# churn-predict

July 19, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df=pd.read_excel(r'C:\3 company work for intenship\Sunbase\customer_churn_large_dataset.xlsx')
```

[3]:
```
df.shape
```

[3]:
```
(100000, 9)
```

[4]:
```
df.head()
```

[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 |

[5]:
```
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
```

[6]:
```
df.isnull().sum()
```

[6]:
```
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
```

[7]:
```
df.describe()
```

[7]:

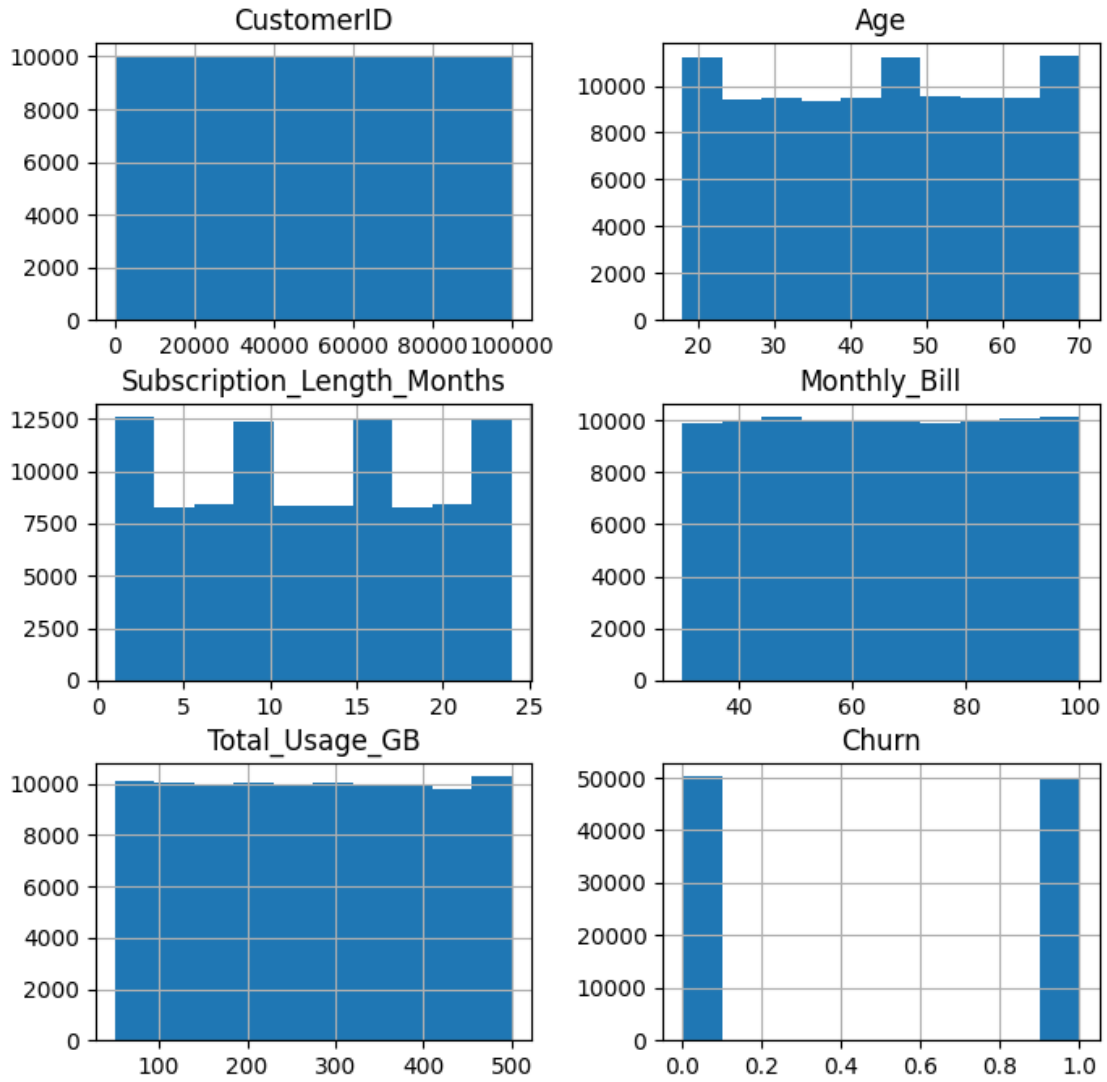
| | CustomerID | Age | Subscription_Length_Months | \ |
|---|---|---|---|---|
| count | 100000.000000 | 100000.000000 | 100000.000000 | |
| mean | 50000.500000 | 44.027020 | 12.490100 | |
| std | 28867.657797 | 15.280283 | 6.926461 | |
| min | 1.000000 | 18.000000 | 1.000000 | |
| 25% | 25000.750000 | 31.000000 | 6.000000 | |
| 50% | 50000.500000 | 44.000000 | 12.000000 | |
| 75% | 75000.250000 | 57.000000 | 19.000000 | |
| max | 100000.000000 | 70.000000 | 24.000000 | |

| | Monthly_Bill | Total_Usage_GB | Churn |
|---|---|---|---|
| count | 100000.000000 | 100000.000000 | 100000.000000 |
| mean | 65.053197 | 274.393650 | 0.497790 |
| std | 20.230696 | 130.463063 | 0.499998 |
| min | 30.000000 | 50.000000 | 0.000000 |
| 25% | 47.540000 | 161.000000 | 0.000000 |
| 50% | 65.010000 | 274.000000 | 0.000000 |
| 75% | 82.640000 | 387.000000 | 1.000000 |
| max | 100.000000 | 500.000000 | 1.000000 |

[8]:
```
df['Location'].value_counts()
```

[8]:
```
Houston          20157
Los Angeles      20041
Miami            20031
Chicago          19958
New York         19813
Name: Location, dtype: int64
```

[9]:
```
df.hist(figsize=(8,8))
plt.show()
```




[10]:
```
print(len(pd.unique(df['Location'])))
```

5

[11]:
```
df.nunique()
```

[11]:
```
CustomerID                    100000
Name                          100000
Age                               53
Gender                             2
Location                           5
Subscription_Length_Months        24
Monthly_Bill                    7001
Total_Usage_GB                   451
Churn                              2
dtype: int64
```

[12]:
```
df = df.drop(columns=['CustomerID', 'Name'])
```

[13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 7 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   Age                          100000 non-null  int64
 1   Gender                       100000 non-null  object
 2   Location                     100000 non-null  object
 3   Subscription_Length_Months   100000 non-null  int64
 4   Monthly_Bill                 100000 non-null  float64
 5   Total_Usage_GB               100000 non-null  int64
 6   Churn                        100000 non-null  int64
dtypes: float64(1), int64(4), object(2)
memory usage: 5.3+ MB
```

[14]:
```
df.head()
```

[14]:

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

[15]:
```
churned_LA = df.Churn[(df.Location == 'Los Angeles') & (df.Churn == 1)].count()
count_LA = df.Churn[df.Location == 'Los Angeles'].count()
print("Percent of People Who Churned from Los Angeles --->", churned_LA*100/
 ↪count_LA,'%')



churned_NY = df.Churn[(df.Location == 'New York') & (df.Churn == 1)].count()
count_NY = df.Churn[df.Location == 'New York'].count()
print("Percent of People Who Churned from New York --->", churned_NY*100/
 ↪count_NY,'%')


churned_Mi = df.Churn[(df.Location == 'Miami') & (df.Churn == 1)].count()
count_Mi = df.Churn[df.Location == 'Miami'].count()
print("Percent of People Who Churned from Miami --->", churned_Mi*100/
 ↪count_Mi,'%')


churned_Chi = df.Churn[(df.Location == 'Chicago') & (df.Churn == 1)].count()
count_Chi = df.Churn[df.Location == 'Chicago'].count()
print("Percent of People Who Churned from Chicago --->", churned_Chi*100/
 ↪count_Chi,'%')



churned_H = df.Churn[(df.Location == 'Houston') & (df.Churn == 1)].count()
count_H = df.Churn[df.Location == 'Houston'].count()
print("Percent of People Who Churned from Houston --->", churned_H*100/
 ↪count_H,'%')
```

```
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 %
```

[16]:
```
churned_m = df.Churn[(df.Gender == 'Male') & (df.Churn == 1)].count()
count_m = df.Churn[df.Gender == 'Male'].count()
print("Percent of Males Who Churned --->", churned_m*100/count_m,'%')


churned_f = df.Churn[(df.Gender == 'Female') & (df.Churn == 1)].count()
count_f = df.Churn[df.Gender == 'Female'].count()
print("Percent of Females Who Churned --->", churned_f*100/count_f,'%')
```

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

[17]:
```
from sklearn.preprocessing import StandardScaler,LabelEncoder
```

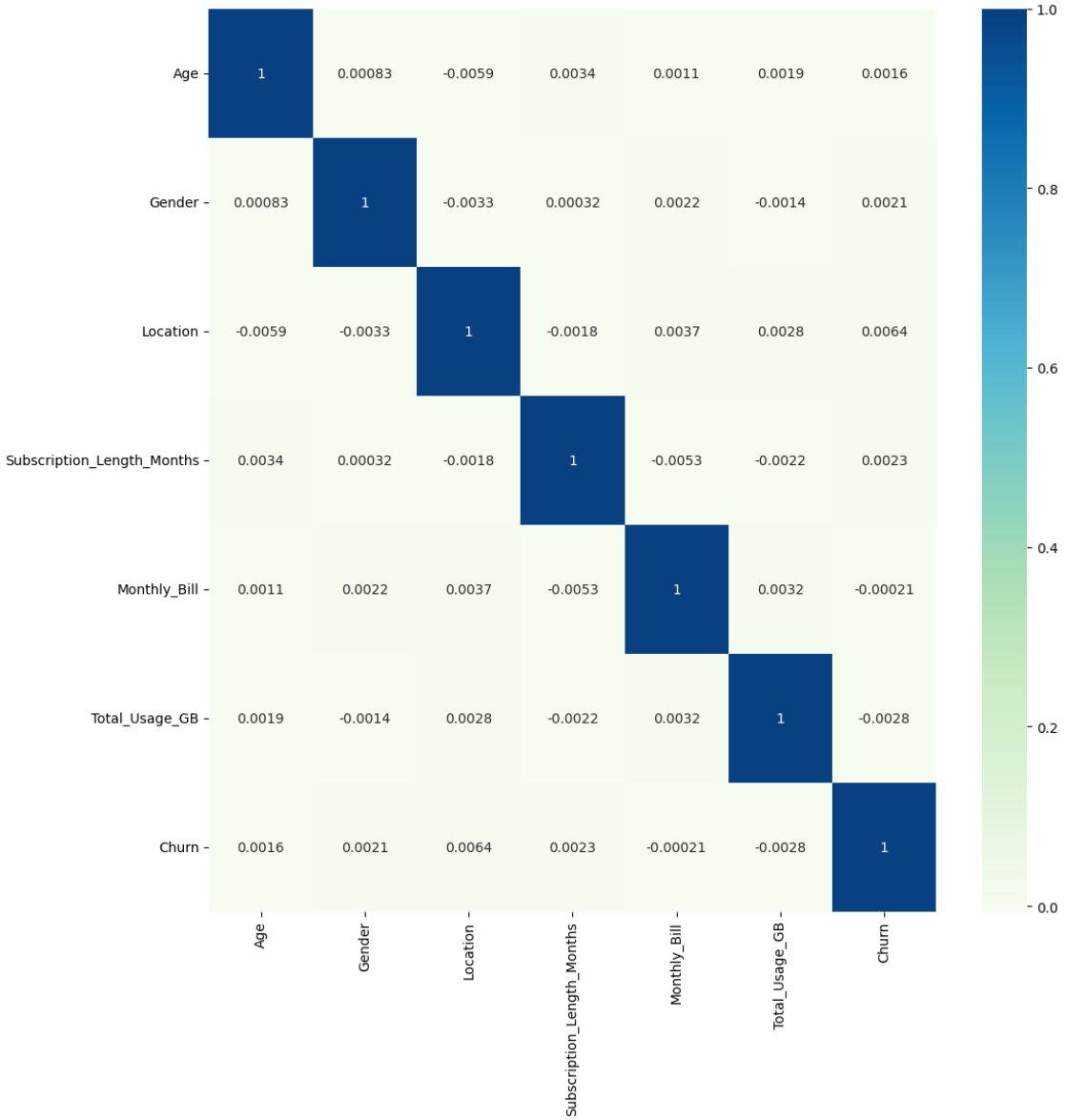
[18]:
```
df['Gender'] = LabelEncoder().fit_transform(df['Gender'])
df['Location'] = LabelEncoder().fit_transform(df['Location'])
```

[19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 7 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   Age                          100000 non-null  int64  
 1   Gender                       100000 non-null  int32  
 2   Location                     100000 non-null  int32  
 3   Subscription_Length_Months   100000 non-null  int64  
 4   Monthly_Bill                 100000 non-null  float64
 5   Total_Usage_GB               100000 non-null  int64  
 6   Churn                        100000 non-null  int64  
dtypes: float64(1), int32(2), int64(4)
memory usage: 4.6 MB
```

[20]:
```
plt.figure(figsize=(12,12))
sns.heatmap(data=df.corr(), annot=True,cmap = "GnBu")
```

[20]:
```
<Axes: >
```




[21]:
```
data_encoded = df.sample(frac=1, random_state=42)
```

[22]:
```
data_encoded.shape
```

[22]: (100000, 7)

[23]:
```
X = data_encoded.drop('Churn', axis=1)
y = data_encoded['Churn']
```

```
from sklearn.model_selection import train_test_split, cross_val_score,␣
 ↪cross_val_predict
from sklearn.metrics import accuracy_score



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=7)
```

[24]:
```
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 : 50.080000000000005 %
```

[25]:
```
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 : 50.105 %
```

[ ]:
```

```