ASSIGNMENTS - 2

Assignment Date	25 September2022
Student Name	Ajay E
Team ID	PNT2022TMID25121
Maximum Marks	2 Marks

In [1]:

```
#1Load the dataset
from google.colab import drive
```

In [2]:

```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [3]:

```
from pandas.io.formats.style import plt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
```

In [4]:

```
df=pd.read_csv('/content/drive/MyDrive/IBM/dataset')
```

In [5]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, cal
l drive.mount("/content/drive", force_remount=True).
```

In [6]:

```
df.head()
```

Out[6]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balanc
0	1	15634602	Hargrave	619	France	Female	42	2	0.0
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8
3	4	15701354	Boni	699	France	Female	39	1	0.0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8

In [

7]:

```
#6 check for categorical column and perform encoding
dummy=pd.get_dummies(df['Gender'])
dummy.head()
```

Out[7]:

	Female	Male
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0

In [8]:

```
df2=pd.concat((df,dummy),axis=1)
df2.head()
```

Out[8]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balanc
0	1	15634602	Hargrave	619	France	Female	42	2	0.(
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8
3	4	15701354	Boni	699	France	Female	39	1	0.(
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8

In [9]:

```
df2.drop(['Gender'],axis=1)
```

Out[9]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Age	Tenure	Balance	N
0	1	15634602	Hargrave	619	France	42	2	0.00	
1	2	15647311	Hill	608	Spain	41	1	83807.86	
2	3	15619304	Onio	502	France	42	8	159660.80	
3	4	15701354	Boni	699	France	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	43	2	125510.82	
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	39	5	0.00	
9996	9997	15569892	Johnstone	516	France	35	10	57369.61	
9997	9998	15584532	Liu	709	France	36	7	0.00	
9998	9999	15682355	Sabbatini	772	Germany	42	3	75075.31	
9999	10000	15628319	Walker	792	France	28	4	130142.79	

10000 rows × 15 columns

In [

In [10]:

```
df2=df2.drop(['Gender'],axis=1)
df2.head()
```

Out[10]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Age	Tenure	Balance	NumC
0	1	15634602	Hargrave	619	France	42	2	0.00	
1	2	15647311	Hill	608	Spain	41	1	83807.86	
2	3	15619304	Onio	502	France	42	8	159660.80	
3	4	15701354	Boni	699	France	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	43	2	125510.82	

In [1]:

1

```
df2=df2.drop(['Male'],axis=1)
df2.head()
```

Out[11]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Age	Tenure	Balance	NumC
0	1	15634602	Hargrave	619	France	42	2	0.00	
1	2	15647311	Hill	608	Spain	41	1	83807.86	
2	3	15619304	Onio	502	France	42	8	159660.80	
3	4	15701354	Boni	699	France	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	43	2	125510.82	

In [1]:

In [12]:

```
df2.rename(columns={"Female":"Gender"})
```

Out[12]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Age	Tenure	Balance	N
0	1	15634602	Hargrave	619	France	42	2	0.00	
1	2	15647311	Hill	608	Spain	41	1	83807.86	
2	3	15619304	Onio	502	France	42	8	159660.80	
3	4	15701354	Boni	699	France	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	43	2	125510.82	
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	39	5	0.00	
9996	9997	15569892	Johnstone	516	France	35	10	57369.61	
9997	9998	15584532	Liu	709	France	36	7	0.00	
9998	9999	15682355	Sabbatini	772	Germany	42	3	75075.31	
9999	10000	15628319	Walker	792	France	28	4	130142.79	

10000 rows × 14 columns

In [13]:

```
df.shape
```

Out[13]:

```
(10000, 14)
```

In [1]:

4

```
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 [1]:

In [15]:

```
#4 Handling Missing Values
df.isnull().any()
```

Out[15]:

```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

In [1]:

6

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

Out[16]:

```
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 [17]:

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

Out[17]:

```
0
```

In [18]:

```
#7 split the data into independent and dependent variable
y=df2['Tenure']
y
```

Out[18]:

```
0        2
1        1
2        8
3        1
4        2
       ..
9995     5
9996    10
9997     7
9998     3
9999     4
Name: Tenure, Length: 10000, dtype: int64
```

In [1]:

9

```
X=df.drop(columns=['Balance'],axis=1)
X.head()
```

Out[19]:

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

In [1]:

In [20]:

```
X=df.iloc[:,7:10]
X
```

Out[20]:

	Tenure	Balance	NumOfProducts
0	2	0.00	1
1	1	83807.86	1
2	8	159660.80	3
3	1	0.00	2
4	2	125510.82	1
...	...	...	...
9995	5	0.00	2
9996	10	57369.61	1
9997	7	0.00	1
9998	3	75075.31	2
9999	4	130142.79	1

10000 rows × 3 columns

In [21]:

```
print(X.shape)
```

```
(10000, 3)
```

In [ ]:

```
```

22

```
Y=df.iloc[:,7]
Y
```

Out[22]:

```
0        2
1        1
2        8
3        1
4        2
       ..
9995     5
9996    10
9997     7
9998     3
9999     4
Name: Tenure, Length: 10000, dtype: int64
```

In [23]:

```
print(y.shape)
```

```
(10000,)
```

In [24]:

```
#8 Scale the independent variable
from sklearn.preprocessing import scale
x_scaled=pd.DataFrame(scale(X),columns=X.columns)
x_scaled.head()
```

Out[24]:

	Tenure	Balance	NumOfProducts
0	-1.041760	-1.225848	-0.911583
1	-1.387538	0.117350	-0.911583
2	1.032908	1.333053	2.527057
3	-1.387538	-1.225848	0.807737
4	-1.041760	0.785728	-0.911583

In [25]:

```
#9 Train test split
from sklearn.model_selection import train_test_split
```

In [26]:

```
# Split data (train & test data)
X_train, X_test, y_train, y_test = train_test_split(X,y)
```

In [ ]:

27

```
#display shape
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(7500, 3)
(2500, 3)
(7500,)
(2500,)
```

In [28]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30,random_state=0)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(7000, 3)
(3000, 3)
(7000,)
(3000,)
```

In [29]:

```
print(np.mean(X_train))
print(np.mean(X_test))
```

```
Tenure              4.989429
Balance          75204.853421
NumOfProducts       1.534286
dtype: float64
Tenure              5.067333
Balance          79474.972977
NumOfProducts       1.520667
dtype: float64
```

In [ ]:

```
```

30

```
#3 Descriptive stastical analysis
df.describe()
```

Out[30]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	N
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	

In [ ]:

In [31]:

```
df.NumOfProducts.value_counts()
```

Out[31]:

```
1    5084
2    4590
3     266
4      60
Name: NumOfProducts, dtype: int64
```

In [ ]:

```
```

32

```
df.corr()
```

Out[32]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOf
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495	-0.009067	
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883	-0.012419	
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842	0.006268	
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997	0.028308	-
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000	-0.012254	
Balance	-0.009067	-0.012419	0.006268	0.028308	-0.012254	1.000000	-
NumOfProducts	0.007246	0.016972	0.012238	-0.030680	0.013444	-0.304180	
HasCrCard	0.000599	-0.014025	-0.005458	-0.011721	0.022583	-0.014858	
IsActiveMember	0.012044	0.001665	0.025651	0.085472	-0.028362	-0.010084	
EstimatedSalary	-0.005988	0.015271	-0.001384	-0.007201	0.007784	0.012797	
Exited	-0.016571	-0.006248	-0.027094	0.285323	-0.014001	0.118533	-

In [ ]:

In [33]:

```
df['Age'].mean()
```

Out[33]:

```
38.9218
```

In [34]:

```
df['Gender'].value_counts()
```

Out[34]:

```
Male      5457
Female    4543
Name: Gender, dtype: int64
```

In [ ]:

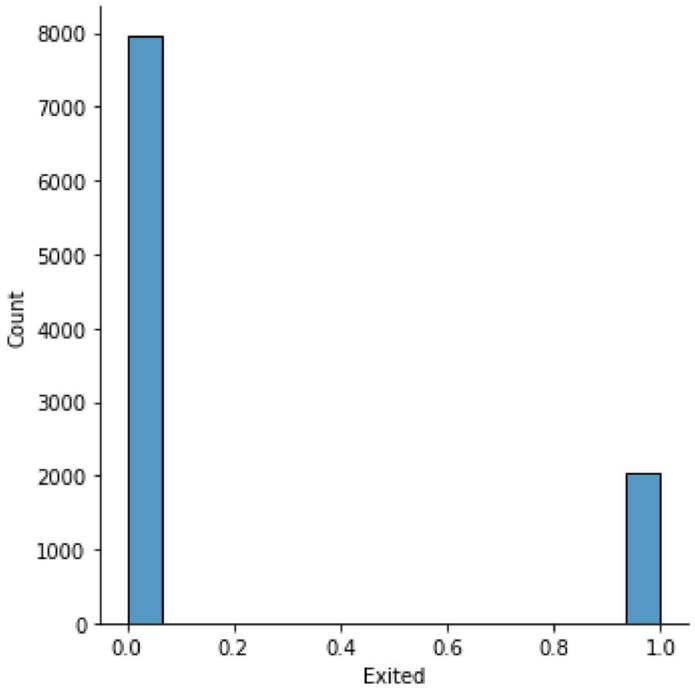
```
#2 Visualization
```

61

```
#univariate analysis
sns.displot(df.Exited)
```

Out[61]:

```
<seaborn.axisgrid.FacetGrid at 0x7f8a972818d0>
```



In [ ]:

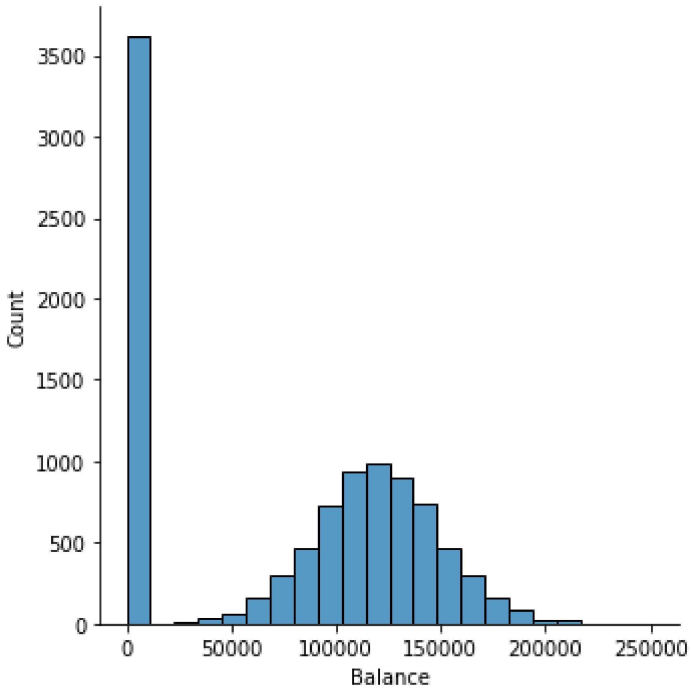
```
```

60

```
sns.displot(df.Balance)
```

Out[60]:

```
<seaborn.axisgrid.FacetGrid at 0x7f8a97db4a50>
```

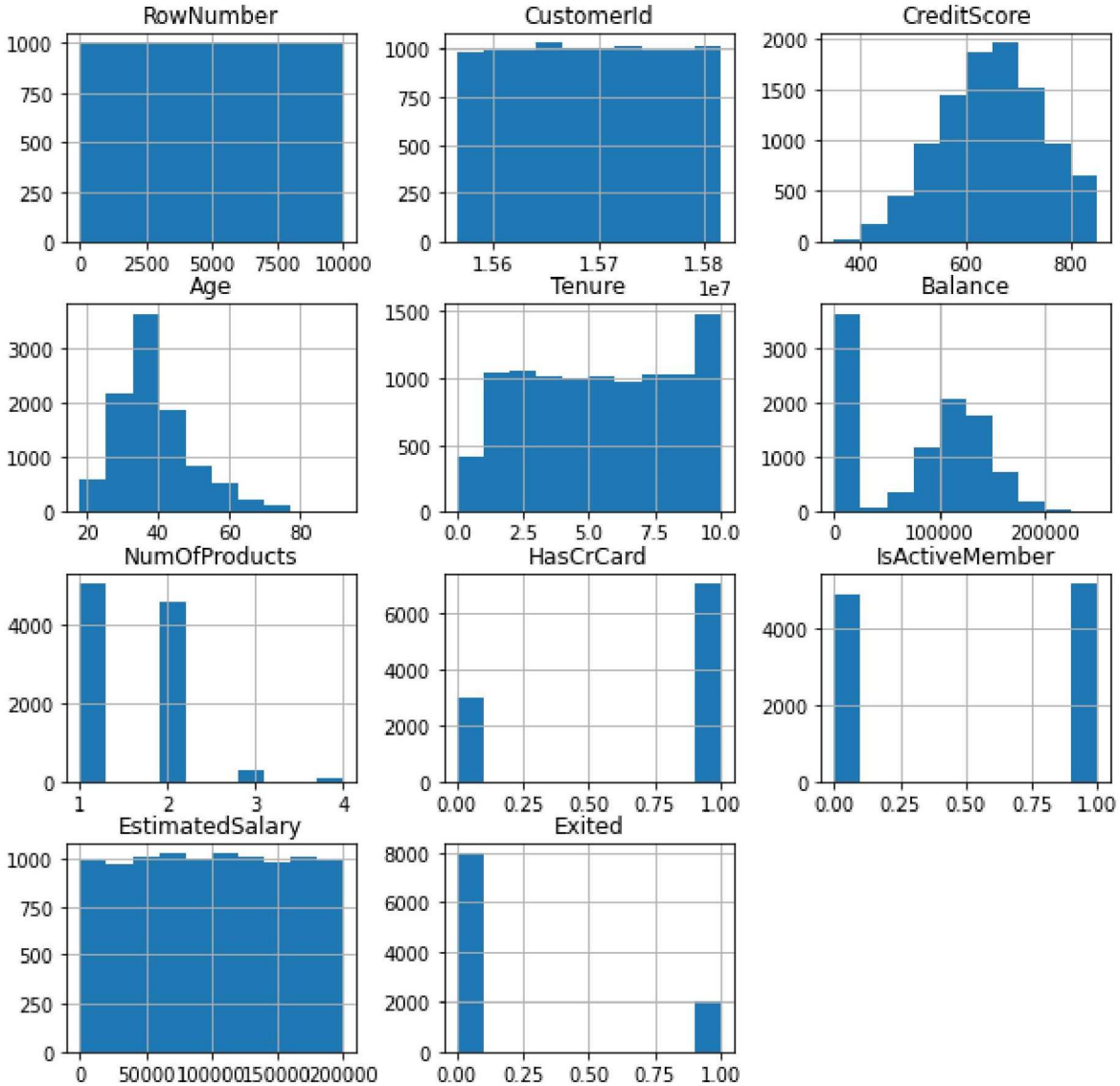


37

```
df.hist(figsize=(10,10))
```

Out[37]:

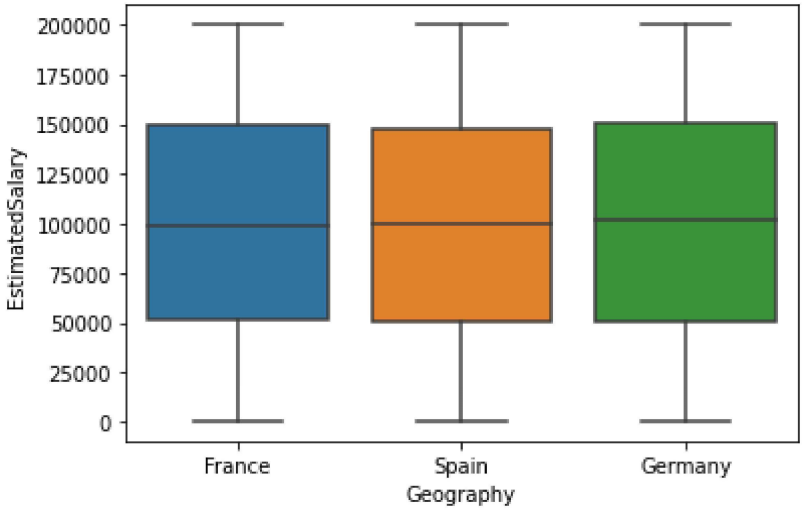
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9ca3a150>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9c9ee790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9c9a4d90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9c96a3d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9c924cd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9c8e9550>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9c8a1d10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9c869290>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9c8692d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9c81c550>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9c7fdb50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8a9c74b690>]],
      dtype=object)
```



In [ ]:

In [62]:

```
#Boxplot
sns.boxplot(x='Geography',y='EstimatedSalary',data=df)
plt.show()
```



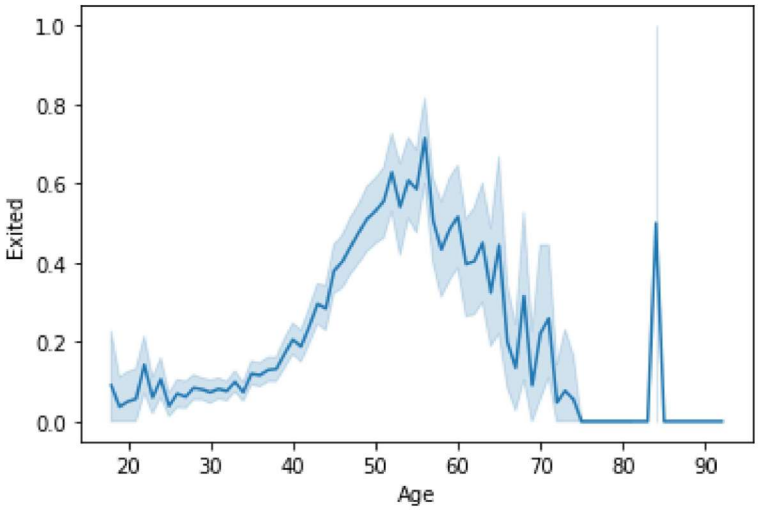
39

```
# Bivariate Analysis on continuous variable
sns.lineplot(df.Age,df.Exited)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a9c438650>
```



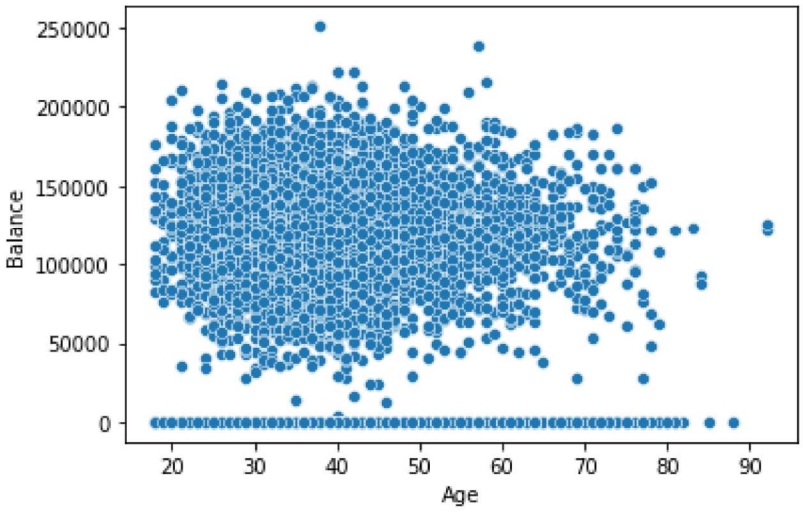
In [40]:

```
sns.scatterplot(df.Age,df.Balance)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a9c308b10>
```



In [4]:

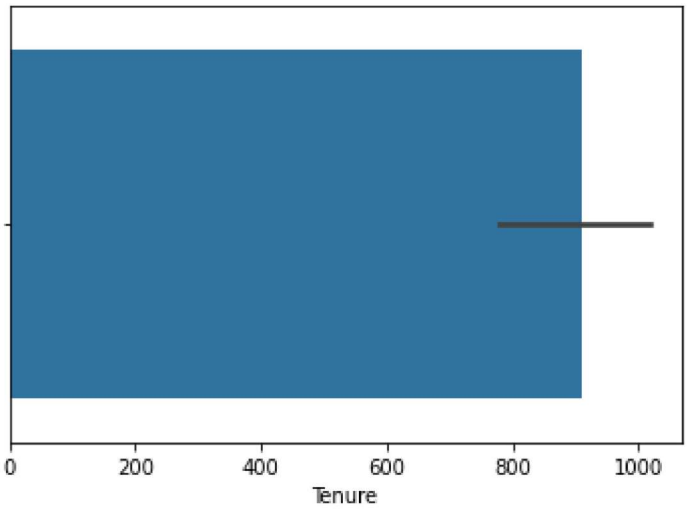
1

```
# Bivariate Analysis on categorical variable
sns.barplot(df.Tenure.value_counts())
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a9c313150>
```



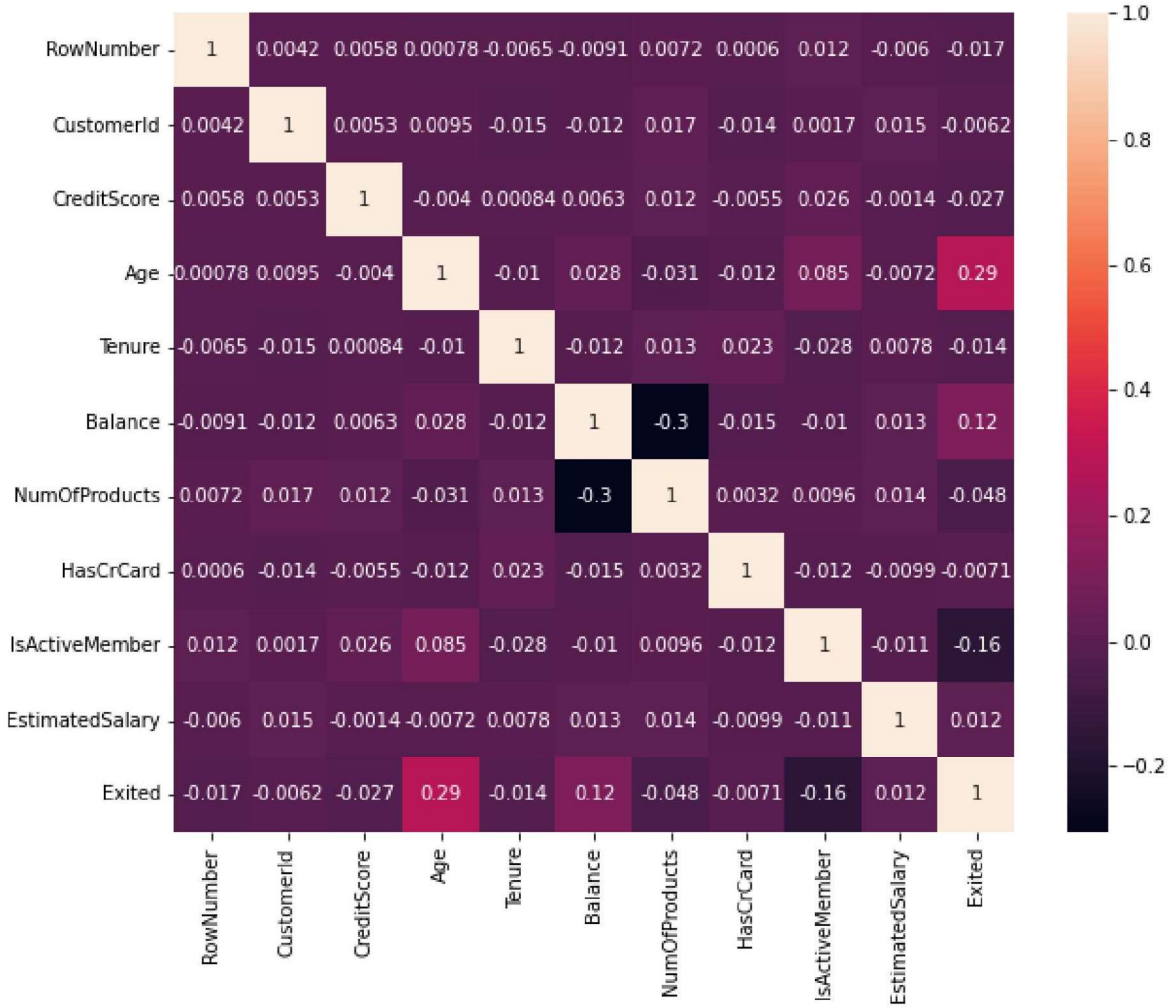
In [4]:

2

```
fig=plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),annot=True)
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a9c313050>
```



In [4]:

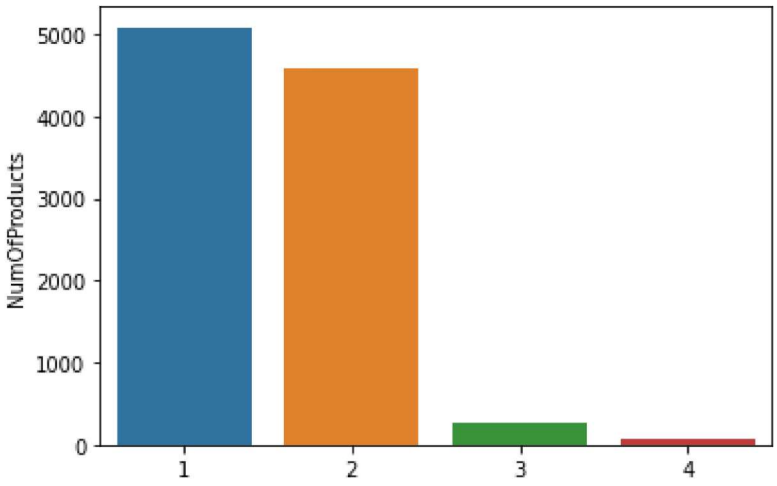
3

```
sns.barplot(df.NumOfProducts.value_counts().index,df.NumOfProducts.value_counts())
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[43]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a9c04b6d0>
```



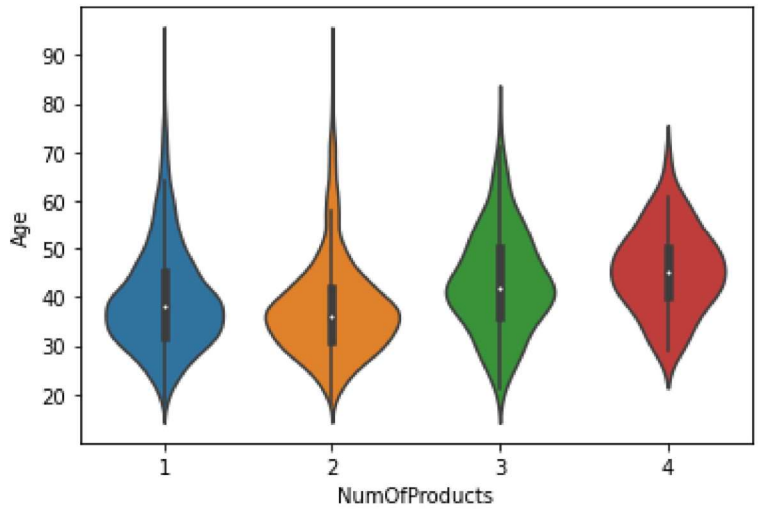
In [4]:

In [65]:

```
sns.violinplot(x='NumOfProducts',y='Age',data=df,size=8)
plt.show
```

Out[65]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [4]:

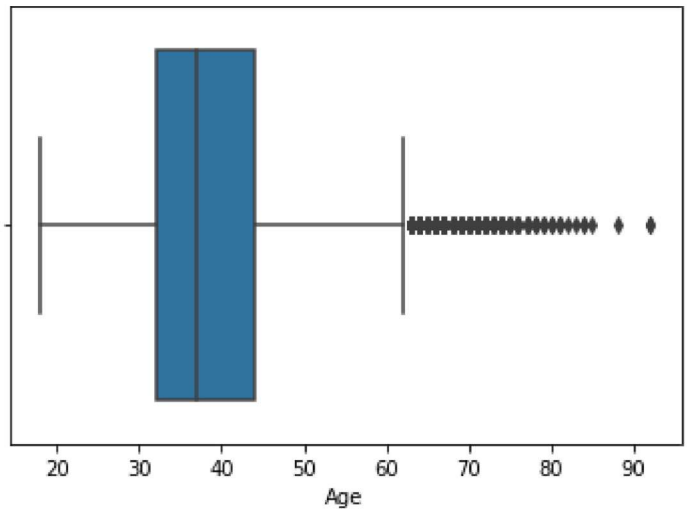
5

```
sns.boxplot(df.Age)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[45]:

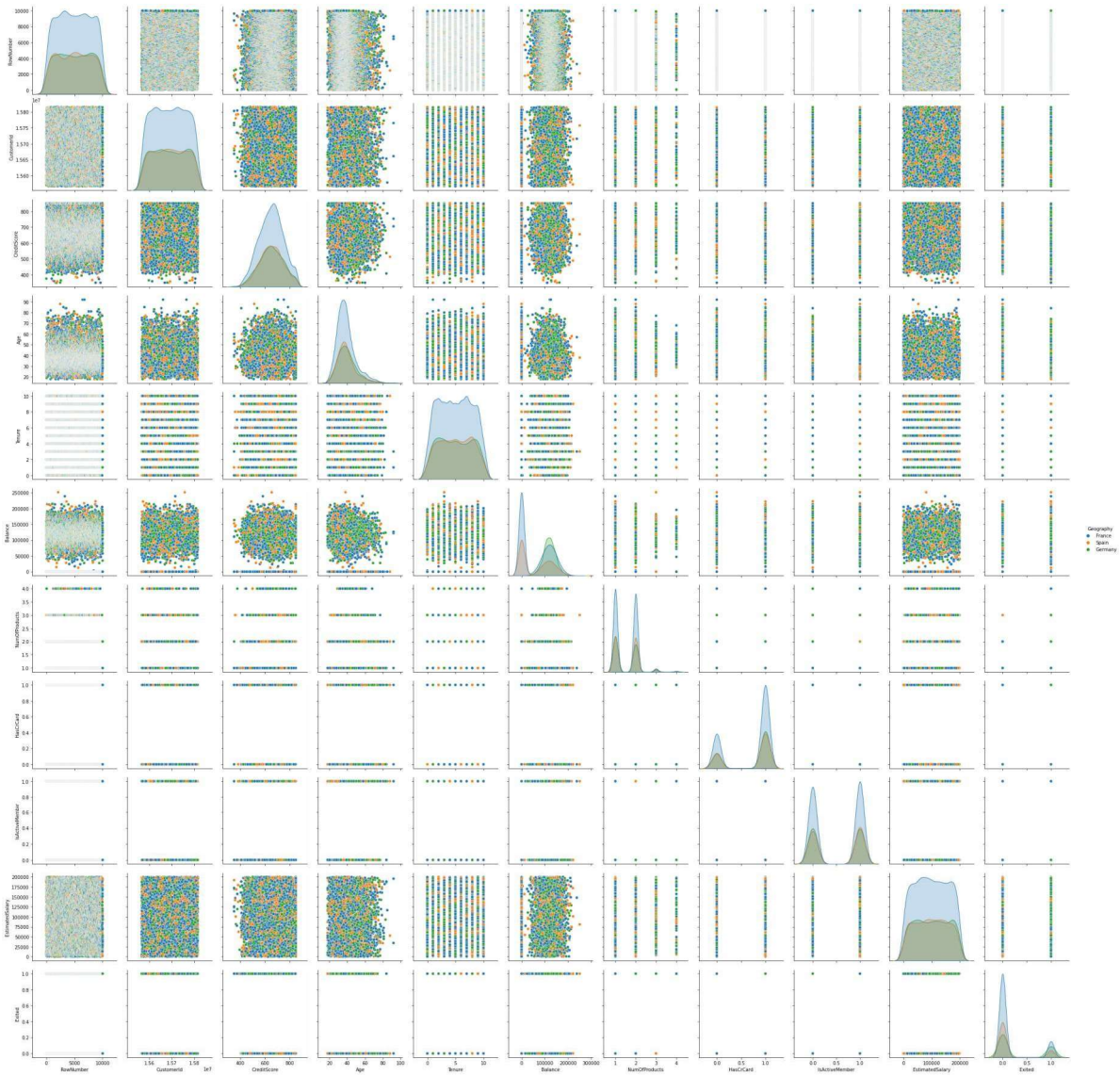
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a9a8329d0>
```



In [ ]:

66

```
#Multivariate analysis
sns.pairplot(data=df,hue='Geography',height=3)
plt.show()
```



In [47]:

```
#5 Find the outlier and replace the outlier
#find the limits
upper_limit=df['Age'].mean()+3*df['Age'].std()
lower_limit=df['Age'].mean()-3*df['Age'].std()
print('upper_limit',upper_limit)
print('lower_limit',lower_limit)
```

```
upper_limit 70.38521935511383
lower_limit 7.458380644886169
```

In [48]:

```
df.loc[(df['Age']>upper_limit)|(df['Age']<lower_limit)]
```

Out[48]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	B
85	86	15805254	Ndukaku	652	Spain	Female	75	10	
158	159	15589975	Maclean	646	France	Female	73	6	97
230	231	15808473	Ringrose	673	France	Male	72	1	
252	253	15793726	Matveyeva	681	France	Female	79	0	
310	311	15712287	Pokrovskii	652	France	Female	80	4	
...	...	...	...	...	...	...	...	...	
9646	9647	15603111	Muir	850	Spain	Male	71	10	69
9671	9672	15636061	Pope	649	Germany	Male	78	4	68
9736	9737	15644103	Wells	659	Spain	Male	78	2	151
9894	9895	15704795	Vagin	521	France	Female	77	6	
9936	9937	15653037	Parks	609	France	Male	77	1	

133 rows × 14 columns

In [49]:

```
#trimming - delete the outlier
new_df=df.loc[(df['Age']<upper_limit)&(df['Age']>lower_limit)]
print('Before removing outlier:',len(df))
print('After removing outlier:',len(new_df))
```

```
Before removing outlier: 10000
After removing outlier: 9867
```

In [5]:

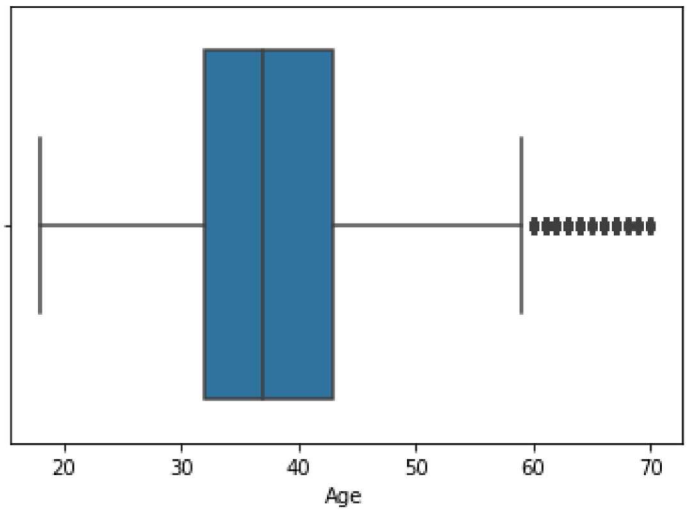
0

```
sns.boxplot(new_df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a97f01790>
```



In [5]:

In [51]:

```
#copying-change the outlier value to upper or lower limits values
new_df=df.copy()
new_df.loc[(new_df['Age']>upper_limit),'Age']=upper_limit
new_df.loc[(new_df['Age']<lower_limit),'Age']=lower_limit
```

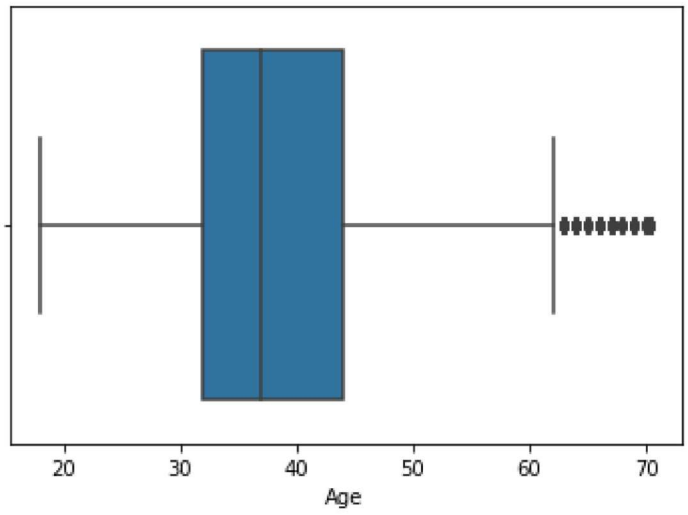
2

```
sns.boxplot(new_df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[52]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a98031ad0>
```



In [5]:

In [53]:

```
len(new_df)
```

Out[53]:

```
10000
```

In [54]:

```
#IQR method
q1=df['Age'].quantile(0.25)
q3=df['Age'].quantile(0.75)
iqr=q3-q1
```

In [55]:

```
q1,q3,iqr
```

Out[55]:

```
(32.0, 44.0, 12.0)
```

In [5]:

6

```
upper_limit=q3+(1.5*iqr)
lower_limit=q3-(1.5*iqr)
lower_limit,upper_limit
```

Out[56]:

```
(26.0, 62.0)
```

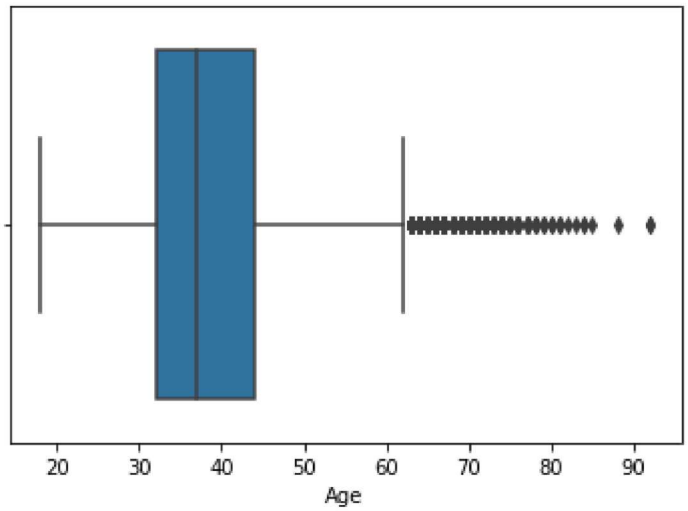
In [57]:

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a98077ed0>
```



In [58]:

```
#copying-change the outlier value to upper or lower limits values
new_df=df.copy()
new_df.loc[(new_df['Age']>upper_limit),'Age']=upper_limit
new_df.loc[(new_df['Age']<lower_limit),'Age']=lower_limit
```

In [5]:

9

```
sns.boxplot(new_df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[59]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a97e1a690>
```