importing essential librariess for build a ml model

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

In [3]:
```
df = pd.read_csv('Churn_Modelling.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

In [4]:
```
#shape of the dataset
df.shape
```

Out[4]:
```
(10000, 14)
```

In [5]:
```
#give the information about the attributes of the dataset
df.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 [ ]:
```
feature information checking null values
```

In [6]:
```
df.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]:
```
df.size
```

Out[7]:
```
140000
```

In [12]:
```
df.columns
```

Out[12]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [15]:
```
#drawing columns which are not necessary for prediction
df.dropna(inplace=True)
df.drop(columns=['RowNumber','CustomerId','Surname'],axis=1,inplace=True)
```

In [16]:
```
df.head()
```

Out[16]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

In [10]:
```
#give null value info
df.isnull().sum()
```

Out[10]:
```
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 [11]:
```
#give statistical summary of data
df.describe()
```

Out[11]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

In [12]:
```
df.tail()
```

Out[12]:

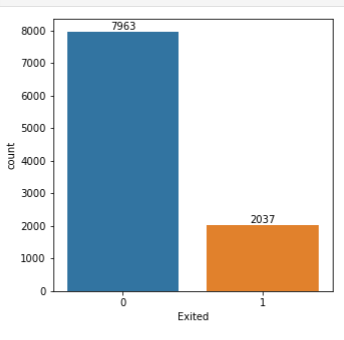
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

In [13]:
```
#EDA (exploratory data analysis)
df['Exited'].value_counts(normalize=True)*100
```

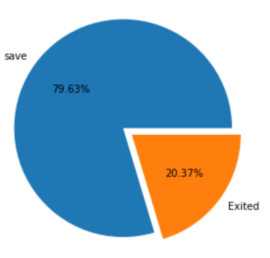
Out[13]:
```
0    79.63
1    20.37
Name: Exited, dtype: float64
```

In [14]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [15]:
```
plt.figure(figsize = (5,5))
ax = sns.countplot(x = df ['Exited'],data=df)
for label in ax.containers:ax.bar_label(label);
```



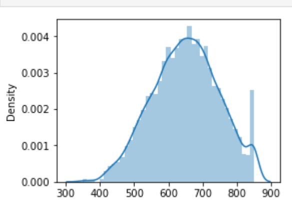
In [16]:
```
plt.figure(figsize=(5,5))
target_val = df ['Exited'].value_counts().values
target_name = df['Exited'].value_counts().index
plt.pie(x = target_val,labels=['save','Exited'],autopct = '%1.2f%%',explode=[0.1,0])
plt.show()
```



In [17]:
```
print('skewness:',df['CreditScore'].skew())
print('mode:',df['CreditScore'].mode())
```

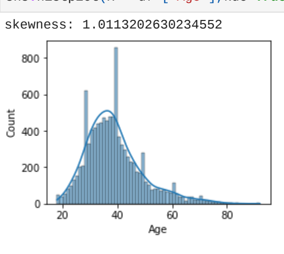
```
skewness: -0.07160660820092675
mode: 0    850
Name: CreditScore, dtype: int64
```

In [18]:
```
plt.figure(figsize=(4,3))
sns.distplot(x =df ['CreditScore']);
```

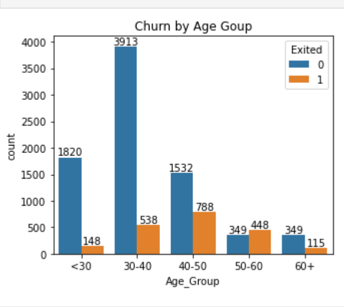


In [19]:
```
#handling cateorical columns

print('skewness:',df['Age'].skew())
plt.figure(figsize = (4,3))
sns.histplot(x = df ['Age'],kde=True);
```

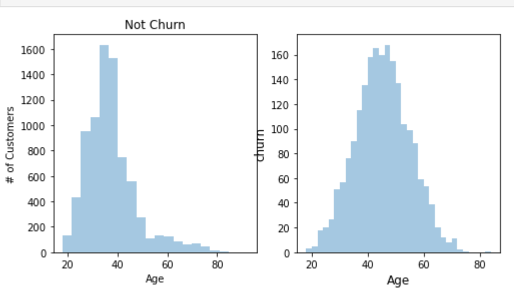
```
skewness: 1.0113202630234552
```



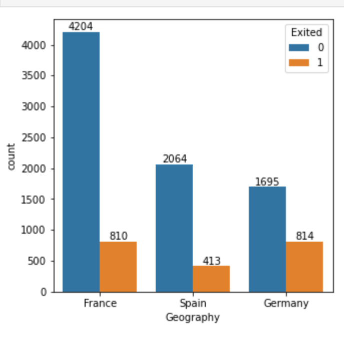
In [20]:
```
# churn by group of age

df['Age_Group'] = pd.cut(df['Age'],
bins=[0,30,40,50,60,np.inf],
labels=['<30','30-40','40-50','50-60','60+'])
plt.figure(figsize=(5,4))
ax = sns.countplot(x = 'Age_Group',hue='Exited',data=df)
plt.title('Churn by Age Goup')
for label in ax.containers:
    ax.bar_label(label);
```



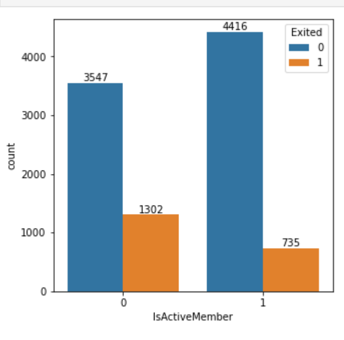
In [21]:
```
plt.figure(figsize=(8,4))
plt.subplot(1,2,1)
ax=sns.distplot(df[df['Exited']==0]['Age'],hist=True,kde=False,bins=20)
ax.set_ylabel('# of Customers')
ax.set_xlabel('Age')
ax.set_title('Not Churn')
plt.subplot(1,2,2)
ax = sns.distplot(df[df['Exited']==1]['Age'],hist=True,kde=False)
ax.set_xlabel('Age',size = 12)
ax.set_ylabel('churn',size =12);
```



In [22]:
```
#geographical column

plt.figure(figsize=(5,5))
ax=sns.countplot(x=df['Geography'],hue='Exited',data=df)
for label in ax.containers:
    ax.bar_label(label);
```



In [23]:
```
plt.figure(figsize=(5,5))
ax=sns.countplot(x=df['IsActiveMember'],hue='Exited',data=df)
for label in ax.containers:
    ax.bar_label(label);
```



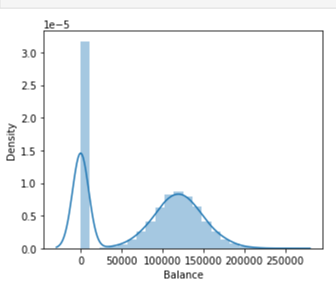
In [24]:
```
df['Balance'].value_counts()
```

Out[24]:
```
0.00         3617
130170.82       2
105473.74       2
85304.27        1
159397.75       1
             ...
81556.89        1
112687.69       1
108698.96       1
238387.56       1
130142.79       1
Name: Balance, Length: 6382, dtype: int64
```

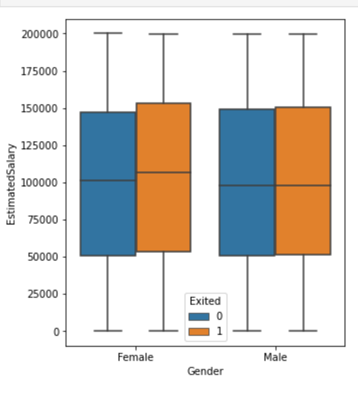
In [25]:
```
#due to more no.of zero balance customers
df['Balance'].where(df['Balance']==0,1).value_counts()
```

Out[25]:
```
1.0    6383
0.0    3617
Name: Balance, dtype: int64
```

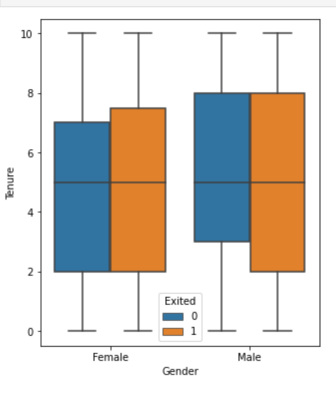
In [26]:
```
plt.figure(figsize=(5,4))
sns.distplot(df['Balance']);
```



In [27]:
```
plt.figure(figsize=(5,6))
sns.boxplot(y='EstimatedSalary',x ='Gender',hue ='Exited',data=df);
```



In [28]:
```
plt.figure(figsize=(5,6))
sns.boxplot(y='Tenure',x='Gender',hue='Exited',data=df);
```



In [29]:
```
df['Geography'].unique()
```

Out[29]:
```
array(['France', 'Spain', 'Germany'], dtype=object)
```

In [31]:
```
df.replace({'France':0,'Spain':1,'Germany':2},inplace=True)
df['Gender'].unique()
```

Out[31]:
```
array(['Female', 'Male'], dtype=object)
```

In [32]:
```
df.replace({'Female':0,'Male':1},inplace=True)
```

In [33]:
```
df.head()
```

Out[33]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Age_Group
0	1	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88	1	40-50
1	2	15647311	Hill	608	1	0	41	1	83807.86	1	0	1	112542.58	0	40-50
2	3	15619304	Onio	502	0	0	42	8	159660.80	3	1	0	113931.57	1	40-50
3	4	15701354	Boni	699	0	0	39	1	0.00	2	0	0	93826.63	0	30-40
4	5	15737888	Mitchell	850	1	0	43	2	125510.82	1	1	1	79084.10	0	40-50

In [34]:
```
df.columns
```

Out[34]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited', 'Age_Group'],
      dtype='object')
```

In [37]:
```
x=df[['CreditScore','Geography','Gender','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary']]
y=df['Exited']
```

In [ ]:
```
#splitting the dataset into train testset
```

In [38]:
```
from sklearn.model_selection import train_test_split
```

In [39]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

In [44]:
```
len(x_train)
```

Out[44]:
```
8000
```

In [45]:
```
len(x_test)
```

Out[45]:
```
2000
```

In [ ]:
```
#feature Scaling
```

In [48]:
```
from sklearn.preprocessing import StandardScaler
```

In [56]:
```
scaler = StandardScaler()
```

In [57]:
```
x_train= scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

In [58]:
```
x_train
```

Out[58]:
```
array([[ 0.35649971, -0.90598864,  0.91324755, ...,  0.64920267,
         0.97481699,  1.36766974],
       [-0.20389777,  1.50315516,  0.91324755, ...,  0.64920267,
         0.97481699,  1.6612541 ],
       [-0.96147213,  0.29858326,  0.91324755, ...,  0.64920267,
        -1.02583358, -0.25280688],
       ...,
       [ 0.86500853, -0.90598864, -1.09499335, ..., -1.54035103,
        -1.02583358, -0.1427649 ],
       [ 0.15932282, -0.90598864,  0.91324755, ...,  0.64920267,
        -1.02583358, -0.05082558],
       [ 0.47065475,  1.50315516,  0.91324755, ...,  0.64920267,
         0.97481699, -0.81456811]])
```

In [ ]:
```
#random forest classifier
```

In [61]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [62]:
```
model = RandomForestClassifier()
model
```

Out[62]:
```
RandomForestClassifier()
```

In [63]:
```
model_train=model.fit(x_train,y_train)
model_train
```

Out[63]:
```
RandomForestClassifier()
```

In [64]:
```
model_test=model.predict(x_test)
model_test
```

Out[64]:
```
array([0, 0, 0, ..., 1, 0, 0], dtype=int64)
```

In [ ]:
```
#model Accuracy
```

In [65]:
```
from sklearn.metrics import accuracy_score
```

In [66]:
```
score = accuracy_score(y_test,model_test)
score
```

Out[66]:
```
0.8685
```

In [ ]:

In [ ]: