**APPENDIX**

**Load the required Python Libraries and read data into Python**

**# Give the location of the file**

path = "cust\_seg.csv"

#Load the required Python Libraries

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

import numpy as np

import matplotlib.pyplot as plt

import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)

import seaborn as sns

sns.set()

**#Read data as dataframes in Python using the Pandas Library.**

cust\_seg = pd.read\_csv(path)

**#View data**

cust\_seg.head()

EDA

# Show data information

cust\_seg.info()

cust\_seg = cust\_seg.replace("NA", np.nan)

cust\_seg["age"] = cust\_seg["age"].apply(lambda string: string.strip())

cust\_seg["antiguedad"] = cust\_seg["antiguedad"].apply(lambda string: string.strip())

cust\_seg["age"] = cust\_seg["age"].replace("NA", np.nan)

cust\_seg["antiguedad"] = cust\_seg["antiguedad"].replace("NA", np.nan)

#Show missing data

cust\_seg.isnull().sum()

# Show duplicated Rows

duplicated\_rows = cust\_seg[cust\_seg.duplicated()]

print('The number of duplicated rows = ', duplicated\_rows.shape[0])

# Show description of the dataset

round(cust\_seg.describe(include='all'),2)

#Show correlation between columns pf the dataset

cust\_seg.corr()

SOLVE FOR MISSING DATA

# Check for outliers

plt.figure(figsize=(17,9))

plt.subplot(1,2,1)

sns.boxplot(data = cust\_seg, y = 'sexo', x = "renta", dodge=False)

plt.subplot(1,2,2)

sns.boxplot(data = cust\_seg, y = 'age', x = "indrel\_1mes", dodge=False)

plt.tight\_layout()

plt.show()

Replacing With Mean

This is the most common method of imputing missing values of numeric columns. If there are outliers then the mean will not be appropriate. In such cases, outliers need to be treated first.

renta and indrel\_1mes features are numeric columns with outliers,hence replacing with mean will not be appropriate.

Replacing With Mode

Mode is the most frequently occurring value. It is used in the case of categorical features.

You can use the ‘fillna’ method for imputing the categorical columns ‘sexo’, ‘ind\_empleado’,‘pais\_residencia’,'tiprel\_1mes','indresi','indext','canal\_entrada','indfall','nomprov','conyuemp'.

cust\_seg['sexo'] = cust\_seg['sexo'].fillna(cust\_seg['sexo'].mode()[0])

cust\_seg['ind\_empleado'] = cust\_seg['ind\_empleado'].fillna(cust\_seg['ind\_empleado'].mode()[0])

cust\_seg['pais\_residencia'] = cust\_seg['pais\_residencia'].fillna(cust\_seg['pais\_residencia'].mode()[0])

cust\_seg['tiprel\_1mes'] = cust\_seg['tiprel\_1mes'].fillna(cust\_seg['tiprel\_1mes'].mode()[0])

cust\_seg['indresi'] = cust\_seg['indresi'].fillna(cust\_seg['indresi'].mode()[0])

cust\_seg['indext'] = cust\_seg['indext'].fillna(cust\_seg['indext'].mode()[0])

cust\_seg['canal\_entrada'] = cust\_seg['canal\_entrada'].fillna(cust\_seg['canal\_entrada'].mode()[0])

cust\_seg['indfall'] = cust\_seg['indfall'].fillna(cust\_seg['indfall'].mode()[0])

cust\_seg['nomprov'] = cust\_seg['nomprov'].fillna(cust\_seg['nomprov'].mode()[0])

cust\_seg['conyuemp'] = cust\_seg['conyuemp'].fillna(cust\_seg['conyuemp'].mode()[0])

cust\_seg['age'] = cust\_seg['age'].fillna(cust\_seg['age'].mode()[0])

cust\_seg['antiguedad'] = cust\_seg['antiguedad'].fillna(cust\_seg['antiguedad'].mode()[0])

cust\_seg['fecha\_alta'] = cust\_seg['fecha\_alta'].fillna(cust\_seg['fecha\_alta'].mode()[0])

cust\_seg['fecha\_dato'] = cust\_seg['fecha\_dato'].fillna(cust\_seg['fecha\_dato'].mode()[0])

cust\_seg['fecha\_alta'] = cust\_seg['fecha\_alta'].fillna(cust\_seg['fecha\_alta'].mode()[0])

cust\_seg['ult\_fec\_cli\_1t'] = cust\_seg['ult\_fec\_cli\_1t'].fillna(cust\_seg['ult\_fec\_cli\_1t'].mode()[0])

Replacing With Median

Median is the middlemost value. It’s better to use the median value for imputation in the case of outliers.

You can use ‘fillna’ method for imputing the column ‘renta’,'ind\_nomina\_ult1','ind\_nom\_pens\_ult1','ind\_nuevo','indrel','indrel\_1mes','tipodom','cod\_prov','ind\_actividad\_cliente' with the median value.

cust\_seg['renta'] = cust\_seg['renta'].fillna(cust\_seg['renta'].median())

cust\_seg['ind\_nomina\_ult1'] = cust\_seg['ind\_nomina\_ult1'].fillna(cust\_seg['ind\_nomina\_ult1'].median())

cust\_seg['ind\_nom\_pens\_ult1'] = cust\_seg['ind\_nom\_pens\_ult1'].fillna(cust\_seg['ind\_nom\_pens\_ult1'].median())

cust\_seg['ind\_nuevo'] = cust\_seg['ind\_nuevo'].fillna(cust\_seg['ind\_nuevo'].median())

cust\_seg['indrel'] = cust\_seg['indrel'].fillna(cust\_seg['indrel'].median())

cust\_seg['indrel\_1mes'] = cust\_seg['indrel\_1mes'].fillna(cust\_seg['indrel\_1mes'].median())

cust\_seg['tipodom'] = cust\_seg['tipodom'].fillna(cust\_seg['tipodom'].median())

cust\_seg['cod\_prov'] = cust\_seg['cod\_prov'].fillna(cust\_seg['cod\_prov'].median())

cust\_seg['ind\_actividad\_cliente'] = cust\_seg['ind\_actividad\_cliente'].fillna(cust\_seg['ind\_actividad\_cliente'].median())

#Solve for data type, convert object data type to date format

from datetime import datetime, timedelta

cust\_seg['fecha\_alta'] = pd.to\_datetime(cust\_seg['fecha\_alta'])

cust\_seg['fecha\_dato'] = pd.to\_datetime(cust\_seg['fecha\_dato'])

cust\_seg['ult\_fec\_cli\_1t'] = pd.to\_datetime(cust\_seg['ult\_fec\_cli\_1t'])

cust\_seg['fecha\_alta']

cust\_seg['fecha\_dato']

#Convert age (object) to int

cust\_seg['age'] = cust\_seg['age'].astype(int)

# Convert antiguedad(object) to int

cust\_seg['antiguedad'] = cust\_seg['antiguedad'].astype(int)

#Verify data type after data type conversions

cust\_seg.info()

Feature Analysis

fecha\_dato(Date)

In [28]:

plt.figure(figsize**=**(17,9))

sns.histplot(data**=**cust\_seg, x**=**"fecha\_dato",bins**=**40 , color**=** 'y')

plt.title('Date Distribution', fontsize**=**20)

plt.ylabel('Frequency')

plt.xlabel('Customer Dates')

ncodpers(Customer Codes)

plt.figure(figsize**=**(17,9))

sns.histplot(data**=**cust\_seg, x**=**"ncodpers",bins**=**40 , color**=** 'y')

plt.title('Cusomer Code Distribution', fontsize**=**20)

plt.ylabel('Frequency')

plt.xlabel('Customer Codes')

ind\_empleado(Employee Index)

plt.figure(figsize=(17,9))

sns.countplot(data=cust\_seg, x="ind\_empleado", palette='rocket')

plt.title("Employee Index Countplot", fontsize=20)

plt.ylabel('Frequency')

plt.xlabel('Employee Index')

pais\_residencia(Customer's country residence)

plt.figure(figsize=(22,11))

sns.histplot(data=cust\_seg, x="pais\_residencia",bins=40 , color= 'g')

plt.xticks(rotation=90)

plt.title('Customers country residence Distribution', fontsize=20)

plt.ylabel('Frequency')

plt.xlabel('Customers country residence')

sexo(Customer's Sex)

plt.figure(figsize=(17,9))

sns.countplot(data=cust\_seg, x="sexo", palette='rocket')

plt.title("Customers Sex Countplot", fontsize=20)

plt.ylabel('Frequency')

plt.xlabel('Customers sex')

Age

plt.figure(figsize=(17,9))

sns.histplot(data=cust\_seg, x='age', bins=47)

plt.title('Age Distribution', fontsize=25)

plt.ylabel('Frequency')

plt.xlabel('Age')

antiguedad

plt.figure(figsize=(17,9))

sns.histplot(data=cust\_seg, x="antiguedad", bins=40,color ='g')

plt.title('Customer Seniority Distribution', fontsize=20)

plt.xticks(rotation=90)

plt.ylabel('Frequency')

plt.xlabel('Customer Seniority in Months')

indrel\_1mes(Customer Type)

plt.figure(figsize=(17,9))

sns.countplot(data=cust\_seg, x="indrel\_1mes", palette='rocket')

plt.title("Customer type Countplot", fontsize=20)

plt.ylabel('Frequency')

plt.xlabel('Customers type')

indresi

plt.figure(figsize=(17,9))

sns.countplot(data=cust\_seg, x="indresi", palette='rocket')

plt.title("Residence Index Countplot", fontsize=20)

plt.ylabel('Frequency')

plt.xlabel('Residence Index')

nomprov

plt.figure(figsize=(22,11))

sns.histplot(data=cust\_seg, x="nomprov",bins=40 , color= 'g')

plt.xticks(rotation=90)

plt.title('Customers province Distribution', fontsize=20)

plt.ylabel('Frequency')

plt.xlabel('Customers Province')

'ind\_ahor\_fin\_ult1' vs ind\_cco\_fin\_ult1(Current vs Saving Account)

g=sns.catplot('ind\_ahor\_fin\_ult1','ind\_cco\_fin\_ult1',data=cust\_seg,kind='bar',palette = 'rocket', height=8.27, aspect=18/8.27)

ax = g.facet\_axis(0,0)

for p in ax.patches:

ax.text(p.get\_x() + 0.015,

p.get\_height() \* 1.02,

"{:.0f}".format(p.get\_height()),

color='black', rotation='horizontal', size='medium')

plt.ylabel('currents', fontsize=16)

plt.xlabel('Savings', fontsize=16)

plt.title('Current VS Savings Account',fontsize=20)

plt.xticks(rotation=45)

Time Series

Here I want to create a new dataset that will be the same as Master Datset, but the feature 'fecha\_dato' will be it's index so as to work properly Time Series Visualizations

cust\_seg\_t=cust\_seg.set\_index('fecha\_dato')

cust\_seg\_t.sort\_values('fecha\_dato').head()

**Here we shall create a model based on the clustering principle.**

K-means clustering is an unsupervised clustering algorithm and that it belongs to the non-hierarchical class of clustering algorithms.

Hence we use the K-means clustering as it also addresses unnamed columns which are classified as unsupervised(unlabelled)column. The model was a comparison lessons from the following websites:

<https://towardsdatascience.com/customer-segmentation-in-python-9c15acf6f945>

<https://www.data-mania.com/blog/customer-profiling-and-segmentation-in-python/>

STEPS: Gather the data, Create Recency Frequency Monetary (RFM) table, Manage skewness and scale each variable, Explore the data, Cluster the data, Interpret the result

**Gathering Data**

#Install Python Libraries required for the model

# ! pip install xlrd

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)

import seaborn as sns

sns.set()

#Define the path to read data from

path = "cust\_seg\_transformed.csv"

# Read csv file in dataframe

df = pd.read\_csv(path)

df = df[df['ncodpers'].notna()]

df\_fix = df.sample(10000, random\_state = 42)

df\_fix.shape

df\_fix.head()

### **Data Preprocessing**

#### **Create RFM Table**

In this section, I will group the customers into different classes. In the case of matrices with binary encoding, the most suitable metric for the calculation of distances is the Hamming's metric. Note that the kmeans method of sklearn uses a Euclidean distance that can be used, but it is not to the best choice in the case of categorical variables. However, in order to use the Hamming's metric, we need to use the kmodes package which is not available on the current plateform. Hence, I will make an RFM Table which are numeric.

To make the RFM table, we can create these columns, such as Recency, Frequency, and MonetaryValue column.

To get the recency column, we can use the ind\_nuevo(New customer Index. 1 if the customer registered in the last 6 months)

To create the frequency column, we can use the ind\_actividad\_cliente(Activity index (1, active customer; 0, inactive customer)

Lastly, to create the monetary value column, we can sum all deposits(short,medium and large )for each customer.

Data Preprocessing Create RFM Table

# Create TotalSum colummn

df\_fix["TotalSum"] = df\_fix["ind\_deco\_fin\_ult1"] + df\_fix["ind\_deme\_fin\_ult1"] + df\_fix["ind\_dela\_fin\_ult1"]

# Aggregate data by each customer

customers = df\_fix.groupby(['ncodpers']).agg({

'ind\_nuevo': 'count',

'ind\_actividad\_cliente': 'count',

'TotalSum': 'sum'})

# Rename columns

customers.rename(columns = {'ind\_nuevo': 'Recency',

'ind\_actividad\_cliente': 'Frequency',

'TotalSum': 'MonetaryValue'}, inplace=True)

Manage Skewness

fig, ax = plt.subplots(1, 3, figsize=(15,3))

sns.distplot(customers['Recency'], ax=ax[0])

sns.distplot(customers['Frequency'], ax=ax[1])

sns.distplot(customers['MonetaryValue'], ax=ax[2])

plt.tight\_layout()

plt.show()