PROBLEM STATEMENT:

TO CHECK HOW BEST FIT IS IT?

importing the libraries # #

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

DATA COLLECTION

In [82]:
```
df=pd.read_csv(r"C:\Users\MY HOME\Downloads\insurance.csv")
df
```

Out[82]:

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

1338 rows × 7 columns

DATA CLEANING

In [83]:
```
df.head()
```

Out[83]:

	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 [84]:
```
df.tail()
```

Out[84]:

	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 [85]:
```
df.describe()
```

Out[85]:

	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

to check missing values

In [86]:
```
df.isnull().sum()
```

Out[86]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [87]:
```
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 [88]:
```
sex={"sex":{"female":0,"male":1}}
df=df.replace(sex)
df
```

Out[88]:

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

1338 rows × 7 columns

In [89]:
```
smoker={"smoker":{"yes":1,"no":0}}
df=df.replace(smoker)
df
```

Out[89]:

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

1338 rows × 7 columns

In [94]:
```
df.drop("region",axis=1)
```

Out[94]:

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

1338 rows × 6 columns

In [55]:
```
#taking selected columns from dataset
df=df[['age','bmi']]
df
```

Out[55]:

	age	bmi
0	19	27.900
1	18	33.770
2	28	33.000
3	33	22.705
4	32	28.880
...	...	...
1333	50	30.970
1334	18	31.920
1335	18	36.850
1336	21	25.800
1337	61	29.070

1338 rows × 2 columns

In [56]:
```
df.head(10)
```

Out[56]:

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

In [57]:
```
df.tail(10)
```

Out[57]:

	age	bmi
1328	23	24.225
1329	52	38.600
1330	57	25.740
1331	23	33.400
1332	52	44.700
1333	50	30.970
1334	18	31.920
1335	18	36.850
1336	21	25.800
1337	61	29.070

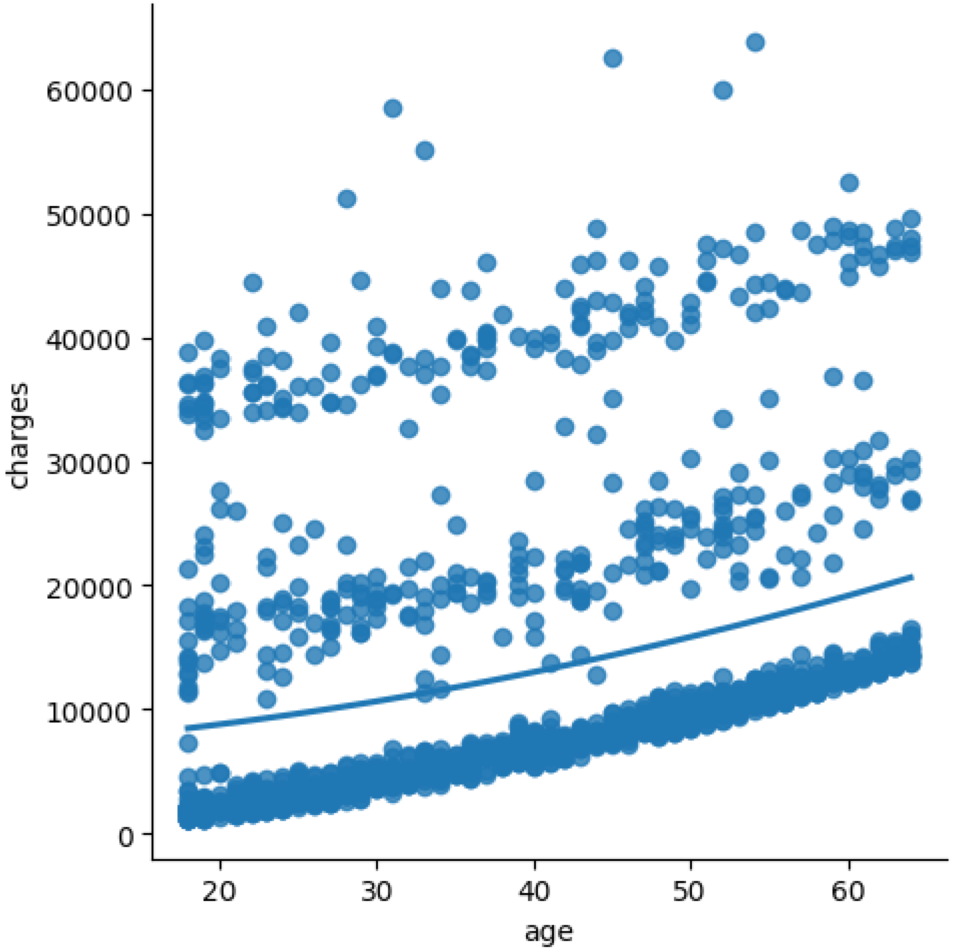
In [58]:
```
sex={"sex":{"female":0,"male":1}}
df=df.replace(sex)
df
```

Out[58]:

	age	bmi
0	19	27.900
1	18	33.770
2	28	33.000
3	33	22.705
4	32	28.880
...	...	...
1333	50	30.970
1334	18	31.920
1335	18	36.850
1336	21	25.800
1337	61	29.070

1338 rows × 2 columns

In [95]:
```
sns.lmplot(x="age",y="charges",order=2,data=df,ci=None)
plt.show()
```



In [102]:
```
df.drop("region",axis=1)
```

Out[102]:

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

1338 rows × 6 columns

DATA VISUALIZATION

In [107]:
```
features=df.columns[:5]
target=df.columns[-1]
```

In [108]:
```
x=df[features].values
y=df[target].values
```

In [109]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.3)
```

In [110]:

```
a=LinearRegression()
a.fit(x_train,y_train)
```

Out[110]:

```
LinearRegression()
```

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

In [111]:

```
print(a.score(x_test,y_test))
```

```
0.7577296238137559
```

In [112]:

```
a=LinearRegression()
a.fit(x_train,y_train)
train_score_a=a.score(x_train,y_train)
test_score_a=a.score(x_test,y_test)
print("\nLinearModel:\nThe train score for lr model is {}".format(train_score_a))
print("The train score for lr model is {}".format(test_score_a))
```

```

LinearModel:
The train score for lr model is 0.7182473271142464
The train score for lr model is 0.7577296238137559
```

RIDGE regression

In [ ]:
```
-------> we are going to do Ridge regression to check for better accuracy/model.
```

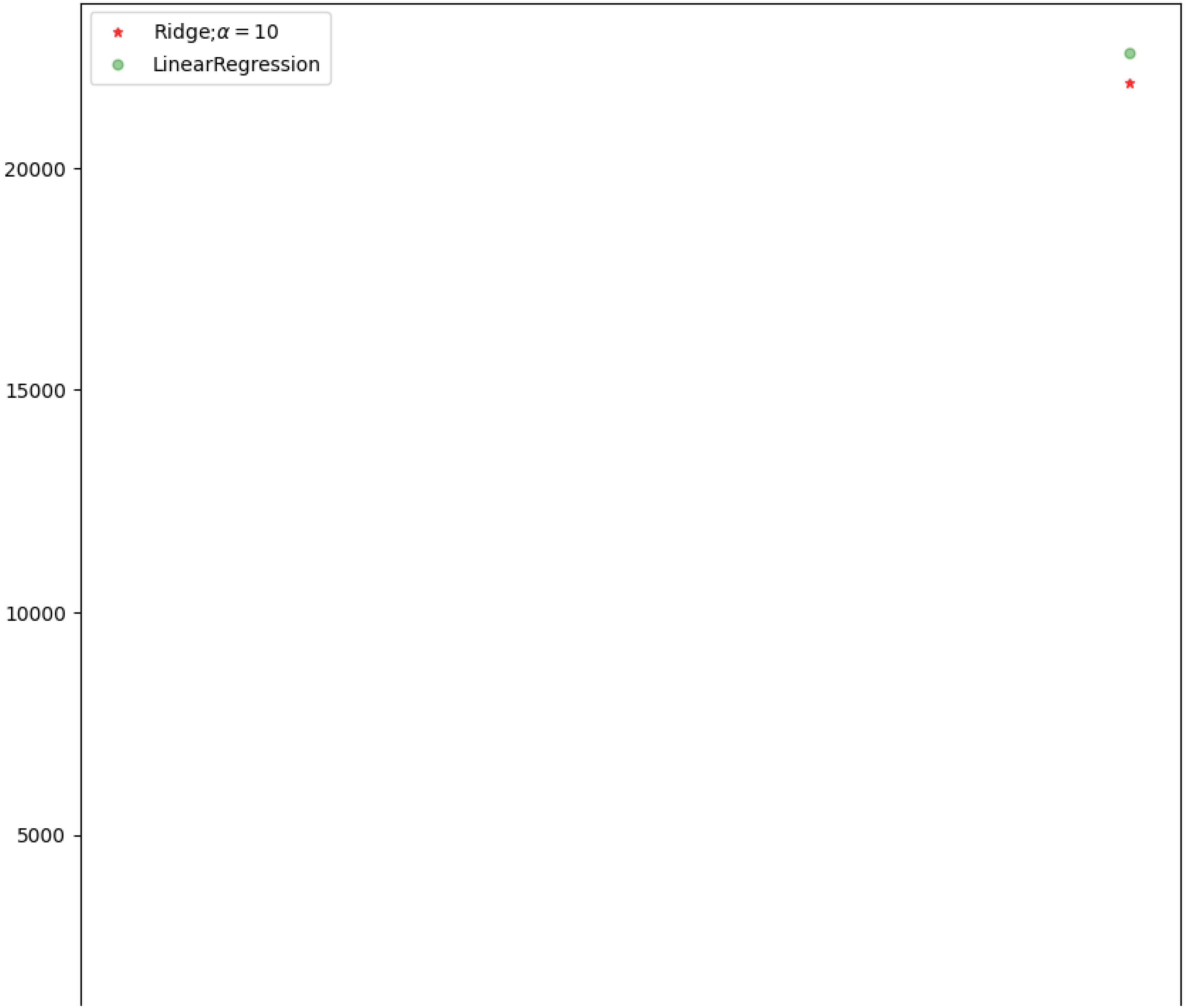
In [113]:
```
from sklearn.linear_model import Ridge, RidgeCV, Lasso
```

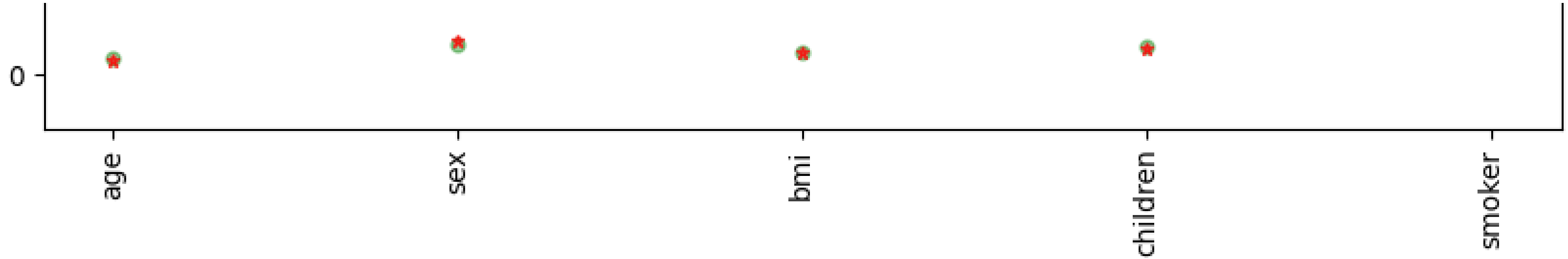
In [114]:
```
ridge=Ridge(alpha=2)
ridge.fit(x_train,y_train)
train_score_ridge=ridge.score(x_train,y_train)
test_score_ridge=ridge.score(x_test,y_test)
print("\nLinearRegression\n",(train_score_ridge))
print(test_score_ridge)
```

```
LinearRegression
 0.7176943354666219
0.7547364699646179
```

In [115]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridge.coef_,alpha=0.7,linestyle='None',marker='*',markersize=5,color='red',label=r'Ridge;$\alp
plt.plot(features,a.coef_,alpha=0.4,linestyle='None',marker='o',markersize=5,color='green',label='LinearRegressi
plt.xticks(rotation=90)
plt.legend()
plt.show()
```


Ridge;$\alpha = 10$
LinearRegression
20000
15000
10000
5000



LASSO regression

In [ ]:
```
------->to check best fit
```

In [116]:
```
lasso=Lasso(alpha=100)
lasso=lasso.fit(x_train,y_train)
train_score_lasso=lasso.score(x_train,y_train)
test_score_lasso=lasso.score(x_test,y_test)
print(train_score_lasso)
print(test_score_lasso)
```

```
0.717493707984145
0.7558673596080301
```

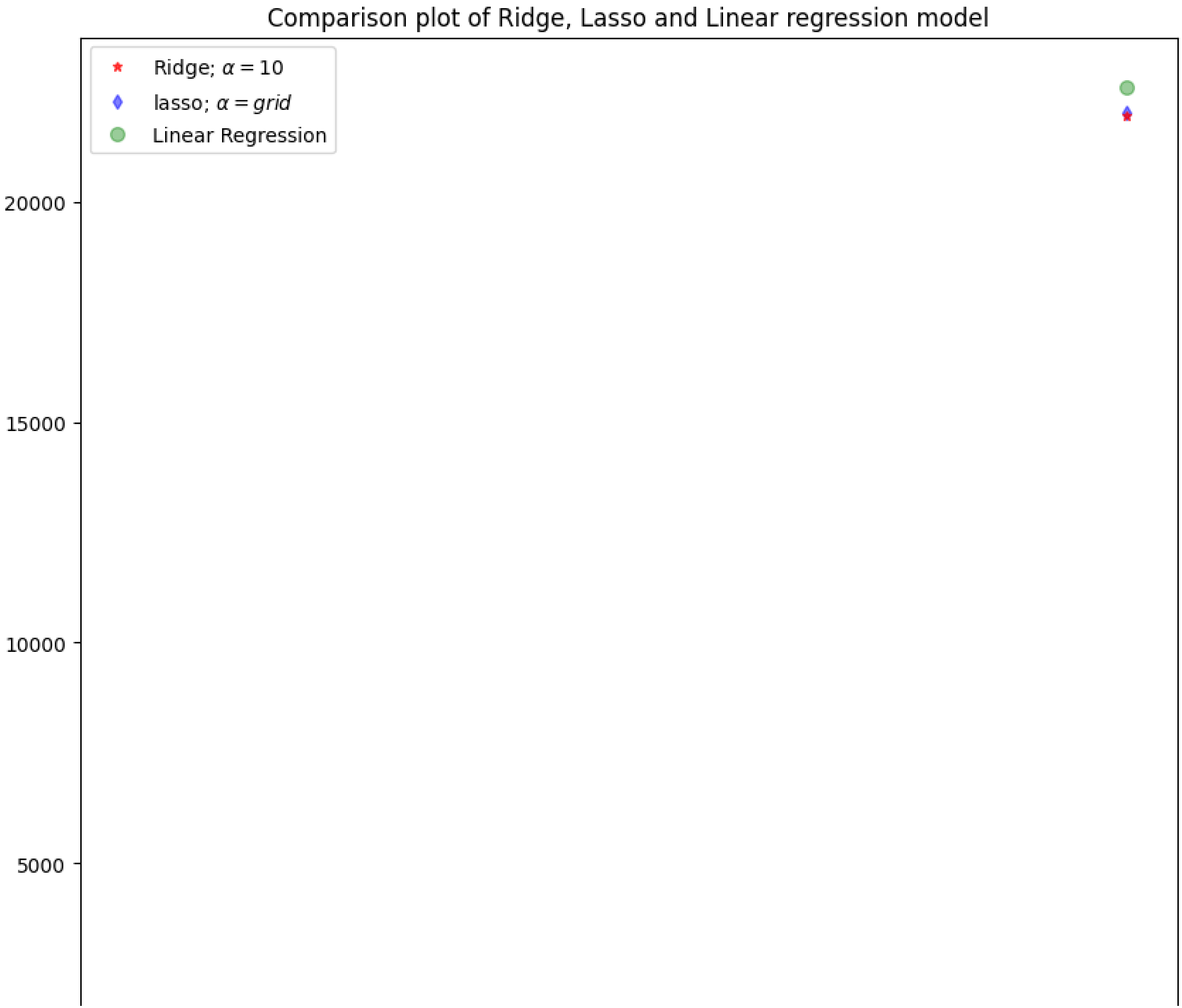
In [117]:

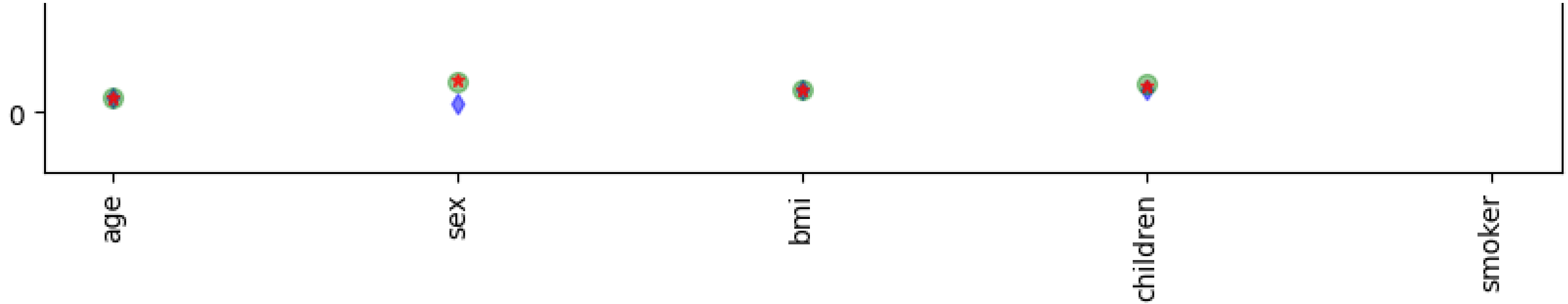
```
 #plot size
plt.figure(figsize = (10, 10)) #add plot for ridge regression
plt.plot(features,ridge.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge; $\al

#add plot for lasso regression
plt.plot(lasso.coef_,alpha=0.5,linestyle='none',marker='d',markersize=5,color='blue',label=r'lasso; $\alpha = gr

#add plot for linear model
plt.plot(features,a.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regress

#rotate ax
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```


Comparison plot of Ridge, Lasso and Linear regression model
Ridge; $\alpha = 10$
lasso; $\alpha = grid$
Linear Regression
20000
15000
10000
5000



ELASTICNET

In [118]:
```
from sklearn.linear_model import ElasticNet
```

In [119]:
```
a=ElasticNet()
a.fit(x,y)
print(a.coef_)
print(a.intercept_)
```

```
[ 244.74498193  323.34788404  324.21935152  389.31828171 5839.32681943]
-8052.400589902743
```

In [120]:
```
y_pred_elastic=a.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
83960709.7230712
```

after doing the Linear Regression we got 0.75 means 75% accuracy. to check for better model to fit we did Ridge and Lasso Regresion.then,we got a very minimal variation.so,there is no difference in terms of accuracy. to check for better accuracy we are going to do Logistic regression.

LOGISTIC REGRESSION

PROBLEM STAETOMENT:

```
TO CHECK IS THIS BEST FIT OR NOT COMPARED TO LINEAR REGRESSION
```

importing required libraries #

In [49]:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [50]:
```
df=pd.read_csv(r"C:\Users\MY HOME\Downloads\insurance.csv")
df
```

Out[50]:

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

1338 rows × 7 columns

DATA CLEANING

In [51]:
```
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 [52]:
```
df.describe()
```

Out[52]:

	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 [53]:
```
df.shape
```

Out[53]: (1338, 7)

In [54]:
```
sex={"sex":{"female":0,"male":1}}
df=df.replace(sex)
df
```

Out[54]:

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

1338 rows × 7 columns

In [55]:
```
df.pop("charges")
```

Out[55]:
```
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
           ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

In [56]:
```
df.pop("region")
```

Out[56]:
```
0       southwest
1       southeast
2       southeast
3       northwest
4       northwest
          ...
1333    northwest
1334    northeast
1335    southeast
1336    southwest
1337    northwest
Name: region, Length: 1338, dtype: object
```

In [57]:
```
print('This DataFrame has %d rows and %d columns'%(df.shape))
```

```
This DataFrame has 1338 rows and 5 columns
```

In [58]:
```
features=df.iloc[:,0:4]
target=df.iloc[:,-1]
print('The features matrix has %d Rows and %d columns'%(features_matrix.shape))
print('The features matrix has %d Rows and %d columns'%(np.array(target_vector).reshape(-1,1).shape))
```

```
The features matrix has 1338 Rows and 4 columns
The features matrix has 1338 Rows and 1 columns
```

In [59]:
```
features_Standardized=StandardScaler().fit_transform(features)
```

In [60]:
```
algorithm=LogisticRegression(penalty=None,dual=False,tol=1e-4,C=1.0,fit_intercept=True,intercept_scaling=1,class
```

In [61]:
```
Logistic_Regression_Model=algorithm.fit(features_Standardized,target)
```

In [62]:
```
observation=[[1,1,0,1]]
```

In [63]:
```
predictions=Logistic_Regression_Model.predict(observation)
print('The model predicted the observtaion to belong to class %s'%(predictions))
print('The algorithm was trained to predict one of the two calsses : %s'%(algorithm.classes_))
```

```
The model predicted the observtaion to belong to class ['no']
The algorithm was trained to predict one of the two calsses : ['no' 'yes']
```

In [64]:
```
print("""The model says the prbability of the observation we passed belonging to calss ['0'] Is %s"""%(algorithm
print()
print("""The model says the prbability of the observation we passed belonging to calss ['1'] Is %s"""%(algorithm
```

```
The model says the prbability of the observation we passed belonging to calss ['0'] Is 0.7724108950235217

The model says the prbability of the observation we passed belonging to calss ['1'] Is 0.22758910497647833
```

DECISION TREE

In [65]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

In [69]:

```
df
```

Out[69]:

	age	sex	bmi	children	smoker
0	19	0	27.900	0	yes
1	18	1	33.770	1	no
2	28	1	33.000	3	no
3	33	1	22.705	0	no
4	32	1	28.880	0	no
...	...	...	...	...	...
1333	50	1	30.970	3	no
1334	18	0	31.920	0	no
1335	18	0	36.850	0	no
1336	21	0	25.800	0	no
1337	61	0	29.070	0	yes

1338 rows × 5 columns

In [70]:

```
df["sex"].value_counts()
```

Out[70]:

```
sex
1    676
0    662
Name: count, dtype: int64
```

In [71]:
```
df["smoker"].value_counts()
```

Out[71]:
```
smoker
no      1064
yes      274
Name: count, dtype: int64
```

In [72]:
```
x=["age","sex","children","bmi"]
y=["0","1"]
all_inputs=df[x]
all_classes=df["smoker"]
```

In [74]:
```
x_train,x_test,y_train,y_test=train_test_split(all_inputs,all_classes,test_size=0.5)
x_train.shape,x_test.shape
```

Out[74]:
```
((669, 4), (669, 4))
```

In [75]:
```
s=DecisionTreeClassifier(random_state=20)
s.fit(x_train,y_train)
score=s.score(x_test,y_test)
print(score)
```

```
0.680119581464873
```

RANDOM FOREST CLASSIFICATION

In [77]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

Out[77]:
```
RandomForestClassifier()
```

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

In [78]:
```
params={"max_depth":[1,23,4,56,85],"min_samples_leaf":[4,6,8,10,12],"n_estimators":[8,9,10,65,42]}
```

In [79]:
```
from sklearn.model_selection import GridSearchCV
```

In [80]:
```
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2)
grid_search.fit(x_train,y_train)
print(grid_search.score(x_test,y_test))
```

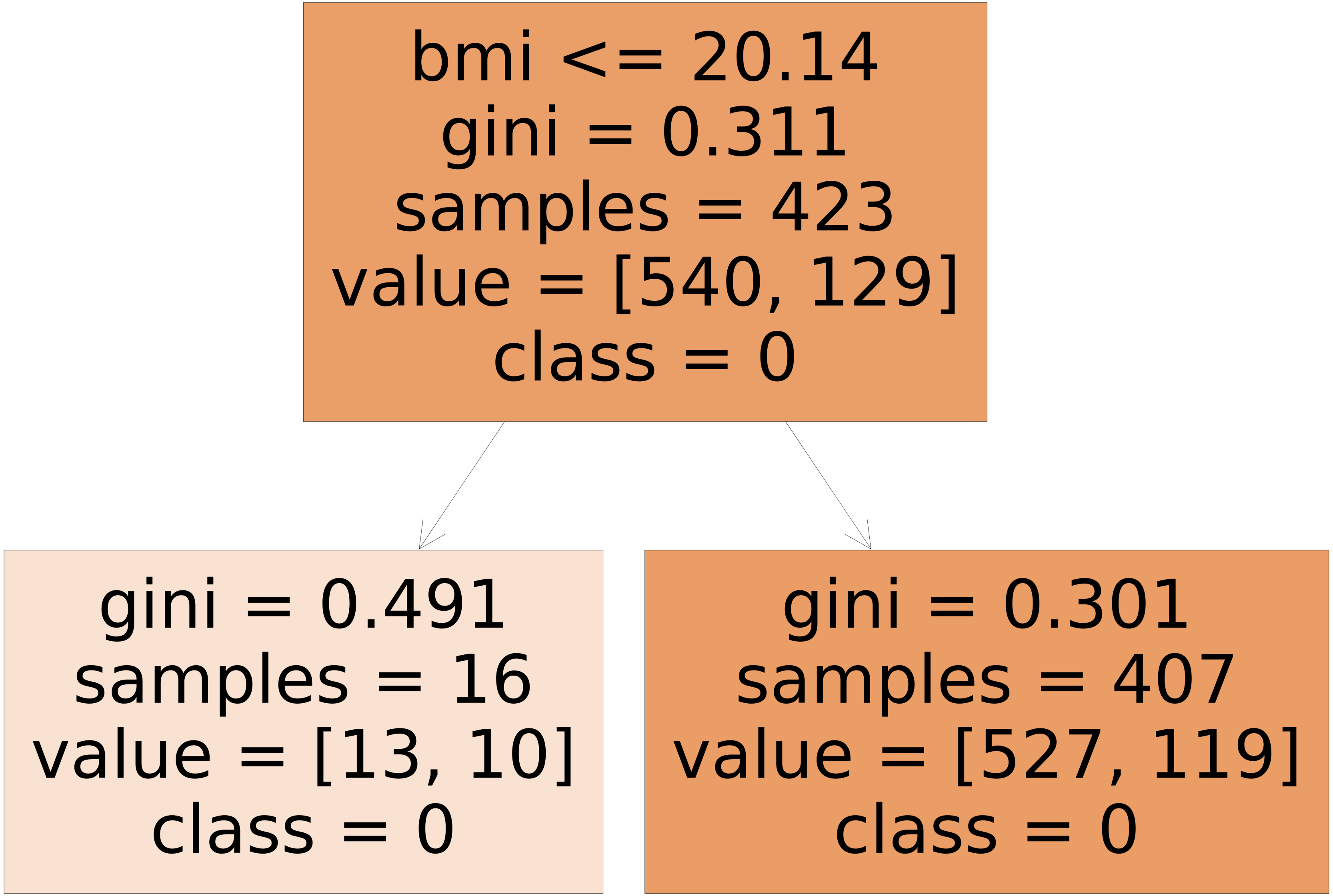
```
0.7952167414050823
```

In [81]:
```
p=grid_search.best_estimator_
print(p)
```

```
RandomForestClassifier(max_depth=1, min_samples_leaf=4, n_estimators=8)
```

In [87]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,60))
plot_tree(p.estimators_[3],feature_names=x,class_names=["0","1"],filled=True)
```

Out[87]:
```
[Text(0.5, 0.75, 'bmi <= 20.14\ngini = 0.311\nsamples = 423\nvalue = [540, 129]\nclass = 0'),
 Text(0.25, 0.25, 'gini = 0.491\nsamples = 16\nvalue = [13, 10]\nclass = 0'),
 Text(0.75, 0.25, 'gini = 0.301\nsamples = 407\nvalue = [527, 119]\nclass = 0')]
```


bmi <= 20.14
gini = 0.311
samples = 423
value = [540, 129]
class = 0
gini = 0.491
samples = 16
value = [13, 10]
class = 0
gini = 0.301
samples = 407
value = [527, 119]
class = 0

In [88]:
```
p.feature_importances_
```

Out[88]:
```
array([0.125, 0.125, 0.   , 0.75 ])
```

In [89]:
```
imp=pd.DataFrame({"varname":x_train.columns,"Imp":p.feature_importances_})
```

In [90]:
```
imp.sort_values(by="Imp",ascending=True)
```

Out[90]:

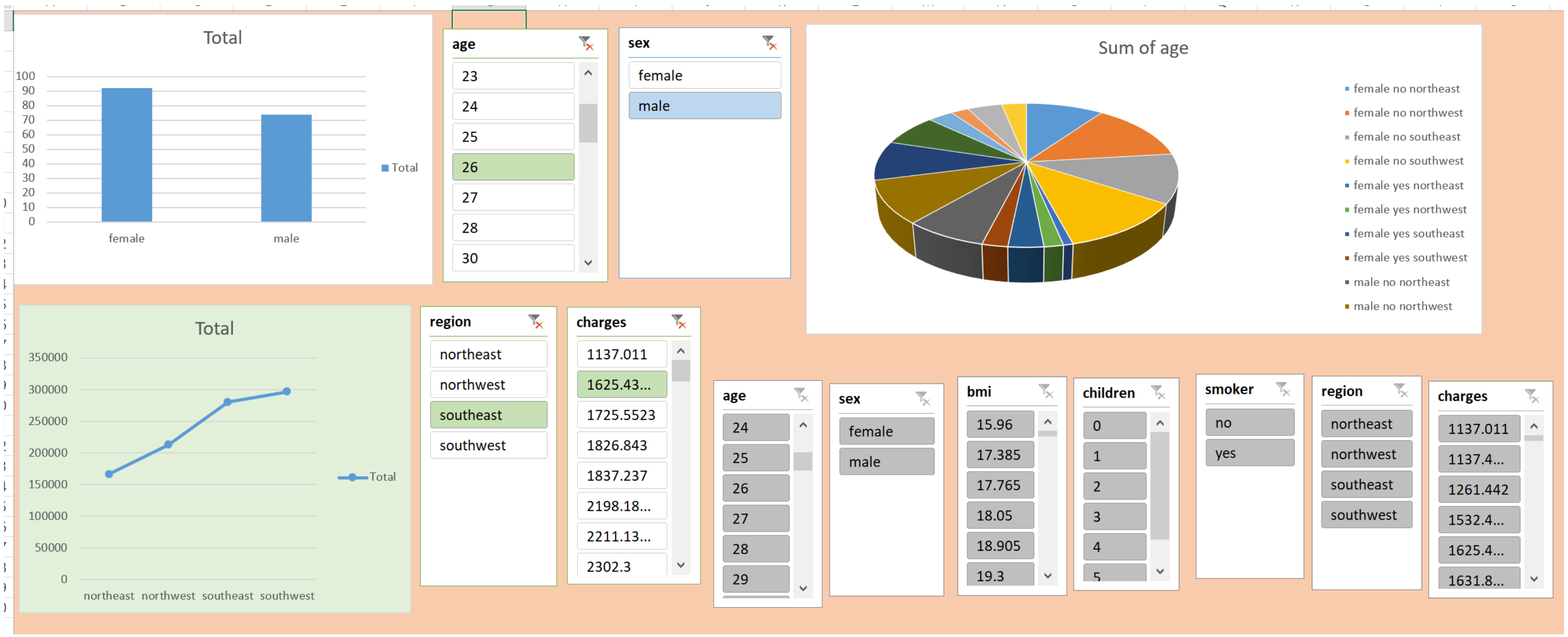
	varname	Imp
2	children	0.000
0	age	0.125
1	sex	0.125
3	bmi	0.750

after doing logistic regression we got 77%.so we did Decision Tree and Random forest classifier for better accuracy. for Decision tree we got 68% of accuacy. for Random forest classifier we got 79% of accuracy.

CONLCUSION:

```
            ------>Based on all model accuracies we conclude that the Random forest classification is somewhat
best fit campared to all other models with 79% of accuracy.
```

DASHBOARD


Total
100
90
80
70
60
50
40
30
20
10
0
female
male
Total
age
23
24
25
26
27
28
30
sex
female
male
Sum of age
female no northeast
female no northwest
female no southeast
female no southwest
female yes northeast
female yes northwest
female yes southeast
female yes southwest
male no northeast
male no northwest
Total
350000
300000
250000
200000
150000
100000
50000
0
northeast northwest southeast southwest
Total
region
northeast
northwest
southeast
southwest
charges
1137.011
1625.43...
1725.5523
1826.843
1837.237
2198.18...
2211.13...
2302.3
age
24
25
26
27
28
29
sex
female
male
bmi
15.96
17.385
17.765
18.05
18.905
19.3
children
0
1
2
3
4
5
smoker
no
yes
region
northeast
northwest
southeast
southwest
charges
1137.011
1137.4...
1261.442
1532.4...
1625.4...
1631.8...

In [ ]: