PROBLEM STATEMENT:- TO PREDICT THE INSURANCE CHARGES BASED ON VARIOUS FEATURES OF THE DATASET

IMPORTING THE ESSENTIAL LIBRARIES:-

In [1]:

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

LOADING THE DATASET:-

In [2]:

```
df=pd.read_csv(r"C:\Users\hp\Downloads\insurance.csv")
df
```

Out[2]:

	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 × 7 columns

TO PRINT THE FIRST 5 ROWS OF A DATASET:-

In [3]:

```
df.head()
```

Out[3]:

	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:-

In [4]:

```
df.tail()
```

Out[4]:

	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:-

In [5]:

```
df.describe()
```

Out[5]:

	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:-

In [6]:

```
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:-

In [7]:

```
df.shape
```

Out[7]:

```
(1338, 7)
```

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

In [8]:

```
df.isnull().any()
```

Out[8]:

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

SUBSET OF A DATASET

In [9]:

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

Out[9]:

	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 × 4 columns

CONVERTING THE CATAGORICAL VALUE INTO INTEGER:-

In [10]:

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

Out[10]:

	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 × 4 columns

TO CHECK THE SHAPE OF THE SUBSETTED DATASET:-

In [11]:

```
df.shape
```

Out[11]:

```
(1338, 4)
```

EXPLORATARY DATA ANALYSIS(EDA)

IMPORTING THE LIBRARIES FOR THE VISUVALIZATION

In [12]:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

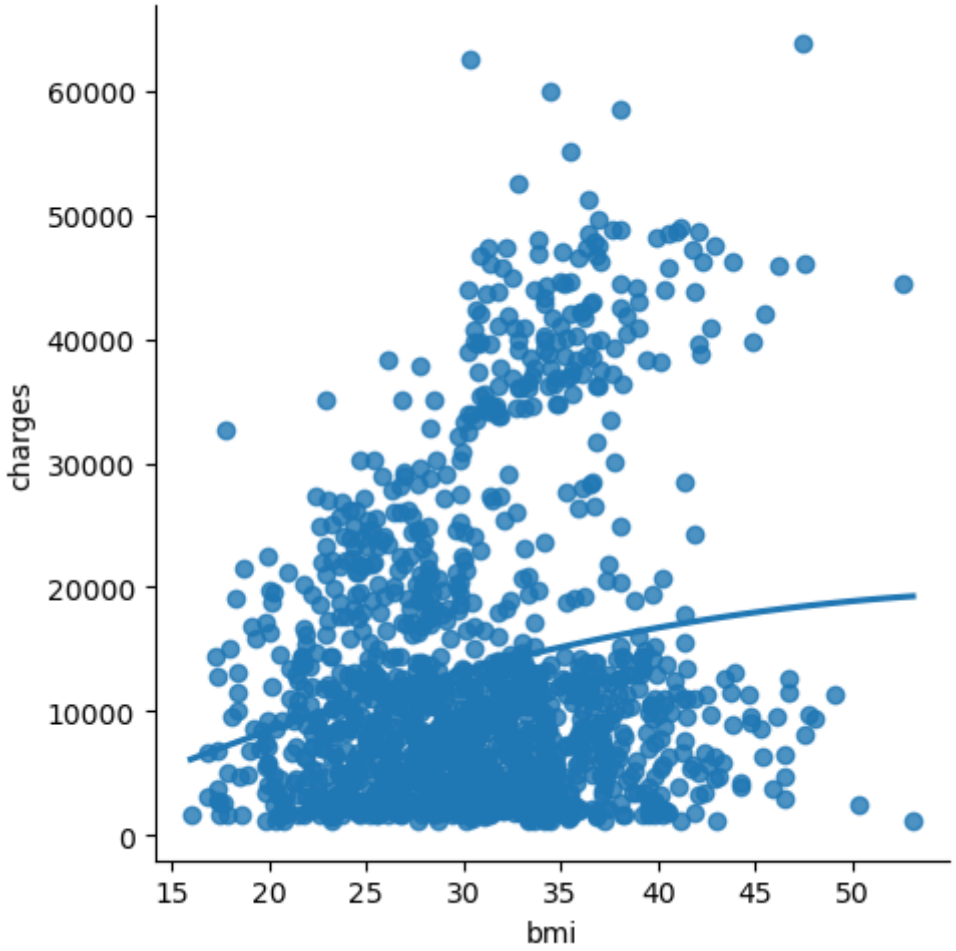
PLOTTING THE GRAPH:-

In [13]:

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

Out[13]:

```
<seaborn.axisgrid.FacetGrid at 0x1b69b56e980>
```



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

In [14]:

```
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

In [15]:

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

TO CHECK THE REGRESSION SCORE:-

In [16]:

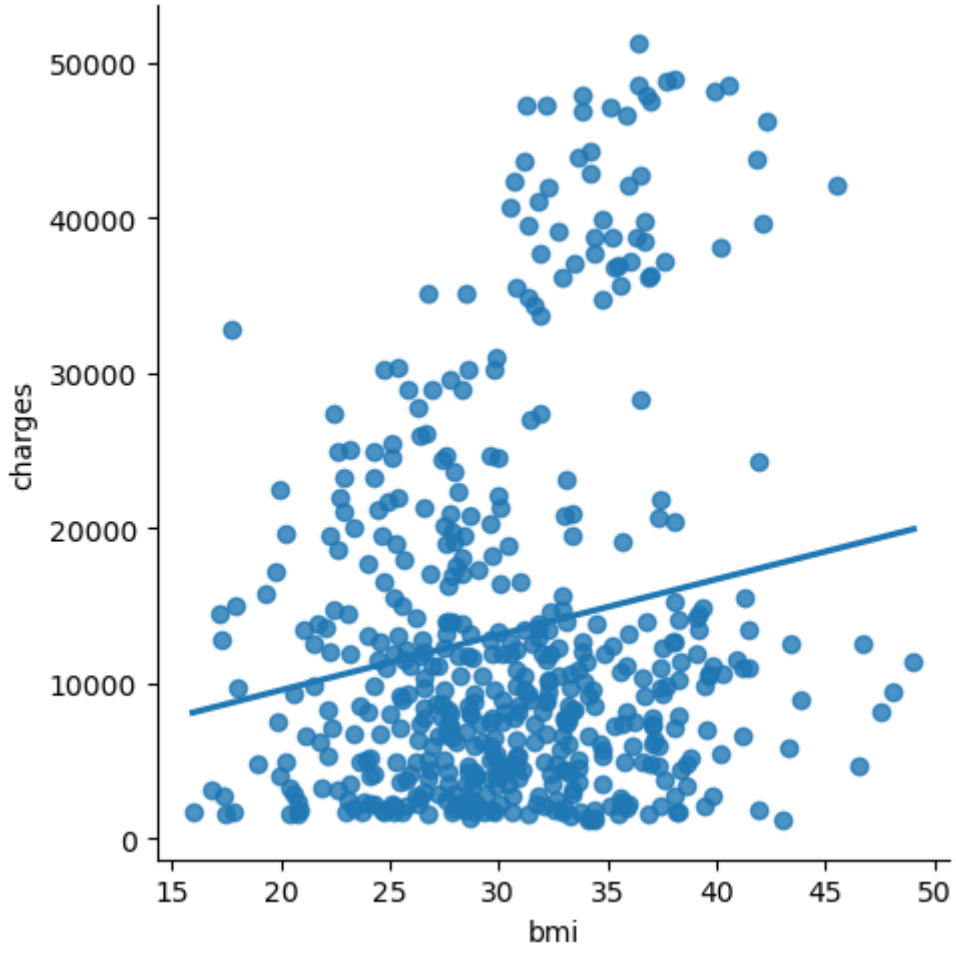
```
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.033168935746127115
```

TO PLOT THE SUBSETTED DATASET:-

In [17]:

```
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)
```



In [18]:

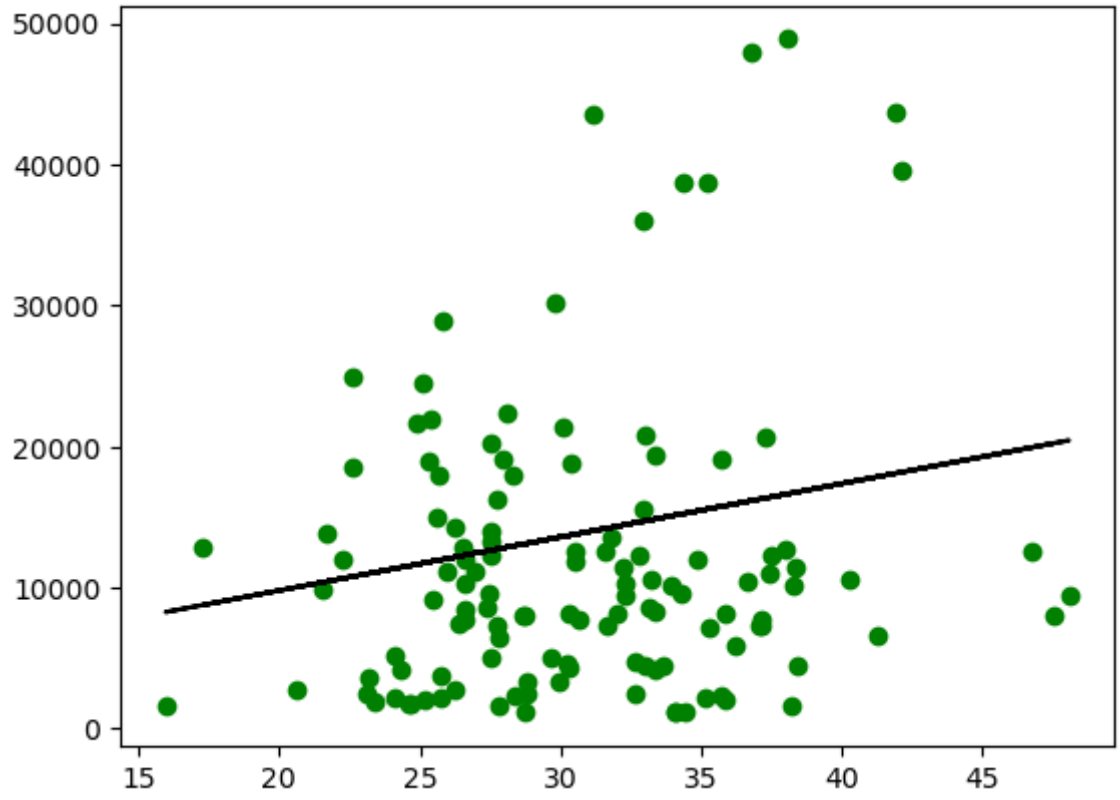
```
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression Score:",regr.score(x_test,y_test))
```

```
Regression Score: -0.011614122340630528
```

TO PLOT THE PREDICTED DATA:-

In [19]:

```
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:-

In [20]:

```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

DEFINING THE TARGET AND FEATURE VECTORS:-

In [21]:

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

TO FIT THE TARGET AND FEATURE VECTORS:-

In [22]:

```
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)
```

In [23]:

```
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:-

In [24]:

```
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:-

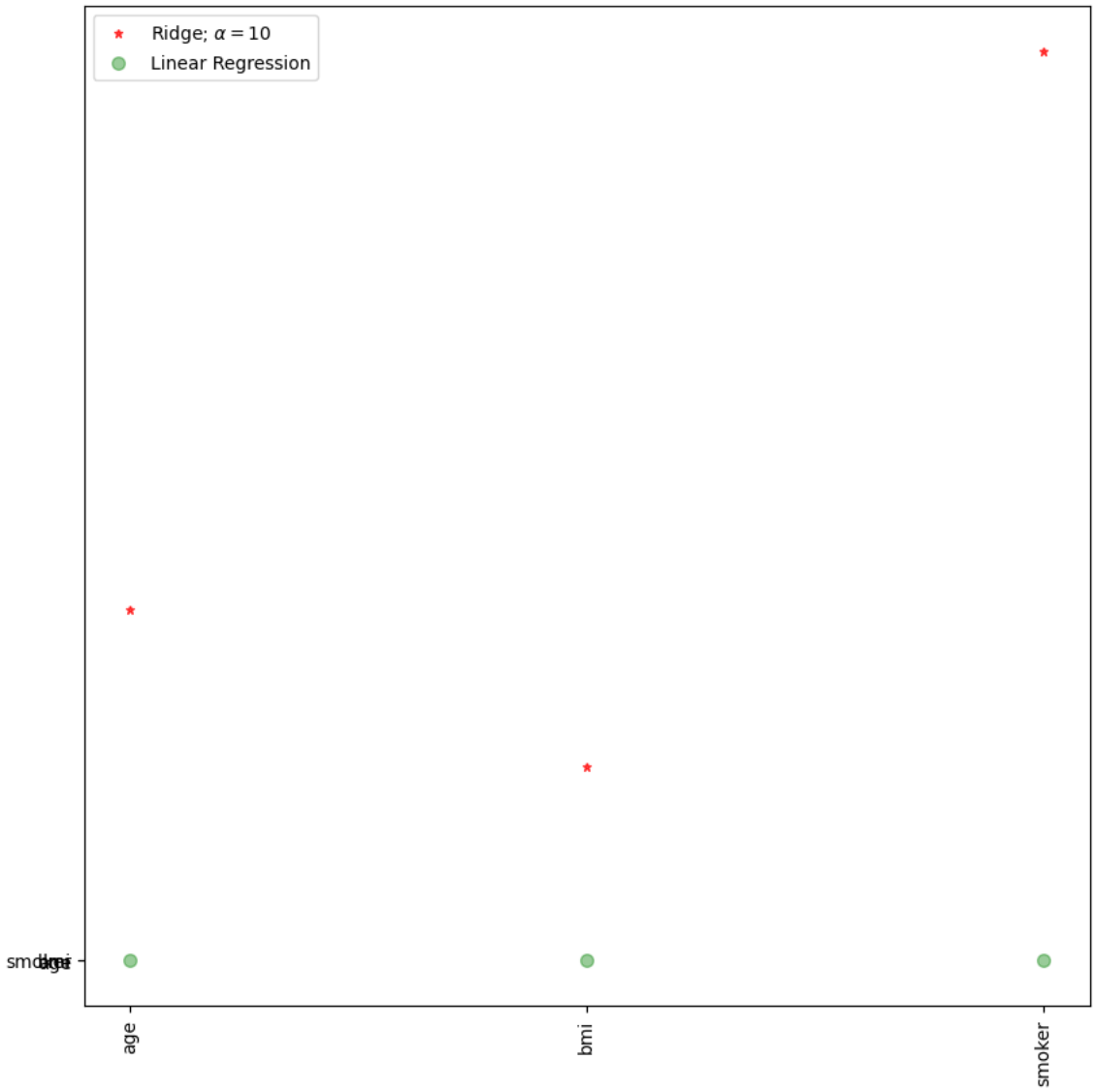
In [25]:

```
lr=LinearRegression()
```

PLOTTING THE GRAPH:-

In [26]:

```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,colo
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color="green",label
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



LASSO MODEL:-

In [27]:

```
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))
```

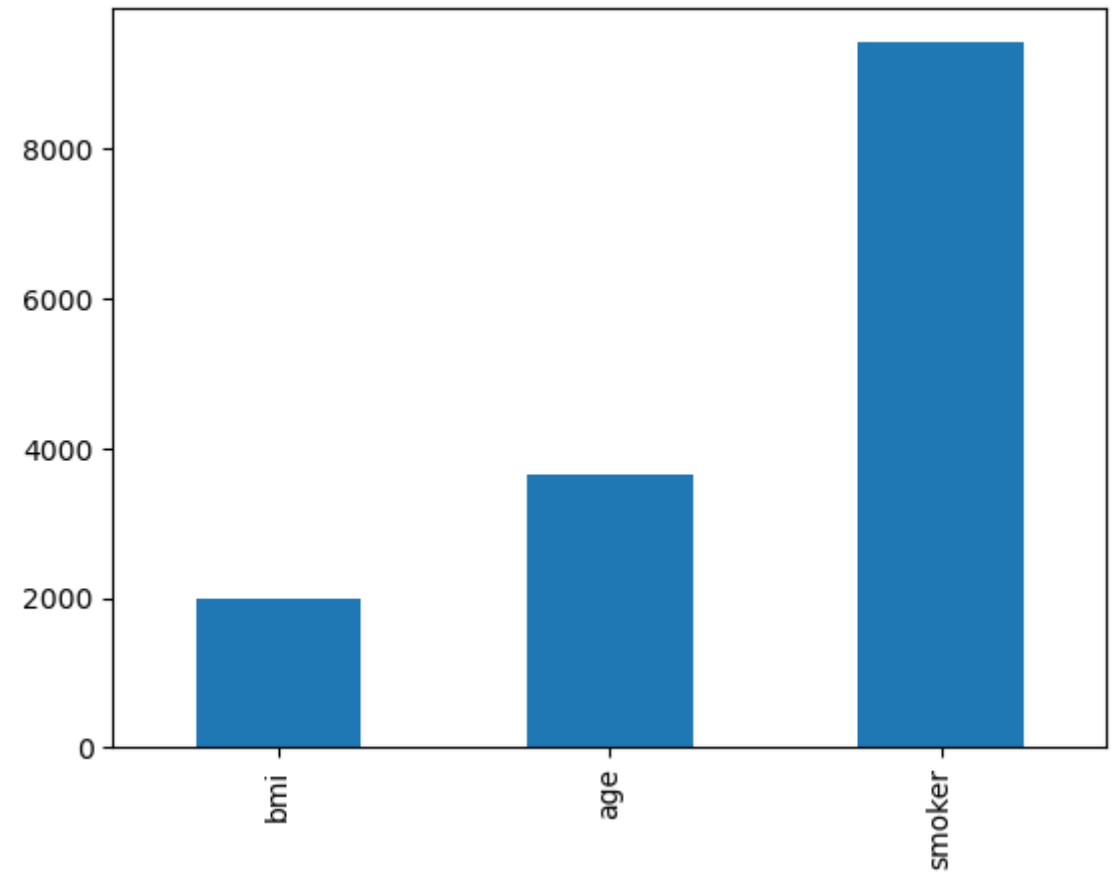
```
Lasso Model:-
The train score for ls model is 0.727786036307324
The test score for ls model is0.7872229669132395
```

In [28]:

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

Out[28]:

```
<Axes: >
```



LASSO CV MODEL:-

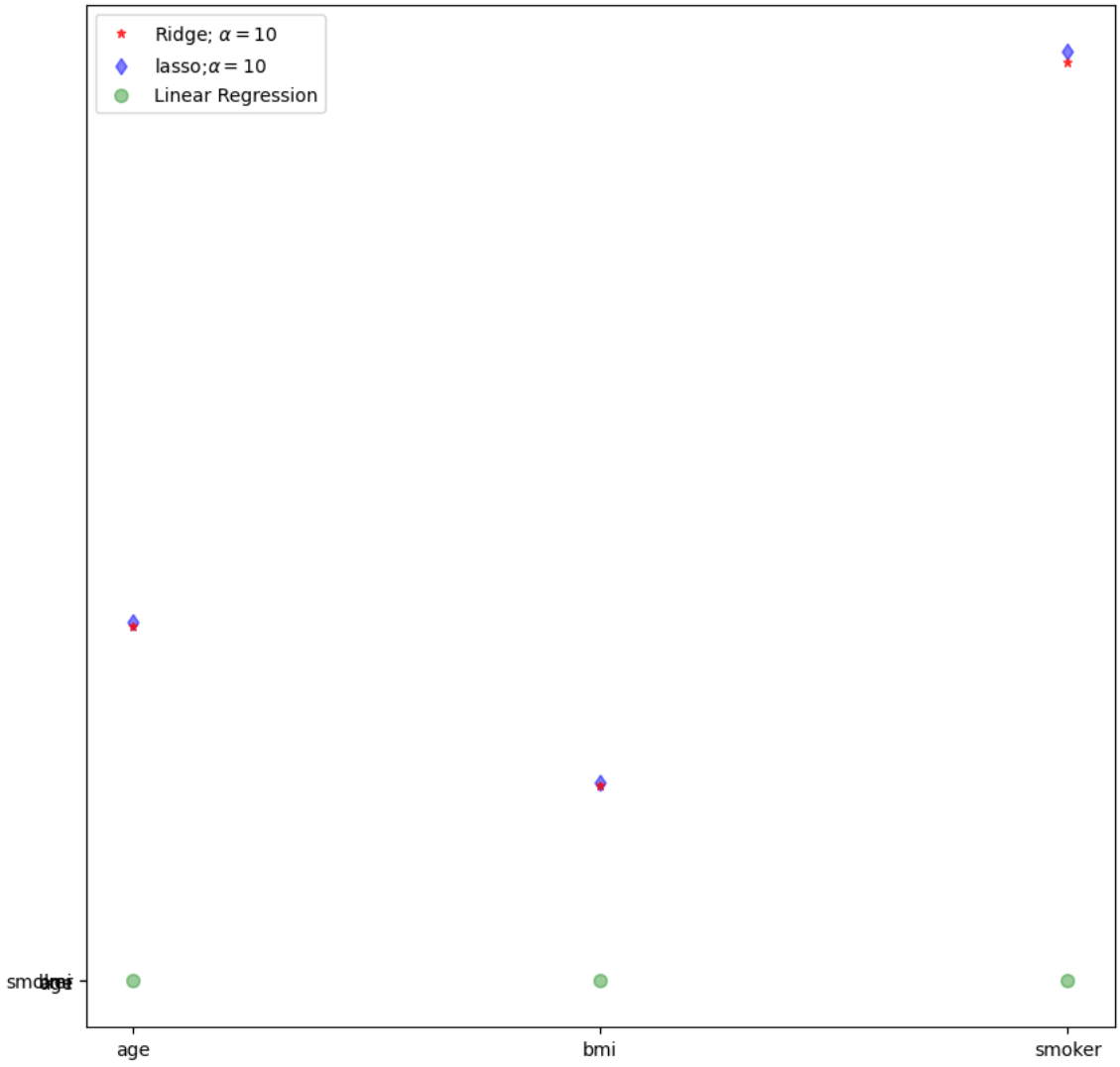
In [29]:

```
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
```

In [30]:

```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,colo
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color="green",label
plt.legend()
plt.show()
```



ELASTIC NET:-

In [31]:

```
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 0.727786036307324
The test score for ls model is0.7872229669132395
```

In [32]:

```
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:-

In [33]:

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

LOADING THE DATASET:-

In [34]:

```
df=pd.read_csv(r"C:\Users\hp\Downloads\insurance.csv")
df
```

Out[34]:

	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 × 7 columns

In [35]:

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

In [36]:

```
df.shape
```

Out[36]:

```
(1338, 7)
```

In [37]:

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

Out[37]:

	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

In [38]:

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

Out[38]:

	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

In [39]:

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

In [40]:

```
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
```

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

In [41]:

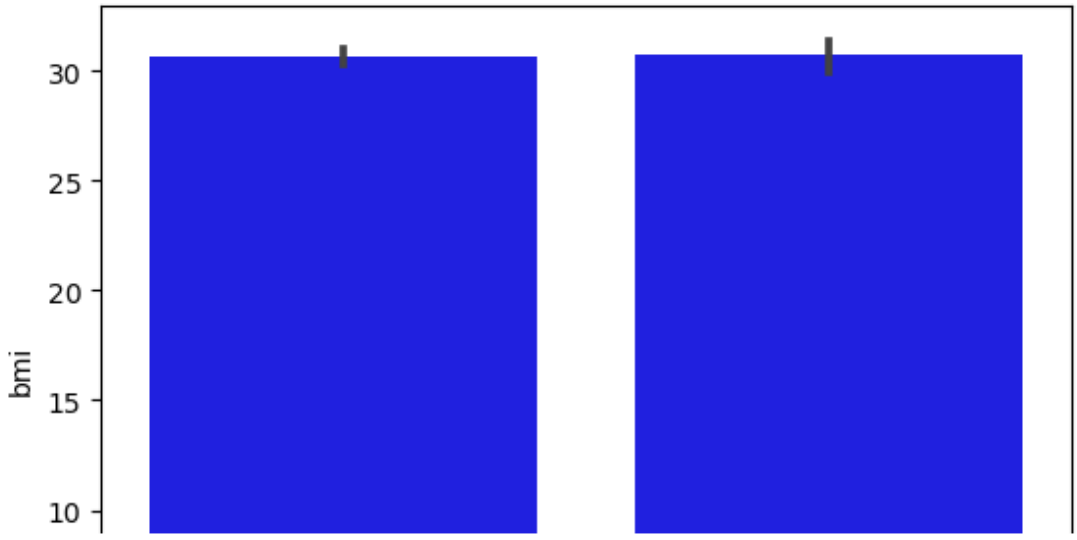
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [42]:

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

Out[42]:

```
<Axes: xlabel='smoker', ylabel='bmi'>
```



In [43]:

```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [44]:

```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_intercept=True,i
```

In [45]:

```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [46]:

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

In [47]:

```
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]
```

In [48]:

```
print('The algoritham was trained to predict one of the two classes:%s'%(algorithm.class
```

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

In [49]:

```
print(" " "The Model says the probability of the observation we passed belonging to clas
print()
```

```
 The Model says the probability of the observation we passed belonging to
class['0'] Is 0.8057078436306042
```

In [50]:

```
print(" " "The Model says the probability of the observation we passed belonging to clas
```

```
 The Model says the probability of the observation we passed belonging to
class['1'] Is 0.19429215636939584
```

In [51]:

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

In [52]:

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

```
Logistic Regression Score: 0.7952167414050823

C:\Users\hp\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143:
DataConversionWarning: A column-vector y was passed when a 1d array was ex
pected. Please change the shape of y to (n_samples, ), for example using r
avel().
  y = column_or_1d(y, warn=True)
```

CONCLUSION:-

THE SCORE OF LOGISTIC REGRESSION IS :0.7952167414050823

DESICION TREE:-

In [53]:

```
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
```

In [54]:

```
df=pd.read_csv(r"C:\Users\hp\Downloads\insurance.csv")
df
```

66	61	female	39.100	2	no	southwest	14235.072000
67	40	male	26.315	1	no	northwest	6389.377850
68	40	female	36.190	0	no	southeast	5920.104100
69	28	male	23.980	3	yes	southeast	17663.144200
70	27	female	24.750	0	yes	southeast	16577.779500
71	31	male	28.500	5	no	northeast	6799.458000
72	53	female	28.100	3	no	southwest	11741.726000
73	58	male	32.010	1	no	southeast	11946.625900
74	44	male	27.400	2	no	southwest	7726.854000
75	57	male	34.010	0	no	northwest	11356.660900
76	29	female	29.590	1	no	southeast	3947.413100
77	21	male	35.530	0	no	southeast	1532.469700
78	22	female	39.805	0	no	northeast	2755.020950

In [55]:

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

Out[55]:

	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

In [56]:

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

Out[56]:

	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

In [57]:

```
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)
```

In [58]:

```
clf=DecisionTreeClassifier(random_state=0)
```

In [59]:

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

Out[59]:

```
DecisionTreeClassifier(random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [60]:

```
score=clf.score(x_test,y_test)
print(score)
```

```
0.8048780487804879
```

CONCLUSION:-

THE SCORE OF THE DECISION TREE IS : 1.0

RANDOM FOREST:-

In [61]:

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

In [62]:

```
df=pd.read_csv(r"C:\Users\hp\Downloads\insurance.csv")
df
```

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
46	18	female	38.665	2	no	northeast	3393.356350
47	28	female	34.770	0	no	northwest	3556.922300
48	60	female	24.530	0	no	southeast	12629.896700
49	36	male	35.200	1	yes	southeast	38709.176000
50	18	female	35.625	0	no	northeast	2211.130750

In [63]:

```
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
```

In [64]:

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

Out[64]:

	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

In [65]:

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

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
49	36	1	35.200	1	1	southeast	38709.176000
50	18	0	35.625	0	0	northeast	2211.130750
51	21	0	33.630	2	0	northwest	3579.828700

In [78]:

```
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)
```

In [80]:

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

In [82]:

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

```
C:\Users\hp\AppData\Local\Temp\ipykernel_2080\2874065197.py:3: DataConvers
ionWarning: A column-vector y was passed when a 1d array was expected. Ple
ase change the shape of y to (n_samples,), for example using ravel().
  rfc.fit(x,y)
```

Out[82]:

```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [83]:

```
rf=RandomForestClassifier()
```

In [84]:

```
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]}
```

In [86]:

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring='accuracy')
grid_search.fit(x,y)
```

```
C:\Users\hp\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:686: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples,), fo
r example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\hp\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:686: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples,), fo
r example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\hp\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:686: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples,), fo
r example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\hp\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:686: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples,), fo
r example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

In [87]:

```
grid_search.best_score_
```

Out[87]:

```
0.7952167414050823
```

In [88]:

```
rf_best=grid_search.best_estimator_
print(rf_best)
```

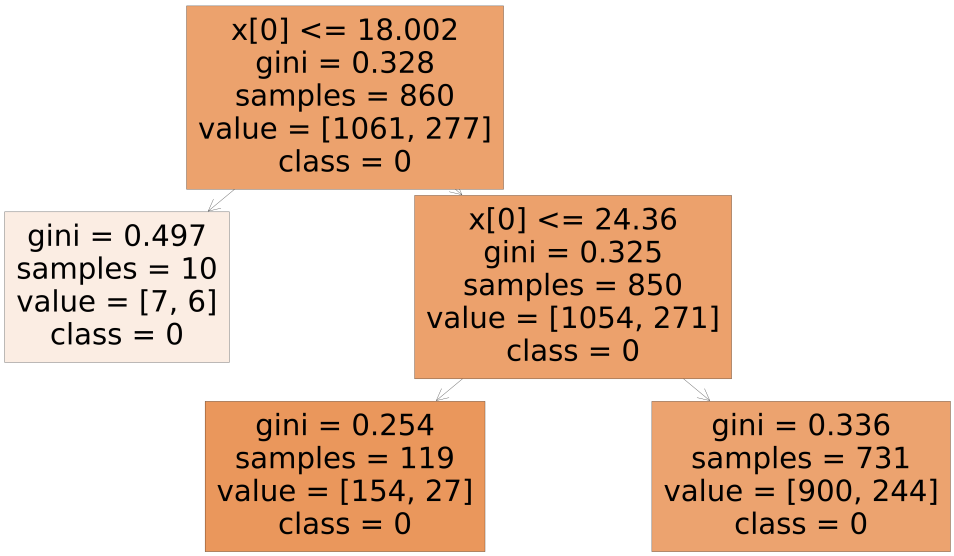
```
RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=10)
```

In [91]:

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

Out[91]:

```
[Text(0.4, 0.8333333333333334, 'x[0] <= 18.002\ngini = 0.328\nsamples = 86
0\nvalue = [1061, 277]\nclass = 0'),
 Text(0.2, 0.5, 'gini = 0.497\nsamples = 10\nvalue = [7, 6]\nclass = 0'),
 Text(0.6, 0.5, 'x[0] <= 24.36\ngini = 0.325\nsamples = 850\nvalue = [105
4, 271]\nclass = 0'),
 Text(0.4, 0.16666666666666666, 'gini = 0.254\nsamples = 119\nvalue = [15
4, 27]\nclass = 0'),
 Text(0.8, 0.16666666666666666, 'gini = 0.336\nsamples = 731\nvalue = [90
0, 244]\nclass = 0')]
```



In [92]:

```
rf_best.feature_importances_
```

Out[92]:

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

In [96]:

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

Out[96]:

	Imp
0	1.0

CONCLUSION:-

THE SCORE OF RANDOM FOREST IS :-1.0

THE LOGISTIC REGRESSION HAS HIGHEST ACCURACY.SO WE PREFER LOGISTIC REGRESSION

DASHBOARD USING EXCEL:-