Problem Statement:-

Which model is suitable for Insurance Dataset

Import all the required packages

In [1]:

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

Step-1:- Data Collection

In [2]:

```
df=pd.read_csv(r"C:\Users\thara\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

Step-2:- Data Preprocessing

In [3]:

```
# The shape of a DataFrame is a tuple of array dimensions that tells the number of rows and columns of a given DataFrame
df.shape
```

Out[3]:

```
(1338, 7)
```

In [4]:

```
# The head() returns the first n rows for the object based on position.

df.head()
```

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

In [5]:

```
# The tail function in Python displays the last five rows of the dataframe by default.

df.tail()
```

Out[5]:

	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 [6]:

```
# To check the null values.

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

Out[6]:

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

There Are No Null Values In The Above DataFrame

In [7]:

```
# Check the data types for each column.

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]:

```
# Explore numerical columns stats.
# The describe function only displays the numerical columns present in the Dataset.

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]:

```
# Explore our age variable.

df.age.describe()
```

Out[9]:

```
count    1338.000000
mean       39.207025
std        14.049960
min        18.000000
25%        27.000000
50%        39.000000
75%        51.000000
max        64.000000
Name: age, dtype: float64
```

In [10]:

```
df.duplicated().sum()
```

Out[10]:

```
1
```

In [11]:

```
df=df.drop_duplicates()
df
```

Out[11]:

	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 [12]:

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

```
      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

[1337 rows x 7 columns]
```

In [13]:

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

Out[13]:

	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:

Feature Scaling is done to normalize the features in the dataset into a finite range.

In [14]:

```
#Training the model

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

In [15]:

```
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)
```

Step-3:- Data Visualization

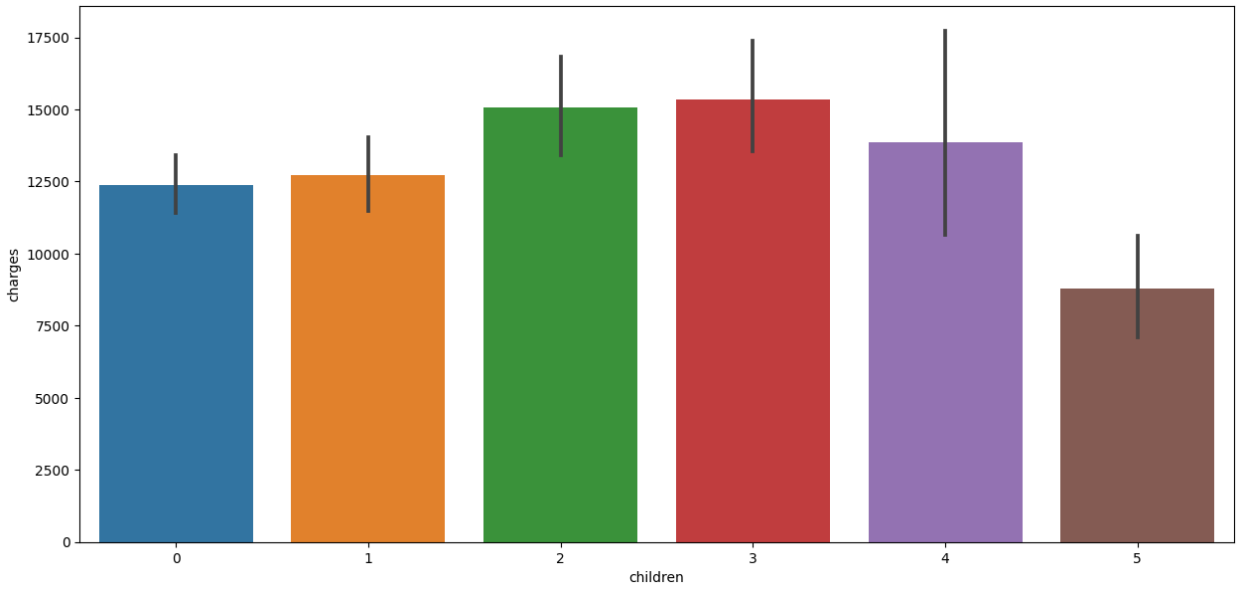
In [16]:

```
# Explore expenses relationships.
# A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights

plt.figure(figsize=(15,7))
sns.barplot(x=df.children, y=df.charges)
```

Out[16]:

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



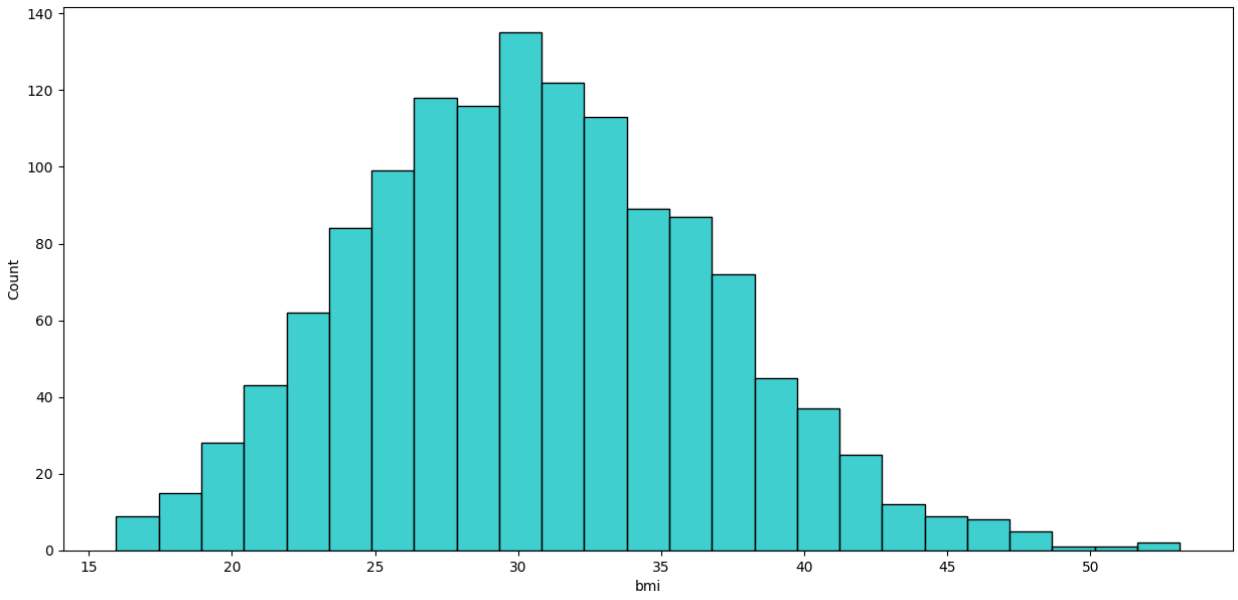
In [17]:

```
# Visualize our BMI column.
# A histogram is a graph showing frequency distributions.
# It is a graph showing the number of observations within each given interval.

plt.figure(figsize=(15,7))
sns.histplot(data=df, x='bmi',color='c')
```

Out[17]:

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



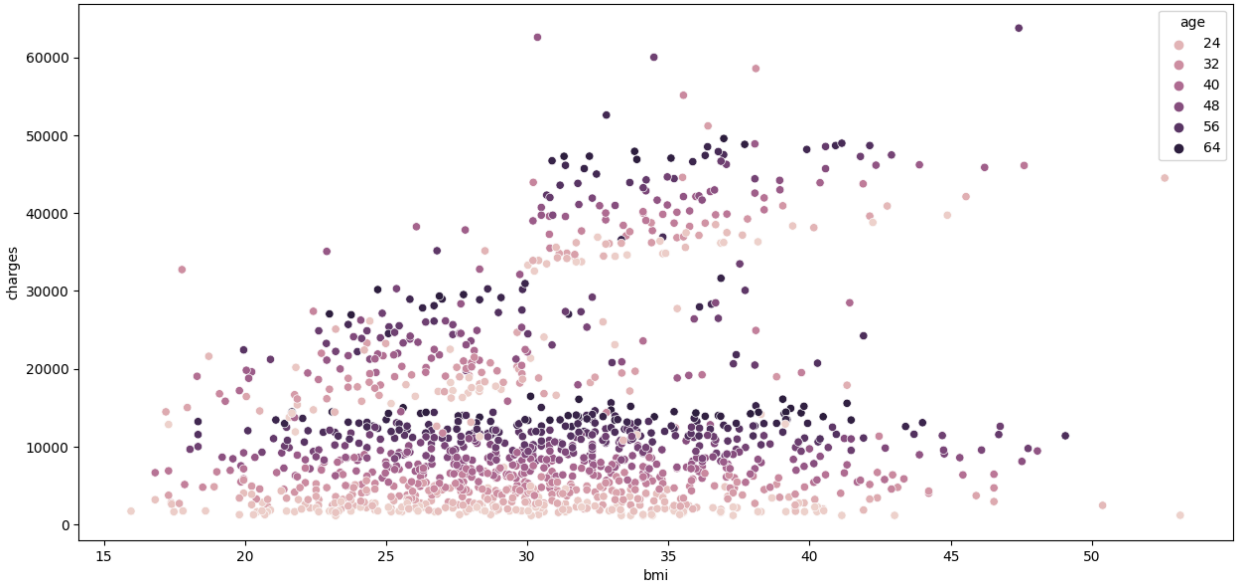
In [18]:

```
# With Pyplot, you can use the scatter() function to draw a scatter plot.

# The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values
plt.figure(figsize=(15,7))
sns.scatterplot(x=df.bmi, y=df.charges,hue=df.age)
```

Out[18]:

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



In [19]:

```
df.charges.corr(df.bmi)
```

Out[19]:

```
0.19840083122624932
```

In [20]:

```
df.charges.corr(df.children)
```

Out[20]:

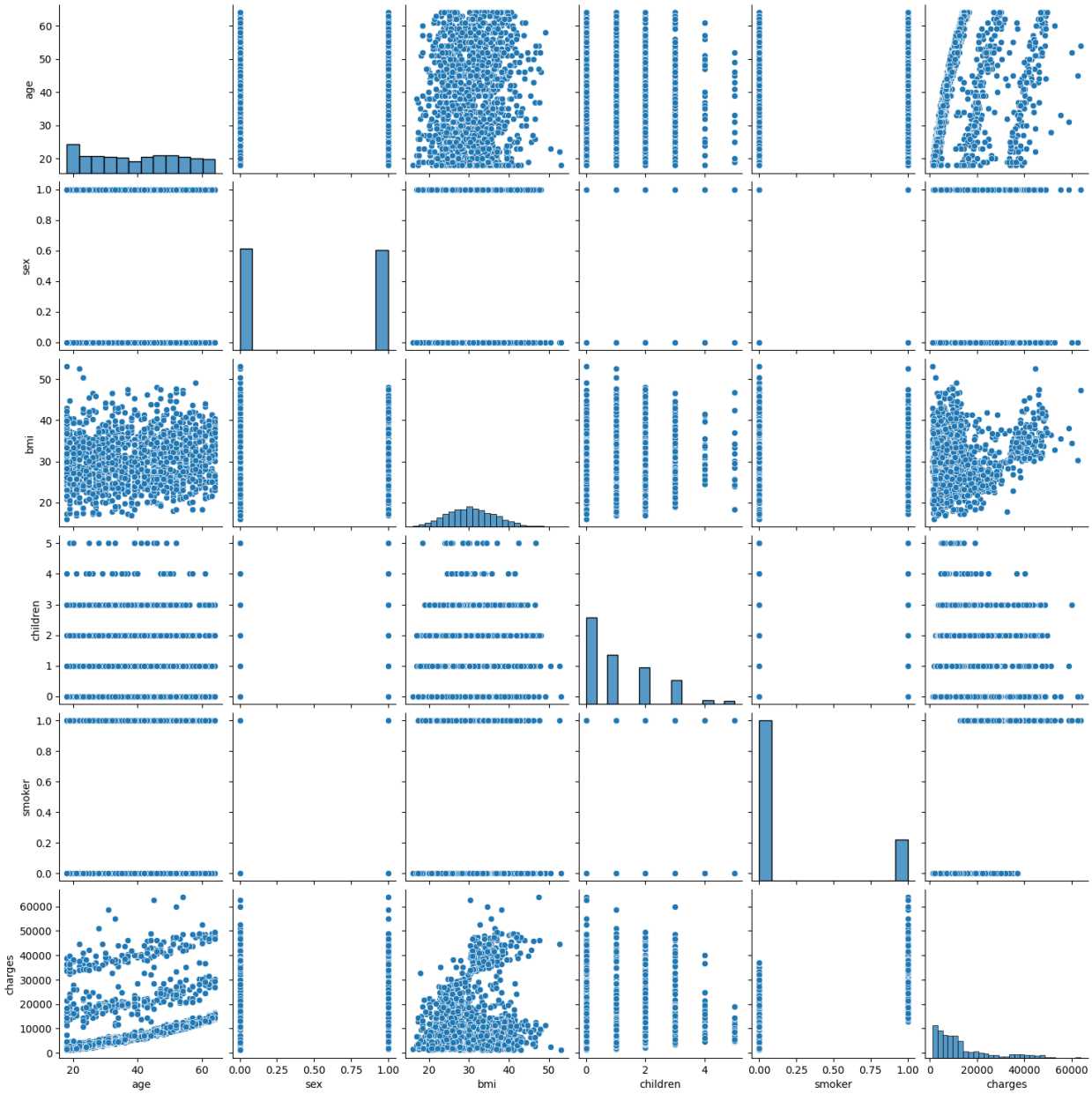
```
0.06738935083963239
```

In [21]:

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

Out[21]:

```
<seaborn.axisgrid.PairGrid at 0x269335f2ad0>
```



Step-4:- Data Modelling

Applying LinearRegression

In [22]:

```
# Fit Model to the training set

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[22]:

```
LinearRegression
LinearRegression()
```

In [23]:

```
# Predict the test set result

y_pred = regressor.predict(X_test)
```

In [24]:

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

In [25]:

```
score
```

Out[25]:

```
0.757820140931517
```

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

Applying LogisticRegression

In [26]:

```
#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 [27]:

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

```
C:\Users\thara\AppData\Local\Programs\Python\Python310\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 to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[27]:

LogisticRegression

LogisticRegression(max_iter=10000)

In [28]:

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

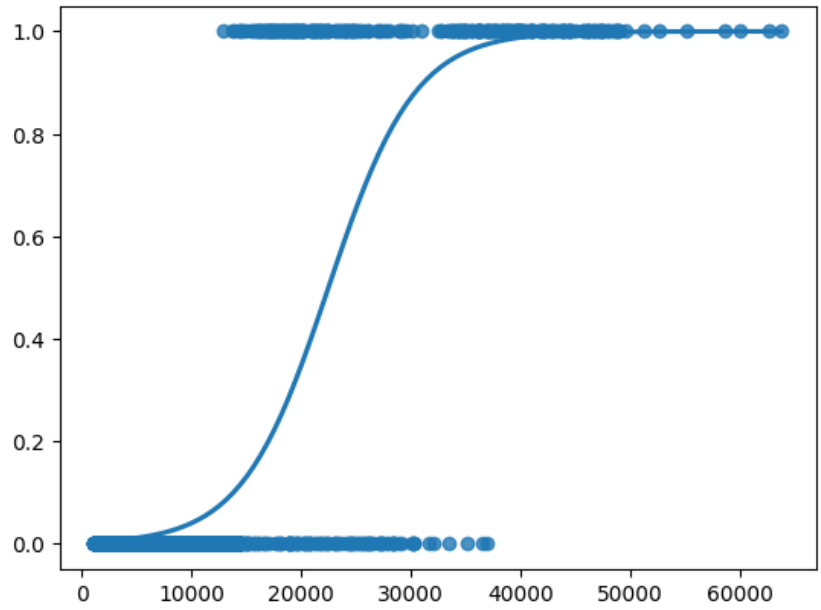
```
0.9253731343283582
```

In [29]:

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

Out[29]:

```
<Axes: >
```



We got the best fit score for Logistic Regression.

Applying Decision Tree

we are going to check that if we may get better accuracy by implementing Decision Tree and RandomForest

In [43]:

```
#Decision tree

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

Out[43]:

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

In [44]:

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

```
0.900497512437811
```

Random Forest

In [45]:

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

```
C:\Users\thara\AppData\Local\Temp\ipykernel_25876\1232785509.py:4: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  rfc.fit(X_train,y_train)
```

Out[45]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [49]:

```
rf = RandomForestClassifier()
```

In [50]:

```
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 [51]:

```
from sklearn.model_selection import GridSearchCV

grid_search = GridSearchCV(estimator=rf,param_grid=params,cv = 2, scoring='accuracy')
grid_search.fit(x_train,y_train)
```

```
C:\Users\thara\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\model_selection\_valid
ation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\thara\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\model_selection\_valid
ation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\thara\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\model_selection\_valid
ation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\thara\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\model_selection\_valid
ation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\thara\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\model_selection\_valid
ation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().
```

In [52]:

```
grid_search.best_score_
```

Out[52]:

```
0.9219193250242501
```

In [53]:

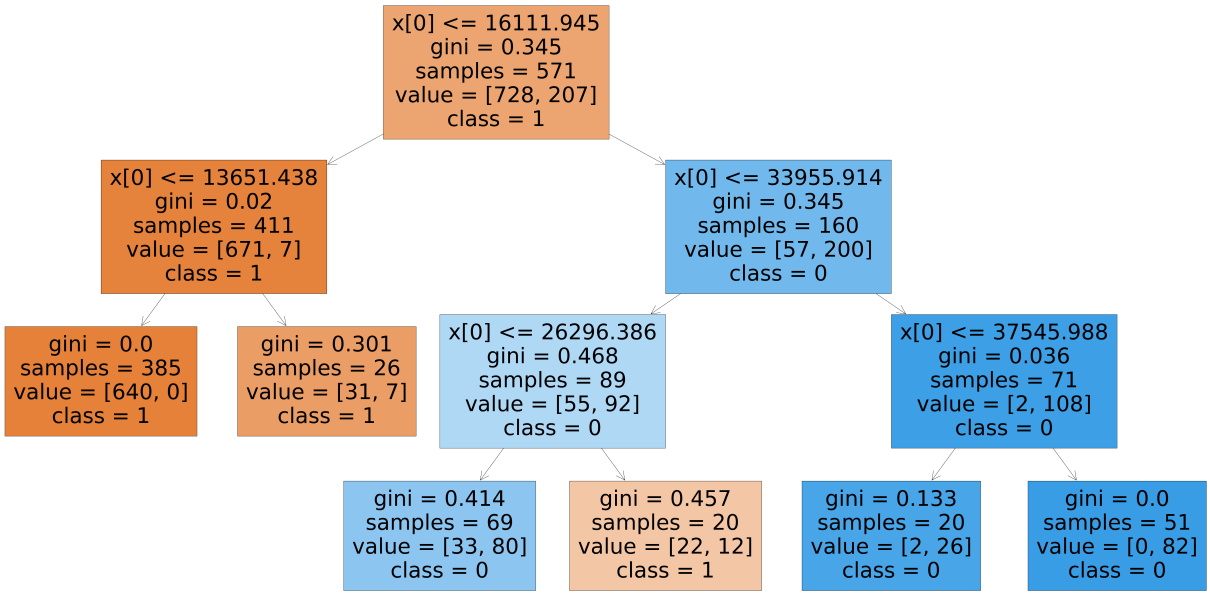
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[53]:

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=3, min_samples_leaf=20, n_estimators=50)
```

In [61]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],class_names=['1','0'],filled=True);
```



In [63]:

```
score=rfc.score(X_test,y_test)
print(score)
```

```
0.7786069651741293
```

Conclusion:

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

In [ ]:

```

```