Assignment 1 Task 1

Name: Elroy Chua Ming Xuan **UOW ID:** 7431673 **Data Set:** Customer Churn Dataset https://www.kaggle.com/datasets/muhammadshahidazeem/customer-churn-dataset

Step 1: Create one Pandas dataframe from the two CSV files

In order to merge the training and test data, first, we will load both data sets using Pandas, and then we will concatenate them into a single dataframe. Assume that we have two files, train.csv and test.csv, from the given link. Below is the Python code to perform this task.

In [13]:
```
#importing libraries
import pandas as pd
# Load the train_train_train_data
test_data = pd.read_csv('customer_churn_dataset-testing-master.csv')
train_data = pd.read_csv('customer_churn_dataset-training-master.csv')

# Concatenate the dataframes
df = pd.concat([test_data, train_data], ignore_index=True)

#Data
print("Data")
# Inspect the first few rows of the dataset
df.head()

# Identify missing values
print(df.isnull().sum())
```

```
Data
   CustomerID   Age  Gender  Tenure  Usage Frequency  Support Calls  \
0         2.0  30.0  Female    39.0             14.0            5.0
1         3.0  65.0  Female    49.0              1.0           10.0
2         4.0  55.0  Female    14.0              4.0            6.0
3         5.0  58.0    Male    38.0             21.0            7.0
4         6.0  23.0    Male    32.0             20.0            5.0

   Payment Delay Subscription Type Contract Length  Total Spend  \
0           18.0          Standard          Annual        932.0
1            8.0             Basic         Monthly        557.0
2           18.0             Basic       Quarterly        185.0
3            7.0          Standard         Monthly        396.0
4            8.0             Basic         Monthly        617.0

   Last Interaction  Churn
0              17.0    1.0
1               6.0    1.0
2               3.0    1.0
3              29.0    1.0
4              20.0    1.0
CustomerID           1
Age                  1
Gender               1
Tenure               1
Usage Frequency      1
Support Calls        1
Payment Delay        1
Subscription Type    1
Contract Length      1
Total Spend          1
Last Interaction     1
Churn                1
dtype: int64
```

Step 2: Once missing values are identified, remove the missing values from the dataset.

Missing data can be identified with the isna() function in Pandas (refer to previous cell). To clean missing values, we can use various imputation methods like mean, median, mode imputation, or more advanced techniques like using regression models. For this case we will use .dropna() to remove the missing values from the dataset as it is a row of missing values.

In [14]:
```
# If there are missing values, we drop them
df.dropna(inplace=True)
# Dropping the missing values
print("Check for missing values after dropping data")
print(df.isnull().sum())
```

```
Check for missing values after dropping data
CustomerID           0
Age                  0
Gender               0
Tenure               0
Usage Frequency      0
Support Calls        0
Payment Delay        0
Subscription Type    0
Contract Length      0
Total Spend          0
Last Interaction     0
Churn                0
dtype: int64
```

Step 3: Perform z-score normalization of the values in the attribute "Last Interaction". Show the mean and variance of the normalized values

Z-score normalization is a scaling method that transforms the data into a distribution with a mean of 0 and a standard deviation of 1.

In [15]:
```
# Get the mean value of the column
mean = train_data["Last Interaction"].mean()
print("Mean of values before normalisation: ", mean)

# Get the standard deviation of the column
std_dev = train_data["Last Interaction"].std()
print("Standard deviation of values: ", std_dev)

print()

# Perform z-score normalization on "Last Interaction", and add it to a new column
train_data["Last Interaction - Normalized "] = (
    train_data["Last Interaction"] - mean / std_dev)

# Get the mean and variance of the normalized data:
norm_mean = train_data["Last Interaction - Normalized "].mean()
norm_var = train_data["Last Interaction - Normalized "].var()

# print the stuff
print("Mean of normalized values: ", norm_mean)
print("Variance of normalized values: ", norm_var)
```

```
-7.831066837512298e-17
0.9999999999999998
```

Step 4: Create five bins for the attribute "Total Spend" such that the bins contain (approximately) equivalent numbers of records.

We can use the qcut function from pandas which creates bins of equal size based on the quantiles of the data.

In [16]:
```
# Create bins
df['Total Spend Bins'] = pd.qcut(df['Total Spend'], q=5)
# Check the bins
print(df['Total Spend Bins'].value_counts())
```

```
Total Spend Bins
(99.999, 378.0]    101271
(719.0, 859.0]     101223
(578.0, 719.0]     101069
(378.0, 578.0]     100822
(859.0, 1000.0]    100821
Name: count, dtype: int64
```

Step 5: Apply one-hot-encoding to the attribute "Contract Length".

One-hot encoding is a process of converting categorical data variables so they can be provided to machine learning algorithms to improve predictions. Pandas provides get_dummies function which is used to convert categorical variable into dummy/indicator variables.

In [17]:
```
# One-hot encoding
df = pd.concat(
    [df, pd.get_dummies(df['Contract Length'], prefix='Contract Length')], axis=1)

# Check the new columns
print(df.columns)
```

```
Index(['CustomerID', 'Age', 'Gender', 'Tenure', 'Usage Frequency',
       'Support Calls', 'Payment Delay', 'Subscription Type',
       'Contract Length', 'Total Spend', 'Last Interaction', 'Churn',
       'Last Interaction Normalized', 'Total Spend Bins',
       'Contract Length_Annual', 'Contract Length_Monthly',
       'Contract Length_Quarterly'],
      dtype='object')
```

Step 6: Define at least one new attribute based on existing attribute, and explain your reason behind your definition.

In [21]:
```
# convert Contract Length to numeric (float) type
df['Contract Length'] = pd.to_numeric(df['Contract Length'], errors='coerce')
#5. Apply one-hot-encoding to the attribute "Contract Length".
df['Spend per Month'] = df['Total Spend'] / df['Contract Length']
#Display first 5 rows of the data frame after the transformation.
df.head()
```

Out[21]:

	CustomerID	Age	Gender	Tenure	Usage Frequency	Support Calls	Payment Delay	Subscription Type	Contract Length	Total Spend	Last Interaction	Churn	Last Interaction Normalized	Total Spend Bins	Contract Length_Annual	Contract Length_Monthly	Contract Length_Quarterly	Spen pe Mont
0	2.0	30.0	Female	39.0	14.0	5.0	18.0	Standard	NaN	932.0	17.0	1.0	0.277572	(859.0, 1000.0]	True	False	False	Na
1	3.0	65.0	Female	49.0	1.0	10.0	8.0	Basic	NaN	557.0	6.0	1.0	-1.000267	(378.0, 578.0]	False	True	False	Na
2	4.0	55.0	Female	14.0	4.0	6.0	18.0	Basic	NaN	185.0	3.0	1.0	-1.348768	(99.999, 378.0]	False	False	True	Na
3	5.0	58.0	Male	38.0	21.0	7.0	7.0	Standard	NaN	396.0	29.0	1.0	1.671578	(378.0, 578.0]	False	True	False	Na
4	6.0	23.0	Male	32.0	20.0	5.0	8.0	Basic	NaN	617.0	20.0	1.0	0.626073	(578.0, 719.0]	False	True	False	Na