Experiment:-2

Objective:- logistic regression

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd


data_set= pd.read_csv('/content/sample_data/User_Data.csv')


x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values


from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)



from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)


# import the regressor
from sklearn.tree import DecisionTreeRegressor

# create a regressor object
regressor = DecisionTreeRegressor(random_state = 0)

# fit the regressor with X and Y data
regressor.fit(x_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=0)
```

```
y_pred = regressor.predict(x_test)


print('Decision tree',y_pred)
```

```
Decision tree [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 1. 1. 0. 1. 0. 0. 0. 0. 1.
 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0.
 0. 1. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 1. 0. 1. 0. 0. 0. 1. 1. 0.
 0. 1. 0. 0. 0. 0. 1. 1. 1. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1. 0. 0. 0. 1.
 0. 1. 1. 1.]
```

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
print('Decision tree',cm)
```

```
Decision tree [[62  6]
 [ 4 28]]
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score


print('Decision tree',accuracy_score(y_test, y_pred))
```

```
Decision tree 0.9
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)


from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)


from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
classifier.fit(x_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred= classifier.predict(x_test)
print(y_pred)
```

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

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[64  4]
 [ 3 29]]
```

```
from sklearn.metrics import accuracy_score, f1_score
print('KNN',accuracy_score(y_test, y_pred))
```

```
KNN 0.93
```

```
from sklearn import linear_model
logr = linear_model.LogisticRegression()
logr.fit(x_train,y_train);
```

```
y_pred = logr.predict(x_test)
print('logistic', y_pred)
```

```
logistic [0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0
 0 0 1 0 1 1 1 1 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1]
```

```
print('Logistic',accuracy_score(y_test, y_pred));
```

```
Logistic 0.89
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)


from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

```
GaussianNB
GaussianNB()
```

```
y_pred = classifier.predict(x_test)
print('naive bayes', y_pred)
```

```
naive bayes [0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0
 0 0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1]
```

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[65  3]
 [ 7 25]]
```

```
print('naive bayes',accuracy_score(y_test, y_pred));
```

```
naive bayes 0.9
```

```
from sklearn.svm import SVC # "Support vector classifier"
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(x_train, y_train)
```

```
SVC(kernel='linear', random_state=0)
```

```
y_pred= classifier.predict(x_test)
print('SVM',y_pred)
```

```
SVM [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0
 0 0 1 0 1 1 1 1 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1]
```

```
print('SVM',accuracy_score(y_test, y_pred));
```

```
SVM 0.9
```