```
"""
Example 6.1 Support Vector Machine Classification
@author: Austin R.J. Downey
"""

import IPython as IP
IP.get_ipython().run_line_magic('reset', '-sf')

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

cc = plt.rcParams['axes.prop_cycle'].by_key()['color']
plt.close('all')


# This code will build and train a support vector machine classifier with soft
# for the iris flower data set.

#%% Load your data

# We will use the Iris data set.  This dataset was created by biologist Ronald
# Fisher in his 1936 paper "The use of multiple measurements in taxonomic
# problems" as an example of linear discriminant analysis
iris = sk.datasets.load_iris()

# for simplicity, extract some of the data sets
X = iris['data'] # this contains the length of the petals and sepals
Y = iris['target'] # contains what type of flower it is
Y_names = iris['target_names'] # contains the name that aligns with the type of the flower
feature_names = iris['feature_names'] # the names of the features

# plot the Sepal data
plt.figure(figsize=(6.5,3))
plt.subplot(121)
plt.grid(True)
plt.scatter(X[Y==0,0],X[Y==0,1],marker='o',zorder=10)
plt.scatter(X[Y==1,0],X[Y==1,1],marker='s',zorder=10)
plt.scatter(X[Y==2,0],X[Y==2,1],marker='d',zorder=10)
plt.xlabel(feature_names[0])
plt.ylabel(feature_names[1])

plt.subplot(122)
plt.grid(True)
plt.scatter(X[Y==0,2],X[Y==0,3],marker='o',label=Y_names[0],zorder=10)
plt.scatter(X[Y==1,2],X[Y==1,3],marker='s',label=Y_names[1],zorder=10)
plt.scatter(X[Y==2,2],X[Y==2,3],marker='d',label=Y_names[2],zorder=10)
plt.xlabel(feature_names[2])
plt.ylabel(feature_names[3])
plt.legend(framealpha=1)
plt.tight_layout()

#%% Extract just the petal space of the code

X_petal = X[50:, (2, 3)]  # petal length, petal width
y_petal = Y[50:] == 2

#%% Build and train the SVM classifier

# build handles to regularize the model data and a Linear Support Vector Classification.
scaler = sk.preprocessing.StandardScaler()
svm_clf = sk.svm.LinearSVC(C=1000000,max_iter=10000)

```

```python
# build the model pipeline of regularization and a Linear Support Vector Classification.
scaled_svm_clf = sk.pipeline.Pipeline([
        ("scaler", scaler),
        ("linear_svc", svm_clf),
    ])

# train the data
scaled_svm_clf.fit(X_petal, y_petal)


#%% Build and plot the decision boundary along with the curbs

# Convert to unscaled parameters as the SVM is solved in a scaled space.
w = svm_clf.coef_[0] / scaler.scale_
b = svm_clf.decision_function([-scaler.mean_ / scaler.scale_])

# At the decision boundary, w0*x0 + w1*x1 + b = 0
# => x1 = -w0/w1 * x0 - b/w1
x0 = np.linspace(4, 5.9, 200)
decision_boundary = -w[0]/w[1] * x0 - b/w[1]

margin = 1/w[1]
curbs_up = decision_boundary + margin
curbs_down = decision_boundary - margin

#%% Plot the data and the classifier

plt.figure()
plt.grid(True)
plt.scatter(X[Y==1,2],X[Y==1,3],marker='s',label=Y_names[1],zorder=10)
plt.scatter(X[Y==2,2],X[Y==2,3],marker='d',label=Y_names[2],zorder=10)
plt.xlabel(feature_names[2])
plt.ylabel(feature_names[3])
plt.legend(framealpha=1)
plt.tight_layout()

# plot the decision boundy and margins
plt.plot(x0, decision_boundary, "k-", linewidth=2)
plt.plot(x0, curbs_up, "k--", linewidth=2)
plt.plot(x0, curbs_down, "k--", linewidth=2)


#%% Find the misclassified instancances and add a circle to mark them

# Find support vectors (LinearSVC does not do this automatically) and add them
# to the SVM handle
t = y_petal * 2 - 1 # convert 0 and 1 to -1 and 1
support_vectors_idx = (t * (X_petal.dot(w) + b) < 1) # find the locations
# of the miss classifed data points that fall withing the vectors
svs = X_petal[support_vectors_idx]

plt.scatter(svs[:, 0], svs[:, 1], s=180,marker='o', facecolors='none',edgecolors='k')

#%% compute the confusion matirx and F1 score

y_predicted = scaled_svm_clf.predict(X_petal)
confusion_matrix = sk.metrics.confusion_matrix(y_predicted, y_petal)
f1_score = sk.metrics.f1_score(y_predicted, y_petal)

print(f1_score)
```