GGE 6322: IMAGE PROCESSING AND COMPUTER VISION

SUPPORT VECTOR MACHINE

BY VAASUDEVAN SRINIVASAN PRESENTED ON MARCH 26, 2019 09:30

1. A Gentle Introduction to Classification and its jargons 😊
2. Types of Classification (Supervised) 🤐
3. Support Vector Machines 🙂
4. Parameter Optimization 🤕
5. Code Along 😋

A GENTLE INTRODUCTION TO CLASSIFICATION AND ITS JARGONS 😊

WHAT IS CLASSIFICATION ?

Classification is the problem of identifying to which of a set of categories (sub-populations) a **new observation** belongs, on the **basis of a training set of data** containing observations (or instances) whose **category membership is known.**

CLASSIFIER WHAT?

An **algorithm** that implements classification, especially in a concrete implementation, is known as a **classifier**. The term "classifier" sometimes also refers to the **mathematical function**, implemented by a classification algorithm, that maps input data to a category.

FEATURES AND REGIONS

A crude but **functional definition** of a feature is something that can be **measured in an image**. A feature is therefore a number or a set of numbers derived from a digital image.

Features are associated with **image regions**. An object within an image has a set of measurements (features) that can be used to characterize it.

TRAINING AND TESTING

"It is **standard practice** to measure and classify a set of data to establish a normal range for the features to be used in automatic classification. This is what is referred to as **training**, and it is an essential part of building a recognition."

CLASS

One of a set of enumerated target values for a label. For example, in a binary classification model that detects spam, the two classes are **spam and not spam**. In a multi-class classification model that identifies dog breeds, the classes would be **poodle, beagle, pug**, and so on.

Classification = **Class** - ification

CLASSIFICATION MODEL

A type of machine learning model for distinguishing among two or more discrete classes.

DECISION BOUNDARY

The **separator** between classes learned by a model in a binary class or multi-class classification problems.

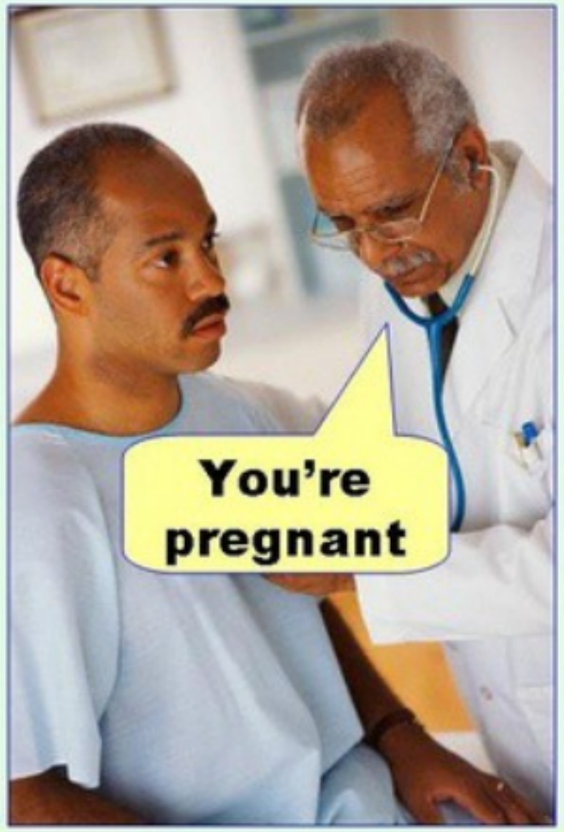
CONFUSION MATRIX

An **NxN table** that summarizes how successful a classification model's predictions were..!!

Predicted Value (predicted by the test)		Actual Value (as confirmed by experiment) positives	Actual Value (as confirmed by experiment) negatives
	positives	**TP** True Positive	**FP** False Positive
	negatives	**FN** False Negative	**TN** True Negative

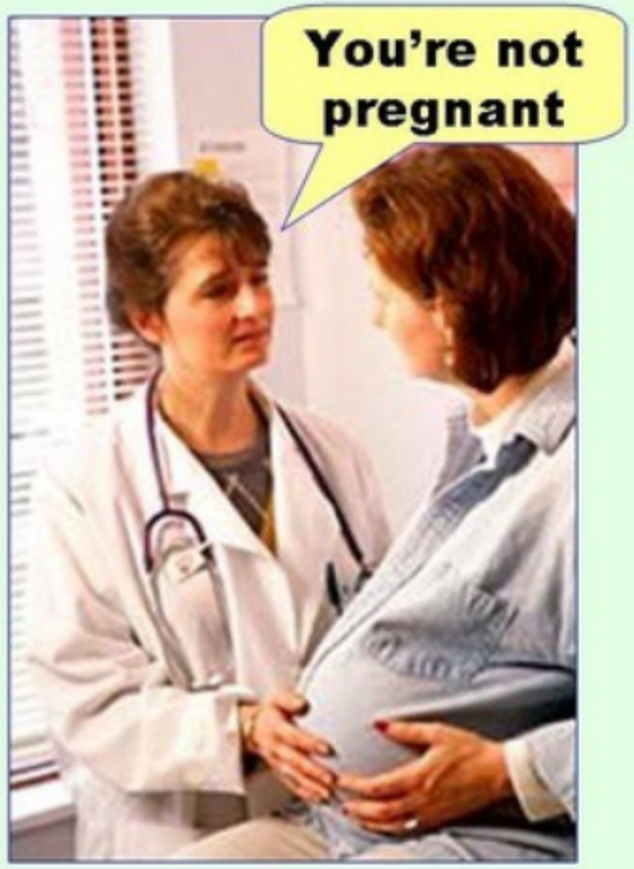
Type I error

(false positive)



Type II error

(false negative)



ACCURACY:

The fraction of predictions that a classification model got right.

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Number Of Examples}}$$

PRECISION:

Precision identifies the frequency with which a model was correct when predicting the positive class.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives+False Positives}}$$

RECALL:

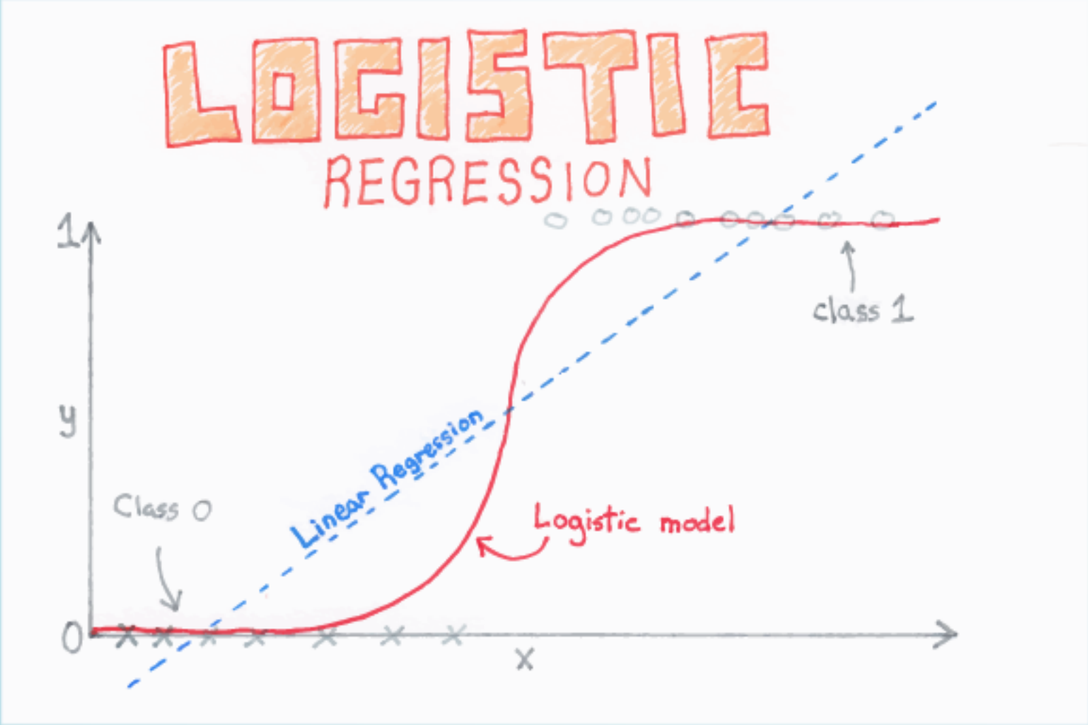
Out of all the possible positive labels, how many did the model correctly identify?

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives+False Negatives}}$$

TYPES OF SUPERVISED CLASSIFICATION 😬

1.) LOGISTIC REGRESSION

Logistic regression is kind of like linear regression but is used when the dependent variable is not a number, but something else (like a Yes/No response)



2.) K-NEAREST NEIGHBOURS (K-NN)

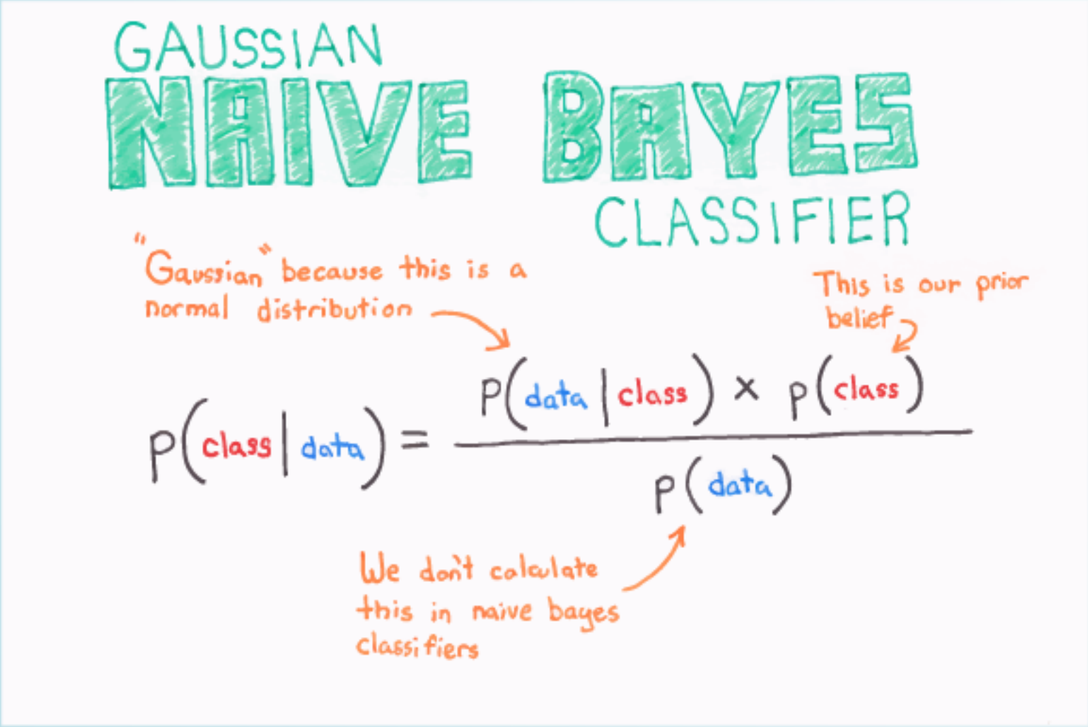
K-NN algorithm is one of the **simplest classification algorithm** and it is used to identify the data points that are separated into several classes to predict the classification of a new sample point. K-NN is a non-parametric, **lazy learning algorithm**. It classifies new cases based on a **similarity measure** (e.g. distance functions).

Some of the distance metrics that are mentioned in the book are:

- Pythagorean distance
- Manhattan distance or city block distance
- Mahanalobis distance

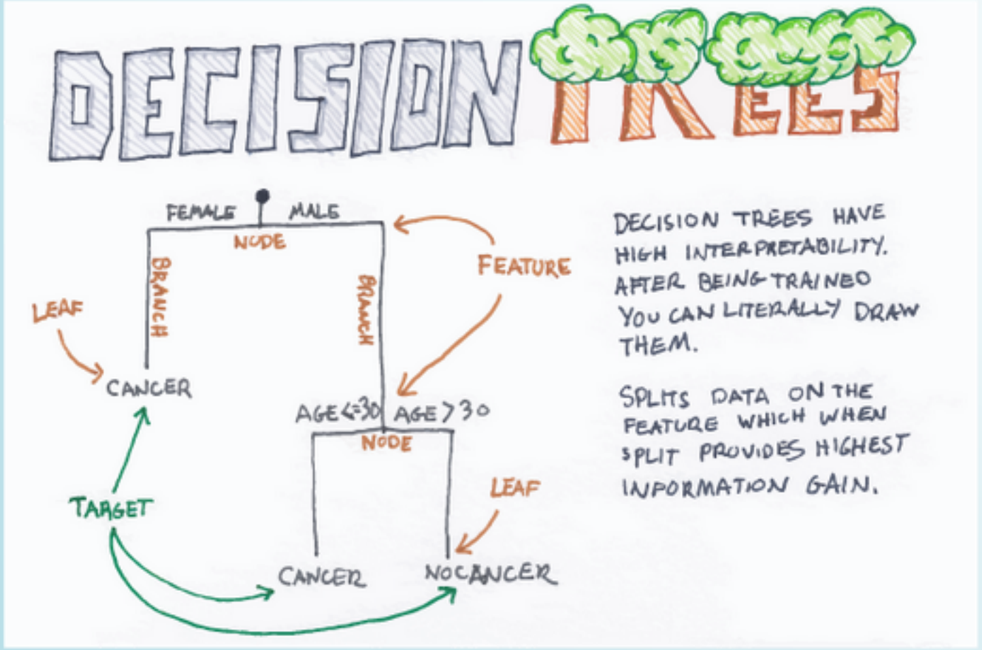
3.) NAIVE BAYES

Naive Bayes classifier is based on Bayes' theorem with the independence assumptions between predictors.



4.) DECISION TREE CLASSIFICATION

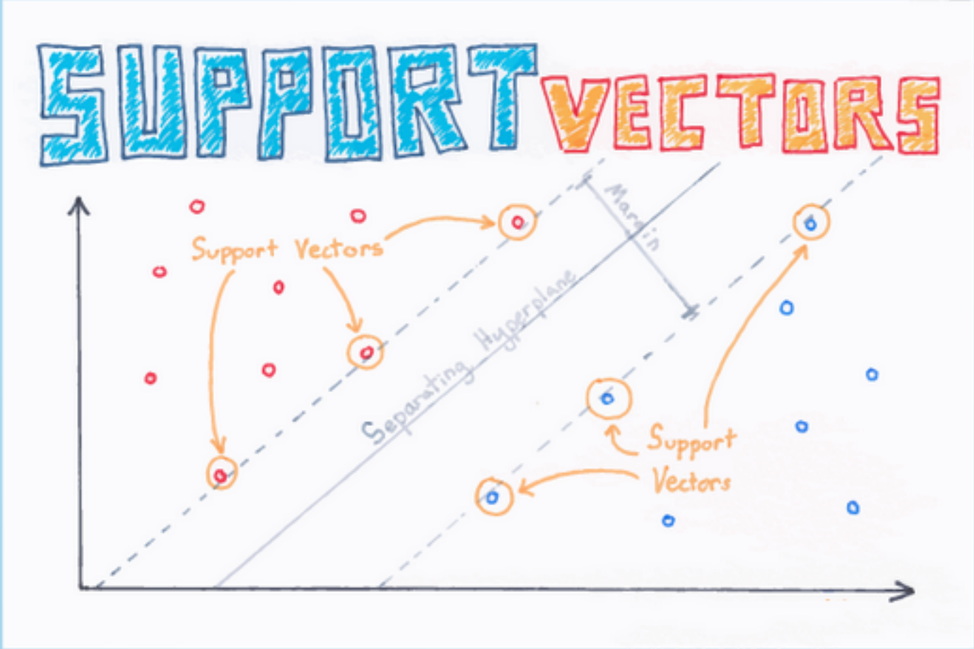
Decision tree builds **classification or regression models in the form of a tree structure**. It breaks down a dataset into **smaller and smaller subsets** while at the same time an associated decision tree is incrementally developed. The final result is a **tree with decision nodes and leaf nodes**.

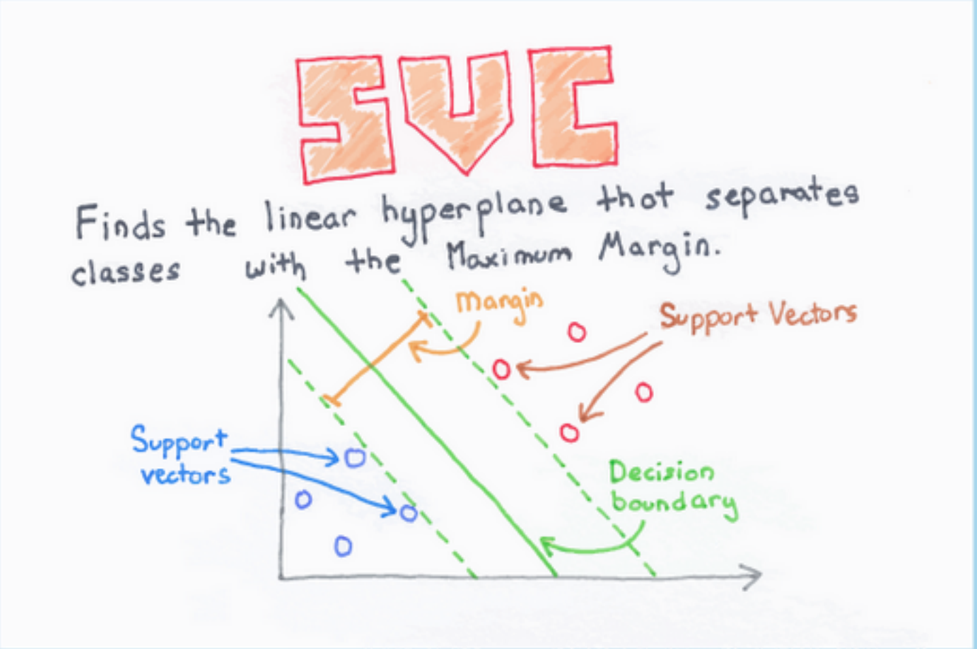


SUPPORT VECTOR MACHINES 🙂

WHAT IS SVM ?

Support Vector is used for **both regression and Classification**. It is based on the concept of decision planes that define decision boundaries. A decision plane(hyperplane) is one that separates between a set of objects having different class memberships.

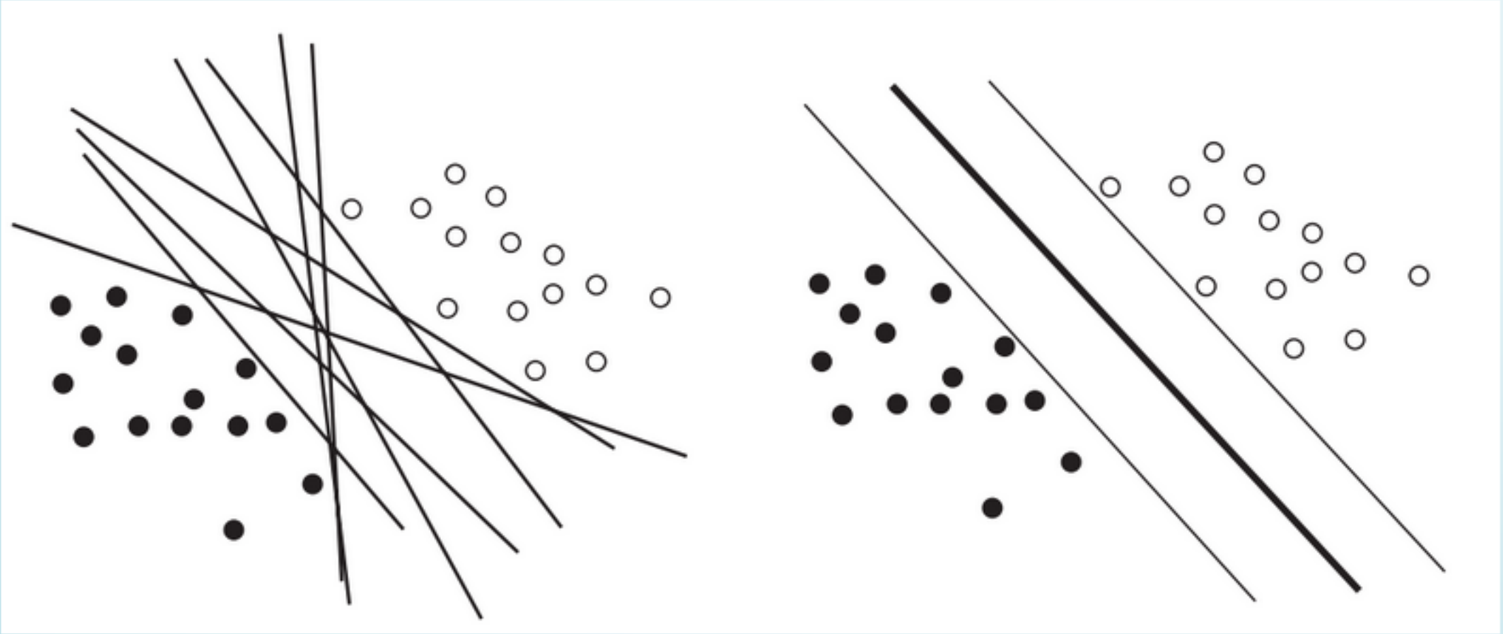




HYPER-PLANE ?

It performs classification by finding the **hyperplane** that maximizes the margin between the two classes with the help of support vectors. It is a linear function that divides **N-dimensional data** into two parts.

A hyperplane in an n-dimensional Euclidean space is a flat, n-1 dimensional subset of that space that divides the space into two disconnected parts.

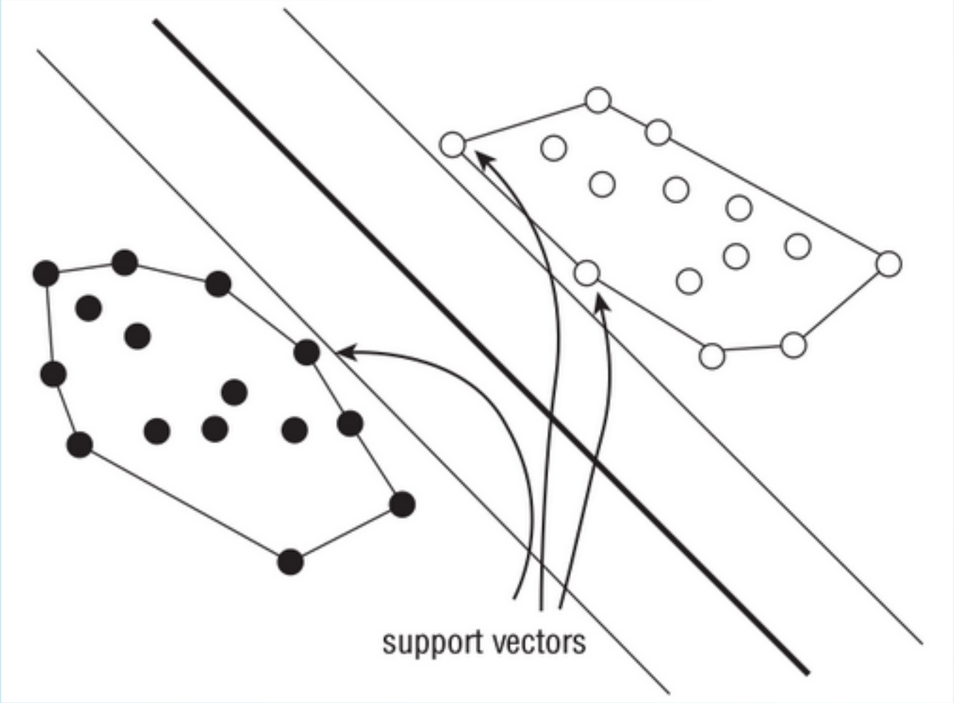


CONVEX HULL AND SUPPORT VECTORS

The basic idea, though, is to use feature vectors on the convex hull of the data sets as candidates to be used to guide the optimization.

The candidates are called support vectors and are illustrated, along with the convex hulls for the data sets.

The support vectors completely define the maximal margin line, which is the line that passes as **far as possible from all three of those vectors**. There can be more than three support vectors, but not fewer.

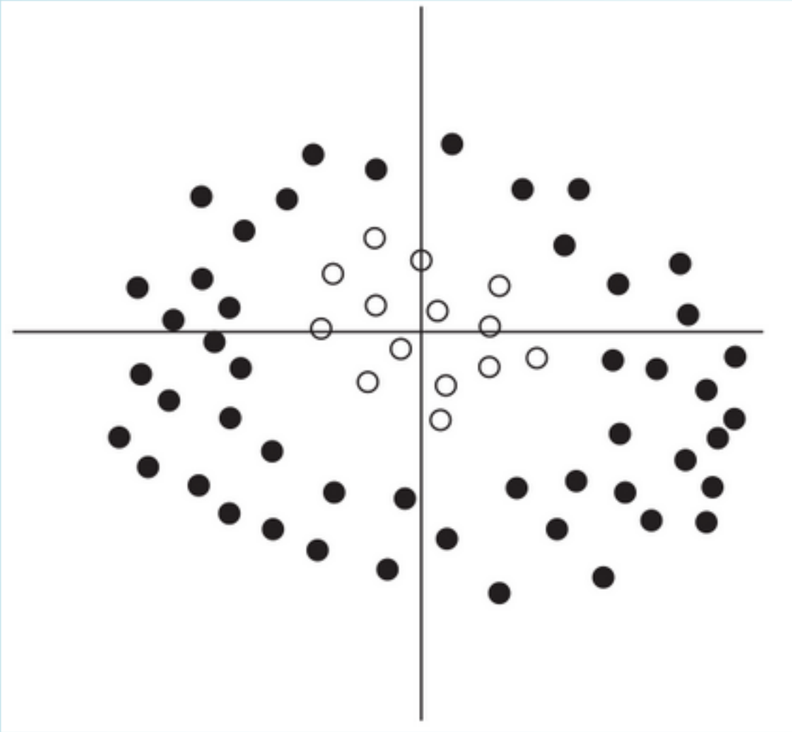


NON-LINEAR HYPERPLANE ?

Support vector machines can also find non-linear boundaries between classes, which is their another major advantage over other classification methods.

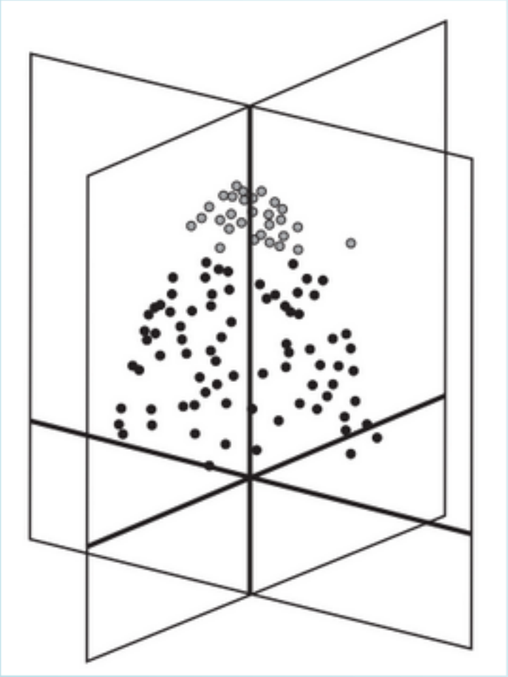
It is accomplished by **transforming** those feature vectors so that a linear boundary can be found.

Consider the below example.



TRANSFORMATION

- Add a dimension and transform the points appropriately into a third dimension. Voila..!!

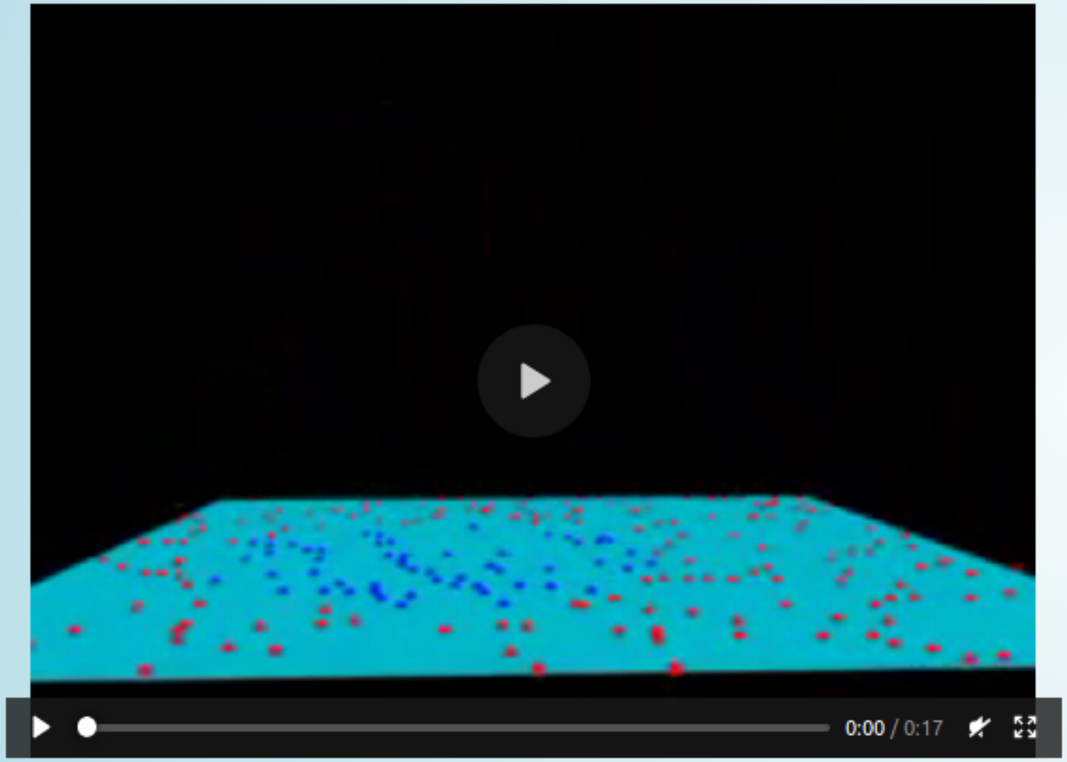


KERNELS

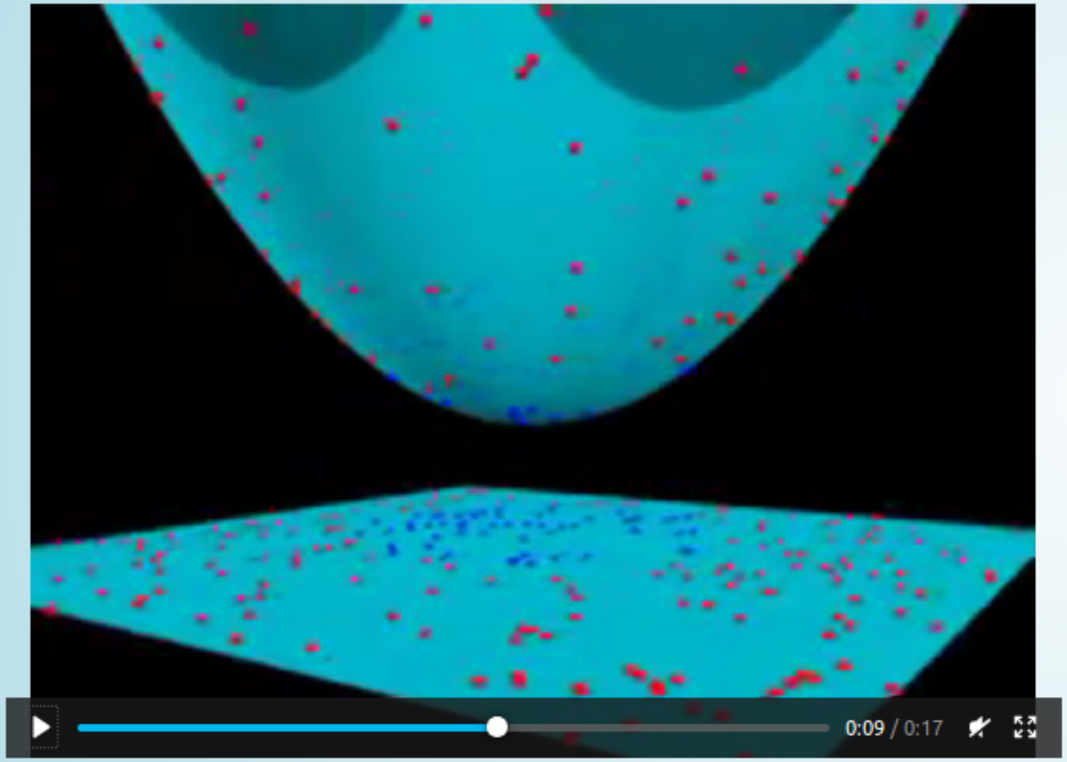
In SVM parlance, any given transformation uses a **kernel**, which is the function that projects the data from one dimension into other dimension.

$$K(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} \mathbf{X}_i \cdot \mathbf{X}_j & \text{Linear} \\ (\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C)^d & \text{Polynomial} \\ \exp(-\gamma |\mathbf{X}_i - \mathbf{X}_j|^2) & \text{RBF} \\ \tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C) & \text{Sigmoid} \end{cases}$$

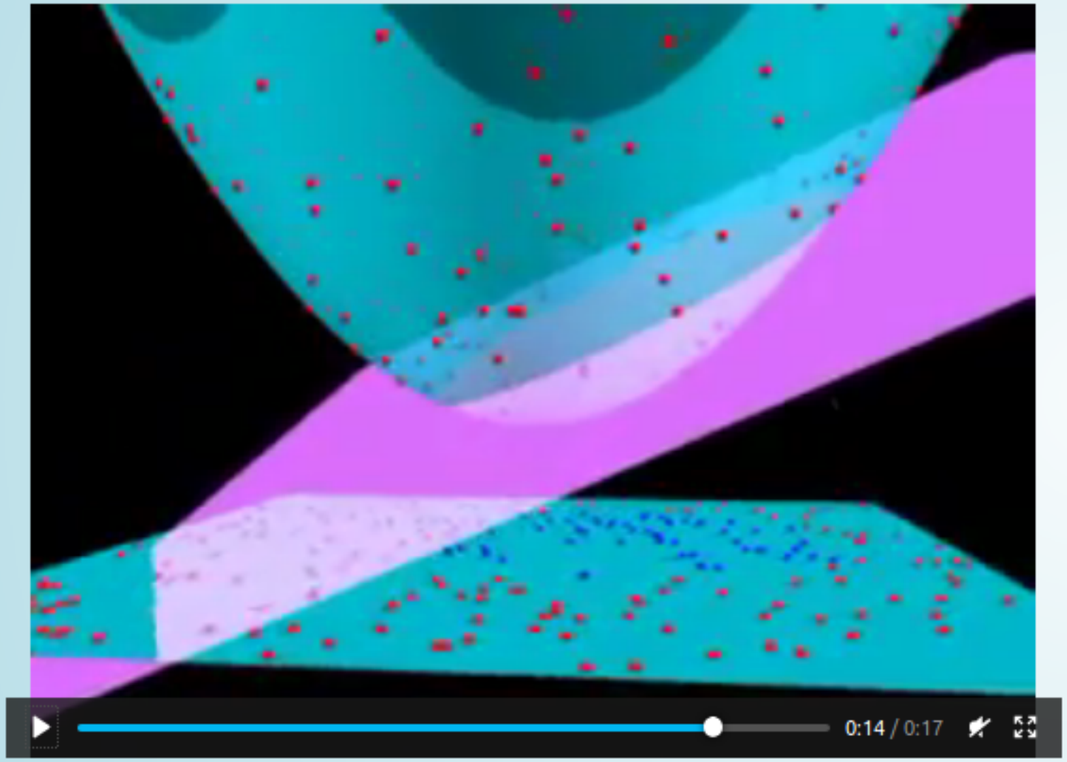
Source: http://www.youtube.com/watch?v=3liCbRZPrZA



Source: http://www.youtube.com/watch?v=3liCbRZPrZA



Source: http://www.youtube.com/watch?v=3liCbRZPrZA



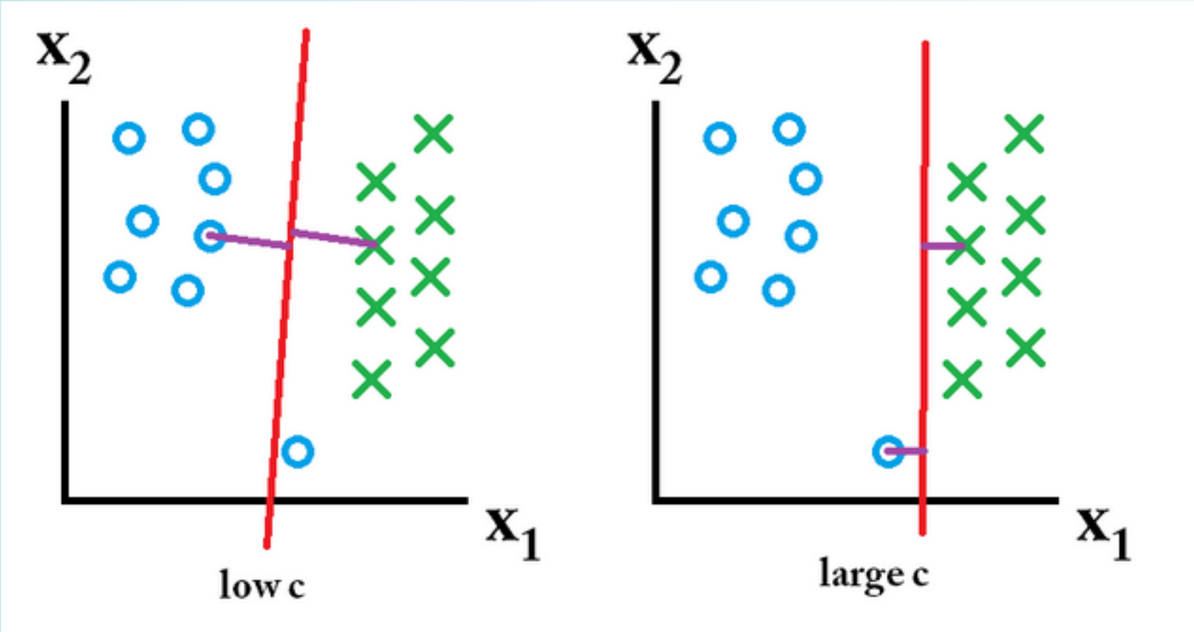
PARAMTER OPTIMISATION 🤕

SVM C PARAMETER

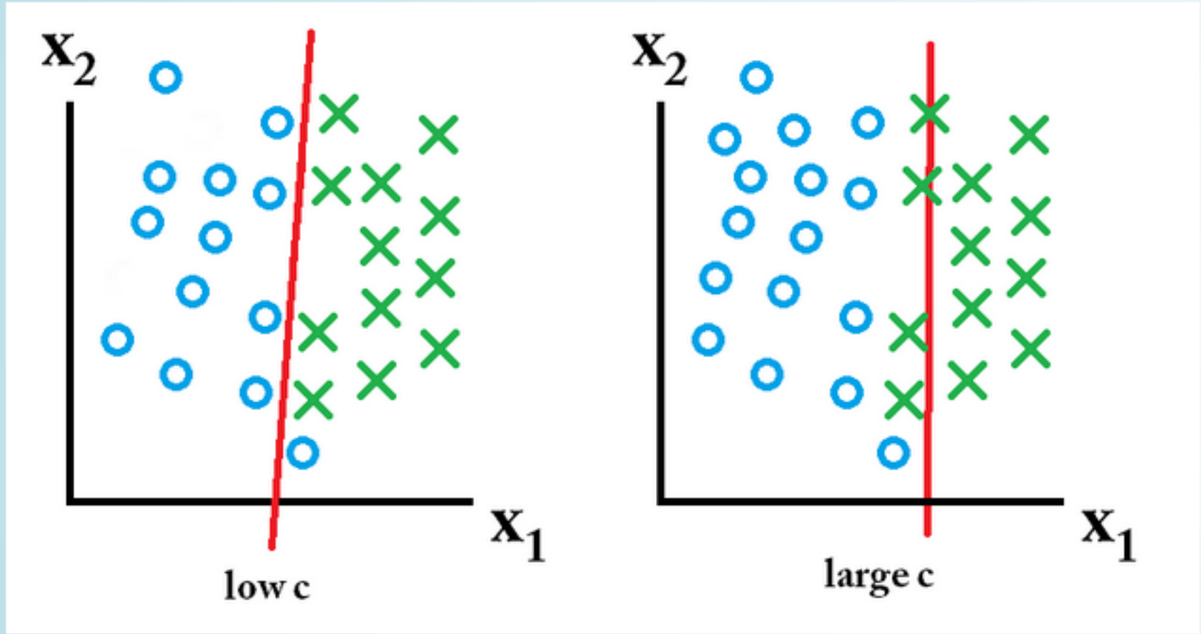
C Parameter controls the power between **Smooth Decision Boundary** and **Classifying points correctly**.

- For large values of C, the optimization will choose a smaller-margin hyperplane.
- A very small value of C will cause the optimizer to look for a larger-margin separating hyperplane.

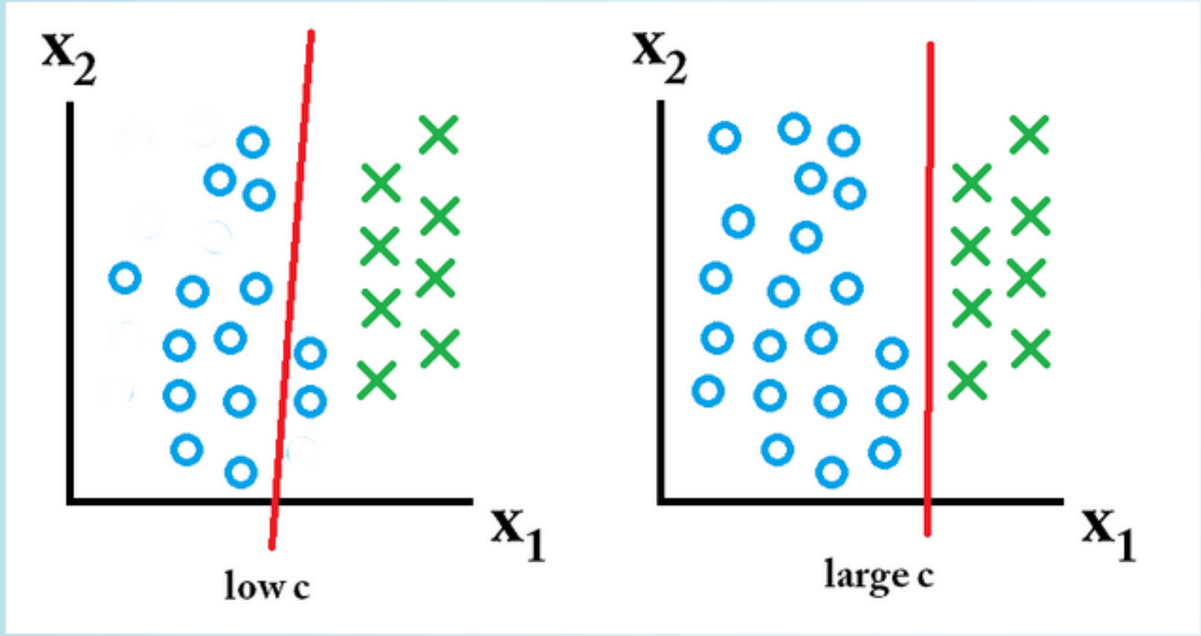
WHICH ONE IS THE BEST ?



LOW C IS BEST



WAIT...!!! LARGE C IS BEST.



HENCE THE TRADE-OFF BUT HOW TO FIND C ?

The usual way to adjust the C parameter is by a **Grid search**.

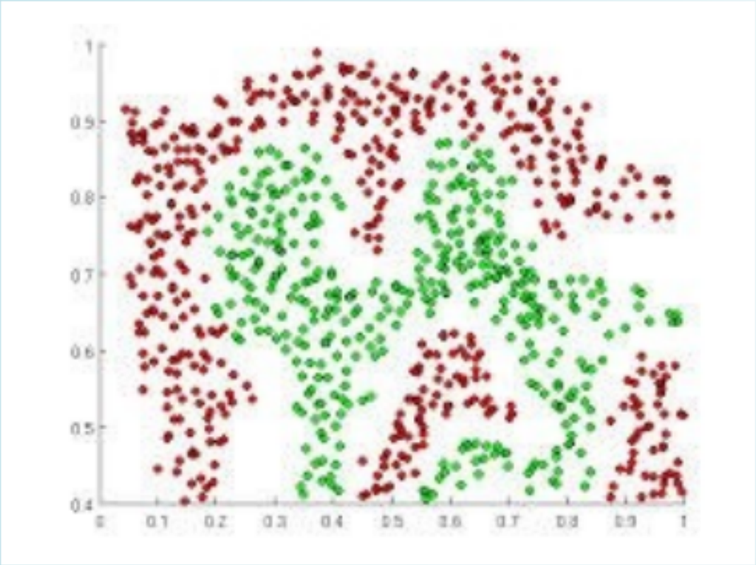
Typical Steps:

- Set a range of feasible values for C, for instance C in [0,15]
- Look for the average error using a 5 or 10 fold cross validation using the training set and keep the best value
- Perform the same procedure but on a finer search

SVM GAMMA PARAMETER

It defines how far the influence of a single training example reaches. If it has a low value it means that every point has a far reach and conversely high value of gamma means that every point has close reach.

Consider the below image:



- Imagine "raising" the green points, then we can separate them from the red points with a plane (hyperplane)
- A small gamma gives us a pointed bump in the higher dimensions, a large gamma gives us a softer, broader bump.

Note: To "raise" the points, we must use the RBF (Radial Bias function) Kernel..!!

CODE ALONG 😍😋

In [1]:

```
# Importing the modules
import matplotlib.pyplot as plt
import pandas as pd
import cowsay

cowsay.dragon("Modules are imported successfully..!!")
```

```
 _____________________________________
< Modules are imported successfully..!! >
 =====================================
                                        \
                                         \
                                          \
                                           \
                                                        / \  //\
                                                 |\___/|      /   \//  \\
                                                 /0  0  \__  /    //  | \ \
                                                /     /  \/_/    //   |  \  \
                                                \@_^_\@'/   \/_   //    |   \   \
                                                //_^_/     \/_ //     |    \    \
                                             ( //) |          \///      |     \     \
                                           ( / /) _|_ /   )  //       |      \     _\
                                         ( // /) '/,_ _ _/  ( ; -.    |    _ _\.-~        .-~~~~^-.
                                       (( / / )) ,-{        _      `-.|.-~-.           .~         `.
                                      (( // / ))  '/\      /                 ~-. _ .-~      .-~^-.  \
                                      (( /// ))      `.   {            }                   /      \  \
                                       (( / ))     .----~-.\        \-'                 .~         \  `. \^-.
                                                  ///.----..>        \             _ -~             `.  ^-`  ^-_
                                                    ///-._ _ _ _ _ _ _}^ - - - - ~                     ~-- ,.-~
                                                                                                       /.-~
```

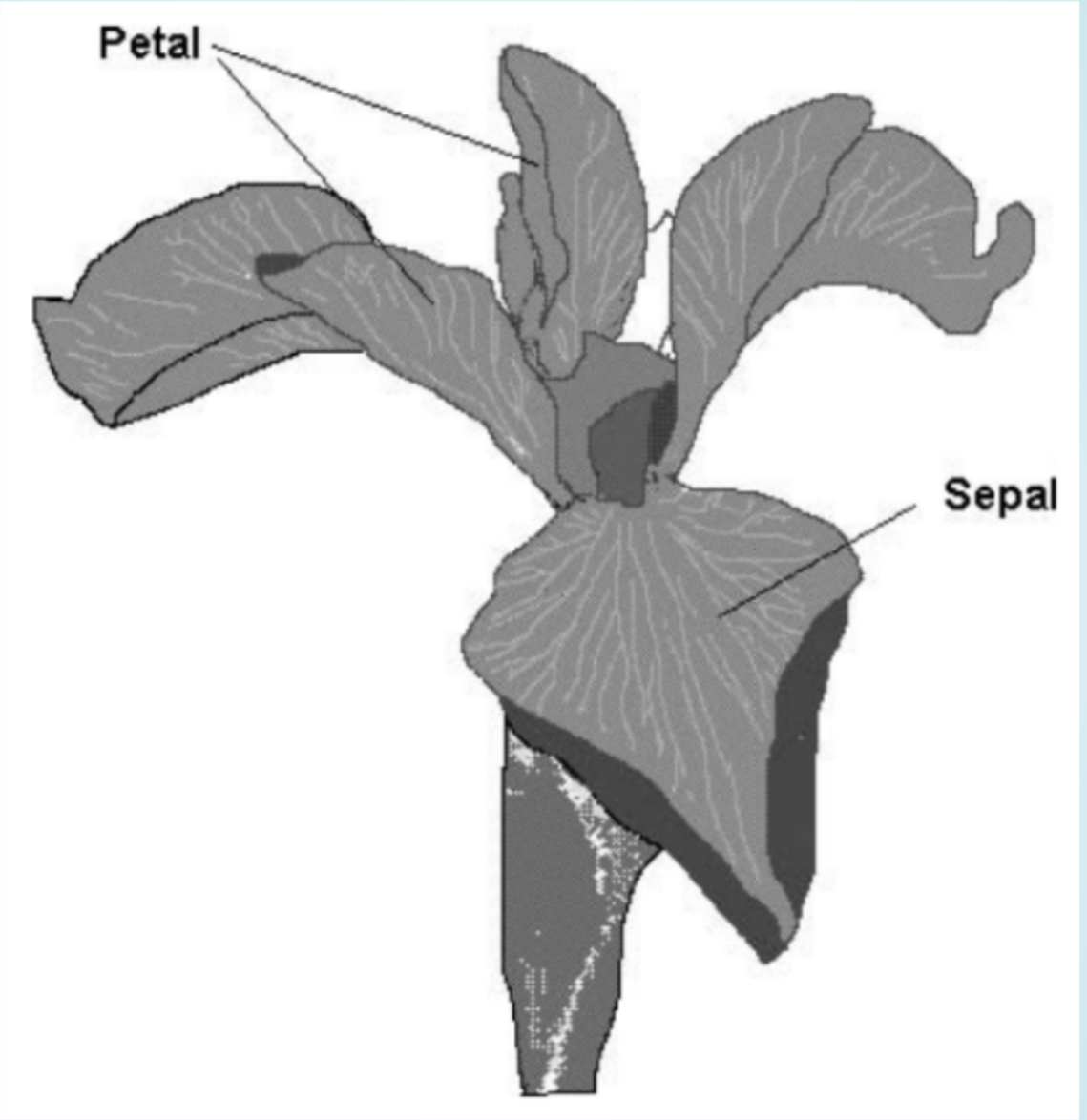
In [2]:

```
# Iris Dataset
cols = ["SLength", "SWidth", "PLength", "PWidth", "Class"]
types = ["Setosa", "Versicolor", "Virginica"]
iris = pd.read_csv("Chap8_Datas_Code/iris-data.txt", sep="\t", names=cols)

iris.head()
```

Out[2]:

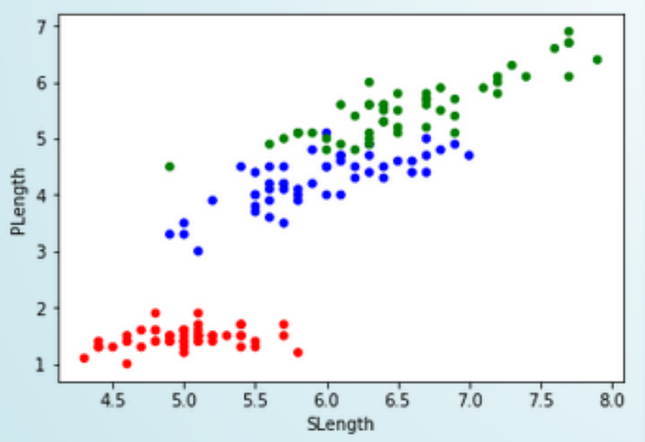
	SLength	SWidth	PLength	PWidth	Class
0	5.1	3.5	1.4	0.2	1
1	4.9	3.0	1.4	0.2	1
2	4.7	3.2	1.3	0.2	1
3	4.6	3.1	1.5	0.2	1
4	5.0	3.6	1.4	0.2	1

SETOSA	VERISCOLOR	VIRGINICA
		


Petal
Sepal

In [3]:
```
# Plotting
colors = [{1:'red', 2:'blue', 3:'green'}[i] for i in iris.Class]
iris.plot.scatter(x='SLength', y='PLength', c=colors) # Plotting is done through Pandas
```

Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66ae8d0eb8>
```



SCIKIT-LEARN TO THE RESCUE

In [4]:

```
# Scikit-learn
from sklearn import svm, model_selection as ms
from sklearn.metrics import *

# For Visualisation
from mlxtend.plotting import plot_decision_regions
```

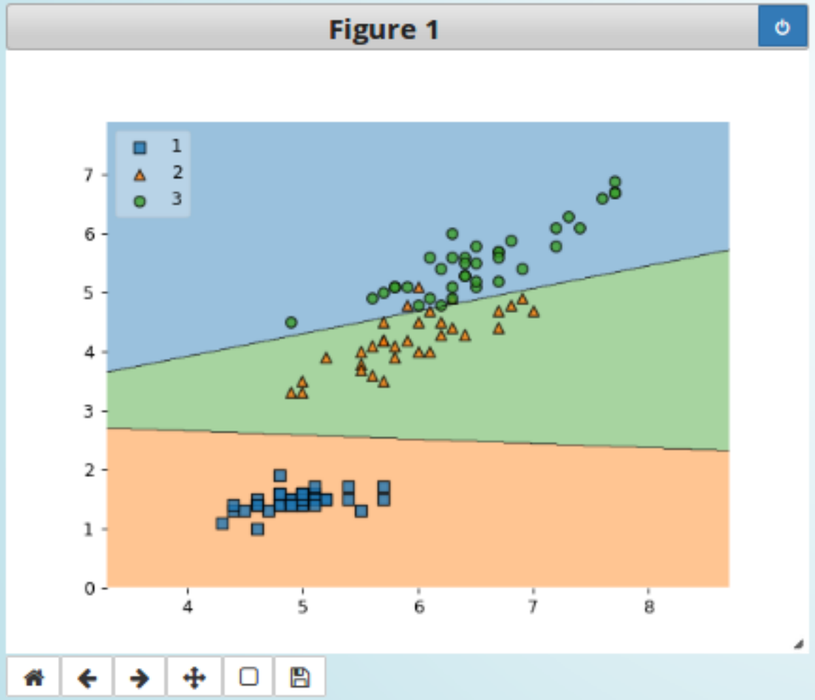
In [5]:

```
# Split the Dataset into Training and Testing

iris_len = pd.DataFrame([iris.SLength, iris.PLength, iris.Class]).transpose()

train, test = ms.train_test_split(iris_len, test_size=0.3, random_state=1)
cTrain, cTest = train.pop('Class'), test.pop('Class')
```

In [6]:

```
# Classifier
clf = svm.SVC(kernel="linear", C=1.0)
clf.fit(train, cTrain)

# Predict
predicted = clf.predict(test)
print("Accuracy: {:.2f}%".format(accuracy_score(cTest, predicted) *100), "\n")

# Visualise the Hyper-plane
%matplotlib notebook
X, Y = train.values, cTrain.values.astype(pd.np.integer)
plot_decision_regions(X=X, y=Y, clf=clf, legend=2)
```

```
Accuracy: 95.56%
```

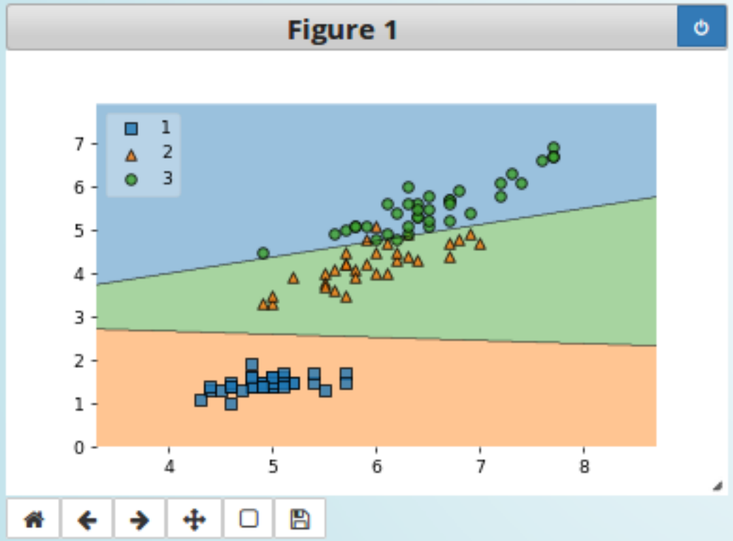


Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f669fbac208>

In [8]:

```
# Classifier
clf = svm.SVC(kernel="linear", C=5.0)
clf.fit(train, cTrain)

# Predict
predicted = clf.predict(test)
print("Accuracy: {:.2f}%".format(accuracy_score(cTest, predicted) *100), "\n")

# Visualise the Hyper-plane
%matplotlib notebook
X, Y = train.values, cTrain.values.astype(pd.np.integer)
plot_decision_regions(X=X, y=Y, clf=clf, legend=2)
```

```
Accuracy: 100.00%
```

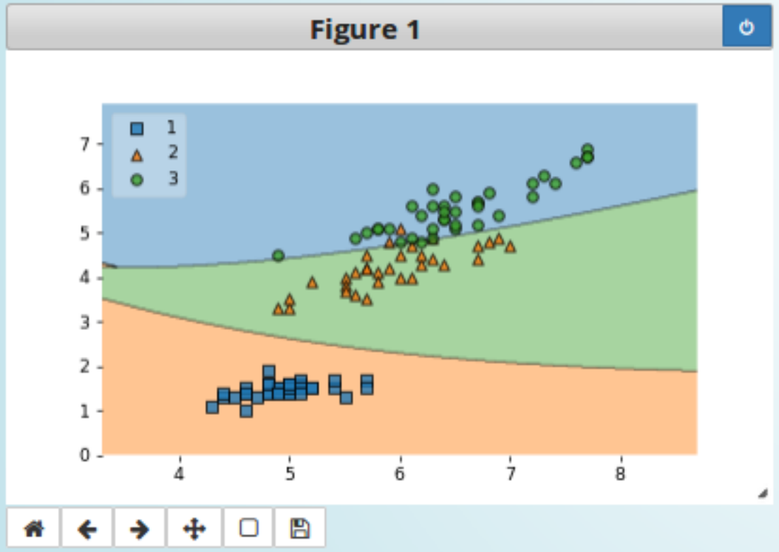


Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f669fb11908>

In [9]:

```
# Classifier
clf = svm.SVC(kernel="poly", gamma=0.1)
clf.fit(train, cTrain)

# Predict
predicted = clf.predict(test)
print("Accuracy: {:.2f}%".format(accuracy_score(cTest, predicted) *100), "\n")

# Visualise the Hyper-plane
%matplotlib notebook
X, Y = train.values, cTrain.values.astype(pd.np.integer)
plot_decision_regions(X=X, y=Y, clf=clf, legend=2)
```

```
Accuracy: 100.00%
```

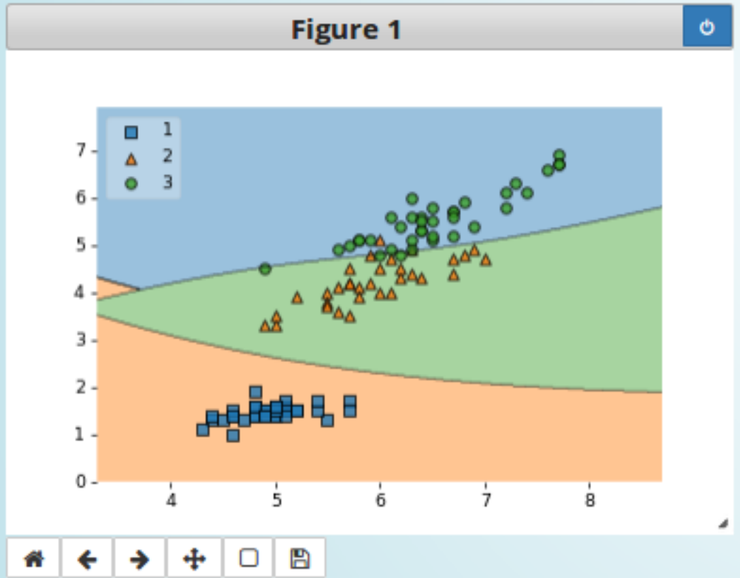


Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f669fadd978>

In [10]:

```
# Classifier
clf = svm.SVC(kernel="poly", gamma=10)
clf.fit(train, cTrain)

# Predict
predicted = clf.predict(test)
print("Accuracy: {:.2f}%".format(accuracy_score(cTest, predicted) *100), "\n")

# Visualise the Hyper-plane
%matplotlib notebook
X, Y = train.values, cTrain.values.astype(pd.np.integer)
plot_decision_regions(X=X, y=Y, clf=clf, legend=2)
```

```
Accuracy: 97.78%
```

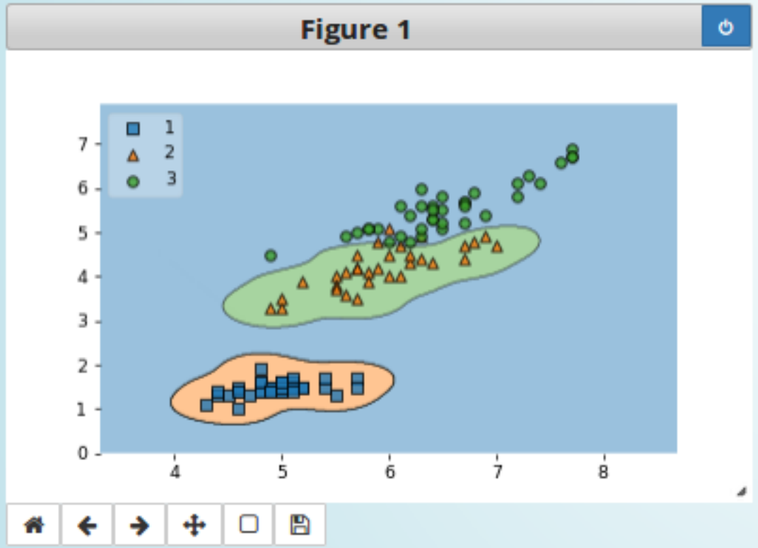


Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f669faa0080>

In [11]:

```
# Classifier
clf = svm.SVC(kernel="rbf", gamma=10)
clf.fit(train, cTrain)

# Predict
predicted = clf.predict(test)
print("Accuracy: {:.2f}%".format(accuracy_score(cTest, predicted) *100), "\n")

# Visualise the Hyper-plane
%matplotlib notebook
X, Y = train.values, cTrain.values.astype(pd.np.integer)
plot_decision_regions(X=X, y=Y, clf=clf, legend=2)
```

```
Accuracy: 95.56%
```

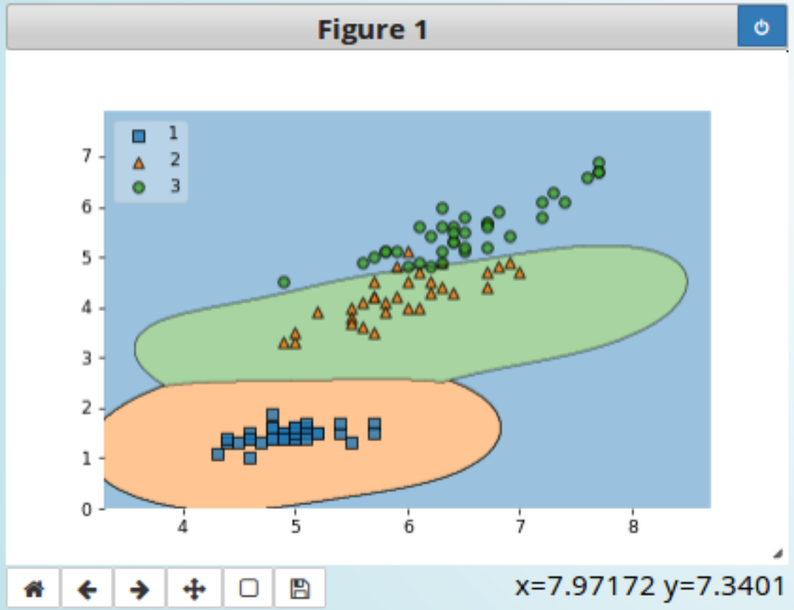


Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f669fa76ac8>

In [12]:

```
# Classifier
clf = svm.SVC(kernel="rbf", gamma=1)
clf.fit(train, cTrain)

# Predict
predicted = clf.predict(test)
print("Accuracy: {:.2f}%".format(accuracy_score(cTest, predicted) *100), "\n")

# Visualise the Hyper-plane
%matplotlib notebook
X, Y = train.values, cTrain.values.astype(pd.np.integer)
plot_decision_regions(X=X, y=Y, clf=clf, legend=2)
```

```
Accuracy: 97.78%
```



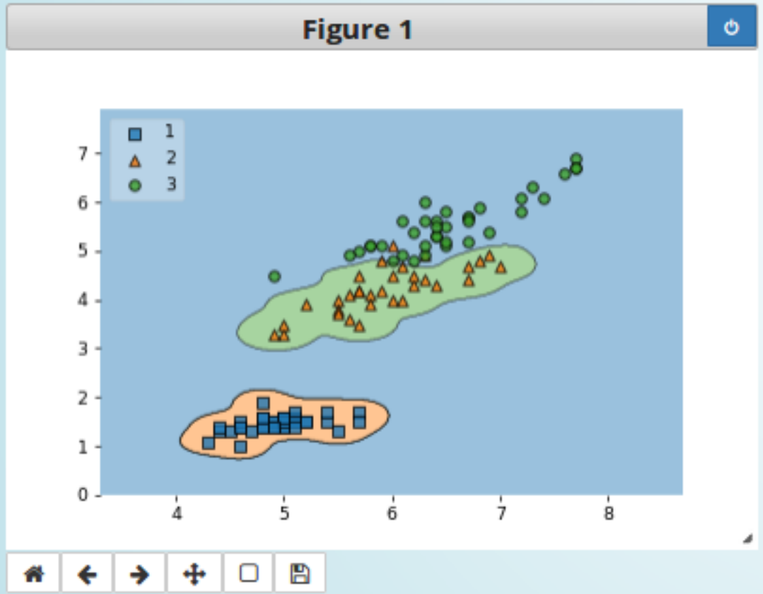
Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f669fa64470>
```

In [13]:

```
# Classifier
clf = svm.SVC(kernel="rbf", gamma=20)
clf.fit(train, cTrain)

# Predict
predicted = clf.predict(test)
print("Accuracy: {:.2f}%".format(accuracy_score(cTest, predicted) *100), "\n")

# Visualise the Hyper-plane
%matplotlib notebook
X, Y = train.values, cTrain.values.astype(pd.np.integer)
plot_decision_regions(X=X, y=Y, clf=clf, legend=2)
```

```
Accuracy: 93.33%
```



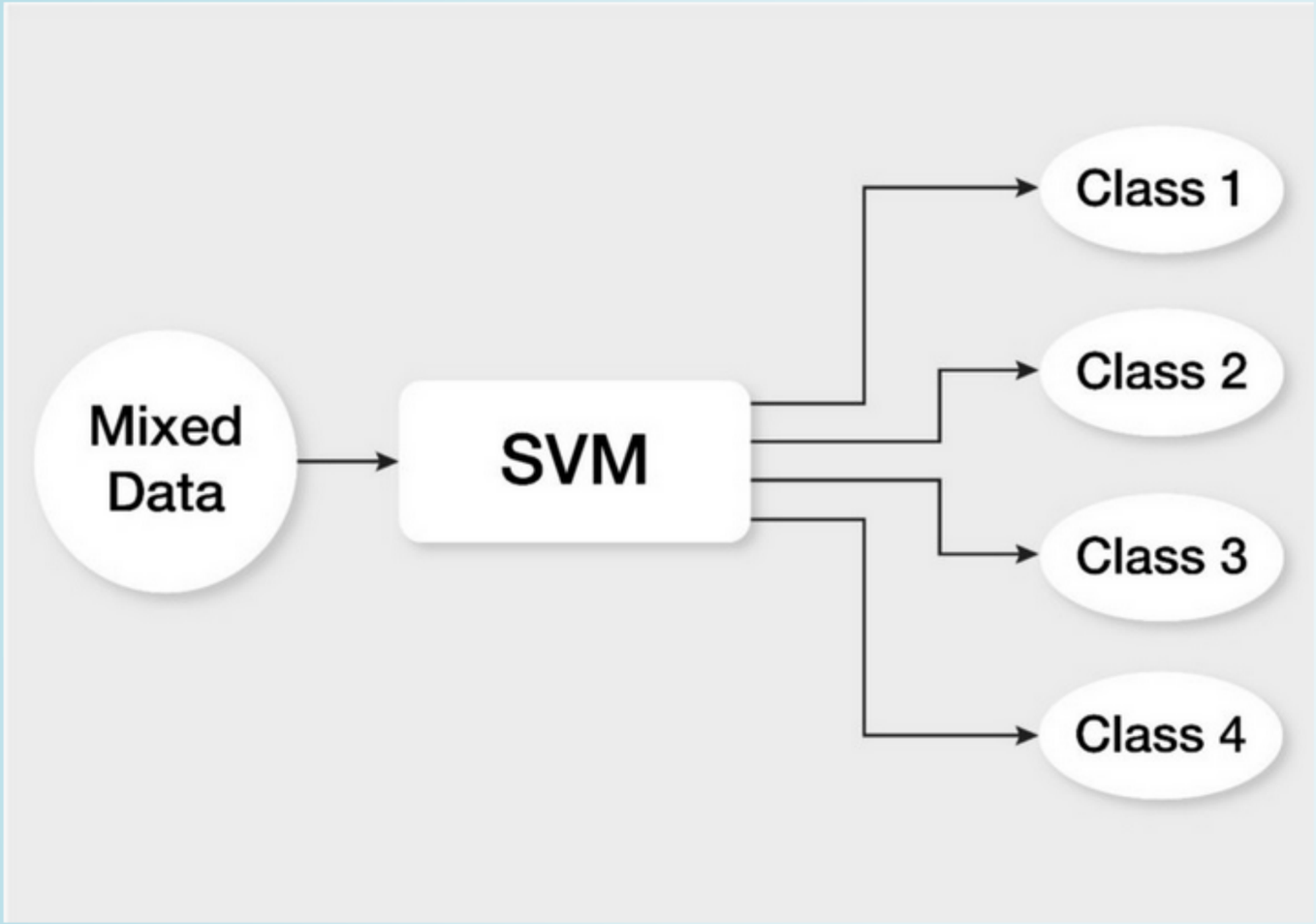
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f669f94b5f8>

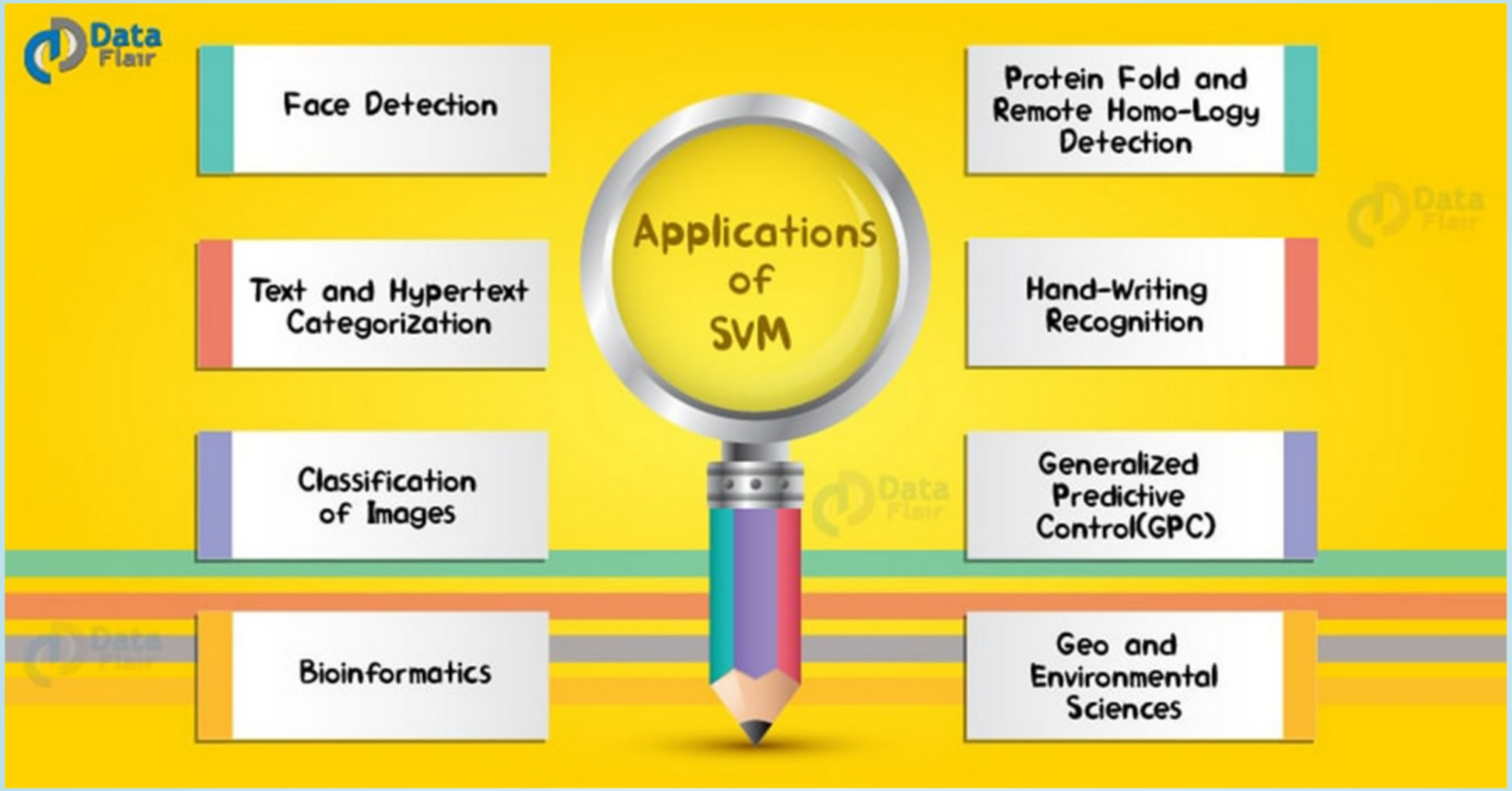
In [14]:

```
1 # Reports
2 print(classification_report(cTest, predicted, target_names=types))
3 print(confusion_matrix(cTest, predicted))
```

```
              precision    recall  f1-score   support

      Setosa       1.00      0.93      0.96        14
  Versicolor       1.00      0.89      0.94        18
   Virginica       0.81      1.00      0.90        13

   micro avg       0.93      0.93      0.93        45
   macro avg       0.94      0.94      0.93        45
weighted avg       0.95      0.93      0.94        45

[[13  0  1]
 [ 0 16  2]
 [ 0  0 13]]
```

SVM IN IMAGE PROCESSING / REMOTE SENSING (BONUS) 😊


Mixed Data
SVM
Class 1
Class 2
Class 3
Class 4


Data Flair
Face Detection
Text and Hypertext Categorization
Classification of Images
Bioinformatics
Applications of SVM
Protein Fold and Remote Homo-Logy Detection
Hand-Writing Recognition
Generalized Predictive Control(GPC)
Geo and Environmental Sciences

REFERENCES

- Algorithms for Image Processing and Computer Vision Second Edition by J.R. Parker (pdf)
- https://en.wikipedia.org/wiki/Statistical_classification
- https://developers.google.com/machine-learning/glossary/
- https://towardsdatascience.com/supervised-machine-learning-classification-5e685fe18a6d
- https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel
- https://www.researchgate.net/post/In_support_vector_machinesSVM_how_we_adjust_the_parameter_C_why_we_use_this_paramete
- https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989
- https://www.quora.com/What-are-C-and-gamma-with-regards-to-a-support-vector-machine
- https://data-flair.training/blogs/applications-of-svm/

PYTHON

- Scikit-learn | Matplotlib | Numpy | Pandas | Ipython | Mlxtend | Cowsay

Built with **Jupyter-Notebook** and hosted with **mybinder**