In [2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```python
data = pd.read_csv("C:\\Users\\vaibhav vishal\\OneDrive\\Documents\\Churn_M
```

In [4]:
```python
data.head()
```

Out[4]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	8380
2	3	15619304	Onio	502	France	Female	42	8	15966
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	12551

In [5]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [7]:
```
data.columns
```

Out[7]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [8]:
```
data = data.drop(['RowNumber', 'CustomerId', 'Surname'],axis=1)
data
```

Out[8]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	619	France	Female	42	2	0.00	1	1
1	608	Spain	Female	41	1	83807.86	1	0
2	502	France	Female	42	8	159660.80	3	1
3	699	France	Female	39	1	0.00	2	0
4	850	Spain	Female	43	2	125510.82	1	1
...	...	...	...	...	...	...	...	...
9995	771	France	Male	39	5	0.00	2	1
9996	516	France	Male	35	10	57369.61	1	1
9997	709	France	Female	36	7	0.00	1	0
9998	772	Germany	Male	42	3	75075.31	2	1
9999	792	France	Female	28	4	130142.79	1	1

10000 rows × 11 columns

In [9]:
```
data = pd.get_dummies(data,drop_first = True)
data.head()
data = data.astype(int)
data
```

Out[9]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estin
0	619	42	2	0	1	1	1	
1	608	41	1	83807	1	0	1	
2	502	42	8	159660	3	1	0	
3	699	39	1	0	2	0	0	
4	850	43	2	125510	1	1	1	
...	...	...	...	...	...	...	...	
9995	771	39	5	0	2	1	0	
9996	516	35	10	57369	1	1	1	
9997	709	36	7	0	1	0	1	
9998	772	42	3	75075	2	1	0	
9999	792	28	4	130142	1	1	0	

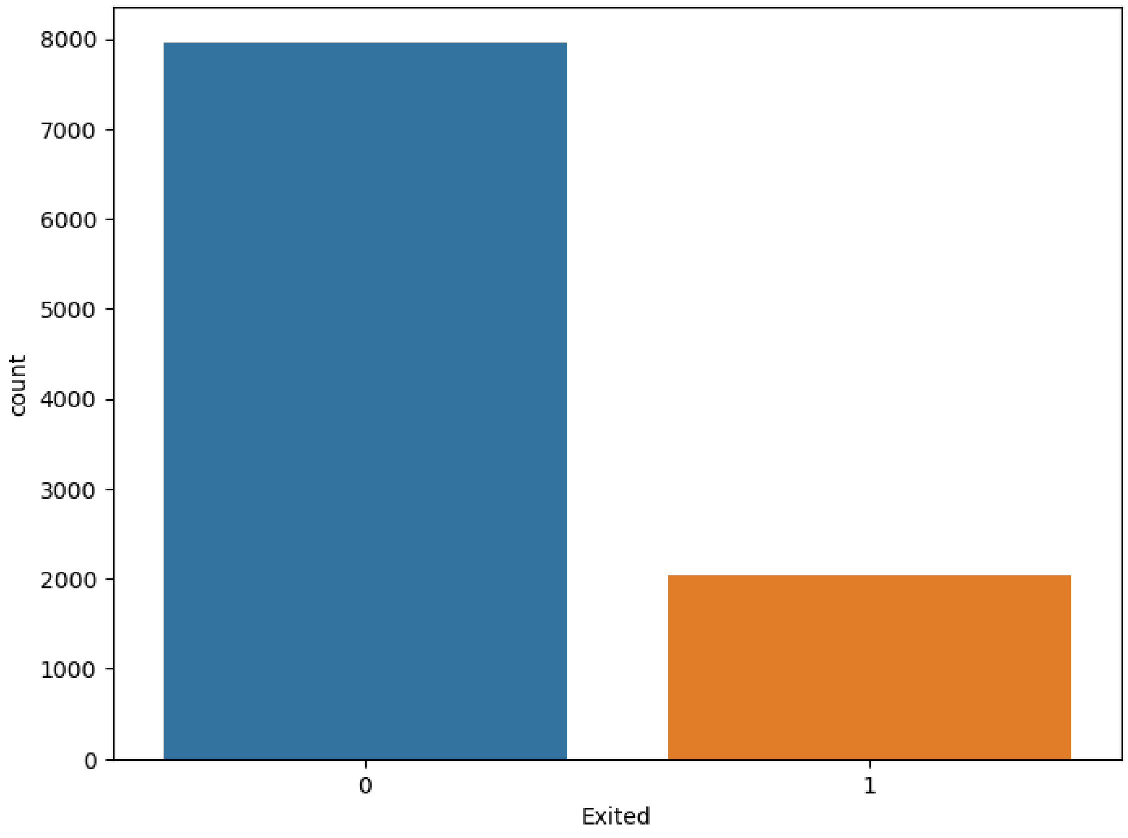
10000 rows × 12 columns

In [10]:
```
data['Exited'].value_counts()
```

Out[10]:
```
Exited
0    7963
1    2037
Name: count, dtype: int64
```

In [11]:
```
plt.figure(figsize =(8,6))
sns.countplot(x='Exited',data = data)
```

Out[11]:
```
<Axes: xlabel='Exited', ylabel='count'>
```



In [12]:
```
X = data.drop('Exited',axis=1)
y = data['Exited']
```

In [13]:
```
pip install imblearn
```

```
Defaulting to user installation because normal site-packages is not writea
ble
Collecting imblearn
  Obtaining dependency information for imblearn from https://files.pythonh
osted.org/packages/81/a7/4179e6ebfd654bd0eac0b9c06125b8b4c96a9d0a8ff9e9507
eb2a26d2d7e/imblearn-0.0-py2.py3-none-any.whl.metadata (https://files.pyth
onhosted.org/packages/81/a7/4179e6ebfd654bd0eac0b9c06125b8b4c96a9d0a8ff9e9
507eb2a26d2d7e/imblearn-0.0-py2.py3-none-any.whl.metadata)
  Downloading imblearn-0.0-py2.py3-none-any.whl.metadata (355 bytes)
Requirement already satisfied: imbalanced-learn in c:\programdata\anaconda
3\lib\site-packages (from imblearn) (0.10.1)
Requirement already satisfied: numpy>=1.17.3 in c:\programdata\anaconda3\l
ib\site-packages (from imbalanced-learn->imblearn) (1.24.3)
Requirement already satisfied: scipy>=1.3.2 in c:\programdata\anaconda3\li
b\site-packages (from imbalanced-learn->imblearn) (1.11.1)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\programdata\anaco
nda3\lib\site-packages (from imbalanced-learn->imblearn) (1.3.0)
Requirement already satisfied: joblib>=1.1.1 in c:\programdata\anaconda3\l
ib\site-packages (from imbalanced-learn->imblearn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anac
onda3\lib\site-packages (from imbalanced-learn->imblearn) (2.2.0)
Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Installing collected packages: imblearn
Successfully installed imblearn-0.0
Note: you may need to restart the kernel to use updated packages.
```

In [15]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression , LogisticRegression
from sklearn.metrics import r2_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingRegres
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, ra
print('Training Shape: ', X_train.shape)
print('Testing Shape: ', X_test.shape)
```

```
Training Shape:  (9000, 11)
Testing Shape:  (1000, 11)
```

In [17]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [18]:
```
X_train_scaled
```

Out[18]:
```
array([[-0.47944328,  0.19687202, -0.00234647, ..., -0.5761528 ,
        -0.57700814,  0.91105005],
       [ 1.04580863,  1.33803657,  1.03625698, ..., -0.5761528 ,
        -0.57700814, -1.09763453],
       [-0.85297437, -0.08841912,  1.03625698, ..., -0.5761528 ,
         1.73307782,  0.91105005],
       ...,
       [ 0.86941896, -0.08841912, -1.38715108, ..., -0.5761528 ,
        -0.57700814, -1.09763453],
       [ 0.16386025,  0.38706611,  1.03625698, ..., -0.5761528 ,
        -0.57700814,  0.91105005],
       [ 0.47513615,  1.14784248, -1.38715108, ...,  1.73565068,
        -0.57700814,  0.91105005]])
```

In [33]:
```
from sklearn import svm
threshold = 0.5
y_train_classified = [1 if value > threshold else 0 for value in y_train]
svm = svm.SVC()
svm.fit(X_train_scaled, y_train_classified)
```

Out[33]:
```
SVC()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [34]:
```
y_test_classified = [1 if value > threshold else 0 for value in y_test]
accuracy2 = svm.score(X_test_scaled, y_test_classified)
print("Model Accuracy:", accuracy1)
```

```
Model Accuracy: 0.809
```

In [ ]:
```

```

In [35]:
```
threshold = 0.5
y_train_classified = [1 if value > threshold else 0 for value in y_train]
LR = LogisticRegression()
LR.fit(X_train_scaled, y_train_classified)
```

Out[35]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [36]:
```
y_test_classified = [1 if value > threshold else 0 for value in y_test]
accuracy1 = LR.score(X_test_scaled, y_test_classified)
print("Model Accuracy:", accuracy2)
```

```
Model Accuracy: 0.865
```

In [37]:
```
threshold = 0.5
y_train_classified = [1 if value > threshold else 0 for value in y_train]
dt = DecisionTreeClassifier()
dt.fit(X_train_scaled, y_train_classified)
```

Out[37]:
```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [38]:
```
y_test_classified = [1 if value > threshold else 0 for value in y_test]
accuracy4 = dt.score(X_test_scaled, y_test_classified)
print("Model Accuracy:", accuracy3)
```

```
Model Accuracy: 0.863
```

In [39]:
```
threshold = 0.5
y_train_classified = [1 if value > threshold else 0 for value in y_train]
rf = RandomForestClassifier()
rf.fit(X_train_scaled, y_train_classified)
```

Out[39]:
```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [40]:
```
y_test_classified = [1 if value > threshold else 0 for value in y_test]
accuracy3 = rf.score(X_test_scaled, y_test_classified)
print("Model Accuracy:", accuracy4)
```

```
Model Accuracy: 0.797
```

In [41]:
```
threshold = 0.5
y_train_classified = [1 if value > threshold else 0 for value in y_train]
KNN = KNeighborsClassifier()
KNN.fit(X_train_scaled, y_train_classified)
```

Out[41]:
```
KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [42]:
```
y_test_classified = [1 if value > threshold else 0 for value in y_test]
accuracy5 = KNN.score(X_test_scaled, y_test_classified)
print("Model Accuracy:", accuracy5)
```

```
Model Accuracy: 0.84
```

In [43]:
```
from sklearn.ensemble import GradientBoostingClassifier
threshold = 0.5
y_train_classified = [1 if value > threshold else 0 for value in y_train]
GBC = GradientBoostingClassifier()
GBC.fit(X_train_scaled, y_train_classified)
```

Out[43]:
```
GradientBoostingClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [44]:
```
y_test_classified = [1 if value > threshold else 0 for value in y_test]
accuracy6 = GBC.score(X_test_scaled, y_test_classified)
print("Model Accuracy:", accuracy6)
```

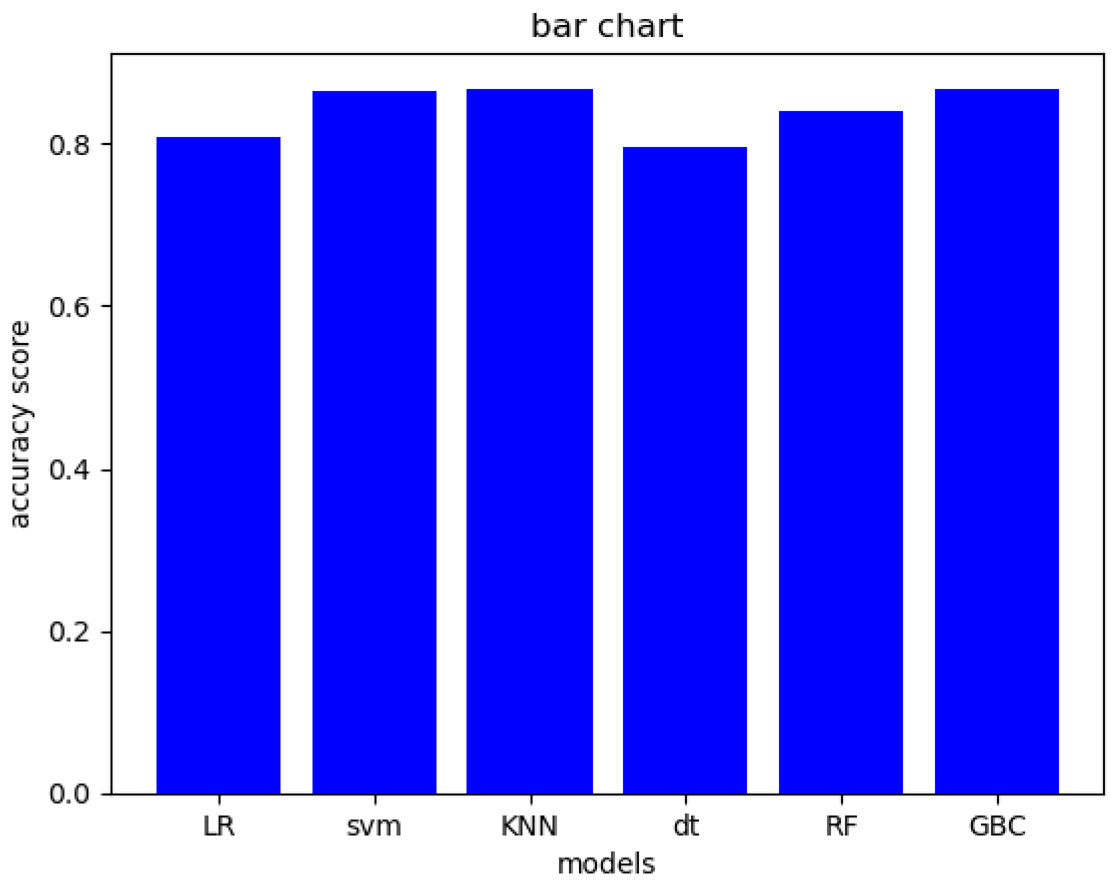
```
Model Accuracy: 0.867
```

In [45]:
```
performance_summary = pd.DataFrame({
    'Model':['LR','svm','KNN','dt','rf','GBC'],
    'ACC':[accuracy1,
           accuracy2,
           accuracy3,
           accuracy4,
           accuracy5,
           accuracy6
          ]
})
performance_summary
```

Out[45]:

	Model	ACC
0	LR	0.809
1	svm	0.865
2	KNN	0.868
3	dt	0.797
4	rf	0.840
5	GBC	0.867

```
In [46]: x=['LR','svm','KNN','dt','RF','GBC']
         y=[0.809,0.865,0.868,0.797,0.840,0.867]
         plt.bar(x,y,color='blue')
         plt.xlabel('models')
         plt.ylabel('accuracy score')
         plt.title('bar chart')
         plt.show()
```



```
In [ ]:
```