

# Configuration Manual

MSc Research Project  
Cloud Computing

Nilam Choudhari  
Student ID: x20154003

School of Computing  
National College of Ireland

Supervisor: Mr. Aqueel Qazmi

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Nilam Choudhari
<b>Student ID:</b>	x20154003
<b>Programme:</b>	Cloud Computing
<b>Year:</b>	2021
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Mr.Aqueel Qazmi
<b>Submission Due Date:</b>	16/12/2021
<b>Project Title:</b>	Configuration Manual
<b>Word Count:</b>	XXX
<b>Page Count:</b>	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.

<b>Signature:</b>	
<b>Date:</b>	15th December 2021

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

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

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

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Nilam Choudhari  
x20154003

## 1 Introduction

This article expands on the specifications of proposed system , as well as the software and hardware utilized in the implementation of the project. Also it outlines the procedures taken in the development of the research project, "Detection and analysis of Network Layer Security challenges in cloud using machine-learning Algorithm."

## 2 System Configuration

### 2.1 Software Specification

Jupyter Notebook-The downloaded data was split into training, testing, and validation portions using Jupyter Notebook, an open source program.

Python 3.7(64 bit)

### 2.2 Hardware Specification

Lenovo IdeaPad C340, 256 GB SSD, 8 GB RAM. Processor: 1.8 GHz, Intel Core, i5

Below figure shows the anaconda prompt used for accessing Jupyter notebook as shown in figure 1

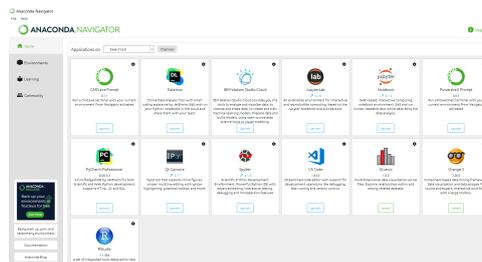


Figure 1: Anaconda prompt

I have downloaded all the libraries and listed them in the conda prompt as shown in figure 2. 1) matplotlib - this was used for data visualization and plotting of graphs 2) Pandas for data analysis and also manipulation of data 3)Scikit learn is the machine learning library



Extracted feature names from the kddcup names file. Since it does contain any name for the label, we will add a column name 'connection-type' for the labels (target variable)

dst_host_same_srv_port_size	dst_host_srv_diff_host_size	dst_host_serror_size	dst_host_srv_serror_size	dst_host_rerror_size	dst_host_srv_rerror_size	connection_type
1.00	0.0	0.0	0.0	0.0	0.0	normal
0.00	0.0	0.0	0.0	0.0	0.0	NPT
0.00	0.0	0.0	0.0	0.0	0.0	normal
0.25	0.0	0.0	0.0	0.0	0.0	normal
0.20	0.0	0.0	0.0	0.0	0.0	normal
0.17	0.0	0.0	0.0	0.0	0.0	normal
0.14	0.0	0.0	0.0	0.0	0.0	normal
0.12	0.0	0.0	0.0	0.0	0.0	NPT
0.11	0.0	0.0	0.0	0.0	0.0	normal
0.10	0.0	0.0	0.0	0.0	0.0	NPT

Figure 5: Coloum mislabelled

```
In [8]: kddcup_data.rename(columns={kddcup_data.columns[4]: "connection_type", inplace = True})
kddcup_data = kddcup_data.drop(["", ...])
```

Figure 6: Renaming and dropping

Then we checked the datatypes for training data

```
In [9]: kddcup_data.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4898431 entries, 0 to 4898430
Data columns (total 42 columns):
 #   Column                Dtype
---  ---
 0  duration              int64
 1  protocol_type         object
 2  service               object
 3  flag                  object
 4  src_bytes             int64
 5  dst_bytes             int64
 6  state                int64
 7  wrong_fragment       int64
 8  urgent               int64
 9  host                 int64
10  num_failed_logins    int64
11  logged_in            int64
12  num_compromised     int64
13  root_shell           int64
14  su_attempted         int64
15  num_root             int64
16  num_files_created   int64
17  num_files_deleted   int64
18  num_access_files    int64
19  num_successful_pads int64
20  is_login_attempt    int64
21  is_login_success    int64
22  count                int64
23  serror              float64
24  serror_rate         float64
25  srv_serror_rate     float64
26  rerror              float64
27  srv_rerror_rate     float64
28  wrong_rerror_rate   float64
29  diff_srv_rerror_rate float64
30  diff_rerror_rate    float64
31  diff_host_count     int64
32  diff_host_srv_count int64
33  diff_host_same_srv_rate float64
```

Figure 7: Data types

Then checked the size of the data as shown in figure 8

Later checked missing values in dataset as shown in figure 9

Repeated the same steps for the Testing dataset as well. Checked the sizes of both test and training dataset as shown in figure 10

Later combined both train and test data as shown in figure 11

Next we want to perform numeric encoding on the categorical features. Pandas has a function get-dummies() for this. This function adds additional columns based on the categories present in the feature.As shown in figure 12.

observed that there is an extra '.' in the 'connection-type' column after the connection type. For better visualization and interpretation, removed the '.'. This is shwn in figure 13.

Later we checked for duplicates and removed them from the training dataset.As shiwn in figure 14

Originally, we had 4898431 rows in the data set. After removing the duplicates, we have 1074992 rows. This shows that there were too many redundant rows in the data set. Since this data is a TCP dump, having same values is very much expected.

Next, we scale all the numeric features between 0 and 1. We are given a detailed description of the data set features denoting which features are continuous and which are discrete along with the data types.



```
In [28]: from sklearn.preprocessing import MinMaxScaler

numeric_features = train_data.dtypes[train_data.dtypes != "object"].index
min_max_scaler = MinMaxScaler()
train_data[numerical_features] = min_max_scaler.fit_transform(train_data[numerical_features])
train_data.head()
```

```
Out[28]:
```

	duration	sec_buys	sec_hires	last_working_hour	urgent	hot	num_viol_buys	logged_in	num_complaints	Reg_R1C2	Reg_R1C3	Reg_R1C4
0	0.0	1.558102e-07	3.441000e-06	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
1	0.0	1.173844e-06	3.458054e-06	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
2	0.0	1.718104e-06	6.274846e-06	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
3	0.0	1.688454e-06	1.551709e-06	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
4	0.0	1.731833e-06	3.718034e-06	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0

Score: = 124 columns

Figure 15: scaling to 0 and 1

Here, we are dealing with binary classification i.e. whether a connection is an attack or not. In our data set, columns are labelled as either 'normal' or as the attack type. So we need to need to denote normal connections as one class type and all the attack types as another class. The new class label would be 0 if the connection is normal and 1 if it is an attack.

```
In [29]: train_data["label"] = train_data["connection_type"] != "normal"
train_data["label"] = train_data["label"].apply(lambda x: int(x))
train_data.head()
```

```
Out[29]:
```

	label
0	0
1	0
2	0
3	0
4	0

Name: label, dtype: int64

Let's see how the attack types are distributed over the data set

Figure 16: Binary classification of data

### 3.3 Models

The snippets in this section will be all about the different models that were used for implementing the project

```
NUMERIC_HEADERS
```

```
In [30]: from sklearn.metrics import GradientBoostingClassifier, RandomForestClassifier
from sklearn.linear_model import LogisticRegression, Perceptron
from sklearn.metrics import accuracy_score, roc_curve
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import MinMaxScaler
```

```
In [31]: train_data = train_data.drop("connection_type", axis=1)
train_data.head()
```

```
Out[31]:
```

	duration	sec_buys	sec_hires	last_working_hour	urgent	hot	num_viol_buys	logged_in	num_complaints	Reg_R1C2	Reg_R1C3	Reg_R1C4
0	0.0	1.558102e-07	3.441000e-06	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
1	0.0	1.173844e-06	3.458054e-06	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
2	0.0	1.718104e-06	6.274846e-06	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
3	0.0	1.688454e-06	1.551709e-06	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
4	0.0	1.731833e-06	3.718034e-06	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0

5 rows x 124 columns

Figure 17: Library import for model training

Accuracy of Multi layer perceptron is highest than other models

We have also used ROC curve and AUC here for evaluating the results. They are used to determine best cut-off which has the lowest false positive rate and the greatest genuine positive rate which is true positive.

```
In [40]: print("For Logistic Regression Classifier:\nAccuracy Score is: ")
accuracy_score(test_data["label"], logreg.predict(test_data.iloc[:,1:])) * 100
For Logistic Regression Classifier:
Accuracy Score is:
Out[40]: 91.96425188275688
```

Figure 18: Logistic Regression Accuracy

```
In [41]: model.append("Logistic Regression")
fpr, tpr, thresholds = roc_curve(test_data["label"], logreg.predict(test_data.iloc[:,1:]))
roc_auc = auc(fpr, tpr)
fpr.append(fpr)
tpr.append(tpr)
roc_auc.append(roc_auc)

In [42]: random_forest = RandomForestClassifier().fit(train_data.iloc[:,1:], train_data["label"])
random_forest

Out[42]: RandomForestClassifier()

In [43]: print("For Random Forest Classifier:\nAccuracy Score is: ")
accuracy_score(test_data["label"], random_forest.predict(test_data.iloc[:,1:])) * 100
For Random Forest Classifier:
Accuracy Score is:
Out[43]: 92.58718648384616
```

Figure 19: Random forest accuracy

```
In [44]: gradient_boosting = GradientBoostingClassifier().fit(train_data.iloc[:,1:], train_data["label"])
gradient_boosting

Out[44]: GradientBoostingClassifier()

In [45]: print("For Gradient Boosting Classifier:\nAccuracy Score is: ")
accuracy_score(test_data["label"], gradient_boosting.predict(test_data.iloc[:,1:])) * 100
For Gradient Boosting Classifier:
Accuracy Score is:
Out[45]: 92.68826584282846
```

Figure 20: Gradient Boosting Accuracy

```
In [46]: print("For Perceptron Classifier:\nAccuracy Score is: ")
accuracy_score(test_data["label"], perceptron.predict(test_data.iloc[:,1:])) * 100
For Perceptron Classifier:
Accuracy Score is:
Out[46]: 81.53184718177464

In [48]: model.append("Perceptron")
fpr, tpr, thresholds = roc_curve(test_data["label"], perceptron.predict(test_data.iloc[:,1:]))
roc_auc = auc(fpr, tpr)
fpr.append(fpr)
tpr.append(tpr)
roc_auc.append(roc_auc)
```

Figure 21: Perceptron Accuracy

```
In [51]: multilayer_perceptron = MLPClassifier(hidden_layer_sizes=(100,)).fit(train_data.iloc[:,1:], train_data["label"])
multilayer_perceptron

Out[51]: MLPClassifier()

In [52]: print("For Multilayer Perceptron Classifier:\nAccuracy Score is: ")
accuracy_score(test_data["label"], multilayer_perceptron.predict(test_data.iloc[:,1:])) * 100
For Multilayer Perceptron Classifier:
Accuracy Score is:
Out[52]: 93.800114489335
```

Figure 22: Multi-layer Perceptron Accuracy

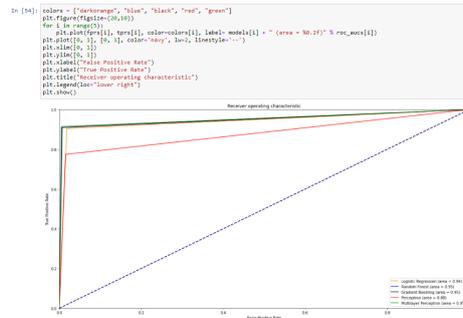


Figure 23: ROC Curve