DataGlacier Week 9 Deliverables
Project: Cross Selling Recommendations
Group Name: MacroStaff
Group Specialization: Data Analyst

Group Members:

1. ### Wasiq Ahmed:
   Email: wasiqahmed7@gmail.com

   Country: Turkey

   University: Bilkent University

2. ### Samuel Bailey
   Email: sjbrsa@gmail.com

   Country: South Africa

   University: University of Cape Town

Problem Description

A company called XYZ Credit Union currently has data about its customers and their purchased banking products. Since this data is highly clustered and not properly formatted, the company needs assistance in cleaning the data of potential invalid or repeated entries.

Github Repository

https://github.com/Wasiq147/DataGlacier-proejct.git

Data Cleaning and Transformation (Method 1)

Data Cleaning with Python on 'Test.csv'

```
import pandas as pd

df = pd.read_csv (r'/content/drive/MyDrive/Test.csv', header=0)
df.dropna(subset = ["ind_empleado"], inplace=True) #remove rows with blank entries in Employee Index
df.dropna(subset = ["canal_entrada"], inplace=True) #remove rows with blank entries in Channels Used by Customers to Join
df.dropna(subset = ["cod_prov"], inplace=True) #remove rows with blank entries in Province Code
df.dropna(subset = ["segmento"], inplace=True) #remove rows with blank entries in Customer Segmentation

df["tiprel_1mes"].replace({"": "R"}, inplace=True)
#In Customer Relation Type, all relations are mentioned except Potential Customers, which were set as blank entries, now changed
df["conyuemp"].replace({"N": "1"}, inplace=True)
#In Spouse Index, all clients that have a spouse are labelled correctly, as per data type description

df['fecha_alta'] = pd.to_datetime(df.fecha_alta)
df['fecha_alta'] = df['fecha_alta'].dt.strftime('%d-%m-%Y') #All date entries are formatted using the same date format

df.drop(df[df["antiguedad"] < 1].index, inplace=True) #All entries that list negative customer seniority are removed

df.drop(df[df["age"] < 18].index, inplace=True)
df.drop(df[df["age"] > 116].index, inplace=True)
#All clients listed as having ages lower than 18 or higher than 116 are eliminated
```

Data Cleaning with Python on 'Train.csv'

```
import pandas as pd

df = pd.read_csv(r'/content/drive/MyDrive/Train.csv', header = 0)
df.dropna(subset = ["ind_empleado"], inplace=True) #remove rows with blank entries in Employee Index
df.dropna(subset = ["pais_residencia"], inplace=True) #remove rows with blank entries in Customer's Country Residence
df.dropna(subset = ["sexo"], inplace=True) #remove rows with blank entries in Customer's Sex
df["age"] = df.drop(df[df["age"] == "NA"].index, inplace=True) #remove rows where Customer Age was written as NA

df.drop(df[df["age"] < 18].index, inplace=True)
df.drop(df[df["age"] > 116].index, inplace=True)
#All clients listed as having ages lower than 18 or higher than 116 are eliminated

df.dropna(subset = ["ind_nuevo"], inplace=True) #remove rows with blank entries in Customer Indices

df["antiguedad"] = pd.to_numeric(df["antiguedad"])
df.drop(df[df["antiguedad"] < 0].index, inplace=True) #All entries that list negative Customer Seniority are removed
df.drop(df[df["antiguedad"] == "NA"].index, inplace=True) #remove rows where Customer Seniority was written as NA
df["indrel"] = df.drop(df[df["indrel"] == "NA"].index, inplace=True) #remove rows where Customer Primary status was written as NA

df.dropna(subset = ["tiprel_1mes"], inplace=True) #remove rows with blank entries in Customer Type
df.dropna(subset = ["indresi"], inplace=True) #remove rows with blank entries in Residence Index
df.dropna(subset = ["indext"], inplace=True) #remove rows with blank entries in Foreigner Index
df.dropna(subset = ["canal_entrada"], inplace=True) #remove rows with blank entries in Channls Used by Customers to Join
df.dropna(subset = ["indfall"], inplace=True) #remove rows with blank entries in Deceased Index

df["tipodom"] = df.drop(df[df["tipodom"] == "NA"].index, inplace=True) #remove rows where Customer Address Type was written as NA
df.dropna(subset = ["nomprov"], inplace=True) #remove ros with blank entries in Province Names
df["ind_actividad_cliente"] = df.drop(df[df["ind_actividad_cliente"] == "NA"].index, inplace=True) #remove rows where Customer Activity
df.dropna(subset = ["segmento"], inplace=True) #remove rows with blank entries in Customer Segmentation

df.dropna(subset = ["fecha_alta"], inplace=True) #remove rows with blank entries for when customers became holders at bank

df['fecha_alta'] = pd.to_datetime(df.fecha_alta)
df['fecha_alta'] = df['fecha_alta'].dt.strftime('%d-%m-%Y') #All date entries are formatted using the same date format
```

Data Cleaning and Transformation (Method 2)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Store the paths to the two files containing the data
path_train = 'drive/MyDrive/Cross Selling/Train.csv'
path_test = 'drive/MyDrive/Cross Selling/Test.csv'

# Load the data into two data frames containing the data from each file
df1 = pd.read_csv(path_train, header=0)
df2 = pd.read_csv(path_test, header=0)

# Combine (not join) the data frames
df = pd.concat([df1, df2], axis=0)
#%%
# Print the number of missing values for each variable
df2.isna().sum()
```

```
#%%
# The following command returns an empty data frame
df[df['indrel'] == 99][df['ult_fec_cli_1t'].isna()]
# This is because where the value of indrel is 1 the variable 'ult_fec_cli_1t'
# can have no value by its definition, and so it really has no missing values.

# Next we consider the variables indrel_1mes and tiprel_1mes
df[df["indrel_1mes"].isna()]
df[df["tiprel_1mes"].isna()]
# and impute their modes for their missing values
df['indrel_1mes'].fillna(df['indrel_1mes'].mode()[0], inplace = True)
df['tiprel_1mes'].fillna(df['tiprel_1mes'].mode()[0], inplace = True)

# We impute the variable for income simply using the overall mean
df['renta'].fillna(df['renta'].mean(), inplace = True)

# Looking the sex variable, the missing values could represent people who
# refused to declare their gender or who don't identify as either male or
# female. It is better not to impute but simply to label these observations X
df['sexo'].fillna('X', inplace = True)

# Considering missing values of the variable conyuemp indicating marriages
# between customers and employees:
df[df['conyuemp'] == 'S']
# There is a large number of missing values just in the test data, but of the
# non-missing values, the customers who are spouses of employees appear to be
# extremely uncommon (only 1 in the test data), so we impute the mode value to this variable
df['conyuemp'].fillna(df['conyuemp'].mode()[0], inplace = True)

# For the small number of missing values in the categorical variable
# canal_entrada we use mode imputation again
df['canal_entrada'].fillna(df['canal_entrada'].mode()[0], inplace = True)

# For the cod_prov and nomprov variables, simple imputation of the modes would
# be inappropriate and we use hot-deck imputation in this case
df.fillna(method='ffill', inplace=True)
df2[df2['segmento'] == '01 - TOP']
```