```
#!/usr/bin/env python
# coding: utf-8

# In[88]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
from scipy import stats
from sklearn.preprocessing import OneHotEncoder


# In[11]:


dataset = pd.read_csv('C:\\Users\\Devi\\Downloads\\Churn_Modelling (1) (1).csv')


# In[12]:


dataset
```

In[13]:

dataset.head()

In[14]:

dataset.tail()

Univariate Analysis

In[6]:

df_1=dataset.loc[dataset['NumOfProducts']==1]

df_2=dataset.loc[dataset['NumOfProducts']==2]

df_3=dataset.loc[dataset['NumOfProducts']==3]

In[7]:

plt.plot(df_1['Age'],np.zeros_like(df_1['Age']))

plt.plot(df_2['Age'],np.zeros_like(df_2['Age']))

```
plt.plot(df_3['Age'],np.zeros_like(df_3['Age']))

plt.xlabel('Age')

plt.show()



# # Bivariate Analysis


# In[8]:



sns.FacetGrid(dataset,hue="NumOfProducts",size=5).map(plt.scatter,"Age","Geography").add_legend();



# # Multivariate Analysis


# In[9]:



sns.pairplot(dataset,hue="NumOfProducts",size=5)



# # Descriptive Statistics


# In[16]:



dataset.sum()
```

In[17]:

```
dataset.sum(axis=1)
```

In[18]:

```
dataset.median()
```

In[19]:

```
dataset.mean()
```

In[20]:

```
dataset.max()
```

In[21]:

```
dataset.std()
```

```
# In[22]:
```

```
dataset.var()
```

```
# In[24]:
```

```
Age=dataset.Age
Age.value_counts()
```

```
# In[25]:
```

```
dataset.describe()
```

```
# # Handle Null Values
```

```
# In[27]:
```

```
dataset.shape
```

```
# In[28]:
```

```
dataset.isnull()
```

```
# In[31]:
```

```
dataset.isnull().sum()
```

```
# In[32]:
```

```
dataset.isnull().sum().sum()
```

```
# # Outlier
```

```
# In[58]:
```

```
sns.displot(dataset['Gender'])
```

```
# In[59]:
```

```
sns.boxplot(x='Gender',y='Age',data=dataset)
```

```
# In[60]:
```

```
sns.boxplot(y='Age',data=dataset)
```

```
# In[61]:
```

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

```
# In[67]:
```

```
data1=dataset[dataset['Age']<40]
```

In[68]:

```
sns.boxplot(y='Age',data=data1)
```

categorial Encoding

In[70]:

```
data_tips=pd.get_dummies(dataset)
data_tips
```

In[75]:

```
one_encde=OneHotEncoder(sparse=False)
encoded_arr=one_encde.fit_transform(dataset[['CustomerId','CreditScore','Age','Tenure']])
encoded_arr
```

split the data into dependent and independent

In[85]:

```
x=dataset.iloc[:,1:4]
y=dataset.iloc[:,4]
x
y
```

In[ ]:

In[ ]:

In[ ]: