STEP1:-Problem statement: To predict how best the data fits and which model suits

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
```

Data Collection

In [2]:

```
df=pd.read_csv(r"C:\Users\smb06\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 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.shape
```

Out[5]:

```
(1338, 7)
```

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]:

```
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
```

To find null values

In [8]:

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

Out[8]:

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

To find duplicate values

In [9]:

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

Out[9]:

```
1
```

In [10]:

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

Out[10]:

	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 [11]:

```
T={"sex":{"female":1,"male":0}}
df=df.replace(T)
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 [12]:

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

```
      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 x 7 columns]
```

Feature Scaling:To Split the data into train and test data

In [13]:

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

In [14]:

```
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)
```

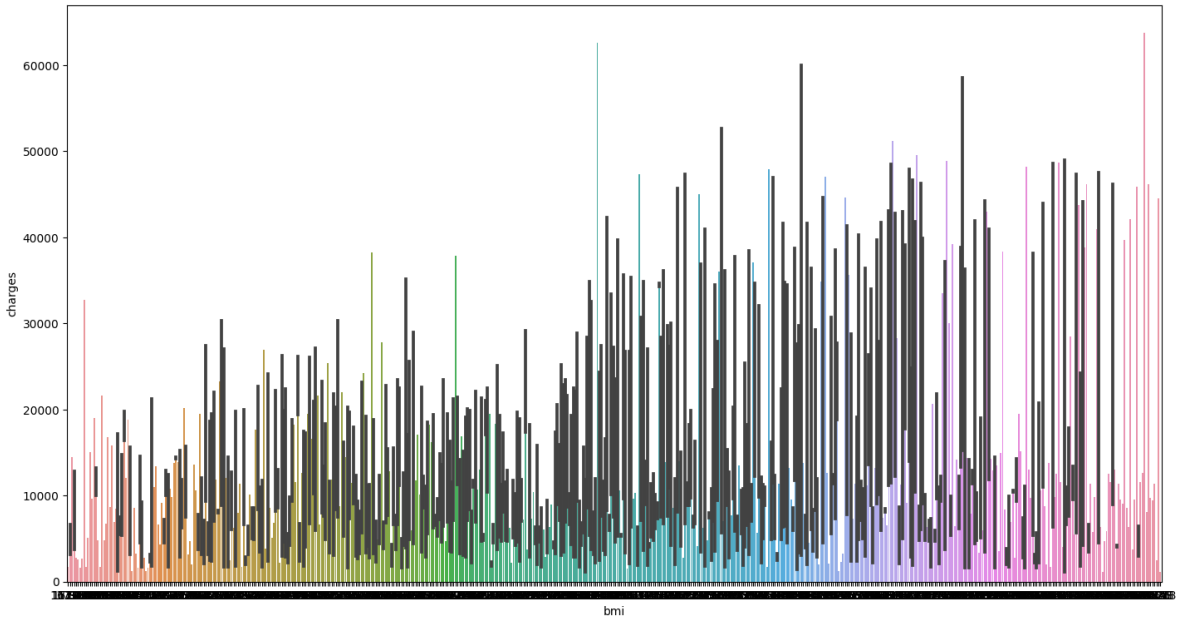
STEP 3:-Visualization

In [15]:

```
#explore expenses relationships
#a bar plot shows the relationship between a numeric and a categoric variable
plt.figure(figsize=(17,9))
sns.barplot(x=df.bmi,y=df.charges)
```

Out[15]:

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

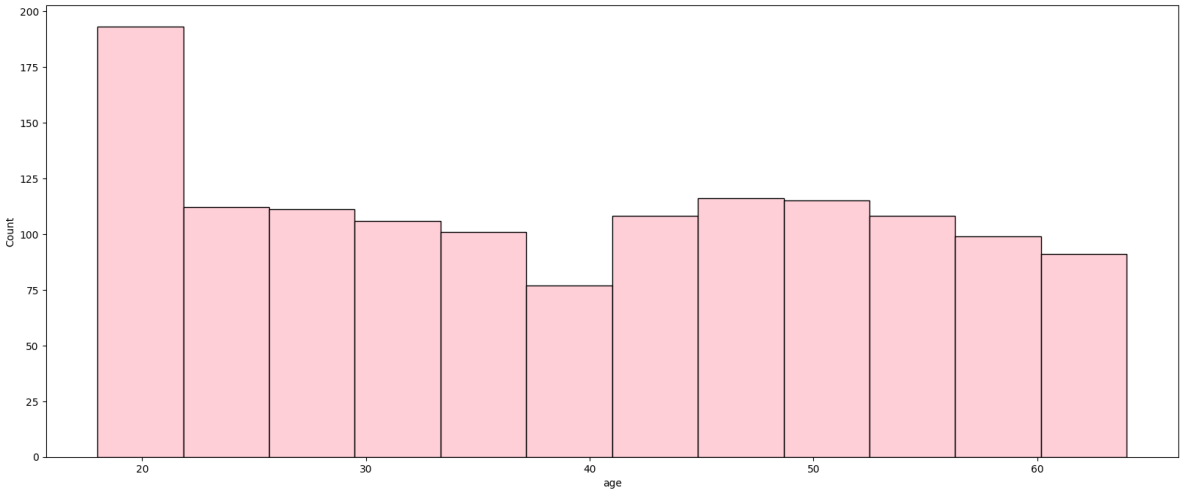


In [16]:

```
#A pie chart is a graphical representation technique that displays data in a circular-shaped
plt.figure(figsize=(20,8))
sns.histplot(data=df,x='age',color='pink')
```

Out[16]:

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

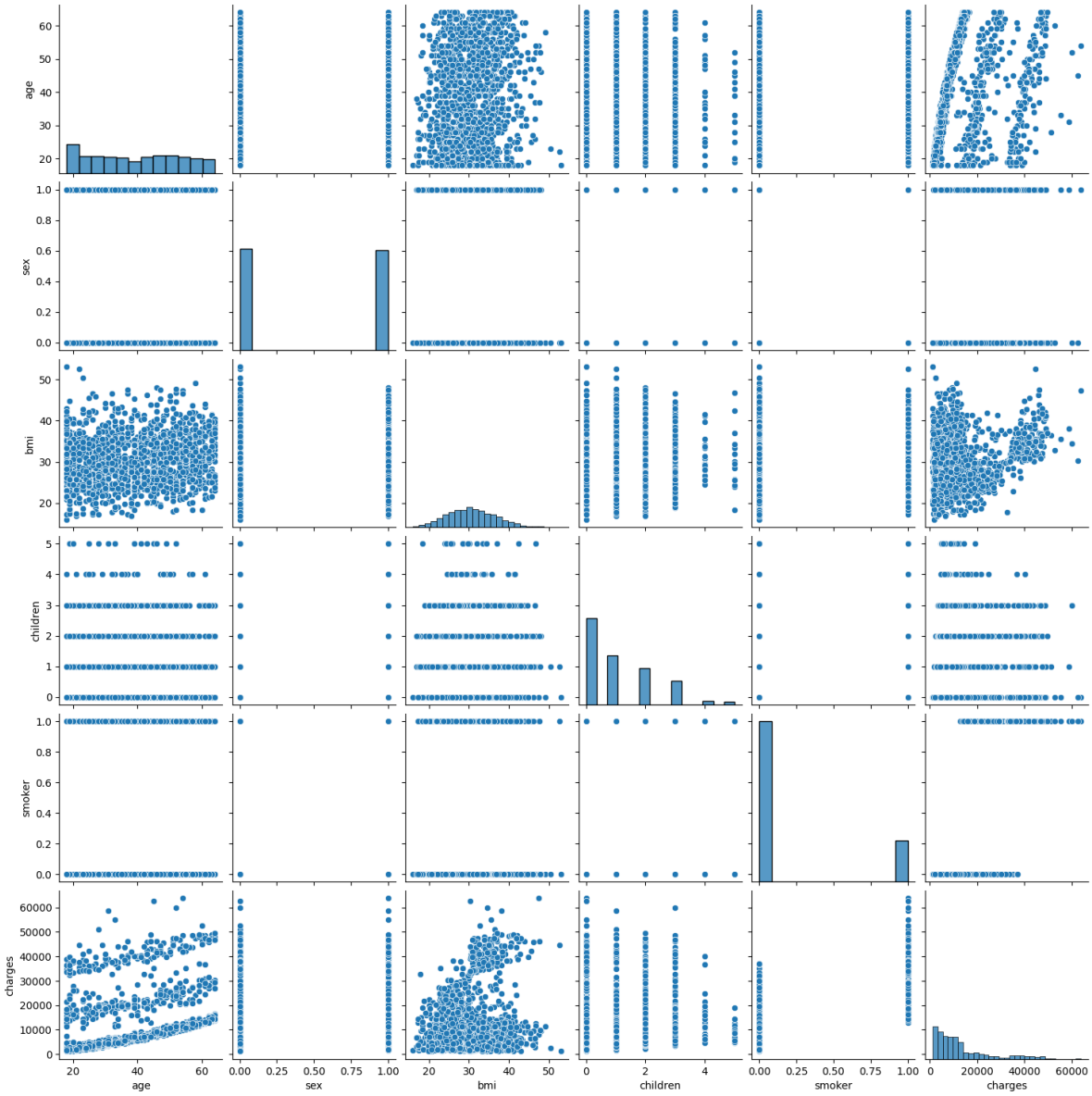


In [17]:

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

Out[17]:

```
<seaborn.axisgrid.PairGrid at 0x24a3795df50>
```

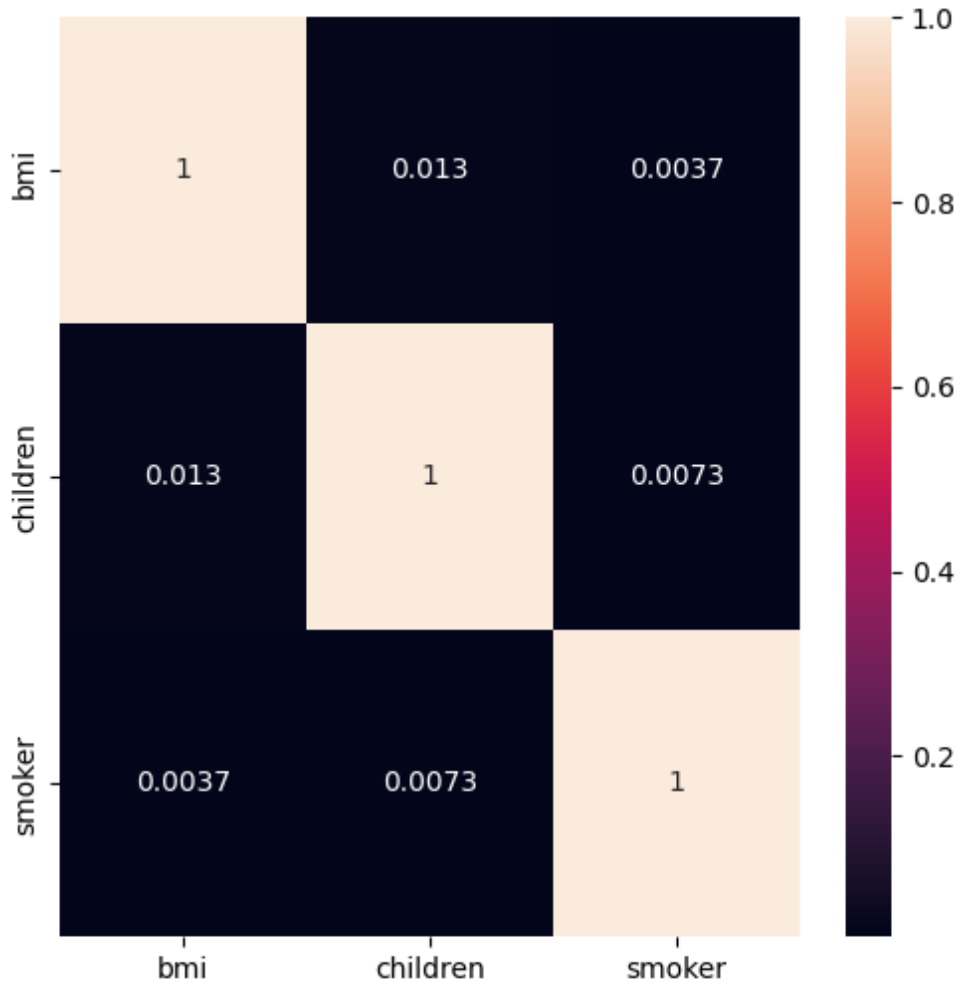


In [18]:

```
Insuranced=df[['bmi','children','smoker']]
plt.figure(figsize=(6,6))
sns.heatmap(Insuranced.corr(),annot=True)
```

Out[18]:

<Axes: >



Step 4:- Data Modelling

Linear Regression

In [23]:

```
#fit model to the training set

from sklearn.linear_model import LinearRegression
regression = LinearRegression()
regression.fit(x_train,y_train)
```

Out[23]:

```
LinearRegression()
```

In [24]:

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

```
0.7206352641782701
```

LOGISTIC REGRESSION

In [27]:

```
#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 [28]:

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

```
C:\Users\smb06\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\utils\validation.py:1143: DataConversionWarning: A column-vector y was pa
ssed 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[28]:

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

In [29]:

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

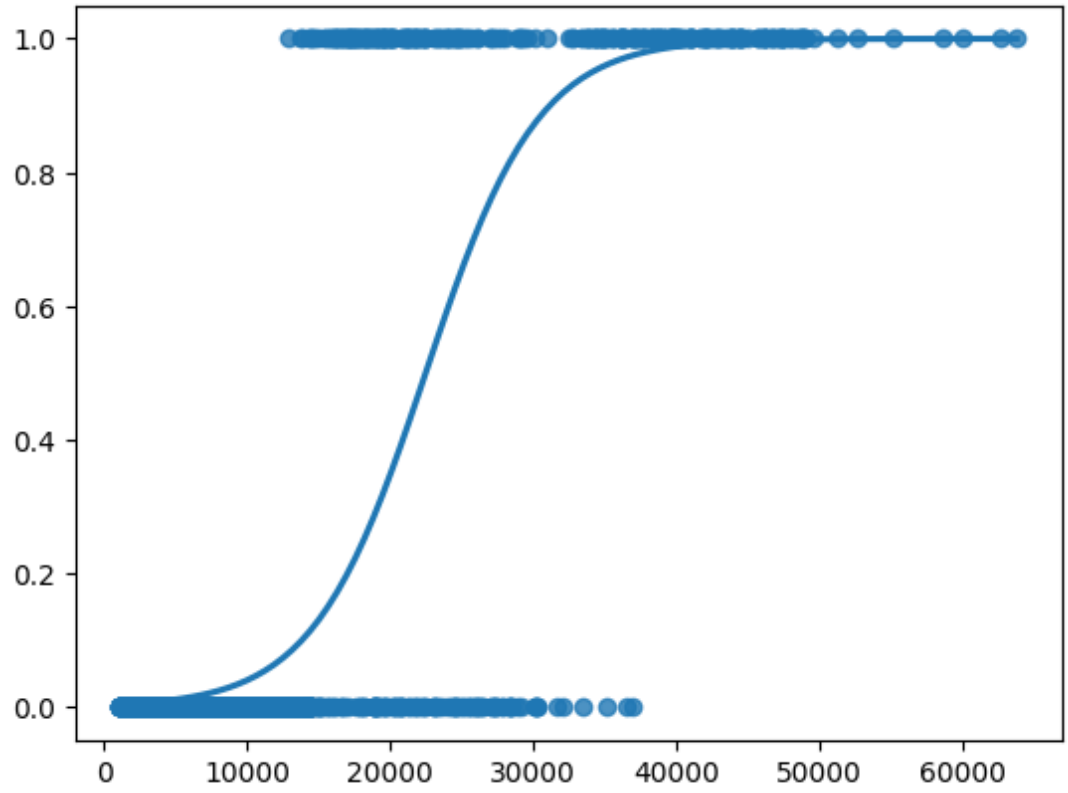
```
0.9253731343283582
```

In [30]:

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

Out[30]:

```
<Axes: >
```



We got the best fit curve for Logistic Regression .Now

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

Decision Tree

In [32]:

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

Out[32]:

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

In [33]:

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

```
0.900497512437811
```

RANDOM FOREST

In [35]:

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

```
C:\Users\smb06\AppData\Local\Temp\ipykernel_14736\2638823938.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[35]:

RandomForestClassifier

RandomForestClassifier()

In [36]:

```
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 [37]:

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

In [39]:

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

```
C:\Users\smb06\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
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\smb06\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
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\smb06\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
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\smb06\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
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

In [40]:

```
grid_search.best_score_
```

Out[40]:

```
0.9219193250242501
```

In [41]:

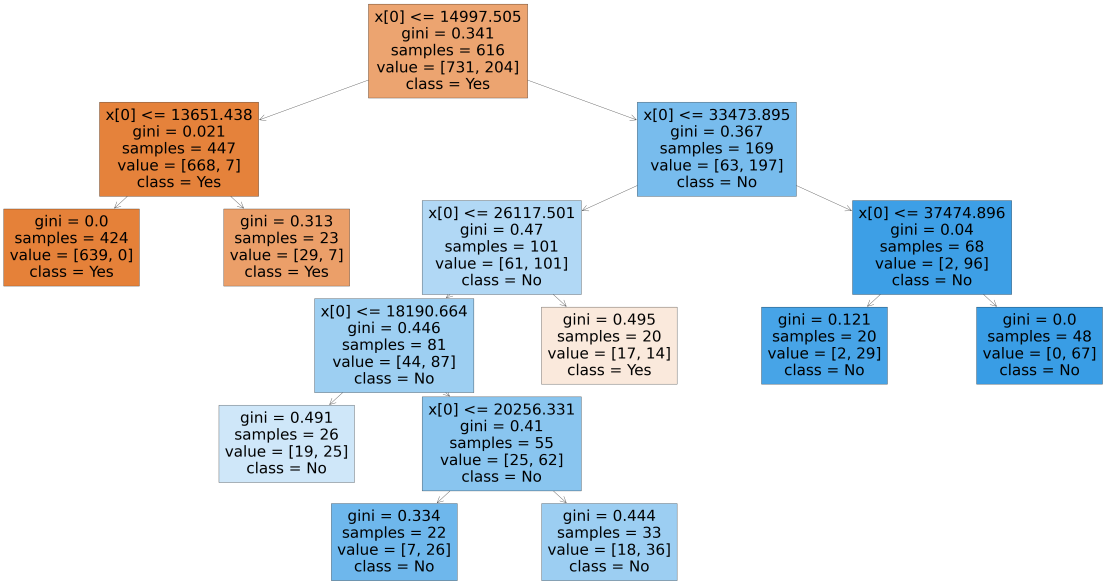
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[41]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, min_samples_leaf=20, n_estimators=30)
```

In [43]:

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



In [44]:

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

```
0.900497512437811
```

CONCLUSION :

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

In [ ]:

```

```