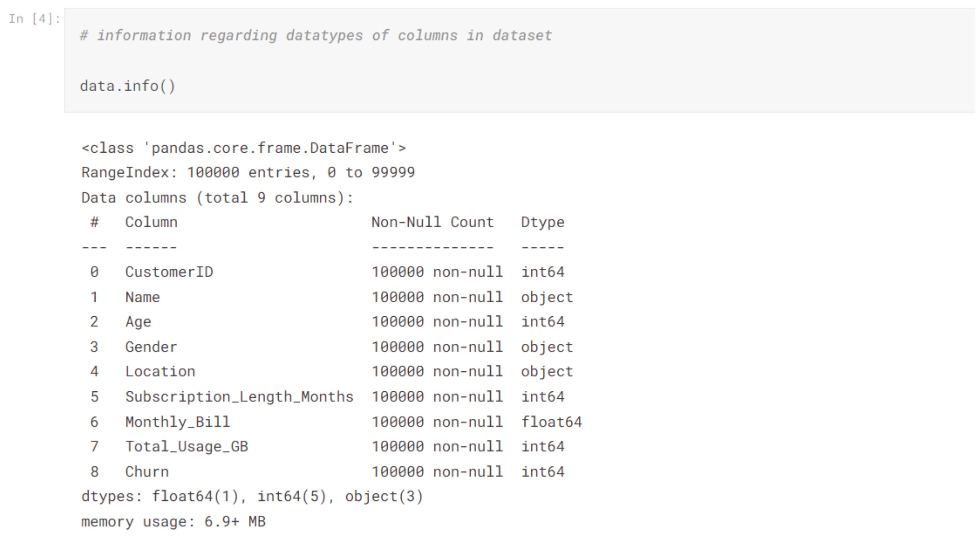
Customer Churn Predictor App

- **Datatypes of the columns:**



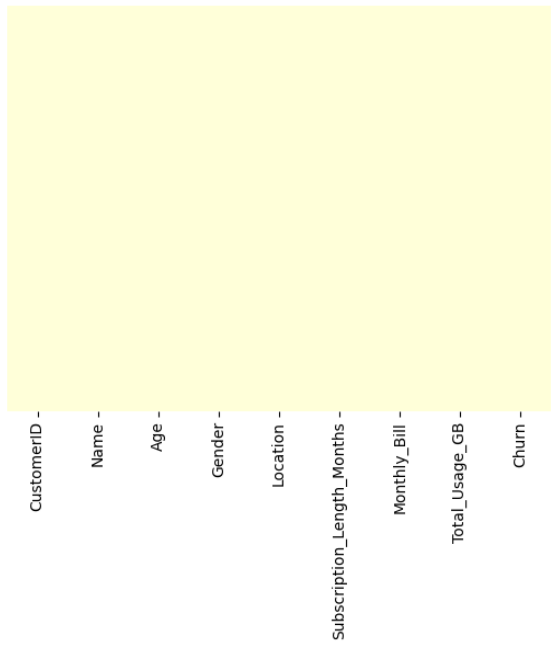
```
In [4]:
# information regarding datatypes of columns in dataset

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
```

Gender, Location, and Name are object types, the rest are all numeric types (int or float)

- **CHECKING FOR NULL VALUES**

Heatmap:

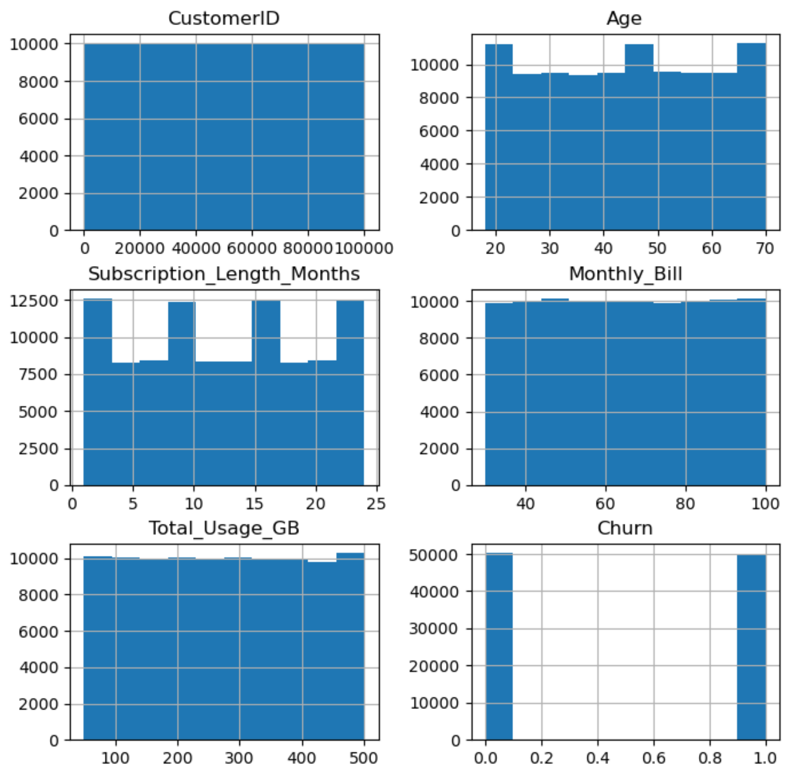


Rechecking….

```
CustomerID                    0
Name                          0
Age                           0
Gender                        0
Location                      0
Subscription_Length_Months    0
Monthly_Bill                  0
Total_Usage_GB                0
Churn                         0
```

There are no missing values in our dataset, so we do not require any imputations using mean, mode etc.

- **Dataset HISTOGRAM PLOTS**



Seeing the histogram plots, it seems that there aren't any outliers. For the "Churn" column, it appears that both classes have an equal distribution of 50k each. So, it is a balanced dataset.

- **No. of unique values in each column**

```
In [8]: print(len(pd.unique(data['Location'])))
```

```
5
```

```
In [9]: locations = []
        for loc in data['Location']:
            if loc not in locations:
                locations.append(loc)
        print(locations)
```

```
['Los Angeles', 'New York', 'Miami', 'Chicago', 'Houston']
```

- **Percentage of people churned by location**

```
Percent of People Who Churned from Los Angeles ---> 49.2989371787835 %
Percent of People Who Churned from New York ---> 50.36592136476051 %
Percent of People Who Churned from Miami ---> 50.30203185063152 %
Percent of People Who Churned from Chicago ---> 49.829642248722315 %
Percent of People Who Churned from Houston ---> 49.10949049957831 %
```

Almost 50% of all people in each location churned and rest 50% retained, so location possibly has little influence on churn.

- **Percentage of people churned by gender**

```
Percent of Males Who Churned ---> 49.88550538325566 %
Percent of Females Who Churned ---> 49.67341086506293 %
```

Almost 50% from each gender churned and rest 50% retained, so gender also possibly has little in -fluence on churn.

- **Data Preparation**

1. CustomerID and Name do not influence customer churn, so they can be dropped from our dataset.

```
data = data.drop(columns=['CustomerID', 'Name'])
```

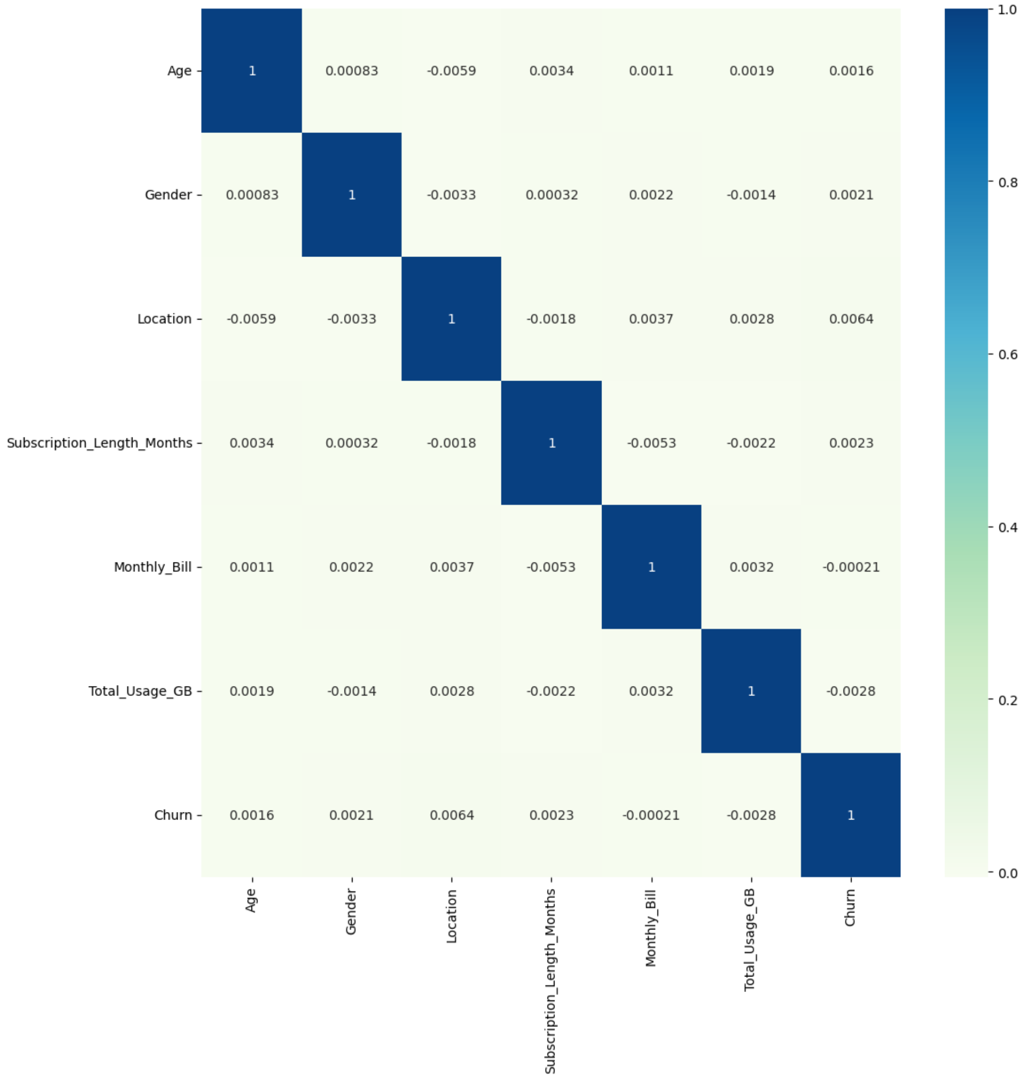
2. Gender is categorical data, so we should go for label encoding.

```
data['Gender'] = LabelEncoder().fit_transform(data['Gender'])
```

3. There are 5 unique locations --- 'Los Angeles', 'New York', 'Miami', 'Chicago', And ‘Houston' ---- so locations can also be label encoded.

```
data['Location'] = LabelEncoder().fit_transform(data['Location'])
```

- **HeatMap Analysis to check for correlation among the columns**



There is very little correlation of churn with the other columns.

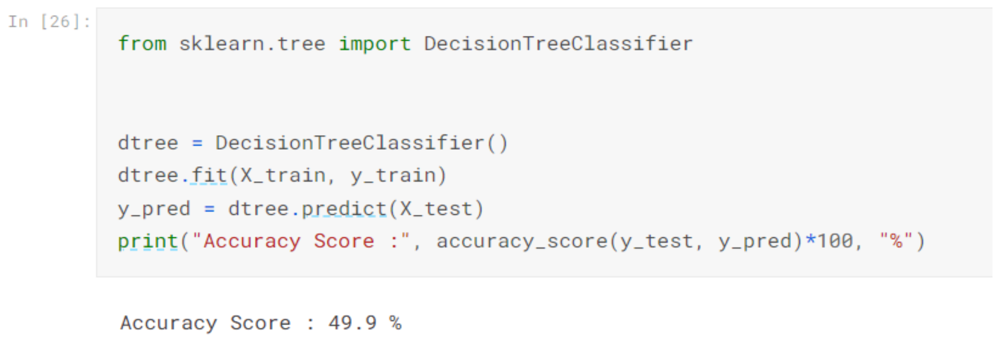
- **Split dataset into train-test**

```
#test size 20% and train size 80%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=7)
```

We use a 80:20 ratio for training data and test data split. We have also extracted our target variable in y and given features in X, so that after split, `X_train, X_test, y_train, y_test` **can be used in our models easily.**

- **Testing different models.....**

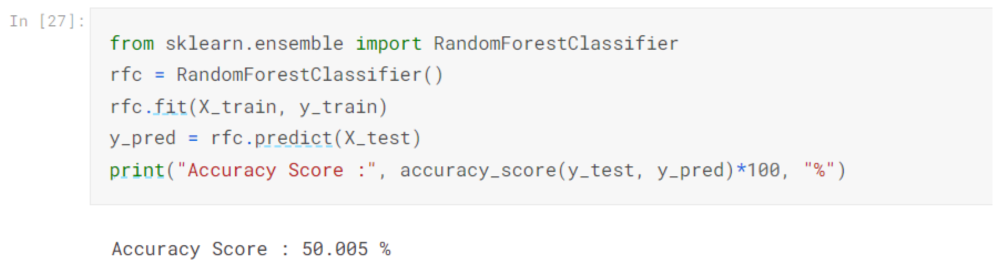
1. Decision Tree



```
In [26]:
from sklearn.tree import DecisionTreeClassifier


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

```
Accuracy Score : 49.9 %
```

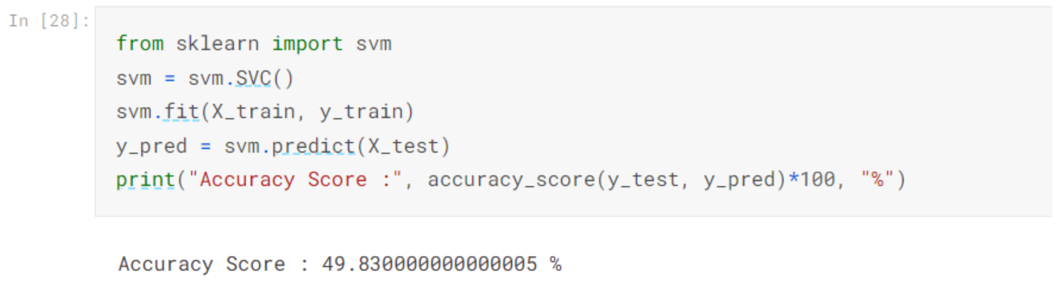
2. Random Forest



```
In [27]:
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

```
Accuracy Score : 50.005 %
```

3. Support Vector Machine



```
In [28]:
from sklearn import svm
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, "%")
```

```
Accuracy Score : 49.830000000000005 %
```

4. Multi-layer Perceptron

```
model = keras.Sequential([
    keras.layers.Input(shape=(X_train.shape[1],)),
    keras.layers.Dense(8, activation='relu'),
    keras.layers.Dense(4, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')      # Output layer with 1 neuron and sigmoid activation
])
```

I designed a simple MLP in Keras with 2 hidden layers and the output layer has 1 neuron with sigmoid activation so that binary classification is possible. Binary Cross Entropy is taken as the loss function because it better identifies where the model makes a probability prediction that deviates from the ground truth substantially and then penalizes the model, thereby leading to more accurate classification.

```
accuracy = accuracy_score(np.asarray(y_test, dtype = int), np.asarray(y_pred, dtype = int))
print("Accuracy Score:", accuracy * 100, "%")
```

```
Accuracy Score: 49.864999999999995 %
```

Still, only about 50% accuracy is obtained in all models.

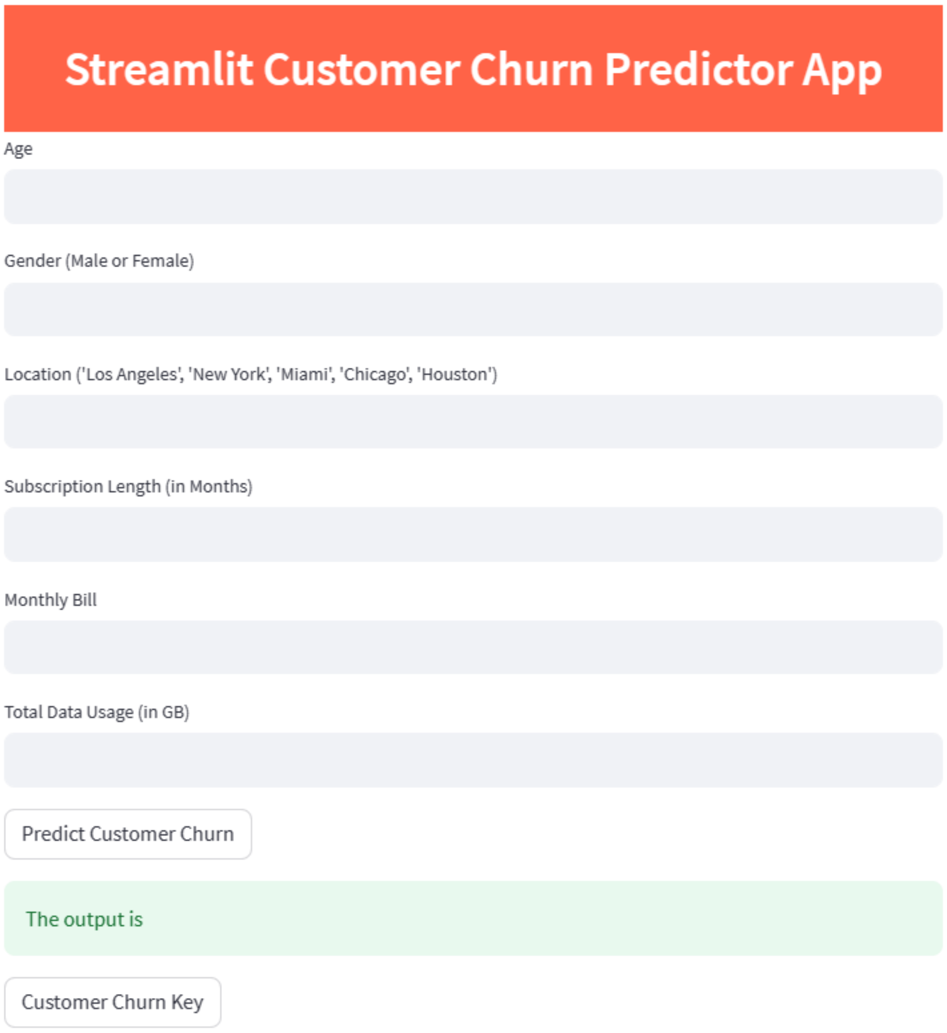
- **Saving the model**

```
In [34]:  model.save("model.h5")
```

The model is saved as .h5 file which can be loaded later to make predictions on unseen data. The web_app.py file shows the deployment of the model.

- **Model Deployment Demonstration**

Home Page


Streamlit Customer Churn Predictor App

Age

Gender (Male or Female)

Location ('Los Angeles', 'New York', 'Miami', 'Chicago', 'Houston')

Subscription Length (in Months)

Monthly Bill

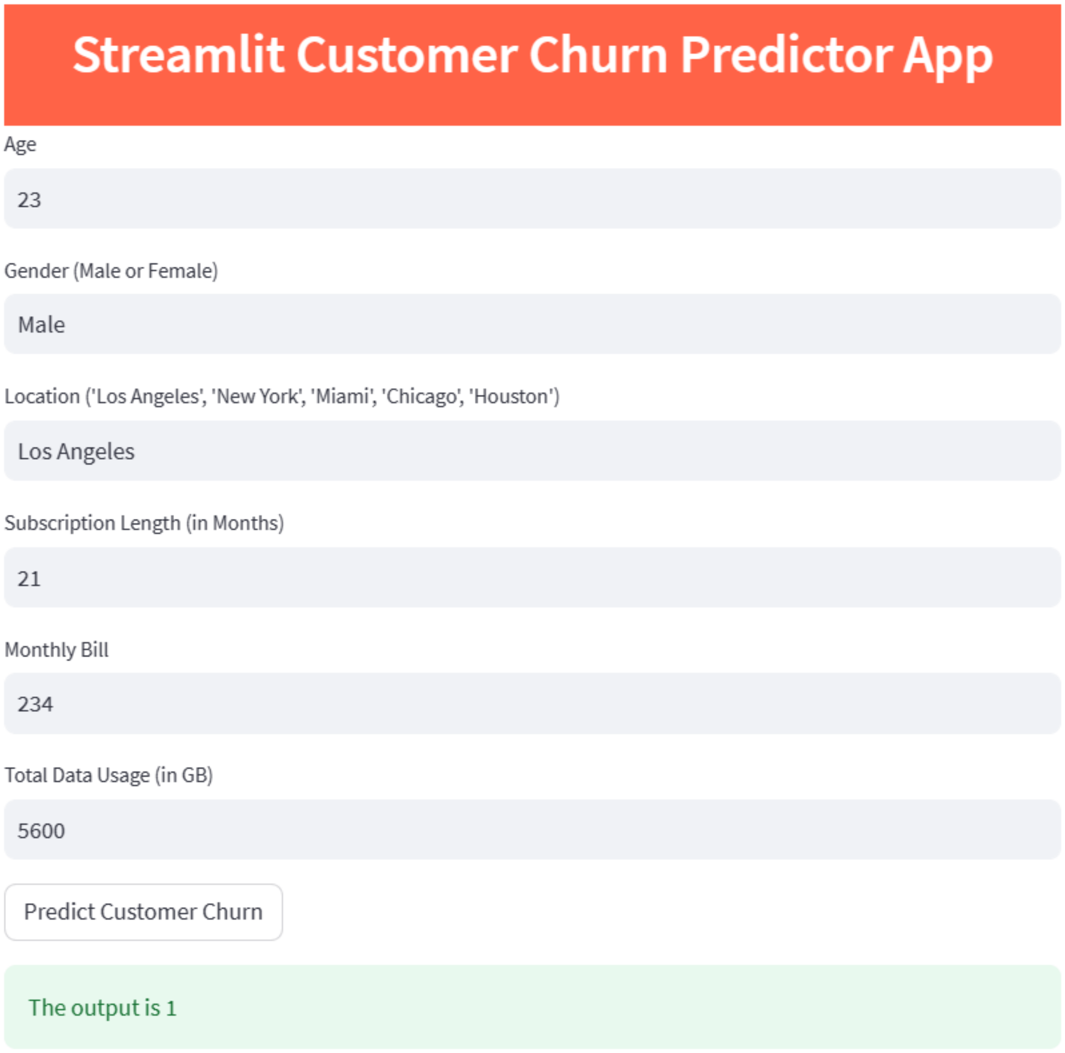
Total Data Usage (in GB)

Predict Customer Churn

The output is

Customer Churn Key

Prediction



Streamlit Customer Churn Predictor App

Age

23

Gender (Male or Female)

Male

Location ('Los Angeles', 'New York', 'Miami', 'Chicago', 'Houston')

Los Angeles

Subscription Length (in Months)

21

Monthly Bill

234

Total Data Usage (in GB)

5600

The output is 1