```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('/content/Credit Score Classification Dataset.csv')
df
```

| | Age | Gender | Income | Education | Marital Status | Number of Children | Home Ownership | Credit Score |
|---|---|---|---|---|---|---|---|---|
| **0** | 25 | Female | 50000.0 | Bachelor's Degree | Single | 0 | Rented | High |
| **1** | 30 | Male | 100000.0 | Master's Degree | Married | 2 | Owned | High |
| **2** | 35 | Female | 75000.0 | Doctorate | Married | 1 | Owned | High |
| **3** | 40 | Male | 125000.0 | High School Diploma | Single | 0 | Owned | High |
| **4** | 45 | Female | 100000.0 | Bachelor's Degree | Married | 3 | Owned | High |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **159** | 29 | Female | 27500.0 | High School Diploma | Single | 0 | Rented | Low |
| **160** | 34 | Male | 47500.0 | Associate's Degree | Single | 0 | Rented | Average |
| **161** | 39 | Female | 62500.0 | Bachelor's Degree | Married | 2 | Owned | High |
| **162** | 44 | Male | 87500.0 | Master's Degree | Single | 0 | Owned | High |
| **163** | 49 | Female | 77500.0 | Doctorate | Married | 1 | Owned | High |

164 rows × 8 columns

```
df.head()
```

| | Age | Gender | Income | Education | Marital Status | Number of Children | Home Ownership | Credit Score |
|---|---|---|---|---|---|---|---|---|
| **0** | 25 | Female | 50000.0 | Bachelor's Degree | Single | 0 | Rented | High |
| **1** | 30 | Male | 100000.0 | Master's Degree | Married | 2 | Owned | High |
| **2** | 35 | Female | 75000.0 | Doctorate | Married | 1 | Owned | High |
| **3** | 40 | Male | 125000.0 | High School Diploma | Single | 0 | Owned | High |
| **4** | 45 | Female | 100000.0 | Bachelor's Degree | Married | 3 | Owned | High |

```
df.tail()
```

| | Age | Gender | Income | Education | Marital Status | Number of Children | Home Ownership | Credit Score |
|---|---|---|---|---|---|---|---|---|
| **159** | 29 | Female | 27500.0 | High School Diploma | Single | 0 | Rented | Low |
| **160** | 34 | Male | 47500.0 | Associate's Degree | Single | 0 | Rented | Average |
| **161** | 39 | Female | 62500.0 | Bachelor's Degree | Married | 2 | Owned | High |
| **162** | 44 | Male | 87500.0 | Master's Degree | Single | 0 | Owned | High |
| **163** | 49 | Female | 77500.0 | Doctorate | Married | 1 | Owned | High |

```
df.shape
```

```
(164, 8)
```

```
df.dtypes
```

```
Age                    int64
Gender                object
Income               float64
Education             object
Marital Status        object
Number of Children     int64
Home Ownership        object
Credit Score          object
dtype: object
```

```
df=df.drop(['Gender'],axis=1)
```

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

```
Income                9
Education             0
Marital Status        0
Number of Children    0
Home Ownership        0
Credit Score          0
dtype: int64
```

```
# Filling the missing values
df['Income'].fillna(df['Income'].mean(),inplace=True)
df
```

| | Age | Income | Education | Marital Status | Number of Children | Home Ownership | Credit Score |
|---|---|---|---|---|---|---|---|
| **0** | 25 | 50000.0 | Bachelor's Degree | Single | 0 | Rented | High |
| **1** | 30 | 100000.0 | Master's Degree | Married | 2 | Owned | High |
| **2** | 35 | 75000.0 | Doctorate | Married | 1 | Owned | High |
| **3** | 40 | 125000.0 | High School Diploma | Single | 0 | Owned | High |
| **4** | 45 | 100000.0 | Bachelor's Degree | Married | 3 | Owned | High |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **159** | 29 | 27500.0 | High School Diploma | Single | 0 | Rented | Low |
| **160** | 34 | 47500.0 | Associate's Degree | Single | 0 | Rented | Average |
| **161** | 39 | 62500.0 | Bachelor's Degree | Married | 2 | Owned | High |
| **162** | 44 | 87500.0 | Master's Degree | Single | 0 | Owned | High |
| **163** | 49 | 77500.0 | Doctorate | Married | 1 | Owned | High |

164 rows × 7 columns

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

```
Age                   0
Income                0
Education             0
Marital Status        0
Number of Children    0
Home Ownership        0
Credit Score          0
dtype: int64
```

```
df.describe()
```

| | Age | Income | Number of Children |
|---|---|---|---|
| **count** | 164.000000 | 164.000000 | 164.000000 |
| **mean** | 37.975610 | 83951.612903 | 0.652439 |
| **std** | 8.477289 | 31285.974346 | 0.883346 |
| **min** | 25.000000 | 25000.000000 | 0.000000 |
| **25%** | 30.750000 | 60000.000000 | 0.000000 |
| **50%** | 37.000000 | 83951.612903 | 0.000000 |
| **75%** | 45.000000 | 105000.000000 | 1.000000 |
| **max** | 53.000000 | 162500.000000 | 3.000000 |

## Label encoding

```
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
df['Education']=lb.fit_transform(df['Education'])
```

```
df['Marital Status']=lb.fit_transform(df['Marital Status'])
```

```
df['Home Ownership']=lb.fit_transform(df['Home Ownership'])
```

```
df.dtypes
```

```
Age                    int64
Income                 float64
Education              int64
Marital Status         int64
Number of Children     int64
Home Ownership         int64
Credit Score           object
dtype: object
```

## Separating input and output samples

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

```
(array([[2.50000000e+01, 5.00000000e+04, 1.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 1.00000000e+00],
        [3.00000000e+01, 1.00000000e+05, 4.00000000e+00, 0.00000000e+00,
         2.00000000e+00, 0.00000000e+00],
        [3.50000000e+01, 7.50000000e+04, 2.00000000e+00, 0.00000000e+00,
         1.00000000e+00, 0.00000000e+00],
        [4.00000000e+01, 1.25000000e+05, 3.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 0.00000000e+00],
        [4.50000000e+01, 1.00000000e+05, 1.00000000e+00, 0.00000000e+00,
         3.00000000e+00, 0.00000000e+00],
        [5.00000000e+01, 1.50000000e+05, 4.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00],
        [2.60000000e+01, 4.00000000e+04, 0.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 1.00000000e+00],
        [3.10000000e+01, 6.00000000e+04, 1.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 1.00000000e+00],
        [3.60000000e+01, 8.00000000e+04, 4.00000000e+00, 0.00000000e+00,
         2.00000000e+00, 0.00000000e+00],
        [4.10000000e+01, 1.05000000e+05, 2.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 0.00000000e+00],
        [4.60000000e+01, 9.00000000e+04, 3.00000000e+00, 0.00000000e+00,
         1.00000000e+00, 0.00000000e+00],
        [5.10000000e+01, 1.35000000e+05, 1.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00],
        [2.70000000e+01, 3.50000000e+04, 3.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 1.00000000e+00],
        [3.20000000e+01, 5.50000000e+04, 0.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 1.00000000e+00],
        [3.70000000e+01, 7.00000000e+04, 1.00000000e+00, 0.00000000e+00,
         2.00000000e+00, 0.00000000e+00],
        [4.20000000e+01, 9.50000000e+04, 4.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 0.00000000e+00],
        [4.70000000e+01, 8.50000000e+04, 2.00000000e+00, 0.00000000e+00,
         1.00000000e+00, 0.00000000e+00],
        [5.20000000e+01, 1.25000000e+05, 3.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00],
        [2.80000000e+01, 3.00000000e+04, 0.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 1.00000000e+00],
        [3.30000000e+01, 5.00000000e+04, 3.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 1.00000000e+00],
        [3.80000000e+01, 6.50000000e+04, 1.00000000e+00, 0.00000000e+00,
         2.00000000e+00, 0.00000000e+00],
        [4.30000000e+01, 8.00000000e+04, 4.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 0.00000000e+00],
        [4.80000000e+01, 7.00000000e+04, 2.00000000e+00, 0.00000000e+00,
         1.00000000e+00, 0.00000000e+00],
        [5.30000000e+01, 1.15000000e+05, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00],
        [2.90000000e+01, 2.50000000e+04, 3.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 1.00000000e+00],
        [3.40000000e+01, 4.50000000e+04, 0.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 1.00000000e+00],
        [3.90000000e+01, 6.00000000e+04, 1.00000000e+00, 0.00000000e+00,
         2.00000000e+00, 0.00000000e+00],
        [4.40000000e+01, 7.50000000e+04, 4.00000000e+00, 1.00000000e+00,
         0.00000000e+00, 0.00000000e+00],
        [4.90000000e+01, 6.50000000e+04, 2.00000000e+00, 0.00000000e+00,
         1.00000000e+00, 0.00000000e+00],
```

## Training and testing data

## Training and testing data

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
```

```
x_train
```

```
array([[4.50000000e+01, 1.15000000e+05, 1.00000000e+00, 0.00000000e+00,
        3.00000000e+00, 0.00000000e+00],
       [5.10000000e+01, 1.35000000e+05, 1.00000000e+00, 0.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [4.50000000e+01, 1.10000000e+05, 1.00000000e+00, 0.00000000e+00,
        3.00000000e+00, 0.00000000e+00],
       [3.40000000e+01, 4.75000000e+04, 0.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 1.00000000e+00],
       [2.60000000e+01, 4.00000000e+04, 0.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 1.00000000e+00],
       [4.40000000e+01, 7.50000000e+04, 4.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [4.20000000e+01, 1.05000000e+05, 4.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [3.10000000e+01, 6.50000000e+04, 1.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 1.00000000e+00],
       [4.10000000e+01, 1.10000000e+05, 2.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [2.70000000e+01, 3.75000000e+04, 3.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 1.00000000e+00],
       [4.50000000e+01, 1.00000000e+05, 1.00000000e+00, 0.00000000e+00,
        3.00000000e+00, 0.00000000e+00],
       [4.00000000e+01, 1.30000000e+05, 3.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [4.00000000e+01, 1.30000000e+05, 3.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [4.10000000e+01, 1.10000000e+05, 2.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [4.70000000e+01, 8.39516129e+04, 2.00000000e+00, 0.00000000e+00,
        1.00000000e+00, 0.00000000e+00],
       [3.10000000e+01, 6.75000000e+04, 1.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 1.00000000e+00],
       [4.60000000e+01, 9.00000000e+04, 3.00000000e+00, 0.00000000e+00,
        1.00000000e+00, 0.00000000e+00],
       [4.00000000e+01, 1.42500000e+05, 3.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [2.80000000e+01, 8.39516129e+04, 0.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 1.00000000e+00],
       [2.50000000e+01, 5.00000000e+04, 1.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 1.00000000e+00],
       [2.50000000e+01, 5.75000000e+04, 1.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 1.00000000e+00],
       [3.00000000e+01, 1.12500000e+05, 4.00000000e+00, 0.00000000e+00,
        2.00000000e+00, 0.00000000e+00],
       [4.00000000e+01, 1.42500000e+05, 3.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [4.20000000e+01, 1.00000000e+05, 4.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [5.10000000e+01, 1.40000000e+05, 1.00000000e+00, 0.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [4.90000000e+01, 6.50000000e+04, 2.00000000e+00, 0.00000000e+00,
        1.00000000e+00, 0.00000000e+00],
       [5.10000000e+01, 1.40000000e+05, 1.00000000e+00, 0.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [4.30000000e+01, 8.39516129e+04, 4.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [3.40000000e+01, 4.50000000e+04, 0.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 1.00000000e+00],
```

```
x_test
```

```
array([[2.50000000e+01, 5.50000000e+04, 1.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 1.00000000e+00],
       [4.80000000e+01, 8.39516129e+04, 2.00000000e+00, 0.00000000e+00,
        1.00000000e+00, 0.00000000e+00],
       [2.60000000e+01, 5.50000000e+04, 1.00000000e+00, 0.00000000e+00,
        1.00000000e+00, 0.00000000e+00],
       [3.90000000e+01, 6.25000000e+04, 1.00000000e+00, 0.00000000e+00,
        2.00000000e+00, 0.00000000e+00],
       [3.50000000e+01, 9.00000000e+04, 2.00000000e+00, 0.00000000e+00,
        1.00000000e+00, 0.00000000e+00],
       [2.50000000e+01, 5.50000000e+04, 1.00000000e+00, 1.00000000e+00,
        0.00000000e+00, 1.00000000e+00],
       [4.80000000e+01, 8.25000000e+04, 2.00000000e+00, 0.00000000e+00,
        1.00000000e+00, 0.00000000e+00],
       [4.80000000e+01, 8.25000000e+04, 2.00000000e+00, 0.00000000e+00,
```

```
[4.80000000e+01, 8.25000000e+04, 2.00000000e+00, 0.00000000e+00,
 1.00000000e+00, 0.00000000e+00],
[3.60000000e+01, 9.50000000e+04, 4.00000000e+00, 0.00000000e+00,
 2.00000000e+00, 0.00000000e+00],
[4.60000000e+01, 9.50000000e+04, 3.00000000e+00, 0.00000000e+00,
 1.00000000e+00, 0.00000000e+00],
[3.30000000e+01, 5.00000000e+04, 3.00000000e+00, 1.00000000e+00,
 0.00000000e+00, 1.00000000e+00],
[2.90000000e+01, 2.75000000e+04, 3.00000000e+00, 1.00000000e+00,
 0.00000000e+00, 1.00000000e+00],
[4.20000000e+01, 9.50000000e+04, 4.00000000e+00, 1.00000000e+00,
 0.00000000e+00, 0.00000000e+00],
[5.00000000e+01, 1.60000000e+05, 4.00000000e+00, 0.00000000e+00,
 0.00000000e+00, 0.00000000e+00],
[2.90000000e+01, 2.50000000e+04, 3.00000000e+00, 1.00000000e+00,
 0.00000000e+00, 1.00000000e+00],
[3.00000000e+01, 1.05000000e+05, 4.00000000e+00, 0.00000000e+00,
 2.00000000e+00, 0.00000000e+00],
[3.20000000e+01, 8.50000000e+04, 4.00000000e+00, 1.00000000e+00,
 0.00000000e+00, 1.00000000e+00],
[2.70000000e+01, 3.75000000e+04, 3.00000000e+00, 1.00000000e+00,
 0.00000000e+00, 1.00000000e+00],
[5.20000000e+01, 1.30000000e+05, 3.00000000e+00, 0.00000000e+00,
 0.00000000e+00, 0.00000000e+00],
[4.70000000e+01, 8.50000000e+04, 2.00000000e+00, 0.00000000e+00,
 1.00000000e+00, 0.00000000e+00],
[3.70000000e+01, 7.50000000e+04, 1.00000000e+00, 0.00000000e+00,
 2.00000000e+00, 0.00000000e+00],
[2.80000000e+01, 3.00000000e+04, 0.00000000e+00, 1.00000000e+00,
 0.00000000e+00, 1.00000000e+00],
[2.70000000e+01, 3.50000000e+04, 3.00000000e+00, 1.00000000e+00,
 0.00000000e+00, 1.00000000e+00],
[4.10000000e+01, 1.05000000e+05, 2.00000000e+00, 1.00000000e+00,
 0.00000000e+00, 0.00000000e+00],
[3.50000000e+01, 8.00000000e+04, 2.00000000e+00, 0.00000000e+00,
 1.00000000e+00, 0.00000000e+00],
[3.80000000e+01, 6.75000000e+04, 1.00000000e+00, 0.00000000e+00,
 2.00000000e+00, 0.00000000e+00],
[5.00000000e+01, 1.62500000e+05, 4.00000000e+00, 0.00000000e+00,
 0.00000000e+00, 0.00000000e+00],
[4.40000000e+01, 8.75000000e+04, 4.00000000e+00, 1.00000000e+00,
 0.00000000e+00, 0.00000000e+00],
[3.40000000e+01, 1.05000000e+05, 1.00000000e+00, 0.00000000e+00,
 1.00000000e+00, 1.00000000e+00],
```

```
y_train
```

```
array(['High', 'High', 'High', 'Average', 'Average', 'High', 'High',
       'Average', 'High', 'Low', 'High', 'High', 'High', 'High', 'High',
       'Average', 'High', 'High', 'Low', 'High', 'Average', 'High',
       'High', 'High', 'High', 'High', 'High', 'High', 'Average', 'High',
       'High', 'High', 'Low', 'High', 'Average', 'High', 'Low', 'High',
       'Low', 'High', 'Average', 'Low', 'High', 'Average', 'High',
       'Average', 'High', 'High', 'Average', 'High', 'Low', 'High',
       'Average', 'High', 'High', 'High', 'Average', 'Average', 'High',
       'High', 'High', 'Average', 'High', 'High', 'Average', 'High',
       'High', 'High', 'Low', 'Average', 'High', 'High', 'High', 'High',
       'Average', 'High', 'High', 'High', 'High', 'High', 'High', 'High',
       'High', 'High', 'Average', 'High', 'High', 'High', 'Average',
       'Average', 'High', 'High', 'High', 'High', 'High', 'High',
       'Average', 'High', 'High', 'High', 'High', 'High', 'Average',
       'High', 'High', 'Average', 'High', 'High', 'High', 'High',
       'Average', 'High', 'High', 'High'], dtype=object)
```

```
y_test
```

```
array(['Average', 'High', 'Average', 'High', 'High', 'Average', 'High',
       'High', 'High', 'High', 'Average', 'Low', 'High', 'High', 'Low',
       'High', 'High', 'Low', 'High', 'High', 'High', 'Low', 'Low',
       'High', 'High', 'High', 'High', 'High', 'High', 'Average', 'High',
       'High', 'High', 'High', 'High', 'High', 'High', 'Average', 'High',
       'Low', 'High', 'Average', 'Low', 'Average', 'Average', 'Average',
       'High', 'High', 'High', 'Average'], dtype=object)
```

## Normalisation

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

```
x_train
```

```
array([[ 0.77593283,  0.94515333, -0.73865505, -0.98260737,  2.66391337,
        -0.66561098],
       [ 1.49704432,  1.59465735, -0.73865505, -0.98260737, -0.69792564,
        -0.66561098],
       [ 0.77593283,  0.78277732, -0.73865505, -0.98260737,  2.66391337,
        -0.66561098],
       [-0.5461049 , -1.24692275, -1.45227095,  1.01770049, -0.69792564,
         1.50237907],
       [-1.50758688, -1.49048675, -1.45227095,  1.01770049, -0.69792564,
         1.50237907],
       [ 0.65574758, -0.35385472,  1.40219264,  1.01770049, -0.69792564,
        -0.66561098],
       [ 0.41537709,  0.62040132,  1.40219264,  1.01770049, -0.69792564,
        -0.66561098],
       [-0.90666064, -0.67860673, -0.73865505,  1.01770049, -0.69792564,
         1.50237907],
       [ 0.29519184,  0.78277732, -0.02503915,  1.01770049, -0.69792564,
        -0.66561098],
       [-1.38740164, -1.57167476,  0.68857674,  1.01770049, -0.69792564,
         1.50237907],
       [ 0.77593283,  0.45802531, -0.73865505, -0.98260737,  2.66391337,
        -0.66561098],
       [ 0.17500659,  1.43228134,  0.68857674,  1.01770049, -0.69792564,
        -0.66561098],
       [ 0.29519184,  0.78277732, -0.02503915,  1.01770049, -0.69792564,
        -0.66561098],
       [ 1.01630333, -0.06314929, -0.02503915, -0.98260737,  0.42268736,
        -0.66561098],
       [-0.90666064, -0.59741872, -0.73865505,  1.01770049, -0.69792564,
         1.50237907],
       [ 0.89611808,  0.1332733 ,  0.68857674, -0.98260737,  0.42268736,
        -0.66561098],
       [ 0.17500659,  1.83822136,  0.68857674,  1.01770049, -0.69792564,
        -0.66561098],
       [-1.26721639, -0.06314929, -1.45227095,  1.01770049, -0.69792564,
         1.50237907],
       [-1.62777213, -1.16573474, -0.73865505,  1.01770049, -0.69792564,
         1.50237907],
       [-1.62777213, -0.92217074, -0.73865505,  1.01770049, -0.69792564,
         1.50237907],
       [-1.02684589,  0.86396532,  1.40219264, -0.98260737,  1.54330037,
        -0.66561098],
       [ 0.17500659,  1.83822136,  0.68857674,  1.01770049, -0.69792564,
        -0.66561098],
       [ 0.41537709,  0.45802531,  1.40219264,  1.01770049, -0.69792564,
        -0.66561098],
       [ 1.49704432,  1.75703335, -0.73865505, -0.98260737, -0.69792564,
        -0.66561098],
       [ 1.25667382, -0.67860673, -0.02503915, -0.98260737,  0.42268736,
        -0.66561098],
       [ 1.49704432,  1.75703335, -0.73865505, -0.98260737, -0.69792564,
        -0.66561098],
       [ 0.53556233, -0.06314929,  1.40219264,  1.01770049, -0.69792564,
        -0.66561098],
       [-0.5461049 , -1.32811075, -1.45227095,  1.01770049, -0.69792564,
         1.50237907],
```

```
x_test
```

```
array([[-1.62777213, -1.00335874, -0.73865505,  1.01770049, -0.69792564,
         1.50237907],
       [ 1.13648857, -0.06314929, -0.02503915, -0.98260737,  0.42268736,
        -0.66561098],
       [-1.50758688, -1.00335874, -0.73865505, -0.98260737,  0.42268736,
        -0.66561098],
       [ 0.05482134, -0.75979473, -0.73865505, -0.98260737,  1.54330037,
        -0.66561098],
       [-0.42591965,  0.1332733 , -0.02503915, -0.98260737,  0.42268736,
        -0.66561098],
       [-1.62777213, -1.00335874, -0.73865505,  1.01770049, -0.69792564,
         1.50237907],
       [ 1.13648857, -0.11029071, -0.02503915, -0.98260737,  0.42268736,
        -0.66561098],
       [ 1.13648857, -0.11029071, -0.02503915, -0.98260737,  0.42268736,
        -0.66561098],
       [-0.3057344 ,  0.29564931,  1.40219264, -0.98260737,  1.54330037,
        -0.66561098],
       [ 0.89611808,  0.29564931,  0.68857674, -0.98260737,  0.42268736,
        -0.66561098],
```

```
[-0.66629015, -1.16573474,  0.68857674,  1.01770049, -0.69792564,
  1.50237907],
[-1.14703114, -1.89642677,  0.68857674,  1.01770049, -0.69792564,
  1.50237907],
[ 0.41537709,  0.29564931,  1.40219264,  1.01770049, -0.69792564,
 -0.66561098],
[ 1.37685907,  2.40653738,  1.40219264, -0.98260737, -0.69792564,
 -0.66561098],
[-1.14703114, -1.97761477,  0.68857674,  1.01770049, -0.69792564,
  1.50237907],
[-1.02684589,  0.62040132,  1.40219264, -0.98260737,  1.54330037,
 -0.66561098],
[-0.78647539, -0.02910271,  1.40219264,  1.01770049, -0.69792564,
  1.50237907],
[-1.38740164, -1.57167476,  0.68857674,  1.01770049, -0.69792564,
  1.50237907],
[ 1.61722957,  1.43228134,  0.68857674, -0.98260737, -0.69792564,
 -0.66561098],
[ 1.01630333, -0.02910271, -0.02503915, -0.98260737,  0.42268736,
 -0.66561098],
[-0.18554915, -0.35385472, -0.73865505, -0.98260737,  1.54330037,
 -0.66561098],
[-1.26721639, -1.81523877, -1.45227095,  1.01770049, -0.69792564,
  1.50237907],
[-1.38740164, -1.65286276,  0.68857674,  1.01770049, -0.69792564,
  1.50237907],
[ 0.29519184,  0.62040132, -0.02503915,  1.01770049, -0.69792564,
 -0.66561098],
[-0.42591965, -0.19147871, -0.02503915, -0.98260737,  0.42268736,
 -0.66561098],
[-0.06536391, -0.59741872, -0.73865505, -0.98260737,  1.54330037,
 -0.66561098],
[ 1.37685907,  2.48772538,  1.40219264, -0.98260737, -0.69792564,
 -0.66561098],
[ 0.65574758,  0.0520853 ,  1.40219264,  1.01770049, -0.69792564,
 -0.66561098],
[-0.5461049 ,  0.62040132, -0.73865505, -0.98260737,  0.42268736,
  1.50237907],
```

## Model creation and Performance evaluation

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
knn=KNeighborsClassifier(n_neighbors=7)
naive=GaussianNB()
vector=SVC()
lst_models=[knn,naive,vector]
```

```
for i in lst_models:
  print(i)
  i.fit(x_train,y_train)
y_pred=i.predict(x_test)
print("**********************************************************************")
print(confusion_matrix(y_test,y_pred))
print("**********************************************************************")
print(accuracy_score(y_test,y_pred))
print("**********************************************************************")
print(classification_report(y_test,y_pred))
```

```
KNeighborsClassifier(n_neighbors=7)
**********************************************************************
[[ 8  2  1]
 [ 1 31  0]
 [ 3  0  4]]
**********************************************************************
0.86
**********************************************************************
              precision    recall  f1-score   support

     Average       0.67      0.73      0.70        11
        High       0.94      0.97      0.95        32
         Low       0.80      0.57      0.67         7

    accuracy                           0.86        50
   macro avg       0.80      0.76      0.77        50
weighted avg       0.86      0.86      0.86        50

GaussianNB()
**********************************************************************
```

```
**********************************************************************
[[ 9  2  0]
 [ 1 31  0]
 [ 0  0  7]]
**********************************************************************
0.94
**********************************************************************
              precision    recall  f1-score   support

     Average       0.90      0.82      0.86        11
        High       0.94      0.97      0.95        32
         Low       1.00      1.00      1.00         7

    accuracy                           0.94        50
   macro avg       0.95      0.93      0.94        50
weighted avg       0.94      0.94      0.94        50

SVC()
**********************************************************************
[[ 9  2  0]
 [ 1 31  0]
 [ 3  0  4]]
**********************************************************************
0.88
**********************************************************************
              precision    recall  f1-score   support

     Average       0.69      0.82      0.75        11
        High       0.94      0.97      0.95        32
         Low       1.00      0.57      0.73         7

    accuracy                           0.88        50
   macro avg       0.88      0.79      0.81        50
weighted avg       0.89      0.88      0.88        50
```