IRON HACK

SUPERVISED LEARNING ALGORITHMS

WHAT WILL WE COVER?

- **Recap of what is supervised learning**
- **Core Concepts required**
- **Classification Algorithms**
- **Regression Algorithms**

SUPERVISED: SHOULD I PLAY GOLF?

The objective is to predict if based on the weather conditions of a particular day, we should go play Golf?

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY
0	Rainy	Hot	High	False
1	Rainy	Hot	High	True
2	Overcast	Hot	High	False
3	Sunny	Mild	High	False
4	Sunny	Cool	Normal	False
5	Sunny	Cool	Normal	True
6	Overcast	Cool	Normal	True
7	Rainy	Mild	High	False
8	Rainy	Cool	Normal	False
9	Sunny	Mild	Normal	False
10	Rainy	Mild	Normal	True
11	Overcast	Mild	High	True
12	Overcast	Hot	Normal	False
13	Sunny	Mild	High	True

SUPERVISED: SHOULD I PLAY GOLF?

But how does my algorithm learn?

Based on past datapoints!!!

This is the "Supervised" component

These are called the labels

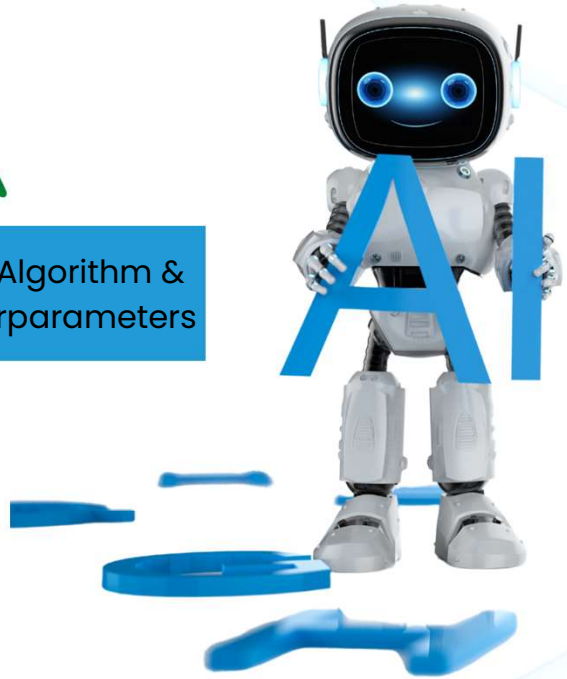
Supervised Learning is performed on labelled data

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

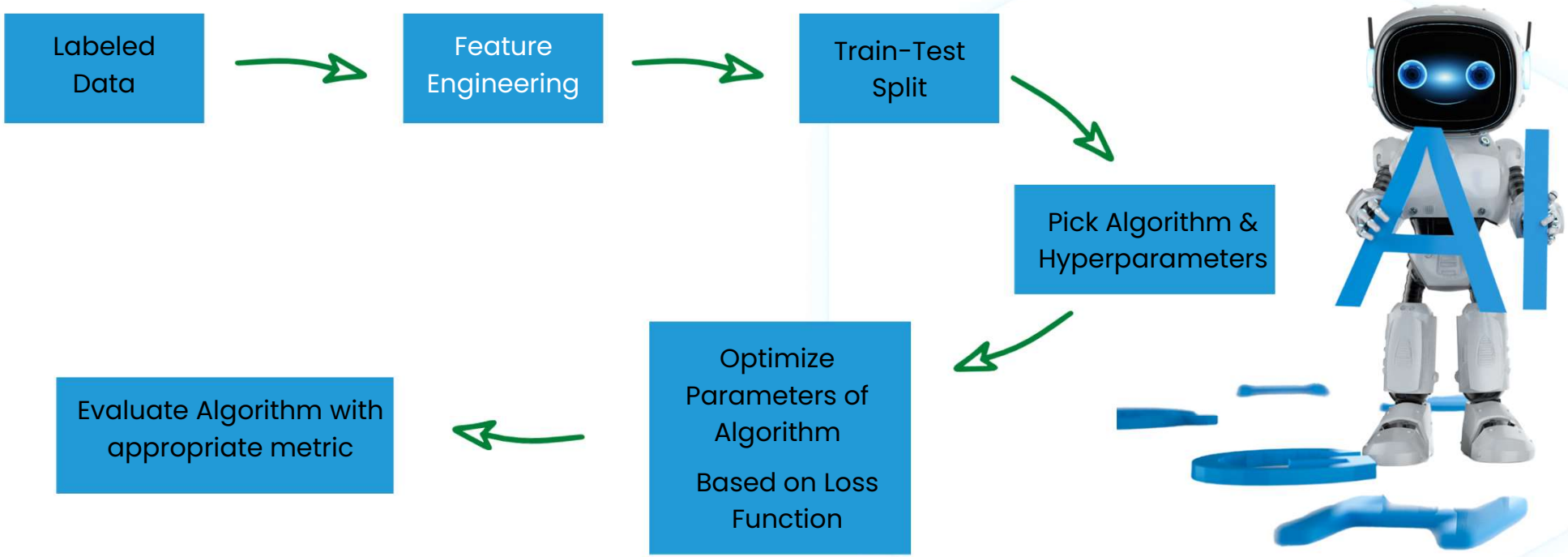
THE MACHINE LEARNING PROCESS

Labeled Data → Feature Engineering → Train-Test Split → Pick Algorithm & Hyperparameters → Optimize Parameters of Algorithm Based on Loss Function → Evaluate Algorithm with appropriate metric

Process of obtaining labeled data is tricky and many times the bottleneck of obtaining an efficient algorithm



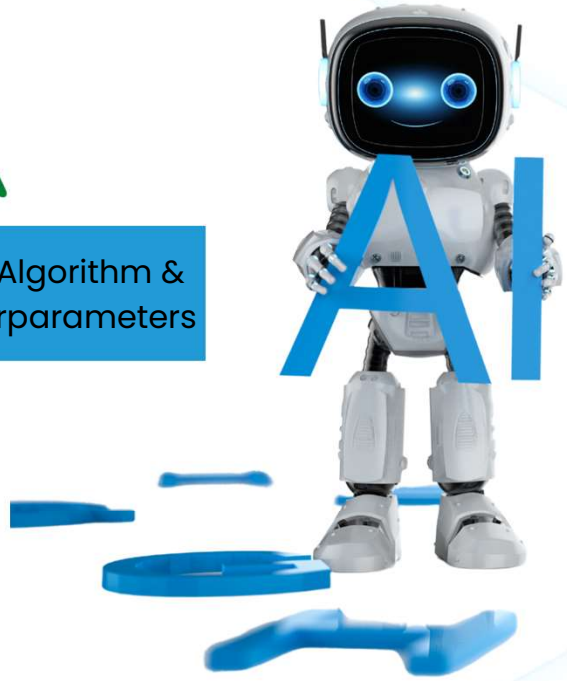
THE MACHINE LEARNING PROCESS



THE MACHINE LEARNING PROCESS

Labeled Data → Feature Engineering → Train-Test Split → Pick Algorithm & Hyperparameters → Optimize Parameters of Algorithm Based on Loss Function → Evaluate Algorithm with appropriate metric

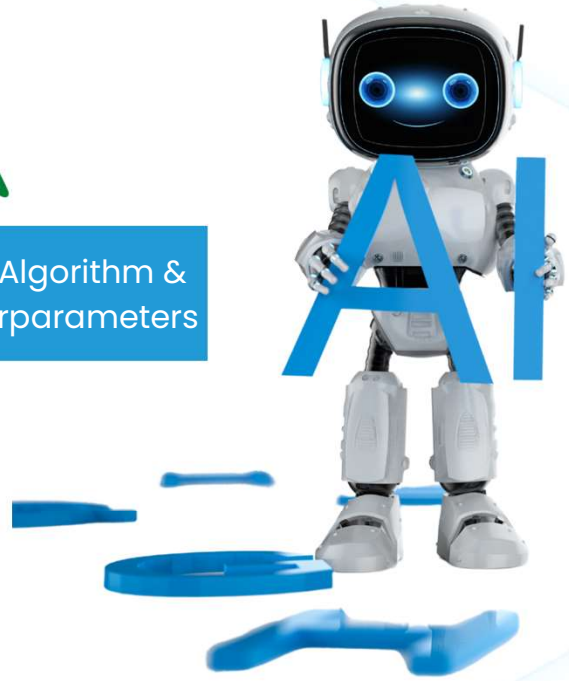
Can be done in a conventional way or with more advaned technique (balancing dataset or cross validation)



THE MACHINE LEARNING PROCESS

Labeled Data → Feature Engineering → Train-Test Split → Pick Algorithm & Hyperparameters → Optimize Parameters of Algorithm Based on Loss Function → Evaluate Algorithm with appropriate metric

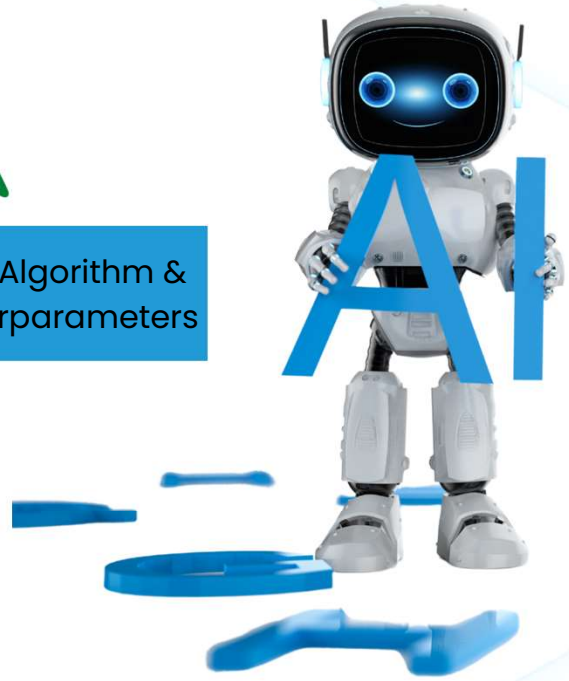
This is where you earn your salary. With experience you will have intuition of which algorithm will be best suited for your situation. The hyperparameters have to be set



THE MACHINE LEARNING PROCESS

Labeled Data → Feature Engineering → Train-Test Split → Pick Algorithm & Hyperparameters → Optimize Parameters of Algorithm Based on Loss Function → Evaluate Algorithm with appropriate metric

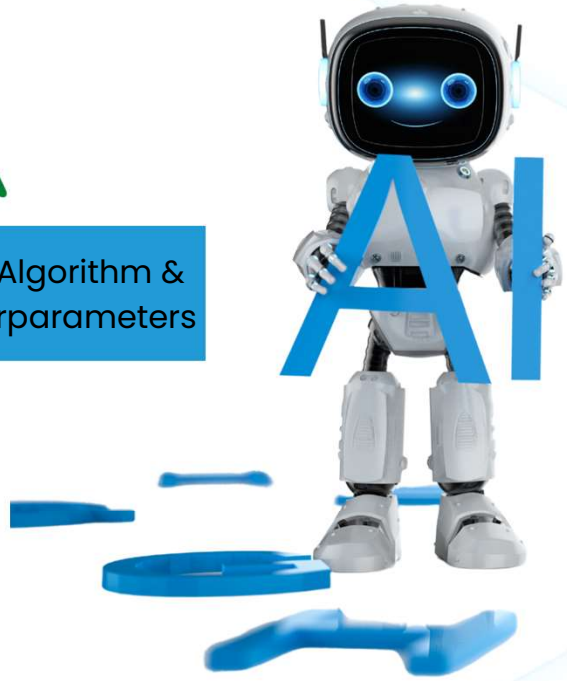
Loss Function vs Cost Function
Attributes a Penalty to each incorrect prediction.
Parameters are optimized to minimize this



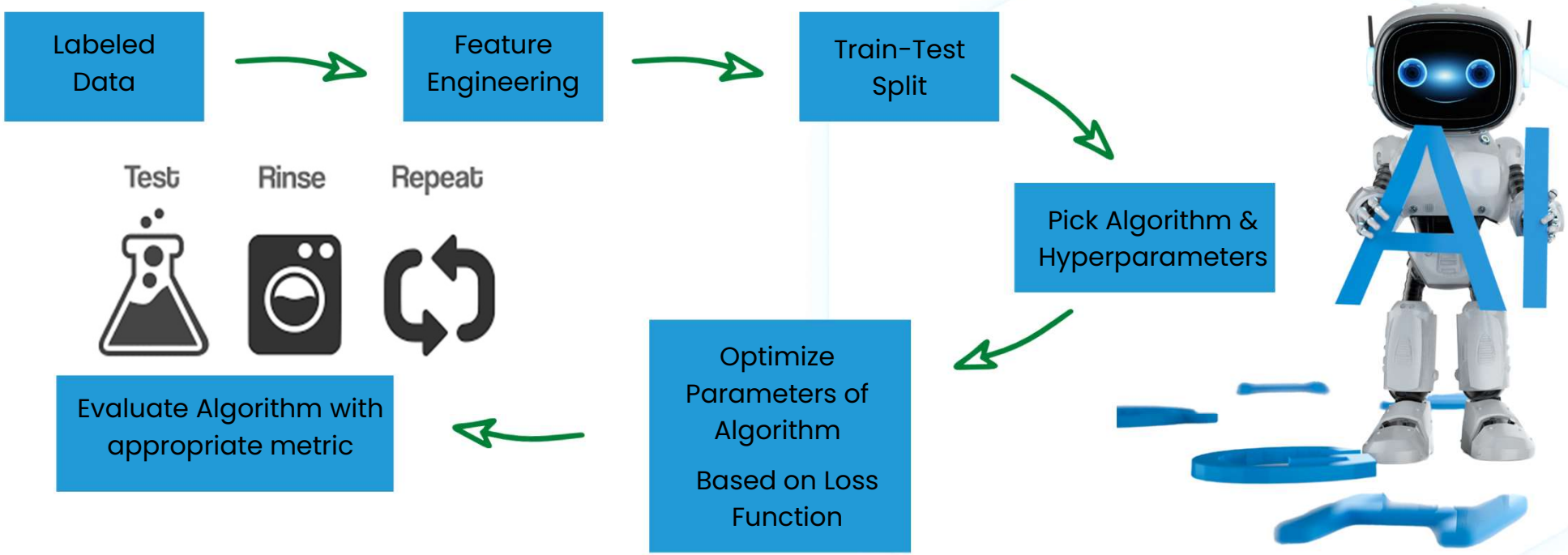
THE MACHINE LEARNING PROCESS

Labeled Data → Feature Engineering → Train-Test Split → Pick Algorithm & Hyperparameters → Optimize Parameters of Algorithm Based on Loss Function → Evaluate Algorithm with appropriate metric

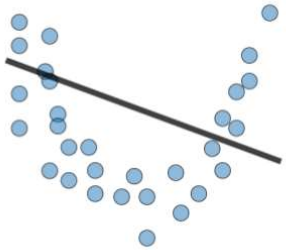
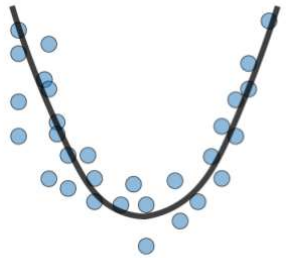
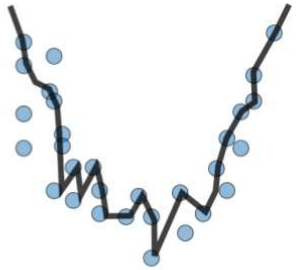
Once we are happy with our algorithm training we need to see an appropriate evaluation metric to make our final judgement



THE MACHINE LEARNING PROCESS



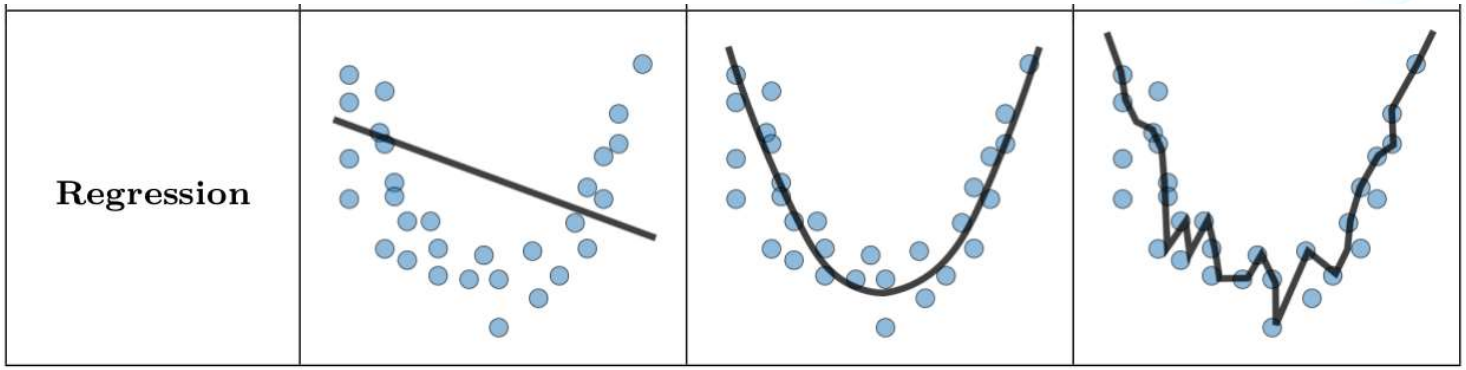
CLASSIFICATION ALGORITHMS

	Underfitting	Just right	Overfitting
Symptoms	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Regression			

- **This is the biggest danger when creating machine learning algorithms**

Underfitting: model is too simplistic
Overfitting: model adapts too much to the training data and does not generalize properly

CLASSIFICATION ALGORITHMS



- **Model Generelization is how well will the trained model perform on new unseen data?**
- **This is why we perform the train-test split!!!**
- **This way we give the algorithm some data to train, and then a smaller sample to test how does the model behave on unseen data**

CLASSIFICATION ALGORITHMS

- **KNN** (Regression Friendly) (Multi Class)
- **Logistic Regression**
- **Decision Trees** (Regression Friendly) (Multi Class)
- **Naive-Bayes Classifier** (Multi Class)
- **Support Vector Machine (SVM)**

CLASSIFICATION ALGORITHMS

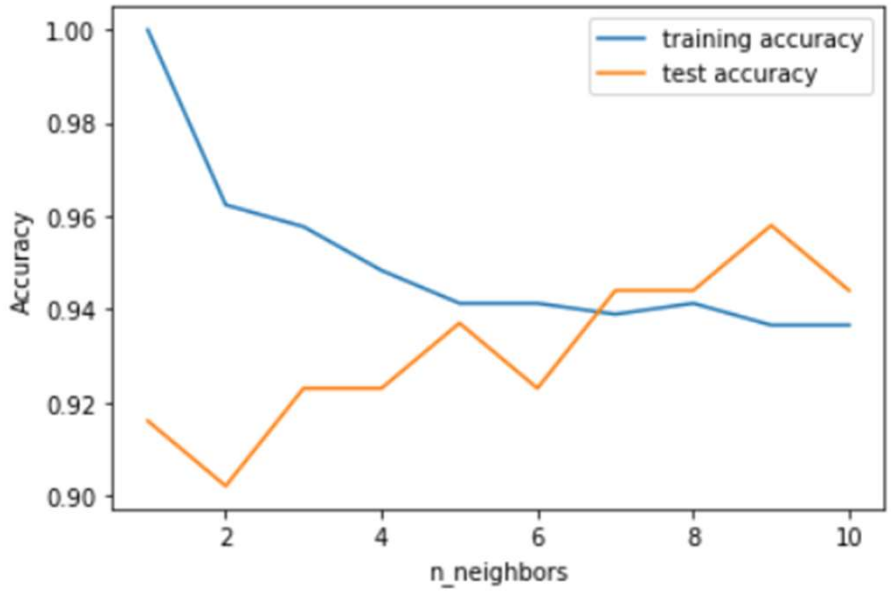
K- Nearest Neighbours (KNN)

The Copy-Cat of the Neighbours

Hyperparameters:

- **K - # of neighbours**
- **Method of weight of neighbours**

K NEAREST NEIGHBOURS



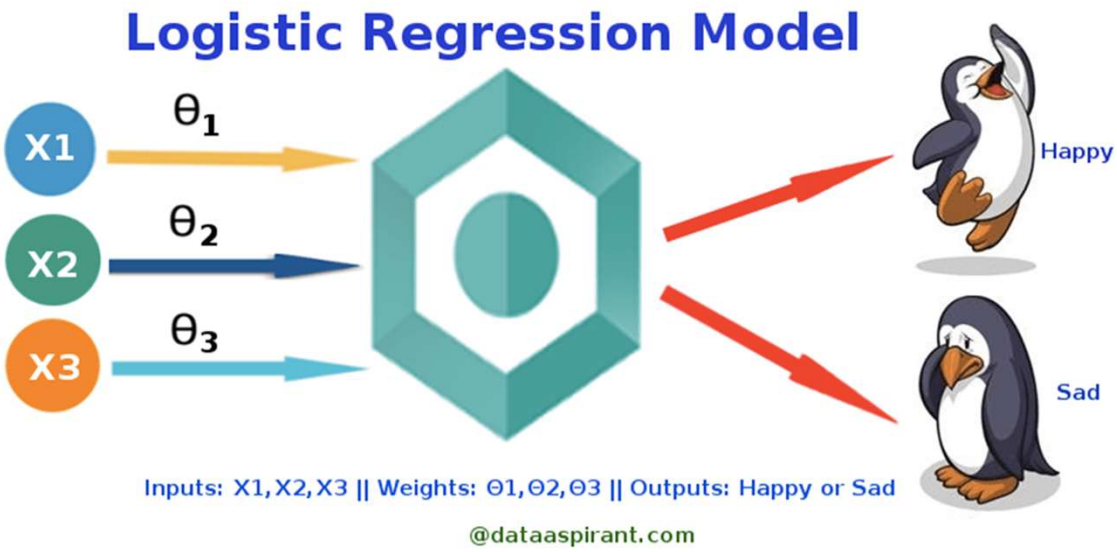
Example of Overfitting and training vs test tradeoff

go to colab

Jupyter Notebook on this Algorithm

Data Squad | Supervised Learning

LOGISTIC REGRESSION

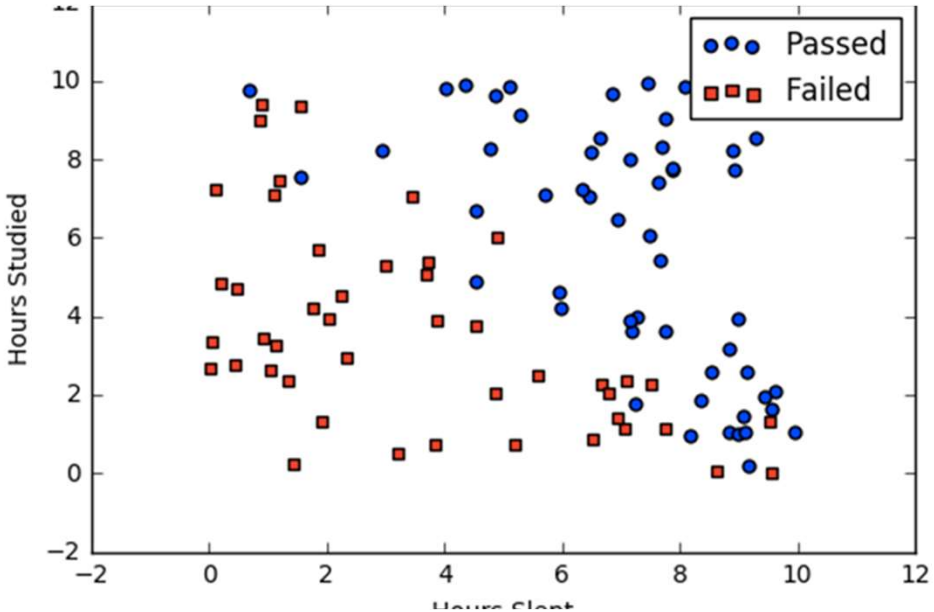


Binary Classification Algorithm

Takes in numerical inputs and

Hyperparameter: Decision Boundary

LOGISTIC REGRESSION



Studied	Slept	Passed
4.85	9.63	1
8.62	3.23	0
5.43	8.23	1
9.21	6.34	0

Steps:

- **Perform usual linear Regression**
- **Pass result through sigmoid function**
- **Compare result to decision boundary**
- **Make Prediction**

LOGISTIC REGRESSION - LINEAR REGRESSION STEP

Studied	Slept	Passed
4.85	9.63	1
8.62	3.23	0
5.43	8.23	1
9.21	6.34	0

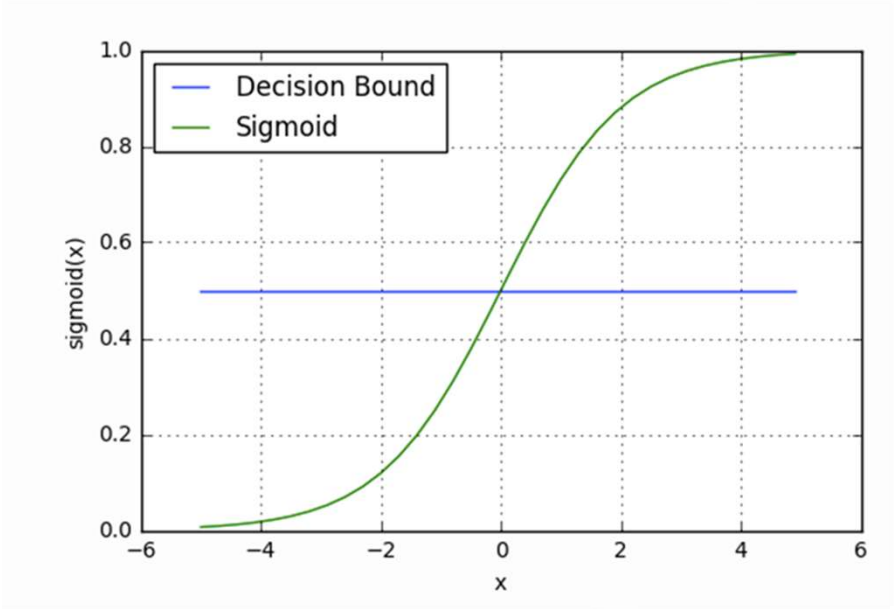
$$z = W_0 + W_1 Studied + W_2 Slept$$

Here there is the need to optimize the parameters of the linear regression

But do we compare z directly to something?
NO

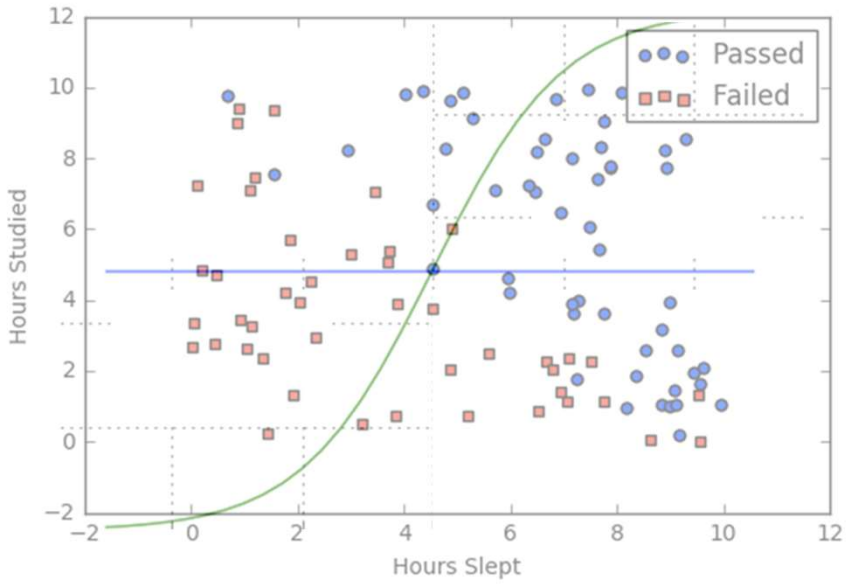
LOGISTIC REGRESSION - SIGMOID FUNCTION

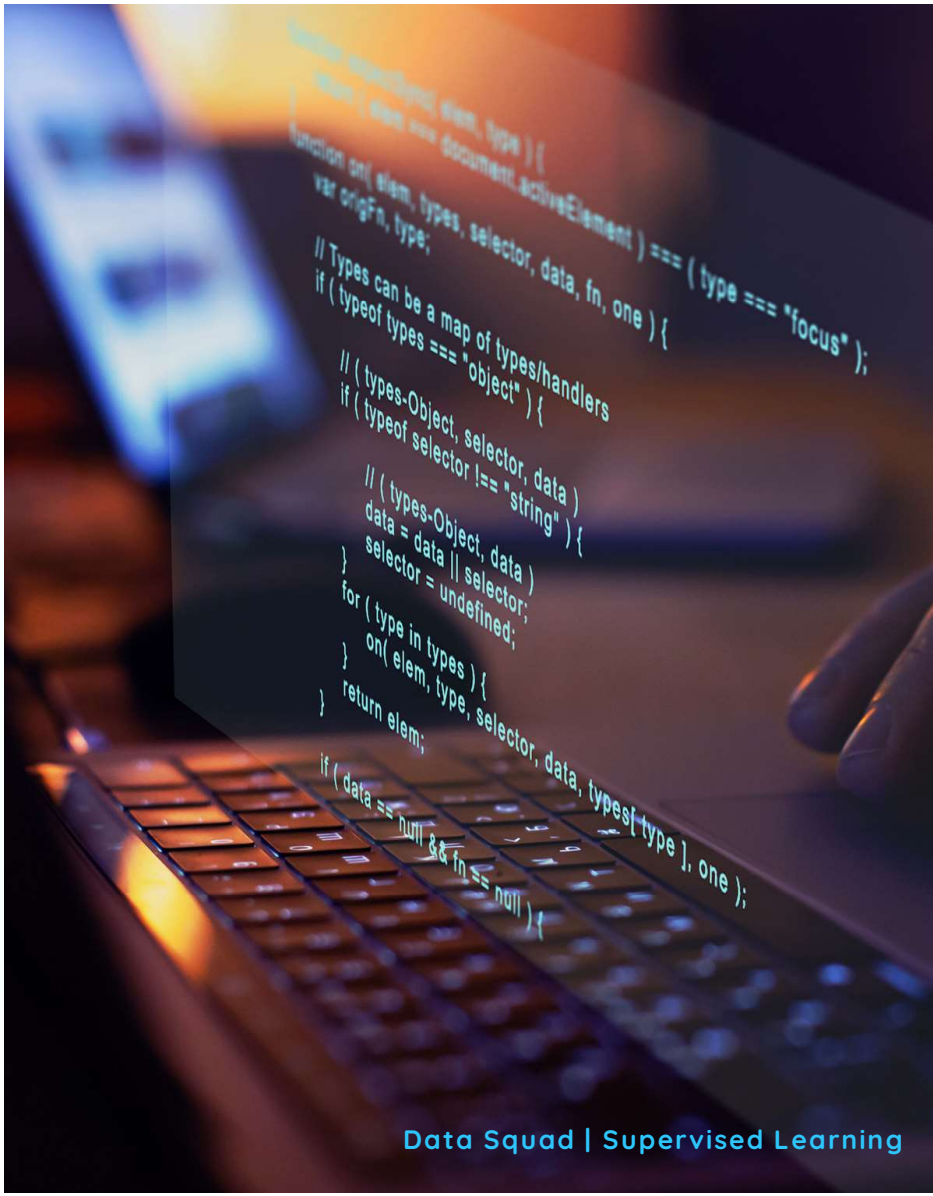
This is where things become Nonlinear



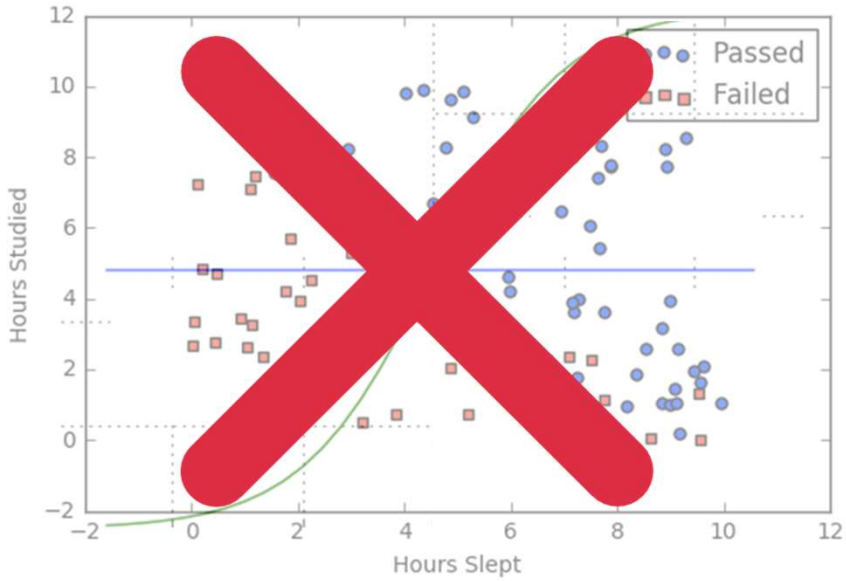
$$z = W_0 + W_1 Studied + W_2 Slept$$

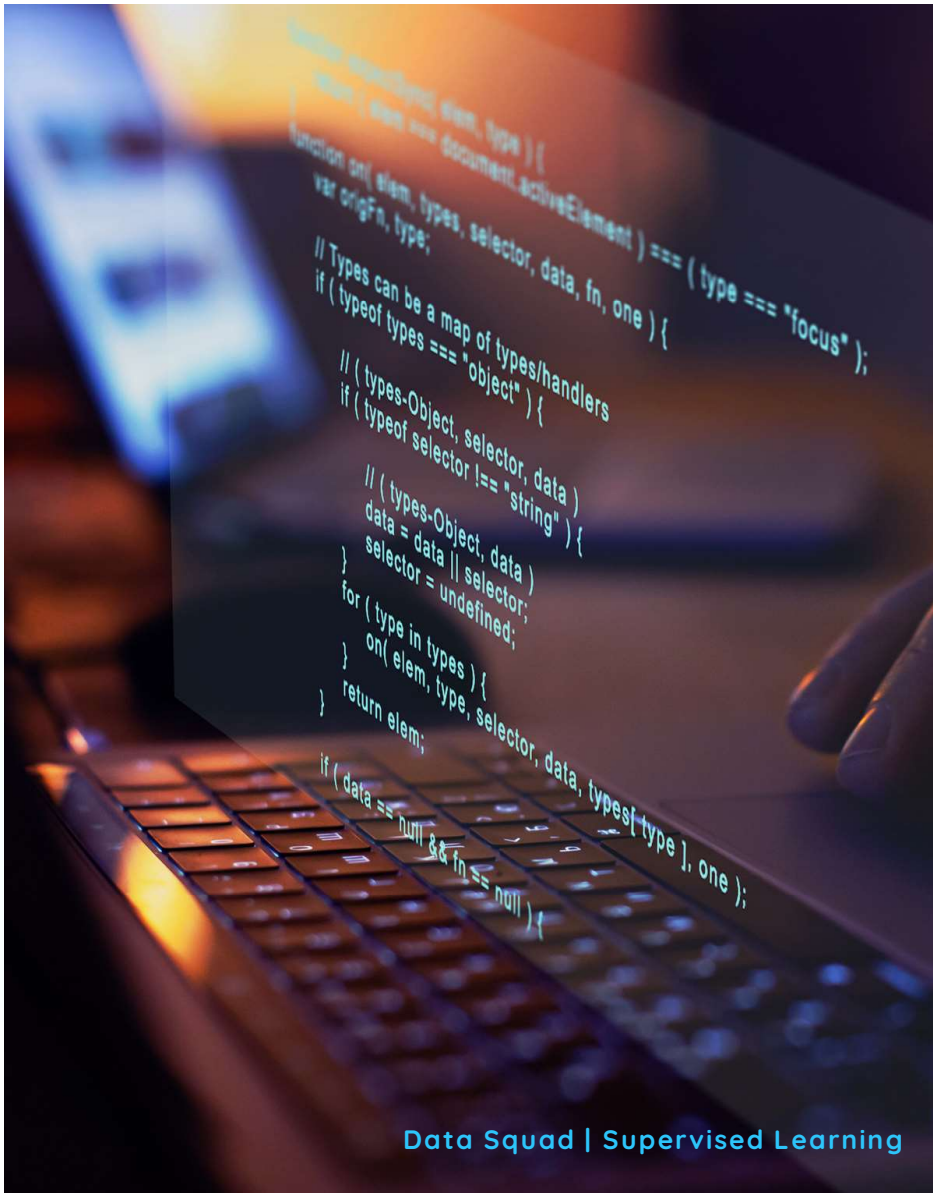
LOGISTIC REGRESSION





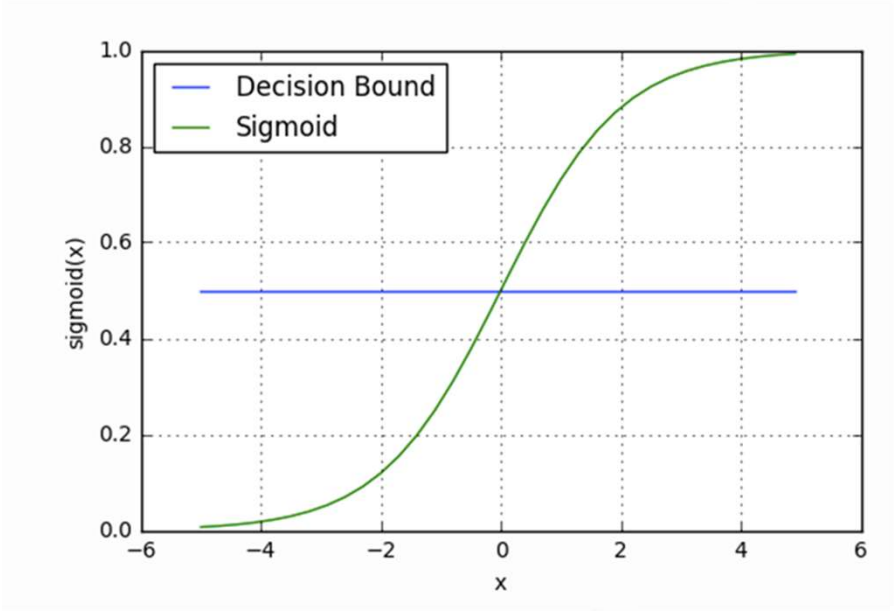
LOGISTIC REGRESSION





LOGISTIC REGRESSION - SIGMOID FUNCTION

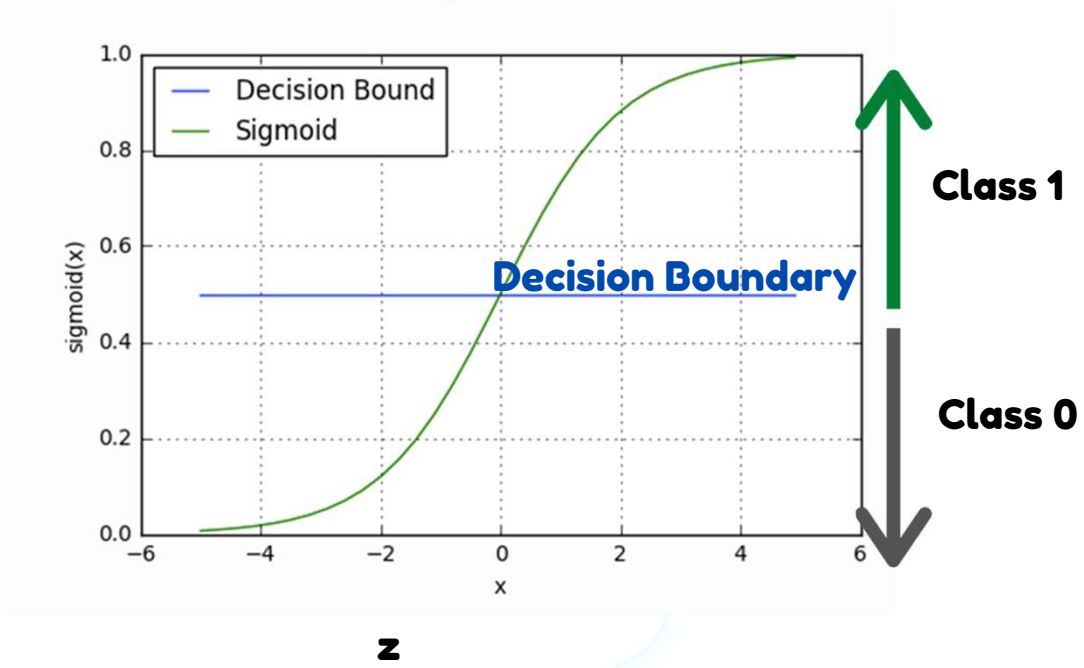
This is where things become Nonlinear



$$z = W_0 + W_1 Studied + W_2 Slept$$

$$f(z) = \frac{1}{1 + e^{-z}}$$

LOGISTIC REGRESSION - SIGMOID FUNCTION

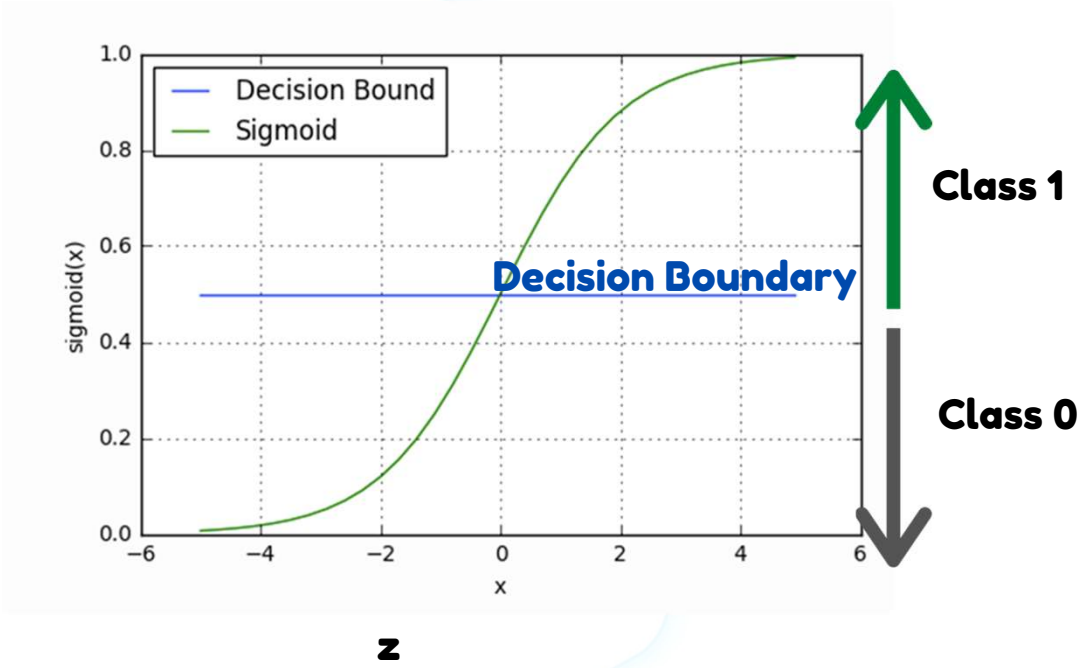


$$z = W_0 + W_1 Studied + W_2 Slept$$

$$f(x) = \frac{1}{1 + e^{-x}}$$

$$P(class = 1) = \frac{1}{1 + e^{-z}}$$

LOGISTIC REGRESSION - SIGMOID FUNCTION

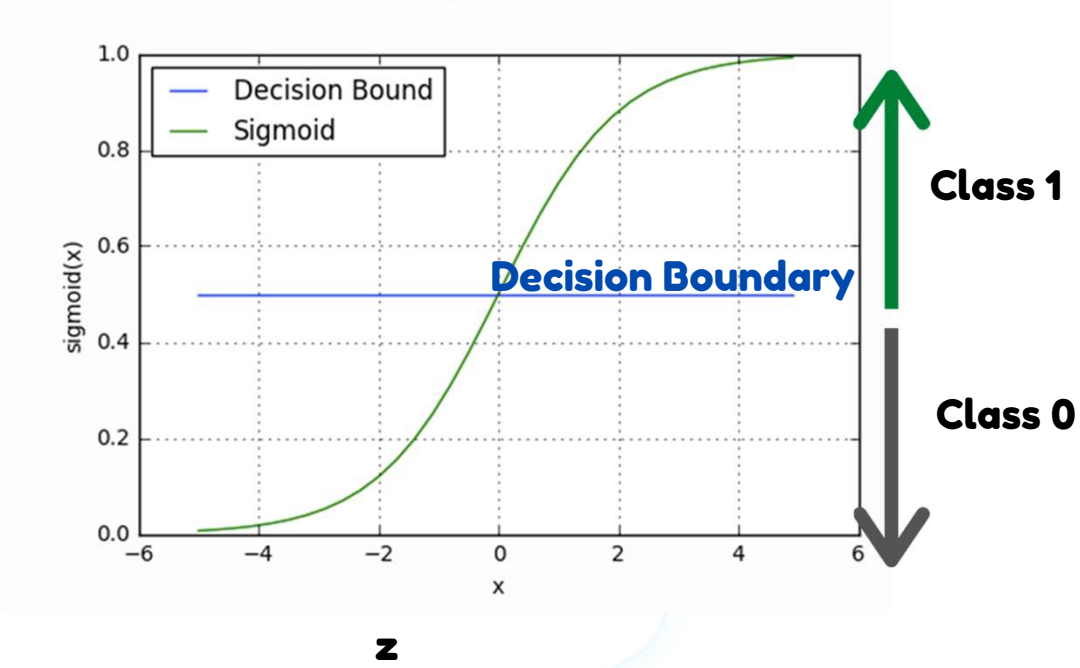


$$z = W_0 + W_1 Studied + W_2 Slept$$

For a set of the regression parameters, predictions are made and then compared with labels

instead of maximising the # of correct answers, ML algos usually try to minimize the # errors...

LOGISTIC REGRESSION - SIGMOID FUNCTION



$$z = W_0 + W_1 Studied + W_2 Slept$$

For a set of the regression parameters, predictions are made and then compared with labels

Cost Function is used to evaluate "Penalty"

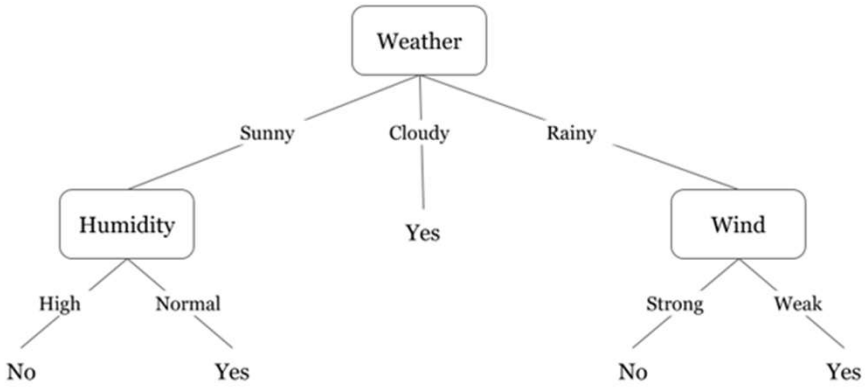
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \quad \text{if } y = 1$$

$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \quad \text{if } y = 0$$

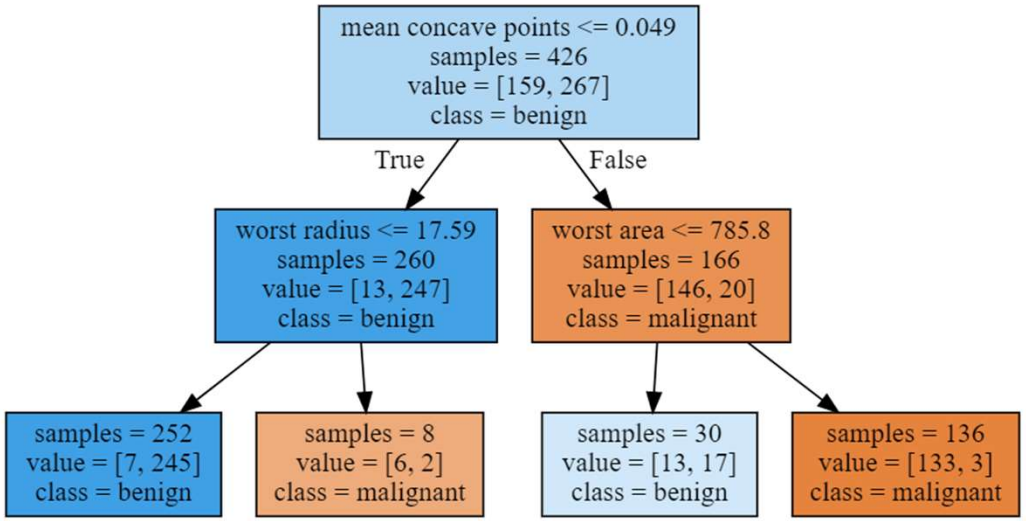
Jupyter Notebook on this Algorithm

DECISION TREES



- **Each level of the tree asks a "question" about a specific feature and divides the paths into several nodes**
- **number of levels is known as the tree depth**
- **Trees are constructed by trying different combinations of features**

DECISION TREES



Very simple to build and train

Very robust to different types of features

Quickly Overfit if hyperparameters are not controlled

Generalization Problems: what happens if feature outside possible ranges appears?

Online Learning problem: quickly change when re-trained

Jupyter Notebook on this Algorithm

NAIVE BAYES – WHAT ABOUT THOSE PESKY CATEGORICALS?

Let's say you have a table of registered voters in the USA, and the following features

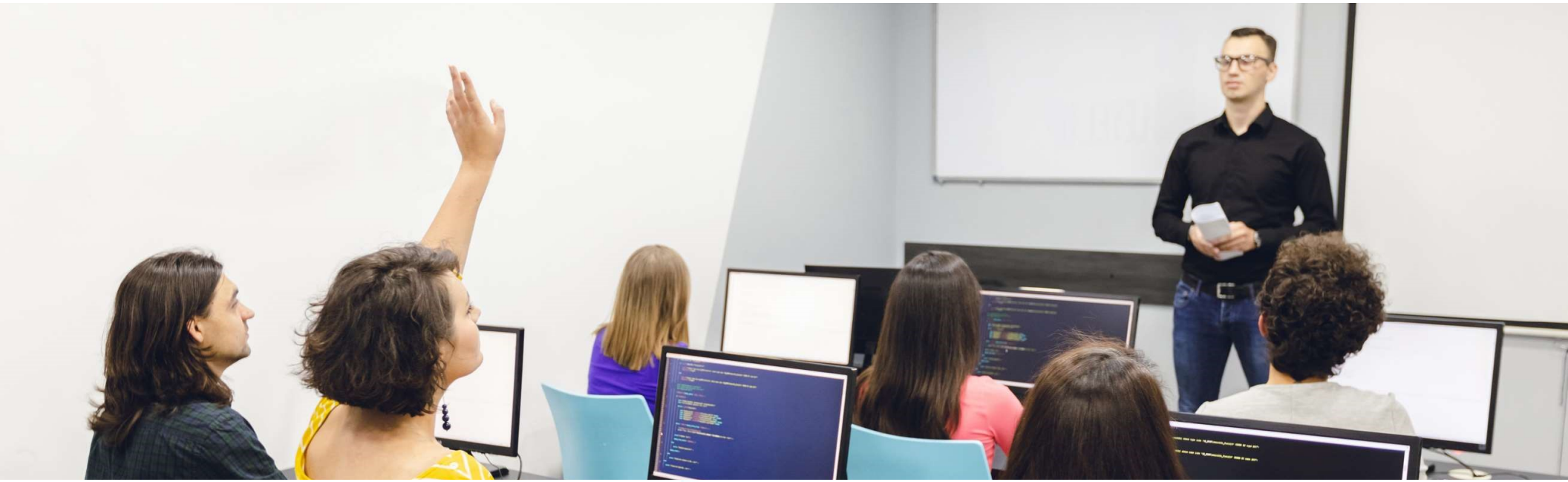
Registered	Age group	Wealth	Education	Location
Democrat	Young	Middle Class	Higher	Urban
Independent	Middle Aged	Middle Class	Post Grad	Urban
Republican	Middle Aged	Rich	Post Grad	Rural
Democrat	Old	Middle Class	Higher	Rural
Republican	Middle Aged	Middle Class	Higher	Rural
Republican	Old	Poor	Basic	Rural
Democrat	Middle Aged	Middle Class	Post Grad	Urban
Democrat	Young	Poor	Basic	Urban
Independent	Old	Middle Class	Higher	Rural

You are interested in, given a non-registered voter for which you know the features, to understand how likely that voter is to vote Democrat, Republican or Independent.

This is not an exercise that you can do with a regression or even using classic statistics since you have no numeric variables whatsoever, so what can you do in this case?

Let's suppose we get the following prospective voter features:

Age group	Wealth	Education	Location
Old	Middle Class	Post Grad	Rural



NOTE ON SENSITIVIY AND GENERALIZATIONS IN THE CLASSROOM DUE TO POLITICAL GENERALIZATIONS

NAIVE BAYES

We could try to fit this voter with the majority of the voters that have exactly these features

Age group	Wealth	Education	Location
Old	Middle Class	Post Grad	Rural

But this has 2 problems

- **We may never have observed this exact combination of characteristics**
- **We don't want a "one rule" for all Old, Middle Class, Post Grad, Rural people. Some combinations are certainly more indicative of party affiliation than others**

What we really would like is to be able to compare

$$P(party = D|Age = O, Wealth = MC, Edu = PG, Loc = R)$$
$$P(party = R|Age = O, Wealth = MC, Edu = PG, Loc = R)$$

NAIVE BAYES - back to reality, simplified

Let's try a simpler problem

$$P(party = D|Edu = PG)$$
$$P(party = R|Edu = PG)$$

$$\frac{\text{postgrad democrats}}{\text{total postgrads}}$$

In this case we could estimate this probability directly (see arrow) but let's try a different approach.
Remember Bayes Rule?

$$P(party = D|Edu = PG) = \frac{P(Edu = PG|party = D)P(party = D)}{P(Edu = PG)}$$

$$P(party = R|Edu = PG) = \frac{P(Edu = PG|party = D)P(party = R)}{P(Edu = PG)}$$

NAIVE BAYES - back to reality, simplified

Now, if we are only interested in which of these is larger, we don't care about the denominator

$$P(party = D|Edu = PG) = \frac{P(Edu = PG|party = D)P(party = D)}{\cancel{P(Edu = PG)}}$$

$$P(party = R|Edu = PG) = \frac{P(Edu = PG|party = R)P(party = R)}{\cancel{P(Edu = PG)}}$$

And we can estimate the right hand side parameters from our table… how?

The point here is that **we can compute the probability of the party given the features from the probability of the features given the party…**

NAIVE BAYES - back to the painful reality

But our problem is not this simple, let's add back the other variables

$$P(party = R | Age = O, Wealth = MC, Edu = PG, Loc = R)$$

What we will do is the same as in the simplified case: **compute the probability of the party given the features from the probability of the features given the party**

$$\frac{P(Age = O|party = R) * P(Wealth = MC|party = R) * P(Edu = PG|party = R) * P(LOC = R|party = R) * P(party = R)}{P(Age = O, Wealth = MC, Edu = PG, Loc = R)}$$

Technically, this requires us to assume that the features are independent from each other in the presence of the voting information, which we assume is more informative than any individual feature. This is the "Naive" assumption in Naive Bayes.

NAIVE BAYES - Don't freak out!

Now this formula looks daunting, but don't freak out yet

$$\frac{P(Age = O|party = R) * P(Wealth = MC|party = R) * P(Edu = PG|party = R) * P(LOC = R|party = R) * P(party = R)}{P(Age = O, Wealth = MC, Edu = PG, Loc = R)}$$

What are we really saying here?

- The denominator is irrelevant because for the democrats we will have the same denominator. Since we are only interested in answering who has the higher probability, R's or D's, we don't need to compute it.
- The numerator has a bunch of factors, but they all look the same: what is the probability of being old given that one is republican? What is the probability of being middle class, given that one is republican? Etc etc. We can estimate all of these from our table!

$$P(Age = O|party = R) \sim \frac{\text{old republicans}}{\text{total republicans}}$$

NAIVE BAYES - Don't freak out!

So in the end our rule

$$\frac{P(Age = O|party = R) * P(Wealth = MC|party = R) * P(Edu = PG|party = R) * P(LOC = R|party = R) * P(party = R)}{P(Age = O, Wealth = MC, Edu = PG, Loc = R)}$$

becomes

$$\frac{\text{old repubs}}{\text{total repubs}} * \frac{\text{wealthy repubs}}{\text{total repubs}} * \frac{\text{PG repubs}}{\text{total repubs}} * \frac{\text{rural repubs}}{\text{total repubs}} * \frac{\text{repubs}}{\text{total}}$$

versus

$$\frac{\text{old dems}}{\text{total dems}} * \frac{\text{wealthy dems}}{\text{total dems}} * \frac{\text{PG dems}}{\text{total dems}} * \frac{\text{rural dems}}{\text{total dems}} * \frac{\text{dems}}{\text{total}}$$

We just need to compare these quantities to decide which is more likely. Notice we ignore the denominator because it is equal for both the democratic and the republican case.

NAIVE BAYES – Who's this guy after all?

Registered	Age group	Wealth	Education	Location
Democrat	Young	Middle Class	Higher	Urban
Independent	Middle Aged	Middle Class	Post Grad	Urban
Republican	Middle Aged	Rich	Post Grad	Rural
Democrat	Old	Middle Class	Higher	Rural
Republican	Middle Aged	Middle Class	Higher	Rural
Republican	Old	Poor	Basic	Rural
Democrat	Middle Aged	Middle Class	Post Grad	Urban
Democrat	Young	Poor	Basic	Urban
Independent	Old	Middle Class	Higher	Rural

Age group	Wealth	Education	Location
Old	Middle Class	Post Grad	Rural

$$\frac{\text{old repubs}}{\text{total repubs}} * \frac{\text{wealthy repubs}}{\text{total repubs}} * \frac{\text{PG repubs}}{\text{total repubs}} * \frac{\text{rural repubs}}{\text{total repubs}} * \frac{\text{repubs}}{\text{total}}$$

$$\frac{1}{3} * \frac{1}{3} * \frac{1}{3} * \frac{3}{3} * \frac{3}{9} \sim 0.01234$$

$$\frac{\text{old dems}}{\text{total dems}} * \frac{\text{wealthy dems}}{\text{total dems}} * \frac{\text{PG dems}}{\text{total dems}} * \frac{\text{rural dems}}{\text{total dems}} * \frac{\text{dems}}{\text{total}}$$

$$\frac{1}{4} * \frac{3}{4} * \frac{1}{4} * \frac{1}{4} * \frac{4}{9} \sim 0.00521$$

Since 0.01234 > 0.00521, mystery guy is more likely to be a Republican than a Democrat

NAIVE BAYES - In short

So, if you have to decide between

$$P(A = a_i|B, C, D, E, ...)$$

For different a_i, you can instead compare

$$P(B|A = a_i) * P(C|A = a_i) * P(D|A = a_i) * ...$$

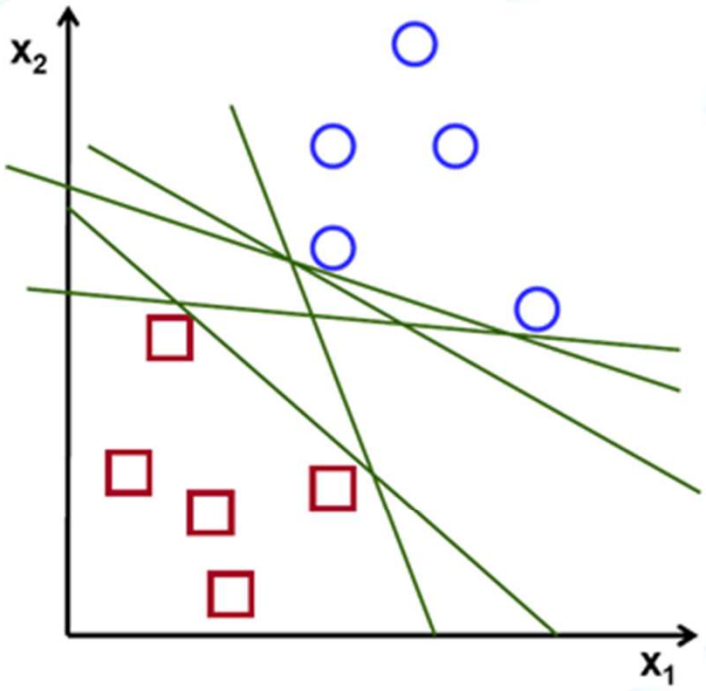
(which should be easy from your table of prior observations)

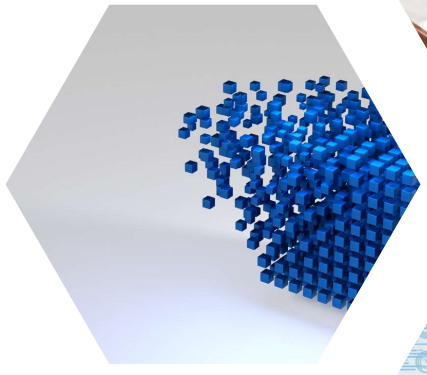
And take the a_i which makes this value largest.

NAIVE BAYES - Issues

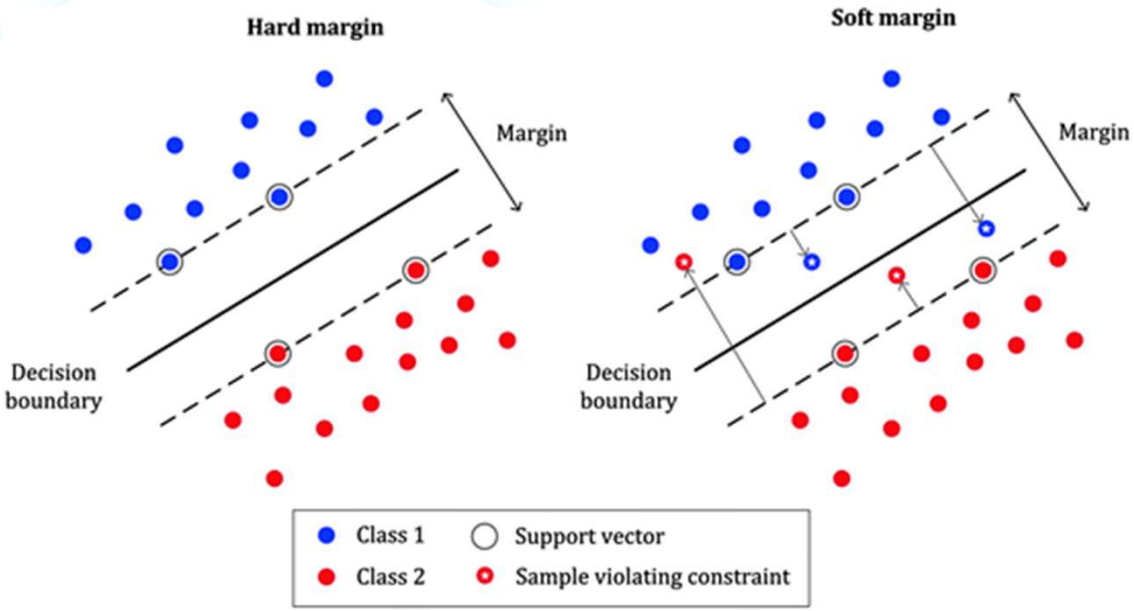
- The independence assumption: we essentially double count features that are erroneously assumed independent
- Zero-frequency problem: probability of said class automatically becomes 0 for said value. We can solve this by adding 1 to every single observation, but we need a somewhat large dataset to make this work.
- Conditioning: as we get more and more features, our numbers become very tiny and computers may have trouble doing math with

SUPPORT VECTOR MACHINE (SVM)





SUPPORT VECTOR MACHINE (SVM)



Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

MACHINE VECTORSUPPORT

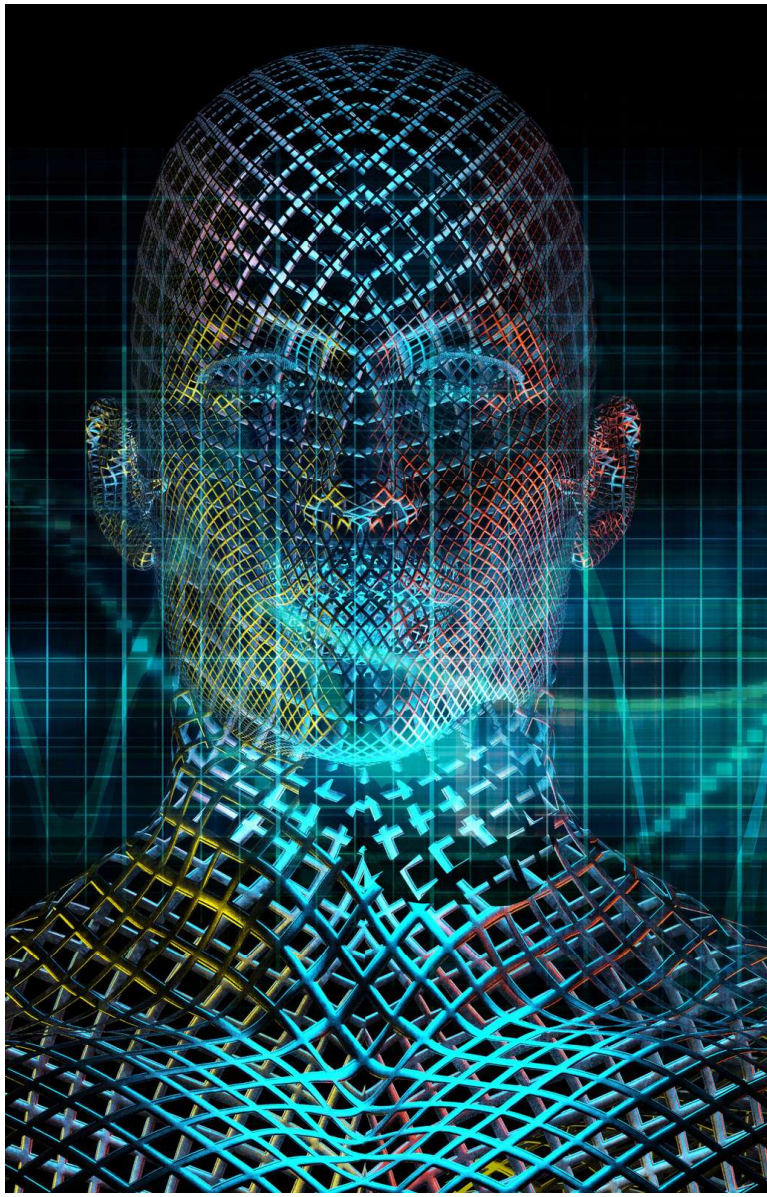
- Draws the line separating the two classes that maximises the distance
- between the data and the line.
- . Defined by support vectors, those data points that are closest to the dividing line

SVM

Can Model NonLinear Boundaries (if nonlinear version is used)

Unlikely to overfit

Memory intensive: is more successful on small dataset



IRON HACK

EVALUATION METRICS

PERFORMANCE EVALUATION METRICS - ACCURACY

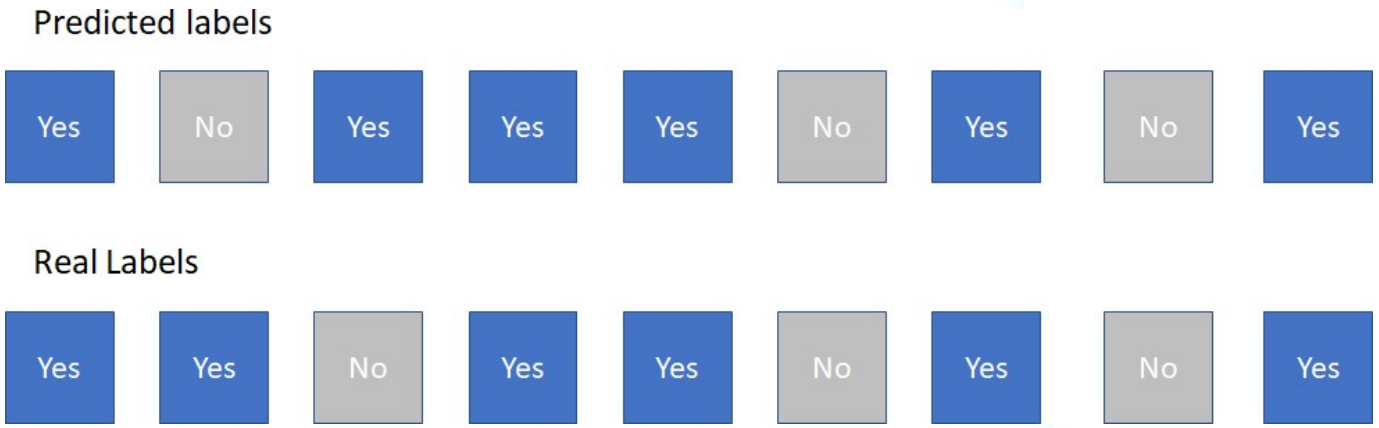
So far we have just been evaluating the performance of our algorithms based on accuracy

$$\frac{\#\text{correct Prediction}}{\text{Total \# of Predictions}}$$

PERFORMANCE EVALUATION METRICS - ACCURACY

So far we have just been evaluating the performance of our algorithms based on accuracy

$$\frac{\#\text{correct Prediction}}{\text{Total \# of Predictions}}$$



PERFORMANCE EVALUATION METRICS - ACCURACY

So far we have just been evaluating the performance of our algorithms based on accuracy

$$\frac{\#\text{correct Prediction}}{\text{Total \# of Predictions}}$$

PERFORMANCE EVALUATION METRICS - ACCURACY

So far we have just been evaluating the performance of our algorithms based on accuracy

#correct Prediction

Total # of Predictions

$$Accuracy = \frac{\#\ correct\ predictions}{\#\ total\ predictions} = \frac{6}{9} \sim 66\%$$

Predicted labels

Yes	No	Yes	Yes	Yes	No	Yes	No	Yes

Real Labels

Yes	Yes	No	Yes	Yes	No	Yes	No	No

PERFORMANCE EVALUATION METRICS - ACCURACY

How can accuracy be deceptive?

- **Imbalanced Datasets**

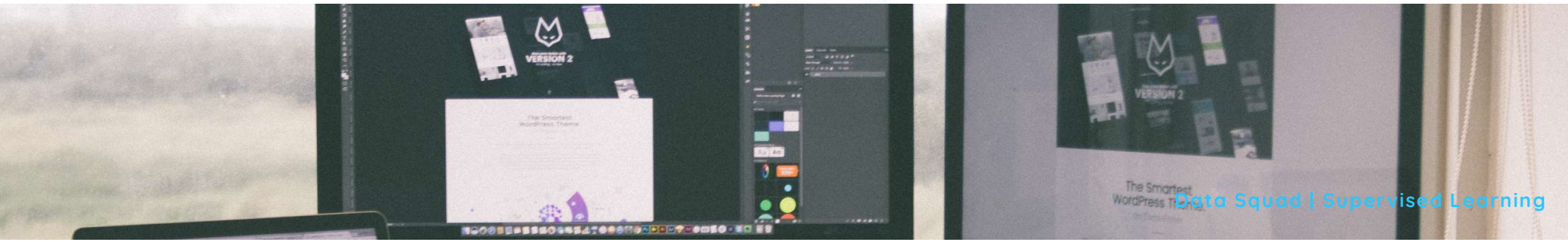
- **Importance of incorrect prediction of a certain class**

$$\frac{\#\text{correct Prediction}}{\text{Total \# of Predictions}}$$

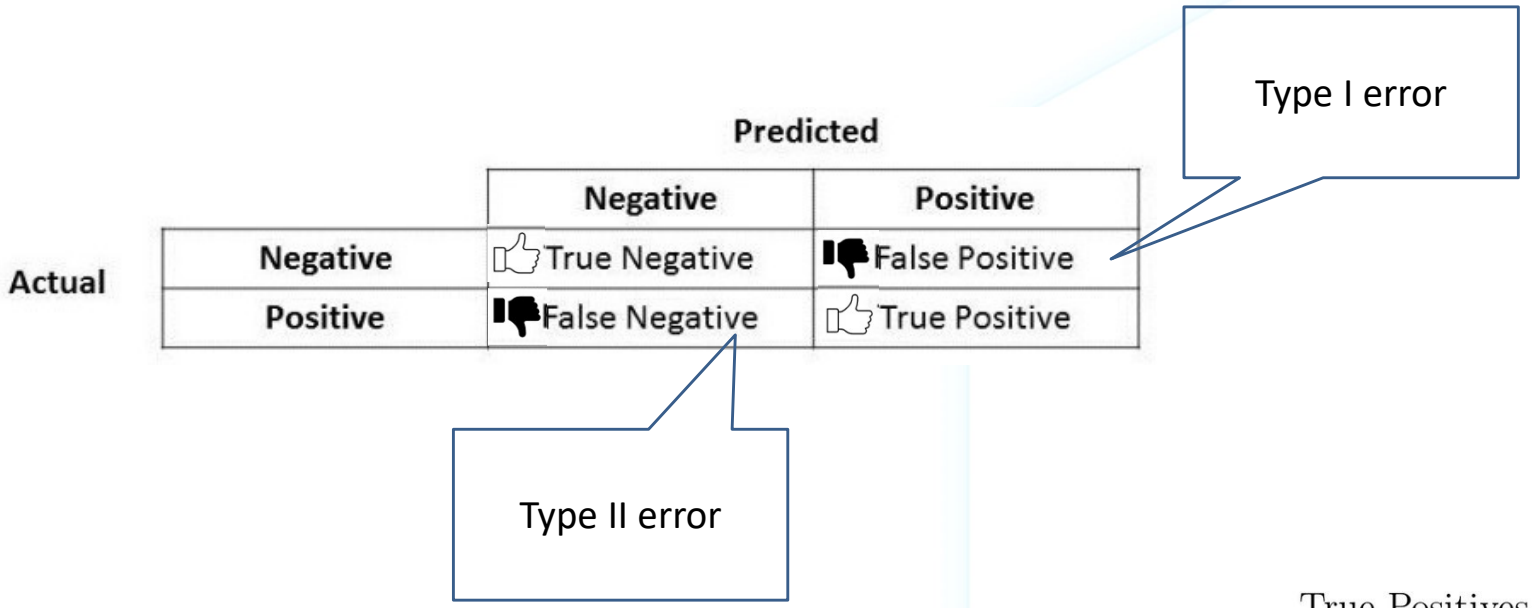
PERFORMANCE EVALUATION METRICS – CONFUSION MATRIX

		Predicted: Negative	Predicted: Positive
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All predictions}}$$



PERFORMANCE EVALUATION METRICS – CONFUSION MATRIX



$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All predictions}}$$

PERFORMANCE EVALUATION METRICS – CONFUSION MATRIX

		Predicted: Negative	Predicted: Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Type I error

Type II error

Type I Error

Type II Error

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All predictions}}$$

PERFORMANCE EVALUATION METRICS – CONFUSION MATRIX

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

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

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All predictions}}$$

PERFORMANCE EVALUATION METRICS - PRECISION & RECALL

Let's describe these two concepts through words & intuition

Precision of Class A:

From all the datapoints I predicted to be "A", how many were A?

Example: From a universe of 1000 people, we predicted 10 had coronavirus. In fact only 8 out of these 10 were actually infected.
Precision = 8/10 == 80%

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

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

PERFORMANCE EVALUATION METRICS - PRECISION & RECALL

Let'sdescribethesetwoconceptsthroughwords&intuition

Recall of Class A:

Fromallthedatapointsthattrulywere"A",howmanydidwepredict?

Example: From a universe of 1000 people, 100 are actually infected with coronavirus(wedonotknowthis,ofcourse)wecorrectlypredictedthat50 peoplehadthevirus
Recall=50/100==50%

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

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

PERFORMANCE EVALUATION METRICS - PRECISION & RECALL

Which one should we prioritize?

DEPENDS ON THE PROBLEM

However, we have to learn how to decide which one we want to prioritize: Predict if a patient has cancer

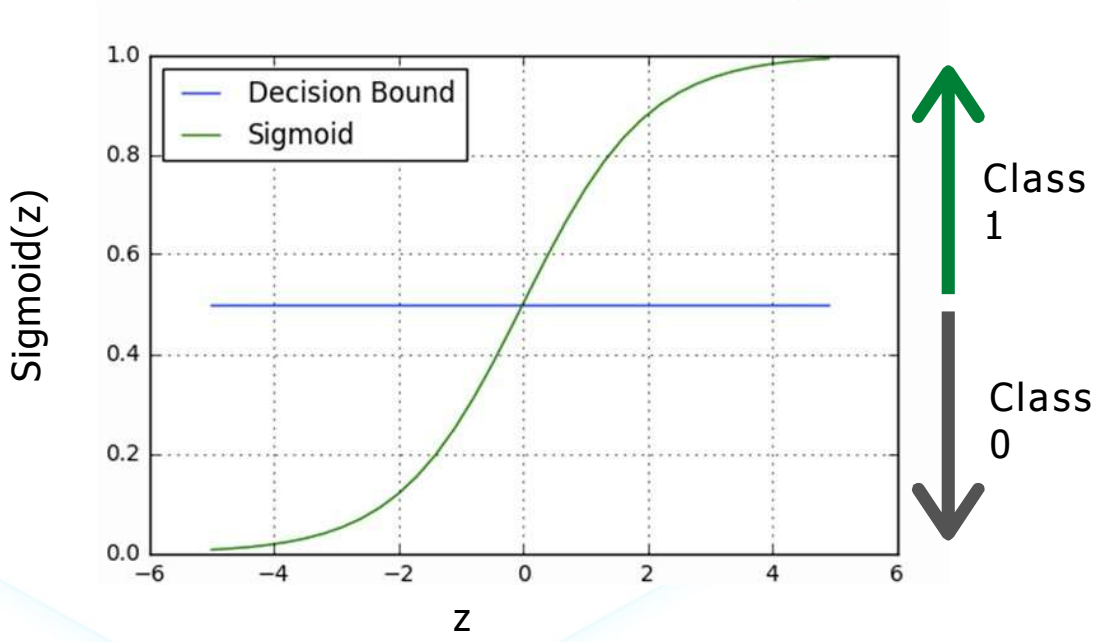
Brute Force Precision: only claim the patient has cancer if every single test comes up true multiple times in a row

Brute Force Recall: assume that a patient has cancer "to play it safe" before more tests come in

PERFORMANCE EVALUATION METRICS - PRECISION & RECALL

Which one should we prioritize?

PERFORMANCE EVALUATION METRICS - LOGISTIC REGRESSION



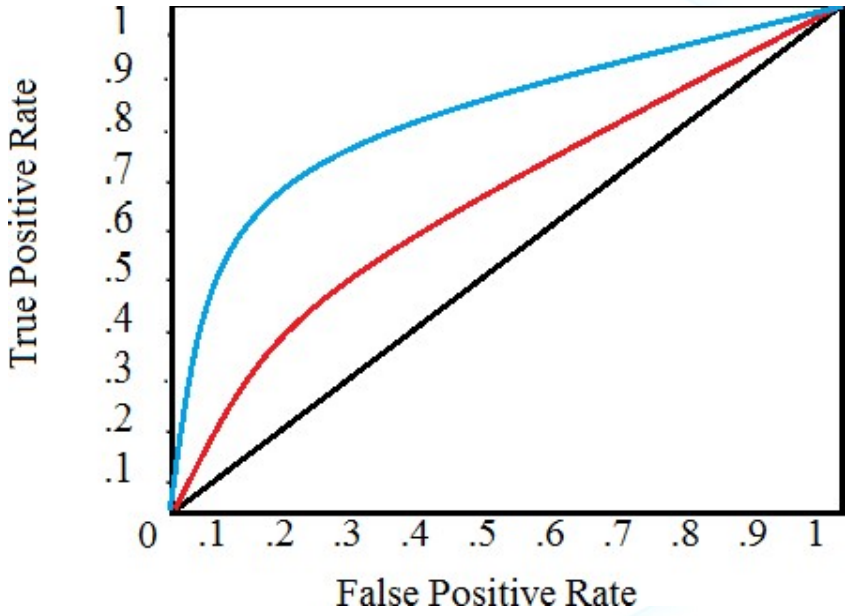
How do you relate the decision boundary to precision/recall in the logistic regression algorithm?

PERFORMANCE EVALUATION METRICS - F1 SCORE

A Trade-off between both such that the algorithm doesn't improve too much one of them at the expense of the other

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

PERFORMANCE EVALUATION METRICS - ROC CURVE



It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0. Put another way, it plots the false alarm rate versus the hit rate.

ANY QUESTIONS ?