

Student Name	Princy S
Student Reg Number	960519104058
Assignment Number	02

1 .DownloadDataset:Churn-Modelling

2.Load the Dataset import numpy as np import pandas as pd
import seaborn as sns import matplotlib.pyplot as plt df =
d.read_csv('/content/drive/MyDrive/Churn_Modelling.csv')

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age
1			619	France	Female	42
			608	Spain	Female	41
			502	France	Female	42
			699	France	Female	39
			850	Spain	Female	43

df.head()

15634602 Hargrave

2 15647311 Hill

2 3 1 5619304 Onio

3 4 15701354 Boni

4 5 15737888 Mitchell

Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
02	0.00	1		1
1	83807.86	1		1
2	8	159660.80	3	1
3	1	0.00	2	0
4	2	125510.82	1	1

EstimatedSalary Exited

o 101348.88 1

1 1 12542.58 0

2 113931.57 1

3 93826.63

4 79084.10

```
df = df.drop(columns=['RowNumber', 'CustomerId', 'Surname'])
```

```
df.head()
```

2

5

0

2

F

r

a

n

c

e

F

e

m

a

l

e

4

2

8

1

5

9

6

6

0

.

8

0

3

3

6

9

9

F

r

a

n

c

e

F

e

m

a

l

e

3

9

1

0

.

0

0

2

4

8

5

0

S

p
a
i
n
F
e
m
a
l
e

4
3

2
1
2
5
5
1
0
.
8
2

1

HasCrCard IsActiveMember EstimatedSalary Exited

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

```
df['IsActiveMember'] = df['IsActiveMember'].astype('category')
df['Exited'] = df['Exited'].astype('category')
df['HasCrCard'] = df['HasCrCard'].astype('category')
```

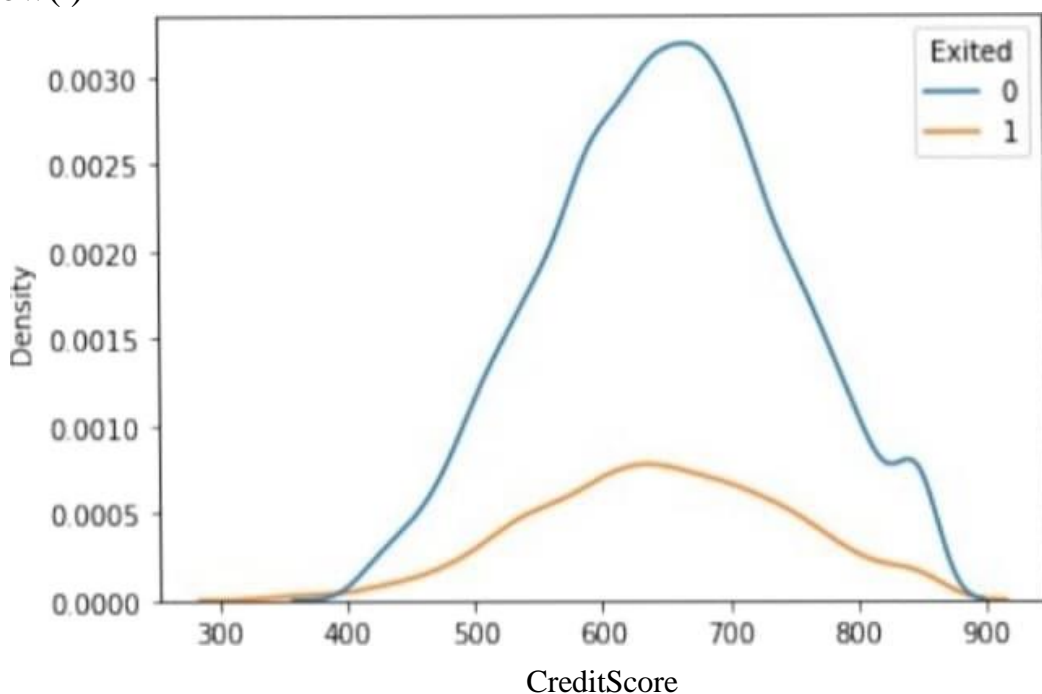
3. Perform

Univariate Analysis

Bi Variate Analysis

Multi • Variate Analysis

```
sns.deplot(x='CreditScore', data = df, hue = 'Exited')
plt.show()
```

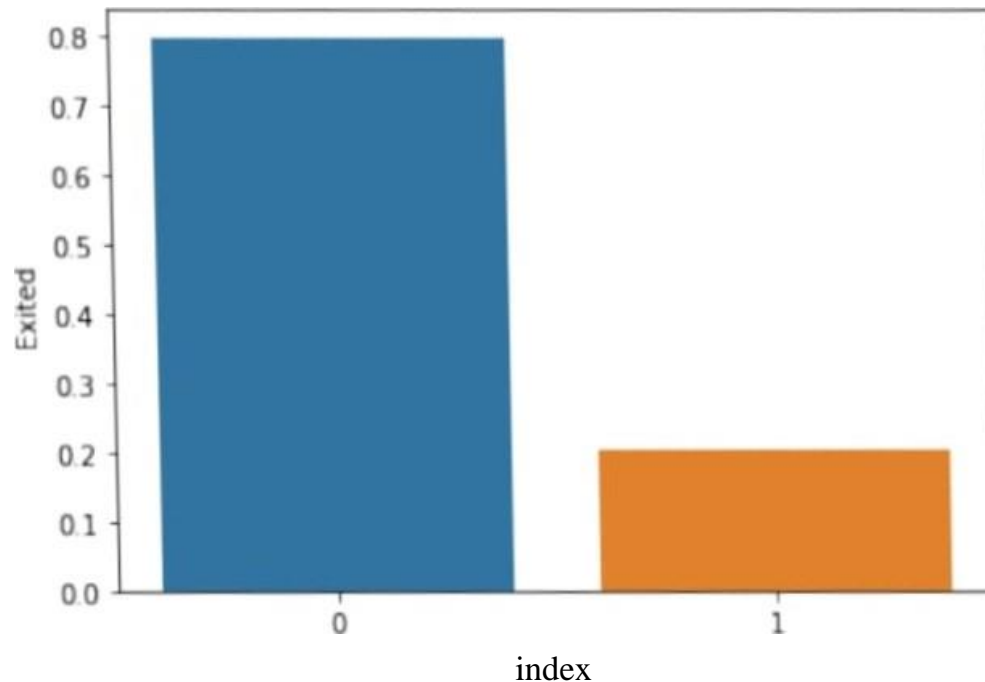


```
density = df.groupby('Exited').value_counts(normalize=True).reset_index()
sns.index = 'Exited'
```

index Exited

0 0.7963

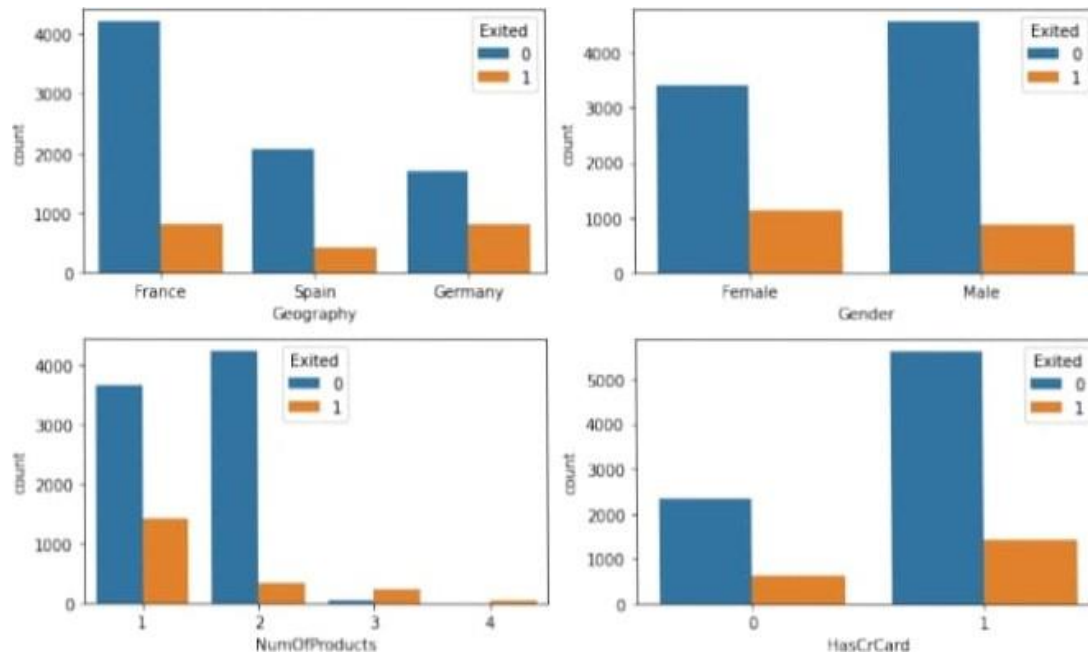
1 0.2037



```
categorical = categorical.drop(['CreditScore', 'Age', 'Tenure', 'Balance', 'EstimatedSalary'])
fig, axes = plt.subplots(2, 10, 6)
axes = axes.flatten()

for row in range(rows):
    cols = min(2, categorical.shape[1] - row * 2)
    for col in range(cols):
        col_name = categorical.columns[2 * row + col]
        sns.countplot(data=categorical, x=col_name, hue="Exited", ax=axes[row][col])

plt.tight_layout()
```



4. Descriptive statistics bold

text df . info()

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

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 11 columns) :

Column

9 EstimatedSalary 10 Exited dtypes:
category(3) , memory usage: 654.8+ df .
describe()

0 C creditScore

1 Geography 10000.000000 10000 .060000 mean 650

2 Gender .528800

3 Age 1.530200

4 Tenure Non-Null Count Dtype

5 Balance 10006 non-null int64

6 NumOfProducts 10000 non-null object

7 HasCrCard 10000 non-null object

8 IsActiveMember 10000 non-null int64

EstimatedSalary 10000 non-null int64

r

10000 non-
null float64
10000 non-null int64
10000 non-null
category
10000 non-null
category
10000 non-
null
float64
10000
non - null
category
float64(2
)
,
int64(4) ,
object(2)
KB

Age
Tenure
Balance

10000.000000
10000.000000
10000.000000

38.921800
5.012800
76485.889288

std	96.653299	10.487806	2.892174	62397.405202
0.581654				
min	350.000000	18.000000	0.000000	0.000000
.000000				
25%	584.000000	32.000000	3.000000	0.000000
1.000000				
	652.000000	37.000000	5.000000	97198.540000
1.000000				
	718.000000	44.000000	7.000000	127644.240000
2.000000				
max	850.000000	92.000000	10.000000	250898.090000
4.000000				

```

EstimatedSalary
count 10000.000000 mean
      106090.239881 std
      57510.492818 min
      11.580000 25%
      51002.110000
50% 100193.915000 75%
      149388.247500 max
      199992.480000

```

5. Handle Missing Values

```
df.isna().sum()
```

```

CreditScore      0
Geography        0
Gender Age       0
Tenure           0
Balance          0
NumOfProducts   0
HasCrCard        0
IsActiveMember   0
EstimatedSalary  0

```

```
Exited dtype: int64
```

In this dataset there is no missing values

6. Find the outliers and replace the outliers

Finding Outliers

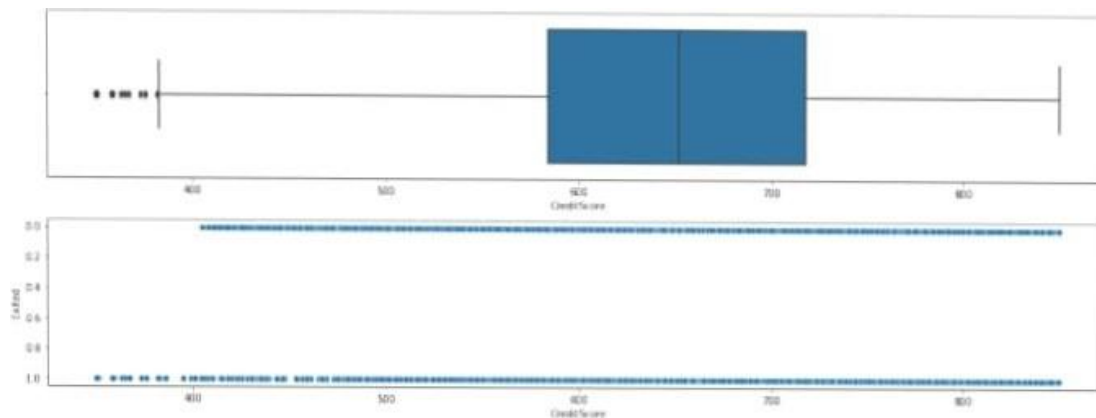
```

def box_scatter (data, x, y) :
    fig, _ (ax1, ax2) = plt . subplots(nrows=2, ncols=1, figsize=(16 , 6) )
    sns . x=x, ax=ax1) sns . x=x,y=y,ax=ax2)

```

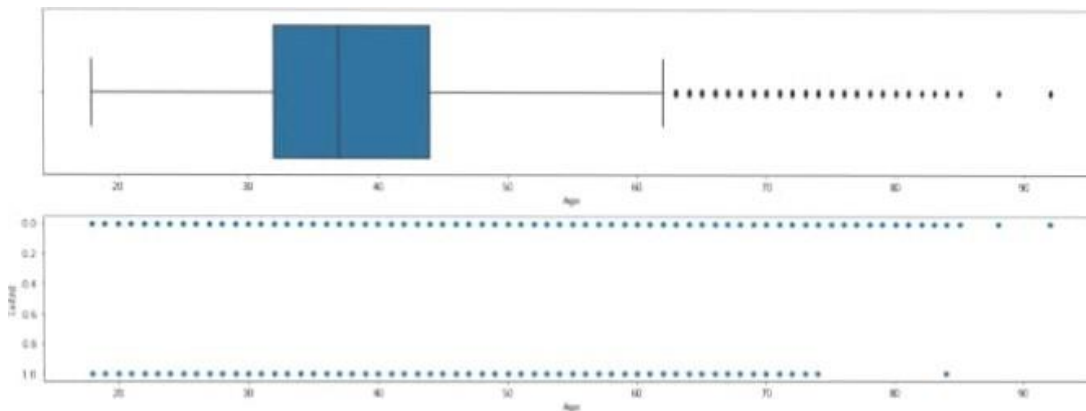
```
box scatter(df, 'CreditScore', 'Exited' );
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(df.loc[df['CreditScore'] < 400])}")
```

of Bivariate Outliers: 19



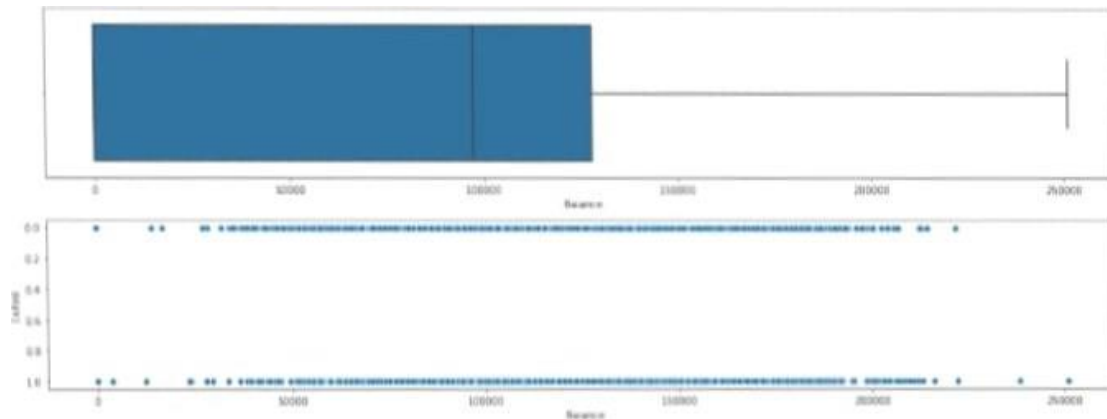
```
box scatter(df, 'Age', 'Exited' );
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(df.loc[df['Age'] > 87])}")
```

of Bivariate Outliers: 3

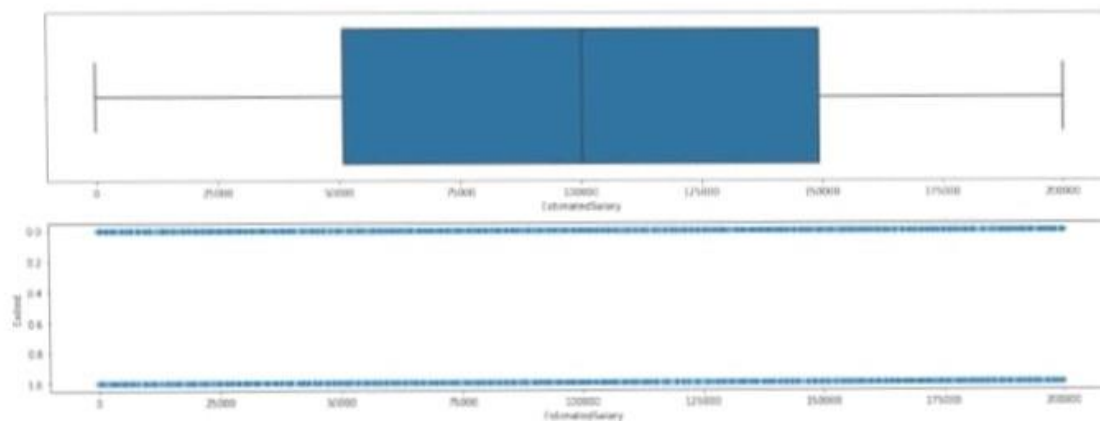


```
box scatter(df, 'Balance', 'Exited' );
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(df.loc[df['Balance'] > 220000])}")
```

of Bivariate Outliers: 4



```
box scatter(df, 'EstimatedSalary', 'Exited' );
plt.tight_layout()
```



Removing The Outliers

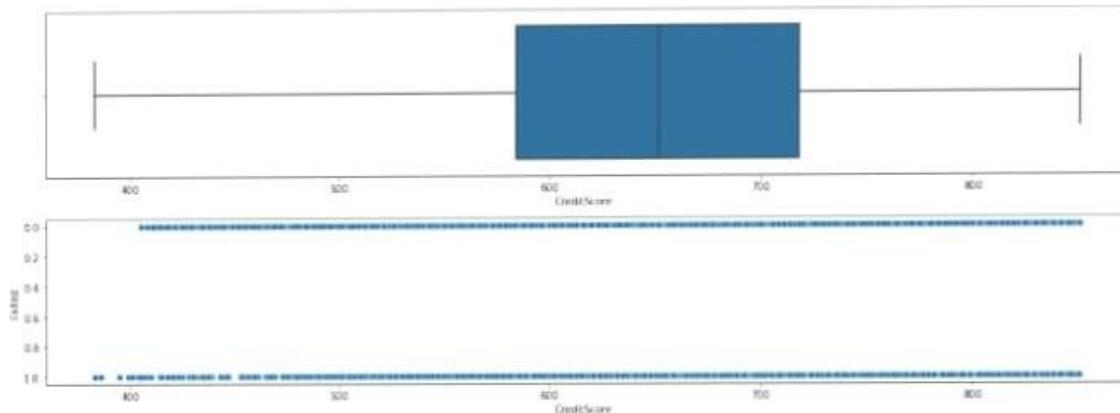
```
for i in df:
    if df[i].dtype == 'int64' or df[i].dtype == 'float64':
        q1 = df[i].quantile(0.25)
        q3 = df[i].quantile(0.75)
        iqr = q3 - q1
        lower = q1 - 1.5 * iqr
        upper = q3 + 1.5 * iqr
        df[i] = df[i].between(lower, upper)
```

```
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(df.loc[df['CreditScore'] > 800])}")
```

Bivariate Outliers: {len(df.loc[df['CreditScore'] > 800])}

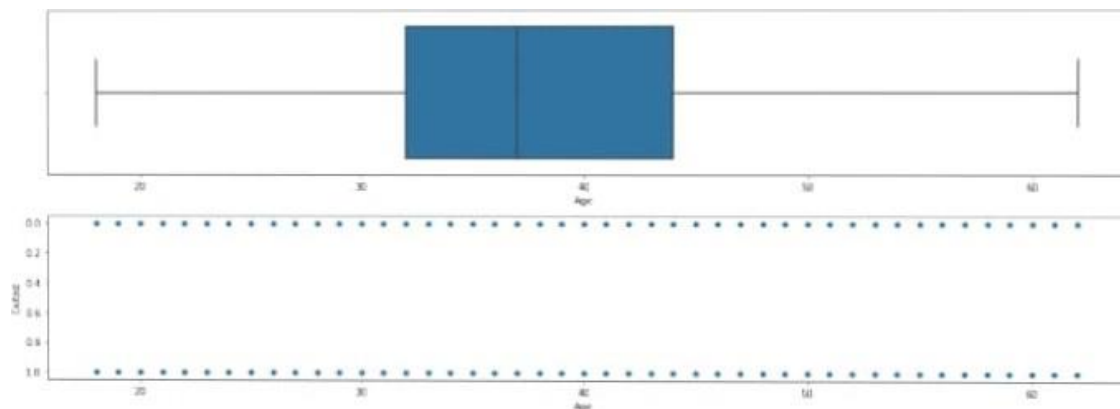
<

of Bivariate Outliers: 19



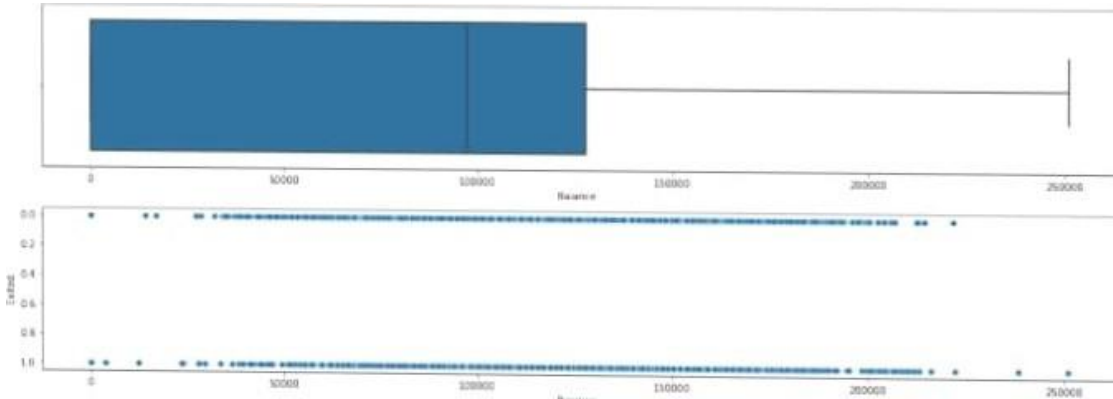
```
box_scatter(df, 'Age', 'Exited'); plt.tight_layout()
print(f"# of Bivariate Outliers: {len(df.loc[df['Age'] > 80])}")
```

of Bivariate Outliers: 0



```
box_scatter(df, 'Balance', 'Exited');
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(df.loc[df['Balance'] > 2206061])}")
```

of Bivariate Outliers: 4



7. Check for Categorical columns and perform encoding.
 from sklearn.preprocessing import LabelEncoder encoder=LabelEncoder()
 for i in df:

```
if df[i].dtype=='object' or df[i].dtype=='category':
    encoder.fit_transform(df[i])
```

8. Split the data into dependent and independent variables.

```
x=df.iloc[:, :-1]
```

X. head()

	CreditScore	Geography	Gender	Age	Tenure	Balance
0	619.0			42.0	2.0	0.00
1	608.0		2	41.0	1.0	83867.86
2	502.0		0	42.0	8.0	159660.80
3	699.0		0	39.0	1.0	0.00
4	850.0		2	43.0	2.0	125510.82

HasCrCard IsActiveMember EstimatedSalary

1	1	101348.88	1	112542.58
2	1	113931.57		

```

3    93826.63
4    1    1    79084.10
iloc[:, -11]

```

```

y. head ( )
0    1
1    0
2    1
3    0
4    0

```

Name: Exited , dtype: int64

9. Scale the independent variables from sklearn .
preprocessing import StandardScaler scaler=StandardScaler()
x=scaler.fit_transform(x) print (x)

```

[(-0.32687761 -0.90188624 -1.09598752... 0. 64609167 0 .97024255
0. 021886491

```



```

[-0.44080365 1.51506738 -1.09598752 . . -1.54776799 o .97024255
 o. 21653375]
[-1.53863634 -o .90188624 -1.09598752 . . . o. 64609167-1.03067011
 0.2406869
[ O. 60524449 -o .90188624 -1.09598752 . . -1.54776799 o .97024255
 -1.008643081
[ 1.25772996 0.30659057 0.91241915 . . o .64609167-1.03067011
 -o. 125230711
t 1.4648682 -o .90188624 -1.09598752 . . o. 64609167-1.03067011
 -1.07636976M

```

10. Split the data into training and testing.

```

from sklearn . model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(x , y , test_size=0.20)

print (x_train . shape)
print (x_test . shape)
(8000, 10)
(2000, 10)

print (y_train . shape)
print (y_test . shape)
(8000 , )
(2000, )

```