**ICT 305 Topics in IT**

**Assessment 2: Phishing Email Detection Using Deep Learning**

**Objective**

The objective of this lab is to provide hands-on experience in detecting phishing emails using deep learning techniques. Students will use a dataset from Kaggle, preprocess the data, and apply a deep learning model to classify emails as phishing or legitimate.

**Dataset**

The dataset used in this lab can be found on Kaggle:

* [Phishing Email Detection Using Deep Learning](https://www.kaggle.com/code/kirollosashraf/phishing-email-detection-using-deep-learning)​ ([Kaggle](https://www.kaggle.com/code/kirollosashraf/phishing-email-detection-using-deep-learning#:~:text=URL%3A%20https%3A%2F%2Fwww.kaggle.com%2Fcode%2Fkirollosashraf%2Fphishing))​
* [Phishing E-Mail Detection](https://www.kaggle.com/code/zekielkatmis/phishing-e-mail-detection)​ ([Kaggle](https://www.kaggle.com/code/zekielkatmis/phishing-e-mail-detection#:~:text=URL%3A%20https%3A%2F%2Fwww.kaggle.com%2Fcode%2Fzekielkatmis%2Fphishing))​

**Steps**

**Step 1: Load the Dataset**

Download the dataset from Kaggle and load it into your environment.

|  |
| --- |
| !pip install kaggle  !mkdir -p ~/.kaggle  !cp kaggle.json ~/.kaggle/  !chmod 600 ~/.kaggle/kaggle.json  !kaggle datasets download -d subhajournal/phishingemails  !unzip /content/phishingemails.zip |
| import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  import numpy as np  import plotly.express as px  from sklearn.feature\_extraction.text import TfidfVectorizer,CountVectorizer  from sklearn.linear\_model import LogisticRegression  from sklearn.tree import DecisionTreeClassifier  from sklearn.ensemble import RandomForestClassifier  from sklearn.neural\_network import MLPClassifier  from sklearn.model\_selection import train\_test\_split  from tensorflow.keras.preprocessing.text import Tokenizer  from tensorflow.keras.layers import Embedding,GRU,LSTM,Bidirectional,SimpleRNN  from tensorflow.keras.utils import pad\_sequences  from sklearn.preprocessing import LabelEncoder  from keras.models import Sequential  from keras.layers import Dense,Dropout  import tensorflow as tf  import warnings  warnings.filterwarnings('ignore') |
| df = pd.read\_csv("Phishing\_Email.csv")  df.head() |

**Step 2: Data Preprocessing**

Preprocess the data by cleaning and transforming it into a format suitable for machine learning models.

|  |
| --- |
| df.isnull().sum()  df.drop(["Unnamed: 0"],axis=1,inplace=True)  df.dropna(inplace=True,axis=0)  df.drop\_duplicates(inplace=True)  # Create the bar chart  fig = px.bar(df['Email Type'].value\_counts(), x=df['Email Type'].value\_counts().index, y=df['Email Type'].value\_counts().values,               color=['blue', 'red'], labels={'x': 'Category', 'y': 'Count'},               title="Categorical Distribution")  # Show the plot  fig.show()  # Create the pie chart  fig\_pie = px.pie(df['Email Type'].value\_counts(), names=df['Email Type'].value\_counts().index,                   values=df['Email Type'].value\_counts().values, title="Categorical Distribution")  # Show the pie chart  fig\_pie.show()  le = LabelEncoder()  df["Email Type"] = le.fit\_transform(df["Email Type"])  import re  def preprocess\_text(text):      # Remove hyperlinks      text = re.sub(r'http\S+', '', text)      # Remove punctuations      text = re.sub(r'[^\w\s]', '', text)      # Convert to lowercase      text = text.lower()      # Remove extra spaces      text = re.sub(r'\s+', ' ', text).strip()      return text  # Apply the preprocess\_text function to the specified column in the DataFrame  df["Email Text"] =df["Email Text"].apply(preprocess\_text)  from wordcloud import WordCloud  #combine all rows into a single string  all\_mails = " ".join(df['Email Text'])  #create a wordcloud object  word\_cloud = WordCloud(stopwords="english",width=800,height=400,background\_color='white').generate(all\_mails)  plt.figure(figsize=(10,6))  plt.imshow(word\_cloud,interpolation='bilinear')  plt.axis("off")  plt.show()  #combine all rows into a single string  all\_mails = " ".join(df['Email Text'])  #create a wordcloud object  word\_cloud = WordCloud(width=800,height=400,background\_color='white',max\_words=10000).generate(all\_mails)  plt.figure(figsize=(10,6))  plt.imshow(word\_cloud,interpolation='bilinear')  plt.axis("off")  plt.show()  tf = TfidfVectorizer(stop\_words="english",max\_features=10000) #dimension reduction  feature\_x = tf.fit\_transform(df["Email Text"]).toarray()  x\_train,x\_test,y\_train,y\_test = train\_test\_split(feature\_x,y\_tf,train\_size=0.8,random\_state=0) |

**Step 3: Train a Deep Learning Model**

Use a simple deep learning model to train on the dataset.

|  |
| --- |
| #naive bayes works with condtional probability  from sklearn.naive\_bayes import MultinomialNB  nb = MultinomialNB()  nb.fit(x\_train,y\_train) |

**Step 4: Evaluate the Model**

Evaluate the model's performance on the test set.

|  |
| --- |
| from sklearn.metrics import accuracy\_score,f1\_score,classification\_report,ConfusionMatrixDisplay,confusion\_matrix  pred\_nav = nb.predict(x\_test)  # Checking the performance  print(f"accuracy from native bayes: {accuracy\_score(y\_test,pred\_nav)\*100:.2f} %")  print(f"f1 score from naive bayes: {f1\_score(y\_test,pred\_nav)\*100:.2f} %")  print("classification report :\n\n",classification\_report(y\_test,pred\_nav))  #confusion matrix  clf\_nav = confusion\_matrix(y\_test,pred\_nav)  cx\_ = ConfusionMatrixDisplay(clf\_nav,display\_labels=['pishing\_mail','safe\_mail']).plot()  plt.show() |

[Full Code](https://colab.research.google.com/drive/1yS9uQjexPc1w-KCZPz4Mwlz-QQBANvHg?usp=sharing)

**Step 5: Implement Real-Time Phishing Detection**

Flask Application with Corrected Preprocessing

Here’s the complete Flask application with the corrected loading and transformation process:

*Step 1: HTML Templates with Bootstrap*

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Phishing Email Detection</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

<style>

body {

background-color: #f8f9fa;

}

.container {

margin-top: 50px;

}

.card {

padding: 20px;

}

textarea {

resize: none;

}

</style>

</head>

<body>

<div class="container">

<div class="card">

<h2 class="card-title text-center">Phishing Email Detection</h2>

<form action="/predict" method="post">

<div class="form-group">

<label for="email\_content">Enter Email Text:</label>

<textarea class="form-control" id="email\_content" name="email\_content" rows="10" placeholder="Enter email text here..."></textarea>

</div>

<button type="submit" class="btn btn-primary btn-block">Detect</button>

</form>

</div>

</div>

</body>

</html>

**result.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Phishing Email Detection - Result</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

<style>

body {

background-color: #f8f9fa;

}

.container {

margin-top: 50px;

}

.card {

padding: 20px;

}

</style>

</head>

<body>

<div class="container">

<div class="card">

<h2 class="card-title text-center">Phishing Email Detection Result</h2>

<p class="text-center">The email is classified as: <strong>{{ prediction }}</strong></p>

<a href="/" class="btn btn-secondary btn-block">Back to Home</a>

</div>

</div>

</body>

</html>

Step 2: Flask Application (app.py)

Run your Flask application by executing the following command in your terminal:

python app.py

Navigate to http://127.0.0.1:5000/ in your web browser to access the application.

from flask import Flask, request, render\_template

import joblib

import re

from sklearn.feature\_extraction.text import TfidfVectorizer

# Load the trained model and the TF-IDF vectorizer

model = joblib.load('email\_spam\_nb.pkl')

vectorizer = joblib.load('tfidf\_vectorizer.pkl')

app = Flask(\_\_name\_\_)

def preprocess\_email(text):

# Remove hyperlinks

text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)

# Remove punctuation

text = re.sub(r'[^\w\s]', '', text)

# Convert to lowercase

text = text.lower()

# Remove extra spaces

text = re.sub(r'\s+', ' ', text).strip()

return text

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

if request.method == 'POST':

email\_content = request.form['email\_content']

# Preprocess the email content

email\_content = preprocess\_email(email\_content)

# Transform the email content using the TF-IDF vectorizer

email\_tfidf = vectorizer.transform([email\_content])

# Predict using the trained model

prediction = model.predict(email\_tfidf)

result = 'Phishing' if prediction[0] == 1 else 'Safe'

return render\_template('result.html', prediction=result)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)