Problem Statement:

To find the chances of a person to be a smoker or not based on his age,sex,bmi,region they belongs to

In [1]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
```

Data cleaning and preprocessing

In [2]:
```
df=pd.read_csv(r"C:\Users\my pc\downloads\insurance.csv")
df        # reading data insurance csv file  to 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

In [3]:
```
df.head()  #displaying first 5 rows
```

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()    #displaying last 5 rows
```

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.info()     #information about the data
```

```
<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 [6]:
```
df.describe()     #describes about the data
```

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.isna().any()   #checking for null values
```

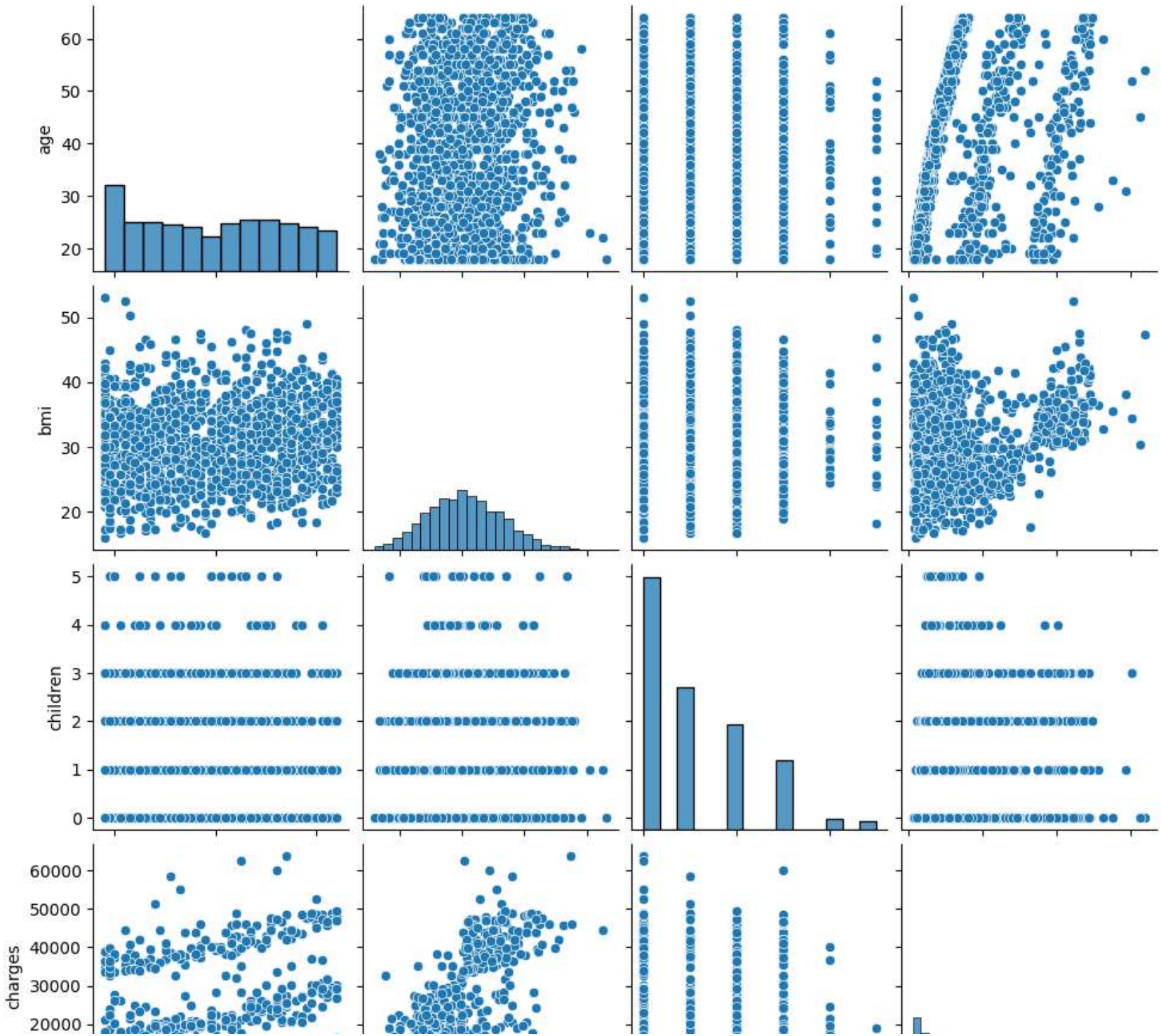
Out[7]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

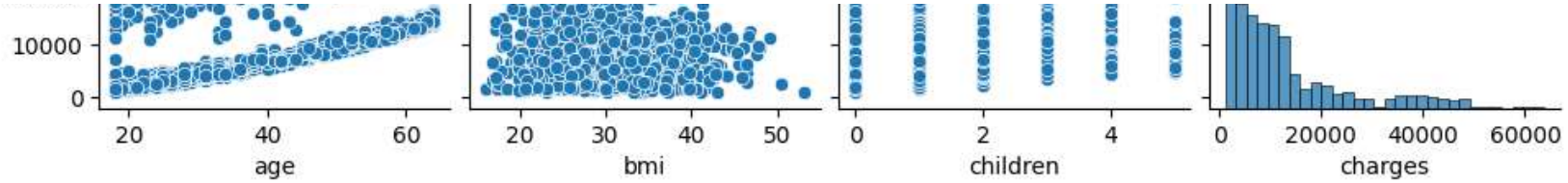
In [8]:
```
df.shape    #checking the shape of the dataset ,contains 1338 rows and 7 columns
```

Out[8]:
```
(1338, 7)
```

In [9]:
```
sns.pairplot(df)    #checking the relation for each and every column to every column
```

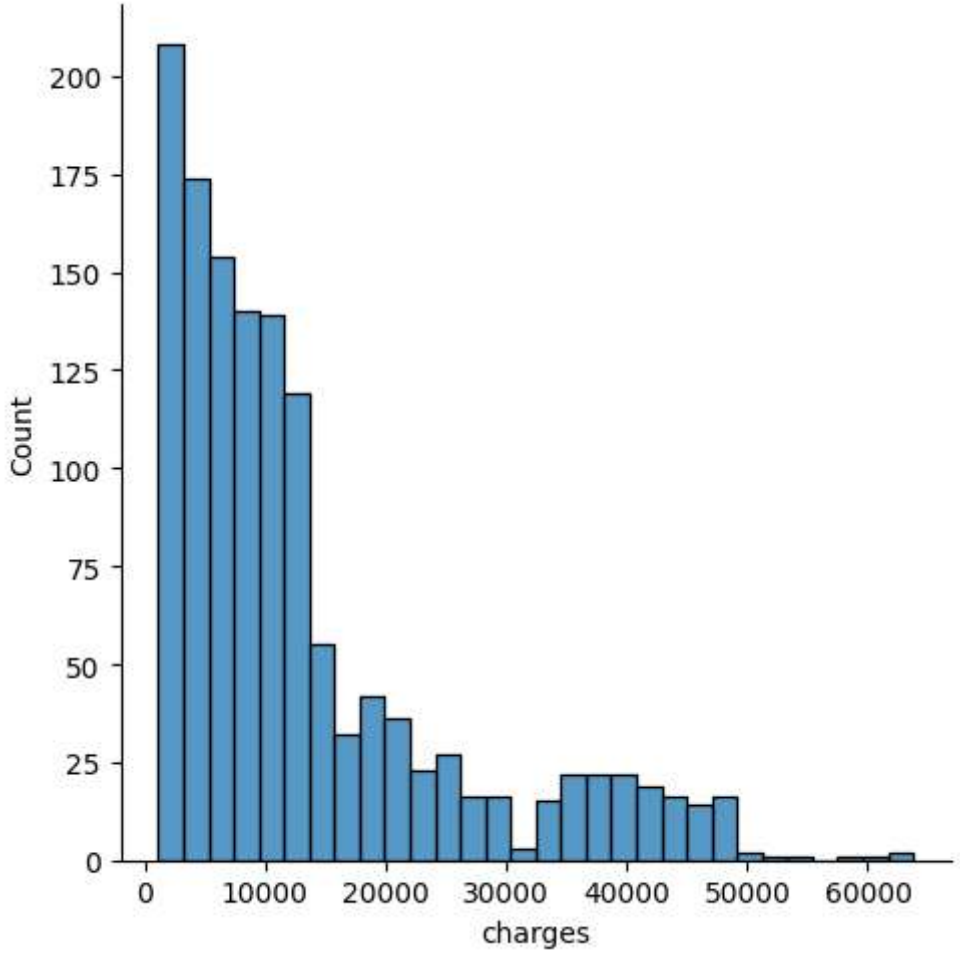
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x20e79c28b50>
```


age
bmi
children
charges
60
50
40
30
20
50
40
30
20
5
4
3
2
1
0
60000
50000
40000
30000
20000



In [10]:
```
sns.displot(df["charges"])
```

Out[10]: <seaborn.axisgrid.FacetGrid at 0x20e7d8c83a0>



In [11]:
```
df["region"].value_counts()   #counting the number of labels in region column
```

Out[11]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [12]:
```
df["sex"].value_counts()    #counting the number of labels in sex column
```

Out[12]:
```
sex
male      676
female    662
Name: count, dtype: int64
```

In [13]:
```
df["smoker"].value_counts()     #counting the number of labels in smoker column
```

Out[13]:
```
smoker
no     1064
yes     274
Name: count, dtype: int64
```

In [14]:
```
r={"smoker":{"yes":1,"no":0}}
df=df.replace(r)              #replacing the string with numerical values
df
```

Out[14]:

	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 [15]:
```
r={"sex":{"male":1,"female":0}}
df=df.replace(r)        #replacing the string with numerical values
df
```

Out[15]:

	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 [16]:
```
r={"region":{"northeast":1,"northwest":2,"southeast":3,"southwest":4}}
df=df.replace(r)          #replacing the string with numerical values
df
```

Out[16]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	4	16884.92400
1	18	1	33.770	1	0	3	1725.55230
2	28	1	33.000	3	0	3	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	1	2205.98080
1335	18	0	36.850	0	0	3	1629.83350
1336	21	0	25.800	0	0	4	2007.94500
1337	61	0	29.070	0	1	2	29141.36030

1338 rows × 7 columns

In [17]:
```
features=df.columns[0:6]
features
```

Out[17]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region'], dtype='object')
```

In [18]:
```
target=df.columns[-1]
target                  #Assigning feature/input columns
```

Out[18]:
```
'charges'
```

In [19]:
```
#Training our model
from sklearn.model_selection import train_test_split
x=df[features].values
y=df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.50,random_state=17)
x_train.shape
x_test.shape
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

In [20]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()      #calling linearregression method
```

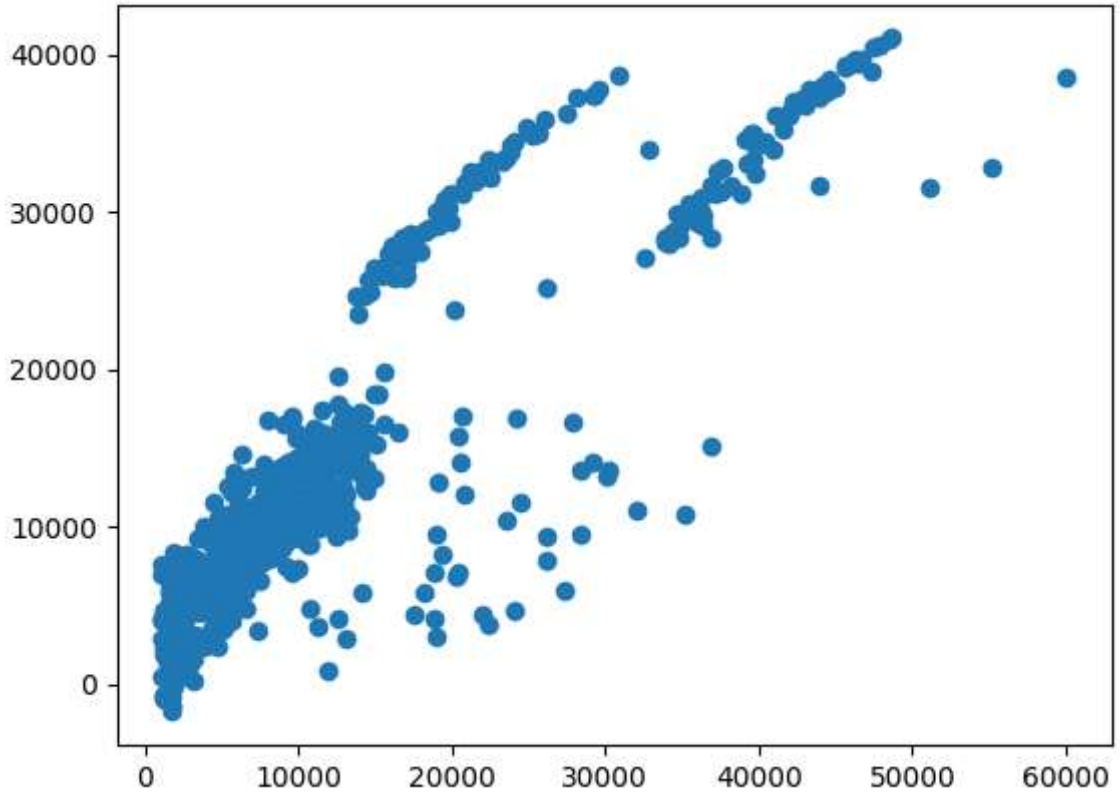
In [21]:
```
reg.fit(x_train,y_train)     #fitting the data
```

Out[21]:
```
LinearRegression
LinearRegression()
```

In [22]:
```
#accuracy score
print(reg.score(x_test,y_test))
```

```
0.7546033183870404
```

In [23]:
```
y_pred=reg.predict(x_test)       #predicting target variable
plt.scatter(y_test,y_pred)
plt.show()
```




```
"""Conclusion:
         Here we have predicted the charges for a person based on his age,sex,bmi,
         children,smoker and the region.The accuracy is 75% so,it is somewhat fitted for this dataframe."""
```

Ridge Regression

In [24]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge
```

In [25]:
```
ridge=Ridge(alpha=10)
ridge.fit(x_train,y_train)    #fitting data
```

Out[25]:
```
Ridge
Ridge(alpha=10)
```

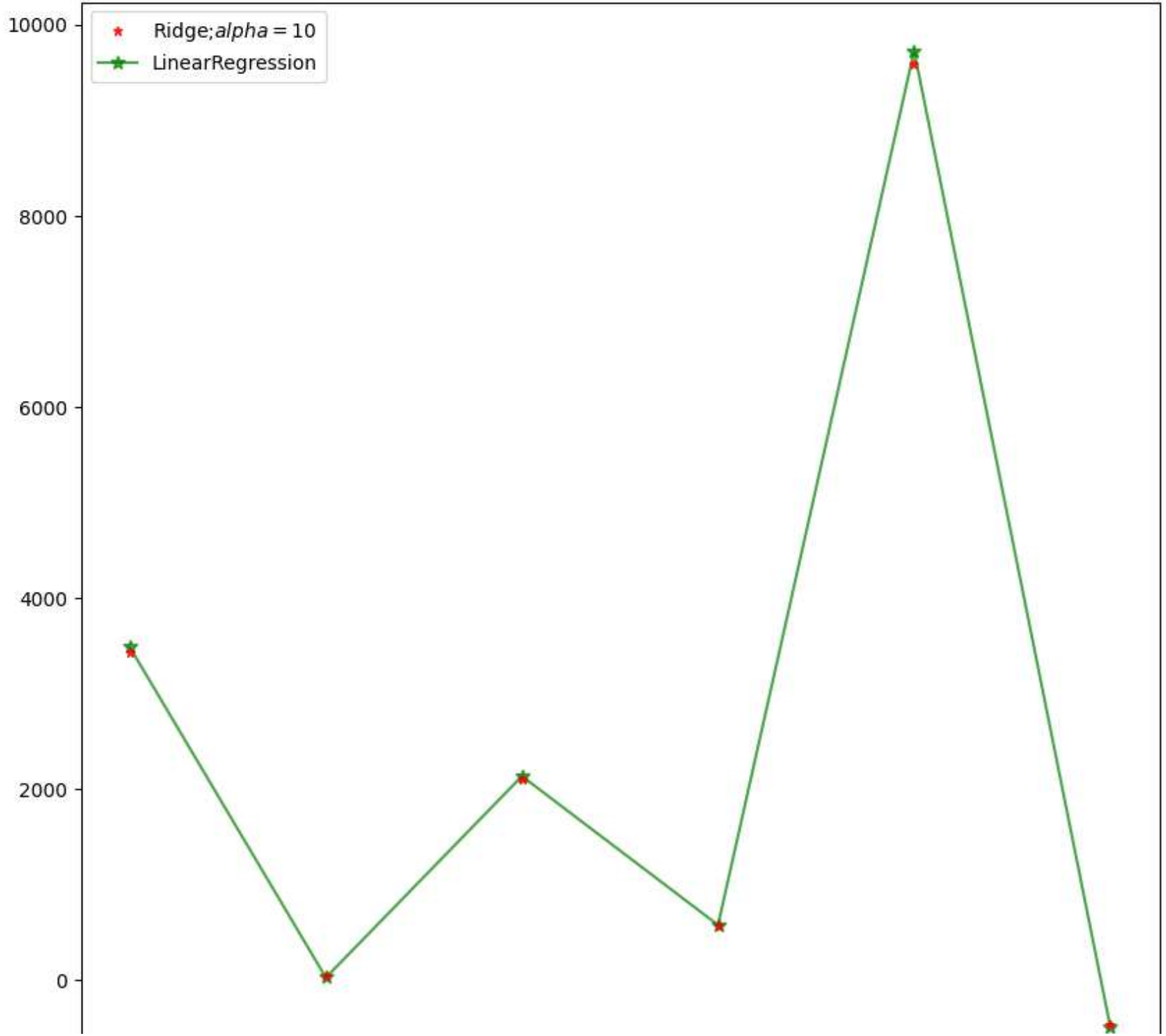
In [26]:
```
train_score_ridge=ridge.score(x_train,y_train)  #checking score for train data
test_score_ridge=ridge.score(x_test,y_test)      #checking score for test data
```

In [27]:
```
print("Ridge 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 is0.741091563188467
The test score for ridge model is0.754840671522456
```

In [28]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridge.coef_,alpha=0.8,linestyle="none",marker="*",markersize=5,color="red",
         label=r'Ridge;$alpha=10$',zorder=7)
plt.plot(features,reg.coef_,alpha=0.7,linestyle=None,marker="*",
         markersize=7,color="green",label="LinearRegression")
plt.xticks(rotation=90)
plt.legend()
plt.show()
```


Ridge;alpha = 10
LinearRegression
10000
8000
6000
4000
2000
0

Lasso Regression

In [29]:
```
from sklearn.linear_model import Lasso
```

In [30]:
```
lasso=Ridge(alpha=10)
lasso.fit(x_train,y_train)  #fitting the data
```

Out[30]:
```
Ridge(alpha=10)
```

In [31]:
```
train_score_lasso=lasso.score(x_train,y_train)
test_score_lasso=lasso.score(x_test,y_test)   #checking the score
```

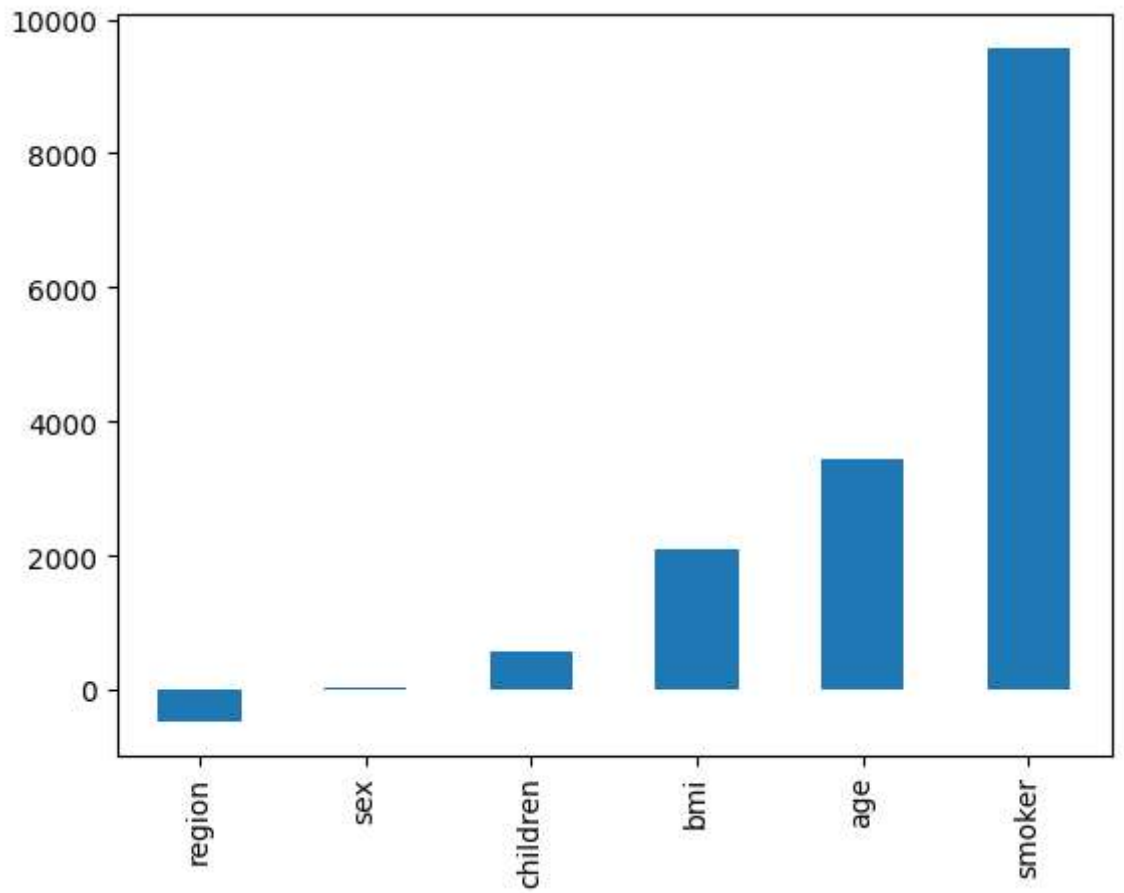
In [32]:
```
print("Lasso Model:\n")
print("The train score for Lasso model is{}".format(train_score_ridge))
print("The test score for Lasso model is{}".format(test_score_ridge))
```

```
Lasso Model:

The train score for Lasso model is0.741091563188467
The test score for Lasso model is0.754840671522456
```

In [33]:
```
pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

Out[33]: <Axes: >



Elastic Net

In [34]:
```
from sklearn.linear_model import ElasticNet
```

In [35]:
```
ec=ElasticNet()    #calling elastic net into ec
```

In [36]:
```
ec.fit(x,y)      #fitting the model
```

Out[36]:
```
ElasticNet
ElasticNet()
```

In [37]:
```
y_pred=ec.predict(x_train)
mean_squared_error=np.mean((y_pred-y_train)**2)   #error
print(mean_squared_error)
```

```
524927067.3880748
```

In [38]:
```
ec.score(x,y)   #accuracy
```

Out[38]:
```
0.3927384383881489
```

Logistic Regression

In [39]:
```
from sklearn.linear_model import LogisticRegression
```

In [40]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [41]:
```
print("This DataFrame has %d rows and %d columns"%(df.shape))
```

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

In [42]:
```
df.head()   #displaying the first 5 rows
```

Out[42]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	4	16884.92400
1	18	1	33.770	1	0	3	1725.55230
2	28	1	33.000	3	0	3	4449.46200
3	33	1	22.705	0	0	2	21984.47061
4	32	1	28.880	0	0	2	3866.85520

"""For this logistic Regression we cannot take charges as a target because it consists of continuous values.so,we have taken age,sex,bmi and region as features and smoker as an output column."""

In [43]:
```
features_matrix=df[["age","sex","bmi","region"]]
features_matrix.columns=["age","sex","bmi","region"]
target_matrix=df.iloc[:,-3]
```

In [44]:
```
print("The feature matrix has %d rows and %d columns"%(features_matrix.shape))
```

```
The feature matrix has 1338 rows and 4 columns
```

In [45]:
```
print("The target matrix has %d rows and %d columns"%(np.array(target_matrix).reshape(-1,1).shape))
```

```
The target matrix has 1338 rows and 1 columns
```

In [46]:
```
#fitting the data
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [47]:
```
algorithm=LogisticRegression(max_iter=100)   #calling the function
```

In [48]:
```
Logistic_Regression_model=algorithm.fit(features_matrix_standardized,target_matrix)
```

In [49]:
```
observation=[[28,1,28.880,2]]    #observation values taken randomly from the feature columns
```

In [50]:
```
predictions=Logistic_Regression_model.predict(observation)  #prediction
```

In [51]:
```
print("The model prdicted the observation belong to class %s"%(predictions))
```

```
The model prdicted the observation belong to class [0]
```

In [52]:
```
print("The model says the probability of the observation we passed belonging to class['0'] is %s"
      %(algorithm.predict_proba(observation)[0][0]))
```

```
The model says the probability of the observation we passed belonging to class['0'] is 0.9315964753030876
```

"'with including children column the accuracy is 0.929785829605 since it is not neccesary
we excluded it"'

In [53]:
```
print("The model says the probability of the observation we passed belonging to class['1'] is %s"
      %(algorithm.predict_proba(observation)[0][1]))
```

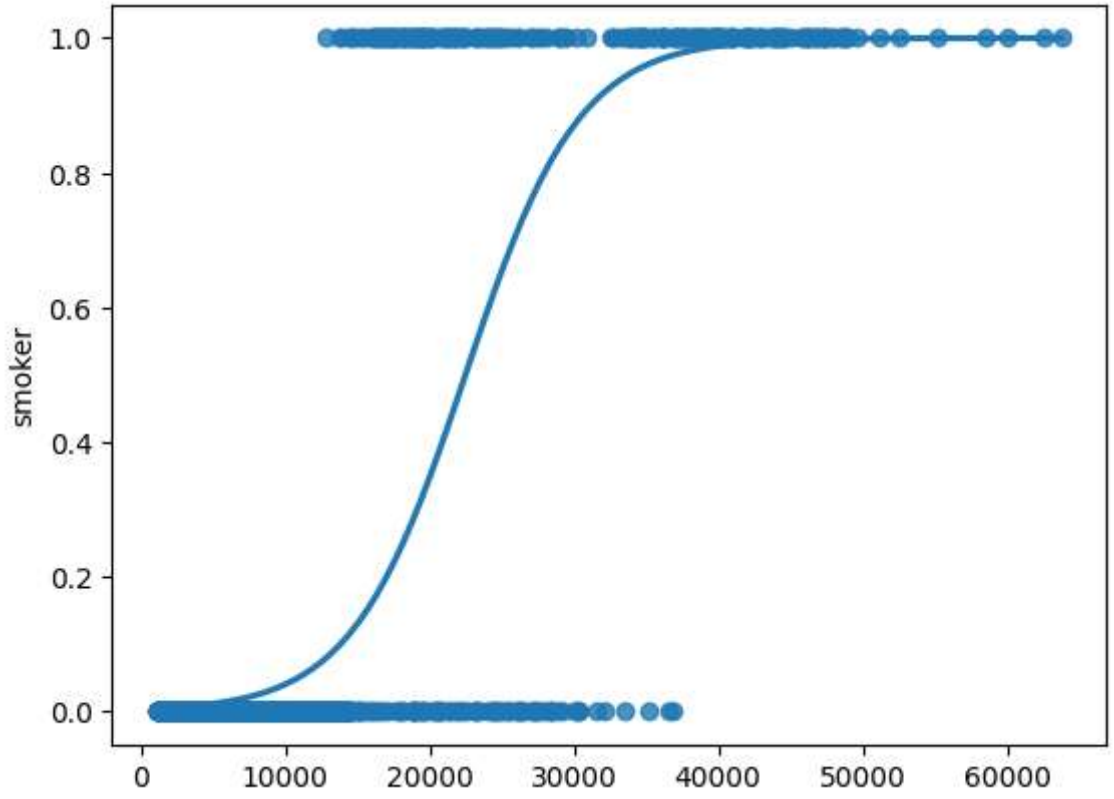
```
The model says the probability of the observation we passed belonging to class['1'] is 0.06840352469691234
```

In [54]:
```
x1=np.array(df["charges"]).reshape(-1,1)
```

"'with including children column the accuracy is 0.070214170394 since it is not neccesary
we excluded it"'

In [55]:
```
sns.regplot(x=x1,y=target_matrix,data=df,logistic=True,ci=None)  #plotting the graph
```

Out[55]:
```
<Axes: ylabel='smoker'>
```



"""Conclusion: The score for Logistic Regression is 93.1% it is a best model for this dataset"""

LinearRegression

In [56]:
```
df.drop("charges",axis=1)
```

35	19	1	20.425	0	0	2
36	62	0	32.965	3	0	2
37	26	1	20.800	0	0	4
38	35	1	36.670	1	1	1
39	60	1	39.900	0	1	4
40	24	0	26.600	0	0	1
41	31	0	36.630	2	0	3
42	41	1	21.780	1	0	3
43	37	0	30.800	2	0	3
44	38	1	37.050	1	0	1
45	55	1	37.300	0	0	4
46	18	0	38.665	2	0	1
47	28	0	34.770	0	0	2

In [57]:
```
x=df[["age","sex","bmi","region"]]
x.columns=["age","sex","bmi","region"]
x
```

Out[57]:

	age	sex	bmi	region
0	19	0	27.900	4
1	18	1	33.770	3
2	28	1	33.000	3
3	33	1	22.705	2
4	32	1	28.880	2
5	31	0	25.740	3
6	46	0	33.440	3
7	37	0	27.740	2
8	37	1	29.830	1
9	60	0	25.840	2
10	25	1	26.220	1

In [58]:
```
y=df[["smoker"]]
y.columns=["smoker"]
y
```

Out[58]:

	smoker
0	1
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

In [59]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.50,random_state=17)
from sklearn.linear_model import LinearRegression  #splitting the data into
reg=LinearRegression()
```

In [60]:
```
reg.fit(x_train,y_train)
```

Out[60]:
```
LinearRegression
LinearRegression()
```

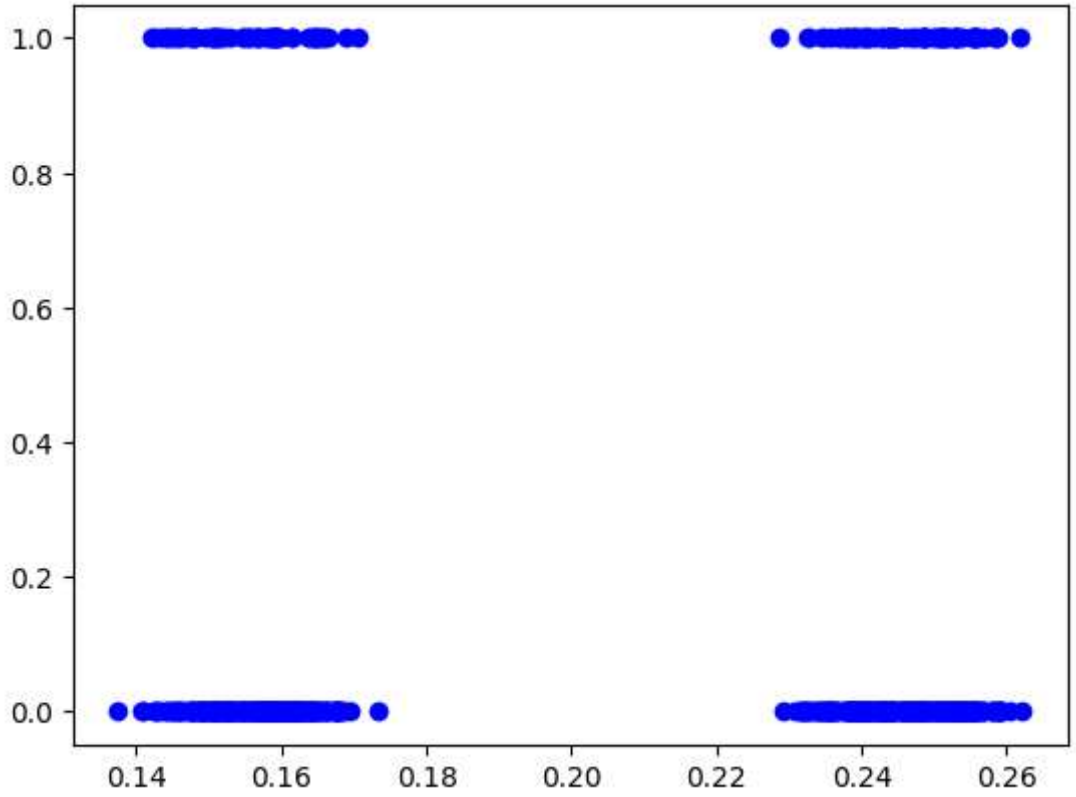
In [61]:
```
print(reg.score(x_train,y_train))
```

```
0.013065640760836161
```

In [62]:
```
print(reg.score(x_test,y_test))
```

```
-0.004285100055760882
```

In [63]:
```
y_pred=reg.predict(x_test)
plt.scatter(y_pred,y_test,color="b")
plt.show()
```




```
"""Conclusion:
         For Linear Regression the acccuracy is very low i.e., 4%. so, it is unfit for this dataset"
```

Decision Trees

In [64]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [65]:
```
x=["age","sex","bmi","region"]
y=[0,1]
all_inputs=df[x]
all_classes=df["smoker"]
```

In [66]:
```
x_train,x_test,y_train,y_test=train_test_split(all_inputs,all_classes,test_size=0.25)
```

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

In [68]:
```
clf.fit(x_train,y_train)                #Fitting training into th model (DecisionTreeClassifier)
```

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

In [69]:
```
score=clf.score(x_test,y_test)#To find the score for test data
```

In [70]:
```
print(score)  # 0.6626865671641791
```

```
0.6477611940298508
```

In [71]:
```
clf.score(x_train,y_train)        #To find the score for training data
```

Out[71]:
```
0.9990029910269193
```

```
"""Conclusion:
         The score for the training data is 99% and for testing data is 65% so,this model is fit
         for the training data,whereas it is unfit for the testing data.It is overfitted"""
```

Random Forest

In [72]:
```
df1=df.drop("charges",axis=1)
df1
```

Out[72]:

	age	sex	bmi	children	smoker	region
0	19	0	27.900	0	1	4
1	18	1	33.770	1	0	3
2	28	1	33.000	3	0	3
3	33	1	22.705	0	0	2
4	32	1	28.880	0	0	2
5	31	0	25.740	0	0	3
6	46	0	33.440	1	0	3
7	37	0	27.740	3	0	2
8	37	1	29.830	2	0	1
9	60	0	25.840	0	0	2
10	25	1	26.220	0	0	1

In [73]:
```
df2=df1.drop("children",axis=1)
df2
```

Out[73]:

	age	sex	bmi	smoker	region
0	19	0	27.900	1	4
1	18	1	33.770	0	3
2	28	1	33.000	0	3
3	33	1	22.705	0	2
4	32	1	28.880	0	2
5	31	0	25.740	0	3
6	46	0	33.440	0	3
7	37	0	27.740	0	2
8	37	1	29.830	0	1
9	60	0	25.840	0	2
10	25	1	26.220	0	1

In [74]:
```
x=df2.drop("smoker",axis=1)
y=df2["smoker"]
```

In [75]:
```
x
```

Out[75]:

	age	sex	bmi	region
0	19	0	27.900	4
1	18	1	33.770	3
2	28	1	33.000	3
3	33	1	22.705	2
4	32	1	28.880	2
5	31	0	25.740	3
6	46	0	33.440	3
7	37	0	27.740	2
8	37	1	29.830	1
9	60	0	25.840	2
10	25	1	26.220	1

In [76]:
```
y
```

Out[76]:
```
0       1
1       0
2       0
3       0
4       0
5       0
6       0
7       0
8       0
9       0
10      0
11      1
12      0
13      0
14      1
15      0
16      0
17      0
18      0
```

In [77]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
x_train.shape,x_test.shape
```

Out[77]:
```
((1003, 4), (335, 4))
```

In [78]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

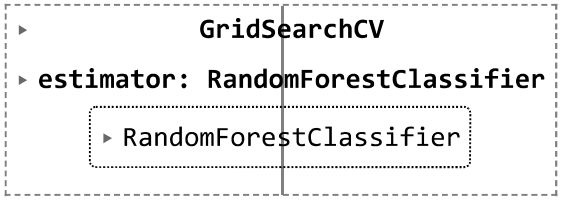
Out[78]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [79]:
```
rf=RandomForestClassifier()
```

In [80]:
```
params={'max_depth':[2,3,5,10,20],
       'min_samples_leaf':[5,10,20,50,100,200],
       'n_estimators':[10,25,30,50,100,500]}
```

In [81]:
```
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[81]:



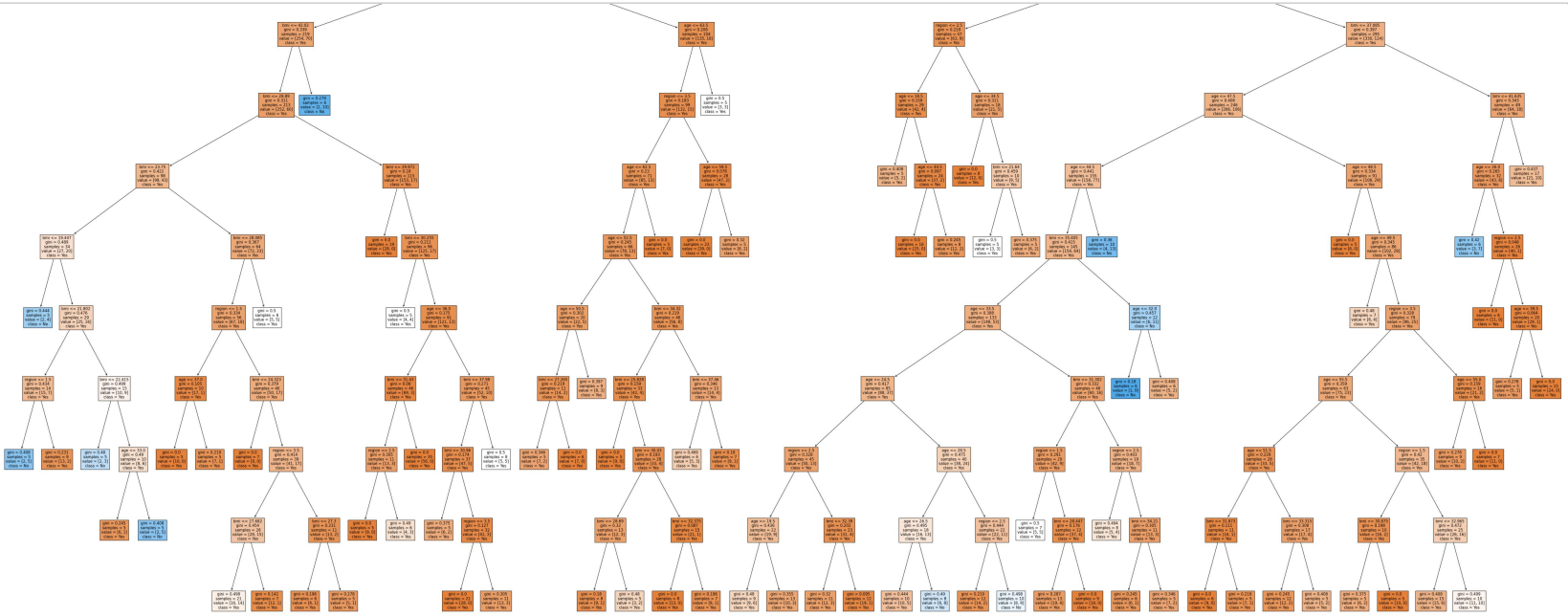
In [82]:
```
grid_search.best_score_    #finding accuracy  #0.7976079713083792
```

Out[82]:
```
0.7986059753003952
```

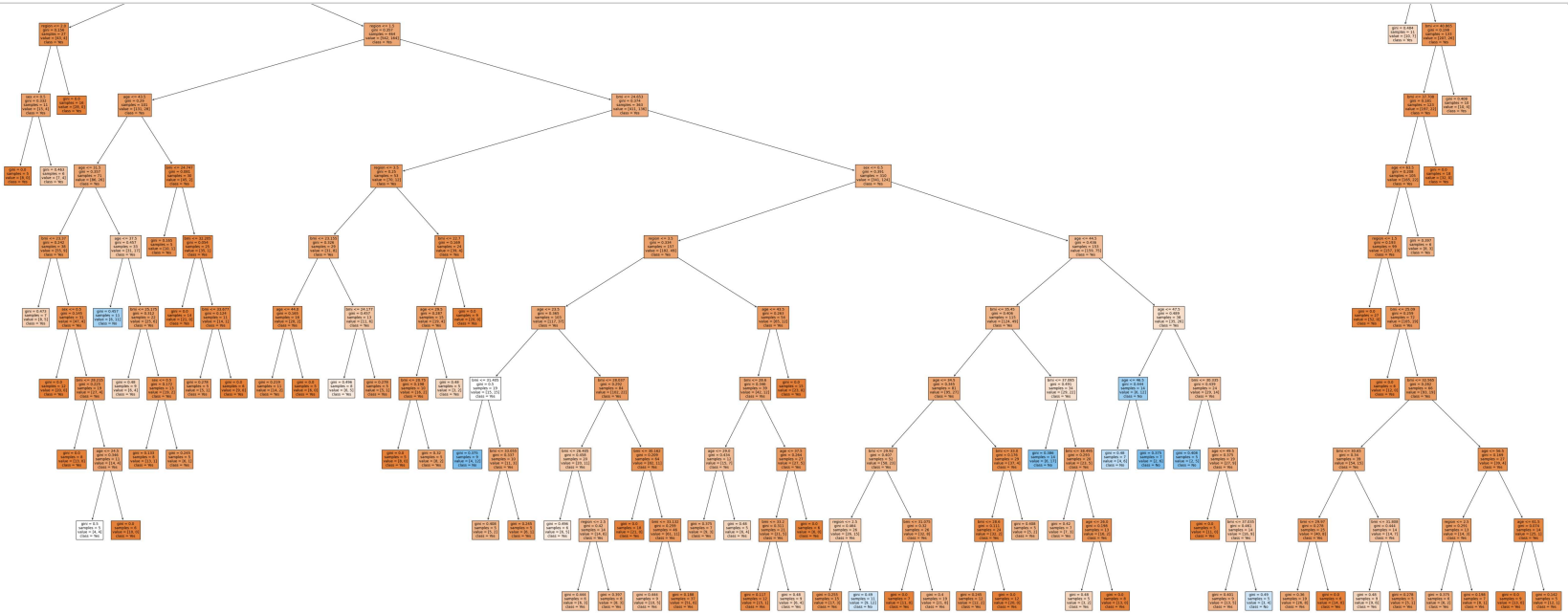
In [83]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=10, min_samples_leaf=5, n_estimators=500)
```

In [84]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],filled=True)
```



In [85]:
```
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)
```



In [86]:
```
rf_best.feature_importances_
```

Out[86]:
```
array([0.37949307, 0.05211943, 0.46491543, 0.10347207])
```

In [87]:
```
imp_df=pd.DataFrame({"varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[87]:

	varname	Imp
2	bmi	0.464915
0	age	0.379493
3	region	0.103472
1	sex	0.052119

In [88]:
```
print(rfc.score(x_train,y_train))
```

```
0.9990029910269193
```

In [89]:
```
print(rfc.score(x_test,y_test))
```

```
0.7164179104477612
```

```
"""Conclusion:
         The score for the training data is 99% and for testing data is 71% so,this model is fit
         for the training data,whereas it is unfit for the testing data.It is overfitted."""
```

Conclusion:

Here for all the models the feature columns are age,sex,bmi and the region they belongs to and the target column is whether a person is smoker or not.
After checking and comparing the scores for all other models LodisticRegression is best suited for this dataframe with the accuracy of 93.16%

Model saving for all the models

For LinearRegression

In [90]:
```
import pickle
fname="y_pred"
pickle.dump(reg,open(fname,'wb'))
```

For Logistic Regression

In [91]:
```
import pickle
fname="pred"
pickle.dump(algorithm,open(fname,'wb'))
```