Assignment 2

ASSIGNMENT 2 - DATA VISUALIZATION AND DATA PREPROCESSING

1. Dataset downloaded as “model.csv”

2.Load the dataset

[7]:
```
#importing libraries
import pandas as pd
#load the dataset
df=pd.read_csv("model.csv")
df
```

[7]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	\
0	1	15634602	Hargrave	619	France	Female	42	
1	2	15647311	Hill	608	Spain	Female	41	
2	3	15619304	Onio	502	France	Female	42	
3	4	15701354	Boni	699	France	Female	39	
4	5	15737888	Mitchell	850	Spain	Female	43	
…	…	…	…	…	…	…	…	
9995	9996	15606229	Obijiaku	771	France	Male	39	
9996	9997	15569892	Johnstone	516	France	Male	35	
9997	9998	15584532	Liu	709	France	Female	36	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	
9999	10000	15628319	Walker	792	France	Female	28	

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
0	2	0.00	1	1	1	
1	1	83807.86	1	0	1	
2	8	159660.80	3	1	0	
3	1	0.00	2	0	0	
4	2	125510.82	1	1	1	
…	…	…	…	…	…	
9995	5	0.00	2	1	0	
9996	10	57369.61	1	1	1	
9997	7	0.00	1	0	1	
9998	3	75075.31	2	1	0	

9999	4	130142.79		1	1	0

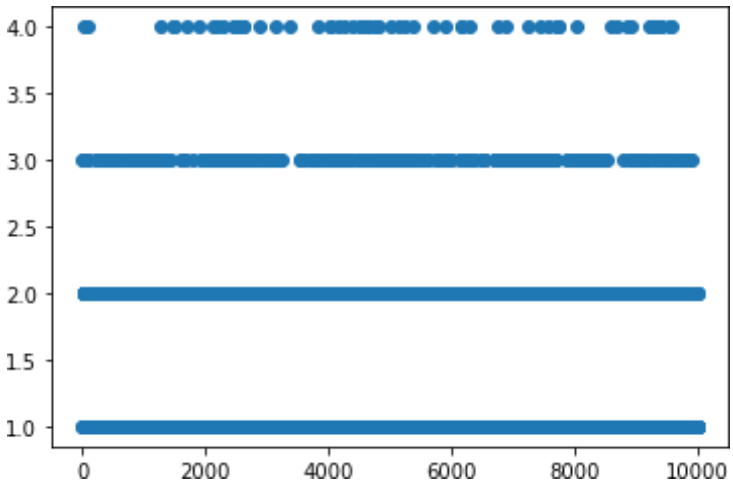
	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0
...	...	...
9995	96270.64	0
9996	101699.77	0
9997	42085.58	1
9998	92888.52	1
9999	38190.78	0

[10000 rows x 14 columns]

3. Perform Below Visualizations

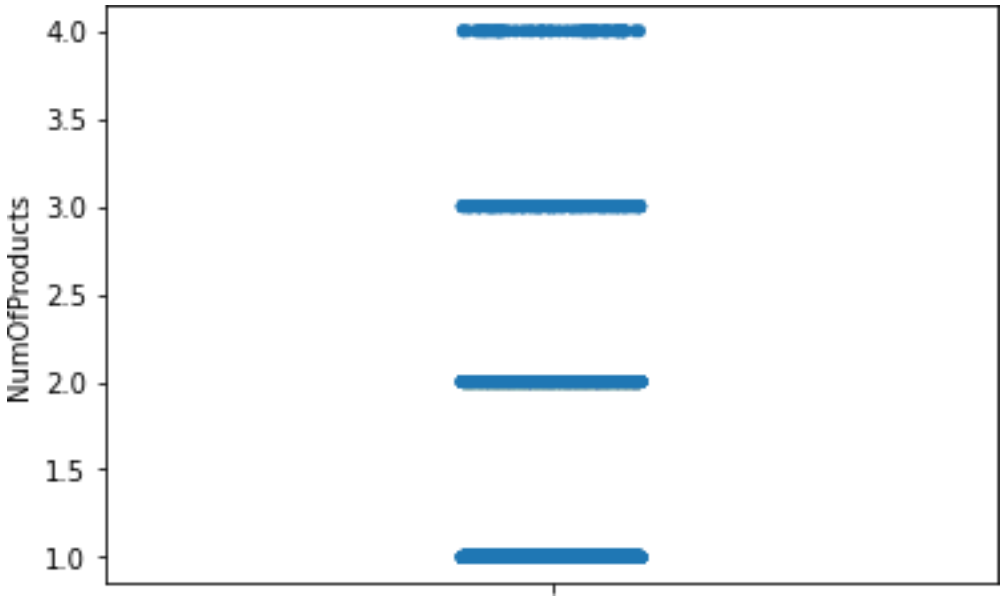
3.1 Univariate Analysis

[1]:
```
#scatterplot
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
#load the dataset
df=pd.read_csv("model.csv")
plt.scatter(df.index,df['NumOfProducts'])
plt.show()
```



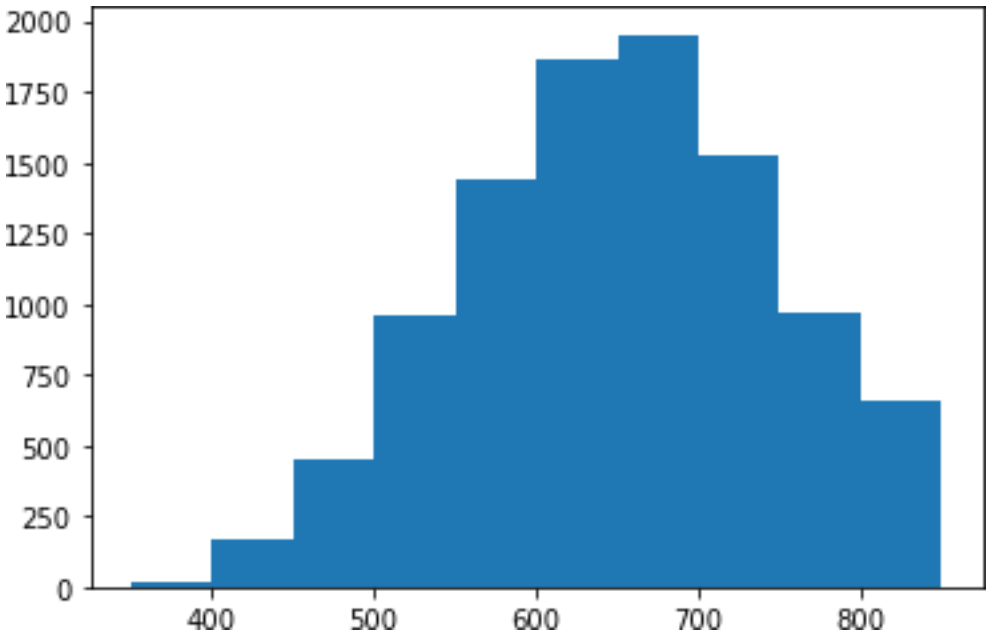
[7]:
```
#strip plot
sns.stripplot(y='NumOfProducts')
```

[7]: <AxesSubplot:ylabel='NumOfProducts'>



[8]:
```
#histogram
plt.hist(df['CreditScore'])
```

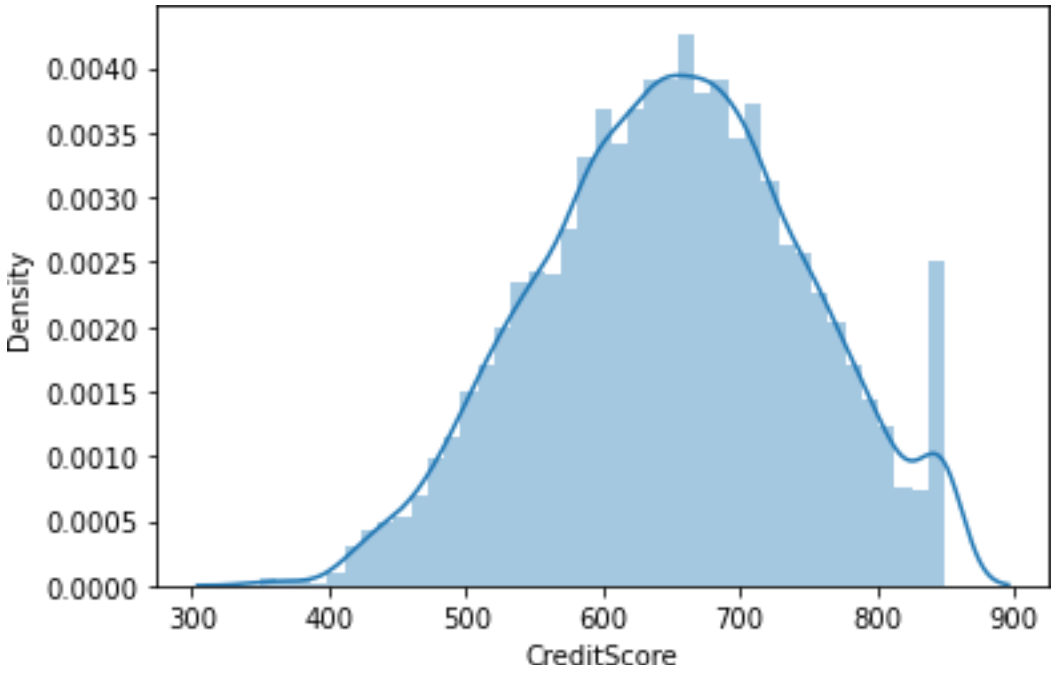
[8]: (array([19., 166., 447., 958., 1444., 1866., 1952., 1525., 968.,
655.]),
array([350., 400., 450., 500., 550., 600., 650., 700., 750., 800., 850.]),
<BarContainer object of 10 artists>)



[4]:
```
import seaborn as sns
import pandas as pd
df=pd.read_csv("model.csv")
sns.distplot(df['CreditScore'])
```

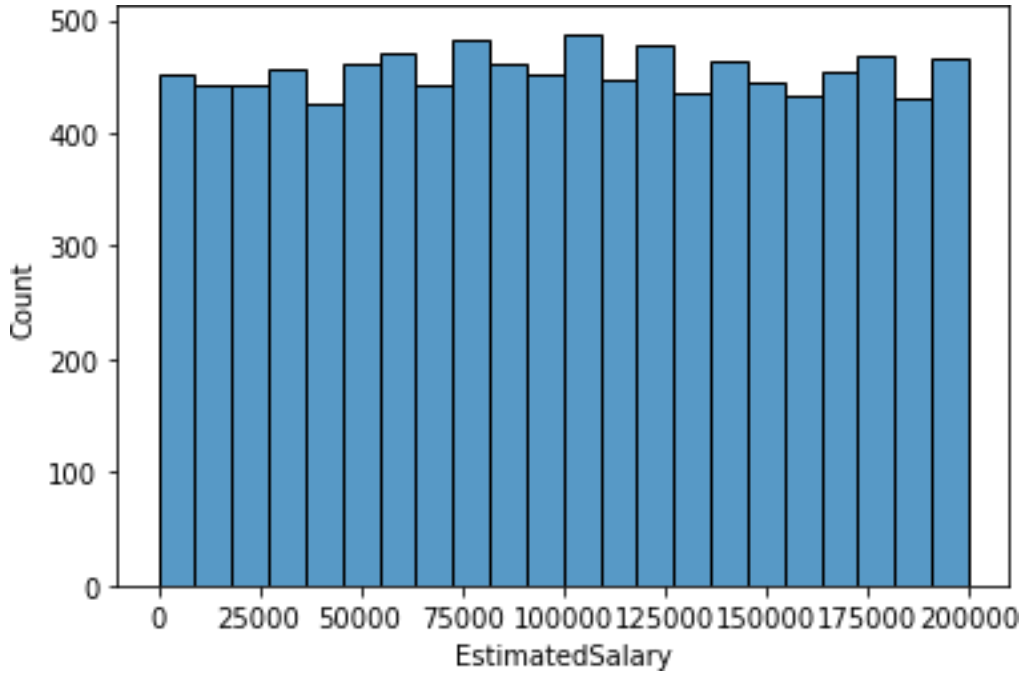
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-levelfunction with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

[4]: <AxesSubplot:xlabel='CreditScore', ylabel='Density'>



[6]:
```
sns.histplot(df['EstimatedSalary'])
```

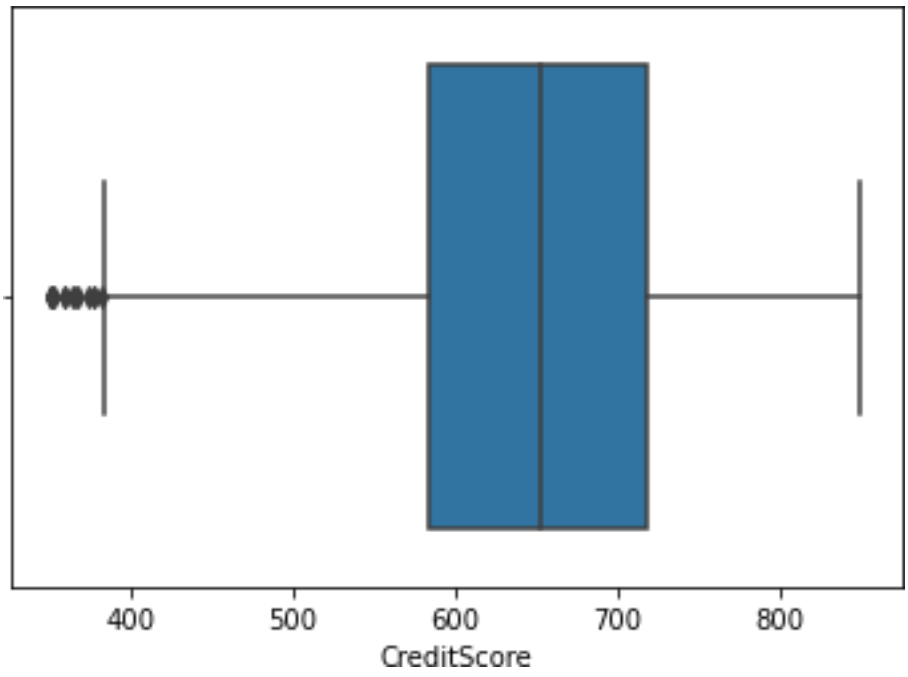
[6]: <AxesSubplot:xlabel='EstimatedSalary', ylabel='Count'>



[10]:
```
#boxplot
sns.boxplot(df['CreditScore'])
```

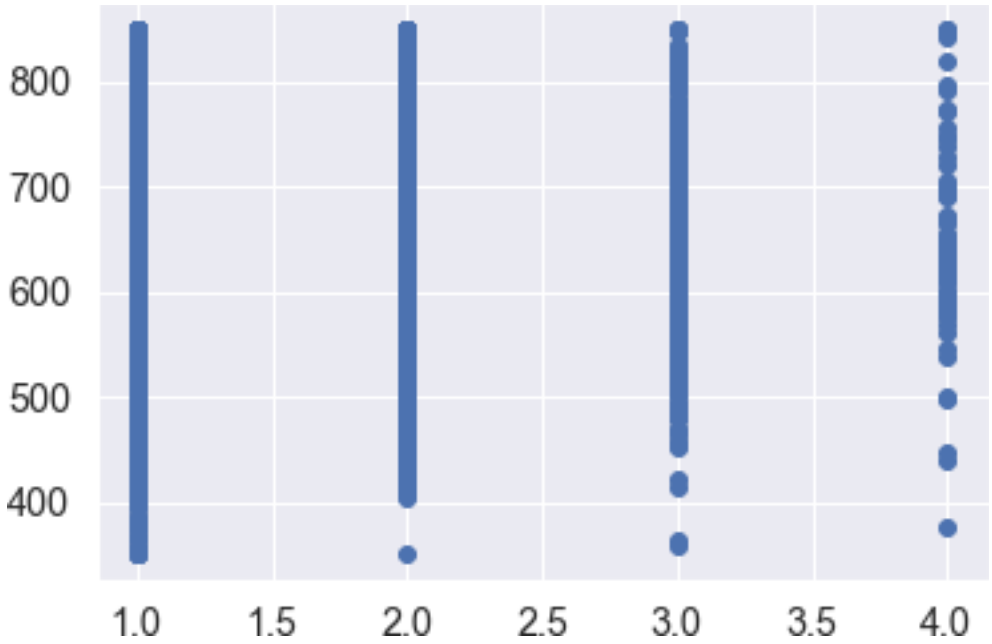
C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

[10]: <AxesSubplot:xlabel='CreditScore'>



3.2 Bivariate Analysis

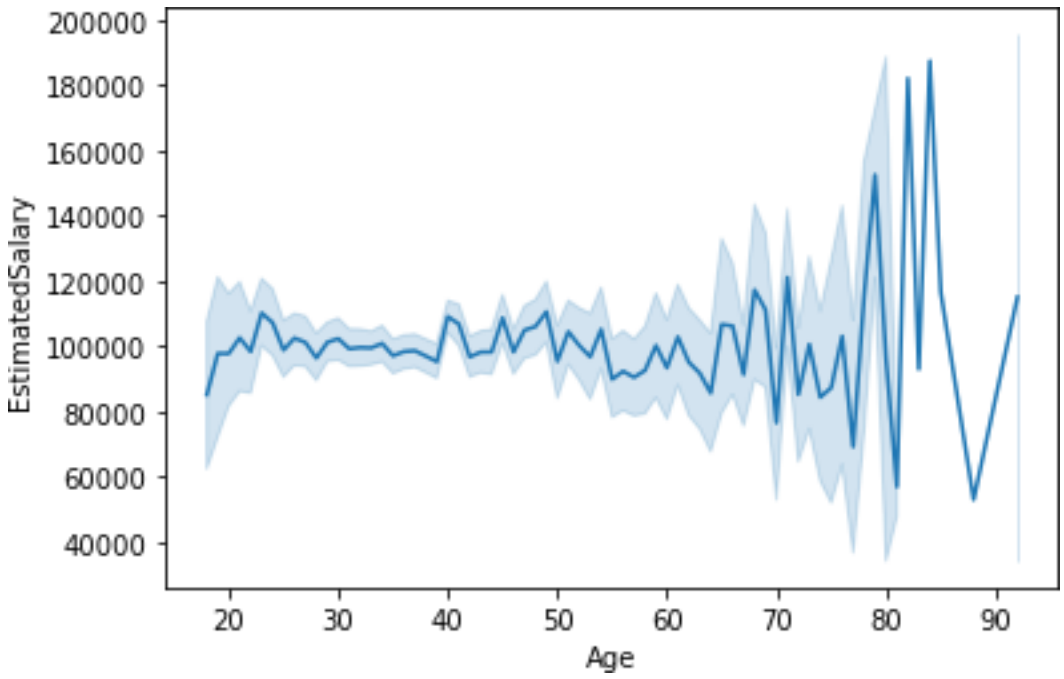
[21]:
```
#scatter plot
plt.scatter(df.NumOfProducts, df.CreditScore)
plt.show()
```



[8]:
```
sns.lineplot(df['Age'],df['EstimatedSalary'])
```

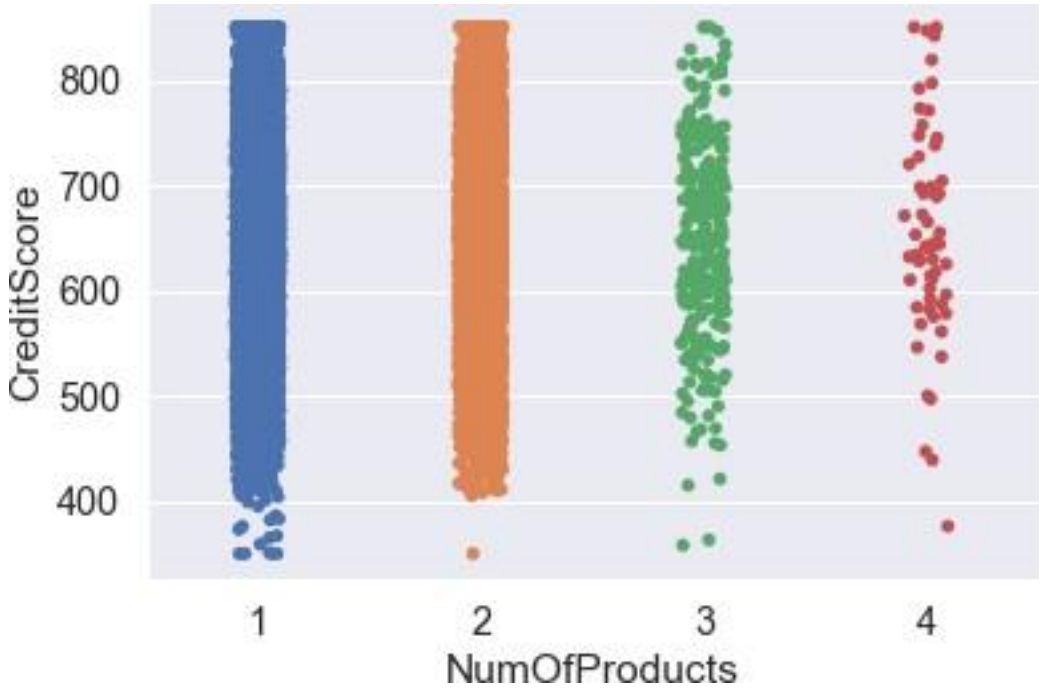
C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

[8]: <AxesSubplot:xlabel='Age', ylabel='EstimatedSalary'>



[22]:
```
#strip plot
sns.stripplot(x=df['NumOfProducts'],y=df['CreditScore'])
```

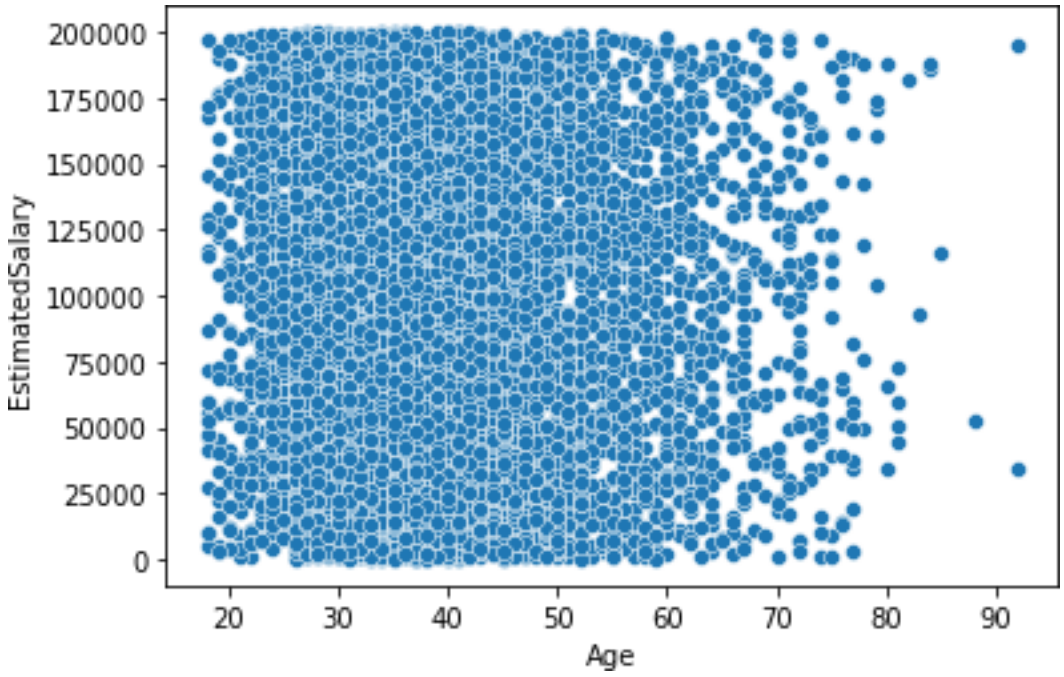
[22]: <AxesSubplot:xlabel='NumOfProducts', ylabel='CreditScore'>



[10]:
```
sns.scatterplot(df['Age'],df['EstimatedSalary'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

[10]: <AxesSubplot:xlabel='Age', ylabel='EstimatedSalary'>



3.3 Mulitivariate Analysis

[12]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style('darkgrid')
sns.set(font_scale=1.3)

df=pd.read_csv('model.csv')
```

```
df
```

[12]:

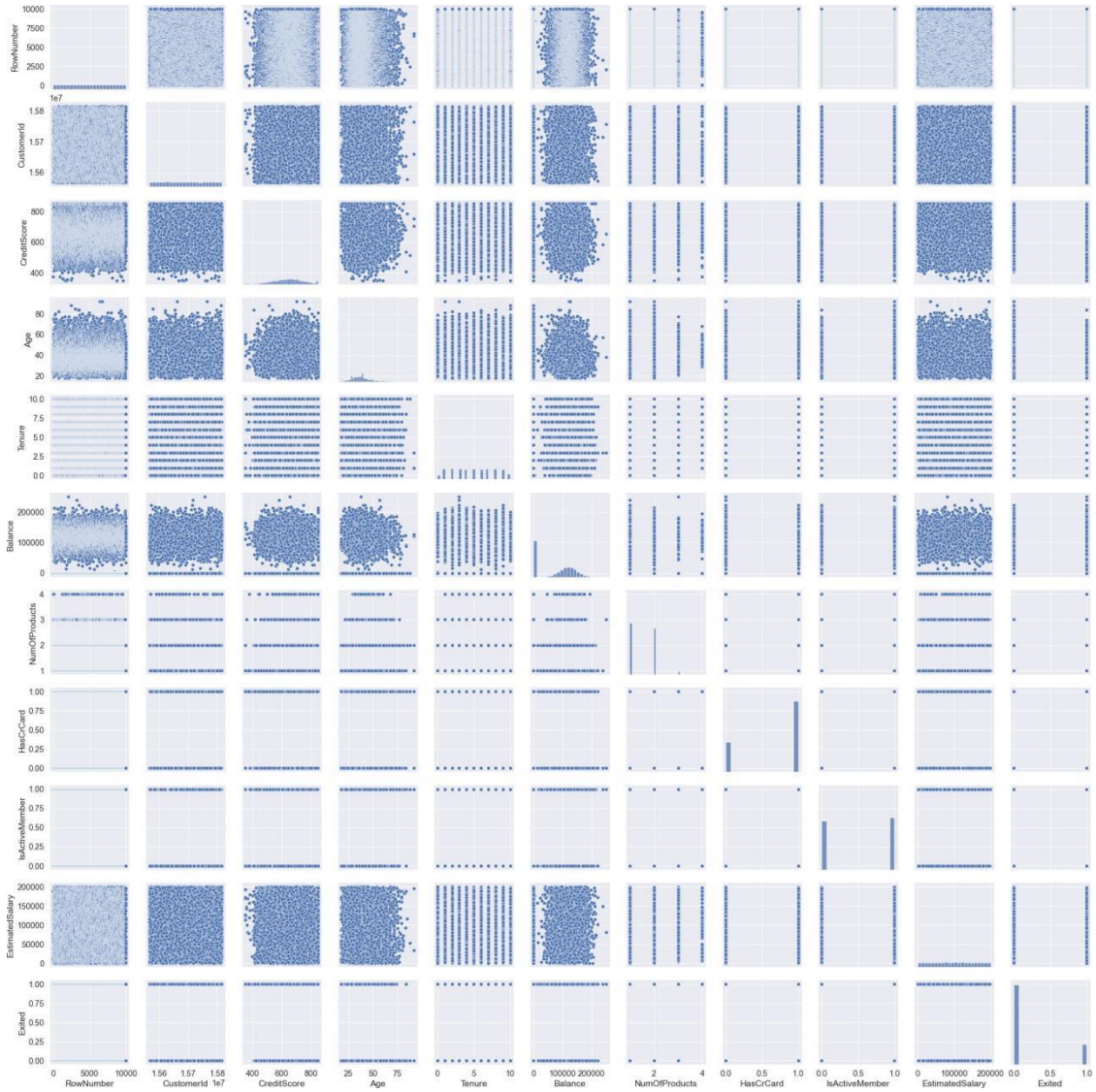
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	\
0	1	15634602	Hargrave	619	France	Female	42	
1	2	15647311	Hill	608	Spain	Female	41	
2	3	15619304	Onio	502	France	Female	42	
3	4	15701354	Boni	699	France	Female	39	
4	5	15737888	Mitchell	850	Spain	Female	43	
…	…	…	…	…	…	…	…	
9995	9996	15606229	Obijiaku	771	France	Male	39	
9996	9997	15569892	Johnstone	516	France	Male	35	
9997	9998	15584532	Liu	709	France	Female	36	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	
9999	10000	15628319	Walker	792	France	Female	28	

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

	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0
…	…	…
9995	96270.64	0
9996	101699.77	0
9997	42085.58	1
9998	92888.52	1
9999	38190.78	0

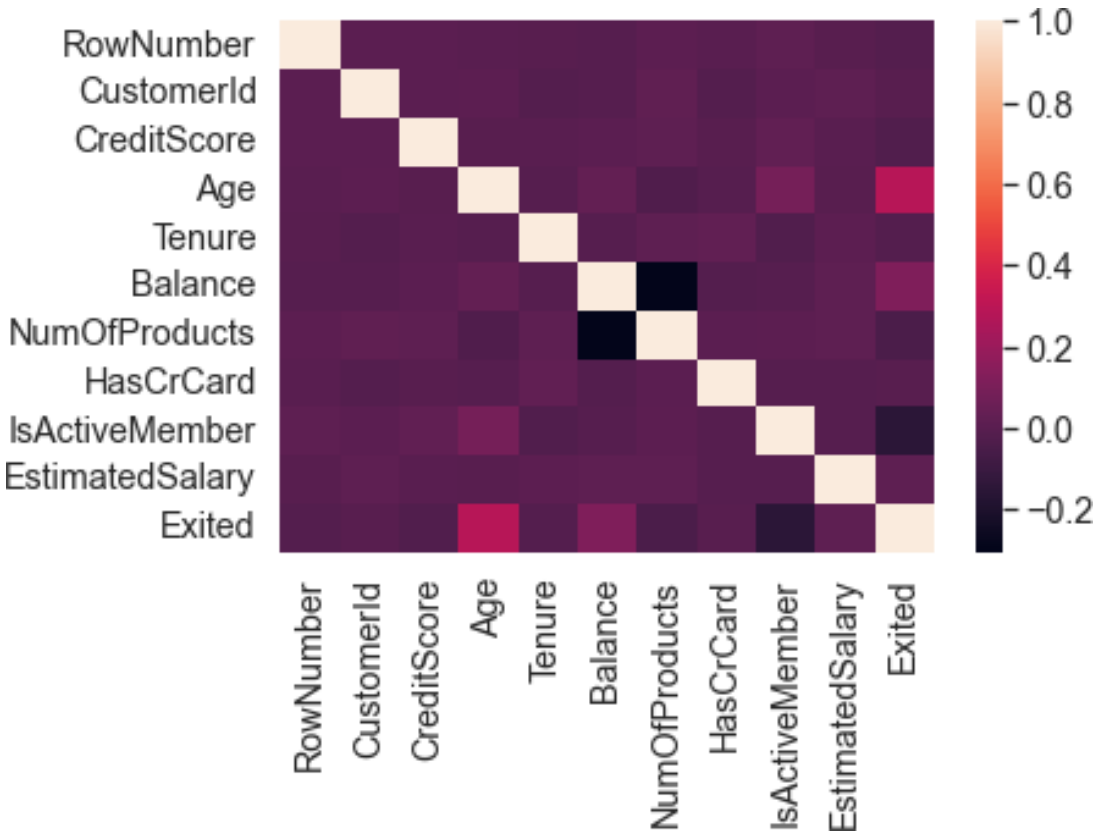
[10000 rows x 14 columns]

[13]:
```
#pairplot
sns.pairplot(df);
```



[14]:
```
sns.heatmap(df.corr())
```

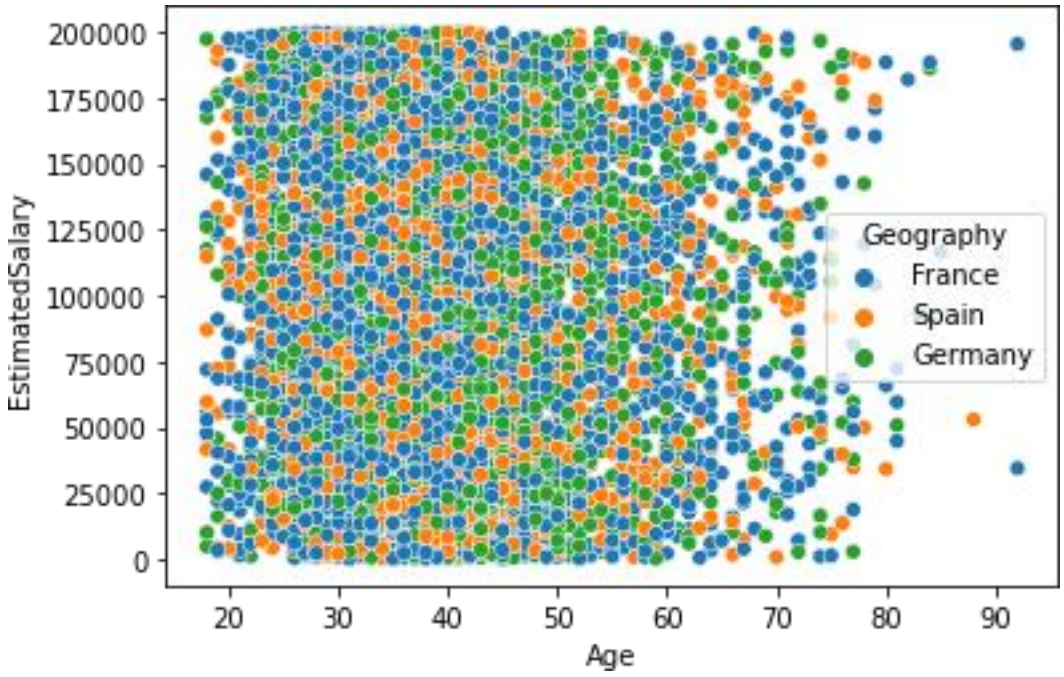
[14]: <AxesSubplot:>



[11]:
```
sns.scatterplot(df['Age'],df['EstimatedSalary'],df['Geography'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y, hue. Fromversion 0.12, the only valid positional argument will be `data`, and passingother arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

[11]: <AxesSubplot:xlabel='Age', ylabel='EstimatedSalary'>



4. Perform descriptive statistics on the dataset.

[4]:
```
#load the dataset
import pandas as pd
data=pd.read_csv("model.csv")
data.head()
```

[4]:

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

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
0	2	0.00	1	1	1	
1	1	83807.86	1	0	1	
2	8	159660.80	3	1	0	
3	1	0.00	2	0	0	
4	2	125510.82	1	1	1	

	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0

2	113931.57	1
3	93826.63	0
4	79084.10	0

[12]:
```
df.mean()
```

```
C:\Users\janar vijay\AppData\Local\Temp\ipykernel_9188\3698961737.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select
only valid columns before calling the reduction.
  df.mean()
```

[12]:
```
RowNumber          5.000500e+03
CustomerId         1.569094e+07
CreditScore        6.505288e+02
Age                3.892180e+01
Tenure             5.012800e+00
Balance            7.648589e+04
NumOfProducts      1.530200e+00
HasCrCard          7.055000e-01
IsActiveMember     5.151000e-01
EstimatedSalary    1.000902e+05
Exited             2.037000e-01
dtype: float64
```

[13]:
```
df.describe()
```

[13]:

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

	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
count	10000.000000	10000.000000	10000.00000	10000.000000	
mean	76485.889288	1.530200	0.70550	0.515100	
std	62397.405202	0.581654	0.45584	0.499797	
min	0.000000	1.000000	0.00000	0.000000	
25%	0.000000	1.000000	0.00000	0.000000	
50%	97198.540000	1.000000	1.00000	1.000000	
75%	127644.240000	2.000000	1.00000	1.000000	
max	250898.090000	4.000000	1.00000	1.000000	

	EstimatedSalary	Exited

count	10000.000000	10000.000000
mean	100090.239881	0.203700
std	57510.492818	0.402769
min	11.580000	0.000000
25%	51002.110000	0.000000
50%	100193.915000	0.000000
75%	149388.247500	0.000000
max	199992.480000	1.000000

[5]:
```
data.head(10)
```

[5]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	\
0	1	15634602	Hargrave	619	France	Female	42	
1	2	15647311	Hill	608	Spain	Female	41	
2	3	15619304	Onio	502	France	Female	42	
3	4	15701354	Boni	699	France	Female	39	
4	5	15737888	Mitchell	850	Spain	Female	43	
5	6	15574012	Chu	645	Spain	Male	44	
6	7	15592531	Bartlett	822	France	Male	50	
7	8	15656148	Obinna	376	Germany	Female	29	
8	9	15792365	He	501	France	Male	44	
9	10	15592389	H?	684	France	Male	27	

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
0	2	0.00	1	1	1	
1	1	83807.86	1	0	1	
2	8	159660.80	3	1	0	
3	1	0.00	2	0	0	
4	2	125510.82	1	1	1	
5	8	113755.78	2	1	0	
6	7	0.00	2	1	1	
7	4	115046.74	4	1	0	
8	4	142051.07	2	0	1	
9	2	134603.88	1	1	1	

	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0
5	149756.71	1
6	10062.80	0
7	119346.88	1
8	74940.50	0
9	71725.73	0

[6]:
```
data.tail()
```

[6]:

	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

[7]:
```
data.tail(10)
```

[7]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9990	9991	15798964	Nkemakonam	714	Germany	Male	33	3	35016.60	1	1	0		
9991	9992	15769959	Ajuluchukwu	597	France	Female	53	4	88381.21	1	1	0		
9992	9993	15657105	Chukwualuka	726	Spain	Male	36	2	0.00	1	1	0		
9993	9994	15569266	Rahman	644	France	Male	28	7	155060.41	1	1	0		
9994	9995	15719294	Wood	800	France	Female	29	2	0.00	2	0	0		
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0		
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1		
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1		
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0		
9999	10000	15628319	Walker	792	France	Female	28							

9999	4	130142.79	1	1	0

	EstimatedSalary	Exited
9990	53667.08	0
9991	69384.71	1
9992	195192.40	0
9993	29179.52	0
9994	167773.55	0
9995	96270.64	0
9996	101699.77	0
9997	42085.58	1
9998	92888.52	1
9999	38190.78	0

[8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999Data
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
```

[9]:
```
data.shape
```

[9]: (10000, 14)

[11]:
```
data.median()
```

```
C:\Users\janar vijay\AppData\Local\Temp\ipykernel_5088\4184645713.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select
only valid columns before calling the reduction.
```

```
data.median()
```

[11]:
```
RowNumber          5.000500e+03
CustomerId         1.569074e+07
CreditScore        6.520000e+02
Age                3.700000e+01
Tenure             5.000000e+00
Balance            9.719854e+04
NumOfProducts      1.000000e+00
HasCrCard          1.000000e+00
IsActiveMember     1.000000e+00
EstimatedSalary    1.001939e+05
Exited             0.000000e+00
dtype: float64
```

[12]:
```
data.mode()
```

[12]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	\
0	1	15565701	Smith	850.0	France	Male	37.0	
1	2	15565706	NaN	NaN	NaN	NaN	NaN	
2	3	15565714	NaN	NaN	NaN	NaN	NaN	
3	4	15565779	NaN	NaN	NaN	NaN	NaN	
4	5	15565796	NaN	Na N	Na N	Na N	NaN	
…	…	…	…	…	…	…	…	
9995	9996	15815628	NaN	…	…	…	…	NaN
9996	9997	15815645	NaN	Na N	Na N	Na N	NaN	
9997	9998	15815656	NaN	NaN	NaN	NaN	NaN	
9998	9999	15815660	NaN	NaN	NaN	NaN	NaN	
9999	10000	15815690	NaN	NaN	NaN	NaN	NaN	

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
0	2.0	0.0	1.0	1.0	1.0	
1	NaN	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	Na N	NaN	NaN	
…	…	…	…	…	…	
9995	NaN	NaN	…	NaN	NaN	
9996	NaN	NaN	Na N	NaN	NaN	
9997	NaN	NaN	NaN	NaN	NaN	
9998	NaN	NaN	NaN	NaN	NaN	
9999	NaN	NaN	NaN	NaN	NaN	

	EstimatedSalary	Exited0
	24924.92	0.0

1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN
…	…	…
9995	NaN	NaN
9996	NaN	NaN
9997	NaN	NaN
9998	NaN	NaN
9999	NaN	NaN

[10000 rows x 14 columns]

5. Handle the Missing values.

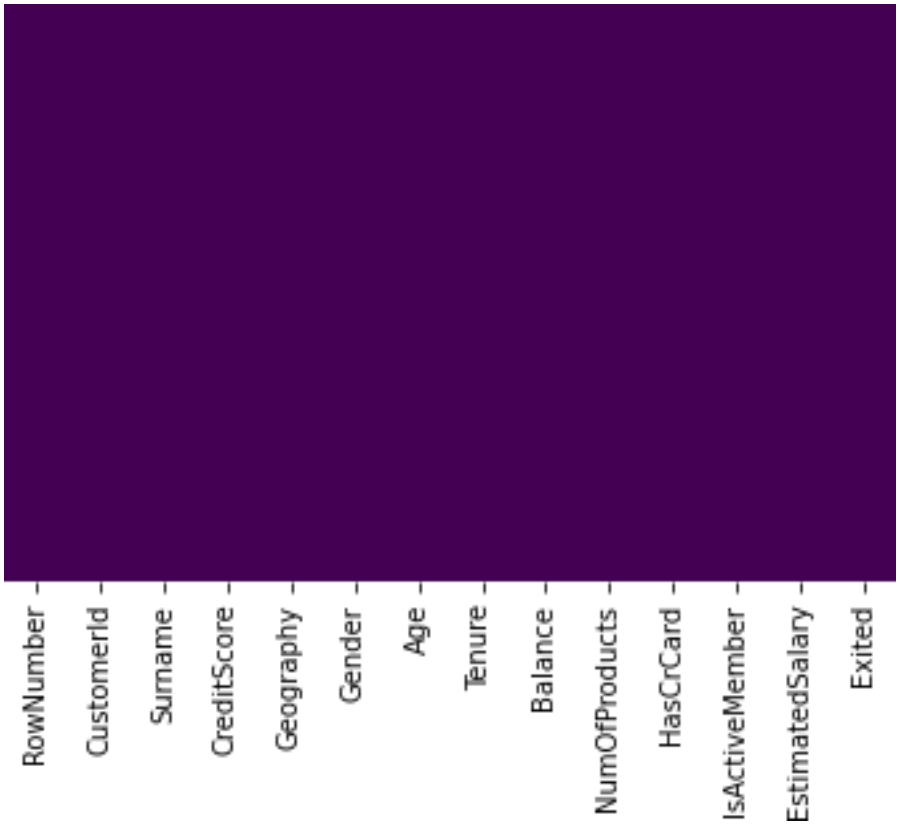
[7]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

#read data
train=pd.read_csv('model.csv',sep=',')
```

[15]:
```
df.isnull().any()
```

[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
```

[8]:
```
#missing data in model.csv
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

[8]: <AxesSubplot:>



in our data no missing values so we have to take titanic data set to perform handlng missing values////////

[7]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

#read data
train=pd.read_csv('train.csv')
train.head()
```

[7]:

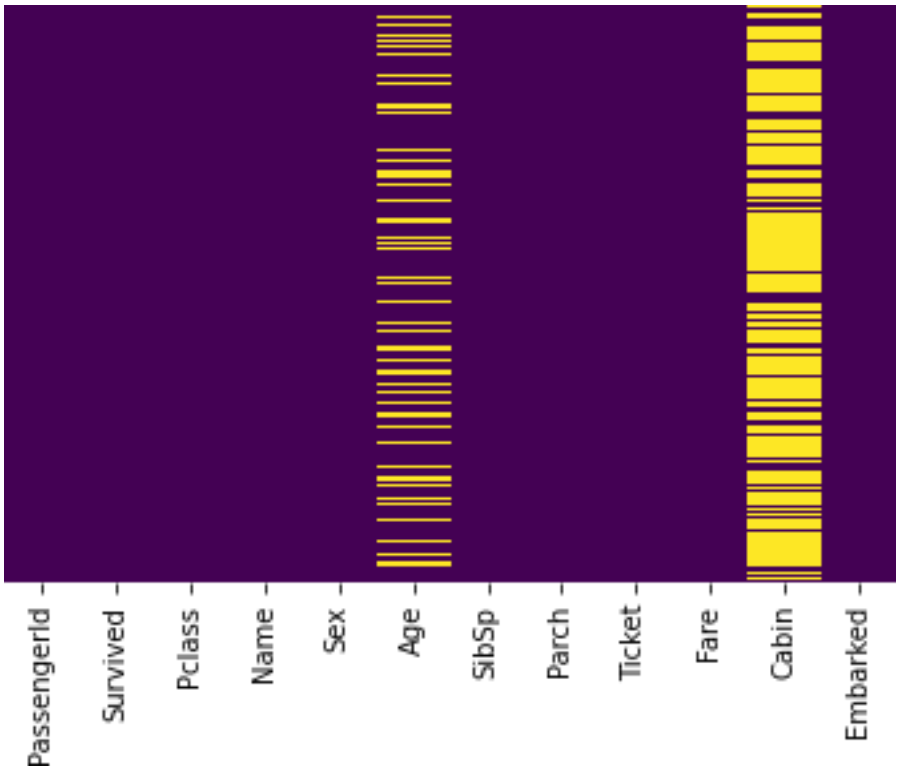
	PassengerId	Survived	Pclass	\
0		1	0	3
1		2	1	1
2		3	1	3
3		4	1	1
4		5	0	3

	Nam	Sex	Age	SibSp	\
0	eBraund, Mr. Owen Harris	male	22.0	1	
1	Cumings, Mrs. John Bradley (Florence Briggs Th…	female	38.0	1	
2	Heikkinen, Miss. Laina	female	26.0	0	
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	
4	Allen, Mr. William Henry	male	35.0	0	

	Parch	Ticket	Fare	Cabin	Embarked
0	0	A/5 21171	7.2500	NaN	S
1	0	PC 17599	71.2833	C85	C
2	0	STON/O2. 3101282	7.9250	NaN	S
3	0	113803	53.1000	C123	S
4	0	373450	8.0500	NaN	S

[4]:
```
#missing data
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

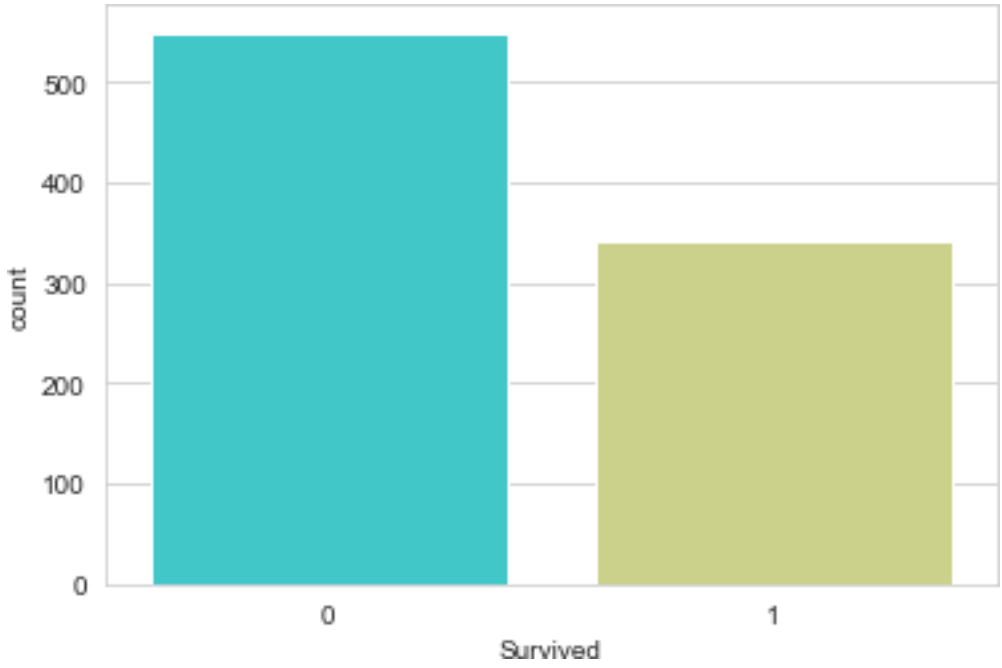
[4]: <AxesSubplot:>



missing values are shown above as diagramical representatiom

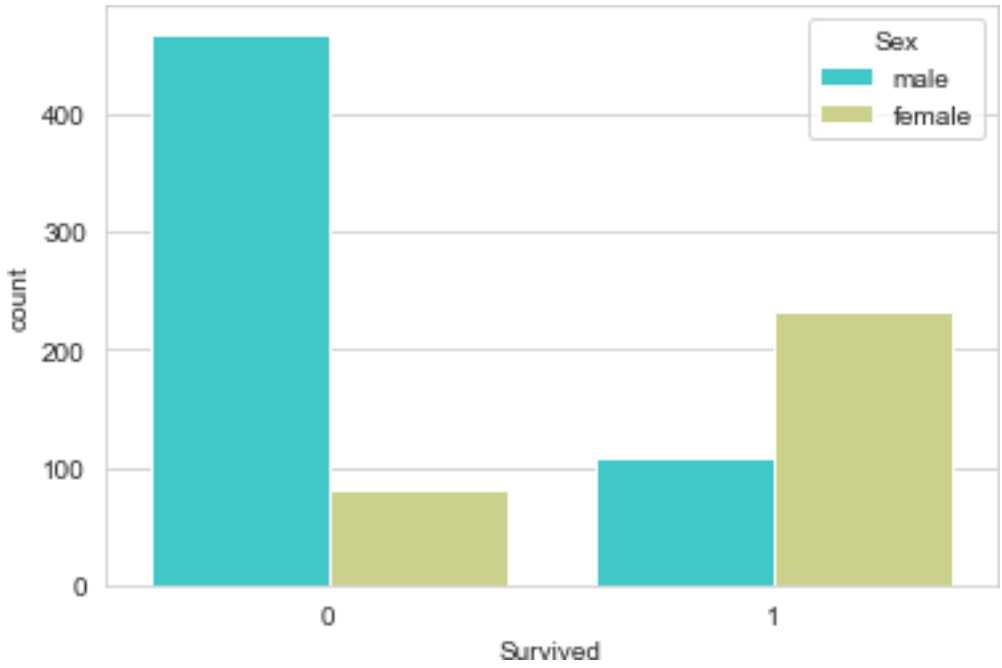
[14]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived',data=train,palette='rainbow')
```

[14]: <AxesSubplot:xlabel='Survived', ylabel='count'>



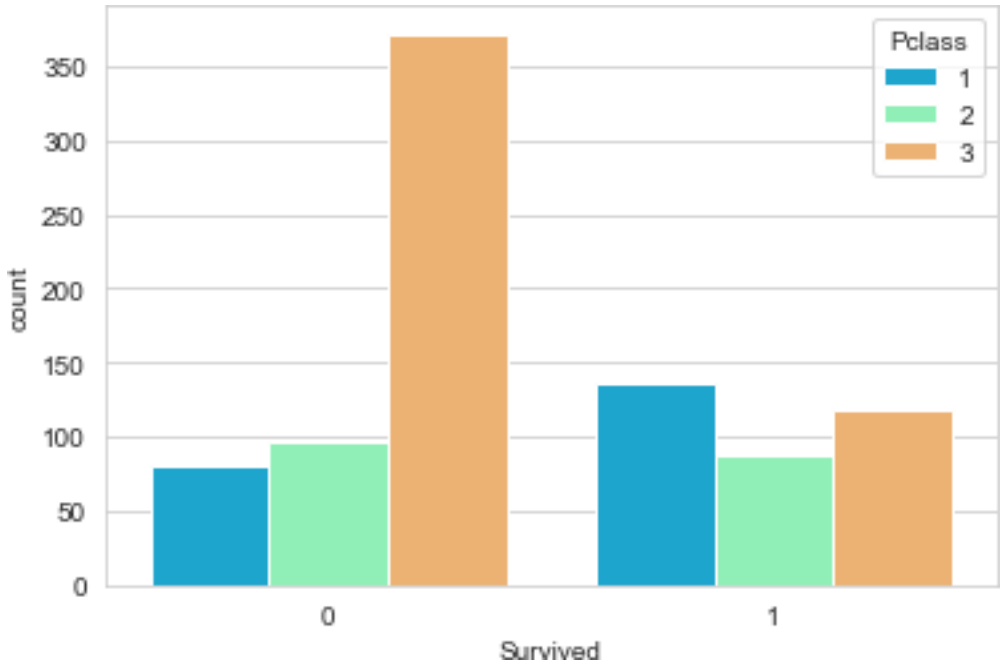
[13]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Sex',data=train,palette='rainbow')
```

[13]: <AxesSubplot:xlabel='Survived', ylabel='count'>



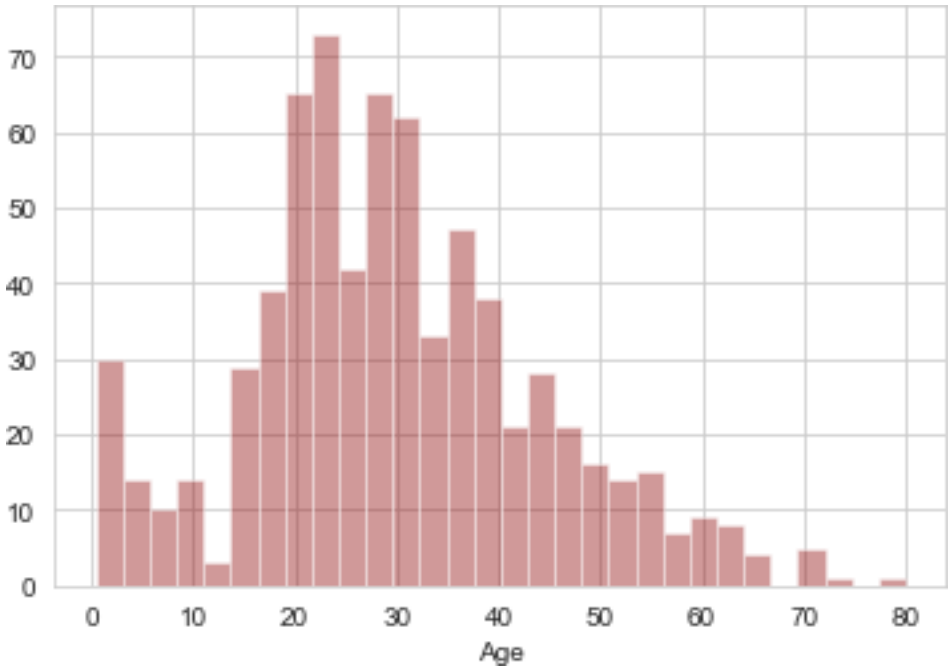
[12]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Pclass',data=train,palette='rainbow')
```

[12]: <AxesSubplot:xlabel='Survived', ylabel='count'>



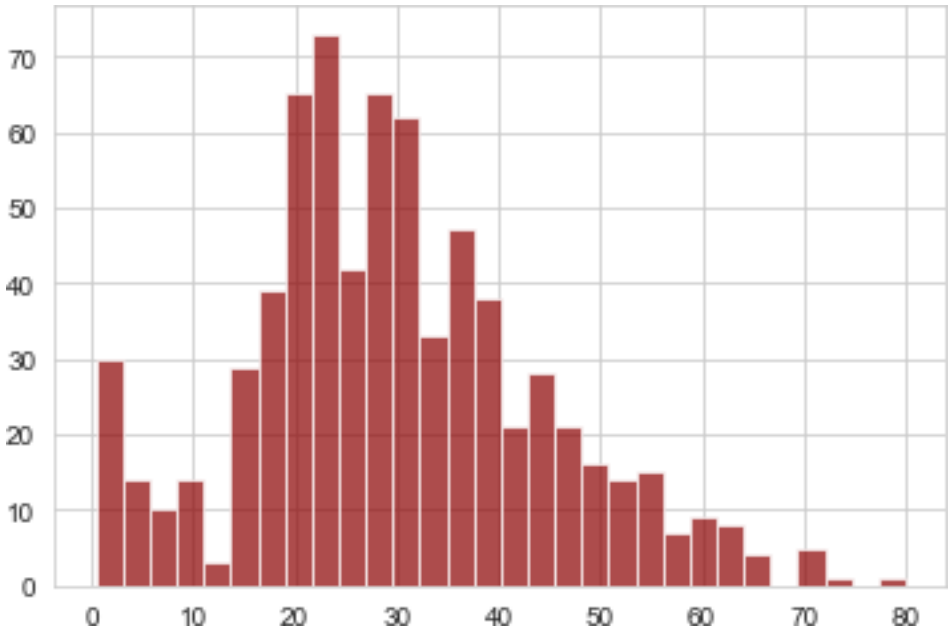
[17]:
```
sns.distplot(train['Age'].dropna(),kde=False,color='darkred',bins=30)
```

[17]: <AxesSubplot:xlabel='Age'>



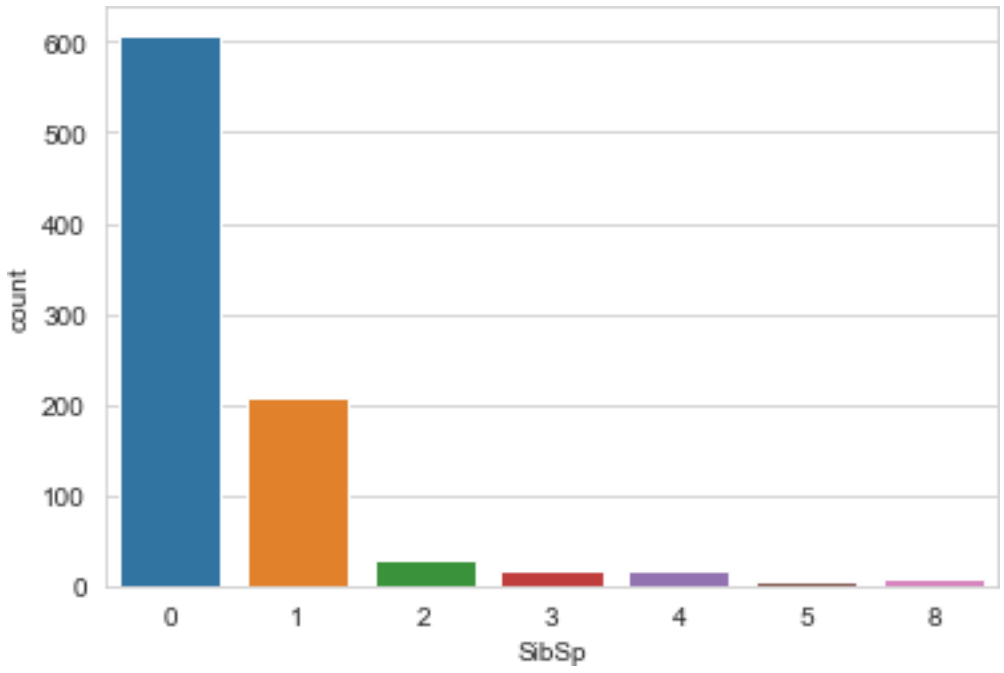
[18]:
```
train['Age'].hist(bins=30,color='darkred',alpha=0.7)
```

[18]: <AxesSubplot:>



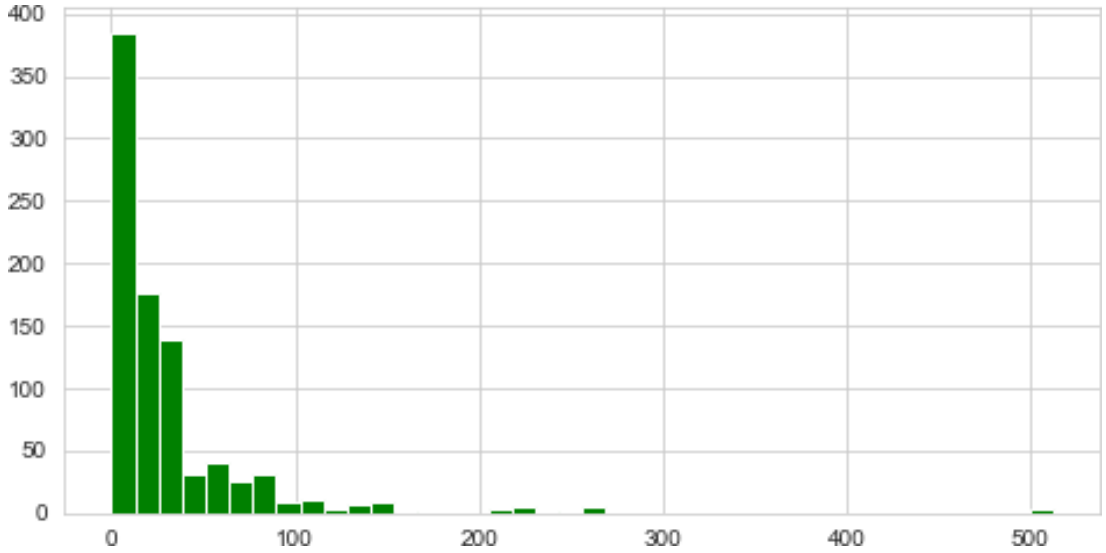
[19]:
```
sns.countplot(x='SibSp',data=train)
```

[19]: <AxesSubplot:xlabel='SibSp', ylabel='count'>



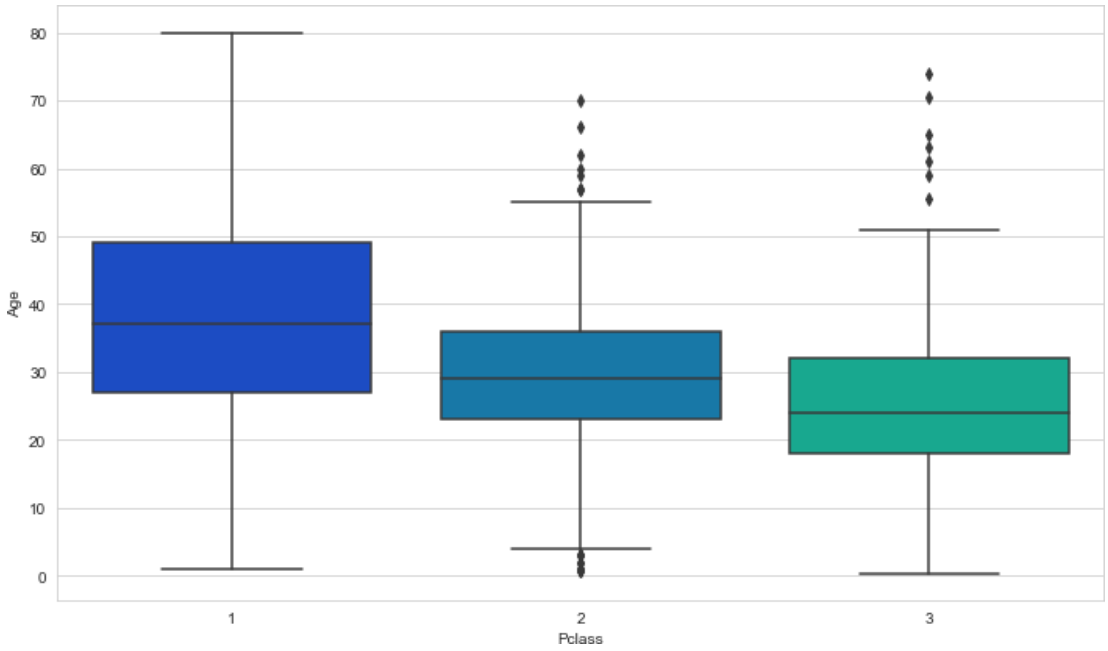
[20]:
```
train['Fare'].hist(color='green',bins=40,figsize=(8,4))
```

[20] : <AxesSubplot:>



[21]:
```
#Data cleaning
plt.figure(figsize=(12,7))
sns.boxplot(x='Pclass',y='Age',data=train,palette='winter')
```

[21] : <AxesSubplot:xlabel='Pclass', ylabel='Age'>



we ca see the wealthier passengers in the higher classes tend to be older,which makes sense averageage values to impute based on Pclass for Age

[29]:
```
def impute_age(cols):
    Age=cols[0]
    Pclass=cols[1]

    if pd.isnull(Age):

        if Pclass==1:
            return 37

        elif Pclass ==2:
            return 29

        else:
            return 24
    else:
        return Age
```

Now Apply This Function

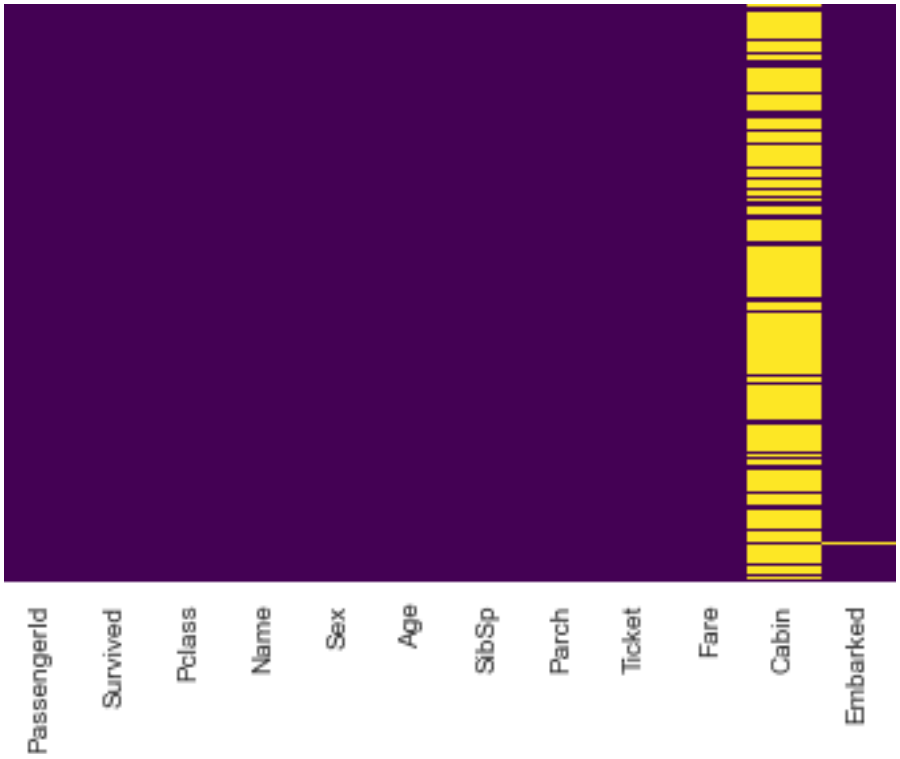
```
train['Age'] = train[['Age','Pclass']].apply(impute_age,axis=1)
```

Now let's check Heapmap Again..

[30]:

[31]:
```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

[31]: <AxesSubplot:>



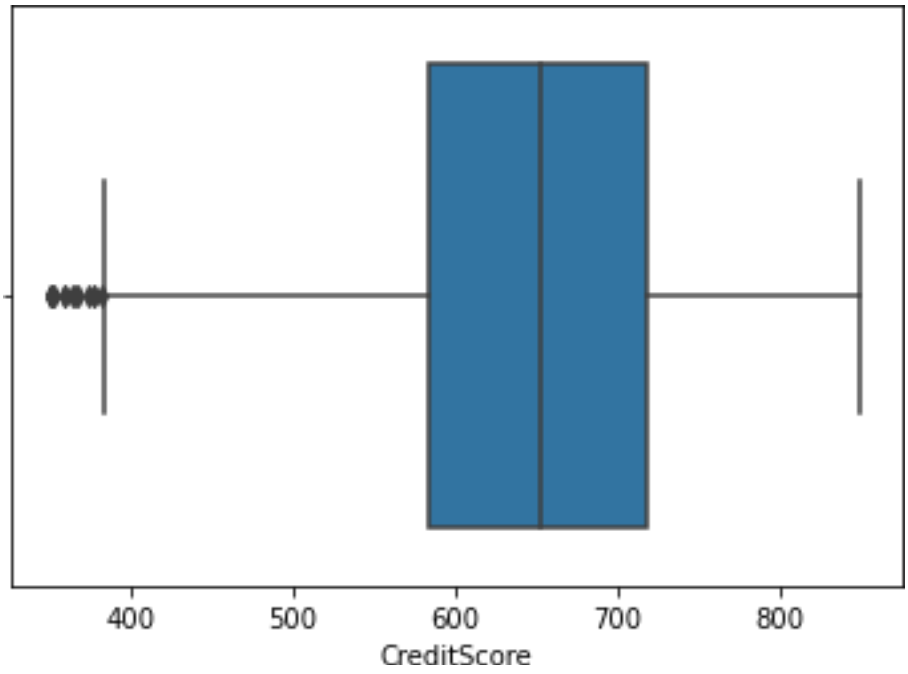
Now The Age Missing Values Can be Handled.

6. Find the outliers and replace the outliers

[11]:
```
#plotting outliers
sns.boxplot(df["CreditScore"])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

[11]: <AxesSubplot:xlabel='CreditScore'>



[16]:
```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#load the dataset
df=pd.read_csv('model.csv')
df
```

[16]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	\
0	1	15634602	Hargrave	619	France	Female	42	
1	2	15647311	Hill	608	Spain	Female	41	
2	3	15619304	Onio	502	France	Female	42	
3	4	15701354	Boni	699	France	Female	39	
4	5	15737888	Mitchell	850	Spain	Female	43	
…	…	…	…	…	…	…	…	
9995	9996	15606229	Obijiaku	771	France	Male	39	
9996	9997	15569892	Johnstone	516	France	Male	35	
9997	9998	15584532	Liu	709	France	Female	36	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	
9999	10000	15628319	Walker	792	France	Female	28	

Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\02
0.00	1	1	1		

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

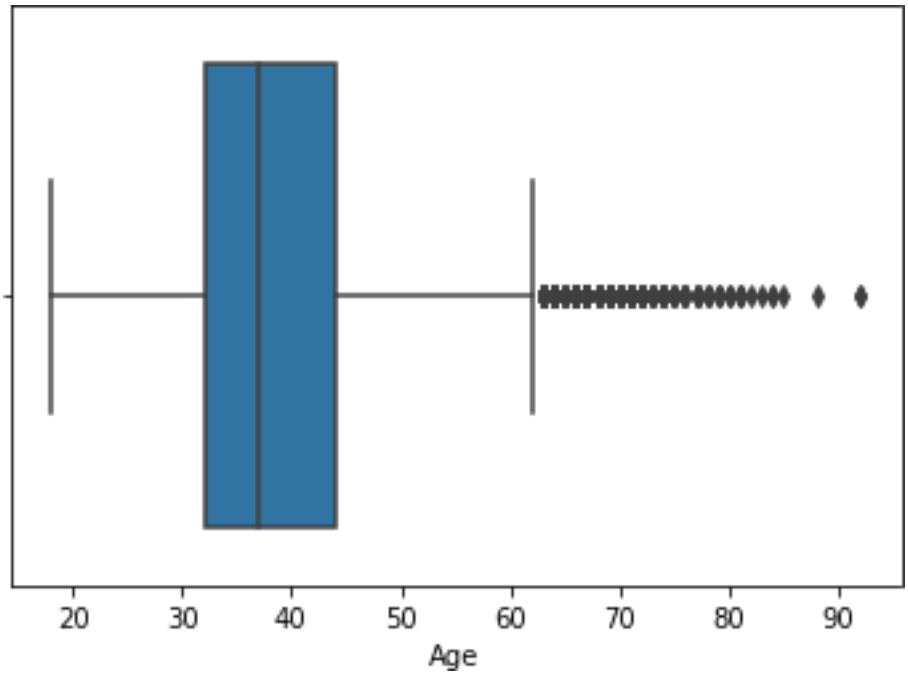
	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0
...	...	...
9995	96270.64	0
9996	101699.77	0
9997	42085.58	1
9998	92888.52	1
9999	38190.78	0

[10000 rows x 14 columns]

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

[39]:
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```

[39]: <AxesSubplot:xlabel='Age'>



[12]:
```
qnt=df.quantile(q=(0.75,0.25))

qnt
```

[12]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	\
0.75	7500.25	15753233.75	718.0	44.0	7.0	127644.24	
0.25	2500.75	15628528.25	584.0	32.0	3.0	0.00	

	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.75	2.0	1.0	1.0	149388.2475	0.0
0.25	1.0	0.0	0.0	51002.1100	0.0

upper=q3+1.5*iqr lower=q1-1.5*iqr iqr=q3-q1

[14]:
```
iqr = qnt.loc[0.75]-qnt.loc[0.25] #iqr calculations
iqr
```

[14]:

RowNumber	4999.5000
CustomerId	124705.5000
CreditScore	134.0000
Age	12.0000
Tenure	4.0000
Balance	127644.2400
NumOfProducts	1.0000
HasCrCard	1.0000

```
IsActiveMember          1.0000
EstimatedSalary     98386.1375
Exited                  0.0000
dtype: float64
```

[26]:
```
#lower extreme values
lower=qnt.loc[0.25] - 1.5*iqr
lower
```

[26]:
```
RowNumber           -4.998500e+03
CustomerId           1.544147e+07
CreditScore          3.830000e+02
Age                  1.400000e+01
Tenure              -3.000000e+00
Balance             -1.914664e+05
NumOfProducts       -5.000000e-01
HasCrCard           -1.500000e+00
IsActiveMember      -1.500000e+00
EstimatedSalary -9.657710e+04Exited
                     0.000000e+00
dtype: float64
```

[27]:
```
#upper extreme values
upper=qnt.loc[0.75] + 1.5*iqr

upper
```

[27]:
```
RowNumber            1.499950e+04
CustomerId           1.594029e+07
CreditScore          9.190000e+02
Age                  6.200000e+01
Tenure               1.300000e+01
Balance              3.191106e+05
NumOfProducts        3.500000e+00
HasCrCard            2.500000e+00
IsActiveMember       2.500000e+00
EstimatedSalary  2.969675e+05 Exited
                     0.000000e+00
dtype: float64
```

[18]:
```
df.mean()
```

```
C:\Users\janar vijay\AppData\Local\Temp\ipykernel_10016\3698961737.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select
only valid columns before calling the reduction.
  df.mean()
```

[18]:
```
RowNumber          5.000500e+03
CustomerId         1.569094e+07
CreditScore        6.505288e+02
Age                3.892180e+01
Tenure             5.012800e+00
Balance            7.648589e+04
NumOfProducts      1.530200e+00
HasCrCard          7.055000e-01
IsActiveMember     5.151000e-01
EstimatedSalary    1.000902e+05
Exited             2.037000e-01
dtype: float64
```

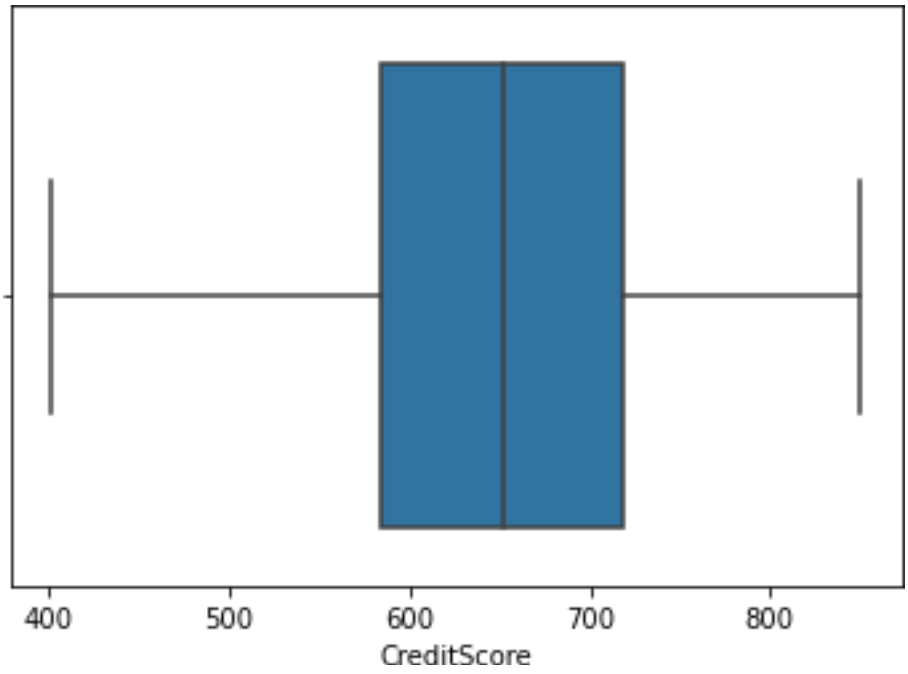
Replacing outlier

[45]:
```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#load the dataset
df=pd.read_csv('model.csv')
df['CreditScore']=np.where(df['CreditScore']<400,402,df['CreditScore'])
```

[49]:
```
#remove outlier on the CreditScore column
sns.boxplot(df["CreditScore"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```

[49]: <AxesSubplot:xlabel='CreditScore'>

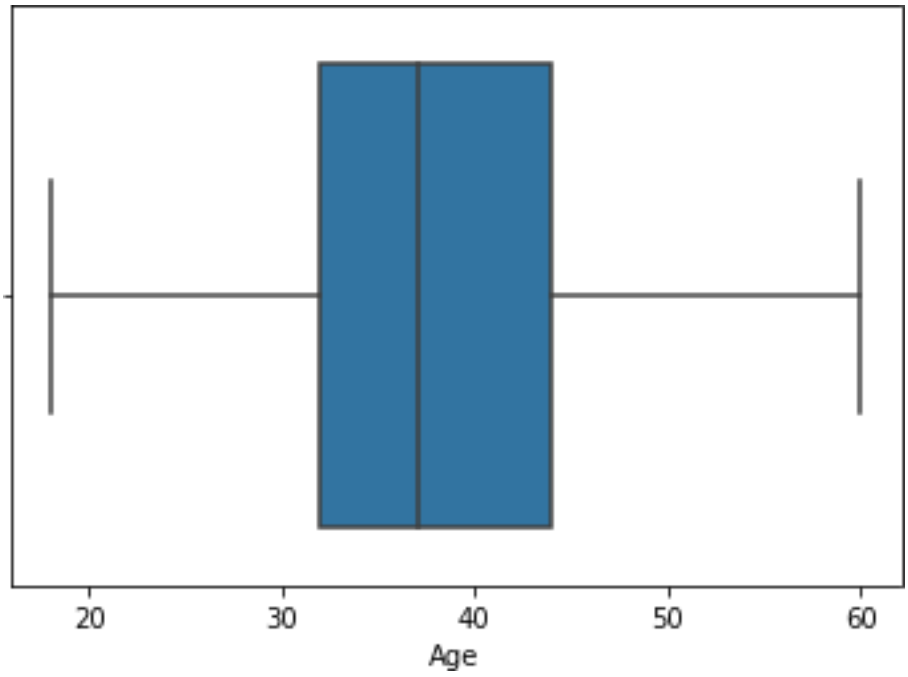


[50]:
```
#remove outlier on the Age column
df['Age']=np.where(df['Age']>60,50,df['Age'])
```

[51]:
```
sns.boxplot(df["Age"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```

[51]: <AxesSubplot:xlabel='Age'>



7. Check for Categorical columns and perform encoding

[53]:
```
df.head()
```

[53]:

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

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
0	2	0.00	1	1	1	
1	1	83807.86	1	0	1	
2	8	159660.80	3	1	0	
3	1	0.00	2	0	0	
4	2	125510.82	1	1	1	

	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0

encoding

[57]:
```
#manually handling categorincal data

df['Gender'].replace({'Female':1,'Male':2},inplace=True)
df.head()
```

[57]:

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

[62]:
```
df['Geography'].replace({'France':100,'Spain':200},inplace=True)
df.head()
```

[62]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	100	1	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	200	1	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	100	1	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	100	1	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	200	1	43	2	125510.82	1	1	1	79084.10	0

```
2          113931.57          1
3           93826.63          0
4           79084.10          0
```

[60]:
```
#dummy variable function
df_main=pd.get_dummies(df,columns=['Geography'])
df_main
```

[60]:
```
      RowNumber  CustomerId     Surname  CreditScore  Gender  Age  Tenure  \
0             1    15634602    Hargrave          619       1   42       2
1             2    15647311        Hill          608       1   41       1
2             3    15619304        Onio          502       1   42       8
3             4    15701354        Boni          699       1   39       1
4             5    15737888    Mitchell          850       1   43       2
...         ...         ...         ...          ...     ...  ...     ...
9995       9996    15606229     Obijiaku          771       2   39       5
9996       9997    15569892    Johnstone          516       2   35      10
9997       9998    15584532          Liu          709       1   36       7
9998       9999    15682355    Sabbatini          772       2   42       3
9999      10000    15628319       Walker          792       1   28       4

         Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  \
0           0.00              1          1               1        101348.88
1       83807.86              1          0               1        112542.58
2      159660.80              3          1               0        113931.57
3           0.00              2          0               0         93826.63
4      125510.82              1          1               1         79084.10
...          ...            ...        ...             ...              ...
9995        0.00              2          1               0         96270.64
9996    57369.61              1          1               1        101699.77
9997        0.00              1          0               1         42085.58
9998    75075.31              2          1               0         92888.52
9999   130142.79              1          1               0         38190.78

      Exited  Geography_100  Geography_200  Geography_Germany
0          1              1              0                  0
1          0              0              1                  0
2          1              1              0                  0
3          0              1              0                  0
4          0              0              1                  0
...      ...            ...            ...                ...
9995       0              1              0                  0
9996       0              1              0                  0
9997       1              1              0                  0
9998       1              0              0                  1
9999       0              1              0                  0
```

[10000 rows x 16 columns]

8. Split the data into dependent and independent variables.

[20]:
```
#target variable or  dependent variable.
x=df.iloc[:,0:2]
x
```

[20]:

	RowNumber	CustomerId
0	1	15634602
1	2	15647311
2	3	15619304
3	4	15701354
4	5	15737888
…	…	…
9995	9996	15606229
9996	9997	15569892
9997	9998	15584532
9998	9999	15682355
9999	10000	15628319

[10000 rows x 2 columns]

[22]:
```
#independent variables
y=df['EstimatedSalary']
y
```

[22]:
```
0         101348.88
1         112542.58
2         113931.57
3          93826.63
4          79084.10
             …
9995       96270.64
9996      101699.77
9997       42085.58
9998       92888.52
9999       38190.78
Name: EstimatedSalary, Length:       10000, dtype:     float64
```

9. Scale the independent variables

[23]:
```
from sklearn.preprocessing import scale
```

[27]:
```
x=scale(x)
x
```

[27]:
```
array([[-1.73187761,  -0.78321342],
       [-1.7315312 ,  -0.60653412],
       [-1.73118479,  -0.99588476],
       ...,
       [ 1.73118479,  -1.47928179],
       [ 1.7315312 ,  -0.11935577],
       [ 1.73187761,  -0.87055909]])
```

10. Split the data into training and testing

[41]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x_scale, y, test_size=0.
↪3, random_state=0)
x_train
```

[41]:
```
array([[ 0.92889885,  -0.79703192],
       [ 1.39655257,   0.71431365],
       [-0.4532777 ,   0.96344969],
       ...,
       [-0.60119484,  -1.62052514],
       [ 1.67853045,  -0.37403866],
       [-0.78548505,  -1.36411841]])
```

[42]:
```
x_train.shape
```

[42]: (7000, 2)

[43]:
```
x_test.shape
```

[43]: (3000, 2)

[44]:
```
y_train.shape
```

[44]: (7000,)

[45]:
```
y_test.shape
```

[45]: (3000,)