Embedded AI

Dr.Akshita Chanchlani

Agenda

- Model Evaluation
- KNN
- SVM
- Ensemble Learning

Model Evaluation Regression

Regression Model Evaluation Metrics

- For evaluation of regression model, following metrics are used
  - MAE
  - MSE
  - RMSE
  - R2
  - Adjusted R2

Mean Absolute Error (MAE)

- The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction (absolute)
- It measures accuracy for continuous variables
- The MAE is the average over the verification sample of the absolute values of the differences between forecast and the corresponding observation $|y - \hat{y}|$
- The MAE is a linear score which means that all the individual differences are weighted equally in the average

$$\frac{\sum |y - \hat{y}|}{n}$$

Mean Squared Error (MSE)

- In statistics, the mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the error
- That is, the average squared difference between the estimated values and the actual value
- MSE is a risk function, corresponding to the expected value of the squared error loss
- The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate
- The MSE is a measure of the quality of an estimator
- As it is derived from the square of Euclidean distance, it is always a positive value with the error decreasing as the error approaches zero

$$\frac{\sum (y - \hat{y})^2}{n}$$

Root Mean Squared Error (RMSE)

- RMSE is the most popular evaluation metric used in regression problems
- It follows an assumption that error are unbiased and follow a normal distribution
- Here are the key points to consider on RMSE:
  - The power of ‘square root’ empowers this metric to show large number deviations
  - The ‘squared’ nature of this metric helps to deliver more robust results which prevents cancelling the positive and negative error values
- It avoids the use of absolute error values which is highly undesirable in mathematical calculations
- When we have more samples, reconstructing the error distribution using RMSE is considered to be more reliable
- RMSE is highly affected by outlier values. Hence, make sure you’ve removed outliers from your data set prior to using this metric.
- As compared to mean absolute error, RMSE gives higher weightage and punishes large errors

R-Squared (R2)

- We learned that when the RMSE decreases, the model's performance will improve
- But these values alone are not intuitive
- When we talk about the RMSE metrics, we do not have a benchmark to compare
- This is where we can use R-Squared metric
- In other words how good our regression model as compared to a very simple model that just predicts the mean value of target from the train set as predictions

Adjusted R-Squared

- A model performing equal to baseline would give R-Squared as 0
- Better the model, higher the r2 value
- The best model with all correct predictions would give R-Squared as 1
- However, on adding new features to the model, the R-Squared value either increases or remains the same
- R-Squared does not penalize for adding features that add no value to the model
- So an improved version over the R-Squared is the adjusted R-Squared

$$\bar{R}^2 = 1 - \left(1 - R^2\right)\left[\frac{n-1}{n-(k+1)}\right]$$

- k: number of features
- n: number of samples

Model Evaluation Classification

Classification Model Evaluation Metrics

- For evaluation of classification model, following metrics are used
  - Confusion Matrix
  - F1 Score
  - Auc-Roc

Confusion Matrix

- A confusion matrix is an N X N matrix, where N is the number of classes being predicted
- The confusion matrix provides more insight into not only the performance of a predictive model, but also which classes are being predicted correctly, which incorrectly, and what type of errors are being made

Observed	Predicted	
1	1	TP
1	0	FN
0	0	TN
0	1	FP

		Predicted condition	
	Total population = P + N	Predicted condition **positive** (PP)	Predicted condition **negative** (PN)
Actual condition	Actual condition positive (P)	**True positive** (TP), hit	**False negative** (FN), Type II error, miss, underestimation
	Actual condition negative (N)	**False positive** (FP), Type I error, false alarm, overestimation	**True negative** (TN), correct rejection

TP vs FP vs TN vs FN









Cat | Cat | Cat | Cat | No Cat | No Cat | No Cat | Cat | Cat

TP vs FP vs TN vs FN









FP	TP	FP	TP	TN	FN	TN	TP	TP
Cat	Cat	Cat	Cat	No Cat	No Cat	No Cat	Cat	Cat

TP = 4

FP = 2

Total P = 6

TN = 2

FN = 1

Total N = 3

Total = 9

Accuracy









Cat | Cat | Cat | Cat | No Cat | No Cat | No Cat | Cat | Cat

How many we got right ?

Accuracy : How many we got right?









FP TP FP TP TN FN TN TP TP

Cat Cat Cat Cat No Cat No Cat No Cat Cat Cat

Correct = TP + TN / Total

= 6 / 9

= 2/3

= 0.66

Precision

- Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive
- Precision is a good measure to determine, when the costs of False Positive is high
- For instance, in email spam detection, a false positive means that an email that is non-spam (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

Precision









Cat | Cat | Cat | Cat | No Cat | No Cat | No Cat | Cat | Cat

Out of all Cat predictions how many we got right ?

Precision : Out of all Cat predictions how many we got right ?









FP	TP	FP	TP	TN	FN	TN	TP	TP
Cat	Cat	Cat	Cat	No Cat	No Cat	No Cat	Cat	Cat

True positive = 4
Total positive = 6
Precision of + ve = 4/6 = 2/3= 0.66

True Negative = 2
Total Negative = 3
Precision of -ve = 2/3 = 0.66

Recall

- Recall actually calculates how many of the Actual Positives our model capture through labelling it as Positive (True Positive)
- Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative
- For instance, in fraud detection or sick patient detection, if a fraudulent transaction (Actual Positive) is predicted as non-fraudulent (Predicted Negative), the consequence can be very bad for the bank
- Similarly, in sick patient detection, if a sick patient (Actual Positive) goes through the test and predicted as not sick (Predicted Negative), the cost associated with False Negative will be extremely high if the sickness is contagious

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

Recall









Cat | Cat | Cat | Cat | No Cat | No Cat | No Cat | Cat | Cat

Out of all Cat truth how many we got right ?

Recall : Out of all Cat truth how many we got right ?









FP	TP	FP	TP	TN	FN	TN	TP	TP
Cat	Cat	Cat	Cat	No Cat	No Cat	No Cat	Cat	Cat

True positive = 4
Total Actual positive = 5
Recall of + ve = 4/5 = 0.80

True Negative = 2
Total Actual Negative = 4
Recall of -ve = 2/4 = 0.50

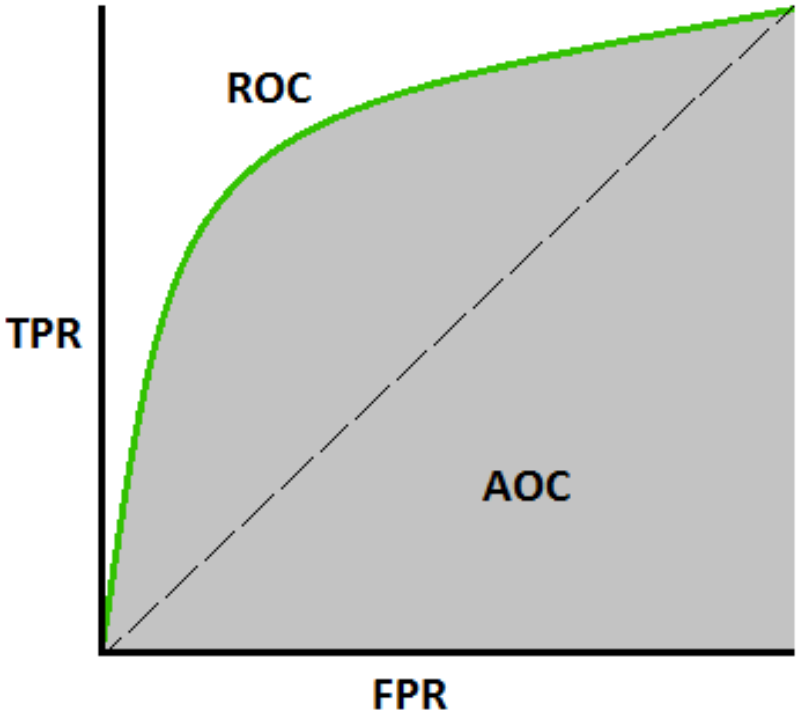
F1 Score

- The F1 score is the harmonic mean of the precision and recall
- The highest possible value of an F-score is 1.0, indicating perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero
- The F1 score is also known as the Sørensen–Dice coefficient or Dice similarity coefficient (DSC)

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Receiver Operating Characteristic (ROC)

- ROC curve is a metric that assesses the model ability to distinguish between binary classes
- It is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings
- The TPR is also known as sensitivity, recall or probability of detection in machine learning
- The FPR is also known as the probability of false alarm and can be calculated as 1– specificity
- Points above the diagonal line represent good classification (better than random)
- The model performance improves if it becomes skewed towards the upper left corner



Receiver Operating Characteristic (ROC)

TPR (True Positive Rate) / Recall /Sensitivity

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

Image 3

Specificity

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Image 4

FPR

$$\text{FPR} = 1 - \text{Specificity}$$

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

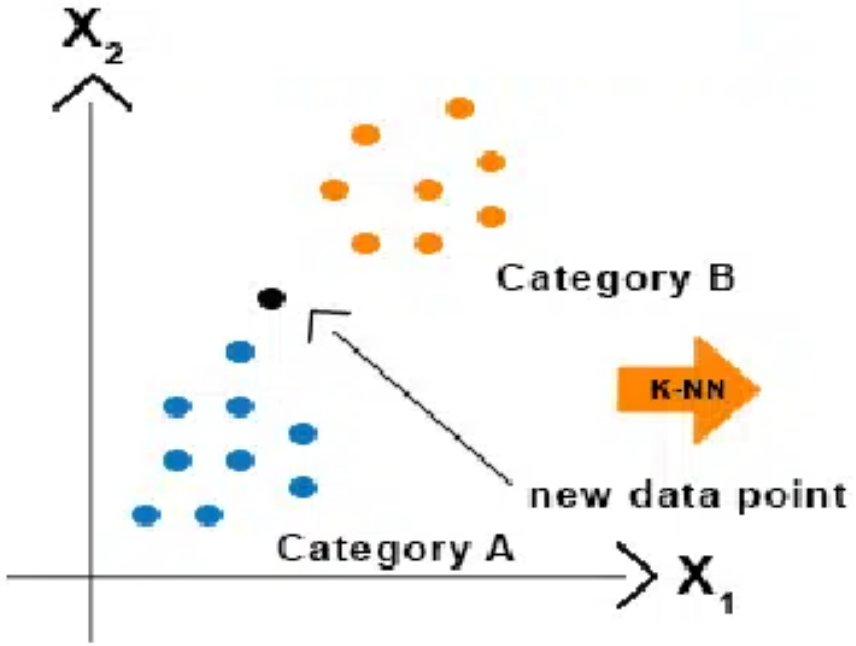
K-Nearest Neighbours

Overview

- The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems
- However, it is more widely used in classification problems in the industry
- It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection
- The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

How does it work?

Initially there are two categories
New data point value is to be predicted



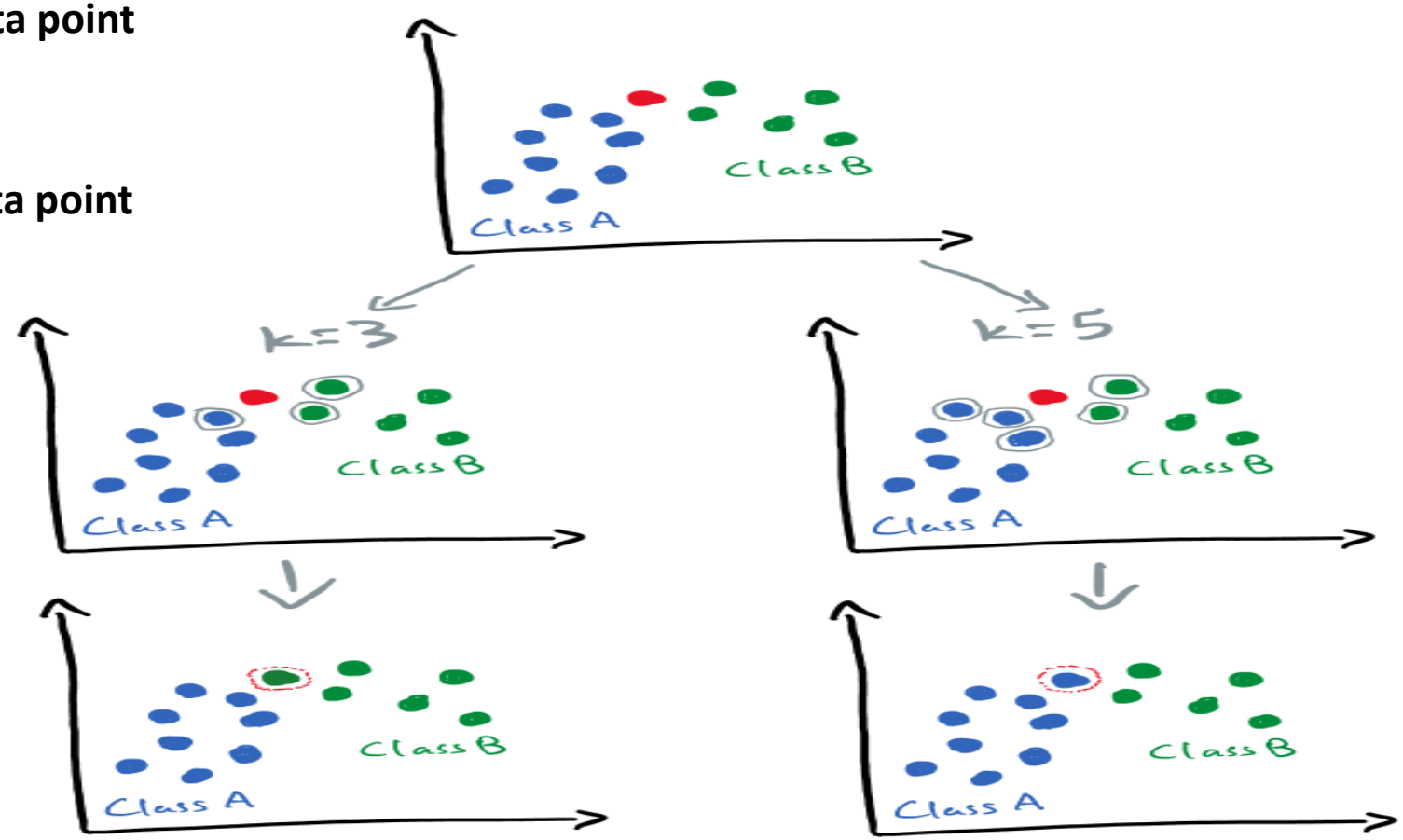
Distance from New data point to category A nearest value is calculated
Distance from New data point to category B nearest value is calculated

so according to that new data point assigned to category A

How does it work?

K **= 3**
We have to find the three closest data points
(three nearest neighbors) to the new (red) data point

K **= 5**
We have to find the five closest data points
(three nearest neighbors) to the new (red) data point



How does it work ?

- A case is classified by a majority vote of its neighbours, with the case being assigned to the class most common amongst its K nearest neighbours measured by a distance function
- If K = 1, then the case is simply assigned to the class of its nearest neighbour.
- Note: all three distance measures are only valid for continuous variables.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

How does it work ?

- Choosing the optimal value for K is best done by first inspecting the data
- In general, a large K value is more precise as it reduces the overall noise but there is no guarantee
- Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value
- Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

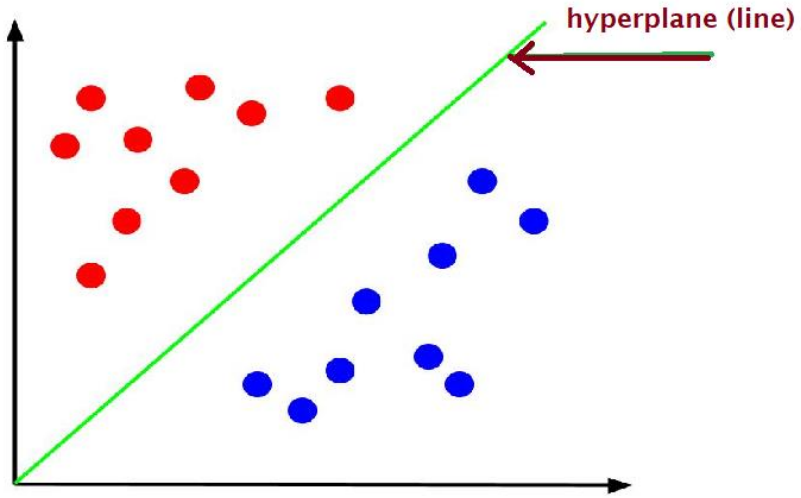
Applications of KNN

- Recommender system
- Relevant document classification
- OCR

Support Vector Machine (SVM)

Overview

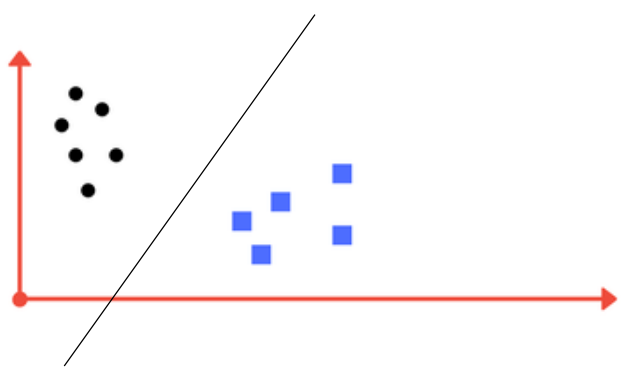
- It is a supervised machine learning algorithm that can be used for both classification and regression
- However, it is mostly used in **classification problems**
- The objective of the support vector machine algorithm is to find a **hyperplane** in an N-dimensional space(N — the number of features) that distinctly classifies the data points

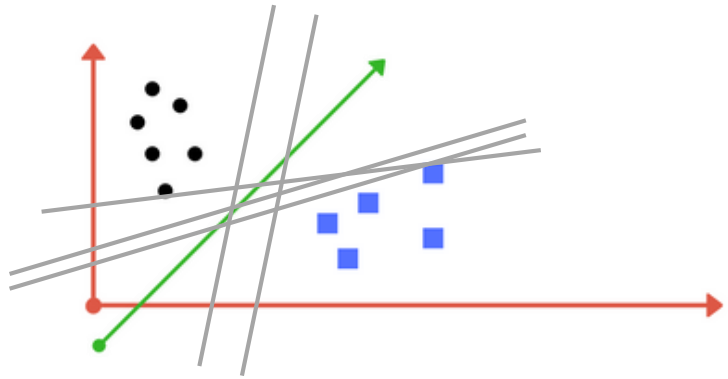


How does it work ?

- SVM separates the classes using hyperplane
- In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side
- To separate the two classes of data points, there are many possible hyperplanes that could be chosen
- Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes

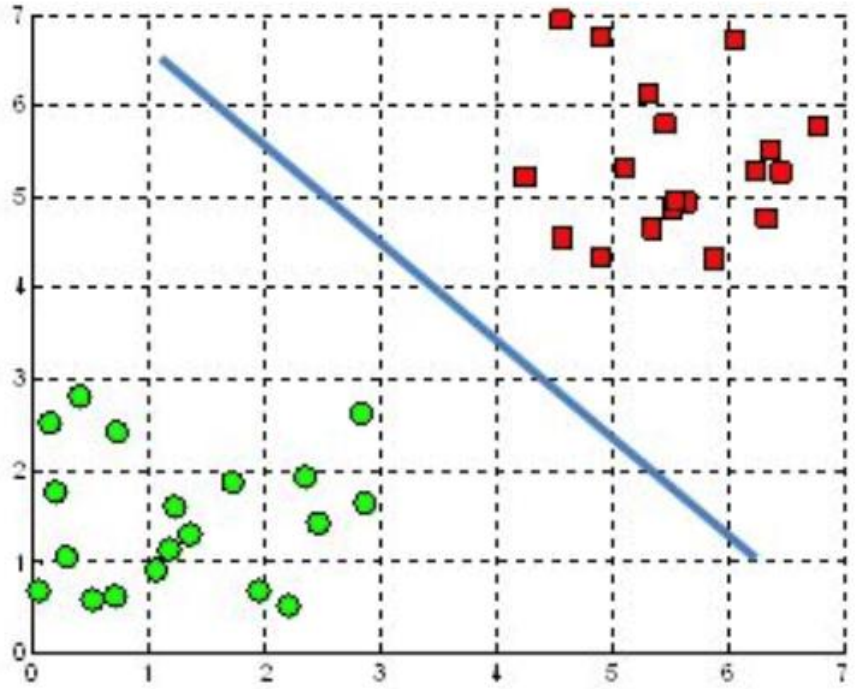
If hyperplane is chosen in this way



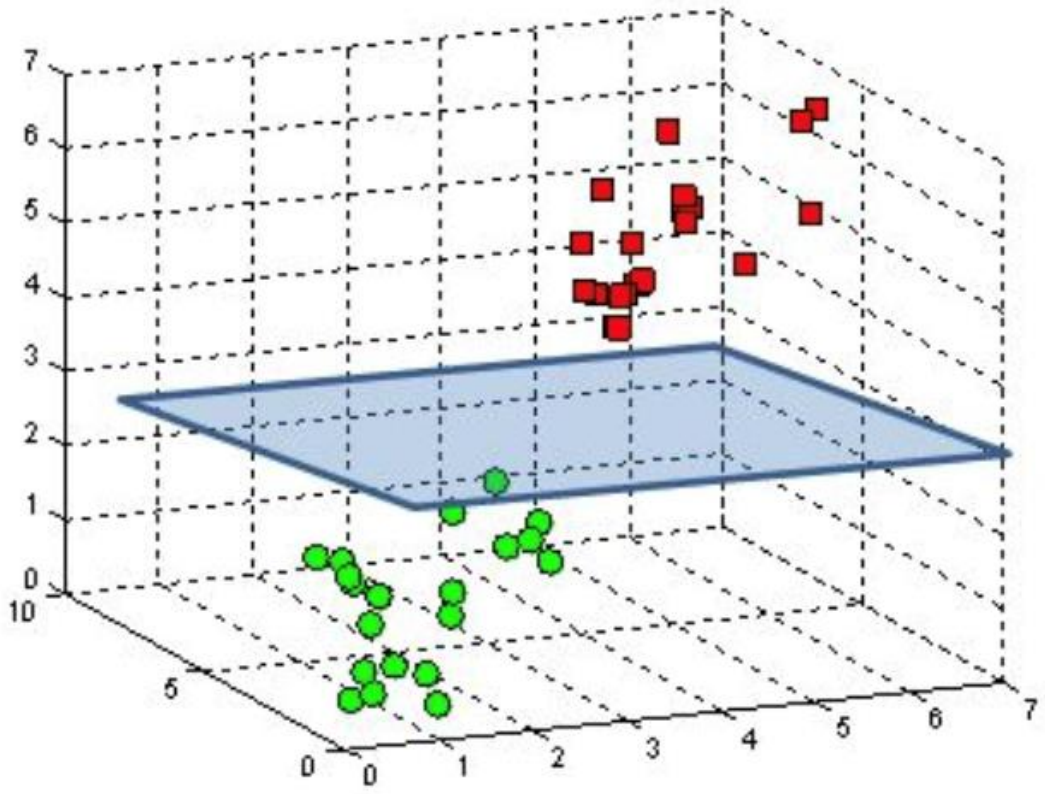


Hyperplane

A hyperplane in $\mathbb{R}^2$ is a line

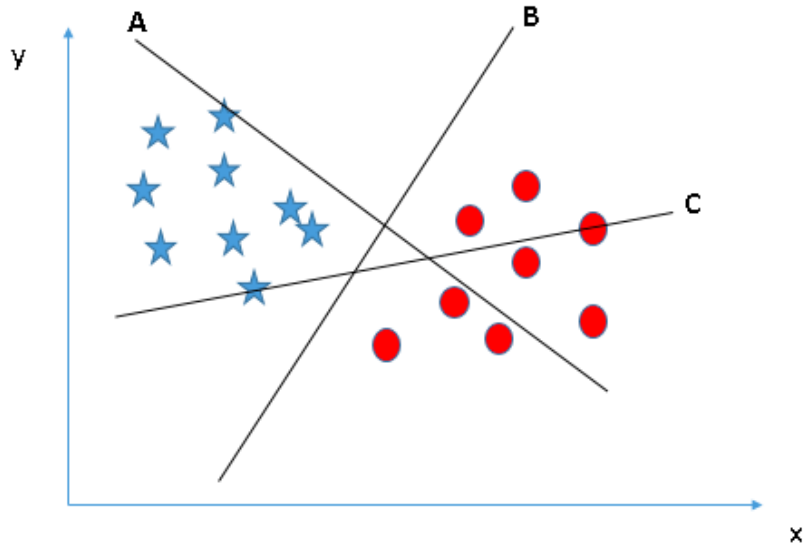


A hyperplane in $\mathbb{R}^3$ is a plane



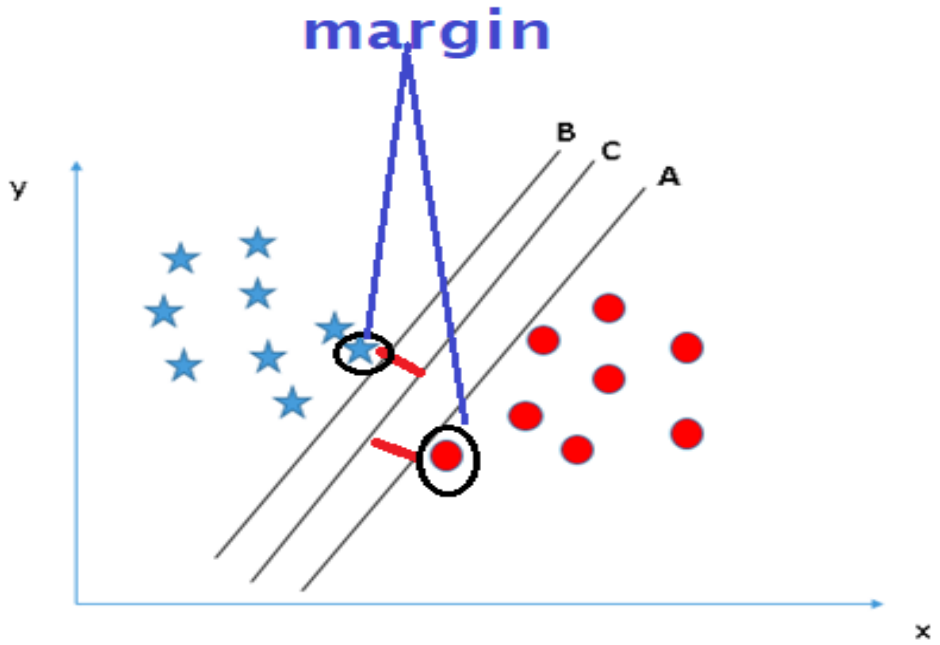
Scenario 1

- Here, we have three hyper-planes (A, B and C)
- Now, identify the right hyper-plane to classify star and circle
- You need to remember a thumb rule to identify the right hyper-plane
  - **Select the hyper-plane which segregates the two classes better (No mis-classification)**
- In this scenario, **hyper-plane "B" has excellently performed this job**



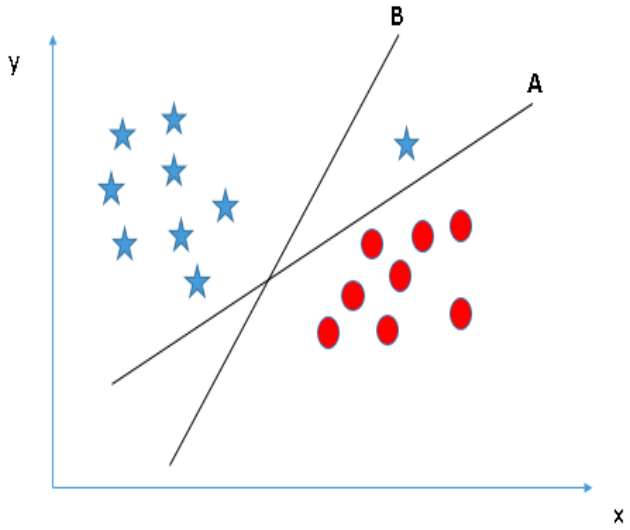
Scenario 2

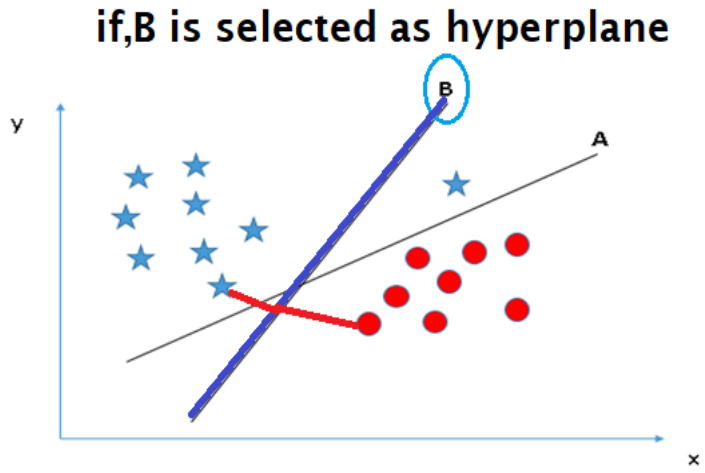
- Here, we have three hyper-planes (A, B and C) and all are segregating the classes well
- Now, How can we identify the right hyper-plane?
- Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin.**

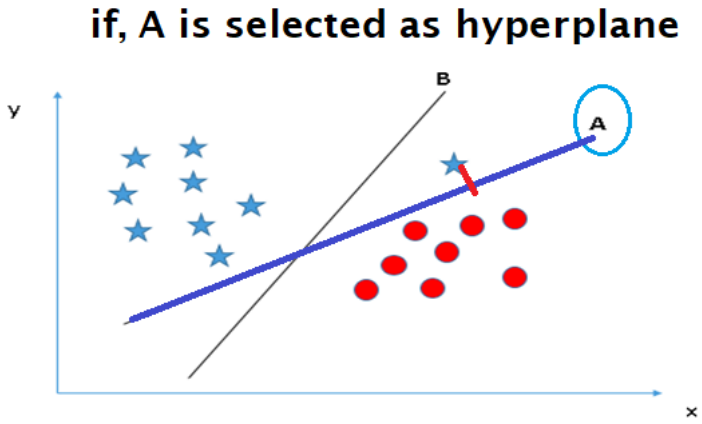


Scenario 3

- Use the rules as discussed in previous section to identify the right hyper-plane
- Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.**
- But, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin
- Here, hyper-plane B has a classification error and A has classified all correctly
- Therefore, the right hyper-plane is **A.**

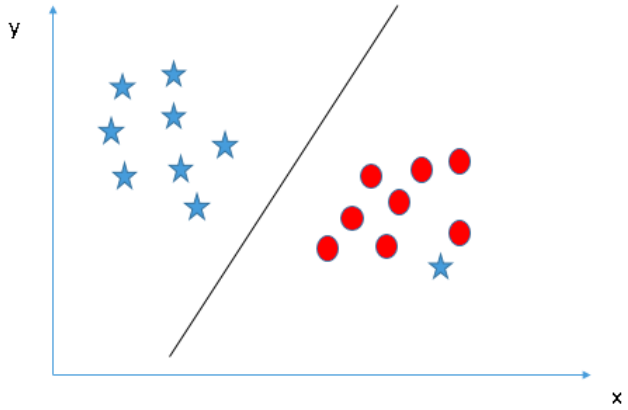


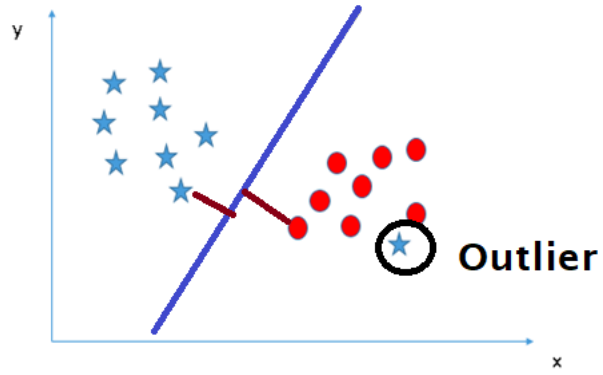




Scenario 4

- Unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier
- SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin
- Hence, we can say, SVM is robust to **outliers.**



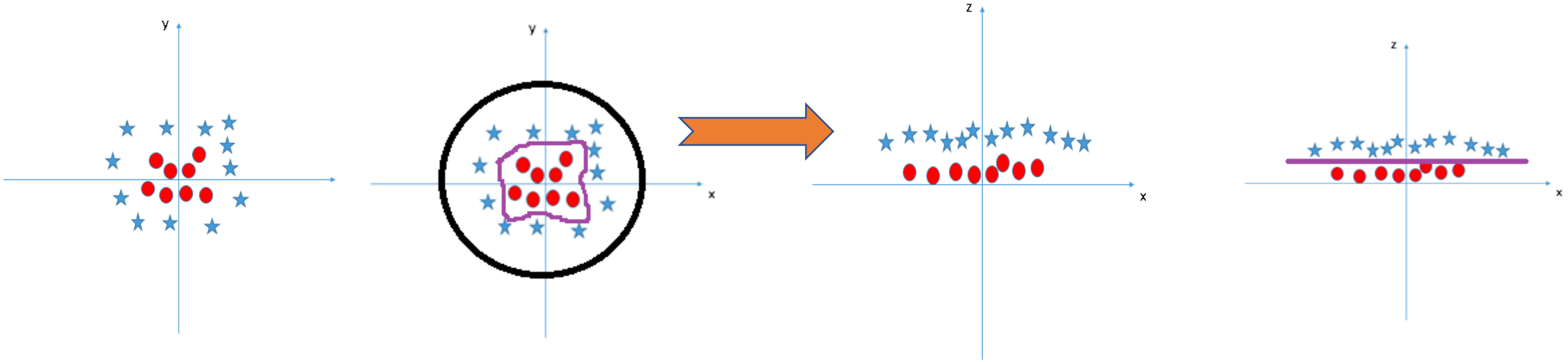


Scenario 5

- In the scenario below, we can't have linear hyper-plane between the two classes

- SVM solves this problem by introducing **kernel**

$$z=x^2+y^2$$

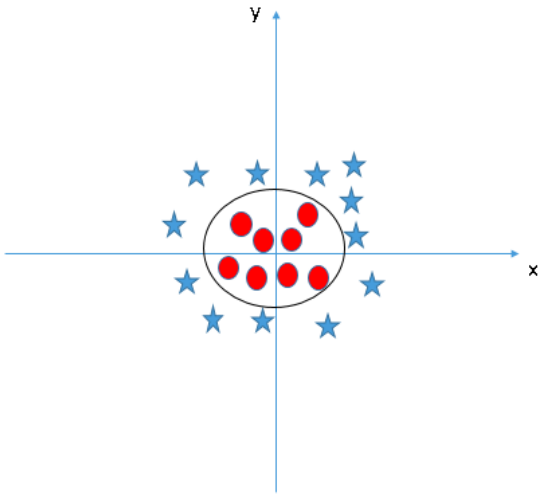


Scenario 5

- Points to consider are:
  - All values for z would be positive always because z is the squared sum of both x and y
  - In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.
- In SVM, it is easy to have a linear hyper-plane between these two classes, but for such scenarios, SVM uses a trick called as **Kernel**.
- These are functions which takes low dimensional input space and transform it to a higher dimensional space, i.e. it converts non separable problem to separable problem
- It is mostly useful in non-linear separation problem
- Simply put, it does some extremely complex data transformations,

  then find out the process to separate the data based on the

  labels or outputs you've defined.



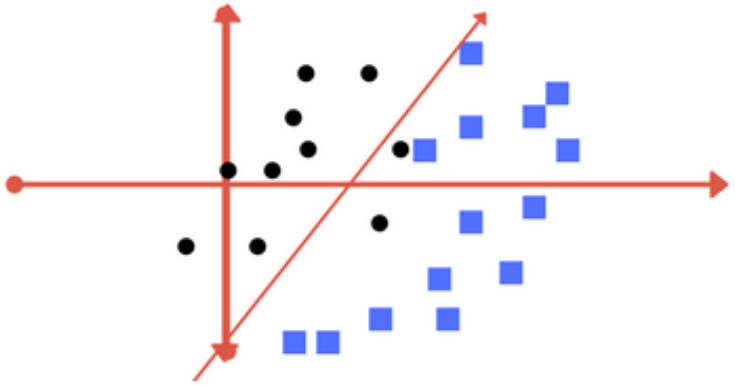
Tuning Parameters - Kernels

- The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

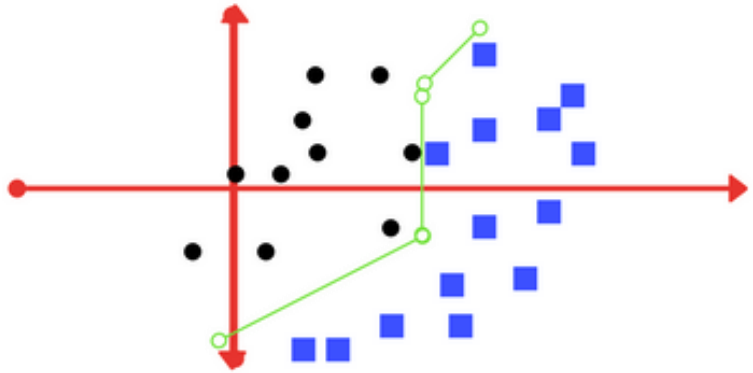
Tuning Parameters - Regularization

- The Regularization parameter tells the SVM optimization how much you want to avoid misclassifying each training example
- For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly
- Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points

Low regularization

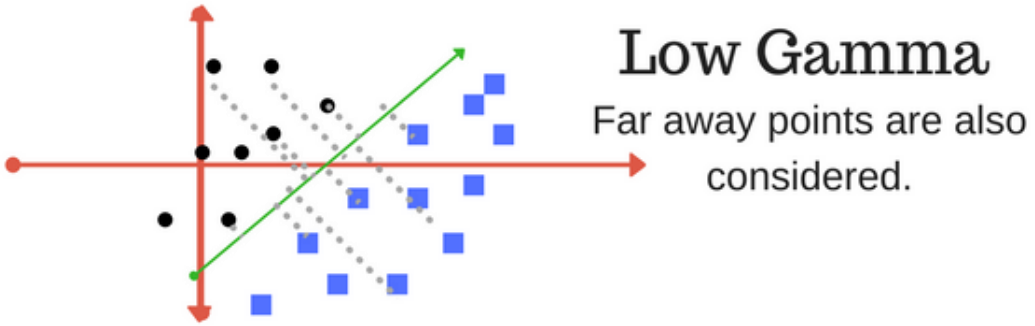


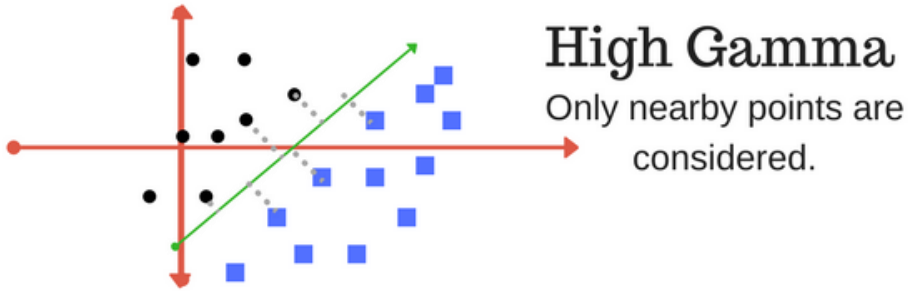
High regularization



Tuning Parameters - Gamma

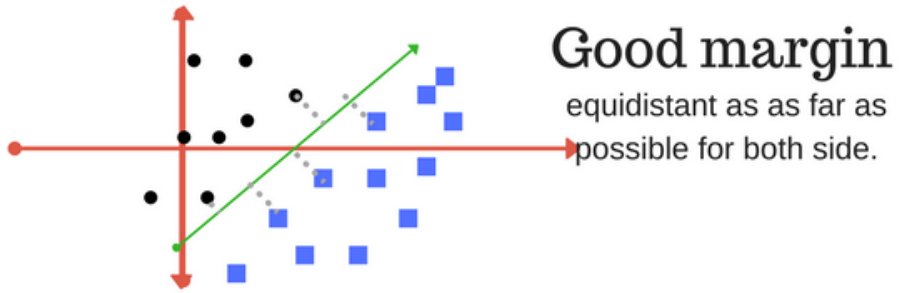
- The gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’
- In other words
  - With low gamma, points far away from plausible separation line are considered in calculation for the separation line
  - Where as high gamma means the points close to plausible line are considered in calculation.





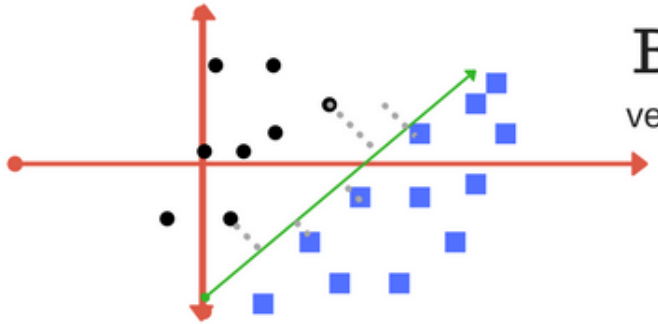
Tuning Parameters - Margin

- A margin is a separation of line to the closest class points
- A good margin is one where this separation is larger for both the classes



Good margin

equidistant as as far as possible for both side.



Bad margin

very close to blue class.

Ensemble Learning

Overview

- Ensemble is the art of combining diverse set of learners (individual models like decision tree, logistic regression, knn) together to improvise on the stability and predictive power of the model.
- The ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions.
- Primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection of a poor one
- The ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions.
- In learning models, noise, variance, and bias are the major sources of error. The ensemble methods in machine learning help minimize these error-causing factors, thereby ensuring the accuracy and stability of machine learning (ML) algorithms.
- Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, nonstationary learning and error-correcting

Basic Ensemble Methods

Mode

- "mode" is the number or value that most often appears in a dataset of numbers or values.
- In this ensemble technique, machine learning professionals use a number of models for making predictions about each data point.
- The predictions made by different models are taken as separate votes.
- Subsequently, the prediction made by most models is treated as the ultimate prediction.

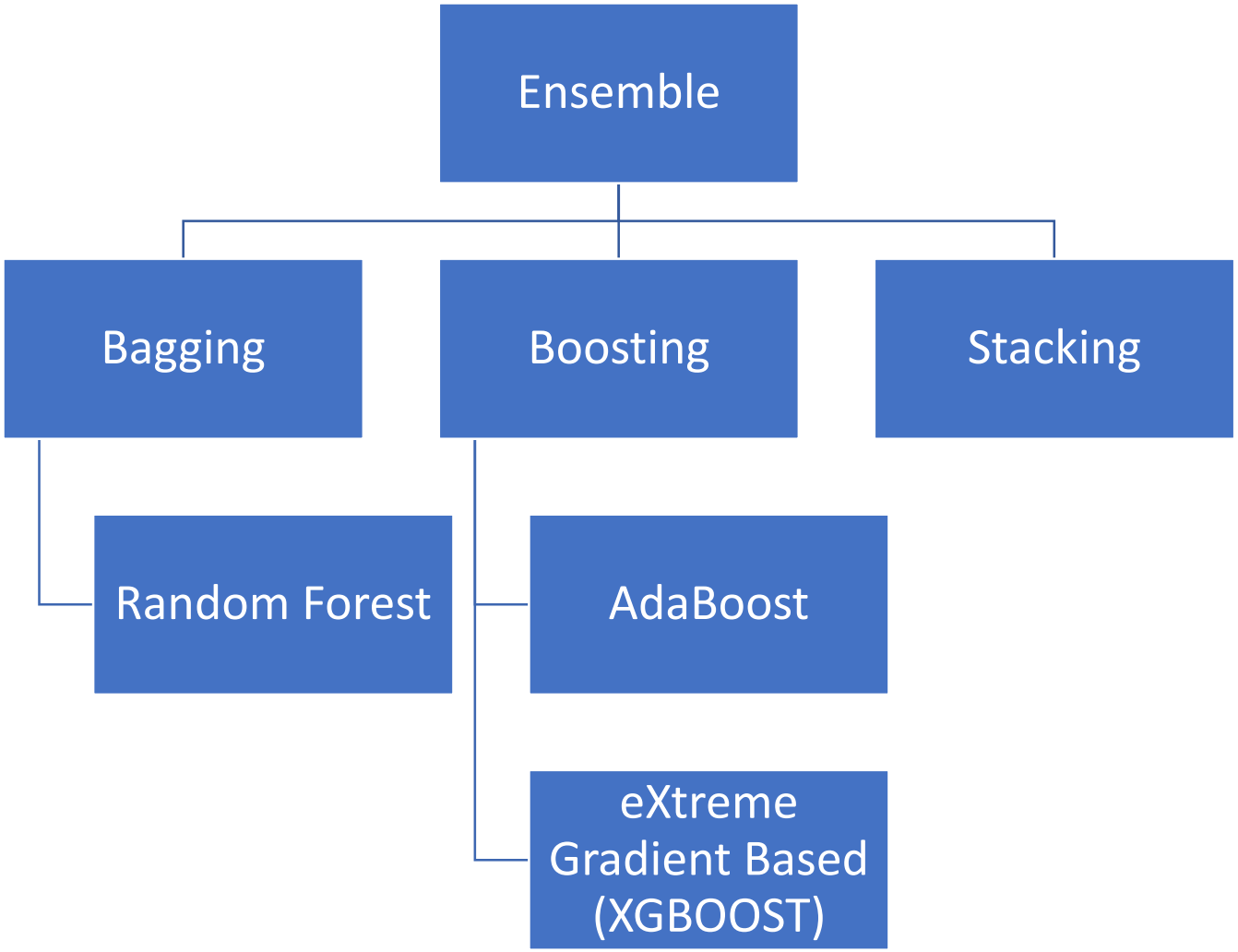
Mean/Average

- ensemble technique, data analysts take the average predictions made by all models into account when making the ultimate prediction.

The Weighted Average

- data scientists assign different weights to all the models in order to make a prediction, where the assigned weight defines the relevance of each model.

Advance Ensemble Learning Methods

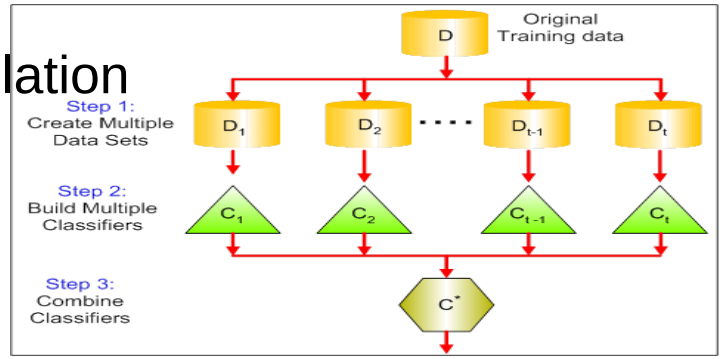


Three main classes of Ensemble Learning

- Bagging((Bootstrap Aggregating) involves fitting many decision trees on different samples of the same dataset and averaging the predictions.
- Stacking involves fitting many different models types on the same data and using another model to learn how to best combine the predictions. An iterative ensemble technique, "boosting," adjusts an observation's weight based on its last classification.
- Boosting involves adding ensemble members sequentially that correct the predictions made by prior models and outputs a weighted average of the predictions.

Bagging

- Bagging tries to implement similar learners/ same algorithm on small sample populations and then takes a mean of all the predictions
- The primary goal of "bagging" or "bootstrap aggregating" ensemble method is to minimize variance errors in decision trees.
- The objective here is to randomly create samples of training datasets with replacement (subsets of the training data).
- The subsets are then used for training decision trees or models.
- Consequently, there is a combination of multiple models, which reduces variance, as the average prediction generated from different models is much more reliable and robust than a single model or a decision tree.
- In generalized bagging, you can use different learners on different population
- This helps us to reduce the variance error
- Algorithm
  - Random Forest (decision tree)



Boosting

Boosting

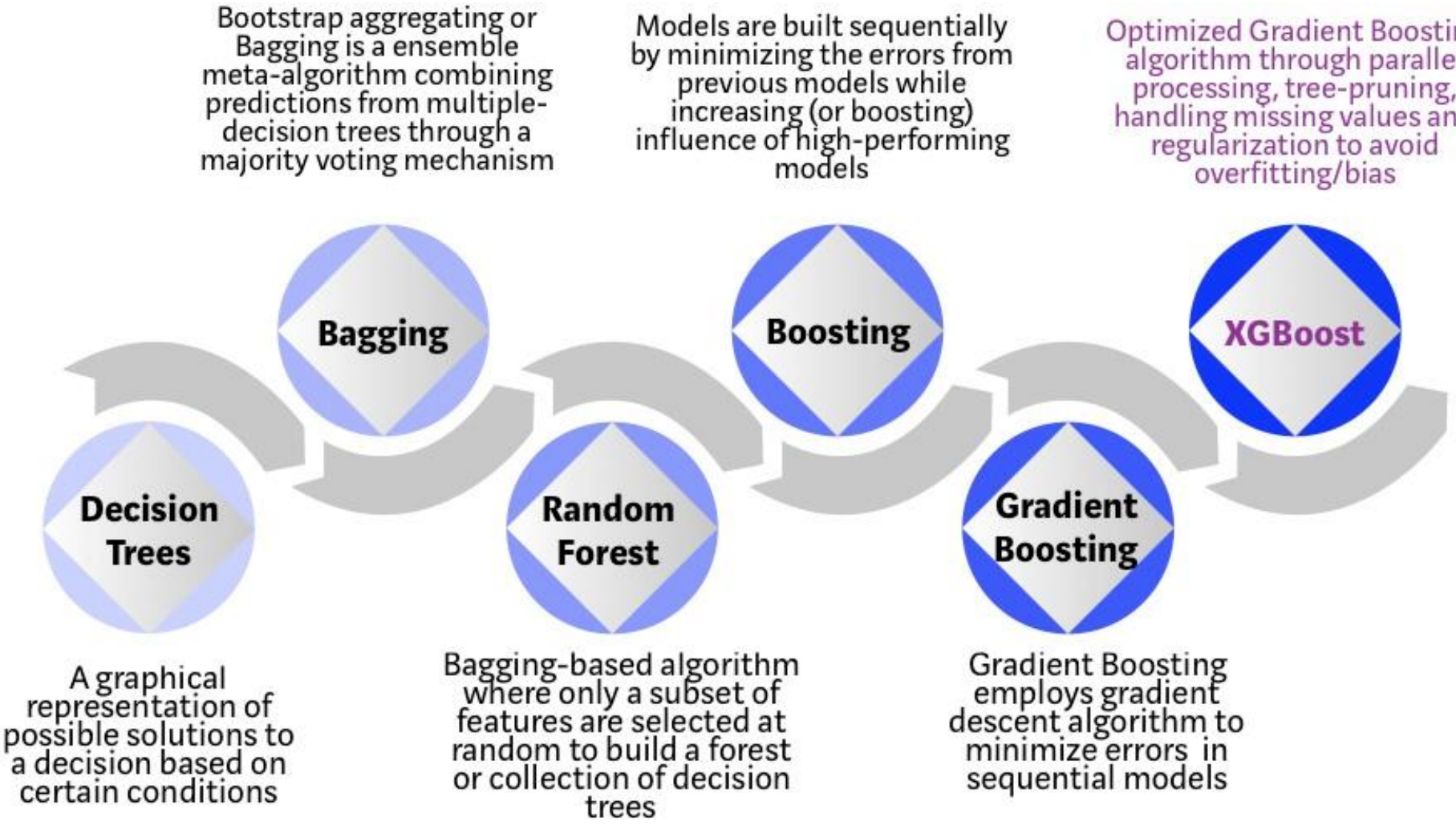
- Boosting refers to a family of algorithms that are able to convert weak learners to strong learners
- Boosting is an iterative technique which adjust the weight of an observation based on the last classification
- An iterative ensemble technique, "boosting," adjusts an observation's weight based on its last classification.
- In case observation is incorrectly classified, "boosting" increases the observation's weight, and vice versa.
- Boosting algorithms reduce bias errors and produce strong predictive models.
- If an observation was classified incorrectly, it tries to increase the weight of this observation and vice versa
- Algorithms
  - AdaBoost
  - Gradient Boosting
  - eXtreme Gradient Boosting

XGBoost

Overview

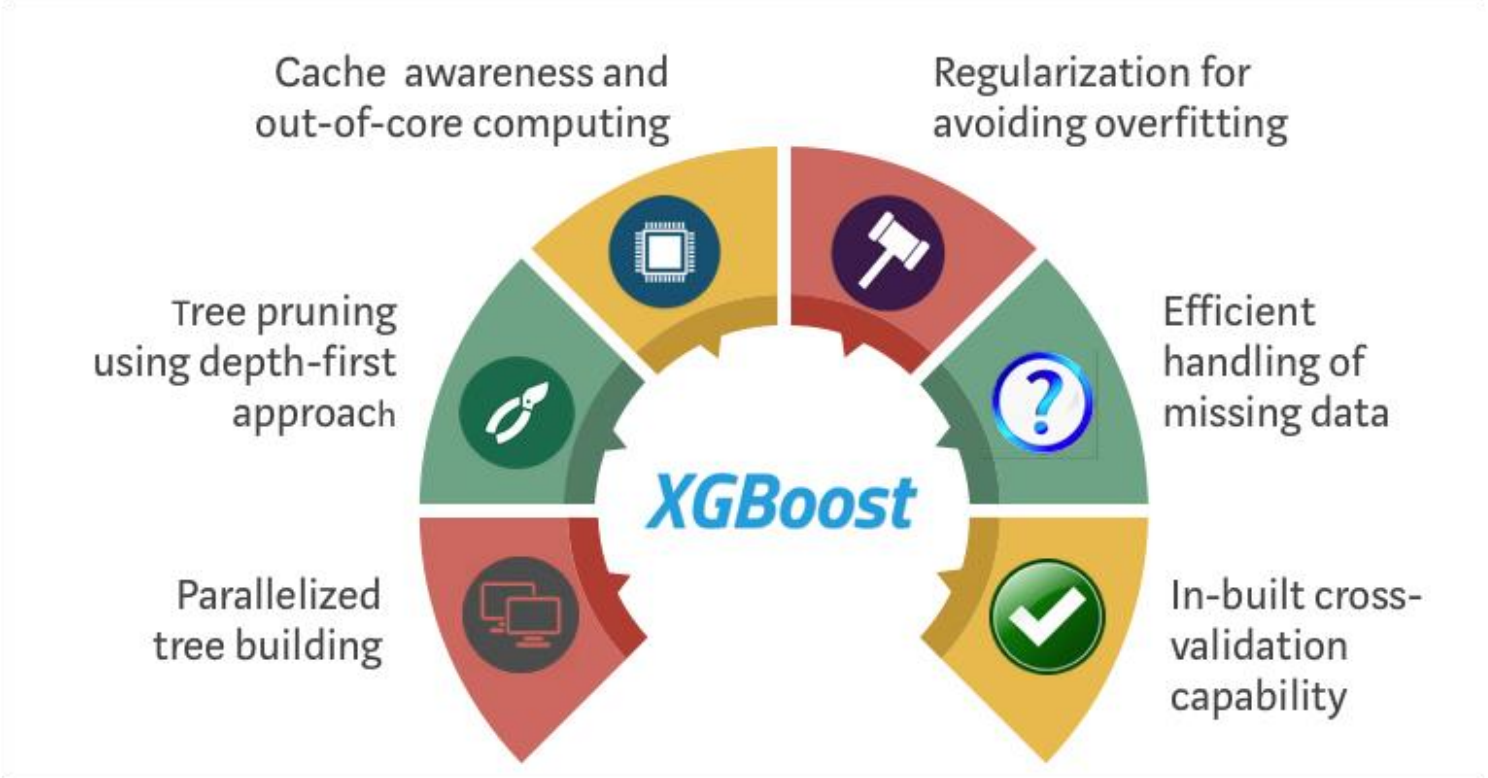
- XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework
- XGBoost algorithm was developed as a research project at the University of Washington
- Since its introduction, this algorithm has not only been credited with winning numerous Kaggle competitions but also for being the driving force under the hood for several cutting-edge industry applications
- As a result, there is a strong community of data scientists contributing to the XGBoost open source projects with ~350 contributors and ~3,600 commits on GitHub

Evolution


Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

Why does it perform so well?

- XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners using the gradient descent architecture
- However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.



System Optimization

- **Parallelization**
  - XGBoost approaches the process of sequential tree building using parallelized implementation
  - This is possible due to the interchangeable nature of loops used for building base learners; the outer loop that enumerates the leaf nodes of a tree, and the second inner loop that calculates the features
  - This nesting of loops limits parallelization because without completing the inner loop (more computationally demanding of the two), the outer loop cannot be started
  - Therefore, to improve run time, the order of loops is interchanged using initialization through a global scan of all instances and sorting using parallel threads
  - This switch improves algorithmic performance by offsetting any parallelization overheads in computation
- **Tree Pruning**
  - The stopping criterion for tree splitting within GBM framework is greedy in nature and depends on the negative loss criterion at the point of split
  - XGBoost uses ‘max_depth’ parameter as specified instead of criterion first, and starts pruning trees backward
  - This ‘depth-first’ approach improves computational performance significantly.

System Optimization

- **Hardware Optimization**
  - This algorithm has been designed to make efficient use of hardware resources
  - This is accomplished by cache awareness by allocating internal buffers in each thread to store gradient statistics
  - Further enhancements such as ‘out-of-core’ computing optimize available disk space while handling big data-frames that do not fit into memory.

Benefits

- **Parallel Computing:** It is enabled with parallel processing (using OpenMP); i.e., when you run xgboost, by default, it would use all the cores of your laptop/machine.
- **Regularization:** I believe this is the biggest advantage of xgboost. GBM has no provision for regularization. Regularization is a technique used to avoid overfitting in linear and tree-based models.
- **Enabled Cross Validation:** In R, we usually use external packages such as caret and mlr to obtain CV results. But, xgboost is enabled with internal CV function (we'll see below).
- **Missing Values:** XGBoost is designed to handle missing values internally. The missing values are treated in such a manner that if there exists any trend in missing values, it is captured by the model.
- **Flexibility:** In addition to regression, classification, and ranking problems, it supports user-defined objective functions also. An objective function is used to measure the performance of the model given a certain set of parameters. Furthermore, it supports user defined evaluation metrics as well.

Benefits

- **Availability:** Currently, it is available for programming languages such as R, Python, Java, Julia, and Scala.
- **Save and Reload:** XGBoost gives us a feature to save our data matrix and model and reload it later. Suppose, we have a large data set, we can simply save the model and use it in future instead of wasting time redoing the computation.
- **Tree Pruning:** Unlike GBM, where tree pruning stops once a negative loss is encountered, XGBoost grows the tree upto max_depth and then prune backward until the improvement in loss function is below a threshold.

Stacking

Stacking

- Stacking is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor
- The base level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level model as features
- The base level often consists of different learning algorithms and therefore stacking ensembles are often heterogeneous