# PROG8421 - Programming for Big Data

**Final Project**

**Group 1 Members:**

|  |  |
| --- | --- |
| **Name: Sudhan Shrestha** | **Name: Jose Encarnacion** |
| **Student ID: 8889436** | **Student ID: 8982860** |

**Part I: Data Overview**

***Prerequisites***

**Loading the libraries**

**Code**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

import datetime

warnings.filterwarnings("ignore")

**Comment**

Importing all the required libraries on the first cell of the notebook.

**Screenshot**

**A screen shot of a computer program

Description automatically generated**

1. **Use the pandas library to read the data file and to create the data frame.**

**Code**

data\_path ='customers.csv' # setting the path to the data file

df = pd.read\_csv(data\_path, delimiter='\t') # reading the csv file, with delimiter as tab since data are separated by tabs

df

**Comment**

Reading the data from the csv file using read\_csv() function we used delimiter as “\t” as the values in the csv files were separated by tab.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Display the first 5 rows and the last 3 rows of your data.**

**Code**

df.head() # first 5 rows of the dataframe

df.tail(3) # last 3 rows of the dataframe

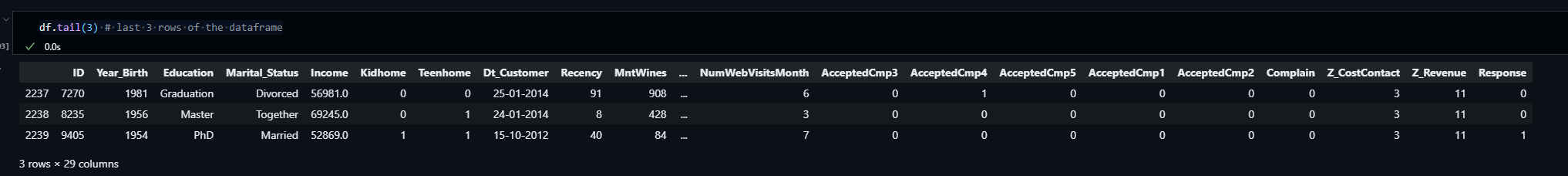
**Comment**

Using the head() and tail(3) to display the first five rows and the last 3 rows of the dataframe.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

****

1. **Show quick statistics of your data**

**Code**

df.describe()

**Comment**

The function describe() was used to show the quick statistical information about the numeric data in the dataframe.

**Screenshot**

**A screenshot of a computer screen

Description automatically generated**

1. **Show the data type of each column**

**Code**

print("Data types of all columns:")

df.dtypes

**Comment**

The function dtypes is used to display the datatypes of all the columns of the dataframe.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Show how many columns and rows in your data.**

**Code**

rows = df.shape[0] # number of rows in the dataframe

columns = df.shape[1] # number of columns in the dataframe

print(f"Number of rows: {rows}")

print(f"Number of columns: {columns}")

**Comment**

The function shape was used to show the numbers of rows and columns in the dataframe by passing the respective parameters 0 for rows and 1 for columns.

**Screenshot**

**A screenshot of a computer code

Description automatically generated**

1. **Show the list of columns in your data frame.**

**Code**

columns\_list = df.columns.tolist() # convert the column names to a list

print("List of columns:")

columns\_list

**Comment**

The function df.columns is first used to list all the columns of the dataframe then tolist() function is used to convert it as a list and then we display all the columns in our dataframe.

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **Show the number of duplicated rows in your data.**

**Code**

duplicate\_count = df.duplicated().sum() # count the number of duplicate rows

print(f"The number of duplicate rows is: {duplicate\_count}")

**Comment**

The function duplicated() was used to get the duplicated values in the dataframe and sum() function was used on top of it to get the total sum of these duplicated values.

**Screenshot**

**A screen shot of a computer code

Description automatically generated**

**Part II: Data Preparation and Cleaning**

1. **Rename the columns belonging to Products by removing Mnt from them.**
   1. **You should have: 'Wines','Fruits','Meat','Fish','Sweet','Gold'**

**Code**

columns\_to\_rename = ['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']

new\_column\_names = ['Wines', 'Fruits', 'Meat', 'Fish', 'Sweet', 'Gold']

dfm = df.copy() # creating a copy of the original dataframe

dfm.rename(columns=dict(zip(columns\_to\_rename, new\_column\_names)), inplace=True) # renaming the columns

dfm[new\_column\_names].head()

**Comment**

A list of the column names to be replaced and the new names were created then we created a copy of the dataframe and used the rename() function by passing the original name and the names to replace with in the dataframe. The inplace() function is used to make the changes affect the dataframe itself.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Rename the columnsbelonging to Place by removing Num and Purchases from them.** 
   1. **You should have: 'Web ','Catalog', 'Store'**

**Code**

columns\_to\_rename = ['NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases']

new\_column\_names = ['Web', 'Catalog', 'Store']  # Corrected column names needed to rename

dfm.rename(columns=dict(zip(columns\_to\_rename, new\_column\_names)), inplace=True)

dfm[new\_column\_names].head()

**Comment**

We renamed the columns here similarly as done in question 8 by utilizing the rename() function.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Keep only NumDealsPurchases column in the Promotion section**

**Code**

# Removing all columns from the promotion except the 'NumDealsPurchases' column

columns\_to\_remove = ['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5','Response']

dfm.drop(columns=columns\_to\_remove, inplace=True)

dfm.columns

**Comment**

Removing all the columns from the Promotion section except ‘NumDealsPurchase”. The function drop() was used while passing the list of columns to remove and inplace() was set to True to set the changes in the dataframe directly.

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **Delete the column NumWebVisitsMonth**

**Code**

columns\_to\_delete = ['NumWebVisitsMonth']

dfm['NumWebVisitsMonth'].head()

dfm.drop(columns=columns\_to\_delete, inplace=True)

# dfm['NumWebVisitsMonth'].head()

**Comment**

The column ‘NumWebVisitsMonths” was deleted as requestion using the drop() function.

**Screenshot**

**A screen shot of a computer code

Description automatically generated**

1. **The attribute Year\_Birth is not helpful to segment your data. Instead create a new derived column named Age.**

**Code**

current\_year = datetime.datetime.now().year # getting the current year

dfm['Age'] = current\_year - dfm['Year\_Birth'] # calculating the age (current year - year of birth)

dfm['Age'].head() # displaying the first 5 rows of the 'Age' column

**Comment**

Here we created a new column named ‘Age’ where we got the current year using the datetime module and subtracted it to the ‘Year\_Birth’ column to get the Age.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Create a new column named Spending that holds the total amount spent on all products categories.**

**Code**

dfm['Spending'] = dfm['Wines'] + dfm['Fruits'] + dfm['Meat'] + dfm['Fish'] + dfm['Sweet'] + dfm['Gold']

dfm['Spending'].head()

**Comment**

A new column named ‘Spending’ was created by summing the values in the columns ‘Wines’, ‘Fruits’, ‘Meat’, ‘Fish’, ‘Sweet’ and ‘Gold’.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Change the Dt\_Customer type from object to datetime. Use the format YYYY-MM-DD.**

**Code**

# Convert the 'Dt\_Customer' column to datetime format

dfm['Dt\_Customer'] = pd.to\_datetime(dfm['Dt\_Customer'], format='%d-%m-%Y')

dfm['Dt\_Customer'].head()

**Comment**

The format of the date for the Dt\_customer was changed to YYYY-MM-DD format using the to\_datetime() function.

**Screenshot**

**A screen shot of a computer program

Description automatically generated**

1. **Show the unique values of the column data Marital\_Status.**

**Code**

print("The unique values in the 'Marital\_Status' column are:")

print(dfm['Marital\_Status'].unique())

**Comment**

The unique values of the Maritial\_Status column was displayed using the unique() function to get those values.

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **It is better to work with lower categorical values. Hence, we’ll classify customers in two segments for their martial status: change the column Marital status as follows:** 
   1. **Change the values: Divorced, Single, Absurd and Widow to 'Alone'**
   2. **Change Married and Together to 'Not Alone'**

**Code**

new\_values = {

    'Divorced': 'Alone',

    'Single': 'Alone',

    'Absurd': 'Alone',

    'Widow': 'Alone',

    'Married': 'Not Alone',

    'Together': 'Not Alone'

}

dfm['Marital\_Status'] = dfm['Marital\_Status'].replace(new\_values)

dfm['Marital\_Status'].head()

**Comment**

The values for the column ‘Marital\_Status’ were changes to ‘Alone’ for Divorced, Single, Absurd and Widow and ‘Not Alone’ for Married and Together values. The replace() function was used to replace these values.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Show the unique values of the column Education.**

**Code**

print("The unique values in the 'Education' column are:")

print(dfm['Education'].unique())

**Comment**

The unique values in the Education columns were displayed using the unique() function.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **We want to segment the customers in 2 groups based on their education level: change the column Education as follows:** 
   1. **Change the values: 'Basic' and '2n Cycle' to 'Undergraduate',**
   2. **All other values to 'Postgraduate'**

**Code**

columns = ['Education']

new\_values = {

    'Basic': 'Undergraduate',

    '2n Cycle': 'Undergraduate'

}

dfm['Education'] = dfm['Education'].replace(new\_values)

dfm['Education'].loc[~dfm['Education'].isin(['Undergraduate'])] = 'Postgraduate'

dfm['Education'].loc[dfm['Education'].isin(['Undergraduate'])].head()

print("The unique values in the 'Education' column after replacing values are:")

print(dfm['Education'].unique())

**Comment**

In the Education column the values were replaced by ‘Undergraduate’ for Basic and 2nCycle and the rest were replaced by ‘Postgraduate’. Using the replace() function and isin() was used to check if the values were in Undergraduate first and if not it was later replaced to ‘Postgraduate’. Then we printed out the unique values from the ‘Education’ column again to see if the data are changes.

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **Have a look at column Income. Investigate the presence of outliers? Delete them accordingly.**

**Code**

columns = ['Income']

dfm[columns].describe()

# checking for outliers in the income using boxplot

sns.boxplot(dfm['Income'], color='lightgreen')

plt.title('Income Column Boxplot')

plt.show()

# removing the outliers using IQR method

q1 = dfm['Income'].quantile(0.25)

q3 = dfm['Income'].quantile(0.75)

iqr = q3 - q1

# removing the outliers by only keeping the values that are within the IQR range

dfm = dfm[(dfm['Income'] >= q1 - 1.5 \* iqr) & (dfm['Income'] <= q3 + 1.5 \* iqr)]

sns.boxplot(dfm['Income'], color='skyblue')

plt.title('Income Column Boxplot (After Outlier Removal)')

plt.show()

**Comment**

First we looked into the data using describe() function just to get an idea of the data’s stats, then we created a box plot to visualize the data distribution. From the boxplot it was evident that there were some outliers in the data.

We removed the outliers by only keeping the data within tie IQR (Inter Quartile Range) and removing the rest hence removing the outlies and plotted the boxplot again to see the data, where the outliers were removed.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

**A white rectangular chart with green lines

Description automatically generated**

**A diagram with a blue rectangle

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

1. **Show quick statistics for the column Income.**

**Code**

columns = ['Income']

dfm[columns].describe()

**Comment**

The statistics of the ‘Income’ column were displayed using the describe() function.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Show the number of missing values for each column.**

**Code**

columns = dfm.columns

missing\_values = dfm[columns].isnull().sum()

print(missing\_values)

**Comment**

The missing values count were display by using the isnull() and sum() function together which gives the sum of missing values in each columns.

*There were no missing values, in the original dataframe there were some missing values in the income column which might have been imputed during outlier removal.*

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **Fill the missing values with the average.**

**Code**

# filling the missing values with the mean

dfm.fillna(dfm.mean(), inplace=True) # replacing the missing values with the mean

print(f"The missing values in the datafram are: \n{dfm.isnull().sum()}")

**Comment**

Even though our missing values were already removed, the code was still implemented for reporting purposes. In order to do this fillna() method is used to fill the missing values and mean() was used to replace it with the mean values of that column.

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **Create a new column named Children to hold the total number of children for every customer.**

**Code**

new\_column\_name = 'Children'

dfm[new\_column\_name] = dfm['Kidhome'] + dfm['Teenhome']

dfm[[new\_column\_name]].head()

**Comment**

A new column named ‘Children’ was created whose values was the sum of values from column ‘Kidhome’ and ‘Teenhome’ using arithmetic operation.

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **Create a new column Has\_Complaint as follows: a. 'Has complaint' if the customer complained in the last 2 years, b. Otherwise 'No complaint'**

**Code**

columns = ['Complain']

dfm['Has\_Complaint'] = dfm['Complain'].apply(lambda x: 'Has complaint' if x == 1 else 'No complaint')

dfm['Has\_Complaint'].head()

dfm['Has\_Complaint'].unique() # checking the unique values to see changes

**Comment**

A new column named ‘Has Complaint’ was created where we inserted values in this column based on the values of the ‘Complain’ column if the value there is 1 and then we insert ‘Has complaint’ else we add ‘No complaint’. This was possible using the apply() function with a asynchronous lambda function. We later also check the values in the new column using the unique() function, to display the unique values here.

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **Show the first 5 rows of customers with ‘No complaint’**

**Code**

dfm[dfm['Has\_Complaint'] == 'No complaint'].head()

**Comment**

The first five rows where the ‘Has\_Complaint’ column has value ‘No complaint’ were displayed using the head() function.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

**Part III: Data Visualization & Analytics**

1. **Let’s analyse the profile of customers based on their background factors. To do so, we will use our cleaned data frame (in part II) to create a new one with the columns: Age, Spending, Marital\_Status, Education, Income, Children, Dt\_Customer and Has\_Complaint.**

**Code**

columns\_to\_add = ['Age', 'Spending', 'Marital\_Status', 'Education', 'Income', 'Children','Dt\_Customer', 'Has\_Complaint']

df\_customer\_profile = dfm[columns\_to\_add]

df\_customer\_profile

**Comment**

A new dataframe named df\_customer\_profile was created which included only the columns: Age, Spending, Marital\_Status, Education, Income, Children, Dt\_Customer and Has\_Complaint which we will be using for visualization.

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Create histogram to show the number of children per customers in your data. What do you see?**

**Code**

# histogram to show the number of childerns per customer

sns.histplot(df\_customer\_profile['Children'],bins= range(df\_customer\_profile['Children'].min(),df\_customer\_profile['Children'].max()+1), color='#7986CB')

# creating labels and titles

plt.title('Histogram of Children per Customer')

plt.xlabel('Number of Children')

plt.ylabel('Frequency')

**Comment**

A histogram showing the frequency of number of children was created, here we set the bins to be minimum values and the maximum values +1 of the values in the ‘Children’ column for better visualization. For plot, histplot() of seaborn library was used.

**Analysis on the histogram**

* Analysis on the histogram indicates that most of the customers have 1-2 children’s with more than 1000 of them in this segment.
* The second largest group comprises of customers with 1 or fewer children comprising around 600 of them.
* Finally, around 400 of the customers fall into the group who have children between the range 2 and 3.

**Screenshot**

A graph of a number of children

Description automatically generated

**A screenshot of a computer

Description automatically generated**

1. **Create a histogram and a Boxplot to show the Income of customers. What do you find?**

**Code**

# histogram to show the income of customers

sns.histplot(df\_customer\_profile['Income'], color='#7986CB', kde=True)

plt.title('Income Distribution of Customers')

plt.xlabel('Income')

plt.ylabel('Frequency')

plt.show()

sns.boxplot(df\_customer\_profile['Income'], color='lightgreen')

plt.title('Income Distribution of Customers')

plt.xlabel('Income')

plt.show()

**Comment**

A histogram and boxplot were created to show the income of the customer, using seaborn histplot() and boxplot() to create these plot.

**Analysis on the histogram**

* It seems that the peak of the histogram is around $40000-$50000 salary range, while the major of the data distribution lies between $20000 to around $80000 range.
* The distribution of the histogram seems roughly symmetrical suggesting that the most customers have income clustered around the center distribution, although there is a slight tail around the end suggesting a minor left skewness.

**Screenshot**

**A graph of income distribution

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Analysis on the boxplot**

* It seems that the median income range of the customers is around $50000 range which aligns with what we observed in the histogram.
* The Interquartile range consists of where almost 50% of the data lies and from the boxplot it is clear that most of the customers salaries lies in the range of around $40000 to $60000.
* The whiskers of the boxplot extend quite a bit suggesting a wide range of income, but there does not seem to be any significant outlier in the data.

**Screenshot**

A green and black bar chart

Description automatically generated

**A screenshot of a computer

Description automatically generated**

1. **Create a histogram and a Boxplot to show the Spending of customers. What do you find?**

**Code**

# Histogram for Spending of Customers

sns.histplot(df\_customer\_profile['Spending'], color='#00838F', edgecolor='black', kde=True)

plt.title('Spending Distribution of Customers')

plt.xlabel('Spending')

plt.ylabel('Frequency')

plt.show()

# Boxplot for Spending of Customers

sns.boxplot(df\_customer\_profile['Spending'], color='lightgreen')

plt.title('Spending Distribution of Customers')

plt.xlabel('Spending')

plt.show()

**Comment**

Similarly, a histogram and box plot were created using seaborn, but we did an analysis on the spending of customers here.

**Analysis on histogram**

* + The histogram shows a right-skewed distribution suggesting that most of the customers spend in low amounts, with the cluster towards the left side of the plot.
  + The peak of the histogram is around the range of 0-250 indicating that most customers spend less amount around more than 800 of them.

**Screenshot**

**A graph of a number of customers

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Analysis on the boxplot:**

* + It seems that the median spending of the customers is around 400$.
  + The Interquartile range plot indicates that most of the salaries of the customers lies from around $50 to $1000.
  + The whiskers of the boxplot extend quite a bit suggesting a wide range of spendings, where there seems to be some outlier which does not necessarily needs to be delt with.

**Screenshot**

A diagram with green squares

Description automatically generated

**A screenshot of a computer

Description automatically generated**

1. **Create a histogram and a boxplot to analyse the customers based on their Age.**

**Code**

# Histogram for Age of Customers

sns.histplot(df\_customer\_profile['Age'], color='#00838F',bins=35)

plt.title('Age Distribution of Customers')

plt.xlabel('Age')

plt.ylabel('Frequency')

plt.show()

# Boxplot for Age of Customers

sns.boxplot(df\_customer\_profile['Age'], color='lightgreen')

plt.title('Age Distribution of Customers')

plt.xlabel('Age')

plt.show()

**Comment**

Similarly, a histogram and boxplot using the seaborn library were plotted for the analysis of customers based on their age.

**Analysis of the histogram**

* + The distribution of the roughly symmetric, with a slight right-skewness indicating that most of the customers are in the younger age group between 40-60.
  + The peak of the histogram suggests that most of the customers are around the age of 50 with almost 300 of them.
  + The slight tail around the end suggests that there are some customers that are more than 80 and above with a few outliers indicating people above the age of 120.

**Screenshot**

**A graph of a number of customers

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**Analysis of boxplot**

* Similar to what is observed in the histogram, the median age seems to be around 50.
* The IQR suggests that most of the customer segment lies between the age of around 40 - 65 range.
* There are some outliers present in the data which show the age of few customers more than 120 years which needs to be delt with if the data is used for relevant analysis.

**Screenshot**

A chart with green and black lines

Description automatically generated

**A screenshot of a computer

Description automatically generated**

1. **Create a histogram to analyse the Education level of the company’s customers. What do you find?** 
   1. **The column Education is defined as Object and has 2 values (Undergraduate, Postgraduate). You need to shape the values as numeric to be able to plot the histogram**

**Code**

# histogram to show the education level of the company’s customers

sns.histplot(df\_customer\_profile['Education'],color='teal')

plt.title("Histogram of Education level of the Company's Customer")

plt.xlabel('Education Level')

plt.ylabel('Frequency')

plt.show()

# # convert to categorical data to numeric

df\_customer\_profile['Education'] = df\_customer\_profile['Education'].map({'Postgraduate': 1, 'Undergraduate': 0})

sns.histplot(df\_customer\_profile['Education'],color='teal', bins=5)

plt.title("Histogram of Education level of the Company's Customer (After converting to numeric)")

plt.xlabel('Education Level')

plt.ylabel('Frequency')

plt.show()

**Comment**

We first created a histogram of the Education of the customer, then we changed the Education data which was categorical into numerical by using the map() and changed them to 1 for Postgraduate and 0 for Undergraduate as required in the question.

**Analysis on the histogram**

* + Based on this analysis it is clear that majority of the customers have an education level of 'Postgraduate' with around 1900 of them, while that with 'Undergraduate' level of education is comparatively less with around 250 of them.

**Screenshot**

A graph of a graph with a blue bar

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

**Analysis on the histogram**

* The data remains the same we just converted the 'Postgraduate' level of education to 1 which seems the same as the histogram above with around 1900 of them in this level and the ‘Undergraduate' level of education is casted as 0 which is still the same around 250 of them.

**Screenshot**

**A graph of a graph

Description automatically generated with medium confidence**

A screenshot of a computer

Description automatically generated