Problem Statement: To predict how best the data fits and which model suits

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
```

Data collection

In [3]:
```
df=pd.read_csv(r"C:\Users\rubin\Downloads\insurance.csv")
df
```

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
...	...	...	...	...	...	...	...
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

Data Cleaning and Preprocessing

In [4]:
```
df.head()
```

Out[4]:

	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

In [5]:
```
df.tail()
```

Out[5]:

	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

In [6]:
```
df.shape
```

Out[6]:
```
(1338, 7)
```

In [7]:
```
df.describe()
```

Out[7]:

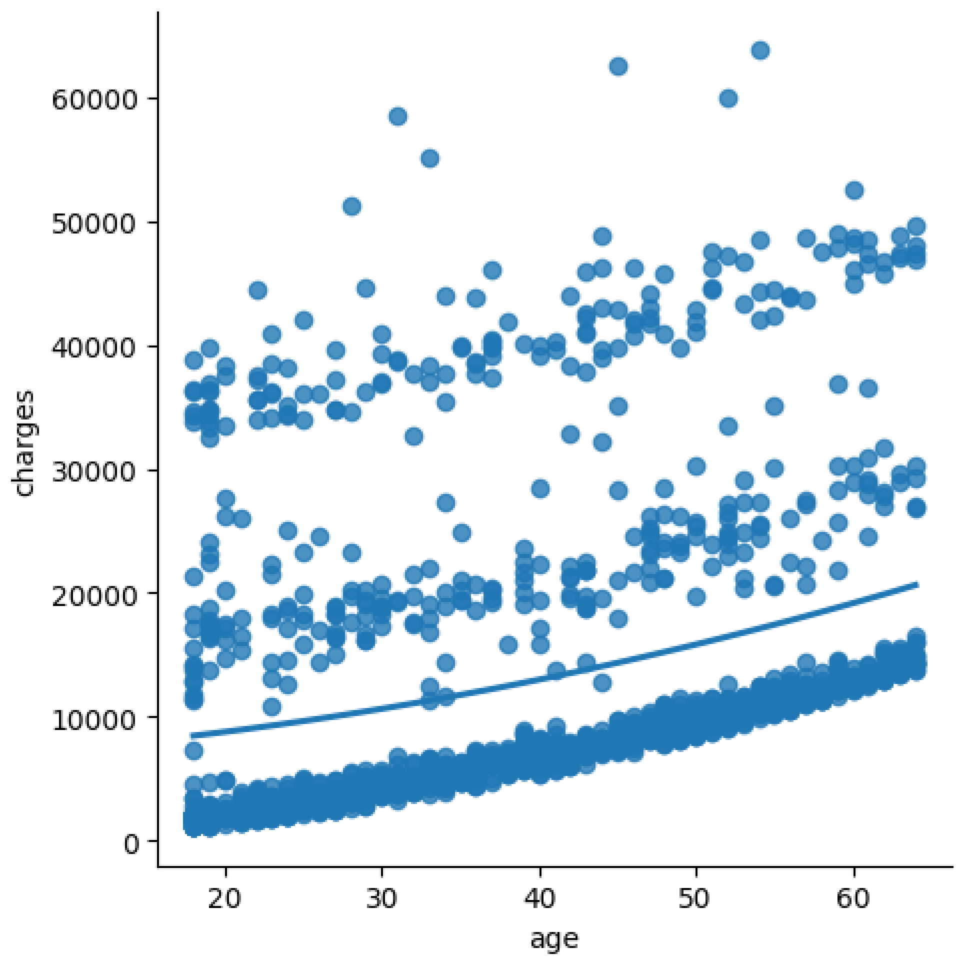
	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000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

In [8]:
```
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 [9]:
```
sns.lmplot(x="age",y="charges",data=df,order=2,ci=None)
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x243c7290850>



From the above scatter plot we can able to know that the aged people charges are low

In [10]:
```
df.fillna(method='ffill',inplace=True)
```

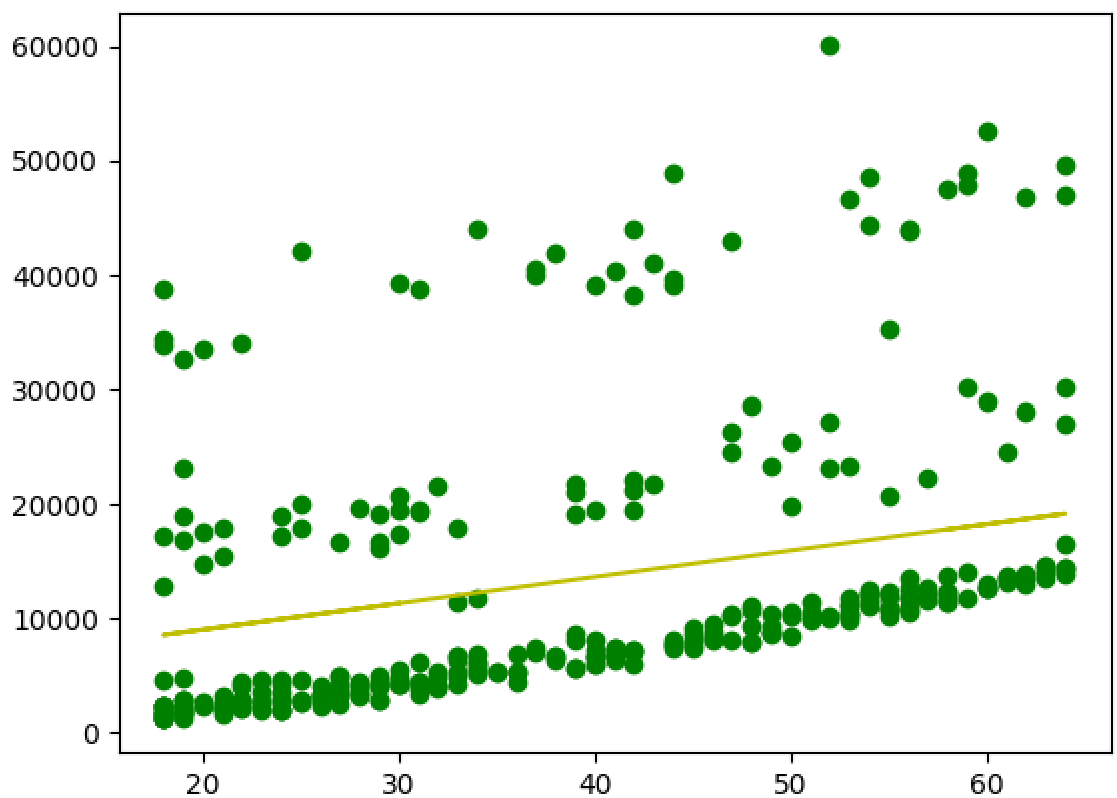
In [11]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['charges']).reshape(-1,1)
```

In [12]:
```
df.dropna(inplace=True)
```

In [13]:
```
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))
```

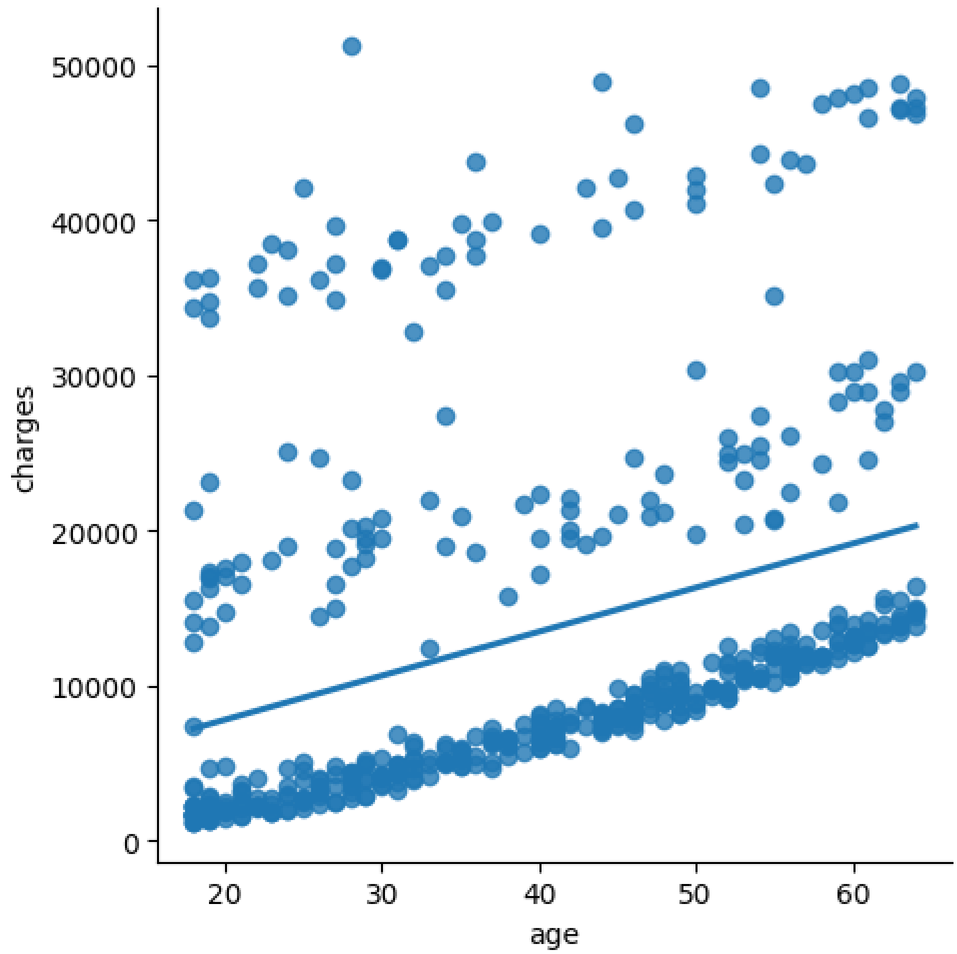
```
0.1373952553426976
```

In [14]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='g')
plt.plot(x_test,y_pred,color='y')
plt.show()
```



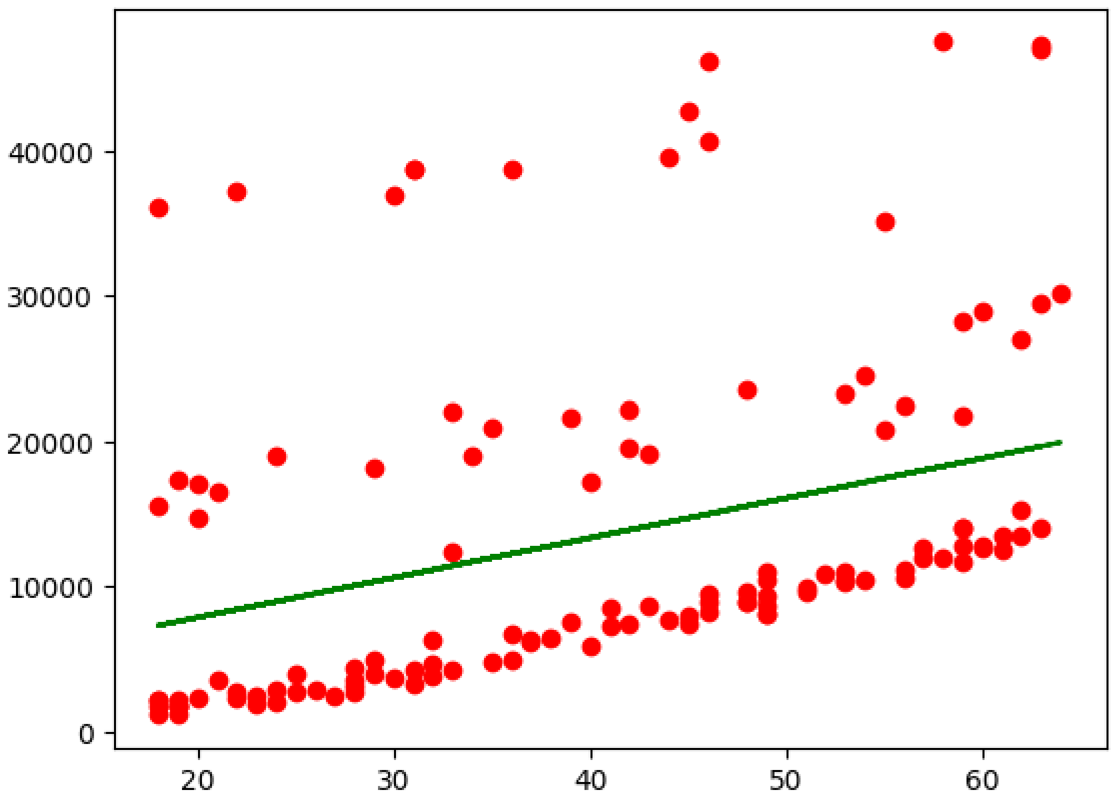
In [15]:
```
df500=df[:][:500]
sns.lmplot(x="age",y="charges",data=df500,order=1,ci=None)
```

Out[15]: <seaborn.axisgrid.FacetGrid at 0x243a4f02530>



In [16]:
```
df500.fillna(method='ffill',inplace=True)
x=np.array(df500['age']).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:",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='r')
plt.plot(x_test,y_pred,color='g')
plt.show()
```

```
Regression: 0.1392391637942142
```



In [17]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 Score:",r2)
```

```
R2 Score: 0.1392391637942142
```

In [18]:
```
df.isnull().sum()
```

Out[18]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

There are no null values in the given data set

Implementing Ridge&Lasso Regression Model

In [19]:
```
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
```

In [20]:
```
convert={"sex":{"male":1,"female":2}}
df=df.replace(convert)
df
```

Out[20]:

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

1338 rows × 7 columns

In [21]:
```
convert={"smoker":{"yes":1,"no":2}}
df=df.replace(convert)
df
```

Out[21]:

	age	sex	bmi	children	smoker	region	charges
0	19	2	27.900	0	1	southwest	16884.92400
1	18	1	33.770	1	2	southeast	1725.55230
2	28	1	33.000	3	2	southeast	4449.46200
3	33	1	22.705	0	2	northwest	21984.47061
4	32	1	28.880	0	2	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	2	northwest	10600.54830
1334	18	2	31.920	0	2	northeast	2205.98080
1335	18	2	36.850	0	2	southeast	1629.83350
1336	21	2	25.800	0	2	southwest	2007.94500
1337	61	2	29.070	0	1	northwest	29141.36030

1338 rows × 7 columns

In [22]:
```
convert={"region":{"southeast":3,"southwest":4,"northeast":5,"northwest":6}}
df=df.replace(convert)
df
```

Out[22]:

	age	sex	bmi	children	smoker	region	charges
0	19	2	27.900	0	1	4	16884.92400
1	18	1	33.770	1	2	3	1725.55230
2	28	1	33.000	3	2	3	4449.46200
3	33	1	22.705	0	2	6	21984.47061
4	32	1	28.880	0	2	6	3866.85520
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	2	6	10600.54830
1334	18	2	31.920	0	2	5	2205.98080
1335	18	2	36.850	0	2	3	1629.83350
1336	21	2	25.800	0	2	4	2007.94500
1337	61	2	29.070	0	1	6	29141.36030

1338 rows × 7 columns

In [23]:
```
features = df.columns[0:1]
target = df.columns[-1]
#X and y values
X = df[features].values
y = df[target].values
#splot
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, randc
print("The dimension of x_train is {}".format(X_train.shape))
print("The dimension of x_test is {}".format(X_test.shape))
#Scale features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
The dimension of x_train is (936, 1)
The dimension of x_test is (402, 1)
```

In [24]:
```
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)
print("\nRidge Model:\n")
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 is 0.1038565773808342
The test score for ridge model is 0.04500705629317203
```

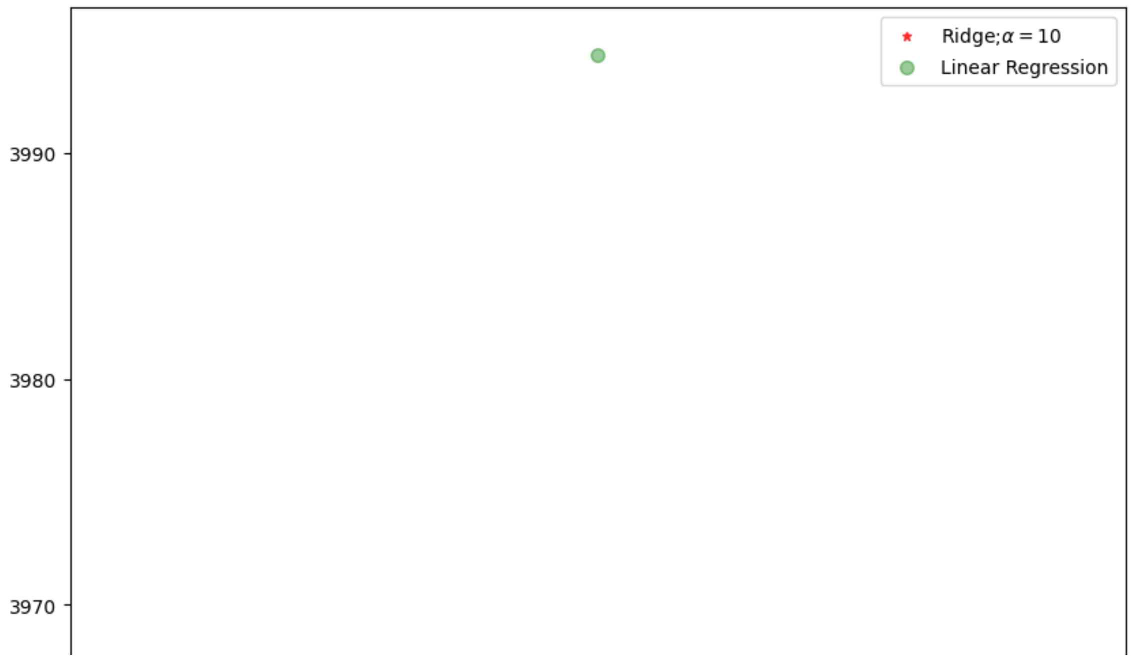
In [25]:
```
lr = LinearRegression()
#Fit model
lr.fit(X_train, y_train)
#predict
#prediction = lr.predict(X_test)
#actual
actual = y_test
train_score_lr = lr.score(X_train, y_train)
test_score_lr = lr.score(X_test, y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))
```

```

Linear Regression Model:

The train score for lr model is 0.10386818385382768
The test score for lr model is 0.04432028433481028
```

In [26]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
#plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,
plt.xticks(rotation=90)
plt.legend()
#plt.title("comparison plot of Ridge,Lasso and Linear regression model")
plt.show()
```



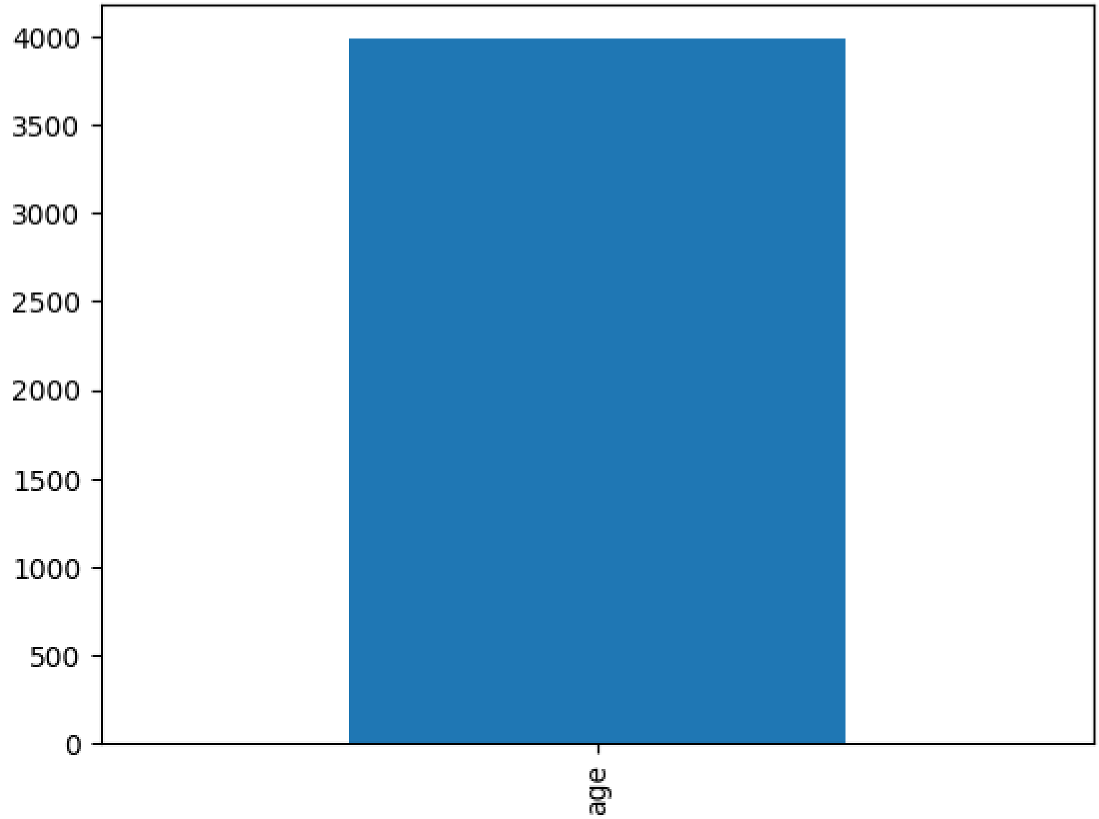
In [27]:
```
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso Model:

The train score for ls model is 0.1038675328268055
The test score for ls model is 0.04448522322607196
```

In [28]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "ba
```

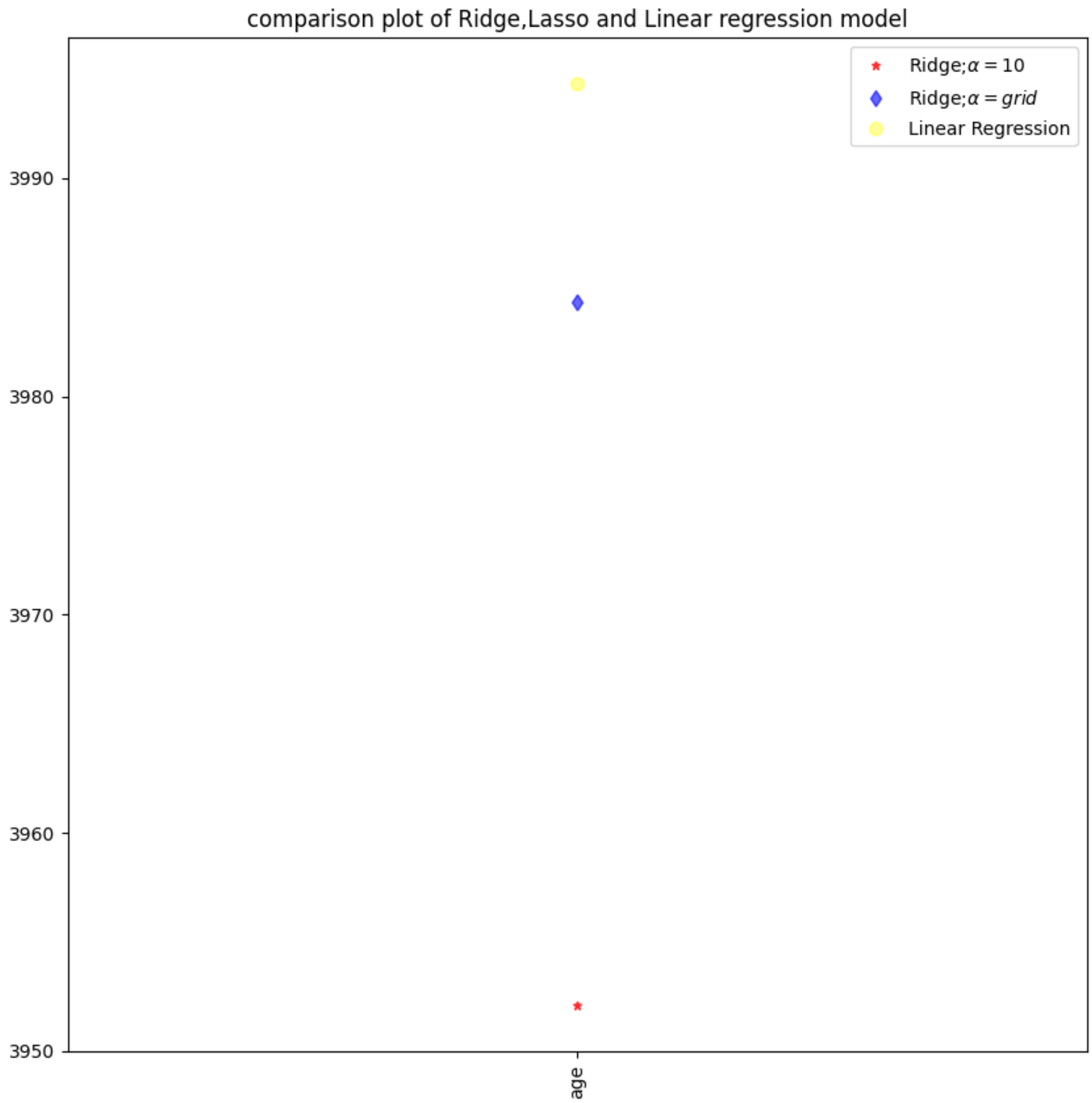
Out[28]: <Axes: >



In [29]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(X_tra
print(lasso_cv.score(X_train,y_train))
print(lasso_cv.score(X_test,y_test))
```

```
0.1038675328268055
0.04448522322607196
```

In [30]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
plt.plot(lasso_cv.coef_,alpha=0.6,linestyle='none',marker='d',markersize=6,col
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,
plt.xticks(rotation=90)
plt.legend()
plt.title("comparison plot of Ridge,Lasso and Linear regression model")
plt.show()
```



In [31]:
```
from sklearn.linear_model import RidgeCV
#Ridge Cross validation
ridge_cv = RidgeCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10]).fit(X_train, y_t
#score
print("The train score for ridge model is {}".format(ridge_cv.score(X_train, y
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_
```

```
The train score for ridge model is 0.10385657738083431
The train score for ridge model is 0.0450705629316 67
```

In [ ]:

Elastic net regression

In [32]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
regr.score(X,y)
```

```
[257.0684655]
3191.532406056682
```

Out[32]:
```
0.08940532368214038
```

In [33]:
```
y_pred_elastic=regr.predict(X_train)
```

In [34]:
```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print("Mean Squared Error on test set",mean_squared_error)
```

```
Mean Squared Error on test set 256816236.54565856
```

Logistic Regression

In [35]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [36]:
```
df=pd.read_csv(r"C:\Users\rubin\Downloads\insurance.csv")
df
```

Out[36]:

	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 [37]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [38]:
```
print('This DataFrame has %d Rows and %d columns'%(df.shape))
```

```
This DataFrame has 1338 Rows and 7 columns
```

In [39]:
```
convert={"smoker":{"yes":1,"no":2}}
df=df.replace(convert)
df
```

18	56	male	40.300	0	2	southwest	10602.385000
19	30	male	35.300	0	1	southwest	36837.467000
20	60	female	36.005	0	2	northeast	13228.846950
21	30	female	32.400	1	2	southwest	4149.736000
22	18	male	34.100	0	2	southeast	1137.011000
23	34	female	31.920	1	1	northeast	37701.876800
24	37	male	28.025	2	2	northwest	6203.901750
25	59	female	27.720	3	2	southeast	14001.133800
26	63	female	23.085	0	2	northeast	14451.835150
27	55	female	32.775	2	2	northwest	12268.632250
28	23	male	17.385	1	2	northwest	2775.192150
29	31	male	36.300	2	1	southwest	38711.000000
30	22	male	35.600	0	1	southwest	35585.576000

In [43]:
```
convert={"region":{"southeast":3,"southwest":4,"northeast":5,"northwest":6}}
df=df.replace(convert)
df
```

45	8	2	4	37.300	55
46	9	2	5	38.665	18
47	9	2	6	34.770	28
48	9	2	3	24.530	60
49	8	1	3	35.200	36
50	9	2	5	35.625	18
51	9	2	6	33.630	21
52	8	1	4	28.000	48
53	8	1	3	34.430	36
54	9	2	6	28.690	40
55	8	1	6	36.955	58
56	9	2	5	31.825	58
57	8	1	3	31.680	18

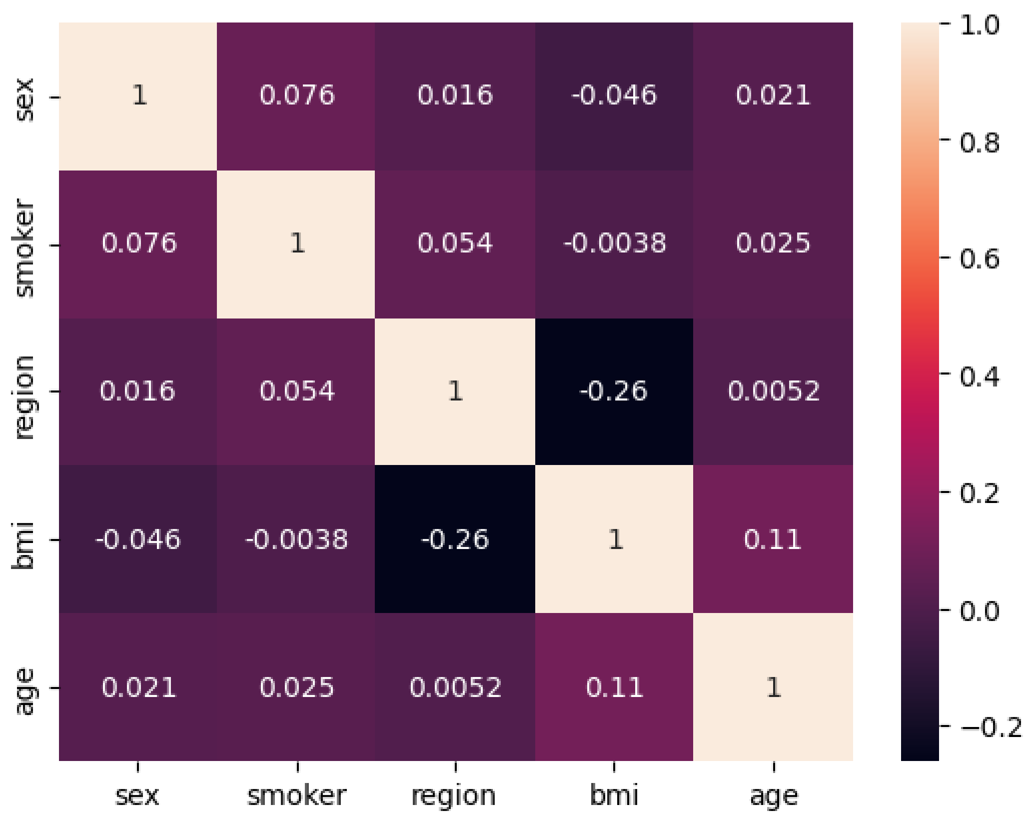
In [85]:
```
convert={"sex":{"male":8,"female":9}}
df=df.replace(convert)
df
```

Out[85]:

	age	sex	bmi	children	smoker	region	charges
0	19	9	27.900	0	yes	2	16884.924000
1	18	8	33.770	1	no	1	1725.552300
2	28	8	33.000	3	no	1	4449.462000
3	33	8	22.705	0	no	4	21984.470610
4	32	8	28.880	0	no	4	3866.855200
5	31	9	25.740	0	no	1	3756.621600
6	46	9	33.440	1	no	1	8240.589600
7	37	9	27.740	3	no	4	7281.505600
8	37	8	29.830	2	no	3	6406.410700
9	60	9	25.840	0	no	4	28923.136920
10	25	8	26.220	0	no	3	2721.320800

In [44]:
```
df=df[['sex','smoker','region','bmi','age']]
sns.heatmap(df.corr(),annot=True)
```

Out[44]: <Axes: >



In [45]:
```
features_matrix=df.iloc[:,0:4]
```

In [46]:
```
target_vector=df.iloc[:,-3]
```

In [47]:
```
print('The Features Matrix Has %d Rows And %d Column(s)'%(features_matrix.shap
```

```
The Features Matrix Has 1338 Rows And 4 Column(s)
```

In [48]:
```
print('The Target Matrix Has %d Rows And %d Column(s)'%(np.array(target_vector
```

```
The Target Matrix Has 1338 Rows And 1 Column(s)
```

In [49]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [50]:
```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_interc
```

In [51]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_ve
```

In [52]:
```
observation=[[1,0,0.99539,-0.05889,]]
```

In [53]:
```
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 [6]
```

In [54]:
```
print('The Algorithm Was Trained To Predict One of The Two Classes: %s'%(algor
```

```
The Algorithm Was Trained To Predict One of The Two Classes: [3 4 5 6]
```

In [55]:
```
print("""The Model Says The Probability Of The Observation We Passed Belonging
```

```
The Model Says The Probability Of The Observation We Passed Belonging To Clas
s['0'] Is 8.73705876312762e-10
```

In [56]:
```
print()
```

In [57]:
```
print("""The Model Says The Probabaility Of The Observation We Passed Belongin
```

```
The Model Says The Probabaility Of The Observation We Passed Belonging To Cla
ss['1'] Is 0.0006558137260463323
```

In [58]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
```

In [59]:
```
lerg=LogisticRegression()
lerg.fit(x,y)
print(lerg.score(x,y))
```

```
0.7952167414050823

C:\Users\rubin\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\utils\validation.py:1143: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Decision Tree Regression

In [60]:
```
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 [61]:
```
df=pd.read_csv(r"C:\Users\rubin\Downloads\insurance.csv")
df
```

Out[61]:

	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

In [62]:
```python
df['region'].value_counts()
```

Out[62]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [63]:
```python
df['bmi'].value_counts()
```

Out[63]:
```
bmi
32.300    13
28.310     9
30.495     8
30.875     8
31.350     8
30.800     8
34.100     8
28.880     8
33.330     7
35.200     7
25.800     7
32.775     7
27.645     7
32.110     7
38.060     7
25.460     7
30.590     7
27.360     7
```

In [64]:
```python
convert={"sex":{"male":1,"female":0}}
df=df.replace(convert)
df
```

Out[64]:

	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 [65]:
```
x=["bmi","children"]
y=["yes","no"]
all_inputs=df[x]
all_classes=df["sex"]
```

In [66]:
```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_s
```

In [67]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [68]:
```
clf.fit(x_train,y_train)
```

Out[68]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [69]:
```
score=clf.score(x_test,y_test)
print(score)
```

```
0.4507462686567164
```

Random Forest

In [70]:
```
import pandas as pd
import numpy as ny
import matplotlib.pyplot as plt,seaborn as sns
```

In [71]:
```
df=pd.read_csv(r"C:\Users\rubin\Downloads\insurance.csv")
df
```

Out[71]:

	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

In [72]:
```
df['charges'].value_counts()
```

Out[72]:
```
charges
1639.563100     2
16884.924000    1
29330.983150    1
2221.564450     1
19798.054550    1
13063.883000    1
13555.004900    1
44202.653600    1
10422.916650    1
7243.813600     1
11945.132700    1
6311.952000     1
1682.597000     1
5272.175800     1
27218.437250    1
19719.694700    1
4877.981050     1
46255.112500    1
```

In [73]:
```
m={"region":{"southeast":1,"southwest":2,"northeast":3,"northwest":4}}
df=df.replace(m)
print(df)
```

```
36    62  female  32.965         3     no       4  15612.193350
37    26    male  20.800         0     no       2   2302.300000
38    35    male  36.670         1    yes       3  39774.276300
39    60    male  39.900         0    yes       2  48173.361000
40    24  female  26.600         0     no       3   3046.062000
41    31  female  36.630         2     no       1   4949.758700
42    41    male  21.780         1     no       1   6272.477200
43    37  female  30.800         2     no       1   6313.759000
44    38    male  37.050         1     no       3   6079.671500
45    55    male  37.300         0     no       2  20630.283510
46    18  female  38.665         2     no       3   3393.356350
47    28  female  34.770         0     no       4   3556.922300
48    60  female  24.530         0     no       1  12629.896700
49    36    male  35.200         1    yes       1  38709.176000
50    18  female  35.625         0     no       3   2211.130750
51    21  female  33.630         2     no       4   3579.828700
52    48    male  28.000         1    yes       2  23568.272000
53    36    male  34.430         0    yes       1  37742.575700
54    40  female  28.690         3     no       4   8059.679100
55    58    male  36.955         2    yes       4  47496.494450
```

In [74]:
```
df.shape
```

Out[74]:
```
(1338, 7)
```

In [75]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[75]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [76]:
```
rf=RandomForestClassifier()
```

In [77]:
```
params={'max_depth':[2,3,5,10,20],'min_samples_leaf':[5,10,20,50,100,200],'n_e
```

In [78]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring="accuracy
grid_search.fit(x_train,y_train)
```

Out[78]:
```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

In [79]:
```
grid_search.best_score_
```

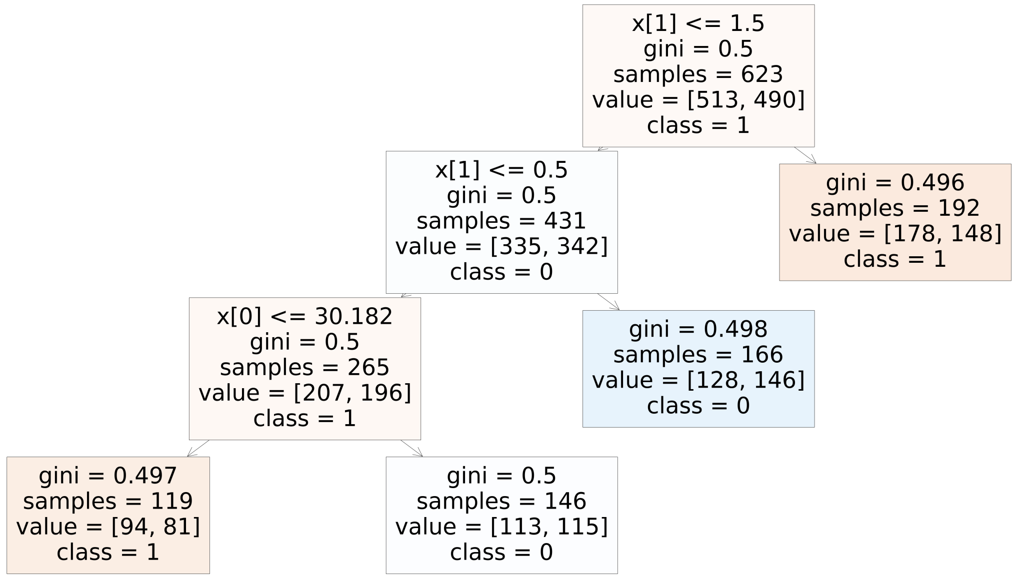
Out[79]:
```
0.5384112253580489
```

In [80]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=10, min_samples_leaf=100, n_estimators=30)
```

In [81]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['1','0'],filled=True);
```



In [82]:

```
rf_best.feature_importances_
```

Out[82]:

```
array([0.81698821, 0.18301179])
```

In [83]:

```
imp_df=pd.DataFrame({"Variance":x_train.columns,"Imp":rf_best.feature_importar
imp_df.sort_values(by="Imp",ascending=False)
```

Out[83]:

	Variance	Imp
0	bmi	0.816988
1	children	0.183012

In [84]:

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

```
0.48059701492537316
```

Conclusion:- For the given insurance data set have performed Linear,Logistic,Decision Tree and Random Forest models of Regressions,and have concluded that the most accuracy is occured in LogisticRegression i.e 79 percent when compare to other Regression models.

And concluded that " LOGISTIC REGRESSION"

In [ ]: