```
#-----------------------------------------------------
#-----------------------------------------------------
# Ejemplo de Uso de Support Vector Machines (SVM)
#-----------------------------------------------------

#-----------------------------------------------------
#0. Setup inicial
#
# Liberías necesarias para que funcione el algoritmo.
#-----------------------------------------------------
#=====================================================
# Setup de librerías
#-----------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from   sklearn.metrics import confusion_matrix, accuracy_score
import seaborn as sns
from   IPython.display import clear_output
#=====================================================


#-----------------------------------------------------
# Carga de base de datos
#-----------------------------------------------------
#-----------------------------------------------------
# En estos ejemplos usaremos las siguientes bases de datos de características para probar clasificadores:
#
# F2: Training data: 800 samples with 2 features. Testing data: 400 samples with 2 features. Classes: 0...1
#
# G2: Training data: 800 samples with 2 features Testing data: 200 samples with 2 features. Classes: 1...2
#
# G3: Training data: 1200 samples with 2 features. Testing data: 600 samples with 2 features. Classes: 1...3
#
# G4: Training data: 800 samples with 2 features .Testing data: 400 samples with 2 features. Classes: 1...4
#
# P2: Training data: 1052 samples with 2 features. Testing data: 262 samples with 2 features. Classes: 0...1
#-----------------------------------------------------
#=====================================================

!wget https://www.dropbox.com/. . ./G2.zip
!unzip -qq G2

clear_output()

print('Datos F2, F40, G3, G4 y P2 cargados.')
#=====================================================


#-----------------------------------------------------
# Funciones necesarias
#-----------------------------------------------------
#=====================================================
def load_features(prefix):
    Xtrain  = np.load(prefix+'_Xtrain.npy')            # training samples
    Xtest = np.load(prefix+'_Xtest.npy')             # testing samples
    ytrain  = np.ravel(np.load(prefix+'_dtrain.npy'))  # training labels
    ytest = np.ravel(np.load(prefix+'_dtest.npy'))   # testing labels
    print('Training data: '+str(Xtrain.shape[0]) +' samples with '+str(Xtrain.shape[1]) +' features')
    print(' Testing data: '+str(Xtest.shape[0])+' samples with '+str(Xtest.shape[1])+' features')
    print('      Classes: '+str(int(np.min(ytrain)))+'...'+str(int(np.max(ytrain))))
```

```
62      return Xtrain,ytrain,Xtest,ytest
63
64  def print_confusion(dt,ds,show_heatmap=0,Cnorm=1):
65      # dt - true, ds - predicted
66      C   = confusion_matrix(dt,ds)
67      print('Confusion Matrix:')
68      print(C)
69      acc = accuracy_score(dt,ds)
70      acc_st = "{:.2f}".format(acc*100)
71      print('Accuracy = '+str(acc_st))
72      if show_heatmap:
73        sns.heatmap(C/Cnorm, annot=True, cbar=None, cmap="Blues")
74        plt.title("Confusion Matrix"), plt.tight_layout()
75        plt.ylabel("True Class"), plt.xlabel("Predicted Class")
76        plt.show()
77
78  def plot_features(X,d,st,show=1):
79      dmin = int(np.min(d))
80      dmax = int(np.max(d))
81      #colors =
    np.array(["red","green","blue","yellow","pink","black","orange","purple","beige","brow
    n","gray","cyan","magenta"])
82      #colors = 'Greens'
83      for j in range(dmin,dmax+1):
84          plt.scatter(X[d==j,0],X[d==j,1],label=str(j),s=27)
85      plt.grid(True)
86      plt.legend()
87      plt.xlabel('$x_1$',fontsize=14)
88      plt.ylabel('$x_2$',fontsize=14)
89      plt.title('Feature Space - '+st,fontsize=14)
90      if show==1:
91          plt.show()
92
93  def plot_decision_lines(clf,X,show=0,decisionline=1):
94      # based on example of https://scikit-learn.org
95      h = 0.075
96      x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
97      y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
98      xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
99                           np.arange(y_min, y_max, h))
100     if decisionline == 1:
101         Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
102         Z = Z.reshape(xx.shape)
103         plt.contourf(xx, yy, Z, cmap=plt.cm.Reds, alpha=0.8)
104     plt.xlim(x_min, x_max)
105     plt.ylim(y_min, y_max)
106     if show==1:
107         plt.show()
108
109 def plot_decision_heatmap(clf,X,d,st,show=0):
110     # based on example of https://scikit-learn.org
111     h = 0.075
112     x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
113     y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
114     xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
115                          np.arange(y_min, y_max, h))
116     Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
117     Z = Z.reshape(xx.shape)
118     plt.pcolormesh(xx, yy, -Z, cmap=plt.cm.RdBu)
119     plt.xlim(x_min, x_max)
120     plt.ylim(y_min, y_max)
121     plt.xticks(())
122     plt.yticks(())
123     plt.axis('tight')
124     plot_features(X,d,st)
125     if show==1:
126         plt.show()
127
128
```

```
129
130
131 def show_clf_results(clf,X,d,Xt,dt,d0,ds,st,decisionline=1):
132     gs = gridspec.GridSpec(1, 2)
133     fig = plt.figure(figsize=(18, 6))
134     print('Training:')
135     acc = accuracy_score(d,d0)
136     accst = f'Acc = {acc:.4f}'
137     ax = plt.subplot(gs[0,0])
138     print_confusion(d,d0)                                  # confusion matrix in training
139     plot_decision_lines(clf,X,0,decisionline)               # decision lines
140     plot_features(X,d,st+' - Training: '+accst,0)          # feature space in training
141     ax = plt.subplot(gs[0,1])
142     print('Testing:')
143     acc = accuracy_score(ds,dt)
144     accst = f'Acc = {acc:.4f}'
145     print_confusion(dt,ds)                                 # confusion matrix in testing
146     plot_decision_lines(clf,X,0,decisionline)               # decision lines
147     plot_features(Xt,dt,st+' - Testing: '+accst,1)         # feature space in testing
148 #=====================================================
149
150
151 #-----------------------------------------------------
152 # Training/Testing Data
153 #-----------------------------------------------------
154 #=====================================================
155 data = 'G2' # <= puede ser 'G2','G3','G4','P2','F2' (ver explicación más arriba)
156 print('Cargando '+data+'...')
157
158 Xtrain,ytrain,Xtest,ytest   = load_features(data + '/' + data)   # load training and
    testing data
159
160 plot_features(Xtrain,ytrain,'Training')
161 plot_features(Xtest,ytest,'Testing')
162 #=====================================================
163
164
165 #-----------------------------------------------------
166 # Parámetros
167 #-----------------------------------------------------
168 # Explicación de los parámetros C y gamma del SVM:
    https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html
169 #-----------------------------------------------------
170 #-----------------------------------------------------
171 # SVM lineal
172 #-----------------------------------------------------
173 #=====================================================
174 from sklearn.svm import SVC
175
176 # Training
177 clf = SVC(kernel = "linear", gamma=0.2, C=0.1)
178
179 clf.fit(Xtrain, ytrain)
180
181 # Evaluation on training subset
182 y0  = clf.predict(Xtrain)
183
184 # Evaluation on testing subset
185 ypred  = clf.predict(Xtest)
186
187 # Display results
188 show_clf_results(clf,Xtrain,ytrain,Xtest,ytest,y0,ypred,'SVM-Lin',decisionline=1)
189 #=====================================================
190
191 #=====================================================
192 plot_decision_heatmap(clf,Xtrain,ytrain,'SVM-Lin (Training)')
193 #=====================================================
194
195 #-----------------------------------------------------
```

```
# SVM Polinomial
#-----------------------------------------------------
#=====================================================
from sklearn.svm import SVC

# Training
clf = SVC(kernel = "poly", gamma=0.2, degree = 3, C=0.1)

clf.fit(Xtrain, ytrain)

# Evaluation on training subset
y0  = clf.predict(Xtrain)

# Evaluation on testing subset
ypred  = clf.predict(Xtest)

# Display results
show_clf_results(clf,Xtrain,ytrain,Xtest,ytest,y0,ypred,'SVM-Poly',decisionline=1)
#=====================================================

#=====================================================
plot_decision_heatmap(clf,Xtrain,ytrain,'SVM-Poly (Training)')
#=====================================================

#-----------------------------------------------------
# SVM RBF
#-----------------------------------------------------
#=====================================================
from sklearn.svm import SVC

# Training
clf = SVC(kernel = "rbf", gamma=0.2,C=0.1)

clf.fit(Xtrain, ytrain)

# Evaluation on training subset
y0  = clf.predict(Xtrain)

# Evaluation on testing subset
ypred  = clf.predict(Xtest)

# Display results
show_clf_results(clf,Xtrain,ytrain,Xtest,ytest,y0,ypred,'SVM-RBF',decisionline=1)
#=====================================================

#=====================================================
plot_decision_heatmap(clf,Xtrain,ytrain,'SVM-RBF (Training)')
#=====================================================

#-----------------------------------------------------
# SVM Sigmoid
#-----------------------------------------------------
#=====================================================
from sklearn.svm import SVC

# Training
clf = SVC(kernel = "sigmoid", gamma=0.01, C=1.5)

clf.fit(Xtrain, ytrain)

# Evaluation on training subset
y0  = clf.predict(Xtrain)

# Evaluation on testing subset
ypred  = clf.predict(Xtest)

# Display results
show_clf_results(clf,Xtrain,ytrain,Xtest,ytest,y0,ypred,'SVM-Sigmoid',decisionline=1)
#=====================================================
```

```
265
266   #=====================================================
267   plot_decision_heatmap(clf,Xtrain,ytrain,'SVM-Sigmoid (Training)')
268   #=====================================================
269
270   #-----------------------------------------------------
271   # Matriz de Confusión y Eficiencia
272   #-----------------------------------------------------
273   #=====================================================
274   from sklearn.svm import SVC as SVM
275
276   # Training
277   clf = SVM(kernel = "rbf", gamma=0.2,C=0.1)
278   #clf = SVM(kernel = "linear", gamma=0.2, C=0.1)
279
280   clf.fit(Xtrain, ytrain)
281
282   # Evaluation on training subset
283   y0  = clf.predict(Xtrain)
284
285   # Evaluation on testing subset
286   ypred  = clf.predict(Xtest)
287
288   # Display results
289   C   = confusion_matrix(ytest,ypred)
290   print('Confusion Matrix:')
291   print(C)
292   acc = accuracy_score(ytest,ypred)
293   acc_st = "{:.2f}".format(acc*100)
294   print('Accuracy = '+str(acc_st))
295   #=====================================================
296
297   #=====================================================
298   #=====================================================
299   #=====================================================
```