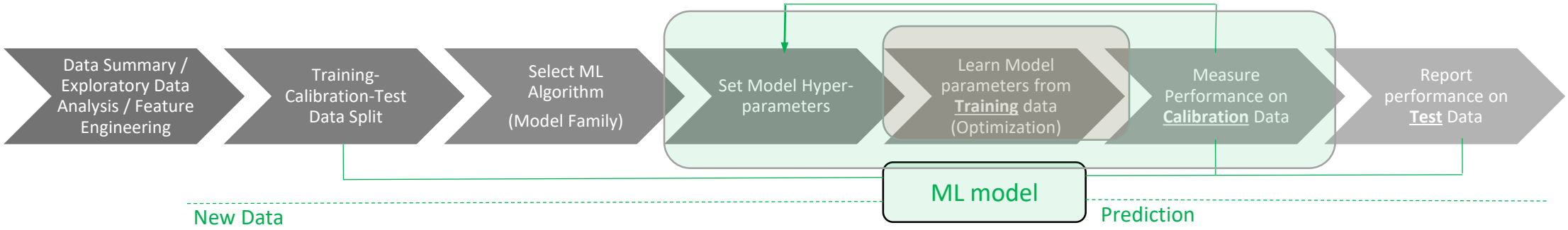
Ensemble Learning

Praphul Chandra

1. James, Gareth, et al. *An introduction to statistical learning*. Vol. 6. New York: springer, 2013.
2. Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning.* Vol. 1. Springer, Berlin: Springer series in statistics, 2001.
3. Kuhn, Max, and Kjell Johnson. *Applied predictive modeling*. New York: Springer, 2013.

Machine Learning Framework

Machine Learning Framework

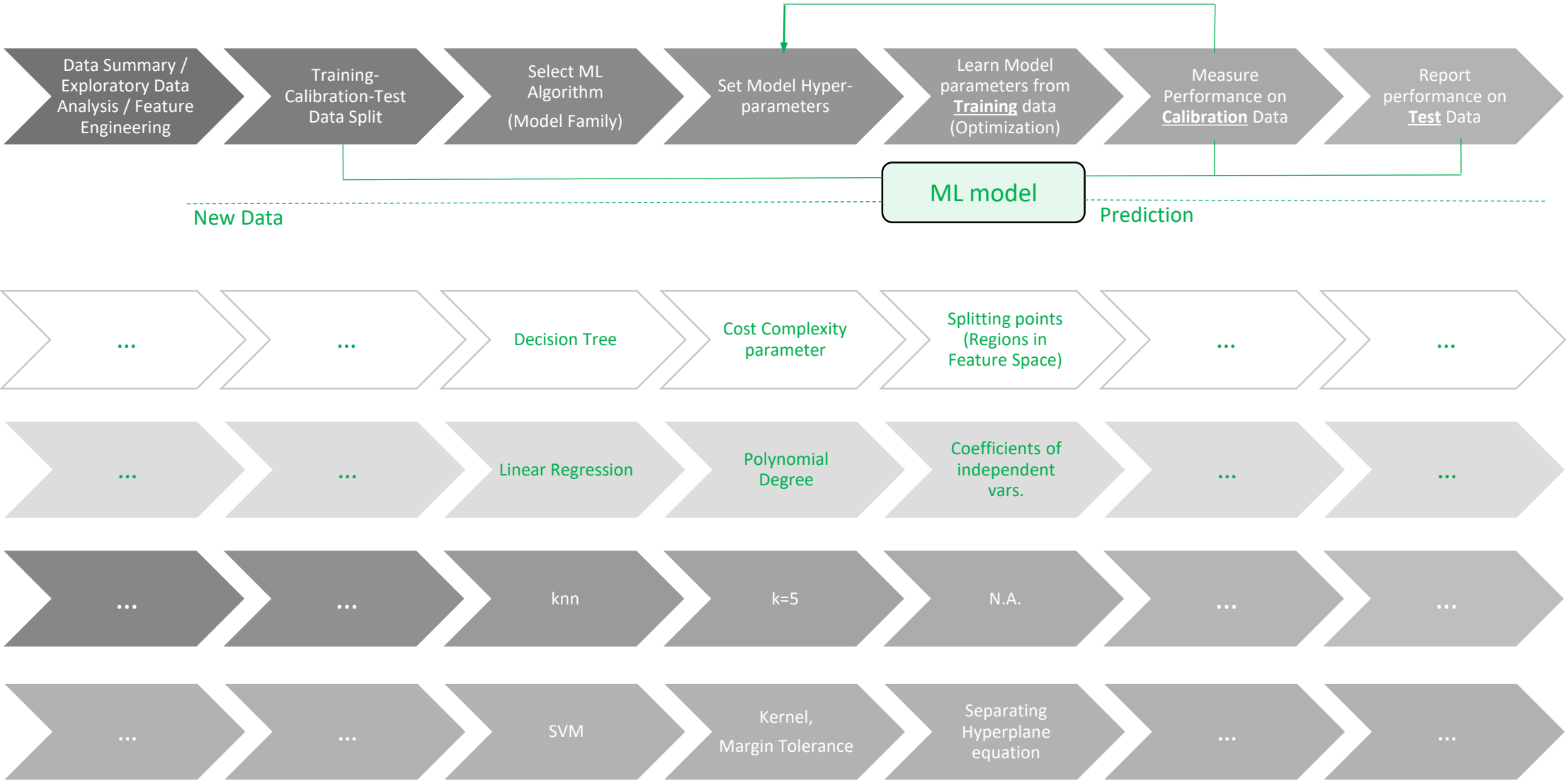


- Hyperparameter Optimization
  - Optimal model complexity
  - **Iterate** over Hyperparameters + CV (grid search)

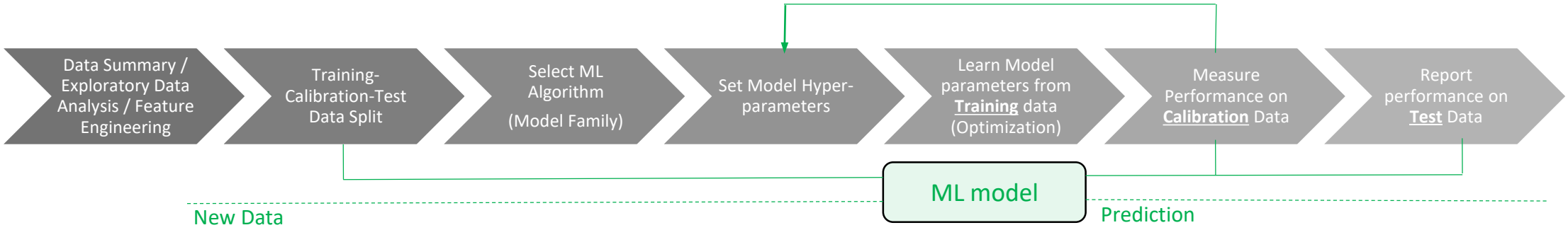
- Parameter Optimization
  - Minimize the Loss Function : L(Y, f(X))
  - Given the model (hyperparameter)
  - Solved using (convex) **optimization** techniques

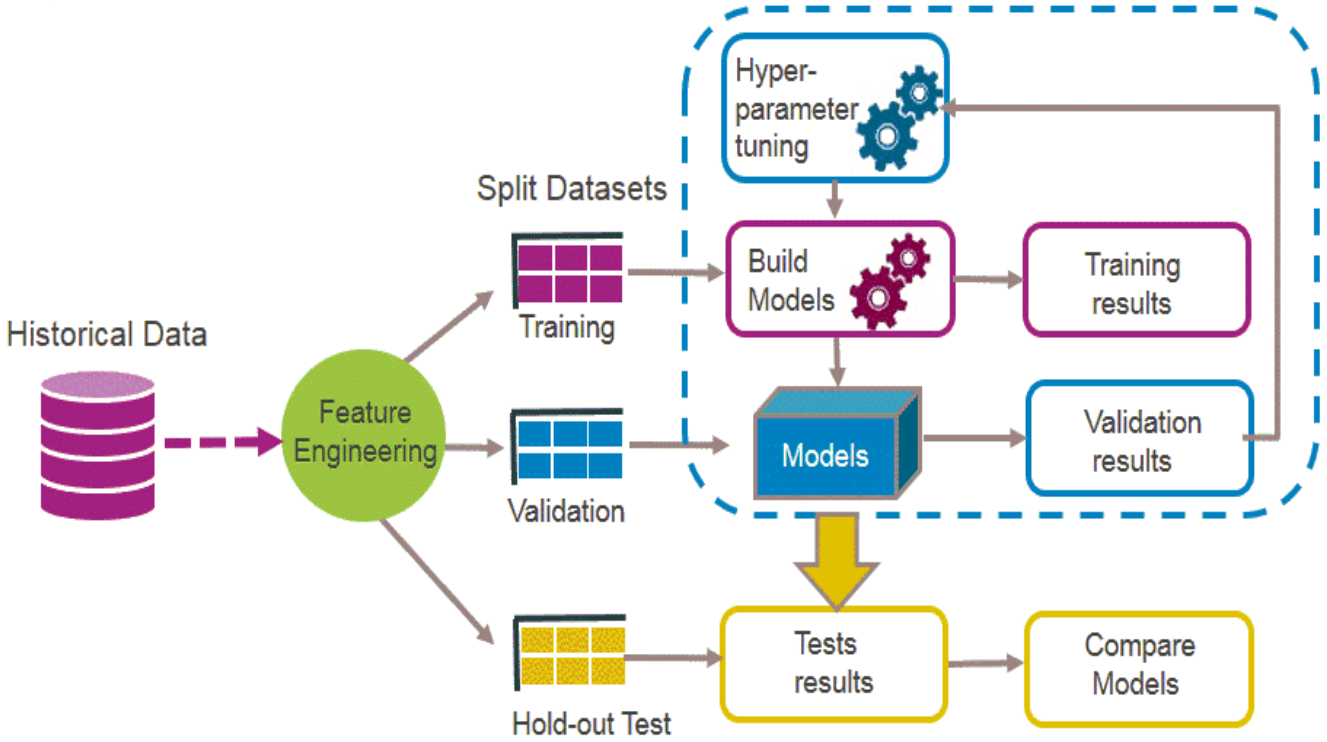
- knn
  - Hyperparameter: Lower k ➔ Higher complexity
  - Parameter: N.A.
- Decision Tree
  - Hyperparameter : Lower α ➔ Higher complexity
  - Parameter: Split points
- Linear Regression
  - Hyperparamet: degree ➔ Higher complexity
  - Parameter: Coefficients of independent variables
- Support Vector Machine
  - Hyperparamater: Kernel, slack
  - Parameter : Coefficients (in the new space)

Machine Learning Framework



Machine Learning Framework (cont'd)





Ensemble Learning

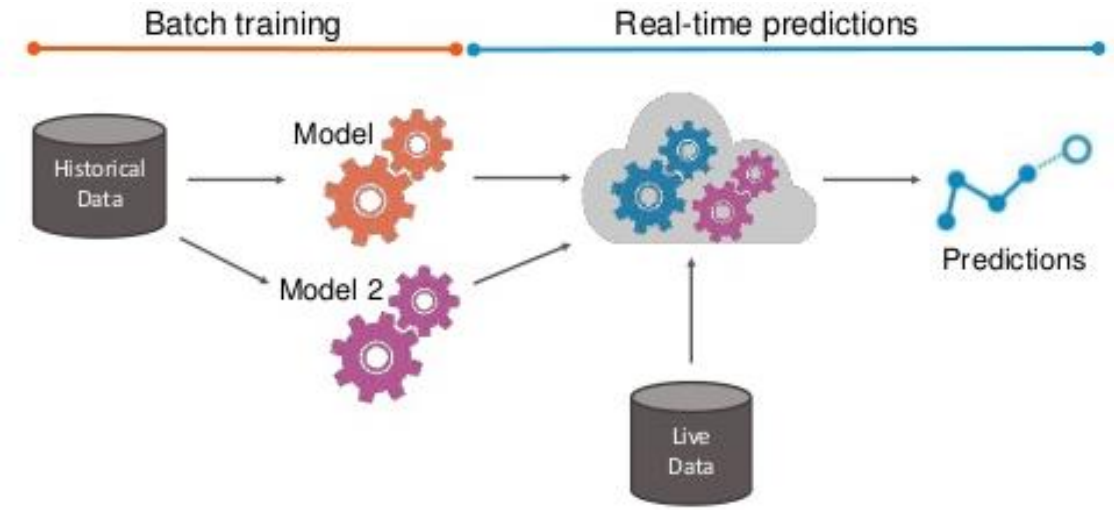
Bagging, Boosting, Stacking etc.

Ensemble Learning : The Big !dea

- ***"Why build 1 when you can have 2 at twice the price?"***
  - Improve model accuracy
  - Different models optimal for different data subsets
  - Reduce variance (Bias-Variance tradeoff)
  - 3?, 4?, 100?

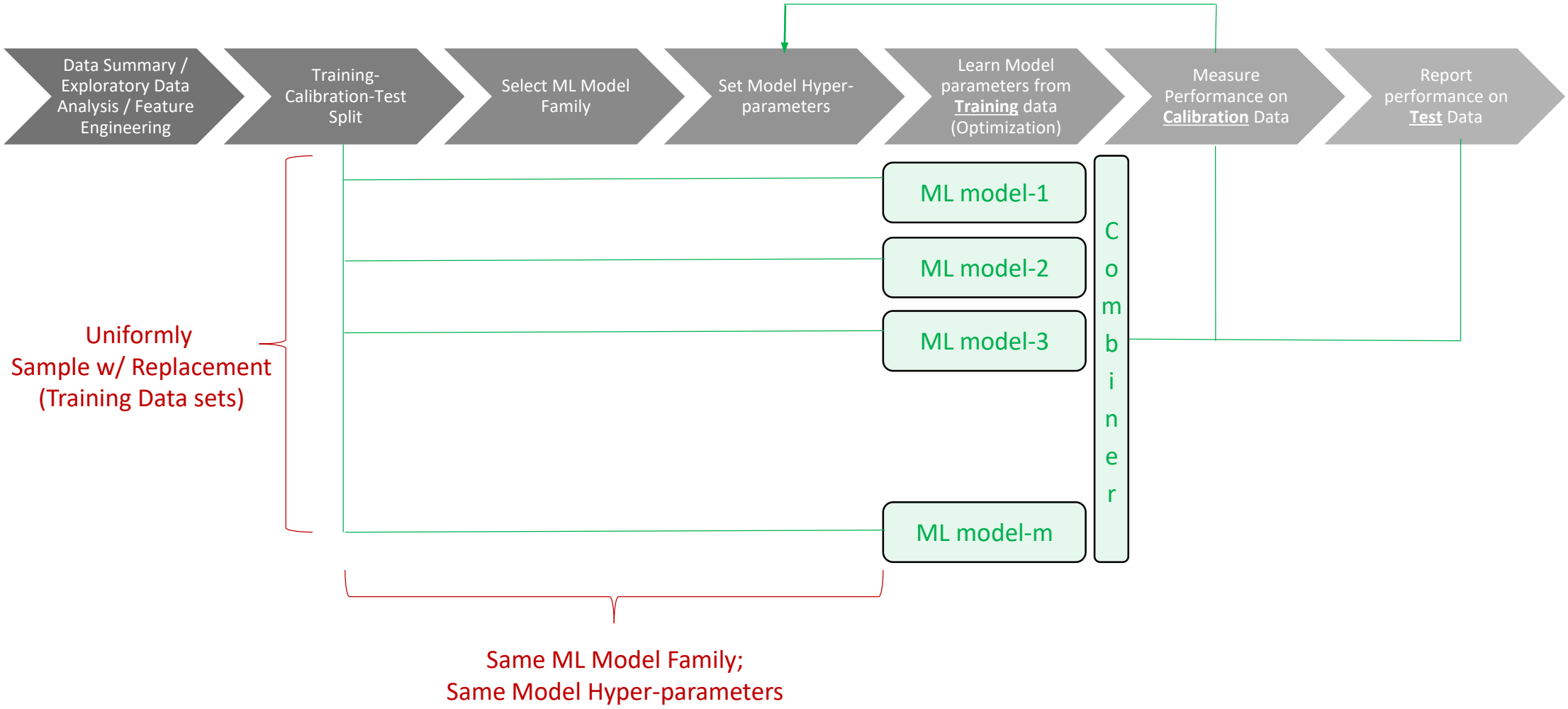
- Aggregating (Combining) results of multiple models
  - Regression: (Weighted) Mean, Median, Max, Min
  - Classification : (Weighted) Majority Voting, Borda

- Building "different" models
  - Model families: knn, decision tree, linear regression
  - Different training data!
  - Different features!



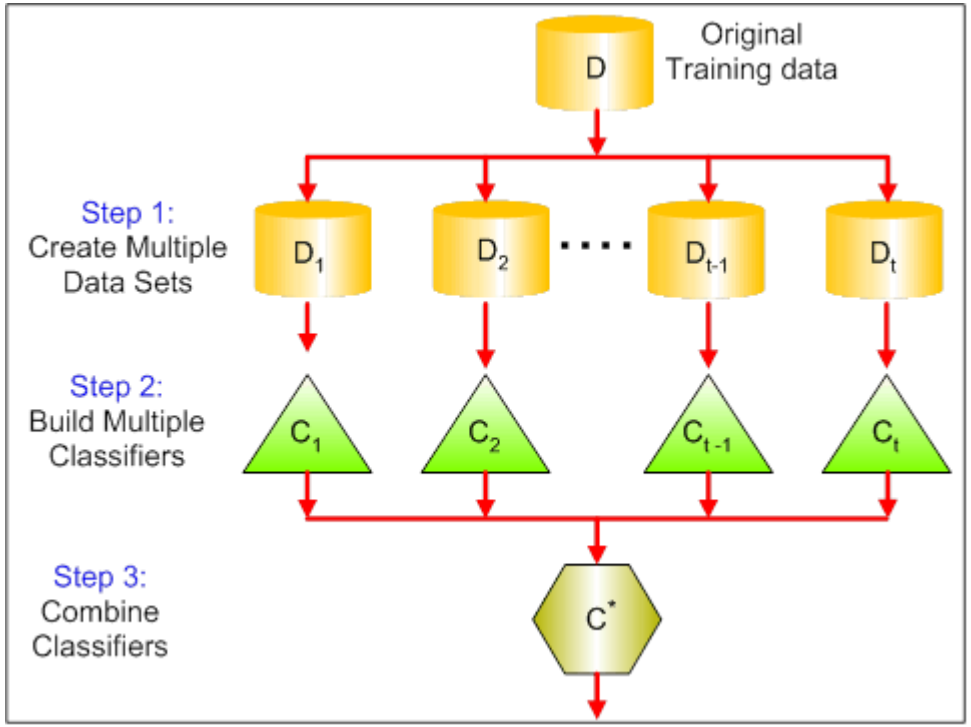
Bagging

Bagging: ML Framework



Bagging: Bootstrap Aggregation

- Different training data!
  - Sample Training Data with Replacement
  - Same algorithm on different subsets of training data
- Bagging
  - Algorithm independent : general purpose technique
  - Well suited for high variance algorithms
  - Variance Reduction: Averaging a set of observations reduces variance
  - Choose # of classifiers to build (B)
- Application
  - Use with high variance algorithms (DT, NN)
  - Easy to parallelize
  - Limitation: Loss of Interpretability
  - Limitation: What if one of the features dominates?

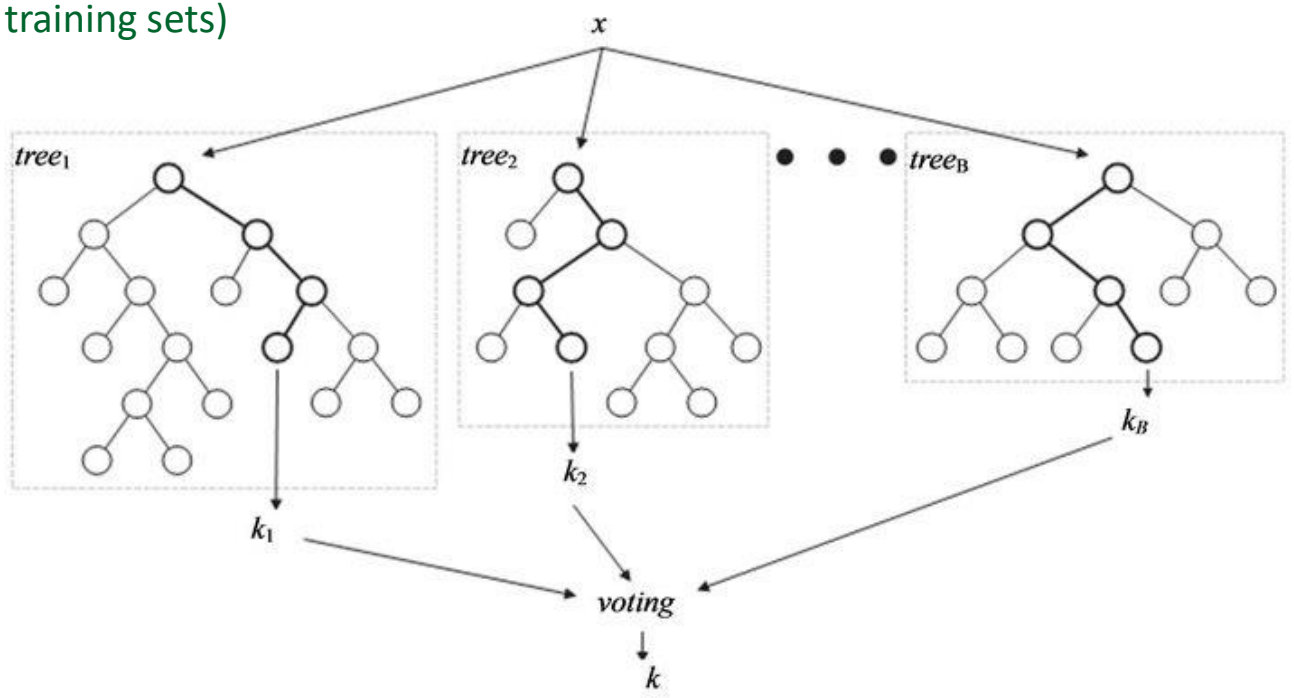


Bagged Trees

- Decision Trees have high variance
  - The resulting tree (model) depends on the underlying training data
  - (Unpruned) Decision Trees tend to overfit
  - One option: Cost Complexity Pruning

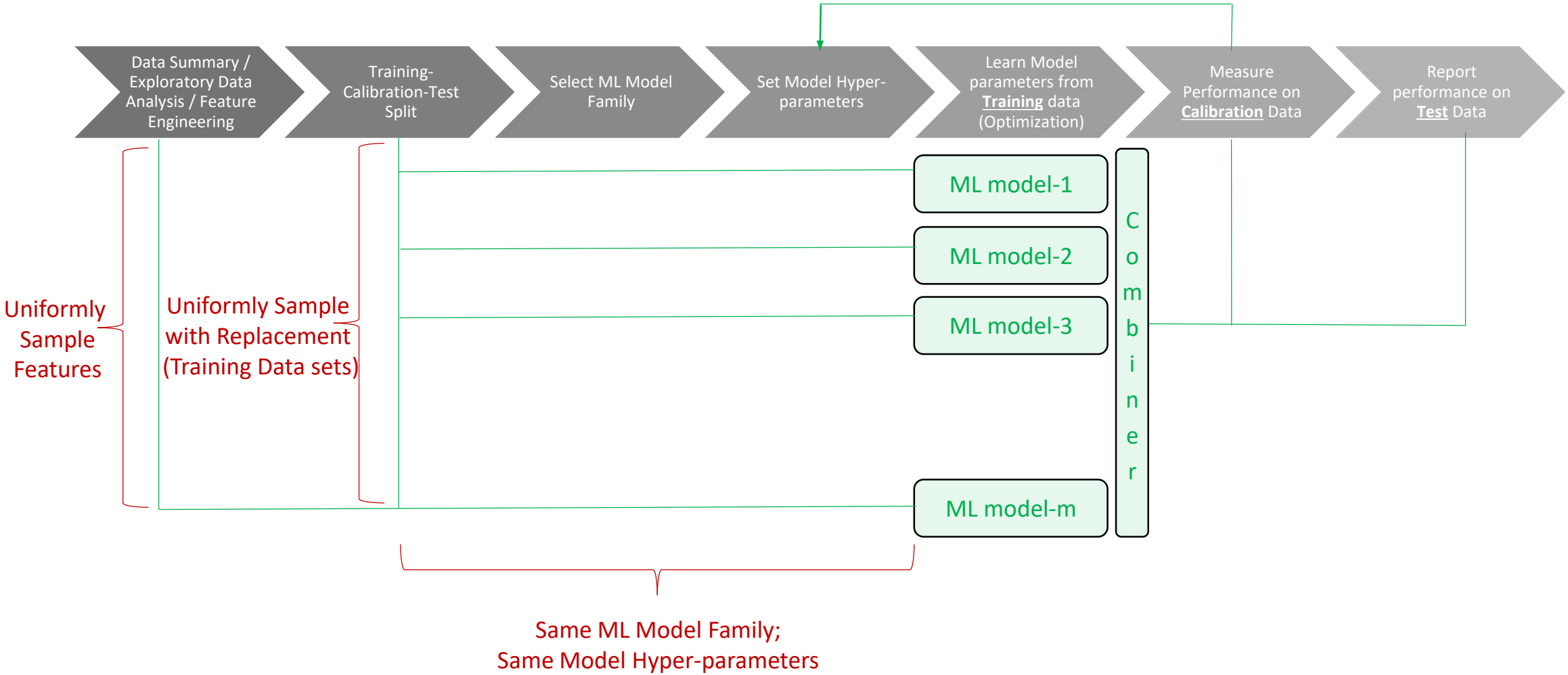
- Bag Trees
  - Sample with replacement (1 Training set → Multiple training sets)
  - Train model on each bootstrapped training set
  - Multiple trees; each different : A garden ☺
  - Each DT predicts; Mean / Majority vote prediction
  - Choose # of trees to build (B)

- Advantages
  - Reduce model variance / instability
  - But Harder to interpret (rules ?)



Random (Feature) Sub-spaces

Random subspaces: ML Framework



Random Feature Subspaces

- Try different features for different training data
  - Sample Training Data with Replacement
  - Build each model using a random subset of the features!
  - Same algorithm on different subsets of training data

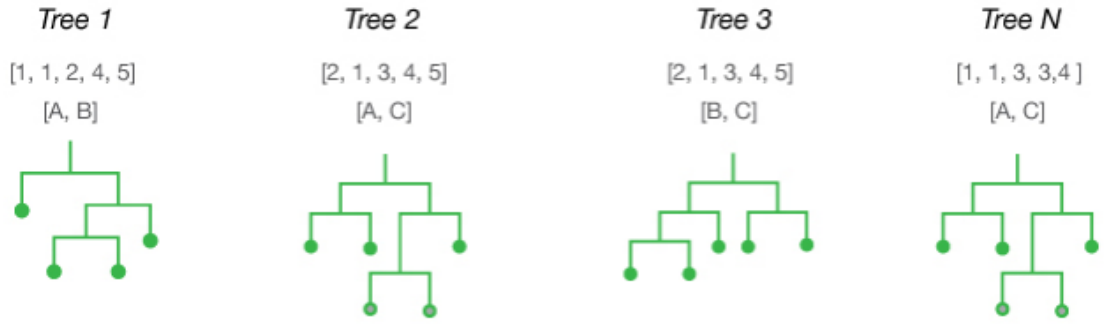
- Why?
  - If there is one very strong predictor & other moderately strong predictors..
  - All models will give high importance to the strong predictor ➔ All models in the ensemble will be similar
  - Choose # of classifiers to build (B) and # of features to sample (m) from p available features
  - Recommended heuristics: m = sqrt(p) *(m = p : Approach reduces to Bagged Trees)*

- Advantages
  - De-correlates the models in the ensemble
  - Improve accuracy of prediction

Random Feature Spaces in Decision Trees : Random Forests

- Decision Trees have high variance
  - The resulting tree (model) depends on the underlying training data
  - Bagged Trees can help reduce variance; Random Forests even more so…

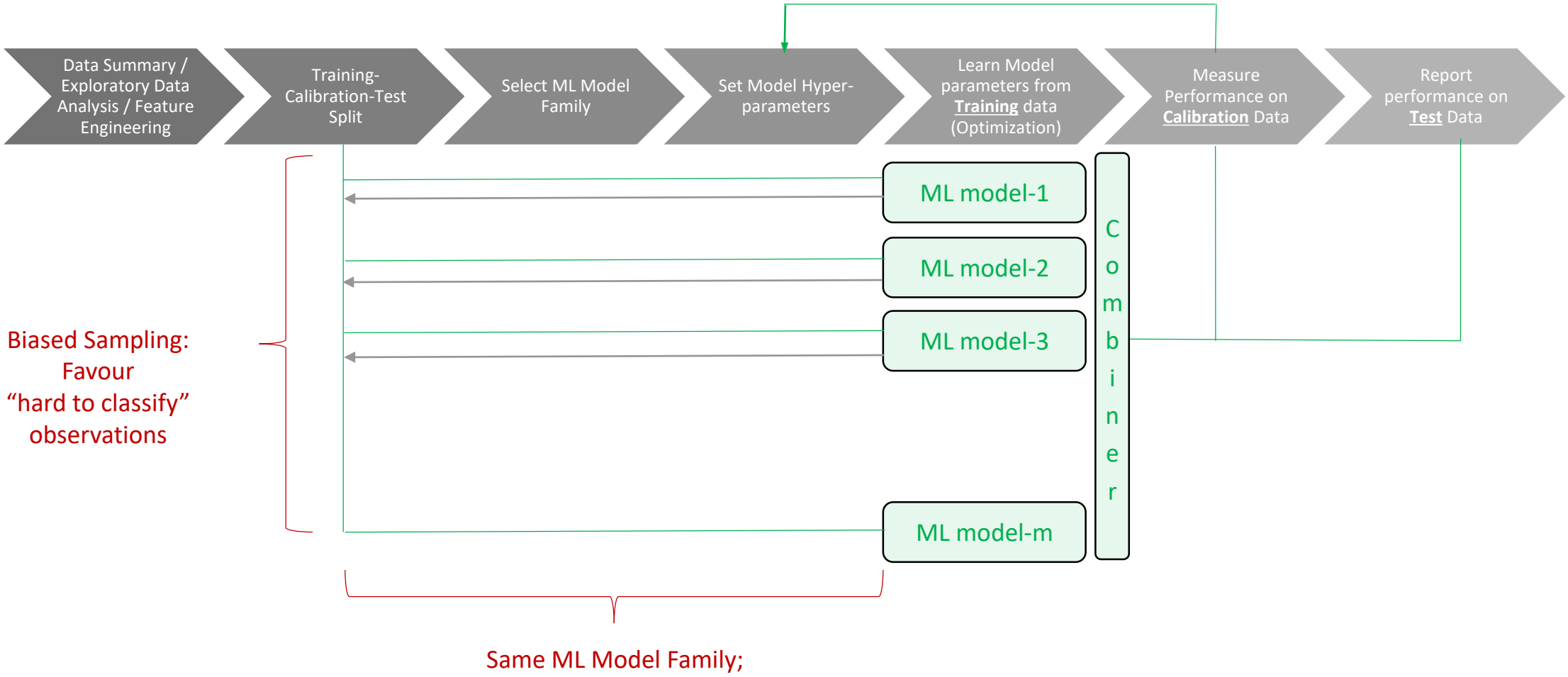
- Random Forests
  - Sample with replacement (1 Training set → Multiple training sets)
  - Train model on each bootstrapped training set
  - Each tree uses a random subset of feature : A random forest
  - Each DT predicts; Mean / Majority vote prediction
  - Faster than bagging (fewer splits to evaluate per tree)
  - Choose # of trees to build (B) and # of features to sample (m) from p available features



The diagram above assumes five observations [1, 2, 3, 4, 5] and three predictor variables [A, B, C]. It shows the construction of four simple (but different) trees. The observations have been sampled with replacement which means that some observations can occur more than once. In the scenario above, two predictor variables are used to grow each tree, rather than using the entire set of predictor variables.

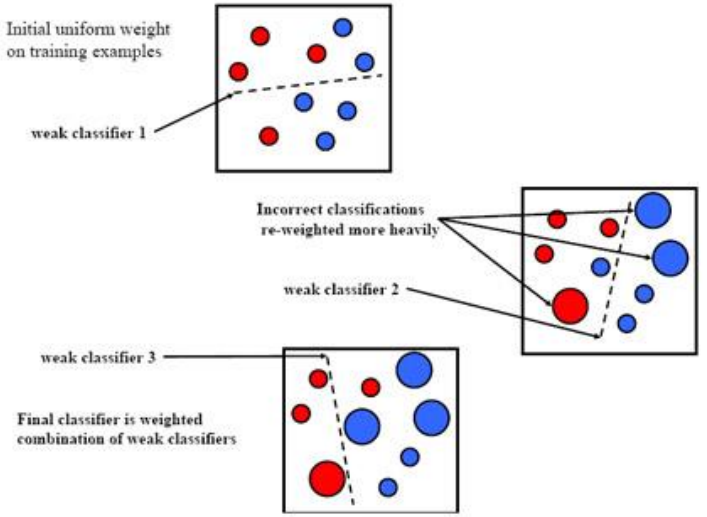
Boosting

Boosting: ML Framework



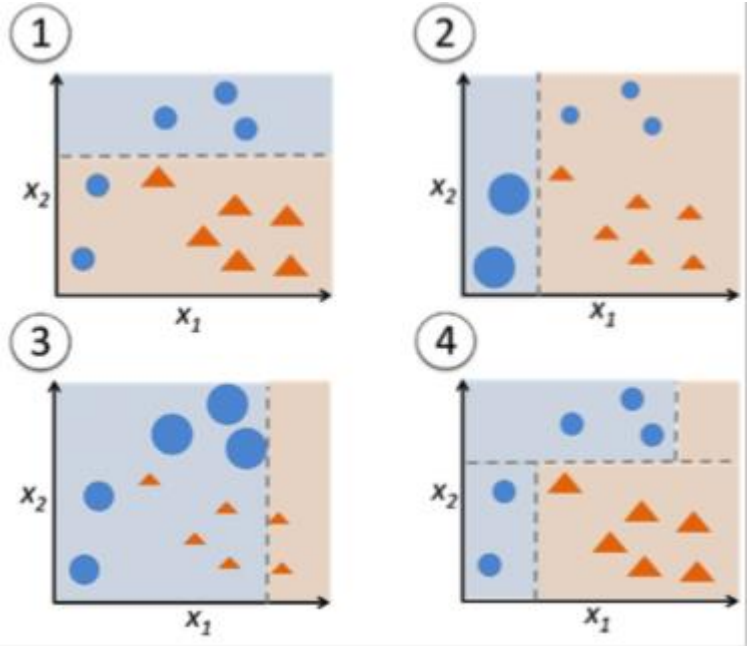
Boosting : Adaboost

- Strategic resampling
  - Model-1 is trained with a random subset of the training data.
  - Model-2 is trained with the most informative subset, **given** Model-1
    - Model-2 is trained on a training data only half of which is correctly classified by Model-1 , and the other half is misclassified.
  - Model-3 is trained with instances on which Model-1 & Model-2 disagree.
  - The three classifiers are combined through a three-way majority vote.



$$H(x) = sign(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

- Series Not Parallel
  - Train a series of classifiers
  - Each classifier focusses on (gives higher weightage to) instances that the previous ones got wrong
  - Then, use a weighted average of predictions



Boosted Trees

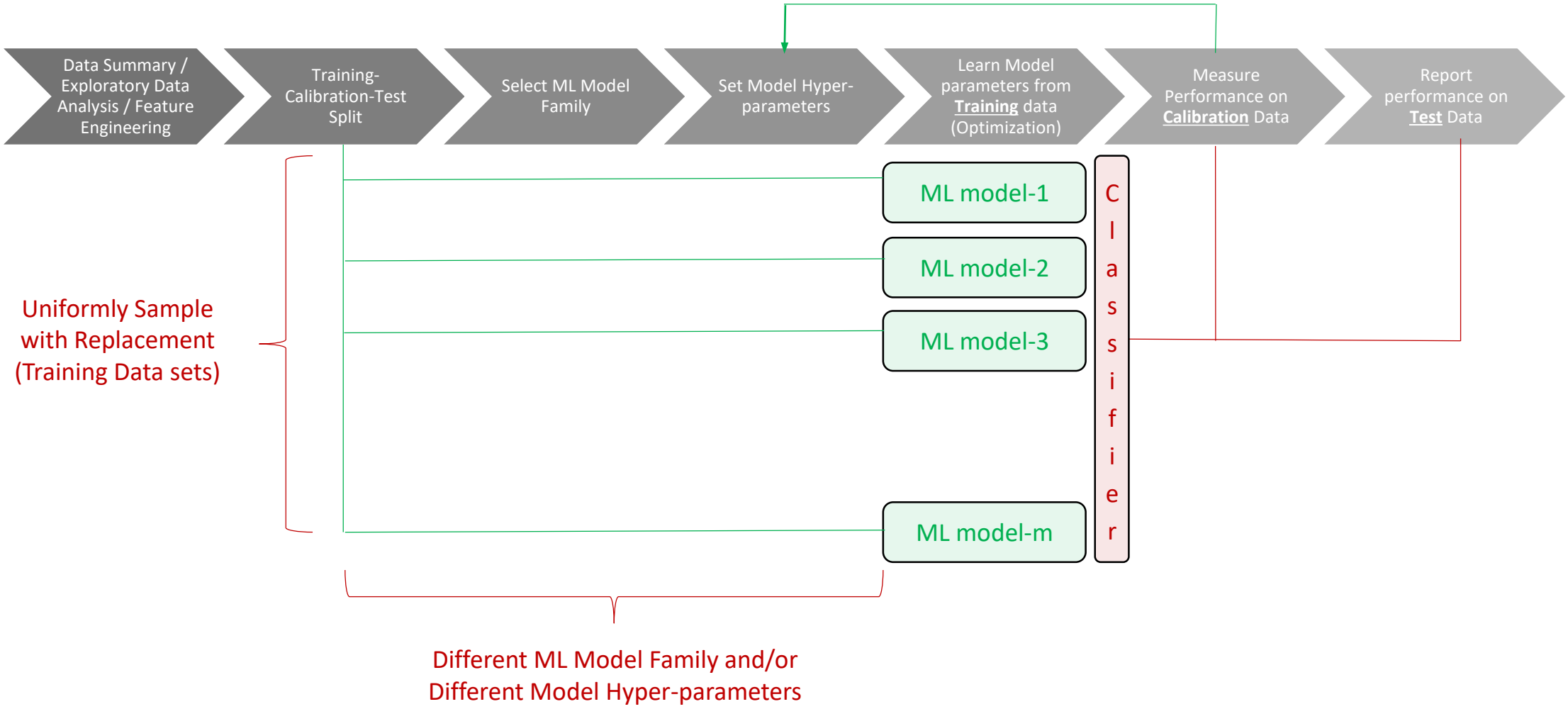
- Key !dea
  - “Combine” multiple weak learners to create a strong one
  - Train multiple models on different subsets (~ bagging)
  - Subsets : Higher weight to “harder” samples (Iterative)

- Boosting Regression Trees
  - Tree-1 is trained with a random subset of the available training data.
  - Tree-2 is trained on the residuals (instead of y) of one of the leaf nodes of Tree-1
    - This effectively gives more weight to observations which were misclassified by Tree-1
  - Tree-3 is trained on the residuals (instead of y) of one of the leaf nodes of Tree-1
  - The three classifiers are combined through a three-way majority vote.

- Advantages
  - Dedicate models to harder samples.
  - But Not easy to parallelize

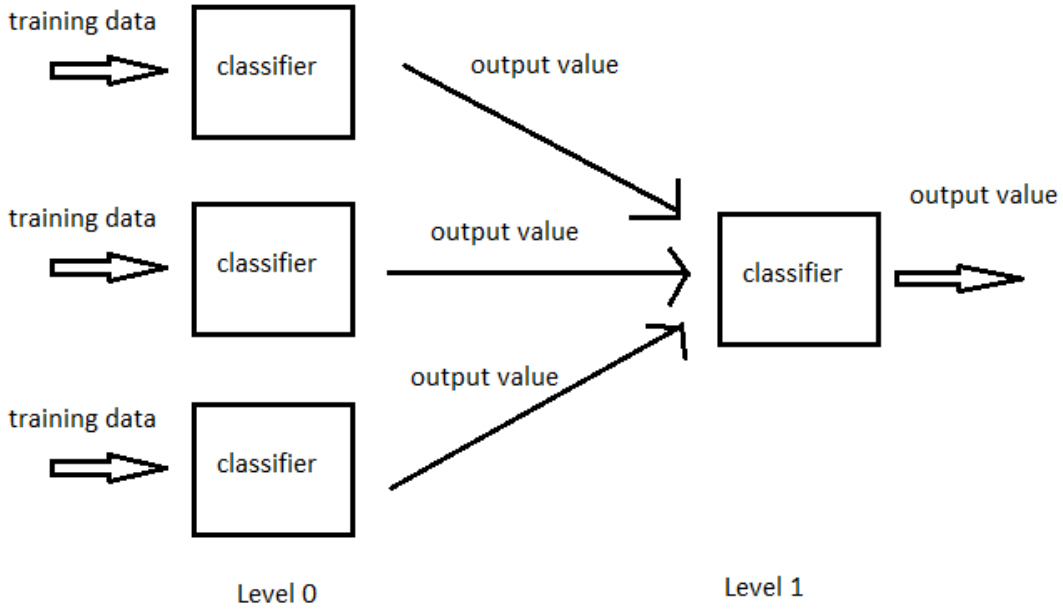
Stacking

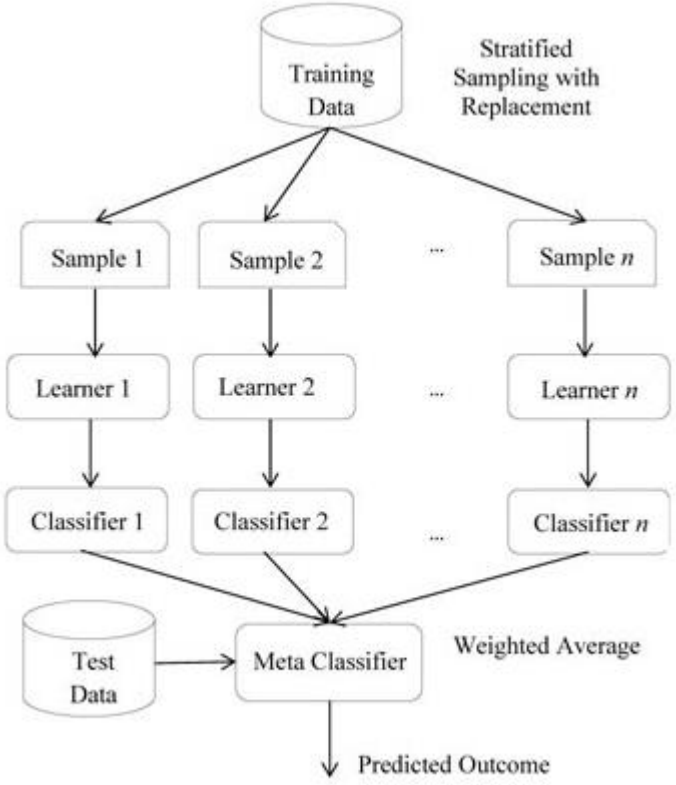
Stacking: ML Framework



Stacking : Stacked Generalization : Blending

- Learn the combiner
  - Combiner: output of Level 0 classifiers will be used as training data for another classifier
  - Level 1 classifier “learns” the combining mechanism.
  - Combiner : A Meta-Classifier which takes as input the output of other classifiers to make a prediction





Ensemble Methods: Summary

- ***"Why build 1 when you can have 2 at twice the price?"***
  - Improve model accuracy
  - Reduce variance (Bias-Variance tradeoff)

- Building "different" models
  - Bagging: Uniformly draw samples from training data with replacement
  - Random (Feature) Subspaces : Build models on subsets of feature on different subsets of training data
  - Boosting : A series of classifiers, each focusing on observations hard to classify by previous classifiers
  - Stacking: Learn the combination rule (instead of voting / mean)

- Ensembling
  - Can be used with any model family / algorithms
  - Often leads to higher accuracy and / or lower variance