STS 4210 Data Mining II - Support Vector Machines

**What is SVM:**

​​Support vector machines (SVM) are a supervised machine learning algorithm that works for both classification and regression problems, as well as outlier detection, but is most commonly and widely used for classification tasks. The goal of this method is to create hyperplanes that can accurately separate data into its distinct classes.

A **hyperplane** is a decision boundary used to separate the two classes in SVM. The hyperplane will be one dimension less than the data, so a dataset with n features will have an n-1 dimensional hyperplane.

**Support vectors** are the points closest to the hyperplane that determine its position and orientation. The distance between support vectors and the hyperplane are maximized in order to find the optimal hyperplane.

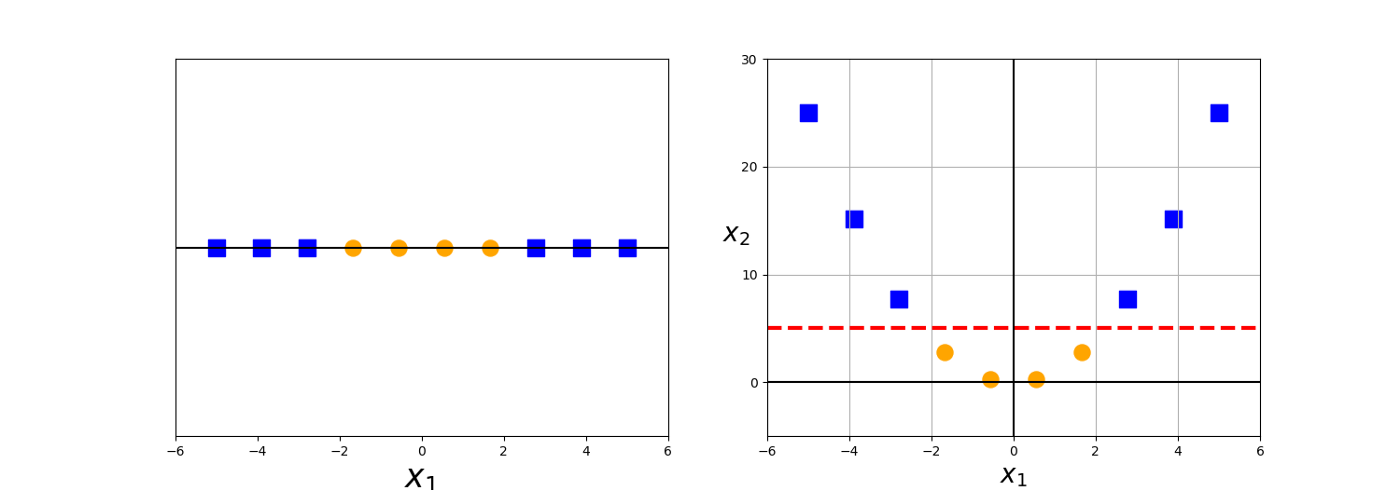
NOTE: They were originally created to address problems associated with logistic regression.

**When to use SVM:**

SVM’s are best used when a hyperplane cannot be made in the current dimensionality of the data. The SVM will increase the dimensionality in order to accurately classify the data.

Example:

The first image shows one dimensional data that is not linearly separable based on the classes. In this case, the dimensionality is increased with the function X2 = X12. By transforming the one dimensional data into a two dimensional space, the data becomes linearly separable based on the classes. It becomes difficult to visualize this concept at higher dimensions, but this simple visualization can help to understand SVM’s a bit better.



**Algorithm behind SVM**

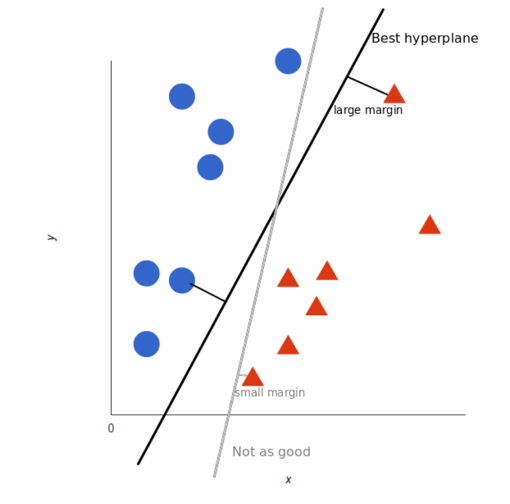
How it works:

* The algorithm creates a line (2D) or a hyperplane (3D or higher) splitting the data into classes.
* The idea is to maximize the separations between two classes through the hyperplane.

**Linear vs. Nonlinear**

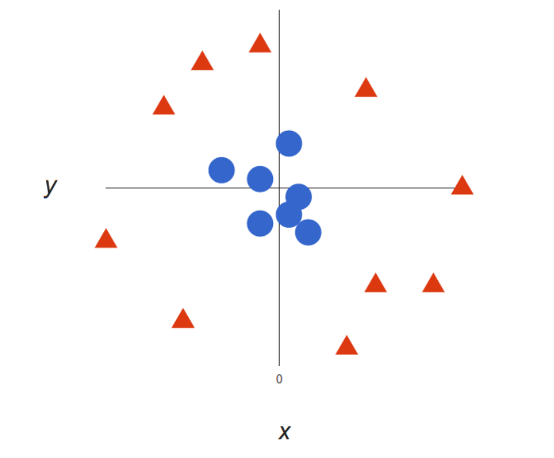
**Linear Data:**

* There are only two axes and two possibilities for data to be classified as, and the data fall linear on these axes.
* The line printed is the decision boundary such that anything on one side will be classified as a circle and anything on the other side will be classified as a triangle.
* The best hyperplane, or line in this case, is the one that maximizes the margins from both shapes – the hyperplane whose distance to the nearest element of each shape is the largest



**Non-Linear Data:**

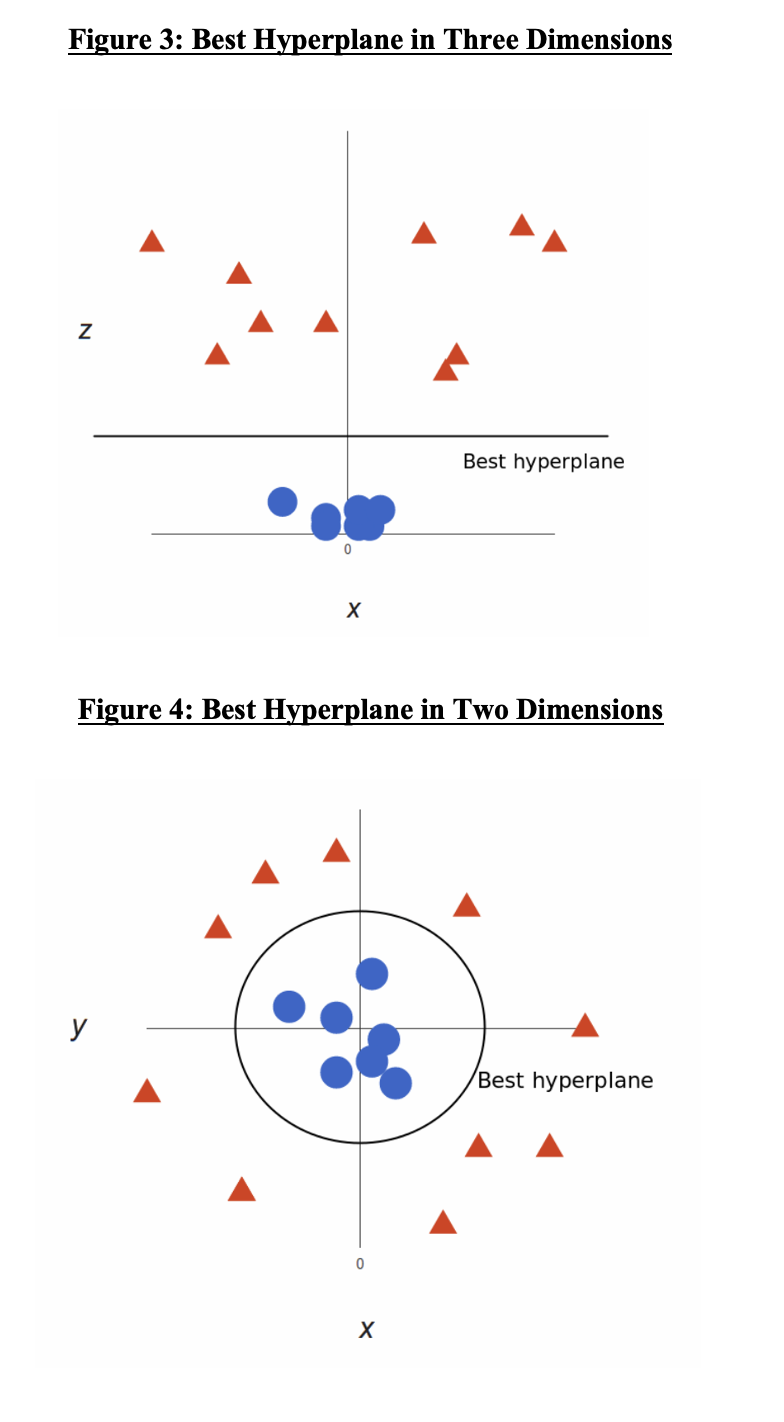
* When data is not linearly separable and we cannot draw a straight line to separate two shapes, we work with data in higher dimensions.



* This is where the support vector machines algorithm and kernel trick become useful. ​​We create a *z* dimension, so that we now have *x*, *y*, and *z* dimensions – a three-dimensional space.
* The best hyperplane becomes a circumference separating the circles and the triangles.
* There are theoretically an infinite number of planes that can perform this task, through either a shift or rotation of the original plane

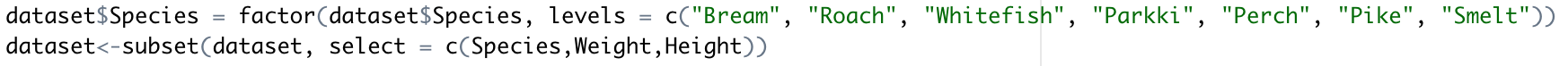
How do we decide which is best?

1. The distances from each point to the hyperplanes are compared
2. The plane that is farthest from the training observations is known as the Maximum Margin Hyperplane (where distances are maximized)
3. Finding this optimal separating hyperplane is the goal of SVM’s

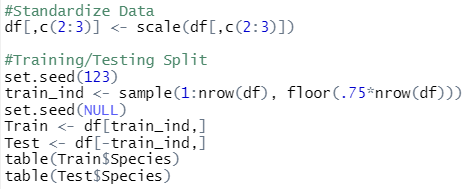


**Implementation of SVM**

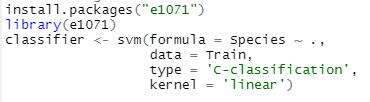
Read in the dataset, factorize the classification variable (Species), and remove all variables except for Height, Weight, and Species.



Split the data using a testing training split and scale the two numeric variables.



Performing SVM:



Collect predictions and find the accuracy:

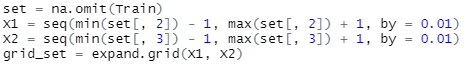


*Accuracy = 72.5%*

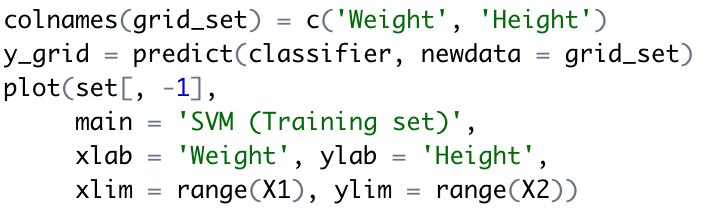
Determine the total number of support vectors



Establishing a sequence for the height and weight variables



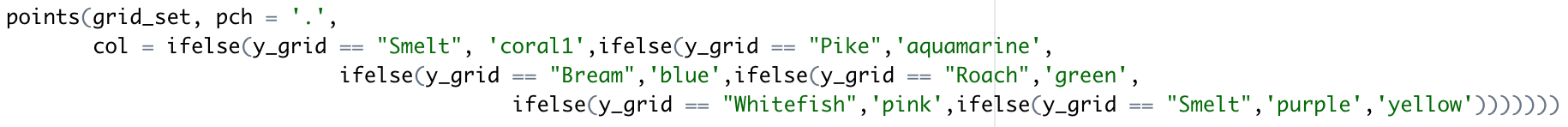
Establishing labels and creating initial plot of points



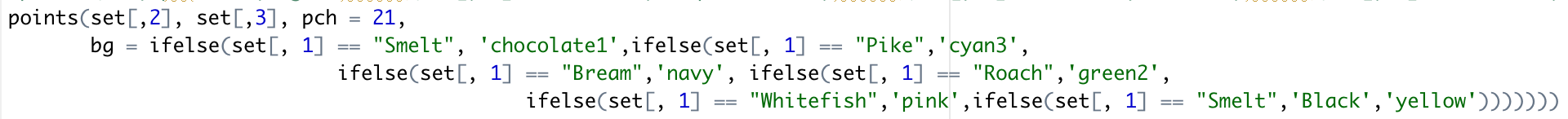
Plotting vector lines

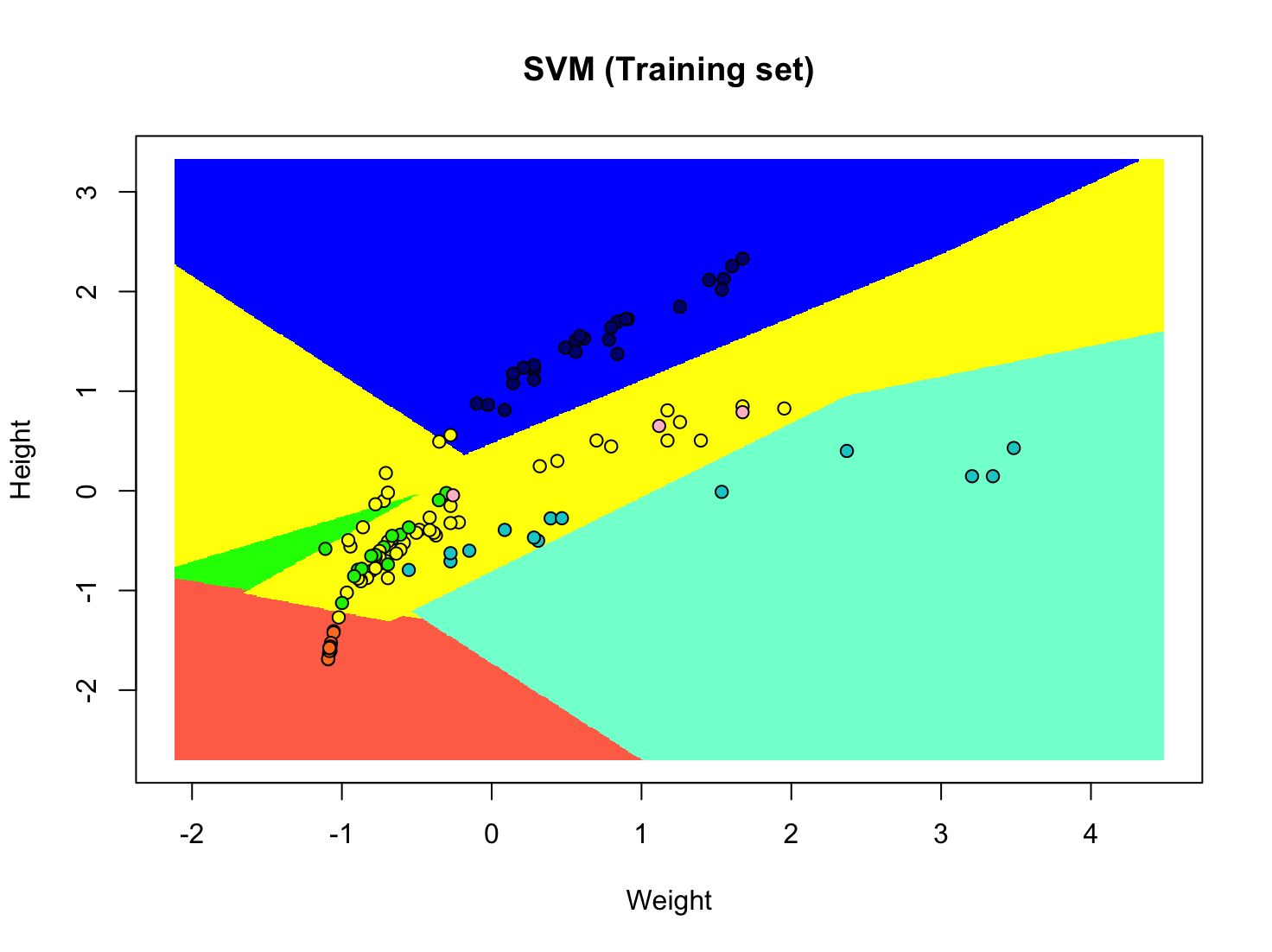


Colorizing those vector lines

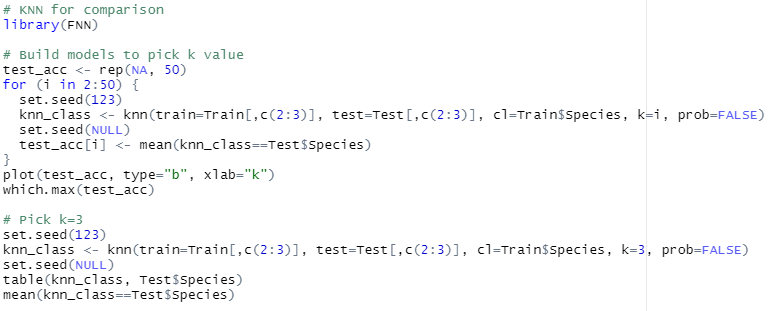


Plotting the points and colorizing them based on their classification

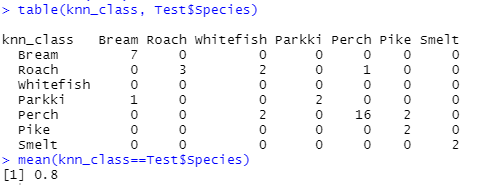




As a point of comparison, we can perform KNN on this same dataset.



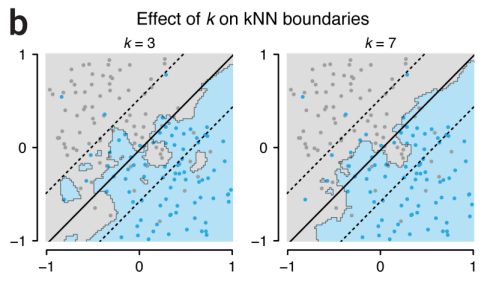
We pick k=3 because it has the greatest accuracy. This model gives us the following results.



**SVM vs. KNN**

SVM can achieve good prediction accuracy even when there are a large number of input variables. KNN, however, is more susceptible to the curse of high dimensionality. It tends to perform worse as the number of inputs increases, particularly when the variables contribute only small amounts of information.

When SVM only produces linear boundaries, it would struggle with nonlinear divisions in the data (such as the example from Figure 2) even if the sample size increased. Conversely, KNN is capable of classifying data that may have non-linear boundaries without looking at the data in higher dimensions. The disadvantage is that these nonlinear boundaries produced by a KNN model can be harder to interpret than linear SVM boundaries.



In the above image, the black lines represent a linear SVM boundary whereas the blue and grey regions depict the KNN boundary for k=3 and k=7. From this, we can see how KNN is more flexible to nonlinear boundaries than SVM without higher dimensions. However, using SVM when the data is mapped to higher dimensions allows us to also find nonlinear boundaries.

**Pros and Cons of SVM**

Advantages:

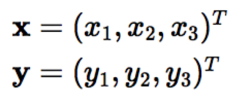
* High Dimensionality - SVM is quite effective when implemented on high dimensional data, utilizing kernel functions to handle the dimensionality
* Memory Efficiency - Only a subset of training points are used and stored in the decision process when making classification predictions, limiting the memory space used
* Versatility - Can handle linear and non-linear data

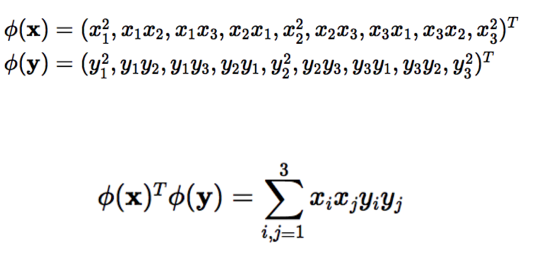
Disadvantages:

* Kernel Selection - SVM is sensitive to the kernel parameters used and can perform poorly when the number of training samples for each object is less than the number of features
* Non-Probabilistic - There is is no probabilistic interpretation for group membership because the classifier works by placing observations on either side of a hyperplane
* Does not perform well when there is noise
* Model takes a long time time to train on large datasets

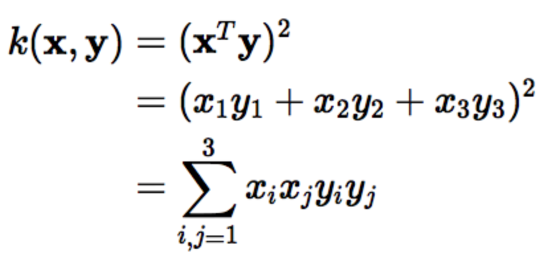
**Kernel Trick:** There are instances where data will have to be mapped to a higher dimension to find the boundary and make the classification. Computational complexity to map *x* and *y* to a scalar is easier when using the kernel function. The kernel trick involves using the dot product of the vectors to transform them in a higher dimensional space.

Without the Dot Product:





With the Dot Product:



**StatQuest Explanation** (1:55 - 5:20)

[Support Vector Machines Part 2: The Polynomial Kernel (Part 2 of 3)](https://www.youtube.com/watch?v=Toet3EiSFcM)