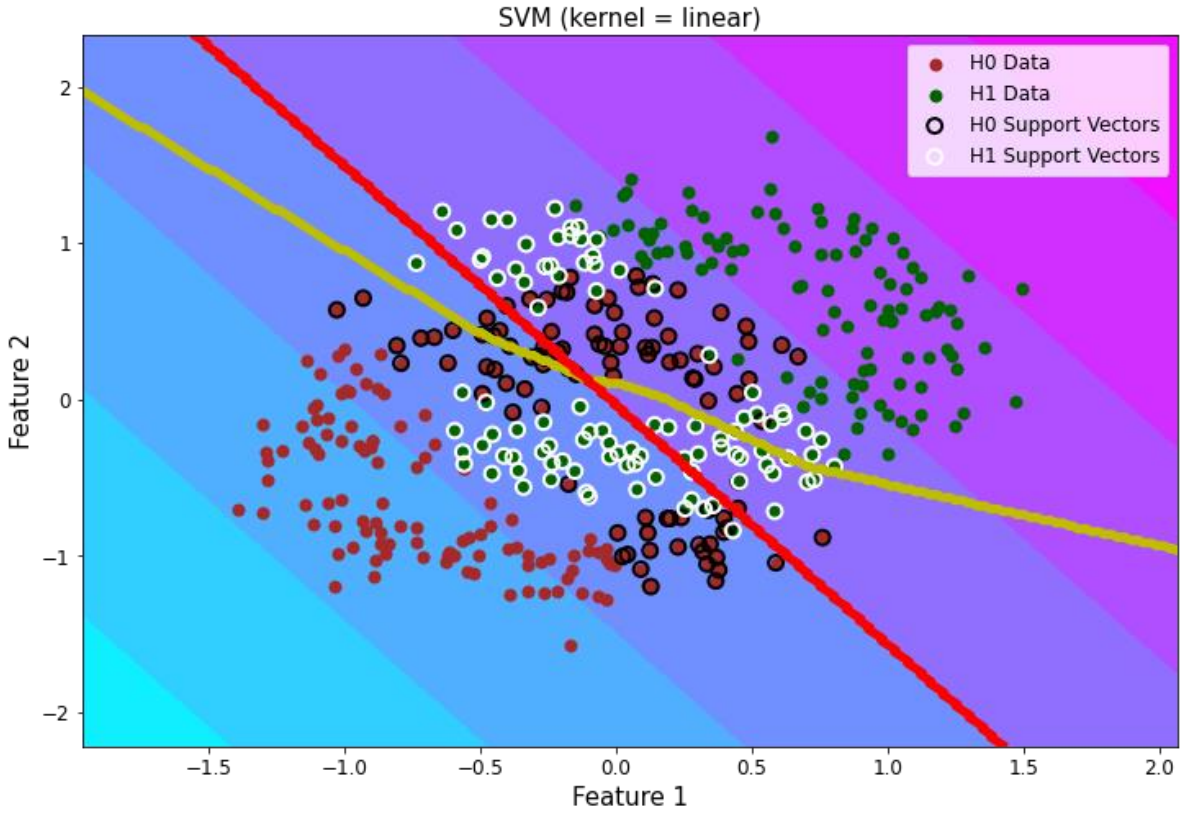
ECE580 Homework 6

Libo Zhang (lz200)

Exploring Support Vector and Relevance Vector Machines Section

Solution for Question 1 (a, b, and c):

The red line is the linear boundary produced by support vector machine (SVM) with a linear kernel (no kernel).
The yellow line is the nearly linear boundary produced by the k-nearest neighbors (KNN) classifier with k = 399.
The black circles are support vectors for class H0.
The white circles are support vectors for class H1.



Question 1 (b) – Comment on the locations and relative sparsity of the support vectors

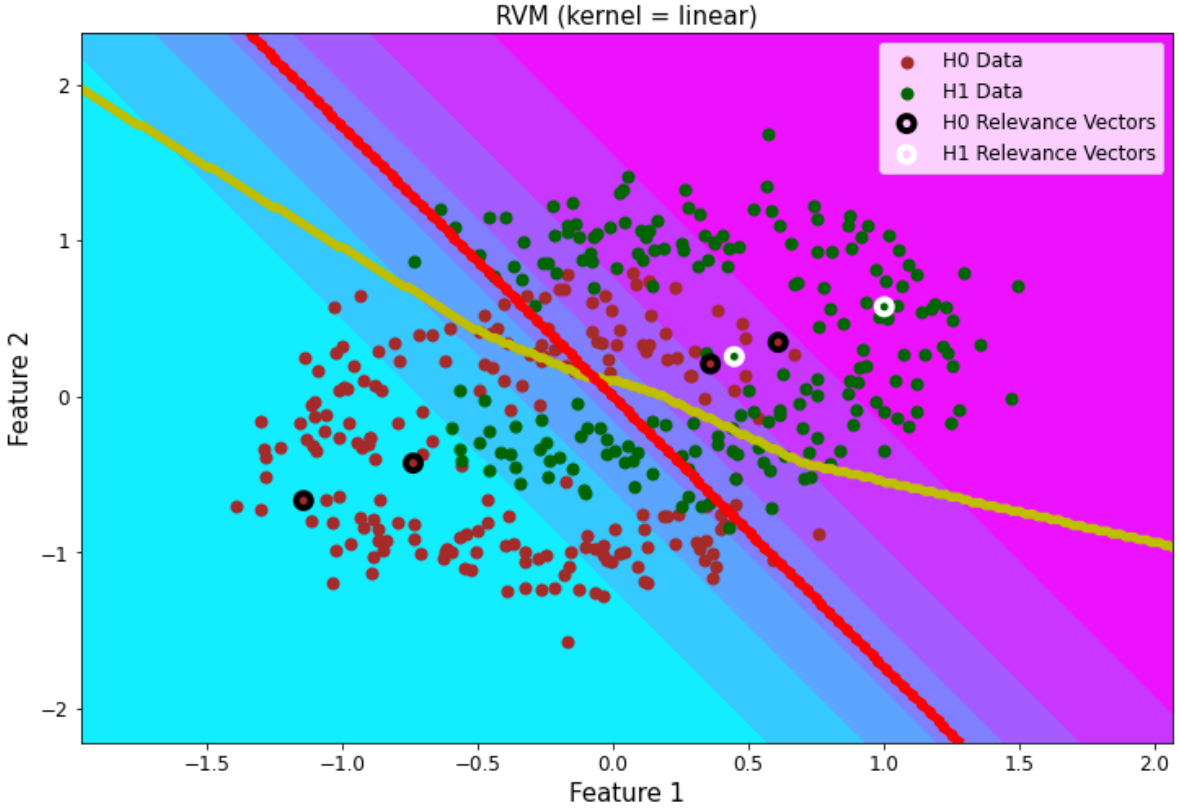
All support vectors (SVs) are close to (or located nearby) the decision boundary, so these support vectors tend to define the boundary between two classes. Both H0 support vectors and H1 support vectors are located on both sides of (or across) the decision boundary. In addition, we can see that there are a large number of support vectors for each class, so the relative sparsity of the support vectors is low.

Question 1 (c) – Comment and Answer to Which linear boundary do you find to be more "trustworthy"?

The location of the SVM linear boundary (red line) is close to the location of the KNN nearly linear boundary (yellow line), and both boundaries are located in the middle of the given two nonlinear clusters of data points. Both linear boundaries have negative slope values, and the SVM linear boundary has a relatively larger absolute scale (magnitude) slope value. I think neither linear boundary is trustworthy for these two nonlinear clusters of data points. If I have to select one, I think the KNN classifier (k = 399) is more trustworthy because it can at least classify the top-left H1 data and the bottom-right H0 data with relatively good accuracy and generality.

Solution for Question 2 (a, b, and c):

The red line is the linear boundary produced by relevance vector machine (RVM) with a linear kernel (no kernel).
The yellow line is the nearly linear boundary produced by the KNN classifier with k = 399.
The black circles are the relevance vectors for class H0.
The white circles are the relevance vectors for class H1.



Question 2 (b) – Comment on the locations and relative sparsity of the relevance vectors.

The relevance vectors (RVs) are located near the center of clusters of data points for each class, so these relevance vectors tend to define centers of mass for classes. Here we have two nonlinear clusters of data points which are entangled with each other, so we can see that 2 RVs for H0 go across the decision boundary and are located near the 2 RVs for H1. As for sparsity, there are 6 relevance vectors in total, 4 for H0 and 2 for H1. The total number of RVs is much less than the total number of SVs as we discussed in Question 1, therefore the relative sparsity of the relevance vectors is high.

Question 2 (c) – Comment and Answer to Which linear boundary do you find to be more "trustworthy"?

The location of the RVM linear boundary (red line) is close to the location of the KNN nearly linear boundary (yellow line), and both boundaries are located in the middle of the given two nonlinear clusters of data points. Both linear boundaries have negative slope values, and the RVM linear boundary has a relatively larger absolute scale (magnitude) slope value. I think neither linear boundary is trustworthy for these two nonlinear clusters of data points. If I have to select one, I think the KNN classifier (k = 399) is more trustworthy because it can at least classify the top-left H1 data and the bottom-right H0 data with relatively good accuracy and generality.

Solution for Question 3:

As discussed previously, the support vectors (SVs) are located near the decision boundary, trying to define the linear boundary between H0 and H1. However, the relevance vectors (RVs) are located near the centers of data clusters, trying to define centers of mass for different classes. Therefore, the number of RVs is less than the number of SVs, which means that the RVs are sparser than SVs.

I think the linear kernel SVM has a strength of low model complexity (simple model), but it has a weakness that there are so many SVs located on both sides of the linear decision boundary, and it is difficult to interpret the result. I think the linear kernel RVM has a strength of high sparsity of RVs, so it is easy to interpret the classification result. However, another weakness is that both linear SVM and linear RVM perform poorly when being applied to nonlinear data clusters, especially when data clusters of different classes are entangled with each other. Such entanglement will force SVs with low sparsity or even RVs with high sparsity to cross over the linear decision boundary (located on both sides), and the classification performance will be negatively affected.

Solution for Question 4:

[1]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.svm import SVC
from sklearn_rvm import EMRVC
from sklearn.neighbors import KNeighborsClassifier
```

Reference the Packages/Toolboxes

I use numpy to process arraies and matrices.
I use pandas to read and process the original datasets.
I use matplotlib to plot decision statistic surfaces.
To plot decision statistic surfaces, I also referenced the ECE580 course materials and the website below.
https://hackernoon.com/how-to-plot-a-decision-boundary-for-machine-learning-algorithms-in-python-3o1n3w07
I use sklearn to implement Support Vector Machine, Relevance Vector Machine, and KNN Classifier.

[2]:
```
filename = "dataSetHorseshoes.csv"
df_data = pd.read_csv(filename, header = None)
data = df_data.to_numpy()
X = np.copy(data[:, 1:])
y = np.copy(data[:, 0])
print(data.shape)
print(X.shape)
print(y.shape)
```

```
(400, 3)
(400, 2)
(400,)
```

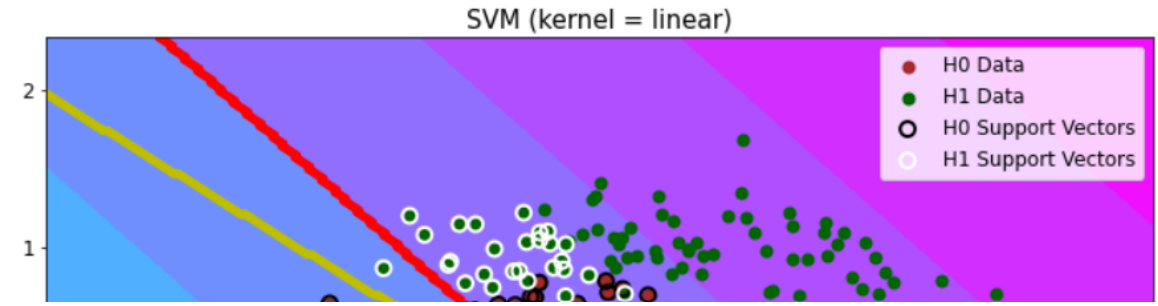
[7]:
```
def majority_vote (k) :
    floor_int = int(np.floor(float(k) / 2))
    return (1 + floor_int)
# print(majority_vote(1))
# print(majority_vote(5))
# print(majority_vote(31))
# print(majority_vote(91))
```

[8]:
```
def visualize_dss (data, model_type = "SVM", kernel_type = "linear") :
    x1max = np.max(data[:, 1])
    x1min = np.min(data[:, 1])
    x2max = np.max(data[:, 2])
    x2min = np.min(data[:, 2])
    x1Range = x1max - x1min
    x2Range = x2max - x2min
    x1 = np.linspace((x1min - (0.2 * x1Range)), (x1max + (0.2 * x1Range)), 251)
    x2 = np.linspace((x2min - (0.2 * x2Range)), (x2max + (0.2 * x2Range)), 251)
    xx, yy = np.meshgrid(x1, x2)
    r1, r2 = xx.flatten(), yy.flatten()
    r1, r2 = r1.reshape((len(r1), 1)), r2.reshape((len(r2), 1))
    grid = np.hstack((r1, r2))
    X = np.copy(data[:, 1:])
    y = np.copy(data[:, 0])

    ###################################################################
    ###################################################################
    if model_type == "SVM" :
        model = SVC(kernel = kernel_type)
        model.fit(X, y)
        vectors = model.support_
        yhat = model.decision_function(grid)
        yhat_H1 = np.copy(yhat)
        threshold = 0.0
    if model_type == "RVM" :
        model = EMRVC(kernel = kernel_type, gamma = "scale")
        model.fit(X, y)
        vectors = model.relevance_
        yhat = model.predict_proba(grid)
        yhat_H1 = np.copy(yhat[:, 1])
        threshold = 0.5

    if kernel_type == "linear" :
        knnmodel = KNeighborsClassifier(n_neighbors = 399)
        knnmodel.fit(X, y)
        major_vote = float(majority_vote(399)) / 399
    if kernel_type == "rbf" :
        knnmodel = KNeighborsClassifier(n_neighbors = 1)
        knnmodel.fit(X, y)
        major_vote = float(majority_vote(1)) / 1
    ###################################################################
```

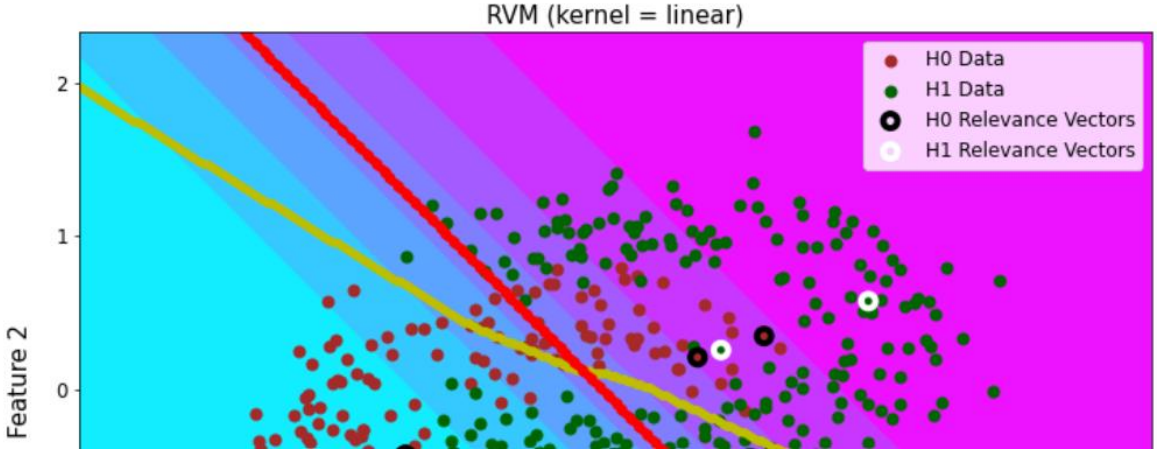
```
    ##################################################################

    yhat_knn = knnmodel.predict_proba(grid)
    yhat_H1_knn = np.copy(yhat_knn[:, 1])
    zzknn = yhat_H1_knn.reshape(xx.shape)

    zz = yhat_H1.reshape(xx.shape)
    figure, axis = plt.subplots()
    c1 = axis.contourf(xx, yy, zz, cmap = mpl.cm.cool)
    # plt.colorbar(c1)
    cknn = axis.contour(xx, yy, zzknn >= major_vote, colors = "y", linewidths = 4)
    c2 = axis.contour(xx, yy, zz >= threshold, colors = "r", linewidths = 4)
    axis.scatter(data[data[:,0]==0., 1], data[data[:,0]==0., 2], label = "H0 Data", linewidths = 2, c = "brown")
    axis.scatter(data[data[:,0]==1., 1], data[data[:,0]==1., 2], label = "H1 Data", linewidths = 2, c = "darkgreen")

    davec = np.copy(data[vectors, :])

    if model_type == "SVM" :
        axis.scatter(davec[davec[:,0]==0.,1],davec[davec[:,0]==0.,2],label="H0 Support Vectors",
                     s=80,facecolors="none",edgecolors="black",linewidths=2)
        axis.scatter(davec[davec[:,0]==1.,1],davec[davec[:,0]==1.,2],label="H1 Support Vectors",
                     s=80,facecolors="none",edgecolors="white",linewidths=2)
    if model_type == "RVM" :
        axis.scatter(davec[davec[:,0]==0.,1], davec[davec[:,0]==0.,2],label="H0 Relevance Vectors",
                     s=80,facecolors="none",edgecolors="black",linewidths=4)
        axis.scatter(davec[davec[:,0]==1.,1], davec[davec[:,0]==1.,2],label="H1 Relevance Vectors",
                     s=80,facecolors="none",edgecolors="white",linewidths=4)
```

```
    figure.set_size_inches((12, 8))
    plt.xticks(fontsize = 12)
    plt.yticks(fontsize = 12)
    plt.xlabel("Feature 1", fontsize = 15)
    plt.ylabel("Feature 2", fontsize = 15)
    plt.title(label = model_type + " (kernel = " + kernel_type + ")", fontsize = 15)
    plt.legend(loc = 1, fontsize = 12)
    plt.show()
    return None
```

[9]:
```
visualize_dss(data, model_type = "SVM", kernel_type = "linear")
```



[10]:
```
visualize_dss(data, model_type = "RVM", kernel_type = "linear")
```



Exploring Kernels in Support Vector and Relevance Vector Machines Section

Solution for Question 5 (a, b, c, and d):

Note: I indeed conduct several experiments with different radial basis function (RBF) parameters (gamma). For sklearn, the gamma parameter can be intuitively (not accurately) understood as the inverse of the RBF radius. This means that a given gamma value determines how far the influence of a single training data point can reach. If the gamma is small (the RBF radius is large), a single training data point's influence is far. If the gamma is large (the RBF radius is small), a single training data point's influence is close.

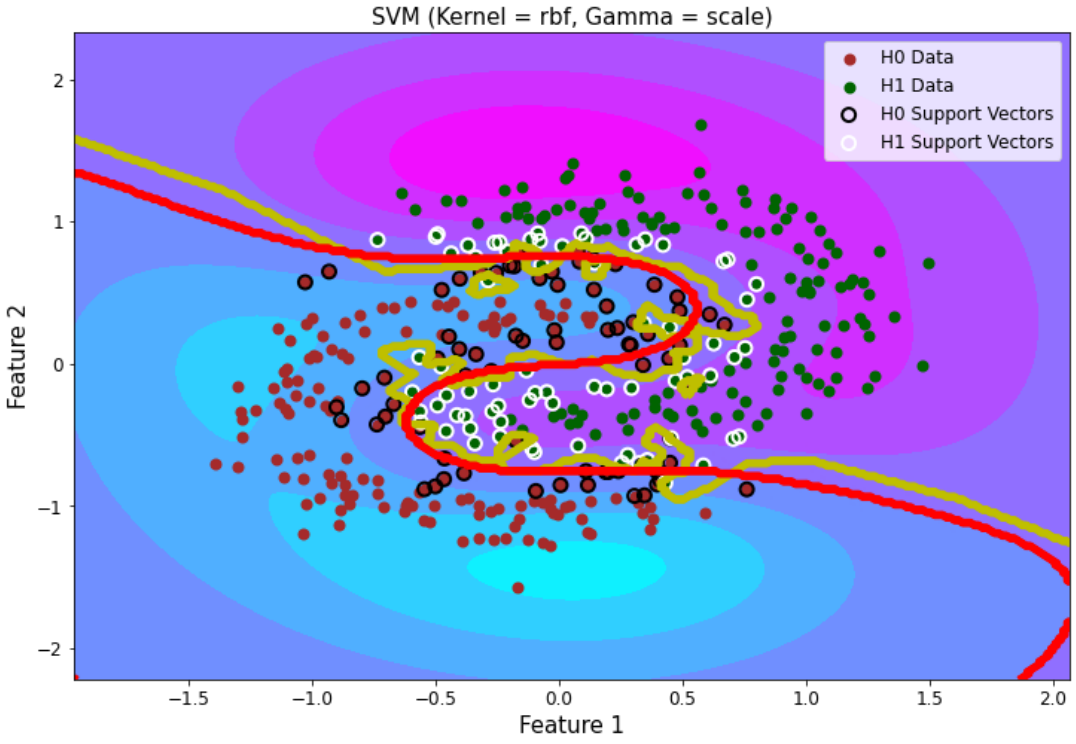
For Question 5 and 6, I provide 3 plots with gamma = "scale" = $\frac{1}{n\_features \times variance(X)} = \frac{1}{2 \times 0.462} \approx 1.083$, gamma = 0.2, and gamma = 5 respectively. **However, for Question 5, 6, 7, and 8, my comments are based on gamma = "scale" = 1.083. I will evaluate the influence of different kernel parameters (gamma, or the intuitively inverse of RBF radius) in Question 9, based on all these plots with different gamma values.**

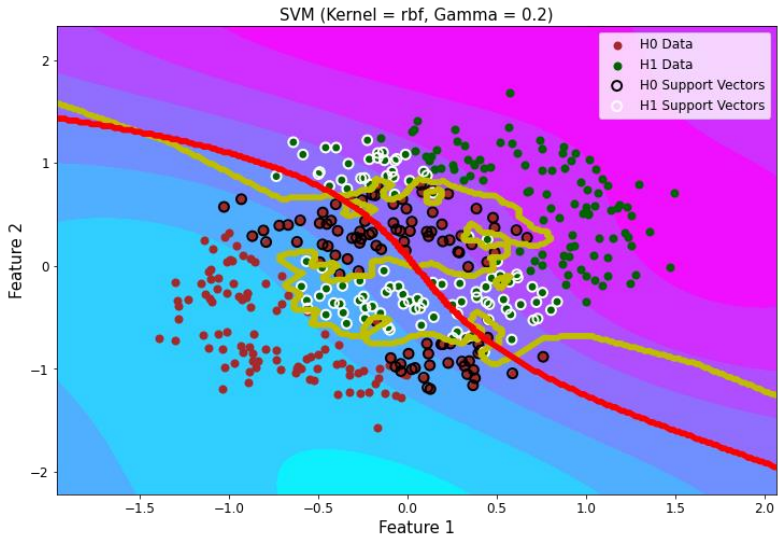
The red line is the nonlinear boundary produced by SVM with an RBF kernel.

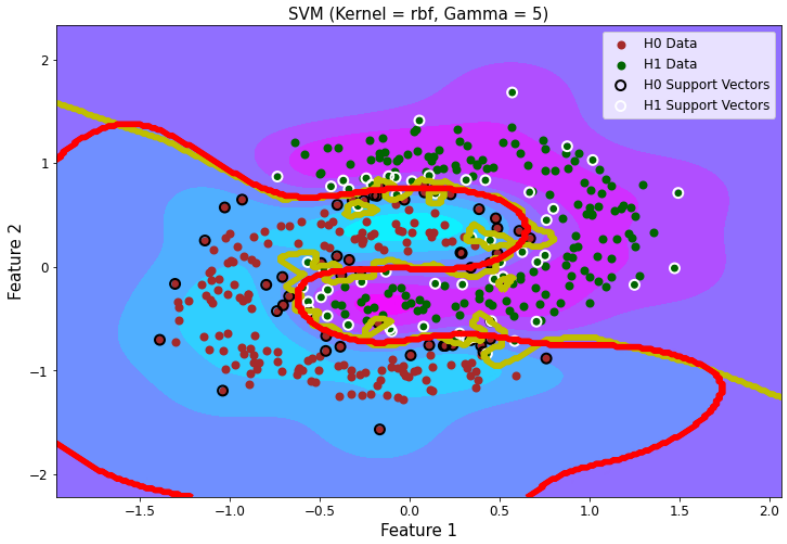
The yellow line is the nonlinear boundary produced by KNN classifier with k = 1.

The black circles are the support vectors for class H0.

The white circles are the support vectors for class H1.







Question 5 (b) – Comment and Answer to Question

All support vectors (SVs) are close to (or located nearby) the red nonlinear decision boundary, though there are still some SVs sitting on both sides, basically the majority of SVs for each class (H0 and H1) are located on one side of the boundary. There are many supports vectors for H0 and H1, so the relative sparsity of the supports vectors is low. These support vectors are characterizing the boundary between class H0 and class H1.

Question 5 (c) – Comment and Answer to Question

The shape (contours) of the decision statistic surface resembles the shape of the nonlinear clusters of data points for H0 and H1, and I think the decision statistic surface does a good job to capture how the H0 and H1 data points are nonlinearly distributed and entangled with each other. Therefore, I think the shape of the decision statistic surface makes intuitive sense.

Question 5 (d) – Comment and Answer to Question

I think the location of the red SVM nonlinear boundary is similar (close) to the location of the yellow KNN classifier (k = 1) nonlinear boundary. However, the shape of the SVM nonlinear boundary is very different from the shape of the KNN nonlinear boundary. We can see that the red SVM nonlinear boundary classifies H0 and H1 training data points in a very smooth way (the red line is very smooth), which means that the SVM with an RBF kernel has good classification accuracy and good generality for unseen data points. However, the yellow KNN nonlinear boundary classifies H0 and H1 training data points in a very winding way (the yellow line is very winding), which means that the KNN classifier with k = 1 has the overfitting problem and has bad generality for unseen data points. Therefore, I find the red SVM (RBF kernel) nonlinear boundary to be more “trustworthy”.

Solution for Question 6 (a, b, c, and d):

Note: I indeed conduct several experiments with different RBF parameters (gamma). For sklearn, the gamma parameter can be intuitively (not accurately) understood as the inverse of the RBF radius. This means that a given gamma value determines how far the influence of a single training data point can reach. If the gamma is small (the RBF radius is large), a single training data point's influence is far. If the gamma is large (the RBF radius is small), a single training data point's influence is close.

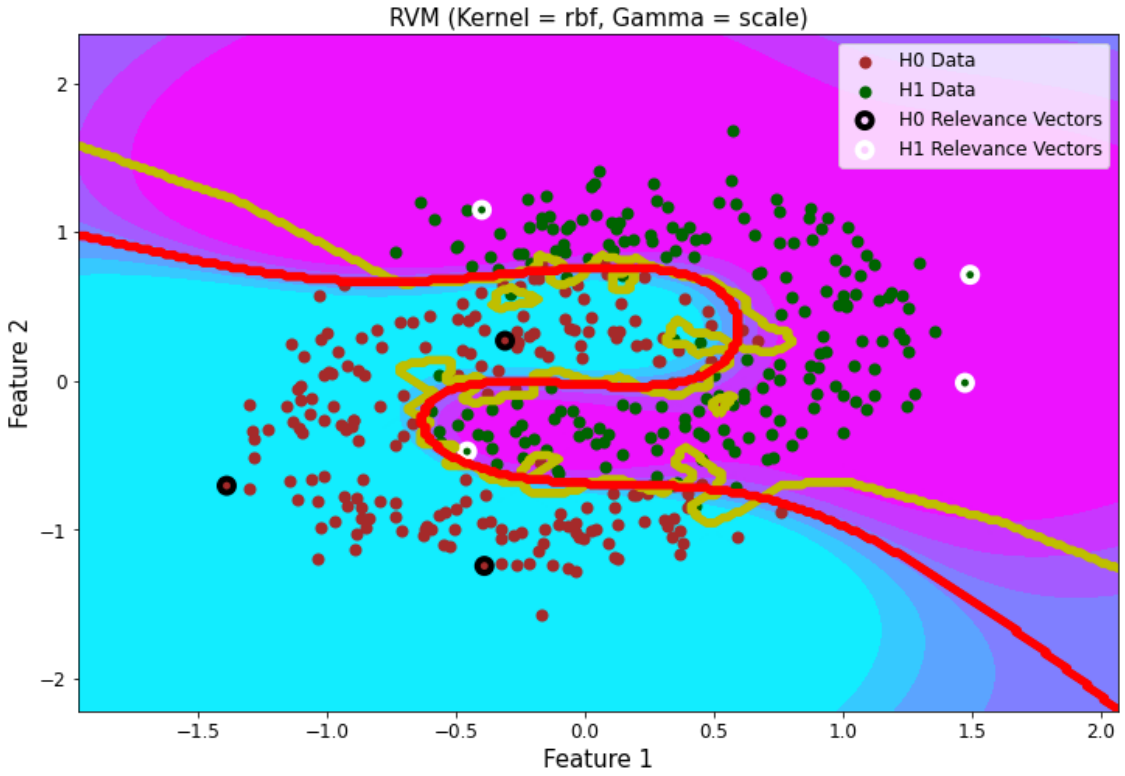
For Question 5 and 6, I provide 3 plots with gamma = "scale" = $\frac{1}{n\_features \times variance(X)} = \frac{1}{2 \times 0.462} \approx 1.083$, gamma = 0.2, and gamma = 5 respectively. **<u>However, for Question 5, 6, 7, and 8, my comments are based on gamma = "scale" = 1.083. I will evaluate the influence of different kernel parameters (gamma, or the intuitively inverse of RBF radius) in Question 9, based on all these plots with different gamma values.</u>**

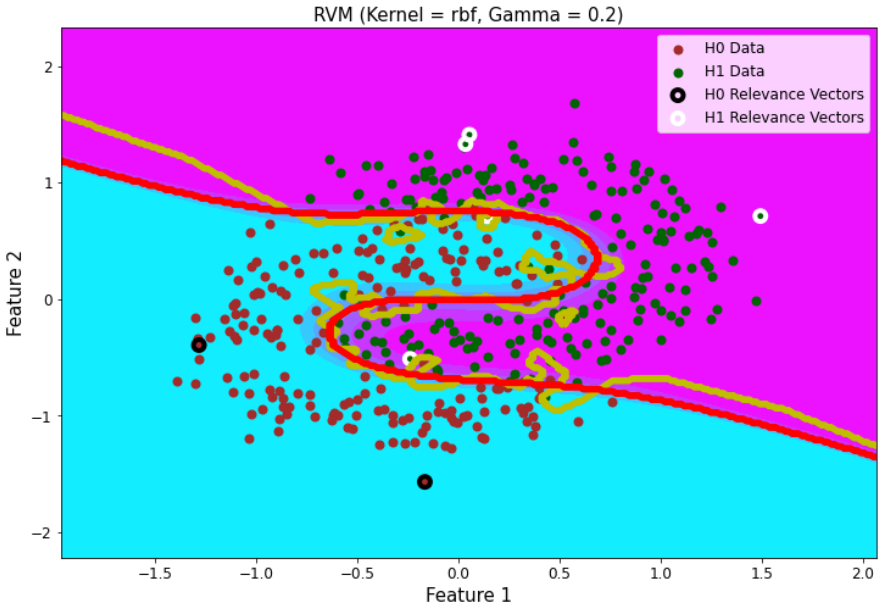
The red line is the nonlinear boundary produced by RVM with an RBF kernel.

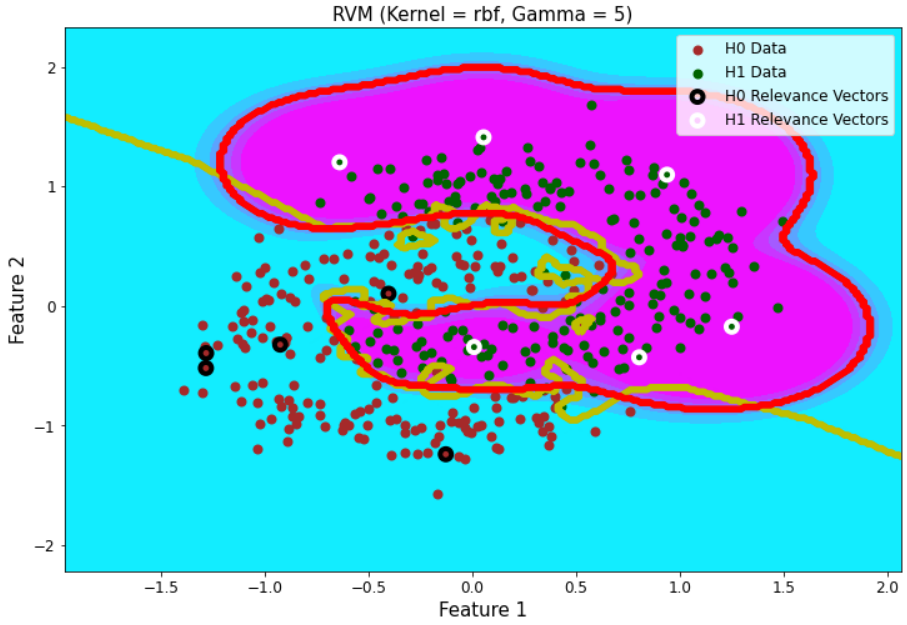
The yellow line is the nonlinear boundary produced by KNN classifier with k = 1.

The black circles are the relevance vectors for class H0.

The white circles are the relevance vectors for class H1.







Question 6 (b) – Comment and Answer to Question

The relevance vectors (RVs) are located near several centers of data clusters. There are 3 RVs for class H0 and 4 RVs for H1, therefore the relative sparsity of the relevance vectors is high. These relevance vectors are characterizing centers of mass for different classes of data points.

Question 6 (c) – Comment and Answer to Question

The shape (contours) of the decision statistic surface resembles the shape of the nonlinear clusters of data points for H0 and H1, and I think the decision statistic surface does a good job to capture how the H0 and H1 data points are nonlinearly distributed and entangled with each other. In addition, compared with SVM (RBF kernel), here the RVM (RBF kernel) provides us with a more concise decision statistic surface on both sides of the red boundary, which indicates that RVM has more confidence to classify whether a data point belongs to H0 or H1. Therefore, I think the shape of the decision statistic surface makes intuitive sense.

Question 6 (d) – Comment and Answer to Question

I think the location of the red RVM nonlinear boundary is similar (close) to the location of the yellow KNN classifier (k = 1) nonlinear boundary. However, the shape of the RVM nonlinear boundary is very different from the shape of the KNN nonlinear boundary. We can see that the red RVM nonlinear boundary classifies H0 and H1 training data points in a very smooth way (the red line is very smooth), which means that the RVM with an RBF kernel has good classification accuracy and good generality for unseen data points. However, the yellow KNN nonlinear boundary classifies H0 and H1 training data points in a very winding way (the yellow line is very winding), which means that the KNN classifier with k = 1 has the overfitting problem and has bad generality for unseen data points. Therefore, I find the red RVM (RBF kernel) nonlinear boundary to be more “trustworthy”.

Solution for Question 7:

As discussed previously, the support vectors (SVs) are located near the decision boundary, trying to define the nonlinear boundary between H0 and H1. However, the relevance vectors (RVs) are located near the centers of data clusters, trying to define centers of mass for different classes. Therefore, the number of RVs is less than the number of SVs, which means that the RVs are sparser than SVs.

Strengths – The nonlinear SVM (RBF kernel) and nonlinear RVM (RBF kernel) both can achieve good training classification accuracy and good generality for unseen data points.

Weaknesses – The optimization of support vectors for SVM (RBF kernel) might be negatively affected if the data clusters of different classes are entangled with each other. This also explains why there are many SVs for both class H0 and H1, the large number of SVs might have negative effects when determining the final nonlinear decision boundary. Given the high sparsity of relevance vectors for RVM (RBF kernel), I am concerned that when RVs are trying to define centers of mass for classes, the "exact" (or the most appropriate) centers might be deviated by some explicit outliers, and therefore imposing negative effects for RVM to determine the final nonlinear decision boundary.

Solution for Question 8:

Factor 1 – The conciseness of decision statistic surface. Whether the decision statistic surface is concise or not is important when we want to generalize our trained model and to test on unseen data set. If I want a model which has more confidence in its classification prediction, I might prefer RVM than SVM.

Factor 2 – Whether we have a large number of training data points. This is important because SVM is a convex optimization problem and RVM additionally provides a probability distribution of scores. Generally, if we have more training data points, then we might get better classification performance with RVM.

Factor 3 – The optimization concerns. This is important because during model training, RVM could get stuck at a local optimum, while SVM can converge to the global optimum (provide a unique optimal solution) all the time. If we have a small amount of training data points, then we might prefer to use SVM.

Factor 4 – Location and sparsity. The support vectors and relevance vectors have very different location and sparsity properties, as discussed previously. If we want high sparsity, we should choose RVM. If we want many vectors to be located near the decision boundary, we should select SVM.

Solution for Question 9:

When choosing a kernel, I think the most important factor is whether the training data points between different classes are linearly separable or not. If data clusters are linearly separable (two clusters are very far away from each other, for example), then I think a linear kernel (no kernel) will be the best option. If we apply a nonlinear kernel (RBF kernel, for example) to linearly separable data clusters, we might have the overfitting problem. However, if data clusters are not linearly separable, such as more than two clusters or two clusters are entangled with each other like the "Horseshoe" data set, then I think a linear kernel (no kernel) will lead to very bad classification performance. Instead, we should use nonlinear kernels such as the RBF kernel to improve classification accuracy and improve model generality for unseen test data points.

When choosing the RBF kernel parameter gamma (or the intuitively inverse of RBF radius), I think there are two important factors that I have to evaluate.

Firstly, how far away are different data clusters located? (The mean values' difference between different Gaussian distributions, for example.)

Secondly, for each data cluster, how diffuse are data points of the same class scattered? (The standard deviation or the variance value of one specific Gaussian distribution, for example.)

As the experimental results of different gamma parameters shown in Question 5 and 6, if the gamma is small (the RBF radius is large), a single training data point's influence is far, and the red decision boundary is trying to cover/divide/classify based on a larger radius range. If the gamma is large (the RBF radius is small), a single training data point's influence is close, and the red decision boundary is trying to cover/divide/classify based on a smaller radius range. Therefore, if the given training data points scatter diffusely but data clusters do not significantly entangle with each other, I think a relatively small gamma (large RBF radius) can achieve good classification accuracy and good generality for unseen test data. However, if the training data points do not scatter diffusely but data clusters entangle with each other, I think we need to choose a relatively large gamma (small RBF radius) to achieve good classification accuracy and to improve model generality for unseen test data.

Solution for Question 10:

[1]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.svm import SVC
from sklearn_rvm import EMRVC
from sklearn.neighbors import KNeighborsClassifier
```

Reference the Packages/Toolboxes

I use numpy to process arraies and matrices.
I use pandas to read and process the original datasets.
I use matplotlib to plot decision statistic surfaces.
To plot decision statistic surfaces, I also referenced the ECE580 course materials and the website below.
https://hackernoon.com/how-to-plot-a-decision-boundary-for-machine-learning-algorithms-in-python-3o1n3w07
I use sklearn to implement Support Vector Machine, Relevance Vector Machine, and KNN Classifier.

[2]:
```
filename = "dataSetHorseshoes.csv"
df_data = pd.read_csv(filename, header = None)
data = df_data.to_numpy()
X = np.copy(data[:, 1:])
y = np.copy(data[:, 0])
print(data.shape)
print(X.shape)
print(y.shape)
```

```
(400, 3)
(400, 2)
(400,)
```

[7]:
```
def majority_vote (k) :
    floor_int = int(np.floor(float(k) / 2))
    return (1 + floor_int)
# print(majority_vote(1))
# print(majority_vote(5))
# print(majority_vote(31))
# print(majority_vote(91))
```

[8]:
```
def visualize_dss (data, model_type = "SVM", kernel_type = "rbf", gamma_value = "scale") :
    x1max = np.max(data[:, 1])
    x1min = np.min(data[:, 1])
    x2max = np.max(data[:, 2])
    x2min = np.min(data[:, 2])
    x1Range = x1max - x1min
    x2Range = x2max - x2min
    x1 = np.linspace((x1min - (0.2 * x1Range)), (x1max + (0.2 * x1Range)), 251)
    x2 = np.linspace((x2min - (0.2 * x2Range)), (x2max + (0.2 * x2Range)), 251)
    xx, yy = np.meshgrid(x1, x2)
    r1, r2 = xx.flatten(), yy.flatten()
    r1, r2 = r1.reshape((len(r1), 1)), r2.reshape((len(r2), 1))
    grid = np.hstack((r1, r2))
    X = np.copy(data[:, 1:])
    y = np.copy(data[:, 0])

    ##############################################################
```

```
    ##################################################################
    if model_type == "SVM" :
        model = SVC(kernel = kernel_type, gamma = gamma_value)
        model.fit(X, y)
        vectors = model.support_
        yhat = model.decision_function(grid)
        yhat_H1 = np.copy(yhat)
        threshold = 0.0
    if model_type == "RVM" :
        model = EMRVC(kernel = kernel_type, gamma = gamma_value)
        model.fit(X, y)
        vectors = model.relevance_
        yhat = model.predict_proba(grid)
        yhat_H1 = np.copy(yhat[:, 1])
        threshold = 0.5

    if kernel_type == "linear" :
        knnmodel = KNeighborsClassifier(n_neighbors = 399)
        knnmodel.fit(X, y)
        major_vote = float(majority_vote(399)) / 399
    if kernel_type == "rbf" :
        knnmodel = KNeighborsClassifier(n_neighbors = 1)
        knnmodel.fit(X, y)
        major_vote = float(majority_vote(1)) / 1
    ##################################################################
```

```
yhat_knn = knnmodel.predict_proba(grid)
yhat_H1_knn = np.copy(yhat_knn[:, 1])
zzknn = yhat_H1_knn.reshape(xx.shape)

zz = yhat_H1.reshape(xx.shape)
figure, axis = plt.subplots()
c1 = axis.contourf(xx, yy, zz, cmap = mpl.cm.cool)
# plt.colorbar(c1)
cknn = axis.contour(xx, yy, zzknn >= major_vote, colors = "y", linewidths = 4)
c2 = axis.contour(xx, yy, zz >= threshold, colors = "r", linewidths = 4)
axis.scatter(data[data[:,0]==0., 1], data[data[:,0]==0., 2], label = "H0 Data", linewidths = 2, c = "brown")
axis.scatter(data[data[:,0]==1., 1], data[data[:,0]==1., 2], label = "H1 Data", linewidths = 2, c = "darkgreen")

davec = np.copy(data[vectors, :])

if model_type == "SVM" :
    axis.scatter(davec[davec[:,0]==0.,1],davec[davec[:,0]==0.,2],label="H0 Support Vectors",
                 s=80,facecolors="none",edgecolors="black",linewidths=2)
    axis.scatter(davec[davec[:,0]==1.,1],davec[davec[:,0]==1.,2],label="H1 Support Vectors",
                 s=80,facecolors="none",edgecolors="white",linewidths=2)
if model_type == "RVM" :
    axis.scatter(davec[davec[:,0]==0.,1], davec[davec[:,0]==0.,2],label="H0 Relevance Vectors",
                 s=80,facecolors="none",edgecolors="black",linewidths=4)
    axis.scatter(davec[davec[:,0]==1.,1], davec[davec[:,0]==1.,2],label="H1 Relevance Vectors",
                 s=80,facecolors="none",edgecolors="white",linewidths=4)
```

```
figure.set_size_inches((12, 8))
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.xlabel("Feature 1", fontsize = 15)
plt.ylabel("Feature 2", fontsize = 15)
plt.title(label = model_type + " (Kernel = " + kernel_type + ", Gamma = " + str(gamma_value) + ")", fontsize = 15)
plt.legend(loc = 1, fontsize = 12)
plt.show()
return None
```

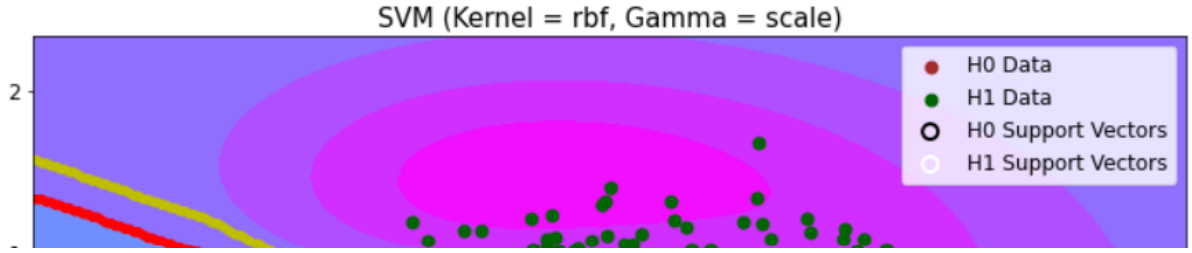
[9]:
```
# visualize_dss(data, model_type = "SVM", kernel_type = "linear")
```

[10]:
```
# visualize_dss(data, model_type = "RVM", kernel_type = "linear")
```

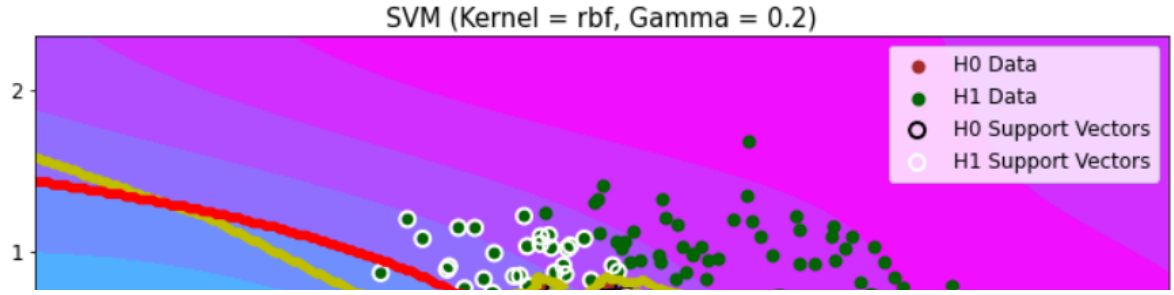
[11]:
```
print("%.3f" % np.var(X))
print("%.3f" % X.var())
print("Gamma = %.3f" % float(1 / (2 * np.var(X))))
```

```
0.462
0.462
Gamma = 1.083
```

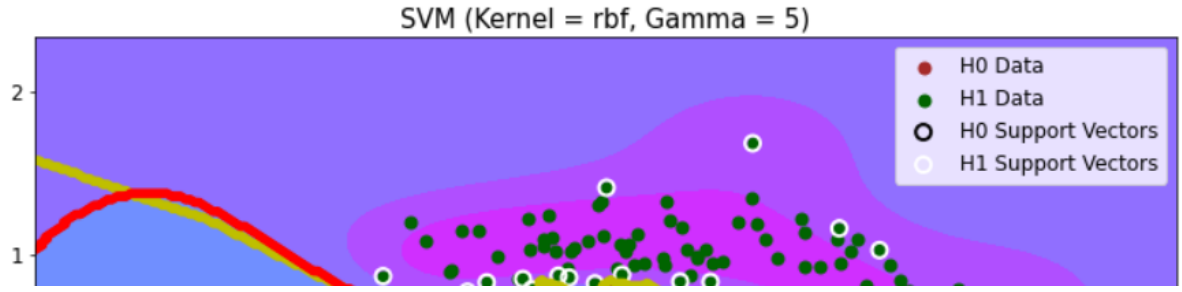
[12]:
```
visualize_dss(data, model_type = "SVM", kernel_type = "rbf", gamma_value = "scale")
```



[13]:
```
visualize_dss(data, model_type = "SVM", kernel_type = "rbf", gamma_value = 0.2)
```

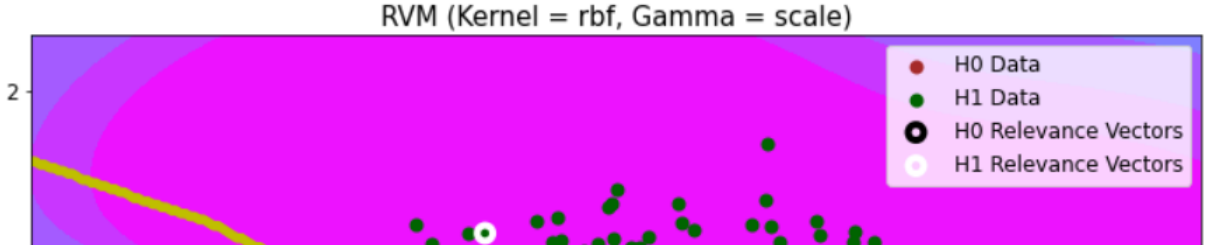


[14]:
```
visualize_dss(data, model_type = "SVM", kernel_type = "rbf", gamma_value = 5)
```

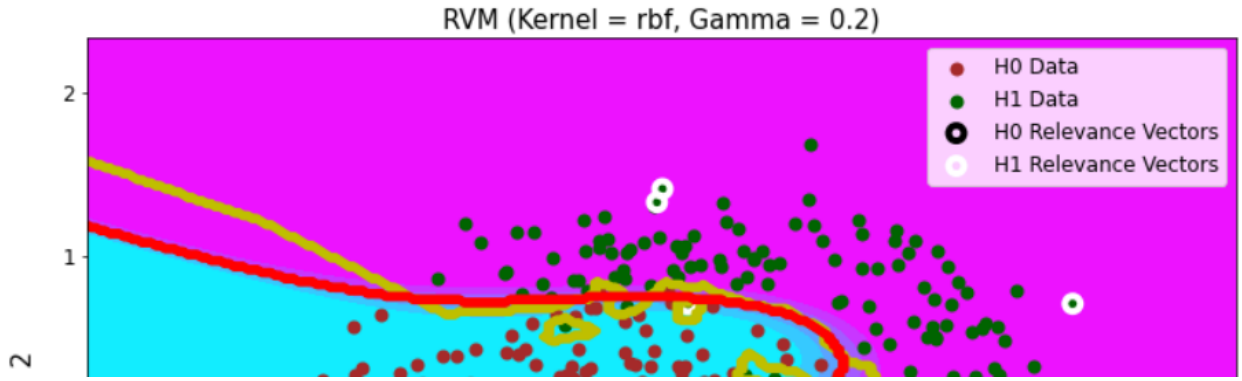


[15]:
```
##########################################################################################
```

[16]:
```
visualize_dss(data, model_type = "RVM", kernel_type = "rbf", gamma_value = "scale")
```



[17]:
```
visualize_dss(data, model_type = "RVM", kernel_type = "rbf", gamma_value = 0.2)
```



[18]:
```
visualize_dss(data, model_type = "RVM", kernel_type = "rbf", gamma_value = 5)
```