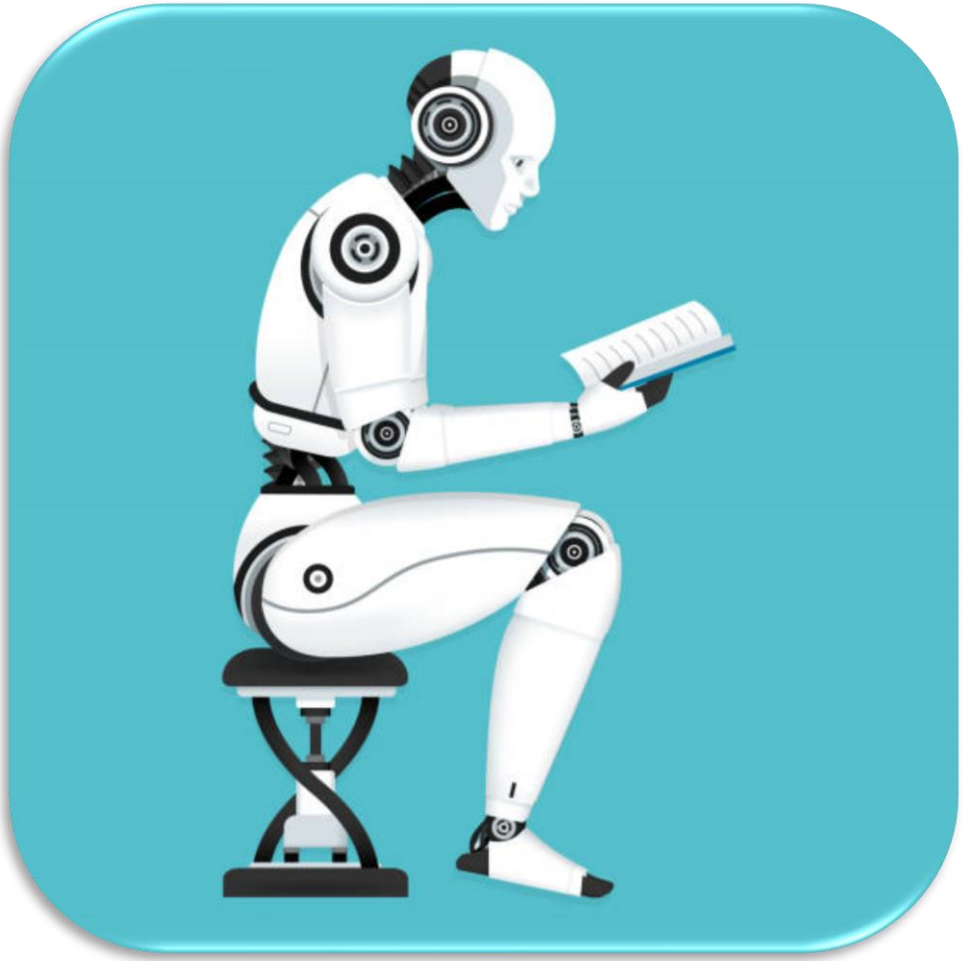
Machine Learning Algorithms

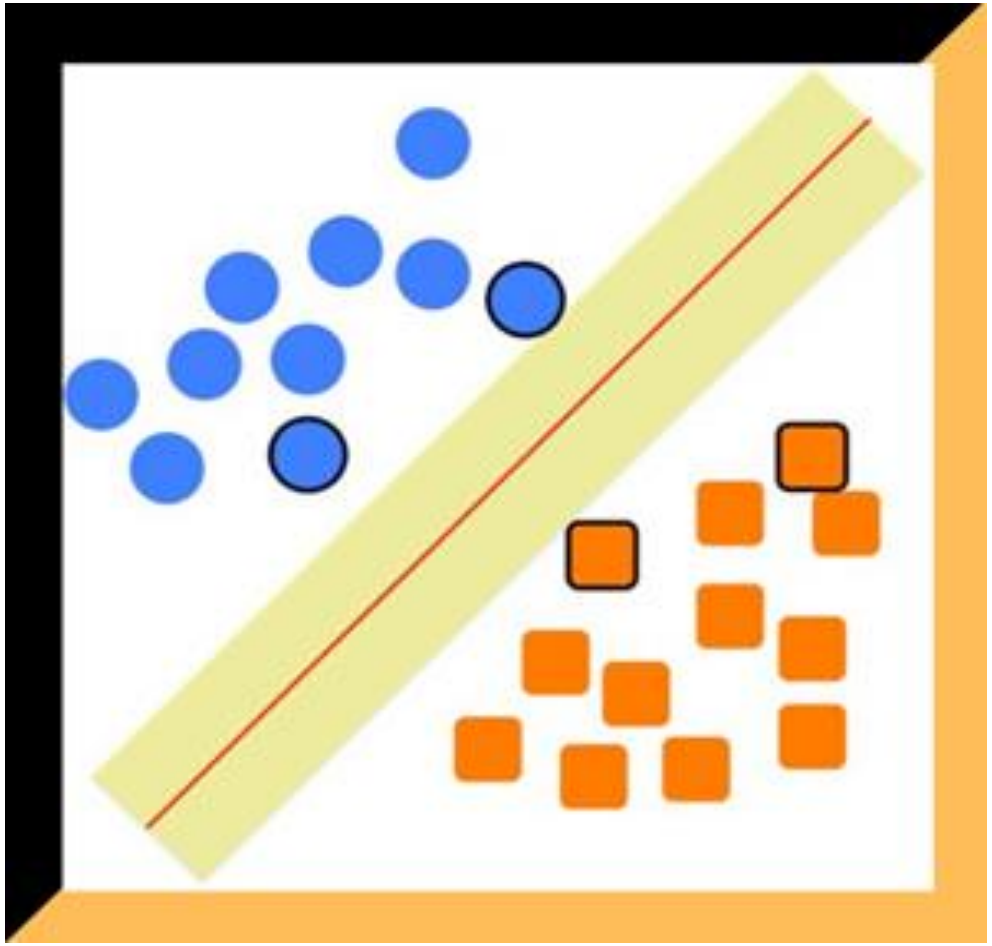


Machine Learning Clip Art - Bing

Objective

- Support Vector Machine
- Features space
- Decision Boundary
- Dimension Expansion
- Hyperplane
- Transformation Approach

Support Vector Machine (SVM)



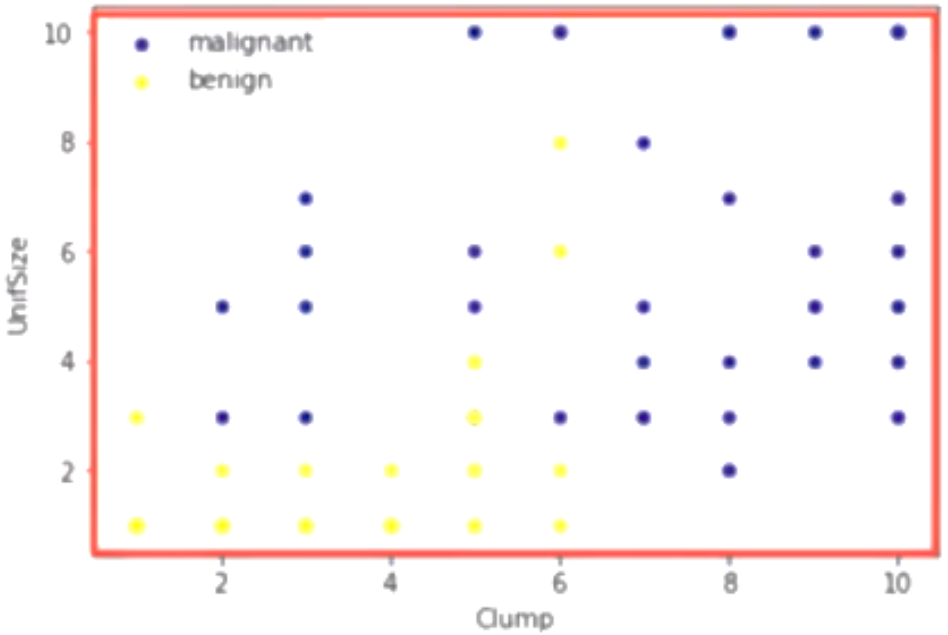
Introduction

- A Support Vector Machine is a supervised algorithm that can classify cases by **finding a separator**.
- SVM works by first, **mapping data to a high-dimensional feature space** so that data points can be categorized, even when the data are not otherwise linearly separable.
- Then, a separator is estimated for the data.
- The data should be transformed in such a way that a separator could be drawn as a hyperplane.

Feature Space

- For example, consider the following figure, which shows the distribution of a small set of cells, only based on their Unit Size and Clump thickness.

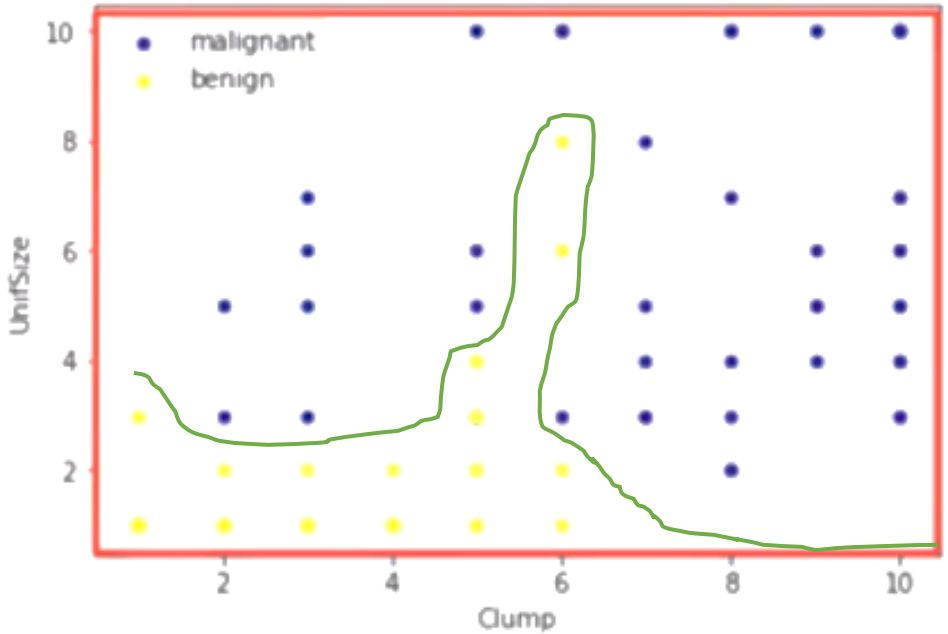
- It represents a linearly, non-separable, dataset.



Scatter Plot

Decision Boundary..

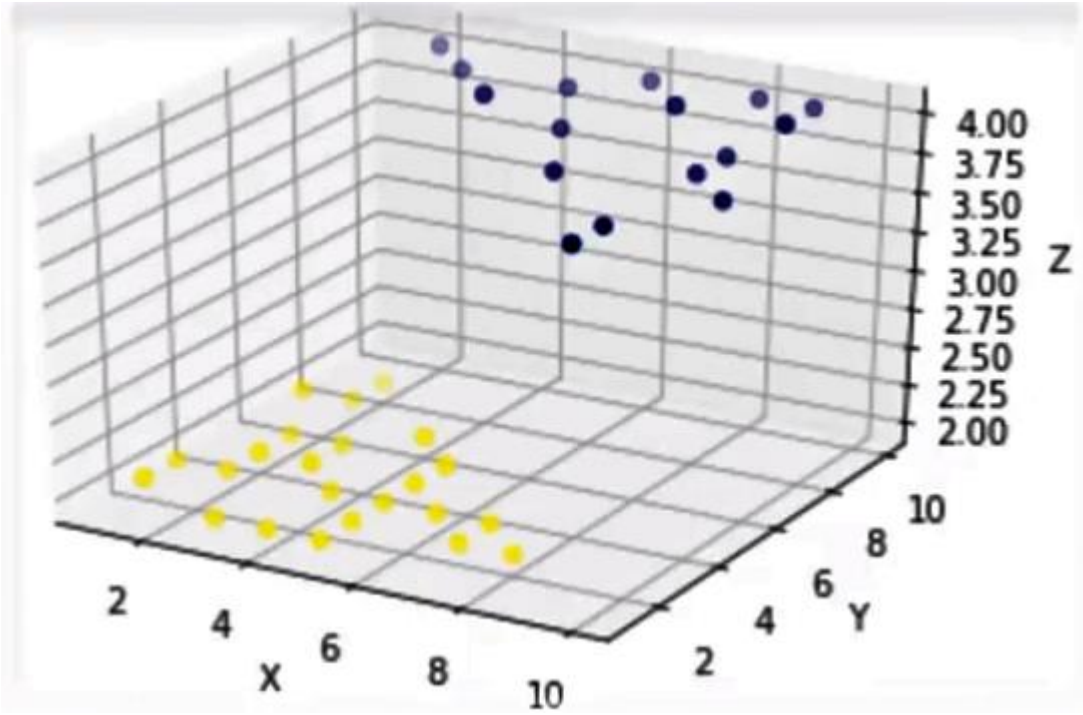
- The two categories can be separated with a curve.
- Not a line that formulates most real world datasets.
- Twist and Turns on trajectory



Scatter Plot

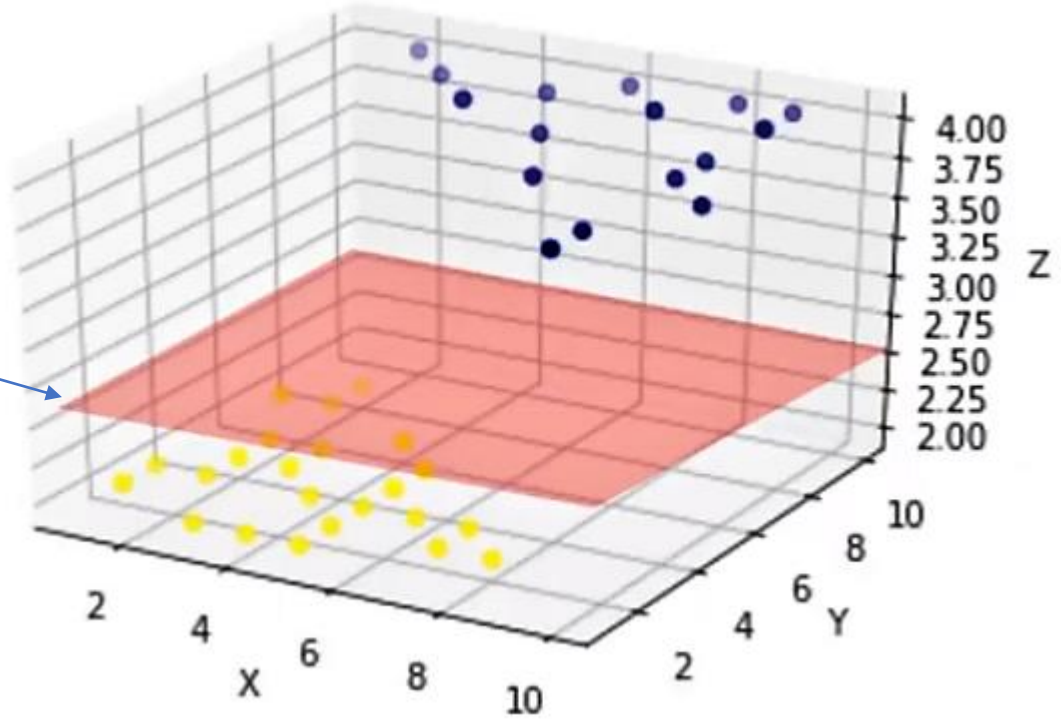
Dimension Expansion

- We can transfer this data to a higher dimensional space.
- For example, mapping it to a 3-dimensional space.
- Separation boundary gets simplified



Hyperplane

- After the transformation, the boundary between the two categories can be defined by a hyperplane.
- As we are now in 3-dimensional space, the separator is shown as a plane.
- **This plane can be used to classify new or unknown cases.**



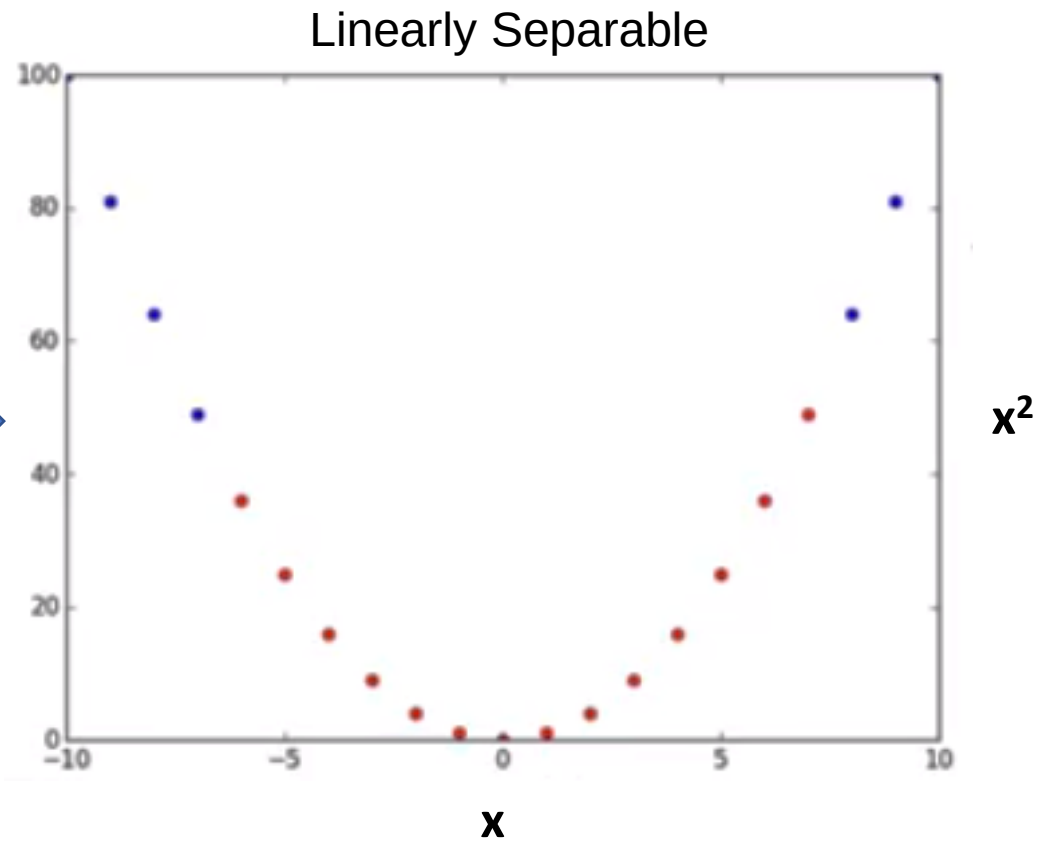
Transformation Approach

- For example, your can increase the dimension of data by

mapping x into a new space using a function, with outputs x and x^2.

$$\emptyset(x) = [x, x^2]$$

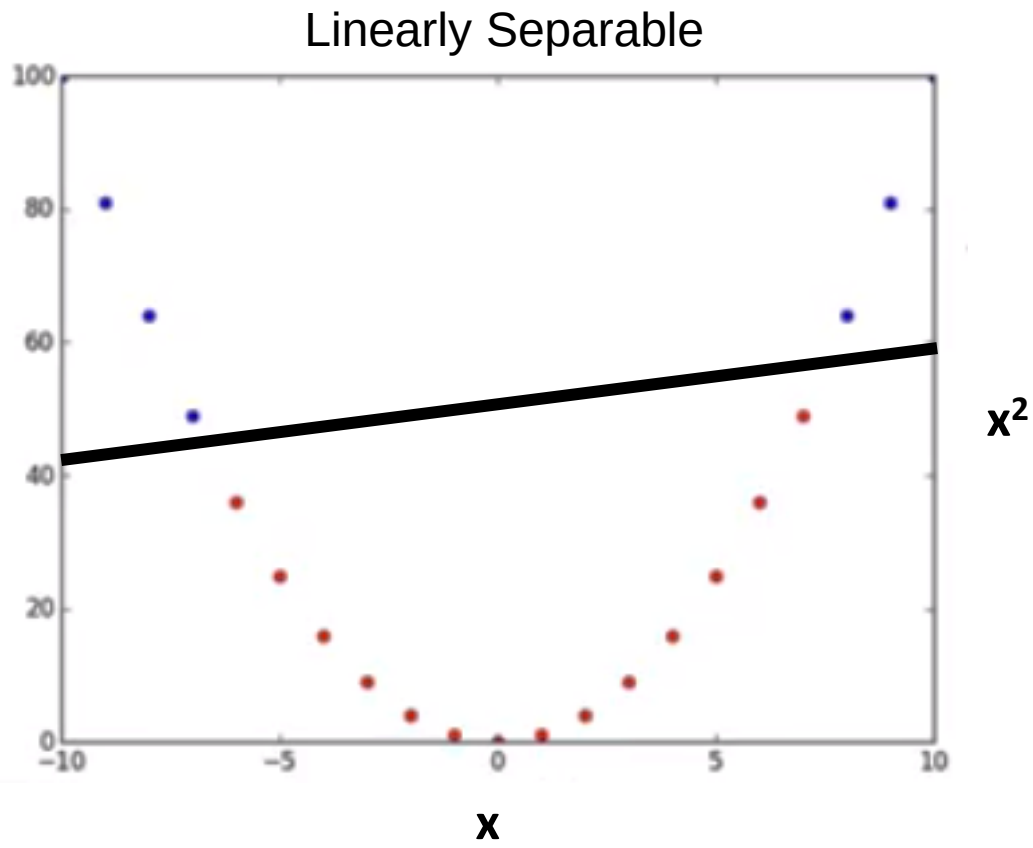
- Now, the data is linearly separable!



Transformation……

- Notice that, as we are in a two dimensional space, the hyperplane is a line dividing a plane into two parts where each class lays on either side.

- Now we can use this line to classify new cases.



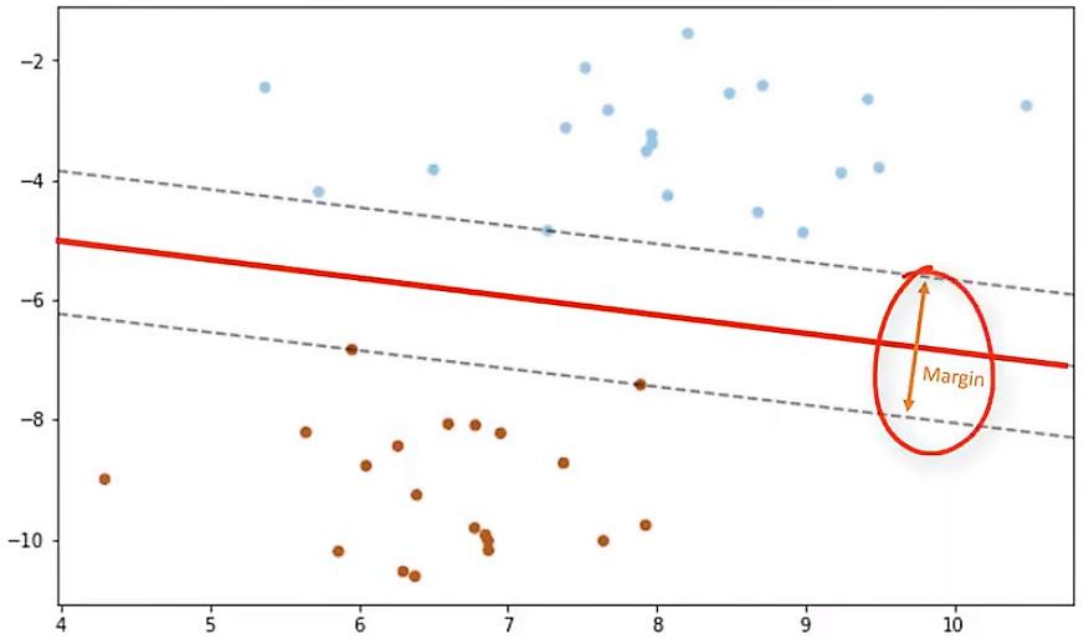
Kernel in Support Vector Machine

- Mapping data into a higher dimensional space is called kernelling.
- The mathematical function used for the transformation is known as the **kernel function**, and can be of different types, such as:
  - Linear,
  - Polynomial,
  - Radial basis function (or RBF), and
  - Sigmoid.
- Already implemented in form of machine learning libraries.
- Choose different functions in turn and compare the results.

Finding optimized separator after transformation

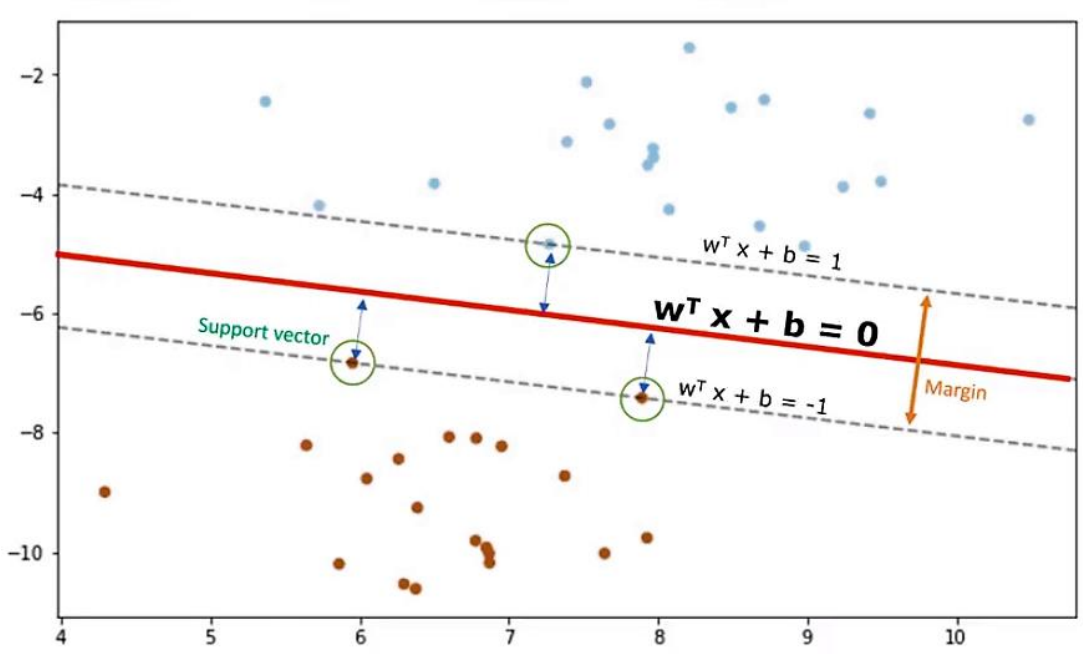
- One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes.

- So, the goal is to choose a hyperplane with as big a margin as possible.



Margins

- Only support vectors matter for achieving our goal; and thus, other training examples can be ignored.
- We try to find the hyperplane in such a way that it has the maximum distance to support vectors, called optimal Hyperplane.
- Hyperplane is learned from training data using an optimization procedure that maximizes the margin;
- This optimization problem can be solved by Gradient descent.



SVM Outcomes

- **The output of the algorithm is the values ‘w’ and ‘b’ for the line.**
- You can make classifications using this estimated line.
- It is enough to plug in input values into the line equation, then, you can calculate whether an unknown point is above or below the line.
- If the equation returns a value greater than 0, then the point belongs to the first class, which is above the line, and vice versa.

Evaluation Metrics in Classification

- **Evaluation metrics explain the performance of a model.**
- Imagine that we have an historical dataset which shows the customer churn for a telecommunication company.
- We have trained the model, and now we want to calculate its accuracy using the test set.
- We pass the test set to our model, and we find the predicted labels.
- Now the question is, “**How accurate is this model?**”
- Basically, we compare the actual values in the test set with the values predicted by the model, to calculate the accuracy of the model.

Evaluation Metrics in Classification

- Evaluation metrics provide a key role in the development of a model, as they **provide insight to areas that might require improvement.**
- There are different model evaluation metrics but we will talk about three popular metrics:

1. Jaccard index
2. F1-score and
3. Log Loss.

Jaccard index

- Also known as the Jaccard Similarity Coefficient
- Let's say *y* = true labels of the dataset.
- And y_cap be predicted values estimated by our classifier.
- Then we can define Jaccard Index as,

the size of intersection divided by the size of union of true and predicted dataset

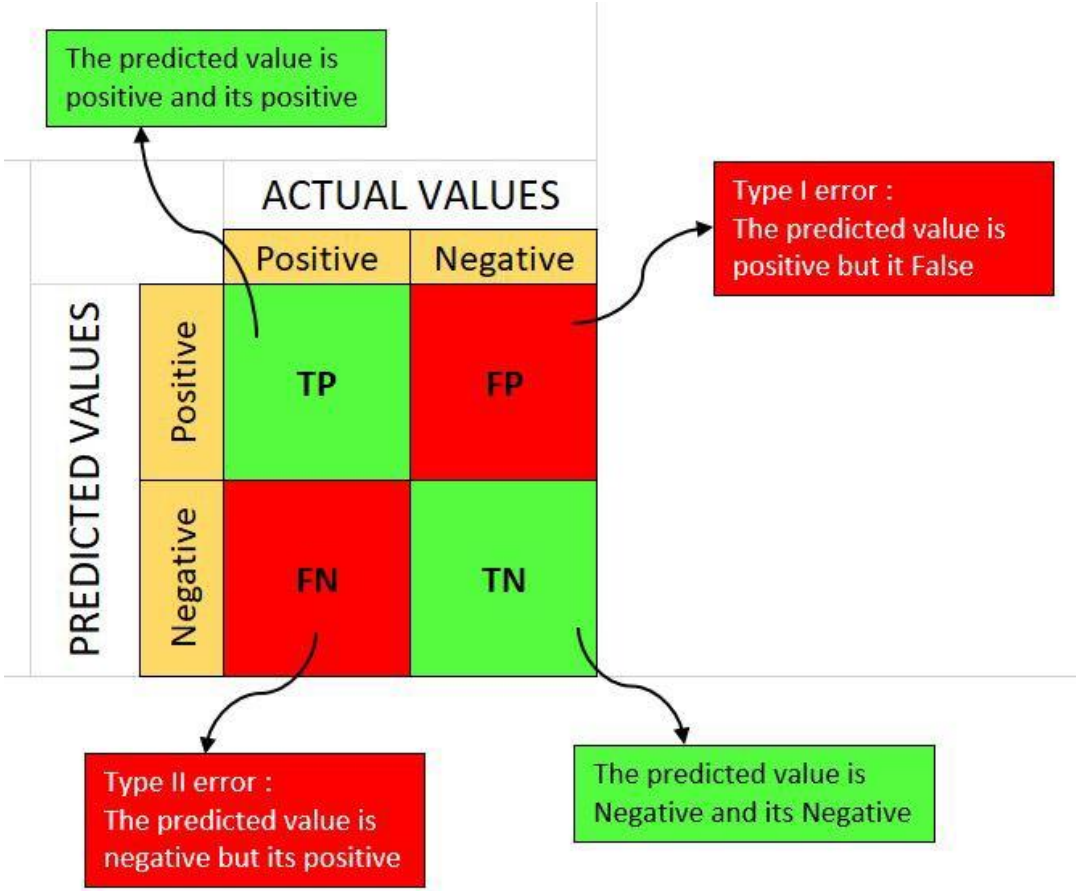
$$J(y,\hat{y}) = \frac{|y \cap \hat{y}|}{|y \cup \hat{y}|} = \frac{|y \cap \hat{y}|}{|y| + |\hat{y}| - |y \cap \hat{y}|}$$

Jaccard index

- **If the entire set of predicted labels for a sample strictly matches with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.**

- Another way of looking at accuracy of classifiers is to look at a confusion matrix.

Confusion Matrix



https://miro.medium.com/max/667/1*3yGLac6F4mTENnj5dBNvNQ.jpeg

Confusion Matrix

- The predicted classes are represented in the columns of the matrix, whereas the actual classes are in the rows of the matrix.
- We then have four cases:

1. **True positives (TP):** the cases for which the classifier predicted ‘spam’ and the emails were actually spam.
2. **True negatives (TN):** the cases for which the classifier predicted ‘not spam’ and the emails were actually real.
3. **False positives (FP):** the cases for which the classifier predicted ‘spam’ but the emails were actually real.
4. **False negatives (FN):** the cases for which the classifier predicted ‘not spam’ but the emails were actually spam.

Realization….

- The entries of the confusion matrix are the number of occurrences of each class for the dataset being analyzed.
- Let's obtain the confusion matrix for our spam filtering algorithm, by using the function confusion_matrix:

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

- The output is:

```
[[724   7]
 [  1  136]]
```

Inference from Conf. Matrix

- Let's interpret results,
- Out of 731 actual instances of 'not spam' (first row), the classifier correctly predicted 724 of them.
- Out of 137 actual instances of 'spam' (second row), the classifier correctly predicted 136 of them.
- Out of all 868 emails, classifier correctly predicted 860 of them.
- This allow us to obtain the accuracy of classification from confusion matrix using below formulae

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Inference from Confusion Matrix

- We obtain again that 99% of the predicted outputs were correctly classified.

- However, the confusion matrix allows us to have a better picture of the performance of the algorithm.

Precision, recall and f1-score

- Precision: When positive result is predicted, how often is it correct.
- Used when we need to limit the number of False Positives

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall: When it is actually positive result, how often does it predict correctly.
- Used when we need to limit the number of False Negatives
- Recall also known as “Sensitivity” or the True Positivite Rate (TPR)

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision, recall and f1-score

- f1-score: this is just the harmonic mean of precision and recall

$$\text{f1-score} = 2 \times \frac{precision \times recall}{precision + recall}$$

- Useful when you need to take precision and recall into account
- If you try to optimize recall you algorithm will predict more examples to belong to positive class, but that may result in many false positive hence low precision
- On other hand, if you try to optimize precision, your model will predict very few examples as positive results (one with highest probability), but the recall will be very low
- f1-score reaches is best value as 1 (represent perfect precision and recall) and the worst value as 0.

Simulation Output

from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))

- *The output is:*

	precision	recall	f1-score	support
False	1.00	0.99	0.99	731
True	0.95	0.99	0.97	137
avg / total	0.99	0.99	0.99	868

Logarithmic (Log) Loss

- Now let's look at another accuracy metric for classifiers.
- **Sometimes, the output of a classifier is the "probability of a class label", instead of the "label".**
- For example, in logistic regression, the output can be the probability of customer churn, i.e., yes (or equals to 1).

- **This probability is a value between 0 and 1.**

- Logarithmic loss (also known as Log loss) measures the performance of a classifier where the predicted output is a probability value between 0 and 1

Log Loss

- In order to calculate Log Loss, classifier needs to assign a probability to each class rather than yielding more likely class.

- Log Loss can be defined as,

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \times p_{ij}$$

where,

N is the number of samples or instances.

M is the number of possible labels

SVM Applications

- Image Analysis such as image classification and digit recognition
- Text mining
- Detecting spam
- Text categorization
- Sentiment analysis
- Gene Expression data classification

Case Study…

- Analysis of the original data showed that many of the characteristics differed significantly between benign and malignant samples.

- You can use the values of these cell characteristics in samples from other patients to give an early indication of whether a new sample might be benign or malignant.

ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl	Mit	Class
1000025	5	1	1	1	2	1	3	1	1	benign
1002945	5	4	4	5	7	10	3	2	1	benign
1015425	3	1	1	1	2	2	3	1	1	malignant
1016277	6	8	8	1	3	4	3	7	1	benign
1017023	4	1	1	3	2	1	3	1	1	benign
1017122	8	10	10	8	7	10		7	1	malignant
1018099	1	1	1	1	2	10	3	1	1	benign
1018561	2	1	2	H	2	1	3	1	1	benign
1033078	2	1	1	1	2	1	1	1	5	benign
1033078	4	2	1	1	2	1	2	1	1	benign

ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl	Mit	Class
1000015	6	1	1	1	7	1	3	1	1	

?

Case Study….

- Here, SVM can be used as a classifier, to train your model to understand **patterns within the data to identify benign or malignant cells.**
- Once the model has been trained, it can be used to predict your new or unknown cell with rather high accuracy.

Hands On