project_0

March 12, 2025

0.1 Project #0 Introduction

0.1.1 Project Outline

1. Define Problem

2. Data Preparation

3. Explanatory Data Analysis (EDA)

4. Data Pre-processing

5. Modeling

6. Model Evaluation

6.5 Model Modification

7. Conclusion

[29]:
```
# packages for data manipulation
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
```

0.1.2 1. Problem Defination

predict what the species of a new data

0.1.3 2. Data Preparation: skip

[30]:
```
iris = sns.load_dataset('iris')
iris.head()
```

[30]:

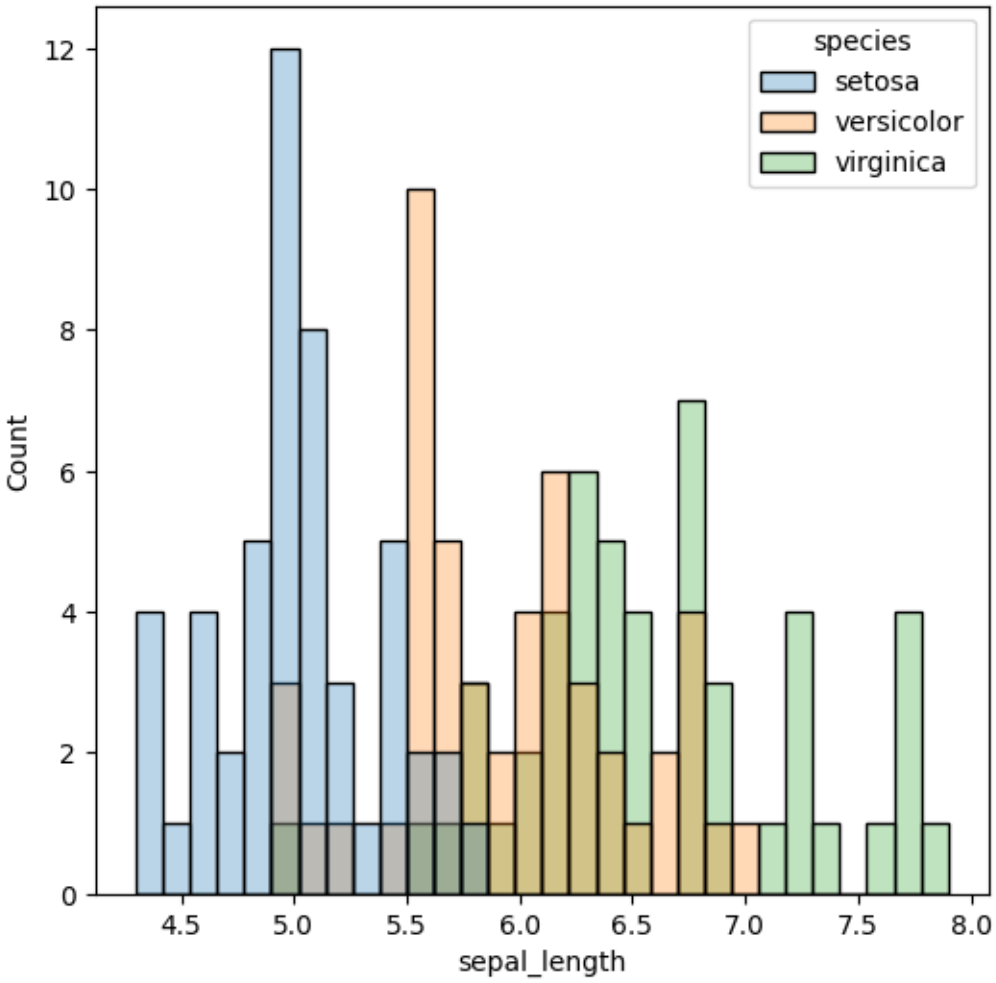
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa

3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

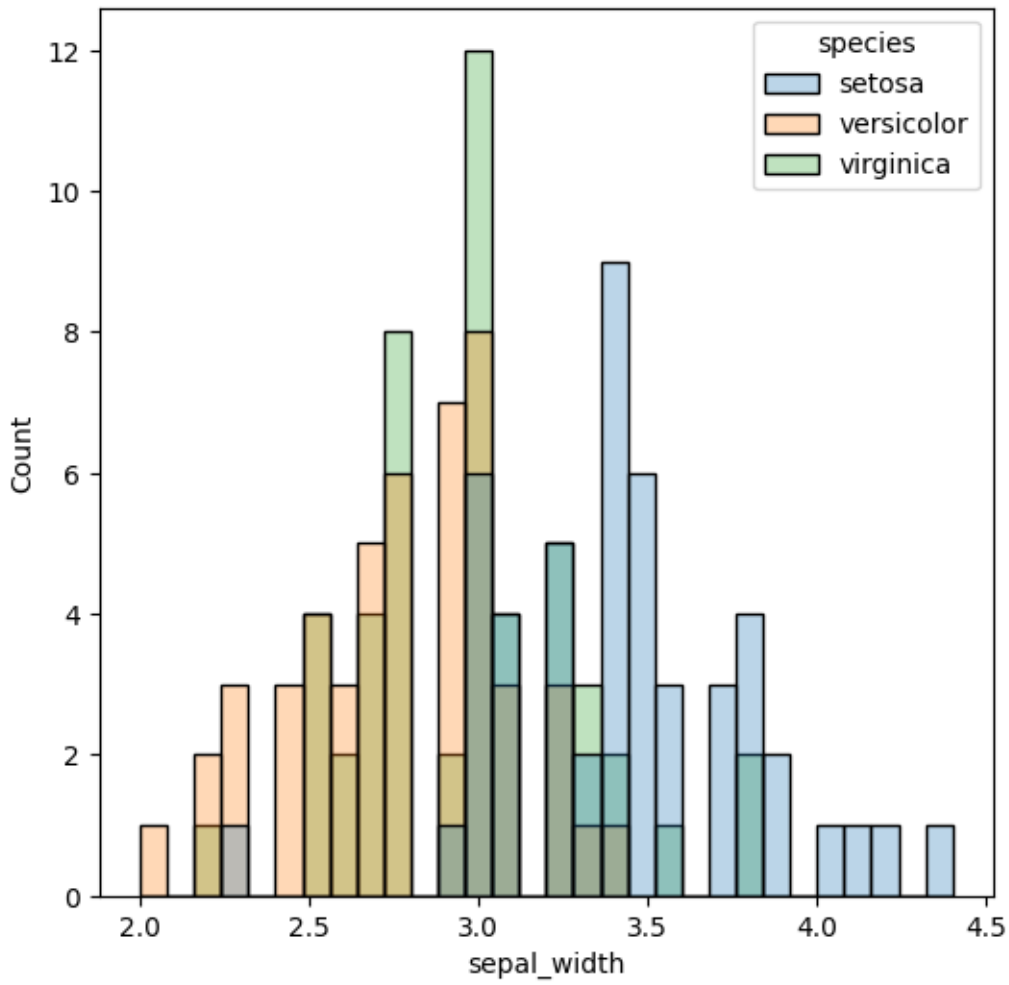
0.1.4 3. EDA

[31]:
```
def hist_graphing(variable, bin):
    plt.figure(figsize=(6,6))
    sns.histplot(data = iris, x = variable, bins = bin, hue = 'species', alpha␣
 ↪= 0.3)
    plt.show()
```

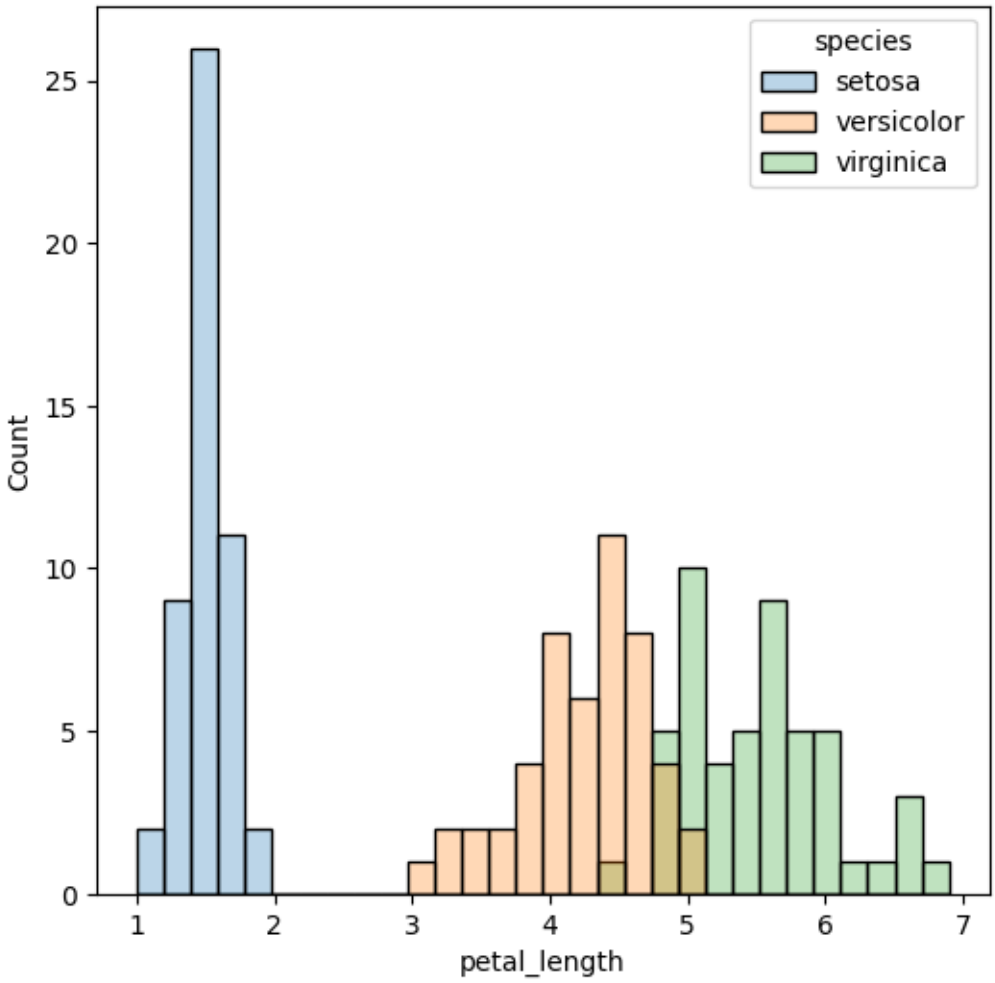
[32]:
```
hist_graphing('sepal_length', 30)
```



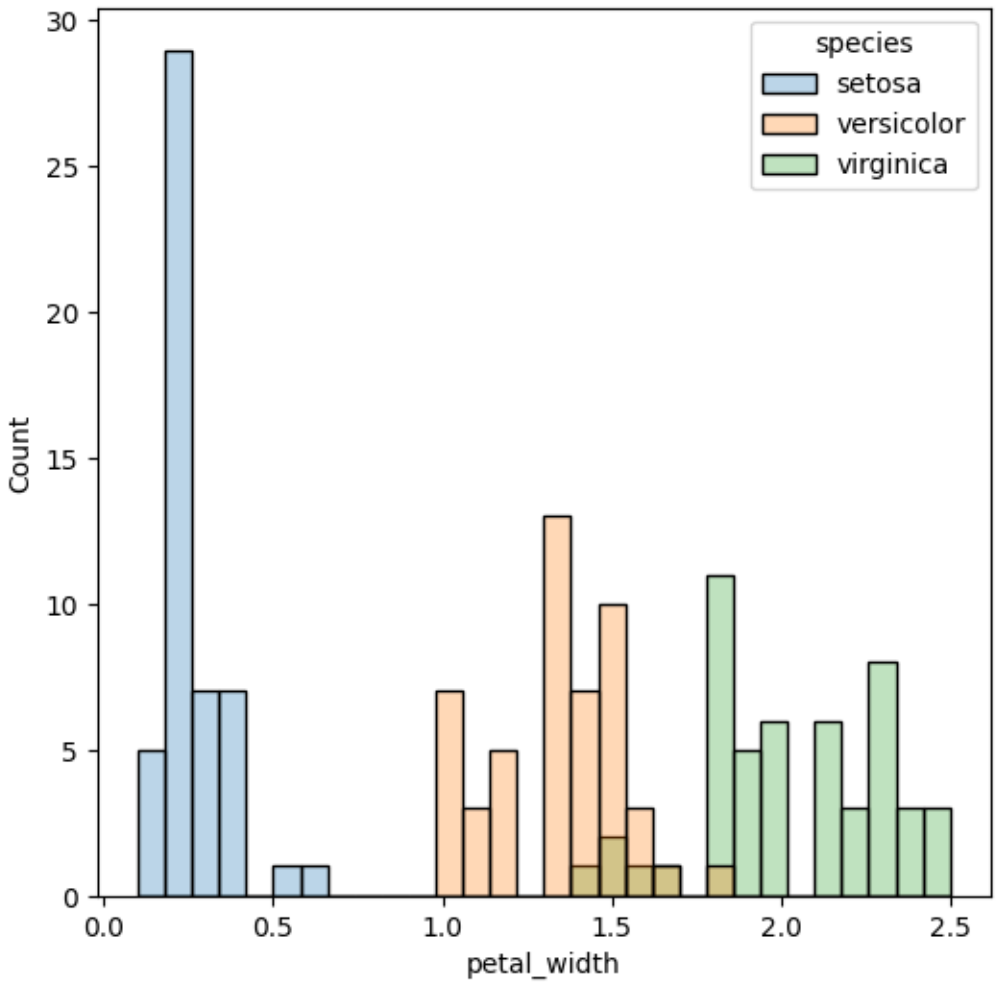
[33]:
```
hist_graphing('sepal_width', 30)
```



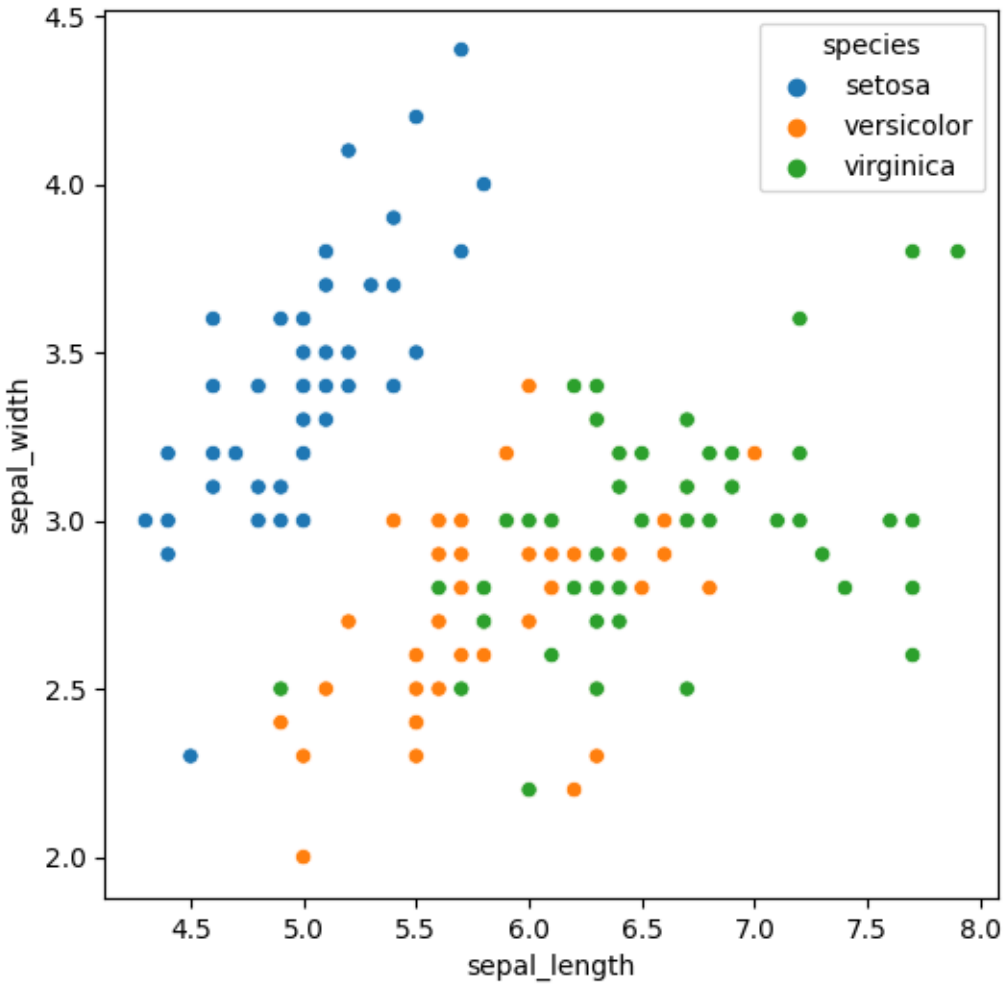
[34]:
```
hist_graphing('petal_length', 30)
```



[35]:
```
hist_graphing('petal_width', 30)
```



[36]:
```
plt.figure(figsize=(6,6))
sns.scatterplot(data = iris, x = 'sepal_length', y = 'sepal_width', hue =␣
↪'species')
plt.show()
```



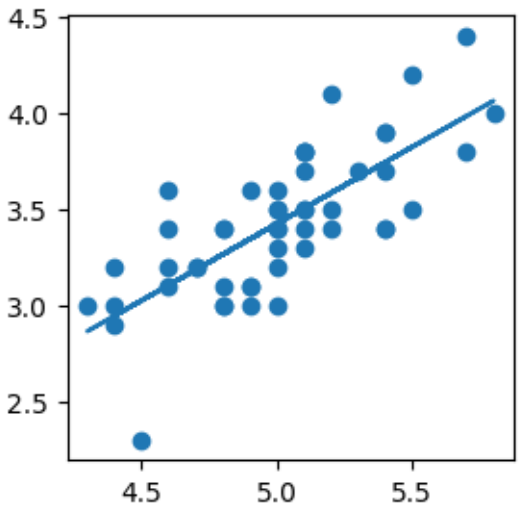
[37]:
```
def scatter_line(df, x, y, figsize):
    p1 = df[x]
    p2 = df[y]

    plt.figure(figsize=(figsize,figsize))
    plt.scatter(data = df, x = x, y = y)

    z = np.polyfit(p1, p2, 1)
    p = np.poly1d(z)
    plt.plot(p1, p(p1))

    # plt.title(df)
    # plt.xlabel(x)
    # plt.ylabel(y)

    plt.show()
```

[38]:
```
iris_setosa = iris[iris.species == 'setosa']
iris_versicolor = iris[iris.species == 'versicolor']
iris_virginica = iris[iris.species == 'virginica']
```

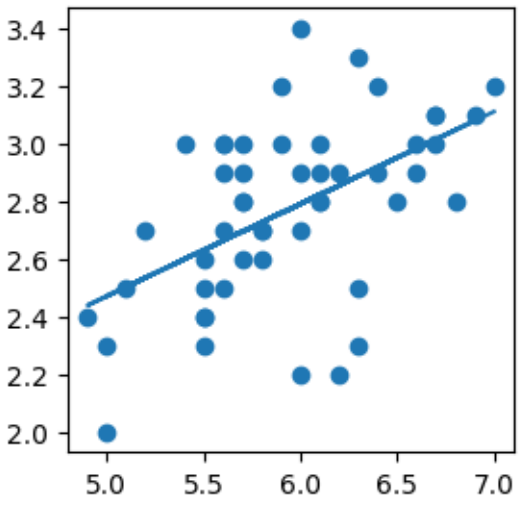
[39]:
```
iris_species = [iris_setosa, iris_versicolor, iris_virginica]
species = ['setosa', 'versicolor', 'virginica']
```

[40]:
```
for i in range(len(iris_species)):
    print(f"{species[i]} correlation visual")
    scatter_line(iris_species[i], 'sepal_length', 'sepal_width', 3)
```

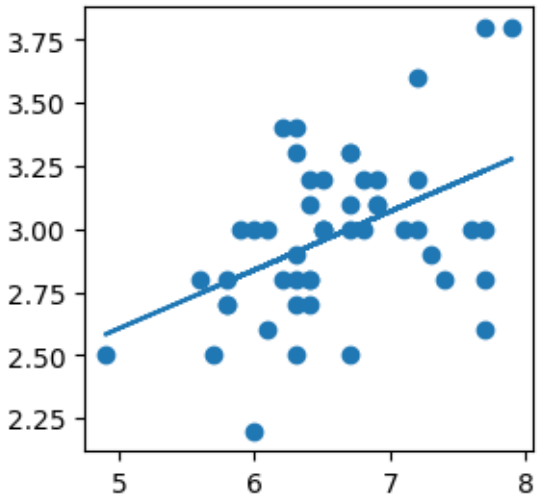
```
setosa correlation visual
```




```
versicolor correlation visual
```

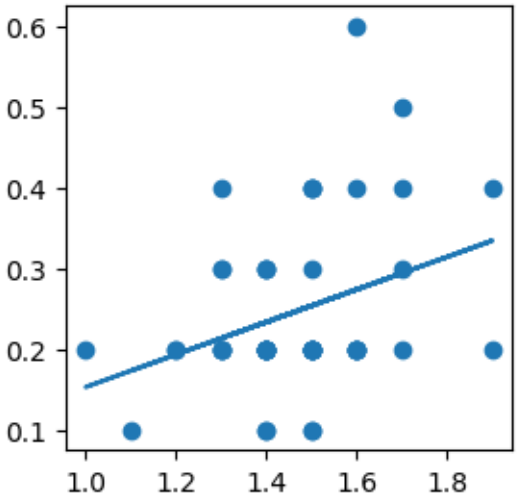


virginica correlation visual

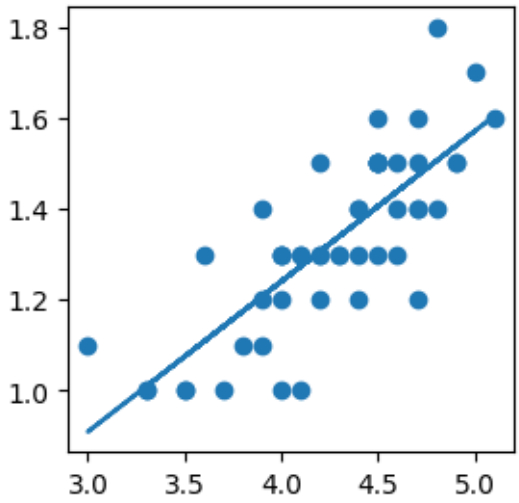


[41]:
```
for i in range(len(iris_species)):
    print(f"{species[i]} correlation visual")
    scatter_line(iris_species[i], 'petal_length', 'petal_width', 3)
```

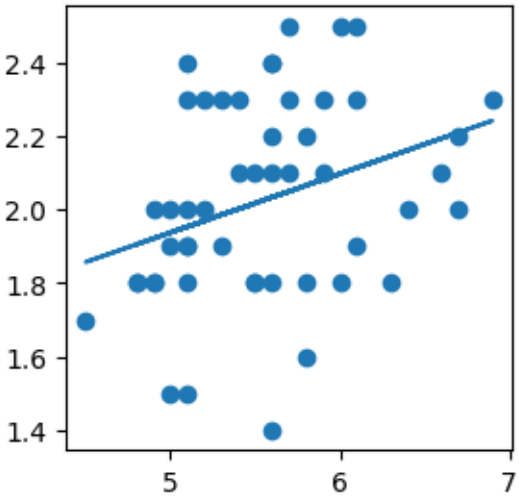
setosa correlation visual



versicolor correlation visual



virginica correlation visual



0.1.5 4. Data Pre-Processing

split data into training and test sets

[42]:
```
#functions required for data-preprocessing and model building
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
```

[43]:
```
explanatory = iris.drop(columns = ['species']) #explanatory variable
response = iris.species #response variable
```

summary statistics; normalization not required

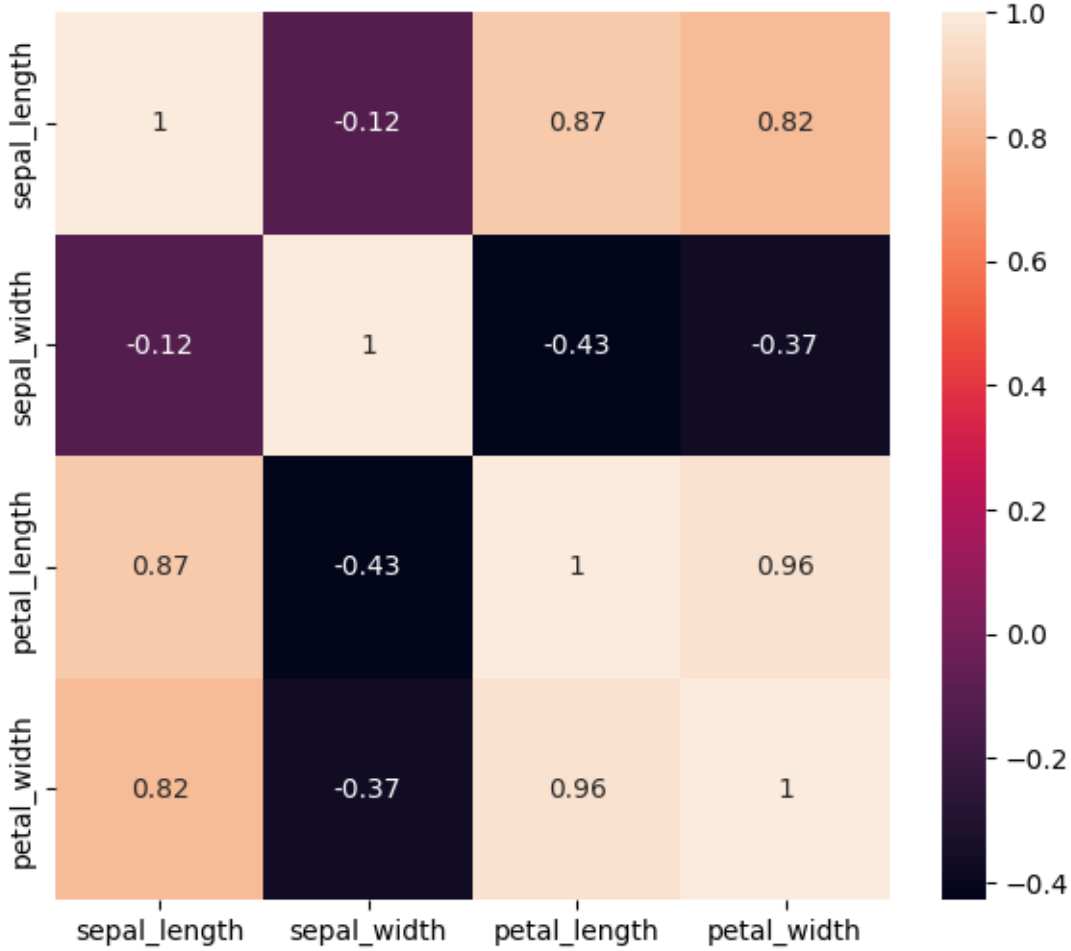
[44]:
```
explanatory.describe() #summary statistics
```

[44]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Observe the correlation between explanatory varialbes

[45]:
```
plt.figure(figsize=(7,6))
sns.heatmap(explanatory.corr(), annot=True)
plt.show()
```



Perform PCA to handle multicollinearity

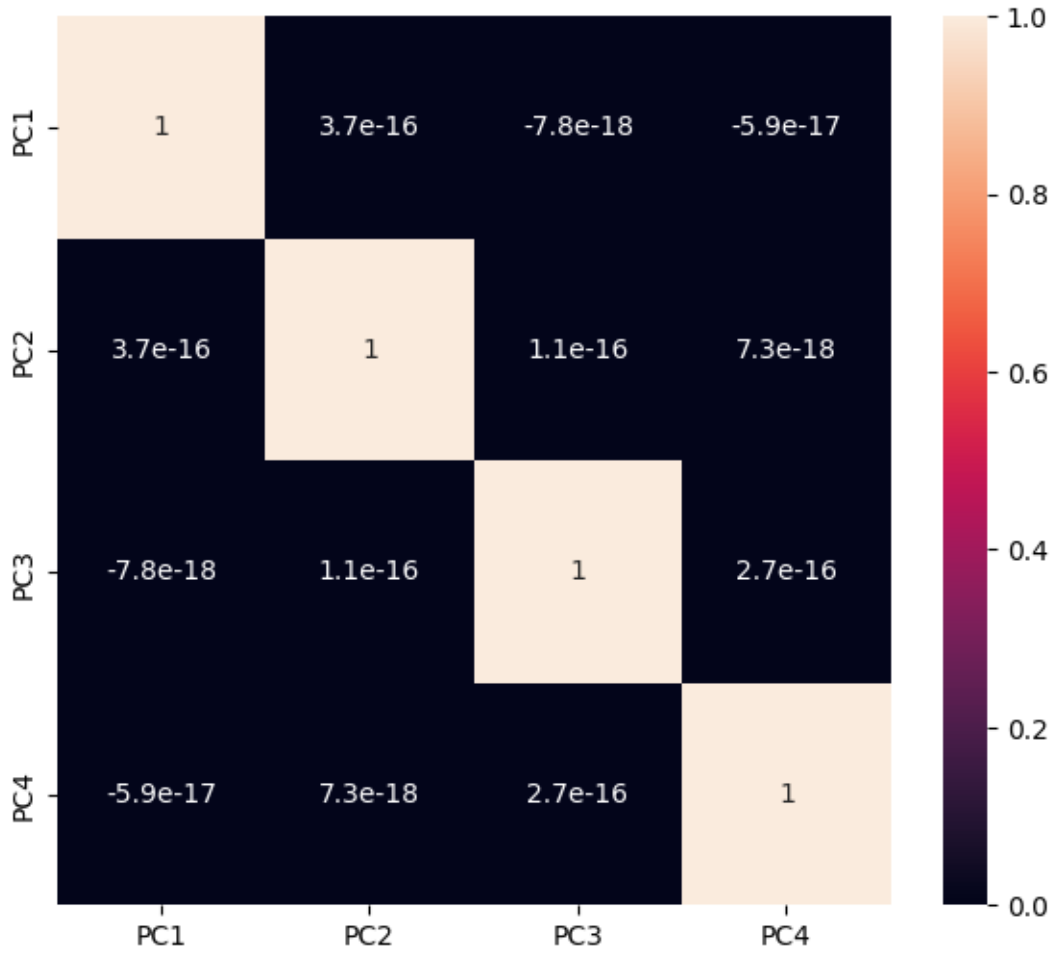
[46]:
```
pca = PCA(random_state = 2000, n_components = None)
explanatory_pca = pca.fit_transform(explanatory) # apply PCA to explanatory

explanatory_pca = pd.DataFrame(explanatory_pca)
explanatory_pca = explanatory_pca.rename(columns={0: "PC1", 1: "PC2", 2: "PC3",␣
↪3: "PC4"}) # set the column names

explanatory_pca.head()
```

[46]:

	PC1	PC2	PC3	PC4
0	-2.684126	0.319397	-0.027915	-0.002262
1	-2.714142	-0.177001	-0.210464	-0.099027
2	-2.888991	-0.144949	0.017900	-0.019968
3	-2.745343	-0.318299	0.031559	0.075576
4	-2.728717	0.326755	0.090079	0.061259

[47]:
```
plt.figure(figsize=(7,6))
sns.heatmap(explanatory_pca.corr(),annot=True)
```

[47]: <Axes: >



Prepare for the model training; split data

[48]:
```
x = explanatory_pca
y = response

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

0.1.6 5. Modling

[49]:
```
from sklearn.tree import DecisionTreeClassifier
```

[50]:
```
model = DecisionTreeClassifier() #define the model

model.fit(x_train, y_train)
```

[50]:
```
DecisionTreeClassifier()
```

0.1.7 6. Evaluation

Make Prediction

[51]:
```
y_train_pred = model.predict(x_train)
y_test_pred = model.predict(x_test)
```

Evaluating

[52]:
```
result = pd.DataFrame({'Actual': y_test, 'Predicted': y_test_pred})
result["result"] = np.where(result.Actual == result.Predicted, "Correct",␣
 ↪"Incorrect")
result
```

[52]:

	Actual	Predicted	result
140	virginica	virginica	Correct
83	versicolor	virginica	Incorrect
74	versicolor	versicolor	Correct
126	virginica	virginica	Correct
34	setosa	setosa	Correct
85	versicolor	versicolor	Correct
108	virginica	virginica	Correct
47	setosa	setosa	Correct
24	setosa	setosa	Correct
131	virginica	virginica	Correct
54	versicolor	versicolor	Correct
120	virginica	virginica	Correct
124	virginica	virginica	Correct
109	virginica	virginica	Correct
40	setosa	setosa	Correct
51	versicolor	versicolor	Correct
29	setosa	setosa	Correct
48	setosa	setosa	Correct
118	virginica	versicolor	Incorrect
136	virginica	virginica	Correct
21	setosa	setosa	Correct
27	setosa	setosa	Correct
32	setosa	setosa	Correct
44	setosa	setosa	Correct
49	setosa	setosa	Correct
104	virginica	virginica	Correct
117	virginica	virginica	Correct
11	setosa	setosa	Correct
132	virginica	virginica	Correct
59	versicolor	versicolor	Correct
105	virginica	virginica	Correct
33	setosa	setosa	Correct
14	setosa	setosa	Correct
36	setosa	setosa	Correct

```
130   virginica  versicolor  Incorrect
82   versicolor  versicolor    Correct
86   versicolor   virginica  Incorrect
99   versicolor  versicolor    Correct
62   versicolor  versicolor    Correct
87   versicolor  versicolor    Correct
67   versicolor  versicolor    Correct
66   versicolor  versicolor    Correct
60   versicolor  versicolor    Correct
52   versicolor   virginica  Incorrect
148   virginica   virginica    Correct
```

[53]:
```
result.loc[result.result == "Incorrect"]
```

[53]:
```
          Actual   Predicted     result
83   versicolor   virginica  Incorrect
118   virginica  versicolor  Incorrect
130   virginica  versicolor  Incorrect
86   versicolor   virginica  Incorrect
52   versicolor   virginica  Incorrect
```

[54]:
```
result["result"].value_counts()
```

[54]:
```
Correct      40
Incorrect     5
Name: result, dtype: int64
```

[56]:
```
from sklearn.metrics import accuracy_score, r2_score
from sklearn.metrics import classification_report, confusion_matrix

#Evaluation
print('Training Accuracy: {:.3f}'.format(accuracy_score(y_train, y_train_pred)))
print('Testing Accuracy: {:.3f}'.format(accuracy_score(y_test, y_test_pred)))

print(confusion_matrix(y_train, y_train_pred))
print(classification_report(y_train, y_train_pred))

print(confusion_matrix(y_test, y_test_pred))
print(classification_report(y_test, y_test_pred))
```

```
Training Accuracy: 1.000
Testing Accuracy: 0.889
[[35  0  0]
 [ 0 35  0]
 [ 0  0 35]]
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        35
```

```
  versicolor       1.00      1.00      1.00        35
   virginica       1.00      1.00      1.00        35

    accuracy                           1.00       105
   macro avg       1.00      1.00      1.00       105
weighted avg       1.00      1.00      1.00       105

[[15  0  0]
 [ 0 12  3]
 [ 0  2 13]]
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       0.86      0.80      0.83        15
   virginica       0.81      0.87      0.84        15

    accuracy                           0.89        45
   macro avg       0.89      0.89      0.89        45
weighted avg       0.89      0.89      0.89        45
```

0.1.8 7. Conclusion

[105]:
```
class classifier:
    def __init__(self, data):
        self.answer = pd.DataFrame(data["species"])
        data = data.drop(columns = ["species"])

        self.data_pca = pca.transform(data)
        self.data_pca = pd.DataFrame(self.data_pca)

    def predict(self):
        return model.predict(self.data_pca)

    def accuracy(self):
        return accuracy_score(self.answer, self.predict())

    def report(self):
        dic = {"Actual": self.answer["species"], "Predicted": self.predict()}
        result = pd.DataFrame(dic)
        result["result"] = np.where(result.Actual == result.Predicted,␣
 ↪"Correct", "Incorrect")

        print("Accuracy:", self.accuracy())
        return result
```

[101]:
```
# select three rows from each species
test_data = sns.load_dataset('iris')

sesota = test_data[test_data["species"] == 'setosa'].head(1)
versicolor = test_data[test_data["species"] == 'versicolor'].head(1)
virginica = test_data[test_data["species"] == 'virginica'].head(1)


combined_data = pd.concat([sesota, versicolor, virginica], ignore_index=True)
combined_data
```

[101]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	7.0	3.2	4.7	1.4	versicolor
2	6.3	3.3	6.0	2.5	virginica

[102]:
```
# select 10 random rows

random_data = test_data.sample(n = 10)
```

[106]:
```
result = classifier(combined_data)

result.report()
```

```
Accuracy: 1.0
```

```
c:\Users\ybrot\AppData\Local\Programs\Python\Python39\lib\site-
packages\sklearn\base.py:439: UserWarning: X does not have valid feature names,
but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
c:\Users\ybrot\AppData\Local\Programs\Python\Python39\lib\site-
packages\sklearn\base.py:439: UserWarning: X does not have valid feature names,
but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

[106]:

	Actual	Predicted	result
0	setosa	setosa	Correct
1	versicolor	versicolor	Correct
2	virginica	virginica	Correct

[107]:
```
result = classifier(random_data)

result.report()
```

```
Accuracy: 1.0
```

```
c:\Users\ybrot\AppData\Local\Programs\Python\Python39\lib\site-
packages\sklearn\base.py:439: UserWarning: X does not have valid feature names,
but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

```
c:\Users\ybrot\AppData\Local\Programs\Python\Python39\lib\site-
packages\sklearn\base.py:439: UserWarning: X does not have valid feature names,
but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

[107]:

	Actual	Predicted	result
3	setosa	setosa	Correct
64	versicolor	versicolor	Correct
88	versicolor	versicolor	Correct
69	versicolor	versicolor	Correct
56	versicolor	versicolor	Correct
131	virginica	virginica	Correct
51	versicolor	versicolor	Correct
60	versicolor	versicolor	Correct
47	setosa	setosa	Correct
90	versicolor	versicolor	Correct