Configuration Manual

MSc Research Project
Programme Name

Vaishali Arora
Student ID:x22109129

School of Computing
National College of Ireland

Supervisor: Ahmed Makki

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vaishali Arora
Student ID:	x22109129
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Ahmed Makki
Submission Due Date:	14/12/2023
Project Title:	Improving the precision of network intrusion detection in edge computing by incorporating optimizers with Bi-directional LSTM
Word Count:	1200
Page Count:	10

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:	
Date:	14th December 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):	

Configuration Manual

Vaishali Arora
x22109129

1 Introduction

By combining optimizers with Bi-directional LSTM, this paper aims to improve the accuracy and precision of network intrusion detection in edge computing and develop a setup manual that details the research work done in detail. The primary motivation behind selecting network intrusion systems as a research topic is to develop a practical algorithm that can reduce attacker traffic to the lowest feasible level. The goal in this case was to refine an algorithm to the point where it could handle external attacks with nearly 99% accuracy. The UNSW-NB15 test data set was used to experiment. There are approximately 175,341 records in the training set and 82,332 records in the testing data set. Various attack kinds are captured in these records. The dataset in question is analyzed using a variety of deep learning models, including LSTM, Bi-LSTM, and Bi-LSTM with PSO. The most optimal algorithm is determined by comparing all three approaches based on several criteria, including accuracy and precision.
The report's remaining sections are arranged as follows: System configuration, which covers hardware and software requirements, is covered in Section 2, while project implementation is covered in Section 3.

2 System Configuration

2.1 Software Specification

- **Google Colaboratory:** Cloud-based Jupyter notebook, Python version **3.10.12** *Python Tutorial* (n.d.) Radovanovic (2022)
- **Email:** Gmail account needed for accessing the drive.
- **Browser:** Any web browser.

2.2 Hardware Requirement

2.2.1 The following hardware was used for the experiment:

- **Processor:** intel Core i5
- **RAM:** 16 GB
- **System Type:** 64-bit operating system

2.2.2 The bare minimum of hardware needs is:

- **Windows:** Window 7 or 10
- **RAM:** 4 GB
- **Disk Space:** 5GB free disk space

3 Project Implementation

3.1 Environment Setup

Having a Gmail account is a prerequisite to using the Google Collaboration. After creating an account, follow the instructions below.

Step 1: Navigate to drive to the folder where the ipynp file with the dataset file is available.
Step 2: Open the notebook file.

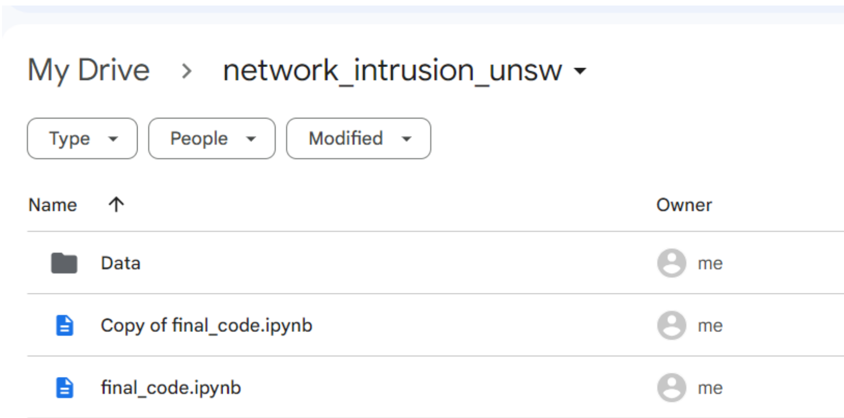

Figure 1: Section 3.1,Step1: ipynb file and dataset file in gdrive

Step 3: Connect the file to Google Collaboratory. Click on **Run all** in runtime. Import the drive using below code mentioned in the screenshot below.

3.2 Package and Libraries

After the successful mounting of the drive in Google Collab, the following libraries are imported before continuing with the code implementation. The following is the list of the needed libraries for execution:

- **NumPy:** Scientific computing library for handling large, multi-dimensional arrays.
- **Pandas:** Data manipulation library providing DataFrame structures.
- **Pickle:** Serialization and deserialization module for Python objects.
- **Six:** Python 2 and 3 compatibility library.

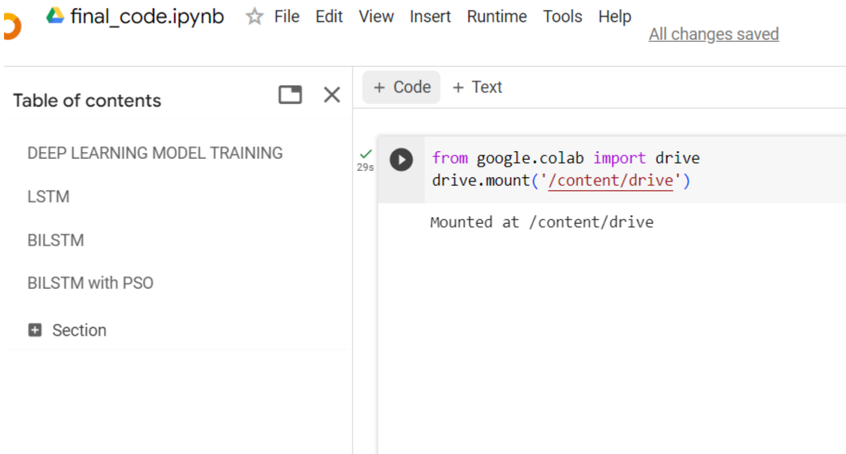

Figure 2: Section 3.1,Step3:Mounting Google Drive in Google Collab

- **Sys:** System-specific parameters and functions.
- **Scikit-learn (Sklearn):** Machine learning library with various tools for classification, regression, clustering, etc.
- **OS (from os import path):** Module for interacting with the operating system.
- **Seaborn:** Data visualization library based on Matplotlib.
- **Plotly:** Interactive plotting library for creating interactive, web-based visualizations.
- **Matplotlib:** 2D plotting library for Python.
- **Scikit-learn Ensemble:** Module for building ensemble models.
- **Keras:** High-level neural networks API.
- **TensorFlow:** Open-source machine learning framework.
- **Adam, SGD, RMSprop:** Optimizers for training neural networks.
- **Sequential:** Linear stack of layers in Keras.
- **Conv1D, MaxPooling1D, Flatten, Dense:** Layers for building Convolutional Neural Networks (CNNs).
- **Dropout, Activation:** Layers for preventing overfitting in neural networks.
- **LSTM (Long Short-Term Memory):** Recurrent neural network layer for sequence prediction.
- **Bidirectional:** Wrapper for making a RNN layer bidirectional.
- **Min-Max Scaler, Standard Scaler:** Preprocessing tools for scaling features.
- **LabelEncoder:** Encoding categorical features as integers.

- **EarlyStopping:** Callback for stopping training when a monitored metric has stopped improving.
- **Classification Report, Accuracy Score, Confusion Matrix, Precision-Recall-Fscore-Support:** Metrics for evaluating model performance.
- **Pyswarms:** Optimization library inspired by swarm intelligence principles.
- **IPython Display:** Module for displaying rich media representations in IPython.

```
[4] #importing all libraries
    import numpy as np
    import pandas as pd
    import pickle
    import six
    import sys
    import sklearn
    from os import path
    import seaborn as sns
    from sklearn import preprocessing
    import plotly.graph_objects as go
    import plotly.express as px
    import plotly.figure_factory as ff
    import matplotlib.pyplot as plt
    from sklearn import ensemble
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.preprocessing import StandardScaler
    from sklearn.preprocessing import LabelEncoder
    from keras.models import Sequential
    from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
    from keras.layers import Dropout, Activation
    from keras.layers import Convolution1D
    import tensorflow as tf
    from tensorflow.keras.optimizers import Adam, SGD, RMSprop
    import keras
    from keras import Model, Sequential, backend
    from keras.layers import LSTM, Dense, Dropout, Bidirectional
    from keras.callbacks import EarlyStopping
    from keras.layers import Input
    from keras.utils import to_categorical
    from sklearn.metrics import classification_report,accuracy_score,confusion_matrix, precision_recall_fscore_support
```

```
[5] from pyswarms.utils.plotters import plot_cost_history, plot_contour, plot_surface
    from pyswarms.utils.plotters.formatters import Mesher, Animator
    from pyswarms.utils.plotters.formatters import Designer
    import matplotlib.pyplot as plt
    from IPython.display import Image
    import pyswarms
    from keras.utils import plot_model
    import keras
    from keras.models import Sequential
    from keras.layers import Activation, LSTM, Dense, Flatten, Dropout, Bidirectional
    import numpy
```

Figure 3: Section:3.2:Required Libraries

4 Phases

The methodology followed for the implementation of the network intrusion detection algorithm is as follows:-

4.1 Data Loading

Figure 4, is presenting here how csv dataset is loaded into the pandas' data frame named 'df'.

```
#loading dataset
df = pd.read_csv('/content/drive/MyDrive/network_intrusion_unsw/Data/UNSW_NB15_training-set.csv')
df.head()
```

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd	ct_src_ltm	ct_srv_dst	is_sm_i
0	1	0.000011	udp	-	INT	2	0	496	0	90909.0902	...	1	2	0	0	0	1	2	
1	2	0.000008	udp	-	INT	2	0	1762	0	125000.0003	...	1	2	0	0	0	1	2	
2	3	0.000005	udp	-	INT	2	0	1068	0	200000.0051	...	1	3	0	0	0	1	3	
3	4	0.000006	udp	-	INT	2	0	900	0	166666.6608	...	1	3	0	0	0	2	3	
4	5	0.000010	udp	-	INT	2	0	2126	0	100000.0025	...	1	3	0	0	0	2	3	

5 rows × 45 columns

Figure 4: Section 4.1: Data Collection

4.2 Dataset Exploration

Figure 5, is presenting the structure of the dataset.

```
[ ] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 82332 entries, 0 to 82331
Data columns (total 45 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 82332 non-null  int64
 1   dur                82332 non-null  float64
 2   proto              82332 non-null  object
 3   service            82332 non-null  object
 4   state              82332 non-null  object
 5   spkts              82332 non-null  int64
 6   dpkts              82332 non-null  int64
 7   sbytes             82332 non-null  int64
 8   dbytes             82332 non-null  int64
 9   rate               82332 non-null  float64
 10  sttl               82332 non-null  int64
 11  dttl               82332 non-null  int64
 12  sload              82332 non-null  float64
 13  dload              82332 non-null  float64
 14  sloss              82332 non-null  int64
 15  dloss              82332 non-null  int64
 16  sinpkt             82332 non-null  float64
 17  dinpkt             82332 non-null  float64
 18  sjit               82332 non-null  float64
 19  djit               82332 non-null  float64
 20  swin               82332 non-null  int64
 21  stcpb              82332 non-null  int64
 22  dtcpb              82332 non-null  int64
 23  dwin               82332 non-null  int64
 24  tcprtt             82332 non-null  float64
 25  synack             82332 non-null  float64
 26  ackdat             82332 non-null  float64
 27  smean              82332 non-null  int64
 28  dmean              82332 non-null  int64
 29  trans_depth        82332 non-null  int64
 30  response_body_len  82332 non-null  int64
 31  ct_srv_src         82332 non-null  int64
 32  ct_state_ttl       82332 non-null  int64
 33  ct_dst_ltm         82332 non-null  int64
 34  ct_src_dport_ltm   82332 non-null  int64
 35  ct_dst_sport_ltm   82332 non-null  int64
 36  ct_dst_src_ltm     82332 non-null  int64
 37  is_ftp_login       82332 non-null  int64
 38  ct_ftp_cmd         82332 non-null  int64
 39  ct_flw_http_mthd   82332 non-null  int64
 40  ct_src_ltm         82332 non-null  int64
 41  ct_srv_dst         82332 non-null  int64
 42  is_sm_ips_ports    82332 non-null  int64
 43  attack_cat         82332 non-null  object
 44  label              82332 non-null  int64
dtypes: float64(11), int64(30), object(4)
memory usage: 28.3+ MB
```

Figure 5: Section 4.2: Defining dataset columns

4.3 Dataset Validation

Figure 6, checks if we have any missing values in the dataframe.

```
df.isnull().sum()
```

```
id                   0
dur                  0
proto                0
service              0
state                0
spkts                0
dpkts                0
sbytes               0
dbytes               0
rate                 0
sttl                 0
dttl                 0
sload                0
dload                0
sloss                0
dloss                0
sinpkt               0
dinpkt               0
sjit                 0
djit                 0
swin                 0
stcpb                0
dtcpb                0
dwin                 0
tcprtt               0
synack               0
ackdat               0
smean                0
dmean                0
trans_depth          0
response_body_len    0
ct_srv_src           0
ct_state_ttl         0
ct_dst_ltm           0
ct_src_dport_ltm     0
ct_dst_sport_ltm     0
ct_dst_src_ltm       0
is_ftp_login         0
ct_ftp_cmd           0
ct_flw_http_mthd     0
ct_src_ltm           0
ct_srv_dst           0
is_sm_ips_ports      0
attack_cat           0
label                0
```

Figure 6: Section 4.3: Checking up on missing values

4.4 DataFrame Loading and Data Type Segmentation

Figure 7, depicts the type of data stored in each column.

```
features_df = pd.read_csv('/content/drive/MyDrive/network_intrusion_unsw/Data/NUSW-NB15_features.csv',encoding='cp1252')
```

```
features_df.head()
```

	No.	Name	Type	Description
0	1	srcip	nominal	Source IP address
1	2	sport	integer	Source port number
2	3	dstip	nominal	Destination IP address
3	4	dsport	integer	Destination port number
4	5	proto	nominal	Transaction protocol

```
features_df['Type '] = features_df['Type '].str.lower()
```

```
# selecting column names of all data types
nominal_names = features_df['Name'][features_df['Type ']=='nominal']
integer_names = features_df['Name'][features_df['Type ']=='integer']
binary_names = features_df['Name'][features_df['Type ']=='binary']
float_names = features_df['Name'][features_df['Type ']=='float']
```

```
cols = df.columns
nominal_names = cols.intersection(nominal_names)
integer_names = cols.intersection(integer_names)
binary_names = cols.intersection(binary_names)
float_names = cols.intersection(float_names)
```

```
for c in integer_names:
  pd.to_numeric(df[c])
```

```
for c in binary_names:
  pd.to_numeric(df[c])
```

```
for c in float_names:
  pd.to_numeric(df[c])
```

Figure 7: Section 4.4: Type-Based Column Selection

4.5 Data Visualization

Various ways in which data is presented using a plotly library.
Figure 8, showcases the distribution of all types of attacks.

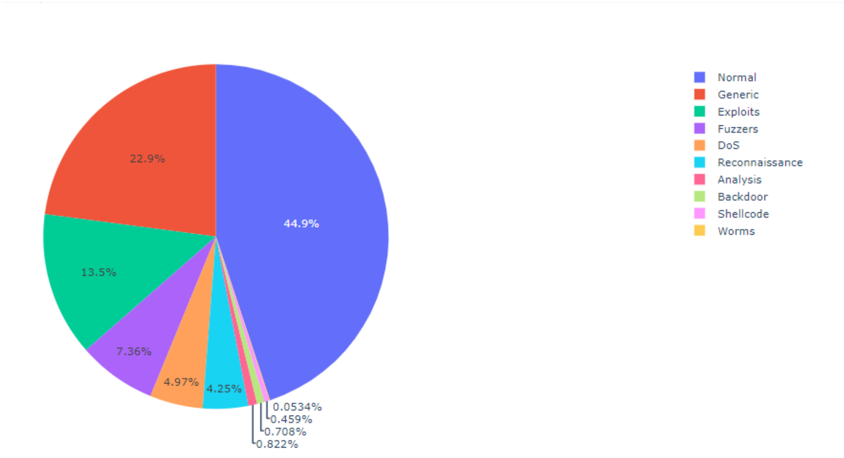

Figure 8: Section 4.5: attack_type percentage

Figure 9, represents a bar chart where the x-axis represents the unique label and the y-axis represents the count.

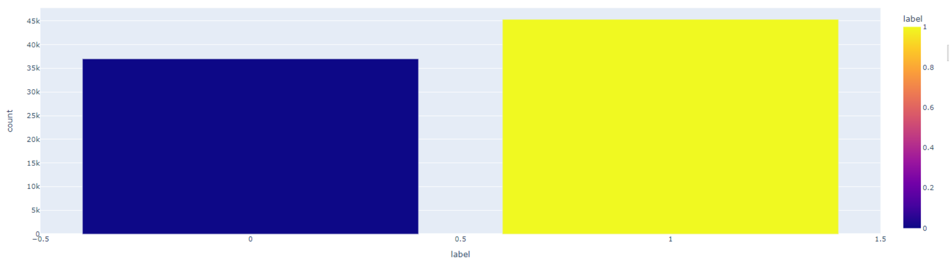

Figure 9: Section 4.5: Bar chart presentation for label field

Figure 10, the code creates and displays a histogram using Plotly Express, visualizing the data distribution of the 'rate' column in the DataFrame df with a blue color scheme.

Figure 11, presents state wise count for each sort of attack.

4.6 Data Pre-processing

During this stage, numerical and categorical data were handled. Below are step-by-step processes for the same:-

- A New frame containing only categorical columns is created.
- Label encoder is applied to transform categorical values('proto' and 'state') into numerical representation in 'data_cat'.
- Generate and visualize the correlation matrix for selected numeric columns.
- Presenting a co-relation matrix of multiple labels.
- Split the data into training and testing sets.

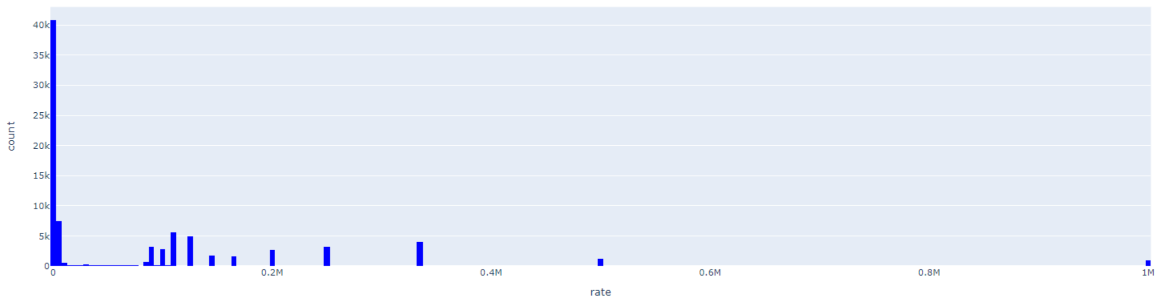

Figure 10: Section 4.5: Histogram for data distribution of ’rate’ column

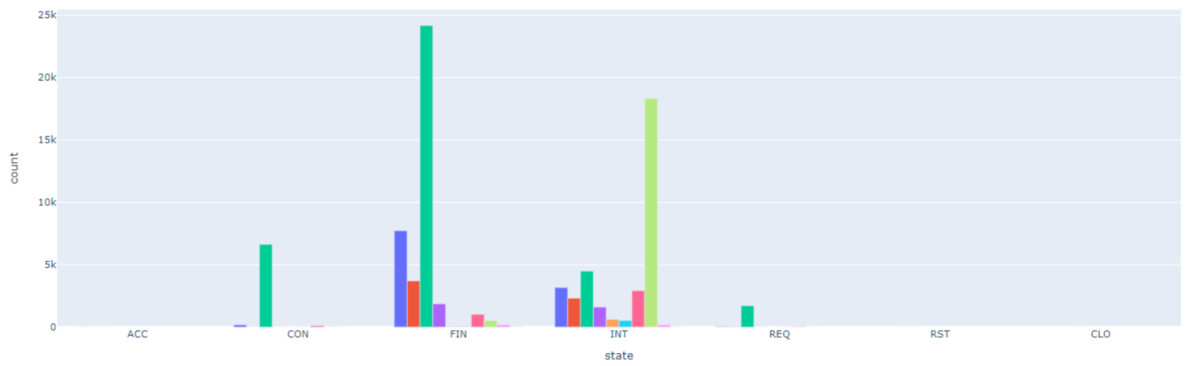

Figure 11: Section 4.5: value count of attack_cat wise state

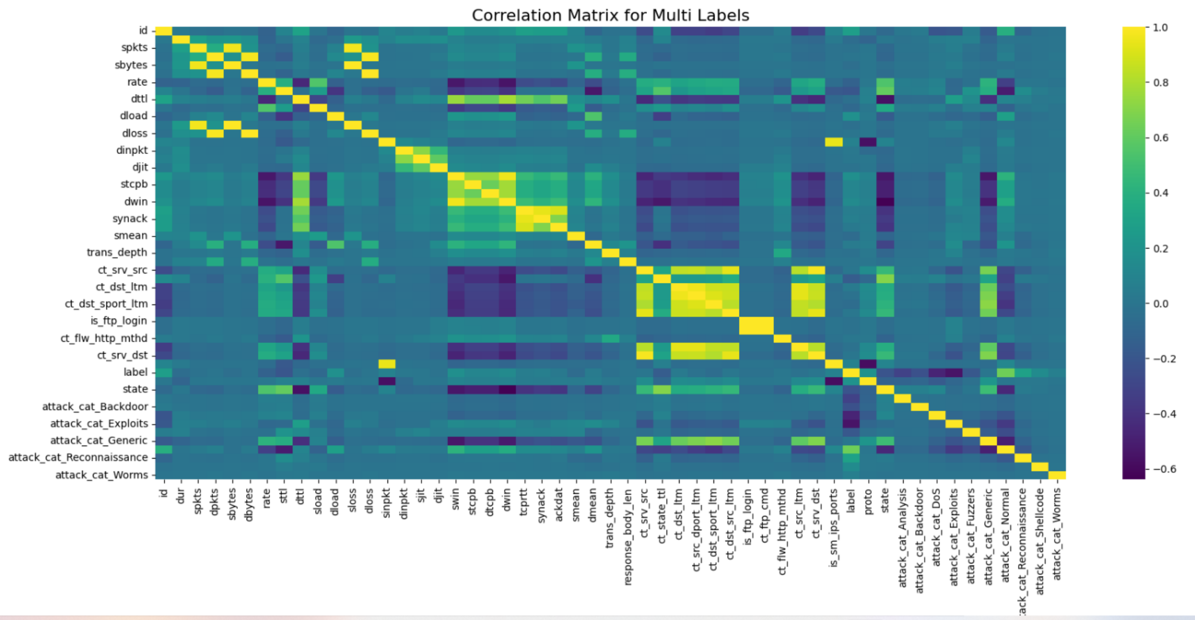

Figure 12: Section 4.6: Co-relation matrix of multiple labels

4.7 Feature Importance

The extra Trees Regressor Model is used here to select the top 25 features and create a dataset using those features. The code is then one-hot encoding the target variable and extracting the index of the maximum value to obtain class labels in the original format.

```
model = ensemble.ExtraTreesRegressor(n_estimators=50, max_depth=30, max_features=0.3, n_jobs=-1, random_state=0)
model.fit(X_train,y_train)
#plot imp
importance = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_],axis=0)
indices = np.argsort(importance)[::-1][:25]
plt.figure(figsize=(7,7))
plt.title("Feature importances")
plt.bar(range(len(indices)), importance[indices], color="g")
plt.xticks(range(len(indices)), fearture_name[indices], rotation='vertical')
plt.xlim([-1, len(indices)])
plt.show()
```

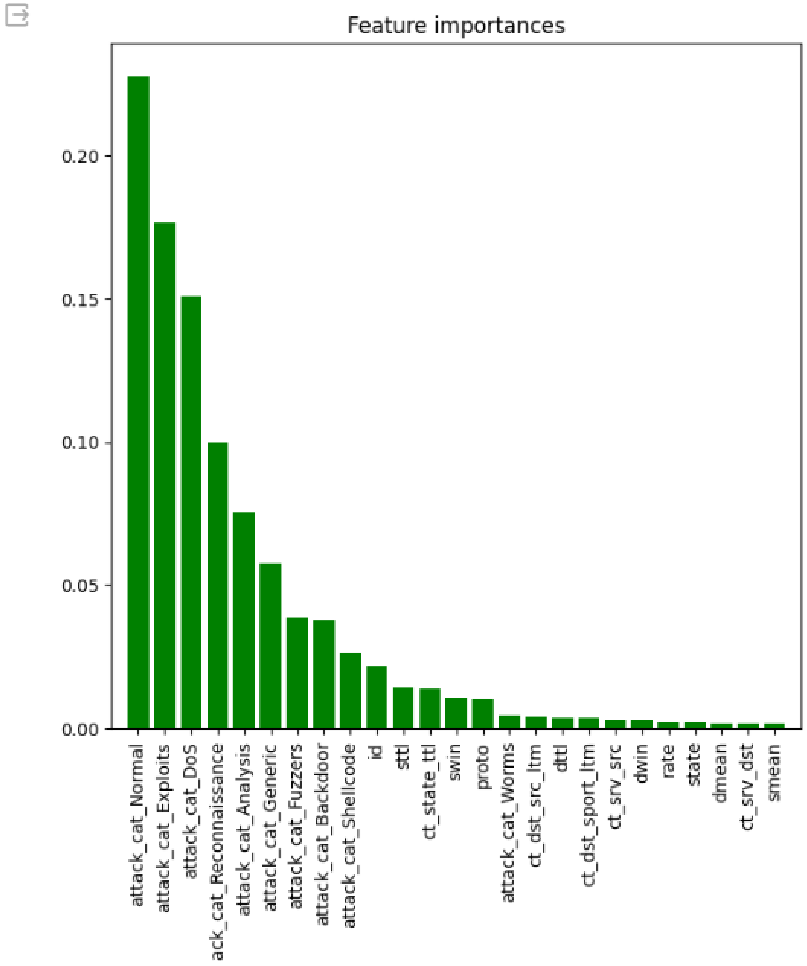

Figure 13: 4.7: Feature Importance

4.8 Comparison of Data Models

Comparison for all the mentioned models is done based on accuracy, confusion matrix, and classification report.

4.8.1 LSTM Model

LSTM model is implemented using the Keras library in Python. It is a type of RNN architecture designed to capture and learn patterns in sequences of data. LSTM is considered the best model for sequential data.

LSTM model was able to achieve 82% accuracy, after the successful implementation of the algorithm.

4.8.2 BILSTM Model

The bi-LSTM Model is an extension of the traditional LSTM that processes input sequences in both forward and backward directions. This is also implemented using Keras. BiLSTM model was able to achieve 97% accuracy, after the successful implementation of the algorithm.

4.8.3 BILSTM with PSO Model

PSO is an optimizer that is integrated with the Bi-LSTM Model to improve performance. PSO is used as hyperparameter tuning. For PSO, pyswarm library is used. BiLSTM with PSO model was able to achieve 98 to 99% on optimization in every run.
COMPARING RESULTS ON ALL MODELS - LSTM, Bi-LSTM, Bi-LSTM with PSO

Table 1: Comparison of Classification Performance for DL Models (LSTM, BiLSTM, and BiLSTM with PSO)

Model	Accuracy	wtd. Avg Precision	wtd. Avg Recall	wtd. Avg F1-Score
LSTM	0.82	0.76	0.82	0.76
BiLSTM	0.97	0.96	0.97	0.96
BiLSTM with PSO	0.98	0.97	0.98	0.97

Python Tutorial (n.d.) Radovanovic (2022)

References

Python Tutorial (n.d.).
URL: *https://docs.python.org/3/tutorial/*

Radovanovic, I. (2022). Google colab - a step-by-step guide - algotrading101 blog.
URL: *https://algotrading101.com/learn/google-colab-guide/*