Code:

import pandas as pd

from sklearn.linear\_model import LogisticRegression

import matplotlib.pyplot as plt

import numpy as np

from IPython.display import display

from matplotlib.colors import ListedColormap

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

path= r"C:\Users\jagathchandra\Downloads\seeds\_dataset.txt"

features = ['Area',

'Perimeter',

'Compactness',

'Length ',

'Width',

'Asymmetry coefficient',

' groove.']

df = pd.read\_csv(path,delimiter='[\t]+',header=None,names=features + ['target'])

display(df)

# Creating Feature array and target array

X = df.iloc[1:, [1,2]].values

X=X.astype('float64')

#print(X)

y = df.loc[1:,'target']

y=y.astype('int64')

#print (y)

print('Class labels:', np.unique(y))

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=1, stratify=y)

print('Labels counts in y:', np.bincount(y)[1:])

print('Labels counts in y\_train:', np.bincount(y\_train)[1:])

print('Labels counts in y\_test:', np.bincount(y\_test)[1:])

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

sc.fit(X\_train)

X\_train\_std = sc.transform(X\_train)

X\_test\_std = sc.transform(X\_test)

# Plotting decision Boundary

def plot\_decision\_regions(X, y, classifier, test\_idx=None, resolution=0.02):

# X = X.to\_numpy()

# y = y.to\_numpy()

# setup marker generator and color map

markers = ('o', 's', '^', 'v', '<')

colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')

cmap = ListedColormap(colors[:len(np.unique(y))])

# plot the decision surface

x1\_min, x1\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

x2\_min, x2\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx1, xx2 = np.meshgrid(np.arange(x1\_min, x1\_max, resolution),

np.arange(x2\_min, x2\_max, resolution))

lab = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)

lab = lab.reshape(xx1.shape)

plt.contourf(xx1, xx2, lab, alpha=1.0, cmap=cmap)

plt.xlim(xx1.min(), xx1.max())

plt.ylim(xx2.min(), xx2.max())

# plot class examples

for idx, cl in enumerate(np.unique(y)):

plt.scatter(x=X[y == cl, 0],

y=X[y == cl, 1],

alpha=1.0,

c=colors[idx],

marker=markers[idx],

label=f'Class {cl}',

edgecolor='black')

# highlight test examples

if test\_idx:

# plot all examples

X\_test, y\_test = X[test\_idx, :], y[test\_idx]

plt.scatter(X\_test[:, 0],

X\_test[:, 1],

c='none',

edgecolor='black',

alpha=1.0,

linewidth=1,

marker='o',

s=100,

label='Test set')

# Building Logistic Regression for Binary Classifier

logistic\_reg = LogisticRegression()

logistic\_reg.fit(X\_train,y\_train)

y\_pred= logistic\_reg.predict(X\_test)

print('predicted:',y\_pred)

print('true class:', np.array(y\_test))

plot\_decision\_regions(X, y, classifier=logistic\_reg)

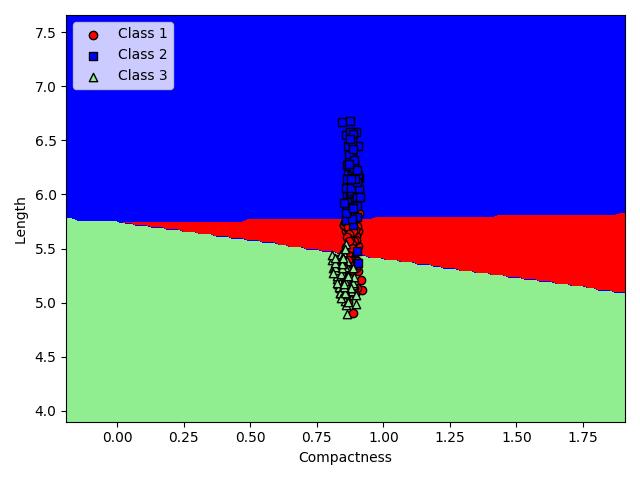
plt.xlabel(features[1])

plt.ylabel(features[2])

plt.legend(loc='upper left')

plt.tight\_layout()

plt.show()

Output Images:

