ASSIGNMENT 4

Data Analytics 1

Create a Linear Regression Model using Python/R to predict home prices using Boston Housing dataset(kaggle.com/c/boston-housing) and predict the value of prices of the house using the given features.

#packages

1.sklearn modules (train_test_split , LinearRegression, make_regression, load_iris) — These will be necessary in loading the iris dataset, preparation of data and fitting of the model.

2.matplotlib pyplot module — Needed to plot the results.

3.pandas and numpy packages — Needed to manipulate the dataframe and its constituent values

Importing required libraries

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model
from sklearn.model_selection import train_test_split
```

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

In [5]:
```
df
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [6]:
```
#Extracting Independent Variables
df_x = df.iloc[:,0:4].values
```

In [7]:
```
df_x
```

Out[7]:
```
array([[  1. ,   5.1,   3.5,   1.4],
       [  2. ,   4.9,   3. ,   1.4],
       [  3. ,   4.7,   3.2,   1.3],
       [  4. ,   4.6,   3.1,   1.5],
       [  5. ,   5. ,   3.6,   1.4],
       [  6. ,   5.4,   3.9,   1.7],
       [  7. ,   4.6,   3.4,   1.4],
       [  8. ,   5. ,   3.4,   1.5],
       [  9. ,   4.4,   2.9,   1.4],
       [ 10. ,   4.9,   3.1,   1.5],
       [ 11. ,   5.4,   3.7,   1.5],
       [ 12. ,   4.8,   3.4,   1.6],
       [ 13. ,   4.8,   3. ,   1.4],
       [ 14. ,   4.3,   3. ,   1.1],
       [ 15. ,   5.8,   4. ,   1.2],
       [ 16. ,   5.7,   4.4,   1.5],
       [ 17. ,   5.4,   3.9,   1.3],
       [ 18. ,   5.1,   3.5,   1.4],
       [ 19. ,   5.7,   3.8,   1.7],
```

In [8]:
```
#Extracting Dependent Variables
df_y = df.iloc[:,5].values
```

In [9]:
```
df_y
```

Out[9]:
```
array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolo
r',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica'], dtype=object)
```

In [10]:
```
from sklearn.preprocessing import LabelEncoder
```

In [11]:
```
df_y = LabelEncoder().fit_transform(df_y)
```

Initializing the Linear Regression Model

linear regression describes the relationship between dependent and independent variables. It describes the relationship by fitting a straight line. In machine learning, we do not use the whole dataset to develop the model. We split the dataset into two portions. One portion is for building the model which is called the training set and the other portion is for evaluating the model which is called the test data set. A key idea in supervised machine learning is the ability to test your model's competency on untouched data. The train-test split essentially uses 75% to generate a best fit line, and uses the remaining 25% to evaluate the model's performance on new data

In [12]:
```
# Instantiating LinearRegression() Model
reg = linear_model.LinearRegression()
# Splitting the Dataset
x_train,x_test,y_train,y_test = train_test_split(df_x,df_y)
```

In [13]:
```
x_train
```

Out[13]:
```
array([[125. ,   6.7,   3.3,   5.7],
       [ 25. ,   4.8,   3.4,   1.9],
       [ 43. ,   4.4,   3.2,   1.3],
       [ 91. ,   5.5,   2.6,   4.4],
       [ 24. ,   5.1,   3.3,   1.7],
       [ 34. ,   5.5,   4.2,   1.4],
       [115. ,   5.8,   2.8,   5.1],
       [114. ,   5.7,   2.5,   5. ],
       [ 59. ,   6.6,   2.9,   4.6],
       [ 87. ,   6.7,   3.1,   4.7],
       [ 80. ,   5.7,   2.6,   3.5],
       [124. ,   6.3,   2.7,   4.9],
       [ 94. ,   5. ,   2.3,   3.3],
       [ 72. ,   6.1,   2.8,   4. ],
       [147. ,   6.3,   2.5,   5. ],
       [ 50. ,   5. ,   3.3,   1.4],
       [  2. ,   4.9,   3. ,   1.4],
       [  7. ,   4.6,   3.4,   1.4],
       [144. ,   6.8,   3.2,   5.9],
```

In [14]:
```
x_test
```

Out[14]:
```
array([[134. ,   6.3,   2.8,   5.1],
       [ 38. ,   4.9,   3.1,   1.5],
       [122. ,   5.6,   2.8,   4.9],
       [133. ,   6.4,   2.8,   5.6],
       [ 41. ,   5. ,   3.5,   1.3],
       [ 55. ,   6.5,   2.8,   4.6],
       [ 40. ,   5.1,   3.4,   1.5],
       [ 12. ,   4.8,   3.4,   1.6],
       [ 75. ,   6.4,   2.9,   4.3],
       [ 22. ,   5.1,   3.7,   1.5],
       [110. ,   7.2,   3.6,   6.1],
       [141. ,   6.7,   3.1,   5.6],
       [ 31. ,   4.8,   3.1,   1.6],
       [  1. ,   5.1,   3.5,   1.4],
       [ 96. ,   5.7,   3. ,   4.2],
       [137. ,   6.3,   3.4,   5.6],
       [ 78. ,   6.7,   3. ,   5. ],
       [ 90. ,   5.5,   2.5,   4. ],
       [123. ,   7.7,   2.8,   6.7],
       [106. ,   7.6,   3. ,   6.6],
       [ 16. ,   5.7,   4.4,   1.5],
       [127. ,   6.2,   2.8,   4.8],
       [113. ,   6.8,   3. ,   5.5],
       [117. ,   6.5,   3. ,   5.5],
       [ 35. ,   4.9,   3.1,   1.5],
       [120. ,   6. ,   2.2,   5. ],
       [138. ,   6.4,   3.1,   5.5],
       [135. ,   6.1,   2.6,   5.6],
       [ 85. ,   5.4,   3. ,   4.5],
       [107. ,   4.9,   2.5,   4.5],
       [ 42. ,   4.5,   2.3,   1.3],
       [ 92. ,   6.1,   3. ,   4.6],
       [ 52. ,   6.4,   3.2,   4.5],
       [ 29. ,   5.2,   3.4,   1.4],
       [131. ,   7.4,   2.8,   6.1],
       [ 68. ,   5.8,   2.7,   4.1],
       [111. ,   6.5,   3.2,   5.1],
       [ 76. ,   6.6,   3. ,   4.4]])
```

In [15]:
```
y_train
```

Out[15]:
```
array([2, 0, 0, 1, 0, 0, 2, 2, 1, 1, 1, 2, 1, 1, 2, 0, 0, 0, 2, 2, 2, 2,
       0, 0, 1, 2, 1, 2, 0, 2, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 0,
       0, 2, 0, 0, 0, 0, 2, 2, 0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 1, 1, 0, 2,
       0, 1, 1, 0, 0, 2, 1, 0, 2, 1, 2, 1, 0, 0, 0, 0, 0, 2, 0, 2, 2, 0,
       1, 1, 1, 1, 1, 2, 2, 1, 1, 2, 0, 2, 0, 2, 1, 0, 1, 0, 0, 0, 1, 1,
       1, 1])
```

In [16]:
```
y_test
```

Out[16]:
```
array([2, 0, 2, 2, 0, 1, 0, 0, 1, 0, 2, 2, 0, 0, 1, 2, 1, 1, 2, 2, 0, 2,
       2, 2, 0, 2, 2, 2, 1, 2, 0, 1, 1, 0, 2, 1, 2, 1])
```

In [17]:
```
#TRAINING THE MODEL
reg.fit(x_train,y_train)
```

Out[17]:
```
LinearRegression
LinearRegression()
```

In machine learning language we do not use the term slope. Instead, it is known as the coefficient of X. Here is the coefficient for this model:

In [18]:
```
#coeff of each feature
print(reg.coef_)
```

```
[ 0.00814555 -0.09293236  0.02948516  0.30000912]
```

Prediction

In [19]:
```
# Making Predictions
y_pred = reg.predict(x_test)
```

In [20]:
```
y_pred
```

Out[20]:
```
array([ 1.82407386,  0.10101874,  1.73137804,  1.95663963,  0.06795441,
        1.01198411,  0.10756892, -0.06262595,  1.09713419, -0.03020549,
        1.86853871,  2.0027699 ,  0.08329402, -0.23716005,  1.30619107,
        2.01620618,  1.30664605,  1.20115981,  2.08438207,  1.93109701,
       -0.11419861,  1.68634548,  1.73245174,  1.79291366,  0.07658208,
        1.69022381,  1.97621203,  1.99491341,  1.33447243,  1.5453982 ,
        0.08718394,  1.35643956,  0.97863384, -0.02132631,  1.99742073,
        1.02997588,  1.62993372,  1.1196427 ])
```

In [21]:
```
y_test
```

Out[21]:
```
array([2, 0, 2, 2, 0, 1, 0, 0, 1, 0, 2, 2, 0, 0, 1, 2, 1, 1, 2, 2, 0, 2,
       2, 2, 0, 2, 2, 2, 1, 2, 0, 1, 1, 0, 2, 1, 2, 1])
```

Model Accuracy using Mean Squared Error

In [22]:
```
print(np.mean(y_pred-y_test)**2)
```

```
0.00037793048037458213
```

In [24]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
print('Mean Absolute Error:', mean_absolute_error(y_test,y_pred))
print('Mean Squared Error:', mean_squared_error(y_test,y_pred))
print('Mean Root Squared Error:', np.sqrt(mean_squared_error(y_test,y_pred
```

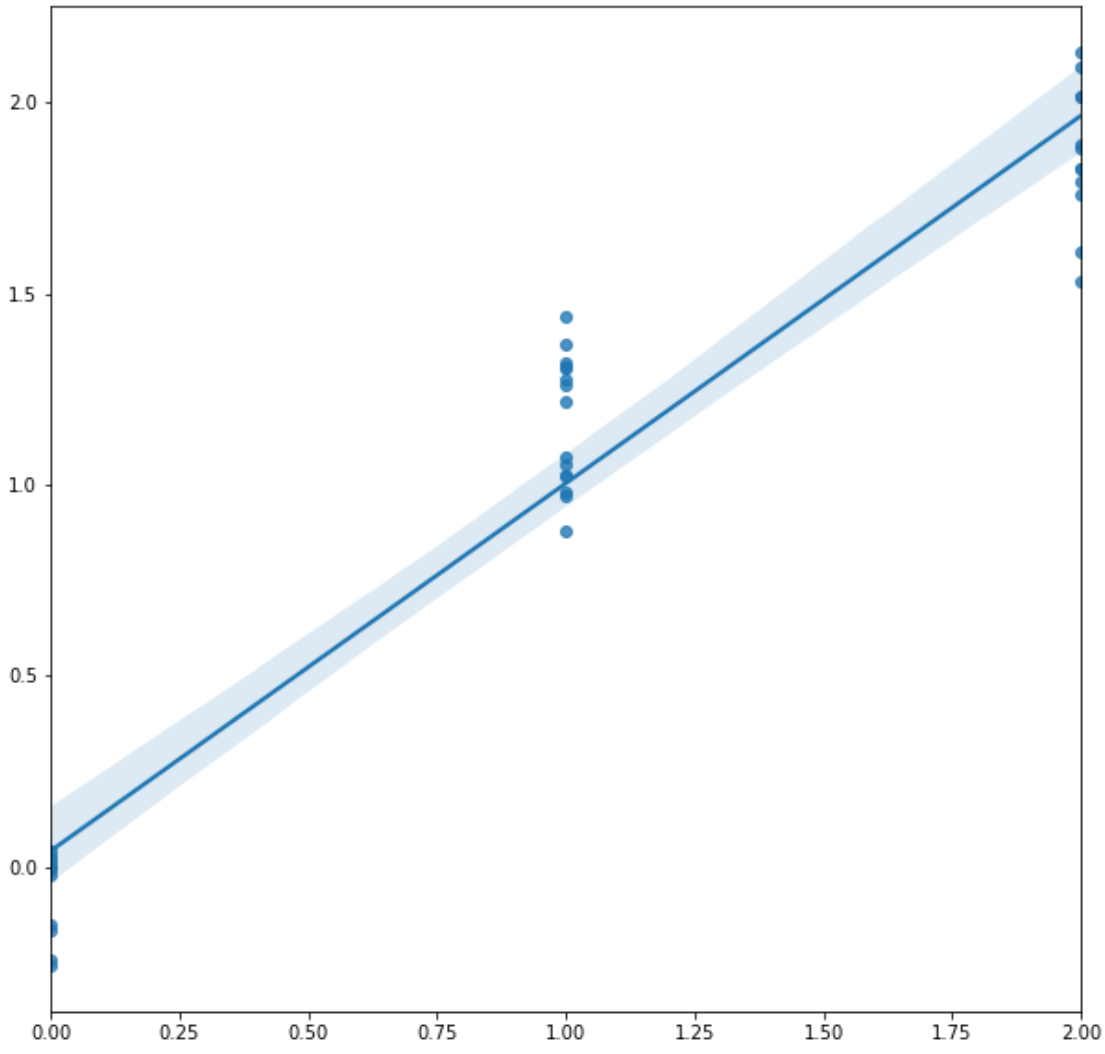
```
Mean Absolute Error: 0.1452618044381814
Mean Squared Error: 0.036874148890227686
Mean Root Squared Error: 0.19202642758283997
```

Data Visualization

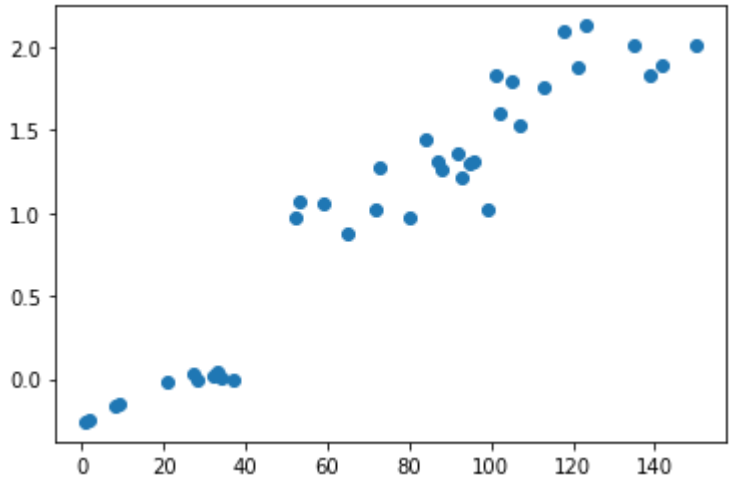
In [58]:
```
plt.figure(figsize=(10,10))
sns.regplot(y_test,y_pred)
```

```
C:\Users\Shravani Sajekar\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword 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(
```

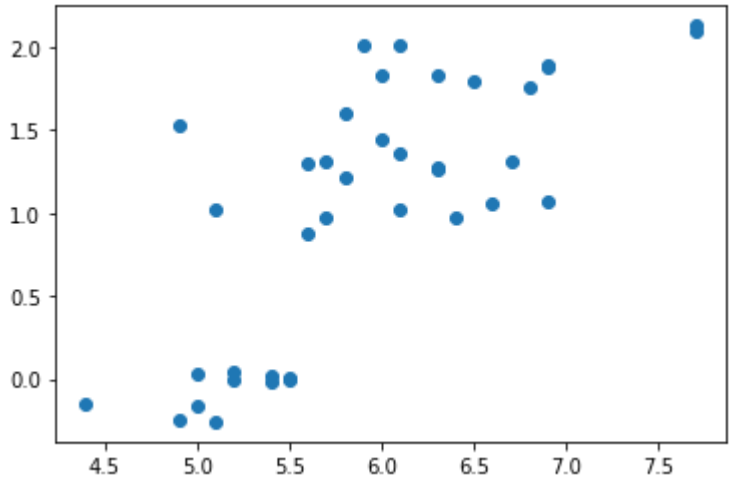
Out[58]: <AxesSubplot:>



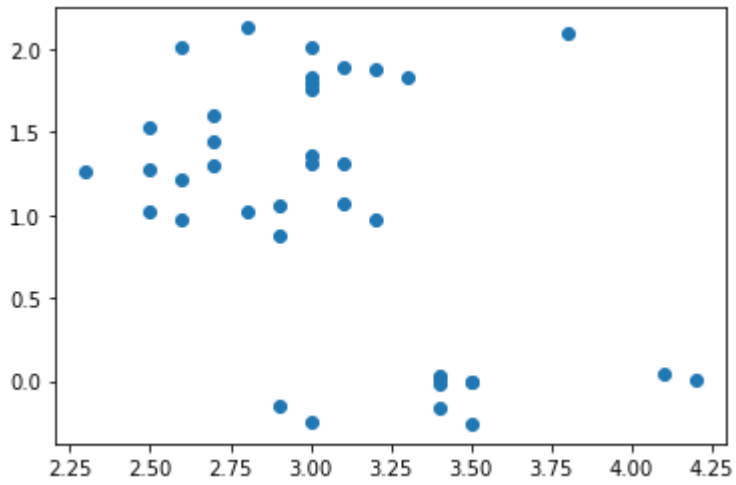
In [63]:
```
#sepal length
plt.scatter(x_test[:,0],y_pred)
plt.show()
```



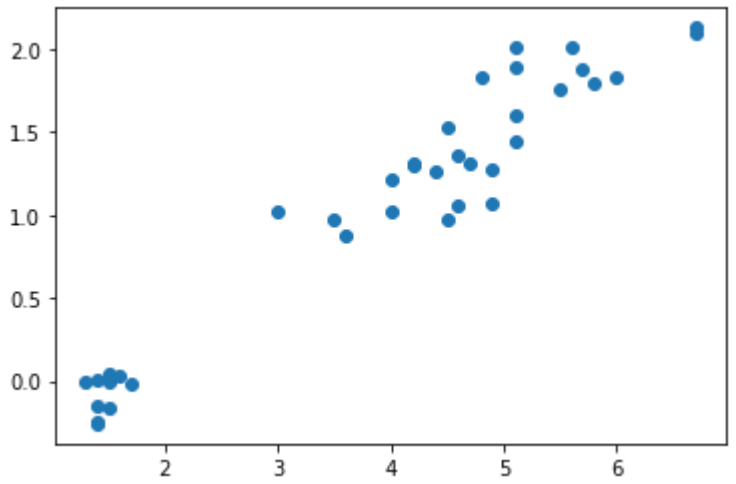
In [64]:
```
#sepal width
plt.scatter(x_test[:,1],y_pred)
plt.show()
```



In [65]:
```
#petal length
plt.scatter(x_test[:,2],y_pred)
plt.show()
```

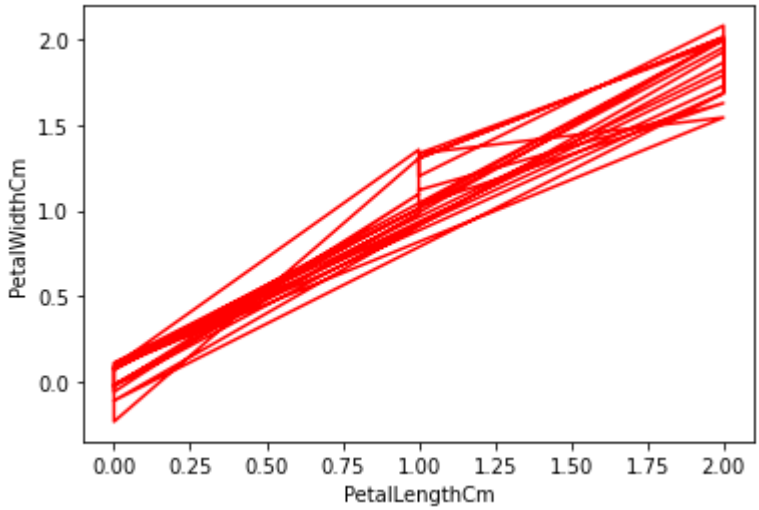


In [66]:
```
#petal width
plt.scatter(x_test[:,3],y_pred)
plt.show()
```



In [29]:
```
plt.plot(y_test, y_pred, color ='red')
plt.xlabel("PetalLengthCm")
plt.ylabel("PetalWidthCm")
```

Out[29]: Text(0, 0.5, 'PetalWidthCm')



In [ ]: