Libraries Required

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

Dataset Import

```
churn_set=pd.read_excel('/content/customer_churn_large_dataset.xlsx')
```

Data Preprocessing

Number of rows and column

```
churn_set.shape
```

```
(100000, 8)
```

Showing first five rows of the data set

```
churn_set.head()
```

```
   CustomerID  Age  Gender     Location  Subscription_Length_Months  \
0           1   63    Male  Los Angeles                          17   
1           2   62  Female     New York                           1   
2           3   24  Female  Los Angeles                           5   
3           4   36  Female        Miami                           3   
4           5   46  Female        Miami                          19   

   Monthly_Bill  Total_Usage_GB  Churn  
0         73.36             236      0  
1         48.76             172      0  
2         85.47             460      0  
3         97.94             297      1  
4         58.14             266      0  
```

Check wheather there is null value in dataset or not

```
churn_set.isnull().sum()
```

```
CustomerID                    0
Age                           0
Gender                        0
Location                      0
Subscription_Length_Months    0
```

```
Monthly_Bill                  0
Total_Usage_GB                0
Churn                         0
dtype: int64
```

Data Analysis and Visualization

Statistical properties of the dataset

```
churn_set.describe()
```

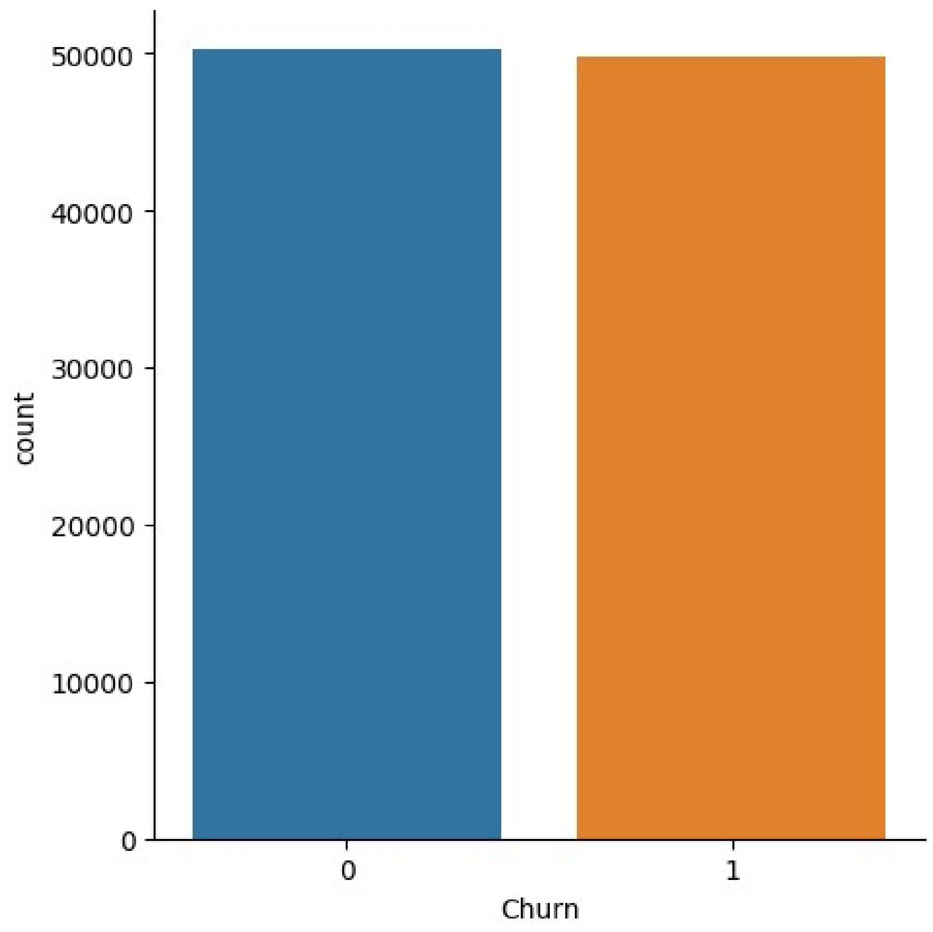
```
          CustomerID            Age  Subscription_Length_Months  \
count  100000.000000  100000.000000               100000.000000
mean    50000.500000      44.027020                   12.490100
std     28867.657797      15.280283                    6.926461
min         1.000000      18.000000                    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
```

Graph plot (To see which properties effect our label)

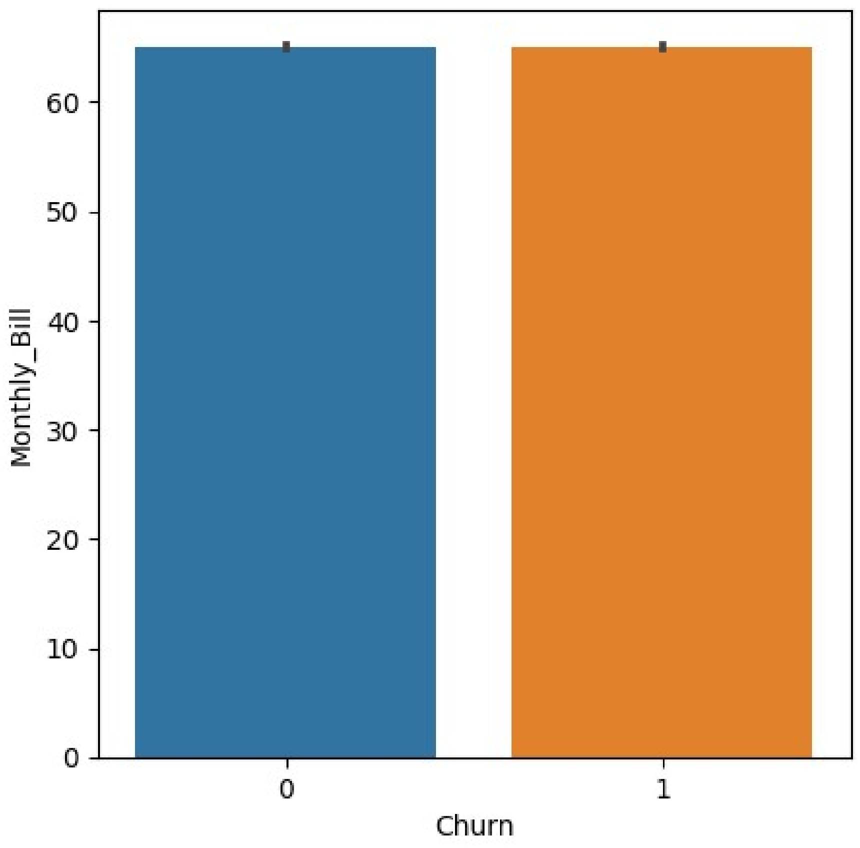
```
sns.catplot(x='Churn',data=churn_set,kind='count')
```

```
<seaborn.axisgrid.FacetGrid at 0x785cca2b2c50>
```



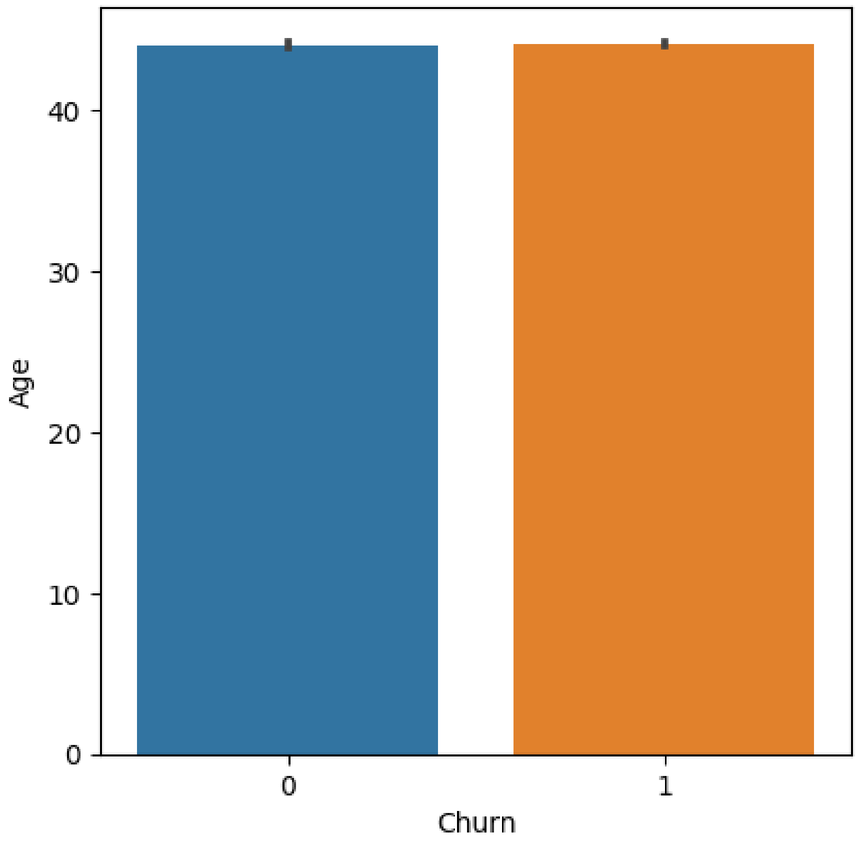
```
plot=plt.figure(figsize=(5,5))
sns.barplot(x='Churn',y='Monthly_Bill',data=churn_set)

<Axes: xlabel='Churn', ylabel='Monthly_Bill'>
```



```
plot=plt.figure(figsize=(5,5))
sns.barplot(x='Churn',y='Age',data=churn_set)
```

```
<Axes: xlabel='Churn', ylabel='Age'>
```

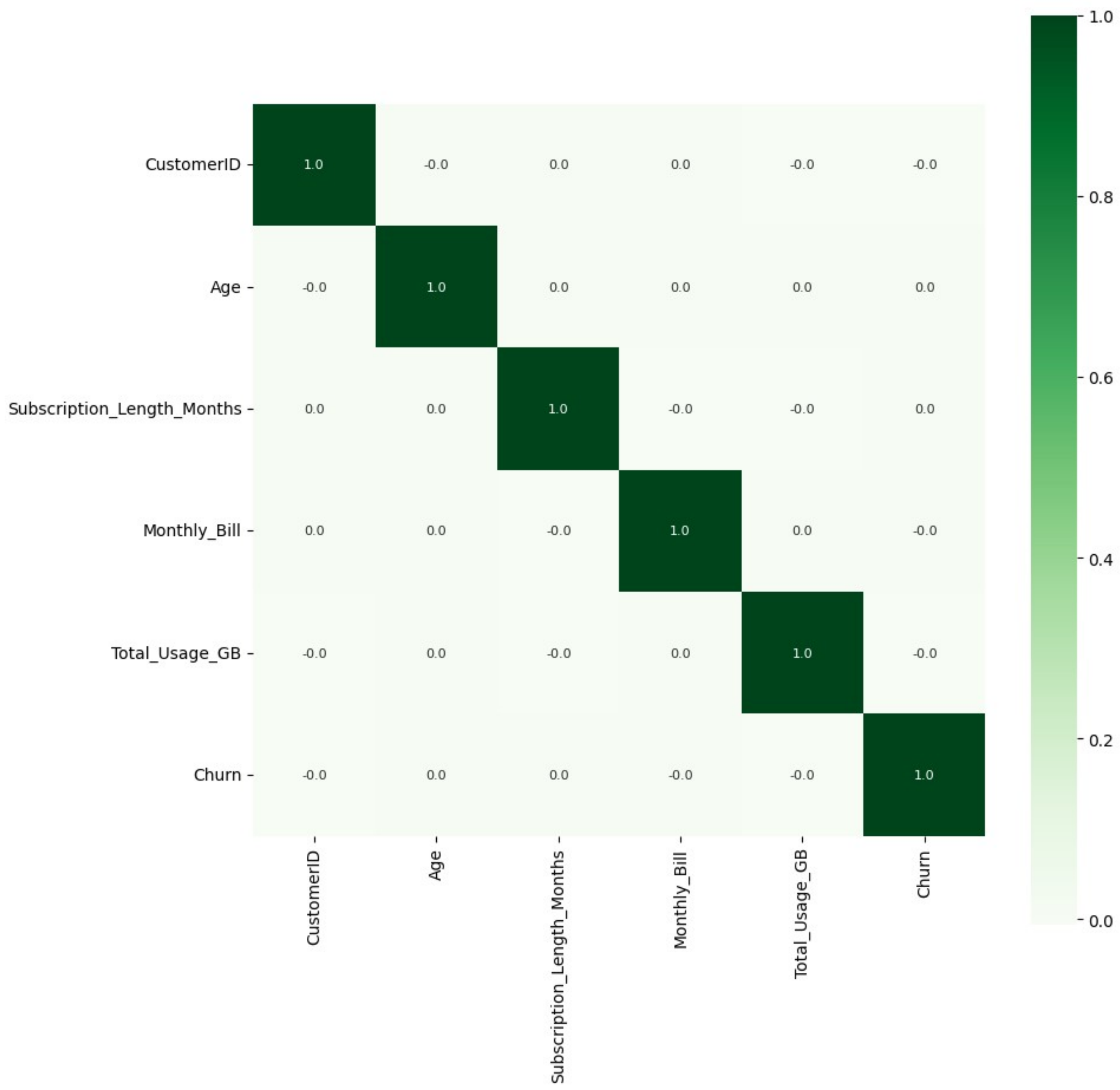


Correlation

```
correlation=churn_set.corr()
plt.figure(figsize=(10,10))
sns.heatmap(correlation,cbar=True,square=True,fmt='.1f',annot=True,ann
ot_kws={'size':8},cmap='Greens')
```

```
<ipython-input-10-580e05bd6ddd>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  correlation=churn_set.corr()

<Axes: >
```



Data Preprocessing

Removing Label for Churn Analysis

```
X = churn_set.drop('Churn',axis=1)

print(X)

       CustomerID  Age  Gender     Location
Subscription_Length_Months  \
0               1   63    Male  Los Angeles
17
1               2   62  Female     New York
1
```

```
2                 3   24   Female   Los Angeles
5
3                 4   36   Female         Miami
3
4                 5   46   Female         Miami
19
...             ...  ...      ...           ...                        ..
.
99995         99996   33     Male       Houston
23
99996         99997   62   Female      New York
19
99997         99998   64     Male       Chicago
17
99998         99999   51   Female      New York
20
99999        100000   27   Female   Los Angeles
19

       Monthly_Bill  Total_Usage_GB
0             73.36             236
1             48.76             172
2             85.47             460
3             97.94             297
4             58.14             266
...             ...             ...
99995         55.13             226
99996         61.65             351
99997         96.11             251
99998         49.25             434
99999         76.57             173

[100000 rows x 7 columns]
```

```python
churn_set["Gender"].value_counts()
```

```
Female    50216
Male      49784
Name: Gender, dtype: int64
```

Label Binarization

```python
Y = churn_set['Churn'].apply(lambda y_value: 1 if y_value>=1 else 0)

print(Y)
```

```
0        0
1        0
2        0
3        1
```

```
4        0
        ..
99995    1
99996    0
99997    1
99998    1
99999    1
Name: Churn, Length: 100000, dtype: int64
```

Model Training and Testing

Splitting data into training and testing

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,rando
m_state=1)

print(Y.shape,Y_train.shape,Y_test.shape)

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

Model: Random Forest Classifier

```
from sklearn.ensemble import RandomForestRegressor

model=RandomForestRegressor()
model.fit(X_train,Y_train)
model.score(X_test,Y_test)

----------------------------------------------------------------------
-----
ValueError                               Traceback (most recent call
last)
<ipython-input-19-86405439abb2> in <cell line: 2>()
      1 model=RandomForestRegressor()
----> 2 model.fit(X_train,Y_train)
      3 model.score(X_test,Y_test)

/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py in
fit(self, X, y, sample_weight)
    343         if issparse(y):
    344             raise ValueError("sparse multilabel-indicator for
y is not supported.")
--> 345         X, y = self._validate_data(
    346             X, y, multi_output=True, accept_sparse="csc",
dtype=DTYPE
    347         )

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in
_validate_data(self, X, y, reset, validate_separately, **check_params)
    582                 y = check_array(y, input_name="y",
**check_y_params)
    583             else:
--> 584                 X, y = check_X_y(X, y, **check_params)
    585             out = X, y
    586 

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in
check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order,
copy, force_all_finite, ensure_2d, allow_nd, multi_output,
ensure_min_samples, ensure_min_features, y_numeric, estimator)
   1104         )
   1105 
-> 1106     X = check_array(
   1107         X,
   1108         accept_sparse=accept_sparse,

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in
check_array(array, accept_sparse, accept_large_sparse, dtype, order,
copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples,
ensure_min_features, estimator, input_name)
    877                     array = xp.astype(array, dtype,
copy=False)
    878                 else:
--> 879                     array = _asarray_with_order(array,
order=order, dtype=dtype, xp=xp)
    880             except ComplexWarning as complex_warning:
    881                 raise ValueError(

/usr/local/lib/python3.10/dist-packages/sklearn/utils/_array_api.py in
_asarray_with_order(array, dtype, order, copy, xp)
    183     if xp.__name__ in {"numpy", "numpy.array_api"}:
    184         # Use NumPy API to support order
--> 185         array = numpy.asarray(array, order=order, dtype=dtype)
    186         return xp.asarray(array, copy=copy)
    187     else:

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in
__array__(self, dtype)
   2068 
   2069     def __array__(self, dtype: npt.DTypeLike | None = None) ->
np.ndarray:
-> 2070         return np.asarray(self._values, dtype=dtype)
   2071 
   2072     def __array_wrap__(

ValueError: could not convert string to float: 'Male'

from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
categorical_features=["Gender","Location"]
```

```
one_hot=OneHotEncoder()
transformer=ColumnTransformer([("one_hot",
                                one_hot,
                                categorical_features)],
                                remainder='passthrough')
transformed_X=transformer.fit_transform(X)
transformed_X
```

```
array([[  0.  ,   1.  ,   0.  , ...,  17.  ,  73.36, 236.  ],
       [  1.  ,   0.  ,   0.  , ...,   1.  ,  48.76, 172.  ],
       [  1.  ,   0.  ,   0.  , ...,   5.  ,  85.47, 460.  ],
       ...,
       [  0.  ,   1.  ,   1.  , ...,  17.  ,  96.11, 251.  ],
       [  1.  ,   0.  ,   0.  , ...,  20.  ,  49.25, 434.  ],
       [  1.  ,   0.  ,   0.  , ...,  19.  ,  76.57, 173.  ]])
```

```
X.head()
```

```
   CustomerID  Age  Gender     Location  Subscription_Length_Months  \
0           1   63    Male  Los Angeles                          17
1           2   62  Female     New York                           1
2           3   24  Female  Los Angeles                           5
3           4   36  Female        Miami                           3
4           5   46  Female        Miami                          19

   Monthly_Bill  Total_Usage_GB
0         73.36             236
1         48.76             172
2         85.47             460
3         97.94             297
4         58.14             266
```

```
pd.DataFrame(transformed_X)
```

```
        0    1    2    3    4    5    6        7     8     9     10
11
0      0.0  1.0  0.0  0.0  1.0  0.0  0.0      1.0  63.0  17.0  73.36
236.0
1      1.0  0.0  0.0  0.0  0.0  0.0  1.0      2.0  62.0   1.0  48.76
172.0
2      1.0  0.0  0.0  0.0  1.0  0.0  0.0      3.0  24.0   5.0  85.47
460.0
3      1.0  0.0  0.0  0.0  0.0  1.0  0.0      4.0  36.0   3.0  97.94
297.0
4      1.0  0.0  0.0  0.0  0.0  1.0  0.0      5.0  46.0  19.0  58.14
266.0
...    ...  ...  ...  ...  ...  ...  ...      ...   ...   ...    ...
...
99995  0.0  1.0  0.0  1.0  0.0  0.0  0.0  99996.0  33.0  23.0  55.13
226.0
```

```
99996  1.0  0.0  0.0  0.0  0.0  0.0  1.0   99997.0  62.0  19.0  61.65
351.0
99997  0.0  1.0  1.0  0.0  0.0  0.0  0.0   99998.0  64.0  17.0  96.11
251.0
99998  1.0  0.0  0.0  0.0  0.0  0.0  1.0   99999.0  51.0  20.0  49.25
434.0
99999  1.0  0.0  0.0  0.0  1.0  0.0  0.0  100000.0  27.0  19.0  76.57
173.0

[100000 rows x 12 columns]
```

```
dummies=pd.get_dummies(churn_set[["Gender","Location"]])
dummies
```

```
       Gender_Female  Gender_Male  Location_Chicago  Location_Houston
\
0                  0            1                 0                 0

1                  1            0                 0                 0

2                  1            0                 0                 0

3                  1            0                 0                 0

4                  1            0                 0                 0

...              ...          ...               ...               ...

99995              0            1                 0                 1

99996              1            0                 0                 0

99997              0            1                 1                 0

99998              1            0                 0                 0

99999              1            0                 0                 0


       Location_Los Angeles  Location_Miami  Location_New York
0                         1               0                  0
1                         0               0                  1
2                         1               0                  0
3                         0               1                  0
4                         0               1                  0
...                     ...             ...                ...
99995                     0               0                  0
99996                     0               0                  1
99997                     0               0                  0
99998                     0               0                  1
99999                     1               0                  0
```

```
[100000 rows x 7 columns]

np.random.seed(24)
X_train,X_test,Y_train_Train,Y_Test = train_test_split(transformed_X,
                                                       Y,
                                                       test_size=0.2)
model.fit(X_train,Y_train)

RandomForestRegressor()

model.score(X_test,Y_test)

-0.03346709386837565

X.head()

   CustomerID  Age  Gender     Location  Subscription_Length_Months  \
0           1   63    Male  Los Angeles                          17
1           2   62  Female     New York                           1
2           3   24  Female  Los Angeles                           5
3           4   36  Female        Miami                           3
4           5   46  Female        Miami                          19

   Monthly_Bill  Total_Usage_GB
0         73.36             236
1         48.76             172
2         85.47             460
3         97.94             297
4         58.14             266
```

Predictive System

```
input_data=(2,62,1,0,0,0,0,1,0,1,48.76,172)

input_data_as_numpy_array = np.asarray(input_data)

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if(prediction[0]==1):
  print('Churned')
else:
  print('Not Churned')

[0.53]
Not Churned
```