iris

August 20, 2019

1 Import necessary packages

```
[2]: import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     from bokeh.io import show, output_file
     from bokeh.plotting import figure
     from bokeh.models import HoverTool, ColumnDataSource, CategoricalColorMapper
     from bokeh.layouts import row, column
     from bokeh.models.widgets import Tabs, Panel
     from sklearn.linear_model import LinearRegression
```

```
[2]: pwd
```

```
[2]: 'C:\\Users\\Lera'
```

2 Create pandas dataframe from file

```
[3]: df = pd.read_csv(r'C:\\Users\\Lera\Documents\iris.csv', names =␣
      ↪['sepal_length','sepal_width','petal_length','petal_width','class'])
     print(df.head())
```

```
   sepal_length  sepal_width  petal_length  petal_width        class
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
```

3 Clean data

```
[4]: df['class'] = df['class'].replace({"Iris-": ""}, regex = True)
     df['class'] = df['class'].astype('category')
     print(df.head())
```

```
   sepal_length  sepal_width  petal_length  petal_width   class
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

4 Create new dataframes for each iris class

[5]:
```
setosa = df[df['class'] == 'setosa']
virginica = df[df['class'] == 'virginica']
versicolor = df[df['class'] == 'versicolor']
```

[6]:
```
flowers = [setosa, virginica, versicolor]
```

5 Create for loop to calculate correlation coefficients for sepal length and sepal width of each iris class

[7]:
```
flowers = [setosa, virginica, versicolor]
for f in flowers:
    r = np.corrcoef(f['sepal_length'], f['sepal_width'])
    print(r[0,1])
```

```
0.7467803732639267
0.4572278163941129
0.5259107172828243
```

6 Create for loop to calculate correlation coefficients for petal length and petal width of each iris class

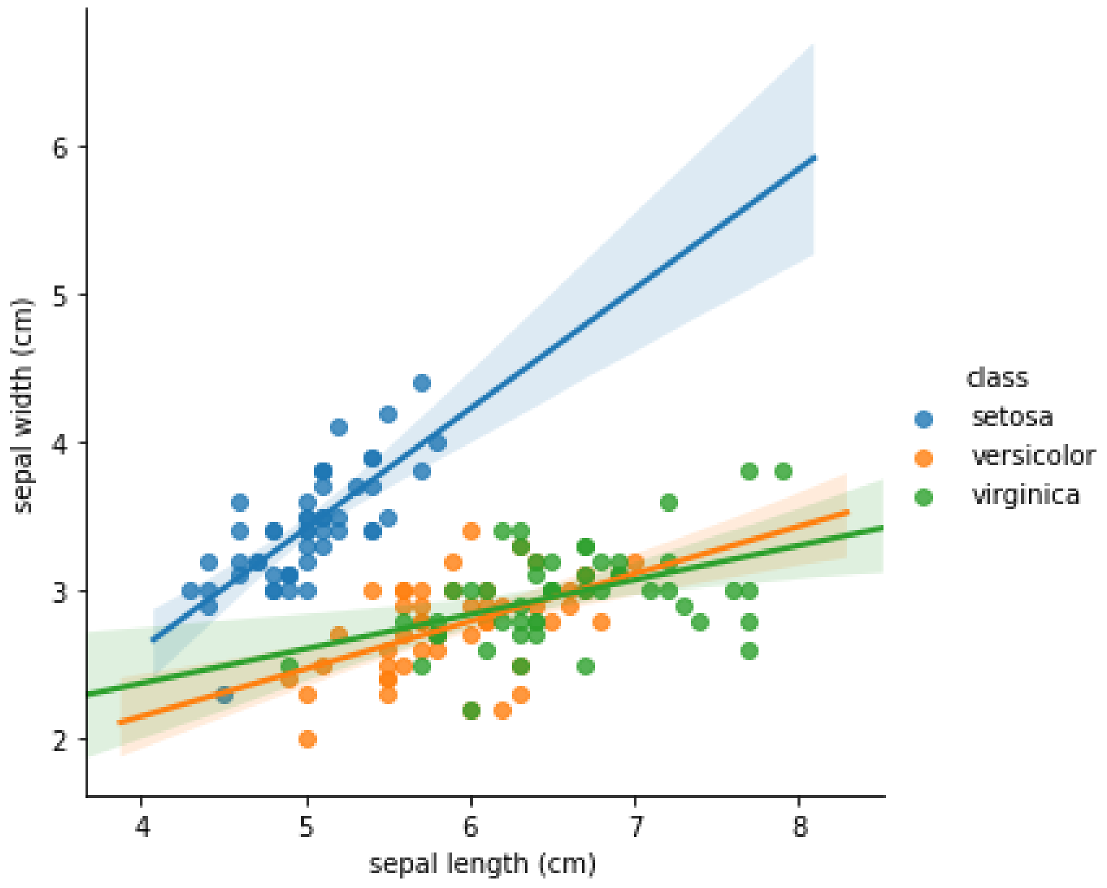
[8]:
```
for f in flowers:
    r = np.corrcoef(f['petal_length'], f['petal_width'])
    print(r[0,1])
```

```
0.30630821115803575
0.3221082159003183
0.7866680885228169
```

7 Create basic least squares line plots using seaborn

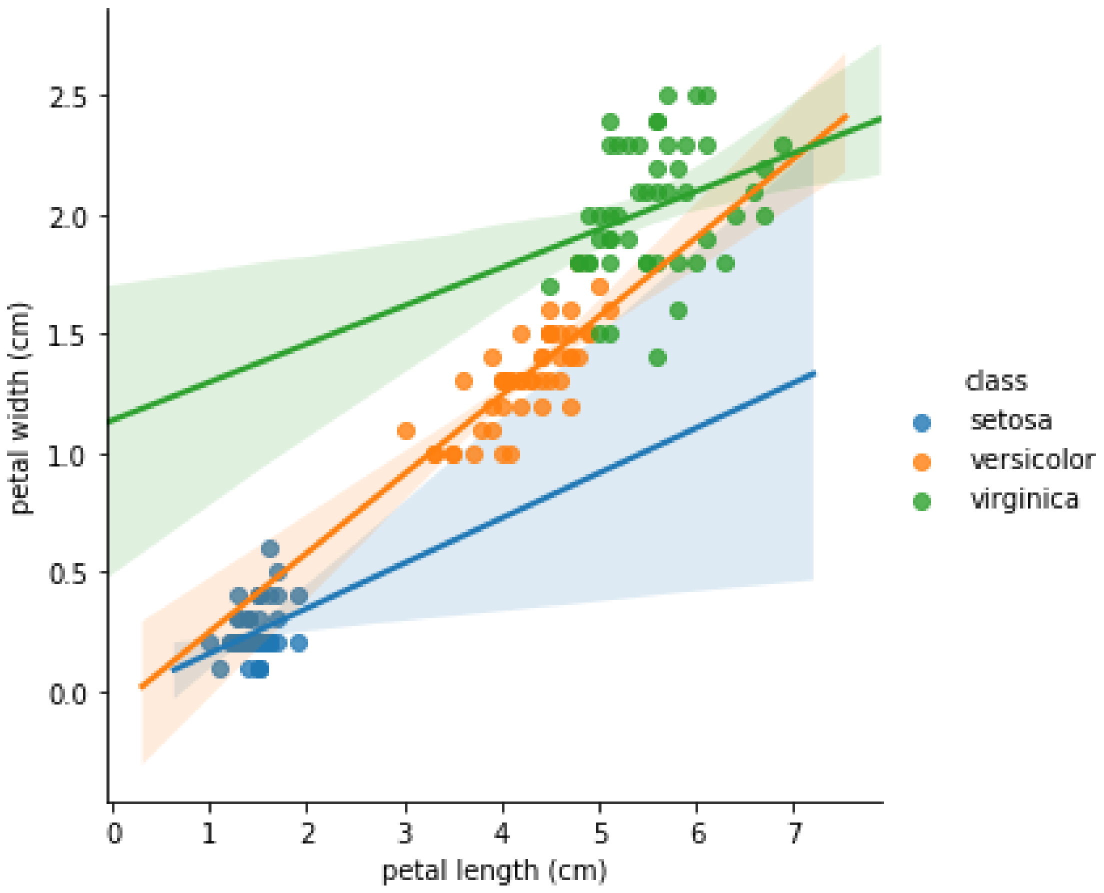
[9]:
```
sns.lmplot(x = 'sepal_length', y = 'sepal_width', hue = 'class', data = df)
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.show()
```

```
C:\Users\Lera\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will
be interpreted as an array index, `arr[np.array(seq)]`, which will result either
in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



[10]:
```
sns.lmplot(x = 'petal_length', y = 'petal_width', hue = 'class', data = df)
plt.xlabel('petal length (cm)')
plt.ylabel('petal width (cm)')

plt.show()
```



8 Calculate slope and intercept for each iris class

[11]:
```
def linreg(x,y):
    slope, intercept = np.polyfit(x,y,1)
    return slope, intercept
```

[12]:
```
m_set, int_set = linreg(setosa['sepal_length'], setosa['sepal_width'])
m_virg, int_virg = linreg(virginica['sepal_length'], virginica['sepal_width'])
m_vers, int_vers = linreg(versicolor['sepal_length'], versicolor['sepal_width'])
```

[13]:
```
x_set = np.array([4,8])
x_virg = np.array([4,8])
x_vers = np.array([4,8])
```

[14]:
```
p_set = np.array([0,7])
p_virg = np.array([0,7])
p_vers = np.array([0,7])
```

[15]:
```
m1_set, int1_set = linreg(setosa['petal_length'], setosa['petal_width'])
m1_virg, int1_virg = linreg(virginica['petal_length'], virginica['petal_width'])
m1_vers, int1_vers = linreg(versicolor['petal_length'],␣
 ↪versicolor['petal_width'])
```

9 Calculate predicted variables

[16]:
```
def pred(x, slope, intercept):
    y = x * slope + intercept
    return y
```

[17]:
```
set_pred = pred(x_set, m_set, int_set)
virg_pred = pred(x_virg, m_virg, int_virg)
vers_pred = pred(x_vers, m_vers, int_vers)
```

[18]:
```
pset_pred = pred(p_set, m1_set, int1_set)
pvirg_pred = pred(p_virg, m1_virg, int1_virg)
pvers_pred = pred(p_vers, m1_vers, int1_vers)
```

10 Create interactive bokeh plots with found slopes, intercepts, and predicted variables

[19]:
```
source = ColumnDataSource(df)
```

[20]:
```
hover1 = HoverTool(tooltips = [('species name','@class'), ('sepal␣
 ↪length','@sepal_length cm'), ('sepal width', '@sepal_width cm')])

plot1 = figure(x_axis_label = 'sepal length (cm)', y_axis_label = 'sepal width␣
 ↪(cm)', title = 'Sepal Length vs. Sepal Width',tools = [hover1])

mapper1 = CategoricalColorMapper(factors = ['setosa', 'virginica',␣
 ↪'versicolor'], palette = ['red', 'green', 'blue'])

plot1.circle('sepal_length', 'sepal_width', source = source, color␣
 ↪=dict(field='class', transform = mapper1), fill_alpha = 0.5, legend =␣
 ↪'class')
plot1.line(x_set, set_pred, color = 'red')
plot1.line(x_virg, virg_pred, color = 'green')
plot1.line(x_vers, vers_pred, color = 'blue')

output_file('plot1.html')
```

[21]:
```
hover2 = HoverTool(tooltips = [('species name', '@class'), ('petal length',␣
 ↪'@petal_length cm'), ('petal width','@petal_width cm')])
```

```
mapper2 = CategoricalColorMapper(factors = ['setosa', 'virginica',␣
↪'versicolor'], palette = ['magenta', 'turquoise', 'cornflowerblue'])

plot2 = figure(x_axis_label = 'petal length (cm)', y_axis_label = 'petal width␣
↪(cm)', title = 'Petal Length vs. Petal Width', tools = [hover2])
plot2.circle('petal_length', 'petal_width', source = source, color = dict(field␣
↪= 'class', transform = mapper2), fill_alpha = 0.5, legend = 'class')
plot2.line(p_set, pset_pred, color = 'magenta')
plot2.line(p_virg, pvirg_pred, color = 'turquoise')
plot2.line(p_vers, pvers_pred, color = 'cornflowerblue')
plot2.legend.location = 'bottom_right'
output_file('plot2.html')
```

[22]:
```
first = Panel(child = plot1, title = 'Sepal')
second = Panel(child = plot2, title = 'Petal')
```

[23]:
```
tabs = Tabs(tabs = [first, second])

show(tabs)
```

11 Carry out statistical analysis of the correlation coefficient

[24]:
```
# Ho: r(virginica sepal) = 0
# Ha: r(virginica sepal) =! 0
```

[25]:
```
r_virginica = np.corrcoef(virginica['sepal_length'],␣
↪virginica['sepal_width'])[0,1]
print(r_virginica)
```

```
0.4572278163941129
```

[26]:
```
perm_replicates = np.empty(10000)
```

[27]:
```
for i in range (10000):
    virg_permuted = np.random.permutation(virginica['sepal_length'])
    perm_replicates[i] = np.corrcoef(virg_permuted,␣
↪virginica['sepal_width'])[0,1]
```

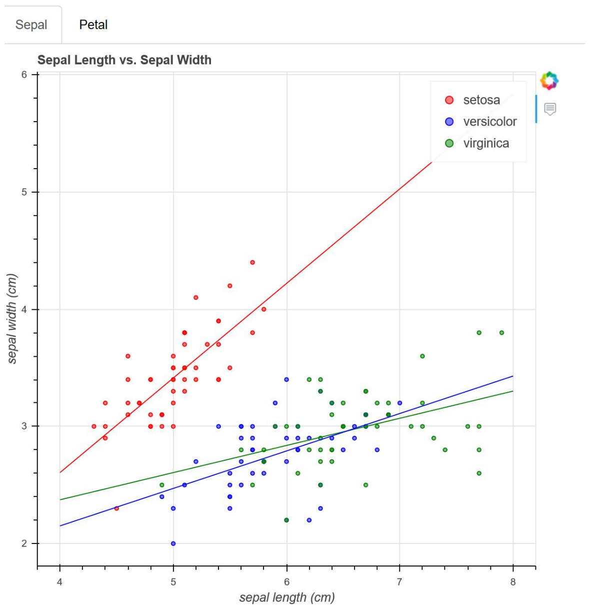
[28]:
```
p = np.sum(perm_replicates >= r_virginica)/len(perm_replicates)
print(p)
```

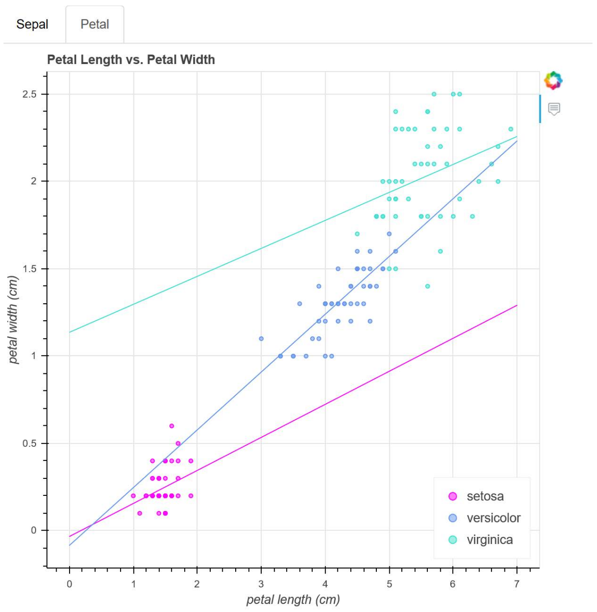
```
0.0004
```

[29]:
```
#Due to the p-value being lower than 0.05, we reject the null hypothesis. This␣
↪means that there is believed to be substantial
```

```
#statistical evidence that there is correlation between the sepal length and␣
↪sepal width in virginica iris flowers.
```

Bokeh output


Sepal
Petal
Sepal Length vs. Sepal Width
setosa
versicolor
virginica
sepal width (cm)
sepal length (cm)


Sepal
Petal
Petal Length vs. Petal Width
setosa
versicolor
virginica
petal width (cm)
petal length (cm)