Configuration Manual

MSc Research Project
Data Analytics

Camila da Silva Weber
Student ID: x20166371

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Camila da Silva Weber
Student ID:	x20166371
Programme:	Master of Data Analytics **Year:** 2023
Module:	MSc Research Project
Lecturer:	Vladimir Milosavljevic
Submission Due Date:	14/08/2023
Project Title:	Detecting Fraudulent Transactions in Ethereum Blockchain via Machine Learning Classification
Word Count:	783 **Page Count:** 9

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: ……

Date: ……

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

Camila da Silva Weber
Student ID: x20166371

1 Introduction

The configuration manual is essential to clarify the information referring to the project in relation to the software used, the hardware and other specifications regarding the programs applied in the project. Since in section 2 and 3 the configuration systems that were used in the project will be specified and in section 4 the software used in the project will be presented, in addition to the pre-processing of the data, presenting the libraries using Python, in addition to the analysis concerning the correlation between the features and the results using Machine learning models for classification.

2 System Configuration

The hardware and software and its configuration system that were applied in the project are described below.

Processor	Intel® Core™ i5
Operating System	Windows 10
RAM	8GB
CPU	NVIDIA RTX 2060

3 Software Specification

The other tools that were in order to carry out the project, separated as programming language, software and browsers.

Pogramming Language	Python
Softwares	Excel, Anaconda, Jupyter Notebook
Browsers	Microsoft Edge and Google Chrome

4 Environment Setup

4.1 Initiation of Jupyter Notebook on Anaconda

The configuration of the project was initiated by the application of Anaconda Navigator since it is an important software where it is possible to download applications and other software to support the research. Jupyter Notebook was implemented in the project and application of the codes using Python since it contains important updates and it facilitates the analysis of the project.

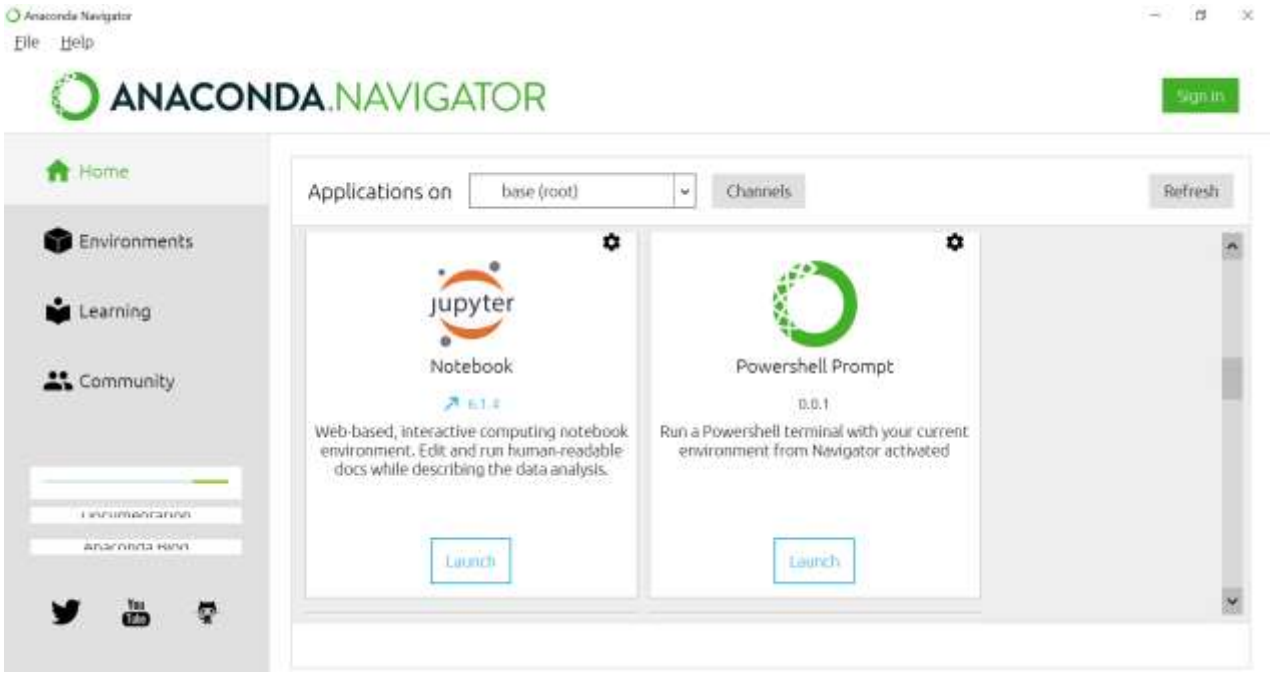

Figure 1: Anaconda Navigator interface

Fig. 2 present the Jupyter Notebook home page, where it is possible to organize and create a file, the format used was ipynb to start the codes in Python.

Figure 2: Jupyter Notebook interface

4.2 Data Preparation and Importing Libraries

The libraries applied in the analysis of the project were added according to the progress of the research and the necessity regarding the use of Machine models implemented.

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn import linear_model
from sklearn.metrics import classification_report
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.svm import SVC

from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

```
from sklearn.ensemble import RandomForestClassifier
```

Figure 3: Libraries Python

4.3 Importing Dataset

Importing data referring to the project in csv format. to Python and thus reading the file to start the ETL (extract, transform and load).

```
df_transaction = pd.read_csv('C:/Users/35383/Desktop/project new/transaction_dataset.csv')
```

```
df_transaction.columns = [i.strip().replace(' ', '_') for i in df_transaction.columns.to_list()]
```

```
df_transaction.head(10)
```

	Unnamed:_0	Index	Address	FLAG	Avg_min_between_sent_tnx	Avg_min_between_received_tnx	Time_Diff_between_firs
0	0	1	0x00009277775ac7d0d59eaad8fee3d10ac6c805e8	0	844.26	1093.71	
1	1	2	0x0002b44ddb1476db43c868bd494422ee4c136fed	0	12709.07	2958.44	
2	2	3	0x0002bda54cb772d040f779e88eb453cac0daa244	0	246194.54	2434.02	
3	3	4	0x00038e6ba2fd5c09aedb96697c8d7b8fa6632e5e	0	10219.60	15785.09	
4	4	5	0x00062d1dd1afb6fb02540ddad9cdebfe568e0d89	0	36.61	10707.77	
5	5	6	0x000895ad78f4403ecd9468900e68d6ee506136fd	0	9900.12	375.48	
6	6	7	0x000d63fc5df52b0204374c2f5a3249779805d5d1	0	69.46	629.44	
7	7	8	0x000e001ab444fa8d6dc4a402f8d7cfc88fe8c64d	0	1497.39	176.84	
8	8	9	0x0012cb699c836049a4bbeaac2d8c4d47c688e0e4	0	0.00	0.00	
9	9	10	0x0012f247c9f980eea0a9ad06893bfd95c3145794	0	2570.59	3336.01	

10 rows × 51 columns

Figure 4: Importing Dataset

4.4 Dataset Pre-processing

Data pre-processing takes place in the data cleaning phase, where not relevant information to the project will be discarded, leaving only the data that will be needed for the analysis.

```
# Deleting other nominal qualitative variables

categorias = df_transaction.select_dtypes('O').columns.astype('category')
df_transaction[categorias]
```

	ERC20_most_sent_token_type	ERC20_most_rec_token_type
0	Cofoundit	Numeraire
1	Livepeer Token	Livepeer Token
2	None	XENON
3	Raiden	XENON
4	StatusNetwork	EOS
...	...	...
9836		GSENetwork
9837		Blockwell say NOTSAFU
9838		Free BOB Tokens - BobsRepair.com
9839	NaN	NaN
9840		INS Promo1

```
# Deleting features with 0 variance as those features will not help in the performance of the model

df_transaction.drop(df_transaction.var()[no_var].index, axis=1, inplace=True)
df_transaction.var()
```

```
FLAG                                                    1.724110e-01
Avg_min_between_sent_tnx                                4.616718e+08
Avg_min_between_received_tnx                            5.327656e+08
Time_Diff_between_first_and_last_(Mins)                 1.042889e+11
Sent_tnx                                                5.733918e+05
Received_Tnx                                            8.851734e+05
Number_of_Created_Contracts                             2.000685e+04
Unique_Received_From_Addresses                          8.917457e+04
Unique_Sent_To_Addresses                                6.960121e+04
min_value_received                                      1.062298e+05
max_value_received                                      1.692294e+08
avg_val_received                                        8.323238e+06
min_val_sent                                            1.921264e+04
max_val_sent                                            4.394646e+07
avg_val_sent                                            5.715935e+04
min_value_sent_to_contract                              5.080371e-08
max_val_sent_to_contract                                2.660652e-07
avg_value_sent_to_contract                              1.046096e-07
total_transactions_(including_tnx_to_create_contract    1.828997e+06
total_Ether_sent                                        1.283952e+11
total_ether_received                                    1.326451e+11
total_ether_sent_contracts                              2.660625e-07
total_ether_balance                                     5.877009e+10
Total_ERC20_tnxs                                        1.835047e+05
ERC20_total_Ether_received                              1.017063e+20
ERC20_total_ether_sent                                  1.275951e+18
ERC20_total_Ether_sent_contract                         3.439675e+07
ERC20_uniq_sent_addr                                    1.014723e+04
```

Figure 5: Dataset Pre-Processing

In addition to cleaning the data, it is necessary to perform a correlation analysis of the main features and delete those with the highest correlation to leave the data with only the information necessary to have the best result in the analysis.

```
# Dropping the highly correlated features


drop = ['total transactions (including tnx to create contract',
        'total ether sent contracts',
        'max val sent to contract',
        ' ERC20 avg val rec',
        ' ERC20 avg val rec',
        ' ERC20 max val rec',
        ' ERC20 min val rec',
        ' ERC20 uniq rec contract addr',
        'max val sent',
        ' ERC20 avg val sent',
        ' ERC20 min val sent',
        ' ERC20 max val sent',
        ' Total ERC20 tnxs',
        'avg value sent to contract',
        'Unique Sent To Addresses',
        'Unique Received From Addresses',
        'total ether received',
        ' ERC20 uniq sent token name',
        'min value received',
        'min val sent',
        ' ERC20 uniq rec addr' ]

drop_new = []

for index, text in enumerate(drop):
    drop_new.append(text.strip().replace(" ", "_"))

df_transaction.drop(drop_new, axis=1, inplace=True)
```

Figure 6: Correlated features

4.5 Machine Learning Algorithms

4.5.1 Logistic Regression

Logistic regression was one of the Machine Learning techniques applied, since it is a classification model and it is a important technique to apply in the analysis of data since it will be focused on detection of frauds, and the prediction usually has a finite number of results, such as yes or no.

```
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.metrics import classification_report
```

```
dataset = pd.read_csv('./clean_transaction.csv')
dataset.head()
```

	FLAG	Avg_min_between_sent_tnx	Avg_min_between_received_tnx	Time_Diff_between_first_and_last_(Mins)	Sent_tnx	Received_Tnx	Number_of_Created_Cont
0	0	844.26	1093.71	704785.63	721	89	
1	0	12709.07	2958.44	1218216.73	94	8	
2	0	246194.54	2434.02	516729.30	2	10	
3	0	10219.60	15785.09	397555.90	25	9	
4	0	36.61	10707.77	382472.42	4598	20	

Figure 7: Logistic Regression libraries

In this phase the data was divided to train and test, the results were improved using 65% of the total data, thus allowing the real performance to be verified.

```
# Better results with 0.65 train
train_size=0.65
```

```
labels_train = np.array(labels[:int(len(labels)*train_size)])
labels_train.reshape(-1, 1)
labels_train
```

```
array([1, 1, 0, ..., 0, 0, 0], dtype=int64)
```

```
labels_test = np.array(labels[int(len(labels)*train_size):])
labels_test.reshape(-1, 1)
labels_train
```

```
array([1, 1, 0, ..., 0, 0, 0], dtype=int64)
```

```
features_train = dataset[:int(len(dataset)*train_size)]
features_train
```

	Avg_min_between_sent_tnx	Avg_min_between_received_tnx	Time_Diff_between_first_and_last_(Mins)	Sent_tnx	Received_Tnx	Number_of_Created_Contract
0	1641.74	2103.12	327679.35	10	148	
1	2811.51	837.98	9812.92	2	5	
2	157.32	0.00	314.65	2	1	
3	20.17	3.92	68.37	3	2	
4	4.38	24303.06	243074.38	10	10	
...	...	...	...	...	...	
6391	1060.09	115459.87	371911.35	133	2	

Figure 8: Logistic Regression Train and Test

There was important results using the Logistic regression model with an accuracy of 83% but it is still not the best model for the project.

```
from sklearn import metrics
```

```
y_predict = lgr.predict(features_test)
```

```
cm = metrics.confusion_matrix(labels_test, y_predict)
```

```
cm
```

```
array([[2643,   22],
       [ 559,  221]], dtype=int64)
```

```
print (classification_report(labels_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.83      0.99      0.90      2665
           1       0.91      0.28      0.43       780

    accuracy                           0.83      3445
   macro avg       0.87      0.64      0.67      3445
weighted avg       0.84      0.83      0.79      3445
```

Figure 9: Logistic Regression Results

4.5.2 Support Vector Machine

The second model applied in the project was the Support Vector Machine it is a supervised machine learning algorithm that can be used for classification or regression. Its major focus is on training and classifying a dataset.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.svm import SVC

from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

```
dados = pd.read_csv("./clean_transaction.csv")

dados.head()
```

	FLAG	Avg_min_between_sent_tnx	Avg_min_between_received_tnx	Time_Diff_between_first_and_last_(Mins)	Sent_tnx	Received_Tnx	Number_of_Created_Cont
0	0	844.26	1093.71	704785.63	721	89	
1	0	12709.07	2958.44	1218216.73	94	8	
2	0	246194.54	2434.02	516729.30	2	10	
3	0	10219.60	15785.09	397555.90	25	9	
4	0	36.61	10707.77	382472.42	4598	20	

Figure 10: SVM libraries

The data was divided to train and test, the results were improved using 70% of the total data, thus allowing the real performance to be verified.

```
# Better results with 70% test

test_size=0.7
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=test_size, random_state=42)
```

```
X_train.head()
```

	Avg_min_between_sent_tnx	Avg_min_between_received_tnx	Time_Diff_between_first_and_last_(Mins)	Sent_tnx	Received_Tnx	Number_of_Created_Contract
3808	105.53	1468.66	1054850.45	560	678	
1138	0.00	0.00	196.07	1	1	
2257	125.71	2436.45	496574.42	132	197	
7588	12.30	4428.66	754914.00	166	170	
1787	164.08	0.82	329.78	2	2	

Figure 11: SVM Train and Test

The SVM model did not present the best results, even with an accuracy of 78% it was over fitting and it did not provide the best insights so it should be a good model for this project.

```
print (classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      1.00      0.88      5367
           1       0.00      0.00      0.00      1522

    accuracy                           0.78      6889
   macro avg       0.39      0.50      0.44      6889
weighted avg       0.61      0.78      0.68      6889
```

Figure 12: SVM Results

4.5.3 Random Forest

The third model applied was the Random Forest, which is also a classification or regression method that works by building several decision trees during training.

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
```

```
dataset = pd.read_csv('./clean_transaction.csv')
dataset.sample(frac=1, random_state=42).reset_index(drop=True, inplace=True)
dataset.head()
```

	FLAG	Avg_min_between_sent_tnx	Avg_min_between_received_tnx	Time_Diff_between_first_and_last_(Mins)	Sent_tnx	Received_Tnx	Number_of_Created_Cont
0	0	844.26	1093.71	704785.63	721	89	
1	0	12709.07	2958.44	1218216.73	94	8	
2	0	246104.54	2434.02	516729.30	2	10	
3	0	10219.60	15785.09	397555.90	25	9	
4	0	36.61	10707.77	382472.42	4598	20	

Figure 13: Random Forest libraries

The data was also divided to train and test, and for the Random Forest model the results were improved using 70% of the total data.

```
from sklearn.model_selection import train_test_split
```

```
# Better results with 70% testing
test_size = 0.7
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=test_size, random_state=42)

print(X_train, X_test)
print(y_train, y_test)
```

```
      Avg_min_between_sent_tnx  Avg_min_between_received_tnx  \
3808                    105.53                       1468.66
1138                      0.00                          0.00
2257                    125.71                       2436.45
7588                     12.30                       4428.66
1787                    164.08                          0.82
...                        ...                           ...
5734                      0.00                      16197.17
5191                      0.00                          0.00
5390                      0.00                          0.00
860                     165.15                          0.00
7270                      2.04                          0.11

      Time_Diff_between_first_and_last_(Mins)  Sent_tnx  Received_Tnx  \
3808                                1054850.45       560           678
1138                                   196.07         1             1
2257                                496574.42       132           197
7588                                754914.00       166           170
1787                                   329.78         2             2
```

Figure 14: Random Forest Train and Test

Random Forest had the best result within the proposed models, with an accuracy of 97% and is the ideal model for analyzing fraud on the Ethereum network.

```
from sklearn import metrics

cm = metrics.confusion_matrix(y_test, pred_model)
```

```
np.unique(y_test, return_counts=True)
```

```
(array([0, 1], dtype=int64), array([5367, 1522], dtype=int64))
```

```
print (classification_report(y_test, pred_model))
```

```
              precision    recall  f1-score   support

           0       0.97      0.98      0.98      5367
           1       0.94      0.91      0.93      1522

    accuracy                           0.97      6889
   macro avg       0.96      0.95      0.95      6889
weighted avg       0.97      0.97      0.97      6889
```

Figure 15: Random Forest Results