Entropy of n-possibilities :

$$S = \sum_i -p_i \log_{(2)} p_i$$

p_i = prob. of i-th ev.

Information $\propto 1/S$ ie ~~Info~~ S↑ info↓

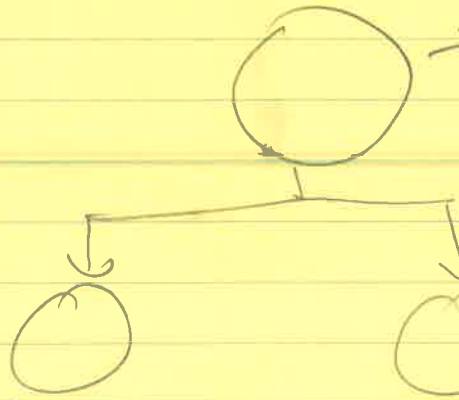
if $p_i = 0$ for some $i \Rightarrow p_i \log p_i = 0$

Information gain = Change in entropy

$$= -(S_f - S_i)$$

basically : upon classifying objects based on the answer of a particular quest. (in dec. tree), if the entropy ~~is S_f of the new~~ of the fin. state is S_f

then Info is gained if $S_f < S_i$



Info gain

$$= S_{parent} - \frac{1}{2}(S_{C_1} + S_{C_2})$$

↑ avg. of children nodes.

More general:

Info gain = Entropy (parent)

$$-\left[\left(\frac{m}{m+n}\right) S_{C_1} + \left(\frac{n}{m+n}\right) S_{C_2}\right]$$

if S_{C_1} contains only m of the total $(m+n)$ data (training) instances

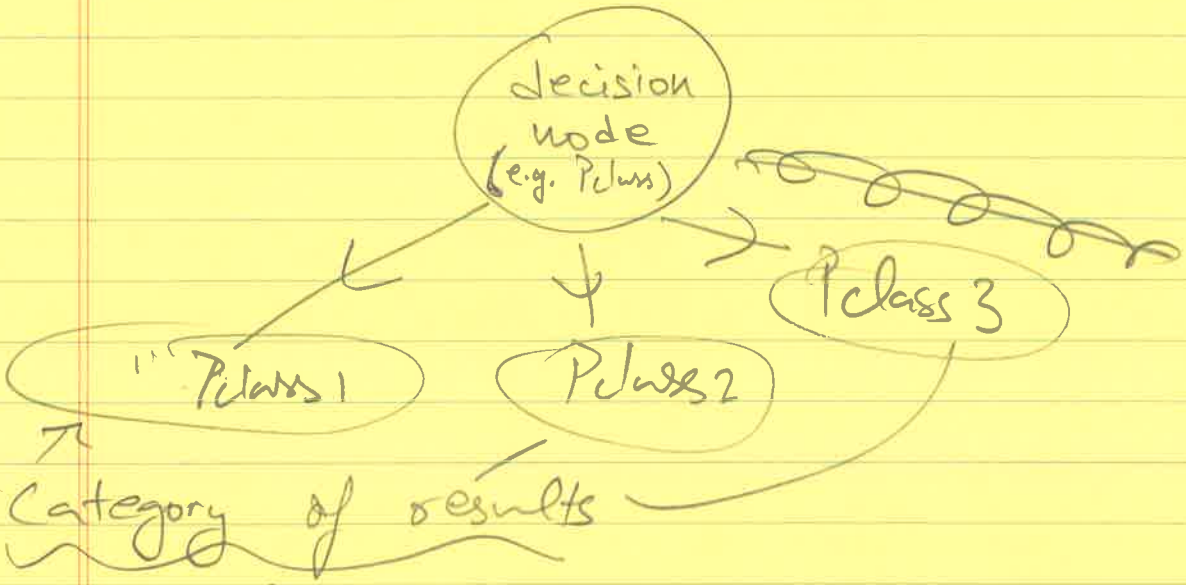
& S_{C_2} contains n of the total $(m+n)$ instances.

→ Making decision trees: Pick the split that gives the largest information gain.

(Basically try splitting by each feature & choose which one gives maximum entropy gain)

Decision trees tend to overfit a lot

In a decision tree, at each node we just split the data according to their result for the query in that node, & then compute the entropy for each category of the results



↳ these are called "leaves" of the nodes. → they become the children of the node we stopped split.

Hyper par.

max-depth → underfits in small.

min_samples_split → overfits if min_samp is small

min_samples_leaf → overfits if min_samples_leaf is small (ie becomes more complex)

df.iterrows

IPython.display.display()

8, 1, 7 %matplotlib inline

One-hot encoding
pd.get_dummies

DT
0.820
0.81
0.798
0.793

min_sample_leaf = 2 → 10 Complexity ↓
min_sample_split = 4 → 3 Complexity ↑

to list

Naive Bayes

Bayes thm.

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

$P(A \cap B)$: no. of ways A can happen given B has happened.

$P(B)$: total ways of B happening = unconditional prob. of B

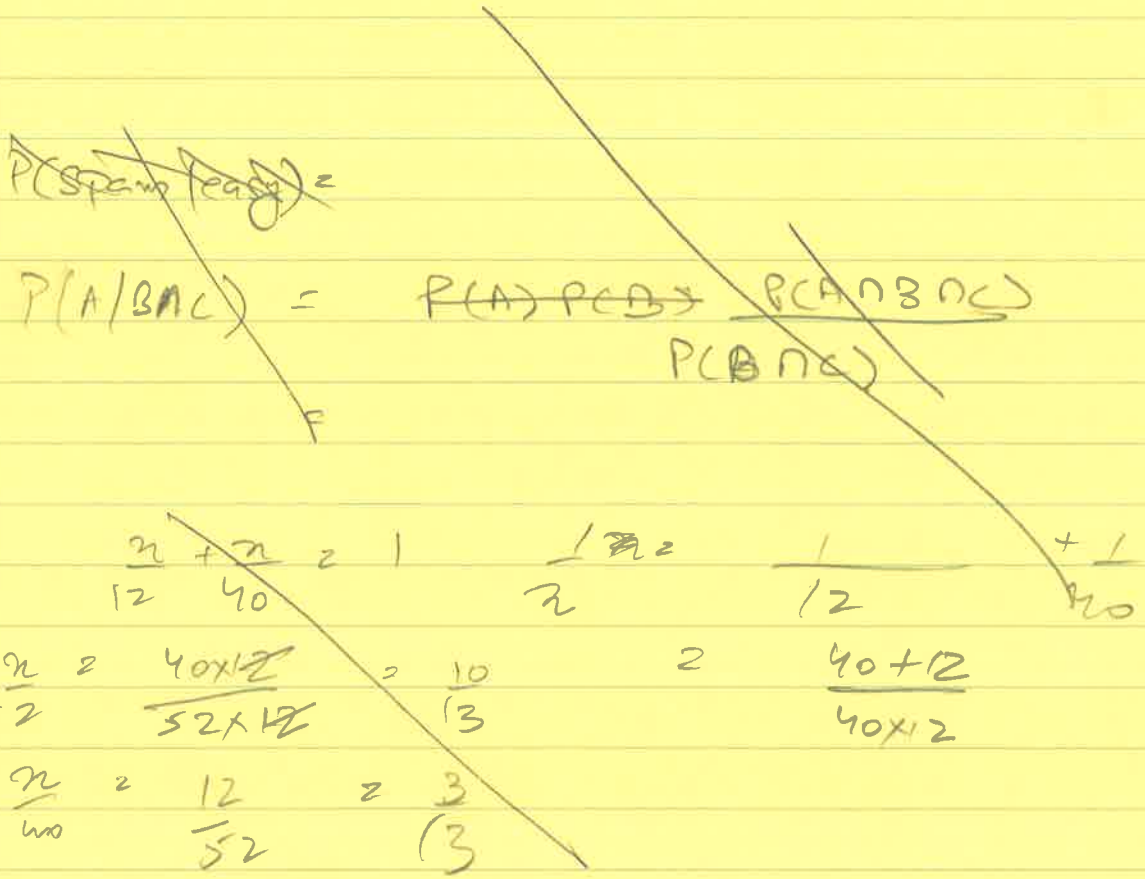
Recall $P(A|B) \neq P(B|A)$

$$P(A|B)\,P(B) = P(B|A)\,P(A) = P(A \cap B)$$

Phenomenenons of false positives: misdiag.

↳ if prob. of ~~error in diag.~~ is much larger than probability of occurrance of the sickness, then prob. of being actually sick upon testing positive is still low.

↳ e.g. lets make a fictitious disease that does not exist ⇒ prob. of some being afflicted with it is 0. Now devise a test for such a disease and it is 99% accurate but even if ~~someone~~ someone test pos., their prob. of being sick is yet



Naive Bayes => Assume that

$P(A \cap B \cap C \ldots) = P(A)\,P(B) \ldots P(C)$

This is only true if the events A, B & C are independent and not correlated

for e.g. in bag of words technique we assume the probability of each word is independent of other words, when in real world this is not entirely true.

Naive Bayes estimator

$$P(A \mid e_1 \cap e_2 \ldots e_n) = \frac{P(e_1 \cap \ldots \cap e_n \mid A)\, P(A)}{P(e_1 \cap \ldots \cap e_n)}$$

$$P(\bar{A} \mid e_1 \cap e_2 \ldots \cap e_n) = \frac{P(e_1 \cap \ldots \cap e_n \mid A)\, P(\bar{A})}{P(e_1 \cap \ldots \cap e_n)}$$

usually one can not evaluate the unconditional prob. $P(e_1 \cap \ldots \cap e_n)$

But $P(e_1 \cap \ldots \cap e_n \mid A)$, $P(e_1 \cap \ldots \cap e_n \mid \bar{A})$

$P(A)$ & $P(\bar{A})$ can be obtained from labelled data

use Naive Bayes

$\Rightarrow P(e_1 \ldots e_n \mid A) = P(e_1 \mid A) \cdot P(e_2 \mid A) \cdot \ldots \cdot P(e_n \mid A)$

$\Rightarrow$

$$P(A \mid e_1 \cap \ldots \cap e_n) \propto P(e_1 \cap \ldots e_n \mid A)\, P(A)$$

$$P(\bar{A} \mid e_1 \cap \ldots \cap e_n) \propto P(e_1 \cap \ldots e_n \mid A)\, P(\bar{A})$$

$\rightarrow$ evaluate proportionality constant by using

$$P(A \mid e_1 \ldots e_n) + P(\bar{A} \mid e_1 \ldots e_n) = 1$$

Pandas. read_table

↳ depricated
use read_csv

Bag of words: Only the frequency of each word appearing in a message is counted, the ordering of words is ignored

↓

Sklearn Count_vectorizer

Str.lower() → in Python string methods library

x.lower() for string in x

** kwargs() ?

string.punctuation → have to import string.

str.maketrans(x,y,z)

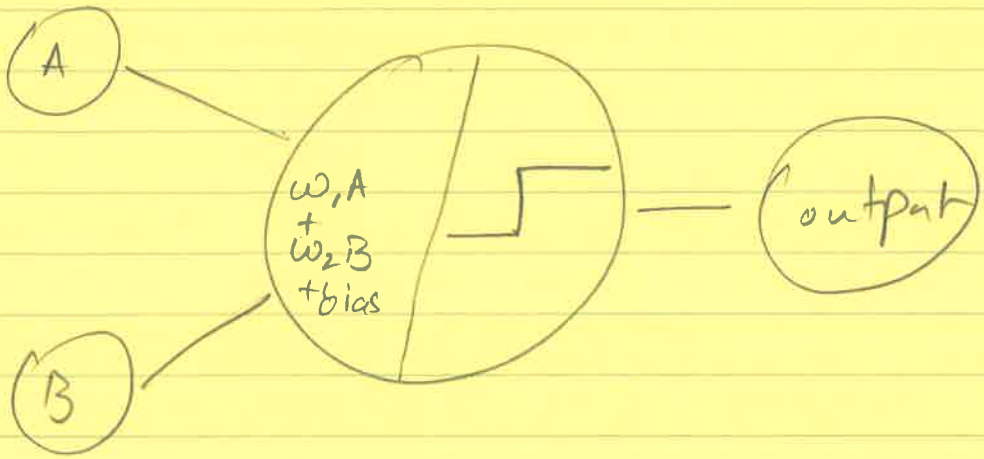
Counter() from collections in Python

Multinomial Naive Bayes: discrete valued features

Gaussian Naive Bayes: continuous values

→ perceptron

~~map~~ → ala udacity



Python : zip( )

1
8
16
7

~~map~~ ~~list~~
How to properly use ~~map~~ map & lamda

np. matmul

Pandas. map (to ~~convert~~ map named classes / label into numerals)

Pandas. apply

3401/4
41208
45222

→ A a good bench mark is the naive predictor, that naively predicts the answer for any input data to be the most frequently occurring class in the training dataset.

↳ shows what a model without any ~~dearon~~ intelligence will look like

→ problem of continuous data not being gaussian ↔ log transform often helps.

Usefullness of SVM: Ability to find optimal & stable for current problem. seperating hyper planes, and lower over-fitting tendency in a large feature space

~~(11,4)~~
~~(11,15,4)~~
~~(9,15,16)~~
~~(16,14,6)~~

(18, 6, 18)

(

Decision tree has too many hyperparameters This can make it difficult to compare complexity of two models if two or more of their hyperparameters are diff. e.g one has larger max_depth but larger "leaf size" other has smaller max_depth but also smaller leaf size

sklearn.preprocessing. polynomial features.

↓

sklear_ linear Regressor

→ Require almost no data preprocessing

Decision trees are ~~useful~~ less prone to skewed data & hence useful in ~~cases~~ ~~where~~ unbalanced class distribution → eg Cancer vs not cancer.

→ limitation : tend to overfit

↳ scale logarithmically with size of data
⇒ can handle large data sets.

Random Forrest : Bootstrapping a sample

Panda. values ?

Python : clone( ) ?

~~X_train.columns.values~~ → pandas?

Passing f_β with a particular value for β

```
scorer = make_scorer(fbeta_score, beta=0.5)
```

Clustering

1) K-means clustering :

How to choose the correct no. of clusters ?

Answer: Elbow method ↓

Compute mean squared distance of ~~each point from cluster~~ and plot it against the number of clusters

Alternately look at Silhouette score

~~education, education-num, occupation, work-class~~

age, sex, capital-gain

~~age, education-num, capital-gain,~~

mar_stat, education-num, age, capital-gain, hours-per-week.

sex

0 1

39.7 & 45.7

45 % 43 %

civ civAf

K Means :

Silhouette ~~sor~~ score : tells us how similar a point is, to its own cluster as opposed to ~~a diff~~ some other cluster

How is this diff. from pandas.concat?

pandas.merge

pandas.pivot_table

pandas.to_coo
↳ to convert
~~Pa~~ to sparse
Dataframe

Other clustering algorithms

→ Hierarchical Clustering

→ DBSCAN : → robust against noise

→ does not require no. of clusters

→ clusters densely packed data points & labels rest as noise

↳ can be used for anomaly/fraud detection

Also HDBSCAN

not in sklearn

Yes choose ε & min-samples s.t. most of the points of a cluster are in 1 cluster & rest are noise

Single link Hierarchical clustering

↳ Dendograms.

↳ Very powerful to get insights about clustering

have to import scipy.cluster.hierarchy

↳ also contains lib. for Hierarchical clustering methods

Agglomeritive clustering algos.

↳ included in sklearn

→ 1) Complet link clustering

↳ distance between the clusters is ~~measured~~ give by the distance between two maximaly seperated points in the clusters

↳ then the two "nearest" clusters are merged.

→ Average link clustering:

↳ Distance between clusters is the averge distance between two points from the clusters.

→ Ward's method

↳ Tries to ensure that merging clusters leads to least increase in variance in the cluster after merging

Cluster validation index

→ Since ~~normalization~~ clustering is based on distance, if different features have different scales, then their contribution to the distance will get weighted accordingly. So it best to normalize all of them to same scale

note that the objective is to make sure the variance in each feature is ~~to~~ given equal weight.

~~→ Reason~~

Seaborn clustermap.

→ DBSCAN

→ Gaussian Mixture Models (GMM)

↳ ~~Did~~ used in speaker verification (see paper by Douglas A Reynolds, Thomas F Quatieri, Robert B Dunn)

↳ Also in Biometric verification.

~~Cluster indices~~

→ Cluster Validation index

1) External indices → (adjusted Rand score)

2) Internal indices (e.g. Silhouette co-efficient)

Should not be used for DBSCAN (since does not understand the concept of noise)

Also not
- the most optimal score if the data is not the form of dense circular cluster
e.g it will not give good score or be good for scoring

ie data is in the form of two co-centric circular clusters

↳ use DBCV

→ Feature Scaling :

→ PCA : Projecting data on the Direction of maximum variance will have minimum loss of information

↓

The principal components of a data are always chosen to be perpendicular to each other.

Rule of thumb: Try to use as many components such that 80% of variance gets explained by them.

PCA can be used for noise reduction.

eigenface : PCA for face recog.

(sk-learn choose PCA with svd_solver = "randomized"

When feature is too large for PCA

→ Random Projection: (based on Johnson-Lindenstrauss lemma)

Basically projection matrix is a matrix of random no.

→ find component along randomly chosen directions

→ might not work for low dimensional feature spaces

→ works quite with when feature space is large.

→ Gaussian Random Projection
→ Sparse Random projection.

Imp → no. of samples must be greater than the no. of components we seek.

ICA: Independent Component Analysis

↳ hopes/assumes that features are admixtures of independent sources & tries to extract these independent sources.

↳ for e.g. 3 or more independent sources of sound in a recording.

↳ This problem is called "blind source seperation"

The independent component must have Non-gaussian distribution (V. imp)

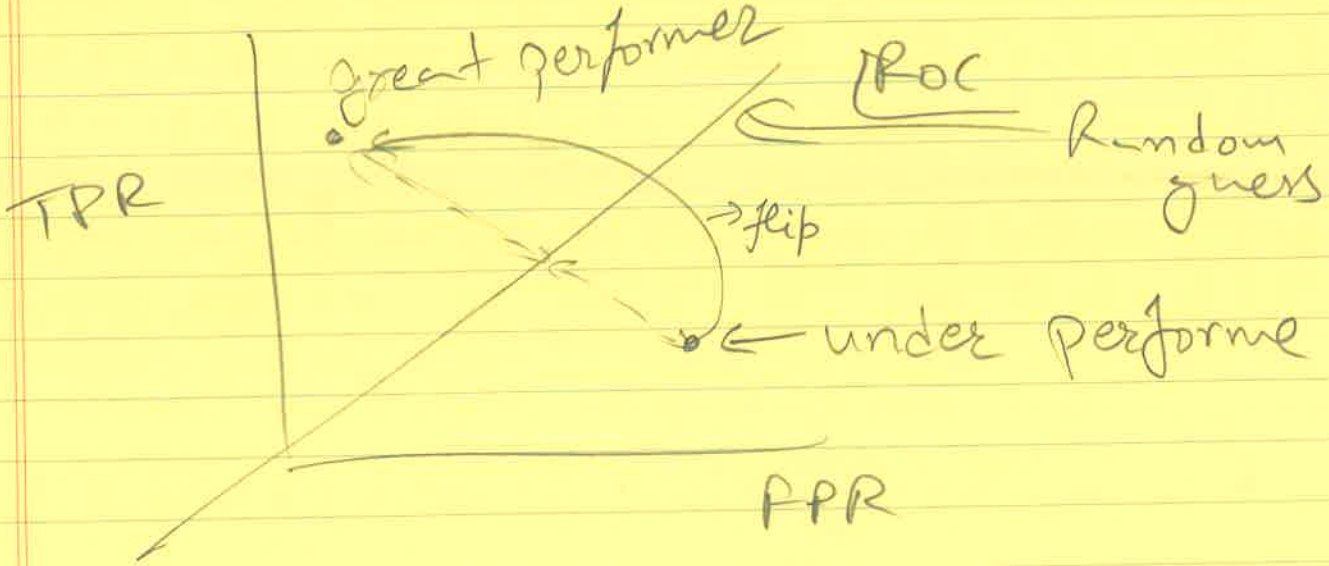
(Fast ICA)

Whitening a dataset?

→ read & understand central limit thm.!

Note that in classification (for 2-class classification) random guesses will give 50% accuracy.

A classifier that underperforms a random guess can be easily mirrored into a good classifier by merely taking the opposite of its ~~predic~~ predictions.



Python zip

np. ravel , np. flatten