IMPORT DATA

In [1]:
```
from sklearn.datasets import load_iris
iris = load_iris()
```

In [2]:
```
X=iris.data #input data sepal length, width, petal length, width
y=iris.target #response vector data: iris type 1, 2, 3

feature_names = iris.feature_names
target_names = iris.target_names
```

In [3]:
```
feature_names
```

Out[3]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [4]:
```
target_names
```

Out[4]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [5]:
```
#x is a numpy array - more feature rich than a python list
type(X)
type(y)
```

Out[5]:
```
numpy.ndarray
```

SPLIT DATA

In [6]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
#dimensionality
print(X_train.shape)
print(X_test.shape)
```

```
(120, 4)
(30, 4)
```

CREATE MODEL

In [7]:
```
#import K Nearest Classifier algorithm to interpret data
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3) #match 3 iris flower types
knn.fit(X_train, y_train)
```

Out[7]:
```
KNeighborsClassifier(n_neighbors=3)
```

In [8]:
```
#import Decision Tree algorithm to interpret data
# from sklearn.tree import DecisionTreeClassifier
# knn = DecisionTreeClassifier()
# knn.fit(X_train, y_train)
```

CHECK OUTPUT

In [9]:
```
#make predictions
y_pred = knn.predict(X_test)
```

In [10]:
```
from sklearn import metrics
print(metrics.accuracy_score(y_test, y_pred))
```

```
0.9333333333333333
```

IMPROVE

- Modify test size
- Optimise K (here 3 is optimal as there are 3 flower classifications)
- Try another model eg. decision tree. (this did not improve accuracy)

PROVIDE FURTHER SAMPLES TO TEST

In [17]:
```
sample = [[3,5,4,2],[2,3,5,4], [1,2,3,1]]
predictions = knn.predict(sample)
print(predictions)
pred_species = [iris.target_names[p] for p in predictions]
print("predictions: ", pred_species)
```

```
[1 2 0]
predictions:  ['versicolor', 'virginica', 'setosa']
```

MODEL PERSISTENCE

In [21]:
```
#save the model as mlbrain.joblib so we don't need to keep creating it
from joblib import dump, load
dump(knn, 'mlbrain.joblib')
```

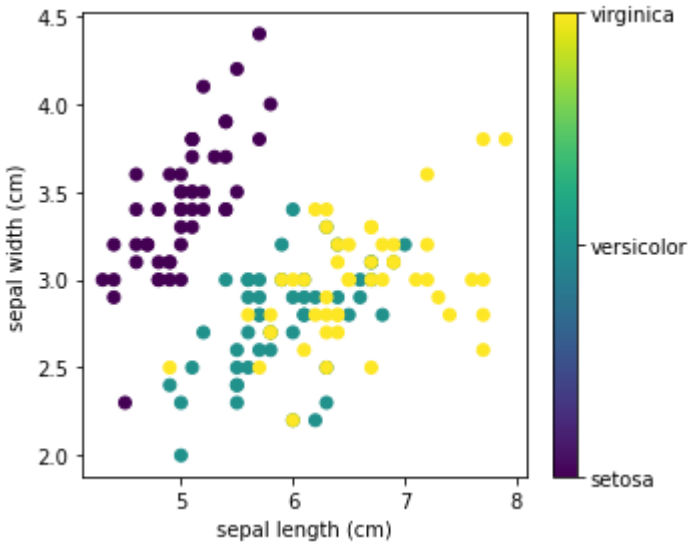
Out[21]:
```
['mlbrain.joblib']
```

In [25]:
```
model = load('mlbrain.joblib')
model.predict(sample)
```

Out[25]:
```
array([1, 2, 0])
```

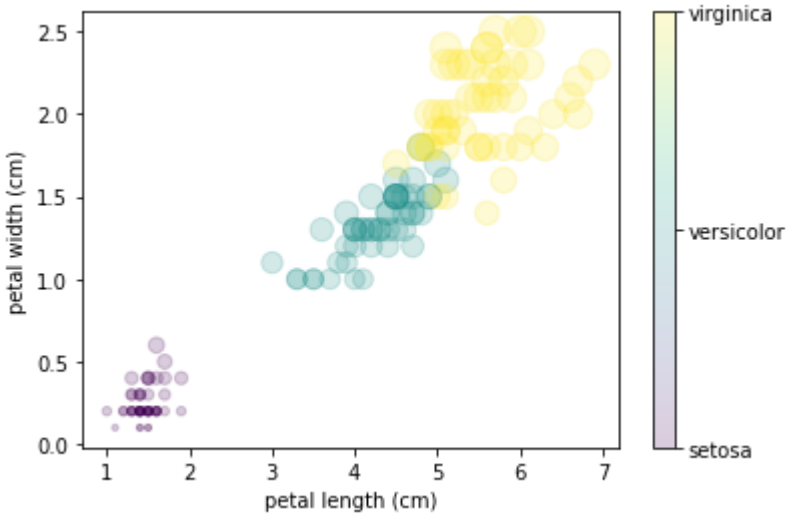
PLOTTING DATA

In [26]:
```
from sklearn.datasets import load_iris
iris = load_iris()
import matplotlib.pyplot as plt

# The indices of the features that we are plotting
x_index = 0
y_index = 1

# colorbar with the Iris target names
formatter = plt.FuncFormatter(lambda i, *args: iris.target_names[int(i)])

#chart configurations
plt.figure(figsize=(5, 4))
plt.scatter(iris.data[:, x_index], iris.data[:, y_index], c=iris.target)
plt.colorbar(ticks=[0, 1, 2], format=formatter)
plt.xlabel(iris.feature_names[x_index])
plt.ylabel(iris.feature_names[y_index])

plt.tight_layout()
plt.show()
```



In [28]:
```
features = iris.data.T

plt.scatter(features[2], features[3], alpha=0.2,
            s=100*features[3], c=iris.target, cmap='viridis')
#https://jakevdp.github.io/PythonDataScienceHandbook/04.02-simple-scatter-plots.html
plt.xlabel(iris.feature_names[2])
plt.ylabel(iris.feature_names[3]);
plt.colorbar(ticks=[0, 1, 2], format=formatter)
```

Out[28]:
```
<matplotlib.colorbar.Colorbar at 0x1cfc46d3f70>
```



In [ ]: