Machine Learning Algorithms in short

① Linear Regression

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

Cost function → Squared Error function

$$\Rightarrow \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y)^2$$

↓ minimize it by adjusting θ_0, θ_1

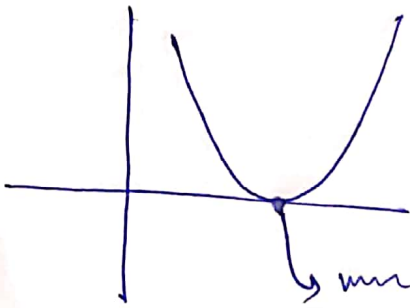
Convergence Algorithm

repeat until { converge

$$\theta_j = \theta_j - \alpha \frac{d}{d\theta_j} J(\theta_0, \theta_1)$$

}

↓ learning rate α → generally 0.01



→ gradient descent

→ minima

Best fit line

Reduced convergence algorithm

repeat until convergence

{

$$\theta_0 = \theta_0 - \alpha \frac{1}{m} (h_\theta(x) - y)$$

$$\theta_1 = \theta_1 - \alpha \frac{1}{m} (h_\theta(x) - y) x$$

}

Performance Metrics

$$R^2 = 1 - \frac{SS\ Residual}{SS\ Total} = 1 - \frac{\sum (y_{(i)} - h_\theta(x)_i)^2}{\sum (y_i - \bar{y})^2}$$

→ The problem with R^2 is that it works in such a way that if features are added even if they are no way correlated then R^2 increases

$$\text{Adjusted } R^2 = 1 - \frac{(1-R^2)(N-1)}{N-P-1}$$

N → no. of data points

P → no. of features

Assumptions of Linear Regression

1. If features are in Normal/Gaussian Distribution, model will get treated well
2. Standardization & scaling data} with 2 scales

   $\mu = 0$, $\sigma = 1$

3. Linearity

Note:-

Overfitting:- When model performs well with training data but fails to perform well with testing data

→ Low Bias and High variance

train ✓ test ✓

→ Model is too complex and size of data set size.

Underfitting :- Model performs bad with training and testing data set.

→ High Bias and low variance

Ridge and Lasso

Ridge (L2 Regularisation)

$(\hat{y}_i - y)^2 + \lambda (\text{slope})^2$ → Hyperparameter

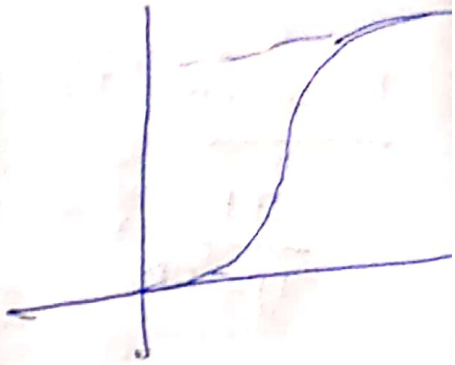
→ To prevent overfitting

Lasso (L1 Regularisation)

$(\hat{y} - y)^2 + \lambda |\text{slope}|$ → helps in feature selection

Logistic Regression

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



Cost function

$$J(\theta_i) = \begin{cases} -\log(h_\theta(x)) & y = 1 \\ -\log(1 - h_\theta(x)) & y = 0 \end{cases}$$

overall → $\frac{1}{2m} \left[-y \log(h_\theta(x) - (1-y) \log(1 - h_\theta(x)) \right]$

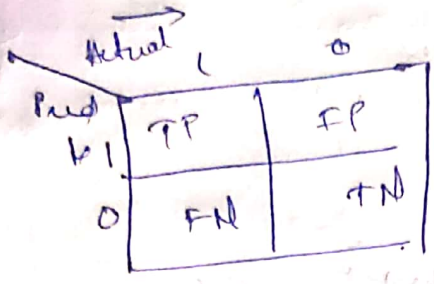
repeat until convergence

{

$$\theta_j = \theta_j - \alpha \frac{d}{d\theta_j} J(\theta_i)$$

}

Performance metrics for classification problem



$$\text{Accuracy} = \left(\frac{TP+TN}{TP+TN+FP+FN}\right)$$

Precision (P)

$$\frac{TP}{TP+FP}$$

Recall (or sensitivity)

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

F-score

$$\Rightarrow (1+\beta^2)\cdot\frac{P \times R}{\beta^2 * P + R}$$

$\beta = 2$ FN & FP

$\beta = 0.5$ FP & FR

$\beta = 1$ FN = FP.

Naive Bayes Algorithm

$P(A \text{ and } B) = P(A) \times P(B/A)$

$P(A \text{ and } B) = P(B \text{ and } A)$

$\boxed{P(A) \times P(B/A) = P(B) \times P(A/B)}$ → Bayes theorem

$$P(B/A) = \frac{P(B) \times P(A/B)}{P(A)}$$

Let B be output and A be input

$$P(y/x_1, x_2 \ldots x_n) = \frac{P(y) \times P(x_1, x_2, x_3 \ldots x_n / y)}{P(x_1, x_2 \ldots x_n)}$$

$$P(y/x_1, x_2 \ldots x_n) \Rightarrow \frac{P(y) \times P(x_1/y_1) \times \ldots P(x_n/y_n)}{P(x_1) \times P(x_2) \times P(x_3) \times P(x_4) \ldots P(x_n)}$$

Note :- Suppose we do binary classification then any probability > 0.5 will be 1 and < 0.5 will be equal to 0.

→ Normalization

$$P(\text{yes}/x_i) \text{ normal} = \frac{P(\text{yes}/x_i)}{P(\text{yes}/x_i) + P(\text{no}/x_i)}$$

Q) Dataset

Day	Outlook	Temp	Humidity	Wind	Playtime
1	Sunny	Hot	High	windy	No
2	Sunny	Hot	High	slow	No
3	Overcast	Hot	High	windy	yes
4	Rain	mild	High	~~slow~~ windy	yes
5	Rain	cool	Normal	windy	yes
6	Rain	cool	normal	slow	no
7	Overcast	cool	normal	slow	yes
8	Sunny	mild	High	windy	no
9	Sunny	cool	Normal	windy	yes
10	Rain	mild	Normal	windy	yes
11	Sunny	mild	Normal	slow	yes
12	Overcast	mild	High	slow	yes
13	Overcast	Hot	normal	windy	yes
14	Rain	mild	High	slow	no

Find $P(yes \mid sunny, hot)$

$$\Rightarrow \frac{P(yes) \times P(sunny/yes) \times P(hot/yes)}{P(sunny) \times P(hot)}$$

Outlook (x_i)

	Y	N	$P(\frac{x_i}{yes})$	$P(\frac{x_i}{no})$
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rain	3	2	3/9	2/5
	9	5		

Temp

	Y	N	$P(\frac{x_i}{yes})$	$P(\frac{x_i}{no})$
Hot	2	2	2/9	2/5
mild	4	2	4/9	2/5
cold	3	1	3/9	1/5
	9	5		

$\Rightarrow \frac{9}{14} \times \frac{2}{9} \times \frac{2}{9} \Rightarrow \frac{2}{63}$

$P(no/sunny, hot) = \frac{5}{14} \times \frac{3}{5} \times \frac{2}{5}$

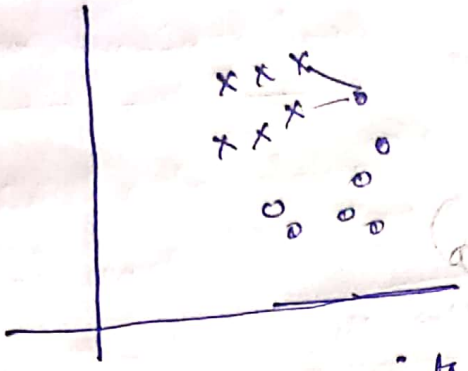
$\Rightarrow \frac{3}{35}$

$P(yes/x_1, x_2) \Rightarrow \frac{\frac{2}{63}}{\frac{3}{35} + \frac{2}{63}}$ } normalisation.

K nearest Neighbours

Classification

Regression

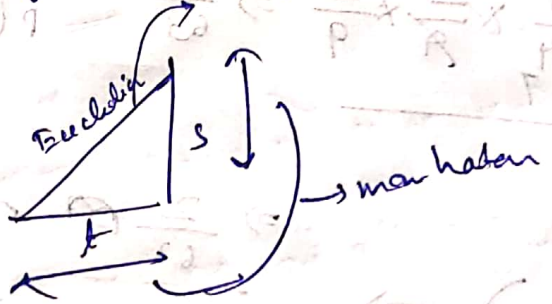


→ Find nearest K points

→ If the no. of points > $\frac{K}{2}$ in any of the subsets (types)
then it belongs to that group

Euclidean distance = $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

Manhattan distance
⇒ t+s



→ For regression the predicted value will be avg. of K nearest neighbours

Problems faced with K nearest neighbours

① Outliers
② Imbalanced dataset.

Incomplete

Decision Tree

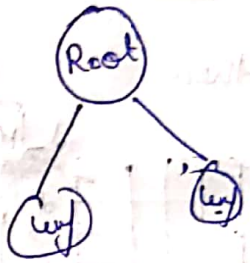
- Regression
- Classification → Nested If else

Classification of a particular leaf node should be done until pure split

→ pure split is when there is purely one class.

→ D.Tree is made by 2 diffnt parameters

- Entropy
- Ginni Index



Entropy

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

Ginni

$$GI = 1 - \sum_{i=1}^{n} (P)^2$$

Features are selected using g. Information gain.

$$Gain(S, i) = H(s) - \sum_{v \in Val} \frac{|S_v|}{|S|} H(S_v)$$

- $H(s)$ → Entropy of root node
- $|S_v|$ → no. of samples in leaf nodes
- $H(S_v)$ → Entropy of leaf node
- $|S|$ → no. of samples in root node

Problem with D-Tree is overfitting

Hyperparameter

- Post pruning (cutting after using the tree.
- Pruning

Note:- More the Info gain of a feature, splitting takes place first

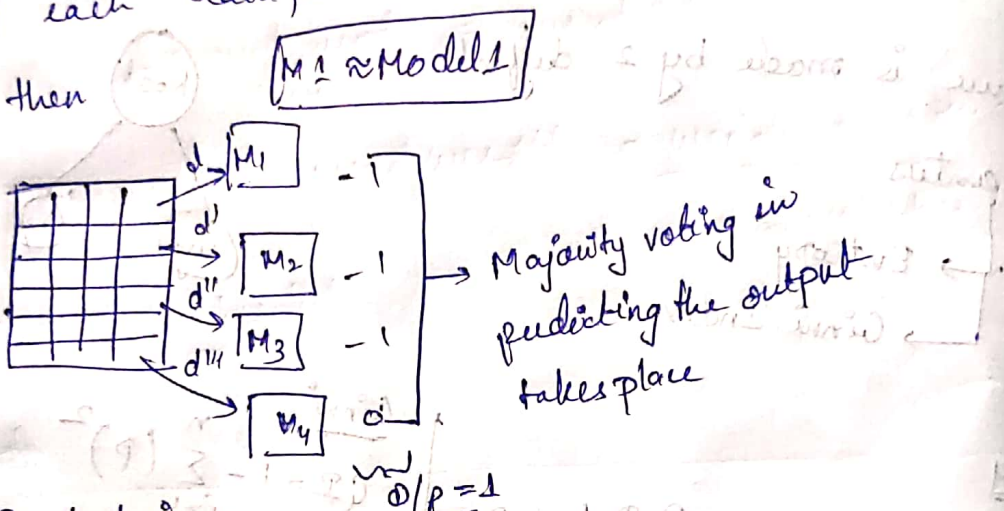
Ensemble Learning

- Bagging
- Boosting

Custom Bagging technique

→ Dividing the datapoints to various sets

→ Training each datapoint with various ML algorithms and then



Bagging techniques

① Random Forest classifier

② Random Forest Regression [Mean of M1, M2 M3, M4]

In Random Forest M1, M2, M3, M4 are all D-Trees

→ No need of normalisation

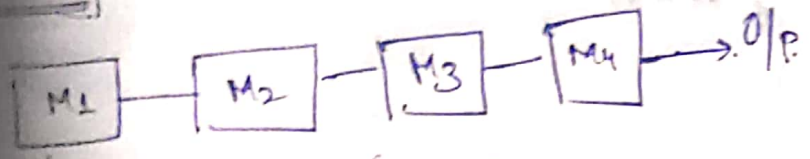
→ Not affected by outliers.

Outliers

→ An outlier is an observation that lies in a abnormal distance from other values in a random sample from population.

Standardisation

Boosting



weak learner

→ Multiple ML model in sequence are used to predict the output

→ O/p is got by voting again

→ Where data points and features are selected.

Adaboost

	d1	d2	d3	d4	O/P	Weight
~~0~~						
1						1/7
2						1/7
3						1/7
4						1/7
5						1/7
6						1/7
7						1/7

→ Assigning equal weights for all words

→

Stump

1 level D-Tree is called a stump

→ The error is then found out

Let the no. of error be n

If no of
weight of 1 word is w and no. of errors are
→ n then TE = w × n
(Total error)

→ Performance of stump = $\frac{1}{2} \log_e \left(\frac{1-TE}{TE} \right)$
(Ps)

→ new sample weight = weight × e^{-Ps}
for correct data

Incorrect word weight = weight × e^{Ps}

→ normalised word weight) = $\frac{\text{weight of it own}}{\text{Total (which is 1)}}$

Then create bucket

Example