In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
```

In [4]:

```
#load the data set
df = pd.read_csv(r"C:\Users\geeta\OneDrive\Documents\insurance.csv")

df.head(10)
```

Out[4]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
0	19	0	27.900	0	1	3	16884.92400	1
1	18	1	33.770	1	0	2	1725.55230	1
2	28	1	33.000	3	0	2	4449.46200	0
3	33	1	22.705	0	0	1	21984.47061	0
4	32	1	28.880	0	0	1	3866.85520	1
5	31	0	25.740	0	0	2	3756.62160	0
6	46	0	33.440	1	0	2	8240.58960	1
7	37	0	27.740	3	0	1	7281.50560	0
8	37	1	29.830	2	0	0	6406.41070	0
9	60	0	25.840	0	0	1	28923.13692	0

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             1338 non-null   int64  
 1   sex             1338 non-null   int64  
 2   bmi             1338 non-null   float64
 3   children        1338 non-null   int64  
 4   smoker          1338 non-null   int64  
 5   region          1338 non-null   int64  
 6   charges         1338 non-null   float64
 7   insuranceclaim  1338 non-null   int64  
dtypes: float64(2), int64(6)
memory usage: 83.8 KB
```

In [7]:

```
df.size
```

Out[7]:

```
10704
```

In [8]:

```
df.shape
```

Out[8]:

```
(1338, 8)
```

In [9]:

```
df.describe()
```

Out[9]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	0.505232	30.663397	1.094918	0.204783	1.515695	13270.422265	0.585202
std	14.049960	0.500160	6.098187	1.205493	0.403694	1.104885	12110.011237	0.492871
min	18.000000	0.000000	15.960000	0.000000	0.000000	0.000000	1121.873900	0.000000
25%	27.000000	0.000000	26.296250	0.000000	0.000000	1.000000	4740.287150	0.000000
50%	39.000000	1.000000	30.400000	1.000000	0.000000	2.000000	9382.033000	1.000000
75%	51.000000	1.000000	34.693750	2.000000	0.000000	2.000000	16639.912515	1.000000
max	64.000000	1.000000	53.130000	5.000000	1.000000	3.000000	63770.428010	1.000000

In [10]:

```
# seperate out features and target value from dataset

X = df.drop(['insuranceclaim'],axis = 1).values
y = df['insuranceclaim'].values
```

In [11]:

```
X.shape
```

Out[11]:

```
(1338, 7)
```

In [12]:

```
y.shape
```

Out[12]:

```
(1338,)
```

In [13]:

```
# split the data in training and testing set

X_train, X_test, y_train,y_test = train_test_split(X,y, test_size = 0.25, random_state = 42)
```

In [14]:

```
print("X_train shape : " , X_train.shape)
print("X_test shape : " , X_test.shape)
print("y_train shape : " , y_train.shape)
print("y_test shape : " , y_test.shape)
```

```
X_train shape :  (1003, 7)
X_test shape :  (335, 7)
y_train shape :  (1003,)
y_test shape :  (335,)
```

In [15]:

```
# Model

clf = DecisionTreeClassifier()

# fitting
clf.fit(X_train,y_train)
```

Out[15]:

```
DecisionTreeClassifier()
```

In [16]:

```
# predicting

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

Out[16]:

```
array([0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1], dtype=int64)
```

In [17]:

```
acc = metrics.accuracy_score(y_test,y_pred)

print("Accuracy : ",acc)
```

```
Accuracy :  0.9701492537313433
```

In [18]:

```
y_pred_df = pd.DataFrame(y_pred)
```

In [19]:

```
y_pred_df["Actual"] = y_test
```

In [20]:

```
y_pred_df.columns = ['Predcited', 'Actual']
```

In [21]:

```
y_pred_df
```

Out[21]:

	Predcited	Actual
0	0	1
1	1	1
2	1	1
3	1	1
4	1	1
...	...	...
330	1	1
331	1	1
332	1	1
333	1	1
334	1	1

335 rows × 2 columns

In [ ]: