0.00203646]]
```

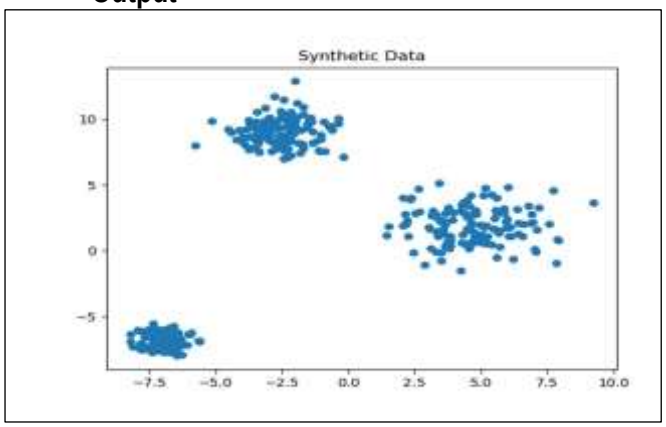


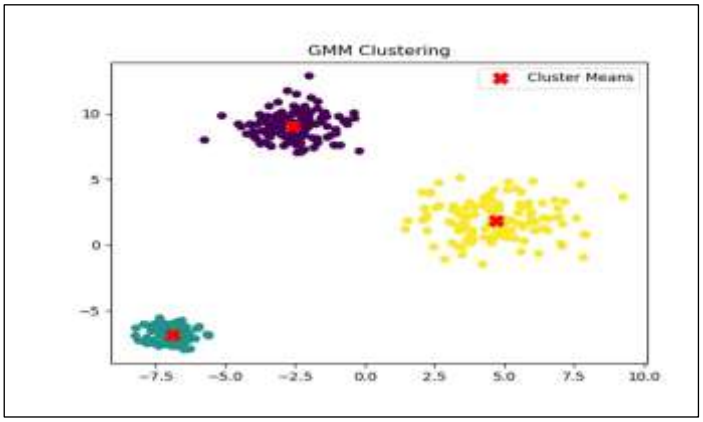


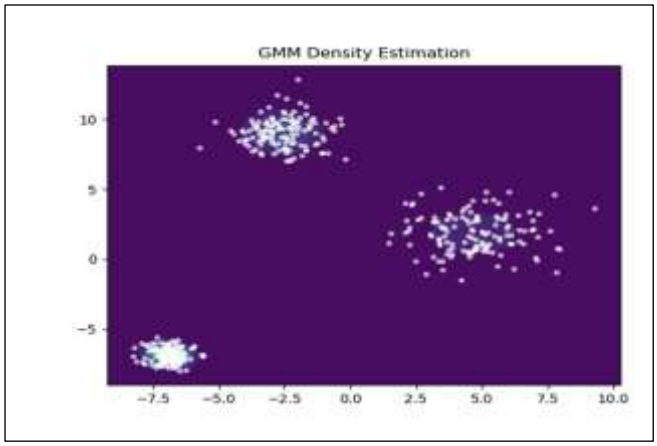
B) Implement Gaussian Mixture Models for density es ma on and unsupervised clustering

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture
from sklearn.datasets import make_blobs
X, y_true = make_blobs(
    n_samples=400,
    centers=3,
    cluster_std=[1.0, 1.5, 0.5],
    random_state=42
)
plt.scatter(X[:, 0], X[:, 1], s=30, cmap="viridis")
plt.title("Synthetic Data")
plt.show()
gmm = GaussianMixture(
    n_components=3,
    covariance_type='full',
    random_state=42
)
gmm.fit(X)
labels = gmm.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=30, cmap="viridis")
plt.scatter(
    gmm.means_[:, 0], gmm.means_[:, 1],
    c="red", s=100, marker="X", label="Cluster Means"
)
plt.title("GMM Clustering")
plt.legend()
plt.show()
x = np.linspace(np.min(X[:, 0]) - 1, np.max(X[:, 0]) + 1, 200)
y = np.linspace(np.min(X[:, 1]) - 1, np.max(X[:, 1]) + 1, 200)
Xgrid, Ygrid = np.meshgrid(x, y)
XX = np.column_stack([Xgrid.ravel(), Ygrid.ravel()])
Z = np.exp(gmm.score_samples(XX))
Z = Z.reshape(Xgrid.shape)
plt.contourf(Xgrid, Ygrid, Z, levels=20, cmap="viridis")
plt.scatter(X[:, 0], X[:, 1], s=10, c="white", alpha=0.6)
plt.title("GMM Density Estimation")
plt.show()
print("Cluster means:\n", gmm.means_)
print("\nCovariances:\n", gmm.covariances_)
print("\nWeights (mixture proportions):\n", gmm.weights_)
```

Output








```
Cluster means:
 [[-2.5878614   9.04995213]
 [-6.89813995 -6.8576282 ]
 [ 4.68804053  1.87582721]]

Covariances:
 [[[ 0.95598561  0.02946797]
  [ 0.02946797  1.03033493]]

 [[ 0.23745661 -0.02287196]
  [-0.02287196  0.25589641]]

 [[ 2.22535416 -0.03180548]
  [-0.03180548  1.8501235 ]]]

Weights (mixture proportions):
 [0.33499948 0.3325     0.33250052]
```

Practical no 7

A) Implement cross-validation techniques (k-fold, stratified, etc.) for robust model Evaluation

```
from sklearn.datasets import load_iris
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import numpy as np
data = load_iris()
X = data.data
y = data.target
model = RandomForestClassifier(random_state=42)
kf = KFold(n_splits=5, shuffle=True, random_state=42)
scores = cross_val_score(model, X, y, cv=kf, scoring='accuracy')
print("K-Fold CV Accuracy Scores:", scores)
print("Mean K-Fold CV Accuracy:", np.mean(scores))
# Stratified K-Fold ensures class proportions are preserved
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
stratified_scores = cross_val_score(model, X, y, cv=skf, scoring='accuracy')
print("\nStratified K-Fold CV Accuracy Scores:", stratified_scores)
print("Mean Stratified K-Fold CV Accuracy:", np.mean(stratified_scores))
from sklearn.model_selec!tion import LeaveOneOut
loo = LeaveOneOut()
loo_scores = cross_val_score(model, X, y, cv=loo, scoring='accuracy')
print("\nLeave-One-Out CV Accuracy Mean:", np.mean(loo_scores))
```

Output

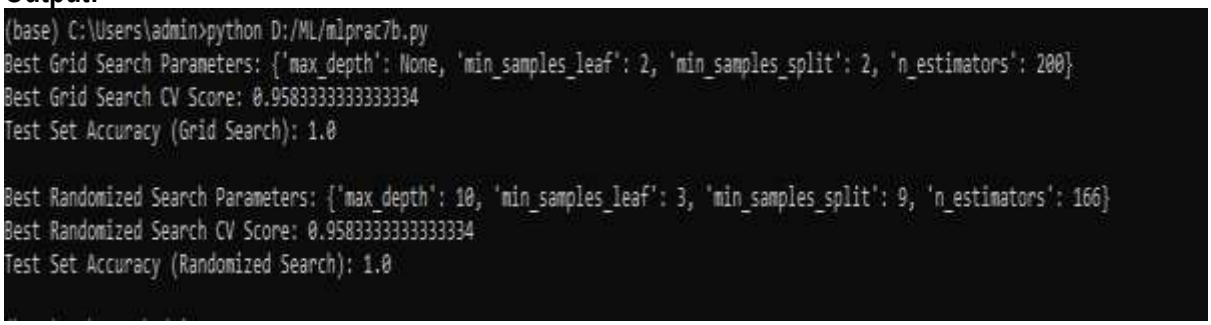
```
(base) C:\Users\admin>python D:/ML/mlprac7a.py
K-Fold CV Accuracy Scores: [1.         0.96666667 0.93333333 0.93333333 0.96666667]
Mean K-Fold CV Accuracy: 0.9600000000000002

Stratified K-Fold CV Accuracy Scores: [0.96666667 0.96666667 0.93333333 0.96666667 0.9       ]
Mean Stratified K-Fold CV Accuracy: 0.9466666666666667

Leave-One-Out CV Accuracy Mean: 0.9533333333333334

(base) C:\Users\admin>
```

B) Systematically explore combinations of hyperparameters to optimize model performance. (Use grid and randomized search)

```
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV, train_test_split
from sklearn.metrics import accuracy_score
from scipy.stats import randint
import numpy as np
data = load_iris()
X = data.data
y = data.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
model = RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}
grid_search = GridSearchCV(
    estimator=model,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=1  # You can increase this (e.g., to 2 or 4) if stable
)
grid_search.fit(X_train, y_train)
print("Best Grid Search Parameters:", grid_search.best_params_)
print("Best Grid Search CV Score:", grid_search.best_score_)
y_pred_grid = grid_search.best_estimator_.predict(X_test)
print("Test Set Accuracy (Grid Search):", accuracy_score(y_test, y_pred_grid))
param_dist = {
    'n_estimators': randint(50, 300),
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': randint(2, 10),
    'min_samples_leaf': randint(1, 5)
}
random_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_dist,
    n_iter=20,
    cv=5,
    scoring='accuracy',
    n_jobs=1,
    random_state=42
)
random_search.fit(X_train, y_train)
```

```
print("\nBest Randomized Search Parameters:", random_search.best_params_)
print("Best Randomized Search CV Score:", random_search.best_score_)
y_pred_random = random_search.best_estimator_.predict(X_test)
print("Test Set Accuracy (Randomized Search):", accuracy_score(y_test, y_pred_random))
```

Output:



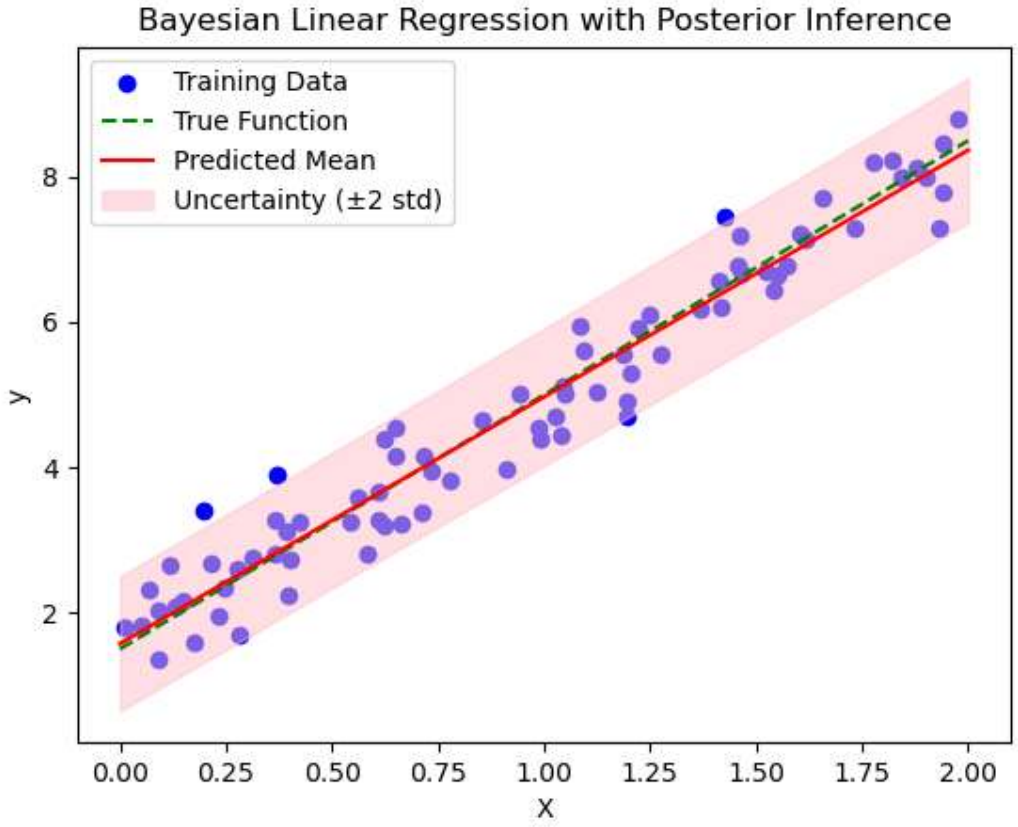
Practical no 8

Implement Bayesian Learning using inferences

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import BayesianRidge
from sklearn.model_selection import train_test_split
np.random.seed(42)
X = 2 * np.random.rand(100, 1)
true_coef = 3.5
true_intercept = 1.5
y = true_coef * X.flatten() + true_intercept + np.random.randn(100) * 0.5
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
from sklearn.linear_model import BayesianRidge
model = BayesianRidge()
model.fit(X_train, y_train)
y_pred, y_std = model.predict(X_test, return_std=True)
plt.scatter(X_train, y_train, color='blue', label='Training Data')
X_line = np.linspace(0, 2, 100).reshape(-1, 1)
y_true = true_coef * X_line.flatten() + true_intercept
plt.plot(X_line, y_true, 'g--', label='True Function')
y_mean, y_std_line = model.predict(X_line, return_std=True)
plt.plot(X_line, y_mean, 'r-', label='Predicted Mean')
plt.fill_between(
    X_line.flatten(),
    y_mean - 2 * y_std_line,
    y_mean + 2 * y_std_line,
    color='pink',
    alpha=0.5,
    label='Uncertainty (±2 std)'
)
plt.legend()
```

```
plt.xlabel('X')
plt.ylabel('y')
plt.title('Bayesian Linear Regression with Posterior Inference')
plt.show()
print("Learned Coefficients Mean:", model.coef_)
print("Learned Intercept:", model.intercept_)
print("Alpha (Precision of noise):", model.alpha_)
print("Lambda (Precision of weights prior):", model.lambda_)
```

Output




```
(base) C:\Users\admin>python D:/ML/mlprac8.py
Learned Coefficients Mean: [3.39733606]
Learned Intercept: 1.5736340376397093
Alpha (Precision of noise): 4.6597831289883205
Lambda (Precision of weights prior): 0.08658179117729597
```

Security Operations Center

JNAN VIKAS MANDAL'S

Mohanlal Raichand Mehta College of Commerce
Diwali Maa College of Science
Amritlal Raichand Mehta College of Arts
Dr. R.T. Doshi College of Computer Science
NAAC Re-Accredited Grade 'A+' (CGPA : 3.31) (3rd Cycle)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that Sonali Rupchand Thakur bearing seat no. 1313626 has done the project work/journal work in the subject of Security Operations Center of semester 3 Practical Examination during the academic year 2025-26 under the guidance of **Dr.Sanjivani Nalkar** being the partial requirement for the fulfilment of the curriculum of Degree in Master of Science in Information Technology under University of Mumbai.

Place: Airoli Date:

Sign of Subject in- Charge Sign of External Examiner

Sign of Coordinator

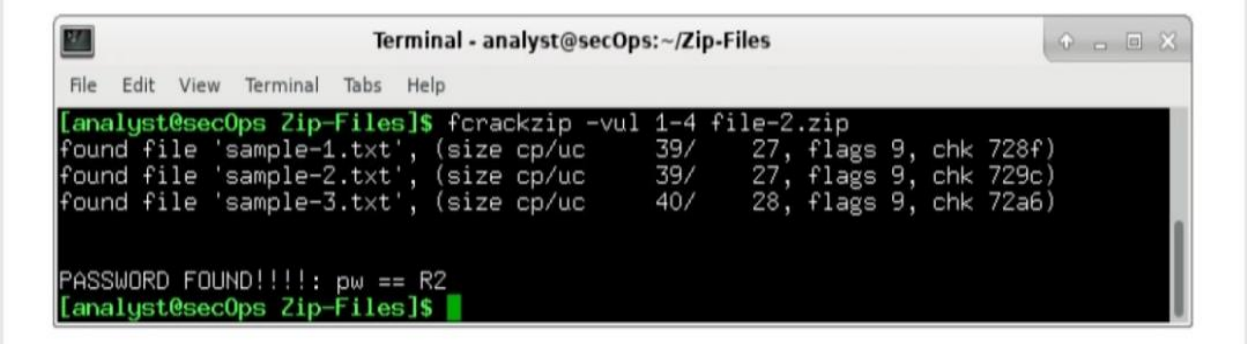
TABLE OF CONTENTS

Practical No	Practicals	Date	Sign
1.	a. Encrypting and Decrypting Data Using a Hacker Tool b. Encrypting and Decrypting Data Using OpenSSL c. Hashing a Text File with OpenSSL and Verifying Hashes		
2.	a. Examining Telnet and SSH in Wireshark		
3.	a. Demonstrate the use of Snort and Firewall Rules b. Demonstrate Extract an Executable from a PCAP c. Demonstrate a practical for Exploring DNS Traffic		
4.	b. Exploring Processes, Threads, Handles, and Windows Registry		
5.	Perform a practical to Attack on a mySQL Database by using PCAP File.		
6.	Create your own syslog Server		
7.	Configure your Linux system to send syslog messages to a syslog server and read them		
8.	Install and Run Splunk on Linux		
9.	Install and Configure ELK on Linux		

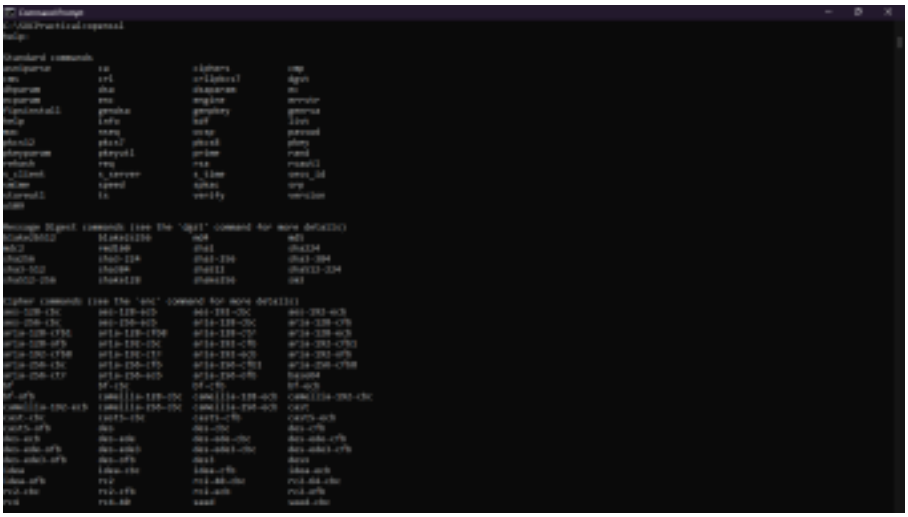
Practical 1

a. Encrypting and Decrypting Data Using a Hacker Tool

Part 1: Create and Encrypt Files
cd ~
mkdir Zip-Files
cd Zip-Files
echo This is a sample text file > sample-1.txt
echo This is a sample text file > sample-2.txt
echo This is a sample text file > sample-3.txt ls
Step 2: Zip and encrypt the text files.
zip –e file-1.zip sample*
enter a one-character password = B
zip –e file-2.zip sample*
enter a one-character password = R2
zip –e file-3.zip sample*
enter a one-character password = 0B1
zip –e file-4.zip sample*
enter a one-character password = Y0Da
zip –e file-5.zip sample*
enter a one-character password = C-3P0 ls -l f*
Attempt to open a zip using an incorrect password
unzip file-1.zip enter wrong password
Part 2: Recover Encrypted Zip File Passwords
Step 1: Introduction to fcrackzip
fcrackzip –h
Step 2: Recovering Passwords using fcrackzip
fcrackzip -vul 1-4 file-1.zip fcrackzip –vul 1-4 file-2.zip fcrackzip –vul 1-4 file-3.zip
fcrackzip –vul 1-4 file-4.zip fcrackzip –vul 1-5 file-5.zip

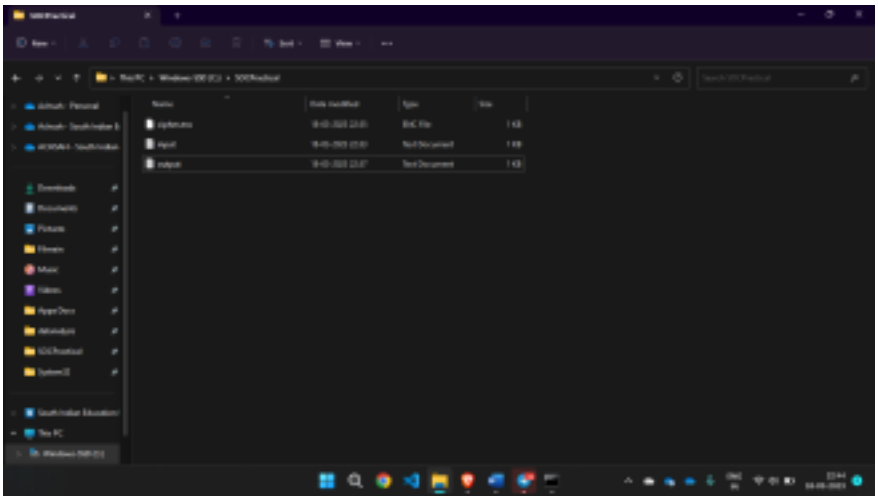


b. Encrypting and Decrypting Data Using OpenSSL



Create a text file named input.txt and add some text or data in it. Encrypt the file with file command. After encrypting the file, a .enc file will be generated. And decrypt it into another file and name it output.txt.

```
C:\SOCPractical>openssl enc -aes-256-ecb -a -salt -in input.txt -out cipher.enc
enter AES-256-ECB encryption password:
Verifying - enter AES-256-ECB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

C:\SOCPractical>openssl enc -d -aes-256-ecb -a -salt -in cipher.enc -out output.txt
enter AES-256-ECB decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

C:\SOCPractical>_
```



c. Hashing a Text File with OpenSSL and Verifying Hashes

```
Part 1: Hashing a Text File with OpenSSL

cd /home/sec_admin/lab.support.files/
cat letter_to_grandma.txt
openssl sha256 letter_to_grandma.txt
nano letter_to_grandma.txt
change the first sentence from 'Hi Grandma' to 'Hi Grandpa'
After the change = <CONTROL+X> , 'Y' , <Enter>
openssl sha256 letter_to_grandma.txt

openssl sha512 letter_to_grandma.txt

Part 2: Verifying Hashes

cat sample.img_SHA256.sig
sha256sum sample.img
```

```
[sec_admin@secOps lab.support.files]$ cat sample.img_SHA256.sig
c56c4724c26eb0157963c0d62b76422116be31804a39c82fd44ddf0ca5013e6a
```

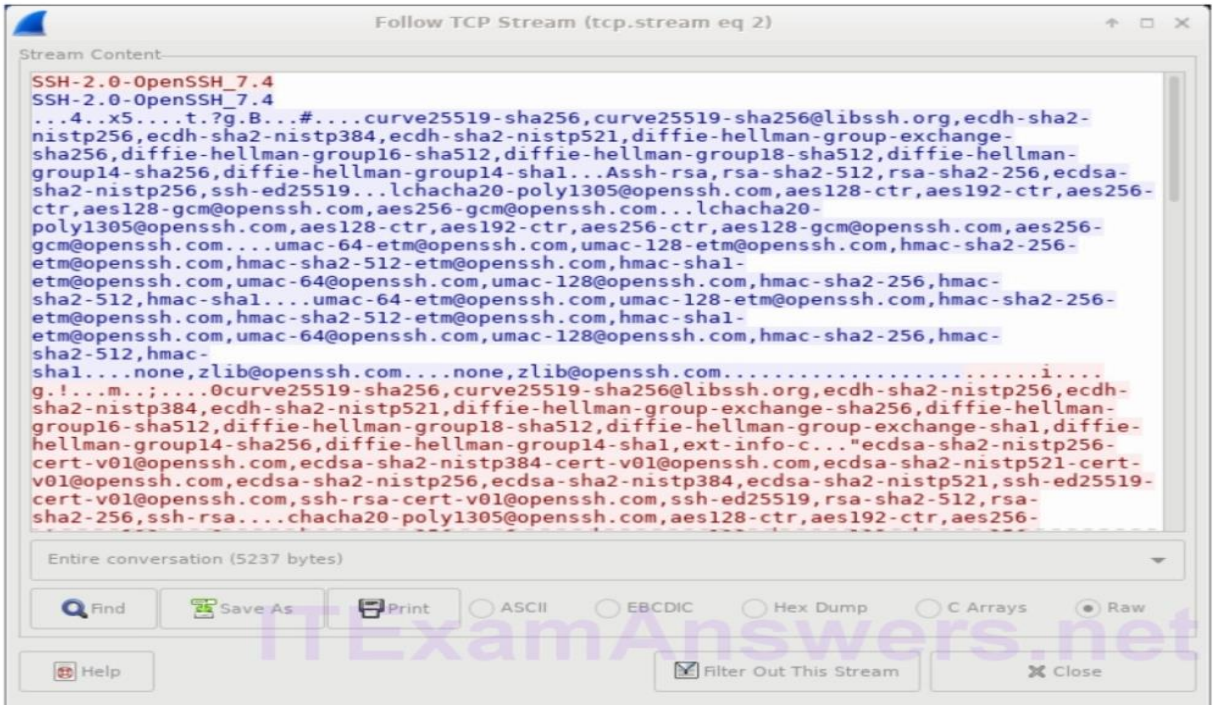
```
[sec_admin@secOps lab.support.files]$ sha256sum sample.img
c56c4724c26eb0157963c0d62b76422116be31804a39c82fd44ddf0ca5013e6a
```

Practical 2

a. Examining Telnet and SSH in Wireshark

Examining Telnet and SSH in Wireshark
Part 1: Examining a Telnet Session with Wireshark
a. Start the CyberOps Workstation VM and log in with username analyst and password cyberops.
b. Open a terminal window and start Wireshark. Press OK to continue after reading the warning message.
c. Start a Wireshark capture on the Loopback: lo interface.
Step 1: Capture data.
a. Start the CyberOps Workstation VM and log in with username analyst and password cyberops.
b. Open a terminal window and start Wireshark. Press OK to continue after reading the warning message.
c. Start a Wireshark capture on the Loopback: lo interface.
d. Open another terminal window. Start a Telnet session to the localhost. Enter username analyst and password cyberops when prompted. Note that it may take several minutes for the "connected to localhost" and login prompt to appear.
write the following cmds:
[analyst@secOps ~]$ sudo systemctl start telnet.socket
[analyst@secOps ~]$ telnet localhost
e. Stop the Wireshark capture after you have provided the user credentials.
Step 2: Examine the Telnet session.
a. Apply a filter that only displays Telnet-related traffic. Enter Telnet in the filter field and click Apply.
b. Right-click one of the Telnet lines in the Packet list section of Wireshark, and from the drop-down list, select Follow TCP Stream.Step 1: Capture data.
a. Start the CyberOps Workstation VM and log in with username analyst and password cyberops.
b. Open a terminal window and start Wireshark. Press OK to continue after reading the warning message.
[analyst@secOps analyst]$ sudo wireshark-gtk
[sudo] password for analyst: cyberops
c. Start a Wireshark capture on the Loopback: lo interface.
d. Open another terminal window. Start a Telnet session to the localhost. Enter username analyst and password cyberops when prompted. Note that it may take several minutes for the "connected to localhost" and login prompt to appear.
[analyst@secOps ~]$ telnet localhost
secOps login: analyst
Password:
Last login: Fri Apr 28 10:50:52 from localhost.localdomain
[analyst@secOps ~]$

e. Stop the Wireshark capture after you have provided the user credentials.
Step 2: Examine the Telnet session.
a. Apply a filter that only displays Telnet-related traffic. Enter Telnet in the filter field and click Apply.
b. Right-click one of the Telnet lines in the Packet list section of Wireshark, and from the drop-down list, select Follow TCP Stream.
c. The Follow TCP Stream window displays the data for your Telnet session with the CyberOps Workstation VM. The entire session is displayed in plaintext, including your password. Notice that the username that you entered is displayed with duplicate characters. This is caused by the echo setting in Telnet to allow you to view the characters that you type on the screen.
d. After you have finished reviewing your Telnet session in the Follow TCP Stream window, click Close.
e. Type exit at the terminal to exit the Telnet session.
[analyst@secOps ~]$ exit
Part 2: Examine an SSH Session with Wireshark
same steps for ssh also just replace telnet with ssh localhost



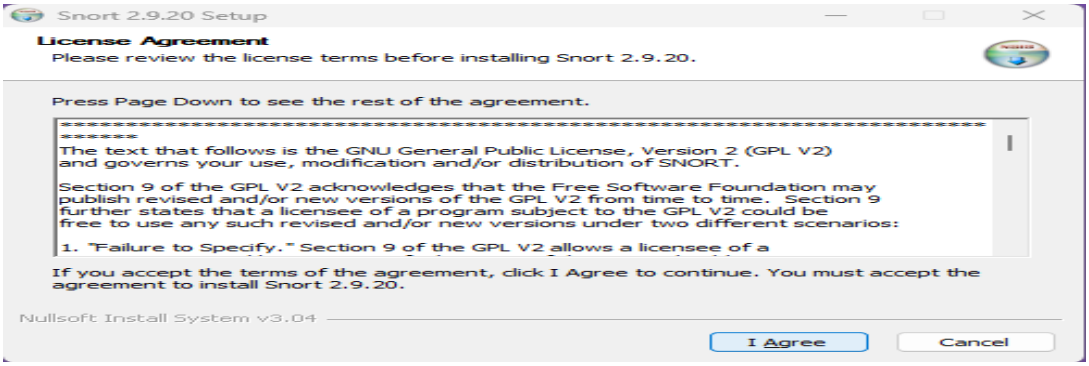
Practical 3

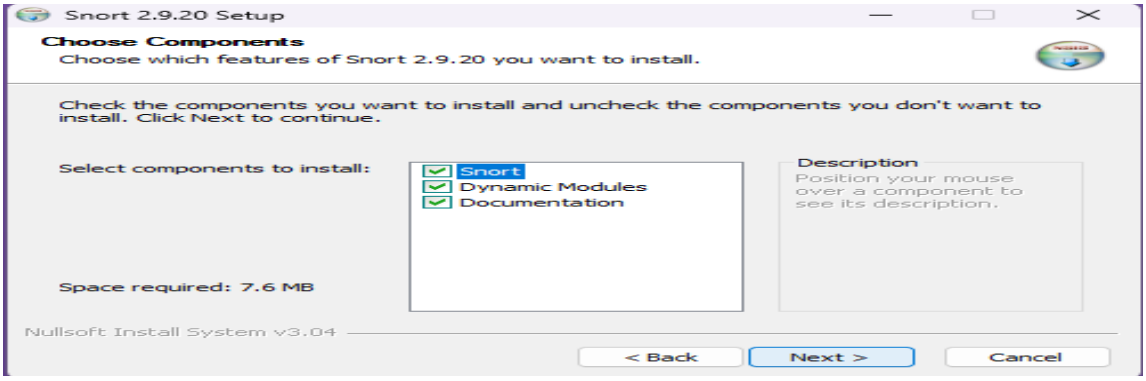
a. Demonstrate the use of Snort and Firewall Rules

Step-1-> Download snort application from https://snort.org/downloads from binaries section Snort_2_9_20_Installer.x64.exe

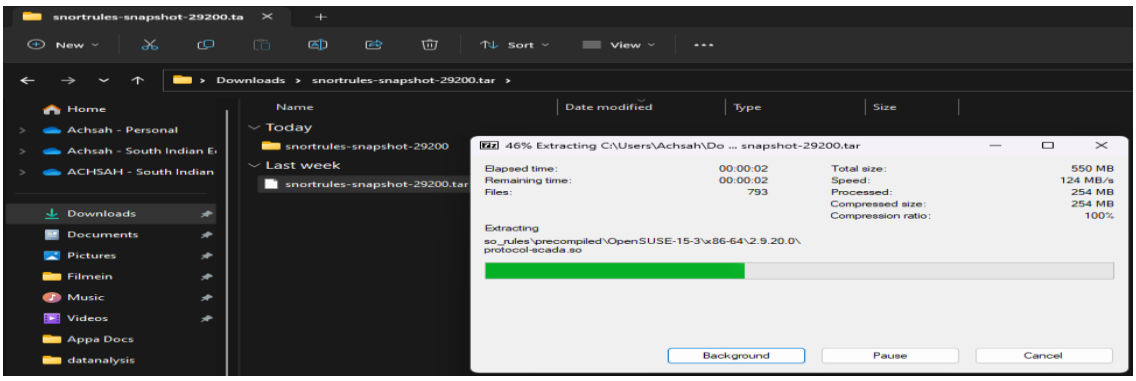
Step-2-> Download rules tar file from the registered section snortrules-snapshot-29200.tar.gz.

Step-3-> Install snort.exe file

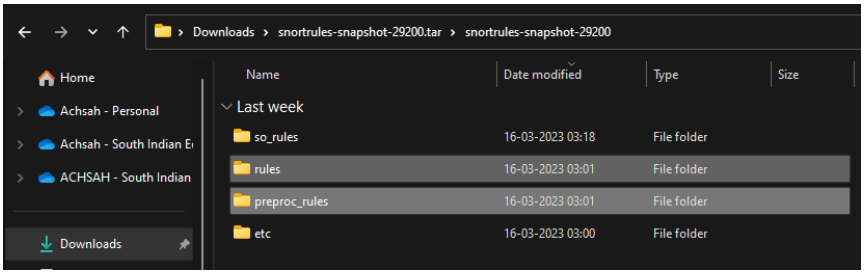




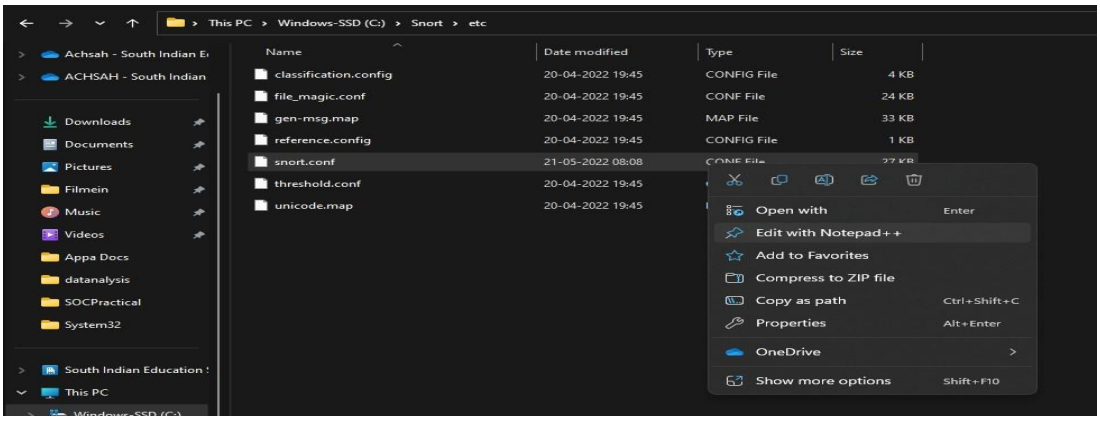
Step-4-> Unzip and extract the .tar file



Step-5-> Copy rules and preproc_rules folder from the extracted folder and paste it to the Snort folder in C:\Snort thus replacing the existing folder in the same.



Step-6-> Download and install notepad++ from https://notepad-plus-plus.org/downloads/ and open the snort.conf file from C:\Snort\etc in the same.



Step-7-> Search for blacklist from the snort folder in C:\Snort, open in notepad++, do save as and name it to whitelist.rules now you will have files with name blacklist.rules and whitelist.rules in the folder.

Step-8-> Edit the config file in the following lines

```
44  # Setup the network addresses you a
45  ipvar HOME_NET 192.168.0.0/24
46
47  # Set up the external network addre
48  ipvar EXTERNAL_NET !$HOME_NET
49
```

```
104 var RULE_PATH C:\Snort\rules
105 # var SO_RULE_PATH ../so_rules
106 var PREPROC_RULE_PATH C:\Snort\preproc_rules
```

```
112 # Set the absolute path appropriately
113 var WHITE_LIST_PATH C:\Snort\rules
114 var BLACK_LIST_PATH C:\Snort\rules
115
```

```
185   #
186   config logdir: C:\Snort\log
187
```

```
246   # path to dynamic preprocessor libraries
247   dynamicpreprocessor directory C:\Snort\lib\snort_dynamicpreprocessor
248
249   # path to base preprocessor engine
250   dynamicengine C:\Snort\lib\snort_dynamicengine\sf_engine.dll
251
252   # path to dynamic rules libraries
253   # dynamicdetection directory /usr/local/lib/snort_dynamicrules
254
```

```
264   # Does nothing in IDS mode
265   # preprocessor normalize_ip4
266   # preprocessor normalize_tcp: ips ecn stream
267   # preprocessor normalize_icmp4
268   # preprocessor normalize_ip6
269   # preprocessor normalize_icmp6
270
```

Step-9-> From line no. 546 to 651 replace forward slash to backslash of the path given.

Step-10-> Open local.rules from C:\Snort\rules in notepad++ and add the following lines of code in it.

```
21
22  alert icmp any any -> any any (msg:"Testing ICMP"; sid:1000001;)
23
24  alert udp any any -> any any (msg:"Testing UDP"; sid:1000002;)
25
26  alert tcp any any -> any any (msg:"Testing TCP"; sid:1000003;)
```

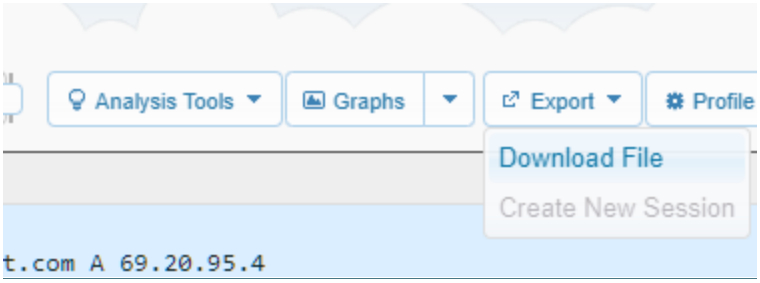
Step-11-> Open command line in the path C:\Snort\bin and run the command snort -W

```
C:\Snort\bin>snort -W

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20-WIN64 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

Index   Physical Address       IP Address      Device Name     Description
-----   ----------------       ----------      -----------     -----------
    1   00:00:00:00:00:00       disabled        \Device\NPF_{70B1D23C-CB1B-4DA2-AACE-6053BA239C62}      WAN Miniport (Network Monitor)
    2   00:00:00:00:00:00       disabled        \Device\NPF_{7D41CBC9-C450-4414-9246-4557C6AF3787}      WAN Miniport (IPv6)
    3   00:00:00:00:00:00       disabled        \Device\NPF_{A02C2DF4-6A26-47AB-841F-EFA013BD4B4D}      WAN Miniport (IP)
    4   E8:D0:FC:E1:EF:50       169.254.39.155  \Device\NPF_{5F24DE34-CF9E-468C-91A3-7FCE4440E6CF}      Bluetooth Device (Personal Area Network)
    5   E8:D0:FC:E1:EF:4F       192.168.0.102   \Device\NPF_{8DD3EE5E-90F0-45E9-B649-C54C390F5069}      Qualcomm Atheros QCA9377 Wireless Network Adapter
    6   FA:D0:FC:E1:EF:4F       169.254.184.44  \Device\NPF_{05E7CDB5-AC63-4D2C-B689-73EBB461E09B}      Microsoft Wi-Fi Direct Virtual Adapter #4
    7   EA:D0:FC:E1:EF:4F       169.254.7.234   \Device\NPF_{FFDD3A86-D792-4D13-A9AF-599631C8FBBA}      Microsoft Wi-Fi Direct Virtual Adapter #3
    8   00:00:00:00:00:00       0000:0000:0000:0000:0000:0000:0000:0000 \Device\NPF_Loopback    Adapter for loopback traffic capture

C:\Snort\bin>
```

Step-12-> Run the command: snort -i 5 -c C:\Snort\etc\snort.conf -A console

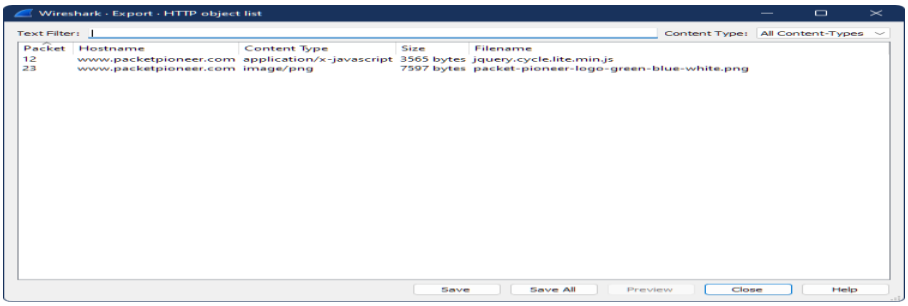
b. Demonstrate Extract an Executable from a PCAP

Step-1-> Download packet capture file from :
https://www.cloudshark.org/captures/a9472fbe700a

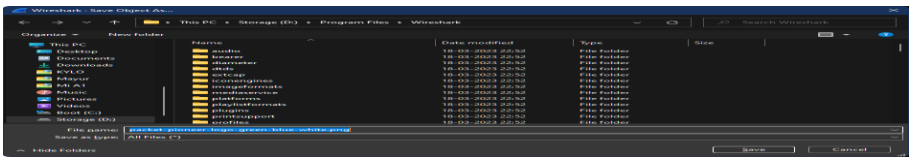


Step-2-> Open the packet capture file in Wireshark by dragging and dropping the capture file into the Wireshark interface.

Step-3-> Click on File->Export Objects-> HTTP

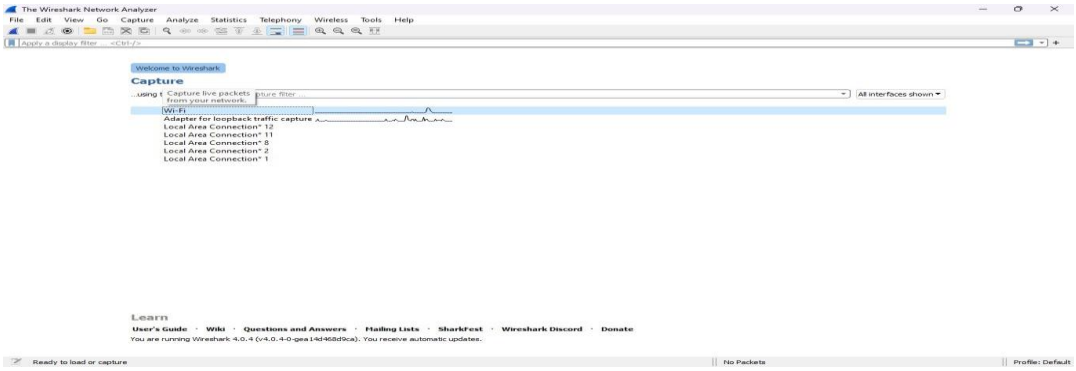


Step-4-> Select image/png and click on Save. Select Save Location.

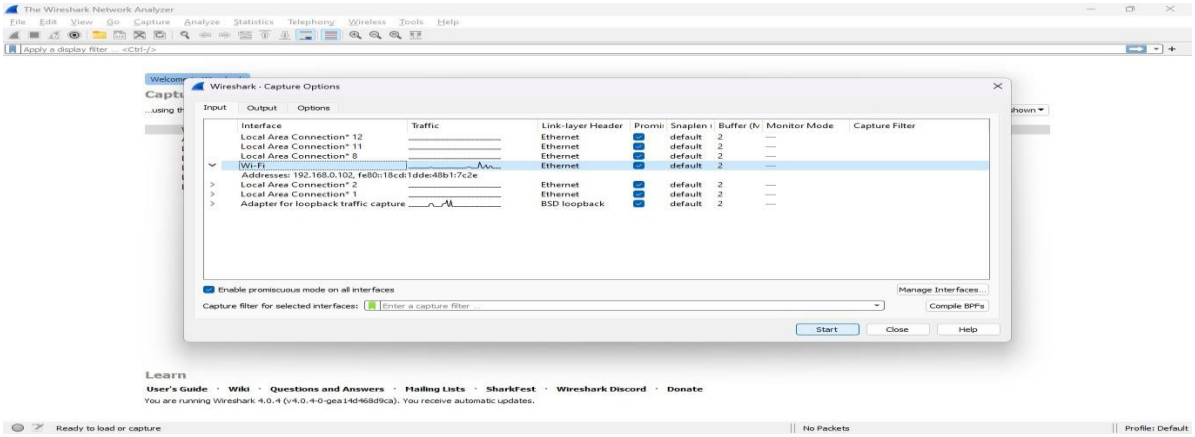


c. Demonstrate a practical for Exploring DNS Traffic

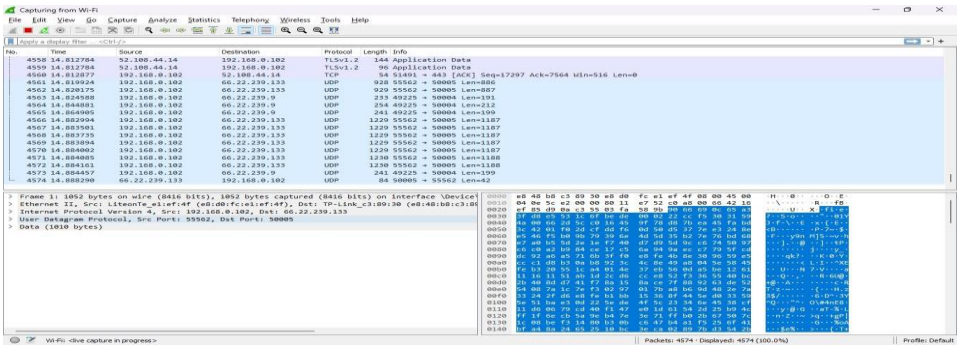
Step-1->Open Wireshark, select Wi-Fi and click on Capture to capture the DNS Traffic



Step-2->Click start



Step-3->Following window will open



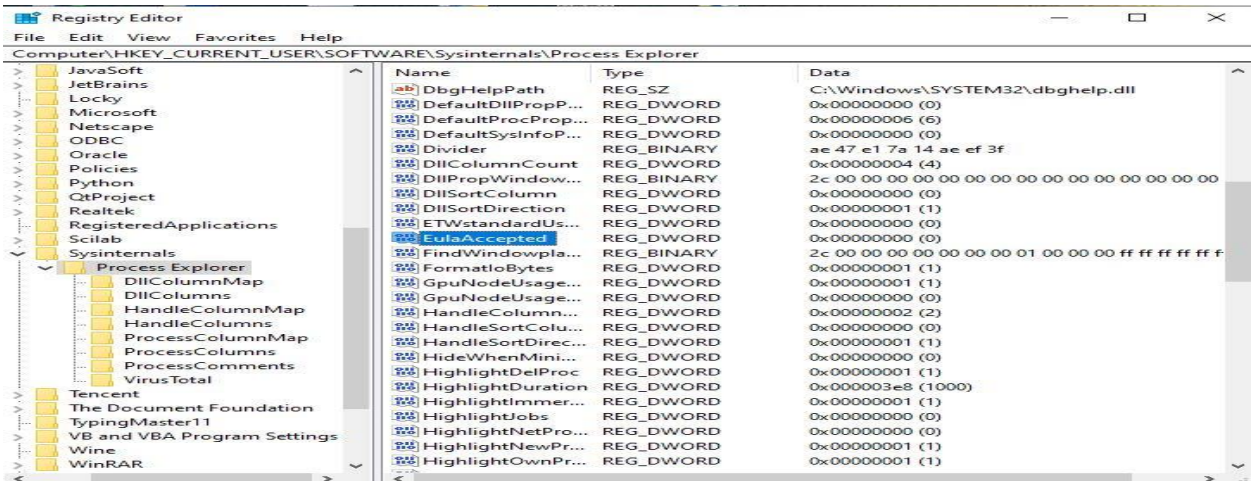
Step-4->Search for dns in the search bar

Step-5->Search for some websites and check them on Wireshark, it will give you the following output **Step-6->**Double click on that and open the DNS arrow and analyse it

Practical 4

b. Exploring Processes, Threads, Handles, and Windows Registry

Download Windows SysInternals Suite.
https://technet.microsoft.com/en- us/sysinternals/bb842062.aspx
After the download is completed, extract the files from the folder.
Open folder -> Open procexp.exe.
The Microsoft Edge process can be terminated in the Process Explorer. Right-click the selected process and select Kill Process. Click OKto continue.
Open a Command Prompt, Drag the Find Window's Processicon into the Command Prompt, then seen in Process Explorer.
In cmd type
ping -t www.cisco.com
these ping.exe display in process explorer.
to check malicious content, right-click conhost.exe -> Check VirusTotal -> Yes -> Ok
Step 1: Exploring Threads and Handles
right-click conhost.exe ->Properties ->Threads = view active thread
Step 2: Explore handles.
View> select Lower Pane View > Handles = view handle associated with conhost.exe process
Exploring Windows Registry
click Start -> Registry Editor -> Yes
HKEY_CURRENT_USER> Software> Sysinternals> Process Explorer.
Scroll down to locate the key EulaAccepted = currently value is (1)
double clicl and change value to 0 (indicates that the EULA was not accepted)
open procexp.exe
Result - EULA licence seen



Practical 5

Perform a practical to Attack on a mySQL Database by using PCAP File.

Part 1: Open Wireshark and load the PCAP file.
Start the CyberOps Workstation VM.
Click Applications > CyberOPS > Wireshark
next go to File >open> /home/analyst/ > lab.support.files > SQL_Lab.pcap
Part 2: View the SQL Injection Attack.
right-click line 13 and select Follow > HTTP Stream.
Find field, enter 1=1. Click Find Next.
Clear display filter
Part 3: The SQL Injection Attack continues...
right-click line 19 and select Follow > HTTP Stream.
Find field, enter 1=1. Click Find Next.
Clear display filter
Part 4: The SQL Injection Attack provides system information.
right-click line 22 and select Follow > HTTP Stream.
Find field, enter 1=1. Click Find Next.
Clear display filter
Part 5: The SQL Injection Attack and Table Information.
right-click line 25 and select Follow > HTTP Stream.
Find field, enter 1=1. Click Find Next.
Clear display filter
Part 6: The SQL Injection Attack Concludes.
right-click line 28 and select Follow > HTTP Stream.
Find field, enter 1=1. Click Find Next.
Which user has the password hash of 8d3533d75ae2c3966d7e0d4fcc69216b?
first search password on find after these visible then go to
https://crackstation.net/ and enter the password and hash the password

Practical 6

Create your own syslog Server

Step 1->Run **sudo apt-get install openssh-server** command to install ssh

```
ubuntu@ubuntu:~$ sudo apt-get install openssh-server
[sudo] password for ubuntu:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-server is already the newest version (1:8.2p1-
0.5).
The following package was automatically installed and
onger required:
  libfprint-2-tod1
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 32 not
d.
```

Step 2->Run **sudo apt-get install net-tools** to install net-tools

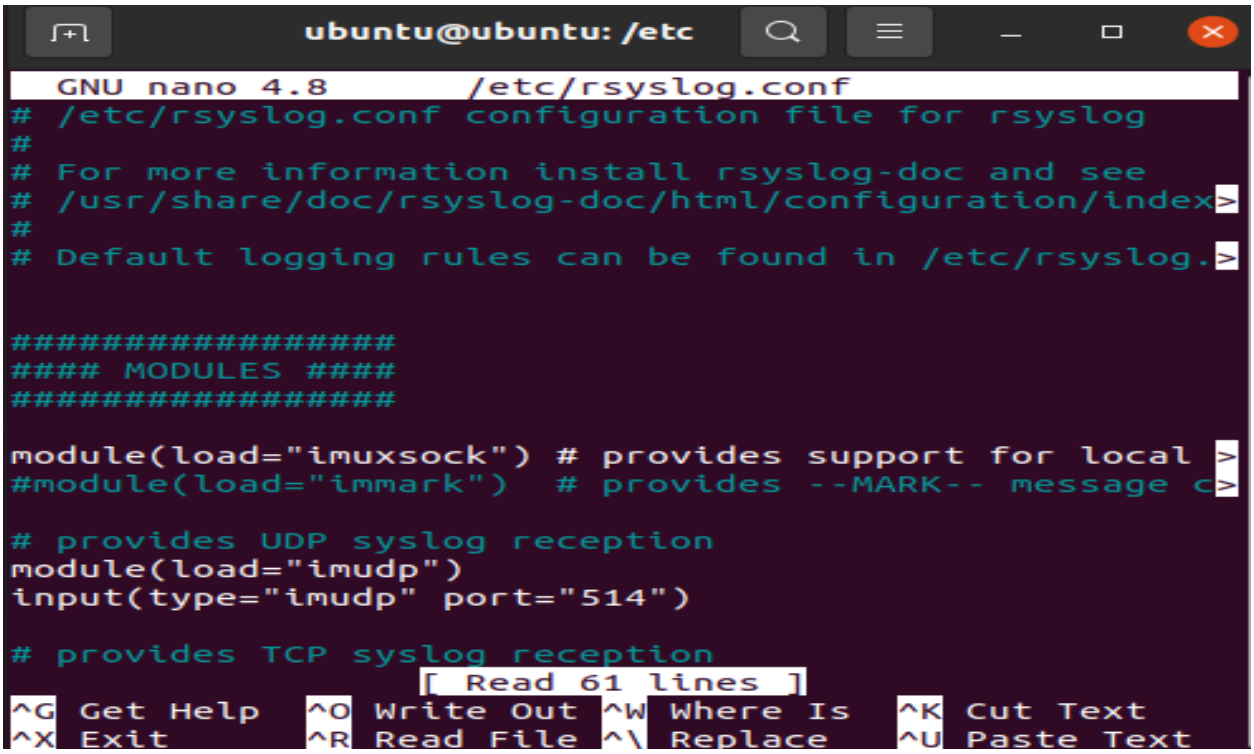
```
ubuntu@ubuntu:~$ sudo apt-get install net-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
net-tools is already the newest version (1.60+gi
d88e-1ubuntu1).
The following package was automatically installe
onger required:
  libfprint-2-tod1
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 3
d
```

Step 3->Run sudo service rsyslog restart to restart rsyslog service

```
ubuntu@ubuntu:/etc$ sudo service rsyslog restart
ubuntu@ubuntu:/etc$
```

Step 4->Edit rsyslog config using sudo nano /etc/rsyslog.conf command and uncomment the lines - module(load="imudp") input(type="imudp" port="514") module(load="imtcp") input(type="imtcp" port="514") Press Ctrl+X & Enter to save file

```
ubuntu@ubuntu:/etc$ sudo nano /etc/rsyslog.conf
```



Step 5-> Restart rsyslog service with **sudo service rsyslog restart** command

```
ubuntu@ubuntu:/etc$ sudo service rsyslog restart
```

Step 6-> Check status of rsyslog service with **sudo service rsyslog status** command

```
ubuntu@ubuntu:/etc$ sudo service rsyslog status
● rsyslog.service - System Logging Service
     Loaded: loaded (/lib/systemd/system/rsyslog.serv
     Active: active (running) since Fri 2023-04-21 04
TriggeredBy: ● syslog.socket
       Docs: man:rsyslogd(8)
             https://www.rsyslog.com/doc/
   Main PID: 14060 (rsyslogd)
      Tasks: 9 (limit: 3449)
     Memory: 1.2M
     CGroup: /system.slice/rsyslog.service
             └─14060 /usr/sbin/rsyslogd -n -iNONE
Terminal
Apr 21 04:16:20 ubuntu systemd[1]: Starting System Lo
Apr 21 04:16:20 ubuntu systemd[1]: Started System Log
Apr 21 04:16:20 ubuntu rsyslogd[14060]: imuxsock: Acq
Apr 21 04:16:20 ubuntu rsyslogd[14060]: rsyslogd's gr
Apr 21 04:16:20 ubuntu rsyslogd[14060]: rsyslogd's us
Apr 21 04:16:20 ubuntu rsyslogd[14060]: [origin softw
lines 1-18/18 (END)
```

Practical 7

Configure your Linux system to send syslog messages to a syslog server

and read them

Step 1: Configuring the Syslog Server (server-syslog)

sudo nano /etc/sysconfig/syslog
sudo apt update
sudo apt install rsyslog
sudo nano /etc/rsyslog.conf
remove # following line
module(load="imudp") # UDP listener
input(type="imudp" port="514")
add line = *.debug   /var/log/messages
sudo systemctl restart rsyslog
sudo netstat -anu | grep 514

Step 2: Configuring the Syslog Client (oer-host)

sudo nano /etc/hosts
add line = 10.10.10.1 server-syslog.domain.com server-syslog loghost
sudo nano /etc/rsyslog.conf
add line = *.debug @loghost
sudo systemctl restart rsyslog
Step 3: Testing the Configuration
logger -p user.debug "Test message from client"
sudo cat /var/log/messages | grep "Test message from client"

Practical 8

Install and Run Splunk on Linux

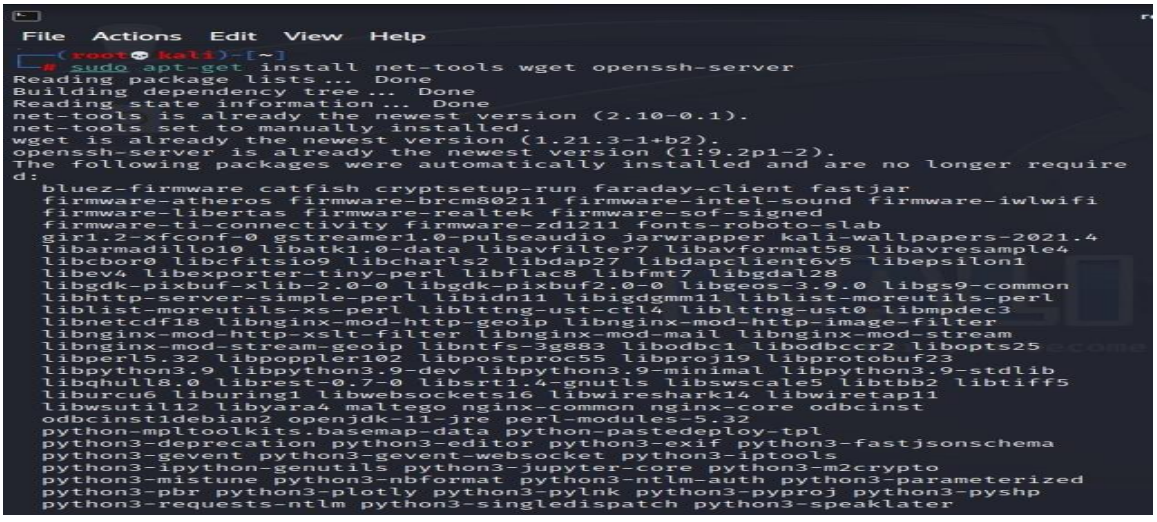
1) sudo aptget update
2) sudo aptget upgrade
3) sudo apt-get install net-tools wget openssh-server
4) sudo systemctl restart ssh
5) sudo nano /etc/rsyslog.conf
6) module(load="imuxsock")
7) module(load="imklog")
8) module(load="imudp") input(type="imudp" port="514")
9) module(load="imtcp") input(type="imtcp" port="514")
11) sudo systemctl restart rsyslog

if we get
$ sudo systemctl restart rsyslog
Failed to restart rsyslog.service: Unit rsyslog.service not found.
then use following command:
$ sudo apt-get purge rsyslog
$ sudo apt-get install rsyslog
$ sudo systemctl restart rsyslog

10)sudo systemctl status rsyslog
11)sudo groupadd splunkdemo
12)sudo useradd -d /opt/splunk -m -g splunk splunkdemo
13) Now go to https://www.splunk.com/ . Register and Click on Free Splunk at top right of
page. Click on Free Trials and Downloads page link. On the next screen Click on Get My Free Trial
under Splunk Enterprise. Next Choose Linux and click on Download for .tgz. Stop the download and
click
14)Stop the download and click on Dowload via Command Line (wget) and copy the command.
15)su -
Password:

16)su - splunkdemo
if error come likes su: user splunkdemo does not exist or the user entry does not contain all the required fields
command
cat /etc/passwd | grep splunkdemo
sudo groupadd splunk
sudo useradd -d /opt/splunk -m -g splunk -s /bin/bash splunkdemo
sudo passwd splunkdemo
cat /etc/passwd | grep splunkdemo
cat /etc/group | grep splunk

cat /etc/passwd | grep splunkdemo
sudo groupadd splunk
sudo useradd -d /opt/splunk -m -g splunk -s /bin/bash splunkdemo
sudo passwd splunkdemo
cat /etc/passwd | grep splunkdemo
cat /etc/group | grep splunk
17)su - splunkdemo
18)paste the command of tgz file(cmd that start wget step 14)
19)tar xvzf splunk-9.4.0-6b4ebe426ca6-linux-amd64.tgz -C /opt (edit these command with your tgz file version)
20)access root using su – (write root password (kali))
21)chown -R splunkdemo: /opt/splunk/
22)cd /opt/splunk/bin
23)./splunk start
24) ./splunk start --accept-license
after these we get "The Splunk web interface is at http://kali:8000"
25)interface on http://kali:8000.
go these link if your step is correct then get login screen else any error in your steps.



Practical 9

Install and Configure ELK on Linux1.sudo apt-get update
password :kali
2.sudo apt-get upgrade
3.sudo apt-get install default-jdk default-jre
4.java -version
5.update-alternatives --config java
6.sudo nano /etc/environment
set the path :
JAVA_HOME="/usr/lib/jvm/java-23-openjdk-amd64/bin/java"
#in case if the above cmd isnt working then to temporary set the path use this command because sudo nano isnt working :
export JAVA_HOME=/usr/lib/jvm/java-23-openjdk-amd64/bin/java
7.source /etc/environment
8.sudo apt-get install apt-transport-https
9.wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg
10.echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list
11.sudo apt-get update
12.sudo apt-get install elasticsearch
13.sudo systemctl daemon-reload
14.sudo systemctl enable elasticsearch.service
15.sudo systemctl start elasticsearch.service
16.sudo systemctl status elasticsearch.service
17.sudo nano /etc/elasticsearch/elasticsearch.yml
network.host: 0.0.0.0
discovery.seed_hosts:[]
xpack.security.enabled: false
18.sudo systemctl restart elasticsearch.service
19.sudo nano /etc/elasticsearch/jvm.options
edit and change to:
-Xms512m
-Xmx512m
20.curl -X GET "localhost:9200"
21.ifconfig
copy the inet 10.0.2.15 append :9200 to it (10.0.2.15:9200\) run this in the browser
22.sudo apt-get install logstash

sudo systemctl start logstash
sudo systemctl enable logstash
sudo systemctl status logstash

23.sudo apt-get install kibana
sudo systemctl start kibana
sudo systemctl enable kibana
sudo systemctl status kibana
24.sudo nano /etc/kibana/kibana.yml
make changes as:
server.port:5601
server.host: "0.0.0.0"
elasticsearch.hosts: ["http://localhost:9200"]
25.sudo systemctl restart kibana

26.sudo apt-get install filebeat
27.sudo nano /etc/filebeat/filebeat.yml
make changes as shown in the screen shots
28.sudo filebeat setup --index-management -E output.logstash.enabled=false -E 'output.elasticsearch.hosts=["0.0.0.0:9200"]'
29.sudo systemctl start filebeat
30.sudo systemctl enable filebeat
31.curl -X GET http://{10.0.2.15}:9200/_cat/indices?v
32.In the browser run the following address:
http://10.0.2.15:9200/_cat/indices?v
http://10.0.2.15:5601