Data Preprocessing

```
Data preprocessing is a crucial step in any machine learning project. It involves loading, exploring,
handling missing data, dealing with outliers, encoding categorical variables, and splitting the data into
training and testing sets
```

```
Import python Libraries
```

In [1]:

```python
import numpy as np
import pandas as pd
```

```
Load the Dataset:
```

In [8]:

```python
df = pd.read_excel(r"C:\Users\ELCOT\Downloads\customer_churn_large_dataset.xlsx")
```

In [10]:

```python
df.head()
```

Out[10]:

	CustomerID	Name	Age	Gender	Location	Subscription_Length_Months	Monthly_Bill	Total_Usage_GB	Chu
0	1	Customer_1	63	Male	Los Angeles	17	73.36	236	
1	2	Customer_2	62	Female	New York	1	48.76	172	
2	3	Customer_3	24	Female	Los Angeles	5	85.47	460	
3	4	Customer_4	36	Female	Miami	3	97.94	297	
4	5	Customer_5	46	Female	Miami	19	58.14	266	

In [11]:

```python
df.shape
```

Out[11]:

```
(100000, 9)
```

```
Initial Data Exploration:
```

In [12]:

```
df.describe()
```

Out[12]:

	CustomerID	Age	Subscription_Length_Months	Monthly_Bill	Total_Usage_GB	Churn
count	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000
mean	50000.500000	44.027020	12.490100	65.053197	274.393650	0.497790
std	28867.657797	15.280283	6.926461	20.230696	130.463063	0.499998
min	1.000000	18.000000	1.000000	30.000000	50.000000	0.000000
25%	25000.750000	31.000000	6.000000	47.540000	161.000000	0.000000
50%	50000.500000	44.000000	12.000000	65.010000	274.000000	0.000000
75%	75000.250000	57.000000	19.000000	82.640000	387.000000	1.000000
max	100000.000000	70.000000	24.000000	100.000000	500.000000	1.000000

In [13]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   CustomerID                  100000 non-null  int64  
 1   Name                        100000 non-null  object 
 2   Age                         100000 non-null  int64  
 3   Gender                      100000 non-null  object 
 4   Location                    100000 non-null  object 
 5   Subscription_Length_Months  100000 non-null  int64  
 6   Monthly_Bill                100000 non-null  float64
 7   Total_Usage_GB              100000 non-null  int64  
 8   Churn                       100000 non-null  int64  
dtypes: float64(1), int64(5), object(3)
memory usage: 6.9+ MB
```

In [14]:

```
df.isnull().sum()
```

Out[14]:

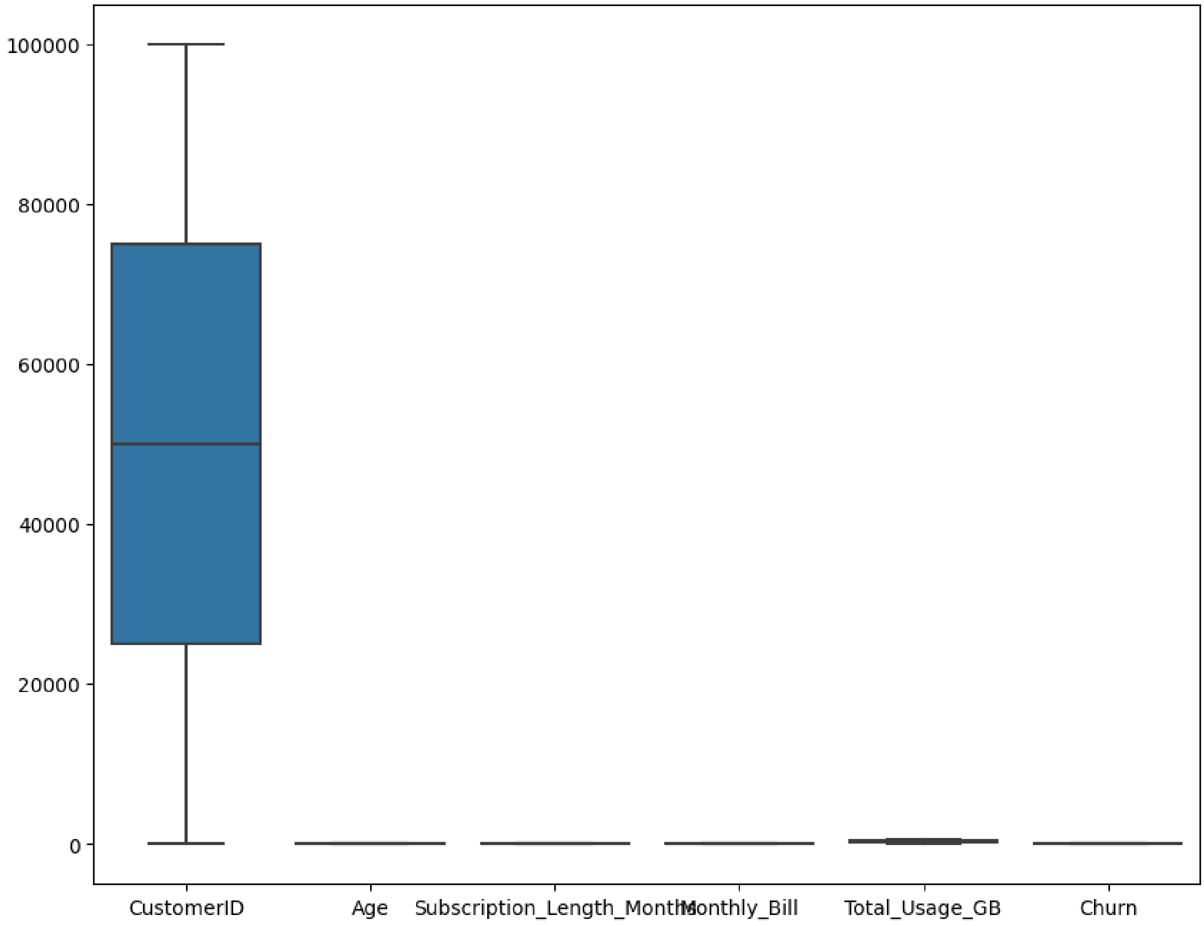
```
CustomerID                    0
Name                          0
Age                           0
Gender                        0
Location                      0
Subscription_Length_Months    0
Monthly_Bill                  0
Total_Usage_GB                0
Churn                         0
dtype: int64
```

```
Handle Missing Data: There is no null values, missing values in df
```

```
Handle Outliers:
```

In [36]:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10,8))
sns.boxplot(df)
plt.show()
```




```
Encode Categorical Variables:
```

In [43]:

```
from sklearn.preprocessing import LabelEncoder
```

In [45]:

```
label_encoder = LabelEncoder()

# Apply label encoding to each categorical column
for col in ['Name', 'Location', 'Gender']:
    df[col] = label_encoder.fit_transform(df[col])
```

```
label encoding may not suitable for nominal variables. so, choose one-hot encoding
```

In [46]:

```
df.dtypes
```

Out[46]:

```
CustomerID                    int64
Name                          int64
Age                           int64
Gender                        int64
Location                      int64
Subscription_Length_Months    int64
Monthly_Bill                float64
Total_Usage_GB                int64
Churn                         int64
dtype: object
```

In [47]:

```
df.head()
```

Out[47]:

	CustomerID	Name	Age	Gender	Location	Subscription_Length_Months	Monthly_Bill	Total_Usage_GB	Churn
0	1	0	63	1	2	17	73.36	236	0
1	2	11112	62	0	4	1	48.76	172	0
2	3	22223	24	0	2	5	85.47	460	0
3	4	33334	36	0	3	3	97.94	297	1
4	5	44445	46	0	3	19	58.14	266	0

```
1-->MALE
0 --> Female
```

```
Split Data into Training and Testing Sets
```

In [50]:

```
from sklearn.model_selection import train_test_split
```

In [52]:

```
X = df.drop('Churn', axis=1)
y = df['Churn']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [53]:

```
X_train
```

Out[53]:

	CustomerID	Name	Age	Gender	Location	Subscription_Length_Months	Monthly_Bill	Total_Usage_GB
75220	75221	72471	54	0	4	5	84.50	205
48955	48956	43286	28	1	4	24	82.06	239
44966	44967	38854	57	1	0	12	52.29	62
13568	13569	3968	19	1	1	19	32.57	173
92727	92728	91922	56	0	3	8	33.52	314
...	...	...	...	...	...	...	...	...
6265	6266	58513	35	1	3	21	67.33	235
54886	54887	49876	56	1	0	13	85.40	347
76820	76821	74248	69	1	1	2	76.24	321
860	861	84557	55	1	0	12	89.19	315
15795	15796	6442	26	0	2	17	70.41	335

80000 rows × 8 columns

In [54]:

```
X_test
```

Out[54]:

	CustomerID	Name	Age	Gender	Location	Subscription_Length_Months	Monthly_Bill	Total_Usage_GB
75721	75722	73027	48	0	1	11	88.48	492
80184	80185	77986	49	1	4	13	40.61	423
19864	19865	10963	31	0	2	5	33.01	276
76699	76700	74114	53	1	4	4	94.66	339
92991	92992	92215	23	0	2	24	82.21	304
...	...	...	...	...	...	...	...	...
32595	32596	25109	38	1	3	20	79.70	118
29313	29314	21463	53	1	2	12	96.75	363
37862	37863	30961	68	1	3	13	39.33	137
53421	53422	48250	34	1	3	13	95.14	498
42410	42411	36016	68	1	0	11	30.91	348

20000 rows × 8 columns

In [55]:

```
y_train
```

Out[55]:

```
75220    1
48955    1
44966    1
13568    1
92727    1
        ..
6265     0
54886    0
76820    1
860      1
15795    0
Name: Churn, Length: 80000, dtype: int64
```

In [56]:

```
y_test
```

Out[56]:

```
75721    0
80184    0
19864    0
76699    1
92991    0
        ..
32595    0
29313    1
37862    1
53421    0
42410    1
Name: Churn, Length: 20000, dtype: int64
```

Feature Engineering

In [57]:

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)
```

In [58]:

```
X_train_poly
```

Out[58]:

```
array([[1.0000000e+00, 7.5221000e+04, 7.2471000e+04, ..., 7.1402500e+03,
        1.7322500e+04, 4.2025000e+04],
       [1.0000000e+00, 4.8956000e+04, 4.3286000e+04, ..., 6.7338436e+03,
        1.9612340e+04, 5.7121000e+04],
       [1.0000000e+00, 4.4967000e+04, 3.8854000e+04, ..., 2.7342441e+03,
        3.2419800e+03, 3.8440000e+03],
       ...,
       [1.0000000e+00, 7.6821000e+04, 7.4248000e+04, ..., 5.8125376e+03,
        2.4473040e+04, 1.0304100e+05],
       [1.0000000e+00, 8.6100000e+02, 8.4557000e+04, ..., 7.9548561e+03,
        2.8094850e+04, 9.9225000e+04],
       [1.0000000e+00, 1.5796000e+04, 6.4420000e+03, ..., 4.9575681e+03,
        2.3587350e+04, 1.1222500e+05]])
```

In [59]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Model Building

In [*]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusi
```

```
Decision Tree
```

In [*]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [*]:

```
# decison tree
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
y_pred = dtree.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

Support Vector Machine

In [*]:

```
from sklearn import svm
```

In [*]:

```
#decison tree
svm = svm.SVC()
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

In [*]:

```
# Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
```

In [*]:

```
# Make predictions on the test data
y_pred = rf_classifier.predict(X_test)
```

In [*]:

```
from sklearn.model_selection import GridSearchCV
param_grid = {'n_estimators': [100, 200, 300], 'max_depth': [10, 20, 30]}
grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5)
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_
```

In [*]:

```
from sklearn.model_selection import cross_val_score
model = RandomForestClassifier(n_estimators=100, max_depth=20, random_state=42)
scores = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy')
average_accuracy = scores.mean()
```

In [*]:

```
average_accuracy
```

In [ ]:

```

```