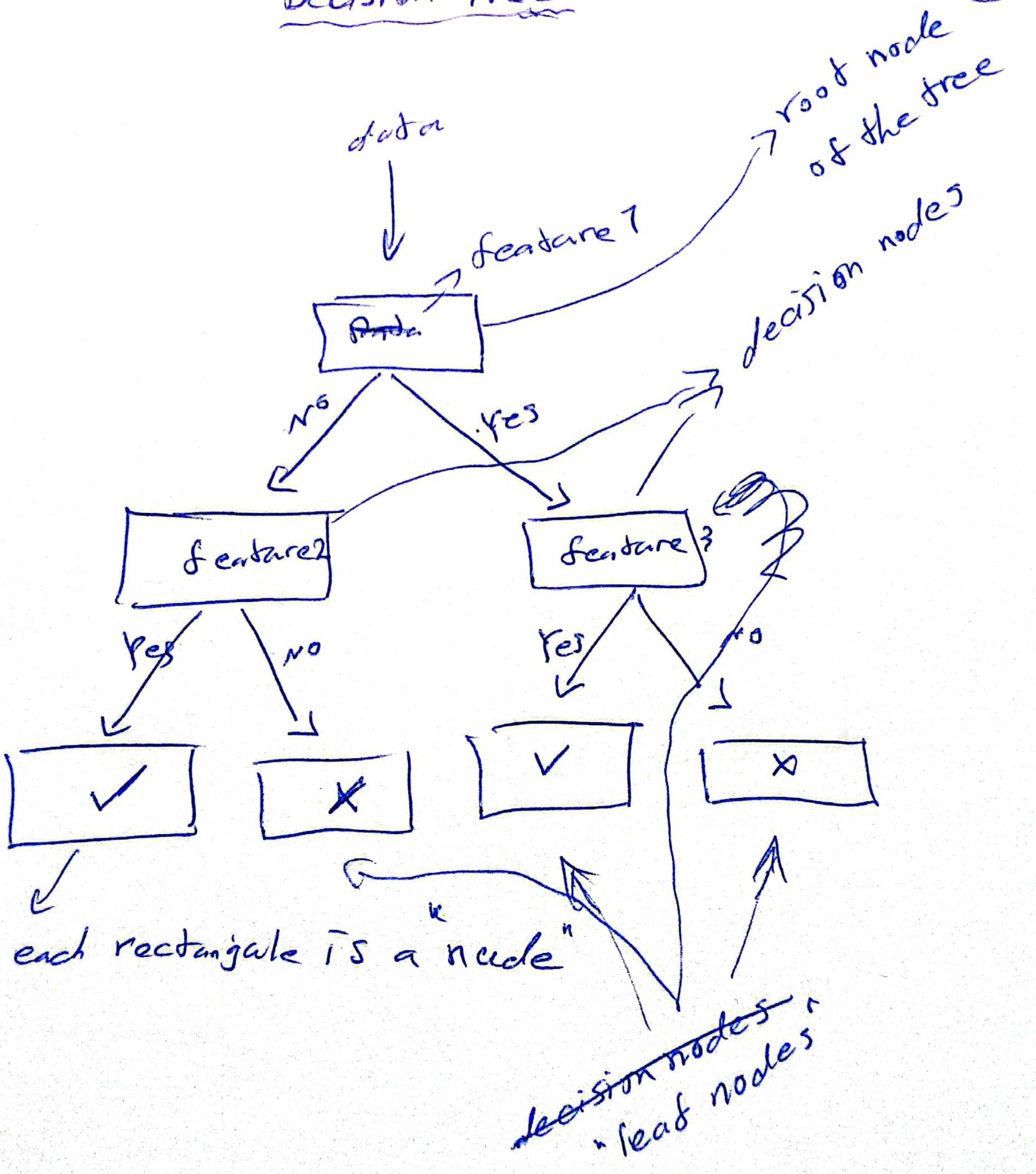
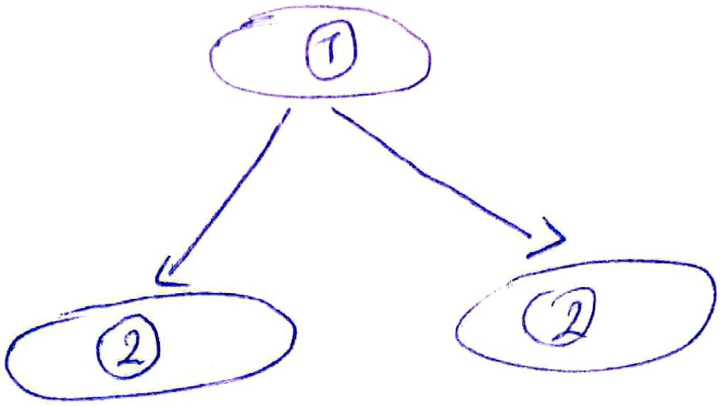
"Decision Tree"



learning Process

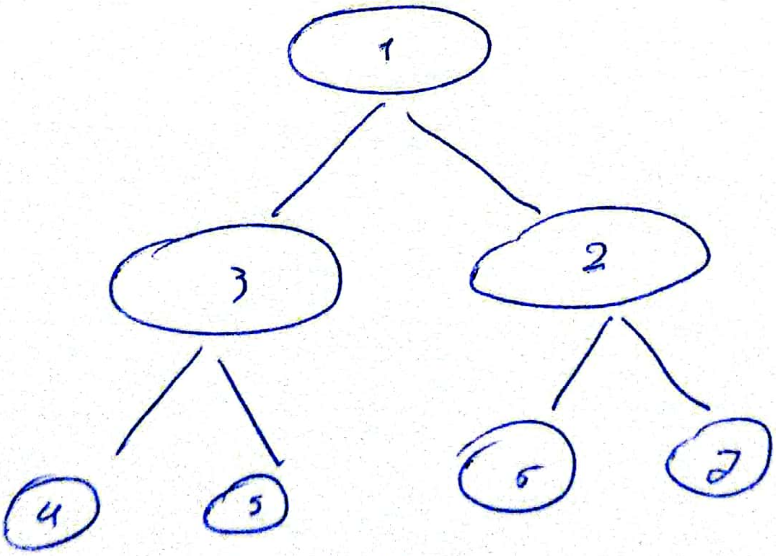
1. Pick a feature to use as "root node" and split data based on that feature.



what would be the
2. Second feature to split subdatas based on it.

...
:

3. what would be the nth. feature to split based on it



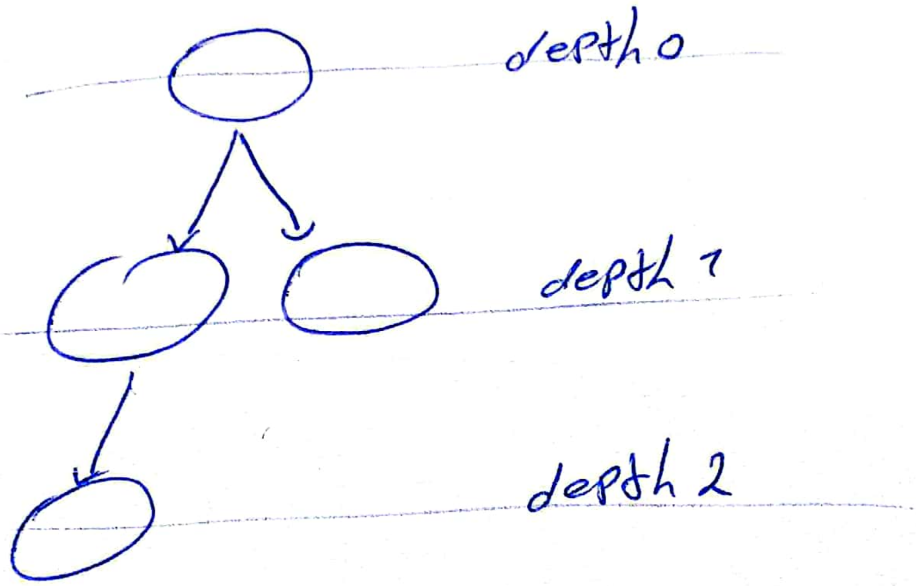
Key decision:

1. How to choose what feature to split on at each node?

use the feature which maximizes the purity or minimizes the impurity

2. ~~How~~ when do you stop splitting ?

- when a node is 100% one class.
- when splitting a node will result in the tree exceeding a maximum depth.



- when improvements in purity score are below a threshold.

- when number of examples in a node is below a threshold.

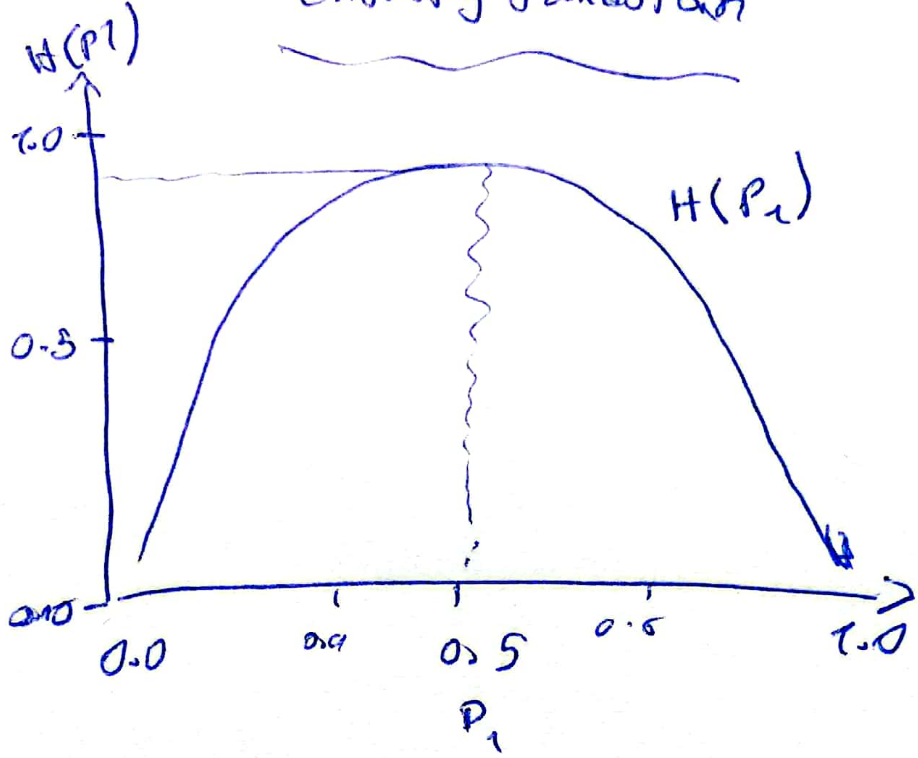
measuring purity

entropy as a measure of impurity.

P_1 = fraction of examples that have the same label.

exp:
$P_1 = \frac{3}{6}$

entropy function



when the data is 50/50 it reaches the highest (1) entropy, means it is in the highest level of entropy (impurity)

exps:

$P_1 = \frac{3}{6}$ $H(P_1) = 1$

$P_1 = \frac{5}{6}$ $H(P_1) = 0.65$ → quite impure

$P_1 = \frac{6}{6}$ $H(P_1) = 0$

$P_1 = \frac{2}{6}$ $H(P_1) = 0.92$

$P_1 = 0$ $H(P_1) = 0$

"formula of entropy function"

if = $P_0 = 1 - P_1$

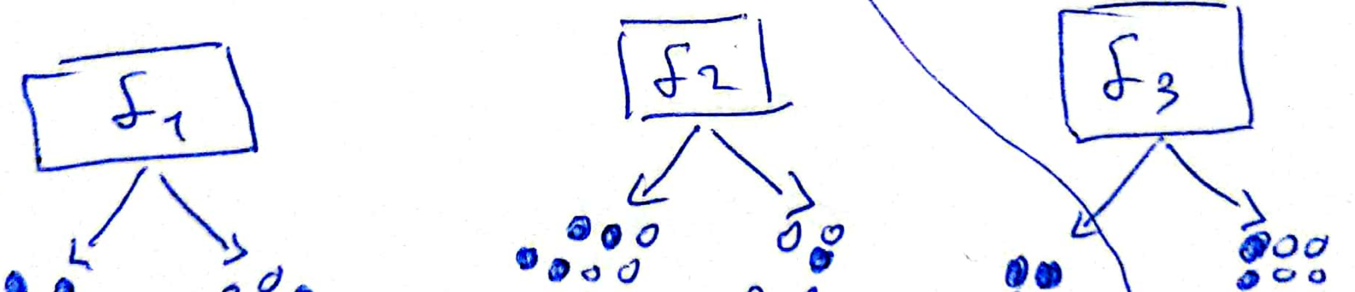
$$\boxed{H(P_1) = -P_1 \log_2(P_1) - P_0 \log_2(P_0)}$$

$$\boxed{= -P_1 \log_2(P_1) - (1-P_1)\log_2(1-P_1)}$$

choosing a split

Information gain:

$$H(0.5) - \left(\frac{4}{10} H\left(\frac{3}{4}\right) + \frac{6}{10} H\left(\frac{2}{6}\right)\right) = 0.12$$



$p_1 = \frac{4}{5}$ $\quad$ $p_1 = \frac{1}{5}$ $\quad$ $p_1 = \frac{4}{7}$ $\quad$ $p_1 = \frac{1}{3}$ $\quad$ $p_1 = \frac{3}{4}$ $\quad$ $p_1 = \frac{2}{6}$

$H(\frac{4}{5}) = 0.72$ $\quad$ $H(\frac{1}{5}) = 0.72$ $\quad$ $H(\frac{4}{7}) = 0.99$ $\quad$ $H(\frac{1}{3}) = 0.92$ $\quad$ $H(\frac{3}{4}) = 0.81$ $\quad$ $H(\frac{2}{6}) = 0.92$

~~weighted~~

compute weighted averages

$$H(0.5) - \left(\frac{5}{10} H\left(\frac{4}{5}\right) + \frac{5}{10} H\left(\frac{1}{5}\right)\right) = 0.28$$

$$H(0.5) - \left(\frac{7}{10} H(0.57) + \frac{3}{10} H\left(\frac{3}{4}\right)\right) = 0.03$$

after computing ~~average~~ weighted average for all features we choose the feature which has the "lowest" weighted average to split on.

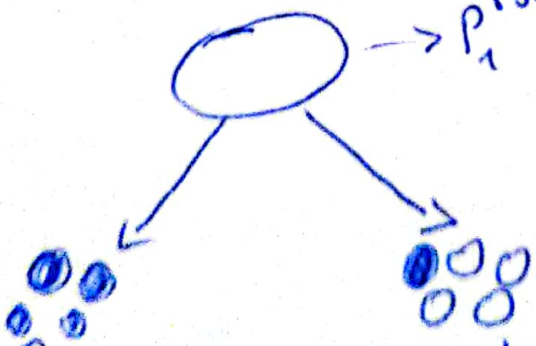
the result of the ~~ave~~ weighted average called information gain.

~~Information gain~~

we split the information on a feature which has the "Bigest" information gain or "lowest" weighted average.

"General formula of computing Information gain"

if,



$p_1^{root} = \frac{5}{10} = 0.5$

$p_1^{left} = \frac{4}{5}$ $\qquad$ $p_1^{right} = \frac{1}{5}$

$w^{left} = \frac{5}{10}$ $\qquad$ $w^{right} = \frac{5}{10}$

$$\text{Information gain} = H(p_1^{root}) - \left(w^{left} \cdot H(p_1^{left}) + w^{right} \cdot H(p_1^{right})\right)$$

Decision tree learning (recursive splitting)

1 start with all examples at the root node

2 calculate information gain for all possible features, and pick the one with the highest information gain.

3 split dataset according to selected feature, and create left and right branches of the tree.

4 keep repeating splitting process until stopping criteria is met:

stopping criteria

- when a node is 100% one class.
- when splitting a node will result in the tree exceeding a maximum depth.
- Information gain from additional splits is less than threshold.
- when the number of examples in a node is below a threshold.

using one-hot encoding of categorical features

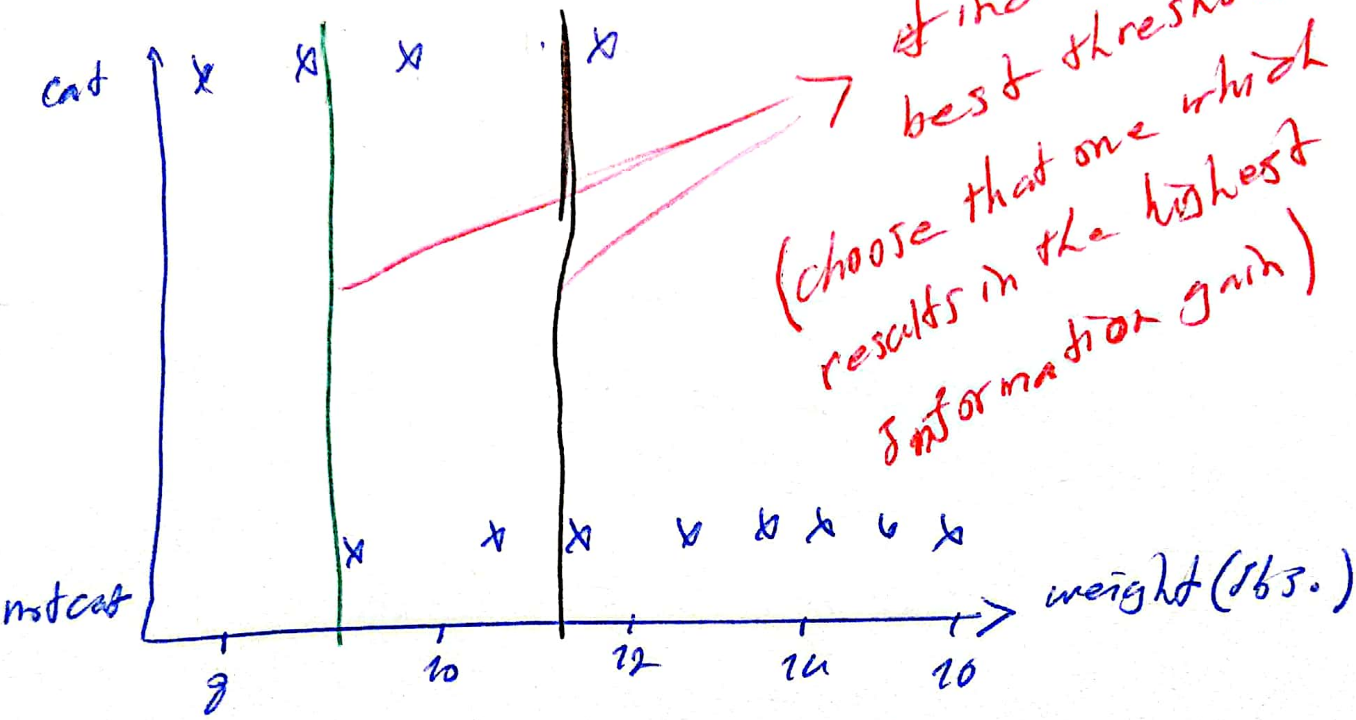
exps of one-Hot encoding

Ear shape	pointy ear	floppy ear	oval ear
pointy	1	0	0
oval	0	0	1
oval	0	0	1
floppy	0	1	0
pointy	1	0	0
oval	0	0	1
floppy	0	1	0
floppy	0	1	0

one-Hot encoding : If a categorical feature can take on k values, create k binary features (0 or 1 variables).

using one-hot encoding, categorical features can be used to train other models like neural networks or logistic regression.

"Decision tree for continuous valued features"



find the best threshold (choose that one which results in the highest information gain)

$$H(0.5) - \left(\frac{2}{12} H\left(\frac{2}{2}\right) + \frac{10}{12} H\left(\frac{2}{10}\right)\right) \approx 0.3$$

$$H(0.5) - \left(\frac{3}{12} H\left(\frac{3}{5}\right) + \frac{7}{12} H\left(\frac{3}{7}\right)\right) \approx 0.4$$

---

regression tree

choosing a split

based on variance of targets in each node.

exp: Lvariance = 1.47 Rvariance = 27.87

w left $= \frac{5}{10}$ w right $= \frac{5}{10}$

$$\boxed{(\text{overall var}) - \left(\frac{5}{10} \times 1.47 + \frac{5}{10} \times 27.87\right)}$$ → information gain

using multiple decision Trees

(Tree ensemble)

"Sampling with replacement"

choose a subset of original dataset with replacement (you may choose ~~and~~ examples one by one ~~and~~ with replacement) and train a tree with that subset.

"random forest algorithm"

(bagged decision Tree)

Given training set of size m:

For $b = 1$ to B:

T = use sampling with replacement to create a new training set of size m.

(T.size = m)

Train a tree with T ~~size (m)~~ -

at the end there will be b different trained tree.

~~bagged~~ bagged decision tree

random forest

At each node, when choosing a feature to use to split, if n features are available, Pick a random subset of $k < n$ features and allow the algorithm to only choose from that subset of features.

$\boxed{k = \sqrt{n}}$ a typical choice for k when n is large.

"XG Boost"

(extreme Gradient Boosting)

- The most common way to use (implement) decision tree algorithm.

Given training set of size m:

-for b = 1 to B:

- use sampling withe replacement to create a new training set of size m (But instead of picking from all examples with equal (1/m) probability, make it more likely to pick misclassified examples from previously trained trees.

Train a tree on the new data set.

Classification

```
from xgboost import XGBClassifier, XGBRegressor

model = XGBClassifier()
model.fit(X_train, Y_train)
y_pred = model.predict(X_test)
```

regression

```
model = XGBRegressor()
model.fit(X_train, Y_train)
y_pred = model.predict(X_test)
```

"when to use decision Trees"

"Decision Trees and tree ensembles"

- works well on tabular (structured) data.
- not reconended for unstructured data (images, audio, text).
- decision trees are fast.
- small decision trees may be human interpretable.

"neural networks"

- works well on all types of data, including tabular (structured) and unstructured data.

NN

- maybe slower than a decision tree.
- NN works with transfer learning.
- when building a system of multiple models working together, it might be easier to string together multiple neural networks.

2024.02.15

L. Samani