Health Insurance Cost Prediction Using Linear Regression

In [1]:
```
# import libarires
```

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import style
```

In [3]:
```
# Data Collection & Analysis
```

In [4]:
```
df = pd.read_csv("insurance.csv")
df
```

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
...	...	...	...	...	...	...	...
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 [5]:
```
df.head()
```

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

In [6]:
```
df.shape
```

Out[6]:
```
(1338, 7)
```

In [7]:
```
# getting some informations about the dataset
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]:
```
# checking for missing values
df.isnull().sum()
```

Out[8]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
```

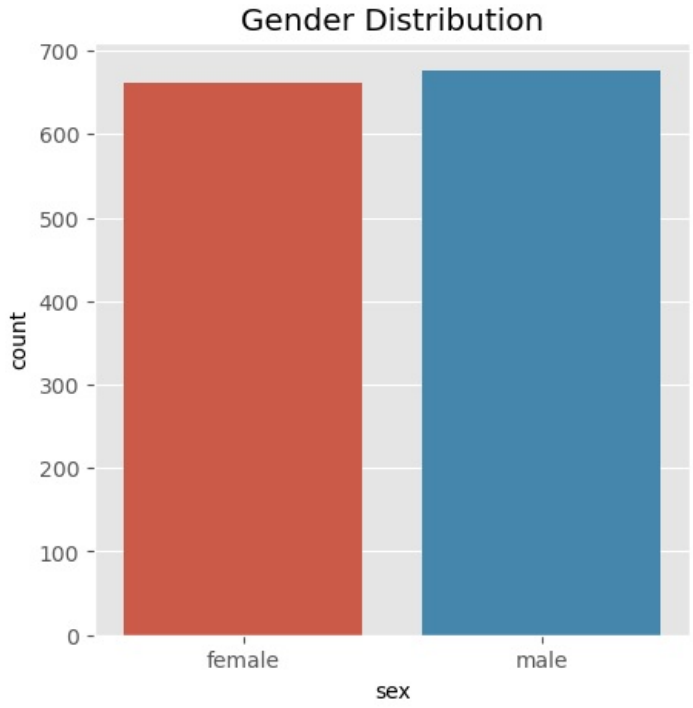
In [10]:
```
df.describe()
```

Out[10]:

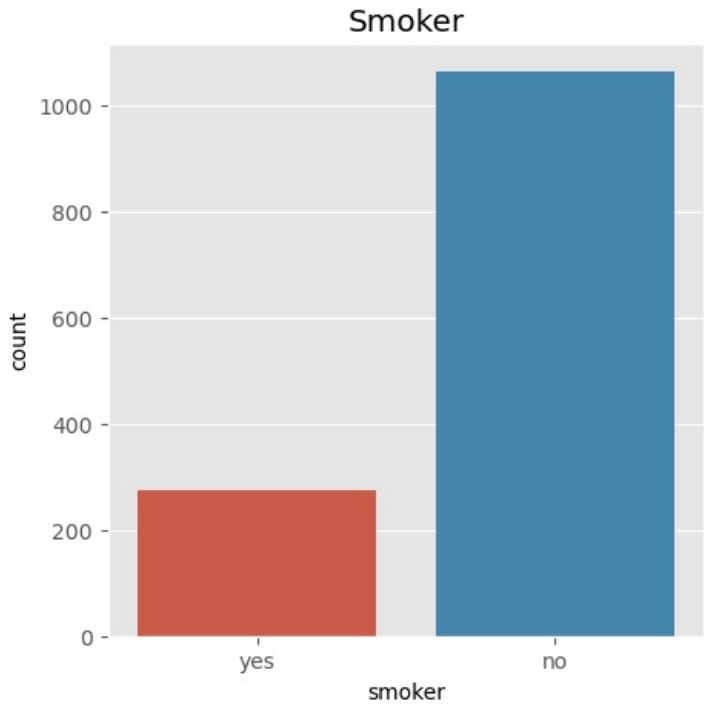
	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

In [11]:
```
# Data Analysis
```

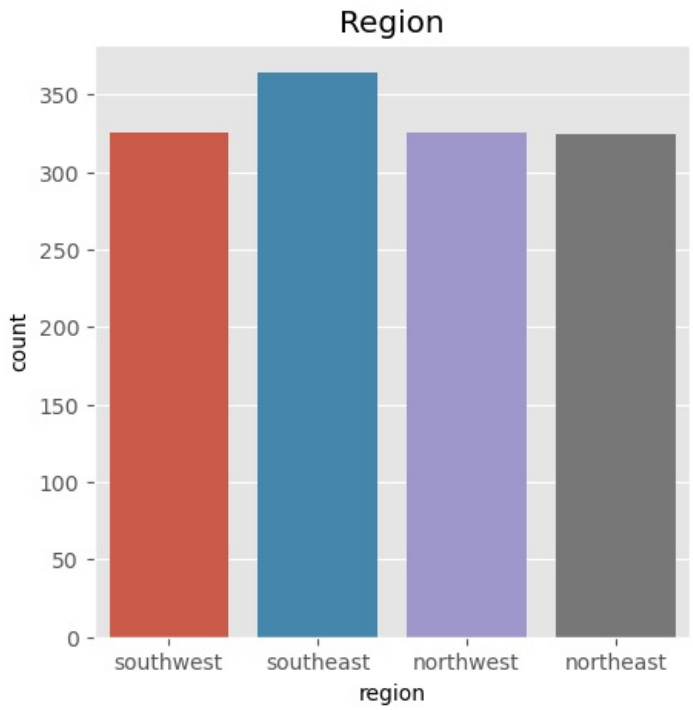
In [12]:
```
plt.figure(figsize=(5,5))
style.use('ggplot')
sns.countplot(x='sex', data=df)
plt.title('Gender Distribution')
plt.show()
```



In [13]:
```
plt.figure(figsize=(5,5))
sns.countplot(x='smoker', data=df)
plt.title('Smoker')
plt.show()
```

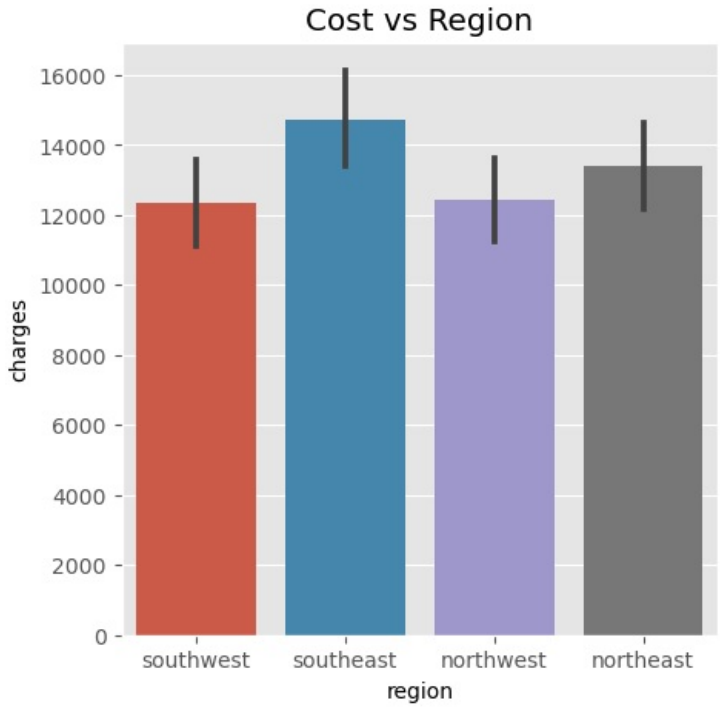


In [14]:
```
plt.figure(figsize=(5,5))
sns.countplot(x='region', data=df)
plt.title('Region')
plt.show()
```



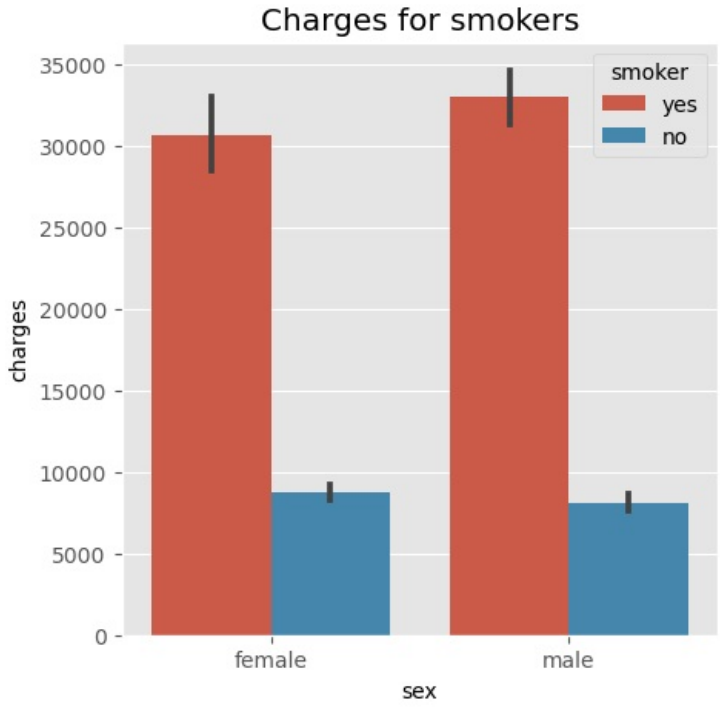
In [15]:
```
plt.figure(figsize=(5,5))
sns.barplot(x='region', y='charges', data=df)
plt.title('Cost vs Region')
```

Out[15]:
```
Text(0.5, 1.0, 'Cost vs Region')
```

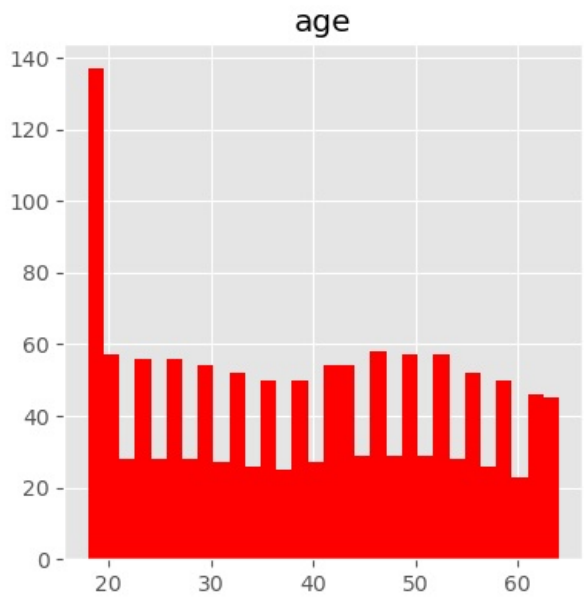


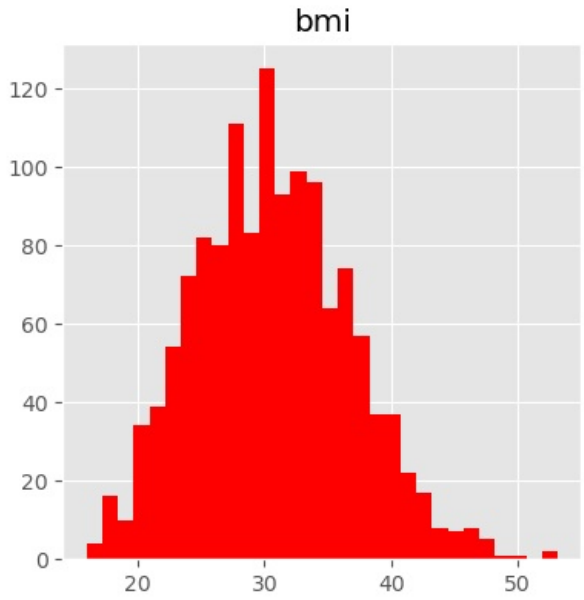
In [16]:
```
plt.figure(figsize=(5,5))
sns.barplot(x='sex', y='charges',hue='smoker', data=df)
plt.title('Charges for smokers')
```

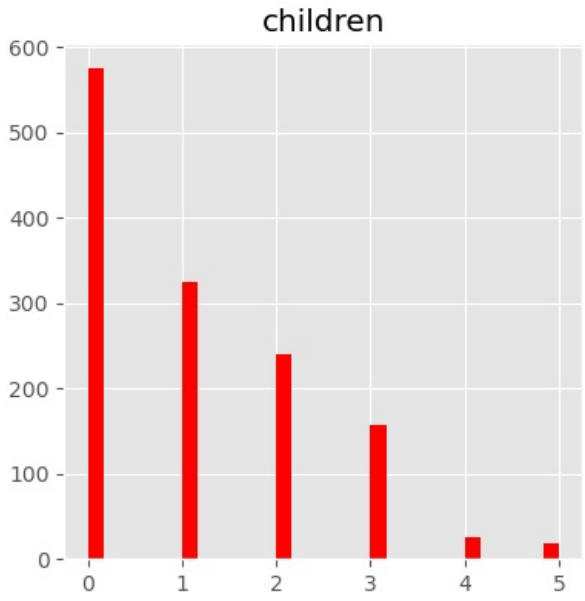
Out[16]:
```
Text(0.5, 1.0, 'Charges for smokers')
```

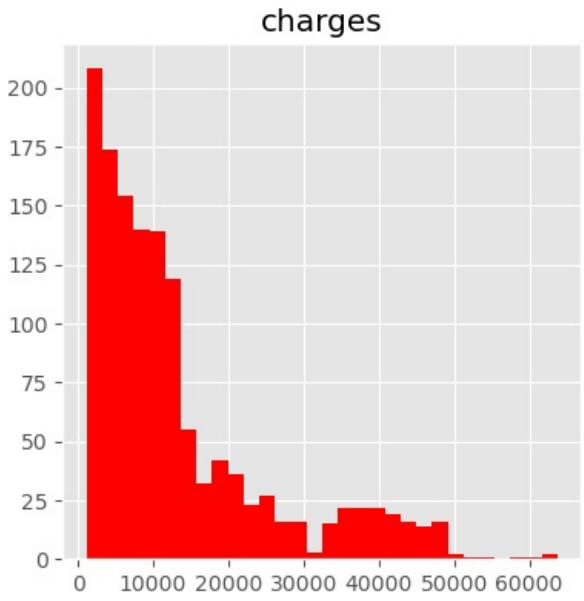


In [17]:
```
df[['age','bmi','children','charges']].hist(bins=30, figsize=(10,10), color='red')
plt.show()
```









In [18]:
```
df.head()
```

Out[18]:

	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

Data Pre-Processing

Encoding the categorical features

In [19]:
```
# encoding sex columns
df['sex'] = df['sex'].apply({'male':0, 'female':1}.get)

# encoding smoker columns
df['smoker'] = df['smoker'].apply({'yes':1, 'no':0}.get)

# ecoding region columns
df['region'] = df['region'].apply({'southwest':1, 'southeast':2, 'northwest':3, 'northeast':4}.get)
```

In [20]:
```
df.head()
```

Out[20]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	1	16884.92400
1	18	0	33.770	1	0	2	1725.55230
2	28	0	33.000	3	0	2	4449.46200
3	33	0	22.705	0	0	3	21984.47061
4	32	0	28.880	0	0	3	3866.85520

split dataset

In [21]:
```
X = df.drop(['charges', 'sex'], axis=1)
y = df.charges
```

In [22]:
```
X
```

Out[22]:

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

1338 rows × 5 columns

In [23]:
```
y
```

Out[23]:
```
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 [24]:
```
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=50)
print("X_train shape: ", X_train.shape)
print("X_test shape: ", X_test.shape)
print("y_train shpae: ", y_train.shape)
print("y_test shape: ", y_test.shape)
```

```
X_train shape:  (1070, 5)
X_test shape:  (268, 5)
y_train shpae:  (1070,)
y_test shape:  (268,)
```

Linear Regression Algorithm

In [25]:
```
from sklearn.linear_model import LinearRegression
```

In [26]:
```
lr= LinearRegression()
```

In [27]:
```
lr.fit(X_train, y_train)
```

Out[27]:
```
▾ LinearRegression
LinearRegression()
```

In [28]:
```
lr.fit(X_train, y_train)
```

```
pred = lr.predict(X_test)
```

In [29]:
```
lr.coef_
```

Out[29]:
```
array([  252.85956498,   322.04036635,   440.82269067, 23491.60815901,
         254.95586578])
```

In [30]:
```
lr.intercept_
```

Out[30]:
```
-12567.56397479529
```

In [31]:
```
y_pred = lr.predict(X_test)
y_pred
```

Out[31]:
```
array([ 7875.25276013, 11631.30739024, 17161.55394556, 10883.74870025,
         462.10558552, 39143.21874088, 26093.15735159,  7828.21055711,
        1100.40524204, 11612.99514527,  9657.82805958, 31942.2238696 ,
        7770.21929688, 31261.44404138,  7434.28896779,  9014.22114929,
       11641.7219218 ,  4828.88972902, 11903.03899027,  8357.07480459,
       -1775.95785327, 36979.88205044,  7114.39733105,  8977.11691202,
        3824.1576691 , 11007.38562435,  5740.6668405 , 38026.37349202,
         444.62396048,   438.03351526, 37185.18213552,  3906.19155673,
       14576.86300467,  1183.50762221,  1601.26523811,  5779.77514172,
        9096.96864333,  1366.261262  , 11415.69407865, 38687.79928222,
        9667.09266505,  6833.68141748, 16585.24906062, 13110.48911799,
       35687.98246635, 13536.38776849, 34516.18283992,  4140.03943145,
       12571.35462214,  9357.04585416, 39855.33444486,  9595.23543637,
        6148.10940097, 37816.30521294, 11664.948911  , 35213.17725974,
       10468.663892  ,  4357.67709918, 11890.38252215, 13784.00705213,
       13898.47740043,  7414.96427875,  3191.80418471,  1965.62895465,
        9297.40597488, 10819.21807103,  6233.90231392, 13744.64686668,
        5227.79382878, 14650.41592651, 13858.42813485,  5831.61724864,
        4116.22686717,  9701.59424859, 12225.58853542, 25097.07415246,
        8312.42682556,  9784.57816879, 33648.633584  ,  4821.52515047,
        3902.99114524, 27648.64084948,  1034.13373014, 13980.37071838,
        2504.03822021,  4297.002228  , 14082.81951482,  4878.95108959,
        3087.16279288,  3152.03388529,  3912.94174319, 11060.40961767,
        4070.31330113, 28687.85450078, 33151.11510109, 11195.34875004,
        8844.60053202, 10071.20489812, 24903.84993265,  3805.17170819,
        6335.17539926,   692.59504257, 36679.26494097, 13035.35360826,
       28365.56045949,  9725.89556136, 32313.54384143, 10722.31975568,
       10979.35019502,  8111.83894166,  7720.12940792, 10224.48026975,
        4581.35048814,  7254.82950524,  2852.85016396,  4716.32523868,
        7362.60218977, 11690.40058798, 16587.25586798, 33335.50693912,
        3012.89814921,  7472.46852556,  3333.69865791,  -420.69016004,
       12437.56585398, 10540.97150762,  7808.97536152, 11211.59187002,
       10414.51796901,  5356.34993736, 39783.42160584, 13223.34208083,
       37575.77383557, 16104.91427957, 29564.17782296, 34247.05480808,
       29690.95327904,  7282.17623691,  9131.61759771,  5138.53684674,
       13211.98344162,  4115.5147072 ,  5074.33175464,  6137.00708622,
        8178.19767996, 11417.55568836,  8250.25615475,  9165.05695789,
        6969.97071373,  6531.39304727, 34233.97156484,  6752.59255197,
        3812.73797097, 30422.15989543,  5969.7726887 , 11439.87018596,
        2520.29214347, 14923.22911622, 29970.23580447, 14196.88298631,
       11642.55572324, 11195.60333944,  9014.787291  ,  9732.74377749,
        3217.64701865,  5581.54986575,  8818.57191004,  2830.36521571,
       30875.28725577,  2589.78541163, 34816.63311838,  2663.71132718,
       11013.57184228,  8630.18282983, 32844.65132241,  3733.14446697,
       34835.78055559, 10006.53051771, 29552.84761828, 12193.10364804,
       11223.67491893, 15219.56639771,  9114.04421217,  9244.47774418,
        7150.31240442, 36677.05288539, 35697.27015164,  9442.24091547,
       32623.45408477,  8933.96338004,  6630.21721276, 31624.75006866,
        8763.05501041, 12949.4025212 , 10746.64641534, 12441.67885094,
       33043.99534654,  5740.43759798,  2822.25632915, 31212.074028  ,
       26160.89531414, 10256.34875978, 10647.24347593, 15183.87883179,
        9126.37731636, 27003.0425899 ,  9841.36572152, 12681.19696364,
       36987.01873907, 15450.77119624, 28205.51382683, 12996.28280978,
       11613.91935462, 38925.58598974, 17281.60147441,  4243.75207982,
       13393.93398156,  9332.69384762, 30461.83708177,   619.76561596,
        1478.22389859, 31919.73800689,  6957.43636343,  5381.67505628,
        5389.78394283, 12037.832925  , 10035.59602237, 11075.0950585 ,
        4775.83629284, 11109.85976766, 30595.08814275,  6200.08716098,
       10401.37004722,  6032.97116778, 31732.08734798, 14611.19095604,
       12326.17027357,  8574.6035187 , 13145.91355829, 28309.02658168,
       17279.32565475,  7263.03506887,  4843.75550933, 15030.7074972 ,
        8165.80578391, 11325.37087114, 26713.6806414 ,  7508.53096976,
        2064.66290247, 11416.89913874, 12859.23348567,  5589.33548224,
        8288.52612043,  9995.14879868, 12894.2590534 , 12877.54755957,
       15524.83512579, 10383.92051456,  4127.81528088,  9391.25843321,
        7284.94215766, 15739.68785072, 12542.34699488,  4451.26913707,
        4007.8364327 , 11002.53102457, 15863.67739388, 29880.38058832])
```

In [32]:
```
import numpy as np
```

In [33]:
```
pd.DataFrame(np.c_[X_test, y_test, y_pred], columns = ['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'ch
```

Out[33]:

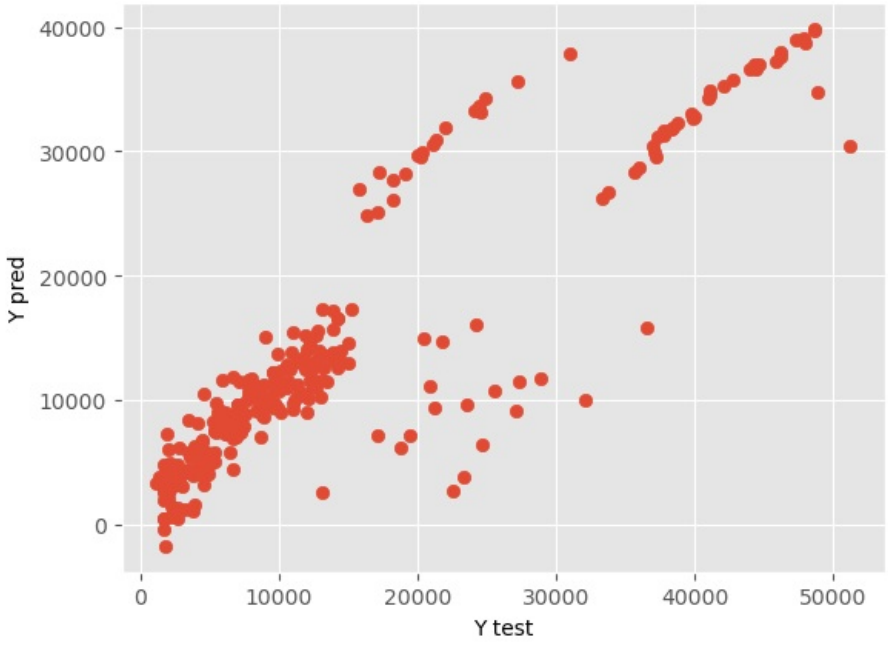
	age	sex	bmi	children	smoker	region	charges
0	38.0	30.69	1.0	0.0	2.0	5976.83110	7875.252760
1	35.0	43.34	2.0	0.0	2.0	5846.91760	11631.307390
2	64.0	40.48	0.0	0.0	2.0	13831.11520	17161.553946
3	52.0	31.20	0.0	0.0	1.0	9625.92000	10883.748700
4	26.0	17.67	0.0	0.0	3.0	2680.94930	462.105586
...	...	...	...	...	...	...	...
263	26.0	29.48	1.0	0.0	2.0	3392.36520	4451.269137
264	31.0	23.60	2.0	0.0	1.0	4931.64700	4007.836433
265	52.0	30.20	1.0	0.0	1.0	9724.53000	11002.531025
266	61.0	33.33	4.0	0.0	2.0	36580.28216	15863.677394
267	27.0	36.08	0.0	1.0	2.0	37133.89820	29880.380588

268 rows × 7 columns

In [34]:
```
lr.score(X_train, y_train)
```

Out[34]:
```
0.7399096306456938
```

In [35]:
```
plt.scatter(y_test, pred)
plt.xlabel('Y test')
plt.ylabel('Y pred')
plt.show()
```



Support Vector Regression

Feactur Salling

In [36]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(X_train)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

In [37]:
```
from sklearn.svm import SVR
```

In [38]:
```
svr_rbf=SVR(kernel='rbf',epsilon=1)
svr_rbf.fit(X_train, y_train)
svr_rbf.score(X_test, y_test)
```

Out[38]:
```
-0.13342477140009046
```

In [39]:
```
svr_linear = SVR (kernel='linear')
svr_linear.fit(X_train, y_train)
svr_linear.score(X_test, y_test)
```

Out[39]:
```
-0.048683760196334136
```

In [40]:
```
svr_poly = SVR (kernel='poly',degree=2,)
svr_poly.fit(X_train, y_train)
svr_poly.fit(X_test, y_test)
```

Out[40]:
```
SVR
SVR(degree=2, kernel='poly')
```

In [41]:
```
svr_linear.predict([X_test[0]])
```

Out[41]:
```
array([8798.24026248])
```

In [42]:
```
y_pred = svr_linear.predict(X_test)
y_pred
```

Out[42]:
```
array([ 8798.24026248,  8849.42767089,  9854.65448083,  9244.0103705 ,
        8230.07035333, 11123.16982049,  9417.73813975,  9005.67233468,
        8360.14657268,  9546.48776608,  9054.20063686, 10433.20153851,
        8835.03406874, 10121.36357521,  8903.88035336,  9354.16992688,
        9627.74714938,  8540.46379121,  9506.82256337,  8404.63251531,
        7849.19864176, 10834.55089202,  8515.19507289,  8853.35165901,
        8369.37742729,  9437.19418073,  8759.65778528, 10639.47584966,
        7960.66734568,  8039.38320731, 10461.30483279,  8133.82867561,
        9802.32493084,  8094.67787295,  8227.89421289,  8242.95842336,
        9236.3235371 ,  8069.89128624,  9560.58604044, 11109.24746875,
        8906.15961515,  8864.86781842,  9737.49478362,  9636.14931824,
       10931.89295725,  9658.62633499, 10495.41238089,  8099.04717219,
        9727.00657679,  9075.19294856, 10986.76974983,  8983.48469052,
        8453.50606814, 11081.97518988,  9149.29914684, 10459.30164165,
        9410.38059819,  8805.10194031,  8809.47755987,  9447.42240119,
        9721.35370749,  8883.53149805,  8158.45031199,  8068.69781865,
        9003.87166981,  9122.85406707,  8614.82039939,  9326.35554799,
        8583.21666694,  9574.59682053,  9668.96632371,  8582.8756898 ,
        8218.7951182 ,  8836.4421519 ,  9317.04807149,  9304.70969975,
        8842.93132747,  9248.89396988, 10666.98338042,  8080.76194573,
        8280.65019506,  9675.91426521,  8089.17238759,  9643.64144117,
        8165.61717264,  8253.83959013,  9547.57020874,  8423.68691607,
        8304.13842764,  8545.81817623,  8430.717739  ,  9012.20930609,
        8445.9871672 ,  9630.35824654, 10512.18249873,  9433.71745187,
        8870.94423314,  9339.13407644,  9298.50570652,  8467.4283245 ,
        8550.0752379 ,  7968.62913699, 10884.08854967,  9683.41952487,
       10108.71568183,  9127.20671736, 10115.67806807,  9378.36336587,
        9306.98378401,  8765.66567247,  8743.57696392,  9232.35919501,
        8173.11589444,  8308.63790477,  8097.18448756,  8457.92213375,
        8662.69119561,  9608.87230109,  9885.89770494, 10678.07682858,
        8062.85721168,  8645.78094517,  8032.99119786,  7992.07850226,
        9474.3070198 ,  8714.34938965,  8856.72089312,  9294.00272928,
        9099.63418924,  8259.31280705, 11053.87181234,  9479.80813628,
       10544.67108114,  9641.0305047 , 10026.69735295, 10797.19340321,
       10021.25602624,  8878.56319125,  9067.95495646,  8589.86727149,
        9627.77727892,  8507.33770334,  8499.37116921,  8503.54166312,
        8888.30881532,  9052.21186039,  8720.43194277,  9069.73804588,
        8958.38780876,  8476.03366758, 10516.29618048,  8562.76813268,
        8079.02646139,  9765.67047948,  8207.47841207,  9341.49842386,
        8106.93996666,  9514.65464158, 10169.75462702,  9609.71754041,
        9069.65268622,  9333.66021157,  8795.36673143,  9356.80225511,
        8338.27759192,  8555.11356342,  8889.13675774,  8126.41215721,
       10278.44138901,  8009.10582393, 10415.20568464,  8131.98728737,
        9537.4582208 ,  8922.54477632, 10193.33937777,  8105.71572158,
       10694.88340511,  8967.4456187 ,  9607.74380079,  9227.56578895,
        9344.77346516,  9573.84735604,  8907.42607906,  9341.83456282,
        8575.53801875, 10804.38527689, 10563.98881981,  9178.70785561,
       10126.33336866,  9231.79923341,  8558.83893697, 10073.12464561,
        9145.97007594,  9539.71879441,  9468.99327159,  9334.90760909,
       10208.54688161,  8659.58036584,  8096.20218864, 10189.18943692,
        9438.23187047,  9523.36208106,  9086.67331364,  9611.45813697,
        8818.68266266,  9934.94864127,  9260.22909016,  9541.31925838,
       10825.26359035,  9432.23219425,  9774.13123   ,  9712.11001595,
        9248.43090037, 10827.60738471,  9718.97746605,  8211.95901027,
        9613.89091655,  9184.69770257,  9856.08847321,  8006.45694212,
        8072.78137087, 10073.78902446,  9137.68302235,  8577.94550698,
        8608.15547555,  9271.55360779,  9167.10475451,  9249.45393149,
        8368.57450219,  9140.98828856, 10100.66853085,  8225.09451687,
        8729.60012968,  8379.69807989, 10007.15998187,  9642.75306103,
        9301.25392342,  9109.96971369,  9637.286717  ,  9608.68313754,
        9938.77754015,  8640.46607541,  8151.59151639,  9308.45825581,
        8558.46057917,  9616.88504901,  9475.71358624,  8607.47277363,
        8122.26035148,  9350.97745839,  9556.55205356,  8574.39178684,
        8722.37013203,  9335.27787706,  9389.60977495,  9636.77232577,
        9781.66997073,  9447.83516203,  8188.50815834,  9226.75356139,
        8700.36390091,  9779.05196179,  9557.29890983,  8328.90855616,
        8435.17415857,  9268.26201292,  9804.88500608,  9747.57038701])
```

In [43]:
```
y_test
```

Out[43]:
```
589      5976.83110
383      5846.91760
534     13831.11520
284      9625.92000
821      2680.94930
           ...
871      3392.36520
496      4931.64700
578      9724.53000
1012    36580.28216
203     37133.89820
Name: charges, Length: 268, dtype: float64
```

MSE & RMSE

In [44]:
```
from sklearn.metrics import mean_squared_error
import numpy as np

mse = mean_squared_error(y_test, y_pred)
rsme=np.sqrt(mse)
print('MSE=',mse)
print('RMSE=',rsme)
```

```
MSE= 168095236.72704417
RMSE= 12965.154712807871
```

In [45]:
```
data = {'age':50, 'bmi':25, 'children':2, 'smoker':1, 'region':2}
index = [0]
cust_df = pd.DataFrame(data, index)
cust_df
```

Out[45]:

	age	bmi	children	smoker	region
0	50	25	2	1	2

In [46]:
```
cost_pred = lr.predict(cust_df)
print("The medical insurance cost of the new customer is: ", cost_pred)
```

```
The medical insurance cost of the new customer is:  [33009.58870502]
```

Decision Tree regression - ML Model training

In [47]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [48]:
```
lr = DecisionTreeRegressor(criterion='squared_error')
lr.fit(X_train, y_train)
```

Out[48]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

Check Score of Model

In [49]:
```
lr.score(X_test, y_test)
```

Out[49]:
```
0.7584259775075534
```

In [50]:
```
pred = lr.predict(X_test)
pred
```

Out[50]:
```
array([ 5974.3847  ,  5124.1887  , 14319.031   ,  9140.951   ,
       21595.38229 , 48517.56315 , 17085.2676  ,  7518.02535 ,
       21595.38229 , 11073.176   ,  6358.77645 , 23807.2406  ,
        6775.961   , 35491.64    ,  7045.499   , 22192.43711 ,
       12913.9924  ,  5031.26955 , 12925.886   ,  2904.088   ,
        1728.897   , 46255.1125  ,  4466.6214  ,  5028.1466  ,
        3847.674   , 11082.5772  ,  5125.2157  , 43753.33705 ,
        1719.4363  ,  1704.5681  , 45702.02235 ,  2913.569   ,
       16455.70785 ,  3500.6123  ,  4005.4225  ,  3597.596   ,
        9140.951   ,  2709.1119  , 24513.09126 , 52590.82939 ,
        6238.298   ,  6455.86265 , 15170.069   , 12925.886   ,
       29523.1656  , 13393.756   , 39125.33225 ,  2020.5523  ,
       14254.6082  ,  8703.456   , 48970.2476  ,  7209.4918  ,
        3659.346   , 27808.7251  , 12797.20962 , 42211.1382  ,
       11345.519   ,  6877.9801  , 19496.71917 , 11436.73815 ,
       13607.36875 ,  6455.86265 , 18955.22017 ,  2200.83085 ,
        7749.1564  ,  8252.2843  ,  4347.02335 ,  9391.346   ,
        4402.233   , 12648.7034  , 13393.756   ,  4415.1588  ,
        6113.23105 ,  6474.013   ,  8765.249   , 16884.924   ,
        6593.5083  ,  9225.2564  , 26109.32905 , 14133.03775 ,
        2690.1138  , 19933.458   ,  2302.3     , 13919.8229  ,
        2457.21115 ,  4561.1885  , 12629.1656  ,  6799.458   ,
        3044.2133  ,  4134.08245 ,  2867.1196  ,  8162.71625 ,
        3645.0894  , 36397.576   , 23401.30575 , 10436.096   ,
       15828.82173 , 12105.32    , 16884.924   , 23241.47453 ,
        4915.05985 ,  1719.4363  , 43254.41795 , 12913.9924  ,
       19444.2658  ,  8569.8618  , 40103.89    , 19749.38338 ,
       10600.5483  ,  5910.944   ,  5974.3847  ,  9288.0267  ,
       10795.93733 ,  3994.1778  ,  1639.5631  ,  4340.4409  ,
        4922.9159  , 12142.5786  , 14692.66935 , 35069.37452 ,
       16586.49771 ,  4738.2682  ,  1135.9407  ,  1627.28245 ,
       12629.1656  ,  5124.1887  ,  7151.092   , 25333.33284 ,
        8551.347   ,  3597.596   , 48970.2476  , 26467.09737 ,
       45702.02235 , 11365.952   , 20149.3229  , 24873.3849  ,
       20984.0936  ,  6067.12675 ,  8703.456   ,  5267.81815 ,
       12730.9996  ,  5002.7827  ,  4661.28635 ,  6113.23105 ,
        6548.19505 ,  7160.3303  ,  3935.1799  ,  8232.6388  ,
        7358.17565 ,  4449.462   , 39727.614   ,  4922.9159  ,
        1964.78    , 35585.576   ,  1633.0444  ,  9487.6442  ,
       10795.93733 , 11396.9002  , 20984.0936  , 12644.589   ,
        7160.3303  ,  9290.1395  ,  3935.1799  , 11345.519   ,
        3213.62205 , 11737.84884 ,  7537.1639  , 11774.159275,
       21774.32215 ,  1621.8827  , 42560.4304  ,  2639.0429  ,
       12523.6048  ,  6849.026   , 38746.3551  ,  1135.9407  ,
       41919.097   ,  8059.6791  , 36307.7983  , 10381.4787  ,
       11286.5387  , 11881.358   ,  6186.127   , 27322.73386 ,
        5428.7277  , 47055.5321  , 42969.8527  ,  9101.798   ,
       40103.89    ,  9566.9909  ,  4922.9159  , 40103.89    ,
        8444.474   , 10923.9332  , 12044.342   , 11163.568   ,
       39871.7043  ,  5267.81815 ,  1639.5631  , 39597.4072  ,
       17352.6803  , 24513.09126 ,  6849.026   , 11365.952   ,
        4518.82625 , 19023.26    ,  9617.66245 , 12644.589   ,
       60021.39897 , 11552.904   , 18328.2381  , 16455.70785 ,
       10118.424   , 47462.894   , 12574.049   ,  1815.8759  ,
       13019.16105 ,  9282.4806  , 44585.45587 ,  2597.779   ,
        1708.92575 , 40003.33225 ,  8428.0693  ,  5031.26955 ,
        5031.26955 ,  9391.346   ,  8219.2039  ,  7935.29115 ,
        2897.3235  ,  7935.29115 , 19719.6947  ,  3292.52985 ,
        5124.1887  ,  3393.35635 , 55135.40209 , 11743.9341  ,
        9620.3307  ,  8444.474   , 12925.886   , 34254.05335 ,
       15170.069   , 15828.82173 ,  1532.4697  ,  9414.92    ,
        4949.7587  , 12815.44495 , 33750.2918  , 14358.36437 ,
       18955.22017 , 10807.4863  , 12231.6136  ,  4137.5227  ,
        6334.34355 , 11454.0215  , 10381.4787  , 12430.95335 ,
       13228.84695 , 25992.82104 , 26018.95052 ,  9095.06825 ,
        5934.3798  , 13887.204   , 12925.886   ,  2902.9065  ,
       17626.23951 ,  9748.9106  , 13919.8229  , 36219.40545 ])
```

In [51]:
```
y_test
```

Out[51]:
```
589      5976.83110
383      5846.91760
534     13831.11520
284      9625.92000
821      2680.94930
           ...
871      3392.36520
496      4931.64700
578      9724.53000
1012    36580.28216
203     37133.89820
Name: charges, Length: 268, dtype: float64
```

Random forest regression - Ml model training

n_estimators value increase

In [52]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [53]:
```
lr = RandomForestRegressor(n_estimators=40,criterion='squared_error')
lr.fit(X_train,y_train)
```

Out[53]:
```
RandomForestRegressor(n_estimators=40)
```

In [54]:
```
lr.score(X_train,y_train)
```

Out[54]:
```
0.9743421918500932
```

In [55]:
```
y_pred = lr.predict(X_test)
y_pred
```

Out[55]:
```
array([ 5980.82695125,  5430.111865  , 14262.93354125, 10508.59988375,
        9816.83427125, 47729.87825875, 17274.106835  ,  8000.4168915 ,
        8564.44802075, 12358.657274  ,  7591.814537  , 22894.1842275 ,
        6818.36036   , 39360.35969875,  6952.90702125, 17262.04385925,
       19183.836219  ,  4452.2705105 , 17974.38237025,  4400.35951875,
        2596.3746895 , 47114.6656735 ,  8541.39757025,  9102.352244  ,
        6692.0857685 , 12369.05355   , 10724.79924325, 44171.3436825 ,
        2861.46046425,  1752.90161125, 45926.55839925,  7764.05217975,
       15840.7986665 ,  6760.280358  ,  4250.91481575,  6224.775346  ,
       11582.61018675,  3584.18429075, 20670.73325525, 48420.26270725,
        6579.74268   ,  6983.0357275 , 19149.6372035 , 15979.8209205 ,
       28882.52321025, 13529.47784125, 40232.7630775 ,  2039.5838665 ,
       14859.6362345 ,  8782.829816  , 47910.2430525 ,  7041.13853875,
        7064.80438825, 27041.9037475 , 10307.16452875, 41403.3092725 ,
       12103.93680425,  6931.36411125,  8324.7409935 , 18312.5580065 ,
       13878.10493425,  7111.3226985 ,  8661.116307  ,  1691.4381625 ,
        7934.9756535 ,  8067.25075725,  4513.4208365 ,  9428.5982425 ,
        5921.682392  , 12682.876543  , 13579.47689   ,  4847.8958805 ,
        8703.2869435 ,  6403.0749365 , 10544.3813935 , 17253.20453375,
        7004.86266625, 10384.898536  , 25382.257753  ,  9757.546975  ,
        3071.7509325 , 18657.167495  , 13782.37611775, 17652.4130325 ,
        2213.65590625,  4261.981594  , 14961.9264635 ,  5433.64815175,
        3109.0571705 ,  5127.7777795 ,  3338.2388325 ,  8342.22229275,
        3669.791815  , 35583.59419   , 25858.79875375, 13259.754189  ,
       10799.00655125, 12817.708811  , 16986.8563575 ,  6003.81448025,
        8576.3938645 ,  1888.86461375, 46039.2031815 , 15126.0320515 ,
       19480.32275375,  8545.67518   , 40287.54939475, 12416.42280175,
       10372.6968625 ,  6137.821651  ,  6108.48981125,  9030.65418975,
        5787.24550425,  7185.4291655 ,  2124.45321525,  6652.64826825,
        5332.1922    , 12435.42300875, 14459.5138475 , 25089.96057   ,
        9357.52505925,  6346.2391565 ,  4893.51405   ,  1690.72717375,
       14281.06633725,  7125.125331  ,  6555.74675   ,  9288.45446425,
        8588.2753275 ,  7436.81527875, 48349.6979625 , 15835.71516725,
       44508.04668375, 11277.79477375, 20213.942085  , 25429.74652375,
       20345.3750825 ,  6356.4289325 ,  8864.302416  ,  7124.09074625,
       12307.5653725 ,  6153.12804125,  5489.879567  ,  7579.74185   ,
        6915.47305775,  7229.69474625,  5301.00651   , 11551.1905545 ,
        8283.01634475,  5668.87185575, 41332.89976725,  6662.1493575 ,
       11523.44169954, 38445.7784345 ,  1759.39913   , 10219.37601325,
        7097.84092125, 15387.64569625, 20608.76289125, 12241.64487   ,
        9176.91061875, 10031.47244   ,  5884.79529375, 11938.82660075,
        3008.2780425 ,  9145.796837  ,  6598.47658375,  2310.97098204,
       26629.1776015 ,  1603.79598375, 43118.67044625, 11799.1862315 ,
       16266.48124575,  6963.85982125, 40643.22933625,  3449.35647975,
       41645.44562925,  7640.26043425, 37808.805105  ,  9936.8624425 ,
       14292.68920525, 14252.3704125 ,  6136.29633625, 13707.432518  ,
        5747.743924  , 45882.5088295 , 42042.487605  ,  9568.4551835 ,
       40247.65733675, 10070.72161975,  6313.2392715 , 39360.193457  ,
       10897.800178  , 11950.7496855 , 11672.27391775, 12250.125316  ,
       43496.28323675,  5616.04971025,  1742.59455375, 39415.43466175,
       20170.83820375, 19260.2326465 ,  8731.20761875, 12119.25809575,
        5913.75450875, 19578.17056   , 11507.43083925, 12888.9412315 ,
       47287.59634775, 10923.64057175, 18404.81501125, 20030.155892  ,
       10595.86612525, 49986.1851075 , 12711.9177525 ,  4060.09868875,
       13043.90902825, 12408.66912325, 40133.930704  ,  2751.421462  ,
        1757.49892625, 39886.7527025 ,  8690.816895  ,  6030.878992  ,
        4983.5391865 ,  9494.3714    ,  9590.73389625,  9222.069817  ,
        4255.74721975,  9083.32012225, 20599.32556   ,  8518.98555925,
        6398.09120125,  8169.198694  , 43139.487715  , 12005.716169  ,
        9711.29985375,  9887.26800575, 15979.8209205 , 34831.30279375,
       19662.10523925,  9001.62172   ,  1795.15440325,  9069.86598125,
        6025.543206  , 12738.5201825 , 34680.07166275,  8130.55646425,
        9212.87606575, 12636.66014325, 11447.403895  ,  4158.13414875,
        5380.97766625, 11055.33282125, 11967.41495325, 12864.54602175,
       14773.1906885 , 15061.84107875, 10231.27924075,  9693.552276  ,
        7466.8915625 , 13540.52334625, 19220.56734225,  3221.31741625,
        7365.4564885 , 11208.8530925 , 15747.18165375, 37592.96437975])
```

In [56]:
```
y_test
```

Out[56]:
```
589      5976.83110
383      5846.91760
534     13831.11520
284      9625.92000
821      2680.94930
           ...
871      3392.36520
496      4931.64700
578      9724.53000
1012    36580.28216
203     37133.89820
Name: charges, Length: 268, dtype: float64
```

Implementing Ridge and Lasso Regression

In [57]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [58]:
```
rd = Ridge()
rd.fit(X_train,y_train)
rd.score(X_test,y_test)
```

Out[58]:
```
0.7864389767895823
```

In [59]:
```
ls = Lasso()
ls.fit(X_train,y_train)
ls.score(X_test,y_test)
```

Out[59]:
```
0.7865095570626367
```

In [60]:
```
rd = Ridge(alpha=2)
rd.fit(X_train,y_train)
rd.score(X_test,y_test)
```

Out[60]:
```
0.7863416031259258
```

In [61]:
```
ls = Lasso(alpha=2)
ls.fit(X_train,y_train)
ls.score(X_test,y_test)
```

Out[61]:
```
0.7864837496865126
```

k fold cross validation

In [64]:
```
from sklearn.model_selection import cross_val_score, KFold

# Create a k-fold cross-validator
kf = KFold(n_splits=5, shuffle=True, random_state=50)

# Perform k-fold cross-validation for linear SVM
scores = cross_val_score(lr, X_train, y_train, cv=kf)

print("Cross-validation scores:", scores)
kfold_mean_score = np.mean(scores)
print("Mean Accuracy:",kfold_mean_score)
```

```
Cross-validation scores: [0.7897392  0.77262679 0.84978669 0.84489666 0.86737382]
Mean Accuracy: 0.8248846336828374
```

In [ ]: