PROBLEM STATEMENT:WHICH MODEL IS BETTER FOR INSURANCE DATA SET

IMPORTING NECESSARY LIBRARY FILES

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt,seaborn as sns
```

READING THE DATASET

In [2]:

```
df=pd.read_csv(r"C:\Users\mural\Downloads\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 AND PREPROCESSING

In [3]:

```
df.head()
```

Out[3]:

	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 [4]:

```
df.tail()
```

Out[4]:

	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 [5]:

```
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 [6]:

```
df.describe()
```

Out[6]:

	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 [7]:

```
# CHECKING FOR NULL VALUES
df.isnull().sum()
```

Out[7]:

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

In [8]:

```
#Checking for duplicates
df.duplicated().sum()
```

Out[8]:

```
1
```

In [9]:

```
#Droping the duplicates
df=df.drop_duplicates()
df
```

Out[9]:

	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

1337 rows × 7 columns

In [10]:

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

In [11]:

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

Out[11]:

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

1337 rows × 7 columns

FEATURE SCALING:TO SPLIT THE DATA INTO TRAIN AND TEST DATA

In [12]:

```
x=df[['sex','age','bmi','children','smoker']]
y=df['charges']
```

In [13]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100)
```

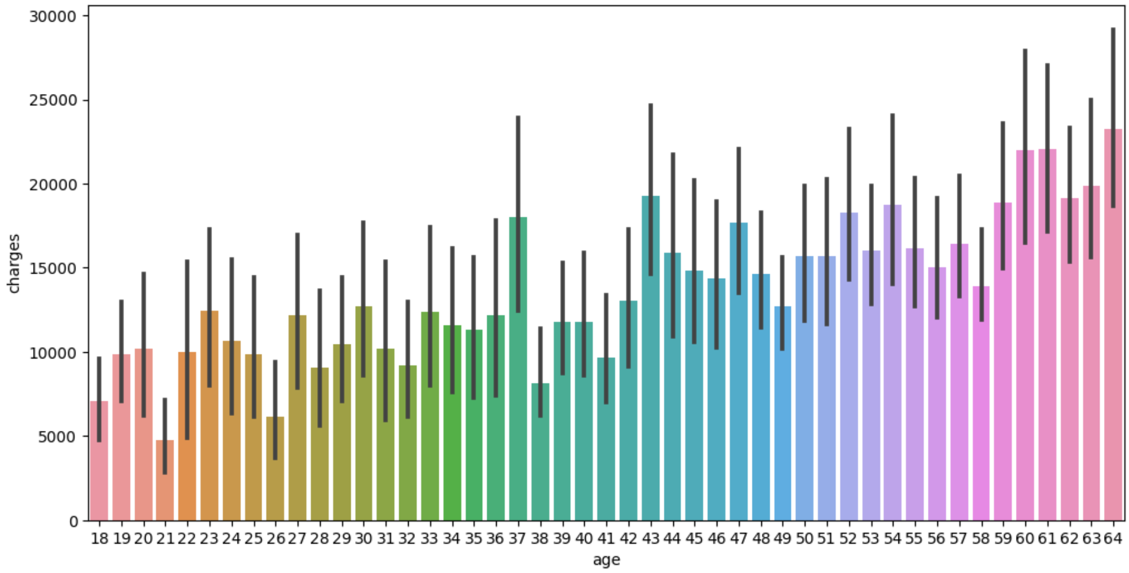
DATA VISUALIZATION

In [14]:

```
#Bar plots are a type of data visualization used to represent data in the form of rectan
plt.figure(figsize=(12,6))
sns.barplot(x='age',y='charges',data=df)
```

Out[14]:

```
<Axes: xlabel='age', ylabel='charges'>
```

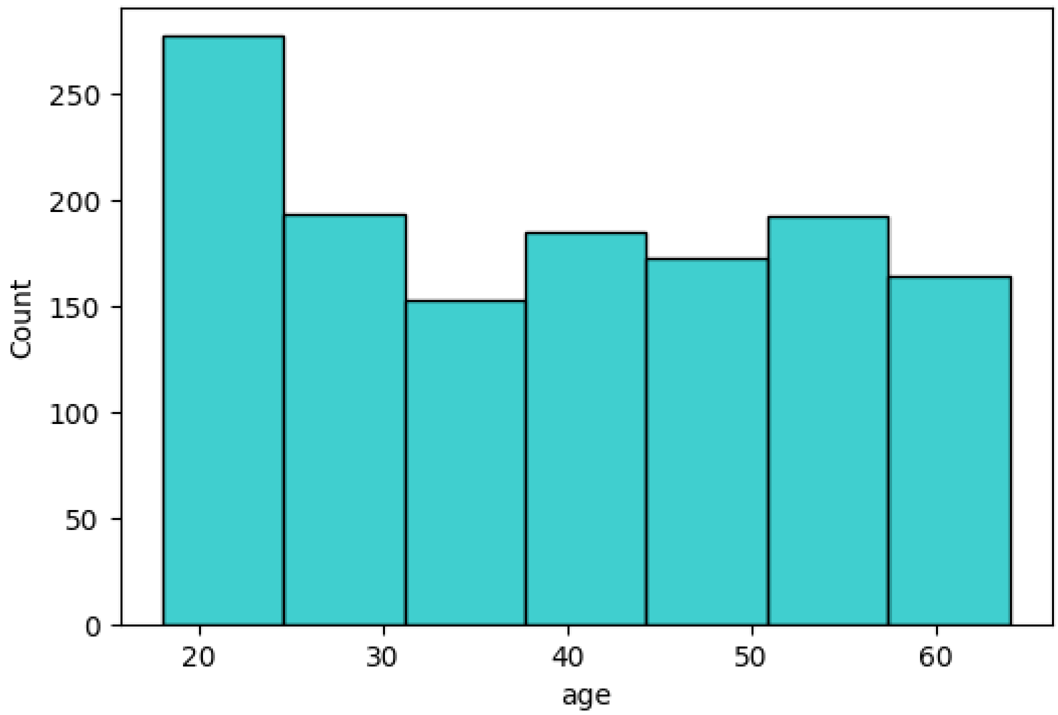


In [15]:

```
plt.figure(figsize=(6,4))
sns.histplot(df['age'], bins=7, color='c')
```

Out[15]:

```
<Axes: xlabel='age', ylabel='Count'>
```

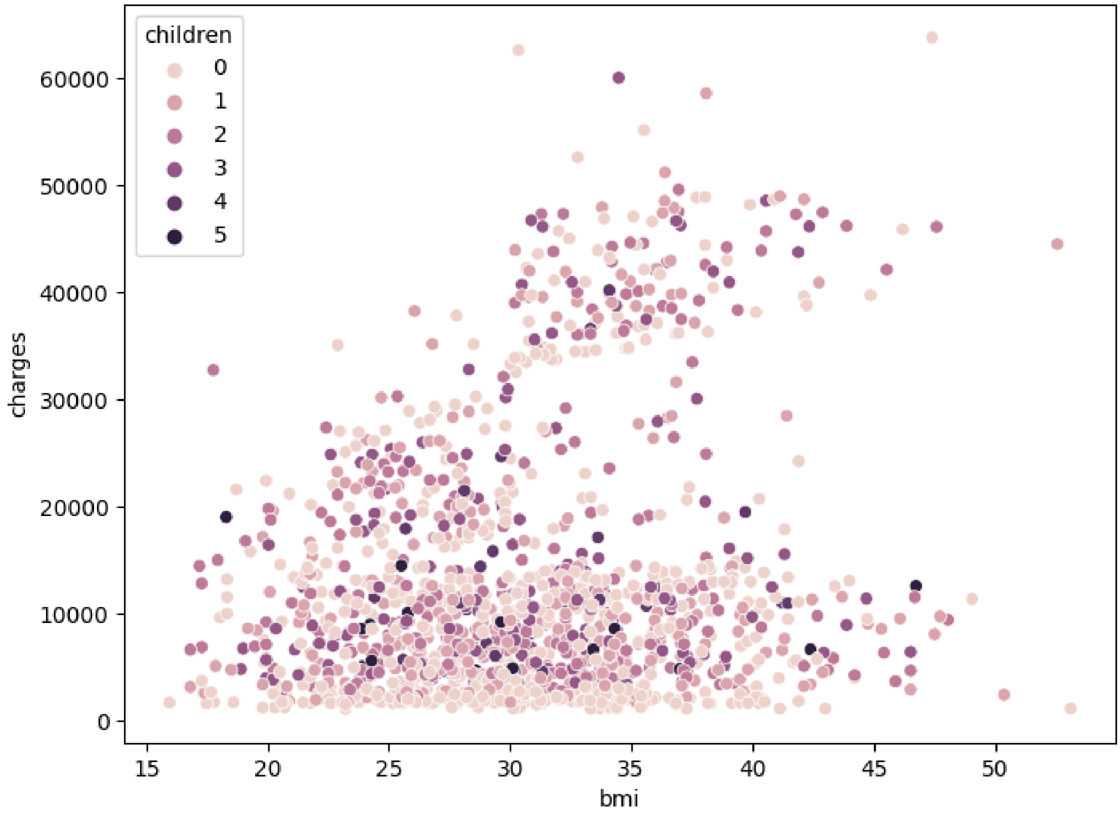


In [16]:

```
plt.figure(figsize=(8,6))
sns.scatterplot(x=df.bmi,y=df.charges,hue=df.children)
```

Out[16]:

```
<Axes: xlabel='bmi', ylabel='charges'>
```

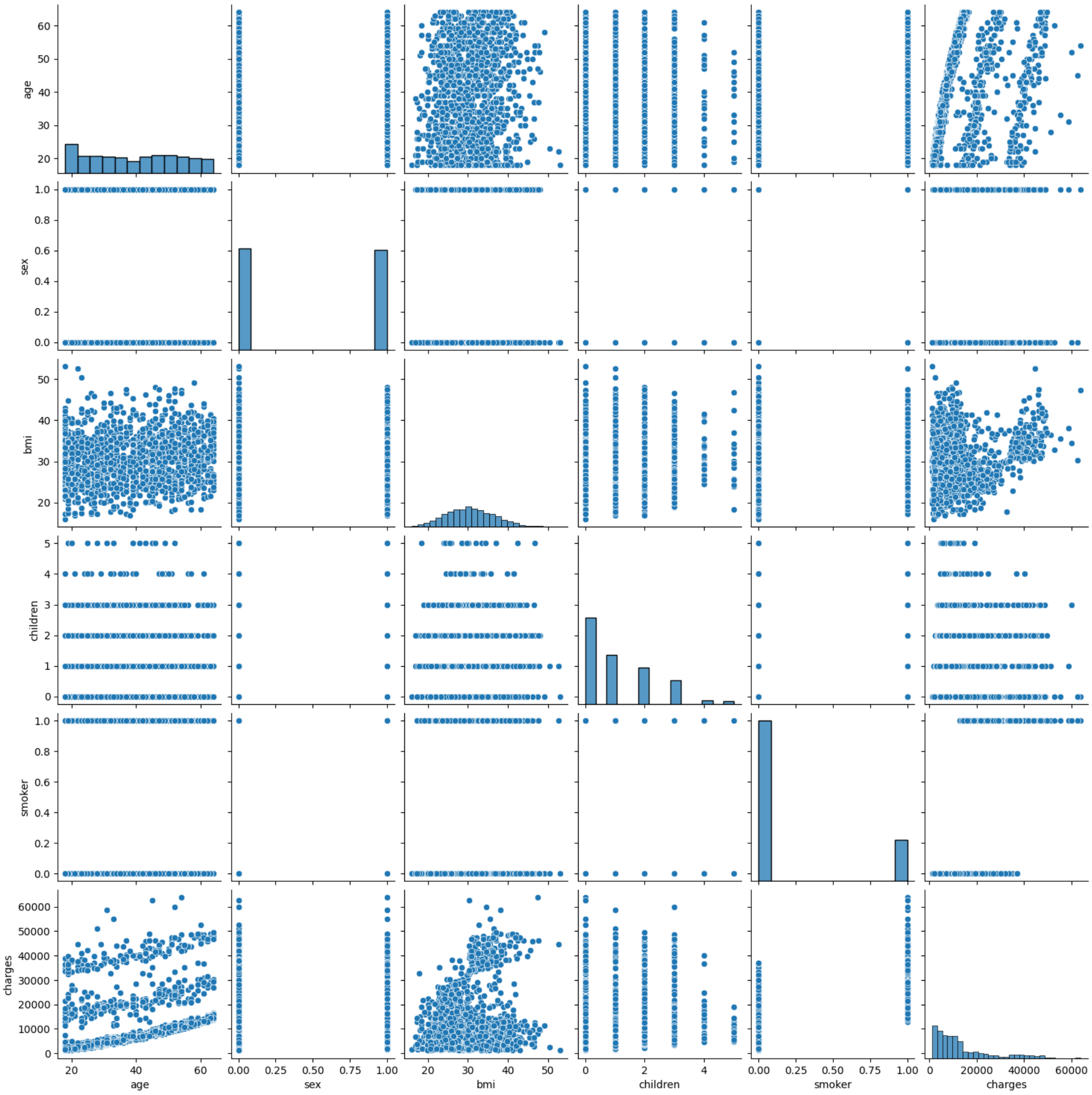


In [17]:

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

Out[17]:

```
<seaborn.axisgrid.PairGrid at 0x1b80313b390>
```



LINEAR REGRESSION

In [18]:

```
from sklearn.linear_model import LinearRegression
ln=LinearRegression()
#training the algorithm
ln.fit(x_train,y_train)
```

Out[18]:

```
▾ LinearRegression
LinearRegression()
```

In [19]:

```
#prediction
y_pred=ln.predict(x_test)
print(y_pred)
```

```
[ 4707.74208499  5270.00144692  8420.37778701  2117.85423276
 24629.55217829 37462.67995089  6736.16903059 11888.12283755
 30009.9300288  15930.67835221 15521.88288054 11804.87021395
 11025.25303489  4098.23414127  9822.10972576 32502.25870281
  8084.25364438 13806.19337299  7271.02092582 18327.45709951
 14316.41888638 11317.78561384 16933.07506832 31914.9410167
 13847.66083987 32663.47526599  6337.11980733 40074.8809138
 32391.60862503 11326.6356485  19032.50721655  2423.17394598
 39660.21308053 13774.69426205  4250.34393253 12025.32139773
 11241.35617671  8785.93248832  3886.78280671 39262.46049103
 10386.58259565 12535.99712811  5543.04962195  8788.36035042
   788.75157947  5635.27023401  5802.10544485  1763.42138759
  2127.96745926  1621.03939977 17381.46570332 17302.4038081
  8428.466086   11373.6225086   7751.03595426 11261.02673164
  3448.98181357 31410.08734219  8631.96023679 12739.84509555
  3816.05525616 27972.66813065  6017.19011374 12386.75808722
 14679.58742965 10136.9710914   1157.51457789  8739.20760429
 11824.26835562 37693.38263588 10276.93288536  4429.38728972
 11124.62705144 28460.86853747  9558.1502172    717.98257923
```

In [20]:

```
from sklearn.metrics import r2_score
score=r2_score(y_test,y_pred)
```

In [21]:

```
score
```

Out[21]:

```
0.7578201409315171
```

In [64]:

```
# We have  a R^2 score of 0.75 which tells that our model is accurate
```

LOGISTIC REGRESSION

In [34]:

```
#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=0.3,random_state=1)
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(max_iter=10000)
```

In [35]:

```
lr.fit(x_train,y_train)
```

```
C:\Users\mural\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\utils\validation.py:1143: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y t
o (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[35]:

```
▾        LogisticRegression
LogisticRegression(max_iter=10000)
```

In [36]:

```
score=lr.score(x_test,y_test)
print(score)
```

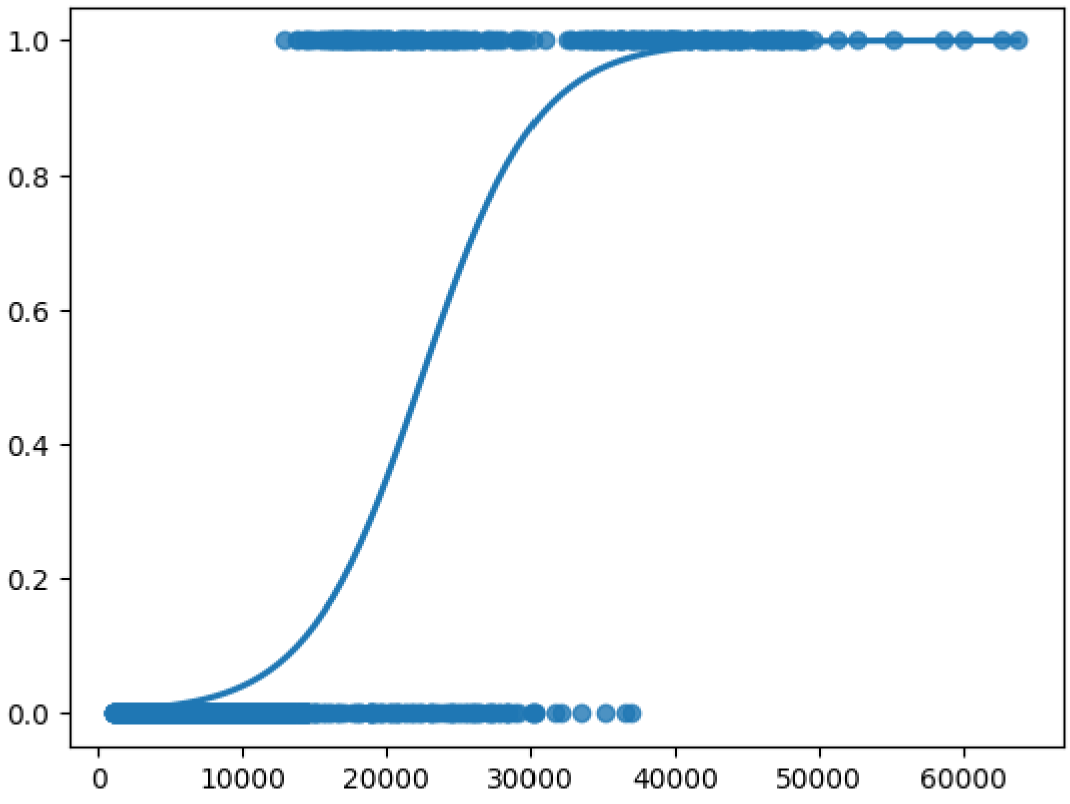
```
0.9253731343283582
```

In [37]:

```
sns.regplot(x=x,y=y,data=df,logistic=True,ci=None)
```

Out[37]:

```
<Axes: >
```



In [66]:

```
# We got the best score for Logistic Regression
```

In [67]:

```
#Now we are going to check that if we may get better accuracy by implementing Decision T
```

Decision Tree

In [38]:

```
#Decision tree
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(random_state=0)
clf.fit(x_train,y_train)
```

Out[38]:

```
▾        DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [39]:

```
score=clf.score(x_test,y_test)
print(score)
```

```
0.900497512437811
```

Random Forest

In [55]:

```
#Random forest classifier
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
C:\Users\mural\AppData\Local\Temp\ipykernel_8652\2638823938.py:4: DataCon
versionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples,), for example using ravel
().
  rfc.fit(x_train,y_train)
```

Out[55]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [56]:

```
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 [57]:

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

In [58]:

```
grid_search.fit(x_train,y_train)
```

```
ge the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mural\AppData\Local\Programs\Python\Python311\Lib\site-packag
es\sklearn\model_selection\_validation.py:686: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mural\AppData\Local\Programs\Python\Python311\Lib\site-packag
es\sklearn\model_selection\_validation.py:686: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mural\AppData\Local\Programs\Python\Python311\Lib\site-packag
es\sklearn\model_selection\_validation.py:686: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mural\AppData\Local\Programs\Python\Python311\Lib\site-packag
es\sklearn\model_selection\_validation.py:686: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please chan
```

In [59]:

```
grid_search.best_score_
```

Out[59]:

```
0.9219216127674372
```

In [60]:

```
rf_best=grid_search.best_estimator_
rf_best
```

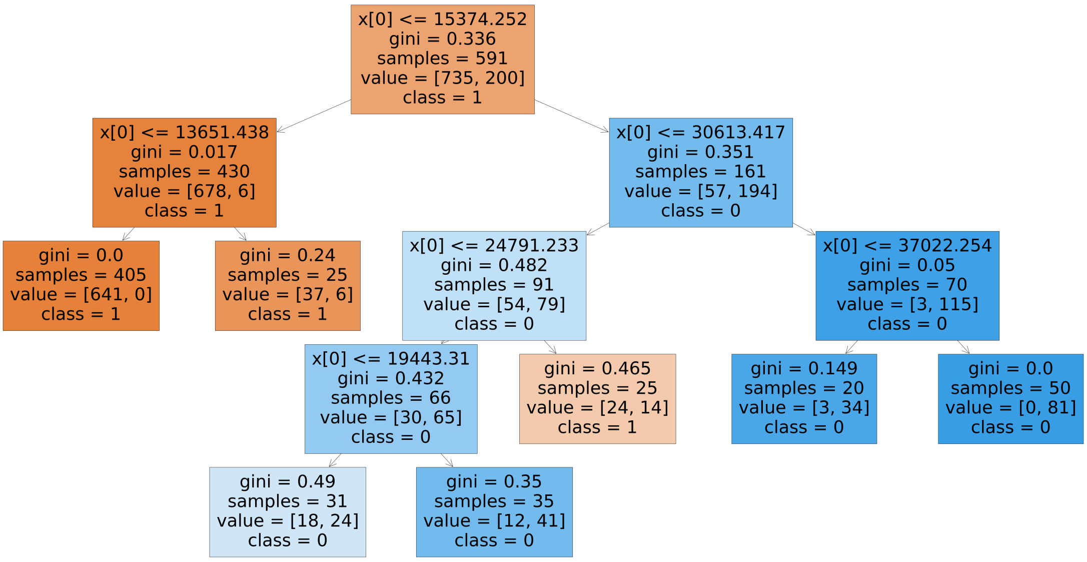
Out[60]:

```
▾                       RandomForestClassifier

RandomForestClassifier(max_depth=10, min_samples_leaf=20, n_estimators=1
0)
```

In [61]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['1','0'],filled=True);
```



In [63]:

```
score=rfc.score(x_test,y_test)
score
```

Out[63]:

```
0.900497512437811
```

CONCLUSION : Based on accuracy scores of all models that were implemented we can conclude that"Logistic Regression" is the best model for the given data set

In [ ]: