Modul 3

Supervised Part 3

Data Science Program

Purwadhika
Startup and Coding School

Outline

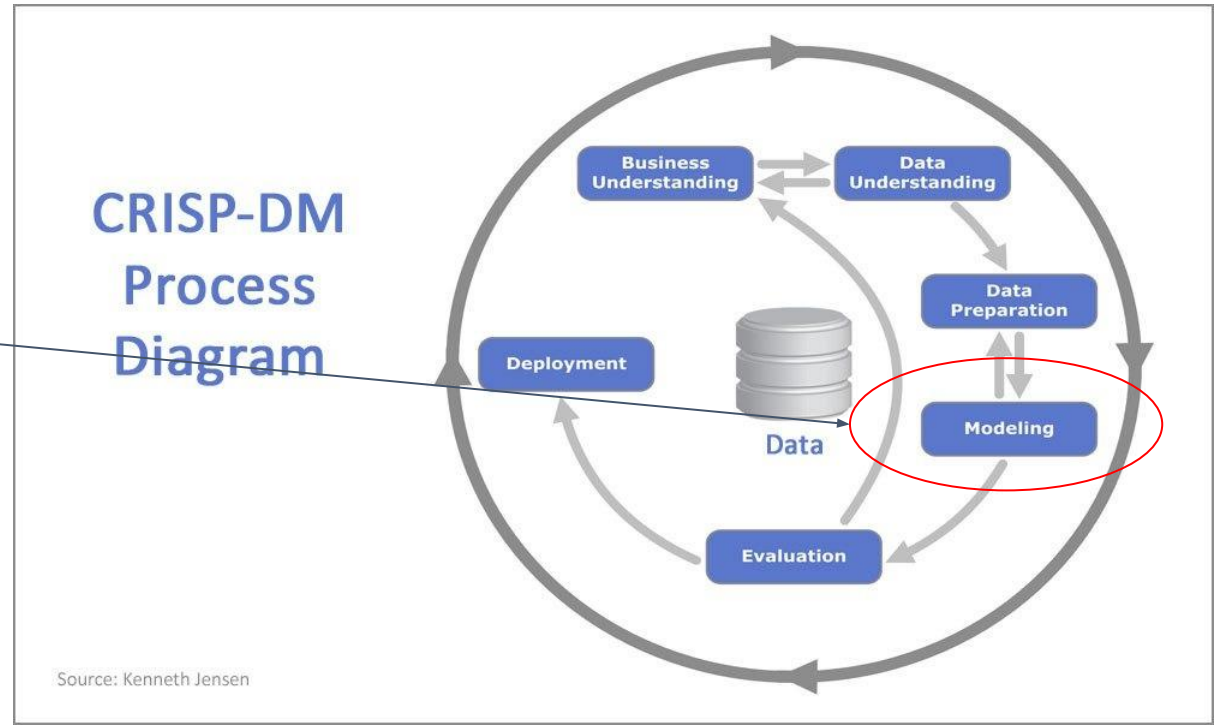
What is Ensemble ?

Combine multiple models of various type

- voting classification
- stacking

Combine multiple models of similar type

- bagging
- boosting



Purwadhika
Startup and Coding School

What is Ensemble ?

What is Ensemble ?

- Combining several prediction result from different machine learning algorithm to make more robust system

ID	RegLog	DT	KNN	RF	N-Net	Result
1	1	1	1	1	0	1
2	1	0	0	1	1	1
3	1	0	0	0	0	0
4	0	0	1	0	0	0
5	1	1	1	1	1	1
….	…	…	…	…	…	…

- Obtained from majority vote (*one of ensemble method)
- more robust result

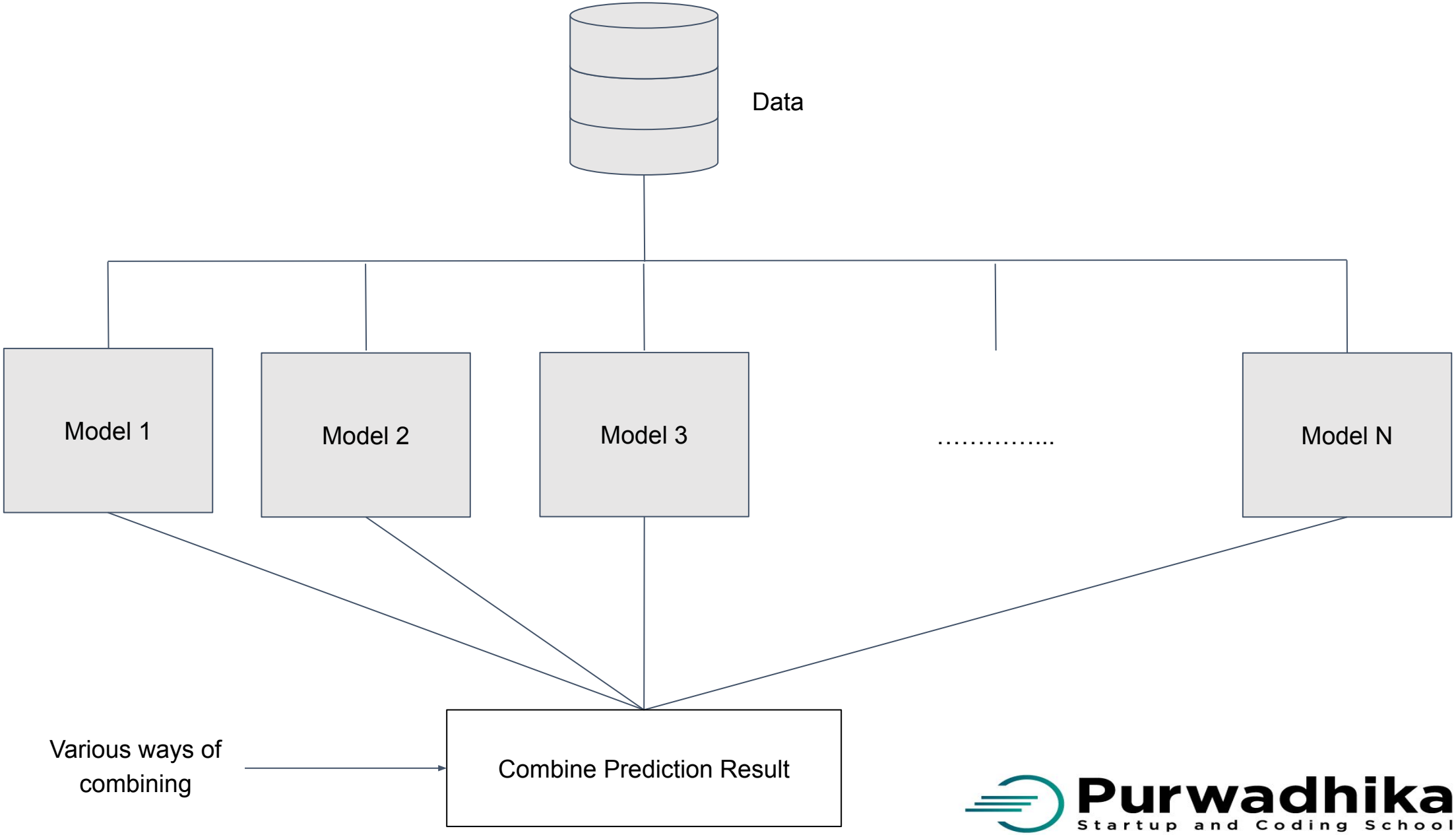
After the modeling we obtained that the accuracy:

- Reglog : 80%
- DT : 78%
- KNN : 83%
- RF : 84%
- Neural Network : 84%

What should we do to combine those method so that the model performance will improve ?

- Answer : Ensemble Method
- The performance expected to increase

Purwadhika
Startup and Coding School



Purwadhika
Startup and Coding School

What is Main Challenge for Developing Ensemble Model ?

- Not to obtain highly accurate base learner, but rather to obtain uncorrelated based model
- High accuracy can be accomplished if differential base model misclassify different training examples, even if the base classifier performance is low

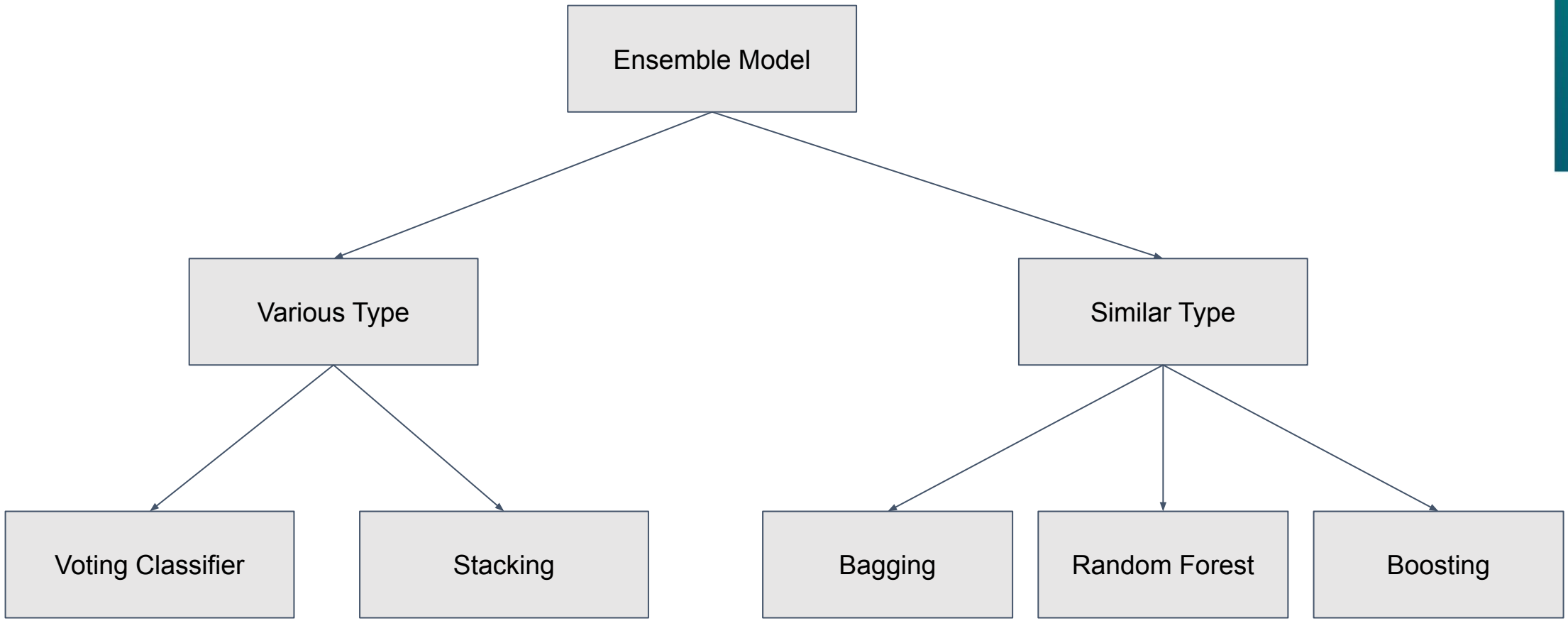
Analogy

- Imagine a meeting room with experts from different background discussing company stock. Everyone could contribute opinion or suggestion based on their individual point of view. The opinion will be varied.
- Taking account the opinion as input will result more robust final decision, more accurate and less likely to be biased.

Why do We Need an Ensemble Model ?

- One way that can be achieved to increase the accuracy of predictions and to obtain more robust result
- No algorithm that can be always accurate
- Ensemble model gives the global picture

Ensemble Type



Purwadhika
Startup and Coding School

Various Type

Purwadhika
Startup and Coding School

Various Type Ensemble

Combine multiple model which has various type :

- Voting Classification
- Stacking

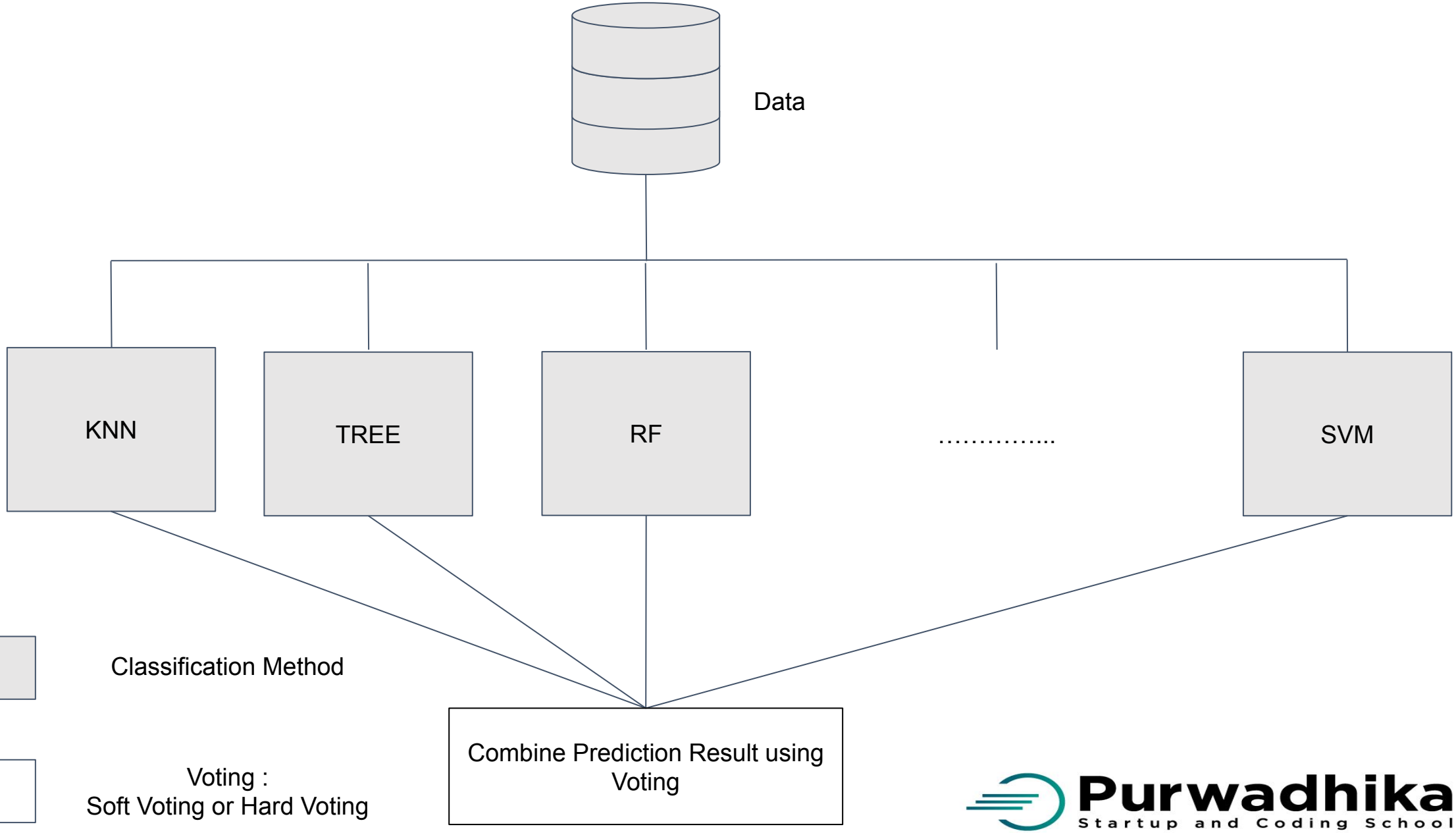
Voting Classification

- Create multiple stand-alone models from training dataset
- Wrap the model using voting classifier
- combine the prediction through majority voting as the final result of prediction

ID	RegLog	DT	KNN	RF	N-Net	Result
1	1	1	1	1	0	1
2	1	0	0	1	1	1
3	1	0	0	0	0	0
4	0	0	1	0	0	0
5	1	1	1	1	1	1
….	…	…	…	…	…	…

- Obtained from majority vote (*hard voting)
- more robust result

Purwadhika
Startup and Coding School



Purwadhika
Startup and Coding School

Hard Voting

ID	RegLog	DT	KNN	RF	N-Net	Result
1	1	1	1	1	0	1
2	1	0	0	1	1	1
3	1	0	0	0	0	0
4	0	0	1	0	0	0
5	1	1	1	1	1	1
….	…	…	…	…	…	…

- hard voting
- combine the class directly
- more robust result

Purwadhika
Startup and Coding School

Soft Voting

ID	RegLog	DT	KNN	RF	N-Net	Result	Class (>0.5)
1	0.87	0.77	0.45	0.61	0.30	0.6	1
2	0.77	0.38	0.78	0.69	0.71	0.666	1
3	0.65	0.40	0.3	0.48	0.32	0.43	0
4	0.30	0.45	0.6	0.45	0.32	0.424	0
5	0.67	0.87	0.70	0.61	0.88	0.746	1
….	…	…	…	…	…	…	…

- soft voting
- combine the score
- more robust result

Purwadhika
Startup and Coding School

Python Exercise : Voting Classifier

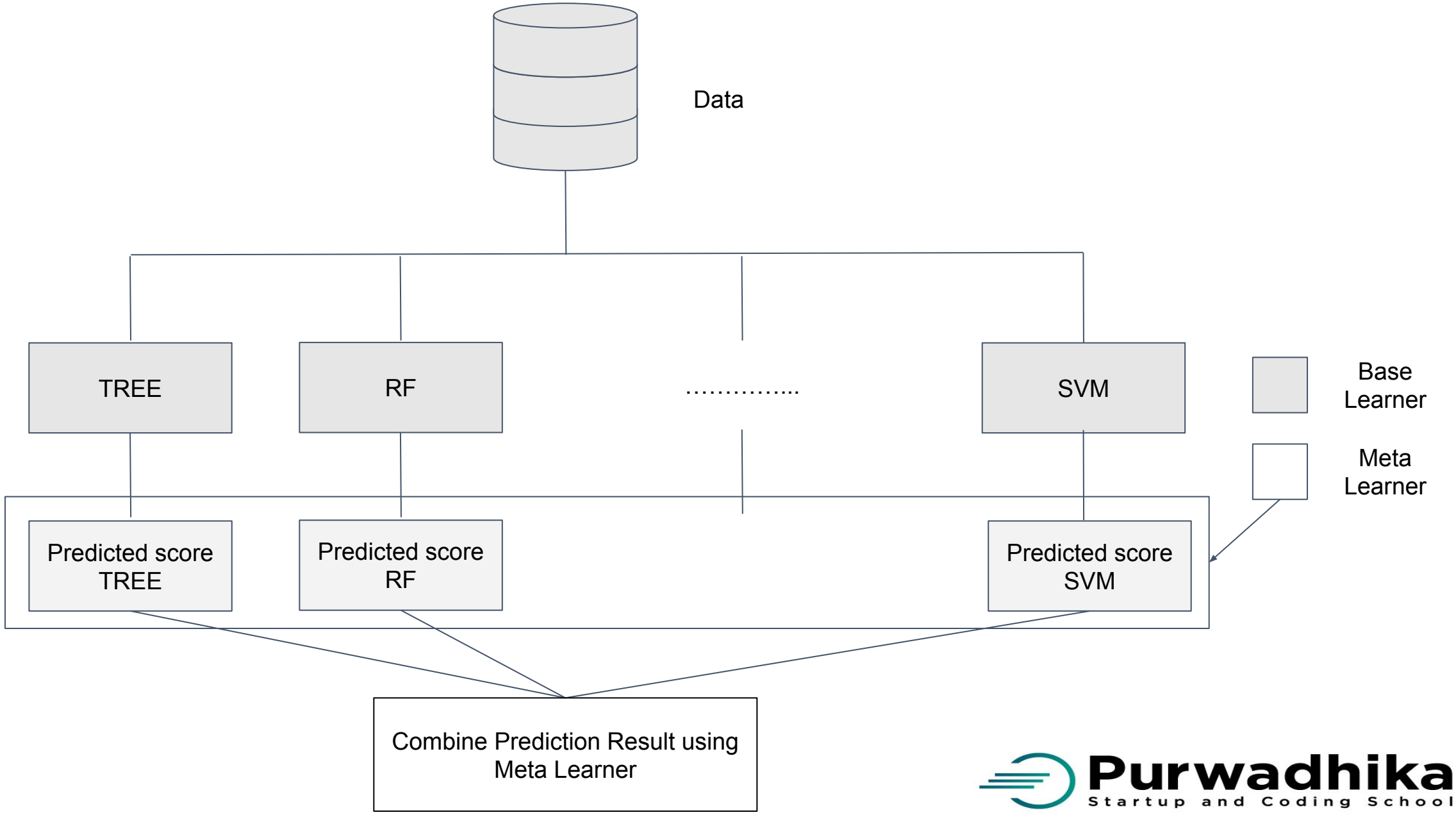
Analyze data white_wine.csv

- Apply Voting Classifier
  - target : quality (quality >6 → Y = 1)
  - features : density alcohol
- Validate the model using precision, recall and f1 score in 20% testing data
- Apply soft voting classifier method, using these following method:
  - logistic regression
  - decision tree : max depth 5
  - knn : nearest neighbor 3
- Apply soft voting classifier method, using these following method
  - 3rd degree polynomial features + logistic regression
  - decision tree : max depth 5
  - standard scaler + knn : nearest neighbor 3

Purwadhika
Startup and Coding School

Stacking

- Create multiple stand-alone models (base learners) from training dataset
- Wrap the model using new model (a meta learner) with predicted value from base learners as the features
- Final result obtained from meta learner



Purwadhika
Startup and Coding School

Advantages and Disadvantages

Advantages:

- Proven method to improve prediction performance
- Model more robust and stable in most scenarios
- For those who love competition, this ingredient wins almost all competition

Disadvantages:

- Reduce model interpretability
- Time consuming and even less practical
- Base model to ensemble is hard to master

Similar Type : Tree Based Model

Purwadhika
Startup and Coding School

Tree Based Model

- Bagging
- Random Forest
- Boosting

Bootstrap Sample Illustration

Original Sample

ID	X	y
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Bootstrap Sample 1

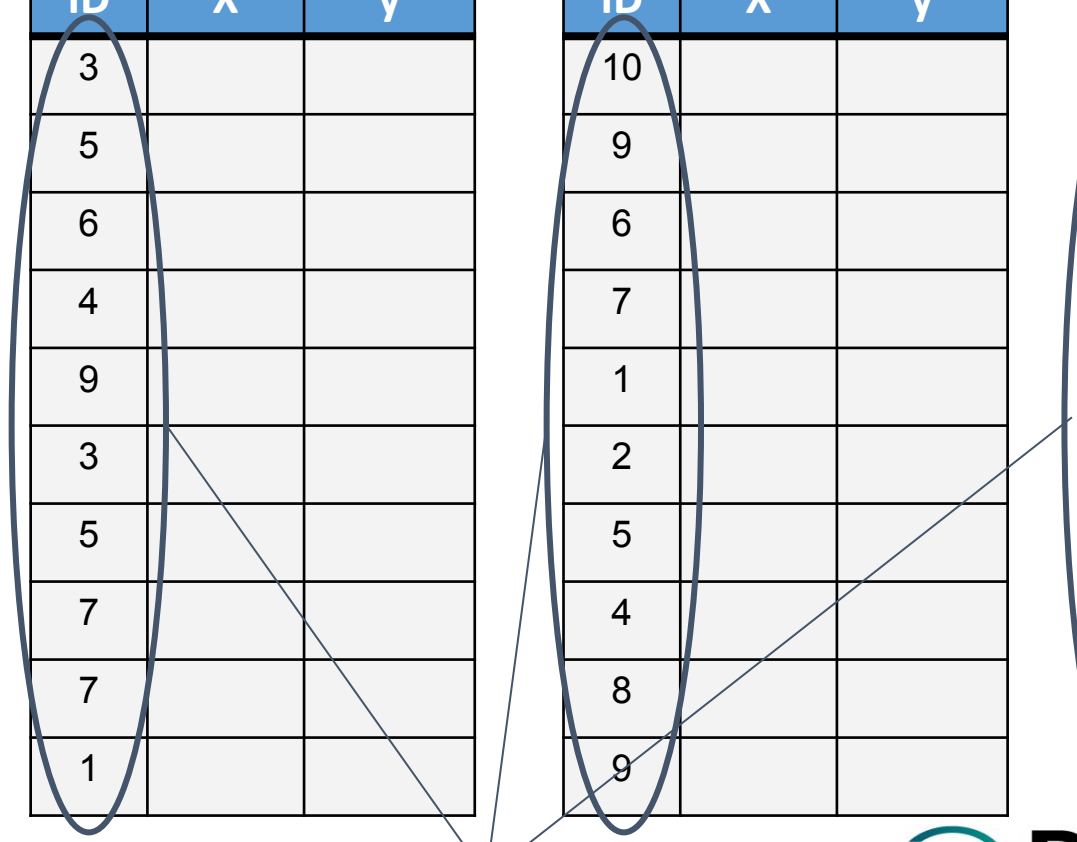
ID	X	y
3		
5		
6		
4		
9		
3		
5		
7		
7		
1		

Bootstrap Sample 2

ID	X	y
10		
9		
6		
7		
1		
2		
5		
4		
8		
9		

Bootstrap Sample 3

ID	X	y
1		
3		
4		
2		
4		
7		
9		
10		
1		
3		



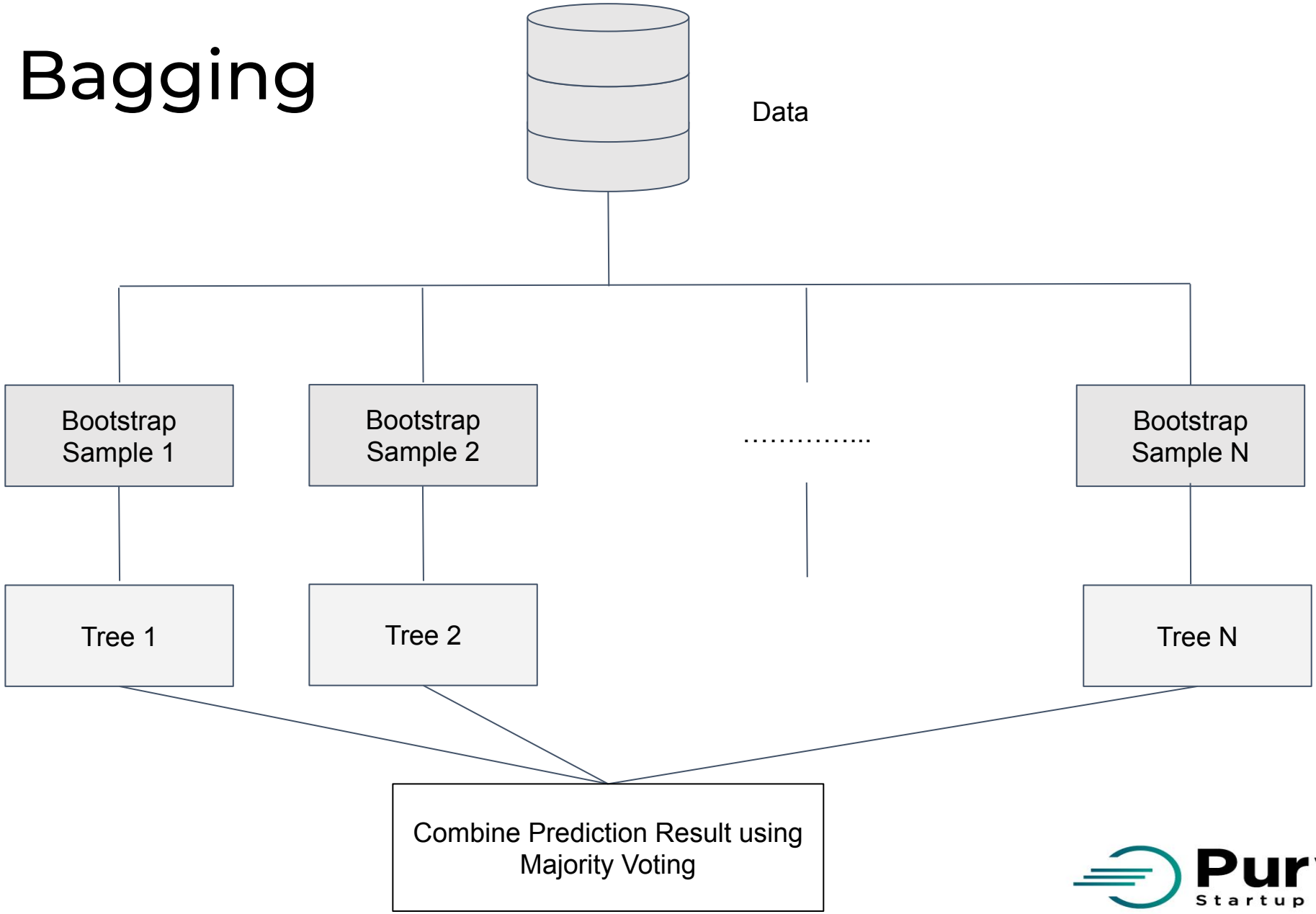
Random sampling with replacement

Purwadhika
Startup and Coding School

Bagging

- Bagging is an ensemble technique, bagging = bootstrap aggregating.
- Aim of this method is to reduce variance
- Training data is split into multiple samples with replacement (bootstrap sample)
- making a tree from each bootstrap sample
- final result is the majority vote

Bagging



Purwadhika
Startup and Coding School

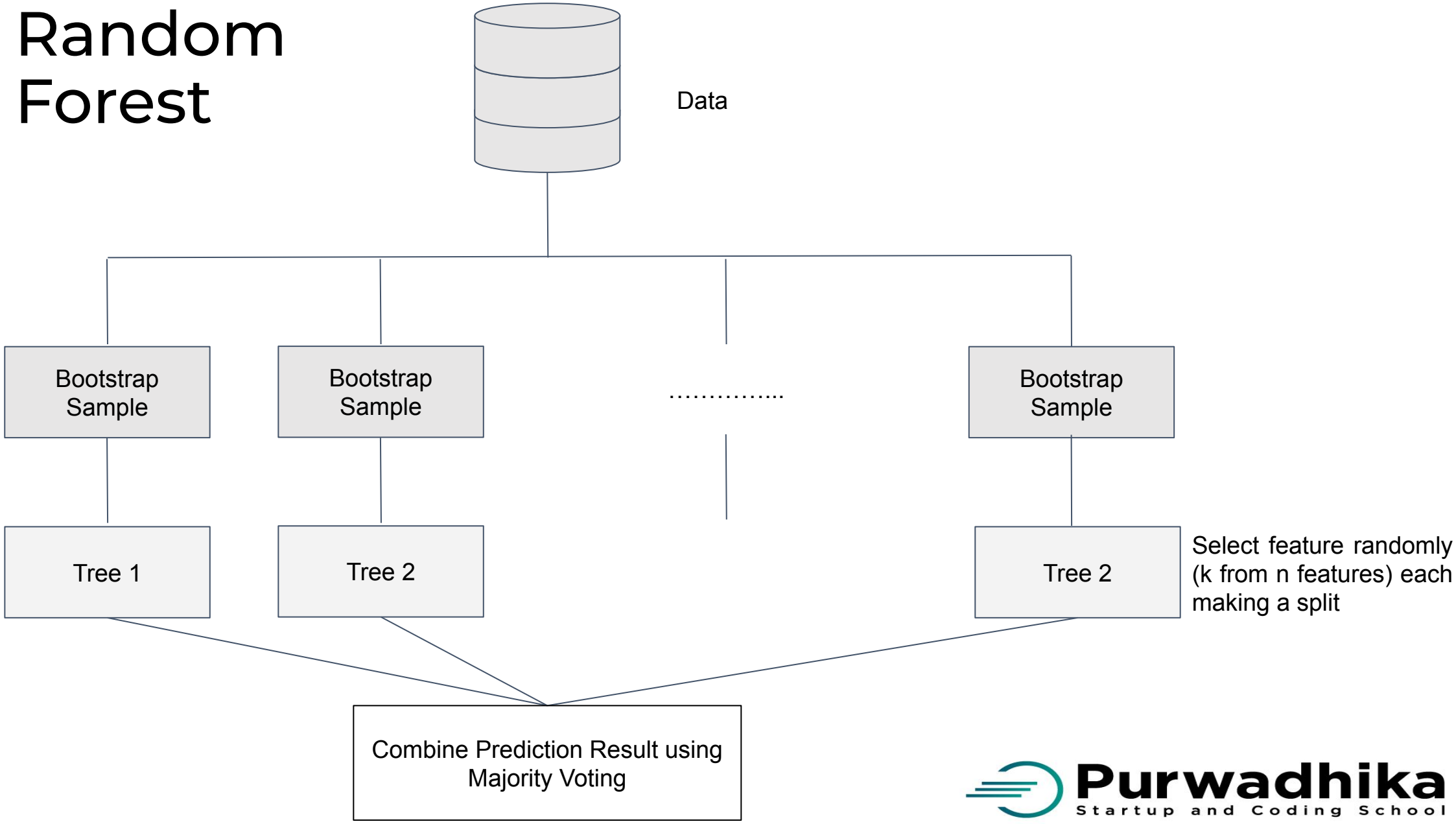
Why Do We Need Bagging ?

- Decision Tree may result in :
  - an unstable model
  - The variance is very high
  - The model tend to overfit

Random Forest

- Random Forest is an ensemble technique that the resulted from modification of bagging
- Aim of this method is to reduce variance
- Training data is split into multiple samples with replacement (bootstrap sample)
- making a tree from each bootstrap sample with feature randomization
- final result is the majority vote

Random Forest



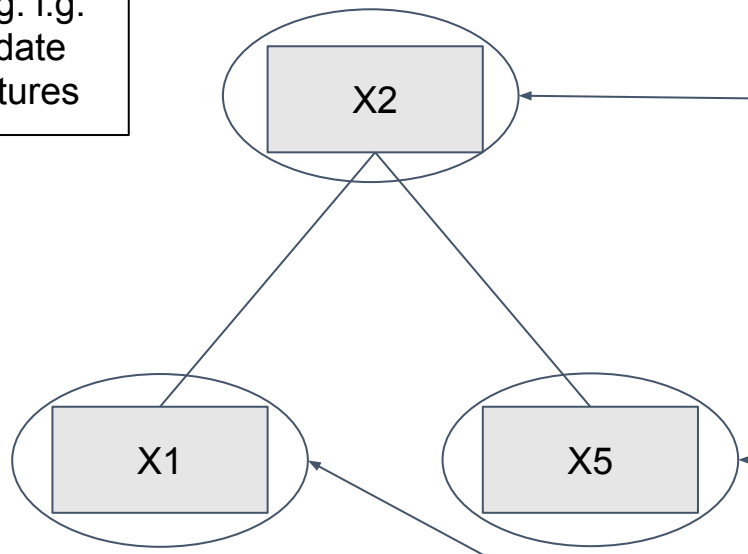
Purwadhika
Startup and Coding School

Why Do We Need Random Forest ?

- Decision Tree may result in an unstable model
- The variance is very high
- The model tend to overfit
- We need to randomized the feature because :
  - We need to obtain different tree
  - bagging may resulted in very similar tree, so each tree are correlated with another
  - there is no point combining similar tree

Illustration : Each Making Tree

In Each Splitting: i.g. select 3 candidate splitter of 8 features



Var	X1	X2	X3	X4	X5	X6	X7	X8
Gini		0.2	0.3			0.4		

Var	X1	X2	X3	X4	X5	X6	X7	X8
Gini	0.2				0.1			0.49

Var	X1	X2	X3	X4	X5	X6	X7	X8
Gini	0.1			0.25				0.4

Selected Feature to make splitting

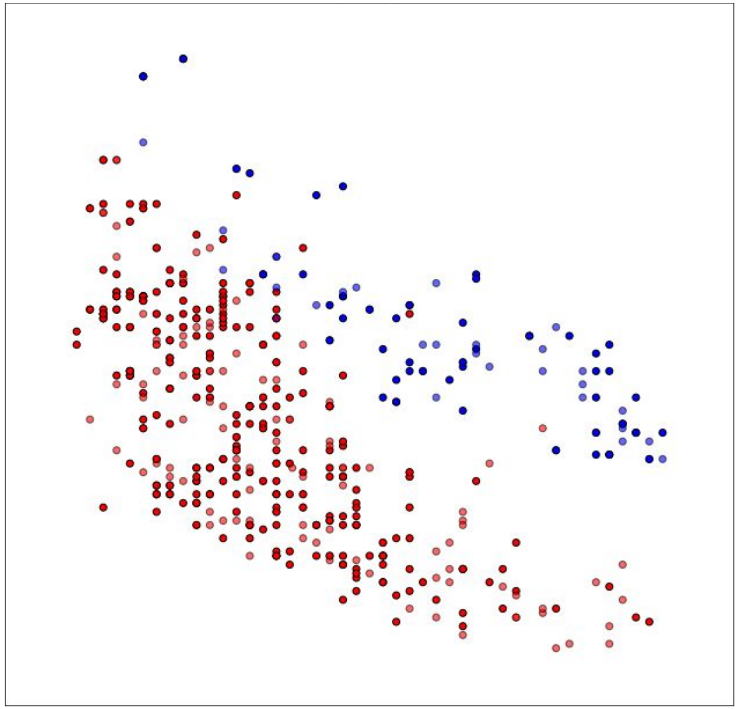
Purwadhika
Startup and Coding School

How to Optimize Random Forest ?

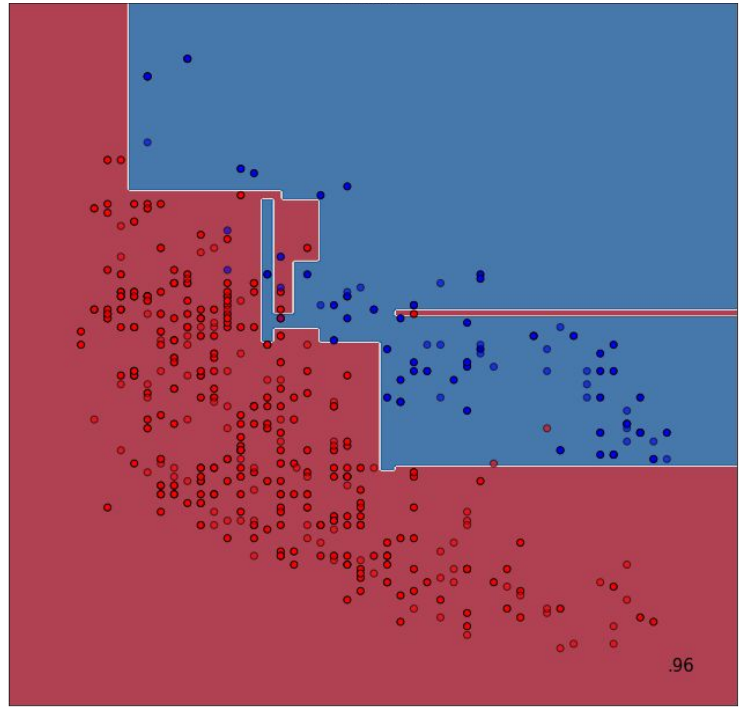
- How many trees should we make ?
- How many features are needed to select randomly ?
- Decision Tree Hyperparameters

Decision Boundary : Tree vs Random Forest

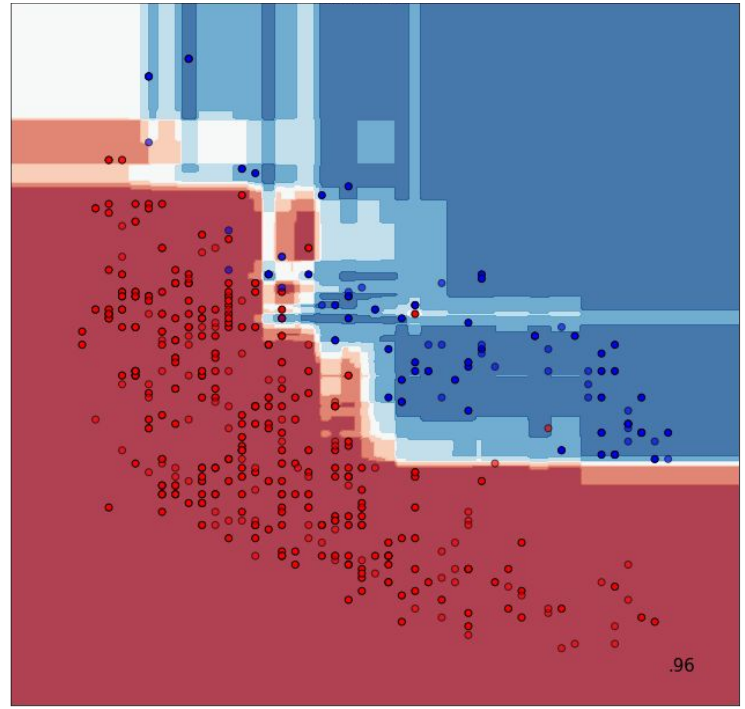
Input



Decision Tree



Random Forest



Purwadhika
Startup and Coding School

Advantages and Disadvantages

Advantages:

- Handle high dimensionality data and results Importance of variable, a handy features
- Has methods of balancing error in data sets where classes are imbalanced
- Data not used for training during bootstrapping is used for testing data, called out-of-bag samples. Error estimated on these data is accurate.

Disadvantages:

- less interpretable
- It works better for classification, but not for regression

Python Exercise : Random Forest

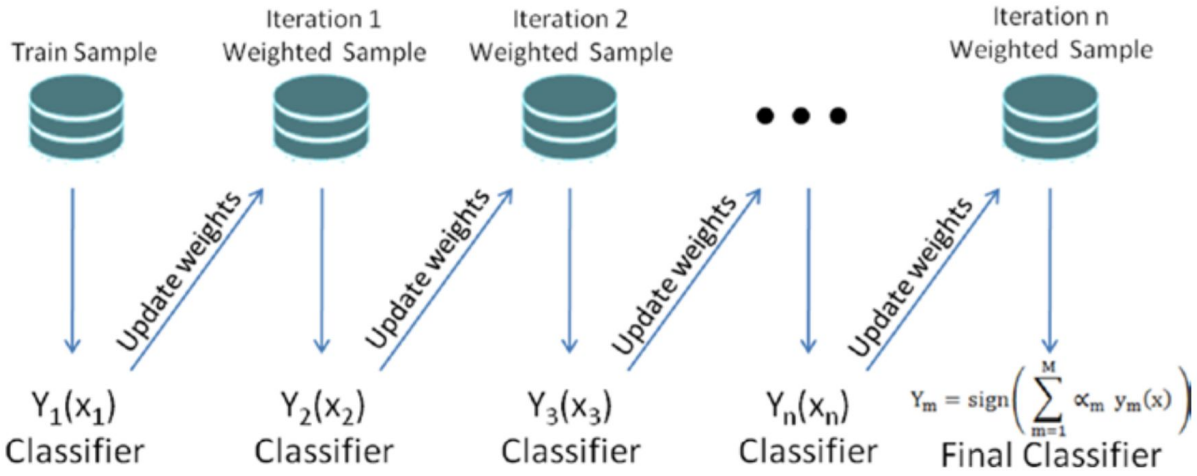
Analyze data bankloan.csv

- Apply Random Forest:
  - target : default
  - features : age, employ, employ, debtinc, creddebt, othdebt
- Splitting ratio 80:20 stratified
- Apply Random Forest n_estimator 20 max features 4 max depth 3
- Compute precision, recall, f1 score in test set

Purwadhika
Startup and Coding School

Boosting

- Boosting is an iterative process to update a model (weak learner) by increasing the weights of the records that wrongly classified
- Transform weak learner into strong learner
- Weak learner is a model whose performance is at least slightly better than random chance
- in bagging and RF each tree is independent, in boosting tree grow sequentially



Purwadhika
Startup and Coding School

Weight Concept

Let’s say you have a data :

- 4 5 6 7

  mean of the data is 5.5

  obtained from (4 + 5 + 6 + 7)/4

  weight of each data is 1/4

- Now we want to give more weights to the first data for some reason.

  The weight is twice of another data, so it will be the same as you have data : 4 4 5 6 7

  mean of the data will be 5.2

  obtained from (4 + 4 + 5 + 6 + 7)/5

Data	4	5	6	7
Weights	1/4	1/4	1/4	1/4

Data	4	4	5	6	7
Weights	1/5	1/5	1/5	1/5	1/5

Data	4	5	6	7
Weights	2/5	1/5	1/5	1/5

Purwadhika
Startup and Coding School

The Methodology Of Boosting

Initialization step: for each example x, set

$D(x)=\frac{1}{N}$, where N is the number of examples

Iteration step (for t=1…T):

1.Find best weak classifier $h_t(x)$ using weights $D_t(x)$

2. Compute the error rate ε_t as

$$\varepsilon_t = \sum_{i=1}^{N} D(x_i).I[y_i \neq h_t(x_i)]$$

3. assign weight α_t to classifier $h_t(x)$ in the final hypothesis

$$\alpha_t = \log((1-\varepsilon_t)/\varepsilon_t)$$

4. For each x_i , $D(x_i) = D(x_i).\exp(\alpha_t.I[y_i \neq h_t(x_i)])$

5.Normalize $D(x_i)$ so that $\sum_{i=1}^{N} D(x_i) = 1$

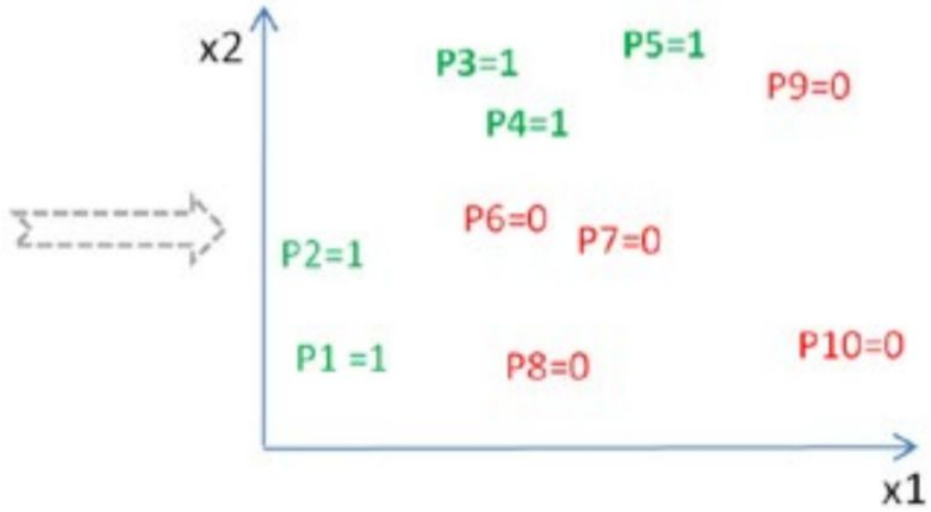
$$f_{final}(x) = sign\ [\sum \alpha_t\ h_t(x)]$$

Purwadhika
Startup and Coding School

Weight Changes illustration

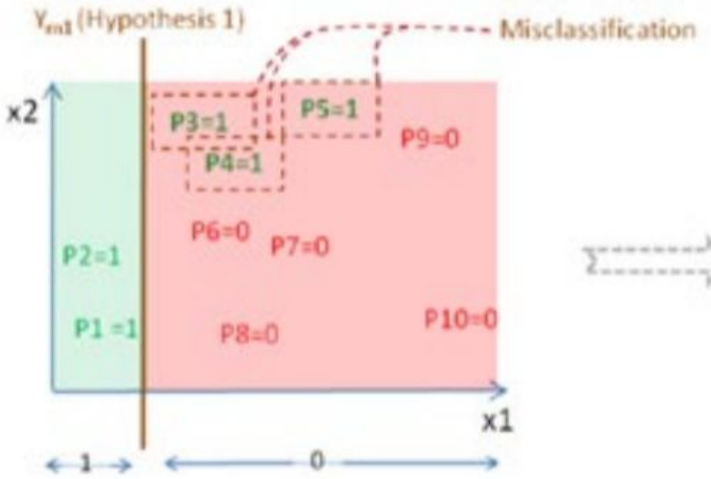
Original Sample

ID	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Class Label	1	1	1	1	1	0	0	0	0	0
Label Weights	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1



Purwadhika
Startup and Coding School

Iteration 1



$$e_{m1} = \frac{(0.1 + 0.1 + 0.1)}{3} = 0.30$$

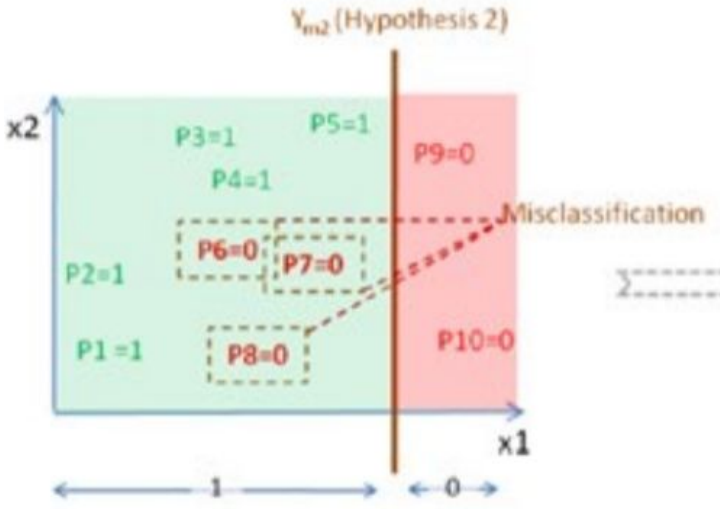
$$\alpha_{m1} = \frac{1}{2}\log\left\{\frac{1 - 0.30}{0.30}\right\} = 0.42$$

$$W_{m1} = 0.1 * e^{(0.42)} = 0.15$$

ID	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Class Label	1	1	1	1	1	0	0	0	0	0
Predicted Label	1	1	0	0	0	0	0	0	0	0
Prediction Label Weights	0.1	0.1	0.15	0.15	0.15	0.1	0.1	0.1	0.1	0.1

Purwadhika
Startup and Coding School

Iteration 2



$$\epsilon_{m2} = \frac{(0.1 + 0.1 + 0.1)}{1.15} = 0.26$$

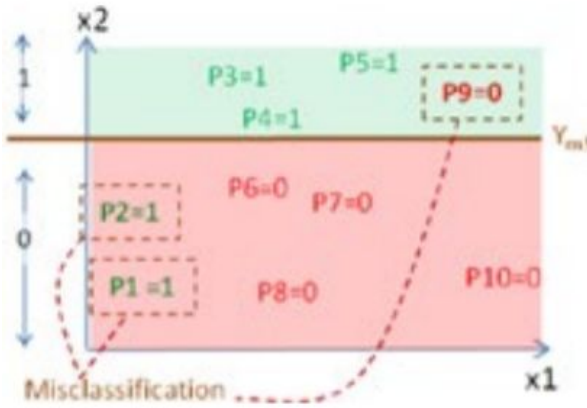
$$\alpha_{m2} = \frac{1}{2} \log\left(\frac{1 - 0.26}{0.26}\right) = 0.52$$

$$W_{m2} = 0.1 * e^{0.52} = 0.17$$

ID	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Class Label	1	1	1	1	1	0	0	0	0	0
Predicted Label	1	1	1	1	1	1	1	1	0	0
Prediction Label Weights	0.1	0.1	0.15	0.15	0.15	0.17	0.17	0.17	0.1	0.1

Purwadhika
Startup and Coding School

Iteration 3



Y_{m1} (Hypothesis 3)

$$\epsilon_{m3} = \frac{(0.1 + 0.1 + 0.1)}{1.33} = 0.23$$

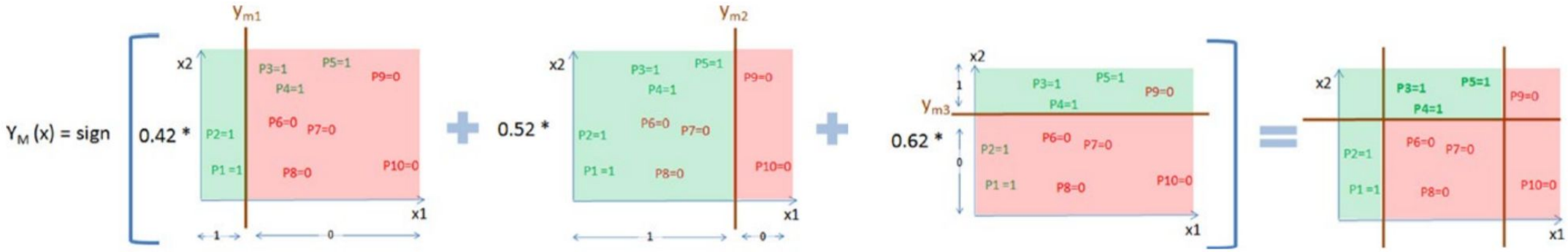
$$\alpha_{m3} = \frac{1}{2} \log\left\{\frac{1 - 0.23}{0.23}\right\} = 0.62$$

$$W_{m3} = 0.1 * e^{0.62} = 0.19$$

ID	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Class Label	1	1	1	1	1	0	0	0	0	0
Predicted Label	0	0	1	1	1	0	0	0	1	0
Prediction Label Weights	0.19	0.19	0.15	0.15	0.15	0.17	0.17	0.17	0.19	0.1

Purwadhika
Startup and Coding School

Final Model



Combine Model from each Iteration

Purwadhika
Startup and Coding School

Hyperparameters Tuning in Boosting

- Number of estimator/iteration (B)

  this can cause overfitting if set too large (depend on L)
- Shrinkage parameter (L) : control the changes in weight

  Typical value 0.01 to 0.001

  smaller L may require larger value of B to achieve good performance

  too small cause longer time to train model, too big may resulted in a non-optimal result
- Tree parameters to control depth : max_depth, min_samples_split, min_samples_leaf

Some Boosting Method

These method are further modification of boosting method

- Gradient Boosting
- Extreme Gradient Boosting
- Light Gradient Boosting
- CatBoost

Extreme Gradient Boosting is well known by its better performance and shorter training time compared to another boosting method because its parallel processing

Apply Several Preprocessing Method to Modeling at once and do hyperparameter Tuning Model Boosting

data : adult.csv

target : income

preprocess:

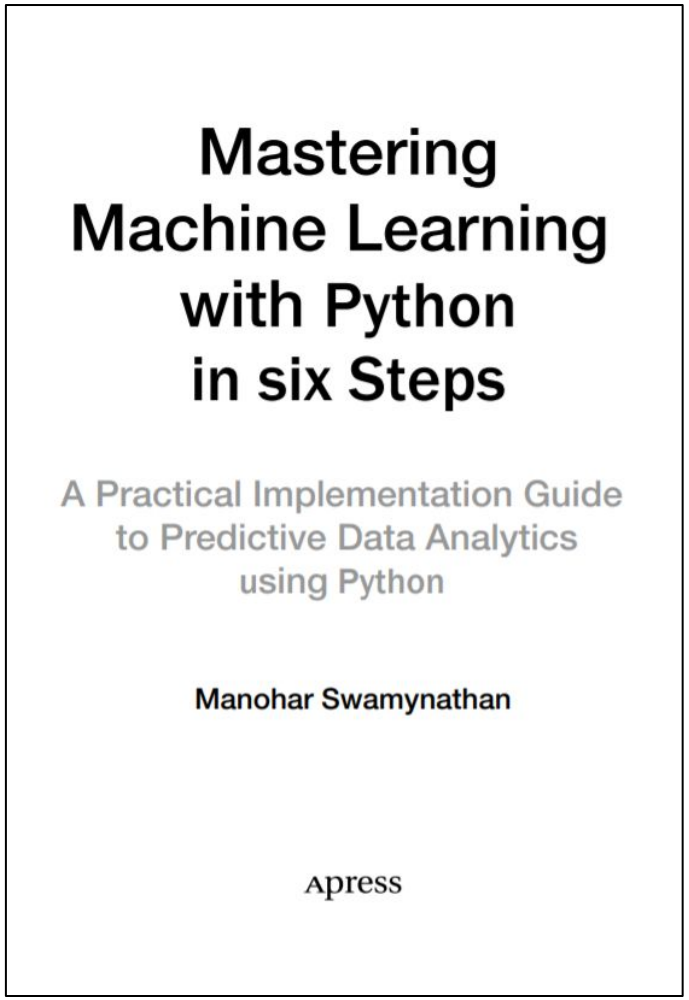
1. missing value : simple imputer with constant
2. one hot encoding : relationship, race, sex
3. binary encoding : workclass, marital status, occupation, native country
4. ordinal encoding : education (already encoded)
5. no treatment : numerical
6. out : fnlwgt

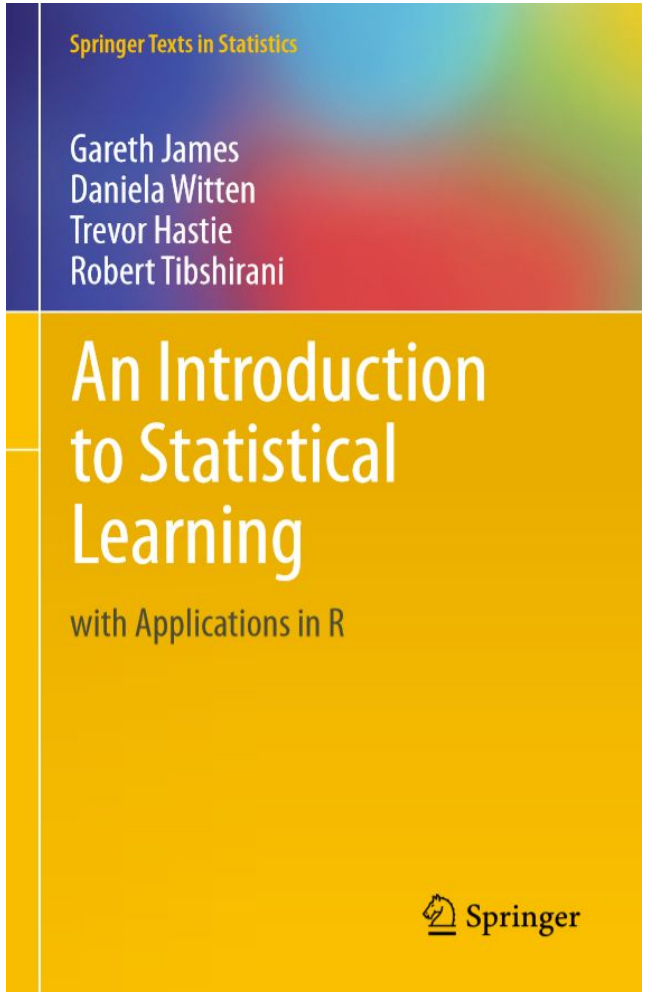
Apply Several Preprocessing Method to Modeling at once and do hyperparameter Tuning Model Boosting

Random state 10, data splitting 70:30 stratify

1. tree : model tree(max depth 3) and compute precision, recall, f1 in test set then compute the feature importances and show the tree
2. adaboost : tree adaptive boosting(max depth 3, n estimator(B) 200, learning rate 0.1) and compute precision, recall, f1 in test set then compute the feature importances
3. gbc : gradient boosting(max depth 3, n estimator(B) 200, learning rate 0.1) and compute precision, recall, f1 in test set then compute the feature importances
4. xgb :extreme gradient boosting(max depth 3, n estimator(B) 200, learning rate 0.1) and compute precision, recall, f1 in test set then compute the feature importances
5. Model selection using grid search (tree, tree adaboost, gbc, xgb) optimized by f1 and using stratified CV 5 fold
6. Hyperparameter tuning tree adaboost (optimize B L max depth) optimized by f1 and using stratified CV 5 fold
7. Evaluate the tuned model in test set precision, recall, f1 score ROC, PRC and compare the result (before after)

References





Purwadhika
Startup and Coding School

References

https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

Purwadhika
Startup and Coding School