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\munigreeshma\Downloads\insurance (1).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.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 [4]:
```
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]:
```
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]:
```
df.shape
```

Out[6]: (1338, 7)

In [7]:
```
df.describe()
```

Out[7]:

	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

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
```

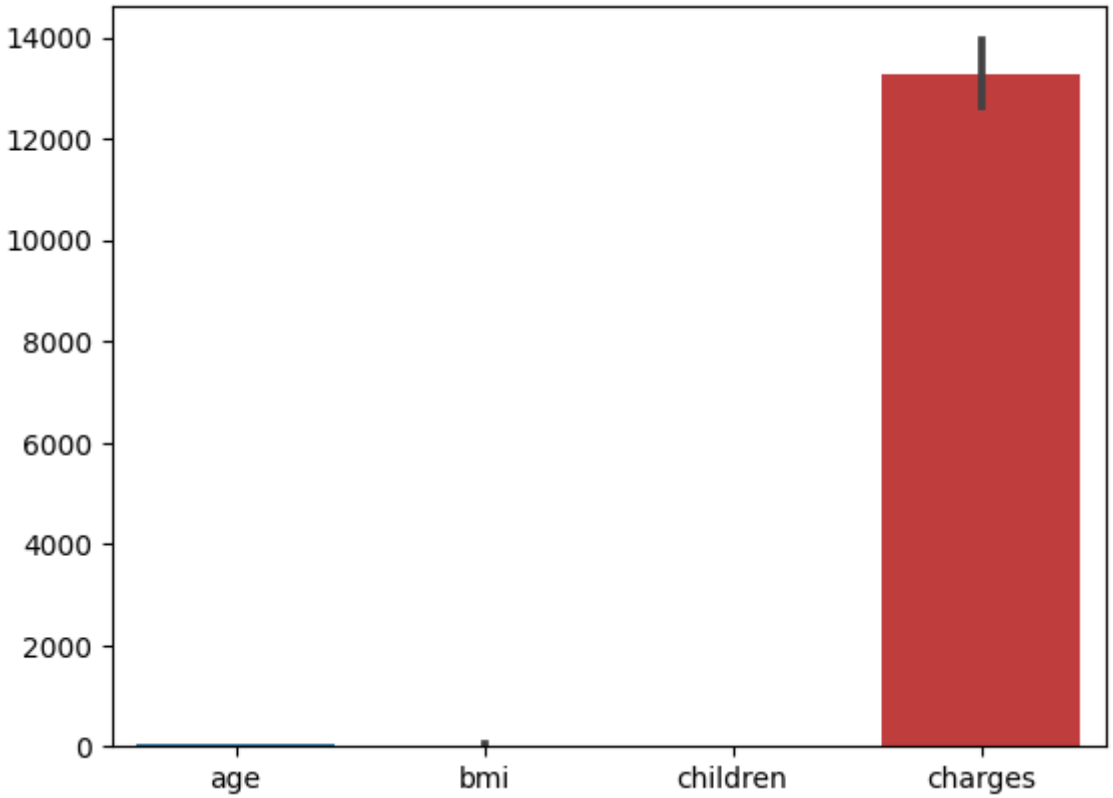
Out[10]:

	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

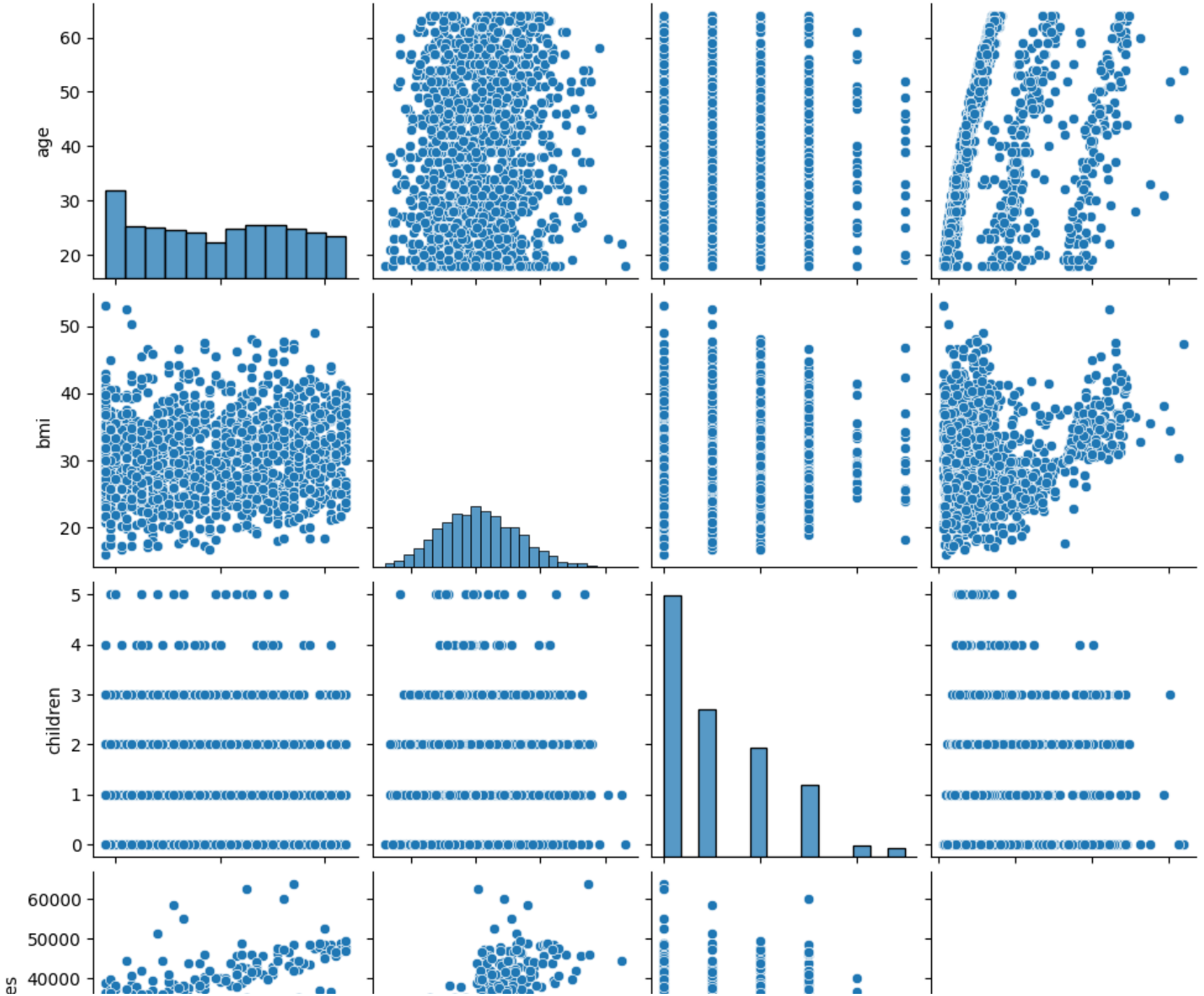
In [11]:
```
sns.barplot(df)
```

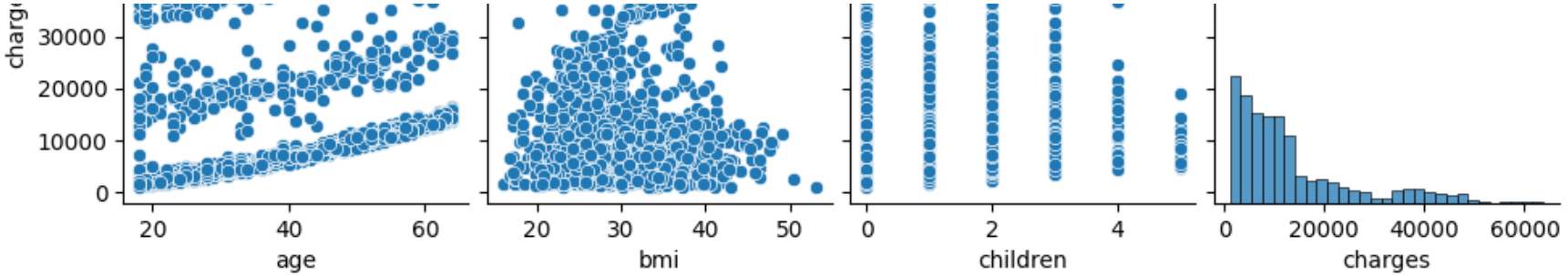
Out[11]: <Axes: >



In [12]:
```
sns.pairplot(df)
```

Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x1f435bb0590>
```


age
bmi
children
60
50
40
30
20
50
40
30
20
5
4
3
2
1
0
60000
50000
40000



In [13]:
```
df.columns
```

Out[13]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
```

In [14]:
```
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 [15]:
```
t={"smoker":{"yes":1,"no":0}}
df=df.replace(t)
df
```

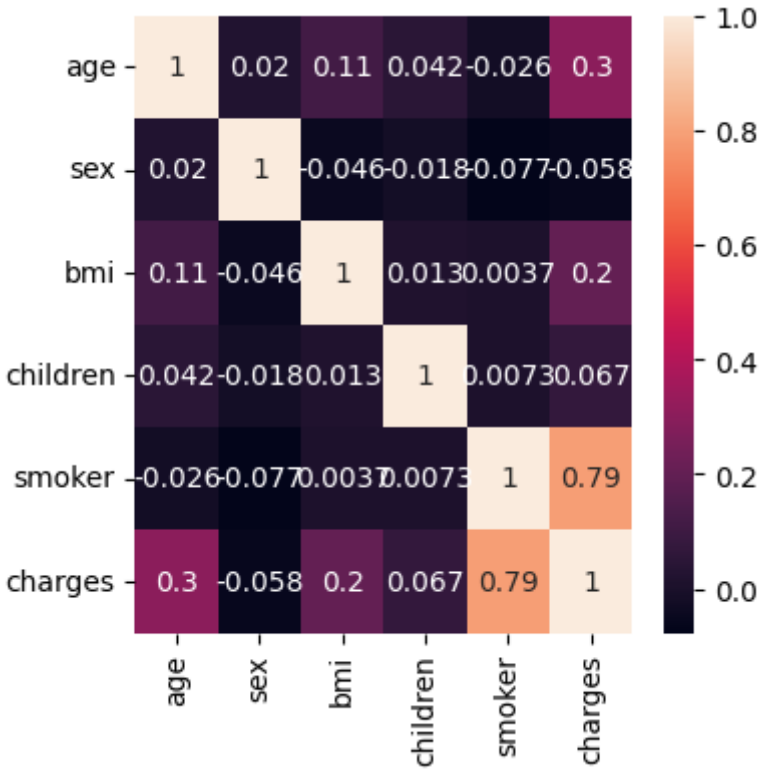
Out[15]:

	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

In [16]:
```
ho=df[['age', 'sex', 'bmi', 'children', 'smoker', 'charges']]
plt.figure(figsize=(4,4))
sns.heatmap(ho.corr(),annot=True)
```

Out[16]: <Axes: >

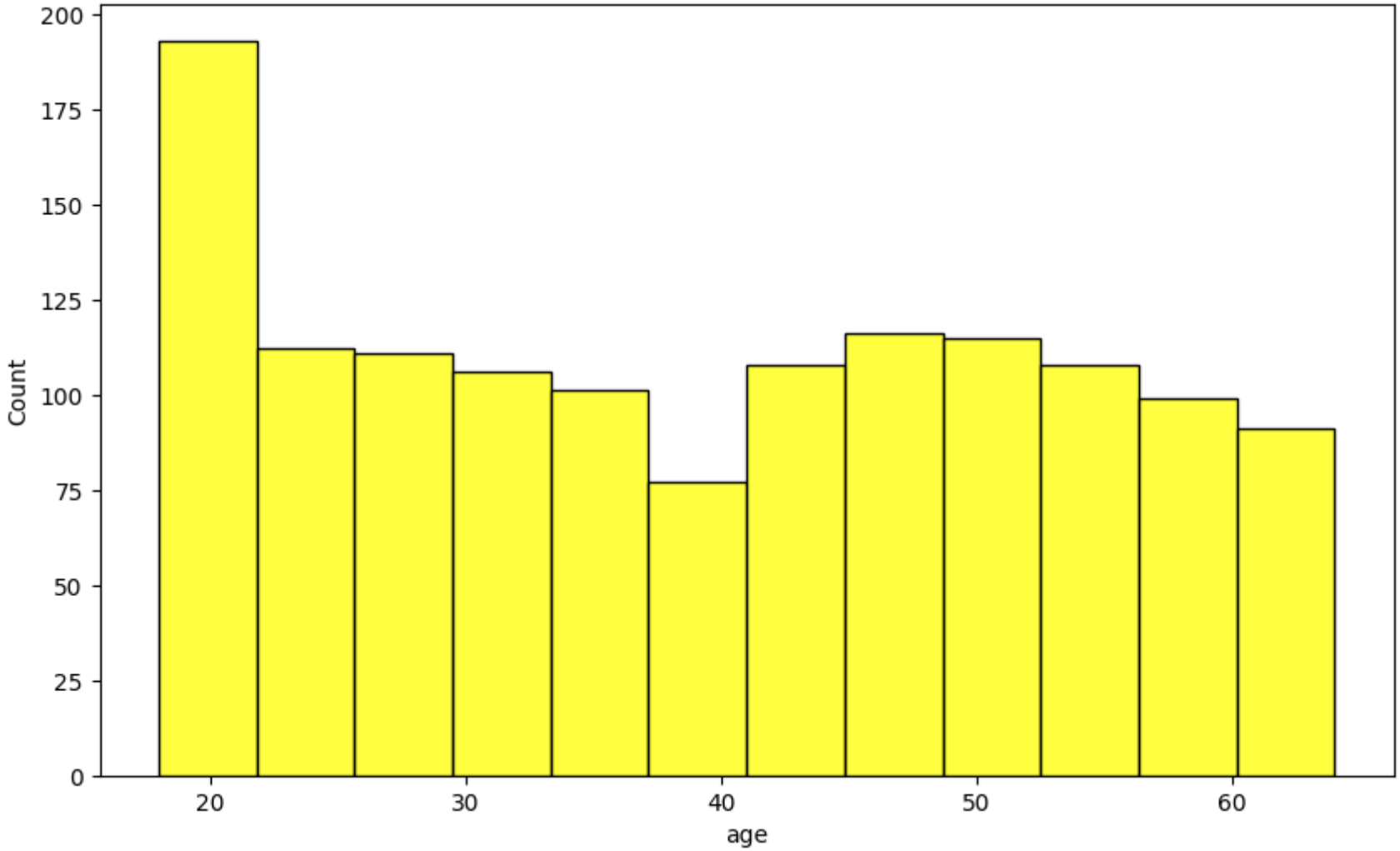


In [17]:
```
x=df[['age', 'sex', 'bmi', 'children', 'smoker']]
y=df['charges']
```

Data Visualize:Visualization The Unique Counts

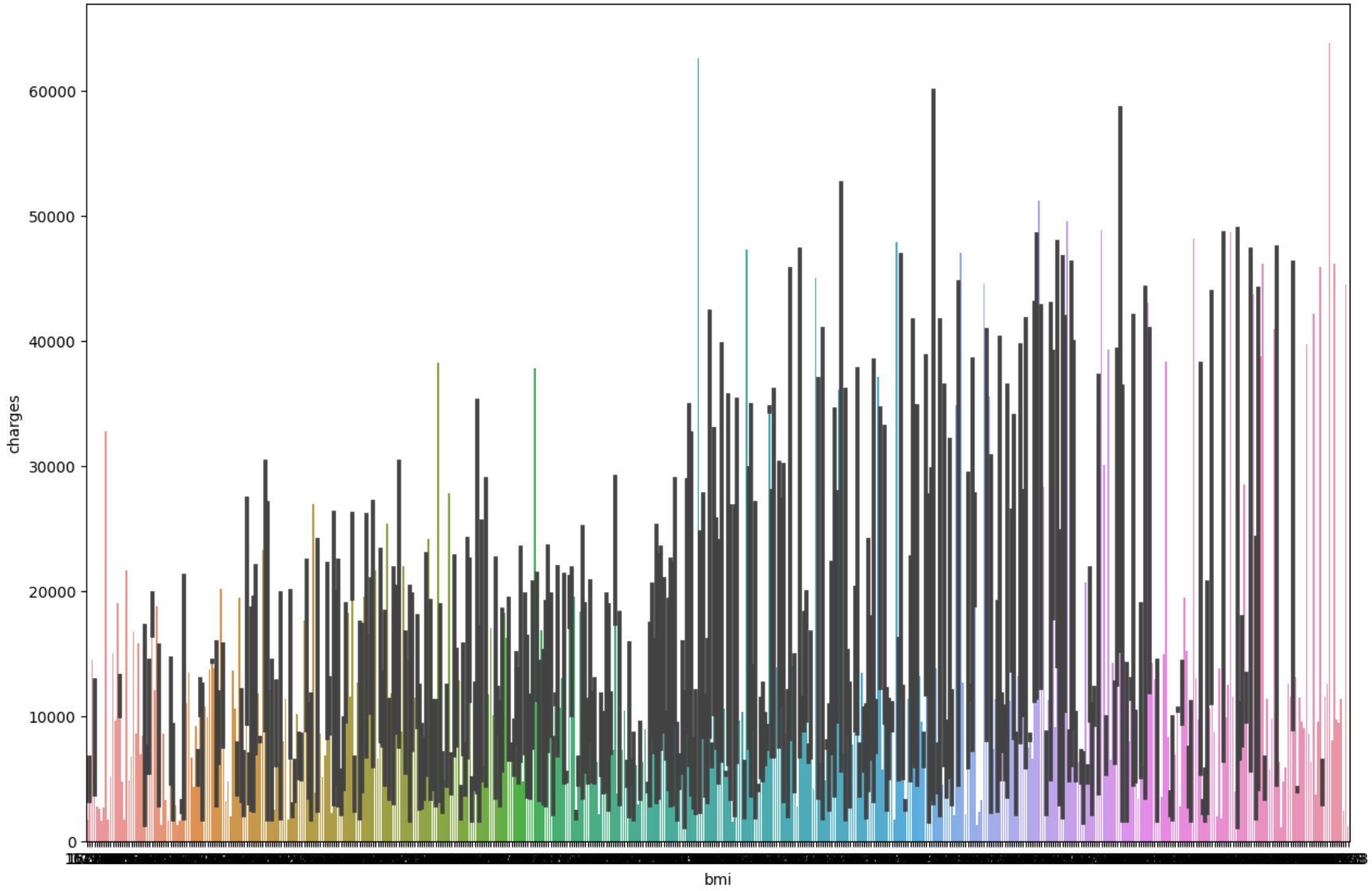
In [25]:
```
plt.figure(figsize=(10,6))
sns.histplot(data=df,x='age',color='yellow')
```

Out[25]: <Axes: xlabel='age', ylabel='Count'>



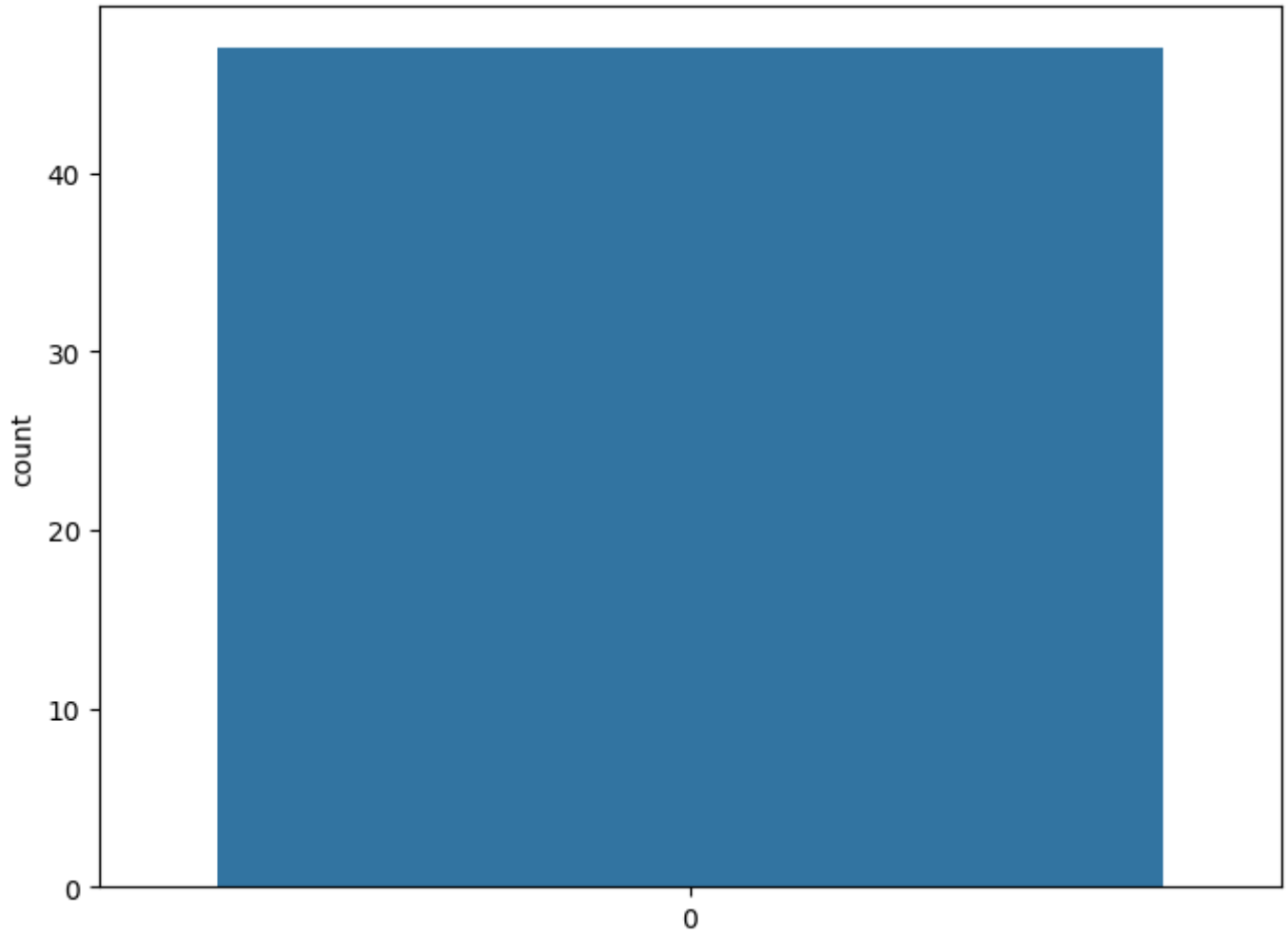
In [26]:
```
plt.figure(figsize=(15,10))
sns.barplot(x=df.bmi,y=df.charges)
```

Out[26]: <Axes: xlabel='bmi', ylabel='charges'>



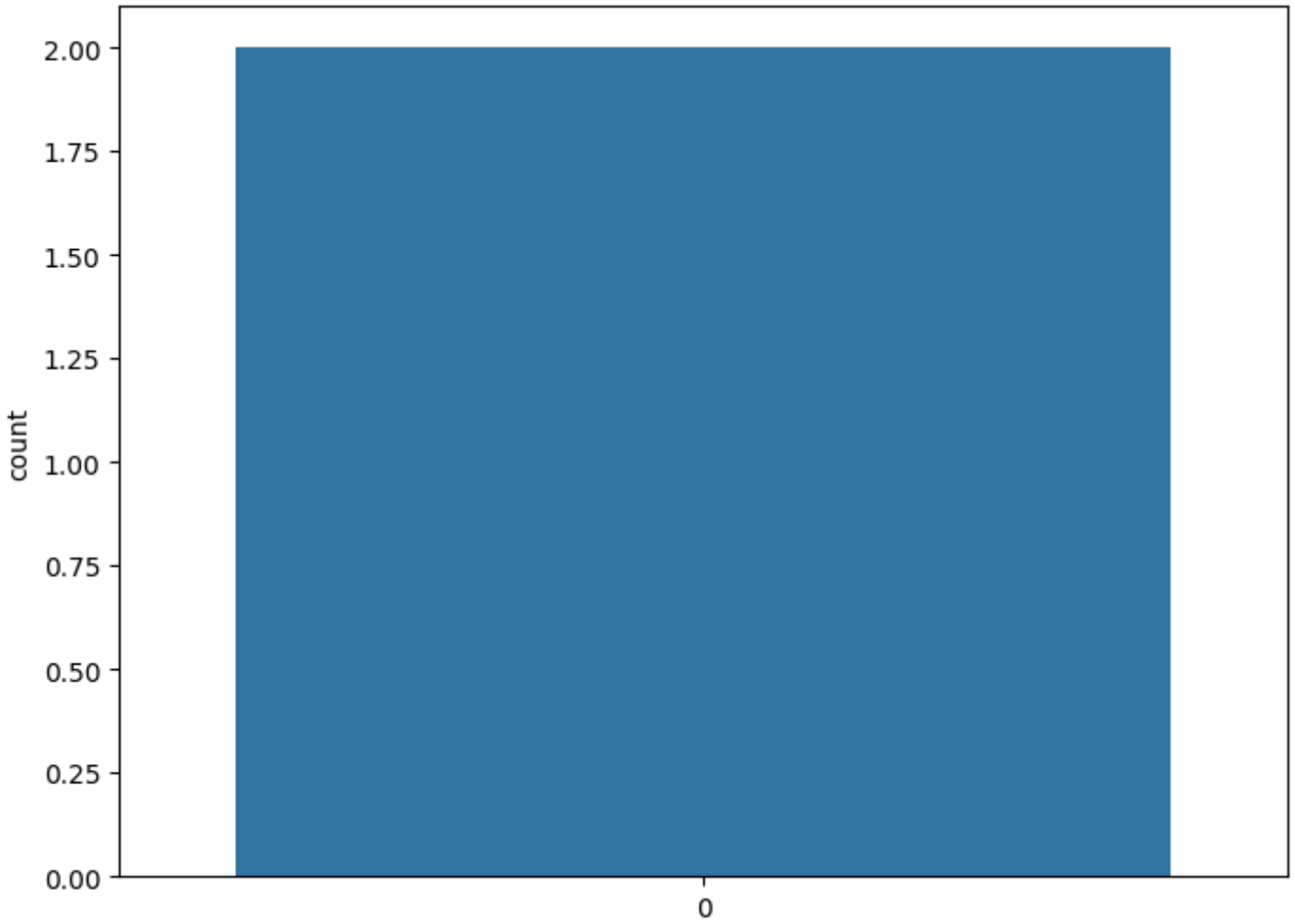
In [28]:
```
plt.figure(figsize=(8,6))
sns.countplot(df['age'].unique())
```

Out[28]: <Axes: ylabel='count'>



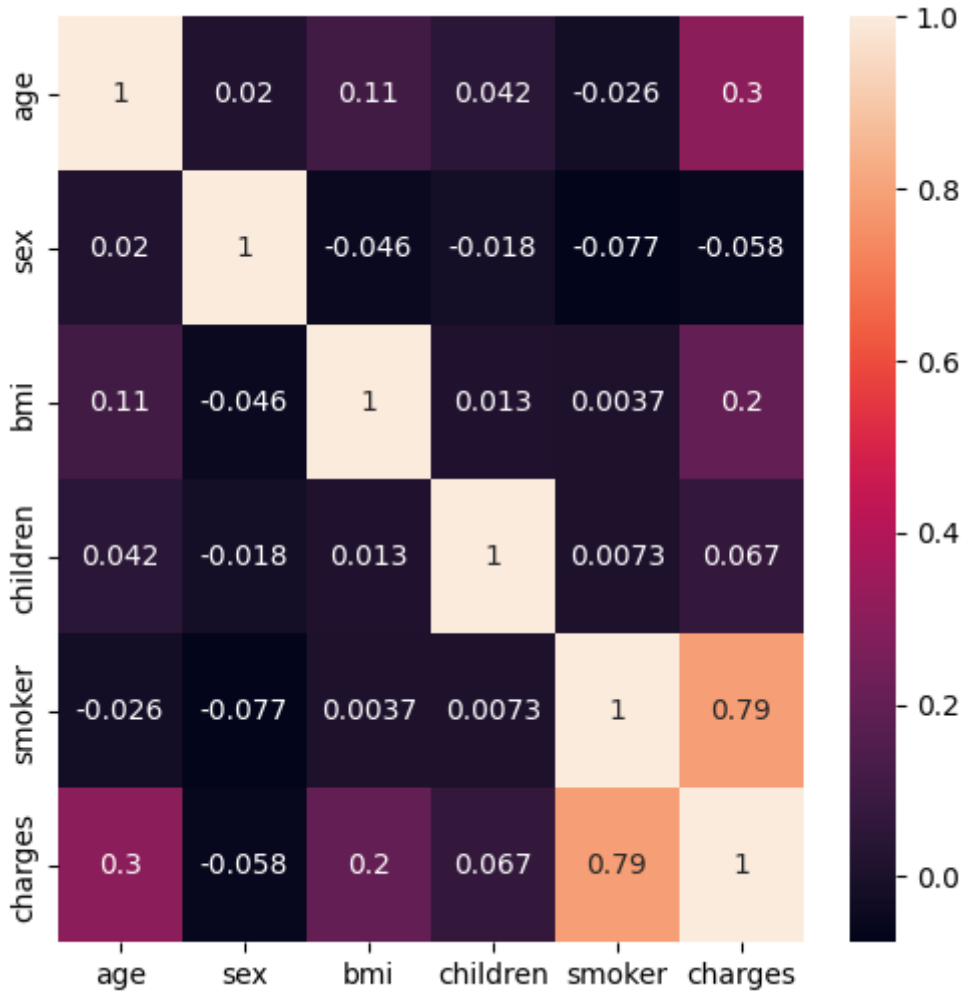
In [58]:
```
plt.figure(figsize=(8,6))
sns.countplot(df['sex'].unique())
```

Out[58]: <Axes: ylabel='count'>



In [31]:
```
Insuranced=df[['age','sex','bmi','children','smoker','charges']]
plt.figure(figsize=(6,6))
sns.heatmap(Insuranced.corr(),annot=True)
```

Out[31]: <Axes: >



To Find Mean And Median

In [35]:
```
print(df["age"].mean(skipna=True))
print(df["age"].median(skipna=True))
```

```
39.222139117427076
39.0
```

In [36]:
```
print(df["children"].mean(skipna=True))
print(df["children"].median(skipna=True))
```

```
1.0957367240089753
1.0
```

In [37]:
```
print(df["bmi"].mean(skipna=True))
print(df["bmi"].median(skipna=True))
```

```
30.66345175766642
30.4
```

In [38]:
```
print(df["charges"].mean(skipna=True))
print(df["charges"].median(skipna=True))
```

```
13279.121486655948
9386.1613
```

Logistic Regresssion

In [40]:
```
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
from sklearn.preprocessing import StandardScaler
lr=LogisticRegression(max_iter=10000)
```

In [41]:
```
lr.fit(x_train,y_train)
```

```
C:\Users\munigreeshma\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 to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[41]:

LogisticRegression

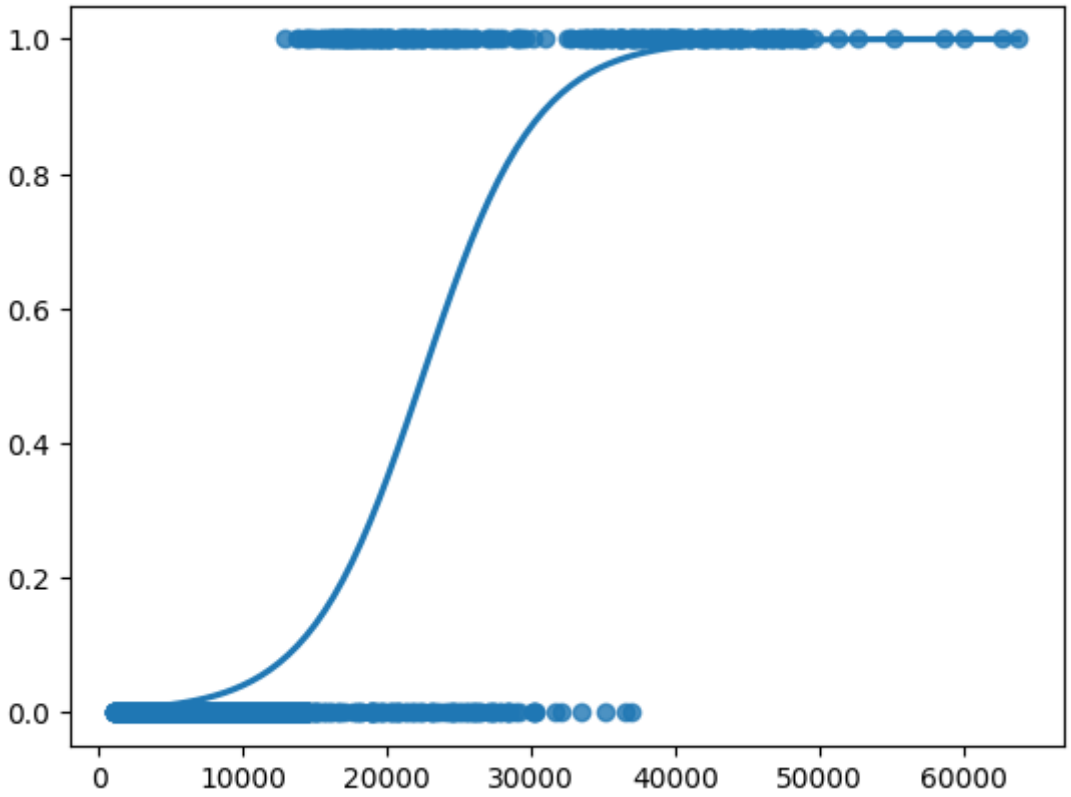
LogisticRegression(max_iter=10000)

In [42]:
```
score=lr.score(x_test,y_test)
print(score)
```

```
0.9253731343283582
```

In [43]:
```
sns.regplot(x=x,y=y,data=df,logistic=True,ci=None)
```

Out[43]: <Axes: >



Decision Tree

In [44]:
```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(random_state=0)
clf.fit(x_train,y_train)
```

Out[44]:
```
DecisionTreeClassifier

DecisionTreeClassifier(random_state=0)
```

In [45]:
```
score=clf.score(x_test,y_test)
print(score)
```

```
0.900497512437811
```

Random Forest

In [47]:
```
#Random forest classifier
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
C:\Users\munigreeshma\AppData\Local\Temp\ipykernel_35692\2638823938.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[47]:
```
RandomForestClassifier

RandomForestClassifier()
```

In [48]:
```
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 [49]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
```

In [50]:
```
grid_search.fit(x_train,y_train)
```

```
C:\Users\munigreeshma\AppData\Local\Programs\Python\Python311\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\munigreeshma\AppData\Local\Programs\Python\Python311\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\munigreeshma\AppData\Local\Programs\Python\Python311\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\munigreeshma\AppData\Local\Programs\Python\Python311\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\munigreeshma\AppData\Local\Programs\Python\Python311\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 [51]:
```
grid_search.best_score_
```

Out[51]: 0.9219193250242501

In [52]:
```
rf_best=grid_search.best_estimator_
rf_best
```

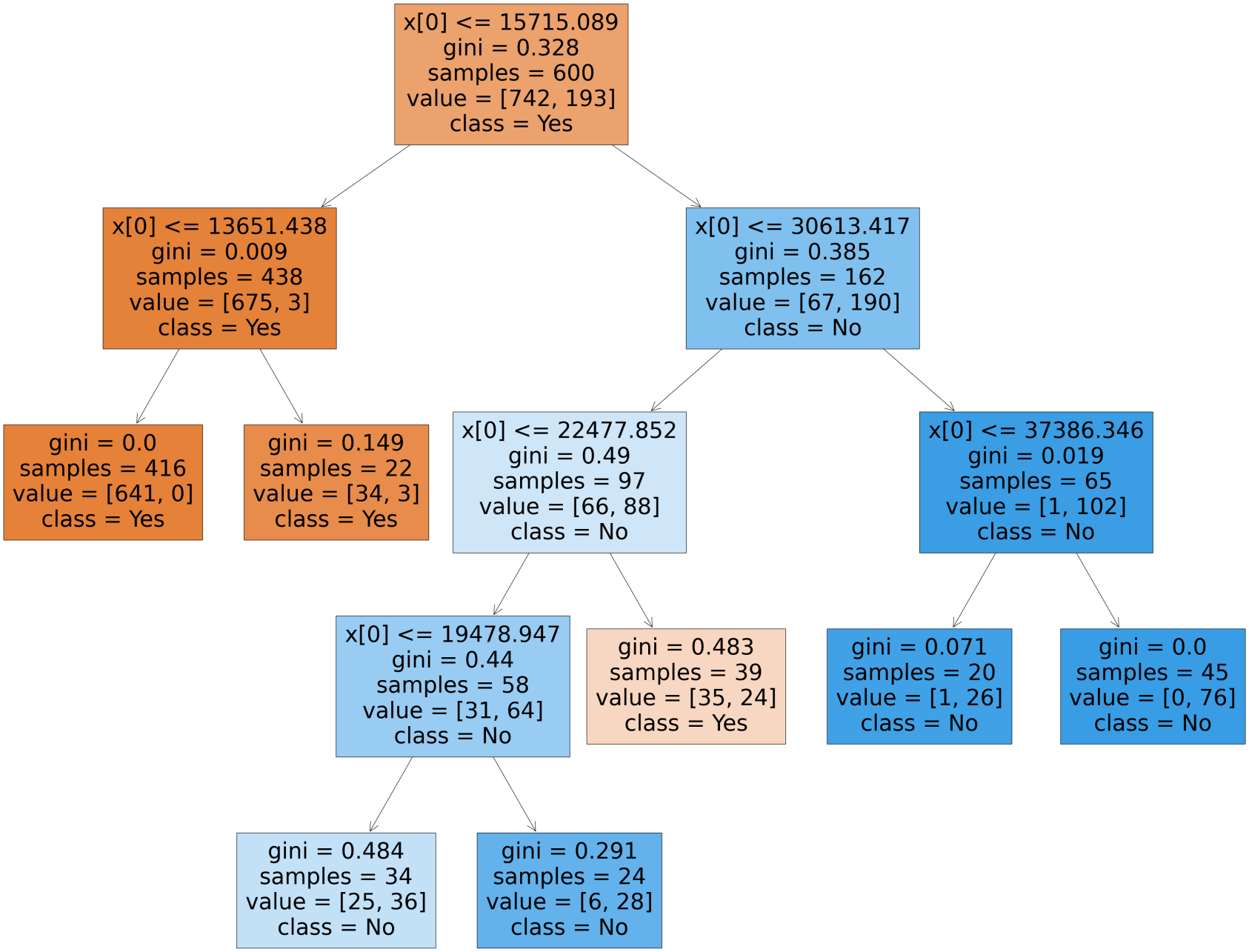
Out[52]:

RandomForestClassifier

RandomForestClassifier(max_depth=5, min_samples_leaf=20, n_estimators=50)

In [54]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(50,40))
plot_tree(rf_best.estimators_[5],class_names=['Yes',"No"],filled=True);
```


x[0] <= 15715.089
gini = 0.328
samples = 600
value = [742, 193]
class = Yes
x[0] <= 13651.438
gini = 0.009
samples = 438
value = [675, 3]
class = Yes
x[0] <= 30613.417
gini = 0.385
samples = 162
value = [67, 190]
class = No
gini = 0.0
samples = 416
value = [641, 0]
class = Yes
gini = 0.149
samples = 22
value = [34, 3]
class = Yes
x[0] <= 22477.852
gini = 0.49
samples = 97
value = [66, 88]
class = No
x[0] <= 37386.346
gini = 0.019
samples = 65
value = [1, 102]
class = No
x[0] <= 19478.947
gini = 0.44
samples = 58
value = [31, 64]
class = No
gini = 0.483
samples = 39
value = [35, 24]
class = Yes
gini = 0.071
samples = 20
value = [1, 26]
class = No
gini = 0.0
samples = 45
value = [0, 76]
class = No
gini = 0.484
samples = 34
value = [25, 36]
class = No
gini = 0.291
samples = 24
value = [6, 28]
class = No

In [55]:
```
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