```
# Ignore  the warnings
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')

# data visualisation and manipulation
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

import missingno as msno

# configure
# sets matplotlib to inline and displays graphs below the corresponding cell.
%matplotlib inline
style.use('fivethirtyeight')
sns.set(style='whitegrid',color_codes=True)

# import the necessary modelling algos.
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC, SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

# model selection
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, roc_curve, roc_auc_score
from sklearn.model_selection import GridSearchCV

from imblearn.over_sampling import SMOTE

# preprocess
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler, StandardScaler, LabelEncoder, OneHotEncoder

# ANN and DL libraries
from keras import backend as K
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam, SGD, Adagrad, Adadelta, RMSprop
from keras.utils import to_categorical

import tensorflow as tf
import random as rn



df=pd.read_excel('/content/customer_churn_large_dataset.xlsx')


df.head ()
```

	CustomerID	Name	Age	Gender	Location	Subscription_Length_Months	Mont
0	1	Customer_1	63	Male	Los Angeles	17	
1	2	Customer_2	62	Female	New York	1	
2	3	Customer_3	24	Female	Los Angeles	5	
3	4	Customer_4	36	Female	Miami	3	

```
df.shape
```

```
(100000, 9)
```

```
df.columns
```

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

```
df.info()  # no null or Nan values.
```

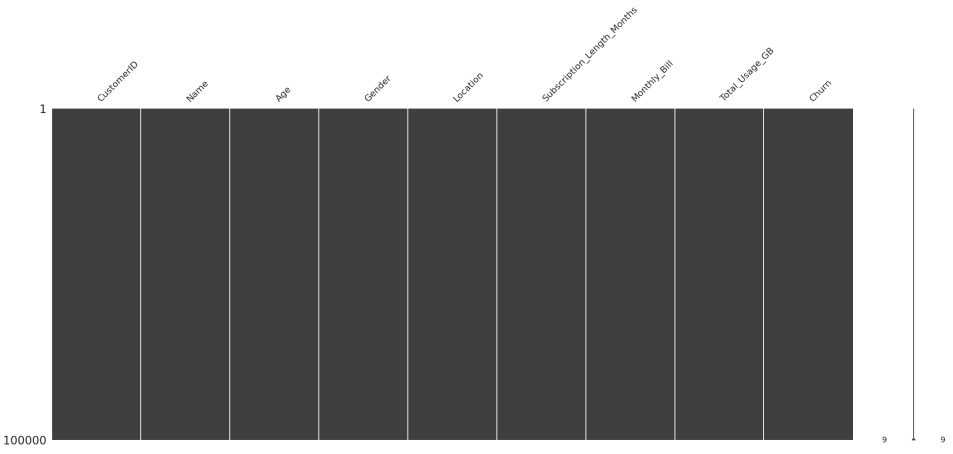
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   CustomerID                  100000 non-null  int64  
 1   Name                        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
```

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

```
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
```

```
msno.matrix(df) # just to visualize.
```

```
<Axes: >
```




```
df.columns
```

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

```
df.head()
```

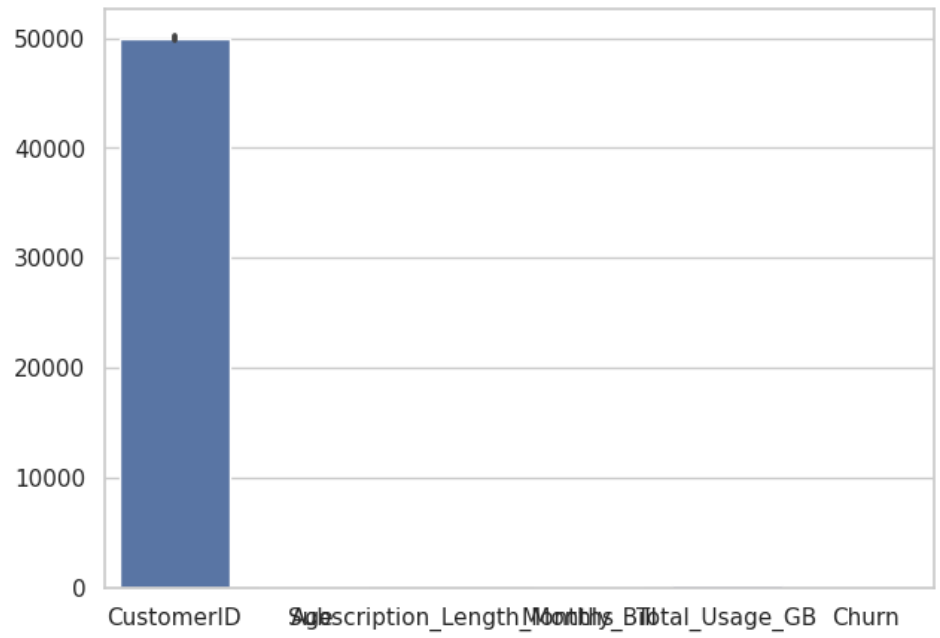
	CustomerID	Name	Age	Gender	Location	Subscription_Length_Months	Mont
0	1	Customer_1	63	Male	Los Angeles	17	
1	2	Customer_2	62	Female	New York	1	
2	3	Customer_3	24	Female	Los Angeles	5	
3	4	Customer_4	36	Female	Miami	3	

```
df.describe()
```

	CustomerID	Age	Subscription_Length_Months	Monthly_Bill	Tot
count	100000.000000	100000.000000	100000.000000	100000.000000	1(
mean	50000.500000	44.027020	12.490100	65.053197	
std	28867.657797	15.280283	6.926461	20.230696	
min	1.000000	18.000000	1.000000	30.000000	
25%	25000.750000	31.000000	6.000000	47.540000	
50%	50000.500000	44.000000	12.000000	65.010000	
75%	75000.250000	57.000000	19.000000	82.640000	
max	100000.000000	70.000000	24.000000	100.000000	

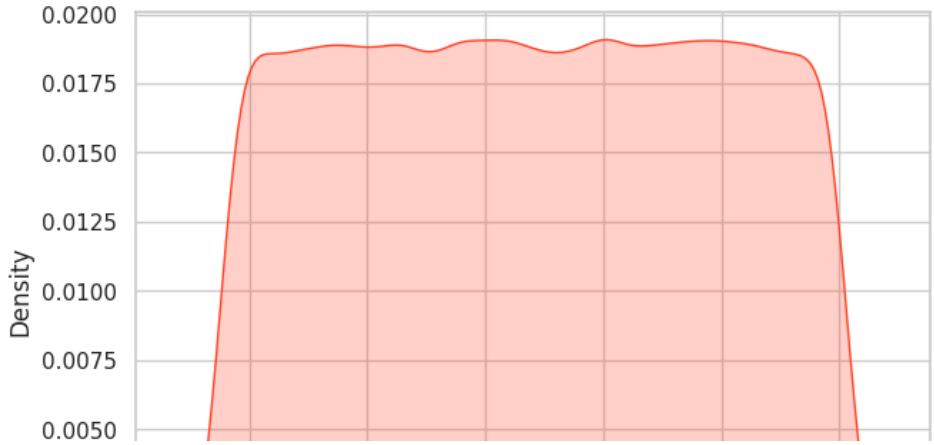
```
sns.barplot(data=df)
```

```
<Axes: >
```



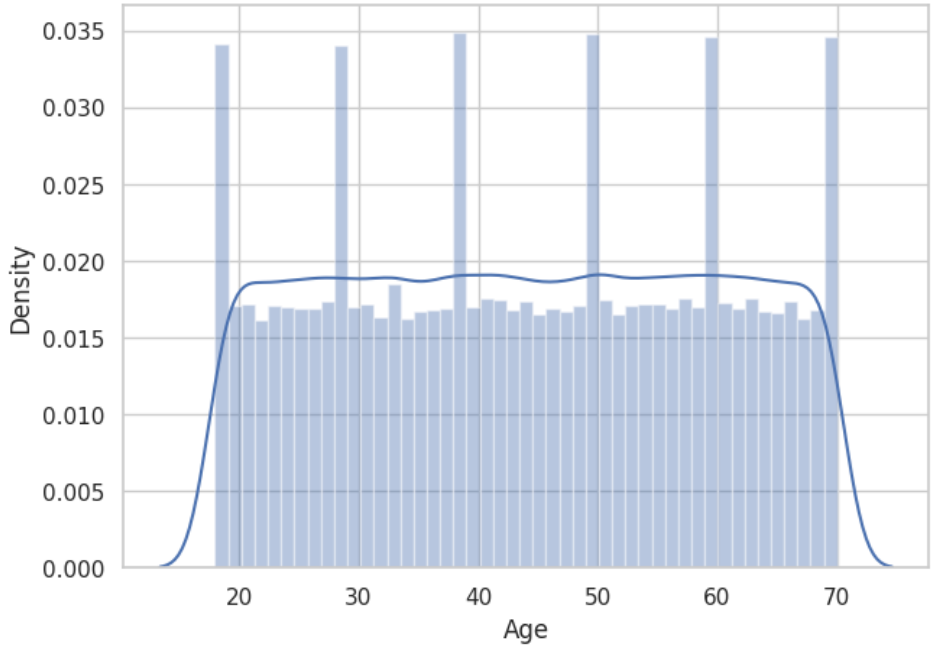

```
sns.kdeplot(df['Age'],shade=True,color='#ff4125')
```

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




```
sns.distplot(df['Age'])
```

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

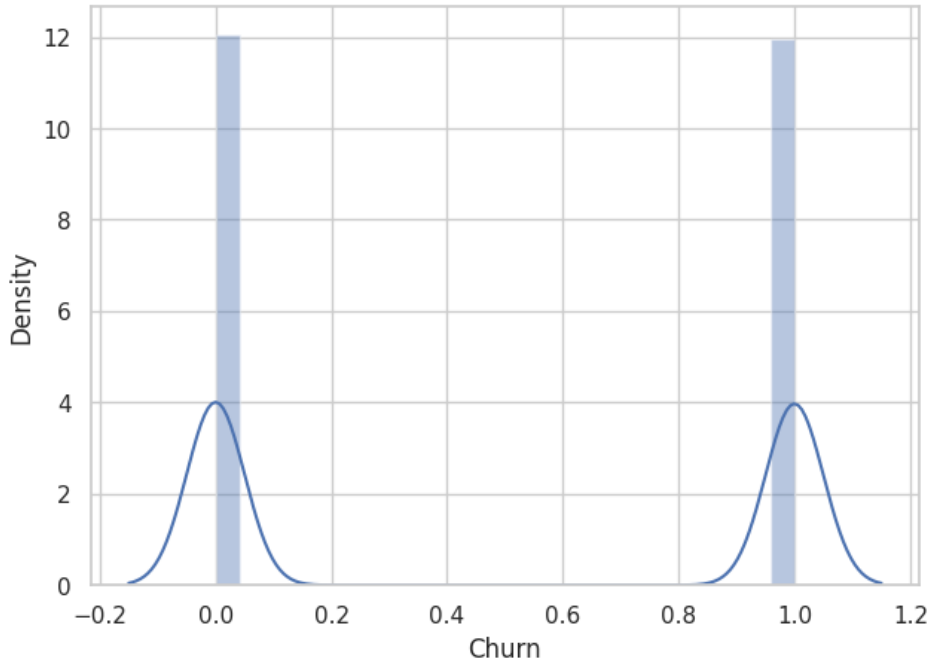



```
sns.distplot(df['Total_Usage_GB'])
```

```
<Axes: xlabel='Total_Usage_GB', ylabel='Density'>
```

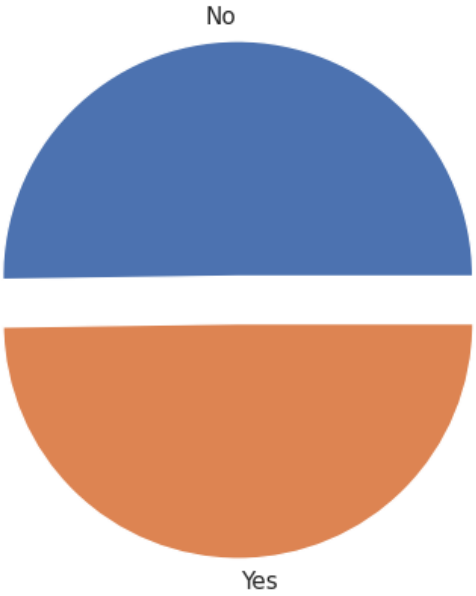
```
sns.distplot(df['Churn'])
```

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



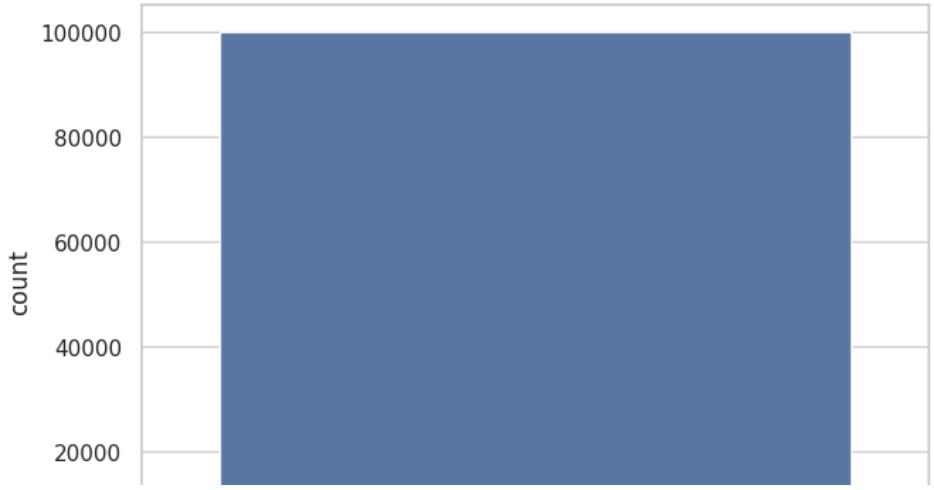

```
attrition_count = pd.DataFrame(df['Churn'].value_counts())
attrition_count
import seaborn as sns
import matplotlib.pyplot as plt
plt.pie(attrition_count['Churn'], labels =['No', 'Yes'], explode = (0.2,0))
```

```
([<matplotlib.patches.Wedge at 0x7a9413c09ba0>,
  <matplotlib.patches.Wedge at 0x7a9413c09ab0>],
 [Text(-0.009025809021426776, 1.2999686668422084, 'No'),
  Text(0.007637326005080358, -1.0999734866129693, 'Yes')])
```




```
sns.countplot(df['Churn'])
```

```
<Axes: ylabel='count'>
```




```
df.drop(['CustomerID'], axis = 1)
```

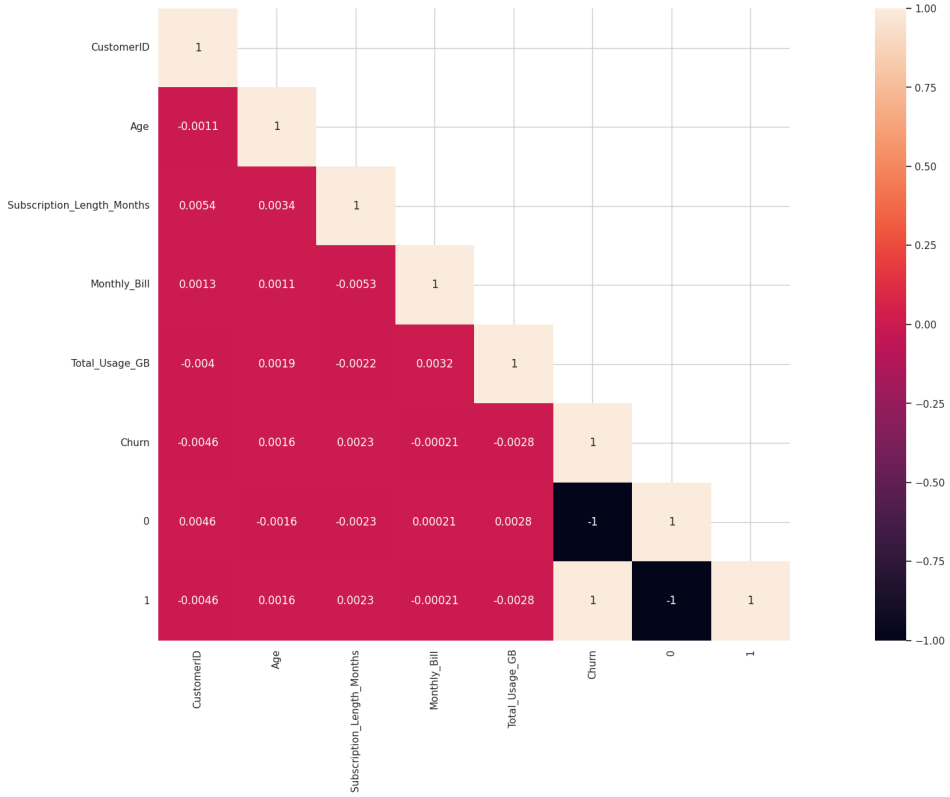
	Name	Age	Gender	Location	Subscription_Length_Months	Monthly
0	Customer_1	63	Male	Los Angeles	17	
1	Customer_2	62	Female	New York	1	
2	Customer_3	24	Female	Los Angeles	5	
3	Customer_4	36	Female	Miami	3	
4	Customer_5	46	Female	Miami	19	
...	...	...	...	...	...	
99995	Customer_99996	33	Male	Houston	23	
99996	Customer_99997	62	Female	New York	19	
99997	Customer_99998	64	Male	Chicago	17	
99998	Customer_99999	51	Female	New York	20	
				Los		

```
attrition_dummies = pd.get_dummies(df['Churn'])
attrition_dummies.head()
df= pd.concat([df, attrition_dummies], axis = 1)
df.head()
```

	CustomerID	Name	Age	Gender	Location	Subscription_Length_Months	Mont
0	1	Customer_1	63	Male	Los Angeles	17	
1	2	Customer_2	62	Female	New York	1	
2	3	Customer_3	24	Female	Los Angeles	5	
3	4	Customer_4	36	Female	Miami	3	
4	5	Customer_5	46	Female	Miami	19	

```
#corelation matrix.
cor_mat= df.corr()
mask = np.array(cor_mat)
mask[np.tril_indices_from(mask)] = False
fig=plt.gcf()
fig.set_size_inches(30,12)
sns.heatmap(data=cor_mat,mask=mask,square=True,annot=True,cbar=True)
```

```
<Axes: >
```




```
def transform(feature):
    le=LabelEncoder()
    df[feature]=le.fit_transform(df[feature])
    print(le.classes_)


cat_df=df.select_dtypes(include='object')
cat_df.columns
```

```
Index(['Name', 'Gender', 'Location'], dtype='object')
```

```
for col in cat_df.columns:
    transform(col)
```

```
['Customer_1' 'Customer_10' 'Customer_100' ... 'Customer_99997'
 'Customer_99998' 'Customer_99999']
['Female' 'Male']
['Chicago' 'Houston' 'Los Angeles' 'Miami' 'New York']
```

```
X = df.drop('Churn', axis=1)  # X contains all columns except 'churn'
y = df['Churn']


x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=42)


x_train.columns = x_train.columns.astype(str)
x_test.columns = x_test.columns.astype(str)
```

```
from sklearn.ensemble import RandomForestClassifier

# Create the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model on the training data
rf_model.fit(x_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
from sklearn.metrics import precision_score, recall_score, f1_score
rf_model.fit(x_train, y_train)
y_pred = rf_model.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
classification_rep =(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
print("Confusion Matrix:")
print(confusion)
print("Classification Report:")
print(classification_rep)
precision =(precision_score(y_test, y_pred))
print(f"Precision: {precision:.2f}")
recall =(recall_score(y_test, y_pred))
print(f"Recall: {recall:.2f}")
f1 = f1_score(y_test, y_pred)
print(f"F1-score: {f1:.2f}")
```

```
Accuracy: 1.00
Confusion Matrix:
[[12578     0]
 [    0 12422]]
Classification Report:
(75721    0
80184    0
19864    0
76699    1
92991    0
        ..
21271    0
34014    0
81355    1
65720    0
11627    1
Name: Churn, Length: 25000, dtype: int64, array([0, 0, 0, ..., 1, 0, 1]))
Precision: 1.00
Recall: 1.00
F1-score: 1.00
```

```
# using ANN
model=Sequential()
model.add(Dense(input_dim=10,units=8,activation='relu'))
model.add(Dense(units=23,activation='relu'))
model.add(Dense(units=23,activation='sigmoid'))


model.compile(optimizer=Adam(lr=0.01),loss='binary_crossentropy',metrics=['accuracy'])


model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 8)                 88

 dense_13 (Dense)            (None, 23)                207

 dense_14 (Dense)            (None, 23)                552
```

```
=================================================================
Total params: 847
Trainable params: 847
Non-trainable params: 0
_________________________________________________________________
```

✓ 1s completed at 18:03