problem Statement:predict insurance charges and bmi age

1.Data Collection

In [1]:

```
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
```

In [2]:

```
df=pd.read_csv(r"C:\Users\monim\Downloads\insurance (1).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

2.Data cleaning and Preprocessing

In [3]:

```
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 [4]:

```
df.isna().sum()
```

Out[4]:

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [6]:

```
df.describe()
```

Out[6]:

	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 [7]:

```
df.shape
```

Out[7]:

```
(1338, 7)
```

In [8]:

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

Out[8]:

```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [9]:

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

Out[9]:

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

1338 rows × 7 columns

In [10]:

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

Out[10]:

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

1338 rows × 7 columns

In [11]:

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

Out[11]:

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

1338 rows × 7 columns

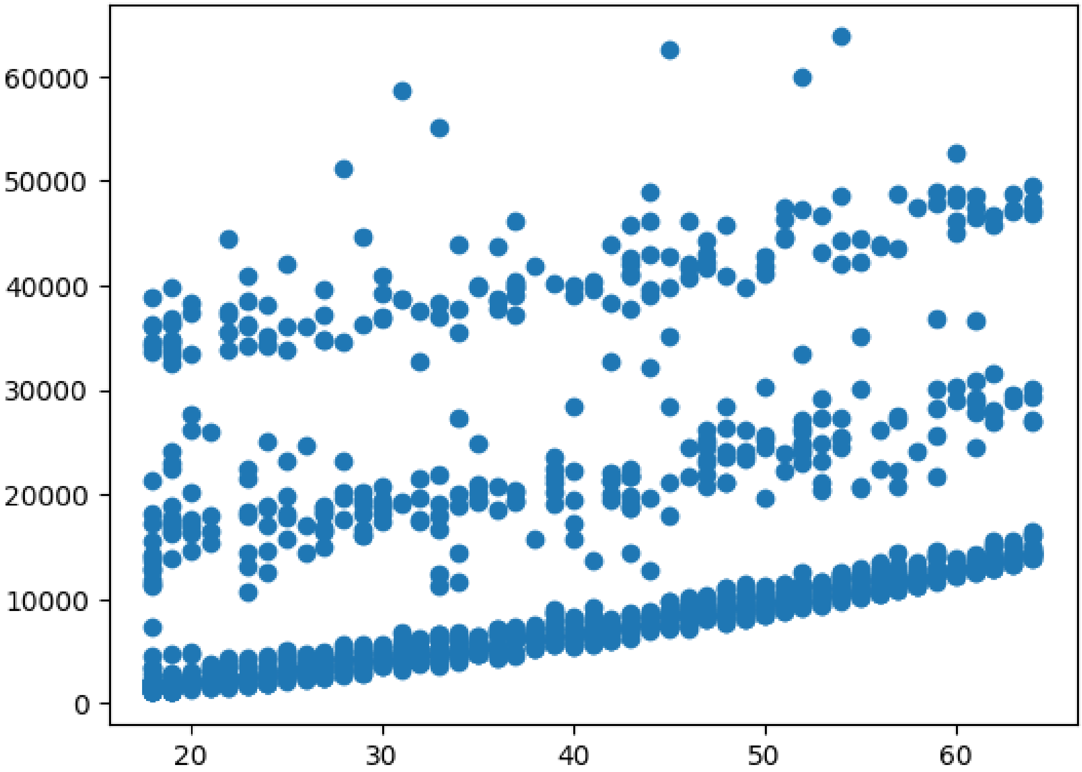
DATA VISUALIZATION

In [12]:

```
#Relationship between age and charges
plt.scatter(df['age'],df['charges'])
```

Out[12]:

```
<matplotlib.collections.PathCollection at 0x20e13292e30>
```

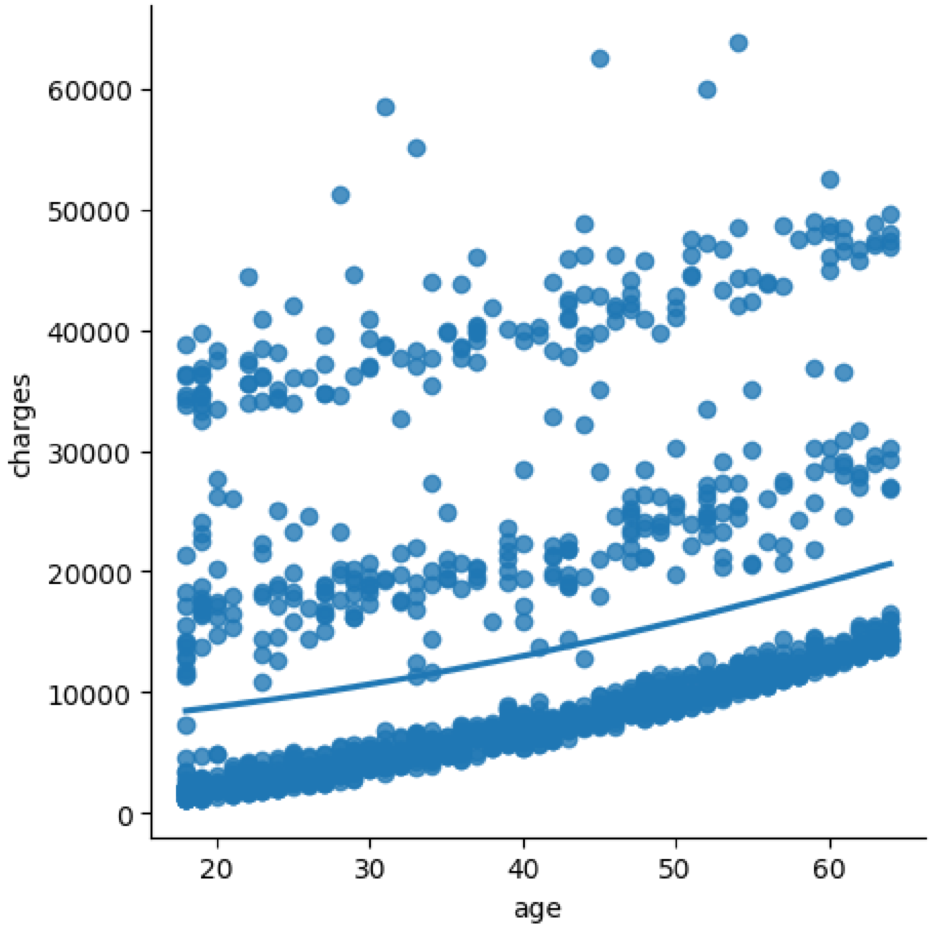


In [13]:

```
x=['age']
y=['charges']
```

In [14]:

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



In [15]:

```
df.fillna(method='ffill',inplace=True)
```

In [16]:

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

In [17]:

```
df.dropna(inplace=True)
```

splitting the data train and test

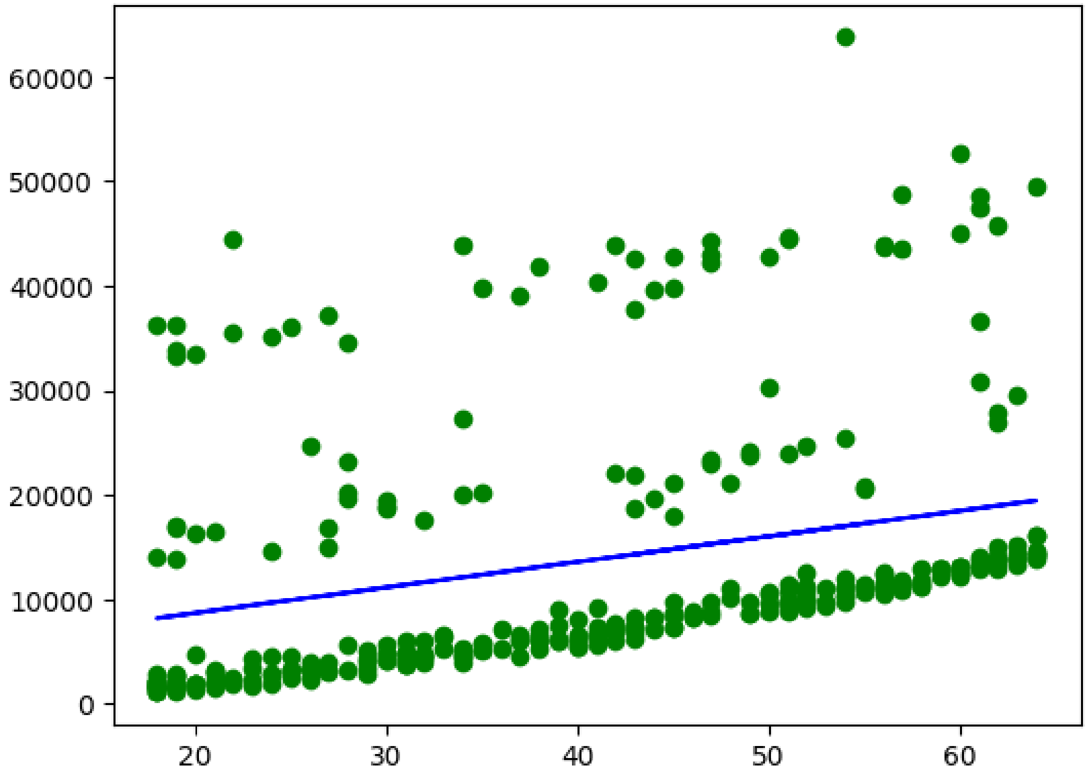
In [18]:

```
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))
```

```
0.11604973026137633
```

In [19]:

```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='green')
plt.plot(x_test,y_pred,color='b')
plt.show()
```

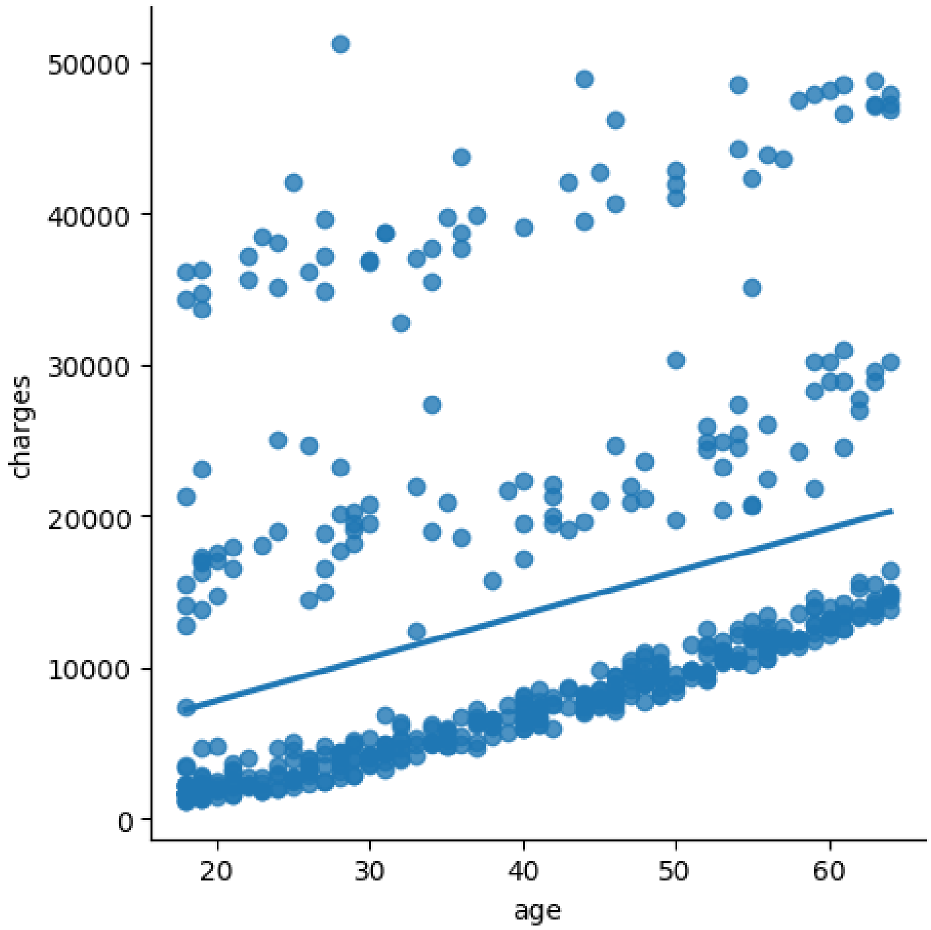


In [20]:

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

Out[20]:

```
<seaborn.axisgrid.FacetGrid at 0x20e133ef8b0>
```



In [21]:

```
df500.fillna(method='ffill',inplace=True)
```

EVALUATION OF MODEL

In [22]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[22]:

```
▾ LinearRegression
LinearRegression()
```

In [23]:

```
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2score:",r2)
```

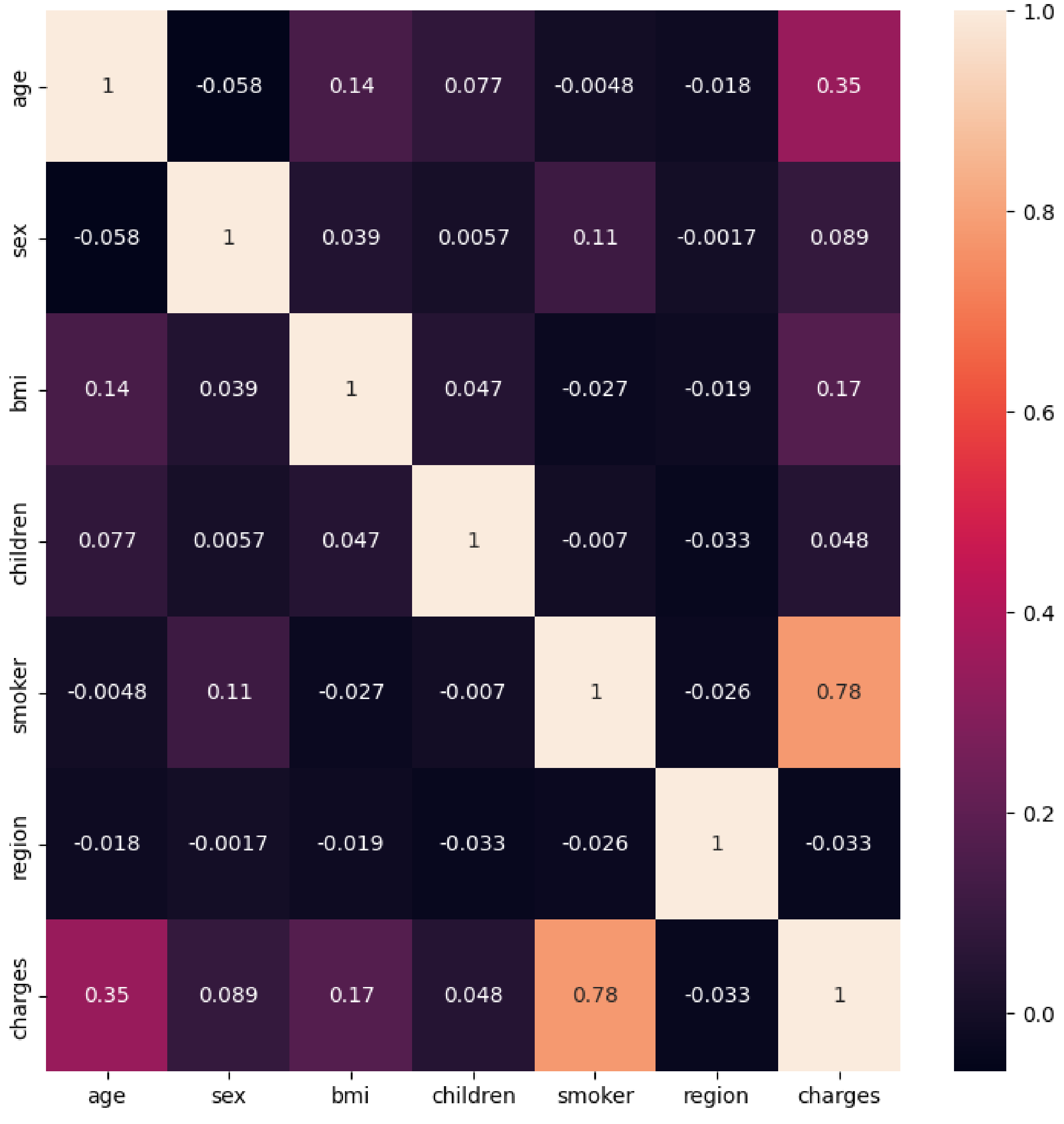
```
R2score: 0.11604973026137633
```

CONCULSION:The model is 4% it is worstfit

RIDGE REGRESSION

In [24]:

```
from sklearn.linear_model import  Lasso,Ridge
from sklearn.preprocessing import StandardScaler
plt.figure(figsize=(9,9))
sns.heatmap(df500.corr(),annot=True)
plt.show()
```



In [25]:

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

In [26]:

```
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)
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 [27]:

```
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("\nLinearRegression 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))
```

```

LinearRegression model:

The train score for lr model is 0.09414049248111778
The test score for lr model is 0.07333921956861744
```

RIDGE

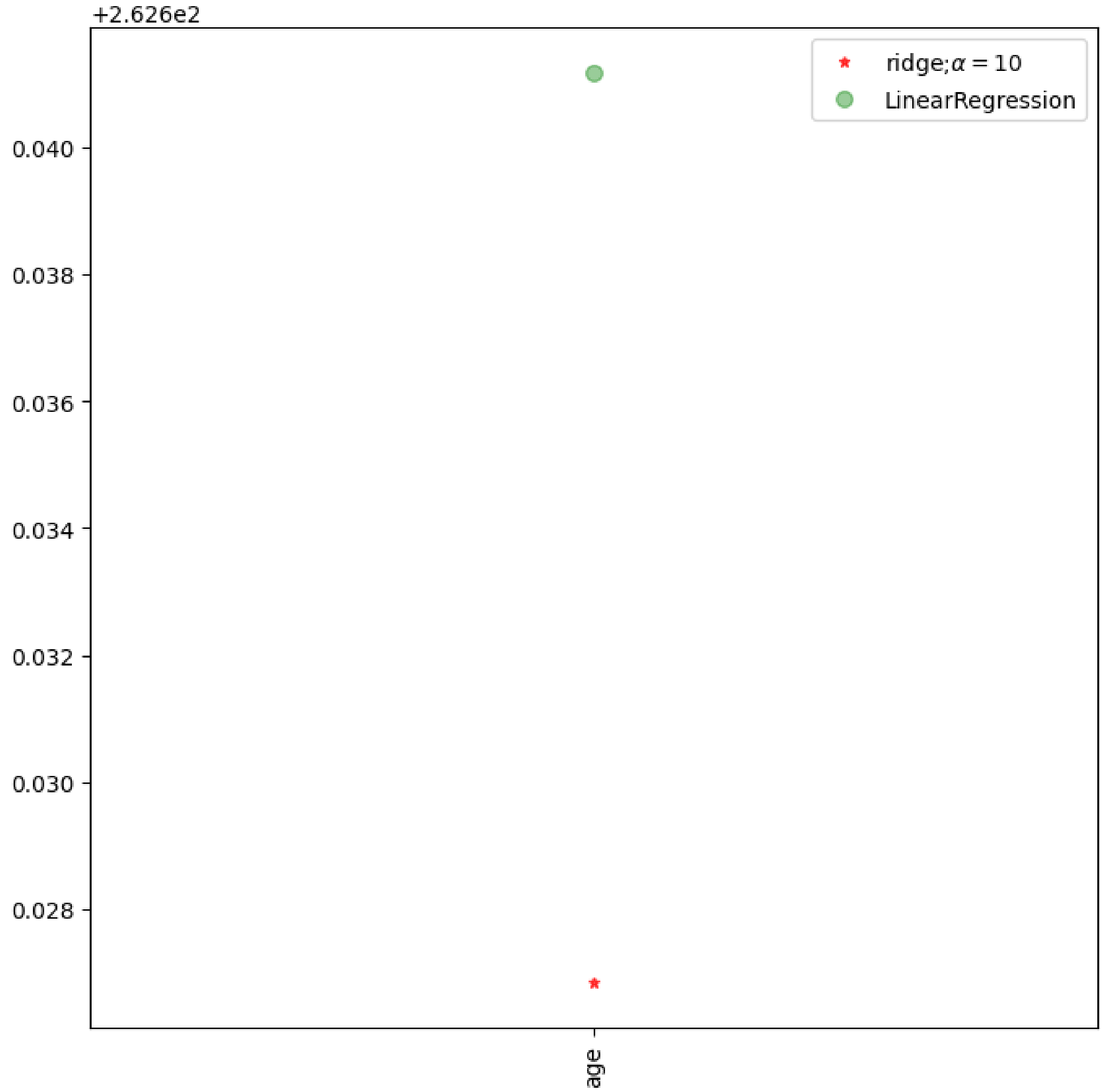
In [28]:

```
# Ridge Regression Model
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test score for ridge regression
train_score_ridge=ridgeReg.score(x_train,y_train)
test_score_ridge=ridgeReg.score(x_test,y_test)
print("\nRidge 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 is 0.09414049220130205
The test score for ridge model is 0.07333977758393473
```

In [29]:

```
plt.figure(figsize=(8,8))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='re
         label=r'ridge;$\alpha=10$',zorder=7)
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',la
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



LASSOCV

In [30]:

```
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))
```

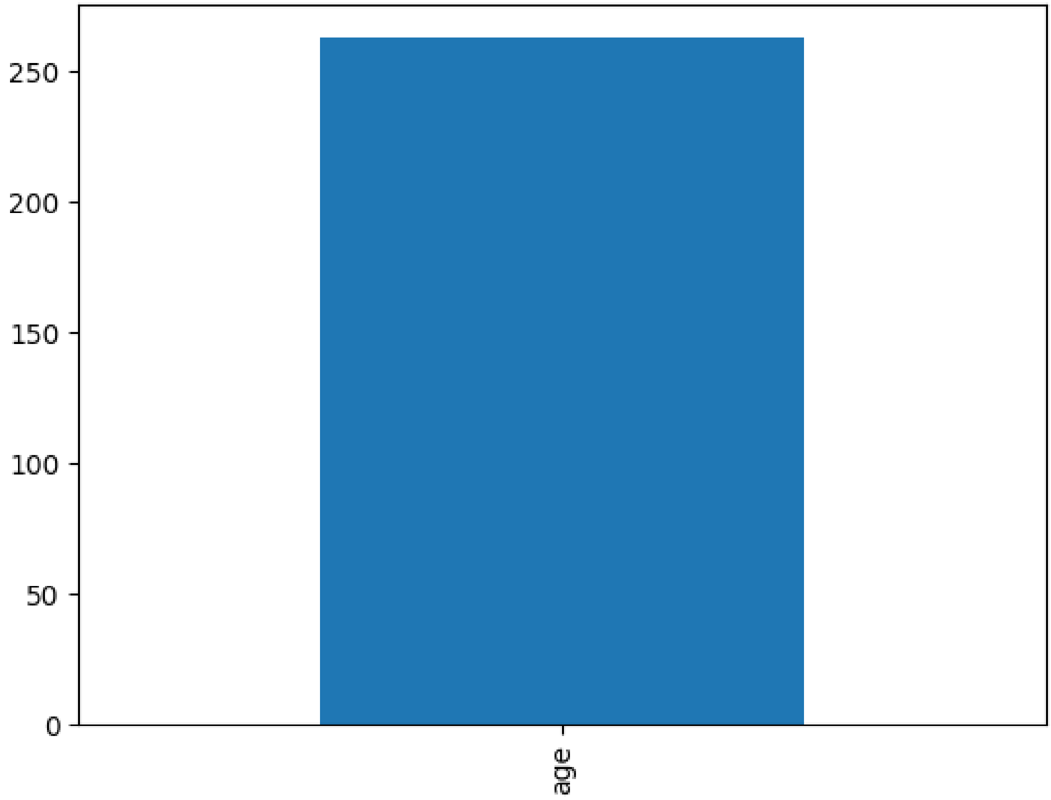
```
Lasso Model:

The train score for ls model is 0.09414048892688687
The test score for ls model is 0.07334120586832915
```

In [31]:

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

Out[31]:

```
<Axes: >
```



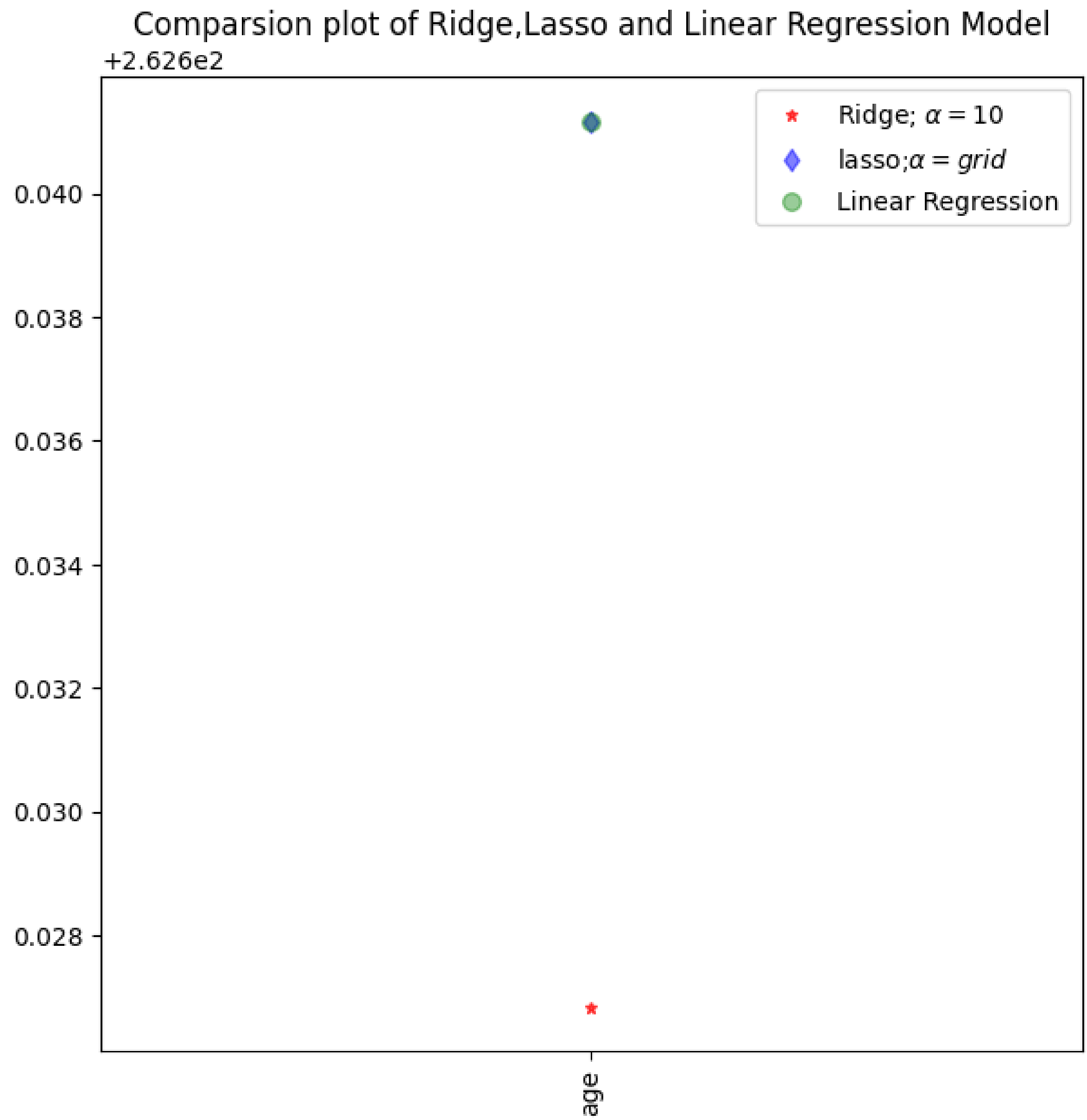
In [32]:

```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.09414049248111778
0.0733921958851453
```

In [33]:

```
#plot size
plt.figure(figsize = (7, 7))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='re
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',la
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparsion plot of Ridge,Lasso and Linear Regression Model")
plt.show()
```



In [34]:

```
#using the linear CV model
from sklearn.linear_model import RidgeCV

#Using the linear CV model
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_train)
#score
print("The train score for ridge model is {}".format(ridge_cv.score(x_train, y_train)))
print("The train score for ridge model is {}".format(ridge_cv.score(x_test, y_test)))
```

```
The train score for ridge model is 0.09414049220130227
The train score for ridge model is 0.07333977758374921
```

ELASTICNET

In [35]:

```
from sklearn.linear_model import ElasticNet
ne=ElasticNet()
ne.fit(x_train,y_train)
print(ne.coef_)
print(ne.intercept_)
```

```
[261.97015916]
2776.139448743348
```

In [36]:

```
y_pred_elastic=ne.predict(x_train)
```

In [37]:

```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
130065728.92376928
```

In [38]:

```
ne=ElasticNet()
ne.fit(x_train,y_train)
print(ne.score(x_train,y_train))
```

```
0.09413987801408741
```

CONCLUSION:The model has 9% accuracy

.....LOGISTIC REGRESSION.....

In [39]:

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

In [40]:

```
df=pd.read_csv(r"C:\Users\monim\Downloads\insurance (1).csv")
df
```

Out[40]:

	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 [41]:

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

In [42]:

```
print('this DataFrame has %d Rows and %d columns'%(df.shape))
```

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

In [43]:

```
df.head()
```

Out[43]:

	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 [44]:

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

Out[44]:

	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

In [45]:

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

Out[45]:

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

In [46]:

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

Out[46]:

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

In [47]:

```
features=df[['age','sex','bmi','region']]
features.columns=['age','sex','bmi','region']
target=df[['smoker']]
target.columns=['smoker']
```

In [48]:

```
print('The Features Matrix Has %d Rows And %d Columns(s)'%(features.shape))
```

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

In [49]:

```
features_standardized=StandardScaler().fit_transform(features)
```

In [50]:

```
algorithm=LogisticRegression(max_iter=1000)
```

In [55]:

```
Logistic_Regression_Model=algorithm.fit(features_standardized,target)
```

```
C:\Users\monim\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\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, ), fo
r example using ravel().
  y = column_or_1d(y, warn=True)
```

In [56]:

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

In [57]:

```
observation=[[1,0,0.99539,5]]
```

In [58]:

```
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 [1]
```

In [59]:

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

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

In [60]:

```
print("""The model says the probability of the observation we passed belonging to class['b']is
"""%(algorithm.predict_proba(observation)[0][0]))
print()
print("""The model says the probability of the observation we passed belonging to class['b']is
"""%(algorithm.predict_proba(observation)[0][1]))
```

```
The model says the probability of the observation we passed belonging to class
['b']is 0.29289566147251156


The model says the probability of the observation we passed belonging to class
['b']is 0.7071043385274884
```

In [61]:

```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-3,1)
```

In [62]:

```
lr=LogisticRegression()
lr.fit(x,y)
print(lr.score(x,y))
```

```
0.7952167414050823

C:\Users\monim\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\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, ), fo
r example using ravel().
  y = column_or_1d(y, warn=True)
```

In [63]:

```
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
plt.figure(figsize=(10,10))
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,random_state=17)
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)

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

In [64]:

```
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("\nLinearRegression 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))
```

```
LinearRegression model:

The train score for lr model is 0.07447061146193878
The test score for lr model is 0.10891203216512224
```

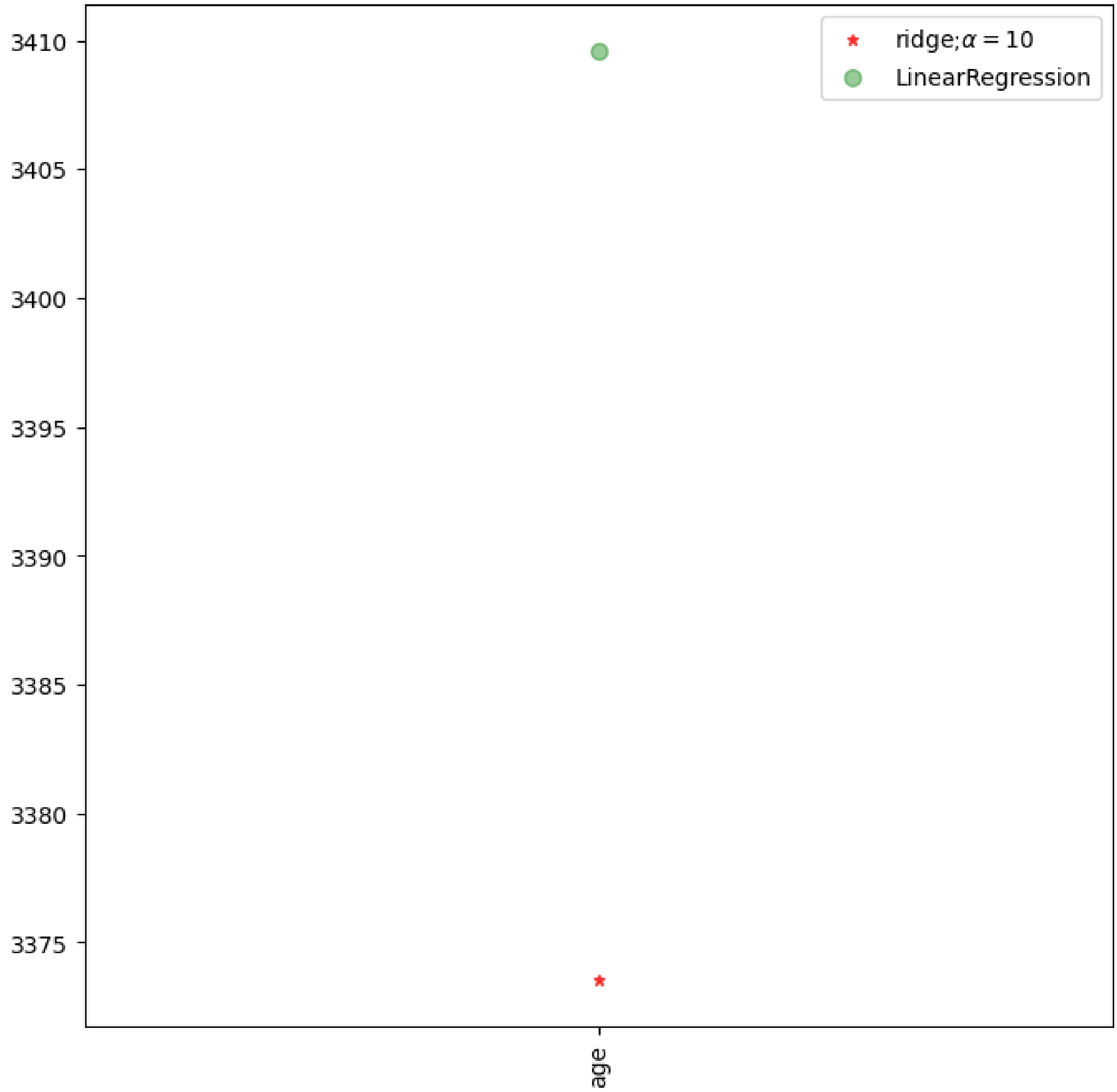
In [65]:

```
# Ridge Regression Model
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test score for ridge regression
train_score_ridge=ridgeReg.score(x_train,y_train)
test_score_ridge=ridgeReg.score(x_test,y_test)
print("\nRidge 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 is 0.07446228994221393
The test score for ridge model is 0.10855133360950642
```

In [66]:

```
plt.figure(figsize=(8,8))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='re
         label=r'ridge;$\alpha=10$',zorder=7)
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',la
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [67]:

```
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))
```

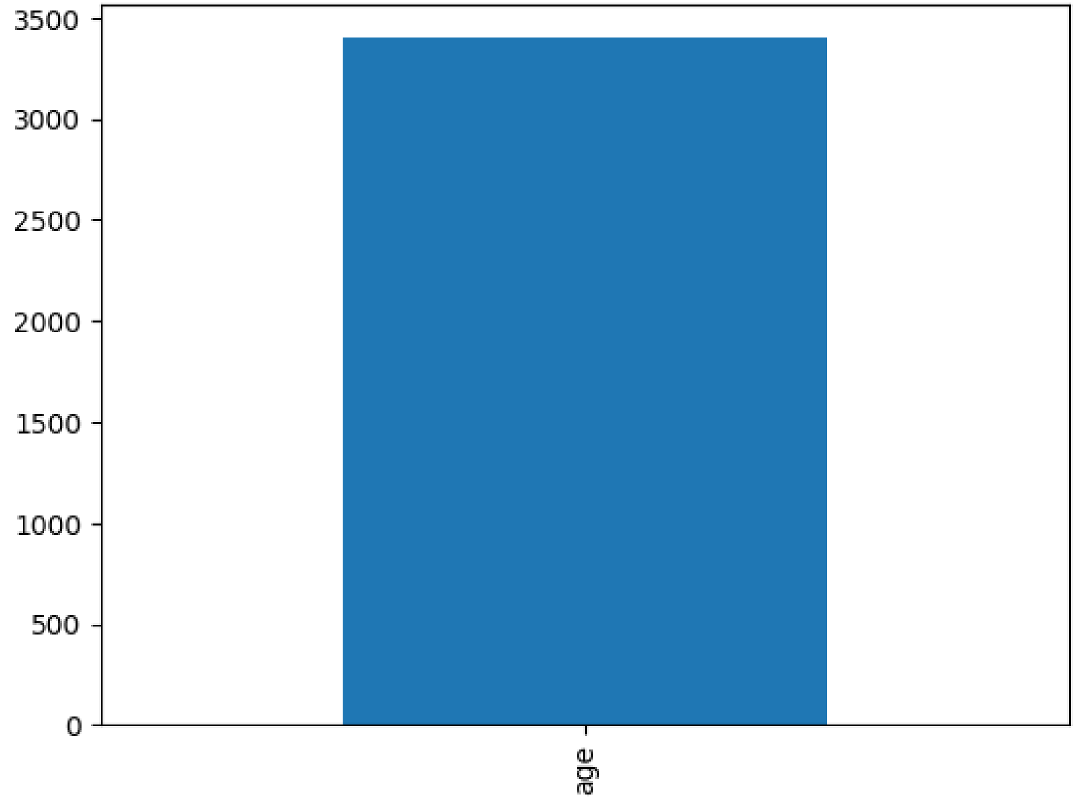
```
Lasso Model: 

The train score for ls model is 0.07446997086306062
The test score for ls model is 0.10881427793326703
```

In [68]:

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

Out[68]:

```
<Axes: >
```



In [69]:

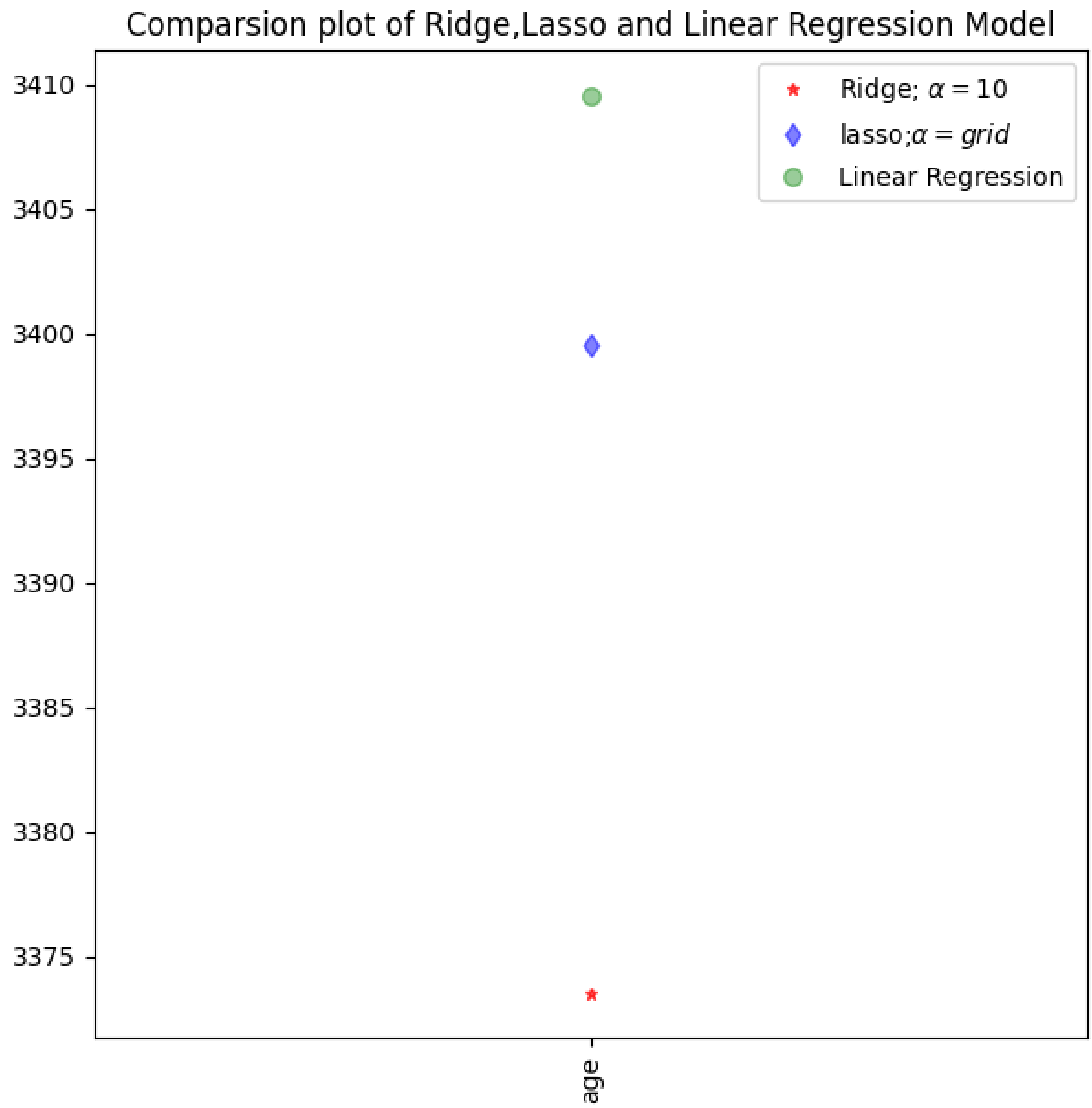
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.07446997086306062
0.10881427793326703
```

```
C:\Users\monim\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\linear_model\_coordinate_descent.py:1568: DataConversionWarning: A column-vec
tor 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)
```

In [70]:

```
#plot size
plt.figure(figsize = (7, 7))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='re
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',la
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparsion plot of Ridge,Lasso and Linear Regression Model")
plt.show()
```



In [71]:

```
#using the linear CV model
from sklearn.linear_model import RidgeCV

#Using the linear CV model
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_train)
#score
print("The train score for ridge model is {}".format(ridge_cv.score(x_train, y_train)))
print("The train score for ridge model is {}".format(ridge_cv.score(x_test, y_test)))
```

```
The train score for ridge model is 0.07446228994221393
The train score for ridge model is 0.10855133360950775
```

In [72]:

```
from sklearn.linear_model import ElasticNet
ne=ElasticNet()
ne.fit(x_train,y_train)
print(ne.coef_)
print(ne.intercept_)
```

```
[2272.71208683]
[13823.74618136]
```

In [73]:

```
y_pred_elastic=ne.predict(x_train)
```

In [74]:

```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
161269146.41663846
```

In [75]:

```
ne=ElasticNet()
ne.fit(x_train,y_train)
print(ne.score(x_train,y_train))
```

```
0.06619124466434823
```

DECISION TREE

In [76]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [77]:

```
x=["age","sex","bmi","children","region"]
y=["Yes","No"]
all_inputs=df[x]
all_classes=df["smoker"]
```

In [78]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.30)
```

In [79]:

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

In [81]:

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

Out[81]:

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

In [82]:

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

```
0.6965174129353234
```

RANDOM FOREST

In [83]:

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

In [84]:

```
x=df.drop('smoker',axis=1)
y=df['smoker']
```

In [85]:

```
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)
x_train.shape,x_test.shape
```

Out[85]:

```
((936, 6), (402, 6))
```

In [86]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[86]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [87]:

```
rf=RandomForestClassifier()
```

In [88]:

```
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 [89]:

```
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[89]:

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

In [90]:

```
grid_search.best_score_
```

Out[90]:

```
0.954059829059829
```

In [91]:

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

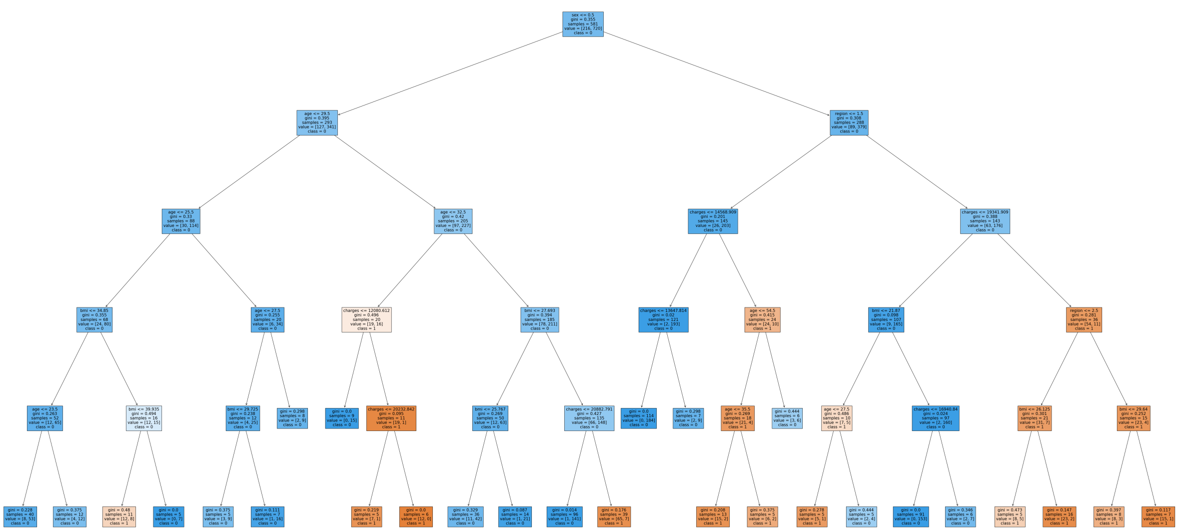
```
RandomForestClassifier(max_depth=5, min_samples_leaf=5, n_estimators=25)
```

In [92]:

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

Out[92]:

```
[Text(0.4947916666666667, 0.9166666666666666, 'sex <= 0.5\ngini = 0.355\nsample
s = 581\nvalue = [216, 720]\nclass = 0'),
 Text(0.2708333333333333, 0.75, 'age <= 29.5\ngini = 0.395\nsamples = 293\nvalu
e = [127, 341]\nclass = 0'),
 Text(0.15625, 0.5833333333333334, 'age <= 25.5\ngini = 0.33\nsamples = 88\nval
ue = [30, 114]\nclass = 0'),
 Text(0.08333333333333333, 0.4166666666666667, 'bmi <= 34.85\ngini = 0.355\nsam
ples = 68\nvalue = [24, 80]\nclass = 0'),
 Text(0.041666666666666664, 0.25, 'age <= 23.5\ngini = 0.263\nsamples = 52\nval
ue = [12, 65]\nclass = 0'),
 Text(0.020833333333333332, 0.08333333333333333, 'gini = 0.228\nsamples = 40\nv
alue = [8, 53]\nclass = 0'),
 Text(0.0625, 0.08333333333333333, 'gini = 0.375\nsamples = 12\nvalue = [4, 12]
\nclass = 0'),
 Text(0.125, 0.25, 'bmi <= 39.935\ngini = 0.494\nsamples = 16\nvalue = [12, 15]
\nclass = 0'),
 Text(0.10416666666666667, 0.08333333333333333, 'gini = 0.48\nsamples = 11\nval
ue = [12, 8]\nclass = 1'),
 Text(0.14583333333333334, 0.08333333333333333, 'gini = 0.0\nsamples = 5\nvalue
= [0, 7]\nclass = 0'),
 Text(0.22916666666666666, 0.4166666666666667, 'age <= 27.5\ngini = 0.255\nsamp
les = 20\nvalue = [6, 34]\nclass = 0'),
 Text(0.20833333333333334, 0.25, 'bmi <= 29.725\ngini = 0.238\nsamples = 12\nva
lue = [4, 25]\nclass = 0'),
 Text(0.1875, 0.08333333333333333, 'gini = 0.375\nsamples = 5\nvalue = [3, 9]\n
class = 0'),
 Text(0.22916666666666666, 0.08333333333333333, 'gini = 0.111\nsamples = 7\nval
ue = [1, 16]\nclass = 0'),
 Text(0.25, 0.25, 'gini = 0.298\nsamples = 8\nvalue = [2, 9]\nclass = 0'),
 Text(0.3854166666666667, 0.5833333333333334, 'age <= 32.5\ngini = 0.42\nsample
s = 205\nvalue = [97, 227]\nclass = 0'),
 Text(0.3125, 0.4166666666666667, 'charges <= 12080.612\ngini = 0.496\nsamples
= 20\nvalue = [19, 16]\nclass = 1'),
 Text(0.2916666666666667, 0.25, 'gini = 0.0\nsamples = 9\nvalue = [0, 15]\nclas
s = 0'),
 Text(0.3333333333333333, 0.25, 'charges <= 20232.842\ngini = 0.095\nsamples =
11\nvalue = [19, 1]\nclass = 1'),
 Text(0.3125, 0.08333333333333333, 'gini = 0.219\nsamples = 5\nvalue = [7, 1]\n
class = 1'),
 Text(0.3541666666666667, 0.08333333333333333, 'gini = 0.0\nsamples = 6\nvalue
= [12, 0]\nclass = 1'),
 Text(0.4583333333333333, 0.4166666666666667, 'bmi <= 27.693\ngini = 0.394\nsam
ples = 185\nvalue = [78, 211]\nclass = 0'),
 Text(0.4166666666666667, 0.25, 'bmi <= 25.767\ngini = 0.269\nsamples = 50\nval
```



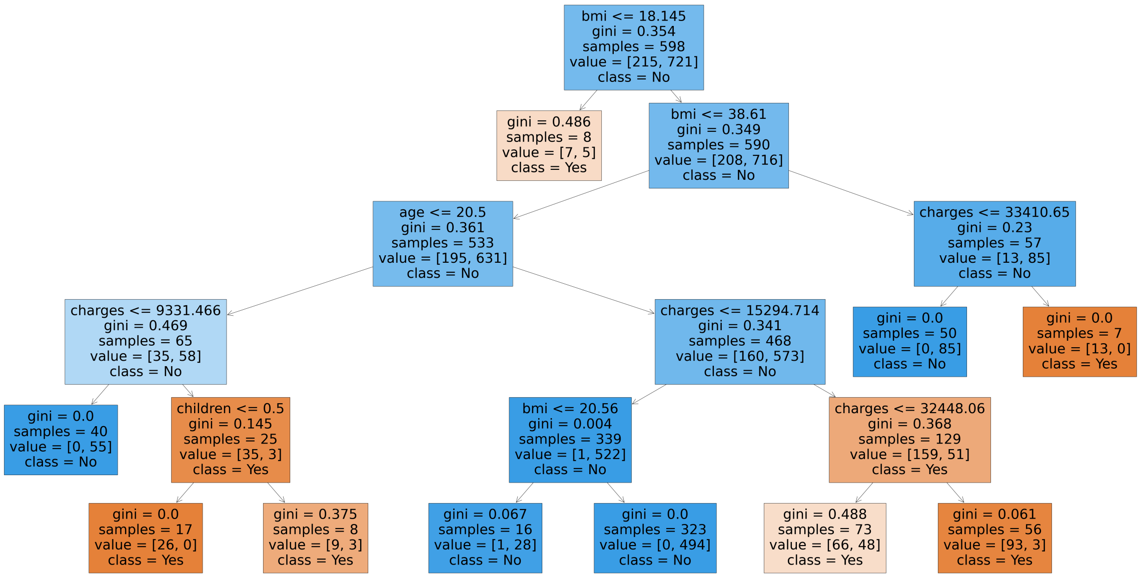
```
lass = 0'),
 Text(0.5833333333333334, 0.25, 'gini = 0.298\nsamples = 7\nvalue = [2, 9]\ncla
ss = 0'),
 Text(0.6458333333333334, 0.4166666666666667, 'age <= 54.5\ngini = 0.415\nsampl
es = 24\nvalue = [24, 10]\nclass = 1'),
 Text(0.625, 0.25, 'age <= 35.5\ngini = 0.269\nsamples = 18\nvalue = [21, 4]\nc
lass = 1'),
 Text(0.6041666666666666, 0.08333333333333333, 'gini = 0.208\nsamples = 13\nval
ue = [15, 2]\nclass = 1'),
 Text(0.6458333333333334, 0.08333333333333333, 'gini = 0.375\nsamples = 5\nvalu
e = [6, 2]\nclass = 1'),
 Text(0.6666666666666666, 0.25, 'gini = 0.444\nsamples = 6\nvalue = [3, 6]\ncla
ss = 0'),
 Text(0.8333333333333334, 0.5833333333333334, 'charges <= 19341.909\ngini = 0.3
88\nsamples = 143\nvalue = [63, 176]\nclass = 0'),
 Text(0.75, 0.4166666666666667, 'bmi <= 21.87\ngini = 0.098\nsamples = 107\nval
ue = [9, 165]\nclass = 0'),
 Text(0.7083333333333334, 0.25, 'age <= 27.5\ngini = 0.486\nsamples = 10\nvalue
= [7, 5]\nclass = 1'),
 Text(0.6875, 0.08333333333333333, 'gini = 0.278\nsamples = 5\nvalue = [5, 1]\n
class = 1'),
 Text(0.7291666666666666, 0.08333333333333333, 'gini = 0.444\nsamples = 5\nvalu
e = [2, 4]\nclass = 0'),
 Text(0.7916666666666666, 0.25, 'charges <= 16940.84\ngini = 0.024\nsamples = 9
7\nvalue = [2, 160]\nclass = 0'),
 Text(0.7708333333333334, 0.08333333333333333, 'gini = 0.0\nsamples = 91\nvalue
= [0, 153]\nclass = 0'),
 Text(0.8125, 0.08333333333333333, 'gini = 0.346\nsamples = 6\nvalue = [2, 7]\n
class = 0'),
 Text(0.9166666666666666, 0.4166666666666667, 'region <= 2.5\ngini = 0.281\nsam
ples = 36\nvalue = [54, 11]\nclass = 1'),
 Text(0.875, 0.25, 'bmi <= 26.125\ngini = 0.301\nsamples = 21\nvalue = [31, 7]
\nclass = 1'),
 Text(0.8541666666666666, 0.08333333333333333, 'gini = 0.473\nsamples = 5\nvalu
e = [8, 5]\nclass = 1'),
 Text(0.8958333333333334, 0.08333333333333333, 'gini = 0.147\nsamples = 16\nval
ue = [23, 2]\nclass = 1'),
 Text(0.9583333333333334, 0.25, 'bmi <= 29.64\ngini = 0.252\nsamples = 15\nvalu
e = [23, 4]\nclass = 1'),
 Text(0.9375, 0.08333333333333333, 'gini = 0.397\nsamples = 8\nvalue = [8, 3]\n
class = 1'),
 Text(0.9791666666666666, 0.08333333333333333, 'gini = 0.117\nsamples = 7\nvalu
e = [15, 1]\nclass = 1')]
```

In [93]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=x.columns,class_names=["Yes","No"],filled=True
```

Out[93]:

```
[Text(0.5535714285714286, 0.9166666666666666, 'bmi <= 18.145\ngini = 0.354\nsam
ples = 598\nvalue = [215, 721]\nclass = No'),
 Text(0.48214285714285715, 0.75, 'gini = 0.486\nsamples = 8\nvalue = [7, 5]\ncl
ass = Yes'),
 Text(0.625, 0.75, 'bmi <= 38.61\ngini = 0.349\nsamples = 590\nvalue = [208, 71
6]\nclass = No'),
 Text(0.39285714285714285, 0.5833333333333334, 'age <= 20.5\ngini = 0.361\nsamp
les = 533\nvalue = [195, 631]\nclass = No'),
 Text(0.14285714285714285, 0.4166666666666667, 'charges <= 9331.466\ngini = 0.4
69\nsamples = 65\nvalue = [35, 58]\nclass = No'),
 Text(0.07142857142857142, 0.25, 'gini = 0.0\nsamples = 40\nvalue = [0, 55]\ncl
ass = No'),
 Text(0.21428571428571427, 0.25, 'children <= 0.5\ngini = 0.145\nsamples = 25\n
value = [35, 3]\nclass = Yes'),
 Text(0.14285714285714285, 0.08333333333333333, 'gini = 0.0\nsamples = 17\nvalu
e = [26, 0]\nclass = Yes'),
 Text(0.2857142857142857, 0.08333333333333333, 'gini = 0.375\nsamples = 8\nvalu
e = [9, 3]\nclass = Yes'),
 Text(0.6428571428571429, 0.4166666666666667, 'charges <= 15294.714\ngini = 0.3
41\nsamples = 468\nvalue = [160, 573]\nclass = No'),
 Text(0.5, 0.25, 'bmi <= 20.56\ngini = 0.004\nsamples = 339\nvalue = [1, 522]\n
class = No'),
 Text(0.42857142857142855, 0.08333333333333333, 'gini = 0.067\nsamples = 16\nva
lue = [1, 28]\nclass = No'),
 Text(0.5714285714285714, 0.08333333333333333, 'gini = 0.0\nsamples = 323\nvalu
e = [0, 494]\nclass = No'),
 Text(0.7857142857142857, 0.25, 'charges <= 32448.06\ngini = 0.368\nsamples = 1
29\nvalue = [159, 51]\nclass = Yes'),
 Text(0.7142857142857143, 0.08333333333333333, 'gini = 0.488\nsamples = 73\nval
ue = [66, 48]\nclass = Yes'),
 Text(0.8571428571428571, 0.08333333333333333, 'gini = 0.061\nsamples = 56\nval
ue = [93, 3]\nclass = Yes'),
 Text(0.8571428571428571, 0.5833333333333334, 'charges <= 33410.65\ngini = 0.23
\nsamples = 57\nvalue = [13, 85]\nclass = No'),
 Text(0.7857142857142857, 0.4166666666666667, 'gini = 0.0\nsamples = 50\nvalue
= [0, 85]\nclass = No'),
 Text(0.9285714285714286, 0.4166666666666667, 'gini = 0.0\nsamples = 7\nvalue =
[13, 0]\nclass = Yes')]
```



istic
erving the
uracy