Assigment5

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

```
pd.read_csv('Social_Network_Ads.csv')
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

```
pd.read_csv
```

```
<function pandas.io.parsers.readers.read_csv(filepath_or_buffer: 'FilePath | ReadCsvBuffer[bytes] | ReadCsvBuffer[str]', *, sep:
'str | None | lib.NoDefault' = <no_default>, delimiter: 'str | None | lib.NoDefault' = None, header: "int | Sequence[int] | None |
Literal['infer']" = 'infer', names: 'Sequence[Hashable] | None | lib.NoDefault' = <no_default>, index_col: 'IndexLabel |
Literal[False] | None' = None, usecols=None, squeeze: 'bool | None' = None, prefix: 'str | lib.NoDefault' = <no_default>,
mangle_dupe_cols: 'bool' = True, dtype: 'DtypeArg | None' = None, engine: 'CSVEngine | None' = None, converters=None,
true_values=None, false_values=None, skipinitialspace: 'bool' = False, skiprows=None, skipfooter: 'int' = 0, nrows: 'int | None' =
None, na_values=None, keep_default_na: 'bool' = True, na_filter: 'bool' = True, verbose: 'bool' = False, skip_blank_lines: 'bool'
= True, parse_dates=None, infer_datetime_format: 'bool' = False, keep_date_col: 'bool' = False, date_parser=None, dayfirst: 'bool'
= False, cache_dates: 'bool' = True, iterator: 'bool' = False, chunksize: 'int | None' = None, compression: 'CompressionOptions' =
'infer', thousands: 'str | None' = None, decimal: 'str' = '.', lineterminator: 'str | None' = None, quotechar: 'str' = '"',
quoting: 'int' = 0, doublequote: 'bool' = True, escapechar: 'str | None' = None, comment: 'str | None' = None, encoding: 'str |
None' = None, encoding_errors: 'str | None' = 'strict', dialect: 'str | csv.Dialect | None' = None, error_bad_lines: 'bool | None'
= None, warn_bad_lines: 'bool | None' = None, on_bad_lines=None, delim_whitespace: 'bool' = False, low_memory=True, memory_map:
'bool' = False, float_precision: "Literal['high', 'legacy'] | None" = None, storage_options: 'StorageOptions' = None) ->
'DataFrame | TextFileReader'>
```

```
df=pd.read_csv('Social_Network_Ads.csv')
```

```
df.shape
```

```
(400, 5)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

```
df.head()
```

```
User ID  Gender  Age  EstimatedSalary  Purchased
```

```
df.describe()
```

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

```
df.isnull().sum()
```

```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

```
df['Gender'].value_counts()
```

```
Female    204
Male      196
Name: Gender, dtype: int64
```

```
df['Gender'].replace(['Female','Male'],[0,1],inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    int64
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(5)
memory usage: 15.8 KB
```

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(10,5))
```

```
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
```

```
sns.boxplot(df)
```

```
<Axes: >
```


```
df['User ID'].value_counts()
```

```
15624510    1
15767681    1
15589449    1
15791373    1
15688172    1
           ..
15675185    1
15792102    1
15722758    1
15745232    1
15594041    1
Name: User ID, Length: 400, dtype: int64
```

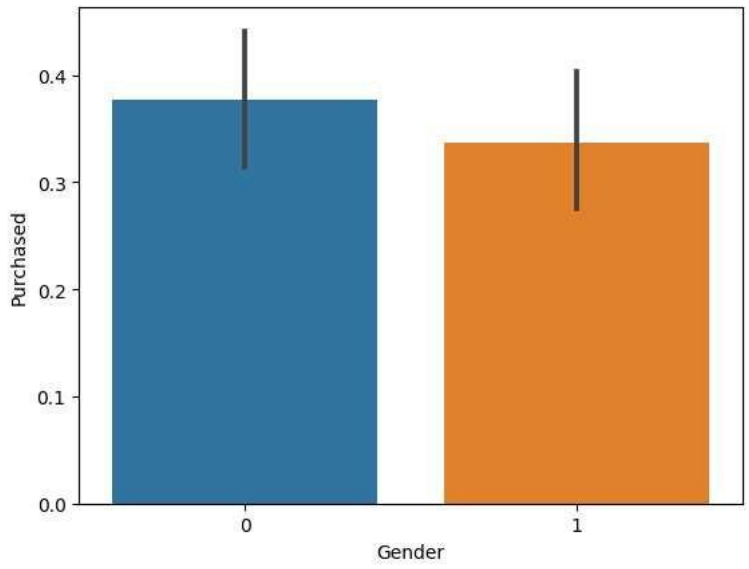
```
df.drop([('User ID')],axis=1,inplace=True)
```

User ID	Gender	Age	EstimatedSalary	Purchased

```
for i in df.columns:
  sns.boxplot(x=i,data=df)
  plt.show()
```

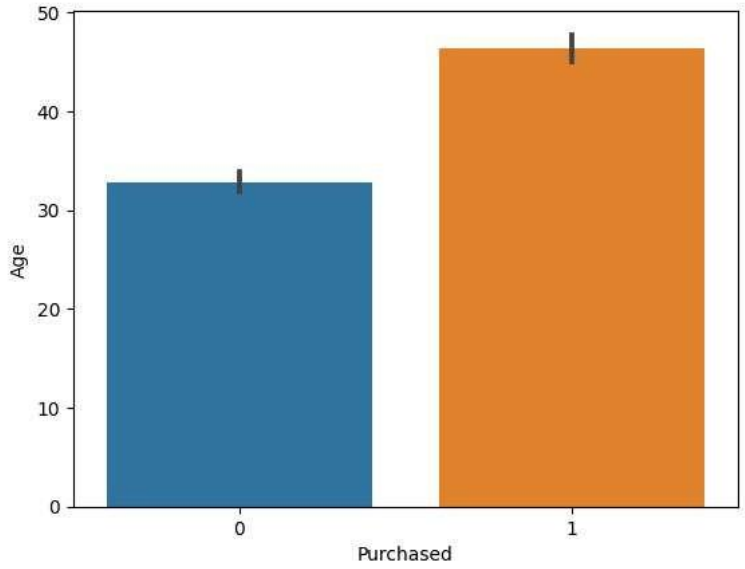
```
sns.barplot(x='Gender',y='Purchased',data=df)
```

```
<Axes: xlabel='Gender', ylabel='Purchased'>
```



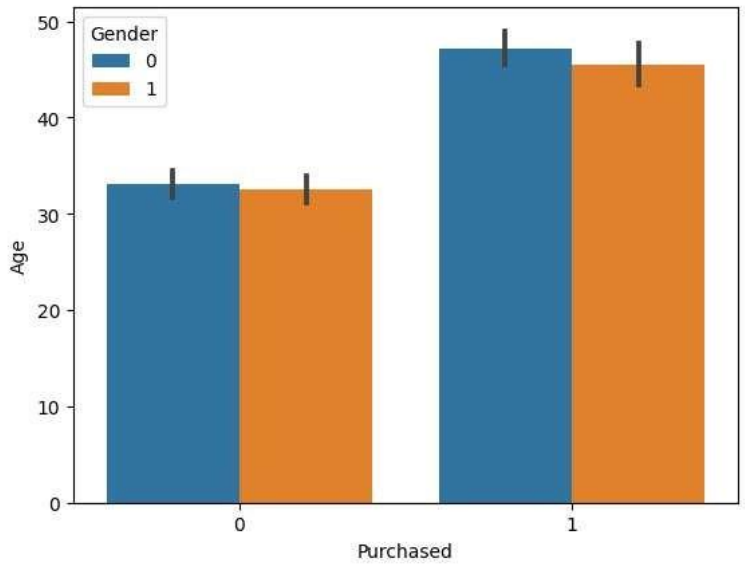

```
sns.barplot(x='Purchased',y='Age',data=df)
```

```
<Axes: xlabel='Purchased', ylabel='Age'>
```



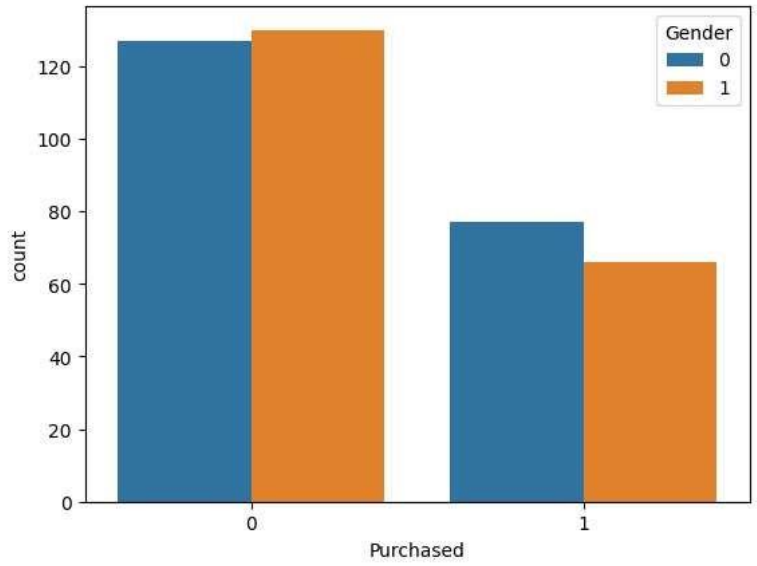

```
sns.barplot(x='Purchased',y='Age',data=df,hue='Gender')
```

```
<Axes: xlabel='Purchased', ylabel='Age'>
```



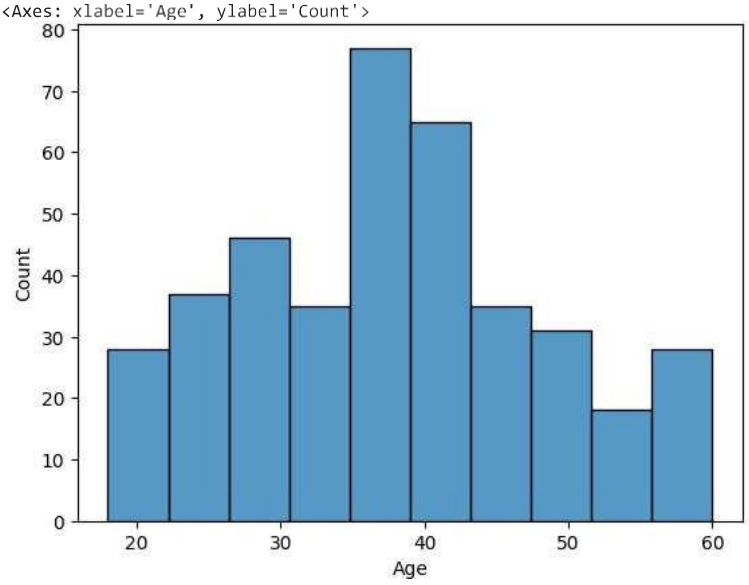

```
sns.countplot(x='Purchased',data=df,hue='Gender')
```

```
<Axes: xlabel='Purchased', ylabel='count'>
```



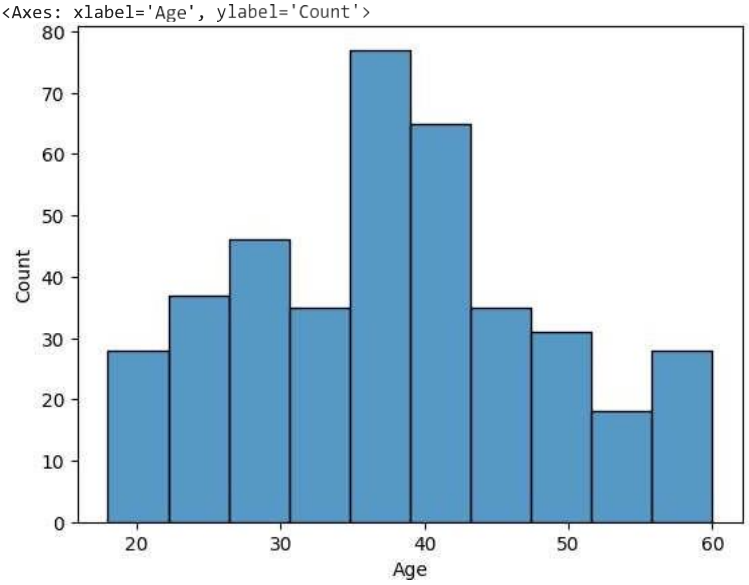

```
sns.histplot(x='Age',data=df)
```

```
<Axes: xlabel='Age', ylabel='Count'>
```



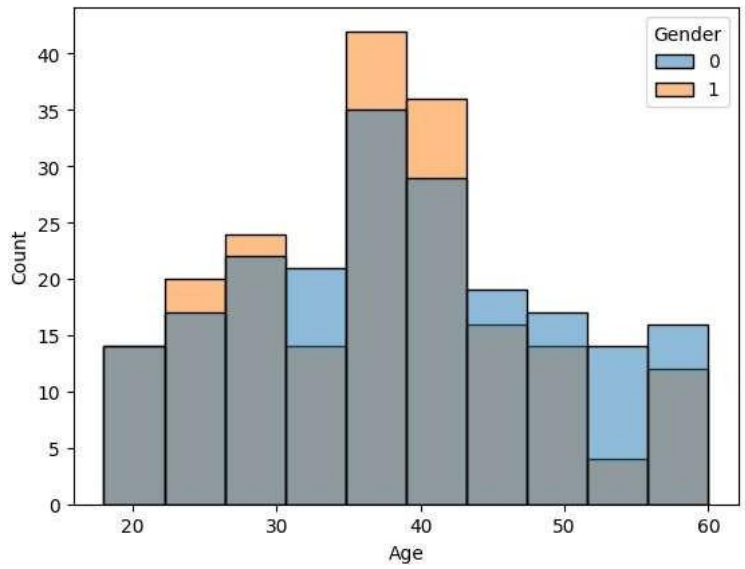

```
sns.histplot(x='Age',data=df,bins=10)
```

```
<Axes: xlabel='Age', ylabel='Count'>
```



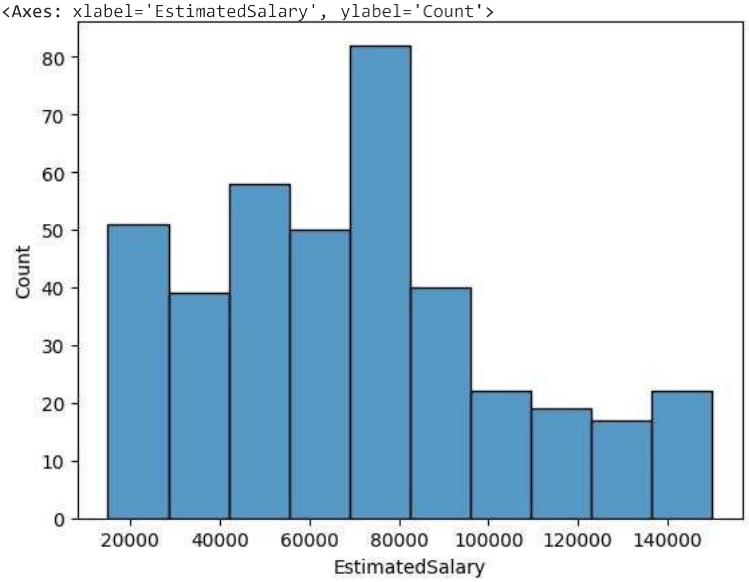

```
sns.histplot(x='Age',data=df,hue='Gender')
```

```
<Axes: xlabel='Age', ylabel='Count'>
```



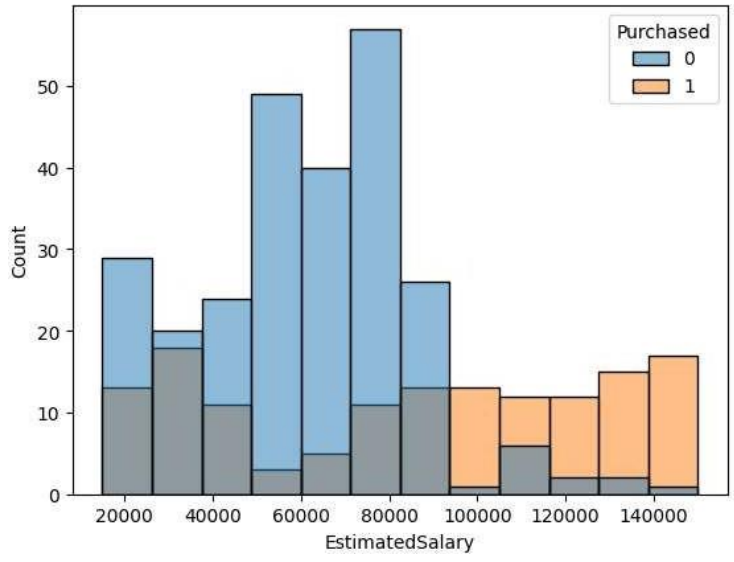

```
sns.histplot(x='EstimatedSalary',data=df,bins=10)
```

```
<Axes: xlabel='EstimatedSalary', ylabel='Count'>
```




```
sns.histplot(x='EstimatedSalary',data=df,hue='Purchased')
```

```
<Axes: xlabel='EstimatedSalary', ylabel='Count'>
```



Building Logistics Regression Model

1. Set x & y
2. Splitting dataset into train_test set
3. Create Logistics Regression
4. Train the model
5. Test the model
6. Evaluate the model

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

x

	Gender	Age	EstimatedSalary
0	1	19	19000
1	1	35	20000
2	0	26	43000
3	0	27	57000
4	1	19	76000
...	...	...	...
395	0	46	41000
396	1	51	23000
397	0	50	20000
398	1	36	33000
399	0	49	36000

400 rows × 3 columns

y

```
0      0
1      0
2      0
3      0
4      0
395    1
396    1
397    1
398    0
399    1
Name: Purchased, Length: 400, dtype: int64
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=1)
```

```
x_train.shape
```

```
(320, 3)
```

```
x_test.shape
```

```
(80, 3)
```

```
y_train.shape
```

```
(320,)
```

```
y_test.shape
```

```
(80,)
```

```
from sklearn.linear_model  import LogisticRegression
```

```
logr=LogisticRegression()
```

```
logr
```

```
LogisticRegression()
```

```
logr.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
y_pred= logr.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix, precision_score, accuracy_score, recall_score
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[48,  0],
       [32,  0]])
```

```
accuracy_score(y_test,y_pred)
```

```
0.6
```

```
precision_score(y_test,y_pred)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined a
  _warn_prf(average, modifier, msg_start, len(result))
0.0
```

```
recall_score(y_test,y_pred)
```

```
0.0
```

with scaling

```
from sklearn.preprocessing import StandardScaler
```

```
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

```
x_train
```

```
array([[-1.        , -0.80330081, -1.19121795],
       [-1.        ,  0.75697997, -1.36859801],
       [ 1.        ,  0.85449752,  1.43991958],
       [ 1.        , -0.51074816, -1.48685138],
       [ 1.        , -1.48592365,  0.37563923],
       [ 1.        , -1.19337101,  0.55301929],
       [ 1.        ,  1.04953262, -1.04340124],
       [ 1.        , -0.21819552, -0.30431766],
       [ 1.        ,  0.95201507, -1.33903467],
       [ 1.        , -1.09585346, -1.07296458],
       [-1.        , -0.51074816,  1.97205975],
       [ 1.        ,  2.21974321, -1.0138379 ],
       [-1.        ,  1.43960282, -1.39816136],
       [-1.        ,  0.07435713, -0.39300769],
       [ 1.        , -1.19337101,  0.64170932],
       [-1.        ,  2.02470811, -0.89558452],
       [ 1.        ,  1.14705017,  0.58258263],
       [-1.        , -0.02316042,  0.2869492 ],
       [-1.        , -0.21819552,  0.25738586],
       [-1.        , -0.31571307, -0.74776781],
       [ 1.        , -1.68095875, -0.57038775],
       [ 1.        ,  0.85449752,  0.58258263],
       [-1.        , -0.60826571, -1.0138379 ],
       [-1.        ,  0.95201507, -1.13209127],
       [ 1.        , -0.21819552, -0.54082441],
       [ 1.        ,  0.17187468,  0.81908937],
       [-1.        , -0.41323061,  1.32166621],
```

```
       [ 1.        ,  1.14705017,  0.52345594],
       [ 1.        ,  0.75697997,  0.31651254],
       [-1.        ,  0.65946243, -0.86602118],
       [ 1.        ,  0.36690978, -0.27475432],
       [-1.        ,  0.46442733, -0.45213438],
       [ 1.        , -0.21819552,  0.13913248],
       [-1.        ,  0.36690978,  0.10956914],
       [-1.        , -0.99833591,  0.81908937],
       [-1.        , -0.70578326,  1.41035623],
       [ 1.        ,  0.36690978, -0.48169772],
       [ 1.        ,  0.36690978, -0.48169772],
       [-1.        , -1.68095875,  0.40520257],
       [-1.        ,  0.85449752, -0.80689449],
       [-1.        , -0.99833591, -1.10252793],
       [-1.        , -0.21819552,  0.0800058 ],
       [ 1.        ,  1.14705017, -1.19121795],
       [ 1.        , -0.21819552,  0.67127266],
       [ 1.        , -0.02316042,  0.19825917],
       [-1.        , -0.51074816,  1.43991958],
       [-1.        , -0.12067797,  0.19825917],
       [ 1.        , -1.68095875,  0.52345594],
       [-1.        ,  0.07435713, -0.54082441],
       [-1.        ,  1.14705017, -0.95471121],
       [ 1.        ,  0.26939223, -0.09737426],
       [ 1.        , -0.02316042,  0.25738586],
       [-1.        ,  2.21974321, -0.65907778],
       [-1.        ,  1.04953262,  2.06074978],
       [ 1.        ,  0.26939223,  0.0800058 ],
       [-1.        , -0.12067797, -0.15650095],
       [ 1.        , -1.09585346,  0.37563923],
       [ 1.        , -0.41323061, -1.10252793],
```

x_test

```
array([[ 1.        , -0.12067797, -1.04340124],
       [-1.        ,  0.17187468, -0.21562763],
       [ 1.        , -0.12067797,  1.46948292],
       [ 1.        ,  0.17187468,  1.58773629],
       [-1.        , -1.09585346,  1.46948292],
       [-1.        ,  0.07435713, -0.09737426],
       [-1.        , -1.68095875, -0.95471121],
       [ 1.        ,  1.14705017,  0.61214597],
       [ 1.        , -0.60826571, -1.48685138],
       [ 1.        ,  1.04953262,  2.14943981],
       [-1.        , -0.31571307,  0.10956914],
       [-1.        ,  0.17187468,  0.13913248],
       [ 1.        , -0.21819552,  0.10956914],
       [-1.        ,  1.04953262,  1.85380638],
       [-1.        ,  1.53712037,  0.40520257],
       [ 1.        ,  1.82967301,  1.91293307],
       [ 1.        ,  2.21974321,  0.43476591],
       [ 1.        , -0.99833591, -0.30431766],
       [-1.        , -0.90081836,  0.55301929],
       [ 1.        ,  2.21974321,  0.99646943],
       [-1.        ,  0.26939223,  0.19825917],
       [-1.        ,  1.24456772,  0.58258263],
       [-1.        ,  0.65946243,  2.09031313],
       [ 1.        ,  0.95201507, -0.74776781],
       [ 1.        ,  0.75697997, -1.25034464],
       [ 1.        , -1.09585346, -1.57554141],
       [-1.        ,  2.02470811, -0.62951444],
       [ 1.        ,  1.14705017,  0.16869583],
       [-1.        ,  1.53712037, -1.0138379 ],
       [-1.        ,  1.43960282,  1.35122955],
       [ 1.        ,  0.17187468, -0.77733115],
       [ 1.        , -1.7784763 ,  0.22782251],
       [-1.        , -1.87599385,  0.52345594],
       [ 1.        ,  1.92719056,  0.16869583],
       [-1.        , -0.99833591,  0.46432926],
       [ 1.        , -0.70578326,  0.34607588],
       [ 1.        , -1.48592365, -1.48685138],
       [-1.        , -0.51074816,  0.52345594],
       [-1.        ,  1.24456772, -1.4277247 ],
       [ 1.        , -1.7784763 , -1.27990798],
       [-1.        ,  0.95201507,  2.23812984],
       [-1.        ,  2.02470811,  0.96690609],
       [ 1.        , -0.31571307,  1.38079289],
       [-1.        , -1.3884061 , -0.06781092],
       [ 1.        ,  1.82967301, -0.24519098],
       [-1.        , -0.60826571,  1.46948292],
       [-1.        ,  1.04953262, -0.98427455],
       [-1.        ,  0.95201507,  1.32166621],
       [-1.        ,  0.17187468,  0.31651254],
       [-1.        ,  1.43960282, -0.89558452],
       [ 1.        , -1.29088856, -0.30431766],
       [ 1.        , -0.02316042, -0.45213438],
       [-1.        ,  0.46442733,  0.34607588],
       [ 1.        ,  0.85449752, -1.19121795],
       [ 1.        ,  0.46442733,  0.13913248],
```

```
       [-1.        , -0.02316042, -0.18606429],
       [ 1.        ,  2.21974321, -0.77733115],
```

```
logr.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
y_pred= logr.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix, precision_score, accuracy_score, recall_score
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[40,  8],
       [ 6, 26]])
```

```
accuracy_score(y_test,y_pred)
```

```
0.825
```

```
precision_score(y_test,y_pred)
```

```
0.7647058823529411
```

```
recall_score(y_test,y_pred)
```

```
0.8125
```