Modeling data (with decision trees)

In [20]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn as sk
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
```

In [21]:
```python
url = 'https://raw.githubusercontent.com/nlihin/data-analytics/main/datasets/iris.csv'
iris_df = pd.read_csv(url)
iris_df.head()
```

Out[21]:

	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

1. EDA

Get to know the data

How many instances (rows) with each label (flower species)?

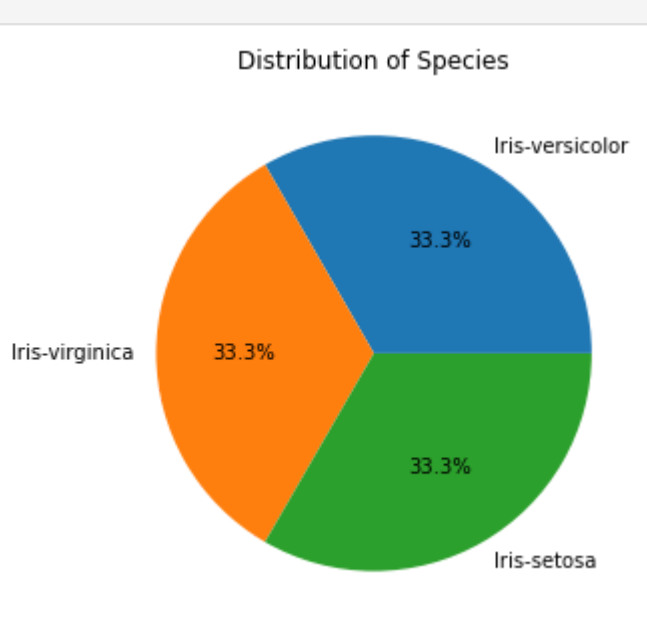
In [22]:
```python
pd.pivot_table(iris_df,index=['class'],values=['sepal_length','sepal_width','petal_length','petal_width'], aggf
```

Out[22]:

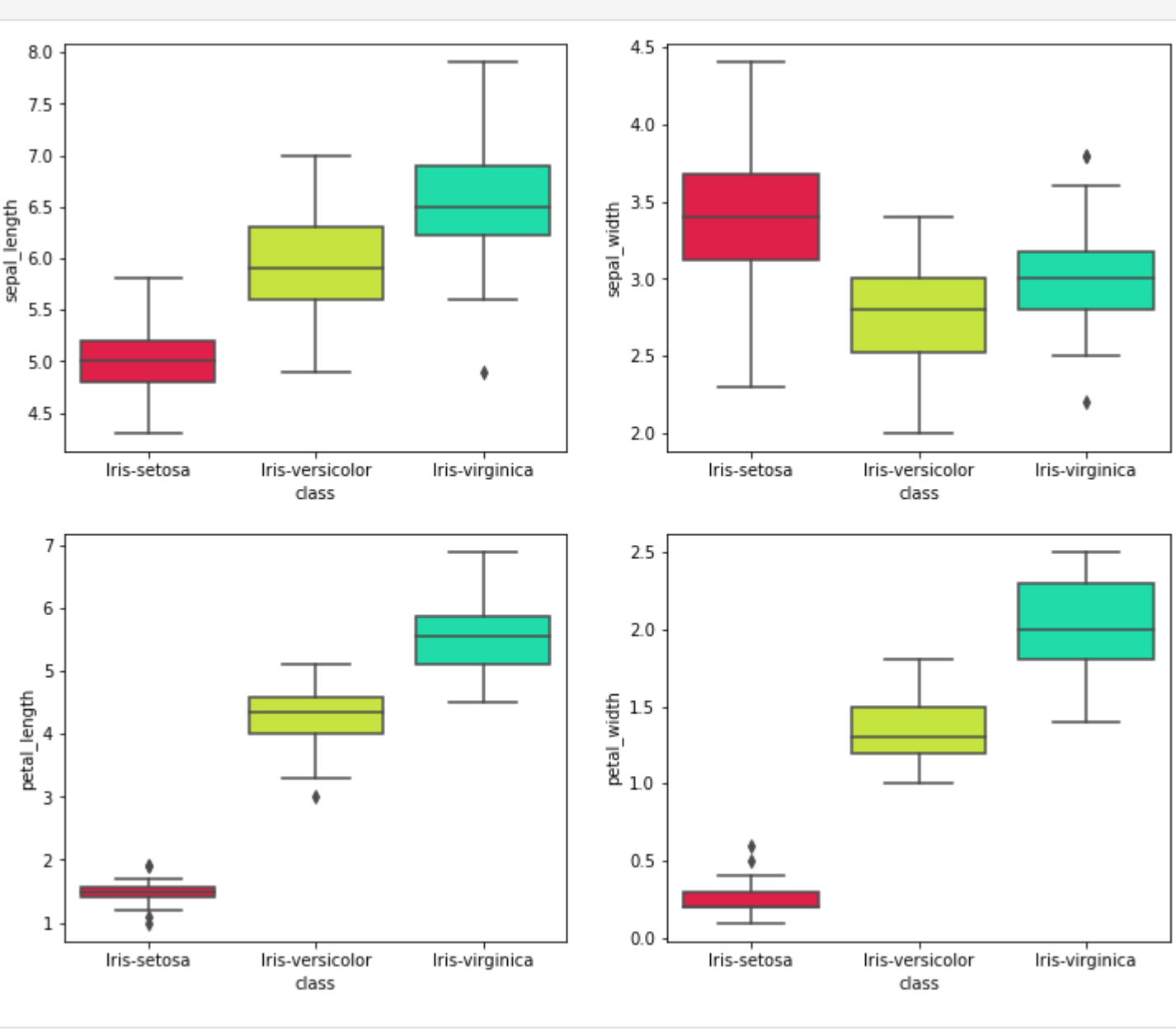
class	petal_length	petal_width	sepal_length	sepal_width
Iris-setosa	50.0	50.0	50.0	50.0
Iris-versicolor	50.0	50.0	50.0	50.0
Iris-virginica	50.0	50.0	50.0	50.0

Another way to look at it - species percentage

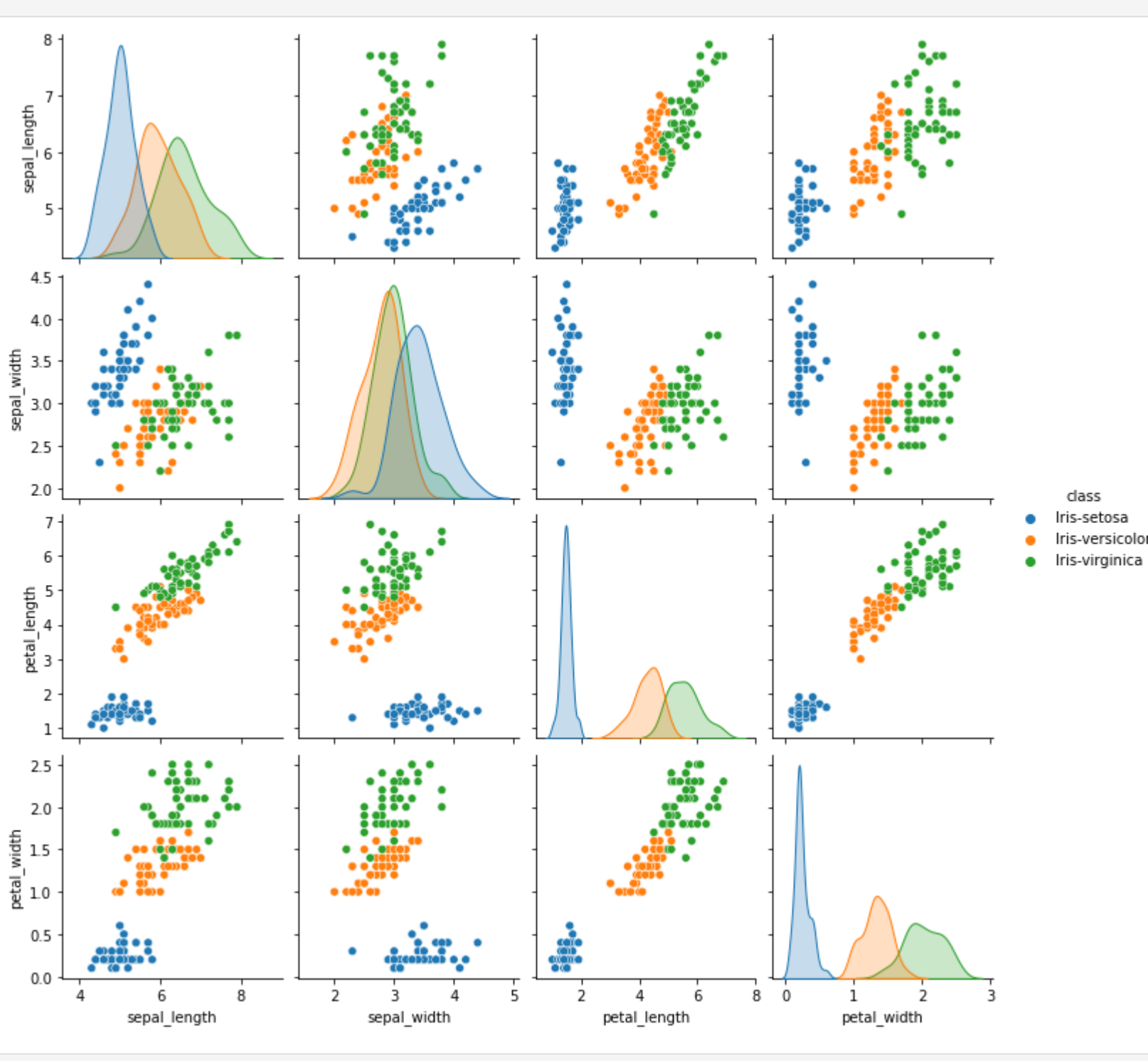
In [23]:
```python
plt.figure(1, figsize=(5,5))
plt.title("Distribution of Species")
iris_df['class'].value_counts().plot.pie(autopct="%1.1f%%") #try removing this - what happens?
plt.ylabel("")  #try removing this - what happens?
plt.show()
```



In [48]:
```python
fig, axes = plt.subplots(2,2,figsize=(12,10))
sns.boxplot(x = 'class', y = 'sepal_length', data = iris_df, palette="gist_ncar_r", ax = axes[0,0])
sns.boxplot(x = 'class', y = 'sepal_width', data = iris_df, palette="gist_ncar_r", ax = axes[0,1])
sns.boxplot(x = 'class', y = 'petal_length', data = iris_df, palette="gist_ncar_r", ax = axes[1,0])
sns.boxplot(x = 'class', y = 'petal_width', data = iris_df, palette="gist_ncar_r", ax = axes[1,1])
plt.show()
```



In [25]:
```python
sns.pairplot(iris_df, hue='class')
plt.show()
```



In [26]:
```python
plt.figure(figsize=(7,4))
sns.heatmap(iris_df.corr(),annot=True,cmap='Greens') #draws  heatmap with input as the correlation matrix calcu
plt.show()
```

Observations

- The Sepal Width and Length are not correlated
- The Petal Width and Length are highly correlated

We will use all the features for training the algorithm and check the accuracy.

Then we will use 1 Petal Feature and 1 Sepal Feature to check the accuracy of the algorithm as we are using only 2 features that are not correlated.

Thus we can have a variance in the dataset which may help in better accuracy. We will check it later.

2. Build a classification model

2.1 Define the model:

we will use a decision tree based on the gini index (gini index is the default)

In [27]:
```python
my_model = sk.tree.DecisionTreeClassifier(criterion="gini")
```

2.2 Fit the model on the data:

Define what are the features (x) and what is the target (y)

In [28]:
```python
features = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
X = iris_df[features]
y = iris_df['class']
```

Use the model we chose on the features we chose (X and y)

In [29]:
```python
my_model.fit(X, y)
```

Out[29]:
```
DecisionTreeClassifier()
```

now model1 is a tree, and it has tree features:

In [30]:
```python
type(my_model)
```

Out[30]:
```
sklearn.tree._classes.DecisionTreeClassifier
```

Let's see what we got:

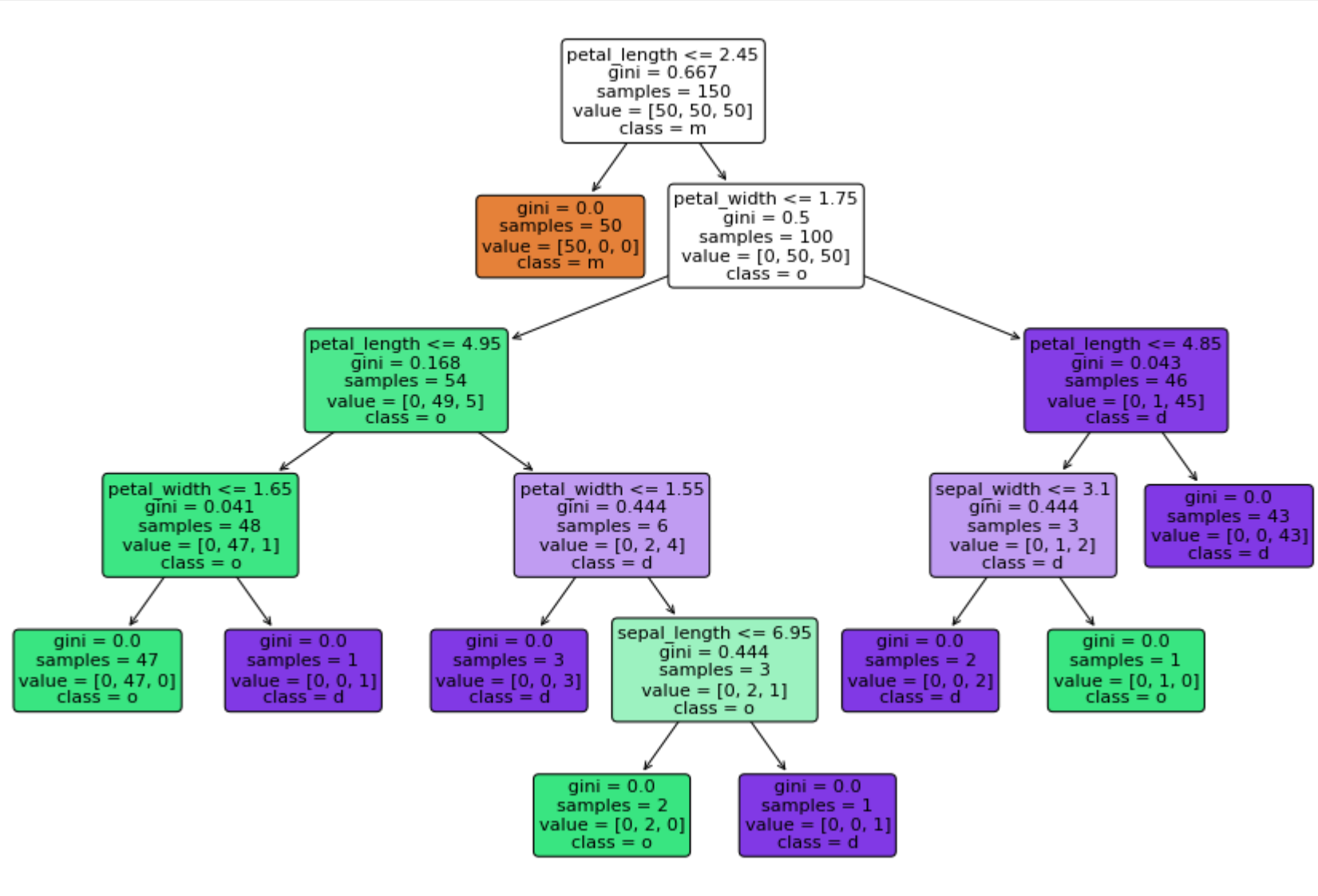
In [31]:
```python
text_representation = tree.export_text(my_model)
print(text_representation)
```

```
|--- feature_2 <= 2.45
|   |--- class: Iris-setosa
|--- feature_2 >  2.45
|   |--- feature_3 <= 1.75
|   |   |--- feature_2 <= 4.95
|   |   |   |--- feature_3 <= 1.65
|   |   |   |   |--- class: Iris-versicolor
|   |   |   |--- feature_3 >  1.65
|   |   |   |   |--- class: Iris-virginica
|   |   |--- feature_2 >  4.95
|   |   |   |--- feature_3 <= 1.55
|   |   |   |   |--- class: Iris-virginica
|   |   |   |--- feature_3 >  1.55
|   |   |   |   |--- feature_0 <= 6.95
|   |   |   |   |   |--- class: Iris-versicolor
|   |   |   |   |--- feature_0 >  6.95
|   |   |   |   |   |--- class: Iris-virginica
|   |--- feature_3 >  1.75
|   |   |--- feature_2 <= 4.85
|   |   |   |--- feature_1 <= 3.10
|   |   |   |   |--- class: Iris-virginica
|   |   |   |--- feature_1 >  3.10
|   |   |   |   |--- class: Iris-versicolor
|   |   |--- feature_2 >  4.85
|   |   |   |--- class: Iris-virginica
```

Vizualize:

In [32]:
```python
fig = plt.figure(figsize=(15,10))

tree.plot_tree(my_model,
               feature_names = features,
               class_names = 'model1.classes_',
               filled=True,
               rounded = True)
plt.show()
```



That's nice. But how well will it predict on new data?

2.3 Predict

Split the data to 70% train set and 30% test

In [33]:
```python
X_train, X_test, y_train, y_test = sk.model_selection.train_test_split(X, y, test_size=0.3, random_state=1)
```

In [34]:
```python
X_train.head(2)
```

Out[34]:

	sepal_length	sepal_width	petal_length	petal_width
118	7.7	2.6	6.9	2.3
18	5.7	3.8	1.7	0.3

In [35]:
```python
X_test.head()
```

Out[35]:

	sepal_length	sepal_width	petal_length	petal_width
14	5.8	4.0	1.2	0.2
98	5.1	2.5	3.0	1.1
75	6.6	3.0	4.4	1.4
16	5.4	3.9	1.3	0.4
131	7.9	3.8	6.4	2.0

In [36]:
```python
y_test.head()
```

Out[36]:
```
14         Iris-setosa
98     Iris-versicolor
75     Iris-versicolor
16         Iris-setosa
131     Iris-virginica
Name: class, dtype: object
```

Use the training data to train the model (a new model - model2)

In [37]:
```python
model2 = sk.tree.DecisionTreeClassifier(criterion="gini")
model2.fit(X_train,y_train)
```

Out[37]:
```
DecisionTreeClassifier()
```

Finally - predict:

In [38]:
```python
prediction2 = model2.predict(X_test)
prediction2
```

Out[38]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor'], dtype=object)
```

present in a nicer format:

In [39]:
```python
df = pd.DataFrame(columns = ['class', 'predictions'])
```

In [40]:
```python
df['class'] = y_test
df = df.reset_index(drop=True)
```

In [41]:
```python
df['predictions'] = pd.Series(prediction2)
df
```

Out[41]:

	class	predictions
0	Iris-setosa	Iris-setosa
1	Iris-versicolor	Iris-versicolor
2	Iris-versicolor	Iris-versicolor
3	Iris-setosa	Iris-setosa
4	Iris-virginica	Iris-virginica
5	Iris-versicolor	Iris-versicolor
6	Iris-virginica	Iris-virginica
7	Iris-setosa	Iris-setosa
8	Iris-setosa	Iris-setosa
9	Iris-virginica	Iris-virginica
10	Iris-versicolor	Iris-versicolor
11	Iris-setosa	Iris-setosa
12	Iris-virginica	Iris-virginica
13	Iris-versicolor	Iris-versicolor
14	Iris-versicolor	Iris-versicolor
15	Iris-setosa	Iris-setosa
16	Iris-versicolor	Iris-versicolor
17	Iris-versicolor	Iris-versicolor
18	Iris-setosa	Iris-setosa
19	Iris-setosa	Iris-setosa
20	Iris-versicolor	Iris-versicolor
21	Iris-versicolor	Iris-versicolor
22	Iris-versicolor	Iris-virginica
23	Iris-setosa	Iris-setosa
24	Iris-virginica	Iris-virginica
25	Iris-versicolor	Iris-versicolor
26	Iris-setosa	Iris-setosa
27	Iris-setosa	Iris-setosa
28	Iris-versicolor	Iris-versicolor
29	Iris-virginica	Iris-virginica
30	Iris-versicolor	Iris-versicolor
31	Iris-virginica	Iris-virginica
32	Iris-versicolor	Iris-versicolor
33	Iris-virginica	Iris-virginica
34	Iris-virginica	Iris-virginica
35	Iris-setosa	Iris-setosa
36	Iris-versicolor	Iris-versicolor
37	Iris-setosa	Iris-setosa
38	Iris-versicolor	Iris-versicolor
39	Iris-virginica	Iris-virginica
40	Iris-virginica	Iris-virginica
41	Iris-setosa	Iris-setosa
42	Iris-virginica	Iris-versicolor
43	Iris-virginica	Iris-virginica
44	Iris-versicolor	Iris-versicolor

2.4 Evaluate

There are various ways to measure accuracy:

The most obvious - (number of correct)/(total)

In [42]:
```python
sk.metrics.accuracy_score(prediction2, y_test)
```

Out[42]:
```
0.9555555555555556
```

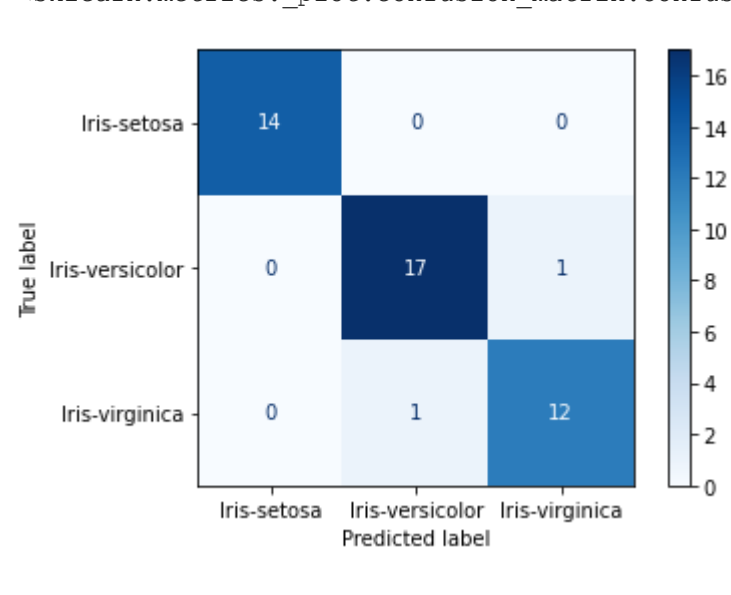
A truth table

In [43]:
```python
sk.metrics.confusion_matrix(y_test, prediction2)
```

Out[43]:
```
array([[14,  0,  0],
       [ 0, 17,  1],
       [ 0,  1, 12]], dtype=int64)
```

In [44]:
```python
sk.metrics.plot_confusion_matrix(model2, X_test, y_test, cmap=plt.cm.Blues)
```

Out[44]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x296166d9850>
```



Precision, Recall and F1 score - most common. But out of scope for this course

In [45]:
```python
print(sk.metrics.classification_report(y_test, prediction2, target_names=model2.classes_))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.94      0.94      0.94        18
 Iris-virginica       0.92      0.92      0.92        13

       accuracy                           0.96        45
      macro avg       0.96      0.96      0.96        45
   weighted avg       0.96      0.96      0.96        45
```

Precision, Recall and F1 score - most common. But out of scope for this course

In [52]:
```python
print(sk.metrics.classification_report(y_test, prediction2, target_names=model2.classes_))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.94      0.94      0.94        18
 Iris-virginica       0.92      0.92      0.92        13

       accuracy                           0.96        45
      macro avg       0.96      0.96      0.96        45
   weighted avg       0.96      0.96      0.96        45
```

3. Other models

3.1 Petal vs Sepal

Which attributes were the most important in the training data?

In [55]:
```python
dt = sk.tree.DecisionTreeClassifier(random_state=0)
dt.fit(X_train, y_train)
pred = dt.predict(X_test)
acc = sk.metrics.accuracy_score(pred, y_test)
```

In [56]:
```python
dt.feature_importances_
```

Out[56]:
```
array([0.02146947, 0.02146947, 0.06316954, 0.89389153])
```

organize it in a nicer format:

In [57]:
```python
feature_importances = pd.DataFrame(dt.feature_importances_,
                                   index = iris_df.columns[:4],
                                    columns=['importance']).sort_values('importance',
                                                                        ascending=False)
feature_importances.head()
```

Out[57]:

	importance
petal_width	0.893892
petal_length	0.063170
sepal_length	0.021469
sepal_width	0.021469

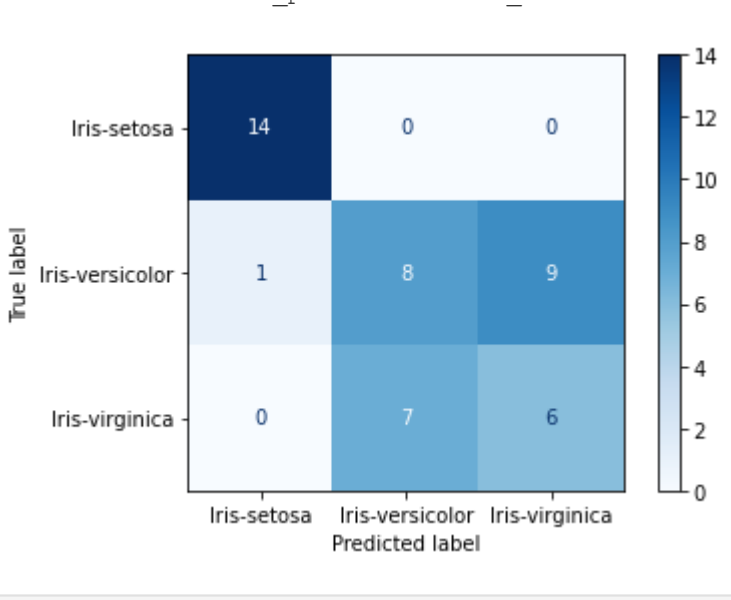
We see that petal is most important. What happens if we only use sepal?

In [58]:
```python
X_s = iris_df[['sepal_length','sepal_width']]
y_s = iris_df['class']
X_train_s, X_test_s, y_train_s, y_test_s = sk.model_selection.train_test_split(X_s, y_s, test_size=0.3, random_
model_s = sk.tree.DecisionTreeClassifier(criterion="gini")
model_s.fit(X_train_s, y_train_s)
prediction_s = model_s.predict(X_test_s)
print('The accuracy of the Decision Tree is',sk.metrics.accuracy_score(prediction_s,y_test_s))
```

```
The accuracy of the Decision Tree is 0.6222222222222222
```

In [59]:
```python
sk.metrics.plot_confusion_matrix(model_s, X_test_s, y_test_s, cmap=plt.cm.Blues)
```

Out[59]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x296174155b0>
```



In [60]:
```python
df2 = pd.DataFrame(columns = ['class', 'predictions'])
df2['class'] = y_test_s
df2 = df2.reset_index(drop=True)
df2['predictions'] = pd.Series(prediction_s)
```

Now let's try a model with two features: one sepal and one petal:

In [61]:
```python
#X_s = iris_df[['sepal_length','sepal_width']]
X_m = iris_df[['sepal_length','petal_width']]
y_m = iris_df['class']
X_train_m, X_test_m, y_train_m, y_test_m = sk.model_selection.train_test_split(X_m, y_m, test_size=0.3, random_
model_mix = sk.tree.DecisionTreeClassifier(criterion="gini", random_state=1)
model_mix.fit(X_train_m,y_train_m)
prediction_mix = model_mix.predict(X_test_m)
print('The accuracy of the Decision Tree is',sk.metrics.accuracy_score(prediction_mix, y_test_m))
```

```
The accuracy of the Decision Tree is 0.9555555555555556
```

3.2 an SVM model

In [62]:
```python
from sklearn.svm import SVC
svm_model = SVC(random_state = 0)
svm_model.fit(X_train, y_train)
svm_pred = svm_model.predict(X_test)
sk.metrics.accuracy_score(svm_pred, y_test)
```

Out[62]:
```
0.9777777777777777
```

> Terminology:
>
> - features - attributes
> - class/target - the feature we want to predict
> - A guide to trees - scikit-learn
>
> A summary of new functions:
>
> - `plot.pie(autopct="%1.1f%%")` - a pie plot. autopct adds percentages to each of the slices of the pie chart, 1.1 - percentages with 1 number after the decimal dot.
> - DecisionTreeClassifier documentation
> - `my_model = sk.tree.DecisionTreeClassifier(criterion="gini")` - decision tree model with the gini index as the node splitting criterion
> - `my_model.fit(X, y)` - train the model. X are the features and y is the target/class.
> - `my_model.feature_importances_` - shows how important is each of the features in model `my_model`
> - `sk.model_selection.train_test_split` - splits the data to train and test
> - `tree.export_text(my_model)` - shows the tree of the model in text format. documentation
> - `tree.plot_tree` - plots a nice decision tree. documentation
> - Metrics documentation
> - `sk.metrics.accuracy_score` - computes the model's jaccard accuracy score
> - `sk.metrics.confusion_matrix` - a truth table
> - `sk.metrics.plot_confusion_matrix` - a confusion matrix