In [1]:
```python
import pandas as pd
import numpy as np
from sklearn import preprocessing,svm
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import RidgeCV
from sklearn.linear_model import Lasso
from sklearn.linear_model import LassoCV
from sklearn.linear_model import ElasticNet
from sklearn import metrics
```

Data Collection

Read the Data

In [5]:
```
df=pd.read_csv(r"file:///D:\Users\DELL\Downloads\Insurance-1.csv")
df
```

Out[5]:

	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

In [6]:
```
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 [7]:
```
df.columns
```
Out[7]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
```

In [8]:
```
df.head()
```
Out[8]:

	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 [9]:
```
df.tail()
```
Out[9]:

	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 [11]:
```
df.shape
```
Out[11]:
```
(1338, 7)
```

In [15]:
```
df.describe()
```

Out[15]:

	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 [16]:
```
df.duplicated().sum()
```

Out[16]: 1

To find Unique Values

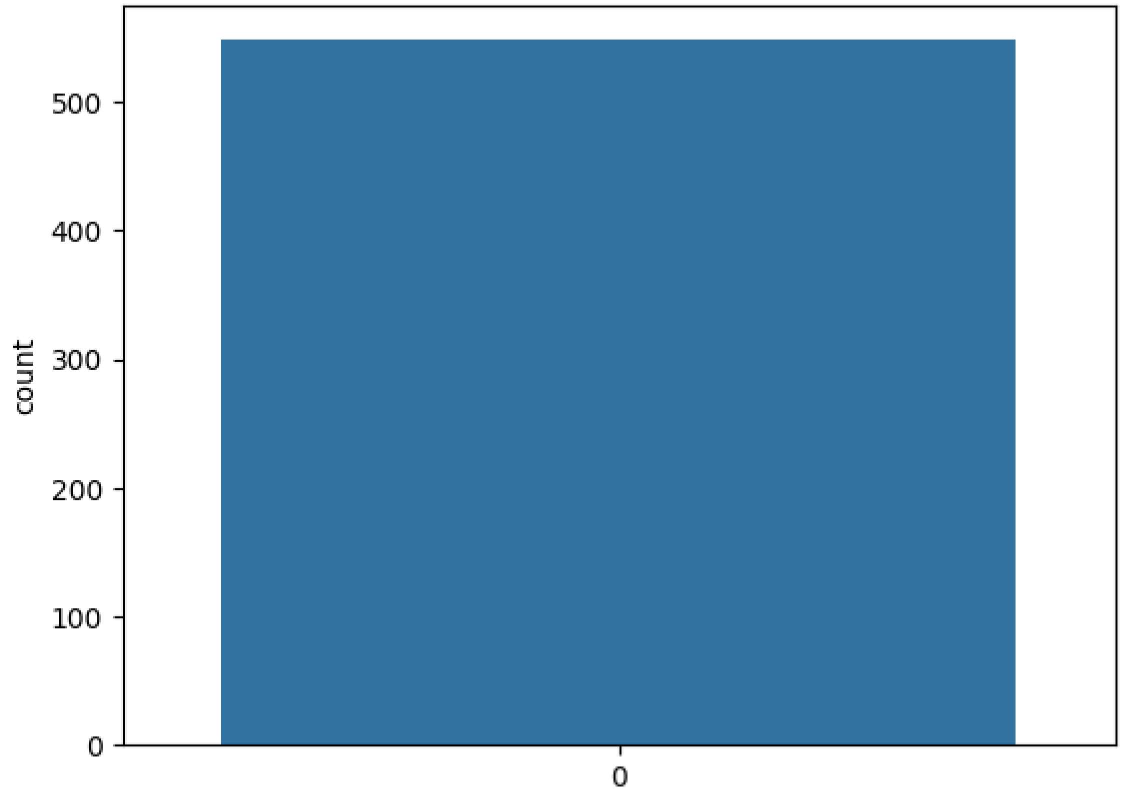
In [17]:
```
df['age'].unique()
df['children'].unique()
df['bmi'].unique()
```

```
Out[17]: array([27.9  , 33.77 , 33.   , 22.705, 28.88 , 25.74 , 33.44 , 27.74 ,
       29.83 , 25.84 , 26.22 , 26.29 , 34.4  , 39.82 , 42.13 , 24.6  ,
       30.78 , 23.845, 40.3  , 35.3  , 36.005, 32.4  , 34.1  , 31.92 ,
       28.025, 27.72 , 23.085, 32.775, 17.385, 36.3  , 35.6  , 26.315,
       28.6  , 28.31 , 36.4  , 20.425, 32.965, 20.8  , 36.67 , 39.9  ,
       26.6  , 36.63 , 21.78 , 30.8  , 37.05 , 37.3  , 38.665, 34.77 ,
       24.53 , 35.2  , 35.625, 33.63 , 28.   , 34.43 , 28.69 , 36.955,
       31.825, 31.68 , 22.88 , 37.335, 27.36 , 33.66 , 24.7  , 25.935,
       22.42 , 28.9  , 39.1  , 36.19 , 23.98 , 24.75 , 28.5  , 28.1  ,
       32.01 , 27.4  , 34.01 , 29.59 , 35.53 , 39.805, 26.885, 38.285,
       37.62 , 41.23 , 34.8  , 22.895, 31.16 , 27.2  , 26.98 , 39.49 ,
       24.795, 31.3  , 38.28 , 19.95 , 19.3  , 31.6  , 25.46 , 30.115,
       29.92 , 27.5  , 28.4  , 30.875, 27.94 , 35.09 , 29.7  , 35.72 ,
       32.205, 28.595, 49.06 , 27.17 , 23.37 , 37.1  , 23.75 , 28.975,
       31.35 , 33.915, 28.785, 28.3  , 37.4  , 17.765, 34.7  , 26.505,
       22.04 , 35.9  , 25.555, 28.05 , 25.175, 31.9  , 36.   , 32.49 ,
       25.3  , 29.735, 38.83 , 30.495, 37.73 , 37.43 , 24.13 , 37.145,
       39.52 , 24.42 , 27.83 , 36.85 , 39.6  , 29.8  , 29.64 , 28.215,
       37.   , 33.155, 18.905, 41.47 , 30.3  , 15.96 , 33.345, 37.7  ,
       27.835, 29.2  , 26.41 , 30.69 , 41.895, 30.9  , 32.2  , 32.11 ,
       31.57 , 26.2  , 30.59 , 32.8  , 18.05 , 39.33 , 32.23 , 24.035,
       36.08 , 22.3  , 26.4  , 31.8  , 26.73 , 23.1  , 23.21 , 33.7  ,
       33.25 , 24.64 , 33.88 , 38.06 , 41.91 , 31.635, 36.195, 17.8  ,
       24.51 , 22.22 , 38.39 , 29.07 , 22.135, 26.8  , 30.02 , 35.86 ,
       20.9  , 17.29 , 34.21 , 25.365, 40.15 , 24.415, 25.2  , 26.84 ,
       24.32 , 42.35 , 19.8  , 32.395, 30.2  , 29.37 , 34.2  , 27.455,
       27.55 , 20.615, 24.3  , 31.79 , 21.56 , 28.12 , 40.565, 27.645,
       31.2  , 26.62 , 48.07 , 36.765, 33.4  , 45.54 , 28.82 , 22.99 ,
       27.7  , 25.41 , 34.39 , 22.61 , 37.51 , 38.   , 33.33 , 34.865,
       33.06 , 35.97 , 31.4  , 25.27 , 40.945, 34.105, 36.48 , 33.8  ,
       36.7  , 36.385, 34.5  , 32.3  , 27.6  , 29.26 , 35.75 , 23.18 ,
       25.6  , 35.245, 43.89 , 20.79 , 30.5  , 21.7  , 21.89 , 24.985,
       32.015, 30.4  , 21.09 , 22.23 , 32.9  , 24.89 , 31.46 , 17.955,
       30.685, 43.34 , 39.05 , 30.21 , 31.445, 19.855, 31.02 , 38.17 ,
       20.6  , 47.52 , 20.4  , 38.38 , 24.31 , 23.6  , 21.12 , 30.03 ,
       17.48 , 20.235, 17.195, 23.9  , 35.15 , 35.64 , 22.6  , 39.16 ,
       27.265, 29.165, 16.815, 33.1  , 26.9  , 33.11 , 31.73 , 46.75 ,
       29.45 , 32.68 , 33.5  , 43.01 , 36.52 , 26.695, 25.65 , 29.6  ,
       38.6  , 23.4  , 46.53 , 30.14 , 30.   , 38.095, 28.38 , 28.7  ,
       33.82 , 24.09 , 32.67 , 25.1  , 32.56 , 41.325, 39.5  , 34.3  ,
       31.065, 21.47 , 25.08 , 43.4  , 25.7  , 27.93 , 39.2  , 26.03 ,
       30.25 , 28.93 , 35.7  , 35.31 , 31.   , 44.22 , 26.07 , 25.8  ,
       39.425, 40.48 , 38.9  , 47.41 , 35.435, 46.7  , 46.2  , 21.4  ,
```

```
23.8  , 44.77 , 32.12 , 29.1  , 37.29 , 43.12 , 36.86 , 34.295,
23.465, 45.43 , 23.65 , 20.7  , 28.27 , 35.91 , 29.   , 19.57 ,
31.13 , 21.85 , 40.26 , 33.725, 29.48 , 32.6  , 37.525, 23.655,
37.8  , 19.   , 21.3  , 33.535, 42.46 , 38.95 , 36.1  , 29.3  ,
39.7  , 38.19 , 42.4  , 34.96 , 42.68 , 31.54 , 29.81 , 21.375,
40.81 , 17.4  , 20.3  , 18.5  , 26.125, 41.69 , 24.1  , 36.2  ,
40.185, 39.27 , 34.87 , 44.745, 29.545, 23.54 , 40.47 , 40.66 ,
36.6  , 35.4  , 27.075, 28.405, 21.755, 40.28 , 30.1  , 32.1  ,
23.7  , 35.5  , 29.15 , 27.   , 37.905, 22.77 , 22.8  , 34.58 ,
27.1  , 19.475, 26.7  , 34.32 , 24.4  , 41.14 , 22.515, 41.8  ,
26.18 , 42.24 , 26.51 , 35.815, 41.42 , 36.575, 42.94 , 21.01 ,
24.225, 17.67 , 31.5  , 31.1  , 32.78 , 32.45 , 50.38 , 47.6  ,
25.4  , 29.9  , 43.7  , 24.86 , 28.8  , 29.5  , 29.04 , 38.94 ,
44.   , 20.045, 40.92 , 35.1  , 29.355, 32.585, 32.34 , 39.8  ,
24.605, 33.99 , 28.2  , 25.   , 33.2  , 23.2  , 20.1  , 32.5  ,
37.18 , 46.09 , 39.93 , 35.8  , 31.255, 18.335, 42.9  , 26.79 ,
39.615, 25.9  , 25.745, 28.16 , 23.56 , 40.5  , 35.42 , 39.995,
34.675, 20.52 , 23.275, 36.29 , 32.7  , 19.19 , 20.13 , 23.32 ,
45.32 , 34.6  , 18.715, 21.565, 23.   , 37.07 , 52.58 , 42.655,
21.66 , 32.   , 18.3  , 47.74 , 22.1  , 19.095, 31.24 , 29.925,
20.35 , 25.85 , 42.75 , 18.6  , 23.87 , 45.9  , 21.5  , 30.305,
44.88 , 41.1  , 40.37 , 28.49 , 33.55 , 40.375, 27.28 , 17.86 ,
33.3  , 39.14 , 21.945, 24.97 , 23.94 , 34.485, 21.8  , 23.3  ,
36.96 , 21.28 , 29.4  , 27.3  , 37.9  , 37.715, 23.76 , 25.52 ,
27.61 , 27.06 , 39.4  , 34.9  , 22.   , 30.36 , 27.8  , 53.13 ,
39.71 , 32.87 , 44.7  , 30.97 ])
```

In [18]:
```
sns.countplot(df['bmi'].unique())
```

Out[18]: <Axes: ylabel='count'>

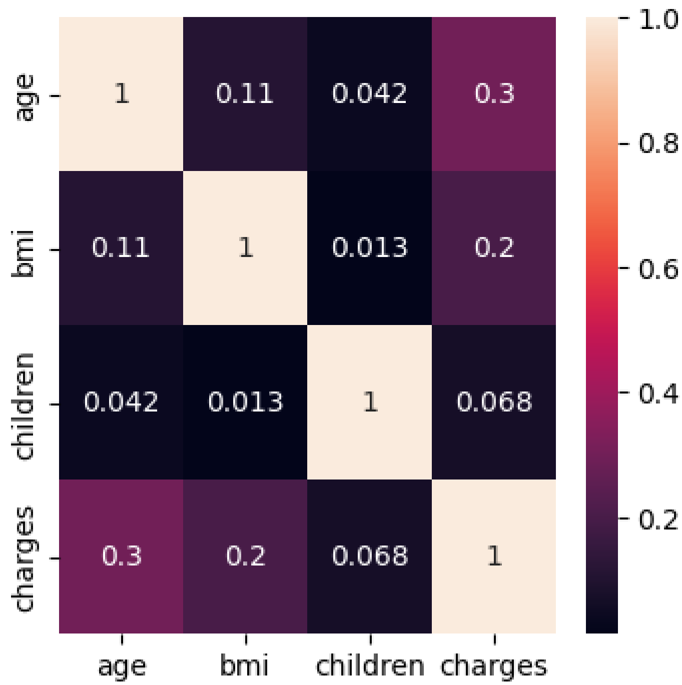


In [19]:
```
df.isnull().sum()
```

Out[19]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [20]:
```
Insuranced=df[['age','bmi','children','charges']]
plt.figure(figsize=(4,4))
sns.heatmap(Insuranced.corr(),annot=True)
```

Out[20]: <Axes: >



In [21]:
```
df.replace(np.nan,'0',inplace=True)
```

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

In [22]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['charges']).reshape(-1,1)
```

In [23]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.09167793902260402
```

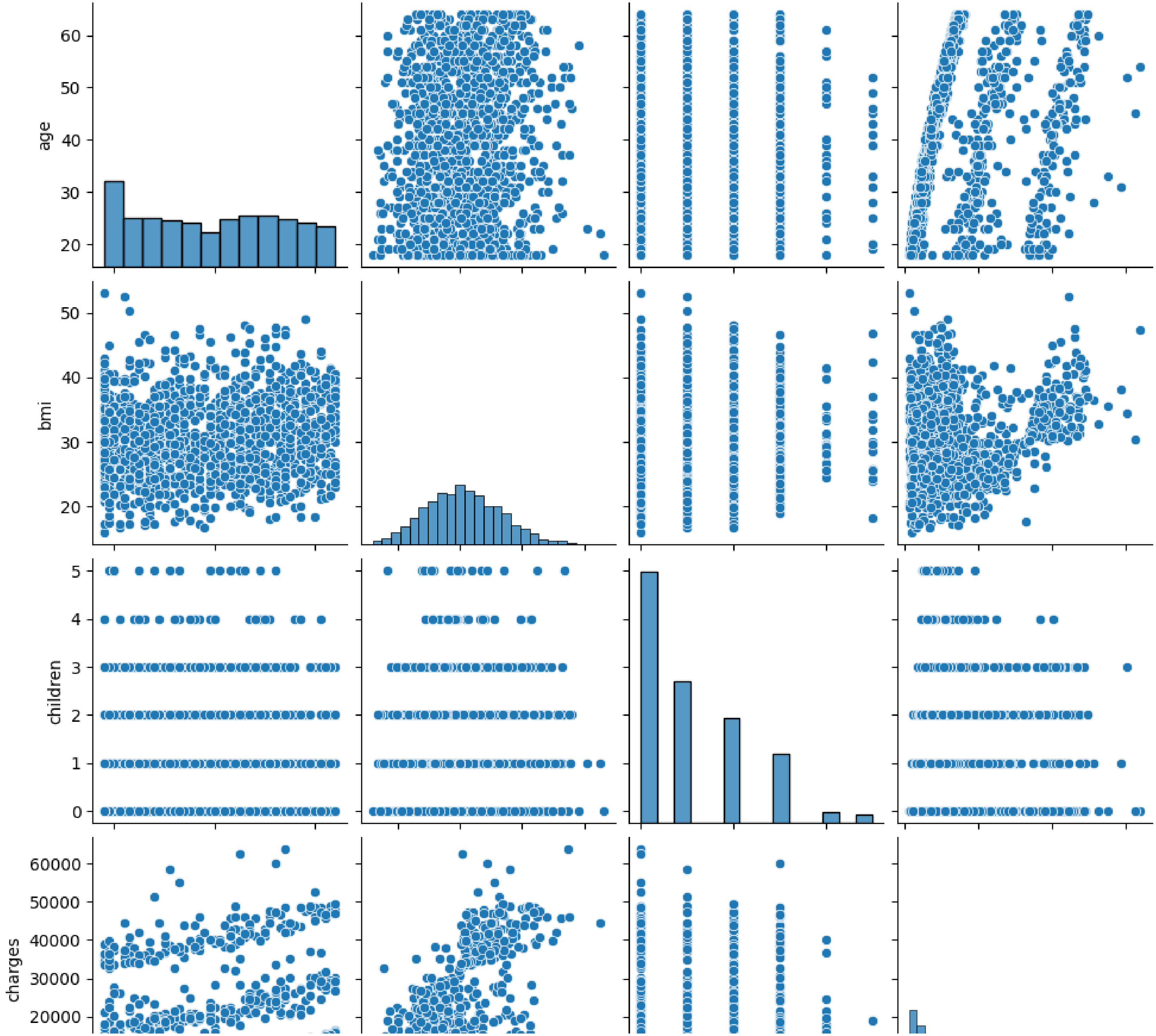
In the Linear Regression is not suitable for this model because of accuracy is very less

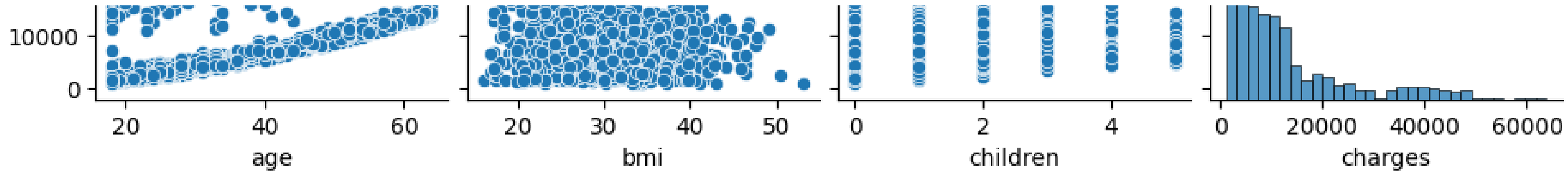
Logistisc Regression

In [24]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [25]:
```
sns.pairplot(df)
```

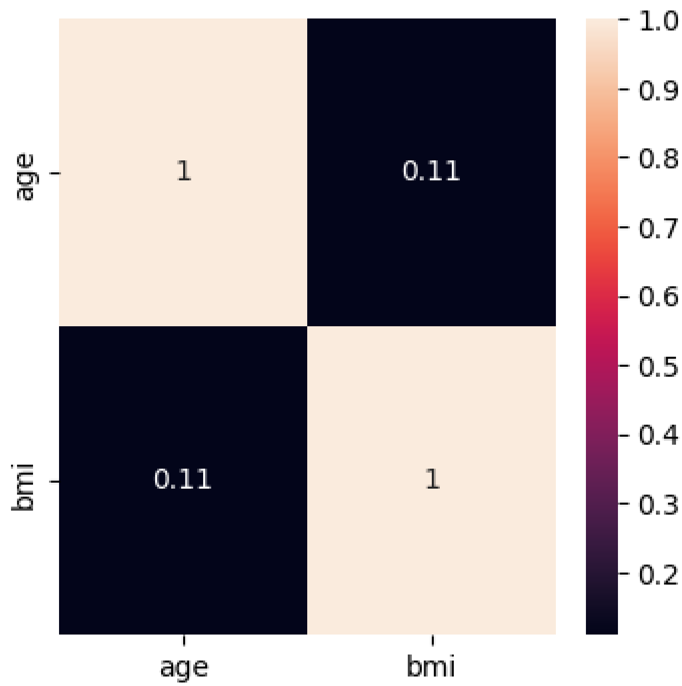
Out[25]:
```
<seaborn.axisgrid.PairGrid at 0x215a49d2bd0>
```


age
bmi
children
charges
60
50
40
30
20
5
4
3
2
1
0
60000
50000
40000
30000
20000



In [26]:
```
Insuranced=df[['age','bmi']]
plt.figure(figsize=(4,4))
sns.heatmap(Insuranced.corr(),annot=True)
```

Out[26]: <Axes: >



In [27]:
```
x = df.iloc[:,:-1].values
y = df.iloc[:,1].values
```

Split the train and test dataset

In [29]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2)
```

In [30]:
```
ml = LogisticRegression()
```

In [31]:
```
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=0.25,random_state=1)
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(max_iter=10000)
```

In [32]:
```
lr.fit(x_train,y_train)
```

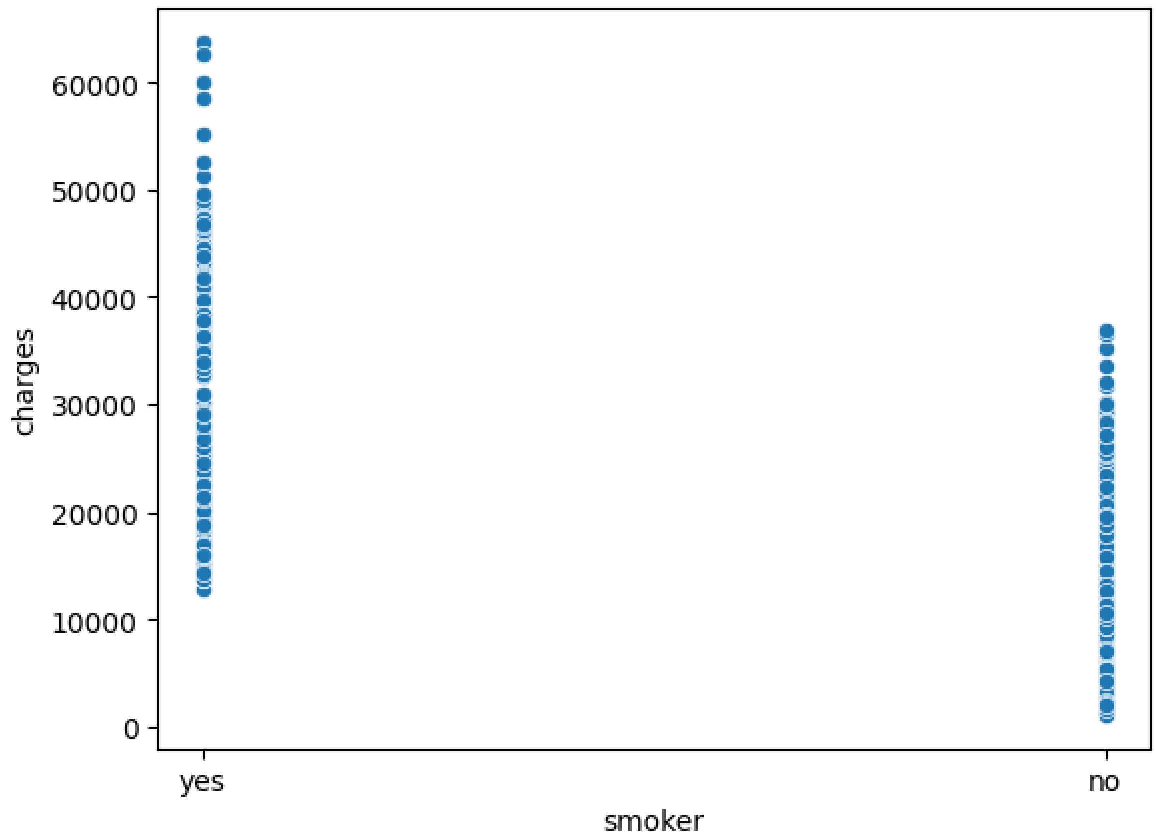
Out[32]:
```
▾ LogisticRegression
LogisticRegression(max_iter=10000)
```

In [33]:
```
score=lr.score(x_test,y_test)
print(score)
```

```
0.48059701492537316
```

In [34]:
```
sns.scatterplot(data=df,x='smoker',y='charges')
```

Out[34]: <Axes: xlabel='smoker', ylabel='charges'>



Decision Tree

In [35]:
```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()
clf.fit(x_train,y_train)
```

Out[35]:
```
▾ DecisionTreeClassifier

DecisionTreeClassifier()
```

In [36]:
```
score=clf.score(x_test,y_test)
print(score)
```

```
0.36716417910447763
```

random forest

In [37]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[37]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [38]:
```
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 [39]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
```

In [40]:
```
grid_search.fit(x_train,y_train)
```

Out[40]:

▸ **GridSearchCV**

▸ **estimator: RandomForestClassifier**

▸ RandomForestClassifier

In [41]:
```
grid_search.best_score_
```

Out[41]:
```
0.5134591375018887
```

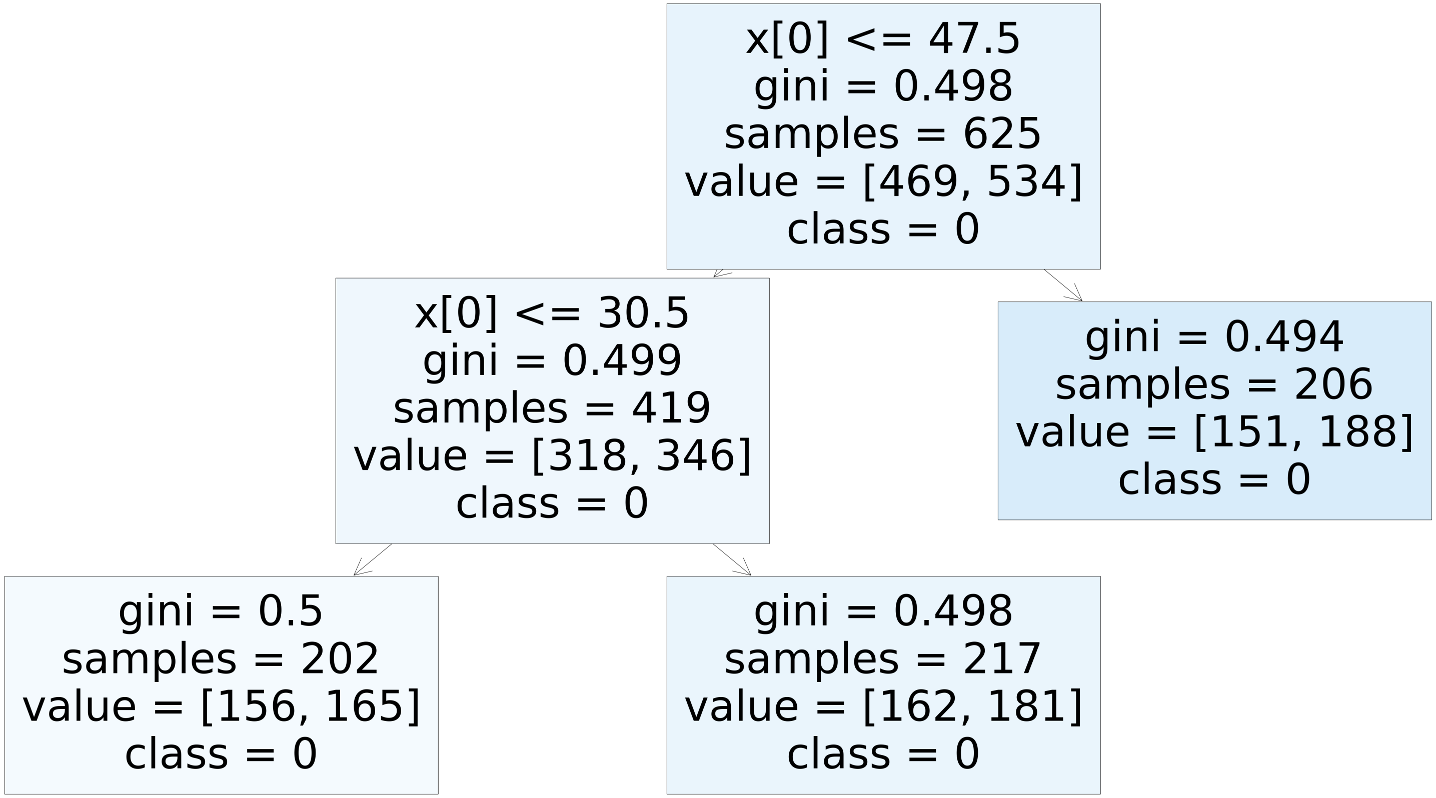
In [42]:
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[42]:

▾ RandomForestClassifier

```
RandomForestClassifier(max_depth=2, min_samples_leaf=200, n_estimators=10)
```

In [43]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['1','0'],filled=True);
```



In [44]:
```
score=rfc.score(x_test,y_test)
print(score)
```

```
0.3701492537313433
```

In [45]:
```
convert={"sex":{'male':1,'female':2}}
df=df.replace(convert)
df
```

Out[45]:

	age	sex	bmi	children	smoker	region	charges
0	19	2	27.900	0	yes	southwest	16884.92400
1	18	1	33.770	1	no	southeast	1725.55230
2	28	1	33.000	3	no	southeast	4449.46200
3	33	1	22.705	0	no	northwest	21984.47061
4	32	1	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	no	northwest	10600.54830
1334	18	2	31.920	0	no	northeast	2205.98080
1335	18	2	36.850	0	no	southeast	1629.83350
1336	21	2	25.800	0	no	southwest	2007.94500
1337	61	2	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

In [46]:
```
from sklearn.metrics import r2_score
```

In [47]:
```
import pickle
```

In [48]:
```
filename="Prediction"
pickle.dump(rfc,open(filename,'wb'))
```

In [ ]:
```

```