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
from sklearn.preprocessing import StandardScaler
```

In [2]:

```
df=pd.read_csv(r"C:\Users\sweet\Downloads\insuranceex.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 and Preprocessing

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]:

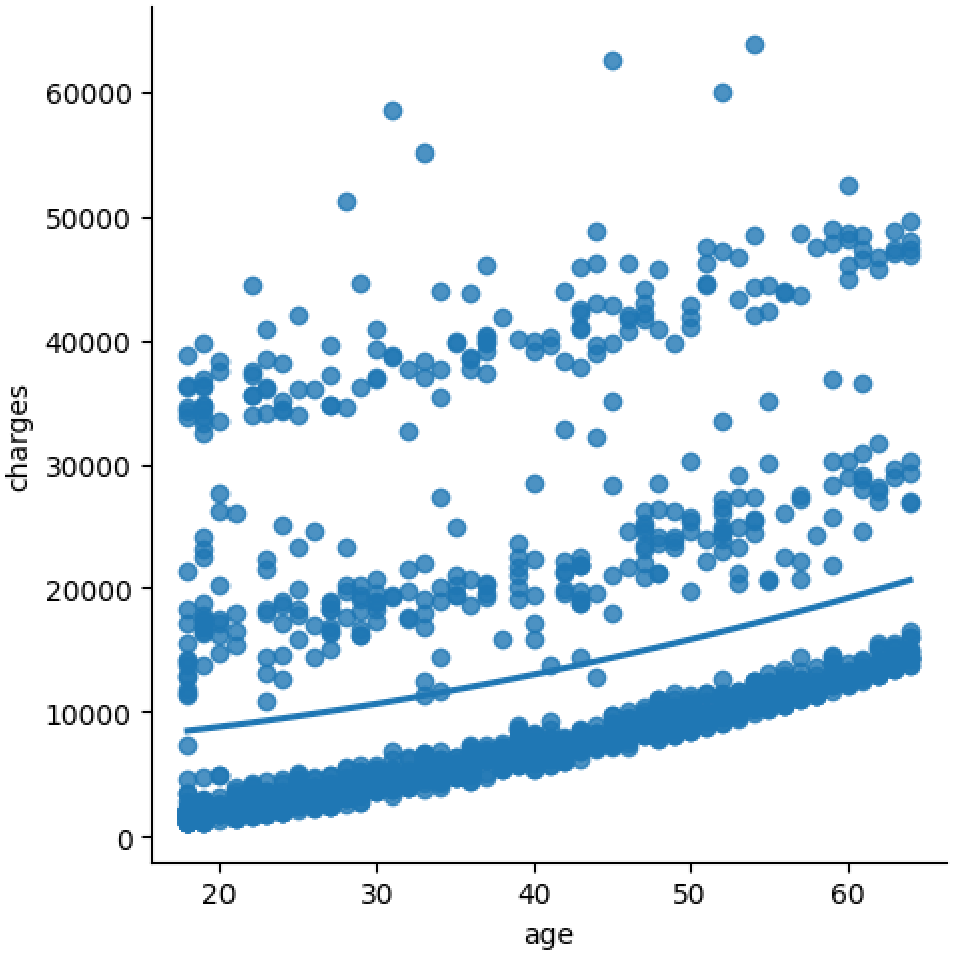
	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

In [7]:
```
df.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]: sns.lmplot(x="age",y="charges",data=df,order=2,ci=None)
```

```
Out[8]: <seaborn.axisgrid.FacetGrid at 0x12592b72250>
```



```
In [9]: df.fillna(method='ffill',inplace=True)
```

In [10]:

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

In [11]:

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

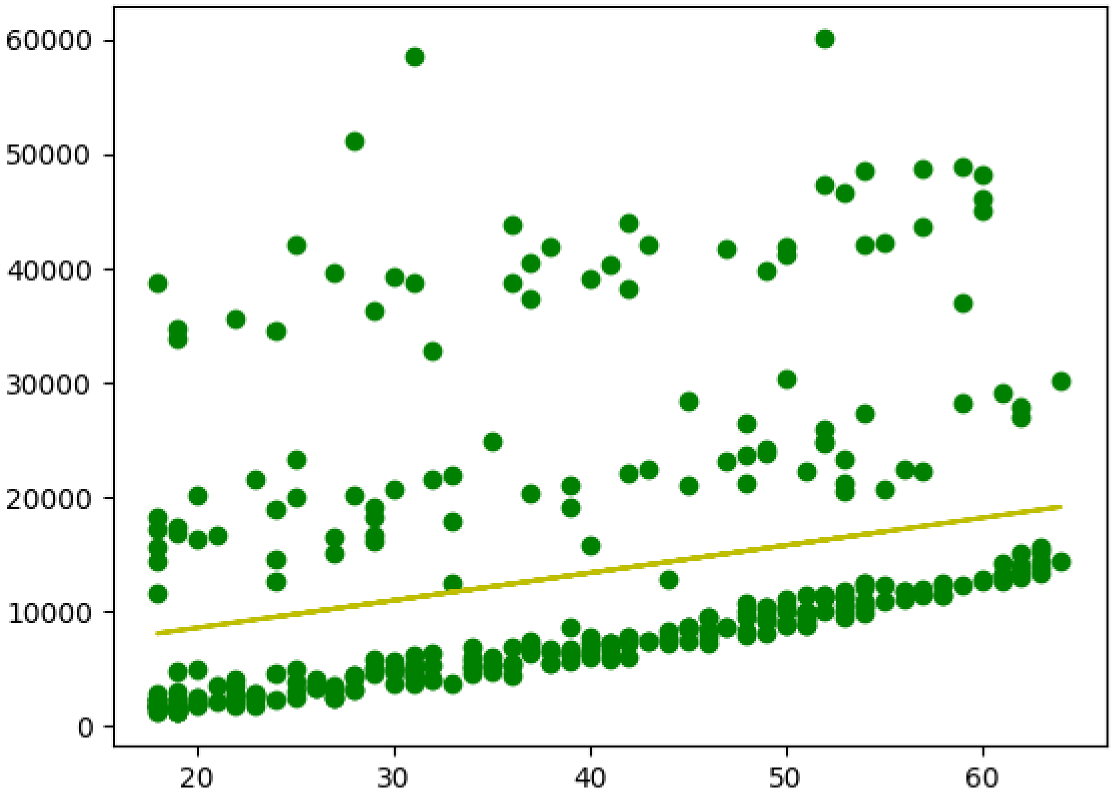
In [12]:

```
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))
```

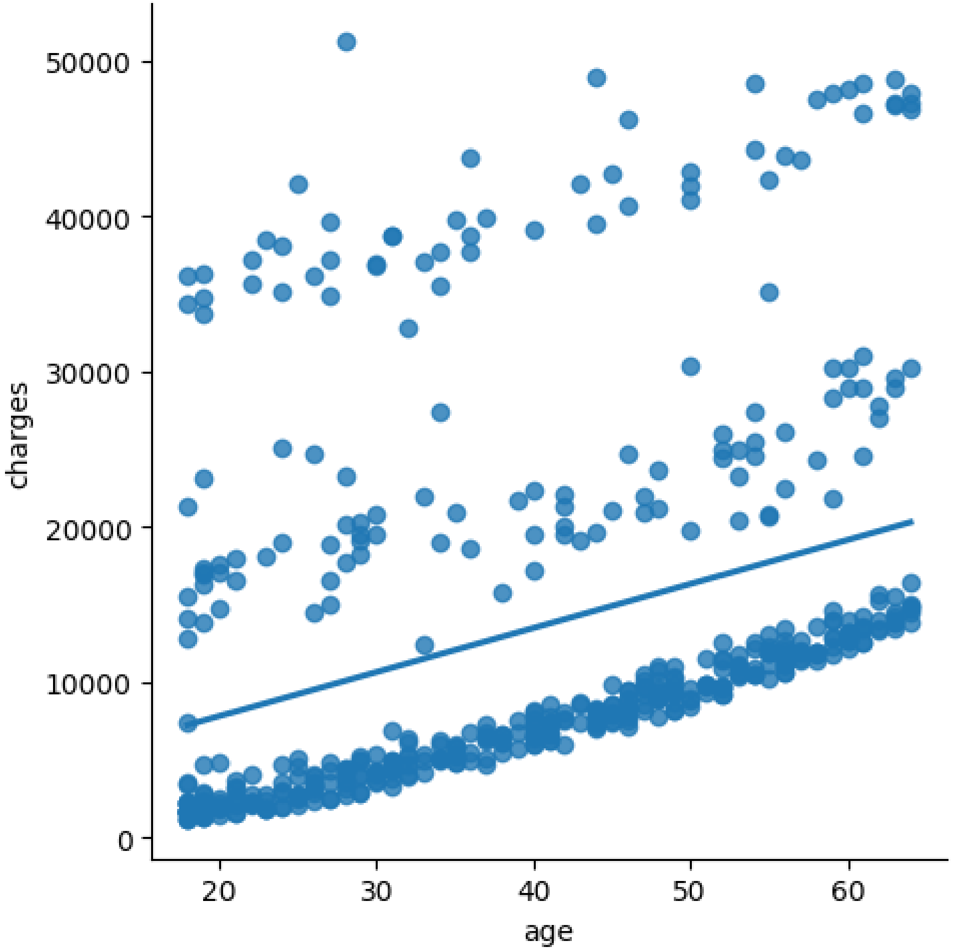
```
0.10968904282635228
```

In [13]:

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



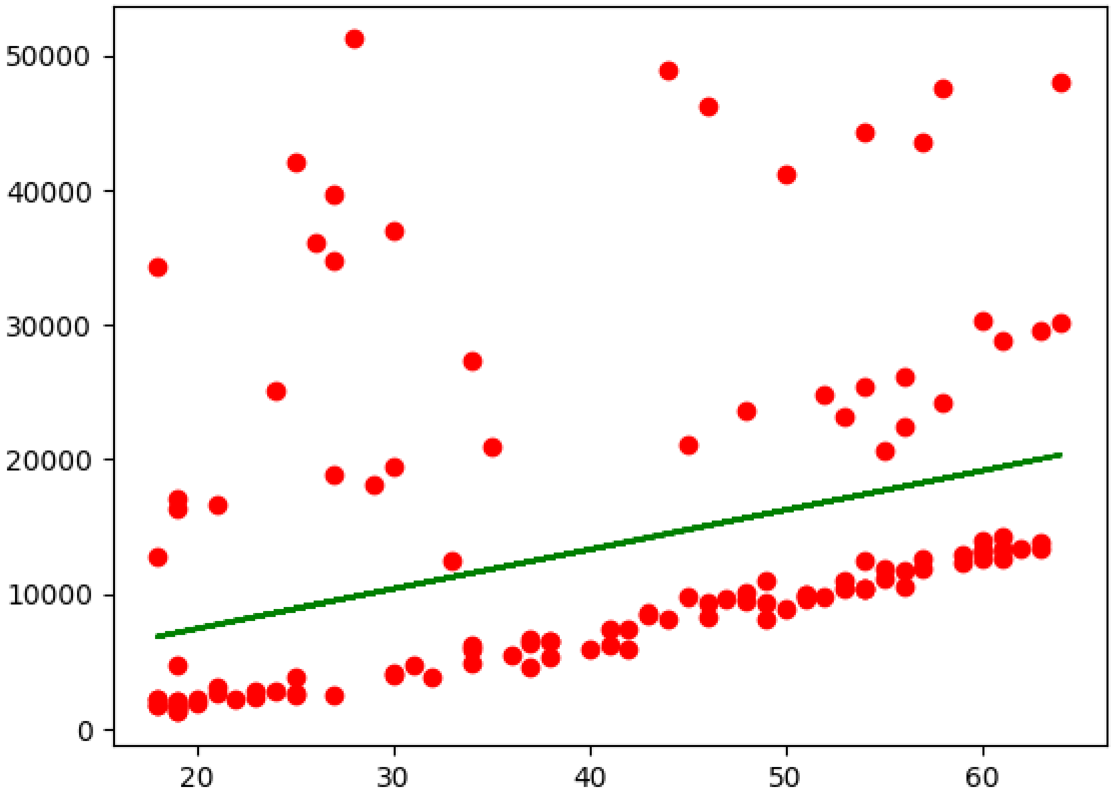
```
In [14]: df500=df[:][:500]
         sns.lmplot(x="age",y="charges",data=df500,order=1,ci=None)

Out[14]: <seaborn.axisgrid.FacetGrid at 0x12592c25f90>
```



In [15]:

```
df500.fillna(method='ffill',inplace=True)
x=np.array(df500['age']).reshape(-1,1)
y=np.array(df500['charges']).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='r')
plt.plot(x_test,y_pred,color='g')
plt.show()
```

```
Regression: 0.09329384184108624
```



In [16]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 Score:",r2)
```

```
R2 Score: 0.09329384184108624
```

In [17]:

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

Out[17]:

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

Implemention of Ridgeand Lasso Regression model

In [22]:

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

In [23]:

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

Out[23]:

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

1338 rows × 7 columns
```

In [24]:

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

Out[24]:

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

1338 rows × 7 columns

In [25]:

```
convert={"region":{"southeast":3,"southwest":4,"northeast":5,"northwest":6}}
df=df.replace(convert)
df
```

Out[25]:

```
      age  sex     bmi  children  smoker  region      charges
   0   19    2  27.900         0       1       4  16884.92400
   1   18    1  33.770         1       2       3   1725.55230
   2   28    1  33.000         3       2       3   4449.46200
   3   33    1  22.705         0       2       6  21984.47061
   4   32    1  28.880         0       2       6   3866.85520
 ...  ...  ...     ...       ...     ...     ...          ...
1333   50    1  30.970         3       2       6  10600.54830
1334   18    2  31.920         0       2       5   2205.98080
1335   18    2  36.850         0       2       3   1629.83350
1336   21    2  25.800         0       2       4   2007.94500
1337   61    2  29.070         0       1       6  29141.36030

1338 rows × 7 columns
```

In [27]:

```
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)
```

In [28]:

```
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
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 0.07446228994221393
The test score for ridge model is 0.10855133360950642
```

In [29]:

```
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))
```

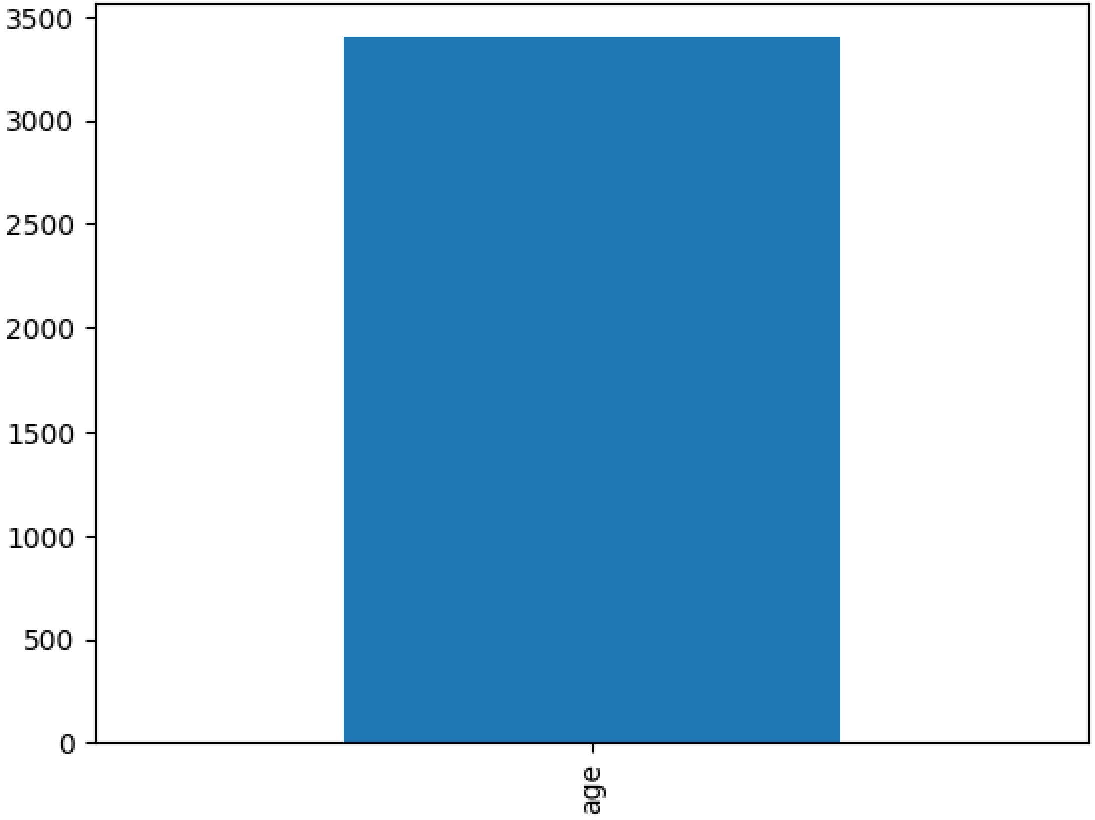
```

Linear Regression Model:

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

In [31]:

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



In [38]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).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
```

In [40]:

```
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='Ridge;$\alpha=
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.6,linestyle='none',marker='d',markersize=6,color='blue',label='Ridge;$\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()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[40], line 3
      1 plt.figure(figsize=(10,10))
      2 #add plot for ridge regression
----> 3 plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label='Rid
ge;$\alpha=10$',border=7)
      4 #add plot for lasso regression
      5 plt.plot(lasso_cv.coef_,alpha=0.6,linestyle='none',marker='d',markersize=6,color='blue',label='Ridge;$\alp
ha=grid$')

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\matplotlib\pyplot.py:2812, in plot(scalex, scale
y, data, *args, **kwargs)
   2810 @_copy_docstring_and_deprecators(Axes.plot)
   2811 def plot(*args, scalex=True, scaley=True, data=None, **kwargs):
-> 2812     return gca().plot(
   2813         *args, scalex=scalex, scaley=scaley,
   2814         **({"data": data} if data is not None else {}), **kwargs)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\matplotlib\axes\_axes.py:1688, in Axes.plot(self
```

In [42]:

```
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(ridge_cv.score(X_train, y_train))
print(ridge_cv.score(X_test, y_test))
```

```
0.07446228994221393
0.10855133360950775
```

Elastic net regression

In [43]:

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

```
[257.0684655]
3191.532406056682
```

Out[43]:

```
0.08940532368214038
```

In [44]:

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

In [45]:

```
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 267460995.25217086
```

Logistic Regression

In [46]:

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

In [47]:

```
df=pd.read_csv(r"C:\Users\sweet\Downloads\insuranceex.csv")
df
```

Out[47]:

	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 [48]:

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

In [49]:

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

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

In [50]:

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

Out[50]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	1	southwest	16884.924000
1	18	male	33.770	1	2	southeast	1725.552300
2	28	male	33.000	3	2	southeast	4449.462000
3	33	male	22.705	0	2	northwest	21984.470610
4	32	male	28.880	0	2	northwest	3866.855200
5	31	female	25.740	0	2	southeast	3756.621600
6	46	female	33.440	1	2	southeast	8240.589600
7	37	female	27.740	3	2	northwest	7281.505600
8	37	male	29.830	2	2	northeast	6406.410700
9	60	female	25.840	0	2	northwest	28923.136920
10	25	male	26.220	0	2	northeast	2721.320800
11	62	female	26.290	0	1	southeast	27808.725100

In [51]:

```
convert={"sex":{"male":8,"female":9}}
df=df.replace(convert)
df
```

Out[51]:

	age	sex	bmi	children	smoker	region	charges
0	19	9	27.900	0	1	southwest	16884.924000
1	18	8	33.770	1	2	southeast	1725.552300
2	28	8	33.000	3	2	southeast	4449.462000
3	33	8	22.705	0	2	northwest	21984.470610
4	32	8	28.880	0	2	northwest	3866.855200
5	31	9	25.740	0	2	southeast	3756.621600
6	46	9	33.440	1	2	southeast	8240.589600
7	37	9	27.740	3	2	northwest	7281.505600
8	37	8	29.830	2	2	northeast	6406.410700
9	60	9	25.840	0	2	northwest	28923.136920
10	25	8	26.220	0	2	northeast	2721.320800
11	62	9	26.290	0	1	southeast	27808.725100

In [52]:

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

In [53]:

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

In [55]:

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

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

In [57]:

```
print('The Target Matrix Has %d Rows And %d Column(s)'%(np.array(target_vector).reshape(-1,1).shape))
```

```
The Target Matrix Has 1338 Rows And 1 Column(s)
```

In [58]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [60]:
```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_intercept=True,intercept_scaling=1,class_weigh
```

In [62]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [63]:
```
observation=[[1,0,0.99539,-0.05889,]]
```

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 [2]
```

In [67]:
```
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: [1 2]
```

In [85]:
```
print(" " "The model says the probability of the observation we passed belonging to class[0] Is %s" " "%(algorithm.pre
```

```
 The model says the probability of the observation we passed belonging to class[0] Is 0.1942921563693959
```

In [86]:
```
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 0.1942921563693959
```

In [87]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
```

In [88]:

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

```
0.7952167414050823

C:\Users\sweet\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\validation.py:1143: DataConve
rsionWarning: A column-vector 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)
```

decision tree regression

In [89]:

```
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 [90]:
```
df=pd.read_csv(r"C:\Users\sweet\Downloads\insuranceex.csv")
df
```

Out[90]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.924000
1	18	male	33.770	1	no	southeast	1725.552300
2	28	male	33.000	3	no	southeast	4449.462000
3	33	male	22.705	0	no	northwest	21984.470610
4	32	male	28.880	0	no	northwest	3866.855200
5	31	female	25.740	0	no	southeast	3756.621600
6	46	female	33.440	1	no	southeast	8240.589600
7	37	female	27.740	3	no	northwest	7281.505600
8	37	male	29.830	2	no	northeast	6406.410700
9	60	female	25.840	0	no	northwest	28923.136920
10	25	male	26.220	0	no	northeast	2721.320800
11	62	female	26.290	0	yes	southeast	27808.725100

In [91]:
```
df['region'].value_counts()
```

Out[91]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

```
In [92]: df['bmi'].value_counts()
```

```
Out[92]: bmi
32.300    13
28.310     9
30.495     8
30.875     8
31.350     8
30.800     8
34.100     8
28.880     8
33.330     7
35.200     7
25.800     7
32.775     7
27.645     7
32.110     7
38.060     7
25.460     7
30.590     7
27.360     7
```

In [93]:
```
convert={"sex":{"male":1,"female":0}}
df=df.replace(convert)
df
```

```
175  63  0  37.700  0  yes  southwest  48824.450000
176  38  1  27.835  2   no  northwest   6455.862650
177  54  1  29.200  1   no  southwest  10436.096000
178  46  0  28.900  2   no  southwest   8823.279000
179  41  0  33.155  3   no  northeast   8538.288450
180  58  1  28.595  0   no  northwest  11735.879050
181  18  0  38.280  0   no  southeast   1631.821200
182  22  1  19.950  3   no  northeast   4005.422500
183  44  0  26.410  0   no  northwest   7419.477900
184  44  1  30.690  2   no  southeast   7731.427100
185  36  1  41.895  3  yes  northeast  43753.337050
186  26  0  29.920  2   no  southeast   3981.976800
187  30  0  30.900  3   no  southwest   5325.651000
```

In [94]:
```
x=["bmi","children"]
y=["yes","no"]
all_inputs=df[x]
all_classes=df["sex"]
```

In [96]:
```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.03)
```

In [97]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [98]:

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

Out[98]:

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

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [99]:

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

```
0.4878048780487805
```

random forest

In [100]:

```
import pandas as pd
import numpy as ny
import matplotlib.pyplot as plt,seaborn as sns
```

In [101]:

```
df=pd.read_csv(r"C:\Users\sweet\Downloads\insuranceex.csv")
df
```

Out[101]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.924000
1	18	male	33.770	1	no	southeast	1725.552300
2	28	male	33.000	3	no	southeast	4449.462000
3	33	male	22.705	0	no	northwest	21984.470610
4	32	male	28.880	0	no	northwest	3866.855200
5	31	female	25.740	0	no	southeast	3756.621600
6	46	female	33.440	1	no	southeast	8240.589600
7	37	female	27.740	3	no	northwest	7281.505600
8	37	male	29.830	2	no	northeast	6406.410700
9	60	female	25.840	0	no	northwest	28923.136920
10	25	male	26.220	0	no	northeast	2721.320800
11	62	female	26.290	0	yes	southeast	27808.725100

In [102]:
```
df['charges'].value_counts()
```

Out[102]:
```
charges
1639.563100     2
16884.924000    1
29330.983150    1
2221.564450     1
19798.054550    1
13063.883000    1
13555.004900    1
44202.653600    1
10422.916650    1
7243.813600     1
11945.132700    1
6311.952000     1
1682.597000     1
5272.175800     1
27218.437250    1
19719.694700    1
4877.981050     1
46255.112500    1
2585.850650     1
```

In [103]:

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

```
122    20  female  28.975         0     no       4   2257.475250
123    44    male  31.350         1    yes       3  39556.494500
124    47  female  33.915         3     no       4  10115.008850
125    26  female  28.785         0     no       3   3385.399150
126    19  female  28.300         0    yes       2  17081.080000
127    52  female  37.400         0     no       2   9634.538000
128    32  female  17.765         2    yes       4  32734.186300
129    38    male  34.700         2     no       2   6082.405000
130    59  female  26.505         0     no       3  12815.444950
131    61  female  22.040         0     no       3  13616.358600
132    53  female  35.900         2     no       2  11163.568000
133    19    male  25.555         0     no       4   1632.564450
134    20  female  28.785         0     no       3   2457.211150
135    22  female  28.050         0     no       1   2155.681500
136    19    male  34.100         0     no       2   1261.442000
137    22    male  25.175         0     no       4   2045.685250
138    54  female  31.900         3     no       1  27322.733860
139    22  female  36.000         0     no       2   2166.732000
140    34    male  22.420         2     no       3  27375.904780
141    26    male  32.490         1     no       3   3490.549100
```

In [106]:

```
df.shape
```

Out[106]:

```
(1338, 7)
```

In [107]:

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

Out[107]:

```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [108]:
```
rf=RandomForestClassifier()
```

In [110]:
```
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 [113]:
```
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[113]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [115]:
```
grid_search.best_score_
```

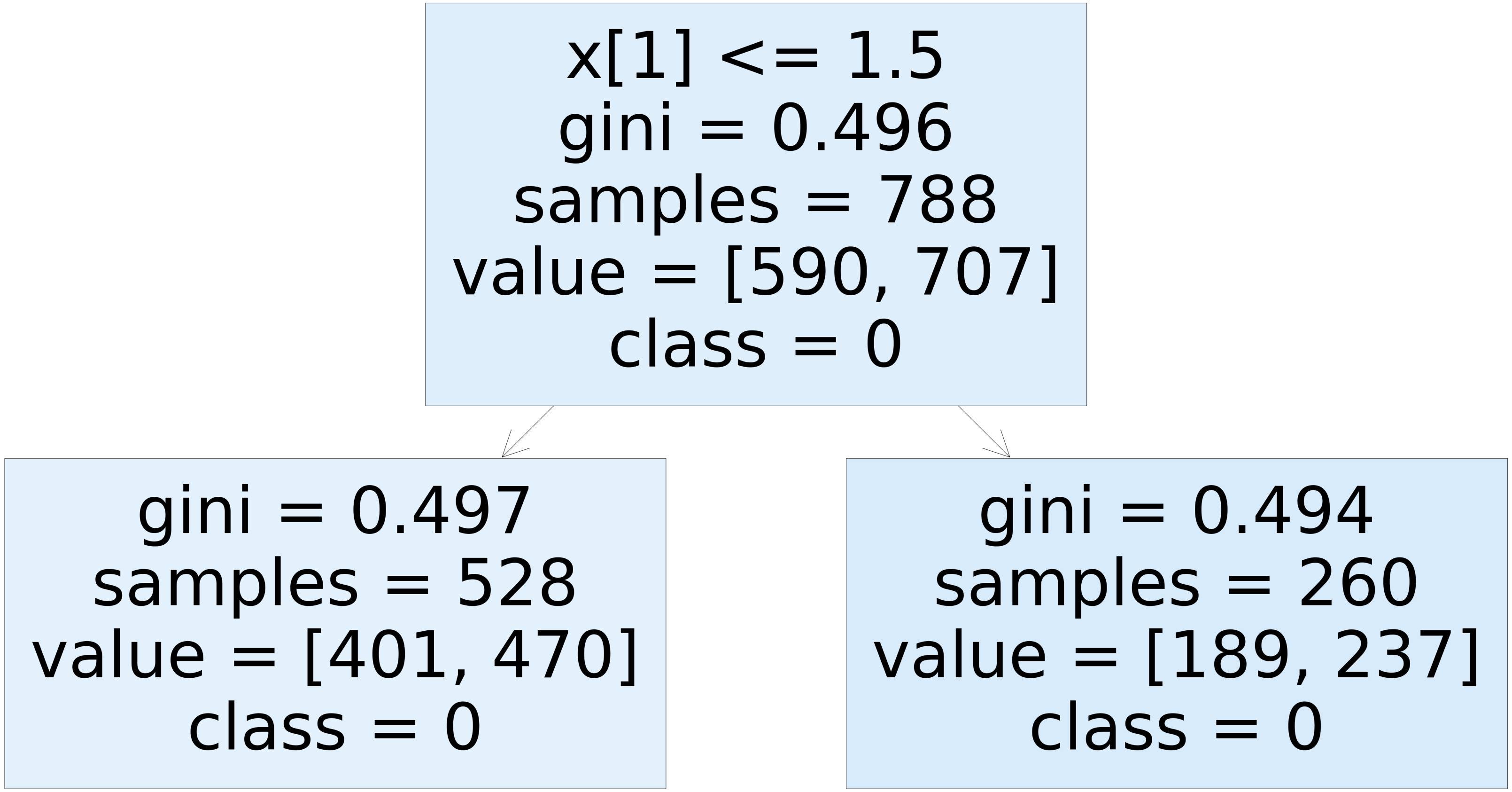
Out[115]:
```
0.5219628012707109
```

In [116]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=20, min_samples_leaf=200, n_estimators=10)
```

In [117]:

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



In [119]:

```
rf_best.feature_importances_
```

Out[119]:

```
array([0.47848867, 0.52151133])
```

In [122]:

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

```
0.4146341463414634
```