Making Friends with Machine Learning

FROM GOOGLE CLOUD BY CASSIE KOZYRKOV

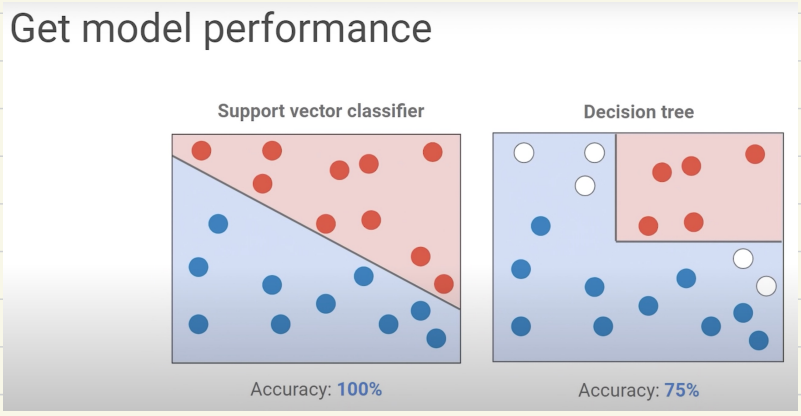
MFML - Part 1

What is ML?

EXECUTION OF MACHINES TO LEARN AND CREATE A "RECIPE" / CODE TO CONNECT AND EXPLAIN PATTERNS.

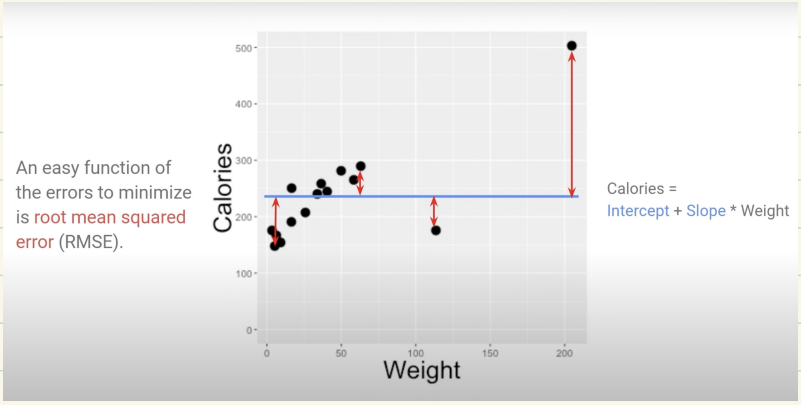
RECIPE == MODEL

Which model do we trust?



IN ORDER TO DECIDE WHICH MODEL YOU SHOULD PICK, YOU CAN COMPARE BOTH MODELS USING SOME METRIC, IN THIS CASE: ACCURACY

Linear regression

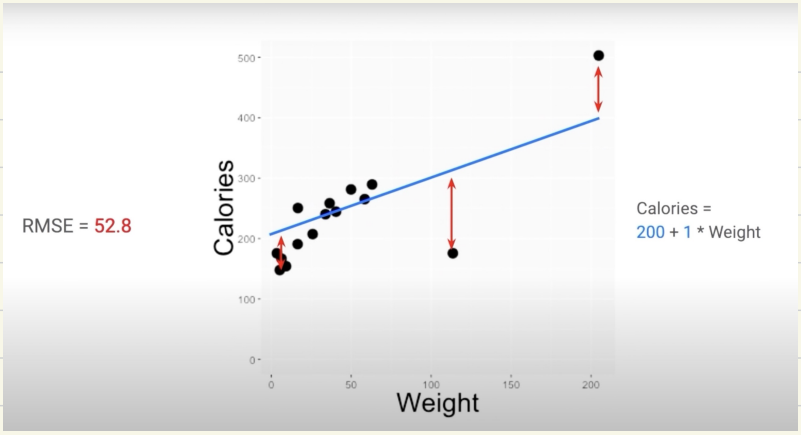


THE FORMULA FOR YOUR LINE WOULD BE:

CALORIES = INTERCEPT + SLOPE * WEIGHT

CURRENT LINE: CALORIES = 236.9 + 0 * WEIGHT

RMSE = 84.6



YOU COULD RANDOMLY PICK PARAMETERS OR

YOU THEN RUN AN OPTIMIZATION FUNCTION TO REDUCE RMSE.

IF YOU ADD % FAT AS A VARIABLE, THEN YOU'RE GOING TO HAVE A PLANE:

$$\text{CALORIES} = \text{INTERCEPT} + \underbrace{\text{SLOPE}_1 * \text{WEIGHT}}_{\text{WEIGHT PART}} + \underbrace{\text{SLOPE}_2 * \%\text{FAT}}_{\%\text{ FAT PART}}$$

IF YOU ADD EXTRA VARS, YOU'LL HAVE A HYPERPLANE.

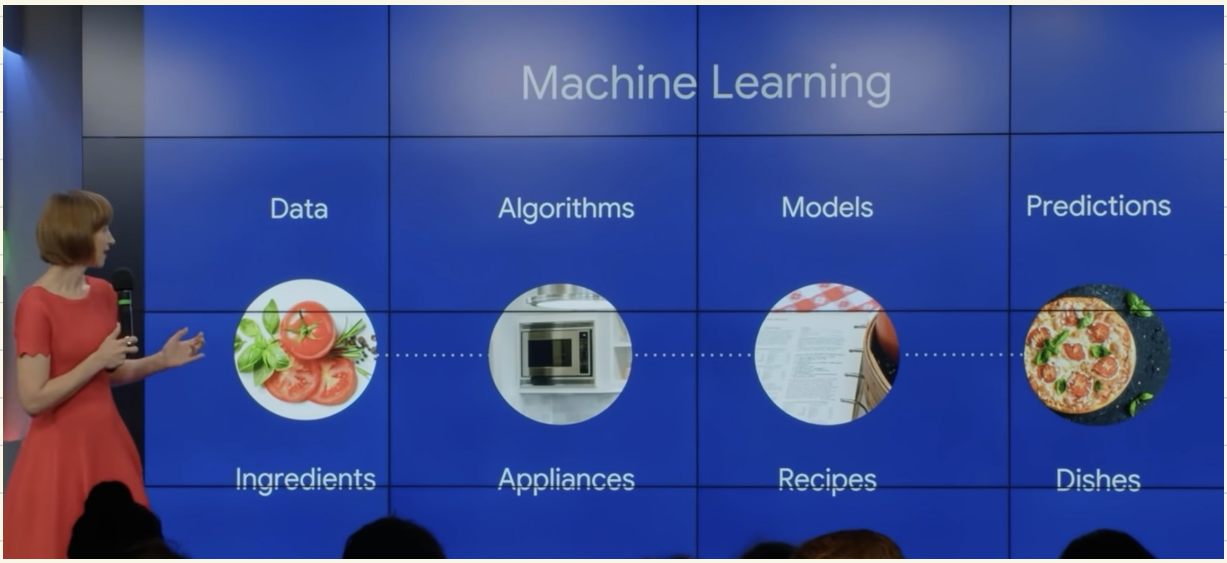
What is AI?

VISION
COMMUNICATION
PLAYING GAMES
→ TASKS →

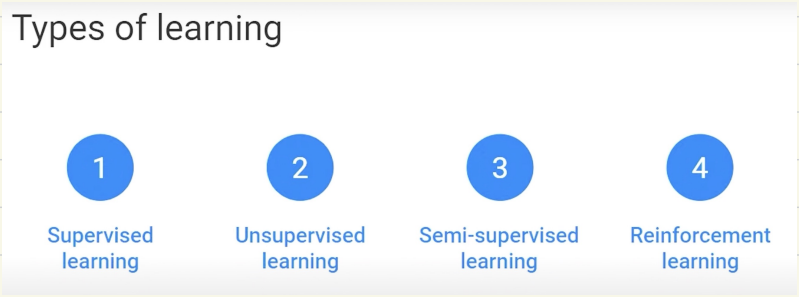
→ ? →

RAT OR NOT A RAT?

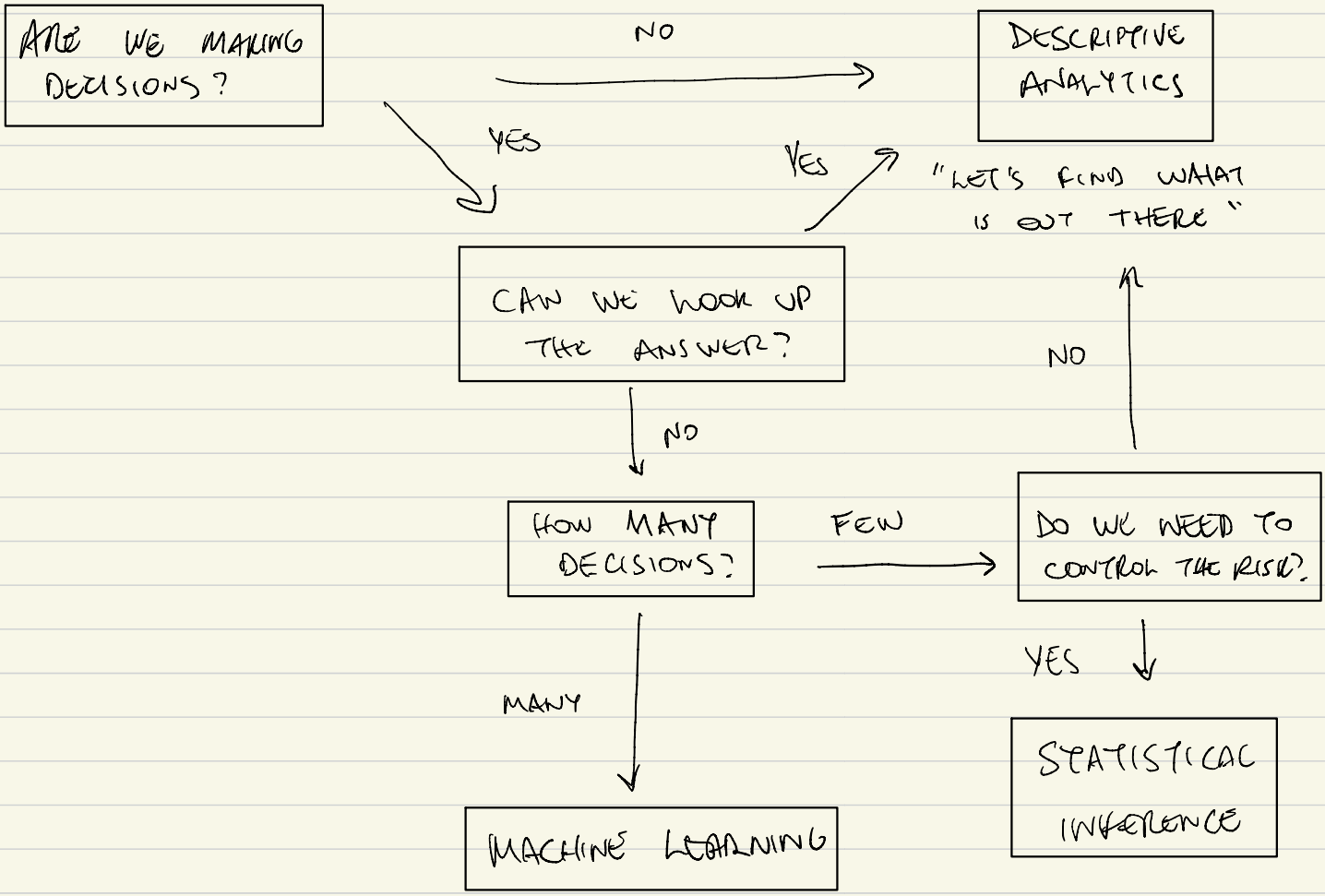
A GOOD ANALOGY TO COOKING WOULD BE:



MFML - Part 2

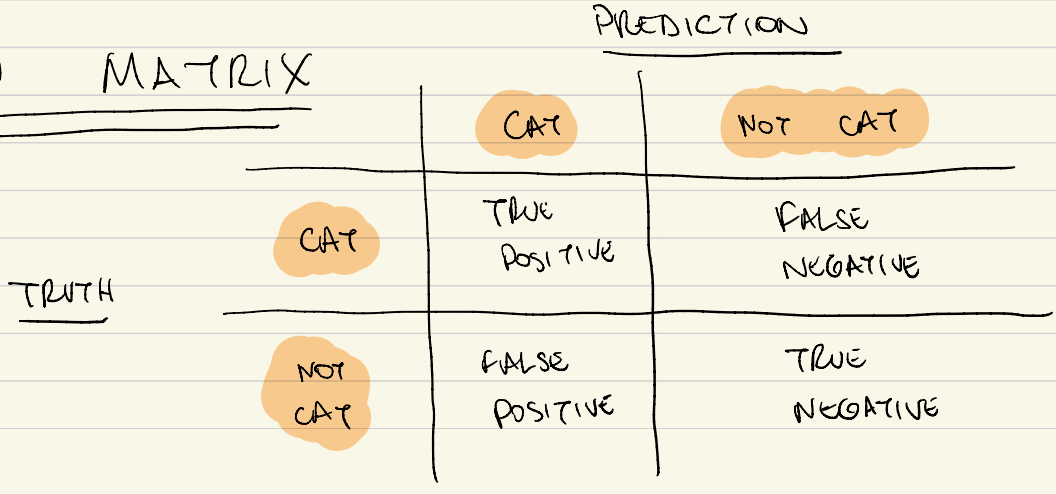


1. WHEN YOU HAVE A LABEL
2. NO LABEL, CREATE "GROUPS"
3. SOME LABEL, SOME NOT
4. SYSTEM TAKES A SEQ OF ACTIONS.



Performance metrics

CONFUSION MATRIX



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

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

"OF ALL PREDICTED AS CAT, HOW MANY WERE ACTUALLY CAT?"

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

"OF ALL TRUE CATS, HOW MANY WERE CLASSIFIED?"

Loss function

A LOSS FUNCTION IS THE ACTUAL FUNCTION YOUR ML ALGORITHM WILL OPTIMIZE.

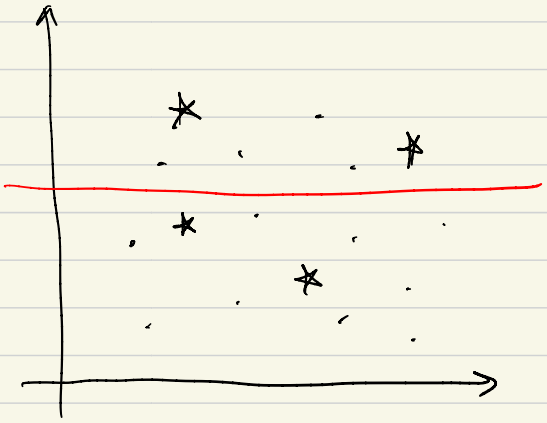
THAT WILL BE DIFFERENT FROM WHAT THE BUSINESS IS LOOKING AT (PERFORMANCE METRIC).

EXAMPLE:

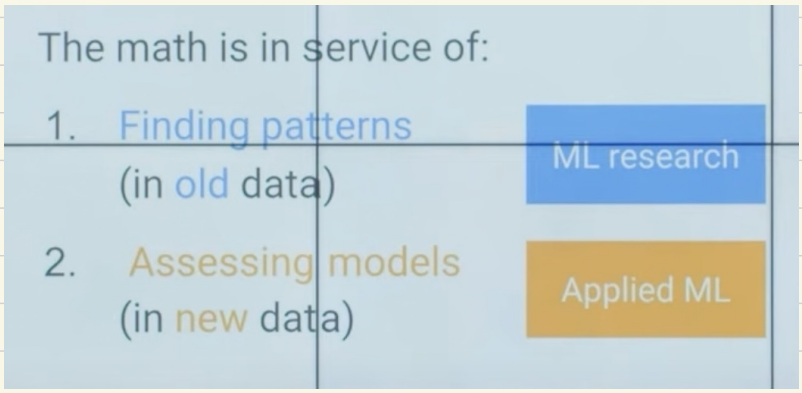
CROSS-ENTROPY LOSS : $H(\theta) = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot \log\left(p_\theta(y_i)\right) + (1 - y_i) \cdot \log\left(1 - p_\theta(y_i)\right)$

OVER / UNDER - FITTING :

OVERFITTING FINDING ONLY THAT PATTERN AND NOT GENERALIZING FOR OTHER.



UNDERFITTING NOT FINDING ANY REAL PATTERN.



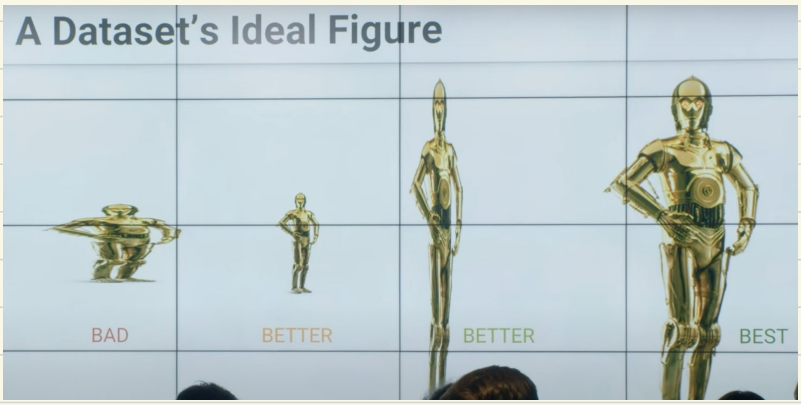
1. TRAINING YOUR MODEL
2. TESTING IN NEW DATA

MFML - Part 3

12 steps of ML

1 - FIND AN APPLICATION WHERE ML IS USEFUL / SET OBJECTIVE
2 - GET DATA
3 - SPLIT DATA (TRAINING, VALIDATION AND TEST DATASET)
4 - EXPLORE DATA (VIZ, SANITY CHECKS AND FEAT. ENG)
5 - GET TOOLS (ALGORITHMS ; CODE LIBS)
6 - TRAIN MODELS (CANDIDATES ; TRAINING PERF.)
7 - TUNE AND DEBUG YOUR MODELS (HYPERPARAMETERS TUNING)
8 - VALIDATE YOUR MODEL (SUCCESS ON FRESH DATA)
9 - TEST YOUR MODEL (DECISION PROCESS ; LIVE OR NOT ; PERFORMANCE ESTIMATE)
10 - BUILD YOUR ML SYSTEM (PRODUCTION-READY ; AUTOMATED RETRAINING)
11 - MAKE LAUNCH DECISION (SERVING MODEL TO USERS ; POLICY LAYER)
12 - MONITOR AND MAINTAIN YOUR SYSTEM (MONITORING & MAINTENANCE PLAN)

Step 6 - train models



THE BEST OPTION PROPOSED HAS PLENTY ROWS AND ENOUGH COLUMNS.

★ DON'T USE ALL VARIABLES AVAILABLE (COSTLY & NOT EFFICIENT)

★ SELECT TOP VARIABLES AND MOVE ON TO THE FUN PART : TRAINING !

→ SEND YOUR DATA THROUGH A BUNCH OF ALGORITHMS AND TINKER AS MUCH AS YOU LIKE.

FIT : PERFORMANCE ON OBJECTIVE .

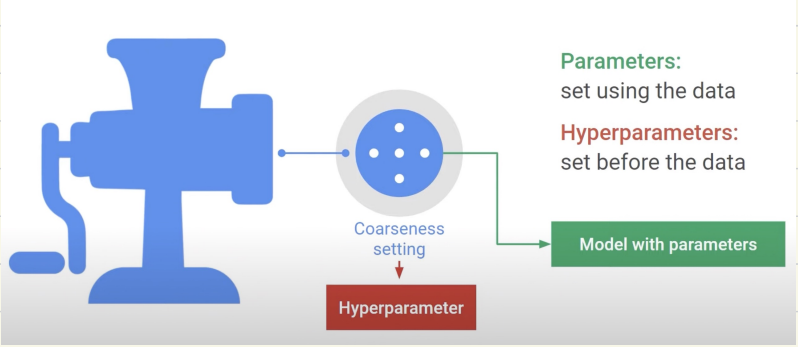
YOUR MISSION : MAKE IT FIT !

IF YOU SEE IT'S OVERFITTING:

1. TRY TO SIMPLIFY
2. REGULARIZATION
3. CHECK IF YOU ARE NOT USING DATA FROM THE FUTURE

Step 7 - hyperparameter

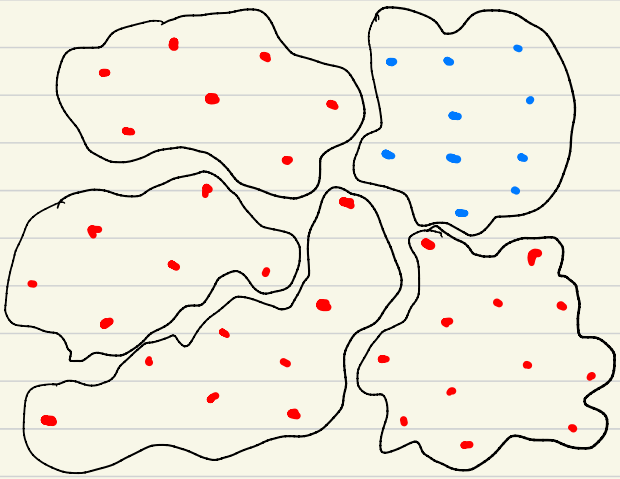
AFTER TRAINING YOU CAN DO SOME HYPERPARAMETER TUNING:



FIRST CHOOSE WHICH ALGORITHM YOU WANT, SECOND, YOU TRY DIFFERENT HYPERPARAMETERS AND STORE THEIR METRICS.

CROSS VALIDATION IS USUALLY USED TO CALCULATE HYPERPARAMETERS.

K-FOLD CROSS VALIDATION



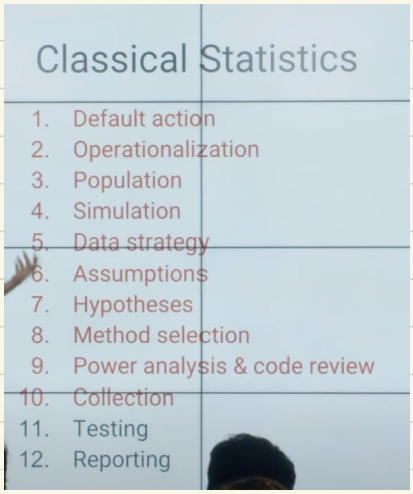
$K = 5 \Rightarrow$ 4 FOLDS FOR TRAINING
1 FOLD FOR VALIDATION

IMPORTANT NOTE: DON'T DEBUG WITH YOUR VALIDATION DATASET.

MACHINE LEARNING STORY

DATA:
DEFAULT ACTION:
NULL HYPOTHESIS:
ALTERNATIVE HYPOTHESIS:
ALTERNATIVE ACTION:

TEST DATASET
STOP PROJECT
MODEL SUCKS
MODEL WORKS
GO TO STEP 10



P-VALUE = PROBABILITY OF OBTAINING A SAMPLE AT LEAST AS EXTREME AS THE ONE WE JUST OBSERVED GIVEN THAT THE NULL HYPOTHESIS IS TRUE.

IF P-VALUE IS TOO SMALL, THEN H_0 IS REJECTED.

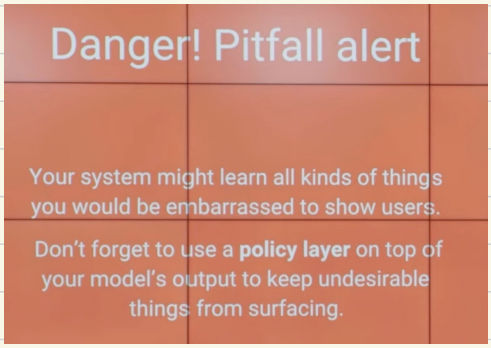
ANALOGY WITH CRIME: A P-VALUE IS THE PROBABILITY OF FINDING AT LEAST AS MUCH DAMNING EVIDENCE IF THE PERSON IS ACTUALLY INNOCENT.

Training-serving Skew

SOMETIMES YOUR TRAINING-VALIDATION-TEST DATA SIMPLY ISN'T LIKE YOUR SERVING DATA. THIS MEANS YOUR MODEL WORKS... BUT ONLY IN A DIFFERENT WORLD.

THIS IS YOUR OTHER NIGHTMARE! (1ST IS OVERFITTING)

Policy Layer



HAVE A POLICY LAYER IN PLACE IN CASE THINGS GO BAD AND YOU NEED TO TAKE YOUR ML MODEL OUT.

WHAT IS THE ALTERNATIVE IN CASE YOUR MODEL BREAKS?

MFML - Part 4

K-Means

FOR CLUSTERING WHEN THERE ARE NO LABELS.

K STANDS FOR # OF CLUSTERS YOU WOULD LIKE HAVING.

1. FIRST CENTROIDS ARE CHOSEN RANDOMLY AND CONTAIN MEMBERS
2. SECOND THE CENTROIDS ARE RECOMPUTED - CLOSER POINTS ARE GROUPED IN THAT CLUSTER
3. REPEAT 2. UNTILL IT CONVERGES

K-NN

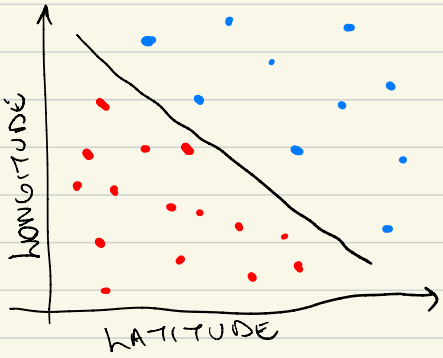
* NOTHING IN COMMON WITH K-MEANS.

REGRESSION TECHNIQUE THAT PREDICTS A LABEL OF A RECORD BASED ON K NEAREST RECORDS, THAT CONTAINS A LABEL.

K-NN := K NEAREST NEIGHBORS

SVM

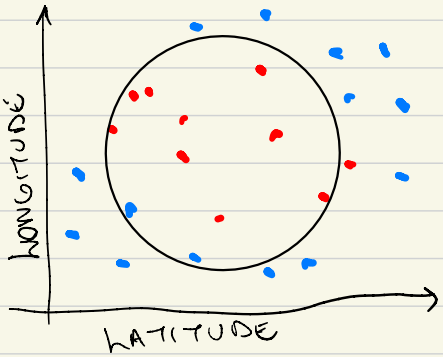
IT'S ABOUT BUILDING WALLS IN YOUR DATA.

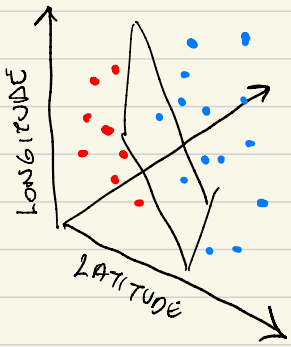


THE GOAL IS TO FIND THE HYPERPLANE THAT BEST SEPARATE THOSE TWO LABELS.

SVM TRIES TO MAXIMIZE THE MARGIN IN THE SURROUNDINGS OF THE PLANE.

SVM := SUPPORT VECTOR CLASSIFIER





WHAT SVM DOES IS TO ADD ONE EXTRA DIMENSION WHEN THERE IS THE NEED FOR FLEXIBLE BOUNDARY, NOT JUST LINEAR.

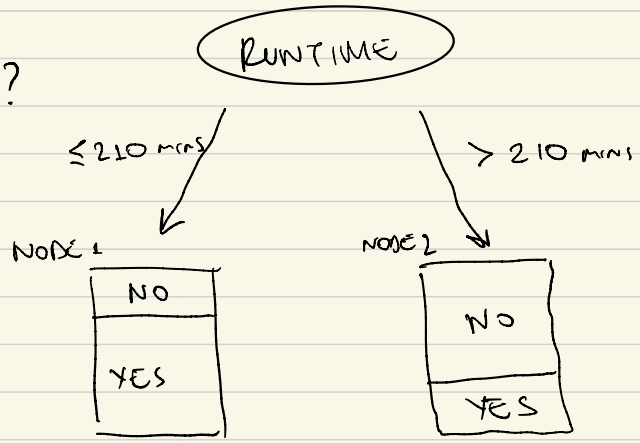
* USES A KERNEL!

Tree-based methods

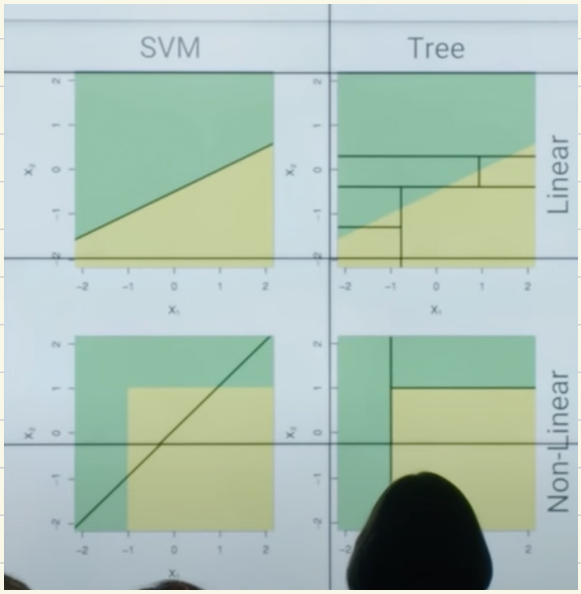
IT'S BASICALLY A BUNCH OF "IF THIS, THEN THAT" RULES FOR YOUR DATA.

EXAMPLE: DID YOU LIKE THE MOVIE?

TARGET FEATURE = LIKED THE MOVIE



* THE RESULTING RECIPES ARE EASY TO DESCRIBE TO HUMANS.



VISUALLY, YOU'RE CREATING BLOCKS OR TILES.

Bagging

1. RANDOMLY SELECTS SEVERAL SUBSAMPLES OF YOUR DATA
2. MAKE A TREE PER COLLECTION
3. LET EACH TREE VOTE ON YOUR NEW INSTANCE

RANDOM FORESTS

BAGGING

+

FOR EACH TREE, RANDOMLY PICK A DIFFERENT SUBSET OF FEATURES

Ensemble

BUILD LOTS OF TREES => LOTS OF DIFFERENT MODELS

THEN, LET EACH ONE VOTE ON A NEW INSTANCE.

Naive Bayes

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

OR

$$P(\text{LABEL} \mid \text{EVIDENCE}) = \frac{P(\text{EVIDENCE} \mid \text{LABEL}) \cdot P(\text{LABEL})}{P(\text{EVIDENCE})}$$

NAIVE BAYES CLASSIFIER ASSUMES FEATURES ARE ALL INDEPENDENT.

★ JUST COUNT THINGS SEPARATELY

THEREFORE { P(LABEL | EVIDENCE) = 75%.
P(NOT LABEL | EVIDENCE) = 25%.

Regression

REGRESSION REFERS TO FITTING MODELS LIKE:

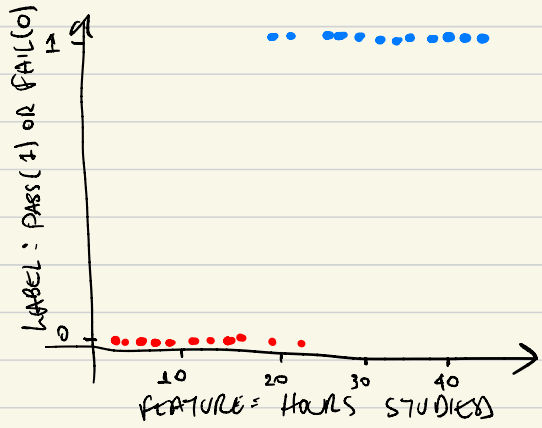
OUTPUT = PARAMETER_1 * FEATURE_1 +
PARAMETER_2 * FEATURE_2 +
PARAMETER_3 * FEATURE_3 + ...

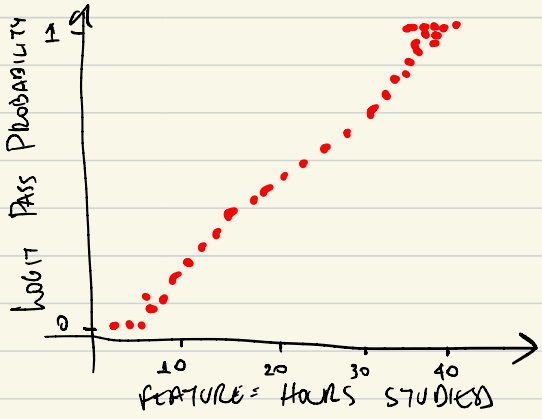
IT'S A VERY OLD METHODOLOGY.

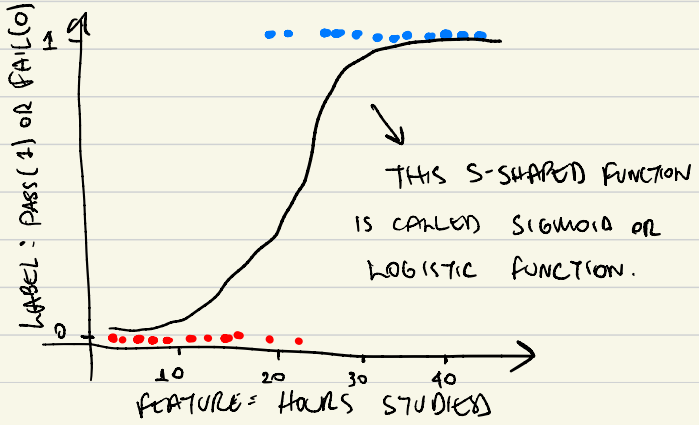
ONE OF THE MOST FAMOUS TYPES OF REGRESSION IS THE

LOGISTIC REGRESSION

IT'S GREAT FOR BINARY PROBLEMS, EXAMPLE:







THIS S-SHAPED FUNCTION IS CALLED SIGMOID OR LOGISTIC FUNCTION.

NOW WE FIT:

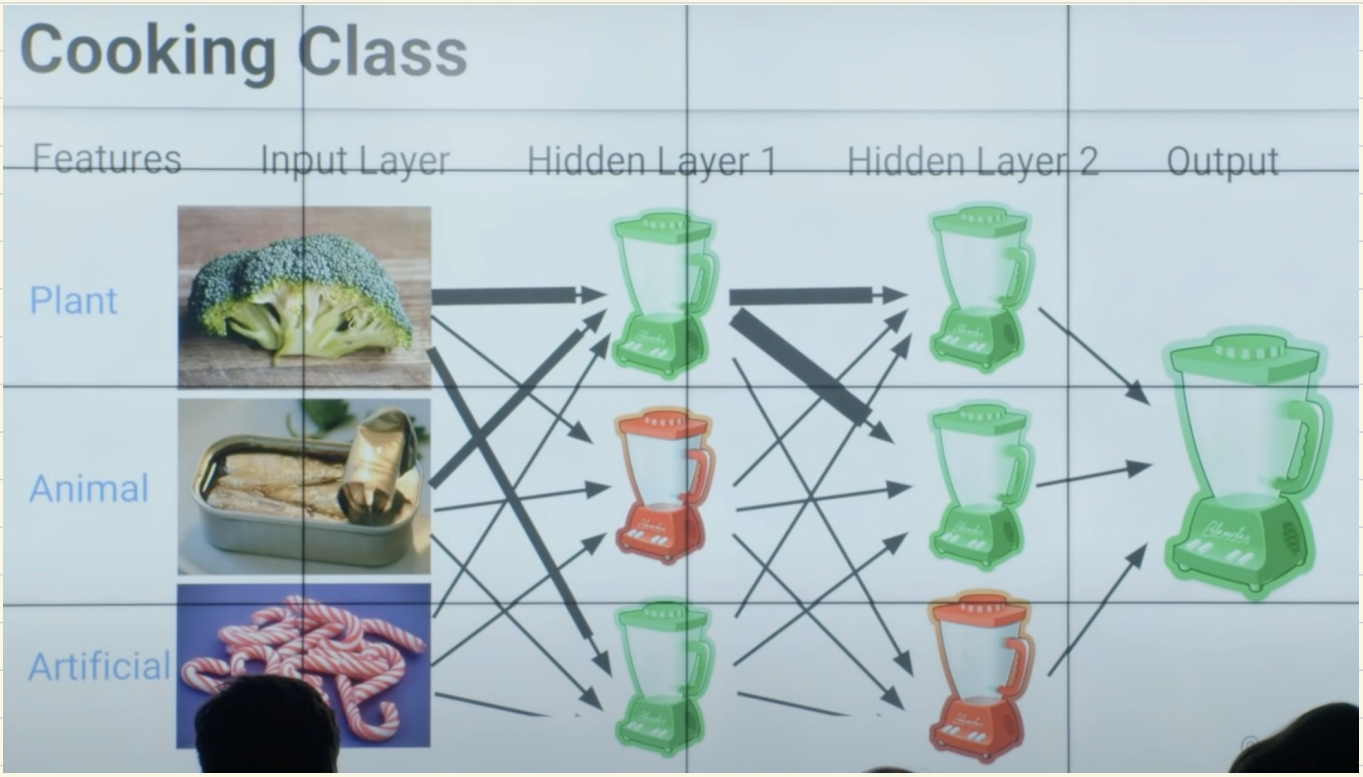
$$\text{LOGIT}(P) = \text{INTERCEPT} + \text{SLOPE} * X$$

$$P = \frac{e^{\text{LOGIT}(P)}}{1 + e^{\text{LOGIT}(P)}}$$

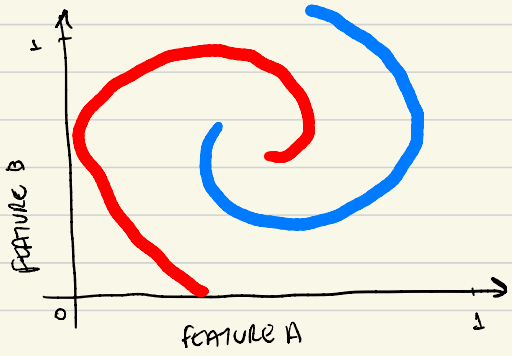
Neural Networks / Deep Learning

DEEP LEARNING IS JUST NN WITH MORE THAN ONE LAYER, ALSO KNOWN AS DEEP NN.

NN IS JUST LAYERS OVER LAYERS WITH DATA TRANSFORMATION.



IMAGINE YOU'VE A PROBLEM LIKE THIS ONE:



- NO K-NN, TREE BASED OR REGRESSION ALGORITHM WILL BE ABLE TO SPLIT THIS PATTERN.
- NN WILL DO SEVERAL TRANSFORMATIONS AND THIS COULD BE SOLVED.
- ALL NN TRANSFORMATIONS ALLOW THE ALGORITHM EXPLOIT COMPLEX STRUCTURES.

THE TRAINING OF NN:

1. PICK RANDOM STARTING WEIGHTS
2. FORWARD PROPAGATION
3. BACK PROPAGATION

PROS: COMPLEX TRANSFORMATIONS, i.e., BEST AT FITTING!

CONS: COSTLY, HARD TO DEBUG AND HARD TO INTERPRET. BEST AT OVERFITTING!