MINI PROJECT

PROBLEM STATEMENT : Which model is suitable for Insurance Dataset

Importing packages

In [86]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Reading the data

In [88]:
```
df=pd.read_csv(r"C:\Users\Lenovo\OneDrive\Desktop\Data Sets\insurance.csv")
df
```

Out[88]:

	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 [89]:
```
df.shape
```

Out[89]:
```
(1338, 7)
```

In [90]:
```
df.head()
```

Out[90]:

	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 [91]:
```
df.tail()
```

Out[91]:

	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 [92]:
```
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 [93]:
```
df.describe()
```

Out[93]:

	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 [94]:
```
df.isnull().sum()
```

Out[94]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [95]:
```
smoker={"smoker":{"yes":1,"no":0}}
df=df.replace(smoker)
df
```

Out[95]:

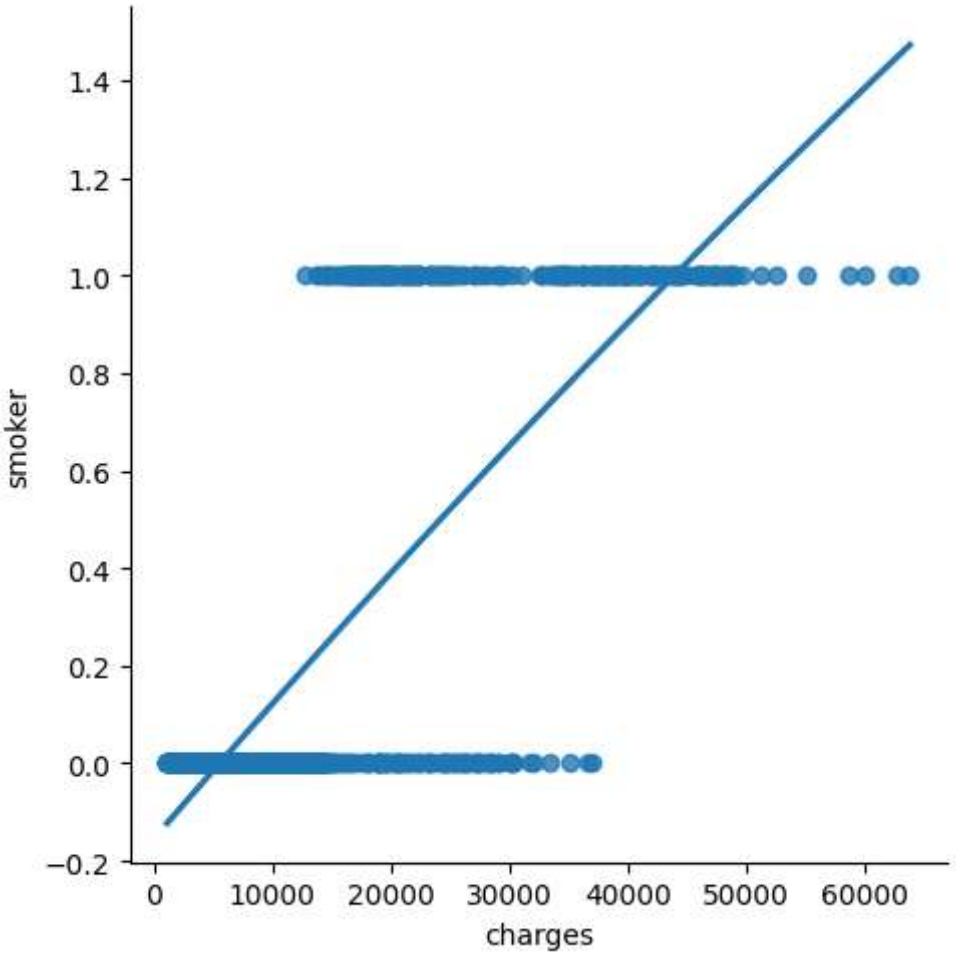
	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	1	southwest	16884.92400
1	18	male	33.770	1	0	southeast	1725.55230
2	28	male	33.000	3	0	southeast	4449.46200
3	33	male	22.705	0	0	northwest	21984.47061
4	32	male	28.880	0	0	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	0	northwest	10600.54830
1334	18	female	31.920	0	0	northeast	2205.98080
1335	18	female	36.850	0	0	southeast	1629.83350
1336	21	female	25.800	0	0	southwest	2007.94500
1337	61	female	29.070	0	1	northwest	29141.36030

1338 rows × 7 columns

Data Visualization

In [96]:
```
sns.lmplot(x="charges",y="smoker",data=df,order=2,ci=None)
```

Out[96]:
```
<seaborn.axisgrid.FacetGrid at 0x19102284890>
```



Applying Linear Regression

In [97]:
```
#Linear regression
from sklearn.model_selection import train_test_split
from sklearn import preprocessing,svm
from sklearn.linear_model import LinearRegression
```

In [98]:
```
regr=LinearRegression()
regr.fit(X_train,Y_train)
```

Out[98]:
```
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.

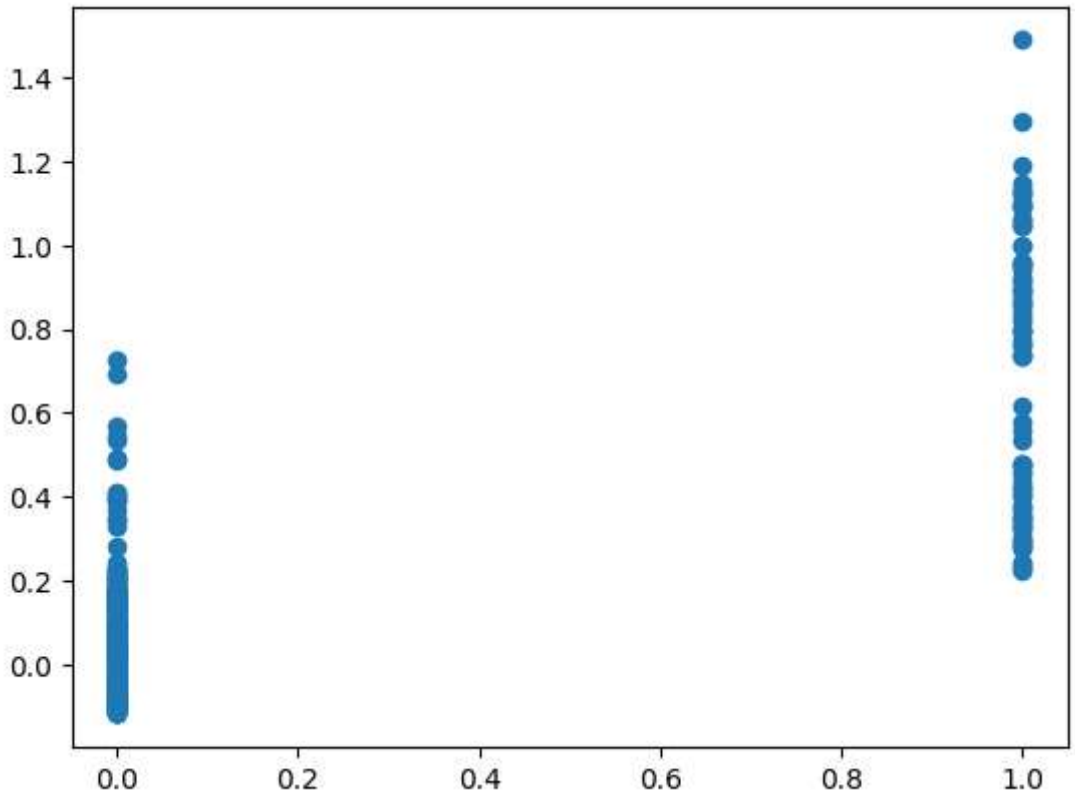
In [99]:
```
score=regr.score(X_test,Y_test)
print(score)
```

```
0.6474860981235602
```

In [100]:
```
predictions=regr.predict(X_test)
```

In [101]:
```
plt.scatter(Y_test,predictions)
```

Out[101]:
```
<matplotlib.collections.PathCollection at 0x191026af090>
```



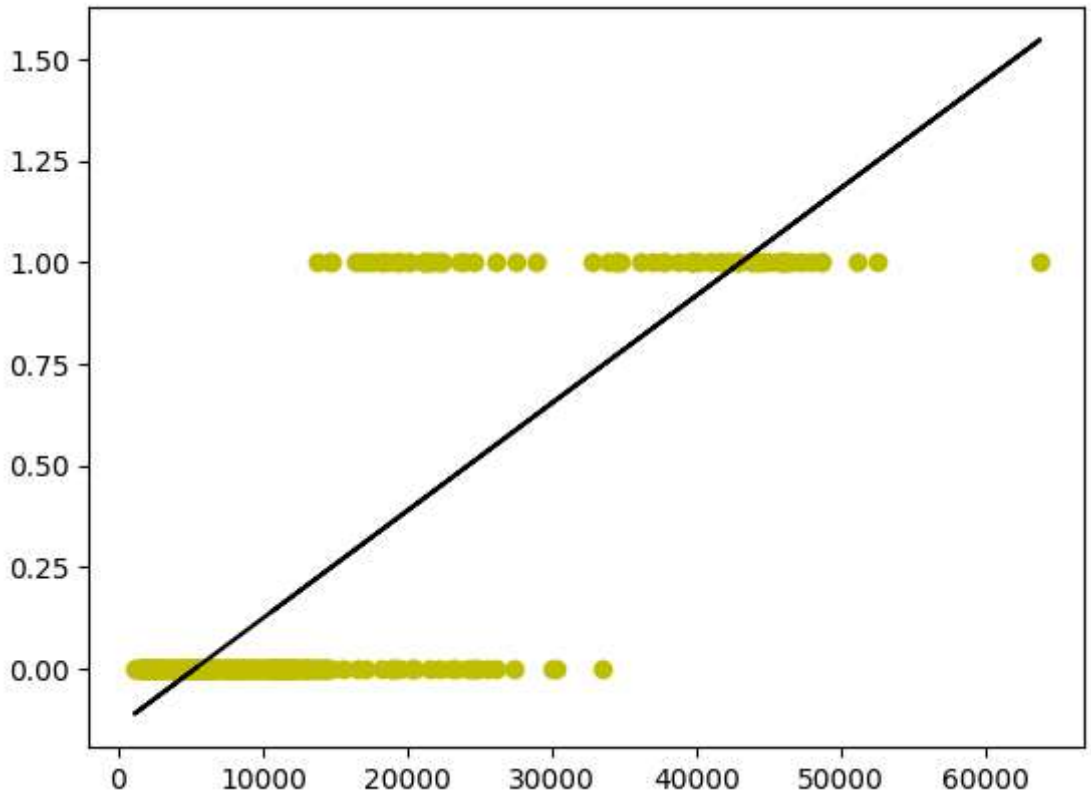
In [102]:
```
x=np.array(df['charges']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
df.dropna(inplace=True)
```

In [103]:
```
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.25)
regr.fit(X_train,Y_train)
regr.fit(X_train,Y_train)
```

Out[103]:
```
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 [104]:
```
y_pred=regr.predict(X_test)
plt.scatter(X_test,Y_test,color='y')
plt.plot(X_test,y_pred,color='k')
plt.show()
```



Here we did'nt got the accuracy for Linear Regression so we are going to implement Logistic Regression

In [105]:
```
#Logistic regression
x=np.array(df['charges']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
df.dropna(inplace=True)
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.25)
```

In [106]:
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(max_iter=1000)
```

In [107]:
```
lr.fit(X_train,Y_train)
```

```
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\utils\validation.py:1143: DataConversionWarning: A column-vector y was p
assed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[107]:
```
LogisticRegression(max_iter=1000)
```

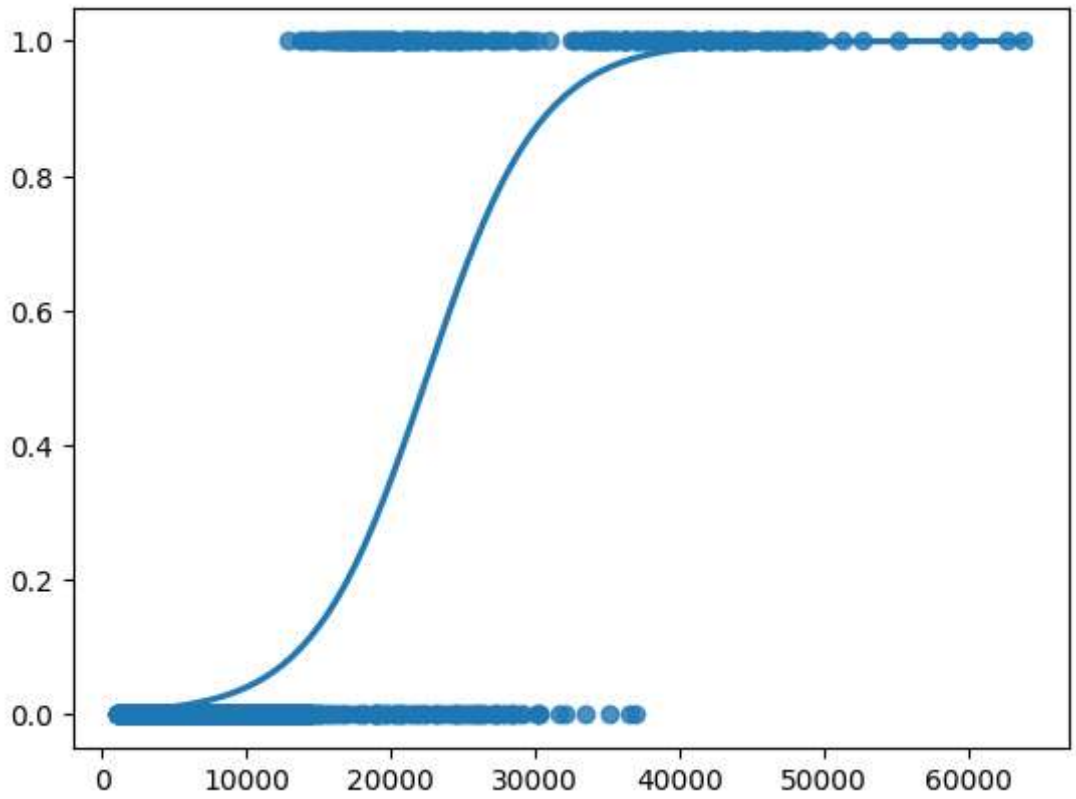
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [108]:
```
score=lr.score(X_test,Y_test)
print(score)
```

```
0.9253731343283582
```

In [109]:
```
sns.regplot(x=x,y=y,data=df,logistic=True,ci=None)
```

Out[109]:
```
<Axes: >
```



Here we got the best fit curve for Logistic Regression.Now we are going to check that if we get better accuracy by implementing Decision Tree and Random Forest

Decision Tree

In [110]:
```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(random_state=0)
clf.fit(X_train,Y_train)
```

Out[110]:
```
DecisionTreeClassifier(random_state=0)
```
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 [111]:
```
score=clf.score(X_test,Y_test)
print(score)
```

```
0.9014925373134328
```

Random Forest

In [112]:
```
#RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(X_train,Y_train)
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_5492\3367015011.py:4: DataConver
sionWarning: A column-vector y was passed when a 1d array was expected. Pleas
e change the shape of y to (n_samples,), for example using ravel().
  rfc.fit(X_train,Y_train)
```

Out[112]:
```
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 [113]:
```
params = {'max_depth': [2,3,5,10,20],
'min_samples_leaf': [5,10,20,50,100,200],
'n_estimators': [10,25,30,50,100,200]}
```

In [114]:
```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=rfc,param_grid=params,cv = 2, scoring='acc
```

In [115]:
```
grid_search.fit(X_train,Y_train)
```

```
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\s
klearn\model_selection\_validation.py:686: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape o
f y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\s
klearn\model_selection\_validation.py:686: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape o
f y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\s
klearn\model_selection\_validation.py:686: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape o
f y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\s
klearn\model_selection\_validation.py:686: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape o
f y to (n_samples,), for example using ravel().
```

In [119]:
```
grid_search.best_score_
```

Out[119]:
```
0.9362112428529394
```

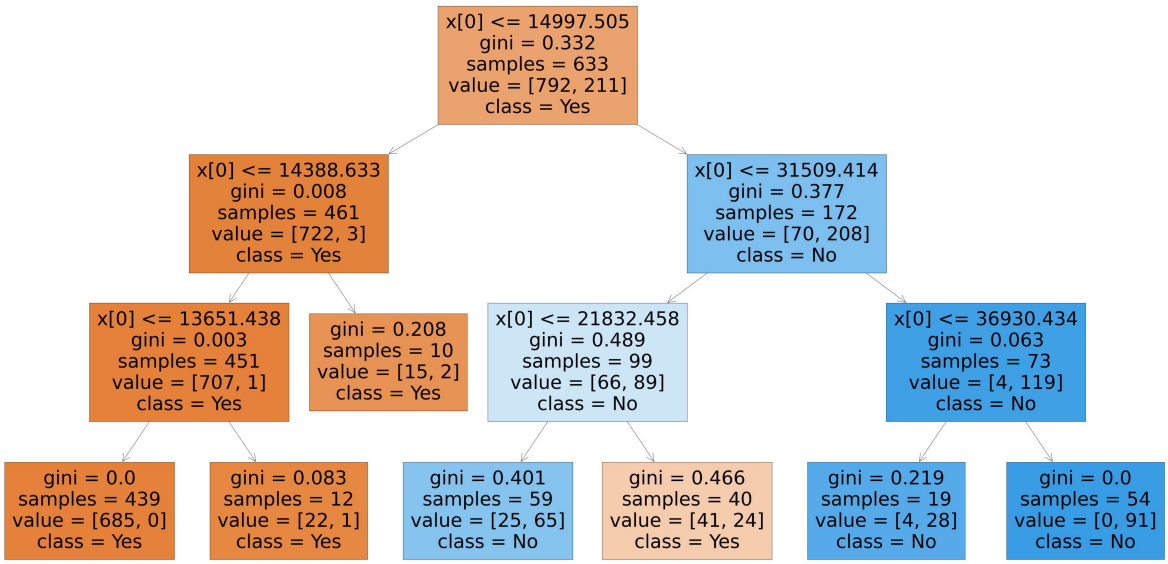
In [120]:
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[120]:
```
RandomForestClassifier(max_depth=3, min_samples_leaf=10, n_estimators=10)
```

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 [121]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],class_names=['Yes',"No"],filled=True)
```

Out[121]:
```
[Text(0.4583333333333333, 0.875, 'x[0] <= 14997.505\ngini = 0.332\nsamples =
633\nvalue = [792, 211]\nclass = Yes'),
 Text(0.25, 0.625, 'x[0] <= 14388.633\ngini = 0.008\nsamples = 461\nvalue =
[722, 3]\nclass = Yes'),
 Text(0.16666666666666666, 0.375, 'x[0] <= 13651.438\ngini = 0.003\nsamples =
451\nvalue = [707, 1]\nclass = Yes'),
 Text(0.08333333333333333, 0.125, 'gini = 0.0\nsamples = 439\nvalue = [685,
0]\nclass = Yes'),
 Text(0.25, 0.125, 'gini = 0.083\nsamples = 12\nvalue = [22, 1]\nclass = Ye
s'),
 Text(0.3333333333333333, 0.375, 'gini = 0.208\nsamples = 10\nvalue = [15, 2]
\nclass = Yes'),
 Text(0.6666666666666666, 0.625, 'x[0] <= 31509.414\ngini = 0.377\nsamples =
172\nvalue = [70, 208]\nclass = No'),
 Text(0.5, 0.375, 'x[0] <= 21832.458\ngini = 0.489\nsamples = 99\nvalue = [6
6, 89]\nclass = No'),
 Text(0.4166666666666667, 0.125, 'gini = 0.401\nsamples = 59\nvalue = [25, 6
5]\nclass = No'),
 Text(0.5833333333333334, 0.125, 'gini = 0.466\nsamples = 40\nvalue = [41, 2
4]\nclass = Yes'),
 Text(0.8333333333333334, 0.375, 'x[0] <= 36930.434\ngini = 0.063\nsamples =
73\nvalue = [4, 119]\nclass = No'),
 Text(0.75, 0.125, 'gini = 0.219\nsamples = 19\nvalue = [4, 28]\nclass = N
o'),
 Text(0.9166666666666666, 0.125, 'gini = 0.0\nsamples = 54\nvalue = [0, 91]\n
class = No')]
```



In [122]:
```
score=rfc.score(X_test,Y_test)
print(score)
```

```
0.9014925373134328
```

Conclusion : Based on accuracy scores of the implemented models we can conclude that "Logistic Regression" is the best model for the given dataset