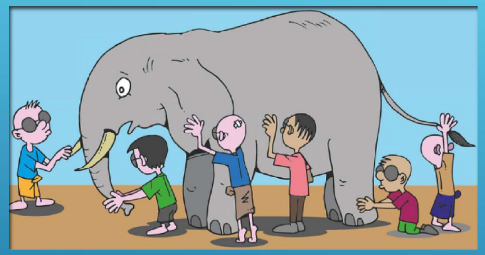
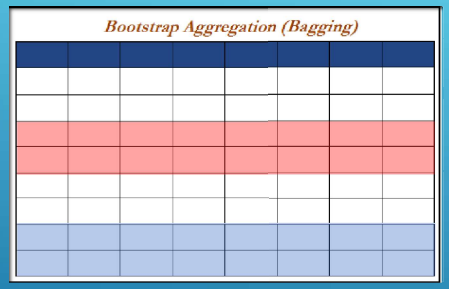


- Combine decisions from multiple models to improve overall performance
- Help minimize causes of error due to noise, bias and variance
- Major schemes:
  - Bagging
  - Boosting

# PART2: DECISION TREE ENSEMBLES



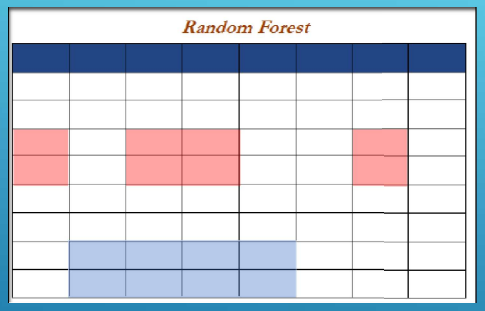



Two samples (pink, blue) with all variables

# BAGGING

- **Bootstrap aggregation** or **bagging** introduced by Leo Breiman in 1994.
- Bootstrapped datasets are created by **sampling with replacement**.
- Build a number of decision trees on bootstrapped samples from training data
- Combine the results of the models by **averaging** or **majority voting**
- The algorithm aims to reduce the chance of overfitting.
- Due to all variables selected, order of candidate/variable chosen to split remains more or less the same for all the individual trees.
- Variance reduction on correlated individual entities does not work effectively while aggregating them.





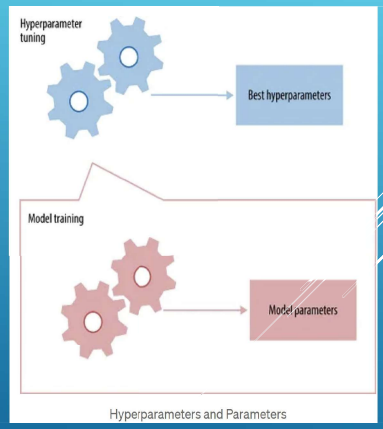

# RANDOM FORESTS

- In bagging, a random sample with replacement is selected to train every tree in the ensemble. The tree is trained on all the features.
- While each decision tree in the random forest is given a randomly selected subset of features and a randomly selected subset of the dataset for the selected features to **ensure low correlation among decision trees.**



# PARAMETER VS. HYPERPARAMETER

- model ***parameters*** are learned during training — such as the *slope* and *intercept* in a linear regression
- ***hyperparameters*** must be set by the data scientist **before** training
- Scikit-Learn implements a set of sensible default hyperparameters for all models, but these are not guaranteed to be optimal for a problem.
- Hyperparameter tuning relies more on experimental results than theory, and thus the best method to determine the optimal settings is to try many different combinations evaluate the performance of each model.



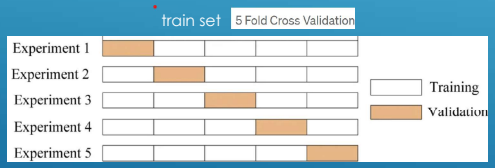

Hyperparameters and Parameters

- n_estimators: number of trees considered for majority voting in the forest. The more trees, the better the performance, but more computation time. It is usually set as 100, 200, 500, and so on.
- max_features: max number of features consider for each best splitting point search. Typically, for an m-dimensional dataset, rounded sqrt(m) is a recommended value for max_features. This can be specified as max_features="sqrt" in scikit-learn.
- max_depth: max number of levels in each DT. It tends to overfit if it is too deep, or to underfit if it is too shallow.
- min_samples_split: min number of data points placed in a node before the node is split. Too small a value tends to cause overfitting, while too large a value is likely to introduce underfitting. 10, 30, and 50 might be good options to start with.
- min_samples_leaf = min number of data points allowed in a leaf node
- bootstrap = method for sampling data points (with or without replacement)
- Practically, application of **grid search** on tuning different combinations of hyperparameters will provide better and more robust results.

## HYPERPARAMETERS IN RANDOM FOREST

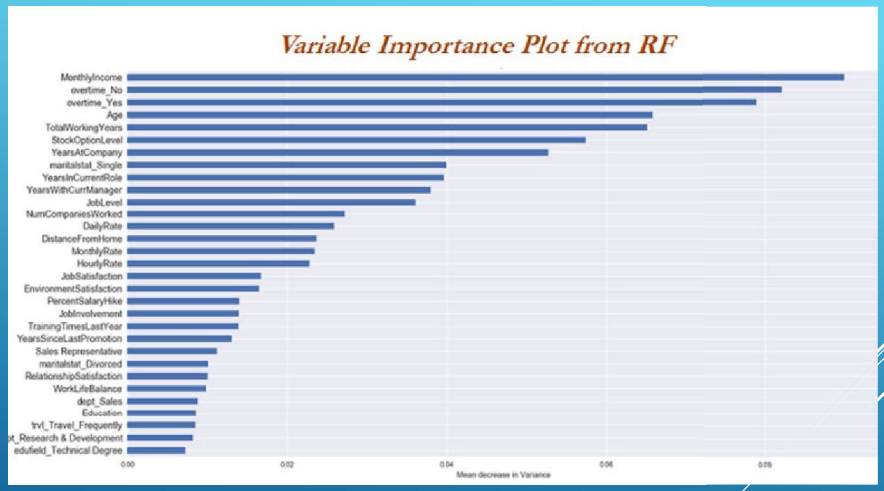


- For hyperparameter tuning, many iterations of K-Fold CV process are performed, each time using different model settings. Then, compare all of the models, select the best one, train it on the full training set, and then evaluate on the testing set.
- If we have 10 sets of combinations of hyperparameters and are using 5-Fold CV, that represents 50 training loops.
- Fortunately, model tuning with K-Fold CV can be automatically implemented in Scikit-Learn.
- Using SK-Learn's RandomizedSearchCV method will randomly sample from the defined grid of hyperparameter ranges, and, performing K-Fold CV with each combination of values.
- GridsearchCV method




## HYPERPARAMETER TUNING WITH CROSS VALIDATION








- In boosting, all models are trained in sequence, instead of in parallel as in bagging.
- Each model is trained on the same dataset, but each data sample is under a different weight factoring, in the previous model's success.
- The weights are reassigned after a model is trained, which will be used for the next training round.
- In general, weights for mispredicted samples are increased to stress their prediction difficulty.
- There are many boosting algorithms e.g. AdaBoost, Gradient Boosting and XGBoost; boosting algorithms differ mostly in their weighting scheme.
- Boosting relies on creating a series of weak learners each of which might not be good for the entire data set but is good for some part of the data set. Thus, each model actually boosts the performance of the ensemble.
- Boosting has shown better predictive accuracy than bagging.

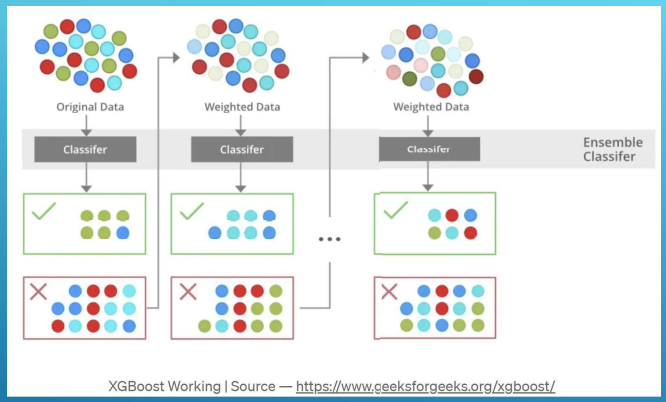
## BOOSTING

# XGBOOST

- XGBoost is a gradient boosting algorithm, originally developed by *Tianqi Chen*.
- It works by combining a number of weak learners to form a strong learner.
- XGBoost works by training a number of decision trees. Each tree is trained on a subset of the data, and the predictions from each tree are combined to form the final prediction.
- A number of most important parameters include:

  - ***max_depth:*** The maximum depth of the decision trees.
  - ***eta:*** The learning rate.
  - ***gamma:*** The minimum loss reduction required to make a split.
  - ***subsample:*** The fraction of the training data that is used to train each tree.






XGBoost Working | Source — https://www.geeksforgeeks.org/xgboost/

XGBoost has been shown to outperform other machine learning algorithms in a variety of tasks, including classification, regression and ranking.



# DIFFERENCES BETWEEN BAGGING AND BOOSTING

| | Bagging | Boosting |
|---|---|---|
| **Differences** | Individual models are built separately | Each new model is influenced by the performance of those built previously |
| | Equal weight is given to all models | Weights a model's contribution by its performance |



- Produce rules in simple English sentences, easily interpreted and presented to senior management without any editing.
- DT can be applied to either classification or regression problems.
- Able to handle both numerical and categorical variables.
- DT is a non-parametric model.
- No assumptions are made on the underlying distribution of the data.
- Useful in data exploration: DT is one of the fastest ways to identify the most significant variables.
- Overfitting/ high variance error is one of the most practical difficulties for DT models. The problem can be solved by pruning and ensemble techniques.