INSURANCE DATASET

Logistic Regression

1.PROBLEM STATEMENT: To predict and analyze the Female and Male Smoker in the region

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

DATA COLLECTION

In [2]:
```
df=pd.read_csv(r"C:\Users\manis\OneDrive\Pictures\Documents\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

DATA CLEANING

In [3]:
```
df.info()
#TO FIND IF THE NULL VALUE ARE HAVE OR NOT
```

```
<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['region'].value_counts()
#TO KNOW THE REGION COUNT VALUE
```

Out[4]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [5]:
```
convert={'sex':{"female":1,"male":0}}
df=df.replace(convert)
df
#REPLACED THE FEMALE AS 1 AND MALE AS 0 BECAUSE IT WAS IN STRING
```

Out[5]:

	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 [6]:
```
convert={'region':{"southeast":1,"southwest":2,"northwest":3,"northeast":4}}
df=df.replace(convert)
df
#REPLACING THE STRING TO NUMERIC VALUES
```

Out[6]:

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

1338 rows × 7 columns

In [7]:
```
x=['sex','bmi','children','region','charges']
y=["yes","no"]
```

In [8]:
```
all_inputs=df[x]
all_classes=df['smoker']
x_train,x_test,y_train,y_test=train_test_split(all_inputs,all_classes,train_size
```

In [9]:
```
dc=DecisionTreeClassifier()
dc.fit(x_train,y_train)
#FITTING THE X AND TRAIN IN THE DECISION TREE CLASSIFIER
```

Out[9]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [10]:
```
dc.score(x_test,y_test)
#TO FIND THE ACCURACY SCORE
```

Out[10]:
```
0.9446935724962631
```

RANDOM FOREST

using insurance dataset

In [11]:
```
from sklearn.ensemble import RandomForestClassifier
#LIBRARY FOR RANDOMFOREST
```

In [12]:
```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
#TO FIT THE X AND Y TRAIN OF RANDOMFOREST
```

Out[12]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [13]:

```
rf=RandomForestClassifier()
params={'max_depth':[2,3,4,5,6],'min_samples_leaf':[5,10,15,20,50,100],'n_estima
#PARAMETERS ARE USED TO SPLIT NODES
```

In [14]:

```
from sklearn.model_selection import GridSearchCV
#GRIDSEARCH IS TO FIND THE PARAMETER VALUES FROM THE SET OF PARAMETERS THAT WERE
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring='accuracy')
grid_search.fit(x_train,y_train)
#TO FIT THE X AND Y TRAIN IN GRID SEARCH
```

Out[14]:

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

In [15]:

```
grid_search.best_score_
```

Out[15]: 0.9536464384663509

In [16]:

```
rf_best=grid_search.best_estimator_
print(rf_best)
#THE ESTIMATOR IS STORED IN RF_BEST
#BEST ESTIMATOR IS USED TO CALL THE PREDICT AND SCORE
```

```
RandomForestClassifier(max_depth=4, min_samples_leaf=5)
```

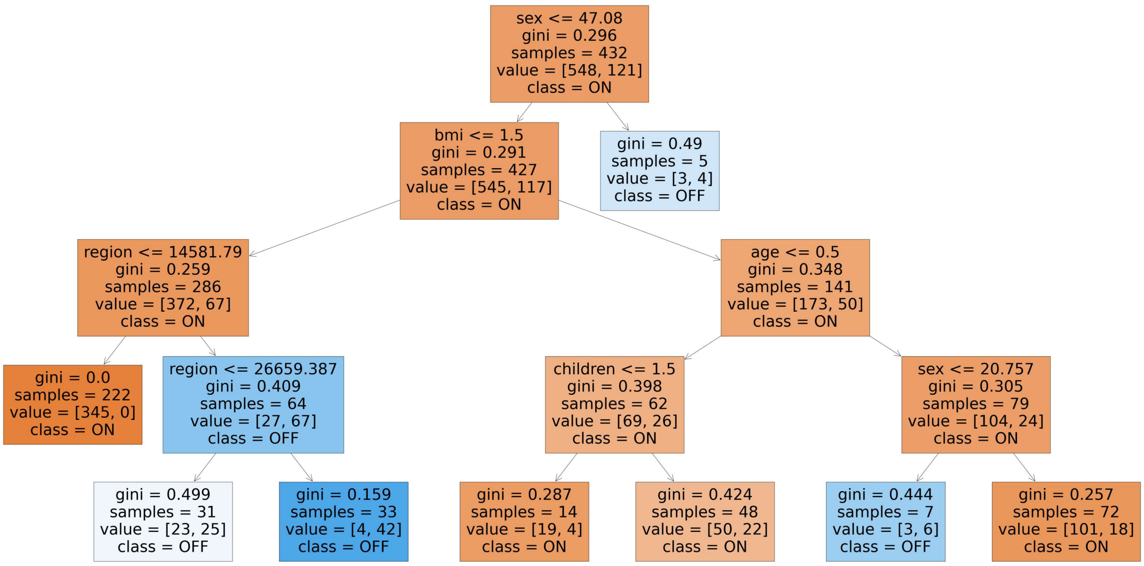
In [17]:

```
x=df.drop('smoker',axis=1)
y=df['smoker']
#HERE THE SMOKER COLUMN WAS DROP IN X AND STORED IN Y
```

In [20]:

```
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['ON','OFF'
#TO PLOT THE DECISIONTREE WE NEED TO IMPORT THE LIBRARY PLOT_TREE AND DECISIONTR
```

Out[20]:
```
[Text(0.5, 0.9, 'sex <= 47.08\ngini = 0.296\nsamples = 432\nvalue = [548, 121]
\nclass = ON'),
 Text(0.4230769230769231, 0.7, 'bmi <= 1.5\ngini = 0.291\nsamples = 427\nvalue
= [545, 117]\nclass = ON'),
 Text(0.15384615384615385, 0.5, 'region <= 14581.79\ngini = 0.259\nsamples = 28
6\nvalue = [372, 67]\nclass = ON'),
 Text(0.07692307692307693, 0.3, 'gini = 0.0\nsamples = 222\nvalue = [345, 0]\nc
lass = ON'),
 Text(0.23076923076923078, 0.3, 'region <= 26659.387\ngini = 0.409\nsamples = 6
4\nvalue = [27, 67]\nclass = OFF'),
 Text(0.15384615384615385, 0.1, 'gini = 0.499\nsamples = 31\nvalue = [23, 25]\n
class = OFF'),
 Text(0.3076923076923077, 0.1, 'gini = 0.159\nsamples = 33\nvalue = [4, 42]\ncl
ass = OFF'),
 Text(0.6923076923076923, 0.5, 'age <= 0.5\ngini = 0.348\nsamples = 141\nvalue
= [173, 50]\nclass = ON'),
 Text(0.5384615384615384, 0.3, 'children <= 1.5\ngini = 0.398\nsamples = 62\nva
lue = [69, 26]\nclass = ON'),
 Text(0.46153846153846156, 0.1, 'gini = 0.287\nsamples = 14\nvalue = [19, 4]\nc
lass = ON'),
 Text(0.6153846153846154, 0.1, 'gini = 0.424\nsamples = 48\nvalue = [50, 22]\nc
lass = ON'),
 Text(0.8461538461538461, 0.3, 'sex <= 20.757\ngini = 0.305\nsamples = 79\nvalu
e = [104, 24]\nclass = ON'),
 Text(0.7692307692307693, 0.1, 'gini = 0.444\nsamples = 7\nvalue = [3, 6]\nclas
s = OFF'),
 Text(0.9230769230769231, 0.1, 'gini = 0.257\nsamples = 72\nvalue = [101, 18]\n
class = ON'),
 Text(0.5769230769230769, 0.7, 'gini = 0.49\nsamples = 5\nvalue = [3, 4]\nclass
= OFF')]
```



In [21]:
```
rf_best.feature_importances_
```

Out[21]:
```
array([0.00566554, 0.06884245, 0.01722534, 0.01753561, 0.89073106])
```

In [22]:
```
df1=pd.DataFrame({'Varname':x_train.columns,'Imp':rf_best.feature_importances_})
```

In [23]:
```
df1.sort_values(by='Imp',ascending=False)
```

Out[23]:

	Varname	Imp
4	charges	0.890731
1	bmi	0.068842
3	region	0.017536
2	children	0.017225
0	sex	0.005666

CONCLUSION

In [ ]:
```
IN DECISION TREE THE SCORE OF X AND Y IS 94% AND IN THE RANDOM FOREST THE SCORE
RANDOM FOREST IS HIGHEST IN THE ACCURACY.
```