Problem statement: To Predict How Best the Data Fits and To Pridect the Insurance Based on the given Features

1)Data collection

#Importing Libraries

In [1]:
```
#Importing libraries
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 [2]:
```
#Reading data
df=pd.read_csv(r"C:\Users\Jayadeep\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

2)Data cleaning and Preprocessing

#Exploratory data anlysis

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

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

In [5]:
```
df.shape
```

Out[5]:
```
(1338, 7)
```

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

Out[6]:
```
<bound method NDFrame.describe of       age     sex     bmi  children smoker     region      charges
0      19  female  27.900         0    yes  southwest  16884.92400
1      18    male  33.770         1     no  southeast   1725.55230
2      28    male  33.000         3     no  southeast   4449.46200
3      33    male  22.705         0     no  northwest  21984.47061
4      32    male  28.880         0     no  northwest   3866.85520
...   ...     ...     ...       ...    ...        ...          ...
1333   50    male  30.970         3     no  northwest  10600.54830
1334   18  female  31.920         0     no  northeast   2205.98080
1335   18  female  36.850         0     no  southeast   1629.83350
1336   21  female  25.800         0     no  southwest   2007.94500
1337   61  female  29.070         0    yes  northwest  29141.36030

[1338 rows x 7 columns]>
```

In [7]:
```
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 [8]:
```
df.isnull().any()
```

Out[8]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [9]:
```
df.isna().sum()
```

Out[9]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [10]:
```
df['region'].value_counts()
```

Out[10]:
```
southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

In [11]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

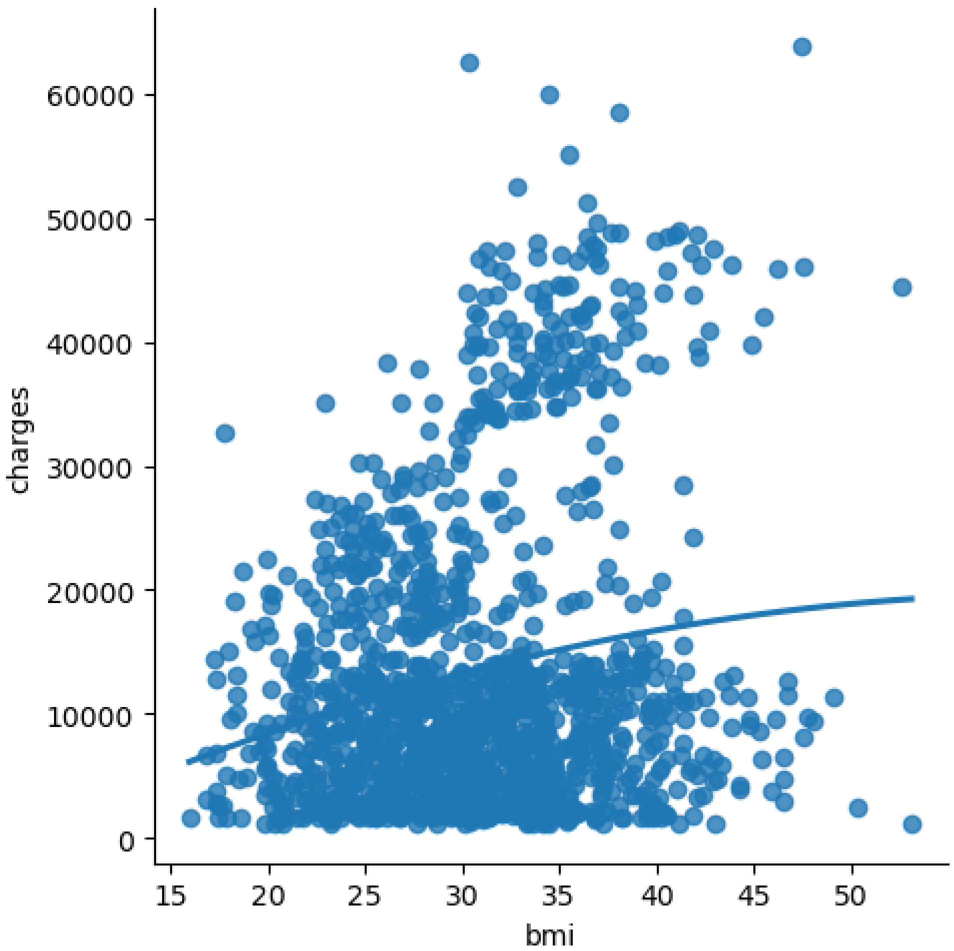
Out[11]:

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

1338 rows × 7 columns

3)Data Visualization

In [12]:
```
sns.lmplot(x='bmi',y='charges',order=2,data=df,ci=None)
plt.show()
```

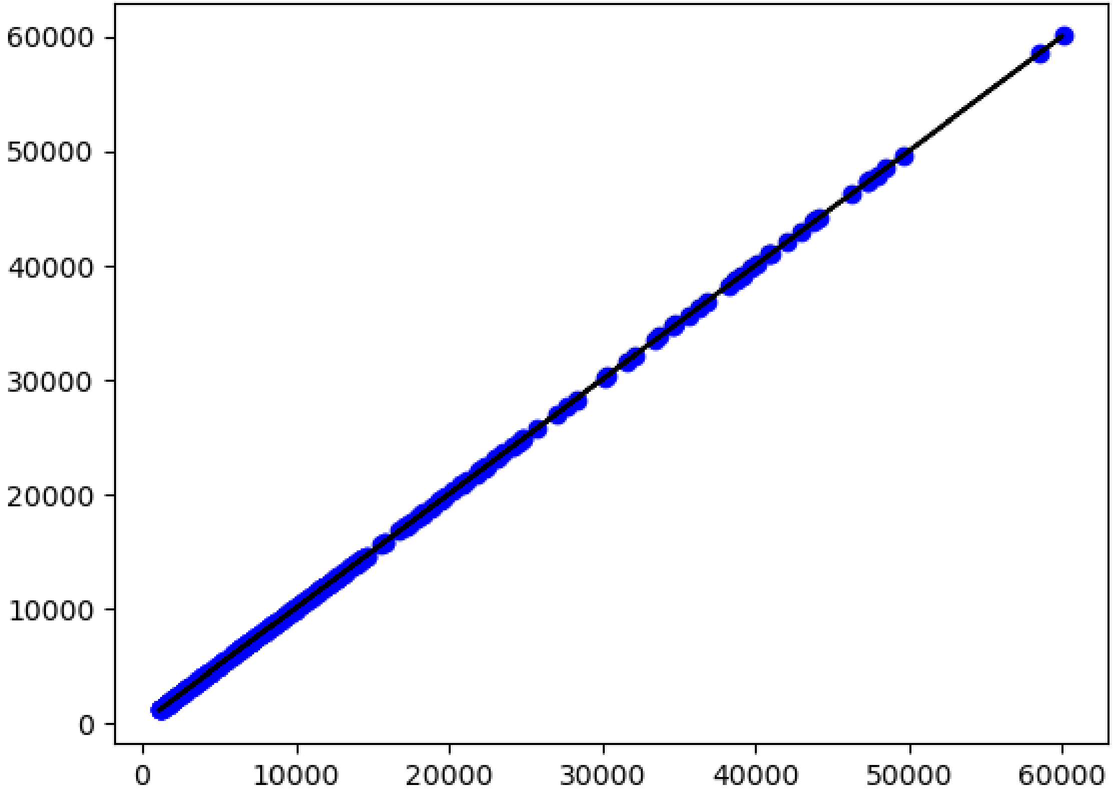


In [13]:
```
x=np.array(df['bmi']).reshape(-1,1)
y=x=np.array(df['charges']).reshape(-1,1)
```

In [14]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=1)
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

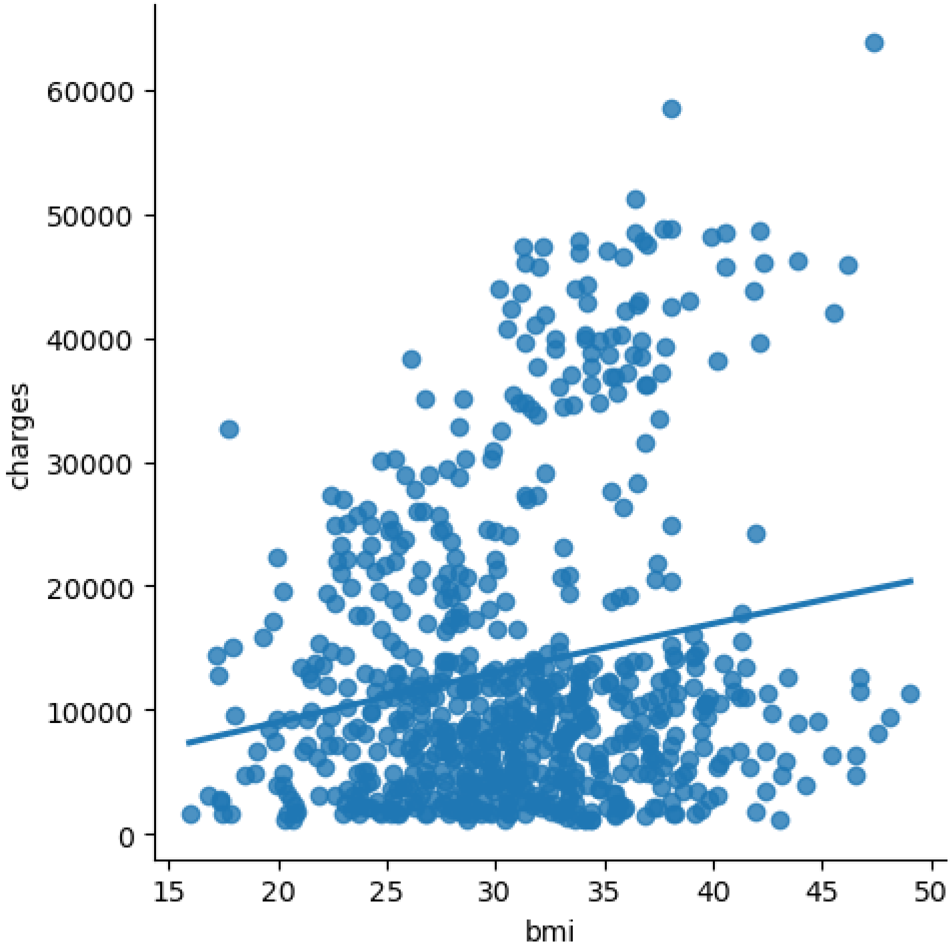
```
1.0
```

In [15]:
```
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 [16]:
```
df700=df[:][:700]
sns.lmplot(x='bmi',y='charges',order=2,ci=None,data=df700)
plt.show()
```



In [17]:
```
df700.fillna(method='ffill',inplace=True)
```

In [18]:
```
x=np.array(df700["bmi"]).reshape(-1,1)
y=np.array(df700['charges']).reshape(-1,1)
```

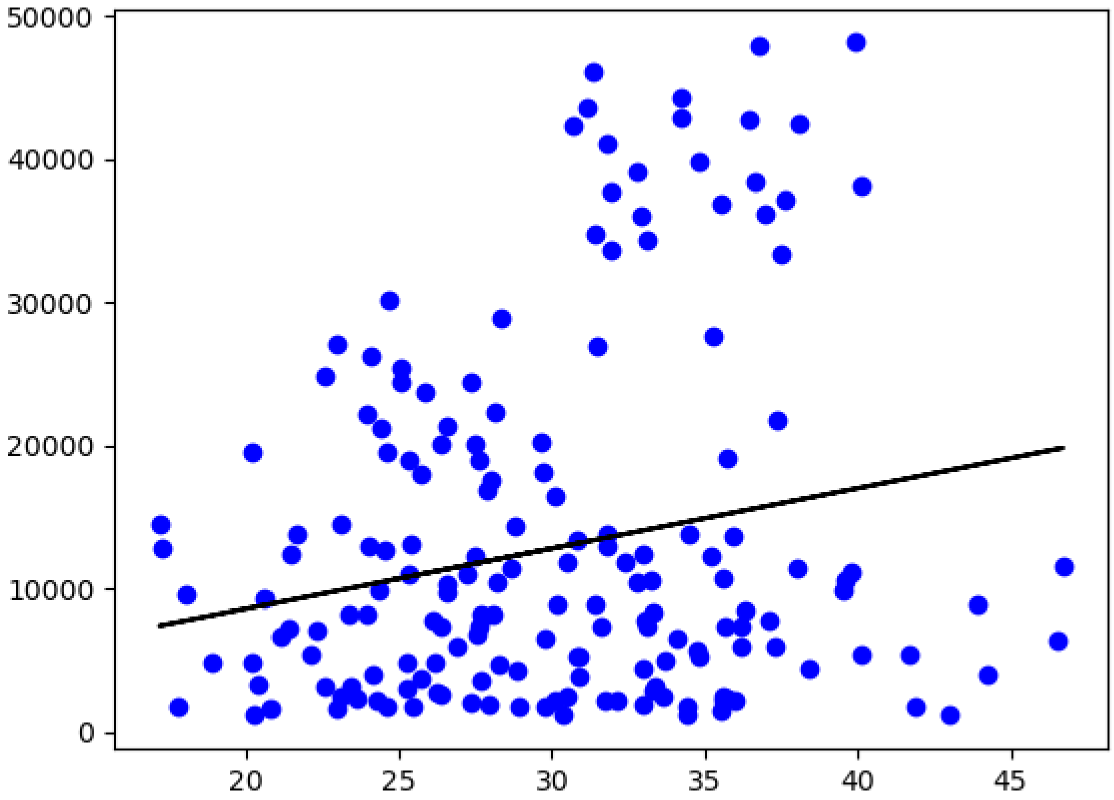
In [19]:
```
df700.dropna(inplace=True)
```

4)Building the Model

In [20]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

```
0.022173472781516046
```

In [21]:
```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



4)Evaluation of model (or) Predicting the Output

In [22]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [23]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
r2=r2_score(y_test,y_pred)
print(r2)
```

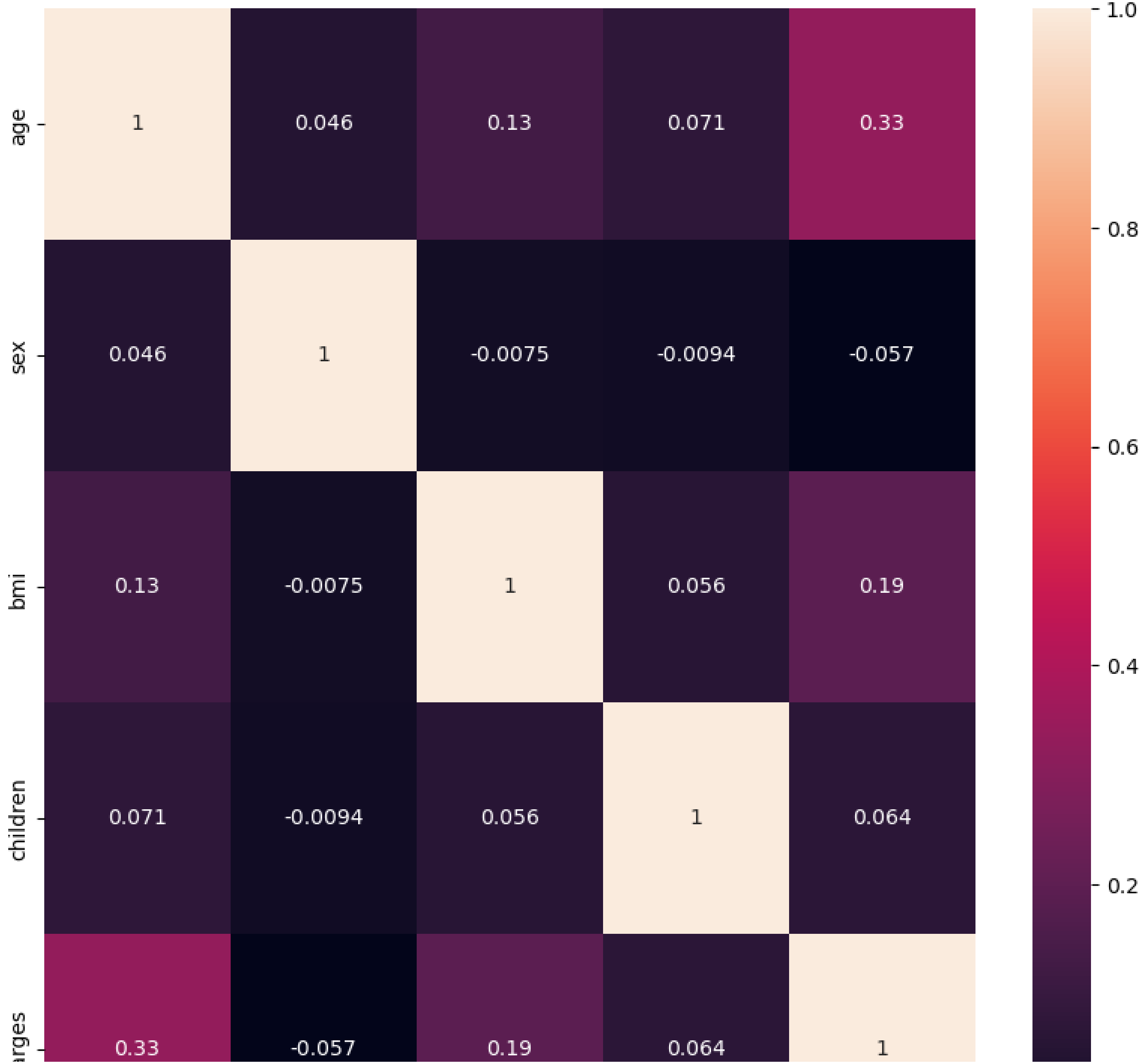
```
0.022173472781516046
```

The accuracy of the Linear Regression is 0.0221

Ridge Regression

In [24]:
```
#Importing Libraries
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [25]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df700.corr(),annot=True)
plt.show()
```


age
sex
bmi
children
arges
1
0.046
0.13
0.071
0.33
0.046
1
-0.0075
-0.0094
-0.057
0.13
-0.0075
1
0.056
0.19
0.071
-0.0094
0.056
1
0.064
0.33
-0.057
0.19
0.064
1
1.0
0.8
0.6
0.4
0.2

In [26]:
```
features=df.columns[0:1]
target=df.columns[-1]
```

In [27]:
```
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=1)
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 [28]:
```
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 0.09109639738057144
The test score for lr model is 0.08490473916580776
```

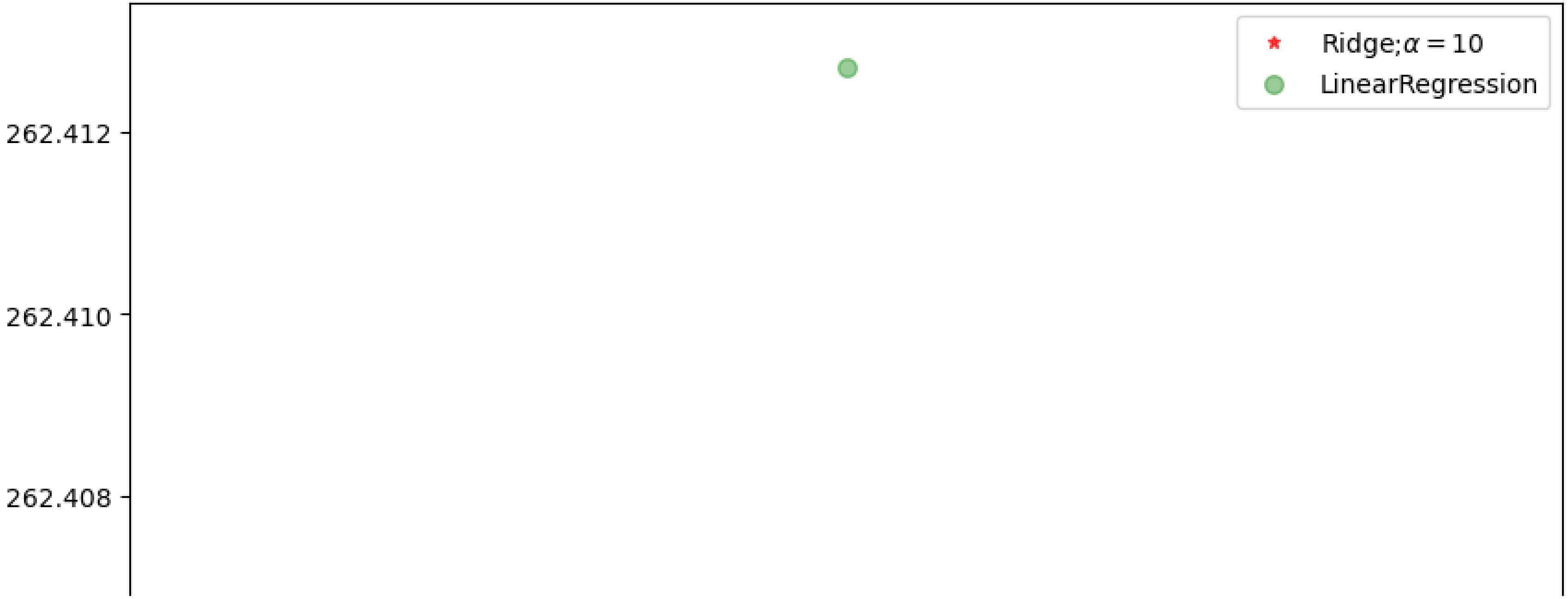
In [29]:
```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test scorefor ridge regression
train_score_ridge = ridgeReg.score(x_train, y_train)
test_score_ridge = ridgeReg.score(x_test, y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```
Ridge Model:

The train score for ridge model is 0.09109639711159623
The test score for ridge model is 0.08490538609860176
```

In [30]:
```
plt.figure(figsize=(10,10))
```

Out[30]:
```
<Figure size 1000x1000 with 0 Axes>
```

In [31]:
```
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color='red',label=r'Ridge;$\alpha=
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



The accuracy of the Ridge Model is 0.091096

Lasso Regression

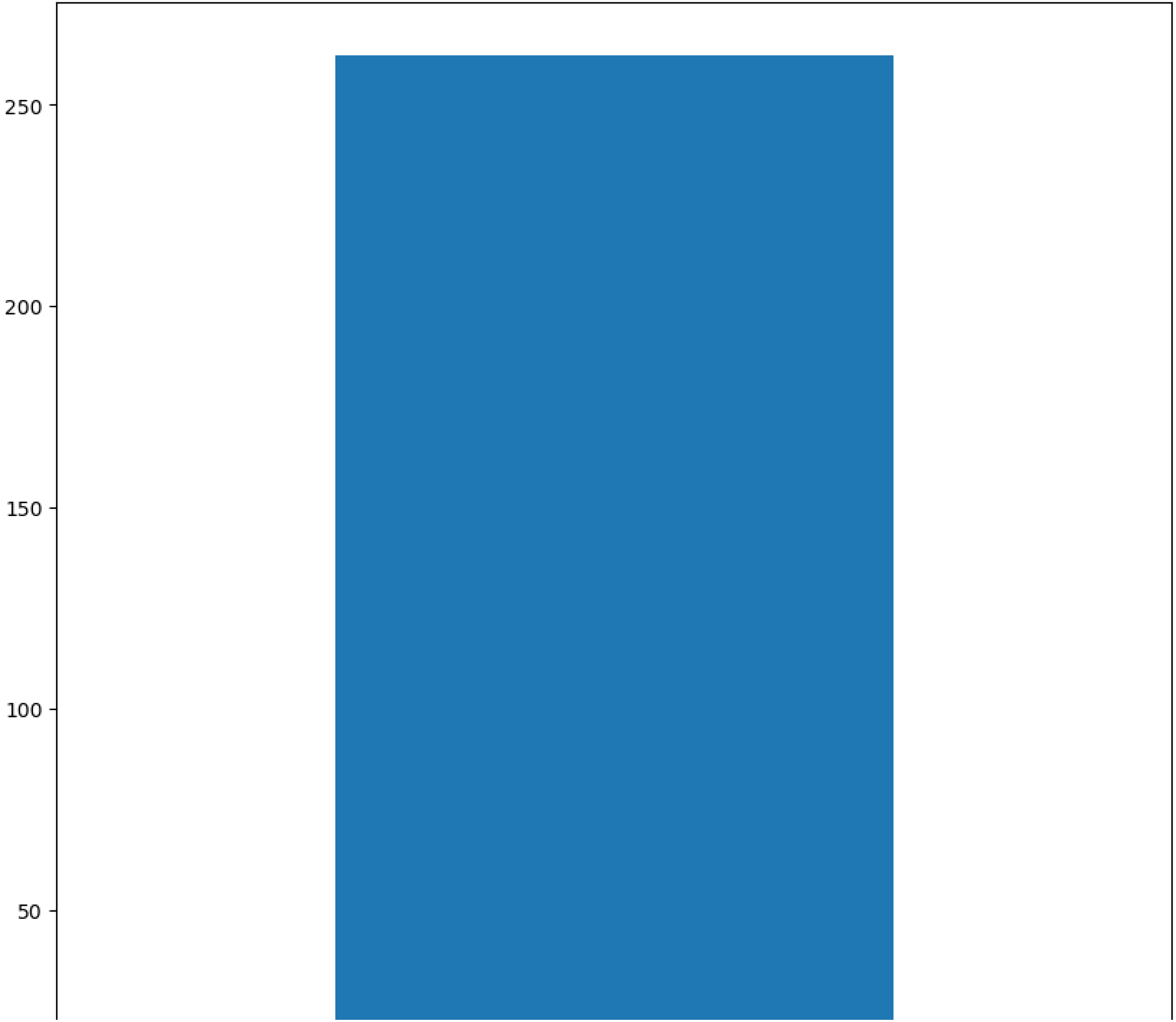
In [32]:
```
#Importing libraries
lasso= Lasso(alpha=10)
lasso.fit(x_train,y_train)
#train and test scorefor ridge regression
train_score_ls = lasso.score(x_train, y_train)
test_score_ls= lasso.score(x_test, y_test)
print("\nLasso Model:\n")
print("The train score for lasso model is {}".format(train_score_ls))
print("The test score for lasso model is {}".format(test_score_ls))
```

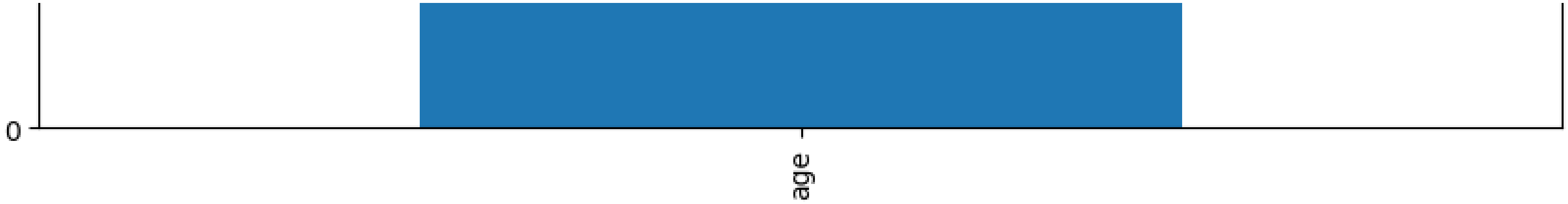
```
Lasso Model:

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

In [33]:
```
plt.figure(figsize=(10,10))
```

Out[33]:
```
<Figure size 1000x1000 with 0 Axes>
```

In [34]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
plt.show()
```


250
200
150
100
50



In [35]:
```
from sklearn.linear_model import LassoCV
```

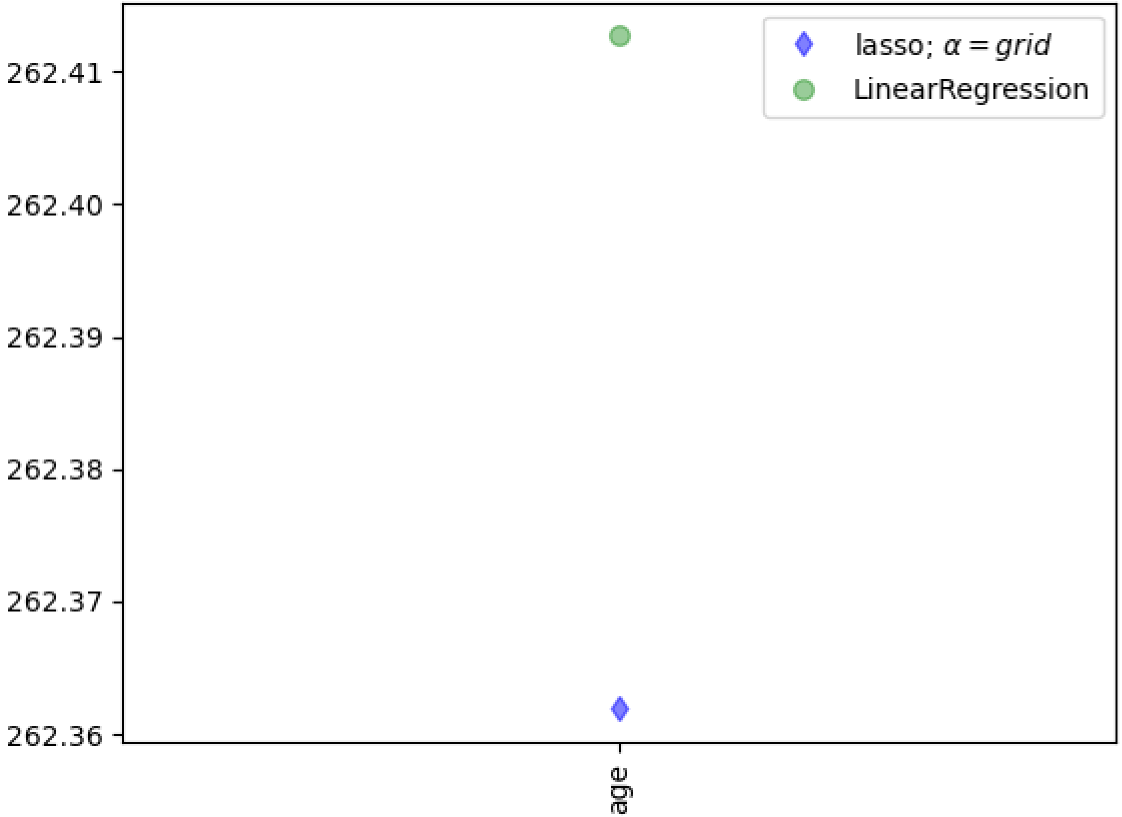
In [36]:
```
#using the linear cv model
from sklearn.linear_model import RidgeCV
#cross validation
ridge_cv=RidgeCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(ridge_cv.score(x_train,y_train))
print(ridge_cv.score(x_test,y_test))
```

```
0.09109639711159634
0.08490538609865872
```

In [37]:
```
#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))
```

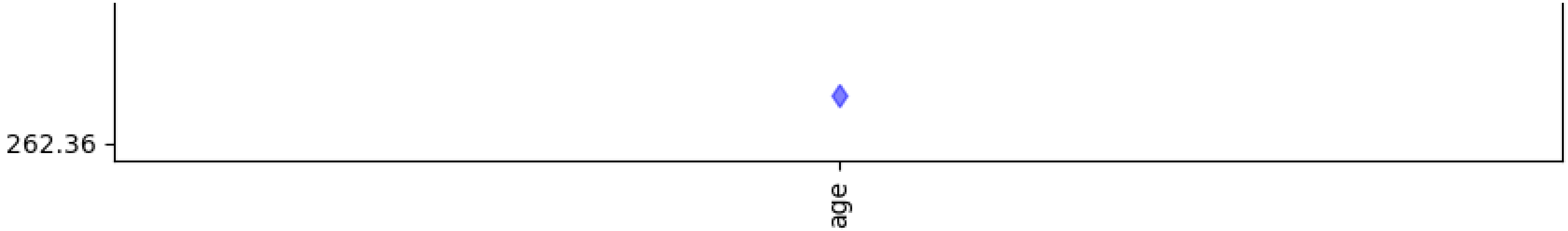
```
0.09109639395809044
0.08490704421828055
```

In [38]:
```
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha = grid$
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [39]:
```
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge; $\alpha
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha = grid$
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```


Comparison plot of Ridge, Lasso and Linear regression model
Ridge; $\alpha = 10$
lasso; $\alpha = grid$
Linear Regression
262.41
262.40
262.39
262.38
262.37



The accuracy of the Lasso Model is 0.091096

ElasticNet Regression

In [40]:
```
from sklearn.linear_model import ElasticNet
```

In [41]:
```
el=ElasticNet()
el.fit(x_train,y_train)
print(el.coef_)
print(el.intercept_)
el.score(x,y)
```

```
[261.74450967]
3115.0831774262424
```

Out[41]:
```
0.08930616764094623
```

In [42]:
```
y_pred_elastic=el.predict(x_train)
```

In [43]:
```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
135077142.70714515
```

The accuracy of the ElasticNet is 0.08930

Logistic Regression

In [44]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [45]:
```
df=pd.read_csv(r"C:\Users\Jayadeep\Downloads\insurance.csv")
df
```

Out[45]:

	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 [46]:
```
df.shape
```

Out[46]:
```
(1338, 7)
```

In [47]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [48]:
```
print('This Dataset has %d rows and %d columns'%(df.shape))
```

```
This Dataset has 1338 rows and 7 columns
```

In [49]:
```
df.head()
```

Out[49]:

	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 [50]:
```
df.describe
```

```
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
51      21  female  33.630         2     no  northwest   3579.828700
52      48    male  28.000         1    yes  southwest  23568.272000
53      36    male  34.430         0    yes  southeast  37742.575700
54      40  female  28.690         3     no  northwest   8059.679100
55      58    male  36.955         2    yes  northwest  47496.494450
56      58  female  31.825         2     no  northeast  13607.368750
57      18    male  31.680         2    yes  southeast  34303.167200
58      53  female  22.880         1    yes  southeast  23244.790200
59      34  female  37.335         2     no  northwest   5989.523650
60      43    male  27.360         3     no  northeast   8606.217400
61      25    male  33.660         4     no  southeast   4504.662400
62      64    male  24.700         1     no  northwest  30166.618170
63      28  female  25.935         1     no  northwest   4133.641650
```

In [51]:
```
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 [52]:
```
df.isnull().sum()
```

Out[52]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [53]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[53]:

	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 [54]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

Out[54]:

	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 [55]:
```
convert={"region":{"southeast":1,"southwest":2,"northeast":3,"northwest":4}}
df=df.replace(convert)
df
```

Out[55]:

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

In [56]:
```
features_matrix=df.iloc[:,0:4]
```

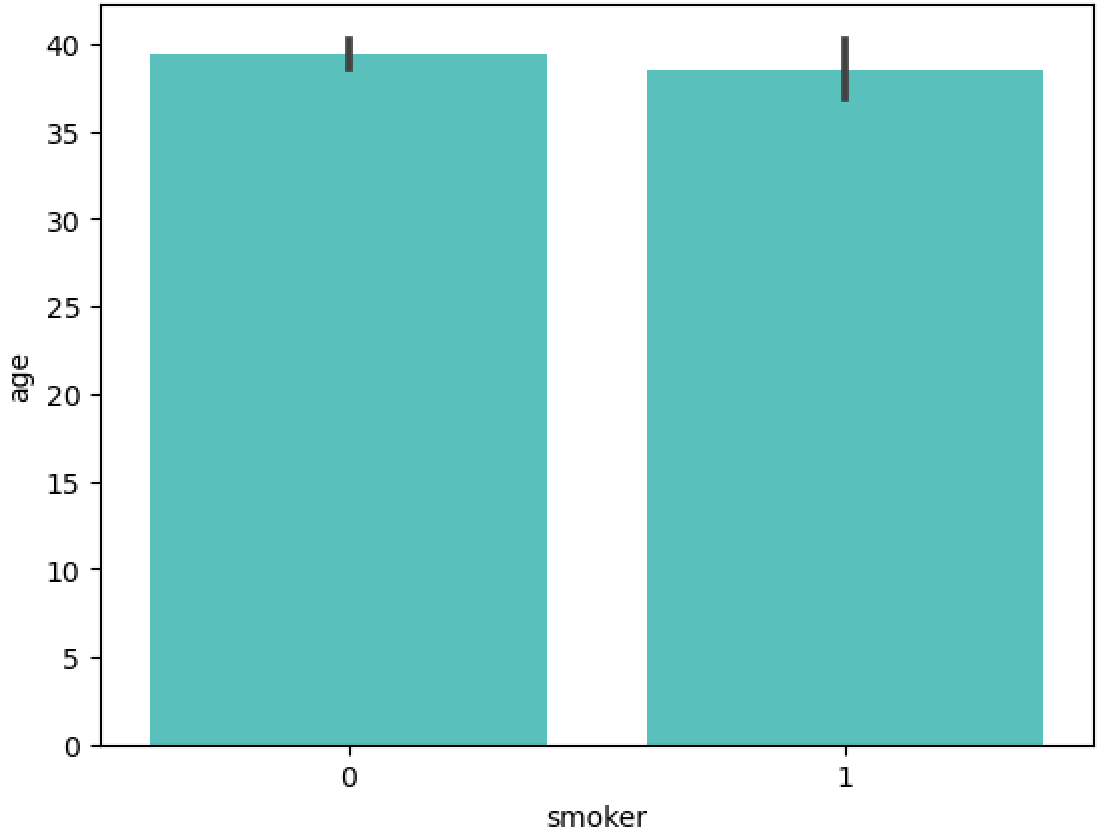
In [57]:
```
target_vector=df.iloc[:,-3]
```

In [58]:
```
print('The Feature Matrix has %d Rows and %d columns(s)'%(features_matrix.shape))
print('The Target Matrix has %d Rows and %d columns(s)'%(np.array(target_vector).reshape(-1,1).shape))
```

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

In [59]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [60]:
```
sns.barplot(x='smoker', y='age', data=df, color="mediumturquoise")
plt.show()
```



In [61]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [62]:
```
algorithm=LogisticRegression(max_iter=10000)
```

In [63]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [64]:
```
observation=[[1,0,0.99539,-0.0588]]
```

In [65]:
```
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 [66]:
```
print('The  algoritham was trained to predict one of the two classes:%s'%(algorithm.classes_))
```

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

In [67]:
```
print(" " "The Model says the probability of the observation we passed belonging to class[0] %s" " "%(algorithm.predic
print()
```

```
 The Model says the probability of the observation we passed belonging to class[0] 0.8057075871331396
```

In [68]:
```
print(" " "The Model says the probability of the observation we passed belonging to class['1'] Is %s" " "%(algorithm.p
```

```
 The Model says the probability of the observation we passed belonging to class['1'] Is 0.19429241286686041
```

In [69]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
```

In [70]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.05)
lo=LogisticRegression()
lo.fit(x_train,y_train)
print(lo.score(x_test,y_test))
```

```
0.746268656716418

C:\Users\Jayadeep\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vecto
r 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)
```

The accuracy of the Logistic Regression is 0.7462

Decision Tree

In [71]:
```
#Importing Libraries
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 [72]:
```
df=pd.read_csv(r"C:\Users\Jayadeep\Downloads\insurance.csv")
df
```

14	27	male	42.130	0	yes	southeast	39611.757700
15	19	male	24.600	1	no	southwest	1837.237000
16	52	female	30.780	1	no	northeast	10797.336200
17	23	male	23.845	0	no	northeast	2395.171550
18	56	male	40.300	0	no	southwest	10602.385000
19	30	male	35.300	0	yes	southwest	36837.467000
20	60	female	36.005	0	no	northeast	13228.846950
21	30	female	32.400	1	no	southwest	4149.736000
22	18	male	34.100	0	no	southeast	1137.011000
23	34	female	31.920	1	yes	northeast	37701.876800
24	37	male	28.025	2	no	northwest	6203.901750
25	59	female	27.720	3	no	southeast	14001.133800
26	63	female	23.085	0	no	northeast	14451.835150

In [73]:
```
df.shape
```

Out[73]:
```
(1338, 7)
```

In [74]:
```
df.isnull().any()
```

Out[74]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [75]:
```
df['region'].value_counts()
```

Out[75]:
```
southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

In [76]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

Out[76]:

	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 [77]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[77]:

	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 [78]:
```
x=["bmi","children"]
y=["Yes","No"]
all_inputs=df[x]
all_classes=df["sex"]
```

In [79]:
```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.03)
```

In [80]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [81]:
```
clf.fit(x_train,y_train)
```

Out[81]:
```
DecisionTreeClassifier(random_state=0)
```

In [82]:
```
score=clf.score(x_test,y_test)
print(score)
```

```
0.3902439024390244
```

The accuracy of the Decision Tree is 0.39024

Random Forest

In [83]:
```
import pandas as pd
import numpy as  np
import matplotlib.pyplot as plt ,seaborn as sns
```

In [84]:
```
df=pd.read_csv(r"C:\Users\Jayadeep\Downloads\insurance.csv")
df
```

9	60	female	25.840	0	no	northwest	28923.136920
10	25	male	26.220	0	no	northeast	2721.320800
11	62	female	26.290	0	yes	southeast	27808.725100
12	23	male	34.400	0	no	southwest	1826.843000
13	56	female	39.820	0	no	southeast	11090.717800
14	27	male	42.130	0	yes	southeast	39611.757700
15	19	male	24.600	1	no	southwest	1837.237000
16	52	female	30.780	1	no	northeast	10797.336200
17	23	male	23.845	0	no	northeast	2395.171550
18	56	male	40.300	0	no	southwest	10602.385000
19	30	male	35.300	0	yes	southwest	36837.467000
20	60	female	36.005	0	no	northeast	13228.846950
21	30	female	32.400	1	no	southwest	4149.736000

In [85]:
```
df.shape
```

Out[85]:
```
(1338, 7)
```

In [86]:
```
df['region'].value_counts()
```

Out[86]:
```
southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

In [87]:
```
df['bmi'].value_counts()
```

Out[87]:
```
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
24.320     7
```

In [88]:

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

```
    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
11   62    1  26.290         0    yes  southeast  27808.725100
12   23    0  34.400         0     no  southwest   1826.843000
13   56    1  39.820         0     no  southeast  11090.717800
14   27    0  42.130         0    yes  southeast  39611.757700
15   19    0  24.600         1     no  southwest   1837.237000
16   52    1  30.780         1     no  northeast  10797.336200
17   23    0  23.845         0     no  northeast   2395.171550
```

In [89]:
```
n={"smoker":{"yes":1,"no":0}}
df=df.replace(n)
print(df)
```

```
49    36   0  35.200       1      1  southeast  38709.176000
50    18   1  35.625       0      0  northeast   2211.130750
51    21   1  33.630       2      0  northwest   3579.828700
52    48   0  28.000       1      1  southwest  23568.272000
53    36   0  34.430       0      1  southeast  37742.575700
54    40   1  28.690       3      0  northwest   8059.679100
55    58   0  36.955       2      1  northwest  47496.494450
56    58   1  31.825       2      0  northeast  13607.368750
57    18   0  31.680       2      1  southeast  34303.167200
58    53   1  22.880       1      1  southeast  23244.790200
59    34   1  37.335       2      0  northwest   5989.523650
60    43   0  27.360       3      0  northeast   8606.217400
61    25   0  33.660       4      0  southeast   4504.662400
62    64   0  24.700       1      0  northwest  30166.618170
63    28   1  25.935       1      0  northwest   4133.641650
64    20   1  22.420       0      1  northwest  14711.743800
65    19   1  28.900       0      0  southwest   1743.214000
66    61   1  39.100       2      0  southwest  14235.072000
67    40   0  26.315       1      0  northwest   6389.377850
68    40   1  36.190       0      0  southeast   5920.104100
```

In [90]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[90]:
```
RandomForestClassifier()
```

In [91]:
```
rf=RandomForestClassifier()
params={'max_depth':[2,3,5,20],
        'min_samples_leaf':[5,10,20,50,100,200],
        'n_estimators':[10,25,30,50,100,200]}
```

In [92]:
```
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[92]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [2, 3, 5, 20],
                         'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy')
```

In [93]:
```
grid_search.best_score_
```

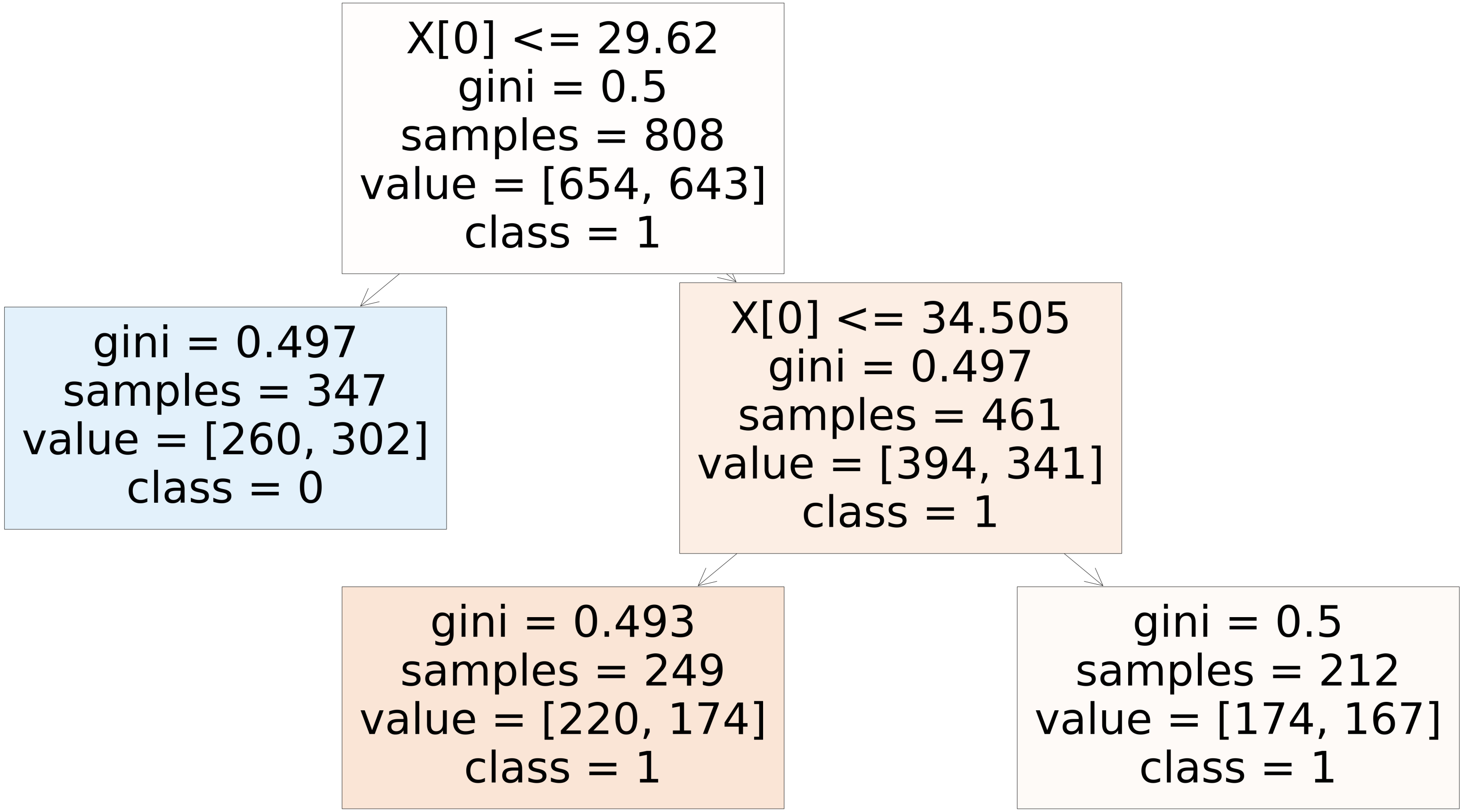
Out[93]:
```
0.5250718103825449
```

In [94]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

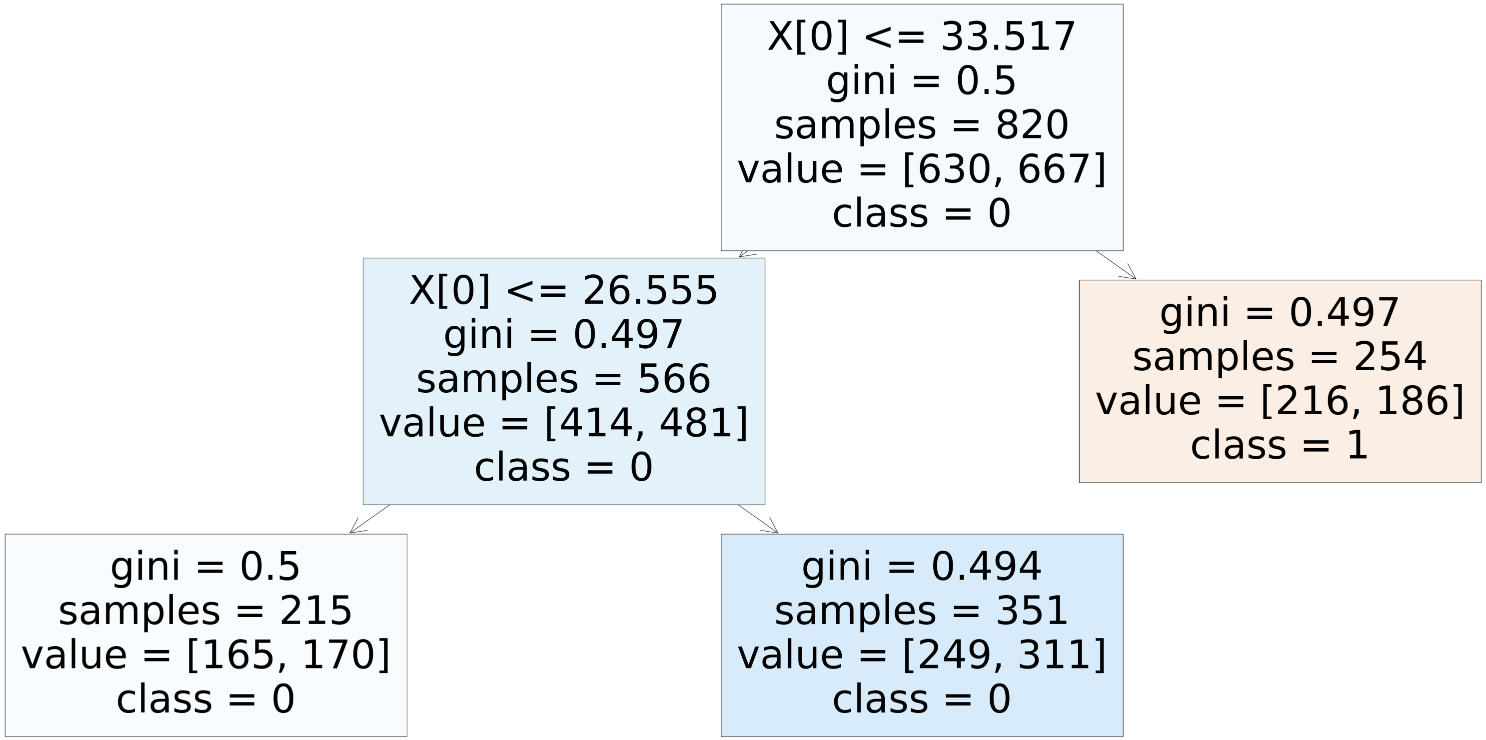
In [95]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=2, min_samples_leaf=200, n_estimators=25)
```

In [96]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['1','0'],filled=True);
plt.show()
```



In [97]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(70,30))
plot_tree(rf_best.estimators_[6],class_names=["1","0"],filled=True);
plt.show()
```



In [98]:
```
rf_best.feature_importances_
```

Out[98]:
```
array([0.78503705, 0.21496295])
```

In [99]:
```
rf=RandomForestClassifier(random_state=0)
```

In [100]:
```
rf.fit(x_train,y_train)
```

Out[100]:
```
RandomForestClassifier(random_state=0)
```

In [101]:
```
score=rf.score(x_test,y_test)
print(score)
```

```
0.4146341463414634
```

The accuracy of the Random Forest is 0.4146

CONCLUSION:

The given dataset is "Insurance",we need to find the bestfit Model. As per the data set ,we have used different types of models,that different models got different types of accuracyies.In this process Ridge and Lasso got same accuracy of 0.091 so,we should not consider that.For the ElasticNet model I got the accuracy of 0.089306.For the highest accuracy,I have done so many models among those ElasticNet got highest accuracy.I have done so many visuvalization graps as per the given Features.

Therefore ElasticNet Regression is the Bestfit for this Model.