# Machine Learning

## Lecture 14-15: Ensemble Learning Methods

COURSE CODE: CSE451

2021

# Course Teacher

**Dr. Mrinal Kanti Baowaly**

Associate Professor

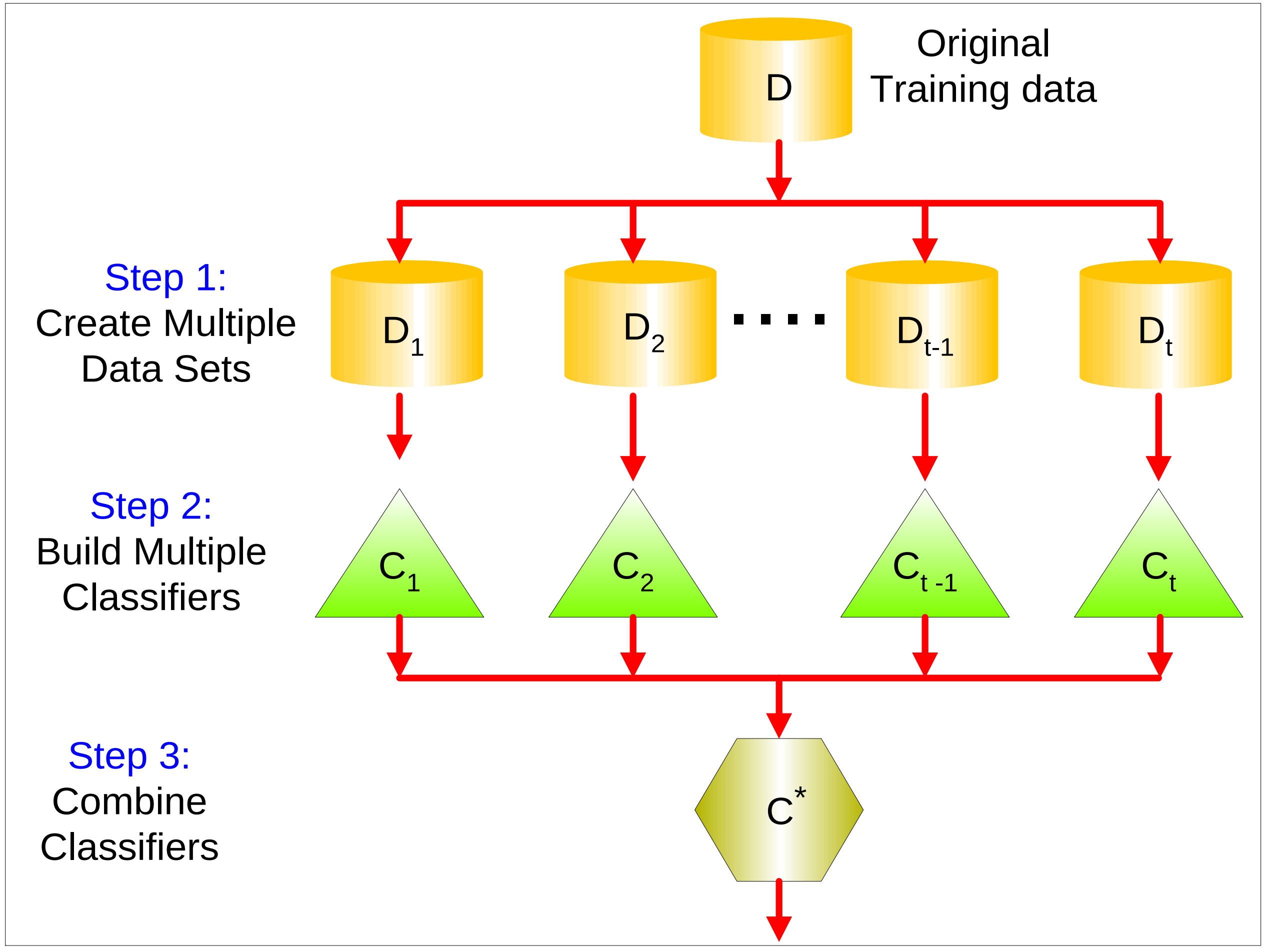
Department of Computer Science and Engineering, Bangabandhu Sheikh Mujibur Rahman Science and Technology University, Bangladesh.

Email: baowaly@gmail.com

# Ensemble Learning

- A powerful way to improve the performance of your model
- Construct a set of classifiers from training data
- Predict class label of test data by combining the predictions made by multiple classifiers or models
- Examples: Random Forest, AdaBoost, Stochastic Gradient Boosting, Gradient Boosting Machine(GBM), XGBoost, LightGBM, CatBoost

# General Approach



# Simple Ensemble Techniques

- Max Voting
- Averaging
- Weighted Averaging

# Max Voting

- Multiple models are used to make predictions for each data point
- The predictions by each model are considered as a 'vote'
- The predictions which we get from the majority of the models are used as the final prediction
- Generally used for classification problems
- For example, when you asked 5 of your colleagues to rate your movie (out of 5); we'll assume three of them rated it as 4 while two of them gave it a 5. Since the majority gave a rating of 4, you can take the final rating of the movie as 4. You can consider this as taking the mode of all the predictions.

# Averaging

- Similar to the max voting technique, multiple predictions are made for each data
- Take an average of predictions from all the models and use it to make the final prediction.
- Averaging can be used in regression or classification problems.
- For example, in the previous case study of max voting, the averaging method would take the average of all the values, i.e. (5+4+5+4+4)/5 = 4.4.
  Hence, final rating of the movie is 4.4.

# Weighted Averaging

- This is an extension of the averaging method.
- All models are assigned different weights defining the importance of each model for prediction.
- For example, if two of your colleagues are critics, while others have no prior experience in this field, then the answers by these two friends are given more importance as compared to the other people.

  The result can be calculated as [(5*0.23) + (4*0.23) + (5*0.18) + (4*0.18) + (4*0.18)] = 4.41.

  Hence, final rating of the movie is 4.41.

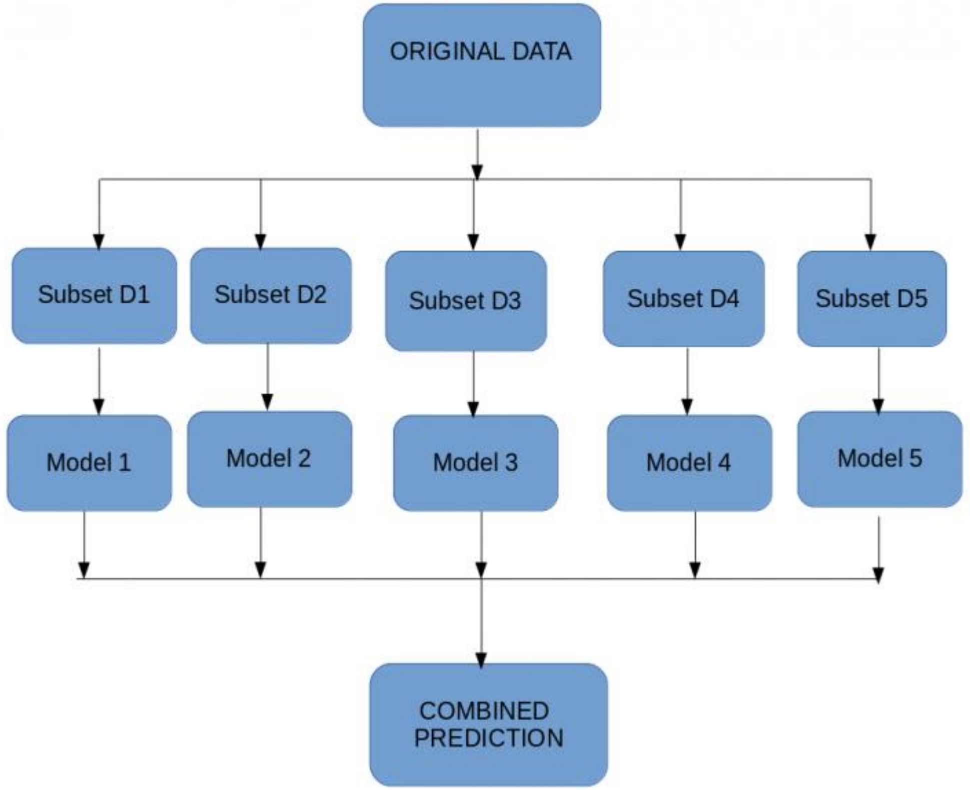
Implementation: [AnalyticsVidhya](), [GeeksForGeeks]()

# Advanced Ensemble Techniques

- **Bagging:** The idea behind bagging is combining the results of multiple models (for instance, all decision trees) to get a generalized result.
- **Boosting:** Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model.
- **Stacking:** Stacking is an ensemble learning technique that uses multiple models’ (called base models) predictions as features to build a new model (called meta-model).

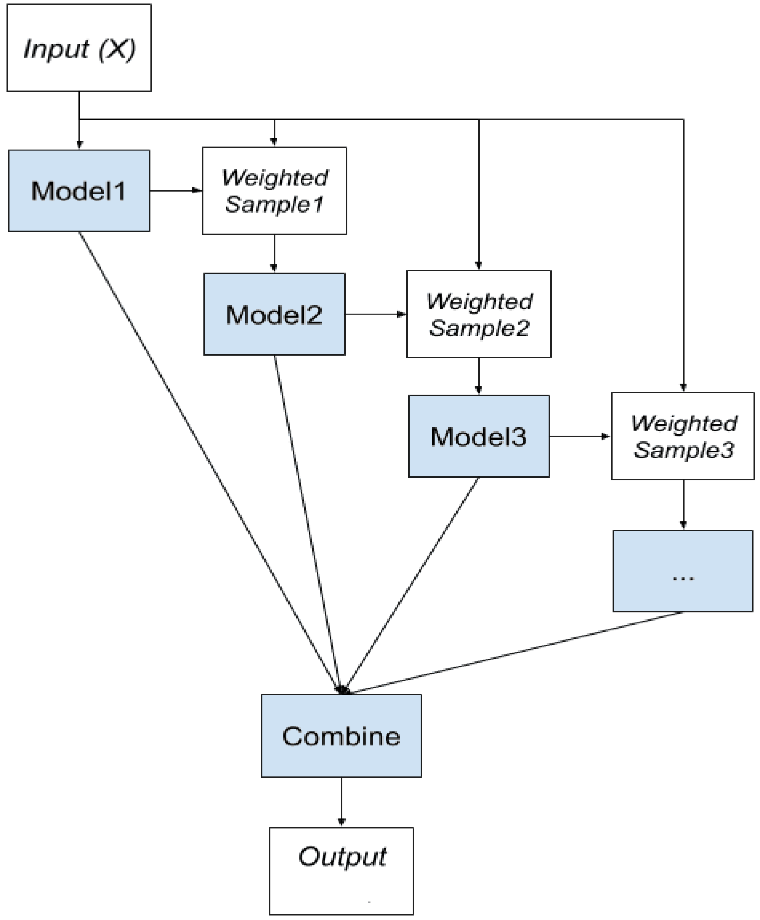
# Bagging

- Multiple subsets are created from the original dataset, selecting observations with replacement (called bootstrapping).
- A base model (weak model) is created on each of these subsets.
- The models run in parallel and are independent of each other.
- The final predictions are determined by combining the predictions from all the models



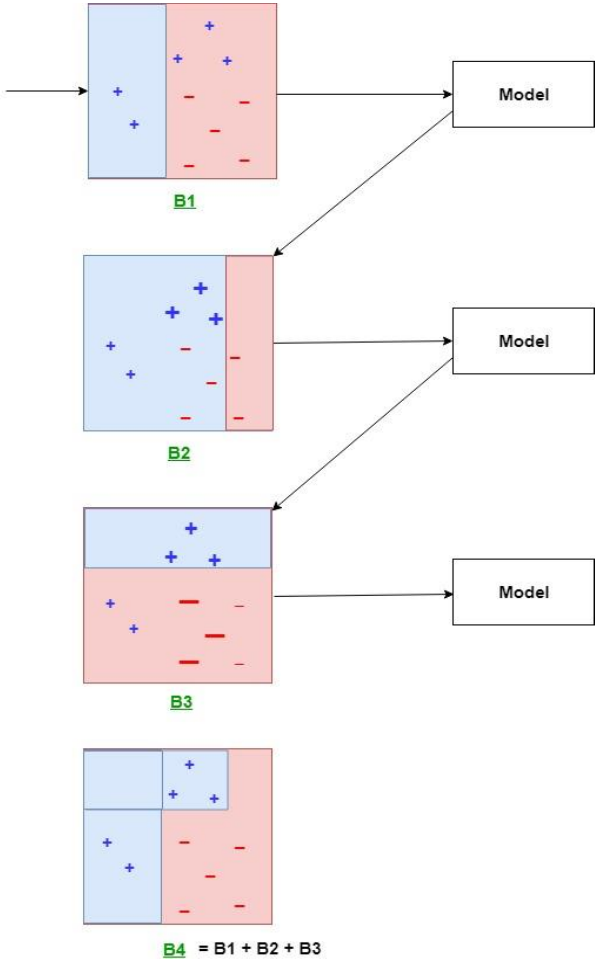
# Boosting

1. A base (weak) learner takes all the distributions and assign equal weight or attention to each observation.
2. If there is any prediction error caused by the base learning algorithm, then we pay higher weight or attention to observations having prediction error.
3. Apply the next base learning algorithm.
4. Repeat step 2 to 3 until the algorithm can correctly classify the output or maximum number of iterations is reached.
5. The weak learners are combined to form a strong learner that will predict a more accurate outcome.



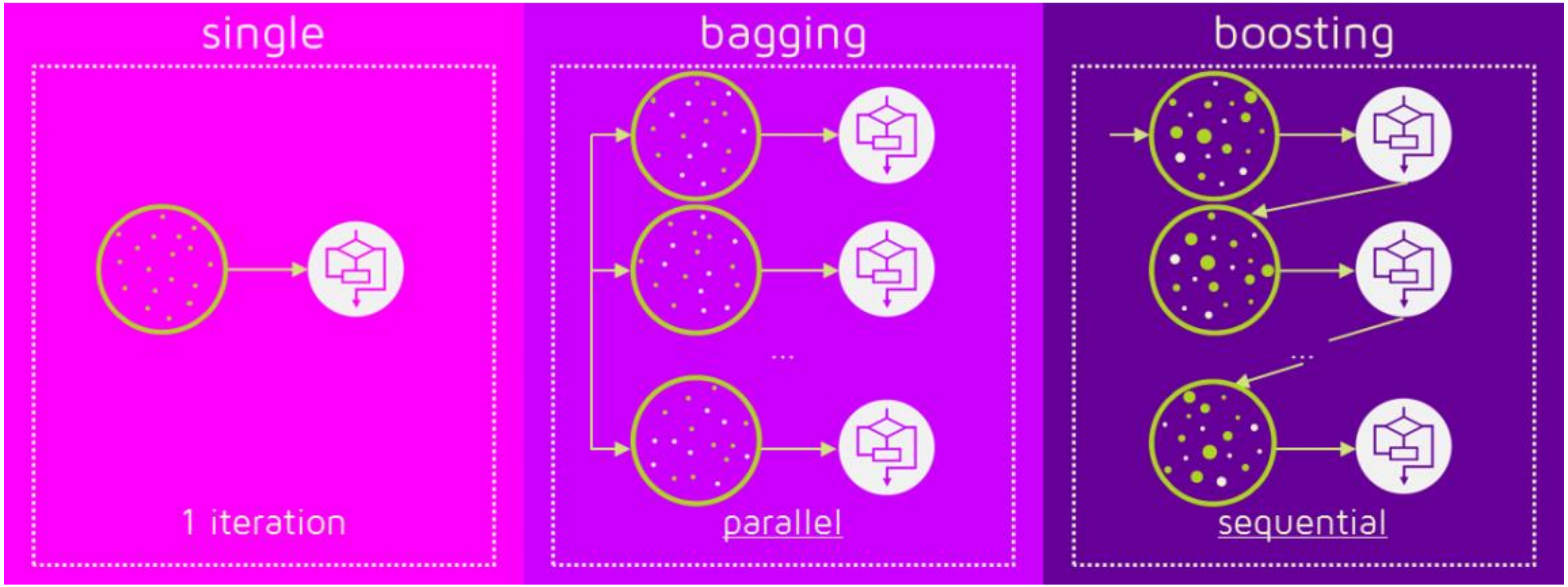
# An Example of Boosting (AdaBoost)

- B1 consist of 10 data points which consist of two types namely plus(+) and minus(-) and 5 of which are plus(+) and other 5 are minus(-) and each one has been assigned equal weight initially. The first model tries to classify the data points and generates a vertical separator line but it wrongly classifies 3 plus(+) as minus(-).
- B2 consists of the 10 data points from the previous model in which the 3 wrongly classified plus(+) are weighted more so that the current model tries more to classify these pluses(+) correctly. This model generates a vertical separator line which correctly classifies the previously wrongly classified pluses(+) but in this attempt, it wrongly classifies three minuses(-).
- B3 consists of the 10 data points from the previous model in which the 3 wrongly classified minus(-) are weighted more so that the current model tries more to classify these minuses(-) correctly. This model generates a horizontal separator line which correctly classifies the previously wrongly classified minuses(-).
- B4 combines together B1, B2 and B3 in order to build a strong prediction model which is much better than any individual model used.



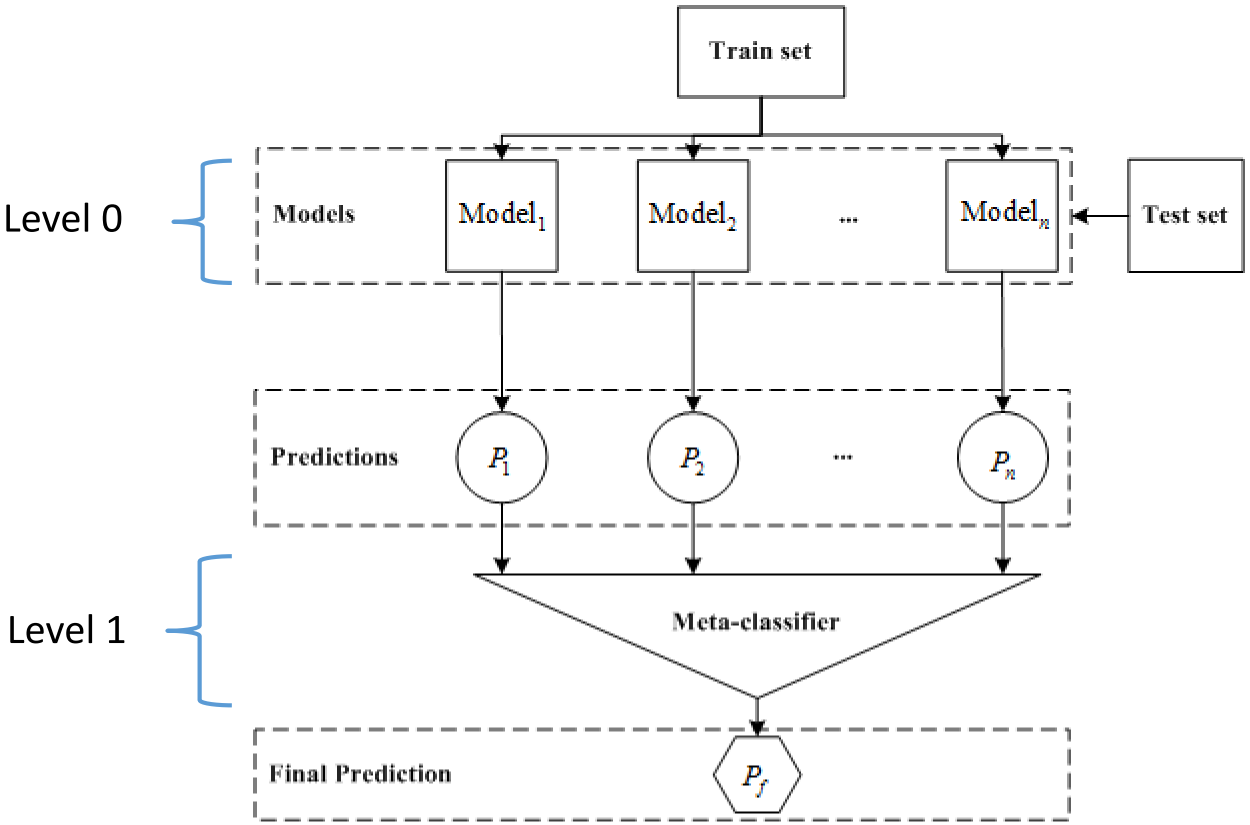

Another Example: [Dataaspirant]

# HW: Difference between Bagging and Boosting




Ref: [QuantDare]

# Stacking Ensemble Learning




Source and Implementation:
[GeeksForGeeks](), [AnalyticsVidhya]()

# Random Forests Classifier

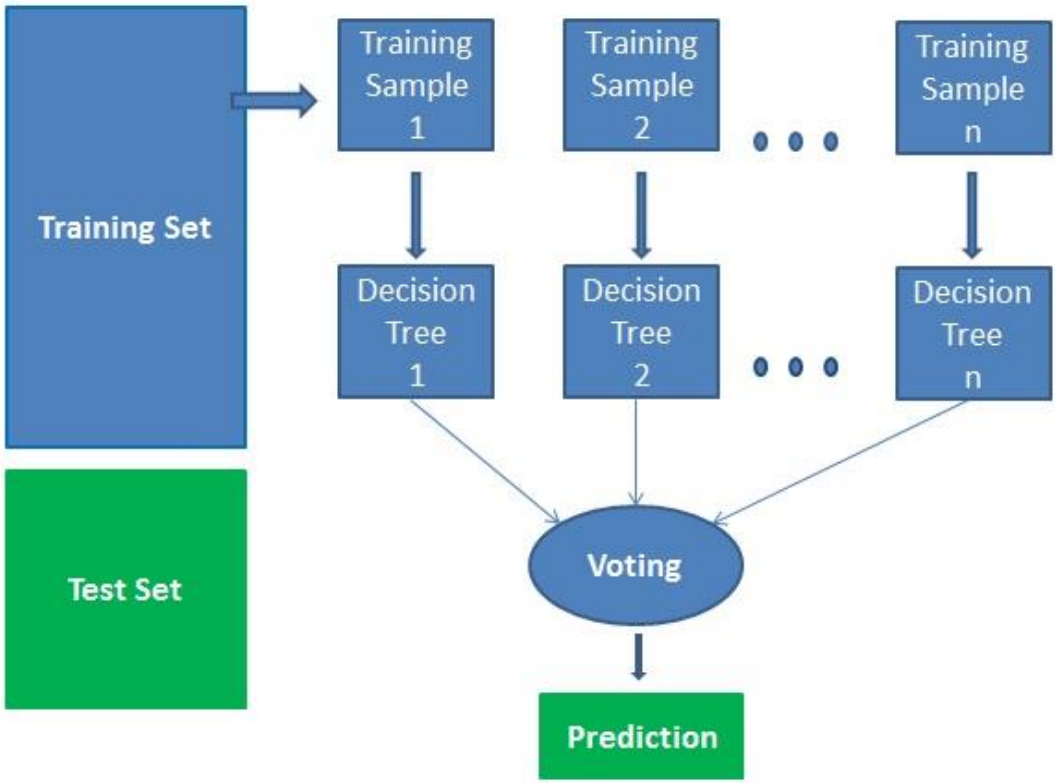
- The random forests algorithm
- How does the algorithm work?
- Its advantages and disadvantages
- Comparison between random forests and decision trees
- Finding important features
- Building a classifier with scikit-learn

# Random Forests Algorithm

- It is a popular supervised learning algorithm.
- Random forest builds multiple decision trees on various random samples (subsets) from a given dataset (called forest), takes the prediction from each tree and predicts the final output based on the majority votes of the predictions.
- It is based on ‘bagging’ ensemble method that yields a more accurate and stable prediction.
- It can be used both for classification and regression.

# How does the algorithm work?

- Select random samples from a given dataset (using bootstrapping).
- Construct a decision tree for each sample and get a prediction result from each decision tree.
- Final prediction is made by selecting the prediction with the most votes (for classification) or averaging the predictions (for regression).



# Advantages of Random Forests

- Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.
- It likely does not suffer from the overfitting problem because it creates multiple trees on random subsets, takes the average or most votes of the predictions of the trees, which cancel out the biases. The randomness and voting or averaging mechanisms in random forests elegantly solve the overfitting problem.
- It can handle missing data.
- It can be used in both classification and regression problems.

# Disadvantages of Random Forests

- Random forests is slow because it builds multiple decision trees and makes the final prediction by combining the predictions of each individual tree.
- The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree

# Random Forest vs Decision Tree

- Random forest is a set of multiple decision trees whereas decision tree is a single tree.
- Deep decision tree may suffer from overfitting, but random forest prevents overfitting by creating multiple trees on random subsets.
- Decision tree is computationally faster, but random forest is slower.
- Random forests is difficult to interpret, while a decision tree is easily interpretable and can be converted to rules.

# Finding Important Features

- Random forests offers a good feature selection indicator.
- Scikit-learn provides an extra variable(feature_importances_) with the model, which shows the relative importance or contribution of each feature in the prediction.
- It automatically computes the relevance score of each feature in the training phase. Then it scales the relevance down so that the sum of all scores is 1. The higher the score, the more important the feature.
- This score will help you choose the most important features and drop the least important ones for model building.
- Random forest uses gini importance (or impurity-based feature importance) to calculate the importance of each feature.

# More on Random Forest (Do it by yourself)

- Build a Random Forest classifier with scikit-learn
- Find important features of a Random Forest classifier with scikit-learn
- Build both Decision Tree and Random Forest classifiers and compare their performances
- Why does Random Forest model outperform the Decision Tree?

Source: [DataCamp](), [AnalyticsVidhya]()

# Advanced Boosting Methods

- What is GBM?
- What is XGBoost?
- What is LightGBM?
- Advantages of using Light GBM and XGBoost
- Build classifiers using GBM, LightGBM and XGBoost
- Compare GBM, LightGBM and XGBoost
- Which algorithm takes the crown: LightGBM or XGBoost?

Source: AnalyticsVidhya [1], [2]

# Advanced Boosting Methods(Cont..)

- What is CatBoost?
- Advantages of CatBoost library
- CatBoost in comparison to other boosting algorithms
- Installing CatBoost
- Solving ML challenge using CatBoost

Source: AnalyticsVidhya, Dataaspirant

# Comparison of CatBoost to other boosting algorithms

| | CatBoost | | LightGBM | | XGBoost | | H2O | |
|---|---|---|---|---|---|---|---|---|
| | Tuned | Default | Tuned | Default | Tuned | Default | Tuned | Default |
| Adult | **0.26974** | 0.27298 +1.21% | 0.27602 +2.33% | 0.28716 +6.46% | 0.27542 +2.11% | 0.28009 +3.84% | 0.27510 +1.99% | 0.27607 +2.35% |
| Amazon | **0.13772** | 0.13811 +0.29% | 0.16360 +18.80% | 0.16716 +21.38% | 0.16327 +18.56% | 0.16536 +20.07% | 0.16264 +18.10% | 0.16950 +23.08% |
| Click prediction | **0.39090** | 0.39112 +0.06% | 0.39633 +1.39% | 0.39749 +1.69% | 0.39624 +1.37% | 0.39764 +1.73% | 0.39759 +1.72% | 0.39785 +1.78% |
| KDD appetency | 0.07151 | **0.07138** −0.19% | 0.07179 +0.40% | 0.07482 +4.63% | 0.07176 +0.35% | 0.07466 +4.41% | 0.07246 +1.33% | 0.07355 +2.86% |
| KDD churn | **0.23129** | 0.23193 +0.28% | 0.23205 +0.33% | 0.23565 +1.89% | 0.23312 +0.80% | 0.23369 +1.04% | 0.23275 +0.64% | 0.23287 +0.69% |
| KDD internet | **0.20875** | 0.22021 +5.49% | 0.22315 +6.90% | 0.23627 +13.19% | 0.22532 +7.94% | 0.23468 +12.43% | 0.22209 +6.40% | 0.24023 +15.09% |

# A Comprehensive Course on Ensemble Learning


Enroll now

# Study Materials of Ensemble Methods

- AnalyticsVidhya: A Comprehensive Guide to Ensemble Learning (with Python codes)
- GeeksForGeeks: Ensemble Method in Python
- AnalyticsVidhya: Basics of Ensemble Learning Explained in Simple English
- Dataaspirant: How the Kaggle winners algorithm XGBoost algorithm works