National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Taranjyot Chawla
Student ID: X21153078

School of Computing
National College of Ireland

Supervisor: Prof. Jorge Basilio

National College of Ireland Project
Submission Sheet School of Computing

National College of Ireland

Student Name:	Taranjyot Chawla
Student ID:	X21153078
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Prof. Jorge Basilio
Submission Due Date:	15/12/2022
Project Title:	Configuration Manual
Word Count:	923
Page Count:	15

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:	15th December 2022

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

Taranjyot Chawla
X21153078

1 Introduction

The main motive of the presented research is to Predict whether a particular transaction can be fraudulent or not. In this research, Logistic Regression, Random Forest Classifier, Decision Tree Models are used and these models are trained two times, firstly including all the features from the dataset and secondly eliminating some non-relevant features. The whole project is implemented using python libraries. The configuration manual is a guide to explain the implemented project and tools and setups required to run the project.

2 System Specification

Processor : Intel® Core™ i5
Memory(RAM) Installed : 8 GB
Storage: 128 GB SSD

3 Software Tools

The tools required for this project are:

- Anaconda Navigator
- Python
- Jupyter Notebook

4 Software Installation

This is a step by step explanation of the implementation.
Download and install python from https://www.python.org/downloads/


Install Python
Welcome to the Python Installer
Introduction
Read Me
Licence
Destination Select
Installation Type
Installation
Summary
This package will install Python 3.9.6 for macOS 10.9 or later.
Python for macOS consists of the Python programming language interpreter and its batteries-included standard library to allow easy access to macOS features. It also includes the Python integrated development environment, IDLE. You can also use the included pip to download and install third-party packages from the Python Package Index.
At the end of this install, click on Install Certificates to install a set of current SSL root certificates.
Go Back
Continue

Figure 1:


Install Python
Important Information
Introduction
Read Me
Licence
Destination Select
Installation Type
Installation
Summary
This package will install Python 3.9.6 for macOS 10.9 or later for the following architecture(s): x86_64.
Certificate verification and OpenSSL
This package includes its own private copy of OpenSSL 1.1.1. The trust certificates in system and user keychains managed by the Keychain Access application and the security command line utility are not used as defaults by the Python ssl module. A sample command script is included in /Applications/Python 3.9 to install a curated bundle of default root certificates from the third-party certifi package (https://pypi.org/project/certifi/). Double-click on Install Certificates to run it.
The bundled pip has its own default certificate store for verifying download connections.
Install Options
You can control some aspects of what is installed by this package. To see the options, click on the Customize button in the Installation Type step of the macOS installer app. Click on a package name in the list
Print...
Save...
Go Back
Continue

Figure 2:

Figure 3:

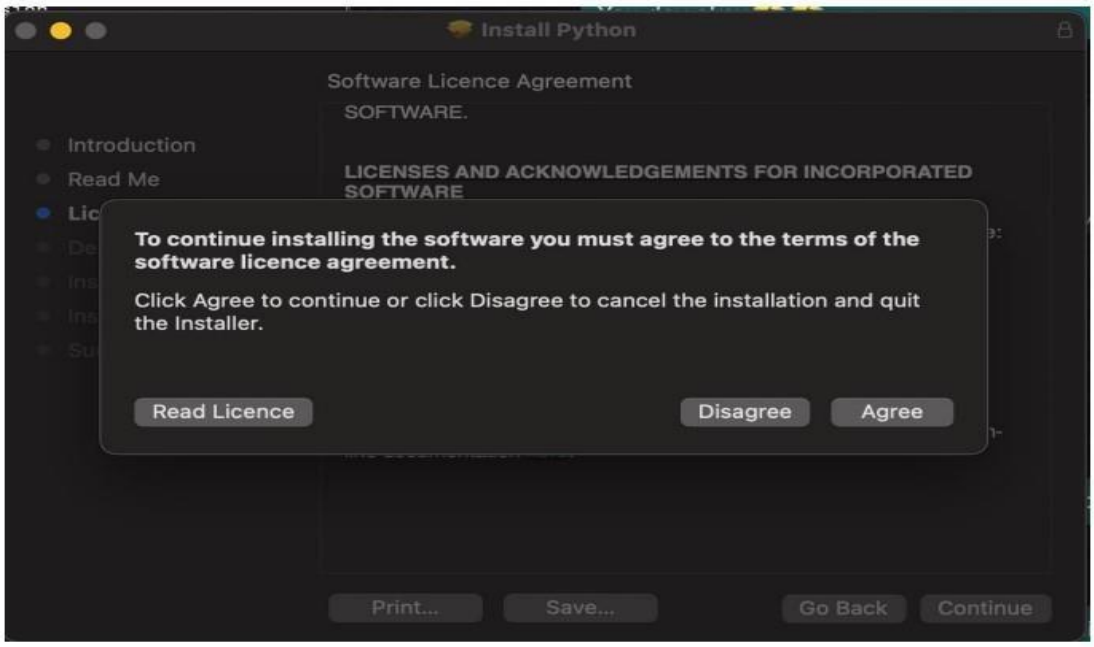

Figure 4:

Figure 5:

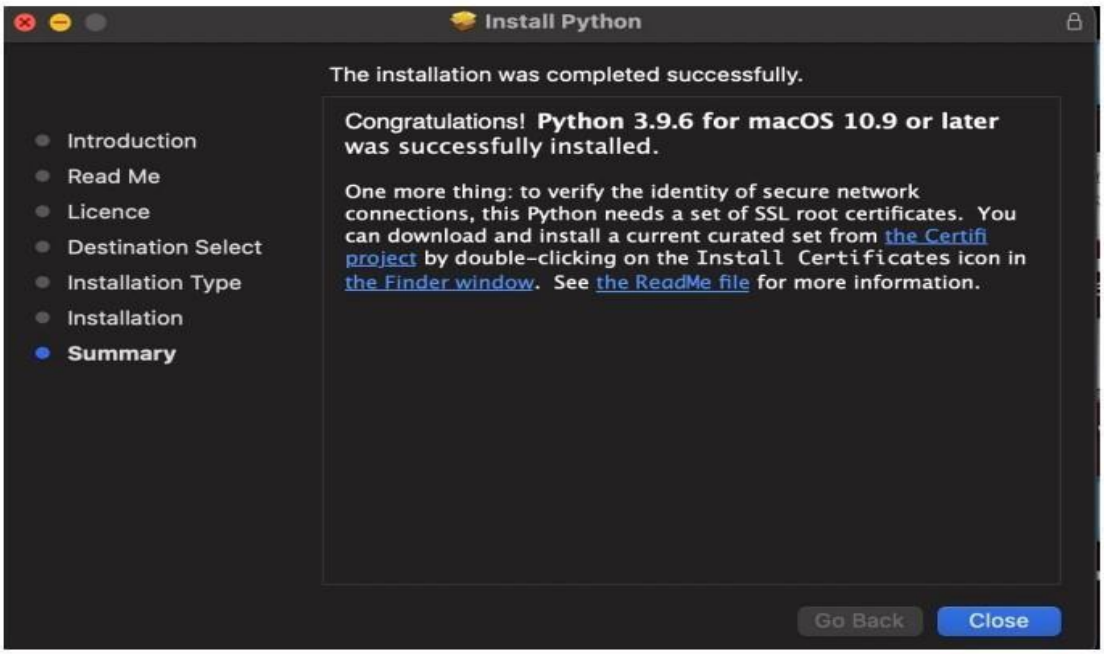

Figure 6:

Downloading and installing anaconda from https://www.anaconda.com/

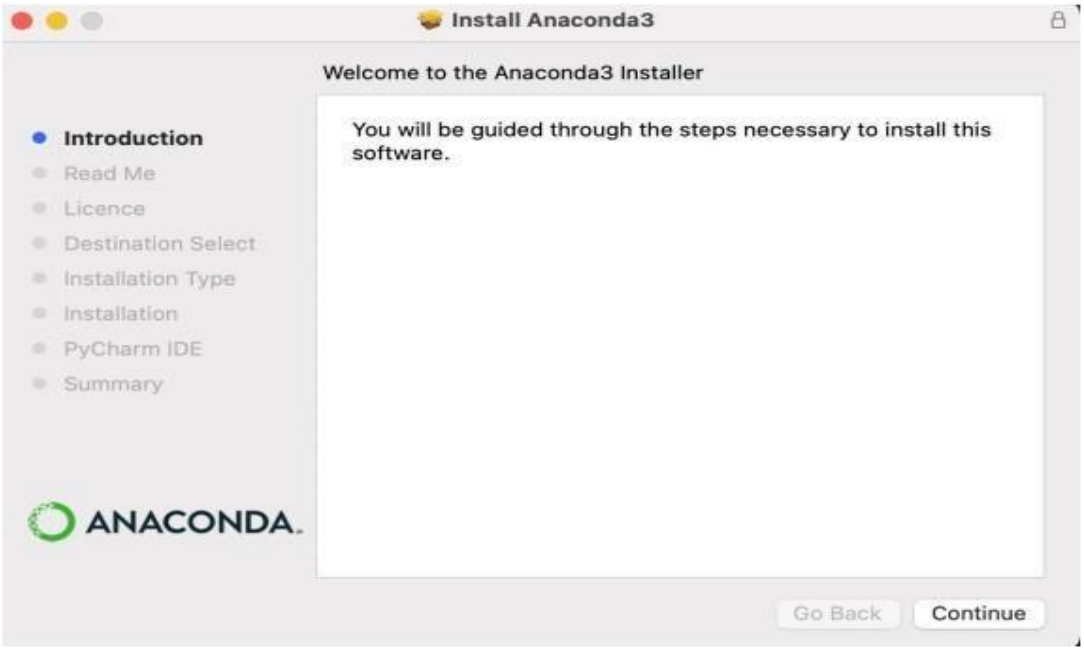

Figure 7:

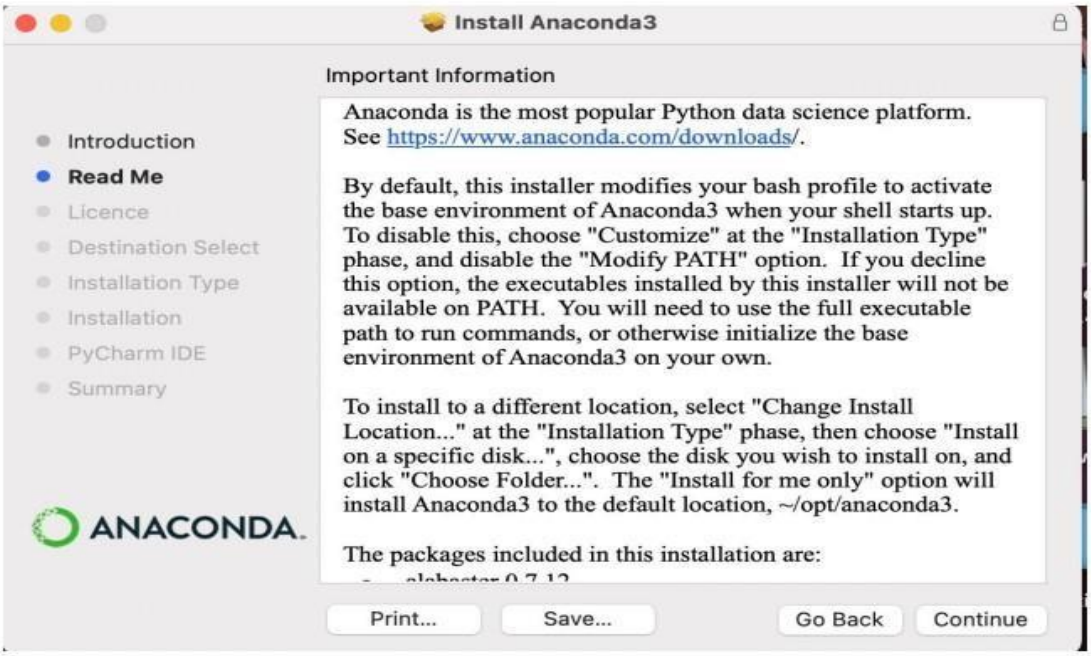

Figure 8:

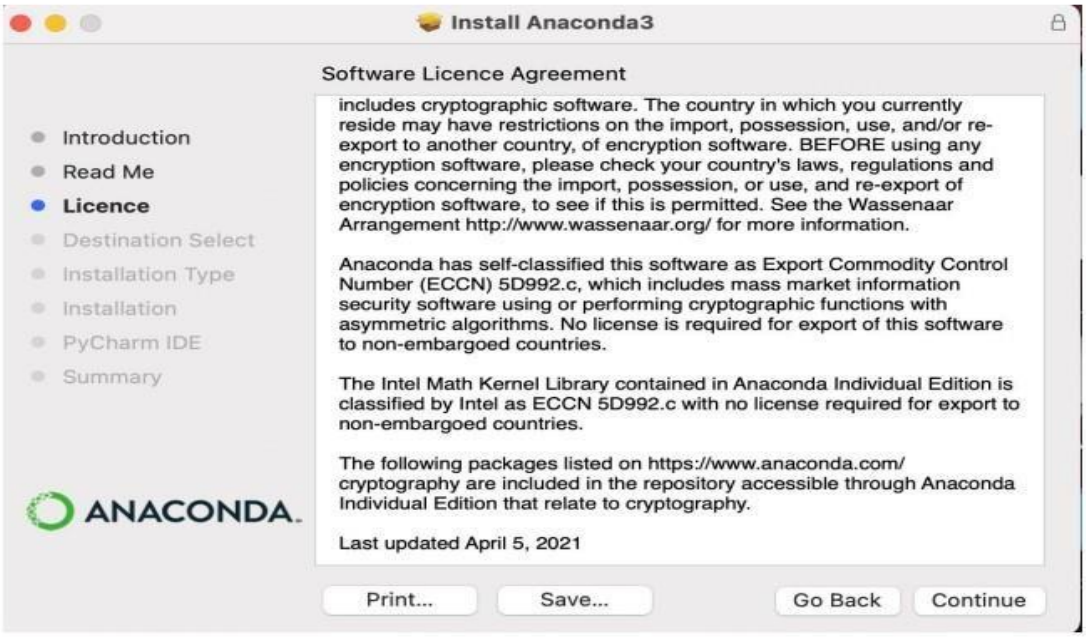

Figure 9:

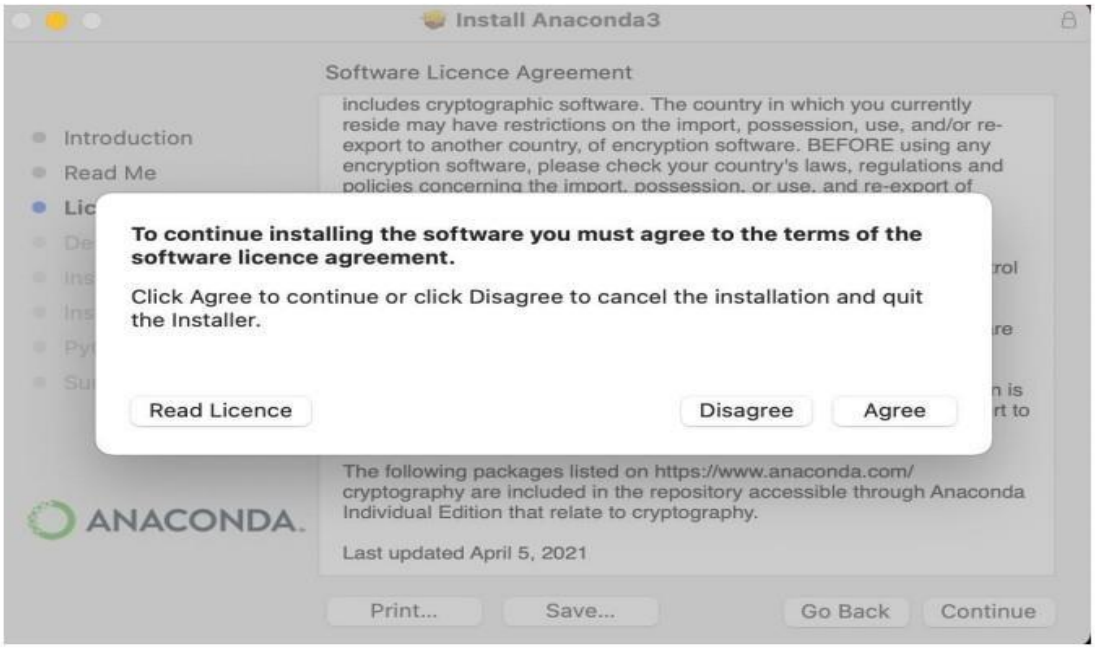

Figure 10:

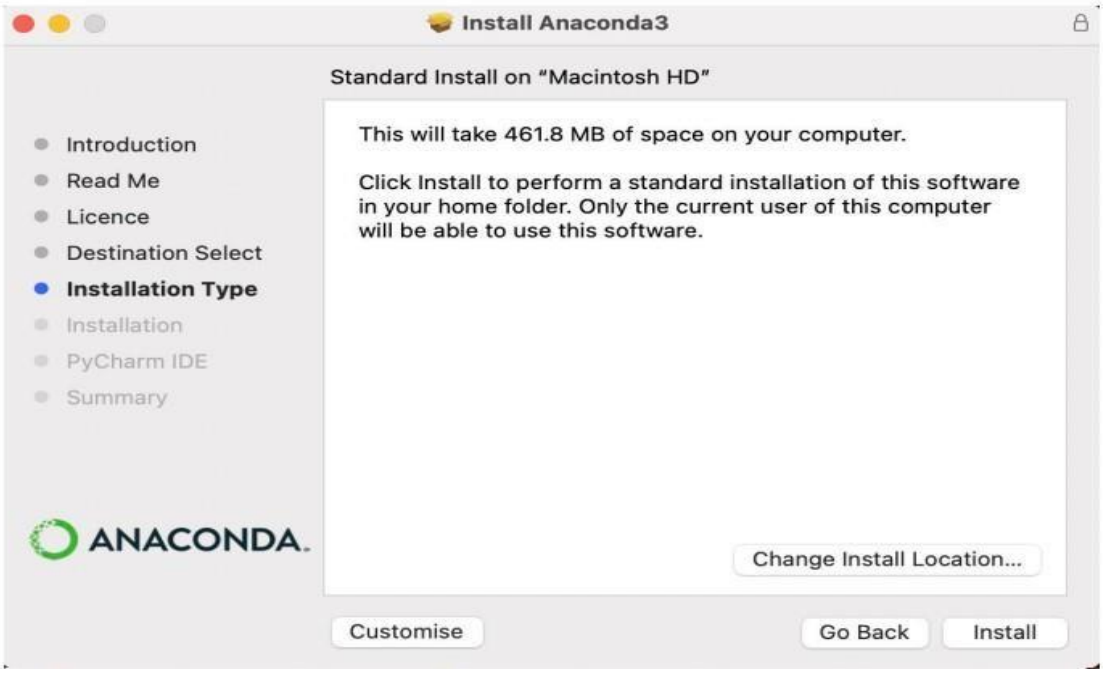

Figure 11:

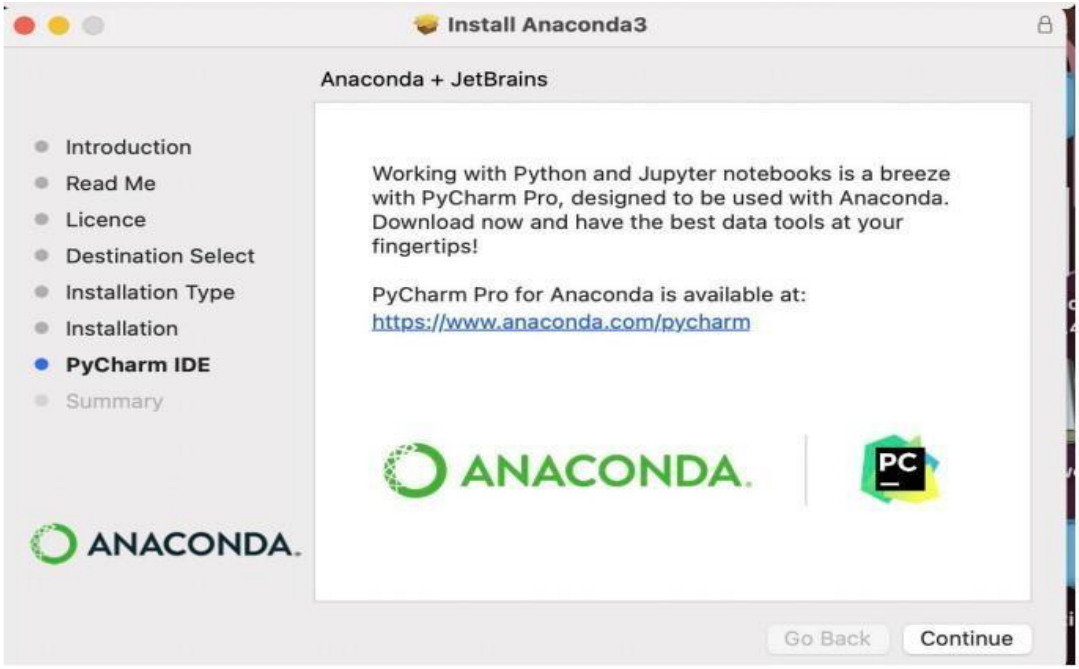

Figure 12:

5 Implementation

The following packages and libraries are utilized:

NumPy
Pandas
Matplotlib
Seaborn
Sklearn.

5.1 Importing Libraries

The following stpes were taken to import the libraries:

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from seaborn import heatmap
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report, f1_score, accuracy_score, precision_recall_curve,
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, f1_score, accuracy_score, precision_recall_curve,
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report, f1_score, accuracy_score, precision_recall_curve,
```

Figure 13: Importing required Libraries and Packages

5.2 Loading The data

```
# Reading the File in a panda data frame

payments = pd.read_csv('/Users/taranjyotsingh/Desktop/Online Fraud Detection Dataset/Fraud Detection.csv')
print(payments.head())
```

Figure 14: Loading the data

5.3 Data Analysis, preparation and visualisation

5.3.1 Columns/Features in Dataset

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	\
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	

	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	M1979787155	0.0	0.0	0	0
1	M2044282225	0.0	0.0	0	0
2	C553264065	0.0	0.0	1	0
3	C38997010	21182.0	0.0	1	0
4	M1230701703	0.0	0.0	0	0

Figure 15: 5 Top Records of the dataset

```
# Printing all Columns of dataset
print(list(payments.columns))

#  Printing the shape of data
print(f'The dataset has shape {payments.shape}')
```

```
['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig', 'nameDest', 'oldbalanceDest', 'newbalanceDe
st', 'isFraud', 'isFlaggedFraud']
The dataset has shape (6362620, 11)
```

Figure 16: Printing the features of dataset

5.3.2 Types of Payments

```
# exploring the type of transaction
print(payments['type'].value_counts())
```

```
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: type, dtype: int64
```

Figure 17: Types of Payments

5.3.3 NameOrig Feature Analysis

```
#Investigating to check unique customers
payments.nameOrig.unique()
```

```
array(['C1231006815', 'C1666544295', 'C1305486145', ..., 'C1162922333',
       'C1685995037', 'C1280323807'], dtype=object)
```

Figure 18: nameorig Feature

```
#investigating to see how many times a customer started a transaction
payments.nameOrig.value_counts()
```

```
C1902386530    3
C363736674     3
C545315117     3
C724452879     3
C1784010646    3
              ..
C98968405      1
C720209255     1
C1567523029    1
C644777639     1
C1280323807    1
Name: nameOrig, Length: 6353307, dtype: int64
```

Figure 19: Value Count of feature nameorig

5.3.4 namedest Feature Analysis

```
##Checking which recipients stand out


payments.nameDest.unique()
```

```
array(['M1979787155', 'M2044282225', 'C553264065', ..., 'C1850423904',
       'C1881841831', 'C2080388513'], dtype=object)
```

Figure 20: namedest Feature

5.3.5 Checking mean of Amount

```
#Checking the average amount
payments['amount'].mean()
```

```
179861.90354912292
```

Figure 21: Mean of Amount Feature

5.3.6 Checking Corelation between features

```
# show Correlation
correlation = payments.corr()
print(correlation['isFraud'].sort_values(ascending = False))
```

```
isFraud           1.000000
amount            0.076688
isFlaggedFraud    0.044109
step              0.031578
oldbalanceOrg     0.010154
newbalanceDest    0.000535
oldbalanceDest   -0.005885
newbalanceOrig   -0.008148
Name: isFraud, dtype: float64
```

Figure 22: Co relation

5.4 Separating Fraudulent and Non-fraudulent transactions from our target variable

```
# Sepearating Fraudulent transaction from non fraudulent

Fraudulent_Transaction = payments[payments.isFraud ==1]
Not_Fraudulent_Transaction = payments[payments.isFraud ==0]

print('Fraudulent Transaction: {}'.format(len(Fraudulent_Transaction)))
print('Not Fraudulent Transaction: {}'.format(len(Not_Fraudulent_Transaction)))
```

```
Fraudulent Transaction: 8213
Not Fraudulent Transaction: 6354407
```

Figure 23: Separating Fraudulent and non fraudulent transactions

5.4.1 Handeling the class imbalance

To handle the class imbalance we have unsersampled the data by using random sampler.

```
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
```

```
Y = payments['isFraud']
```

```
features = payments.drop('isFraud', axis=1)
X=features
```

```
Non_Fraudulent_Sample = Not_Fraudulent_Transaction.sample(n=8213)
```

```
# Combining the two datasets

Final_dataset = pd.concat([Non_Fraudulent_Sample, Fraudulent_Transaction], axis=0)
```

```
Final_dataset['isFraud'].value_counts()
```

```
0    8213
1    8213
Name: isFraud, dtype: int64
```

Figure 24: Undersampling of Fraudulent transactions

5.5 Implementing HotEncoding

Applying one hot encoding, In order to convert categorical variables to a form which is suitable for the training of models.

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
```

```
encoder = OneHotEncoder(handle_unknown='ignore', sparse=False, drop=None,)
```

```
#perform one-hot encoding on 'type' column
encoder_df =  pd.get_dummies(Final_dataset, columns=['type','nameOrig','nameDest'], prefix=['type','nameOrig',
                                                                                             'nameDest'])
```

Figure 25: One Hot Encoding

5.6 Spliting the data into 70 percent training and 20 percent testing

```
Y = encoder_df['isFraud']
```

```
features = encoder_df.drop('isFraud', axis=1)
```

```
X = features
```

```
#create X_train, X_test, Y_train, Y_test
# using test_size of 20%

X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2, stratify=Y, random_state=2)
```

Figure 26: Split of data into training and testing

5.7 Machine Learning Algorithms

5.7.1 Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression()
```

```
#Training model with Training data
model.fit(X_train, Y_train)
```

```
model_pred = model.predict(X_test)
```

```
# Obtain model probabilities
probs = model.predict_proba(X_test)
```

```
#importing the methods
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report, f1_score, accuracy_score, precision_recall_curve,
roc_auc_score
```

```
print('\nClassification Report:')
print(classification_report(Y_test, model_pred))
```

Figure 27: Logistic Regression

```
# ACCURACY SCORE
print('Accuracy:',accuracy_score(Y_test, model_pred))
```

```
Accuracy: 0.9038344491783323
```

Figure 28: Accuracy

```
print('AUC Score:')
print(roc_auc_score(Y_test, probs[:,1]))
```

```
AUC Score:
0.9589364348057696
```

Figure 29: AUC for Logistic Regression

```
#define metrics
y_pred_proba = model.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(Y_test,  y_pred_proba)
auc = metrics.roc_auc_score(Y_test, y_pred_proba)


#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
```

Figure 30: ROC for Logistic Regression

```
# Calculate average precision and the P-R curve
average_precision = average_precision_score(Y_test, model_pred)
average_precision
```

```
0.86823886147918 69
```

Figure 31: Logistic Regression Precision

5.7.2 Random Forest

```
from sklearn.ensemble import RandomForestClassifier

# Define the model as the random forest
model = RandomForestClassifier(random_state=5, n_estimators=20)

model.fit(X_train,Y_train)

RandomForestClassifier(n_estimators=20, random_state=5)

model_pred = model.predict(X_test)

# Obtain model probabilities
probs = model.predict_proba(X_test)

#importing the methods
from sklearn.metrics import confusion_matrix, classification_report, f1_score, accuracy_score, precision_recall_curve,
```

Figure 32: Training Random Forest

```
# Print classification report using predictions
print('Classification_Report:\n',classification_report(Y_test, model_pred))
```

```
Classification_Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1643
           1       0.98      0.98      0.98      1643

    accuracy                           0.98      3286
   macro avg       0.98      0.98      0.98      3286
weighted avg       0.98      0.98      0.98      3286
```

Figure 33:

```
# Print ROC_AUC score using probabilities
print('AUC Score:')
print(roc_auc_score(Y_test, probs[:, 1]))
```

```
AUC Score:
0.996403340089033
```

Figure 34: AUC Score for Random Forest

```
# Print confusion matrix using predictions
pd.DataFrame(confusion_matrix(Y_test, model_pred),
             columns=['Predicted Negative(0) ', 'Predicted Positive(1)'],
             index=['Actually Negative(0)', 'Actually Positive(1)'])
```

	Predicted Negative(0)	Predicted Positive(1)
Actually Negative(0)	1607	36
Actually Positive(1)	39	1604

Figure 35: Confusion Matrix for Random Forest

```
# Calculate average precision and the P-R curve
average_precision = average_precision_score(Y_test, model_pred)
average_precision
```

```
0.9667013048706263
```

Figure 36: Precision Random Forest

6 Conclusion

The implementation of the code is shown in the document and the codes are commented for better understanding, for better readability the document is divided into sections and subsections.

References:

Artifacts.zip

21153078_Research Project