Naive Bayes Model

relies on the Bayes theorem

$$p(y|x) = \frac{P(x|y) \times P(y)}{p(x)}$$

$$p(y|w_1 \dots w_n) = \frac{P(y) \times \prod_{i=0}^{n} p(w_i|y)}{P(y) \times \prod_{i=0}^{n} p(w_i|y) + P(\neg y) \times \prod_{i=0}^{n} p(w_i|\neg y)}$$

Prior probability

likelihood of being spam

Frequency

Can be done simply by counting the words.

But a better way is by doing

TF-IDF. → Inverse Doc. Frequency: Measures the weight of a word based on its corpus

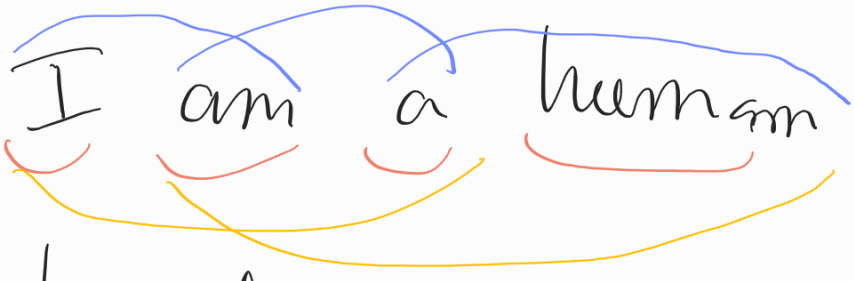
↓

Term Frequency

Counting words

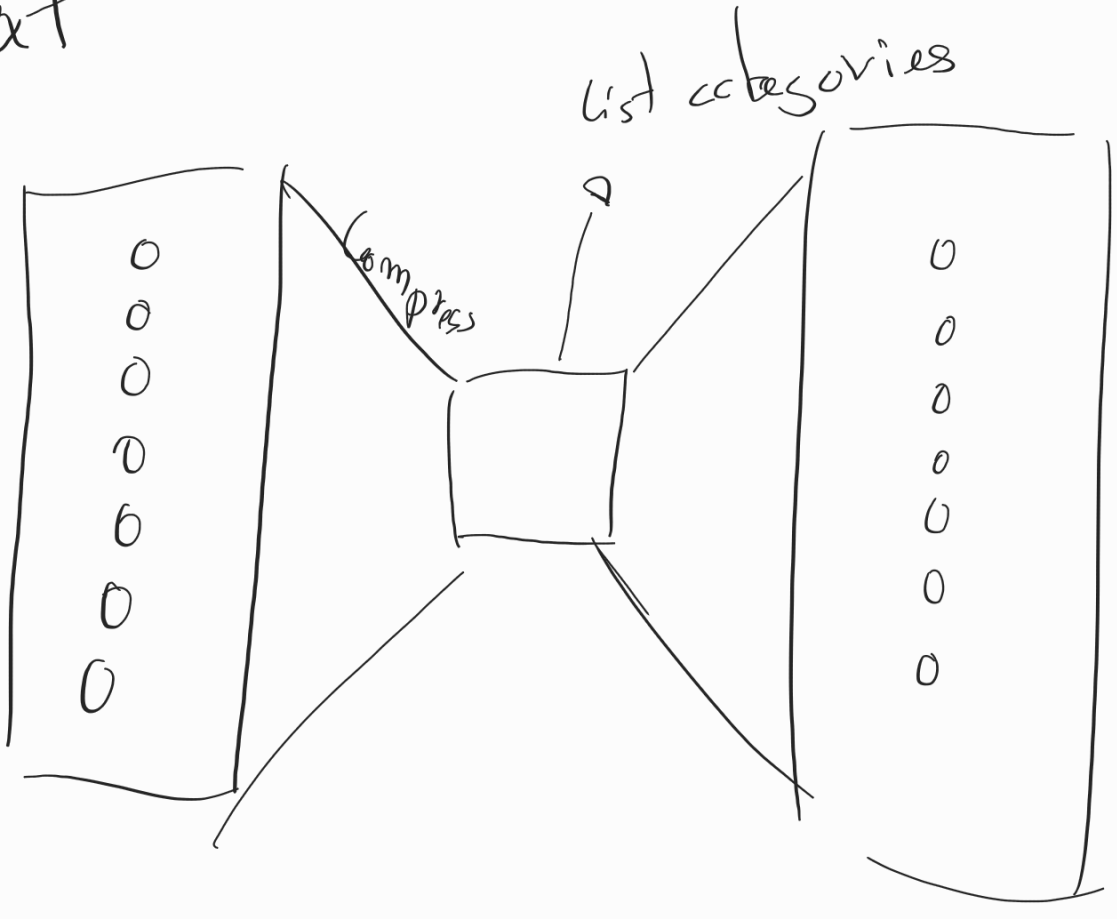
TF-IDF = TF × IDF

n-gram

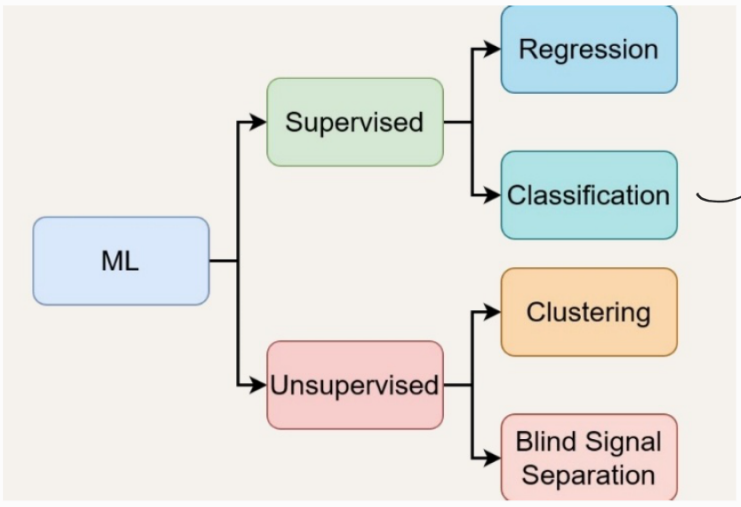


Units of words

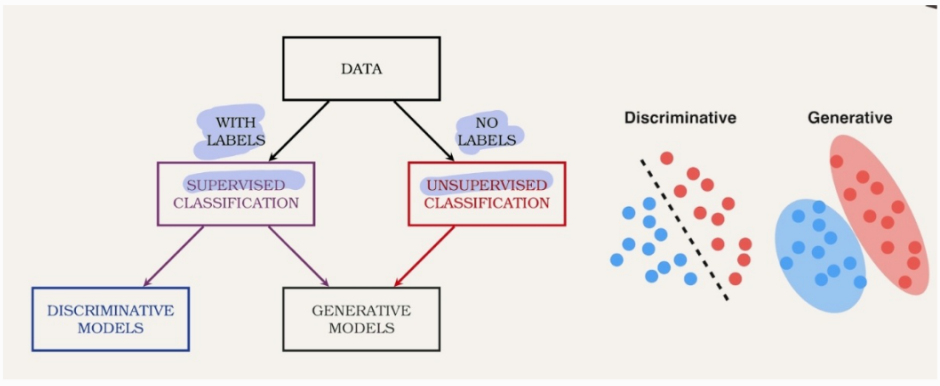
Fast text



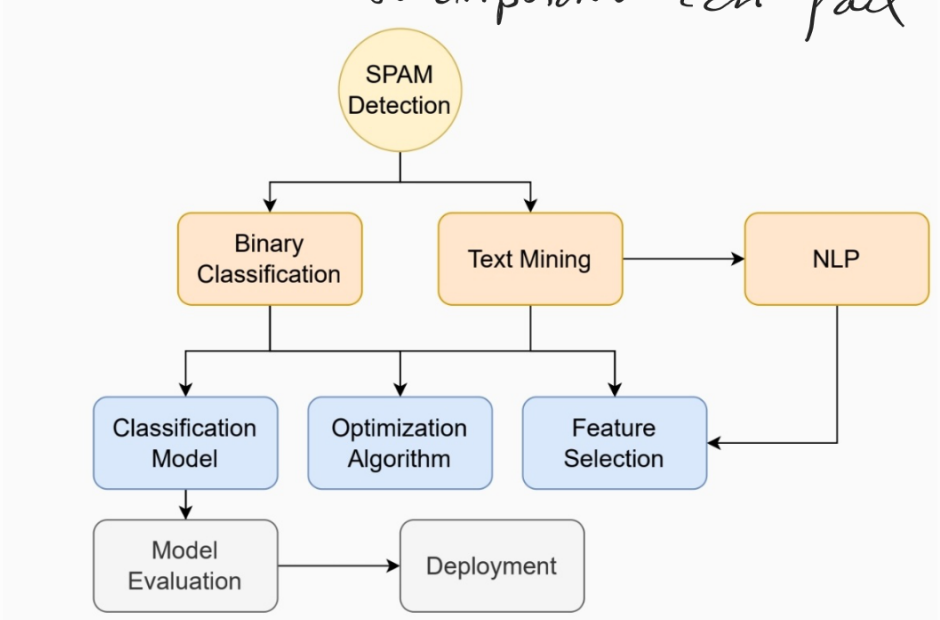
ML → improve it's own performance
→ programmed to learn through trial and error



binary classification



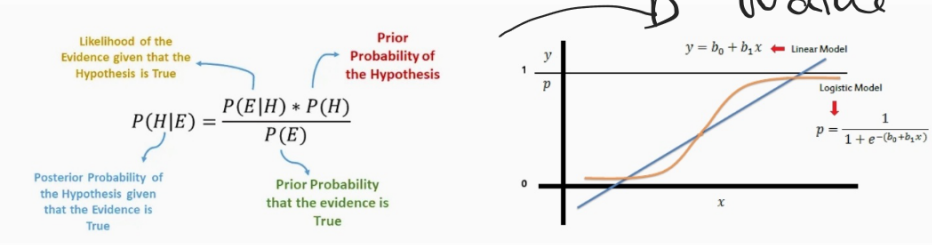
Limitations → biased assumptions exist
↳ simplification of reality
↳ Assumptions can fail



Binary classification → 1 or 0 → spam or ham
→ task of classifying the elements of a set into 2 groups

Text mining → make high quality info from text
↓ → uses bow Model (bag of words)

NLP → Method to give computers ability to understand text → Naive Bayes



$$acc = \frac{TP + TN}{(TP + TN + FP + FN)}$$

bow vs word embeddings → ordered
→ similar words close in the embedding space
unseen words are connected more easily

bow
↓ struggle on finding context, bad semantics
↓ no relation btw words

NB
text-mining
tokenization → word → token of → sentence
sentence → token of → paragraph

Lemmatization → rocks → rock
better → good
simplify in word to is "base"

term frequency → count word, more effective TF-IDF

↳ How frequen word is on a document

TF-IDF → how relevant a word is on the doc

↳ How many times appears on the doc +
the inverse doc freq → which is how rare or
common a word is, closer to 0 the more
common

i.e. 0.1 common → 0
never hits 0 or 1 I think

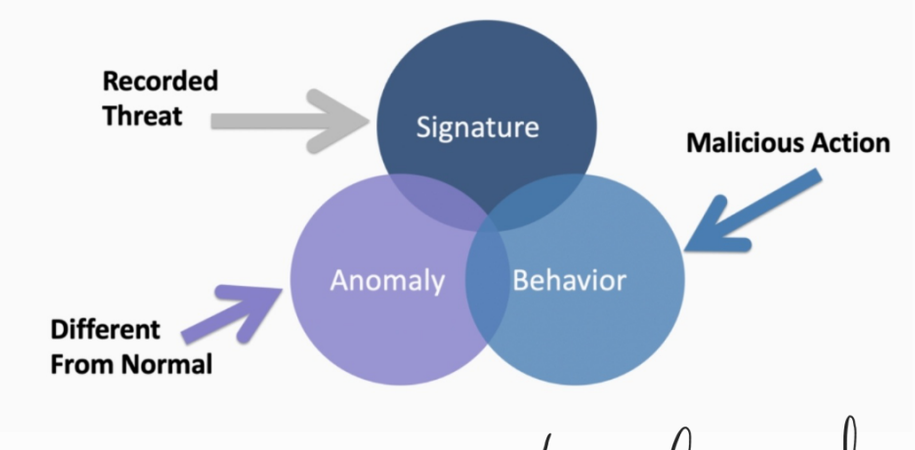
Particle swarm → optimization algorithm
→ find min and max

Gradient descendant → opt algs
→ find min with derived func

Linear Regression →

↑

Logistic Regression → Logict curve is the invers



Signature → compare hashes, of virus for example to find similar ones

Behaviour detection → malicious behaviour if similarities are found alarms so on

Anomaly Detection → zero days, threat intelligence
↳ outliers, find data that differ the norm
↳ fraud ↳ attacks ↳ noise ↳ data errors

Point anomaly → far from the rest

contextual → abnormality is context specific

collective anomaly → Group of outliers

clustering → find a structure with meaning

↳ Density based – clusters as a dense region

↳ Hierarchical Based → tree type, new clusters formed using old ones

↳ Partioning

BSS → Blind Signal Separation, separation of signals in a set of mixed signals

PCA → principal components, makes indexes

ICA → Independent // , // //

Auto encoders → Giving normal data it can recreate it if the data is anomalous it will

Have a hard time Reading, making it cumbersome

Missing values

- Replace "" with the mean of the collumn
  - Can only be done on numerical values
- Imputer

Encode

- Hot encoder
- Ordinal encoder

Feature scaling

Ocb the is 1000 vs 5 it will be more weigh on 1000 so we scale then so it doesn't give importance to one over the other feature.

Confusion Matrix

	1	0
1	TP	FP
0	FN	TN

Accuracy

Guesses was correct

Precission ⟶ good when FP high

Acc of positive predictions → TP/(TP+FP)

Recall ⟶ good when FN high

$TP/(TP+FN)$

F_1 - score

Mix of recall and Precision

MCC from -1 to 1 and takes in consideration all of 4 squares of Conf matrix

Roc grafic of TP on hps understand the TP on FP with diff thresholds.

MCNemar

Per comparar models

	2c	2w
1c	A	B
1w	C	D

Pe → Encapsulates data necessary for Windows to run the app, including dynamic libraries, API...

Feature Importance

Logistic Regression (LR)

Random forest Feature (RFF)

Decision tree (DT) → uses CART

Permutation Feature

↳ Feature Importance

↳ scrambles a column alone, done to all

↳ repeats 3, 5, 10 or more times