**OPTIMIZED PHISHING DETECTION THROUGH URL ANALYSIS BY A GRADIENT BOOSTING RNN ENSEBLE**

**CODING**

**Predict.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Phishing Detection</title>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">

    <style>

        body {

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

            background: linear-gradient(to bottom right, #e3f2fd, #bbdefb);

            display: flex;

            justify-content: center;

            align-items: center;

            min-height: 100vh;

            margin: 0;

        }

        .container1 {

            background: white;

            padding: 40px 30px;

            border-radius: 16px;

            box-shadow: 0 12px 24px rgba(0, 0, 0, 0.1);

            width: 100%;

            max-width: 460px;

            text-align: center;

        }

        h2 {

            color: #1a237e;

            font-size: 26px;

            margin-bottom: 10px;

        }

        label {

            font-size: 15px;

            color: #333;

            margin-bottom: 10px;

            display: block;

            text-align: left;

        }

        input[type="text"] {

            width: 80%;

            padding: 12px 14px;

            font-size: 15px;

            border: 1px solid #cfd8dc;

            border-radius: 8px;

            margin-bottom: 25px;

            outline: none;

            transition: border 0.3s ease-in-out;

        }

        input[type="text"]:focus {

            width: 90%;

            padding: 12px 14px;

            font-size: 16px;

            border: 1px solid #ccc;

            border-radius: 8px;

            box-sizing: border-box;

            margin-bottom: 20px;

        }

        button {

            background-color: #1976d2;

            color: white;

            padding: 12px;

            width: 50%;

            font-size: 16px;

            border: none;

            border-radius: 8px;

            cursor: pointer;

            transition: background 0.3s ease;

        }

        button:hover {

            background-color: #1565c0;

        }

    </style>

</head>

<body>

    <div class="container1">

        <h2>🔍 Phishing URL Detection</h2>

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

            <label for="url">Enter URL:</label>

            <input type="text" id="url" name="url" placeholder="https://example.com" required>

            <button type="submit">Check URL</button>

        </form>

    </div>

</body>

</html>

**train.py**

import pandas as pd

import numpy as np

import joblib

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

import xgboost as xgb

from catboost import CatBoostClassifier

import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Sequential

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

# Load dataset

df = pd.read\_csv("dataset/phishing\_data.csv", encoding="ISO-8859-1")

df["Label"] = df["Label"].map({"good": 0, "bad": 1})

X = df["URL"]

y = df["Label"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

vectorizer = TfidfVectorizer(stop\_words="english", max\_features=5000)

X\_train\_tfidf = vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = vectorizer.transform(X\_test)

xgb\_model = xgb.XGBClassifier(n\_estimators=100, max\_depth=5, learning\_rate=0.1, use\_label\_encoder=False, eval\_metric="logloss")

xgb\_model.fit(X\_train\_tfidf, y\_train)

cat\_model = CatBoostClassifier(iterations=100, depth=5, learning\_rate=0.1, verbose=0)

cat\_model.fit(X\_train\_tfidf, y\_train)

tokenizer = Tokenizer(num\_words=5000, oov\_token="<OOV>")

tokenizer.fit\_on\_texts(X\_train)

X\_train\_seq = tokenizer.texts\_to\_sequences(X\_train)

X\_test\_seq = tokenizer.texts\_to\_sequences(X\_test)

X\_train\_pad = pad\_sequences(X\_train\_seq, maxlen=100, padding="post")

X\_test\_pad = pad\_sequences(X\_test\_seq, maxlen=100, padding="post")

vocab\_size = 5000

embedding\_dim = 64

lstm\_model = Sequential([

    Embedding(vocab\_size, embedding\_dim, input\_length=100),

    LSTM(64, return\_sequences=True),

    Dropout(0.3),

    LSTM(32),

    Dense(1, activation="sigmoid")

])

lstm\_model.compile(loss="binary\_crossentropy", optimizer="adam", metrics=["accuracy"])

lstm\_model.fit(X\_train\_pad, y\_train, validation\_data=(X\_test\_pad, y\_test), epochs=5, batch\_size=32)

joblib.dump(vectorizer, "models/tfidf\_vectorizer.pkl")

joblib.dump(xgb\_model, "models/xgb\_model.pkl")

joblib.dump(cat\_model, "models/cat\_model.pkl")

joblib.dump(tokenizer, "models/tokenizer.pkl")

lstm\_model.save("models/lstm\_model.h5")

print("Models trained and saved successfully!")

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

sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm", fmt=".2f")

plt.title("Correlation Matrix of MFCC Features")

plt.tight\_layout()

plt.savefig("correlation\_matrix.png")

plt.show()

print("Correlation matrix saved as 'correlation\_matrix.png'")

**app.py**

from flask import Flask, render\_template, request

import numpy as np

import pickle

from tensorflow.keras.preprocessing.sequence import pad\_sequences

import joblib

import tensorflow as tf

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

# Load models and tools

vectorizer = joblib.load("models/tfidf\_vectorizer.pkl")

xgb\_model = joblib.load("models/xgb\_model.pkl")

cat\_model = joblib.load("models/cat\_model.pkl")

tokenizer = joblib.load("models/tokenizer.pkl")

lstm\_model = tf.keras.models.load\_model("models/lstm\_model.h5")

# Feature extraction function

def preprocess\_text(url):

    url\_tfidf = vectorizer.transform([url])

    url\_seq = tokenizer.texts\_to\_sequences([url])

    url\_pad = pad\_sequences(url\_seq, maxlen=100)

    return url\_tfidf, url\_pad

# Prediction + Threat Score

def predict\_url(url):

    url\_tfidf, url\_pad = preprocess\_text(url)

    xgb\_prob = xgb\_model.predict\_proba(url\_tfidf)[:, 1][0]

    cat\_prob = cat\_model.predict\_proba(url\_tfidf)[:, 1][0]

    lstm\_prob = lstm\_model.predict(url\_pad)[0][0]

    confidence\_scores = np.array([xgb\_prob, cat\_prob, lstm\_prob])

    weights = confidence\_scores / confidence\_scores.sum()

    ensemble\_prob = (weights[0] \* xgb\_prob) + (weights[1] \* cat\_prob) + (weights[2] \* lstm\_prob)

    ensemble\_pred = 1 if ensemble\_prob >= 0.5 else 0

    confidence\_percentage = round(ensemble\_prob \* 100, 2)

    return {

        "Final\_Ensemble\_Prediction": int(ensemble\_pred),

        "Confidence": confidence\_percentage

    }

# Home route

@app.route("/")

def home():

    return render\_template("predict.html")

# Prediction form handler

@app.route("/predict\_page", methods=["GET", "POST"])

def predict\_page():

    if request.method == "POST":

        url = request.form.get("url")

        if not url:

            return render\_template("result.html", url=url, prediction="Error: URL missing")

        prediction\_result = predict\_url(url)

        return render\_template(

            "result.html",

            url=url,

            ensemble=prediction\_result["Final\_Ensemble\_Prediction"],

            threat\_score=prediction\_result["Confidence"]

        )

    return render\_template("predict.html")

# Run the Flask app

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)

**result.html**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Phishing Detection</title>

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

  <style>

    body {

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      background: linear-gradient(to right, #e0f7fa, #ffffff);

      margin: 0;

      padding: 0;

      display: flex;

      flex-direction: column;

      align-items: center;

    }

    header {

      padding: 30px 0 10px;

      text-align: center;

    }

    h1 {

      font-size: 32px;

      font-weight: 700;

      color: #1b1b1b;

    }

    .result-container {

      background: #fff;

      padding: 30px 40px;

      border-radius: 16px;

      box-shadow: 0 8px 25px rgba(0, 0, 0, 0.1);

      max-width: 550px;

      width: 90%;

      text-align: center;

    }

    .result-container h2 {

      font-size: 24px;

      color: #2b2b2b;

      margin-bottom: 10px;

    }

    .url-text {

      color: #555;

      font-weight: 500;

      word-wrap: break-word;

      font-size: 15px;

      margin: 10px 0 20px;

    }

    .final-prediction {

      display: inline-block;

      padding: 10px 16px;

      font-size: 16px;

      font-weight: 600;

      border-radius: 8px;

      margin-bottom: 20px;

    }

    .phishing {

      background-color: #ffe5e5;

      color: #d32f2f;

      border: 1px solid #d32f2f;

    }

    .safe {

      background-color: #e5ffe5;

      color: #388e3c;

      border: 1px solid #388e3c;

    }

    .progress-container {

      width: 100%;

      background-color: #eee;

      border-radius: 10px;

      height: 25px;

      overflow: hidden;

      margin: 20px 0;

    }

    .progress-bar {

      height: 100%;

      background-color: #2196f3;

      text-align: center;

      color: white;

      line-height: 25px;

      font-weight: bold;

      transition: width 1s ease-in-out;

    }

    .gauge-label {

      font-weight: 600;

      font-size: 15px;

      color: #333;

      margin-top: 5px;

    }

    .back-button {

      background-color: #007BFF;

      color: white;

      text-decoration: none;

      padding: 10px 18px;

      font-size: 15px;

      border-radius: 6px;

      display: inline-flex;

      align-items: center;

      gap: 8px;

      transition: background 0.3s ease;

      margin-top: 20px;

    }

    .back-button:hover {

      background-color: #0056b3;

    }

    .back-button::before {

      content: "🔁";

    }

    @media screen and (max-width: 600px) {

      .result-container {

        padding: 20px;

      }

    }

  </style>

</head>

<body>

  <header>

    <h1>Phishing Detection</h1>

  </header>

  <div class="result-container">

    <h2>🔍 Prediction Result</h2>

    <p><strong>URL:</strong></p>

    <p class="url-text">{{ url }}</p>

    <div class="final-prediction {% if ensemble == 1 %}phishing{% else %}safe{% endif %}">

      {% if ensemble == 1 %} 🟥 Phishing {% else %} 🟩 Legitimate {% endif %}

    </div>

    <div class="progress-container">

      <div class="progress-bar" style="width: {{ threat\_score|float }}%;">

        {{ threat\_score }}%

      </div>

    </div>

    <div class="gauge-label">Confidence Score</div>

    <a href="/predict\_page" class="back-button">Try Again</a>

  </div>

</body>

</html>