Support Vector Machines on Iris flower data set

Introduction

This tutorial walks you through implementing scikit-learn’s SVM on the Iris training set. It demonstrates the use of a few other functions from scikit-learn such as train_test_split and classification_report.

A Note About The Data

The data for this tutorial is famous. Called, the iris dataset, it contains four variables measuring various parts of iris flowers of three related species, and then a fourth variable with the species name.

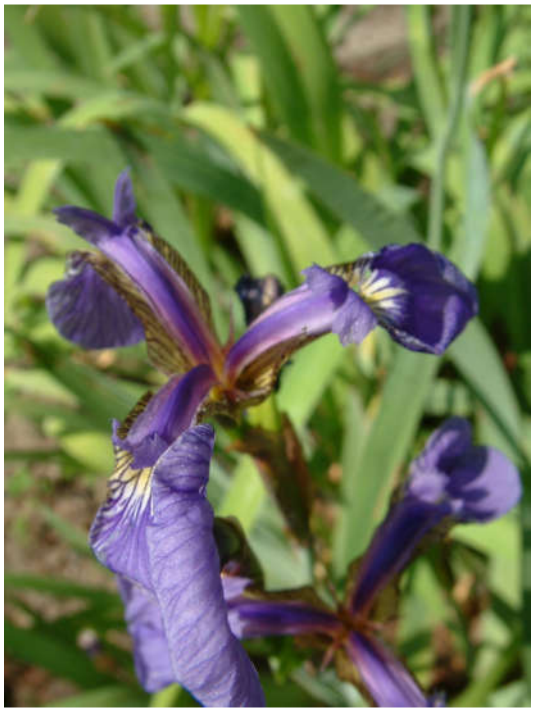
The reason it is so famous in machine learning and statistics communities is because the data requires very little preprocessing (i.e. no missing values, all features are floating numbers, etc.).

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by Sir Ronald Fisher in the 1936 as an example of discriminant analysis.

The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor), so 150 total samples. Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

In [1]:
```
# The Iris Setosa
from IPython.display import Image
url = 'http://upload.wikimedia.org/wikipedia/commons/5/56/Kosaciec_szczecinkowaty_I
ris_setosa.jpg'
Image(url,width=300, height=300)
```

Out[1]:



The iris dataset contains measurements for 150 iris flowers from three different species.

The three classes in the Iris dataset:

```
Iris-setosa (n=50)
Iris-versicolor (n=50)
Iris-virginica (n=50)
```

The four features of the Iris dataset:

```
sepal length in cm
sepal width in cm
petal length in cm
petal width in cm
```

Loading the data

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
iris = sns.load_dataset('iris')
```

Exploratory Data Analysis

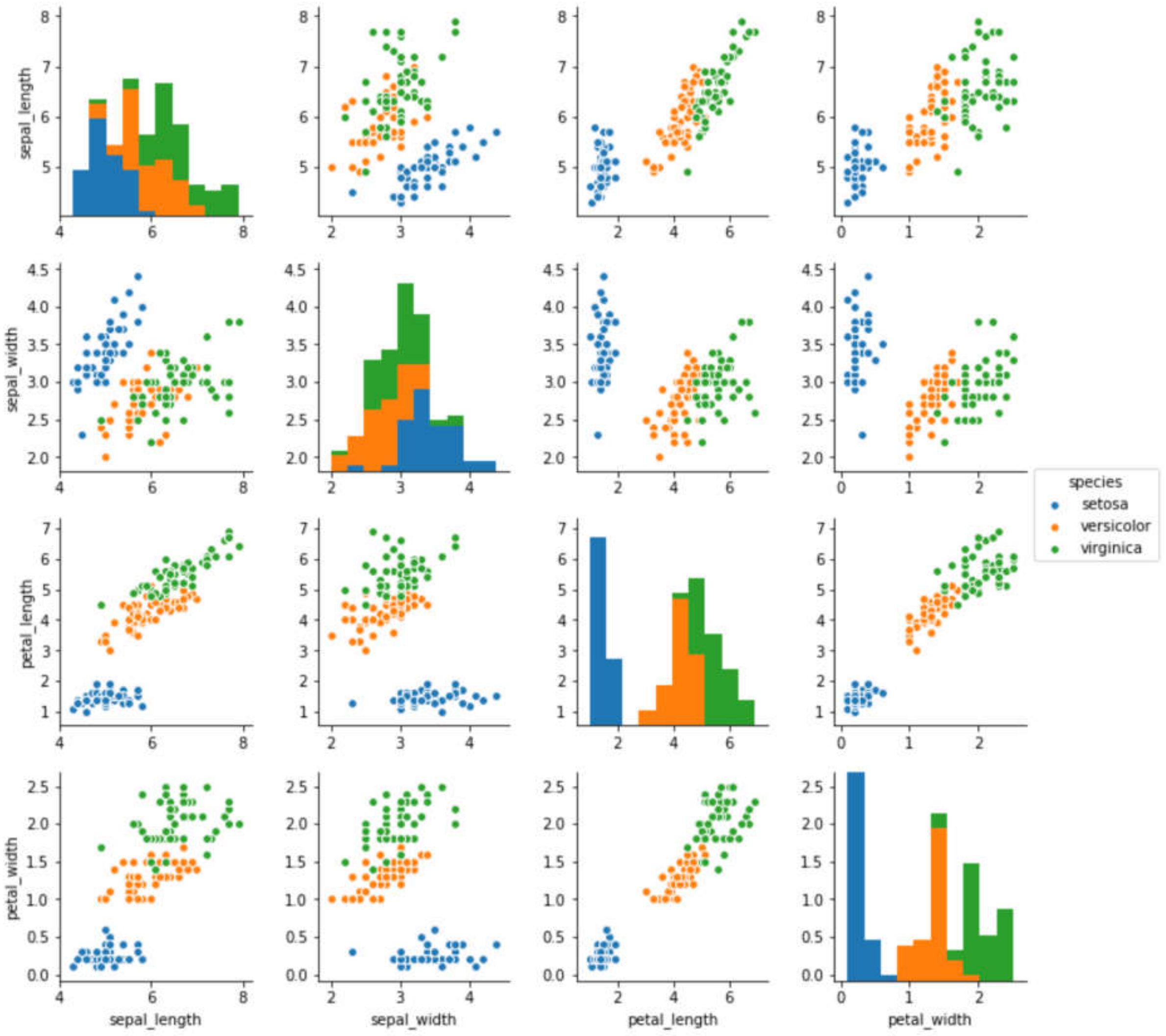
In [3]:
```
iris.head()
```

Out[3]:

	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

```
In [4]: #Creating a pairplot of the data set
        sns.pairplot(iris,hue='species')
```

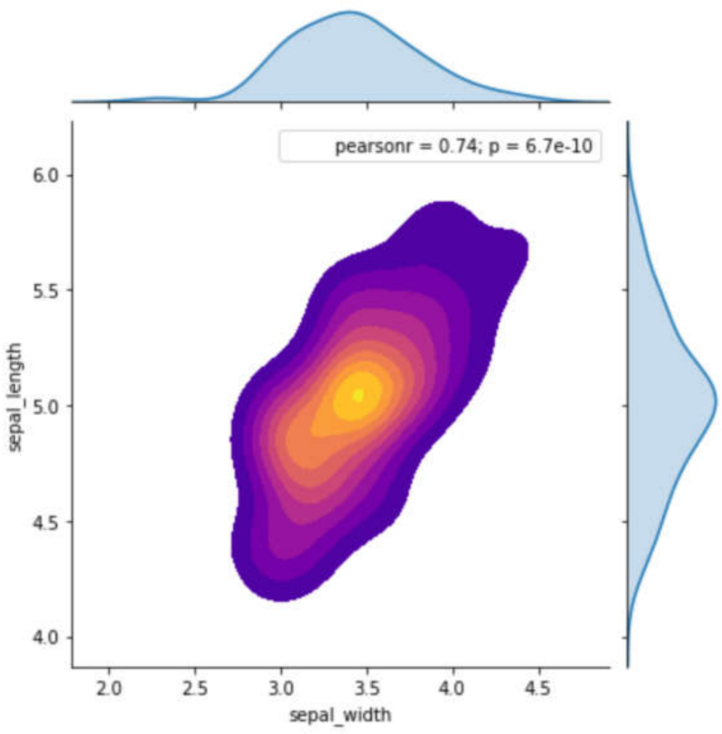
```
Out[4]: <seaborn.axisgrid.PairGrid at 0x1965f8c6470>
```



Creating a kde plot of sepal_length versus sepal width for setosa species of flower.

```
In [5]:  setosa = iris[iris['species']=='setosa']
         sns.jointplot( setosa['sepal_width'], setosa['sepal_length'],kind='kde',cmap="plasm
         a",shade_lowest=False)
```

```
Out[5]:  <seaborn.axisgrid.JointGrid at 0x196641e8400>
```



Train Test Split

Splitting data into a training set and a testing set.

```
In [6]:  from sklearn.model_selection import train_test_split
```

```
In [7]:  X = iris.drop('species',axis=1)
         y = iris['species']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

Train Support Vector Machine Classifier

```
In [8]:  from sklearn.svm import SVC
```

```
In [9]:  svc_model = SVC()
```

```
In [10]: svc_model.fit(X_train,y_train)
```

```
Out[10]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False)
```

Model Evaluation

In [11]:
```
predictions = svc_model.predict(X_test)
```

In [12]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [13]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[14  0  0]
 [ 0 16  0]
 [ 0  2 13]]
```

In [14]:
```
print(classification_report(y_test,predictions))
```

```
             precision    recall  f1-score   support

     setosa       1.00      1.00      1.00        14
 versicolor       0.89      1.00      0.94        16
  virginica       1.00      0.87      0.93        15

avg / total       0.96      0.96      0.96        45
```

Using Gridsearch to tune parameters

Với Grid Search, giả dụ giá trị của 2 parameter lần lượt từ 0-9. Grid Search sẽ lần lượt ghép từng giá trị của param 1 với param 2 để tính toán độ chính xác của model. Đảm bảo không bỏ sót cặp parameter nào.

Ưu điểm: Diệt nhầm còn hơn bỏ sót, nên thường được ưu tiên lựa chọn.

Nhược điểm: Tuy nhiên đối với các model cần thiết lập nhiều parameter và nhiều giá trị thì việc tunning sẽ mất rất nhiều thời gian, hàng giờ, vài giờ thậm chỉ có thể tính bằng ngày.

In [15]:
```
from sklearn.grid_search import GridSearchCV
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: Depre
cationWarning: This module was deprecated in version 0.18 in favor of the model_
selection module into which all the refactored classes and functions are moved.
Also note that the interface of the new CV iterators are different from that of
this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\grid_search.py:42: Deprecatio
nWarning: This module was deprecated in version 0.18 in favor of the model_selec
tion module into which all the refactored classes and functions are moved. This
module will be removed in 0.20.
  DeprecationWarning)
```

In [16]:
```
param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001]}
```

In [17]:
```
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=2)
grid.fit(X_train,y_train)
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
```

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
[CV] C=0.1, gamma=1 ..................................................
[CV] ......................................... C=0.1, gamma=1 -   0.0s
[CV] C=0.1, gamma=1 ..................................................
[CV] ......................................... C=0.1, gamma=1 -   0.0s
[CV] C=0.1, gamma=1 ..................................................
[CV] ......................................... C=0.1, gamma=1 -   0.0s
[CV] C=0.1, gamma=0.1 ................................................
[CV] ....................................... C=0.1, gamma=0.1 -   0.0s
[CV] C=0.1, gamma=0.1 ................................................
[CV] ....................................... C=0.1, gamma=0.1 -   0.0s
[CV] C=0.1, gamma=0.1 ................................................
[CV] ....................................... C=0.1, gamma=0.1 -   0.0s
[CV] C=0.1, gamma=0.01 ...............................................
[CV] ...................................... C=0.1, gamma=0.01 -   0.0s
[CV] C=0.1, gamma=0.01 ...............................................
[CV] ...................................... C=0.1, gamma=0.01 -   0.0s
[CV] C=0.1, gamma=0.01 ...............................................
[CV] ...................................... C=0.1, gamma=0.01 -   0.0s
[CV] C=0.1, gamma=0.001 ..............................................
[CV] ..................................... C=0.1, gamma=0.001 -   0.0s
[CV] C=0.1, gamma=0.001 ..............................................
[CV] ..................................... C=0.1, gamma=0.001 -   0.0s
[CV] C=0.1, gamma=0.001 ..............................................
[CV] ..................................... C=0.1, gamma=0.001 -   0.0s
[CV] C=1, gamma=1 ....................................................
[CV] ........................................... C=1, gamma=1 -   0.0s
[CV] C=1, gamma=1 ....................................................
[CV] ........................................... C=1, gamma=1 -   0.0s
[CV] C=1, gamma=1 ....................................................
[CV] ........................................... C=1, gamma=1 -   0.0s
[CV] C=1, gamma=0.1 ..................................................
[CV] ......................................... C=1, gamma=0.1 -   0.0s
[CV] C=1, gamma=0.1 ..................................................
[CV] ......................................... C=1, gamma=0.1 -   0.0s
[CV] C=1, gamma=0.1 ..................................................
[CV] ......................................... C=1, gamma=0.1 -   0.0s
[CV] C=1, gamma=0.01 .................................................
[CV] ........................................ C=1, gamma=0.01 -   0.0s
[CV] C=1, gamma=0.01 .................................................
[CV] ........................................ C=1, gamma=0.01 -   0.0s
[CV] C=1, gamma=0.01 .................................................
[CV] ........................................ C=1, gamma=0.01 -   0.0s
[CV] C=1, gamma=0.001 ................................................
[CV] ....................................... C=1, gamma=0.001 -   0.0s
[CV] C=1, gamma=0.001 ................................................
[CV] ....................................... C=1, gamma=0.001 -   0.0s
[CV] C=1, gamma=0.001 ................................................
[CV] ....................................... C=1, gamma=0.001 -   0.0s
[CV] C=10, gamma=1 ...................................................
[CV] .......................................... C=10, gamma=1 -   0.0s
[CV] C=10, gamma=1 ...................................................
[CV] .......................................... C=10, gamma=1 -   0.0s
[CV] C=10, gamma=1 ...................................................
[CV] .......................................... C=10, gamma=1 -   0.0s
[CV] C=10, gamma=0.1 .................................................
[CV] ........................................ C=10, gamma=0.1 -   0.0s
[CV] C=10, gamma=0.1 .................................................
[CV] ........................................ C=10, gamma=0.1 -   0.0s
[CV] C=10, gamma=0.1 .................................................
[CV] ........................................ C=10, gamma=0.1 -   0.0s
[CV] C=10, gamma=0.01 ................................................
[CV] ....................................... C=10, gamma=0.01 -   0.0s
[CV] C=10, gamma=0.01 ................................................
```

```
[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:    0.1s finished
```

Out[17]:
```
GridSearchCV(cv=None, error_score='raise',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
       fit_params={}, iid=True, n_jobs=1,
       param_grid={'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001]},
       pre_dispatch='2*n_jobs', refit=True, scoring=None, verbose=2)
```

In [18]:
```
grid_predictions = grid.predict(X_test)
```

In [19]:
```
print(confusion_matrix(y_test,grid_predictions))
```

```
[[14  0  0]
 [ 0 16  0]
 [ 0  2 13]]
```

In [20]:
```
print(classification_report(y_test,grid_predictions))
```

```
             precision    recall  f1-score   support

     setosa       1.00      1.00      1.00        14
 versicolor       0.89      1.00      0.94        16
  virginica       1.00      0.87      0.93        15

avg / total       0.96      0.96      0.96        45
```

Exercise

Using Breast Cancer dataset that import from Sklearn. Your task uses SVM to build a classifier which can predict whether they survived or not?

Deadline: 9 am next Tuesday

Link: https://www.dropbox.com/request/oZ8rwjKNRzYtGaAc78ZI (https://www.dropbox.com/request/oZ8rwjKNRzYtGaAc78ZI)

Reference

Read more about:

```
- Girdsearch: http://scikit-learn.org/0.15/auto_examples/grid_search_digits.html
```