Problem Statement :To predict the given dataset is best fit or not

Linear Regression

1.Data Collection :

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]:
```
dt=pd.read_csv(r"C:\Users\91903\Downloads\insurance.csv")
dt
```

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

1338 rows × 7 columns

2.Data Cleaning and Preprocessing :

In [3]:
```
dt.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]:
```
dt.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]:
```
dt.describe()
```

Out[5]:

	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 [6]:
```
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 [7]:
```
dt.isnull().sum()
```

Out[7]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [8]:
```
dt=dt[['age','charges']]
dt.columns=['Age','Charge']
```

In [9]:
```
dt.head()
```

Out[9]:

	Age	Charge
0	19	16884.92400
1	18	1725.55230
2	28	4449.46200
3	33	21984.47061
4	32	3866.85520

In [10]:
```
dt.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   Charge  1338 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 21.0 KB
```

In [11]:
```
dt.isna().any()
```

Out[11]:
```
Age       False
Charge    False
dtype: bool
```

In [12]:
```
convert={"sex":{"female":1,"male":0}}
dt=dt.replace(convert)
dt
```

Out[12]:

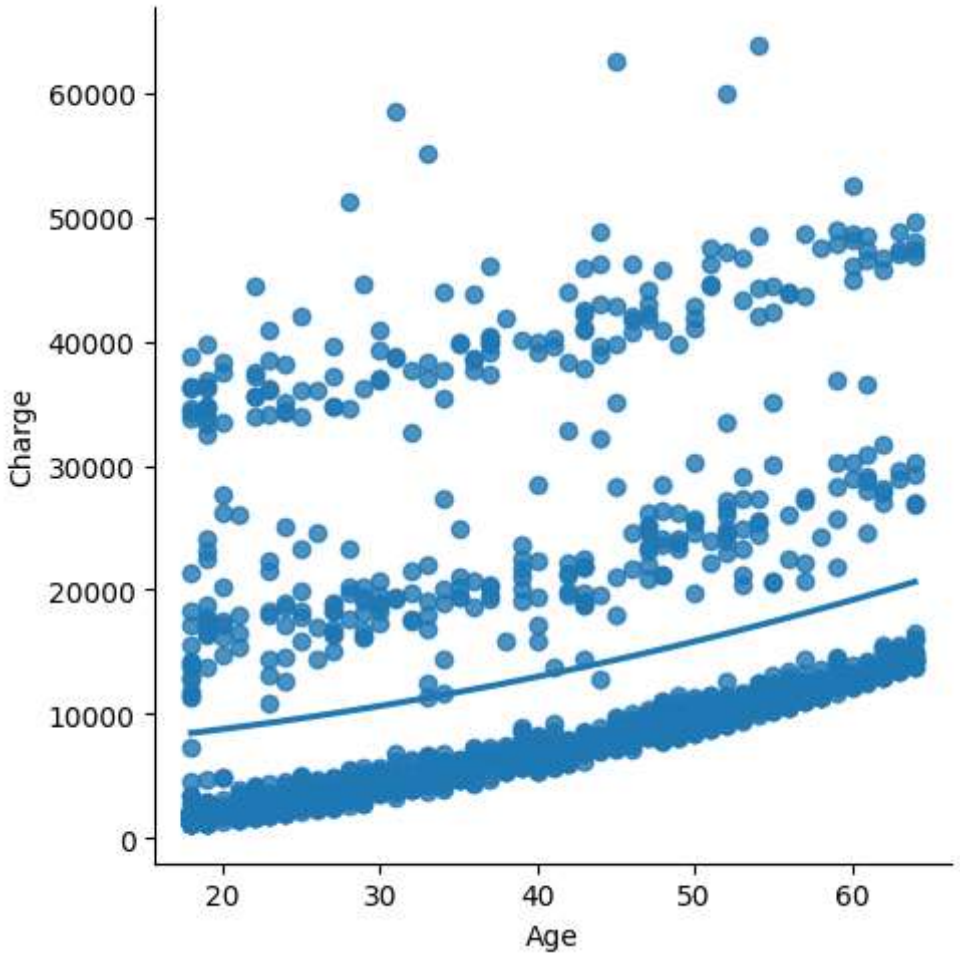
	Age	Charge
0	19	16884.92400
1	18	1725.55230
2	28	4449.46200
3	33	21984.47061
4	32	3866.85520
...	...	...
1333	50	10600.54830
1334	18	2205.98080
1335	18	1629.83350
1336	21	2007.94500
1337	61	29141.36030

1338 rows × 2 columns

3. Data Visualization :

In [13]:
```
sns.lmplot(x='Age',y='Charge',data=dt,order=2,ci=None)
```

Out[13]:
```
<seaborn.axisgrid.FacetGrid at 0x292e7b58b20>
```



In [14]:
```
dt.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   Charge  1338 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 21.0 KB
```

In [15]:
```
dt.describe()
```

Out[15]:

	Age	Charge
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 [16]:
```
dt.isnull().sum()
```

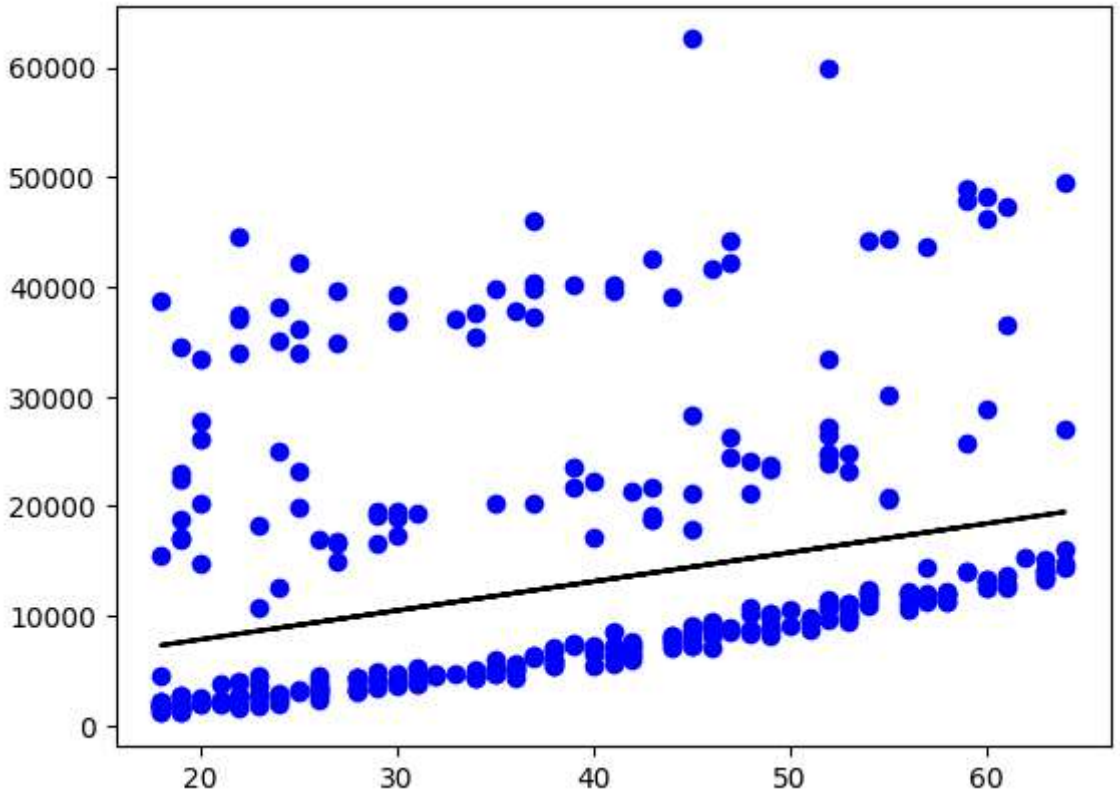
Out[16]:
```
Age       0
Charge    0
dtype: int64
```

In [17]:
```
x=np.array(dt['Age']).reshape(-1,1)
y=np.array(dt['Charge']).reshape(-1,1)
```

In [18]:
```
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))
```

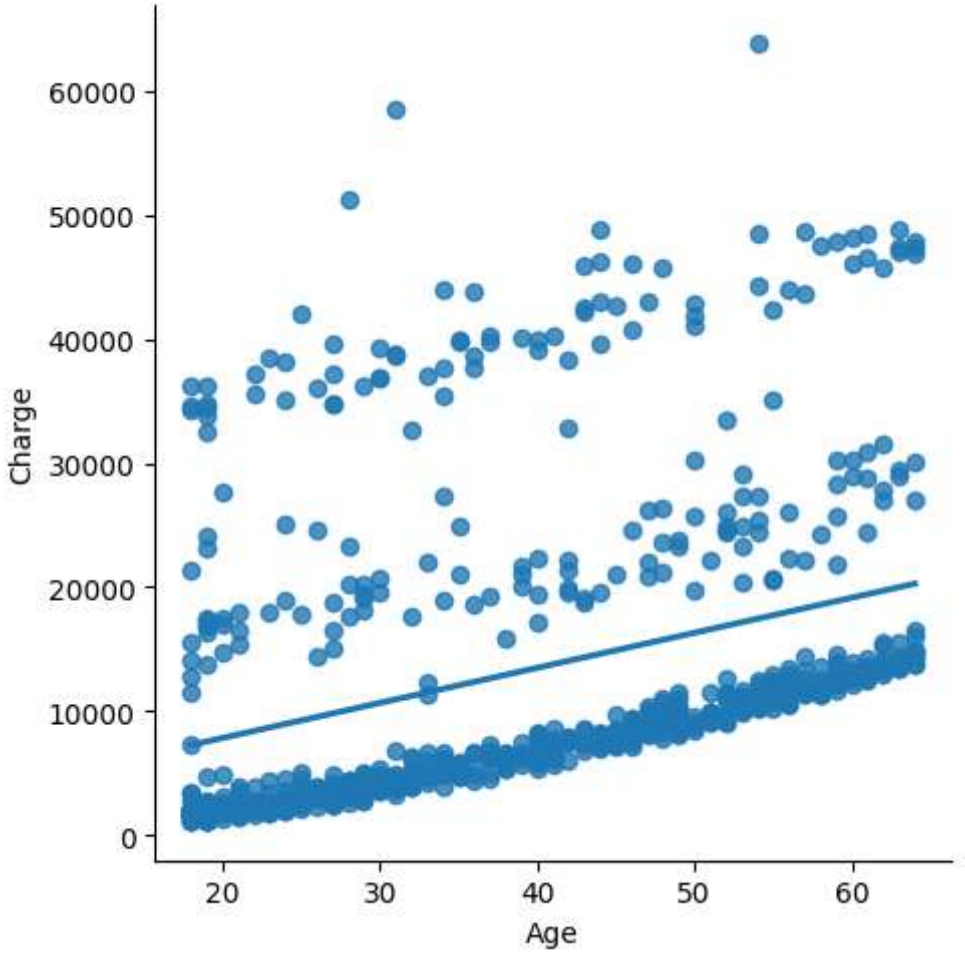
```
0.0530517362341556
```

In [19]:
```
y_pred=reg.predict(X_test)
plt.scatter(X_test,y_test,color='b')
plt.plot(X_test,y_pred,color='k')
plt.show()
```



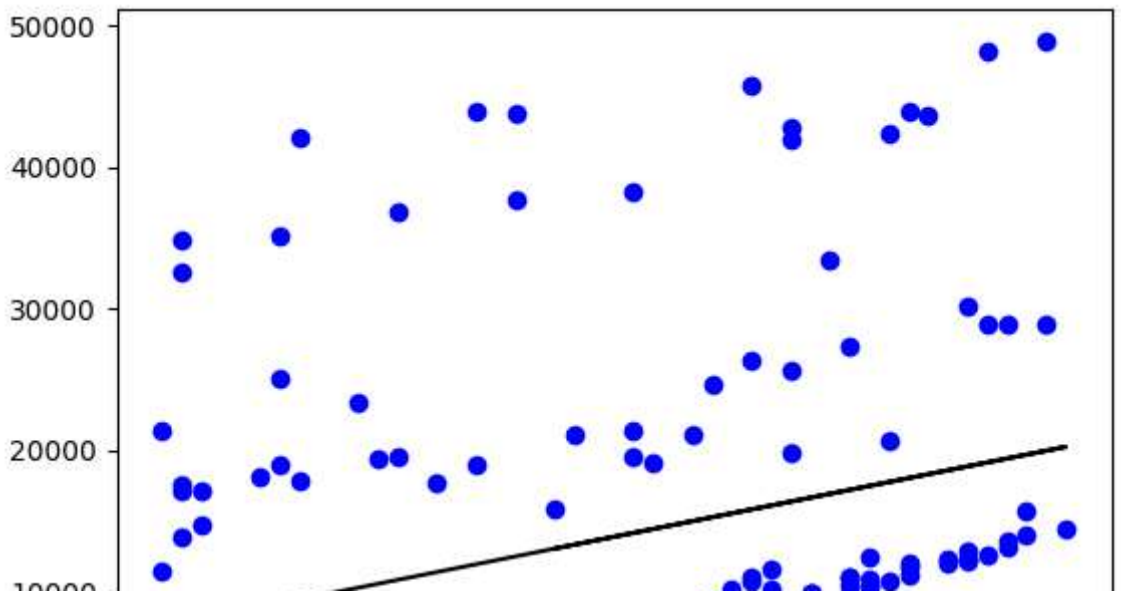
In [20]:
```
dt700=dt[:][:700]
sns.lmplot(x="Age",y="Charge",data=dt700,order=1,ci=None)
```

Out[20]:
```
<seaborn.axisgrid.FacetGrid at 0x292d57b2bc0>
```



In [21]:
```
dt700.fillna(method='ffill',inplace=True)
X=np.array(dt700['Age']).reshape(-1,1)
y=np.array(dt700['Charge']).reshape(-1,1)
dt700.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: 0.1254965865409955
```



Evaluation of model

In [22]:
```
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: 0.1254965865409955
```

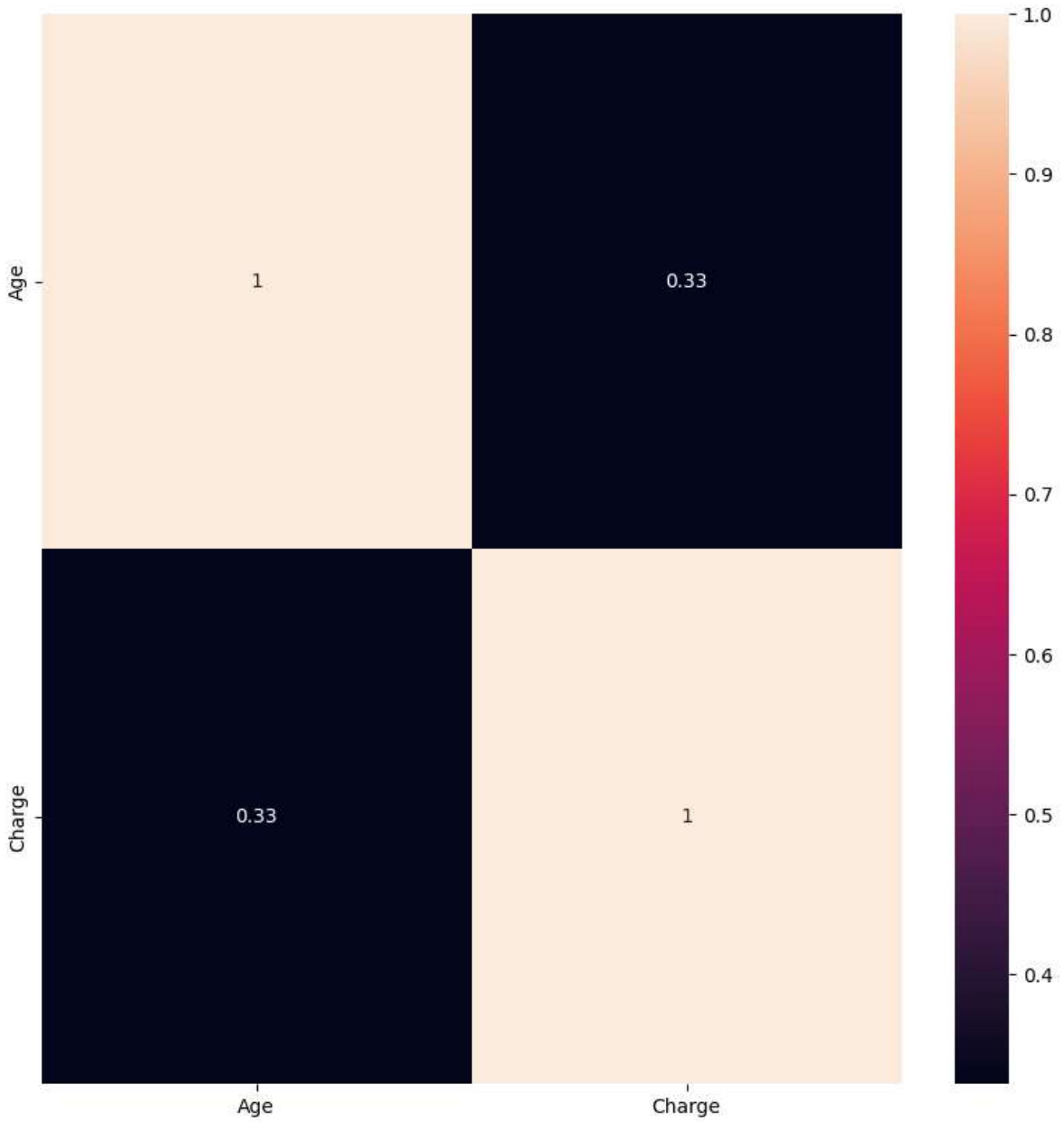
In [23]:
```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,y_pred))
print('MSE:',metrics.mean_squared_error(y_test,y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
MAE: 8815.979229397746
MSE: 118650646.71553652
RMSE: 10892.68776361172
```

Ridge Regression

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

In [25]:
```
plt.figure(figsize=(10,10))
sns.heatmap(dt700.corr(),annot=True)
plt.show()
```



In [26]:
```
features = dt.columns[0:2]
target = dt.columns[-1]
#X and y values
X = dt[features].values
y = dt[target].values
#splot
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, randc
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 [27]:
```
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
```

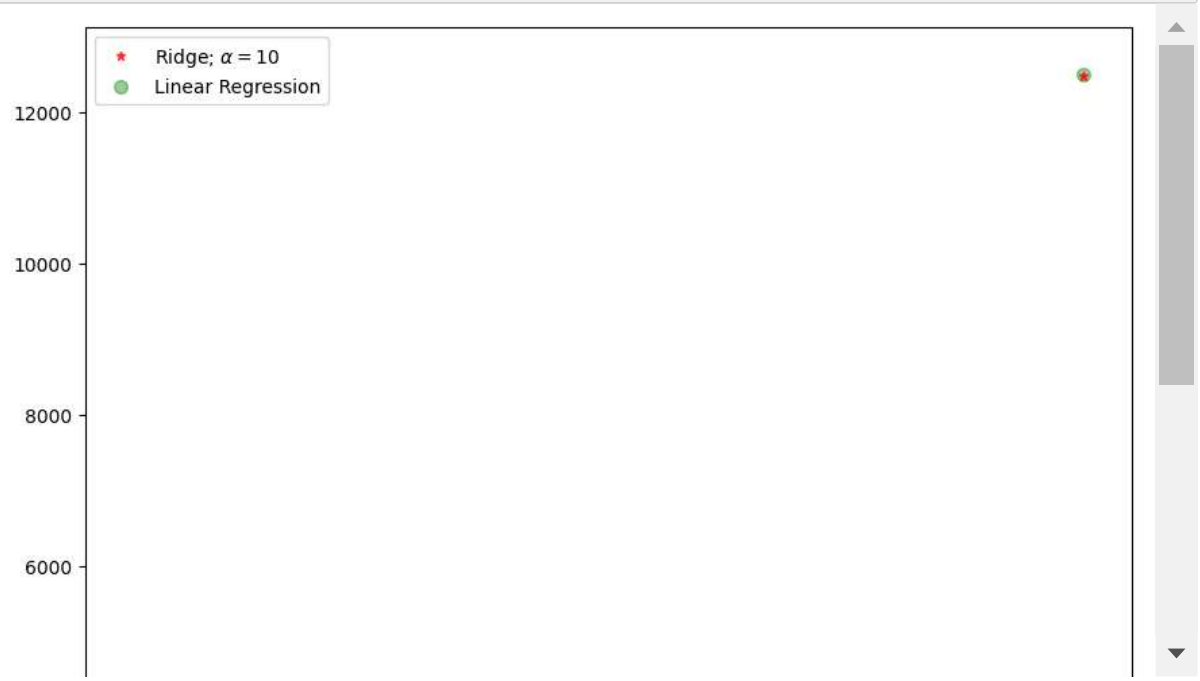
In [28]:
```
ridgeReg = Ridge(alpha=1)
ridgeReg.fit(X_train,y_train)
#train and test scorefor ridge regression
train_score_ridge = ridgeReg.score(X_train, y_train)
test_score_ridge = ridgeReg.score(X_test, y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```
Ridge Model:

The train score for ridge model is 0.9999987697834629
The test score for ridge model is 0.9999988155162083
```

In [29]:
```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers

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



Lasso Regression

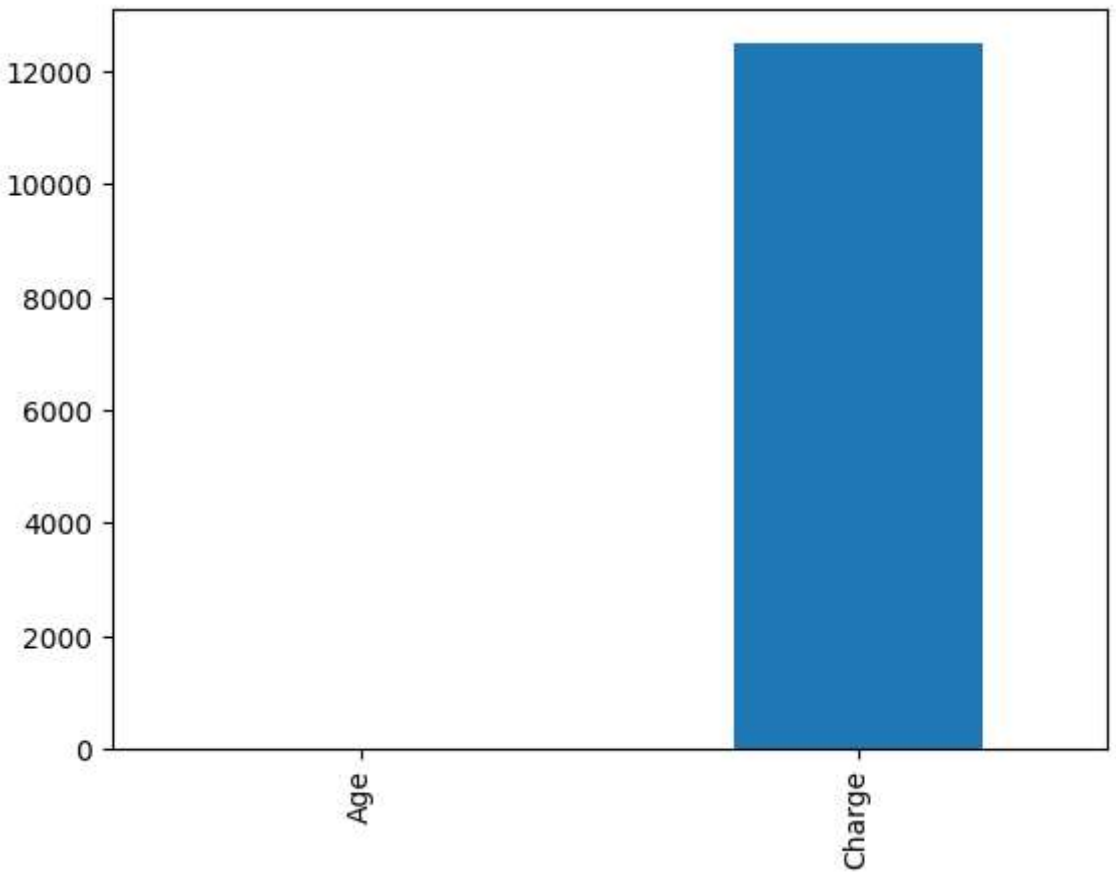
In [30]:
```
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The train score for ls model is {}".format(test_score_ls))
```

```
Lasso Model:

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

In [31]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "ba
```

Out[31]: <Axes: >

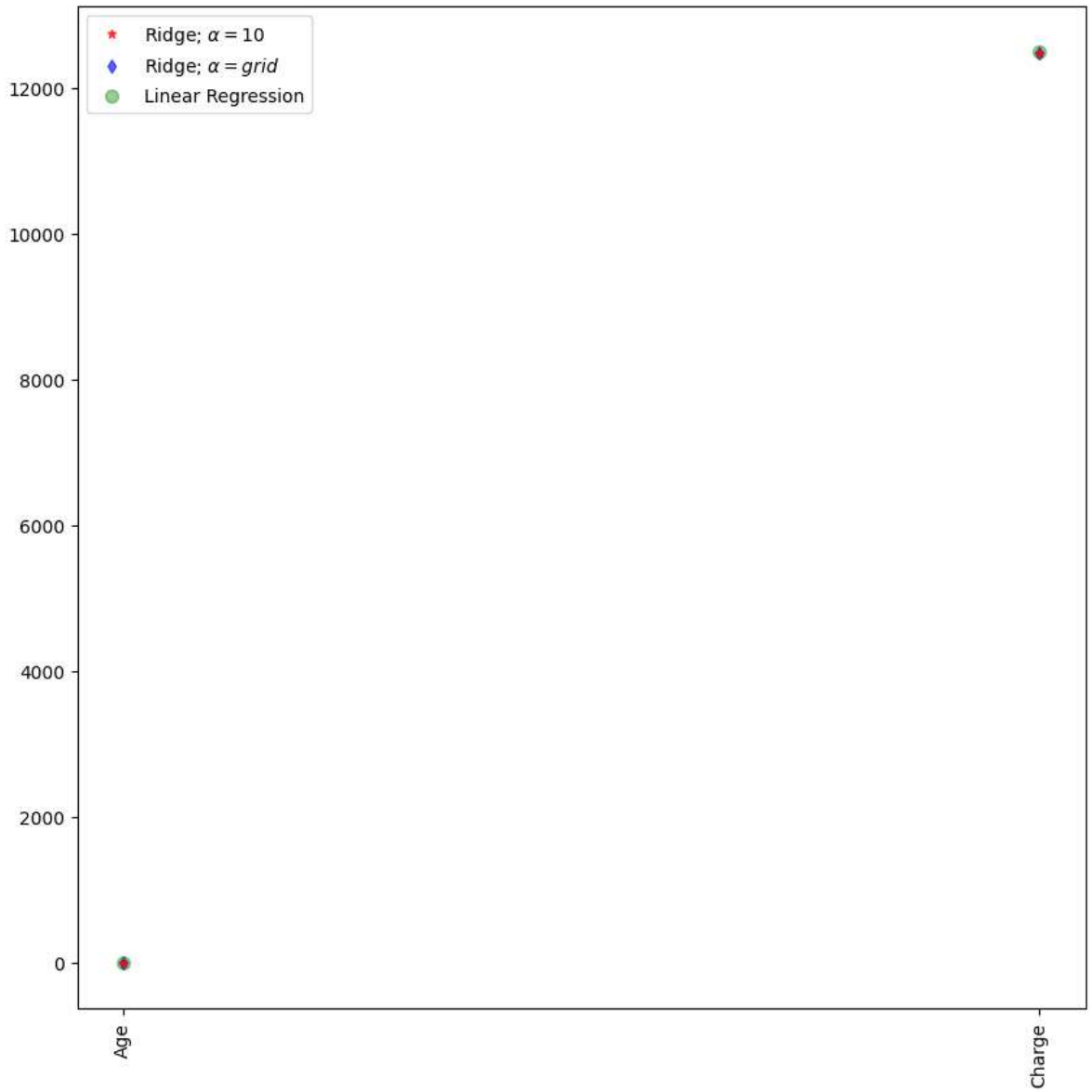


In [32]:
```
#Using the linear CV model
from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.9999999999996456
0.999999999999531
```

In [33]:
```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
plt.plot(features,ridgeReg.coef_,alpha=0.6,linestyle='none',marker='d',markers

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



In [34]:
```
#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_t
#score
print("The train score for ridge model is {}".format(ridge_cv.score(X_train, y
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_
```

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

ElasticNet Regression

In [35]:
```
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.         0.99999999]
9.055664850166067e-05
```

In [36]:
```
y_pred_elastic=regr.predict(X_train)
```

In [37]:
```
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 347174894.85874796
```

Logistic Regression

In [38]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [39]:
```
df=pd.read_csv(r"C:\Users\91903\Downloads\insurance.csv")
df
```

Out[39]:

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

In [41]:
```
print('The DataFrame has %d Rows and %d columns'%(df.shape))
```

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

In [42]:
```
df.head()
```

Out[42]:

	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 [43]:
```
df.tail()
```

Out[43]:

	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 [44]:
```
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 [45]:
```
df.isnull().sum()
```

Out[45]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [46]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

205	28	1	28.880	1	no	northeast	4337.735200
206	59	0	26.400	0	no	southeast	11743.299000
207	35	0	27.740	2	yes	northeast	20984.093600
208	63	1	31.800	0	no	southwest	13880.949000
209	40	0	41.230	1	no	northeast	6610.109700
210	20	0	33.000	1	no	southwest	1980.070000
211	40	0	30.875	4	no	northwest	8162.716250
212	24	0	28.500	2	no	northwest	3537.703000
213	34	1	26.730	1	no	southeast	5002.782700
214	45	1	30.900	2	no	southwest	8520.026000
215	41	1	37.100	2	no	southwest	7371.772000
216	53	1	26.600	0	no	northwest	10355.641000
217	27	0	23.100	0	no	southeast	2483.736000

In [47]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[47]:

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

In [48]:
```
convert={"region":{"southwest":1,"southeast":2,"northwest":3,"northeast":4}}
df=df.replace(convert)
df
```

Out[48]:

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

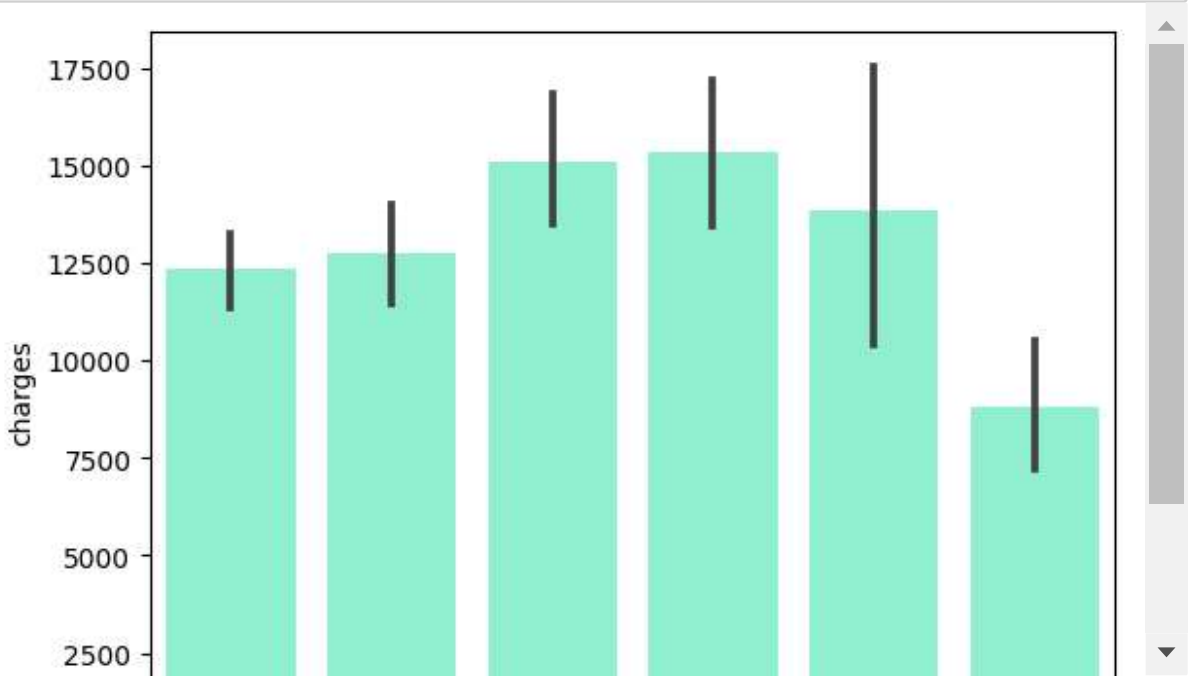
In [49]:
```
features_matrix=df.iloc[:,0:6]
```

In [50]:
```
target_vector=df.iloc[:,-2]
```

In [51]:
```
print('The Feature Matrix Has %d Rows and %d Column(s)'%(features_matrix.shape
print('The Target Matrix Has %d Rows and %d Column(s)'%(np.array(target_vector
```

```
The Feature Matrix Has 1338 Rows and 6 Column(s)
The Target Matrix Has 1338 Rows and 1 Column(s)
```

In [52]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.barplot(x='children',y='charges',data=df,color='aquamarine')
plt.show()
```



In [53]:
```
x=np.array(df['sex']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
```

In [54]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [55]:
```
algorithm=LogisticRegression(max_iter=10000)
```

In [56]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_ve
```

In [57]:
```
observation=[[1,0,0.99539,-0.085889,0.8524299999999999,0.02306]]
```

In [58]:
```
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 [59]:
```
print('The algorithm was Trained to predict one of the Two Classes %s'%(algori
```

```
The algorithm was Trained to predict one of the Two Classes [1 2 3 4]
```

In [60]:
```
print(""" The Model says The probabilty of the observation we passed Belonging
print()
print(""" The Model says The probabilty of the observation we passed Belonging
print()
print(""" The Model says The probabilty of the observation we passed Belonging
print()
print(""" The Model says The probabilty of the observation we passed Belonging
```

```
 The Model says The probabilty of the observation we passed Belonging to clas
s['1'] Is 0.0005958269000088093

 The Model says The probabilty of the observation we passed Belonging to clas
s['2'] Is 0.6080148927600352

 The Model says The probabilty of the observation we passed Belonging to clas
s['3'] Is 0.39073187301769985

 The Model says The probabilty of the observation we passed Belonging to clas
s['4'] Is 0.0006574073222560058
```

In [61]:
```
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))
```

```
0.7910447761194029

C:\Users\91903\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\utils\validation.py:1143: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [62]:
```
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_squared_error(X_test,y_test)))
```

```
MAE: 0.6417910447761194
MSE: 0.6417910447761194
RMSE: 0.8011186209146055
```

Decision Tree

In [63]:
```
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 [64]:
```
df=pd.read_csv(r"C:\Users\91903\Downloads\insurance.csv")
df
```

Out[64]:

	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

In [65]:
```
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 [66]:
```
df['smoker'].value_counts()
```

Out[66]:
```
smoker
no     1064
yes     274
Name: count, dtype: int64
```

In [67]:
```
df['children'].value_counts()
```

Out[67]:
```
children
0    574
1    324
2    240
3    157
4     25
5     18
Name: count, dtype: int64
```

In [68]:
```
convert={"sex":{"female":1,"male":2}}
df=df.replace(convert)
df
```

Out[68]:

	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
10	25	2	26.220	0	no	northeast	2721.320800

In [69]:
```
convert={"region":{"southwest":1,"southeast":2,"northwest":3,"northeast":4}}
df=df.replace(convert)
df
```

19	30	2	35.300	0	yes	1	36837.467000
20	60	1	36.005	0	no	4	13228.846950
21	30	1	32.400	1	no	1	4149.736000
22	18	2	34.100	0	no	2	1137.011000
23	34	1	31.920	1	yes	4	37701.876800
24	37	2	28.025	2	no	3	6203.901750
25	59	1	27.720	3	no	2	14001.133800
26	63	1	23.085	0	no	4	14451.835150
27	55	1	32.775	2	no	3	12268.632250
28	23	2	17.385	1	no	3	2775.192150
29	31	2	36.300	2	yes	1	38711.000000
30	22	2	35.600	0	yes	1	35585.576000
31	18	1	26.315	0	no	4	2198.189850

In [70]:
```
x=["sex","children","region"]
y=["yes","no"]
all_inputs=df[x]
all_classes=df["smoker"]
```

In [71]:
```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_s
```

In [72]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [73]:
```
clf.fit(x_train,y_train)
```

Out[73]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [74]:
```
score=clf.score(x_test,y_test)
print(score)
```

```
0.8149253731343283
```

Random Forest

In [75]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [76]:
```
df=pd.read_csv(r"C:\Users\91903\Downloads\insurance.csv")
df
```

Out[76]:

	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

In [77]:
```
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 [78]:
```
x=df.drop('charges',axis=1)
y=df['charges']
```

In [79]:
```
df['smoker'].value_counts()
```

Out[79]:
```
smoker
no     1064
yes     274
Name: count, dtype: int64
```

In [80]:
```
convert={"sex":{"female":1,"male":2}}
df=df.replace(convert)
df
```

Out[80]:

	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
10	25	2	26.220	0	no	northeast	2721.320800

In [81]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[81]:

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

In [82]:
```
convert={"region":{"southwest":1,"southeast":2,"northwest":3,"northeast":4}}
df=df.replace(convert)
df
```

Out[82]:

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

In [83]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[83]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [84]:
```
rf=RandomForestClassifier()
params={'max_depth':[2,3,5,20],
 'min_samples_leaf':[5,10,20,50,100,200],
 'n_estimators':[10,25,30,50,100,200]}
```

In [85]:
```
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[85]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

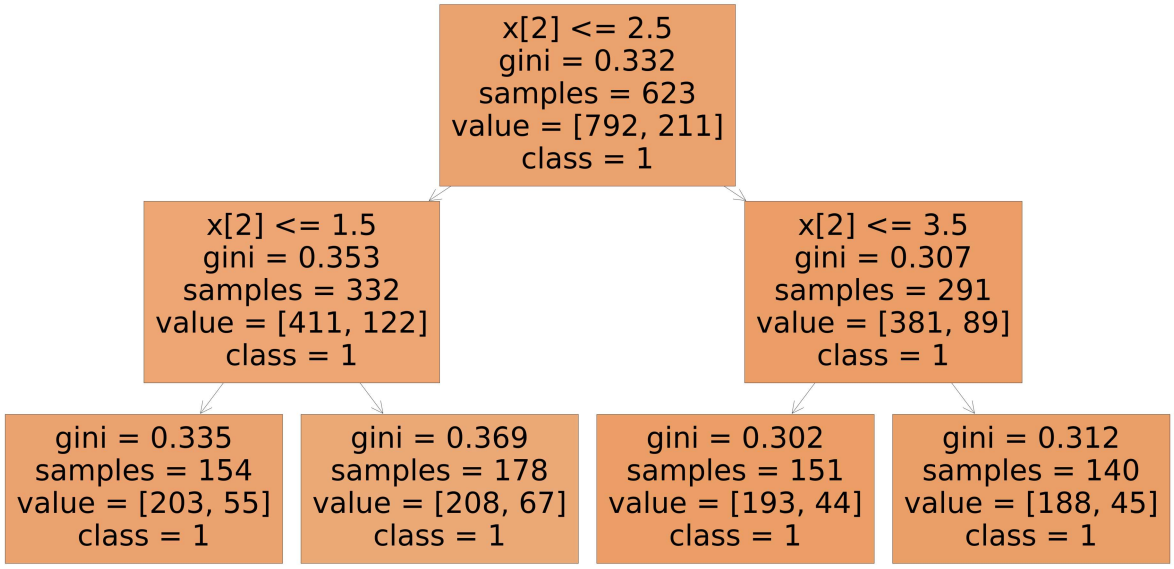
In [86]:
```
grid_search.best_score_
```

Out[86]:
```
0.7886338876032795
```

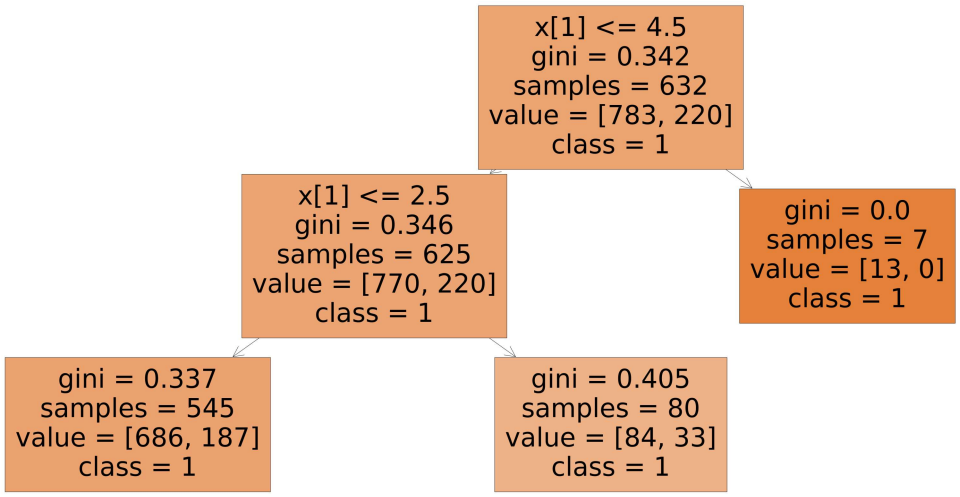
In [87]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

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

In [88]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['1','0'],filled=True);
```



In [89]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(70,30))
plot_tree(rf_best.estimators_[6],class_names=["1","0"],filled=True);
```



In [90]:
```
rf_best.feature_importances_
```

Out[90]:
```
array([0.26212434, 0.46466037, 0.27321529])
```

In [91]:
```
rf=RandomForestClassifier(random_state=0)
```

In [92]:
```
rf.fit(x_train,y_train)
```

Out[92]:
```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

In [93]:
```
score=rf.score(x_test,y_test)
print(score)
```

```
0.8149253731343283
```

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 0.81,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 [ ]: