Python ~Data cleaning , modelling (Logistic regression)

Sahidtiasadason Chindu

ABSTRACT

In this report analysis is carried out on a marketing campaign dataset based on a case of a retailer company in computer accessories. The dataset consist of 19 variables and 1500 cases. The data will first be prepared by identifying missing values and errors in data entry. Then variables which are not related to the analysis will be eliminated and the some variable will be transformed according to the needs of the analysis. After preparing the data, Python programmes will be developed to analyse the summary statistics, correlation and Euclidean distance. Finally a logistic regression model will be built using Python and the model will be checked for adequacy.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. DATA UNDERSTANDING

This report focuses on the analysis of marketing campaign dataset based on a case of a retailer company in computer accessories. The dataset obtained is in 'csv' format, named 'Marketing Campaign data' and it contains a total of 1500 customer records. Each of the customer records consist of 19 variables, which includes socio-demographic and product ownership information. The target variable, AFFINITY_CARD is a binary variable.

A basic observation was carried out and it is observed that there are inconsistencies in the dataset as well as missing values. A table of variables with all issues and suitability of a variable was made from this basic observation. This table is labelled as Table 1.1 and is shown on the next page.

From Table 1.1 it can be concluded that the quality of the data is not perfect and that the dataset need to be cleaned and some of the variables need to be categorized. Furthermore, there are missing values which has to be dealt with.

Further analysis had to be carried out on some of the variables. The variable 'HOUSEHOLD_SIZE' has some data entry error. Instead of having an integer number for every case, there were some cases labelled as '4-May' and '6-Aug'. The first step was to analyse the excel file, after analysing the file it was discovered that even though it was displayed as 4-May and 6-Aug, it was actually input as 4/5/2018 and 6/8/2018. The excel file automatically changed this input to display as 4-May and 6-Aug. After checking all the inputs for this specific variable, it was observed that none of the cases had 4,5,6,7 and 8 but there were cases for 9+. After much investigation, it was assumed that 4/5/2018 should actually represent 4-5 and 6/8/2018 should actually represent 6-8. The other variable that required investigation was YRS_RESIDENCE. This variable has cases with'0' this could mean a data entry error, '0' could indicate a missing value' or it could just indicate a customer who has

been in that residence for less than a year. After much consideration, it was decided to assume that '0' indicated that the customer has been living in the residence for less than a year.

For the variable EDUCATION it was noted that there are many categories and some of these categories only contains a few cases. For better modelling some of the categories will be combined. Refer to Table 2.1 for the classification of all variables.

Variable name	Comments
CUST_ID	This is an ID variable and hence should not affect the analysis
CUST_GENDER	Variable transformation is needed (F-0, M-1)
AGE	Variable is suitable for analysis
CUST_MARITAL_STATUS	Variable is suitable for analysis
COUNTRY_NAME	Variable transformation is needed. Country name into ordinal numbers.
CUST_INCOME_LEVEL	Variable transformation is needed. 5 ordinal numbers
EDUCATION	Variable transformation is needed. Change to ordinal (***reduce categories for better modelling).***
OCCUPATION	Missing values indicated by '?'
HOUSEHOLD_SIZE	Variable transformation is needed. Change to ordinal. Error in data entry ***(investigate further)***
YRS_RESIDENCE	Minimum value for this variable is 0, this could be an error in entry or '0' indicates a missing value. ***(investigate further)***
AFFINITY_CARD	Variable is suitable for analysis. (Target variable)
BULK_PACK_DISKETTES	Variable is suitable for analysis
FLAT_PANEL_MONITOR	Variable is suitable for analysis
HOME_THEATER_PACKAGE	Variable is suitable for analysis
BOOKKEEPING_APPLICATION	Variable is suitable for analysis
PRINTER_SUPPLIES	Min and max value is 1, indicates that all 1500 cases have outcome 1, this variable will have no effect of the analysis. ***Eliminate***
Y_BOX_GAMES	Variable is suitable for analysis

OS_DOC_SET_KANJI	Only three outcomes of '1' and the other 1497 cases are 0. Not a reliable variable for the prediction. ***Eliminate***
COMMENTS	73 comments are missing. **Eliminate**

Table 1. 1 Variable analysis table

2. DATA PREPARATION

2.1 Overview

From the previous chapter it is clear that some of the variables have to be transformed into ordinal or nominal, some variables to be rejected .Table 2.1 shows a breakdown of rejected variables and transformed variables and their respective codes.

Variable name	Description	level	Code	Code description
CUST_ID	Unique ID	ID	-	-
CUST_GENDER	Sex of customer	Binary	0 1	Female Male
AGE	Age of customer	Interval	numerical	-
CUST_MARITAL_STATUS	Marital status of customer	Ordinal	1 2 3 4 5 6 7	Married NeverM Divorc. Separ. Widowed Mabsent Mar-AF
COUNTRY_NAME	Country of residence	Ordinal	1 2 3 4 5 6 7 8 9 10 11	United states of America Argentina Italy Brazil Germany Poland Canada United Kingdom Saudi Arabia Denmark Others

CUST_INCOME_LEVEL	Income of customer	Ordinal	1 2 3 4 5	<50000 50000-109999 110000-149999 150000-299999 >300000
EDUCATION	Education level of customer	Ordinal	1 2 3 4 5 	Presch 1st-4th 5th-6th 7th -8th 9th 10th 11th 12th HS-grad <Bach. Assoc-A Assoc-V Bach. Masters Profsc Phd
OCCUPATAION	Occupation of customer	Ordinal	1 2 3 4 5 6 7 8 9 10 11 12 13 14	Exec. Crafts Sales Cleric Prof. Other Machine Transp. Handler TechSup Farming Protec House-s Armed-F
HOUSEHOLD_SIZE	Household size	Ordinal	1 2 3	1 2 3

			4 5 6	4-5 6-8 9+
YRS _RESIDENCE	Duration at current residence	Interval	-	-
AFFINITY _CARD	Affinity card holder?	Binary	0 1	No Yes
BULK_PACK _DISKETTES	Bought bulk pack diskettes?	Binary	0 1	No Yes
FLAT_PANEL _MONITOR	Bought flat panel monitor?	Binary	0 1	No Yes
HOME_THEATER _PACKAGE	Bought Home theatre package?	Binary	0 1	No Yes
BOOKKEEPING _APPLICATION	Bought Bookkeeping application?	Binary	0 1	No Yes
PRINTER _SUPPLIES	Reject	Reject	-	-
Y_BOX_GAMES	Bought Y box games?	Binary	0 1	No Yes
OS_DOC_SET _KANJI	Reject	Reject	-	-
COMMENTS	Comments from customer	Reject	-	-

Table 2. 1 List of variable details

2.2 Data Cleaning

In this section the cases with missing value or with data entry error will be eliminated. From Table 1.1 it was observed that the variable 'OCCUPATION' has missing values which was recorded as '?'. This cannot be used in the analysis and hence these cases has to be deleted. The variable 'COMMENTS' have missing values, these cases will not be removed as it was decided that variable would be eliminated. The following Python script shown in Figure 2.1 is

used to remove the cases with 'missing' values. After running the below script, the total number of customer cases left are 1351.

```
#############################################################################################
import pandas as pd
import numpy as np
data= pd.read_csv('Marketing Campaign data.csv')
data

# obtain a summary statistics, from which the min value can be
#observed to check for missing values for certain variables.
data.describe()

# checking for empty entry in the dataset
data.isnull().sum()

# Deleting '?' in variable OCCUPATION and NAN values
data=data.replace({'?':np.nan}).dropna()

# altenative method to droping NAN values
data1 = data[~pd.isnull(data)]
#############################################################################################
```

```
[1351 rows x 19 columns]
```

Figure 2. 1 Removing cases with missing values (Python script)

2.3 Variable Reduction

Three variables were identified to be eliminated in Table 1.1. These variables are 'COMMENTS', 'OS_DOC_SET_KANJI', and 'PRINTER_SUPPLIES'. The variable 'COMMENTS' was eliminated as it requires a dedicated test mining tool. The Python script in Figure 2.2 is used to carry out this elimination process. After removing the variables, the dataset now contains a total of 16variables and 1351 cases.

```
#############################################################################################
# Dropping variables
data.drop(['PRINTER_SUPPLIES','COMMENTS','OS_DOC_SET_KANJI'],axis=1)

# alternative method
list_drop = ['PRINTER_SUPPLIES','COMMENTS','OS_DOC_SET_KANJI']
data.drop(list_drop, axis=1, inplace=True)
#############################################################################################
```

[1351 rows x 16 columns]

Figure 2. 2 Removing variables (Python script)

2.4 Variable Transformation

Multiple variables were identified to require transformation during the data understanding section. There variables will be transformed according the codes stated in Table 2.1. This section shows the Python script for each individual transformation that was carried out.

The first transformation was carried out in CUST_GENDER. Male and female was represented as M and F, this was replaced by 1 and 0 respectively. The Python script in Figure 2.3 shows how this was carried out along with the result of the transformation on the first 5 cases.

```
##########################################################################################
# changing customer gender into 1 and 0 for M and F respectively
gender = {'M': 1,'F': 0}
data.CUST_GENDER = [gender[item] for item in data.CUST_GENDER]
# alternative method
data1.CUST_GENDER[data.CUST_GENDER == 'M'] = 1
data1.CUST_GENDER[data.CUST_GENDER == 'F'] = 0
##########################################################################################
```

	CUST_ID	CUST_GENDER	AGE	CUST_MARITAL_STATUS	COUNTRY_NAME
0	101501	0	41	NeverM	United States of America
1	101502	1	27	NeverM	United States of America
2	101503	0	20	NeverM	United States of America
3	101504	1	45	Married	United States of America
4	101505	1	34	NeverM	United States of America
5	101506	1	38	Married	United States of America

Figure 2. 3 Transforming CUST_GENDER (Python script)

The next transformation carried out was on COUNTRY_NAME. The first step in this was to identify how many countries were recorded.

United States of America	1214
Argentina	42
Italy	33
Brazil	13
Germany	8
Poland	7
United Kingdom	6
Canada	5
Saudi Arabia	5
Denmark	4
Singapore	3
New Zealand	3
Japan	2
China	2
Spain	1
Turkey	1
France	1
South Africa	1

Table 2. 2 Frequency of countries

Using the first Python command shown in Figure 2.4, Table 2.2 was created which gives a breakdown on how many times each country appears in the dataset as well a list of all the countries in the dataset. From the table it is observed there are a few rare events which only occurred once in the dataset. It would be more sensible to classify these into one category as it could improve model performance. Hence, cases with less than 5 frequency will be grouped together and will be coded as specified in TABLE 2.1. The Python script and the output of the first few cases is shown in Figure 2.4

```
###############################################################################################
# Checking frequency of each country in the data
pd.value_counts(data['COUNTRY_NAME'])
# Country name into ordinal numbers
country_code = {
                'United States of America':1,
                'Argentina':2,
                'Italy':3,
                'Brazil':4,
                'Germany':5,
                'Poland':6,
                'Canada':7,
                'United Kingdom':8,
                'Saudi Arabia':9,
                'Denmark':10,
                'China':11,
                'Singapore':11,
                'New Zealand':11,
                'Japan':11,
                'Spain':11,
                'Turkey':11,
                'Australia':11,
                'France':11,
                'South Africa':1}
data.COUNTRY_NAME = [country_code[item] for item in data.COUNTRY_NAME]
###############################################################################################
```

	CUST_ID	CUST_GENDER	AGE	CUST_MARITAL_STATUS	COUNTRY_NAME
0	101501	0	41	NeverM	1
1	101502	1	27	NeverM	1
2	101503	0	20	NeverM	1
3	101504	1	45	Married	1
4	101505	1	34	NeverM	1
5	101506	1	38	Married	1
6	101507	1	28	Married	1
7	101508	1	19	NeverM	1
8	101509	1	52	Married	4

Figure 2. 4 Transforming COUNTRY_NAME into ordinal (Python script)

CUST_INCOME_LEVEL will be classified into 5 categories as stated in Table 2.1. The Python script used for this classification along with the output of the first few cases are shown in Figure 2.5

```
##########################################################################################
# Checking number of categories that is already classified
pd.value_counts(data['CUST_INCOME_LEVEL'])
# Customer income level into ordinal
income_level= {
'J: 190,000 - 249,999':4,
'L: 300,000 and above':5,
'I: 170,000 - 189,999':4,
'K: 250,000 - 299,999':4,
'F: 110,000 - 129,999':3,
'G: 130,000 - 149,999':3,
'E: 90,000 - 109,999':2,
'H: 150,000 - 169,999':4,
'B: 30,000 - 49,999':1,
'C: 50,000 - 69,999':2,
'D: 70,000 - 89,999':2,
'A: Below 30,000':1}
data.CUST_INCOME_LEVEL= [income_level[item] for item in data.CUST_INCOME_LEVEL]
##########################################################################################
```

	CUST_INCOME_LEVEL	EDUCATION	OCCUPATION	HOUSEHOLD_SIZE	YRS_RESIDENCE
0	4	Masters	Prof.	2	4
1	4	Bach.	Sales	2	3
2	4	HS-grad	Cleric.	2	2
3	1	Bach.	Exec.	3	5
4	4	Masters	Sales	9+	5
5	4	HS-grad	Other	3	4
6	4	< Bach.	Sales	3	3
7	4	HS-grad	Sales	2	2
8	4	Bach.	Other	3	5
9	5	Bach.	Sales	2	3
10	4	Bach.	Sales	2	5

Figure 2. 5 Transforming CUST_INCOME_LEVEL into ordinal (Python script)

The variable EDUCATION has been classified into ordinal as per Table 2.1. The number of categories of EDUCATION was reduced to 5 categories. The Python script and the regarding output for the first few cases of this process is shown in Figure 2.6

```
##########################################################################################
# Checking current classification of education
pd.value_counts(data['EDUCATION'])
# Education into ordinal level
education= {
'HS-grad':4,
'< Bach.':4,
'Bach.':5,
'Masters':5,
'Assoc-V':4,
'Assoc-A':4,
'10th':2,
'11th':3,
'Profsc':5,
'7th-8th':2,
'9th':2,
'PhD':5,
'12th':3,
'5th-6th':1,
'Presch.':1,
'1st-4th':1,}
data.EDUCATION = [education[item] for item in data.EDUCATION]
##########################################################################################
```

	CUST_INCOME_LEVEL	EDUCATION	OCCUPATION	HOUSEHOLD_SIZE	YRS_RESIDENCE
0	4	5	Prof.	2	4
1	4	5	Sales	2	3
2	4	4	Cleric.	2	2
3	1	5	Exec.	3	5
4	4	5	Sales	9+	5
5	4	4	Other	3	4
6	4	4	Sales	3	3
7	4	4	Sales	2	2
8	4	5	Other	3	5
9	5	5	Sales	2	3
10	4	5	Sales	2	5

Figure 2. 6 Transforming EDUCATION into ordinal (Python script)

HOUSEHOLD_SIZE was classified into ordinal as per Table 2.1. The data entry error was automatically corrected when it was loaded into Python. Python has changed 4-May as 4-5 and 6-Aug as 6-8. This is probably because Python identified this variable as an integer hence converted the date into an integer value. The Python script used to classify this variable into ordinal is shown in Figure 2.7.

```
#############################################################################################
# Identify how the data entry error was identified by python
pd.value_counts(data['HOUSEHOLD_SIZE'])
# household into ordinal level
household= {'1':1,'2':2,'3':3,'4-5':4, '6-8':5, '9+':6}
data.HOUSEHOLD_SIZE = [household[item] for item in data.HOUSEHOLD_SIZE]
#############################################################################################
```

	CUST_INCOME_LEVEL	EDUCATION	OCCUPATION	HOUSEHOLD_SIZE	YRS_RESIDENCE
0	4	12	Prof.	2	4
1	4	11	Sales	2	3
2	4	9	Cleric.	2	2
3	1	11	Exec.	3	5
4	4	12	Sales	6	5
5	4	9	Other	3	4
6	4	10	Sales	3	3
7	4	9	Sales	2	2
8	4	11	Other	3	5
9	5	11	Sales	2	3
10	4	11	Sales	2	5

Figure 2. 7 Transforming HOUSEHOLD_SIZE into ordinal (Python script)

Further transformation was carried out on two other variables, OCCUPATION and CUST_MARITAL_STATUS. Both variables were changed to ordinal, as indicated in Table 2.1. Now the entire dataset only has numerical values. Figure 2.8 shows the Python script used for this transformation.

```
#############################################################################################
pd.value_counts(data['OCCUPATION'])
occupation= {
'Exec.':1,
'Crafts':2,
'Sales':3,
'Cleric.':4,
'Prof.':5,
'Other':6,
'Machine':7,
'Transp.':8,
'Handler':9,
'TechSup':10,
'Farming':11,
'Protec.':12,
'House-s':13,
'Armed-F':14,}
data.OCCUPATION=[occupation[item] for item in data.OCCUPATION]

pd.value_counts(data['CUST_MARITAL_STATUS'])
marital={
'Married':1,
'NeverM':2,
'Divorc.':3,
'Separ.':4,
'Widowed':5,
'Mabsent':6,
'Mar-AF':7,}
data.CUST_MARITAL_STATUS=[marital[item] for item in data.CUST_MARITAL_STATUS]
#############################################################################################
```

Figure 2. 8 Transforming other variable into ordinal (Python script)

3. DATA ANALYSIS

3.1 Summary statistics

The summary statistics of a variables reveals many information. It is a good practice to check the summary statistics of variables before proceeding onto modelling. Figure 3.1 shows a Python code designed to calculate the summary statistics of any variables. The output of the summary statistics for the variable EDUCATION is shown in Table 3.1.

```
Choose the variables number from list shown
customer gender          - 1
Age                      - 2
Marital status           - 3
Country name             - 4
Income level             - 5
Education                - 6
Occupation               - 7
Household size           - 8
Yrs residence            - 9
Affinity card            - 10
Bulk Pack Diskettes      - 11
Flat panel monitor       - 12
Home theater package     - 13
Bookkeeping application  - 14
Y box games              - 15

Enter the respective number of variable to obtain summary statistics:6
The sum is 5468.
The Mean is 4.047372
The Standard Deviation is 0.819651
The Skewness is -1.348569
The Kurtosis is 2.544475
```

Table 3. 1 Summary Statistics Table for EDUCATION

The Python script is designed in a way to prompt the user to choose a variable number from the shown list. Once the user chooses the input, the summary statistics will be displayed. Figure 3.1 shows the Python script that creates the output shown in Table 3.1.

```
#############################################################################################
# create a dictonary for all variables
dic={ 1:'CUST_GENDER',
    2:'AGE',
    3:'CUST_MARITAL_STATUS',
    4:'COUNTRY_NAME',
    5:'CUST_INCOME_LEVEL',
    6:'EDUCATION',
    7:'OCCUPATION',
    8:'HOUSEHOLD_SIZE',
    9:'YRS_RESIDENCE',
    10:'AFFINITY_CARD',
    11:'BULK_PACK_DISKETTES',
    12:'FLAT_PANEL_MONITOR',
    13:'HOME_THEATER_PACKAGE',
    14:'BOOKKEEPING_APPLICATION',
    15:'Y_BOX_GAMES',}
# provide a list of variable number to choose from
print('Choose the variables number from list shown',
    'customer gender          - 1',
    'Age                      - 2',
    'Marital status           - 3',
    'Country name             - 4',
    'Income level             - 5',
    'Education                - 6',
    'Occupation               - 7',
    'Household size           - 8',
    'Yrs residence            - 9',
    'Affinity card            - 10',
    'Bulk Pack Diskettes      - 11',
    'Flat panel monitor       - 12',
    'Home theater package     - 13',
    'Bookkeeping application  - 14',
    'Y box games              - 15',
    sep="\n")
# store the user choice
x=int(input(
'Enter the respective number of variable to obtain summary statistics:'))
# Calculation
SUM=data[dic[x]].sum()
MEAN=data[dic[x]].mean()
Standard_deviation=data[dic[x]].std()
Skewness=data[dic[x]].skew()
Kurtosis=data[dic[x]].kurt()
print('The sum is %d.' % SUM,
       'The Mean is %f' %MEAN,
       'The Standard Deviation is %f' %Standard_deviation,
       'The Skewness is %f' % Skewness,
       'The Kurtosis is %f' %Kurtosis,
        sep="\n")
#############################################################################################
```

Figure 3. 1 Summary statistics of chosen variable (Python script)

3.2 Correlation

The correlation of each variable with the target variable should be explored to understand which variable or variables have the most impact on the outcome of the target variable. While there are many measures of association for variables which are measured at the ordinal or higher level of measurement, correlation is the most commonly used approach (Robin, 2012).

A Python script is used to create a table showing the correlation of the explanatory variables on the target variable from the highest correlation to the lowest correlation. Figure 3.2 shows the Python script along with the results table.

```
##########################################################################################
# Correlation of target variable with other variables
Correlation= data.corr()['AFFINITY_CARD']
print(Correlation.sort_values(ascending=False))
##########################################################################################
```

```
AFFINITY_CARD              1.000000
YRS_RESIDENCE              0.345993
EDUCATION                  0.286274
HOME_THEATER_PACKAGE       0.285380
AGE                        0.248632
CUST_GENDER                0.233862
BOOKKEEPING_APPLICATION    0.166456
HOUSEHOLD_SIZE             0.050336
COUNTRY_NAME               0.022155
CUST_INCOME_LEVEL         -0.017484
BULK_PACK_DISKETTES       -0.018654
CUST_ID                   -0.028447
FLAT_PANEL_MONITOR        -0.029114
Y_BOX_GAMES               -0.284964
Name: AFFINITY_CARD, dtype: float64
```

Figure 3. 2 Correlation of explanatory variables to target variables (Python script)

From Figure 3.2 it can be identified that YRS_RESIDENCE has the highest positive correlation and Y_BOX_GAMES has the highest negative correlation. This means to say that an increase in YRS_RESIDENCE will have an increasing probability of taking AFFINITY_CARD. Likewise for a negative correlation, a decrease in probability of taking an AFFNITY_CARD.

3.3 Euclidean distance

The basis of many measures of similarity and dissimilarity is Euclidean distance. Euclidean distance is the square root of the sum of squared differences between corresponding elements of the two vectors. Euclidean distance is most often used to compare profiles of respondents across variables (Steve, 2014). For example, suppose the data consist of demographic information on a sample of individuals, arranged as a respondent-by-variable matrix. Each row of the matrix is a vector of *m* numbers, where *m* is the number of variables. This can be used to evaluate the similarity (or, in this case, the distance) between any pair of rows (Steve, 2014).

```
##########################################################################################
# Euclidean Distance
from scipy.spatial import distance
# prompt to enter Customer ID
Customer_ID_1= int(input(
'Enter ID number of customer:'))
Customer_ID_2= int(input(
'Ebter ID number of next customer:'))
# Euclidean distance calculation
euc_dst= distance.euclidean(data.loc[Customer_ID_1],data.loc[Customer_ID_2])
print('The Euclidean distance is %f.' % euc_dst)
##########################################################################################
```

```
Enter ID number of customer:5

Ebter ID number of next customer:6
The Euclidean distance is 10.630146.
```

Figure 3. 3 Euclidean distance calculation (Python script)

Figure 3.3 shows the Python script and the result for calculation the Euclidean distance between two customers. The Python code will prompt the user to input the Customer_ID_1 and Customer_ID_2 for the calculation. In the above code the Customer_ID_1 is set as 5 and Customer_ID_2 is set as 6. The resulting Euclidean distance is calculated as 10.630

4. DATA EXPLORATION

4.1 Histogram

One of the most effective way of analysing the data is visualising the data. The shape of a histogram also shows the observed values spread around the location and also revels information about skewness and kurtosis (Shahbaba, 2012). Here a Python script(shown in Figure 4.2) is developed to create histogram of any variable chosen by a user.

Running the Python code prompts the user to enter the respective number of the variable that the histogram should be created for. The output when an user chooses variable 7 (OCCUPATION) is shown below in Figure 4.1

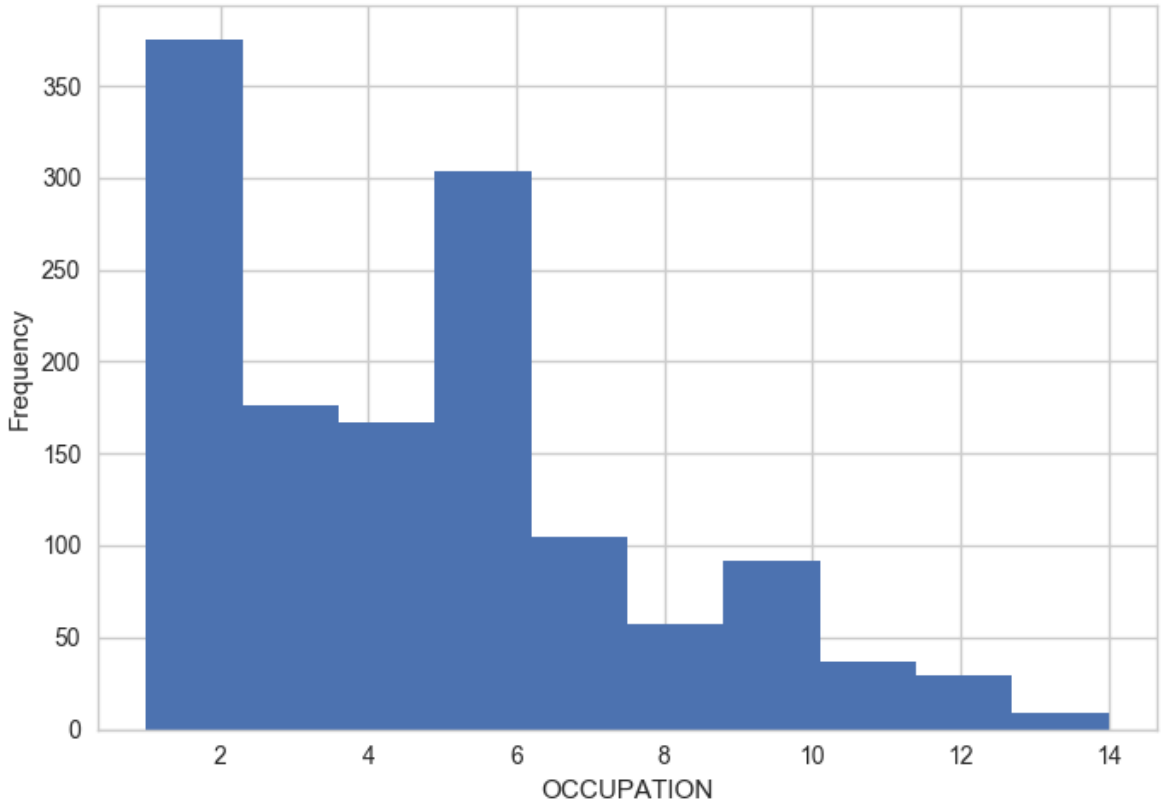

Figure 4. 1 Histogram of OCCUPATION

From Figure 4.1 the user can see a breakdown on how many customers recorded had a job as a sales person, a prof and so on. The bins specified in the histogram is 10, but this could be edited according to the needs. Increasing nbins could give a more detailed breakdown of the histogram for analysis. Decreasing nbins could give a more general histogram which may be easier to analyse.

```
##########################################################################################
import matplotlib.pyplot as plt

# create a dictonary for all variables
dic={
1:'CUST_GENDER',
2:'AGE',
3:'CUST_MARITAL_STATUS',
4:'COUNTRY_NAME',
5:'CUST_INCOME_LEVEL',
6:'EDUCATION',
7:'OCCUPATION',
8:'HOUSEHOLD_SIZE',
9:'YRS_RESIDENCE',
10:'AFFINITY_CARD',
11:'BULK_PACK_DISKETTES',
12:'FLAT_PANEL_MONITOR',
13:'HOME_THEATER_PACKAGE',
14:'BOOKKEEPING_APPLICATION',
15:'Y_BOX_GAMES',}
# provide a list of variable number to choose from

print('Choose the variables number from list shown',
    'customer gender          - 1',
    'Age                      - 2',
    'Marital status           - 3',
    'Country name             - 4',
    'Income level             - 5',
    'Education                - 6',
    'Occupation               - 7',
    'Household size           - 8',
    'Yrs residence            - 9',
    'Affinity card            - 10',
    'Bulk Pack Diskettes      - 11',
    'Flat panel monitor       - 12',
    'Home theater package     - 13',
    'Bookkeeping application  - 14',
    'Y box games              - 15',
    sep="\n")

# store the user choice
num=int(input('Enter the respective number of the variable:'))

#plot the histogram
data[dic[num]].plot(kind='hist',stacked=True,bins=10)
# label x axis according the user choice
plt.xlabel(dic[num])
##########################################################################################
```

Figure 4. 2 Histogram (Python script)

4.2 Scatter plot

A scatterplot shows the relationship between two quantitative variables measured for the same individuals and each individual in the data appears as a point on the graph (Diana.M, 2010). The Python script shown in Figure 4.3 is used to create scatterplot for any two variables chosen by a user.

```
#############################################################################################
# create a dictonary for all variables
dic={
1:'CUST_GENDER',
2:'AGE',
3:'CUST_MARITAL_STATUS',
4:'COUNTRY_NAME',
5:'CUST_INCOME_LEVEL',
6:'EDUCATION',
7:'OCCUPATION',
8:'HOUSEHOLD_SIZE',
9:'YRS_RESIDENCE',
10:'AFFINITY_CARD',
11:'BULK_PACK_DISKETTES',
12:'FLAT_PANEL_MONITOR',
13:'HOME_THEATER_PACKAGE',
14:'BOOKKEEPING_APPLICATION',
15:'Y_BOX_GAMES',}
# provide a list of variable number to choose from

print('Choose the variables number from list shown',
    'customer gender          - 1',
    'Age                      - 2',
    'Marital status           - 3',
    'Country name             - 4',
    'Income level             - 5',
    'Education                - 6',
    'Occupation               - 7',
    'Household size           - 8',
    'Yrs residence            - 9',
    'Affinity card            - 10',
    'Bulk Pack Diskettes      - 11',
    'Flat panel monitor       - 12',
    'Home theater package     - 13',
    'Bookkeeping application  - 14',
    'Y box games              - 15',
    sep="\n")

# store the user choice
x=int(input('Enter the respective number of the variable for x axis:'))
y=int(input('Enter the respective number of the variable for y axis:'))

#plot the scatter plot
plt.scatter(data[dic[x]],data[dic[y]])
# label x and y axis
plt.xlabel(dic[x])
plt.ylabel(dic[y])
#############################################################################################
```

Figure 4. 3 Scatter plot (Python script)

Running the script shown in Figure 4.3 prompts the user to choose the variables to be plotted. The first prompt lets the user chooses the variable number to be in the x-axis and then second prompt for the y-axis. Choosing this variables will output a scatterplot of the chosen two variables. As an example when a user chooses 4 and then 6 the scatter plot shown in Figure 4.4 will be created.

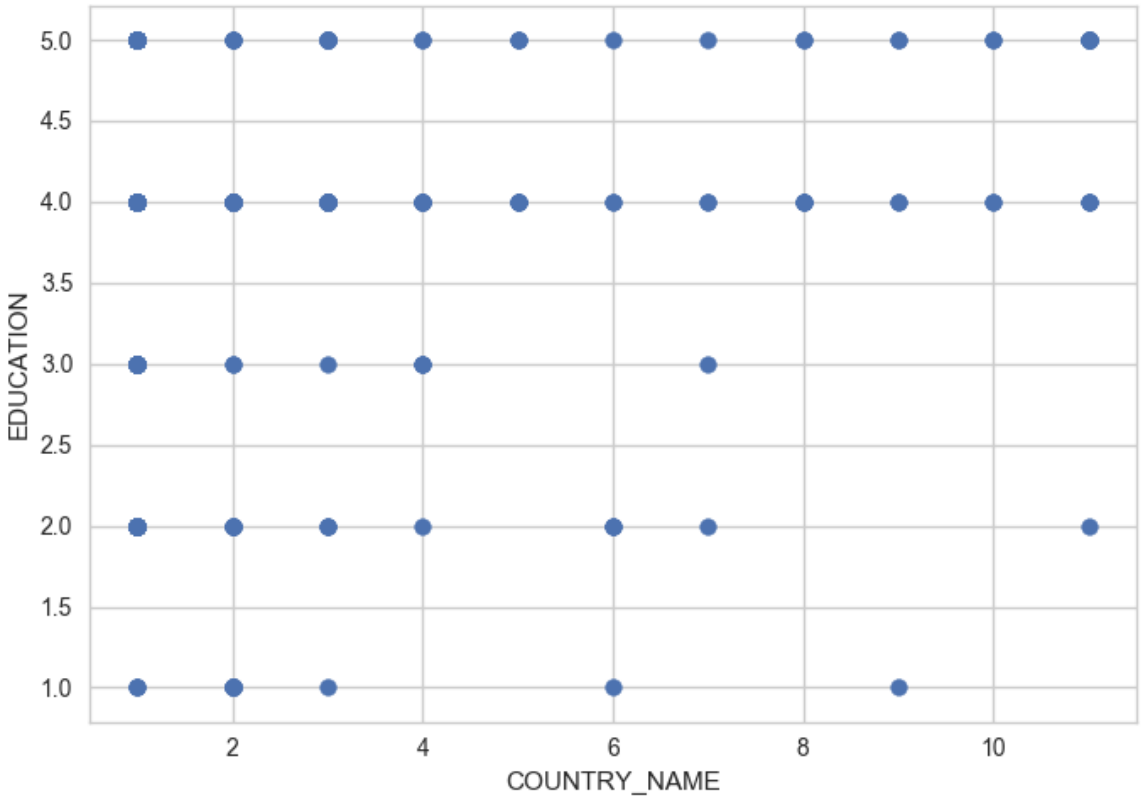

Figure 4. 4 Scatter plot of two chosen variables

Analysis on the data can be carried out from this plot for example, in United Kingdom the collected data only entitles of customers with high education levels only.

5. DATA MODELLING

There are many choices of modelling techniques available today which deals with various kinds of data. For this dataset, the modelling technique that will be used is the logistic regression model. Logistic regression is a machine learning classification algorithm that is used to predict the probability of categorical dependent variable (Apoorva, 2017). In logistic regression the dependent variable is a binary variable that contains data codes as 1 or 0. In other words the logistic regression model predicts probability of Y=1 as a function of X.

Before building the model, it is a good practice to partition the data into training and testing dataset so that the testing dataset could be used to test the data. The modified dataset is split into 75% for training and 25% testing. The model was fitted using the training dataset and the results are shown in Table 5.1

Logit Regression Results

Dep. Variable:	AFFINITY_CARD	No. Observations:	1013
Model:	Logit	Df Residuals:	999
Method:	MLE	Df Model:	13
Date:	Sun, 20 May 2018	Pseudo R-squ.:	0.3180
Time:	23:03:00	Log-Likelihood:	-393.46
converged:	True	LL-Null:	-576.94
		LLR p-value:	2.092e-70

	coef	std err	z	P>|z|	[0.025	0.975]
CUST_GENDER	-0.1660	0.243	-0.684	0.494	-0.642	0.310
AGE	-0.0402	0.012	-3.396	0.001	-0.063	-0.017
CUST_MARITAL_STATUS	-1.5059	0.170	-8.884	0.000	-1.838	-1.174
COUNTRY_NAME	0.0207	0.062	0.334	0.739	-0.101	0.142
CUST_INCOME_LEVEL	-0.1646	0.149	-1.105	0.269	-0.457	0.127
EDUCATION	0.7926	0.164	4.839	0.000	0.472	1.114
OCCUPATION	-0.1030	0.030	-3.463	0.001	-0.161	-0.045
HOUSEHOLD_SIZE	-0.1778	0.105	-1.690	0.091	-0.384	0.028
YRS_RESIDENCE	0.2818	0.070	4.046	0.000	0.145	0.418
BULK_PACK_DISKETTES	0.9557	0.541	1.766	0.077	-0.105	2.016
FLAT_PANEL_MONITOR	-0.5355	0.459	-1.167	0.243	-1.435	0.364
HOME_THEATER_PACKAGE	0.7960	0.301	2.648	0.008	0.207	1.385
BOOKKEEPING_APPLICATION	-0.6934	0.491	-1.414	0.157	-1.655	0.268
Y_BOX_GAMES	-1.4552	0.367	-3.961	0.000	-2.175	-0.735

Table 5. 1 Logistic regression Results

Observing Table 5.1 it can be observed that most of the variables has a p-value smaller than 0.05. This indicated that most of the variables are significant to the model. Variables with p-value of 0.05 could be removed from the model to have some improvements to the model. In this case, these variables are kept in the model.

Fitting the logistic regression model gives the output shown:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

These output are the default setting used by the logistic regression model. These setting could be configured to fine tune the regression model to improve accuracy.

After fitting the model, the test dataset was used to check the adequacy of the model. A confusion matrix was created to view the classifications done by the regression model. This matrix allows the user to identify how many right and wrong predictions are there for both '0' and '1' outcome. Table 5.2 shows the confusion matrix table with the results obtained from the regression model.

	Predicted: NO	Predicted: Yes
Actual: NO	217	25
Actual: Yes	53	43

Table 5. 2 Confusion matrix

The accuracy of the model, as calculated in the Python code is 0.77. That means a misclassification rate of 23%. The misclassification can also be calculated from the confusion matrix by the formula, $misclassification = \frac{Total\ incorrect\ predictions}{Total\ number\ of\ cases}$.

Finally a ROC curve was created to check the adequacy of the model. Figure 5.1 shows the ROC curve for the test data set.

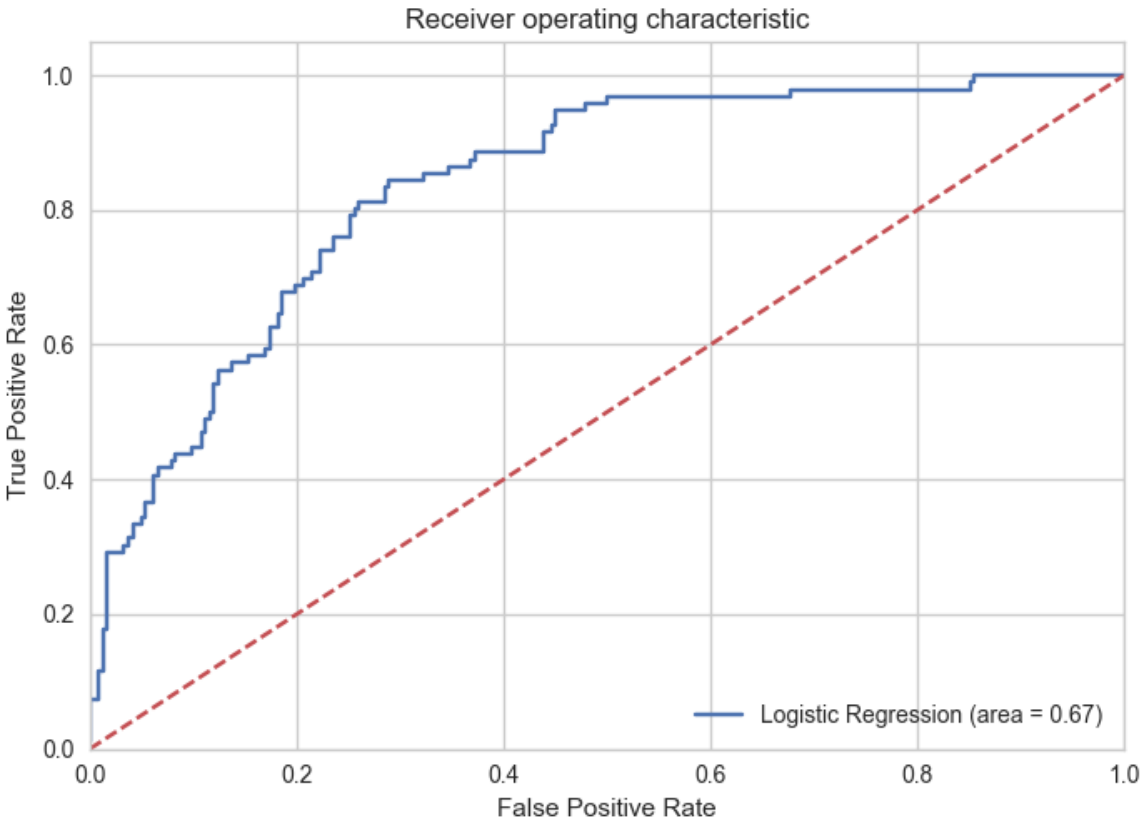

Figure 5. 1 ROC curve

The receiver operating characteristic (ROC) curve is a common tool used with binary classifiers. The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible. Judging from the ROC curve, the created model is not a bad model. A ROC curve can be used to make a decision on the decision threshold. The AUC is calculated for the ROC curve as shown in the bottom left of figure 5.1, the AUC value of 0.67 should be set as the decision threshold to get the best accuracy for the model.

The Python script along with all the explanation of the codes are shown in Figure 5.2.

```
#############################################################################################
# Data modelling LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
# classifying explanatory variables into x and target variable into y
y = data.iloc[:,[10]]
X = data.iloc[:,[1,2,3,4,5,6,7,8,9,11,12,13,14,15]]
# spliting data into training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
#checking training and test data partition
X_train.shape
X_test.shape
# implementing the model
import statsmodels.api as sm
logit_model=sm.Logit(y_train,X_train)
#checking summary statistics to check for significance of variable
result=logit_model.fit()
print(result.summary())
# Fit logistic regression to the training set
logic = LogisticRegression(random_state=0)
logic.fit(X_train, y_train)

# Predict test set results and confusion matrix
y_pred = logic.predict(X_test)
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
# Accuracy of model
print
('Accuracy on test data:{:.2f}'.format(logic.score(X_test, y_test)))

# Creating ROC curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logic.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logic.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
# plot base line
plt.plot([0, 1], [0, 1],'r--')
# Set axis limit for x-axis and y-axis
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
# label axis
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
# Dispay AUC at lower right
plt.legend(loc="lower right")

#############################################################################################
```

Figure 5. 2 Logistic regression mode (Python script)

5. DISCUSSION AND REFLECTION

The dataset was cleaned and analysed using a logistic regression model. A misclassification rate of 23% was obtained for the training dataset. This could be further improved by improving the setting of the regression model, exploring every variables to observe trends, eliminating variables with low p-values and classifying rare categories together. To carry out such process on Python is time consuming and a lot of work.

Python is effective to carry out the data preparation part of data mining. Transforming variables and deleting rows with missing values can be done effectively. The downside of Python is mainly when it comes to data exploring and data modelling. Software such as SAS Enterprise miner is able to carry out data exploring in more efficient and effective ways. Modelling on Python is time consuming as each model has to be individually created and coded, this comes along with errors in codes which consumes a lot of time to rectify. Overall Python should be used in the data preparation part of data mining, but exploring data and modelling should be left to advanced software such as SAS Enterprise miner which allows multiple models to be created within minutes.

REFERENCES

Apoorva, A. (2017, Mar 31). *Logistic Regression.Simplified*. Retrieved from Data Science Group,llTR: https://medium.com/data-science-group-iitr/logistic-regression-simplified-9b4efe801389

Diana.M, P. (2010). Scatterplots and correlation. In *The basic practice of statistics(6thed.).*

Robin, B. (2012, September 19). *An introduction to statistics Correlation*. Retrieved from http://www.floppybunny.org/robin/web/virtualclassroom/stats/basics/part9.pdf

Shahbaba, B. (2012). Data Exploration. In *An introduction to statistics through biological data* (pp. 17-59).

Steve, B. (2014). *Distance and correlation*. Retrieved from Analytictech: http://www.analytictech.com/mb876/handouts/distance_and_correlation.htm