PROBLEM STATEMENT:Which model is suitable for the insurance dataset

1.DATA COLLECTION

In [24]:

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

In [25]:

```
df=pd.read_csv(r"C:\Users\Teju\Downloads\insurance.csv")
df
```

Out[25]:

	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

2.DATA CLEANING AND PREPROCESSING

In [26]:

```
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 [27]:

```
df.head()
```

Out[27]:

	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 [28]:

```
df.tail()
```

Out[28]:

	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 [29]:

```
df.columns
```

Out[29]:

```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dt
ype='object')
```

In [30]:

```
df.describe()
```

Out[30]:

	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 [31]:

```
df.shape
```

Out[31]:

```
(1338, 7)
```

To Find The Null Values

In [32]:

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

Out[32]:

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

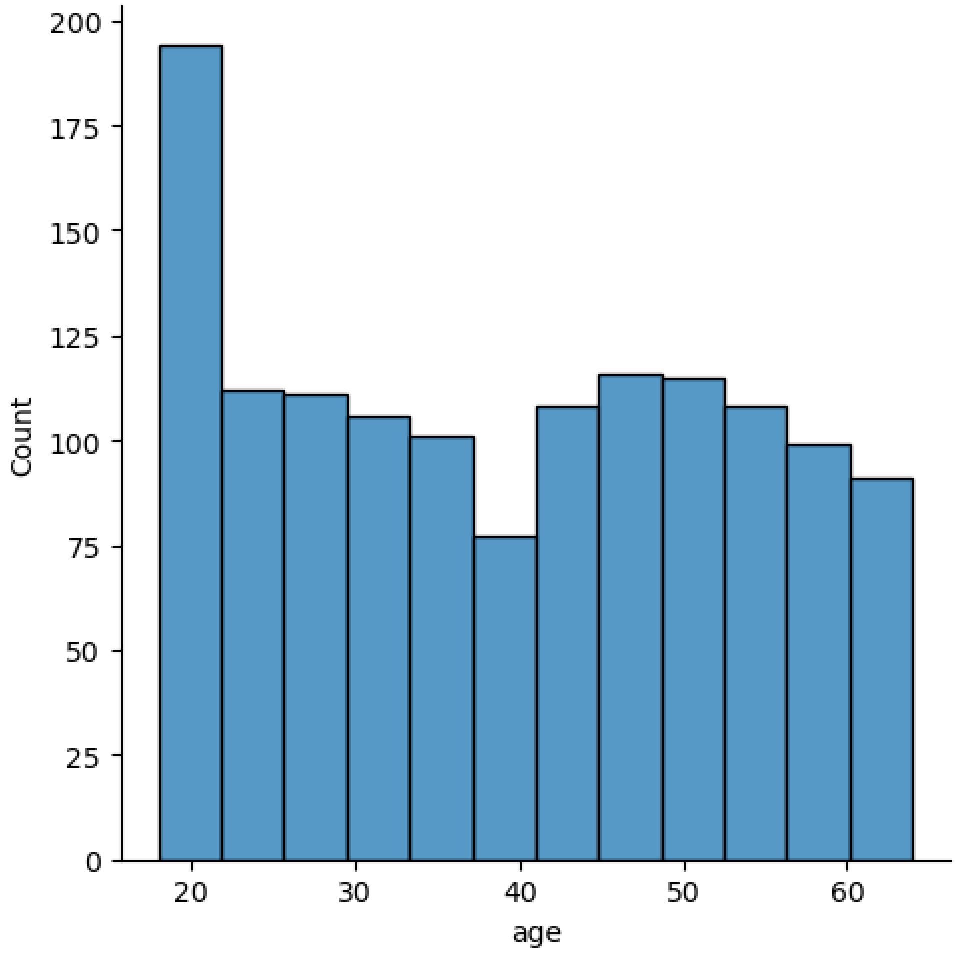
Data Visualization

In [33]:

```
sns.displot(df['age'])
```

Out[33]:

```
<seaborn.axisgrid.FacetGrid at 0x25576883e50>
```

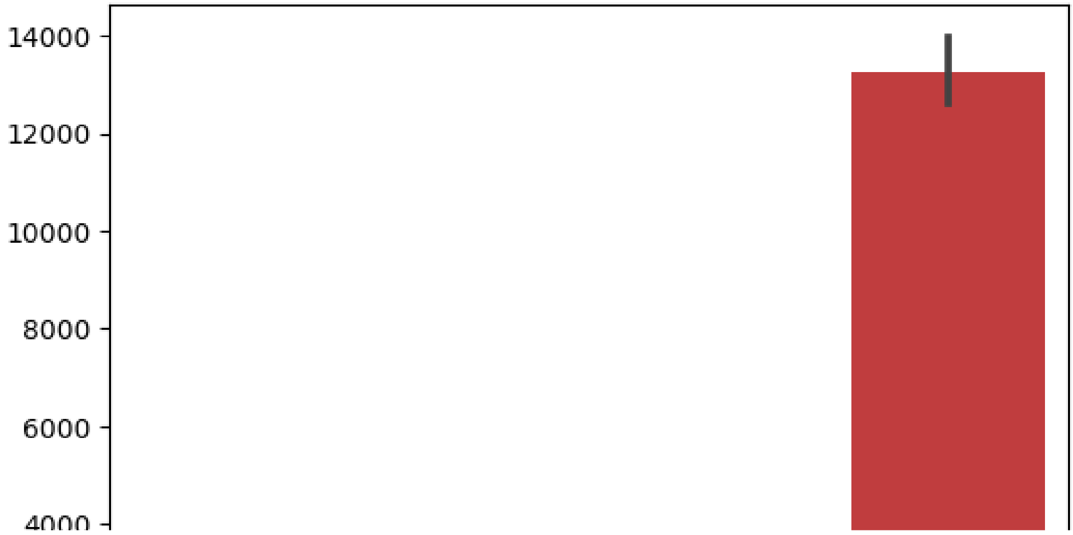


In [34]:

```
sns.barplot(df)
```

Out[34]:

```
<Axes: >
```

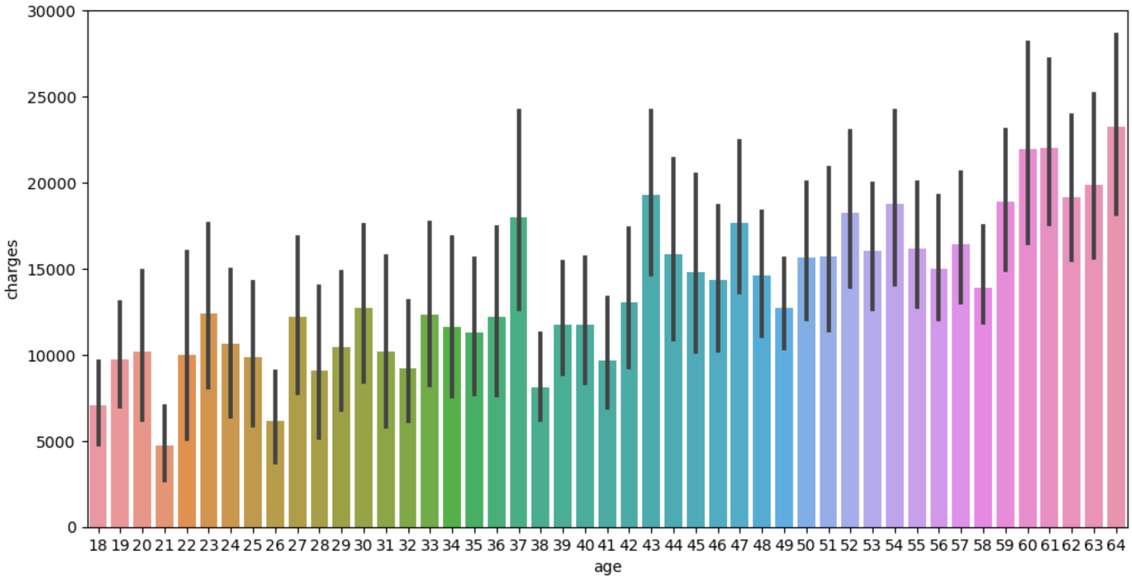


In [57]:

```
plt.figure(figsize=(12,6))
sns.barplot(x='age',y='charges',data=df)
```

Out[57]:

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

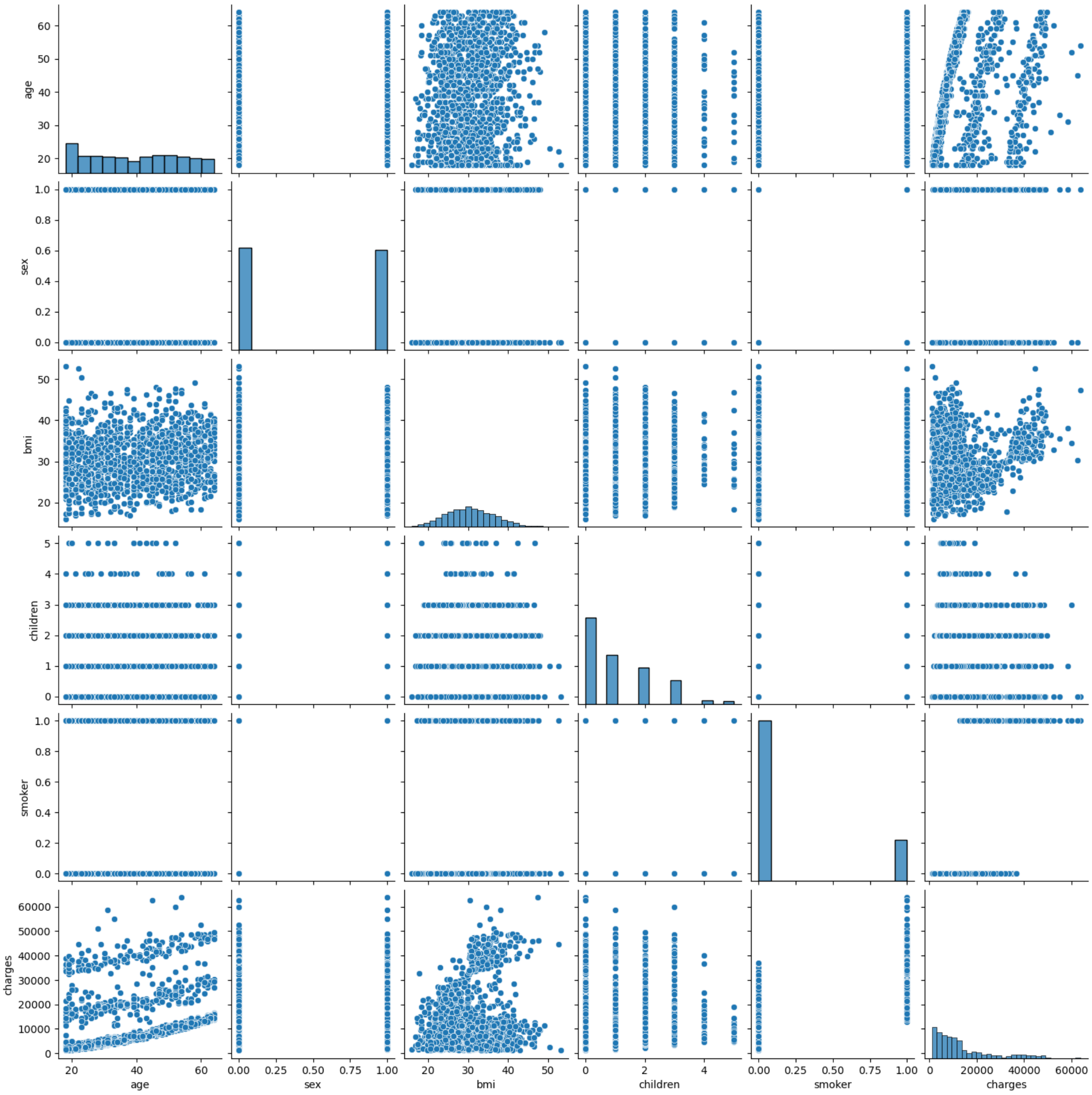


In [58]:

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

Out[58]:

```
<seaborn.axisgrid.PairGrid at 0x255785f72b0>
```



In [35]:

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

Out[35]:

	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 [36]:

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

Out[36]:

	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

1338 rows × 7 columns

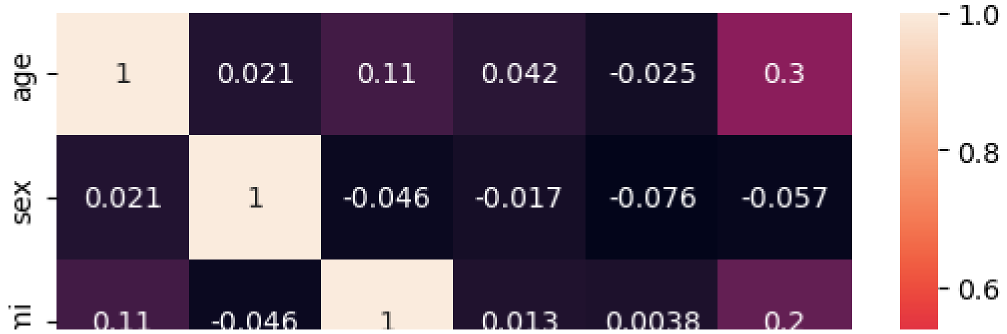
In [37]:

```
Ins=df[['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges']]
sns.heatmap(Ins.corr(),annot=True)
```

```
C:\Users\Teju\AppData\Local\Temp\ipykernel_1680\3993229711.py:2: Future
Warning: The default value of numeric_only in DataFrame.corr is depreca
ted. In a future version, it will default to False. Select only valid c
olumns or specify the value of numeric_only to silence this warning.
  sns.heatmap(Ins.corr(),annot=True)
```

Out[37]:

```
<Axes: >
```



Feature Scaling- To split the data into training and test data

In [38]:

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

Linear Regression

In [39]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
ln=LinearRegression()
ln.fit(x_train,y_train)
```

Out[39]:

```
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 [40]:

```
print(ln.intercept_)
score=ln.score(x_test,y_test)
score
```

```
-12401.788652269326
```

Out[40]:

```
0.7680881643600721
```

By using Linear Regression we didn't the accuracy for this model. So we will use Logistic Regression

Logistic Regression

In [41]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [44]:

```
x=np.array(df['smoker']).reshape(-1,1)
x=np.array(df['age']).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=42)
lr=LogisticRegression(max_iter=10000)
```

In [45]:

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

Out[45]:

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

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 [46]:

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

Out[46]:

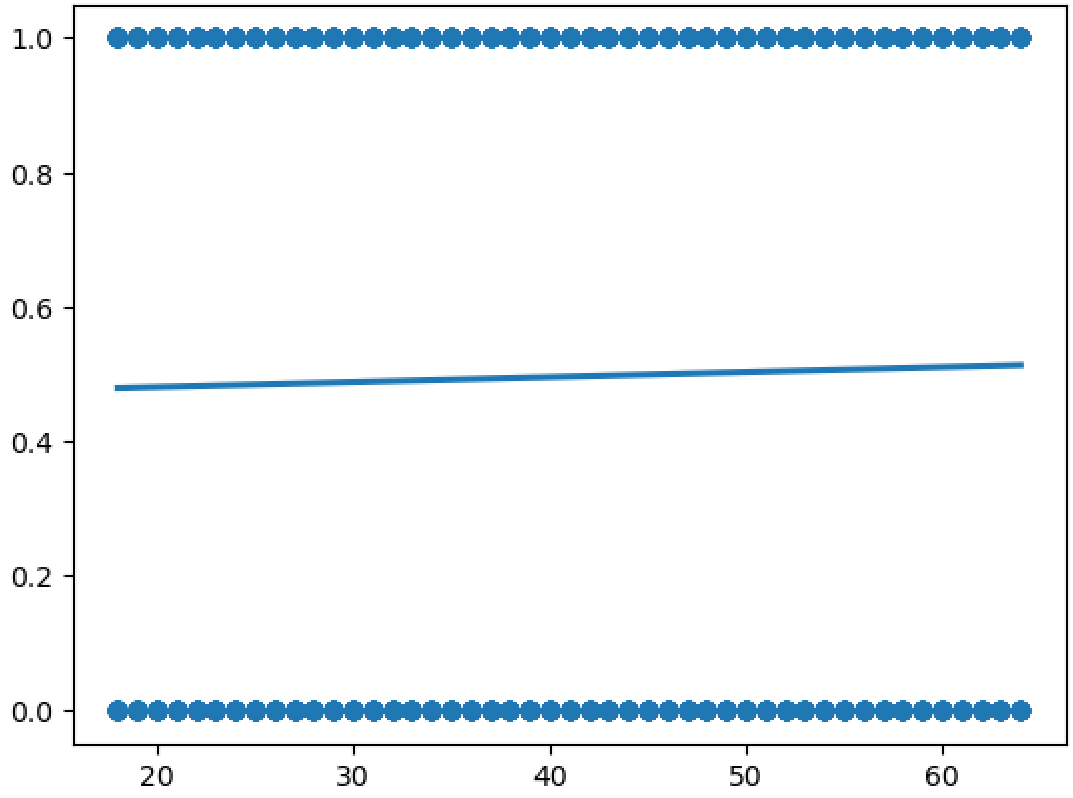
```
0.48756218905472637
```

In [47]:

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

Out[47]:

```
<Axes: >
```

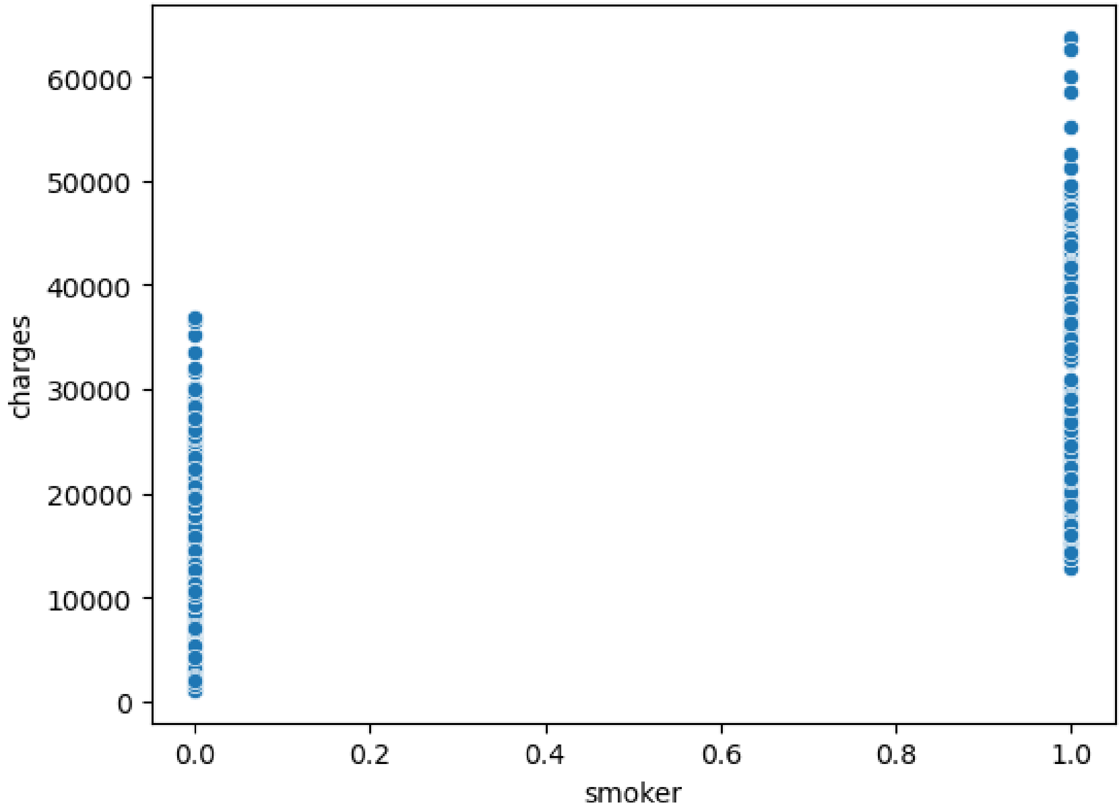


In [48]:

```
sns.scatterplot(data=df,x='smoker',y='charges')
```

Out[48]:

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



Decision Tree

In [49]:

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
```

Out[49]:

```
DecisionTreeClassifier()
```

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 [50]:

```
dtc.score(x_test,y_test)
```

Out[50]:

```
0.3880597014925373
```

Random Forest

In [51]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[51]:

```
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 [52]:

```
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 [53]:

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

In [54]:

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

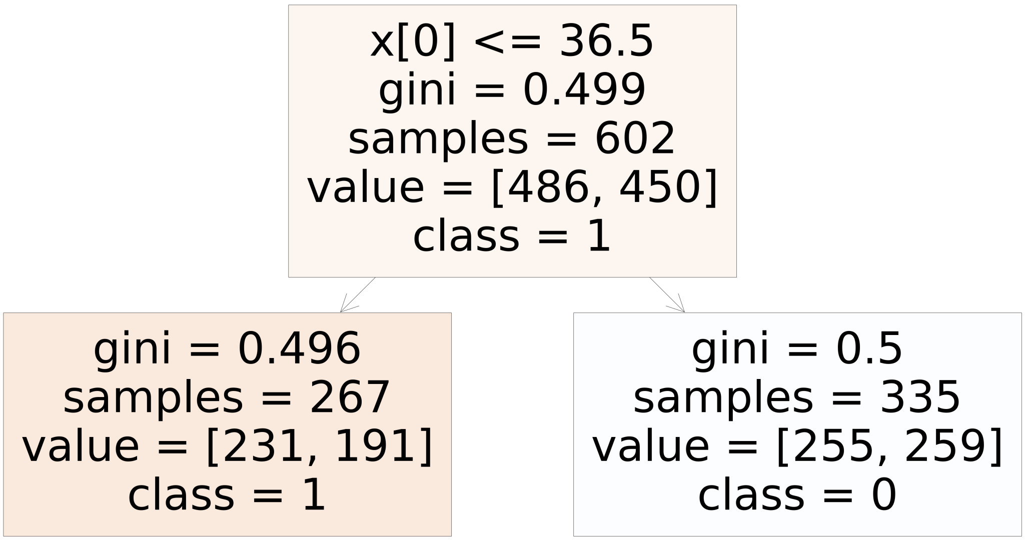
Out[54]:

```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy')
```

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 [62]:

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



In [55]:

```
grid_search.best_score_
```

Out[55]:

```
0.5128205128205128
```

In [56]:

```
grid_search.best_estimator_
```

Out[56]:

```
RandomForestClassifier(max_depth=2, min_samples_leaf=200, 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.

Conclusion:

Based on accuracy of all models we can conclude that Logistic Regression is the best model for given dataset

In [ ]:

```

```