National College of Ireland

Optimising Real-Time Threat Detection: A Hybrid SVM and ANN Approach

Academic Internship
MSc Cybersecurity

Chethanprasad Narasimhamurthy
Student ID: X22180591

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Chethanprasad Narasimhamurthy		
Student ID:	X22180591		
Programme:	MSc CyberSecurity	**Year:**	2023-2024
Module:	Academic Internship		
Supervisor:	Vikas Sahni		
Submission Due Date:	14/12/2023		
Project Title:	Optimising Real-Time Threat Detection: A Hybrid SVM and ANN Approach		

Word Count: 1223 **Page Count:** 6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Chethanprasad Narasimhamurthy

Date: 14/12/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Optimising Real-Time Threat Detection: A Hybrid SVM and ANN approach

Chethanprasad Narasimhamurthy
Student ID: X22180591

This guide is designed to assist in setting up a python environment, specifically tailored for machine learning applications. To begin, ensure python is installed on your system by downloading it from the official python website. Following this, the manual orchestrates the installation of critical python libraries—numpy, pandas, matplotlib, seaborn, and scikit-learn—via pip, specifying version details to ensure compatibility.

For an optimal coding experience, Jupyter Notebook is recommended and can be effortlessly installed using a provided command. Dive deeper into machine learning capabilities by installing TensorFlow 2.15.0 specifically for the MLPClassifier.

The manual includes a crucial verification step, prompting users to create a Jupyter Notebook and execute code to confirm a flawless setup. To troubleshoot potential issues, a helpful note suggests updating libraries effortlessly through a single command.

1. Installation of python 3.11.0:

Ensure that python has been installed on the system. The version which is required can be downloaded from the official link[1].

2. Required python libraries installation:

A terminal or command prompt should be opened, and the following commands should be executed to install the numpy, pandas, matplotlib, seaborn and scikit-learn python libraries using pip. The version details of the libraries are numpy version: 1.26.2, pandas version: 2.1.3 matplotlib: 3.8.2, seaborn: 0.13.0, scikit-learn version: 1.3.2

pip install numpy==1.26.2
pip install pandas==2.1.3
pip install matplotlib==3.8.2
pip install seaborn==0.13.0
pip install scikit-learn==1.3.2

3. Installation of jupyter notebook:

jupyter notebook is preferred for code execution, it can be installed using the command:
Juputer version: 1.0.0

pip install jupyter==1.0.0

Jupyter notebook can then be launched by typing ‘jupyter notebook’ in the terminal.

[1] https://www.python.org/ftp/python/3.11.0/python-3.11.0-amd64.exe

4. TensorFlow installation (for MLPClassifier):

TensorFlow version: 2.15.0

pip install tensorflow==2.15.0

5. Installation verification:

A new jupyter notebook should be created, and the provided code should be copied and pasted for execution. It should be ensured that there are no errors, and the required libraries are successfully imported.

6. Additional Note:

If issues related to library versions or missing dependencies are encountered, the libraries can be updated using the following commands:

pip install --upgrade numpy pandas matplotlib seaborn scikit-learn tensorflow

Software version summary table:

Software Name	Version	Download URL
Python	3.11.0	https://www.python.org/ftp/python/3.11.0/python-3.11.0-amd64.exe
Jupyter	1.0.0	https://jupyter.org/install

Libraries version summary table:

Library Name	Version	Download URL
numpy	1.26.2	https://files.pythonhosted.org/packages/dd/2b/205ddff2314d4eea852e31d53b8e55eb3f32b292efc3dd86bd827ab9019d/numpy-1.26.2.tar.gz
pandas	2.1.3	https://files.pythonhosted.org/packages/86/ff/662dde2193fc93b8547b073db20472b9676f944d907247a46c9c5bc45bfc/pandas-2.1.3.tar.gz
matplotlib	3.8.2	https://files.pythonhosted.org/packages/fb/ab/38a0e94cb01dacb50f06957c2bed1c83b8f9dac6618988a37b2487862944/matplotlib-3.8.2.tar.gz
seaborn	0.13.0	https://files.pythonhosted.org/packages/06/6f/caf0741c5787358b0efba3b4db7f8235e3a48e719ad2444bbd51485f966c/seaborn-0.13.0.tar.gz
scikit-learn	1.3.2	https://files.pythonhosted.org/packages/88/00/835e3d280fdd7784e76bdef91dd9487582d7951a7254f59fc8004fc8b213/scikit-learn-1.3.2.tar.gz
TensorFlow	2.15.0	https://files.pythonhosted.org/packages/93/21/9b035a4f823d6aee2917c75415be9a95861ff3d73a0a65e48edbf210cec1/tensorflow-2.15.0-cp311-cp311-win_amd64.whl

Data Preprocessing:

In the data preprocessing, several crucial steps are taken to enhance the dataset's suitability for modelling. Firstly, numerical feature scaling is implemented using the RobustScaler to bolster the model's resilience against outliers. This method involves removing the median and scaling data based on the interquartile range, effectively reducing the model's sensitivity to extreme values. Additionally, categorical features undergo one-hot encoding, a technique that enriches the model's ability to interpret categorical information. Notably, the drop_first parameter is set to True during one-hot encoding to mitigate multicollinearity, where the presence of one category can be inferred from others. Furthermore, a transformation is applied to the target column, specifically for binary classification. Instances labelled as "normal" receive the value 0, while other instances are assigned the value 1. This targeted transformation sets the stage for a binary classification task, aimed at distinguishing normal instances from those indicative of a potential attack.

The preprocessing steps ensure the robustness and interpretability of the model. Numerical feature scaling is applied using the RobustScaler to handle outliers effectively. The 'preprocess' function takes the original dataset ('dataframe'), a list of categorical columns ('cat_cols'), a scaler object ('scaler'), and the target column name ('target_col') as inputs. It separates numerical and categorical columns from the original dataset. Numerical columns are scaled using the 'Scaling' function.

Scaled numerical features are concatenated with the original categorical features. Categorical features are encoded using one-hot encoding to enhance the model's ability to interpret categorical information.

Categorical columns are identified based on predefined lists ('cat_cols_1' and 'cat_cols_2'). This modular approach allows the script to adapt to different dataset structures. The drop_first parameter is set to True during one-hot encoding to prevent multicollinearity. The target column undergoes transformation for binary classification, where instances labeled as "normal" are assigned the value 0, and other instances are assigned the value 1.

The pre-processed data frame and the target column name are returned. The dataset is split into training and testing sets using the 'train_test_split' function from the Scikit-learn library. This ensures an unbiased evaluation of the model's performance.

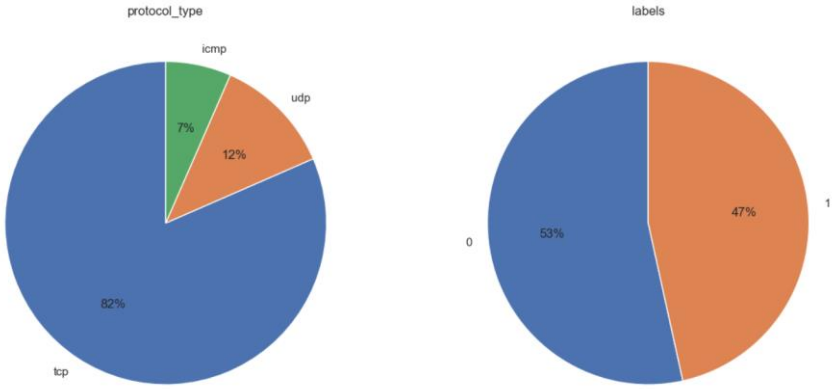

Figure 1: Pie chart of NSL-KDD[2] dataset distribution and the percentage of attack.

[2] https://www.kaggle.com/datasets/kiranmahesh/nslkdd

The pie chats above give the distribution of various protocol in the NSL-KDD[2] dataset. The dataset file analysed contains 82% of TCP protocol, 12% of UDP and, 7% of ICMP traffics. Another plot gives the attack percentage of attack vector. The label named as '1' defines the attack and '0' for non-attack patterns.

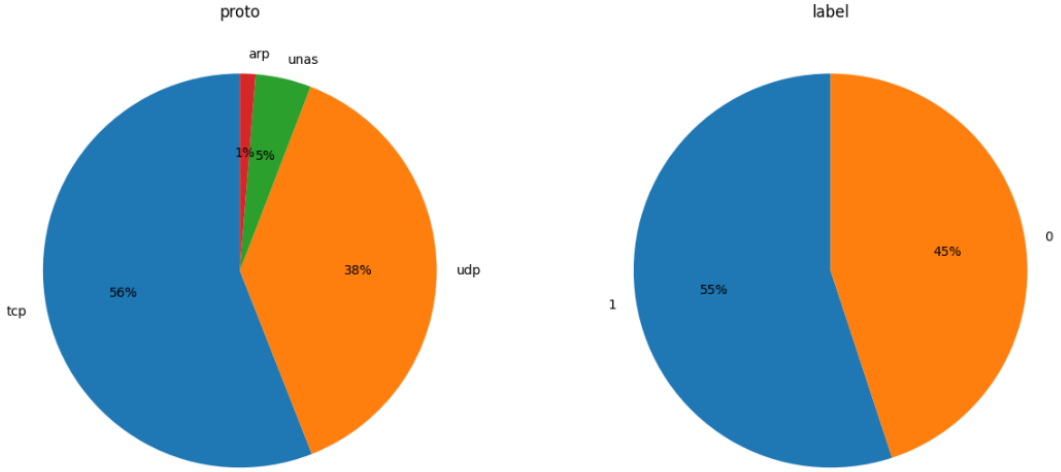

Figure 2: Pie chart of UNSW-NB15[3] dataset distribution and the percentage of attack.

The pie chats above give the distribution of various protocol in the UNSW-NB15**Error! Bookmark not defined.** dataset. The dataset file analysed contains 56% of TCP protocol, 38% of UDP, 5% of UNAS, and 1% of ARP traffics. Another plot gives the attack percentage of attack vector. The label named as '1' defines the attack and '0' for non-attack patterns.

References

Jupyter. (n.d.). *Installing Jupyter*. Retrieved from Jupyter: https://jupyter.org/install

pipy.org. (n.d.). *matplotlib 3.8.2*. Retrieved from pipy.org: https://pypi.org/project/matplotlib/

pipy.org. (n.d.). *scikit-learn 1.3.2*. Retrieved from pipy.org: https://pypi.org/project/scikit-learn/

pipy.org. (n.d.). *seaborn 0.13.0*. Retrieved from pipy.org: https://pypi.org/project/seaborn/

pipy.org. (n.d.). *tensorflow 2.15.0*. Retrieved from pipy.org: https://pypi.org/project/tensorflow/

pypi.org. (n.d.). *numpy 1.26.2*. Retrieved from pypi.org: https://pypi.org/project/numpy/

pypi.org. (n.d.). *pandas 2.1.3*. Retrieved from pypi.org: https://pypi.org/project/pandas/

python.org. (n.d.). *Python 3.11.0*. Retrieved from Python: https://www.python.org/downloads/release/python-3110/

[3] https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15