EMAIL SPAM CLASSIFICATION

EX NO: 1

DATE: 10.02.2023

AIM:

To Build a Solution to Identify and Prevent Phishing Attacks via Email

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Load Your Email Dataset And Perform Any Necessary Pre-processing Steps.

Step-3: Extract Features from the Pre-processed Email Data

Step-4: Create and Train the Model

Step-5: Use the Trained Model to Make Predictions on the Testing Set

Step-6: Assess the Performance of the Model by Comparing the Predicted Labels with the True Labels from the Testing Set

Step-7: If Necessary, Fine-Tune the Hyper parameters of the Chosen Algorithm to Improve the Model's Performance

Step-8: Deploy the Model

PROGRAM:

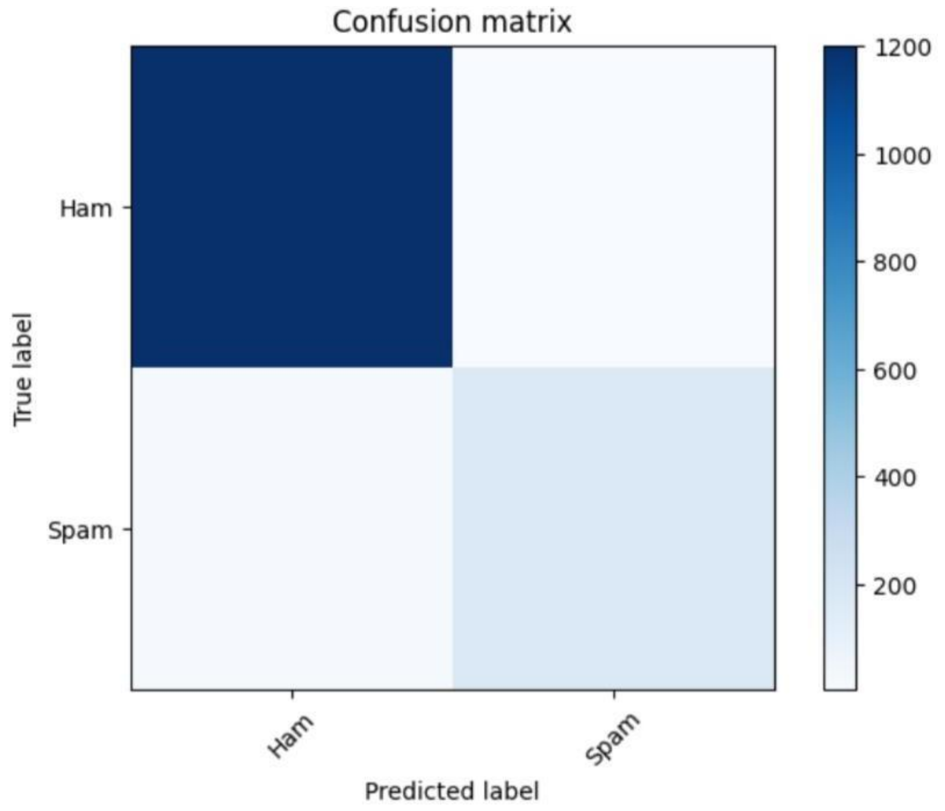
```
import matplotlib.pyplot as plt
import nltk
import numpy as np
import pandas as pd
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn import svm
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import classification_report,
confusion_matrix
from sklearn.model_selection import GridSearchCV
df = pd.read_csv('spam.csv', encoding='utf-8')
ham_spam = {'ham': 0, 'spam': 1}
df['Label'] = df['Label'].map(ham_spam)
```

```
nltk.download('stopwords')
ps = PorterStemmer()
stop_words = set(stopwords.words('english'))
df['EmailText'] = df['EmailText'].apply(
    lambda x: ' '.join([ps.stem(word).lower() for word in
x.split() if word not in stop_words]))
train = df.sample(frac=0.75)
test = df.drop(train.index)
print('Training set size: ', len(train))
print('Testing set size: ', len(test))
vectorizer = CountVectorizer()
vectorizer.fit(df['EmailText'])
train_data = vectorizer.transform(train['EmailText'])
test_data = vectorizer.transform(test['EmailText'])
tuned_parameters = dict(kernel=['rbf', 'linear'], gamma=[1e-3, 1e-
4], C=[1, 10, 100, 1000])
model = GridSearchCV(svm.SVC(), tuned_parameters, cv=5, n_jobs=-1)
model.fit(train_data, train['Label'])
pred: np.ndarray = model.predict(test_data)
print('Model Accuracy: ', round(np.mean(pred == test['Label']) *
100, 3), '%')
print(classification_report(test['Label'], pred))
cm: np.ndarray = confusion_matrix(test['Label'], pred)
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion matrix')
plt.colorbar()
tick_marks = np.arange(2)
plt.xticks(tick_marks, ['Ham', 'Spam'], rotation=45)
plt.yticks(tick_marks, ['Ham', 'Spam'])
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
test_examples = test.sample(n=5)
inv_ham_spam = {v: k for k, v in ham_spam.items()}
```

```
    pred: np.ndarray =
    model.predict(vectorizer.transform(test_examples['EmailText']))
    print("EmilText\t\t\tPredicted\tActual")
    for i in range(len(test_examples)):
        print(test_examples.iloc[i]['EmailText'][:20],
              '\t', inv_ham_spam[pred[i]],
              '\t', inv_ham_spam[test_examples.iloc[i]['Label']])
```

OUTPUT:

```
Model Accuracy:  98.492 %


              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1207
           1       0.97      0.91      0.94       186

    accuracy                           0.98      1393
   macro avg       0.98      0.95      0.97      1393
weighted avg       0.98      0.98      0.98      1393
```



```
EmilText                     Predicted      Actual
er, hello, thing did          ham            ham
&lt;#&gt; mca. but c          ham            ham
sunshin hols. to cla          spam           spam
if let this, want ho          ham            ham
slept? i thinkthi ti          ham            ham
```

RESULT:

The conclusion of running this code would be the accuracy score, which indicates the performance of the SVM model in classifying the emails as spam or non-spam. The accuracy score ranges from 0 to 1, where a higher value signifies better performance.

TEXT SUMMARIZATION USING NLP

EX NO: 2

DATE: 17.02.2023

AIM:

To Implement NLP tasks for text analysis or text generation

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Load the text you want to summarize into a string variable

Step-3: Select an appropriate algorithm or model for text summarization

Step-4: Use the selected algorithm or model to generate a summary from the pre-processed text

Step-5: Print or display the generated summary

PROGRAM:

```
from transformers import pipeline

summarizer = pipeline("summarization")

text = """ Hear me, Subjects of Ymir. My name is Eren Yeager. I'm
adressing my fellow Subjects of Ymir, speaking to you directly
through the

power of the Founder. All the walls on the island of Paradis have
crumbled to the ground, and the legions of Titans burried with in

have begun their march. My only goal is to protect the lives of
the people of Paradis the island where I was born. Right now, the

nations of the world are united in the desire to exterminate my
people, and it won't end with our island. They won't be satsified

until every last Subject of Ymir is dead. I won't let them have
their way. The Titans of the walls, will continue their march
until
```

```
every trace of life beyond our shores is trampled flat, and the
people of Paradis are all that remains of humanity."""


summary = summarizer(text, max_length=130, min_length=30,
do_sample=False)
print(summary[0]['summary_text'])
```

OUTPUT:

```
All the walls on the island of Paradis have crumbled to the ground
, and the legions of Titans have begun their march . The Titans of
the walls, will continue their march until every trace of life be
yond our shores is trampled flat
```

RESULT:

The conclusion of running the code generates a summary by selecting the top-ranked sentences according to the specified number of sentences desired in the summary. The summary is then displayed or printed.

ENCRYPTION OF USER DATA USING PYTHON

EX NO: 3

DATE: 24.02.2023

AIM:

To create a solution to protect sensitive data such as p/w or credit card information in a database

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Define Encryption Algorithm

Step-3: Generate or Load Encryption Key

Step-4: Encrypt the Data

Step-5: Save or Transmit Encrypted Data

Step-6: Deploy the Model

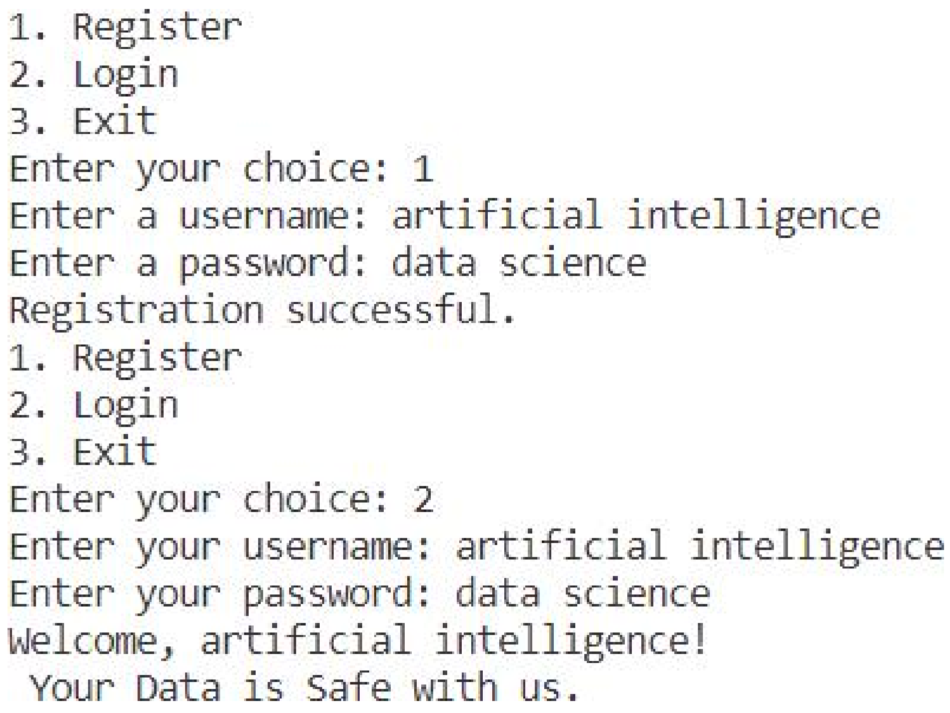
PROGRAM:

```
import hashlib

import json


def encrypt_password(password):

    sha256 = hashlib.sha256() # Create a SHA-256 hash object.

    password_bytes = password.encode('utf-8')    # Convert the
password string to bytes

    sha256.update(password_bytes)    # Update the hash object with
the password bytes

    encrypted_password = sha256.hexdigest()    # Get the encrypted
password in hexadecimal format

    return encrypted_password
```

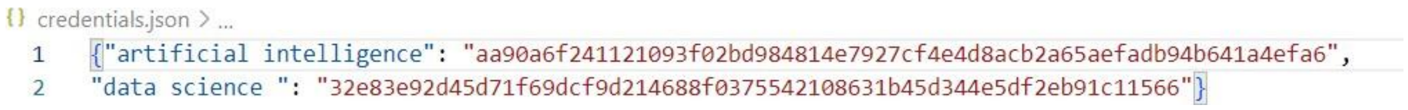
```
def save_credentials(username, password):

    try:# Load existing credentials from the JSON file

        with open('credentials.json', 'r') as file:

            credentials = json.load(file)

    except FileNotFoundError:

        credentials = {}

    encrypted_password = encrypt_password(password)     # Encrypt
the password

    credentials[username] = encrypted_password     # Add the new
credentials to the dictionary

    with open('credentials.json', 'w') as file:     # Save the
updated credentials to the JSON file

        json.dump(credentials, file)


def login():

    username = input("Enter your username: ")     # Get the
username and password from the user

    password = input("Enter your password: ")

    with open('credentials.json', 'r') as file:    # Load the
credentials from the JSON file

        credentials = json.load(file)


    if username in credentials:     # Check if the username exists

        encrypted_password = encrypt_password(password)         #
Encrypt the entered password

        if encrypted_password == credentials[username]:         #
Compare the encrypted password with the stored password
```

```
            print(f"Welcome, {username}!\n Your Data is Safe with
us. \n ")


        else:
            print("Incorrect password.")
    else:
        print("Username not found.")


# Main program
while True:
    print("1. Register\n2. Login\n3. Exit")
    choice = input("Enter your choice: ")


    if choice == '1':
        username = input("Enter a username: ")
        password = input("Enter a password: ")
        save_credentials(username, password)
        print("Registration successful.")
    elif choice == '2':
        login()
    elif choice == '3':
        break
    else:
        print("Invalid choice. Please try again.")
```

OUTPUT:



```
1. Register
2. Login
3. Exit
Enter your choice: 1
Enter a username: artificial intelligence
Enter a password: data science
Registration successful.
1. Register
2. Login
3. Exit
Enter your choice: 2
Enter your username: artificial intelligence
Enter your password: data science
Welcome, artificial intelligence!
 Your Data is Safe with us.
```



```
credentials.json > ...
{"artificial intelligence": "aa90a6f241121093f02bd984814e7927cf4e4d8acb2a65aefadb94b641a4efa6",
"data science ": "32e83e92d45d71f69dcf9d214688f0375542108631b45d344e5df2eb91c11566"}
```

RESULT:

This Python code demonstrates a basic implementation of encryption. The code typically includes importing the necessary libraries or modules, defining the encryption algorithm, and applying it to a given message or data.

TEXT GENERATION USING GPT NEO

EX NO: 4

DATE: 03.03.2023

AIM:

To implement recurrent neural networks (RNNs) or transformer models (GPT) for NLP tasks like sentiment analysis or text generation

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Load a pre-trained GPT-Neo model and the corresponding tokenizer

Step-3: Use the loaded model and tokenizer to generate text

Step-4: Post-process and Display the Text

PROGRAM:

```
from transformers import pipeline # First line

generator = pipeline('text-generation', model='EleutherAI/gpt-neo-
1.3B') # Second line

prompt = "The current stock market" # Third line

res = generator(prompt, min_length=50, do_sample=True,
temperature=0.9) # Fourth line

print(res[0]['generated_text'])
```

OUTPUT:

```
The current stock market crisis is just the latest piece of eviden
ce that the government is not prepared to deal with the problem at
hand.

The economic crisis is being caused by a set of structural problem
s in the economy that cannot be solved by the government. In the m
eantime, we are seeing ever greater numbers of people going bankru
pt of their possessions as the financial institutions that once to
ok care of them collapse.
```

```
One of these institutions to be named in the new government plan t
o deal with the financial crisis is housing finance.

According to a recent report on the website of the Financial Times
, the financial crisis is now seen as the worst financial crisis s
ince the Great Depression. The financial crisis was caused by the
banking system that was bailed out when it was in trouble. When th
e banks collapsed in 2008, the government created new banks to tak
e care of the problems caused by the banks to the economy.

This bailout was one of the worst government bailouts that the wor
ld has ever seen and no one in the government that has a say in it
wants to undo it. This is even if you believe that the bankers wer
e the real problem in the crisis.

Of course it is all speculation, as if there are billions of dolla
rs in bonds
```

RESULT:

The outcome of running this code is the generated text, which is produced based on the patterns and knowledge learned by GPT-Neo during its training. The generated text can be adjusted by fine-tuning the generation parameters to control aspects like length, randomness, and the number of generated sequences

OBJECT DETECTION USING YOLOV8

EX NO: 5

DATE: 10.03.2023

AIM:

To implement object detection algorithms like Faster R-CNN or YOLO for detecting and localizing objects in images.

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Download the pre-trained YOLOv4 weights file

Step-3: Load the YOLOv4 model architecture and weights4

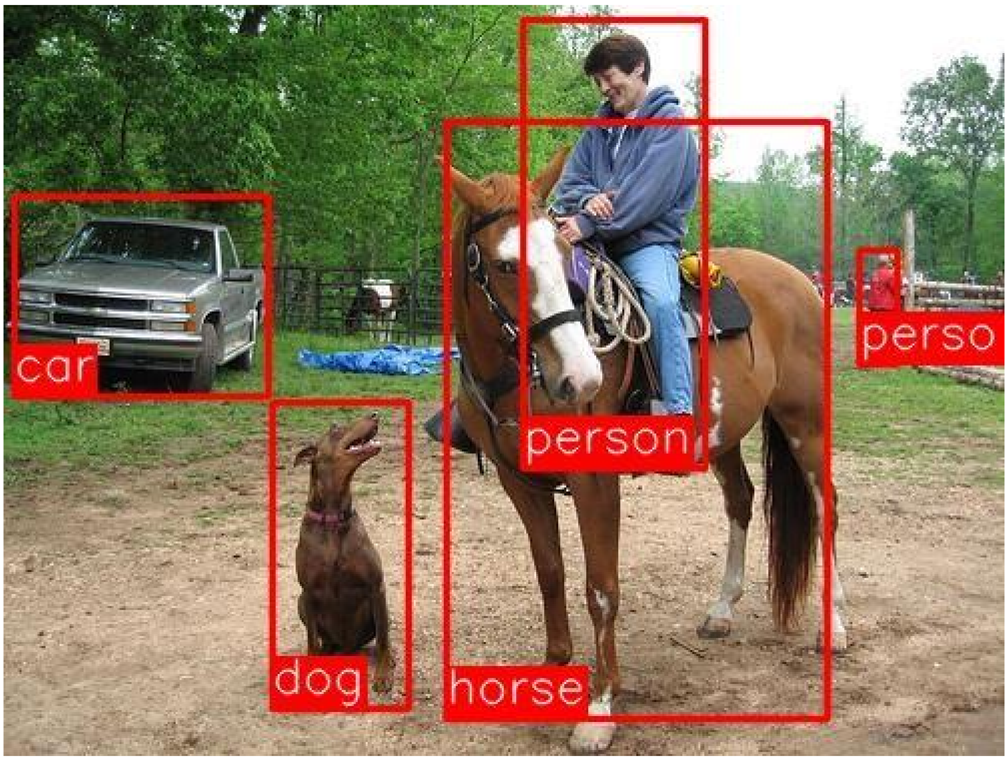
Step-4: Load the input image you want to perform object detection on using OpenCV

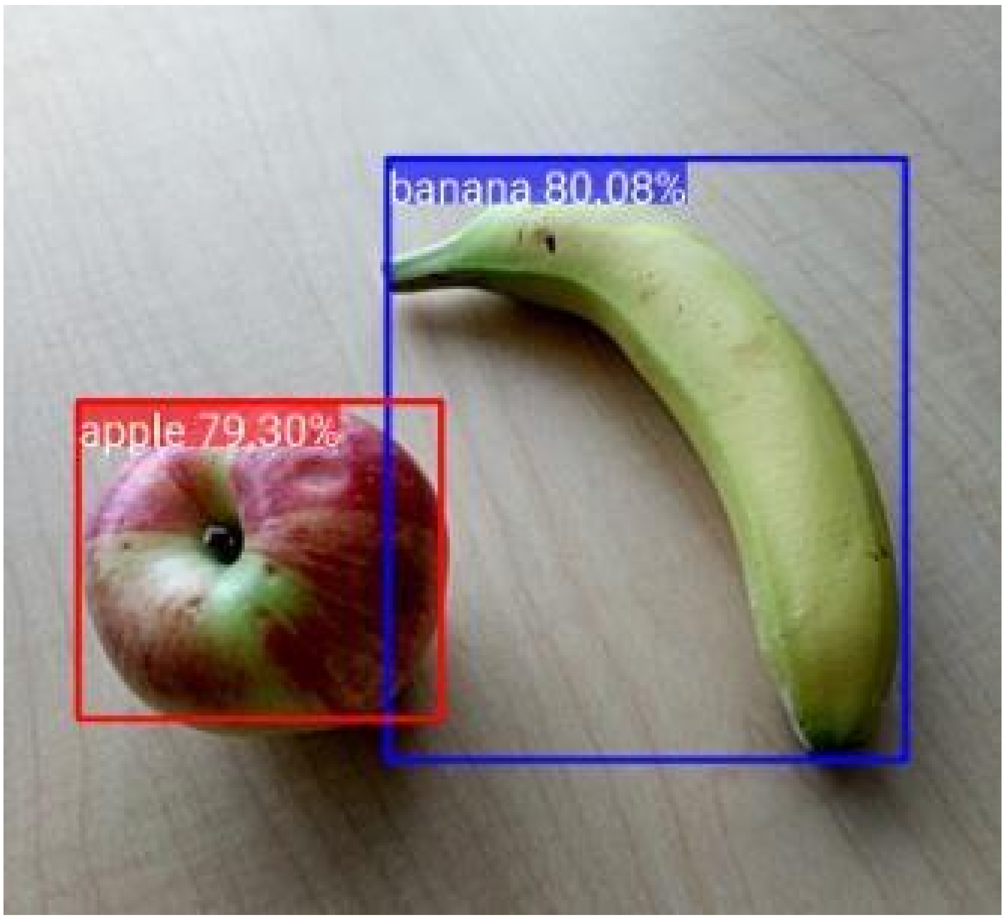
Step-5: Display or save the output image with the bounding boxes and class labels overlaid

PROGRAM:

```
""" NOTE THAT BEFORE RUNNING THE PROGRAM KINDLY INSTALL THE
FOLLOWING MODULES
'ultralytics', 'opencv-python','torch==2.0.0' """

import cv2
from ultralytics import YOLO

# Load the YOLOv8 model
model = YOLO('yolov8n.pt')

# Open the video file
#video_path = "path/to/your/video/file.mp4"
cap = cv2.VideoCapture(0)

# Loop through the video frames
while cap.isOpened():
    # Read a frame from the video
    success, frame = cap.read()
    if success:
        # Run YOLOv8 inference on the frame
        results = model(frame)
        annotated_frame = results[0].plot()

        # Display the annotated frame
        cv2.imshow("Object Detection", annotated_frame)

        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    else:
        # Break the loop if the end of the video is reached
        break

# Release the video capture object and close the display window
cap.release()
cv2.destroyAllWindows()
```

OUTPUT:





RESULT:

This code involves loading the YOLOv4 model and weights, pre-processing the input image or video, performing object detection by passing the data through the network, and post-processing the detection results to display or store them.

STOCK MARKET PREDICTION

EX NO: 6

DATE: 17.03.2023

AIM:

To develop a solution for big mart sales prediction - Stock market prediction using RNN

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Clean and pre-process the collected data to prepare it for training the prediction model

Step-3: Choose a suitable machine learning algorithm for stock market prediction

Step-4: Evaluate the performance of the trained model using the testing dataset

Step-5: Visualize the predicted stock prices or other relevant metrics to gain insights and assess the performance of the prediction model

PROGRAM:

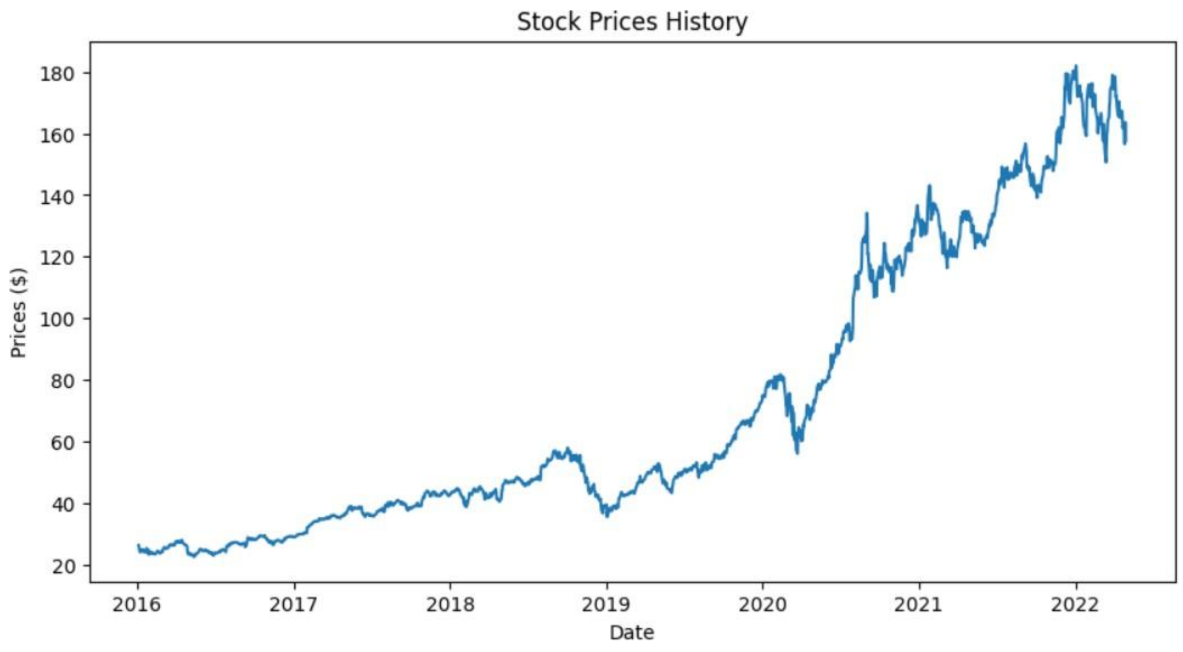
```
import math

import yfinance as yf

import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers


stock_data = yf.download('AAPL', start='2016-01-01', end='2022-05-
01')

stock_data.head()
```

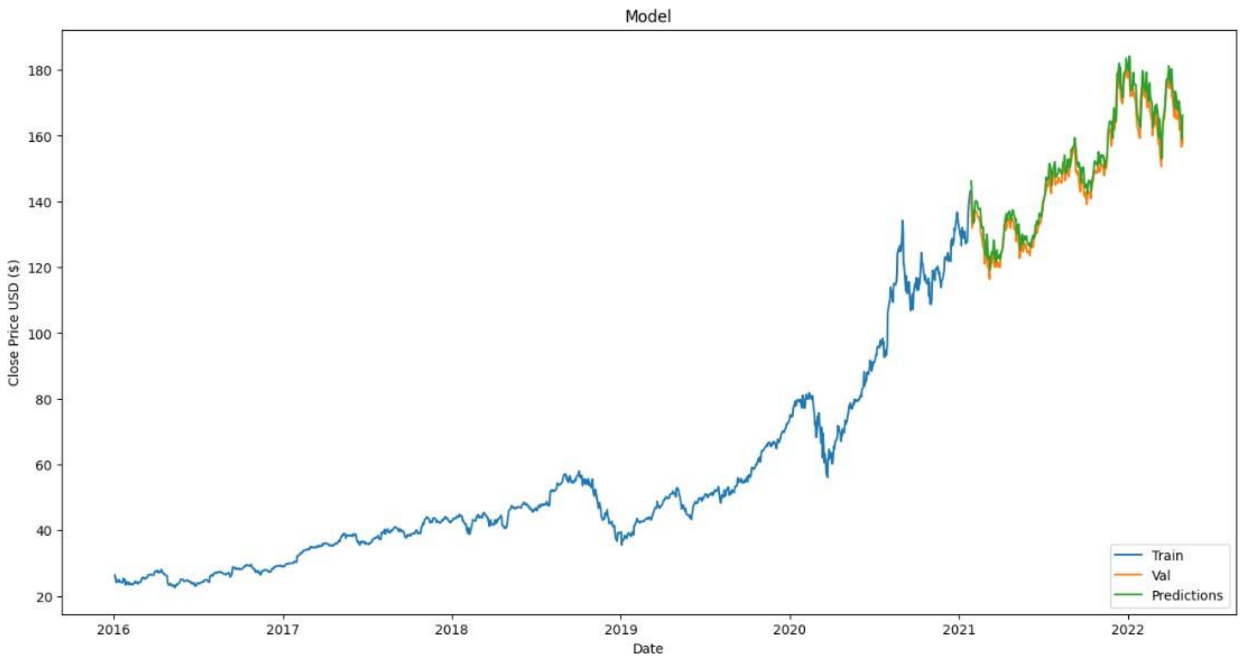
```
plt.figure(figsize=(10, 5))

plt.title('Stock Prices History')

plt.plot(stock_data['Close'])

plt.xlabel('Date')

plt.ylabel('Prices ($)')

close_prices = stock_data['Close']

values = close_prices.values

print(len(values))

training_data_len = math.ceil(len(values)* 0.8)


scaler = MinMaxScaler(feature_range=(0,1))

scaled_data = scaler.fit_transform(values.reshape(-1,1))

train_data = scaled_data[0: training_data_len, :]

print(len(train_data))


x_train = []

y_train = []


for i in range(60, len(train_data)):

    x_train.append(train_data[i-60:i, 0])

    y_train.append(train_data[i, 0])


x_train, y_train = np.array(x_train), np.array(y_train)

x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1],1))

test_data = scaled_data[training_data_len-60: , : ]
```

```
x_test = []
y_test = values[training_data_len:]


for i in range(60, len(test_data)):
  x_test.append(test_data[i-60:i, 0])


x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
model = keras.Sequential()
model.add(layers.LSTM(128, return_sequences=True, name="Layer_1",
input_shape=(x_train.shape[1], 1)))
model.add(layers.LSTM(128, return_sequences=True, name="Layer_2",
input_shape=(x_train.shape[1], 1)))
model.add(layers.LSTM(128, return_sequences=False,
name="Layer_3"))
model.add(layers.Dense(25, name="DLayer_1"))
model.add(layers.Dense(1, name="DLayer_2"))
model.summary()
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size= 1, epochs=8)
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
rmse = np.sqrt(np.mean(predictions - y_test)**2)
data = stock_data.filter(['Close'])
train = data[:training_data_len]
validation = data[training_data_len:]
validation['Predictions'] = predictions
```

```
plt.figure(figsize=(16,8))

plt.title('Model')

plt.xlabel('Date')

plt.ylabel('Close Price USD ($)')

plt.plot(train)

plt.plot(validation[['Close', 'Predictions']])

plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')

plt.show()
```

OUTPUT:





RESULT:

Stock market prediction using Python involves collecting historical data, pre-processing it, training a prediction model, and evaluating its performance and visualize the predicted stock prices or other relevant metrics to gain insights and communicate the results effectively

SIMPLE LINEAR REGRESSION BETWEEN TWO VARIABLES

EX NO: 7

DATE: 24.03.2023

AIM:

To Implement simple linear regression and analyse the relationship between two variables

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Load or create a dataset containing the two variables you want to perform linear regression

Step-3: Train the linear regression model using the training data

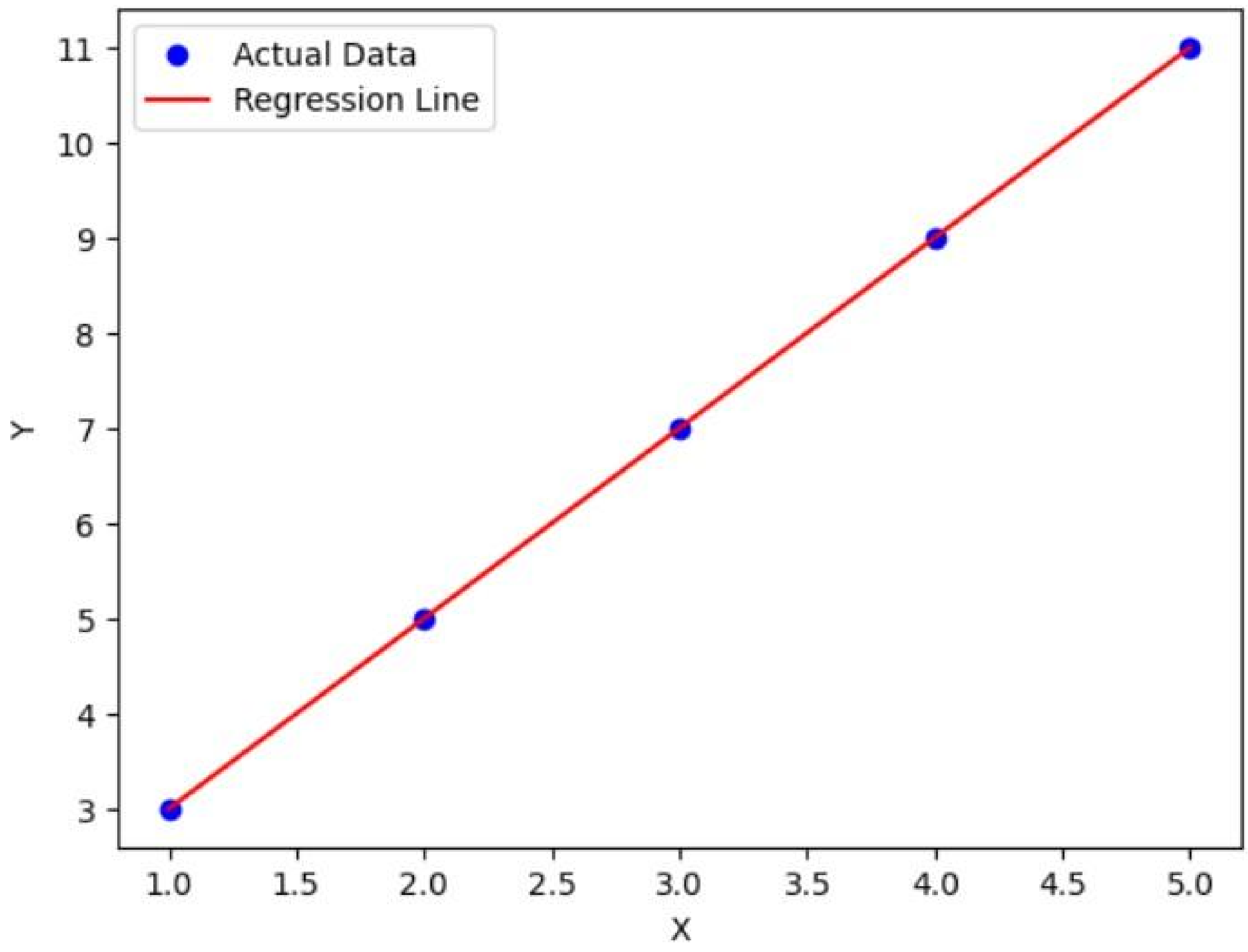
Step-4: Evaluate the performance of the trained model using the testing dataset

Step-5: Visualize the relationship between the two variables and the linear regression line

PROGRAM:

```
import numpy as np

import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4, 5])  # Independent variable

y = np.array([3, 5, 7, 9, 11])  # Dependent variable

# Calculate the slope (m) and intercept (b) of the regression line

m, b = np.polyfit(x, y, 1)


# Generate predicted y values

y_pred = m * x + b


# Plot the original data and the regression line

plt.scatter(x, y, color='blue', label='Actual Data')

plt.plot(x, y_pred, color='red', label='Regression Line')
plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()

# Analyze the relationship


correlation = np.corrcoef(x, y)[0, 1]  # Correlation coefficient

r_squared = correlation ** 2 # Coefficient of determination

print(f"Correlation coefficient: {correlation}")

print(f"Coefficient of determination (R-squared): {r_squared}")
```

OUTPUT:



```
Correlation coefficient: 0.9999999999999999
Coefficient of determination (R-squared): 0.9999999999999998
```

RESULT:

The implementation of simple linear regression between two variables using Python allows us to analyze the relationship between the variables and make predictions based on the observed data. By fitting a linear regression model, we can determine the coefficient and intercept that describe the linear relationship and the graph is displayed

CHURN DATA PREDICTION USING MACHINE LEARNING MODEL

EX NO: 8

DATE: 31.03.2023

AIM:

To build machine learning models to predict customer churn or attrition based on historical customer data and relevant features

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Clean and pre-process the collected data to prepare it for model training

Step-3: Analyze the data and engineer additional features that capture customer behavior, usage patterns, or other relevant information

Step-4: Select an appropriate machine learning algorithm for churn prediction

Step-5: Display the churn prediction

PROGRAM:

```
import pandas as pd

from sklearn.model_selection import train_test_split

import numpy as np

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

X = pd.get_dummies(df.drop(['Churn', 'Customer ID'], axis=1))

y = df['Churn'].apply(lambda x: 1 if x=='Yes' else 0)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=.2)

from tensorflow.keras.models import Sequential, load_model

from tensorflow.keras.layers import Dense

from sklearn.metrics import accuracy_score
```

```
model = Sequential()

model.add(Dense(units=32, activation='relu',
input_dim=len(X_train.columns)))

model.add(Dense(units=64, activation='relu'))

model.add(Dense(units=1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='sgd',
metrics='accuracy')

X_train = np.asarray(X_train).astype(np.float32)

y_train = np.asarray(y_train).astype(np.float32)

model.fit(X_train, y_train, epochs=50, batch_size=32)

X_test = np.asarray(X_test).astype(np.float32)

y_hat = model.predict(X_test)

y_hat = [0 if val < 0.5 else 1 for val in y_hat]

print(accuracy_score(y_test, y_hat))
```

OUTPUT:

```
Epoch 1/50
177/177 [==============================] - 1s 3ms/step - loss: 0.5058 - accuracy: 0.754
9
Epoch 2/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4848 - accuracy: 0.773
7
Epoch 3/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4778 - accuracy: 0.782
1
Epoch 4/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4733 - accuracy: 0.779
6
Epoch 5/50
177/177 [==============================] - 0s 2ms/step - loss: 0.4702 - accuracy: 0.784
7
Epoch 6/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4683 - accuracy: 0.783
3
Epoch 7/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4651 - accuracy: 0.784
6
Epoch 8/50
```

```
177/177 [==============================] - 1s 4ms/step - loss: 0.4621 - accuracy: 0.785
1
Epoch 9/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4674 - accuracy: 0.780
1
Epoch 10/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4630 - accuracy: 0.781
7
Epoch 11/50
177/177 [==============================] - 0s 2ms/step - loss: 0.4582 - accuracy: 0.783
5
Epoch 12/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4597 - accuracy: 0.786
2
Epoch 13/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4565 - accuracy: 0.785
3
Epoch 14/50
177/177 [==============================] - 0s 2ms/step - loss: 0.4585 - accuracy: 0.786
9
Epoch 15/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4550 - accuracy: 0.785
6
Epoch 16/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4523 - accuracy: 0.786
5
Epoch 17/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4513 - accuracy: 0.790
2
Epoch 18/50
177/177 [==============================] - 0s 2ms/step - loss: 0.4512 - accuracy: 0.786
3
Epoch 19/50
177/177 [==============================] - 0s 2ms/step - loss: 0.4512 - accuracy: 0.787
0
Epoch 20/50
177/177 [==============================] - 0s 2ms/step - loss: 0.4536 - accuracy: 0.785
8
Epoch 21/50
177/177 [==============================] - 0s 2ms/step - loss: 0.4516 - accuracy: 0.783
7
Epoch 22/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4512 - accuracy: 0.786
0
Epoch 23/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4494 - accuracy: 0.784
6
Epoch 24/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4482 - accuracy: 0.785
8
Epoch 25/50
177/177 [==============================] - 0s 2ms/step - loss: 0.4467 - accuracy: 0.789
4
Epoch 26/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4491 - accuracy: 0.788
1
Epoch 27/50
177/177 [==============================] - 0s 2ms/step - loss: 0.4556 - accuracy: 0.783
3
Epoch 28/50
```

```
177/177 [==============================] - 1s 3ms/step - loss: 0.4496 - accuracy: 0.786
2
Epoch 29/50
177/177 [==============================] - 0s 2ms/step - loss: 0.4451 - accuracy: 0.788
6
Epoch 30/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4473 - accuracy: 0.784
9
Epoch 31/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4480 - accuracy: 0.786
7
Epoch 32/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4457 - accuracy: 0.784
7
Epoch 33/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4452 - accuracy: 0.785
6
Epoch 34/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4452 - accuracy: 0.786
3
Epoch 35/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4436 - accuracy: 0.789
9
Epoch 36/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4453 - accuracy: 0.785
6
Epoch 37/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4431 - accuracy: 0.783
9
Epoch 38/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4407 - accuracy: 0.790
1
Epoch 39/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4409 - accuracy: 0.789
0
Epoch 40/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4396 - accuracy: 0.791
1
Epoch 41/50
177/177 [==============================] - 1s 3ms/step - loss: 0.4436 - accuracy: 0.789
7
Epoch 42/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4427 - accuracy: 0.790
1
Epoch 43/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4403 - accuracy: 0.792
7
Epoch 44/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4429 - accuracy: 0.788
5
Epoch 45/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4434 - accuracy: 0.793
6
Epoch 46/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4399 - accuracy: 0.787
9
Epoch 47/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4402 - accuracy: 0.790
8
Epoch 48/50
```

```
177/177 [==============================] - 0s 3ms/step - loss: 0.4388 - accuracy: 0.794
5
Epoch 49/50
177/177 [==============================] - 0s 3ms/step - loss: 0.4371 - accuracy: 0.795
6

Epoch 50/50
177/177 [==============================] - 0s 2ms/step - loss: 0.4390 - accuracy: 0.793
1
45/45 [==============================] - 0s 2ms/step

0.7856635911994322
```

RESULT:

The implementation of a machine learning model for churn data prediction using Python enables us to accurately identify customers at risk of churning. Overall, churn prediction using machine learning empowers businesses to proactively address customer churn and improve customer retention strategies.

MEETING VIDEO SUMMARIZATION

EX NO: 9

DATE: 07.04.2023

AIM:

To develop a python code for Summarizing the organization's meetings using Deep Learning

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Apply automatic speech recognition (ASR) techniques to convert the speech in the meeting video into text

Step-3: Utilize text summarization techniques to generate summaries of the transcribed text.

Step-4: Display the summarized text

PROGRAM:

```
from transformers import pipeline

from youtube_transcript_api import YouTubeTranscriptApi

#You Can Add Your Meeting here Instead of a Youtube Link

youtube_video =
"https://www.youtube.com/watch?v=QpBTM0GO6xI&pp=ygUJZ29vZ2xlIGlv"

video_id = youtube_video.split("=")[1]

YouTubeTranscriptApi.get_transcript(video_id)

transcript = YouTubeTranscriptApi.get_transcript(video_id)

result = ""

for i in transcript:

    result += ' ' + i['text']

#print(result)

print(len(result))

summarizer = pipeline('summarization')
```

```
num_iters = int(len(result)/1000)

summarized_text = []

for i in range(0, num_iters + 1):

  start = 0

  start = i * 1000

  end = (i + 1) * 1000

  #print("input text \n" + result[start:end])

  out = summarizer(result[start:end])

  out = out[0]

  out = out['summary_text']

  #print("Summarized text\n"+out)

  summarized_text.append(out)

len(str(summarized_text))

print(str(summarized_text)[1:-1])
```

OUTPUT:

```
" AI has been applying AI to make products radically more helpful for a while .
With generative AI, we're taking the next step. Magic Editor recreates parts of
the bench and balloons that were not captured in the original shot. As a finishi
ng touch, you can punch up the sky. It's truly magical. Imagine if you could see
your whole trip in advance with Immersive View for the first time .", " PaLM 2 i
s highly-capable, but it shines when fine-tuned on domain-specific knowledge . B
ard's math, reasoning, and reasoning skills made a huge leap forward, underpinni
ng its ability to help developers with programming . We are removing the wait li
st and opening up Bard to over 180 countries and territories .", " As you collab
orate with Bard, you'll be able to tap into services from Google and extensions
with partners to help you do things never before possible . Starting next month,
it will be available to beta users with six more generative AI features across W
orkspace . Sidekick instantly reads and processes the document and offers some r
eally great suggestions .", " There's an AI-powered snapshot that quickly gives
you the lay of the land on a topic . Tapping any of these options will bring you
into our brand new conversational mode . If you're in the US, you can join the w
aitlist today by tapping the Labs icon in the latest Google app or Chrome deskto
```

```
p .", " Project Tailwind aims to create a personalized and private AI model that
has expertise in the information that you give to it . It's important to celebra
te the incredible progress in AI and the immense potential that it has for peopl
e in society everywhere, we must also recognize that it's an emerging technology
.", " We're introducing Unknown Tracker Alerts to help you find your phone . We'
re combining Androi with Androio to improve our Find My Device experience . We w
ill ensure that every one of our AI-generated images has metadata and markup in
the original file to give you context around the world .", " Google's new genera
tive-AI wallpapers, you choose what inspires you, and then we create a beautiful
wallpaper to fit your vision . We're using Google's text-to-image diffusion mode
ls to generate completely new and original wallpapers . The new Pixel 7a, Google
's Pixel Fold, combines Tensor G2, Android innovation and AI for an incredible p
hone that unfolds into an incredible compact tablet .", " Google's Pixel Tablet
is the only tablet engineered by Google and designed specifically to be helpful
in your hand and in the place they are used the most, the home . The shift with
AI is as big as they come, says Google's Sundar PICHAI . Google's developer comm
unity is key to unlocking the enormous opportunities ahead ."
```

RESULT:

The generated video summary provides a concise overview of the meeting, making it easier for participants to review and extract relevant information. However, meeting video summarization is a complex task that requires careful handling of audio and visual data, as well as accurate speech recognition and text summarization algorithms.

SENTIMENT ANALYSIS

EX NO: 10

DATE: 21.04.2023

AIM:

To create a deep learning model for sentimental analysis of social media post

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Load the dataset containing text data for sentiment analysis and Ensure that the data is labeled with sentiment labels (positive, negative, neutral).

Step-3: Train a sentiment analysis model using a supervised learning algorithm

Step-4: Use the trained model to predict sentiment on new, unseen text data.

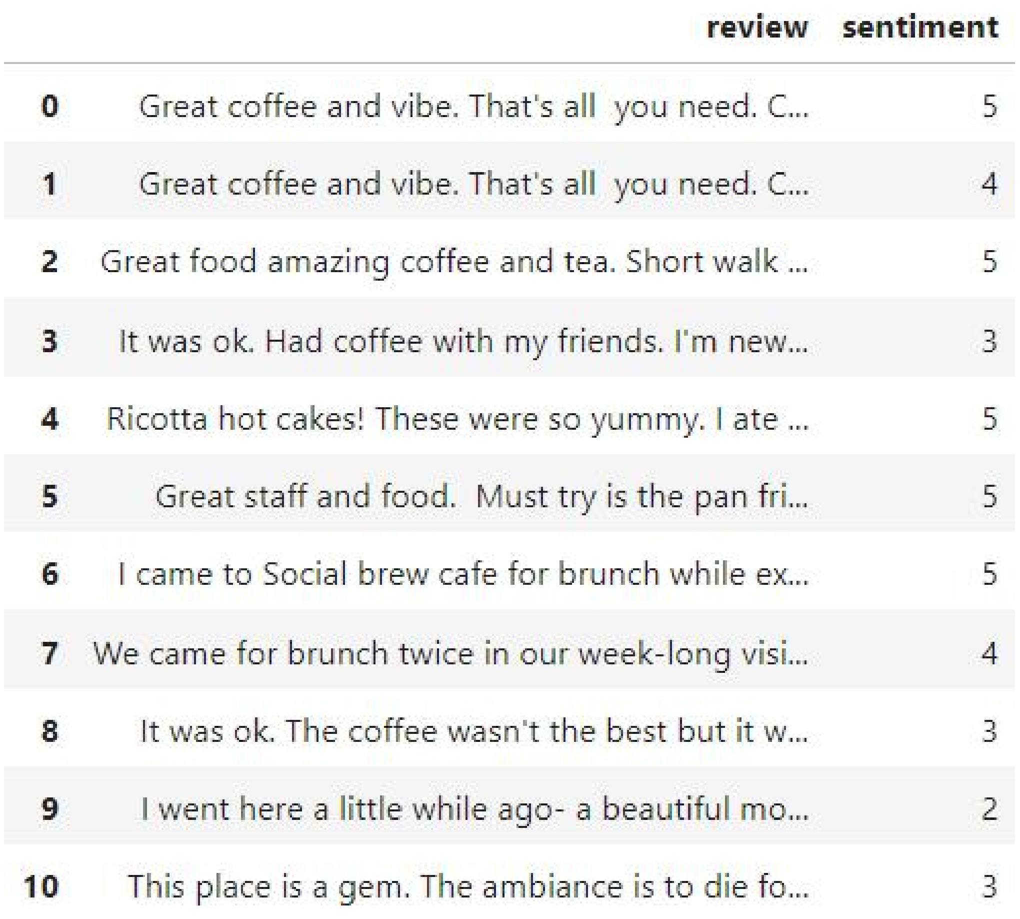
Step-5: Visualize the sentiment analysis results using plots or charts

PROGRAM:

```
from transformers import AutoTokenizer,
AutoModelForSequenceClassification

import torch

import requests

from bs4 import BeautifulSoup

import re

tokenizer = AutoTokenizer.from_pretrained('nlptown/bert-base-
multilingual-uncased-sentiment')


model =
AutoModelForSequenceClassification.from_pretrained('nlptown/bert-
base-multilingual-uncased-sentiment')

tokens = tokenizer.encode('It was good but couldve been better.
Great', return_tensors='pt')
```

```
result = model(tokens)

int(torch.argmax(result.logits))+1

r = requests.get('https://www.yelp.com/biz/social-brew-cafe-
pyrmont')

soup = BeautifulSoup(r.text, 'html.parser')

regex = re.compile('.*comment.*')

results = soup.find_all('p', {'class':regex})

reviews = [result.text for result in results]

import numpy as np

import pandas as pd

df = pd.DataFrame(np.array(reviews), columns=['review'])

df['review'].iloc[0]

def sentiment_score(review):

    tokens = tokenizer.encode(review, return_tensors='pt')

    result = model(tokens)

    return int(torch.argmax(result.logits))+1


sentiment_score(df['review'].iloc[1])

df['sentiment'] = df['review'].apply(lambda x:
sentiment_score(x[:512]))

specific = df['review'].iloc[3]

print(sentiment_score(specific))
```

OUTPUT:



	review	sentiment
0	Great coffee and vibe. That's all you need. C...	5
1	Great coffee and vibe. That's all you need. C...	4
2	Great food amazing coffee and tea. Short walk ...	5
3	It was ok. Had coffee with my friends. I'm new...	3
4	Ricotta hot cakes! These were so yummy. I ate ...	5
5	Great staff and food. Must try is the pan fri...	5
6	I came to Social brew cafe for brunch while ex...	5
7	We came for brunch twice in our week-long visi...	4
8	It was ok. The coffee wasn't the best but it w...	3
9	I went here a little while ago- a beautiful mo...	2
10	This place is a gem. The ambiance is to die fo...	3

```
"It was ok. Had coffee with my friends. I'm new in the area, still need to
discover new places."


    3
```

RESULT:

The results provide valuable insights into the sentiment of the text, allowing us to understand public opinion, customer feedback, or social media sentiment.

FRAUD DETECTION

EX NO: 11

DATE: 28.04.2023

AIM:

To implement Cyber Security Detection framework using Machine Learning

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Clean and pre-process the collected data to prepare it for model training

Step-3: Train a fraud detection model using a suitable machine learning algorithm

Step-4: Deploy the Model

PROGRAM:

```
import pandas as pd

from sklearn.metrics import accuracy_score

from sklearn.metrics import f1_score

from sklearn.model_selection import train_test_split

from xgboost import XGBClassifier

from sklearn.preprocessing import LabelEncoder

data = pd.read_csv("creditCard.csv")

#data = data.dropna()

data['Class'].isna().any()

total_transactions = len(data)

normal = len(data[data.Class == 0])

fraudulent = len(data[data.Class == 1])

fraud_percentage = round(fraudulent / normal * 100, 2)

print(f'Total number of Transactions are {total_transactions}')

print(f'Number of Normal Transactions are {normal}')
print(f'Number of fraudulent Transactions are {fraudulent}')

print(f'Percentage of fraud Transactions is {fraud_percentage *
100}%')

X = data.drop('Class', axis=1).values

y = data["Class"].values

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=1)

xgb = XGBClassifier()

xgb.fit(X_train, y_train)

xgb_yhat = xgb.predict(X_test)


print(f'Accuracy score of the XGBoost model is
{(accuracy_score(y_test, xgb_yhat) * 100):.2f} %')

print(f'F1 score of the XGBoost model is {(f1_score(y_test,
xgb_yhat) * 100):.2f} %')
```

OUTPUT:

```
Total number of Transactions are 284807
Number of Normal Transactions are 284315
Number of fraudulent Transactions are 492
Percentage of fraud Transactions is 17.0%
Accuracy score of the XGBoost model is 99.96 %
F1 score of the XGBoost model is 85.00 %
```

RESULT:

The implementation of fraud detection using Python provides a means to identify and prevent fraudulent activities within a system. Fraud detection using Python serves as a powerful tool in safeguarding against fraudulent activities and minimizing financial losses.

FAKE PROFILE DETECTION

EX NO: 12

DATE: 05.05.2023

AIM:

To solve the Fake Profile Detection problem using Machine Learning

PROCEDURE:

Step-1: Import the Required Libraries and Modules

Step-2: Collect a dataset of genuine and fake ID images for training and testing.

Step-3: Train a machine learning model using the extracted features

Step-4: integrate the fake ID detection system with existing ID verification processes or systems

Step-5: Deploy the model

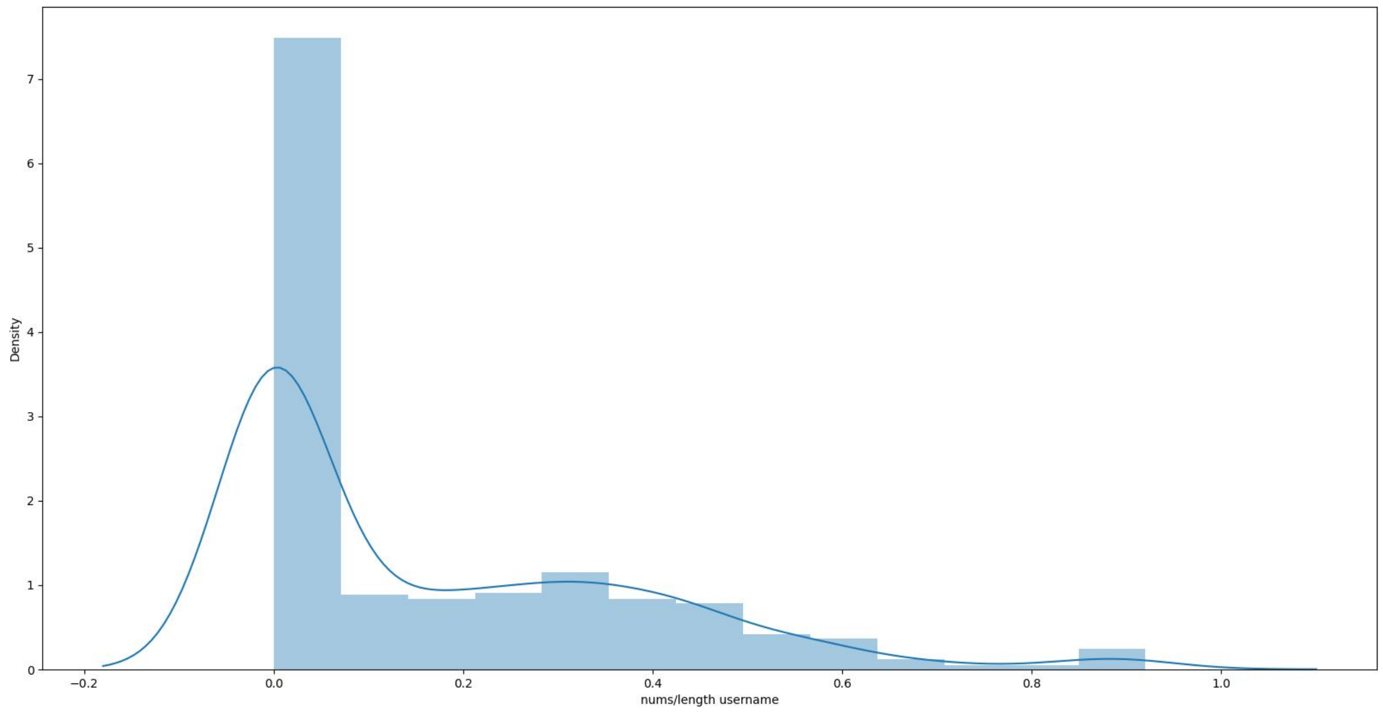
PROGRAM:

```
import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns


import tensorflow as tf

from tensorflow import keras

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

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.metrics import Accuracy


from sklearn import metrics

from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import
classification_report,accuracy_score,roc_curve,confusion_matrix


# Load the training dataset
instagram_df_train=pd.read_csv('train.csv')
instagram_df_train


# Load the testing data
instagram_df_test=pd.read_csv('test.csv')
instagram_df_test


# Visualize the data
sns.countplot(instagram_df_train['fake'])
plt.show()


# Visualize the private column data
sns.countplot(instagram_df_train['private'])
plt.show()


# Visualize the data
plt.figure(figsize = (20, 10))
sns.distplot(instagram_df_train['nums/length username'])
plt.show()


# Correlation plot
plt.figure(figsize=(20, 20))
```

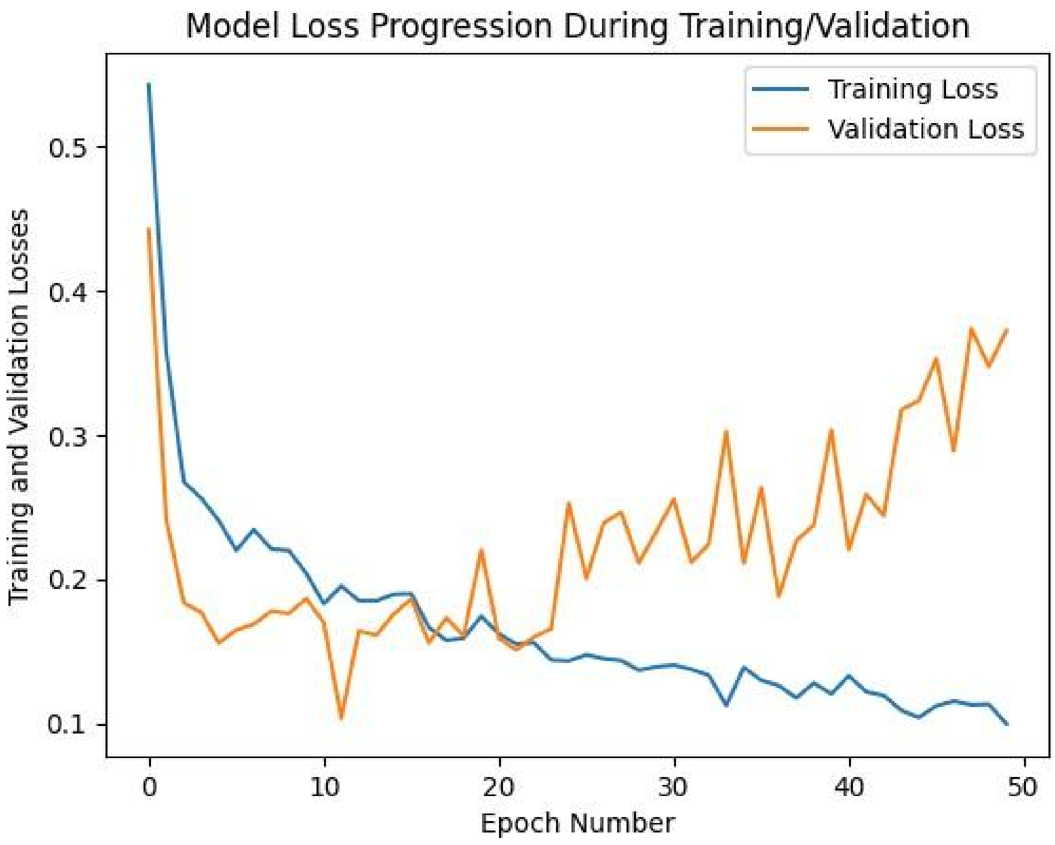
```
cm = instagram_df_train.corr()

ax = plt.subplot()

sns.heatmap(cm, annot = True, ax = ax)

plt.show()


# Training and testing dataset (inputs)

X_train = instagram_df_train.drop(columns = ['fake'])

X_test = instagram_df_test.drop(columns = ['fake'])

X_train


# Training and testing dataset (Outputs)

y_train = instagram_df_train['fake']

y_test = instagram_df_test['fake']

y_train


# Scale the data before training the model

from sklearn.preprocessing import StandardScaler, MinMaxScaler


scaler_x = StandardScaler()

X_train = scaler_x.fit_transform(X_train)

X_test = scaler_x.transform(X_test)


y_train = tf.keras.utils.to_categorical(y_train, num_classes = 2)

y_test = tf.keras.utils.to_categorical(y_test, num_classes = 2)


import tensorflow.keras
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout


model = Sequential()
model.add(Dense(50, input_dim=11, activation='relu'))
model.add(Dense(150, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(150, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(25, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(2,activation='softmax'))


model.summary()


model.compile(optimizer = 'adam', loss =
'categorical_crossentropy', metrics = ['accuracy'])
epochs_hist = model.fit(X_train, y_train, epochs = 50,  verbose =1,
validation_split = 0.1)


print(epochs_hist.history.keys())


plt.plot(epochs_hist.history['loss'])
plt.plot(epochs_hist.history['val_loss'])


plt.title('Model Loss Progression During Training/Validation')
plt.ylabel('Training and Validation Losses')
plt.xlabel('Epoch Number')

plt.legend(['Training Loss', 'Validation Loss'])

plt.show()


predicted = model.predict(X_test)


predicted_value = []

test = []

for i in predicted:

    predicted_value.append(np.argmax(i))


for i in y_test:

    test.append(np.argmax(i))


print(classification_report(test, predicted_value))


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

cm=confusion_matrix(test, predicted_value)

sns.heatmap(cm, annot=True)

plt.show()
```

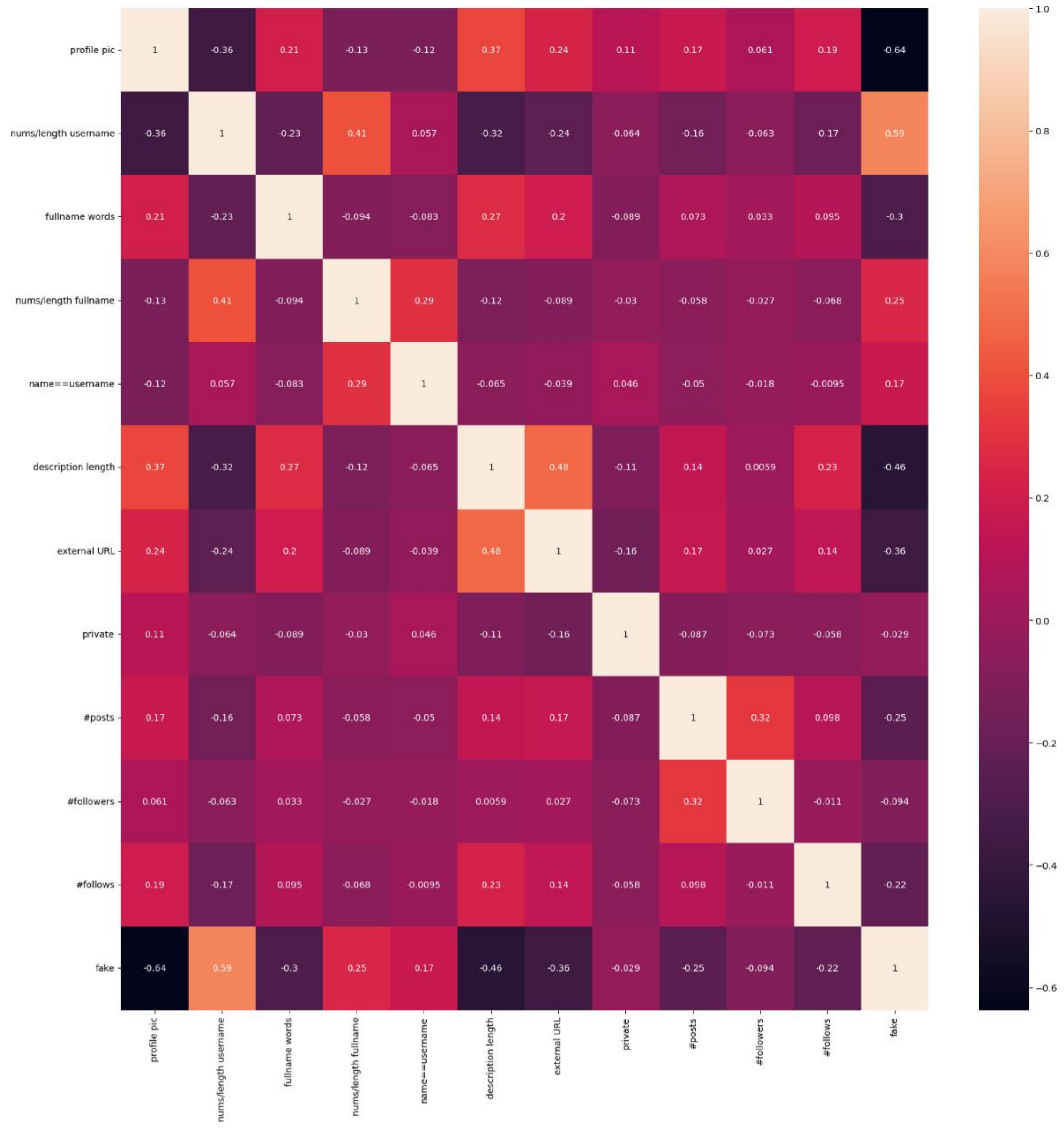
OUTPUT:



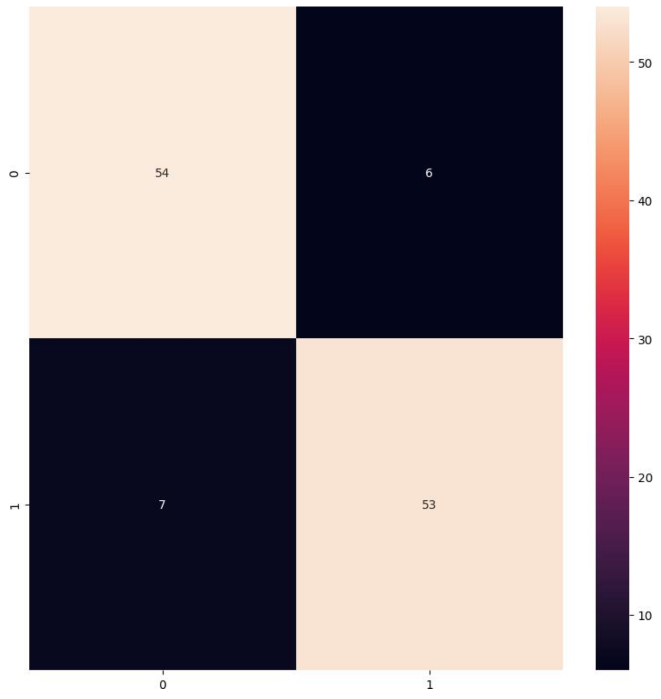
Density
nums/length username
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7



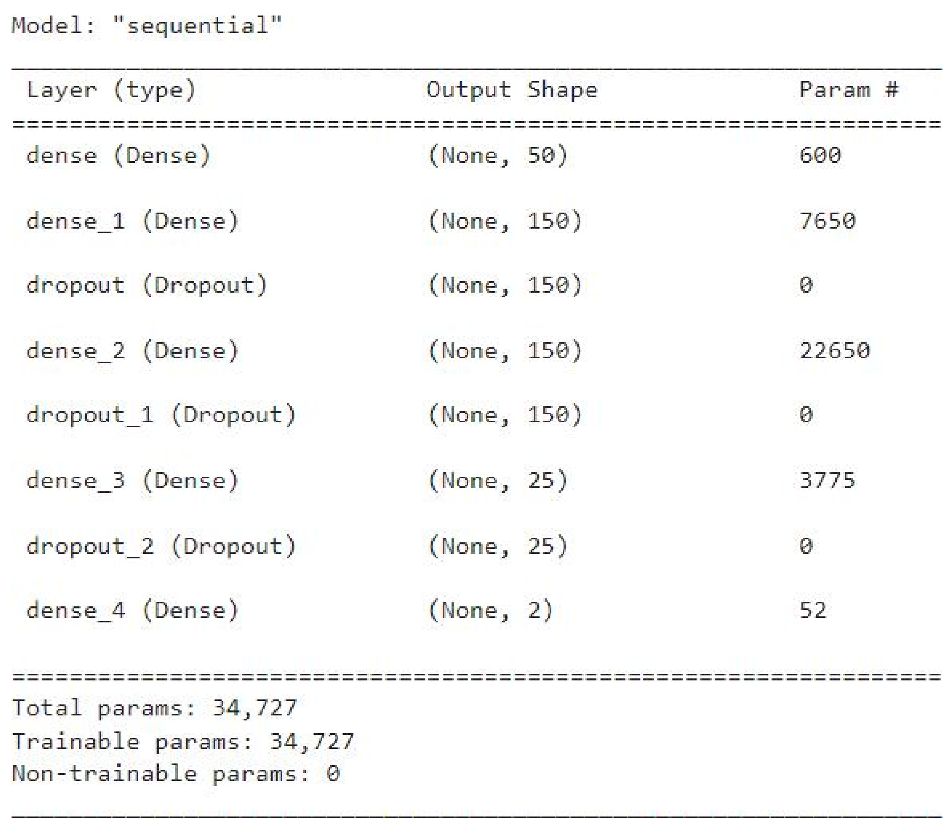
Model Loss Progression During Training/Validation
Training Loss
Validation Loss
Training and Validation Losses
Epoch Number
0.1
0.2
0.3
0.4
0.5
0
10
20
30
40
50



profile pic
nums/length username
fullname words
nums/length fullname
name==username
description length
external URL
private
#posts
#followers
#follows
fake
profile pic: 1, -0.36, 0.21, -0.13, -0.12, 0.37, 0.24, 0.11, 0.17, 0.061, 0.19, -0.64
nums/length username: -0.36, 1, -0.23, 0.41, 0.057, -0.32, -0.24, -0.064, -0.16, -0.063, -0.17, 0.59
fullname words: 0.21, -0.23, 1, -0.094, -0.083, 0.27, 0.2, -0.089, 0.073, 0.033, 0.095, -0.3
nums/length fullname: -0.13, 0.41, -0.094, 1, 0.29, -0.12, -0.089, -0.03, -0.058, -0.027, -0.068, 0.25
name==username: -0.12, 0.057, -0.083, 0.29, 1, -0.065, -0.039, 0.046, -0.05, -0.018, -0.0095, 0.17
description length: 0.37, -0.32, 0.27, -0.12, -0.065, 1, 0.48, -0.11, 0.14, 0.0059, 0.23, -0.46
external URL: 0.24, -0.24, 0.2, -0.089, -0.039, 0.48, 1, -0.16, 0.17, 0.027, 0.14, -0.36
private: 0.11, -0.064, -0.089, -0.03, 0.046, -0.11, -0.16, 1, -0.087, -0.073, -0.058, -0.029
#posts: 0.17, -0.16, 0.073, -0.058, -0.05, 0.14, 0.17, -0.087, 1, 0.32, 0.098, -0.25
#followers: 0.061, -0.063, 0.033, -0.027, -0.018, 0.0059, 0.027, -0.073, 0.32, 1, -0.011, -0.094
#follows: 0.19, -0.17, 0.095, -0.068, -0.0095, 0.23, 0.14, -0.058, 0.098, -0.011, 1, -0.22
fake: -0.64, 0.59, -0.3, 0.25, 0.17, -0.46, -0.36, -0.029, -0.25, -0.094, -0.22, 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6



0
1
54
6
7
53
50
40
30
20
10



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                600

 dense_1 (Dense)             (None, 150)               7650

 dropout (Dropout)           (None, 150)               0

 dense_2 (Dense)             (None, 150)               22650

 dropout_1 (Dropout)         (None, 150)               0

 dense_3 (Dense)             (None, 25)                3775

 dropout_2 (Dropout)         (None, 25)                0

 dense_4 (Dense)             (None, 2)                 52

=================================================================
Total params: 34,727
Trainable params: 34,727
Non-trainable params: 0
_________________________________________________________________
```

```
              precision    recall  f1-score   support

           0       0.89      0.90      0.89        60
           1       0.90      0.88      0.89        60

    accuracy                           0.89       120
   macro avg       0.89      0.89      0.89       120
weighted avg       0.89      0.89      0.89       120
```

RESULT:

The implementation of fake profile detection using Python provides a means to identify and mitigate the presence of fake or fraudulent user profiles within a system. fake profile detection using Python serves as a valuable tool in maintaining the trustworthiness and security of online platforms and minimizing potential harm caused by fake profiles.