# Your Python code here  
x = 10  
y = 20  
result = x + y  
print(result)

# import all necessary libraries  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
import missingno as msno  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.metrics import accuracy\_score  
from sklearn.preprocessing import LabelEncoder  
  
# Read the Excel file  
df = pd.read\_csv('C:\Users\KOFI ADUKPO\Downloads\Streamline\_Delivery\_Service.csv')  
  
#check for the shape of data.  
df.shape  
(7043, 17)  
  
# check for column information  
df.info()  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 7043 entries, 0 to 7042  
Data columns (total 17 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 customerID 7043 non-null object   
 1 gender 7043 non-null object   
 2 senior 7043 non-null int64   
 3 family\_plan 7043 non-null object   
 4 Dependents 7043 non-null object   
 5 customer\_since 7043 non-null int64   
 6 contact\_preference 7043 non-null object   
 7 fraud\_protection 7043 non-null object   
 8 order\_backup 7043 non-null object   
 9 tech\_support 7043 non-null object   
 10 streaming\_movies 7043 non-null object   
 11 contract\_type 7043 non-null object   
 12 paperless 7043 non-null object   
 13 payment\_type 7043 non-null object   
 14 monthly\_bill 7043 non-null float64  
 15 total\_billed 7043 non-null object   
 16 complaint 7043 non-null object   
dtypes: float64(1), int64(2), object(14)  
memory usage: 935.5+ KB  
  
# Obtaining unique value counts for each column  
for column in df.columns:  
 counts = df[column].value\_counts()  
 print(f"Value counts for {column}:")  
 print(counts)  
 print()  
Value counts for customerID:  
0002-ORFBO 1  
6616-AALSR 1  
6625-UTXEW 1  
6625-IUTTT 1  
6625-FLENO 1  
 ..  
3352-RICWQ 1  
3352-ALMCK 1  
3351-NQLDI 1  
3351-NGXYI 1  
9995-HOTOH 1  
Name: customerID, Length: 7043, dtype: int64  
  
Value counts for gender:  
Male 3555  
Female 3488  
Name: gender, dtype: int64  
  
Value counts for senior:  
0 5901  
1 1142  
Name: senior, dtype: int64  
  
Value counts for family\_plan:  
No 3641  
Yes 3402  
Name: family\_plan, dtype: int64  
  
Value counts for Dependents:  
No 4933  
Yes 2110  
Name: Dependents, dtype: int64  
  
Value counts for customer\_since:  
1 613  
72 362  
2 238  
3 200  
4 176  
 ...   
28 57  
39 56  
44 51  
36 50  
0 11  
Name: customer\_since, Length: 73, dtype: int64  
  
Value counts for contact\_preference:  
Mobile 3096  
Phone 2421  
Email 1526  
Name: contact\_preference, dtype: int64  
  
Value counts for fraud\_protection:  
No 5024  
Yes 2019  
Name: fraud\_protection, dtype: int64  
  
Value counts for order\_backup:  
No 4614  
Yes 2429  
Name: order\_backup, dtype: int64  
  
Value counts for tech\_support:  
No 4999  
Yes 2044  
Name: tech\_support, dtype: int64  
  
Value counts for streaming\_movies:  
No 4311  
Yes 2732  
Name: streaming\_movies, dtype: int64  
  
Value counts for contract\_type:  
Month-to-month 3875  
Two year 1695  
One year 1473  
Name: contract\_type, dtype: int64  
  
Value counts for paperless:  
Yes 4171  
No 2872  
Name: paperless, dtype: int64  
  
Value counts for payment\_type:  
Credit card (automatic) 5499  
Bank transfer (automatic) 1544  
Name: payment\_type, dtype: int64  
  
Value counts for monthly\_bill:  
20.05 61  
19.85 45  
19.95 44  
19.90 44  
20.00 43  
 ..  
116.55 1  
106.85 1  
68.55 1  
108.80 1  
67.85 1  
Name: monthly\_bill, Length: 1585, dtype: int64  
  
Value counts for total\_billed:  
 11  
20.2 11  
19.75 9  
19.9 8  
20.05 8  
 ..  
2387.75 1  
6302.8 1  
2058.5 1  
829.55 1  
3707.6 1  
Name: total\_billed, Length: 6531, dtype: int64  
  
Value counts for complaint:  
No 5174  
Yes 1869  
Name: complaint, dtype: int64  
  
# checking for missing values  
df.isnull().sum()  
customerID 0  
gender 0  
senior 0  
family\_plan 0  
Dependents 0  
customer\_since 0  
contact\_preference 0  
fraud\_protection 0  
order\_backup 0  
tech\_support 0  
streaming\_movies 0  
contract\_type 0  
paperless 0  
payment\_type 0  
monthly\_bill 0  
total\_billed 0  
complaint 0  
dtype: int64  
  
msno.bar(df)  
plt.title('Missing Data Percentage')  
plt.show()  
  
# checking for duplicate  
df.duplicated()  
0 False  
1 False  
2 False  
3 False  
4 False  
 ...   
7038 False  
7039 False  
7040 False  
7041 False  
7042 False  
Length: 7043, dtype: bool  
  
# Create a box plot for monthly bill  
plt.boxplot(df['monthly\_bill'])  
plt.title('Box Plot of monthly\_bill')  
plt.ylabel('Values')  
plt.show()  
  
# Create a box plot for customer\_since  
plt.boxplot(df['customer\_since'])  
plt.title('Box Plot of customer\_since')  
plt.ylabel('Values')  
plt.show()  
  
  
# Plot the class distribution  
class\_counts.plot(kind='bar')  
plt.xlabel('Gender')  
plt.ylabel('Count')  
plt.title('Class Distribution')  
plt.show()  
  
# dropping rows with empty string cells in 'total\_billed'  
df = df[df['total\_billed'] != ' ']  
  
bins = [0, 24, 48, 72]  
labels = ['long', 'longer', 'longest']  
# keep copy of original df  
df\_1 = df.copy()  
  
# Group integer columns into categories  
df['customer\_since'] = pd.cut(df['customer\_since'], bins=bins, labels=labels)  
  
bins = [18, 52, 86, 120]  
labels = ['high', 'higher', 'highest']  
df['monthly\_bill'] = pd.cut(df['monthly\_bill'], bins=bins, labels=labels)  
  
bins = [18, 2907, 5796, 8685]  
labels = ['high', 'higher', 'highest']  
df['total\_billed'] = pd.cut(df['total\_billed'], bins=bins, labels=labels)  
  
# convert column type to object  
df['senior'] = df['senior'].astype(object)  
  
# Extract the feature columns (X) and target column (y)  
X = df.drop(columns=['complaint','customerID'])  
y = df['complaint']  
  
# Split the dataset into training and testing sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Print the shapes of the training and testing sets  
print("X\_train shape:", X\_train.shape)  
print("X\_test shape:", X\_test.shape)  
print("y\_train shape:", y\_train.shape)  
print("y\_test shape:", y\_test.shape)  
X\_train shape: (5625, 15)  
X\_test shape: (1407, 15)  
y\_train shape: (5625,)  
y\_test shape: (1407,)  
  
k = 5 # Number of neighbors to consider  
knn = KNeighborsClassifier(n\_neighbors=k)  
  
# train madel  
knn.fit(X\_train, y\_train)  
  
# make predictions  
y\_pred = knn.predict(X\_test)  
  
# evaluate model  
accuracy = accuracy\_score(y\_test, y\_pred)  
print("Accuracy:", accuracy)  
Accuracy: 0.7583511016346838