Machine learning:

A computer program P is said to learn with experience E on class of tasks T. if the performance on tasks T, measured by P improves with experience 'E'.

- Recognizing spoken words
- Driving autonomous cars
- Identifying newly found astronomical structures
- Learning to play word-class backgammon

Designing a learning system:-

1) Choosing training experience:-
   - Type of feedback ← direct / indirect on the each move on the performance
   - Degree of control on sequence of training examples
   - Choosing a representation for distributing training examples

2) Choosing target function:-
   - $T: B \rightarrow M$ (chooseMove)
   - $V: B \rightarrow R$ (Real no.)
   - Learning the target function is called function approximation.

$V(b) = 100$ win $V(b) = -100$ loose $V(b) = 0$ draw. b = final state

- $V(b) = \hat{V}(b)$

3) Choosing a representation for target function:

$$\hat{V}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6$$

4) Choosing a function approximation algorithm

$\langle b, V_{train}(b) \rangle$ $b \in B$

ordered pair of each move & score associated with it is given.

5) Estimating training values:-
Assign the training value of $V_{train}(b)$ for any intermediate board state b to be $\hat{V}(Successor(b))$, $\hat{V}$ is current approximation & Successor is next board state.

$$V_{train}(b) \leftarrow \hat{V}(Successor(b))$$

6) Adjusting the weights:-

We need define best hypothesis, or set of weights which minimizes the squared error E between the training values & the values predicted by hypothesis $\hat{V}$

$$E \equiv \sum (V_{train}(b) - \hat{V}(b))^2$$

LMS weight update rule:

$$w_i \leftarrow w_i + \eta (V_{train}(b) - \hat{V}(b)) x_i$$

Issues in ML:-
→ How much data is sufficient
→ Which algorithm perform best
→ When and how can prior knowledge be helpful
→ Strategy for choosing training experience
→ How can learner automatically alter its representation

Chapter 2:-
Concept Learning:-
A task acquiring a potential hypothesis that best fits the training examples.

Inductive learning hypothesis:- Any hypothesis found to approximate the target function over a large set of training examples will also approximate target function over other unobserved examples.

attributes/instances & target function
Instance space = The target hypothesis that we want to learn is represented as a conjunction of all attributes
possible distinct instances = multiplication of no. of possible values for each attributes.

Hypothesis space:
Set of all possible hypotheses

If sky has 3 possible values and other attributes have 2 values
instances space = $3 \times 2 \times 2 \times 2 \times 2 \times 2$.

Hypothesis space = $(3+2) * (2+2) * (2+2) * (2+2) * (2+2) * (2+2)$
+2 because one for ϕ & one for ?

Concept learning ~~space~~ search space: - Task of searching through a large space of hypotheses to find the hypothesis that best fits the training example.

General hypothesis = $h = \langle ?, ?, ?, ?, ?, ? \rangle$
Specific hypothesis = $h = \langle \phi, \phi, \phi, \phi, \phi, \phi \rangle$

Given hypothesen h_j and h_k, h_j is said to be more general or equal to h_k, if any ~~on~~ instance that satisfies h_k will also satisfies h_j.

$$h_j \geq h_k \equiv \forall x \, (h_k(x) = 1 \rightarrow h_j(x) = 1)$$

Find-S algorithm:- maximally
Finding the ~~most~~ specific hypothesis.
1) Initialize h to ~~most~~ specific hypothesis $\langle \phi, \phi, \phi, \phi, \phi, \phi \rangle$
2) For every positive training example x,
For every attribute contraint a_i in h
- if the contraint is satisfied by h, then do nothing
- else replace a_i in h by the next more general contri
-nt that satisfies x.
3) Output h

* Ignores the negative example
* Hypothesis is consistent with positive training examples

Issues in Find-S algorithm

→ Has the learner converged to the target concept? - Is that is only consistent hypothesis present?

→ Why most specific hypothesis?

→ Are training examples consistent? What if noise/errors are present and its ignoring the negative examples as well.

→ What if there are several maximally specific consistent hypothesis? Learner should be able to backtrack to get the more generalized one.

Consistent hypothesis:-

Hypothesis h is consistent over training example D. with target function c, iff $h(x) = c(x)$ for each $\langle x, c(x) \rangle$ in D

$$Consistent(h, D) \equiv \forall x (\langle x, c(x) \rangle \in D), h(x) = c(x)$$

Version space :- Set of all consistent hypothesis

$$VS = \{ h \in H \mid Consistent(h, D) \}$$

List-then-eliminate algorithm:-

1) List all possible hypothesis in H in version space
2) foreach hypothesis h in H
   remove any from version space if $h(x) \neq c(x)$
3) output versionspace

* Works if finite hypotheses space is present

Candidate-Elimination algorithm :-

→ Works on List-then-eliminate principle
→ More compact representation of version space
→ Gives more specific & general hypotheris
→ Gives the boundaries, that delimits the version space

Algorithm:

Initialize more general hypothesis G
Initialize more specific hypothesis S
for each training example d, do
if it is a +ve example:
    For each s in S
    - Remove s if its not consistent with d
    - Add ~~me~~ next to general constraint in S which satisfies
    - Remove g from G if any are more specific than s in d.
if it is a -ve example:
    for each g in G
    - Remove g if not consistent
    - Add next to specific constraint in g which satisfies d.
    - Remove g if any hypotheris is less general than another hypothesis in G

Remarks:→ There should not be errors in training examples
→ There should be a hypothesis that correctly describes target function

Biased hypothesis:- Only conjunction of all attributes.

~~L(xi, Dc) → L(xi. (Dc ∧ xi) → L(xi, Dc)~~

Inductive bias:- Inductive bias of L is the set of assumptions B such that for new instances x_i
$(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)$

Chapter 3:-

Decision trees:- ~~Disjo~~ Disjunction of conjunction of constraints of attribute values

When to use decision trees:-
→ Attributes have multiple values
→ Distinct output/class value
→ Disjunctive principle needed

→ If training data consists missing values or errors.

ID3 - Iterative Dichotomiser.
Repeatedly divides the attributes into 2 or more subgroups.
→ Top-down approach
→ Never back-tracks

→ Maximum information gain or minimum entropy
→ Entropy is the degree of uncertainty corresponding with a random attribute.

$$Entropy(S) \equiv -(P_{\oplus} \log_2 P_{\oplus} + P_{\ominus} \log_2 P_{\ominus})$$

→ Information Gain is the expected reduction in entropy caused by partitioning the examples according to the attributes.

$$Gain(S, A) \equiv Entropy(S) - \sum \frac{|S_v|}{|S|} Entropy(S_v)$$

Advantages of decision tree :-
→ Searches complete hypothesis space.
→ Uses all training examples
→ Less sensitive to errors

Disadvantages :-
→ Only one current hypothesis. No capability of representing all consistent hypothesis.
→ No backtracking

Training set :- * Data used to train the model.
* For every epoch, model learn over same data to learn the features.

Validation set :- * Validating the model
* How well the model is learning
* Give information on how to adjust hyperparameters (learning rate)

* How to generalize the training examples
* Ensure the model is not overfit

Test set:
* Data set is a set of data used to test the trained model
* Unlabelled data is given

Cross-validation:-
→ Statistical model to check ability of training model
→ Select & compare the model.
→ Divide the dataset into k-equal size groups
→ Shuffle the dataset & then divide.
→ Take 1 group for testing & k-1 for training
→ k-1 used to fit the model
→ 1 group to evaluate the model
→ Get the evaluation score
→ Discard the model
→ Repeat this for k times

Types
- Representative
- k=10
- k=n

high variance & high bias (over-estimation)
(change of data)

- Representative — large amount of dataset
- k=10 — safer side. Low bias + modest variance
- k=n — Leave out one cross validation (LOOCV)
  - each training example considered for train & testing

K-fold API:-
→ sci-kit library
→ KFold (split no., whether to split or not, seed (no. used prior to shuffle))

Issues in decision tree:

1) How to avoid How deeply to grow the decision tree
→ Over fitting
→ Training – fitting the model over a training data set to get reliable prediction on unseen data
→ If there is a hypothesis h_1, $h_1 \in H$ said to overfit such that h_1 has less error than h_2 on training examples but h_2 has less error than h_1 over complete distribution of instances
→ Reason :• Error in training example
• Small number of examples associated with leaf node
→ More complex → ~~Bad~~ Tree depth more
→ Train performance ↑ but ↓ testing performance.

2) Handling continuous value attributes
→ ~~changing the continue value to discrete~~
→ Adding a new discrete valued attribute that partitions the continuous values into discrete value intervals

3) Choosing an appropriate attribute measure
→ Alternate way of choosing an attribute
→ If we take 'day' as root it classifies correctly but it has broader distribution.
→ It is a poor classifier because, its actually not a useful attribute
→ Gain ratio. – uses split information to know how broadly & uniformly splits the data

→ $\text{split}(S, A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$

→ $\text{Gain Ratio}(S, A) = \frac{\text{Gain}(S, A)}{\text{split}(S, A)}$

4) Handling missing values
→ most common data

5) Handling attributes with different cost.
→ $\frac{Gain(S,A)}{cost(A)}$
→ If possible attributes with low cost taken
→ for reliable classification - high cost attributes.

Gini-index or impurity:
→ Degree/probability of a variable being wrongly classifying when randomly chosen.
→ & lower gini-index preferred
→ ~~0 &~~ Between 0 and 1
→ 0 : only one class or all elements in one class
→ 1 : elements randomly distributed over ~~all~~ different classes
→ 0.5 : elements ~~randomly~~ equally distributed over different classes.
→ $1 - \sum_{i=1}^{c} (P_i)^2$

Avoid over-fitting:-

How to know the correct length of tree
1) Direct approach - 2/3 training & 1/3 validation
2) Create a decision tree & then check if extending node helps in improving the accuracy
3) Minimum Length Description principle - encoding the training example & hypothesis - Halting the growth when encoding size minimized

Pruning:-
1) Reduced error pruning
→ Pruning of decision node
→ Remove sub-tree
→ Consider it as a leaf node & add most commonly classified

→ Do it only the resultant works fine
→ this will remove of any leaf nodes added because of irregularities

Rule post-pruning:-

→ Over-fit the model
→ Convert the tree as rules
→ Generalize the rules, removing preconditions
→ Arrange it based on accuracy

Internals - 2 :-

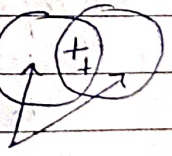
Computational learning theory:-

Sample complexity – Number of training examples
Computational complexity – Computational resources
Mistake bound – Incremental learning
– No. of misclassified training example

Learning problems → models/algorithm → Sample
→ Computational
→ mistake bound

Let us take the true error for a hypothesis (h) with target concept C and instance distribution D it is the → probability that h will misclassify an instance drawn at random according to D

$$error_D(h) \equiv \Pr_{x \in D}(h(x) \neq c(x))$$



Approximating True error for hypothesis

Selecting h with true error 0:
→ Select h with ~~true~~ training error 0
→ But there may be many h, we don't know which to select

Let true $error_D(h) < \varepsilon$, (small)
Not all training instances succeed, only with $1-\delta$ (small) probability will

ε-exhausted
The version space $VS_{H,D}$ is said to be exhausted wrt target concept C & D (instance distribution) if for all hypothesis (h) in VS_{HD} has true error less than ε.
$(\forall h \in VS_{HD})\ error_D(h) < \varepsilon$
How many training examples will ε-exhaust the VS?
Haussler's theorem
For a finite hypotheses space H, D is sequence $m \geq 1$ random independent examples for target concept C, then for any $0 \leq \varepsilon \leq 1$, the probability of VS wrt H and D for C is not ε-exhausted is less than $|H|e^{-\varepsilon m}$

Proof:-
→ Let $h_1, h_2, h_3, \ldots h_k$ be hypotheses space H, which have true error $> \varepsilon$ for D and C
$H_{Bad} = \{h_1, h_2, h_3, h_4, \ldots h_4\}$
→ We will fail to exhaust the VS if one of these hypothesis happens to be consistent with all training examples m
→ The probability of a hypothesis having $error_D(h) > \varepsilon$ being consistent with one of the training example is at most $(1-\varepsilon)$
→ This hypothesis getting consistent with all m training examples at most $(1-\varepsilon)^m$
→ If we have k hypothesis
$k(1-\varepsilon)^m$
→ But $k \leq |H| \Rightarrow |H|(1-\varepsilon)^m$
→ General inequality eg $a \leq \varepsilon \leq 1 \Rightarrow (1-\varepsilon) \leq e^{-\varepsilon}$
$\Rightarrow k(1-\varepsilon)^m \leq |H|(1-\varepsilon)^m \leq |H|e^{-\varepsilon m}$
→ Let us derive training examples no. wrt error rate $< \delta$
$|H|e^{-\varepsilon m} \leq \delta$
$m \geq 1/\varepsilon (\ln(|H|) + \ln(1/\delta))$

→ Number of training examples 'm' is sufficient to tell the model is probably approximately correct hypothesis

PAC Learning:-

Let us consider a concept class C over set instances X by length * n, learner L uses hypotheses space H. The C is PAC learnable by L if ∀c ∈ C, Distribution D over X, such that $0 < \varepsilon < 1/2$, $0 < \delta < 1/2$, then the L will with prob atleast $(1-\delta)$ output a hypothesis $h \in H$ $error_{(D)}(h) \leq \varepsilon$, with time polynomial to $1/\varepsilon$, $1/\delta$, n, size(c)

Agnostic learning:- The learner will not make any assumption that the target concept c is representable by H, it will simply just produce hypothesis with minimum training error ~~$c \notin H$~~ don't assume $c \in H$

$Pr[error_D(h) <> error_D(h) + \varepsilon] \leq 2e^{-2\varepsilon^2 m}$

$Pr[error_D(h) > error_D(h) + \varepsilon] \leq e^{-2m\varepsilon^2}$

$Pr[(\exists h \in H)[error_D(h) > error_D(h) + \varepsilon]] \leq |H| e^{-2m\varepsilon^2}$

$$m \geq \frac{1}{2\varepsilon^2}\left(\ln(|H|) + \ln(1/\delta)\right)$$

Shattering a set of instances :-

~~Let the sub~~ The instace set S is shattered by H. if $h \in H$ consistent with dichotomy.

$2^{|S|}$ dichotomies through H

Vapnik-Chervonenkis dimension :-

VC(H) is defined over H and X is the size of the largest finite subset of X shattered by H

$$m \geq \frac{1}{\varepsilon}\left(4\log_2(2/\delta) + 8\,VC(H)\log_2(13/\varepsilon)\right)$$

Conditional probability - Probability of an event happening given that it has relationship with one or more other events

Bayes theorem - Calculates the test r. Takes the test result probability and calculates the real probability that the
Bayes theorem - Takes the test result and gives the real probability that the test identified the event.

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

$$p(h|D) = \frac{P(D|h) * P(h)}{P(D)}$$

→ prior prob of h (prior knowledge of chance that h is correct)

→ The prior prob of D, prob that D is observed

→ The prob of D observed given that h is correct

→ Posterior prob h holds after D has been observed

MAP hypothesis

Maximally probable hypothesis - Maximum Posteriori hypothesis

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}}\ P(h|D)$$

$$= \underset{h \in H}{\operatorname{argmax}}\ \frac{P(D|h) * P(h)}{P(D)}$$

$$= \underset{h \in H}{\operatorname{argmax}}\ P(D|h) * P(h)$$

$P(D|h)$ → likelihood of D given h

Any hypothesis that maximizes $P(D|h)$ - Maximum likelihood

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}}\ P(D|h)$$

Brute - force

→ Let the learner considers a finite hypotheses space H over training instances X, where the task to learn target concept $C: X \to \{0,1\}$

$(x_1, d_1)\ (x_2, d_2) \ldots (x_m, d_m)$

x_i → training instance

d_i → target value $\quad d_i = c(x_i)$

$\forall h \in H \quad P(h_i|D) = \frac{P(D|h_i) * P(h_i)}{P(D)}$

Then HMAP

Assumptions

1) D is noise free $d_i = c(x)$

2) C is contained in H. $[(\exists h \in H)(\forall x \in X)(c(x) = h(x))]$

3) No need to consider any hypothesis has different prior probable

$$p(h) = \frac{1}{|H|} \quad \forall h \in H$$

$$P(D|h) = \begin{cases} 0, & \text{if } h(x_i) \neq c(x_i) \quad \forall x_i \in X \\ 1, & \text{if } h(x_i) = c(x_i) \quad \forall x_i \in X \end{cases}$$

$(d_1, d_2, d_3, \ldots d_m)$ sequence of $(x_1, x_2, x_3, \ldots x_m)$ & h holds

$$P(h|D) = \frac{P(D|h)\ P(h)}{P(D)} \longrightarrow \text{Inconsistent hypotheses}$$

$$= \frac{0 * P(h)}{P(D)} = 0.$$

$$P(h|D) = \frac{P(D|h)\ P(h)}{P(D)} \longrightarrow \text{Consistent hypothesis}$$

$$= \frac{1 * 1/|H|}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|}$$

Bayes optimal classifier

h_1, h_2, h_3 posterior prob = 0.4, 0.3, 0.3

h_1 is MAP if new instance x is classified as '-ve by h_1 it is -ve.

Let us take h_1 says '-ve, h_2 & h_3 says '+ve'.

Here most probable classification is different than HMAP.

Most probable classification is given as by combining prediction of all hypotheses weighted posterior hypothesis

$$p(v_j|D) = \sum_{h_i \in H} p(v_j|h_i)\ p(h_i|D)$$

$$\underset{v_j \in V}{\text{argmax}} \sum_{h_i \in H} P(v_j|h_i)\ P(h_i|D)$$

Naive Bayes classifier:-

→ X is conjunction of attributes

→ $v_j \in V$

$$V_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j / a_1, a_2, a_3, \cdots a_n)$$

$$= \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2, a_3 \cdots a_n / v_j) P(v_j)}{P(a_1, a_2, \cdots a_n)}$$

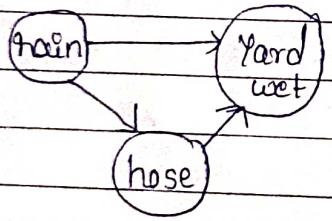
$$= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2, \cdots a_n / v_j) P(v_j)$$

$$V_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i / v_j)$$

Naive Bayes network

→ Compact, flexible, interpretable representation of joint prob distribution

→ Acyclic directed graph - relation blw variables



$P(Y, H, R) = P(Y|H,R) P(H|R) P(R)$

Instance based learning:-

KNN - Select k value

- Select k nearest neighbors for new datapoint
- Count no. of datapoints in each category
- Assign it to the category where got most no. neighbors
- Lazy - Non parametric (Functional form)

Case Based Reasoning:-

→ Intelligent system

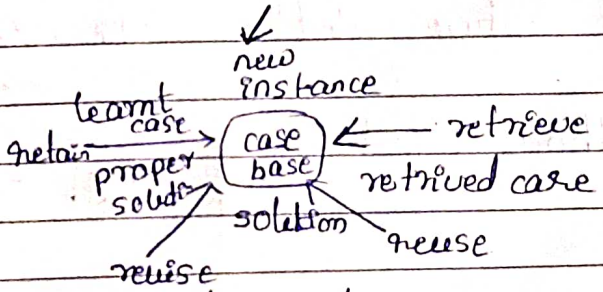
→ experiences stored as cases

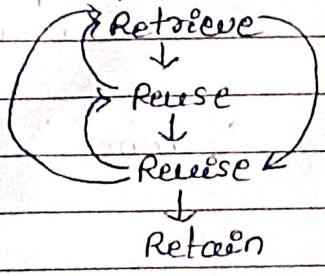
→ Recall similar experience from memory

→ Reuse it for new instance

→ Save this new case

→ Similar problem has similar solution Reuse & Retain
problem – context
solution – pointer to other relevant cases
– solution quality assessment
– steps to solution
Ex:- Technical diagnosis of a car fault
R4 based
→ Retrieve – Check for similar case
– Similarity index, explanation method, case based.
→ Reuse – Transform / adapt
– interactive, description, rule based, model, constraint
→ Repair – Simulation, real world
Revise – Verify, improve
→ Retain – Add it to case base
– similarity, case-base, solution adaption





Applications :- Help-desk, customer support, disease diagnosis product recommendation, Text classification

Confusion matrix :-
→ NXN matrix used to evaluate the performance of the classification model, where 'N' is the no. of target class-es
→ Compares actual & predicted target values

		Actual values T	Actual values F
Predicted values	T	TP	FP → Type-1
	F	FN → Type-2	TN

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

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

Specificity (TNR) True negative rate.

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

$$\text{F1-score} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{Recall}}}$$

harmonic mean

Accuracy - balanced data (TP & TN)
F1-score - Imbalanced data (FP & FN)
Precision - FP
Recall - FN

Feature selection:-
→ Selecting few features from set of features pool.
→ Avoid data redundancy/overfitting/irrelevant features
→ Data reduction, less storage, less complexity, less time, reduced dimension, increased accuracy

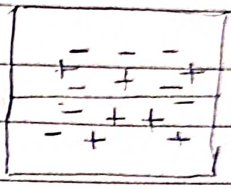
→ Filter based method: chi-square, correlation, information gain
→ Wrapper method: features → subset → Train → Performance
  → Expensive
  → Forward feature select^n → Backward feature elimination → Recursive feature elimination
  selecting best subset
→ Embedded method: features → subset ↔ model + performance

Scikit-learn:-
→ Dropping feature with 0 variance or low variance.
  variance Threshold (◉)
  if we want 80% of 0's
  $0.8 * (1-0.8) = 0.16$
  VarianceThreshold (threshold = 0.16)
→ Univariate feature selection - Selecting features based on statistical ranking. chi-square test, correlation
  Feature scoring, ranking, selection
→ model based selection - ML models to select
→ Pipeline - Data preprocessing → mutual information → cross validation
  → hyper parameter tuning

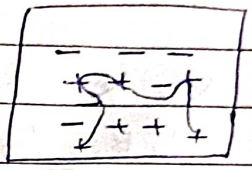
Approximating the target function
→ $\forall \in \; Y = f(X)$
→ Generalization means how well the concept learned by model can apply on new unseen data
→ Generalizing new data
→ $MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y}_p)^2$

Types of error - ① Noise error

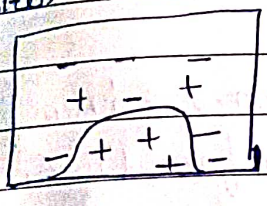
② Bias - Make assumptions
simplistic model with low variance
Underfitting

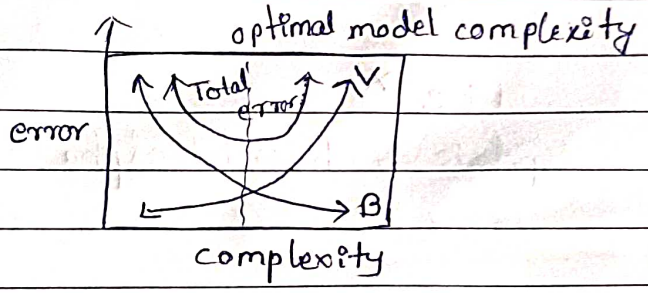


③ Variance - Fluctuations in data
- Learns too much
- Overfitting



Bias-variance tradeoff





Overfitting - Learns too much on the fluctuations, pattern, noise that hurts the model performance
- Interprets noise as patterns
- Good training performance but poor test data generalization
- Complex features

Underfitting - Neither learns nor generalize
- Performance not good
- No complexity

k-fold cross validation & hyperparameter tuning
Reducing no. of features
Regularization - $J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^i)^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$

Logistic regression:
→ Binary classification

Linear regression → $y = b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n$

$$y = b_0 + b_1 * x \qquad P = \frac{1}{1+e^{-y}}$$

$$\ln\left(\frac{P}{1-P}\right) = b_0 + b_1 x$$

$$\text{transformed} = \frac{1}{1+e^{-x}}$$

- $X = b_0 + b_1x_1 + b_2x_2$

Let $b_0 = b_1 = b_2 = 0$

$$\frac{1}{1+e^{-x}}$$

Update ⇒ $b = b + \text{alpha} * (X - \text{transformed}) * \text{transformed} * (1 - \text{transformed}) * x_i$

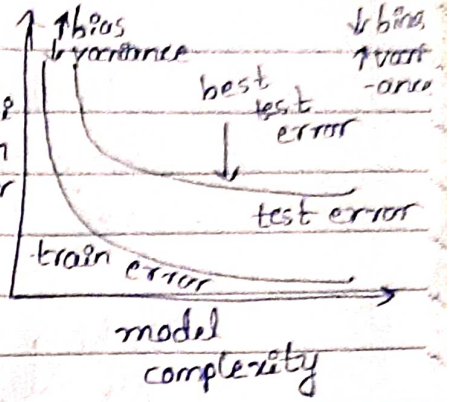
Ensemble method:-

→ Learning algorithm
→ Two or more algorithms are combined of similar or dissimilar types called bare learns, so that it incorpora-tes the prediction values from all the bare learners

→ Averaging:- Taking average of the predicted values in regression models or while calculating the probabili-ty in classification models
→ Majority vote:- Considering the prediction with majori-ty vote.
→ Weighted average:- Providing weight for every model and then calculating the average value. Giving high or low preference to specific models

Bagging :-

→ Error due to bias
→ Error due to variance
→ Err
Low bias → ↑ flexible model
High bias → ↓ flexible model
variance → Fluctuation in the data
↓ bias → ↑ variance
↑ bias → ↓ variance
we need ↓ bias & ↓ variance



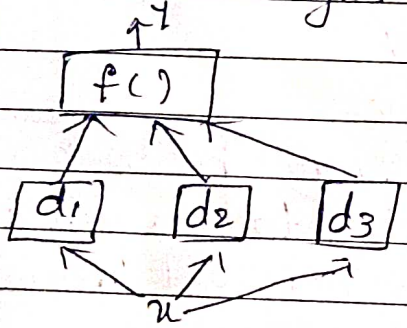
→ Averaging slightly different version of same model → ↑ perform
→ Building models in parallel on various sample data and voting on the prediction from all models
→ Aggr Bootstrap aggregation
→ It is a sampling technique
→ Selecting 'n' sample rows from the 'n' original dataset with replacement
→ All records are equally likely to be a part of bootstrap
→ Multiple bootstraps are created and provided as input to generate decision trees
→ Reduces the variance and overfitting
→ Increases stability & accuracy
ex:- Random forest.
→ Probability of instance present in the bootstrap sample $(1 - 1/n)^n$

Boosting :-

→ Models built in series
→ Weights are adjusted on what it learnt in previous model.
→ weak ones become strong ones
→ First algorithm trained on entire dataset
→ Weights are adjusted or increased for the poorly predicted dataset in previous model.

→ It is done because one training model may not be good for entire dataset but can be good for some data. Therefore it increases the accuracy
→ It boosts the overall accuracy
→ It reduces the bias
→ There can be the chance of model being overfit. Therefore hyper parameter tuning is needed.
→ ex:- XGBoost, GBM, ADABOOST.

Stacking :-
→ Multiple layers of machine learning models placed one over another.
→ Top layer model takes decisions based on output of models in layers below it.



→ fold predictions
→ Any no. of layers can be used
→ Averaging, majority vote, weighted average can be used in top layer

Advantages of ensemble method :-

- Accuracy ↑
- Stability ↑
- Robustness ↑

Disadvantages of ensemble method :-

- Time consuming
- Selection of models for creating an ensemble is difficult.