# Configuration Manual

MSc Internship

Cyber Security

Maaz Hasan

Student ID: x18178359

School of Computing

National College of Ireland

Supervisor: Ben Fletcher

**National College of Ireland Project Submission Sheet School of Computing**

| | |
|---|---|
| **Student Name:** | Maaz Hasan |
| **Student ID:** | X18178359 |
| **Programme:** | MSc in Cyber Security |
| **Year:** | 2019 |
| **Module:** | MSc Internship |
| **Supervisor:** | Ben Fletcher |
| **Submission Due Date:** | 12/12/2019 |
| **Project Title:** | Configuration Manual |
| **Word Count:** | 805 |
| **Page Count:** | 12 |

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.
I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

| | |
|---|---|
| **Signature:** | Imhasan |
| **Date:** | 12th December 2019 |

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

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | Q |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | Q |
| **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. | Q |

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

Maaz Hasan

x18178359

## 1 Introduction

This configuration manual provides the details of the proposed work and model used for intrusion detection system. Hybrid IDS is developed using machine learning approaches. It combines Random Forest classification and K-Means clustering. This will use both mis-use detection and anomaly detection for improving performance of the IDS. These algorithms are evaluated for the four categories of attacks based on accuracy, false-alarm-rate, and detection-rate besides other metrics like precision, recall and F1-score. The technical details of individual machine learning methods can be found in literature such as [1], [2], [3], [4], [5], [6] and [7].

## 2 System Configuration

This section provides an overview of the system used for the implementation of this model.

### Hardware Specification

This project is developed using a laptop running Windows 8 operating system. The system specifications are as shown below.

**Operating System: Windows 10.1**
**Intel i7 CPU with 1.60 GHz**
**64 bit operating system**

# 3 Software Specification

This section describes details of the tools and technologies used while developing the project.

| Tool | Version | Description |
|---|---|---|
| Python for Windows[1] | 3.7 | Python language support. |
| Anaconda for Windows[2] | 2019.10 | Data science platform with many IDEs. |
| Jupyter[3] | 3 | Chosen IDE |

Table 1: Tools used in this model

# 4 Working

This section illustrates step by step procedure used for setting up the proposed model and demonstrates its working.

**Software Installation**

Python data science environment is created using the following URL.

https://www.anaconda.com/distribution/#download-section

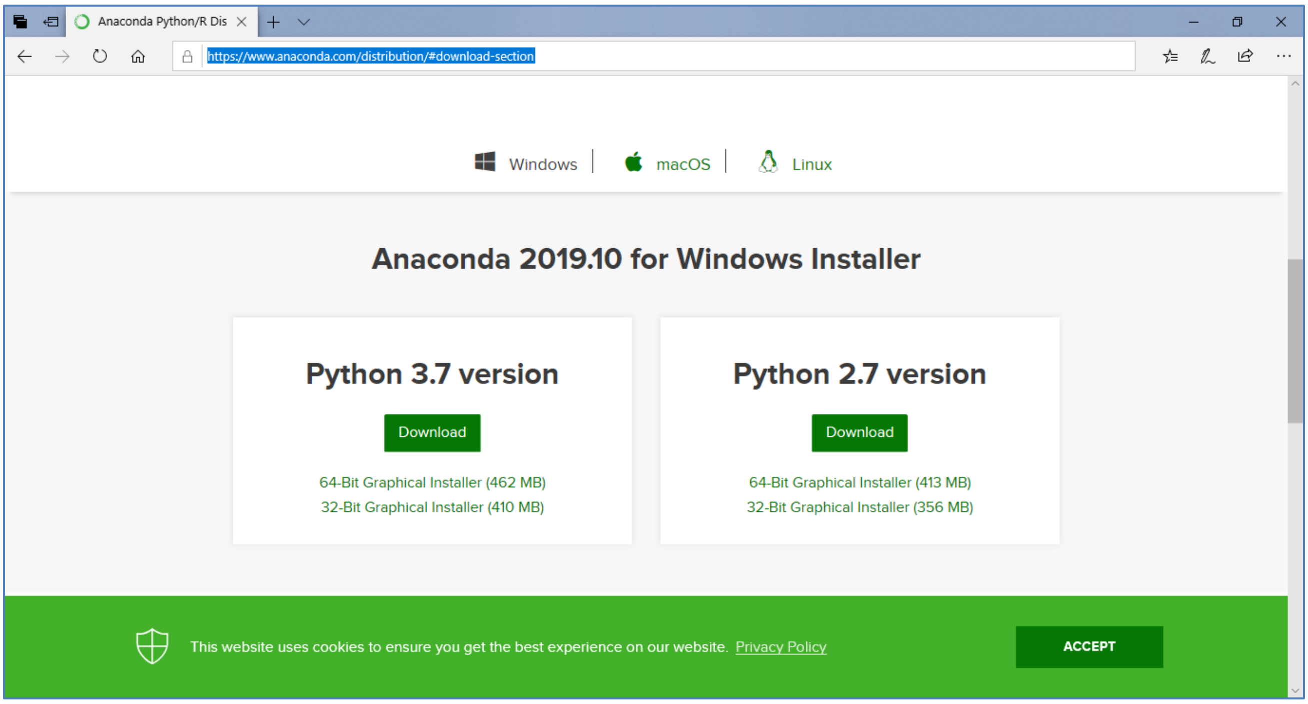


Figure 1: Anaconda installation with Python 3.7

**Implementation**

[1] https://www.anaconda.com/distribution/#download-section
[2] https://www.anaconda.com/distribution/#download-section
[3] https://www.anaconda.com/distribution/#download-section

Jupyter IDE is used for the implementation of IDE. It is part of the Anaconda platform based on Python.

To run the project:

1. Open Jupyter
2. Load the project
3. Load the files to run

### KMeans clustering with Random Forest Classifiers

```
kmeans_prob_col = 'kmeans_rf_prob'
kmeans_pred_col = 'kmeans_rf_pred'

prob_cols.append(kmeans_prob_col)
pred_cols.append(kmeans_pred_col)
# KMeans clustrering
from pyspark.ml.clustering import KMeans

t0 = time()
kmeans_slicer = VectorSlicer(inputCol="indexed_features", outputCol="features",
                             names=list(set(selectFeaturesByAR(ar_dict, 0.1)).intersection(numeric_cols)))

kmeans = KMeans(k=8, initSteps=25, maxIter=100, featuresCol="features", predictionCol="cluster", seed=seed

kmeans_pipeline = Pipeline(stages=[kmeans_slicer, kmeans])

kmeans_model = kmeans_pipeline.fit(scaled_train_df)

kmeans_train_df = kmeans_model.transform(scaled_train_df).cache()
kmeans_cv_df = kmeans_model.transform(scaled_cv_df).cache()
kmeans_test_df = kmeans_model.transform(scaled_test_df).cache()

print(time() - t0)

# Function for describing the contents of the clusters
def getClusterCrosstab(df, clusterCol='cluster'):
    return (df.crosstab(clusterCol, 'labels2')
              .withColumn('count', col('attack') + col('normal'))
              .withColumn(clusterCol + '_labels2', col(clusterCol + '_labels2').cast('int'))
              .sort(col(clusterCol +'_labels2').asc()))

kmeans_crosstab = getClusterCrosstab(kmeans_train_df).cache()
kmeans_crosstab.show(n=30)
# Function for splitting clusters
def splitClusters(crosstab):
    exp = ((col('count') > 25) & (col('attack') > 0) & (col('normal') > 0))

    cluster_rf = (crosstab
```

```
# Function for splitting clusters
def splitClusters(crosstab):
    exp = ((col('count') > 25) & (col('attack') > 0) & (col('normal') > 0))

    cluster_rf = (crosstab
        .filter(exp).rdd
        .map(lambda row: (int(row['cluster_labels2']), [row['count'], row['attack']/row['count']]))
        .collectAsMap())

    cluster_mapping = (crosstab
        .filter(~exp).rdd
        .map(lambda row: (int(row['cluster_labels2']), 1.0 if (row['count'] <= 25) | (row['normal'] == 0) else 0.0))
        .collectAsMap())

    return cluster_rf, cluster_mapping

kmeans_cluster_rf, kmeans_cluster_mapping = splitClusters(kmeans_crosstab)

print(len(kmeans_cluster_rf), len(kmeans_cluster_mapping))
print(kmeans_cluster_mapping)
kmeans_cluster_rf
from pyspark.ml.classification import RandomForestClassifier

# This function returns Random Forest models for provided clusters
def getClusterModels(df, cluster_rf):
    cluster_models = {}

    labels_col = 'labels2_cl_index'
    labels2_indexer.setOutputCol(labels_col)

    rf_slicer = VectorSlicer(inputCol="indexed_features", outputCol="rf_features",
                             names=selectFeaturesByAR(ar_dict, 0.05))

    for cluster in cluster_rf.keys():
        t1 = time()
        rf_classifier = RandomForestClassifier(labelCol=labels_col, featuresCol='rf_features', seed=seed,
```

## Visualization via PCA

```
t0 = time()
pca_slicer = VectorSlicer(inputCol="indexed_features", outputCol="features", names=selectFeaturesByAR(ar_dict, 0.05))

pca = PCA(k=2, inputCol="features", outputCol="pca_features")
pca_pipeline = Pipeline(stages=[pca_slicer, pca])

pca_train_df = pca_pipeline.fit(scaled_train_df).transform(scaled_train_df)
print(time() - t0)
t0 = time()
viz_train_data = np.array(pca_train_df.rdd.map(lambda row: [*row['pca_features'], row['labels2_index'], row['labels5_index']
plt.figure()
plt.scatter(x=viz_train_data[:,0], y=viz_train_data[:,1], c=viz_train_data[:,2], cmap="Set1")
plt.figure()
plt.scatter(x=viz_train_data[:,0], y=viz_train_data[:,1], c=viz_train_data[:,3], cmap="Set1")
plt.show()
print(time() - t0)
```

## Data loading

```
# Creating local SparkContext with 8 threads and SQLContext based on it
sc = pyspark.SparkContext(master='local[8]')
sc.setLogLevel('INFO')
sqlContext = SQLContext(sc)
from pyspark.sql.types import *
from pyspark.sql.functions import udf, split, col
import pyspark.sql.functions as sql

train20_nsl_kdd_dataset_path = "NSL_KDD_Dataset/KDDTrain+_20Percent.txt"
train_nsl_kdd_dataset_path = "NSL_KDD_Dataset/KDDTrain+.txt"
test_nsl_kdd_dataset_path = "NSL_KDD_Dataset/KDDTest+.txt"

col_names = np.array(["duration","protocol_type","service","flag","src_bytes",
    "dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins",
    "logged_in","num_compromised","root_shell","su_attempted","num_root",
    "num_file_creations","num_shells","num_access_files","num_outbound_cmds",
    "is_host_login","is_guest_login","count","srv_count","serror_rate",
    "srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate",
    "diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
    "dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
    "dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
    "dst_host_rerror_rate","dst_host_srv_rerror_rate","labels"])

nominal_inx = [1, 2, 3]
binary_inx = [6, 11, 13, 14, 20, 21]
numeric_inx = list(set(range(41)).difference(nominal_inx).difference(binary_inx))

nominal_cols = col_names[nominal_inx].tolist()
binary_cols = col_names[binary_inx].tolist()
numeric_cols = col_names[numeric_inx].tolist()
# Function to load dataset and divide it into 8 partitions
def load_dataset(path):
    dataset_rdd = sc.textFile(path, 8).map(lambda line: line.split(','))
    dataset_df = (dataset_rdd.toDF(col_names.tolist()).select(
                    col('duration').cast(DoubleType()),
                    col('protocol_type').cast(StringType()),
```

## Experimental Results

```
        normal  attack
normal   13316      12
attack      26   11779

Accuracy = 0.998488
AUC = 0.998449

False Alarm Rate = 0.00090036
Detection Rate = 0.997798
F1 score = 0.99839

             precision    recall  f1-score   support

        0.0       1.00      1.00      1.00     13328
        1.0       1.00      1.00      1.00     11805

avg / total       1.00      1.00      1.00     25133
```

```
        normal  attack
normal    8262    1449
attack     182   12651

Accuracy = 0.927653
AUC = 0.918303

False Alarm Rate = 0.149212
Detection Rate = 0.985818
F1 score = 0.939442

             precision    recall  f1-score   support

        0.0       0.98      0.85      0.91      9711
        1.0       0.90      0.99      0.94     12833

avg / total       0.93      0.93      0.93     22544
```

```
        normal  attack
normal   13195     133
attack       2   11803

Accuracy = 0.994629
AUC = 0.994926

False Alarm Rate = 0.00997899
Detection Rate = 0.999831
F1 score = 0.994314

             precision    recall  f1-score   support

        0.0       1.00      0.99      0.99     13328
        1.0       0.99      1.00      0.99     11805

avg / total       0.99      0.99      0.99     25133
```

```
        normal  attack
normal    8367    1344
attack     830   12003

Accuracy = 0.903566
AUC = 0.898462

False Alarm Rate = 0.1384
Detection Rate = 0.935323
F1 score = 0.91696

             precision    recall  f1-score   support

        0.0       0.91      0.86      0.89      9711
        1.0       0.90      0.94      0.92     12833

avg / total       0.90      0.90      0.90     22544
```

## References


[1] Om, H., & Kundu, A. (2012). *A hybrid system for reducing the false alarm rate of anomaly intrusion detection system. 2012 1st International Conference on Recent Advances in Information Technology (RAIT).* P1-6.

[2] Lin, W.-C., Ke, S.-W., & Tsai, C.-F. (2015). *CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. Knowledge-Based Systems, 78, 13–21.*

[3] Elbasiony, R. M., Sallam, E. A., Eltobely, T. E., & Fahmy, M. M. (2013). *A hybrid network intrusion detection framework based on random forests and weighted k-means. Ain Shams Engineering Journal, 4(4), 753–762.*

[4] Gupta, G. P., & Kulariya, M. (2016). *A Framework for Fast and Efficient Cyber Security Network Intrusion Detection Using Apache Spark. Procedia Computer Science, 93, 824–831.*

[5] Nadiammai, G. V., & Hemalatha, M. (2014). *Effective approach toward Intrusion Detection System using data mining techniques. Egyptian Informatics Journal, 15(1), 37–50.*

[6] Aljawarneh, S., Aldwairi, M., & Yassein, M. B. (2018). Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. Journal of Computational Science, 25, 152–160.

[7] Aslahi-Shahri, B. M., Rahmani, R., Chizari, M., Maralani, A., Eslami, M., Golkar, M. J., & Ebrahimi, A. (2015). *A hybrid method consisting of GA and SVM for intrusion detection system. Neural Computing and Applications, 27(6), 1669–1676.*

[8] Zhang, M.L. and Zhou, Z.H., 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition*, *40*(7), pp.2038-2048.

[9] Dy, J.G. and Brodley, C.E., 2004. Feature selection for unsupervised learning. *Journal of machine learning research*, *5*(Aug), pp.845-889.

[10] Lee, S.L.A., Kouzani, A.Z. and Hu, E.J., 2010. Random forest based lung nodule classification aided by clustering. *Computerized medical imaging and graphics*, *34*(7), pp.535-542.

[11] Visa, S., Ramsay, B., Ralescu, A.L. and Van Der Knaap, E., 2011. Confusion Matrix-based Feature Selection. *MAICS*, *710*, pp.120-127.

[12] Parker, J.R., 2001. Rank and response combination from confusion matrix data. *Information fusion*, *2*(2), pp.113-120.

[13] GeeksforGeeks. (2020). *Random Forest Regression in Python - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/random-forest-regression-in-python/ [Accessed 3 Feb. 2020].

[14] ritchieng.github.io. (2020). *K-nearest Neighbors (KNN) Classification Model*. [online] Available at: https://www.ritchieng.com/machine-learning-k-nearest-neighbors-knn/ [Accessed 3 Feb. 2020]