

▼ Adnaan Shaikh TYA164

```
from sklearn import datasets
from sklearn import svm
import numpy as np
import matplotlib.pyplot as plt
```

```
iris_dataset = datasets.load_iris()
```

```
print("Iris data set Description :: ", iris_dataset['DESCR'])
```

```
➞ Iris data set Description :: .. _iris_dataset:
```

```
Iris plants dataset
```

```
-----
```

```
**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
```

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken from Fisher's paper. Note that it's the same as in R, but not as in the UCI

Machine Learning Repository, which has two wrong data points.

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.

.. topic:: 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.
- Dasarthy, 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.

```
print("Iris feature data :: ", iris_dataset['data'])
```

```
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]]
```

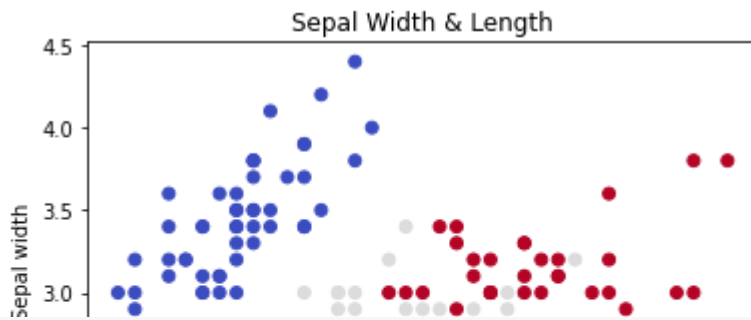
[5.5 4.2 1.4 0.2]
[4.9 3.1 1.5 0.2]
[5. 3.2 1.2 0.2]
[5.5 3.5 1.3 0.2]
[4.9 3.6 1.4 0.1]
[4.4 3. 1.3 0.2]
[5.1 3.4 1.5 0.2]
[5. 3.5 1.3 0.3]
[4.5 2.3 1.3 0.3]
[4.4 3.2 1.3 0.2]
[5. 3.5 1.6 0.6]
[5.1 3.8 1.9 0.4]
[4.8 3. 1.4 0.3]
[5.1 3.8 1.6 0.2]
[4.6 3.2 1.4 0.2]
[5.3 3.7 1.5 0.2]
[5. 3.3 1.4 0.2]
[7. 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. 1]

```
print("Iris target :: ", iris_dataset['target'])
```

[illegible]

```
def visuvalize_sepal_data():
    iris = datasets.load_iris()
    X = iris.data[:, :2]
    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()
```



```
def visuvalize_petal_data():
    iris = datasets.load_iris()
    X = iris.data[:, 2:]
    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()
```



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 (Length and Width)

```
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
C = 1.0

svc = svm.SVC(kernel='linear', C=C).fit(X, y)
```

```
lin_svc = svm.LinearSVC(C=C).fit(X, y)
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)
poly_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning: Li
ConvergenceWarning,
```

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.

```
h = .02

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))

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)):

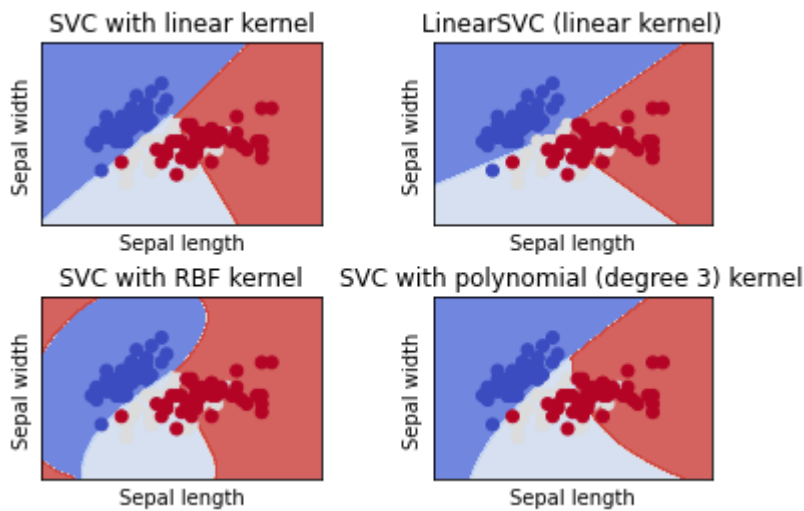
    plt.subplot(2, 2, i + 1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)

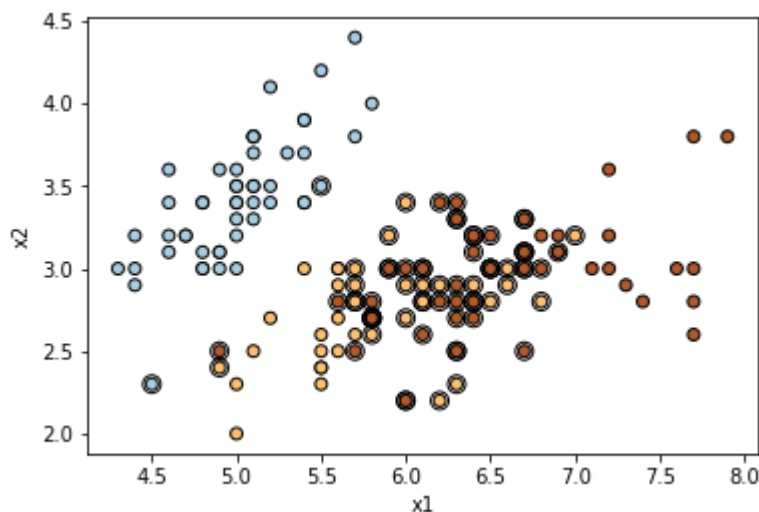
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
    plt.xlabel('Sepal length')
    plt.ylabel('Sepal width')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])
```

```
plt.show()
```



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.

```
plt.scatter(clf.support_vectors_[0], clf.support_vectors_[1], s=80, facecolors="none", zorder=10)
plt.scatter(X[:, 0], X[:, 1], c=y, zorder=10, cmap=plt.cm.Paired, edgecolors="k")
plt.xlabel("x1")
plt.ylabel("x2")
plt.show()
```



✓ 1s completed at 23:02

● ×