National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Boluwatife Joseph Omoworare
Student ID: 20185944

School of Computing
National College of Ireland

Supervisor: Hicham Rifai

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Boluwatife Joseph Omoworare

Student ID: 20185944

Programme: Data Analytics **Year:** 2021

Module: Master of science research project

Lecturer: Hicham Rifai

Submission Due Date: 16/12/2021

Project Title: Feature Selection and Machine Learning Algorithms for an Improved Credit Card Fraud Detection System

Word Count: 1294 **Page Count:** 20

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

Boluwatife Joseph Omoworare

20185944

1. INTRODUCTION

This is a step-by-step breakdown and guide to the implementation of the project, detailing the programming languages used in building the models, the specs of the system on which the model was built, the minimum reequipment a system must possess to successfully run the models, and the algorithms that applied in developing the model.

2. Hardware requirements

The system requirement includes both the software and Hardware requirements used for the implementation of this research. These are highlighted in the table below.

	Minimum Requirement	Current System Setup
Software	Chrome browser	Chrome browser
	Python Programming Language v3	Python Programming Language v3.8.3
	Anaconda Individual Edition v4	Anaconda Individual Edition v4.8.3
	Windows 8	Mac OS 11
Hardware	8GB RAM	16GB RAM
	150GB HDD	500GB HDD

Figure 1 system setup

Note: the computer system is working at optimum capacity at the time of implementation

3. Environment requirement

for the implementation of this research, 3 primary tools should be available on your workstation

I. Python programming language
II. Anaconda Navigator IDE
III. Jupyter notebook

3.1 Python programming language

To set up python on your workstation, the first step is to visit https://www.python.org/downloads/ and download version 3.8 for your operating system 3.8, for further documentation on installation and basic python setup visit https://www.python.org/downloads/ .

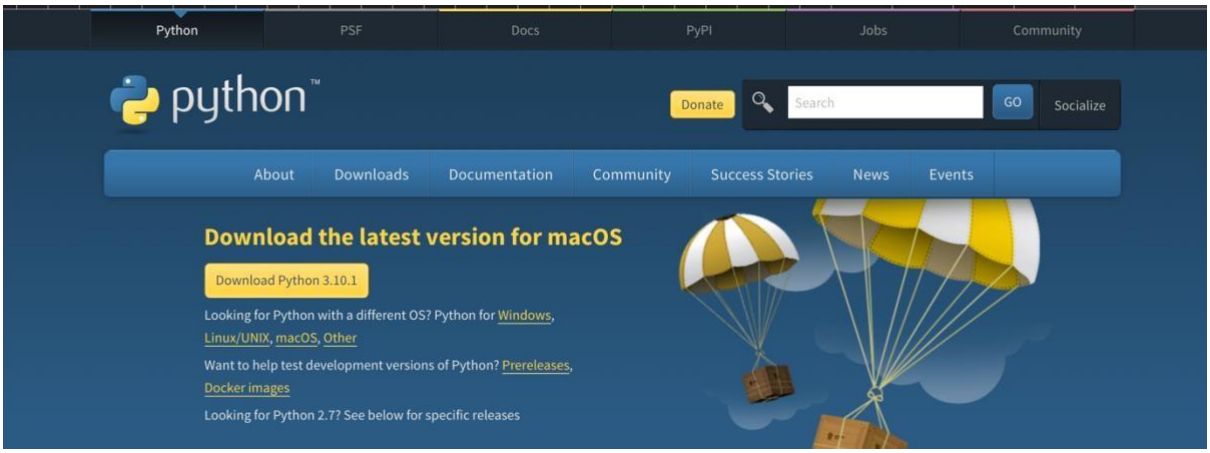


Figure 2 python website

Note: Python 3.8 is the version on my current workstation newer version could be used but older versions may not possess some of the necessary libraries needed for implementation

3.2. **Anaconda Navigator IDE**

To install the anaconda on your system visit https://www.anaconda.com/products/individual the website automatically detects the operating system you are working on and presents a download link to the latest version of anaconda required by your operating system.

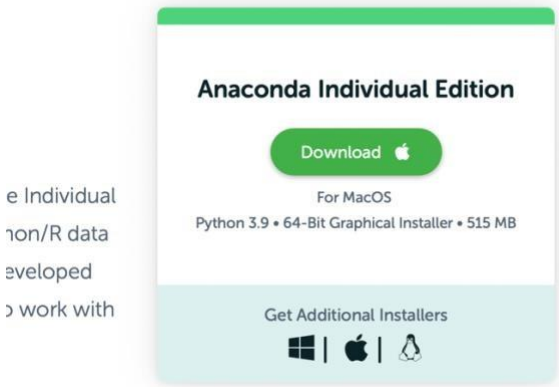
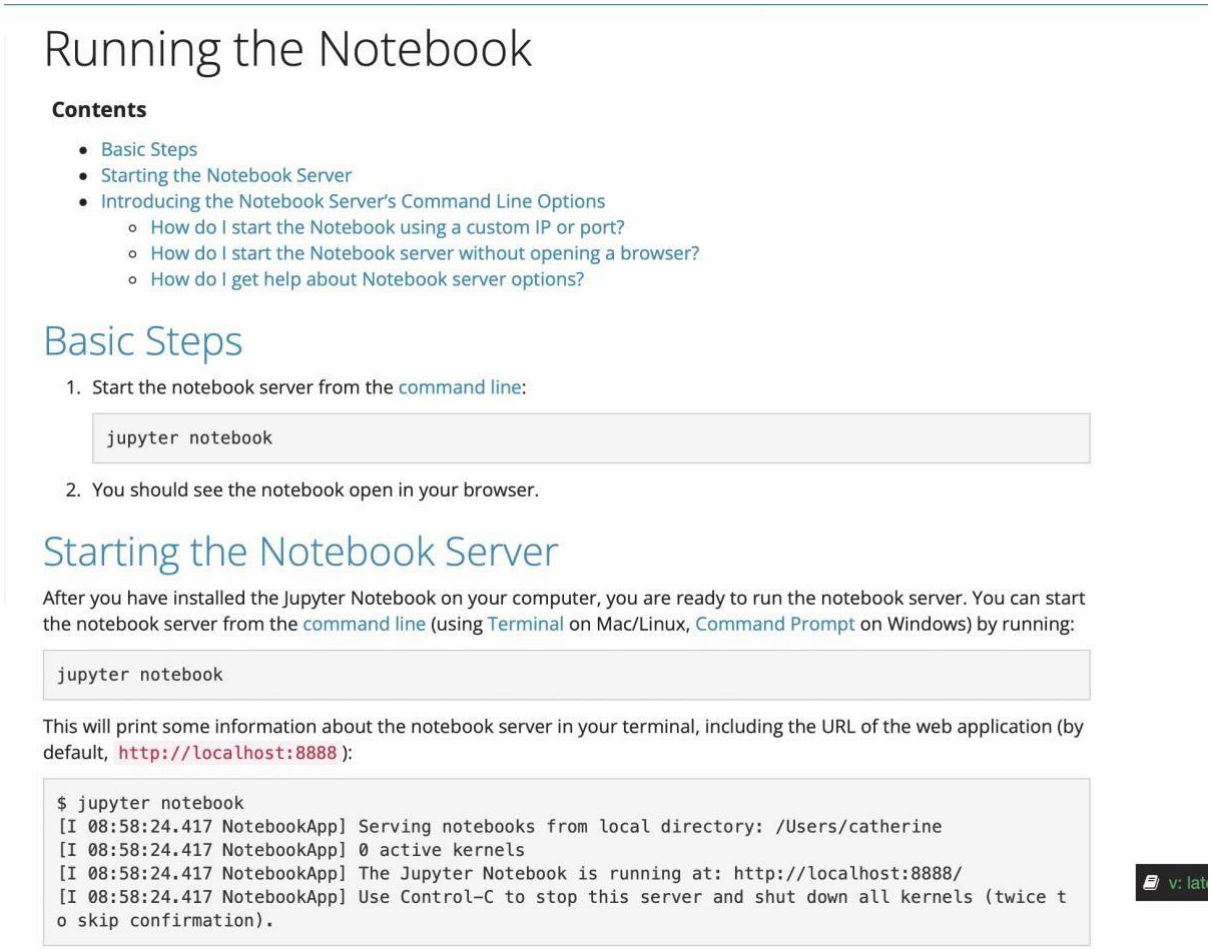


Figure 3 Anaconda download link

3.3. **Jupyter notebook**

Jupyter notebook used in these environment was accessed through Anaconda Navigator IDE, so you have to ensure that anaconda navigator IDE has been properly installed, to install jupyter notebook in the anaconda environment visit https://testjupyter.readthedocs.io/en/latest/install.html there is a notebook linked to the page for step by step installation of jupyter notebook.



Running the Notebook

Contents

- Basic Steps
- Starting the Notebook Server
- Introducing the Notebook Server's Command Line Options
  - How do I start the Notebook using a custom IP or port?
  - How do I start the Notebook server without opening a browser?
  - How do I get help about Notebook server options?

Basic Steps

1. Start the notebook server from the command line:

```
jupyter notebook
```

2. You should see the notebook open in your browser.

Starting the Notebook Server

After you have installed the Jupyter Notebook on your computer, you are ready to run the notebook server. You can start the notebook server from the command line (using Terminal on Mac/Linux, Command Prompt on Windows) by running:

```
jupyter notebook
```

This will print some information about the notebook server in your terminal, including the URL of the web application (by default, `http://localhost:8888`):

```
$ jupyter notebook
[I 08:58:24.417 NotebookApp] Serving notebooks from local directory: /Users/catherine
[I 08:58:24.417 NotebookApp] 0 active kernels
[I 08:58:24.417 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/
[I 08:58:24.417 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice t
o skip confirmation).
```

Figure 4 jupyter installation guide

4. Implementation

4.1 data source

The dataset used was a synthetic dataset created by a researcher in the IBM TJ Watson Research Centre, containing over 24 million records, and 15 columns. The link to the dataset is http://arxiv.org/abs/1910.03033

4.2 preprocessing (notebook file 1)

Data preprocessing was carried out on the dataset to make it more suitable for our model the preprocessing steps can be seen in the attached notebook file named *CC Fraud Detection- Preprocessing.* A breakdown of the steps is detailed below.

Importing necessary libraries

Import Python Libraries

```
In [1]: #libraries used for importing and exploring the data
        import pandas as pd
        import numpy as np
        from pandas import Series, DataFrame
        from imblearn.over_sampling import SMOTE
        from sklearn.preprocessing import MinMaxScaler

        #libraries for data visualizaiton
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline

        #Default plot size and type
        plt.rc("font", size=14)
        plt.figure(figsize=(7,5))
        plt.gray()

        <Figure size 504x360 with 0 Axes>
```

Figure 5 pre-processing libraries

libraries necessary for model training and testing, performance evaluation, and visualization were also imported at the beginning of notebook2 and 3

```
In [1]: #libraries used for importing and exploring the data
        import pandas as pd

        #libraries used for model training and testing
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.ensemble import RandomForestClassifier

        #libraries used for performance evaluation
        from sklearn import metrics
        from sklearn.metrics import f1_score
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import confusion_matrix
        from mlxtend.plotting import plot_confusion_matrix
        from sklearn.metrics import roc_curve, auc
        from sklearn.metrics import classification_report
        from sklearn.metrics import precision_recall_fscore_support as score

        #libraries for data visualizaiton
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline

        #Default plot size and type
        plt.rc("font", size=14)
        plt.figure(figsize=(7,5))
        plt.gray()

        <Figure size 504x360 with 0 Axes>
```

Figure 6 other libraries

Exploratory data analysis was carried out on the dataset to help better understand the dataset, at this point the shape and size of the dataset, data types of the columns, and missing values were detected.

NOTE:

From the understanding of the data structure, the following steps were performed to adequately process the data before applying feature selection and machine learning algorithms.

1. Delete columns with a high number of null values. These were the merchant state, zip code, and errors columns.
2. Rename "Is Fraud?" to "Fraud".
3. Remove the $ symbol from the amount column and converted it from object to float data type.
4. Encode object columns like the time, use chip, merchant name, and fraud columns.
5. Balance the dataset using random under-sampling and oversampling. Random Under-sampling was performed on the majority class (non-fraud) to reduce it to 1million records after which random oversampling was performed on the minority class (fraud) to increase the number of records from 29k to 1million.

6. Scale Amount column using Robust Scaler.

They were implemented in the order listed above in the notebook file

The processed dataframe was saved as processed_data.csv, a new dataframe to be used in model building

```
In [23]: #Save processed dataframe which would be imported for model training and testing
         new_df.to_csv(r'processed_data.csv', index = False, header = True)
```

Figure 7 processed_data.csv

4.3. Feature Selection (notebook 2)

The next step is to apply feature selection processed dataframe by plotting the correlation matrix and checking the Correlation between other columns and Label columns to determine the columns to drop before modelling. this can be seen in the notebook named *Modelling with Feature Engineering*.

```
In [3]: corr_matrix = df.corr()
        corr_matrix = abs(corr_matrix)
        f, ax = plt.subplots(figsize=(20, 9))
        sns.heatmap(corr_matrix, vmax=0.8, vmin=0.05, annot=True)
Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x7fdfae7e1850>
```

Figure 8 correlation matrix

Correlation between other column and Label column

```
In [4]: corr_matrix['Fraud'].sort_values(ascending=False)
Out[4]: Fraud            1.000000
        Merchant City    0.312880
        Use Chip         0.233853
        scaled_amount    0.210819
        MCC              0.164482
        Time             0.080569
        Merchant Name    0.062269
        Month            0.048687
        Card             0.039462
        Year             0.030187
        Day              0.024611
        User             0.008628
        Name: Fraud, dtype: float64
```

Figure 9 correlation check

4.4 split dataset

The dataset was split 80:20 for training and testing respectfully

Split Dataset For Training & Testing

```
In [13]: #80:20 Split for Training & Testing Data respectively
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=1)
```

Figure 10 dataset split

Note: this was done twice in the implementation first in notebook 2 and again in notebook 3 as all the models require that the dataframe be split before modelling

4.5 model building

4 models were built in total 2 random forest classifier models with feature selection and one without feature selection and 2 logistic regression classifier models were also built one with feature selection and the other without feature selection.

Model 1: Random Forest Classifier

```
In [14]: FE_RandForest= RandomForestClassifier(n_estimators=100, max_depth=None, random_state=123, max_leaf_nodes=None)
         FE_RandForest= FE_RandForest.fit(X_train, y_train)
         FE_RandForest
Out[14]: RandomForestClassifier(random_state=123)

In [15]: y_pred_FE_RF = FE_RandForest.predict(X_test)
         FE_RFacc = FE_RandForest.score(X_test, y_test)

In [16]: print('Accuracy score= {:.4f}'.format(FE_RFacc))
         Accuracy score= 0.9833
```

Figure 11 random forest classifier

Model 2: Logistic Regression Classifier

```
In [20]: FE_LogReg = LogisticRegression(tol = 1e-6, penalty = 'l2', solver = 'liblinear', random_state = 123)
         FE_LogReg = FE_LogReg.fit(X_train, y_train)
         FE_LogReg
Out[20]: LogisticRegression(random_state=123, solver='liblinear', tol=1e-06)

In [21]: y_pred_FE_LR = FE_LogReg.predict(X_test)
         FE_LRacc = FE_LogReg.score(X_test, y_test)
         print('Accuracy score= {:.4f}'.format(FE_LRacc))
         Accuracy score= 0.6986
```

Figure 12 logistic regression classifier

Note: the models with feature selection were implemented in notebook 2 and the models without feature selection were implemented in notebook 3

5. Evaluation

   All the models were in 4 stages listed below

   I. Accuracy score
   II. Confusion matrix
   III. Classification report
   IV. AUC

5.1. Accuracy score

This is used to determine how accurate the model is in determining fraudulent and legitimate transactions. it is called using a simple line of code after creating each model.

```
In [16]: print('Accuracy score= {:.4f}'.format(FE_RFacc))
         Accuracy score= 0.9833
```

Figure 13 accuracy score

4.2. Confusion matrix

This plots the true positive, false positive, false negative, and true negative of the model to further determine how effective the model is.

```
In [17]: print("confusion matrix")
         CMRF = confusion_matrix(y_test, y_pred_FE_RF)
         fig, ax = plot_confusion_matrix(conf_mat = CMRF,figsize=(10, 10),
                                         show_absolute=True,
                                         show_normed=True,
                                         colorbar=True)

         ax.set_xticklabels(['']  + class_labels)
         ax.set_yticklabels(['']  + class_labels)
         plt.show()
         confusion matrix
```

Figure 14 confusion matrix

4.3. Classification report

The classification report shows the following precision, recall, f1-score, and accuracy for both labelled columns (non-fraud and fraud) as seen below.

	precision	recall	f1-score	support
Non Fraud	0.9832	0.9834	0.9833	199885
Fraud	0.9834	0.9833	0.9833	200115
accuracy			0.9833	400000
macro avg	0.9833	0.9833	0.9833	400000
weighted avg	0.9833	0.9833	0.9833	400000

Figure 15 classification report

4.4. AUC

This is a graph that represents the accuracy and effectiveness of the model

AUC for Random Forest

```
In [19]: fprFE_RF, tprFE_RF, thresholds = roc_curve(y_test, y_pred_FE_RF)
         roc_auc = auc(fprFE_RF, tprFE_RF)
         plt.figure(figsize=[8,6])
         plt.title('Receiver Operating Characteristic')
         plt.plot(fprFE_RF, tprFE_RF, 'b', linewidth= 4, label = 'AUC = %0.4f' % roc_auc)
         plt.legend(loc = 'lower right')
         plt.plot([0, 1], [0, 1],'r--')
         plt.xlim([0, 1])
         plt.ylim([0, 1])
         plt.ylabel('True Positive Rate')
         plt.xlabel('False Positive Rate')
         plt.show()
```

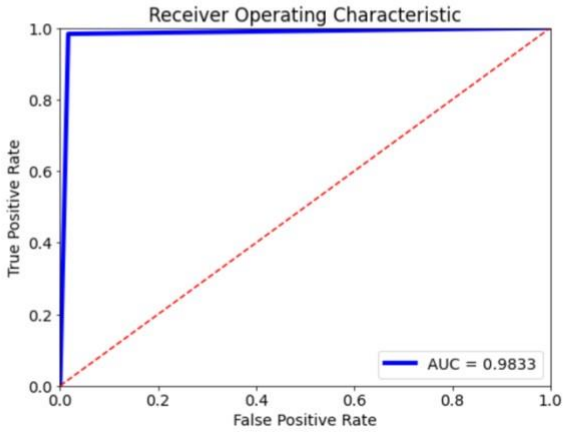
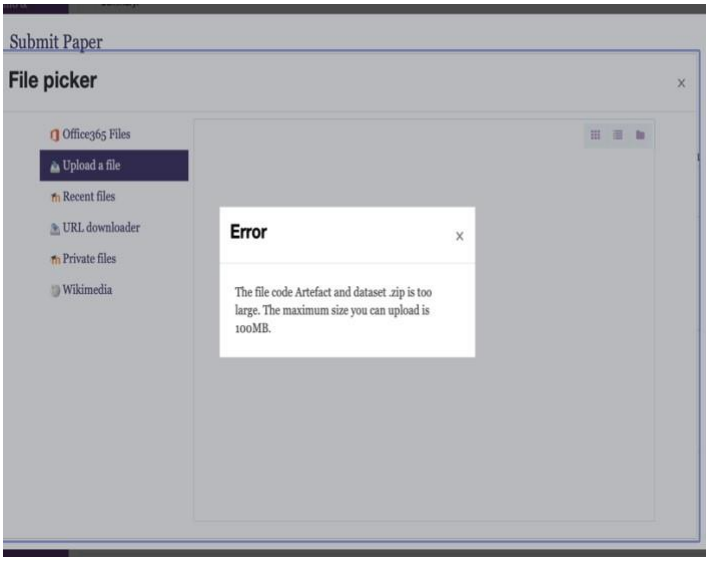


Figure 16 AUC

Note: every model was individually evaluated with these 4 evaluation techniques.

Note: my original zip file for code Artefact and the dataset is 114.1mb and the maximum capacity for upload on Moodle is 100mb so I excluded my original dataset from the zip file.



Kindly see alternative links to the datasets below.

Google drive link

https://drive.google.com/file/d/1FiJWl37lzanicDYSiU_GTqLzepXXxk5Y/view?usp=sharing

Download link. http://arxiv.org/abs/1910.03033