User Data Protection using AI Cybersecurity

A Project Report
Presented to
The Faculty of the College of
Engineering

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Software Engineering

By

Mirsaeid Abolghasemi
Varun Bhaseen
Gulnara Timokhina

May 2021

APPROVED

Vijay Eranti, Project Advisor

Dan Harkey, Director, MS Software Engineering

Rod Fatoohi, Department Chair

ABSTRACT

User Data Protection using AI Cybersecurity

By

Mirsaeid Abolghasemi, Varun Bhaseen, Gulnara Timokhina

A user's privacy protection is of prime importance nowadays. The majority of user privacy violations happen through phishing attacks. As per the reports published by the "Anti-phishing working group for Q4 2020", it has been observed that the number of phishing sites detected was 637,302 of which 84% now use Secure Sockets Layer (SSL) protection. A phishing attack can be detected from paid or freeware anti-virus software, corporate email phishing detection, and user intelligence or awareness.

The primary goal in a phishing attack is to exploit human weaknesses. The challenge with current phishing detection is the lack of availability of a reliable state-of-art tool which could compensate for these weaknesses. The majority of phishing detection technology is based on a classical approach where the agent (detector) relies on information on which it has been trained. The agent does not factor into a real-time artificial intelligence-based detection which can detect the most recent evolved techniques on which phishing attacks are based.

In this project, we will be proposing an artificial intelligence-based real-time detector that can protect user's privacy details by constantly scanning a web page for malicious scripts, phishing contents, domain authenticity, and logo identifiers. Our approach will be using not only natural language processing but also computer vision to detect the authenticity of the web page and the use of logos or images on that web page. The outcome will be to run a plugin on a browser that can consume minimal resources and can give a quick scan notification about the safety of the web page.

Acknowledgments

The authors are deeply indebted to Professor Vijay Eranti for his invaluable comments and assistance in the preparation of this study.

Table of Contents

List of Figures

Chapter 1. Project Overview

1.1. Introduction

Phishing is a security vulnerability that aims to trick unsuspecting users by mixing social engineering and website spoofing techniques into stealing their sensitive details (e.g., password, bank, or financial details). A typical phishing attack's lifecycle begins with the receipt of a fake e-mail, SMS, or instant message from scammers trying to make users think and believe that it comes from a legitimate source. The messages typically use persuasive claims and a link pointing to a fake web page that mimics the legitimate web page of the target brand. If the user enters their credentials, the life cycle of the attack will conclude by submitting confidential information to phishers which can be misused for online fraud or the misuse of private data.

Phishing is one of the oldest and well-known security vulnerability exploitation techniques which relies more on human weaknesses rather than a technological weakness as stated by Bozkir et al. [1] and because of the human factor involved it becomes difficult to eliminate it. Although there are many ways in which a phishing website can be detected the traditional approach has been to rely on databases that maintain a list of these websites and can prevent them from opening at user’s browser but as mentioned by Abdelnabi et al. [5] any zero-day detections are not possible in these traditional approaches. The update of the database takes a considerable amount of time and usually, a certain number of users have already been affected.

Hence to compensate for human weakness Huang et al [2] in their paper have suggested a more artificial intelligence or deep learning-based technique in which the model can more readily be used to identify the malicious URL for the web page. Based on validation the output of the model will classify whether the web page is phishing or not. Whereas Le et al. [3] suggest a more focused and object detection-based approach where they have enhanced certain features and validate the URL based on a certain character and word frequencies.

Here in this paper, we try to combine various techniques for phishing detection and aim to create an AI agent which can assist users by flagging messages ensuring that users are aware of the current web page status. The AI agent will scan the entire web page to determine if the web page is safe or is intended as phishing.

1.2. Proposed Areas of Study and Academic Contribution

The project scope is bounded on assumption that attackers will try to modify or morph slight changes to existing web pages and use the same to deceit an unsuspecting user into revealing details. The slight changes can be minute to human awareness like changes to Logo, or web page URL. The project hence is focusing on three major aspects with first being validating the authenticity of a logo followed by validating the URL address and finally trying to predict if the web page has a malicious (phishing) intent.

The project is combining all three checking or validation criteria into a single ensemble model. The models are based on deep learning techniques and will be ensembled based on multitasking multi-classification techniques. The model will eventually be saved and used in a web browser extension as a knowledge model or AI agent model which will compensate and enable users to be safe from phishing attacks.

Following are the proposed areas of study for this project from an academic perspective:

- Use pipelines to create a production-level application model
- Use ensemble technique to balance the multitask multi-class output
- Use ML operations techniques to create a commercial-ready to market application
- Combining object detection techniques with NLP (Natural Language Processing) techniques.
- Deploying an ensemble model on a web browser application framework optimized for performance and speed without compromising on CPU/GPU usage at runtime.

1.3. Current State of the Art

Phishing as stated earlier and with reports from APWG (Anti Phishing Work Group) is a multi-million-dollar industry and hence there are many traditional and AI-based techniques that are available to use commercially. Here we have listed out few techniques in a similar context to our project and we have inspired few techniques from each of them.

One of the novel techniques we came across was in the article "LogoSENSE" [4]. In this paper, the author highlights what are the different underlying features that are present in the dataset and how to carry out feature extraction. Also, it explains techniques like HOG (Histogram of Oriented Gradients) and what role it plays in detecting the phishing website by comparing logos and images. This paper uses machine learning techniques rather than deep learning techniques, but the approach to enhance the features in the dataset is useful and can be reused. The authors of this research paper have created their dataset (LogoSENSE) on which they carry out feature engineering and data preprocessing to identify whether a given page is phishing or not by simply analyzing the logo on the page.

The second paper that we have identified is “Phishing URL Detection via CNN and Attention-Based Hierarchical RNN” [1]. The authors have used CNN and RNN to extract URL texts or characters or words and use them as features to authenticate and verify a website. The authors worked on a dataset that takes a screenshot of a web page and then uses neural networks to extract features and classify the web pages.

The third technique which we identified is URLNet [2] The authors of this paper focus on using lexical features in a URL to identify whether the website is phishing. This is achieved by using a CNN model that can focus on character level and word level embeddings. The authors have built their dataset and carried out feature extraction using position-sensitive bigrams and trigrams. The conclusion for this paper was to build a state-of-the-art model which can detect phishing website by analyzing the URL. We will be using the feature extraction technique and try to combine it with other URL validation techniques listed earlier for the detection of the phishing website

Then there is Malicious Web Content Detection Using Machine Learning [3]. This paper addresses the imbalanced dataset and what approach should be taken for sampling. The paper also addresses the different dependencies that are there for the creation of applications for data science (Machine learning and Deep learning models).

VisualPhishNet highlights the zero-day phishing website detection approach. What it essentially highlights is that if a phishing website is created today and it is not a part of any phishing website database then how good the model is to detect this fresh new website as a Phishing website [5].

Chapter 2. Project Architecture

2.1. Browser Extension Application Architecture

The figure below provides a high-level architecture for the application. In our application concept, we use two deep learning models based on YOLO and CNN-RNN multi-task multi-label classification. To get better phishing prediction, these algorithms will be applied to the features extracted from web pages like logo, texts, and address bar.

The deep learning models will be used in our application to get these extracted features and then it will classify the webpages to show that a webpage is phishing, suspicious, or legitimate. The application will be packaged as a web browser extension and/or a plugin

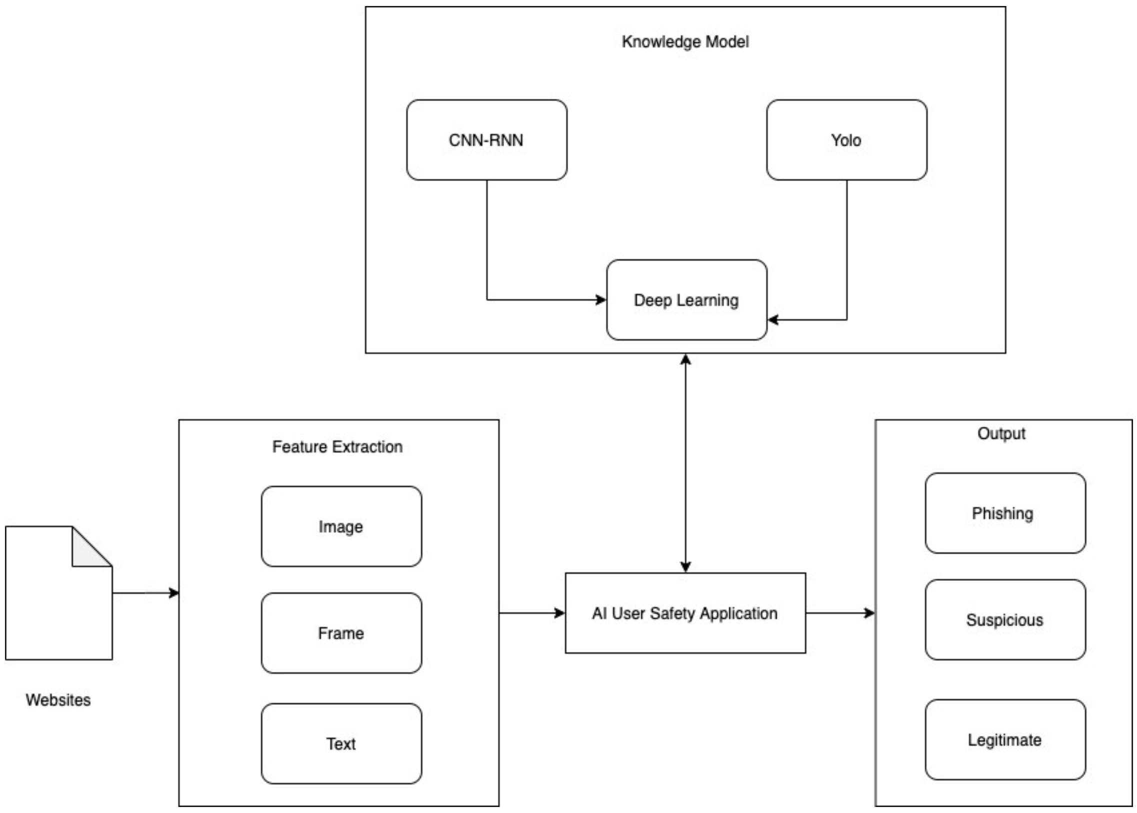


Figure 1: High-level application Architecture

2.2. Model Architecture

It is useful to understand the underlying sub-architecture of the deep learning-based artificial intelligence agent. As seen in figure-2 below we have two different techniques that we will ensemble using a multi-task multi-gate model.

The first technique is YOLO (You Only Look Once), which is an object detection technique that is used to validate a logo present on a web page. YOLO scans a web page, locates the logo as an object. Then it calculates a confidence score. We have set a threshold value (currently 65%) for the confidence score to qualify a logo as a legitimate logo.

The second technique is the Char-CNN-RNN model (Character Convolutional Neural Network and Recurrent Neural Network). First, the URL of a web page is extracted and then Char-CNN-RNN is used to analyze the URL. The URL of a web page is usually morphed by phishers to subdue an unsuspected user and trick them into revealing their information. The model will essentially check to ensure that the URL is valid or if it is a phishing URL and will subsequently raise the flag.

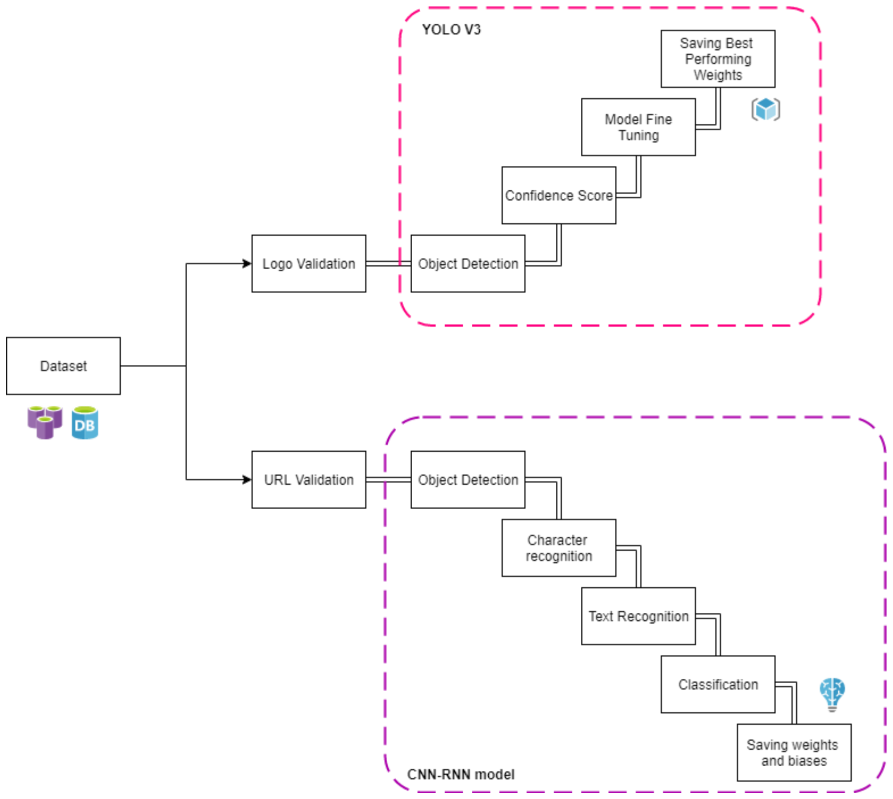


Figure 2: Deep Learning Model Architecture

Chapter 3. Technology Descriptions

3.1. Client-End Plugin

3.1.1. Package Structure

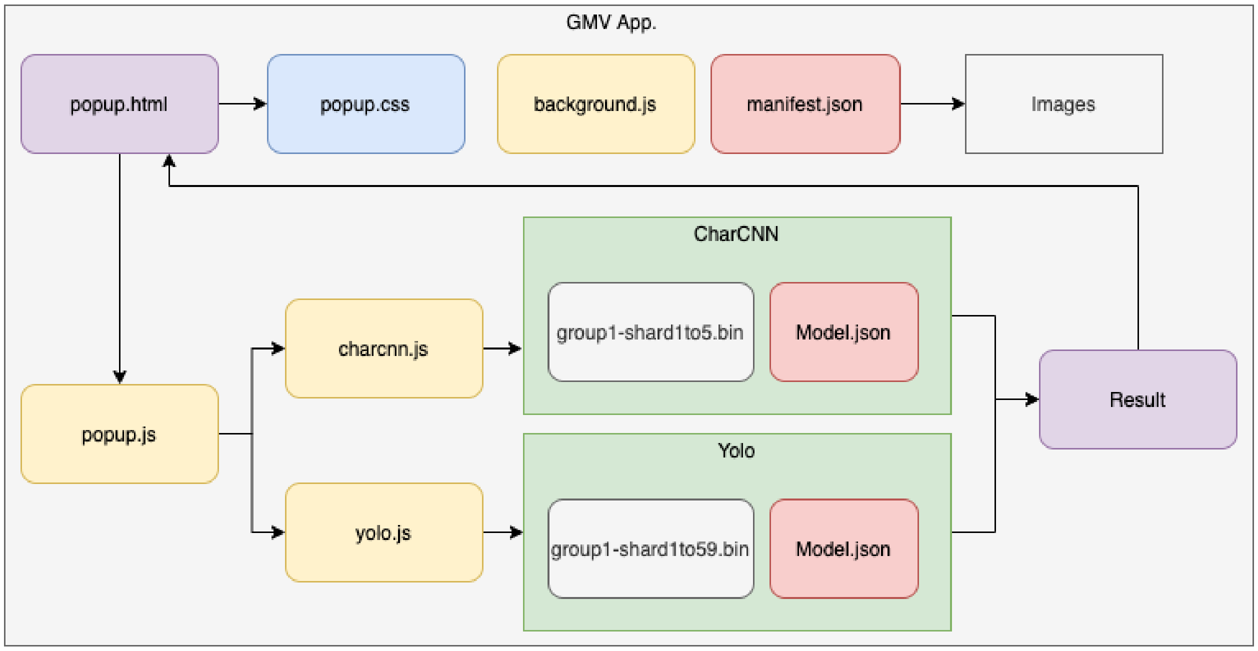


Figure 3: Package Structure of GMV application

We use the Google Chrome extension as our client-end plugin. So, to use the plugin app, first, the GMV extension should be installed on Google Chrome. The user should navigate to the "chrome://extensions/" address on Google Chrome. Then the user should click on the "Load unpacked" button to select the GMV extension folder. After that user can pin the GMV extension to check a website is phishing or not.

The app includes a popup HTML file which calls popup CSS file and popup JavaScript file. The popup JavaScript file will call two JavaScript files charcnn.js and yolo.js. the charcnn.js file will get the webpage URL and will pass to the CharCNN model. The model will return a result tensor that shows the URL is a phishing URL or not. The yolo.js file will take a screenshot and will send it to the Yolo model. The Yolo model will return a tensor that includes Confidence Scores or Objectness Scores. Every Chrome extension package has a JSON file, manifest.json, that provides important information such as the name of the extension, the version, and other settings. Background JavaScript can respond to browser events and execute certain actions, which means the background page is loaded or unloaded when it is required. To use this background JavaScript file, it should be declared in the manifest.json file.

3.1.2. *Plugin Styling*

The plugin app consists of static and template directories which comprises HTML-CSS styling scripts. The scripts are based on SJSU color coding (#0055A2, #E5A823, #939597). There are three files to create the user interface. The popup HTML file is the main page to run the app and show the result to the user. This HTML file will interact with popup.css and popup.js files.

3.1.3. *Plugin Libraries*

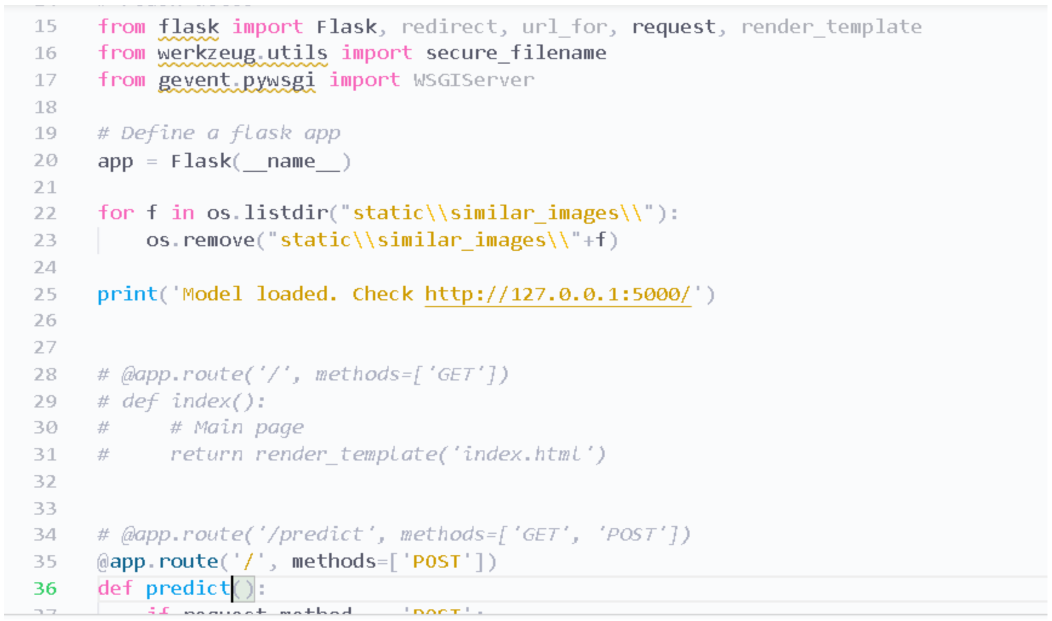
Following are the key libraries that are being used in the plugin for development purposes:

- TensorFlow JS: We are using TensorFlow JS to import ML-DL models in the plugin and predict the output tensors. We are also using the library to convert the inputs to tensors for the Char-CNN model

3.2. Middleware Hooks

3.2.1. *HTTP POST Chrome transmission*

We are using Flask and Werkzeug in the back-end to handle the HTTP POST routing from the client end. Below is a screenshot for POST routing in the flask:



```
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer

# Define a flask app
app = Flask(__name__)

for f in os.listdir("static\\similar_images\\"):
    os.remove("static\\similar_images\\"+f)

print('Model loaded. Check http://127.0.0.1:5000/')


# @app.route('/', methods=['GET'])
# def index():
#     # Main page
#     return render_template('index.html')


# @app.route('/predict', methods=['GET', 'POST'])
@app.route('/', methods=['POST'])
def predict():
```

Figure 4: Screenshot for Back-end Flask routing using HTTP POST

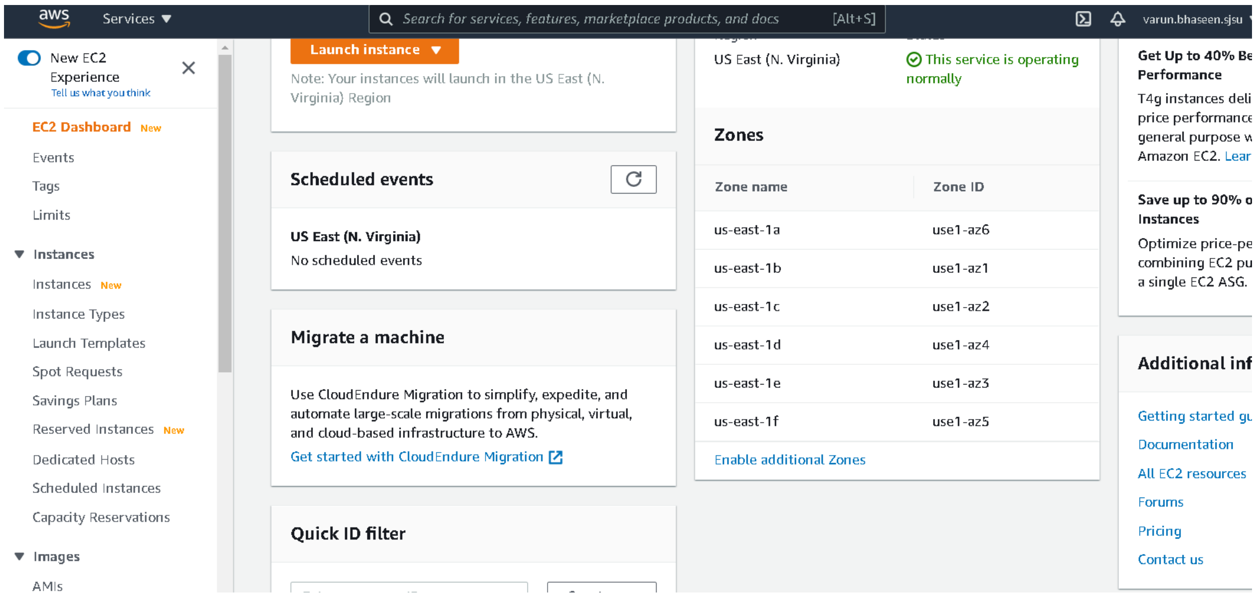
3.2.2. Data Encryption

Data encryption is being handled by a token exchange between the back-end server and the client end based on the user session. The token is stored as a cookie. Refer source code for detailed method on token exchange and authorization and authentication.

3.3. Back-end Server

3.3.1. Cloud Service Provider

The back-end server is hosted on AWS EC2.



Note: Your instances will launch in the US East (N. Virginia) Region

Scheduled events

US East (N. Virginia)
No scheduled events

Migrate a machine

Use CloudEndure Migration to simplify, expedite, and automate large-scale migrations from physical, virtual, and cloud-based infrastructure to AWS.
Get started with CloudEndure Migration

Quick ID filter

US East (N. Virginia) — This service is operating normally

Zones

Zone name	Zone ID
us-east-1a	use1-az6
us-east-1b	use1-az1
us-east-1c	use1-az2
us-east-1d	use1-az4
us-east-1e	use1-az3
us-east-1f	use1-az5

Enable additional Zones

Get Up to 40% Be Performance
T4g instances deli price performance general purpose w Amazon EC2. Lear

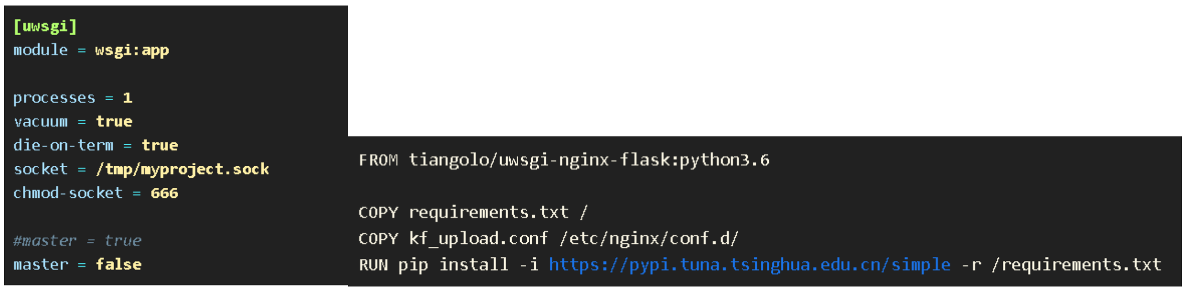
Save up to 90% o Instances
Optimize price-pe combining EC2 pu a single EC2 ASG.

Additional inf
Getting started gu
Documentation
All EC2 resources
Forums
Pricing
Contact us

Figure 5: AWS EC2 console and Dashboard for Server Management

3.3.2. Web Server Host (Docker-Nginx-UWSGI)

The Container used for hosting the backend is on the Docker image.



```
[uwsgi]
module = wsgi:app

processes = 1
vacuum = true
die-on-term = true
socket = /tmp/myproject.sock
chmod-socket = 666

#master = true
master = false
```

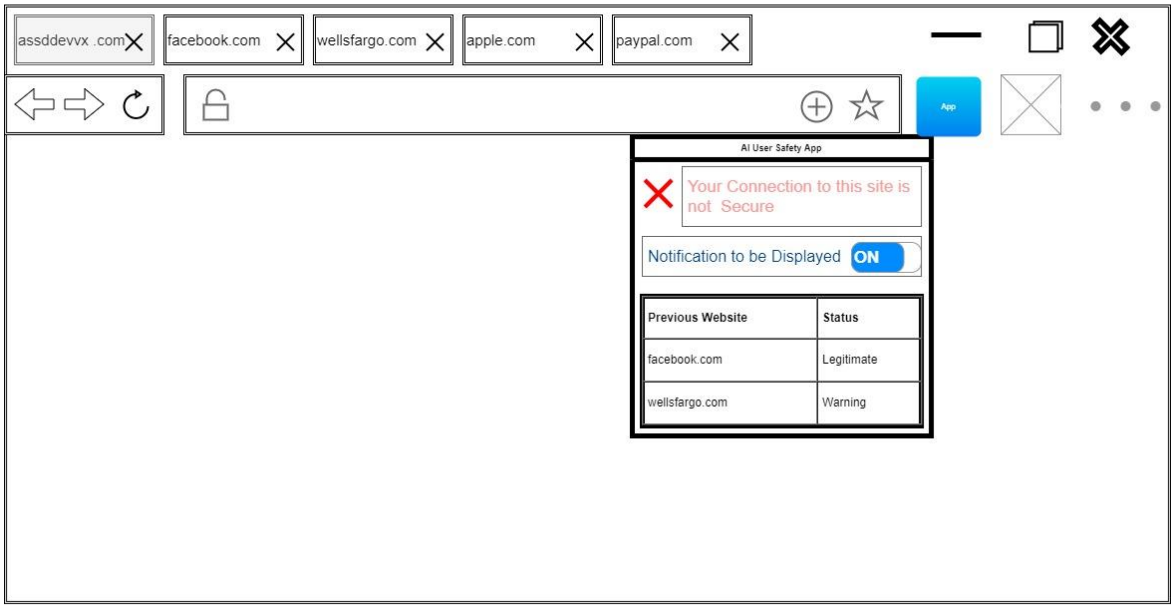
```
FROM tiangolo/uwsgi-nginx-flask:python3.6

COPY requirements.txt /
COPY kf_upload.conf /etc/nginx/conf.d/
RUN pip install -i https://pypi.tuna.tsinghua.edu.cn/simple -r /requirements.txt
```

Figure 6: UWSGI and DockerFile configuration for Back-End application

Chapter 4. Project Design

4.1. Plugin Application Design

4.1.1. Wireframe Design



Your Connection to this site is not Secure

Notification to be Displayed ON

Previous Website	Status
facebook.com	Legitimate
wellsfargo.com	Warning

Figure 7: Wireframe for the plugin

4.1.2. UML Class Diagram

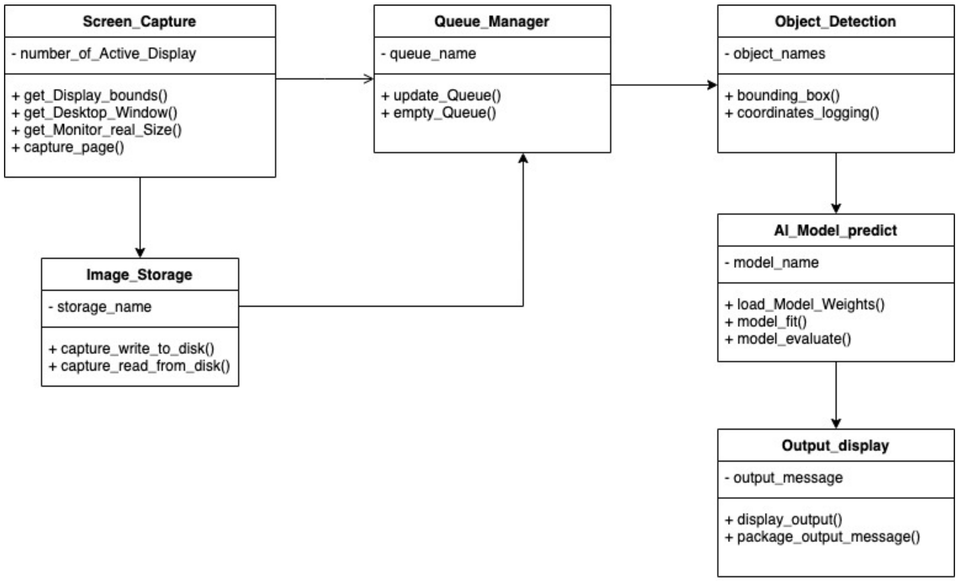


Figure 8: UML Class Diagram for Chrome Plugin Application

4.1.3. UML Sequence Diagram

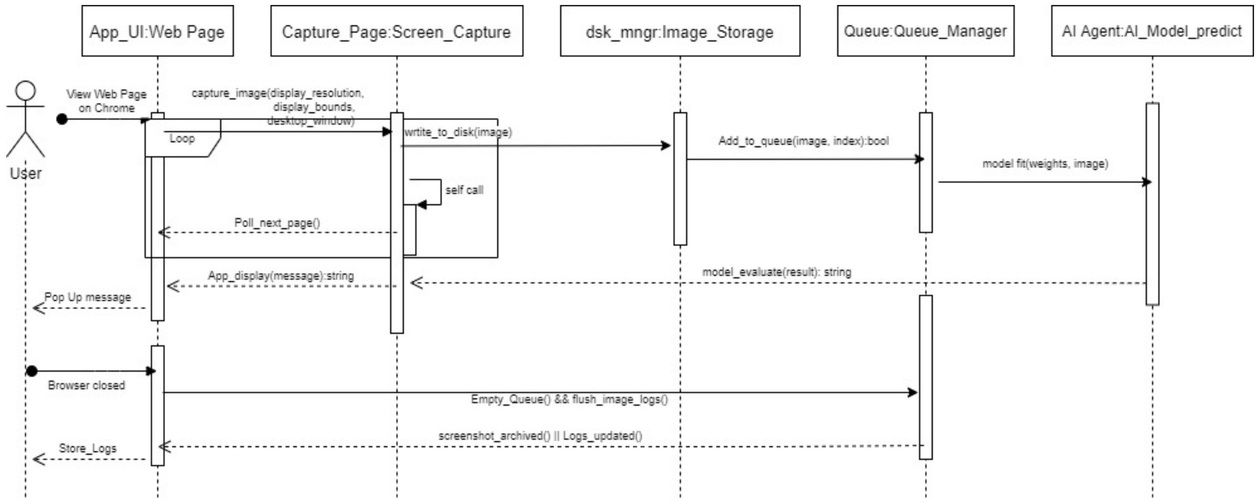

App_UI:Web Page
Capture_Page:Screen_Capture
dsk_mngr:Image_Storage
Queue:Queue_Manager
AI Agent:AI_Model_predict
User
View Web Page on Chrome
capture_image(display_resolution, display_bounds, desktop_window)
Loop
wrtite_to_disk(image)
Add_to_queue(image, index):bool
model fit(weights, image)
self call
Poll_next_page()
App_display(message):string
model_evaluate(result): string
Pop Up message
Browser closed
Empty_Queue() && flush_image_logs()
screenshot_archived() || Logs_updated()
Store_Logs

Figure 9: UML Sequence Diagram for Chrome Plugin Application

4.2. AI Cybersecurity Model Design

4.2.1. YOLO Graph Model Architecture (TensorBoard)

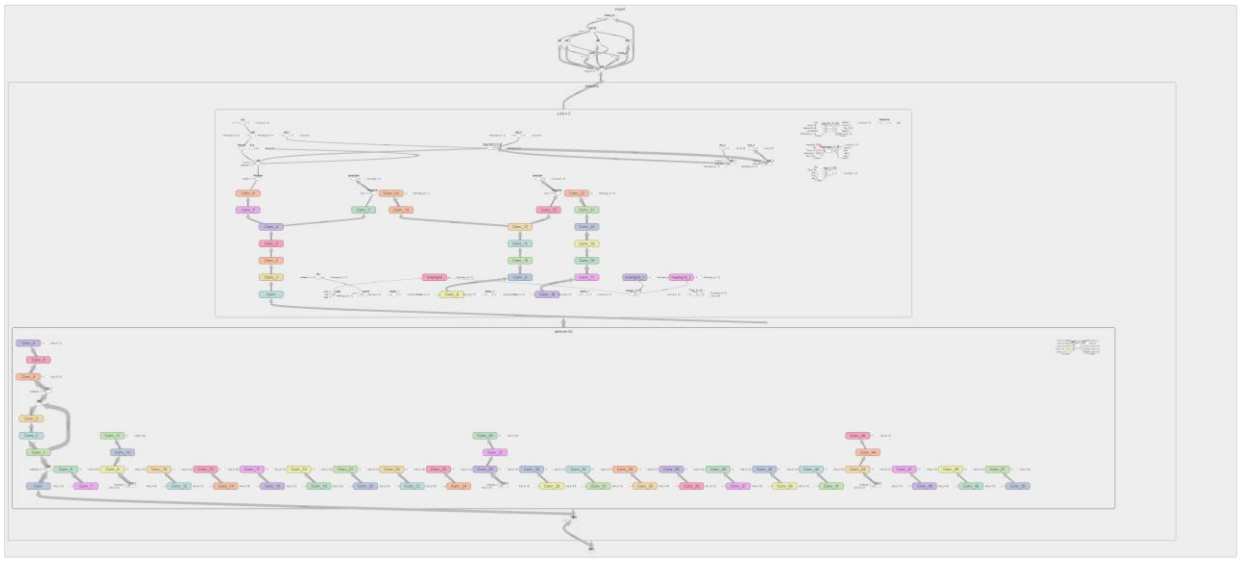


Figure 10: YOLO GraphModel Architecture in TensorBoard

4.3. Backend Prediction UML Class diagram

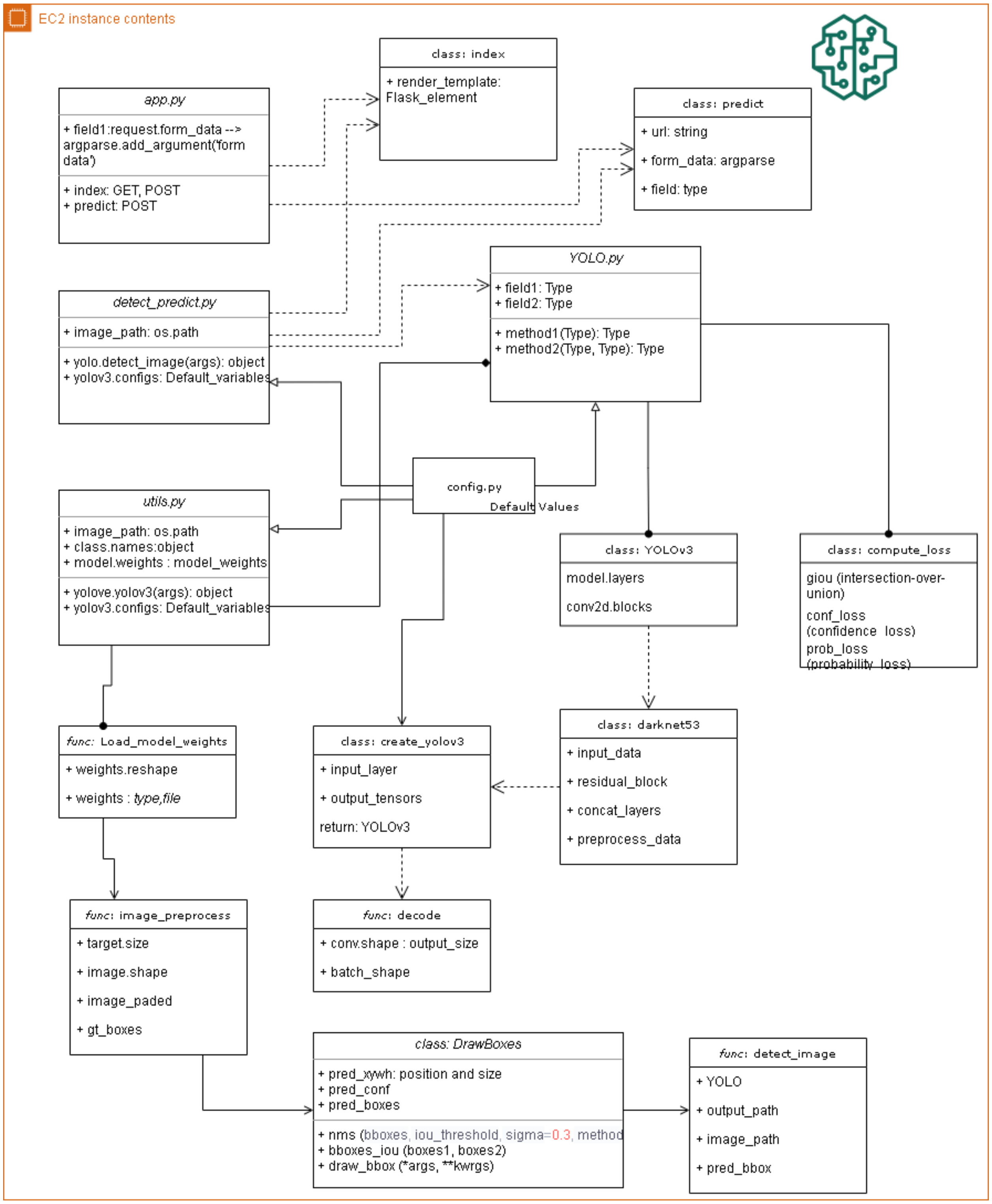


Figure 11: UML Class Diagram for Back-End Flask Application on Object Detection using YOLO

Chapter 5. Project Implementation

5.1. Plugin Application Implementation

5.1.1. Key Objective

The primary objective of the project is to create a web browser plugin or extension which can act as a personalized Artificial Intelligent agent to detect a phishing website and protect personal data. Following are the list of objectives that are being achieved from the plugin:

a. Capture a user session and maintain a queue of web pages visited

b. Validate URL and predict phishing using Char-CNN

c. Validate the web page

d. Capture URI's and send a POST to the backend server for object detection and prediction

e. Compute and flash the result

5.1.2. Market Innovation

Internet is an essential utility in our day-to-day life, we manage our social connections, banking, most financial activities almost exclusively online. This usage and dependency have opened up plenty of ways for our vital private information to be intercepted and misused for malicious activities or theft over the web. One such attack a user can be a victim of is Phishing. Phishers use social engineering and create a mockup of essential websites which can be used to steal our private information.

Here we are trying to have a personalized AI agent which can help protect a user's private data from being stolen from a phishing website. This is executed by validating the URL for authenticity and validate the logo and get a confidence score on the authenticity of the same. This is followed by content validation to check and identify any tell-tale signs of spamming or phishing.

This is the first time that a solution is being developed which will be combining the results from a YOLO object detection with that of Char-CNN to validate a web page and check for phishing. YOLO is a state-of-art object detection model used to identify objects in a video or an image. We have created a custom dataset and trained both YOLO and Char-CNN models to help us achieve results aligned to our desired objective of phishing detection.

5.1.3. Design Principles and Approach Followed

The design principle and guidelines followed for this project can be seen below:

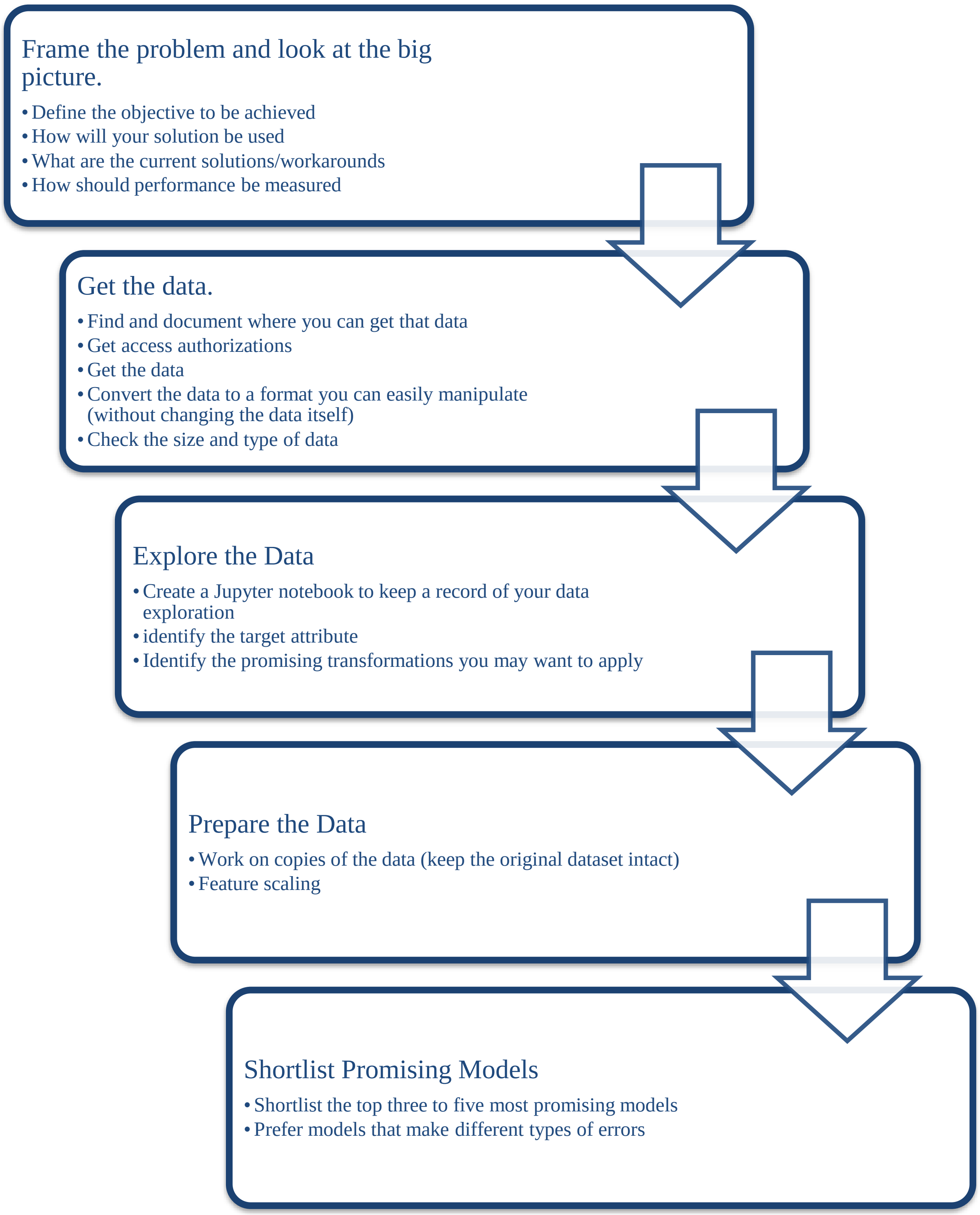

Frame the problem and look at the big picture.
• Define the objective to be achieved
• How will your solution be used
• What are the current solutions/workarounds
• How should performance be measured
Get the data.
• Find and document where you can get that data
• Get access authorizations
• Get the data
• Convert the data to a format you can easily manipulate (without changing the data itself)
• Check the size and type of data
Explore the Data
• Create a Jupyter notebook to keep a record of your data exploration
• identify the target attribute
• Identify the promising transformations you may want to apply
Prepare the Data
• Work on copies of the data (keep the original dataset intact)
• Feature scaling
Shortlist Promising Models
• Shortlist the top three to five most promising models
• Prefer models that make different types of errors

Figure 12: Design Principles and approach for the project design and development

5.2. AI Cybersecurity Model Implementation

5.2.1. Key Objectives

The primary objective of the AI cybersecurity model (YOLO and Char-CNN) used was to serve as an intelligence engine that can help the plugin to respond quickly to an analysis of a webpage and determine if the evaluated web page is phishing or not.

The objectives targeted to achieve by the back-end cybersecurity model are:

a. Process the input received from the client end and determine the confidence score on the objects detected

b. Ensure the entire data pipeline is secured and encrypted over the web

c. Return the results to the client to be computed and published

5.2.2. Ensemble vs Stand-alone vs Cascade modeling

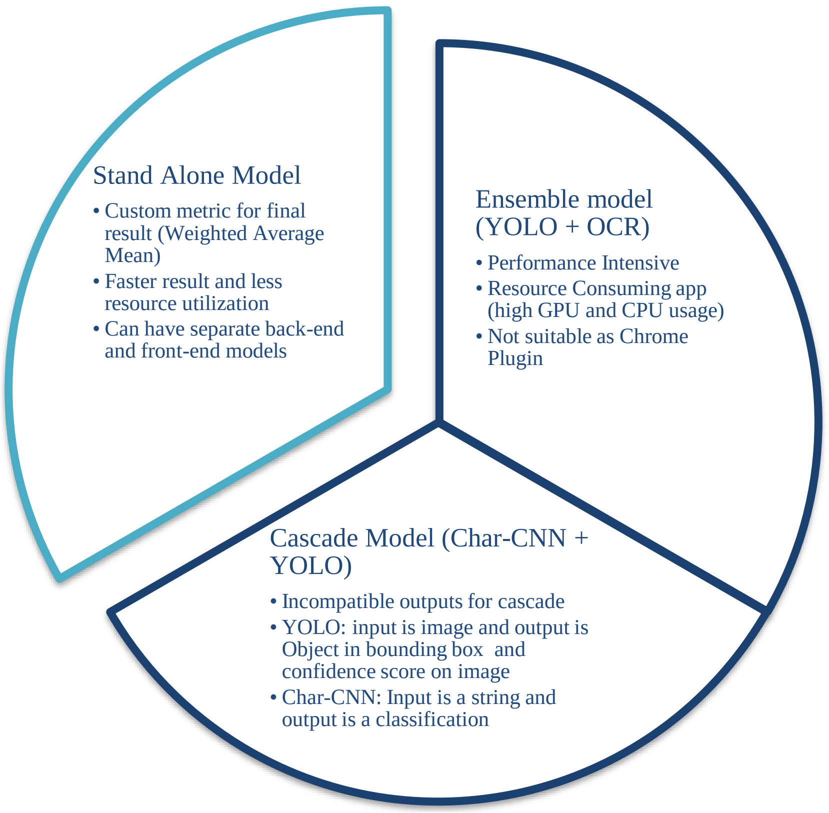

Stand Alone Model
• Custom metric for final result (Weighted Average Mean)
• Faster result and less resource utilization
• Can have separate back-end and front-end models
Ensemble model (YOLO + OCR)
• Performance Intensive
• Resource Consuming app (high GPU and CPU usage)
• Not suitable as Chrome Plugin
Cascade Model (Char-CNN + YOLO)
• Incompatible outputs for cascade
• YOLO: input is image and output is Object in bounding box and confidence score on image
• Char-CNN: Input is a string and output is a classification

Figure 13: Comparision between Ensemble vs Standalone and Cascade Model.

Chapter 6. Testing and Verification

6.1. AI Model Validation and Testing

6.1.1. YOLO Model Validation and Testing Process

ID	MLDL001
Functionality	Logo Validation
Model	YOLO v3, YOLO v4-Tiny
Technique	Transfer Learning
Data Format Input	Images
Dataset	Self-Annotated Logo Dataset
Sources	• Archive.org • Google Image Search Crawling
Volume	100 samples per target brand (Total 1000 Images)
Labelling	Hyperlabel
Split Ratio	• Train: 80% • Validation: 20% of Train • Test: 20%
Overfitting Check	• K-Fold Cross Validation (Training Dataset (80%)) • Image Augmentations (Training Dataset (80%))
Underfitting Check	• Number of Epochs in training to be varied from 500 to 1000 • Best performing model to be selected
Metrics	• Confidence Score • IOU (intersection over union) • NMS (Non-Maximum Suppression • Accuracy

6.1.2. Char-CNN Model Validation and Testing Process

ID	MLDL002
Functionality	URL Validation
Model	Char-CNN
Stage-1 Input	Images
Stage-2 Input	Strings
Technique	Transfer Learning or Retraining all layers
Dataset	Prelabelled
Sources	• PhishTank
Volume	120 samples per target brand (1200 URLs)
Split Ratio	• Train: 80% • Test: 20%
Overfitting Check	• Early Stopping • Regularization • Masking
Metrics	• Recall

	• Precision • F1 Score • Accuracy

6.2. Application Testing

6.2.1. Test Plan

The test plan for the plugin app covers different testing types and different levels to make sure with specific confidence that our application works as expected. These testing types are such as A/B Testing, Hypothesis testing, Vulnerability testing, Pentest, User Acceptance Test. To do testing at different levels requires balancing the time and level of confidence to reach and spend for testing. The test plan also covers three levels of testing like Unit Testing, Integration Testing, and End-to-end Testing. In Unit Testing, all individual units are tested. The module, component, or class will be tested to validate its correctness. In Integration Testing, the sub-systems and modules and their interfaces will be tested. In this case, the way that how different parts of the application interact together will be tested. In end-to-end testing, the application will be tested as a whole. For instance, the user interactions with the application and how to show the result should be tested.

In the A/B test, two versions (A and B) of a variable are compared, which are identical except for one variation that might affect a user's behavior. This test is considered the simplest form of controlled experiment. In fact, by adding more variants to the test, it becomes more complex.

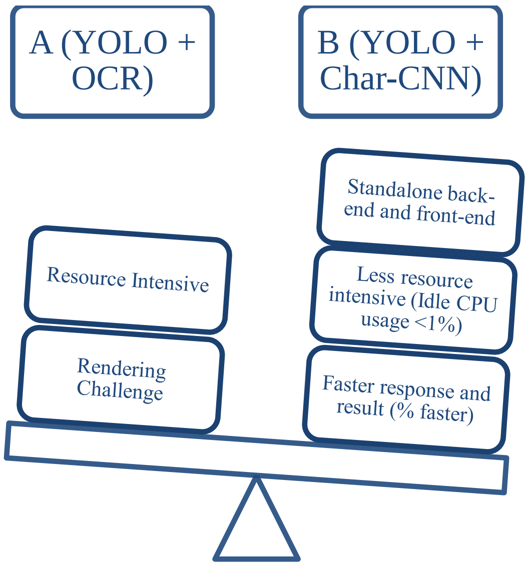
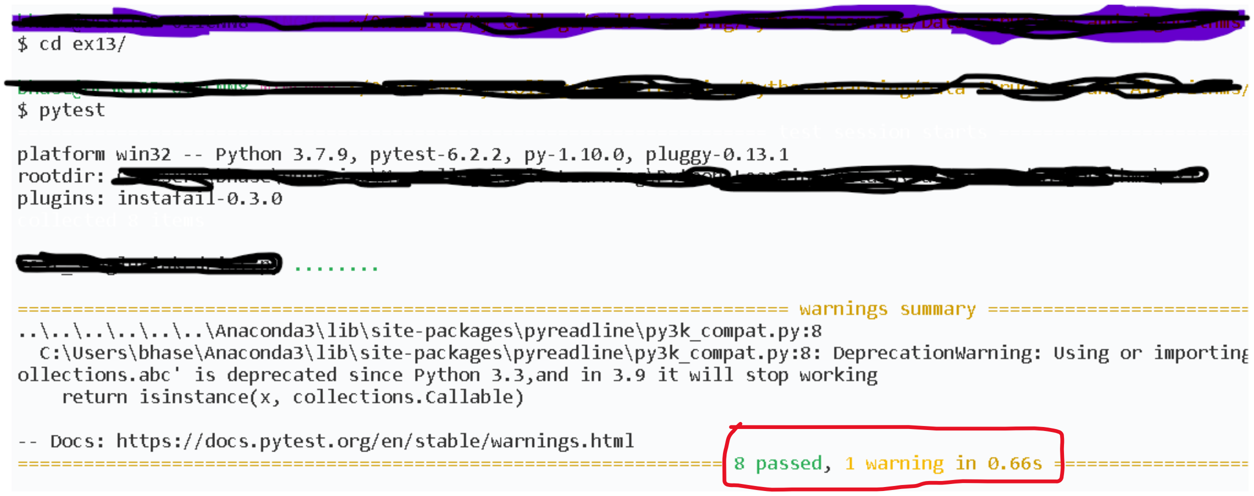

A (YOLO + OCR)
B (YOLO + Char-CNN)
Standalone back-end and front-end
Resource Intensive
Less resource intensive (Idle CPU usage <1%)
Rendering Challenge
Faster response and result (% faster)

Figure 14: A/B Testing for two variations of prediction models

6.2.2. Unit Tests

By creating unit tests, the valid inputs and outputs will be declared for a given function. Therefore, refactoring the code will be easier because there will be confidence that the code works correctly when all the tests pass

Below is the test report:



```
[redacted]
$ cd ex13/

[redacted]
$ pytest

platform win32 -- Python 3.7.9, pytest-6.2.2, py-1.10.0, pluggy-0.13.1
rootdir: [redacted]
plugins: instafail-0.3.0

[redacted] ........

====================================================================== warnings summary ========================
..\..\..\..\..\..\Anaconda3\lib\site-packages\pyreadline\py3k_compat.py:8
  C:\Users\bhase\Anaconda3\lib\site-packages\pyreadline\py3k_compat.py:8: DeprecationWarning: Using or importing
ollections.abc' is deprecated since Python 3.3,and in 3.9 it will stop working
    return isinstance(x, collections.Callable)

-- Docs: https://docs.pytest.org/en/stable/warnings.html
================================================================= 8 passed, 1 warning in 0.66s =================
```

Figure 15: Screenshot of a sample of unit test cases from the Back-end server

Chapter 7. Performance and Benchmarks

To Be Published in Final Version.

Chapter 8. Deployment, Operations, Maintenance

To Be Published in Final Version.

Chapter 9. Summary, Conclusions, and Recommendations

To be Published in Final Version.

Glossary

References

1. Huang, Yongjie, Yang, Qiping, Qin, Jinghui, & Wen, Wushao. (2019). **Phishing URL Detection via CNN and Attention-Based Hierarchical RNN**. *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*

2. Hung Le, Quang Pham, Doyen Sahoo, Steven C.H. Hoi. 2018. **URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection**. *In Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17), 13 pages.*

3. Desai, A., Jatakia, J., Naik, R., & Raul, N. (2017, May**). Malicious web content detection using machine learning**. *In 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT) (pp. 1432-1436). IEEE.*

4. Bozkir, Ahmet Selman, & Aydos, Murat. (2020). **LogoSENSE: A companion HOG based logo detection scheme for phishing web page and E-mail brand recognition**. *Computers & Security, 95, 101855.*

5. Abdelnabi, Sahar, Krombholz, Katharina, & Fritz, Mario. (2019). **VisualPhishNet: Zero-Day Phishing Website Detection by Visual Similarity**. https://arxiv.org/abs/1909.00300

6. Redmon, Joseph, Divvala, Santosh, Girshick, Ross, & Farhadi, Ali. (2016). **You Only Look Once: Unified, Real-Time Object Detection**. *2016, 779-788.*

Appendices

Appendix A.