Aprendizado Supervisionado

Regressão Logística

1 - Bibliotecas

In [1]:
```
%matplotlib notebook
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import neighbors
from sklearn.datasets import make_blobs


from matplotlib.colors import ListedColormap

cmap_light = ListedColormap(['#FFFFAA', '#AAFFAA', '#AAAAFF','#EFEFEF'])
cmap_bold  = ListedColormap(['#FFFF00', '#00FF00', '#0000FF','#000000'])
```

2 - Funções para plotagem

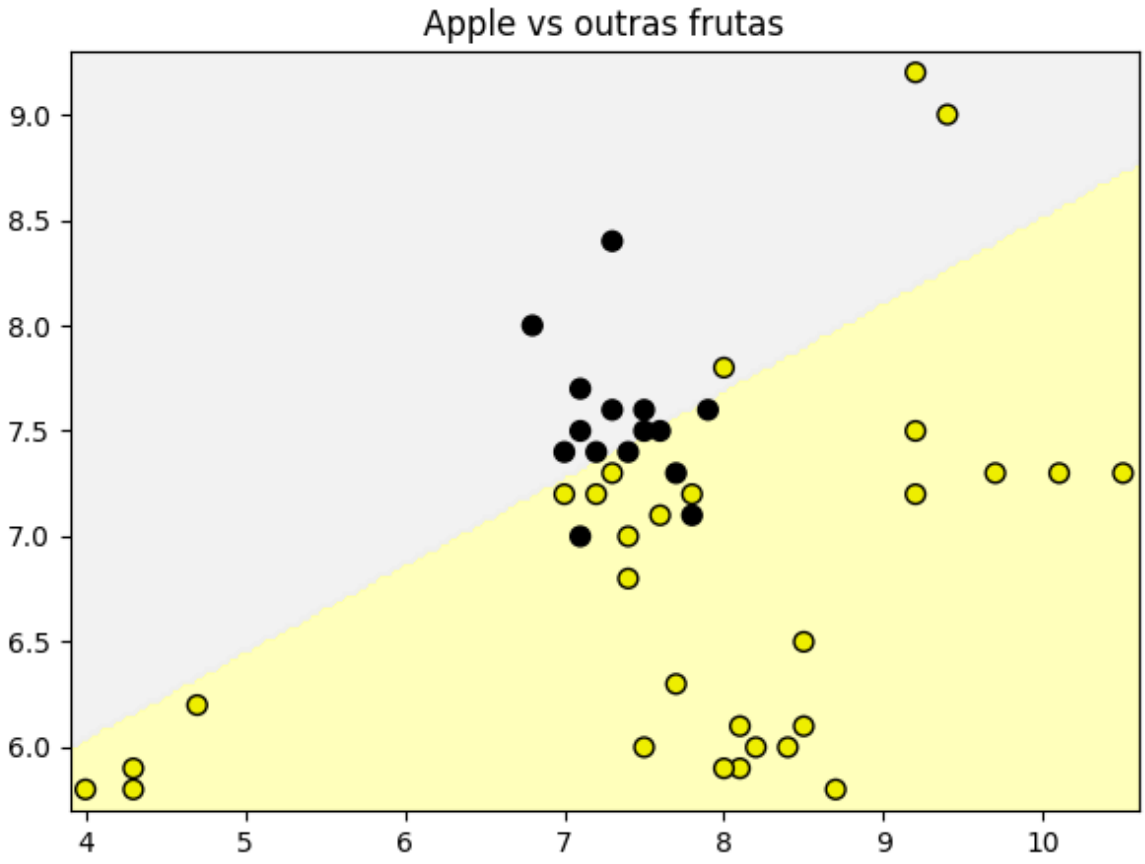
In [2]:
```
def plot_class_regions_for_classifier_subplot(clf, X, y,
                                              X_test, y_test,
                                              title, subplot,
                                              target_names = None,
                                              plot_decision_regions = True):

    numClasses = np.amax(y) + 1
    color_list_light = ['#FFFFAA', '#EFEFEF', '#AAFFAA', '#AAAAFF']
    color_list_bold = ['#EEEE00', '#000000', '#00CC00', '#0000CC']
    cmap_light = ListedColormap(color_list_light[0:numClasses])
    cmap_bold  = ListedColormap(color_list_bold[0:numClasses])

    h = 0.03
    k = 0.5
    x_plot_adjust = 0.1
    y_plot_adjust = 0.1
    plot_symbol_size = 50

    x_min = X[:, 0].min()
    x_max = X[:, 0].max()
    y_min = X[:, 1].min()
    y_max = X[:, 1].max()
    x2, y2 = np.meshgrid(np.arange(x_min-k, x_max+k, h), np.arange(y_min-k, y_max+k, h))

    P = clf.predict(np.c_[x2.ravel(), y2.ravel()])
    P = P.reshape(x2.shape)

    if plot_decision_regions:
        subplot.contourf(x2, y2, P, cmap=cmap_light, alpha = 0.8)

    subplot.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold, s=plot_symbol_size, edgecolor =
    subplot.set_xlim(x_min - x_plot_adjust, x_max + x_plot_adjust)
    subplot.set_ylim(y_min - y_plot_adjust, y_max + y_plot_adjust)
```

```
    if (X_test is not None):
        subplot.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cmap_bold, s=plot_symbo
        train_score = clf.score(X, y)
        test_score  = clf.score(X_test, y_test)
        title = title + "\nTreinamento = {:.2f}, Teste = {:.2f}".format(train_score, test_

    subplot.set_title(title)

    if (target_names is not None):
        legend_handles = []
        for i in range(0, len(target_names)):
            patch = mpatches.Patch(color=color_list_bold[i], label=target_names[i])
            legend_handles.append(patch)
        subplot.legend(loc=0, handles=legend_handles)


def plot_class_regions_for_classifier(clf, X, y,
                                      X_test=None, y_test=None,
                                      title=None,
                                      target_names = None,
                                      plot_decision_regions = True):

    numClasses = np.amax(y) + 1
    color_list_light = ['#FFFFAA', '#EFEFEF', '#AAFFAA', '#AAAAFF']
    color_list_bold = ['#EEEE00', '#000000', '#00CC00', '#0000CC']
    cmap_light = ListedColormap(color_list_light[0:numClasses])
    cmap_bold  = ListedColormap(color_list_bold[0:numClasses])

    h = 0.03
    k = 0.5
    x_plot_adjust = 0.1
    y_plot_adjust = 0.1
    plot_symbol_size = 50

    x_min = X[:, 0].min()
    x_max = X[:, 0].max()
    y_min = X[:, 1].min()
    y_max = X[:, 1].max()
    x2, y2 = np.meshgrid(np.arange(x_min-k, x_max+k, h), np.arange(y_min-k, y_max+k, h))

    P = clf.predict(np.c_[x2.ravel(), y2.ravel()])
    P = P.reshape(x2.shape)
    plt.figure()
    if plot_decision_regions:
        plt.contourf(x2, y2, P, cmap=cmap_light, alpha = 0.8)

    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold, s=plot_symbol_size, edgecolor = 'bl
    plt.xlim(x_min - x_plot_adjust, x_max + x_plot_adjust)
    plt.ylim(y_min - y_plot_adjust, y_max + y_plot_adjust)

    if (X_test is not None):
        plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cmap_bold, s=plot_symbol_si
        train_score = clf.score(X, y)
        test_score  = clf.score(X_test, y_test)
        title = title + "\nTrain score = {:.2f}, Test score = {:.2f}".format(train_score,

    if (target_names is not None):
        legend_handles = []
        for i in range(0, len(target_names)):
            patch = mpatches.Patch(color=color_list_bold[i], label=target_names[i])
            legend_handles.append(patch)
        plt.legend(loc=0, handles=legend_handles)

    if (title is not None):
        plt.title(title)
    plt.show()
```

3 - Exemplo de regressão logística

In [5]:

```
from sklearn.linear_model import LogisticRegression

fruits = pd.read_table('./Data/fruit_data_with_colors.txt')

#fruits.head()

X_fruits_2d = fruits[['height', 'width']]
y_fruits_2d = fruits['fruit_label']

fig, subaxes = plt.subplots(1, 1, figsize=(7, 5))
y_fruits_apple = y_fruits_2d == 1   # apples vs todas as outras frutas

X_train, X_test, y_train, y_test = (train_test_split(X_fruits_2d.to_numpy(),
                                                     y_fruits_apple.to_numpy(),
                                                     random_state = 0))

clf = LogisticRegression(C=100).fit(X_train, y_train)

plot_class_regions_for_classifier_subplot(clf, X_train, y_train, None,
                                          None, 'Apple vs outras frutas',
                                          subaxes)

print(clf.predict([[6,8]]))

print(clf.predict([[10,7]]))

print(clf.score(X_train, y_train))

print(clf.score(X_test, y_test))
```

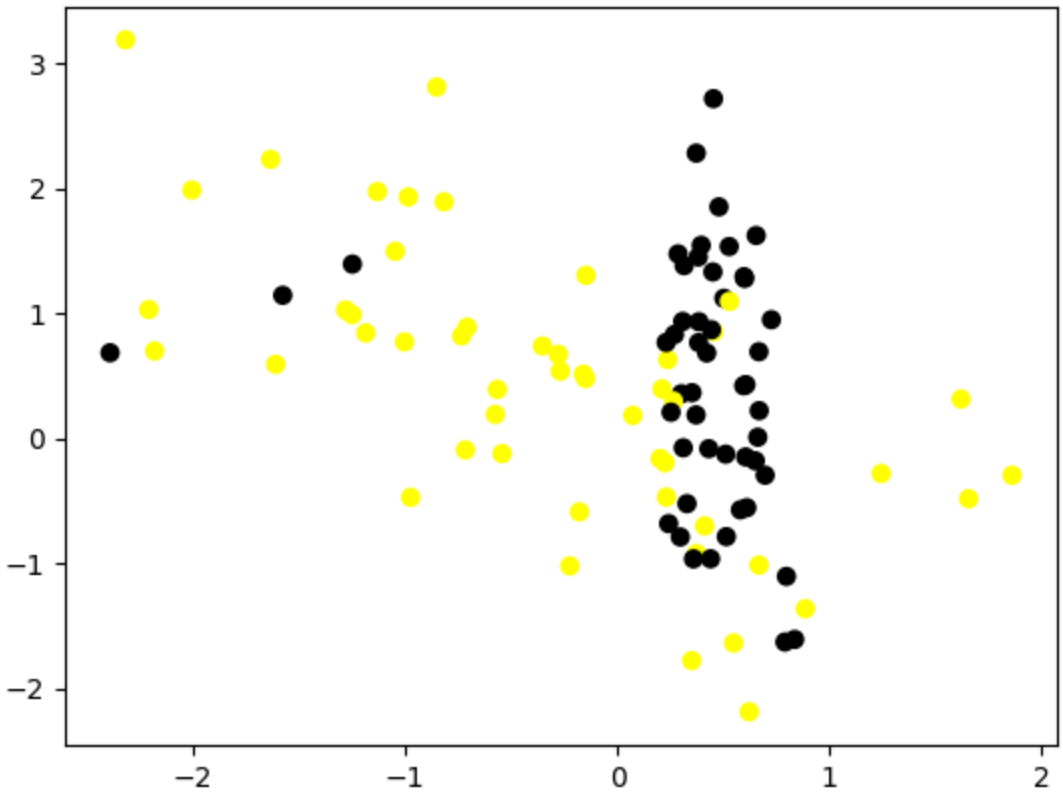



```
[ True]
[False]
0.7954545454545454
0.7333333333333333
```

4 - Criação de dados sintéticos

In [6]:

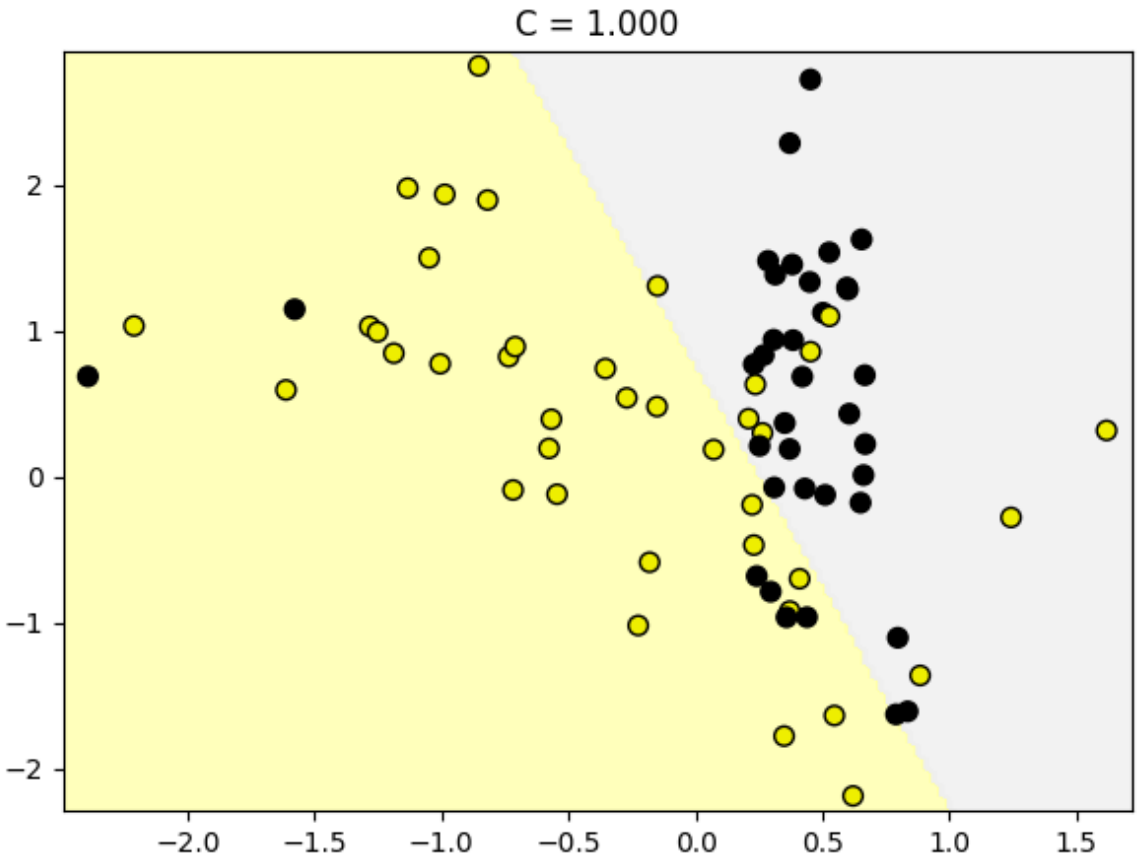
```
from sklearn.datasets import make_classification

plt.figure()

X_C2, y_C2 = make_classification(n_samples = 100, n_features=2,
                                 n_redundant=0, n_informative=2,
                                 n_clusters_per_class=1, flip_y = 0.2,
                                 class_sep = 0.5, random_state=0)

plt.scatter(X_C2[:, 0], X_C2[:, 1], c=y_C2, marker='o', cmap=cmap_bold)

plt.show()
```



5 - Regressão logística nos dados sintéticos

In [7]:

```
from sklearn.linear_model import LogisticRegression



X_train, X_test, y_train, y_test = train_test_split(X_C2, y_C2,
                                                    random_state = 0)

fig, subaxes = plt.subplots(1, 1, figsize=(7, 5))
clf = LogisticRegression().fit(X_train, y_train)
title = 'C = {:.3f}'.format(1.0)
plot_class_regions_for_classifier_subplot(clf, X_train, y_train,
                                          None, None, title, subaxes)

print(clf.score(X_train, y_train))
print(clf.score(X_test, y_test))
```

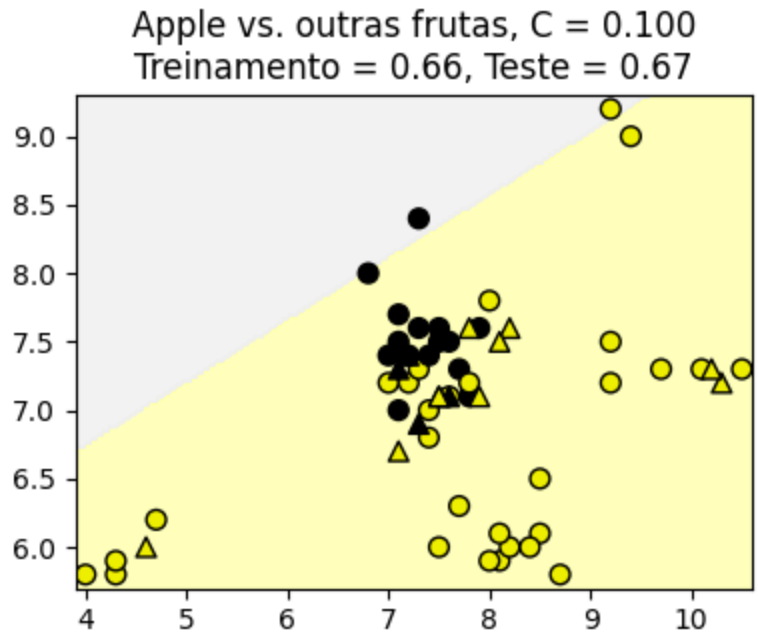


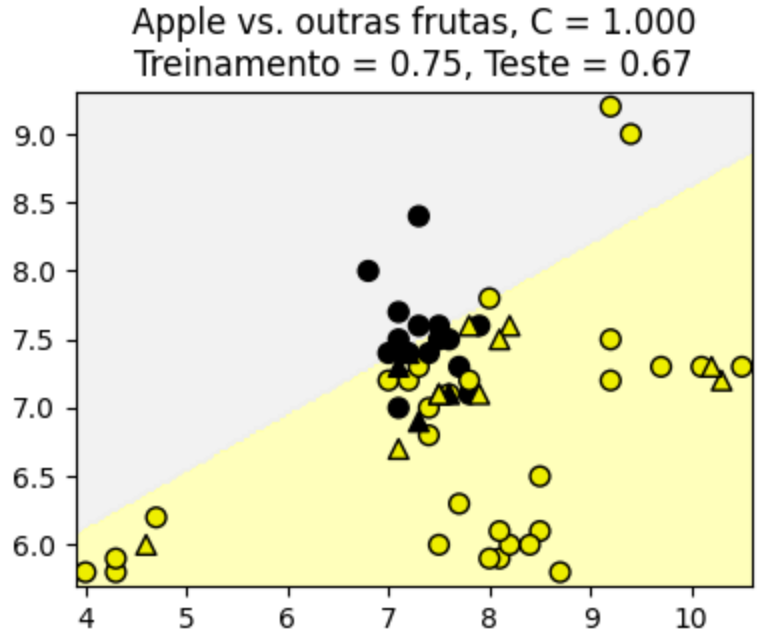

```
0.7866666666666666
0.76
```

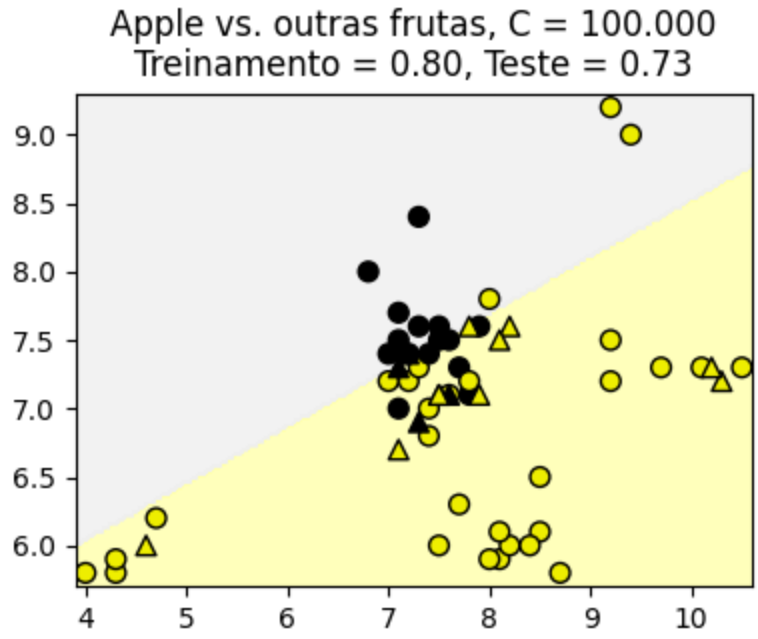
5 - Regressão logística e o parâmetro C

In [9]:

```
X_train, X_test, y_train, y_test = (
train_test_split(X_fruits_2d.to_numpy(),
                 y_fruits_apple.to_numpy(),
                 random_state=0))

fig, subaxes = plt.subplots(3, 1, figsize=(4, 10))

for this_C, subplot in zip([0.1, 1, 100], subaxes):
    clf = LogisticRegression(C=this_C).fit(X_train, y_train)
    title ='Apple vs. outras frutas, C = {:.3f}'.format(this_C)

    plot_class_regions_for_classifier_subplot(clf, X_train, y_train,
                                              X_test, y_test, title,
                                              subplot)
plt.tight_layout()
```





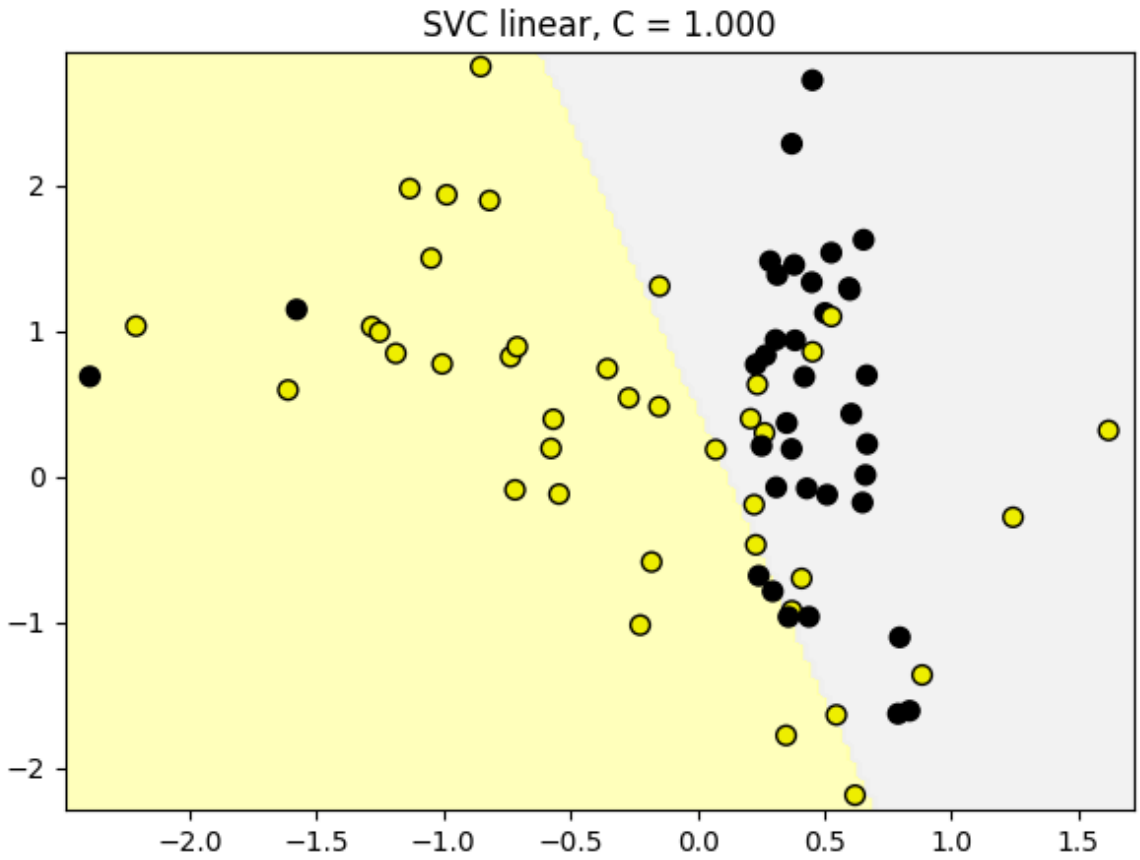


Máquinas de Vetores de Suporte

(Support Vector Machines)

6 - O exemplo linear

In [10]:
```
from sklearn.svm import SVC

X_train, X_test, y_train, y_test = train_test_split(X_C2, y_C2, random_state = 0)

fig, subaxes = plt.subplots(1, 1, figsize=(7, 5))
this_C = 1.0

clf = SVC(kernel = 'linear', C=this_C).fit(X_train, y_train)

title = 'SVC linear, C = {:.3f}'.format(this_C)
plot_class_regions_for_classifier_subplot(clf, X_train, y_train, None, None, title, subaxe

print(clf.score(X_train, y_train))
print(clf.score(X_test, y_test))
```

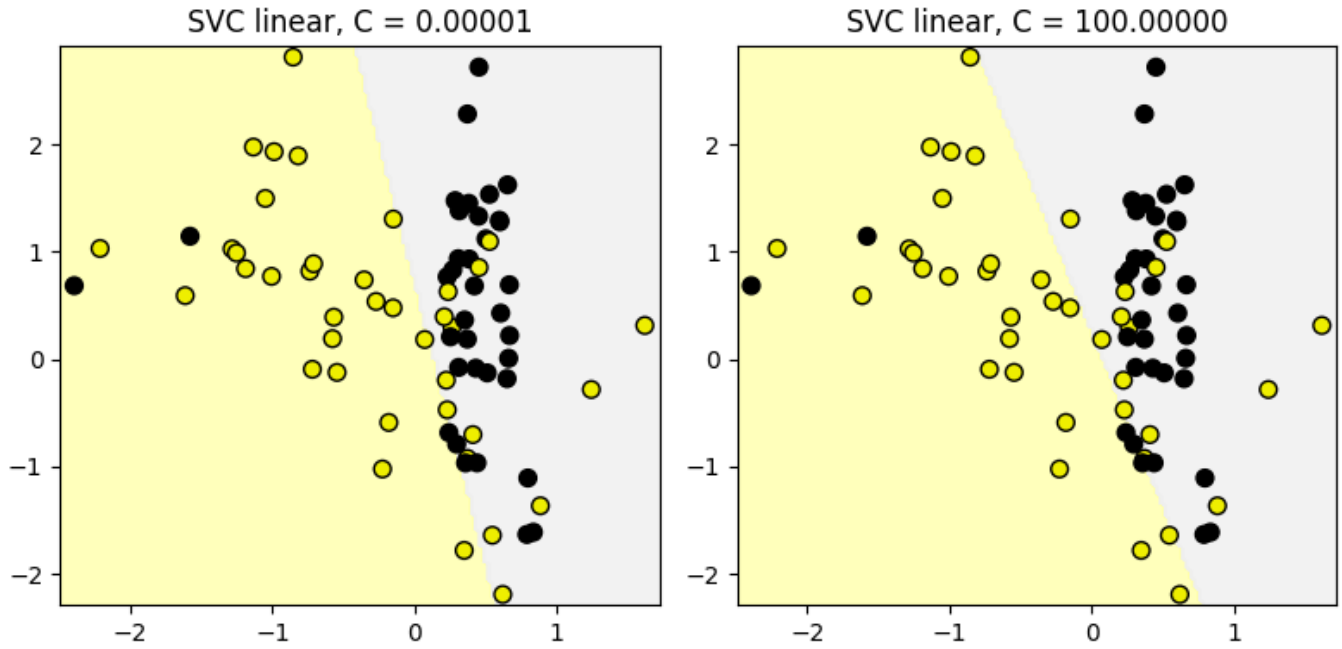



```
0.76
0.8
```

7 - SVM e o parâmetro C

In [11]:
```
from sklearn.svm import LinearSVC

X_train, X_test, y_train, y_test = train_test_split(X_C2, y_C2, random_state = 0)

fig, subaxes = plt.subplots(1, 2, figsize=(8, 4))

for this_C, subplot in zip([0.00001, 100], subaxes):
    clf = LinearSVC(C=this_C).fit(X_train, y_train)
```

```
    title = 'SVC linear, C = {:.5f}'.format(this_C)
    print(clf.score(X_train, y_train))
    print(clf.score(X_test, y_test))
    print('\n')
    plot_class_regions_for_classifier_subplot(clf, X_train, y_train,
                                              None, None, title, subplot)
plt.tight_layout()
```




```
0.7733333333333333
0.8


0.7466666666666667
0.8


/Users/marinaramalhetedesouza/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/sklea
rn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to converge, increase the numbe
r of iterations.
  warnings.warn(
```

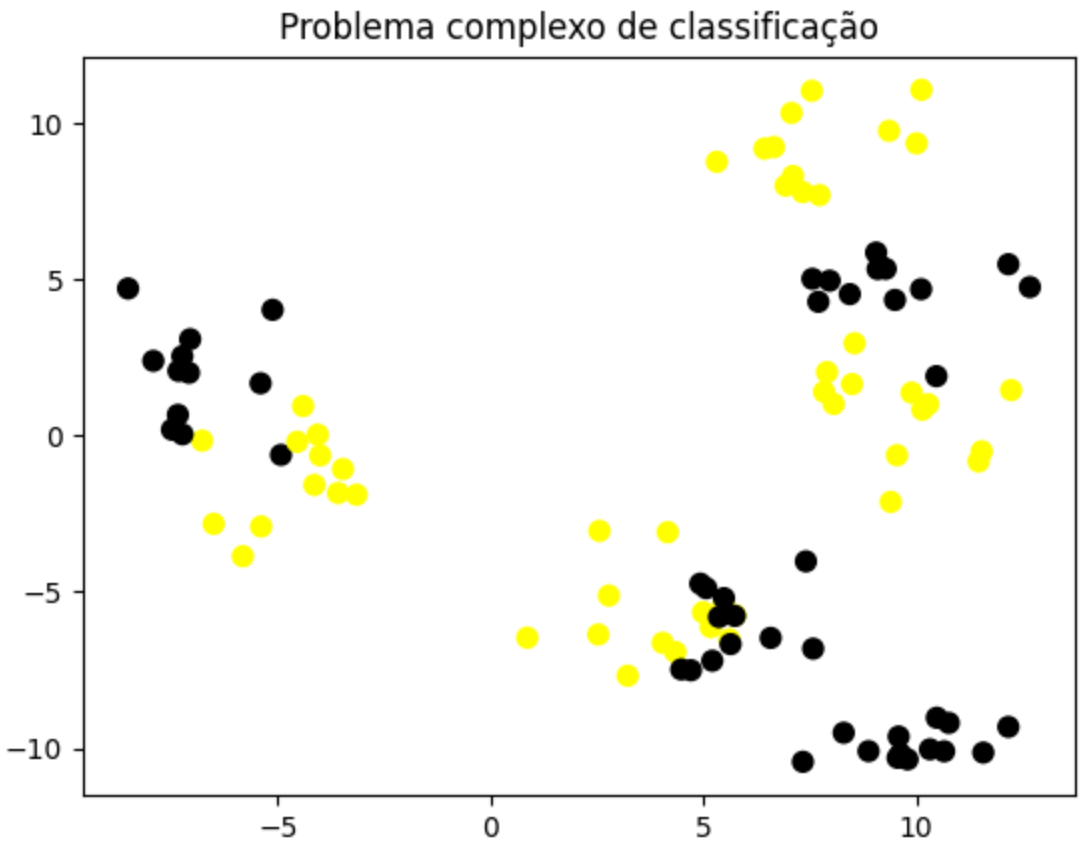
Máquinas de Vetores de Suportes - Kernelização

(Kernelized Support Vector Machines)

8 - Base de dados sintética com problema complexo de classificação

In [12]:

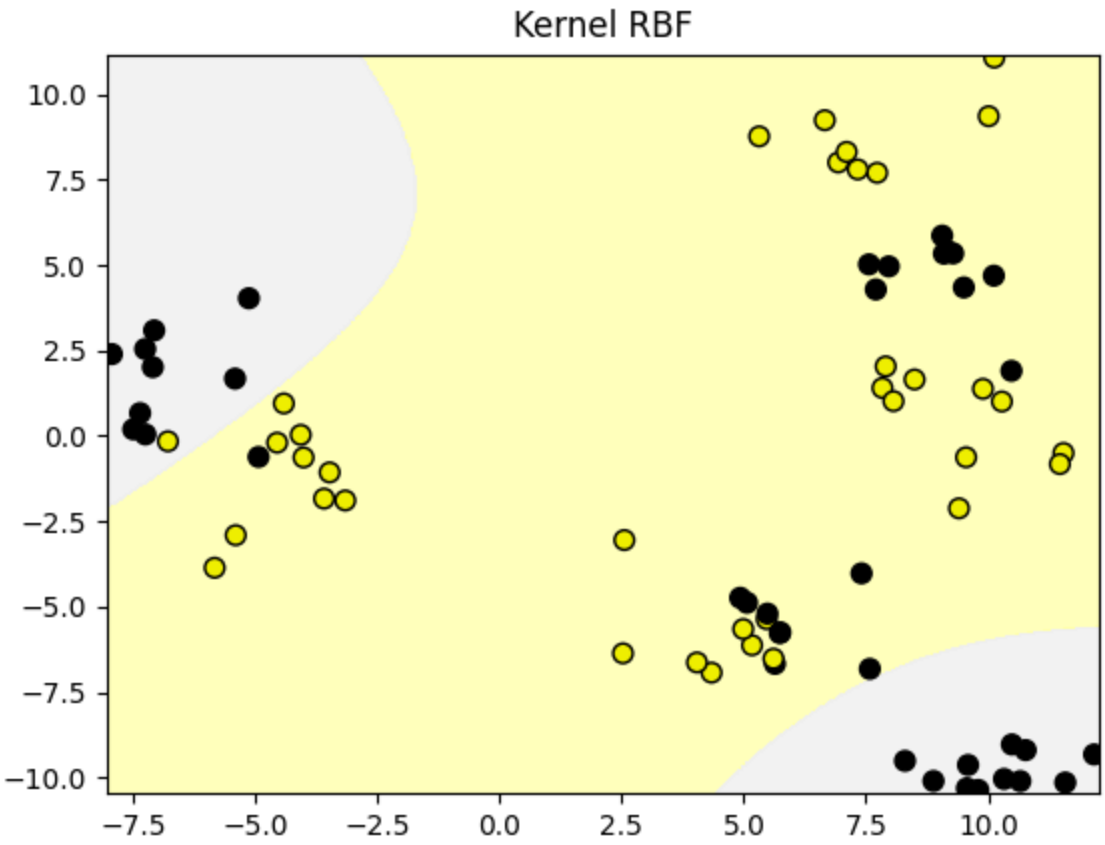
```
X_D2, y_D2 = make_blobs(n_samples = 100, n_features = 2, centers = 8,
                        cluster_std = 1.3, random_state = 4)
y_D2 = y_D2 % 2
plt.figure()
plt.title('Problema complexo de classificação')
plt.scatter(X_D2[:,0], X_D2[:,1], c=y_D2,
            marker= 'o', s=50, cmap=cmap_bold)
plt.show()
```

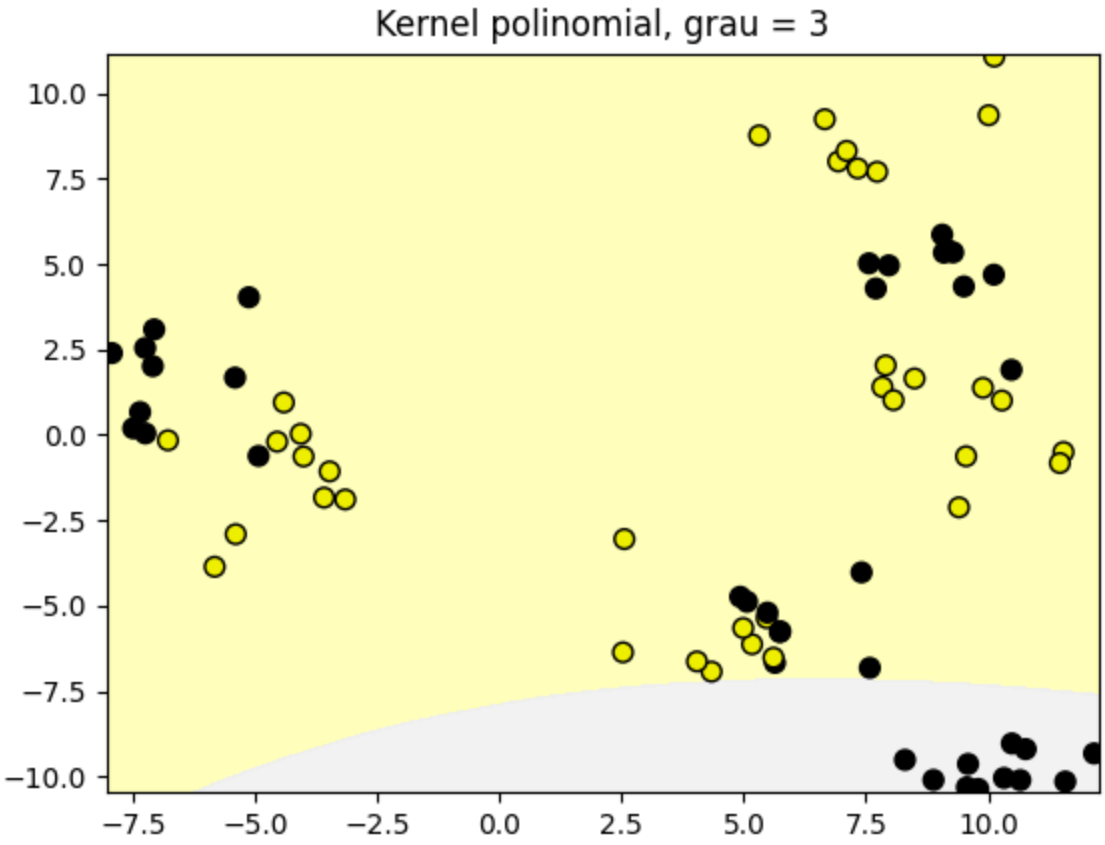


9 - Kernel RBF vs. Polinomial

In [13]:

```
from sklearn.svm import SVC

X_train, X_test, y_train, y_test = train_test_split(X_D2, y_D2, random_state = 0)

# Padrão é RBF
plot_class_regions_for_classifier(SVC().fit(X_train, y_train),
                                  X_train, y_train, None, None,
                                  'Kernel RBF')

# Kernel polinomial com grau = 3
plot_class_regions_for_classifier(SVC(kernel = 'poly', degree = 3)
                                  .fit(X_train, y_train), X_train,
                                  y_train, None, None,
                                  'Kernel polinomial, grau = 3')
```

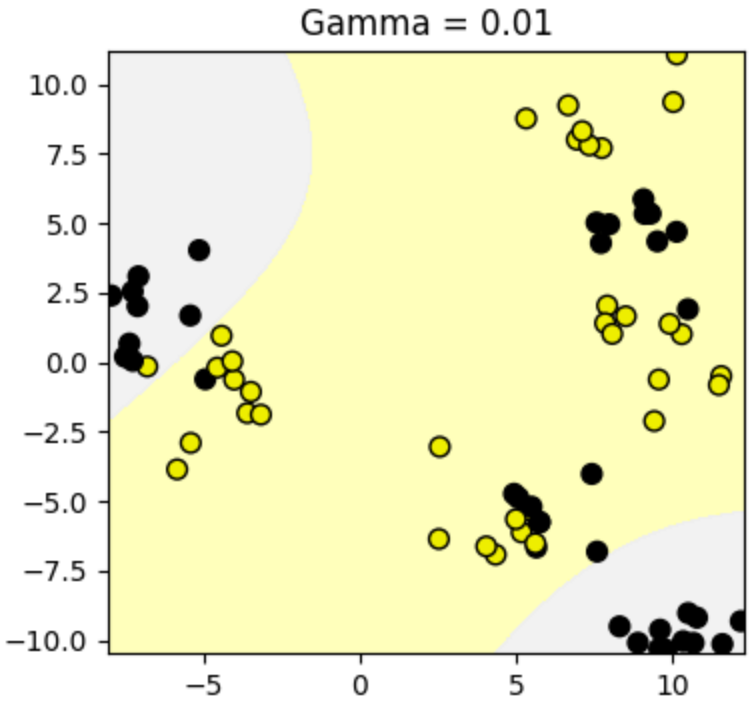


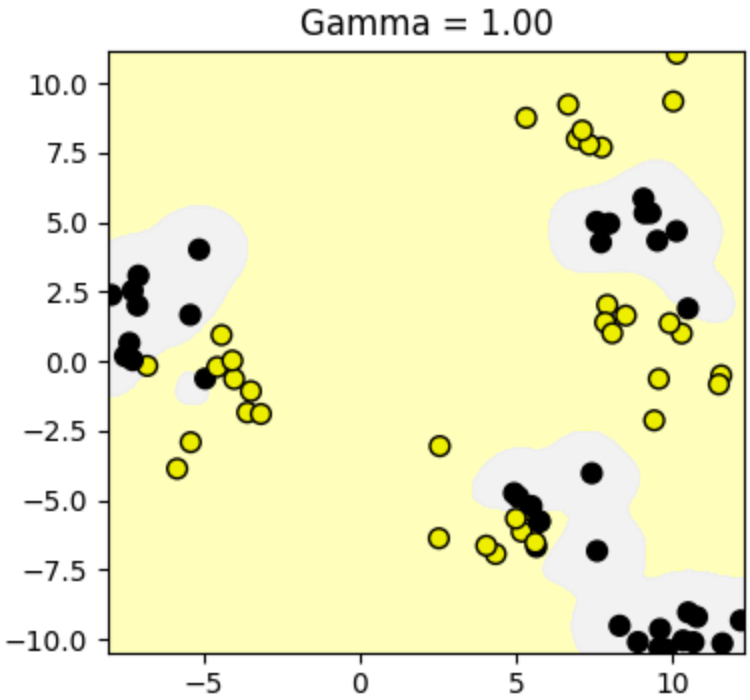


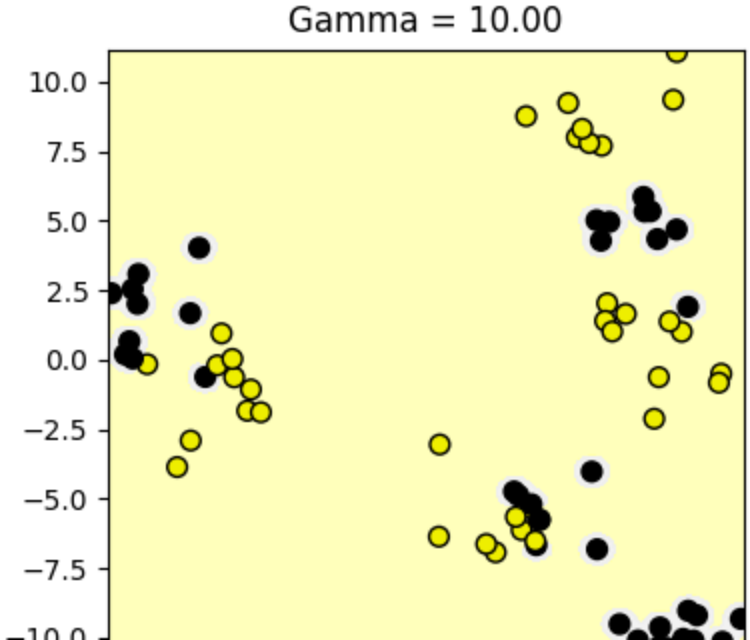
10 - Kernel RBF e o parâmetro Gamma

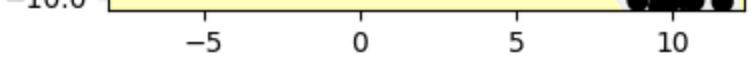
In [14]:

```
X_train, X_test, y_train, y_test = train_test_split(X_D2, y_D2, random_state = 0)

fig, subaxes = plt.subplots(3, 1, figsize=(4, 11))

for this_gamma, subplot in zip([0.01, 1.0, 10.0], subaxes):
    clf = SVC(kernel = 'rbf', gamma=this_gamma).fit(X_train, y_train)
    title = 'Gamma = {:.2f}'.format(this_gamma)
    plot_class_regions_for_classifier_subplot(clf, X_train, y_train,
                                              None, None, title, subplot)
    plt.tight_layout()
```


Gamma = 0.01
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−5
0
5
10


Gamma = 1.00
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−5
0
5
10

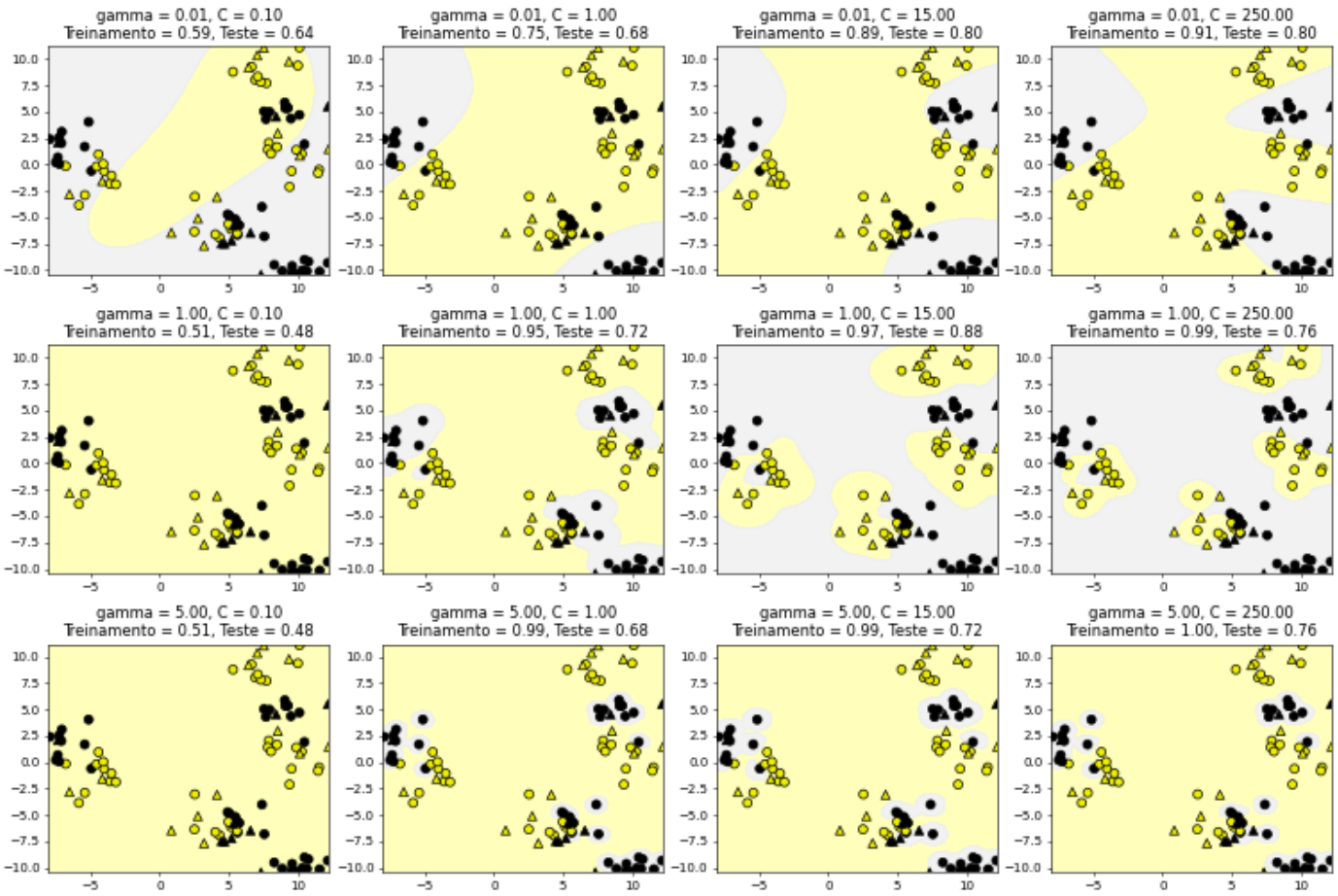

Gamma = 10.00
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5



11 - Kernel RBF + parâmetros C e Gamma

In [15]:

```
from sklearn.svm import SVC

X_train, X_test, y_train, y_test = train_test_split(X_D2, y_D2, random_state = 0)

fig, subaxes = plt.subplots(3, 4, figsize=(15, 10), dpi=50)

for this_gamma, this_axis in zip([0.01, 1, 5], subaxes):

    for this_C, subplot in zip([0.1, 1, 15, 250], this_axis):
        title = 'gamma = {:.2f}, C = {:.2f}'.format(this_gamma, this_C)
        clf = SVC(kernel = 'rbf', gamma = this_gamma,
                  C = this_C).fit(X_train, y_train)
        plot_class_regions_for_classifier_subplot(clf, X_train, y_train,
                                                  X_test, y_test, title,
                                                  subplot)
        plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=1.0)
```



In [ ]: