ML Supervised Revision Notes

KNN

Why is KNN required ?

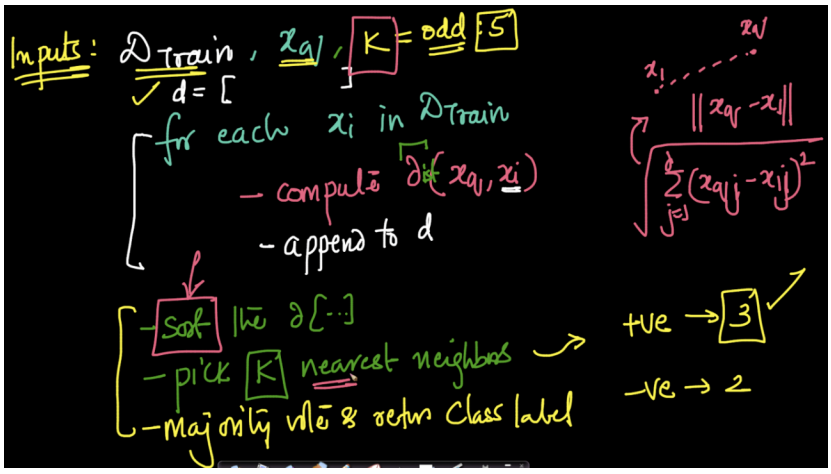
- Simple to use for multiclass classification
    - One v/s rest method would have to be employed for Logistic regression
- Non-linear data handling
    - Logistic regression needs creating polynomial features

What is the only assumption of KNN ?

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

What is the KNN Algorithm ?

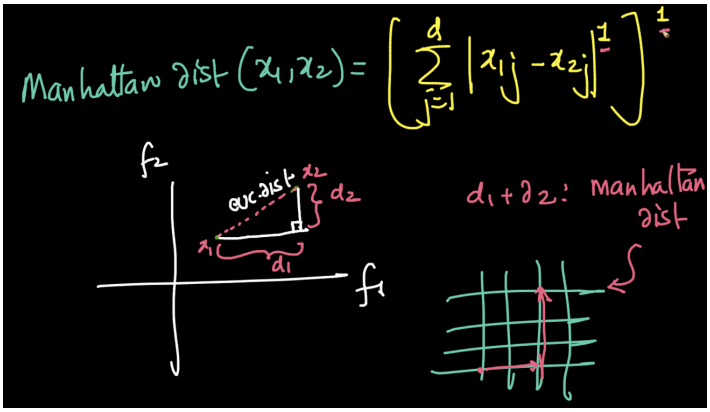
1. Training → No training required, just store the training data
2. Testing Prediction:
    a. Find out the distance of all the points from each point using a distance metric
    b. Filter out k-nearest neighbors of each point
    c. Assign majority class label to the point



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)

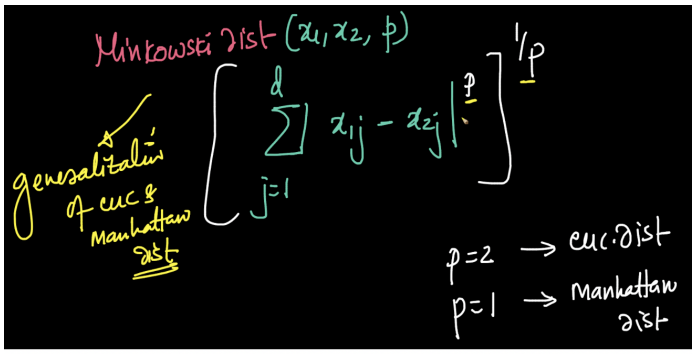
2. Euclidean Distance

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

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

3. Minkowski Distance
   - Generalized version for pth degree

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



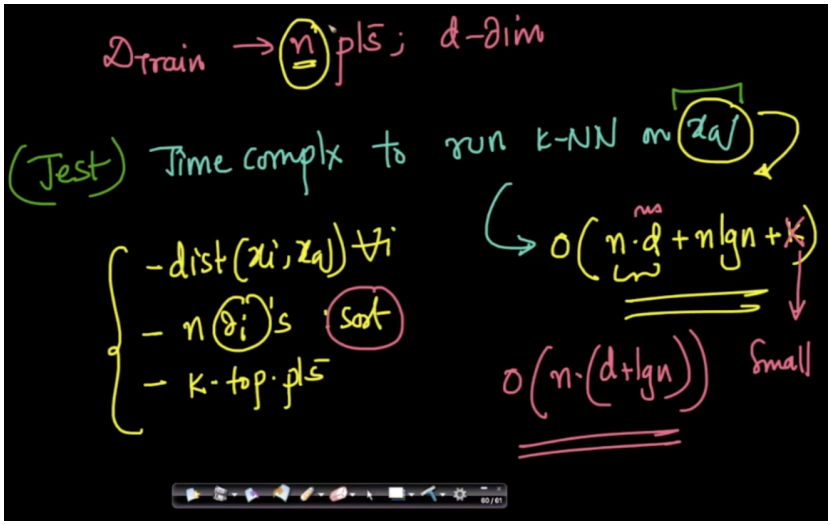
But should we give equal weightage to all the k neighbors ?

- 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)

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

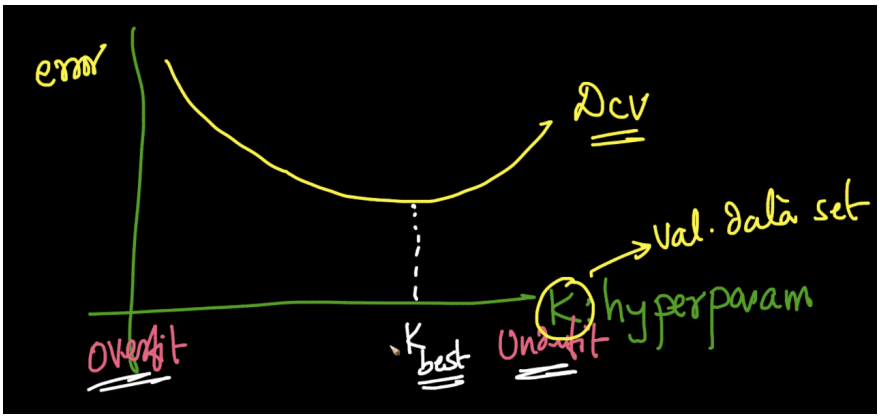
What is the train/test time complexity of KNN?

- 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 ?

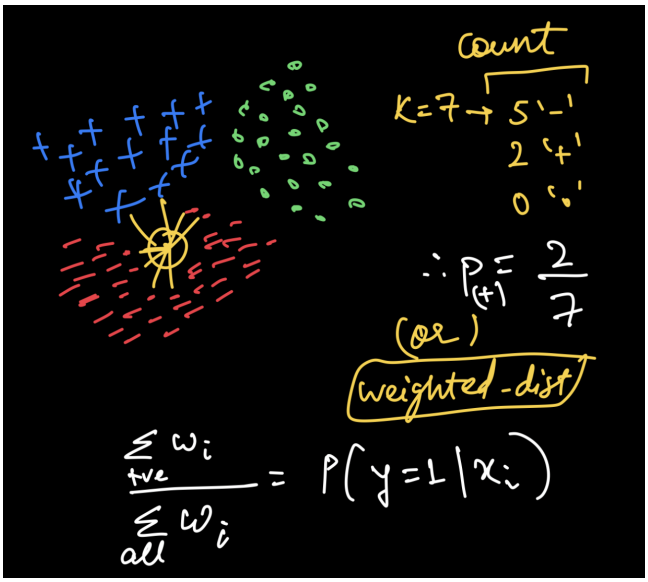
- High variance : Small K -> Starts fitting outliers/noise



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

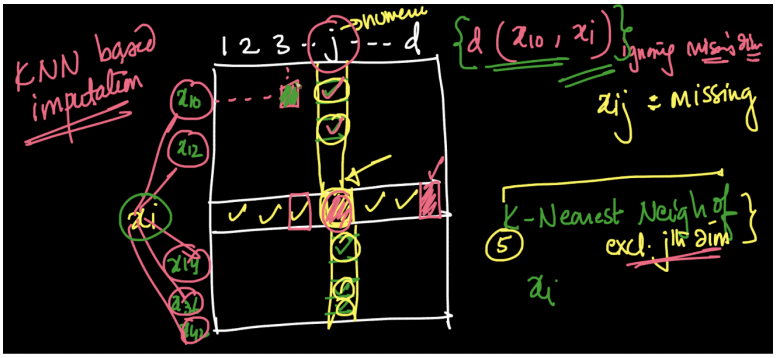
What is the Probabilistic Class label for KNN ?

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



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

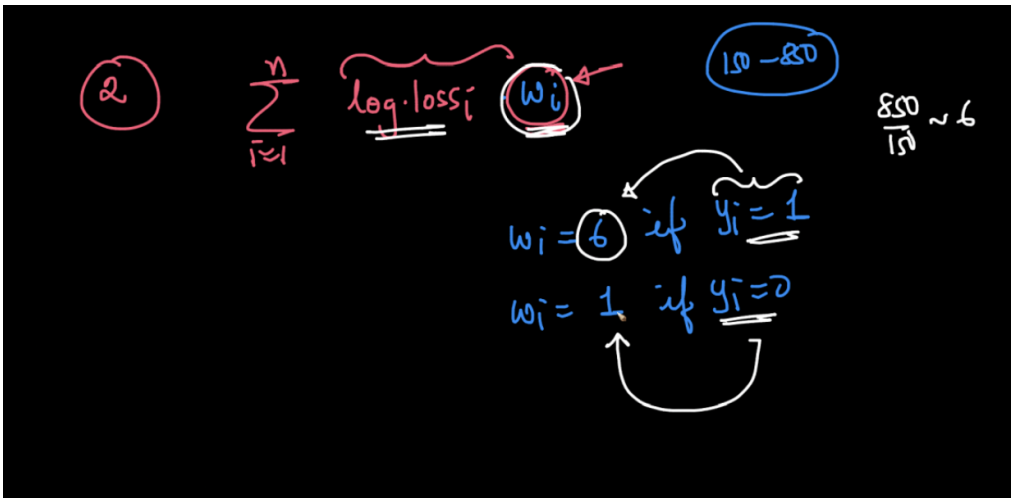
2. **Effect of Imbalance data on Logistic Regression:** Suppose if we have more -ve class samples than +ve class samples. Then the -ve class dominates the log loss function.

   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.

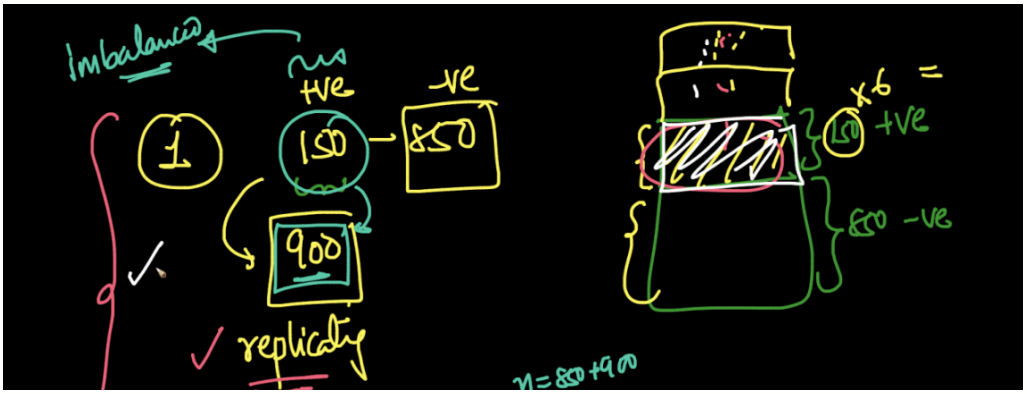
3. **Handling Imbalance Data**

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

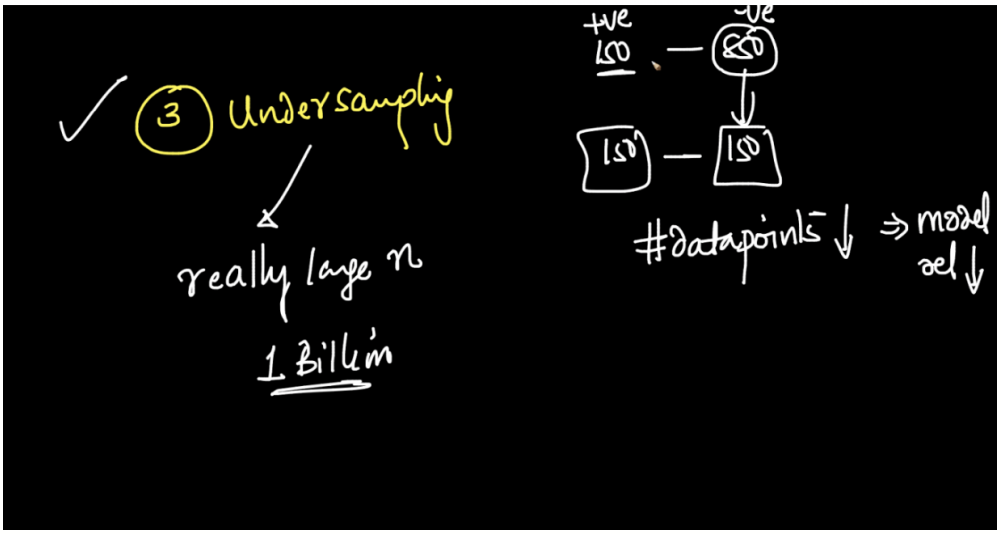
   $$w_{minority} = \frac{Number\ of\ samples\ of\ Majority}{Number\ of\ samples\ of\ Minority}\ ;\ w_{majority} = 1$$



   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.



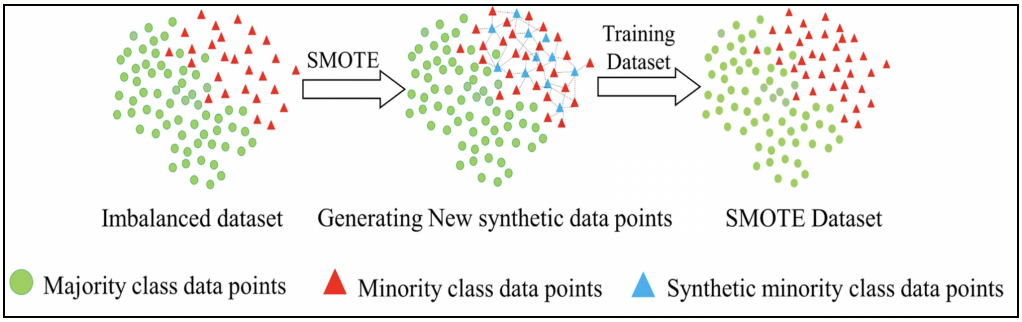
Note: Leads to Information loss that reduces model reliability . Hence only reliable if number of samples is really large

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.

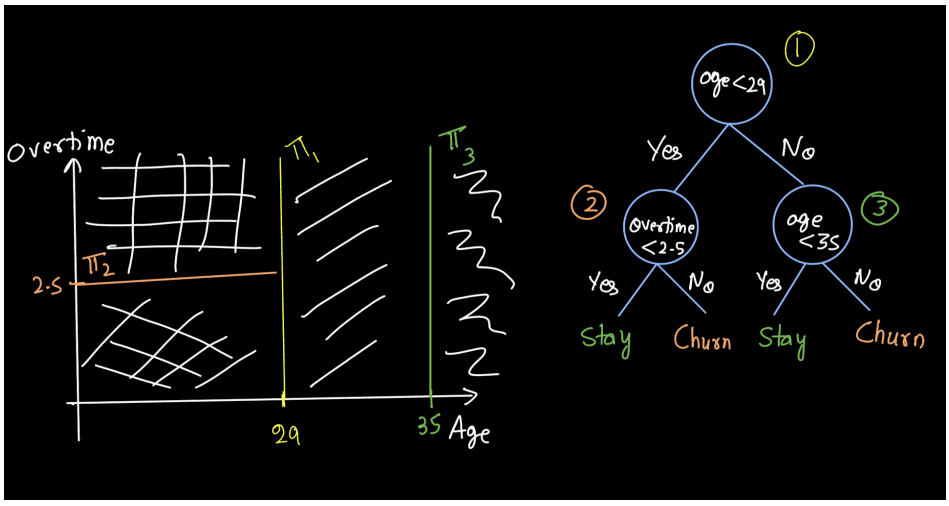
This creates a new sample datapoint x_{new}: $x_1 + [random\ value] \times d$

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

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

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

Entropy for binary classification

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

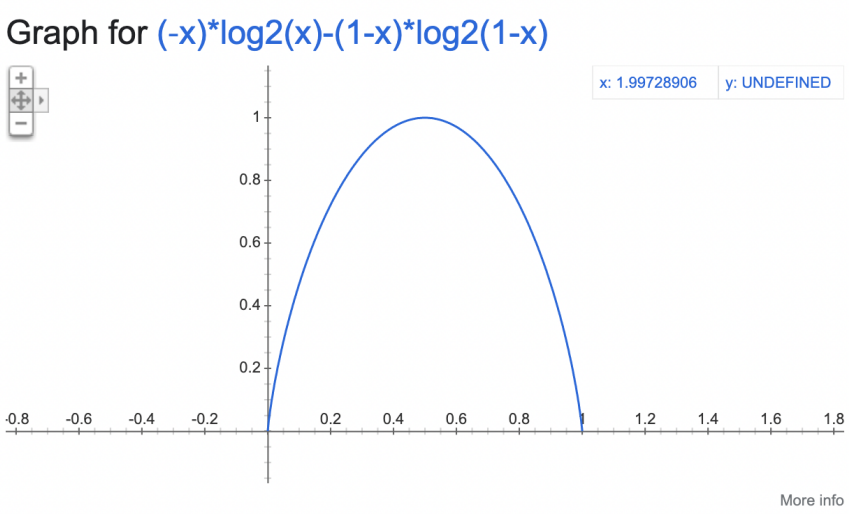
where

- 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:

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

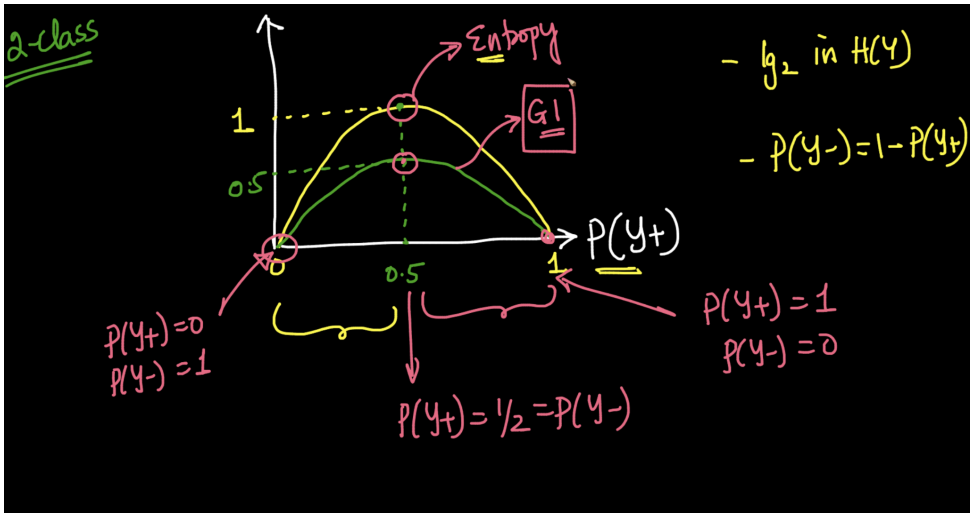
For a binary classification:

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

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

Calculating Feature Importance

Feature importance is calculated based on Information Gain.

- Feature with more information gain is considered more important.

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)

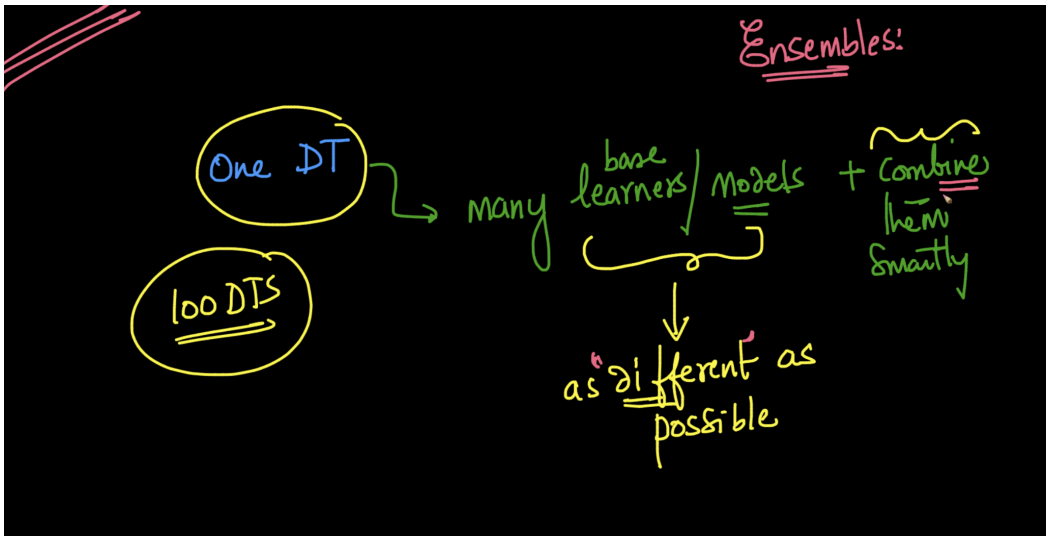
n: Nodes in the tree

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: By adjusting the maximum depth, you control how deep each tree in the ensemble can grow. Deeper trees may capture more complex relationships in the data but also risk overfitting.

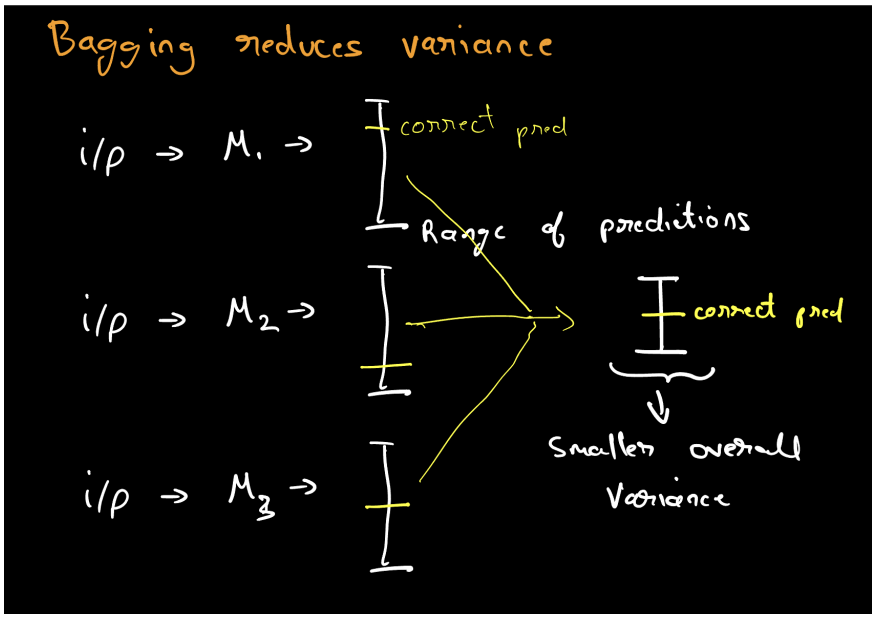
How to perform Validation in Random Forests ?

- 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 (Gini reduction and Information gain)
- 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. I.e. Each base learner can be built in parallel.
- 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,

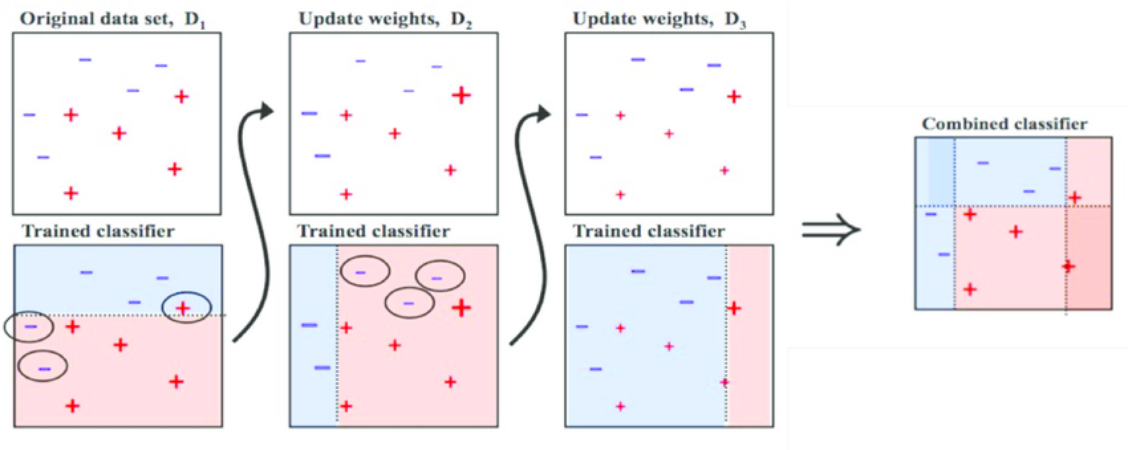
- we build a bunch of simple models and
- Each model is trained using the residual of the previous model.
- use these model to build an additive weighted model

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

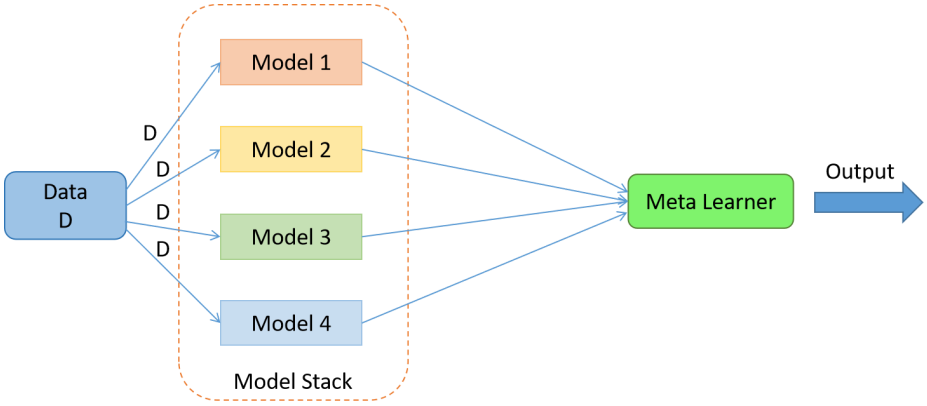
What is LightGBM ?

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

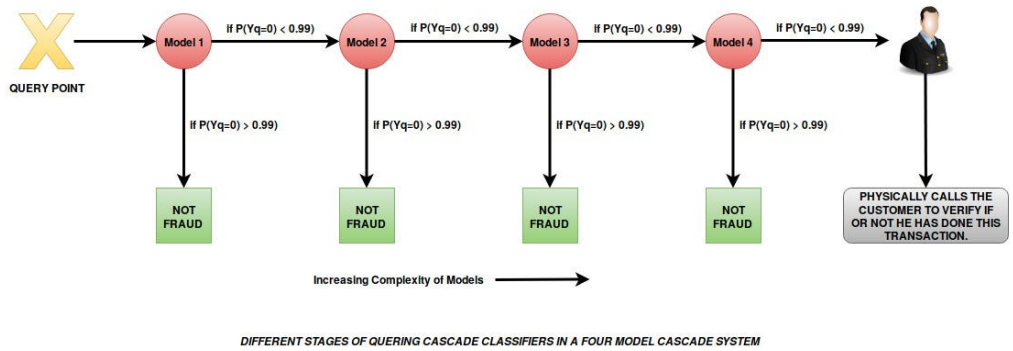
What is Stacking ?

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



DIFFERENT STAGES OF QUERING CASCADE CLASSIFIERS IN A FOUR MODEL CASCADE SYSTEM

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

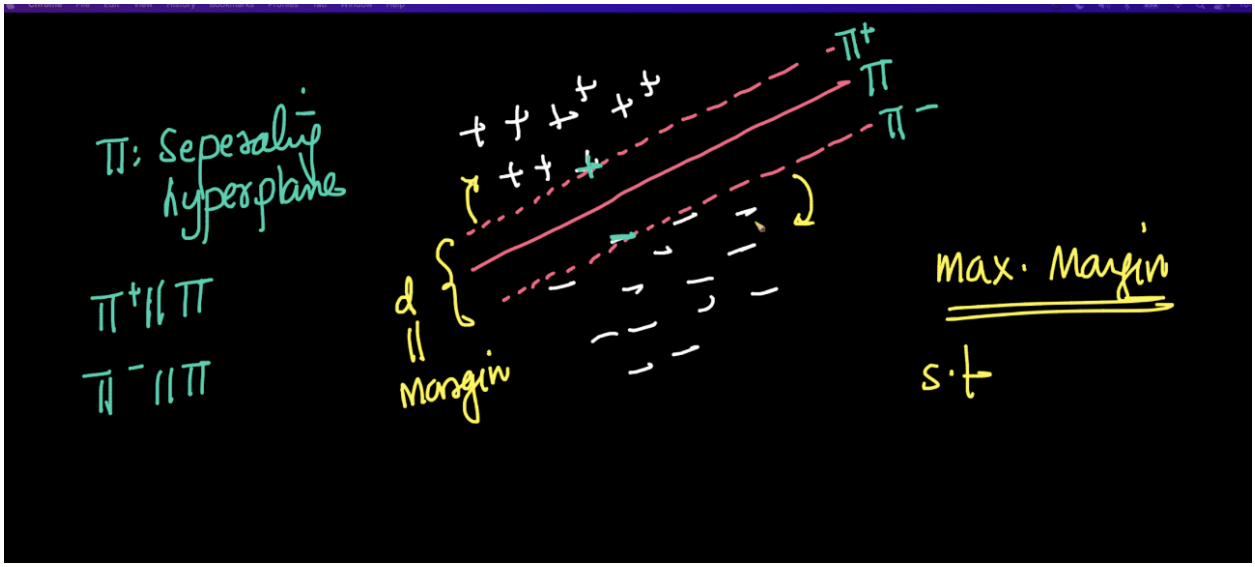
What about the explainability of the model?

- 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.

- **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 maximize generalization.

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 kernelization with LogReg and Deep Learning 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:

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

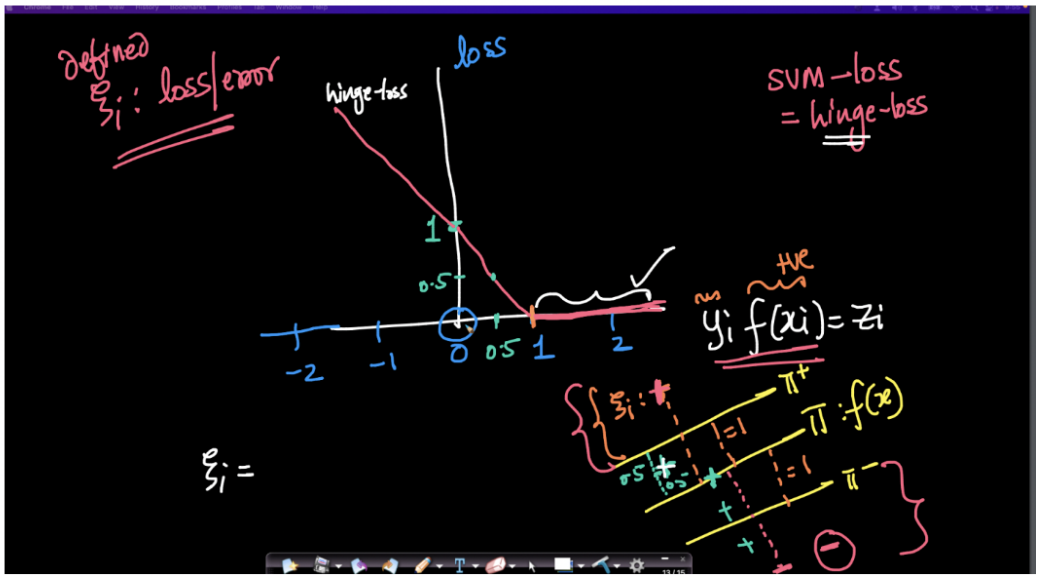
   Where

   - 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 data points 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 categorical variables to dummy variables	Numerical, need to change categorical variables to dummy variables	Numerical, need to change categorical variables to dummy variables	Numerical, need to change categorical variables to dummy variables	Mixed (sklearn only accept numerical features so need to encode categorical to numeric)	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, Column sample, row sample 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 optimization has by default the regularization term	Less sensitive, if k is big enough, very sensitive 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
Multicollinearity	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 interpret	Computationally scales very well with large numbers of data points and features Easy to interpret	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-parametric (like kNN), but faster in testing	- No Feature Pre-processing - Performs better than Decision Trees - non-Parametric - Parallely runs model

	Less prone to overfitting	Less prone to overfitting			- Cannot be extrapolated like LR	
Disadvantages	Tends to underfit By default does only linear regression, without considering non-linear relationships Very prone to outliers and multicollinearity	Multi-class classification computationally expensive Very prone to outliers and multicollinearity 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	- It's a metric algorithm - 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 variations features - images	- When simple models does not produce desired results - When data is large