

- 1.1. There are numerous examples of supervised machine learning in use today.

For example, fraud detection by credit card companies based on your prior usage behavior.

<https://towardsdatascience.com/detecting-credit-card-fraud-using-machine-learning-a3d83423d3b8>

Another example is precision medicine in healthcare, in which prognosis and treatment is customized to a particular patient.

<https://towardsdatascience.com/precision-medicine-and-machine-learning-11060caa3065>

- 1.2. Unsupervised learning has been applied to drug discovery by biomedical institutions and pharmaceutical companies. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6552674/>
- 1.3. Reinforcement Learning has many applications including applications in trading and finance. <https://neptune.ai/blog/reinforcement-learning-applications>
- 1.4. The code and output are below:

```

-----
def new_soln(sentence):
    return "".join([char for char in sentence if char not in "aeiouAEIOU"])

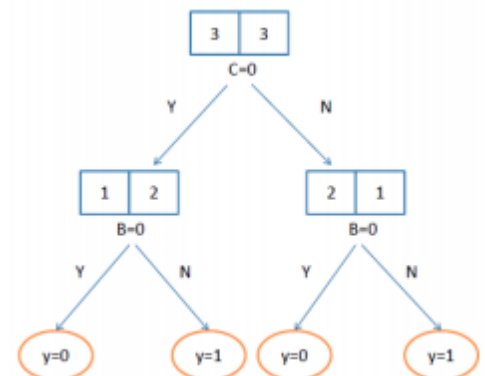
sentence = "The quick brown fox jumps over the lazy dog"
print("Filtered result: " + new_soln(sentence))
-----
Filtered result: Th qck brwn fx jmps vr th lzy dg

```

---

- 2.1. For  $n$  binary attributes, there are  $2^n$  possible values. Each value can be mapped to 2 outputs/labels. So, in this scenario, there are  $2^{(2^n)}$  possible decision trees. This is the same as the number of Boolean functions. [https://en.wikipedia.org/wiki/Boolean\\_function](https://en.wikipedia.org/wiki/Boolean_function)

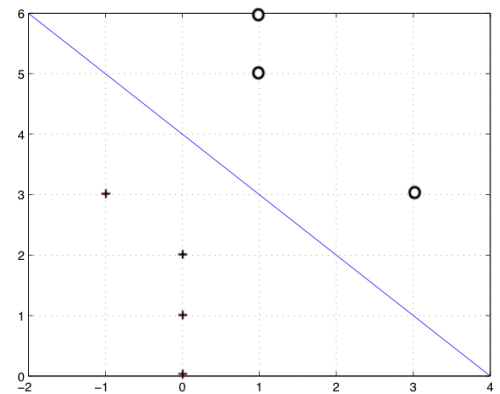
- 2.2. (a) The entropy of the output  $Y$  is 1.
- (b) If you split at  $A$ , then the output is partitioned as  $A=0$ :  $Y=0\ 0\ 1$  and  $A=1$ :  $0\ 1\ 1$   
 So,  $H(y|A=0)=0.9183$  and  $H(y|A=1)=0.9183$  and  
 then  $H(Y|A)=Pr(A=0) \cdot H(y|A=0) + Pr(A=1) \cdot H(y|A=1)=0.9183$   
 Similarly:  $H(Y|B)=1$  and  $H(Y|C)=0.9183$   
 So, we see that  $B$  is worse than  $A$  or  $C$ , either of which is good. Randomly choose 1 and continue to build the tree.
- (c) In the example below, I chose to split at  $C$  but you could also have chosen to get another decision tree. So, the decision tree in this case is not unique.



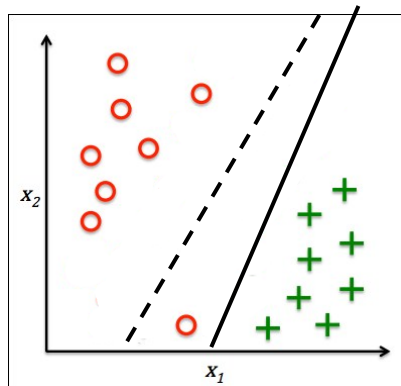
3.1. (a). The plot is as follows. Positive samples are plotted as '+' and negative samples are plotted as 'o'. The solid line is the large margin solution.

(b)  $w = [-1 \ -1]$  and  $b = 4$ , making the solution:  $y_{\text{hat}} = \text{sign}(w^t \cdot [x_1, x_2]^t + 4)$ . Any suitable scaling of the above function is also a valid solution.

(c) Note that all of the new positive points are in the interior of the positive class, so they actually do not affect the SVM linear boundary. Other linear classifiers such as the perceptron will change since they respond to the mean of positive class (which is shifted by the addition of new positive points).



3.2.



- The solid line is the SVM linear classifier when  $C$  is large.
- The dotted line is the SVM linear classifier when  $C$  is small.
- The large  $C$  solution resembles the hard margin SVM.
- $C$  controls the penalty for misclassifications. So large  $C$  leads to not allowing misclassifications, which leads to overfitting to noisy samples and outliers. Small  $C$  leads to allowing misclassifications and enables ignoring outliers.

4.1. a) Accuracy is not a good metric for imbalanced datasets.

b) Precision is important when being right (positive prediction is correct) outweighs detecting all positives. For example, recommendation systems.

c) Recall is important when false negatives are catastrophic. For example, disease detection.

4.2. It is possible for classifier 1 to have higher accuracy but lower precision and recall than classifier 2. In general, whether a classifier is better than the other depends on the measure used to judge "better", so it is important to make sure that you choose the measure that matches your problem.

		Classifier 1		Classifier 2		
		actual class		actual class		Accuracy=(TP+TN)/(TP+TN+FP+FN) Classifier 1: (4+0)/(4+1+5+0)=4/10 Classifier 2: (0+3)/(0+2+5+3)=3/10 < 4/10 Precision=TP/(TP+FP)
		class 1 (neg)	class 2 (pos)	class 1 (neg)	class 2 (pos)	Recall=TP/(TP+FN) Classifier 1: Precision=Recall=0 (since TP=0) Classifier 2: Precision=3/8 Recall=3/5
predicted class	class 1 (neg)	4 (TN)	5 (FN)	0 (TN)	2 (FN)	
	class 2 (pos)	1 (FP)	0 (TP)	5 (FP)	3 (TP)	