Chapter 9 - Ex2: Iris - Multiple Linear Regression

Cho dữ liệu Iris.xls

Yêu cầu: Thực hiện linenear regression để từ sepallength, sepalwidth, petallength => dự đoán petalwidth

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 sepallength, sepalwidth, petallength (inputs) và petalwidth (outputs) với tỷ lệ dữ liệu test là 0.2
3. Áp dụng linrear regression
4. Vẽ hình. Nhận xét kết quả
5. Nếu sepallength, sepalwidth, petallength là 4.5, 3.1, 1.6 => petalwidth là bao nhiêu?
6. Áp dụng lựa chọn thuộc tính quan trọng cho model. Xây dựng lại model sau khi lựa chọn các thuộc tính quan trọng.

In [1]:
```
#!pip install pandas-profiling==2.7.1
```

In [2]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
# %cd '/content/gdrive/My Drive/MDS5_2022/Practice_2022/Chapter9/'
```

In [3]:
```
import pandas as pd
import pandas_profiling as pp
```

In [4]:
```
iris = pd.read_excel("Iris.xls")
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   sepallength  150 non-null    float64
 1   sepalwidth   150 non-null    float64
 2   petallength  150 non-null    float64
 3   petalwidth   150 non-null    float64
 4   iris         150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:
```python
iris.head()
```

Out[5]:

	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 [6]:
```python
profile = pp.ProfileReport(iris)
profile
```

```
Summarize dataset:                    19/? [00:05<00:00, 3.01it/s, Completed]
Generate report structure: 100%       1/1 [00:01<00:00, 1.34s/it]
Render HTML: 100%                     1/1 [00:00<00:00, 1.72it/s]
```

Overview

Overview   Reproduction   Warnings **4**

Dataset statistics

Number of variables	5
Number of observations	150
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	3
Duplicate rows (%)	2.0%
Total size in memory	15.1 KiB
Average record size in memory	103.2 B

Variable types

NUM	4
CAT	1

Variables

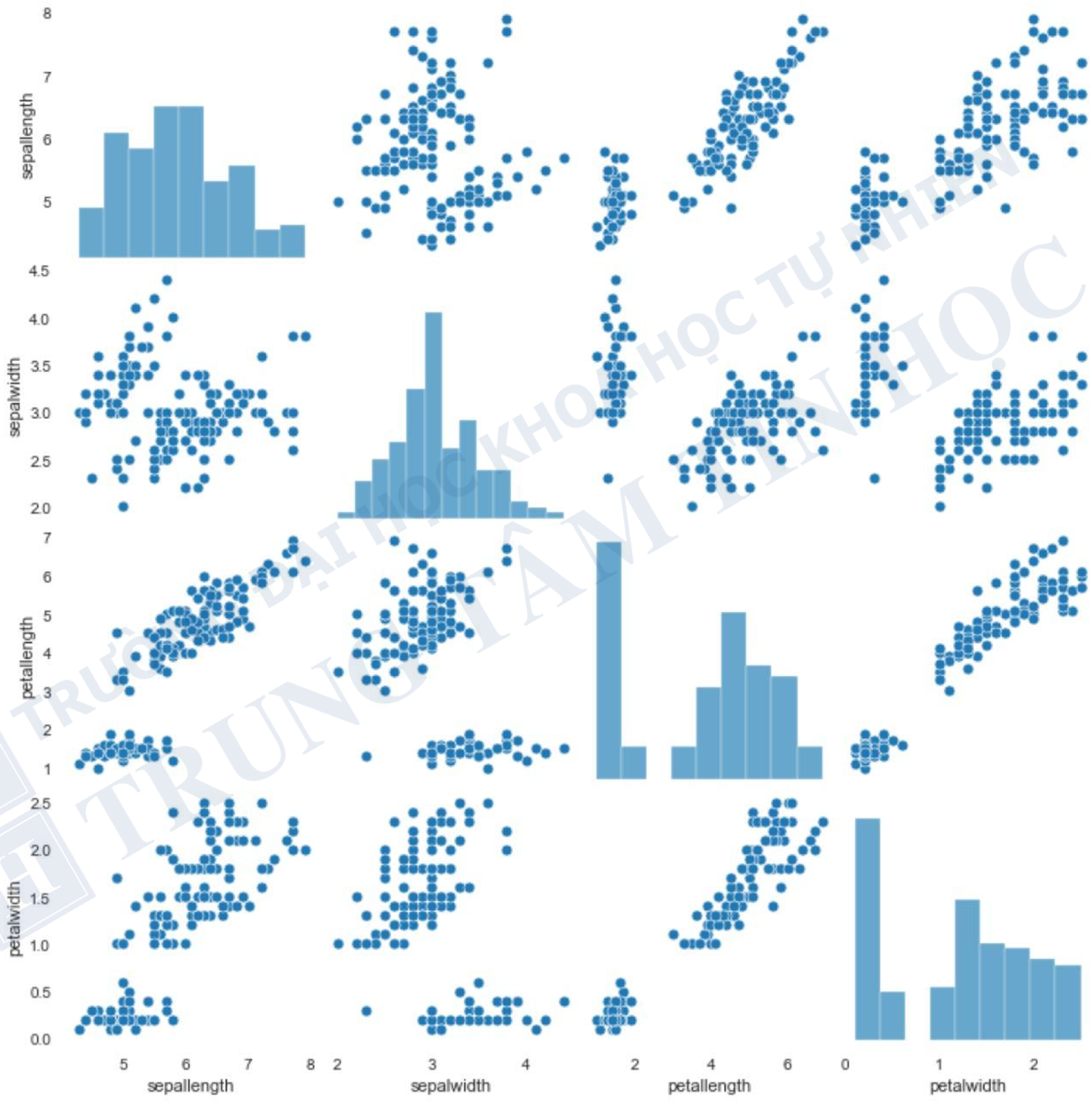
```
Out[6]:
```

```
In [7]: profile.to_file("output.html")
```

```
Export report to file:                                   1/1 [00:00<00:00,
100%                                                     43.59it/s]
```

```
In [8]: import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [9]: sns.pairplot(iris)
        plt.show()
```



In [10]:
```
inputs = iris[['sepallength','sepalwidth', 'petallength']]
inputs.head()
```

Out[10]:

	sepallength	sepalwidth	petallength
0	5.1	3.5	1.4
1	4.9	3.0	1.4
2	4.7	3.2	1.3
3	4.6	3.1	1.5
4	5.0	3.6	1.4

In [11]:
```
outputs = iris[['petalwidth']]
outputs.head()
```

Out[11]:

	petalwidth
0	0.2
1	0.2
2	0.2
3	0.2
4	0.2

In [12]:
```
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

In [13]:
```
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)

regr1 = linear_model.LinearRegression()
regr1 = regr1.fit(X_train, y_train)
```

In [14]:
```
y_pred = regr1.predict(X_test)
```

In [15]:
```
# 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: 0.04
Variance score: 0.94
```

In [16]:
```
# Score = 94% => model fits with ~ 94% data => This is suitable model.
```

```
In [17]: print('Variance score: %.2f' % r2_score(y_test, y_pred)) # y real, y predict
```

```
Variance score: 0.94
```

```
In [18]: # Check the score of train and test
```

```
In [19]: regr1.score(X_train, y_train)
```

```
Out[19]: 0.9357294091323384
```

```
In [20]: regr1.score(X_test, y_test)
```

```
Out[20]: 0.9436762270965792
```

```
In [21]: # Both training data and testing data have high score.
         # => Choose this model.
```

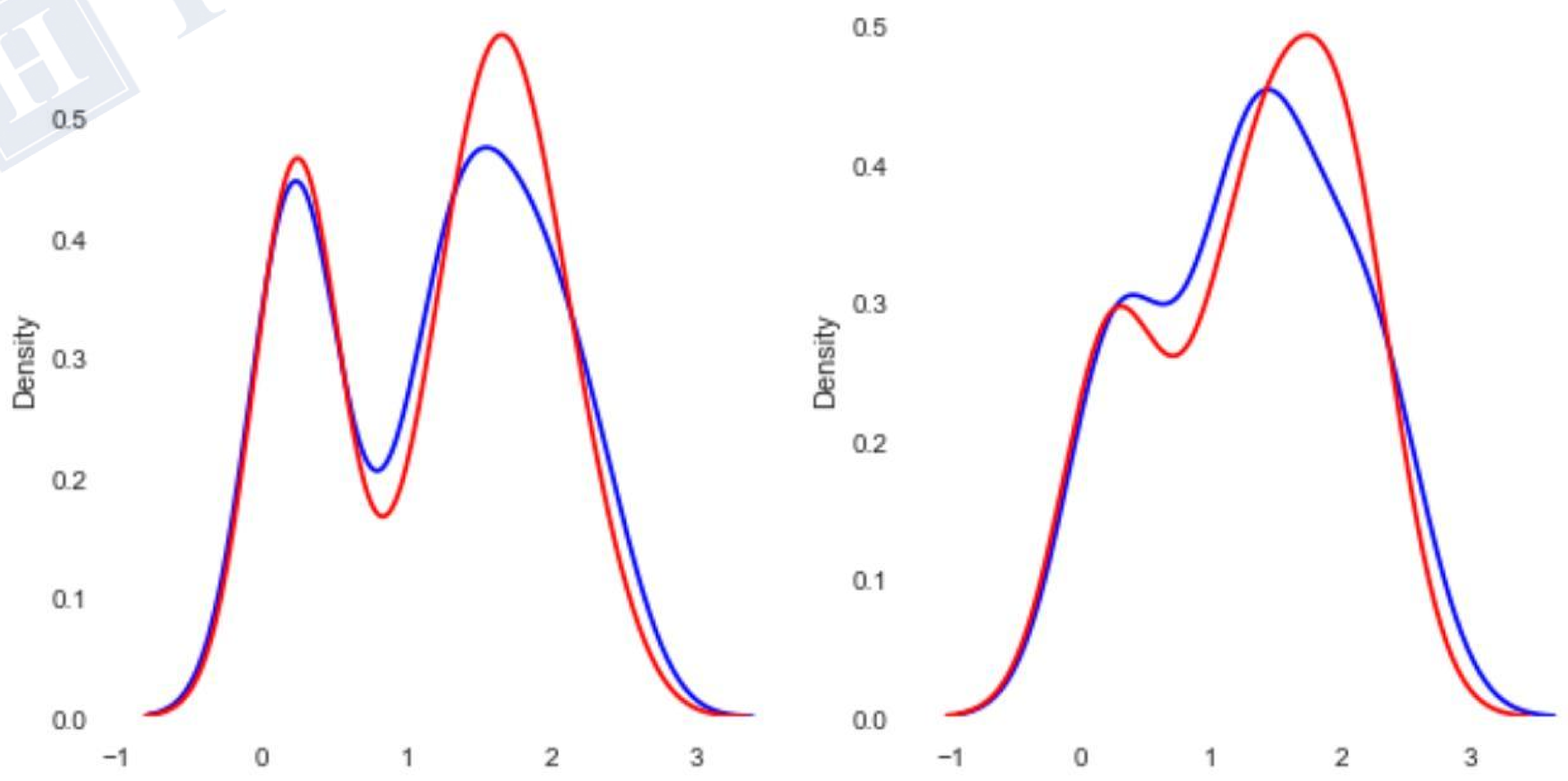
```
In [22]: # The coefficients
         m=regr1.coef_
         b=regr1.intercept_
         print('Coefficients: \n', m)
         print('Interceft: \n', b)
```

```
Coefficients: 
 [[-0.16565783  0.17339064  0.50356191]]
Interceft: 
 [-0.25647234]
```

```
In [23]: # Visualization
         y_train_hat = regr1.predict(X_train)
         y_test_hat = regr1.predict(X_test)
```

In [24]:
```python
plt.figure(figsize=(10,5))
plt.subplot(1, 2, 1)
ax1 = sns.distplot(y_train, hist=False, color="b", label='Train Actual')
sns.distplot(y_train_hat, hist=False, color="r", label='Train Predict', ax=ax1)
plt.subplot(1,2,2)
ax2 = sns.distplot(y_test, hist=False, color="b", label='Test Actual')
sns.distplot(y_test_hat, hist=False, color="r", label='Test Predict', ax=ax2)
plt.show()
```

```
c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: Futu
reWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level functio
n with similar flexibility) or `kdeplot` (an axes-level function for kernel den
sity plots).
  warnings.warn(msg, FutureWarning)
c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: Futu
reWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level functio
n with similar flexibility) or `kdeplot` (an axes-level function for kernel den
sity plots).
  warnings.warn(msg, FutureWarning)
c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: Futu
reWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level functio
n with similar flexibility) or `kdeplot` (an axes-level function for kernel den
sity plots).
  warnings.warn(msg, FutureWarning)
c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: Futu
reWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level functio
n with similar flexibility) or `kdeplot` (an axes-level function for kernel den
sity plots).
  warnings.warn(msg, FutureWarning)
```



In [25]:
```
# Make new prediction
x_now = [[4.5, 3.1, 1.6]]
y_now = regr1.predict(x_now)
print(y_now)
```

```
[[0.34127746]]
```

Select important features

Solution 1: SelectKBest

In [26]:
```
# Univariate Selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
```

In [27]:
```
# Apply SelectKBest class to extract all best features
bestfeatures = SelectKBest(score_func=f_regression, k='all')
fit = bestfeatures.fit(inputs,outputs)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(inputs.columns)
```

```
c:\program files\python36\lib\site-packages\sklearn\utils\validation.py:73: Dat
aConversionWarning: 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().
  return f(**kwargs)
```

In [28]:
```
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  # naming the dataframe columns
print(featureScores.nlargest(3,'Score'))  # print 3 best features
```

```
         Specs        Score
2  petallength  1876.657813
0  sepallength   299.194957
1   sepalwidth    21.554378
```

In [29]:
```
# 2 features have highest scores
X_now = inputs[['petallength', 'sepallength']]
```

In [30]:
```
X_train_n, X_test_n, y_train_n, y_test_n = train_test_split(X_now, outputs,
                                                            test_size=0.20)

regr_n = linear_model.LinearRegression()
regr_n = regr1.fit(X_train_n, y_train_n)
```

In [31]:
```
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(outputs, regr_n.predict(X_now)))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regr_n.score(X_now, outputs))
```

```
Mean squared error: 0.04
Variance score: 0.93
```

In [32]:
```
print("Train's score:", regr_n.score(X_train_n, y_train_n))
```

```
Train's score: 0.9318674342068523
```

In [33]:
```
print("Test's score:", regr_n.score(X_test_n, y_test_n))
```

```
Test's score: 0.9115055217267548
```

Solution 2: Correlation Matrix with Heatmap

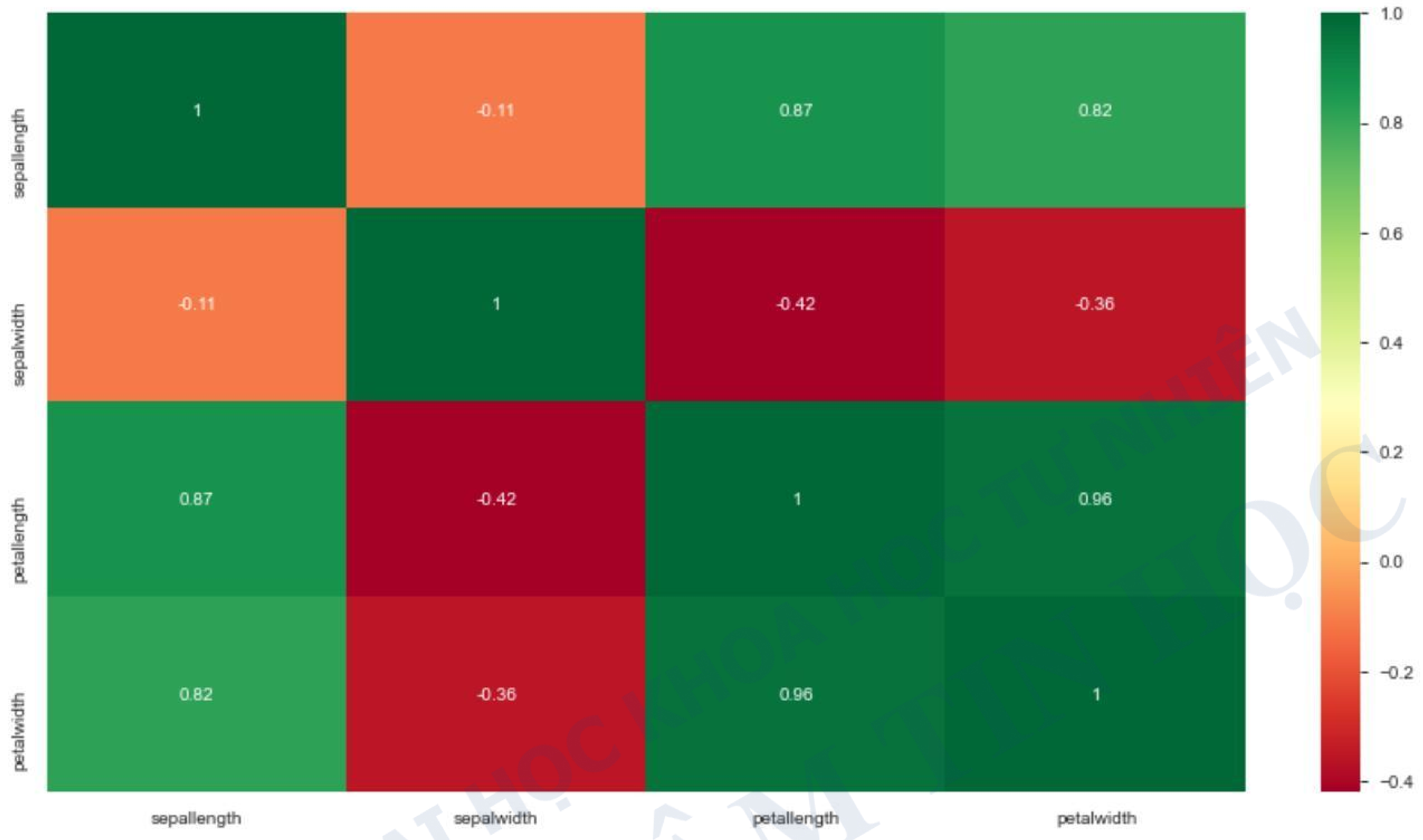
In [34]:
```
# get correlations of each features in dataset
data_sub = iris.iloc[:,0:4]
corrmat = data_sub.corr()
top_corr_features = corrmat.index
```

In [35]:
```
data_sub.corr()
```

Out[35]:

	sepallength	sepalwidth	petallength	petalwidth
sepallength	1.000000	-0.109369	0.871754	0.817954
sepalwidth	-0.109369	1.000000	-0.420516	-0.356544
petallength	0.871754	-0.420516	1.000000	0.962757
petalwidth	0.817954	-0.356544	0.962757	1.000000

In [36]:
```
plt.figure(figsize=(15,8))
#plot heat map
g=sns.heatmap(data_sub[top_corr_features].corr(),cmap="RdYlGn", annot=True) # ann
```



In [ ]:
```
```