In [59]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import  confusion_matrix
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder, StandardScaler, OneHotEncoder
from sklearn.metrics import mean_absolute_error, mean_squared_error
import joblib
```

Task1

Linear Regression

Task 1: Use Linear_regresssion.csv file and apply linear regression to predict Y on basis of x Evaluate through R2 method and check score of your predicted data with respect to Actual data during This process remember all the working cycle of ML (Data, preprocessing & cleaning, visualization, model, test, deployment)

In [2]:
```
df = pd.read_csv("linear_reg.csv")
```

In [27]:
```
df.dropna(axis=0)
df.head(13)
```

Out[27]:

	X	Y
0	1	3.888889
1	2	4.555556
2	3	5.222222
3	4	5.888889
4	5	6.555556
5	6	7.222222
6	7	7.888889
7	8	8.555556
8	9	9.222222
9	10	9.888889
10	11	10.555556
11	12	11.222222
12	13	11.888889

In [4]:
```
x = np.array(df["X"]).reshape(-1, 1)
y=np.array(df["Y"]).reshape(-1, 1)
```

In [5]:
```
#split test and train data
x_train,x_test ,y_train ,y_test = train_test_split(x,y,test_size = 0.2 , random_state = 42)
x_train
print("\nx_train",x_train)
print("\nx_train",x_test)
print("\nx_train",y_train)
print("\nx_train",y_test)
```

```
x_train [[233]
 [ 60]
 [  7]
 [186]
 [174]
 [ 31]
 [ 23]
 [257]
 [ 57]
 [187]
 [109]
 [127]
 [231]
 [194]
 [130]
 [283]
 [ 83]
 [ 85]
```

In [6]:
```
#train model

model = LinearRegression()
model.fit(x_train,y_train)
```

Out[6]:
```
LinearRegression
LinearRegression()
```

In [7]:
```
# R2 method and check score of your predicted data with respect to Actual data

model.intercept_
```

Out[7]:
```
array([6.58544043])
```

In [8]:
```
#model score
model.score(x_test,y_test)
```

Out[8]:
```
0.9966137419987549
```

In [9]:
```
#model Coefcient
model.coef_
```

Out[9]:
```
array([[0.6331428]])
```

In [10]:
```
#show us the predicted value

y_predicted = model.predict(x_test)
result  =pd.DataFrame({'Actual':y_test.flatten(),'Predicted':y_predicted.flatten()})
result
```

Out[10]:

	Actual	Predicted
0	139.222222	135.746572
1	181.222222	175.634568
2	105.222222	103.456289
3	9.888889	12.916868
4	159.222222	154.740855
5	154.555556	150.308856
6	134.555556	131.314572
7	76.555556	76.231148
8	7.222222	10.384297
9	120.555556	118.018573
10	161.888889	157.273427
11	41.888889	43.307723
12	149.222222	145.243714
13	33.888889	35.710009
14	125.222222	122.450573
15	151.222222	147.143142
16	196.555556	190.196852
17	144.555556	140.811714
18	102.555556	100.923718
19	113.888889	111.687145
20	55.888889	56.603722
21	79.222222	78.763720
22	169.888889	164.871140
23	170.555556	165.504283
24	73.222222	73.065434
25	31.888889	33.810581

	Actual	Predicted
26	191.222222	185.131710
27	200.555556	193.995709
28	108.555556	106.622003
29	162.555556	157.906569
30	15.222222	17.982011
31	113.222222	111.054002
32	25.888889	28.112296
33	19.888889	22.414010
34	147.222222	143.344285
35	83.222222	82.562576
36	8.555556	11.650583
37	63.888889	64.201435
38	34.555556	36.343152
39	52.555556	53.438008
40	65.888889	66.100864
41	54.555556	55.337436
42	194.555556	188.297424
43	43.888889	45.207151
44	55.222222	55.970579
45	45.888889	47.106580
46	159.888889	155.373998
47	156.555556	152.208284
48	77.888889	77.497434
49	157.888889	153.474570
50	123.888889	121.184287
51	99.888889	98.391146
52	163.222222	158.539712

	Actual	Predicted
53	53.888889	54.704293
54	201.888889	195.261995
55	189.222222	183.232281
56	68.555556	68.633435
57	65.222222	65.467721
58	131.888889	128.782001
59	20.555556	23.047153

In [11]:
```
#Error Calculation

mae = mean_absolute_error(y_test,y_predicted)
mse = mean_squared_error(y_test,y_predicted)
rmse = np.sqrt(mse)
print(f'Mean absolute error: {mae}')
print(f'Mean squared error: {mse}')
print(f'Root mean squared error: {rmse}')
```

```
Mean absolute error: 2.82931966176093
Mean squared error: 11.554304857221412
Root mean squared error: 3.3991623758245813
```

In [12]:
```
Act_minus_predict = sum((y_test - y_predicted)**2)
Act_minus_ActMean = sum((y_test - y_test.mean())**2)
r2 = 1 - Act_minus_predict/Act_minus_ActMean

print("r2 :" ,r2)
```

```
r2 : [0.99661374]
```

In [13]:
```
joblib.dump(model,"LinearMode.pkl")
```

Out[13]:
```
['LinearMode.pkl']
```

In [14]:
```
# predicting the Value

model = joblib.load("LinearMode.pkl")
```

In [72]:
```
#def predict_species(val):
#    return model.predict(val)

##X = int(input("Enter Value of X to predict y:"))

#predictions = predict_species(x)
#print("Predicted species:", predictions)
```

In [ ]:

In [ ]:

Task 2

#Use insurance.csv and apply both Linear and Logistic regression and predict amount of Insurance on basis of input parameters. #Evaluate and compare the difference in both regression and score of predicted data with respect to Actual data during This #process remember all the working cycle of ML (Data, preprocessing & cleaning, visualization, model, test, deployment)

In [62]:
```
data = pd.read_csv("insurance.csv")
#cleaning data
data = data.loc[:, ~data.columns.duplicated()]
data.dropna(axis =1 ,how='all')
data.dropna(axis=0, how='all')
data.drop_duplicates()

data
```

Out[62]:

	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 [63]:
```
df = data.drop(['charges'],axis = 1)
x = df
y =data['charges']
numeric = ['age','bmi','children']
categoric= ["region",'smoker','sex']

categoric
```

Out[63]:
```
['region', 'smoker', 'sex']
```

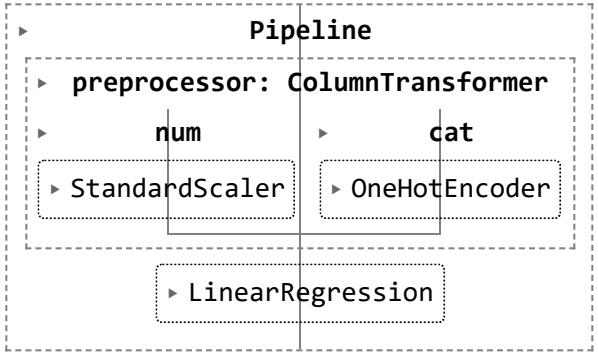
In [64]:
```
numeric_transfomer = Pipeline(steps=[('scalar', StandardScaler())])
catego_transfomer = Pipeline(steps=[('encoder', OneHotEncoder())])
```

In [65]:
```
preprocessor = ColumnTransformer(
transformers =[
    ('num',numeric_transfomer,numeric),
    ('cat',catego_transfomer,categoric)
])
```

In [70]:
```
pipeline = Pipeline(steps=[
('preprocessor',preprocessor),
('classifier',LinearRegression())])
```

In [71]:
```
pipeline.fit(x,y)
```

Out[71]:



In [73]:
```
joblib.dump(pipeline, 'linear.pkl')
```

Out[73]:
```
['linear.pkl']
```

In [ ]: