Detection of Non-Contemporaneous Activity in an Electronic System Using Unsupervised Machine Learning

MSc Research Project
Data Analytics

Chris Miller
Student ID: x20166788

School of Computing
National College of Ireland

Supervisor: Mohammed Hasanuzzaman

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Chris Miller
Student ID:	x20166788
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Mohammed Hasanuzzaman
Submission Due Date:	01/02/2023
Project Title:	Detection of Non-Contemporaneous Activity in an Electronic System Using Unsupervised Machine Learning
Word Count:	1552
Page Count:	21

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:	Chris Miller
Date:	1st February 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):	

Detection of Non-Contemporaneous Activity in an Electronic System Using Unsupervised Machine Learning

Chris Miller
x20166788

1 Introduction

This configuration manual details the pre-requisite requirements needed in order to run the experiments, graphical user interface (GUI), and PowerBI report associated with the research project titled "Detection of Non-Contemporaneous Activity in an Electronic System Using Unsupervised Machine Learning". It also includes the specifications of the machine used to run the experiments, pre-requisite python libraries, code snippets, and references to the appropriate python notebooks. Instructions pertaining to the execution of the python notebooks have also been included to ensure the prospective researcher can execute the experiments without issue and can amend the code for their particular use case. All code, datasets in comma-separated value (CSV) files, database backups, and the PowerBI report are available in GitHub[1] on request due to Turnitin size limitations.

2 System specification

Table 1 details the system specification that was used to run all components of the research project. The specifications listed in this table should be considered as minimum requirements.

Table 1: Materials and equipment versions

Material/Equipment	**Version**
Operating system	Windows Windows 10 Pro 10.0.19044 N/A Build 19044
Processor	Intel(R) Core(TM) i5-3320M
Memory	16GB
Storage	240 GB solid state

3 System software

Table 2 details the required software used to run the research project and must be installed in advance of completing any of the experiments described in the research project report.

[1] https://github.com/ChrisMillerMSDA/researchproject

It is assumed that the prospective researcher has the knowledge and skills to install these software packages.

Table 2: Software and versions

Software	**Version**
Programming language	Python 3.8.12
Development environment	Jupyter notebooks 6.4.3
	Anaconda navigator 2.1.1
Source and repository database	PostgreSQL 14
Database instance IP	127.0.0.1
Database instance port	5432
Database management	pgAdmin 4.30
Visualisations	Microsoft PowerBI desktop Version: 2.109.1021.0 64-bit (September 2022)

4 Data creation

As detailed in the research project report, python will be used to create the synthetic data used by this project. This section therefore describes the structure of the python notebooks used to create these datasets in order to ensure that the datasets can be created successfully for future work.

The product configuration CSV file name is detailed in Table 3. This file can be updated by the prospective researcher if the product name, step name, or step durations need to be changed to reflect their research requirements.

Table 3: Data creation - product configuration file

File name
config.csv

As 3 datasets were created, there are therefore 3 notebooks required to create each of the datasets as per Table 4. The first notebook listed must be executed first as it creates the databases used to store the source table and repository objects, otherwise the 2nd and 3rd notebooks can be executed in any order. Each of the notebooks is self contained with no dependencies on any other notebook. A sample of the packages used is included in Figure 1. When executing each of these notebooks the connection details for the PostgreSQL instance, source and repository databases must be provided as per Figures 2, 3 and 4 respectively. When the notebooks are executed the executor must enter the password to connect to the PostgreSQL instance and databases as per Figure 5. Note that the same password was used for each connection for convenience.

In order to adjust the size of the dataset created for the SIMPLE dataset, change the number of batches from 10001 to the desired number as per Figure 6, for the COMPLEX1 dataset the number of batches and standard deviation ranges can be changed as per Figure 7, likewise as per Figure 8 for the COMPLEX2 dataset.

A number of comma separated value (CSV) files are generated to store the SIMPLE, COMPLEX1, and COMPLEX2 datasets for the subsequent experiments as per Table 5.

Install required packages

```python
# import required packages
import psycopg2
import pandas as pd
```

```python
from getpass import getpass
```

```python
import psycopg2
```

```python
from datetime import datetime
```

```python
from datetime import timedelta
```

```python
import random
```

```python
import numpy as np
```

Figure 1: Data creation package details

```python
#Build the database connection string for the default postgres database
#reference: https://naysan.ca/2020/05/09/pandas-to-postgresql-using-psycopg2-bulk-insert-performance-benchmark/
param_dic = {
    "host"      : "127.0.0.1",
    "port"      : "5432",
    "database"  : "postgres",
    "user"      : "postgres",
    "password"  : password
}
```

Figure 2: PostgreSQL instance connection configuration

```python
#Build the database connection string
#reference: https://naysan.ca/2020/05/09/pandas-to-postgresql-using-psycopg2-bulk-insert-performance-benchmark/
param_dic = {
    "host"      : "127.0.0.1",
    "port"      : "5432",
    "database"  : "sourcedb",
    "user"      : "postgres",
    "password"  : password
}
```

Figure 3: PostgreSQL source database connection configuration

Create a connection to the repository database

```python
#Build the database connection string
#reference: https://naysan.ca/2020/05/09/pandas-to-postgresql-using-psycopg2-bulk-insert-performance-benchmark/
param_dic = {
    "host"      : "127.0.0.1",
    "port"      : "5432",
    "database"  : "repository",
    "user"      : "postgres",
    "password"  : password
}
```

Figure 4: PostgreSQL repository database connection configuration

Get the database password from the user

```
password = getpass( 'Password: ' )
```

```
Password: ········
```

Figure 5: Prompt to enter PostgreSQL password

```
#Outer loop to loop through each set of batches X times
for cnt in range(1,10001):
```

Figure 6: How to change the number of records generated - SIMPLE dataset

```
#initialise variables
num_batches=10000
std_dev_start=1
std_dev_end=5
```

Figure 7: How to change the number of records generated - COMPLEX1 dataset

```
#initialise variables
num_batches=10000
std_dev_start=1
std_dev_end=10
```

Figure 8: How to change the number of records generated - COMPLEX2 dataset

Table 4: Data creation notebooks

Dataset	Notebook name
SIMPLE	01 DB config and data generation.ipynb
COMPLEX1	01b DB config and data generation - using normal distribution.ipynb
COMPLEX2	01c DB config and data generation - using normal distribution - complex2.ipynb

Table 5: Data creation - generated CSV files

Dataset	CSV file
SIMPLE	sourcetable.csv
SIMPLE with anomaly label	sourcetable_with_anomalies.csv
COMPLEX1	sourcetable_nd.csv
COMPLEX1 with anomaly label	pc_nd_copy_sorted_times.csv
COMPLEX2	sourcetable_nd2.csv
COMPLEX2 with anomaly label	pc_nd_copy_sorted_times2.csv

5 Data visualisation

Three separate notebooks were used to visualise the anomalies for each dataset as per Table 6. Each of the notebooks is self contained and takes the CSV files as listed in Table 5 as an input to create box plots, histograms, and scatter plots for each product/step combination. Figures 9, 10, and 11, include such visualisations for the COMPLEX2 dataset, product 10 and step 10 combination. The validity of normal distribution of the data can be confirmed by reviewing a sample of the histograms generated by these notebooks. The plots are stored in the folders listed in Table 7 under the current working directory. Note that the directories are created if they do not exist as per Figure 12. The packages used in each notebook is included in Figure 13. The step duration is also calculated within this set of notebooks and a number of comma separated variable (CSV) files are generated containing the SIMPLE, COMPLEX1, and COMPLEX2 datasets as per Table 8. These CSV files are required by the subsequent experiments.

Table 6: Data visualisation notebooks

Dataset	Notebook name
SIMPLE	02 data visualisation.ipynb
COMPLEX1	02b data visualisation.ipynb
COMPLEX2	02c data visualisation.ipynb

6 IQR and Z-score

Three separate notebooks were used to complete the interquartile range (IQR) and Z-score experiments as detailed in Table 9. Each of the notebooks is self contained and takes the CSV files as listed in Table 8 as an input. The upper and lower IQR limits and percentiles, and the Z-score threshold can be adjusted by updating the code detailed in Figures 14 and 15 respectively. The performance metrics for each method are then calculated by

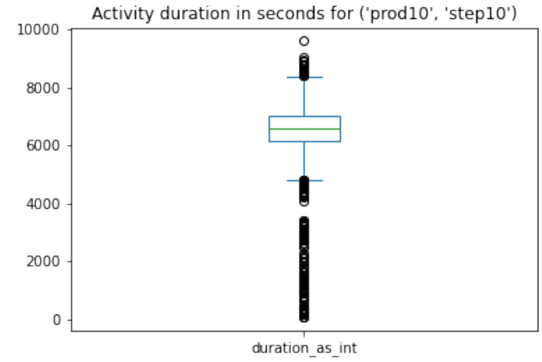


Figure 9: Box plot visualisation for product 10, step 10

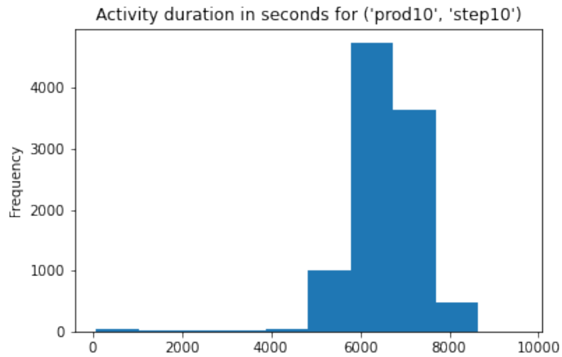


Figure 10: Histogram visualisation for product 10, step 10

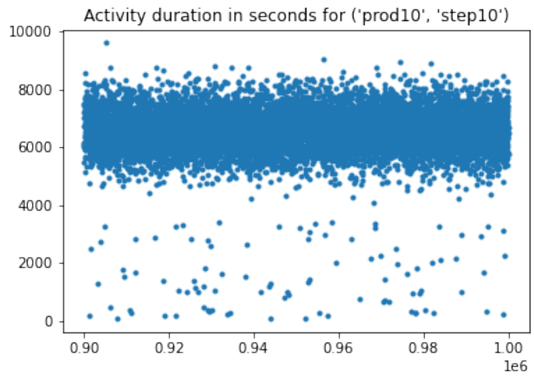


Figure 11: Scatter plot visualisation for product 10, step 10

```
dirname='plots'
if os.path.isdir(dirname):
    print(dirname,' exists')
else:
    os.mkdir(dirname)
```

Figure 12: Create directory for plots

Install required packages

```
# import required packages
import psycopg2
import pandas as pd
```

```
from getpass import getpass
```

```
import psycopg2
```

```
from datetime import datetime
```

```
from datetime import timedelta
```

```
import random
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import os
```

Figure 13: Packages for data visualisation

Table 7: Data visualisation folders

Dataset	Folder
SIMPLE	plots
COMPLEX1	plots_nd
COMPLEX2	plots_nd2

Table 8: Data visualisation - generated CSV files

Dataset	CSV file
SIMPLE	sourcetable_with_duration.csv
COMPLEX1	sourcetable_nd_with_duration.csv
COMPLEX2	sourcetable_nd2_with_duration.csv

comparing the labelled anomaly to the prediction. The IQR performance metrics for the COMPLEX2 dataset are included in Figure 16, similar performance metrics are generated and reviewed for each of the methods used in this research project. The packages used in each notebook are included in Figure 17.

Table 9: IQR and Z-score notebooks

Dataset	Notebook name
SIMPLE	03 IQR method.ipynb
COMPLEX1	03b IQR method.ipynb
COMPLEX2	03c IQR method.ipynb

7 K-means

Three separate notebooks were used to complete the K-means experiments as detailed in Table 10. Each of the notebooks is self contained and takes the CSV files as listed in Table 8 as an input. As detailed in the research project report, the value for k, that is, the number of clusters, is derived intuitively from the number of product/step combinations as detailed in Figures 18 and 19 where the K-means model is built. The n_init parameter can be tuned in order to avoid the issue of local minima as per Figure 20. The packages used in each notebook is included in Figure 21.

8 Isolation Forest

Three separate notebooks were used to complete the Isolation Forest experiments as detailed in Table 11. Each of the notebooks is self contained and takes the CSV files as listed in Table 8 as an input.

As detailed in the research project report, the contamination level, that is, the number of potential anomalies, and the number of estimators, n_estimators, can be tuned as detailed in Figure 22. The max_features parameter will also need to be adjusted if the shape of the dataset is changed. The packages used in each notebook is included in Figure 23.

```python
# Find Q1 and Q3
Q1,Q3 = np.percentile(data['duration_as_int'] , [25,75])

# Find IQR, upper limit, lower limit
IQR = Q3 - Q1
ul = Q3+1.5*IQR
ll = Q1-1.5*IQR

# Find IQR, upper limit, lower limit
IQR = Q3 - Q1
```

Figure 14: IQR method - how to change upper and lower limits

```python
# Get the Z score
z = np.abs(stats.zscore(tempdf))

# Find outliers
outliers_temp = tempdf[(z>3).all(axis=1)]
outliers = outliers.append(outliers_temp)

# create a copy of the current data subset
data2=data.copy(deep=True)
data2.reset_index(inplace=True)
z.reset_index(inplace=True, drop=True)
data2['predicted'] = (z>3).astype(int)
```

Figure 15: Z-score method - how to change the Z-score threshold

```python
# Evaluate the IQR method
TP=len(sourcetable_anomalies.loc[(sourcetable_anomalies['anomaly']=='Y') & (sourcetable_anomalies['predicted']==1)])
TN=len(sourcetable_anomalies.loc[(sourcetable_anomalies['anomaly']=='N') & (sourcetable_anomalies['predicted']==0)])
FP=len(sourcetable_anomalies.loc[(sourcetable_anomalies['anomaly']=='N') & (sourcetable_anomalies['predicted']==1)])
FN=len(sourcetable_anomalies.loc[(sourcetable_anomalies['anomaly']=='Y') & (sourcetable_anomalies['predicted']==0)])
Accuracy=(TP+TN)/(TP+FP+TN+FN)
Precision=(TP)/(TP+FP)
Recall=(TP)/(TP+FN)
F1Score=(2*Precision*Recall)/(Precision*Recall)
TPR=(TP)/(TP+FN)
FPR=(FP)/(FP+TN)

print('TP: ',TP)
print('FP: ',FP)
print('TN: ',TN)
print('FN: ',FN)
print('Accuracy: ',Accuracy)
print('Precision: ',Precision)
print('Recall: ',Recall)
print('TPR: ',TPR)
print('FPR: ',FPR)
```

```
TP:  9902
FP:  3027
TN:  987071
FN:  0
Accuracy:  0.996973
Precision:  0.7658751643591926
Recall:  1.0
TPR:  1.0
FPR:  0.0030572731184185806
```

Figure 16: IQR metrics

Install required packages

```
# import required packages
import psycopg2
import pandas as pd
```

```
from getpass import getpass
```

```
import psycopg2
```

```
from datetime import datetime
```

```
from datetime import timedelta
```

```
import random
```

```
import numpy as np
```

```
import scipy.stats as stats
```

```
import time
```

Figure 17: IQR and Z-score packages

how many unique product/step combinations are there?

```
count_unique = sourcetable['prod_step'].nunique()                   # Apply unique function
print(count_unique)
```

```
100
```

Figure 18: K-means method - setting of k value

Build the k-means model

```
# https://medium.com/analytics-vidhya/clustering-on-mixed-data-types-in-python-7c22b3898086
# https://towardsdev.com/outlier-detection-using-k-means-clustering-in-python-214188fc90e8
time_1 = time.time()
kmeans = KMeans(n_clusters=count_unique, random_state=1234)
clusters = kmeans.fit(sourcetable_norm)
time_2 = time.time()
time_interval = time_2 - time_1
print(time_interval)
```

Figure 19: K-means method - building the model using k value

```
# Kmeans with different hyperparameters
# https://medium.com/analytics-vidhya/clustering-on-mixed-data-types-in-python-7c22b3898086
# https://towardsdev.com/outlier-detection-using-k-means-clustering-in-python-214188fc90e8
time_1 = time.time()
kmeans2 = KMeans(n_clusters=count_unique, random_state=1234, n_init=100)
clusters2 = kmeans2.fit(sourcetable_norm)
time_2 = time.time()
time_interval = time_2 - time_1
print(time_interval)
```

Figure 20: K-means method - avoiding the issue of local minima

Install required packages ¶

```
# import required packages
import psycopg2
import pandas as pd
```

```
from getpass import getpass
```

```
import psycopg2
```

```
from datetime import datetime
```

```
from datetime import timedelta
```

```
import random
```

```
import numpy as np
```

```
from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as plt
```

```
import scipy.stats as stats
```

```
import time
```

Figure 21: K-means packages

Table 10: K-means notebooks

Dataset	**Notebook name**
SIMPLE	04 k means clustering method.ipynb
COMPLEX1	04b k means clustering method.ipynb
COMPLEX2	04c k means clustering method.ipynb

Table 11: Isolation Forest notebooks

Dataset	**Notebook name**
SIMPLE	05 Isolation Forest.ipynb
COMPLEX1	05a Isolation Forest.ipynb
COMPLEX2	05b Isolation Forest.ipynb

9 Restricted Boltzmann Machines

Three separate notebooks were used to complete the Restricted Boltzmann Machines experiments as detailed in Table 12. Each of the notebooks is self contained and takes the CSV files as listed in Table 8 as an input.

As detailed in the research project report, the learning_rate and batch_size, can be tuned as detailed in Figures 24 and 25 respectively. The packages used in each notebook is included in Figure 26.

Table 12: Restricted Boltzmann Machines

Dataset	**Notebook name**
SIMPLE	06 RBM.ipynb
COMPLEX1	06b RBM.ipynb
COMPLEX2	06c RBM.ipynb

10 Adaptive Resonance Theory

Three separate notebooks were used to complete the Adaptive Resonance Theory experiments as detailed in Table 13. Each of the notebooks is self contained and takes the CSV files as listed in Table 8 as an input.

As detailed in the research project report, the vigilance parameter, can be tuned as detailed in Figure 27. The packages used in each notebook is included in Figure 28.

11 Graphical User Interface

One notebook was used to design and launch the graphical user interface (GUI) as per Table 14. This notebook connects to the configured source and repository databases and uses RBM to determine if there are any potential anomalies. The default parameters can be reviewed and amended as per Figure 29. The packages used in this notebook is included in Figure 30.

```
for x in range(10,110,10):
    time_1 = time.time()

    IF = IsolationForest(n_estimators=x, contamination=0.001, max_features=10, random_state=1234)
    IF_predictions = IF.fit_predict(sourcetable_norm)

    time_2 = time.time()
    time_interval = time_2 - time_1

    outlier_index = where(IF_predictions==-1)
    outlier_index_array = np.asarray(outlier_index)
    num_outliers=outlier_index_array.shape

    print('n_estimators=',x,' time=', time_interval, ' num_outliers=', num_outliers)
```

```
n_estimators= 10  time= 30.301640033721924  num_outliers= (1, 0)
n_estimators= 20  time= 57.817610025405884  num_outliers= (1, 8)
n_estimators= 30  time= 86.42144083976746  num_outliers= (1, 8)
n_estimators= 40  time= 120.02882409095764  num_outliers= (1, 953)
n_estimators= 50  time= 146.59977221488953  num_outliers= (1, 953)
n_estimators= 60  time= 162.93362975120544  num_outliers= (1, 953)
n_estimators= 70  time= 178.13719129562378  num_outliers= (1, 953)
n_estimators= 80  time= 201.99508118629456  num_outliers= (1, 953)
n_estimators= 90  time= 226.88621616363525  num_outliers= (1, 953)
n_estimators= 100  time= 251.91715908050537  num_outliers= (1, 953)
```

Figure 22: Isolation Forest - contamination and n_estimators

Table 13: Adaptive Resonance Theory

Dataset	Notebook name
SIMPLE	07 ART2.ipynb
COMPLEX1	07b ART2.ipynb
COMPLEX2	07c ART2.ipynb

12 PowerBI Visualisation

One PowerBI visualisation report was developed to visualise the anomalies as per Table 15. This PowerBI report connects to the postgreSQL repository database to display the source records, with a potential anomaly indicator, and the activity duration presented in table format, and the count of activity durations visualised in a stacked column chart. The PowerBI report will connect to the project database by default to acquire the required data, however the connection details can be changed by initially entering the model view mode as per Figure 31. Edit the query as per Figure 32. Click on data source settings as per Figure 33. Select the change source button on the data source settings screen as per Figure 34. The source server and database can be changed as per Figure 35.

Filters can be used to select a specific product and step combination to facilitate visual anomaly detection. In order to view the filters, click on the report icon as per Figure 36. Select view, then filters, from the menu as per Figure 37. The filters will be

Table 14: Graphical User Interface - notebook

Notebook name
08 - PySimpleGUI.ipynb

Install required packages

```
# import required packages
import psycopg2
import pandas as pd
```

```
from getpass import getpass
```

```
import psycopg2
```

```
from datetime import datetime
```

```
from datetime import timedelta
```

```
import random
```

```
import numpy as np
```

```
from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as plt
```

```
import scipy.stats as stats
```

```
from sklearn.ensemble import IsolationForest
```

```
from numpy import where
```

```
import time
```

```
from sklearn import preprocessing
```

Figure 23: Isolation Forest - packages

Build the RBM model - looping through different learning rates.

```
for lr in [.0001, .001, .01, .1, 1]:
    print('lr: ',lr)

    # https://www.todaysoftmag.com/article/747/restricted-boltzmann-machines
    # https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.BernoulliRBM.html
    # Fit the model to the data X.
    # Compute the hidden layer activation probabilities, P(h=1|v=X).
    # returns Latent representations of the data.
    time_1 = time.time()
    #rbm = RBM(n_components=900, learning_rate=0.05, batch_size=100, n_iter=50)
    rbm = BernoulliRBM(n_components=1, random_state=1234, learning_rate=lr)
    model_rbm=rbm.fit_transform(sourcetable_norm)
    time_2 = time.time()
    time_interval = time_2 - time_1
    print('time: ',time_interval)
```

Figure 24: Restricted Boltzmann Machines - learning_rate

Run the best performing model - with different batch sizes

```
lr=0.0001
for bs in [50, 100, 200]:
    print('lr: ',lr)
    print('bs: ',bs)

    # https://www.todaysoftmag.com/article/747/restricted-boltzmann-machines
    # https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.BernoulliRBM.html
    # Fit the model to the data X.
    # Compute the hidden layer activation probabilities, P(h=1|v=X).
    # returns Latent representations of the data.
    time_1 = time.time()
    #rbm = RBM(n_components=900, learning_rate=0.05, batch_size=100, n_iter=50)
    rbm = BernoulliRBM(n_components=1, random_state=1234, learning_rate=lr, batch_size=bs)
    model_rbm=rbm.fit_transform(sourcetable_norm)
    time_2 = time.time()
    time_interval = time_2 - time_1
    print('time: ',time_interval)
```

Figure 25: Restricted Boltzmann Machines - batch_size

Install required packages

```
# import required packages
import psycopg2
import pandas as pd
```

```
from getpass import getpass
```

```
import psycopg2
```

```
from datetime import datetime
```

```
from datetime import timedelta
```

```
import random
```

```
import numpy as np
```

```
from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as plt
```

```
import scipy.stats as stats
```

```
import time
```

```
from sklearn.neural_network import BernoulliRBM
```

Figure 26: Restricted Boltzmann Machines - packages

```python
# ART training

time_1 = time.time()

w = ART2(sourcetable_norm2, rho=0.9987)

time_2 = time.time()
time_interval = time_2 - time_1
```

Figure 27: Adaptive Resonance Theory - vigilance parameter

```python
import numpy as np
import warnings
import math
import matplotlib.pyplot as plt
```

```python
import scipy.stats as stats
```

```python
import time
```

Figure 28: Adaptive Resonance Theory - packages

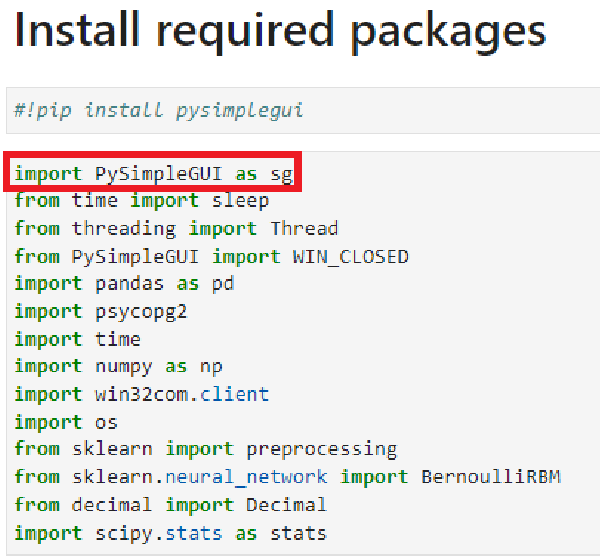
GUI class and launch

read_table 4/3

```python
# https://stackoverflow.com/questions/69925499/pysimplegui-table-not-automatically-refreshing
class GUI():

    def __init__(self):
        self.data=read_table_blank()
        self.layout = [
            [   [sg.Text("Enter source DB IP address:",    size=(25, 1), font="Lucida",justification='right'),sg.Input(key='IP',default_text='127.0.0.1')],
                [sg.Text("Enter source DB port:",    size=(25, 1), font="Lucida",justification='right'),sg.Input(key='PORT',default_text='5432')],
                [sg.Text("Enter source DB name:",    size=(25, 1), font="Lucida",justification='right'),sg.Input(key='DB',default_text='sourcedb')],
                [sg.Text("Enter source DB username:",    size=(25, 1), font="Lucida",justification='right'),sg.Input(key='DBUSER',default_text='postgres')],
                [sg.Text('Enter source DB password', size=(25, 1),font="Lucida",justification='right'), sg.InputText('', key='PASSWORD', password_char='*')],
                [sg.Text("Enter SQL:",    size=(25, 1), font="Lucida",justification='right'),sg.Multiline(key='SQL',default_text='select * from public.sourcetable_nd2 limit 100000;
                [sg.Text("Enter repository DB IP address:",    size=(25, 1), font="Lucida",justification='right'),sg.Input(key='REP_IP',default_text='127.0.0.1')],
                [sg.Text("Enter repository DB port:",    size=(25, 1), font="Lucida",justification='right'),sg.Input(key='REP_PORT',default_text='5432')],
                [sg.Text("Enter repository DB name:",    size=(25, 1), font="Lucida",justification='right'),sg.Input(key='REP_DB',default_text='repository')],
                [sg.Text("Enter repository DB username:",    size=(25, 1), font="Lucida",justification='right'),sg.Input(key='REP_DBUSER',default_text='postgres')],
                [sg.Text('Enter repository DB password', size=(25, 1),font="Lucida",justification='right'), sg.InputText('', key='REP_PASSWORD', password_char='*')],
                [sg.Text("Notification email address:",size=(25, 1), font='Lucida',justification='right'),sg.Input(key='EMAIL',default_text='x20166788@student.ncirl.ie')],
                [sg.Text("Enter RBM learning rate:",size=(25, 1), font='Lucida',justification='right'),sg.Input(key='RBMLR',default_text='0.0001')],
                [sg.Text("Enter RBM batch_size:",size=(25, 1), font='Lucida',justification='right'),sg.Input(key='RBMBS',default_text='800')],
```

Figure 29: Graphical User Interface - default parameters



Install required packages

```python
#!pip install pysimplegui

import PySimpleGUI as sg
from time import sleep
from threading import Thread
from PySimpleGUI import WIN_CLOSED
import pandas as pd
import psycopg2
import time
import numpy as np
import win32com.client
import os
from sklearn import preprocessing
from sklearn.neural_network import BernoulliRBM
from decimal import Decimal
import scipy.stats as stats
```

Figure 30: Graphical User Interface - packages

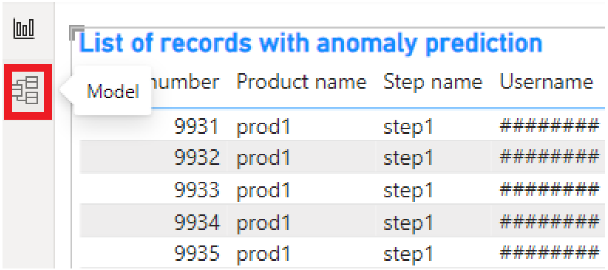


Figure 31: PowerBI - view model

displayed in the report where a specific product/step can be selected as per Figure 38.

Table 15: PowerBI visualisation

PowerBI file name
ADS - Anomaly Detection System PowerBI report.pbix

13 PostgreSQL Database Restore

Two PostgreSQL database backups have been submitted with the project to allow the prospective researcher to restore the repository and sourcedb databases. This will allow the GUI and PowerBI report to be tested without having to re-run the data generation notebooks.

The database creation scripts are detailed in Table 16, these scripts can be run from pgadmin or by using psql to create the destination databases prior to database restore.

The database backups files are detailed in Table 17, these backups can be restored from pgadmin or by using psql, sample restore scripts have been provided as per Table 18.

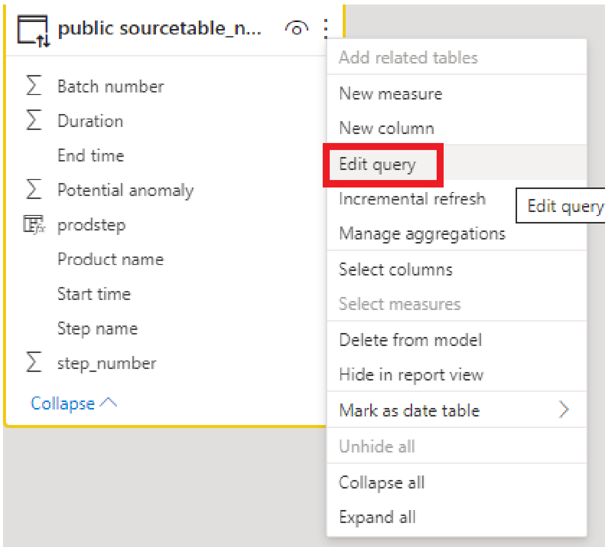


Figure 32: PowerBI - edit query

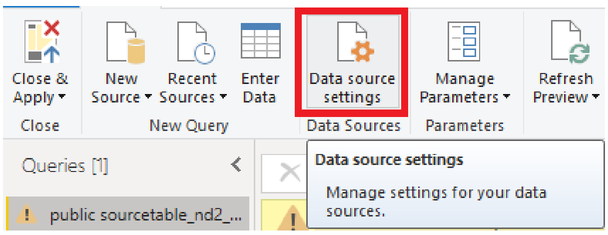


Figure 33: PowerBI - datasource settings icon

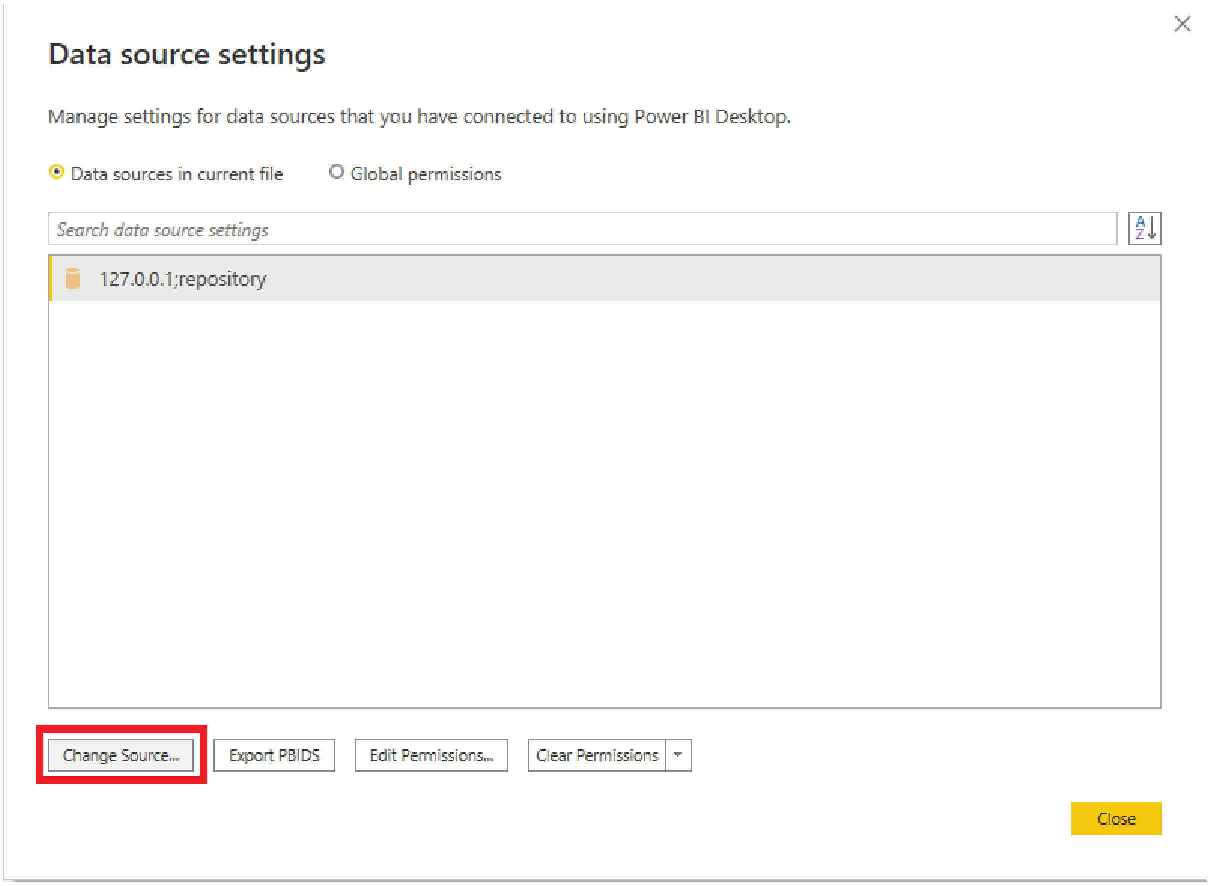


Figure 34: PowerBI - datasource settings

Figure 35: PowerBI - change data source settings

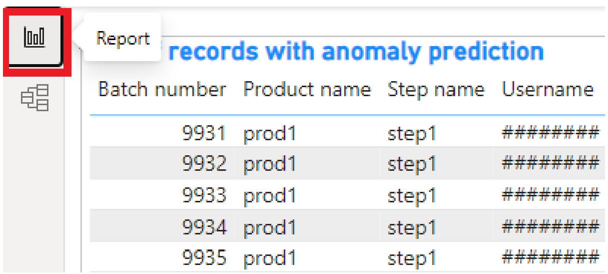


Figure 36: PowerBI - report icon

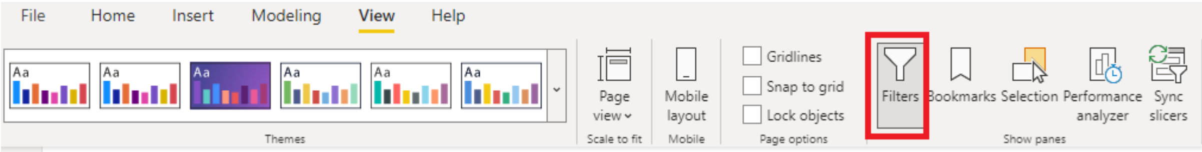


Figure 37: PowerBI - view filters

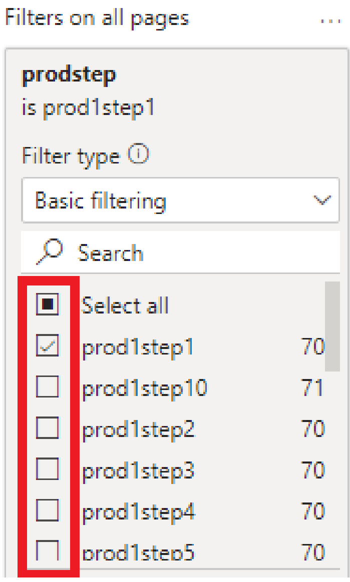


Figure 38: PowerBI - select a filter

References

There are no reference in this configuration manual.

Table 16: Database creation scripts

Database	Script name
repository	repository.sql
sourcedb	sourcedb.sql

Table 17: Database backup files

Database	Script name
repository	repository.tar
sourcedb	sourcedb.tar

Table 18: Database restore scripts

Database	Script name
repository	restore-repository.cmd
sourcedb	restore-sourcedb.cmd