# Machine learning

## Simple Linear Regression

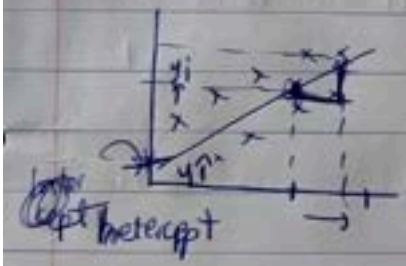
Supervised ML → Regression — O/p → Continuous
→ Classification — O/p → Binary, multiclass categories

**Dataset**

| Weight | Height |
|---|---|
| 74 | 170 |
| 80 | 180 |

Train
New Weight → model → New height

actual → Predicted
Error → $\{y_i - \hat{y}_i\}$




$\hat{y} = mx + c$

$h_\theta(x) = \theta_0 + \theta_1 x$

$\theta_0$ = Intercept

$\theta_1$ = Slope (or) coeffient

**Cost function**

$$J(\theta_0, \theta_1) = \frac{1}{n}\sum_{i=1}^{n} (y_i - h_\theta(x))^2$$ → mean squared error

$y_i$ = actual value

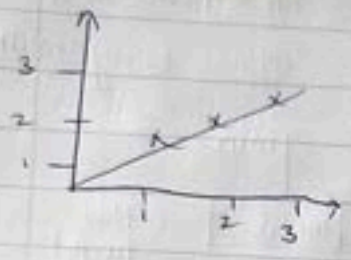
$h_\theta(x)$ = predictor value

$n$ = no. of datapoint

(To get best fit line)

Final aim → Minimize Cost fuction $J(\theta_0, \theta_1) = \frac{1}{n}\sum_{i=1}^{n} (y_i - h_\theta(x))^2$

Optimization : {minize the cost function}




$h_\theta(x) = \theta_0 + \theta_1 x_i$

$\theta_0 = 0$

$\boxed{h_\theta(x) = \theta_1 x_1}$

Let $\theta_1 = 1$

$x = 1 \quad h_\theta(x) = 0 + 1(1)$

$h_\theta(x) = 1$

$x = 2 \quad h_\theta(x) = 0 + 1(2)$

$h_\theta(x) = 2$

| x | y | $h_\theta(x)$ |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 3 | 3 |

Let $\theta_1 = 0.5$

$h_\theta(1) = 0 + 1(0.5)$

$= 0.5$

$h_\theta(2) = 0 + 0.5(2)$

$= 1$

$= 0 + 0.5(3)$

$= 1.5$

Cost fuction

$$J(\theta_1) = \frac{1}{n} \sum_{i=1}^{n} \overbrace{(y_i - h_\theta(x))^2}^{\text{error}}$$

$n = 3$

$$= \frac{1}{3} \left[ (1-1)^2 + (2-2)^2 + (3-3)^2 \right]$$

$J(\theta_1) = 0$

$\boxed{\theta_1 = 0.5}$

$$= \frac{1}{3} \left[ (1-0.5)^2 + (2-1)^2 + (3-1.5)^2 \right]$$

$$= \frac{1}{3} \left[ 0.25 + 1 + 2.25 \right]$$

# Logistic Regression

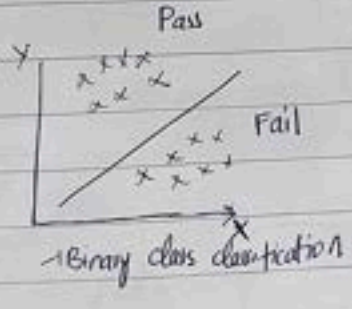
To solve classification
- Binary classification → O/P → 2 categories
- Multiclass classification → O/P → >2 categories

Dataset

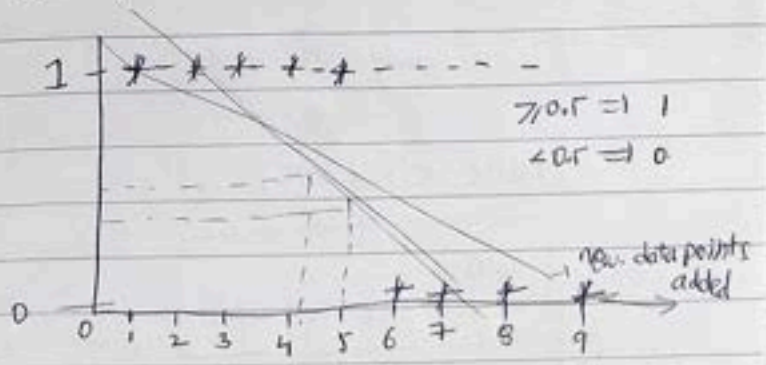
| No. of play hours (Independent feature) | Pass/Fail (y) (O/P or dependent feature) |
|---|---|
| 9 | 0 Fail |
| 8 | 0 Fail |
| 7 | 0 Fail |
| 6 | 0 Fail |
| 5 | 1 Pass |
| 4 | 1 Pass |

Train → Model → Accuracy

New data → Model → P/F




Can we solve this classification problem using Regression?




if new data points added (or having outliers) our predictions are going wrong.

Why we cannot used linear Regression for classification.

① Best fit line changes because of outliers → prediction goes wrong

② The outcomes comes >1 and <0

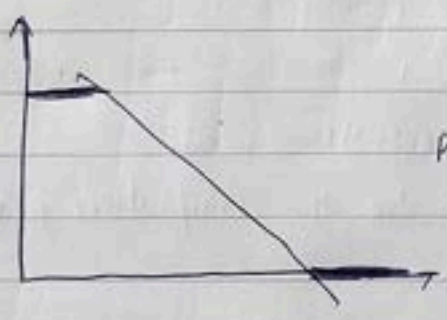
⇓

Logistic Regression

↓

0 to 1 → Squashing Technique

How Logistic Regression ~~squares~~ solves classification problem



$h_\theta(x) = \theta_0 + \theta_1 x_1$ → Best fit line

Prediction

⇓

Signoid Actoration function

⇓

0 to 1

Squashing the best fit line

⇑ ⇓ z

$h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1)$

$$\sigma = \frac{1}{1+e^{-z}} \Rightarrow 0 \text{ to } 1$$

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1)}}$$

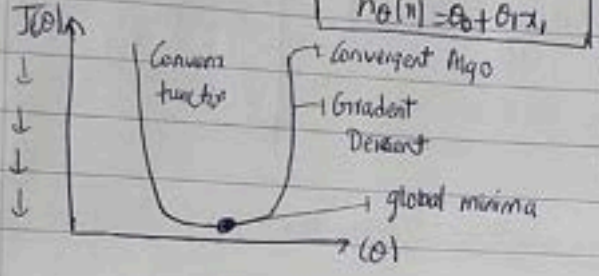
# Cost function

$\hat{y}_i$ → predicted

## Linear Regression cost function

$$J(\theta_0, \theta_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x))^2$$

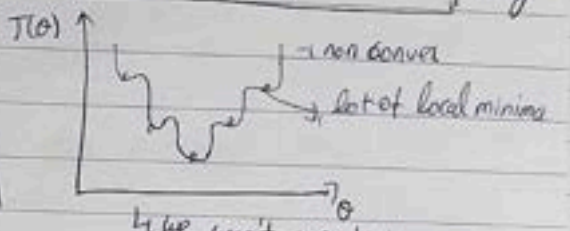
$$h_\theta(x) = \theta_0 + \theta_1 x_1$$




## Logistic Regression cost function

$$J(\theta_0, \theta_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2$$

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1)}}$$ → 0 to 1 only




↳ we won't use this one of cost function

## log loss function

$h_\theta(x)_i$ = predicted value  $y_i$ = actual data

$$J(\theta_0, \theta_1) = -y_i \log(h_\theta(x)_i) - (1 - y_i) \log(1 - h_\theta(x))$$ → cost function

If $y_i = 1$

$$J(\theta_0, \theta_1) = \begin{cases} -\log(h_\theta(x)_i) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

**Final Aim** Minimize cost function $J(\theta_0, \theta_1)$ by changing $\theta_0$ & $\theta_1$

## Convergence Algorium

Repeat until convergence

{

$\theta_j : \theta_j - \alpha \frac{\partial}{\partial \theta} J(\theta_0, \theta_1)$

}

## Logistic Regression with Regularization Parameter

### Cost function

$J(\theta_0, \theta_1) = -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x))$

$h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1)}}$

$$J(\theta_0, \theta_1) = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$

Ridge
Reduce Overfitting

$J(\theta_0, \theta_1) = -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x)) + L_2 \text{ Regulization}$

$J(\theta_0, \theta_1) = -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x)) + L_1 \text{ Regulization}$

$L_1$ feature selection
→ lasso

$J(\theta_0, \theta_1) = -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x)) + L_2 \text{ reg} + L_1 \text{ reg}$ → elasic.

$\lambda$ = hyperparameter

## $L_2$ Regularization → Reduce overfitting

$$J(\theta_0, \theta_1) = -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x)) + \lambda \sum_{i=1}^{n} (\text{slope})^2$$

## $L_1$ Regularization

$$J(\theta_0, \theta_1) = -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x)) + \lambda \sum_{i=1}^{n} |\text{slope}|$$

## Elastic Net

$$J(\theta_0, \theta_1) = -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x)) + \lambda \sum_{i=1}^{n} (\text{slope})^2 + \lambda \sum_{i=1}^{n} |\text{slope}|$$

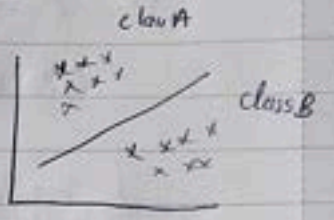
$$C \propto \frac{1}{\lambda} \qquad C = 2.0 \quad \lambda = \frac{1}{2}$$

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2)

from sklearn.preprocessing import StandardScaler → z-score mean, std.
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)

from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state=0).fit(X_train_scaled, y_train)
```

# Performance Metrics : Accuracy, Precision, Recall & F-Beta




## Topics to covered

1. Confusion Matrix
2. Accuracy
3. Precision
4. Recall
5. F-Beta Score

## DATASET

| $x_1$ | $x_2$ | $y$ | $\hat{y}$ Actual value → predict | |
|---|---|---|---|---|
| — | — | 1 | 0 | 1 → Wrong Prediction |
| — | — | 1 | 1 | 1 → correct |
| — | — | 0 | 0 | 0 → correct |
| — | — | 1 | 1 | 1 → correct |
| — | — | 1 | 1 | 1 → correct |
| — | — | 0 | 1 | 1 → Wrong |
| — | — | 1 | 0 | |

## 1. Confusion matrix

| predicted value ($\hat{y}$) \ actual values ($y$) | 1 | 0 |
|---|---|---|
| 1 | 2+1 | 1+1=2 |
| 0 | 1 | 1 |

accurate data

| Predicted \ Actual | 1 | 0 |
|---|---|---|
| 1 | TP | FP |
| 0 | FN | TN |

True positive
False positive
False negative
True Negative

$$\text{Model Acc} = \frac{TP + TN}{TP + FP + FN + TN}$$

$$= \frac{3+1}{3+2+1+1} = \frac{4}{7} \quad 57.1\%$$

~~③ Precision~~

DATASET → Imbalance dataset

1000 datapoints → 900 → 1, 100 → 0 } Imbalanced dataset

| actual \ Predict | 1 | 0 |
|---|---|---|
| 1 | 900 | 100 |
| 0 | 0 | 0 |

Ex Dum model → 1

Acc = 90% ✓ getting wrong

so are another 3.

③ Precision = $\frac{TP}{TP+FP}$ } Out of all the actual value how many are correctly predicted

| actual \ Predy | 1 | 0 |
|---|---|---|
| 1 | TP | FP |
| 0 | FN | TN |

FP is important → To reduce FP ↓↓

④ Recall = $\frac{TP}{TP+FN}$ } out of all the predicted value how may are predicte with actual values

| | 1 | 0 |
|---|---|---|
| 1 | TP | FP |
| 0 | FN | TN |

FN is Important → To reduce FN ↓↓

Use case 1 → Spam Classification

Text → Model → Spam / Not Spam

| | 1 | 0 |
|---|---|---|
| 1 | TP | FP |
| 0 | FN | TN |

TP ← Mail → Spam (1), Model → Spam (1)

TN ← Mail → not a spam (0), Model → Not a spam (0) }

Important ← FP ← Mail → Not a spam (0) } Wrong prediction (Blunder)
Model → spam (1)

FN → Mail → spam (1) } Wrong prediction
Model → Not a spam (0)

→ We use precision
↓
→ Usecase 2 → FN is Important

To predict whether a person has diabetes (or) not.

| | Diabetes (1) | No Diabetic (0) |
|---|---|---|
| Diabetes (1) | TP | FP |
| no Diabetes (0) | FN ↓↓ | TN |

↓
Recall
$\frac{TP}{TP+FN}$

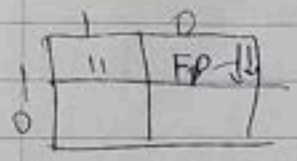
TP ← Actual → Diabetes
Model → Diabetes } Correct

TN ← Actual → no Diabetes
Model → No Diabetes } Correct

FP → Actual → no Diabetes (0)
Model → Diabetes (1) } Wrong Prediction

FN → Actual → Diabetes (1)
Model → No Diabetes (0) } Blunder

Assignment :- Tomorrow the stock market will crash or not




Predicted + it will crash. 1

Predict it will not crash. 0.

No big problem

FP ⇓ → Precision → $\frac{TP}{TP+FP}$

⑤ F + Beta score $= (1+\beta^2)\frac{\text{Precision} + \text{Recall}}{\text{Precision} + \text{Recall}}$

① If FP and FN are both are important

$\beta = 1$

$$F_1 \text{ score} = (1+1)\frac{P*R}{P+R}$$ → Harmonic mean

② If FP is more important the FN.

$\beta = 0.5$

$$F_{0.5} \text{ score} = (0.1+0.25)\cdot\frac{P*R}{P+R}$$

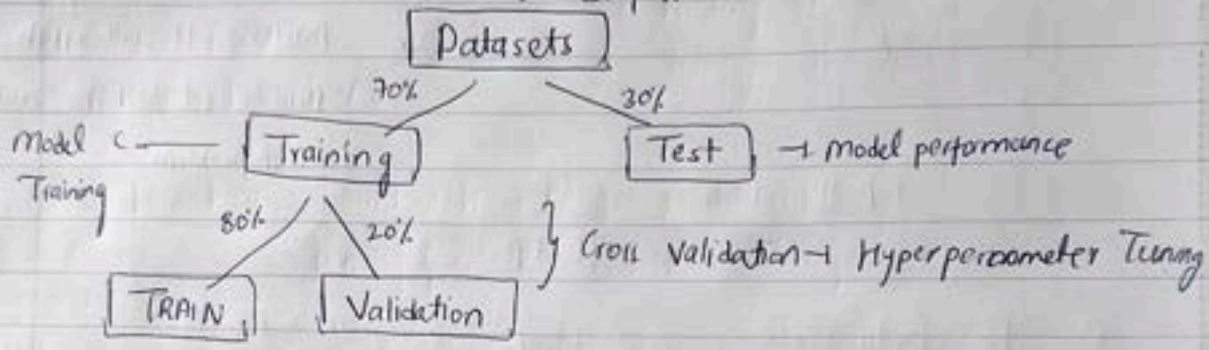
③ If FN is more important the FP

$\beta = 2$

$$F_2 \text{ score} = (1+4)\frac{P*R}{P+R}$$

# Cross validation and its Types.

→ this is used further in hyperparamert tuning




Random_state

[ 85%, 75%, 70%, 60% ] → to find Avegege accuray → we do Cross validation)

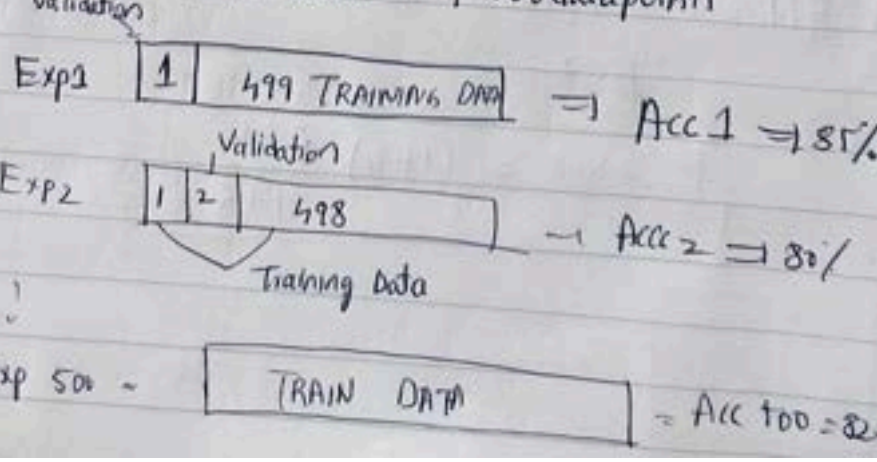
TRAINING → CV = 5

| | | | | |
|---|---|---|---|---|
| Exp1 | TRAIN | VALIDATION | Acc1 | |
| Exp | TRIN | VAILIDATION | Acc2 | Mean |
| ! | ! | ! | ! | |
| Exn | TRAIN | | | |

## Types of cross validation

### ① Leave One Out Cross Validation (LOOCV)

TRAINING DATA → 500 datapoints



Disadvantage

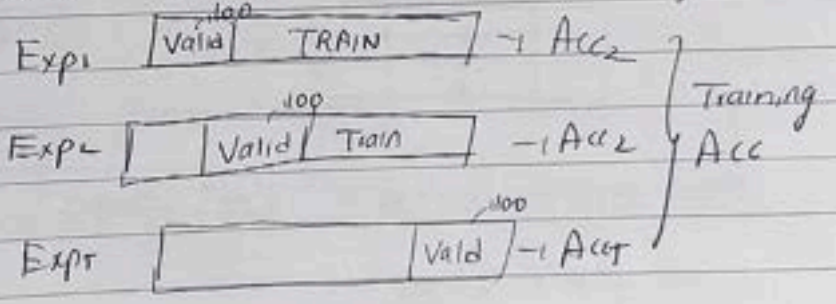
① Time complexity is huge for Training dataset

② Model overfit → Training Acc ↑↑
New data → Acc ↓↓

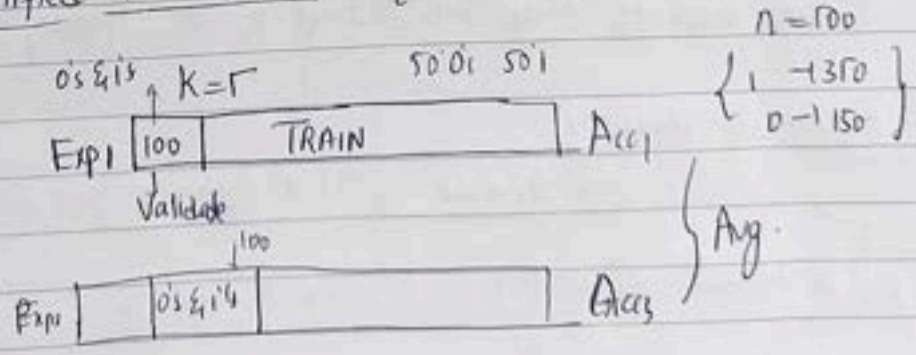
② → leave p out CV — p = 10, 20, 100.
p = hyperparameter

③ K fold cross validation

K = 5 n = 500 Validation data = $\frac{500}{5} = 100$




④ Stratified k Fold C.V {Imbalanced Dataset}



# Hyper parameter Tuning with cross validation

↳ Finding the best parameter while training the model

① Grid search CV
② Randomized search CV

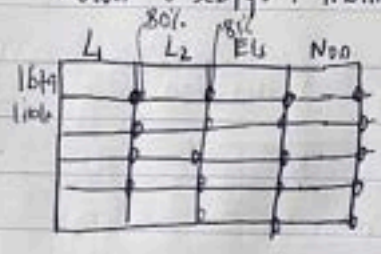
## ① Grid search CV [Grid search + Cross validation]

[2 combination]

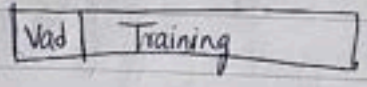
penalty {'l1', 'l2', 'elastic net', None}

This say which parameter (combination) is useful

Solver {'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga'}




Logistic Regression ($L_1$, Solver) ⇓ Cross validation [K fold CV]




Acc1 } Avg Acc

Disadvantage

1) Time complexity increase with huge dataset is for Training the model

## ② Randomized search CV

n_iter = 10 + CV = 5

10 different Combination + CV=5 ⇒ 50

⇓

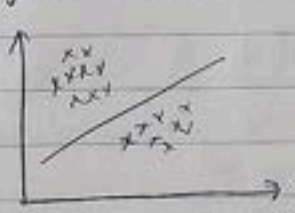
Select the best parameter

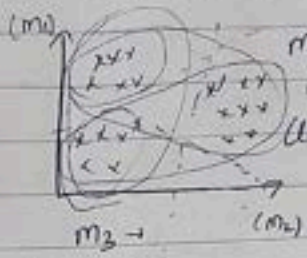
Advantage: The time complexity is decreased

How to remove outlier IQR = 1.5 * of

# Logistic Regression For Multiclass classification

Also combine each 2 each 2






① OVR → One versus Rest (V/s)

② Multinomial

| $t_1$ | $t_2$ | $t_3$ | O/P |
|---|---|---|---|
| – | – | – | 1 |
| – | – | – | 0 |
| – | – | – | 2 |
| – | – | – | 2 |
| – | – | – | 0 |
| – | – | – | 1 |

## One versus Rest

One hot encoding

| $t_1$ | $t_2$ | $t_3$ | O/P | $m_1$ $O_1$ | $m_2$ $O_2$ | $O_3$ |
|---|---|---|---|---|---|---|
| – | – | – | 1 | 0 | 1 | 0 |
| – | – | – | 0 | 1 | 0 | 0 |
| – | – | – | 2 | 0 | 0 | 1 |
| – | – | – | 2 | 0 | 0 | 1 |
| – | – | – | 0 | 1 | 0 | 0 |
| – | – | – | 1 | 0 | 1 | 0 |

$m_1$ model → I/P = $t_1$ $t_2$ $t_3$ O/P = $O_1$

// Binary classification

$m_2$ model → I/P = $t_1, t_2, t_3$ O/P = $O_2$

$m_3$ model → I/P = $t_1, t_2, t_3$ O/P = $O_3$

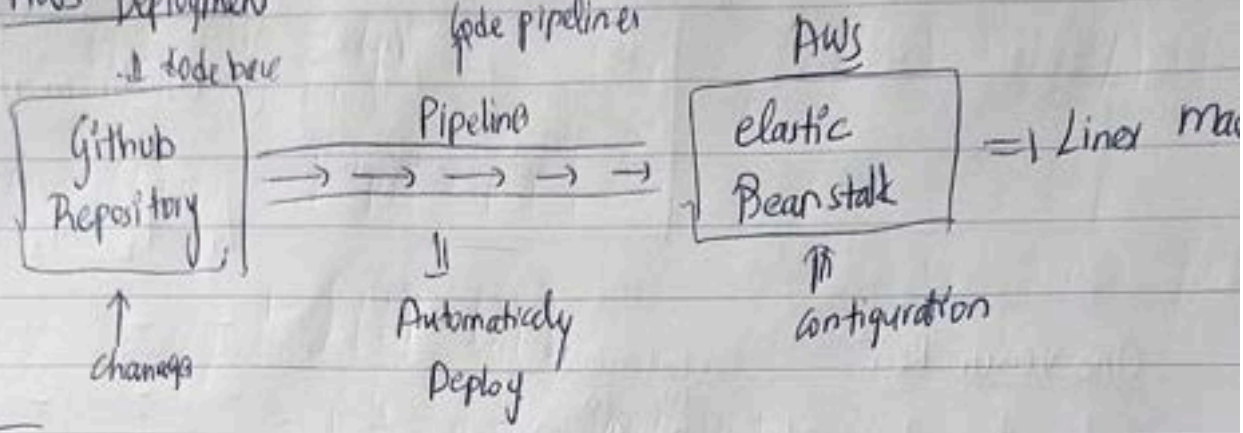
New test Data → $[m_1, m_2, m_3]$

$[0.25, 0.25, 0.5]$

⇓

New DATA ⟶ $O_2$

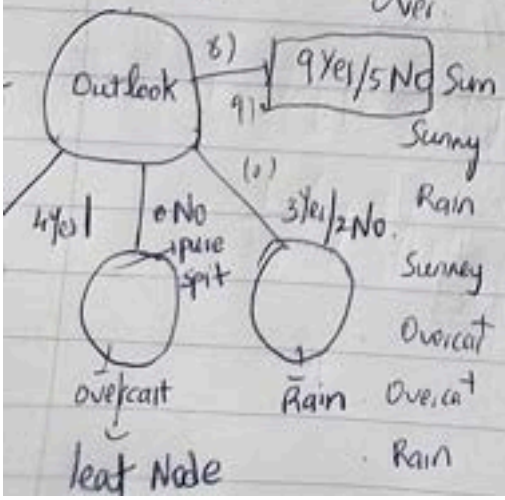
## AWS Deployment




## Decision Tree

### Dataset

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| 1) | Sunny | Hot | High | Weak | No |
| 2) | Sunny | Hot | High | Strong | No |
| 3) | Overcast | hot | High | Weak | Yes |
| 4 | Rain | Mild | High | Weak | Yes |
| 5) | Rain | | | | Yes |
| 6) | Rain | | | | No |
| 7) | Over | | | | Yes |
| 8) | Sun | | | | No |
| 9) | Sunny | | | | Yes |
| | Rain | | | | Yes |
| | Sunny | | | | Yes |
| | Overcast | | | | Yes |
| | Over.cast | | | | Yes |
| | Rain | | | | No |



① Purity Spit check :- Pure split (or) Impure split

- Entropy
- Gini Impurity

} measure of purity

② What feature you need to select to start the split → Information Gain

## ① Purity Check

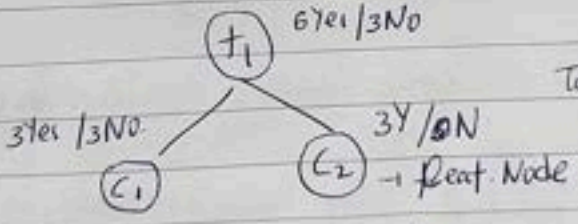
Binary Classification

### ① Entropy

$$H(S) = -P_+ \log_2 P_+ - P_- \log_2 P_-$$

$P_+$ = probability of positive category

$P_-$ = probability of negative category




$$P_+ = \frac{3}{6} = \frac{1}{2}$$ (Total positive count / Total count)

$$P_- = \frac{3}{6} = \frac{1}{2}$$ (Total negative count / Total count)

$$H(C_1) = -P_+ \log_2 P_+ - P_- \log_2 P_-$$

$$= -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log \frac{3}{6}$$

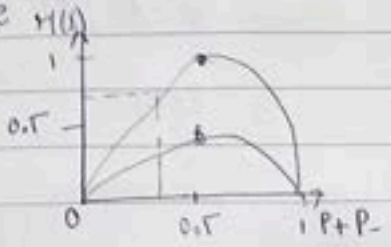
$$\boxed{H(C_1) = 1}$$ → This is impure

### ② Gini Impurity

$$G.I = 1 - \sum_{i=1}^{n} (p)^2$$

$$H(C_2) = \frac{-3}{3} \log_2 \frac{3}{3} - \frac{0}{3} \log \frac{0}{3}$$

$$\boxed{H(C_2) = 0}$$ pure split



② Gini Impurity → max = 0.5

$$G.I = 1 - \sum_{i=1}^{n} (p)^2$$

Gini $(C_1) = 1 - [(p_+)^2 + (p_-)^2]$

$= 1 - [(\frac{1}{2})^2 + (\frac{1}{2})^2]$

$= 1 - \frac{1}{2} = 0.5$ → Impure spit

3 Yes 0 No

$G.I(C_2) = 1 - [(3/3)^2 + (0/3)^2]$

$= 1 - 1$

$C_2 = 0$ → Purity spilt

Multiclass classification Problem : 3 categories In O/P

$$H(S) = -P_{C_1} \log_2 P_{C_1} - P_{C_2} \log_2 P_{C_2} - P_{C_3} \log_2 P_{C_3}$$

$$G.I = 1 - [(P_{C_1})^2 + (P_{C_2})^2 + (P_{C_3})^2]$$

② Information Gain : → Which feature to select to start the spit ?

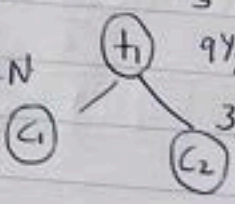
$$\text{Gain}(S, t_1) = H(S) - \sum_{v \in Val} \frac{|S_v|}{|S|} H(S_v)$$ → Entropy of child node

$H(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$

$= -\frac{9}{14} \log 9/14 - \frac{5}{14} \log (5/14)$ 6 Y 2 N

$\simeq 0.94$

| $t_1$ | $t_2$ | $t_3$ | O/P |
|---|---|---|---|
| | | | 9 Y / 5 N ← H(S) = 0.94 |
| | | | 3 Y / 3 No |



$$H(C_1) = -\frac{6}{8}\log_2\left(\frac{6}{8}\right) - \frac{2}{8}\log_2\left(\frac{2}{8}\right) \simeq 0.81$$

$$H(C_2) = -\frac{3}{6}\log_2\left(\frac{3}{6}\right) - \frac{3}{6}\log\left(\frac{3}{6}\right) = 1$$

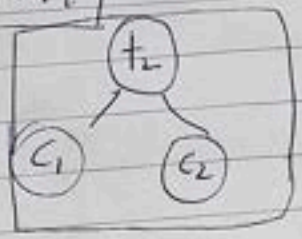
$$\text{Gain}(S, f_1) = H(S) - \sum_{v \in \text{val}} \frac{|S_v|}{|S|} H(S_v) \quad \text{Entropy of categorical}$$

Total no. of rows, Total no. rows

$$= 0.94 - \left[\frac{8}{14} \times 0.81 + \frac{6}{14} \times 1\right]$$

$\frac{8}{14}$: Total count; $0.81$: $H(C_1)$; $\frac{6}{14}$: Total Cow; $1$: $H(C_2)$

$$\boxed{\text{Gain}(S, f_1) = 0.049}$$




$\Rightarrow$ Information Gain = 0.01

$$\boxed{\text{Gain}(S, t_2) = 0.01} \quad \boxed{\text{Gain}(S, t_1) = 0.049}$$

We need to start splitting using ~~$f_2$~~ feature

~~Entropy~~

## Entropy vs Gini Impurity

When dataset is small → Entropy [log formula]

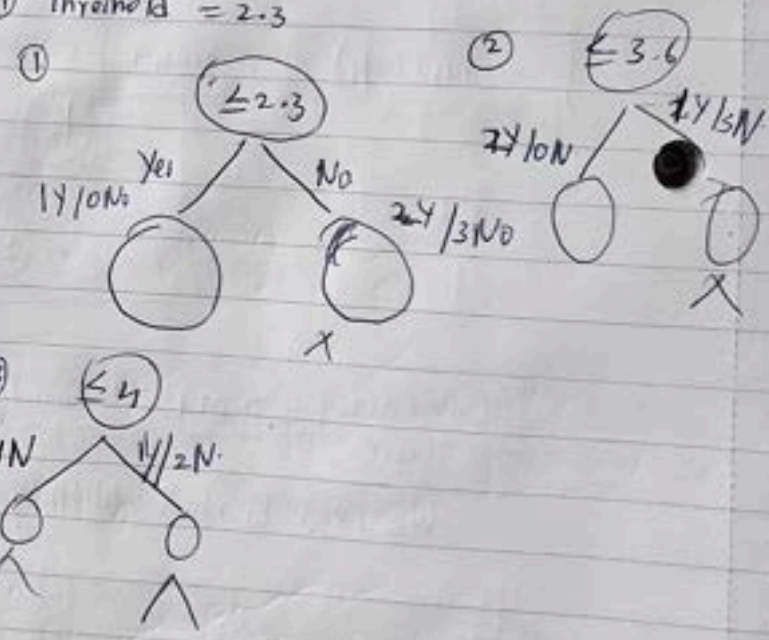
When dataset is huge → Gini Impurity [simple maths]

### What if my feature is continuous?

| $f_1$ | O/P |
|---|---|
| 2.3 | Yes |
| 3.6 | Yes |
| 4 | No |
| 5.2 | No |
| 6.7 | Yes |
| 7.8 | No |

① Sort the feature $f_1$

① Threshold = 2.3




Time Complexity is huge ~~then~~ when dataset is huge

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
```

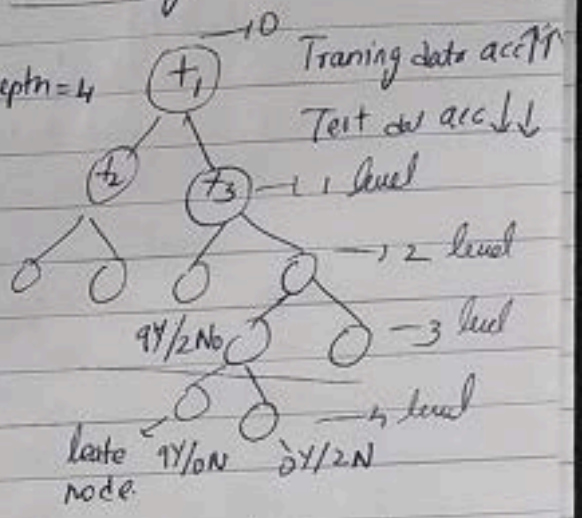
## Decision Tree post pruning and pre pruning [Reduce overfitting]

Training Data

$t_1$ $t_2$ $t_3$ O/P

Generalized model

Reduce overfitting




### ① Post Pruning

① Construct the entire decision Tree to complete leaf node

② Pruning the decision

③ ~~for~~ Suitable for smaller Dataset

### ② Pre pruning

① No Hyperparmeter Tuning to select Best parameters
(gridcv1 · Random RCv)

| Decision Tree Classifier | Decision Tree Regressor |
|---|---|
| ① Entropy | ① Variance Reduction |
| ② G.I (Gini) | ② Variance |
| ③ Information Gain | |

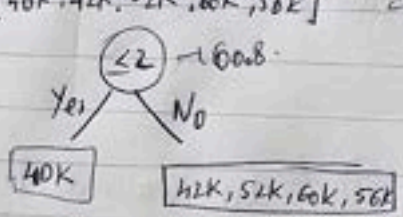
Dataset

| exp | career Gap | Salary (o/p → continuous) |
|---|---|---|
| 2 | Yes | 40k |
| 2.5 | Yes | 42k |
| 3 | No | 52k |
| 4 | No | 60k |
| 4.5 | Yes | 56k |

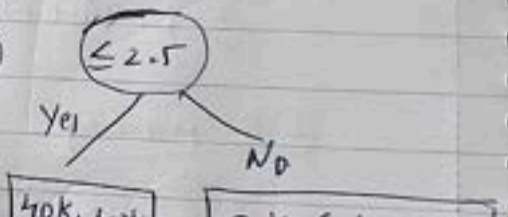
$\bar{y}$ = 50k ← avg. salary (mean)

Variance Reduction

① [40k, 42k, 52k, 60k, 56k]




②




Final aim → Variance reduction

$$\text{Variance or error} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y_i})^2 \quad \text{[mean squared error]}$$

$$\text{Variance of Root} = \frac{1}{5}\left[(40-50)^2 + (42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2\right]$$

$$= \frac{1}{5}\left[100 + 64 + 4 + 100 + 36\right]$$

$$= 60.8$$

Variance of left Ch = $\frac{1}{1}[(40-50)^2$

$= 100//$

Variance of right = $\frac{1}{4}\left((42-50)^2+(52-50)^2+(60-50)^2+(50-56)^2\right.$

$= 51//$

Variance of Reduction = Var (Root) $- \sum w_i$ Var (child)

$= 60.8 - \left[\frac{1}{5} * 100 + \frac{4}{5} * 51\right]$

$= 0$

left side having only one line

by total elements

Variance Reduction = 0

Slect the greteest Variance Reduction