# INSURANCE DATASET

Logistic Regression

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

In [3]:

```
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 [4]:

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

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]:

```
0.9387144992526159
```

# 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]:

```
0.9447269639824828
```

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=6, min_samples_leaf=5)
```

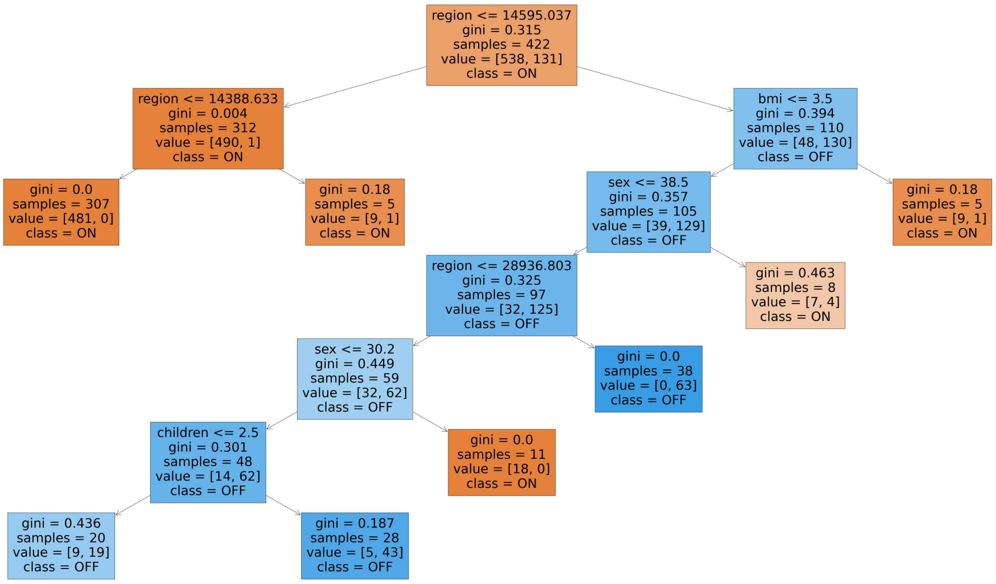
In [17]:

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

In [18]:

```
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[18]: [Text(0.5, 0.9285714285714286, 'region <= 14595.037\ngini = 0.315\nsamples = 42
         2\nvalue = [538, 131]\nclass = ON'),
          Text(0.25, 0.7857142857142857, 'region <= 14388.633\ngini = 0.004\nsamples = 3
         12\nvalue = [490, 1]\nclass = ON'),
          Text(0.125, 0.6428571428571429, 'gini = 0.0\nsamples = 307\nvalue = [481, 0]\n
         class = ON'),
          Text(0.375, 0.6428571428571429, 'gini = 0.18\nsamples = 5\nvalue = [9, 1]\ncla
         ss = ON'),
          Text(0.75, 0.7857142857142857, 'bmi <= 3.5\ngini = 0.394\nsamples = 110\nvalue
         = [48, 130]\nclass = OFF'),
          Text(0.625, 0.6428571428571429, 'sex <= 38.5\ngini = 0.357\nsamples = 105\nval
         ue = [39, 129]\nclass = OFF'),
          Text(0.5, 0.5, 'region <= 28936.803\ngini = 0.325\nsamples = 97\nvalue = [32,
         125]\nclass = OFF'),
          Text(0.375, 0.35714285714285715, 'sex <= 30.2\ngini = 0.449\nsamples = 59\nval
         ue = [32, 62]\nclass = OFF'),
          Text(0.25, 0.21428571428571427, 'children <= 2.5\ngini = 0.301\nsamples = 48\n
         value = [14, 62]\nclass = OFF'),
          Text(0.125, 0.07142857142857142, 'gini = 0.436\nsamples = 20\nvalue = [9, 19]
         \nclass = OFF'),
          Text(0.375, 0.07142857142857142, 'gini = 0.187\nsamples = 28\nvalue = [5, 43]
         \nclass = OFF'),
          Text(0.5, 0.21428571428571427, 'gini = 0.0\nsamples = 11\nvalue = [18, 0]\ncla
         ss = ON'),
          Text(0.625, 0.35714285714285715, 'gini = 0.0\nsamples = 38\nvalue = [0, 63]\nc
         lass = OFF'),
          Text(0.75, 0.5, 'gini = 0.463\nsamples = 8\nvalue = [7, 4]\nclass = ON'),
          Text(0.875, 0.6428571428571429, 'gini = 0.18\nsamples = 5\nvalue = [9, 1]\ncla
         ss = ON')]
```


region <= 14595.037 gini = 0.315 samples = 422 value = [538, 131] class = ON
region <= 14388.633 gini = 0.004 samples = 312 value = [490, 1] class = ON
bmi <= 3.5 gini = 0.394 samples = 110 value = [48, 130] class = OFF
gini = 0.0 samples = 307 value = [481, 0] class = ON
gini = 0.18 samples = 5 value = [9, 1] class = ON
sex <= 38.5 gini = 0.357 samples = 105 value = [39, 129] class = OFF
gini = 0.18 samples = 5 value = [9, 1] class = ON
region <= 28936.803 gini = 0.325 samples = 97 value = [32, 125] class = OFF
gini = 0.463 samples = 8 value = [7, 4] class = ON
sex <= 30.2 gini = 0.449 samples = 59 value = [32, 62] class = OFF
gini = 0.0 samples = 38 value = [0, 63] class = OFF
children <= 2.5 gini = 0.301 samples = 48 value = [14, 62] class = OFF
gini = 0.0 samples = 11 value = [18, 0] class = ON
gini = 0.436 samples = 20 value = [9, 19] class = OFF
gini = 0.187 samples = 28 value = [5, 43] class = OFF

```
In [19]: rf_best.feature_importances_
```

```
Out[19]: array([0.00629788, 0.08285052, 0.01589012, 0.0140862 , 0.88087528])
```

```
In [20]: df1=pd.DataFrame({'Varname':x_train.columns,'Imp':rf_best.feature_importances_})
```

```
In [21]: df1.sort_values(by='Imp',ascending=False)
```

Out[21]:

|   | Varname | Imp |
|---|---|---|
| **4** | charges | 0.880875 |
| **1** | bmi | 0.082851 |
| **2** | children | 0.015890 |
| **3** | region | 0.014086 |
| **0** | sex | 0.006298 |

In [10]:

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

Out[10]:

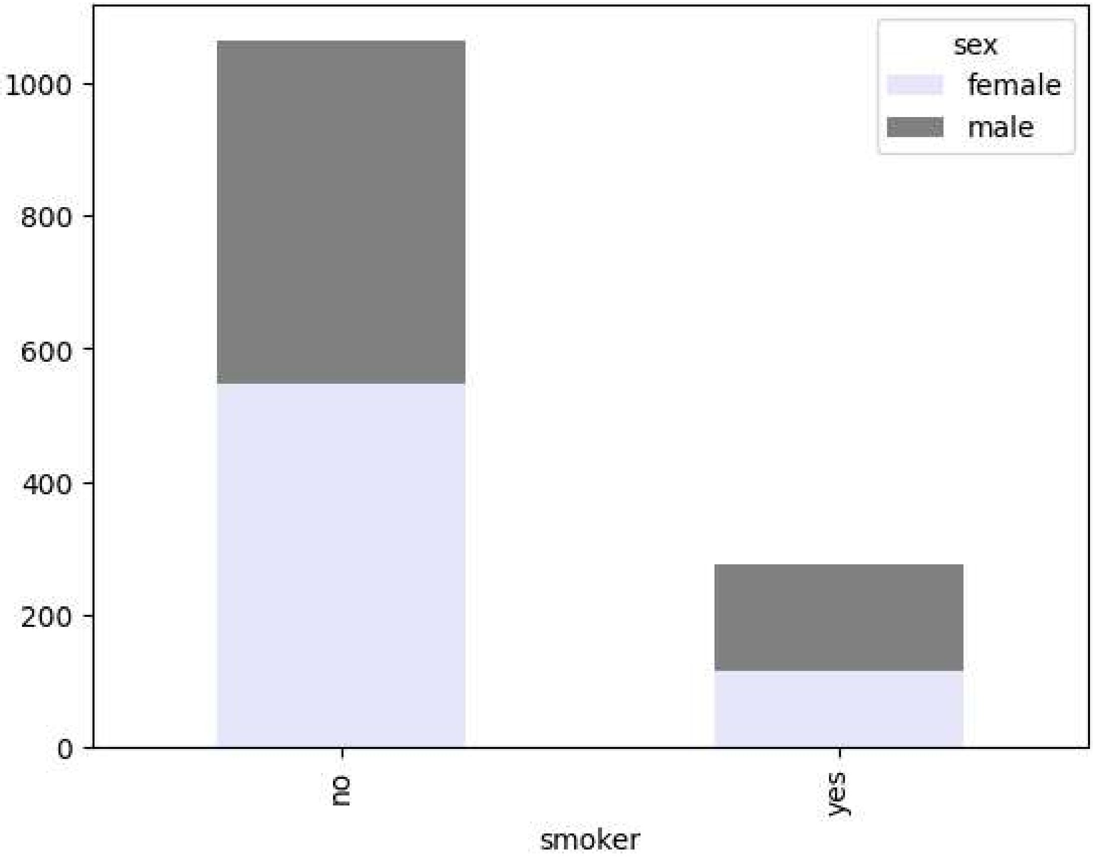
| **sex** | **female** | **male** |
|---|---|---|
| **smoker** | | |
| **no** | 547 | 517 |
| **yes** | 115 | 159 |

In [16]:

```
v.plot(kind='bar',stacked=True,color=["lavender","grey"],grid=False)
```

Out[16]:

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



# CONCLUSION

IN DECISION TREE THE SCORE OF X AND Y IS 94% AND IN THE RANDOM FOREST THE SCORE IS 95% COMPARING THE BOTH RANDOM FOREST IS HIGHEST IN THE ACCURACY AND AS PER THE PROBLEM STATEMENT MALE SMOKERS ARE HIGHER THEN FEMALE SMOKER.