Machine Learning.

Date / /

Page No.

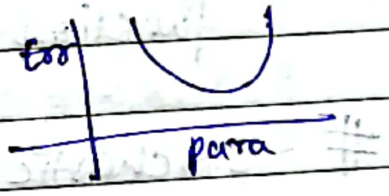
Gradient descent (Batch)

↳ optimisation algorithm

↳ find values of parameters (coeff) of function that minimises cost

create a plot with error (cost/loss func) vs parameter.

↳ aims to reduce error and find optimal value of parameter.



In straight line there are 2 parameters and need to optimise both.

↳ we take ~~small~~ step sizes and reduce the error.

delta = derivative (Cost function)

Step size = (Learning rate)*(delta)

Learning rate controls how much the coefficients can change on each update.

When to stop?

↳ step size ~~=0~~ close to zero (<0.001)

↳ total steps (epochs), usually >1000

Date / /
Page No.

In gradient descent, for linear regression, the cost / loss function is the residual error
$= (Y - Y_{pred})^2$
i.e. ~~sum~~ mean of all squared errors.
the derivative is taken of this function wrt the parameters.

Stochastic Gradient descent
↳ for big dataset the regular gradient descent is slow as there is a lot of calculation in each iteration, to reduce this
↳ select a point randomly and apply algo instead whole data set
↳ usually used when redundant data ~~set instances~~

schedule:
↳ The way learning rate changes from relatively high to relatively low.

mini batch
↳ selects some points (small subset) randomly in each iteration.

Date / /
Page No.

Note: If there is a new point we can just apply the algo from the point we left (i.e. using the previously obtained m, c) instead of ~~resetting~~ resetting the values of parameters.

pros of Batch Gradient
↳ More stable convergence and error gradient than stochastic
↳ direct path is taken towards min

Cons
↳ Can converge at local minima
↳ Slow learning rate since an update is performed only after going thru all observation.

Pros of mini
↳ more stable convergence than stochastic
↳ computation efficient
↳ fast learning as more freq updates

Cons
↳ need to figure mini batch size hyperparameters.

Date / /
Page No.

Pros ~~and~~ of stochastic
↳ easy to fit in memory
↳ likely to reach near min faster that Batch incase of large dataset
↳ freq updates
↳ creates plenty of oscillations which can be helpful for getting out of local mins.

Cons
↳ go in wrong directions
↳ frequent updates are computationally expensive due to using all resources for processing one training sample at a time.

Formulas / steps

$$\underset{=e}{fn} = \frac{(Y - Y_{pred})^2}{n}$$

$$\frac{de}{dm} = \frac{d(Y - (mx+c))^2}{dm \quad n}$$

$$= d \quad -\frac{2}{n}(Y - (mx+c))(x)$$

$$= \frac{-2}{n}(Y - Y_{pred})(X)$$

$$\frac{de}{dc} = \frac{-2}{n}(Y - Y_{pred})$$

Note: capital Y, Y_{pred}, X means they are series of data.

Date / /
Page No.

$m = m - L\frac{de}{dm}$

$c = c - L\frac{de}{dm}$

$\frac{de}{dm}$ may be written as $dm \rightarrow delta(m)$

LOGISTIC REGRESSION

↳ use to predicts Binary outputs
  ↳ True of false
  ↳ for classification
↳ instead of straight line a Sigmoid function is drawn

$$[\ sig(t) = \frac{1}{1+e^{-x}} \quad [0, 1]$$

Here:
Data is fit into linear regression model, which then be acted upon a logistic function predicting the target categorical variable.

Type
↳ Binary → 2 outcomes (T or F)
↳ Multinomial → 3 or more without ordering
↳ Ordinal → 3 or more with ordering

Date / /

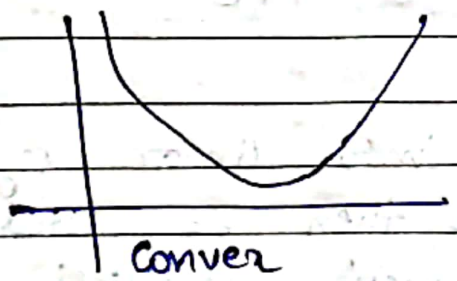
Page No.

Decision boundary :

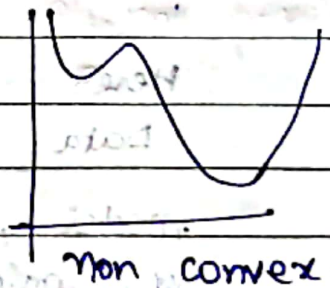
↳ To predict which class a data belongs a threshold can be set

↳ can be linear or non linear

In logistic, ~~mean~~ Squared ~~mean~~ error - Cannot be taken as cost function because then it will be a non convex function of parameters. Gradient descent will converge into global minimum only if function is convex.



Convex



Non convex.

Wald's test

↳ To know if variable helps in prediction or not.

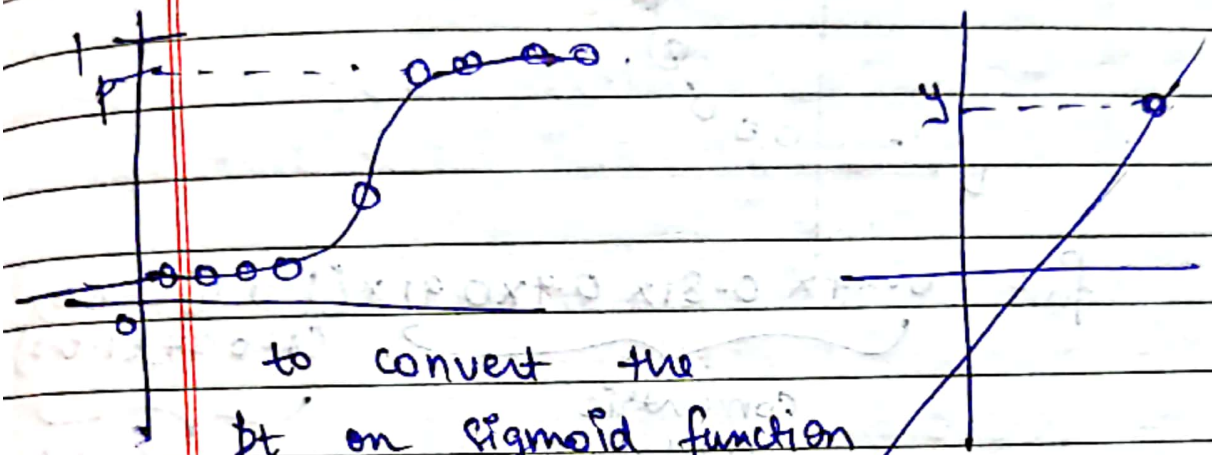
~~Totes useless~~ "Totes useless" → not helping

Here loss function is → max likelihood function.

Date / /

Page No.

Logistic Regression is a part of Generalized linear model.



to convert the pt on sigmoid function i.e p on a straight line, i.e. y

$$y = \log\left(\frac{p}{p-1}\right)$$

} Transformation.

at $p=1 \Rightarrow y=\infty$

$p=0 \Rightarrow y=-\infty$

$p=0.5 \Rightarrow y=0$

Reversely

$$p = \frac{1}{1+e^{-y}}$$

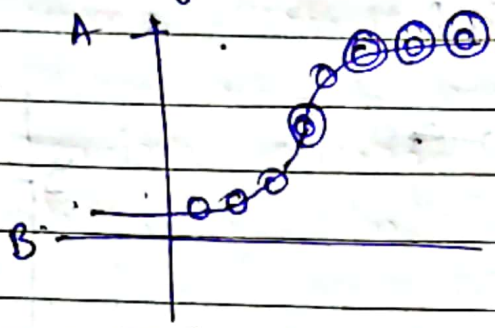
Coefficient are ~~the~~ Intercepts and slope (Linear line)

Coeff	estimated	Standard error	z value, $z = \frac{Est}{e}$
intercept	c		} Wald's test
slope	m		

Date / /

Page No.

calculating values of likelihood function



$f_n = \underbrace{0.49 \times 0.81 \times 0.9 \times 0.9}_{\text{Concentric circle probability of getting A}} \times \underbrace{(1-0.6)(1-0.3)(1-0.27)(1-0.2)}_{\text{Single circle}}$

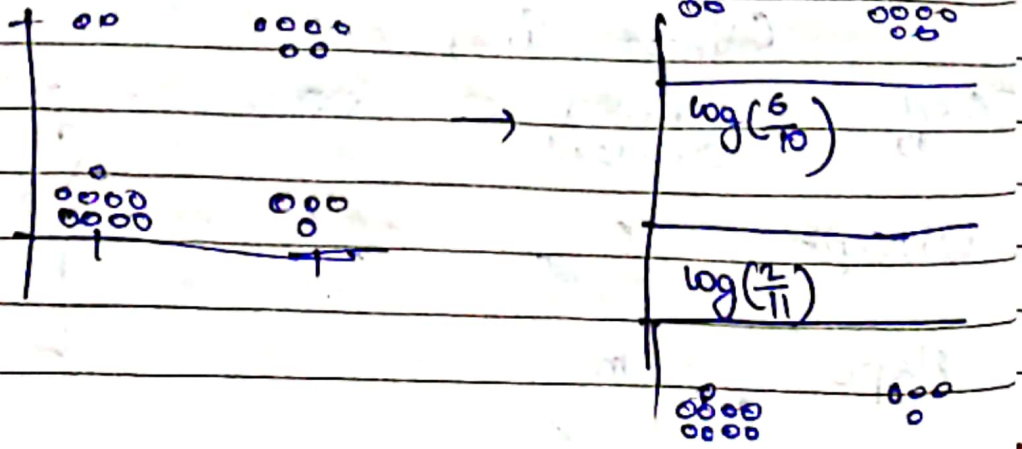
~~This done to scale it in the same term~~ i.e probability of not getting B.

→ we can also take log of whole thing to analyse

→ aims to get maximum likelihood.

Logistic regression in terms of discrete

↳ similar to t-test



Date / /
Page No.

size = log(odds normal) β_1 + β_2 (log(odd mutant / odd normal))

Note:

↳ logistic function = sigmoid function

↳ features can be categorical or continous but outcome is discrete

↳ linear decision boundary = hyperplane

↳ in equation

$$z = \frac{1}{1+e^{-y}} \quad \left\{ y = mx + c \right.$$

m controls slope of rise

c controls location of midpoint

pros

↳ Makes no assumption about distribution of classes in feature space

↳ easily extanable to multiple classes (multinomial Regression)

↳ Natural probabilistic view of class prediction

↳ quick to train

↳ very fast at classifying unknown records

↳ Good accuracy

↳ Resistant to overfitting

↳ can find useful variables (important coefficients "wald's test")

- dis

↳ linear decision boundary.

Date / /
Page No.

↳ assumes linear relationship among dependent and independent
↳ dependent is bound to be discrete.
↳ requires average or no multi-collinearity b/w independent vars.

DECISION TREE (DT)
↳ used to visually and explicitly represent decision and decision making.
↳ predictive modelling, based on branch series
classification trees → classify
regression trees → continous values

↳ refered to as CART (Class^n and reg^n trees)

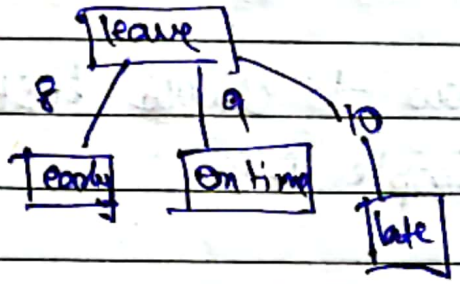
Recursive Binary Splitting
↳ All features are considered and different split points are tried and tested using a cost function.
↳ best cost one is selected.

DT uses "inductive learning task"
↳ use particular facts to make more generalised conclusions.

Date / /
Page No.

We can also represent as a set of rules.

DT



rules

if leave == 8
  Status = early
else if leave == 9
  Status = on time
else if leave == 10
  Status = late.

ID3 → Iterative Dichotomiser 3
↳ classification algo
↳ greedy
↳ building DT
↳ selects best attribute i.e. yields maximum information gain (IG) or minimum entropy (H)

common terms in DT
Node → feature / attribute
branch → decision / rule
leaf → outcome.

DT is constructed in top-down recursive divide and conquer manner in ID3 (no backtracking)

Date / /
Page No.

pseudo code
↳ ppt "Lec9 Decision Tree" . page 14

ID3's bias (How to choose best DT)
↳ prefers simple DT

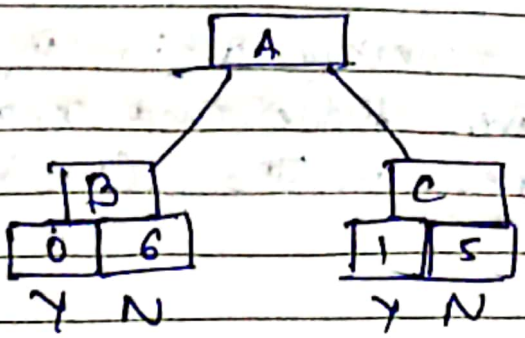
Entropy (H)
↳ measures the disorderness/impurity
↳ minimum (0) when all records belongs to one class i.e. most information
↳ maximum log(C) [in case of Binary 1] when records are equally distributed among all classes
↳ small entropy → more pure

* pure → when records ~~of~~ belong to same class.

Entropy (t) = H(t)

$$= -\sum_j p(j/t) \log_2 p(j/t)$$

ex:



Date / /
Page No.

$$H(B) = -\frac{0}{6}\log\left(\frac{0}{6}\right) - \frac{6}{6}\log\left(\frac{6}{6}\right) = 0$$

$$H(L) = -\frac{1}{6}\log\left(\frac{1}{6}\right) - \frac{5}{6}\log\left(\frac{5}{6}\right) = 0.65$$

$p(j|t)$ → relative frequency of class j at node t.

Information Gain (IG)

$$IG = H(p) - \left(\sum_{i}^{k} \frac{n_i}{n} H(i)\right)$$

$n = \sum n_i$

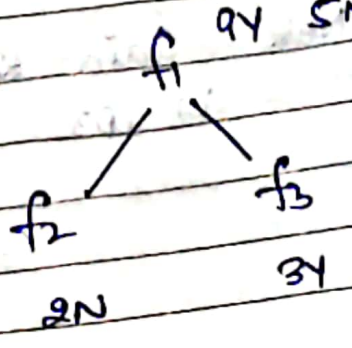
i.e. split p into k partitions and n_i is the no. of records in i^{th} partition.

↳ aim to max this

↳ measures reduction in H

↳ con: tends to prefer splits that results in large no. of partition each small.

example:



Date / /

Page No.

$$IG = H(f_1) - \frac{8}{14} H(f_2) - \frac{6}{14} H(f_3)$$

$$= -\frac{9}{14}\log\left(\frac{9}{14}\right) - \frac{5}{14}\log\left(\frac{5}{14}\right) - \left[\frac{8}{14}\left(-\frac{6}{8}\log\left(\frac{6}{8}\right) - \frac{2}{8}\log\left(\frac{2}{8}\right)\right) - \frac{6}{14}\left(-\frac{3}{6}\log\left(\frac{3}{6}\right) - \frac{3}{6}\log\left(\frac{3}{6}\right)\right)\right]$$

$$IG = 0.049$$

To overcome the con of IG

↳ Gain Ratio

↳ $\frac{IG}{\text{Split Info}}$

$$\text{Split Info} = -\sum_{i}^{k} \frac{n_i}{n} \log\left(\frac{n_i}{n}\right)$$

(Terms same as IG)

↳ this adjusts GAIN (IG) by the entropy of partitioning

↳ Higher entropy partitioning is penalised

↳ used by C4.5 (extension of ID3)

Note: Gain Ratio is not used by ID3

↳ ID3 only used IG.

Date / /
Page No.

Another ✱ Criteria to measure impurity

↳ GINI index for node t

$$GINI(t) = 1 - \sum_j p(j|t)^2$$

{ same $p(j|t)$ as H }

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

↳ aims to minimise GINI split value

↳ min 0 → best

↳ max $1 - \frac{1}{n_c}$ → worst

↳ used by CART, SLIQ, SPRINT

Specify test conditions

- attribute type
  - ~~Nom~~ nominal
  - → ordinal
  - → continuous
- no of ways to split
  - → Binary
  - → multi~~nomial~~ way

Date / /

Page No.

overfitting and underfitting
↳ similar to regression models

overfitting results in DT that are
more complex ~~o~~ than necessary
↳ Tree growth went too far
↳ Number of instances gets smaller
(more leaves match single example)

↳ Training error → not good estimate

To avoid overfitting
↦ pre prune
→ post prune

DT

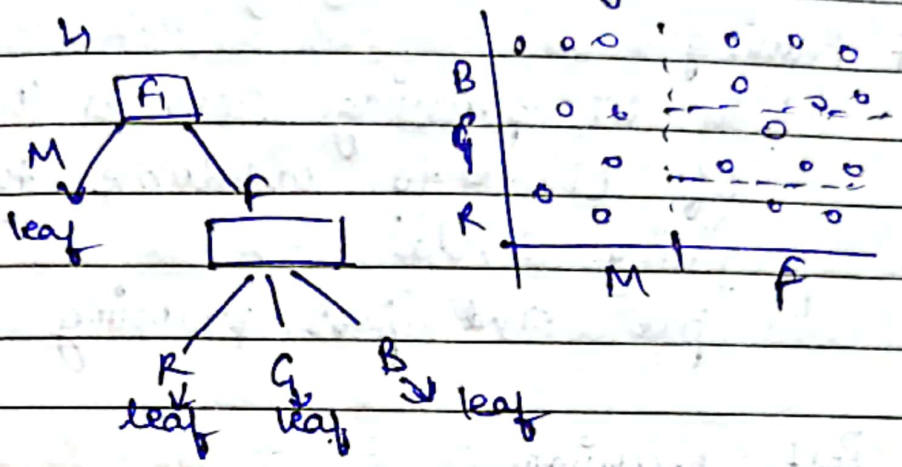
pros
↳ Inexpensive to construct
↳ Extremely fast in classification
↳ easy to interpret (for small)
↳ Good accuracy

cons
↳ Axis parallel decision boundary
↳ Redundancy
↳ need data to fit in memory
↳ need to re train new data

Date / /
Page No.

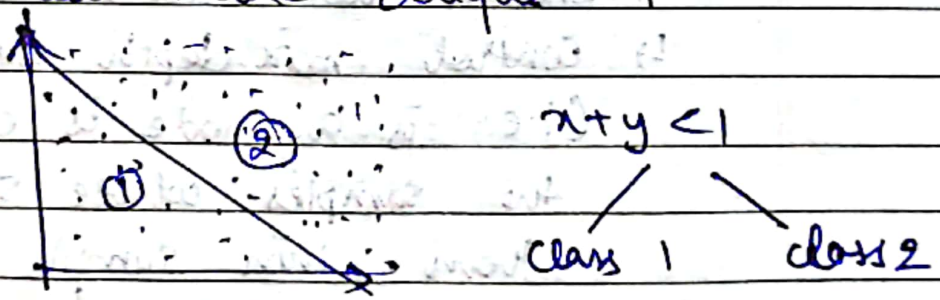
axis parallel boundary

↳



Decision surfaces are axis aligned.

Can also make oblique DT



↳ Test cond^n → more attributes
↳ expensive to find optimal test cond^n
↳ more expressive representation.

Note:
Subtree replication
↳ Same subtrees appears in multiple branches.

Date / /
Page No.

Pruning
↳ In DT pruning. removes the branches of DT to overcome the over-fitting condi
↳ pre and post pruning

Post pruning
↳ use after construction of DT
↳ use when very large depth
↳ known as backward pruning
↳ control max depth and min samples (i.e. is a node is classifying the samples whose size is less than min sample cond^n)

Pre pruning
↳ used before construction of DT
↳ done using hyper parameter tuning
↳ overcome the overfitting issue

Hyper parameter tuning
↳ In begining we don't know which criterion (GINI or ENTROPY), what min depth, what min sample size should be used.
↳ use range of values for each parameter and find the best value.

Date / /
Page No.

pros of Pre and post
↳ creates simple and interpretable trees
↳ reduces overfitting
↳ since pruning selects best cross validated subtrees, pruned trees tends to fit the data well.

cons of both
↳ may eliminate some interesting information

Pre vs post

Pre	Post
↳ faster	↳ slower
↳ may not give good results as the approach is greedy and it may ignore splits that have subsequent imp splits	↳ results in better pruned tree (cross validation is performed at every step)
↳ forward pruning	↳ backward pruning

Measures for performance

Date / /
Page No.

Bias: The inability of ML methods like linear regression to capture the true relationship (i.e. curved line)

variance: The difference in fits b/w data sets is called variance. (b/w training and testing)

Note: 3 commonly used methods for finding sweet spot b/w simple and complicated model.

- ↳ regularization
- ↳ boosting
- ↳ bagging

High bias → underfitting of training
↳ over generalising

High variance → overfitting

Total error = $Bias^2$ + variance + irreducible error

We need to trade off bias and variance such that model doesn't suffer from over or under fitting.

Date / /

Page No.

L2

Ridge Regression [L2 Regularisation]

↳ If the model is going in overfitting we introduce some amount of bias to help reducing the variance.

linear regression ~~introduces~~ minimises ~~sum~~ of squared errors by finding mean ~~to~~ values of parameters

by Ridge regression minimizes mean squared error + $\lambda(\text{slope})^2$

By doing this we find better fit. to the find the optimal value of λ we ~~&~~ use cross validation (usually 10 fold)

$\lambda(\text{slope})^2$ → Ridge regression ~~penalty~~ penalty.

↳ Ridge Regression works for continuous and discrete.

Date / /

Page No.

L1

: Lasso Regression.

↳ Lasso Regression penalty =
$\lambda * |slope|$

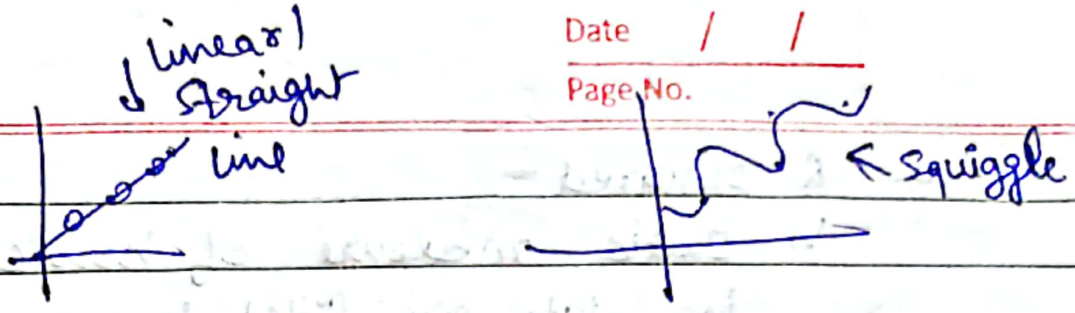
↳ can be applied to same content as Ridge Regression

Ridge Regression can shrink the slope asymptotically close to 0 while Lasso regression can shrink the slope to 0.

* Lasso Regression can exclude useless variable from equations, it is little better than Ridge Regression at reducing the variance in model that contains a lot of useless variables.
but ridge better where there are more useful.

Note: As we increase λ, slope ↓ but not to 0 ~~ever~~ in L2
but there is a kink at 0 in L1 as we increase λ.

Date / /
Page No.



Regression analysis
↳ form of predictive modelling tech which investigates relationship b/w dependent and independent variables.

Types
↳ linear
↳ logistic
↳ polynomial.

linear regression selection criteria
↳ classification and regression capability
↳ data quality
↳ computational complexity

Formula
⇒ $y = mx + c$

$$m = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sum (x-\bar{x})^2}$$

find c using m, $\bar{y}$, $\bar{x}$ in eqn.

Date / /
Page No.

R squared

↳ static measure of how close the data are fitted to regression model.

↳ coefficient of determination

↳ " " " multiple "

↳ cousin of R (correlation), just square of it.

if R^2 of a = 0.8 and R^2(b) = 0.4 then that means b is 2x better than a (but can't say with R)

$R \in [-1, 1]$ [-1, 1 → best
0 → worst)

$R^2 \in [0, 1]$ 0 → worst
1 → best.

formula

$$R^2 = 1 - \frac{RSS}{TSS}$$

$$RSS = \sum (y_p - y)^2$$

$$TSS = \sum (y - \bar{y})^2$$

y → actual

$\bar{y}$ → mean

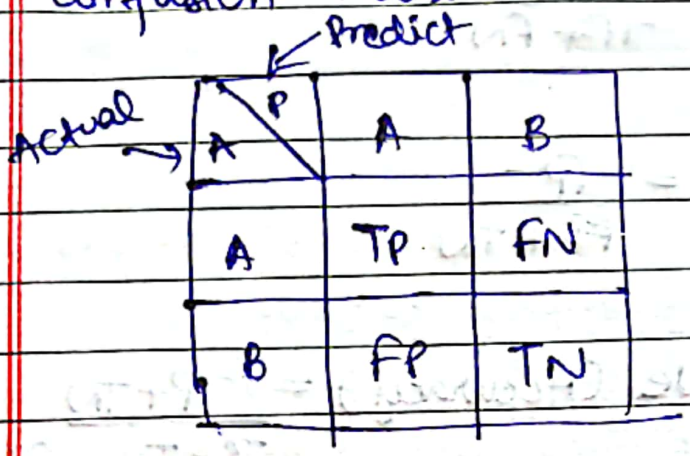
y_p = predicted.

low R^2 is not always bad because in some cases prediction is tough and a 0.5 R^2 is also acceptably good.

Date / /

Page No.

Confusion Matrix.



TP → True positive

FN → false negative

FP → false positive

TN → True negative

How to remember?

Draw fig, mark A → +, B → −

~~to tell~~

	A(+)	B(−)
A(+)	(+)(+) +	(+)(−) −
B(−)	(−)(+) +	(−)(−) −
	− +	+ −
	F P	T N

I term ↓ ↑ II term

~~I term~~ → false/True Positive/Negative

easy.

Date / /

Page No.

TP rate = $\frac{TP}{TP + FN}$

FP rate = $\frac{FP}{FP + TN}$

Succes rate (Accuracy) = $\frac{TP + TN}{TP + TN + FP + FN}$

Error rate = 1 − Succes rate

Sensitivity = $\frac{TP}{TP + FN}$ } measures ↳ detect +ve class

Specificity = $\frac{TN}{TN + FP}$ } not detecting too many false positive

~~Specifity~~ ↳ used to confirm results of sensitivity.