Proof of Concept - Detecting Phishing Websites Using Machine Learning

COMP_3260 - Computer Network Security

Rakhat Bektas(t00686769) & Raiyan Mokhammad (t00668359)

Instructions of how to implement and evaluate our MM model

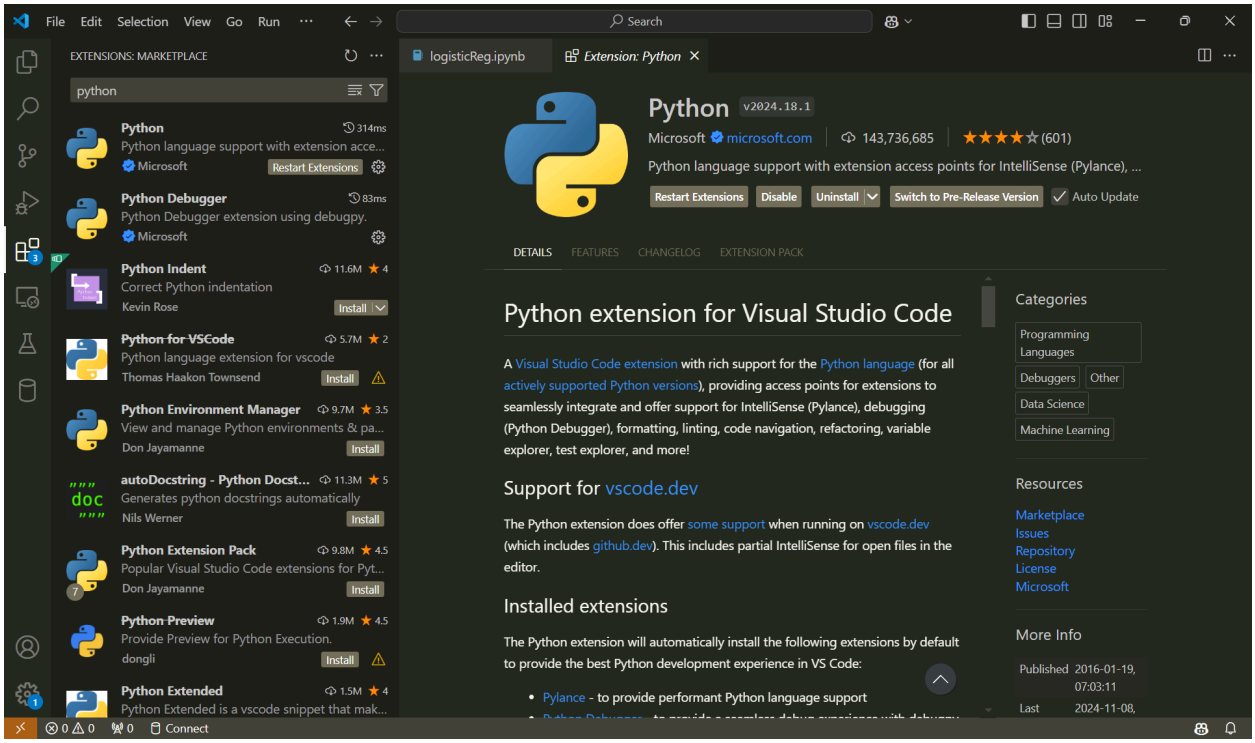
What you need:

- Windows 11 Operating System
- Visual Studio Code
- Mobile Data and ability to share a hotspot with your system (if your router blocks phishing websites)
- Unzipped version of the provided zip file(NAME)

The step-by-step testing run (in our python file all steps are in the logical sequence and explanations for each step, refer to it for better understanding):

1. Open the file(filename.py) in Visual Studio Code.
2. Install needed extension(python) in the extension tab on the left tab

Then in the search bar type python and install it, after type jupyter and install jupyter



3. Install required libraries using terminal in the VS Code. Install them one by one.
   Type: ‘pip install scikit-learn’ or ‘py -m pip install scikit-learn’
   ‘pip install matplotlib’ or ‘py -m pip install matplotlib’
   ‘pip install pandas‘ or ‘py -m pip install pandas’
   ‘pip install numpy’ or ‘pip -m install numpy’
   ‘pip install bs4’ or ‘py -m pip install bs4’
   Example of one of the installations:

```
PS C:\Users\m_ray\OneDrive\Документы\Raiyan\Fall_2024\COMP_3260\Proof Of Concept> py -m pip install bs4
Collecting bs4
  Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Collecting beautifulsoup4 (from bs4)
  Downloading beautifulsoup4-4.12.3-py3-none-any.whl.metadata (3.8 kB)
Collecting soupsieve>1.2 (from beautifulsoup4->bs4)
  Downloading soupsieve-2.6-py3-none-any.whl.metadata (4.6 kB)
```

4. After installing all required libraries you will need to import these functions (Step 1 in python file):

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
import numpy as np
import pickle
import re
from bs4 import BeautifulSoup
from urllib.parse import urlparse
import requests
```

5. Now we can start with a training stage of the model. In our solution, we have chosen to use Decision Tree as is has high accuracy and is easy to implement (Step 2 in python file). In this step make sure that the dataset is in the correct format, you might need to change the slashes in the path, for our solution these slashed '\\ worked fine':

5.1 Import dataset and change the final label:

```
data = pd.read_csv("C:\\Users\\m_ray\\OneDrive\\Документы\\Raiyan\\Fall_2024\\COMP_3260\\Proof Of C

d = {1:'Legitimate', 0: 'Phishing'} #change names of the labels in the table for more clear output

data['label'] = data['label'].map(d) #maps new labels to dataset
```

5.2 Define features upon which the model will be learning:

```
features = ['IsHTTPS', 'HasDescription', 'HasSocialNet', 'HasCopyrightInfo', 'NoOfImage', 'NoOfJS']
```

5.3 Define which features (labels) for training and which for the results:

```
x = data[features] #features for trainig model on
y = data['label'] #target feature
```

5.4 Divide the whole dataset into train and test parts, then predict the accuracy:

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42, stratify=y)

# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=8)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

5.5 Show the accuracy and confusion matrix of the model:

```
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",accuracy_score(y_test, y_pred))
print("Confusion Matrix:",conf_matrix)
print('\nClassification Report:')
print(classification_rep)
```

```
Accuracy: 0.9841148271069129
Confusion Matrix: [[20539   306]
 [  254 14154]]

Classification Report:
              precision    recall  f1-score   support

  Legitimate       0.99      0.99      0.99     20845
    Phishing       0.98      0.98      0.98     14408

    accuracy                           0.98     35253
   macro avg       0.98      0.98      0.98     35253
weighted avg       0.98      0.98      0.98     35253
```

5.6 Save the model:

```
# Save the model as a .pkl file
with open('DT_phishing_model.pkl', 'wb') as file:
    pickle.dump(clf, file)
```

6. Before running the code you should disconnect from your usual router and share your internet data from the phone to your computer. It is required because most of the modern routers have their own firewall protectors which can block the code from retrieving required information from the URLs that we are testing.
7. Now that the model is trained and saved, we now can export the model in the Visual Studio and evaluate it with some real data (Step 3):

7.1 Import the model:

```
# Load the model from the .pkl file
with open('DT_phishing_model.pkl', 'rb') as file:
    loaded_model = pickle.load(file)
```

7.2 Define URLs for detection:

```
url1 = "https://www.youtube.com/" #legit
url2 = "http://www.verkaufsfunnel.com"  #phishing
```

7.3 Get the required features from the websites and convert them to data frames:

```
legit_html = get_html(url1)
phishing_html = get_html(url2)

legit_result = analyze_html(legit_html, url1)
phishing_result = analyze_html(phishing_html, url2)

legit_features = covert_dict(legit_result)
phishing_features = covert_dict(phishing_result)

# Create DataFrames, this is the only format the model c
legit_df = np.array(legit_features).reshape(1, -1)
phishing_df = np.array(phishing_features).reshape(1, -1)
```

7.4 Predict the results for the URLs:

```
#Pricts if the url is phishing or legitimate
predict_url1 = loaded_model.predict(legit_df)
predict_url2 = loaded_model.predict(phishing_df)
```

```
URL 1 is: ['Legitimate']
URL 2 is: ['Phishing']
```

8. For the first run we would recommend not changing URLs and leave them as default to make sure it runs well. Run all the sections one after another.

9. If the results are correct you can proceed with your own experimentation with the code by changing the default URLs to your own to see if the model will detect the phishing website. Do so by changing variables named:'url1' and 'url2'

Thank you for running our code, If you have any problems with the code please reach out to us at 'T00686769@mytru.ca' or 'T00668359@mytru.ca'.