problem statement :which model is suitable best for insurance dataset

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

data collection

Read the data

In [2]:
```
df=pd.read_csv(r"C:\Users\rakesh\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

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

to find null values

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
```

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 [72]:
```
df.duplicated().sum()
```

Out[72]:
```
1
```

To find Unique Values

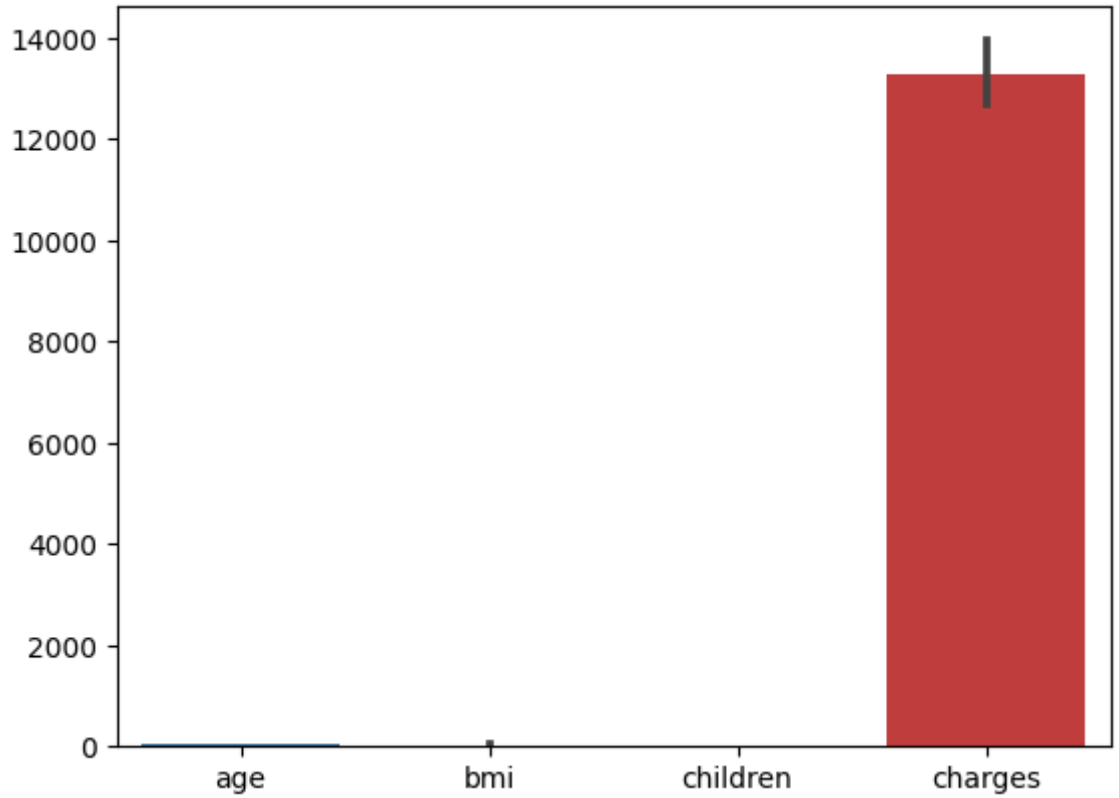
In [73]:
```
df['age'].unique()
df['children'].unique()
df['bmi'].unique()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\indexes\base.py:3652, in Index.get_loc
(self, key)
   3651 try:
-> 3652     return self._engine.get_loc(casted_key)
   3653 except KeyError as err:

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\_libs\index.pyx:147, in pandas._libs.index.I
ndexEngine.get_loc()

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\_libs\index.pyx:176, in pandas._libs.index.I
ndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'bni'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[73], line 3
      1 df['age'].unique()
      2 df['children'].unique()
----> 3 df['bni'].unique()

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\frame.py:3761, in DataFrame.__getitem__
(self, key)
   3759 if self.columns.nlevels > 1:
   3760     return self._getitem_multilevel(key)
-> 3761 indexer = self.columns.get_loc(key)
   3762 if is_integer(indexer):
   3763     indexer = [indexer]

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\indexes\base.py:3654, in Index.get_loc
(self, key)
   3652     return self._engine.get_loc(casted_key)
   3653 except KeyError as err:
-> 3654     raise KeyError(key) from err
```

```
3655 except TypeError:
3656     # If we have a listlike key, _check_indexing_error will raise
3657     #  InvalidIndexError. Otherwise we fall through and re-raise
3658     #  the TypeError.
3659     self._check_indexing_error(key)

KeyError: 'bni'
```

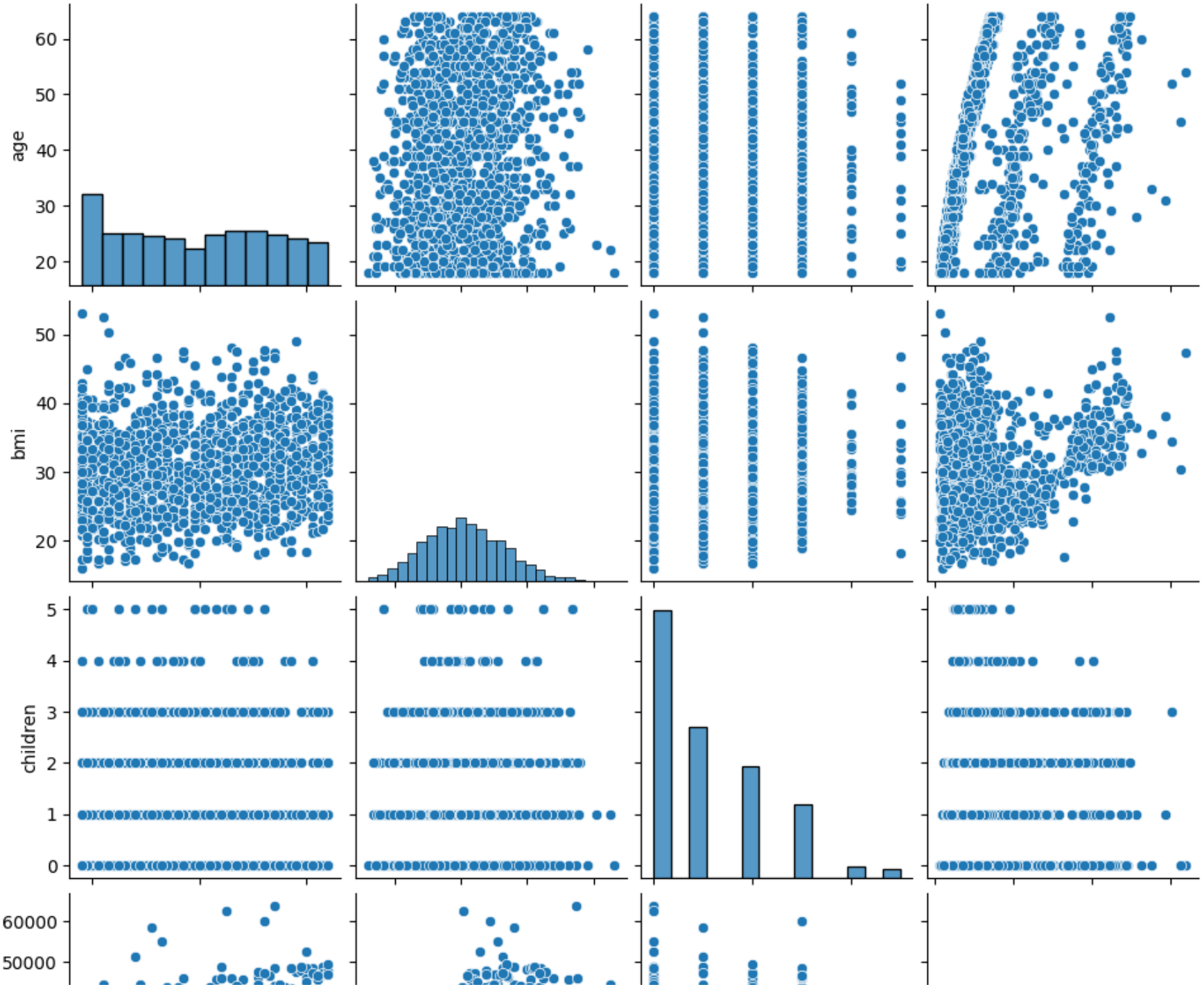
In [9]:
```
sns.barplot(df)
```

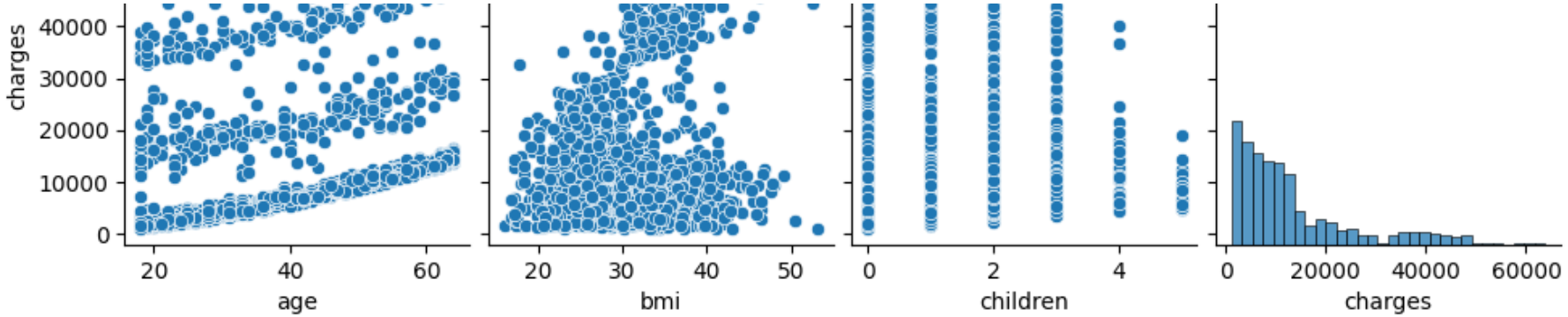
Out[9]: <Axes: >



In [10]:
```
sns.pairplot(df)
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x1547f574850>
```


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



In [11]:
```
df.columns
```

Out[11]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
```

In [12]:
```
t={"smoker":{"yes":1,"no":0}}
df=df.replace(t)
df
```

Out[12]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	1	southwest	16884.92400
1	18	male	33.770	1	0	southeast	1725.55230
2	28	male	33.000	3	0	southeast	4449.46200
3	33	male	22.705	0	0	northwest	21984.47061
4	32	male	28.880	0	0	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	0	northwest	10600.54830
1334	18	female	31.920	0	0	northeast	2205.98080
1335	18	female	36.850	0	0	southeast	1629.83350
1336	21	female	25.800	0	0	southwest	2007.94500
1337	61	female	29.070	0	1	northwest	29141.36030

1338 rows × 7 columns

In [13]:
```
h={"sex":{"male":1,"female":0}}
df=df.replace(h)
df
```

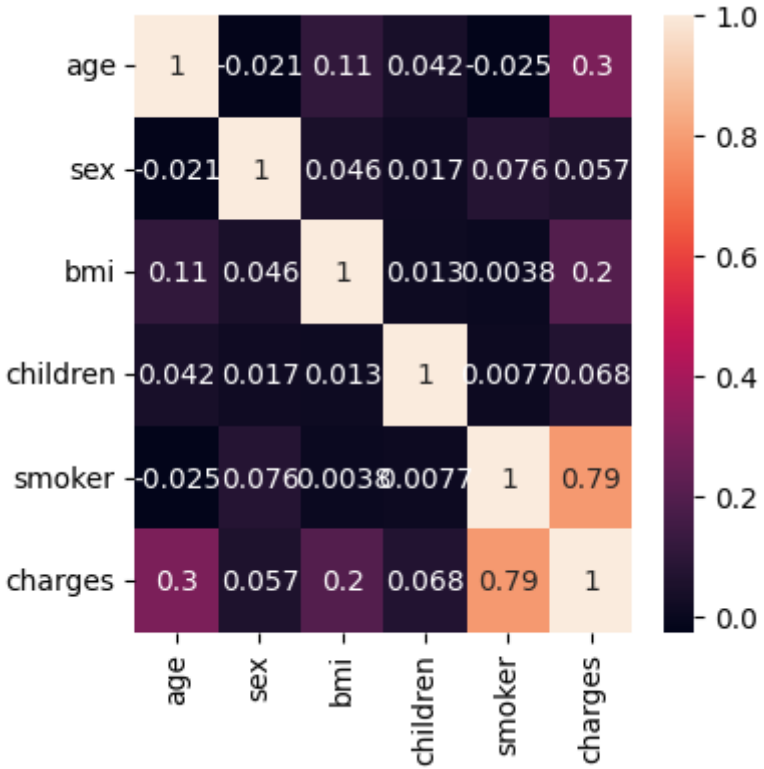
Out[13]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	southwest	16884.92400
1	18	1	33.770	1	0	southeast	1725.55230
2	28	1	33.000	3	0	southeast	4449.46200
3	33	1	22.705	0	0	northwest	21984.47061
4	32	1	28.880	0	0	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	0	northwest	10600.54830
1334	18	0	31.920	0	0	northeast	2205.98080
1335	18	0	36.850	0	0	southeast	1629.83350
1336	21	0	25.800	0	0	southwest	2007.94500
1337	61	0	29.070	0	1	northwest	29141.36030

1338 rows × 7 columns

In [14]:
```
ho=df[['age', 'sex', 'bmi', 'children', 'smoker', 'charges']]
plt.figure(figsize=(4,4))
sns.heatmap(ho.corr(),annot=True)
```

Out[14]: <Axes: >



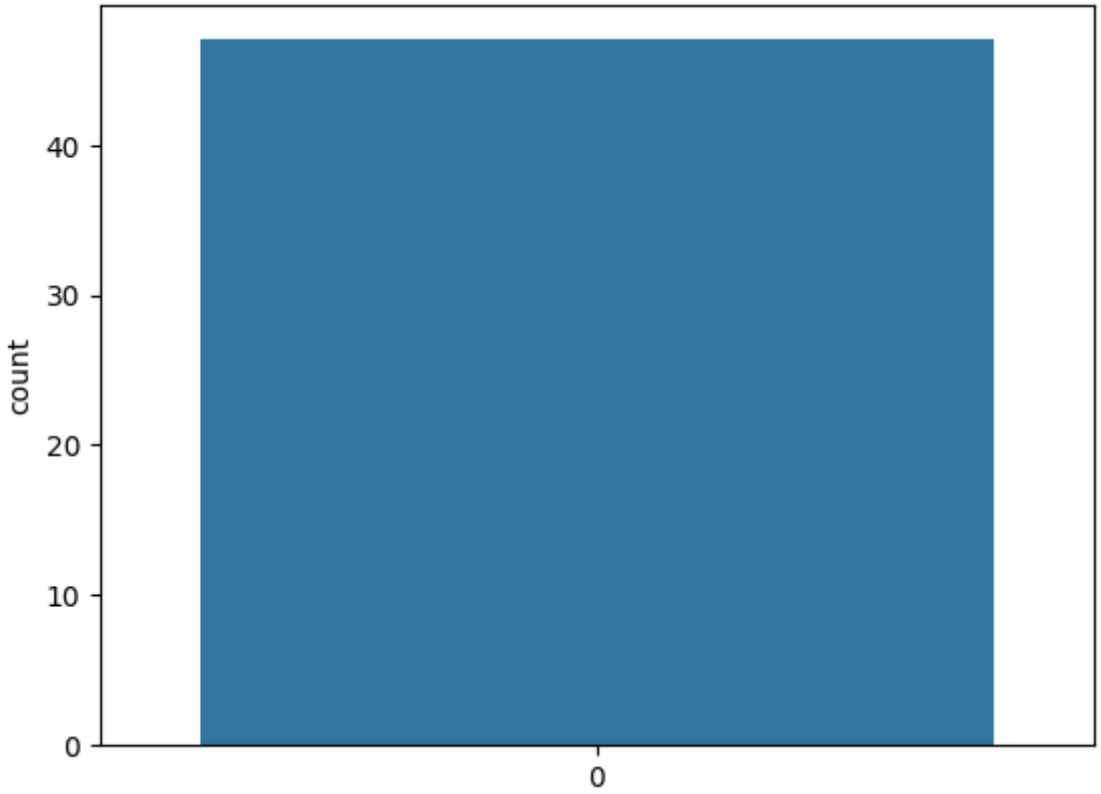
In [15]:
```
x=df[['age', 'sex', 'bmi', 'children', 'smoker']]
y=df['charges']
```

In [16]:
```
x=df[['age', 'sex', 'bmi', 'children', 'smoker']]
y=df['charges']
```

Data Visualize:Visualization the unique counts

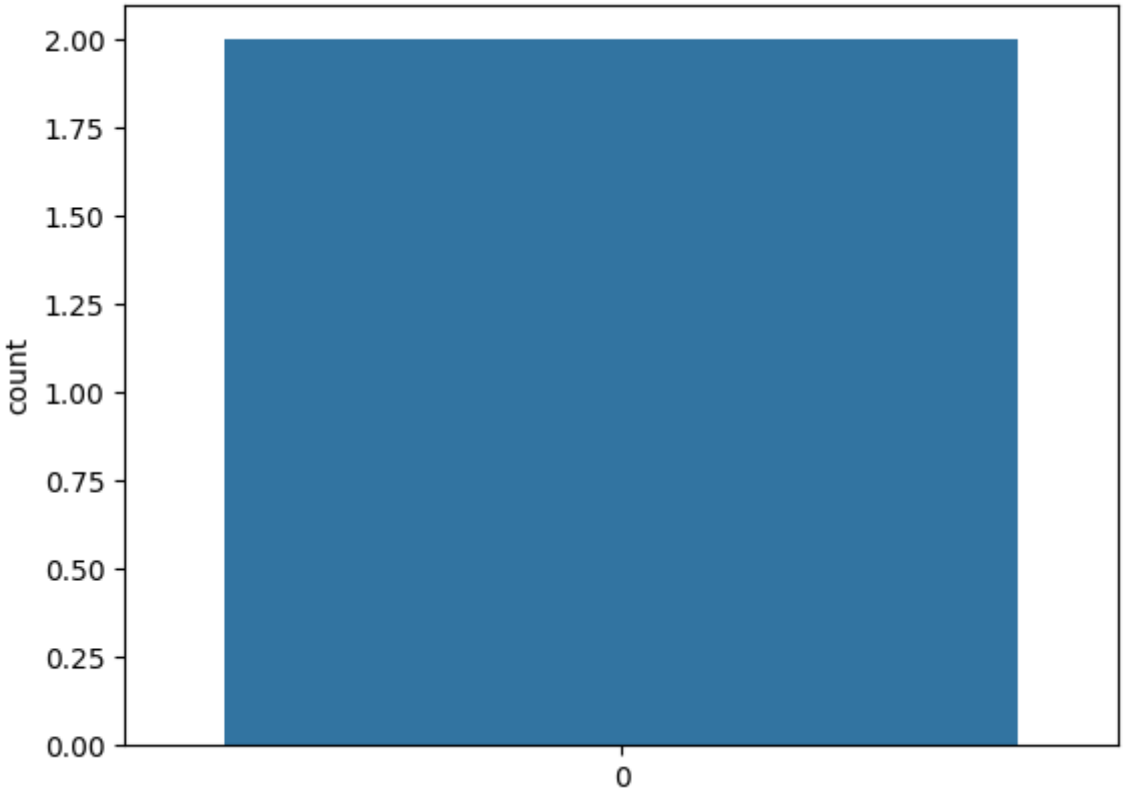
In [17]:
```
sns.countplot(df['age'].unique())
```

Out[17]:
```
<Axes: ylabel='count'>
```



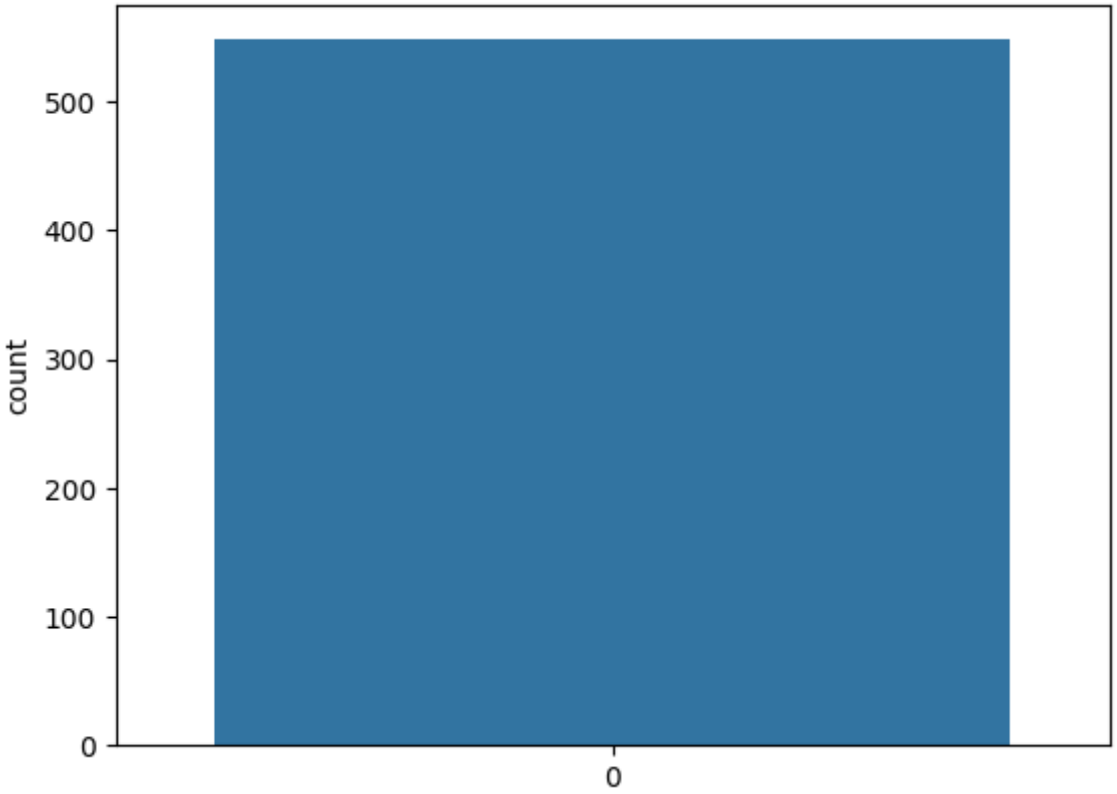
In [18]:
```
sns.countplot(df['sex'].unique())
```

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



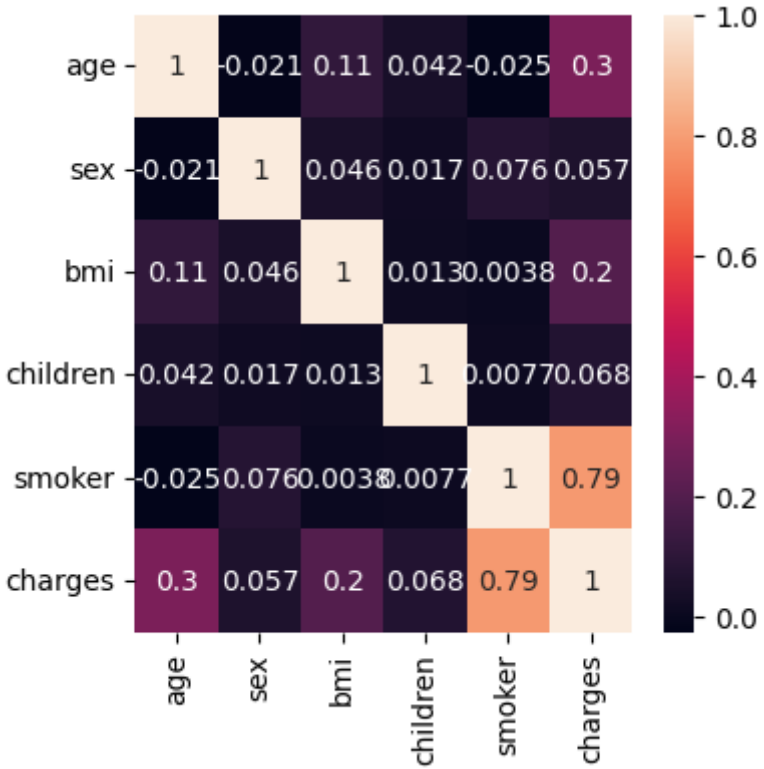
In [19]:
```
sns.countplot(df['bmi'].unique())
```

Out[19]:
```
<Axes: ylabel='count'>
```



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

Out[20]: <Axes: >



to find Mean and median

In [21]:
```
print(df["age"].mean(skipna=True))
print(df["age"].median(skipna=True))
```

```
39.20702541106129
39.0
```

In [22]:
```
print(df["children"].mean(skipna=True))
print(df["children"].median(skipna=True))
```

```
1.0949177877429
1.0
```

In [23]:
```
print(df["bmi"].mean(skipna=True))
print(df["bmi"].median(skipna=True))
```

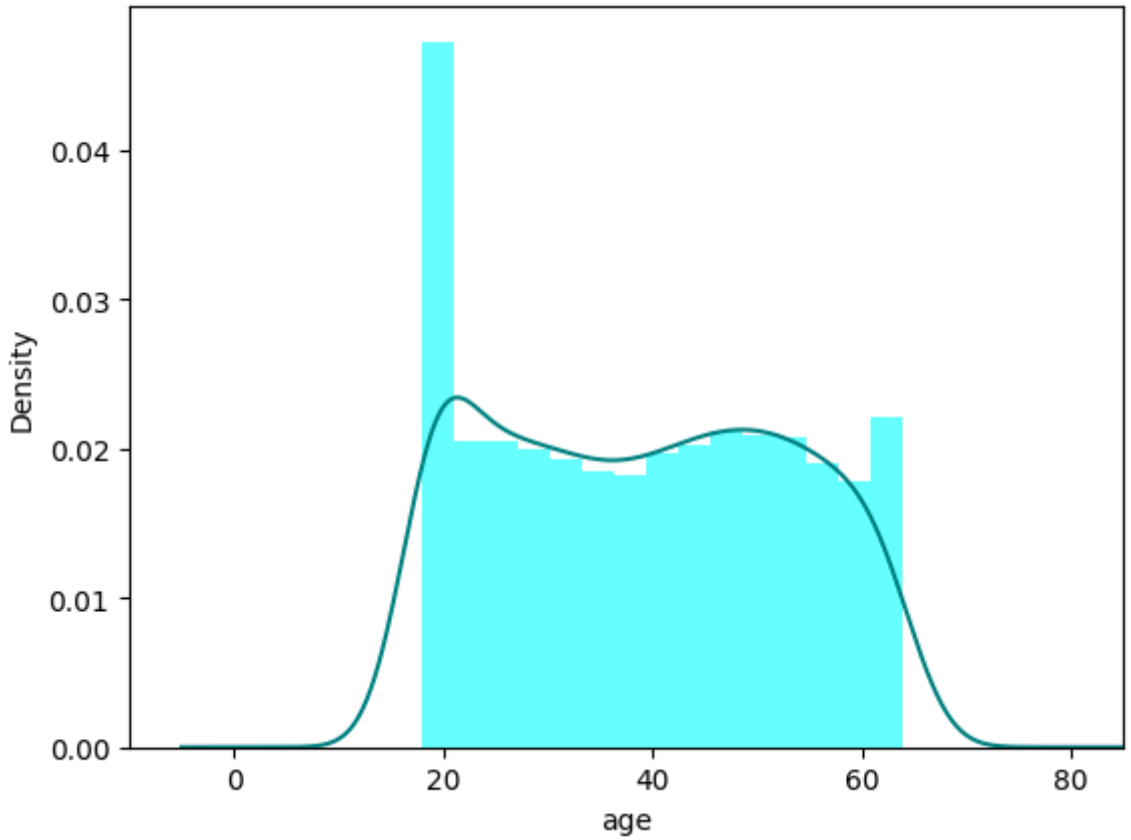
```
30.66339686098655
30.4
```

In [24]:
```
print(df["charges"].mean(skipna=True))
print(df["charges"].median(skipna=True))
```

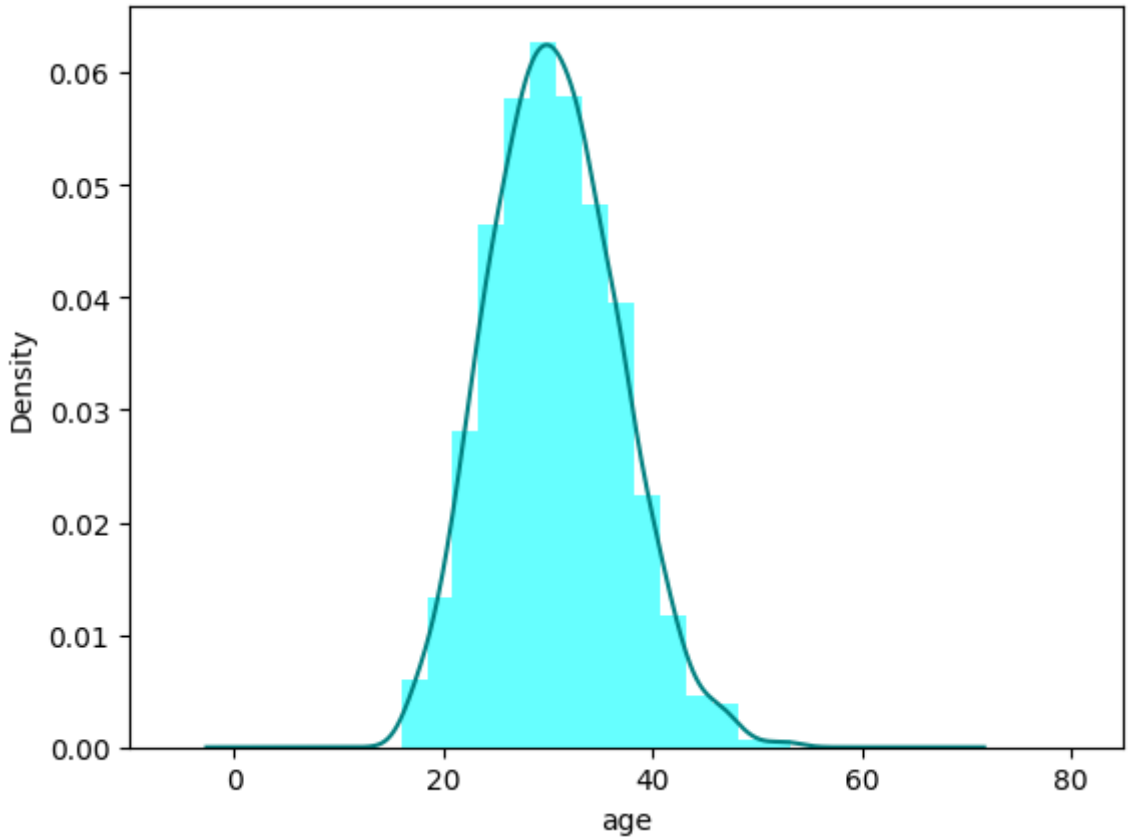
```
13270.422265141257
9382.033
```

data visualization in histogram

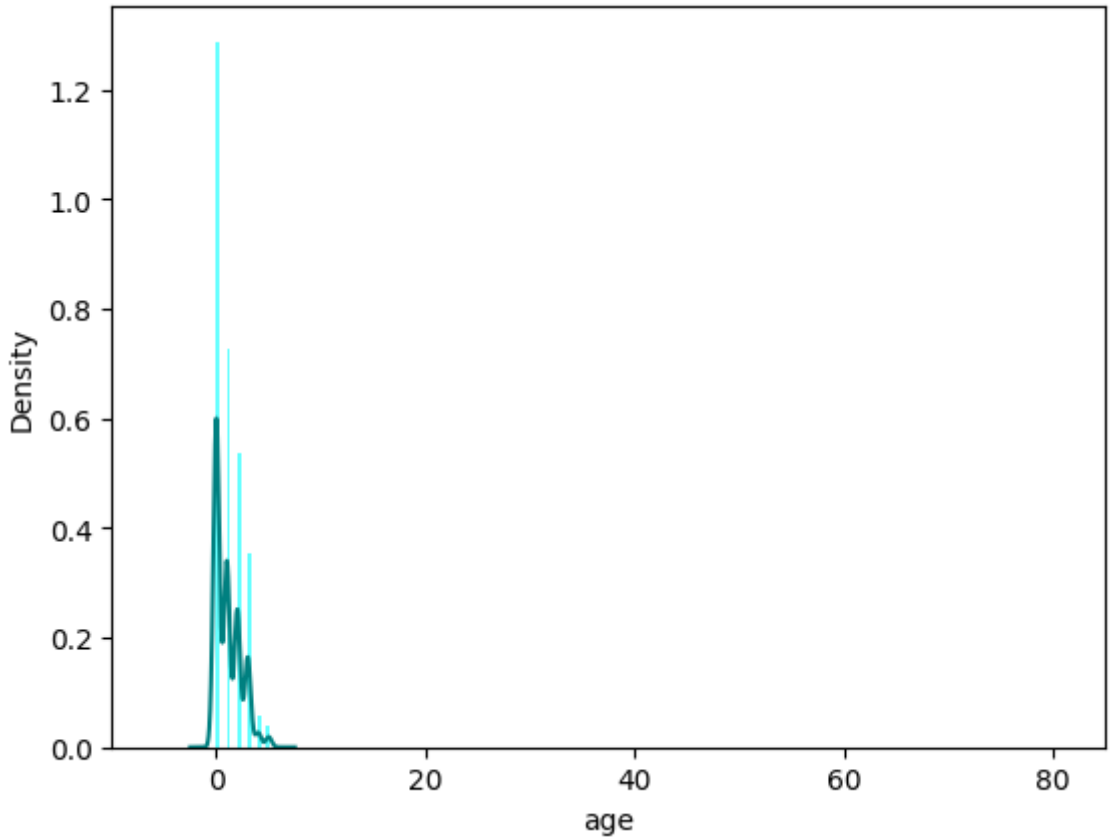
In [25]:
```
ax=df["age"].hist(bins=15,density=True,stacked=True,color='cyan',alpha=0.6)
df["age"].plot(kind='density',color='teal')
ax.set(xlabel='age')
plt.xlim(-10,85)
plt.show()
```



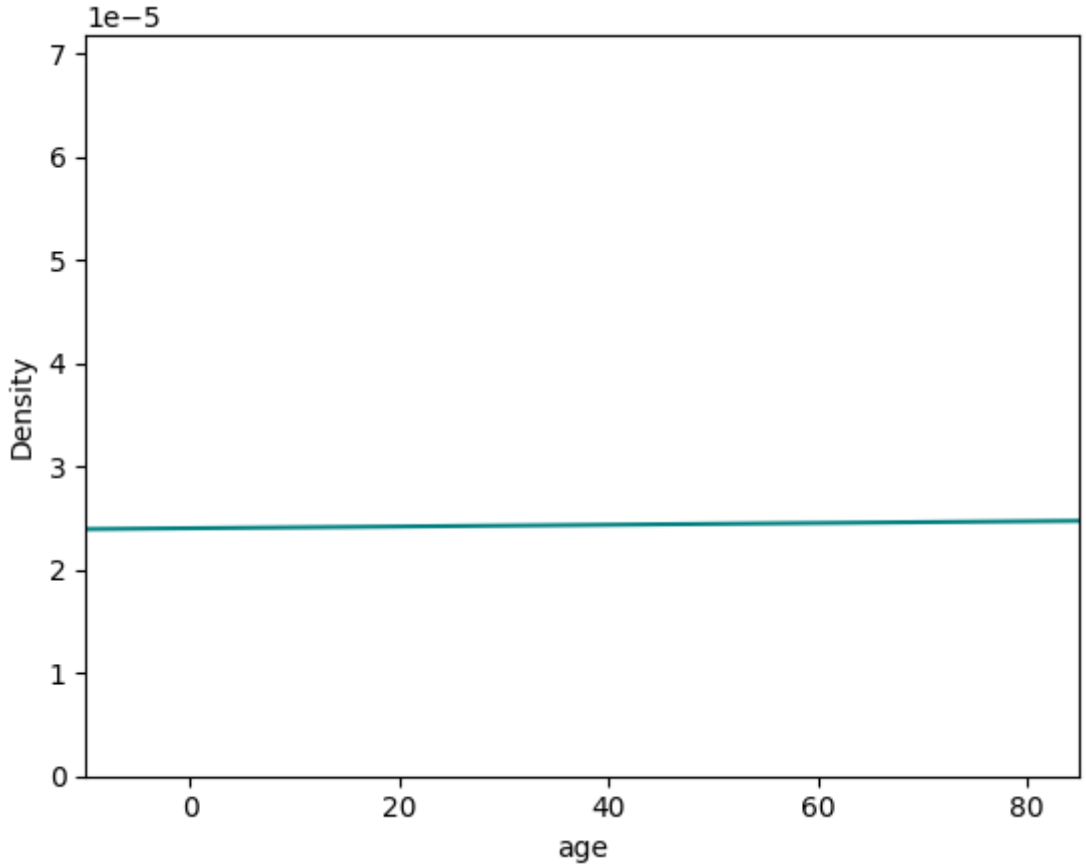
In [26]:
```
ax=df["bmi"].hist(bins=15,density=True,stacked=True,color='cyan',alpha=0.6)
df["bmi"].plot(kind='density',color='teal')
ax.set(xlabel='age')
plt.xlim(-10,85)
plt.show()
```



In [27]:
```
ax=df["children"].hist(bins=15,density=True,stacked=True,color='cyan',alpha=0.6)
df["children"].plot(kind='density',color='teal')
ax.set(xlabel='age')
plt.xlim(-10,85)
plt.show()
```



In [28]:
```
ax=df["charges"].hist(bins=15,density=True,stacked=True,color='cyan',alpha=0.6)
df["charges"].plot(kind='density',color='teal')
ax.set(xlabel='age')
plt.xlim(-10,85)
plt.show()
```



To Check The Null Values

In [29]:
```
df.replace(np.nan,'0',inplace=True)
```

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

In [30]:
```
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)
```

In [74]:
```
from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(X_train,y_train)
print(regr.score(x_test,y_test))
```

```
-4082835.427787215
```

In [32]:
```
score=regr.score(X_test,y_test)
print(score)
```

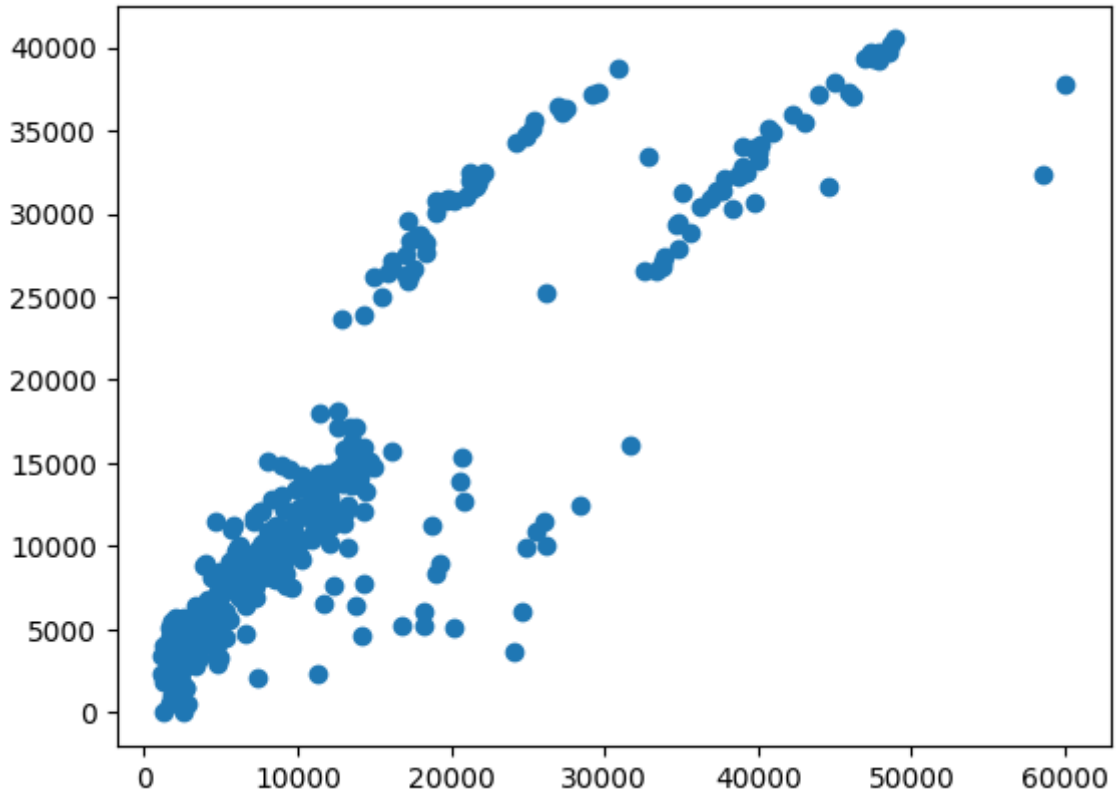
```
0.780095696440481
```

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

In [33]:
```
predictions=regr.predict(X_test)
```

In [34]:
```
plt.scatter(y_test,predictions)
```

Out[34]:
```
<matplotlib.collections.PathCollection at 0x15407232ad0>
```

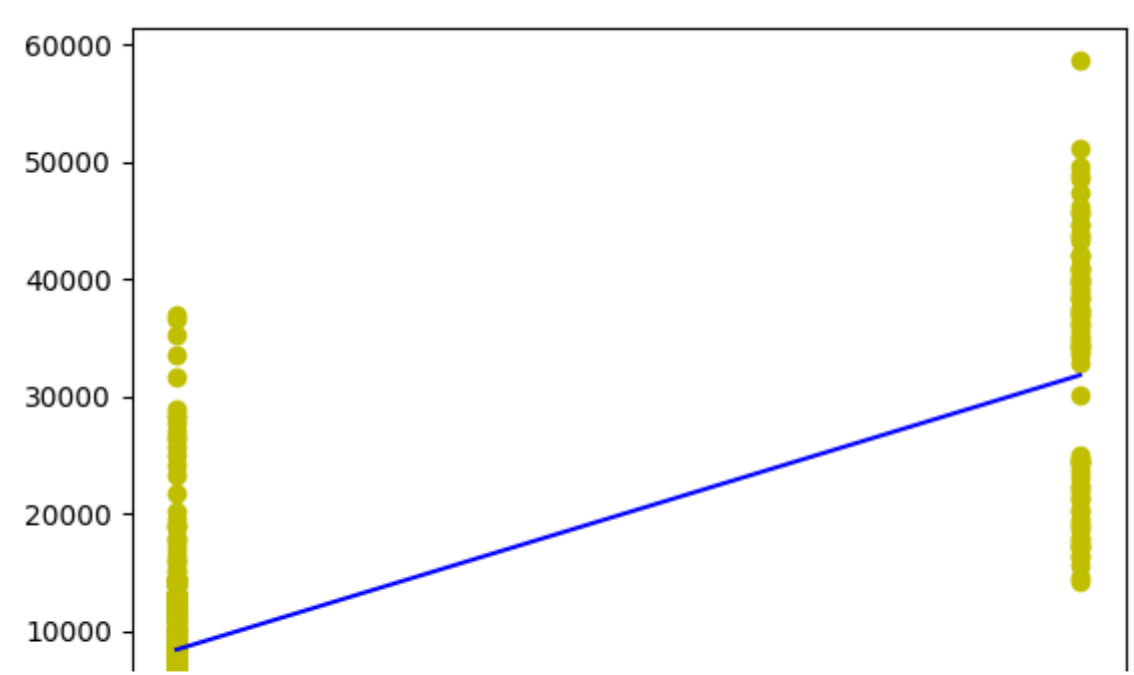


In [35]:
```
x=np.array(df['smoker']).reshape(-1,1)
y=np.array(df['charges']).reshape(-1,1)
df.dropna(inplace=True)
```

In [36]:
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[36]:
```
LinearRegression
LinearRegression()
```

In [37]:
```
y_pred=regr.predict(X_test)
plt.scatter(X_test,y_test,color='y')
plt.plot(X_test,y_pred,color='b')
plt.show()
```



Logistic Regresssion

In [75]:
```
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 [76]:
```
lr.fit(x_train,y_train)
```

```
C:\Users\SASIDHAR ROYAL\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[76]:
```
LogisticRegression
LogisticRegression(max_iter=10000)
```

In [40]:
```
score=lr.score(x_test,y_test)
print(score)
```

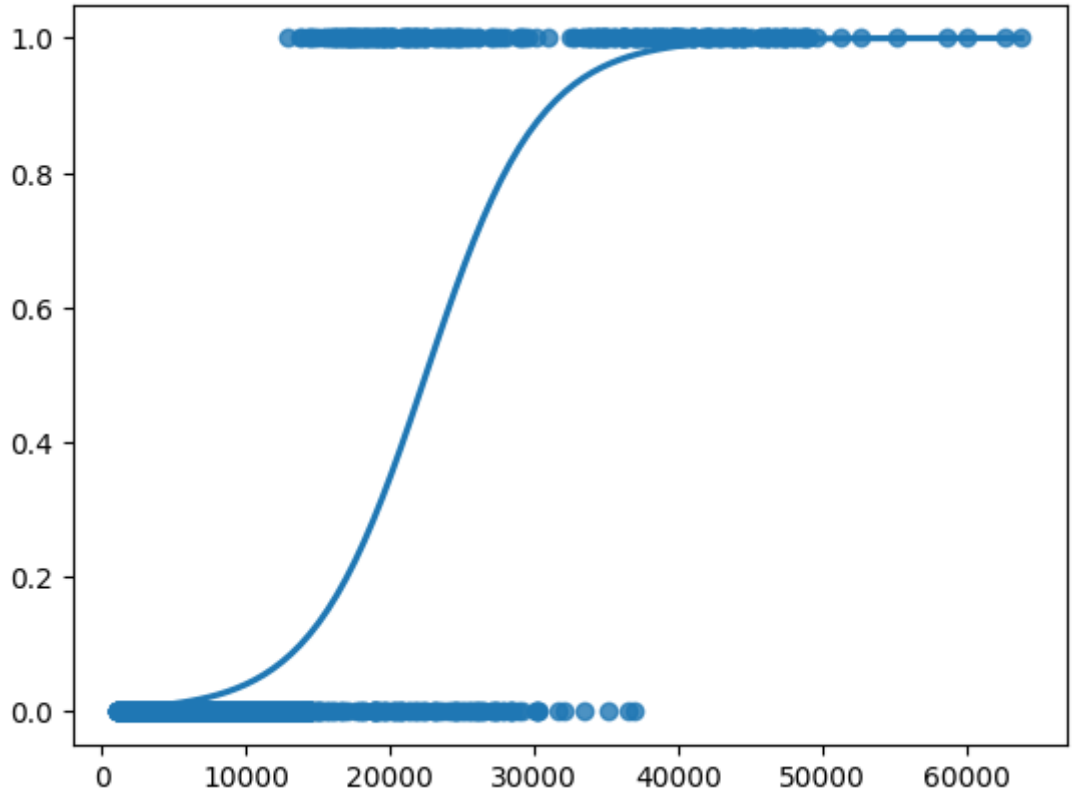
```
0.8930348258706468
```

In [41]:
```
pip install statsmodels
```

```
Requirement already satisfied: statsmodels in c:\users\sasidhar royal\appdata\local\programs\python\python311\lib\si
te-packages (0.14.0)
Requirement already satisfied: numpy>=1.18 in c:\users\sasidhar royal\appdata\local\programs\python\python311\lib\si
te-packages (from statsmodels) (1.24.3)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in c:\users\sasidhar royal\appdata\local\programs\python\python311
\lib\site-packages (from statsmodels) (1.10.1)
Requirement already satisfied: pandas>=1.0 in c:\users\sasidhar royal\appdata\local\programs\python\python311\lib\si
te-packages (from statsmodels) (2.0.2)
Requirement already satisfied: patsy>=0.5.2 in c:\users\sasidhar royal\appdata\local\programs\python\python311\lib\s
ite-packages (from statsmodels) (0.5.3)
Requirement already satisfied: packaging>=21.3 in c:\users\sasidhar royal\appdata\local\programs\python\python311\li
b\site-packages (from statsmodels) (23.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\sasidhar royal\appdata\local\programs\python\pytho
n311\lib\site-packages (from pandas>=1.0->statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\sasidhar royal\appdata\local\programs\python\python311\lib\s
ite-packages (from pandas>=1.0->statsmodels) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in c:\users\sasidhar royal\appdata\local\programs\python\python311\lib
\site-packages (from pandas>=1.0->statsmodels) (2023.3)
Requirement already satisfied: six in c:\users\sasidhar royal\appdata\local\programs\python\python311\lib\site-packa
ges (from patsy>=0.5.2->statsmodels) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [42]:
```
sns.regplot(x=x,y=y,data=df,logistic=True,ci=None)
```

Out[42]: <Axes: >



Decision Tree

In [43]:
```
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)
```

```
0.8880597014925373
```

Random Forest

In [45]:
```
#Random forest classifier
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(X_train,y_train)
```

```
C:\Users\SASIDHAR ROYAL\AppData\Local\Temp\ipykernel_15444\2470359396.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 [46]:
```
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 [47]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
```

In [48]:
```
grid_search.fit(X_train,y_train)
```

```
C:\Users\SASIDHAR ROYAL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validat
ion.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\SASIDHAR ROYAL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validat
ion.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\SASIDHAR ROYAL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validat
ion.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\SASIDHAR ROYAL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validat
ion.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\SASIDHAR ROYAL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validat
ion.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples,), for example using ravel().
```

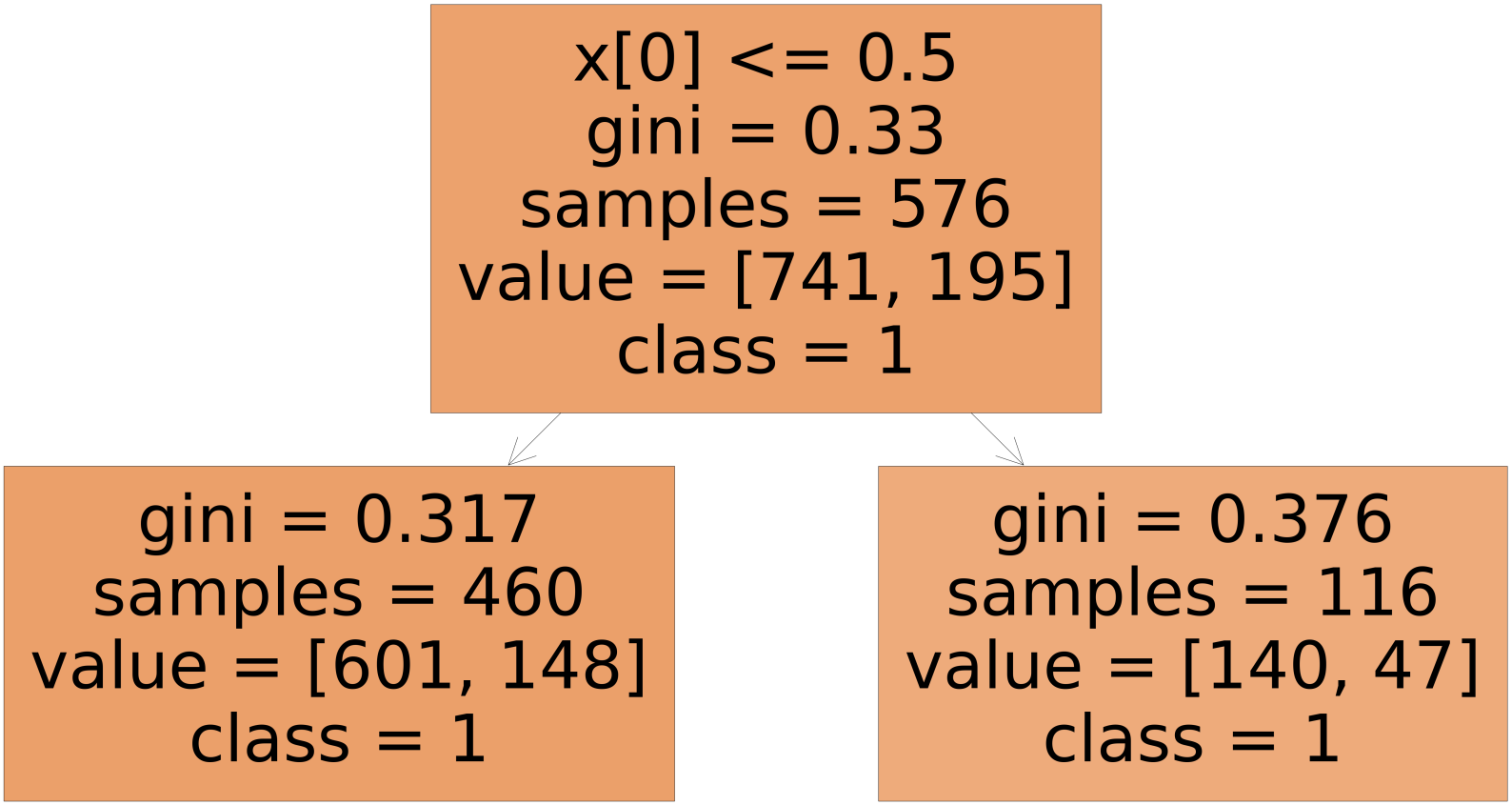
In [49]:
```
grid_search.best_score_
```

Out[49]:
```
0.7938034188034188
```

In [50]:
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[50]:
```
RandomForestClassifier
RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=10)
```

In [52]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['1','0'],filled=True);
```



In [53]:
```
score=rfc.score(x_test,y_test)
print(score)
```

```
0.7985074626865671
```

In [68]:
```
import pickle
linreg=LinearRegression()
linreg.fit(X_train,y_train)
```

Out[68]:
```
▾ LinearRegression

LinearRegression()
```

In [69]:
```
filename="prediction"
pickle.dump(linreg,open(filename,'wb'))
```

import pandas as pd import pickle filename="prediction" model=pickle.load(open(filename,'rb'))

In [77]:
```
1 rk=[[22.1],[11.5]]
2 predresult=model.predict(rk)
3 predresult
4
```

Out[77]:
```
array([[1.20237169],
       [0.72004474]])
```

Conclusion

for the above diiferent typees of models we get accuracy based on the accuracy we can predict the which model is better for this dataset for this dataset.Logistic regression and Decision tree getting more accuracy among all the models ,but logistic regression is more accuracy than Decision tree.so, the given data set is best fit for LogisticRegression

In [ ]: