Configuration Manual

MSc Research Project
Olaitan Olanlokun

Forename Surname
Student ID: x20113897

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Olaitan Olanlokun……………………………………………………………………

Student ID: ………20113897……………………………………………………………………………..……

Programme: ……Cyber Security……………………………… **Year:** ……2021………………..

Module: …MSc Research Project ……………………………………………………….………

Supervisor: …… Ross Spelman …………………………………………………………………….………

Submission Due Date: …………16/08/2021…………………………………………………………………….………

Project Title: … Feature Based Selection Technique for Credit Card Fraud Detection System Using Machine Learning Algorithms

……………………………………………………………………………………………………….………

Word Count: ………185……………………… **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: ……15/08/2021…………………………………………………………………………………

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):	

1 Introduction

The configuration manual serves as a guide to reproduce the project proposed using machine learning algorithms to detect spear phishing emails in organizations. This manual also talks about the overall setup needed to install all tools required for the implementation of the project.

2 System Specification

The configuration of the system used is:

- MacBook Pro (13-inch, M1, 2020)
- Memory: 8GB
- Hard Drive: 512GB

3 Software Tools

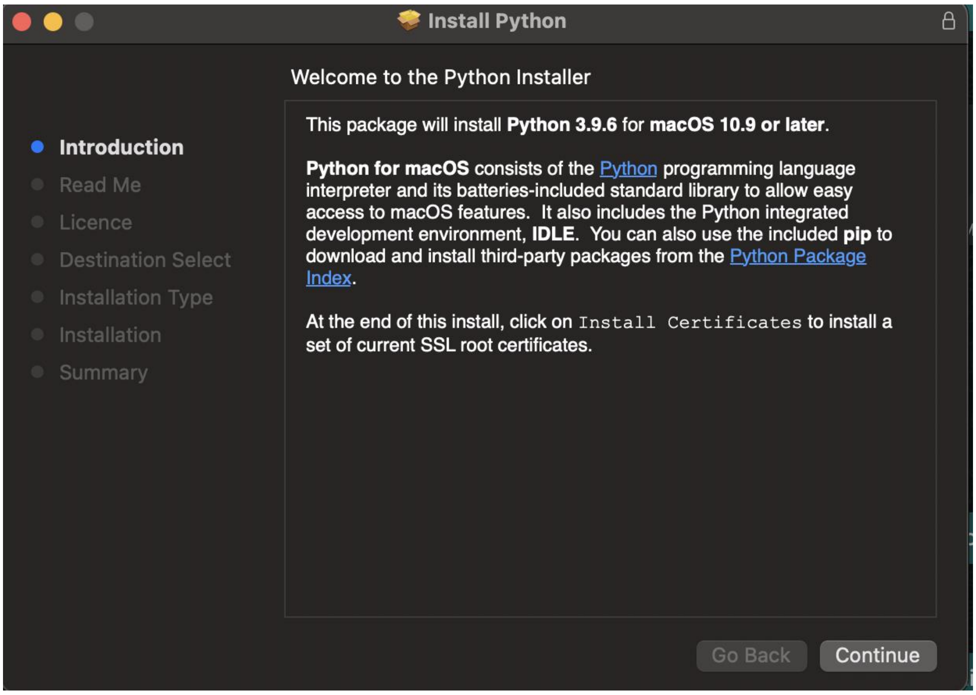
The tools used for this project are:

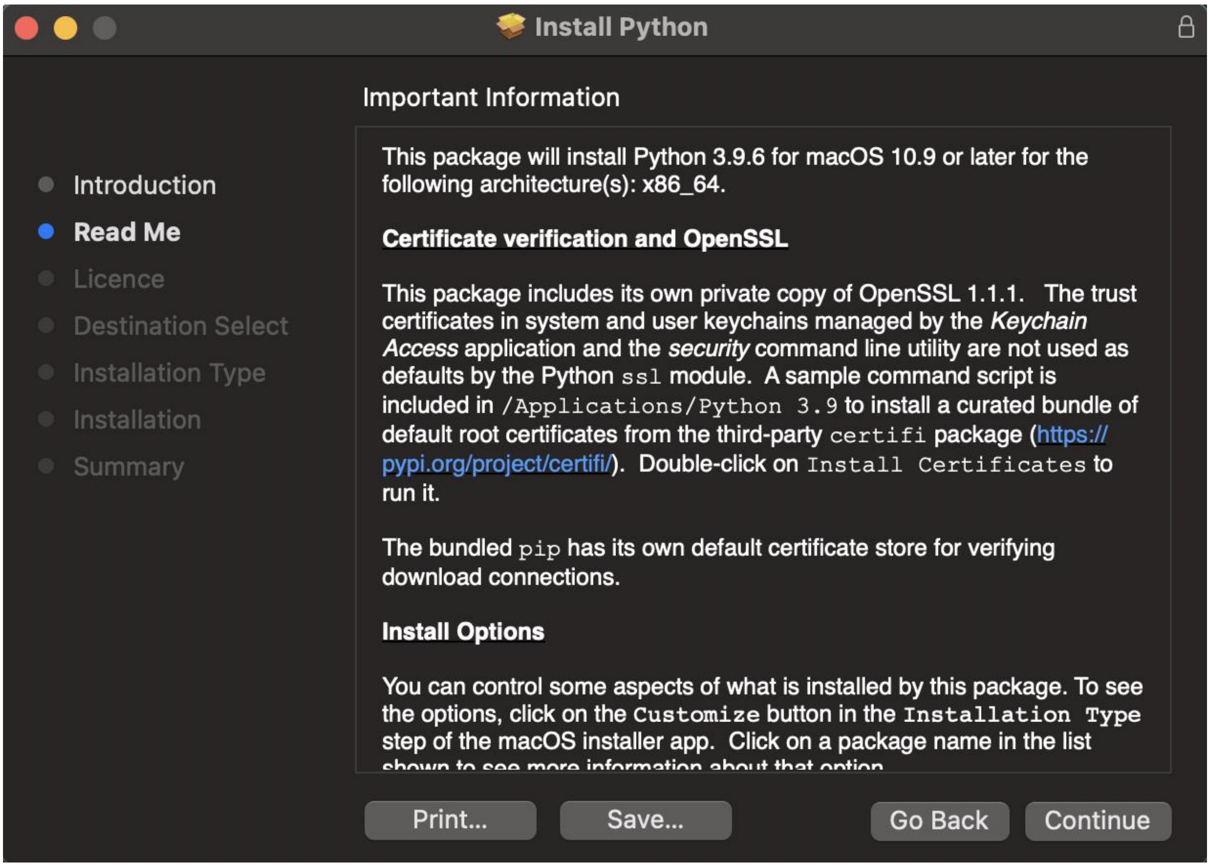
- Anaconda Navigator
- Python
- Jupyter Notebook

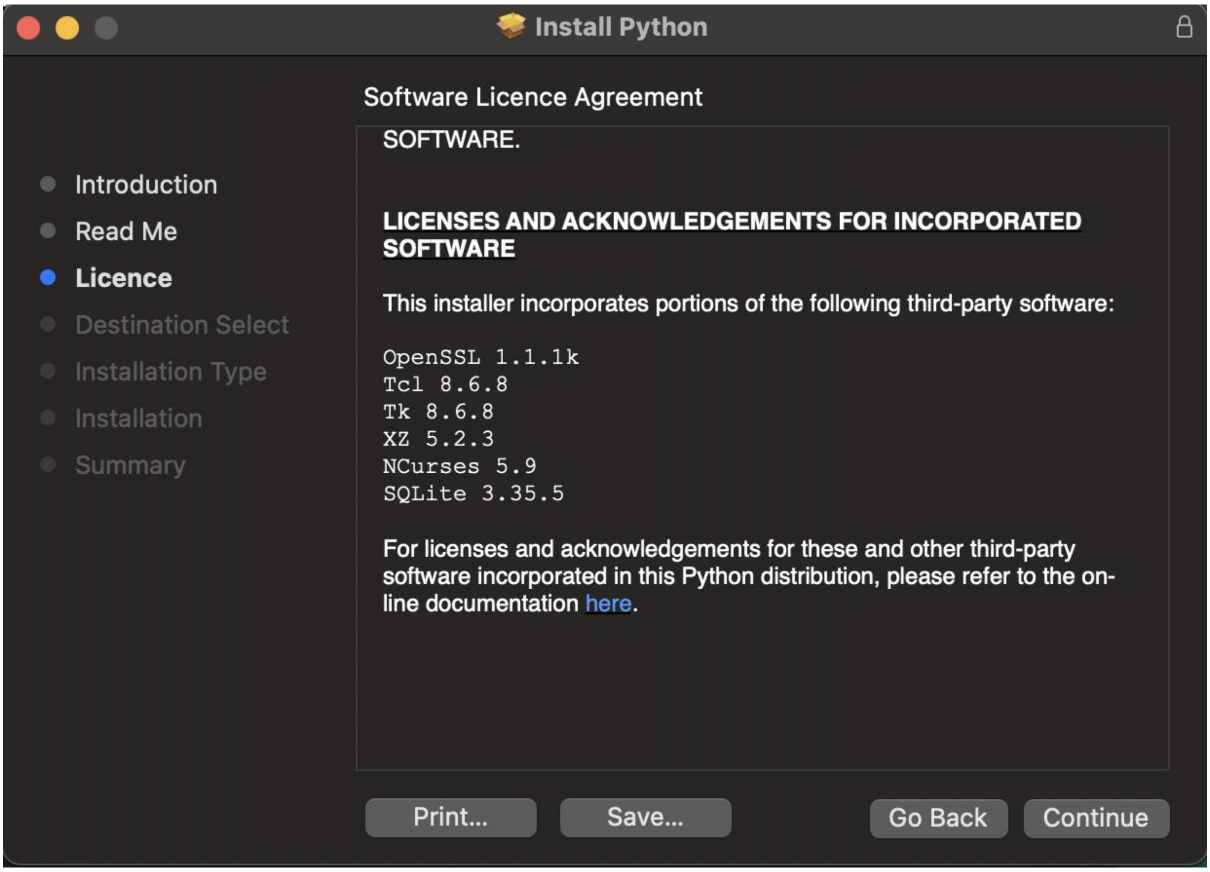
4 Software Installation

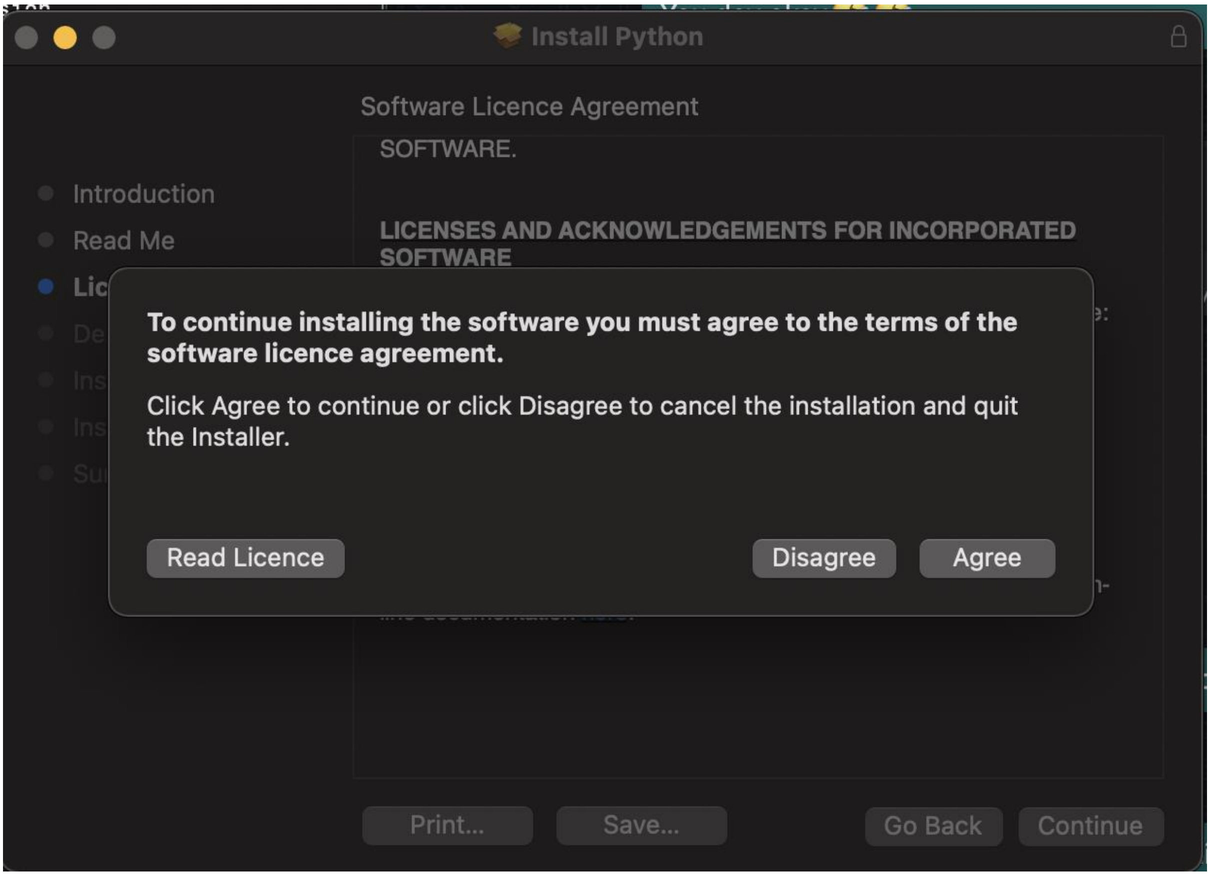
This is the step-by-step process of the implementation phase

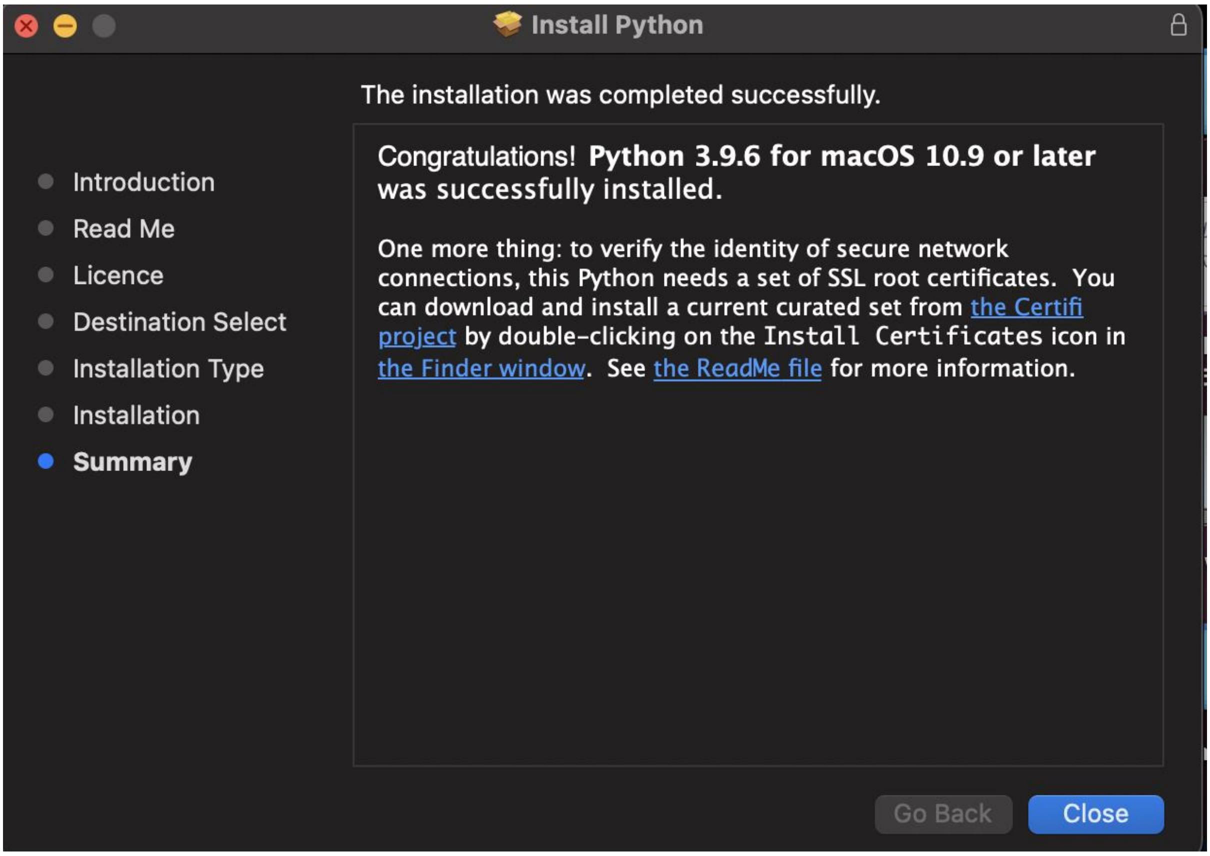
Download and install python 3.9.6 from (https://www.python.org/downloads/macos/)


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


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

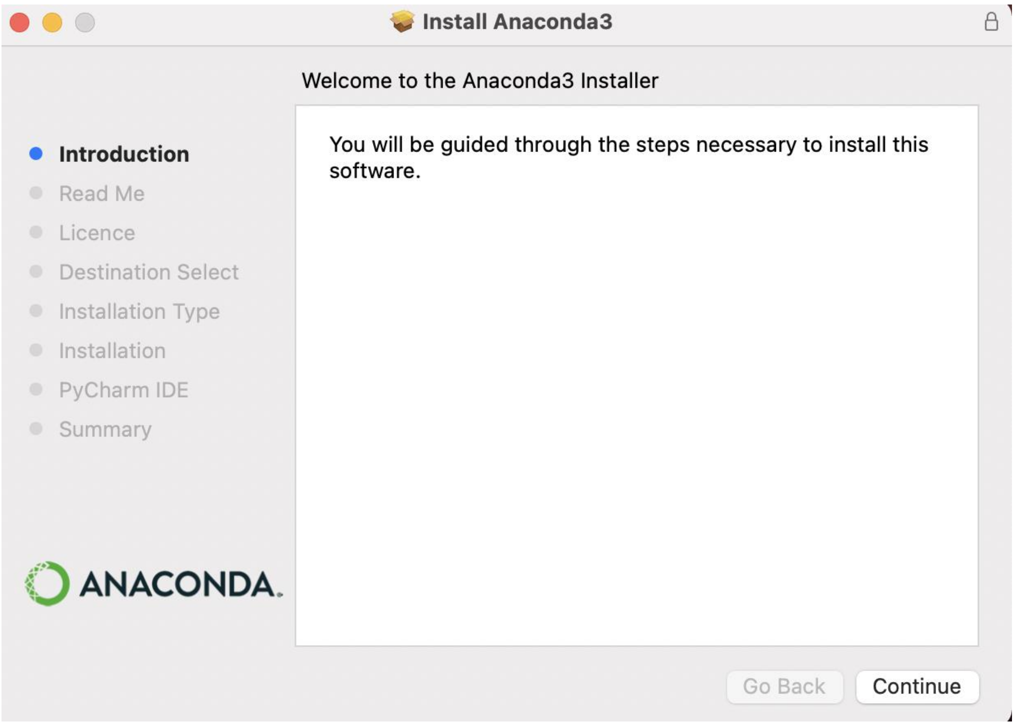

Install Python
Software Licence Agreement
Introduction
Read Me
Licence
Destination Select
Installation Type
Installation
Summary
SOFTWARE.
LICENSES AND ACKNOWLEDGEMENTS FOR INCORPORATED SOFTWARE
This installer incorporates portions of the following third-party software:
OpenSSL 1.1.1k
Tcl 8.6.8
Tk 8.6.8
XZ 5.2.3
NCurses 5.9
SQLite 3.35.5
For licenses and acknowledgements for these and other third-party software incorporated in this Python distribution, please refer to the on-line documentation here.
Print...
Save...
Go Back
Continue

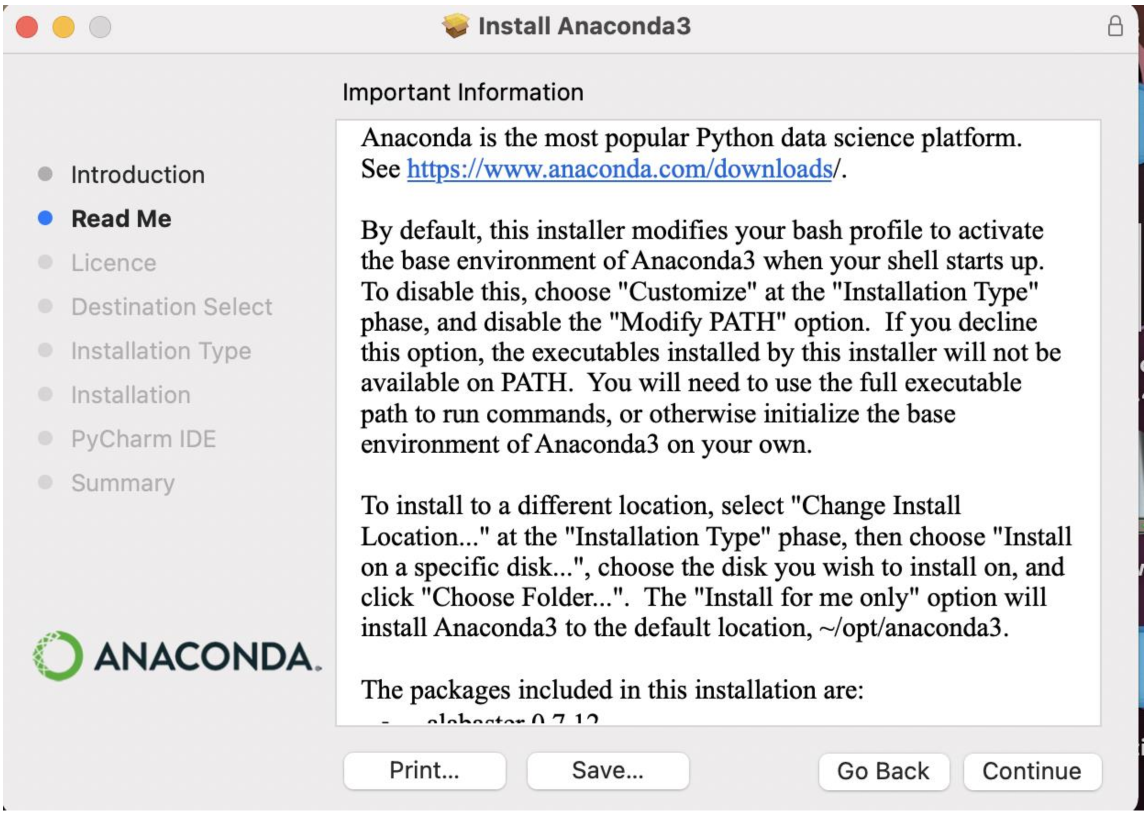

Install Python
Software Licence Agreement
SOFTWARE.
LICENSES AND ACKNOWLEDGEMENTS FOR INCORPORATED SOFTWARE
Introduction
Read Me
To continue installing the software you must agree to the terms of the software licence agreement.
Click Agree to continue or click Disagree to cancel the installation and quit the Installer.
Read Licence
Disagree
Agree
Print...
Save...
Go Back
Continue

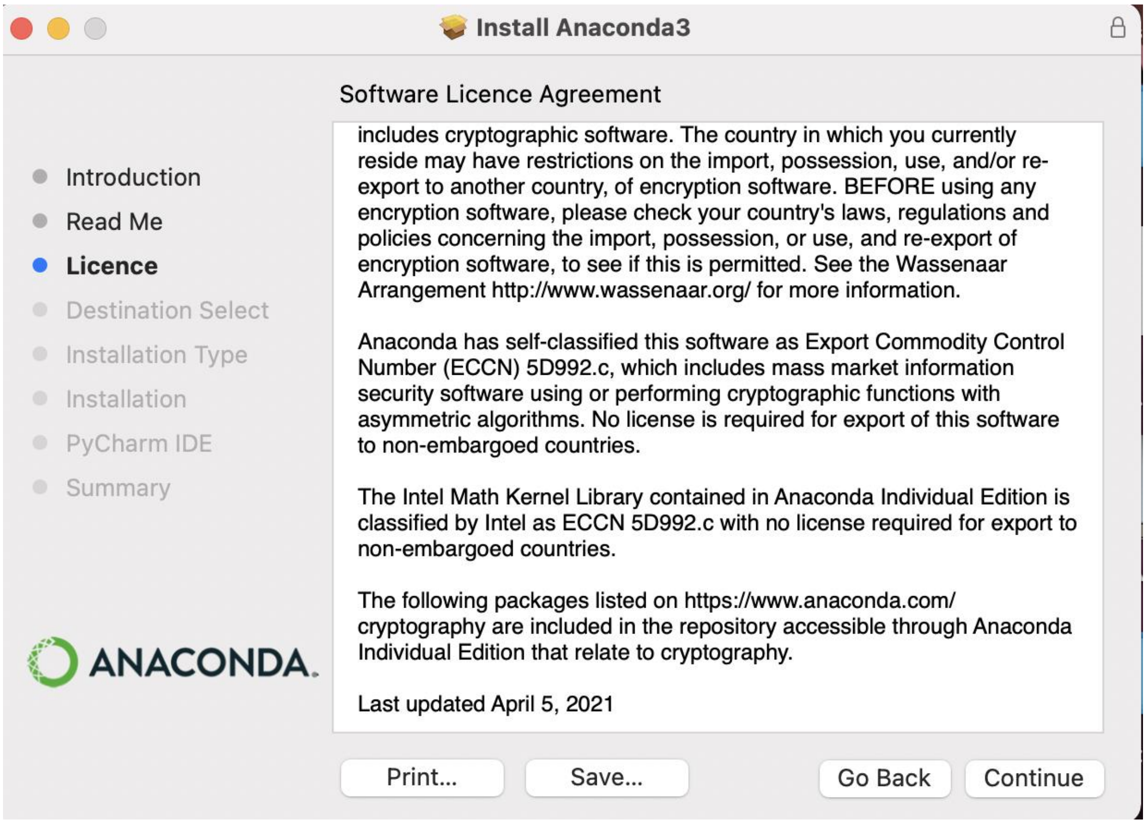


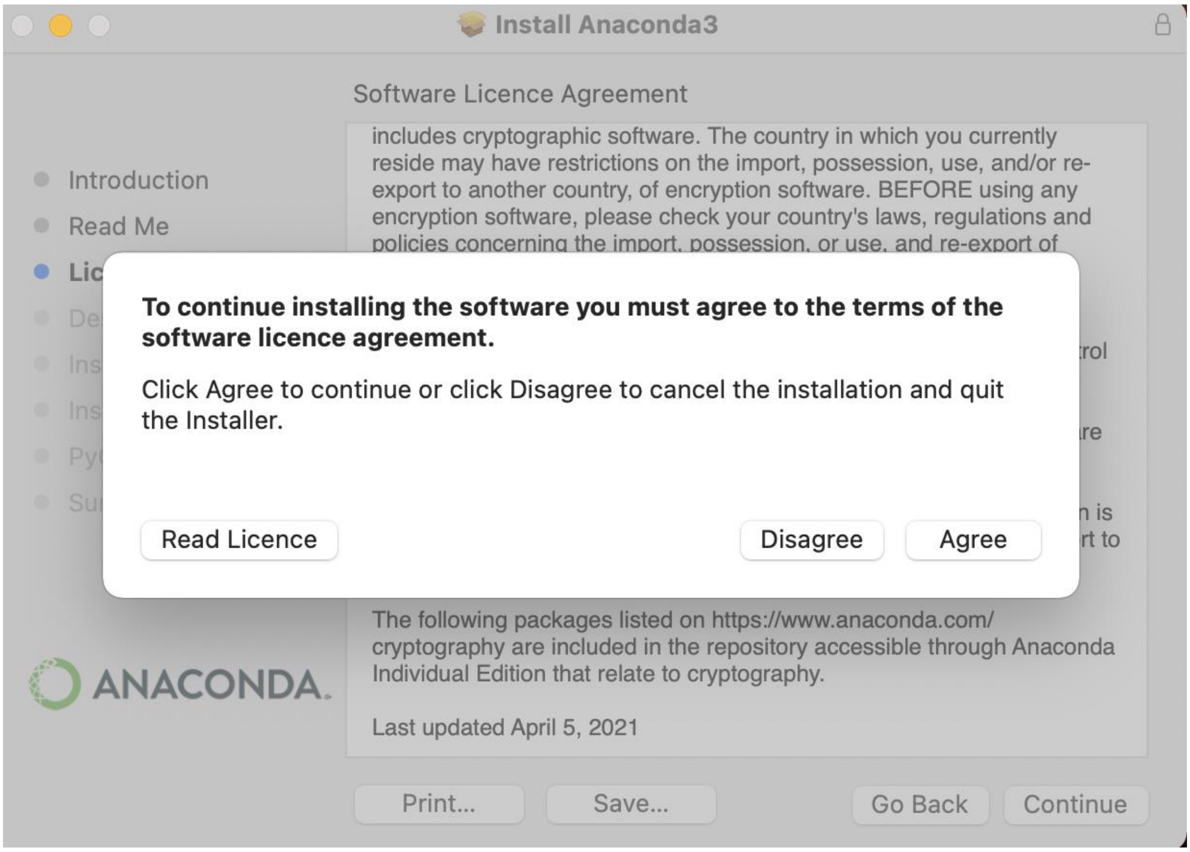
Download and install Anaconda from

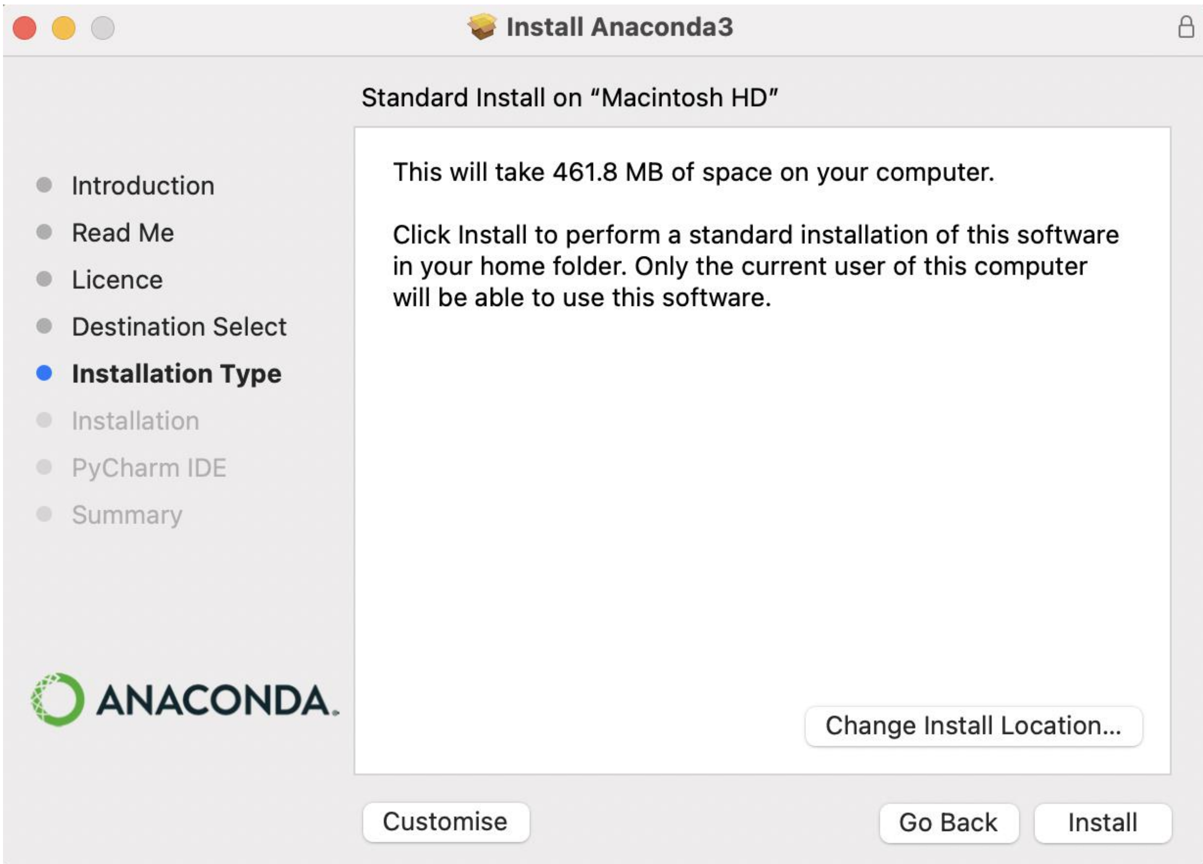
(https://www.anaconda.com/products/individual#macos)

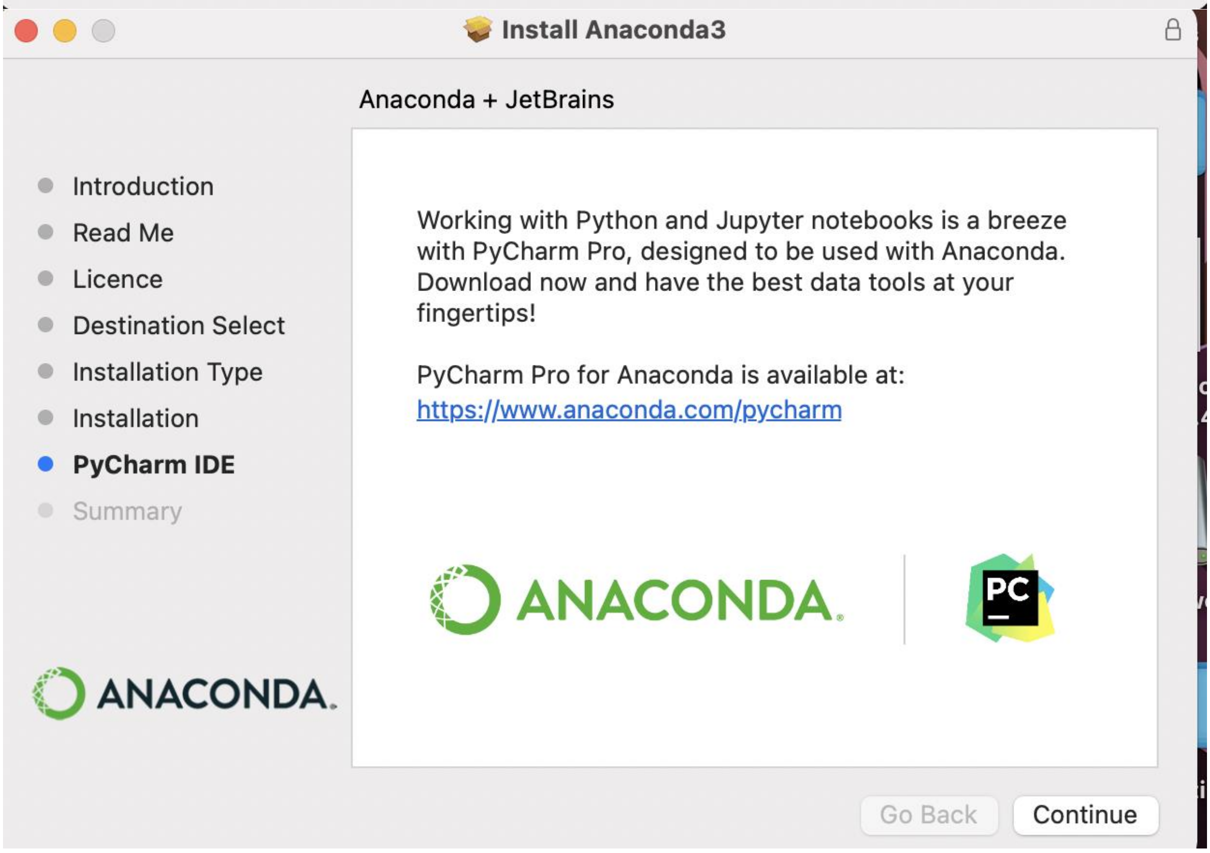





Install Anaconda3
Software Licence Agreement
Introduction
Read Me
Licence
Destination Select
Installation Type
Installation
PyCharm IDE
Summary
ANACONDA.
includes cryptographic software. The country in which you currently reside may have restrictions on the import, possession, use, and/or re-export to another country, of encryption software. BEFORE using any encryption software, please check your country's laws, regulations and policies concerning the import, possession, or use, and re-export of encryption software, to see if this is permitted. See the Wassenaar Arrangement http://www.wassenaar.org/ for more information.
Anaconda has self-classified this software as Export Commodity Control Number (ECCN) 5D992.c, which includes mass market information security software using or performing cryptographic functions with asymmetric algorithms. No license is required for export of this software to non-embargoed countries.
The Intel Math Kernel Library contained in Anaconda Individual Edition is classified by Intel as ECCN 5D992.c with no license required for export to non-embargoed countries.
The following packages listed on https://www.anaconda.com/cryptography are included in the repository accessible through Anaconda Individual Edition that relate to cryptography.
Last updated April 5, 2021
Print...
Save...
Go Back
Continue


Install Anaconda3
Software Licence Agreement
To continue installing the software you must agree to the terms of the software licence agreement.
Click Agree to continue or click Disagree to cancel the installation and quit the Installer.
Read Licence
Disagree
Agree
The following packages listed on https://www.anaconda.com/cryptography are included in the repository accessible through Anaconda Individual Edition that relate to cryptography.
Last updated April 5, 2021
Print...
Save...
Go Back
Continue


Install Anaconda3
Standard Install on "Macintosh HD"
Introduction
Read Me
Licence
Destination Select
Installation Type
Installation
PyCharm IDE
Summary
ANACONDA.
This will take 461.8 MB of space on your computer.
Click Install to perform a standard installation of this software in your home folder. Only the current user of this computer will be able to use this software.
Change Install Location...
Customise
Go Back
Install


Install Anaconda3
Anaconda + JetBrains
Introduction
Read Me
Licence
Destination Select
Installation Type
Installation
PyCharm IDE
Summary
ANACONDA.
Working with Python and Jupyter notebooks is a breeze with PyCharm Pro, designed to be used with Anaconda. Download now and have the best data tools at your fingertips!
PyCharm Pro for Anaconda is available at:
https://www.anaconda.com/pycharm
ANACONDA.
PC
Go Back
Continue

5 Implementation

The packages and libraries used are:

- NumPy
- Pandas
- Matplotlib
- Seaborn
- Sklearn

The following steps were taken

1. Importing Python Libraries which was used to perform all the machine learning algorithm analysis, visualization and modelling.

```
import pandas as pd
import numpy as np
import seaborn as sns
from array import *
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from itertools import cycle
from matplotlib import cm
from scipy.stats import norm
from sklearn.metrics import roc_curve, auc
from prettytable import PrettyTable

from keras.models import Model
from keras.layers import LSTM, Activation, Input, Embedding
from keras.optimizers import RMSprop, Adam
from keras.layers import Activation
from keras.models import Sequential
from keras.layers import Dense, Conv1D, Flatten, Dropout

from sklearn.preprocessing import StandardScaler
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from sklearn.metrics import confusion_matrix

from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import f_classif
from sklearn.feature_selection import mutual_info_classif
from sklearn.svm import LinearSVC
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import ExtraTreesClassifier

import warnings
warnings.filterwarnings("ignore")
```

Figure : Importing Libraries and packages

2. Loading the data

```python
data = pd.read_csv(r"creditcard.csv")
```

Figure : Importing Dataset

3. Data analysis, visualization and preparation

5 top records of data

```python
data.head(5)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	A
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	

5 rows × 31 columns

5 last records of data

```python
data.tail(5)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348	1.436807	0.250034	0.943651	0.
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255	0.068472	-0.
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371	0.004455	-0.
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668	0.108821	0.
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267	-0.002415	0.

5 rows × 31 columns

Coloumns/features in data

```
data.columns
```

```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

Length of data

```
print('lenght of data is', len(data))
```

```
lenght of data is 284807
```

Shape of data

```
data.shape
```

```
(284807, 31)
```

Checking duplicate data

```
current=len(data)
print('Rows of data before Delecting ', current)
```

```
Rows of data before Delecting  284807
```

```
data=data.drop_duplicates()
```

```
now=len(data)
print('Rows of data before Delecting ', now)
```

```
Rows of data before Delecting  283726
```

```
diff=current-now
print('Duplicated rows are ', diff)
```

```
Duplicated rows are  1081
```

4. Labelling fraudulent and legitimate card transaction

Distribution of Time and Amount

```
fig, ax = plt.subplots(1, 2, figsize=(18,4))
amount_val = data['Amount'].values
time_val = data['Time'].values
sns.distplot(amount_val, ax=ax[0], color='r')
ax[0].set_title('Distribution of Transaction Amount', fontsize=14)
ax[0].set_xlim([min(amount_val), max(amount_val)])
sns.distplot(time_val, ax=ax[1], color='b')
ax[1].set_title('Distribution of Transaction Time', fontsize=14)
ax[1].set_xlim([min(time_val), max(time_val)])
plt.show()
```

Under Sampling with random sample and getting the class non-fruad 0 same as fruad 1

```
data = data.sample(frac=1)
f_data = data.loc[data['Class'] == 1]
non_f_data = data.loc[data['Class'] == 0][:473]
normal_d_data = pd.concat([f_data, non_f_data])
new_data = normal_d_data.sample(frac=1, random_state=42)
```

Distribution of Class Now

```
sns.countplot(data= new_data, x = "Class")
plt.title('Class Distributions \n (0: No Fraud || 1: Fraud)', fontsize=14)
```

Correlation of data before undersampling and after undersampling

```
f, (ax1, ax2) = plt.subplots(2, 1, figsize=(24,20))
corr = data.corr()
sns.heatmap(corr, cmap='coolwarm_r', annot_kws={'size':20}, ax=ax1)
ax1.set_title("Before undersampling Correlation Matrix", fontsize=14)

sub_sample_corr = new_data.corr()
sns.heatmap(sub_sample_corr, cmap='coolwarm_r', annot_kws={'size':20}, ax=ax2)
ax2.set_title('After undersampling Correlation Matrix', fontsize=14)
plt.show()
```

5. Feature selection/extraction

Splitting the input features and output/label separately

```
input_features=new_data.drop('Class',axis=1)
target=new_data['Class']
```

```
input_features
```

Removing the fatures with low Variance

```
def get_low_var_cols(df, thres=80):
    low_var_cols = []
    for col in df.columns:
        percent_count = df[col].value_counts()/len(df)*100
        if percent_count.max() > thres:
            low_var_cols.append(col)
    return low_var_cols
```

```
# Identify columns that has same data across 90% of the rows
variance_thres = 80
low_var_cols = get_low_var_cols(X, variance_thres)
print('{} \n\n are columns with same data across 80% of the rows'.format(
    str(low_var_cols)[1:-1]))
```

```
 are columns with same data across 80% of the rows
```

```
# Drop the low variance columns
drop_cols = low_var_cols
X = X.drop(drop_cols, axis=1)
```

```
X.head()
```

Univariate feature selection

- using mutual_info_classif technique to extrcat the most useful features from the data

```
MiC_M= SelectKBest(mutual_info_classif, k=20)
MiC_M.fit(X, y)
MiC_features = MiC_M.transform(X)
```

```
feature_names = list(X.columns[MiC_M.get_support(indices=True)])
MiC_features=pd.DataFrame(MiC_features)
MiC_features.columns=feature_names
MiC_features.head()
```

Univariate feature selection

- using f_classif technique to extrcat the most useful features from the data

```
FCF_M= SelectKBest(f_classif, k=20)
FCF_M.fit(X, y)
FCF_features = FCF_M.transform(X)
```

```
feature_names = list(X.columns[FCF_M.get_support(indices=True)])
FCF_features=pd.DataFrame(FCF_features)
FCF_features.columns=feature_names
FCF_features.head()
```

Feature selection using SelectFromModel

- L1-based feature selection

```
#by using the SVM L1 based features, we selected most useful features
L_svc = LinearSVC(C=0.01, penalty="l1", dual=False).fit(X, y)
L_svc = SelectFromModel(L_svc, prefit=True)
L_svc_Features = L_svc.transform(X)
```

```
feature_names = list(X.columns[L_svc.get_support(indices=True)])
L_svc_Features=pd.DataFrame(L_svc_Features)
L_svc_Features.columns=feature_names
L_svc_Features.head()
```

Feature selection using SelectFromModel

- Tree-based feature selection

```
ETC_M = ExtraTreesClassifier(n_estimators=40)
ETC_M.fit(X, y)
ETC_M = SelectFromModel(ETC_M, prefit=True)
ETC_M_Features = ETC_M.transform(X)
```

```
feature_names = list(X.columns[ETC_M.get_support(indices=True)])
ETC_M_Features=pd.DataFrame(ETC_M_Features)
ETC_M_Features.columns=feature_names
ETC_M_Features.head()
```

Hybrid Feature selection

```
Hybrid_Features=pd.concat([MiC_features, FCF_features, L_svc_Features, ETC_M_Features], axis=1)
#drop all those features
Hybrid_Features = Hybrid_Features.loc[:,~Hybrid_Features.columns.duplicated()]
```

```
Hybrid_Features.shape[1]  #Features count
```

Spliting Dataset into 70% Training and 30% Testing

```
X_train, X_test, y_train, y_test = train_test_split(Hybrid_Features, target, test_size=0.30)
```

```
X_train.shape
```

```
(662, 24)
```

```
y_train.shape
```

```
(662,)
```

```
X_train.shape
```

```
(662, 24)
```

```
y_test.shape
```

```
(284,)
```

6. Machine learning Algorithm

Random Forest Model

```
RF=RandomForestClassifier(n_estimators=300, random_state=52)
RF= RF.fit(X_train , y_train)
RF
```

```
RandomForestClassifier(n_estimators=300, random_state=52)
```

Accuracy

```
y_pred = RF.predict(X_test)
RF_1=RF.score(X_test, y_test)
print('Accuracy= {:.3f}'.format(RF.score(X_test, y_test)))
```

```
Accuracy= 0.954
```

Precision

```
print('Precision',round(f1_score(y_test, y_pred),2),'%')
```

```
Precision 0.95 %
```

Recall

```
print('Recall',round(recall_score(y_test, y_pred),2),'%')
```

```
Recall 0.92 %
```

F1

```
rf_f1=round(f1_score(y_test, y_pred),3)
print('F1',round(f1_score(y_test, y_pred),2),'%')
```

```
F1 0.95 %
```

AUC-ROC curve

```
y_pred1 = RF.predict_proba(X_test)
fpr = {}
tpr = {}
n_classes = 2
roc_auc = {}
for i in range(n_classes):
    fpr[i], tpr[i], roc_auc[i] = roc_curve(y_test, y_pred1[:,i], pos_label=i)
    roc_auc[i] = auc(fpr[i], tpr[i])
colors = cycle(['blue', 'red'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=1.5,
             label='AUC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=1.5)
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('AUC of Multi Class')
plt.legend(loc="lower right")
plt.show()
```

Logistic Regression Model

]:

```
LR=LogisticRegression()
LR= LR.fit(X_train , y_train)
LR
```

]:

```
LogisticRegression()
```

Accuracy

]:

```
y_pred = LR.predict(X_test)
LR_1=LR.score(X_test, y_test)
print('Accuracy= {:.3f}'.format(RF.score(X_test, y_test)))
```

```
Accuracy= 0.954
```

Precision

]:

```
print('Precision',round(f1_score(y_test, y_pred),2),'%')
```

```
Precision 0.93 %
```

Recall

]:

```
print('Recall',round(recall_score(y_test, y_pred),2),'%')
```

```
Recall 0.91 %
```

F1

```
lr_f1=round(f1_score(y_test, y_pred),3)
print('F1',round(f1_score(y_test, y_pred),2),'%')
```

```
F1 0.93 %
```

AUC-ROC curve

```
y_pred1 = LR.predict_proba(X_test)
fpr = {}
tpr = {}
n_classes = 2
roc_auc = {}
for i in range(n_classes):
    fpr[i], tpr[i], roc_auc[i] = roc_curve(y_test, y_pred1[:,i], pos_label=i)
    roc_auc[i] = auc(fpr[i], tpr[i])
colors = cycle(['blue', 'red'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=1.5,
             label='AUC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=1.5)
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('AUC of Multi Class')
plt.legend(loc="lower right")
plt.show()
```

XGBoost Model

```
XGB=xgb.XGBClassifier()
XGB= XGB.fit(X_train , y_train)
XGB
```

```
[15:00:52] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=4, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

Accuracy

```
y_pred = XGB.predict(X_test)
XGB_1=XGB.score(X_test, y_test)
print('Accuracy= {:.3f}'.format(XGB.score(X_test, y_test)))
```

```
Accuracy= 0.937
```

Precision

```
print('Precision',round(f1_score(y_test, y_pred),2),'%')
```

```
Precision 0.94 %
```

Recall

```
print('Recall',round(recall_score(y_test, y_pred),2),'%')
```

```
Recall 0.92 %
```

F1

```
xgb_f1=round(f1_score(y_test, y_pred),3)
print('F1',round(f1_score(y_test, y_pred),2),'%')
```

```
F1 0.94 %
```

AUC-ROC curve

```
y_pred1 = XGB.predict_proba(X_test)
fpr = {}
tpr = {}
n_classes = 2
roc_auc = {}
for i in range(n_classes):
    fpr[i], tpr[i], roc_auc[i] = roc_curve(y_test, y_pred1[:,i], pos_label=i)
    roc_auc[i] = auc(fpr[i], tpr[i])
colors = cycle(['blue', 'red'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=1.5,
             label='AUC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=1.5)
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('AUC of Multi Class')
plt.legend(loc="lower right")
plt.show()
```

Preparing the dataset for ANN model

```
data = pd.read_csv("creditcard.csv")
data.drop("Time", axis=1, inplace=True)
limit = int(0.9*len(data))
train = data.loc[:limit]
val_test = data.loc[limit:]
val_test.reset_index(drop=True, inplace=True)
val_test_limit = int(0.5*len(val_test))
val = val_test.loc[:val_test_limit]
test = val_test.loc[val_test_limit:]

train_positive = train[train["Class"] == 1]
train_positive = pd.concat([train_positive] * int(len(train) / len(train_positive)), ignore_index=True)
noise = np.random.uniform(0.9, 1.1, train_positive.shape)
train_positive = train_positive.multiply(noise)
train_positive["Class"] = 1
train_extended = train.append(train_positive, ignore_index=True)
train_shuffled = train_extended.sample(frac=1, random_state=0).reset_index(drop=True)

X_train = train_shuffled.drop(labels=["Class"], axis=1)
Y_train = train_shuffled["Class"]
X_val = val.drop(labels=["Class"], axis=1)
Y_val = val["Class"]
X_test = test.drop(labels=["Class"], axis=1)
Y_test = test["Class"]

scaler = StandardScaler()
X_train[X_train.columns] = scaler.fit_transform(X_train)
X_val[X_val.columns] = scaler.transform(X_val)
X_test[X_test.columns] = scaler.transform(X_test)
```

ANN model preparation

ANN Model

```
def ANN_model():
    model = Sequential()
    model.add(Dense(64, activation="relu", input_dim=(X_train.shape[1])))
    model.add(Dense(32, activation="relu"))
    model.add(Dense(16, activation="relu"))
    model.add(Dense(8, activation="relu"))
    model.add(Dense(4, activation="relu"))
    model.add(Dense(2, activation="relu"))
    model.add(Dense(1, activation="sigmoid"))
    return model
```

Accuracy

```
test_results = model.evaluate(X_test, Y_test)
y_pred = model.predict_classes(X_test)
print("Accuracy= {}.".format(test_results[1]))
```

```
446/446 [==============================] - 0s 841us/step - loss: 0.0568 - accuracy: 0.9992
WARNING:tensorflow:From <ipython-input-83-d48417203ab5>:2: Sequential.predict_classes (from tensorflow.python.keras.engine.sequential) is deprecated and will be removed
after 2021-01-01.
Instructions for updating:
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses a `softmax` last-layer activation).* `(m
odel.predict(x) > 0.5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
Accuracy= 0.9991573691368103.
```

Precision

```
print('Precision',round(f1_score(Y_test, y_pred, average='micro'),3),'%')
```

```
Precision 0.999 %
```

Recall

```
print('Recall',round(recall_score(Y_test, y_pred, average='micro'),3),'%')
```

```
Recall 0.999 %
```

F1

```
ann_f1=round(f1_score(Y_test, y_pred, average='micro'),3)
print('F1',round(f1_score(Y_test, y_pred, average='micro'),3),'%')
```

```
F1 0.999 %
```

7. Result and Evaluation

Comparison of all Models

]:

```
x = PrettyTable()
print('\n')
print("Comparison of ANN model with")
print("above tradional ML algorithms on F1 score as required")
x.field_names = ["Model", "Accuracy"]
x.add_row(["Random Forest Algorithm", rf_f1])
x.add_row(["Logistic Regression Algorithm",  lr_f1])
x.add_row(["XGboost Algorithm", xgb_f1])
x.add_row(["ANN Model", ann_f1])
print(x)
print('\n')
```