PROBLEM STATEMENT:To check how best fitis it?

importing the libraries

In [1]:

```
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
```

Data Collection

In [2]:

```
df=pd.read_csv(r"C:\Users\samit\OneDrive\Desktop\jupyter\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

Data cleaning

In [3]:

```
df=df[['bmi','charges']]
df.columns=['bmi','char']
```

In [4]:

```
df.head(10)
```

Out[4]:

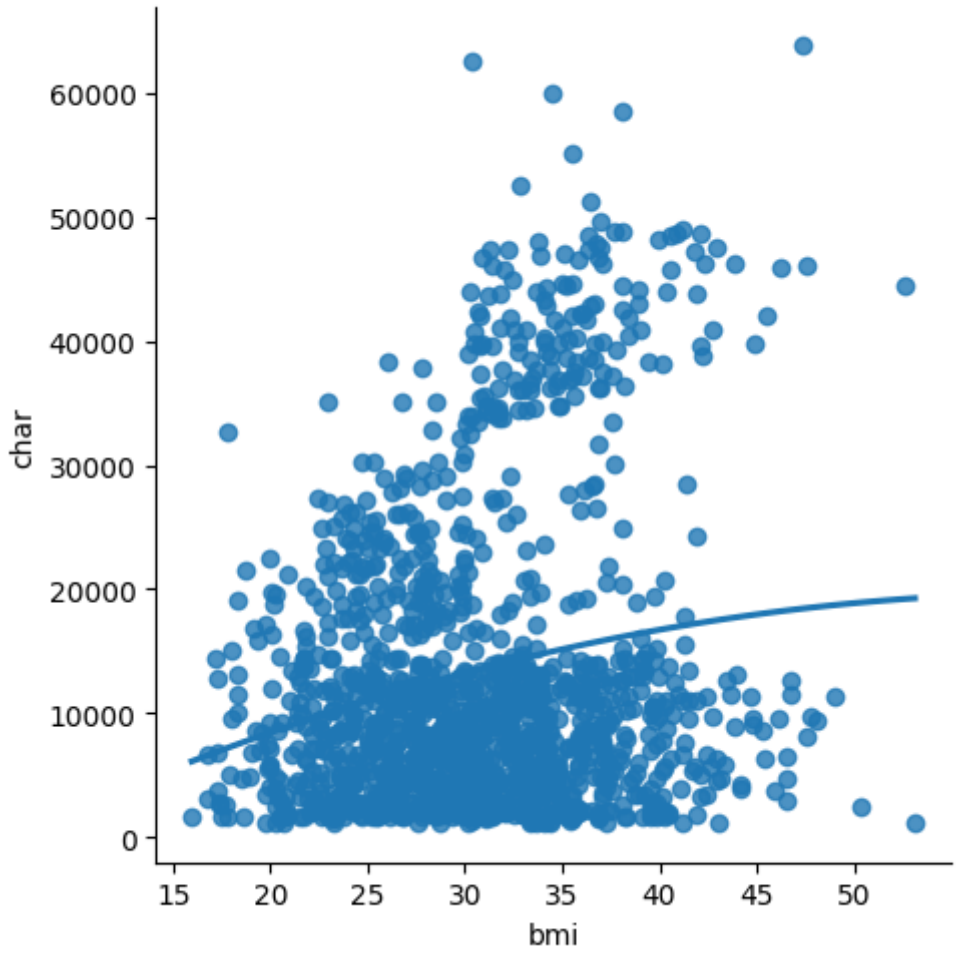
	bmi	char
0	27.900	16884.92400
1	33.770	1725.55230
2	33.000	4449.46200
3	22.705	21984.47061
4	28.880	3866.85520
5	25.740	3756.62160
6	33.440	8240.58960
7	27.740	7281.50560
8	29.830	6406.41070
9	25.840	28923.13692

In [5]:

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

Out[5]:

```
<seaborn.axisgrid.FacetGrid at 0x17caa4cd690>
```



In [6]:

```
df.describe()
```

Out[6]:

	bmi	char
count	1338.000000	1338.000000
mean	30.663397	13270.422265
std	6.098187	12110.011237
min	15.960000	1121.873900
25%	26.296250	4740.287150
50%	30.400000	9382.033000
75%	34.693750	16639.912515
max	53.130000	63770.428010

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   bmi     1338 non-null   float64
 1   char    1338 non-null   float64
dtypes: float64(2)
memory usage: 21.0 KB
```

In [8]:

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

Out[8]:

```
bmi     0
char    0
dtype: int64
```

In [9]:

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

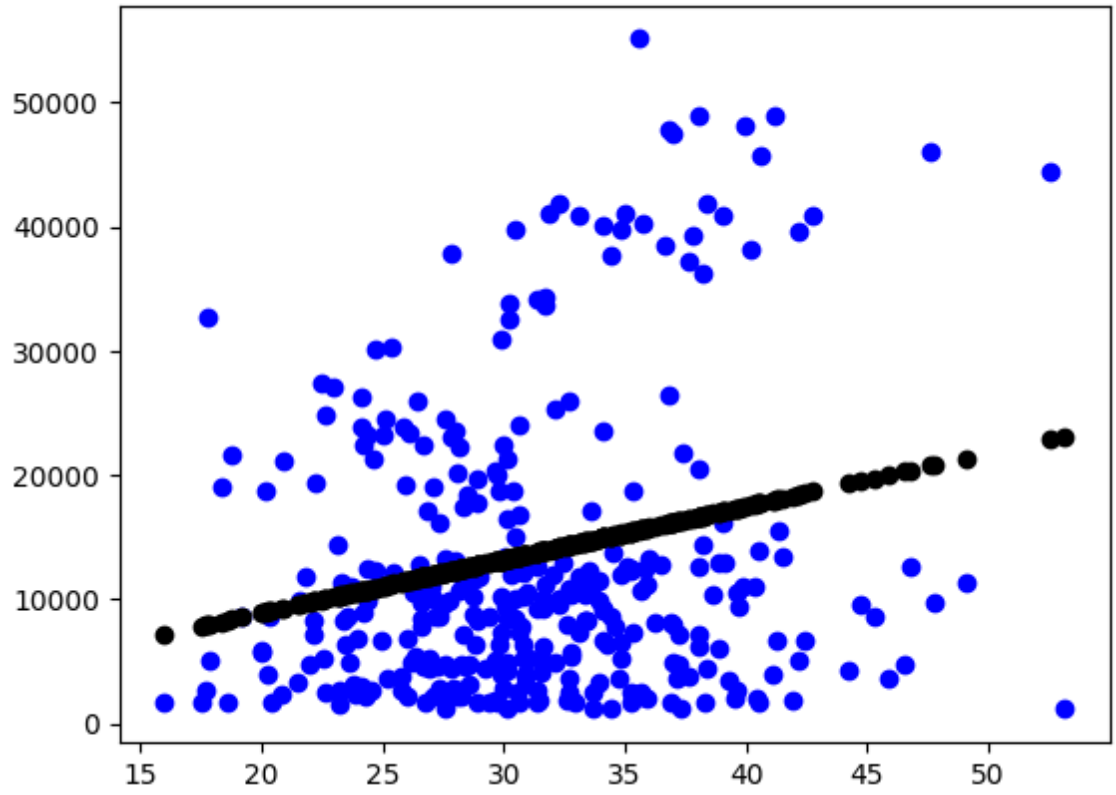
In [10]:

```
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))
```

```
0.01911273362002952
```

In [11]:

```
y_pred=regr.predict(X_test)
plt.scatter(X_test,y_test,color='b')
plt.scatter(X_test,y_pred,color='k')
plt.show()
```

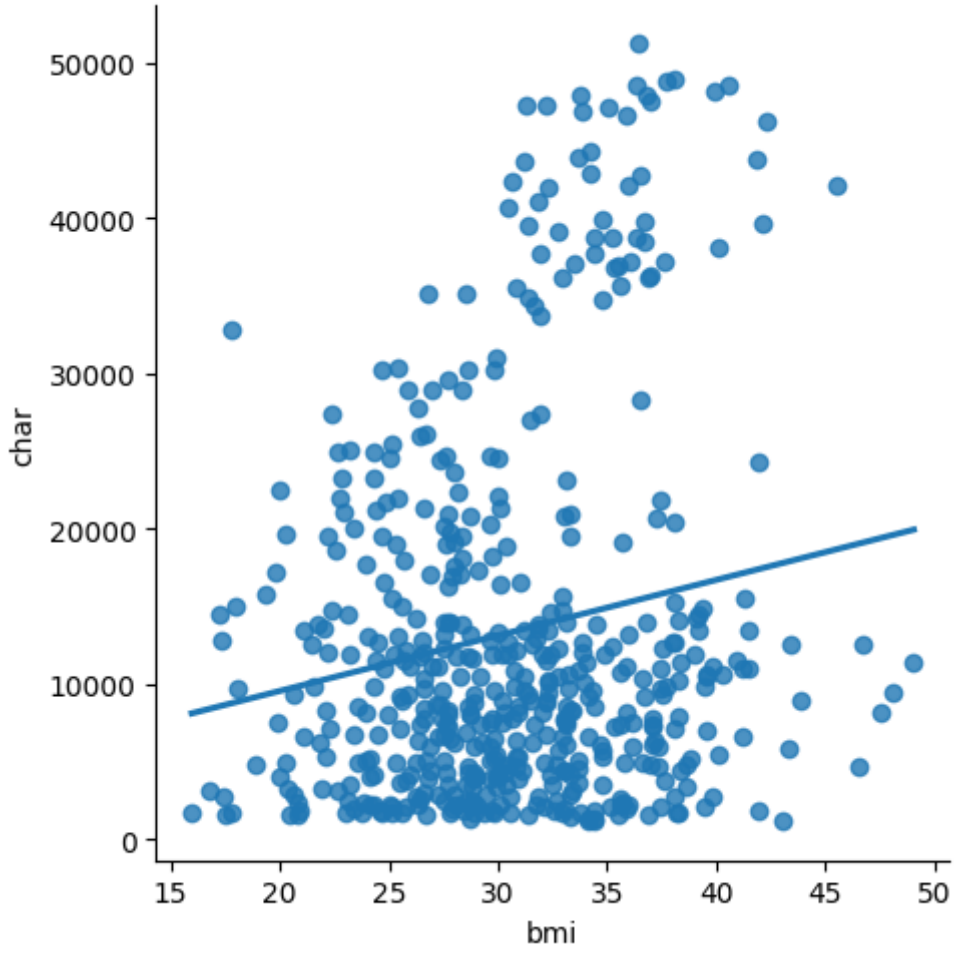


In [12]:

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

Out[12]:

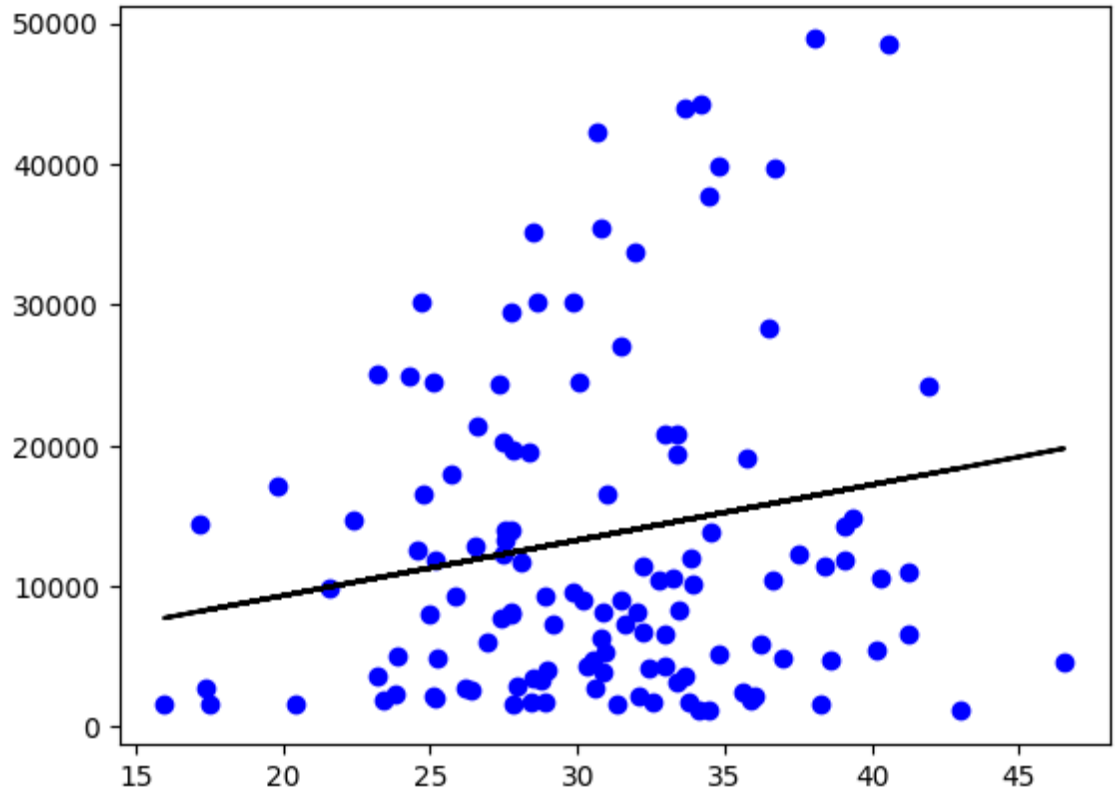
```
<seaborn.axisgrid.FacetGrid at 0x17cc8114ed0>
```



In [13]:

```
df500.fillna(method='ffill',inplace=True)
X=np.array(df500['bmi']).reshape(-1,1)
y=np.array(df500['char']).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='b')
plt.plot(X_test,y_pred,color='k')
plt.show()
```

```
Regression: 0.0039764126492880525
```



In [14]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
mode1=LinearRegression()
mode1.fit(X_train,y_train)
y_pred=mode1.predict(X_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: 0.0039764126492880525
```

Ridge,Lasso,ElasticNet

In [15]:

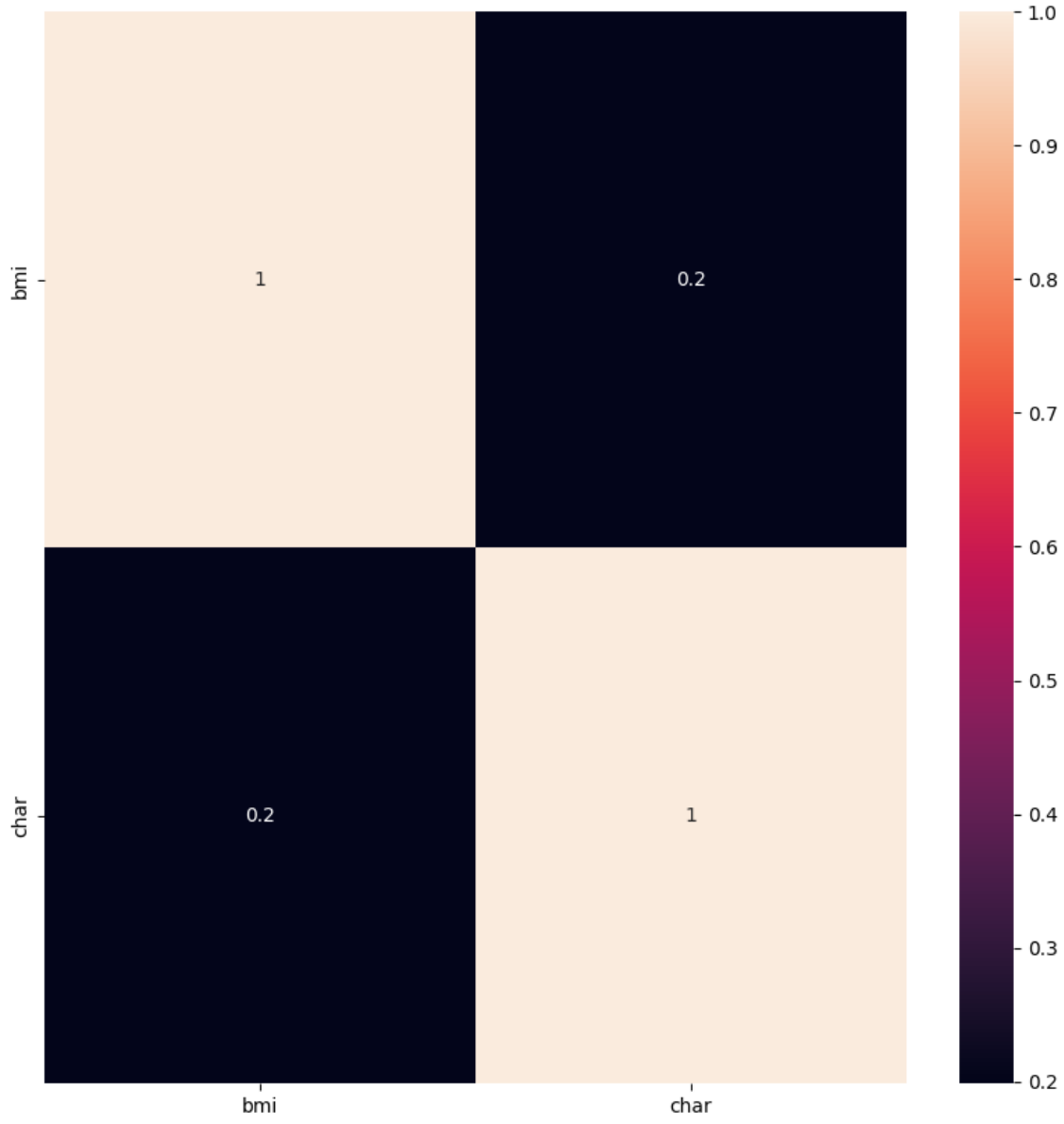
```
from sklearn.linear_model import Ridge, RidgeCV, Lasso
from sklearn.preprocessing import StandardScaler
```

In [16]:

```
plt.figure(figsize = (10,10))
sns.heatmap(df.corr(), annot = True)
```

Out[16]:

<Axes: >



In [17]:

```
features = df.columns[0:2]
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, 2)
The dimension of X_test is (402, 2)
```

In [18]:

```
#Model
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 1.0
The test score for lr model is 1.0
```

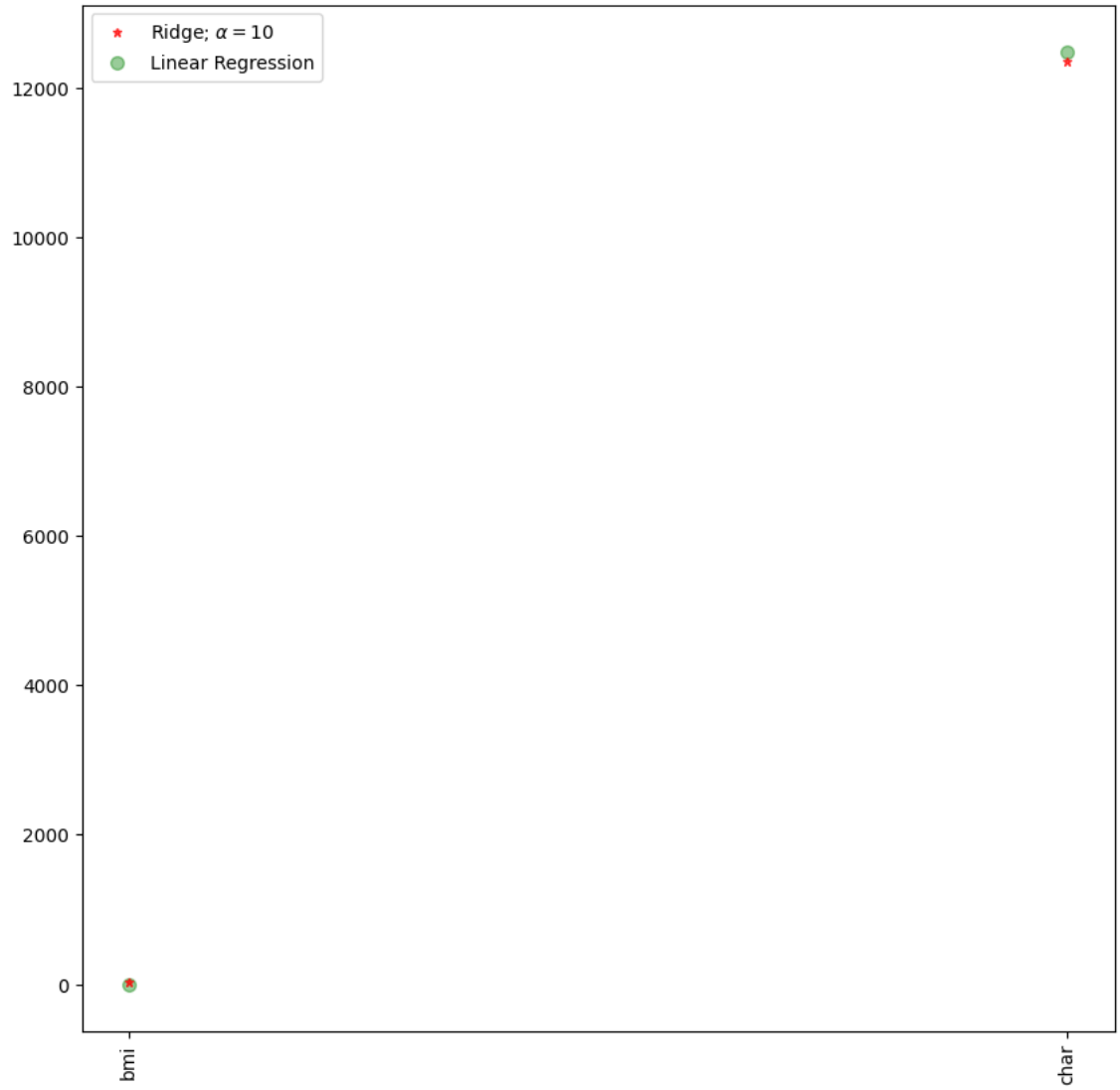
In [19]:

```
#Ridge Regression Model
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 0.9998827490191866
The test score for ridge model is 0.9998744489679483
```

In [20]:

```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,colo
#plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',la
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gre
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



In [21]:

```
#Lasso regression model
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))
```

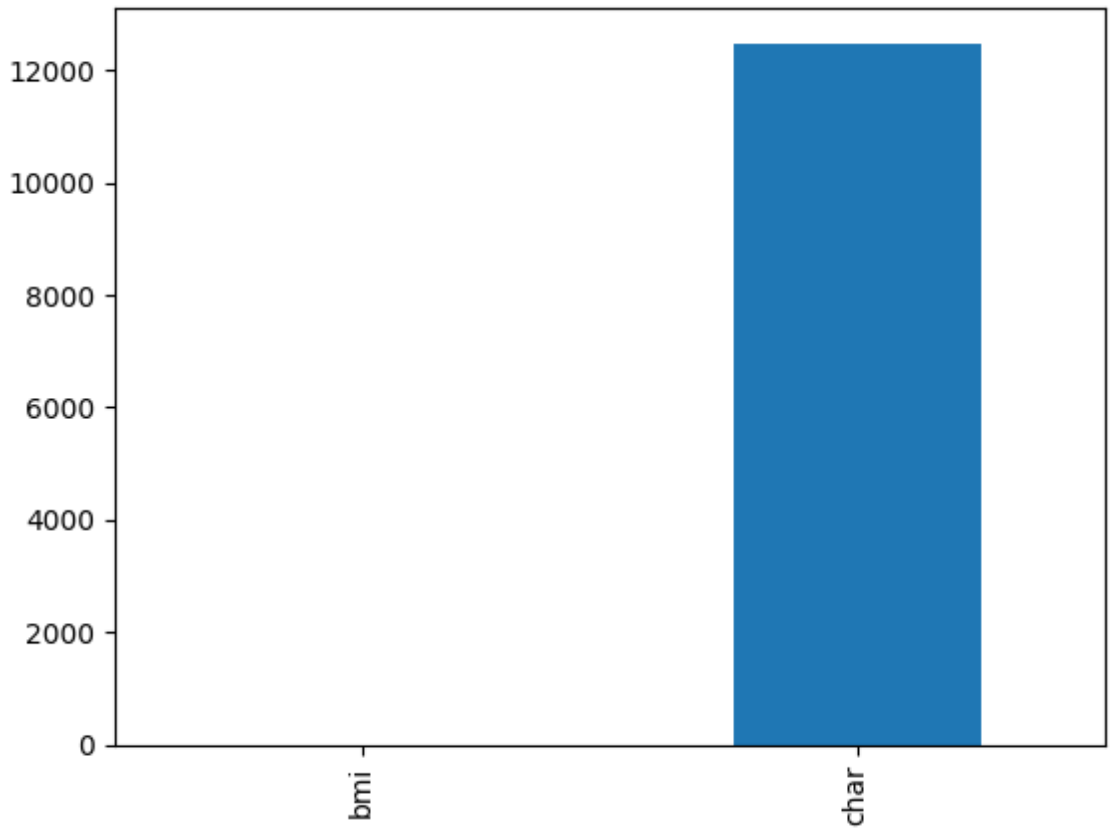
```
Lasso Model:

The train score for ls model is 0.9999993594011218
The test score for ls model is 0.9999993415784849
```

In [22]:

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

Out[22]:

```
<Axes: >
```



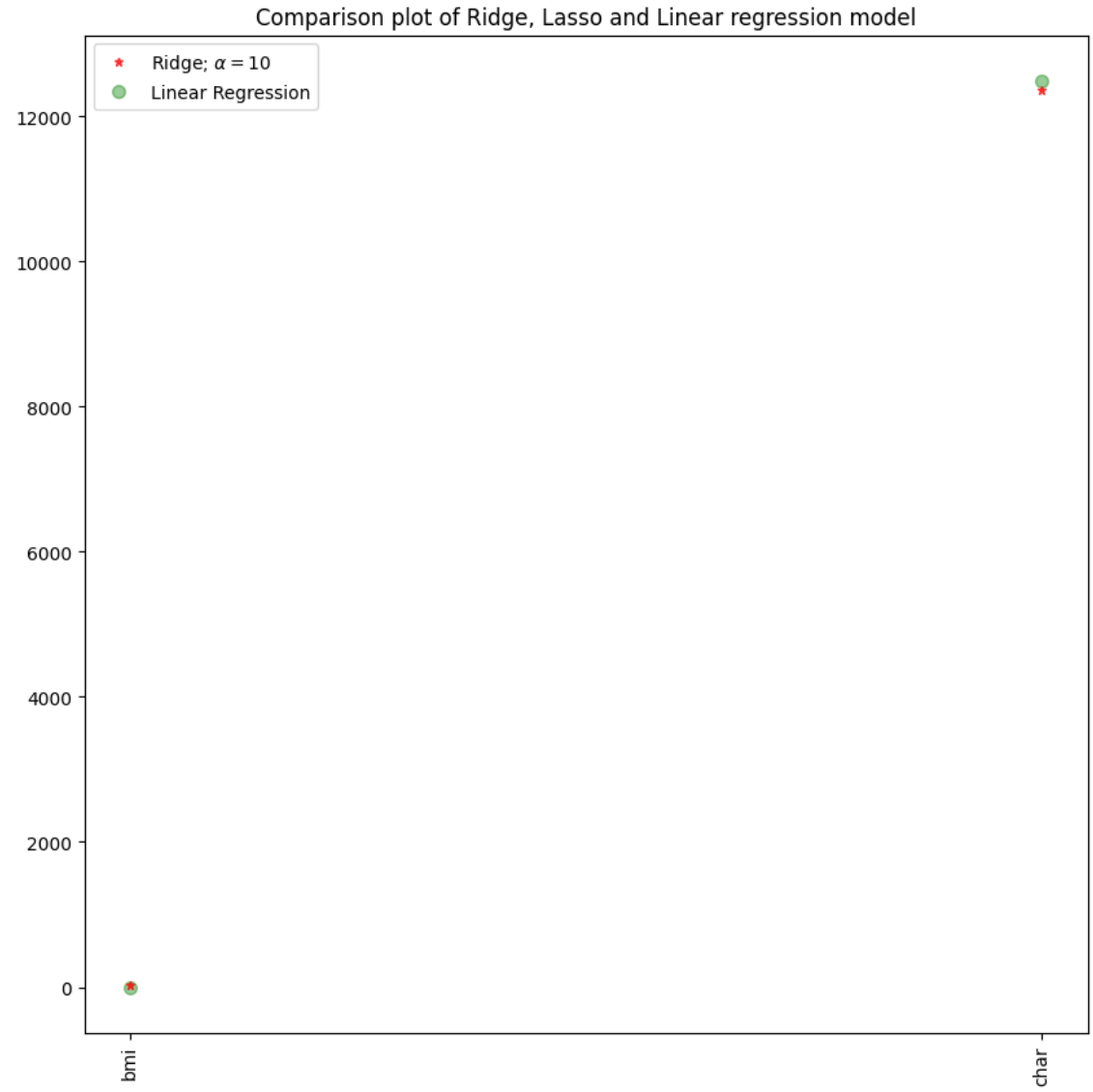
In [23]:

```
#Using the linear CV model
from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).fit(X_trai
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.9999999999984559
0.9999999999980432
```

In [24]:

```
#plot size
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,colo
#add plot for lasso regression
#plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue'
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gre
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```



In [25]:

```
#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 0.9999999999999891
The train score for ridge model is 0.9999999999999883
```

In [26]:

```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.         0.99999999]
9.055665395862889e-05
```

In [27]:

```
y_pred_elastic=regr.predict(X_train)
```

In [28]:

```
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 347174894.85874784
```

In [ ]:

```

```

Decision Tree

In [29]:

```
import numpy as ny
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [30]:

```
df=pd.read_csv(r"C:\Users\samit\OneDrive\Desktop\jupyter\insurance.csv")
df
```

Out[30]:

	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 [31]:

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

Out[31]:

```
age
18    69
19    68
50    29
51    29
47    29
46    29
45    29
20    29
48    29
52    29
22    28
49    28
54    28
53    28
21    28
26    28
24    28
25    28
28    28
27    28
23    28
43    27
29    27
30    27
41    27
42    27
44    27
31    27
40    27
32    26
33    26
56    26
34    26
55    26
57    26
37    25
59    25
58    25
36    25
38    25
35    25
39    25
61    23
60    23
63    23
62    23
64    22
Name: count, dtype: int64
```

In [32]:

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

Out[32]:

```
sex
male      676
female    662
Name: count, dtype: int64
```

In [33]:

```
convert={'sex':{"female":1,"male":2}}
df=df.replace(convert)
df
```

Out[33]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	yes	southwest	16884.92400
1	18	2	33.770	1	no	southeast	1725.55230
2	28	2	33.000	3	no	southeast	4449.46200
3	33	2	22.705	0	no	northwest	21984.47061
4	32	2	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	2	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

In [34]:

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

Out[34]:

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

1338 rows × 7 columns

In [35]:

```
x=["age","sex","bmi"]
y=["yes","no"]
all_inputs=df[x]
all_classes=df["smoker"]
```

In [36]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.25)
```

In [37]:

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

In [38]:

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

Out[38]:

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

In [39]:

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

Out[39]:

```
0.6955223880597015
```

Randomforest

In [40]:

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

Out[40]:

```
RandomForestClassifier
RandomForestClassifier()
```

In [41]:

```
rf=RandomForestClassifier()
```

In [42]:

```
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 [43]:

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[43]:

```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [44]:

```
grid_search.best_score_
```

Out[44]:

```
0.7966099673163632
```

In [45]:

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

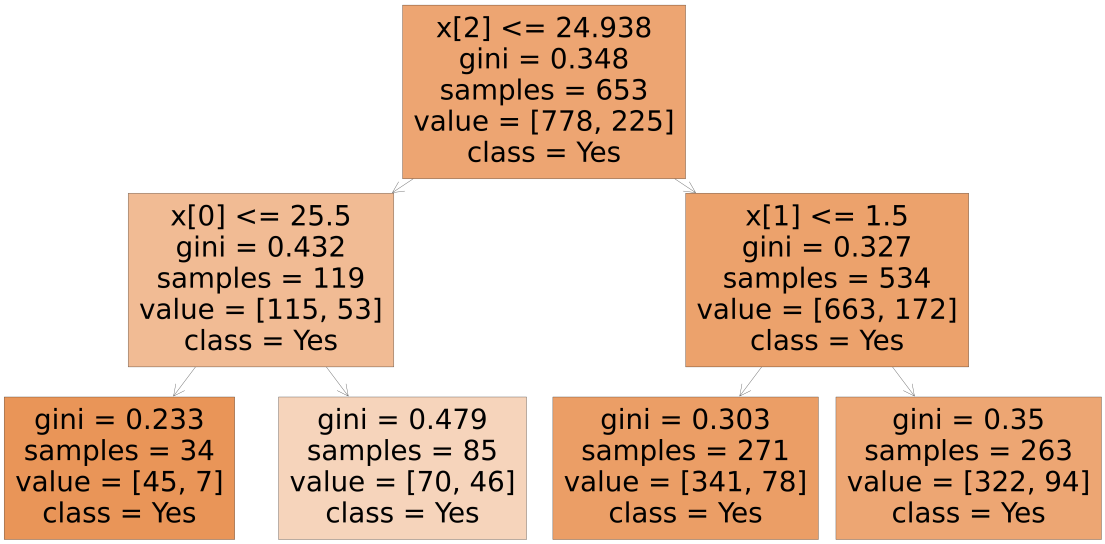
```
RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=10)
```

In [46]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],class_names=['Yes','No'],filled=True)
```

Out[46]:

```
[Text(0.5, 0.8333333333333334, 'x[2] <= 24.938\ngini = 0.348\nsamples = 6
53\nvalue = [778, 225]\nclass = Yes'),
 Text(0.25, 0.5, 'x[0] <= 25.5\ngini = 0.432\nsamples = 119\nvalue = [11
5, 53]\nclass = Yes'),
 Text(0.125, 0.16666666666666666, 'gini = 0.233\nsamples = 34\nvalue = [4
5, 7]\nclass = Yes'),
 Text(0.375, 0.16666666666666666, 'gini = 0.479\nsamples = 85\nvalue = [7
0, 46]\nclass = Yes'),
 Text(0.75, 0.5, 'x[1] <= 1.5\ngini = 0.327\nsamples = 534\nvalue = [663,
172]\nclass = Yes'),
 Text(0.625, 0.16666666666666666, 'gini = 0.303\nsamples = 271\nvalue =
[341, 78]\nclass = Yes'),
 Text(0.875, 0.16666666666666666, 'gini = 0.35\nsamples = 263\nvalue = [3
22, 94]\nclass = Yes')]
```

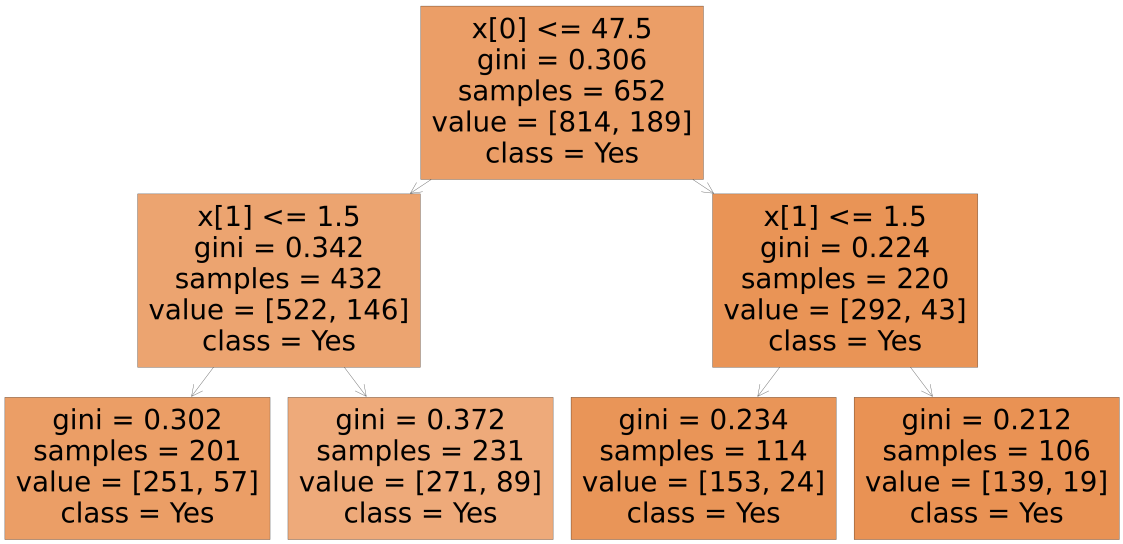


In [47]:

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

Out[47]:

```
[Text(0.5, 0.8333333333333334, 'x[0] <= 47.5\ngini = 0.306\nsamples = 652
\nvalue = [814, 189]\nclass = Yes'),
 Text(0.25, 0.5, 'x[1] <= 1.5\ngini = 0.342\nsamples = 432\nvalue = [522,
146]\nclass = Yes'),
 Text(0.125, 0.16666666666666666, 'gini = 0.302\nsamples = 201\nvalue =
[251, 57]\nclass = Yes'),
 Text(0.375, 0.16666666666666666, 'gini = 0.372\nsamples = 231\nvalue =
[271, 89]\nclass = Yes'),
 Text(0.75, 0.5, 'x[1] <= 1.5\ngini = 0.224\nsamples = 220\nvalue = [292,
43]\nclass = Yes'),
 Text(0.625, 0.16666666666666666, 'gini = 0.234\nsamples = 114\nvalue =
[153, 24]\nclass = Yes'),
 Text(0.875, 0.16666666666666666, 'gini = 0.212\nsamples = 106\nvalue =
[139, 19]\nclass = Yes')]
```



In [48]:

```
rf_best.feature_importances_
```

Out[48]:

```
array([0.46196743, 0.07635051, 0.46168206])
```

In [49]:

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

Out[49]:

	Varname	Imp
0	age	0.461967
2	bmi	0.461682
1	sex	0.076351

conclusion:

```
Based on all model accuracies we conclude that the Randomforest classification is
some what best fit compared to all other models.
```