In [ ]:

```
#importing libraries
import numpy as np,pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sb
```

reading the dataset

In [2]:

```
df=pd.read_csv(r"C:\Users\MY HOME\Desktop\datascience\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

perfoming the basic preprocessing steps

In [3]:

```
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 [4]:

```
df.shape
```

Out[4]:

```
(1338, 7)
```

In [5]:

```
df.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]:

```
df.isna().any()
```

Out[6]:

```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [7]:

```
df.head()
```

Out[7]:

	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 [8]:

```
df.tail()
```

Out[8]:

	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

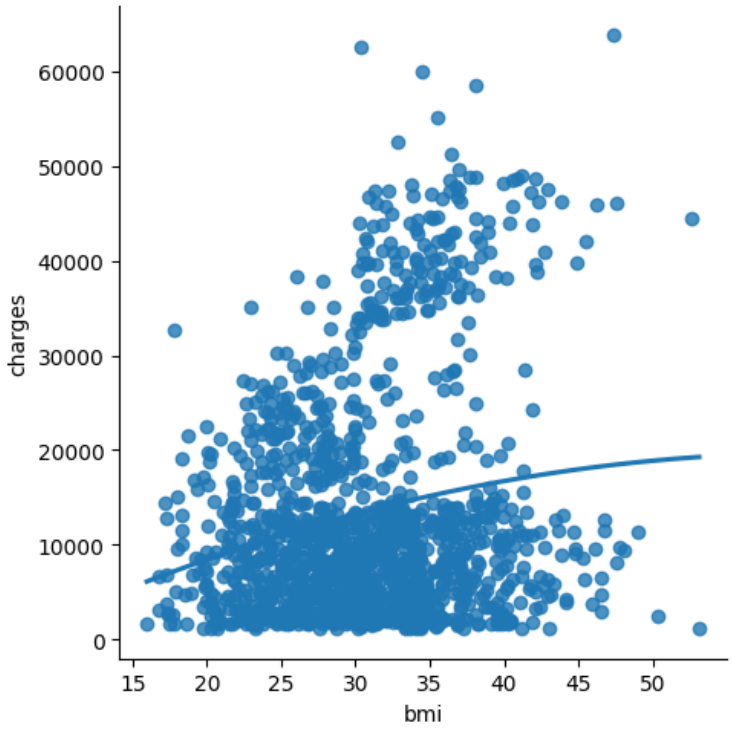
```
# permorfing the data analysis
```

In [9]:

```
1 sb.lmplot(x="bmi",y="charges",data=df,order=2,c i=None)
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x21d0e99fa90>
```



In [10]:

```
1 df.drop("charges",axis=1)
```

Out[10]:

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

1338 rows × 6 columns

In [11]:

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

Out[11]:

	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 [12]:

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

Out[12]:

	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 [13]:

```
features=df.columns[0:5]
target=df.columns[-1]
```

In [14]:

```
x=df[features].values
y=df[target].values
```

In [15]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

```
# building a model
```

In [16]:

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

Out[16]:

```
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 [17]:

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

```
0.7335428743506633
```

In [18]:

```
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\n")
print("The 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 0.7603956164064063
The train score for lr model is 0.7335428743506633
```

```
# ridge and lasso Regression

```

In [20]:

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

In [21]:

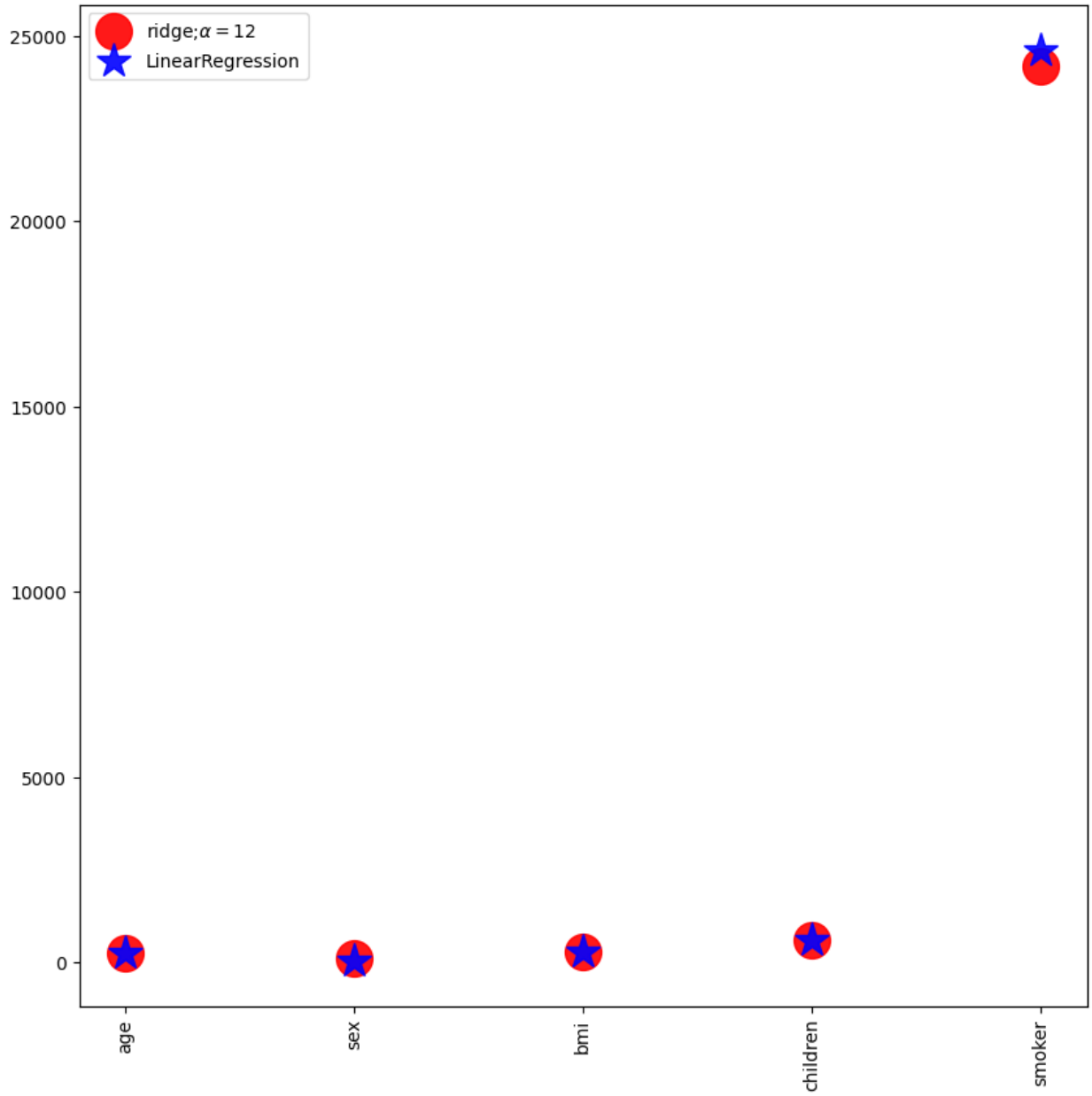
```
ridge=Ridge(alpha=2)
ridge.fit(x_train,y_train)
train_score_ridge=ridge.score(x_train,y_train)
test_score_ridge=ridge.score(x_test,y_test)
print("\nLinearRegression\n")
print(train_score_ridge)
print(test_score_ridge)
```

```
LinearRegression

0.76018709161429
0.7350061477802261
```

In [22]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridge.coef_,alpha=0.9,linestyle="None",markersize=20,color="Red",label=r"ridge;$\alpha=12$",marke
plt.plot(features,a.coef_,alpha=0.9,linestyle="None",markersize=20,color="blue",label="LinearRegression",marker="*"
plt.xticks(rotation=90)
plt.legend()
plt.show()
```




```
# lasso Regression
```

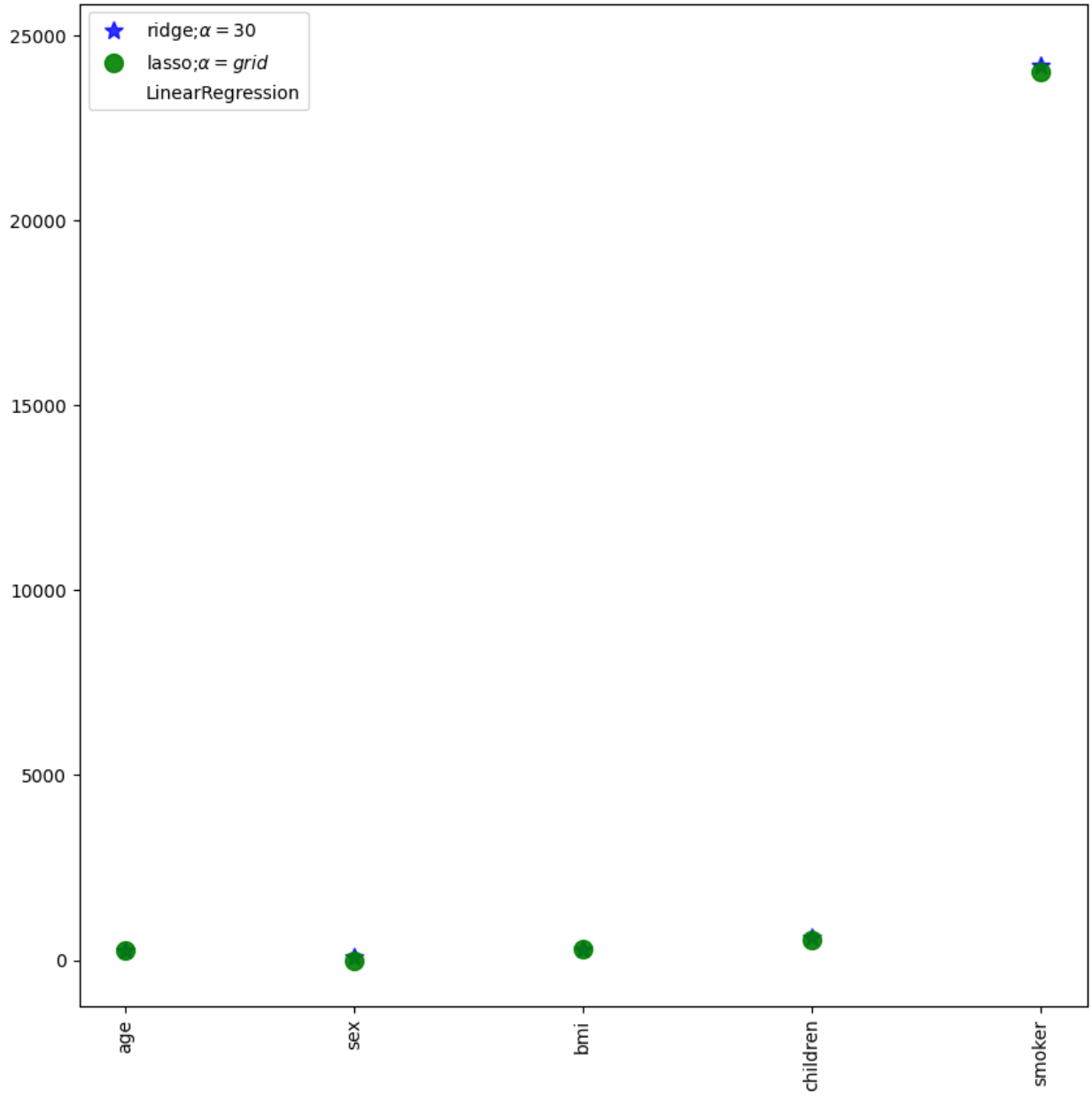
In [24]:

```
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)
```

```
0.7599483955052437
0.7359124297917283
```

In [25]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridge.coef_,alpha=0.8,marker="*",markersize=10,linestyle="none",color="blue",label=r"ridge;$\alpha
plt.plot(lasso.coef_,alpha=0.9,marker='o',markersize=10,linestyle="none",color="green",label=r"lasso;$\alpha=grid$"
plt.plot(features,a.coef_,alpha=0.7,linestyle="None",markersize=5,color="blue",label=r"LinearRegression")
plt.xticks(rotation=90)
plt.legend()
plt.show()
```




```
# Elasticnet
```

In [26]:

```
from sklearn.linear_model import ElasticNet

```

In [27]:

```
a=ElasticNet()
a.fit(x,y)
print(a.coef_)
print(a.intercept_)


```

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

```
# calculating the error rate
```

In [29]:

```
y_pred_elastic=a.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
94682466.88215785
```

```
# logistic regression
```

In [31]:

```
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
```

```
# Reading the dataFrame
```

In [32]:

```
df=pd.read_csv(r"C:\Users\MY HOME\Desktop\datascience\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

```
# performing basic pre-processing steps
```

In [33]:

```
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
```

df.describe()

In [34]:

```
df.describe()
```

Out[34]:

	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 [35]:

```
df.shape
```

Out[35]:

```
(1338, 7)
```

```
# changing the column name
```

In [36]:

```
df=df[["sex","smoker"]]
df.columns=["sex","smoker"]
```

In [37]:

```
df.head()
```

Out[37]:

	sex	smoker
0	female	yes
1	male	no
2	male	no
3	male	no
4	male	no

In [38]:

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

Out[38]:

	sex	smoker
0	0	yes
1	1	no
2	1	no
3	1	no
4	1	no
...	...	...
1333	1	no
1334	0	no
1335	0	no
1336	0	no
1337	0	yes

1338 rows × 2 columns

In [39]:

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

Out[39]:

	sex	smoker
0	0	1
1	1	0
2	1	0
3	1	0
4	1	0
...	...	...
1333	1	0
1334	0	0
1335	0	0
1336	0	0
1337	0	1

1338 rows × 2 columns

Building the model

In [40]:

```
features_matrix=df.iloc[:,0:2]
target_vector=df.iloc[:,-1]
```

In [41]:

```
print('The target matrix has %d rows and %d column(S)'%(np.array(target_vector).reshape(-1,1).shape))

```

```
The target matrix has 1338 rows and 1 column(S)
```

In [42]:

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

In [43]:

```
algorithm = LogisticRegression(penalty=None,dual=False,tol=1e-1,C= 1.0,fit_intercept=True,intercept_scaling=1,class_
```

In [44]:

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

In [45]:

```
observation=[[0,1]]
```

In [46]:

```
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 [47]:

```
print(" " "The Model says the probability of the observation we passed belonging to class['0'] Is %s" " "%(algorithm
print()
```

```
 The Model says the probability of the observation we passed belonging to class['0'] Is 0.104736832089055
49
```

In [48]:

```
print(" " "The Model says the probability of the observation we passed belonging to class['1'] Is %s" " "%(algorithm
```

```
 The Model says the probability of the observation we passed belonging to class['1'] Is 0.895263167910944
5
```

```
# Decision tree
```

In [50]:

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

```
# reading the DataFrame
```

In [51]:

```
df=pd.read_csv(r"C:\Users\MY HOME\Desktop\datascience\insurance.csv")
df
```

Out[51]:

	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 [52]:

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

Out[52]:

```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [53]:

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

Out[53]:

	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 [54]:

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

Out[54]:

	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 [55]:

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

In [56]:

```
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[56]:

```
((669, 5), (669, 5))
```

```
# Building a model
```

In [57]:

```
s=DecisionTreeClassifier(random_state=20)
s.fit(x_train,y_train)
score=s.score(x_test,y_test)
print(score)
```

```
0.8041853512705530
```

```
C:\Users\MY HOME\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:439: UserWarni
ng: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

```
# Random Forest#
```

In [59]:

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

```

```
# Reading the DataFrame
```

In [60]:

```
df=pd.read_csv(r"C:\Users\MY HOME\Desktop\datascience\insurance.csv")
df
```

Out[60]:

	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 [61]:

```
df.describe()
```

Out[61]:

	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 [62]:

```
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 [63]:

```
df.isna().any()
```

Out[63]:

```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [64]:

```
df.shape
```

Out[64]:

```
(1338, 7)
```

In [65]:

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

Out[65]:

	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 [66]:

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

Out[66]:

```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [67]:

```
r={"region":{"southeast":0,"southwest":1,"northeast":2,"northwest":3}}
df=df.replace(r)
df
```

Out[67]:

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

1338 rows × 7 columns

In [68]:

```
x=df.drop("smoker",axis=1)
y=df["smoker"]
```

```
# Building a model
```

In [69]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

In [70]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [71]:

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

Out[71]:

```
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 [72]:

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

In [73]:

```
from sklearn.model_selection import GridSearchCV
```

In [78]:

```
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))

```

```
0.9491778774289985
```

In [80]:

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

```
RandomForestClassifier(max_depth=85, min_samples_leaf=4, n_estimators=42)
```

In [81]:

```
from sklearn.tree import plot_tree
```

In [86]:

```
plt.figure(figsize=(80,60))
plot_tree(p.estimators_[5],feature_names=x.columns,class_names=["0","1"],filled=True)
```

Out[86]:

```
[Text(0.48, 0.9444444444444444, 'age <= 63.5\ngini = 0.313\nsamples = 417\nvalue = [539, 130]\nclass =
0'),
 Text(0.4, 0.8333333333333334, 'charges <= 15374.252\ngini = 0.308\nsamples = 410\nvalue = [533, 125]\ncl
ass = 0'),
 Text(0.16, 0.7222222222222222, 'region <= 1.5\ngini = 0.004\nsamples = 305\nvalue = [498, 1]\nclass =
0'),
 Text(0.08, 0.6111111111111112, 'gini = 0.0\nsamples = 158\nvalue = [261, 0]\nclass = 0'),
 Text(0.24, 0.6111111111111112, 'charges <= 14201.54\ngini = 0.008\nsamples = 147\nvalue = [237, 1]\nclas
s = 0'),
 Text(0.16, 0.5, 'gini = 0.0\nsamples = 143\nvalue = [234, 0]\nclass = 0'),
 Text(0.32, 0.5, 'gini = 0.375\nsamples = 4\nvalue = [3, 1]\nclass = 0'),
 Text(0.64, 0.7222222222222222, 'charges <= 33473.895\ngini = 0.343\nsamples = 105\nvalue = [35, 124]\ncl
ass = 1'),
 Text(0.56, 0.6111111111111112, 'bmi <= 30.305\ngini = 0.451\nsamples = 65\nvalue = [35, 67]\nclass =
1'),
 Text(0.48, 0.5, 'age <= 32.5\ngini = 0.344\nsamples = 53\nvalue = [19, 67]\nclass = 1'),
 Text(0.28, 0.3888888888888889, 'charges <= 20764.42\ngini = 0.245\nsamples = 26\nvalue = [6, 36]\nclass
= 1'),
 Text(0.2, 0.2777777777777778, 'children <= 1.5\ngini = 0.056\nsamples = 21\nvalue = [1, 34]\nclass =
1'),
 Text(0.12, 0.16666666666666666, 'gini = 0.0\nsamples = 15\nvalue = [0, 27]\nclass = 1'),
 Text(0.28, 0.16666666666666666, 'gini = 0.219\nsamples = 6\nvalue = [1, 7]\nclass = 1'),
 Text(0.36, 0.2777777777777778, 'gini = 0.408\nsamples = 5\nvalue = [5, 2]\nclass = 0'),
 Text(0.68, 0.3888888888888889, 'children <= 1.5\ngini = 0.416\nsamples = 27\nvalue = [13, 31]\nclass =
1'),
 Text(0.52, 0.2777777777777778, 'age <= 48.0\ngini = 0.473\nsamples = 15\nvalue = [10, 16]\nclass = 1'),
 Text(0.44, 0.16666666666666666, 'region <= 1.5\ngini = 0.48\nsamples = 8\nvalue = [9, 6]\nclass = 0'),
 Text(0.36, 0.05555555555555555, 'gini = 0.49\nsamples = 4\nvalue = [3, 4]\nclass = 1'),
 Text(0.52, 0.05555555555555555, 'gini = 0.375\nsamples = 4\nvalue = [6, 2]\nclass = 0'),
 Text(0.6, 0.16666666666666666, 'gini = 0.165\nsamples = 7\nvalue = [1, 10]\nclass = 1'),
 Text(0.84, 0.2777777777777778, 'age <= 46.5\ngini = 0.278\nsamples = 12\nvalue = [3, 15]\nclass = 1'),
 Text(0.76, 0.16666666666666666, 'bmi <= 23.658\ngini = 0.165\nsamples = 8\nvalue = [1, 10]\nclass = 1'),
 Text(0.68, 0.05555555555555555, 'gini = 0.278\nsamples = 4\nvalue = [1, 5]\nclass = 1'),
 Text(0.84, 0.05555555555555555, 'gini = 0.0\nsamples = 4\nvalue = [0, 5]\nclass = 1'),
 Text(0.92, 0.16666666666666666, 'gini = 0.408\nsamples = 4\nvalue = [2, 5]\nclass = 1'),
 Text(0.64, 0.5, 'gini = 0.0\nsamples = 12\nvalue = [16, 0]\nclass = 0'),
 Text(0.72, 0.6111111111111112, 'gini = 0.0\nsamples = 40\nvalue = [0, 57]\nclass = 1'),
 Text(0.56, 0.8333333333333334, 'gini = 0.496\nsamples = 7\nvalue = [6, 5]\nclass = 0')]
```

```
1 p.feature_importances_
```

Out[88]:

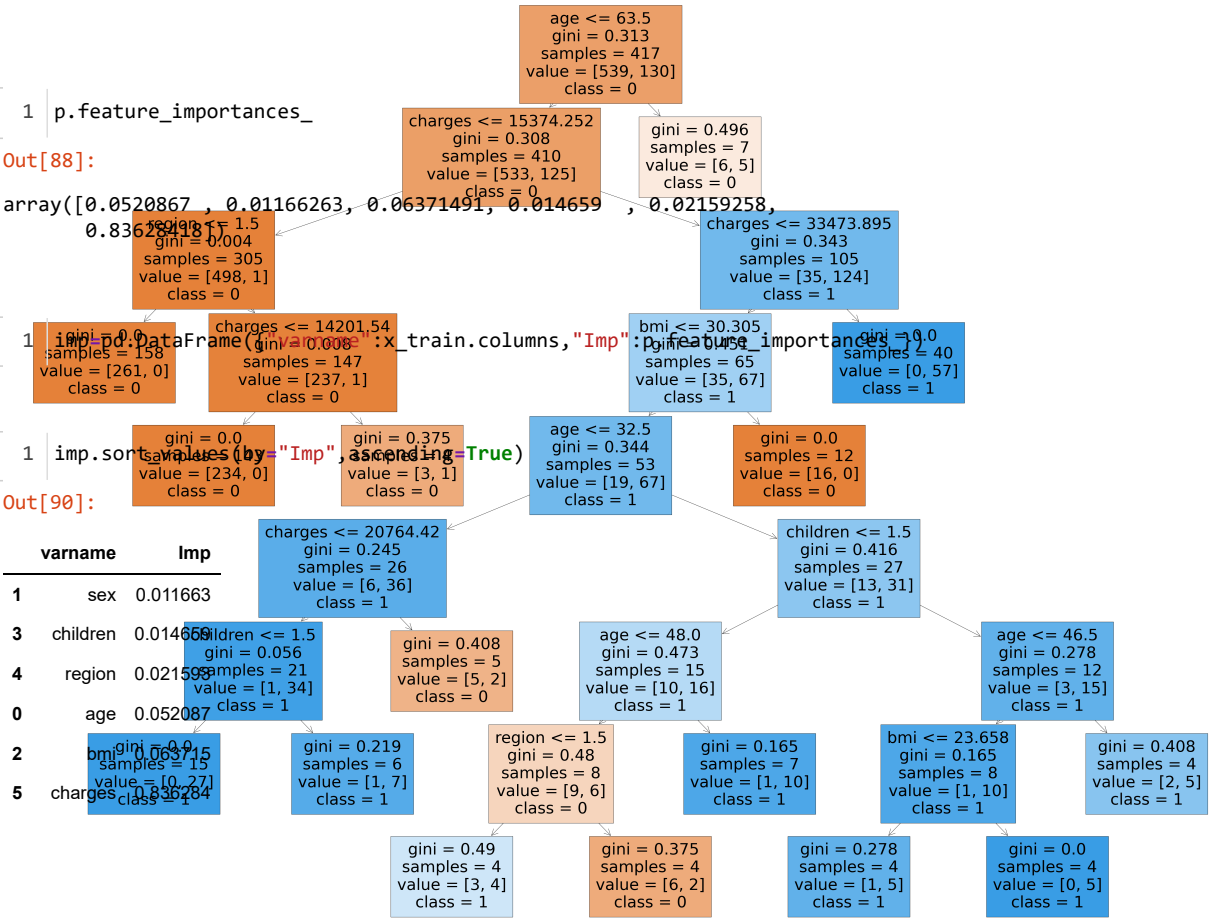
```
array([0.0520867 , 0.01166263, 0.06371491, 0.014659  , 0.02159258,
       0.83628418])
```

```
1 imp=pd.DataFrame({"varname":x_train.columns,"Imp":p.feature_importances_})
```

```
1 imp.sort_values(by="Imp",ascending=True)
```

Out[90]:

	varname	Imp
1	sex	0.011663
3	children	0.014659
4	region	0.021593
0	age	0.052087
2	bmi	0.063715
5	charges	0.836284



In [ ]:

```
1 
2 
3 
4 
```

```
e main intention of this conclusion is:
om the above models like:Linearregression,logisticRegression,randonForest,Descision tree and ridge and lassoregressions
have concluded that randomforest is the best model...it is best fit one.....and the accuracy of that model is 94%.
```