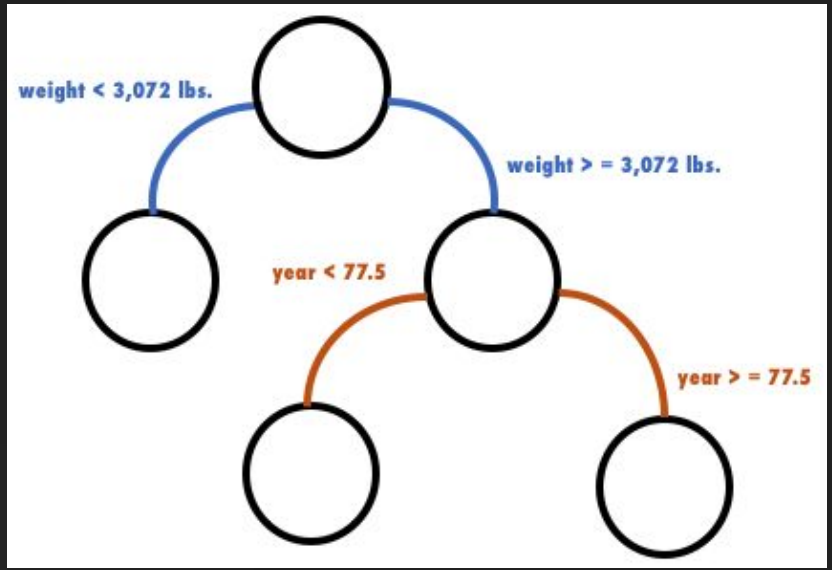

Strive
SCHOOL

/ Tree Models

Tree models

/ Tree models are based on if-else decisions. This kind of models needs less preprocessing that models based on multiplications. Tree models are:

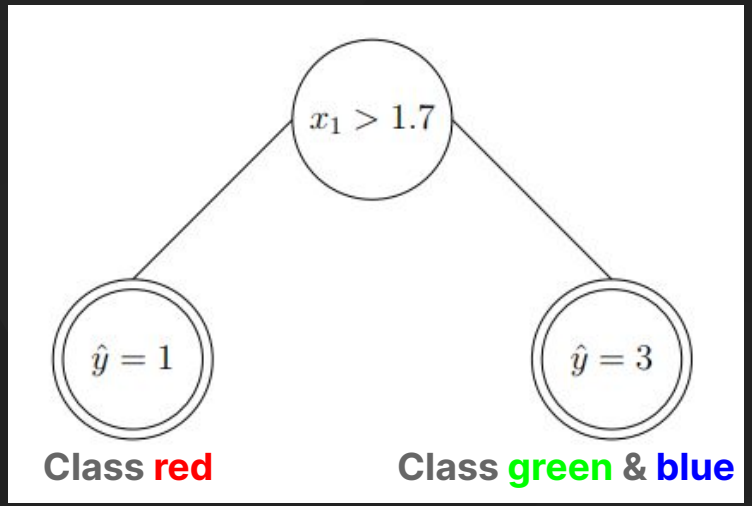
- Decision trees
- Random Forest
- Extremely Randomized Trees
- Adaboost
- Gradient Boosting
  - XGBoost
  - LightGBM
  - Catboost

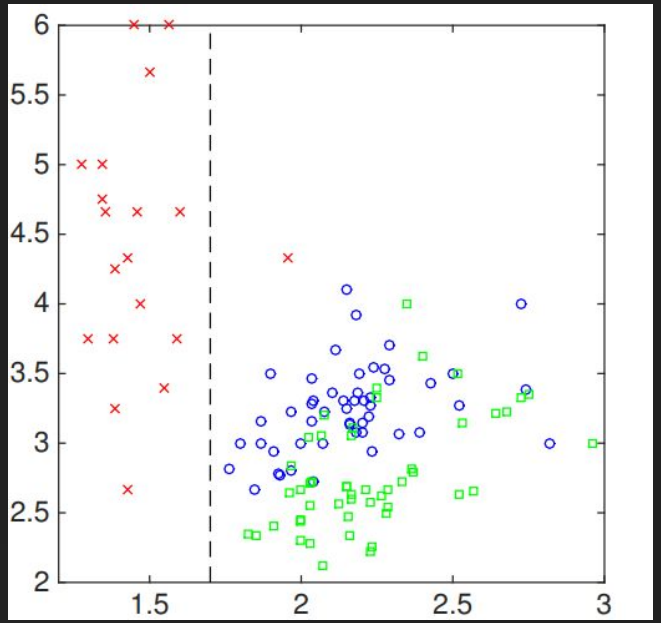


Decision Tree
For classification

```
from sklearn.tree import DecisionTreeClassifier
```

/ Tree models splits the “variable space” in regions (like boxes).

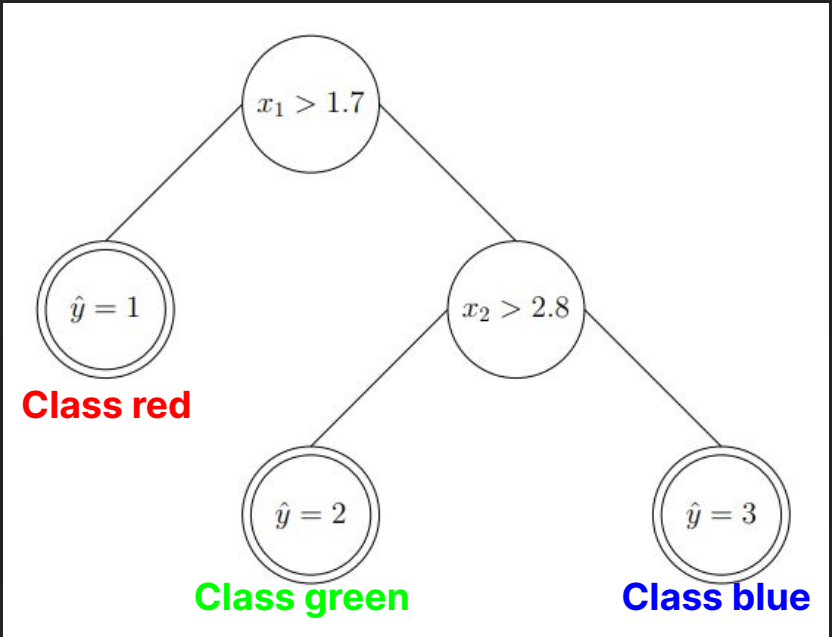


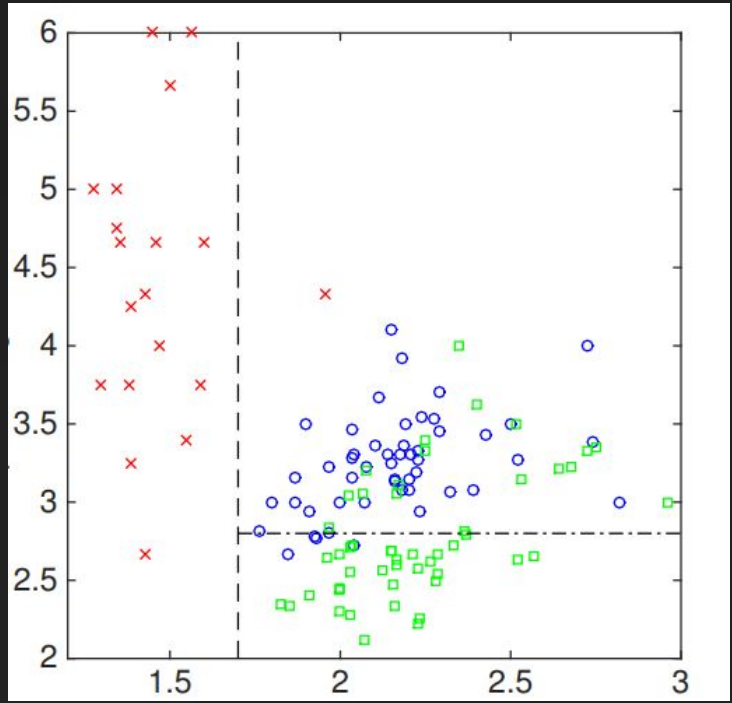


Decision Tree
For classification

```
from sklearn.tree import DecisionTreeClassifier
```

/ Tree models splits the “variable space” in regions (like boxes).

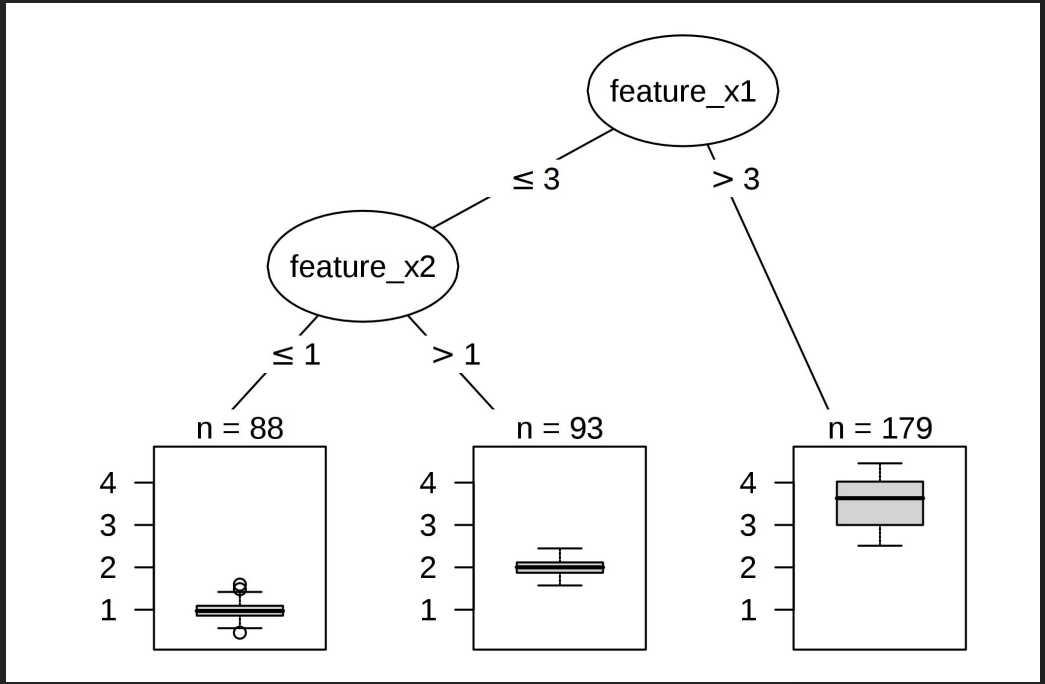




Decision Tree
For regression

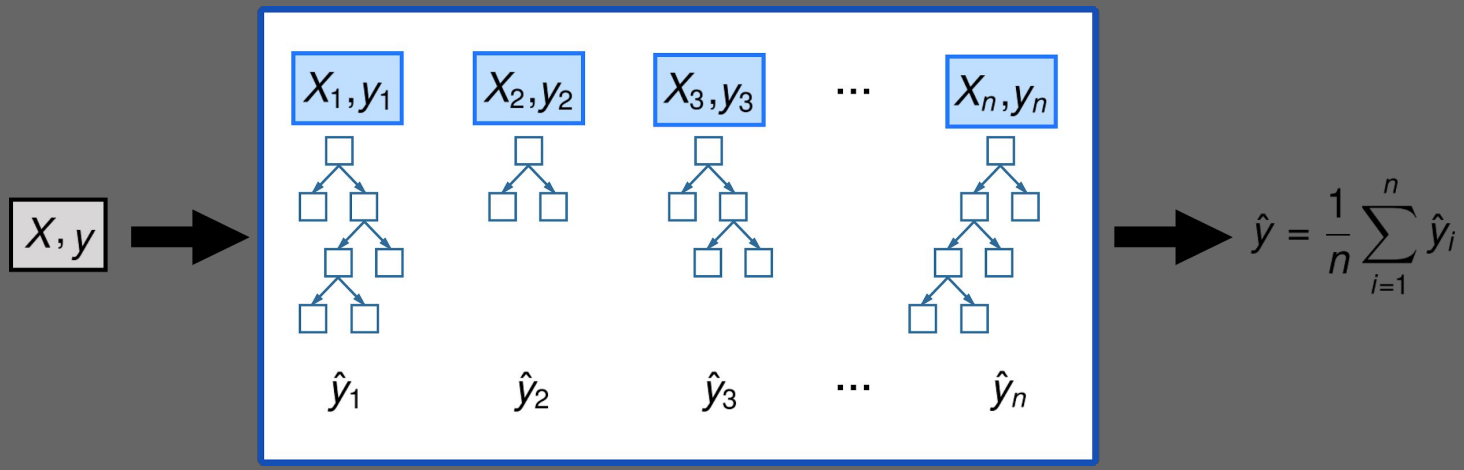
```
from sklearn.tree import DecisionTreeRegressor
```

/ Each leaf of the tree is a fixed value (not a class)



Random Forest (RF)
& Extremely Randomized Trees

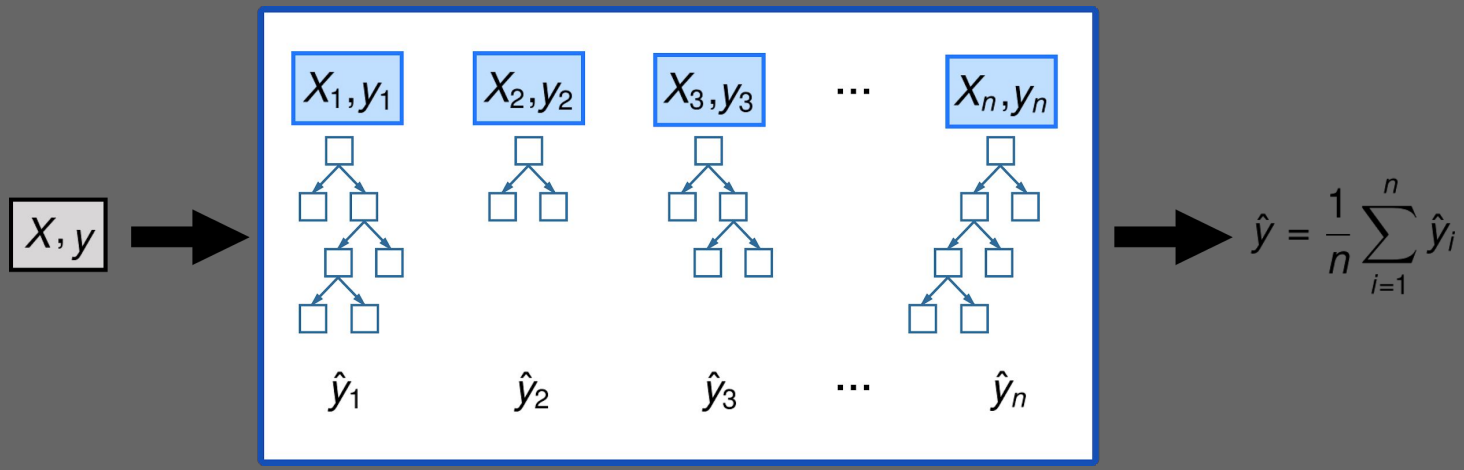
/ Each tree is constructed with a random subset of the training data (by picking not all the variables and not all the samples). The final predictions is an average of all the trees.



Random Forest (RF)

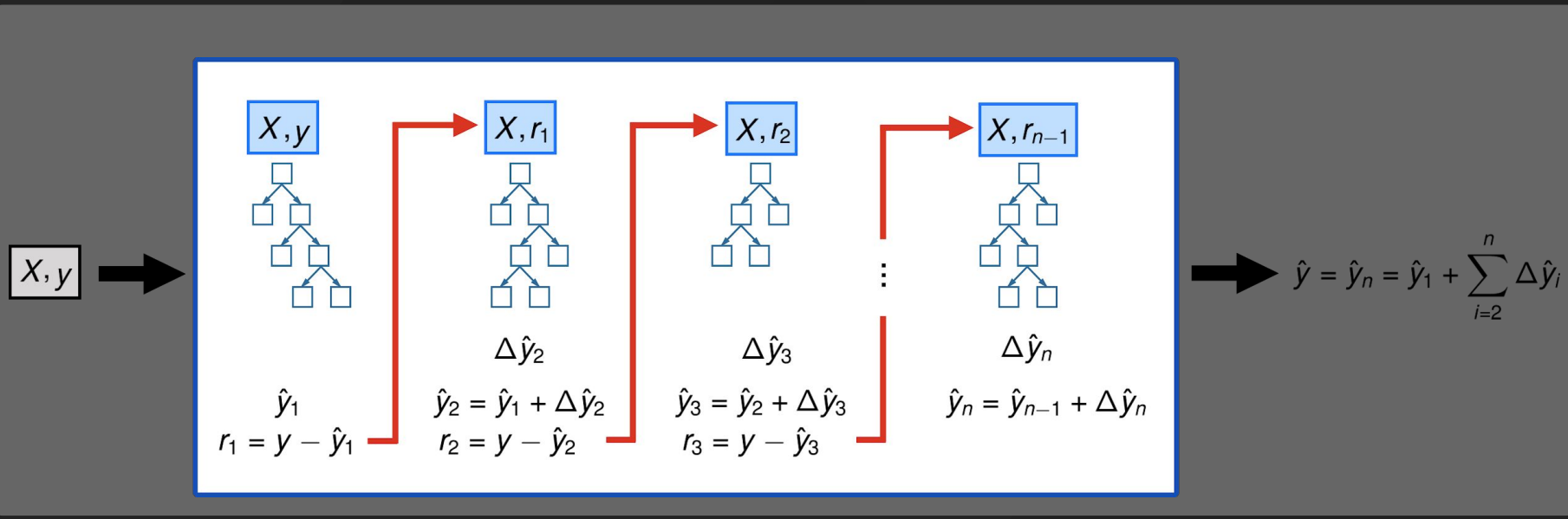
& Extremely Randomized Trees

```
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor
```



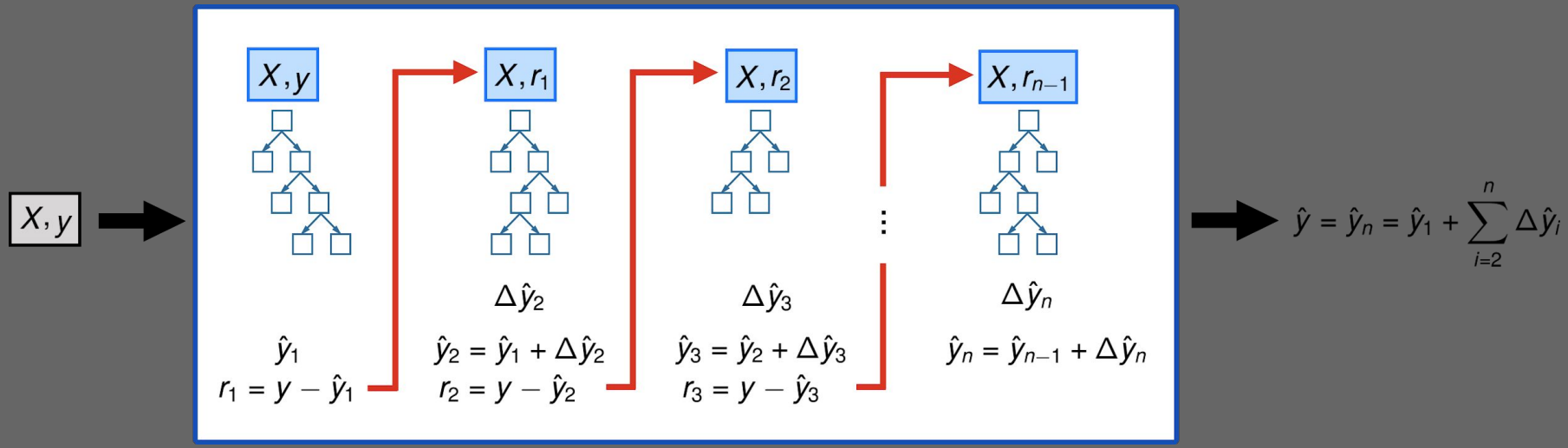
Gradient Boosting (GBM)

/ Every new tree improves the error of the previous trees.



Gradient Boosting (GBM)

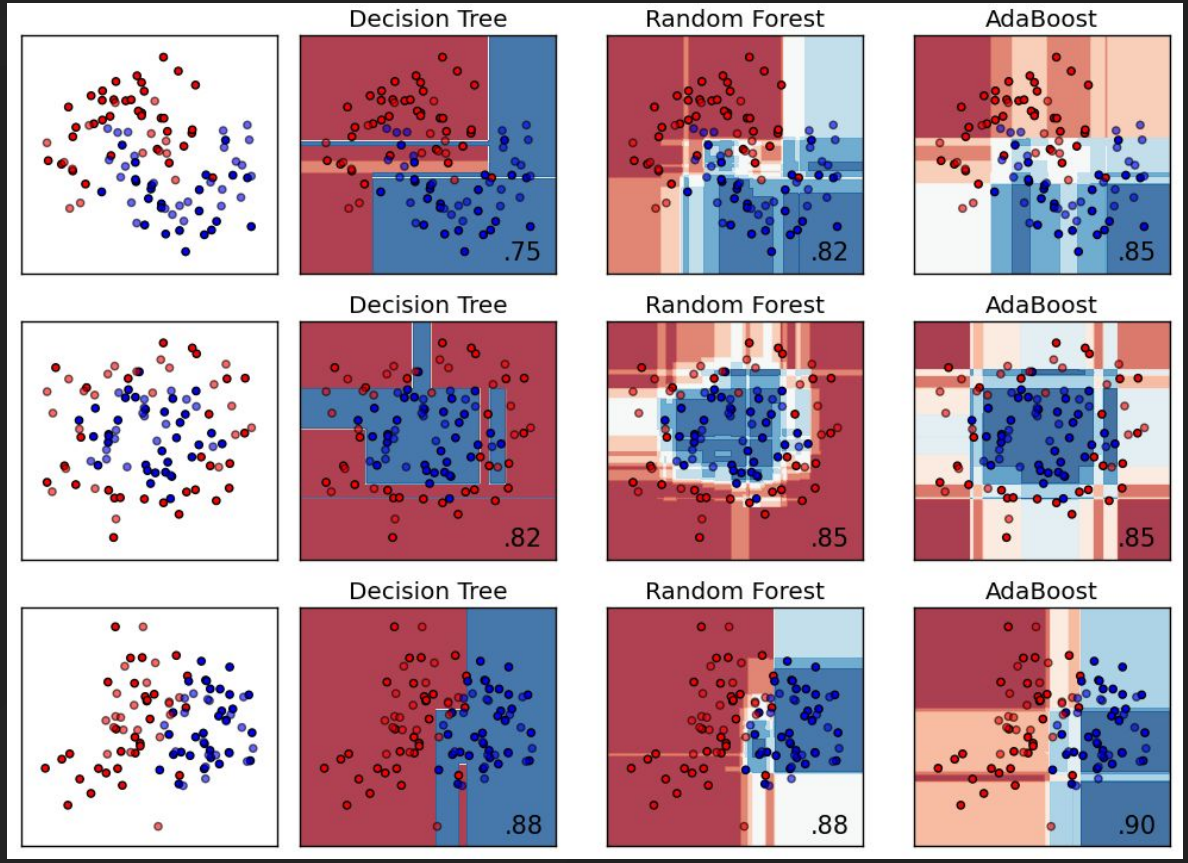
```
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
from xgboost          import XGBClassifier,              XGBRegressor
from lightgbm         import LGBMClassifier,             LGBMRegressor
from catboost         import CatBoostClassifier,         CatBoostRegressor
```



Decision boundary

/ Tree models always have perpendicular (90 degrees) boundaries (like boxes)

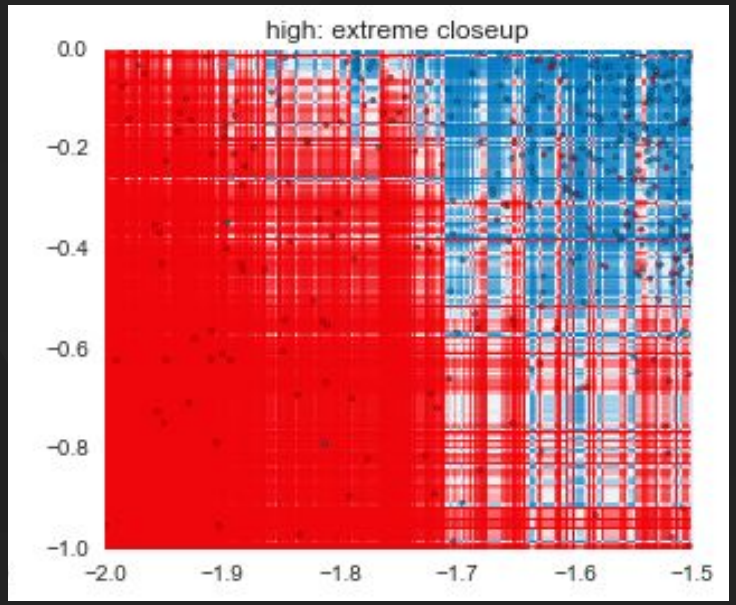
Adaboost and GBM are very similar models.



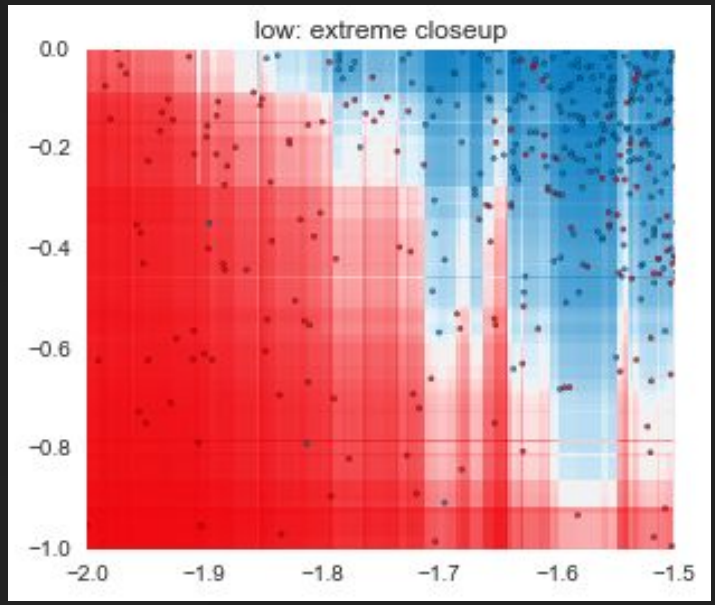
Decision boundary

/ Hyperparameter tuning is very important

Bad tree (overfitting)



Good tree



GBM Hyperparams

/ Hyperparameter tuning is very important

- Number of trees: 100...1000
- Maximum tree depth: 2...10
- % of rows used to build the tree: 80...90
- % of feats used to build the tree: 80...90
- Learning Rate: 0...1 (the lower the better, but slower. Try a log scale)

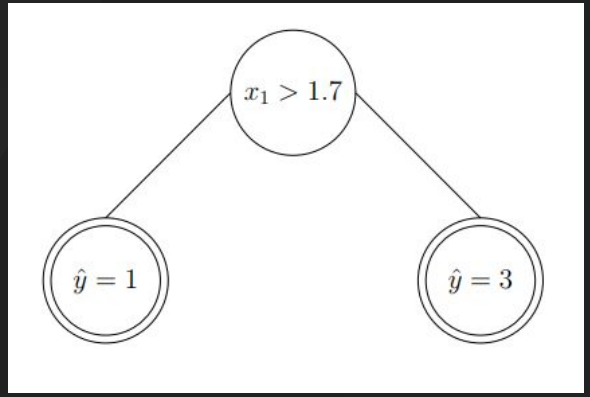
/ Preprocessing

The rule of thumb is:

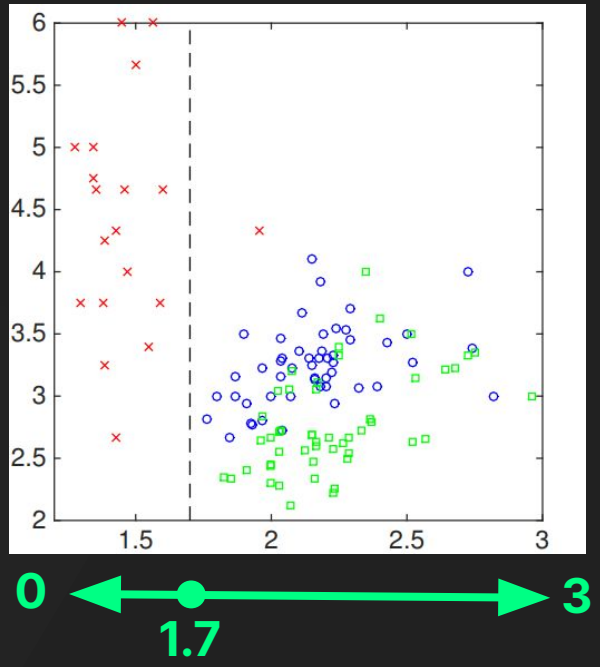
- Numerical variables: Do nothing
- Categorical variables: Ordinal encoding
    - For high cardinality (>1000 categories): Binary encoding

Numerical variables

/ Numerical features does not need any scaling or normalization. Because the tree finds an optimal threshold to split by.



Between 0 and 3: Finds 1.7 as the threshold



Categorical variables

Ordinal encoding

Categorical Feature		Numeric
Louise	=>	1
Gabriel	=>	2
Emma	=>	3
Adam	=>	4
Alice	=>	5
Raphael	=>	6
Chloe	=>	7
Louis	=>	8
Jeanne	=>	9
Arthur	=>	10

Binary encoding

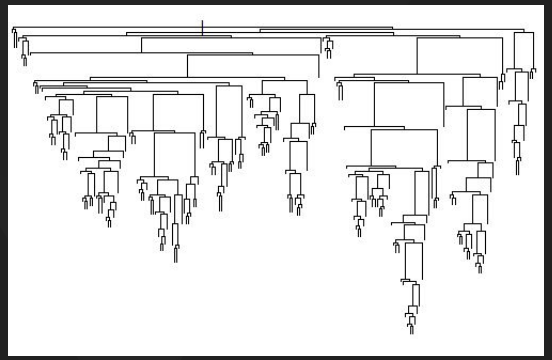
			Binary Encoded			
Categorical Feature	=		x1	x2	x4	x8
Louise	=>	1	1	0	0	0
Gabriel	=>	2	0	1	0	0
Emma	=>	3	1	1	0	0
Adam	=>	4	0	0	1	0
Alice	=>	5	1	0	1	0
Raphael	=>	6	0	1	1	0
Chloe	=>	7	1	1	1	0
Louis	=>	8	0	0	0	1
Jeanne	=>	9	1	0	0	1
Arthur	=>	10	0	1	0	1

One Hot Encoding

Categorical Feature		f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
Louise	=>	1	0	0	0	0	0	0	0	0	0
Gabriel	=>	0	1	0	0	0	0	0	0	0	0
Emma	=>	0	0	1	0	0	0	0	0	0	0
Adam	=>	0	0	0	1	0	0	0	0	0	0
Alice	=>	0	0	0	0	1	0	0	0	0	0
Raphael	=>	0	0	0	0	0	1	0	0	0	0
Chloe	=>	0	0	0	0	0	0	1	0	0	0
Louis	=>	0	0	0	0	0	0	0	1	0	0
Jeanne	=>	0	0	0	0	0	0	0	0	1	0
Arthur	=>	0	0	0	0	0	0	0	0	0	1

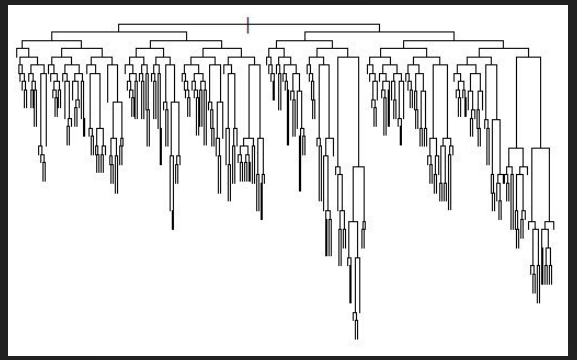
Categorical variables

Ordinal enc. (good)



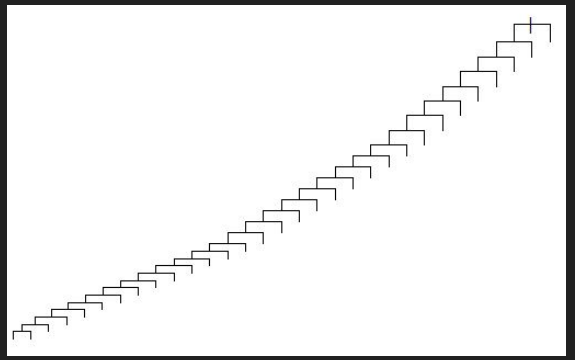
Good for small cardinality
(less than 1000 categories)

Binary enc. (good)



Good for big cardinality
(more than 1000 categories)

One Hot Enc. (Bad)



Reference

/ Q&A

What are your doubts?


Strive
SCHOOL