ML-1 Revision Notes

- **Introduction to ML:**

How is Machine Learning different from Classical Programming?

Classical Programming has rigid rules written by programmers and a lot of hardcoding is involved while ML needs data as input and predicts based on the decision it learns after training on the data.

Classical Programming: Rigid rules written by programmers, a lot of hardcoding.
Machine Learning (ML): Rules are learnt from training a system/model on the data.

When to use ML over Classical Programming ?

ML > Classical Prog. → applications which may not have easily visible patterns.

How to categorize the different types of ML models?

Based on type of Learning ML algo is categorized as:

- Supervised Learning: When output label (Y) is present in the data
- Unsupervised Learning: Output label (Y) is not in the data
- Reinforcement Learning: Used for AI in games, where there is State, Environment and Reward

Criteria: Type of Learning

1. Supervised: When output label (Y) is present in the training data
2. Unsupervised: Output label (Y) is not in the training data
3. Reinforcement: Agent takes an action in an “environment” for maximizing “reward” and changes its “state”

Criteria: Type of Task

1. Classification: Data needs to be classified into categories
2. Regression: Hyperplane needs to closest to datapoints
3. Clustering: Grouping of similar items together as one

4. Recommendation: To recommend items → more likeable to the datapoint
5. Forecasting: Data pattern understanding → predicts future values

---

- **Linear Regression:**

  How does the data look for Linear Regression ?

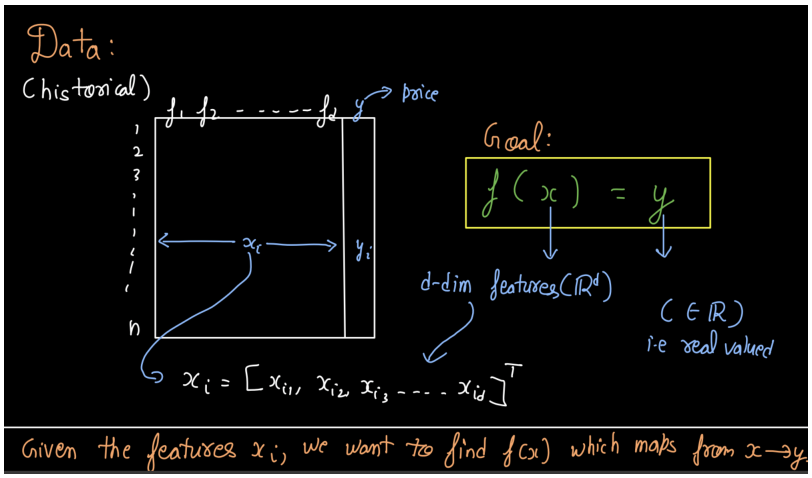
  Regression Problem → Data: n samples with $[f_1, f_2, \ldots\ldots, f_d] \in R^{(n \times d)}$ where

  - sample X_i consists the features
  - That has a target label $y_i \in R$.

  How does the training data look for the Regression problem?

  Regression → Supervised task, target y_i is numerical

  1. Input sample = $X_i \in R^d$, $X_i = [x_{i1}, \ldots\ldots, x_{id}]^T$
  2. Output sample = $y_i \in R$
  3. Training example = (X_i, y_i)
  4. Training Dataset = $\{(X_i, y_i),\ i = 1, 2, .. n\}$,



What is the goal of the ML model ?

Ans: To find $f: X \rightarrow y$ such that $f(x_i) \approx y_i$

How to define function f ?

Algebraic Intuition → find $\hat{y} = f(x_i)$, we can say for Linear Regression:

$$f(x_{i1}, x_{i2}, \ldots\ldots, x_{id}) = w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3} + \ldots\ldots + w_d x_{id} + w_0$$

$$\hat{y}_i = f(x_i) = \sum_{j=1}^{D} w_j x_{ij} + w_0$$

Now, $W^T = [w_1, w_2 \ldots\ldots w_d]$ and $X = [x_{i1}, x_{i2} \ldots\ldots x_{id}]$, then:

$$\hat{y}_i = f(x_i) = W^T X + w_0$$

How does the ML model find the function f ?

Ans: updating weights of the model on the training dataset

How does the ML model update weights ?
Ans: by finding the gradients of the weight through loss function

Is $f(x_i)$ in Linear Regression analogous to $y = mx + c$?
Ans: Yes, it is.

Linear Regression: finding a best D+1 Dimensional line/hyperplane that fits the D-Dimensional data such that $\hat{y}_q \approx y_q$

How to find the best fit line of Linear Regression model ?
Ans: By optimizing the weights vector $W^T = [w_1, w_2, \ldots\ldots, w_d]$

How to say Linear Regression is accurate ?
Ans: minimize $y - \hat{y}$

What loss function to use for optimization ?
Ans: Mean Square Error → finds the mean of the square difference between $\hat{y}$, y .

$$min_{w,w_0} \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

How to find optimal weights ?
Ans: Gradient Descent → minimizes the Mean Squared error to reach global minima

How to find the Gradients of Mean Square Error ?
Ans: We define loss function as :

$$L(w, w_0) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (w^T x_i + w_0))^2$$

On finding gradients with respect to w, loss becomes:

$$\frac{\partial L(w,w_0)}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial (y_i-(w^T x_i+w_0))^2}{\partial w} = \frac{2}{n}\sum_{i=1}^{n}(y_i - (w^T x_i + w_0))\frac{\partial (y_i-(w^T x_i+w_0))}{\partial w}$$

As we know $\frac{d(uv+c+a)}{du} = v$, hence on simplifying, the equation becomes:

$$\frac{\partial L(w,w_0)}{\partial w} = \frac{2}{n}\sum_{i=1}^{n}(y_i - (w^T x_i + w_0))(-x_i)$$

Similarly gradient for w_0 becomes:

$$\frac{\partial L(w,w_0)}{\partial w_0} = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i-(w^T x_i+w_0))^2}{\partial w_0} = \frac{2}{n}\sum_{i=1}^{n}(y_i - (w^T x_i + w_0))\frac{\partial (y_i-(w^T x_i+w_0))}{\partial w_0}$$

$$\frac{\partial L(w,w_0)}{\partial w_0} = \frac{2}{n}\sum_{i=1}^{n}(y_i - (w^T x_i + w_0))(x_i)$$

Updating weights (w, w_0) with a learning rate α :

$$w = w + \alpha \times \frac{\partial l(w,w_0)}{\partial w}$$

$$w_0 = w_0 + \alpha \times \frac{\partial l(w,w_0)}{\partial w_0}$$

Why use Learning Rate ?
Ans: Learning Rate α → hyperparameter to change rate at which Gradient Descent reach global minima

What happens if a too small value of Learning Rate α is used ?
Ans: makes Gradient Descent reach the global minima **very slowly**

What happens if a too large value of Learning Rate α is used ?

Ans: may make the Gradient Descent **overshoot** the global minima

What will be the simplest model for predicting a value ?
Ans: Mean model → the mean of the entire data as its prediction.

After training the model, how to measure model performance ?
Ans: R-Squared **metric**. → measures the performance of Linear Regression over a mean model. It is Defined as:

$R^2 = 1 - \frac{SS_{res}}{SS_{total}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$, where $\bar{y}_i$ is the mean model.

What will be the best value of R^2 ?

Ans: 1, when $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = 0$.

What will be the minimum value of R^2 ?

Ans: - ∞ , when $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 >> \sum_{i=1}^{n}(y_i - \bar{y}_i)^2$

What happens to R-squared if we add a new feature ?
Ans: if the feature is a relevant, R-square ↑.

But if the Feature is not relevant,
R-square would have ↓ when model performance gets worse → countered by a small weight value assigned to this new feature.

Will the R-Square value increase or remain the same ?
Ans: Both are possible → small weights added to new feature might slightly increase model performance

As R-Square fails to compare performance and model complexity (no of features), What other metrics to use ?

Ans: Adjusted R-Squared, defined as:

$$AdjR^2 = 1 - \left[\frac{(1-R^2)(n-1)}{(n-d-1)}\right],$$

where n is the number of samples, and d is the number of features

How does Adj R-Square compare performance and model complexity ?

Ans: if number of features (d) increases

with no significant feature:

- R^2 remains constant or slightly increased ⇒ (n-d-1) ↓ ⇒ Adj-R-squared ↓

with significant feature:

- R^2 ↑ significantly ⇒ $[\frac{(1-R^2)(n-1)}{(n-d-1)}]$ ↓ ⇒ Adj-R-squared ↑

How to determine which features impacts the model most during prediction ?
Ans: The feature with highest weight → most important feature

What does the -ve sign mean in weight of model ?
Ans: The -ve sign means → if feature value ↑ , $\hat{y}$ ↓

Why perform column standardization ?
Ans: Removes ambiguity of which feature is important.

For example: if there are two features $f1$, $f2$:
- $f1$ value range >>> $f2$ value range → Weight of $f1$ >>> Weight of $f2$ ⇒ $f1$ important feature even though $f2$ is the important one.

Now Column Standardization makes : $f1$ value range ≈ $f2$ value range → Weight of $f1$ < Weight of $f2$ ⇒ $f2$ becomes important feature

Assumptions of Linear Regression :

a. **Assumption of Linearity**:
linear relationship between the features x and target variable y

b. **Features are not multi-collinear:**

What is collinearity ?
Ans: 2 features (f_1, f_2), have a linear relationship between them. $f_1 = \alpha f_2$

What is Multi-collinearity ?
Ans: feature f_1 has collinearity across multiple features f_2 , f_3 , f_4

$$f_1 = \alpha_1 f_2 + \alpha_2 f_3 + \alpha_3 f_4$$

Why is Multi Collinearity a problem ?

Ans: Multi Collinearity → non-reliablity on the feature importance and model interpretability.

How to resolve Multi- Collinearity ?
Ans: Using Variance Inflation factor (VIF), defined as:

$$VIF\ for\ f_j = \frac{1}{1-R^2_j};\ where\ R^2\ is\ Rsquared$$

VIF algorithm works as :

- Calculate VIF of each features

- if VIF >= 5 → high Multi-collinearity

- Remove feature having the highest VIF

- Recalculate the VIF for the remaining feature

- Again remove the next feature having the highest VIF

- Repeat till all VIF<5 or some number of iteration is reached

c. Errors are normally distributed:

Used to ensure there are no outliers present in the data

d. Heteroskedasticity should not exist:

Heteroskedasitcity → unequal scatter of the error term → not having the same variance

Why Heteroskedasticity is a problem ?
Ans: model inaccurate or outliers in the data.

How to check Heteroskedasticity ?

Ans: Plotting a Residual plot → Errors ($y - \hat{y}$) vs prediction ($\hat{y}$)

e. No AutoCorrelation:

What is AutoCorrelation ?
Ans: When the current feature value depends upon its previous value

Why is AutoCorrelation a problem ?

Ans: Linear regression assumes $\hat{y}_1 = f(x)$ has to be independent of $\hat{y}_2 = f(x+1)$ → AutoCorrelation contradicts this assumption.

Is there any other way to solve Linear Regression ?

Ans: Closed Form/ Normal Equation

Why use Normal Equation ?
Ans: Finds the optimal weights without any iterating steps as done in Gradient Descent.

The optimal weights: $W = (X^T X)^{-1} X^T Y$

Where X → feature matrix: $R^{n\,(Sample\ size) \times d\,(d\ dimensional)}$, and Y → target vector: $R^{n\,(Sample\ size) \times 1}$

Why even use gradient descent ?
Ans: $(X^T X)^{-1}$ → computationally expensive operation ⇒ Not used when number of Features is high.

What if there is non-linearity in the data , can Linear Regression work ?
Ans: No

What modifications can be done to Linear Regression for the model to be complex enough to fit non-linear data ?

Ans: By using **Polynomial Regression** → transforms linear equation of linear Regression to Polynomial equations

How does Polynomial Regression work ?
Ans: if Linear Regression has $\hat{y} = w_1 f_1 + w_2 f_2 + \ldots\ldots w_d f_d + w_0$;
polynomial introduces terms like $f_1^{\,2}$, $f_3 = f_3^{\,4}$, $f_2^{\,2} = \alpha f_2 + \alpha_0$

making the Model complex for handling non-linearity.

- **Bias-Variance,Regularization:**

  If model A : covers all datapoints with high degree features
  Model B : misses out only a handful datapoints using lower degree features,

  Then which model is better ?

  Ans: Model B generalizes on the data → model captures the pattern of the data and not get influenced by Outliers (hence those points are missed out)

  Model B , a simpler model than Model A →**Occam's Razor**

  What can we say about model A ?
  Ans: Model overfits the data → fitting to outliers/noises in the data

  When to say model underfits the data ?
  Ans: When the model is not able to predict most of the datapoints in the data → model has poor performance.

  How is data split ?
  Ans: Training, Validation and testing dataset.

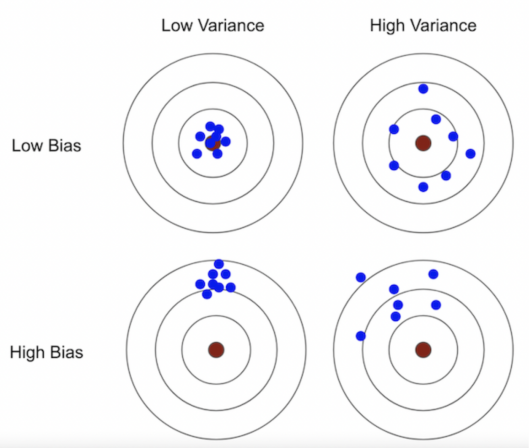
  How is training and test data relate to underfit and overfit model ?
  Ans: an underfitted model → has a high training and test loss
  - An Overfitted model → very low training loss but a high testing loss.

  What is a suitable model ?
  Ans: a tradeoff between both such tha model has a low training and testing loss
  → perfectly fit model

  We can understand Underfit and overfit using Bias and Variance.

  What do we mean by Bias and Variance ?

  Ans: Understanding bias- variance with a target shooting example



Observe
- High Bias → have a wrong aim
- High Variance → an unsteady aim.

How is underfit related to Bias and Variance ?
Ans: Now in Underfitting → predictions are consistent but are wrong→ for different training sets → **High Bias and low Variance**

How is overfit related to Bias and Variance ?
Ans: Now in Overfitting → predictions vary too much and are wrong → for different training set → **Low Bias and High Variance.**

How to control Underfit Overfit tradeoff to find the perfect model ?

Ans: Regularization in the loss function → adds a term $\sum_{j=1}^{d} w^2_j$ → making weight=0 → for insignificant features

How does Regularization makes weights = 0 ?
Ans: With optimization algorithm, → minimizes the values of w_j

$$TotalLoss = \min_{w_j} Lossfunction + \lambda \sum_{j=1}^{d} w^2_j$$

How to control Regularization ?
Ans: By using regularization parameter λ:

since too much regularization → makes the model underfit the data

Too little regularization → makes the model overfit.

Thus ⇒ λ becomes hyperparameter → on tuning gives the overfit-underfit tradeoff

Is squaring of weights the only way for Regularization ?

Ans: No, Regularization is majorly of three types:

A. **L1 / Lasso Regularization :** Uses the term $\sum_{j=1}^{d} |w_j|$ → has $\frac{d|w_j|}{w_j} = 0$, when $w_j = 0$ → making the **weight vector sparse.**

B. **L2/ Ridge/ Tikhonov regularization** : Uses the term $\sum_{j=1}^{d} w_j^2$ → have close to 0 values → for insignificant features.

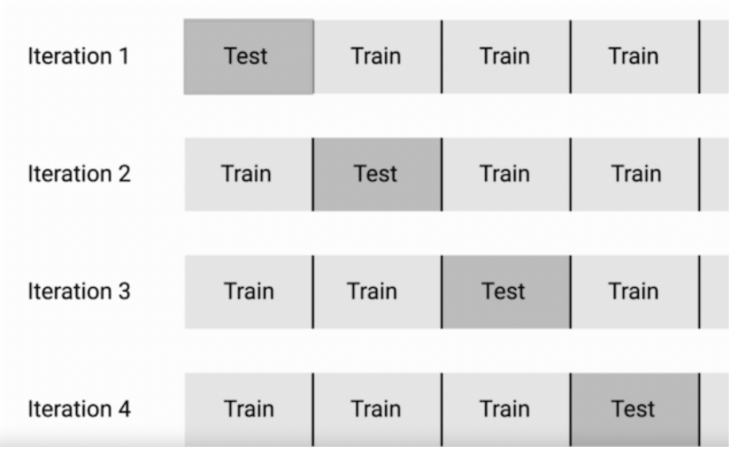
C. **ElasticNet Regularization:** Combination of both L1 and L2 Regularization → with λ_1 and λ_2 as regularization parameters respectively.

$$TotalLoss = \min_{w_j} Lossfunction + \lambda_1 \sum_{j=1}^{d} w^2{}_j + \lambda_2 \sum_{j=1}^{d} |w_j|$$

Why split data into Validation dataset ?
Ans: hyperparameter tuning → done only on Validation data → test data solely used for evaluating the model on unseen data.

What are the steps for a model building ?
Ans: The steps are :
- Train model → with some regularization parameter λ → on training data
- Measure the model performance → with different value of parameters → on the Validation dataset
- Pick the hyperparameters of the best performing model
- Measure the performance of the Best performing model on Test data.

If data is too small to have a validation dataset, what to do then ?
Ans: use **k-Fold CV algorithm** since**:**
- splits data into k smaller sets
- for each iteration, the model trained on k-1 folds
- validated on 1 fold
- performance is avg over all the iterations.



Note: Though k-Fold is a computationally expensive algorithm, it is useful when dataset is small.

- **Logistic Regression**

  Why do we need Logistic Regression ?
  - Useful for binary classification

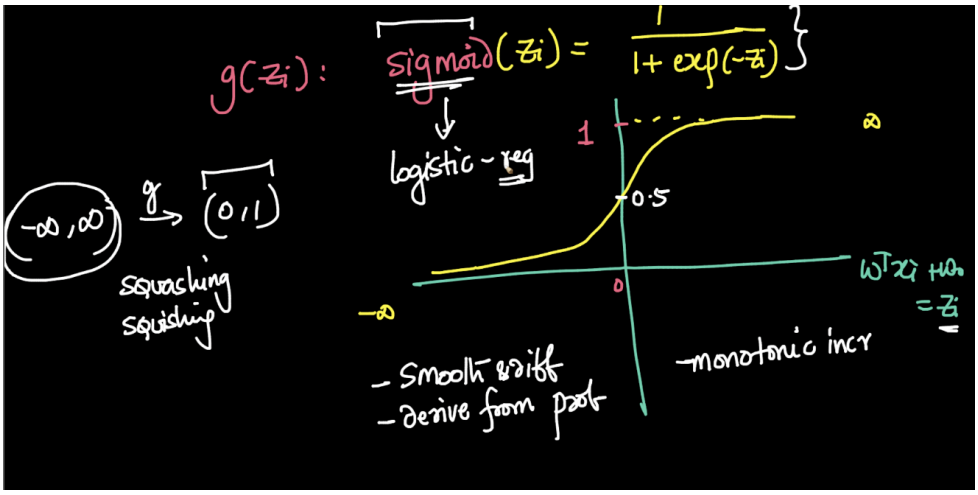
  What are the assumptions of Logistic regression ?
  - Data should be linearly separable

  What is the goal of this algorithm ?
  - To find a hyperplane π which accurately separates the data

  How to perform prediction through logistic regression ?
  - Given labels -> $y_i \in \{0, 1\}$
  - Compute a linear function of x -> $z_i = w^t x_i + w_0 \in \{-\infty, \infty\}$
  - Compute Sigmoid(z_i) -> $\sigma(z_i) = \frac{1}{1+e^{-z_i}}$
  - Predicted label -> $\hat{y}_i = 1 \; if \; \sigma(z_i) > threshold \; else \; 0$

What are the properties of Sigmoid function ?

- Range -> (0, 1)
- Smooth and differentiable at all points



What is the derivative of a logistic regression model ?

$$\sigma'(z) = \sigma(z) \times [1 - \sigma(z)]$$

What is the significance of the sigmoid function ?

- $\sigma(z_i)$ is the probability of x_i belonging to class 1

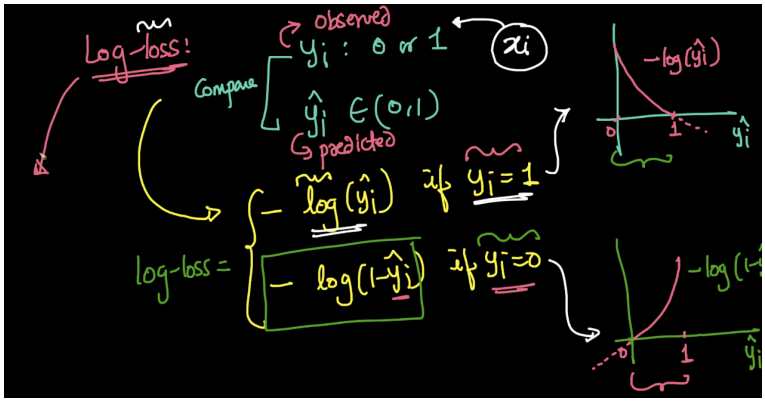
Which loss function is used for Logistic Regression ?

- **Log loss ->** Combination of:
  - $-\log(\hat{y}_i)$ when $y_i = 1$
  - $-\log(1 - \hat{y}_i)$ when $y_i = 0$.

$$Logloss = -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$$

Total loss becomes:

$$TotalLoss = Logloss + \lambda \sum_{j=1}^{d} w_j^2$$



How does logloss help train logistic regression model ?

- $-\log(\hat{y}_i)$ -> High when $\hat{y}_i = 0$ and very low when $\hat{y}_i = 1$.
- $-\log(1-\hat{y}_i)$ ->High when $\hat{y}_i = 1$ and very low when $\hat{y}_i = 0$.

Thus both of these components help penalize the model most when doing wrong predictions.

Also if $y_i \in \{-1, 1\}$, then $Logloss = \sum_{i=1}^{n} log(1 + e^{-y_i z_i})$

But why can't we use Mean Square Error as in Linear regression ?

- Non-convex curve : contains a lot of local minimas
- Difficult for Gradient Descent to reach global minima

What does the derivative of Logloss look like ?

$\frac{\partial Logloss}{\partial w} = \frac{\partial(-y_i \log(\hat{y}_i) - (1-y_i)\log(1-\hat{y}_i))}{\partial w}$, $\hat{y}_i = \sigma(z_i)$ and $z_i = w^t x_i + w_0$

On solving:

$$\frac{\partial Logloss}{\partial w} = -y \frac{\partial z_i}{\partial w} + y \times \hat{y}_i \frac{\partial z_i}{\partial w} + \hat{y}_i \frac{\partial z_i}{\partial w} - y \times \hat{y}_i \frac{\partial z_i}{\partial w}$$

$$\frac{\partial Logloss}{\partial w} = -y \frac{\partial z_i}{\partial w} + \hat{y}_i \frac{\partial z_i}{\partial w}$$

Also since $\frac{\partial z_i}{\partial w} = x_i$; $\frac{\partial Logloss}{\partial w} = (\hat{y}_i - y)\, x_i$

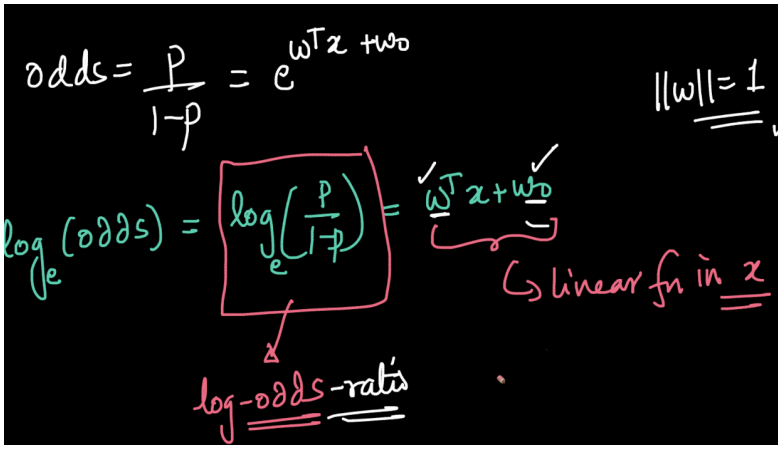
What if we want to predict odds of $y_i = 1$ vs $y_i = 0$?

- **Log - odds:** Shows how the model is similar to a linear model which is predicting log-odds of $y_i = 1 \; vs \; y_i = 0$, defined as **:**

$$log_e(odds) = \log[\frac{p}{1-p}]\ ;\ p = \frac{1}{1+e^{-z_i}} = \frac{e^{z_i}}{e^{z_i}+1}\ and\ 1-p = \frac{1}{e^{z_i}+1}$$

On substituting the value of p and 1-p , and solving it we get

$$log_e(odds) = \log_e[e^{z_i}] = z_i = w^t x_i + w_0$$



Note: Hence the name Regression in Logistic Regression.

Is there any way to use Logistic Regression for multiclass classification ?

- **One vs Rest Method:**
  - For $y_i = \{1, 2, 3.....K\}$, generate k-binary logistic Regression models
  - Final prediction -> Argmax of all of the predictions made by each models

---

- **Classification Metrics**

Issue with Accuracy ?
Ans: Accuracy metric fails when data is imbalanced.

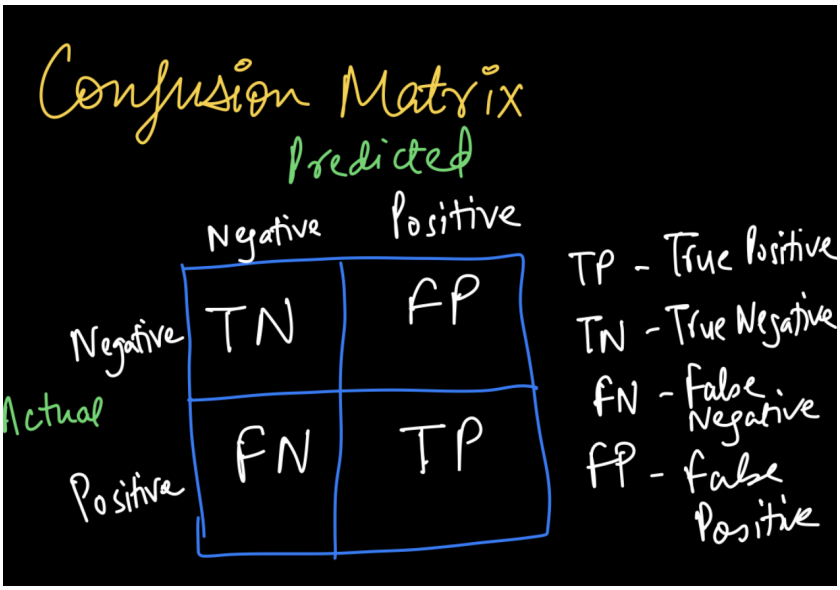
- Consider there are 90 datapoint of class 1 and 10 datapoint of class 0.
- If the model predicts every datapoint as class 1.

⇒ The accuracy of the model is till 90% , which is completely wrong.

What other metric to use ?
Ans: Confusion matrix

When a model predicts, there can be 4 scenarios:

A. **True Positive (TP):** Model predicts True, is Actually True
B. **False Positive (FP):** Model predicts True, is Actually False
    - Aka Type 1 Error
C. **True Negative (TN):** Model predicts False, is Actually False
D. **False Negative (FN):** Model predicts False, is Actually True
    - Aka Type 2 Error



Note:

- For a dumb model that predicts everything as negative, FP = TP = 0
- For an Ideal model that has no incorrect classification, FP = FN = 0

How to use CM to determine if a model with high accuracy is actually good?

Ans. High accuracy can be deceiving. It is only a good model if:

- Both TP and TN should be high
- Both FP and FN should be low

Given a CM, how can we calculate actual positives?

Ans. TP + FN = P (total actual positives)

⇒ Similarly, FP + TN = N (total actual negatives)

How to calculate accuracy from confusion matrix ?

Ans: Accuracy: $Accuracy = \frac{Correct\ Predictions}{Total\ Number\ of\ Predictions} = \frac{TP+TN}{TP+TN+FN+FP}$

Which metric to use, when we cannot afford to have any false positives?

Ans: Precision: It tell us out of all points predicted to be positive, how many are actually postive.

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

For ex

- Misclassification of a spam email as not spam is somewhat acceptable i.e FN
- However, classifying an important mail as spam can lead to major loss i.e. FP

⇒ Here, reducing FP is more critical.

What is the range of precision values?

Ans. Between 0 to 1.

Which metric to use, when we cannot afford to have any false negatives?

Recall / Sensitivity / Hit Rate: It tells us out of all the actually positive points, how many of them are predicted to be positive

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

For ex

- Classifying a heathly person as cancerous and carry out further testing is somewhat acceptable
- However, classifying person with cancer as healthy can be life death situation.

⇒ Here, reducing FN is more critical.

Note:

- **True Positive Rate (TPR):** $TPR = \frac{TP}{TP+FN}$**;** is same as Recall

What is the range of recall values?

Ans. Between 0 to 1.

What all other metrics to look for ?

- True Negative Rate (TNR): $TNR = \frac{TN}{FP+TN}$

- TNR tells out of all the actually negative points, how many have been predicted as False .

- Also called as **Specificity / Selectivity**

- False Positive Rate (FPR):$FPR = \frac{FP}{FP+TN}$

- Intuitively, it tells, out of all datapoints which are actually negative, how many are misclassified as positive

- False Negative Rate (FNR): $FNR = \frac{FN}{TN+TP}$

- Intuitively, it tells, out of all datapoints which are actually positive, how many are misclassified as negative

Which metrics are used in medical domain?

They are used to measure how good a test is at correctly identifying the presence or absence of a disease.

In medical terms,

- **Sensitivity** : proportion of people with the disease who test positive for it
  - ⇒ Test is good to be used as screening test
  - ⇒ There is low chance of missing out a person with disease (low FN)
- **Specificity** : porportion of people without the diease who test negative for it
  - ⇒ It means that test is good for confirmatory test
  - ⇒ There will be low FP

What if F1 score? When is it used?

When both Precision and Recall is equally important measure for the model evaluation, then we use F1-Score:

$$F1Score = \frac{2\times(Precision \times Recall)}{Precision+Recall}$$

For ex, in a fintech company, giving out loans,

- Loss to business if they give loan to people who are unable to repay it (FP)
- Also a loss if they miss out on good people who will be able to repay (FN)

⇒ Here, we need to focus on both FP and FN.

Note:

- F1-score is just Harmonic mean of Precision and Recall.
- F1-score Is also Useful when data is imbalanced.
- F1-score=1 for a dumb model, as both precision and recall are 0.

Why do we take harmonic mean in F1 score instead of arithmetic mean?

Harmonic Mean penalizies the reduction in Precision and Recall more than Arithmetic Mean.

Can we adjust F1 score to give more preference to precision or recall?

A beta parameter can be added to make F-1 Score have more attention on either Precision score or Recall score.

$$Fbeta = \frac{(1+beta^2) \times Precision \times Recall}{beta^2 \times Precision + Recall}$$

- Beta = 2 if Recall more important than Precision.
- Beta = 0.5 when Precision is more important.

What is AU-ROC? Where is it used?

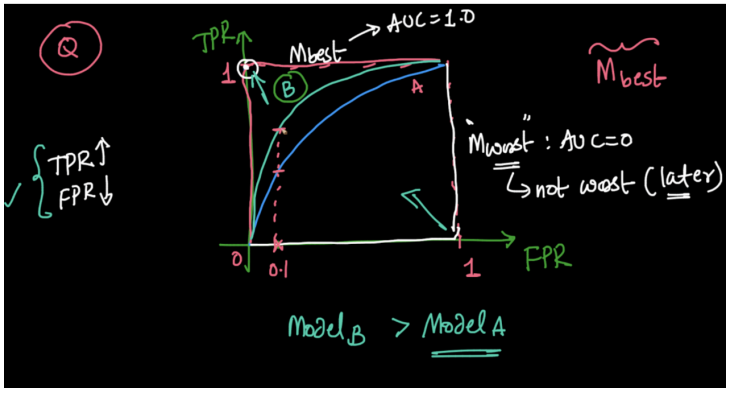
- AU-ROC : Area Under Receiver Operating Characteristic Curve

- Used to find the best model by plotting TPR and FPR by sorting value of $\hat{y}_i$ and keeping them as threshold for final prediction (y_{pred}).

How do we determine the better model using AU-ROC?

After plotting whichever curve has the most area covered tends to be the better model.
For ex:

- Here, AUC of model B > model A
- Hence, model B is better



Note:

- Unlike precision, recall or F1-score, AU-ROC does not work well for highly imbalanced data.

What is the fundamental difference between AUC and the other metrics?

When we calculate Precision, Recall or F1 score

- We calculate it for a certrain threshold on $\hat{y}_i$
- This threshold is 0.5, by default

On the other hand, for AU-ROC

- we are calculating it using all possible thresholds

What will be the AUC of random model?

The ROC curve will be diagonal. ⇒ Hence AUC will be 0.5

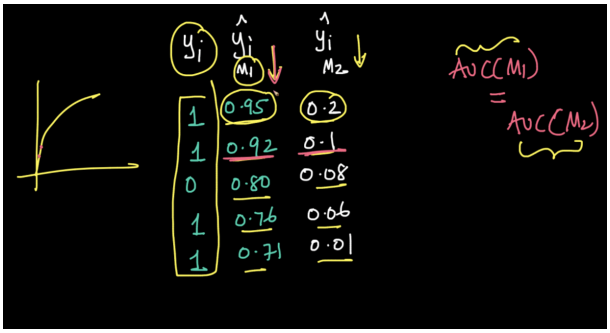
What to do if a model's AUC < 0.5 ?

A simple fix is to invert your predictions.

⇒ After inversing you will get area of 1-(actual area value)

Does AUC depend on the actual values of $\hat{y}_i$?

No. AU-ROC depends on the how the ordering of the $\hat{y}_i$ is done and not on the actual values of $\hat{y}_i$.



For Example, say we have two models M1 and M2

- Actual y labels: [1, 1, 0, 1, 1]
- $\hat{y}_i$ for M1: $[0.95. 0.92,\ 0.80,\ 0.76,\ 0.71]$
- And $\hat{y}_i$ for M2: $[0.2,\ 0.1,\ 0.08,\ 0.06,\ 0.01]$

⇒ Since both have the same ordering of how $\hat{y}_i$ are arranged , hence AUC(M1) = AUC(M2).

AU-ROC is highly sensitive to imbalanced data, what metric can we use there?

We can use Area under the Precision-Recall cure (AU-PRC).
This is a very good metric for imbalanced data.

How is PRC plotted?

- Precision on y axis
- Recall on x axis
- Similar to ROC curve, we'll take each $\hat{y}_i$ as threshold

Then we take the area under PRC curve to get AU-PRC.

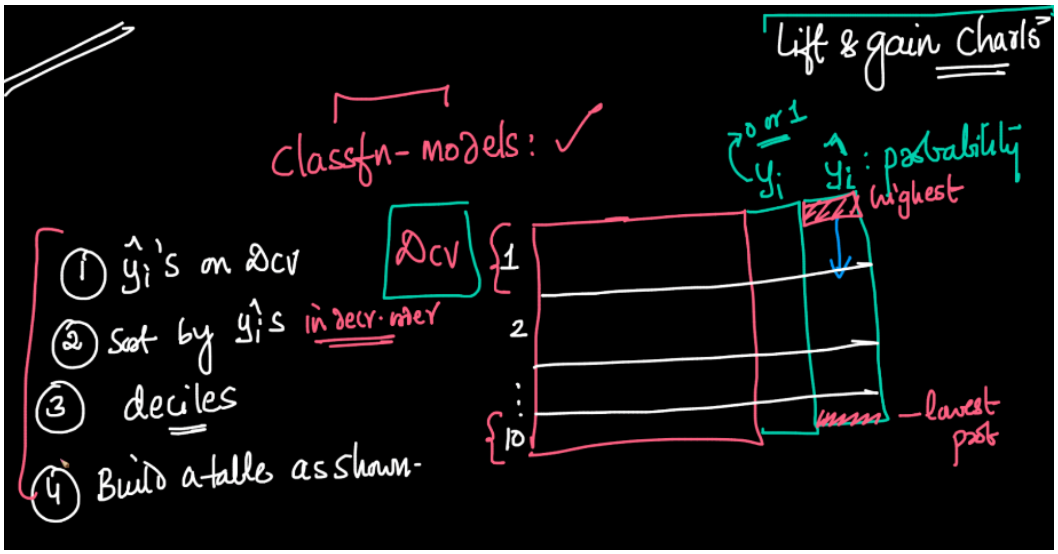
Which metric can we use to know how a model would do business wise?

- Business people need to know how our model's differences would make in the business term compared to random targeting.
- ⇒ Lift and Gain charts are used.

- ⇒ Help us graphically understand the benefit of using that model (in layman terms)

How to make lift and gain charts?

- **Step 1:** Obtain the predictions $\hat{y}_i$ on cross-validation data Dcv
- **Step 2:** Sort the data predicted probability in desc order We will have highest probability at the top, lowest at the bottom.
- **Step 3:** Break the sorted cross valdiation data into 10 groups (or deciles). We get the data for each decile.
- **Step 4:** Using these deciles, we built a table as follows
    - ⇒ 1st decile will have datapoints with highest pred. probability
    - ⇒ 10th decile will have datapoints with lowest pred. probability



Calculation of Gain:

- Dividing the cummulative responses by total number of responses
- i.e. cummulative number of postives by total number of positives.

Calculation of lift:

- Dividing total % of positive point upto that decile (gain) for a smart model by total % of positive points if we had a random model
- in other terms, it is ratio of gain of our model to a gain of random model

⇒ After calculating the lift and gain values, we plot them for each decile.

What does gain mean?

Gain for ith decile tells us "what percentage of positive points are in i^{th} or smaller decile"
For example:

- If gain is 97.87 % for the 8th decile
  ⇒ It means we cover 97% of positive datapoints, till 8th decile.

What does lift mean?

- It means Cummulative percentage of positive points till ith decile divided by cummulative percentage of positive points by random model
- It is intuitively telling how much better is model compared to random model.

How to use Accuracy metric even when data is imbalanced ?
Ans: G-Mean: When data is imbalanced, Geometric-Mean(G-Mean) measures model performance on both the majority and minority clases.

$$GMean = \sqrt{Specificity \times Sensitivity}$$

Which metric to use when? (Cheat Sheet)

- If we want probabilities of classes: **Log loss**
- If classes are balanced: **Accuracy**
- IF classes are imbalanced:
  - If FP is more critical: **Precision.**
  - If FN is more critical: **Recall.**
  - **F1 score** is a balance between precision and recall.
  - If our concern is both classes (TN and TP)**: ROC_AUC**
- If severe imbalance**: PR AUC**

How are performance metrics different from loss functions?

- Loss functions are usually differentiable in the model's parameters.
- Performance metrics don't need to be differentiable.
- A metric that is differentiable can be used as loss function also. For ex: MSE

- **KNN**

Why is KNN required ?

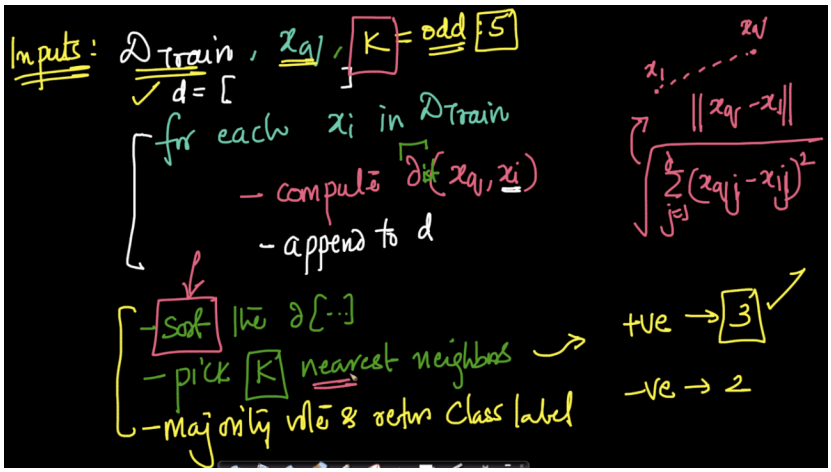
- Simple to use for multiclass classification
  - One v/s rest method would have to be employed for Logistic regression
- Non-linear data handling
  - Logistic regression needs creating polynomial features

What is the only assumption of KNN ?

- Homogeneous Neighborhoods -> Similar things are close to each other

What is the KNN Algorithm ?

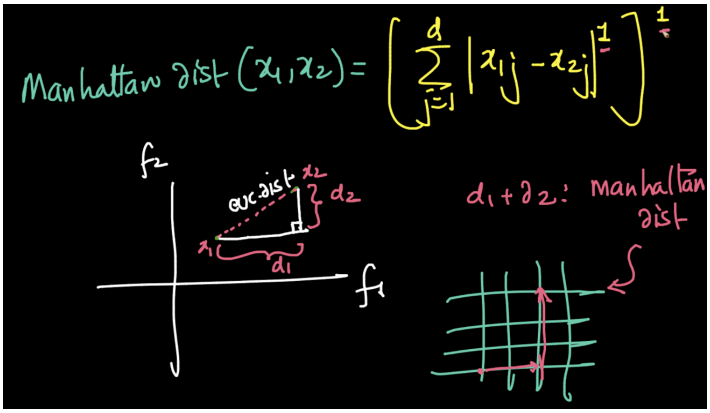
1. Training → No training required, just store the training data
2. Testing Prediction:
    a. Find out the distance of all the points from each point using a distance metric
    b. Filter out k-nearest neighbors of each point
    c. Assign majority class label to the point



What are the metrics used for distance calculation ?

1. Manhattan Distance:

$$d(x_q, x_i) = \sum_{j=1}^{d} |x_{qj} - x_{ij}|$$



- Time complexity -> O(n)

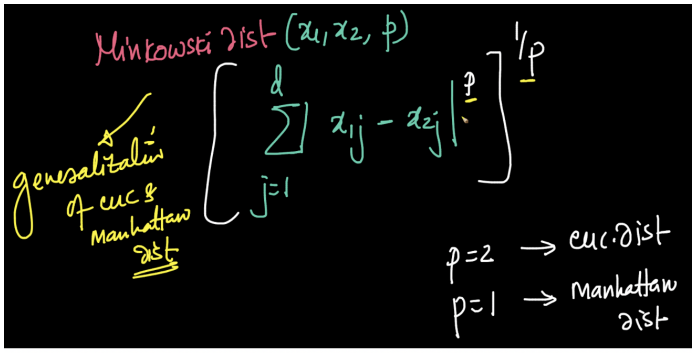
2. Euclidean Distance

$$||x_q - x_i|| = \sqrt{\sum_{j=1}^{d} (x_{qj} - x_{ij})^2}$$

- Suffers from curse of dimensionality
- Time complexity -> O(n)

3. Minkowski Distance
   - Generalized version for pth degree

$$Minkowski(x_q, x_i) = [\sum_{j=1}^{d} |x_q - x_i|^p]^{\frac{1}{p}}$$



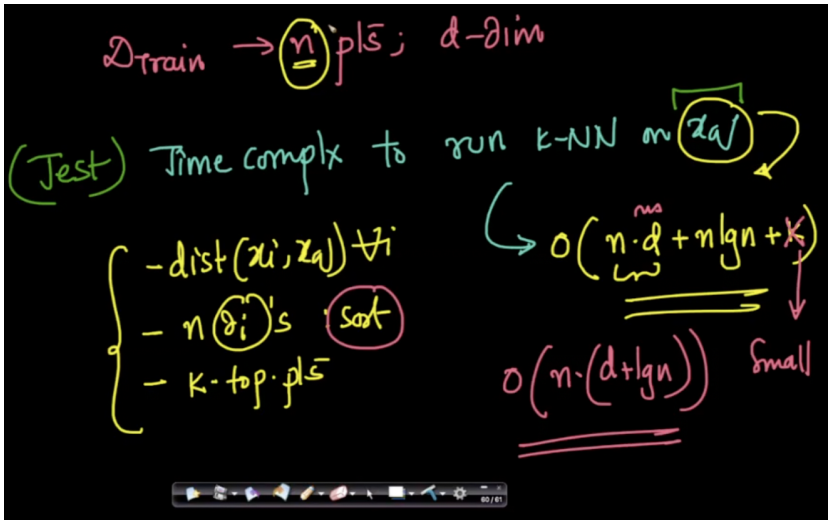
But should we give equal weightage to all the k neighbors ?

- In KNN similar points are closer to one another
- So it makes sense to give more weightage to points closer to x_q
- Advantage -> Class label of x_q depends more on closer points (even if the class label data points are in minority)

$$Weightage = \frac{1}{dist(x_q, x_i)}$$

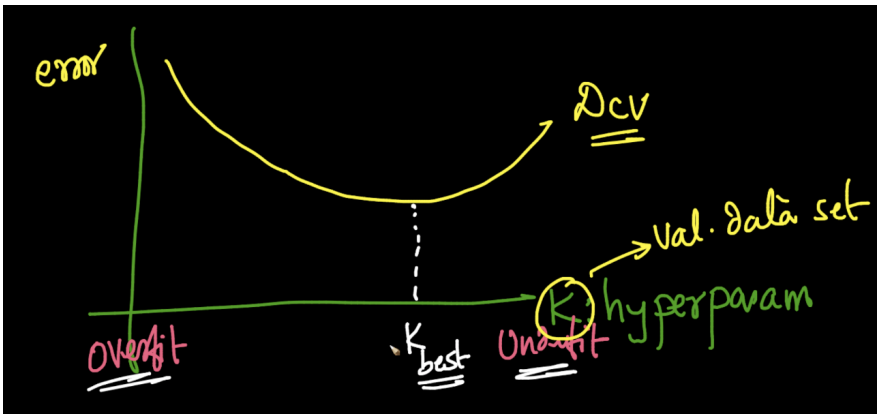
What is the train/test time complexity of KNN?

- Training -> $O(1)$ (since kNN just stores the data points)
- Test -> $O(n(d + logn))$
  - Distance calculation -> $O(nd)$
  - Sorting of distances -> $O(nlog(n))$
- n : Total number of points
- d: No. of features present in the dataset



What is the Bias-Variance Tradeoff of KNNs ?

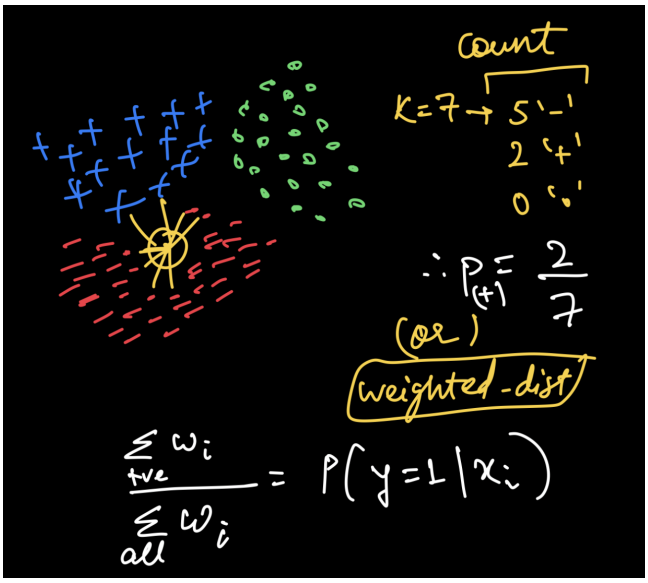
- High variance : Small K -> Starts fitting outliers/noise



- High bias : Large K -> Higher inaccurate predictions since they are affected by far away points also

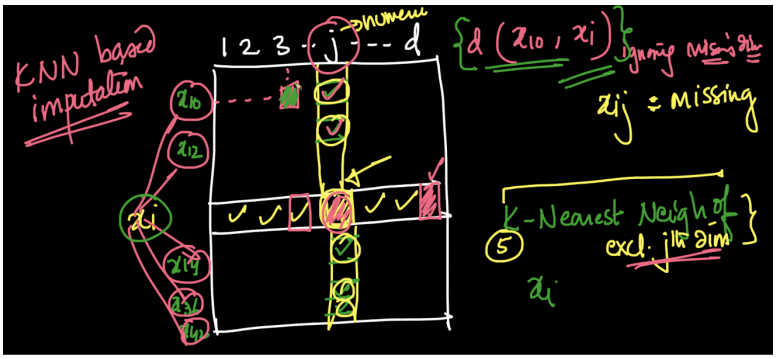
What is the Probabilistic Class label for KNN ?

$$P(y = a \mid x_i) = \frac{Count\ of\ a\ class\ label\ datapoints}{Count\ of\ total\ number\ of\ neighbors} = \frac{\sum_{class=a} w_j}{\sum_{all} w_j}$$



How can kNNs be used for Imputation?

1. Missing value x_{ij} -> Average of jth feature of K nearest neighbors



- **Imbalance Data**

1. **Effect of Imbalance data on kNN:** as value of k increases, the data imbalance impacts the model predictions more and more

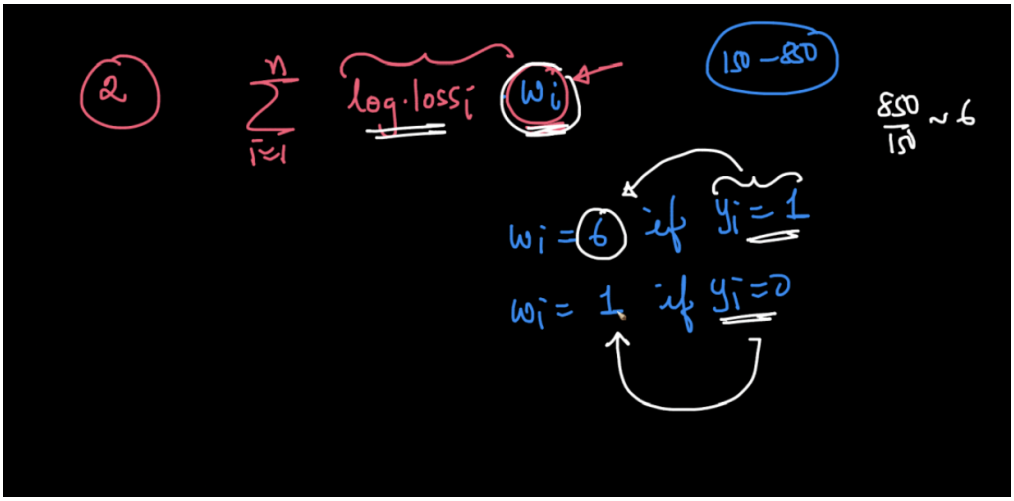
2. **Effect of Imbalance data on Logistic Regression:** Suppose if we have more -ve class samples than +ve class samples. Then the -ve class dominates the log loss function.

Which makes the hyperplane π to be pushed away from the -ve class sample such that it passes the +ve samples, thus making the model predict every point as -ve class.

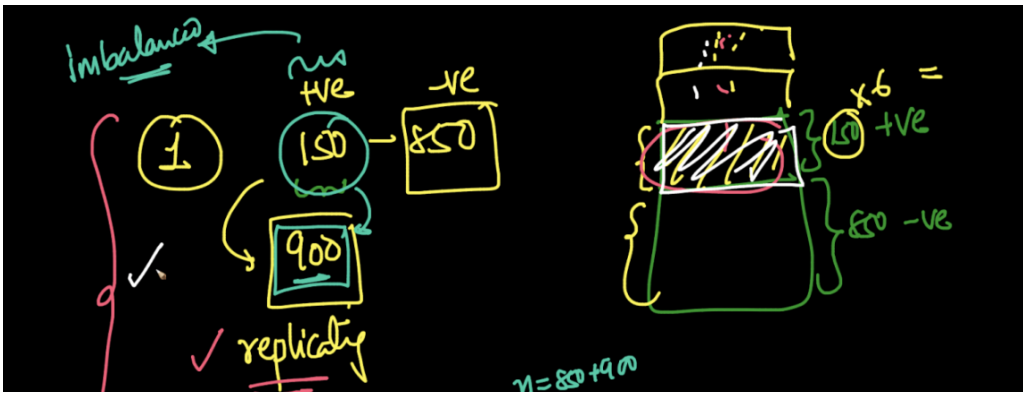
3. Handling Imbalance Data

A. Weighted Loss: Add a class weight to the loss function. Which increases the weightage loss value of the minority class

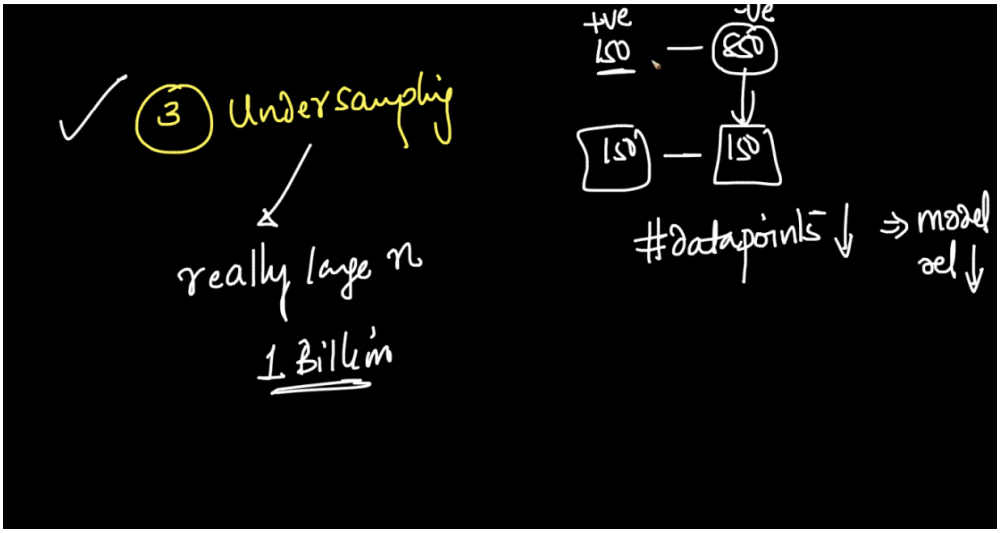
$$w_{minority} = \frac{Number\ of\ samples\ of\ Majority}{Number\ of\ samples\ of\ Minority}\ ;\ w_{majority} = 1$$



B. Oversampling: Replicating the minority class such that, the number of samples in minority class is same as the number of samples in majority class.



C. **Undersampling:** Removing the number of samples of the majority class such that the number of samples in minority class is same as the number of samples in majority class.



Note: Leads to Information loss that reduces model reliability . Hence only reliable if number of samples is really large

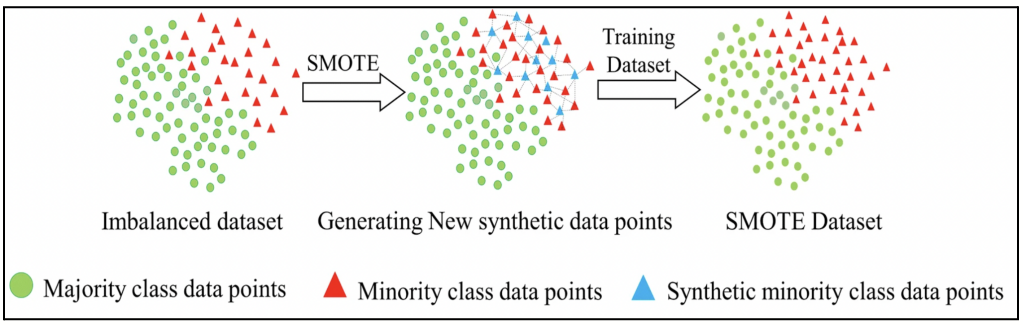
D. **SMOTE:** Works similar to kNN. First it selects a minority sample datapoint x_1.

Then based on the value of k, finds the distance between the k-nearest neighbor and the datapoint d.

Selects a random value between [0,1] for each k-neighbors, and multiplies with the distance vector which is added to the features of datapoint x_1.

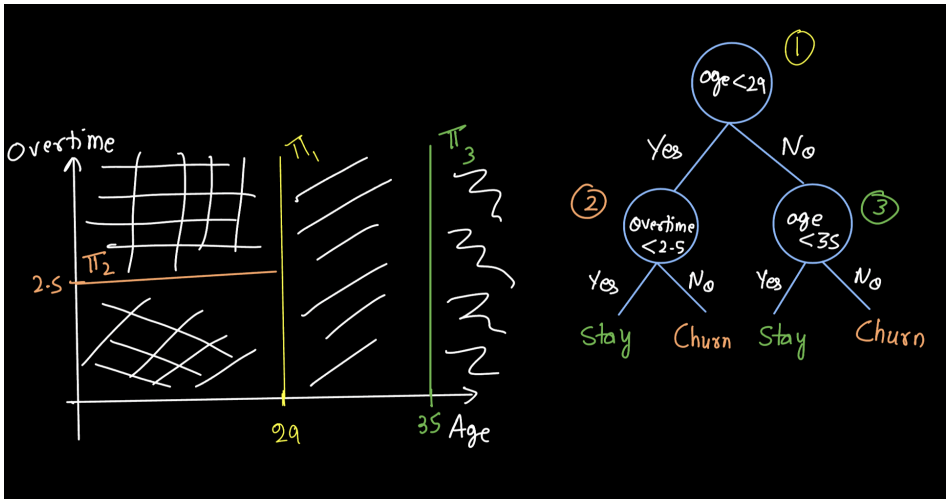
This creates a new sample datapoint x_{new}: $x_1 + [random\ value] \times d$

Thus creating synthetic minority class data samples.



- **Decision Trees**

  What are decision trees ?



- A decision tree is a bunch of nested if else statements (rules).
- Topmost node -> root node
- Bottom nodes -> leaf nodes.

Some Advantages of using decision trees

1. Useful for predicting non-linear boundaries
   - Decision-boundary in DTs are made up of axis-parallel hyperplanes
     How to predict a slanted line then ?
     By using multiple axis-parallel lines in a staircase manner

2. Decision trees are easily interpretable. How ?

   - Each node can be considered as a rule for an if-else based condition
   - As an example the above mentioned decision tree can be broken down into rules as:

     If age < 29:
       If overtime < 2.5hrs:
         Employee will stay
       else:

```
        Employee will churn
    else:
      if age < 35:
        Employee will stay
      else:
        Employee will churn
```

Splitting of nodes

Pure nodes: Nodes which are purely homogeneous in nature i.e. contain data points belonging to only one class

Impure nodes: Nodes which are heterogeneous in nature i.e. contain data points belonging to multiple classes

So what is the purpose of splitting a node ?

- To create pure nodes
- Pure nodes need not be splitted further
  Why ?
  Because in this case uncertainty of prediction would be the lowest

Impurity Measures

How to decide which features to use for splitting nodes ?

- Using impurity measures
- Impurity Measures:
  - They are used to measure the heterogeneity of a node
  - Impurity of pure node = 0

Some Properties of impurity measures for binary classification

- N = #Points belonging to class 1
- M = #Points belonging to class 2

1. Impurity of a pure node = 0
2. It has 2 minimas (N=0, M=0)
3. N=M: Impurity is maximum
4. Impurity is Symmetric around the maxima
5. It increases from minima to maxima then decreases from maxima to minima
6. Should be non-negative for all points

Some impurity measures

1. Entropy

Imagine Y be a discrete random variable:

We define entropy (H) as

- $H(Y) = - \sum_{i=1}^{k} p(y_i)\log(py_i)$

where $p(y_i)$ is the probability that random variable $Y = y_i$

Entropy for binary classification

- $H(Y) = -P(Y=0)\log(P(Y=0)) - P(Y=1)\log(P(Y=1))$

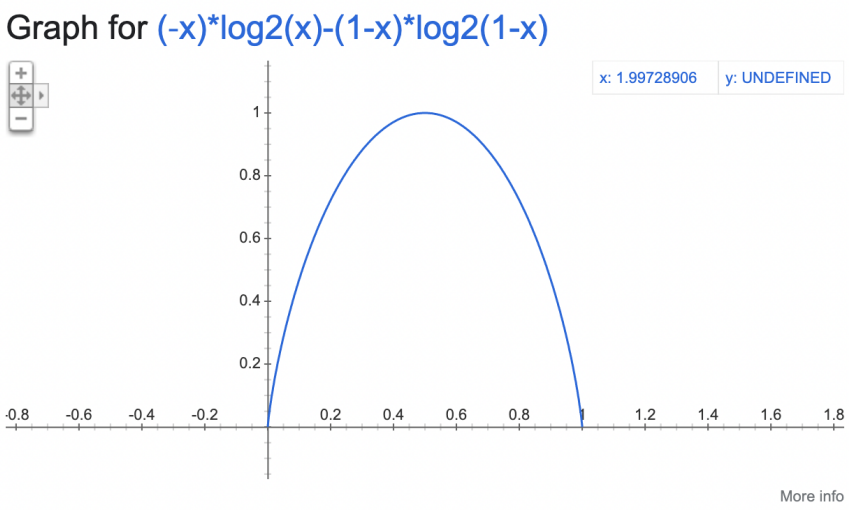
where

- $P(Y=0)$ is the probability $Y = 0$
- $P(Y=1)$ is the probability $Y = 1$

For $P(Y=1) = p$ entropy becomes:

- $H(Y) = -p\log(p) - (1-p)\log(1-p)$

Plot of entropy



Some properties that can be observed from entropy's (for Binary classification) formula/plot

1. The values of the plot range from 0 to 1.
2. The maxima lies at x = 0.5
3. Maximum value of entropy for binary case will be 1 (log base 2).

How to use Entropy for node splitting ?

At each node we try to find that split which minimizes the entropy.
Why ?

- Assuming that recursively splitting in such a manner can lead to pure leaves in all cases
- Greedy in nature. Doesn't necessarily happens

Disadvantages of entropy:

1. It requires a log of probability.
2. log is computationally expensive
3. We have to calculate entropy for each feature at each node
4. This becomes time consuming.

2. Gini Impurity

Gini Impurity of random variable Y is given by:

- $GI(Y) = 1-\sum^{k}_{i=1}p(y_i)^2$

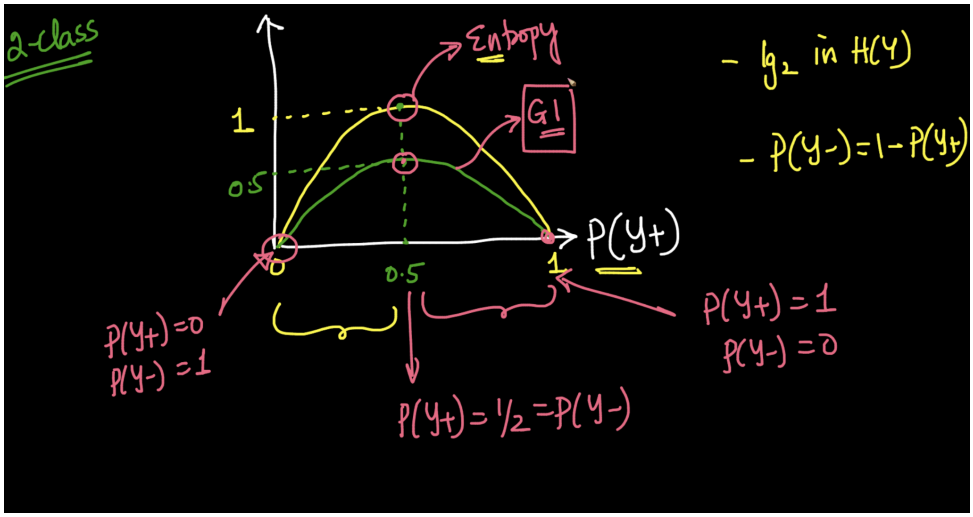
For a binary classification:

$GI(y) = 1-[p(y_i=1)^2+p(y_i=0)^2]$

What is the difference in behaviors of Gini-Impurity and entropy ?

- Gini-Impurity is high when entropy is high
- It is low when the entropy is low

Let's plot the graph of Entropy as well as Gini Impurity:



Notice that,

1. When the nodes are pure i.e for
   - if $p(y_+) = 1$ and $p(y_-) = 0$, or

- if $p(y_+) = 0$ and $p(y_-) = 1$

the Entropy and Gini-Impurity are zero

2. when the probability of $p(y_+) = 1/2$ and $p(y_-) = 1/2$
   the entropy and Gini- Impurity are maximum

So, we can conclude that Gini-Impurity has the same behavior as entropy.

Information Gain and Gini - Reduction

When a node is split into multiple children, each child has a separate impurity

How can we compare these multiple impurity values obtained for each feature ?

- We combine these impurity values

Information Gain: Entropy of parent - Weighted average of entropy of children

Gini Reduction: G(Parent Node) - weighted average of gini impurity of children

Handling numerical features

Typically, we will have threshold and

- We compare the each value of feature with a threshold
- and split them based on the threshold.

How to choose the threshold ?

1. Arrange values of feature in increasing order
2. Set each value of feature as threshold
3. Next, we calculate the IG of that split.
4. Threshold giving the maximum IG is selected as IG obtained

Bias/Variance in Decision Trees

- Overfits when Depth of tree is too high
- Underfits when Depth of tree is too low

Miscellaneous Info:

A decision Stump is a Decision Tree with depth = 1

A shallow tree is with a small depth value

Deep tree is when the tree's depth is very large
Outliers: Outliers impact Decision Tree when depth is very large

Standardization: Standardization does not impacts entropy, Gini Impurity and information gain

Train Complexity: o(nlog(n) d)

Run Time Complexity : o(d)

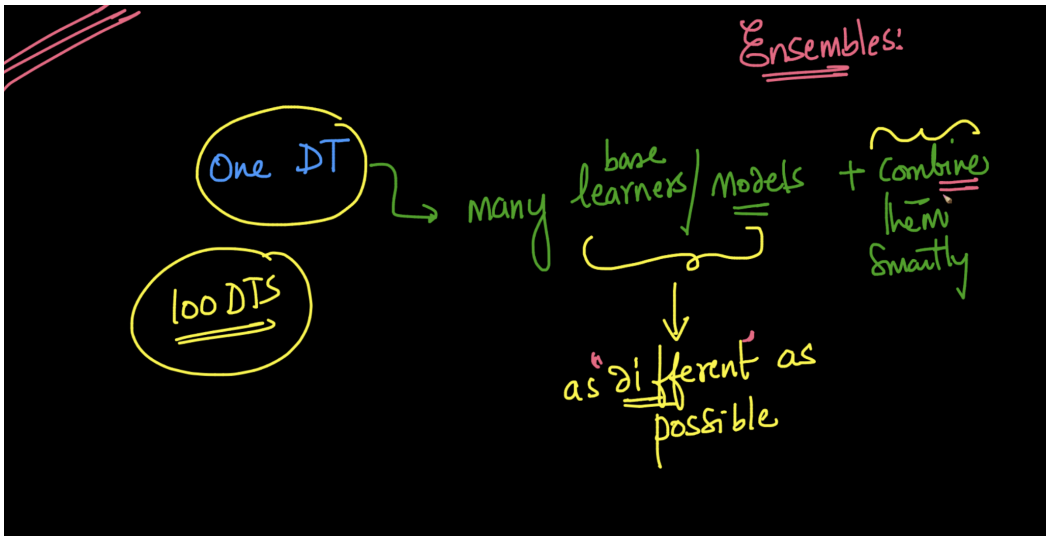
n: Nodes in the tree

d: Depth of the tree

- **Bagging and Boosting:**

  What is Ensemble Learning ?

  Ensemble Learning: **Training multiple base learners** or models which are as different as possible and **combining them smartly**



  What is bagging ?

  Bagging is short for Bootstrap Aggregation where:
  - Bootstrapping -> randomly creating samples of data out of a population with replacement

- Aggregation -> Clubbing predictions of each model to get the final outcome

What are Random Forests ?

A bagging ensemble which contains Decision Trees as the base learners

How to build Random Forests ?

- Sample m data points with replacement to get D_m'
- Train K different models for the K different datasets
- After training we cross validate each model
- Now, we do Aggregation
    - We use majority vote for Classification
    - We use Mean/Median for Regression

How to introduce Randomness ?

1. Row-sampling: For each base learner, we randomly select a subset of training data
2. Column-sampling: For each base learner we can select a subset of the columns
3. Depth tuning: Deeper trees have a higher chance of being different from each other.

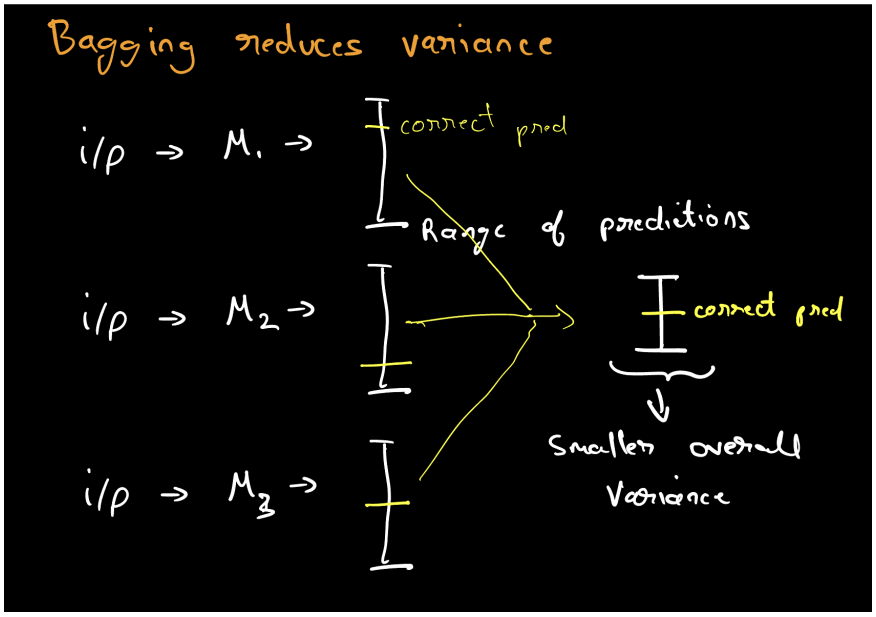
How to perform Validation in Random Forests ?

- Using Out of Bag (OOB) samples -> Samples in the training set which are not selected even once while bootstrapping

What is the Bias/Variance tradeoff in RFs ?

- Decision Trees in RFs have high depths (non-shallow).
- High Depth & less data -> High variance
- Aggregation is performed on these high bias/variance models

How does bagging reduce error ?



- Error = Bias^2 + Variance + Irreducible error.
- Since bagging reduces the variance while keeping the overall bias mostly the same, the error decreases

How is Feature importance calculated in Random Forests ?

- Compute the feature importance of a feature in each Decision Tree
- Take the average of these values.

What if some base learners don't have the feature?

- The importance of that feature will be considered 0 in that Base learner

How to Train Random Forests ?

- Base learners can be trivially parallelised.
- Each model is trained independently
- The time complexity thus becomes O(k * max_depth of tree)

OOB Score: Performance of the ensemble on OOB sample is called OOB score

What are Extra Trees/ Extremely Randomized Trees ?

1. It has random row/column sampling and aggregation
2. Additionally, it randomly picks the threshold (τ) to split for numerical features.
3. More number of base-learners are required

What is Boosting ?

Base learners typically have low variance and high bias

What sort of DT models have high bias?

- Shallow Trees or Decision Stump

The output of these models is combined in an Additive Manner

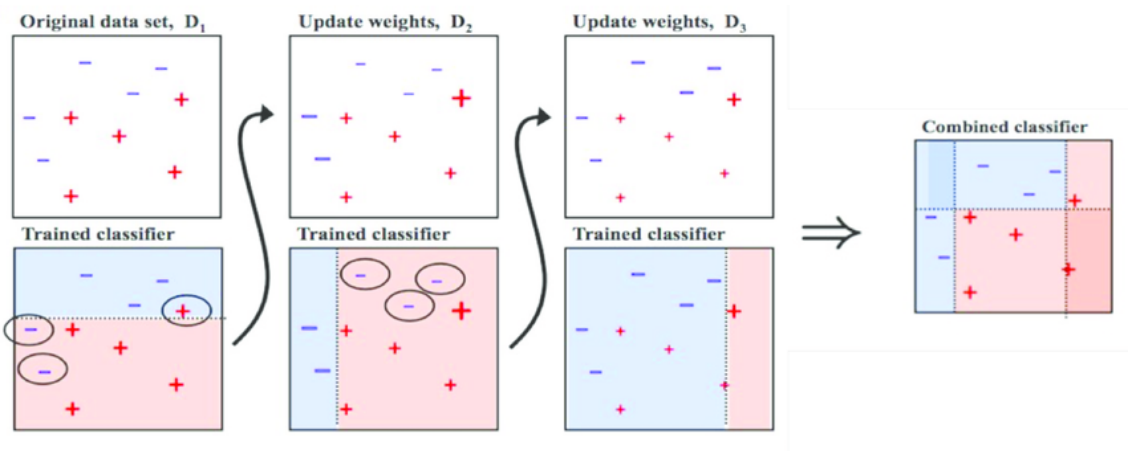
In Boosting,

- we build a bunch of simple models and
- Each model is trained using the residual of the previous model.
- use these model to build an additive weighted model

Process of creating a boosting ensemble

- Firstly, a model is built from the training data.
- Then the second model is built which tries to correct the errors present in the first model.
- Above steps are repeated till either:
  - All points are predicted correctly
  - Max no. of models have been added

AdaBoost



AdaBoost means Adaptive Boosting

- Assigns weight to points proportional to the error produced by the base learner
- So, to get the final model, we multiply each model with a weight.
- The output will be sign of this multiplication i.e. +ve/ -ve

Disadvantages of AdaBoost

- We don't have flexibility to use any loss function

- More susceptible to outliers

What are Gradient Boosted Decision Trees (GBDT) ?

- Fits a decision tree on the residuals error of the previous tree.
- So, each new tree in the ensemble predicts the error made by the previous learner

How to build and train a GBDT ?

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:
$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$
2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:
    $$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$
    2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:
    $$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$
    4. Update the model:
    $$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$
3. Output $F_M(x)$.

What is XGBOOST ?

XGBoost is one the popular library known for its well optimized code

For example :

- if you have numerical feature f_i
    - instead of trying all the values for thresholding,
    - It builds a histogram of data and uses simple rules like quartiles and percentiles to make thresholding.
- It also does multi core optimization (parallelization)
    - it'll compute each branch of a base learner on a different core to speed up the process.

Some commonly used hyperparams of XGBOOST

- Number of estimators (M)
- Depth
- v : learning rate
- Col sampling/ row sampling

What is LightBGM ?

It has majorly 2 optimisations over XGBOOST

1. GOSS - Gradient based one side sampling
   - drop the points in which the
   - $res_{i,\,m}$ is small
2. Exclusive Feature Bundling (EFB)
   - i.e. smart sampling (probability of getting large residual value is higher)
   - It looks at all the dimensions
   - tries finding feature pairs s.t they are exclusive

What is Stacking ?

Here, we are taking the outputs of the perfectly build models and stacking them together to train a Meta-classifier to get the final output

What is Cascading ?

Training a base-learner on a dataset different from the previous base learner such that it contains points wrongly predicted by the previous learner

Why is GBDT used more often than RF?

1. Any differentiable loss function can be used
2. GBDT has cheaper Run-time because
   - the base learners are shallow and
   - Random forest has deeper trees and
   - the number of trees to train in GBDT are comparatively less

When should we use Cascading and stacking ?

Cascading is used when the risk or cost of mistakes is high, and the data is highly imbalanced.

- Like fraud transaction detection in amazon

What about the explainability of the model?

- We make sure that every model is explainable, so that we can explain the output using these models

- We will see few algorithms, like **LIME and SHAP** which can explain any black box algorithm after few lectures in Deep Learning.

Stacking is mostly seen in kaggle competitions, not so much in the real world.

- **Naive Bayes**

1. **Q. Where is Naive Bayes used?**
   Useful when classification of text needs to be done (Spam/Ham, Positive/Negative sentiment, etc.)

2. **Q. What does the training Data look like?**
   Text data needs to be preprocessed as:
   - Tokenization
   - Lower casing
   - Stop word Removal

   Final data to show either occurrence (0 or 1) (**Bernoulli NB**) or count (**Multinomial NB**) of a word in given sentence.
   Since it is not a distance based algorithm, scaling/standardizing data not needed.

3. **Q. What is the Algebraic Intuition behind NB? How is it able to classify the text data?**
   NB needs to calculate two values:
   - Probability that given text is spam P(y=1|text)
   - Probability that given text is ham P(y=0|text)

   Higher value gives the result.

   Q. How are these values calculated?
   These are calculated as the **product** of two components:
   - **Class Priors:**

   $$P(y = 1) = \frac{Number\ of\ Samples\ with\ y=1}{Total\ Samples};\ P(y = 0) = \frac{Number\ of\ Samples\ with\ y=0}{Total\ Samples}$$

   - Likelihoods:

   $$P(w_1,\ldots..,w_d|y = 1) = \prod_{j=1}^{d} P(w_j|y = 1);\ P(w_1,\ldots..,w_d|y = 0) = \prod_{j=1}^{d} P(w_j|y = 0)$$

   Where

- d is the total number of words in the whole dataset

- And $\prod_{j=1}^{d} P(w_j|y = 1) = \frac{number\ of\ texts\ where\ w_j\ exists\ and\ have\ y=1}{number\ of\ samples\ which\ have\ y = 1}$

Hence, the equations become:-

- $P(y = 1|w_1..w_d) = P(y = 1) * P(w_1,.....,w_d|y = 1)$ and
- $P(y = 0|w_1..w_d) = P(y = 0) * P(w_1,.....,w_d|y = 0)$

If $P(y = 1|w_1..w_d) > P(y = 0|w_1..w_d)$, then class label = 1, else class label = 0

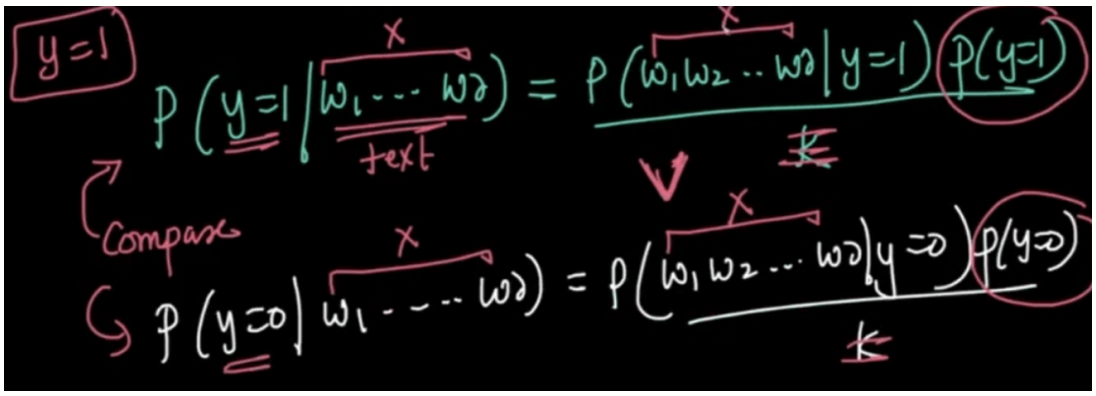
Q. Why is it called "Naive" Bayes theorem?

This algorithm is based on the Bayes theorem, and it makes a naive assumption.

Q. What is the Naive assumption of NB?

The event of occurrence of words is independent of each other, but conditioned on the class label (y = 1 or y = 0). For eg

- P("Nigerian prince" | y=1) = P("Nigerian"|y=1). P("prince"|y=1)



Q. What is the Train time complexity of NB?

- Training Complexity: $O(nd)$, where n is the number of samples and d is the total number of words in the data.

Q. What is the Test Time complexity of NB?

- Testing Complexity: $O(k)$, where k is the number of words in the query text

Why is Laplace/Additive Smoothing used in NB?

Used to prevent likelihood and class prior values from being zero, when a word in the test data is not present in the training data.

$$P(w_j|\ y\ =\ k)\ =\ \frac{n_{jk}+\alpha}{n_k+\alpha C}$$

Where

- n_{jk} is the number of samples which have word j and have class label as k ,
- n_k is the number of samples which have class label as k,
- α is the laplace smoothing which is a hyperparameter and
- C is the distinct values that w_j can take.

Note :

- $C = 2$, in case of Bernoulli NB where w_j can only have 0 or 1 values (exists / does not exist).
- This is done for the likelihood of ALL words, and not just for new words.

Q. What effect does the laplace smoothing value (α) have on the model?

- When α is very large, the model tends to underfit. Since the $P(y = k|w_1 \ldots w_d)$ would depend only on the Class Prior.
- Conversely, the model overfits when α is very small

Q. How is NB affected by Imbalance in the data?

- Naive Bayes does get impacted by Imbalance in the data.
- if the likelihoods are same, then the $P(y = 1|w_1 \ldots w_d)$; $P(y = 0|w_1 \ldots w_d)$ is impacted by class priors and the class which has more samples tends to dominate the prediction
- Hence it is a good idea to re-balance data.
- It does not affect the likelihood values.

Q. How does the problem of Underflow affect NB?

- Underflow happens in Naive bayes because the product of likelihood values of d words would become a very small value. This is because:
  - d is a very large number in practice
  - Likelihood values lie in range: [0,1]

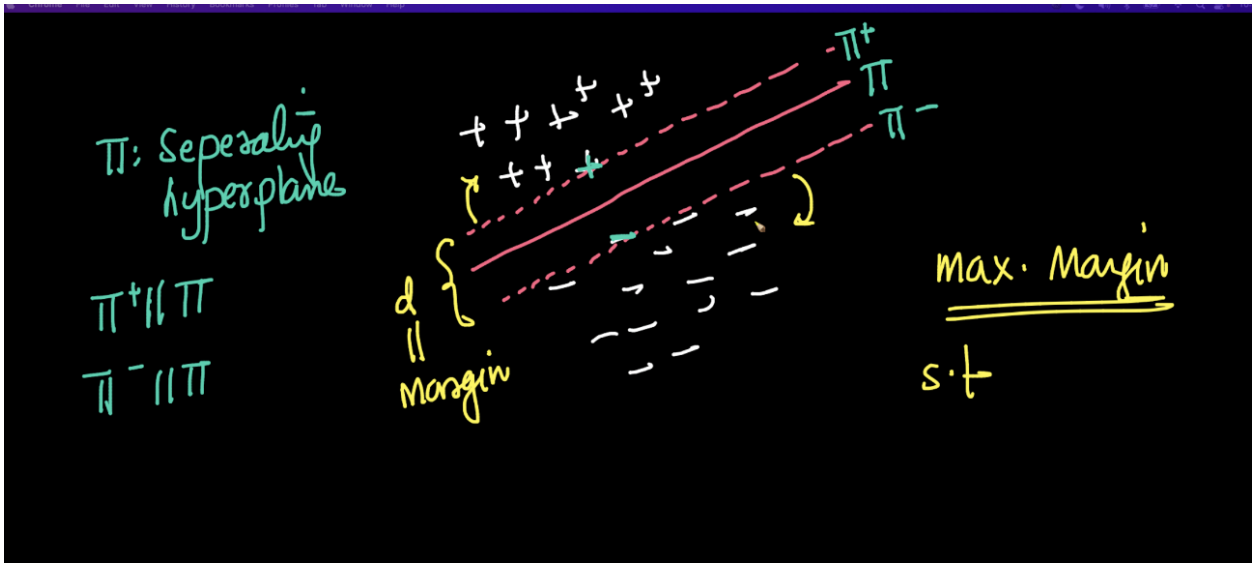
Q. How can we take care of this problem?

Can be resolved by taking logarithm:

$$log(P(y = k|w_1 \ldots w_d)) = log[P(y = k)] + \sum_{j=1}^{d} log[P(w_j|y = k)]$$

- **SVM**

1. **Q. What is the key idea behind SVM?**
    - The best hyperplane π: $w^T x + b = 0$ classifying between 2 classes is the one that has maximum gap/margin (d) between the itself and the closest +ve and -ve datapoints.



Q. How can we find the margin/gap?

The hyperplane parallel to π, that touches the closest +ve point: π^+: $w^T x + b = 1$;

The hyperplane parallel to π, that touches the closest -ve point: π^-: $w^T x + b = -1$

Margin is measured as the distance between them: $d(\pi^+, \pi^-) = \frac{2}{||w||}$;

- where w is the weight of the model.

Q. What optimization problem is SVM trying to solve?

Optimization problem: $max\ \frac{2}{||w||}$

The goal is to maximise generalisation.

2. **Q. What are Support Vectors?**

These are datapoints that:-

- Are within the margin
- Or, are misclassified
- Or, which lie on the hyperplanes (π^+, π^-)
- $\propto_i = 0$ for non support vectors, whereas, $\propto_i > 0$ for support vectors.

3. **Q. What are the different Types of SVM model?**
    A. **Hard Margin SVM:**
        - Simplest form of SVM model.
        - It assumes no datapoint can lie inside the Margin.
        - Rarely works in real life problems.

    B. **Soft Margin SVM:**
        - Introduces ζ as error for incorrectly placed datapoint.
            - $\zeta = 0$; datapoint is placed such that the hyperplane classifies the point correctly
            - $\zeta > 1$; datapoint is placed such that the hyperplane classifies the point incorrectly
            - $\zeta < 1$; datapoint is placed such that hyperplane still manages to classify the point correctly, but lies inside the margin.
        - Linear soft margin SVM is similar to LogReg

    C. **Kernel SVM:** Kernel SVM works on the dual of the SVM model such that it is a Similarity check between two datapoints x_1 and x_2 :

    $Kernel\ Function\ = k(x_1, x_2) = (x_1^t x_2 + c)^m$, where m is the degree of Polynomial

    Note:
        - This is just one example, other kernel functions also exist.
        - Modifies the dual form of SVM by replacing $x_i^T x_j$ with the similarity kernel function.

    Q. What is the kernel trick?
        - **Kernel Trick:** Kernel function projects a d-dim data to d'- dim data (where d'>>d) such that the data points are easily separable
        - Can use kernelisation with LogReg and Deep Learing models also.

    Q. What are the most popularly used kernels?
        - RBF / Gaussian kernel and
        - Quadratic kernel.

    Q. How is RBF kernel SVM different from KNN?

- RBF kernel SVM is similar to KNN geometrically, but differs in runtime complexity: **O(#SV * d),** where
  - #SV: Number of support vectors;
  - d: number of dimensions

4. **Q. How is the Loss Function calculated for SVM?**

   The Loss function consists of two components:

   - **Hinge Loss:** To have minimum ζ , such that $y\ (w^T x\ + b) \geq 1 - \zeta$

   $$\textbf{Hinge Loss: } \frac{1}{n}\sum_{i=1}^{n} \zeta_i$$

   - Max Margin: To have maximum Margin for generalizing on the data

   $$Max\ Margin\text{: } \frac{||w||}{2}$$

   Thus the total loss becomes:

   $$\textbf{Loss: } min_{(w,b)} \frac{||w||}{2} + C\frac{1}{n}\sum_{i=1}^{n} \zeta_i$$

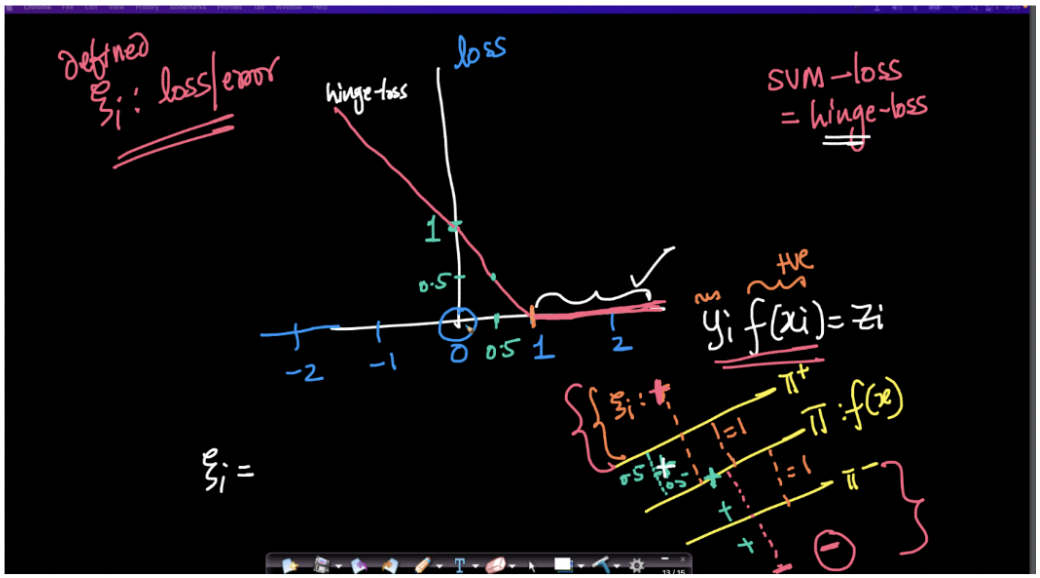
   Where

   - C is a hyperparameter which is analogous to Regularization parameter (λ) and -
   - $\frac{||w||}{2}$ is analogous to L2 Regularization.

   Note:-

   - This is the **Primal form** of SVM
   - Primal form works similar to Log Reg.

Q. What does the plot of hinge loss look like?



Q. What is the Dual form of loss in SVM?

We define a variable $\propto_i$ for each datapoint x_i, such that

- $0 <= \propto_i <= C$
- $\sum_{i=1}^{n} \propto_i y_i = 0$

Dual form of loss function: $max_{\propto_i} \sum_{i=1}^{n} \propto_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \propto_i \propto_j y_i y_j x_i^T x_j$

Note:

- Dual form aids in the kernel trick by doing implicitly transforming the datapoints to higher dimension

Q. How can we add constraints to the Loss function?

$min_{(w,b)} \frac{||w||}{2} + C \frac{1}{n} \sum_{i=1}^{n} \zeta_i + \lambda_1(Constraint\ 1) + \lambda_2(Constraint\ 2) + ... + \lambda_n(Constraint\ n)$

- where λ : Lagrange multiplier
- Adding constraints does not affect convexity of loss function.

5. **Q. How does the value of C affect the model?**

C is the hyperparameter. It causes a tradeoff between maximizing the margin and minimizing ζ

- If C is very large,
  - The SVM model tries to minimize HingeLoss
  - This makes the model have 0 incorrect predictions.
  - Thus making the model **overfit**.
- If C is very small,
  - The model tends to generalize the data
  - Hence, the model tends to have maximum ζ,
  - Thus making the model **underfit.**

6. **Q. What is the impact of Imbalanced Data on SVM?**

SVM is impacted if there is imbalance in Support Vectors. To resolve this:

- Either class weights should be used.
- Or rebalance the data.

7. **Q. What are some Limitations of the SVM model?**

- In some situations RBF kernel is very similar to KNN.
- In practice, GBDT/Random forest still beats SVM
- Time complexity to train SVM is very high: **$O(n^2)$**; n: no of datapoints
- Unlike Deep learning, SVM cannot create new features on its own.
- If we observe, we are just replacing feature engineering in GBDT/Random forest with kernel design in SVM.
- Even the RBF kernel SVM is impacted by outliers.

8. **Q. What is Support Vector Regressor (SVR)?**

   Though not very popularly used, SVM can be used for regression problems also.

   Loss function: $min_{(w,b)} \frac{1}{2}||w||^2 + C.\epsilon$, such that:

   - $y_i - \hat{y}_i <= \epsilon$
   - $\hat{y}_i - y_i <= \epsilon$
   - $\epsilon >= 0$

A Comparative study for each of the ML-1 model

Criteria	Linear Regression	Logistic Regression	SVM	kNN	Decision Trees	Random Forest
Regression	yes	no	yes	yes	yes	yes
Classification	no	yes	yes	yes	yes	yes
Binary/Multi-class	NA	Default: Binary Multi-class by O-vs-O, O-vs-R	Default: Binary Multi-class by O-vs-O, O-vs-R	Multi	Multi/Binary class	Multi/Binary class
Feature Data-type	Numerical, need to change catergorical variables to dummy variables	Numerical, need to change catergorical variables to dummy variables	Numerical, need to change catergorical variables to dummy variables	Numerical, need to change catergorical variables to dummy variables	Mixed	Mixed
Feature Scaling	Required	Required	Required	Required	Not required	Not required
Parametric	Parametric	Parametric	Parametric	Non-paramet	Non-parametric	Non-parametric

				ric		
Hyperparameters	Learning Rate, Regularization	Learning Rate, Regularization	C	K, Distance Metric	Max Depth, Min Samples Leaf, Min Samples Split	number of trees, Columnsampled, rowsample size, Depth of base learners
Training Metric	Least Squared Error	Log-Loss or Binary Cross-Entropy (derived from MLE)	Hinge Loss, looks for maximum margin classifier	NA (no training happens)	Gini Impurity for classification MSE for Regression	Depends on the Base Learners Gini Impurity, MSE or Entropy
Support for Non-linearity	Default: No Yes, by generating high order features	Default: No Yes, by generating high order features	Yes, automatic non-linear creation using non-linear kernel	Yes	Yes	Yes
Affect of Outliers/Noise	Sensitive	Sensitive	Less sensitive	Less sensitive, if k is big enough	Not sensitive to outliers	Not sensitive to outliers
Proneness to Overfitting	Less prone, can overfit in high dimension (curse of dimensionality)	Less prone, can overfit in high dimension (curse of dimensionality)	Relatively prone because its optmisation has by default the regularisation term	Less sensitive, if k is big enough, very senisitive if k is small	Highly prone to overfitting, can be controlled with pruning or hyperparameter tuning	Base learners are highly prone to overfitting, but the overall model is not due to aggregation
Multi-collinearity	Sensitive	Sensitive	Sensitive	"Sensitive" as well, as the additional collinear features will get extra "weightage"	Robust, problem may only come during for understanding feature importance	Not effected, due to Row and Column sampling
Advantages	Computationally scales very well with large numbers of data points and features Easy to	Computationally scales very well with large numbers of data points and features Easy to	Scales well with number of features	No training required, highly interpretable, Easy use for Multi-class	- No feature pre-processing - Can work with mixed data - Simple interpretation - Non-paramteric (like kNN), but	- No Feature Pre-processing - Performs better than Decision Trees - non-Parametric - Parallely runs

	interpret Less prone to overfitting	interpret Less prone to overfitting			faster in testing - Cannot be extrapolated like LR	model
Disadvantages	Tends to underfit By default does only linear regression, without considering non-linear relationships Very prone to outliers and multi-collinearity	Multi-class classification computationally expensive Very prone to outliers and multi-collinearity By default can only do linear classification Tends to underfit in practice	Doesn't scale well with lots of data - only small and medium sized datasets	- Its a metric algortihm - works poorly with high dimensional sparse data - Slow during testing	- Prone to overfitting and outliers - Unstable - small change in data can change tree	- Computationally expensive - needs a large dataset - Choosing the correct number of base learners is exhaustive
When to use	- less data (with less noise), many features - sparse high dimensional data	- less data (with less noise), many features - sparse high dimensional data	- less data, and less features - kernel SVM - less data, more features - linear SVM - Do not use it on data with more than 10K samples - images and sparse data	Never	- use if data has lots of categorical variables - don't use it when data is sparse - do not use if instances are too as compared to possible number of variatios features - images	- When simple models does not produce desire results - When data is large