PROJECT

June 13, 2023

***#PROBLEM STATEMENT*:- TO PREDICT WHICH MODEL IS SUITABLE FOR THE GIVEN DATASET**

IMPORTING THE ESSENTIAL LIBRARIES:-

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
```

LOADING THE DATASET:-

[ ]:
```
df=pd.read_csv(r"/content/insurance.csv")
df
```

[ ]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
…	…	…	…	…	…	…	…
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

[1338 rows x 7 columns]

TO PRINT THE FIRST 5 ROWS OF A DATASET:-

[ ]:
```
df.head()
```

[ ]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

TO PRINT THE LAST 5 ROWS OF A DATASET:-

```
[ ]: df.tail()
```

```
[ ]:       age     sex    bmi  children smoker     region     charges
     1333   50    male  30.97         3     no  northwest  10600.5483
     1334   18  female  31.92         0     no  northeast   2205.9808
     1335   18  female  36.85         0     no  southeast   1629.8335
     1336   21  female  25.80         0     no  southwest   2007.9450
     1337   61  female  29.07         0    yes  northwest  29141.3603
```

TO KNOW THE BASIC INFORMATION OF A DATASET:-

```
[ ]: df.describe()
```

```
[ ]:                 age          bmi     children       charges
     count  1338.000000  1338.000000  1338.000000   1338.000000
     mean     39.207025    30.663397     1.094918  13270.422265
     std      14.049960     6.098187     1.205493  12110.011237
     min      18.000000    15.960000     0.000000   1121.873900
     25%      27.000000    26.296250     0.000000   4740.287150
     50%      39.000000    30.400000     1.000000   9382.033000
     75%      51.000000    34.693750     2.000000  16639.912515
     max      64.000000    53.130000     5.000000  63770.428010
```

TO KNOW THE TYPE OF AN ATTRIBUTE:-

```
[ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

TO KNOW THE SHAPE OF DATSET

```
[ ]: df.shape
```

```
[ ]: (1338, 7)
```

TO CHECK ANY NULL VALUE IS PRESENT OR NOT IN A DATASET:-

```
[ ]: df.isnull().any()
```

```
[ ]: age         False
     sex         False
     bmi         False
     children    False
     smoker      False
     region      False
     charges     False
     dtype: bool
```

SUBSET OF A DATASET

```
[ ]: df=df[['age','bmi','smoker','charges']]
     df
```

```
[ ]:       age     bmi smoker      charges
     0      19  27.900    yes  16884.92400
     1      18  33.770     no   1725.55230
     2      28  33.000     no   4449.46200
     3      33  22.705     no  21984.47061
     4      32  28.880     no   3866.85520
     …   …      …    …           …
     1333   50  30.970     no  10600.54830
     1334   18  31.920     no   2205.98080
     1335   18  36.850     no   1629.83350
     1336   21  25.800     no   2007.94500
     1337   61  29.070    yes  29141.36030

     [1338 rows x 4 columns]
```

CONVERTING THE CATAGORICAL VALUE INTO INTEGER:-

```
[ ]: convert={"smoker":{"yes":1,"no":0}}
     df=df.replace(convert)
     df
```

```
[ ]:       age     bmi  smoker      charges
     0      19  27.900       1  16884.92400
     1      18  33.770       0   1725.55230
     2      28  33.000       0   4449.46200
     3      33  22.705       0  21984.47061
     4      32  28.880       0   3866.85520
     …   …      …    …          …
     1333   50  30.970       0  10600.54830
     1334   18  31.920       0   2205.98080
     1335   18  36.850       0   1629.83350
```

```
1336   21  25.800       0   2007.94500
1337   61  29.070       1  29141.36030

[1338 rows x 4 columns]
```

TO CHECK THE SHAPE OF THE SUBSETTED DATASET:-

[ ]:
```
df.shape
```

[ ]:
```
(1338, 4)
```

EXPLORATARY DATA ANALYSIS(EDA)

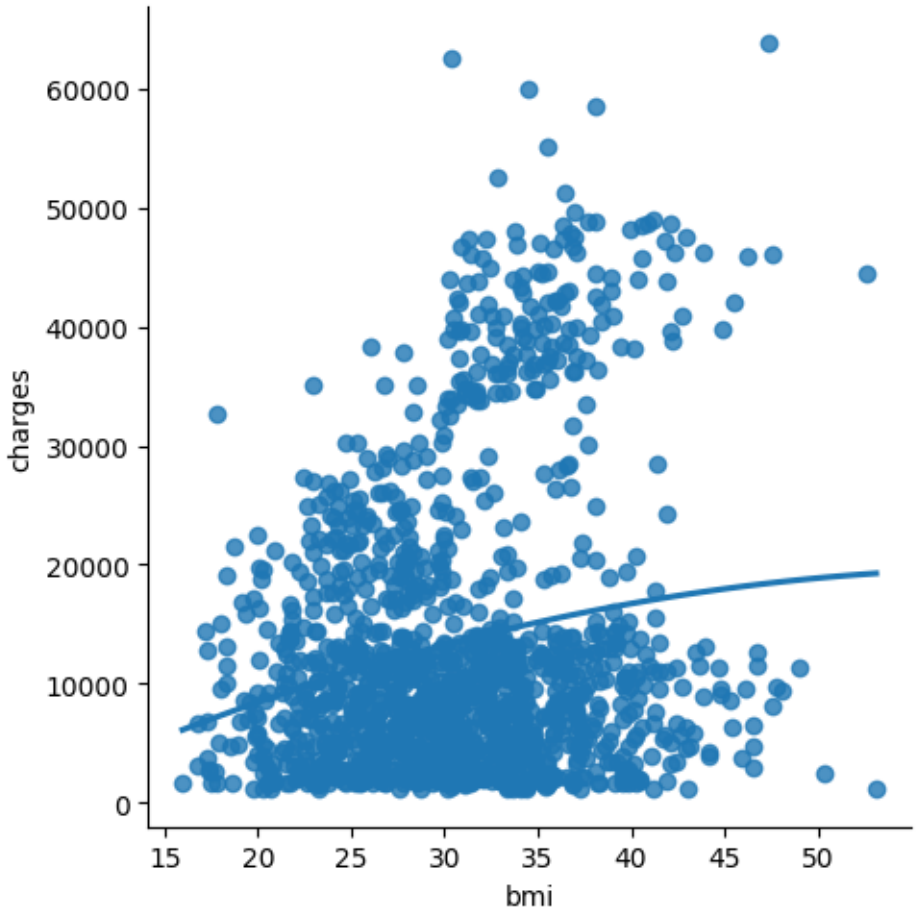
IMPORTING THE LIBRARIES FOR THE VISUVALIZATION

[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

PLOTTING THE GRAPH:-

[ ]:
```
sns.lmplot(x='bmi',y='charges',data=df,order=2,ci=None)
```

[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7f98d28825c0>
```



TO RESHAPE THE DATA(TO MAKE DATA UNIFORM):-

[ ]:
```
x=np.array(df['bmi']).reshape(-1,1)
y=np.array(df['charges']).reshape(-1,1)
```

IMPORTING LIBRARY TO SPLIT TRAINING SET AND TESTING SET

[ ]:
```
from sklearn.model_selection import train_test_split
```

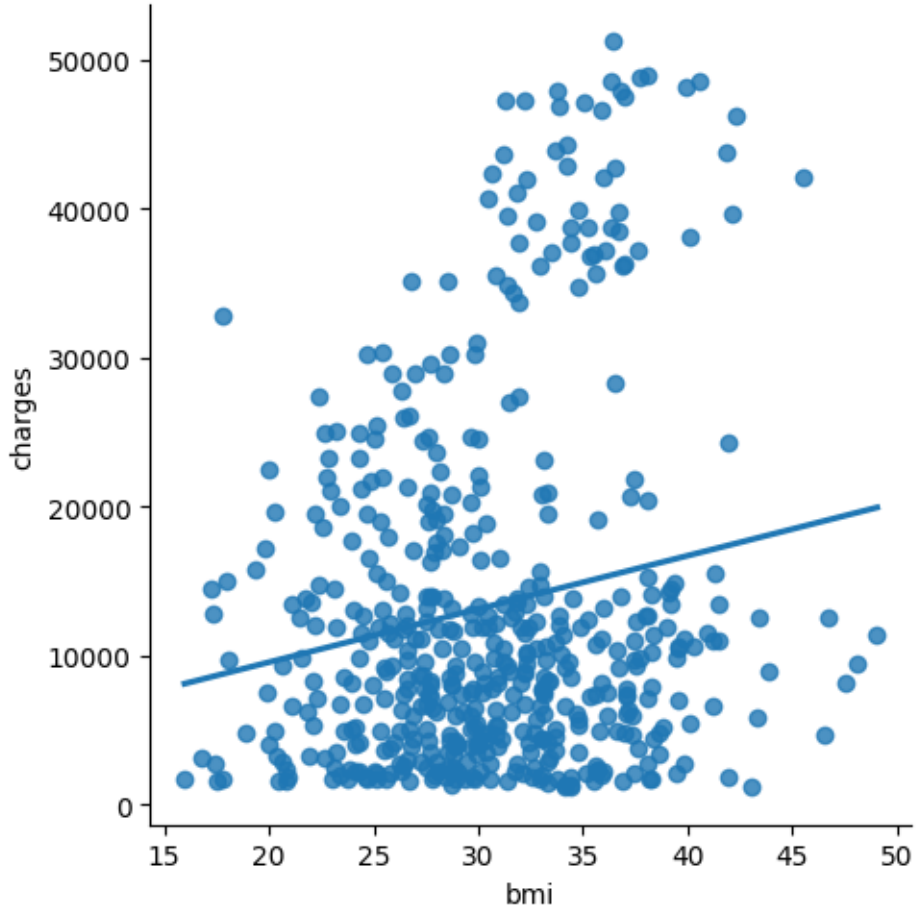
TO CHECK THE REGRESSION SCORE:-

[ ]:
```
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.025339979067911633
```

TO PLOT THE SUBSETTED DATASET:-

[ ]:
```
df500=df[:][:500]
sns.lmplot(x="bmi",y="charges",data=df500,order=1,ci=None)
x=np.array(df500['bmi']).reshape(-1,1)
y=np.array(df500['charges']).reshape(-1,1)
df500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

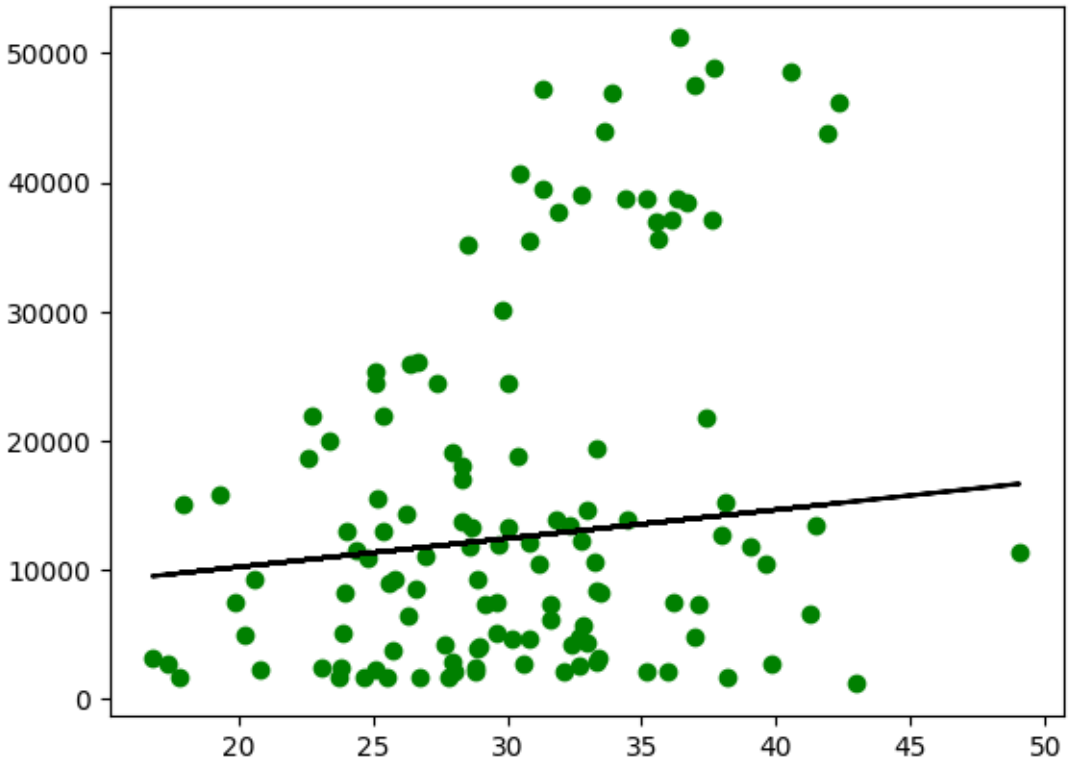


[ ]:
```
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression Score:",regr.score(x_test,y_test))
```

```
Regression Score: 0.004281638573397029
```

TO PLOT THE PREDICTED DATA:-

[ ]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='g')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



#RIDGE MODEL:-

IMPORTING THE PACKAGES:-

```
[ ]: from sklearn.linear_model import Lasso,Ridge
     from sklearn.preprocessing import StandardScaler
```

DEFINING THE TARGET AND FEATURE VECTORS:-

```
[ ]: features= df.columns[0:3]
     target= df.columns[-1]
```

TO FIT THE TARGET AND FEATURE VECTORS:-

```
[ ]: x= df[features].values
     y= df[target].values
     x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
     scaler =StandardScaler()
     x_train=scaler.fit_transform(x_train)
     x_test=scaler.transform(x_test)
```

```
[ ]: ridgeReg=Ridge(alpha=10)
     ridgeReg.fit(x_train,y_train)
     train_score_ridge=ridgeReg.score(x_train,y_train)
```

```
test_score_ridge=ridgeReg.score(x_test,y_test)
```

SCORE OF THE RIDGE MODEL:-

[ ]:
```
print("Ridge Model:-")
print("the train score for ridge model is{}".format(train_score_ridge))
print("the test score for ridge model is{}".format(test_score_ridge))
```

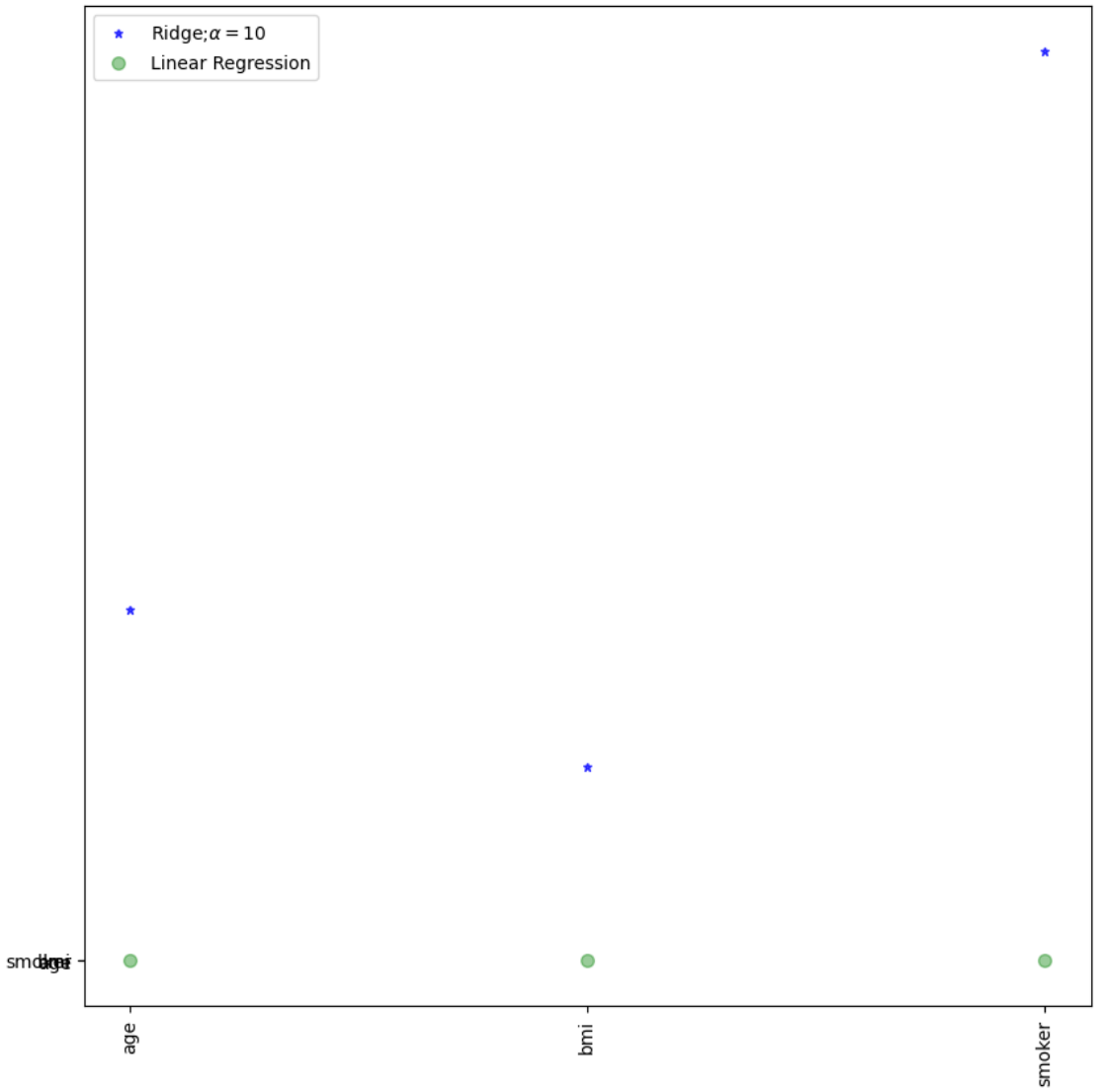
```
Ridge Model:-
the train score for ridge model is0.7277010627441683
the test score for ridge model is0.7865521942258982
```

ASSINGNING LINEAR REGRESSION TO LR:-

[ ]:
```
lr=LinearRegression()
```

PLOTTING THE GRAPH:-

[ ]:
```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.
 ↪7,linestyle='none',marker="*",markersize=5,color='blue',label=r'Ridge;
 ↪$\alpha= 10$',zorder=7)
plt.plot(features,alpha=0.
 ↪4,linestyle='none',marker='o',markersize=7,color="green",label='Linear␣
 ↪Regression')
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



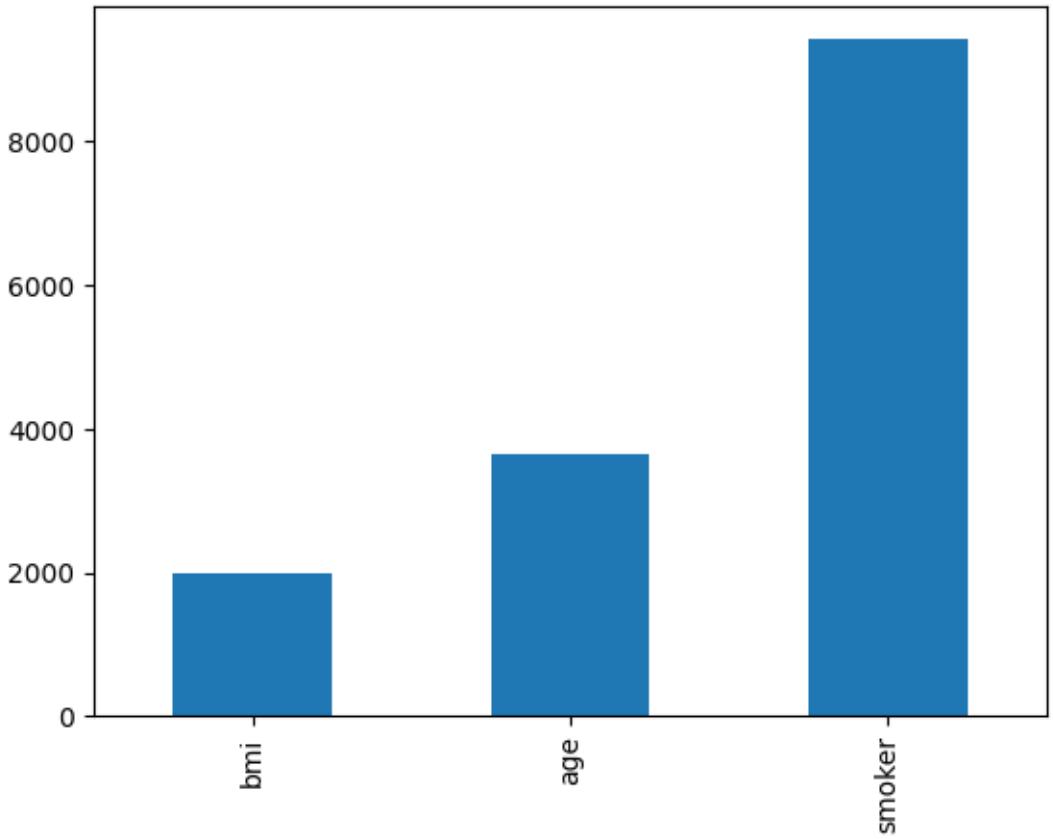
#LASSO MODEL:-

[ ]:
```
print("Lasso Model:-")
lasso=Lasso(alpha=10)
lasso.fit(x_train,y_train)
train_score_ls=lasso.score(x_train,y_train)
test_score_ls=lasso.score(x_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is{}".format(test_score_ls))
```

```
Lasso Model:-
The train score for ls model is 0.7277860363073241
The test score for ls model is0.7872229669132393
```

[ ]:
```
pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

[ ]: <Axes: >



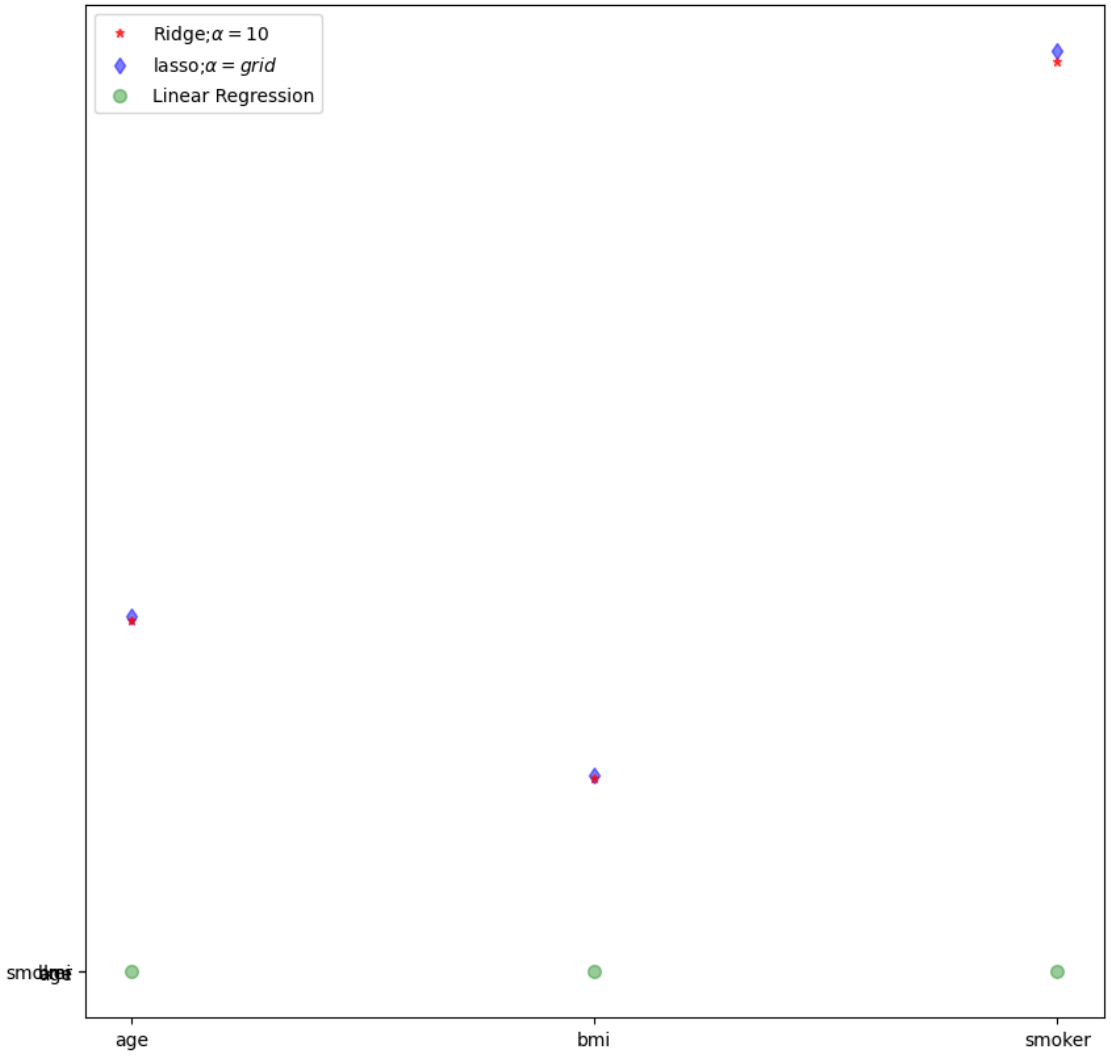
LASSO CV MODEL:-

[ ]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,1,10],random_state=0).
  ↪fit(x_train,y_train)
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

0.7277881381968533
0.7872914671066007

[ ]:
```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.
  ↪7,linestyle='none',marker="*",markersize=5,color='red',label=r'Ridge;
  ↪$\alpha= 10$',zorder=7)
```

```
plt.plot(lasso_cv.coef_,alpha=0.
  ↪5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso;
  ↪$\alpha= grid$')
plt.plot(features,alpha=0.
  ↪4,linestyle='none',marker='o',markersize=7,color="green",label='Linear␣
  ↪Regression')
plt.legend()
plt.show()
```



#ELASTIC NET:-

[ ]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x,y)
```

```
print(regr.coef_)
print(regr.intercept_)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is{}".format(test_score_ls))
```

```
[ 245.83491557  326.12836834 5849.22325507]
-7566.060100878371
The train score for ls model is 0.7277860363073241
The test score for ls model is0.7872229669132393
```

[ ]:
```
y_pred_elastic = regr.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic - y_train)**2)
print(mean_squared_error)
```

```
497661202.9930098
```

CONCLUSION:-

THE LINEAR REGRESSION SCORE IS: -0.014637086277222489

THE RIDGE SCORE IS:-

the train score for ridge model is: 0.7277010627441683

the test score for ridge model is: 0.7865521942258982

The Lasso Score IS:-

The train score for ls model is: 0. 727786036307324

The test score for ls model is:0.78 72229669132395

THE LASSO CV SCORE IS:-

The train score for lasso cv mode l is:0.7277881381968533

The test score for ls model is:0. 7872914671066007

THE ELASTIC NET SCORE IS:-

The train score for ls model is 0.727786036307324

The test score for ls model is0. 7872229669132395

#LOGISTIC REGRESSION:-

IMPORTING THE ESSENTIAL LIBRARIES FOR LOGISTIC REGRESSION:-

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

LOADING THE DATASET:-

```
[ ]: df=pd.read_csv(r"/content/insurance.csv")
     df
```

```
[ ]:       age     sex     bmi  children smoker     region      charges
    0       19  female  27.900         0    yes  southwest  16884.92400
    1       18    male  33.770         1     no  southeast   1725.55230
    2       28    male  33.000         3     no  southeast   4449.46200
    3       33    male  22.705         0     no  northwest  21984.47061
    4       32    male  28.880         0     no  northwest   3866.85520
    …    …      …      …       …     …         …           …
    1333    50    male  30.970         3     no  northwest  10600.54830
    1334    18  female  31.920         0     no  northeast   2205.98080
    1335    18  female  36.850         0     no  southeast   1629.83350
    1336    21  female  25.800         0     no  southwest   2007.94500
    1337    61  female  29.070         0    yes  northwest  29141.36030

    [1338 rows x 7 columns]
```

```
[ ]: pd.set_option('display.max_rows',10000000000)
     pd.set_option('display.max_columns',10000000000)
     pd.set_option('display.width',95)
```

```
[ ]: df.shape
```

```
[ ]: (1338, 7)
```

```
[ ]: convert={"sex":{"male":1,"female":0}}
     df=df.replace(convert)
     df
```

```
[ ]:       age  sex     bmi  children smoker     region       charges
    0       19    0  27.900         0    yes  southwest  16884.924000
    1       18    1  33.770         1     no  southeast   1725.552300
    2       28    1  33.000         3     no  southeast   4449.462000
    3       33    1  22.705         0     no  northwest  21984.470610
    4       32    1  28.880         0     no  northwest   3866.855200
    5       31    0  25.740         0     no  southeast   3756.621600
    6       46    0  33.440         1     no  southeast   8240.589600
    7       37    0  27.740         3     no  northwest   7281.505600
    8       37    1  29.830         2     no  northeast   6406.410700
    9       60    0  25.840         0     no  northwest  28923.136920
    10      25    1  26.220         0     no  northeast   2721.320800
    11      62    0  26.290         0    yes  southeast  27808.725100
    12      23    1  34.400         0     no  southwest   1826.843000
    13      56    0  39.820         0     no  southeast  11090.717800
    14      27    1  42.130         0    yes  southeast  39611.757700
    15      19    1  24.600         1     no  southwest   1837.237000
```

1304	42	1	24.605	2	1	northeast	21259.377950
1305	24	0	27.720	0	0	southeast	2464.618800
1306	29	0	21.850	0	1	northeast	16115.304500
1307	32	1	28.120	4	1	northwest	21472.478800
1308	25	0	30.200	0	1	southwest	33900.653000
1309	41	1	32.200	2	0	southwest	6875.961000
1310	42	1	26.315	1	0	northwest	6940.909850
1311	33	0	26.695	0	0	northwest	4571.413050
1312	34	1	42.900	1	0	southwest	4536.259000
1313	19	0	34.700	2	1	southwest	36397.576000
1314	30	0	23.655	3	1	northwest	18765.875450
1315	18	1	28.310	1	0	northeast	11272.331390
1316	19	0	20.600	0	0	southwest	1731.677000
1317	18	1	53.130	0	0	southeast	1163.462700
1318	35	1	39.710	4	0	northeast	19496.719170
1319	39	0	26.315	2	0	northwest	7201.700850
1320	31	1	31.065	3	0	northwest	5425.023350
1321	62	1	26.695	0	1	northeast	28101.333050
1322	62	1	38.830	0	0	southeast	12981.345700
1323	42	0	40.370	2	1	southeast	43896.376300
1324	31	1	25.935	1	0	northwest	4239.892650
1325	61	1	33.535	0	0	northeast	13143.336650
1326	42	0	32.870	0	0	northeast	7050.021300
1327	51	1	30.030	1	0	southeast	9377.904700
1328	23	0	24.225	2	0	northeast	22395.744240
1329	52	1	38.600	2	0	southwest	10325.206000
1330	57	0	25.740	2	0	southeast	12629.165600
1331	23	0	33.400	0	0	southwest	10795.937330
1332	52	0	44.700	3	0	southwest	11411.685000
1333	50	1	30.970	3	0	northwest	10600.548300
1334	18	0	31.920	0	0	northeast	2205.980800
1335	18	0	36.850	0	0	southeast	1629.833500
1336	21	0	25.800	0	0	southwest	2007.945000
1337	61	0	29.070	0	1	northwest	29141.360300

[ ]:
```
features_matrix=df.iloc[:,0:4]
target_vector=df.iloc[:,-3]
```

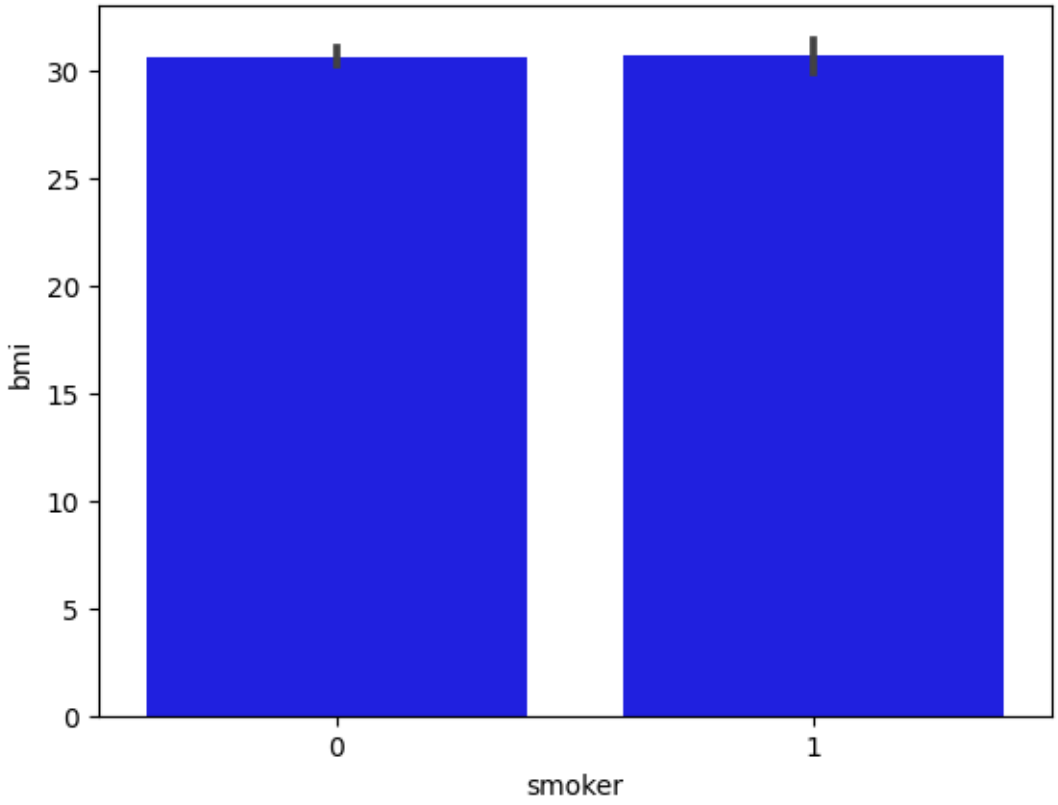
[ ]:
```
print('The Feature Matrix has %d Rows and %d columns(s)'%(features_matrix.
  ↪shape))
print('The Target Matrix has %d Rows and %d columns(s)'%(np.
  ↪array(target_vector).reshape(-1,1).shape))
```

```
The Feature Matrix has 1338 Rows and 4 columns(s)
The Target Matrix has 1338 Rows and 1 columns(s)
```

[ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
sns.barplot(x='smoker',y='bmi',data=df,color='b')
```

[ ]: <Axes: xlabel='smoker', ylabel='bmi'>



[ ]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

[ ]:
```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.
↪0,fit_intercept=True,intercept_scaling=1,class_weight=None,random_state=None,solver='lbfgs'
```

[ ]:
```
Logistic_Regression_Model=algorithm.
↪fit(features_matrix_standardized,target_vector)
```

[ ]:
```
observation=[[1,0,0.99539,-0.05889]]
```

[ ]:
```
predictions=Logistic_Regression_Model.predict(observation)
print('The model predicted the observation to belong to class %s'%(predictions))
```

```
The model predicted the observation to belong to class [0]
```

```
[ ]: print('The algoritham was trained to predict one of the two classes:
       ↪%s'%(algorithm.classes_))
```

```
The algoritham was trained to predict one of the two classes:[0 1]
```

```
[ ]: print(" " "The Model says the probability of the observation we passed␣
       ↪belonging to class['b'] Is %s" " "%(algorithm.
       ↪predict_proba(observation)[0][0]))
     print()
```

```
 The Model says the probability of the observation we passed belonging to
class['b'] Is 0.8057078436306042
```

```
[ ]: print(" " "The Model says the probability of the observation we passed␣
       ↪belonging to class['g'] Is %s " " "%(algorithm.
       ↪predict_proba(observation)[0][1]))
```

```
 The Model says the probability of the observation we passed belonging to
class['g'] Is 0.19429215636939584
```

```
[ ]: x=np.array(df['bmi']).reshape(-1,1)
     y=np.array(df['smoker']).reshape(-1,1)
```

```
[ ]: log=LogisticRegression()
     log.fit(x,y)
     print("Logistic Regression Score:",log.score(x,y))
```

```
Logistic Regression Score: 0.7952167414050823

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

CONCLUSION:-

THE SCORE OF LOGISTIC REGRESSION IS : 0.7952167414050823

#DESICION TREE:-

```
[ ]: import numpy as np
     import pandas as pd
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.tree import DecisionTreeClassifier
```

```
[ ]: df=pd.read_csv(r"/content/insurance.csv")
     df
```

[ ]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.924000
1	18	male	33.770	1	no	southeast	1725.552300
2	28	male	33.000	3	no	southeast	4449.462000
3	33	male	22.705	0	no	northwest	21984.470610
4	32	male	28.880	0	no	northwest	3866.855200
5	31	female	25.740	0	no	southeast	3756.621600
6	46	female	33.440	1	no	southeast	8240.589600
7	37	female	27.740	3	no	northwest	7281.505600
8	37	male	29.830	2	no	northeast	6406.410700
9	60	female	25.840	0	no	northwest	28923.136920
10	25	male	26.220	0	no	northeast	2721.320800
11	62	female	26.290	0	yes	southeast	27808.725100
12	23	male	34.400	0	no	southwest	1826.843000
13	56	female	39.820	0	no	southeast	11090.717800
14	27	male	42.130	0	yes	southeast	39611.757700
15	19	male	24.600	1	no	southwest	1837.237000
16	52	female	30.780	1	no	northeast	10797.336200
17	23	male	23.845	0	no	northeast	2395.171550
18	56	male	40.300	0	no	southwest	10602.385000
19	30	male	35.300	0	yes	southwest	36837.467000
20	60	female	36.005	0	no	northeast	13228.846950
21	30	female	32.400	1	no	southwest	4149.736000
22	18	male	34.100	0	no	southeast	1137.011000
23	34	female	31.920	1	yes	northeast	37701.876800
24	37	male	28.025	2	no	northwest	6203.901750
25	59	female	27.720	3	no	southeast	14001.133800
26	63	female	23.085	0	no	northeast	14451.835150
27	55	female	32.775	2	no	northwest	12268.632250
28	23	male	17.385	1	no	northwest	2775.192150
29	31	male	36.300	2	yes	southwest	38711.000000
30	22	male	35.600	0	yes	southwest	35585.576000
31	18	female	26.315	0	no	northeast	2198.189850
32	19	female	28.600	5	no	southwest	4687.797000
33	63	male	28.310	0	no	northwest	13770.097900
34	28	male	36.400	1	yes	southwest	51194.559140
35	19	male	20.425	0	no	northwest	1625.433750
36	62	female	32.965	3	no	northwest	15612.193350
37	26	male	20.800	0	no	southwest	2302.300000
38	35	male	36.670	1	yes	northeast	39774.276300
39	60	male	39.900	0	yes	southwest	48173.361000
40	24	female	26.600	0	no	northeast	3046.062000
41	31	female	36.630	2	no	southeast	4949.758700
42	41	male	21.780	1	no	southeast	6272.477200
43	37	female	30.800	2	no	southeast	6313.759000
44	38	male	37.050	1	no	northeast	6079.671500
45	55	male	37.300	0	no	southwest	20630.283510

1315	18	male	28.310	1	no	northeast	11272.331390
1316	19	female	20.600	0	no	southwest	1731.677000
1317	18	male	53.130	0	no	southeast	1163.462700
1318	35	male	39.710	4	no	northeast	19496.719170
1319	39	female	26.315	2	no	northwest	7201.700850
1320	31	male	31.065	3	no	northwest	5425.023350
1321	62	male	26.695	0	yes	northeast	28101.333050
1322	62	male	38.830	0	no	southeast	12981.345700
1323	42	female	40.370	2	yes	southeast	43896.376300
1324	31	male	25.935	1	no	northwest	4239.892650
1325	61	male	33.535	0	no	northeast	13143.336650
1326	42	female	32.870	0	no	northeast	7050.021300
1327	51	male	30.030	1	no	southeast	9377.904700
1328	23	female	24.225	2	no	northeast	22395.744240
1329	52	male	38.600	2	no	southwest	10325.206000
1330	57	female	25.740	2	no	southeast	12629.165600
1331	23	female	33.400	0	no	southwest	10795.937330
1332	52	female	44.700	3	no	southwest	11411.685000
1333	50	male	30.970	3	no	northwest	10600.548300
1334	18	female	31.920	0	no	northeast	2205.980800
1335	18	female	36.850	0	no	southeast	1629.833500
1336	21	female	25.800	0	no	southwest	2007.945000
1337	61	female	29.070	0	yes	northwest	29141.360300

[ ]:
```
convert={"sex":{"male":1,"female":0}}
df=df.replace(convert)
df
```

[ ]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	yes	southwest	16884.924000
1	18	1	33.770	1	no	southeast	1725.552300
2	28	1	33.000	3	no	southeast	4449.462000
3	33	1	22.705	0	no	northwest	21984.470610
4	32	1	28.880	0	no	northwest	3866.855200
5	31	0	25.740	0	no	southeast	3756.621600
6	46	0	33.440	1	no	southeast	8240.589600
7	37	0	27.740	3	no	northwest	7281.505600
8	37	1	29.830	2	no	northeast	6406.410700
9	60	0	25.840	0	no	northwest	28923.136920
10	25	1	26.220	0	no	northeast	2721.320800
11	62	0	26.290	0	yes	southeast	27808.725100
12	23	1	34.400	0	no	southwest	1826.843000
13	56	0	39.820	0	no	southeast	11090.717800
14	27	1	42.130	0	yes	southeast	39611.757700
15	19	1	24.600	1	no	southwest	1837.237000
16	52	0	30.780	1	no	northeast	10797.336200
17	23	1	23.845	0	no	northeast	2395.171550

```
1334   18    0  31.920         0     no  northeast   2205.980800
1335   18    0  36.850         0     no  southeast   1629.833500
1336   21    0  25.800         0     no  southwest   2007.945000
1337   61    0  29.070         0    yes  northwest  29141.360300
```

[ ]:
```
convert={'smoker':{"yes":1,"no":0}}
df=df.replace(convert)
df
```

[ ]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	southwest	16884.924000
1	18	1	33.770	1	0	southeast	1725.552300
2	28	1	33.000	3	0	southeast	4449.462000
3	33	1	22.705	0	0	northwest	21984.470610
4	32	1	28.880	0	0	northwest	3866.855200
5	31	0	25.740	0	0	southeast	3756.621600
6	46	0	33.440	1	0	southeast	8240.589600
7	37	0	27.740	3	0	northwest	7281.505600
8	37	1	29.830	2	0	northeast	6406.410700
9	60	0	25.840	0	0	northwest	28923.136920
10	25	1	26.220	0	0	northeast	2721.320800
11	62	0	26.290	0	1	southeast	27808.725100
12	23	1	34.400	0	0	southwest	1826.843000
13	56	0	39.820	0	0	southeast	11090.717800
14	27	1	42.130	0	1	southeast	39611.757700
15	19	1	24.600	1	0	southwest	1837.237000
16	52	0	30.780	1	0	northeast	10797.336200
17	23	1	23.845	0	0	northeast	2395.171550
18	56	1	40.300	0	0	southwest	10602.385000
19	30	1	35.300	0	1	southwest	36837.467000
20	60	0	36.005	0	0	northeast	13228.846950
21	30	0	32.400	1	0	southwest	4149.736000
22	18	1	34.100	0	0	southeast	1137.011000
23	34	0	31.920	1	1	northeast	37701.876800
24	37	1	28.025	2	0	northwest	6203.901750
25	59	0	27.720	3	0	southeast	14001.133800
26	63	0	23.085	0	0	northeast	14451.835150
27	55	0	32.775	2	0	northwest	12268.632250
28	23	1	17.385	1	0	northwest	2775.192150
29	31	1	36.300	2	1	southwest	38711.000000
30	22	1	35.600	0	1	southwest	35585.576000
31	18	0	26.315	0	0	northeast	2198.189850
32	19	0	28.600	5	0	southwest	4687.797000
33	63	1	28.310	0	0	northwest	13770.097900
34	28	1	36.400	1	1	southwest	51194.559140
35	19	1	20.425	0	0	northwest	1625.433750
36	62	0	32.965	3	0	northwest	15612.193350

1306	29	0	21.850	0	1	northeast	16115.304500
1307	32	1	28.120	4	1	northwest	21472.478800
1308	25	0	30.200	0	1	southwest	33900.653000
1309	41	1	32.200	2	0	southwest	6875.961000
1310	42	1	26.315	1	0	northwest	6940.909850
1311	33	0	26.695	0	0	northwest	4571.413050
1312	34	1	42.900	1	0	southwest	4536.259000
1313	19	0	34.700	2	1	southwest	36397.576000
1314	30	0	23.655	3	1	northwest	18765.875450
1315	18	1	28.310	1	0	northeast	11272.331390
1316	19	0	20.600	0	0	southwest	1731.677000
1317	18	1	53.130	0	0	southeast	1163.462700
1318	35	1	39.710	4	0	northeast	19496.719170
1319	39	0	26.315	2	0	northwest	7201.700850
1320	31	1	31.065	3	0	northwest	5425.023350
1321	62	1	26.695	0	1	northeast	28101.333050
1322	62	1	38.830	0	0	southeast	12981.345700
1323	42	0	40.370	2	1	southeast	43896.376300
1324	31	1	25.935	1	0	northwest	4239.892650
1325	61	1	33.535	0	0	northeast	13143.336650
1326	42	0	32.870	0	0	northeast	7050.021300
1327	51	1	30.030	1	0	southeast	9377.904700
1328	23	0	24.225	2	0	northeast	22395.744240
1329	52	1	38.600	2	0	southwest	10325.206000
1330	57	0	25.740	2	0	southeast	12629.165600
1331	23	0	33.400	0	0	southwest	10795.937330
1332	52	0	44.700	3	0	southwest	11411.685000
1333	50	1	30.970	3	0	northwest	10600.548300
1334	18	0	31.920	0	0	northeast	2205.980800
1335	18	0	36.850	0	0	southeast	1629.833500
1336	21	0	25.800	0	0	southwest	2007.945000
1337	61	0	29.070	0	1	northwest	29141.360300

[ ]:
```
x=["children","age"]
y=["0","1"]
all_inputs=df[x]
all_classes=df["smoker"]
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.
↪03)
```

[ ]:
```
clf=DecisionTreeClassifier(random_state=0)
```

[ ]:
```
clf.fit(x_train,y_train)
```

[ ]:
```
DecisionTreeClassifier(random_state=0)
```

[ ]:
```
score=clf.score(x_test,y_test)
print(score)
```

0.7804878048780488

CONCLUSION:-

THE SCORE OF THE DECISION TREE IS : 1.0

1 RANDOM FOREST:-

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

[ ]:
```
df=pd.read_csv(r"/content/insurance.csv")
df
```

[ ]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.924000
1	18	male	33.770	1	no	southeast	1725.552300
2	28	male	33.000	3	no	southeast	4449.462000
3	33	male	22.705	0	no	northwest	21984.470610
4	32	male	28.880	0	no	northwest	3866.855200
5	31	female	25.740	0	no	southeast	3756.621600
6	46	female	33.440	1	no	southeast	8240.589600
7	37	female	27.740	3	no	northwest	7281.505600
8	37	male	29.830	2	no	northeast	6406.410700
9	60	female	25.840	0	no	northwest	28923.136920
10	25	male	26.220	0	no	northeast	2721.320800
11	62	female	26.290	0	yes	southeast	27808.725100
12	23	male	34.400	0	no	southwest	1826.843000
13	56	female	39.820	0	no	southeast	11090.717800
14	27	male	42.130	0	yes	southeast	39611.757700
15	19	male	24.600	1	no	southwest	1837.237000
16	52	female	30.780	1	no	northeast	10797.336200
17	23	male	23.845	0	no	northeast	2395.171550
18	56	male	40.300	0	no	southwest	10602.385000
19	30	male	35.300	0	yes	southwest	36837.467000
20	60	female	36.005	0	no	northeast	13228.846950
21	30	female	32.400	1	no	southwest	4149.736000
22	18	male	34.100	0	no	southeast	1137.011000
23	34	female	31.920	1	yes	northeast	37701.876800
24	37	male	28.025	2	no	northwest	6203.901750
25	59	female	27.720	3	no	southeast	14001.133800
26	63	female	23.085	0	no	northeast	14451.835150
27	55	female	32.775	2	no	northwest	12268.632250

```
1297   28  female  26.510         2     no  southeast   4340.440900
1298   33    male  27.455         2     no  northwest   5261.469450
1299   19  female  25.745         1     no  northwest   2710.828550
1300   45    male  30.360         0    yes  southeast  62592.873090
1301   62    male  30.875         3    yes  northwest  46718.163250
1302   25  female  20.800         1     no  southwest   3208.787000
1303   43    male  27.800         0    yes  southwest  37829.724200
1304   42    male  24.605         2    yes  northeast  21259.377950
1305   24  female  27.720         0     no  southeast   2464.618800
1306   29  female  21.850         0    yes  northeast  16115.304500
1307   32    male  28.120         4    yes  northwest  21472.478800
1308   25  female  30.200         0    yes  southwest  33900.653000
1309   41    male  32.200         2     no  southwest   6875.961000
1310   42    male  26.315         1     no  northwest   6940.909850
1311   33  female  26.695         0     no  northwest   4571.413050
1312   34    male  42.900         1     no  southwest   4536.259000
1313   19  female  34.700         2    yes  southwest  36397.576000
1314   30  female  23.655         3    yes  northwest  18765.875450
1315   18    male  28.310         1     no  northeast  11272.331390
1316   19  female  20.600         0     no  southwest   1731.677000
1317   18    male  53.130         0     no  southeast   1163.462700
1318   35    male  39.710         4     no  northeast  19496.719170
1319   39  female  26.315         2     no  northwest   7201.700850
1320   31    male  31.065         3     no  northwest   5425.023350
1321   62    male  26.695         0    yes  northeast  28101.333050
1322   62    male  38.830         0     no  southeast  12981.345700
1323   42  female  40.370         2    yes  southeast  43896.376300
1324   31    male  25.935         1     no  northwest   4239.892650
1325   61    male  33.535         0     no  northeast  13143.336650
1326   42  female  32.870         0     no  northeast   7050.021300
1327   51    male  30.030         1     no  southeast   9377.904700
1328   23  female  24.225         2     no  northeast  22395.744240
1329   52    male  38.600         2     no  southwest  10325.206000
1330   57  female  25.740         2     no  southeast  12629.165600
1331   23  female  33.400         0     no  southwest  10795.937330
1332   52  female  44.700         3     no  southwest  11411.685000
1333   50    male  30.970         3     no  northwest  10600.548300
1334   18  female  31.920         0     no  northeast   2205.980800
1335   18  female  36.850         0     no  southeast   1629.833500
1336   21  female  25.800         0     no  southwest   2007.945000
1337   61  female  29.070         0    yes  northwest  29141.360300
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
```

```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

[ ]:
```
T={"smoker":{'yes':1,'no':0}}
df=df.replace(T)
df
```

[ ]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	1	southwest	16884.924000
1	18	male	33.770	1	0	southeast	1725.552300
2	28	male	33.000	3	0	southeast	4449.462000
3	33	male	22.705	0	0	northwest	21984.470610
4	32	male	28.880	0	0	northwest	3866.855200
5	31	female	25.740	0	0	southeast	3756.621600
6	46	female	33.440	1	0	southeast	8240.589600
7	37	female	27.740	3	0	northwest	7281.505600
8	37	male	29.830	2	0	northeast	6406.410700
9	60	female	25.840	0	0	northwest	28923.136920
10	25	male	26.220	0	0	northeast	2721.320800
11	62	female	26.290	0	1	southeast	27808.725100
12	23	male	34.400	0	0	southwest	1826.843000
13	56	female	39.820	0	0	southeast	11090.717800
14	27	male	42.130	0	1	southeast	39611.757700
15	19	male	24.600	1	0	southwest	1837.237000
16	52	female	30.780	1	0	northeast	10797.336200
17	23	male	23.845	0	0	northeast	2395.171550
18	56	male	40.300	0	0	southwest	10602.385000
19	30	male	35.300	0	1	southwest	36837.467000
20	60	female	36.005	0	0	northeast	13228.846950
21	30	female	32.400	1	0	southwest	4149.736000
22	18	male	34.100	0	0	southeast	1137.011000
23	34	female	31.920	1	1	northeast	37701.876800
24	37	male	28.025	2	0	northwest	6203.901750
25	59	female	27.720	3	0	southeast	14001.133800
26	63	female	23.085	0	0	northeast	14451.835150
27	55	female	32.775	2	0	northwest	12268.632250
28	23	male	17.385	1	0	northwest	2775.192150
29	31	male	36.300	2	1	southwest	38711.000000

1299	19	female	25.745	1	0	northwest	2710.828550
1300	45	male	30.360	0	1	southeast	62592.873090
1301	62	male	30.875	3	1	northwest	46718.163250
1302	25	female	20.800	1	0	southwest	3208.787000
1303	43	male	27.800	0	1	southwest	37829.724200
1304	42	male	24.605	2	1	northeast	21259.377950
1305	24	female	27.720	0	0	southeast	2464.618800
1306	29	female	21.850	0	1	northeast	16115.304500
1307	32	male	28.120	4	1	northwest	21472.478800
1308	25	female	30.200	0	1	southwest	33900.653000
1309	41	male	32.200	2	0	southwest	6875.961000
1310	42	male	26.315	1	0	northwest	6940.909850
1311	33	female	26.695	0	0	northwest	4571.413050
1312	34	male	42.900	1	0	southwest	4536.259000
1313	19	female	34.700	2	1	southwest	36397.576000
1314	30	female	23.655	3	1	northwest	18765.875450
1315	18	male	28.310	1	0	northeast	11272.331390
1316	19	female	20.600	0	0	southwest	1731.677000
1317	18	male	53.130	0	0	southeast	1163.462700
1318	35	male	39.710	4	0	northeast	19496.719170
1319	39	female	26.315	2	0	northwest	7201.700850
1320	31	male	31.065	3	0	northwest	5425.023350
1321	62	male	26.695	0	1	northeast	28101.333050
1322	62	male	38.830	0	0	southeast	12981.345700
1323	42	female	40.370	2	1	southeast	43896.376300
1324	31	male	25.935	1	0	northwest	4239.892650
1325	61	male	33.535	0	0	northeast	13143.336650
1326	42	female	32.870	0	0	northeast	7050.021300
1327	51	male	30.030	1	0	southeast	9377.904700
1328	23	female	24.225	2	0	northeast	22395.744240
1329	52	male	38.600	2	0	southwest	10325.206000
1330	57	female	25.740	2	0	southeast	12629.165600
1331	23	female	33.400	0	0	southwest	10795.937330
1332	52	female	44.700	3	0	southwest	11411.685000
1333	50	male	30.970	3	0	northwest	10600.548300
1334	18	female	31.920	0	0	northeast	2205.980800
1335	18	female	36.850	0	0	southeast	1629.833500
1336	21	female	25.800	0	0	southwest	2007.945000
1337	61	female	29.070	0	1	northwest	29141.360300

[ ]:
```
T={"sex":{'male':1,'female':0}}
df=df.replace(T)
df
```

[ ]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	southwest	16884.924000
1	18	1	33.770	1	0	southeast	1725.552300

2	28	1	33.000	3	0	southeast	4449.462000
3	33	1	22.705	0	0	northwest	21984.470610
4	32	1	28.880	0	0	northwest	3866.855200
5	31	0	25.740	0	0	southeast	3756.621600
6	46	0	33.440	1	0	southeast	8240.589600
7	37	0	27.740	3	0	northwest	7281.505600
8	37	1	29.830	2	0	northeast	6406.410700
9	60	0	25.840	0	0	northwest	28923.136920
10	25	1	26.220	0	0	northeast	2721.320800
11	62	0	26.290	0	1	southeast	27808.725100
12	23	1	34.400	0	0	southwest	1826.843000
13	56	0	39.820	0	0	southeast	11090.717800
14	27	1	42.130	0	1	southeast	39611.757700
15	19	1	24.600	1	0	southwest	1837.237000
16	52	0	30.780	1	0	northeast	10797.336200
17	23	1	23.845	0	0	northeast	2395.171550
18	56	1	40.300	0	0	southwest	10602.385000
19	30	1	35.300	0	1	southwest	36837.467000
20	60	0	36.005	0	0	northeast	13228.846950
21	30	0	32.400	1	0	southwest	4149.736000
22	18	1	34.100	0	0	southeast	1137.011000
23	34	0	31.920	1	1	northeast	37701.876800
24	37	1	28.025	2	0	northwest	6203.901750
25	59	0	27.720	3	0	southeast	14001.133800
26	63	0	23.085	0	0	northeast	14451.835150
27	55	0	32.775	2	0	northwest	12268.632250
28	23	1	17.385	1	0	northwest	2775.192150
29	31	1	36.300	2	1	southwest	38711.000000
30	22	1	35.600	0	1	southwest	35585.576000
31	18	0	26.315	0	0	northeast	2198.189850
32	19	0	28.600	5	0	southwest	4687.797000
33	63	1	28.310	0	0	northwest	13770.097900
34	28	1	36.400	1	1	southwest	51194.559140
35	19	1	20.425	0	0	northwest	1625.433750
36	62	0	32.965	3	0	northwest	15612.193350
37	26	1	20.800	0	0	southwest	2302.300000
38	35	1	36.670	1	1	northeast	39774.276300
39	60	1	39.900	0	1	southwest	48173.361000
40	24	0	26.600	0	0	northeast	3046.062000
41	31	0	36.630	2	0	southeast	4949.758700
42	41	1	21.780	1	0	southeast	6272.477200
43	37	0	30.800	2	0	southeast	6313.759000
44	38	1	37.050	1	0	northeast	6079.671500
45	55	1	37.300	0	0	southwest	20630.283510
46	18	0	38.665	2	0	northeast	3393.356350
47	28	0	34.770	0	0	northwest	3556.922300
48	60	0	24.530	0	0	southeast	12629.896700

```
1318   35    1  39.710         4       0  northeast  19496.719170
1319   39    0  26.315         2       0  northwest   7201.700850
1320   31    1  31.065         3       0  northwest   5425.023350
1321   62    1  26.695         0       1  northeast  28101.333050
1322   62    1  38.830         0       0  southeast  12981.345700
1323   42    0  40.370         2       1  southeast  43896.376300
1324   31    1  25.935         1       0  northwest   4239.892650
1325   61    1  33.535         0       0  northeast  13143.336650
1326   42    0  32.870         0       0  northeast   7050.021300
1327   51    1  30.030         1       0  southeast   9377.904700
1328   23    0  24.225         2       0  northeast  22395.744240
1329   52    1  38.600         2       0  southwest  10325.206000
1330   57    0  25.740         2       0  southeast  12629.165600
1331   23    0  33.400         0       0  southwest  10795.937330
1332   52    0  44.700         3       0  southwest  11411.685000
1333   50    1  30.970         3       0  northwest  10600.548300
1334   18    0  31.920         0       0  northeast   2205.980800
1335   18    0  36.850         0       0  southeast   1629.833500
1336   21    0  25.800         0       0  southwest   2007.945000
1337   61    0  29.070         0       1  northwest  29141.360300
```

[ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.
 ↪7,random_state=42)
```

[ ]:
```
x=np.array(df['bmi']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
```

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x,y)
```

```
<ipython-input-91-ee86071c5c26>:3: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  rfc.fit(x,y)
```

[ ]:
```
RandomForestClassifier()
```

[ ]:
```
rf=RandomForestClassifier()
```

[ ]:
```
params={'max_depth':[2,3,5,10,20],
'min_samples_leaf':[5,10,20,50,100,200],
'n_estimators':[10,25,30,50,100,200]}
```

[ ]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring='accuracy')
```

```
grid_search.fit(x,y)
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

```
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:909:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  self.best_estimator_.fit(X, y, **fit_params)
```

[ ]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy')
```

[ ]:
```python
grid_search.best_score_
```

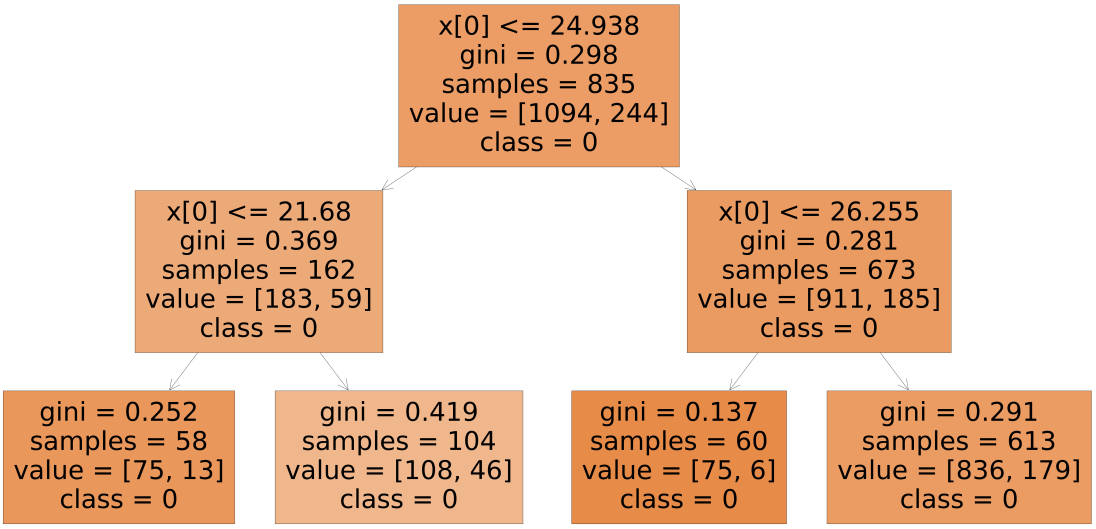
[ ]: 0.7952167414050823

[ ]:
```python
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=10)
```

[ ]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],class_names=['0','1'],filled=True)
```

[ ]:
```
[Text(0.5, 0.8333333333333334, 'x[0] <= 24.938\ngini = 0.298\nsamples =
835\nvalue = [1094, 244]\nclass = 0'),
 Text(0.25, 0.5, 'x[0] <= 21.68\ngini = 0.369\nsamples = 162\nvalue = [183,
59]\nclass = 0'),
 Text(0.125, 0.16666666666666666, 'gini = 0.252\nsamples = 58\nvalue = [75,
13]\nclass = 0'),
 Text(0.375, 0.16666666666666666, 'gini = 0.419\nsamples = 104\nvalue = [108,
46]\nclass = 0'),
 Text(0.75, 0.5, 'x[0] <= 26.255\ngini = 0.281\nsamples = 673\nvalue = [911,
185]\nclass = 0'),
 Text(0.625, 0.16666666666666666, 'gini = 0.137\nsamples = 60\nvalue = [75,
6]\nclass = 0'),
 Text(0.875, 0.16666666666666666, 'gini = 0.291\nsamples = 613\nvalue = [836,
179]\nclass = 0')]
```



[ ]:
```
rf_best.feature_importances_
```

[ ]:
```
array([1.])
```

[ ]:
```
imp_df=pd.DataFrame({"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

[ ]:
```
   Imp
0  1.0
```

CONCLUSION:-

THE SCORE OF RANDOM FOREST IS :- 1.0

-> BY COMPARING ALL THE ABOVE MODELS

#WE CAN CONCLUDE THAT LOGISTIC REGRESSION HAS HIGHEST ACCURACY. SO WE CAN PREFER LOGISTIC REGRESSION FOR THIS DATASET