```
[1] import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
    import sklearn
```

```
    dataset = pd.read_csv(r"/content/Social_Network_Ads (1).csv")
    X = dataset.iloc[:, [1, 2, 3]].values
    y = dataset.iloc[:, -1].values
```

```
[13] dataset.head()
```

	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

```
[4]
    from sklearn.preprocessing import LabelEncoder
    le = LabelEncoder()
    X[:,0] = le.fit_transform(X[:,0])
```

```
[5] 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, random_state = 0)
```

```
[6]
    #feature scaling to the training and test set of independent variables for reducing the size to smaller values
    from sklearn.preprocessing import StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
```

```
[7] #Fitting Decision Tree classifier to the training set
    from sklearn import tree
    from sklearn.tree import DecisionTreeClassifier
    classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
    classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
[10]
    #Predicting the test set result
    y_pred= classifier.predict(X_test)
    y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 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,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1])
```

```
[11] #Creating the Confusion matrix
    from sklearn.metrics import confusion_matrix,accuracy_score
    cm= confusion_matrix(y_test, y_pred)
    cm
```

```
array([[54,  4],
       [ 3, 19]])
```

```
[12]
    ac = accuracy_score(y_test,y_pred)
    ac
```

```
0.9125
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
[2] dataset = pd.read_csv(r"/content/petrol_consumption.csv")
    dataset.head()
```

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol_Consumption
0	9.0	3571	1976	0.525	541
1	9.0	4092	1250	0.572	524
2	9.0	3865	1586	0.580	561
3	7.5	4870	2351	0.529	414
4	8.0	4399	431	0.544	410

```
[3] dataset.describe()
```

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol_Consumption
count	48.000000	48.000000	48.000000	48.000000	48.000000
mean	7.668333	4241.833333	5565.416667	0.570333	576.770833
std	0.950770	573.623768	3491.507166	0.055470	111.885816
min	5.000000	3063.000000	431.000000	0.451000	344.000000
25%	7.000000	3739.000000	3110.250000	0.529750	509.500000
50%	7.500000	4298.000000	4735.500000	0.564500	568.500000
75%	8.125000	4578.750000	7156.000000	0.595250	632.750000
max	10.000000	5342.000000	17782.000000	0.724000	968.000000

```
[4] X = dataset.drop('Petrol_Consumption', axis=1)
    y = dataset['Petrol_Consumption']
```

```
[5] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)
y_pred
```

```
array([547., 414., 574., 554., 574., 554., 648., 649., 414., 464.])
```

```
[8] #compare some of our predicted values with the actual values
    df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
    df
```

	Actual	Predicted
29	534	547.0
4	410	414.0
26	577	574.0
30	571	554.0
32	577	574.0
37	704	554.0
34	487	648.0
40	587	649.0
7	467	414.0
10	580	464.0

```
[42] #To evaluate performance of the regression algorithm
     from sklearn import metrics
     print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
     print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
     print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 53.6
Mean Squared Error: 5726.2
Root Mean Squared Error: 75.6716591598202
```

```
[1] import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
    import sklearn
    import seaborn as sns
```

```
[2] dataset = pd.read_csv(r"/content/diabetes.csv")
    dataset.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
[3] X = dataset.iloc[:,:-1].values
    y = dataset.iloc[:,-1].values
```

```
[4] from sklearn.preprocessing import LabelEncoder
    le = LabelEncoder()
    X[:,0] = le.fit_transform(X[:,0])
```

```
    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, random_state = 0)
```

```
[6] #feature scaling to the training and test set of independent variables for reducing the size to smaller values
    from sklearn.preprocessing import StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
```

```
[7] #Fitting Decision Tree classifier to the training set
    from sklearn import tree
    from sklearn.tree import DecisionTreeClassifier
    classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
    classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

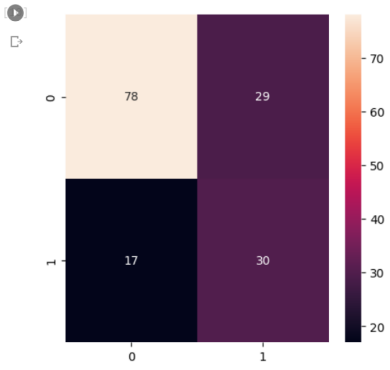
```
[8] #Predicting the test set result
    y_pred= classifier.predict(X_test)
    y_pred
```

```
array([1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0])
```

```
[9] #Creating the Confusion matrix
    from sklearn.metrics import confusion_matrix,accuracy_score
    cm= confusion_matrix(y_test, y_pred)
    cm
```

```
array([[78, 29],
       [17, 30]])
```

```
[10] plt.figure(figsize=(5,5))
     p=sns.heatmap(cm, annot=True)
```



```
ac = accuracy_score(y_test,y_pred)
ac
```

```
0.7012987012987013
```