SOURCE CODE:

```python
# app.py
import streamlit as st
import pandas as pd
import re
import os
import pickle
import base64
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# Configuration
st.set_page_config(page_title="📧 Email Vulnerability Tester", page_icon="🛡",
layout="wide")
st.title("🛡 Email Vulnerability & Phishing Detector")
st.markdown("Test the *security strength* of an email and detect phishing risk
using machine learning.")

# Cached sample data
@st.cache_data
def load_sample_data():
    return pd.DataFrame({
        'email_text': [
            "Dear customer, your account has been compromised. Click here to
reset: http://bit.ly/2YjK5d9",
            "Meeting scheduled for tomorrow at 2 PM. Please prepare the quarterly
report.",
            "URGENT: Your PayPal account has been suspended. Verify your
identity: paypal-secure-verify.com",
        ],
        'label': [1, 0, 1]
    })

# Preprocessing function
def preprocess_email(email):
    email = email.lower()
    email = re.sub(r'http\S+|www\S+|https\S+', ' url ', email)
    email = re.sub(r'[^\w\s]', ' ', email)
    email = re.sub(r'\s+', ' ', email).strip()
    stopwords = set([
        'a', 'an', 'the', 'and', 'or', 'but', 'if', 'because', 'as', 'what',
'which', 'this', 'that',
        'these', 'those', 'then', 'just', 'so', 'than', 'such', 'both',
'through', 'about', 'for',
        'is', 'of', 'while', 'during', 'to', 'from', 'in', 'on', 'by', 'with',
'at', 'you', 'your',
        'we', 'our'
    ])
    tokens = email.split()
    return ' '.join([t for t in tokens if t not in stopwords])

# Manual feature extraction
def extract_features(email):
    email = email.lower()
    return {
        'url_count': len(re.findall(r'http\S+|www\S+|https\S+', email)),
        'urgent_count': sum(w in email for w in ['urgent', 'immediate', 'now',
'alert', 'warning']),
        'financial_count': sum(w in email for w in ['bank', 'account', 'credit',
'debit', 'password', 'login', 'verify', 'payment']),
        'suspicious_domain': int(any(ext in email for ext in ['.xyz', '.info',
'.tk', '.pw', '.cc'])),
        'email_length': len(email)
    }

# Train model
def train_model(data):
    data['processed'] = data['email_text'].apply(preprocess_email)
    X_text = TfidfVectorizer(max_features=500)
    tfidf = X_text.fit_transform(data['processed'])
    tfidf_df = pd.DataFrame(tfidf.toarray(),
columns=X_text.get_feature_names_out())
    manual_df = pd.DataFrame([extract_features(email) for email in
data['email_text']])
    X = pd.concat([tfidf_df, manual_df], axis=1)
    y = data['label']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
    clf = RandomForestClassifier(n_estimators=100).fit(X_train, y_train)
    return {'model': clf, 'vectorizer': X_text, 'feature_names':
X.columns.tolist()}, clf.score(X_test, y_test)
```

```python
# Save model
def save_model(data):
    with open("model.pkl", "wb") as f:
        pickle.dump(data, f)

# Load model
def load_model():
    if os.path.exists("model.pkl"):
        with open("model.pkl", "rb") as f:
            return pickle.load(f)
    return None

# Classify and evaluate vulnerability
def classify_email(email, model_data):
    clean = preprocess_email(email)
    features = extract_features(email)
    feature_df = pd.DataFrame([features])
    tfidf = model_data['vectorizer'].transform([clean])
    tfidf_df = pd.DataFrame(tfidf.toarray(),
columns=model_data['vectorizer'].get_feature_names_out())

    for col in model_data['feature_names']:
        if col not in tfidf_df.columns and col not in feature_df.columns:
            if col in tfidf_df.columns:
                tfidf_df[col] = 0
            else:
                feature_df[col] = 0

    X = pd.concat([tfidf_df, feature_df], axis=1)[model_data['feature_names']]
    pred = model_data['model'].predict(X)[0]
    prob = model_data['model'].predict_proba(X)[0][1]
    return pred, prob, features

# Highlights inside email
def highlight_elements(email):
    highlights = {
        "URLs": re.findall(r'(http\S+|www\S+)', email),
        "Urgent Words": [w for w in ['urgent', 'now', 'alert'] if w in
email.lower()],
        "Financial Terms": [w for w in ['account', 'login', 'password', 'verify',
'bank'] if w in email.lower()],
    }
    return highlights
```

```python
# --- Interface ---

st.subheader("📥 Paste Email for Analysis")
email_input = st.text_area("Enter the email text below:", height=250,
placeholder="Paste suspicious email content here...")

col_analyze, col_train = st.columns(2)
analyze = col_analyze.button("🔍 Analyze Email")
train = col_train.button("⚙ Train New Model")

# Train section
if train:
    with st.spinner("Training model..."):
        data = load_sample_data()
        model_data, accuracy = train_model(data)
        save_model(model_data)
        st.success(f"Model trained with accuracy: {accuracy:.2%}")

# Analyze section
if analyze and email_input:
    model_data = load_model()
    if not model_data:
        st.warning("No model found. Training with sample data...")
        model_data, _ = train_model(load_sample_data())
        save_model(model_data)

    with st.spinner("Analyzing..."):
        prediction, probability, features = classify_email(email_input,
model_data)
        highlights = highlight_elements(email_input)

        st.subheader("🧪 Vulnerability Analysis")

        st.metric("Phishing Probability", f"{probability:.2%}")
        strength = "❌ Very Weak" if probability > 0.75 else "⚠ Medium" if
probability > 0.4 else "✅ Strong"
        st.metric("Email Strength", strength)

        st.markdown("### 🔍 Highlighted Elements in Email")
        for key, values in highlights.items():
            if values:
                st.markdown(f"{key}:** " + ", ".join([f"{v}" for v in values]))
            else:
                st.markdown(f"{key}:** None found")

        st.markdown("### 🧬 Feature Summary")
        st.write(pd.DataFrame([features]).T.rename(columns={0: "Value"}))

        st.markdown("### 📜 Original Email with Highlight")
        highlighted = email_input
        for word in highlights["URLs"] + highlights["Urgent Words"] +
highlights["Financial Terms"]:
            highlighted = re.sub(f"({word})", r"**\\1**", highlighted,
flags=re.IGNORECASE)
        st.markdown(highlighted)
```