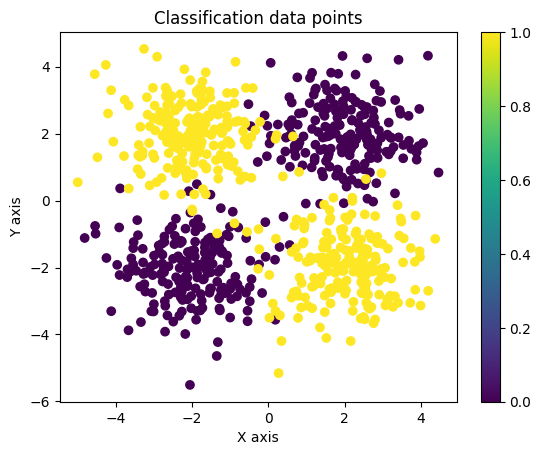
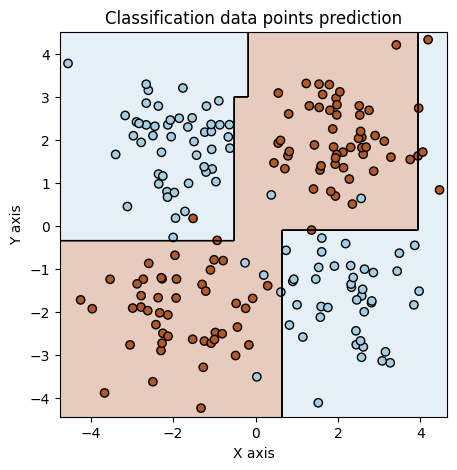
**Decision Tree and SVM Classification Models: Comparison and Hyperparameter Exploration**

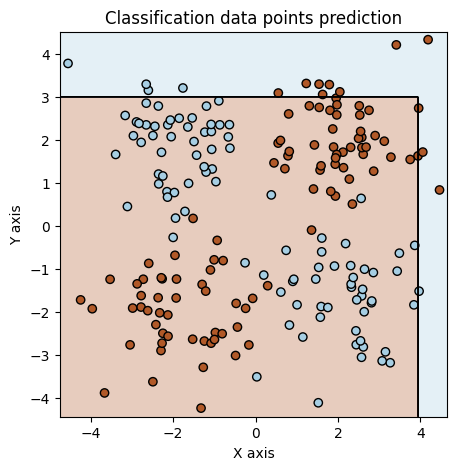
**Original plot:**



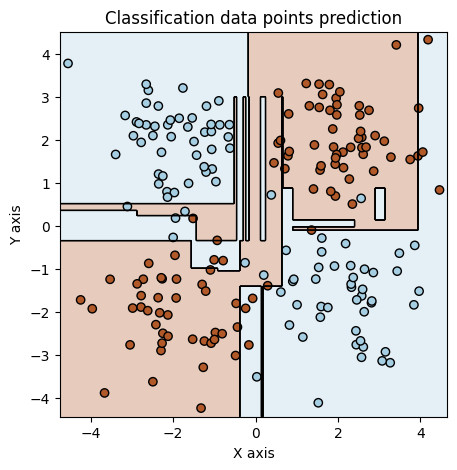
**max\_depth = 5 (Good-fit):**



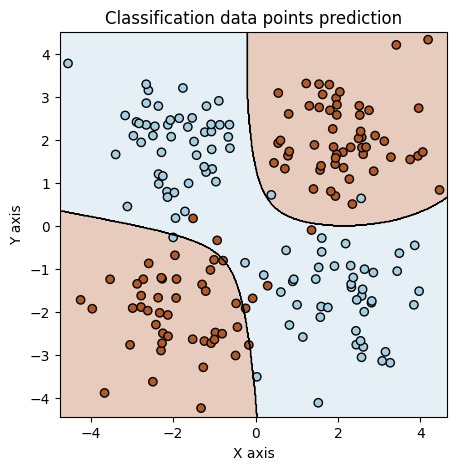
**max\_depth = 2 (Under-fit):**



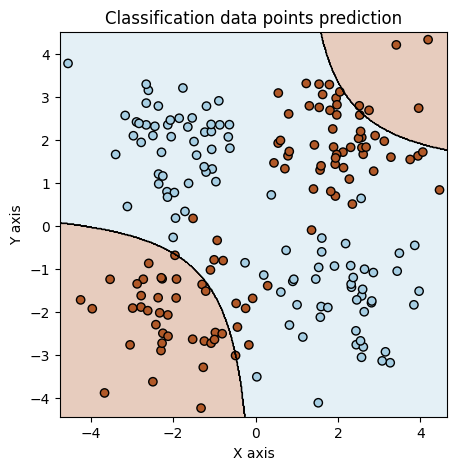
**max\_depth = 100000000000 (Over-fit):**



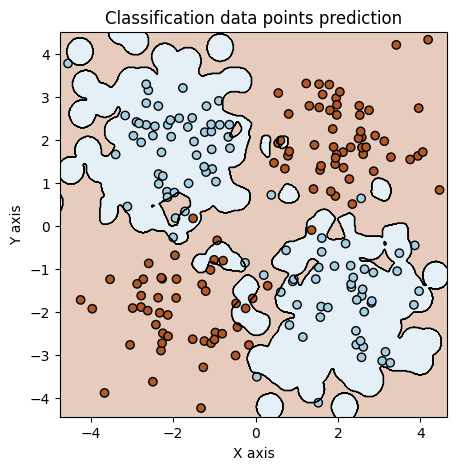
**gamma = 0.1, C =** **10000 (Good fit):**



**gamma = 0.01, C =** **0.1 (Under-fit):**



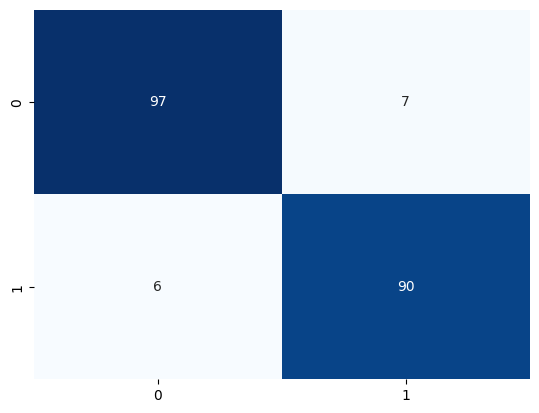
**gamma = 50, C =** **10 (Over-fit):**



**Best parameters for Decision tree:**

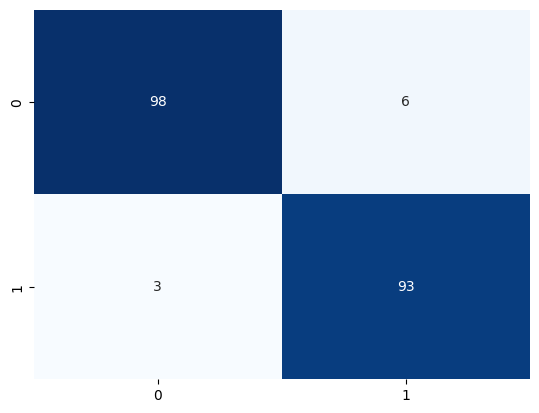
max\_depth = 5  
accuracy ~= 0.93  
Confusion matrix:

**Best parameters for SVM:**



gamma = 0.1

C = 10000  
accuracy ~= 0.96  
Confusion matrix:



**Questions:**

1. How does changing the **max\_depth** hyperparameter affect the Decision Tree Classification model's performance?

2. What do you observe when varying the **C** and **gamma** hyperparameters of the SVM Classification model? How do they affect the model's flexibility and ability to fit the data?

3. From the accuracy and confusion matrix you computed in Step 3, what information do we take?

4. If someone wants to work with this specific dataset, what model would you recommend him work with?

**Answers:**

1. The performance of the model changed radically with the altering of the max\_depth hyperparameter. If it was too low, like max\_depth = 2, the accuracy of the model was very low, so the model was very inaccurate at classifying the unknown data. This can be seen at the plot with the model having only 2 areas for classification, which makes sense as the tree had only 2 nodes and 3 leafs bellow it. At a max\_depth =5, the results were much better, with the accuracy increasing very much (~0.93), so the classification of unknown data was pretty good. The plot also showed much more intuitive areas for the classification of the data, as it followed pretty much the areas of concentration seen at the original plot. At a higher max\_depth, the results started becoming worse though. For example, at an extreme max\_depth = 100000000000, the accuracy fell a bit. There were also signs that the model was trained on the “noise” of the training data with areas of classification (leafs) for very narrow values that led away from the main concentrations of the data.

2. With the change of the C and gamma hyperparameters of the SVM Regression model, there were profound consequences on the accuracy of the model. The gamma value change could be seen visually with the curves of the plot and how well it would follow the cluster of the data. Meanwhile, the C value change could be seen visually with how close the prediction would follow the cluster of the data, or in other words what tolerance it had for outliers in the data. With a smaller C value, there would be a larger margin, so better tolerance for errors. With gamma = 0.1 and C = 10000, the results were very good (accuracy ~0.95). With a very small gamma = 0.01 and a small C = 0.1, the plot would barely even follow the cluster of data, with low accuracy. Finally, with a very high gamma = 50 and fairly high C = 100, the model was over-fitted on the training data, with signs that it “followed” specific training data points as it created small “circles” around them. It also displayed lower accuracy.

3. From the accuracy and the confusion matrix we get how accurate is our model, as well as what kind of errors it makes. The confusion matrix can let us know in more detail what kind of mistakes the model makes as it shows how many True Positives, True Negatives, False Positives and False Negatives our model predicts. The accuracy metric is a bit more general and simple in nature, allowing faster and easier understanding of the overall performance of the model. It is (TP + TN) / Total Predictions.

4. If someone wanted to work with this specific dataset, I would recommend them to work with the SVM. It had better accuracy, as well as the ability to adapt very well with the specific requirements of the person who is using it. Its hyperparameters allow far more fine-tuning than that of the Decision Tree.

**Code:**

The code was written in a .ipybn file for ease of use.

import numpy as np

# Number of samples per class

num\_samples =200

# Generate blue class data

blue\_data = np.random.normal(loc=[2, 2], scale=[1, 1], size=(num\_samples, 2))

# Generate red class data

red\_data = np.random.normal(loc=[-2, -2], scale=[1, 1], size=(num\_samples, 2))

# Generate additional blue class data with mean at [2, -2]

blue\_data\_extra1 = np.random.normal(loc=[2, -2], scale=[1, 1], size=(num\_samples, 2))

# Generate additional red class data with mean at [-2, 2]

red\_data\_extra1 = np.random.normal(loc=[-2, 2], scale=[1, 1], size=(num\_samples, 2))

# Combine data and create labels

X = np.vstack([blue\_data, red\_data, blue\_data\_extra1, red\_data\_extra1])

y = np.array([0] \* (num\_samples \* 2) + [1] \* (num\_samples \* 2))

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.tree import DecisionTreeClassifier

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

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn import svm

def plotDecisionBoundary(clf, X, Y, cmap='Paired\_r'):

    h = 0.02

    x\_min, x\_max = X[:,0].min() - 10\*h, X[:,0].max() + 10\*h

    y\_min, y\_max = X[:,1].min() - 10\*h, X[:,1].max() + 10\*h

    xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),

                         np.arange(y\_min, y\_max, h))

    Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)

    plt.figure(figsize=(5,5))

    plt.contourf(xx, yy, Z, cmap=cmap, alpha=0.30)

    plt.contour(xx, yy, Z, colors='k', linewidths=0.5)

    plt.scatter(X[:,0], X[:,1], c=Y, cmap=cmap, edgecolors='k')

    plt.title("Classification data points prediction")

    plt.xlabel("X axis")

    plt.ylabel("Y axis")

def decisionTree(X\_train, y\_train, X\_test, y\_test, max\_depth):

    decision\_tee\_clf = DecisionTreeClassifier(criterion='entropy', max\_depth=max\_depth)

    decision\_tee\_clf.fit(X\_train, y\_train)

    y\_pred\_dt = decision\_tee\_clf.predict(X\_test)

    print(f'Classification Report: \n {classification\_report (y\_test, y\_pred\_dt)}')

    plotDecisionBoundary(decision\_tee\_clf, X\_test, y\_test)

    return y\_pred\_dt

#rbf kernel: K(x\_1,x\_2)=exp(−γ⋅‖x\_1−x\_2‖^2)

def svmClassification(X\_train, y\_train, X\_test, y\_test, gamma, C):

    svm\_color\_clf = svm.SVC(kernel='rbf', gamma=gamma, C=C)

    svm\_color\_clf.fit(X\_train, y\_train)

    y\_pred\_svm = svm\_color\_clf.predict(X\_test)

    print(f'Classification Report: \n {classification\_report (y\_test, y\_pred\_svm)}')

    plotDecisionBoundary(svm\_color\_clf, X\_test, y\_test)

    return y\_pred\_svm

plt.scatter(X[:,0], X[:,1], c=y)

plt.title("Classification data points")

plt.xlabel("X axis")

plt.ylabel("Y axis")

plt.colorbar()

plt.show()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25)

#Good fit

y\_pred\_dt = decisionTree(X\_train, y\_train, X\_test, y\_test, 5)

cf\_matrix = confusion\_matrix(y\_test, y\_pred\_dt)

sns.heatmap(cf\_matrix, annot= True, fmt='d', cmap='Blues', cbar=False)

#Underfitting

y\_pred\_dt = decisionTree(X\_train, y\_train, X\_test, y\_test, 2)

cf\_matrix = confusion\_matrix(y\_test, y\_pred\_dt)

sns.heatmap(cf\_matrix, annot= True, fmt='d', cmap='Blues', cbar=False)

#Overfitting

y\_pred\_dt = decisionTree(X\_train, y\_train, X\_test, y\_test, 100000000000)

cf\_matrix = confusion\_matrix(y\_test, y\_pred\_dt)

sns.heatmap(cf\_matrix, annot= True, fmt='d', cmap='Blues', cbar=False)

#######SVM#######

#Good fit

y\_pred\_svm = svmClassification(X\_train, y\_train, X\_test, y\_test, 0.1, 10000)

cf\_matrix = confusion\_matrix(y\_test, y\_pred\_svm)

sns.heatmap(cf\_matrix, annot= True, fmt='d', cmap='Blues', cbar=False)

#Underfitting

y\_pred\_svm = svmClassification(X\_train, y\_train, X\_test, y\_test, 0.01, 0.1)

cf\_matrix = confusion\_matrix(y\_test, y\_pred\_svm)

sns.heatmap(cf\_matrix, annot= True, fmt='d', cmap='Blues', cbar=False)

#Overfitting

y\_pred\_svm = svmClassification(X\_train, y\_train, X\_test, y\_test, 50, 10000)

cf\_matrix = confusion\_matrix(y\_test, y\_pred\_svm)

sns.heatmap(cf\_matrix, annot= True, fmt='d', cmap='Blues', cbar=False)