Lab 3: Detection of Phishing Websites Using Cloud-hosted Data Mining

In this lab, we will implement phishing website detection using machine learning in a cloud-hosted environment (Google Colab).

Objective

Use data mining to identify phishing sites and run detection in Google Colab.

1) Prerequisites & Requirements

Component	Minimum Requirement	Notes
Google Colab	Free Google Account	Run notebook in browser
Python Version	3.8+	Default in Colab
Libraries	pandas, numpy, scikit-learn, matplotlib, seaborn, joblib	Install via pip
Dataset	Phishing + Legitimate URLs	CSV format (Kaggle/PhishTank/Mendeley)

2) Environment Setup

Install required libraries in Colab:

```
# Run this first
!pip install -q pandas numpy scikit-learn joblib
# Optional visualization
!pip install -q matplotlib seaborn
```

3) Data Collection & Upload

Upload phishing and legitimate URL datasets either via Colab upload utility or from Google Drive.

- Download open datasets:
  - Phishing URLs: e.g., https://data.mendeley.com/datasets/vfszbj9b36
  - Legitimate URLs: Several datasets are available; sites like PhishTank or Kaggle offer CSVs.
- Upload CSVs to your Colab instance or fetch with Python

```
from google.colab import files
# files.upload()  # uncomment to invoke upload dialog
```

4) Data Preprocessing

Load datasets, detect the URL column, standardize column names, and label the data.

```
import os
import pandas as pd

phishing_path = "phishing-urls.csv"
legit_path    = "legitimate-urls.csv"

def load_csv(path):
    if not os.path.exists(path):
        raise FileNotFoundError(f"{path} not found.")
    df = pd.read_csv(path, dtype=str, low_memory=False)
    return df

phishing = load_csv(phishing_path)
legit    = load_csv(legit_path)

def find_url_col(df):
    candidates = [c for c in df.columns if c.lower() in
('url','link','website','website_url','uri','domain','site')]
    if candidates:
        return candidates[0]
    for c in df.columns:
        sample = df[c].dropna().astype(str).head(200).str.lower()
        if sample.str.contains('http').sum() > 5 or
sample.str.contains('www').sum() > 5:
            return c
    return df.columns[0]

phishing = phishing.rename(columns={find_url_col(phishing):
'url'})[['url']].dropna().reset_index(drop=True)
legit    = legit.rename(columns={find_url_col(legit):
'url'})[['url']].dropna().reset_index(drop=True)

phishing['label'] = 1
legit['label']    = 0

data = pd.concat([phishing, legit], ignore_index=True).sample(frac=1,
random_state=42).reset_index(drop=True)
```

5) Feature Engineering

Extract lexical features from URLs.

```
import re

def extract_features(url):
    return [
        len(url),
        url.count('.'),
        url.count('-'),
        int('@' in url),
        int('https' in url[:8].lower()),
        int(re.search(r'[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+', url) is not
None)
    ]

X = data['url'].apply(extract_features)
X = pd.DataFrame(X.tolist(),
columns=['length','dots','hyphens','has_at','has_https','has_ip'])
y = data['label']
```

6) Train-Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
```

7) Model Training — Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
```

8) Model Training — Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

9) Model Evaluation

```
from sklearn.metrics import classification_report, accuracy_score

for name, model in [("Decision Tree", dt), ("Random Forest", rf)]:
    preds = model.predict(X_test)
    print(f"\n{name}")
    print("Accuracy:", accuracy_score(y_test, preds))
    print(classification_report(y_test, preds))
```

10) Save/Load Model

```
import joblib

joblib.dump(rf, "phishing_rf_model.pkl")
model = joblib.load("phishing_rf_model.pkl")
```

11) Test with Sample URLs

```
test_urls = [
    "http://secure.paypal.com.fake-site/login",
    "https://www.google.com"
]
test_features = pd.DataFrame([extract_features(u) for u in test_urls])
preds = rf.predict(test_features)
print(list(zip(test_urls, preds)))
```

12) Visualization (Optional)

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.histplot(data['url'].apply(len), bins=50)
plt.title("Distribution of URL Lengths")
plt.show()
```

13) Deploying the Script

Google Colab: The notebook itself is interactive; share via link or “File > Save a copy in Drive”.

We can also download the model.pkl and use it.

14) Troubleshooting

Symptom	Cause	Fix
Model accuracy low	Weak features	Add domain-based / lexical features
Colab FileNotFound	Wrong dataset path	Upload via files.upload() or mount Drive

15) Validation & Testing Checklist

✓Dataset loaded and labeled

✓Feature extraction works

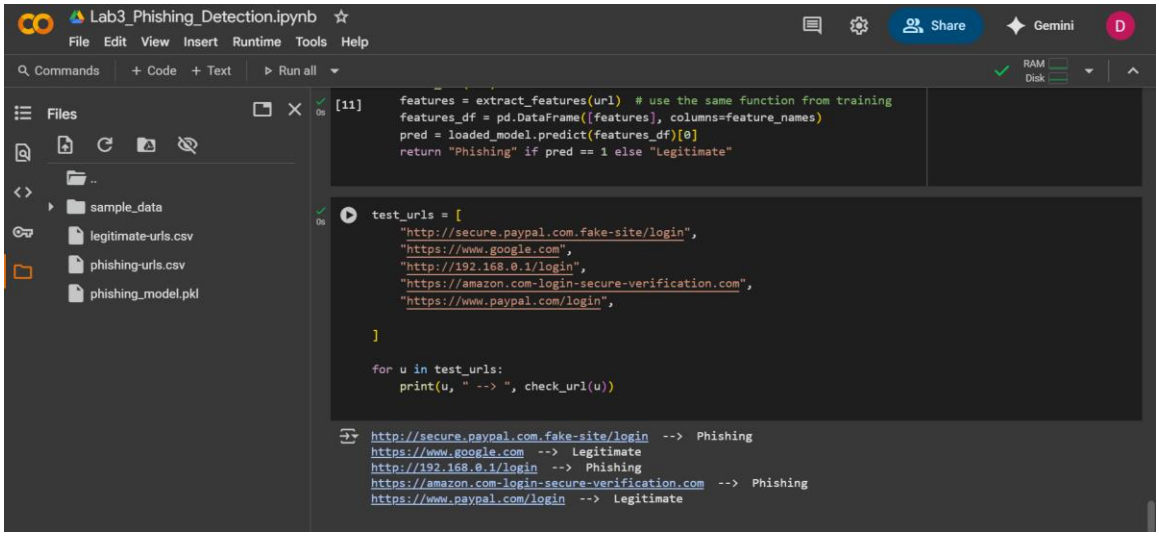
✓Models trained and evaluated

✓Accuracy and classification report printed

✓Test URLs predicted correctly

✓Visualizations generated

Sample Images/ Outputs:



Colab Notebook:

https://colab.research.google.com/drive/1npFZqH0kTH4kR-Do8vdAyt20-OnNdgXt?usp=sharing