problem statement: To Predict How The Best Data Fits

1)Data Collection

In [31]:
```
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 Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
```

In [32]:
```
df=pd.read_csv(r"C:\Users\jangidi veena\OneDrive\Documents\jupyter\insurance.csv")
df
```

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
...	...	...	...	...	...	...	...
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 Data Preprocessing

In [33]:
```
df.head()
```

Out[33]:

	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 [34]:
```
df.tail()
```

Out[34]:

	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 [35]:
```
df.describe()
```

Out[35]:

	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 [36]:
```
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 [37]:
```
df.shape
```

Out[37]:
```
(1338, 7)
```

In [38]:
```
df.isnull().any()
```

Out[38]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [39]:
```
df.isnull().sum()
```

Out[39]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [40]:
```
df['region'].value_counts()
```

Out[40]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

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

Out[93]:

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

1338 rows × 7 columns

In [94]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[94]:

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

1338 rows × 7 columns

In [95]:
```
convert={"region":{"southeast":1,"southwest":2,"northeast":3,"northwest":4}}
df=df.replace(convert)
df
```

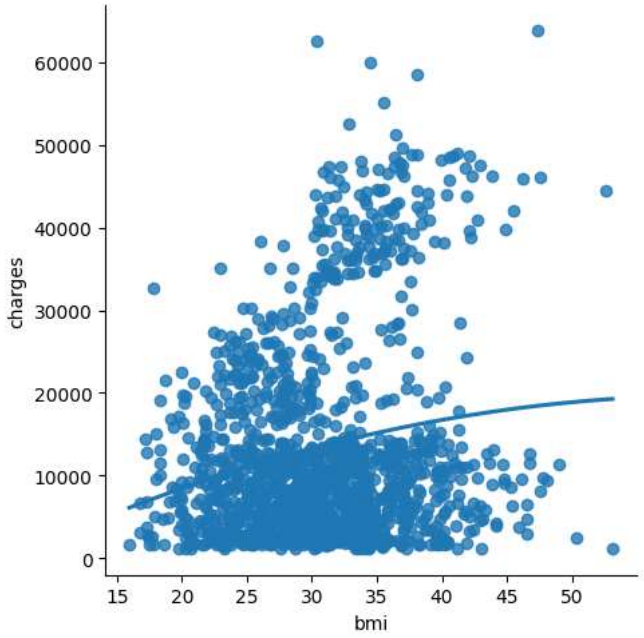
Out[95]:

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

1338 rows × 7 columns

3)Data Visualization

In [96]:
```
sns.lmplot(x='bmi',y='charges',order=2,data=df,ci=None)
plt.show()
```

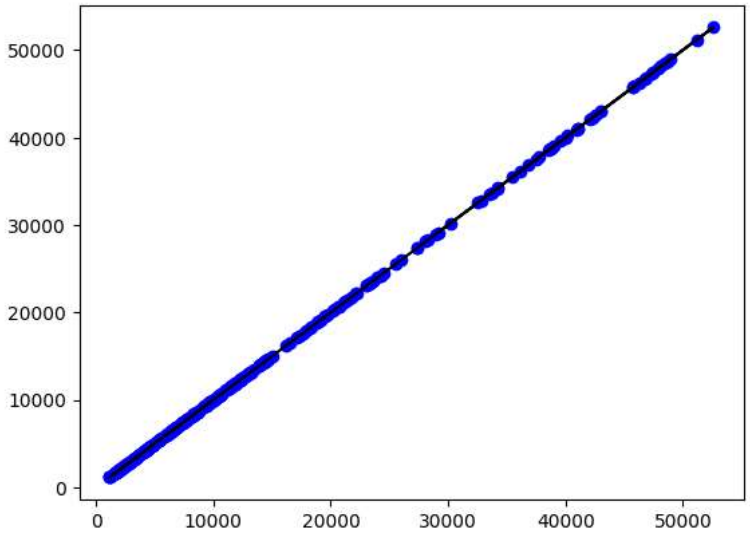


In [97]:
```
x=np.array(df['bmi']).reshape(-1,1)
y=x=np.array(df['charges']).reshape(-1,1)
```

In [98]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

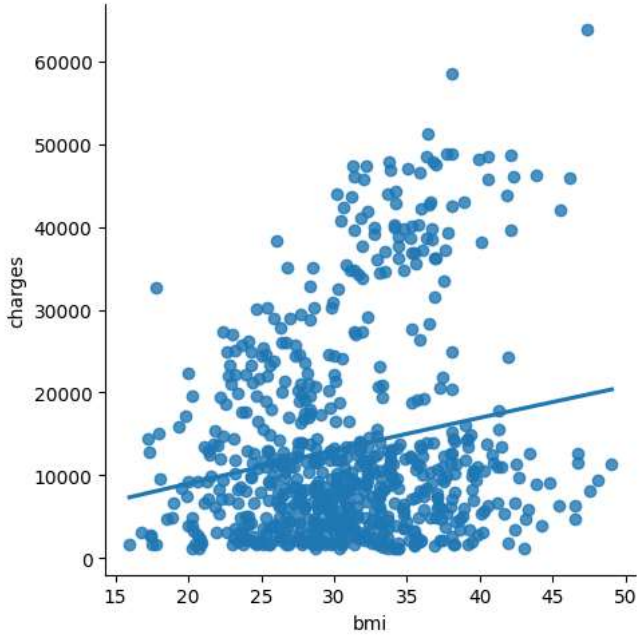
```
1.0
```

In [99]:
```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



Working With Subset of Data

In [100]:
```
df700=df[:][:700]
sns.lmplot(x='bmi',y='charges',order=2,ci=None,data=df700)
plt.show()
```



In [101]:
```
df700.fillna(method='ffill',inplace=True)
```

In [102]:
```
x=np.array(df700["bmi"]).reshape(-1,1)
y=np.array(df700['charges']).reshape(-1,1)
```

In [103]:
```
df700.dropna(inplace=True)
```

In [104]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

```
-0.024111643073524425
```

In [105]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

```
0.06881918421633926
```

Evaluation of model

In [106]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

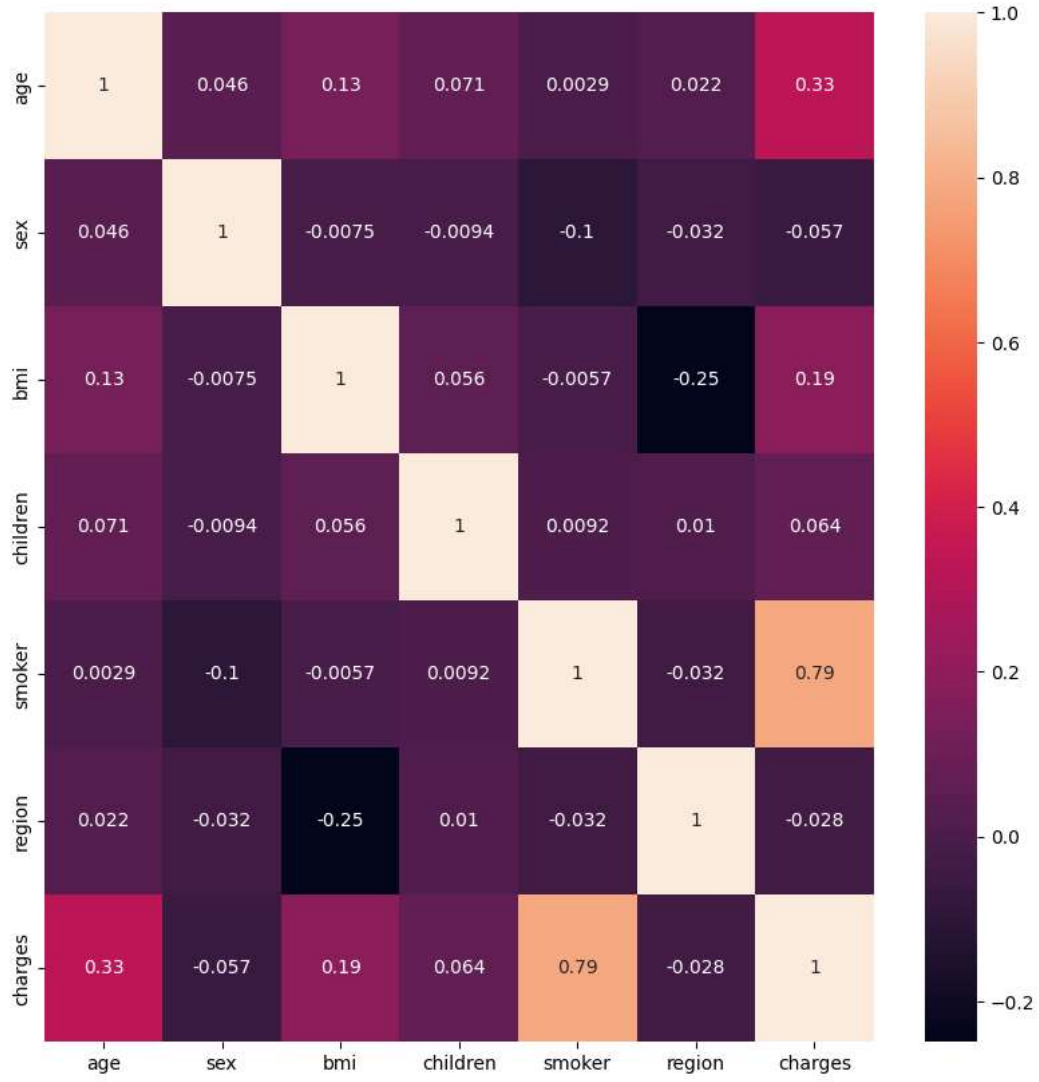
In [107]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
r2=r2_score(y_test,y_pred)
print(r2)
```

```
0.06881918421633926
```

Ridge Regression

In [108]:
```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [109]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df700.corr(),annot=True)
plt.show()
```



In [110]:
```
features=df.columns[0:1]
target=df.columns[-1]
```

In [111]:
```
x=df[features].values
y=df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=1)
print("The dimension of X_train is {}".format(x_train.shape))
print("The dimension of X_test is {}".format(x_test.shape))
```

```
The dimension of X_train is (936, 1)
The dimension of X_test is (402, 1)
```

In [112]:
```
lr = LinearRegression()
#Fit model
lr.fit(x_train, y_train)
#predict
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 0.0910963973805714
The test score for lr model is 0.08490473916580776
```

In [113]:
```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test scorefor ridge regression
train_score_ridge = ridgeReg.score(x_train, y_train)
test_score_ridge = ridgeReg.score(x_test, y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

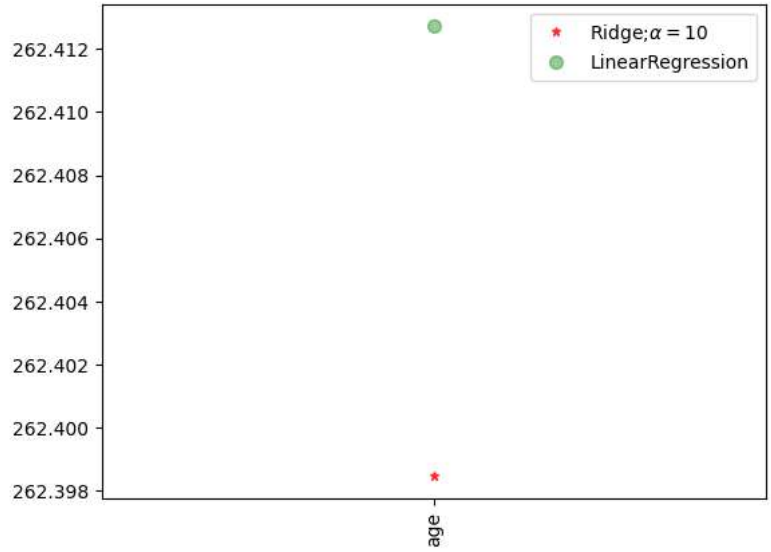
```
Ridge Model:

The train score for ridge model is 0.09109639711159634
The test score for ridge model is 0.08490538609860176
```

In [114]:
```
plt.figure(figsize=(10,10))
```

Out[114]:
```
<Figure size 1000x1000 with 0 Axes>
```

```
<Figure size 1000x1000 with 0 Axes>
```

In [115]:
```
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color='red',label=r'Ridge;$\alpha=10$',zorder
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



Lasso Regression

In [116]:
```
#Importing libraries
lasso= Lasso(alpha=10)
lasso.fit(x_train,y_train)
#train and test scorefor ridge regression
train_score_ls = lasso.score(x_train, y_train)
test_score_ls= lasso.score(x_test, y_test)
print("\nLasso Model:\n")
print("The train score for lasso model is {}".format(train_score_ls))
print("The test score for lasso model is {}".format(test_score_ls))
```

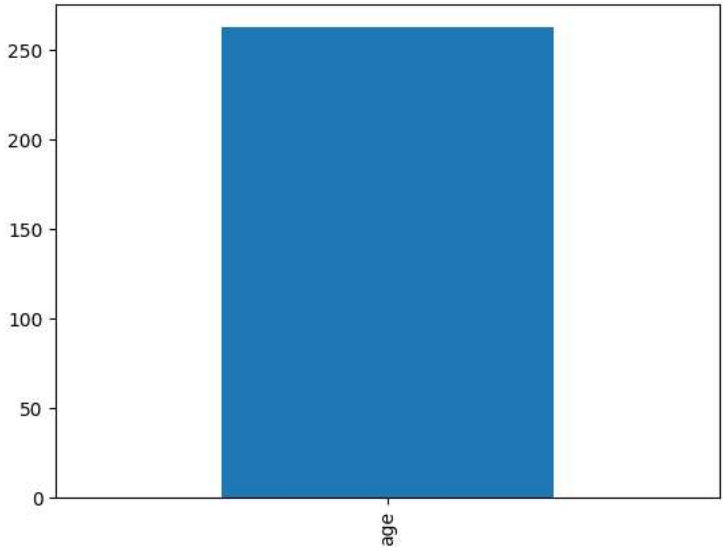
```
Lasso Model:

The train score for lasso model is 0.09109639395809044
The test score for lasso model is 0.08490704421828055
```

In [117]:
```
plt.figure(figsize=(10,10))
```

Out[117]:
```
<Figure size 1000x1000 with 0 Axes>
```

```
<Figure size 1000x1000 with 0 Axes>
```

In [118]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
plt.show()
```



In [119]:
```
from sklearn.linear_model import LassoCV
```

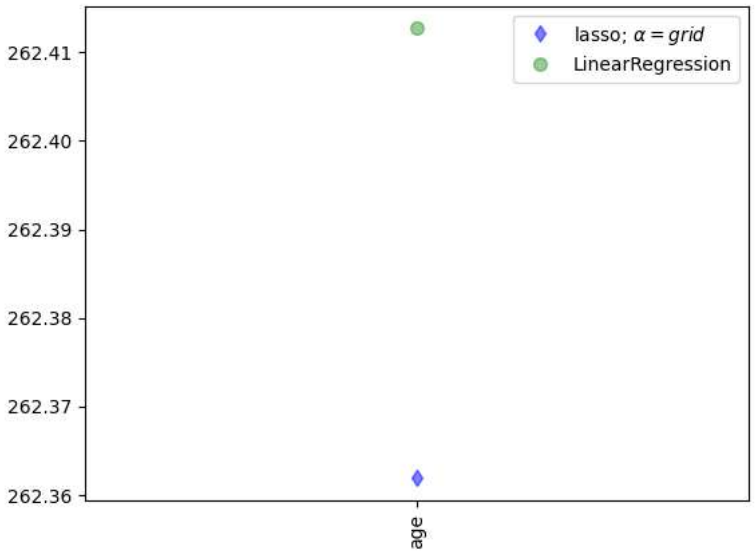
In [120]:
```
#using the linear cv model
from sklearn.linear_model import RidgeCV
#cross validation
ridge_cv=RidgeCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(ridge_cv.score(x_train,y_train))
print(ridge_cv.score(x_test,y_test))
```

```
0.09109639711159612
0.08490538609884613
```

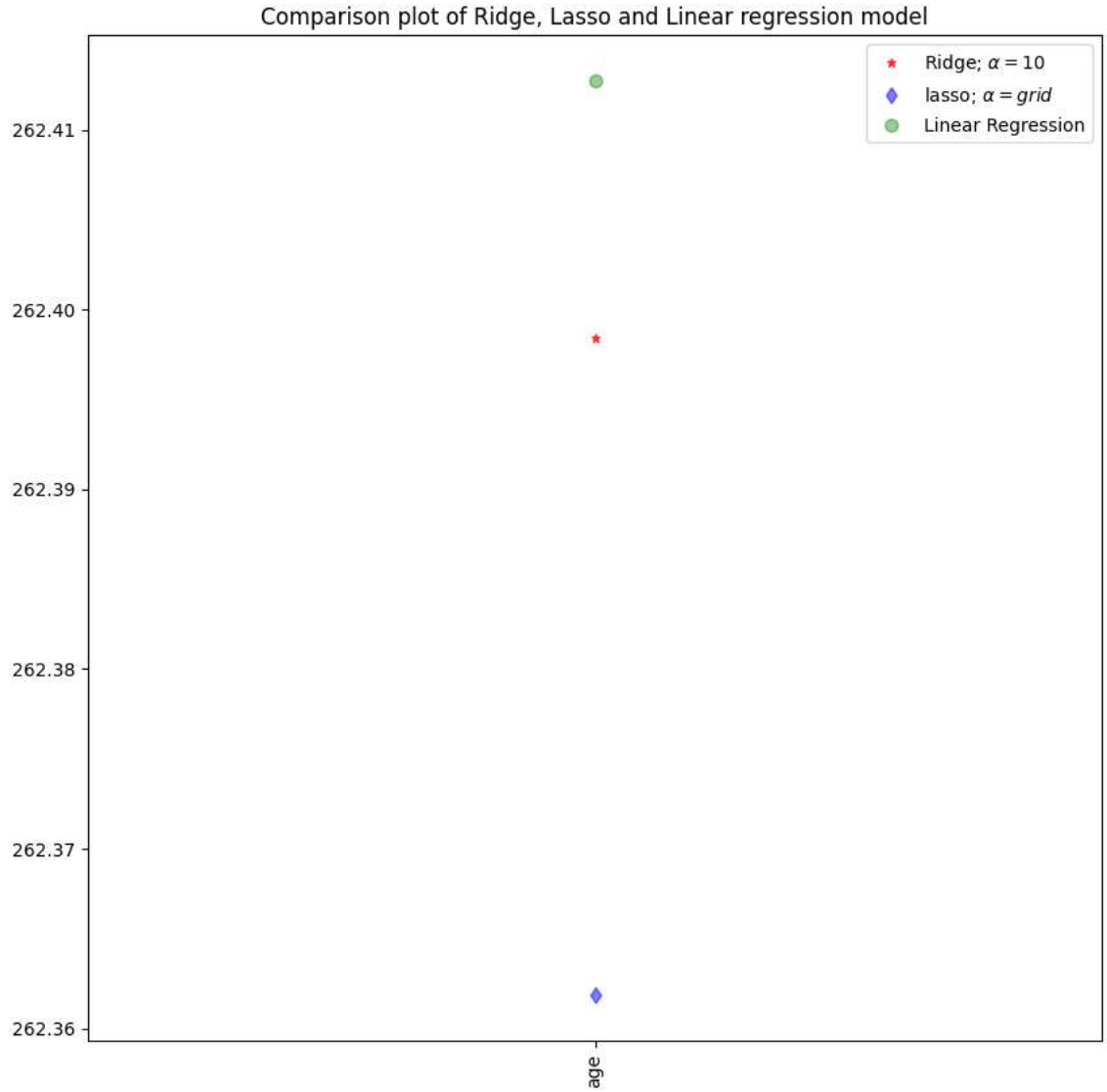
In [121]:
```
#using the linear cv model
from sklearn.linear_model import LassoCV
#cross validation
lasso_cv=LassoCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.09109639395809044
0.08490704421828055
```

In [122]:
```
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha = grid$')
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [123]:
```
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=r'Ridge; $\alpha=10$',zorde
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha = grid$')
#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()
```



The accuracy of the Lasso Model is 0.091096

ElasticNet Regression

In [124]:
```
from sklearn.linear_model import ElasticNet
```

In [125]:
```
el=ElasticNet()
el.fit(x_train,y_train)
print(el.coef_)
print(el.intercept_)
el.score(x,y)
```

```
[261.74450967]
3115.0831774262424
```

Out[125]: 0.08930616764094623

In [126]:
```
y_pred_elastic=el.predict(x_train)
```

In [127]:
```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
135077142.70714515
```

The accuracy of the ElasticNet is 0.08930

Logistic Regression

In [128]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [129]:
```
df=pd.read_csv(r"C:\Users\jangidi veena\OneDrive\Documents\jupyter\insurance.csv")
df
```

Out[129]:

	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 [130]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[130]:

	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 [131]:
```
df.shape
```

Out[131]:
```
(1338, 7)
```

In [132]:
```
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   int64  
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(3), object(2)
memory usage: 73.3+ KB
```

In [133]:
```
df.head()
```

Out[133]:

	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

In [134]:
```
df.tail()
```

Out[134]:

	age	sex	bmi	children	smoker	region	charges
1333	50	male	30.97	3	0	northwest	10600.5483
1334	18	female	31.92	0	0	northeast	2205.9808
1335	18	female	36.85	0	0	southeast	1629.8335
1336	21	female	25.80	0	0	southwest	2007.9450
1337	61	female	29.07	0	1	northwest	29141.3603

In [135]:
```
df.describe()
```

Out[135]:

	age	bmi	children	smoker	charges
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	0.204783	13270.422265
std	14.049960	6.098187	1.205493	0.403694	12110.011237
min	18.000000	15.960000	0.000000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	0.000000	9382.033000
75%	51.000000	34.693750	2.000000	0.000000	16639.912515
max	64.000000	53.130000	5.000000	1.000000	63770.428010

In [136]:
```
df.isnull().any()
```

Out[136]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [137]:
```
df.isna().sum()
```

Out[137]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [145]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[145]:

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

1338 rows × 7 columns

In [146]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

Out[146]:

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

1338 rows × 7 columns

In [147]:
```
convert={"region":{"southeast":1,"southwest":2,"northeast":3,"northwest":4}}
df=df.replace(convert)
df
```

Out[147]:

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

1338 rows × 7 columns

In [148]:
```
features_matrix=df.iloc[:,0:4]
```

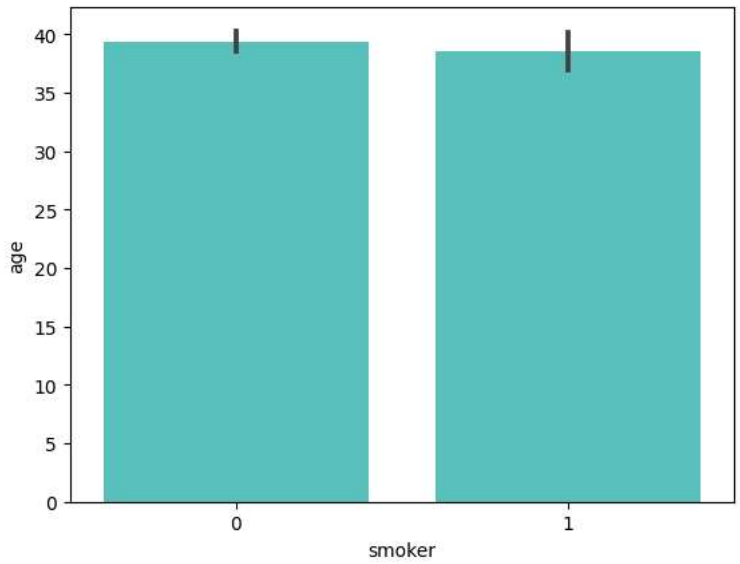
In [149]:
```
target_vector=df.iloc[:,-3]
```

In [150]:
```
print('The Feature Matrix has %d Rows and %d columns(s)'%(features_matrix.shape))
print('The Target Matrix has %d Rows and %d columns(s)'%(np.array(target_vector).reshape(-1,1).shape))
```

```
The Feature Matrix has 1338 Rows and 4 columns(s)
The Target Matrix has 1338 Rows and 1 columns(s)
```

In [151]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [152]:
```
sns.barplot(x='smoker', y='age', data=df, color="mediumturquoise")
plt.show()
```



In [153]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [154]:
```
algorithm=LogisticRegression(max_iter=10000)
```

In [155]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [156]:
```
observation=[[1,0,0.99539,-0.0588]]
```

In [157]:
```
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 [0]
```

In [158]:
```
print('The algoritham was trained to predict one of the two classes:%s'%(algorithm.classes_))
```

```
The algoritham was trained to predict one of the two classes:[0 1]
```

In [160]:
```
odel says the probability of the observation we passed belonging to class[0] %s" " "%(algorithm.predict_proba(observation)[0][0])
```

```
The Model says the probability of the observation we passed belonging to class[0] 0.8057075871331396
```

In [161]:
```
ays the probability of the observation we passed belonging to class['1'] Is %s" " "%(algorithm.predict_proba(observation)[0][0]))
```

```
The Model says the probability of the observation we passed belonging to class['1'] Is 0.8057075871331396
```

In [162]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
```

In [163]:
```
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

C:\Users\jangidi veena\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\validation.py:1143: DataConversi
onWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  y = column_or_1d(y, warn=True)
```

the accuracy of logistic regression is 0.7462

Decision Tree

In [164]:
```
#Importing Libraries
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 [165]:
```
df=pd.read_csv(r"C:\Users\jangidi veena\OneDrive\Documents\jupyter\insurance.csv")
df
```

Out[165]:

	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 [166]:
```
df.head()
```

Out[166]:

	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 [167]:
```
df.tail()
```

Out[167]:

	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 [168]:
```
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 [169]:
```
df.describe()
```

Out[169]:

	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 [170]:
```
df['region'].value_counts()
```

Out[170]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [171]:
```
df.shape
```

Out[171]: (1338, 7)

In [172]:
```
df.isnull().any()
```

Out[172]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [173]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

Out[173]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	yes	southwest	16884.92400
1	18	0	33.770	1	no	southeast	1725.55230
2	28	0	33.000	3	no	southeast	4449.46200
3	33	0	22.705	0	no	northwest	21984.47061
4	32	0	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	0	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 [174]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[174]:

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

1338 rows × 7 columns

In [176]:
```
x=["bmi","children"]
y=["Yes","No"]
all_inputs=df[x]
all_classes=df["sex"]
```

In [177]:
```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.03)
```

In [178]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [179]:
```
clf.fit(x_train,y_train)
```

Out[179]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [180]:
```
score=clf.score(x_test,y_test)
print(score)
```

```
0.43902439024390244
```

The accuracy of the Decision Tree is 0.39024

Random Forest

In [181]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt ,seaborn as sns
```

In [182]:
```
df=pd.read_csv(r"C:\Users\jangidi veena\OneDrive\Documents\jupyter\insurance.csv")
df
```

Out[182]:

	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 [183]:
```
df.head()
```

Out[183]:

	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 [184]:
```
df.tail()
```

Out[184]:

	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 [185]:
```
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 [186]:
```
df.describe()
```

Out[186]:

	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 [187]:
```
df.shape
```

Out[187]:
```
(1338, 7)
```

In [188]:
```
df.isnull().any()
```

Out[188]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [189]:
```
df['region'].value_counts()
```

Out[189]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [190]:
```
df['bmi'].value_counts()
```

Out[190]:
```
bmi
32.300    13
28.310     9
30.495     8
30.875     8
31.350     8
          ..
46.200     1
23.800     1
44.770     1
32.120     1
30.970     1
Name: count, Length: 548, dtype: int64
```

In [191]:
```
m={"sex":{"female":1,"male":0}}
df=df.replace(m)
print(df)
```

```
      age  sex     bmi  children smoker     region      charges
0      19    1  27.900         0    yes  southwest  16884.92400
1      18    0  33.770         1     no  southeast   1725.55230
2      28    0  33.000         3     no  southeast   4449.46200
3      33    0  22.705         0     no  northwest  21984.47061
4      32    0  28.880         0     no  northwest   3866.85520
...   ...  ...     ...       ...    ...        ...         ...
1333   50    0  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 x 7 columns]
```

In [192]:
```
n={"smoker":{"yes":1,"no":0}}
df=df.replace(n)
print(df)
```

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

[1338 rows x 7 columns]
```

In [193]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[193]:
```
RandomForestClassifier()
```

In [194]:
```
RandomForestClassifier()
```

Out[194]:
```
RandomForestClassifier()
```

In [197]:
```
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 [198]:
```
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[198]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [199]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
 param_grid={'max_depth': [2, 3, 5, 20],
 'min_samples_leaf': [5, 10, 20, 50, 100, 200],
 'n_estimators': [10, 25, 30, 50, 100, 200]},
 scoring='accuracy')
```

Out[199]:

GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier

In [200]:
```
grid_search.best_score_
```

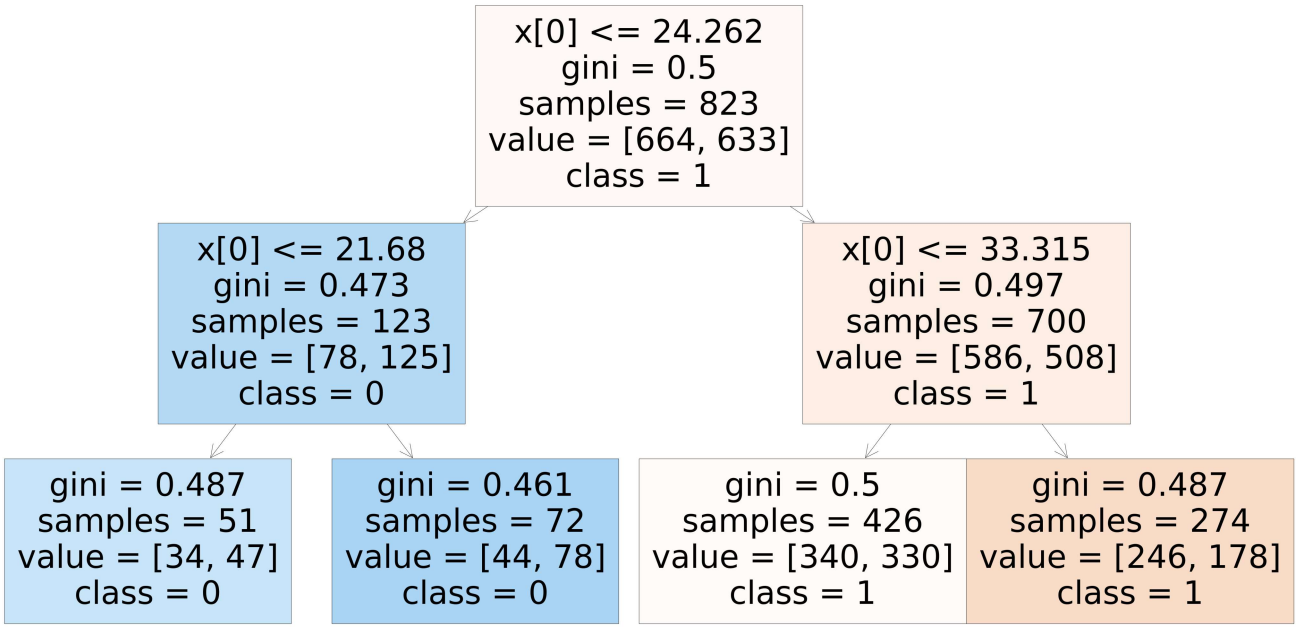
Out[200]:
```
0.5304552112461718
```

In [201]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

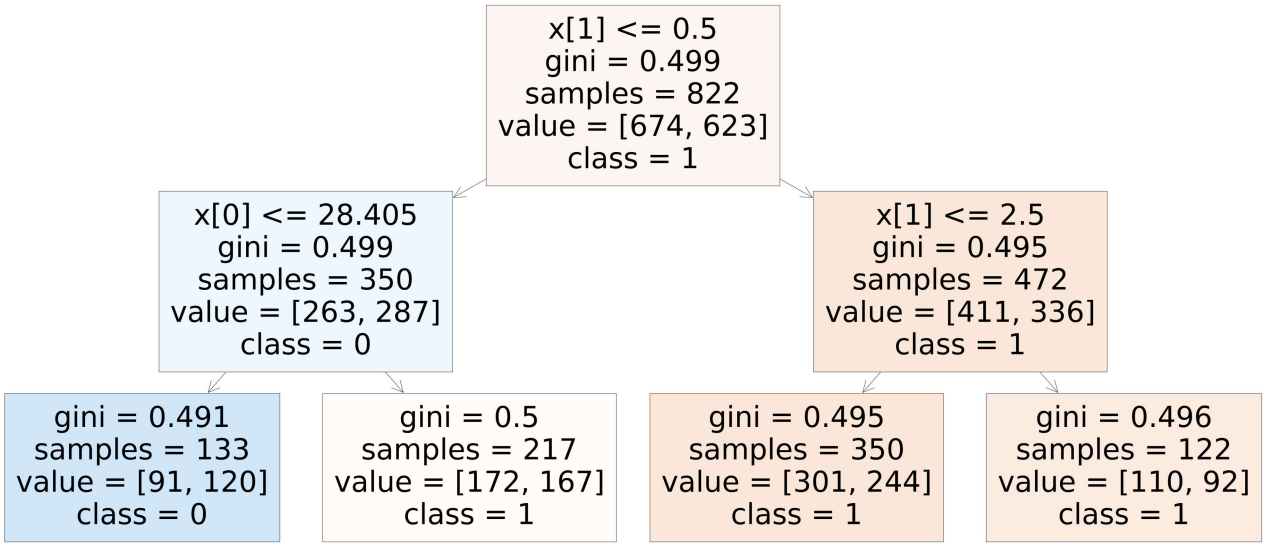
In [202]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=2, min_samples_leaf=50, n_estimators=10)
```

In [203]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['1','0'],filled=True);
plt.show()
```



In [204]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(70,30))
plot_tree(rf_best.estimators_[6],class_names=["1","0"],filled=True);
plt.show()
```



In [205]:
```
rf_best.feature_importances_
```

Out[205]:
```
array([0.82039237, 0.17960763])
```

In [206]:
```
rf=RandomForestClassifier(random_state=0)
```

In [207]:
```
rf.fit(x_train,y_train)
```

Out[207]:
```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

In [208]:
```
score=rf.score(x_test,y_test)
print(score)
```

```
0.3902439024390244
```

The accuracy of the Random Forest is 0.39024

Conclusion

In the given data set we need to find the bestfit Model . in this data set to find accuracy we have used different types of Models. Among all the models ElasticNet Regression is the Bestfit Model

In [ ]:
```
```