Problem Statement:

To predict whether the given dataset is best fit or not.

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.linear_model import Lasso
from sklearn.linear_model import Ridge
```

In [2]:

```
#Step-2:Reading the dataset
df=pd.read_csv(r"C:\Users\91955\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

In [3]:

```
df=df[['age','charges']]
#Taking only the selected two attributes from the dataset
df.columns=['Age','Charges']
#Renaming the columns for easier writing of the code
```

In [4]:

```
df.head(10)
#Displaying only the 1st 10 rows
```

Out[4]:

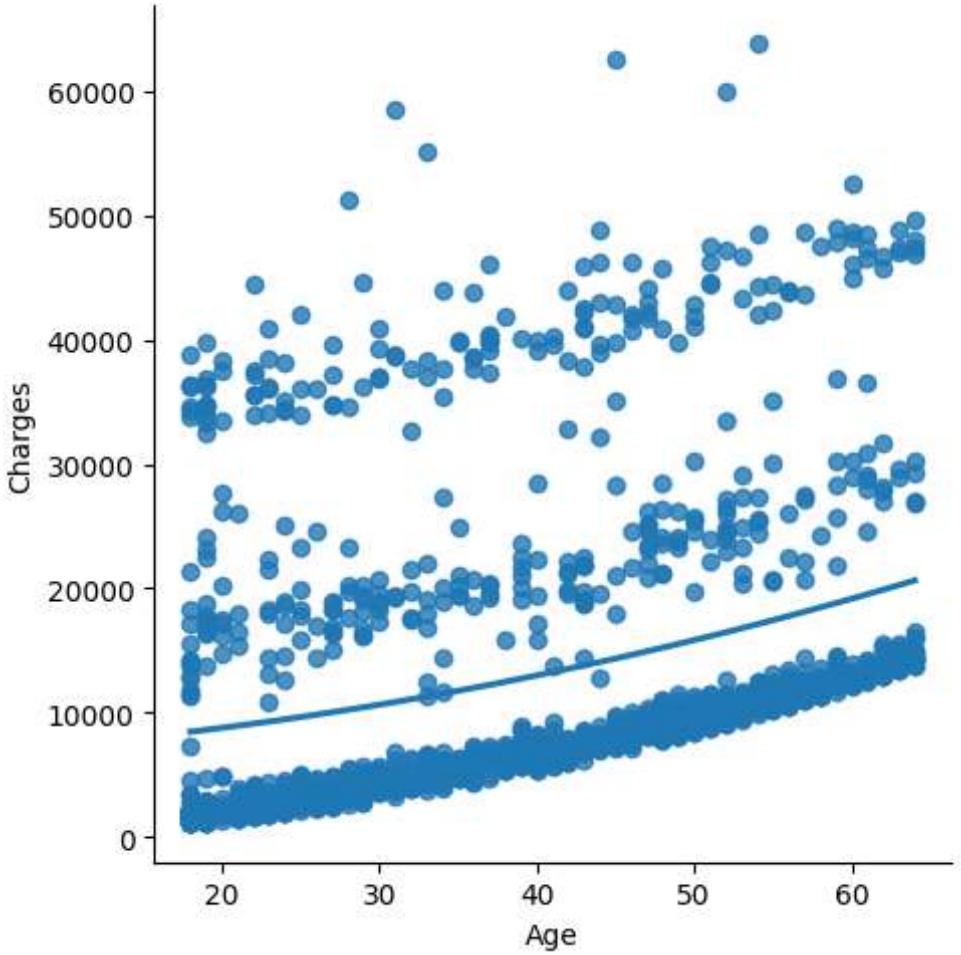
	Age	Charges
0	19	16884.92400
1	18	1725.55230
2	28	4449.46200
3	33	21984.47061
4	32	3866.85520
5	31	3756.62160
6	46	8240.58960
7	37	7281.50560
8	37	6406.41070
9	60	28923.13692

In [5]:

```
sns.lmplot(x='Age',y='Charges',data=df,order=2,ci=None)

```

Out[5]:

```
<seaborn.axisgrid.FacetGrid at 0x16ec2c85e40>
```



In [6]:

```
df.describe()
```

Out[6]:

	Age	Charges
count	1338.000000	1338.000000
mean	39.207025	13270.422265
std	14.049960	12110.011237
min	18.000000	1121.873900
25%	27.000000	4740.287150
50%	39.000000	9382.033000
75%	51.000000	16639.912515
max	64.000000	63770.428010

In [7]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Age      1338 non-null   int64  
 1   Charges  1338 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 21.0 KB
```

In [8]:

```
1 df.fillna(method='ffill')
```

Out[8]:

	Age	Charges
0	19	16884.92400
1	18	1725.55230
2	28	4449.46200
3	33	21984.47061
4	32	3866.85520
...	...	...
1333	50	10600.54830
1334	18	2205.98080
1335	18	1629.83350
1336	21	2007.94500
1337	61	29141.36030

1338 rows × 2 columns

In [9]:

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

Out[9]:

```
Age        0
Charges    0
dtype: int64
```

In [10]:

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

In [11]:

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

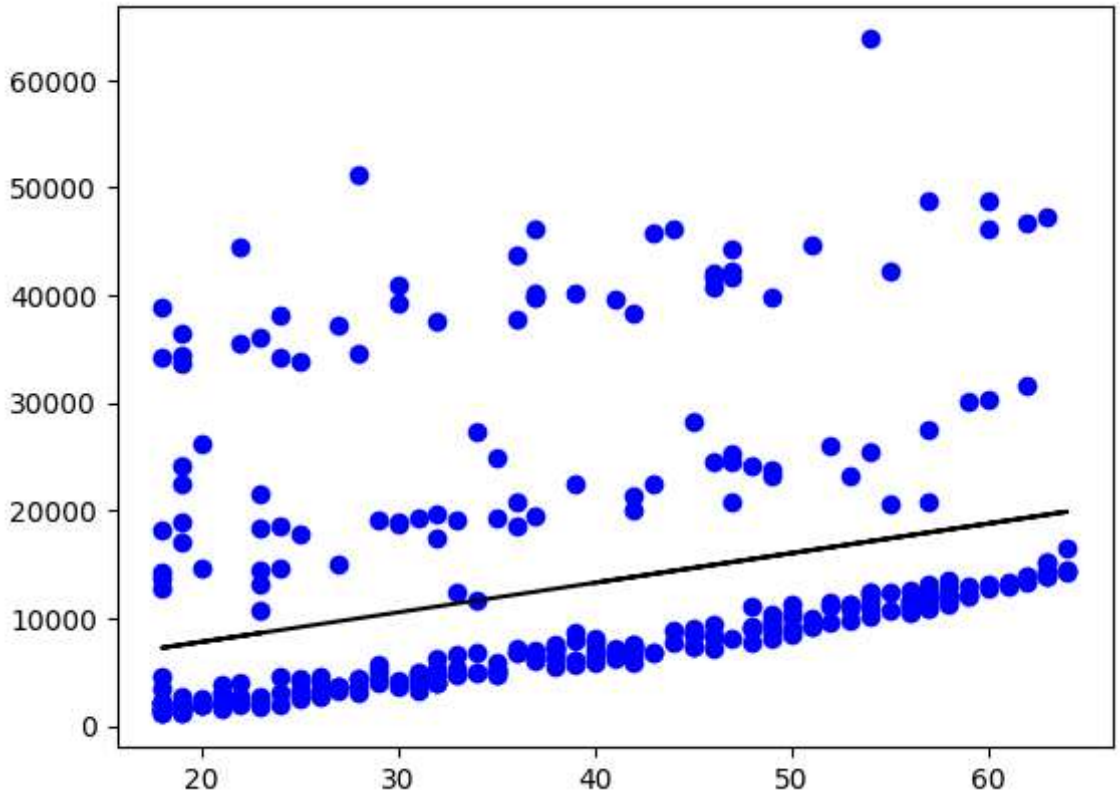
In [12]:

```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
reg=LinearRegression()
reg.fit(X_train,y_train)
print(reg.score(X_test,y_test))
```

```
0.04503038004350668
```

In [13]:

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

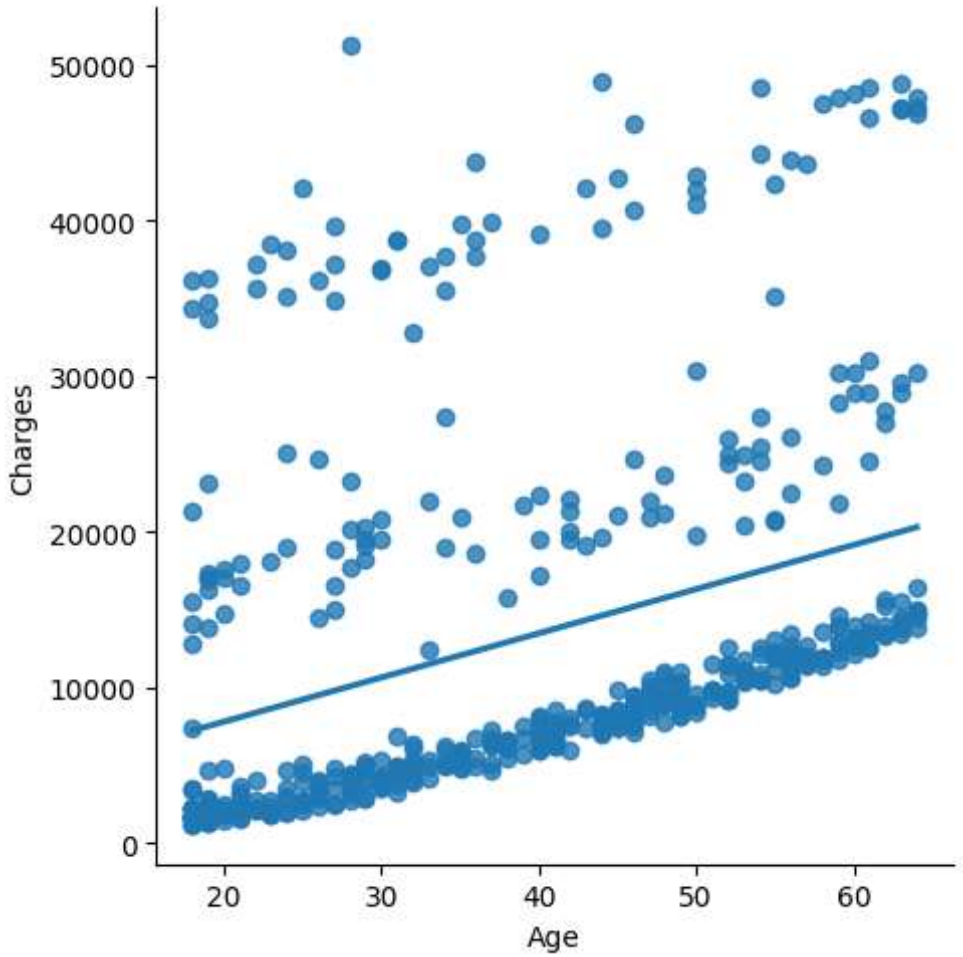


In [14]:

```
df500=df[:][:500]
sns.lmplot(x='Age',y='Charges',data=df500,order=1,ci=None)
```

Out[14]:

```
<seaborn.axisgrid.FacetGrid at 0x16ec827b2b0>
```



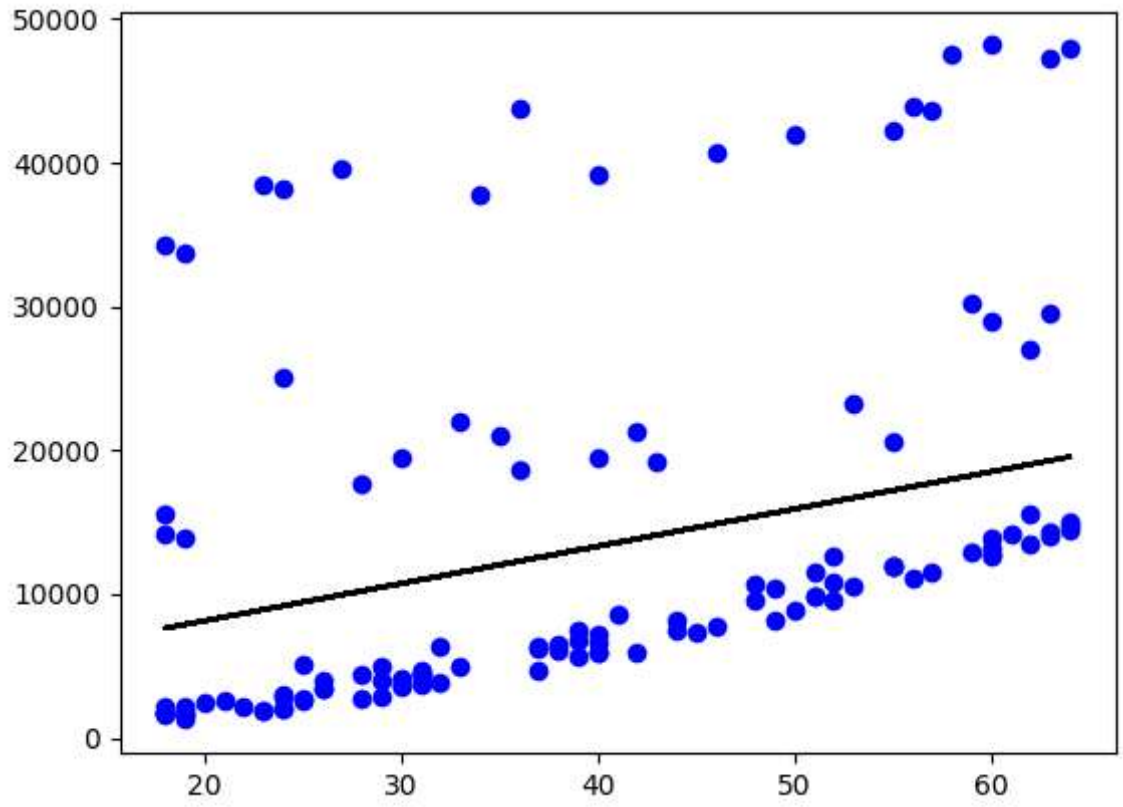
In [15]:

```
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)
reg=LinearRegression()
reg.fit(X_train,y_train)
print("Regression:",reg.score(X_test,y_test))
y_pred=reg.predict(X_test)
plt.scatter(X_test,y_test,color='b')
plt.plot(X_test,y_pred,color='k')
plt.show
```

```
Regression: 0.1525537912566497
```

Out[15]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [16]:

```
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:  0.1525537912566497
```

In [17]:

```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,y_pred))
print('MSE:',metrics.mean_squared_error(y_test,y_pred))
print('RMSE:',np.sqrt(metrics.mean_absolute_error(y_test,y_pred)))
```

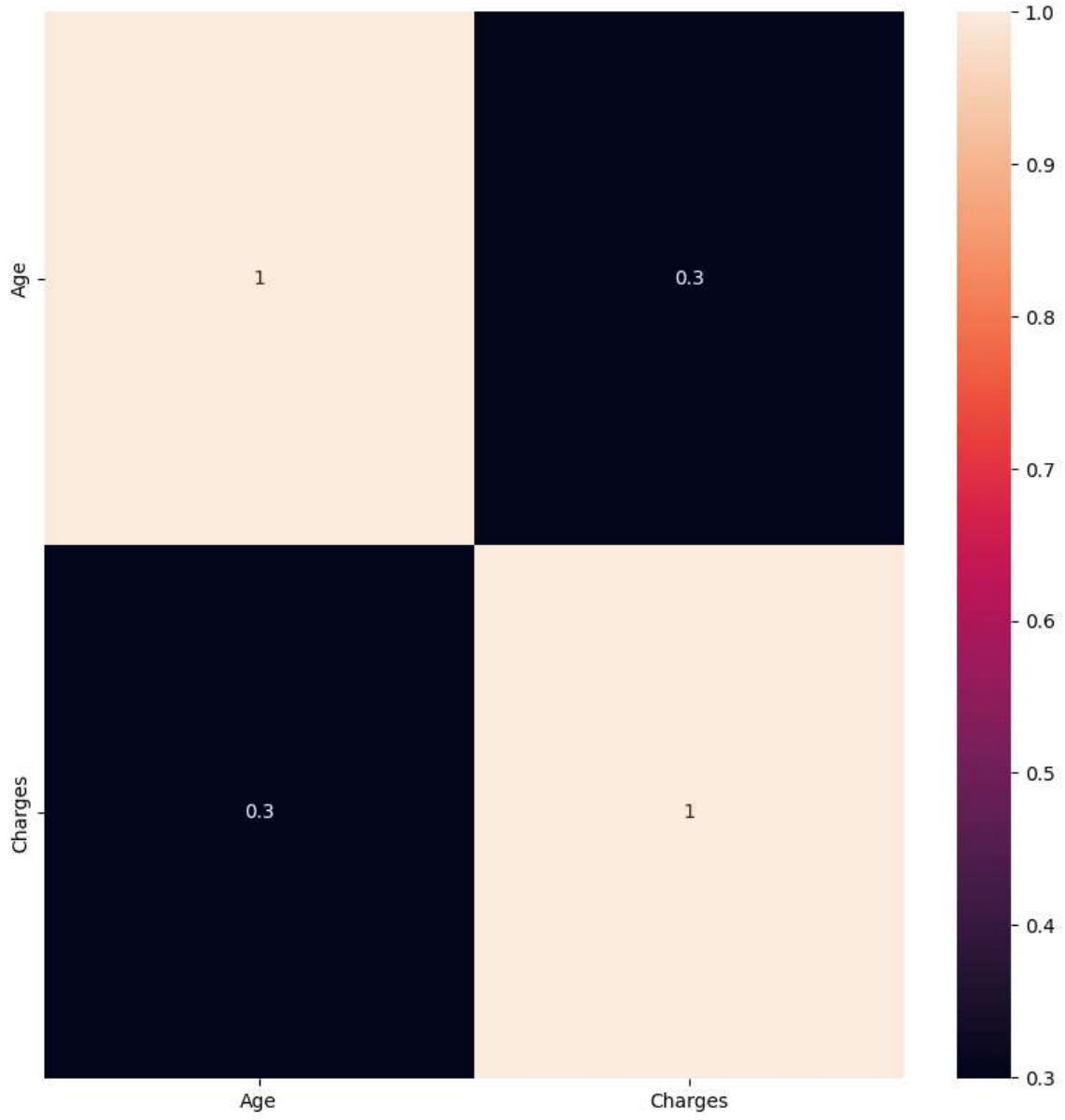
```
MAE: 9346.906572307227
MSE: 142424837.01171198
RMSE: 96.67940097201279
```

In [18]:

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

Out[18]:

```
<Axes: >
```



Ridge Regression

In [19]:

```
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_stat
print("The dimension of X_train is {}".format(X_train.shape))
print("The dimension of X_test is {}".format(X_test.shape))
#Scale features
from sklearn.preprocessing import StandardScaler
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 [20]:

```
#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:")
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 [21]:

```
#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_trair
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_test))
```

```
The train score for ridge model is 0.9999999999999869
The train score for ridge model is 0.9999999999999875
```

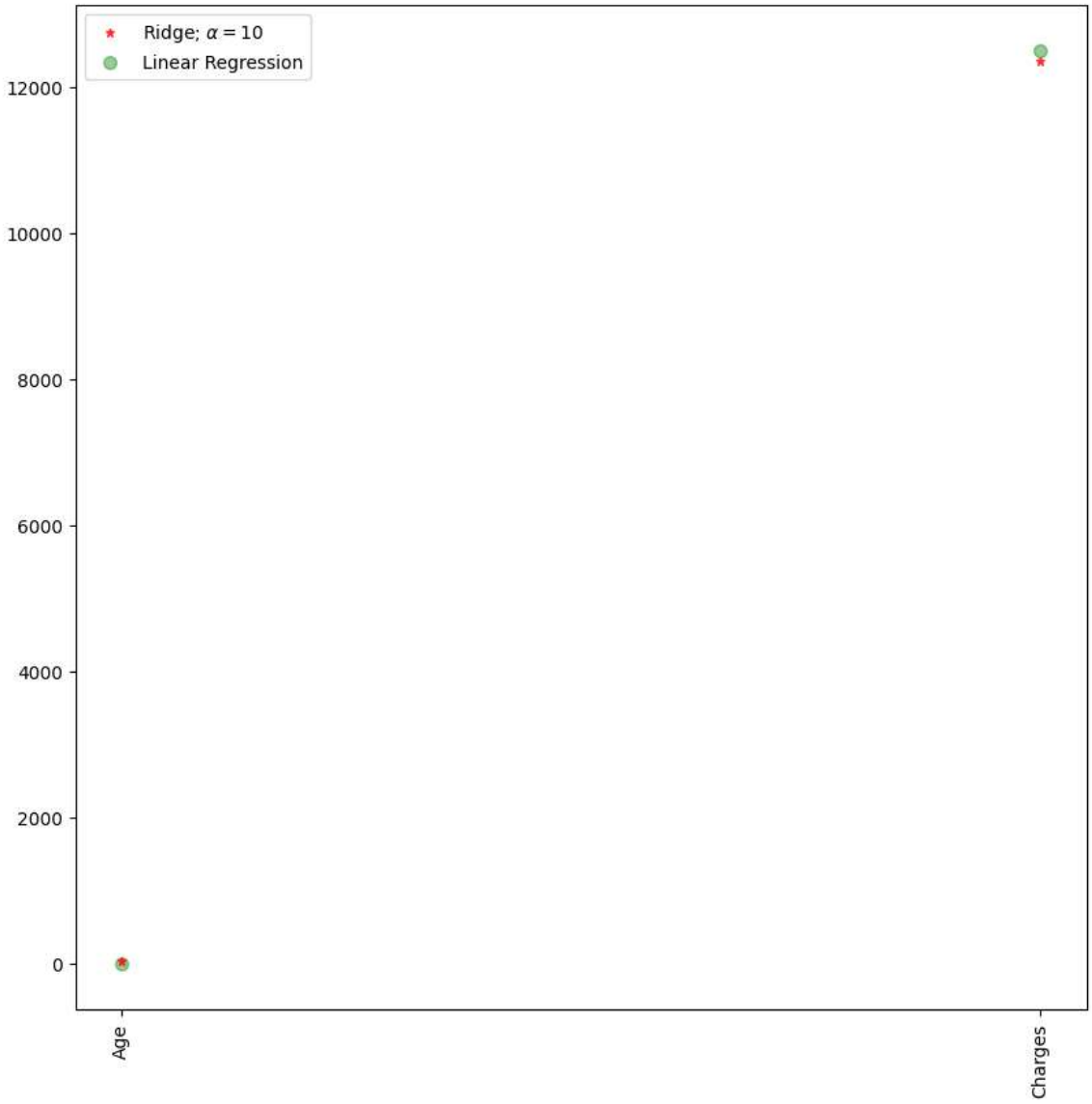
In [22]:

```
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:")
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 0.9998796730521575
The test score for ridge model is 0.9998841048873327
```

In [23]:

```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,

plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color=
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



Lasso Regression

In [24]:

```
#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_
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.9999999999996456
0.999999999999531
```

In [25]:

```
#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))
```

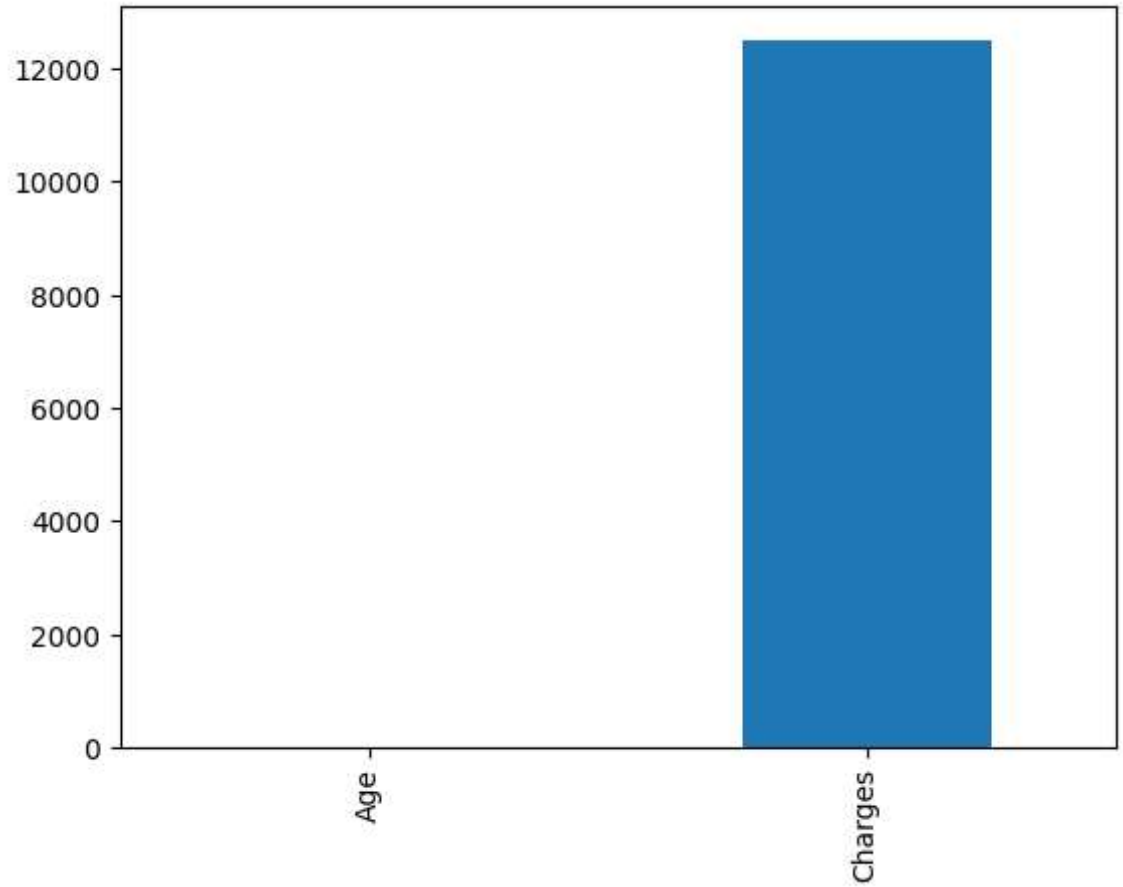
```
Lasso Model:

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

In [26]:

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

Out[26]:

<Axes: >

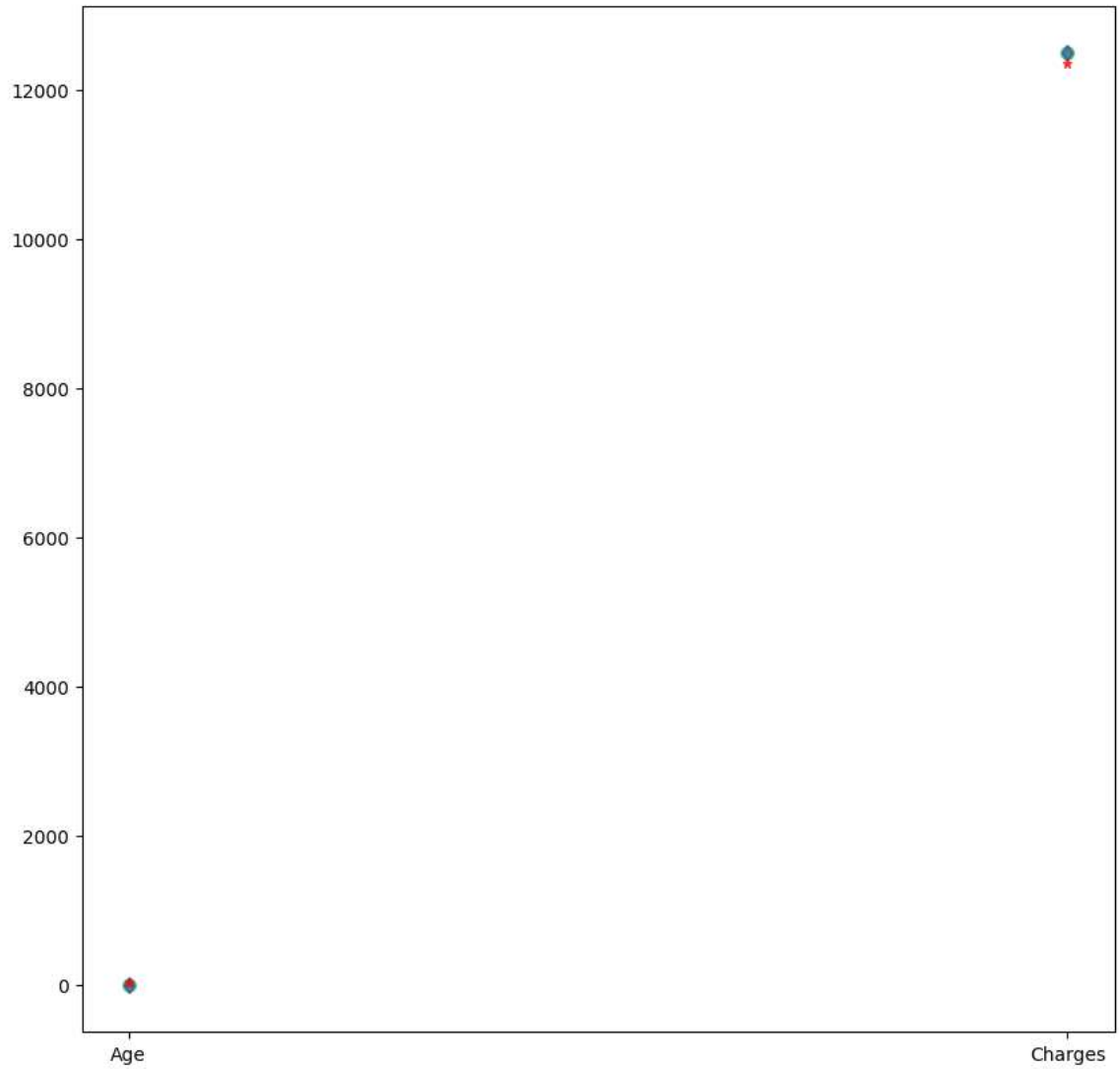


In [27]:

```
#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,
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='bl
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color=
```

Out[27]:

```
[<matplotlib.lines.Line2D at 0x16ec8fdd990>]
```



Elastic Regression

In [28]:

```
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 [29]:

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

Logistic Regression

In [30]:

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid",color_codes=True)
import warnings
warnings.simplefilter(action='ignore')
```

In [31]:

```
db=pd.read_csv(r"C:\Users\91955\Downloads\insurance.csv")
db
```

Out[31]:

	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 [32]:

```
db.head()
```

Out[32]:

	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 [33]:

```
1 db.tail()
```

Out[33]:

	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 [34]:

```
1 db.shape
```

Out[34]:

```
(1338, 7)
```

In [35]:

```
1 db.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 [36]:

```
1 db.describe()
```

Out[36]:

	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 [37]:

```
1 db.isnull().sum()
```

Out[37]:

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

In [38]:

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

Out[38]:

	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 [39]:

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

Out[39]:

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

1338 rows × 7 columns

In [40]:

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

Out[40]:

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

1338 rows × 7 columns

In [41]:

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

In [42]:

```
features_matrix=db.iloc[:,0:6]
```

In [43]:

```
target_vector=db.iloc[:,-2]
```

In [44]:

```
print("The features matrix has %d rows and %d columns"%(features_matrix.shape))
print("The target matrix has %d rows and %d columns"%(np.array(target_vector).reshap
```

```
The features matrix has 1338 rows and 6 columns
The target matrix has 1338 rows and 1 columns
```

In [45]:

```
x=np.array(db['sex']).reshape(-1,1)
y=np.array(db['smoker']).reshape(-1,1)
```

In [46]:

```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [47]:

```
algorithm=LogisticRegression(max_iter=10000)
```

In [48]:

```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [49]:

```
Observation=[[1,0,0.99539,-0.085889,0.8524299999999999,0.02306]]
```

In [50]:

```
predictions=Logistic_Regression_Model.predict(Observation)
print("The model predicted the observation to belonging to class %s"%(predictions))
```

```
The model predicted the observation to belonging to class [2]
```

In [51]:

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

```
The algorithm was trained to predict one of the two classes:[1 2 3 4]
```

In [52]:

```
print("""The model says the probability of the observation we passed belonging to cl
print("""The model says the probability of the observation we passed belonging to cl
print("""The model says the probability of the observation we passed belonging to cl
print("""The model says the probability of the observation we passed belonging to cl
```

```
The model says the probability of the observation we passed belonging to c
lass['0'] is 0.0006646896816406193
The model says the probability of the observation we passed belonging to c
lass['1'] is 0.5652740537171539
The model says the probability of the observation we passed belonging to c
lass['2'] is 0.43340043770162473
The model says the probability of the observation we passed belonging to c
lass['3'] is 0.0006790188995806598
```

In [53]:

```
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))
```

```
0.835820895522388
```

In [54]:

```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(X_test,y_test))
print('MSE:',metrics.mean_squared_error(X_test,y_test))
print('RMSE:',np.sqrt(metrics.mean_absolute_error(X_test,y_test)))
```

```
MAE: 0.44776119402985076
MSE: 0.44776119402985076
RMSE: 0.6691496051182058
```

Decision Tree

In [55]:

```
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 [56]:

```
dt=pd.read_csv(r"C:\Users\91955\Downloads\insurance.csv")
dt
```

Out[56]:

	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

In [57]:

```
1 dt.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 [58]:

```
1 dt['smoker'].value_counts()
```

Out[58]:

```
smoker
no     1064
yes     274
Name: count, dtype: int64
```

In [59]:

```
1 dt['children'].value_counts()
```

Out[59]:

```
children
0    574
1    324
2    240
3    157
4     25
5     18
Name: count, dtype: int64
```

In [60]:

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

Out[60]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	yes	southwest	16884.924000
1	18	2	33.770	1	no	southeast	1725.552300
2	28	2	33.000	3	no	southeast	4449.462000
3	33	2	22.705	0	no	northwest	21984.470610
4	32	2	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	2	29.830	2	no	northeast	6406.410700
9	60	1	25.840	0	no	northwest	28923.136920

In [61]:

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

Out[61]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	yes	1	16884.924000
1	18	2	33.770	1	no	2	1725.552300
2	28	2	33.000	3	no	2	4449.462000
3	33	2	22.705	0	no	3	21984.470610
4	32	2	28.880	0	no	3	3866.855200
5	31	1	25.740	0	no	2	3756.621600
6	46	1	33.440	1	no	2	8240.589600
7	37	1	27.740	3	no	3	7281.505600
8	37	2	29.830	2	no	4	6406.410700
9	60	1	25.840	0	no	3	28923.136920

In [62]:

```
x=["sex","children","region"]
y=["yes","no"]
all_inputs=dt[x]
all_classes=dt["smoker"]
```

In [63]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.
```

In [64]:

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

In [65]:

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

Out[65]:

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

In [66]:

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

```
0.817910447761194
```

Random Forest

In [67]:

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

In [68]:

```
ds=pd.read_csv(r"C:\Users\91955\Downloads\insurance.csv")
ds
```

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

In [69]:

```
ds.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 [70]:

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

In [71]:

```
ds['smoker'].value_counts()
```

Out[71]:

```
smoker
no     1064
yes     274
Name: count, dtype: int64
```

In [72]:

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

10	25	2	26.220	0	no	northeast	2721.320800
11	62	1	26.290	0	yes	southeast	27808.725100
12	23	2	34.400	0	no	southwest	1826.843000
13	56	1	39.820	0	no	southeast	11090.717800
14	27	2	42.130	0	yes	southeast	39611.757700
15	19	2	24.600	1	no	southwest	1837.237000
16	52	1	30.780	1	no	northeast	10797.336200
17	23	2	23.845	0	no	northeast	2395.171550
18	56	2	40.300	0	no	southwest	10602.385000
19	30	2	35.300	0	yes	southwest	36837.467000
20	60	1	36.005	0	no	northeast	13228.846950
21	30	1	32.400	1	no	southwest	4149.736000
22	18	2	34.100	0	no	southeast	1137.011000

In [73]:

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

Out[73]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	yes	1	16884.924000
1	18	2	33.770	1	no	2	1725.552300
2	28	2	33.000	3	no	2	4449.462000
3	33	2	22.705	0	no	3	21984.470610
4	32	2	28.880	0	no	3	3866.855200
5	31	1	25.740	0	no	2	3756.621600
6	46	1	33.440	1	no	2	8240.589600
7	37	1	27.740	3	no	3	7281.505600
8	37	2	29.830	2	no	4	6406.410700
9	60	1	25.840	0	no	3	28923.136920

In [74]:

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

Out[74]:

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

In [75]:

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

In [76]:

```
x
```

Out[76]:

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

In [77]:

```
y
```

Out[77]:

```
0       1
1       2
2       2
3       2
4       2
5       2
6       2
7       2
8       2
9       2
10      2
11      1
12      2
13      2
14      1
15      2
16      2
17      2
```

In [78]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_state=42)
x_train.shape,x_test.shape
```

Out[78]:

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

In [79]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()

```

In [80]:

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

Out[80]:

```
RandomForestClassifier
RandomForestClassifier()
```

In [81]:

```
rf=RandomForestClassifier()
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 [82]:

```
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[82]:

GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier

In [83]:

```
grid_search.best_score_
```

Out[83]:

```
0.9508547008547008
```

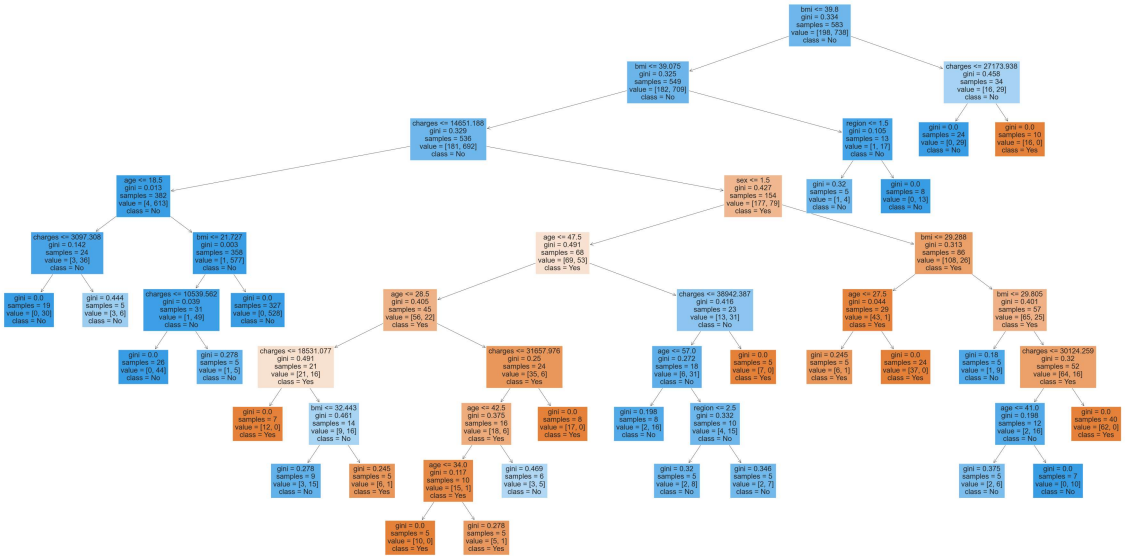
In [84]:

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

```
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=25)
```

In [85]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],feature_names=x.columns,class_names=["Yes","No"],fi
```



In [86]:

```
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"],fi
```

```
= 62\nvalue = [68, 31]\nclass = Yes'),
 Text(0.6666666666666666, 0.4375, 'age <= 26.0\ngini = 0.26\nsamples =
33\nvalue = [44, 8]\nclass = Yes'),
 Text(0.6296296296296297, 0.3125, 'gini = 0.444\nsamples = 5\nvalue =
[2, 4]\nclass = No'),
 Text(0.7037037037037037, 0.3125, 'age <= 49.5\ngini = 0.159\nsamples =
28\nvalue = [42, 4]\nclass = Yes'),
 Text(0.6666666666666666, 0.1875, 'age <= 32.0\ngini = 0.056\nsamples =
20\nvalue = [34, 1]\nclass = Yes'),
 Text(0.6296296296296297, 0.0625, 'gini = 0.245\nsamples = 5\nvalue =
[6, 1]\nclass = Yes'),
 Text(0.7037037037037037, 0.0625, 'gini = 0.0\nsamples = 15\nvalue = [2
8, 0]\nclass = Yes'),
 Text(0.7407407407407407, 0.1875, 'gini = 0.397\nsamples = 8\nvalue =
[8, 3]\nclass = Yes'),
 Text(0.8888888888888888, 0.4375, 'bmi <= 39.225\ngini = 0.5\nsamples =
29\nvalue = [24, 23]\nclass = Yes'),
 Text(0.8518518518518519, 0.3125, 'charges <= 33094.077\ngini = 0.488\n
samples = 24\nvalue = [16, 22]\nclass = No'),
 Text(0.8148148148148148, 0.1875, 'gini = 0.0\nsamples = 12\nvalue =
```

Conclusion:For the given dataset,we have performed linear regression,ridge regression,lasso regression,elastic regression,logistic regression,decision tree,random forest classification.Among all the models,we observed that ,in random forest the accuracy is 0.94,and in the lasso regression we observed,the accuracy is 0.99 where as lasso regression got the highest accuracy than the random forest.So, the best model that suits for the given dataset is linear regession and lasso regression.

In [ ]:

```
1
```