In [1]:

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

In [2]:

```
data = pd.read_csv('datasets_13720_18513_insurance.csv')

print(data.shape)

data.head()
```

```
(1338, 7)
```

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

In [3]:

```
data.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]:

```
data.describe()
```

Out[4]:

	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 [5]:

```
data.isnull().sum()
```

Out[5]:

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

In [6]:

```
data['sex'].value_counts()
```

Out[6]:

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

In [7]:

```
data['age'].value_counts()
```

Out[7]:

```
18    69
19    68
51    29
45    29
46    29
47    29
48    29
50    29
52    29
20    29
26    28
54    28
53    28
25    28
24    28
49    28
23    28
22    28
21    28
27    28
28    28
31    27
29    27
30    27
41    27
43    27
44    27
40    27
42    27
57    26
34    26
33    26
32    26
56    26
55    26
59    25
58    25
39    25
38    25
35    25
36    25
```

```
37    25
63    23
60    23
61    23
62    23
64    22
Name: age, dtype: int64
```

In [8]:

```
data['children'].value_counts()
```

Out[8]:

```
0    574
1    324
2    240
3    157
4     25
5     18
Name: children, dtype: int64
```

In [9]:

```
data['smoker'].value_counts()
```

Out[9]:

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

In [10]:

```
data['region'].value_counts()
```

Out[10]:

```
southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

In [11]:

```
data.columns
```

Out[11]:

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

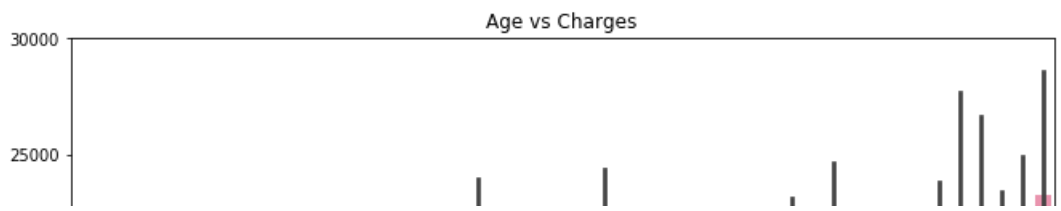
In [12]:

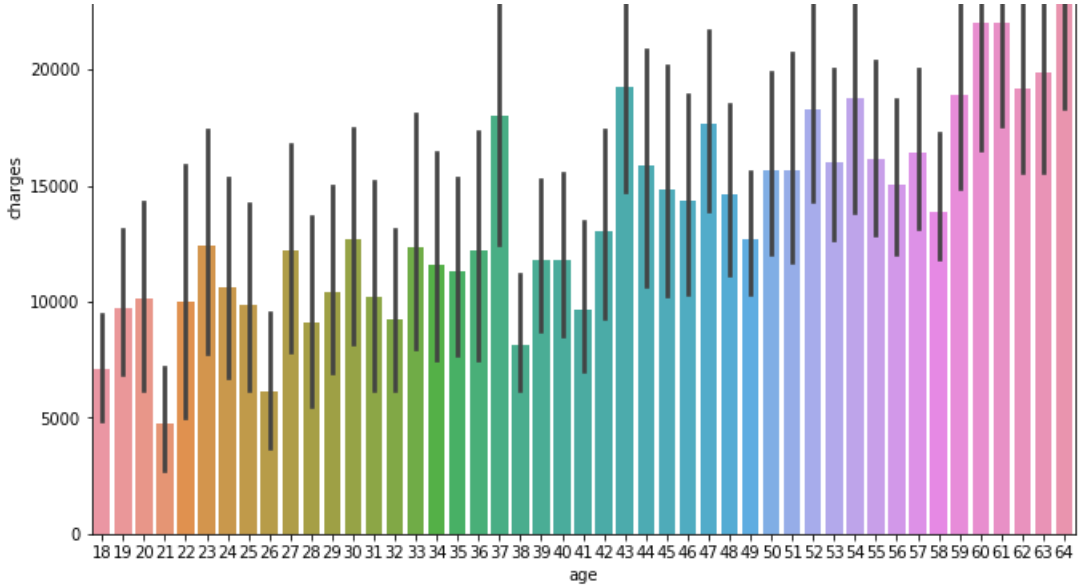
```
plt.figure(figsize = (11, 8))
sns.barplot(x = 'age', y = 'charges', data = data)

plt.title("Age vs Charges")
```

Out[12]:

```
Text(0.5, 1.0, 'Age vs Charges')
```





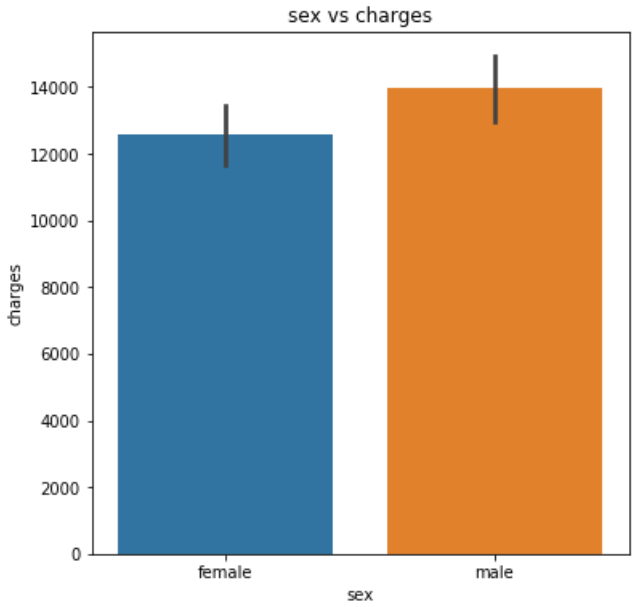
In [13]:

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

plt.title('sex vs charges')
```

Out[13]:

```
Text(0.5, 1.0, 'sex vs charges')
```



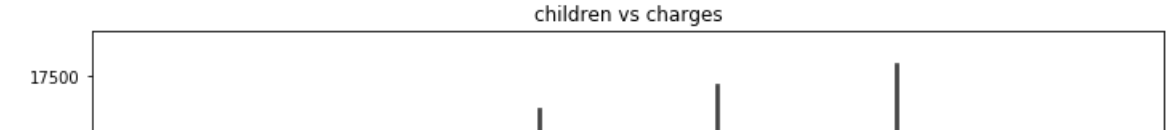
In [14]:

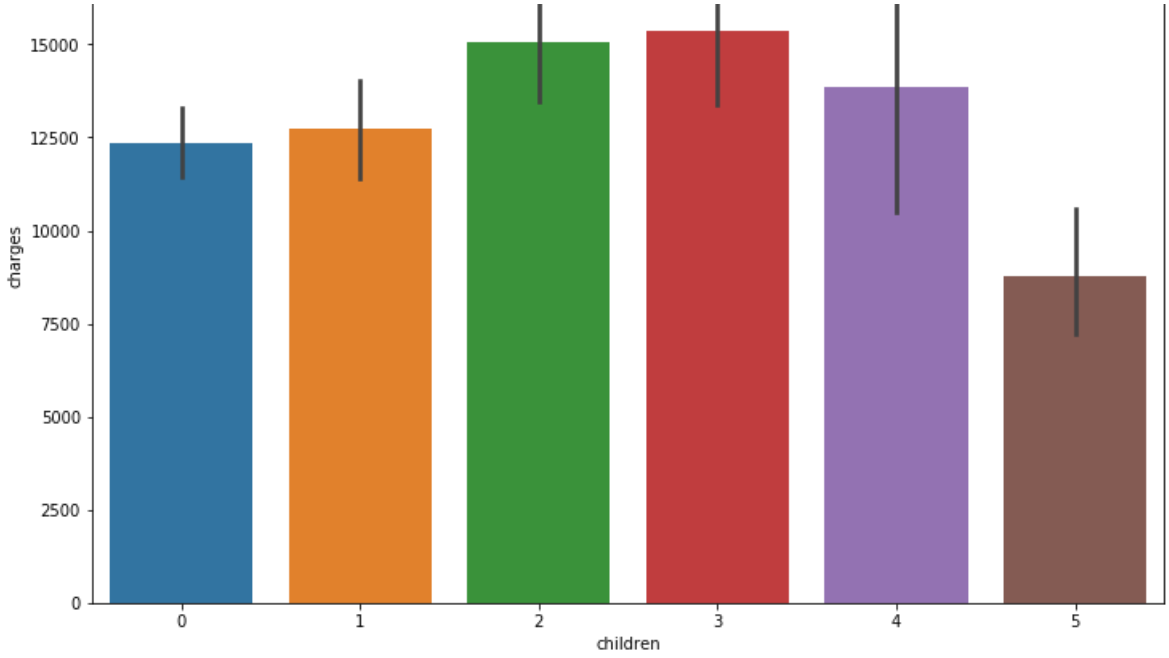
```
plt.figure(figsize = (12, 8))
sns.barplot(x = 'children', y = 'charges', data = data)

plt.title('children vs charges')
```

Out[14]:

```
Text(0.5, 1.0, 'children vs charges')
```





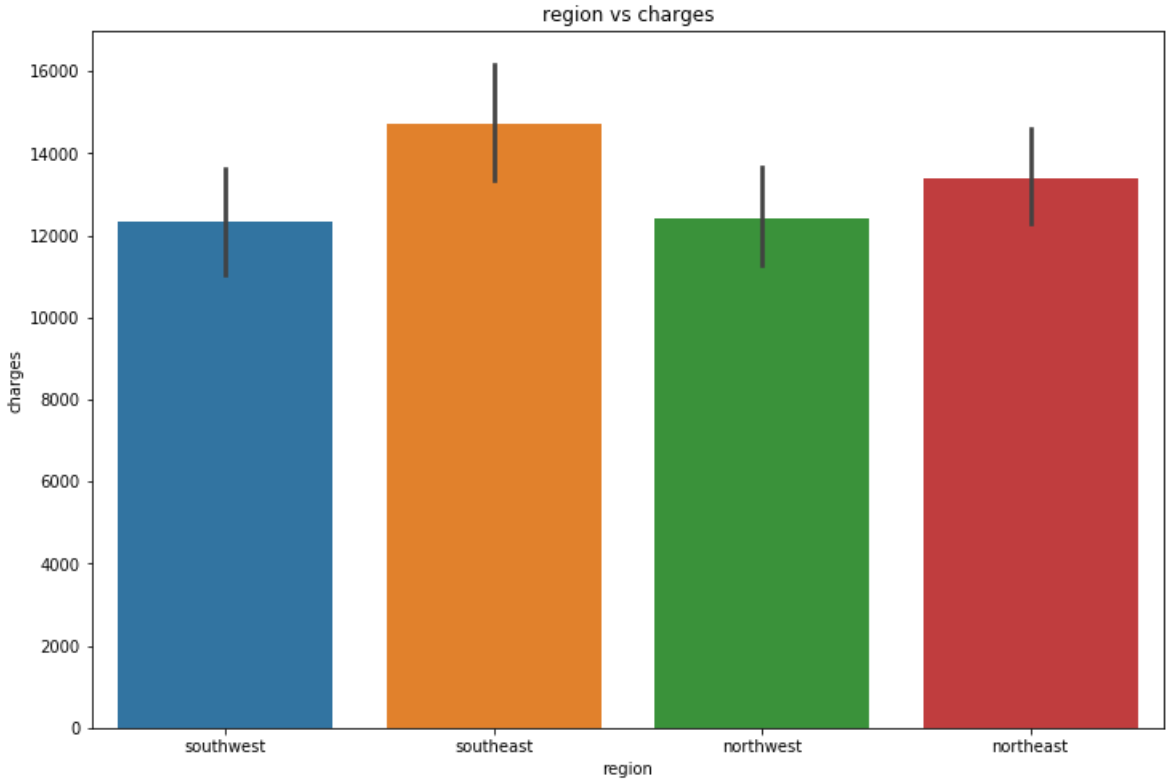
In [15]:

```
plt.figure(figsize = (12, 8))
sns.barplot(x = 'region', y = 'charges', data = data)

plt.title('region vs charges')
```

Out[15]:

```
Text(0.5, 1.0, 'region vs charges')
```



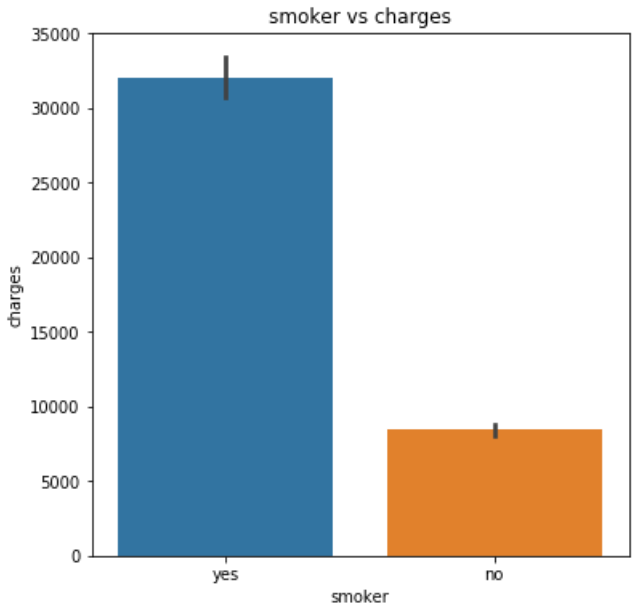
In [16]:

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

plt.title('smoker vs charges')
```

Out[16]:

```
Text(0.5, 1.0, 'smoker vs charges')
```



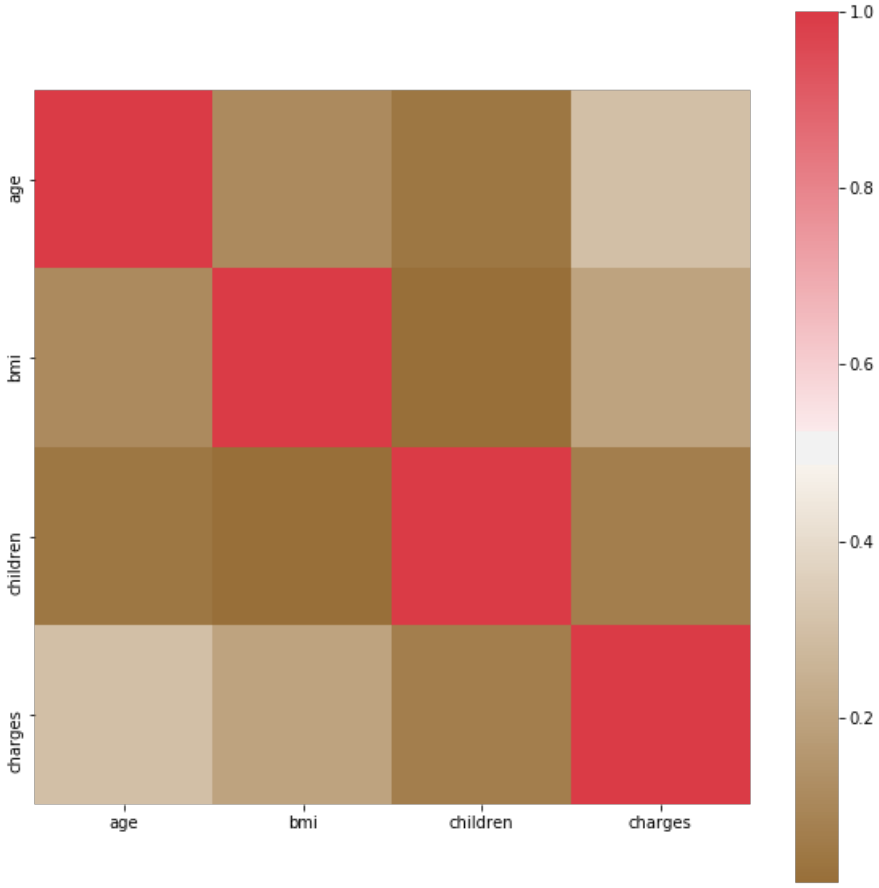
In [17]:

```
f, ax = plt.subplots(figsize = (10, 10))

corr = data.corr()
sns.heatmap(corr, mask = np.zeros_like(corr, dtype = np.bool),
            cmap = sns.diverging_palette(50, 10, as_cmap = True), square = True, ax = ax)
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x13a90f532b0>
```



In [18]:

```
data = data.drop('region', axis = 1)

print(data.shape)

data.columns
```

```
(1338, 6)
```

Out[18]:

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

In [19]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['sex'] = le.fit_transform(data['sex'])
data['smoker'] = le.fit_transform(data['smoker'])
```

In [20]:

```
data['sex'].value_counts()
```

Out[20]:

```
1    676
0    662
Name: sex, dtype: int64
```

In [21]:

```
data['smoker'].value_counts()
```

Out[21]:

```
0    1064
1     274
Name: smoker, dtype: int64
```

In [22]:

```
x = data.iloc[:,:5]
y = data.iloc[:,5]

print(x.shape)
print(y.shape)
```

```
(1338, 5)
(1338,)
```

In [23]:

```
conda install -c anaconda scikit-learn
```

```
Note: you may need to restart the kernel to use updated packages.
```

```
Traceback (most recent call last):
  File "C:\Users\KIIT\Documents\ml\Scripts\conda-script.py", line 11, in <module>
    from conda.cli import main
ImportError: No module named conda.cli
```

In [24]:

```
import sklearn
print (sklearn.__version__)
```

```
0.23.1
```

In [25]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 30)

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1070, 5)
(268, 5)
(1070,)
(268,)
```

In [29]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

In [ ]:

```

```

In [39]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score


lineReg = LinearRegression()
lineReg.fit(x_train, y_train)
lineReg.score(x_test, y_test )
mse = np.mean((y_test - y_pred)**2, axis = None)
print("MSE :", mse)


rmse = np.sqrt(mse)
print("RMSE :", rmse)


r2 = r2_score(y_test, y_pred)
print("r2_score :", r2)
```

```
MSE : 42117696.87942878
RMSE : 6489.814857099452
r2_score : 0.7269570913940737
```

In [40]:

```
y_test
```

Out[40]:

```
338    41919.09700
620     3659.34600
965     4746.34400
128    32734.18630
329     9144.56500
          ...
580    12913.99240
786    12741.16745
321    24671.66334
903     8125.78450
613     6753.03800
```

```
Name: charges, Length: 268, dtype: float64
```

In [41]:

```
y_pred
```

Out[41]:

```
array([41661.602   ,  3956.07145 ,  4894.7533  , 19023.26    ,
        8457.818   , 18903.49141 ,  1141.4451  ,  9225.2564  ,
       62592.87309 , 10959.33    ,  5484.4673  , 20878.78443 ,
       42111.6647  , 11938.25595 ,  1131.5066  , 25678.77845 ,
       20878.78443 ,  9101.798   , 18765.87545 ,  7345.084   ,
        1131.5066  ,  4889.9995  ,  9644.2525  , 12797.20962 ,
       24520.264   , 45710.20785 , 11289.10925 ,  4544.2348  ,
        6082.405   ,  9304.7019  , 14426.07385 , 37133.8982  ,
        1515.3449  ,  9880.068   , 11735.87905 ,  6551.7501  ,
        2007.945   ,  6067.12675 ,  1708.0014  , 20420.60465 ,
        4438.2634  , 24106.91255 ,  1815.8759  , 10594.2257  ,
        6079.6715  , 42112.2356  , 20177.67113 ,  1136.3994  ,
        2639.0429  ,  4830.63    ,  8538.28845 ,  8733.22925 ,
       12222.8983  ,  1967.0227  ,  7209.4918  ,  6117.4945  ,
       19933.458   ,  3353.4703  , 11735.87905 ,  4189.1131  ,
       10450.552   ,  5438.7491  , 10141.1362  ,  3866.8552  ,
        1702.4553  ,  3766.8838  , 26140.3603  , 14426.07385 ,
        4766.022   , 10601.412   , 12523.6048  , 11774.159275,
        1711.0268  ,  8606.2174  , 14119.62    ,  2709.1119  ,
        7740.337   ,  7742.1098  , 17496.306   ,  2690.1138  ,
        2196.4732  ,  7633.7206  ,  5910.944   , 20296.86345 ,
        2196.4732  , 12950.0712  ,  6082.405   , 13470.8044  ,
        8688.85885 ,  7512.267   , 12557.6053  , 10085.846   ,
       36189.1017  ,  6875.961   , 27218.43725 ,  2498.4144  ,
        9282.4806  , 11884.04858 ,  5472.449   , 41999.52    ,
       41097.16175 , 10594.50155 ,  7512.267   ,  3227.1211  ,
       38415.474   , 15817.9857  ,  5926.846   , 41661.602   ,
       32787.45859 ,  2102.2647  , 10065.413   ,  7986.47525 ,
        2803.69785 , 19361.9988  ,  3556.9223  ,  4718.20355 ,
       27346.04207 ,  5989.52365 ,  8116.26885 ,  8023.13545 ,
        1727.54    ,  6610.1097  , 28101.33305 , 11881.9696  ,
       19107.7796  , 13012.20865 , 12638.195   ,  7729.64575 ,
       18767.7377  , 11305.93455 ,  9861.025   ,  2867.1196  ,
       42112.2356  , 34779.615   , 11657.7189  ,  1639.5631  ,
       47269.854   , 17081.08    , 11741.726   ,  3987.926   ,
        9625.92    , 26140.3603  , 23288.9284  , 43578.9394  ,
        5375.038   ,  9957.7216  ,  3227.1211  ,  4906.40965 ,
        7256.7231  , 24520.264   , 24227.33724 ,  7742.1098  ,
        5920.1041  ,  6250.435   ,  6238.298   ,  7512.267   ,
        7882.429475, 11286.5387  , 10325.206   ,  1633.9618  ,
        9704.66805 , 35491.64    ,  9541.69555 ,  7337.748   ,
       10115.00885 , 13224.05705 , 43753.33705 ,  5373.36425 ,
        4151.0287  ,  2302.3     ,  6457.8434  , 11305.93455 ,
        6402.29135 ,  4438.2634  , 39125.33225 ,  8211.1002  ,
        1135.9407  , 17878.90068 , 12638.195   , 17496.306   ,
        8515.7587  , 10564.8845  ,  2007.945   ,  2117.33885 ,
       46889.2612  , 37165.1638  , 19023.26    , 18804.7524  ,
       37079.372   ,  8798.593   ,  1121.8739  ,  3292.52985 ,
       12622.1795  , 10560.4917  , 11774.159275, 11658.37915 ,
       16069.08475 , 21348.706   , 37133.8982  ,  4992.3764  ,
        6082.405   ,  1135.9407  ,  6067.12675 ,  4402.233   ,
       36219.40545 ,  2913.569   ,  2166.732   , 43921.1837  ,
       42211.1382  , 43753.33705 ,  4846.92015 ,  5253.524   ,
       47269.854   ,  7742.1098  ,  3554.203   , 46255.1125  ,
        3556.9223  ,  2203.73595 , 10226.2842  , 16657.71745 ,
        6571.544   , 46718.16325 ,  6948.7008  ,  1252.407   ,
        3353.4703  ,  7640.3092  ,  6079.6715  ,  3866.8552  ,
        7077.1894  , 11482.63485 ,  7512.267   , 35160.13457 ,
        7337.748   , 11090.7178  ,  8277.523   , 19350.3689  ,
       38792.6856  , 11305.93455 ,  5926.846   , 37465.34375 ,
       12979.358   , 13224.693   ,  8442.667   ,  1759.338   ,
       31620.00106 , 13012.20865 , 11774.159275,  1532.4697  ,
        3385.39915 , 10226.2842  ,  4391.652   ,  8825.086   ,
        9304.7019  , 28868.6639  , 17178.6824  ,  4561.1885  ,
       10043.249   ,  9174.13565 , 10959.33    , 35147.52848 ,
        6079.6715  , 12244.531   ,  3761.292   , 11554.2236  ,
       11363.2832  , 17128.42608 , 20878.78443 ,  4766.022   ])
```

In [42]:

```
from sklearn.svm import SVR


model = SVR()

model.fit(x_train, y_train)


y_pred = model predicting the test set .predict(x_test)


mse = np.mean((y_test - y_pred)**2, axis = None)
print("MSE :", mse)


rmse = np.sqrt(mse)
print("RMSE :", rmse)


r2 = r2_score(y_test, y_pred)
print("r2 score :", r2)
```

```
MSE : 174595630.81989565
RMSE : 13213.46399775228
r2 score : -0.13187810353027452
```

In [43]:

```
from sklearn.ensemble import RandomForestRegressor


model = RandomForestRegressor(n_estimators = 40, max_depth = 4, n_jobs = -1)


model.fit(x_train, y_train)


y_pred = model.predict(x_test)


mse = np.mean((y_test - y_pred)**2, axis = None)
print("MSE :", mse)


rmse = np.sqrt(mse)
print("RMSE :", rmse)


r2 = r2_score(y_test, y_pred)
print("r2 score :", r2)
```

```
MSE : 21342553.68193347
RMSE : 4619.800177706117
r2 score : 0.8616393258378883
```

In [44]:

```
print(y_pred)
```

```
[44616.18316753  5572.0692209   5763.82025333 17263.43896387
 10386.6538947   5633.18418697  2716.63133736 10660.88374645
 42755.91164331 13198.8455242   6692.81846882 10641.33292496
 45641.18166748 12815.34492098  2716.63133736 24939.51504809
 10564.71060647  9224.93908308 18330.70996928 10070.2450575
  2716.63133736  6473.43961546 10381.51852926  7888.09271167
 25376.41532092 45238.0854621  13474.62667033  4774.17920073
  6958.0996881  10763.16164063  5016.41093546 37477.65895957
  2921.21489584 10672.56008039 13257.8718383   6473.43961546
  2716.63133736  6692.81846882  2716.63133736  6318.72989031
  6577.88231487 23726.05330159  4085.65667142 13163.60486493
```

```
 6651.75290518 38751.7750549   6592.50571255  2716.63133736
 3825.87696535  6867.55630424  7122.92987681 10170.49259516
13198.8455242   2716.63133736  7001.66452853  7392.15658253
19879.37476325  4774.17920073 13257.8718383   6499.18637607
13273.22516847  6499.80147294 10928.47175937  4724.55456923
 2716.63133736  5531.00365726 10750.68451576  5118.80093969
 5915.83488906 13400.08615106 14305.54249296  2716.63133736
 3598.55111805  9830.73688297 15778.31709922  3598.55111805
 9347.62480949 10125.95494825 20706.04271389  4085.65667142
 2716.63133736  9198.65674219  6223.01079934 18973.80084743
 2716.63133736 14465.76336945  7053.81877907 14612.72030611
10360.54430153  7606.15321828 14560.45605192 13084.96290483
36482.45325338  7079.36503637 26990.97857616  4364.4728668
10695.78357934  7071.30955636  6388.54805289 45238.0854621
44483.92936695 13163.60486493  7507.93485051  4975.34537182
39814.16184086 17820.07452606  6677.78136632 44616.18316753
26275.14417281  2921.21489584 12963.6174723   7307.78516143
 7173.69956058 18097.26800585  4364.4728668   5785.19882269
13163.60486493  6517.00445589  8378.56803982  9941.41110933
 3455.09555019  6993.4170663  25888.88000851 13623.25360304
18973.80084743 13868.2211008  14305.54249296  7240.1419468
23518.67779839 13020.97861477 10660.79163782  4724.55456923
40556.96063394 36243.30418219 13163.60486493  2716.63133736
45803.84647835 17727.20468935 13634.90524573  6434.79527352
12855.61049903 10753.28922816  5621.6938524  45238.0854621
 6892.87350418 10568.38615611  5240.59597321  6737.95532871
 7120.43060001 25534.87896608 13273.22516847  9494.80393337
 6223.01079934  7747.71058785  6692.81846882  7401.64310296
 2716.63133736 13163.60486493 13556.8300636   2716.63133736
 9967.2490732  39034.84563075 10555.31978604  9887.55834101
10946.62535232 13658.4942623  41830.1744174   6388.54805289
 5592.11862333  4364.4728668   6999.16525174 13080.76656556
 7062.20948247  5473.98310124 39363.71925418  9249.2125732
 2716.63133736  4724.55456923 14305.54249296 20706.04271389
10020.58004324 13213.43721768  2716.63133736  2716.63133736
45393.84636998 38116.50167607 17332.8856352   6516.35920619
39376.75302998 10462.91430433  2716.63133736  5240.59597321
13868.2211008  10743.56868732  2716.63133736 13163.60486493
16074.46479349 25728.20153321 37936.83847412  5655.88297381
 6958.0996881   2716.63133736  6692.81846882  5826.20305806
37871.77604381  7217.26440102  2921.21489584 45238.0854621
43372.85406811 43927.93062803  6480.47066562  6516.35920619
45803.84647835  9494.80393337  4724.55456923 45651.51406074
 4364.4728668   2716.63133736 13548.35870238 17332.8856352
 7097.3836195  45238.0854621   7888.7447712   2716.63133736
 4774.17920073  7163.99544045  6651.75290518  4724.55456923
 6757.69195634  2716.63133736  7551.49969095 13020.97861477
 9826.67372133 13163.60486493 10633.4442214  26876.61071211
38751.7750549  13080.76656556  6849.30866375 36792.07635379
14612.72030611 15432.96641439 10381.51852926  2716.63133736
15263.14446287 13868.2211008   2716.63133736  2921.21489584
 4565.3534835  13400.08615106  6616.02972899 10165.4475041
10686.901231   26246.59780014 18632.77860517  6951.59820957
10758.20865819 10457.86921327 13198.8455242  19077.32206552
 6651.75290518 13868.2211008   4724.55456923 13347.11807551
13413.07804578  6604.27432518 10570.1176064   6788.08518871]
```

In [45]:

```
print(y_test)
```

```
338    41919.09700
620     3659.34600
965     4746.34400
128    32734.18630
329     9144.56500
          ...
580    12913.99240
786    12741.16745
321    24671.66334
903     8125.78450
613     6753.03800
Name: charges, Length: 268, dtype: float64
```

In [38]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score


model = DecisionTreeRegressor()


model.fit(x_train, y_train)


y_pred = model.predict(x_test)


mse = np.mean((y_test - y_pred)**2, axis = None)
print("MSE :", mse)


rmse = np.sqrt(mse)
print("RMSE :", rmse)


r2 = r2_score(y_test, y_pred)
print("r2 score :", r2)
```

```
MSE : 42117696.87942878
RMSE : 6489.814857099452
r2 score : 0.7269570913940737
```

In [46]:

```
from sklearn.preprocessing import PolynomialFeatures
poly_regs= PolynomialFeatures(degree= 2)
x_poly= poly_regs.fit_transform(x)
lin_reg_2 =LinearRegression()
lin_reg_2.fit(x_poly, y)
```

Out[46]:

```
LinearRegression()
```

In [68]:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
poly = PolynomialFeatures(degree = 4)
x_poly = poly.fit_transform(x_train)

lin2 = LinearRegression()
lin2.fit(x_poly ,y_train)

predicted_y_test = lin2.predict(poly.fit_transform(x_test))

print("MAE Error: ", mean_absolute_error(predicted_y_test,y_test))
print("MSE Error: ", mean_squared_error(predicted_y_test,y_test))
print("r2 Error: ", r2_score(predicted_y_test,y_test))
```

```
MAE Error:  835723193305814.9
MSE Error:  1.4158493020200626e+30
r2 Error:  -0.1324753671672294
```

In [ ]:

```

```