National College of Ireland

Configuration Manual

MSc Research Project
Fintech

Jamiu Olalekan Oni
Student ID: x19111240

School of Computing
National College of Ireland

Supervisor: Victor Del Rosa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Jamiu Olalekan Oni
Student ID:	X19111240
Programme:	Fintech
Year:	2020
Module:	MSc Research Project
Supervisor:	Victor Del Rosa
Submission Due Date:	17 august 2020
Project Title:	Exploratory analysis of bank marketing campaign using machine; logistic regression, support vector machine and k-nearest neighbour
Word Count:	1004
Page Count:	11

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: .....

17 august 2020

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

Jamiu Olalekan Oni
X19111240

1 Introduction

This chapter provides an overview of how the analysis was made. This means that this report would demonstrate anything needed to replicate this work 'Exploratory Analysis of bank marketing campaign using machine learning; logistic regression, support vector machine and k-nearest neighbour'. Two experiment were carried out for this thesis, the first was implemented using the variables in the original dataset, and the second experiment was implemented using the six important factors gotten from the first experiment. The sections in this work shows how the study was carried out and implemented.

2 Details of Device Specification

- **System used**: Aspire E1-531
- **Ram**: 6 GB (5.82 GB usable)
- **Processor:** intel (R) Celeron (R)1005M @1.90GHz
- **System type**: 64-bit operation system, x64-based processor
- **Speed**: 1.90 GHz

3 Used Software

- Microsoft word
- Anaconda python
- Python
- Jupyter notebook

4 Installed Libraries

Numpy
Pandas
Matplotlib
Sklearn.model_selection
Sklearn.utils
Sklearn.preprocessing
Sklearn.linear_model
Sklearn.metrices
Sklearn.logistic
Sklearn.svm
Sklearn.neighbors

5 Used codes for analysis

```
#Importing the necessary libraries
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams['figure.figsize']=8,4
import warnings
warnings.filterwarnings ('ignore')

#Reading csv file
BankAnal=pd.read_csv('C:\\Users\\OLA\\Desktop\\Marketing Data\\bank.csv')

#Getting top 5 record
BankAnal.head()

#Getting the columns
BankAnal.columns

#Renaming Y column to signed
BankAnal.rename(columns={"y": "signed"}, inplace=True)

# simply change yes/no to 1/0 for RainToday and RainTomorrow
BankAnal['signed'].replace({'no': 0, 'yes': 1},inplace = True)

BankAnal.info()

 #Checking for null values
BankAnal.isnull().sum()

#EDA of Marital Status
value_counts = BankAnal['marital'].value_counts()
sns.set_style("darkgrid")
value_counts.plot.bar(title = 'Marital Status value counts',)
plt.show()

BankAnal.marital.value_counts(normalize=True)

value_counts = BankAnal['signed'].value_counts()
sns.set_style("darkgrid")
value_counts.plot.bar(title = 'Subscribe value counts',)
plt.show()

BankAnal.signed.value_counts(normalize=True)

BankAnal_count =
pd.DataFrame(BankAnal.groupby('marital').size().sort_values(ascending=False).rename('cou
nts').reset_index())

BankAnal_count

#box plot
```

```
viz2=sns.boxplot(data=BankAnal, x='signed', y='age',)
sns.set_style("darkgrid")

sns.barplot(BankAnal.signed,BankAnal.duration)
sns.set_style("darkgrid")
plt.show()

j_df = pd.DataFrame()

j_df[1] = BankAnal[BankAnal['signed'] == 1]['job'].value_counts()
j_df[0] = BankAnal[BankAnal['signed'] == 0]['job'].value_counts()
sns.set_style("darkgrid")
j_df.plot.bar(title = 'Type of Job and deposit')
plt.show()

j_df = pd.DataFrame()

j_df[1] = BankAnal[BankAnal['signed'] == 1]['contact'].value_counts()
j_df[0] = BankAnal[BankAnal['signed'] == 0]['contact'].value_counts()

j_df.plot.bar(title = 'Type of Contact and Subscription')

plt.show()

BankAnal.head()

# Convert Categorical Attributes to Numerical Values
BankAnal['job'] = pd.factorize(BankAnal["job"])[0]
BankAnal['marital'] = pd.factorize(BankAnal["marital"])[0]
BankAnal['education'] = pd.factorize(BankAnal["education"])[0]
BankAnal['default '] = pd.factorize(BankAnal["default"])[0]
BankAnal['housing'] = pd.factorize(BankAnal["housing"])[0]
BankAnal['month'] = pd.factorize(BankAnal["month"])[0]
BankAnal['contact'] = pd.factorize(BankAnal["contact"])[0]
BankAnal['day_of_week'] = pd.factorize(BankAnal["day_of_week"])[0]
BankAnal['poutcome '] = pd.factorize(BankAnal["poutcome"])[0]
BankAnal['loan'] = pd.factorize(BankAnal["loan"])[0]

BankAnal.drop(['default','poutcome'], axis=1,inplace=True)

BankAnal

BankAnal_Count=
pd.DataFrame(BankAnal.groupby('signed').size().sort_values(ascending=False).
                    rename('counts').reset_index())

BankAnal_Count

from sklearn.utils import resample
df_0 =BankAnal[BankAnal.signed ==0]
```

```
df_1 =BankAnal[BankAnal.signed ==1]
df_Target=resample(df_1, replace = True, n_samples= 36548, random_state=0)
df_Target=pd.concat([df_Target,df_0])
df_Target.signed.value_counts()

 #Splitting the data into training and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_Target.drop(['signed'] ,axis=1),
                        df_Target['signed'], test_size=0.3, random_state=0)

from sklearn.metrics import precision_score, recall_score, confusion_matrix,accuracy_score,
\
                        f1_score, classification_report,roc_auc_score


  #Standardizing the data for them to be on the same scale

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train_scaled = sc_X.fit_transform(X_train)
X_test_scaled= sc_X.transform(X_test)

#Building the Machine Learning Model
   #Model 1 Logistic Regression
from sklearn.linear_model import LogisticRegression
   #Fitting the logistic Regression into the training set
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train_scaled, y_train)

#Predicting the test result
y_pred = classifier.predict(X_test_scaled)
ac_sc = accuracy_score(y_test, y_pred)
rc_sc = recall_score(y_test, y_pred)
pr_sc = precision_score(y_test, y_pred)
f1_sc = f1_score(y_test,y_pred)
auc_sc = roc_auc_score(y_test,y_pred)
clas = classification_report(y_test, y_pred)
confusion_m = confusion_matrix(y_test, y_pred)
print("========== Result for Logistic Regression ==========")
print(f"'Accuracy    : {ac_sc:.3f}
Precision   : {pr_sc:.3f}
F1 Score    : {f1_sc:.3f}
Recall  : {rc_sc:.3f}
AUC     : {auc_sc:.3f}
'")
print('Classification Report\n', clas)
print("Confusion Matrix: ")
print(confusion_m)

#Model 2 K Nearest Neighbor
```

```
from sklearn.neighbors import KNeighborsClassifier
#Fitting the KNN into the training set
classifier = KNeighborsClassifier(n_jobs=-1)
classifier.fit(X_train_scaled, y_train)

#Predicting the test result
y_pred2 = classifier.predict(X_test_scaled)
ac_sc = accuracy_score(y_test, y_pred2)
rc_sc = recall_score(y_test, y_pred2)
pr_sc = precision_score(y_test, y_pred2)
f1_sc = f1_score(y_test,y_pred2)
auc_sc = roc_auc_score(y_test,y_pred2)
clas = classification_report(y_test, y_pred2)
confusion_m = confusion_matrix(y_test, y_pred2)
print("========== K Nearest Neighbor ==========")
print(f"'Accuracy    : {ac_sc:.3f}
Precision   : {pr_sc:.3f}
F1 Score    : {f1_sc:.3f}
Recall  : {rc_sc:.3f}
AUC     : {auc_sc:.3f}
'")
print('Classification Report\n', clas)
print(confusion_m)

from sklearn.svm import SVC
classifier = SVC(kernel='linear')
classifier.fit(X_train_scaled, y_train)

#Predicting the test result
y_pred3 = classifier.predict(X_test_scaled)
ac_sc = accuracy_score(y_test, y_pred3)
rc_sc = recall_score(y_test, y_pred3)
pr_sc = precision_score(y_test, y_pred3)
f1_sc = f1_score(y_test,y_pred3)
auc_sc = roc_auc_score(y_test,y_pred3)
clas = classification_report(y_test, y_pred3)
confusion_m = confusion_matrix(y_test, y_pred3)

print("========== SVM ==========")
print(f"'Accuracy    : {ac_sc:.3f}
Precision   : {pr_sc:.3f}
F1 Score    : {f1_sc:.3f}
Recall  : {rc_sc:.3f}
AUC     : {auc_sc:.3f}
'")
print('Classification Report\n', clas)
print("Confusion Matrix: ")
print(confusion_m)

# Selecting best features by Correlation Matrix with Heatmap
```

```
import seaborn as sns
X_fs =  df_Target.drop(['signed'], axis=1)
Y_fs =  df_Target['signed']
corrmat =  df_Target.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap( df_Target[top_corr_features].corr(),annot=True,cmap=plt.cm.Reds)


df = pd.DataFrame(df_Target)
ImportantFeatures = df[[ "contact", "month", 'duration','previous',
                    'euribor3m','nr.employed',"signed"]].copy()
print(ImportantFeatures)

ImportantFeatures.shape

ImportantFeatures.isnull().sum()

 #Splitting the data into training and test set
from sklearn.model_selection import train_test_split
x_train, x_test, Y_train, Y_test = train_test_split(ImportantFeatures.drop(['signed'] ,axis=1),
                                ImportantFeatures['signed'], test_size=0.3, random_state=0)

#Standardizing the data for them to be on the same scale

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
x_train_scaled = sc_X.fit_transform(x_train)
x_test_scaled= sc_X.transform(x_test)

#Building the Machine Learning Model
   #Model 1 Logistic Regression
from sklearn.linear_model import LogisticRegression
   #Fitting the logistic Regression into the training set
classifier = LogisticRegression(random_state=0)
classifier.fit(x_train_scaled, Y_train)

#Predicting the test result
y_pred = classifier.predict(x_test_scaled)
ac_sc = accuracy_score(Y_test, y_pred)
rc_sc = recall_score(Y_test, y_pred)
pr_sc = precision_score(Y_test, y_pred)
f1_sc = f1_score(Y_test,y_pred)
auc_sc = roc_auc_score(Y_test,y_pred)
clas = classification_report(Y_test, y_pred)
confusion_m = confusion_matrix(Y_test, y_pred)

print("========== Result for Logistic Regression ==========")
print(f"'Accuracy    : {ac_sc:.3f}
```

```
Precision   : {pr_sc:.3f}
F1 Score    : {f1_sc:.3f}
Recall  : {rc_sc:.3f}
AUC     : {auc_sc:.3f}
'")
print('Classification Report\n', clas)
print("Confusion Matrix: ")
print(confusion_m)

#Model 2 K Nearest Neighbor
from sklearn.neighbors import KNeighborsClassifier
#Fitting the KNN into the training set
classifier = KNeighborsClassifier(n_jobs=-1)
classifier.fit(x_train_scaled, Y_train)

#Predicting the test result
y_pred2 = classifier.predict(x_test_scaled)
ac_sc = accuracy_score(Y_test, y_pred2)
rc_sc = recall_score(Y_test, y_pred2)
pr_sc = precision_score(Y_test, y_pred2)
f1_sc = f1_score(Y_test,y_pred2)
auc_sc = roc_auc_score(Y_test,y_pred2)
clas = classification_report(Y_test, y_pred2)
confusion_m = confusion_matrix(Y_test, y_pred2)
print("========== K Nearest Neighbor ==========")
print(f"'Accuracy    : {ac_sc:.3f}
Precision   : {pr_sc:.3f}
F1 Score    : {f1_sc:.3f}
Recall  : {rc_sc:.3f}
AUC     : {auc_sc:.3f}
'")
print('Classification Report\n', clas)
print("Confusion Matrix: ")
print(confusion_m)

from sklearn.svm import SVC
classifier = SVC(kernel='linear')
classifier.fit(x_train_scaled, y_train)

#Predicting the test result
y_pred3 = classifier.predict(x_test_scaled)
ac_sc = accuracy_score(Y_test, y_pred3)
rc_sc = recall_score(Y_test, y_pred3)
pr_sc = precision_score(Y_test, y_pred3)
f1_sc = f1_score(Y_test,y_pred3)
auc_sc = roc_auc_score(Y_test,y_pred3)
clas = classification_report(Y_test, y_pred3)
confusion_m = confusion_matrix(Y_test, y_pred3)
print("========== SVM ==========")
print(f"'Accuracy    : {ac_sc:.3f}
```

```
Precision   : {pr_sc:.3f}
F1 Score    : {f1_sc:.3f}
Recall  : {rc_sc:.3f}
AUC     : {auc_sc:.3f}
'")
print('Classification Report\n', clas)
print("Confusion Matrix: ")
print(confusion_m)
```