AmartyaDuttaSectionB_HW2

November 4, 2022

1 CS-5824 / Advanced Machine Learning

2 Assignment 2 Section B [40 Points]

In this assignment, **you need to complete two sections** which are based on:

1. Decision Trees (20 points)
2. Support Vector Machines (20 points)

2.1 Submission guidelines

1. Click the Save button at the top of the notebook.
2. Please make sure to enter your Virginia Tech PID below.
3. Select Edit -> Clear All Output. This will clear all the outputs from all cells (but will keep the content of all cells).
4. Select Runtime -> Restart and Run All. This will run all the cells in order.
5. Once you've rerun everything, select File -> Print -> Save as PDF.
6. Look at the PDF file and make sure all your solutions are there and correctly displayed.
7. Upload **both** the PDF file (saved in step 5) and this notebook.
8. Please **DO NOT** upload any data.

2.1.1 Your VT PID: amartya

3 Section 0. Environment Set Up

Mount your Google Drive in Google Colab:

[1]:
```
from google.colab import drive

drive.mount("/content/gdrive/")
```

```
Mounted at /content/gdrive/
```

Upload all files in the zip to a directory in your Google Drive, then append it to your Python path using sys (please modify `customized_path_to_your_homework` to be the path to your directory):

[2]:
```
import sys
from pathlib import Path

prefix = "/content/gdrive/My Drive/"
customized_path_to_your_homework = "CS 5824 ML/HW2/"
sys_path = prefix + customized_path_to_your_homework
sys.path.append(sys_path)
data_path = Path(sys_path) / "Data"
```

Run some setup code for this notebook. For all randomization done in this assignment, please use the `seed` below as the random state.

[3]:
```
from __future__ import print_function
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import loadmat
from sklearn.exceptions import ConvergenceWarning
import warnings

# We ignore the convergence warnings in this homework, as some of the exercise␣
 ↪will
# always trigger this warning.
warnings.filterwarnings("ignore", category=ConvergenceWarning)

# This is a bit of magic to make matplotlib figures appear inline in the␣
 ↪notebook rather than in a new window.
%matplotlib inline
plt.rcParams["figure.figsize"] = (10.0, 8.0) # set default size of plots
plt.rcParams["image.interpolation"] = "nearest"
plt.rcParams["image.cmap"] = "gray"

# Some more magic so that the notebook will reload external python modules;
# see http://stackoverflow.com/questions/1907993/
 ↪autoreload-of-modules-in-ipython
%load_ext autoreload
%autoreload 2

# Seed for all randomization
seed = 5824
```

4 Section 1. Decision Trees [15 points]

For this problem, we will use a decision tree classifier on a toy dataset provided by SciKit-Learn. We will experiment with the Wine dataset. The data is the results of a chemical analysis of wines grown in the same region in Italy by three different cultivators. There are thirteen different measurements taken for different constituents found in the three types of wine. The features include:

Alcohol

Malic acid

Ash

Alcalinity of ash

Magnesium

Total phenols

Flavanoids

Nonflavanoid phenols

Proanthocyanins

Color intensity

Hue

OD280/OD315 of diluted wines

Proline

The 3 classes are `class_0`, `class_1`, and `class_2`.

4.1 1.1. Data Preparation

First, we need to load the dataset from SciKit-Learn. The `load_wine()` function returns a `sklearn.utils.Bunch` object containing all information about the dataset, such as feature and target names, as well as the full description of the data in its `DESCR` property.

[4]:
```
# Load dataset
from sklearn.datasets import load_wine

wine_dataset = load_wine()
feature_names = wine_dataset.feature_names
target_names = wine_dataset.target_names
wine_X, wine_y = wine_dataset.data, wine_dataset.target
print(wine_dataset.DESCR)
```

```
.. _wine_dataset:

Wine recognition dataset
------------------------

**Data Set Characteristics:**

    :Number of Instances: 178 (50 in each of three classes)
    :Number of Attributes: 13 numeric, predictive attributes and the class
    :Attribute Information:
                - Alcohol
```

- Malic acid
- Ash
- Alcalinity of ash
- Magnesium
- Total phenols
- Flavanoids
- Nonflavanoid phenols
- Proanthocyanins
- Color intensity
- Hue
- OD280/OD315 of diluted wines
- Proline

- class:
    - class_0
    - class_1
    - class_2

:Summary Statistics:

	Min	Max	Mean	SD
Alcohol:	11.0	14.8	13.0	0.8
Malic Acid:	0.74	5.80	2.34	1.12
Ash:	1.36	3.23	2.36	0.27
Alcalinity of Ash:	10.6	30.0	19.5	3.3
Magnesium:	70.0	162.0	99.7	14.3
Total Phenols:	0.98	3.88	2.29	0.63
Flavanoids:	0.34	5.08	2.03	1.00
Nonflavanoid Phenols:	0.13	0.66	0.36	0.12
Proanthocyanins:	0.41	3.58	1.59	0.57
Colour Intensity:	1.3	13.0	5.1	2.3
Hue:	0.48	1.71	0.96	0.23
OD280/OD315 of diluted wines:	1.27	4.00	2.61	0.71
Proline:	278	1680	746	315

:Missing Attribute Values: None
:Class Distribution: class_0 (59), class_1 (71), class_2 (48)
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988

This is a copy of UCI ML Wine recognition datasets.
https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data

The data is the results of a chemical analysis of wines grown in the same

```
region in Italy by three different cultivators. There are thirteen different
measurements taken for different constituents found in the three types of
wine.

Original Owners:

Forina, M. et al, PARVUS -
An Extendible Package for Data Exploration, Classification and Correlation.
Institute of Pharmaceutical and Food Analysis and Technologies,
Via Brigata Salerno, 16147 Genoa, Italy.

Citation:

Lichman, M. (2013). UCI Machine Learning Repository
[https://archive.ics.uci.edu/ml]. Irvine, CA: University of California,
School of Information and Computer Science.

.. topic:: References

  (1) S. Aeberhard, D. Coomans and O. de Vel,
  Comparison of Classifiers in High Dimensional Settings,
  Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of
  Mathematics and Statistics, James Cook University of North Queensland.
  (Also submitted to Technometrics).

  The data was used with many others for comparing various
  classifiers. The classes are separable, though only RDA
  has achieved 100% correct classification.
  (RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data))
  (All results using the leave-one-out technique)

  (2) S. Aeberhard, D. Coomans and O. de Vel,
  "THE CLASSIFICATION PERFORMANCE OF RDA"
  Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of
  Mathematics and Statistics, James Cook University of North Queensland.
  (Also submitted to Journal of Chemometrics).
```

Normalize the data, then split it into train (80%) and test (20%) sets using the provided `seed` and with stratification:

[5]:
```
from pandas.core import window
#Normalizing data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(wine_X)
wine_X_normalised = scaler.transform(wine_X)
wine_X_normalised
```

[5]:
```
array([[ 1.51861254, -0.5622498 ,  0.23205254, …,  0.36217728,
         1.84791957,  1.01300893],
       [ 0.24628963, -0.49941338, -0.82799632, …,  0.40605066,
         1.1134493 ,  0.96524152],
       [ 0.19687903,  0.02123125,  1.10933436, …,  0.31830389,
         0.78858745,  1.39514818],
       …,
       [ 0.33275817,  1.74474449, -0.38935541, …, -1.61212515,
        -1.48544548,  0.28057537],
       [ 0.20923168,  0.22769377,  0.01273209, …, -1.56825176,
        -1.40069891,  0.29649784],
       [ 1.39508604,  1.58316512,  1.36520822, …, -1.52437837,
        -1.42894777, -0.59516041]])
```

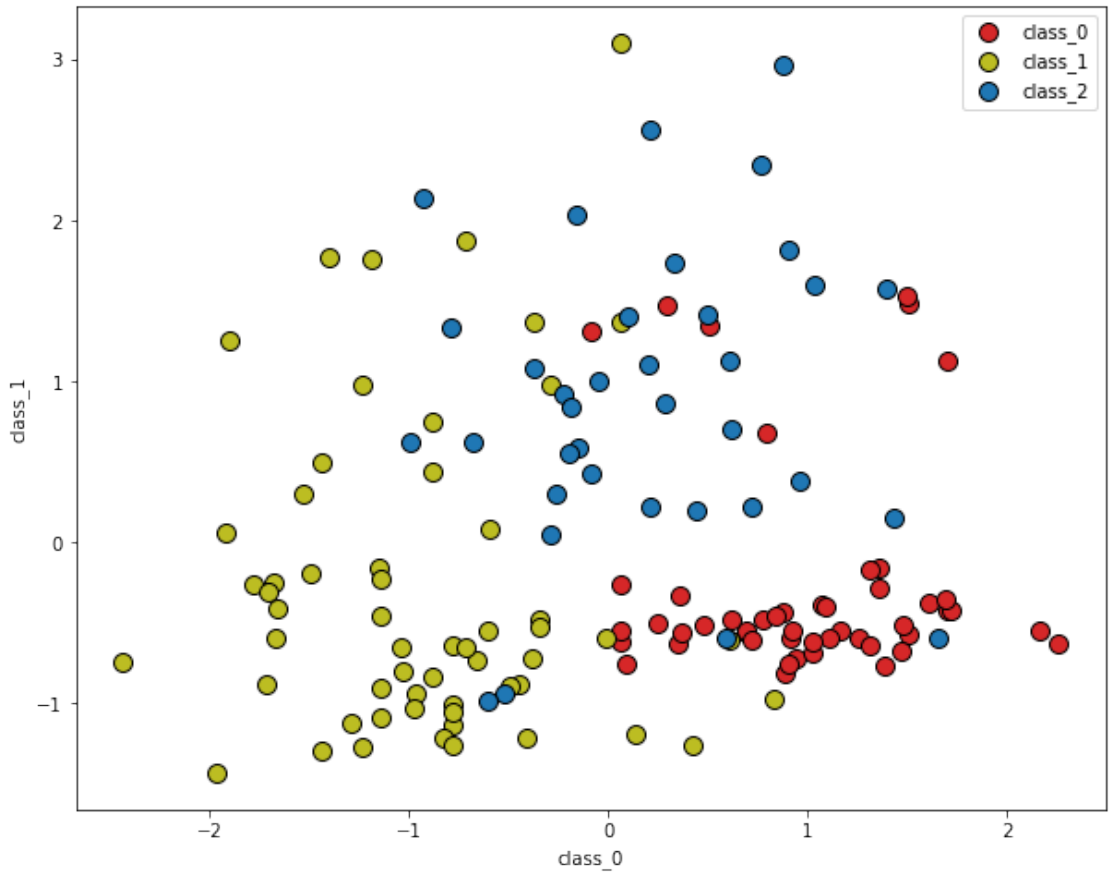
[6]:
```
# TODO: Preprocess data
from sklearn.model_selection import train_test_split
wine_X_train, wine_X_test, wine_y_train, wine_y_test = train_test_split(␣
 ↪wine_X_normalised, wine_y, test_size=0.2, random_state=seed)
```

[7]:
```
wine_X_train.shape, wine_X_test.shape
```

[7]:
```
((142, 13), (36, 13))
```

For ease of visualization, we will be using only the first 2 features (”Alcohol” and ”Malic acid”). Visualize the train set below:

[8]:
```
colors = ["tab:red", "tab:olive", "tab:blue"]
feature1, feature2 = 0, 1 # Chosen features
for label, color in zip(range(len(target_names)), colors):
    idx = np.where(wine_y_train == label)
    plt.scatter(
        wine_X_train[idx, feature1],
        wine_X_train[idx, feature2],
        s=100,
        color=color,
        edgecolor="black",
        label=target_names[label]
    )
plt.xlabel(target_names[0])
plt.ylabel(target_names[1])
plt.legend()
plt.show()
```



4.2 1.2. Training a Decision Tree (5 points)

Using `sklearn.tree.DecisionTreeClassifier`, train a decision tree classifier with entropy as the purity criterion, the provided `seed` as the random state, and a maximum depth of 3.

[9]:
```
# TODO: Train your decision tree
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion = 'entropy',random_state=seed,␣
↪max_depth=3)
```

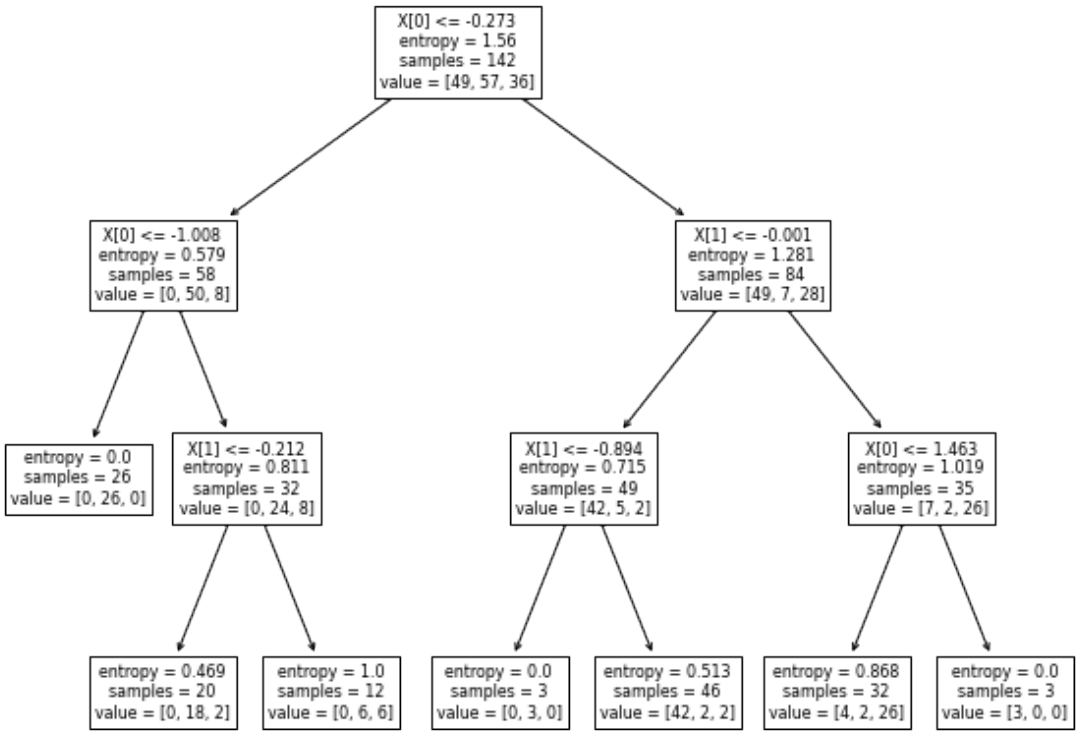
Plot your decision tree as a graph:

[10]:
```
clf.fit(wine_X_train[:,:2],wine_y_train)
```

[10]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=5824)
```

[11]:
```
from sklearn.tree import plot_tree

plot_tree(clf)
```

[11]: [Text(0.4230769230769231, 0.875, 'X[0] <= -0.273\nentropy = 1.56\nsamples = 142\nvalue = [49, 57, 36]'),
 Text(0.15384615384615385, 0.625, 'X[0] <= -1.008\nentropy = 0.579\nsamples = 58\nvalue = [0, 50, 8]'),
 Text(0.07692307692307693, 0.375, 'entropy = 0.0\nsamples = 26\nvalue = [0, 26, 0]'),
 Text(0.23076923076923078, 0.375, 'X[1] <= -0.212\nentropy = 0.811\nsamples = 32\nvalue = [0, 24, 8]'),
 Text(0.15384615384615385, 0.125, 'entropy = 0.469\nsamples = 20\nvalue = [0, 18, 2]'),
 Text(0.3076923076923077, 0.125, 'entropy = 1.0\nsamples = 12\nvalue = [0, 6, 6]'),
 Text(0.6923076923076923, 0.625, 'X[1] <= -0.001\nentropy = 1.281\nsamples = 84\nvalue = [49, 7, 28]'),
 Text(0.5384615384615384, 0.375, 'X[1] <= -0.894\nentropy = 0.715\nsamples = 49\nvalue = [42, 5, 2]'),
 Text(0.46153846153846156, 0.125, 'entropy = 0.0\nsamples = 3\nvalue = [0, 3, 0]'),
 Text(0.6153846153846154, 0.125, 'entropy = 0.513\nsamples = 46\nvalue = [42, 2, 2]'),
 Text(0.8461538461538461, 0.375, 'X[0] <= 1.463\nentropy = 1.019\nsamples = 35\nvalue = [7, 2, 26]'),
 Text(0.7692307692307693, 0.125, 'entropy = 0.868\nsamples = 32\nvalue = [4, 2, 26]'),
 Text(0.9230769230769231, 0.125, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0, 0]')]

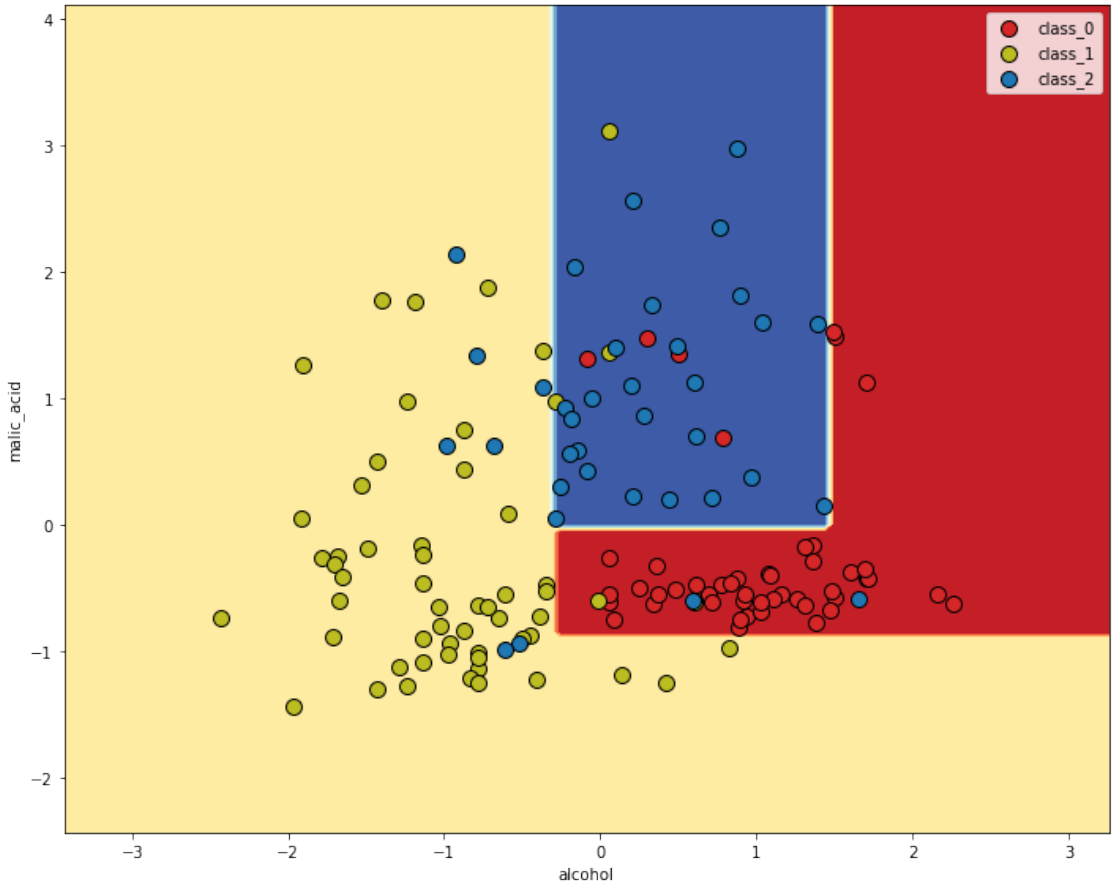


We can also plot the decision boundaries using the `sklearn.inspection.DecisionBoundaryDisplay` class—in particular, its `from_estimator()` method—which is new in SciKit-Learn 1.1.2. Due to package availability reasons, we include it in the attached `utils.py` file.

[12]:
```
from utils import DecisionBoundaryDisplay

def plot_decision_boundary(clf, X, y, chosen_features, feature_names,␣
 ↪target_names):
    feature1, feature2 = chosen_features
    fig, ax = plt.subplots()
    DecisionBoundaryDisplay.from_estimator(
        clf,
        np.column_stack((X[:, feature1], X[:, feature2])),
        cmap=plt.cm.RdYlBu,
        response_method="predict",
        ax=ax,
    )

    for label, color in zip(range(len(target_names)), colors):
        idx = np.where(y == label)
```

```
        plt.scatter(
            X[idx, feature1],
            X[idx, feature2],
            s=100,
            color=color,
            edgecolor="black",
            label=target_names[label]
        )

    plt.xlabel(feature_names[feature1])
    plt.ylabel(feature_names[feature2])
    plt.legend()
    plt.tight_layout()
```

[13]:
```
plot_decision_boundary(
    clf,
    wine_X_train,
    wine_y_train,
    (feature1, feature2),
    feature_names,
    target_names
)
```



4.3 1.3. Predicting (10 points)

There are two different functions for prediction within `DecisionTreeClassifier`.

(1) What are they? Invoke them on the test set in the cells below and look at the outputs. How are they different? How are they related? (2 points)

Your answer: The two different functions to predict using DecisionTreeClassifier are predict() and predict_proba(). The first one predicts the class for each sample while the second one predicts the class probabilities for each input. For each sample, the class with the highest probability as predicted by predict_proba() is the one predicted by predict() method.

[14]:
```
# TODO: Predict function 1
clf.predict(wine_X_test[:,:2])
```

[14]:
```
array([1, 1, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 0, 1, 2,
       1, 1, 2, 1, 1, 1, 2, 1, 1, 0, 0, 1, 2, 1])
```

[15]:
```
# TODO: Predict function 2
clf.predict_proba(wine_X_test[:,:2])
```

[15]:
```
array([[0.        , 1.        , 0.        ],
       [0.        , 0.9       , 0.1       ],
       [0.91304348, 0.04347826, 0.04347826],
       [0.91304348, 0.04347826, 0.04347826],
       [0.91304348, 0.04347826, 0.04347826],
       [0.91304348, 0.04347826, 0.04347826],
       [0.91304348, 0.04347826, 0.04347826],
       [0.91304348, 0.04347826, 0.04347826],
       [0.        , 0.5       , 0.5       ],
       [0.125     , 0.0625    , 0.8125    ],
       [0.        , 1.        , 0.        ],
       [0.91304348, 0.04347826, 0.04347826],
       [0.125     , 0.0625    , 0.8125    ],
       [0.125     , 0.0625    , 0.8125    ],
       [0.        , 0.9       , 0.1       ],
       [0.91304348, 0.04347826, 0.04347826],
       [0.125     , 0.0625    , 0.8125    ],
       [0.91304348, 0.04347826, 0.04347826],
       [0.        , 0.5       , 0.5       ],
       [0.91304348, 0.04347826, 0.04347826],
       [0.        , 1.        , 0.        ],
       [0.125     , 0.0625    , 0.8125    ],
       [0.        , 0.9       , 0.1       ],
       [0.        , 0.5       , 0.5       ],
       [0.125     , 0.0625    , 0.8125    ],
       [0.        , 1.        , 0.        ],
       [0.        , 0.9       , 0.1       ],
       [0.        , 1.        , 0.        ],
       [0.125     , 0.0625    , 0.8125    ],
       [0.        , 0.5       , 0.5       ],
       [0.        , 0.9       , 0.1       ],
       [0.91304348, 0.04347826, 0.04347826],
       [0.91304348, 0.04347826, 0.04347826],
       [0.        , 0.5       , 0.5       ],
       [0.125     , 0.0625    , 0.8125    ],
       [0.        , 1.        , 0.        ]])
```

(2) Compute the accuracy, precision, and F1-score to assess your decision tree's performance below. How is the performance? (3 points)

Your answer:

[16]:
```
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
```

[17]:
```
from os import access
# TODO: Evaluate decision tree's performance
accuracy = clf.score(wine_X_test[:,:2],wine_y_test)
precision = precision_score(wine_y_test,clf.predict(wine_X_test[:,:2]),␣
↪average='macro')
f1 = f1_score(wine_y_test,clf.predict(wine_X_test[:,:2]), average='macro')

print("Accuracy: ",accuracy)
print("Precision: ",precision)
print("F1 score: ",f1)
```

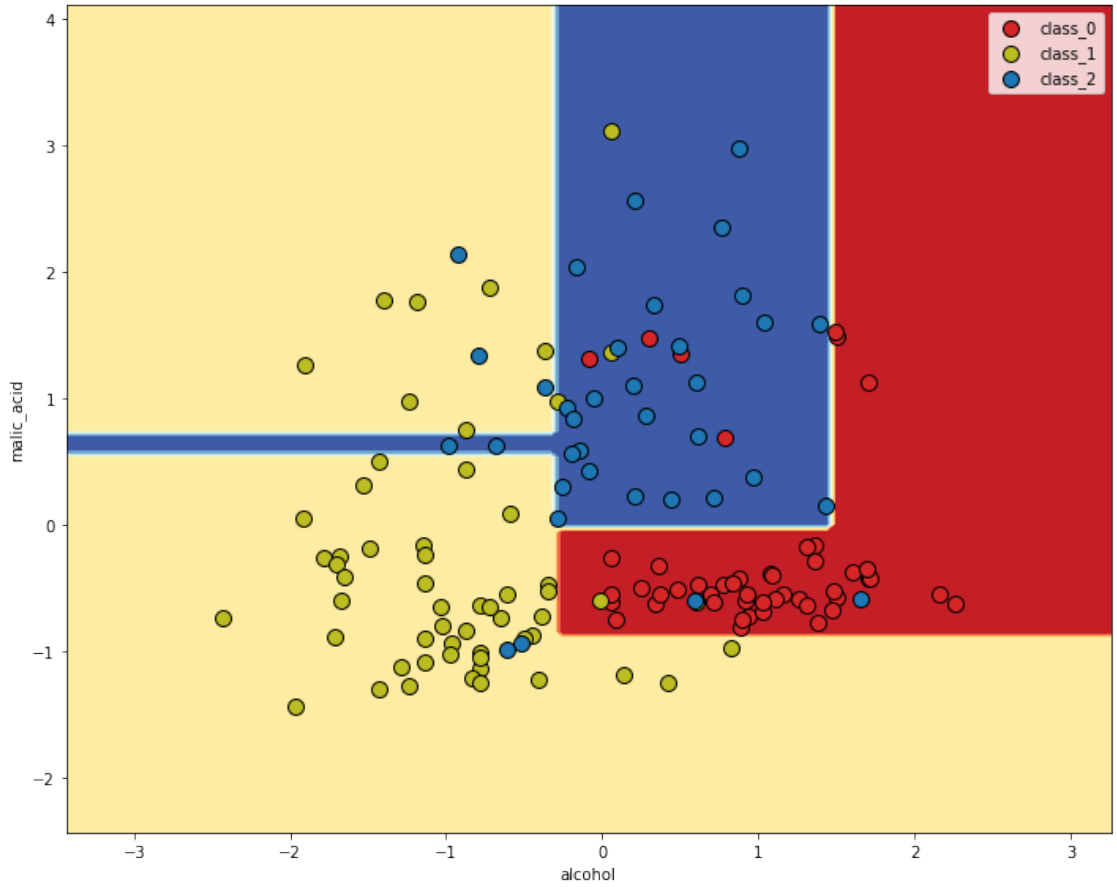
```
Accuracy:  0.75
Precision:  0.7430555555555555
F1 score:  0.7313131313131312
```

(3) Experiment with different parameters (*e.g.*, depth, selection criterion) and observe the change in decision boundaries as well as the evaluation scores. Report the optimal configuration and its corresponding scores. How does changing the depth improve or worsen the performance of the decision tree, specifically in the context of this dataset? (5 points)

Your answer:

[18]:
```
#Setup0
clf = DecisionTreeClassifier(criterion = 'gini',random_state=seed, max_depth=3)
clf.fit(wine_X_train[:,:2],wine_y_train)
plot_decision_boundary(
    clf,
    wine_X_train,
    wine_y_train,
    (feature1, feature2),
    feature_names,
    target_names
)
accuracy = clf.score(wine_X_test[:,:2],wine_y_test)
precision = precision_score(wine_y_test,clf.predict(wine_X_test[:,:2]),␣
↪average='macro')
f1 = f1_score(wine_y_test,clf.predict(wine_X_test[:,:2]), average='macro')

print("Accuracy: ",accuracy)
print("Precision: ",precision)
print("F1 score: ",f1)
```

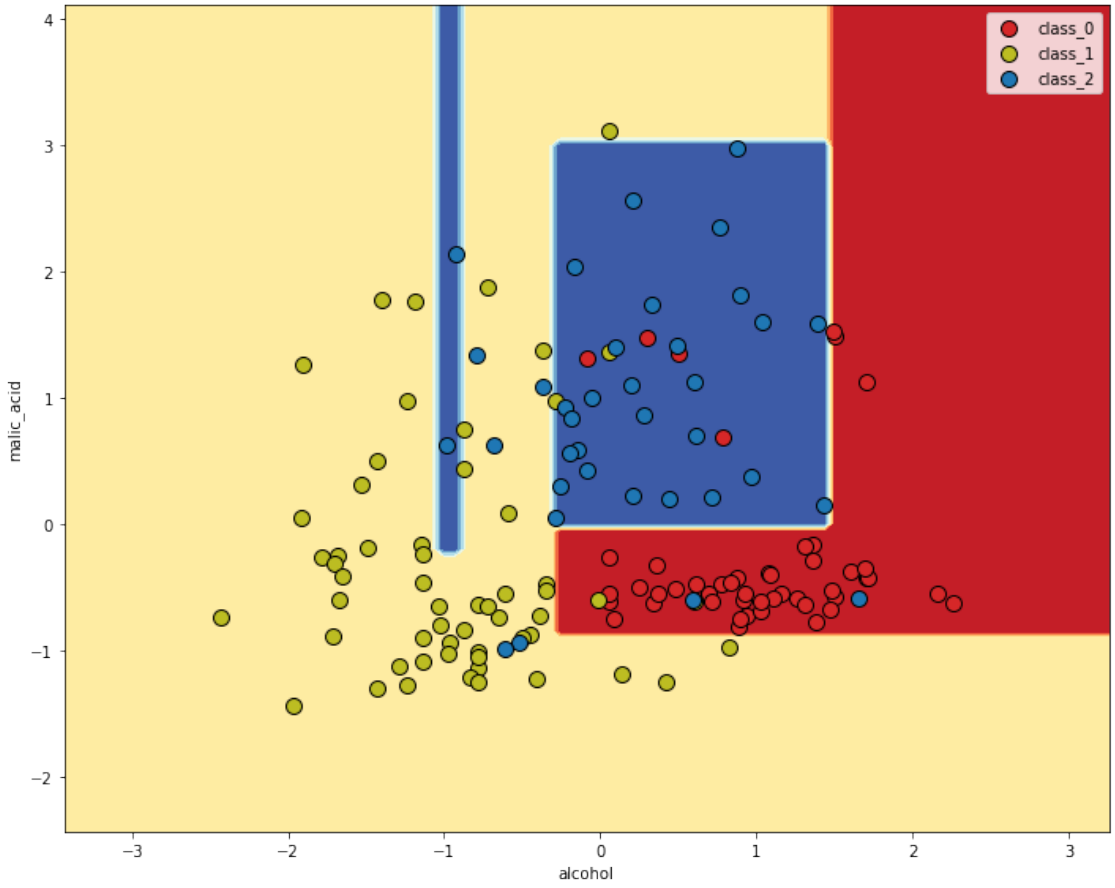
```
Accuracy:  0.75
Precision:  0.7430555555555555
F1 score:  0.7313131313131312
```



[19]:
```
#Setup1
clf = DecisionTreeClassifier(criterion = 'entropy',random_state=seed,␣
 ↪max_depth=4)
clf.fit(wine_X_train[:,:2],wine_y_train)
plot_decision_boundary(
    clf,
    wine_X_train,
    wine_y_train,
    (feature1, feature2),
    feature_names,
    target_names
)
accuracy = clf.score(wine_X_test[:,:2],wine_y_test)
precision = precision_score(wine_y_test,clf.predict(wine_X_test[:,:2]),␣
 ↪average='macro')
f1 = f1_score(wine_y_test,clf.predict(wine_X_test[:,:2]), average='macro')

print("Accuracy: ",accuracy)
print("Precision: ",precision)
```

```
print("F1 score: ",f1)
```

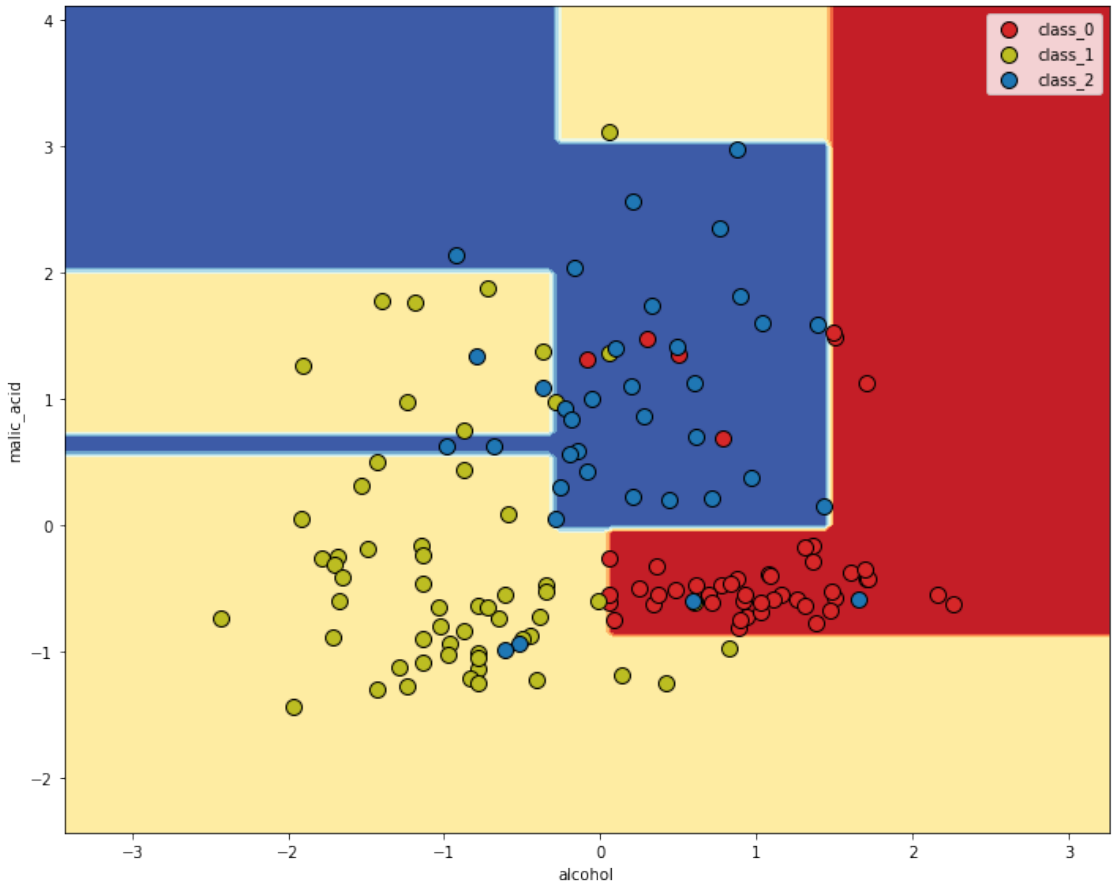
```
Accuracy:  0.7777777777777778
Precision:  0.7703703703703703
F1 score:  0.7634970393591082
```



[20]:
```
#Setup2
clf = DecisionTreeClassifier(criterion = 'gini',random_state=seed, max_depth=4)
clf.fit(wine_X_train[:,:2],wine_y_train)
plot_decision_boundary(
    clf,
    wine_X_train,
    wine_y_train,
    (feature1, feature2),
    feature_names,
    target_names
)
accuracy = clf.score(wine_X_test[:,:2],wine_y_test)
```

```
precision = precision_score(wine_y_test,clf.predict(wine_X_test[:,:2]),␣
 ↪average='macro')
f1 = f1_score(wine_y_test,clf.predict(wine_X_test[:,:2]), average='macro')

print("Accuracy: ",accuracy)
print("Precision: ",precision)
print("F1 score: ",f1)
```

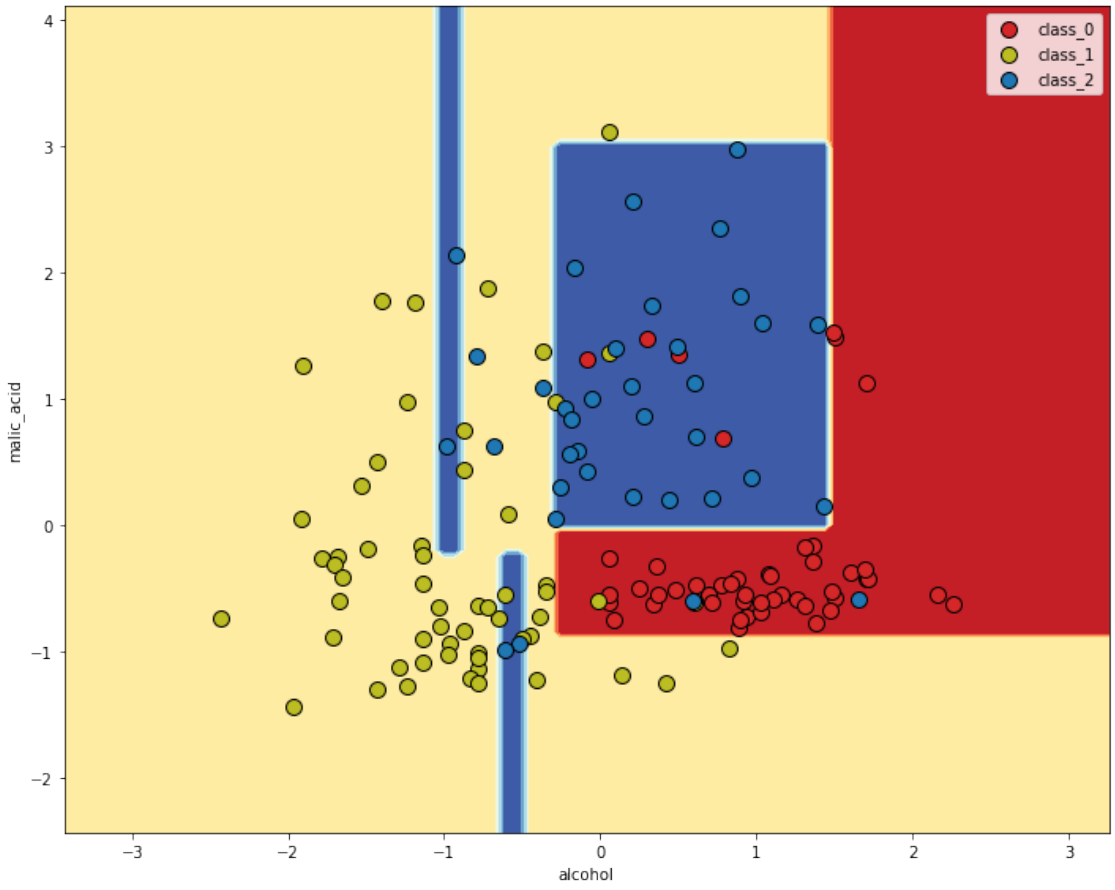
```
Accuracy:  0.75
Precision:  0.7592592592592592
F1 score:  0.7386695906432749
```



[21]:
```
#Setup3
clf = DecisionTreeClassifier(criterion = 'entropy',random_state=seed,␣
 ↪max_depth=5)
clf.fit(wine_X_train[:,:2],wine_y_train)
plot_decision_boundary(
    clf,
    wine_X_train,
```

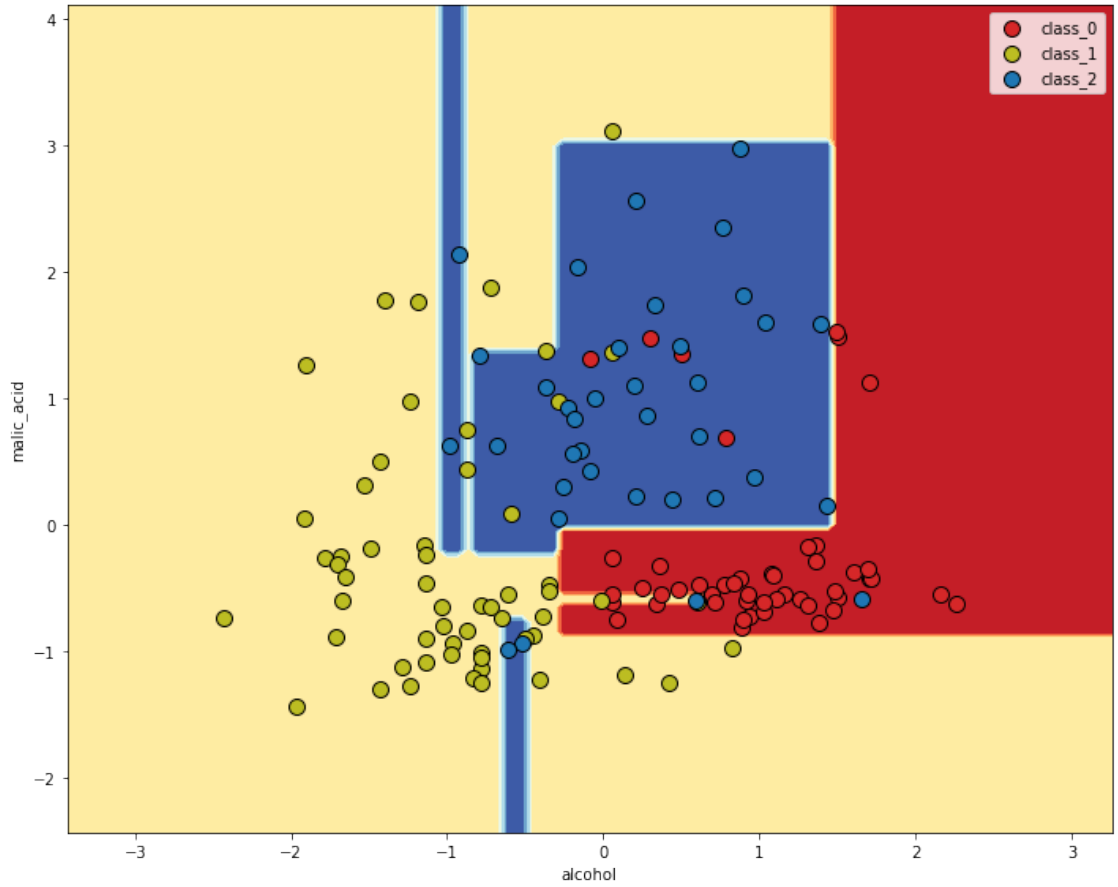
```
    wine_y_train,
    (feature1, feature2),
    feature_names,
    target_names
)
accuracy = clf.score(wine_X_test[:,:2],wine_y_test)
precision = precision_score(wine_y_test,clf.predict(wine_X_test[:,:2]),␣
 ↪average='macro')
f1 = f1_score(wine_y_test,clf.predict(wine_X_test[:,:2]), average='macro')

print("Accuracy: ",accuracy)
print("Precision: ",precision)
print("F1 score: ",f1)
```

```
Accuracy:  0.7777777777777778
Precision:  0.7703703703703703
F1 score:  0.7634970393591082
```



[22]:
```
#Setup4
clf = DecisionTreeClassifier(criterion = 'entropy',random_state=seed,␣
 ↪max_depth=6)
clf.fit(wine_X_train[:,:2],wine_y_train)
plot_decision_boundary(
    clf,
    wine_X_train,
    wine_y_train,
    (feature1, feature2),
    feature_names,
    target_names
)
accuracy = clf.score(wine_X_test[:,:2],wine_y_test)
precision = precision_score(wine_y_test,clf.predict(wine_X_test[:,:2]),␣
 ↪average='macro')
f1 = f1_score(wine_y_test,clf.predict(wine_X_test[:,:2]), average='macro')

print("Accuracy: ",accuracy)
print("Precision: ",precision)
print("F1 score: ",f1)
```

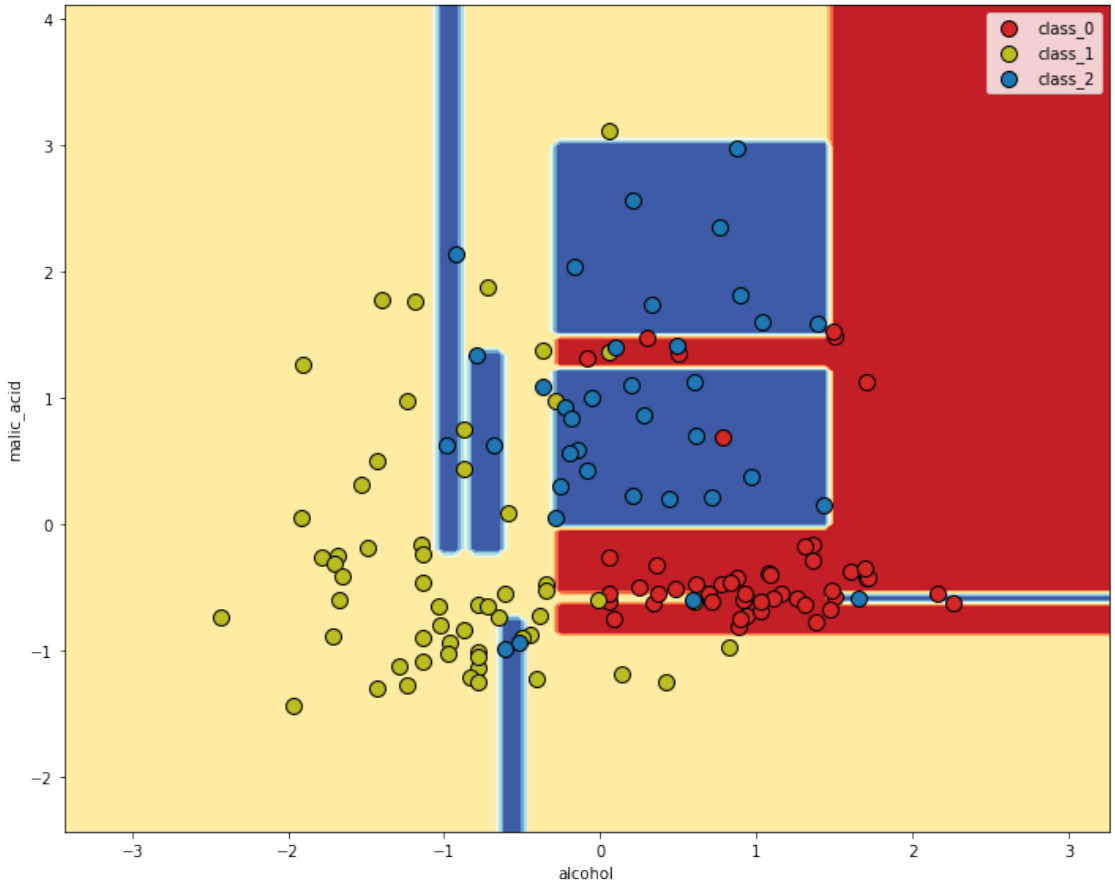
```
Accuracy:  0.75
Precision:  0.75
F1 score:  0.7466977466977466
```



[23]:
```
#Setup5
clf = DecisionTreeClassifier(criterion = 'entropy',random_state=seed,␣
 ↪max_depth=7)
clf.fit(wine_X_train[:,:2],wine_y_train)
plot_decision_boundary(
    clf,
    wine_X_train,
    wine_y_train,
    (feature1, feature2),
    feature_names,
    target_names
)
accuracy = clf.score(wine_X_test[:,:2],wine_y_test)
precision = precision_score(wine_y_test,clf.predict(wine_X_test[:,:2]),␣
 ↪average='macro')
f1 = f1_score(wine_y_test,clf.predict(wine_X_test[:,:2]), average='macro')

print("Accuracy: ",accuracy)
print("Precision: ",precision)
```

```
print("F1 score: ",f1)
```

```
Accuracy:  0.6944444444444444
Precision:  0.6871794871794873
F1 score:  0.6853071780608012
```



The optimal configurations are:

criterion = entropy

max_depth = 4 and 5

Scores:

Accuracy: 0.7777777777777778

Precision: 0.7703703703703703

F1 score: 0.7634970393591082

Upon increasing the depth to 7 from 5, there was a fall in the performance.

5 Section 2. Support Vector Machines [25 points]

In this section, you will experiment with different datasets using SciKit-Learn's implementation of support vector machines (SVMs). This will give you better intuitions about utilizing SVMs when it comes to different two-dimensional datasets. You will also implement a Gaussian kernel for non-linear SVM classification.

5.1 2.1. Linear SVM (5 points)

In this subsection, we will experiment with a dataset that can be separated with a linear decision boundary. We want to experiment with different C values to understand their effects on our linear decision boundary.

Let's first load the dataset (`svm_data1.mat` in MatLab format) using SciPy's `loadmat()` function.

[24]:
```
data1 = loadmat(data_path / "svm_data1.mat")
X1, y1 = data1["X"], data1["y"][:, 0] # "X" and "y" are keys for this dataset
X1.shape, y1.shape # 2D samples + labels
```

[24]:
```
((51, 2), (51,))
```

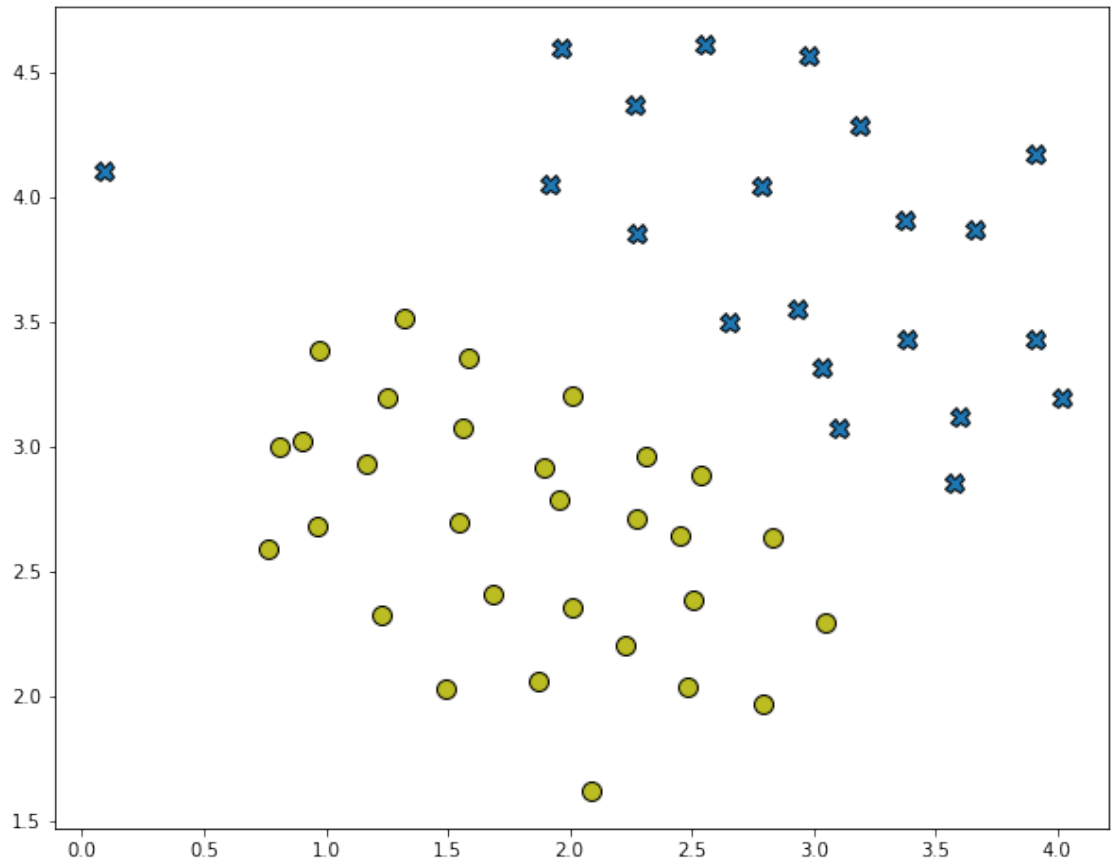
We provide code for plotting data and SVM decision boundaries below:

[25]:
```
def plot_data(X, y, ax=None):
    """Plot 2D dataset."""
    positive = (y == 1)
    negative = (y == 0)
    if ax is None:
        fig, ax = plt.subplots()
    ax.plot(X[positive, 0], X[positive, 1], "X", mew=1, ms=10, mec="k")
    ax.plot(X[negative, 0], X[negative, 1], "o", mew=1, mfc="tab:olive", ms=10,␣
 ↪mec="k")


def plot_linear_boundary(X, y, model, ax=None):
    """Plot linear boundary."""
    if model is None:
        return
    w = model.coef_[0]    # the theta of your SVM classifier
    b = model.intercept_  # the bias of your SVM classifier
    xp = np.array([np.min(X[:, 0]), np.max(X[:, 0])])
    yp = -(w[0] * xp + b) / w[1]
    if ax is None:
        fig, ax = plt.subplots()
    plot_data(X, y, ax)
    ax.plot(xp, yp)
```

```
def plot_nonlinear_boundary(X, y, model, ax=None):
    """Contour plot that delineates a nonlinear boundary."""
    if model is None:
        return

    num_points = math.floor(X.shape[0]/10)
    x1 = np.linspace(min(X[:, 0]), max(X[:, 0]), num_points)
    x2 = np.linspace(min(X[:, 1]), max(X[:, 1]), num_points)
    X1, X2 = np.meshgrid(x1, x2)
    vals = np.zeros(X1.shape)
    for i in range(X1.shape[1]):
        X_ = np.stack((X1[:, i], X2[:, i]), axis=1)
        vals[:, i] = model.predict(X_)
    if ax is None:
        fig, ax = plt.subplots()
    ax.contourf(X1, X2, vals, cmap="YlGnBu", alpha=0.2)
    plot_data(X, y, ax)
```

We want to first visualize our dataset in two-dimensional space:

[26]:
```
# Plot training data
plot_data(X1, y1)
```



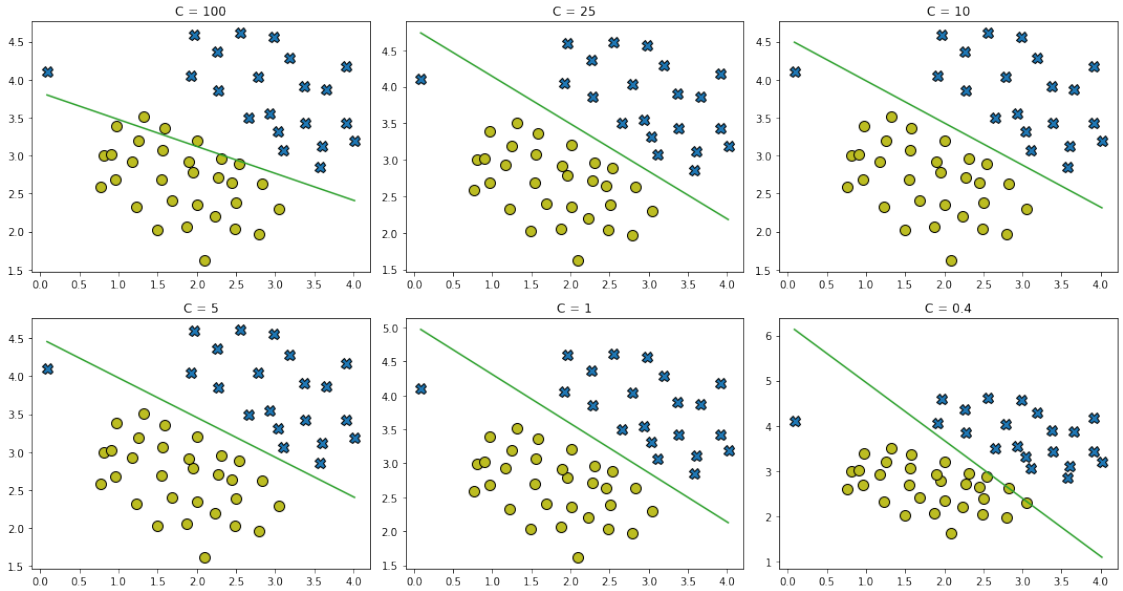
(1) Describe the dataset. How does it lend itself well to the use of an SVM? Are there any abnormalities that may affect our model's performance? (2 points)

Your answer: The dataset has two classes of data. This distribution of data is a good fit for SVM as can be in the plot above. SVM would try to draw a hyperplane while trying to maintain the max distance from the distributions and as can be seen the data is quite uniformly distributed around the gap between them where a hyperplane (a line since this is in R2) could be fit in. However, despite the favorable distribution of the data, there is a sample from class "1" which is an outlier and thus will affect the model's performance.

Now, we want to train linear SVMs for our dataset. You should pick **6 different** C **values** to train your SVMs, then plot all of them side by side for convenient juxtaposition. You should train your models with **L2 penalty** and **hinge loss**. Use the provided `seed`. We provide code for side-by-side plots below.

[27]:
```
from sklearn import svm
import math
from sklearn.svm import LinearSVC

# TODO: Replace None's with appropriate values
Cs = [100, 25, 10, 5, 1, 0.4]
```

```
cols = 3
rows = math.ceil(len(Cs) / cols)

fig, axes = plt.subplots(rows, cols, figsize=(15, 8))
for row in range(rows):
    for col in range(cols):
        C = Cs[row * cols + col]
        # TODO: Train SVM
        svm_clf = LinearSVC( penalty='l2',loss='hinge',C = C, random_state =␣
 ↪seed)
        svm_clf.fit(X1,y1)
        # End of code
        plot_linear_boundary(X1, y1, svm_clf, axes[row, col])
        axes[row, col].set_title(f"C = {C}")
plt.tight_layout()
```



(2) What does C intuitively represent, and how does varying it affect our SVM's decision boundary, according to the graphs? Take into account any abnormalities mentioned above. (3 points)

Your answer: The C parameter tells the SVM optimization how much to avoid misclassifying each training example. When C is large, the distance between the distribution and the hyperplane is more because it tries harder not to misclassify so samples of different classes are further away. Conversely, a very small value of C will cause the optimizer to make a hyperplane that has very little distance between samples and itself, even if that hyperplane misclassifies more points. Thus, when C = 100, it tries not to misclassify even the outlier and ends of having a non optimal hyperplane while when C = 0.4, the margin between samples and hyperplane is so less it ends up misclassifying

as well.

5.2 2.2. Kernel SVM (20 points)

In this part of the homework, you will perform non-linear classification using SVMs, with Gaussian kernels in particular.

5.2.1 2.2.1. Gaussian Kernel (10 points)

Similarly to Gaussian basis functions, we can use Gaussian kernels to find non-linear decision boundaries. As mentioned in lectures and chapter 6.2 of the Bishop textbook (page 296), these kernels are of the form:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right\}$$

Implement a Gaussian kernel matrix for our SVMs below. We will be using `sklearn.svm.SVC` as our model.

[28]:
```
def gaussian_kernel(x, x_prime, sigma):
    k = np.zeros((x.shape[0], x_prime.shape[0]))
    # TODO: Implement Gaussian kernel matrix
    for i,x in enumerate(x):
      for j,y in enumerate(x_prime):
        k[i,j]=np.exp((-1*np.linalg.norm(x-y)**2)/(2.0*(sigma)**2))
    # End of code
    return k
```

Note that according to the documentation, our resulting kernel matrix should have size `(n_samples, n_samples)`. Once you have completed the implemetation of `gaussian_kernel_matrix()`, the following cell will test your kernel function on two provided examples, after which you should expect to see a value of:

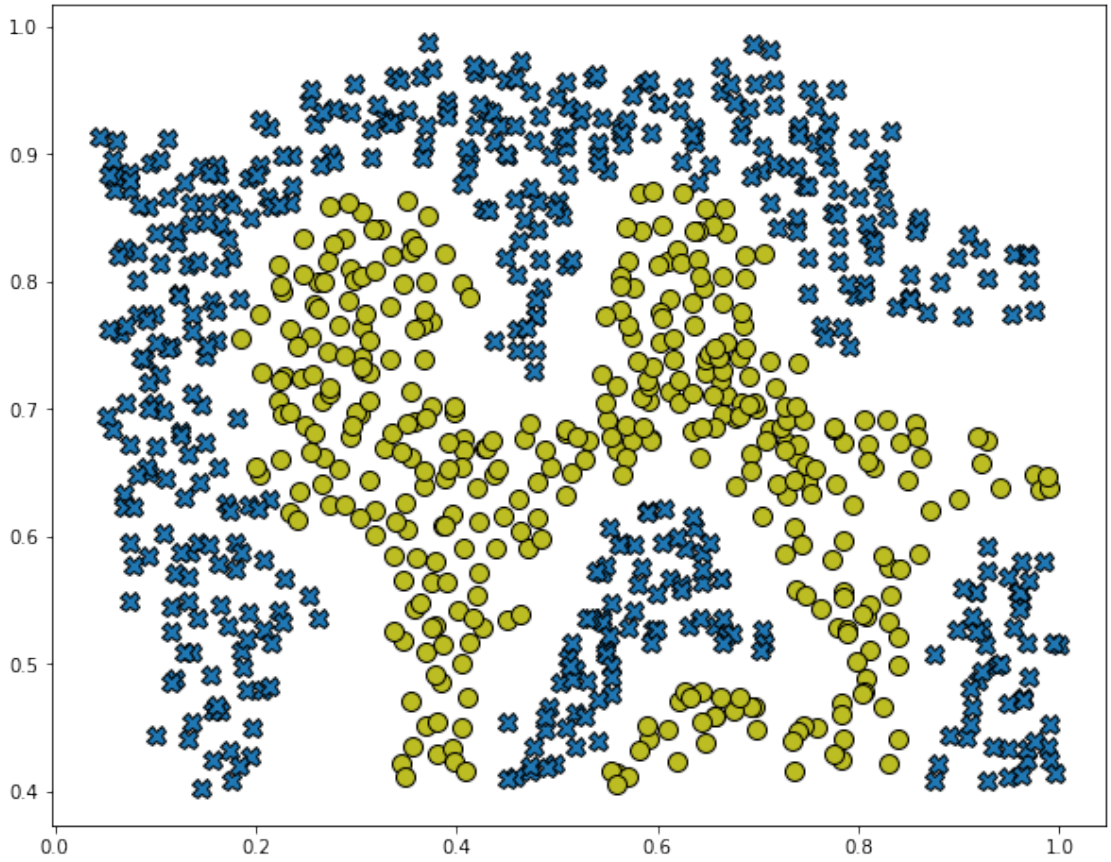
[29]:
```
x1 = np.array([[1, 2, 1],[0, 4, -1]])
x2 = np.array([[1, 2, 1],[0, 4, -1]])
sigma = 2.0

kernel_matrix = gaussian_kernel(x1, x2, sigma)
print(kernel_matrix)
```

```
[[1.         0.32465247]
 [0.32465247 1.        ]]
```

5.2.2 2.2.2. Training SVM with a Gaussian Kernel (5 points)

Let's demonstrate a Gaussian-kernel SVM on a non-linear dataset:

[30]:
```
data2 = loadmat(data_path / "svm_data2.mat")
X2, y2 = data2["X"], data2["y"][:, 0]
plot_data(X2, y2)
```

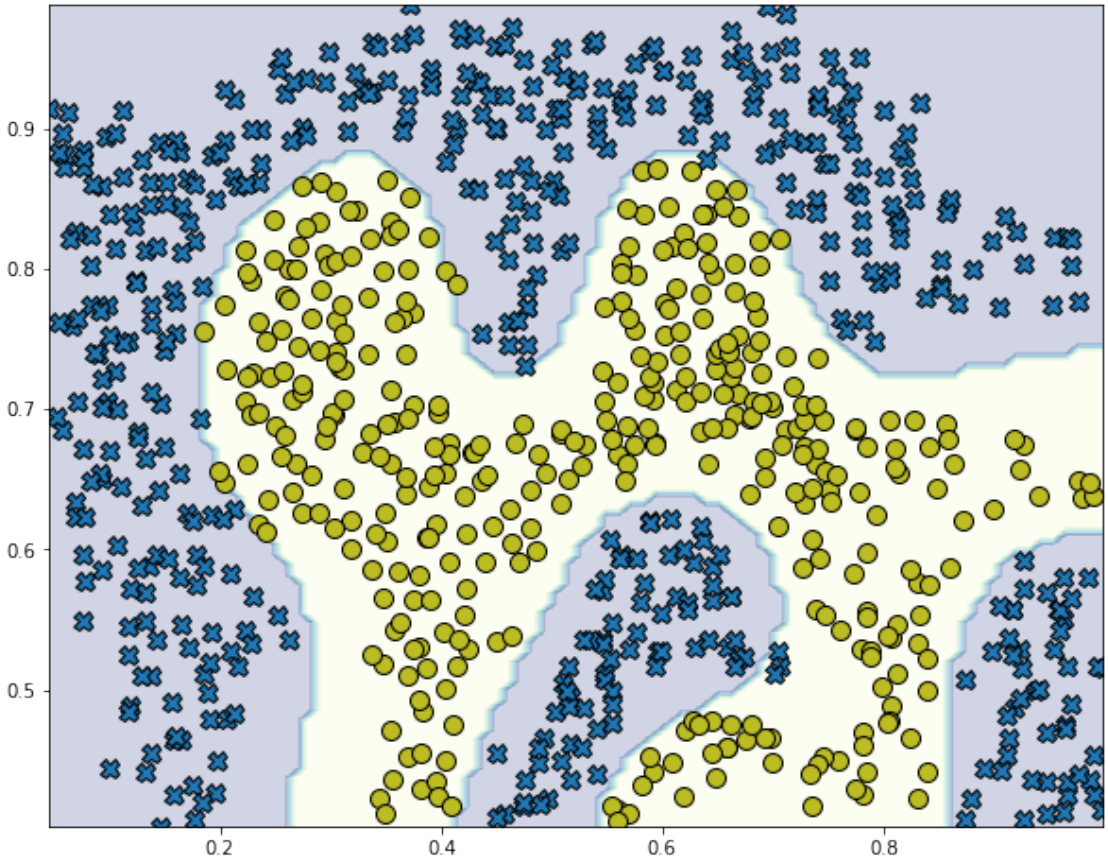


Apply a Gaussian kernel to your non-linear SVC model below; an SVC example with a custom kernel is available here. Note the kernel's input arguments.

[31]:
```
def kernel_wrapper(f, sigma):
  return lambda x, x_prime: f(x, x_prime, sigma)
```

[32]:
```
from sklearn.svm import SVC
```

[33]:
```
sigma = 0.1
# TODO: Apply Gaussian kernel on SVC with provided seed as random_state
model = SVC(kernel=kernel_wrapper(gaussian_kernel, sigma), random_state=seed)
model.fit(X2,y2)
# End of code
plot_nonlinear_boundary(X2, y2, model) # note that this step could take up to 1␣
↪minute
```

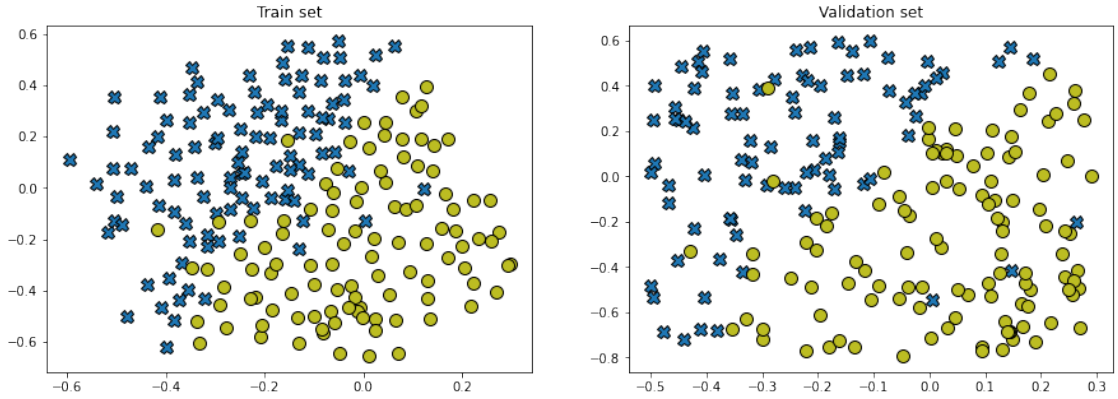


5.2.3 2.2.3. Grid Search Cross Validation (5 points)

In this part of the homework, you will utilize cross validation with a validation set to fine-tune your model. You will use Gaussian-kernel SVMs for this task. From the provided dataset (`svm_data3.mat`), you are given a train and validation set:

[34]:
```
data3 = loadmat(data_path / "svm_data3.mat")

X_train = data3["X"]
y_train = data3["y"][:, 0]
X_val = data3["Xval"]
y_val = data3["yval"][:, 0]

# Plot training and validation data
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))
plot_data(X_train, y_train, ax1)
plot_data(X_val, y_val, ax2)
ax1.set_title("Train set")
ax2.set_title("Validation set")
```

[34]: Text(0.5, 1.0, 'Validation set')



Here, you are to use the validation set (`X_val`, `y_val`) to determine the best combination of C and σ parameters for your SVM, using accuracy as your metric. We suggest trying values in multiplicative steps for these parameters (e.g., 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30). You should try all possible pairs of values for C and σ. For example, trying every possible combination of the 2 parameters among the 8 values listed above would result in $8^2 = 64$ different models being trained and validated. Because this could take a long time, we suggest reserving time to work on this section.

Write code to determine the best combination of C and σ as well as to return the corresponding values in `search_hyperparameter()` below.

[35]:
```
from sklearn.model_selection import GridSearchCV

def return_kernel_values(sigmas):
      kernel_values=[]
      for sigma in sigmas:
        kernel_values.append(kernel_wrapper(gaussian_kernel, sigma))
      return kernel_values
```

[39]:
```
def search_hyperparameter(X_train, y_train, X_val, y_val, Cs, sigmas):
    # TODO: Grid search
    hyperparameter_space={'C':Cs, 'kernel': return_kernel_values(sigmas)}
    gs=GridSearchCV(SVC(kernel=kernel_wrapper(gaussian_kernel, sigma),␣
 ↪random_state=seed),param_grid=hyperparameter_space,scoring="accuracy",n_jobs=4,␣
 ↪return_train_score=True, verbose=True)

    gs.fit(X_val, y_val)
    best_C = gs.best_params_['C']
    best_sigma = gs.best_params_['kernel']
    best_accuracy = gs.best_score_
    best_model = gs.best_estimator_
    return best_C, best_sigma, best_accuracy, best_model
```

Report the best performing parameters along with the accuracy score on `X_val`. You should be able to get an accuracy higher than 0.9. The optimal parameters may not be unique.

[40]:
```
Cs = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]
sigmas = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]

# TODO: Try different SVM hyperparameters
best_C, best_sigma, best_accuracy, best_model = search_hyperparameter(X_train,␣
↪y_train, X_val, y_val, Cs, sigmas)

print("Best C value: ", best_C)
print("Best sigma value: ", best_sigma)
print("Optimum accuracy score: ", best_accuracy)
```

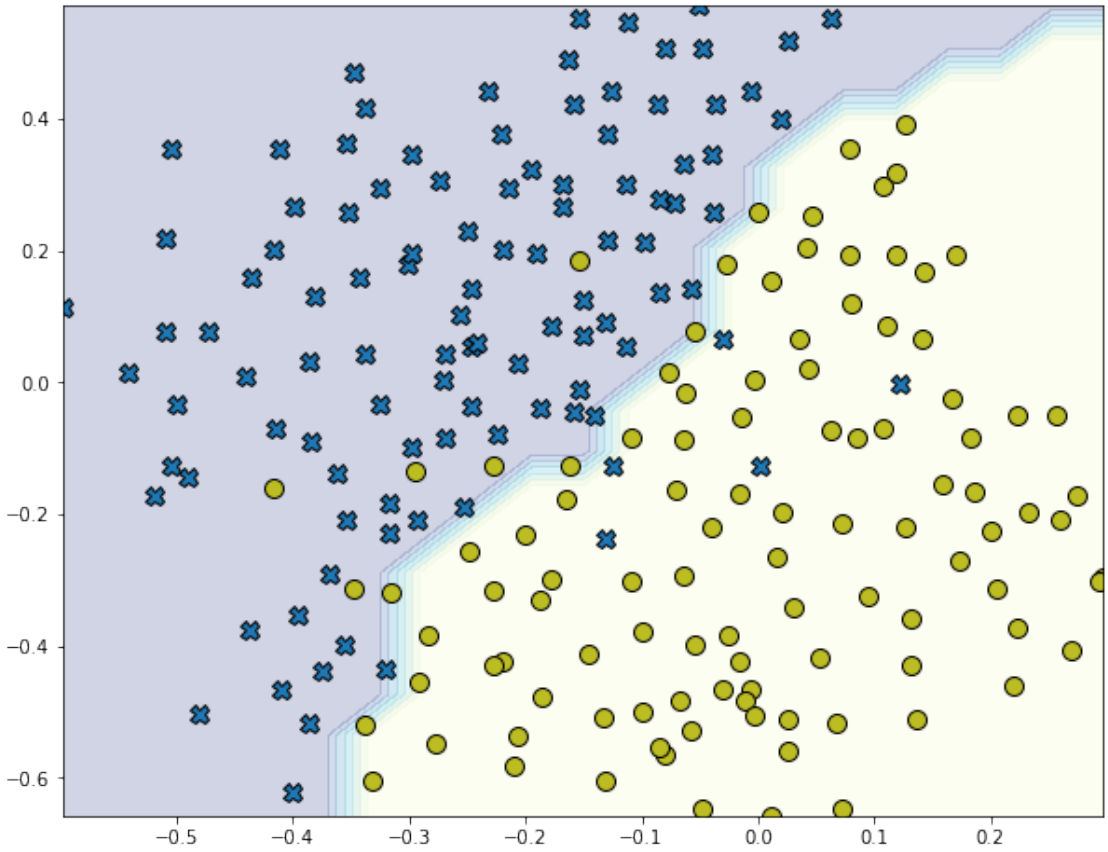
```
Fitting 5 folds for each of 64 candidates, totalling 320 fits
Best C value:  0.3
Best sigma value:  <function kernel_wrapper.<locals>.<lambda> at 0x7f5c295e6320>
Optimum accuracy score:  0.9400000000000001
```

[41]:
```
print("Optimal paramters", best_model)
```

```
Optimal paramters SVC(C=0.3, kernel=<function kernel_wrapper.<locals>.<lambda>
at 0x7f5c295e6320>,
    random_state=5824)
```

Lastly, train the model again with your optimal parameters and plot the decision boundary.

[42]:
```
# TODO: Train and plot model with best configuration
best_model.fit(X_train,y_train)
plot_nonlinear_boundary(X_train, y_train, best_model)
```



[ ]:
```
!apt-get -qq install texlive texlive-xetex texlive-latex-extra pandoc
!pip install --quiet pypandoc
```

```
Extracting templates from packages: 100%
Preconfiguring packages …
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 123942 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1_all.deb …
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2_all.deb …
Unpacking fonts-lato (2.0-2) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/02-poppler-data_0.4.8-2_all.deb …
Unpacking poppler-data (0.4.8-2) …
Selecting previously unselected package tex-common.
Preparing to unpack …/03-tex-common_6.09_all.deb …
Unpacking tex-common (6.09) …
Selecting previously unselected package fonts-lmodern.
Preparing to unpack …/04-fonts-lmodern_2.004.5-3_all.deb …
```

```
Unpacking fonts-lmodern (2.004.5-3) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/05-fonts-noto-mono_20171026-2_all.deb …
Unpacking fonts-noto-mono (20171026-2) …
Selecting previously unselected package fonts-texgyre.
Preparing to unpack …/06-fonts-texgyre_20160520-1_all.deb …
Unpacking fonts-texgyre (20160520-1) …
Selecting previously unselected package javascript-common.
Preparing to unpack …/07-javascript-common_11_all.deb …
Unpacking javascript-common (11) …
Selecting previously unselected package libcupsfilters1:amd64.
Preparing to unpack …/08-libcupsfilters1_1.20.2-0ubuntu3.1_amd64.deb …
Unpacking libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) …
Selecting previously unselected package libcupsimage2:amd64.
Preparing to unpack …/09-libcupsimage2_2.2.7-1ubuntu2.9_amd64.deb …
Unpacking libcupsimage2:amd64 (2.2.7-1ubuntu2.9) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/10-libijs-0.35_0.35-13_amd64.deb …
Unpacking libijs-0.35:amd64 (0.35-13) …
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack …/11-libjbig2dec0_0.13-6_amd64.deb …
Unpacking libjbig2dec0:amd64 (0.13-6) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/12-libgs9-common_9.26~dfsg+0-0ubuntu0.18.04.17_all.deb
…
Unpacking libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.17) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/13-libgs9_9.26~dfsg+0-0ubuntu0.18.04.17_amd64.deb …
Unpacking libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.17) …
Selecting previously unselected package libjs-jquery.
Preparing to unpack …/14-libjs-jquery_3.2.1-1_all.deb …
Unpacking libjs-jquery (3.2.1-1) …
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack …/15-libkpathsea6_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libpotrace0.
Preparing to unpack …/16-libpotrace0_1.14-2_amd64.deb …
Unpacking libpotrace0 (1.14-2) …
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack …/17-libptexenc1_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package rubygems-integration.
Preparing to unpack …/18-rubygems-integration_1.11_all.deb …
Unpacking rubygems-integration (1.11) …
Selecting previously unselected package ruby2.5.
Preparing to unpack …/19-ruby2.5_2.5.1-1ubuntu1.12_amd64.deb …
```

```
Unpacking ruby2.5 (2.5.1-1ubuntu1.12) …
Selecting previously unselected package ruby.
Preparing to unpack …/20-ruby_1%3a2.5.1_amd64.deb …
Unpacking ruby (1:2.5.1) …
Selecting previously unselected package rake.
Preparing to unpack …/21-rake_12.3.1-1ubuntu0.1_all.deb …
Unpacking rake (12.3.1-1ubuntu0.1) …
Selecting previously unselected package ruby-did-you-mean.
Preparing to unpack …/22-ruby-did-you-mean_1.2.0-2_all.deb …
Unpacking ruby-did-you-mean (1.2.0-2) …
Selecting previously unselected package ruby-minitest.
Preparing to unpack …/23-ruby-minitest_5.10.3-1_all.deb …
Unpacking ruby-minitest (5.10.3-1) …
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack …/24-ruby-net-telnet_0.1.1-2_all.deb …
Unpacking ruby-net-telnet (0.1.1-2) …
Selecting previously unselected package ruby-power-assert.
Preparing to unpack …/25-ruby-power-assert_0.3.0-1_all.deb …
Unpacking ruby-power-assert (0.3.0-1) …
Selecting previously unselected package ruby-test-unit.
Preparing to unpack …/26-ruby-test-unit_3.2.5-1_all.deb …
Unpacking ruby-test-unit (3.2.5-1) …
Selecting previously unselected package libruby2.5:amd64.
Preparing to unpack …/27-libruby2.5_2.5.1-1ubuntu1.12_amd64.deb …
Unpacking libruby2.5:amd64 (2.5.1-1ubuntu1.12) …
Selecting previously unselected package libsynctex1:amd64.
Preparing to unpack …/28-libsynctex1_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libtexlua52:amd64.
Preparing to unpack …/29-libtexlua52_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
…/30-libtexluajit2_2017.20170613.44572-8ubuntu0.1_amd64.deb …
Unpacking libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack …/31-libzzip-0-13_0.13.62-3.1ubuntu0.18.04.1_amd64.deb …
Unpacking libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) …
Selecting previously unselected package lmodern.
Preparing to unpack …/32-lmodern_2.004.5-3_all.deb …
Unpacking lmodern (2.004.5-3) …
Selecting previously unselected package preview-latex-style.
Preparing to unpack …/33-preview-latex-style_11.91-1ubuntu1_all.deb …
Unpacking preview-latex-style (11.91-1ubuntu1) …
Selecting previously unselected package t1utils.
Preparing to unpack …/34-t1utils_1.41-2_amd64.deb …
```

```
Unpacking t1utils (1.41-2) …
Selecting previously unselected package tex-gyre.
Preparing to unpack …/35-tex-gyre_20160520-1_all.deb …
Unpacking tex-gyre (20160520-1) …
Selecting previously unselected package texlive-binaries.
Preparing to unpack …/36-texlive-
binaries_2017.20170613.44572-8ubuntu0.1_amd64.deb …
Unpacking texlive-binaries (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package texlive-base.
Preparing to unpack …/37-texlive-base_2017.20180305-1_all.deb …
Unpacking texlive-base (2017.20180305-1) …
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack …/38-texlive-fonts-recommended_2017.20180305-1_all.deb …
Unpacking texlive-fonts-recommended (2017.20180305-1) …
Selecting previously unselected package texlive-latex-base.
Preparing to unpack …/39-texlive-latex-base_2017.20180305-1_all.deb …
Unpacking texlive-latex-base (2017.20180305-1) …
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack …/40-texlive-latex-recommended_2017.20180305-1_all.deb …
Unpacking texlive-latex-recommended (2017.20180305-1) …
Selecting previously unselected package texlive.
Preparing to unpack …/41-texlive_2017.20180305-1_all.deb …
Unpacking texlive (2017.20180305-1) …
Selecting previously unselected package texlive-pictures.
Preparing to unpack …/42-texlive-pictures_2017.20180305-1_all.deb …
Unpacking texlive-pictures (2017.20180305-1) …
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack …/43-texlive-latex-extra_2017.20180305-2_all.deb …
Unpacking texlive-latex-extra (2017.20180305-2) …
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack …/44-texlive-plain-generic_2017.20180305-2_all.deb …
Unpacking texlive-plain-generic (2017.20180305-2) …
Selecting previously unselected package tipa.
Preparing to unpack …/45-tipa_2%3a1.3-20_all.deb …
Unpacking tipa (2:1.3-20) …
Selecting previously unselected package texlive-xetex.
Preparing to unpack …/46-texlive-xetex_2017.20180305-1_all.deb …
Unpacking texlive-xetex (2017.20180305-1) …
Setting up libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.17) …
Setting up libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up libjs-jquery (3.2.1-1) …
Setting up libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up fonts-droid-fallback (1:6.0.1r16-1.1) …
Setting up libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up tex-common (6.09) …
update-language: texlive-base not installed and configured, doing nothing!
Setting up poppler-data (0.4.8-2) …
```

```
Setting up tex-gyre (20160520-1) …
Setting up preview-latex-style (11.91-1ubuntu1) …
Setting up fonts-texgyre (20160520-1) …
Setting up fonts-noto-mono (20171026-2) …
Setting up fonts-lato (2.0-2) …
Setting up libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) …
Setting up libcupsimage2:amd64 (2.2.7-1ubuntu2.9) …
Setting up libjbig2dec0:amd64 (0.13-6) …
Setting up ruby-did-you-mean (1.2.0-2) …
Setting up t1utils (1.41-2) …
Setting up ruby-net-telnet (0.1.1-2) …
Setting up libijs-0.35:amd64 (0.35-13) …
Setting up rubygems-integration (1.11) …
Setting up libpotrace0 (1.14-2) …
Setting up javascript-common (11) …
Setting up ruby-minitest (5.10.3-1) …
Setting up libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) …
Setting up libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.17) …
Setting up libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up fonts-lmodern (2.004.5-3) …
Setting up ruby-power-assert (0.3.0-1) …
Setting up texlive-binaries (2017.20170613.44572-8ubuntu0.1) …
update-alternatives: using /usr/bin/xdvi-xaw to provide /usr/bin/xdvi.bin
(xdvi.bin) in auto mode
update-alternatives: using /usr/bin/bibtex.original to provide /usr/bin/bibtex
(bibtex) in auto mode
Setting up texlive-base (2017.20180305-1) …
mktexlsr: Updating /var/lib/texmf/ls-R-TEXLIVEDIST…
mktexlsr: Updating /var/lib/texmf/ls-R-TEXMFMAIN…
mktexlsr: Updating /var/lib/texmf/ls-R…
mktexlsr: Done.
tl-paper: setting paper size for dvips to a4:
/var/lib/texmf/dvips/config/config-paper.ps
tl-paper: setting paper size for dvipdfmx to a4:
/var/lib/texmf/dvipdfmx/dvipdfmx-paper.cfg
tl-paper: setting paper size for xdvi to a4: /var/lib/texmf/xdvi/XDvi-paper
tl-paper: setting paper size for pdftex to a4:
/var/lib/texmf/tex/generic/config/pdftexconfig.tex
Setting up texlive-fonts-recommended (2017.20180305-1) …
Setting up texlive-plain-generic (2017.20180305-2) …
Setting up texlive-latex-base (2017.20180305-1) …
Setting up lmodern (2.004.5-3) …
Setting up texlive-latex-recommended (2017.20180305-1) …
Setting up texlive-pictures (2017.20180305-1) …
Setting up tipa (2:1.3-20) …
Regenerating '/var/lib/texmf/fmtutil.cnf-DEBIAN'… done.
Regenerating '/var/lib/texmf/fmtutil.cnf-TEXLIVEDIST'… done.
update-fmtutil has updated the following file(s):
```

```
        /var/lib/texmf/fmtutil.cnf-DEBIAN
        /var/lib/texmf/fmtutil.cnf-TEXLIVEDIST
If you want to activate the changes in the above file(s),
you should run fmtutil-sys or fmtutil.
Setting up texlive (2017.20180305-1) …
Setting up texlive-latex-extra (2017.20180305-2) …
Setting up texlive-xetex (2017.20180305-1) …
Setting up ruby2.5 (2.5.1-1ubuntu1.12) …
Setting up ruby (1:2.5.1) …
Setting up ruby-test-unit (3.2.5-1) …
Setting up rake (12.3.1-1ubuntu0.1) …
Setting up libruby2.5:amd64 (2.5.1-1ubuntu1.12) …
Processing triggers for mime-support (3.60ubuntu1) …
Processing triggers for libc-bin (2.27-3ubuntu1.6) …
Processing triggers for man-db (2.8.3-2ubuntu0.1) …
Processing triggers for fontconfig (2.12.6-0ubuntu2) …
Processing triggers for tex-common (6.09) …
Running updmap-sys. This may take some time… done.
Running mktexlsr /var/lib/texmf … done.
Building format(s) --all.
        This may take some time… done.
```

[ ]:
```
!jupyter nbconvert --to PDF "/content/gdrive/MyDrive/Colab Notebooks/
↪AmartyaDuttaSectionB_HW2.ipynb"
```