problem statement: to check which model fits the best for the given data set

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

In [153]:
```
df=pd.read_csv(r"C:\Users\ubinl\Downloads\insurance.csv")
df
```

Out[153]:

	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

2)Data cleaning and prepocessing

In [154]:
```
df.head()
```

Out[154]:

	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 [155]:
```
df.tail()
```

Out[155]:

	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 [156]:
```
df.shape
```

Out[156]:
```
(1338, 7)
```

In [157]:
```
df.describe
```

Out[157]:
```
<bound method NDFrame.describe of       age     sex     bmi  children smok
er     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
```

In [158]:
```
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 [159]:
```
df.isnull().any()
```

Out[159]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [160]:
```
df.isna().sum()
```

Out[160]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [161]:
```
df['region'].value_counts()
```

Out[161]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [162]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

Out[162]:

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

In [163]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[163]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	southwest	16884.924000
1	18	0	33.770	1	0	southeast	1725.552300
2	28	0	33.000	3	0	southeast	4449.462000
3	33	0	22.705	0	0	northwest	21984.470610
4	32	0	28.880	0	0	northwest	3866.855200
5	31	1	25.740	0	0	southeast	3756.621600
6	46	1	33.440	1	0	southeast	8240.589600
7	37	1	27.740	3	0	northwest	7281.505600
8	37	0	29.830	2	0	northeast	6406.410700
9	60	1	25.840	0	0	northwest	28923.136920
10	25	0	26.220	0	0	northeast	2721.320800

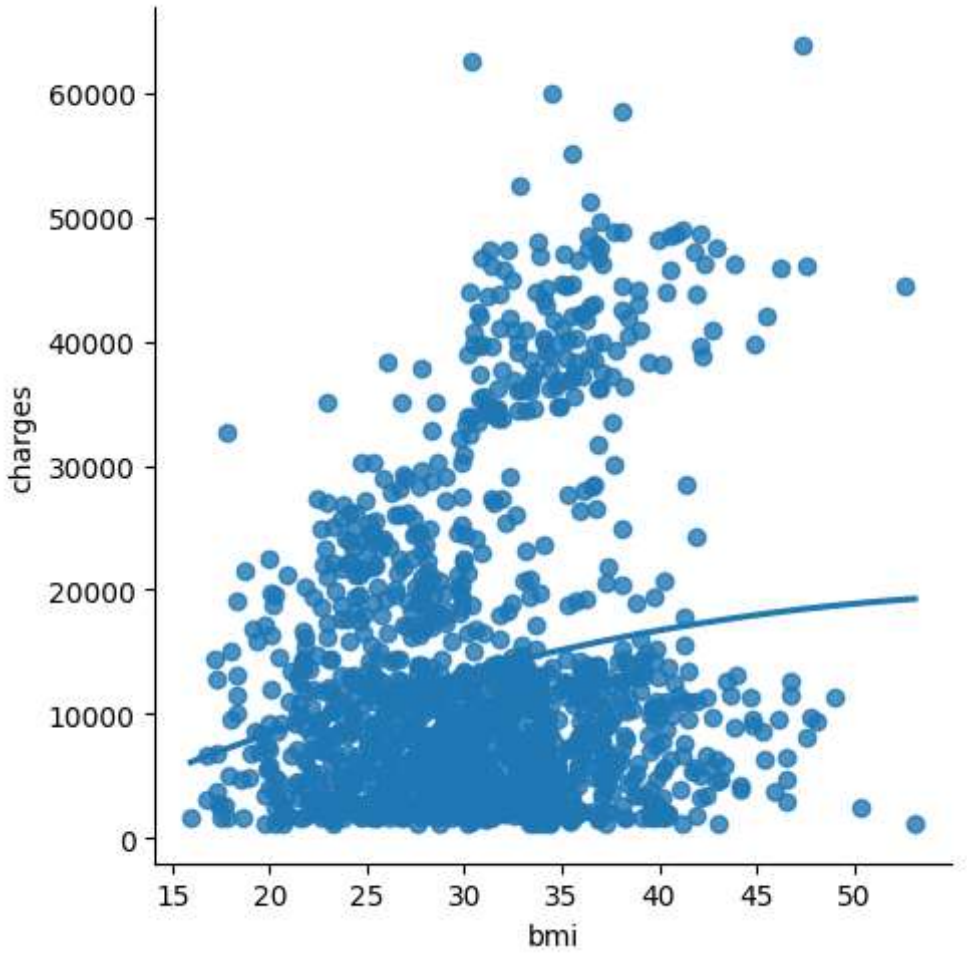
In [164]:
```
convert={"region":{"southwest":0,"northwest":1,"southeast":2,"northeast":3}}
df=df.replace(convert)
df
```

Out[164]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	0	16884.924000
1	18	0	33.770	1	0	2	1725.552300
2	28	0	33.000	3	0	2	4449.462000
3	33	0	22.705	0	0	1	21984.470610
4	32	0	28.880	0	0	1	3866.855200
5	31	1	25.740	0	0	2	3756.621600
6	46	1	33.440	1	0	2	8240.589600
7	37	1	27.740	3	0	1	7281.505600
8	37	0	29.830	2	0	3	6406.410700
9	60	1	25.840	0	0	1	28923.136920
10	25	0	26.220	0	0	3	2721.320800

3)Data Visualization

In [165]:
```
sns.lmplot(x='bmi',y='charges',order=2,data=df,ci=None)
plt.show()
```

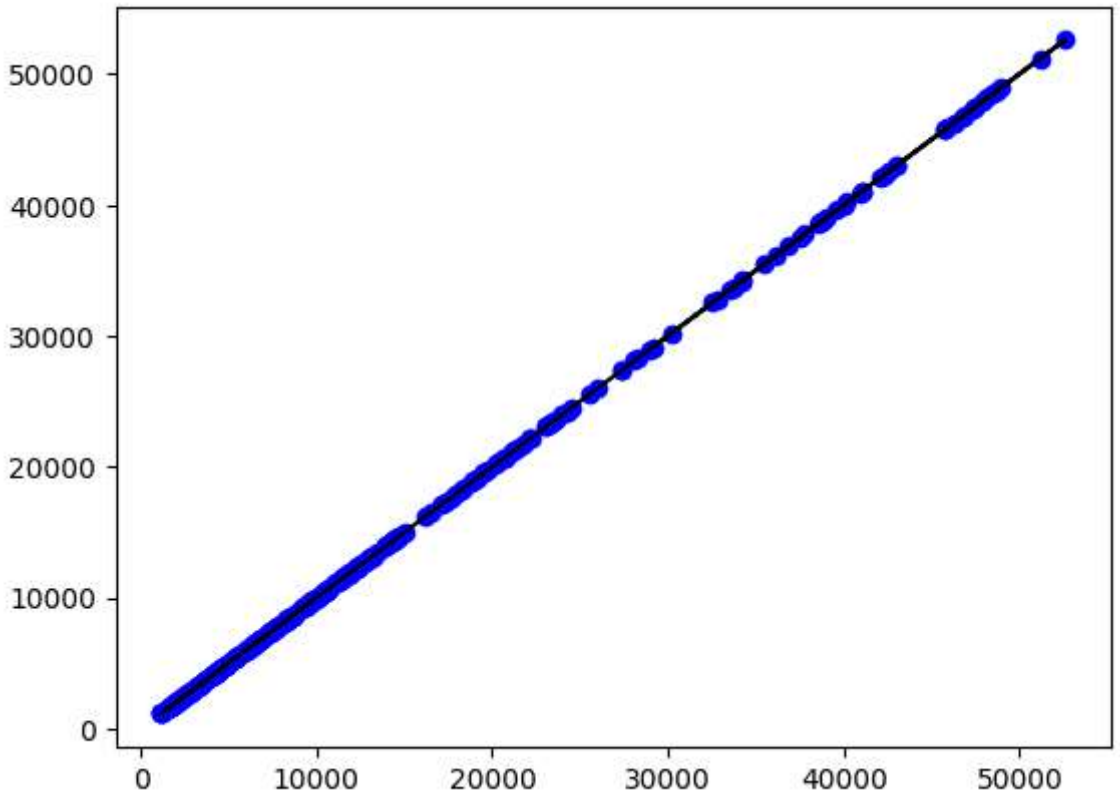


In [166]:
```
x=np.array(df['bmi']).reshape(-1,1)
y=x=np.array(df['charges']).reshape(-1,1)
```

In [167]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

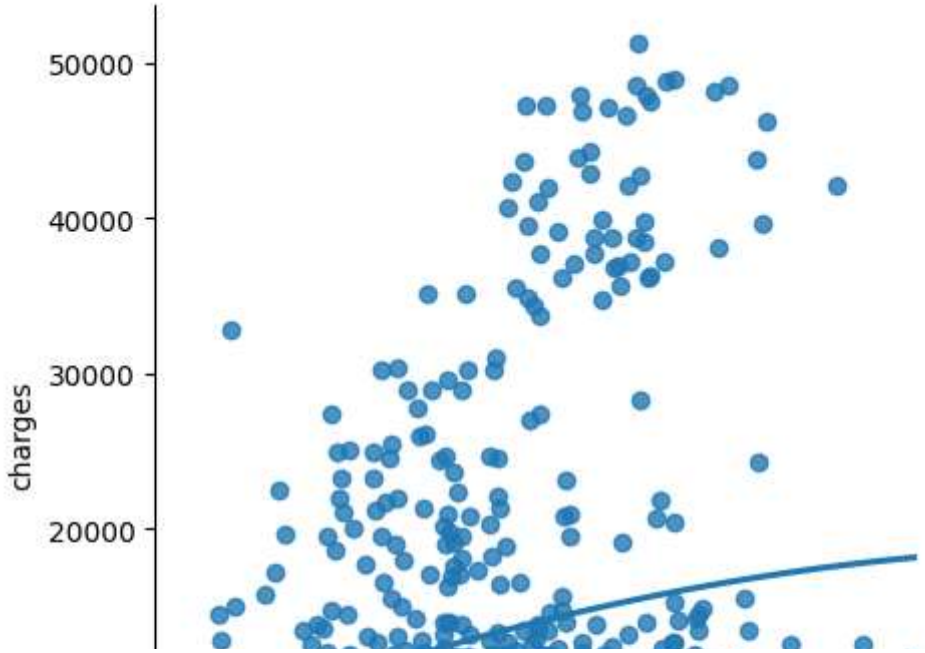
```
1.0
```

In [168]:
```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



working with subset of data

In [169]:
```
df500=df[:][:500]
sns.lmplot(x='bmi',y='charges',order=2,ci=None,data=df500)
plt.show()
```



In [170]:
```
df500.fillna(method='ffill',inplace=True)
```

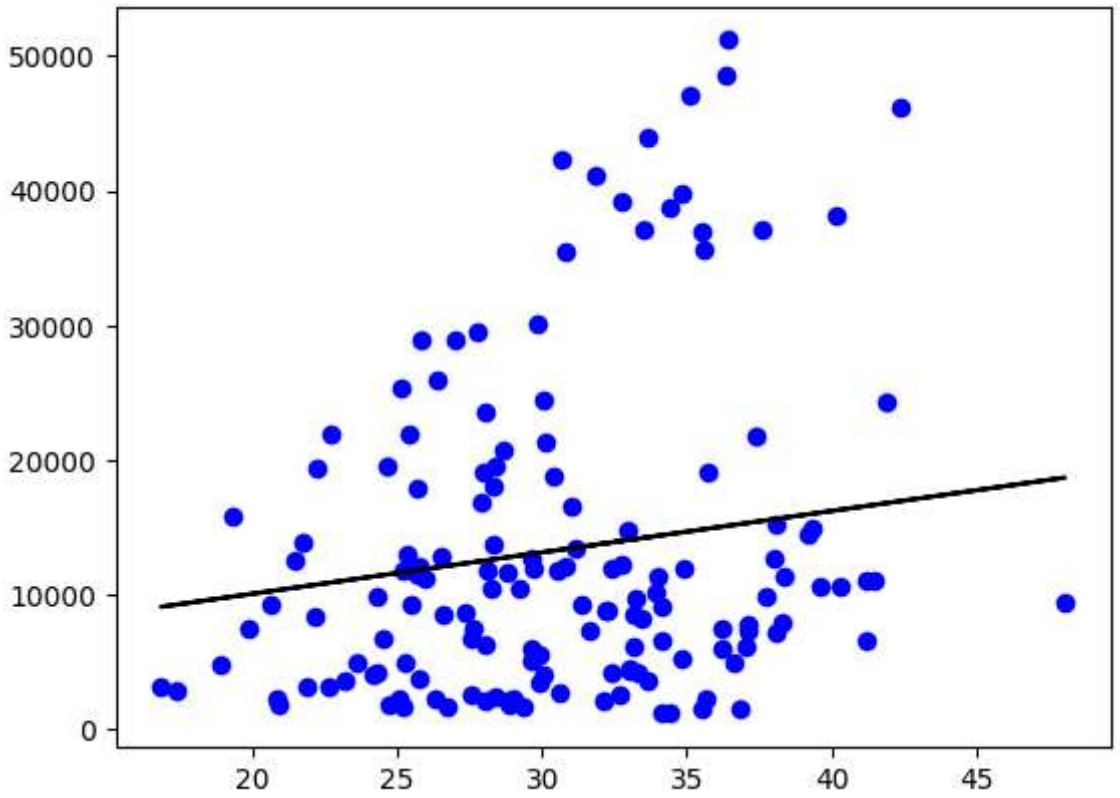
In [171]:
```
x=np.array(df500["bmi"]).reshape(-1,1)
y=np.array(df500['charges']).reshape(-1,1)
```

In [172]:
```
df500.dropna(inplace=True)
```

In [173]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

```
0.04771432155847122
```

In [174]:
```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



Evaluation of model

In [175]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [176]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
r2=r2_score(y_test,y_pred)
print(r2)
```

```
0.04771432155847122
```

Ridge Regression

In [177]:
```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [178]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df500.corr(),annot=True)
plt.show()
```

In [179]:
```
features=df.columns[0:1]
target=df.columns[-1]
x=df[features].values
y=df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state
print("The dimension of X_train is {}".format(x_train.shape))
print("The dimension of X_test is {}".format(x_test.shape))
```

```
The dimension of X_train is (936, 1)
The dimension of X_test is (402, 1)
```

In [180]:
```
lr = LinearRegression()
#Fit model
lr.fit(x_train, y_train)
#predict
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 0.0910963973805714
The test score for lr model is 0.08490473916580776
```

In [181]:
```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test scorefor ridge regression
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))
```

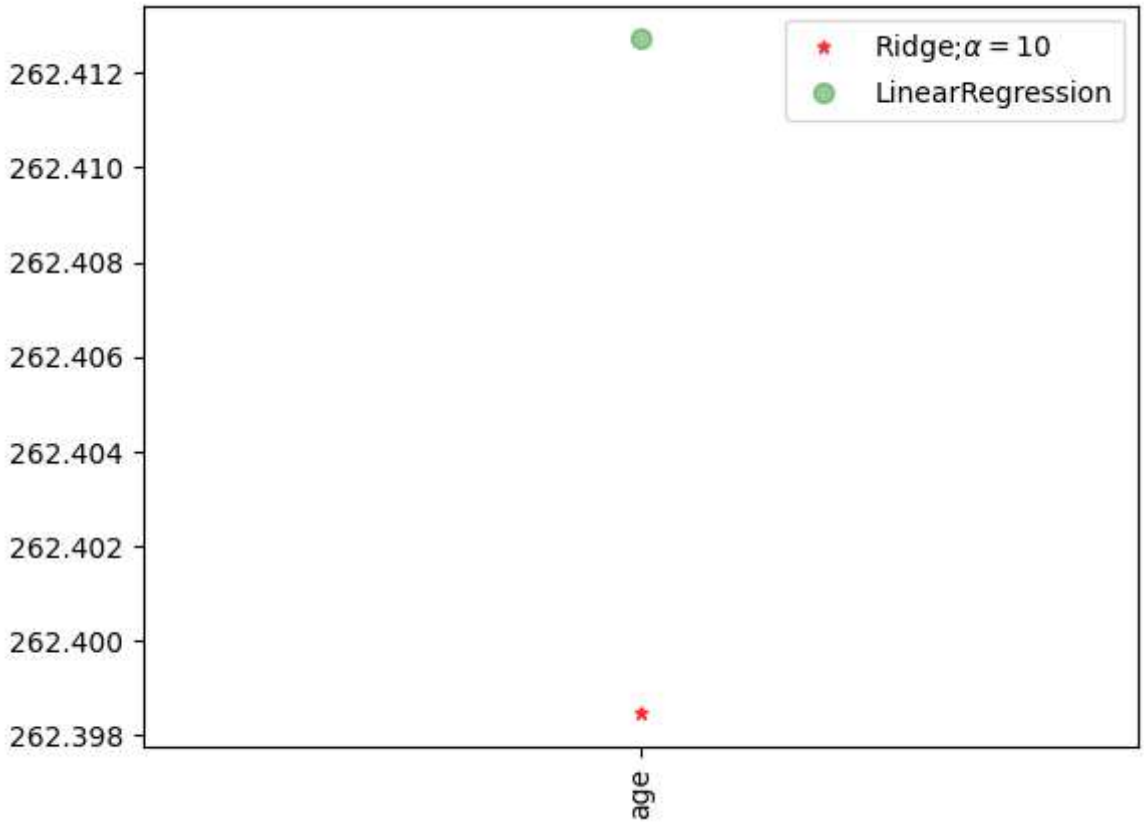
```
Ridge Model:

The train score for ridge model is 0.09109639711159634
The test score for ridge model is 0.08490538609860199
```

In [182]:
```
plt.figure(figsize=(10,10))
```

Out[182]:
```
<Figure size 1000x1000 with 0 Axes>
```

```
<Figure size 1000x1000 with 0 Axes>
```

In [183]:
```
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markers
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



Lasso Regression

In [184]:
```
lasso= Lasso(alpha=10)
lasso.fit(x_train,y_train)
#train and test scorefor ridge regression
train_score_ls = lasso.score(x_train, y_train)
test_score_ls= lasso.score(x_test, y_test)
print("\nRidge Model:\n")
print("The train score for lasso model is {}".format(train_score_ls))
print("The test score for lasso model is {}".format(test_score_ls))
```

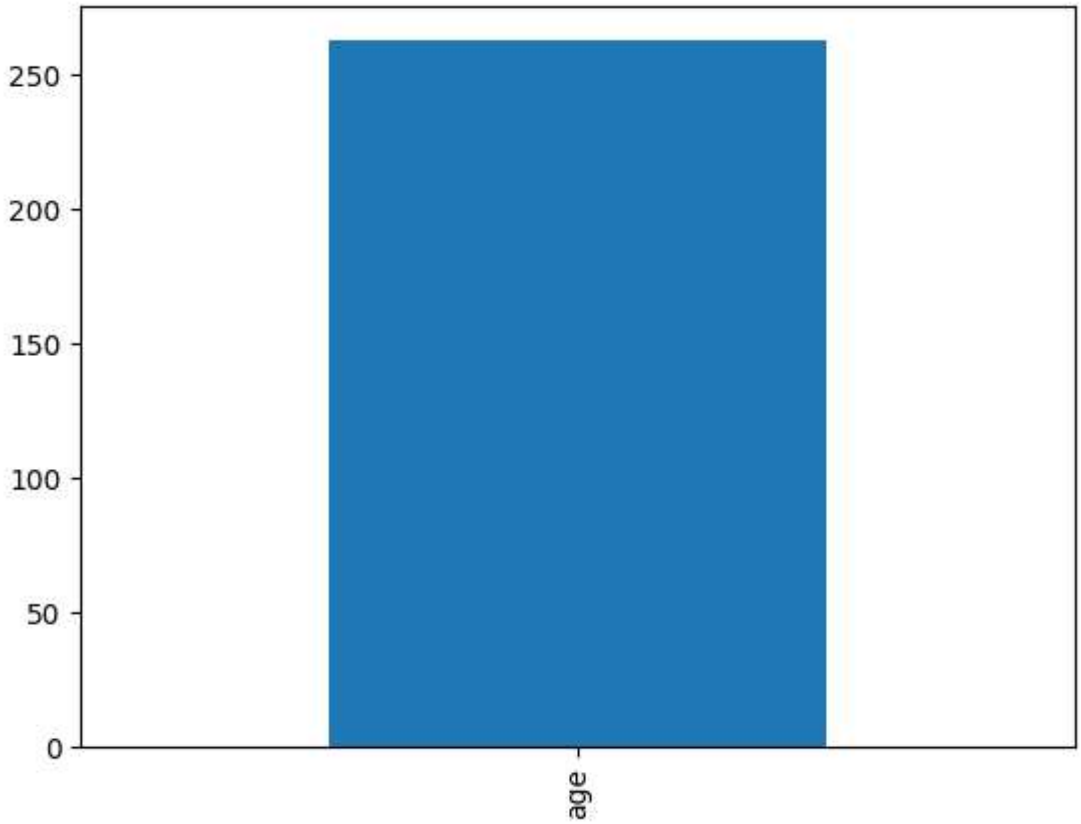
```
Ridge Model:

The train score for lasso model is 0.09109639395809044
The test score for lasso model is 0.08490704421828055
```

In [185]:
```
plt.figure(figsize=(10,10))
```

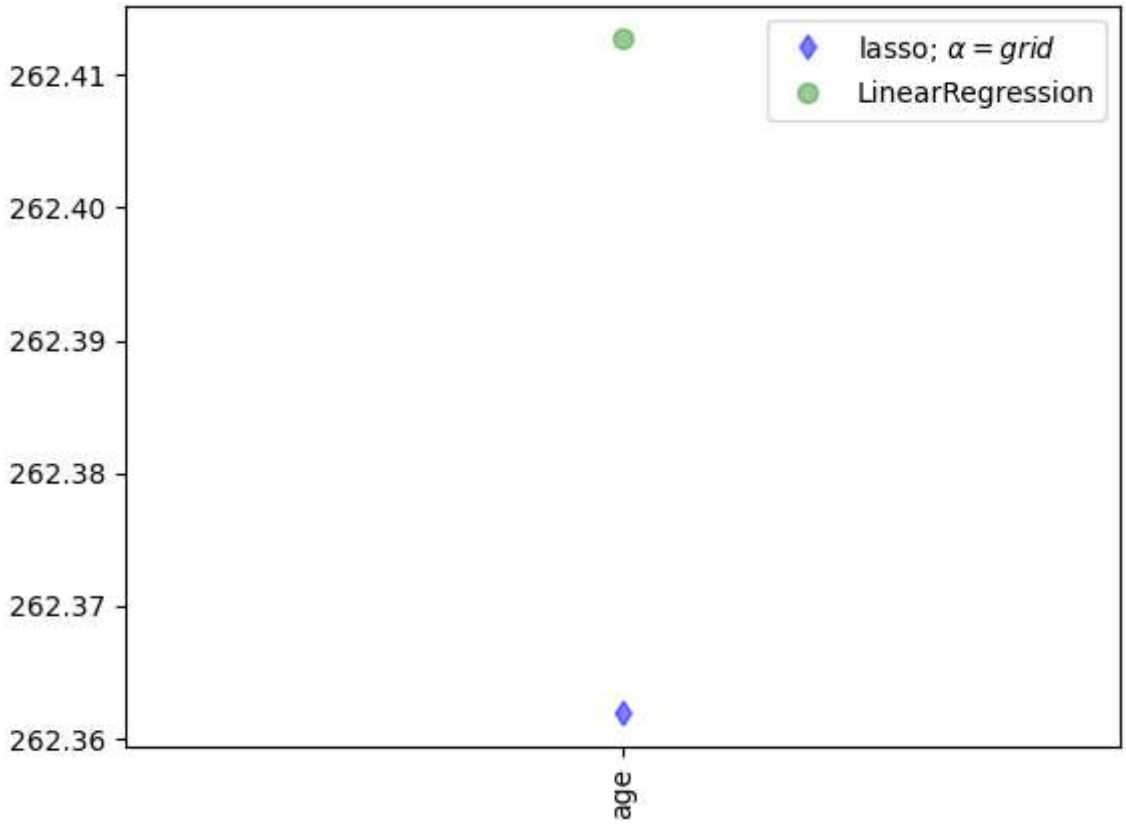
Out[185]:
```
<Figure size 1000x1000 with 0 Axes>
```

<Figure size 1000x1000 with 0 Axes>

In [186]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "ba
plt.show()
```



In [187]:
```
plt.plot(lasso.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.show()
```



In [188]:
```
#using the linear cv model
from sklearn.linear_model import RidgeCV
#cross validation
ridge_cv=RidgeCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(ridge_cv.score(x_train,y_train))
print(ridge_cv.score(x_test,y_test))
```

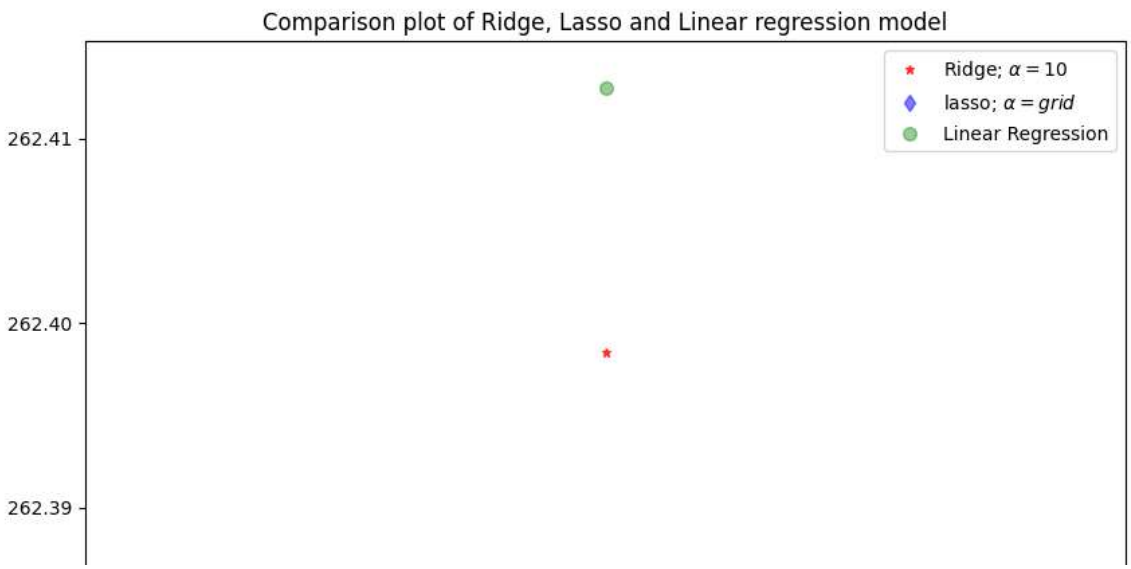
```
0.09109639711159612
0.08490538609885023
```

In [189]:
```
#using the linear cv model
from sklearn.linear_model import LassoCV
#cross validation
lasso_cv=LassoCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.09109639395809044
0.08490704421828055
```

In [190]:
```
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,col
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
```

Out[190]:
```
Text(0.5, 1.0, 'Comparison plot of Ridge, Lasso and Linear regression mode
l')
```



Elasticnet regression

In [191]:
```
from sklearn.linear_model import ElasticNet
el=ElasticNet()
el.fit(x_train,y_train)
print(el.coef_)
print(el.intercept_)
```

```
[261.74450967]
3115.0831774262424
```

In [192]:
```
y_pred_elastic=el.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
135077142.70714515
```

Logistic Regression

In [193]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [194]:
```
df=pd.read_csv(r"C:\Users\ubinl\Downloads\insurance.csv")
df
```

Out[194]:

	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 [195]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [196]:
```
print('This DataFrame has %d Rows and %d columns'%(df.shape))
```

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

In [197]:
```
convert={"smoker":{"yes":1,"no":2}}
df=df.replace(convert)
df
```

Out[197]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	1	southwest	16884.924000
1	18	male	33.770	1	2	southeast	1725.552300
2	28	male	33.000	3	2	southeast	4449.462000
3	33	male	22.705	0	2	northwest	21984.470610
4	32	male	28.880	0	2	northwest	3866.855200
5	31	female	25.740	0	2	southeast	3756.621600
6	46	female	33.440	1	2	southeast	8240.589600
7	37	female	27.740	3	2	northwest	7281.505600
8	37	male	29.830	2	2	northeast	6406.410700
9	60	female	25.840	0	2	northwest	28923.136920
10	25	male	26.220	0	2	northeast	2721.320800

In [198]:
```
convert={"sex":{"male":0,"female":1}}
df=df.replace(convert)
df
```

Out[198]:

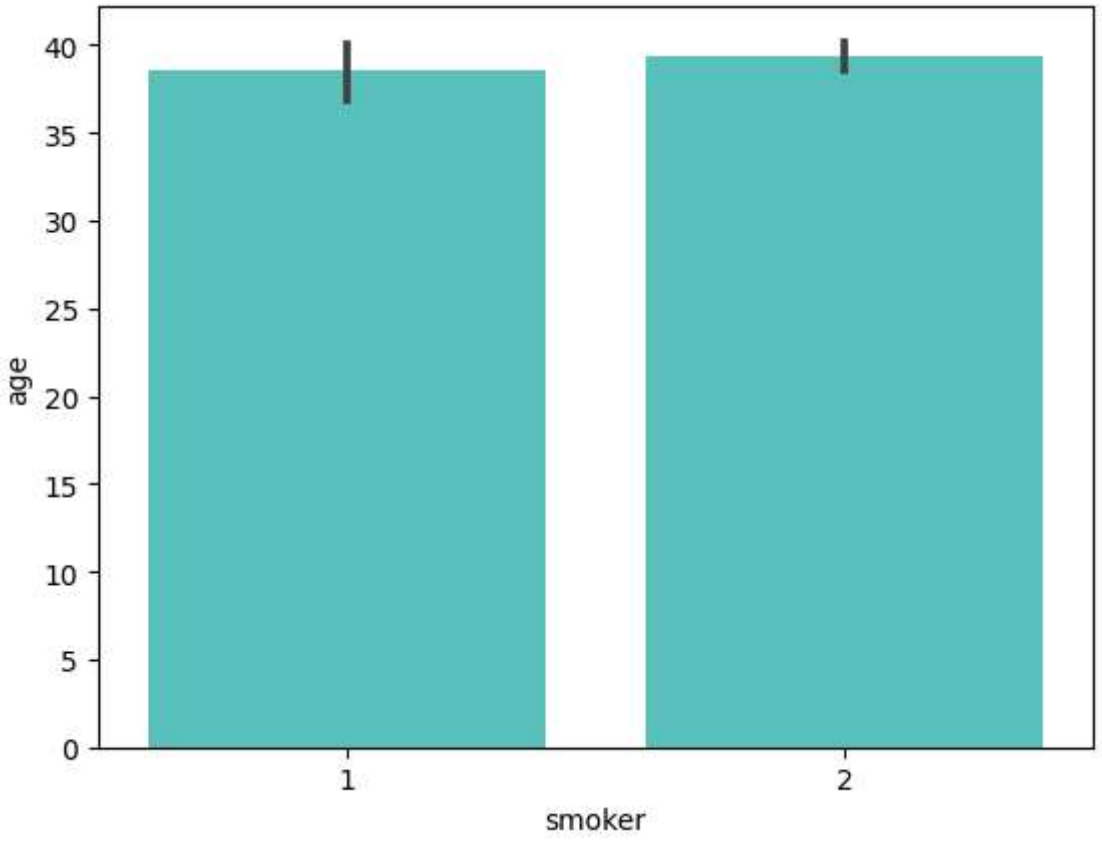
	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	southwest	16884.924000
1	18	0	33.770	1	2	southeast	1725.552300
2	28	0	33.000	3	2	southeast	4449.462000
3	33	0	22.705	0	2	northwest	21984.470610
4	32	0	28.880	0	2	northwest	3866.855200
5	31	1	25.740	0	2	southeast	3756.621600
6	46	1	33.440	1	2	southeast	8240.589600
7	37	1	27.740	3	2	northwest	7281.505600
8	37	0	29.830	2	2	northeast	6406.410700
9	60	1	25.840	0	2	northwest	28923.136920
10	25	0	26.220	0	2	northeast	2721.320800

In [199]:
```
features_matrix=df.iloc[:,0:4]
target_vector=df.iloc[:,-3]
print('The Features Matrix Has %d Rows And %d Column(s)'%(features_matrix.shap
print("The target matrix Has %d Rows And %d Column(s)"%np.array(target_vector)
```

```
The Features Matrix Has 1338 Rows And 4 Column(s)
The target matrix Has 1338 Rows And 1 Column(s)
```

In [200]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [201]:
```
sns.barplot(x='smoker',y='age',data=df,color='mediumturquoise')
plt.show()
```



In [202]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
algorithm=LogisticRegression(max_iter=10000)
```

In [203]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_ve
```

In [204]:
```
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 [2]
```

In [205]:
```
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: [1 2]
```

In [206]:
```
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 0.1942921563693959
```

In [207]:
```
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['0'] Is 0.1942921563693959
```

In [208]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
lerg=LogisticRegression()
lerg.fit(x,y)
print(lerg.score(x,y))
```

```
0.7952167414050823

C:\Users\ubinl\AppData\Local\Programs\Python\Python311\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

In [209]:
```
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 [210]:
```
df=pd.read_csv(r"C:\Users\ubinl\Downloads\insurance.csv")
df
```

Out[210]:

	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 [211]:
```
df['region'].value_counts()
```

Out[211]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [212]:
```
df['bmi'].value_counts()
```

Out[212]:
```
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 [213]:
```
convert={"sex":{"male":1,"female":0}}
df=df.replace(convert)
df
```

Out[213]:

	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 [214]:
```
x=["bmi","children"]
y=["yes","no"]
all_inputs=df[x]
all_classes=df["sex"]
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_s
```

In [215]:
```
clf=DecisionTreeClassifier(random_state=0)
clf.fit(x_train,y_train)
```

Out[215]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [216]:
```
score=clf.score(x_test,y_test)
print(score)
```

```
0.45671641791044776
```

Random Forest

In [217]:
```
import pandas as pd
import numpy as ny
import matplotlib.pyplot as plt,seaborn as sns
```

In [218]:
```
df=pd.read_csv(r"C:\Users\ubinl\Downloads\insurance.csv")
df
```

Out[218]:

	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 [219]:
```
df['charges'].value_counts()
```

Out[219]:
```
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 [220]:
```
m={"region":{"southeast":1,"southwest":2,"northeast":3,"northwest":4}}
df=df.replace(m)
print(df)
```

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

In [221]:
```
df.shape
```

Out[221]:
```
(1338, 7)
```

In [222]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[222]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [223]:
```
rf=RandomForestClassifier()
params={'max_depth':[2,3,5,10,20],'min_samples_leaf':[5,10,20,50,100,200],'n_e
```

In [224]:
```
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[224]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

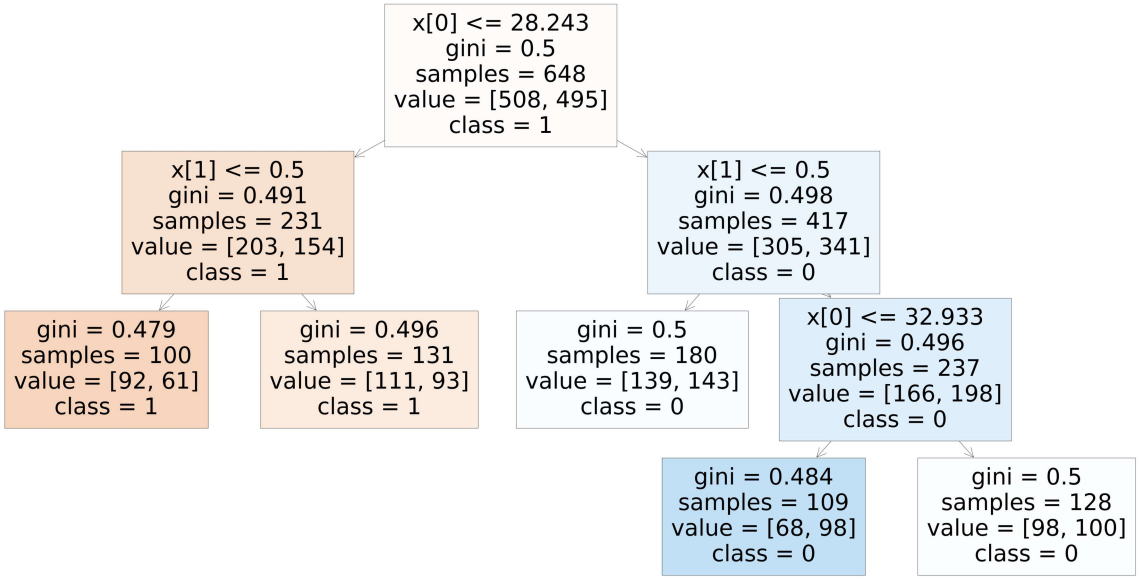
In [225]:
```
grid_search.best_score_
```

Out[225]:
```
0.5393694682348451
```

In [226]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=5, min_samples_leaf=100, n_estimators=25)
```

In [227]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['1','0'],filled=True);
```



In [228]:
```
rf_best.feature_importances_
```

Out[228]:
```
array([0.79907872, 0.20092128])
```

In [229]:
```
imp_df=pd.DataFrame({"Variance":x_train.columns,"Imp":rf_best.feature_importar
imp_df.sort_values(by="Imp",ascending=False)
```

Out[229]:

	Variance	Imp
0	bmi	0.799079
1	children	0.200921

In [230]:
```
rf=RandomForestClassifier(random_state=0)
```

In [231]:
```
rf.fit(x_train,y_train)
```

Out[231]:
```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

In [232]:
```
score=rfc.score(x_test,y_test)
print(score)
```

```
0.4626865671641791
```

/("""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" model best fits for the dat/""")