Classification in the Iris data set
A test example for `Jupyter nbconvert`

Dylan P. Tweed
See Linkedin profile for affiliation

February 8, 2017

This document is the result of a `nbconvert -to pdf` test on a very simple `ipython notebook`. Along with this abstract, title, author, affiliations, dates and keywords metadata are displayed. Furthermore this document should illustrate additional features; captions, labels, bibliography figures and tables rescaling, cells hiding. The orignal notebook was build so that to test multiple templates (see github.com/BreizhZut/Jupyter_nbconvert_pdftemplate) corresponding to documentation, article, book and presentations formats.

Keywords: Jupyter; ipython; nbconvert; template: PDF LaTeX

1 Data

The iris dataset is common test example for machine learning and can be found in the `datasets` packages of `R` or as in this instance the `sklearn` package in `python`. This data set was first published in [Fisher, 1936], in was further use for the purpose of testing machine learning classification algorithm such as in [Ro and Pe, 1973], [Dasarathy, 1980].

```
In [3]: # This code should appear in the codedoc not in the article
        # load the iris data set
        iris = datasets.load_iris()
        # print(iris['DESCR']) # uncomment to test a stream output
```

1. Number of Instances: 150 (50 in each of three classes)
2. Number of Attributes: 4 numeric, predictive attributes and the class
   - sepal length in cm
   - sepal width in cm
   - petal length in cm
   - petal width in cm
3. class:

- Iris-Setosa
- Iris-Versicolour
- Iris-Virginica

```
In [4]: # This code should appear in the codedoc not in the article
        # Copy data as a pandas data frame
        pd_iris = pd.DataFrame(iris.data,columns=\
            ['sepal length','sepal width','petal length','petal width'])
        # Add the target class in the data frame
        target = iris.target
        # Copy the targetnames
        target_names = iris.target_names
```

1.1 Data frames

The 3 class are indicated in the data as integers 0, 1 and 2:

```
In [5]: # This should appear everywhere
        Counter(target)
```

```
Out[5]: Counter({0: 50, 1: 50, 2: 50})
```

With the corresponding class names:

```
In [6]: # This should appear everywhere
        list(target_names)
```

```
Out[6]: ['setosa', 'versicolor', 'virginica']
```

We explore the first few element of the iris data set for each class:

- setosa encoded as 0 (see Table `Out[7]`),
- versicolor encoded as 1 (see Table `Out[8]`)
- virginica encoded as 2 (see Table `Out[9]`).

We note that the row are ordered by class. This is not important here, since we try to test reference to some tables but for machine learning tasks it is advised to shuffle the row both in the data and the target.

```
In [7]: # This code should appear in the codedoc not in the article
        # print the data frame as a table
        pd_iris[target==0].head(n=10)
```

	sepal length	sepal width	petal length	petal width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1

Out[7]: First ten rows corredsponding to the Setosa class

	sepal length	sepal width	petal length	petal width
50	7.0	3.2	4.7	1.4
51	6.4	3.2	4.5	1.5
52	6.9	3.1	4.9	1.5
53	5.5	2.3	4.0	1.3
54	6.5	2.8	4.6	1.5
55	5.7	2.8	4.5	1.3
56	6.3	3.3	4.7	1.6
57	4.9	2.4	3.3	1.0
58	6.6	2.9	4.6	1.3
59	5.2	2.7	3.9	1.4

Out[8]: First ten rows corresponding to the Versicolor class

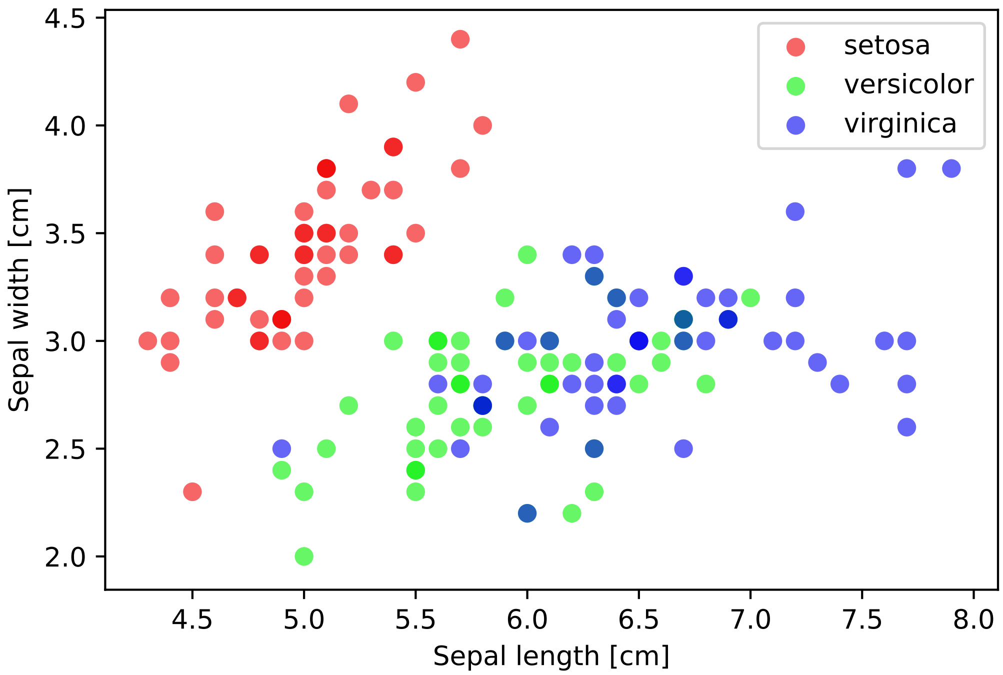
	sepal length	sepal width	petal length	petal width
100	6.3	3.3	6.0	2.5
101	5.8	2.7	5.1	1.9
102	7.1	3.0	5.9	2.1
103	6.3	2.9	5.6	1.8
104	6.5	3.0	5.8	2.2
105	7.6	3.0	6.6	2.1
106	4.9	2.5	4.5	1.7
107	7.3	2.9	6.3	1.8
108	6.7	2.5	5.8	1.8
109	7.2	3.6	6.1	2.5

Out[9]: First ten rows corresponding to the Virginica class

1.2 Visualization

```
In [10]: # This code should appear in the codedoc not in the article
         colors = ["#f00000","#00f000","#0000f0"]
         for col, i, target_name in zip(colors, [0, 1, 2], target_names):
```

```
        byclass = target==i
        plt.scatter(pd_iris['sepal length'][byclass],
                    pd_iris['sepal width'][byclass],
                    c=col, alpha=0.6,label=target_name)
        plt.xlabel('Sepal length [cm]')
        plt.ylabel('Sepal width [cm]')
    plt.legend(scatterpoints=1)
    plt.show()
```



Out[10]: Scatter plot sepal width as a function of the sepal lenght for the iris dataset. As the legend indicates, the color code corresponds to the class.

2 Model

For fun were testing different classification models for the iris dataset using the Support Vector Classification (SVC) method. This exemple is taken from the`sklearn` documantation. We test the SVC methods with:

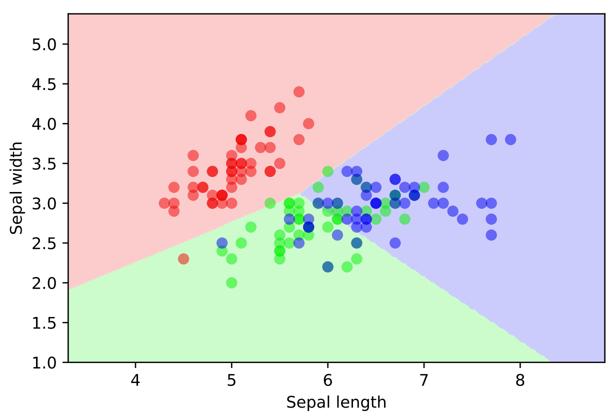
- a linear kernel (see Figure `Out[13]`)
- a Radial Basis Function kernel (RBF, see Figure `Out[14]`)
- a degree 3 polynomial kernel (see Figure `Out[15]`)

```
In [11]: # This code should appear in the codedoc not in the article
         # Select columns
         colsel = np.logical_or(pd_iris.columns.values == "sepal width",
                                pd_iris.columns.values =="sepal length")
         X      = np.array(pd_iris)[:,colsel]
```

```
# Run SVM
C         = 1.0  # SVM regularization parameter
lin_svc   = svm.LinearSVC(C=C).fit(X, target)
rbf_svc   = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, target)
poly_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, target)
# create a mesh to plot in
h = .02  # step size in the mesh
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))
```

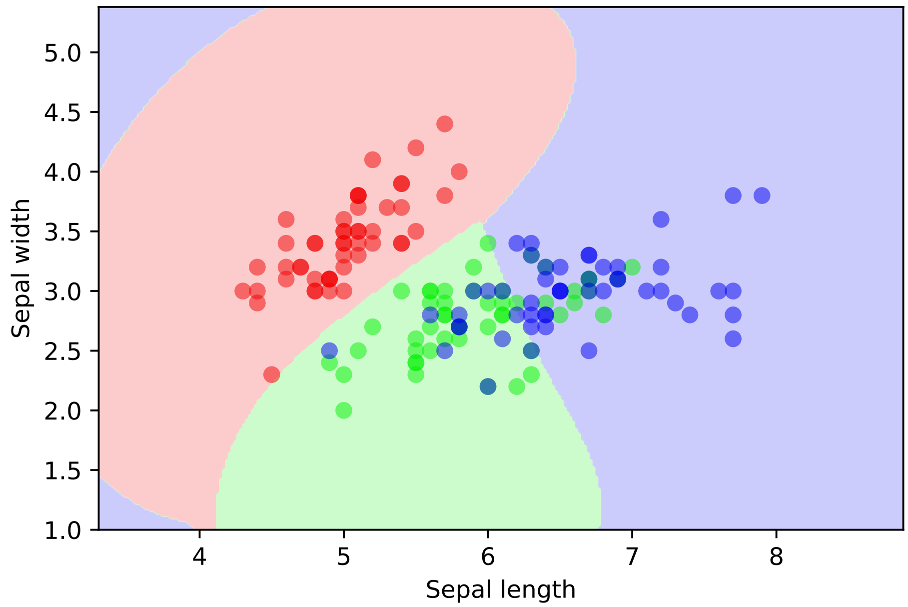
```
In [12]: # This code should appear in the codedoc not in the article
         colors = ["#f00000","#00f000","#0000f0"]
         tcm = LinearSegmentedColormap.from_list("iris target", colors, N=3)
         def plot_svmtest(clf):

             Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
             # Put the result into a color plot
             Z = Z.reshape(xx.shape)
             plt.contourf(xx, yy, Z, cmap=tcm, alpha=0.2)
             # Plot also the training points
             plt.scatter(X[:, 0], X[:, 1], c=target, cmap=tcm, alpha=0.5)
             plt.xlabel('Sepal length')
             plt.ylabel('Sepal width')
             plt.xlim(xx.min(), xx.max())
             plt.ylim(yy.min(), yy.max())
             plt.show()
```

2.1 Linear kernel SVC



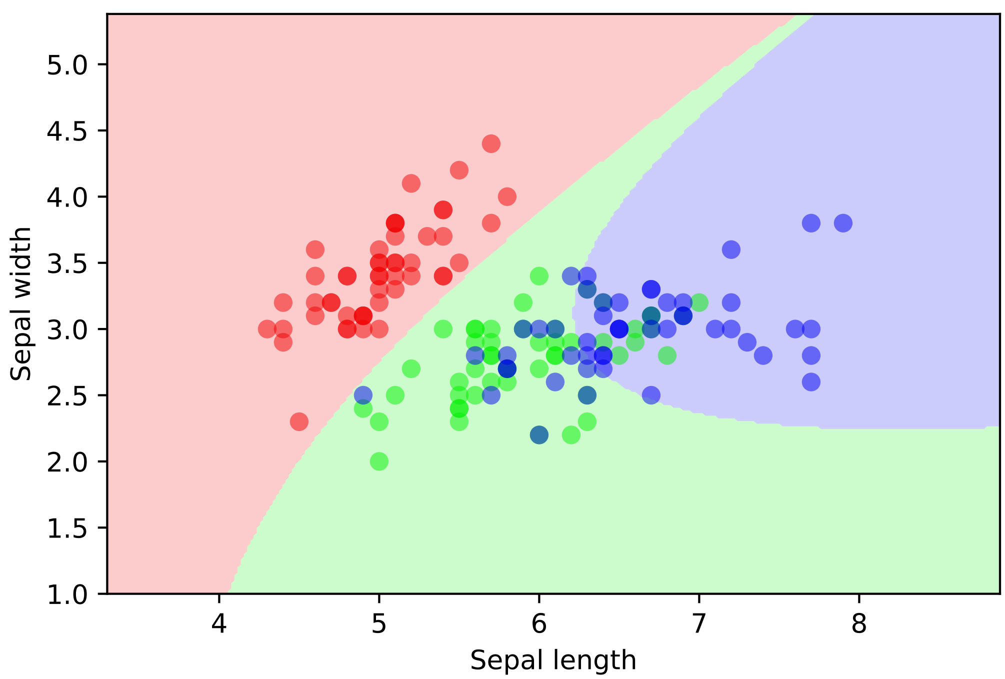
Out[13]: Same as Figure Out[10]. The shaded region correspond to the predictions of Linear SVC model.

2.2 Radial basis function kernel SVC



Out[14]: Same as Figure Out[10]. The shaded region correspond to the predictions of SVC RBF model.

2.3 Polynomial kernel SVC



Out[15]: Same as Figure Out[10]. The shaded region correspond to the predictions of polynomial SVC model.

References

[Dasarathy, 1980] Dasarathy, B. V. (1980). Nosing around the neighborhood: A new system structure and classification rule for recognition in partially exposed environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(1):67–71.

[Fisher, 1936] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188.

[Ro and Pe, 1973] Ro, D. and Pe, H. (1973). *Pattern Classification and Scene Analysis*. Wiley.