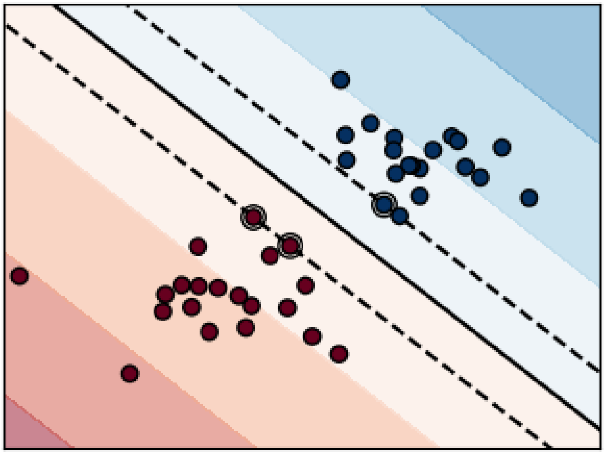
Lab11

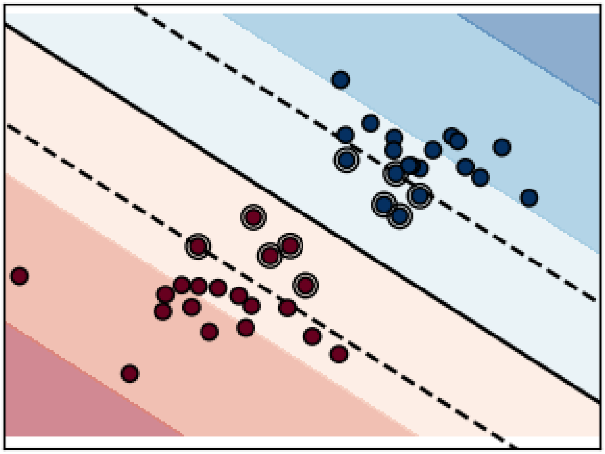
September 26, 2022

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from sklearn import datasets, svm
from sklearn.multiclass import OneVsRestClassifier, OneVsOneClassifier
```

1 Optimal Separating Hyperplanes

[ ]:
```python
# Code source: Gaël Varoquaux
# Modified for documentation by Jaques Grobler
# License: BSD 3 clause

# we create 40 separable points
np.random.seed(0)
X = np.r_[np.random.randn(20, 2) - [2, 2], np.random.randn(20, 2) + [2, 2]]
Y = [0] * 20 + [1] * 20

# figure number
fignum = 1

# fit the model
for name, penalty in (("unreg", 1), ("reg", 0.05)):

    clf = svm.SVC(kernel="linear", C=penalty)
    clf.fit(X, Y)

    # get the separating hyperplane
    w = clf.coef_[0]
    a = -w[0] / w[1]
    xx = np.linspace(-5, 5)
    yy = a * xx - (clf.intercept_[0]) / w[1]

    # plot the parallels to the separating hyperplane that pass through the
    # support vectors (margin away from hyperplane in direction
    # perpendicular to hyperplane). This is sqrt(1+a^2) away vertically in
    # 2-d.
    margin = 1 / np.sqrt(np.sum(clf.coef_**2))
    yy_down = yy - np.sqrt(1 + a**2) * margin
    yy_up = yy + np.sqrt(1 + a**2) * margin

    # plot the line, the points, and the nearest vectors to the plane
    plt.figure(fignum, figsize=(4, 3))
    plt.clf()
    plt.plot(xx, yy, "k-")
    plt.plot(xx, yy_down, "k--")
    plt.plot(xx, yy_up, "k--")

    plt.scatter(
        clf.support_vectors_[:, 0],
        clf.support_vectors_[:, 1],
        s=80,
        facecolors="none",
        zorder=10,
        edgecolors="k",
        cmap=cm.get_cmap("RdBu"),
    )
    plt.scatter(
        X[:, 0], X[:, 1], c=Y, zorder=10, cmap=cm.get_cmap("RdBu"),␣
 ↪edgecolors="k"
    )

    plt.axis("tight")
    x_min = -4.8
    x_max = 4.2
    y_min = -6
    y_max = 6

    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)

    # Put the result into a contour plot
    plt.contourf(XX, YY, Z, cmap=cm.get_cmap("RdBu"), alpha=0.5,␣
 ↪linestyles=["-"])

    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)

    plt.xticks(())
    plt.yticks(())
    fignum = fignum + 1

plt.show()
```



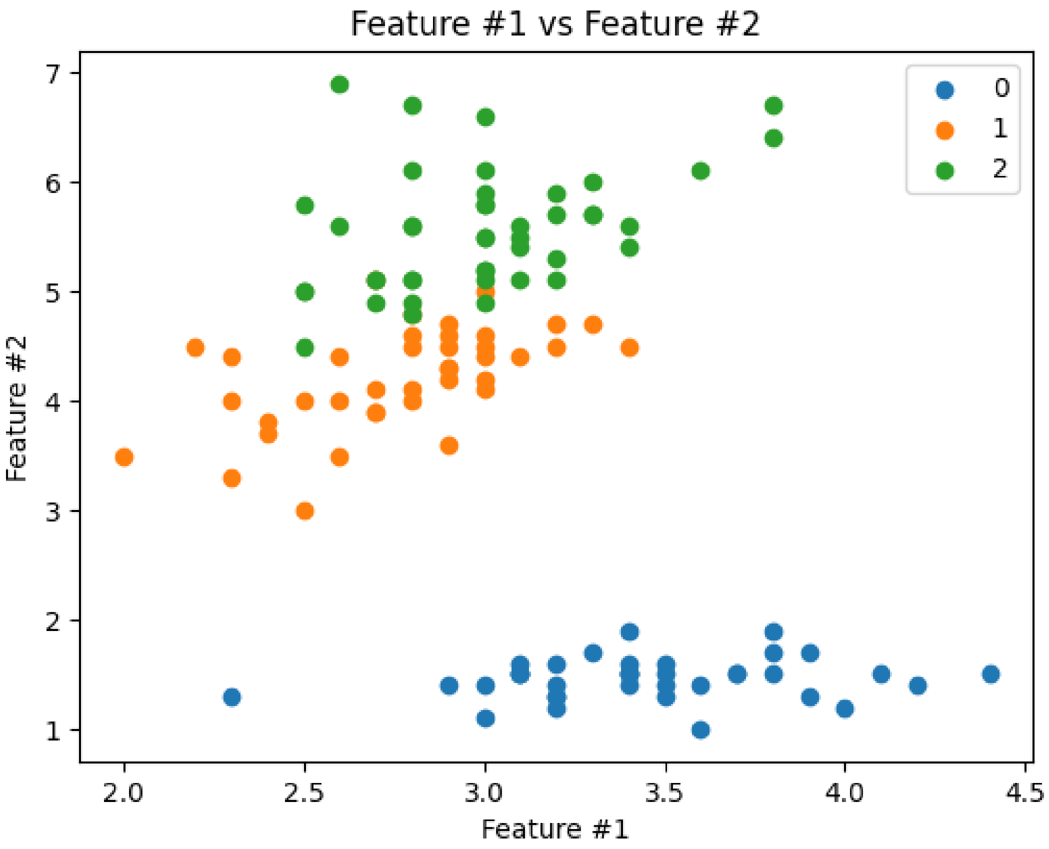


2 Multiclass Support Vector Classification

[ ]:

```
#Load data
x, y = datasets.load_iris(return_X_y=True)

#Indices to keep
ind1 = 1
ind2 = 2

#Shuffle data
ind = np.random.permutation(y.shape[0])
x = (x[ind,:])[:, [ind1, ind2]]
y = y[ind]
```

```
#Split data
n_train = int(y.shape[0]*0.8)
x_train = x[:n_train,:]
y_train = y[:n_train]
x_test  = x[n_train:,:]
y_test  = y[n_train:]


#Visualize
plt.scatter(x_train[y_train==0,0],x_train[y_train==0,1])
plt.scatter(x_train[y_train==1,0],x_train[y_train==1,1])
plt.scatter(x_train[y_train==2,0],x_train[y_train==2,1])
plt.legend([0,1,2])
plt.title('Feature #' + str(ind1) + ' vs Feature #' + str(ind2))
plt.xlabel('Feature #' + str(ind1))
plt.ylabel('Feature #' + str(ind2))
plt.show()
```

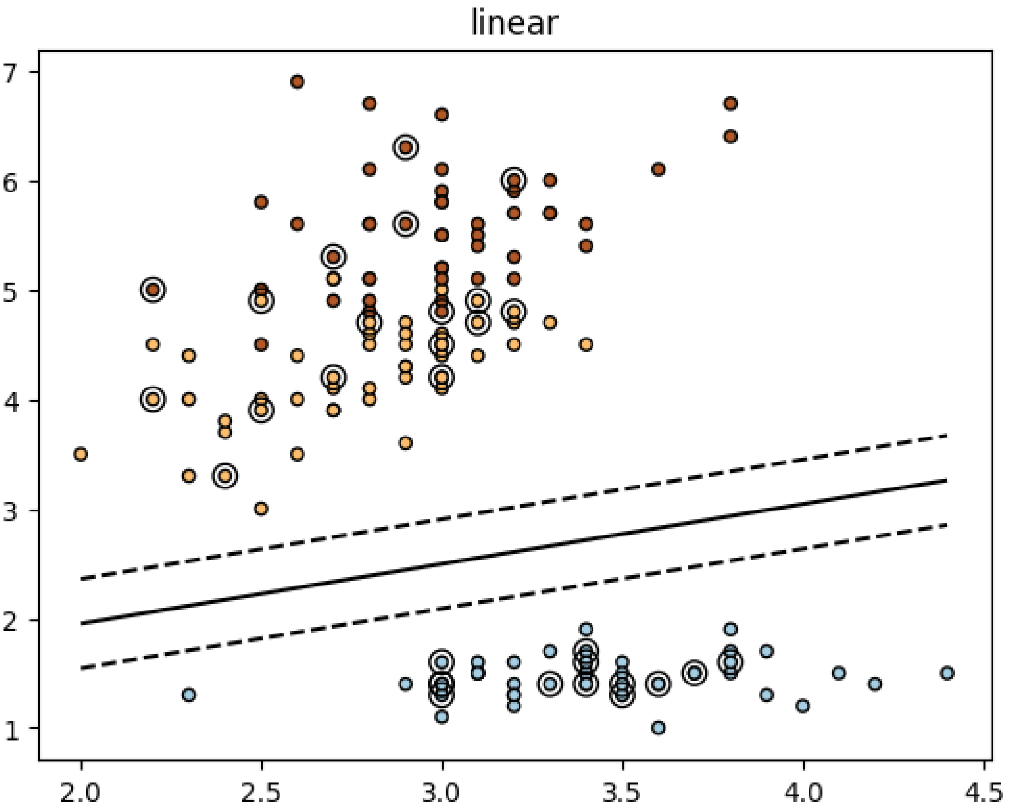


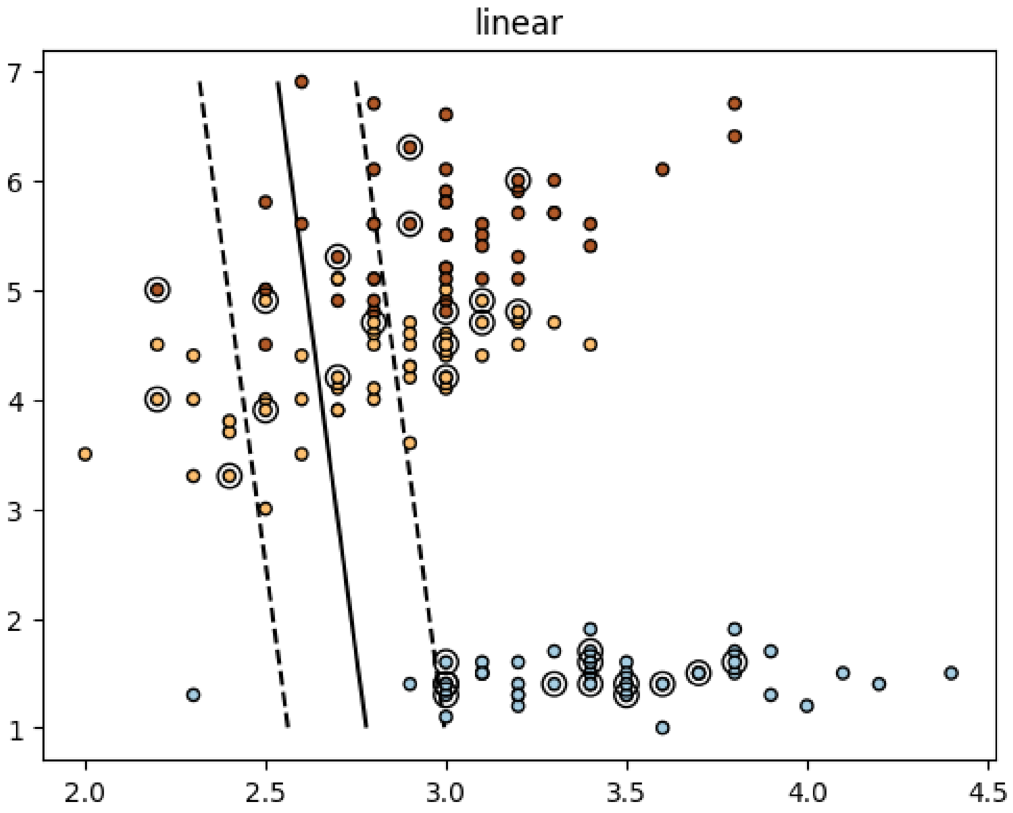
[ ]:

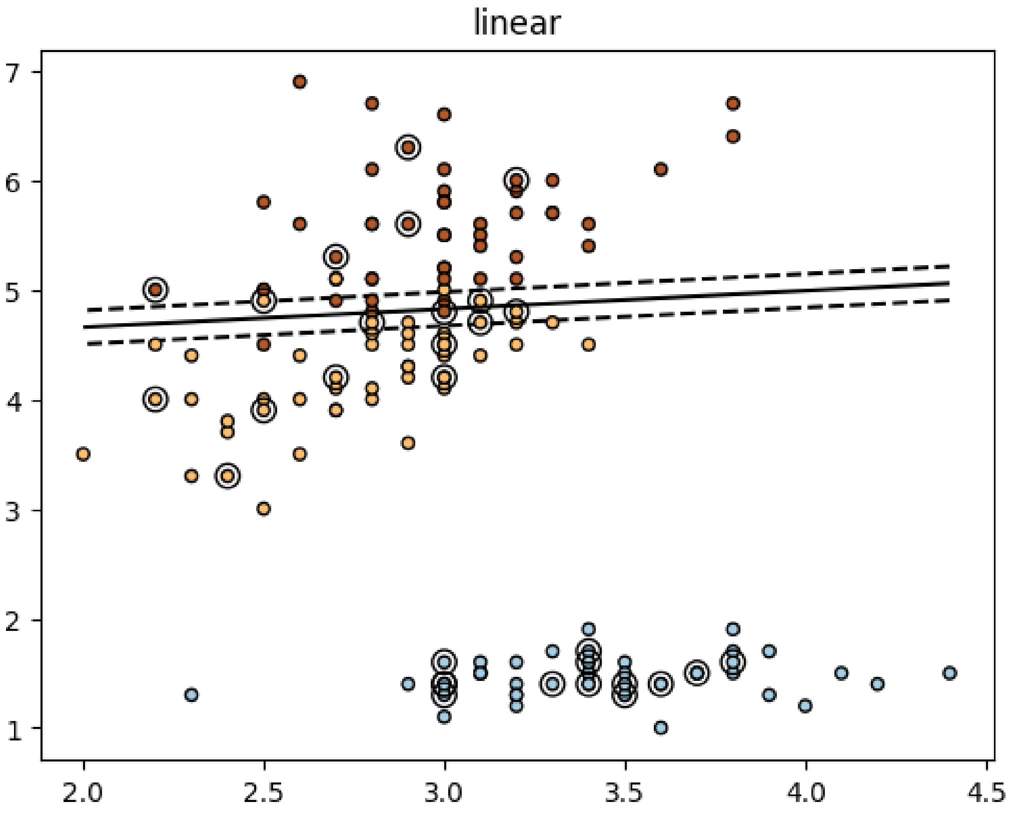
```
#Plotting function
def plot_svm_kernels(clf, X, y, X_test):
    #Plot
    plt.figure()
    plt.clf()
    plt.scatter(X[:, 0], X[:, 1], c=y, zorder=10, cmap=plt.cm.Paired,
                edgecolor='k', s=20)

    # Circle out the test data
    plt.scatter(X_test[:, 0], X_test[:, 1], s=80, facecolors='none',
                zorder=10, edgecolor='k')

    plt.axis('tight')
    x_min = X[:, 0].min()
    x_max = X[:, 0].max()
    y_min = X[:, 1].min()
    y_max = X[:, 1].max()

    XX, YY = np.mgrid[x_min:x_max:200j, y_min:y_max:200j]
    Z = clf.decision_function(np.c_[XX.ravel(), YY.ravel()])

    # Put the result into a color plot
    Z = Z.reshape(XX.shape)
    plt.contour(XX, YY, Z, colors=['k', 'k', 'k'],
                linestyles=['--', '-', '--'], levels=[-.5, 0, .5])

    plt.title(clf.kernel)
    plt.show()
```

[ ]:

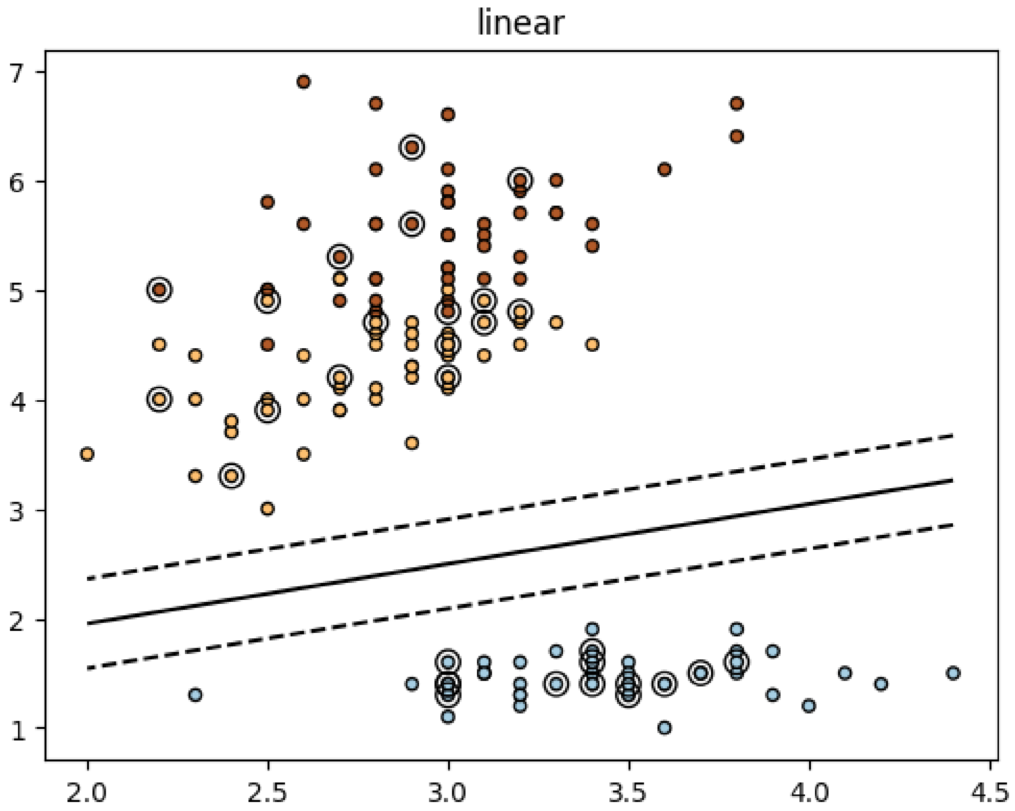
```
clf = OneVsRestClassifier(svm.SVC(kernel='linear'), n_jobs=-1)
clf.fit(x_train, y_train)
for c in clf.estimators_:
    plot_svm_kernels(c, x, y, x_test)
```

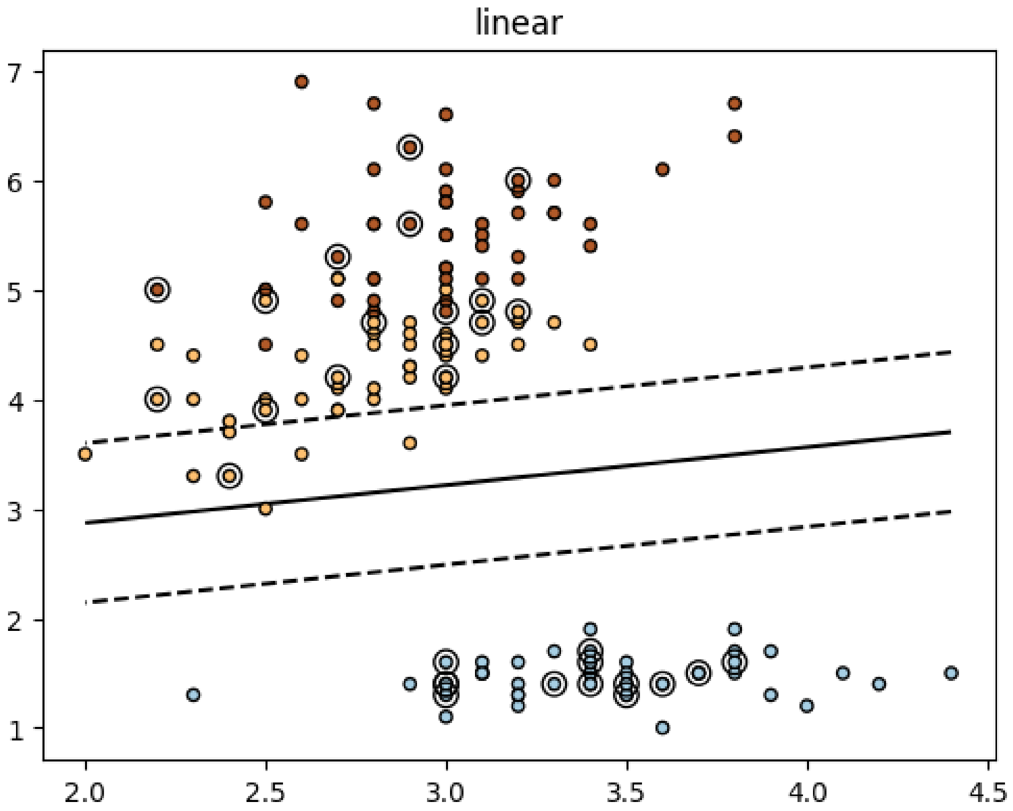

linear
7
6
5
4
3
2
1
2.0
2.5
3.0
3.5
4.0
4.5

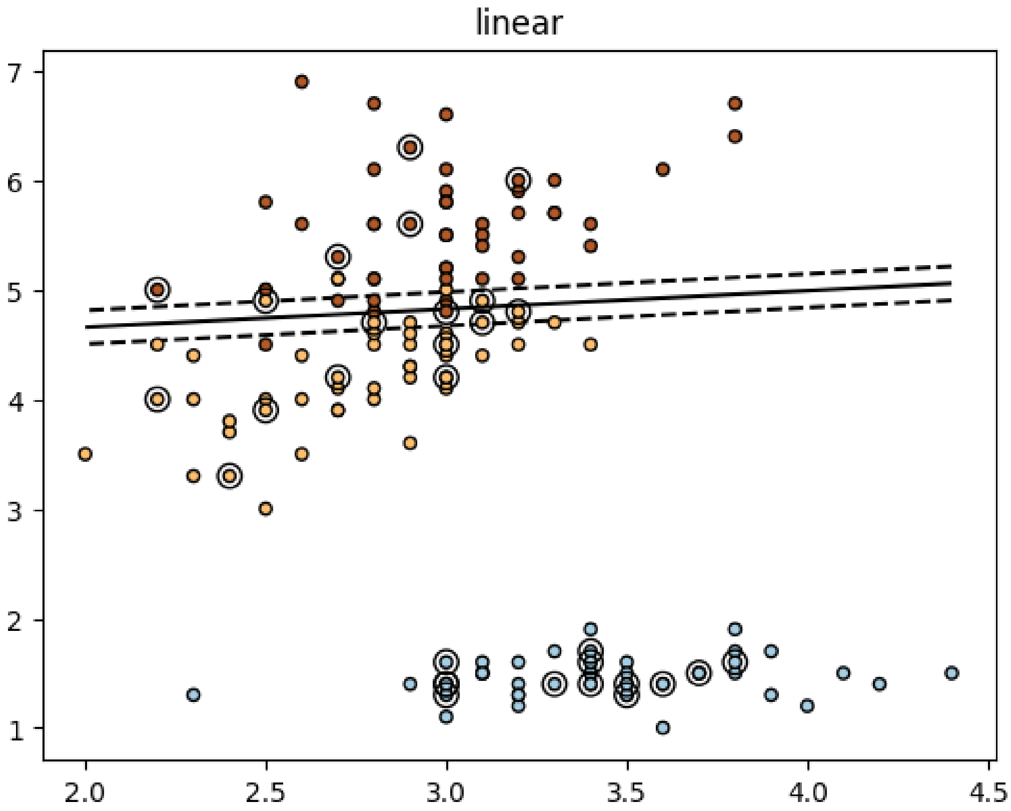

linear
7
6
5
4
3
2
1
2.0
2.5
3.0
3.5
4.0
4.5



```
[ ]: clf = OneVsOneClassifier(svm.SVC(kernel='linear'), n_jobs=-1)
     clf.fit(x_train, y_train)
     for c in clf.estimators_:
         plot_svm_kernels(c, x, y, x_test)
```


linear
7
6
5
4
3
2
1
2.0
2.5
3.0
3.5
4.0
4.5


linear
7
6
5
4
3
2
1
2.0
2.5
3.0
3.5
4.0
4.5


linear
7
6
5
4
3
2
1
2.0
2.5
3.0
3.5
4.0
4.5