In [2]:
```
import pandas as pd
```

Data Preprocessing

In [4]:
```
data = pd.read_excel("customer_churn_large_dataset.xlsx", sheet_name ='Sheet1')
data.head()
```

Out[4]:

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

In [5]:
```
data.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
```

Feature Engineering

In [6]:
```
data.describe()
```

Out[6]:

	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

Note : Since data desription tells that their is no outlier so there is also no need to do feature engineering as data in on same scale.

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
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
```

In [9]:
```
data["Churn"].value_counts()
```

Out[9]:
```
0    50221
1    49779
Name: Churn, dtype: int64
```

Data Viz

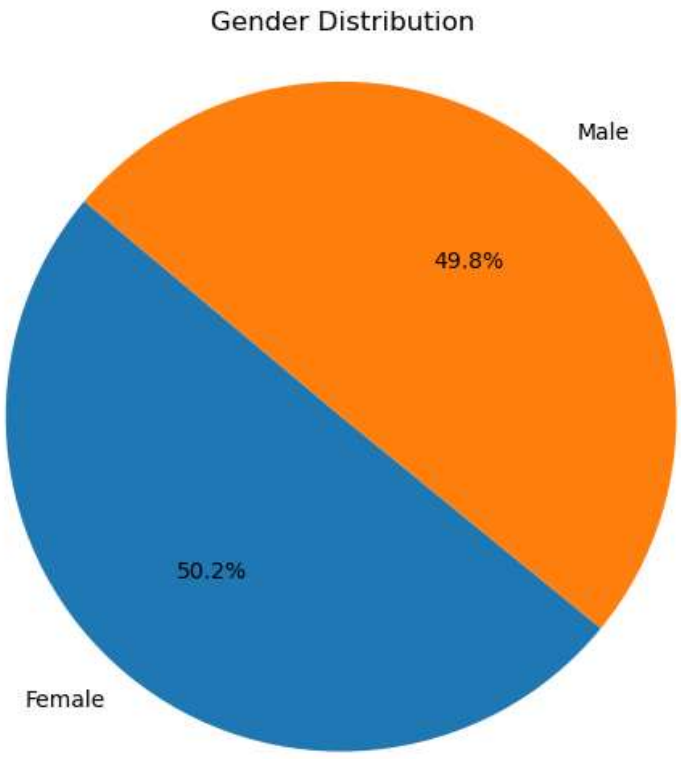
In [11]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [13]:
```
# Count the occurrences of each gender
gender_counts = data['Gender'].value_counts()

# Create a pie chart
plt.figure(figsize=(6, 6))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=140)
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

# Add a title
plt.title('Gender Distribution')

# Display the plot
plt.show()
```



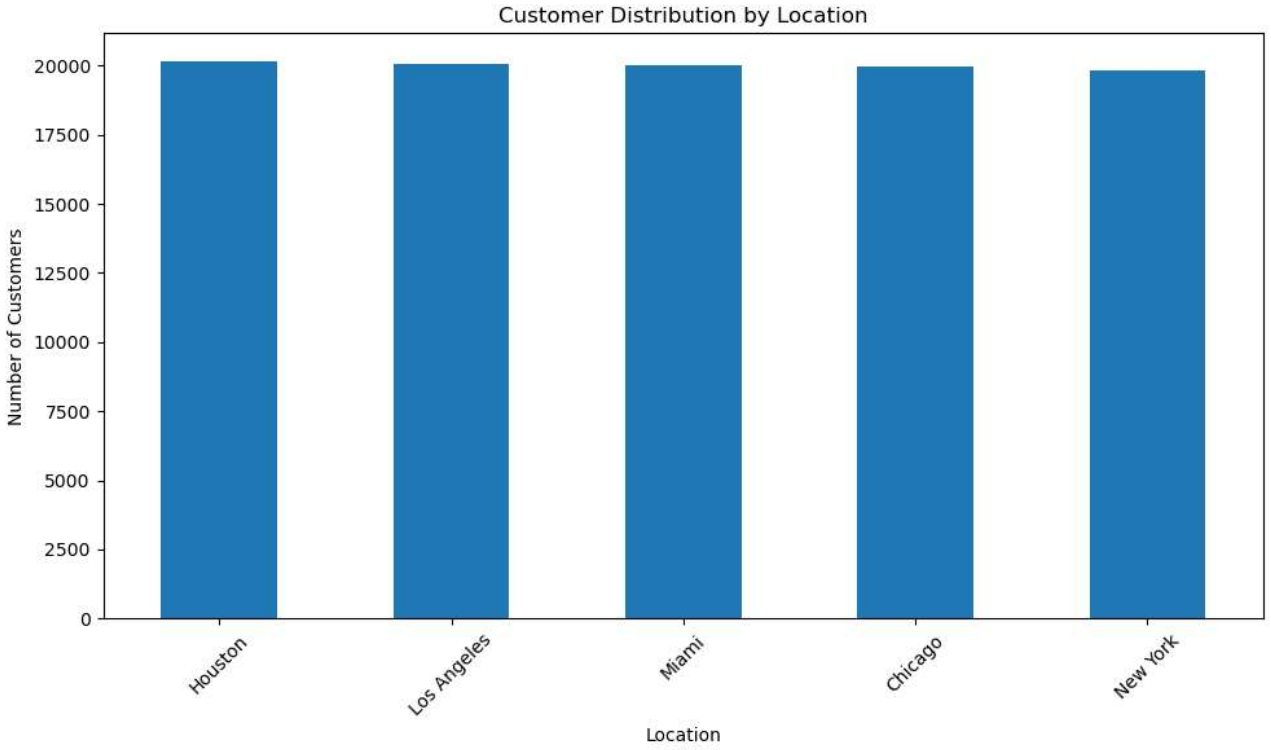
Note : The pie chart shows that our data has approximately similar range of Genders.

In [14]:
```
# Count the occurrences of each location
location_counts = data['Location'].value_counts()

# Create a bar chart
plt.figure(figsize=(10, 6))
location_counts.plot(kind='bar')
plt.xlabel('Location')
plt.ylabel('Number of Customers')
plt.title('Customer Distribution by Location')

# Rotate x-axis labels for better visibility if needed
plt.xticks(rotation=45)

# Display the plot
plt.tight_layout()
plt.show()
```



Note : Location wise also similar range of people had our subscription.

In [15]:
```
data.columns
```

Out[15]:
```
Index(['CustomerID', 'Name', 'Age', 'Gender', 'Location',
       'Subscription_Length_Months', 'Monthly_Bill', 'Total_Usage_GB',
       'Churn'],
      dtype='object')
```

In [16]:
```
x = data[['Age', 'Subscription_Length_Months', 'Monthly_Bill', 'Total_Usage_GB']]

y = data[["Churn"]]
```

In [17]:
```
# To have a glance at correlation between x table features

x.corr()
```

Out[17]:

	Age	Subscription_Length_Months	Monthly_Bill	Total_Usage_GB
Age	1.000000	0.003382	0.001110	0.001927
Subscription_Length_Months	0.003382	1.000000	-0.005294	-0.002203
Monthly_Bill	0.001110	-0.005294	1.000000	0.003187
Total_Usage_GB	0.001927	-0.002203	0.003187	1.000000

Model Building

In [18]:
```
from sklearn.model_selection import train_test_split
```

In [19]:
```
# Split the data into train and test

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2, stratify = y, random_state = 2
```

In [20]:
```
print(x.shape, x_train.shape, x_test.shape)
```

```
(100000, 4) (80000, 4) (20000, 4)
```

In [21]:
```
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

In [22]:
```
# Model Training

gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
svc = SVC(kernel = 'sigmoid', gamma = 1.0)
knc = KNeighborsClassifier()
dtc = DecisionTreeClassifier(max_depth = 5)
lrc = LogisticRegression(solver = 'liblinear', penalty = 'l1')
rfc = RandomForestClassifier(n_estimators = 50, random_state = 2)
abc = AdaBoostClassifier(n_estimators = 50, random_state = 2)
bc = BaggingClassifier(n_estimators = 50, random_state = 2)
etc = ExtraTreesClassifier(n_estimators = 50, random_state = 2)
gbdt = GradientBoostingClassifier(n_estimators = 50, random_state = 2)
xgb = XGBClassifier(n_estimators = 50, random_state = 2)
```

In [23]:
```
clfs = {
    'Support Vector Classification' : svc,
    'KNeighbors Classifier' : knc,
    'Gaussian Naive Bayes' : gnb,
    'Multinomial Naive Bayes': mnb,
    'Bernoulli Naive Bayes' : bnb,
    'Decision Tree Classifier': dtc,
    'Logistic Regression': lrc,
    'Random Forest Classifier': rfc,
    'AdaBoost Classifier': abc,
    'Bagging Classifier': bc,
    'Extra Trees Classifier': etc,
    'Gradient Boosting Classifier': gbdt,
    'XGB Classifier': xgb
}
```

In [24]:
```
def train_classifier(clf, x_train, y_train, x_test, y_test):
    clf.fit(x_train, y_train)
    y_pred = clf.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred)

    return accuracy
```

In [25]:
```
%%time

train_classifier(svc, x_train, y_train, x_test, y_test)
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)

Wall time: 9min 20s
```

Out[25]: 0.5022

In [26]:
```
%%time

accuracy_scores = []

for name, clf in clfs.items():

    accuracy = train_classifier(clf, x_train, y_train, x_test, y_test)

    print("For ", name)
    print("Accuracy - ", accuracy)

    accuracy_scores.append(accuracy)
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)

For  Support Vector Classification
Accuracy -  0.5022

C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:198: DataConversionWar
ning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_sam
ples,), for example using ravel().
  return self._fit(X, y)
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Un
like other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically pres
erves the axis it acts along. In SciPy 1.11.0, this behavior will change: the default value of `keepdim
s` will become False, the `axis` over which the statistic is taken will be eliminated, and the value No
ne will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)

For  KNeighbors Classifier
Accuracy -  0.5007
For  Gaussian Naive Bayes
Accuracy -  0.5002
For  Multinomial Naive Bayes
Accuracy -  0.5013
For  Bernoulli Naive Bayes
Accuracy -  0.5022
For  Decision Tree Classifier
Accuracy -  0.5041

C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)

For  Logistic Regression
Accuracy -  0.50045

C:\Users\lenovo\AppData\Local\Temp\ipykernel_5760\2342298805.py:2: DataConversionWarning: A column-vect
or y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example
using ravel().
  clf.fit(x_train, y_train)

For  Random Forest Classifier
Accuracy -  0.4989
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)

For  AdaBoost Classifier
Accuracy -  0.5092

C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\ensemble\_bagging.py:719: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)

For  Bagging Classifier
Accuracy -  0.50035

C:\Users\lenovo\AppData\Local\Temp\ipykernel_5760\2342298805.py:2: DataConversionWarning: A column-vect
or y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example
using ravel().
  clf.fit(x_train, y_train)

For  Extra Trees Classifier
Accuracy -  0.4995

C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\ensemble\_gb.py:494: DataConversionWarning: A colum
n-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for
example using ravel().
  y = column_or_1d(y, warn=True)

For  Gradient Boosting Classifier
Accuracy -  0.503
For  XGB Classifier
Accuracy -  0.4998
Wall time: 9min 47s
```

In [27]:
```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores}).sort_values('Accuracy
performance_df
```

Out[27]:

	Algorithm	Accuracy
8	AdaBoost Classifier	0.50920
5	Decision Tree Classifier	0.50410
11	Gradient Boosting Classifier	0.50300
0	Support Vector Classification	0.50220
4	Bernoulli Naive Bayes	0.50220
3	Multinomial Naive Bayes	0.50130
1	KNeighbors Classifier	0.50070
6	Logistic Regression	0.50045
9	Bagging Classifier	0.50035
2	Gaussian Naive Bayes	0.50020
12	XGB Classifier	0.49980
10	Extra Trees Classifier	0.49950
7	Random Forest Classifier	0.49890

Model Optimization

In [28]:
```
# Using AdaBoostClassifier since it has max accuracy

model = AdaBoostClassifier(n_estimators = 100, random_state = 2)
model.fit(x_train, y_train)
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[28]:
```
AdaBoostClassifier(n_estimators=100, random_state=2)
```

Model Deployment

In [30]:
```
import numpy as np
```

In [31]:
```
print("Churn Score Prediction : ")
a = float(input('Age: '))
b = float(input('Subscription_Length_Months: '))
c = float(input('Monthly_Bill'))
d = float(input('Total_Usage_GB: '))

features = np.array([[a, b, c, d]])
print("Predicted Churn Score = ", model.predict(features))
```

```
Churn Score Prediction :
Age: 67
Subscription_Length_Months: 470
Monthly_Bill578
Total_Usage_GB: 234
Predicted Churn Score =  [0]

C:\Users\lenovo\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid fea
ture names, but AdaBoostClassifier was fitted with feature names
  warnings.warn(
```

In [ ]:

In [ ]:

In [ ]: