

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



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

<u>ALL</u> 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:	15th December 2021

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

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



Figure 2: Conda list for libraries

3 Project Implementation

3.1 Data Collection

A A A A Manual Resolution Allowed Allo	
C G G Googe & Girld - Geografie. ♦ 182 cm - Cirile. ♦ 192 94 - Hyster. ▲ Maerier-Googe C. ♦ 265 unld2 - exc. 📓 Meaners Interc. В Gear Opt (Cr.20. +) В Green Control (Cr.20. +)	E Factory
KDD Cup 1999 Data	
Abstract	
The a the data set used for The That International Encodings Theorem y and Data Manage Their Competitions, which was held as sequencing with XXD 109 The Fully International Conference on Disording Decouvery and Data Manage The competitions and experience with XXD 109 The Fully International Conference on Disording Decouvery and Data Manage The competitions and experience with XXD 109 The Fully International Conference on Disording Decouvery and Data Manage The competitions and experience with XXD 109 The Fully International Conference on Disording Decouvery and Data Manage The competitions and experience with XXD 109 The Fully International Conference on Disording Decouvery and Data Manage The competitions and experience with XXD 109 The Fully International Conference on Disording Decouvery and Data Manage The competitions and experience with XXD 109 The Fully International Conference on Disording Decouvery and Data Manage The competitions and experience with XXD 109 The Fully International Conference on Disording Decouvery and Data Manage The competitions and experience with XXD 109 The Fully International Conference on Data Manage The competitions and experience on Data Manage Th	cia, a producio
laberaria fie:	
 task incorpora. This is the opposite task deciption gives to competition participants. 	
Duis line	
Standard Standar	
As ASSESSMENT AND A REAL PROVIDENT AND A	

Figure 3: KDD Dataset link

Dataset was picked from the KDD Cup 1999 Data. This data set was used for The Third International Knowledge Discovery and Data Mining Tools Competition, that was conducted in connection with KDD-99, The Fifth International Conference on Knowledge Discovery and Data Mining. The aim for the challenge was to build a comprehensive network intrusion detector, a prediction model capable of differentiating between "bad" connections, known as intrusions or assaults, and "good" regular connections. This database comprises a consistent standard of auditable data, including a wide range of intrusions replicated in a military network context.

3.2 Data preparation

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)

n (5):	<pre>index_uname = comm("biling.nemms").read().split('\n') forture_names = [1]. for forture_names = lid_disp_names[1:]: forture_names = lid_disp_names[1:]: forture_names = speen(forture_names_split('\')[0].strip())</pre>
	feature_names.append("connection_type")

Figure 4: Feature name extraction

After loading the training dataset we saw that the column 'connection-type' is misplaced in the dataset we need to make it appropriate as it is the 'target-variable'. Renamed the column no. 41 as 'connection-type' and dropped column number 42. 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)

pe	connection_typ		dat_host_arv_rerror_rate	dst_host_remor_rate	dst_host_srv_serror_rate	dst_host_serror_rate	dst_host_srv_diff_host_rate	thost_same_src_port_rate
WN.	Na	normal.	0.0	0.0	0.0	0.0	0.0	0.00
N,	No	eornal.	0.0	0.0	0.0	0.0	0.0	1.00
N,	Na	eornal.	0.0	0.0	0.0	0.0	0.0	0.50
N.	Na	eomal.	0.0	0.0	0.0	0.0	0.0	0.33
N	Na	normal.	0.0	0.0	0.0	0.0	0.0	0.25
eN.	Na	eornal.	0.0	0.0	0.0	0.0	0.0	0.20
N,	Na	eornal.	0.0	0.0	0.0	0.0	0.0	0.17
N.	No	normal.	0.0	0.0	0.0	0.0	0.0	0.14
N.	Na	normal.	0.0	0.0	0.0	0.0	0.0	0.12
N.	Na	normal.	0.0	0.0	0.0	0.0	0.0	0.11
N.	Na	eomal.	0.0	0.0	0.0	0.0	0.0	0.10

Figure 5: Coloum mislabelled

Figure 6: Renaming and dropping

(n [8]: kddcup_data.rename(columnsn{ kddcup_data.columns[41]: "connection_type" }, inplace = True)
kddcun_data = kddrum_data ilor[. ..]

Then we checked the datatypes for training data

In [9]:	ddcup_data.info()		
	class 'pandas.core.frame.Dat	frame')	
	langeIndex: 4898431 entries,	to 4896430	
	iste columns (totel 42 column		
	a Column	Otype	
	e duration	10104	
	1 protocol_type	object	
	2 service	abject	
	3 flag	object	
	4 src bytes	14164	
	5 dat bytes	int04	
	6 land	int04	
	7 wrong freement	1nt04	
	a urgent	1nt64	
	9 hot	5et64	
	10 num failed logiss	Setfis	
	11 larged in	Letting.	
	12 num composition	inter a	
	15 cost shell	10004	
	14 sw attempted	1et64	
	15 num root	int04	
	16 num file creations	int04	
	17 num shells	int64	
	10 num access files	1nt64	
	19 num outbound ceds	5et64	
	20 is host login	Setfis	
	21 Is puest login	let64	
	22 count	inter a	
	25 any count	10104	
	24 serror rate	floatf4	
	25 any serror rate	float64	
	26 recor cate	float64	
	27 SEV CRECOF CATE	floates	
	28 same sry rate	floatse	
	25 diff sev rate	floatte	
	30 srv diff host rate	float64	
	31 dst host count	14164	
	32 dst host sry count	10104	
	15 dat bast same any nate	f]cat14	

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 shown 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 [10]: kddcup_data.shape Out[10]: (4898431, 42)

Figure 8: Data size



Figure 9: Check of missing values in dataset



Figure 10: Size checking of training and testing data

Figure 11: Combining test and train dataset

In [21]: data - pd.concat([kddcup_data, kddcup_test_data])

	datation	arc_bytes	dat_bytes	land	wrong_fragment	argent	hot	num_failed_logins	logged_in	ram_compromised	flag_REJ	flag_RSTO	flag_RSTOS
0	0	215	45076	. 0	0	0	0	0	1		0	0	
1	0	162	4528			0	0	0	1			0	
2	0	235	1228			0	0	0	1			0	
3	0	233	2032				0		1			0	
4	0	239	466				0	0	1			0	

Figure 12: numeric encoding

In [25]:	train_data.connection_type.head()
Out[25]:	0 normal. 1 normal.
	2 normal.
	3 normal.
	4 normal.
	Name: connection_type, dtype: object
In [26]:	<pre>train_data['connection_type'] = train_data['connection_type'].apply(lambda x : str(x)[:-1]) train_data.connection_type.head()</pre>
Out[26]:	8 poreal
	2 pormal
	n
	A pormal
	4 normal
	4 normal Name: connection_type, dtype: object

Figure 13: Correction

Figure 14: Duplicate removal

train_data = tr train_data.shep

tra	in_data in_data	[numeric_ .head()	features]	- már	_max_scaler.t	assfor	w(tr	ain_data[numeri	c_feature	s])			
	duration	arc_bytes	det_bytes	land	wrong_fragment	urgent	hot	nem_failed_logins	logged_in	num_compromised	flag_REJ	fag_RSTO	flag_RS1
0	0.0	1.558812e- 07	3.441050e- 65	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	
1	0.0	1.1739446-07	3.455554e- 06	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	
2	0.0	1.7101908-07	9.374494e- 07	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	
3	0.0	1.608458e. 07	1.551219e- 05	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	
	0.0	17319298-	3.710101e-	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	

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.



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

		ouer in	anning											
In [33]:	fre fre fre fre	m sklea m sklea m sklea m sklea m sklea	rn.ensemb) rn.linear, rn.metric rn.meural, rn.prepros	le import model imp i import a network i cessing im	Gradi ort L cours sport port	entBoosting(la ogisticRegress cy_score, auc, HUPClassifier HinHaxScaler	ssifie ion, P roc_c	r, R erce urve	andomForestClas ptron	sifier				
In [34])	tra tra	sin_data sin_data	- train_d	lata.drop(con	ection_type",	axis•1)						
Out[34]:		duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	nun_compromised	 flag_RSTO	file_RSTOSE	file_R
	0	0.0	1.550012a- 07	3.441050e- (6	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	
	ł	0.0	1.1739448- 07	3.455654e- 06	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	
	2	0.0	1.710190e- 07	9.374494e. 07	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	
	3	0.0	1.658450e- 07	1.551219e- 06	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	
	4	0.0	1.731929e- 97	3.710101e- 07	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	
	5 n	ours × 12-	l 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 areused to determine best cut-off which has the lowest false positive rate and the greatest genuine positive rate which is true positive.

In [40]:	<pre>print("For Logistic Regression classifier:\waccuracy_Score is: ") accuracy_score(test_data["label"], logreg.predict(test_data.iloc[:,:-1])) * 100</pre>
	For Logistic Regression classifier: Accuracy Score is:
Dut(40):	91.98429188275848

Figure 18: Logistic Regression Accuracy



Figure 19: Random forst accuracy

In [45])	<pre>gradient_boosting = GradientBoostingClassifier().fit(train_data.iloc[.,.i], train_data["label"]) gradient_boosting</pre>
Out[45]:	GradientBoostingClassifier()
In [46]:	<pre>prist("for Gradient Boosting classifier:\wkccuracy_Score is: ") accuracy_score(test_data["label"], gradient_boosting.predict(test_data.iloc[:,:-1])) * 100</pre>
	For Gradient Boosting classifier: Accuracy Score is:
Out[46])	22.00030345433549

Figure 20: Gradient Boosting Accuracy

In [49]:	<pre>print("for Perceptron classifier:\nAccuracy Score is: ") accuracy_score(test_data["label"], perceptron.predict(test_data.iloc[:,:-1])) * 100</pre>
	For Perceptron classifier: Accuracy Score is:
Out[49]:	81.53194718177404
In [50]:	<pre>imable.tegene("terrogram") ("re_ter_termine"), proceptron.predict(test_data.iloc[:,:-1])) (re_ter_termine", terp) ("re_ter_termine"), rep("ter_termine"), representation ("tert_termine"), representation ("tert_termine"), representation ("tert_termine"), representation ("tert_termine"), representation ("termine"), representation ("te</pre>

Figure 21: Perceptron Accuracy

In [51]:	<pre>multilayer_perceptron = MLPClassifier(hidden_layer_sizes=(100,)).fit(train_data.iloc[:,:-1], train_data["label"]) multilayer_perceptron</pre>
Out[51]:	NuPClassifier()
In [52]:	<pre>print("for Multilayer Perceptron classifier:\nkccuracy Score is: ") accuracy_score(test_data["label"], wultilayer_perceptron.predict(test_data.iloc[:,:-1])) * 100</pre>
	For Hultilayer Perceptron classifier: Accuracy Score is:
Out[52])	95.000115400535

Figure 22: Multi-layer Perceptron Accuracy



Figure 23: ROC Curve