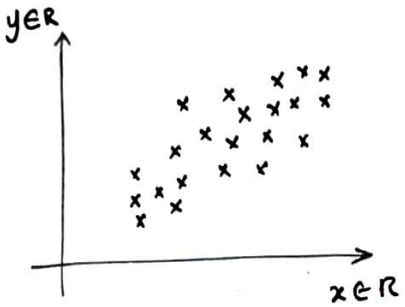
INTRODUCTION TO MACHINE LEARNING:

Consider a data, $D = \{(x_i, y_i) : x_i, y_i \in R^2, i = 1(1)n\}$

	X	Y
1	x_1	y_1
2	x_2	y_2
:	x_3	y_3
:	:	:
n	x_n	y_n

The datapoints can be plotted in a real plane.



$x_i, y_i \in R^2$

GOAL: To build a predictive model to predict y based on x

(ii) to create a function $f(x)$ that accurately predicts y.

(ii) $\boxed{y_i = f(x_i)}$, $\forall\, i = 1(1)n$ ($\forall$ → for all)

↳ Finding a y_i for every value of x_i

However, the equality in the above equation never holds true because in real life it's extremely difficult to capture the exact effect.

x = Area (sqft living)

y = Price

$y = f(x)$

Whatever equation or function you use to predict the price, you won't be able to predict with only the area.

$y = a + bx$

$y = a + bx + cx^2$

$y = a + bx + cx^2 + dx^3$

For a given area, the house can be underconstruction, ready to move in, semifurnished, furnished, 1st floor, 10th floor such that these affect the houseprice.

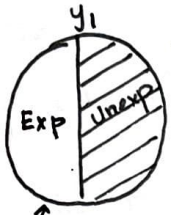
$y = f(x_1, x_2, \ldots, x_6)$, if we do so we may be able to reach closer to y but the equality will still not hold.

x_1 = Area
x_2 = Floor
x_3 = # bedrooms
x_4 = # bathrooms
x_5 = waterfront (Y/N)
x_6 = underconstruction (Y/N)

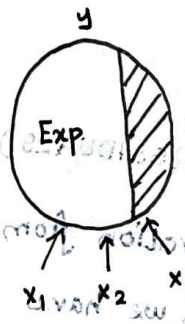
→ Whatever may be the case, exact prediction is not possible; given the inputs and will end up making some error every time we make a prediction.

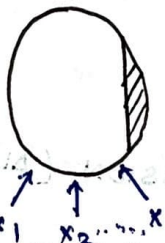
MODIFIED EQUATION:

$y_i = f(x_i) + \varepsilon_i$, $i = 1(1)n$ and ε_i are random noise.



→ There may be no. of reasons that we couldn't account for (quick sale, waiting for price shoot up, suicide) ← random noise or error





This equation holds true

$y_i = f(x_i) + \varepsilon_i$

whereas

$y_i \neq f(x_i)$, $\forall\ i = 1(1)n$

$Y \neq f(x_1, x_2, \ldots, x_6)$

↳ because there's an error factor

ERROR:

→ Error or random error is the unexplained part of y.

→ If ε_i is +ve then $f(x_i)$ has underestimated y_i [$y_i - f(x_i) = +ve$]

→ If ε_i is -ve, then $f(x_i)$ has overestimated y_i. [$y_i - f(x_i) = -ve$]

→ Since errors are inevitable (or) unavoidable, we always focus on designing Statistical frameworks that solves the problem of prediction by minimizing the errors.

Design a model that's minimizing the errors.

LINEAR REGRESSION MODEL:

→ Considering the same data $D = \{(x_i, y_i) : i = 1(1)n, x_i, y_i \in \mathbb{R}^2\}$

Consider a linear model:

$y_i = b_0 + b_1 x_i + \varepsilon_i$ ← consider called a linear model because the dependent variable y_i is linear in the parameters

parameters

Is the model: $y_i = b_0 + b_1 x_i^2 + \varepsilon_i$ a linear model?

It's a linear model because it is linear in parameters.

Linear → Flat surface in p dimensional space.
→ 1 and 2.

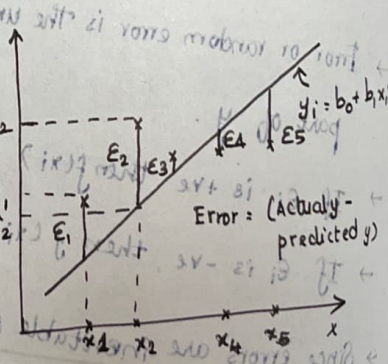
In higher dimension, linear regression is a plane.

Higher the dimensions, there linear regression will take different shapes.

ERRORS IN LINEAR REGRESSION (ALSO CALLED RESIDUALS)

→ Errors are the vertical distance of an observation from the linear regression line which means that if we have two variables x and y,

Error = (Actual y - predicted y)



PARAMETERS IN SIMPLE LINEAR REGRESSION MODEL:

↑ simple - one predictor.

$y_i = b_0 + b_1 x_i + \varepsilon_i, \quad i = 1(1)n$

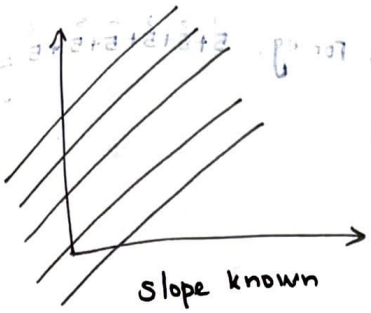
$y = b_0 + b_1 x$ ✗ Incorrect

$y = b_0 + b_1 X$ ✗

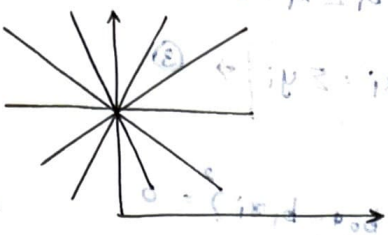
b_0 = y-intercept

b_1 = slope of the line.

A line depends on these two numbers.



Slope known



y-intercept known

Estimate the parameters.

Estimation of Parameters

(Optimization objective in linear regression)

Estimate the parameters such that the sum of squared errors is minimum. (ii) $\sum_{i=1}^{n} \varepsilon_i^2$ is minimum.

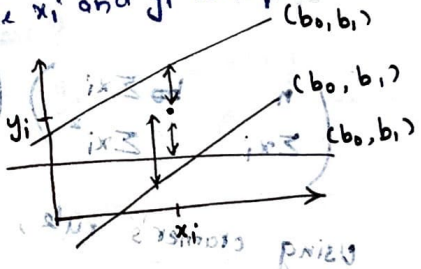
→ choose that particular line with min. SSE.

$\varepsilon_i = ?$, $\varepsilon_i = y_i - \hat{y_i}$

$\varepsilon_i = y_i - (b_0 + b_1 x_i) = y_i - b_0 - b_1 x_i$

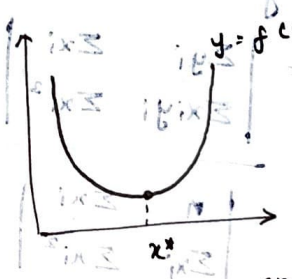
⇒ ε_i is a function that depends on b_0 and b_1.

$\varepsilon_i = f(b_0, b_1) = y_i - b_0 - b_1 x_i$ (where x_i and y_i are fixed)



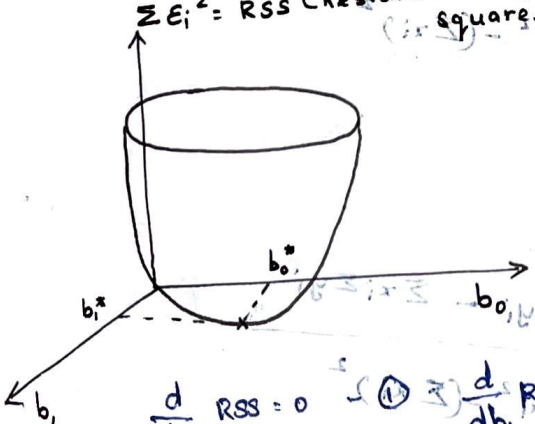
$\sum_{i=1}^{n} \varepsilon_i^2 = \sum (y_i - b_0 - b_1 x_i)^2$

We want to minimize $\sum_{i=1}^{n} \varepsilon_i^2$ by choosing b_0, b_1.



$\frac{d}{dx} f(x) = 0$ and solve for x.

$\sum \varepsilon_i^2$ = RSS (Residual sum of squares)



$\frac{d}{db_0} RSS = 0$ ① $\frac{d}{db_1} RSS = 0$ ② (*)

① ⇒ $\frac{d}{db_0} \sum (y_i - b_0 - b_1 x_i)^2 = 0 \Rightarrow \sum \frac{d}{db_0} (y_i - b_0 - b_1 x_i)^2 = 0$

$\Rightarrow \sum 2 (y_i - b_0 - b_1 x_i)(-1) = 0$

$\Rightarrow -2 \sum \varepsilon_i = 0$

÷ by -2 on both sides

$$\Sigma y_i - n b_0 - b_1 \Sigma x_i = 0$$

$$\Rightarrow \boxed{n b_0 + b_1 \Sigma x_i = \Sigma y_i} \rightarrow (3)$$

Adding constant n times is same as na.

For eg, $5+5+5+5+5+5 = 6 \times 5 = 30$

(2) $$\frac{d}{db_1} \Sigma (y_i - b_0 - b_1 x_i)^2 = 0$$

$$\Sigma \frac{d}{db_1} (y_i - b_0 - b_1 x_i)^2 = 0$$

$$\Sigma 2 (y_i - b_0 - b_1 x_i)(-x_i) = 0$$

$$\Rightarrow -2 \Sigma (x_i y_i - b_0 x_i - b_1 x_i^2) = 0$$

$$\Rightarrow \Sigma x_i y_i - b_0 \Sigma x_i - b_1 \Sigma x_i^2 = 0$$

$$\Rightarrow b_0 \Sigma x_i + b_1 \Sigma x_i^2 = \Sigma x_i y_i \rightarrow (4)$$

$$n b_0 + b_1 \Sigma x_i = \Sigma y_i \quad - (3)$$

$$b_0 \Sigma x_i + b_1 \Sigma x_i^2 = \Sigma x_i y_i \quad - (4)$$

$$\begin{pmatrix} n & \Sigma x_i \\ \Sigma x_i & \Sigma x_i^2 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} \Sigma y_i \\ \Sigma x_i y_i \end{pmatrix}$$

Using cramer's rule,

$$b_0 = \frac{\begin{vmatrix} \Sigma y_i & \Sigma x_i \\ \Sigma x_i y_i & \Sigma x_i^2 \end{vmatrix}}{\begin{vmatrix} n & \Sigma x_i \\ \Sigma x_i & \Sigma x_i^2 \end{vmatrix}} = \frac{\Sigma y_i \Sigma x_i^2 - \Sigma x_i \Sigma x_i y_i}{n \Sigma x_i^2 - (\Sigma x_i)^2}$$

$$b_1 = \frac{\begin{vmatrix} n & \Sigma y_i \\ \Sigma x_i & \Sigma x_i y_i \end{vmatrix}}{\begin{vmatrix} n & \Sigma x_i \\ \Sigma x_i & \Sigma x_i^2 \end{vmatrix}} = \frac{n \Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n \Sigma x_i^2 - (\Sigma x_i)^2}$$

PROBLEM: Predict MPG based on weight of car

y = MPG, x = Weight

Σx_i = np.sum(weight)

Σy_i = np.sum(mpg)

Σx_i^2 = np.sum(weight**2)

$\Sigma x_i y_i$ = np.sum(weight * mpg)

n = len(cars)

b_0 = ?

b_1 = ?

$\hat{MPG} = b_0 + b_1 * Weight$

$\hat{MPG} = 46.325 - (0.007 * Weight)$

Weight = 1000 lbs

$MPG = 46.325 - (0.007 * 1000)$

$MPG = 39.325$.

$\hat{MPG} = 39.329$

= predicted value of MPG for Weight = 1000 lbs

⇓

What exactly are we predicting here?

$\hat{MPG} = E(MPG | Weight)$

$\hat{MPG} = mean(MPG | Weight)$

(ii) If we have n no. of cars

In linear regression we always predict the average and in most cases of ML we predict the average.

Incase of a KNN regressor, we output conditional mean as well.

The predicted average value of MPG of car weighing 1000 lbs is 39.329.

HW for tomorrow:

$y_i = b_0 + b_1 x_i + \varepsilon_i$

$\hat{MPG} = 46.325 - (0.007 * 1000)$

↑ after minimizing error

(ii) ε_i becomes negligible

	Wt	MPG	$\hat{MPG}$	Error
C_1	3504	18	18.95	-0.95
C_2	3504	19	18.95	0.15
C_3	3504	20	18.95	1.05

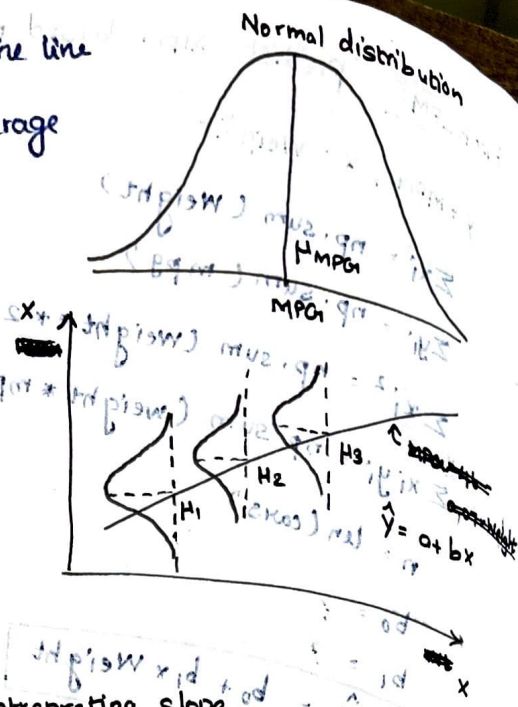
$y_i = b_0 + b_1 x_i + \varepsilon_i$

ε_i is a random variable.

∴ y_i depends on ε_i, y_i is a random variable as well.

The b_0 and b_1 values are such that they are minimized.

The linear regression, every point in the line is considered to be the mean (or) average such that MPGi follows a normal distribution.



$y = a + bx$

↑ slope

with one unit increase in x, y increases by b units.

$\hat{MPGi} = 46.325 - 0.007 \times Wt$

Intrepreting slope

⇒ With 1 unit increase in wt, the MPGi increases by −0.007 miles

→ " , the MPGi decreases by +0.007 miles

→ With 1 lb increase in wt, the MPGi decreases by 0.007 miles.

→ With 1000 lb increase in wt, the MPGi decreases by 7 miles.

↳ Most intrepretable.

$y = a + bx$ → intercept $y = a$ when $x = 0$.

Intrepreting y-intercept

For an equation, $y = a + bx$, a can be intrepreted as the value y takes when $x = 0$.

For eg consider the popular regression model in economics → Expenditure = a + b × Income.

Even though your income is zero, however you have experience or (i) expenditure. Not applicable for all models.

→ a is the expenditure incurred when income = 0. This shows that even there's no earning there may be a fixed expenditure.

→ However intercept in this way isn't possible for many examples. As in our case when we write $\hat{MPGi} = 46.325 - 0.007 \times$ Weight.

Since the weight of a car cannot be 0 pound, the intercept cannot be intrepreted as the same way as before.

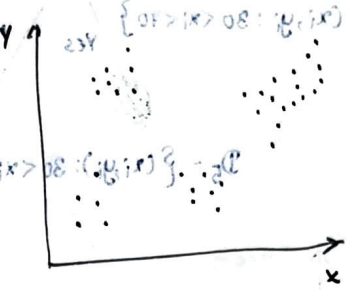
DECISION TREE:

→ KNN does not use any parameters.

→ Linear regression uses parameters → $y_i = b_0 + b_1 x_i$ (parameters)

CART → Classification and Regression

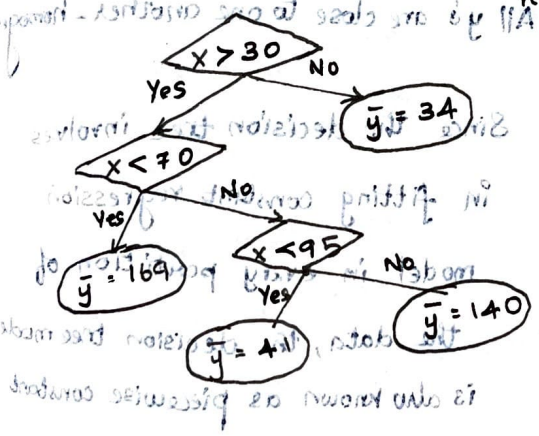
Seeing graph we can't say x and y are related. The issue is that x and y are not linearly related.



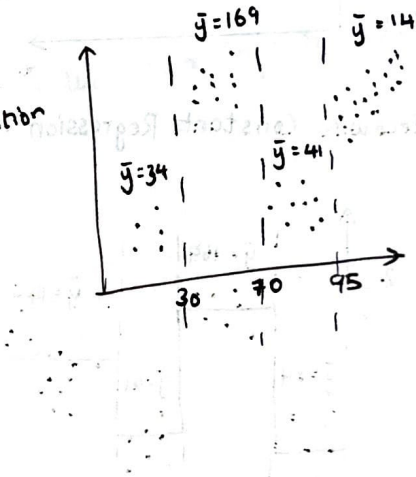
But the relation can be expressed as

$y_i = b_0 + b_1 x_i + b_2 x_i^2 + \ldots + b_p x_i^p$

For unit change in x_i^2, the value of y_i increases by b_2 units keeping all others constant → not intrepretable.



Yes → and condition

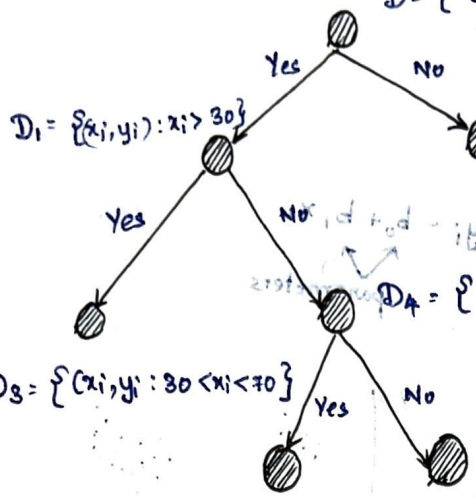


DT: Identifies how data can be identified as cluster.

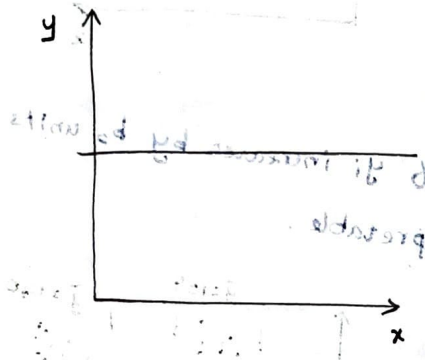
Assumes that value of y in a region, the predicted value of y can be approximated to the average values of y in that region.

: Each decision is represented as a node.

Every node in a decision tree is associated with a subset of data.



Partitioning done on features keeping in mind, values in y are as close as possible.



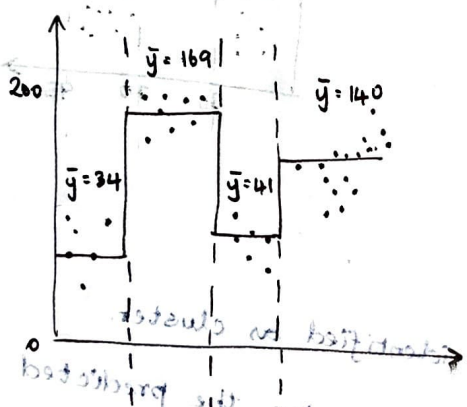
Constant Regression model

$\hat{y} = \bar{y}$

$\hat{y} = a + bx$

$\hat{y} = a = \bar{y}$

Piecewise Constant Regression Model (All y's are close to one another - homogenous)



Since the decision tree involves in fitting constant regression model in every partition of the data, the decision tree model is also known as piecewise constant regression model

→ We usually ignore = sign in continuous distributions since P(X = 30)

→ Splitting can be done only based on the features (x) and homogenity of target variable (Y).

NOTES:

(1) Every node of a decision tree contains a subset of data

(2) The root node of a decision tree contains the entire data.

(3) A CART model can only have two arms, for every decision nodes.

It's right arm always corresponds to value of True and the left arm corresponds to a value of False.

(4) A de regression tree partitions the data into a number of homogenous partitions based on the values of Y (ie the homogenity is observed according to the values of Y).

(5) The partitioning of the data happens in the feature space only.

(6) A decision tree begins with one root node and may have several leaf nodes.

(7) A decision tree with one root node and 2 leaf nodes is called a "decision stump".

Measure of node Impurity for Regression Tree: Decision Tree predicts mean

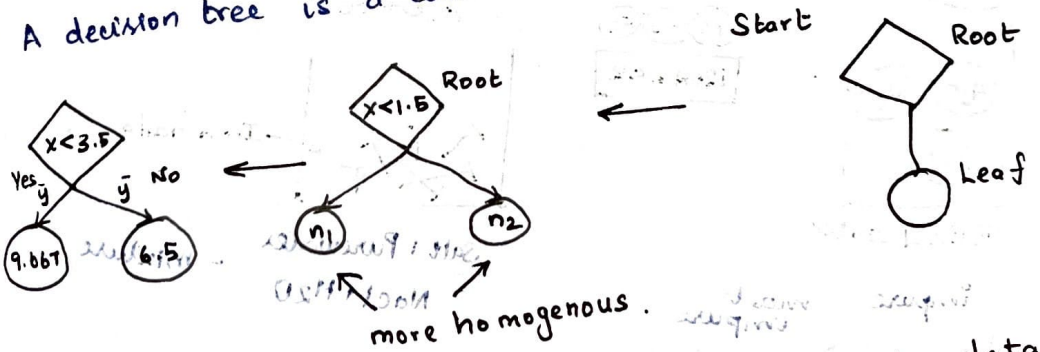
$$D_{node} = \sum_{y \in node} (y_i - \bar{y})^2$$ ← Looks like variance.

Taking the SSE of each interval.

→ If the value of y is same for different values of x in interval, then $y_i = \bar{y}$, then $D_{node} = 0$.

→ The deriance of each node must be minimum and overall deviance must also be minimum.

→ A decision tree is a combination of decision stumps.



Lesser the value of D_T, more the homogenous is the data split.

→ Decision Tree algorithm needs to bestopped (or) otherway is pruning ↓ letting decision tree grow and cutting unwanted leares.

→ All conditions are bounded (ANDED).

Most intrepretable: 1st Decision Tree, 2nd Linear Regression, 3rd KNN

In KNN we are cluster data based on value of x and in decision tree we cluster based on value of y (homegeneity)

→ Different models use different approaches.

Feature selection can't be usually interepretable using decision trees.

→ Not OK → using decision tree or random forests for feauture selection and using those features in linear regression. (NEVER USE)

→ Never use label encoded with numbers (US-1, Japan-2, Europe-3) in linear regression.

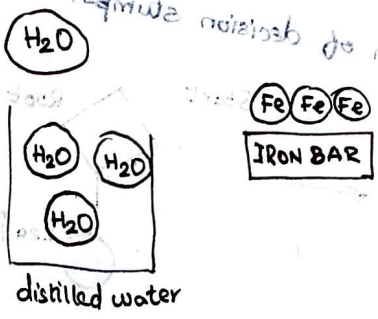
ISLR Book
↳ 20-30 days

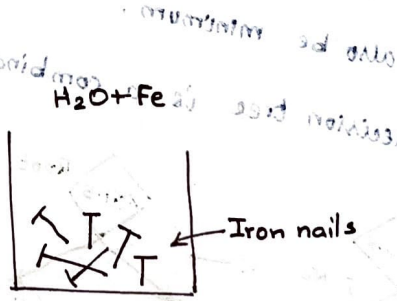
Course era → University of Washington Machine Learning
↳ Assignments in Github or Apply for Compensation.

CLASSIFICATION TREE

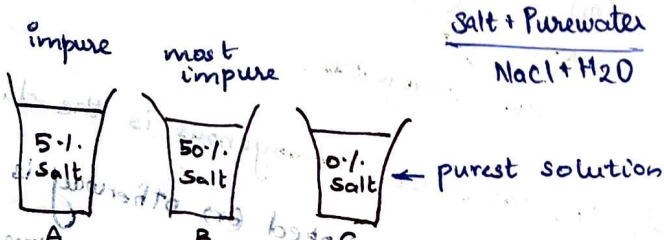
Recursive partioning → A decision (ii) CART algorithm partitions the data in a recursive fashion and therefore it's also known as a recursive partitioning algorithm.

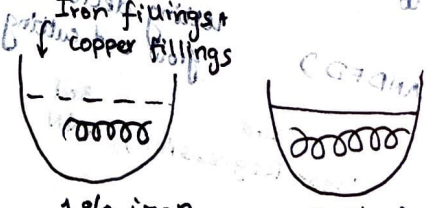
1. Mixture → impure substance
2. Compounds / Elements → pure substance.





Salt + Purewater / NaCl + H_2O = mixture





→ When y is categorical in nature, homogeneity is calculated based on which feature the split
↳ out of 2 features

→ The probability of tumour being malignant given the age of person between 28 and 50 and smokes cigs 8 is 4/5.

MEASURES OF SELECTING THE BESTSPLIT (FOR CLASSIFICATION TREE).

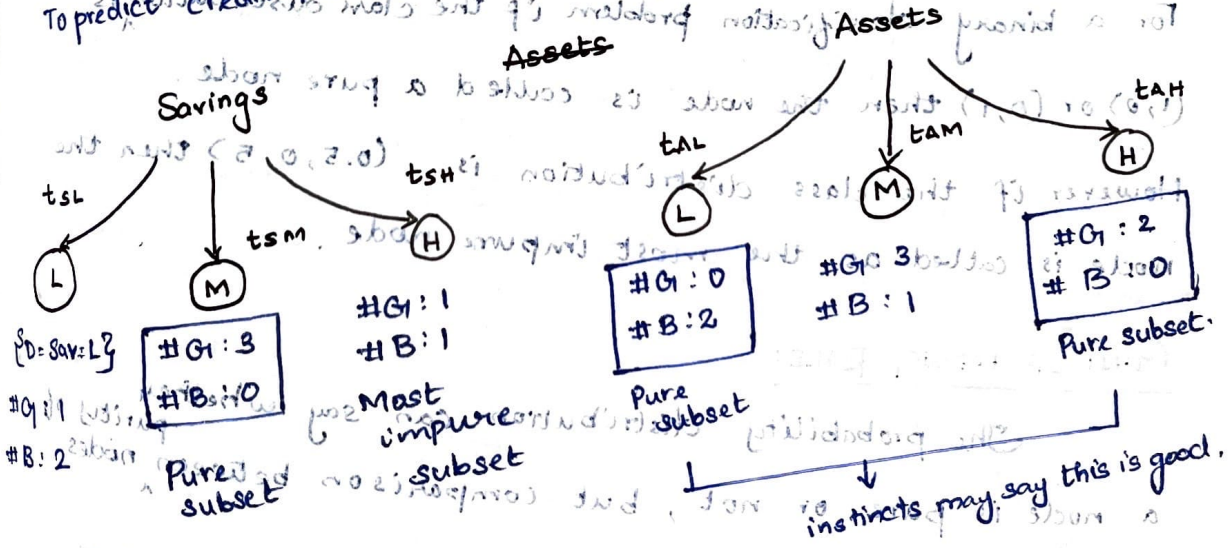
Regression	Classification
Homogenity measured using Total deviance D_T	Purity of node measured

Example data:

CID	Savings	Assets	Credit Risk
1.	M	H	G
2.	L	L	B
3.	M	M	B
4.	L	M	G
5.	H	M	G
6.	L	H	B
7.	M	L	G
8.		M	

Individual labels can be used to partition the data

To predict Credit Risk based on Savings and Assets.



NOTATION:

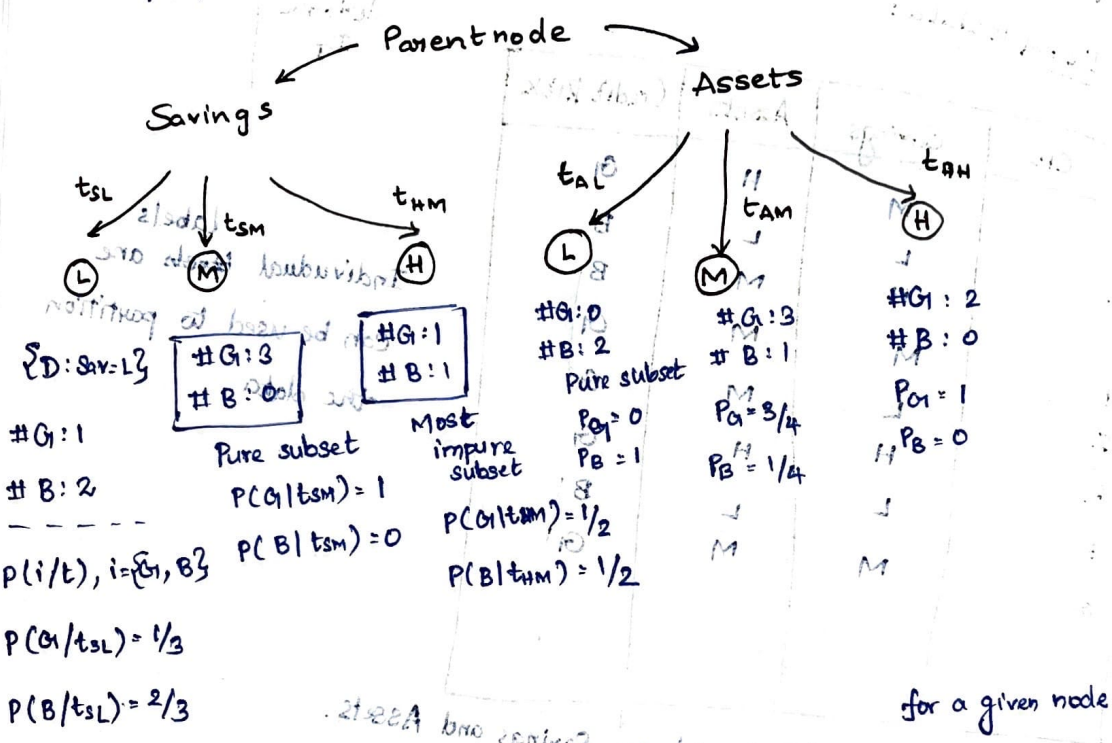
$p(i/t)$ = proportion of observations belonging to the i^{th} class at node t

(e.g) $p(0/t)$ = proportion of observations belonging to class 0 at node t

∴ For a binary classification problem, where $y_i \in \{0,1\}, \forall i$

$p(0/t) + p(1/t) = 1$.

two classes → G and B



$p(i/t), i = \{G, B\}$

$p(G/t_{SL}) = 1/3$

$p(B/t_{SL}) = 2/3$

for a given node

For a binary classification problem if the class distribution is $(1,0)$ or $(0,1)$ then the node is called a pure node.

However if the class distribution is $(0.5, 0.5)$ then the node is called as the most impure node.

GINI SPLITTING RULE:

The probability distribution can say whether purity of a node is pure or not, but comparison between nodes

→ Although the conditional probability distribution of the classes help us to understand the purity of a node, but it's difficult to use such distributions to compare

the purity of multiple nodes.

→ For the purpose of comparison, we need one single number that can help us to understand if one node is purer compared to another.

Genie Splitting Rule helps us to come up with such numbers that are useful for comparing the purity of multiple nodes.

$$Gini(t) = 1 - \sum_i p^2(i/t)$$

For a binary classification problem, $i \in \{0, 1\}$ then

$$Gini(t) = 1 - p^2(0/t) - p^2(1/t)$$

Note:

1. For binary classification, a pure node has a genie = 0 and 1

eg: if node t has a class distribution of (1,0) then

$$Gini(t) = 1 - 1^2 - 0^2 = 0$$

The same holds to if class distribution of t is (0,1)

$$Gini(t) = 1 - 0^2 - 1^2 = 0$$

2. For a binary classification problem an impure node has Gini = 0.5

If a node t has a class distribution of (1/2, 1/2) then

$$Gini(t) = 1 - (1/2)^2 - (1/2)^2 = 1/2 = 0.5$$

3. For a multiclass classification problem, a pure node has Gini equal to 0. (ii) Gini = 0.

Eg: If a node t has 4 classes with a class distribution of (0, 1, 0, 0) then gini(t) = $1 - 0^2 - 1^2 - 0^2 - 0^2 = 0$.

We see that the value of gini decreases with increase in purity of a node and increases with an increase in impurity of a node. Therefore gini can be thought as the measure of badness of a split. (ii) higher the value of gini worse is the split.

Homework: Compute the genie of each node arising due to splits by savings and assets. Express the numbers in decimals.

Comparing purity of 2 parent nodes

$$\text{Gini}(t_{SL}) = 1 - \sum p^2(i)$$

$$= 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 \quad 1 - \frac{1}{9} - \frac{4}{9} = \frac{9-5}{9} = \frac{4}{9} = 0.44$$

$$\text{Gini}(t_{SM}) = 1 - (1)^2 - (0)^2 = 0$$

$$\text{Gini}(t_{SH}) = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = \frac{1}{2} = 0.5$$

$$\text{Gini}(t_{AL}) = 1 - (0)^2 - (1)^2 = 0$$

$$\text{Gini}(t_{AM}) = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 1 - \frac{9}{16} - \frac{1}{16} = \frac{16-10}{16} = \frac{6}{16} = \frac{3}{8}$$

$$= 0.375$$

$$\text{Gini}(t_{AH}) = 1 - (1)^2 - (0)^2 = 0.$$

~~LOGISTIC REGRESSION~~

→ Individual genie values tells us about the purity of the terminal nodes.

→ Savings & Assets are the parent node

To know which of these parent nodes gives us a purer splits, we make use of "Weighted Gini"

$$\text{Weighted Gini}_{(\text{savings})} = \sum \text{Prob} \times \text{Gini}$$

$$= P(\text{Savings} = L) \times \text{Gini}(\text{Savings} = L) + P(\text{Savings} = M) \times \text{Gini}(\text{Savings} = M) + P(\text{Savings} = H) \times \text{Gini}(\text{Savings} = H)$$

$$\left(\frac{3}{8} \times 0.44\right) + \left(\frac{3}{8} \times 0\right) + \left(\frac{2}{8} \times 0.5\right)$$

$$= 0.29$$

$$\text{Weighted Gini}_{(\text{Assets})} = \sum \text{Prob} \times \text{Gini}$$

$$= \left(\frac{2}{8} \times 0\right) + \left(\frac{4}{8} \times \frac{3}{8}\right) + \left(\frac{2}{8} \times 0\right)$$

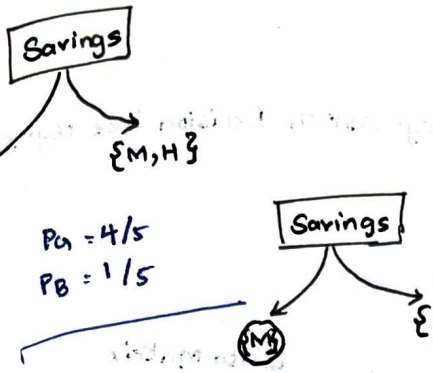
$$= 12/64 = 0.1875$$

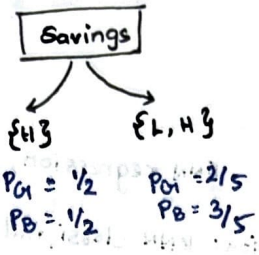
Since Wt. Gini (Assets) < Wt. Gini (Savings), we confirm that splitting on the variables "Assets" gives a purer split.

CREATING A DECISION STUMP USING BINARY SPLIT ONLY (CART MODEL):

→ Cart doesn't allow us to do multiple splits (splits > 2).

To solve the same problem using cart, we need to bifurcate the data as follows (Candidate splits on the variable savings)





$P_G = 1$, $P_B = 0$, $G(M) = 0$

$P_G = 2/5$, $P_B = 3/5$

$G(\{L,H\}) = 1 - (2/5)^2 - (3/5)^2$

$= 0.48$

$= 044$

$wt.Gini = 044$

$wt.Gini = \frac{3}{8} \times 0 + \frac{5}{8} \times 0.48$

$=$

Candidate splits on the variable Assets

Assets

{M,L}

{M,H}

{L,H}

H

L

M

Homework: Find out which candidate split gives a purer subset.

Assets

{L} {M,H}

$P_G = 0/2$, $P_B = 2/2$

$G\{L\} = 1 - (0/2)^2 - (2/2)^2$

$= 0$

$G\{H\} = 1 - (0/2)^2 - (2/2)^2 = 0$

$G\{M,H\} = 1 - (5/6)^2 - (1/6)^2$

Assets

{M} {L,H}

$P_G = 3/4$, $P_B = 1/4$

$P_G = 1/6$, $P_B = 5/6$

$P_G = 2/4$, $P_B = 2/4$

$G\{M\} = 1 - \{(3/4)^2 + (1/4)^2\} = 0.375$

$G\{L,H\} = 1 - (3/6)^2 - (3/6)^2 = 0.5$

$G\{L,H\} = 1 - (1/2)^2 - (1/2)^2 = 0.5$

Assets

{H} {L,M}

$P(H) = 2/8$

$P\{L,M\} = 6/8$

Assets {L}, {M, H}

Wt Gini = $\left(\frac{2}{8} \times 0 + \frac{6}{8} \times 0.277\right)$ Wt. Gini = $\sum$ Prob * Gini

$= 0.207$

Assets {M}, {L, H}

$= \frac{4}{8} \times \left[G(M) + \frac{4}{8} \{L, H\}\right]$

$= 0.437$

Assets {H}, {L, M}

Wt. Gini = $\left(\frac{2}{8} \times 0\right) + \left(\frac{6}{8} \times 0.5\right) = 0.375$

Regression: KNN regression, linear regression, Decision Tree regression.

Classification: KNN classification,

MODEL EVALUATION:

(A) For regression analysis:

Data Matrix

$D = \begin{bmatrix} X & Y \\ & \end{bmatrix}$

M_1: $\hat{y} = a + bx$

$M2 \Rightarrow \hat{y} = a + b_1 x_1 + b_2 x_2$

$M3 \Rightarrow$ 3-NN

$M4 \Rightarrow$ 5-NN

$M5 \Rightarrow$ 

$M6$: 

→ A metric to understand which model is performing the best is required!

→ The most popular model evaluation metrics for regression problems are:

1. MAE (Mean Absolute Error) = $\frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| = \frac{1}{n} \sum_{i=1}^{n} |\varepsilon_i|$

↓ Most interpretable one

It says what is the average deviation of the predicted values from the actual values.

2. MSE (Mean squared Error) = $\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i^2$

3. RMSE (Root Mean Squared Error) = $\sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} = \sqrt{MSE}$

4. MAPE (Mean absolute Percentage Errors) = $\frac{1}{n} \sum_{i=1}^{n} \left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%$

$= \frac{1}{n} \sum_{i=1}^{n} \left|\frac{\varepsilon_i}{y_i}\right| \times 100\%$

MAPE helps us to express the error in percentage form. → $\left(\frac{\text{error}}{\text{actual}}\right) \times 100$

↑ sum of

→ Since all these are error metric the lesser the value, the better it is.

Therefore, among all the models that are fitted on a data, the model with the least value of error should be considered and is expected to give better results.

(B) For Classification models:

We compute something called as confusion Matrix

A confusion matrix is a bivariate table between the actual values and the predicted values. For example, for a binary classification problem with 2 classes "positive" and "Negative", a confusion matrix would look like:

		Predicted	
		+ve	-ve
Actual	+ve	#TP	#FN
	-ve	#FP	#TN

	y	$\hat{y}$	
1.	+	+	← TP
2.	+	-	← FN
3.	-	-	← TN
4.	-	+	← FP
5	+	+	

#TP + #FN = Total no. of actual positive (+ve) cases

#FP + #TN = Total no. of actual negative (-ve) cases

#TP + #FP = Total no. of predicted positive (+ve) cases

#FN + #TN = Total no. of predicted negative (-ve) cases.

TP → Truly classified into +ve class

FN → Falsely classified into -ve class

TN → Truly classified into -ve class

FP → Falsely classified into +ve class

#TP → total no. of true positives

#TN → total no. of true negatives

#FP → toatal no. of false positives

#FN → total no. of false negatives

POPULAR EVALUATION METRICS FOR CLASSIFICATION PROBLEMS:

1. Overall error rate = $\frac{\#\text{mistakes}}{\#\text{obs}} = \frac{\#FN + \#FP}{n}$

2. Overall accuracy = 1 - Overall error = $\frac{\#\text{correct predictions}}{n}$

Overall accuracy = $\frac{\#TP + \#TN}{n}$

3. Sensitivity (or Recall) = True Positive Rate (TPR)

Actual fraudulent transactions } = 150

Out of 150 fraudulent transactions, } 100 were detected right.

Actual \ Prediction	+	−
+	100	50
−	150	9700

False alarm

	+
−	150

→ Even though the transaction is not fraudulent, model predicts it as fraudulent.

100000 transactions

$$\text{Sensitivity} = \frac{\#\text{ correct (+) predictions}}{\#\text{ actual positives}} = \frac{\#TP}{\#TP + \#FN}$$

→ Sensitivity measures the proportion of positive cases that are correctly predicted by the model.

→ In our example, $\text{sensitivity} = \frac{100}{100+50} = \frac{100}{150} = 0.667$

which means 66.67 % of the positive cases are identified correctly by the model.

4. Specificity or True negative rate (TNR)

$$= \frac{\#\text{ correct negative predictions}}{\#\text{ actual}} = \frac{\#TN}{\#TN + \#FP}$$

Specificity measures the proportion of negative cases that are correctly predicted by the model.

In our example, $\text{Specificity} = \frac{9700}{9700+150} = 0.984\ldots$

That is, 98.4 % of the negatives cases are identified correctly by the model.

5. **Precision:** Helps to understand the number of precise predictions.

$$\text{Precision} = \frac{\#\text{ no. of correct (+ve) prediction}}{\text{Total no. of (+ve) prediction}} = \frac{\#TP}{\#TP + \#FP}$$

* Higher precision, lower no. of false alarms

In our example, precision $= \frac{100}{100+150} = \frac{100}{250} = \frac{2}{5} = 0.4$

which means of 40% of the positive-prediction made by the model are correct predictions.

→ Note that, higher the value of the precision, the less will be the no. of false alarms and vice versa.

6. **F1 score (most popular):** HM → harmonic mean

F1 score = H.M (Precision, Recall)

Property of arithmetic mean: A.Mean of 10 and 100 $= \frac{110}{2} = 55$

Mean towards higher value.

It says either precision or recall is high.

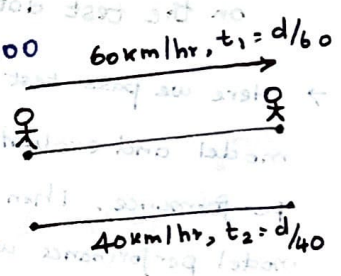
Property of harmonic mean: H.mean of 10 and 100

→ says Both precision and recall is high

Higher value of harmonic mean says higher the both values.

→ Since precision and recall are "rate measures" it's appropriate to use harmonic mean to compute the average of these two rates.

→ Moreover, harmonic mean ensures that, a higher value of F1 score would mean both precision and recall are high.



avg speed = $\frac{\text{Total Dist}}{\text{Total time}}$

$= \frac{2d}{\frac{d}{40} + \frac{d}{60}}$

$= \frac{2}{\frac{1}{40} + \frac{1}{60}}$

$= HM(60, 40)$

avg speed $\neq \frac{60+40}{2}$

In general we don't compute accuracy in training data. Because it gives biased results as models has already seen the training data.

The moment you get the data, divide the data randomly into 2 parts (training and test) even before data exploration. First, secure your test data;

Data exploration and data cleaning

~~VALIDATION~~ TRAIN TEST SPLIT: (TRAIN, VALIDATION TEST SPLIT):

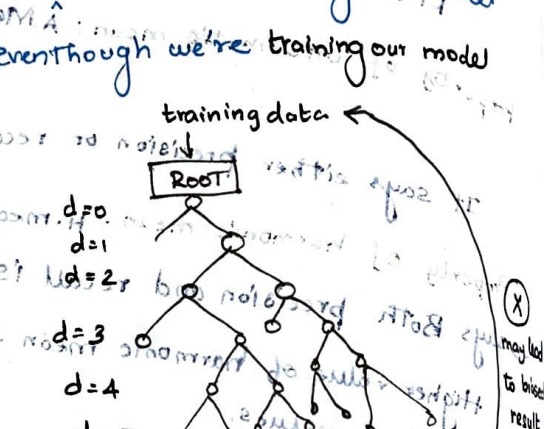
→ We randomly split the data into 2 parts. We call one part as the training data and the other one as test data.

→ The training data is used to train a model and the test data is used to evaluate the out of sample performance of the model.

CAUTION: The test data should never be exposed to the user or to the model until at the final stage of model validation.

→ Somewhere we take whispers from test data and making our model being exposed to test data even though we're training our model on the test data. (*)

→ Here we pass test data into the model and evaluate model performance. Then based on model performance we decide the decide tree to grow more.

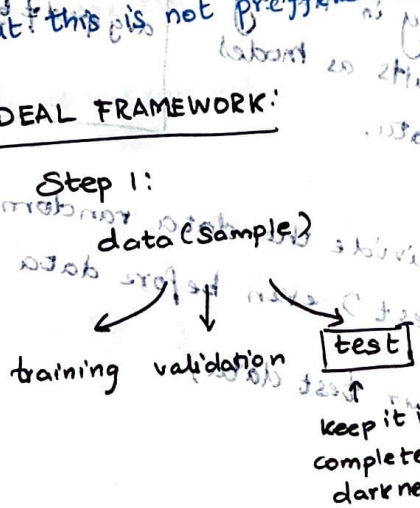


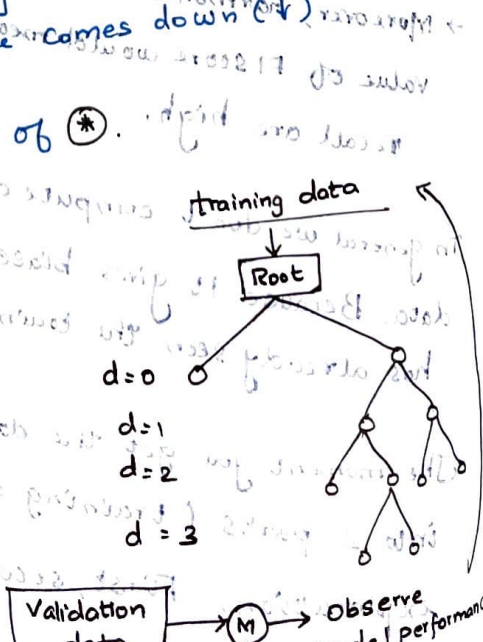
→ Then we observe that when test data is passed model performance is ↑. We come to intuition making decision tree grow results in ↑ in model performance. But once again, when we allowed the tree to grow, the model performance comes down (↓)

→ But this is not preffered because of (*).

IDEAL FRAMEWORK:

Step 1:





Step 2: Data exploration (on training data only).

Step 3: Data cleaning (on training data only).

Only after splitting data; we go with data exploration and then data cleaning.

When sample size is 100, we first split as 80-20 (test ↓ data),

The 80 observations when split results in lesser training data. So we go with cross-validation.

when class distribution is uniform, we make use of SRS.

For regression we can use SRS, since there are no classes.

3. CROSS - VALIDATION:

(A) k-fold cross validation

k is an integer.

Let's say k = 5, then

← training data →

S = | S_1 | S_2 | S_3 | S_4 | S_5 | 5 random and equal splits.

Iteration 1:

hold = S_1 (as validation data)

training (T) = $S_2 \cup S_3 \cup S_4 \cup S_5$

We train our model on T and validate the model on hold.

ALGORITHM:

1. Let S be the training data.
2. Randomly partition S into k equal parts.
   $S_1, S_2, \ldots, S_k$, such that $\bigcup_{i=1}^{k} S_i = S$ (S ↑ training data) & $S_i \cap S_j = \phi$ (ϕ ↓ null) $\forall\ i \neq j$
3. for i from 1 to k do:
4. $h_i = S_i$ (holdout sample)
5. $t_i = \bigcup_{\substack{j=1 \\ j \neq i}}^{k} S_j$ (training sample)
6. m_i = model fitted on t_i
7. e_i = validation score of m_i on h_i
8. end for
9. C.V score = aggregated score of e_i (average).

M_1: $y = a + bx$; S_1 = cvscore1

M_2: dt (depth = p), S_2 = cvscore2

M_3: 5NN ; S_3 = cvscore3

Every time you take data randomly from population and use the data to fit into the regression (for each data we have different parameters for the regression line (because the data changes) for every model we get different values of b_0 and b_1 and

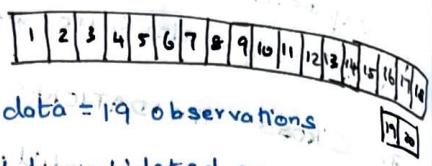
→ Similarly we can also say for classification decisiontree. Each data results in different decision

trees. We compute the error of each models and compute the CV score.

→ Lesser the ~~value~~ size of training data, higher the value of k.

→ If training data = 20 observations and set k = 20.

I'll split such that during every iteration,



validation data = 1 observation and training data = 19 observations which means at a whole my model will be validated on each of the 20 observation (h_i = where i = 1 to 20).

→ If we have ~~3~~ 4 models M_1: logistic, M_2: linear, M_3: decision tree, M_4: 1 KNN, we'll perform cross validation on each of these models and determine the CV score of each model.

→ Suppose M_2 has lesser CV score, we choose M_2 over other models.

(B) LOOCV (Leave one out cross validation)

1. Let S be the training data of n examples.
2. Randomly partition S into n equal parts. $O_1, O_2, \ldots, O_n$ such that $n(O_i) = 1, \forall i$, $\bigcup_{i=1}^{n} O_i = S$, $O_i \neq O_j \ \forall i \neq j$ (No duplicate observations)
3. for i in 1 to n do:
4. $h_i = O_i$ (holdout data)
5. $t_i = \bigcup_{\substack{j=1 \\ j \neq i}}^{n} O_j$ (trainingdata)
6. m_i = model fitted on t_i
7. e_i = validation score of m_i on h_i } inside for loop
8. end for
9. C.V score = aggregated score of e_i

Supervised Machine Learning:

→ we initiate the learning of the algorithm with labelled data

→ But the learning algorithms can be different.

You have a kid and show him square & rectangle. You say that what's length, breadth, if all length are same it's a square. However when lengths > breadth. Then it's a rectangle

→ We show the object to the kid and also we show the learning algorithm.

→ We are supervising the learning process with label.

→ The child has been exposed to the several shapes along with supervision is done using labels and learning algorithm.

→ Discriminate the shape of object.

For every value we're bring into the model, we ~~tell~~ supervise the ~~algorithm~~ learning using the

→ In KNN model Let's say we need to find the y-value of * for a given vector (100, 20, 10). In KNN we find the 5 (k) nearest neighbours by calculating the euclidean distance between * (vector) and all other datapoints (represented as vectors) and find the 5 nearest neighbours. There's no y variable involved while determining the nearest neighbours.

KNN is controversial to be considered as supervised because we don't use the y variable to determine the neighbours.

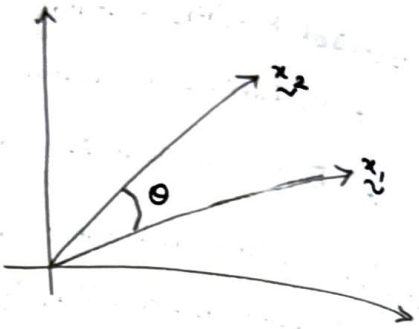
→ We're not definitive what groups to be created, there exists some groups and determine

→ Distance – measure of dissimilarity.

If distance between 2 datapoints is larger, then they are more dissimilar.

→ cos θ - measure of similarity

Cosine between 2 vectors is the measure of correlation between them.



$$\cos\theta = \frac{x_1^T x_2}{\|x_1\| \|x_2\|}$$

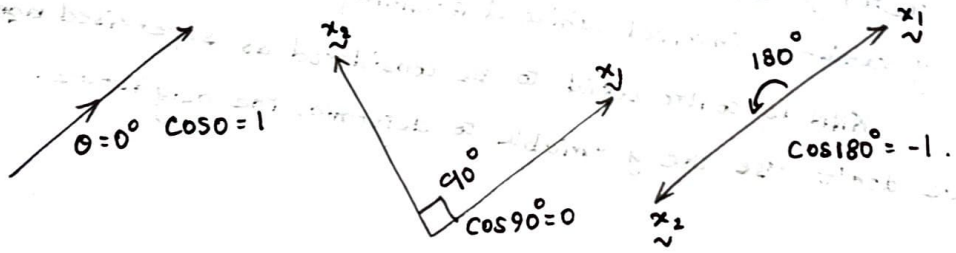
$$= \frac{\sum_{i=1}^{n} x_{1i} \sum_{i=1}^{n} x_{2i}}{\sqrt{\sum_{i=1}^{n} x_{1i}^2} \sqrt{\sum_{i=1}^{n} x_{2i}^2}}$$

$$\cos\theta = \frac{\sum_{i=1}^{n} x_{1i} x_{2i}}{\sqrt{\sum_{i=1}^{n} x_{1i}^2} \sqrt{\sum_{i=1}^{n} x_{2i}^2}} \quad - (1)$$

$$\text{Cor}(x_1, x_2) = \frac{\sum (x_{1i} - \mu_1)(x_{2i} - \mu_2)}{\sqrt{\sum (x_{1i} - \mu_1)^2} \sqrt{\sum (x_{2i} - \mu_2)^2}} \rightarrow *$$

When $\mu_1 = \mu_2$ is 0, then (*) becomes equal to (1).

Special case of correlation coefficient is cosine similarity



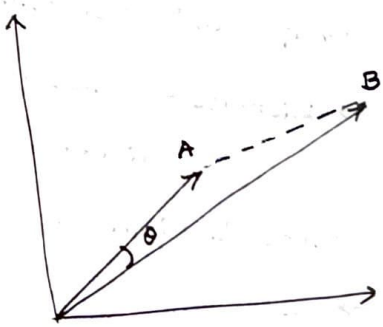
Cosine similarity → −1 to +1. → most similar.

↓ most dissimilar

↦ Netflix recommendation systems

Cosine similarity:

Customer A and B have the affinity to buying products 1 and 2, hence they are more similar. (Group customers based on similarities



Euclidean distance:

Customer A is light buyer and customer B is a heavy buyer. Customer A buys much less than customer A. Distance between them is high (i) more dissimilar.