Chapter 9 - Ex1: Iris - Simple Linear Regression

Cho dữ liệu iris.xls

Yêu cầu: Thực hiện linenear regression để từ pentalwidth => dự đoán pentallength

1. Đọc dữ liệu, trực quan hóa dữ liệu.
2. Tạo X_train, X_test, y_train, y_test từ dữ liệu đọc được là 2 cột pentalwidth (inputs) và pentallength (outputs) với tỷ lệ dữ liệu test là 0.2
3. Áp dụng linrear regression
4. Vẽ hình. Nhận xét kết quả
5. Nếu pentalwidth là 1.5 => pentallength là bao nhiêu?

- url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data' (https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data')

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
# %cd '/content/gdrive/My Drive/MDS5_2022/Practice_2022/Chapter9/'
```

In [2]:
```
import pandas as pd
iris = pd.read_excel("Iris.xls", encoding='utf-8')
```

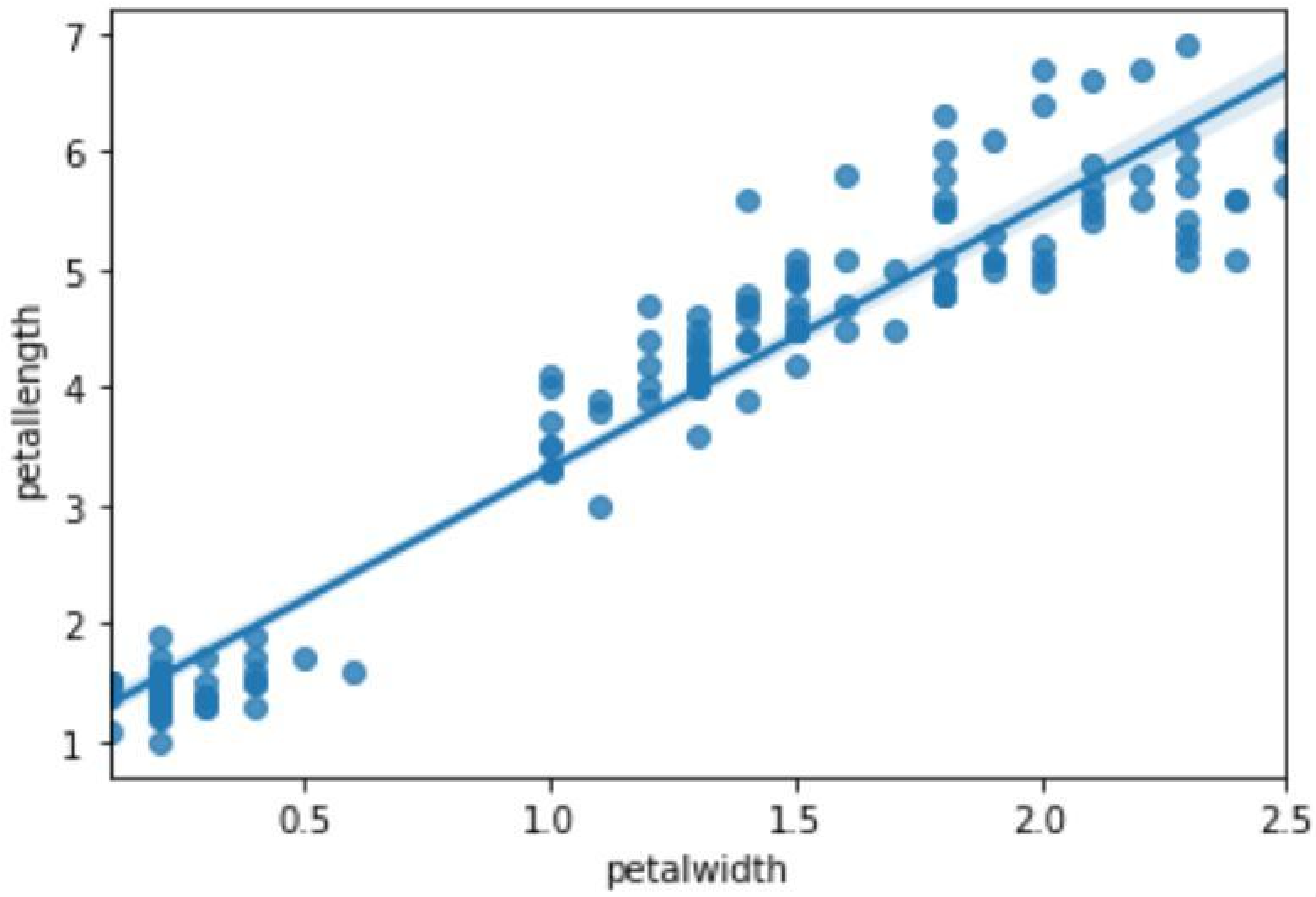
In [3]:
```
iris.head()
```

Out[3]:

	sepallength	sepalwidth	petallength	petalwidth	iris
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [4]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
sns.regplot(data=iris, x='petalwidth', y='petallength')
plt.show()
```



In [6]:
```
inputs = iris[['petalwidth']]
inputs.head()
```

Out[6]:

	petalwidth
0	0.2
1	0.2
2	0.2
3	0.2
4	0.2

In [7]:
```
outputs = iris[['petallength']]
outputs.head()
```

Out[7]:

	petallength
0	1.4
1	1.4
2	1.3
3	1.5
4	1.4

In [8]:
```
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

In [9]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, outputs,
                                                    test_size=0.20) # train data
```

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)

In [10]:
```
# huan luyen bang du lieu train data
regr1 = linear_model.LinearRegression()
regr1 = regr1.fit(X_train, y_train)  # train model
```

In [11]:
```
# kiem tra va du doan voi test data
y_pred = regr1.predict(X_test)
```

In [12]:
```
df = pd.DataFrame({'Actual': pd.DataFrame(y_test.values)[0].values,
                   'Prediction': pd.DataFrame(y_pred)[0].values})
df.head()
```

Out[12]:

	Actual	Prediction
0	6.0	5.107060
1	1.7	1.754246
2	4.7	4.436497
3	5.6	6.448185
4	3.9	4.212976

In [13]:
```
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(outputs, regr1.predict(inputs)))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regr1.score(inputs, outputs))
```

```
Mean squared error: 0.23
Variance score: 0.93
```

In [14]:
```
# Score = 93% => model fits with ~ 93% data => This is suitable model.
```

In [15]:
```
# Check the score of train and test
regr1.score(X_test, y_test) # test
```

Out[15]:
```
0.9458695247784931
```

In [16]:
```
regr1.score(X_train, y_train) # train
```

Out[16]:
```
0.9211926788648582
```

In [17]:
```
# Both training data and testing data have high score. => Choose this model.
```

In [18]:
```
# The coefficients
m=regr1.coef_[0] # chi co 1 m
b=regr1.intercept_
print('Coefficients: \n', m)
print('Interceft: \n', b)
```

```
Coefficients: 
 [2.23520927]
Interceft: 
 [1.08368322]
```

In [19]:
```
# Cung cap x => y_hat = mx + b => ket qua du doan
# hoac dung ten_model.predict(x) => y_hat
```

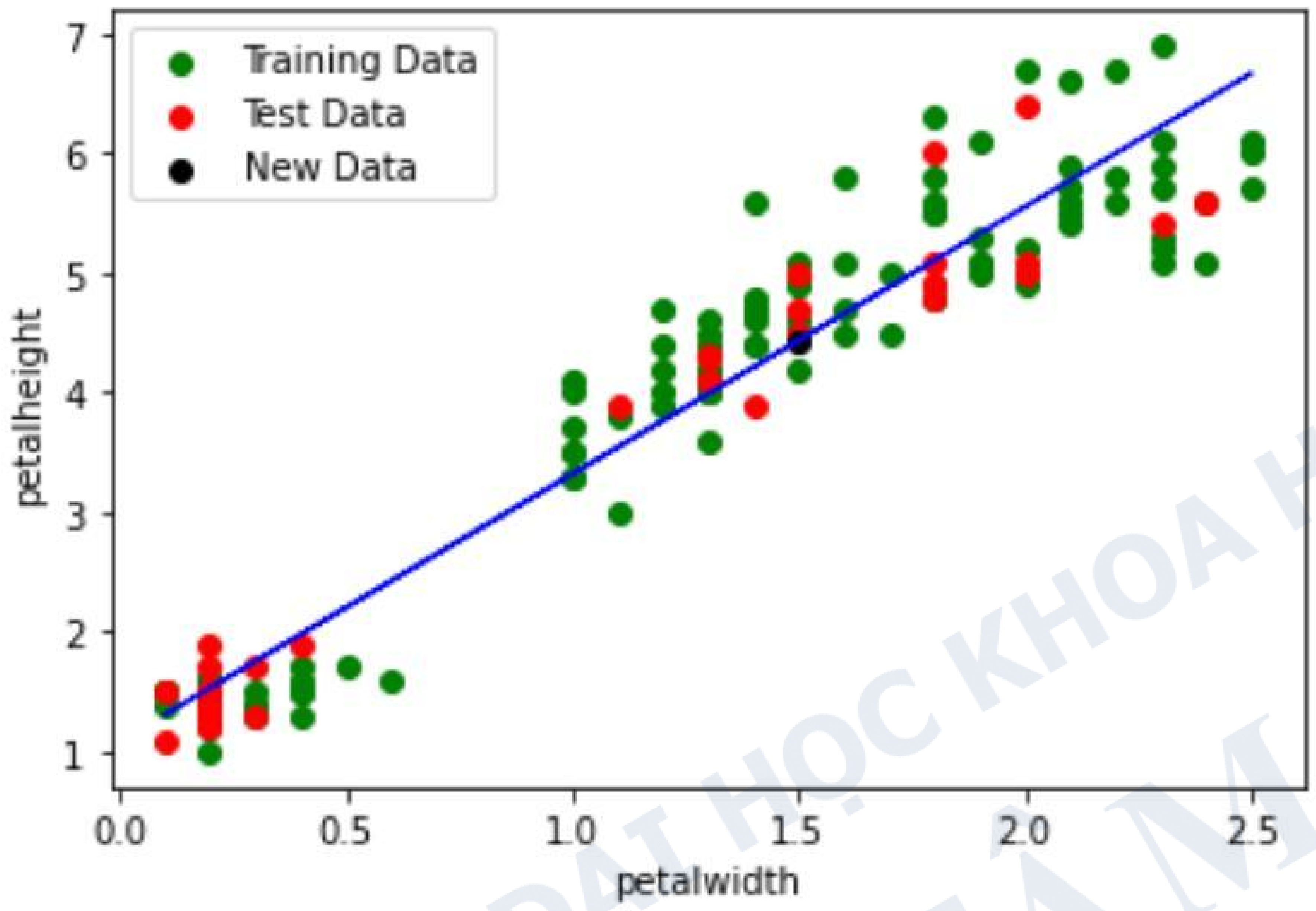
In [20]:
```
# reg_line = [(m* float(x)) + b for x in np.array(inputs)]
reg_line = regr1.predict(inputs)
```

In [21]:
```
x_now = [[1.5]]
y_now = regr1.predict(x_now)
print(y_now)
```

```
[[4.43649712]]
```

In [22]:
```python
# Plot outputs
plt.scatter(X_train, y_train,  color='green', label="Training Data")
plt.scatter(X_test, y_test,  color='red', label= "Test Data")
plt.scatter(x_now, y_now,  color='black', label= "New Data")
plt.plot(inputs,reg_line, color="blue", linewidth=1)

plt.xlabel("petalwidth")
plt.ylabel("petalheight")
plt.legend()
plt.show()
```


Training Data
Test Data
New Data
petalheight
petalwidth