IN(1):

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

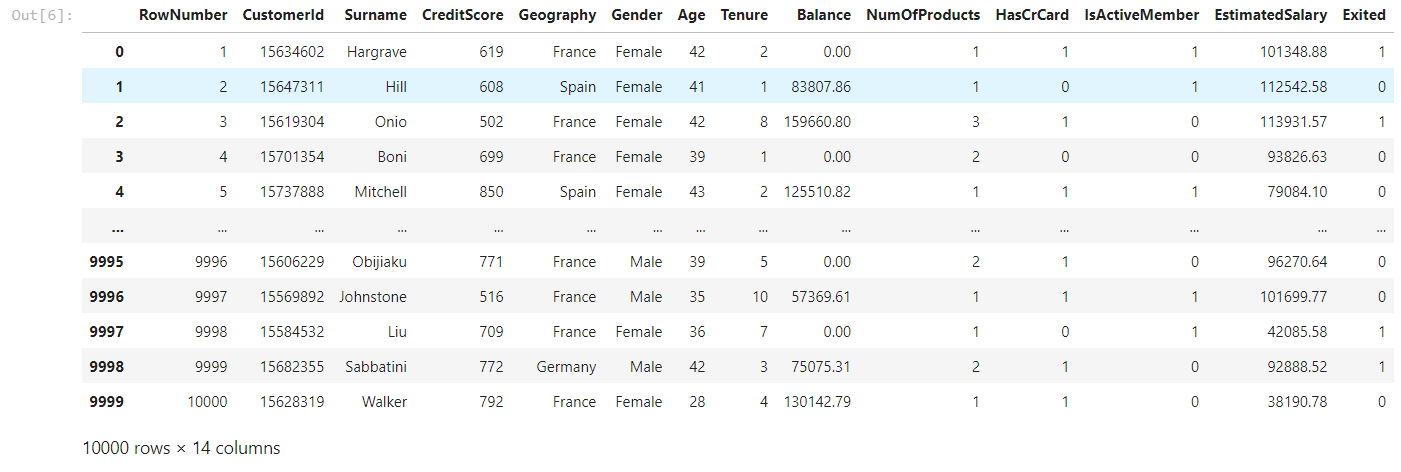
IN(2):

df=pd.read\_csv('/content/Churn\_Modelling.csv')

IN(6):

df

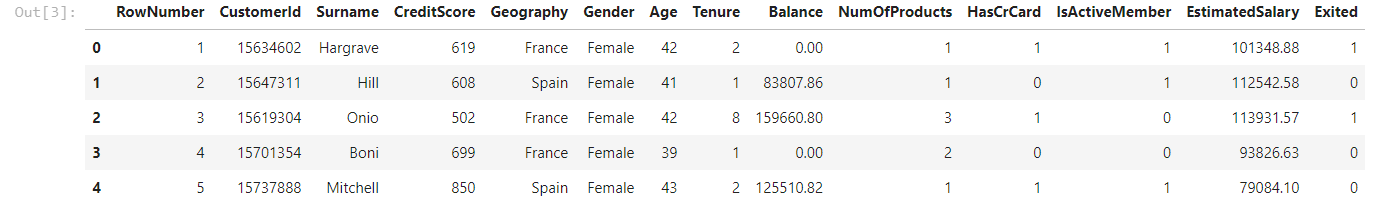
OP(6):



IN(3):

df.head()

OP(3):



IN(4):

df**.**shape

OP(4):



**Univariate,Bivariate and MultiVariate Analysis**

**Univariate Analysis**

IN[9]:

df\_france**=**df**.**loc[df['Geography']**==**'France']

df\_spain**=**df**.**loc[df['Geography']**==**'Spain']

df\_germany**=**df**.**loc[df['Geography']**==**'Germany']

In [17]:

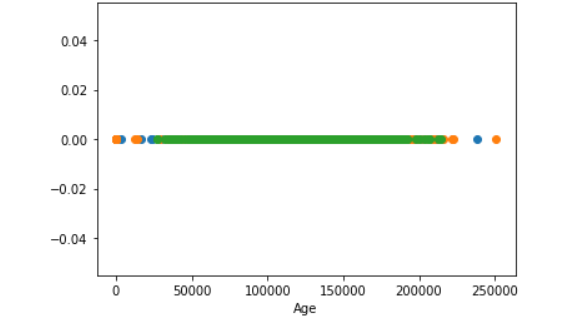
plt**.**plot(df\_france['Balance'],np**.**zeros\_like(df\_france['Balance']),'o')

plt**.**plot(df\_spain['Balance'],np**.**zeros\_like(df\_spain['Balance']),'o')

plt**.**plot(df\_germany['Balance'],np**.**zeros\_like(df\_germany['Balance']),'o')

plt**.**xlabel('Age')

plt**.**show()

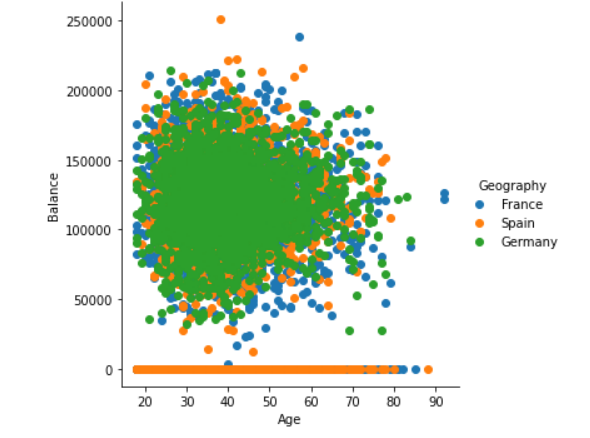


**Bivariate Analysis**

In [18]:

sns**.**FacetGrid(df,hue**=**"Geography",size**=**5)**.**map(plt**.**scatter,"Age","Balance")**.**add\_legend();

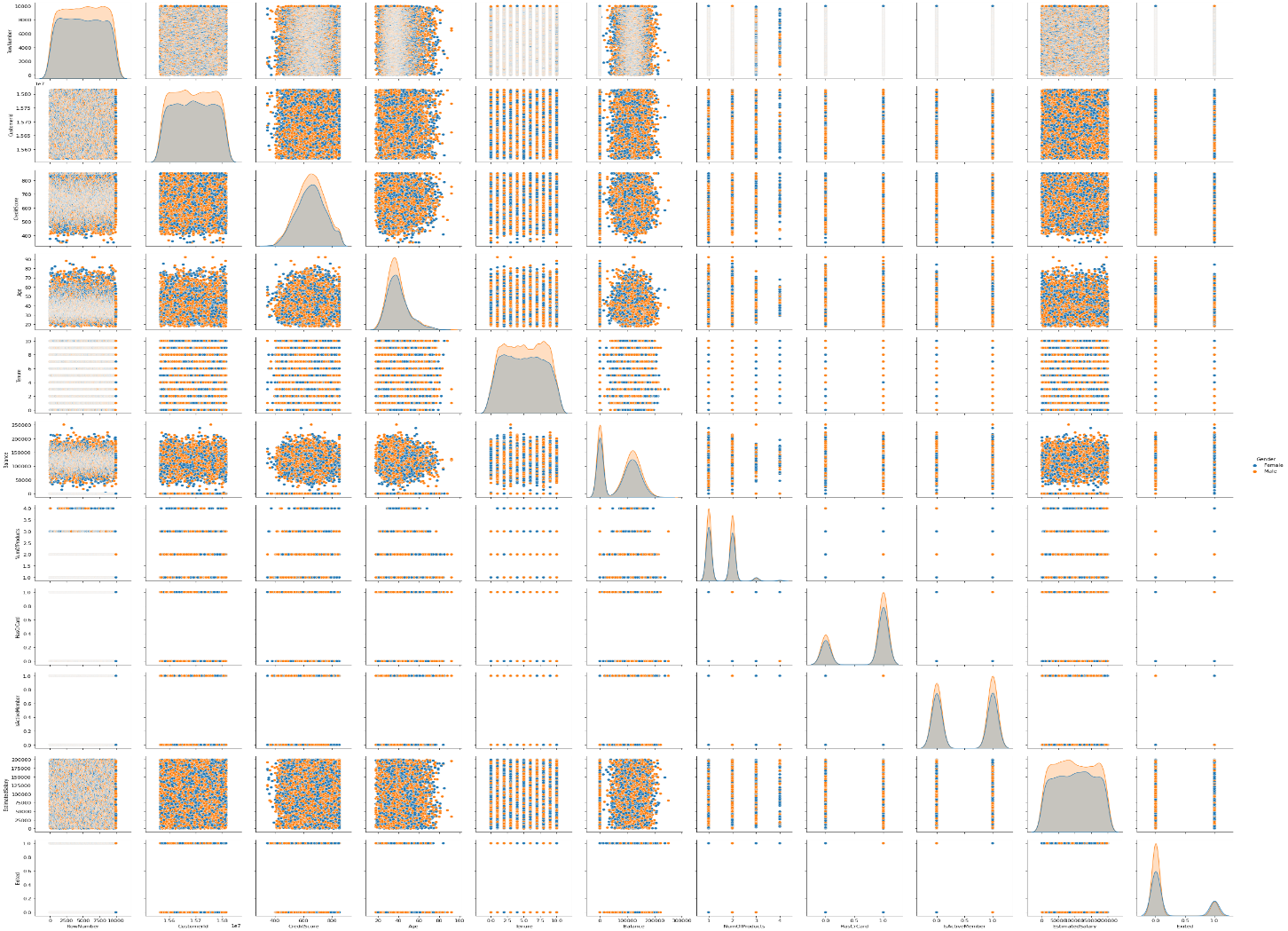
plt**.**show()



**Multivariate Analysis**

In [24]:

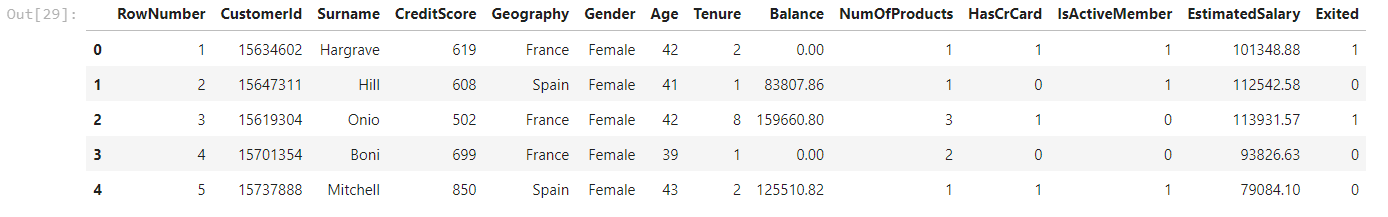
sns**.**pairplot(df,hue**=**"Gender",size**=**3)



**Descriptive Statistics**

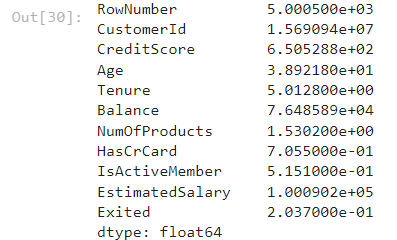
In [29]:

df**.**head()



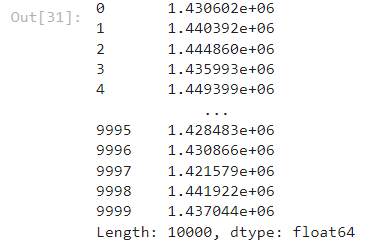
In [30]:

df.mean() # Get the mean of each column



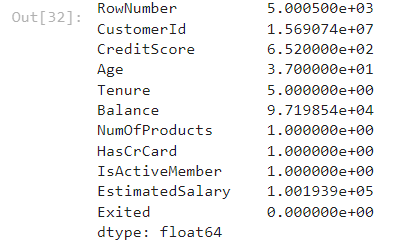
In [31]:

df.mean(axis=1) # Get the mean of each row



In [32]:

df.median() # Get the median of each column



In [39]:

norm\_data = pd.DataFrame(np.random.normal(size=100000))

norm\_data.plot(kind="density",

figsize=(10,10));

plt.vlines(norm\_data.mean(), # Plot black line at mean

ymin=0,

ymax=0.4,

linewidth=5.0);

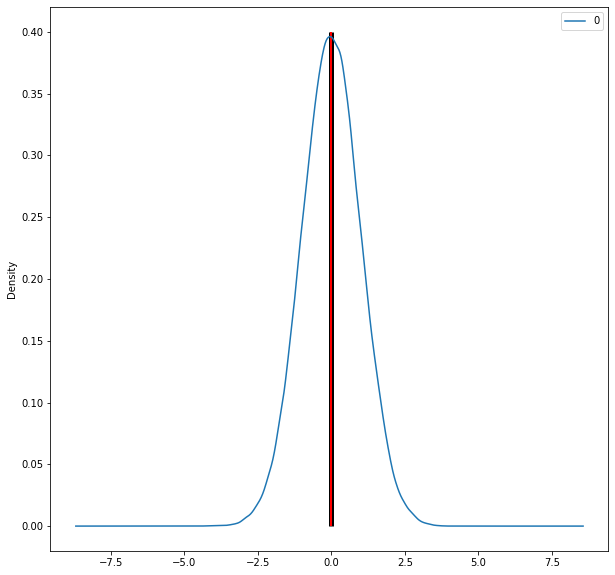
plt.vlines(norm\_data.median(), # Plot red line at median

ymin=0,

ymax=0.4,

linewidth=2.0,

color="red");



In [36]:

skewed\_data = pd.DataFrame(np.random.exponential(size=100000))

skewed\_data.plot(kind="density",

figsize=(10,10),

xlim=(-1,5));

plt.vlines(skewed\_data.mean(), # Plot black line at mean

ymin=0,

ymax=0.8,

linewidth=5.0);

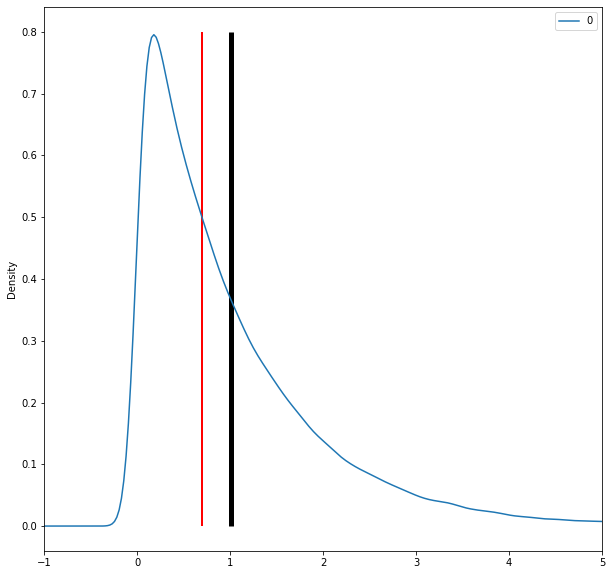
plt.vlines(skewed\_data.median(), # Plot red line at median

ymin=0,

ymax=0.8,

linewidth=2.0,

color="red");



In [40]:

norm\_data = np.random.normal(size=50)

outliers = np.random.normal(15, size=3)

combined\_data = pd.DataFrame(np.concatenate((norm\_data, outliers), axis=0))

combined\_data.plot(kind="density",

figsize=(10,10),

xlim=(-5,20));

plt.vlines(combined\_data.mean(), # Plot black line at mean

ymin=0,

ymax=0.2,

linewidth=5.0);

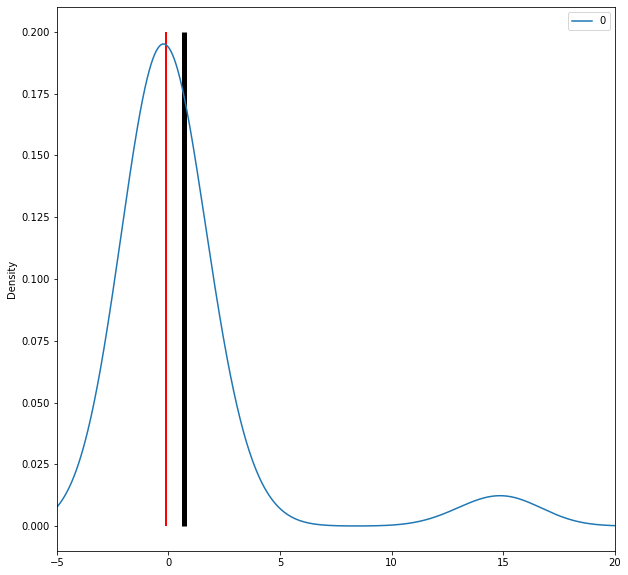
plt.vlines(combined\_data.median(), # Plot red line at median

ymin=0,

ymax=0.2,

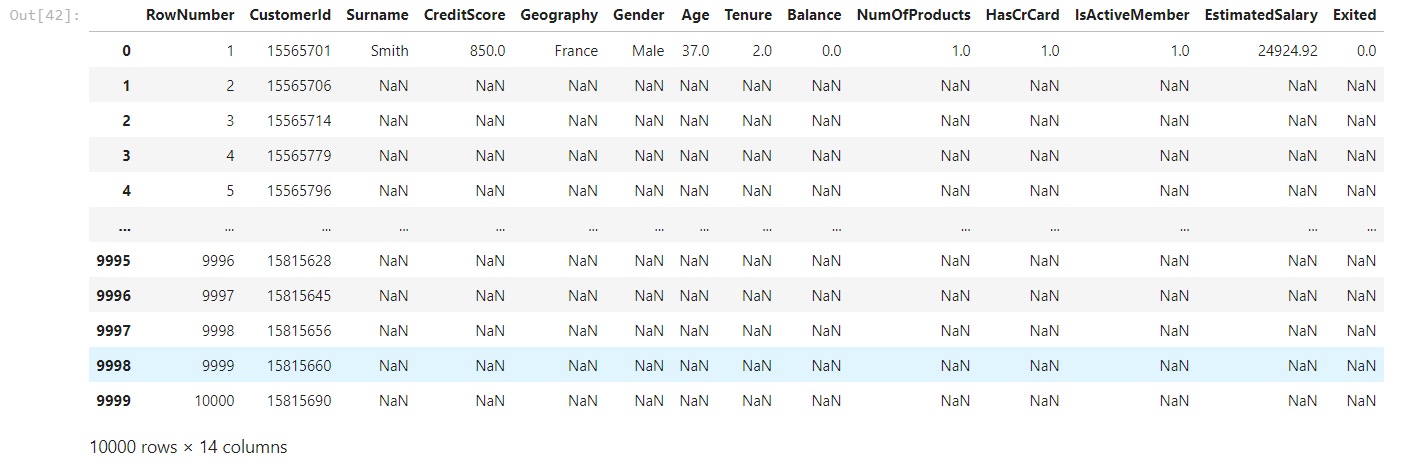
linewidth=2.0,

color="red");



In [42]:

df.mode()



**Measures of Spread**

In [43]:

max(df["Age"]) - min(df["Age"])



In [45]:

five\_num = [df["Age"].quantile(0),

df["Age"].quantile(0.25),

df["Age"].quantile(0.50),

df["Age"].quantile(0.75),

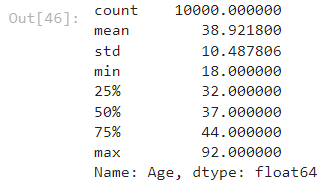
df["Age"].quantile(1)]

five\_num



In [46]:

df["Age"].describe()



In [47]:

df["Age"].quantile(0.75) - df["Age"].quantile(0.25)



In [49]:

df.boxplot(column="Age",

return\_type='axes',

figsize=(8,8))

plt.text(x=0.74, y=22.25, s="3rd Quartile")

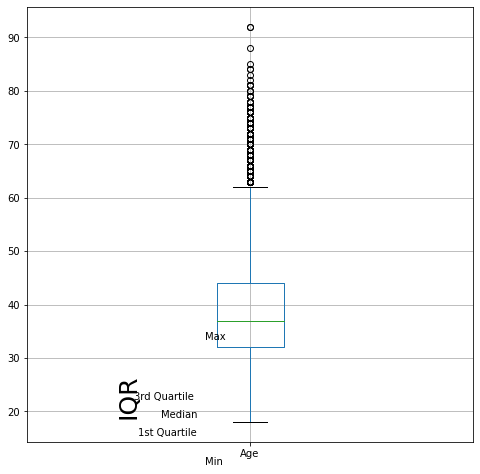
plt.text(x=0.8, y=18.75, s="Median")

plt.text(x=0.75, y=15.5, s="1st Quartile")

plt.text(x=0.9, y=10, s="Min")

plt.text(x=0.9, y=33.5, s="Max")

plt.text(x=0.7, y=19.5, s="IQR", rotation=90, size=25);



In [50]:

df["Age"].var()



In [51]:

df["Age"].std()



In [52]:

abs\_median\_devs = abs(df["Age"] - df["Age"].median())

abs\_median\_devs.median() \* 1.4826



**Skewness and Kurtosis**

In [53]:

df["Age"].skew() # Check skewness



In [54]:

df["Age"].kurt() # Check kurtosis



In [55]:

norm\_data = np.random.normal(size=100000)

skewed\_data = np.concatenate((np.random.normal(size=35000)+2,

np.random.exponential(size=65000)),

axis=0)

uniform\_data = np.random.uniform(0,2, size=100000)

peaked\_data = np.concatenate((np.random.exponential(size=50000),

np.random.exponential(size=50000)\*(-1)),

axis=0)

data\_df = pd.DataFrame({"norm":norm\_data,

"skewed":skewed\_data,

"uniform":uniform\_data,

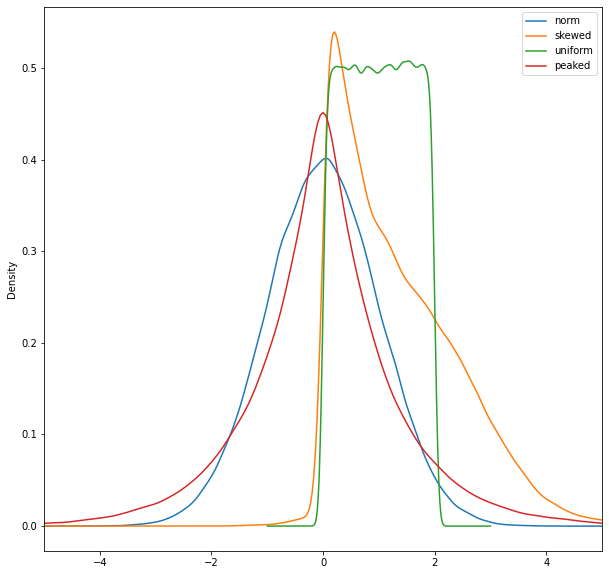
"peaked":peaked\_data})

In [56]:

data\_df.plot(kind="density",

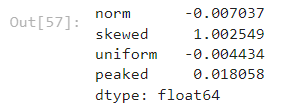
figsize=(10,10),

xlim=(-5,5));



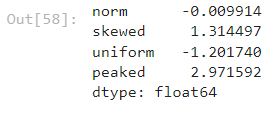
In [57]:

data\_df.skew()



In [58]:

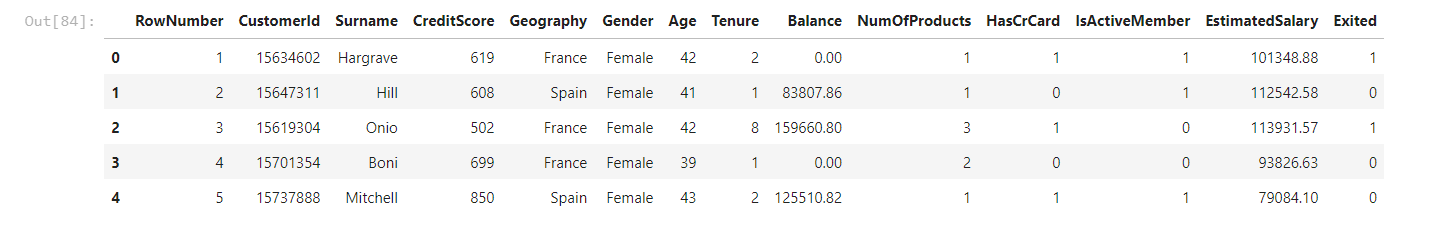
data\_df.kurt()



**Handle the Missing values**

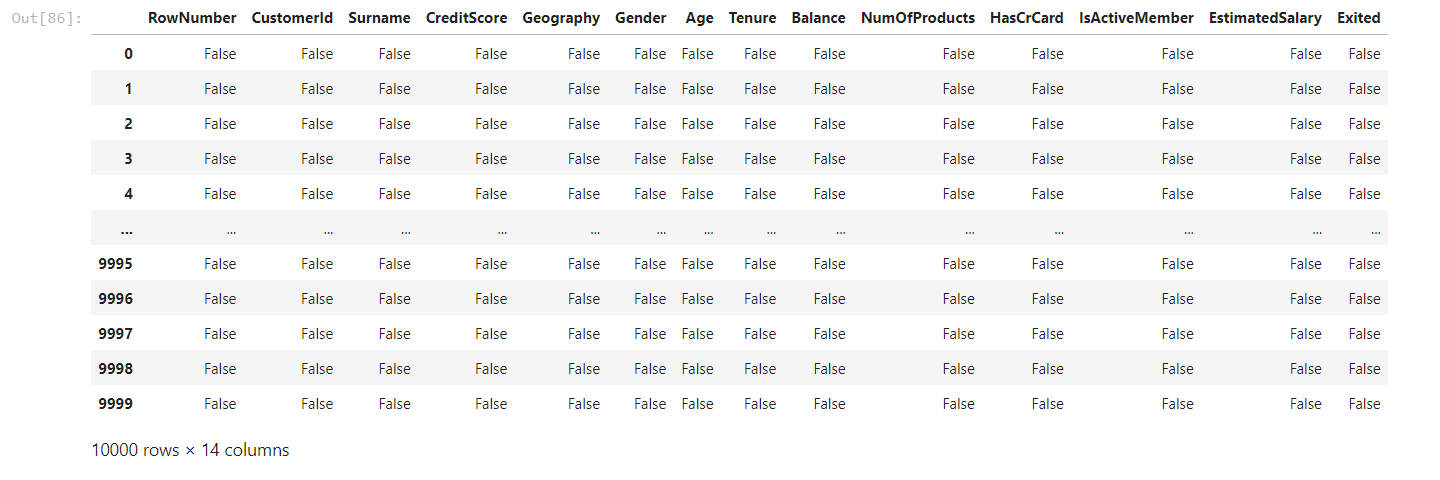
In [83]:

df=pd.read\_csv('/content/Churn\_Modelling.csv')



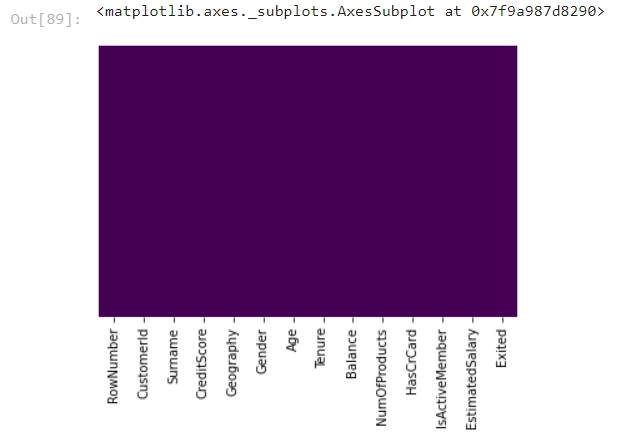
In [84]:

df.head()



In [86]:

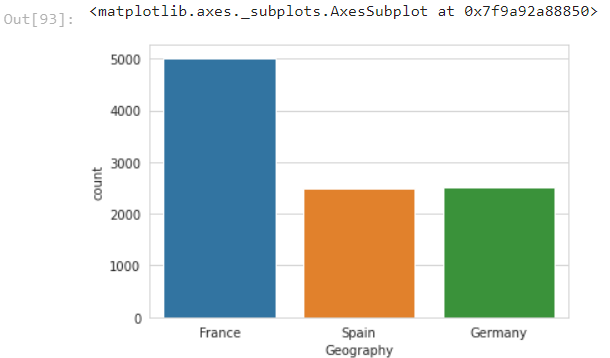
df.isnull()



In [93]:

sns.set\_style('whitegrid')

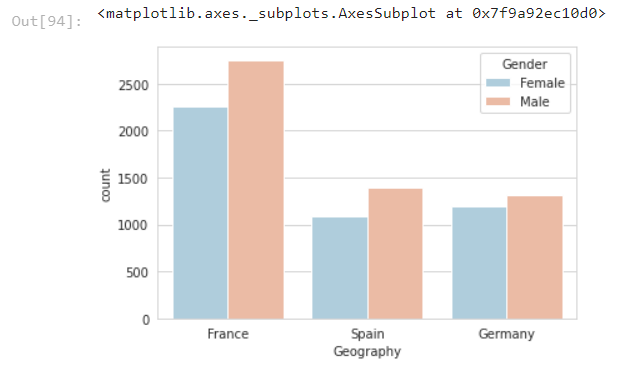
sns.countplot(x='Geography',data=df)



In [94]:

sns.set\_style('whitegrid')

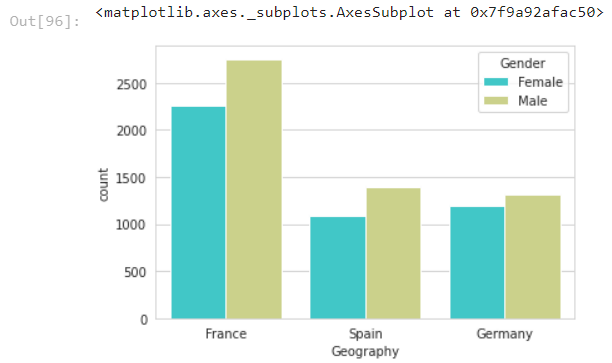
sns.countplot(x='Geography',hue='Gender',data=df,palette='RdBu\_r')



In [96]:

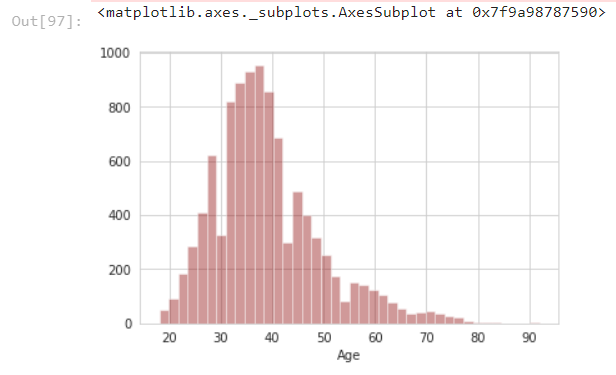
sns.set\_style('whitegrid')

sns.countplot(x='Geography',hue='Gender',data=df,palette='rainbow')



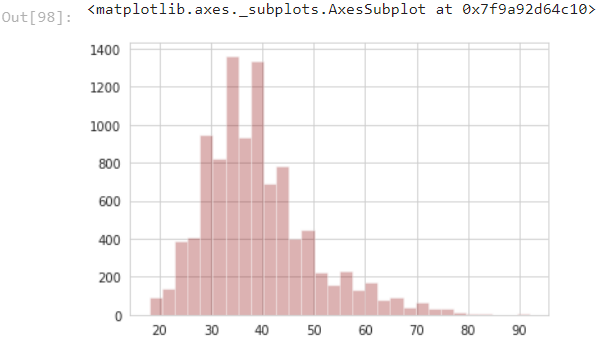
In [97]:

sns.distplot(df['Age'].dropna(),kde=False,color='darkred',bins=40)



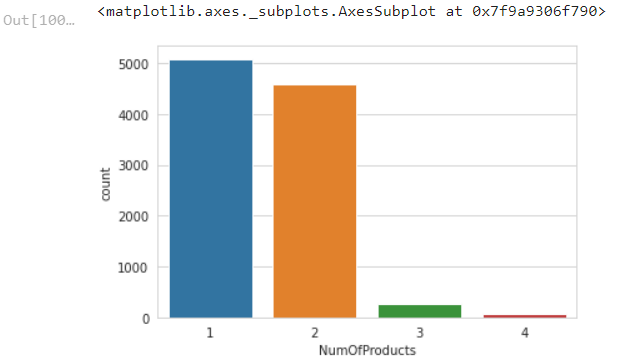
In [98]:

df['Age'].hist(bins=30,color='darkred',alpha=0.3)



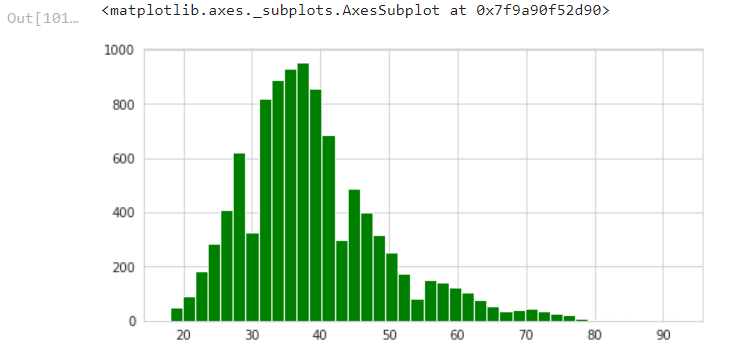
In [100]:

sns.countplot(x='NumOfProducts',data=df)



In [101]:

df['Age'].hist(color='green',bins=40,figsize=(8,4))



**Cufflinks for plots**

In [102]:

**import** cufflinks **as** cf

cf**.**go\_offline()

In [ ]:

df['Age']**.**iplot(kind**=**'hist',bins**=**30,color**=**'green')

**Data Cleaning**

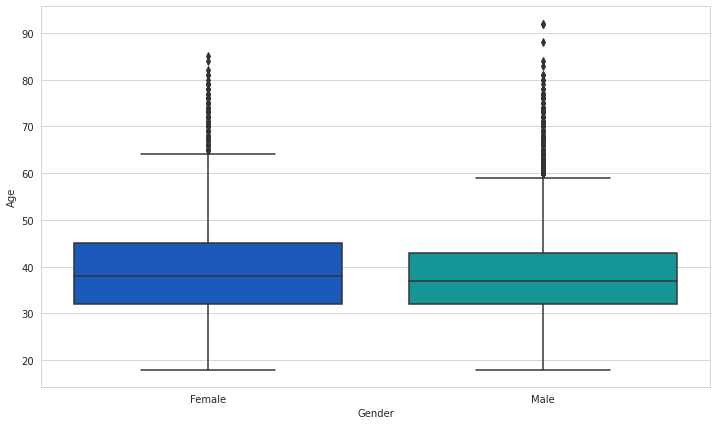
In [107]:

plt**.**figure(figsize**=**(12, 7))

sns**.**boxplot(x**=**'Gender',y**=**'Age',data**=**df,palette**=**'winter')

Out[107]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f9a90f59450>



In [307]:

**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**:

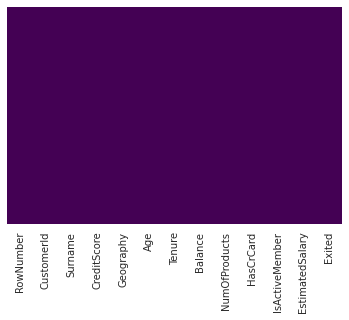
**return** Age

In [122]:

sns**.**heatmap(df**.**isnull(),yticklabels**=False**,cbar**=False**,cmap**=**'viridis')

Out[122]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f9a8aa699d0>



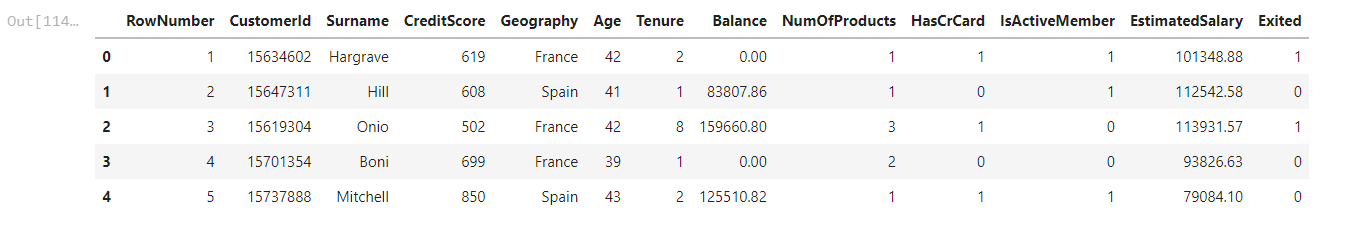
In [112]:

df**.**drop('Gender',axis**=**1,inplace**=True**)

In [114]:

df**.**head()

Out[114]:



**Converting Categorical Features**

In [116]:

df**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 13 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 Age 10000 non-null int64

6 Tenure 10000 non-null int64

7 Balance 10000 non-null float64

8 NumOfProducts 10000 non-null int64

9 HasCrCard 10000 non-null int64

10 IsActiveMember 10000 non-null int64

11 EstimatedSalary 10000 non-null float64

12 Exited 10000 non-null int64

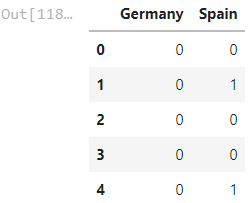
dtypes: float64(2), int64(9), object(2)

memory usage: 1015.8+ KB

In [118]:

pd**.**get\_dummies(df['Geography'],drop\_first**=True**)**.**head()

Out[118]:



In [124]:

df**.**info

Out[124]:

<bound method DataFrame.info of RowNumber CustomerId Surname CreditScore Geography Age Tenure \

0 1 15634602 Hargrave 619 France 42 2

1 2 15647311 Hill 608 Spain 41 1

2 3 15619304 Onio 502 France 42 8

3 4 15701354 Boni 699 France 39 1

4 5 15737888 Mitchell 850 Spain 43 2

... ... ... ... ... ... ... ...

9995 9996 15606229 Obijiaku 771 France 39 5

9996 9997 15569892 Johnstone 516 France 35 10

9997 9998 15584532 Liu 709 France 36 7

9998 9999 15682355 Sabbatini 772 Germany 42 3

9999 10000 15628319 Walker 792 France 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

0 1

1 0

2 1

3 0

4 0

... ...

9995 0

9996 0

9997 1

9998 1

9999 0

[10000 rows x 13 columns]>

In [125]:

sex **=** pd**.**get\_dummies(df['Age'],drop\_first**=True**)

embark **=** pd**.**get\_dummies(df['Balance'],drop\_first**=True**)

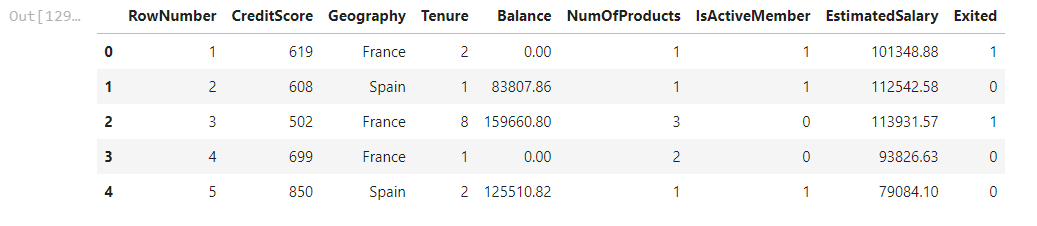
In [127]:

df**.**drop(['Age','HasCrCard','Surname','CustomerId'],axis**=**1,inplace**=True**)

In [129]:

df**.**head()

Out[129]:



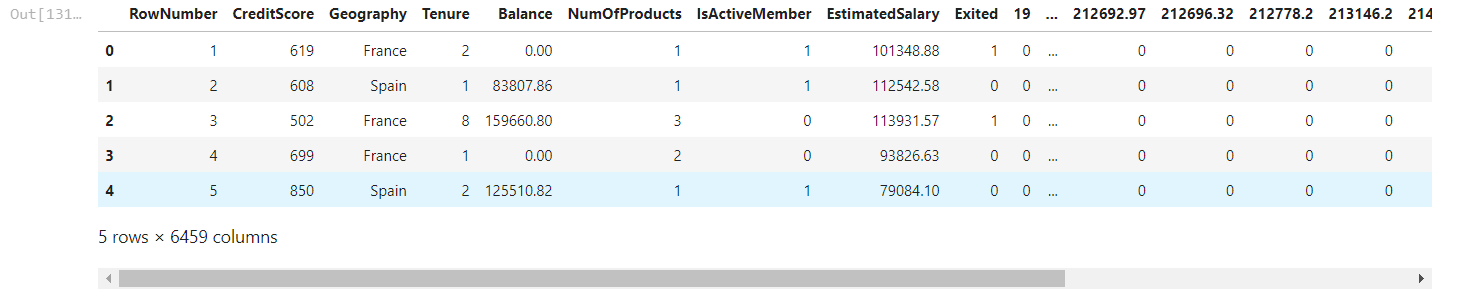
In [130]:

train **=** pd**.**concat([df,sex,embark],axis**=**1)

In [131]:

train**.**head()

Out[131]:



**Find the outliers and replace the outliers**

In [147]:

dataset**=** [11,10,12,14,12,15,14,13,15,102,12,14,17,19,107, 10,13,12,14,12,108,12,11,14,13,15,10,15,12,10,14,13,15,10]

**Detecting outlier using Z score**

**Using Z score**

In [148]:

outliers**=**[]

**def** detect\_outliers(data):

threshold**=**3

mean **=** np**.**mean(data)

std **=**np**.**std(data)

**for** i **in** data:

z\_score**=** (i **-** mean)**/**std

**if** np**.**abs(z\_score) **>** threshold:

outliers**.**append(y)

**return** outliers

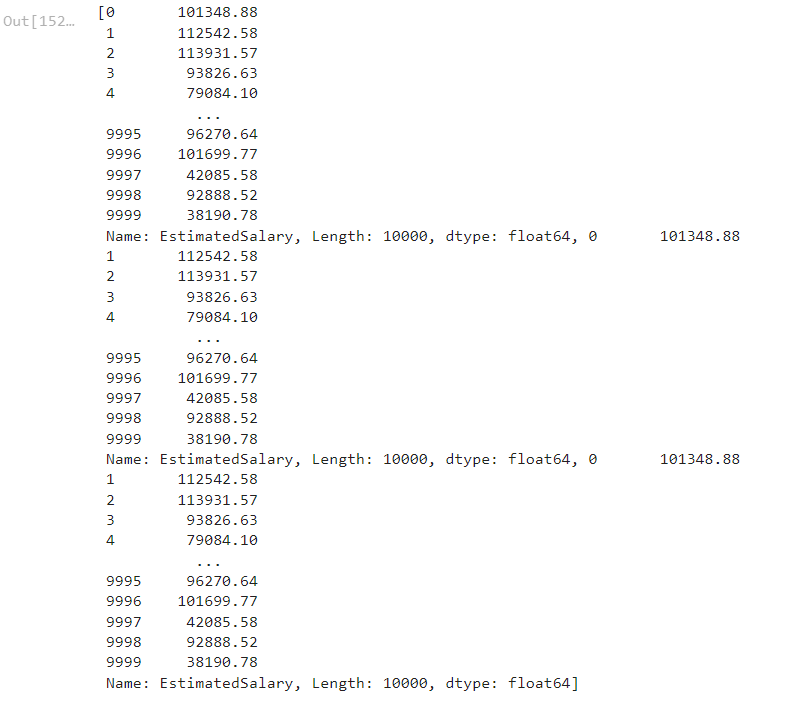
In [151]:

outlier\_pt**=**detect\_outliers(dataset)

In [152]:

outlier\_pt

Out[152]:

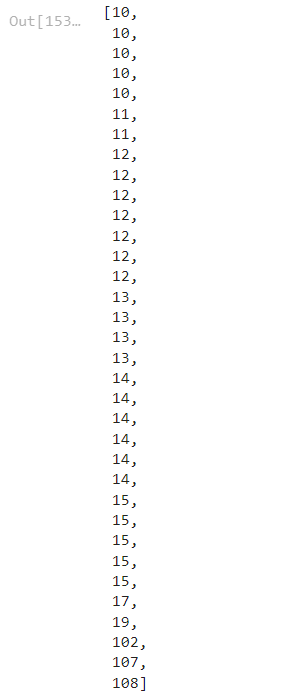


In [153]:

*## Perform all the steps of IQR*

sorted(dataset)

Out[153]:



In [155]:

quantile1, quantile3**=** np**.**percentile(dataset,[25,75])

In [156]:

print(quantile1,quantile3)

12.0 15.0

In [157]:

*## Find the IQR*

iqr\_value**=**quantile3**-**quantile1

print(iqr\_value)

3.0

In [159]:

*## Find the lower bound value and the higher bound value*

lower\_bound\_val **=** quantile1 **-**(1.5 **\*** iqr\_value)

upper\_bound\_val **=** quantile3 **+**(1.5 **\*** iqr\_value)

In [160]:

print(lower\_bound\_val,upper\_bound\_val)

7.5 19.5

**Check for Categorical columns and perform encoding**

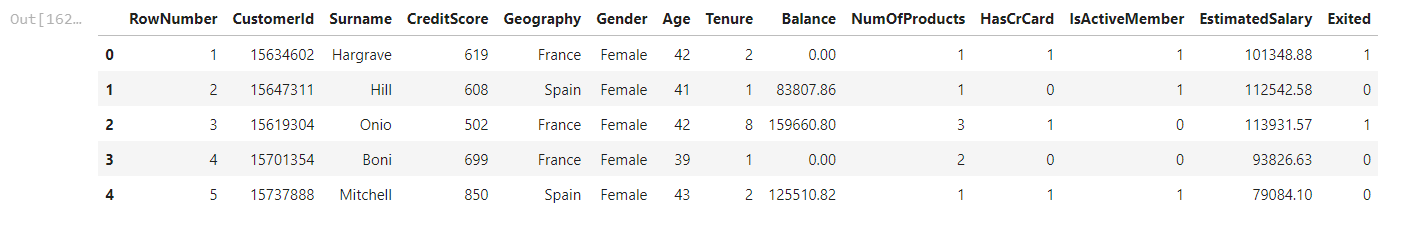
In [161]:

df**=**pd**.**read\_csv('/content/Churn\_Modelling.csv')

In [162]:

df**.**head()

Out[162]:



In [163]:

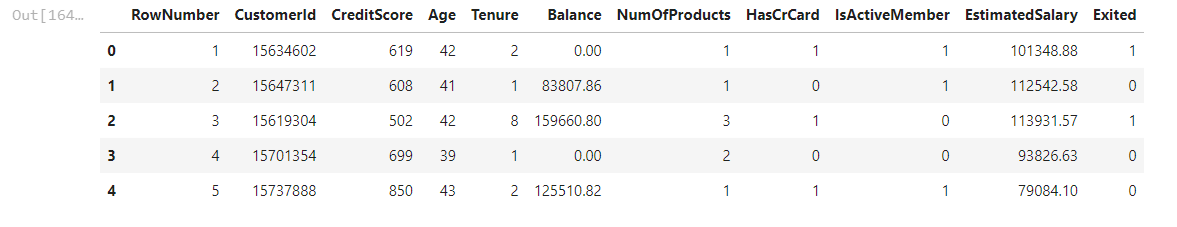
df\_numeric **=** df[['RowNumber', 'CustomerId', 'CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary','Exited']]

df\_categorical **=** df[['Surname', 'Geography', 'Gender']]

In [164]:

df\_numeric**.**head()

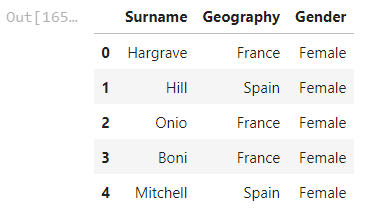
Out[164]:



In [165]:

df\_categorical**.**head()

Out[165]:



In [166]:

print(df['Surname']**.**unique())

print(df['Geography']**.**unique())

print(df['Gender']**.**unique())

['Hargrave' 'Hill' 'Onio' ... 'Kashiwagi' 'Aldridge' 'Burbidge']

['France' 'Spain' 'Germany']

['Female' 'Male']

In [167]:

**from** sklearn.preprocessing **import** LabelEncoder

marry\_encoder **=** LabelEncoder()

In [168]:

marry\_encoder**.**fit(df\_categorical['Gender'])

Out[168]:

LabelEncoder()

In [169]:

marry\_values **=** marry\_encoder**.**transform(df\_categorical['Gender'])

In [170]:

print("Before Encoding:", list(df\_categorical['Gender'][**-**10:]))

print("After Encoding:", marry\_values[**-**10:])

print("The inverse from the encoding result:", marry\_encoder**.**inverse\_transform(marry\_values[**-**10:]))

Before Encoding: ['Male', 'Female', 'Male', 'Male', 'Female', 'Male', 'Male', 'Female', 'Male', 'Female']

After Encoding: [1 0 1 1 0 1 1 0 1 0]

The inverse from the encoding result: ['Male' 'Female' 'Male' 'Male' 'Female' 'Male' 'Male' 'Female' 'Male' 'Female']

In [171]:

residence\_encoder **=** LabelEncoder()

residence\_values **=** residence\_encoder**.**fit\_transform(df\_categorical['Geography'])

print("Before Encoding:", list(df\_categorical['Geography'][:5]))

print("After Encoding:", residence\_values[:5])

print("The inverse from the encoding result:", residence\_encoder**.**inverse\_transform(residence\_values[:5]))

Before Encoding: ['France', 'Spain', 'France', 'France', 'Spain']

After Encoding: [0 2 0 0 2]

The inverse from the encoding result: ['France' 'Spain' 'France' 'France' 'Spain']

In [172]:

**from** sklearn.preprocessing **import** OneHotEncoder

gender\_encoder **=** OneHotEncoder()

In [174]:

**from** sklearn.preprocessing **import** OneHotEncoder

**import** numpy **as** np

gender\_encoder **=** OneHotEncoder()

gender\_reshaped **=** np**.**array(df\_categorical['Gender'])**.**reshape(**-**1, 1)

gender\_values **=** gender\_encoder**.**fit\_transform(gender\_reshaped)

print(df\_categorical['Gender'][:5])

print()

print(gender\_values**.**toarray()[:5])

print()

print(gender\_encoder**.**inverse\_transform(gender\_values)[:5])

0 Female

1 Female

2 Female

3 Female

4 Female

Name: Gender, dtype: object

[[1. 0.]

[1. 0.]

[1. 0.]

[1. 0.]

[1. 0.]]

[['Female']

['Female']

['Female']

['Female']

['Female']]

In [175]:

smoke\_encoder **=** OneHotEncoder()

smoke\_reshaped **=** np**.**array(df\_categorical['Surname'])**.**reshape(**-**1, 1)

smoke\_values **=** smoke\_encoder**.**fit\_transform(smoke\_reshaped)

print(df\_categorical['Surname'][:5])

print()

print(smoke\_values**.**toarray()[:5])

print()

print(smoke\_encoder**.**inverse\_transform(smoke\_values)[:5])

0 Hargrave

1 Hill

2 Onio

3 Boni

4 Mitchell

Name: Surname, dtype: object

[[0. 0. 0. ... 0. 0. 0.]

[0. 0. 0. ... 0. 0. 0.]

[0. 0. 0. ... 0. 0. 0.]

[0. 0. 0. ... 0. 0. 0.]

[0. 0. 0. ... 0. 0. 0.]]

[['Hargrave']

['Hill']

['Onio']

['Boni']

['Mitchell']]

In [176]:

work\_encoder **=** OneHotEncoder()

work\_reshaped **=** np**.**array(df\_categorical['Geography'])**.**reshape(**-**1, 1)

work\_values **=** work\_encoder**.**fit\_transform(work\_reshaped)

print(df\_categorical['Geography'][:5])

print()

print(work\_values**.**toarray()[:5])

print()

print(work\_encoder**.**inverse\_transform(work\_values)[:5])

0 France

1 Spain

2 France

3 France

4 Spain

Name: Geography, dtype: object

[[1. 0. 0.]

[0. 0. 1.]

[1. 0. 0.]

[1. 0. 0.]

[0. 0. 1.]]

[['France']

['Spain']

['France']

['France']

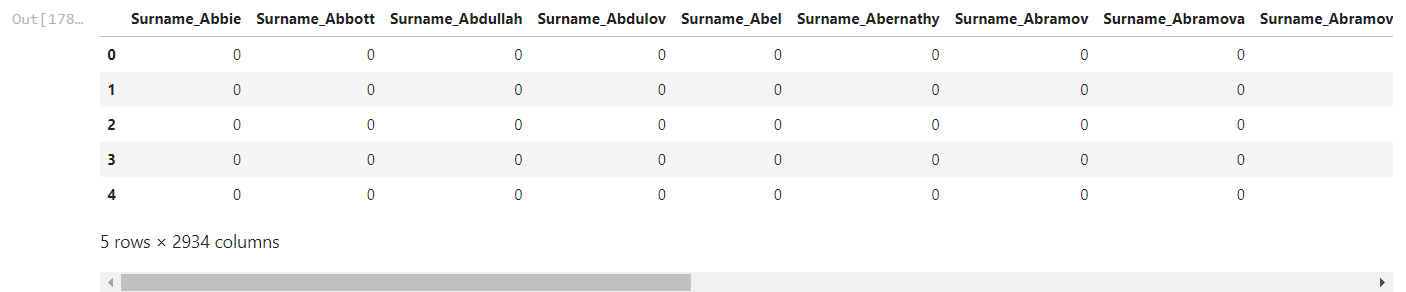
['Spain']]

In [178]:

df\_categorical\_encoded **=** pd**.**get\_dummies(df\_categorical, drop\_first**=True**)

df\_categorical\_encoded**.**head()

Out[178]:

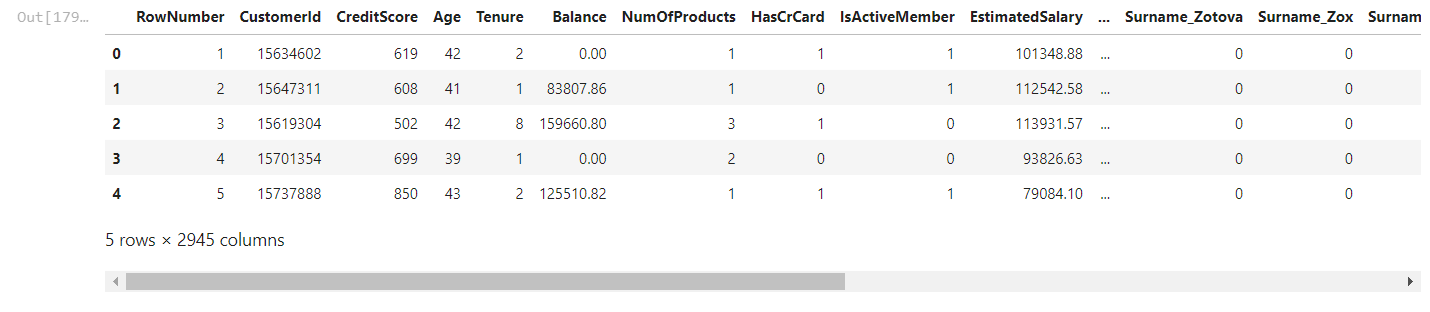


In [179]:

df\_new **=** pd**.**concat([df\_numeric, df\_categorical\_encoded], axis**=**1)

df\_new**.**head()

Out[179]:



**Split the data into dependent and independent variables.**

In [180]:

df**=**pd**.**read\_csv('/content/Churn\_Modelling.csv')

In [182]:

print(df["Balance"]**.**min())

print(df["Balance"]**.**max())

print(df["Balance"]**.**mean())

0.0

250898.09

76485.889288

In [183]:

print(df**.**count(0))

RowNumber 10000

CustomerId 10000

Surname 10000

CreditScore 10000

Geography 10000

Gender 10000

Age 10000

Tenure 10000

Balance 10000

NumOfProducts 10000

HasCrCard 10000

IsActiveMember 10000

EstimatedSalary 10000

Exited 10000

dtype: int64

In [184]:

print(df**.**shape)

(10000, 14)

In [185]:

print(df**.**size)

140000

In [187]:

X **=** df**.**iloc[:, :**-**1]**.**values

print(X)

[[1 15634602 'Hargrave' ... 1 1 101348.88]

[2 15647311 'Hill' ... 0 1 112542.58]

[3 15619304 'Onio' ... 1 0 113931.57]

...

[9998 15584532 'Liu' ... 0 1 42085.58]

[9999 15682355 'Sabbatini' ... 1 0 92888.52]

[10000 15628319 'Walker' ... 1 0 38190.78]]

In [271]:

Y **=** df**.**iloc[:, **-**1]**.**values

print(Y)

[1 0 1 ... 1 1 0]

**Scale the independent variables**

In [215]:

df **=** pd**.**read\_csv('/content/Churn\_Modelling.csv')

x **=** df[['Age', 'Tenure']]**.**values

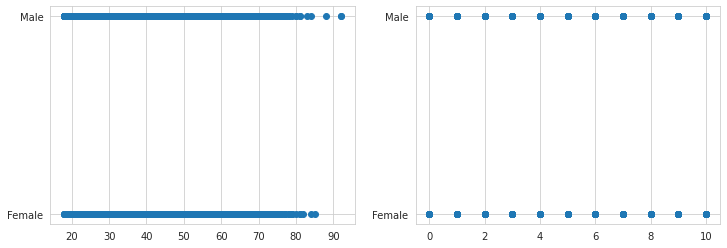
y **=** df['Gender']**.**values

fig, ax **=** plt**.**subplots(ncols**=**2, figsize**=**(12, 4))

ax[0]**.**scatter(x[:,0], y)

ax[1]**.**scatter(x[:,1], y)

plt**.**show()



In [216]:

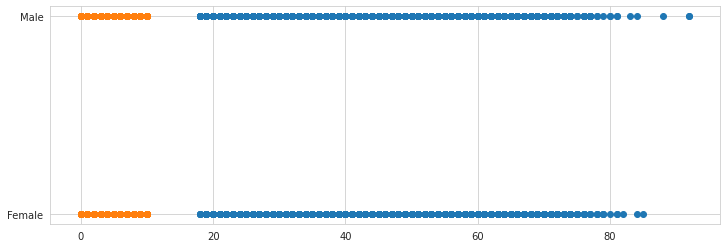
fig, ax **=** plt**.**subplots(figsize**=**(12, 4))

ax**.**scatter(x[:,0], y)

ax**.**scatter(x[:,1], y)

Out[216]:

<matplotlib.collections.PathCollection at 0x7f9a8a854ad0>



In [217]:

fig, ax **=** plt**.**subplots(figsize**=**(12, 4))

ax**.**hist(x[:,0])

ax**.**hist(x[:,1])

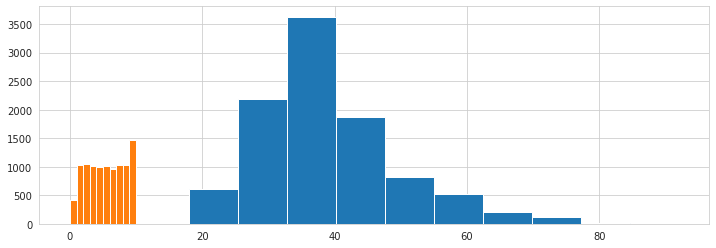
Out[217]:

(array([ 413., 1035., 1048., 1009., 989., 1012., 967., 1028., 1025.,

1474.]),

array([ 0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.]),

<a list of 10 Patch objects>)



In [220]:

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.preprocessing **import** MinMaxScaler

fig, ax **=** plt**.**subplots(figsize**=**(12, 4))

scaler **=** StandardScaler()

x\_std **=** scaler**.**fit\_transform(x)

ax**.**hist(x\_std[:,0])

ax**.**hist(x\_std[:,1])

Out[220]:

(array([ 413., 1035., 1048., 1009., 2001., 0., 1995., 0., 1025.,

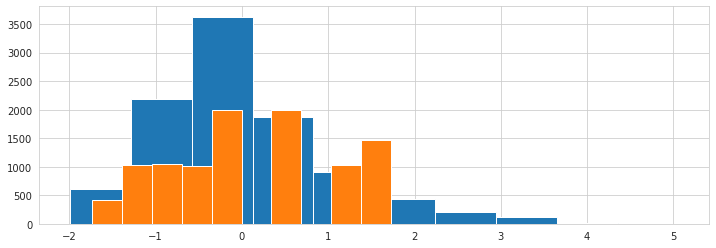
1474.]),

array([-1.73331549, -1.38753759, -1.04175968, -0.69598177, -0.35020386,

-0.00442596, 0.34135195, 0.68712986, 1.03290776, 1.37868567,

1.72446358]),

<a list of 10 Patch objects>)



In [219]:

fig, ax **=** plt**.**subplots(figsize**=**(12, 4))

scaler **=** StandardScaler()

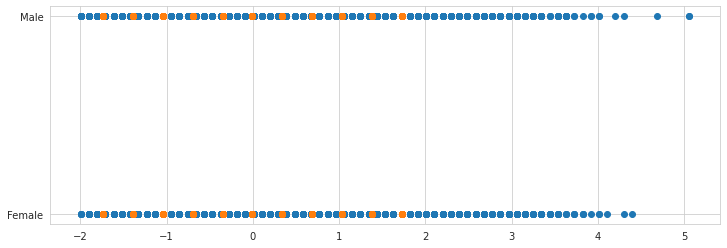
x\_std **=** scaler**.**fit\_transform(x)

ax**.**scatter(x\_std[:,0], y)

ax**.**scatter(x\_std[:,1], y)

Out[219]:

<matplotlib.collections.PathCollection at 0x7f9a8a2fde50>



In [221]:

fig, ax **=** plt**.**subplots(figsize**=**(12, 4))

scaler **=** MinMaxScaler()

x\_minmax **=** scaler**.**fit\_transform(x)

ax**.**hist(x\_minmax [:,0])

ax**.**hist(x\_minmax [:,1])

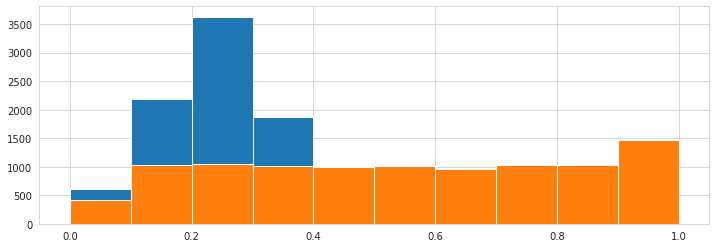
Out[221]:

(array([ 413., 1035., 1048., 1009., 989., 1012., 967., 1028., 1025.,

1474.]),

array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),

<a list of 10 Patch objects>)



In [222]:

fig, ax **=** plt**.**subplots(figsize**=**(12, 4))

scaler **=** MinMaxScaler()

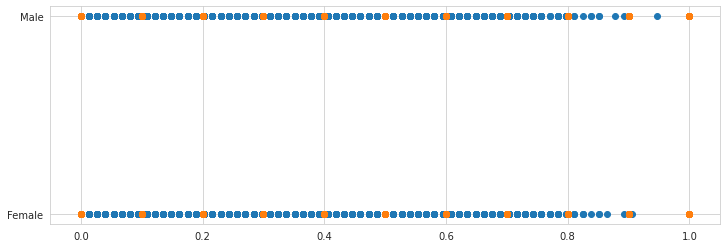
x\_minmax **=** scaler**.**fit\_transform(x)

ax**.**scatter(x\_minmax [:,0], y)

ax**.**scatter(x\_minmax [:,1], y)

Out[222]:

<matplotlib.collections.PathCollection at 0x7f9a8a0cae10>



In [223]:

fig, ax **=** plt**.**subplots(figsize**=**(12, 4))

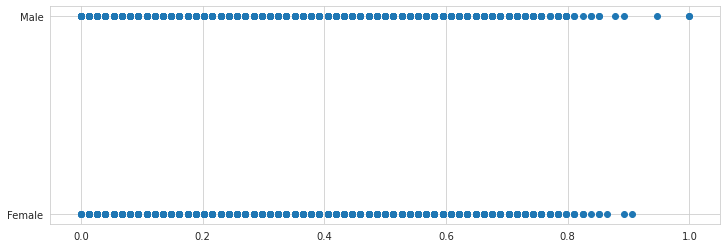
scaler **=** MinMaxScaler()

x\_minmax **=** scaler**.**fit\_transform(x)

ax**.**scatter(x\_minmax [:,0], y)

Out[223]:

<matplotlib.collections.PathCollection at 0x7f9a8a0caf10>



In [224]:

fig, ax **=** plt**.**subplots(figsize**=**(12, 4))

scaler **=** MinMaxScaler()

x\_minmax **=** scaler**.**fit\_transform(x)

ax**.**hist(x\_minmax [:,0])

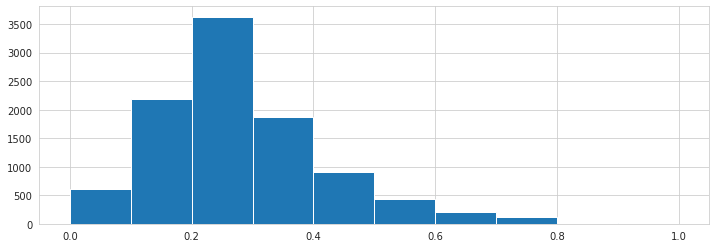
Out[224]:

(array([ 611., 2179., 3629., 1871., 910., 441., 208., 127., 20.,

4.]),

array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),

<a list of 10 Patch objects>)



In [227]:

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.pipeline **import** Pipeline

**from** sklearn.linear\_model **import** SGDRegressor

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.preprocessing **import** MinMaxScaler

**from** sklearn.metrics **import** mean\_absolute\_error

**import** sklearn.metrics **as** metrics

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

*# Import Data*

df **=** pd**.**read\_csv('/content/Churn\_Modelling.csv')

x **=** df[['Age', 'Tenure']]**.**values

y **=** df['Balance']**.**values

*# Split into a training and testing set*

X\_train, X\_test, Y\_train, Y\_test **=** train\_test\_split(x, y)

*# Define the pipeline for scaling and model fitting*

pipeline **=** Pipeline([

("MinMax Scaling", MinMaxScaler()),

("SGD Regression", SGDRegressor())

])

*# Scale the data and fit the model*

pipeline**.**fit(X\_train, Y\_train)

*# Evaluate the model*

Y\_pred **=** pipeline**.**predict(X\_test)

print('Mean Absolute Error: ', mean\_absolute\_error(Y\_pred, Y\_test))

print('Score', pipeline**.**score(X\_test, Y\_test))

Mean Absolute Error: 57120.533393590835

Score 0.0004207814312172653

**Split the data into training and testing**

In [267]:

dataset **=** pd**.**read\_csv('/content/Churn\_Modelling.csv')

print(dataset)

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]

In [287]:

dataset**.**drop(["HasCrCard"],axis**=**1,inplace**=True**)

In [288]:

print(dataset**.**shape)*#no. of rows and colume*

print(dataset**.**head(10))

(10000, 7)

CustomerId CreditScore Age Tenure Balance IsActiveMember \

0 15634602 619 42 2 0.00 1

1 15647311 608 41 1 83807.86 1

2 15619304 502 42 8 159660.80 0

3 15701354 699 39 1 0.00 0

4 15737888 850 43 2 125510.82 1

5 15574012 645 44 8 113755.78 0

6 15592531 822 50 7 0.00 1

7 15656148 376 29 4 115046.74 0

8 15792365 501 44 4 142051.07 1

9 15592389 684 27 2 134603.88 1

EstimatedSalary

0 101348.88

1 112542.58

2 113931.57

3 93826.63

4 79084.10

5 149756.71

6 10062.80

7 119346.88

8 74940.50

9 71725.73

In [289]:

X**=**dataset**.**iloc[:,:**-**1]**.**values

X

Out[289]:

array([[1.5634602e+07, 6.1900000e+02, 4.2000000e+01, 2.0000000e+00,

0.0000000e+00, 1.0000000e+00],

[1.5647311e+07, 6.0800000e+02, 4.1000000e+01, 1.0000000e+00,

8.3807860e+04, 1.0000000e+00],

[1.5619304e+07, 5.0200000e+02, 4.2000000e+01, 8.0000000e+00,

1.5966080e+05, 0.0000000e+00],

...,

[1.5584532e+07, 7.0900000e+02, 3.6000000e+01, 7.0000000e+00,

0.0000000e+00, 1.0000000e+00],

[1.5682355e+07, 7.7200000e+02, 4.2000000e+01, 3.0000000e+00,

7.5075310e+04, 0.0000000e+00],

[1.5628319e+07, 7.9200000e+02, 2.8000000e+01, 4.0000000e+00,

1.3014279e+05, 0.0000000e+00]])

In [290]:

Y**=**dataset**.**iloc[:,**-**1]**.**values

Y

Out[290]:

array([101348.88, 112542.58, 113931.57, ..., 42085.58, 92888.52,

38190.78])

In [291]:

**from** sklearn.model\_selection **import** train\_test\_split

X\_train,X\_test,Y\_train,Y\_test **=** train\_test\_split( X, Y, test\_size **=** 0.25, random\_state **=** 0 )

In [306]:

**from** sklearn.preprocessing **import** StandardScaler

sc**=**StandardScaler()

X\_train **=** sc**.**fit\_transform(X\_train)

X\_test **=** sc**.**transform(X\_test)

print(X\_train)

[[-1.34333028 -0.73550706 0.01526571 0.00886037 0.67316003 -1.03446007]

[ 1.55832963 1.02442719 -0.65260917 0.00886037 -1.20772417 -1.03446007]

[-0.65515619 0.80829492 -0.46178778 1.39329338 -0.35693706 0.96668786]

...

[-1.63542994 0.90092304 -0.36637708 0.00886037 1.36657199 -1.03446007]

[-0.38540456 -0.62229491 -0.08014499 1.39329338 -1.20772417 0.96668786]

[-1.37829524 -0.28265848 0.87396199 -1.37557264 0.51741687 -1.03446007]]

In [305]:

print(X\_test)

[[-1.05852196 -0.55025082 -0.36637708 1.04718513 0.88494297 0.96668786]

[-0.51554728 -1.31185979 0.11067641 -1.02946438 0.43586703 -1.03446007]

[-0.8058485 0.57157862 0.3014978 1.04718513 0.31486378 0.96668786]

...

[ 0.25326371 1.95070838 0.01526571 -1.37557264 0.30819395 -1.03446007]

[-0.17836122 0.29369426 -0.08014499 0.70107688 0.55698791 -1.03446007]

[ 0.40190663 0.870047 -0.74801987 -0.68335613 0.7006957 -1.03446007]]