Assignment -2

Data Visualization and Pre-processing

Assignment Date	30 September 2022
Student Name	M.Sanjay
Student Roll Number	CS19036
Maximum Marks	2 Marks

In [ ]:

```
# Importing required libraries

import numpy as np import
pandas as pd
```

In [ ]:

```
# Reading the dataset

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

In [ ]:

```
# Visualizing 1st 50 data

df.head()
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCar
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	

4	15701354	Boni	699	France	Female	39	1	0.00	2	

In [ ]:

Checking for null values

df.isnull().sum() Out[
]:

RowNumber	0
CustomerId	0
Surname	0
CreditScore	0
Geography	0
Gender	0
Age	0
Tenure	0
Balance	0
NumOfProducts	0
HasCrCard	0
IsActiveMember	0
EstimatedSalary	0
Exited dtype:	0

int64

In [ ]:
df.dtypes

Out[ ]:

RowNumber	int64
CustomerId	int64
Surname	object
CreditScore	int64
Geography	object
Gender	object
Age	int64
Tenure	int64
Balance	float64
NumOfProducts	int64
HasCrCard	int64
IsActiveMember	int64
EstimatedSalary	float64 Exited

int64 dtype:

object

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

In [47]:

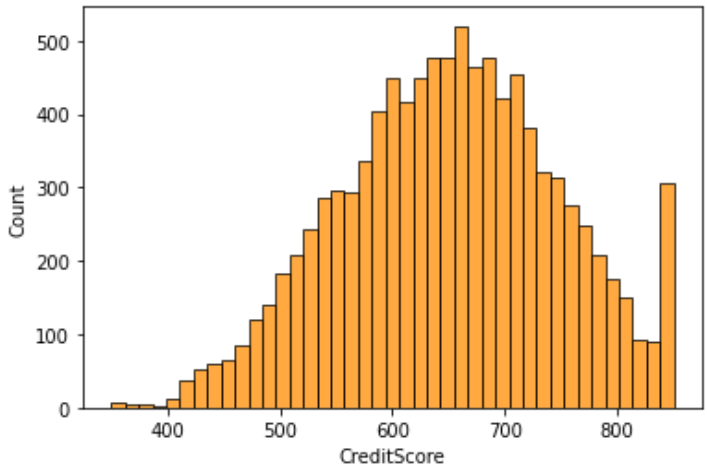
Univariate Analysis

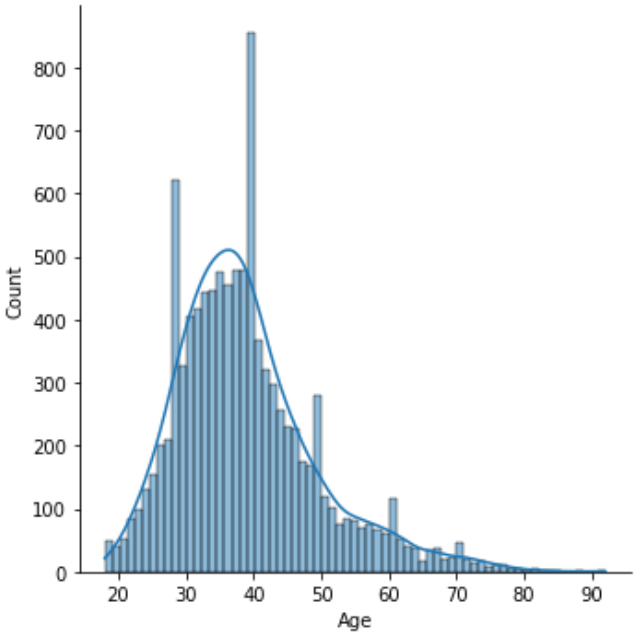
```
sns.histplot(data["CreditScore"],color='darkorange')
```

In [48]:

Out[48]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f831677f6d0>






```
sns.displot(data['Age'], kde=True)
```

In [49]:

Out[49]:

<seaborn.axisgrid.FacetGrid at 0x7f831661b210>

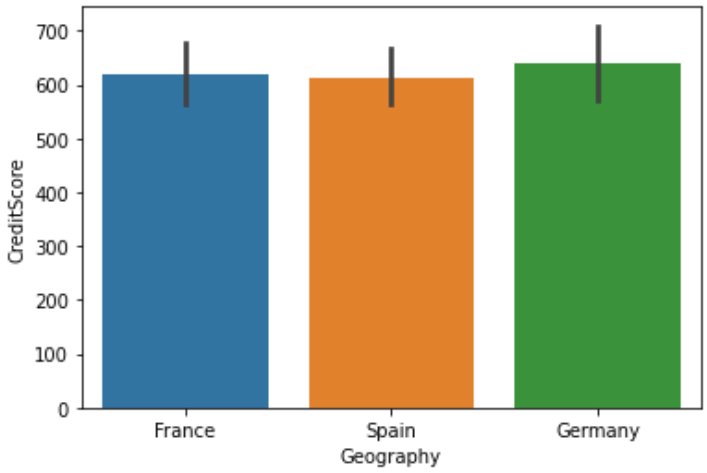
Bi - Variate Analysis

```
sns.barplot(data=data.head(50), x="Geography", y="CreditScore")
```

In [50]:

Out[50]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f8313ce63d0>

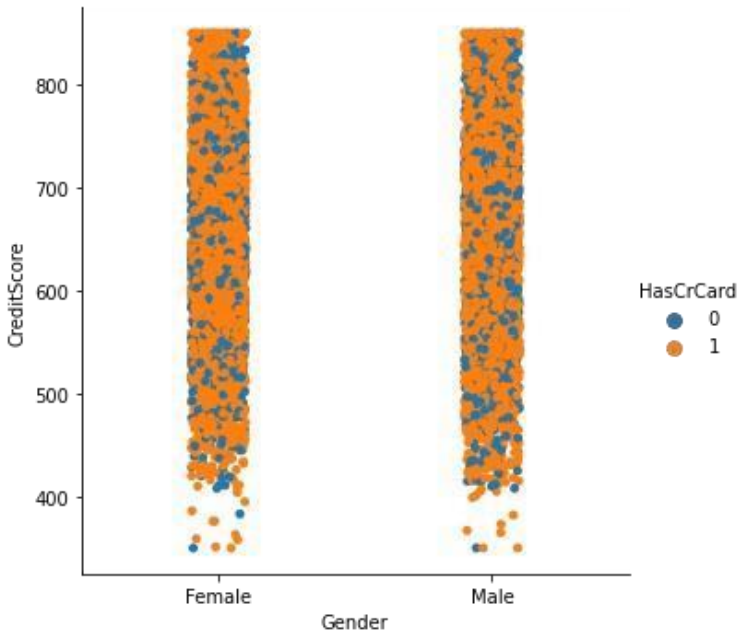



```
sns.catplot(x='Gender', y='CreditScore', hue='HasCrCard', data=data)
```

In [51]:

Out[51]:

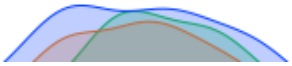
<seaborn.axisgrid.FacetGrid at 0x7f8317198a90>

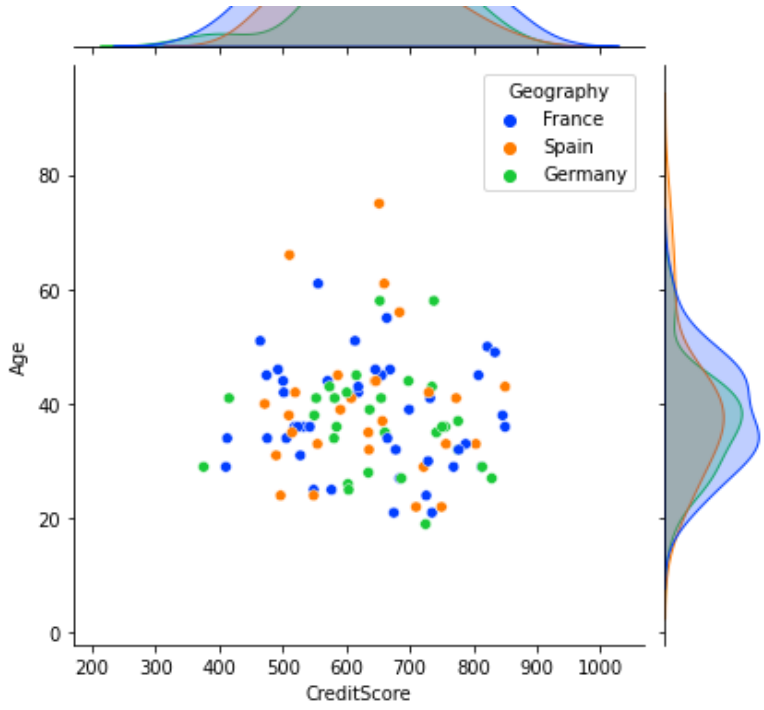


Multi - Variate Analysis

```
sns.jointplot(
    x='CreditScore',
    y='Age',
    data=data.head(100),
    palette='bright',
    hue='Geography');
```

In [52]:



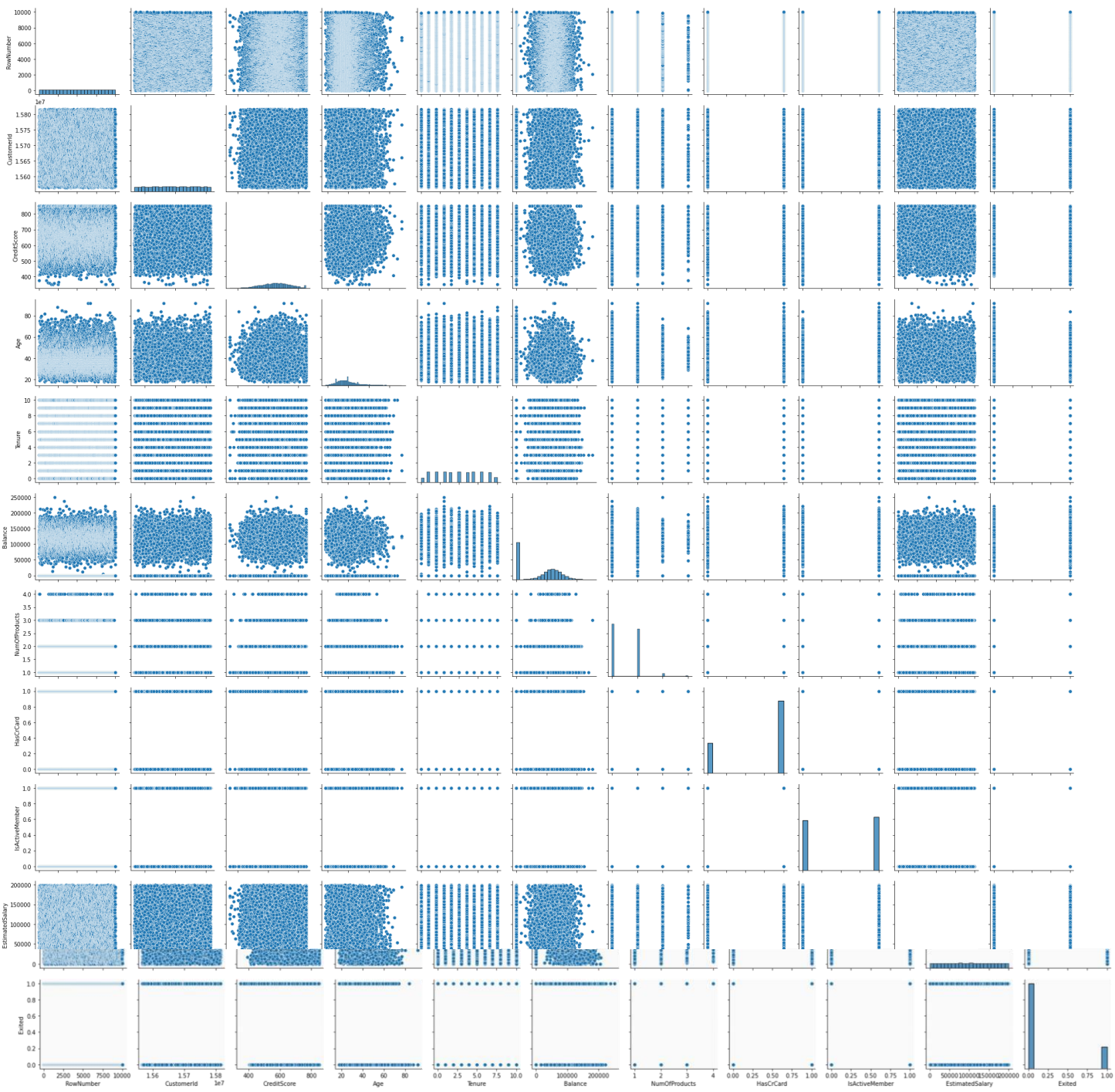



```
sns.pairplot(data)
```

In [53]:

Out[53]:

<seaborn.axisgrid.PairGrid at 0x7f8313a71390>


RowNumber
CustomerId
CreditScore
Age
Tenure
Balance
NumOfProducts
HasCrCard
IsActiveMember
EstimatedSalary
Exited
1e7

Perform descriptive statistics on the dataset

```
data.describe()
```

In [54]:

Out[54]:

	RowNum	CustomerI	CreditScor	Age	Tenure	Balance	NumOfProd ucts	HasCrCa rd	I ber s	d e
count	10000.000000	1.000000e+07	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000		
mean	5000.500000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.705500		
std	2886.895680	7.193619e+07	96.653299	10.487806	2.892174	62397.405202	0.581654	0.455840		
min	1.000000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.000000		
25%	2500.750000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.000000		
50%	5000.500000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.000000		
75%	7500.250000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.000000		
max	10000.000000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.000000		

Handle the Missing values

```
data.isnull().sum()
```

In [55]:

Out[55]:

RowNumber 0

CustomerId 0

Surname 0

CreditScore 0

Geography 0

Gender 0

Age 0

Tenure 0

Balance 0

NumOfProducts 0

HasCrCard 0

IsActiveMember 0

EstimatedSalary 0

Exited 0

dtype: int64

Find the outliers and replace the outliers

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argu
ment will be `data`, and passing other arguments without an explicit keyword will result
```

```
import seaborn as sns

sns.boxplot(data['CreditScore'])
```

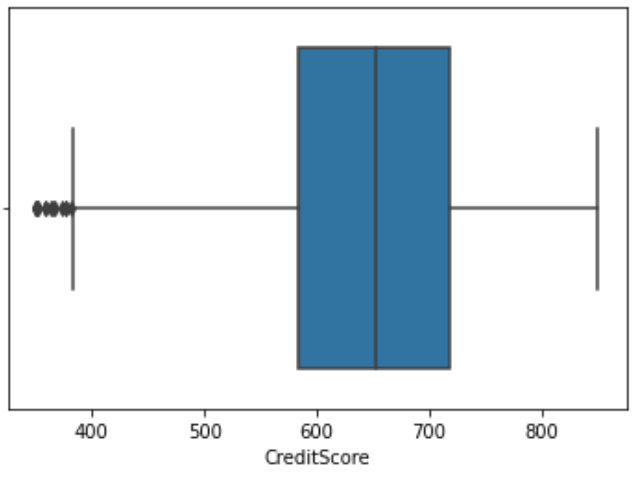
```
in an error or misinterpretation.
```

In [56]:

FutureWarning

Out[56]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f8310b82990>



```
import numpy as np

Q1 = np.percentile(data['CreditScore'], 25,
                   interpolation = 'midpoint')

Q3 = np.percentile(data['CreditScore'], 75,
                   interpolation = 'midpoint')
IQR = Q3 - Q1



#Upper bound
upper = np.where(data['CreditScore'] >= (Q3+1.5*IQR))
#Lower bound
lower = np.where(data['CreditScore'] <= (Q1-1.5*IQR))


print("Q3: ",Q3)
print("Q1: ",Q1)
print("IQR: ",IQR)


mean = data["CreditScore"].mean()

data["CreditScore"] = np.where(data["CreditScore"] > 850, mean, data['CreditScore'])
data["CreditScore"] = np.where(data["CreditScore"] < 400, mean, data['CreditScore'])



sns.boxplot(data['CreditScore'])
```

In [57]:

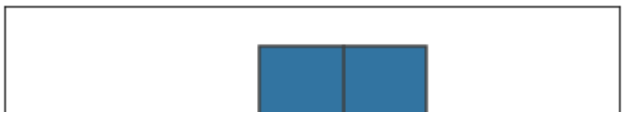
Q3: 718.0

Q1: 584.0

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argu
ment will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation. FutureWarning
```

IQR: 134.0 Out[57]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f83177a7310>



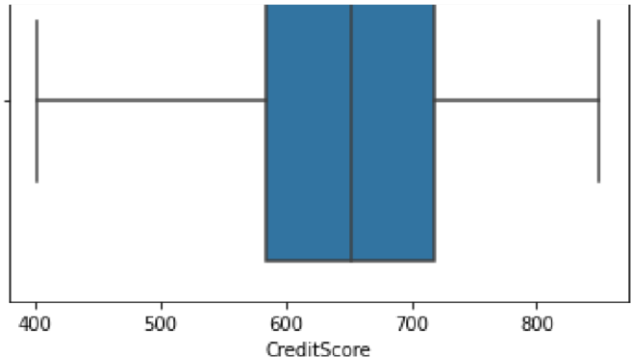
Check for Categorical columns and perform encoding

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data['Geography'] = le.fit_transform(data['Geography'])
data['Gender']  = le.fit_transform(data['Gender'])

data.head()
```

In [58]:



Out[58]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
0	1	15634602	Hargrave	619.0	0	0	42	2	0.00	1	1	1
1	2	15647311	Hill	608.0	2	0	41	1	83807.86	1	0	1
2	3	15619304	Onio	502.0	0	0	42	8	159660.80	3	1	0
4	5	15737888	Mitchell	850.0	2	0	43	2	125510.82	1	1	1
3	4	15701354	Boni	699.0	0	0	39	1	0.00	2	0	0

Split the data into dependent and independent variables

```
y = data['CreditScore'] #dependent
x = data.drop(columns = ['CreditScore'],axis = 1) #independent
```

```
x . head()
```

In [59]:

Out[59]:

	RowNumber	CustomerId	Surname	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
0	1	15634602	Hargrave	0	0	42	2	0.00	1	1	1
1	2	15647311	Hill	2	0	41	1	83807.86	1	0	1
2	3	15619304	Onio	0	0	42	8	159660.80	3	1	0

3	4	15701354	Boni	0	0	39	1	0.00		2	0
4	5	15737888	Mitchell	2	0	43	2	125510.82		1	1

Scale the independent variables

```
names = ['RowNumber','CustomerId','Geography','Gender','Age','Tenure','Balance','NumOfPro
```

In [60]:

ducts','HasCrCard','IsActiveMember','EstimatedSalary','Exited']

In [61]:

```
from sklearn.preprocessing import scale
```

x = scale(x[names]) x Out[61]:

```
array([[-1.73187761, -0.78321342, -0.90188624, ..., 0.97024255,
        0.02188649, 1.97716468],
       [-1.7315312 , -0.60653412, 1.51506738, ..., 0.97024255,
        0.21653375, -0.50577476],
       [-1.73118479, -0.99588476, -0.90188624, ..., -1.03067011,
        0.2406869 , 1.97716468],
       ...,
       [ 1.73118479, -1.47928179, -0.90188624, ..., 0.97024255,
       -1.00864308, 1.97716468],
       [ 1.7315312 , -0.11935577, 0.30659057, ..., -1.03067011,
       -0.12523071, 1.97716468],
       [ 1.73187761, -0.87055909, -0.90188624, ..., -1.03067011,
       -1.07636976, -0.50577476]])
```

In [62]: x = pd.DataFrame(x,columns = names)x

.head()

Out[62]:

	RowNum	Customer Id	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Hy
1	-1.731531	-0.606534	1.515067	1.0959 88	0.19816 4 38	1.3875 50	0.1173	-0.911583	- 0.97024	1.547768	
0	-1.731878	-0.783213	- 0.901886	1.0959 88	0.29351 7 60	1.0417	1.2258 48	-0.911583	0.646092		0.97024
2	-1.731185	-0.995885	- 0.901886	1.0959 88	0.29351 7	1.0329 08	1.3330 53	2.527057	0.646092		-1.03067
3	-1.730838	0.144767	- 0.901886	1.0959 88	0.00745 7 38	1.3875 48	1.2258	0.807737	- 1.547768		-1.03067
4	-1.730492	0.652659	1.515067	1.0959 88	0.38887	1.0417 1 60	0.7857 28	0.911583	0.646092		- 0.97024

Split the data into training and testing

In [69]:

```
from sklearn.model_selection import train_test_split

# Split training and testing data

xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.20,random_state=0)
```

In [70]:

```
# Checking shape of data
```

xtrain.shape,xtest.shape Out[70]: ((8000, 12), (2000, 12))