CASE STUDY OF INSURANCE.CSV ¶

In [1]:

```
!pip install scipy
```

```
Requirement already satisfied: scipy in c:\users\ritesh\appdata\local\progra
ms\python\python39\lib\site-packages (1.6.3)

WARNING: You are using pip version 21.1.1; however, version 21.3.1 is availa
ble.
You should consider upgrading via the 'c:\users\ritesh\appdata\local\program
s\python\python39\python.exe -m pip install --upgrade pip' command.


Requirement already satisfied: numpy<1.23.0,>=1.16.5 in c:\users\ritesh\appd
ata\local\programs\python\python39\lib\site-packages (from scipy) (1.20.3)
```

1. Import neccessry libraries

In [2]:

```

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
```

2. Read the data as a data frame

In [3]:

```
data= pd.read_csv(r'D:Reports/insurance.csv')
df=pd.DataFrame(data)
print(df)
```

```
      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 x 7 columns]
```

Number 3

a. Shape of the data

In [4]:

```
df.shape
```

Out[4]:

```
(1338, 7)
```

b. Data type of each attribute

In [5]:

```
df.dtypes
```

Out[5]:

```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object
```

c. Checking the presence of missing values

In [6]:

```
for i in df.columns: print(data[i].isnull().sum())


```

```
0
0
0
0
0
0
0
```

other method for checking presence missing value

In [7]:

```
numeric = df.columns[df.dtypes !='object']
print(numeric)
```

```
Index(['age', 'bmi', 'children', 'charges'], dtype='object')
```

In [8]:

```
df[numeric].isnull()
```

Out[8]:

	age	bmi	children	charges
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
...	...	...	...	...
1333	False	False	False	False
1334	False	False	False	False
1335	False	False	False	False
1336	False	False	False	False
1337	False	False	False	False

1338 rows × 4 columns

In [9]:

```
character = df.columns[df.dtypes=='object']
print(character)
```

```
Index(['sex', 'smoker', 'region'], dtype='object')
```

In [10]:

```
df[character].isnull()
```

Out[10]:

	sex	smoker	region
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
...	...	...	...
1333	False	False	False
1334	False	False	False
1335	False	False	False
1336	False	False	False
1337	False	False	False

1338 rows × 3 columns

In [11]:

```
df[character].isnull().sum()
```

Out[11]:

```
sex       0
smoker    0
region    0
dtype: int64
```

In [12]:

```
df[numeric].isnull().sum()
```

Out[12]:

```
age         0
bmi         0
children    0
charges     0
dtype: int64
```

d.calculate a 5-number summary

In [13]:

```
df.describe()
```

Out[13]:

	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 [14]:

```
#for Age
print(df['age'].describe()[['min','25%','50%','75%','max']])

#for BMI
print(df['bmi'].describe()[['min','25%','50%','75%','max']])

#for Children
print(df['children'].describe()[['min','25%','50%','75%','max']])

#for Charges
print(df['charges'].describe()[['min','25%','50%','75%','max']])


```

```
min    18.0
25%    27.0
50%    39.0
75%    51.0
max    64.0
Name: age, dtype: float64
min    15.96000
25%    26.29625
50%    30.40000
75%    34.69375
max    53.13000
Name: bmi, dtype: float64
min    0.0
25%    0.0
50%    1.0
75%    2.0
max    5.0
Name: children, dtype: float64
min     1121.873900
25%     4740.287150
50%     9382.033000
75%    16639.912515
max    63770.428010
Name: charges, dtype: float64
```

e. Distribution of ‘bmi’, ‘age’ and ‘charges’ columns.

In [15]:

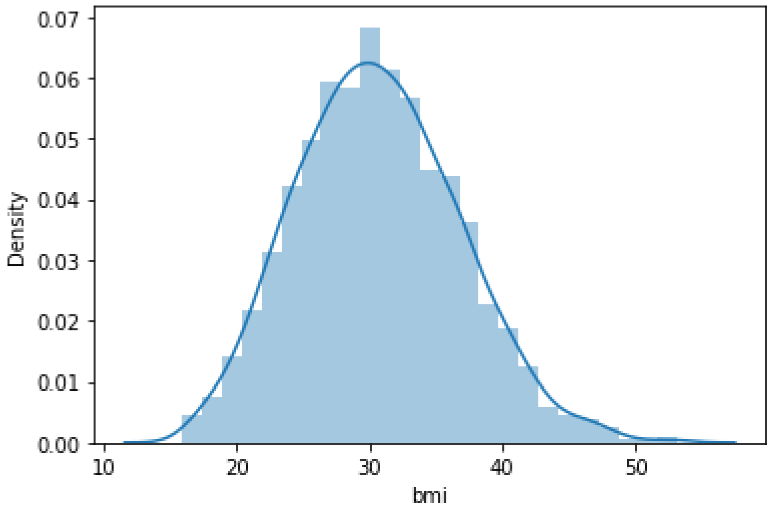
```
sns.distplot(df['bmi'])

```

```
c:\users\ritesh\appdata\local\programs\python\python39\lib\site-packages\sea
born\distributions.py:2557: FutureWarning: `distplot` is a deprecated functi
on and will be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or `histpl
ot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[15]:

```
<AxesSubplot:xlabel='bmi', ylabel='Density'>
```



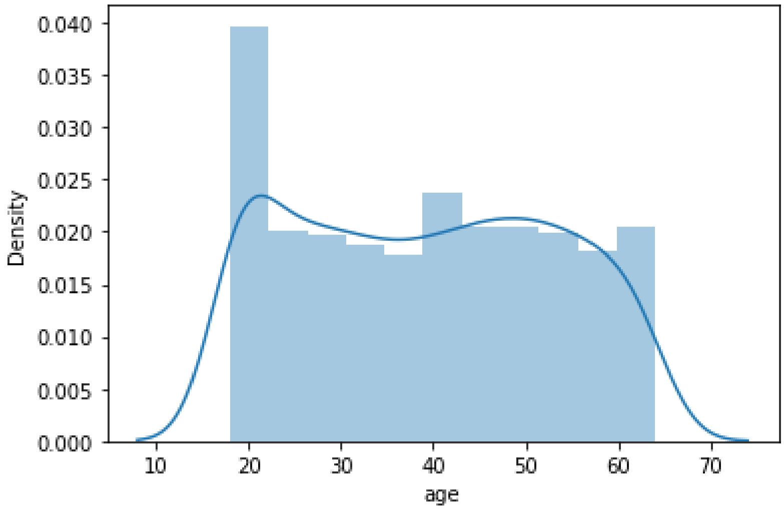
In [16]:

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

```
c:\users\ritesh\appdata\local\programs\python\python39\lib\site-packages\sea
born\distributions.py:2557: FutureWarning: `distplot` is a deprecated functi
on and will be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or `histpl
ot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[16]:

```
<AxesSubplot:xlabel='age', ylabel='Density'>
```



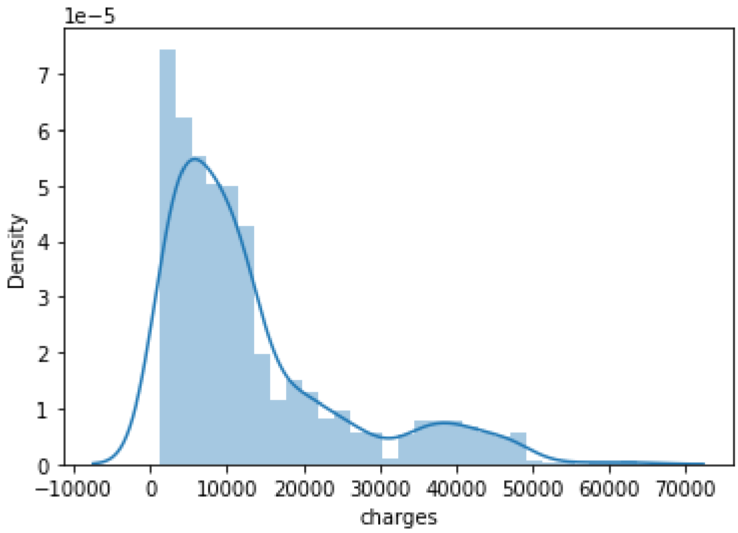
In [17]:

```
sns.distplot(df['charges'])
```

```
c:\users\ritesh\appdata\local\programs\python\python39\lib\site-packages\sea
born\distributions.py:2557: FutureWarning: `distplot` is a deprecated functi
on and will be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or `histpl
ot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[17]:

```
<AxesSubplot:xlabel='charges', ylabel='Density'>
```



In [18]:

```
# Dis_bmi=(
#     df.bmi
#      .value_counts()
#     .value_counts()
#     .sort_index())
# Dis_bmi.plot()
```

In [19]:

```
# Dis_age=(
#     df.age
#      .value_counts()
#     .value_counts()
#     .sort_index())
# Dis_age.plot()
```

In [20]:

```
# Dis_charges=(
#     df.charges
#      .value_counts()
#     .value_counts()
#     .sort_index())
# Dis_charges.plot()
```

f. Measure of skewness of ‘bmi’, ‘age’ and ‘charges’ columns

In [21]:

```
from scipy.stats import skew
```

In [22]:

```
print(skew(df.bmi, bias= False))
```

```
0.28404711059874477
```

In [23]:

```
print(skew(df.age, bias= False))
```

```
0.05567251565291854
```

In [24]:

```
print(skew(df.charges, bias= False))
```

```
1.5158796580240383
```

g. Checking the presence of outliers in ‘bmi’, ‘age’ and ‘charges columns

In [25]:

```
# import warnings
# warnings.filterwarnings('ignore')
# plt.figure(figsize=(16,5))
# plt.subplot(1,2,1)
# sns.distplot(df['bmi'])
# plt.subplot(1,2,2)
# sns.distplot(df['age'])
# plt.show()
```

In [26]:

```
# print("Highest allowed",df['bmi'].mean() + 3*df['bmi'].std())
# print("Lowest allowed",df['bmi'].mean() - 3*df['bmi'].std())
```

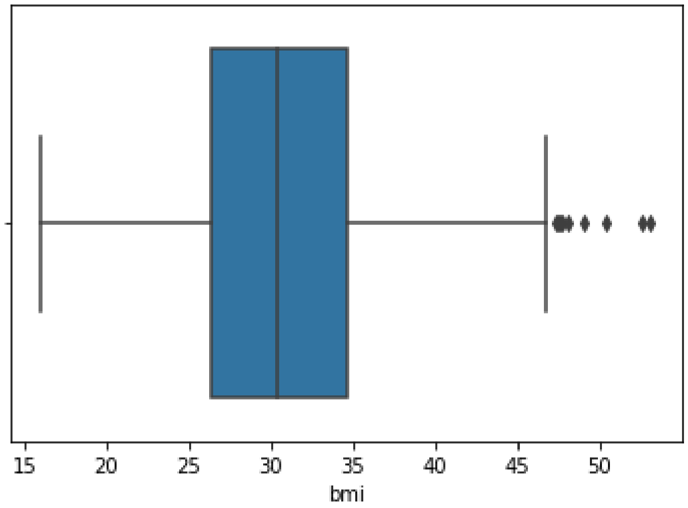
In [27]:

```
sns.boxplot(df['bmi'])
```

```
c:\users\ritesh\appdata\local\programs\python\python39\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```

Out[27]:

```
<AxesSubplot:xlabel='bmi'>
```



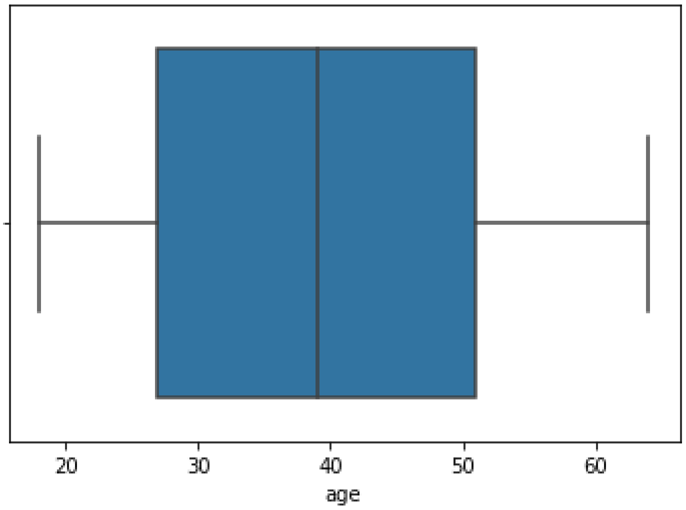
In [28]:

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

```
c:\users\ritesh\appdata\local\programs\python\python39\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```

Out[28]:

```
<AxesSubplot:xlabel='age'>
```



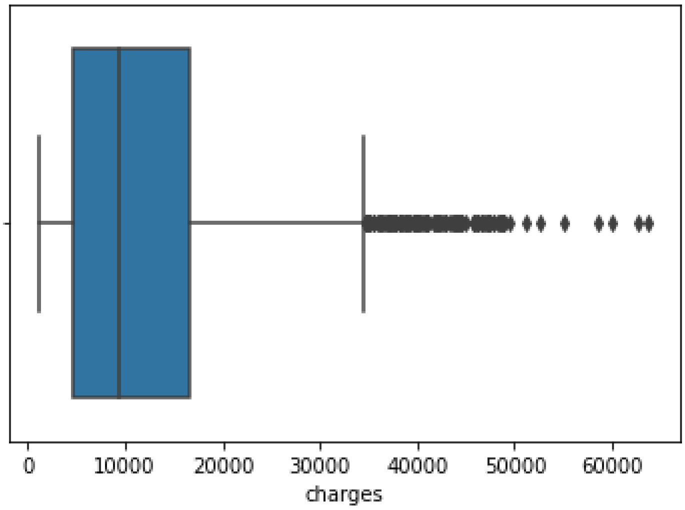
In [29]:

```
sns.boxplot(df['charges'])
```

```
c:\users\ritesh\appdata\local\programs\python\python39\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```

Out[29]:

```
<AxesSubplot:xlabel='charges'>
```



h. Distribution of categorical columns (include children)

In [30]:

```
x=df.iloc[:,:].values
y=df.iloc[:,6].values
print(x)
print(y)
```

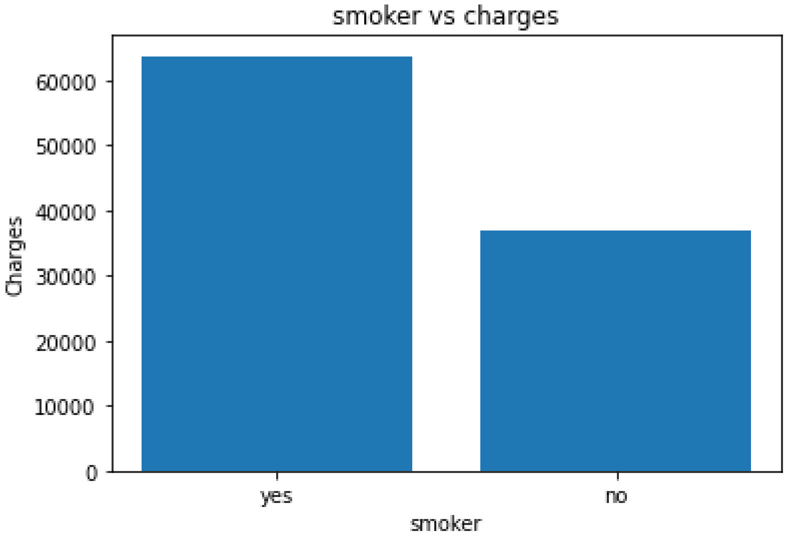
```
[[19 'female' 27.9 ... 'yes' 'southwest' 16884.924]
 [18 'male' 33.77 ... 'no' 'southeast' 1725.5523]
 [28 'male' 33.0 ... 'no' 'southeast' 4449.462]
 ...
 [18 'female' 36.85 ... 'no' 'southeast' 1629.8335]
 [21 'female' 25.8 ... 'no' 'southwest' 2007.945]
 [61 'female' 29.07 ... 'yes' 'northwest' 29141.3603]]
[16884.924   1725.5523  4449.462  ...  1629.8335  2007.945  29141.3603]
```

In [31]:

```
plt.bar(df['smoker'],y,)
plt.xlabel('smoker')
plt.ylabel('Charges')
plt.title('smoker vs charges')
plt.figure()
```

Out[31]:

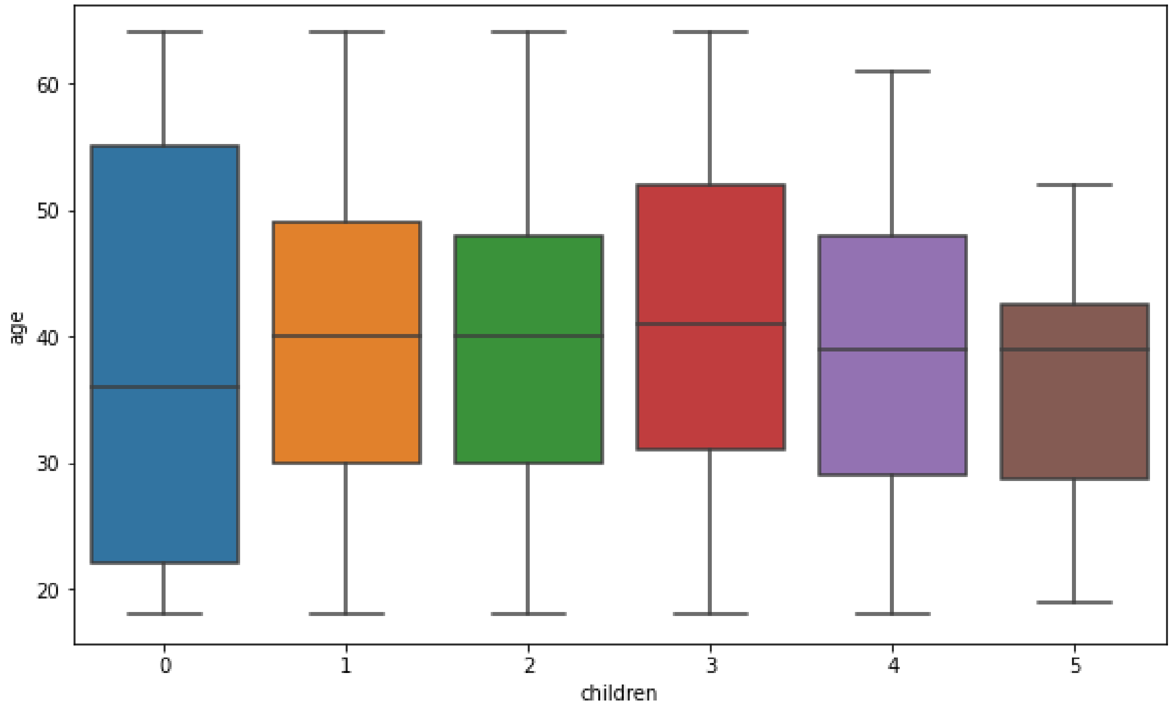
```
<Figure size 432x288 with 0 Axes>
```



```
<Figure size 432x288 with 0 Axes>
```

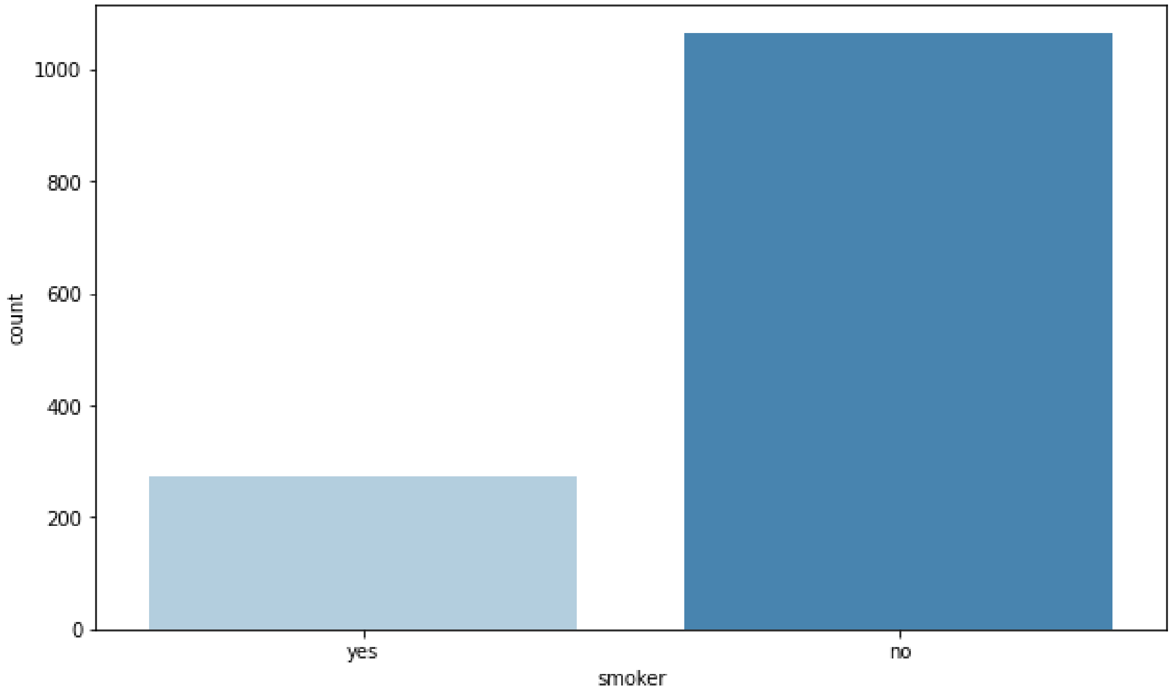
In [32]:

```
plt.figure(figsize=(10,6))
sns.boxplot(x='children',y='age',data=df)
plt.show()
```



In [33]:

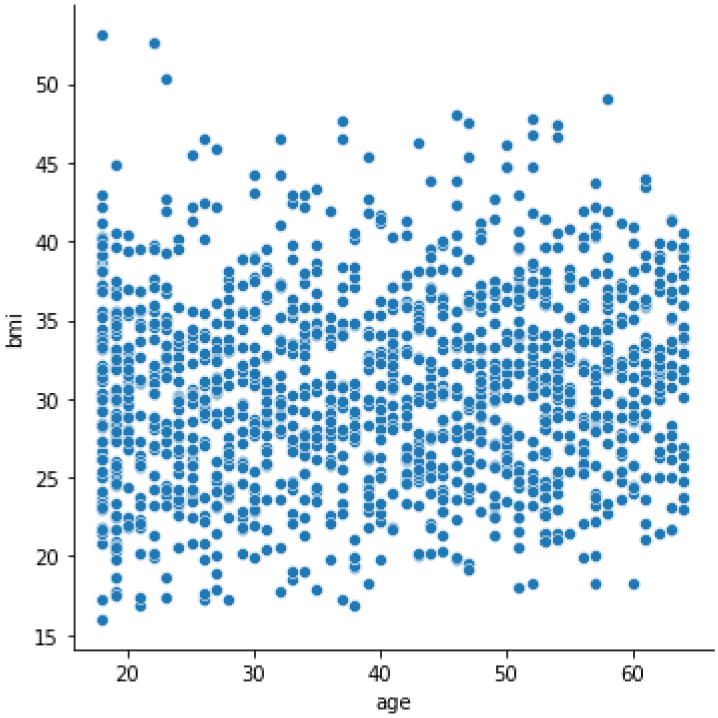
```
plt.figure(figsize=(10,6))
sns.countplot(x='smoker',data=df,palette='Blues')
plt.show()
```



In [34]:

```
plt.figure(figsize=(14,8))
sns.relplot(x='age',y='bmi',data=df)
plt.show()
```

```
<Figure size 1008x576 with 0 Axes>
```



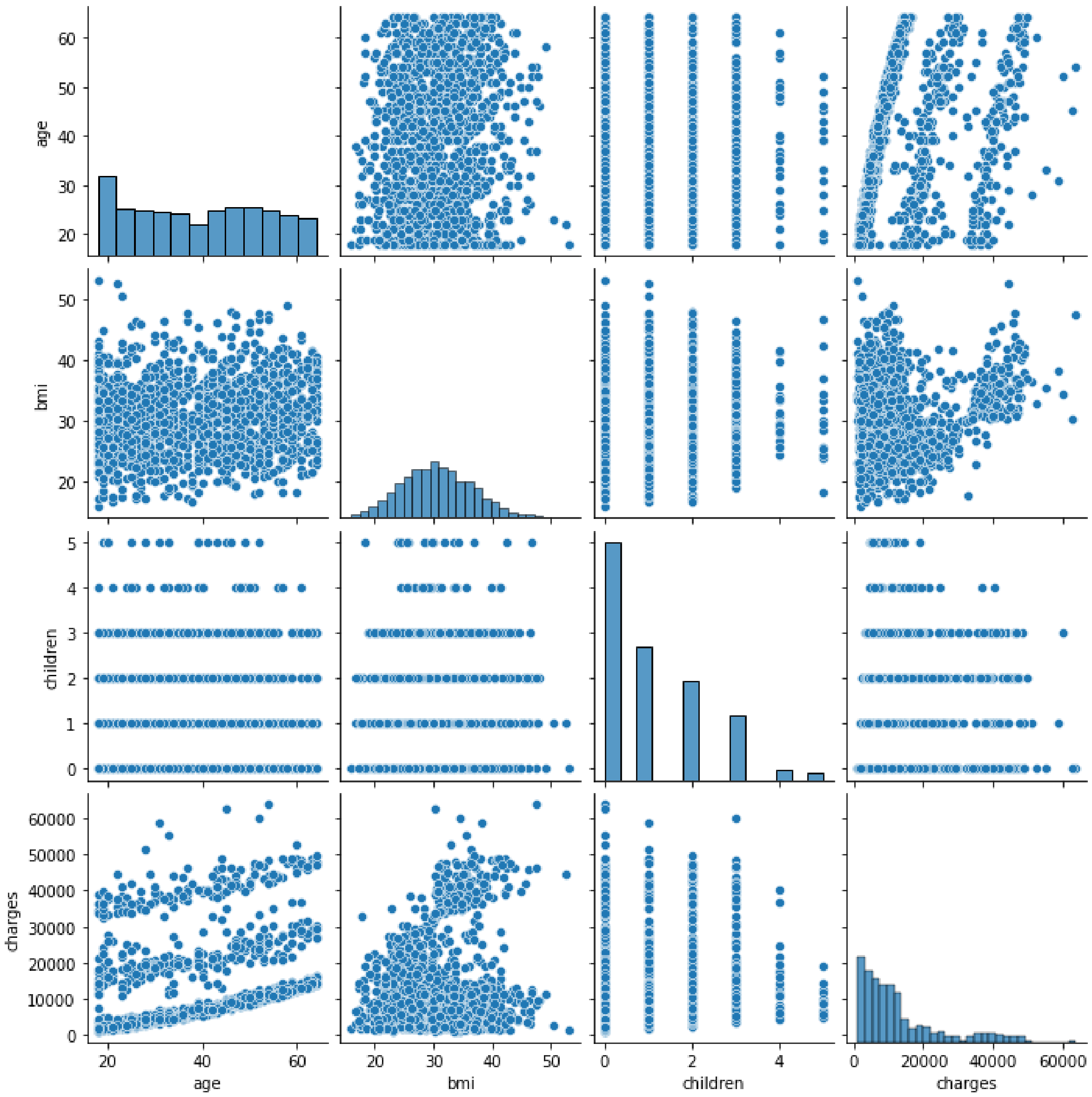
i. Pair plot that includes all the columns of the data frame

In [35]:

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

Out[35]:

```
<seaborn.axisgrid.PairGrid at 0x1a65d044e80>
```



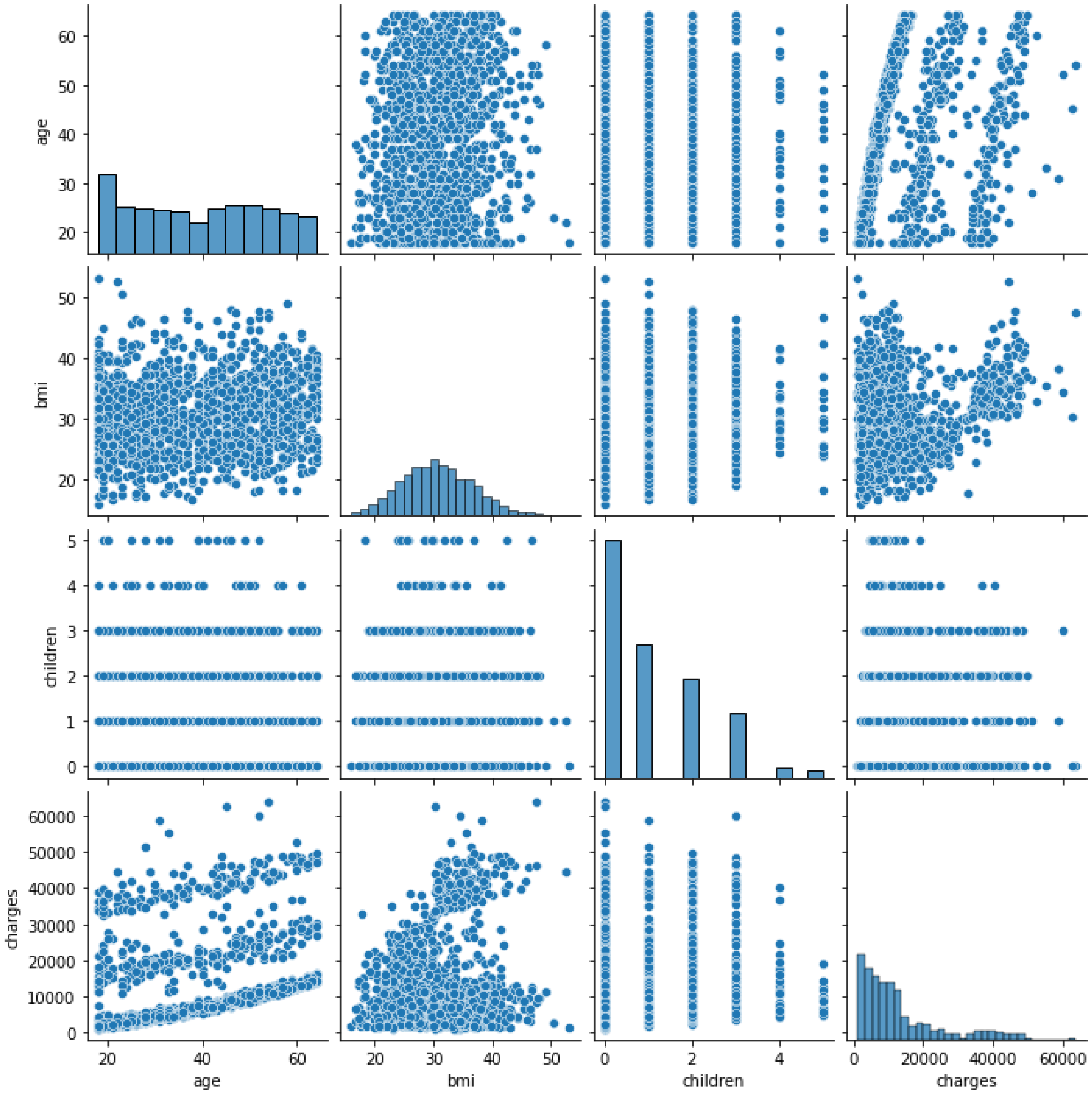
In [36]:

```python
sns.pairplot(df[['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges']])

```

Out[36]:

```
<seaborn.axisgrid.PairGrid at 0x1a65d11d7c0>
```



As there are only 4 attributes numeric in nature , the rest of them are not included in the pair plot.

The diagonal plots represent distribution of the variable relative to itself . The plot between ‘age’ and ‘bmi’ does not indicate any relationship between them. The plot is more like a cloud and lacks direction. Between ‘age’ and ‘charges’, there is no clear relationship, though there seem to be 3 lines of positive relationship between them. It means, there are 3 sets of charges which increase gradually with age. No clear relation between ‘age’ and ‘children’ either. The range of ‘bmi’ decreases as children increases, however there are some extreme values in ‘bmi’ for children value 5.

Number 4

a. Do charges of people who smoke differ significantly from the people who don't?

People who smoke have high charges and the people who do not smoke have considerably very low charges. So, charges do differ for people who smoke from the people who do not smoke but not significantly as there is some intersection of values for both types of people.

In [37]:

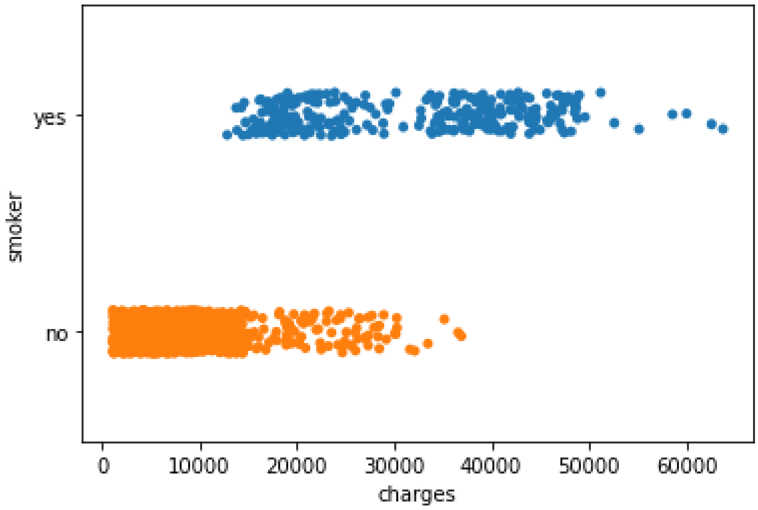
```
sns.stripplot(data['charges'], data['smoker'])

```

```
c:\users\ritesh\appdata\local\programs\python\python39\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variables as keywo
rd args: x, y. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```

Out[37]:

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



In [41]:

```
df.smoker.value_counts()
```

Out[41]:

```
no     1064
yes     274
Name: smoker, dtype: int64
```

b. Does bmi of males differ significantly from that of females?

There is no significant difference in BMI for male and female genders, so no relationship exists between the two.

In [38]:

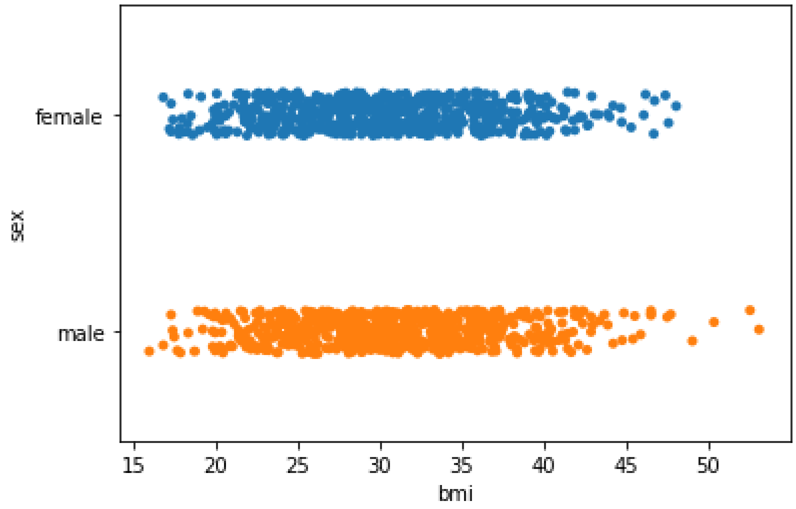
```
sns.stripplot(data['bmi'], data['sex'])

```

```
c:\users\ritesh\appdata\local\programs\python\python39\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variables as keywo
rd args: x, y. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```

Out[38]:

```
<AxesSubplot:xlabel='bmi', ylabel='sex'>
```



In [42]:

```
df.sex.value_counts()
```

Out[42]:

```
male      676
female    662
Name: sex, dtype: int64
```

c. Is the proportion of smokers significantly different in different genders?

The proportions being 58% and 42% for male and female genders who smoke are not significantly different.

In [39]:

```
r of smokers is --", df[df['smoker']=='yes'].shape[0])
r of male smokers is --", df[df['smoker']=='yes'][df['sex']=='male'].shape[0])
 of female smokers is -- ", df[df['smoker']=='yes'][df['sex']=='female'].shape[0])
of smokers who are male is --", (df[df['smoker']=='yes'][df['sex']=='male'].shape[0])/df[df
of smokers who are female is -- ", (df[df['smoker']=='yes'][df['sex']=='female'].shape[0])/
```

```
Total number of smokers is -- 274
Total number of male smokers is -- 159
Total count of female smokers is --  115
Proportion of smokers who are male is -- 0.5802919708029197
Proportion of smokers who are female is --  0.4197080291970803

<ipython-input-39-9869b121467f>:2: UserWarning: Boolean Series key will be r
eindexed to match DataFrame index.
  print("Total number of male smokers is --", df[df['smoker']=='yes'][df['se
x']=='male'].shape[0])
<ipython-input-39-9869b121467f>:3: UserWarning: Boolean Series key will be r
eindexed to match DataFrame index.
  print("Total count of female smokers is -- ", df[df['smoker']=='yes'][df
['sex']=='female'].shape[0])
<ipython-input-39-9869b121467f>:4: UserWarning: Boolean Series key will be r
eindexed to match DataFrame index.
  print("Proportion of smokers who are male is --", (df[df['smoker']=='yes']
[df['sex']=='male'].shape[0])/df[df['smoker']=='yes'].shape[0])
<ipython-input-39-9869b121467f>:5: UserWarning: Boolean Series key will be r
eindexed to match DataFrame index.
  print("Proportion of smokers who are female is -- ", (df[df['smoker']=='ye
s'][df['sex']=='female'].shape[0])/df[df['smoker']=='yes'].shape[0])
```

d. Is the distribution of bmi across women with no children, one child and two children, the same?

Yes, the distributions of ‘bmi’ are nearly same across women with 0, 1 or 2 children.

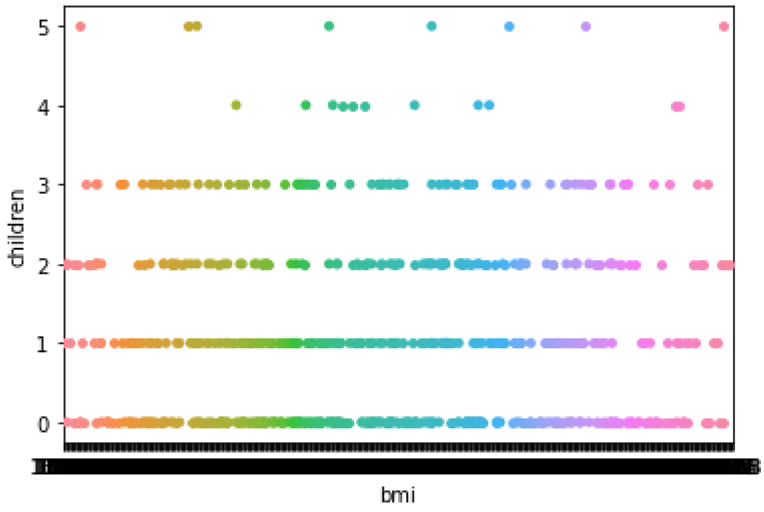
In [40]:

```
sns.stripplot(df['bmi'], df[df['sex']=='female']['children'])
```

```
c:\users\ritesh\appdata\local\programs\python\python39\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variables as keywo
rd args: x, y. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```

Out[40]:

```
<AxesSubplot:xlabel='bmi', ylabel='children'>
```



Number 5

In [62]:

```
from sklearn.datasets import make_classification

X,Y=make_classification()
X,Y
```

Out[62]:

```
(array([[-0.35346275,  0.11470425, -0.76527075, ..., -1.30705054,
          0.8945163 , -0.16308364],
        [ 0.45616974,  0.45554952,  0.69630583, ..., -0.25875887,
          1.83840011,  1.91229045],
        [ 0.60698229,  2.01889117,  0.60645728, ..., -0.0211031 ,
          0.10671858, -0.91777694],
        ...,
        [-0.11235949,  0.26517082, -1.025135  , ...,  0.64714497,
         -1.95875912,  0.68709827],
        [-0.44086909, -1.6436062 ,  0.31800784, ...,  2.69867185,
          0.89515976, -0.5175695 ],
        [-0.61268517, -1.80563453, -1.19828744, ..., -1.39129918,
         -1.32068727, -0.89897423]]),
 array([1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0,
        0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0,
        1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1,
        0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
        0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0]))
```

In [66]:

```
print("The shape of X is {} and Y is {}".format(X.shape,Y.shape))
```

```
The shape of X is (1338, 6) and Y is (1338,)
```

In [67]:

```
np.unique(Y)
```

Out[67]:

```
array([ 1121.8739 ,  1131.5066 ,  1135.9407 , ..., 60021.39897,
       62592.87309, 63770.42801])
```

In [71]:

```
data = pd.DataFrame(np.c_[X,Y])
```

In [72]:

```
data
```

Out[72]:

	0	1	2	3	4	5	6
0	19.0	1.0	27.900	0.0	0.0	1.0	16884.92400
1	18.0	0.0	33.770	1.0	1.0	0.0	1725.55230
2	28.0	0.0	33.000	3.0	1.0	0.0	4449.46200
3	33.0	0.0	22.705	0.0	1.0	3.0	21984.47061
4	32.0	0.0	28.880	0.0	1.0	3.0	3866.85520
...	...	...	...	...	...	...	...
1333	50.0	0.0	30.970	3.0	1.0	3.0	10600.54830
1334	18.0	1.0	31.920	0.0	1.0	2.0	2205.98080
1335	18.0	1.0	36.850	0.0	1.0	0.0	1629.83350
1336	21.0	1.0	25.800	0.0	1.0	1.0	2007.94500
1337	61.0	1.0	29.070	0.0	0.0	3.0	29141.36030

1338 rows × 7 columns

In [ ]:

```

```

Number 6

Data Pre-Processing

Encoding the categorical features

In [63]:

```
# encoding sex column
df.replace({'sex':{'male':0,'female':1}}, inplace=True)

3 # encoding 'smoker' column
df.replace({'smoker':{'yes':0,'no':1}}, inplace=True)

# encoding 'region' column
df.replace({'region':{'southeast':0,'southwest':1,'northeast':2,'northwest':3}}, inplac
```

In [64]:

```
X = df.drop(columns='charges', axis=1)
Y = df['charges']
```

In [65]:

```
print(X)
```

```
      age  sex     bmi  children  smoker  region
0      19    1  27.900         0       0       1
1      18    0  33.770         1       1       0
2      28    0  33.000         3       1       0
3      33    0  22.705         0       1       3
4      32    0  28.880         0       1       3
...   ...  ...     ...       ...     ...     ...
1333   50    0  30.970         3       1       3
1334   18    1  31.920         0       1       2
1335   18    1  36.850         0       1       0
1336   21    1  25.800         0       1       1
1337   61    1  29.070         0       0       3

[1338 rows x 6 columns]
```

In [47]:

```
print(Y)
```

```
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
           ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

In [48]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2
```

In [49]:

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(1338, 6) (1070, 6) (268, 6)
```

Model Training

Linear Regression

In [50]:

```
regressor = LinearRegression()
```

In [51]:

```
regressor.fit(X_train, Y_train)
```

Out[51]:

```
LinearRegression()
```

In [52]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Out[52]:

```
LinearRegression()
```

Model Evaluation

In [53]:

```
# prediction on training data
training_data_prediction =regressor.predict(X_train)
```

In [54]:

```
# R squared value
r2_train = metrics.r2_score(Y_train, training_data_prediction)
print('R squared vale : ', r2_train)
```

```
R squared vale :  0.751505643411174
```

In [55]:

```
# prediction on test data
test_data_prediction =regressor.predict(X_test)
```

In [56]:

```
r2_test = metrics.r2_score(Y_test, test_data_prediction)
print('R squared vale : ', r2_test)
```

```
R squared vale :  0.7447273869684077
```

In [59]:

```
input_data = (31,1,25.74,0,1,0)

# changing input_data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = regressor.predict(input_data_reshaped)
print(prediction)

print('The insurance cost is USD ', prediction[0])
```

```
[3760.0805765]
The insurance cost is USD  3760.0805764960514
```

In [ ]:

```

```