Problem Statement:Which model is suitable for given dataset

Importing Packages

In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

Data Collection

In [4]:

```
df=pd.read_csv(r"C:\Users\sowmika\Downloads\insurance.csv")
df
```

Out[4]:

	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 [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.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 [8]:

```
df.describe()
```

Out[8]:

	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 [9]:

```
df.columns
```

Out[9]:

```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dt
ype='object')
```

In [10]:

```
df.shape
```

Out[10]:

```
(1338, 7)
```

Duplicate Values

In [11]:

```
df.isnull().sum()
```

Out[11]:

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

Data Visualization

In [12]:

```
df['smoker'].value_counts()
```

Out[12]:

```
smoker
no      1064
yes      274
Name: count, dtype: int64
```

In [13]:

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

Out[13]:

```
sex
male      676
female    662
Name: count, dtype: int64
```

In [14]:

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

Out[14]:

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

In [15]:

```
s=pd.crosstab(df['smoker'],df['sex'])
print(s)
```

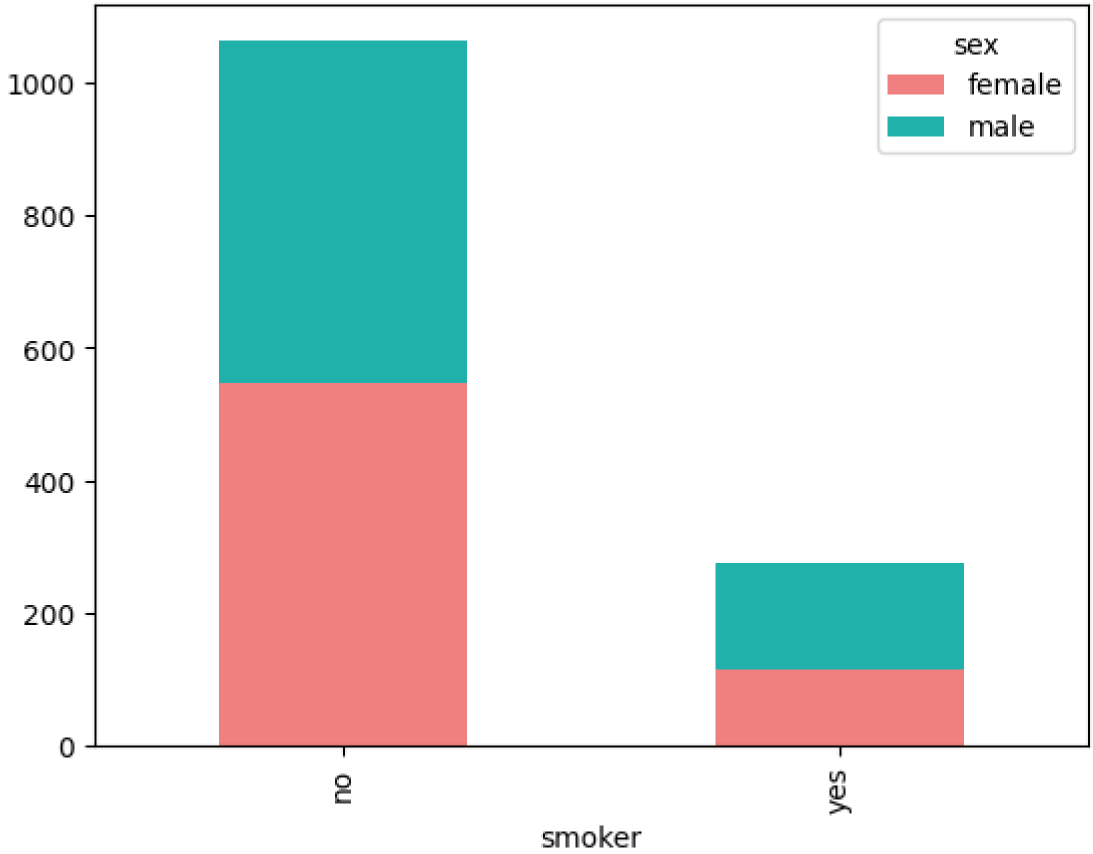
```
sex     female  male
smoker
no         547   517
yes        115   159
```

In [16]:

```
s.plot(kind='bar', stacked=True, color=['lightcoral','LightSeaGreen'],grid=False)
```

Out[16]:

```
<Axes: xlabel='smoker'>
```

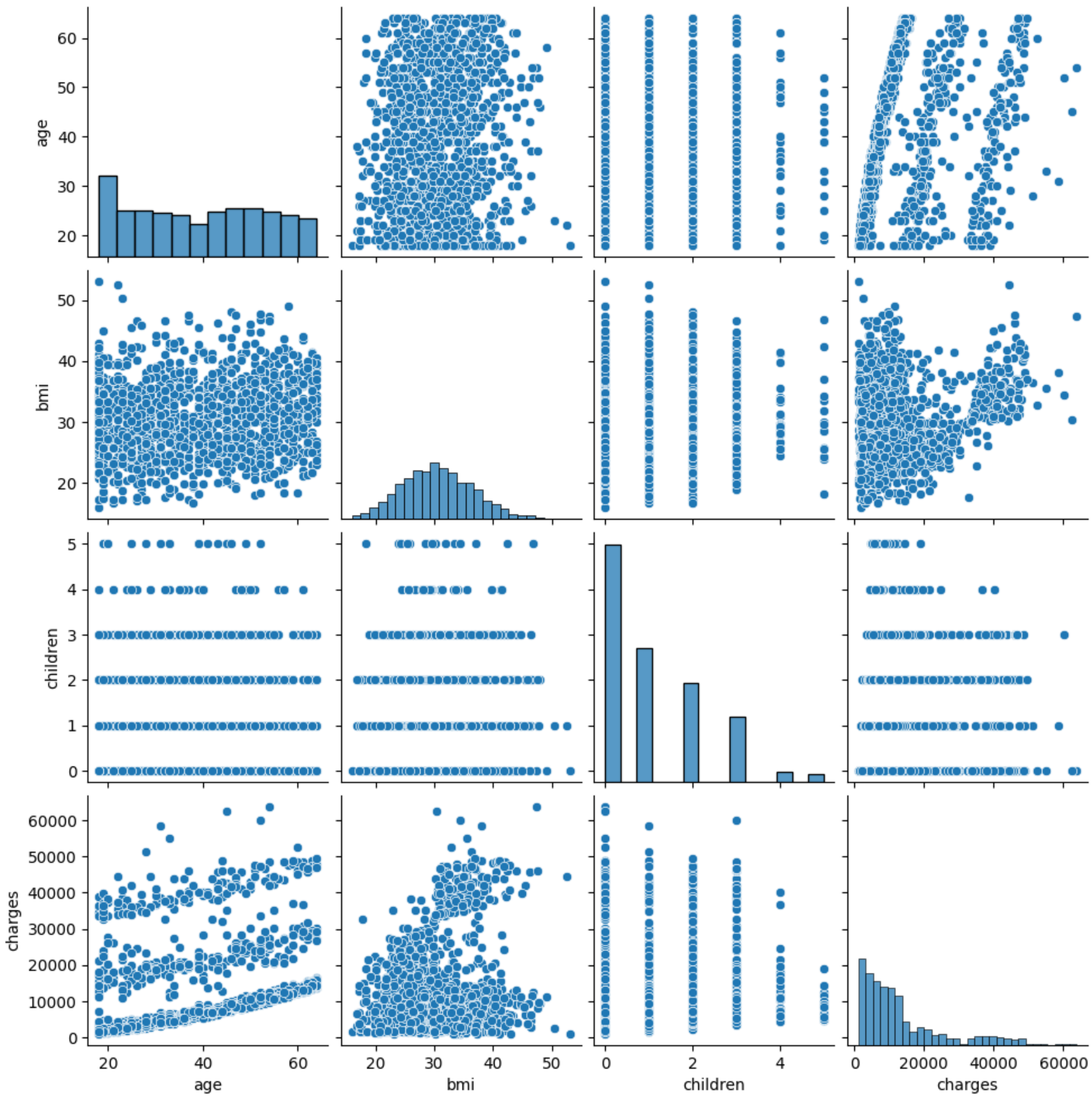


In [17]:

```
sns.pairplot(df)
```

Out[17]:

```
<seaborn.axisgrid.PairGrid at 0x1f9fc738f40>
```



In [18]:

```
c=pd.crosstab(df['age'],df['sex'])
print(c)
```

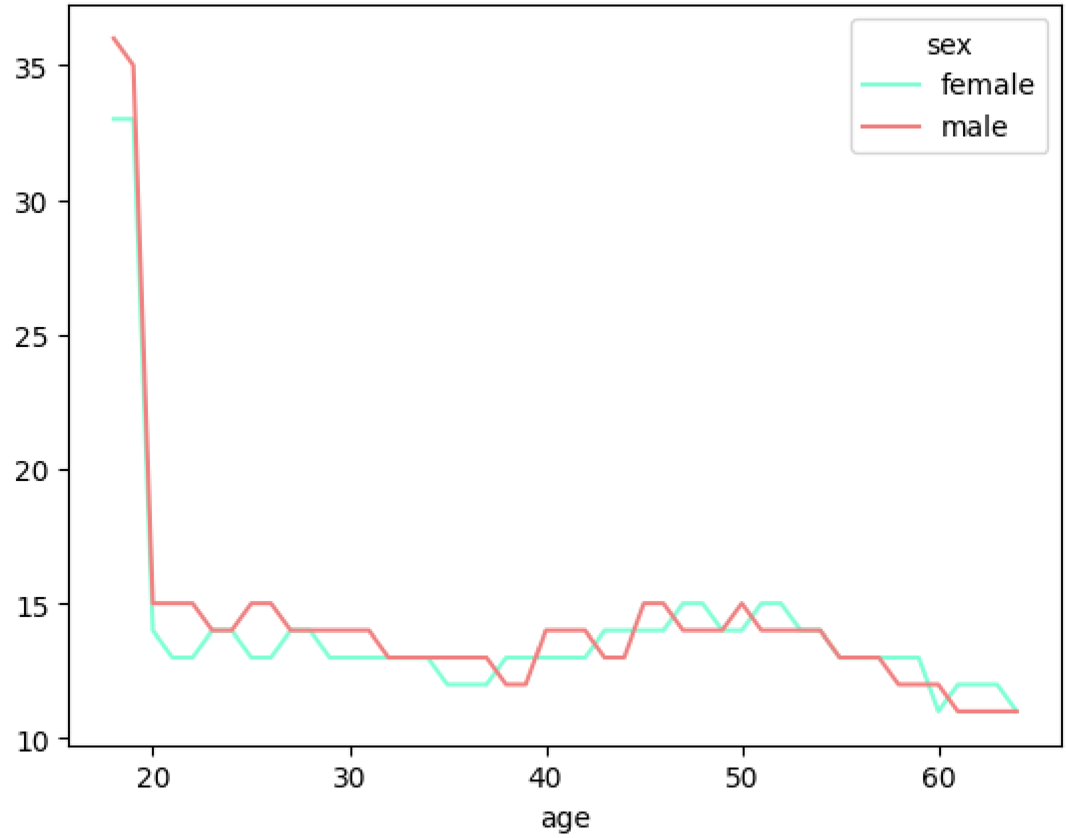
```
sex  female  male
age
18       33    36
19       33    35
20       14    15
21       13    15
22       13    15
23       14    14
24       14    14
25       13    15
26       13    15
27       14    14
28       14    14
29       13    14
30       13    14
31       13    14
32       13    13
33       13    13
34       13    13
35       12    13
36       12    13
37       12    13
38       13    12
39       13    12
40       13    14
41       13    14
42       13    14
43       14    13
44       14    13
45       14    15
46       14    15
47       15    14
48       15    14
49       14    14
50       14    15
51       15    14
52       15    14
53       14    14
54       14    14
55       13    13
56       13    13
57       13    13
58       13    12
59       13    12
60       11    12
61       12    11
62       12    11
63       12    11
64       11    11
```

In [19]:

```
c.plot(kind='line', stacked=False, color=['aquamarine','lightcoral'],grid=False)
```

Out[19]:

```
<Axes: xlabel='age'>
```



In [20]:

```
s = {'region':{'northeast':1,'northwest':2,'southwest':3,'southeast':4}}
df = df.replace(s)
print(df)
```

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

[1338 rows x 7 columns]
```

In [21]:

```
S = {'sex':{'female':1,'male':2}}
df =df.replace(S)
print(df)
```

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

[1338 rows x 7 columns]
```

In [22]:

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

Linear Regression

In [25]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=40)
x_train.shape,x_test.shape
```

Out[25]:

```
((936, 6), (402, 6))
```

Logistic Regression

In [27]:

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

```
0.9378109452736318
```

Decision Tree

In [28]:

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0)
clf.fit(x_train,y_train)
score = clf.score(x_test,y_test)
print(score)
```

0.9601990049751243

RandomForestClassifier

In [29]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
print(rfc.score(x_test,y_test))
```

0.9626865671641791

In [30]:

```
params={'max_depth':[2,5,10,20,25],'min_samples_leaf':[5,20,30,50,100,200],'n_estimators
```

In [31]:

```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring='accuracy')
grid_search.fit(x_train,y_train)
```

Out[31]:

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [32]:

```
grid_search.best_score_
```

Out[32]:

0.9497863247863247

In [33]:

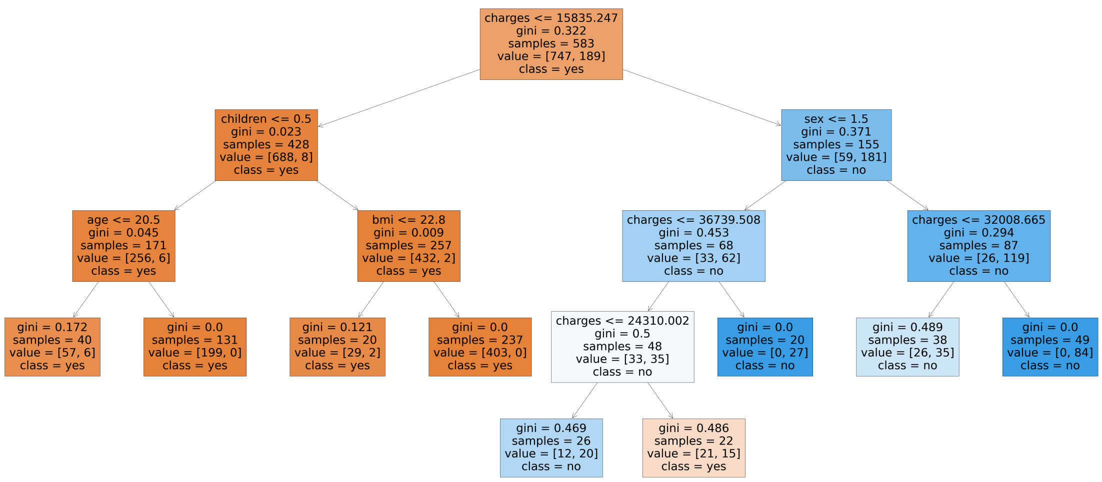
```
rfc_best = grid_search.best_estimator_
```

In [34]:

```
from sklearn.tree import plot_tree
plt.figure(figsize = (90,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['yes','no'],fille
```

Out[34]:

```
[Text(0.5, 0.9, 'charges <= 15835.247\ngini = 0.322\nsamples = 583\nvalue
= [747, 189]\nclass = yes'),
 Text(0.25, 0.7, 'children <= 0.5\ngini = 0.023\nsamples = 428\nvalue = [6
88, 8]\nclass = yes'),
 Text(0.125, 0.5, 'age <= 20.5\ngini = 0.045\nsamples = 171\nvalue = [256,
6]\nclass = yes'),
 Text(0.0625, 0.3, 'gini = 0.172\nsamples = 40\nvalue = [57, 6]\nclass = y
es'),
 Text(0.1875, 0.3, 'gini = 0.0\nsamples = 131\nvalue = [199, 0]\nclass = y
es'),
 Text(0.375, 0.5, 'bmi <= 22.8\ngini = 0.009\nsamples = 257\nvalue = [432,
2]\nclass = yes'),
 Text(0.3125, 0.3, 'gini = 0.121\nsamples = 20\nvalue = [29, 2]\nclass = y
es'),
 Text(0.4375, 0.3, 'gini = 0.0\nsamples = 237\nvalue = [403, 0]\nclass = y
es'),
 Text(0.75, 0.7, 'sex <= 1.5\ngini = 0.371\nsamples = 155\nvalue = [59, 18
1]\nclass = no'),
 Text(0.625, 0.5, 'charges <= 36739.508\ngini = 0.453\nsamples = 68\nvalue
= [33, 62]\nclass = no'),
 Text(0.5625, 0.3, 'charges <= 24310.002\ngini = 0.5\nsamples = 48\nvalue
= [33, 35]\nclass = no'),
 Text(0.5, 0.1, 'gini = 0.469\nsamples = 26\nvalue = [12, 20]\nclass = n
o'),
 Text(0.625, 0.1, 'gini = 0.486\nsamples = 22\nvalue = [21, 15]\nclass = y
es'),
 Text(0.6875, 0.3, 'gini = 0.0\nsamples = 20\nvalue = [0, 27]\nclass = n
o'),
 Text(0.875, 0.5, 'charges <= 32008.665\ngini = 0.294\nsamples = 87\nvalue
= [26, 119]\nclass = no'),
 Text(0.8125, 0.3, 'gini = 0.489\nsamples = 38\nvalue = [26, 35]\nclass =
no'),
 Text(0.9375, 0.3, 'gini = 0.0\nsamples = 49\nvalue = [0, 84]\nclass = n
o')]
```

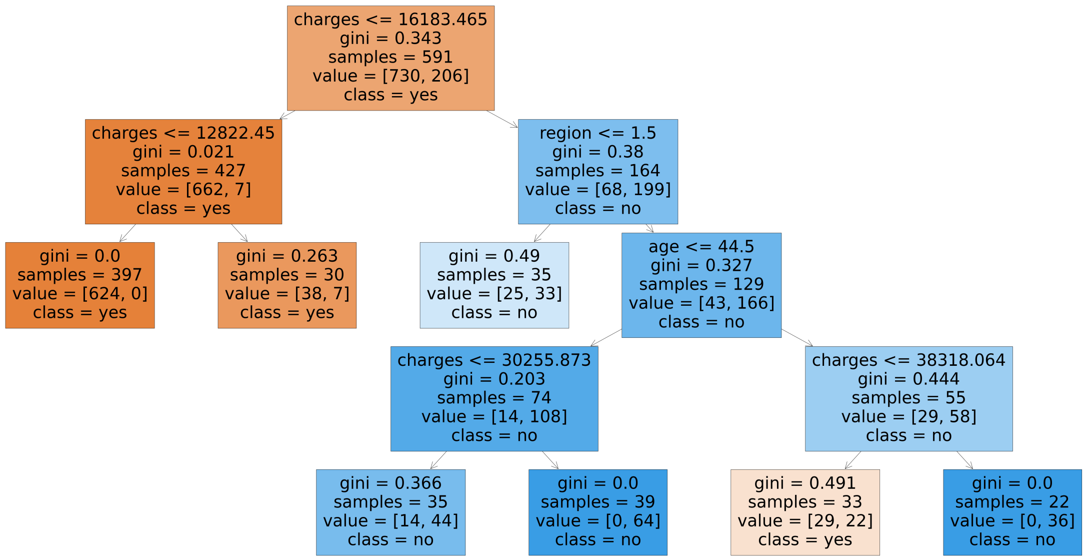


In [35]:

```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[7],feature_names=x.columns,class_names=['yes','no'],fille
```

Out[35]:

```
[Text(0.36363636363636365, 0.9, 'charges <= 16183.465\ngini = 0.343\nsampl
es = 591\nvalue = [730, 206]\nclass = yes'),
 Text(0.18181818181818182, 0.7, 'charges <= 12822.45\ngini = 0.021\nsample
s = 427\nvalue = [662, 7]\nclass = yes'),
 Text(0.09090909090909091, 0.5, 'gini = 0.0\nsamples = 397\nvalue = [624,
0]\nclass = yes'),
 Text(0.2727272727272727, 0.5, 'gini = 0.263\nsamples = 30\nvalue = [38,
7]\nclass = yes'),
 Text(0.5454545454545454, 0.7, 'region <= 1.5\ngini = 0.38\nsamples = 164
\nvalue = [68, 199]\nclass = no'),
 Text(0.45454545454545453, 0.5, 'gini = 0.49\nsamples = 35\nvalue = [25, 3
3]\nclass = no'),
 Text(0.6363636363636364, 0.5, 'age <= 44.5\ngini = 0.327\nsamples = 129\n
value = [43, 166]\nclass = no'),
 Text(0.45454545454545453, 0.3, 'charges <= 30255.873\ngini = 0.203\nsampl
es = 74\nvalue = [14, 108]\nclass = no'),
 Text(0.36363636363636365, 0.1, 'gini = 0.366\nsamples = 35\nvalue = [14,
44]\nclass = no'),
 Text(0.5454545454545454, 0.1, 'gini = 0.0\nsamples = 39\nvalue = [0, 64]
\nclass = no'),
 Text(0.8181818181818182, 0.3, 'charges <= 38318.064\ngini = 0.444\nsample
s = 55\nvalue = [29, 58]\nclass = no'),
 Text(0.7272727272727273, 0.1, 'gini = 0.491\nsamples = 33\nvalue = [29, 2
2]\nclass = yes'),
 Text(0.9090909090909091, 0.1, 'gini = 0.0\nsamples = 22\nvalue = [0, 36]
\nclass = no')]
```



In [36]:

```
rfc_best.feature_importances_
```

Out[36]:

```
array([0.02957835, 0.00478561, 0.02999727, 0.00681847, 0.00549181,
       0.92332849])
```

In [37]:

```
imp_df = pd.DataFrame({"Varname":x_train.columns,'Imp':rfc_best.feature_importances_})
imp_df.sort_values(by='Imp',ascending=False)
```

Out[37]:

	Varname	Imp
5	charges	0.923328
2	bmi	0.029997
0	age	0.029578
3	children	0.006818
4	region	0.005492
1	sex	0.004786

In [38]:

```
df['bmi'].value_counts()
```

Out[38]:

```
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
```

conclusion:-

```
              Based dataset We conclude that male smoker are high compared to female
smokers.we conclude that
    "Logistc regression" is the best model for the given data set
```

In [39]:

```
import pickle
```

In [40]:

```
filename="Insurance prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [ ]:

```

```