Configuration Manual

MSc Research Project
MSc in Cloud Computing

Zeba Mahfuz
Student ID: x22226885

School of Computing
National College of Ireland

Supervisor: Jorge Mario Cortes Mendoza

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Zeba Mahfuz
Student ID:	x22226885
Programme:	MSc in Cloud Computing
Year:	2023-2024
Module:	Research in Computing
Supervisor:	Jorge Mario Cortes Mendoza
Submission Due Date:	16/09/2024
Project Title:	Machine Learning Approaches to analyze Intrusion Detection Risk under Cloud Computing
Word Count:	490
Page Count:	5

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:	Zeba Mahfuz
Date:	16-09-2024

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

Zeba Mahfuz
x22226885

16-09-2024

1 Introduction

1.1 Purpose of the Manual

This configuration manual offers in depth instructions on how to establish and customize an intrusion detection system based on machine learning within a computing setting. It features step by step guidance and visual aids to help users effectively implement and manage the system.

1.2 Overview of the Project

The project is about using machine learning methods to examine network traffic and identify possible security risks in cloud setups. By using algorithms such as K Nearest Neighbors (KNN) and Multilayer Perceptron (MLP) the system categorizes network actions as either normal or intrusive boosting cloud security, with risk evaluation and identification.

2 System Configuration

2.1 Hardware Requirements

- Processor: Minimum 2.4 GHz (Intel Core i5 or higher recommended)
- RAM: At least 8 GB
- Storage: Minimum 20 GB of free space
- Operating System: 64-bit (Windows, macOS, Linux)

2.2 Software Requirements

- Python: Version 3.8 or higher
- Jupyter Notebook: Accessible via Google Colab or installed locally

- Required Libraries:
Scikit-learn: Version 0.24.2 - for machine learning algorithms and data preprocessing.
Pandas: Version 1.3.3 - for data manipulation and handling.
NumPy: Version 1.20.3 - for numerical operations and data processing.
Matplotlib: Version 3.4.3 - for data visualization and creating plots.
Seaborn: Version 0.11.1 - for enhanced statistical data visualization.

- Cloud Platform: Google Colaboratory or an equivalent cloud-based Jupyter notebook environment

3 Project Setup

3.1 Environment Setup

1. **Google Colaboratory:**
   - Access Google Colaboratory at: Google Colab
   - Create a new notebook for coding and executing the project.
   - Import necessary libraries and datasets as shown in the code below.
2. **Local Environment Setup:**
   - Install Anaconda for package and environment management.
   - Create a virtual environment and install the required packages.
   - Launch Jupyter Notebook or your preferred IDE for running the project.

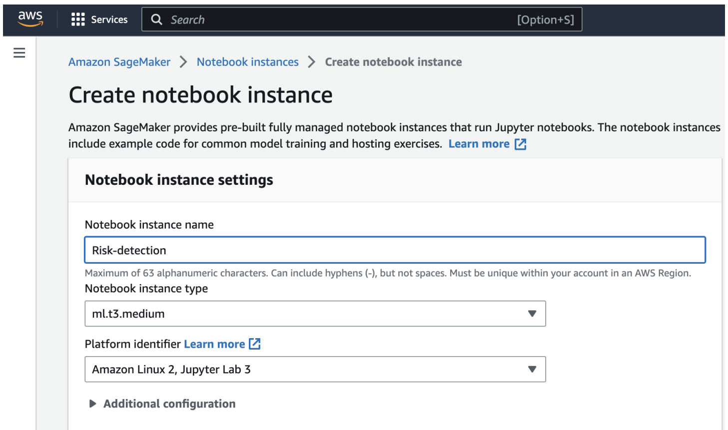


Figure 1: Creation of Jupiter Notebook in Sagemaker

3.2 Dataset Preparation

1. **Dataset:** NSL-KDD dataset used for network intrusion detection.
2. Steps to load the dataset:

- Download the dataset from the provided source: https://www.unb.ca/cic/datasets/nsl.html
- Load the dataset into the notebook using Pandas.

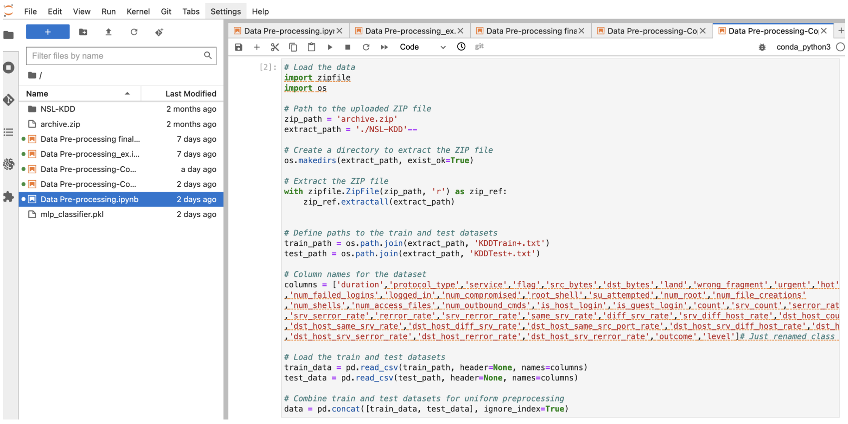


Figure 2: Load the dataset

- Perform necessary data cleaning, feature selection, and encoding.

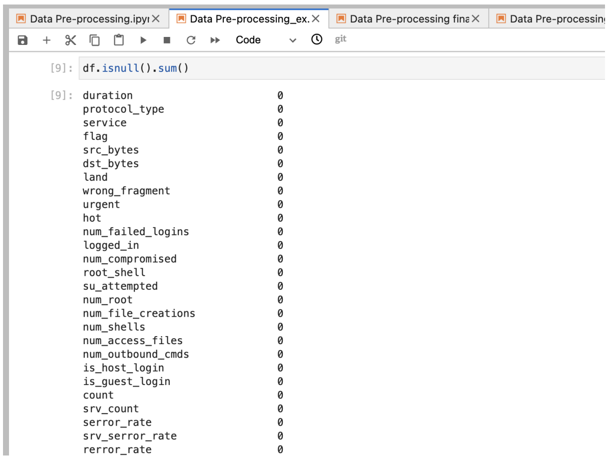
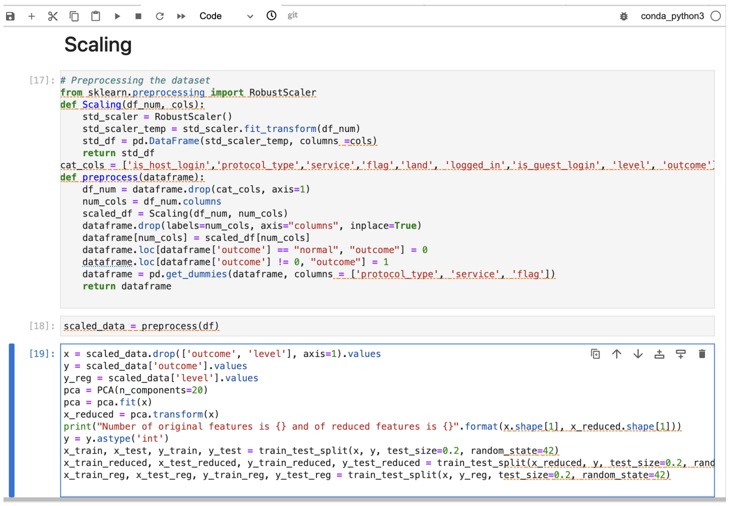


Figure 3: Data Cleaning

3.3 Data Pre-processing

- Apply necessary pre-processing techniques such as scaling, normalization, and handling of null values.



Scaling

```python
[17]: # Preprocessing the dataset
from sklearn.preprocessing import RobustScaler
def Scaling(df_num, cols):
    std_scaler = RobustScaler()
    std_scaler_temp = std_scaler.fit_transform(df_num)
    std_df = pd.DataFrame(std_scaler_temp, columns =cols)
    return std_df
cat_cols = ['is_host_login','protocol_type','service','flag','land', 'logged_in','is_guest_login', 'level', 'outcome'
def preprocess(dataframe):
    df_num = dataframe.drop(cat_cols, axis=1)
    num_cols = df_num.columns
    scaled_df = Scaling(df_num, num_cols)
    dataframe.drop(labels=num_cols, axis="columns", inplace=True)
    dataframe[num_cols] = scaled_df[num_cols]
    dataframe.loc[dataframe['outcome'] == "normal", "outcome"] = 0
    dataframe.loc[dataframe['outcome'] != 0, "outcome"] = 1
    dataframe = pd.get_dummies(dataframe, columns = ['protocol_type', 'service', 'flag'])
    return dataframe

[18]: scaled_data = preprocess(df)

[19]: x = scaled_data.drop(['outcome', 'level'], axis=1).values
y = scaled_data['outcome'].values
y_reg = scaled_data['level'].values
pca = PCA(n_components=20)
pca = pca.fit(x)
x_reduced = pca.transform(x)
print("Number of original features is {} and of reduced features is {}".format(x.shape[1], x_reduced.shape[1]))
y = y.astype('int')
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
x_train_reduced, x_test_reduced, y_train_reduced, y_test_reduced = train_test_split(x_reduced, y, test_size=0.2, rand
x_train_reg, x_test_reg, y_train_reg, y_test_reg = train_test_split(x, y_reg, test_size=0.2, random_state=42)
```

Figure 4: Scaling

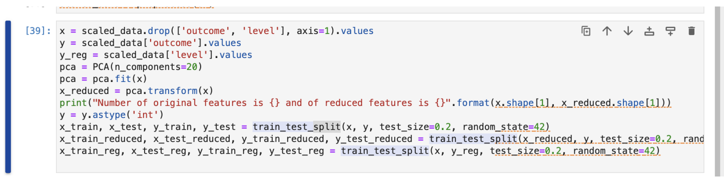
3.4 Model Training and Evaluation

1. Models Implemented:

    - Random Classifier
    - Multilayer Perceptron (MLP)
    - Logistic Regression
    - k-Nearest Neighbors (KNN)

2. Training and Evaluation Process:

    - Split the dataset into training and test sets.



```python
[39]: x = scaled_data.drop(['outcome', 'level'], axis=1).values
y = scaled_data['outcome'].values
y_reg = scaled_data['level'].values
pca = PCA(n_components=20)
pca = pca.fit(x)
x_reduced = pca.transform(x)
print("Number of original features is {} and of reduced features is {}".format(x.shape[1], x_reduced.shape[1]))
y = y.astype('int')
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
x_train_reduced, x_test_reduced, y_train_reduced, y_test_reduced = train_test_split(x_reduced, y, test_size=0.2, rand
x_train_reg, x_test_reg, y_train_reg, y_test_reg = train_test_split(x, y_reg, test_size=0.2, random_state=42)
```

Figure 5: Split Dataset

- Train each model on the training set.
- Evaluate the models using accuracy, precision, recall, and F1-score metrics.
- Compare the performance of models using confusion matrices and validation scores.

3.5 Performance Evaluation

Evaluation of models on the original feature set. The performance metrics are shown in Table 1 and Table 2. Figure 6 displays confusion matrix of KNN model.

Model	Training Accuracy	Test Accuracy	Cross-Validation	F1-Score
Random Classifier	49.87%	50.32%	50.17%	48.07%
Multilayer Perceptron	98.50%	98.50%	97.94%	98.49%
Logistic Regression	89.22%	88.77%	88.99%	88.35%
K-Nearest Neighbors	99.02%	98.87%	98.85%	98.95%

Table 1: Evaluation Results on the Original Feature Set

Model	Training Accuracy	Test Accuracy	Precision	Recall	F1-Score	Cross-Validation
Random Classifier	50.05%	49.68%	46.32%	51.18%	48.55%	50.01%
Multilayer Perceptron	98.15%	98.02%	98.78%	96.95%	97.99%	97.86%
K-Nearest Neighbors	99.02%	98.87%	99.02%	98.56%	98.97%	98.85%
Logistic Regression	89.22%	88.77%	88.47%	87.35%	87.91%	88.99%

Table 2: Evaluation Results of Various Models

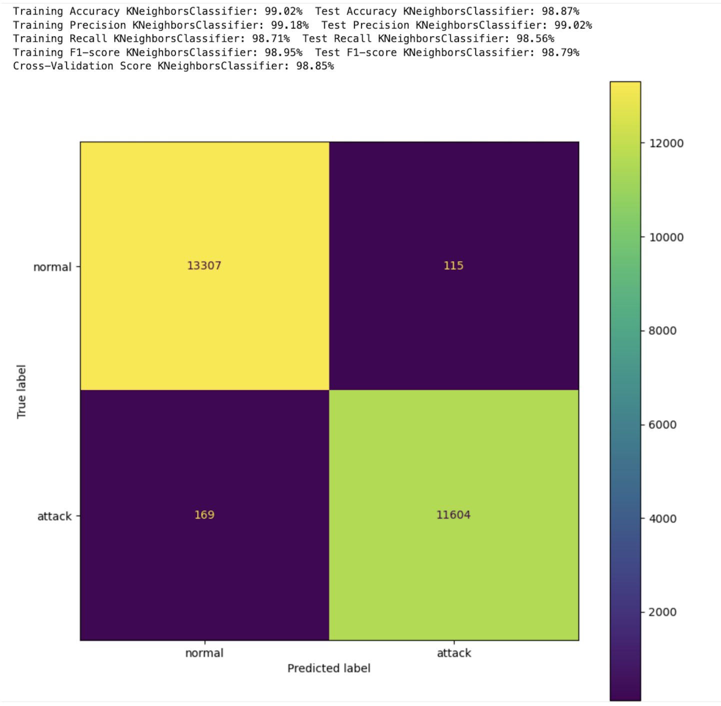


Figure 6: Confusion Matrix of KNN

3.6 Conclusion

This section concludes that KNN and MLP models performed well, demonstrating effectiveness in real world cloud security situations. However, the study emphasized the need for better feature selection and model optimization.