STANDARDSCALER

In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv("data.csv")
```

In [3]:

```
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 182 entries, 0 to 181
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Area                    182 non-null    int64
 1   Sensing Range           182 non-null    int64
 2   Transmission Range      182 non-null    int64
 3   Number of Sensor nodes  182 non-null    int64
 4   Number of Barriers      182 non-null    int64
dtypes: int64(5)
memory usage: 7.2 KB
```

Out[3]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes	Number of Barriers
count	182.000000	182.00000	182.000000	182.000000	182.000000
mean	24375.000000	27.50000	55.000000	250.000000	94.071429
std	15197.252769	7.52069	15.041379	90.248276	65.171006
min	5000.000000	15.00000	30.000000	100.000000	12.000000
25%	9375.000000	21.00000	42.000000	172.000000	42.000000
50%	21875.000000	27.50000	55.000000	250.000000	80.000000
75%	39375.000000	34.00000	68.000000	328.000000	128.750000
max	50000.000000	40.00000	80.000000	400.000000	320.000000

In [4]:

```
correlation_matrix = df.corr()
print(correlation_matrix)
```

```
                                Area  Sensing Range  Transmission Range  \
Area                    1.000000e+00   3.095077e-16        3.095077e-16
Sensing Range           3.095077e-16   1.000000e+00        1.000000e+00
Transmission Range      3.095077e-16   1.000000e+00        1.000000e+00
Number of Sensor nodes -1.162999e-16   1.000000e+00        1.000000e+00
Number of Barriers     -4.234383e-01   8.383655e-01        8.383655e-01

                        Number of Sensor nodes  Number of Barriers
Area                             -1.162999e-16           -0.423438
Sensing Range                     1.000000e+00            0.838365
Transmission Range                1.000000e+00            0.838365
Number of Sensor nodes            1.000000e+00            0.838365
Number of Barriers                8.383655e-01            1.000000
```

In [5]:

```
df=df.dropna()
```

In [6]:

```
df.head()
```

Out[6]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes	Number of Barriers
0	5000	15	30	100	30
1	5000	16	32	112	35
2	5000	17	34	124	42
3	5000	18	36	136	48
4	5000	19	38	148	56

In [7]:

```
df.tail()
```

Out[7]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes	Number of Barriers
177	50000	36	72	352	101
178	50000	37	74	364	107
179	50000	38	76	376	114
180	50000	39	78	388	121
181	50000	40	80	400	128

In [8]:

```
df.shape
```

Out[8]:

```
(182, 5)
```

In [9]:

```
df.dtypes
```

Out[9]:

```
Area                        int64
Sensing Range               int64
Transmission Range          int64
Number of Sensor nodes      int64
Number of Barriers          int64
dtype: object
```

In [10]:

```
X=df.iloc[:,:4]
y=df.iloc[:,4:]
```

In [11]:

```
X.head()
```

Out[11]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes
0	5000	15	30	100
1	5000	16	32	112
2	5000	17	34	124
3	5000	18	36	136
4	5000	19	38	148

In [12]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [13]:

```
X_train, X_test, y_train, y_test = train_test_split (X, y, random_state = 23)
```

In [14]:

```
X_train.head()
```

Out[14]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes
24	5000	39	78	388
171	50000	30	60	280
55	15000	18	36	136
164	50000	23	46	196
45	9375	34	68	328

In [15]:

```
scaler = StandardScaler().fit(X_train)
```

In [16]:

```
print(scaler)
```

```
StandardScaler()
```

In [17]:

```
scaler.mean_
```

Out[17]:

```
array([24972.42647059,    27.41911765,    54.83823529,   249.02941176])
```

In [18]:

```
scaler.scale_
```

Out[18]:

```
array([1.55459152e+04, 7.46812358e+00, 1.49362472e+01, 8.96174830e+01])
```

In [19]:

```
scaler.transform(X_train)
```

Out[19]:

```
array([[-1.2847379 ,  1.55070845,  1.55070845,  1.55070845],
       [ 1.60991316,  0.34558645,  0.34558645,  0.34558645],
       [-0.64148211, -1.26124287, -1.26124287, -1.26124287],
       [ 1.60991316, -0.59173065, -0.59173065, -0.59173065],
       [-1.00331349,  0.88119623,  0.88119623,  0.88119623],
       [-0.64148211,  1.55070845,  1.55070845,  1.55070845],
       [ 0.32340158,  1.55070845,  1.55070845,  1.55070845],
       [ 0.32340158, -1.39514532, -1.39514532, -1.39514532],
       [-1.2847379 ,  1.01509868,  1.01509868,  1.01509868],
       [ 0.92645388,  1.55070845,  1.55070845,  1.55070845],
       [ 0.32340158, -0.85953554, -0.85953554, -0.85953554],
       [ 0.32340158, -1.12734043, -1.12734043, -1.12734043],
       [-1.2847379 ,  0.21168401,  0.21168401,  0.21168401],
       [ 0.32340158,  0.07778157,  0.07778157,  0.07778157],
       [ 0.92645388,  1.28290356,  1.28290356,  1.28290356],
       [ 1.60991316,  1.55070845,  1.55070845,  1.55070845],
       [ 0.92645388, -1.12734043, -1.12734043, -1.12734043],
       [ 0.32340158, -1.66295021, -1.66295021, -1.66295021],
```

In [20]:

```
X_train_scaled = scaler.transform(X_train)
```

In [21]:

```
print(X_train_scaled)
```

```
[[-1.2847379   1.55070845  1.55070845  1.55070845]
 [ 1.60991316  0.34558645  0.34558645  0.34558645]
 [-0.64148211 -1.26124287 -1.26124287 -1.26124287]
 [ 1.60991316 -0.59173065 -0.59173065 -0.59173065]
 [-1.00331349  0.88119623  0.88119623  0.88119623]
 [-0.64148211  1.55070845  1.55070845  1.55070845]
 [ 0.32340158  1.55070845  1.55070845  1.55070845]
 [ 0.32340158 -1.39514532 -1.39514532 -1.39514532]
 [-1.2847379   1.01509868  1.01509868  1.01509868]
 [ 0.92645388  1.55070845  1.55070845  1.55070845]
 [ 0.32340158 -0.85953554 -0.85953554 -0.85953554]
 [ 0.32340158 -1.12734043 -1.12734043 -1.12734043]
 [-1.2847379   0.21168401  0.21168401  0.21168401]
 [ 0.32340158  0.07778157  0.07778157  0.07778157]
 [ 0.92645388  1.28290356  1.28290356  1.28290356]
 [ 1.60991316  1.55070845  1.55070845  1.55070845]
 [ 0.92645388 -1.12734043 -1.12734043 -1.12734043]
 [ 0.32340158 -1.66295021 -1.66295021 -1.66295021]
 [ 0.32340158  1.28290356  1.28290356  1.28290356]
```

In [22]:

```
print(X_train_scaled.mean(axis=0))
```

```
[ 7.18379604e-17  1.24083750e-16  1.24083750e-16 -6.53072367e-18]
```

In [23]:

```
print(X_train_scaled.std(axis=0))
```

```
[1. 1. 1. 1.]
```

In [24]:

```
X_test.head()
```

Out[24]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes
170	50000	29	58	268
58	15000	21	42	172
84	21875	21	42	172
148	39375	33	66	316
64	15000	27	54	244

In [25]:

```
scaler= StandardScaler().fit(X_test)
```

In [26]:

```
scaler.mean_
```

Out[26]:

```
array([22608.69565217,    27.73913043,    55.47826087,   252.86956522])
```

In [27]:

```
scaler.scale_
```

Out[27]:

```
array([1.37865771e+04, 7.58842019e+00, 1.51768404e+01, 9.10610423e+01])
```

In [28]:

```
scaler.transform(X_test)
```

Out[28]:

```
array([[ 1.98680965,  0.16615706,  0.16615706,  0.16615706],
       [-0.55189157, -0.88808082, -0.88808082, -0.88808082],
       [-0.05321812, -0.88808082, -0.88808082, -0.88808082],
       [ 1.21613249,  0.693276  ,  0.693276  ,  0.693276  ],
       [-0.55189157, -0.09740241, -0.09740241, -0.09740241],
       [-0.95989712,  1.0886152 ,  1.0886152 ,  1.0886152 ],
       [ 0.53612324, -0.22918215, -0.22918215, -0.22918215],
       [-0.55189157,  1.61573414,  1.61573414,  1.61573414],
       [-0.55189157,  0.95683547,  0.95683547,  0.95683547],
       [-0.95989712,  0.95683547,  0.95683547,  0.95683547],
       [ 0.53612324,  1.35217467,  1.35217467,  1.35217467],
       [ 0.53612324,  0.42971653,  0.42971653,  0.42971653],
       [-0.95989712,  1.61573414,  1.61573414,  1.61573414],
       [-0.55189157,  0.29793679,  0.29793679,  0.29793679],
       [-0.95989712,  0.693276  ,  0.693276  ,  0.693276  ],
       [ 0.53612324, -0.75630109, -0.75630109, -0.75630109],
       [-0.05321812,  0.82505573,  0.82505573,  0.82505573],
       [ 1.98680965, -1.41519976, -1.41519976, -1.41519976],
       [-0.55189157, -1.67875923, -1.67875923, -1.67875923],
       [ 1.98680965, -1.28342003, -1.28342003, -1.28342003],
       [ 0.53612324,  0.95683547,  0.95683547,  0.95683547],
       [-0.55189157,  0.56149626,  0.56149626,  0.56149626],
       [-0.05321812, -0.22918215, -0.22918215, -0.22918215],
       [-0.55189157,  1.35217467,  1.35217467,  1.35217467],
       [-1.27723478,  0.82505573,  0.82505573,  0.82505573],
       [-0.05321812,  0.693276  ,  0.693276  ,  0.693276  ],
       [-1.27723478,  0.693276  ,  0.693276  ,  0.693276  ],
       [-1.27723478, -0.36096188, -0.36096188, -0.36096188],
       [ 1.98680965,  1.61573414,  1.61573414,  1.61573414],
       [-0.95989712, -1.15164029, -1.15164029, -1.15164029],
       [-0.05321812, -0.62452135, -0.62452135, -0.62452135],
       [ 0.53612324, -1.5469795 , -1.5469795 , -1.5469795 ],
       [-0.95989712, -1.41519976, -1.41519976, -1.41519976],
       [ 0.53612324, -0.36096188, -0.36096188, -0.36096188],
       [-0.05321812, -1.15164029, -1.15164029, -1.15164029],
       [-0.05321812,  0.16615706,  0.16615706,  0.16615706],
       [-1.27723478,  0.56149626,  0.56149626,  0.56149626],
       [ 1.98680965, -0.36096188, -0.36096188, -0.36096188],
       [-0.55189157, -0.75630109, -0.75630109, -0.75630109],
       [-0.05321812,  1.0886152 ,  1.0886152 ,  1.0886152 ],
       [ 1.21613249, -0.88808082, -0.88808082, -0.88808082],
       [-1.27723478, -1.28342003, -1.28342003, -1.28342003],
       [-0.55189157,  1.22039494,  1.22039494,  1.22039494],
       [ 0.53612324, -1.28342003, -1.28342003, -1.28342003],
       [-0.55189157, -0.22918215, -0.22918215, -0.22918215],
       [ 1.98680965, -1.5469795 , -1.5469795 , -1.5469795 ]])
```

In [29]:

```
X_test_scaled = scaler.transform(X_test)
```

In [30]:

```
print(X_test_scaled)
```

```
[[ 1.98680965  0.16615706  0.16615706  0.16615706]
 [-0.55189157 -0.88808082 -0.88808082 -0.88808082]
 [-0.05321812 -0.88808082 -0.88808082 -0.88808082]
 [ 1.21613249  0.693276    0.693276    0.693276  ]
 [-0.55189157 -0.09740241 -0.09740241 -0.09740241]
 [-0.95989712  1.0886152   1.0886152   1.0886152 ]
 [ 0.53612324 -0.22918215 -0.22918215 -0.22918215]
 [-0.55189157  1.61573414  1.61573414  1.61573414]
 [-0.55189157  0.95683547  0.95683547  0.95683547]
 [-0.95989712  0.95683547  0.95683547  0.95683547]
 [ 0.53612324  1.35217467  1.35217467  1.35217467]
 [ 0.53612324  0.42971653  0.42971653  0.42971653]
 [-0.95989712  1.61573414  1.61573414  1.61573414]
 [-0.55189157  0.29793679  0.29793679  0.29793679]
 [-0.95989712  0.693276    0.693276    0.693276  ]
 [ 0.53612324 -0.75630109 -0.75630109 -0.75630109]
 [-0.05321812  0.82505573  0.82505573  0.82505573]
 [ 1.98680965 -1.41519976 -1.41519976 -1.41519976]
 [-0.55189157 -1.67875923 -1.67875923 -1.67875923]
```

In [31]:

```
print(X_test_scaled.mean(axis=0))
```

```
[ 7.72329061e-17 -7.24058494e-17 -7.24058494e-17 -8.20599627e-17]
```

In [32]:

```
print(X_test_scaled.std(axis=0))
```

```
[1. 1. 1. 1.]
```

LINEAR REGRESSION

In [33]:

```
df.head()
```

Out[33]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes	Number of Barriers
0	5000	15	30	100	30
1	5000	16	32	112	35
2	5000	17	34	124	42
3	5000	18	36	136	48
4	5000	19	38	148	56

In [34]:

```
df=df[['Area', 'Sensing Range', 'Transmission Range', 'Number of Sensor nodes', 'Number o
```

In [35]:

```
df
```

Out[35]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes	Number of Barriers
0	5000	15	30	100	30
1	5000	16	32	112	35
2	5000	17	34	124	42
3	5000	18	36	136	48
4	5000	19	38	148	56
...	...	...	...	...	...
177	50000	36	72	352	101
178	50000	37	74	364	107
179	50000	38	76	376	114
180	50000	39	78	388	121
181	50000	40	80	400	128

182 rows × 5 columns

In [36]:

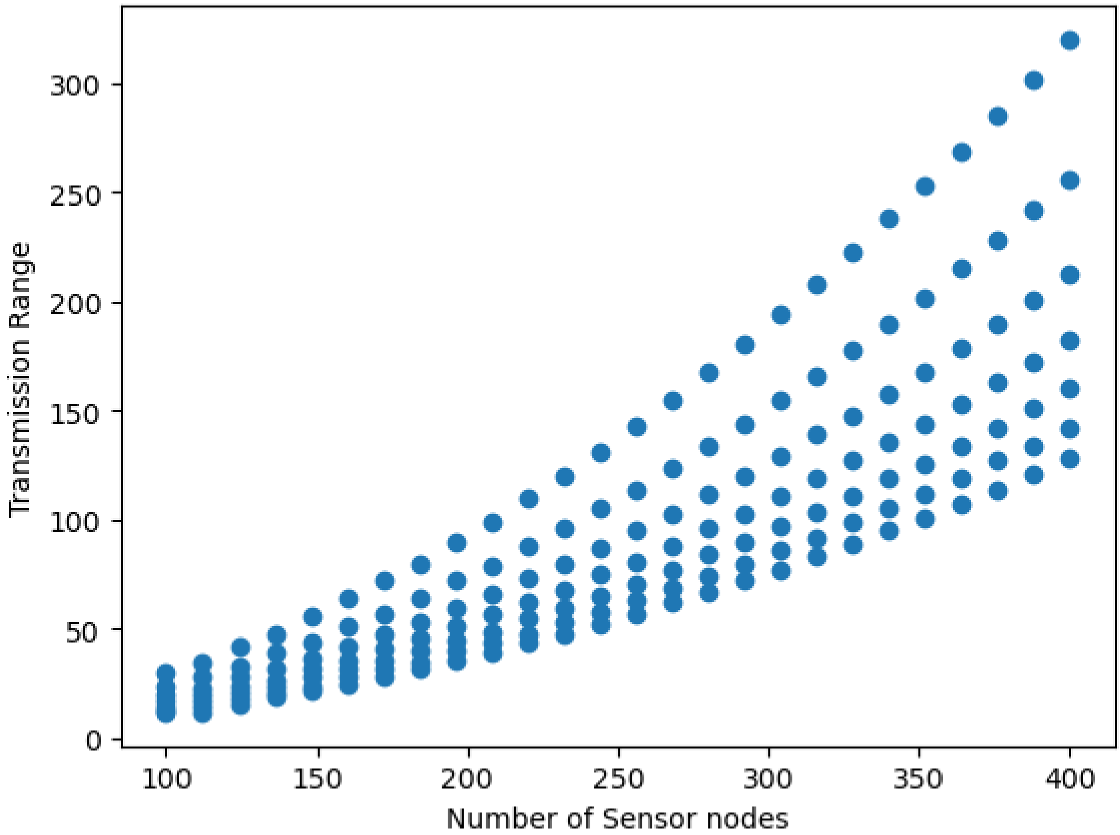
```
X = df['Number of Sensor nodes']
y = df['Number of Barriers']
```

In [37]:

```
import matplotlib.pyplot as plt
plt.scatter(X, y)
plt.xlabel('Number of Sensor nodes')
plt.ylabel('Transmission Range')
```

Out[37]:

```
Text(0, 0.5, 'Transmission Range')
```



In [38]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state = 2
```

In [39]:

```
X_train
```

Out[39]:

```
2      124
138    196
140    220
165    208
151    352
      ...
39     256
91     256
31     160
40     268
83     160
Name: Number of Sensor nodes, Length: 109, dtype: int64
```

In [40]:

```
import numpy as np
X_train = np.array(X_train).reshape(-1, 1)
X_train
```

Out[40]:

```
array([[124],
       [196],
       [220],
       [208],
       [352],
       [244],
       [196],
       [208],
       [292],
       [388],
       [184],
       [160],
       [160],
       [208],
       [184],
       [124],
       [220],
       [316],
```

In [41]:

```
X_test = np.array(X_train).reshape(-1, 1)
X_test
```

Out[41]:

```
array([[124],
       [196],
       [220],
       [208],
       [352],
       [244],
       [196],
       [208],
       [292],
       [388],
       [184],
       [160],
       [160],
       [208],
       [184],
       [124],
       [220],
       [316],
```

In [42]:

```
from sklearn.linear_model import LinearRegression
```

In [43]:

```
lr = LinearRegression()
```

In [44]:

```
lr.fit(X_train, y_train)
```

Out[44]:

```
▾ LinearRegression
LinearRegression()
```

In [45]:

```
c = lr.intercept_
c
```

Out[45]:

```
-53.07596881156522
```

In [46]:

```
m = lr.coef_
m
```

Out[46]:

```
array([0.585255])
```

In [47]:

```
Y_pred_train = m*X_train + c
Y_pred_train.flatten()
```

Out[47]:

```
array([ 19.49565138,  61.63401149,  75.68013152,  68.65707151,
       152.93379173,  89.72625156,  61.63401149,  68.65707151,
       117.81849163, 174.00297178,  54.61095147,  40.56483143,
        40.56483143,  68.65707151,  54.61095147,  19.49565138,
        75.68013152, 131.86461167, 138.88767169, 124.84155165,
        33.54177141,  61.63401149, 159.95685174,  54.61095147,
        26.5187114 , 124.84155165, 174.00297178,  40.56483143,
       110.79543162,  33.54177141,  89.72625156, 117.81849163,
        26.5187114 , 166.97991176, 145.91073171,  12.47259136,
       124.84155165,  26.5187114 ,  47.58789145, 138.88767169,
         5.44953134,  82.70319154,   5.44953134,  82.70319154,
        12.47259136,  82.70319154, 159.95685174,  54.61095147,
        68.65707151,  19.49565138,  54.61095147,  61.63401149,
        12.47259136, 117.81849163, 117.81849163,  89.72625156,
        96.74931158, 138.88767169, 124.84155165,  75.68013152,
        12.47259136, 103.7723716 , 131.86461167,  19.49565138,
         5.44953134, 110.79543162,  82.70319154, 145.91073171,
        47.58789145, 131.86461167,   5.44953134, 117.81849163,
       166.97991176, 159.95685174,  61.63401149,  96.74931158,
        33.54177141,  68.65707151, 152.93379173, 181.0260318 ,
       103.7723716 , 181.0260318 , 110.79543162,  89.72625156,
       181.0260318 ,  96.74931158,  75.68013152,  96.74931158,
       152.93379173,  40.56483143, 124.84155165,  40.56483143,
       145.91073171,  12.47259136, 110.79543162, 159.95685174,
       152.93379173, 117.81849163, 166.97991176,  89.72625156,
       138.88767169,  47.58789145, 166.97991176,  89.72625156,
        96.74931158,  96.74931158,  40.56483143, 103.7723716 ,
        40.56483143])
```

In [48]:

```
y_pred_train1= lr.predict(X_train)
y_pred_train1
```

Out[48]:

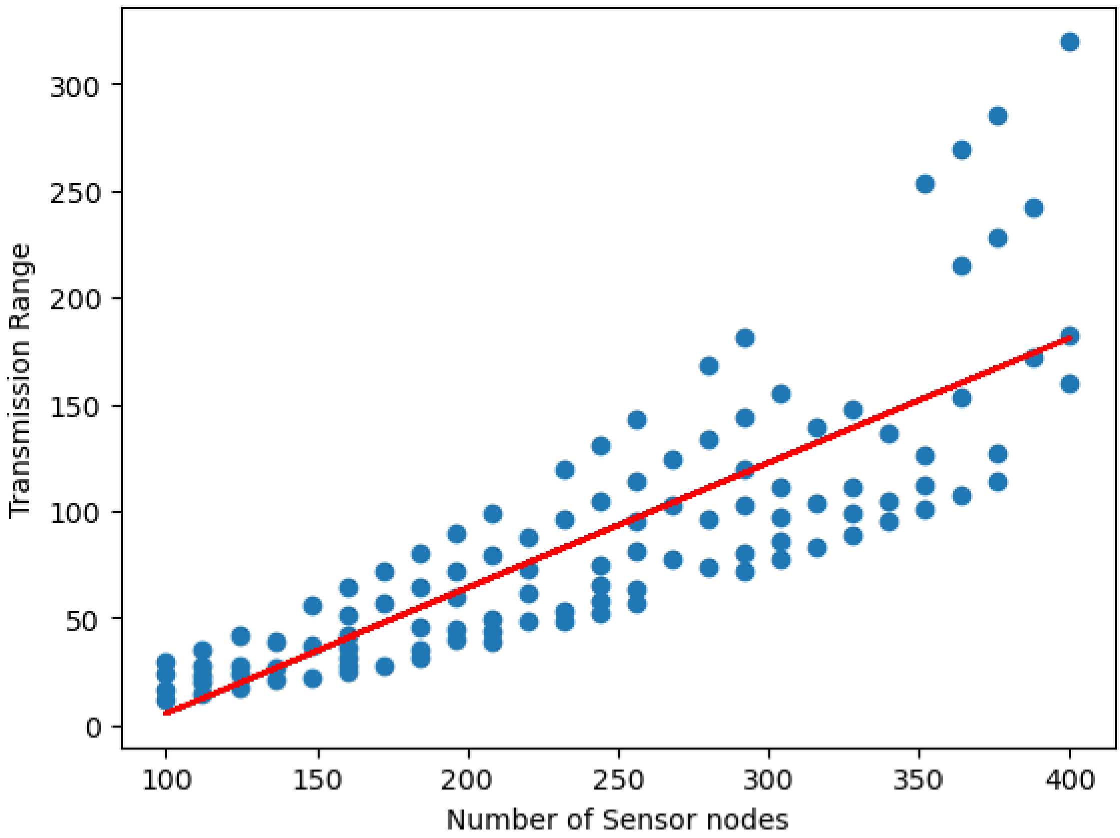
```
array([ 19.49565138,  61.63401149,  75.68013152,  68.65707151,
       152.93379173,  89.72625156,  61.63401149,  68.65707151,
       117.81849163, 174.00297178,  54.61095147,  40.56483143,
        40.56483143,  68.65707151,  54.61095147,  19.49565138,
        75.68013152, 131.86461167, 138.88767169, 124.84155165,
        33.54177141,  61.63401149, 159.95685174,  54.61095147,
        26.5187114 , 124.84155165, 174.00297178,  40.56483143,
       110.79543162,  33.54177141,  89.72625156, 117.81849163,
        26.5187114 , 166.97991176, 145.91073171,  12.47259136,
       124.84155165,  26.5187114 ,  47.58789145, 138.88767169,
         5.44953134,  82.70319154,   5.44953134,  82.70319154,
        12.47259136,  82.70319154, 159.95685174,  54.61095147,
        68.65707151,  19.49565138,  54.61095147,  61.63401149,
        12.47259136, 117.81849163, 117.81849163,  89.72625156,
        96.74931158, 138.88767169, 124.84155165,  75.68013152,
        12.47259136, 103.7723716 , 131.86461167,  19.49565138,
         5.44953134, 110.79543162,  82.70319154, 145.91073171,
        47.58789145, 131.86461167,   5.44953134, 117.81849163,
       166.97991176, 159.95685174,  61.63401149,  96.74931158,
        33.54177141,  68.65707151, 152.93379173, 181.0260318 ,
       103.7723716 , 181.0260318 , 110.79543162,  89.72625156,
       181.0260318 ,  96.74931158,  75.68013152,  96.74931158,
       152.93379173,  40.56483143, 124.84155165,  40.56483143,
       145.91073171,  12.47259136, 110.79543162, 159.95685174,
       152.93379173, 117.81849163, 166.97991176,  89.72625156,
       138.88767169,  47.58789145, 166.97991176,  89.72625156,
        96.74931158,  96.74931158,  40.56483143, 103.7723716 ,
        40.56483143])
```

In [49]:

```
import matplotlib.pyplot as plt
plt.scatter(X_train, y_train)
plt.plot(X_train, y_pred_train1, color='red')
plt.xlabel('Number of Sensor nodes')
plt.ylabel('Transmission Range')
```

Out[49]:

```
Text(0, 0.5, 'Transmission Range')
```



In [50]:

```
y_pred_test1= lr.predict(X_test)
y_pred_test1
```

Out[50]:

```
array([ 19.49565138,  61.63401149,  75.68013152,  68.65707151,
       152.93379173,  89.72625156,  61.63401149,  68.65707151,
       117.81849163, 174.00297178,  54.61095147,  40.56483143,
        40.56483143,  68.65707151,  54.61095147,  19.49565138,
        75.68013152, 131.86461167, 138.88767169, 124.84155165,
        33.54177141,  61.63401149, 159.95685174,  54.61095147,
        26.5187114 , 124.84155165, 174.00297178,  40.56483143,
       110.79543162,  33.54177141,  89.72625156, 117.81849163,
        26.5187114 , 166.97991176, 145.91073171,  12.47259136,
       124.84155165,  26.5187114 ,  47.58789145, 138.88767169,
         5.44953134,  82.70319154,   5.44953134,  82.70319154,
        12.47259136,  82.70319154, 159.95685174,  54.61095147,
        68.65707151,  19.49565138,  54.61095147,  61.63401149,
        12.47259136, 117.81849163, 117.81849163,  89.72625156,
        96.74931158, 138.88767169, 124.84155165,  75.68013152,
        12.47259136, 103.7723716 , 131.86461167,  19.49565138,
         5.44953134, 110.79543162,  82.70319154, 145.91073171,
        47.58789145, 131.86461167,   5.44953134, 117.81849163,
       166.97991176, 159.95685174,  61.63401149,  96.74931158,
        33.54177141,  68.65707151, 152.93379173, 181.0260318 ,
       103.7723716 , 181.0260318 , 110.79543162,  89.72625156,
       181.0260318 ,  96.74931158,  75.68013152,  96.74931158,
       152.93379173,  40.56483143, 124.84155165,  40.56483143,
       145.91073171,  12.47259136, 110.79543162, 159.95685174,
       152.93379173, 117.81849163, 166.97991176,  89.72625156,
       138.88767169,  47.58789145, 166.97991176,  89.72625156,
        96.74931158,  96.74931158,  40.56483143, 103.7723716 ,
        40.56483143])
```

In [51]:

```
print(len(X_train), len(y_train))
print(len(X_test), len(y_test))
```

```
109 109
109 73
```

In [52]:

```
import matplotlib.pyplot as plt
plt.scatter(X_test, y_test)
plt.plot(X_test, y_pred_test1, color='red')
plt.xlabel('Number of Sensor nodes')
plt.ylabel('Transmission Range')
```

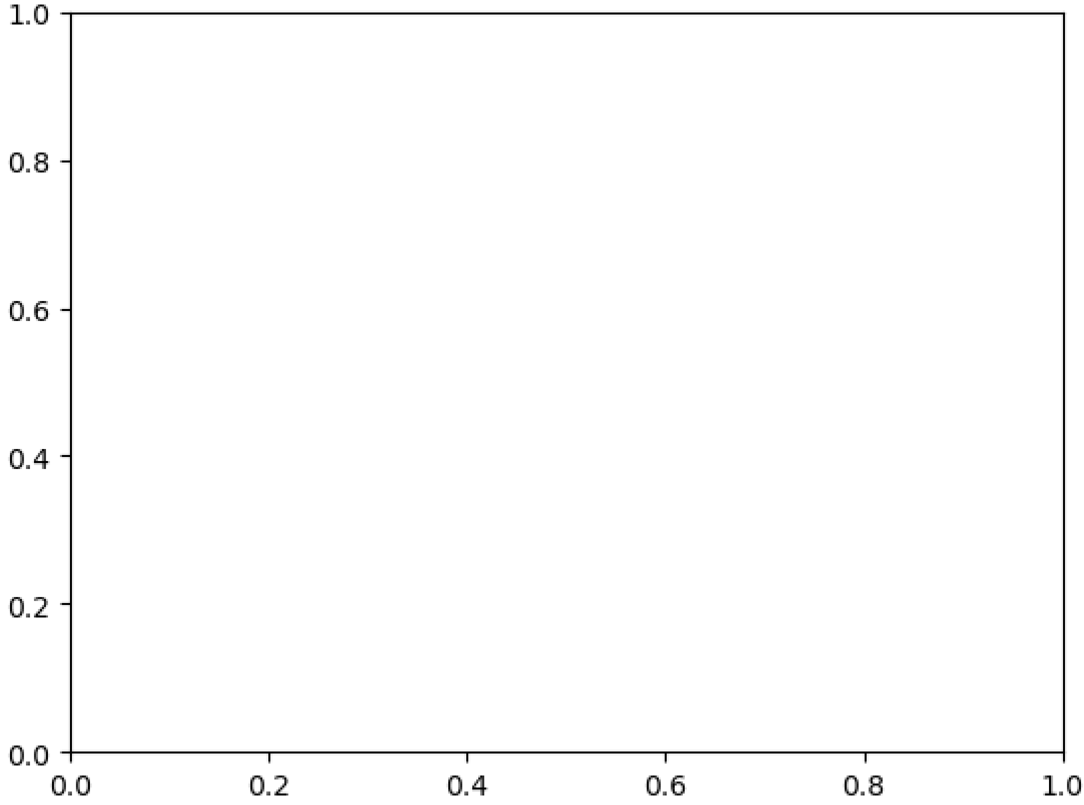
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[52], line 2
      1 import matplotlib.pyplot as plt
----> 2 plt.scatter(X_test, y_test)
      3 plt.plot(X_test, y_pred_test1, color='red')
      4 plt.xlabel('Number of Sensor nodes')

File G:\Anaconda3\Lib\site-packages\matplotlib\pyplot.py:2862, in scatter(x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths, edgecolors, plotnonfinite, data, **kwargs)
   2857 @_copy_docstring_and_deprecators(Axes.scatter)
   2858 def scatter(
   2859         x, y, s=None, c=None, marker=None, cmap=None, norm=None,
   2860         vmin=None, vmax=None, alpha=None, linewidths=None, *,
   2861         edgecolors=None, plotnonfinite=False, data=None, **kwargs):
-> 2862     __ret = gca().scatter(
   2863         x, y, s=s, c=c, marker=marker, cmap=cmap, norm=norm,
   2864         vmin=vmin, vmax=vmax, alpha=alpha, linewidths=linewidths,
   2865         edgecolors=edgecolors, plotnonfinite=plotnonfinite,
   2866         **({"data": data} if data is not None else {}), **kwargs)
   2867     sci(__ret)
   2868     return __ret

File G:\Anaconda3\Lib\site-packages\matplotlib\__init__.py:1442, in _preprocess_data.<locals>.inner(ax, data, *args, **kwargs)
   1439 @functools.wraps(func)
   1440 def inner(ax, *args, data=None, **kwargs):
   1441     if data is None:
-> 1442         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1444     bound = new_sig.bind(ax, *args, **kwargs)
   1445     auto_label = (bound.arguments.get(label_namer)
   1446                   or bound.kwargs.get(label_namer))

File G:\Anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:4584, in Axes.scatter(self, x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths, edgecolors, plotnonfinite, **kwargs)
   4582 y = np.ma.ravel(y)
   4583 if x.size != y.size:
-> 4584     raise ValueError("x and y must be the same size")
   4586 if s is None:
   4587     s = (20 if mpl.rcParams['_internal.classic_mode'] else
   4588          mpl.rcParams['lines.markersize'] ** 2.0)

ValueError: x and y must be the same size
```



In [53]:

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

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

In [54]:

```
df
```

Out[54]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes	Number of Barriers
0	5000	15	30	100	30
1	5000	16	32	112	35
2	5000	17	34	124	42
3	5000	18	36	136	48
4	5000	19	38	148	56
...	...	...	...	...	...
177	50000	36	72	352	101
178	50000	37	74	364	107
179	50000	38	76	376	114
180	50000	39	78	388	121
181	50000	40	80	400	128

182 rows × 5 columns

In [55]:

```
df.head
```

Out[55]:

```
<bound method NDFrame.head of       Area  Sensing Range  Transmission Rang
e  Number of Sensor nodes  \
0     5000             15                   30                      100
1     5000             16                   32                      112
2     5000             17                   34                      124
3     5000             18                   36                      136
4     5000             19                   38                      148
..     ...            ...                  ...                      ...
177  50000             36                   72                      352
178  50000             37                   74                      364
179  50000             38                   76                      376
180  50000             39                   78                      388
181  50000             40                   80                      400

     Number of Barriers
0                    30
1                    35
2                    42
3                    48
4                    56
..                  ...
177                 101
178                 107
179                 114
180                 121
181                 128

[182 rows x 5 columns]>
```

In [56]:

```
df.dropna(inplace = True)
```

In [57]:

```
df = df.drop(['Area', 'Sensing Range', 'Transmission Range'], axis = 1)
xVars = df.drop('Number of Barriers', axis = 1)
yVars = df[['Number of Barriers']]
xTrain, xValid, yTrain, yValid = train_test_split(xVars, yVars, test_size = 0.3, random_s
```

In [58]:

```
print(xTrain.shape)
print(df.shape)
```

```
(127, 1)
(182, 2)
```

In [59]:

```
regressor = RandomForestRegressor(n_estimators = 100, random_state = 42)
```

In [60]:

```
regressor.fit(xTrain, yTrain)
```

```
C:\Users\crisl\AppData\Local\Temp\ipykernel_5052\2921093390.py:1: DataConv
ersionWarning: 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().
  regressor.fit(xTrain, yTrain)
```

Out[60]:

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [61]:

```
y_pred = regressor.predict(xValid)
```

In [62]:

```
y_pred
```

Out[62]:

```
array([113.36621429,  93.76233333, 178.08805051, 151.21784921,
        93.76233333,  75.888     , 178.08805051,  93.76233333,
        77.57272414, 133.01126587, 121.70321681,  35.71882143,
       121.70321681,  54.15714286, 113.36621429,  93.76233333,
       188.75459957, 187.83736508, 113.36621429,  35.71882143,
       133.01126587, 133.01126587, 108.27071429,  35.71882143,
        57.66602381, 113.36621429,  55.629085  , 133.01126587,
        57.66602381,  96.16725397,  93.76233333,  27.62276587,
        64.59147619,  35.71882143, 108.27071429,  27.62276587,
        75.888     ,  96.16725397,  75.888     ,  39.60751587,
        64.59147619,  75.888     , 171.65605556,  75.888     ,
        71.26211652,  54.15714286,  96.16725397,  64.59147619,
       108.27071429, 151.21784921, 187.83736508,  39.60751587,
        23.1694044 ,  35.71882143, 151.21784921])
```

In [63]:

```
y_pred = pd.DataFrame(y_pred, columns =['yPredict'])
```

In [64]:

```
y_pred
```

Out[64]:

	yPredict
0	113.366214
1	93.762333
2	178.088051
3	151.217849
4	93.762333
5	75.888000
6	178.088051
7	93.762333
8	77.572724
9	133.011266
10	121.703217
11	35.718821
12	121.703217
13	54.157143
14	113.366214
15	93.762333
16	188.754600
17	187.837365
18	113.366214
19	35.718821
20	133.011266
21	133.011266
22	108.270714
23	35.718821
24	57.666024
25	113.366214
26	55.629085
27	133.011266
28	57.666024
29	96.167254
30	93.762333
31	27.622766
32	64.591476
33	35.718821
34	108.270714
35	27.622766
36	75.888000

	yPredict
37	96.167254
38	75.888000
39	39.607516
40	64.591476
41	75.888000
42	171.656056
43	75.888000
44	71.262117
45	54.157143
46	96.167254
47	64.591476
48	108.270714
49	151.217849
50	187.837365
51	39.607516
52	23.169404
53	35.718821
54	151.217849

In [65]:

```
yValid
```

Out[65]:

	Number of Barriers
19	223
42	144
154	134
98	136
146	80
15	168
24	302
68	120
115	60
96	119
95	111
160	22
69	129
111	40
45	178
16	181
51	256
127	142
97	127
56	37
174	83
122	104
144	69
30	44
9	99
123	111
60	60
18	208
165	39
143	63
172	72
55	32
90	75
82	32
66	103
29	39
119	84

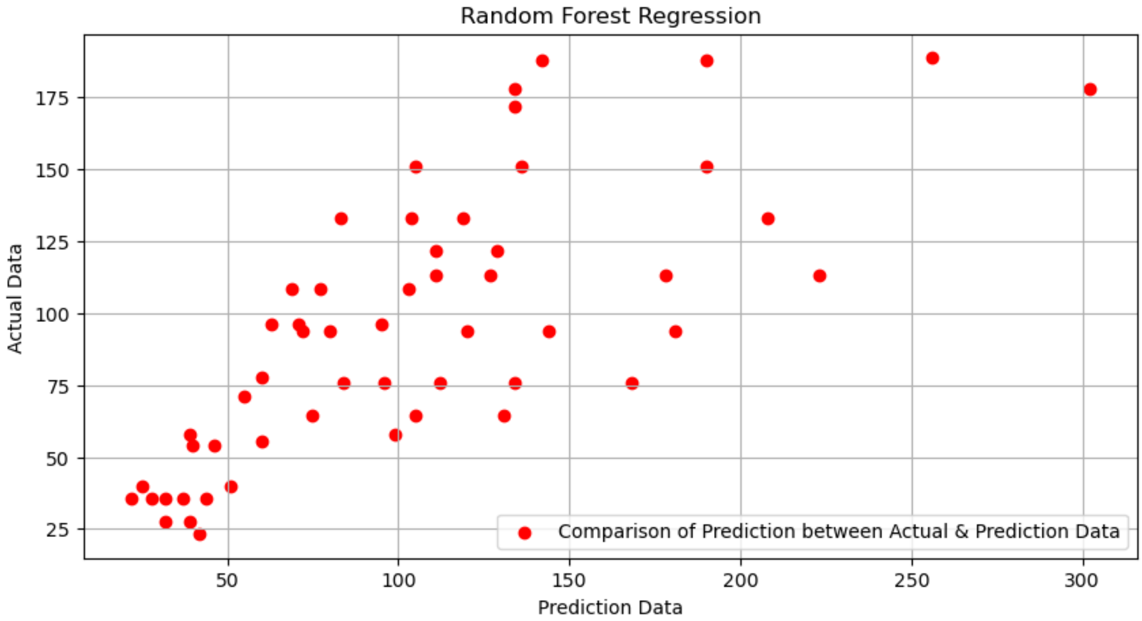
	Number of Barriers
65	95
67	112
31	51
12	131
41	134
126	134
93	96
114	55
85	46
117	71
38	105
118	77
150	105
75	190
161	25
2	42
108	28
46	190

In [66]:

```
plt.figure(figsize = (10, 5))
plt.scatter(yValid, y_pred, color = 'red', label = 'Comparison of Prediction between Actu
plt.legend()
plt.grid()
plt.title('Random Forest Regression')
plt.xlabel('Prediction Data')
plt.ylabel('Actual Data')
plt.show
```

Out[66]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [67]:

```
from sklearn.metrics import r2_score
r2 = r2_score(yValid, y_pred)
print(f'R-squared (R2) score: {r2}')
```

```
R-squared (R2) score: 0.5296291041597245
```