```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Example 4.3 2D Decision boundary for the Iris dataset

Developed for Machine Learning for Mechanical Engineers at the University of
South Carolina

@author: austin_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


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


#%% 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()


#%% plot the Linear decision boundary in 2D "Petal" space

# build the training and target set.
X_train = X[:, (2, 3)]  # petal length, petal width
y_train = Y == 2

```

```
# build the Logistic Regression model
log_reg = sk.linear_model.LogisticRegression(C=10**10)
# Note: The hyper-parameter controlling the regularization strength of a Scikit-Learn
# LogisticRegression model is not alpha (as in other linear models), but its
# inverse: C. The higher the value of C, the less the model is regularized.

# train the Logistic Regression model
log_reg.fit(X_train, y_train)

# build the x values for the predictions over the entire "petal space"
x_grid, y_grid = np.meshgrid(
        np.linspace(2.8, 7, 500),
        np.linspace(0.3, 3, 200),
    )
X_new = np.vstack((x_grid.reshape(-1), y_grid.reshape(-1))).T # build a vector format of the mesh grid

# predict on the vectorized format
y_predict = log_reg.predict(X_new)
y_proba = log_reg.predict_proba(X_new)

# convert back to meshgrid shape for plotting
zz_predict = y_predict.reshape(x_grid.shape)
zz_proba = y_proba[:, 1].reshape(x_grid.shape)

# plot the 2D "petal space"
plt.figure(figsize=(6.5,3))
plt.grid(True)
plt.scatter(X[Y==1,2],X[Y==1,3],marker='s',color=cc[1],label=Y_names[1],zorder=10)
plt.scatter(X[Y==2,2],X[Y==2,3],marker='d',color=cc[2],label=Y_names[2],zorder=10)
plt.contourf(x_grid, y_grid, zz_predict, cmap='Pastel2')
contour = plt.contour(x_grid, y_grid, zz_proba, [0.100,0.5,0.900],cmap=plt.cm.brg)
plt.clabel(contour, inline=1) # add the labels to the plot
plt.xlabel(feature_names[2])
plt.ylabel(feature_names[3])
plt.legend()
plt.tight_layout()
```