Course Work Technical Report

CC7182-Programming for Data Analytics

Rana danial Tajammal- 22026355

Contents

1 Data Understanding

Several libraries, such as Pandas for data manipulation, Matplotlib for data visualization, and Scikit-learn for machine learning tasks, can be used to perform data analysis on the Market Campaign Dataset. These libraries can all be used with the widely used Python programming language, version 3x a particular environment, such as Google Colab or Visual Studio, used to run the code. The Market Campaign Dataset was looked over, shrunk, cleaned, converted, and examined to find models that could forecast the result.

The First Phase to Analysis the Market Campaign Dataset , it consists of :

1.1 Metadata Tables

After read the market campaign dataset by using the Pandas Dataframe object, then use the info( ) Fig .1, which gives the concise summary of DataFrame as observed in Fig.2.

```
# information of each feature in dataset
market_campaign_dataset.info()
```

Fig. 1: Metadata table Code

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CUST_ID                 1500 non-null   int64
 1   CUST_GENDER             1500 non-null   object
 2   AGE                     1500 non-null   int64
 3   CUST_MARITAL_STATUS     1500 non-null   object
 4   COUNTRY_NAME            1500 non-null   object
 5   CUST_INCOME_LEVEL       1500 non-null   object
 6   EDUCATION               1500 non-null   object
 7   OCCUPATION              1500 non-null   object
 8   HOUSEHOLD_SIZE          1500 non-null   object
 9   YRS_RESIDENCE           1500 non-null   int64
 10  AFFINITY_CARD           1500 non-null   int64
 11  BULK_PACK_DISKETTES     1500 non-null   int64
 12  FLAT_PANEL_MONITOR      1500 non-null   int64
 13  HOME_THEATER_PACKAGE    1500 non-null   int64
 14  BOOKKEEPING_APPLICATION 1500 non-null   int64
 15  PRINTER_SUPPLIES        1500 non-null   int64
 16  Y_BOX_GAMES             1500 non-null   int64
 17  OS_DOC_SET_KANJI        1500 non-null   int64
 18  COMMENTS                1427 non-null   object
dtypes: int64(11), object(8)
memory usage: 222.8+ KB
```

Fig. 2: Metadata table Output

As you observed above in Fig. 2, Market Campaign Dataset contains 1500 records which include a header row , there 1499 cases.Each Record contains 19 Attributes, which includes socio-demographic and product ownership information. One of the Feature like COMMENTS

contain the null , missing or blank values and data type could be changed into int64. After reducing and filtering, there will be a reason to transform the features being utilized for analysis, which will happen during transformation.

1.2 Missing and Erroneous Data

In Fig. 3 as you seen , By using the isnull() and sum() functions to check the column name, which column name contains NAN (null or Not a Number in Python terms) or missing values, can be observed by Fig.4.

```
# Identify the Null or Missing Value Columns
market_campaign_dataset.isnull().sum()
```

Fig. 3: Missing or Error Data Code of Each Feature

According Above Fig. 3, Comments Feature in Market Campaign Dataset contains the NAN(null or any type of Missing Values) or Missing Values

```
CUST_ID                      0
CUST_GENDER                  0
AGE                          0
CUST_MARITAL_STATUS          0
COUNTRY_NAME                 0
CUST_INCOME_LEVEL            0
EDUCATION                    0
OCCUPATION                   0
HOUSEHOLD_SIZE               0
YRS_RESIDENCE                0
AFFINITY_CARD                0
BULK_PACK_DISKETTES          0
FLAT_PANEL_MONITOR           0
HOME_THEATER_PACKAGE         0
BOOKKEEPING_APPLICATION      0
PRINTER_SUPPLIES             0
Y_BOX_GAMES                  0
OS_DOC_SET_KANJI             0
COMMENTS                    73
dtype: int64
```

Fig. 4: Missing or Error Data of each Feature

As you observed in Fig.4, COMMENTS Attribute of Market Campaign Dataset contains NaN (null or Not a Number in Python terms) or is the word “Blanks” changed into a “?” Fig. 5 and displaying the data frame afterwards can be observed in Fig. 6.

```
# null or missing values are replace with '?'
market_campaign_dataset['COMMENTS']=market_campaign_dataset['COMMENTS'].replace(np.nan , '?')
#  Blanks are replace with '?'
market_campaign_dataset['COMMENTS']=market_campaign_dataset['COMMENTS'].replace('Blanks' , '?')
# Display the 30 Records of COMMENTS Feature in Dataset
print(market_campaign_dataset['COMMENTS'].head(30))
```

Fig. 5: Missing or Error Data Code of Comments feature

```
50    Can I use my Affinity card to buy bulk purchas...
51    My brother uses the affinity card a lot. I thi...
52    Affinity card is a great idea. But your store ...
53    Thank you, But please remove my name from your...
54    Affinity card is great. I think it is a hassle...
55    Affinity card makese sense only for bulk purch...
56    Can I use my Affinity card to buy bulk purchas...
57    I love shopping with my Affinity Card! Thank y...
58    I don't like your new Affinity Card program.  ...
59    I run a small convenience store. Any chance th...
60    I am unhappy with the service at your store. D...
61    Affinity card is a great idea. But your store ...
62    Thanks but even with your discounts, your prod...
63    I purchased the new mouse pads and love them. ...
64     I love it. Will never shop at other shops again!
65                                                    ?
66    I used to shop at your store, but have stopped...
67    I shop your store a lot.  I love your weekly s...
68    Affinity card makese sense only for bulk purch...
69    Affinity card is great. I think it is a hassle...
70    Thanks but even with your discounts, your prod...
71    I used to shop at your store, but have stopped...
72    Thanks but even with your discounts, your prod...
73    I love shopping with my Affinity Card! Thank y...
74    Forget it. I 'm not giving you all my personal...
75    I am not going to waste my time filling up thi...
76                                                    ?
77    Thanks but even with your discounts, your prod...
78    A lousy idea.  I threw your card away. If you ...
79    I run a small convenience store. Any chance th...
80    I am not going to waste my time filling up thi...
81    The new affinity card is great. Thank you. I d...
82    Don't send me any more promotions.  I get too ...
83    Dear store manager, please do not send me any ...
84    A great program but I have to complain just a ...
85                                                    ?
86                                                    ?
87    Don't send me any more promotions.  I get too ...
88    I purchased the new mouse pads and love them. ...
89    A lousy idea.  I threw your card away. If you ...
90    I shop your store a lot.  I love your weekly s...
91    I purchased the new mouse pads and love them. ...
92    I purchased a new computer from your store rec...
93    I used to shop at your store, but have stopped...
94    Could you send an Affinity Card to my mother i...
95    I am unhappy with the service at your store. D...
96    I love the discounts. But I mostly end upbuyin...
97    The more times that I shop at your store, the ...
98    Affinity card makese sense only for bulk purch...
99    Why didn't you start a program like this befor...
Name: COMMENTS, dtype: object
```

Figure 6: Missing or Error Data Output of Comment Feature

Once we see missing values denoted by a "?" they are changed into True/False values which is achieved with the code in Fig. 7 and displayed in Fig. 8.

```
# Identify the Null or Missing Values
market_campaign_dataset.isna()
```

Fig. 7: Missing or Error Data Code (isnull() isna())

	CUST_ID	CUST_GENDER	AGE	CUST_MARITAL_STATUS	COUNTRY_NAME	CUST_INCOME_LEVEL	EDUCATION	OCCUPATION	HOUSEHOLD_SIZE	YRS_RESIDENCE	AFFINITY_CARD	BULK_PACK_DIS
0	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	
1495	False	False	False	False	False	False	False	False	False	False	False	
1496	False	False	False	False	False	False	False	False	False	False	False	
1497	False	False	False	False	False	False	False	False	False	False	False	
1498	False	False	False	False	False	False	False	False	False	False	False	
1499	False	False	False	False	False	False	False	False	False	False	False	

1500 rows × 19 columns

Fig. 8: Missing or Error Data Output (isnull() isna())

By adding the True/False values for each variable (Fig. 9) and visualizing them in Fig. 10 it is then possible to calculate the precise amount of missing data as a percentage of the total data.

```
# sum the missing or null values of each feature * 100
percentage_market_campaign_dataset=((market_campaign_dataset.isna() | market_campaign_dataset.isnull()).sum() * 100)
# sum divided by total record of dataset and value has 2 decimal
percentage_market_campaign_dataset=(percentage_market_campaign_dataset/len(market_campaign_dataset)).round(2)
# Calculate the Percentage of Each feature
print(percentage_market_campaign_dataset)
```

Fig. 9: Missing or Error Data Code (%)

```
CUST_ID                     0.00
CUST_GENDER                 0.00
AGE                         0.00
CUST_MARITAL_STATUS         0.00
COUNTRY_NAME                0.00
CUST_INCOME_LEVEL           0.00
EDUCATION                   0.00
OCCUPATION                  0.00
HOUSEHOLD_SIZE              0.00
YRS_RESIDENCE               0.00
AFFINITY_CARD               0.00
BULK_PACK_DISKETTES         0.00
FLAT_PANEL_MONITOR          0.00
HOME_THEATER_PACKAGE        0.00
BOOKKEEPING_APPLICATION     0.00
PRINTER_SUPPLIES            0.00
Y_BOX_GAMES                 0.00
OS_DOC_SET_KANJI            0.00
COMMENTS                    4.87
dtype: float64
```

Fig. 10: Missing or Error Data Output (%)

Observing the output in Fig. 10 , we see that COMMENTS Feature in Market Campaign Dataset has 4.87% NAN(Missing or null) or word of Blanks Values.

2. Data Preparation

Data preparation is the process of

- Reducing the Variables from Pandas DataFrame, which do not affect the performance of Target Variable.
- Pre-processing the data involves removing values that don't add anything or that might introduce bias.
- Data transformation is necessary for successful analysis of changing data.

2.1 Reducing Variables

After analyzing the Market Campaign Dataset , it seems some variables of Market Campaign Dataset do not affect the performance of Target Variable, showing the result remains the same. So, reducing the Variables from the Market Campaign Dataset. These Variables are :

- PRINTER_SUPPLIES
- OS_DOC_SET_KANJI
- CUST_ID
- COMMENTS
- HOUSEHOLD_SIZE
- EDUCATION

These variables are removed from the original DataFrame by using drop() built in function, figure . 11 illustrates this process and figure . 12 the output.

```
market_campaign_dataset=market_campaign_dataset.drop(columns=['PRINTER_SUPPLIES' , 'OS_DOC_SET_KANJI'])
```

```
# Drop the columns e-g CUST_ID, COMMENTS , index, HOUSEHOLD_SIZE and EDUCATION
market_campaign_dataset=market_campaign_dataset.drop(columns=['CUST_ID' , 'COMMENTS' , 'index' , 'HOUSEHOLD_SIZE' , 'EDUCATION'])
print(market_campaign_dataset.head())
```

Fig. 11: Feature Reduction Code

```
market_campaign_dataset.head()
```

	CUST_GENDER	AGE	CUST_MARITAL_STATUS	COUNTRY_NAME	OCCUPATION	YRS_RESIDENCE	AFFINITY_CARD	BULK_PACK_DISKETTES	FLAT_PANEL_MONITOR	HOME_THEATER_PACKAGE	BOOKKEEPING_APPLICATION	Y_BOX_GAMES	CUST_INCOME_LEVEL
0	0	41	3	17	9	4	0	1	1	1	1	0	7.0
1	1	27	3	17	11	3	0	1	1	0	1	1	6.0
2	0	20	3	17	1	2	0	1	0	0	1	1	5.0
3	1	45	2	17	3	5	1	0	0	1	1	0	1.0
4	1	34	3	17	11	5	1	1	1	0	1	0	8.0

Fig. 12: Feature Reduction Output

2.2 Data Pre-processing

Data filtering based on the initial data comprehension stage and deleting the '?' unknown records from original Market Campaign Dataset are two processes that can be included in data pre-processing. The outcomes are shown in Fig. 13.

```
# replace the '?' value is null values
market_campaign_dataset=market_campaign_dataset.replace('?' , np.nan)
# remove the null values rows from dataset
market_campaign_dataset=market_campaign_dataset.dropna()
```

Fig. 13: Feature Filtering Code

It can be seen that Firstly , replace the '?' unknown variable from original Campaign Dataset with NAN(Missing or Null Values), Then remove the NAN(Missing or Null) records from original Dataset.

2.3 Data Transforming

Data transformation is the process of transferring information from one format into a more appropriate one. It also deals with replacing any missing values in the data. We'll talk about the transformation rules that were included in the specification.

2.3.1 CUST_GENDER

CUST_GENDER variable CUST_GENDER into binary Female - 0, has a Male 1. The code to achieve this is observed in Fig. 14.

```
# Tranform the CUST_GENDER variable  into binary F - 0, M -1
market_campaign_dataset['CUST_GENDER']=market_campaign_dataset['CUST_GENDER'].apply(lambda cust_gender : 0 if cust_gender== 'F' else 1)
print(market_campaign_dataset['CUST_GENDER'])
```

Fig. 14: *CUST_GENDER* Transformation Code

2.3.2 COUNTRY_NAME

COUNTRY_NAME is a variable and is transformed into an ordinal number based on their occurrence in the data set in descending order. The code to achieve this is observed in Fig. 15.

```
# COUNTRY_NAME into ordinal number based on their occurrence in the data set in descending order.
market_campaign_dataset['COUNTRY_NAME']=sorted(pd.factorize(market_campaign_dataset['COUNTRY_NAME'].sort_values(ascending=False))[0], reverse=True)
print(market_campaign_dataset['COUNTRY_NAME'])
```

Fig. 15: COUNTRY_NAME Transformation Code

2.3.3 CUST_INCOME_LEVEL

CUST_INCOME_LEVEL is a variable , Before Transforming CUST_INCOME_LEVEL columns split into three further columns like CUST_INCOME_LEVEL_0 , CUST_INCOME_LEVEL_1 and CUST_INCOME_LEVEL_2 and remove the CUST_INCOME_LEVEL_0 and CUST_INCOME_LEVEL from original Dataset.The code to achieve this is observed in Fig. 16.

```
# split the CUST_INCOME_LEVE into further three columns
market_campaign_dataset[['CUST_INCOME_LEVEL_0' , 'CUST_INCOME_LEVEL_1' , 'CUST_INCOME_LEVEL_2']]=market_campaign_dataset['CUST_INCOME_LEVEL'].str.split(': | - | and' , expand=True)
# Drop the Columns
market_campaign_dataset=market_campaign_dataset.drop(columns=['CUST_INCOME_LEVEL_0' , 'CUST_INCOME_LEVEL'] , axis=1)
```

Fig. 16: *CUST_INCOME_LEVEL* Split Code

As it will be seen that CUST_INCOME_LEVEL feature of Market Campaign Dataset contains ' Below 30000' and ‘above’. These values of records are replaced with numeric Values of Records. The code to achieve this is observed in fig. 17.

```
# replace the above with digit in CUST_INCOME_LEVE_2 Feature of Dataset
market_campaign_dataset['CUST_INCOME_LEVEL_2']=market_campaign_dataset['CUST_INCOME_LEVEL_2'].replace(' above' , '9999999')
# replace the Below 30000 with digit in CUST_INCOME_LEVE_2 Feature of Dataset
market_campaign_dataset['CUST_INCOME_LEVEL_1']=market_campaign_dataset['CUST_INCOME_LEVEL_2'].replace(' Below 30000' , '30000')
```

Fig. 17: *CUST_INCOME_LEVEL* Replace record value Code

CUST_INCOME_LEVEL is a variable and transformed into ordinal numbers (1 - 12). By using the qcut()function of Pandas Library and define the ordinal numbers(1-12) and transform it.

```
# Drop rows with missing values
market_campaign_dataset = market_campaign_dataset.dropna(subset=['CUST_INCOME_LEVEL_1', 'CUST_INCOME_LEVEL_2'])

# Remove commas from CUST_INCOME_LEVEL_1 column and convert to integer
market_campaign_dataset['CUST_INCOME_LEVEL_1'] = market_campaign_dataset['CUST_INCOME_LEVEL_1'].str.replace(',', '').astype(int)

# Remove commas from CUST_INCOME_LEVEL_2 column and convert to integer
market_campaign_dataset['CUST_INCOME_LEVEL_2'] = market_campaign_dataset['CUST_INCOME_LEVEL_2'].str.replace(',', '').astype(int)

# Convert income level columns to ordinal numbers
for i, col in enumerate(['CUST_INCOME_LEVEL_1', 'CUST_INCOME_LEVEL_2']):
    market_campaign_dataset[col] = pd.qcut(market_campaign_dataset[col], q=12, labels=False, duplicates='drop') + 1

# Add a new column for the combined income level
market_campaign_dataset['CUST_INCOME_LEVEL'] = (market_campaign_dataset['CUST_INCOME_LEVEL_1'] + market_campaign_dataset['CUST_INCOME_LEVEL_2']) / 2
market_campaign_dataset=market_campaign_dataset.drop(columns=['CUST_INCOME_LEVEL_1', 'CUST_INCOME_LEVEL_2'] , axis=1)
# Print the updated dataset
print(market_campaign_dataset.head())
```

Fig. 18: *CUST_INCOME_LEVEL* Transformed Code

2.3.4 EDUCATION

EDUCATION is a predictor variable and converted into ordinal numbers based on USA education Level in descending order. The code to achieve this is observed in Fig. 19.

```
# Define a dictionary with ordinal values for each education level
ordinal_values = {'PhD':15, 'Masters':14, 'Profsc':13, 'Bach.':12, 'Assoc-V':11, 'Assoc-A':10,
                  'HS-grad':9, '< Bach.':8, '12th':7, '11th':6, '10th':5, '9th':4,
                  '7th-8th':3, '5th-6th':2, '1st-4th':1, 'Presch.':0}


# Map the integer values to ordinal values based on the USA education level in descending order
market_campaign_dataset['EDUCATION'] = sorted(market_campaign_dataset['EDUCATION'].apply(lambda x: ordinal_values.get(x)) , reverse=True)
```

Fig. 19: EDUCATION Transformation Code

2.3.5 HOUSEHOLD_SIZE

HOUSEHOLD_SIZE is an exploratory variable and is transformed into ordinal numbers based on number of rooms. The code to achieve this is observed in Fig. 20.

```
ordinal_values_household={ '1' : 0 , '2' : 1 , '3' : 2 , '4-5' : 3 , '6-8' : 4 , '9+' : 5 }


# Map the integer values to ordinal values based on the USA education level in descending order
market_campaign_dataset['HOUSEHOLD_SIZE'] = sorted(market_campaign_dataset['HOUSEHOLD_SIZE'].apply(lambda x: ordinal_values_household.get(x)) , reverse=True)
```

Fig. 20: **HOUSEHOLD_SIZE** Transformation Code

The outcome after transformation can be observed in Fig. 21.

	index	CUST_ID	CUST_GENDER	AGE	CUST_MARITAL_STATUS	COUNTRY_NAME	EDUCATION	OCCUPATION	HOUSEHOLD_SIZE	YRS_RESIDENCE	AFFINITY_CARD	BULK_PACK_DISKETTES	FLAT_PANEL_MONITOR	HOME_THEATER_PACKAGE	BOOKKEEPING_APPLICATION	Y_BOX_GAMES	COMMENTS	CUST_INCOME_LEVEL
0	0	101501	0	41	NeverM	17	15	Prof.	5	4	0	1	1	1	1	0	Shopping at your store is a hassle. I rarely s...	7.0
1	1	101502	1	27	NeverM	17	15	Sales	5	3	0	1	1	0	1	1	Affinity card is great. I think it is a hassle...	6.0
2	2	101503	0	20	NeverM	17	15	Cleric.	5	2	0	1	0	0	1	1	I purchased a new computer recently, but the m...	5.0
3	3	101504	1	45	Married	17	15	Exec.	5	5	1	0	0	1	1	0	Affinity card is great. I think it is a hassle...	1.0
4	4	101505	1	34	NeverM	17	15	Sales	5	5	1	1	1	0	1	0	Why didn't you start a program like this befor...	8.0

Fig. 21: Transformation Output

3. Data Analysis

In order to prepare the data for modeling, data analysis involves checking them for relevance. The mean, variance, and covariance with other variables are only a few examples of the various properties of data that can be evaluated. The picture of how to use the data most effectively is formed as a result of everything. Once more, Python and its libraries focused on data science make it possible to accomplish this with little effort on the side of the programmer.

3.1 Summary Statistics

The sum, mean, and standard deviation are calculated using the Pandas describe() function (Fig. 22). Instead of calling the procedures individually, pass the skew and kurt to retrieve Statistics Summary, Mean, Std, Skew, and Kurtosis, and combine them into a single Pandas data frame. The output it generates is seen in Fig. 23.

```
# summary statistics of sum, mean, standard deviation
Statistics_Summary=market_campaign_dataset.describe()
```

```
skewness=market_campaign_dataset.skew()
kurtosis=market_campaign_dataset.kurt()
Statistics_Summary.loc['skewness']=skewness
Statistics_Summary.loc['kurtosis']=kurtosis
```

Fig. 22: Summary Statistics Code

	index	CUST_ID	CUST_GENDER	AGE	COUNTRY_NAME	EDUCATION	HOUSEHOLD_SIZE	YRS_RESIDENCE	AFFINITY_CARD	BULK_PACK_DISKETTES	FLAT_PANEL_MONITOR	HOME_THEATER_PACKAGE	BOOKKEEPING_APPLICATION	Y_BOX_GAMES	CUST_INCOME_LEVEL
count	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000
mean	755.255656	102256.255656	0.686275	38.679487	1.245852	9.298643	1.918552	4.125943	0.263952	0.641026	0.596531	0.575415	0.880090	0.282805	5.466063
std	436.081457	436.081457	0.464181	13.142837	3.977660	2.595168	1.388145	1.877768	0.440940	0.479881	0.490778	0.494466	0.324978	0.450533	2.603042
min	0.000000	101501.000000	0.000000	17.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	377.250000	101878.250000	0.000000	28.000000	0.000000	8.000000	1.000000	3.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	3.000000
50%	758.500000	102259.500000	1.000000	37.000000	0.000000	9.000000	2.000000	4.000000	0.000000	1.000000	1.000000	1.000000	1.000000	0.000000	6.000000
75%	1131.750000	102632.750000	1.000000	47.000000	0.000000	12.000000	2.000000	5.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	7.000000
max	1499.000000	103000.000000	1.000000	90.000000	17.000000	15.000000	5.000000	14.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	9.000000
skewness	-0.023569	-0.023569	-0.803806	0.576199	3.146027	-0.296162	0.886886	0.790575	1.072277	-0.588641	-0.393973	-0.305496	-2.342709	0.965625	-0.351884
kurtosis	-1.212856	-1.212856	-1.355943	0.085536	8.555609	0.703659	0.352932	1.764425	-0.851508	-1.656002	-1.847574	-1.909555	3.493550	-1.069184	-1.157511

Fig. 23: Summary Statistics Output

3.2 Correlation Data

Obtaining variable correlation values shows how correlated negatively or positively one variable is to another. This can then be used to inform the analyst as to the most important features in the modeling stage.

3.2.1 Correlation Data Table

The code in Fig. 24 shows the correlation of each variable with the target variable like Affinity Card, how to produce the table output in Fig. 25.

```
market_campaign_dataset_corr=market_campaign_dataset.corr()['AFFINITY_CARD'].sort_values()
print(market_campaign_dataset_corr)
```

Fig. 24: Feature Correlation Matrix Code

```
Y_BOX_GAMES                -0.281063
CUST_MARITAL_STATUS        -0.101260
CUST_ID                    -0.034864
index                      -0.034864
FLAT_PANEL_MONITOR         -0.027153
CUST_INCOME_LEVEL          -0.027040
BULK_PACK_DISKETTES        -0.015547
COMMENTS                   -0.000172
OCCUPATION                  0.010677
COUNTRY_NAME                0.029240
HOUSEHOLD_SIZE              0.042548
EDUCATION                   0.048460
BOOKKEEPING_APPLICATION     0.173639
CUST_GENDER                 0.231582
AGE                         0.249937
HOME_THEATER_PACKAGE        0.282478
YRS_RESIDENCE               0.360886
AFFINITY_CARD               1.000000
Name: AFFINITY_CARD, dtype: float64
```

Fig. 25: Feature Correlation Matrix Output

3.2.2 Visualisation of Correlation Data

A "heatmap" is another technique to illustrate how features are correlated; the code to do this is shown in Fig. 26 and results in the plot seen in Fig. 27.

```
plt.figure(figsize=(15,10))
sns.heatmap(market_campaign_dataset.corr() , annot=True , cmap="YlGnBu")
plt.show()
```

Fig. 26: Feature Correlation Heat Map Code

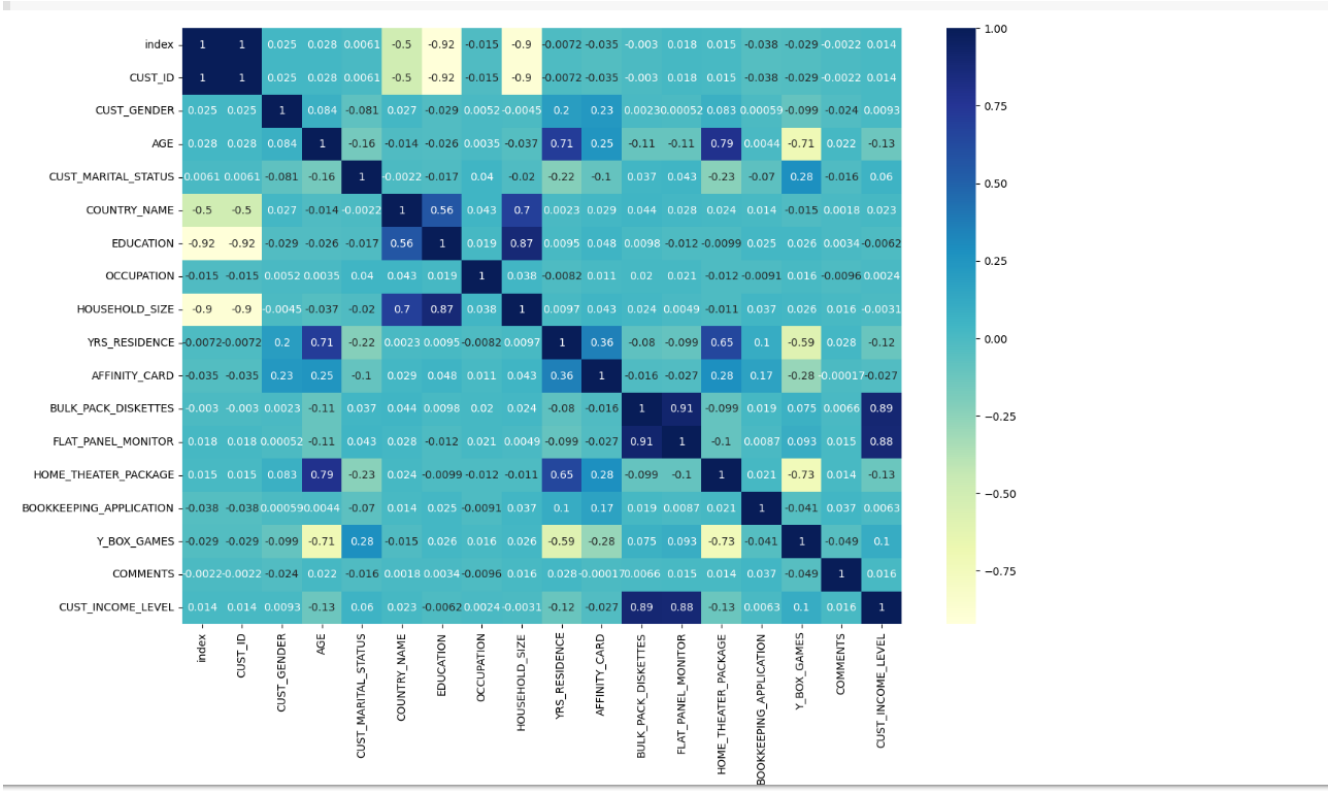

Fig. 27: Feature Correlation Heat Map Output

4. Data Exploration

One way to investigate data is to plot it. The specification specifies that a histogram and scatter plot be produced, and it offers the user the ability to select the variable from the dataframe of Market Campaign Dataset that will be shown up. And plot the Histogram and Scatter Plot.The while loop is used by the plotting functions to allow the user to enter commands into an input box. help improve the end-user experience, interpret user commands for a variety of tasks. Common tasks are;

- x_fig_size (n) - this allows the size of the plot to be altered along the x axis.
- y_fig_size (n) - this allows the size of the plot to be altered along the y axis.
- plot - this plots the currently chose variable or variables for the type of plot being used.
- show- this displays the plot's functionality.
- Title- this display the Name of Graphs

The code for the histogram plot can be seen in Fig. 29 and for the scatter plot in Fig. 31.

4.1 Histogram Plot

Histograms are used to show the relative amount of values for a given variable and can be used for Histograms can be used for discrete, continuous, or categorical data and are used to display the relative abundance of values for a specific variable. The seaborn countplot was used for the histogram plotting function in order to properly display the data. It features a few unique commands, such as column_values, which lists the possible values to plot as an ordinal dictionary and enables the user to choose a variable fast by number rather than name. The user can select the variable to plot in column (n) based on its ordinal value rather than name.

Data can now transition between being converted (i.e., from original values) and being cleaned (i.e., filtered). Particularly helpful when nominal data changes.

4.1.1 Plot the Histogram of any user chosen variables:

By using the input( ) function , user enter the name of variable in Market Campaign Dataset as you seen in Fig. 28 below.

```
# User enter the variable name
variable_name = input("Enter the name of the variable to create a histogram of: ")
```

Fig . 28 Input Variable Enter Code

By using the hist plot function to plot the histogram of input variables in Market Campaign Dataset on different ranges.Fig. 29 and describe the output in fig. 30.As you seen in Fig.29, some common Commands are used :

1. plt :command allows you to create and customize plots in a simple and intuitive way.
2. xlabel : Show the Label on x-axis like AGE
3. ylabel : Show the Label on y-axis like count or frequency
4. title : Display the name of the Histogram Graph.
5. show: Display the Graph

```
# Create the histogram
market_campaign_dataset[variable_name].hist(color='#86bf91', zorder=2, rwidth=0.9)

# Set the title and axis labels
plt.title("Histogram of " + variable_name)
plt.xlabel(variable_name)
plt.ylabel("Frequency")

# Show the histogram
plt.show()
```

Fig. 29 Histogram of Variables Code

```
Text(0.5, 1.0, 'Age Of Empolyers in  Marketing Campaign Dataset')
```

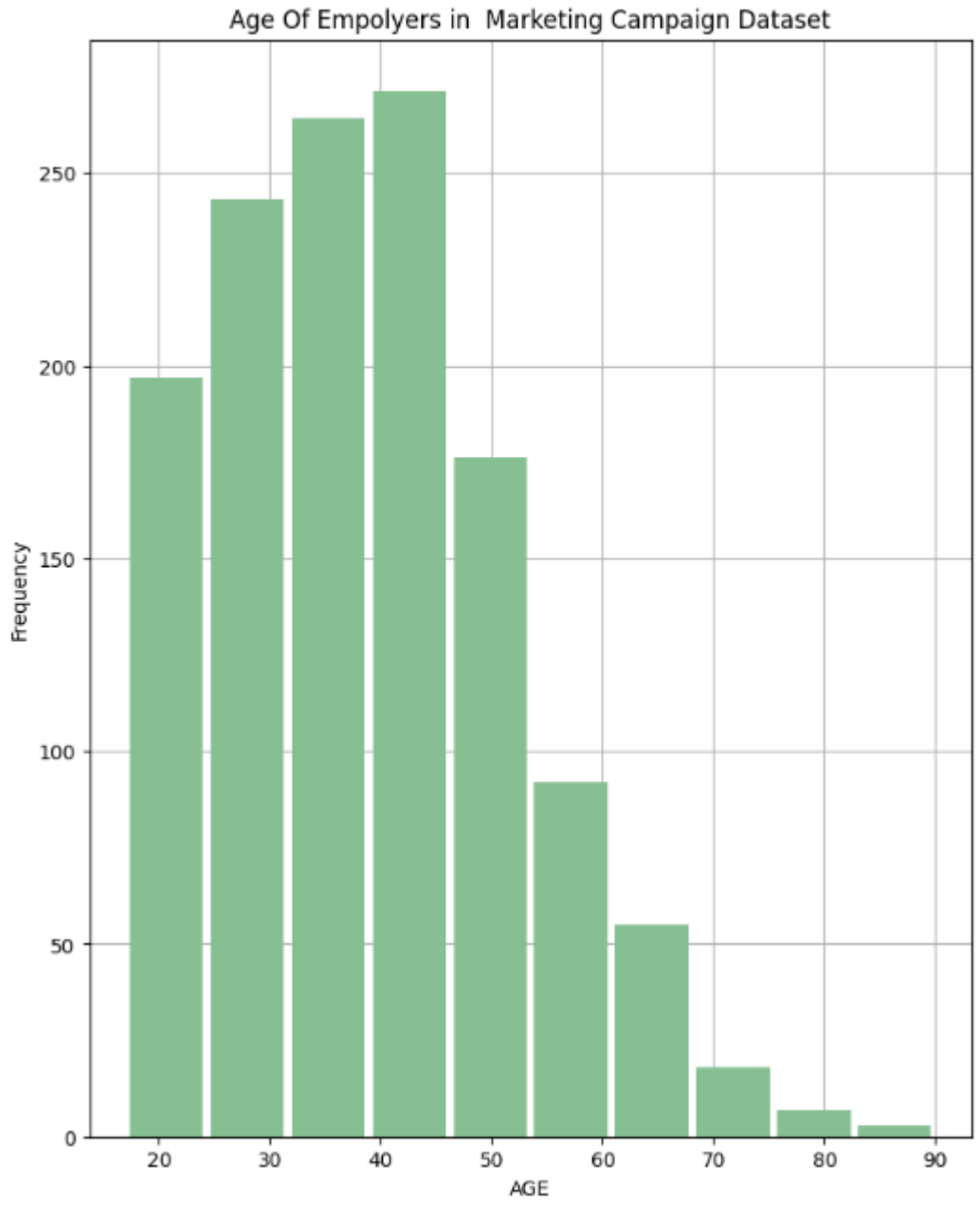

Fig. 30: Histogram Graph

4.2 Scatter Plot

To display the correlations between two variables, utilize scatter plots. This is only achievable with discrete or continuous data, therefore it only applies to modified data. In order to provide the end user with the most functionality possible, the seaborn scatterplot is utilized, which does just that—shows intensity at many occuring intersections of values. Even then, a conventional scatter plot wouldn't provide intensity of values for a particular intersection of values. The scatter plot functionality also has unique commands namely;

- xy_values - lists the available values to plot as an ordinal dictionary, allowing the user to quickly choose the variable by number and not name.
- x (n) - set the variable to plot on the x axis given its place in the ordinal dictionary.
- y (n) - set the variable to plot on the y axis given its place in the

  ordinal dictionary.

4.2.1 Plot the Scatter Graph Between Two Variables :

By using the input( ) function, user enter the name of two variables; one of them may be a target variable in Market Campaign Pandas Dataframe as you observe in Fig. 31 below.

```
# User enter the variable name
Input_variable_name = input("Enter the name of the variable to create a Scatter Plot of: ")
# User enter the Target Variable name
Target_variable_name = input("Enter the name of the variable to create a Scatter Plot of: ")
```

Fig. 31 User Input Variable Enter

Using the scatter function to plot a scatter graph between two variables one of them may be a target variable like Affinity Card and the other variable depends upon the Variable, User enters into the Input Field. as observed in Fig. 32.

```
market_campaign_dataset.plot.scatter(x = Input_variable_name , y = Target_variable_name , color='#FF69B4')
# Set the title and axis labels
plt.title('Scatter Plot of ' + Input_variable_name + ' vs ' + Target_variable_name)
plt.xlabel(Input_variable_name)
plt.ylabel(Target_variable_name)

# Show the scatter plot
plt.show()
```

Fig. 32: Scatter Plot code

```
Text(0.5, 1.0, 'Scatter Plot of Affinity_ Card vs Age')
```

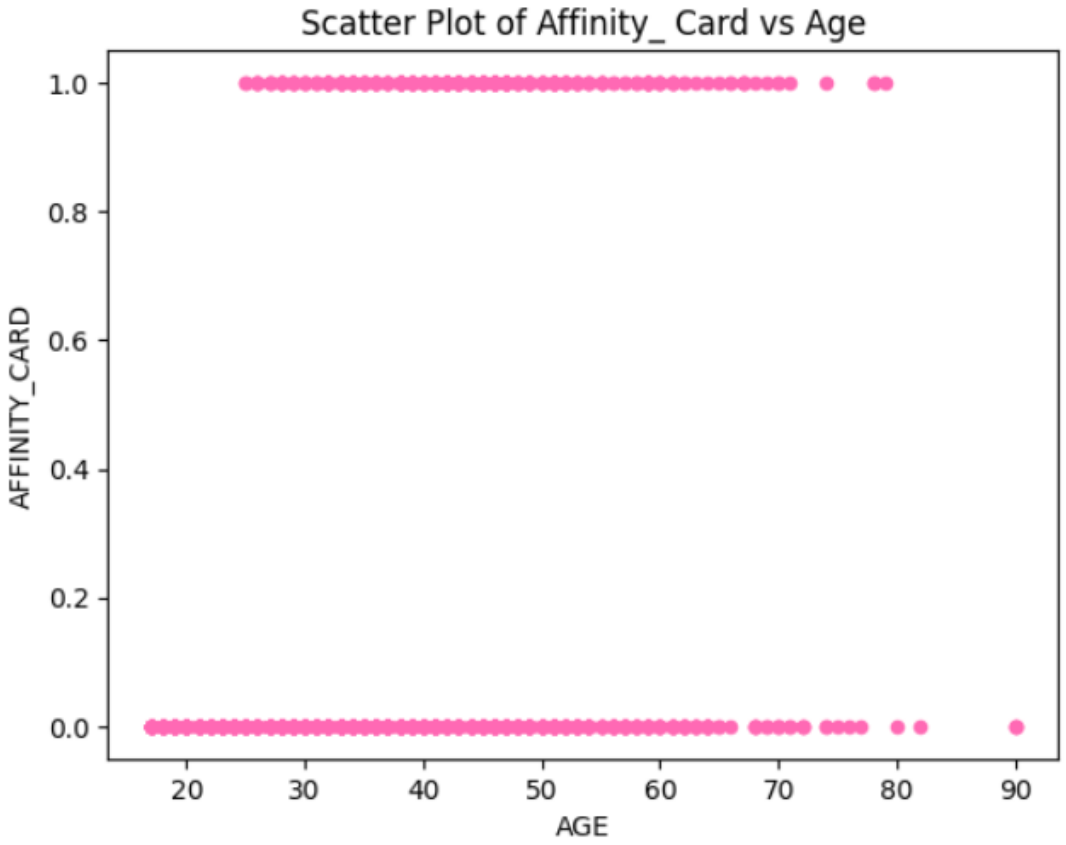

Figure 33: Scatter Plot Output

5. Data Mining

Python data mining is strongly supported by the scikit-learn libraries. Decision Tree Classifier and Random Forest Classifier are the models that were employed in this investigation. In both situations, Python functions were developed to encourage reuse for various features and data. Each model type generates metrics for accuracy, the importance of features, a model score

The best sampling strategy for each model was determined by maximizing the TN (True Negative or predicted low value of affinity card and actual No Affinity card value) rate and minimizing the FP (False Positive rate or predicted High value of Affinity Card). Along with making sure these requirements are met, it was determined that a maximum training accuracy—defined as the value acquired from the unseen test data after training the models with the data (oversampled or not)—was important. With a random state of None, the data was divided into 85% training data and 25% testing data.

The function code can be seen in the accompanying .ipynb file and are named *Random Forest Classification* Fig. 34 and *decision_tree_model* Fig. 35 respectively. Both models are supervised learning models when a target variable *AFFINITY_CARD* was predicted with various transformations.

Fig. 34: Random Forest Modelling Code

Decesion Tree Classification

```
[93] dt_model=tree.DecisionTreeClassifier()
     dt_model=dt_model.fit(X_train , y_train)
```

```
     y_pred=dt_model.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print("Accuracy:", accuracy)

     Accuracy: 0.7620481927710844
```

```
[95] grid=GridSearchCV(dt_model , param_grid={ "criterion" : ['gini' , 'entropy'] , "max_depth" : range(1,10) , "min_samples_split" : range(1,10) ,  "min_samples_leaf" : range(1,5)} , cv=10 , verbose=1 , n_jobs=10)
     grid_dt_model=grid.fit(X_train , y_train)
```

```
[96] y_pred = grid_dt_model.predict(X_test)
     training_accuracy=grid_dt_model.best_score_
     print('training_accuracy=' ,training_accuracy )
     print(f' confusion matrix:')
     print(confusion_matrix(y_test, y_pred))
     print(f' classification report:')
     print(classification_report(y_test, y_pred))
```

```
[97]
     # Calculate the accuracy of the model on the test data
     test_accuracy = accuracy_score(y_test, y_pred)

     # Print the test accuracy
     print(f"Test accuracy: {test_accuracy}")

     Test accuracy: 0.7710843373493976
```

```
[98] y_score1 = grid_dt_model.predict_proba(X_test)[:,1]
     false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(y_test, y_score1)
```

```
[99] print('roc_auc_score for DecisionTree: ', roc_auc_score(y_test, y_score1))

     roc_auc_score for DecisionTree:  0.818798449612403
```

```
[100] plt.subplots(1, figsize=(10,10))
      plt.title('Receiver Operating Characteristic - DecisionTree')
      plt.plot(false_positive_rate1, true_positive_rate1)
      plt.plot([0, 1], ls="--")
      plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
      plt.ylabel('True Positive Rate')
      plt.xlabel('False Positive Rate')
      plt.show()
```

Fig. 35: Decision Tree Classification Modelling Code

5.1 Predictive Model 1 - Random Forest Classification Model

Random Forest is a supervised learning algorithm used for classification and regression related problems. To train the model on the Target Variable :

- Affinity Card (high value =1 or low value =0)

Other Training Variables :

- CUST_GENDER
- AGE
- CUST_MARITAL_STATUS
- COUNTRY_NAME
- CUST_INCOME_LEVEL
- OCCUPATION
- YRS_RESIDENCE,
- BULK_PACK_DISKETTES
- FLAT_PANEL_MONITOR
- HOME_THEATER_PACKAGE
- BOOKKEEPING_APPLICATION
- PRINTER_SUPPLIES
- Y_BOX_GAMES
- OS_DOC_SET_KANJI

The output for the Random Forest Model can be observed in Fig. 36 It can be observed that;

- training data accuracy is 79% and test data accuracy is 80%
- Model Score, how well the model is at predicting the outcome from unseen data is 80%.
- Weighted averages for Precision, recall and f1-score and all >70% which means the model is a good one.

▾ Random Forest Model

```
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
y_pred=rfc.predict(X_test)
# evaluate the model
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(rfc, x, y, scoring='accuracy', cv=cv, n_jobs=-1, error_score='raise')
# report performance
print('Training Accuracy: %.3f (%.3f)' % (np.mean(n_scores), np.std(n_scores)))
print(f' confusion matrix:')
print(confusion_matrix(y_test, y_pred))
print(f' classification report:')
print(classification_report(y_test, y_pred))
```

```
[103] accuracy = accuracy_score(y_test, y_pred)
      print("Testing Accuracy:", accuracy)

      Testing Accuracy: 0.8012048192771084
```

```
[104] y_score2 = rfc.predict_proba(X_test)[:,1]
      false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(y_test, y_score2)
```

```
[105] print('roc_auc_score for DecisionTree: ', roc_auc_score(y_test, y_score2))

      roc_auc_score for DecisionTree:  0.8222726413310644
```

Fig. 36 : Random Forest Model Code

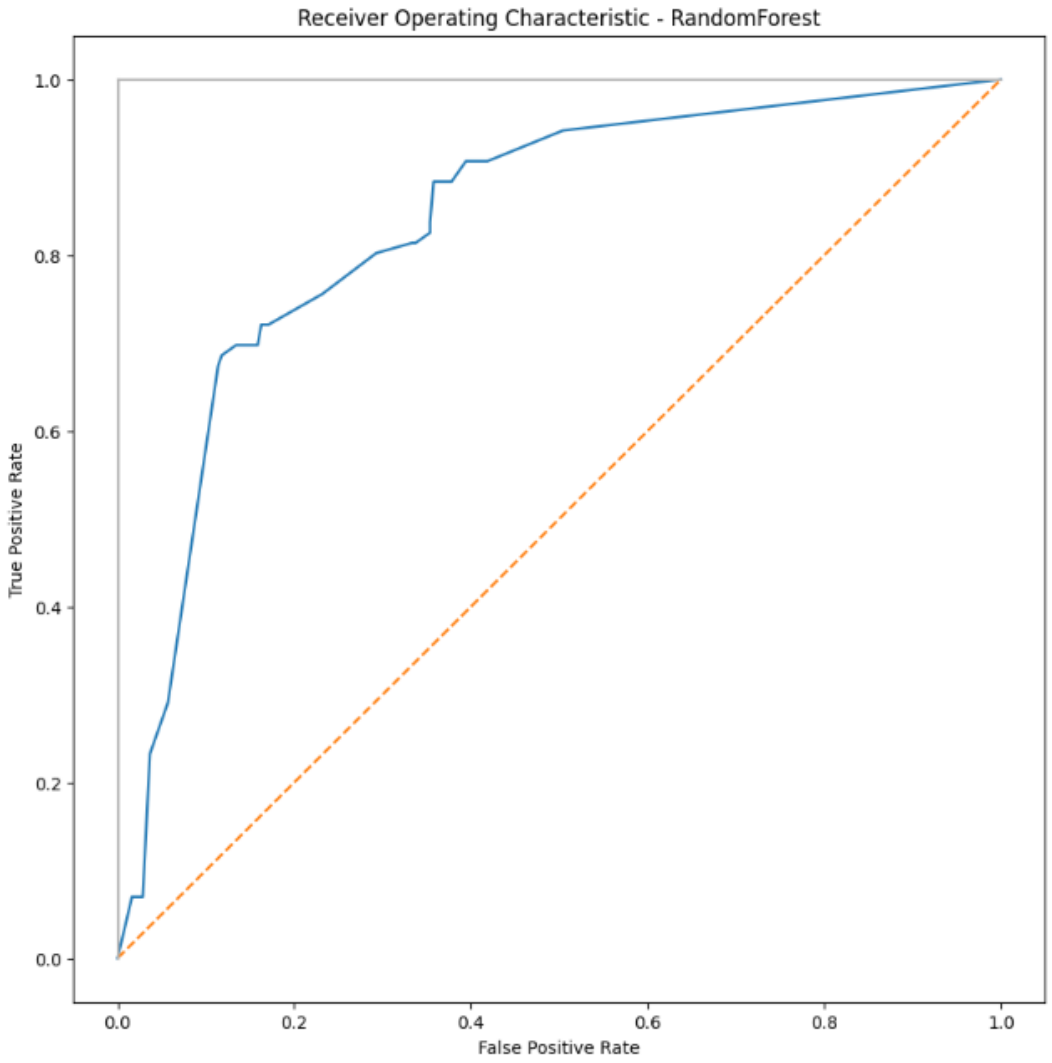

Fig. 37 ROC Random Forest Classification Output

5.2 Predictive Model 2 - Decision Tree Classification

The second supervised model to be used was the Decision Tree Classifier. The decision tree model predicts the answer class for each of the predictors using a recursive, partition-based classifier. consisting of leaf nodes, which indicate areas or partitions of the data space that are labeled with the majority class, and interior nodes, which represent judgements representing the hyperplanes or split points.

To train the model on the Target Variable :

- Affinity Card (high value =1 or low value =0)

Other Training Variables :

- CUST_GENDER
- AGE
- CUST_MARITAL_STATUS
- COUNTRY_NAME
- CUST_INCOME_LEVEL
- OCCUPATION
- YRS_RESIDENCE,
- BULK_PACK_DISKETTES
- FLAT_PANEL_MONITOR
- HOME_THEATER_PACKAGE
- BOOKKEEPING_APPLICATION
- PRINTER_SUPPLIES
- Y_BOX_GAMES
- OS_DOC_SET_KANJI

The output for the Decision Tree Classifier can be observed in Fig. 38 It can be observed that;

- training data accuracy is 80% and test data accuracy is 83%
- Model Score, how well the model is at predicting the outcome from unseen data is 83%.
- Weighted averages for Precision, recall and f1-score and all >90% which means the model is a good one.

Decesion Tree Classification

```
[93] dt_model=tree.DecisionTreeClassifier()
     dt_model=dt_model.fit(X_train , y_train)

     y_pred=dt_model.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print("Accuracy:", accuracy)

     Accuracy: 0.7620481927710844

[95] grid=GridSearchCV(dt_model , param_grid={ "criterion" : ['gini' , 'entropy'] , "max_depth" : range(1,10) , "min_samples_split" : range(1,10) ,  "min_samples_leaf" : range(1,5)} , cv=10 , verbose=1 , n_jobs=10)
     grid_dt_model=grid.fit(X_train , y_train)

[96] y_pred = grid_dt_model.predict(X_test)
     training_accuracy=grid_dt_model.best_score_
     print('training_accuracy=' ,training_accuracy )
     print(f' confusion matrix:')
     print(confusion_matrix(y_test, y_pred))
     print(f' classification report:')
     print(classification_report(y_test, y_pred))

[97]
     # Calculate the accuracy of the model on the test data
     test_accuracy = accuracy_score(y_test, y_pred)

     # Print the test accuracy
     print(f"Test accuracy: {test_accuracy}")

     Test accuracy: 0.7710843373493976

[98] y_score1 = grid_dt_model.predict_proba(X_test)[:,1]
     false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(y_test, y_score1)

[99] print('roc_auc_score for DecisionTree: ', roc_auc_score(y_test, y_score1))

     roc_auc_score for DecisionTree:  0.818798449612403

[100] plt.subplots(1, figsize=(10,10))
      plt.title('Receiver Operating Characteristic - DecisionTree')
      plt.plot(false_positive_rate1, true_positive_rate1)
      plt.plot([0, 1], ls="--")
      plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
      plt.ylabel('True Positive Rate')
      plt.xlabel('False Positive Rate')
      plt.show()
```

Fig. 38 Decision Tree Classification Code

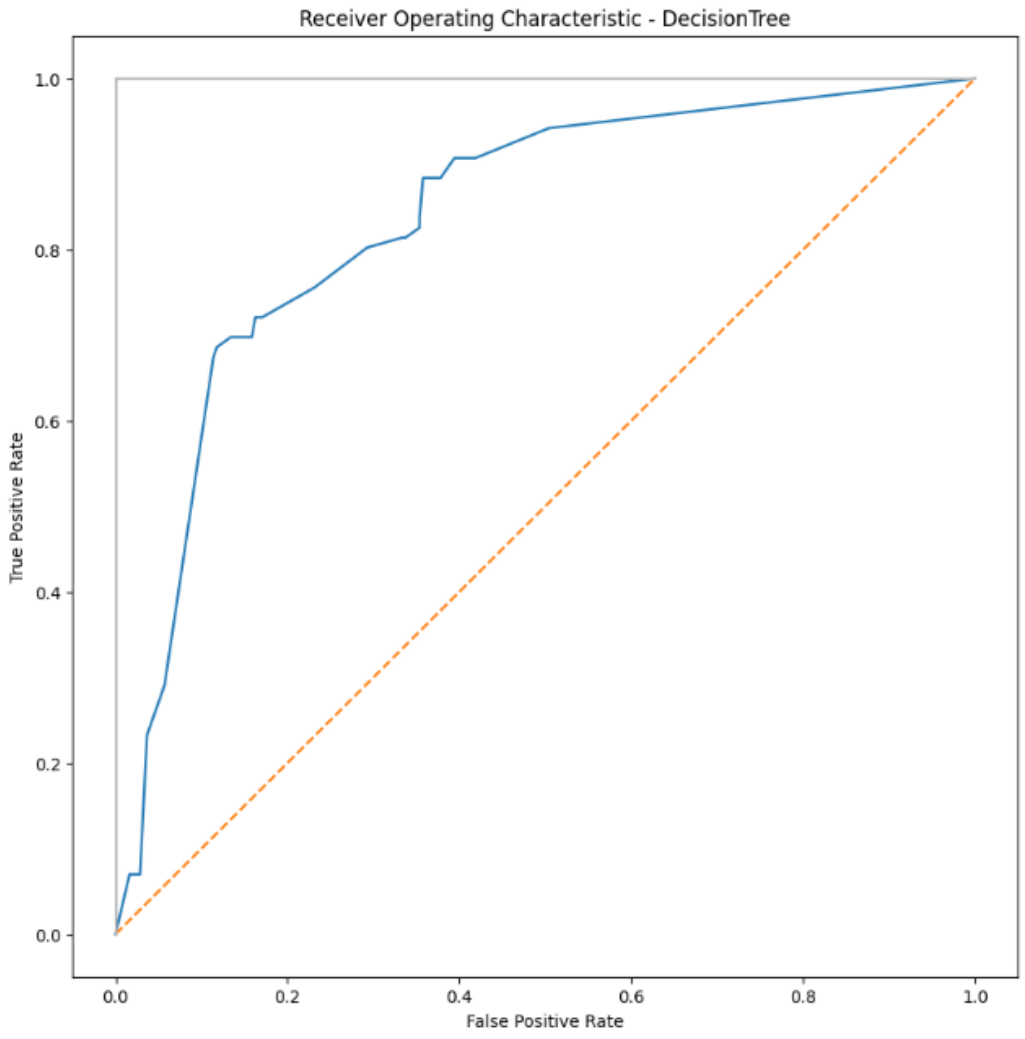

Fig. 39 ROC Decision Tree Classification Output

5.3 Comparison of Models

After models have been developed, it is critical to evaluate their performance in comparison to the data, other models, the resources at hand, and random chance. ROC Curves, Cumulative Gain, and Lift Curve are a few popular data analysis charts that can be used to accomplish this. We shall just use ROC Curves.

5.3.1 ROC Curves

The ROC (Receiver Characteristic Curve), together with an AUC (Area Under the Curve) value (a score of 1.0 signifies a perfect prediction), provides an indicator of how well the model will perform. The ROC shows the true positive rate versus the false positive rate for various threshold values. Figure 37 shows the code to generate a ROC Curve with an AUC for RF and DT, and Figure 39 shows the results. As can be observed, RF has a higher AUC value (0.85 vs. 0.84 for DT) than DT. Although neither score is a flawless 1, they are both higher than a random chance score of 0.5.

```
# calculate the false positive rate, true positive rate, and threshold for the Decision Tree Classifier
roc_auc_dt = auc(false_positive_rate1, true_positive_rate1)


# calculate the false positive rate, true positive rate, and threshold for the Random Forest Classifier
roc_auc_rf = auc(false_positive_rate2, true_positive_rate2)

# plot the ROC curves for both classifiers
plt.plot(false_positive_rate1, true_positive_rate1, color='red', label='Decision Tree (AUC = %0.2f)' % roc_auc_dt)
plt.plot(false_positive_rate2, true_positive_rate2, color='blue', label='Random Forest (AUC = %0.2f)' % roc_auc_rf)

# add labels and legend
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')

# show the plot
plt.show()
```

Fig. 40 : ROC AUC Curve Code

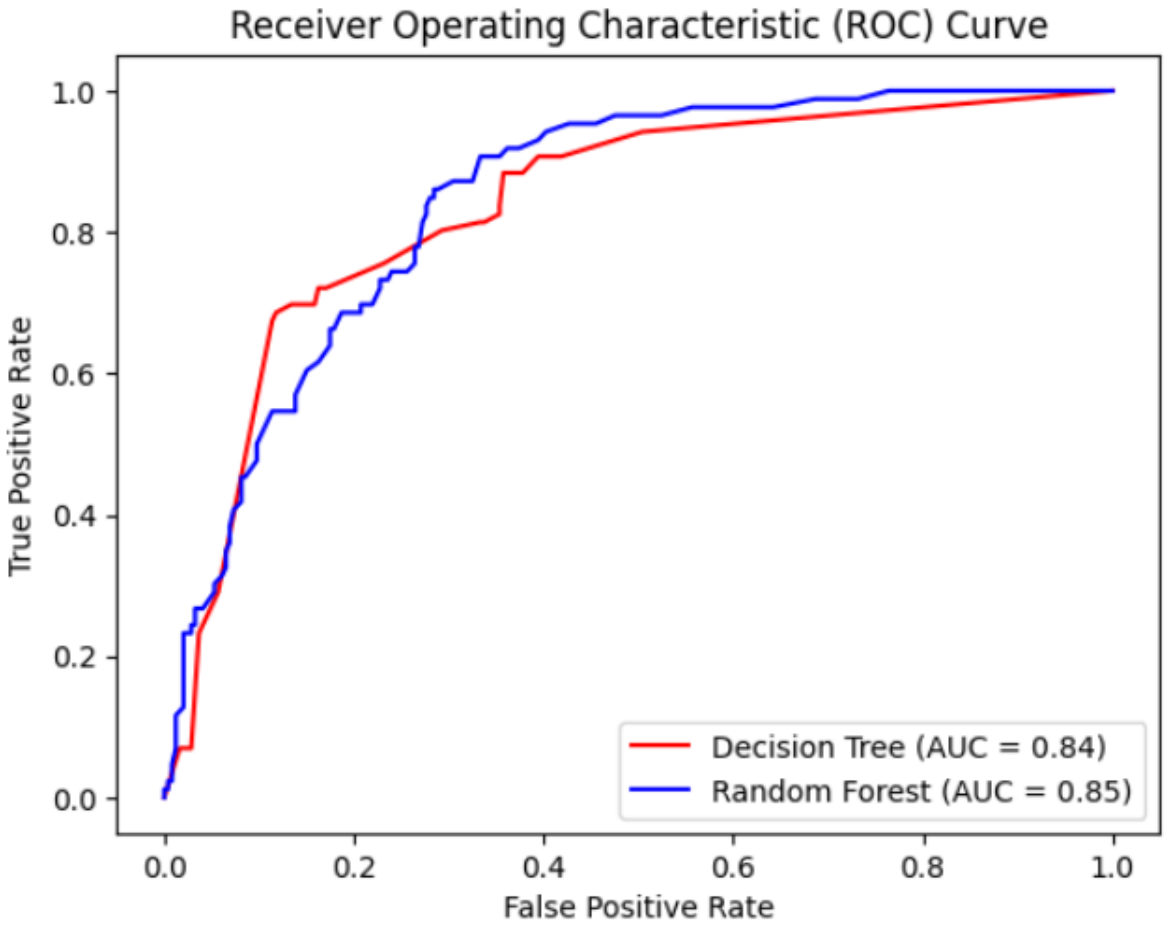

Fig. 41 ROC AUC Curve Output

6. Discussion and Reflection

Without understanding how to programme in Python and showcasing that proficiency, learning about contemporary data analysis and machine learning would fall short. Due to its numerous libraries covering this topic, Python is widely utilised in this industry. It is a crucial tool for contemporary data analysts because it processes data inside of other applications like Power Bi and Hadoop. The capabilities of libraries like scikit-learn and pandas enable data analysts to build feature-rich, accurate code quickly.

Compared to other modern languages such as C++ and Java it has some interesting idiosyncrasies (strict formatting and lack of true multi-threading, Python implements a version of multi-threading) but these are trivial compared to that wealth of supporting libraries and brevity in creating code. e analysis task itself, the variables to be included were in the specification (presumably to benchmark the results), these being;

- CUST_ID
- CUST_GENDER
- AGE
- CUST_MARITAL_STATUS
- COUNTRY_NAME
- CUST_INCOME_LEVEL
- EDUCATION
- OCCUPATION
- HOUSEHOLD_SIZE
- YRS_RESIDENCE
- AFFINITY_CARD
- BULK_PACK_DISKETTES
- FLAT_PANEL_MONITOR
- HOME_THEATER_PACKAGE
- BOOKKEEPING_APPLICATION
- PRINTER_SUPPLIES
- Y_BOX_GAMES
- OS_DOC_SET_KANJI
- COMMENTS

After removing missing or null or Blank variables the variables leave :

- CUST_GENDER
- AGE
- CUST_MARITAL_STATUS
- COUNTRY_NAME
- CUST_INCOME_LEVEL
- OCCUPATION
- YRS_RESIDENCE
- AFFINITY_CARD
- BULK_PACK_DISKETTES
- FLAT_PANEL_MONITOR
- HOME_THEATER_PACKAGE
- BOOKKEEPING_APPLICATION
- Y_BOX_GAMES
- OS_DOC_SET_KANJI

It would be worth adding these features as predictors to see what (if any) influence they have on the outcome. The code presented in this report has been structured (as a workflow and using functions for the modeling process) to allow this to be achieved with minimum effort. They weren’t included

in this report because they weren’t explicitly requested by the stakeholder.

7 Appendices

7.1 Appendix A

Various sampling strategies were used for the Random Forest model and Decision Tree Classifier as observed in Fig. 42. Random Forest Classifier Model shows better result then Decision Tree Classifier. As shown in Compare Fig. 42 RF shows better results than DT.

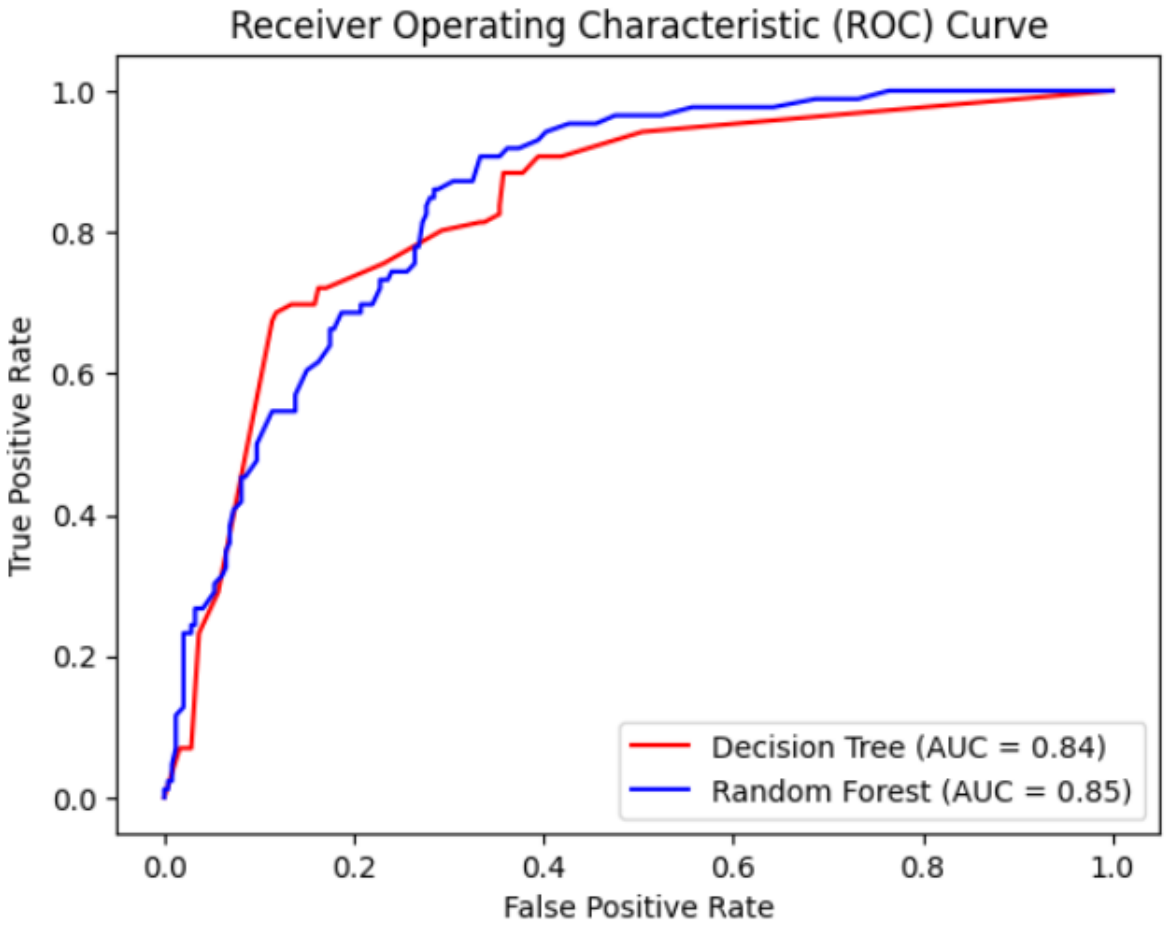

Fig. 42 : Compare the AUC models of DF and RF.

8 . References

1. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
3. https://www.jmp.com/en_be/statistics-knowledge-portal/exploratory-data-analysis/scatter-plot.html
4. https://www.tibco.com/reference-center/what-is-a-histogram-chart