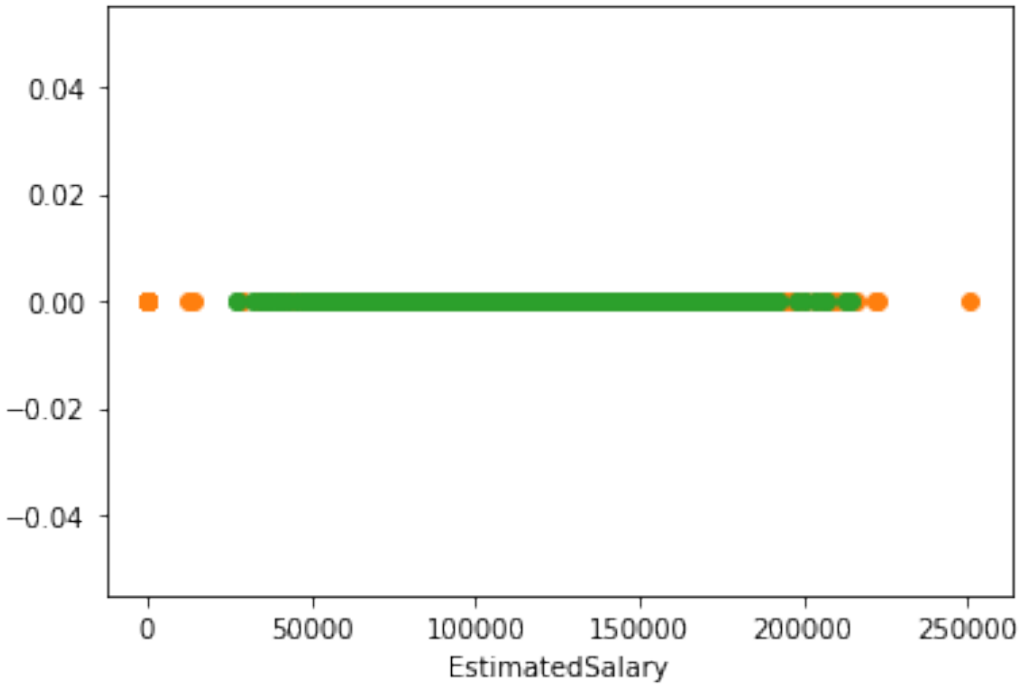
'Univariant

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('/content/Churn_Modelling.csv')

df.head()
```

```
   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
```

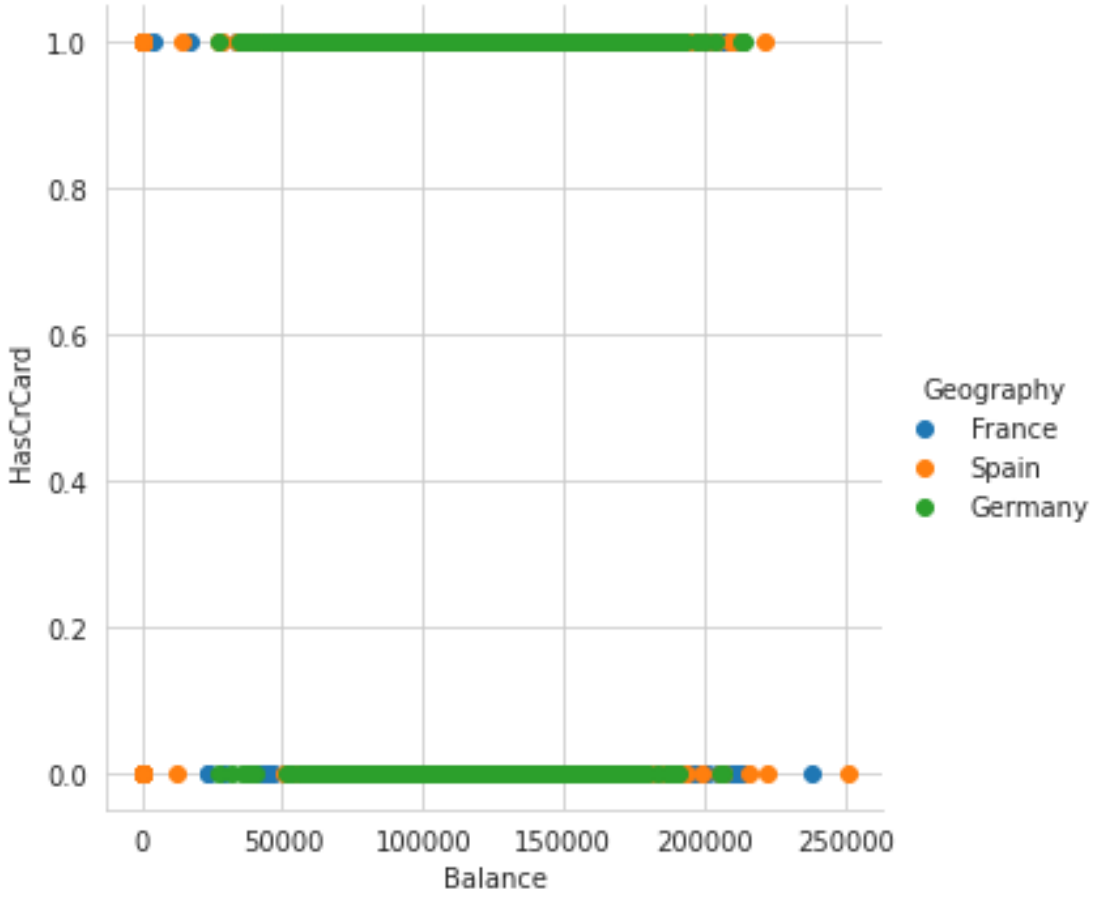
```
df_France  =df.loc[df['Geography']=='France ']
df_Spain=df.loc[df['Geography']=='Spain']
df_Germany=df.loc[df['Geography']=='Germany']
plt.plot(df_France    ['Balance'],np.zeros_like(df_France
     ['Balance']),'o')
plt.plot(df_Spain['Balance'],np.zeros_like(df_Spain['Balance']),'o')
plt.plot(df_Germany['Balance'],np.zeros_like(df_Germany['Balance']),'o
')
plt.xlabel('EstimatedSalary')
plt.show()
```



Bivariate

```
sns.FacetGrid(df,hue="Geography",size=5).map(plt.scatter,"Balance","HasCrCard").add_legend()
plt.show()
```

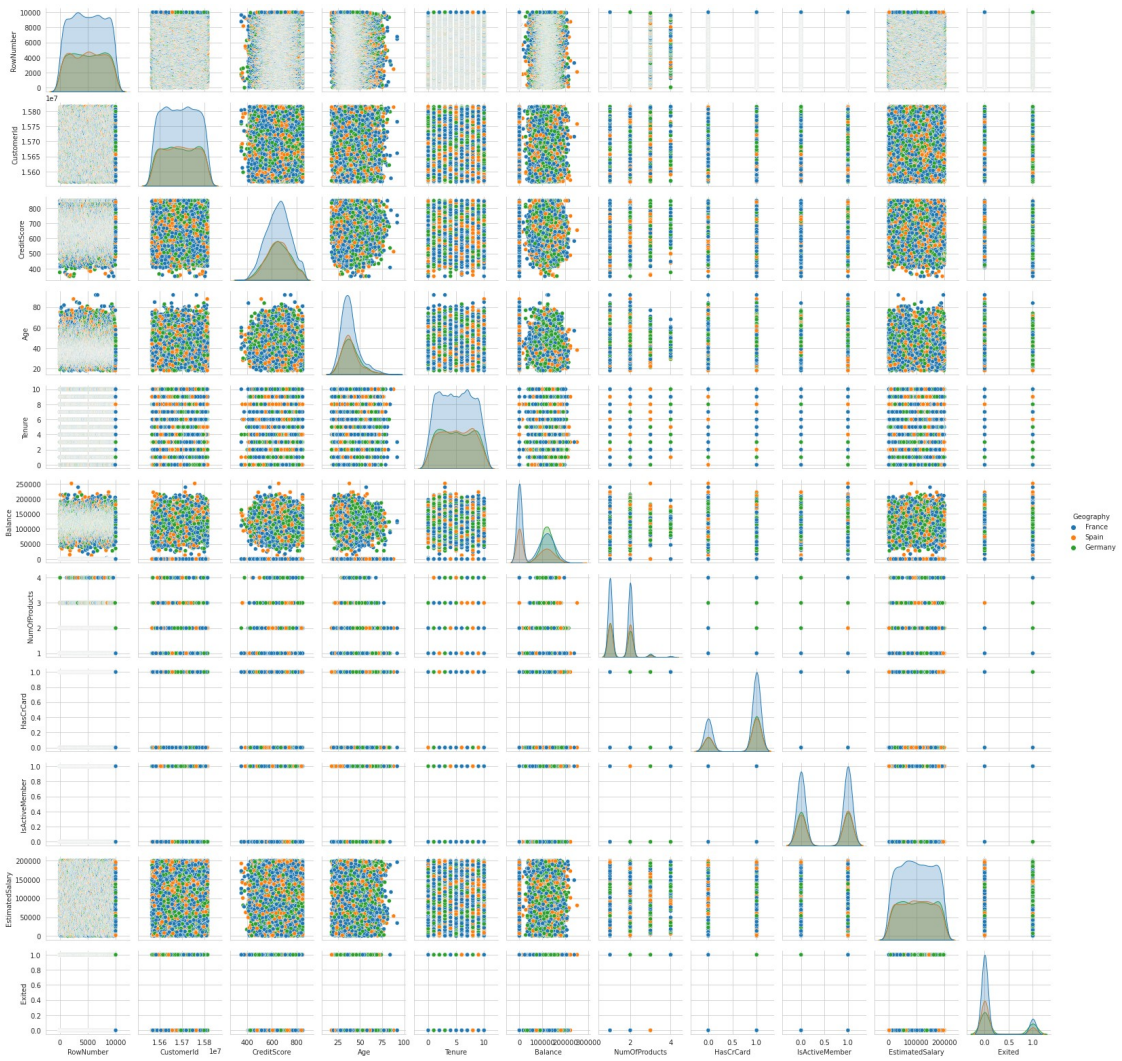
```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:337:
UserWarning: The `size` parameter has been renamed to `height`; please
update your code.
  warnings.warn(msg, UserWarning)
```



Multi - Variate Analysis

```
sns.pairplot(df, hue="Geography", height=2)
```

```
<seaborn.axisgrid.PairGrid at 0x7f2380379110>
```



descriptive statistics

```
df.head()
```

```
   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
```

```
df.mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.
  """Entry point for launching an IPython kernel.

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
```

```
df.mean(axis=1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.
  """Entry point for launching an IPython kernel.

0       1.430602e+06
1       1.440392e+06
2       1.444860e+06
3       1.435993e+06
4       1.449399e+06
           ...
9995    1.428483e+06
```

```
9996    1.430866e+06
9997    1.421579e+06
9998    1.441922e+06
9999    1.437044e+06
Length: 10000, dtype: float64
```

```
df.median()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.
  """Entry point for launching an IPython kernel.
```

```
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
```

```
standard_deviation = df['Balance'].std()
print(standard_deviation)
```

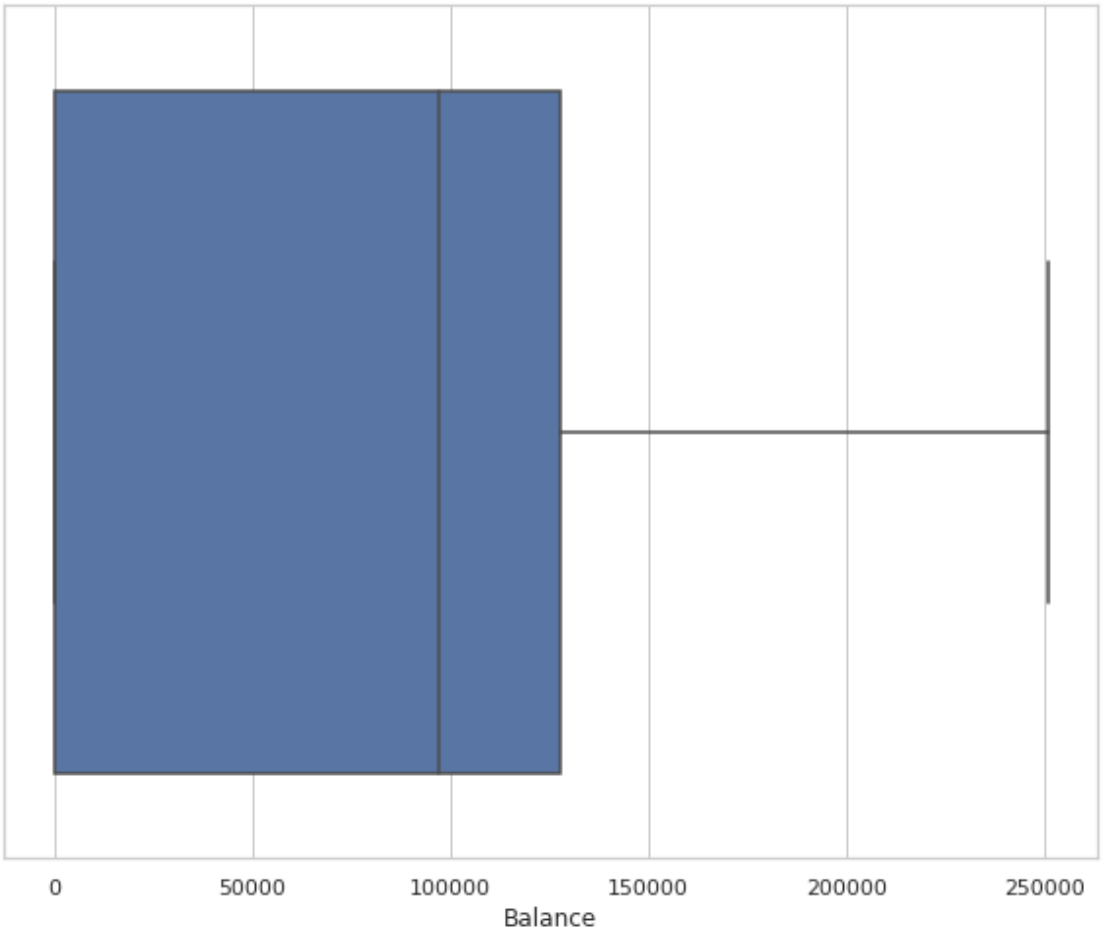
```
62397.405202385955
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
%matplotlib inline
```

```
sns.set(style="whitegrid")
plt.figure(figsize=(10,8))
ax = sns.boxplot(x='Balance', data=df, orient="v")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_core.py:1326:
UserWarning: Vertical orientation ignored with only `x` specified.
  warnings.warn(single_var_warning.format("Vertical", "x"))
```

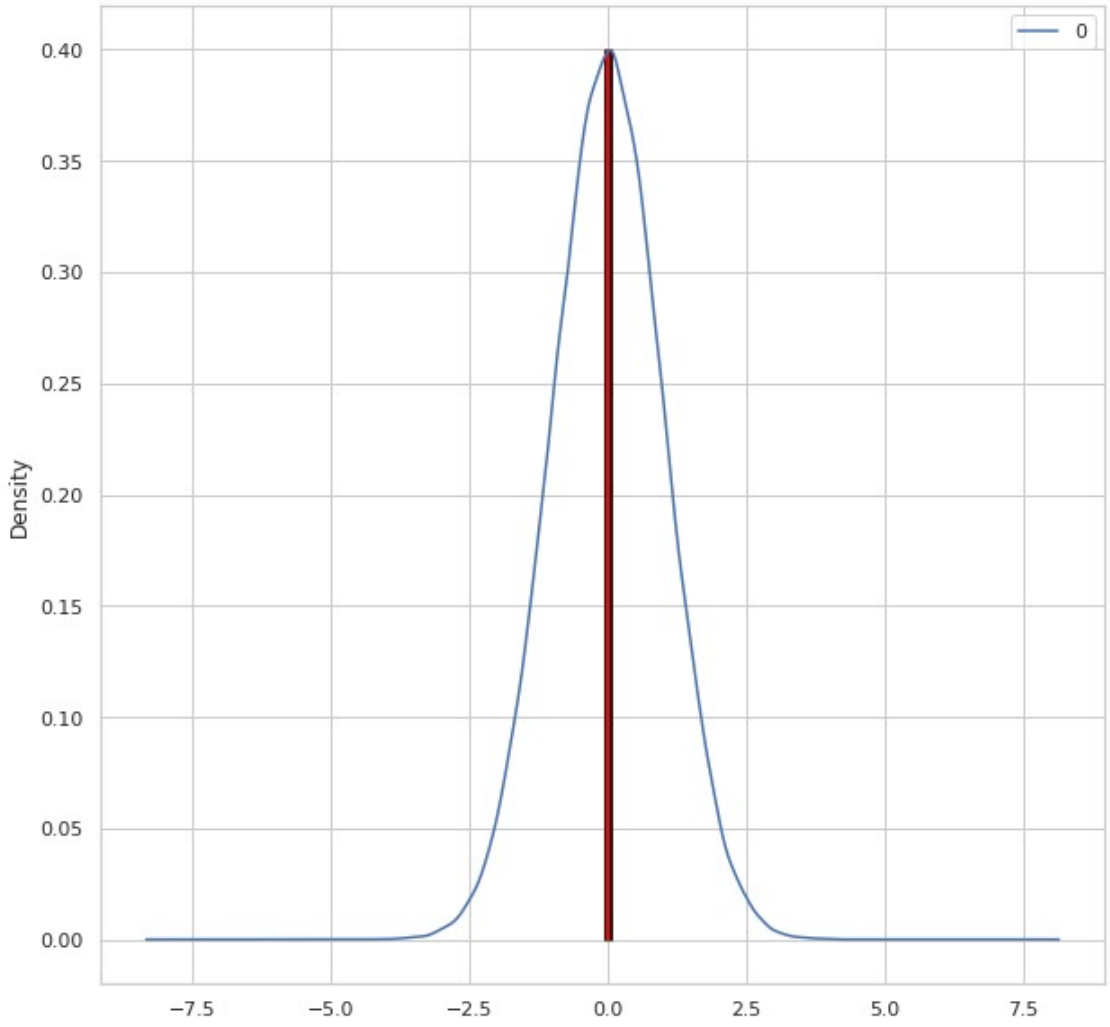


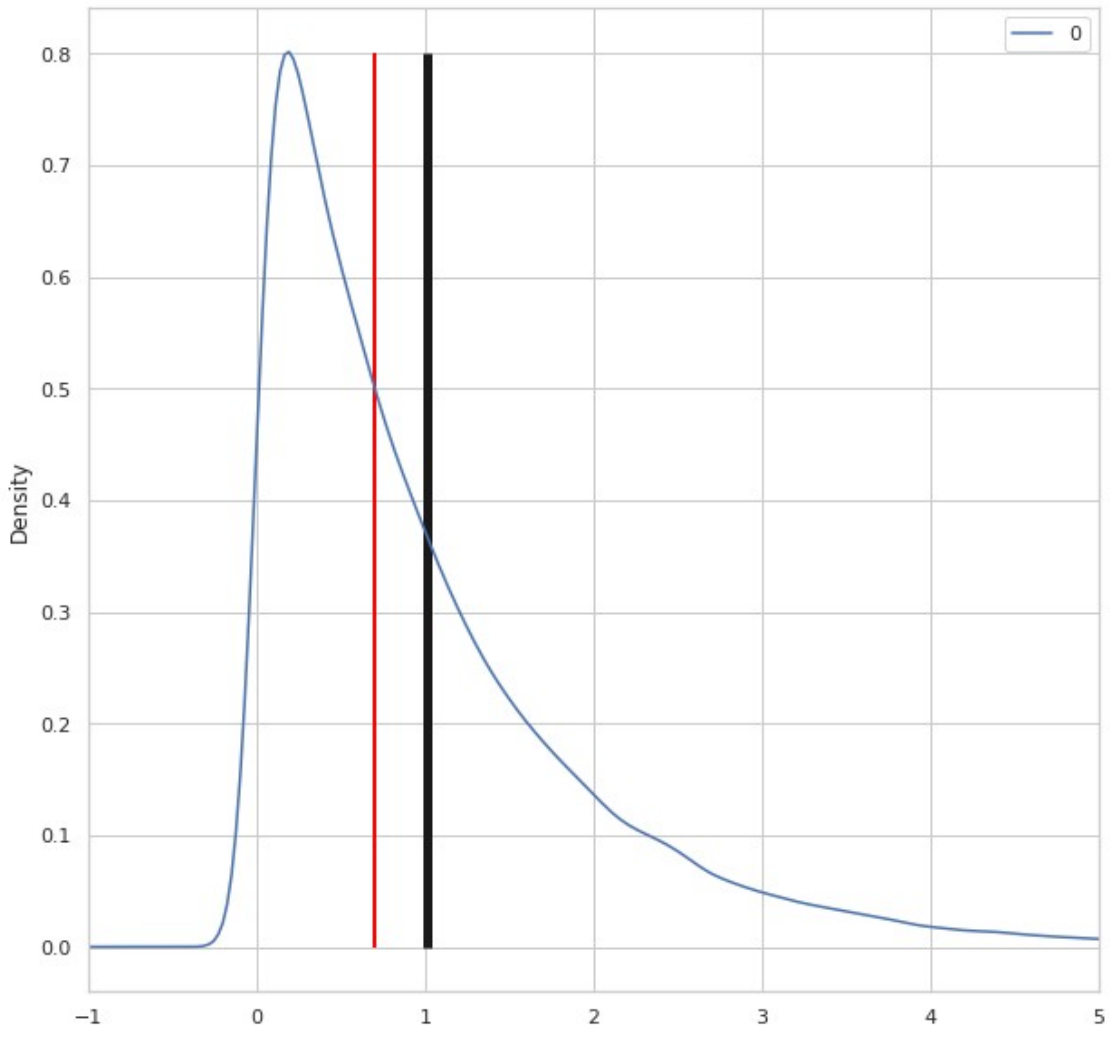

```
average = df['Balance'].mean()
print(average)

med = df['Balance'].median()
print(med)
```

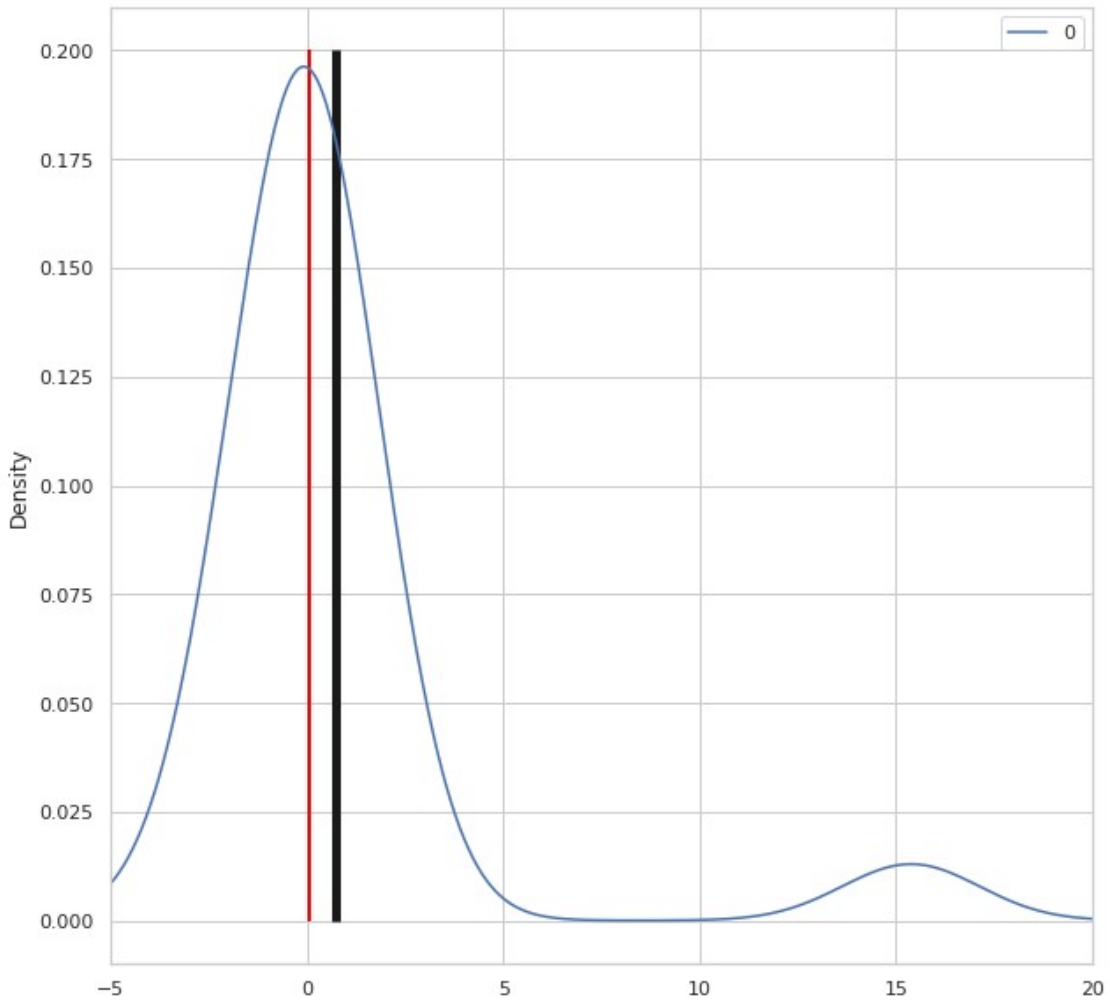
```
76485.889288
97198.54000000001
```

```
norm_data = pd.DataFrame(np.random.normal(size=100000))
norm_data.plot(kind="density",
 figsize=(10,10));
plt.vlines(norm_data.mean(),
 ymin=0,
 ymax=0.4,
 linewidth=5.0);
plt.vlines(norm_data.median(),
 ymin=0,
 ymax=0.4,
 linewidth=2.0,
 color="red");
```

```
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(),
 ymin=0,
 ymax=0.8,
 linewidth=5.0);
plt.vlines(skewed_data.median(),
 ymin=0,
 ymax=0.8,
 linewidth=2.0,
 color="red");

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(),
 ymin=0,
 ymax=0.2,
 linewidth=5.0);
plt.vlines(combined_data.median(),
 ymin=0,
 ymax=0.2,
 linewidth=2.0,
 color="red");
```


0
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5


0
Density
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−1
0
1
2
3
4
5




```
df.mode()
```

	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	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...
9995	9996	15815628	NaN	NaN	NaN	NaN	NaN
9996	9997	15815645	NaN	NaN	NaN	NaN	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            NaN        NaN             NaN
...      ...      ...            ...        ...             ...
9995     NaN      NaN            NaN        NaN             NaN
9996     NaN      NaN            NaN        NaN             NaN
9997     NaN      NaN            NaN        NaN             NaN
9998     NaN      NaN            NaN        NaN             NaN
9999     NaN      NaN            NaN        NaN             NaN

      EstimatedSalary  Exited
0            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]
```

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

```
74
```

```
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
```

```
[18.0, 32.0, 37.0, 44.0, 92.0]
```

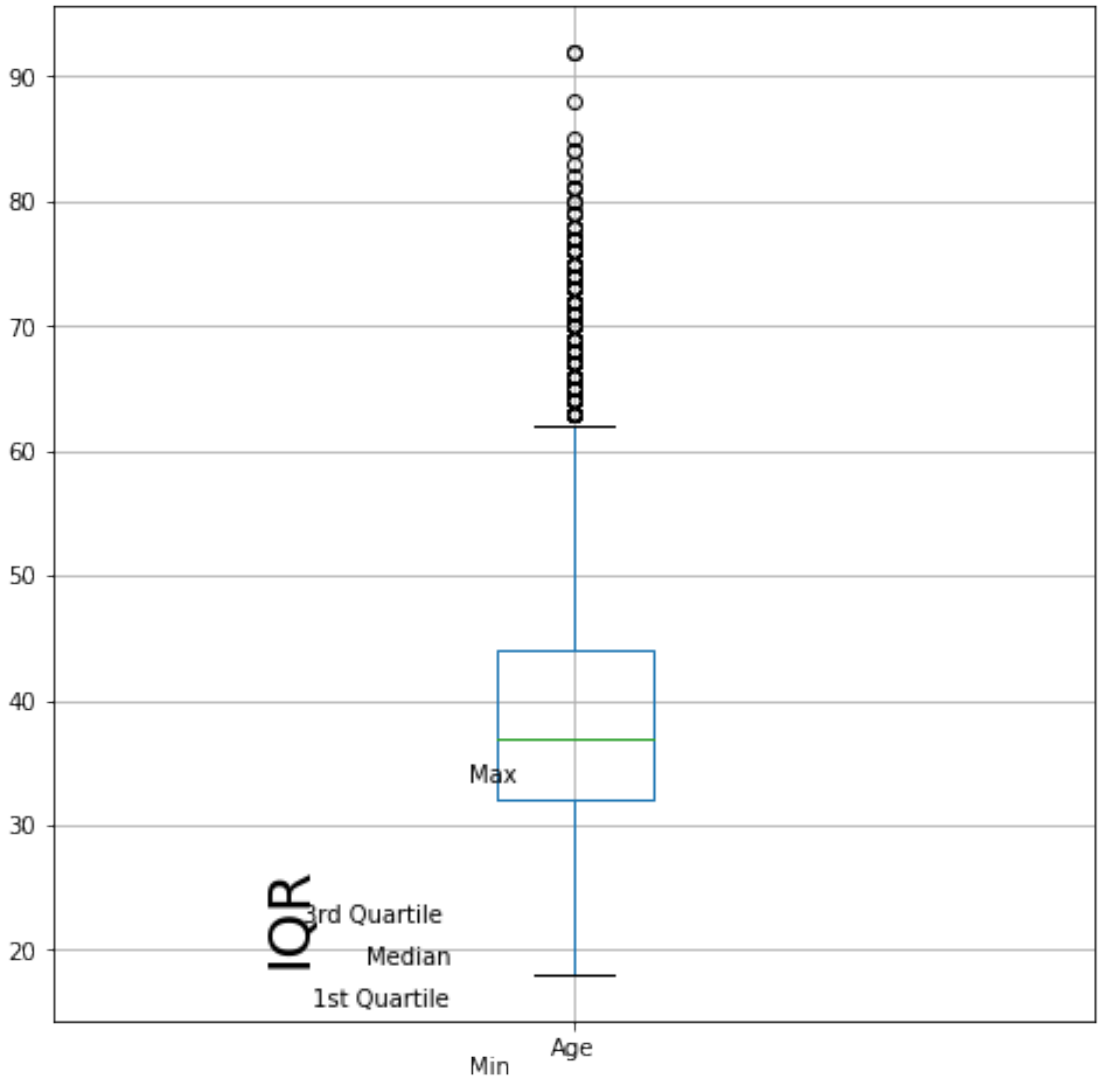
```
df["Age"].describe()
```

```
count    10000.000000
mean        38.921800
std         10.487806
min         18.000000
25%         32.000000
50%         37.000000
75%         44.000000
max         92.000000
Name: Age, dtype: float64
```

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

```
12.0
```

```
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);
```




```
df["Age"].var()
```

```
109.99408416841683
```

```
df["Age"].std()
```

```
10.487806451704609
```

```
abs_median_devs = abs(df["Age"] - df["Age"].median())
abs_median_devs.median() * 1.4826
```

```
8.8956
```

Skewness and Kurtosis

```
df["Age"].skew()
```

```
1.0113202630234552
```

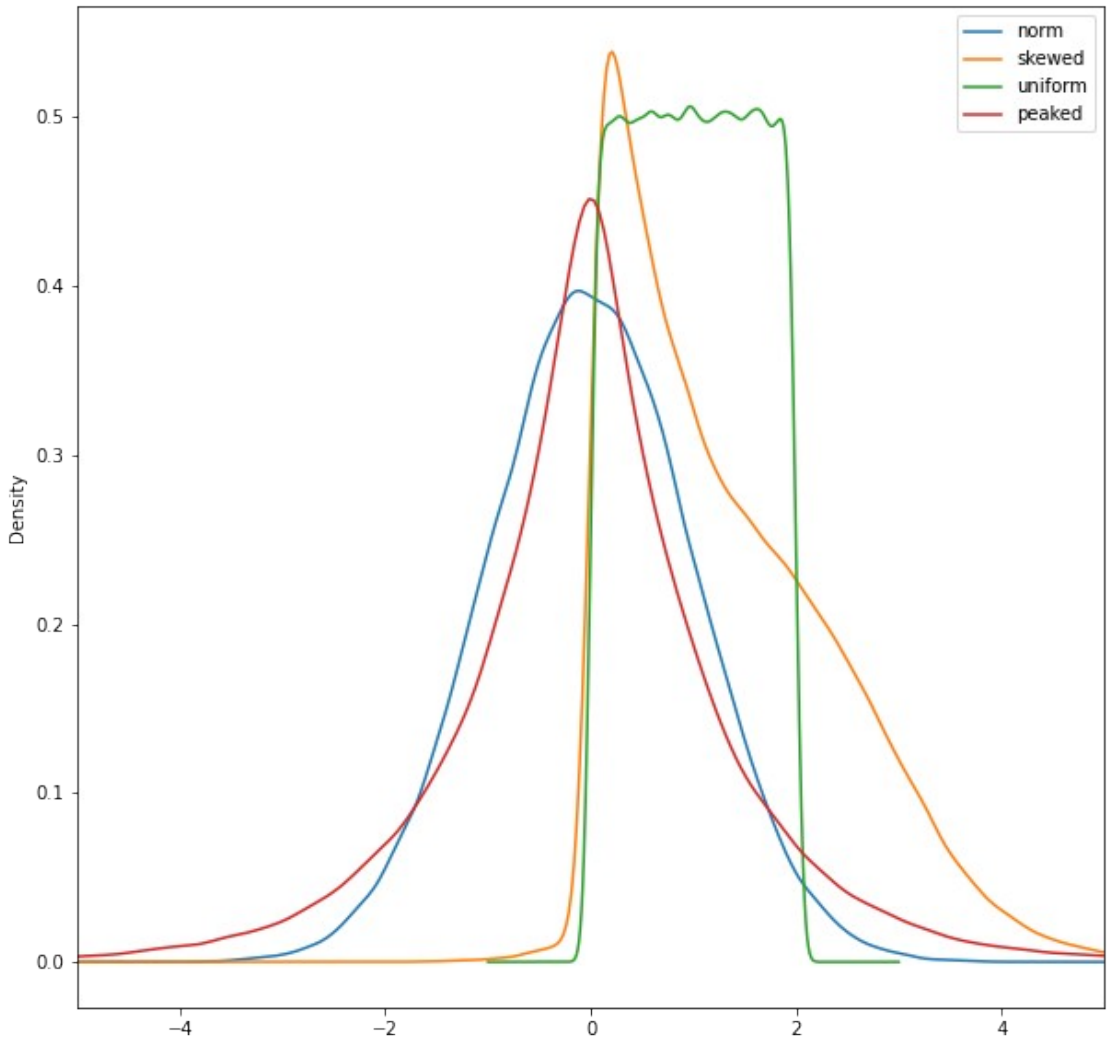
```
df["Age"].kurt()
```

```
1.3953470615086956
```

```
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})

data_df.plot(kind="density",
 figsize=(10,10),
 xlim=(-5,5));
```

```
data_df.skew()
```

```
norm       0.009042
skewed     1.003862
uniform   -0.001527
peaked     0.016763
dtype: float64
```




```
data_df.skew()
```

```
norm       0.009042
skewed     1.003862
uniform   -0.001527
peaked     0.016763
dtype: float64
```

```
data_df.kurt()
```

```
norm      -0.007351
skewed     1.308347
uniform   -1.198145
peaked     2.938200
dtype: float64
```

Handle the Missing values

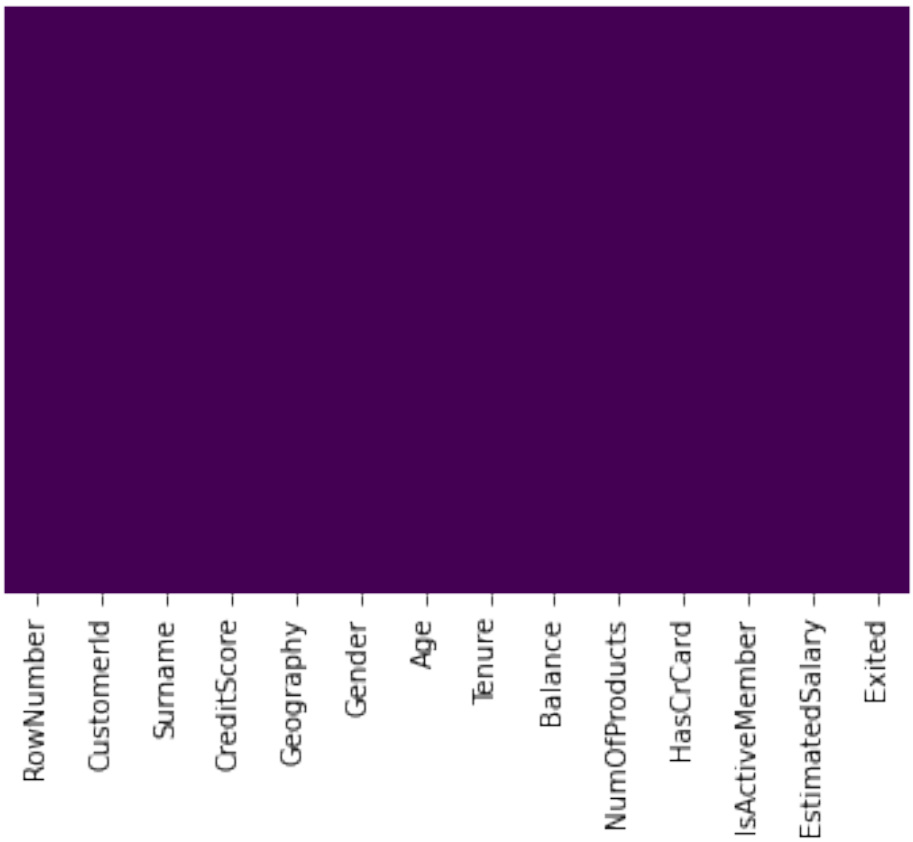
```
df=pd.read_csv('/content/Churn_Modelling.csv')
df.head()
```

```
   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

df.isnull()

      RowNumber  CustomerId  Surname  CreditScore  Geography  Gender
Age  \
0         False       False    False        False      False   False
False
1         False       False    False        False      False   False
False
2         False       False    False        False      False   False
False
3         False       False    False        False      False   False
False
4         False       False    False        False      False   False
False
...         ...         ...      ...          ...        ...     ...
...
9995      False       False    False        False      False   False
False
9996      False       False    False        False      False   False
False
9997      False       False    False        False      False   False
False
9998      False       False    False        False      False   False
```

```
False
9999      False       False    False        False      False   False
False

      Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0      False    False          False      False           False
1      False    False          False      False           False
2      False    False          False      False           False
3      False    False          False      False           False
4      False    False          False      False           False
...      ...      ...            ...        ...             ...
9995   False    False          False      False           False
9996   False    False          False      False           False
9997   False    False          False      False           False
9998   False    False          False      False           False
9999   False    False          False      False           False

      EstimatedSalary  Exited
0               False   False
1               False   False
2               False   False
3               False   False
4               False   False
...               ...     ...
9995            False   False
9996            False   False
9997            False   False
9998            False   False
9999            False   False

[10000 rows x 14 columns]
```

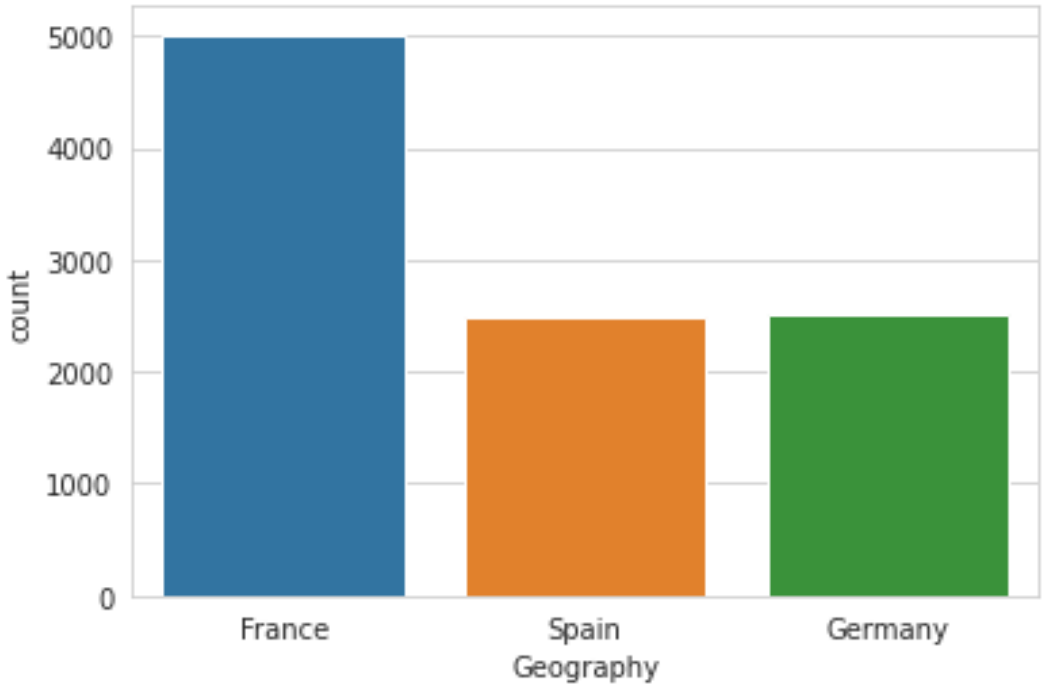
```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2383831dd0>
```



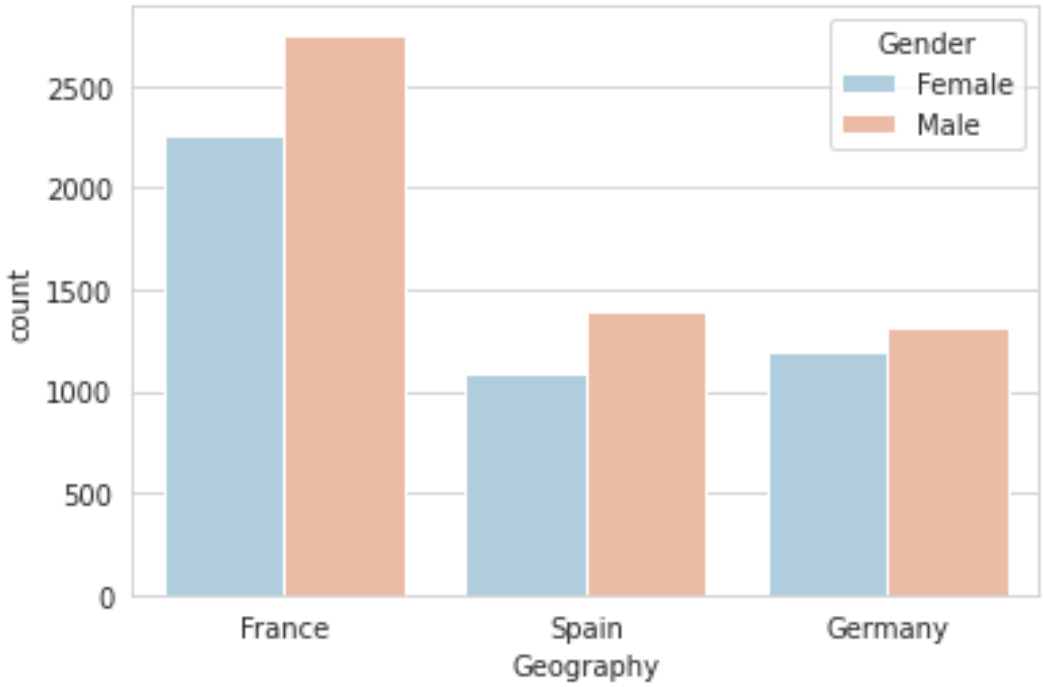

```
sns.set_style('whitegrid')
sns.countplot(x='Geography',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2380761710>
```



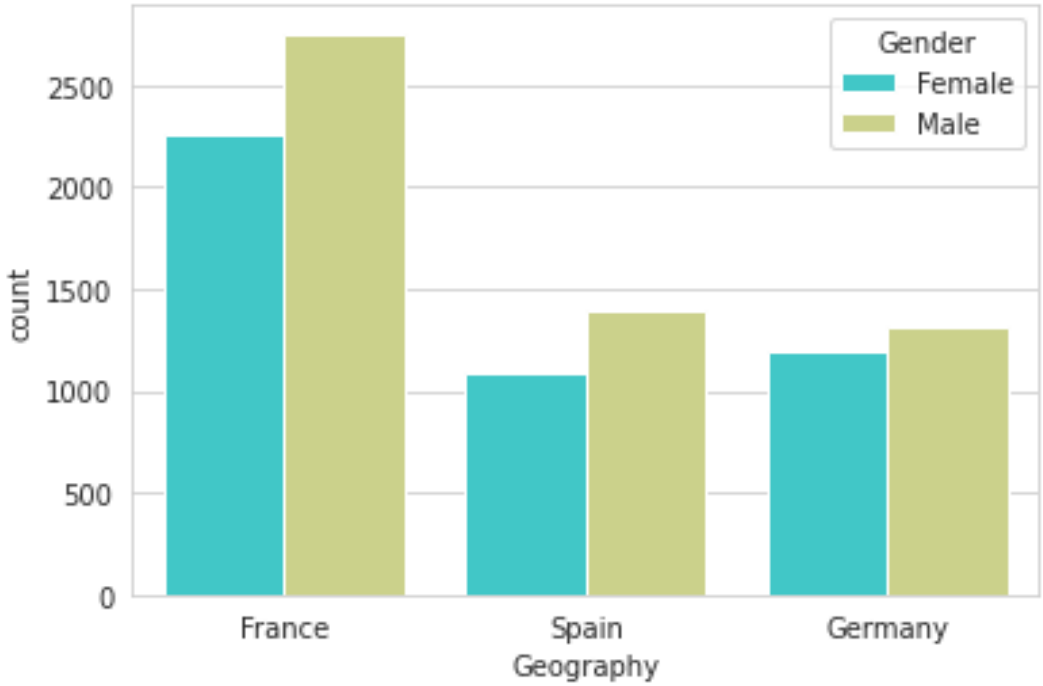

```
sns.set_style('whitegrid')
sns.countplot(x='Geography',hue='Gender',data=df,palette='RdBu_r')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f238073d050>
```



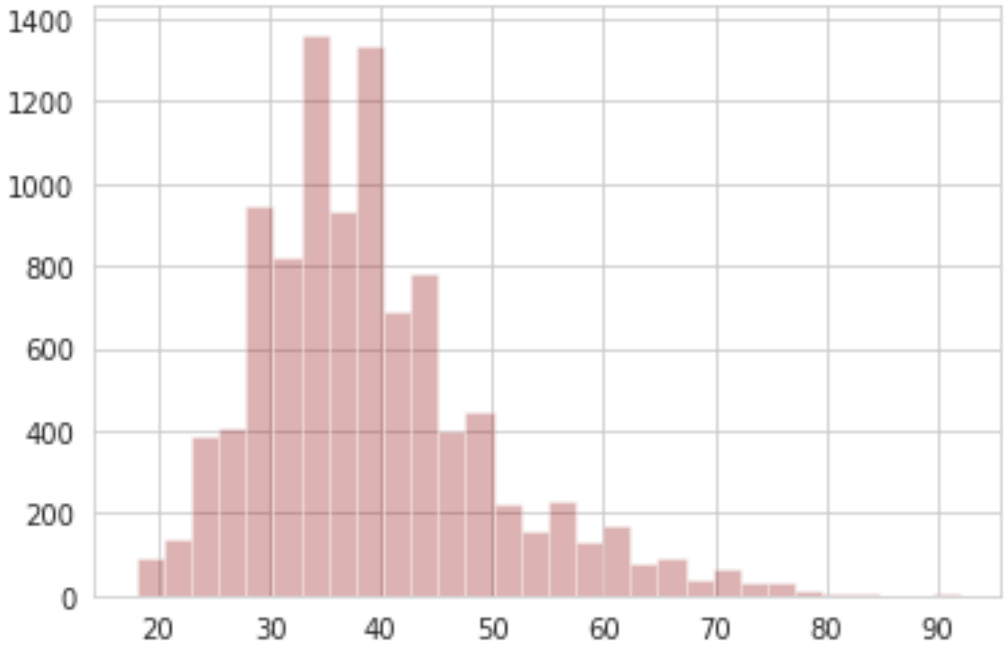

```
sns.set_style('whitegrid')
sns.countplot(x='Geography',hue='Gender',data=df,palette='rainbow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f238069d510>
```



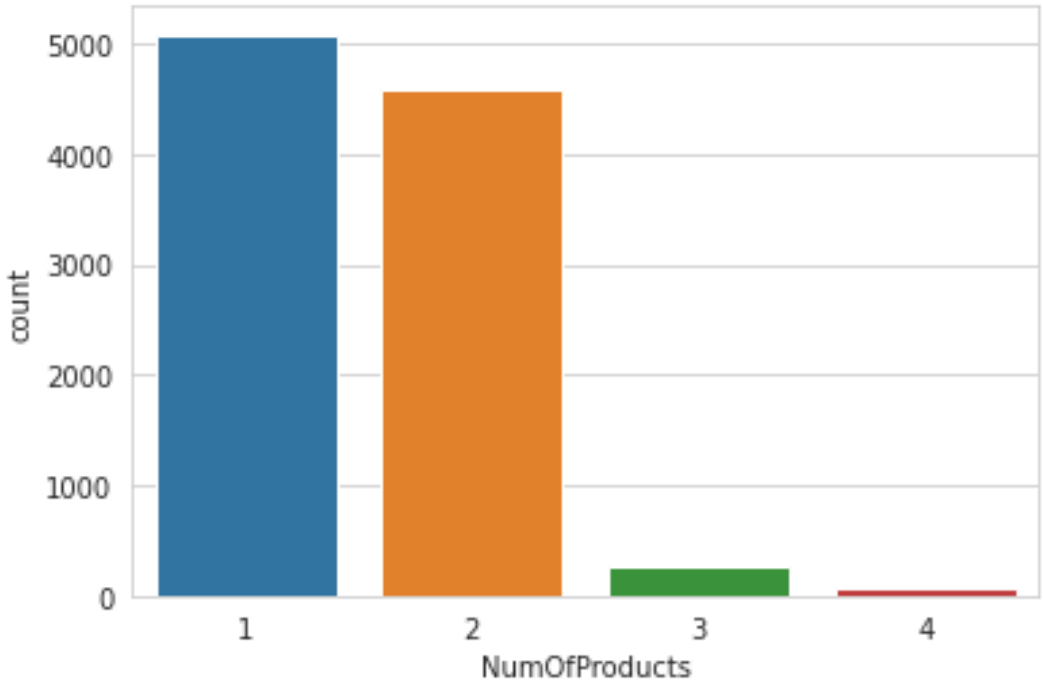

```
df['Age'].hist(bins=30,color='darkred',alpha=0.3)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2380738190>
```



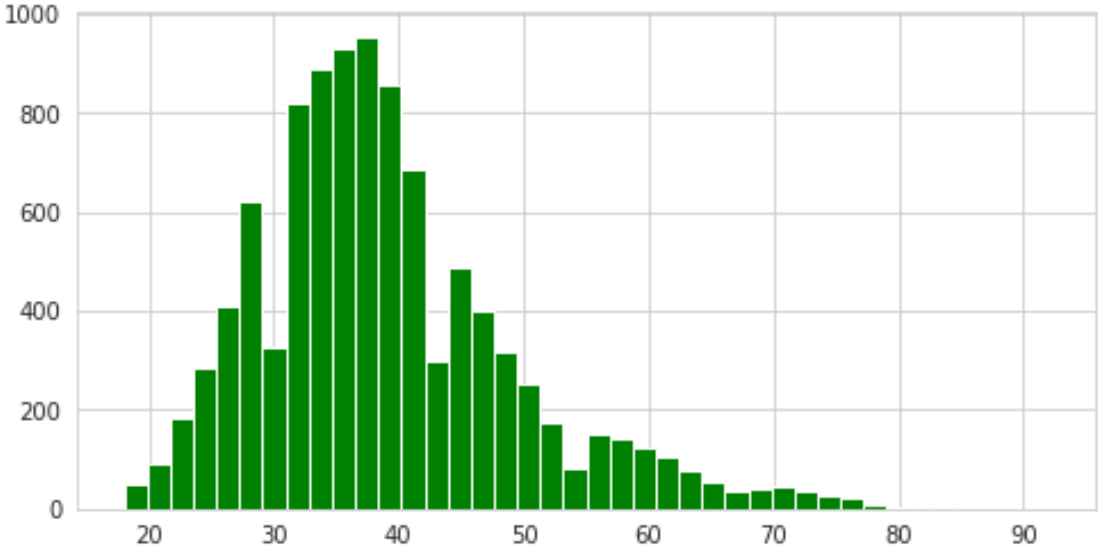

```
sns.countplot(x='NumOfProducts',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f23806193d0>
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2380456650>
```



Find the outliers and replace the outliers

```
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

```
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(z_score)
  return outliers
outlier_pt=detect_outliers(dataset)

outlier_pt
```

```
[3.064712815114584, 3.254305674856025, 3.292224246804313]
```

```
sorted(dataset)
```

```
[10,
 10,
 10,
 10,
 10,
 11,
 11,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 13,
 13,
 13,
 13,
 14,
 14,
 14,
 14,
 14,
 14,
 15,
 15,
 15,
 15,
 15,
```

```
 17,
 19,
 102,
 107,
 108]

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

print(quantile1,quantile3)

12.0 15.0

iqr_value=quantile3-quantile1
print(iqr_value)

3.0

lower_bound_val = quantile1 -(1.5 * iqr_value)
upper_bound_val = quantile3 +(1.5 * iqr_value)

print(lower_bound_val,upper_bound_val)

7.5 19.5
```

Check for Categorical columns and perform encoding

```
df_numeric = df[['RowNumber', 'CustomerId', 'CreditScore', 'Age',
'Tenure',
'Balance',
'NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary','Exited
']]
df_categorical = df[['Surname', 'Geography', 'Gender']]

df_numeric.head()
```

```
   RowNumber  CustomerId  CreditScore  Age  Tenure    Balance
NumOfProducts  \
0          1    15634602          619   42       2       0.00
1
1          2    15647311          608   41       1   83807.86
1
2          3    15619304          502   42       8  159660.80
3
3          4    15701354          699   39       1       0.00
2
4          5    15737888          850   43       2  125510.82
1

   HasCrCard  IsActiveMember  EstimatedSalary  Exited
0          1               1        101348.88       1
1          0               1        112542.58       0
2          1               0        113931.57       1
```

```
3          0               0         93826.63       0
4          1               1         79084.10       0
```

```
df_categorical.head()
```

```
    Surname Geography  Gender
0  Hargrave    France  Female
1      Hill     Spain  Female
2      Onio    France  Female
3      Boni    France  Female
4  Mitchell     Spain  Female
```

```
print(df['Surname'].unique())
print(df['Geography'].unique())
print(df['Gender'].unique())
```

```
['Hargrave' 'Hill' 'Onio' ... 'Kashiwagi' 'Aldridge' 'Burbidge']
['France' 'Spain' 'Germany']
['Female' 'Male']
```

```
from sklearn.preprocessing import LabelEncoder
marry_encoder = LabelEncoder()

marry_encoder.fit(df_categorical['Gender'])

LabelEncoder()
marry_values = marry_encoder.transform(df_categorical['Gender'])
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']
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']
```

```
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']
```

```
from sklearn.preprocessing import OneHotEncoder
gender_encoder = OneHotEncoder()
```

```
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']]
```

```
from sklearn.preprocessing import OneHotEncoder
gender_encoder = OneHotEncoder()
```

```
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']]
```

```
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']]
```

```
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']]
```

```
df_categorical_encoded = pd.get_dummies(df_categorical,
drop_first=True)
df_categorical_encoded.head()
```

```
   Surname_Abbie  Surname_Abbott  Surname_Abdullah  Surname_Abdulov  \
0              0               0                 0                0
1              0               0                 0                0
2              0               0                 0                0
3              0               0                 0                0
4              0               0                 0                0

   Surname_Abel  Surname_Abernathy  Surname_Abramov  Surname_Abramova
\
0             0                  0                0                 0

1             0                  0                0                 0

2             0                  0                0                 0

3             0                  0                0                 0

4             0                  0                0                 0
```

```
   Surname_Abramovich  Surname_Abramowitz  ...  Surname_Zotova
Surname_Zox  \
0                   0                   0  ...               0
0
1                   0                   0  ...               0
0
2                   0                   0  ...               0
0
3                   0                   0  ...               0
0
4                   0                   0  ...               0
0

   Surname_Zubarev  Surname_Zubareva  Surname_Zuev  Surname_Zuyev  \
0                0                 0             0              0
1                0                 0             0              0
2                0                 0             0              0
3                0                 0             0              0
4                0                 0             0              0

   Surname_Zuyeva  Geography_Germany  Geography_Spain  Gender_Male
0               0                  0                0            0
1               0                  0                1            0
2               0                  0                0            0
3               0                  0                0            0
4               0                  0                1            0

[5 rows x 2934 columns]
```

```
df_new = pd.concat([df_numeric, df_categorical_encoded], axis=1)
df_new.head()
```

```
   RowNumber  CustomerId  CreditScore  Age  Tenure    Balance
NumOfProducts  \
0          1    15634602          619   42       2       0.00
1
1          2    15647311          608   41       1   83807.86
1
2          3    15619304          502   42       8  159660.80
3
3          4    15701354          699   39       1       0.00
2
4          5    15737888          850   43       2  125510.82
1

   HasCrCard  IsActiveMember  EstimatedSalary  ...  Surname_Zotova  \
0          1               1        101348.88  ...               0
1          0               1        112542.58  ...               0
```

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

   Surname_Zox  Surname_Zubarev  Surname_Zubareva  Surname_Zuev  \
0            0                0                 0             0
1            0                0                 0             0
2            0                0                 0             0
3            0                0                 0             0
4            0                0                 0             0

   Surname_Zuyev  Surname_Zuyeva  Geography_Germany
Geography_Spain  \
0              0               0                  0                0

1              0               0                  0                1

2              0               0                  0                0

3              0               0                  0                0

4              0               0                  0                1


   Gender_Male
0            0
1            0
2            0
3            0
4            0

[5 rows x 2945 columns]
```

Split The data into dependent and independent variables.

```
df=pd.read_csv('/content/Churn_Modelling.csv')

print(df["Balance"].min())
print(df["Balance"].max())
print(df["Balance"].mean())
```

```
0.0
250898.09
76485.889288
```

```
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
```

```
print(df.shape)
```

```
(10000, 14)
```

```
print(df.size)
```

```
140000
```

```
X = df.iloc[:, :-1].values
print(X)
```

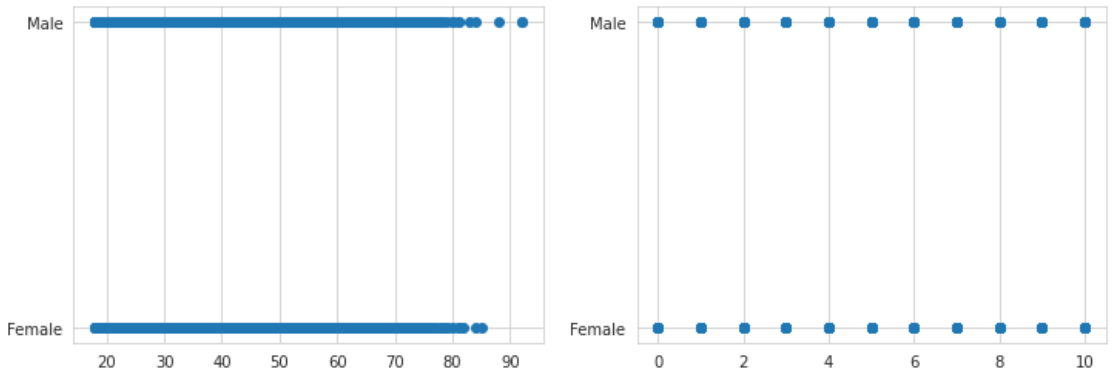
```
[[1 15634602 'Hargrave' ... 1 1 101348.88]
 [2 15647311 'Hill' ... 0 1 112542.58]
 [3 15619304 'Onio' ... 1 0 113931.57]
 ...
 [9998 15584532 'Liu' ... 0 1 42085.58]
 [9999 15682355 'Sabbatini' ... 1 0 92888.52]
 [10000 15628319 'Walker' ... 1 0 38190.78]]
```

```
Y = df.iloc[:, -1].values
print(Y)
```

```
[1 0 1 ... 1 1 0]
```

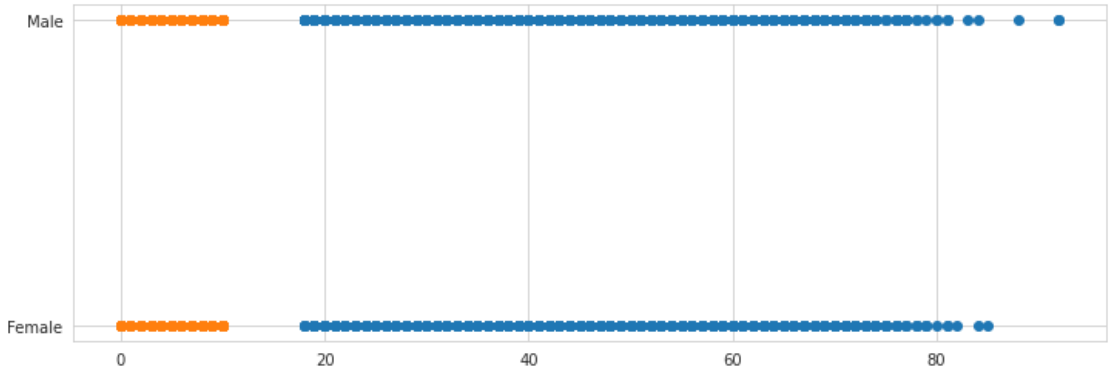
Scale the independent variables

```
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()
```



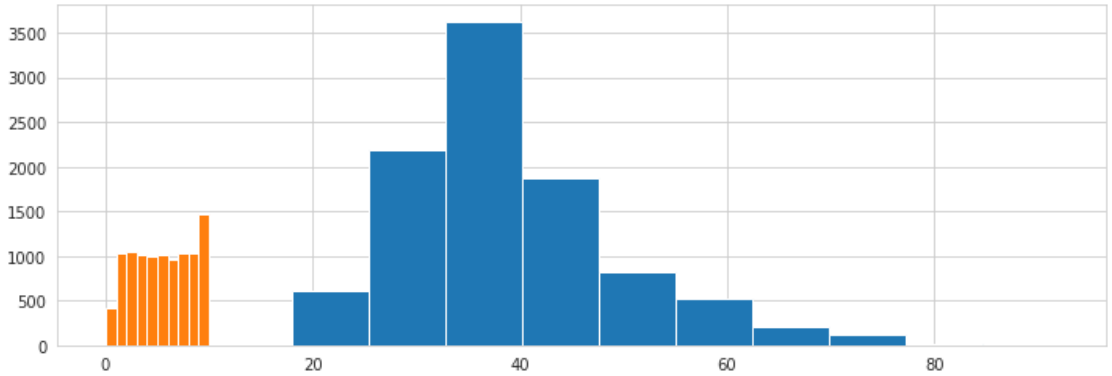

```
fig, ax = plt.subplots(figsize=(12, 4))
ax.scatter(x[:,0], y)
ax.scatter(x[:,1], y)
```

```
<matplotlib.collections.PathCollection at 0x7f237fc6f810>
```



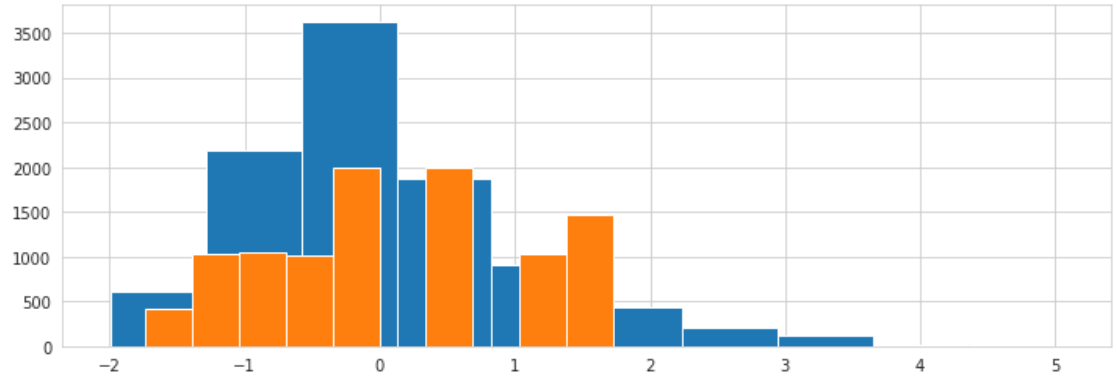

```
fig, ax = plt.subplots(figsize=(12, 4))
ax.hist(x[:,0])
ax.hist(x[:,1])
```

```
(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>)
```



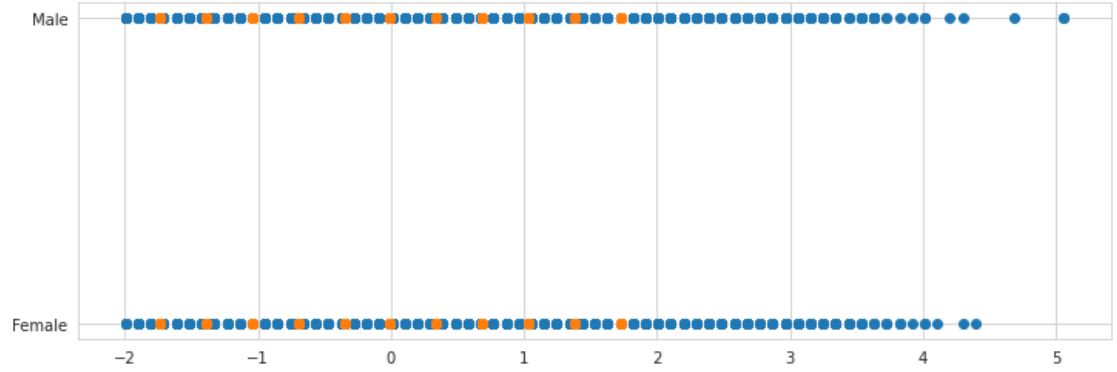
```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
fig, ax = plt.subplots(figsize=(12, 4))
scaler = StandardScaler()
x_std = scaler.fit_transform(x)
ax.hist(x_std[:,0])
ax.hist(x_std[:,1])
```

```
(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>)
```




```
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)
```

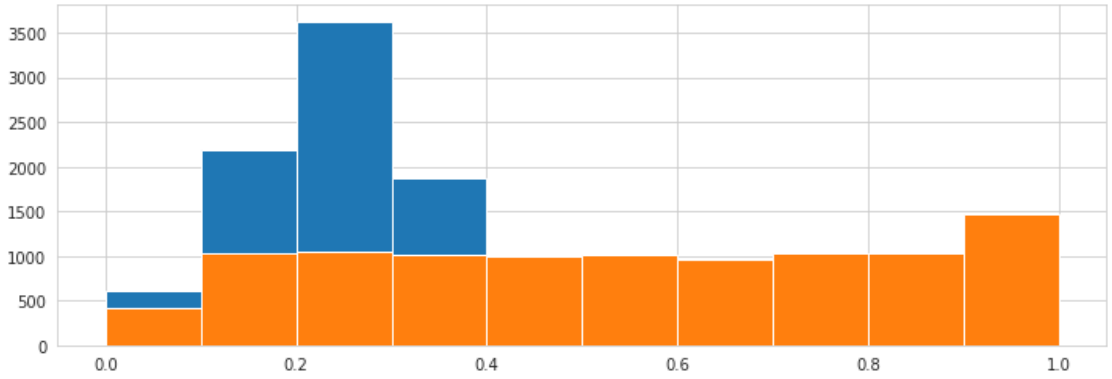
```
<matplotlib.collections.PathCollection at 0x7f237fa39e10>
```




```
fig, ax = plt.subplots(figsize=(12, 4))
scaler = MinMaxScaler()
```

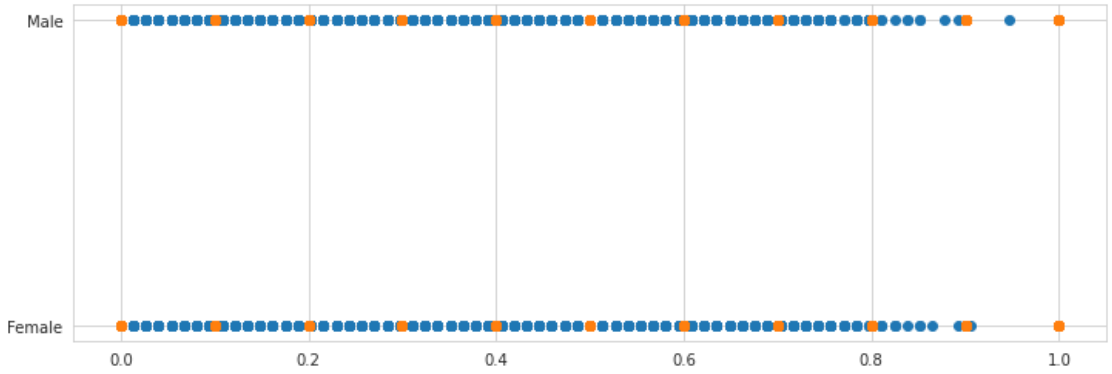
```
x_minmax = scaler.fit_transform(x)
ax.hist(x_minmax [:,0])
ax.hist(x_minmax [:,1])
```

```
(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>)
```



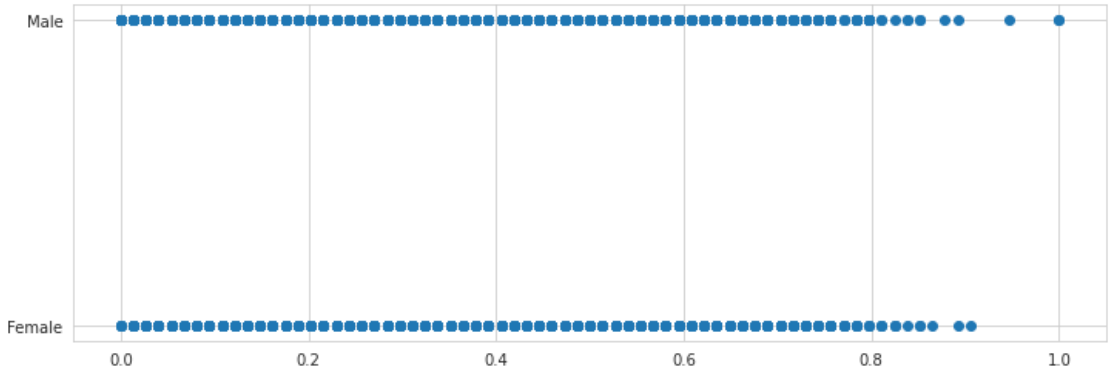

```
fig, ax = plt.subplots(figsize=(12, 4))
scaler = MinMaxScaler()
x_minmax = scaler.fit_transform(x)
ax.scatter(x_minmax [:,0], y)
ax.scatter(x_minmax [:,1], y)
```

```
<matplotlib.collections.PathCollection at 0x7f237f915190>
```



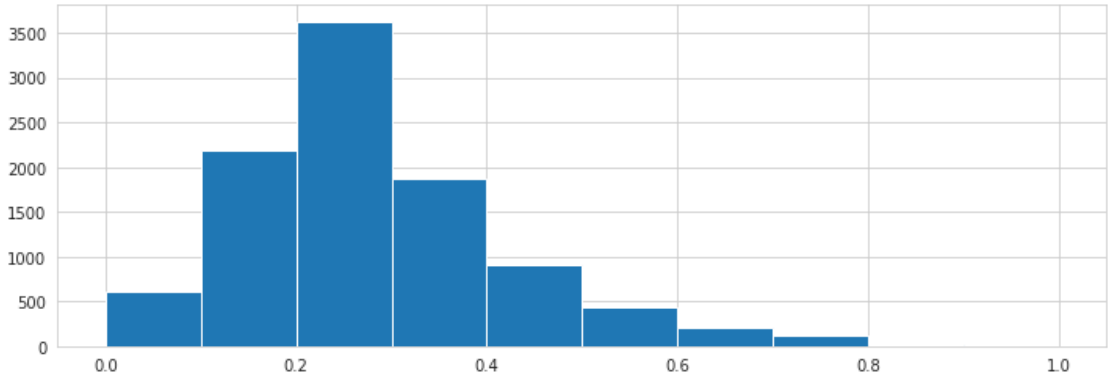

```
fig, ax = plt.subplots(figsize=(12, 4))
scaler = MinMaxScaler()
x_minmax = scaler.fit_transform(x)
ax.scatter(x_minmax [:,0], y)
```

```
<matplotlib.collections.PathCollection at 0x7f237f986750>
```




```
fig, ax = plt.subplots(figsize=(12, 4))
scaler = MinMaxScaler()
x_minmax = scaler.fit_transform(x)
ax.hist(x_minmax [:,0])
```

```
(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>)
```




```
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

df = pd.read_csv('/content/Churn_Modelling.csv')
x = df[['Age', 'Tenure']].values
y = df['Balance'].values
X_train, X_test, Y_train, Y_test = train_test_split(x, y)
pipeline = Pipeline([
 ("MinMax Scaling", MinMaxScaler()),
```

```
 ("SGD Regression", SGDRegressor())
])
pipeline.fit(X_train, Y_train)
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:  56991.01231360579
Score -0.004056380328824938
```

Split the data into training and testing

```
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]
```

```
dataset.drop(["HasCrCard"],axis=1,inplace=True)
print(dataset.shape)
print(dataset.head(10))
```

```
(10000, 13)
   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  IsActiveMember  EstimatedSalary
Exited
0       2       0.00              1               1        101348.88
1
```

```
1       1   83807.86             1               1        112542.58
0
2       8  159660.80             3               0        113931.57
1
3       1       0.00             2               0         93826.63
0
4       2  125510.82             1               1         79084.10
0
5       8  113755.78             2               0        149756.71
1
6       7       0.00             2               1         10062.80
0
7       4  115046.74             4               0        119346.88
1
8       4  142051.07             2               1         74940.50
0
9       2  134603.88             1               1         71725.73
0
```

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

```
array([[1, 15634602, 'Hargrave', ..., 1, 1, 101348.88],
       [2, 15647311, 'Hill', ..., 1, 1, 112542.58],
       [3, 15619304, 'Onio', ..., 3, 0, 113931.57],
       ...,
       [9998, 15584532, 'Liu', ..., 1, 1, 42085.58],
       [9999, 15682355, 'Sabbatini', ..., 2, 0, 92888.52],
       [10000, 15628319, 'Walker', ..., 1, 0, 38190.78]],
dtype=object)
```

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

```
array([1, 0, 1, ..., 1, 1, 0])
```