Classification using Naive Bayes

There are various ways to load data. For example, you can use pandas to load data from a CSV file or from a webpage. You can use load data with Scikit Learn. You can also use Seaborn to load data. Iris is a built-in dataset in Seaborn.

In [1]:

```
import seaborn as sns
iris = sns.load_dataset('iris')
iris.head()
```

Out[1]:

	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

It is a two dimension feature matrix.

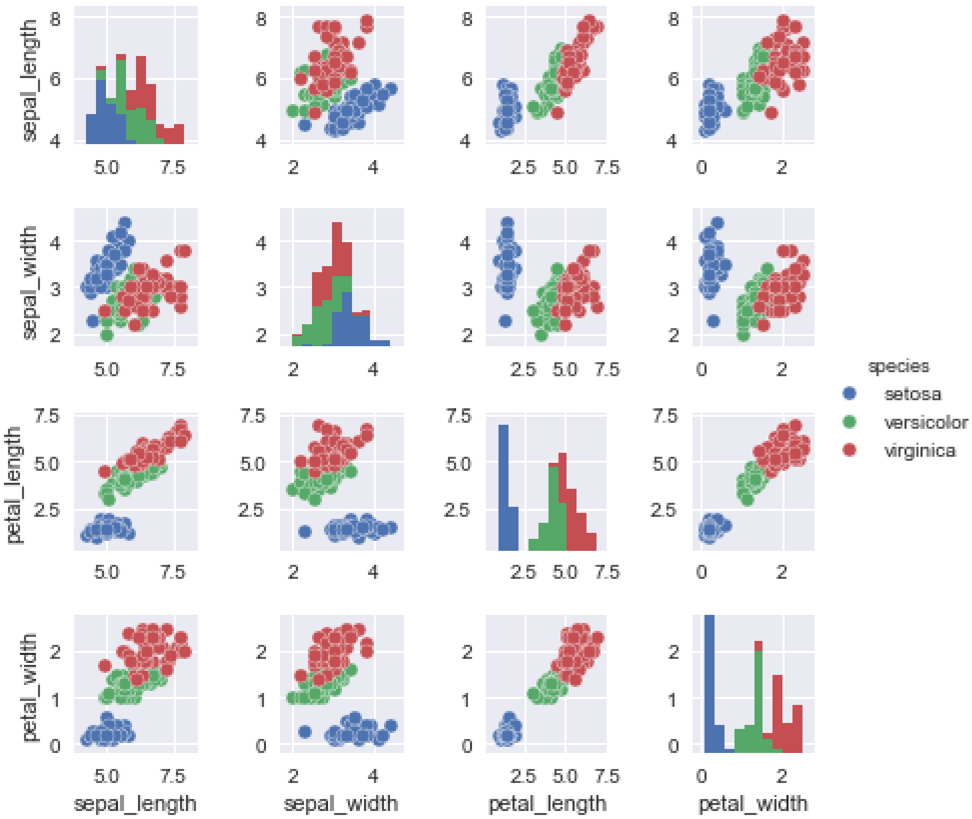
In [3]:

```
%matplotlib inline
```

Set aesthetic parameters in one step.

```
In [4]: sns.set()
        sns.pairplot(iris, hue='species', size=1.5)
```

```
Out[4]: <seaborn.axisgrid.PairGrid at 0x1c5877de518>
```



```
In [5]: import sklearn
        from sklearn.preprocessing import scale
        from sklearn.cross_validation import train_test_split
        from sklearn import metrics
        from sklearn import preprocessing
```

```
C:\Users\by3001pm\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
\cross_validation.py:41: DeprecationWarning: This module was deprecated in ve
rsion 0.18 in favor of the model_selection module into which all the refactor
ed classes and functions are moved. Also note that the interface of the new C
V iterators are different from that of this module. This module will be remov
ed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

```
In [7]: X_iris = iris.drop('species', axis=1)
        X_iris.shape
```

```
Out[7]: (150, 4)
```

```
In [8]: y_iris = iris['species']
        y_iris.shape
```

```
Out[8]: (150,)
```

Split the data into a training set and a testing set using the train_test_split utility function. Learn more about splitting data at http://scikit-learn.org/0.16/modules/generated/sklearn.cross_validation.train_test_split.html (http://scikit-learn.org/0.16/modules/generated/sklearn.cross_validation.train_test_split.html)

In [9]:

```
from sklearn.cross_validation import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X_iris, y_iris,
random_state=1) # random_state generates random sampling
```

In [10]:

```
from sklearn.naive_bayes import GaussianNB # 1. choose model class
model = GaussianNB() # 2. instantiate model
model.fit(Xtrain, ytrain) # 3. fit model to data
y_model = model.predict(Xtest) # 4. predict on new data
```

In [11]:

```
from sklearn.metrics import accuracy_score
accuracy_score(ytest, y_model)
```

Out[11]:

```
0.97368421052631582
```

The accuracy is over 97%