Mini project-1

Problem Statement:Which model is suitable best for health insurance

Step1:Data collection

In [1]:
```
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
from sklearn.linear_model import LogisticRegression
```

Reading the data

In [2]:
```
df=pd.read_csv(r"C:\Users\LENOVO\Downloads\insurance (1).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

Data cleaning and Preprocessing

In [3]:
```
df.shape
```

Out[3]:
```
(1338, 7)
```

In [4]:
```
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 [5]:
```
df.head()
```

Out[5]:

	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 [6]:
```
df.tail()
```

Out[6]:

	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 [7]:
```
df.describe
```

Out[7]:
```
<bound method NDFrame.describe of       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 x 7 columns]>
```

To find null values

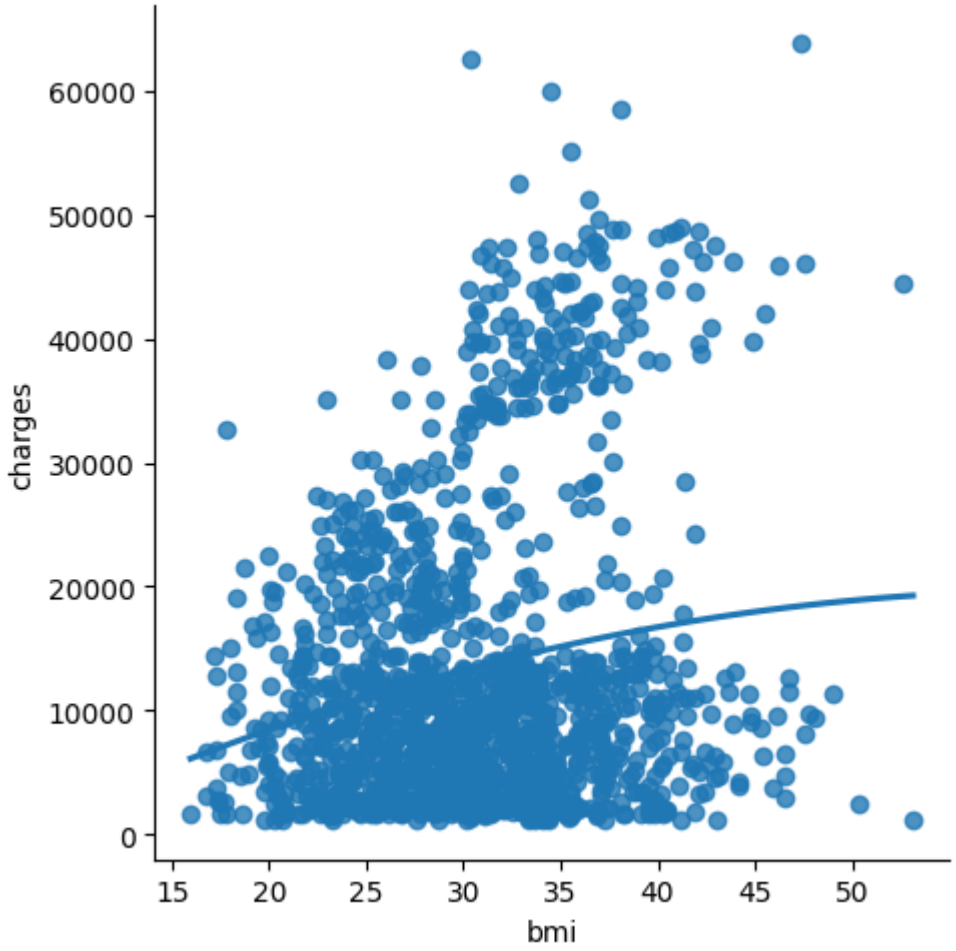
In [8]:
```
df.isna().sum()
```

Out[8]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

visulization

In [9]:
```
sns.lmplot(x="bmi",y="charges",data=df,order=2,ci=None)
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x26a4c054190>



In [10]:
```
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

Feature Selection

In [13]:
```
features=df.columns[0:5]
target=df.columns[-1]
```

In [14]:
```
x=df[features].values
y=df[target].values
```

Linear regression

In [15]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

In [16]:
```
a=LinearRegression()
a.fit(x_train,y_train)
```

Out[16]:
```
▾ LinearRegression
LinearRegression()
```

In [17]:
```
print(a.score(x_test,y_test))
```

```
0.7562715620820689
```

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 0.7405108866062634
The train score for lr model is 0.7562715620820689
```

Ridge

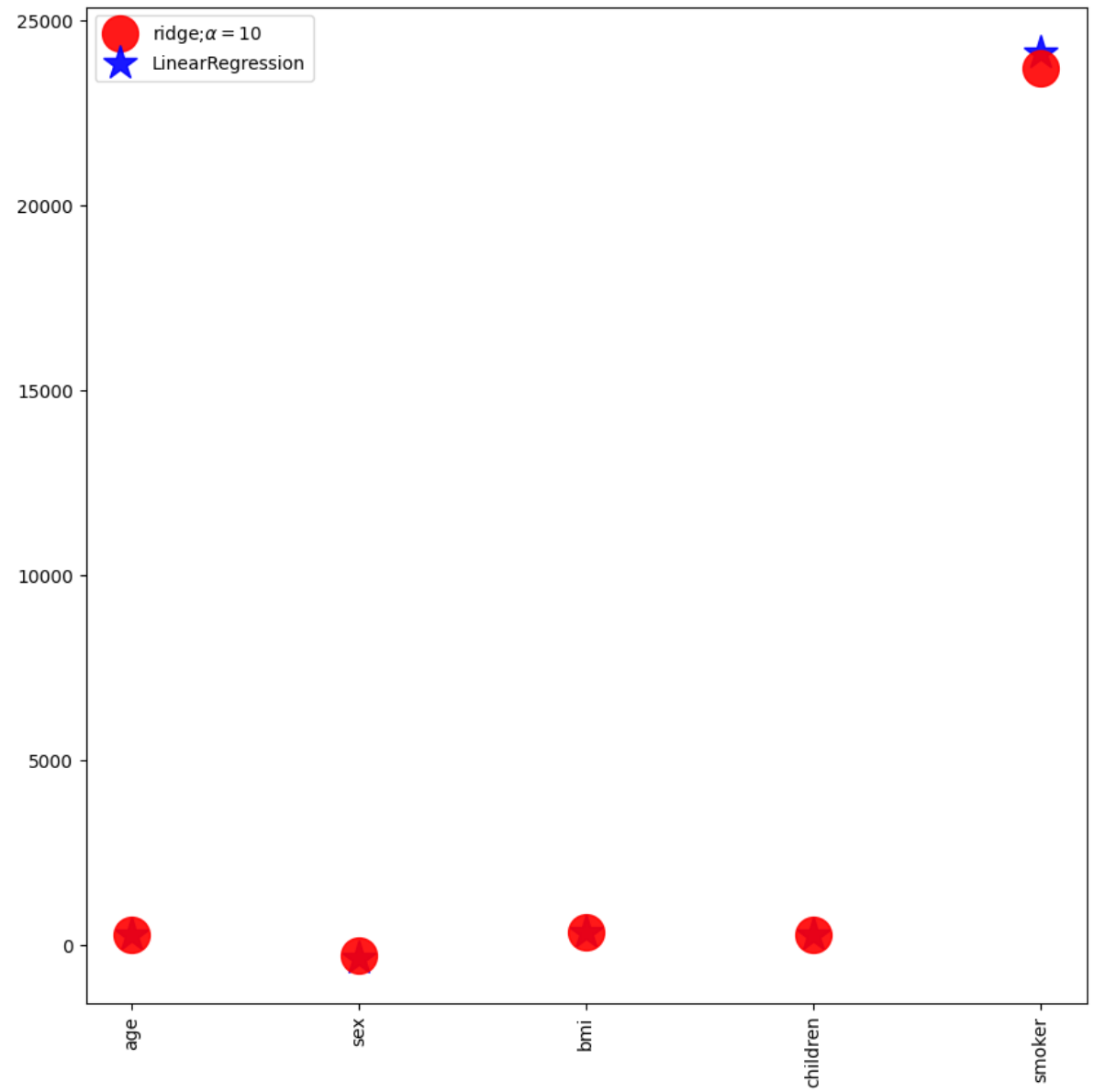
In [19]:
```
#ridge
from sklearn.linear_model import Ridge, RidgeCV, Lasso
```

In [20]:
```
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

0.7403069796189549
0.7567264946359474
```

In [21]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridge.coef_,alpha=0.9,linestyle="None",marker='o',markersize=
plt.plot(features,a.coef_,alpha=0.9,linestyle="None",marker='*',markersize=20,
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

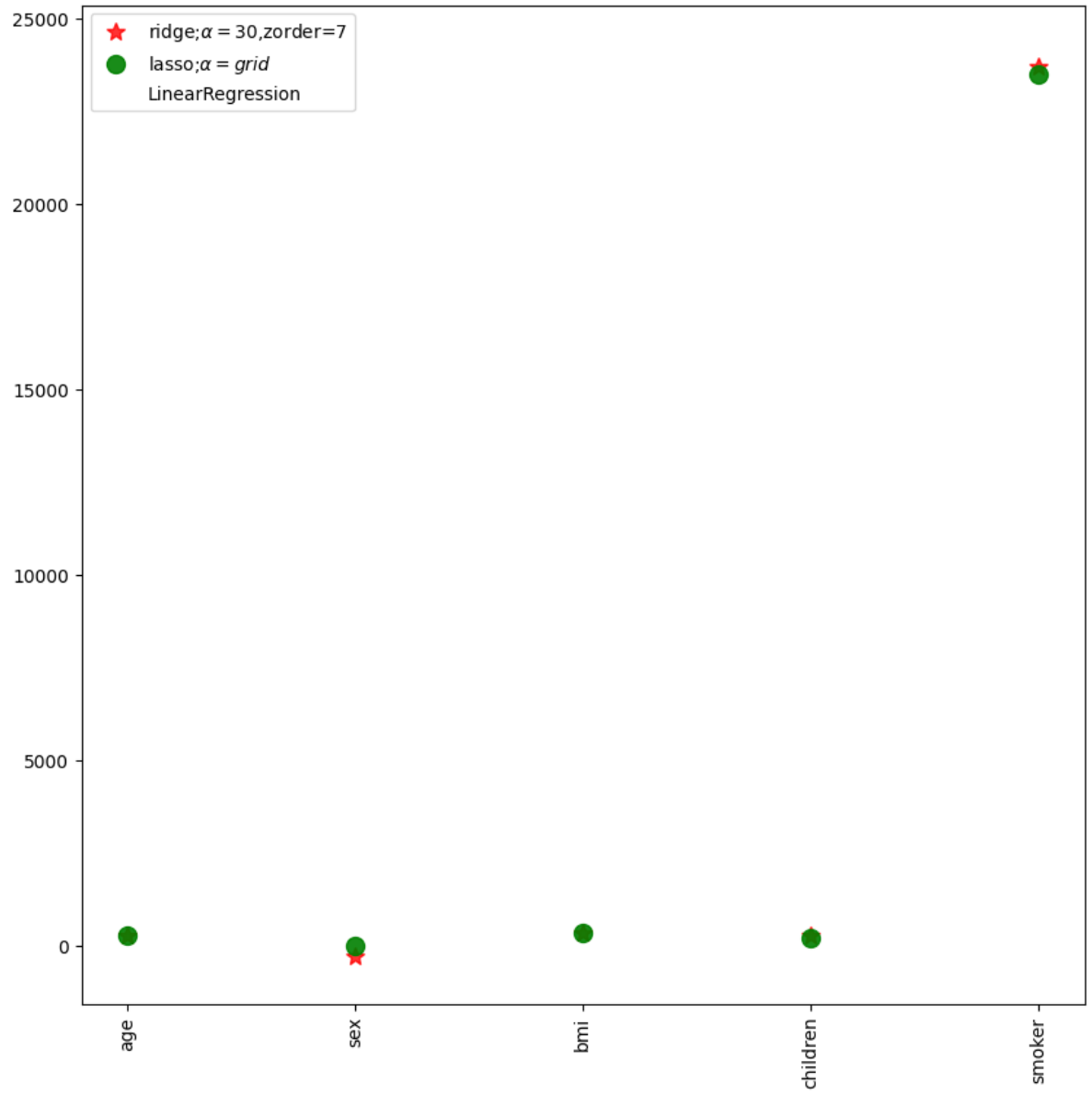


Lasso

In [22]:
```
#lasso
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)
```

```
0.7398882107726026
0.7565576894176124
```

In [23]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridge.coef_,alpha=0.8,marker="*",markersize=10,linestyle="no
plt.plot(lasso.coef_,alpha=0.9,marker='o',markersize=10,linestyle="none",color
plt.plot(features,a.coef_,alpha=0.7,linestyle="None",markersize=5,color="blue"
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



Elastic Net

In [24]:
```
from sklearn.linear_model import ElasticNet
```

In [25]:
```
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 [26]:
```
y_pred_elastic=a.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
94070508.40759112
```

Logistic Regression

In [27]:
```
#Logistic regression
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 [28]:
```
df=pd.read_csv(r"C:\Users\LENOVO\Downloads\insurance (1).csv")
df
```

Out[28]:

	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 [29]:
```
df=df[["sex","smoker"]]
df.columns=["sex","smoker"]
```

In [30]:
```
sex={"sex":{"female":0,"male":1}}
df=df.replace(sex)
df
```

Out[30]:

	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 [31]:
```
smoker={"smoker":{"no":0,"yes":1}}
df=df.replace(smoker)
df
```

Out[31]:

	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

In [32]:
```
features_matrix=df.iloc[:,0:2]
target_vector=df.iloc[:,-1]
```

In [33]:
```
print('The target matrix has %d rows and %d column(S)'%(np.array(target_vector
```

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

In [34]:
```
features_matrix_Standardized=StandardScaler().fit_transform(features_matrix)
```

In [35]:
```
algorithm = LogisticRegression(penalty=None,dual=False,tol=1e-1,C= 1.0,fit_int
```

In [36]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_Standardized,target_ve
```

In [37]:
```
observation=[[0,1]]
```

In [38]:
```
print('The algoritham was trained to predict one of the two classes:%s'%(algor
```

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

In [39]:
```
print(" " "The Model says the probability of the observation we passed belongi
print()
```

```
 The Model says the probability of the observation we passed belonging to cla
ss['0'] Is 0.10473683208905549
```

In [40]:
```
print(" " "The Model says the probability of the observation we passed belongi
```

```
 The Model says the probability of the observation we passed belonging to cla
ss['1'] Is 0.8952631679109445
```

decision tree

In [41]:
```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

In [42]:
```
df=pd.read_csv(r"C:\Users\LENOVO\Downloads\insurance (1).csv")
df
```

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
...	...	...	...	...	...	...	...
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 [43]:
```
df["region"].value_counts()
```

Out[43]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [44]:
```
convert={"sex":{"female":0,"male":1}}
df=df.replace(convert)
df
```

Out[44]:

	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 [45]:
```
x=["age","sex","children","bmi","charges"]
y=["0","1"]
all_inputs=df[x]
all_classes=df["smoker"]
```

In [46]:
```
x_train,X_test,y_train,y_test=train_test_split(all_inputs,all_classes,test_siz
x_train.shape,x_test.shape
```

Out[46]:
```
((669, 5), (669, 5))
```

In [47]:
```
s=DecisionTreeClassifier(random_state=15)
s.fit(x_train,y_train)
score=s.score(x_test,y_test)
print(score)
```

```
0.7817638266068759

C:\Users\LENOVO\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\base.py:439: UserWarning: X does not have valid feature names, but Decis
ionTreeClassifier was fitted with feature names
  warnings.warn(
```

random forest

In [48]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
```

In [49]:
```
df=pd.read_csv(r"C:\Users\LENOVO\Downloads\insurance (1).csv")
df
```

Out[49]:

	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 [50]:
```
convert={"sex":{"female":0,"male":1}}
df=df.replace(convert)
df
```

Out[50]:

	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 [51]:
```
r={"region":{"southeast":0,"southwest":1,"northeast":2,"northwest":3}}
df=df.replace(r)
df
```

Out[51]:

	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 [52]:
```
x=df.drop("smoker",axis=1)
y=df["smoker"]
```

In [53]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

In [54]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [55]:
```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

Out[55]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [56]:
```
params={"max_depth":[1,23,4,56,85],"min_samples_leaf":[4,6,8,10,12],"n_estimat
```

In [57]:
```
from sklearn.model_selection import GridSearchCV
```

In [58]:
```
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
```

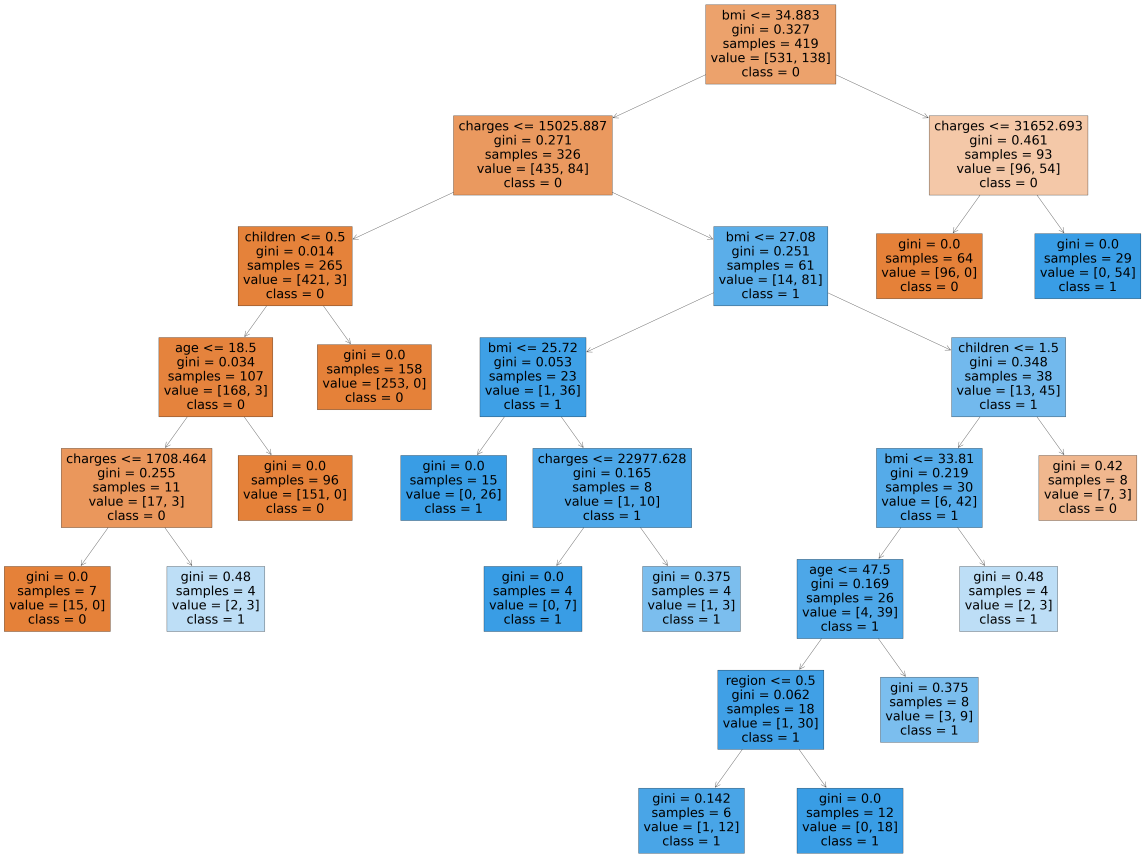
In [59]:
```
p=grid_search.best_estimator_
print(p)
```

```
RandomForestClassifier(max_depth=23, min_samples_leaf=4, n_estimators=42)
```

In [60]:
```
from sklearn.tree import plot_tree
```

In [61]:
```
plt.figure(figsize=(80,60))
plot_tree(p.estimators_[5],feature_names=x.columns,class_names=["0","1"],fille
```

Out[61]:
```
[Text(0.6666666666666666, 0.9375, 'bmi <= 34.883\ngini = 0.327\nsamples = 419
\nvalue = [531, 138]\nclass = 0'),
 Text(0.4666666666666667, 0.8125, 'charges <= 15025.887\ngini = 0.271\nsample
s = 326\nvalue = [435, 84]\nclass = 0'),
 Text(0.26666666666666666, 0.6875, 'children <= 0.5\ngini = 0.014\nsamples =
265\nvalue = [421, 3]\nclass = 0'),
 Text(0.2, 0.5625, 'age <= 18.5\ngini = 0.034\nsamples = 107\nvalue = [168,
3]\nclass = 0'),
 Text(0.13333333333333333, 0.4375, 'charges <= 1708.464\ngini = 0.255\nsample
s = 11\nvalue = [17, 3]\nclass = 0'),
 Text(0.06666666666666667, 0.3125, 'gini = 0.0\nsamples = 7\nvalue = [15, 0]
\nclass = 0'),
 Text(0.2, 0.3125, 'gini = 0.48\nsamples = 4\nvalue = [2, 3]\nclass = 1'),
 Text(0.26666666666666666, 0.4375, 'gini = 0.0\nsamples = 96\nvalue = [151,
0]\nclass = 0'),
 Text(0.3333333333333333, 0.5625, 'gini = 0.0\nsamples = 158\nvalue = [253,
0]\nclass = 0'),
 Text(0.6666666666666666, 0.6875, 'bmi <= 27.08\ngini = 0.251\nsamples = 61\n
value = [14, 81]\nclass = 1'),
 Text(0.4666666666666667, 0.5625, 'bmi <= 25.72\ngini = 0.053\nsamples = 23\n
value = [1, 36]\nclass = 1'),
 Text(0.4, 0.4375, 'gini = 0.0\nsamples = 15\nvalue = [0, 26]\nclass = 1'),
 Text(0.5333333333333333, 0.4375, 'charges <= 22977.628\ngini = 0.165\nsample
s = 8\nvalue = [1, 10]\nclass = 1'),
 Text(0.4666666666666667, 0.3125, 'gini = 0.0\nsamples = 4\nvalue = [0, 7]\nc
lass = 1'),
 Text(0.6, 0.3125, 'gini = 0.375\nsamples = 4\nvalue = [1, 3]\nclass = 1'),
 Text(0.8666666666666667, 0.5625, 'children <= 1.5\ngini = 0.348\nsamples = 3
8\nvalue = [13, 45]\nclass = 1'),
 Text(0.8, 0.4375, 'bmi <= 33.81\ngini = 0.219\nsamples = 30\nvalue = [6, 42]
\nclass = 1'),
 Text(0.7333333333333333, 0.3125, 'age <= 47.5\ngini = 0.169\nsamples = 26\nv
alue = [4, 39]\nclass = 1'),
 Text(0.6666666666666666, 0.1875, 'region <= 0.5\ngini = 0.062\nsamples = 18
\nvalue = [1, 30]\nclass = 1'),
 Text(0.6, 0.0625, 'gini = 0.142\nsamples = 6\nvalue = [1, 12]\nclass = 1'),
 Text(0.7333333333333333, 0.0625, 'gini = 0.0\nsamples = 12\nvalue = [0, 18]
\nclass = 1'),
 Text(0.8, 0.1875, 'gini = 0.375\nsamples = 8\nvalue = [3, 9]\nclass = 1'),
 Text(0.8666666666666667, 0.3125, 'gini = 0.48\nsamples = 4\nvalue = [2, 3]\n
class = 1'),
 Text(0.9333333333333333, 0.4375, 'gini = 0.42\nsamples = 8\nvalue = [7, 3]\n
class = 0'),
 Text(0.8666666666666667, 0.8125, 'charges <= 31652.693\ngini = 0.461\nsample
s = 93\nvalue = [96, 54]\nclass = 0'),
 Text(0.8, 0.6875, 'gini = 0.0\nsamples = 64\nvalue = [96, 0]\nclass = 0'),
 Text(0.9333333333333333, 0.6875, 'gini = 0.0\nsamples = 29\nvalue = [0, 54]
\nclass = 1')]
```



In [62]:
```
p.feature_importances_
```

Out[62]:
```
array([0.03607984, 0.00443573, 0.06696598, 0.02095128, 0.01142237,
       0.86014479])
```

In [63]:
```
imp=pd.DataFrame({"varname":x_train.columns,"Imp":p.feature_importances_})
```

conclusion

For the above Dataset we use different Types of models ,for each and every model we get differentTypes of Accuracies.Based on that accuracies we can conclude which model is best fit for my ourDataset.

Here we got different types accuracies,For linear regression we obtained 75% accuracy,For Logistic regression we obtained 89% accuracy,For Random forest we obtained 94%

accuracy,For decision tree we obtained 78% acuracy.From all the observations we can conclude that Random Forest model is Best fit

In [ ]: