User Guide

Gernate the dataset

- Visual Studio Code
- Python 3.11.8
- BeautifulSoup

Virtual environment

- Anaconda3
- Python 3.7
- Keras
- Tensorflow
- Jupyter Notebook

Installation Steps

To run the project on your local machine, follow these steps:

1. Clone the repository:

```
git clone https://github.com/WSimonHo/smart-website-scanner/tree/master
```

2. Navigate to the project directory:

```
cd ./smart-website-scanner
```

3. Create a virtual environment:

```
conda create --name ai_model python=3.7
```

4. Start the anaconda virtual environment:

```
conda activate ai_model
```

5. Install Tensorflow:

```
conda install tensorflow
```

6. Install Keras:

```
conda install -c conda-forge keras
```

7. Install Jupyter Notebook:

```
conda install jupyter notebook
```

Usage

Load the Website Content by url

1. Open the `smart-website-scanner/model/classify/load_url.py` in VSCode
2. Changed the url csv file want to

```
load_url.py ×
model > classify > load_url.py
import requests
from bs4 import BeautifulSoup
import re
import csv

# List of sites to load
websites = []
with open('./dataset/verified_online.csv', 'r', newline='', encoding='utf-8') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        websites.append(row['url'])
```

3. Update the URL CSV file path to the file you want to use.
4. Run the script. It will load the url and save the website content that is existing.

```python
def load_website(url, sKey):
    try:
        response = requests.get(url, timeout=5)
        response.raise_for_status()  # If the request fails, an HTTPError exception will be raised


        if response.status_code == 200:
            soup = BeautifulSoup(response.text, 'html.parser')

            # Store website content

            # Find and store all URLs
            for link in soup.find_all('a', href=True):
                sub_url = link['href']
                if re.match(r'http[s]?://', sub_url):
                    url_array.append(sub_url)

            file_name = 100000 + sKey
            print(f'success load the {url}')
            with open('website_content_3.csv', 'a', newline='', encoding='utf-8') as csvfile:
                writer = csv.writer(csvfile)
                writer.writerow([file_name,url])

            output_file = './web_content/' + str(file_name) + '.html'
            with open(output_file, "w", encoding="utf-8") as file:
                file.write(response.text)
        else:
            print(f"statu not 200 {url}")
        response.close()

```

Dataset Generate

To generate the dataset for phishing detection, you can follow the example script provided in the `model` directory. Below is a basic usage example:

Generate the Phishing Websites URL Features Dataset

1. Open the `classify_website_generate_url_final.py` in VSCode or Open the `classify_website_generate_url_final.ipynb` in Anaconda
2. Load the csv file that for generate the Phishing Websites URL Features

```python
csv_file = './dataset/website_url.csv'
with open(csv_file, 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip the header row

    # Loop through each row in the CSV file
    for row in reader:
        uid = row[0]
        url = row[1]
        parsed_url = urlparse(url)
        hostname = parsed_url.hostname

        # Check if the URL matches a file in the demo folder
        file_name = os.path.basename(uid+'.html')
        file_path = os.path.join('./web_content', file_name)

        if os.path.exists(file_path):
            start_classify(file_path)
            print( f"file_name {file_name}" )
```

3. Change the name of the data set that will be generated

```python
def save_data_to_csv( item ):
    with open('./dataset/website_content_url.csv', 'a', newline='', encoding='utf-8') as csvfile:
        writer = csv.writer(csvfile)
        # print( f"item {item}" )
        writer.writerow([
            hostname,
            item['website_links'],
            item['anchor_url'],
            item['request_url'],
            item['email_submission'],
            item['different_href_urls'],
            item['right_click_disabled'],
            item['popup_window_text_fields'],
            item['iframe_redirection'],
            item['favicon_external_domain'],
            1
        ])
```

4. Run the script and will see the dataset generated

Generate the Phishing Websites Content Features Dataset

1. Goto the Jupyter Notebook and Open `./model/classify_website_generate_content_final.ipynb`
2. Load the csv file that for generate the Phishing Websites Content Features

```
[2]: df = pd.read_csv('./dataset/dataset_zenodo_fullset.csv')
     df.head(20)
```

3. Change the name of the data set that will be generated

```
In [4]: df.to_csv('./dataset/website_content.csv', index=False)
```

4. Run the script and will see the dataset generated

Merge Legitimate dataset and Phishing dataset into one Dataset

1. Open the `disorganized_data.py` in VSCode
2. Change the datasets name

```
# Read the contents of the first CSV file
file1 = './dataset/website_url_5_b.csv'
data1 = []
with open(file1, 'r') as csv_file1:
    reader1 = csv.reader(csv_file1)
    next(reader1)  # Skip the header row
    data1 = list(reader1)

# Read the contents of the second CSV file
file2 = './dataset/website_url_5_p.csv'
data2 = []
with open(file2, 'r') as csv_file2:
    reader2 = csv.reader(csv_file2)
    next(reader2)  # Skip the header row
    data2 = list(reader2)
```

3. Change the name of the data set that will be generated

```
# Write the merged and disorganized data to a new CSV file
output_file = './dat Loading... d_dataset.csv'
with open(output_file, 'w', newline='') as csv_output:
    writer = csv.writer(csv_output)
    writer.writerow(['url','result'])  # Adjust the column headers as needed

    for row in merged_data:
        writer.writerow(row)
```

4. Run the script and will see the dataset generated

AI Model

To use the AI model for phishing detection, you can follow the example script provided in the `model` directory. Below is a basic usage example:

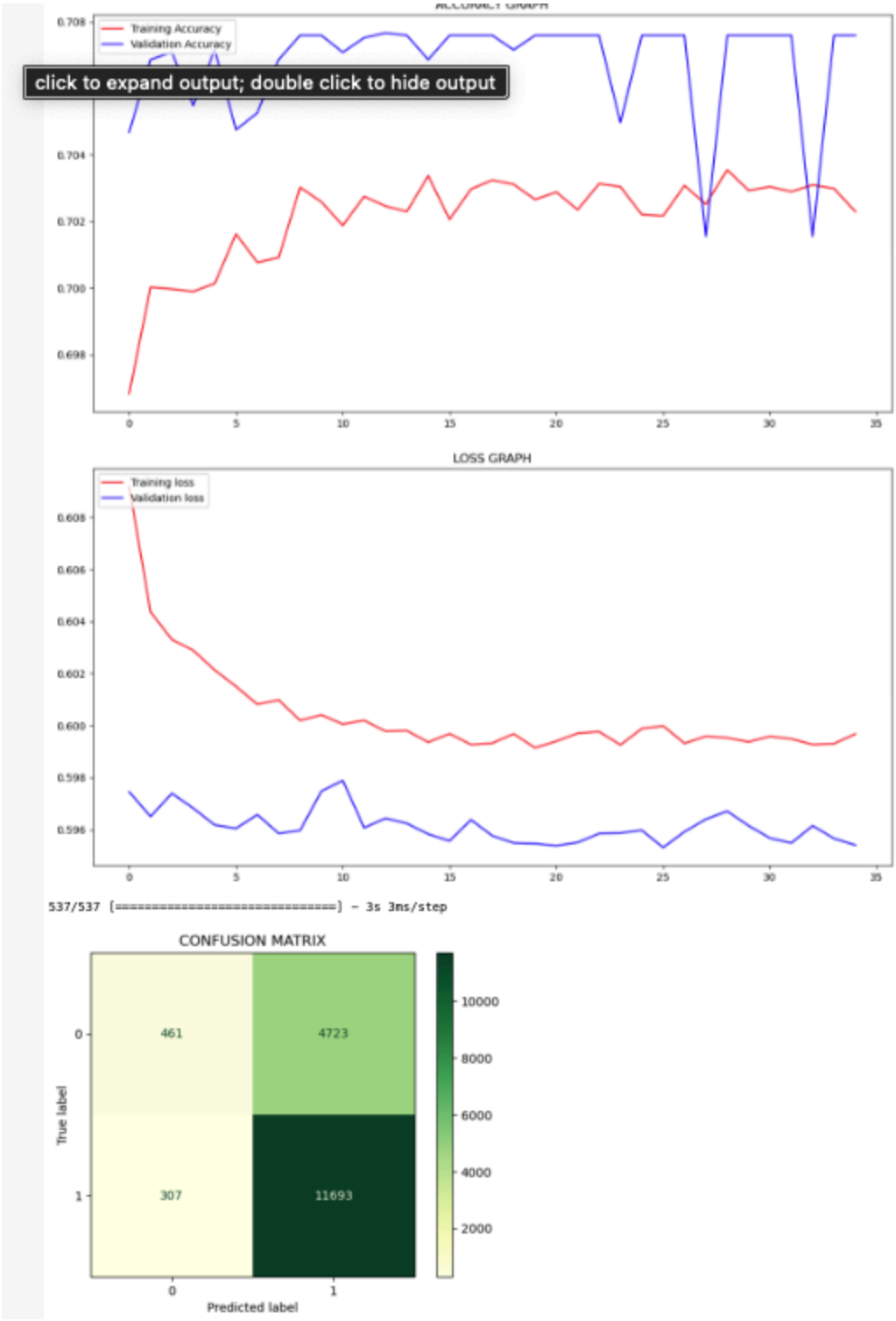
1. Run the Anaconda and Start the Jupyter Notebook
2. Choose one of the Model want to train and test

- [ ] Keras - CNN LSTM - content - v1.0.ipynb
- [ ] Keras - CNN LSTM - content - v1.1.ipynb
- [ ] Keras - CNN LSTM - content v2.0.ipynb
- [ ] Keras - CNN LSTM - content v3.0.ipynb
- [ ] Keras - CNN LSTM - Full v1.0.ipynb
- [ ] Keras - CNN LSTM - Full v2.0.ipynb
- [ ] Keras - CNN LSTM - Full v3.0.ipynb
- [ ] Keras - CNN LSTM - url v1.0.ipynb
- [ ] Keras - CNN LSTM - url v2.0.ipynb
- [ ] Keras - CNN LSTM - url v3.0.ipynb

2. Load the csv file for model 2.1. Run the Classify the dataset (optional)
3. Run the other script (if the dataset is already classified)

```
Y1 = df['result']
X1 = df.drop(columns = ['uid','result'])
```

4. Will Output the result



5. Final will save the ai model to JSON format

```
model_architecture = CNN_LSTM_model1.to_json()
with open('./modal/CNN_LSTM_model_dataset_filter_1_and_0_3.json', 'w') as json_file:
    json_file.write(model_architecture)
```

```
model_architecture = CNN_model1.to_json()
with open('./modal/CNN_model_dataset_filter_1_and_0.json_3', 'w') as json_file:
    json_file.write(model_architecture)
```

```
model_architecture = CNN_LSTM_model1.to_json()
with open('./modal/CNN_LSTM_contentv2_1.json', 'w') as json_file:
    json_file.write(model_architecture)
```

```
model_architecture = CNN_model1.to_json()
with open('./modal/CNN_contentv2_1.json', 'w') as json_file:
    json_file.write(model_architecture)
```