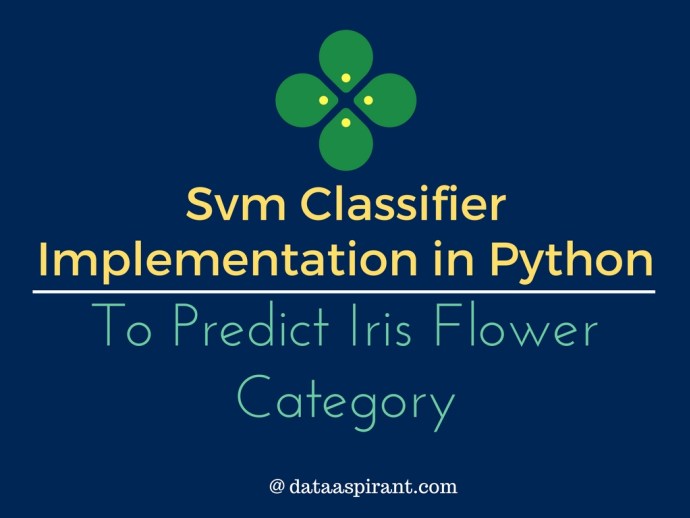
**Support Vector Machine (Svm Classifier) Implemenation In Python With Scikit-Learn**

January 25, 2017 [Saimadhu Polamuri](https://dataaspirant.com/author/saimadhu/)

[5 Comments](https://dataaspirant.com/2017/01/25/svm-classifier-implemenation-python-scikit-learn/#comments)

[Machine Learning](https://dataaspirant.com/category/machine-learning-2/), [python](https://dataaspirant.com/category/python/)



**Iris Classification with Svm Classifier**

**Svm classifier implementation in python with scikit-learn**

Support vector machine classifier is one of the most popular machine learning [classification algorithm](https://dataaspirant.com/2016/09/24/classification-clustering-alogrithms/). Svm classifier mostly used in addressing multi-classification problems. If you are not aware of the multi-classification problem below are examples of multi-classification problems.

**Multi-Classification Problem Examples:**

* Given fruit features like color, size, taste, weight, shape. Predicting the fruit type.
* By analyzing the skin, predicting the different skin disease.
* Given Google news articles, predicting the topic of the article. This could be sport, movie, tech news related article, etc.

**In short:** Multi-classification problem means having more that 2 target classes to predict.

In the first example of predicting the fruit type. The target class will have many fruits like apple, mango, orange, banana, etc. This is same with the other two examples in predicting. The problem of the new article, the target class having different topics like sport, movie, tech news ..etc

In this article, we were going to implement the svm classifier with different kernels. However, we have explained the key aspect of [support vector machine algorithm](https://dataaspirant.com/2017/01/13/support-vector-machine-algorithm/) as well we had implemented [svm classifier in R programming language](https://dataaspirant.com/2017/01/19/support-vector-machine-classifier-implementation-r-caret-package/) in our earlier posts. If you are reading this post for the first time, it’s recommended to chek out the previous post on [svm concepts](https://dataaspirant.com/tag/svm/).

To implement svm classifier in Python, we are going to use the one of most popular classification dataset which is [Iris dataset](https://en.wikipedia.org/wiki/Iris_flower_data_set).  Let’s quickly look at the features and the target variable details of the famous classification dataset.

**Iris Dataset description**



**Irises dataset for classification**

This famous classification dataset first time used in Fisher’s classic 1936 paper, [The Use of Multiple Measurements in Taxonomic Problems](http://rcs.chemometrics.ru/Tutorials/classification/Fisher.pdf). Iris dataset is having 4 features of iris flower and one target class.

**The 4 features are**

* SepalLengthCm
* SepalWidthCm
* PetalLengthCm
* PetalWidthCm

**The target class**

The flower species type is the target class and it having 3 types

* Setosa
* Versicolor
* Virginica

The idea of implementing svm classifier in Python is to use the iris features to train an svm classifier and use the trained svm model to predict the Iris species type. To begin with let’s try to load the Iris dataset. We are going to use the iris data from [Scikit-Learn package](http://scikit-learn.org/).

**Analyzing Iris dataset**

To successfully run the below scripts in your machine you need to install the required packages. It’s better to please go through the [python machine learning packages installation](https://dataaspirant.com/2014/11/01/python-packages-for-datamining/) or [machine learning packages step up](https://dataaspirant.com/2016/03/22/python-datamining-packages-virtual-environment-setup-in-ubuntu/) before running the below scripts.

**Importing Iris dataset from Scikit-Learn**

Let’s first import the required python packages



|  |  |
| --- | --- |
| 1  2  3  4  5 | # Required Packages  from sklearn import datasets # To Get iris dataset  from sklearn import svm     # To fit the svm classifier  import numpy as np  import matplotlib.pyplot as plt            # To visuvalizing the data |

Now let’s import the **iris dataset**



|  |  |
| --- | --- |
| 1  2 | # import iris data to model Svm classifier  iris\_dataset = datasets.load\_iris() |

Using the **DESCR** key over the iris\_dataset, we can get description of the dataset



|  |  |
| --- | --- |
| 1 | print "Iris data set Description :: ", iris\_dataset['DESCR'] |

**Output**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61 | Iris data set Description ::  Iris Plants Database    Notes  -----  Data Set Characteristics:      :Number of Instances: 150 (50 in each of three classes)      :Number of Attributes: 4 numeric, predictive attributes and the class      :Attribute Information:          - sepal length in cm          - sepal width in cm          - petal length in cm          - petal width in cm          - class:                  - Iris-Setosa                  - Iris-Versicolour                  - Iris-Virginica      :Summary Statistics:        ============== ==== ==== ======= ===== ====================                      Min  Max   Mean    SD   Class Correlation      ============== ==== ==== ======= ===== ====================      sepal length:   4.3  7.9   5.84   0.83    0.7826      sepal width:    2.0  4.4   3.05   0.43   -0.4194      petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)      petal width:    0.1  2.5   1.20  0.76     0.9565  (high!)      ============== ==== ==== ======= ===== ====================        :Missing Attribute Values: None      :Class Distribution: 33.3% for each of 3 classes.      :Creator: R.A. Fisher      :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)      :Date: July, 1988    This is a copy of UCI ML iris datasets.  http://archive.ics.uci.edu/ml/datasets/Iris    The famous Iris database, first used by Sir R.A Fisher    This is perhaps the best known database to be found in the  pattern recognition literature.  Fisher's paper is a classic in the field and  is referenced frequently to this day.  (See Duda & Hart, for example.)  The  data set contains 3 classes of 50 instances each, where each class refers to a  type of iris plant.  One class is linearly separable from the other 2; the  latter are NOT linearly separable from each other.    References  ----------     - Fisher,R.A. "The use of multiple measurements in taxonomic problems"       Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to       Mathematical Statistics" (John Wiley, NY, 1950).     - Duda,R.O., & Hart,P.E. (1973) Pattern Classification and Scene Analysis.       (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.     - 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, Vol. PAMI-2, No. 1, 67-71.     - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions       on Information Theory, May 1972, 431-433.     - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II       conceptual clustering system finds 3 classes in the data.     - Many, many more ... |

Now let’s get the iris features and the target classes



|  |  |
| --- | --- |
| 1 | print "Iris feature data :: ", iris\_dataset['data'] |

**Output**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150 | Iris feature data ::  [[ 5.1  3.5  1.4  0.2]  [ 4.9  3.   1.4  0.2]  [ 4.7  3.2  1.3  0.2]  [ 4.6  3.1  1.5  0.2]  [ 5.   3.6  1.4  0.2]  [ 5.4  3.9  1.7  0.4]  [ 4.6  3.4  1.4  0.3]  [ 5.   3.4  1.5  0.2]  [ 4.4  2.9  1.4  0.2]  [ 4.9  3.1  1.5  0.1]  [ 5.4  3.7  1.5  0.2]  [ 4.8  3.4  1.6  0.2]  [ 4.8  3.   1.4  0.1]  [ 4.3  3.   1.1  0.1]  [ 5.8  4.   1.2  0.2]  [ 5.7  4.4  1.5  0.4]  [ 5.4  3.9  1.3  0.4]  [ 5.1  3.5  1.4  0.3]  [ 5.7  3.8  1.7  0.3]  [ 5.1  3.8  1.5  0.3]  [ 5.4  3.4  1.7  0.2]  [ 5.1  3.7  1.5  0.4]  [ 4.6  3.6  1.   0.2]  [ 5.1  3.3  1.7  0.5]  [ 4.8  3.4  1.9  0.2]  [ 5.   3.   1.6  0.2]  [ 5.   3.4  1.6  0.4]  [ 5.2  3.5  1.5  0.2]  [ 5.2  3.4  1.4  0.2]  [ 4.7  3.2  1.6  0.2]  [ 4.8  3.1  1.6  0.2]  [ 5.4  3.4  1.5  0.4]  [ 5.2  4.1  1.5  0.1]  [ 5.5  4.2  1.4  0.2]  [ 4.9  3.1  1.5  0.1]  [ 5.   3.2  1.2  0.2]  [ 5.5  3.5  1.3  0.2]  [ 4.9  3.1  1.5  0.1]  [ 4.4  3.   1.3  0.2]  [ 5.1  3.4  1.5  0.2]  [ 5.   3.5  1.3  0.3]  [ 4.5  2.3  1.3  0.3]  [ 4.4  3.2  1.3  0.2]  [ 5.   3.5  1.6  0.6]  [ 5.1  3.8  1.9  0.4]  [ 4.8  3.   1.4  0.3]  [ 5.1  3.8  1.6  0.2]  [ 4.6  3.2  1.4  0.2]  [ 5.3  3.7  1.5  0.2]  [ 5.   3.3  1.4  0.2]  [ 7.   3.2  4.7  1.4]  [ 6.4  3.2  4.5  1.5]  [ 6.9  3.1  4.9  1.5]  [ 5.5  2.3  4.   1.3]  [ 6.5  2.8  4.6  1.5]  [ 5.7  2.8  4.5  1.3]  [ 6.3  3.3  4.7  1.6]  [ 4.9  2.4  3.3  1. ]  [ 6.6  2.9  4.6  1.3]  [ 5.2  2.7  3.9  1.4]  [ 5.   2.   3.5  1. ]  [ 5.9  3.   4.2  1.5]  [ 6.   2.2  4.   1. ]  [ 6.1  2.9  4.7  1.4]  [ 5.6  2.9  3.6  1.3]  [ 6.7  3.1  4.4  1.4]  [ 5.6  3.   4.5  1.5]  [ 5.8  2.7  4.1  1. ]  [ 6.2  2.2  4.5  1.5]  [ 5.6  2.5  3.9  1.1]  [ 5.9  3.2  4.8  1.8]  [ 6.1  2.8  4.   1.3]  [ 6.3  2.5  4.9  1.5]  [ 6.1  2.8  4.7  1.2]  [ 6.4  2.9  4.3  1.3]  [ 6.6  3.   4.4  1.4]  [ 6.8  2.8  4.8  1.4]  [ 6.7  3.   5.   1.7]  [ 6.   2.9  4.5  1.5]  [ 5.7  2.6  3.5  1. ]  [ 5.5  2.4  3.8  1.1]  [ 5.5  2.4  3.7  1. ]  [ 5.8  2.7  3.9  1.2]  [ 6.   2.7  5.1  1.6]  [ 5.4  3.   4.5  1.5]  [ 6.   3.4  4.5  1.6]  [ 6.7  3.1  4.7  1.5]  [ 6.3  2.3  4.4  1.3]  [ 5.6  3.   4.1  1.3]  [ 5.5  2.5  4.   1.3]  [ 5.5  2.6  4.4  1.2]  [ 6.1  3.   4.6  1.4]  [ 5.8  2.6  4.   1.2]  [ 5.   2.3  3.3  1. ]  [ 5.6  2.7  4.2  1.3]  [ 5.7  3.   4.2  1.2]  [ 5.7  2.9  4.2  1.3]  [ 6.2  2.9  4.3  1.3]  [ 5.1  2.5  3.   1.1]  [ 5.7  2.8  4.1  1.3]  [ 6.3  3.3  6.   2.5]  [ 5.8  2.7  5.1  1.9]  [ 7.1  3.   5.9  2.1]  [ 6.3  2.9  5.6  1.8]  [ 6.5  3.   5.8  2.2]  [ 7.6  3.   6.6  2.1]  [ 4.9  2.5  4.5  1.7]  [ 7.3  2.9  6.3  1.8]  [ 6.7  2.5  5.8  1.8]  [ 7.2  3.6  6.1  2.5]  [ 6.5  3.2  5.1  2. ]  [ 6.4  2.7  5.3  1.9]  [ 6.8  3.   5.5  2.1]  [ 5.7  2.5  5.   2. ]  [ 5.8  2.8  5.1  2.4]  [ 6.4  3.2  5.3  2.3]  [ 6.5  3.   5.5  1.8]  [ 7.7  3.8  6.7  2.2]  [ 7.7  2.6  6.9  2.3]  [ 6.   2.2  5.   1.5]  [ 6.9  3.2  5.7  2.3]  [ 5.6  2.8  4.9  2. ]  [ 7.7  2.8  6.7  2. ]  [ 6.3  2.7  4.9  1.8]  [ 6.7  3.3  5.7  2.1]  [ 7.2  3.2  6.   1.8]  [ 6.2  2.8  4.8  1.8]  [ 6.1  3.   4.9  1.8]  [ 6.4  2.8  5.6  2.1]  [ 7.2  3.   5.8  1.6]  [ 7.4  2.8  6.1  1.9]  [ 7.9  3.8  6.4  2. ]  [ 6.4  2.8  5.6  2.2]  [ 6.3  2.8  5.1  1.5]  [ 6.1  2.6  5.6  1.4]  [ 7.7  3.   6.1  2.3]  [ 6.3  3.4  5.6  2.4]  [ 6.4  3.1  5.5  1.8]  [ 6.   3.   4.8  1.8]  [ 6.9  3.1  5.4  2.1]  [ 6.7  3.1  5.6  2.4]  [ 6.9  3.1  5.1  2.3]  [ 5.8  2.7  5.1  1.9]  [ 6.8  3.2  5.9  2.3]  [ 6.7  3.3  5.7  2.5]  [ 6.7  3.   5.2  2.3]  [ 6.3  2.5  5.   1.9]  [ 6.5  3.   5.2  2. ]  [ 6.2  3.4  5.4  2.3]  [ 5.9  3.   5.1  1.8]] |

As we are said, these are 4 features first 2 were sepal length, sepal width and the next 2 were petal length and width. Now let’s check the target data



|  |  |
| --- | --- |
| 1 | print "Iris target :: ", iris\_dataset['target'] |

**Output**



|  |  |
| --- | --- |
| 1  2  3  4  5 | Iris target ::  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2  2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  2 2] |

**Visualizing the Iris dataset**

Let’s take the individual features like sepal, petal length, and weight and let’s visualize the corresponding target classes with different colors.

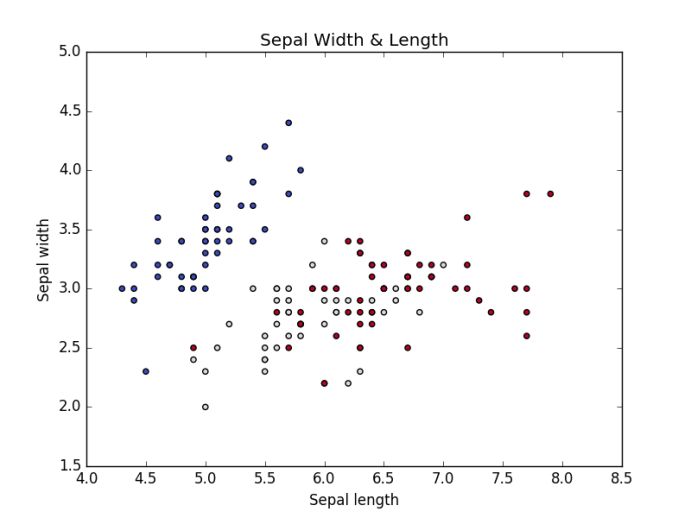
**Visualizing the relationship between sepal and target classes**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | def visuvalize\_sepal\_data():  iris = datasets.load\_iris()  X = iris.data[:, :2]  # we only take the first two features.  y = iris.target  plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)  plt.xlabel('Sepal length')  plt.ylabel('Sepal width')  plt.title('Sepal Width & Length')  plt.show()    visuvalize\_sepal\_data() |

To visualize the Sepal length, width and corresponding target classes we can create a function with name **visuvalize\_sepal\_data.**At the beginning, we are loading the iris dataset to **iris** variable. Next, we are storing the first 2 features in iris dataset which are sepal length and sepal width to variable **x.**Then we are storing the corresponding target values in variable**y**.

As we have seen target variable contains values like **0, 1,2** each value represents the iris flower species type. Then we are plotting the points on **XY** axis on **X-axis** we are plotting Sepal Length values. On Y-axis we are plotting Sepal Width values. If you follow installing instruction correctly on [Installing Python machine learning packages](https://dataaspirant.com/2014/11/01/python-packages-for-datamining/) and run the above code, you will get the below image.



**Iris Sepal length & width Vs Iris Species type**

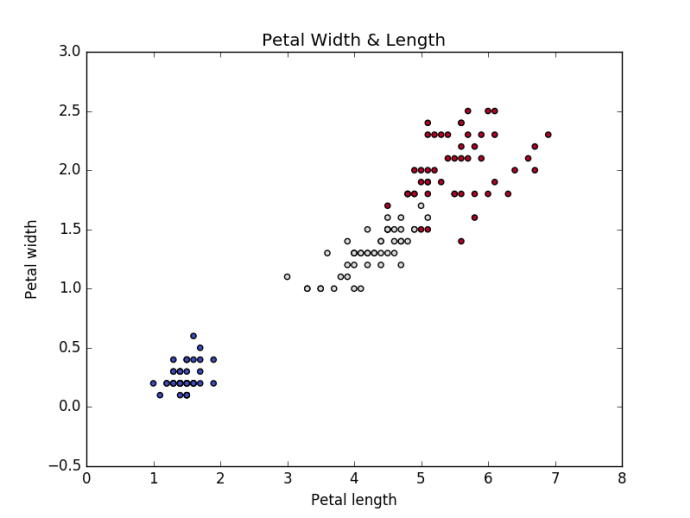
Let’s create the similar kind of graph for Petal length and width

**Visualizing the relationship between Petal and target classes**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | def visuvalize\_petal\_data():  iris = datasets.load\_iris()  X = iris.data[:, 2:]  # we only take the last two features.  y = iris.target  plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)  plt.xlabel('Petal length')  plt.ylabel('Petal width')  plt.title('Petal Width & Length')  plt.show()    visuvalize\_petal\_data() |

If we run the above code, we will get the below graph.



**Iris Petal length & width Vs Species Type**

As we have successfully visualized the behavior of target class (iris species type) with respect to **Sepal** length and width as well as with respect to **Petal** length and width. Now let’s model different kernel Svm classifier by considering only the Sepal features (Length and Width) and only the Petal features (Lenght and Width)

**Modeling Different Kernel Svm classifier using Iris Sepal features**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | iris = datasets.load\_iris()  X = iris.data[:, :2]  # we only take the Sepal two features.  y = iris.target  C = 1.0  # SVM regularization parameter    # SVC with linear kernel  svc = svm.SVC(kernel='linear', C=C).fit(X, y)  # LinearSVC (linear kernel)  lin\_svc = svm.LinearSVC(C=C).fit(X, y)  # SVC with RBF kernel  rbf\_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)  # SVC with polynomial (degree 3) kernel  poly\_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y) |

To model different kernel svm classifier using the iris Sepal features, first, we loaded the iris dataset into **iris**variable like as we have done before. Next, we are loading the **sepal length and width** values into **X** variable, and the target values are stored in **y** variable. Once we are ready with data to model the svm classifier, we are just calling the scikit-learn svm module function with different kernels.

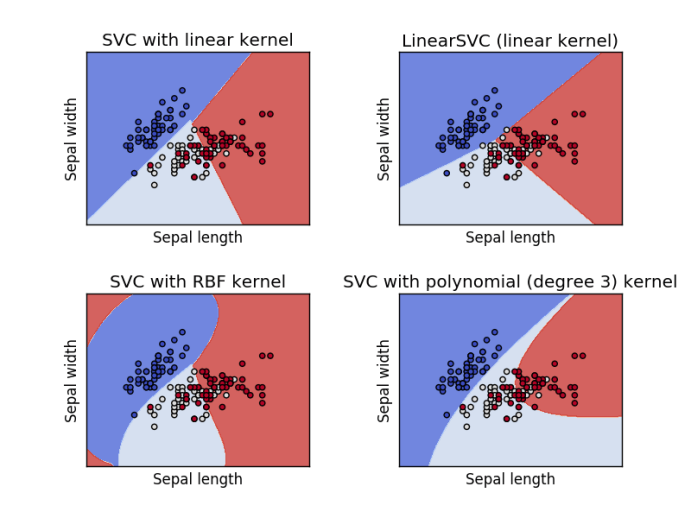
Now let’s visualize the each kernel svm classifier to understand how well the classifier fit the train features.

**Visualizing the modeled svm classifiers with Iris Sepal features**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | h = .02  # step size in the mesh    # create a mesh to plot in  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))  # title for the plots  titles = ['SVC with linear kernel',     'LinearSVC (linear kernel)',      'SVC with RBF kernel',      'SVC with polynomial (degree 3) kernel']      for i, clf in enumerate((svc, lin\_svc, rbf\_svc, poly\_svc)):  # Plot the decision boundary. For that, we will assign a color to each  # point in the mesh [x\_min, x\_max]x[y\_min, y\_max].  plt.subplot(2, 2, i + 1)  plt.subplots\_adjust(wspace=0.4, hspace=0.4)    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=plt.cm.coolwarm, alpha=0.8)    # Plot also the training points  plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)  plt.xlabel('Sepal length')  plt.ylabel('Sepal width')  plt.xlim(xx.min(), xx.max())  plt.ylim(yy.min(), yy.max())  plt.xticks(())  plt.yticks(())  plt.title(titles[i])    plt.show() |

If we run the above code, we will get the below graph. From which we can understand how well different kernel svm classifiers are modeled.



**Svm Classifier with Iris Sepal features**

From the above graphs, you can clearly understand how different kernel modeled with the same svm classifier. Now let’s model the svm classifier with **Petal features** using the same kernel we have used for modeling with Sepal features.

**Modeling Different Kernel Svm classifier using Iris Petal features**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | iris = datasets.load\_iris()  X = iris.data[:, 2:]  # we only take the last two features.  y = iris.target  C = 1.0  # SVM regularization parameter    # SVC with linear kernel  svc = svm.SVC(kernel='linear', C=C).fit(X, y)  # LinearSVC (linear kernel)  lin\_svc = svm.LinearSVC(C=C).fit(X, y)  # SVC with RBF kernel  rbf\_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)  # SVC with polynomial (degree 3) kernel  poly\_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y) |

The above code is much similar to the previously modeled svm classifiers code. The only difference is loading the Petal features into **X** variable. The remaining code is just the copy past from the previously modeled svm classifier code.

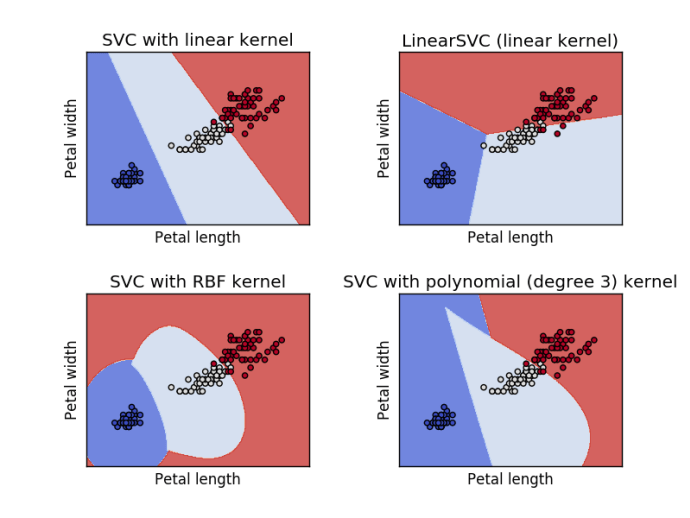
Now let’s visualize the each kernel svm classifier to understand how well the classifier fit the Petal features.

**Visualizing the modeled svm classifiers with Iris Petal features**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | h = .02  # step size in the mesh  # create a mesh to plot in  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))  # title for the plots  titles = ['SVC with linear kernel',    'LinearSVC (linear kernel)',    'SVC with RBF kernel',    'SVC with polynomial (degree 3) kernel']      for i, clf in enumerate((svc, lin\_svc, rbf\_svc, poly\_svc)):      # Plot the decision boundary. For that, we will assign a color to each      # point in the mesh [x\_min, x\_max]x[y\_min, y\_max].      plt.subplot(2, 2, i + 1)      plt.subplots\_adjust(wspace=0.4, hspace=0.4)        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=plt.cm.coolwarm, alpha=0.8)        # Plot also the training points      plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)      plt.xlabel('Petal length')      plt.ylabel('Petal width')      plt.xlim(xx.min(), xx.max())      plt.ylim(yy.min(), yy.max())      plt.xticks(())      plt.yticks(())      plt.title(titles[i])    plt.show() |

If we run the above code, we will get the below graph. From which we can understand how well different kernel svm classifiers are modeled.



**Svm Classifier with Iris Petal features**

This is how the modeled svm classifier looks like when we only use the petal width and length to model. With this, we came to an end. Before put an end to the post lets quickly look how to use the modeled svm classifier to predict iris flow categories.

**Predicting iris flower category**

To Identify the iris flow type using the modeled svm classifier, we need to call the predict function over the fitted model. For example, if you want to predict the iris flower category using the **lin\_svc** model. We need to call **lin\_svc.predict(with the features)**. In our case, these features will include the sepal length and width or petal length and width. If you are not clear with the using the predict function correctly you check [knn classifier with scikit-learn](https://dataaspirant.com/2016/12/30/k-nearest-neighbor-implementation-scikit-learn/).

**Conclusion**

In this article, we learned how to model the support vector machine classifier using different, kernel with Python scikit-learn package. In the process, we have learned how to visualize the data points and how to visualize the modeled svm classifier for understanding the how well the fitted modeled were fit with the training dataset.

**Related Articles**

* [Introduction to support vector machine classifier](https://dataaspirant.com/2017/01/13/support-vector-machine-algorithm/)
* [support vector machine classifier implementation in R programming language](https://dataaspirant.com/2017/01/19/support-vector-machine-classifier-implementation-r-caret-package/)

**Related**

[](https://dataaspirant.com/2017/01/13/support-vector-machine-algorithm/)

[Svm classifier, Introduction to support vector machine algorithm](https://dataaspirant.com/2017/01/13/support-vector-machine-algorithm/)

January 13, 2017

In "Data Science"

[](https://dataaspirant.com/2017/04/21/visualize-decision-tree-python-graphviz/)

[visualize decision tree in python with graphviz](https://dataaspirant.com/2017/04/21/visualize-decision-tree-python-graphviz/)

April 21, 2017

In "Machine Learning"

[](https://dataaspirant.com/2016/09/24/classification-clustering-alogrithms/)

[classification and clustering algorithms](https://dataaspirant.com/2016/09/24/classification-clustering-alogrithms/)

September 24, 2016

In "Data Science"

* [classification algorithms](https://dataaspirant.com/tag/classification-algorithms/)
* [Svm](https://dataaspirant.com/tag/svm/)

[**Previous Post  
Support Vector Machine Classifier Implementation in R with caret package**](https://dataaspirant.com/2017/01/19/support-vector-machine-classifier-implementation-r-caret-package/)

[Data Science](https://dataaspirant.com/category/data-science-2/)

[**Next Post  
How Decision Tree Algorithm works**](https://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/)

[Data Science](https://dataaspirant.com/category/data-science-2/)

Top of Form



Bottom of Form

**AWARDED TOP 75 DATA SCIENCE BLOG**



Dataaspirant awarded top 75 data science blog

**FOLLOW AUTHOR**

**BUILD YOUR CAREER IN AI WITH ANDREW NG DEEP LEARNING COURSES**

[](https://dataaspirant.com/recommends/andrew-ng/deeplearning-course-1-coursera/)

Andrew ng Deep learning courses

**MOST POPULAR POSTS**

* [Building Decision Tree Algorithm in Python with scikit learn](https://dataaspirant.com/2017/02/01/decision-tree-algorithm-python-with-scikit-learn/)
* [How Decision Tree Algorithm works](https://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/)
* [How the random forest algorithm works in machine learning](https://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/)
* [Difference Between Softmax Function and Sigmoid Function](https://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/)
* [Five most popular similarity measures implementation in python](https://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/)
* [Building Random Forest Classifier with Python Scikit learn](https://dataaspirant.com/2017/06/26/random-forest-classifier-python-scikit-learn/)
* [visualize decision tree in python with graphviz](https://dataaspirant.com/2017/04/21/visualize-decision-tree-python-graphviz/)
* [Supervised and Unsupervised learning](https://dataaspirant.com/2014/09/19/supervised-and-unsupervised-learning/)
* [Support Vector Machine Classifier Implementation in R with caret package](https://dataaspirant.com/2017/01/19/support-vector-machine-classifier-implementation-r-caret-package/)
* [How the Naive Bayes Classifier works in Machine Learning](https://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/)