3011979 Practical Python for Data Sciences and Machine Learning

L9: Performance metrics and tree model tuning

Mar 18th, 2022

Sira Sriswasdi, Ph.D.
Research Affairs, Faculty of Medicine
Chulalongkorn University

Performance metrics

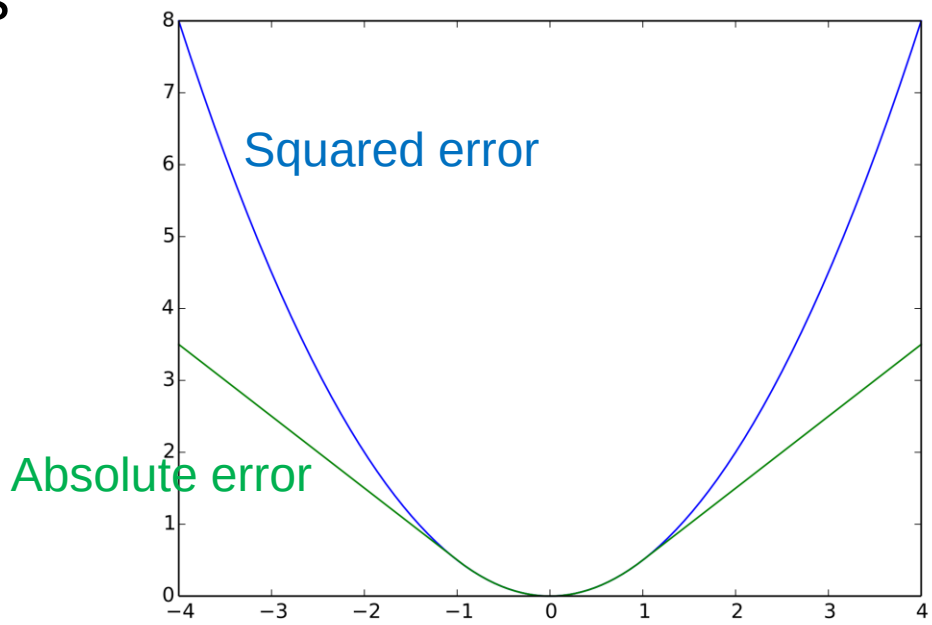
Metrics for regression

sklearn.metrics.mean_squared_error

sklearn.metrics.mean_squared_error(*y_true*, *y_pred*, ***, *sample_weight=None*, *multioutput='uniform_average'*, *squared=True*) [source]

¶

- Located in **sklearn.metrics**
  - The first two arguments are always **y_true** and **y_predicted**
- MSE is sensitive to outliers
- MAE
- MAPE
- R^2

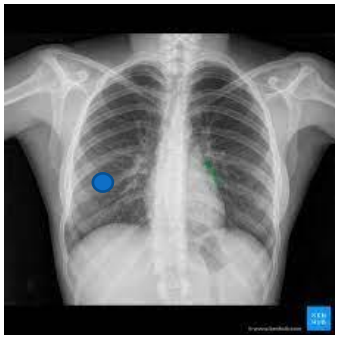


Metrics for classification

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

- Accuracy = (TN + TP) / total
- Precision = TP / (predicted positive)
  = 1 – False Discovery Rate
  = Positive Predictive Value
- Recall = Sensitivity = TP / (all positive)
- F1 = harmonic mean of Precision and Recall

Imbalanced dataset



		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

9,000 pixels

1,000 pixels

- Examples:
  - Lung segmentation of chest x-ray images
  - Rare disease diagnosis
- What happen if the model predict all negatives?
  - Accuracy = 90%
  - Precision = Recall = 0%

Threshold dependency

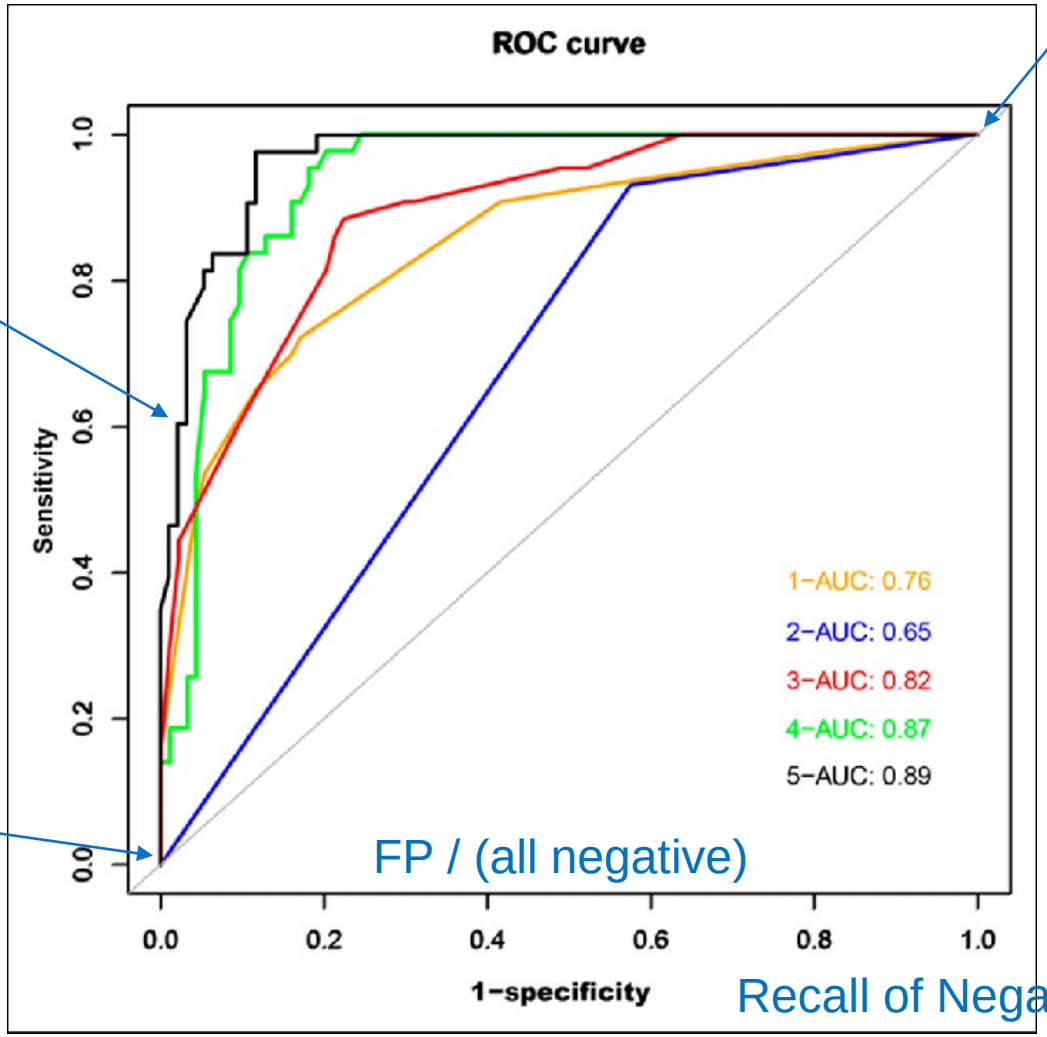
		Predicted: Negative	Predicted: Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive
		Predicted < 0.5	Predicted > 0.5

- Raw prediction in [0, 1]
  - Predicted Negative = raw prediction < *t*
  - Predicted Positive = raw prediction > *t*
  - Default *t* = 0.5
- What if the best threshold is not 0.5?
  - Reduce *t* = more predicted Positive = more Recall
  - Increase *t* = more confidence = more Precision

Threshold-independent metrics

t = 1, predict all Positive

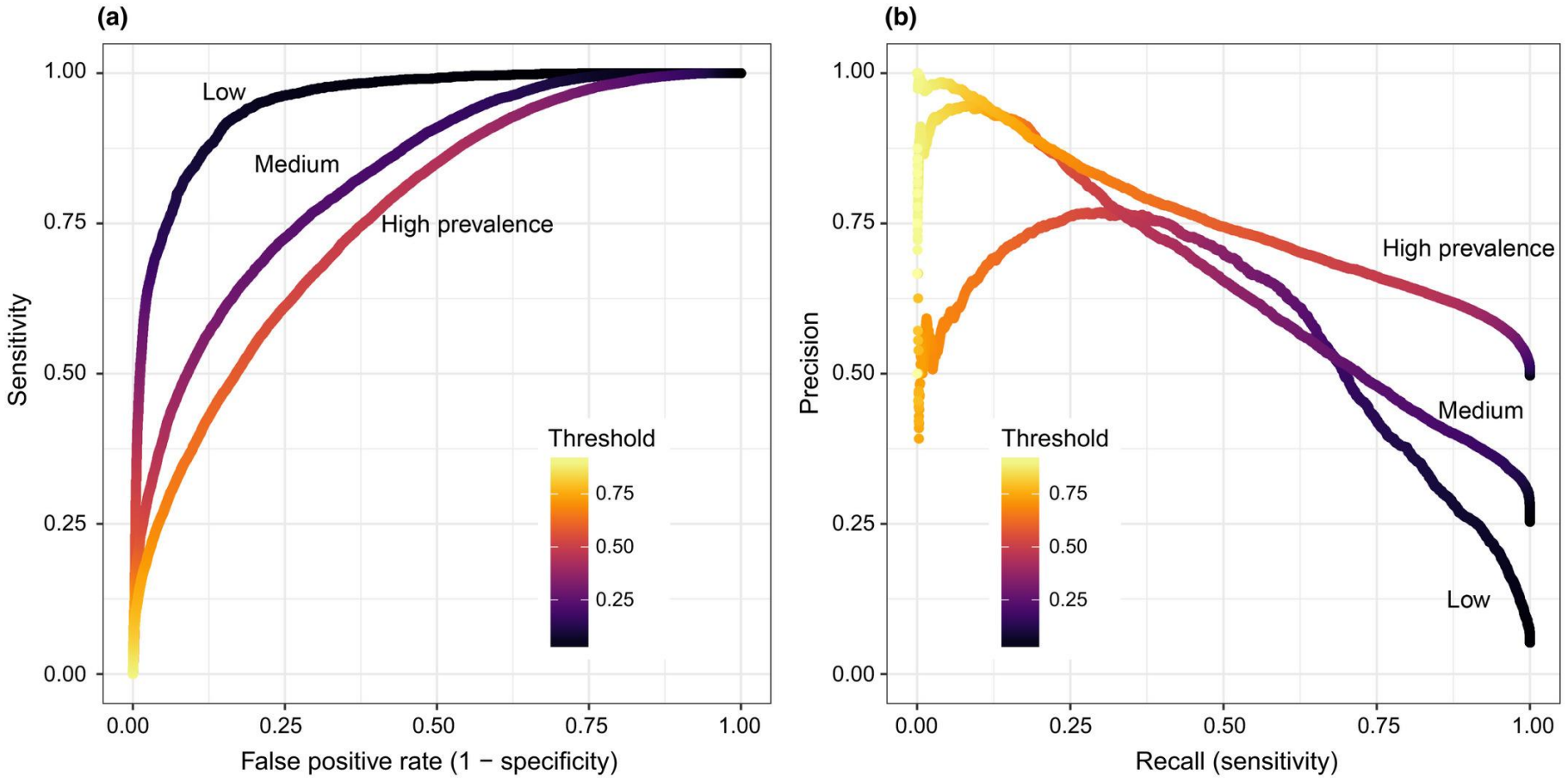
Each point corresponds to setting a threshold, **from 0 to 1**



t = 0, predict all Negative

- Area Under the Receiver Operating Characteristic curve

Precision-Recall Curve



- PR curve focuses on Positive classes
  - Ignore the proportion of Negative samples (specificity)

AUC on imbalanced dataset

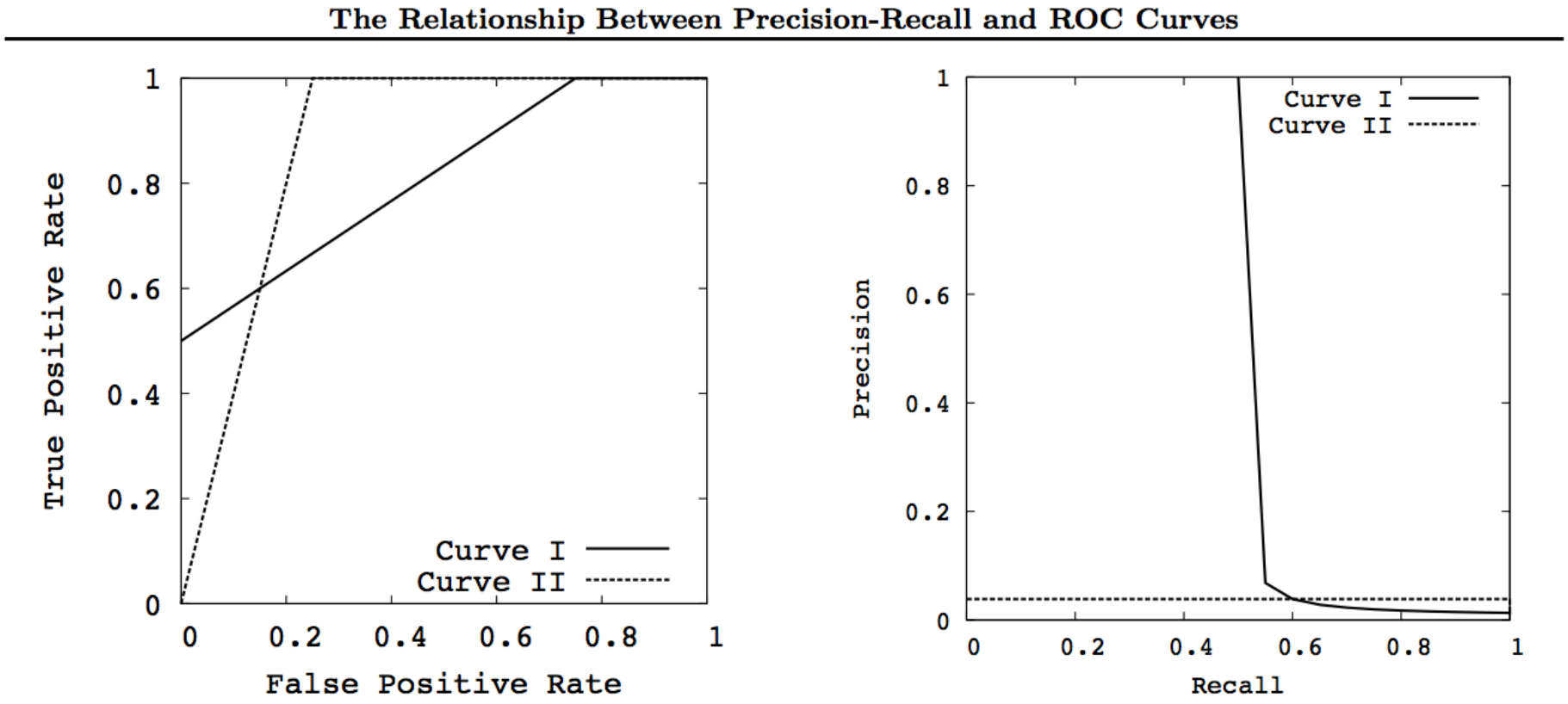


Figure 11. Comparing AUC-ROC for Two Algorithms

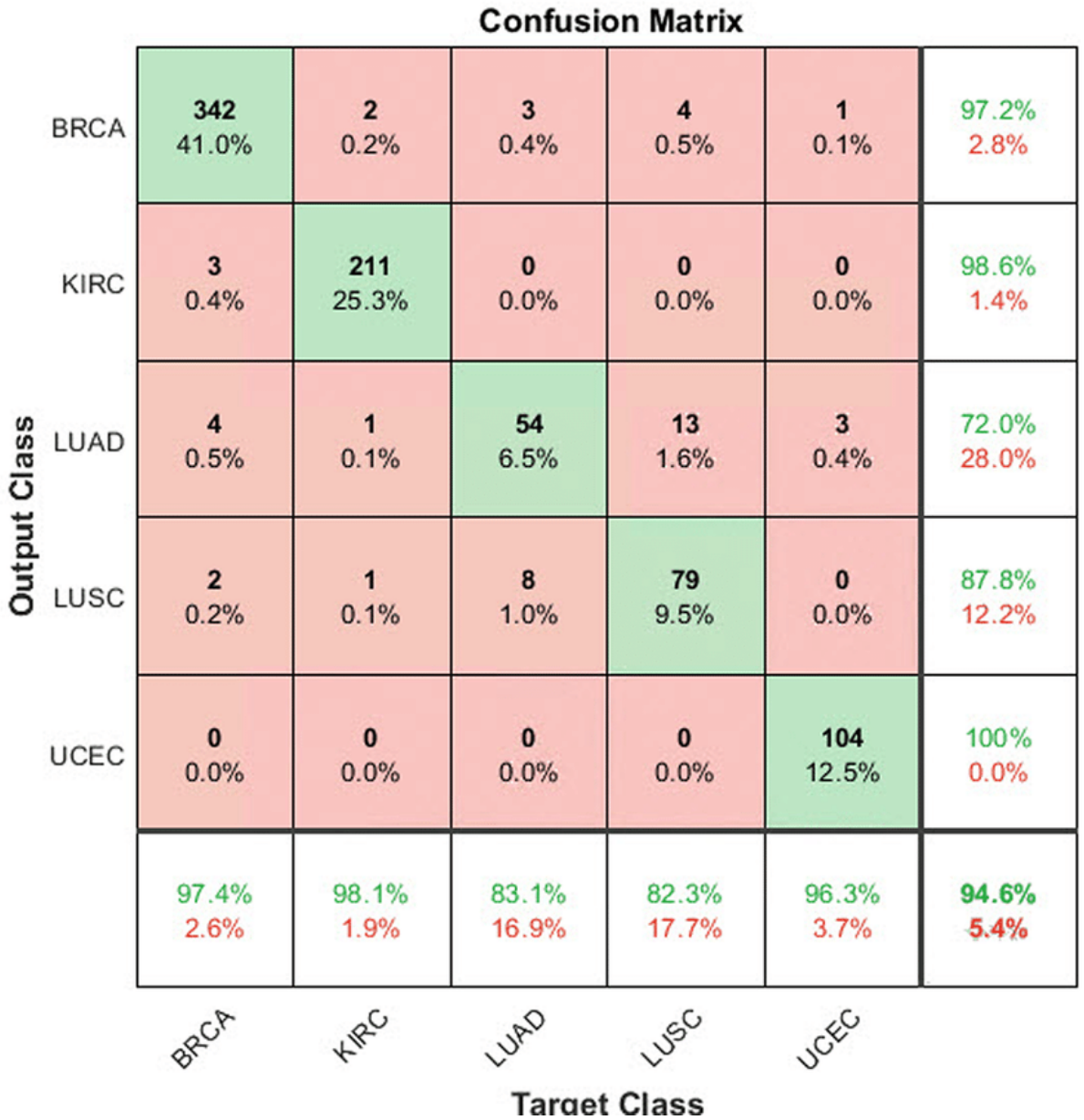
Figure 12. Comparing AUC-PR for Two Algorithms

- AUROC cannot distinguish models as well as AUPRC

sklearn classification metrics

metrics.accuracy_score(y_true, y_pred, *[, ...])	Accuracy classification score.
metrics.average_precision_score(y_true, ...)	Compute average precision (AP) from prediction scores.
metrics.classification_report(y_true, y_pred, *)	Build a text report showing the main classification metrics.
	nt.
metrics.confusion_matrix(y_true, y_pred, *)	Compute confusion matrix to evaluate the accuracy of a classification.
metrics.f1_score(y_true, y_pred, *[, ...])	Compute the F1 score, also known as balanced F-score or F-measure.
metrics.precision_recall_curve(y_true, ...)	Compute precision-recall pairs for different probability thresholds.
metrics.precision_score(y_true, y_pred, *[, ...])	Compute the precision.
metrics.recall_score(y_true, y_pred, *[, ...])	Compute the recall.
metrics.roc_auc_score(y_true, y_score, *[, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
metrics.roc_curve(y_true, y_score, *[, ...])	Compute Receiver operating characteristic (ROC).

Confusion matrix



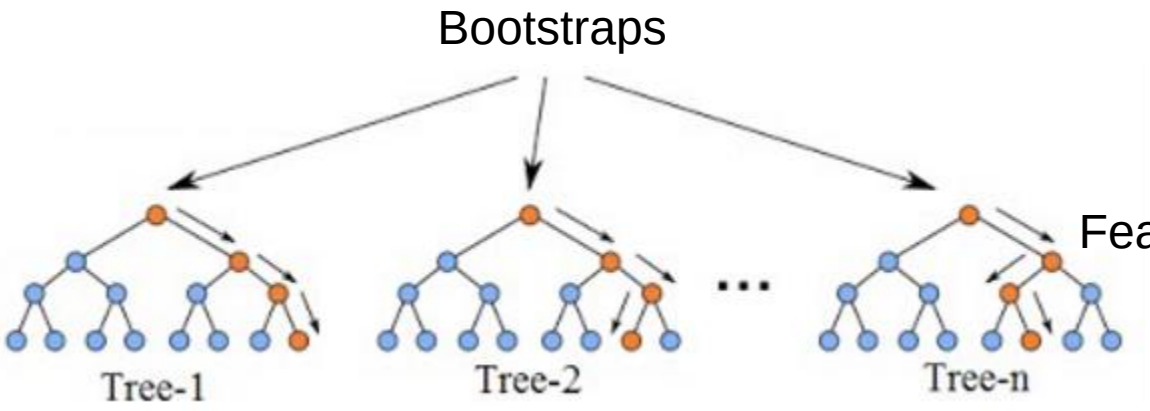
Classification report

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.61      0.55      0.58      3556
           1       0.72      0.77      0.74      5276

   micro avg       0.68      0.68      0.68      8832
   macro avg       0.66      0.66      0.66      8832
weighted avg       0.67      0.68      0.67      8832
```

- Class-by-class Precision, Recall, F1
- Support = number of samples in that class
- Micro average = over all samples
- Macro average = over classes
- Weighted average = over classes, weighted by support

Recap: Hyperparameters of Tree models

Random forest



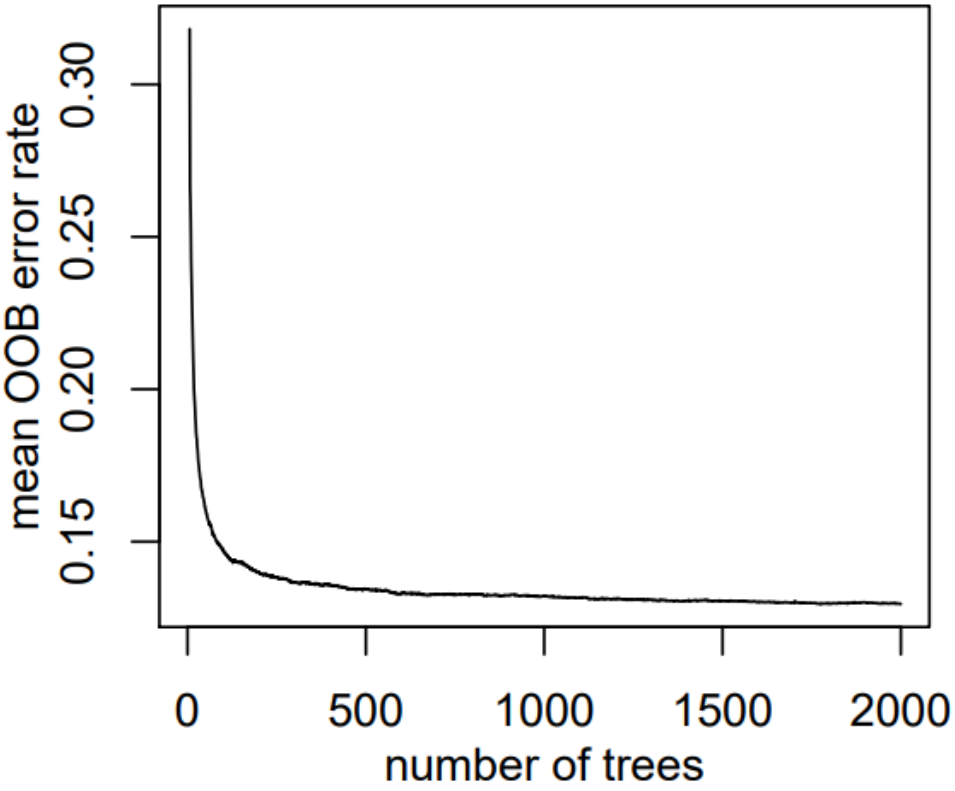
sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

A random forest classifier.

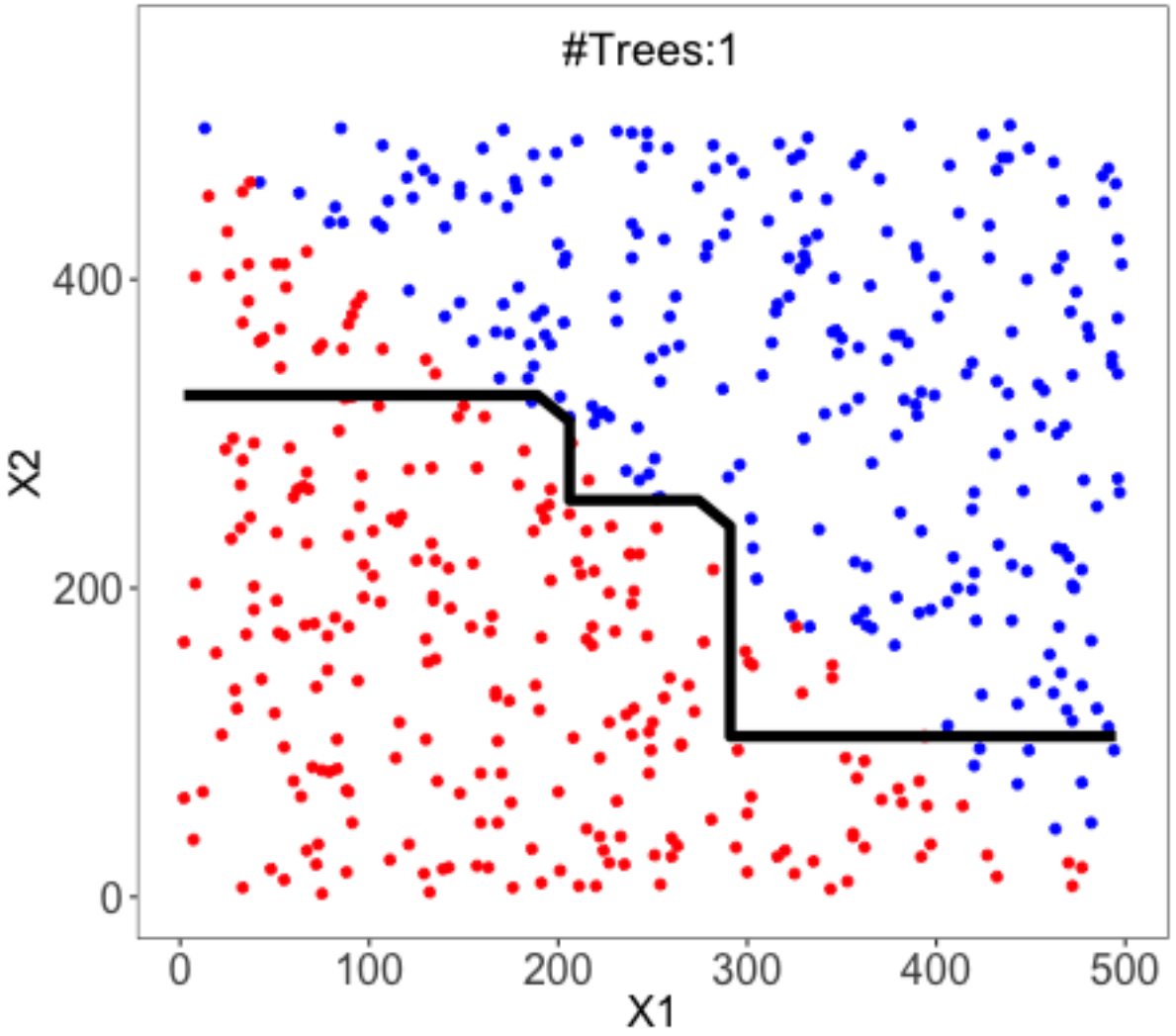
A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Impact of number of trees



- Theoretically, more trees = better performance
  - May fluctuate on **small dataset** and for **accuracy metric**
- **Recommendation**: Set to high value

What causes fluctuation in RF performance?



Impact of *max_feature*

max_features : *{"auto", "sqrt", "log2"}, int or float, default="auto"*

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `round(max_features * n_features)` features are considered at each split.
- If "auto", then `max_features=sqrt(n_features)`.
- If "sqrt", then `max_features=sqrt(n_features)` (same as "auto").
- If "log2", then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

- Too small = noisy features may be used
- Too large = no benefit from feature bagging
- **Recommendation**: Tune as a fraction of feature count

Impact of other hyperparameters

- *max_depth*
- *min_samples_split*
- *min_samples_leaf*

- These prevent overfitting, which is not a big problem for random forest
- **Recommendations**:
  - Tune *max_depth* first, based on the complexity of the task
  - Tune either *min_samples_split* or *min_samples_leaf*, especially for regression task

Framework for tuning in sklearn

sklearn.model_selection.GridSearchCV

class sklearn.model_selection.GridSearchCV(*estimator*, *param_grid*, *, *scoring=None*, *n_jobs=None*, *refit=True*, *cv=None*, *verbose=0*, *pre_dispatch='2*n_jobs'*, *error_score=nan*, *return_train_score=False*) [source]

Exhaustive search over specified parameter values for an estimator.

Important members are fit, predict.

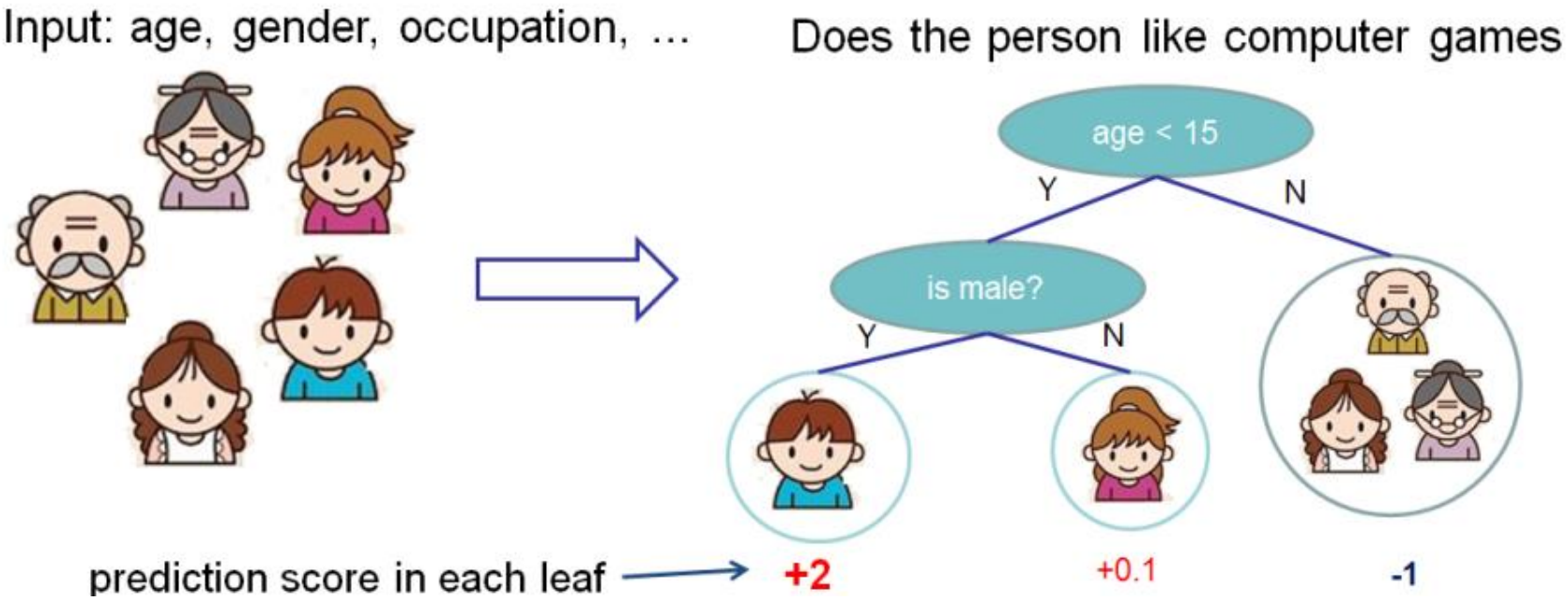
GridSearchCV implements a "fit" and a "score" method. It also implements "score_samples", "predict", "predict_proba", "decision_function", "transform" and "inverse_transform" if they are implemented in the estimator used.

The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid.

- estimator = base model
- param_grid = {max_depth: [10, 50, None], max_features = [0.1, 0.2, 0.3]}
- scoring = names of performance metrics
- cv = number of folds, or pass in custom data splitter

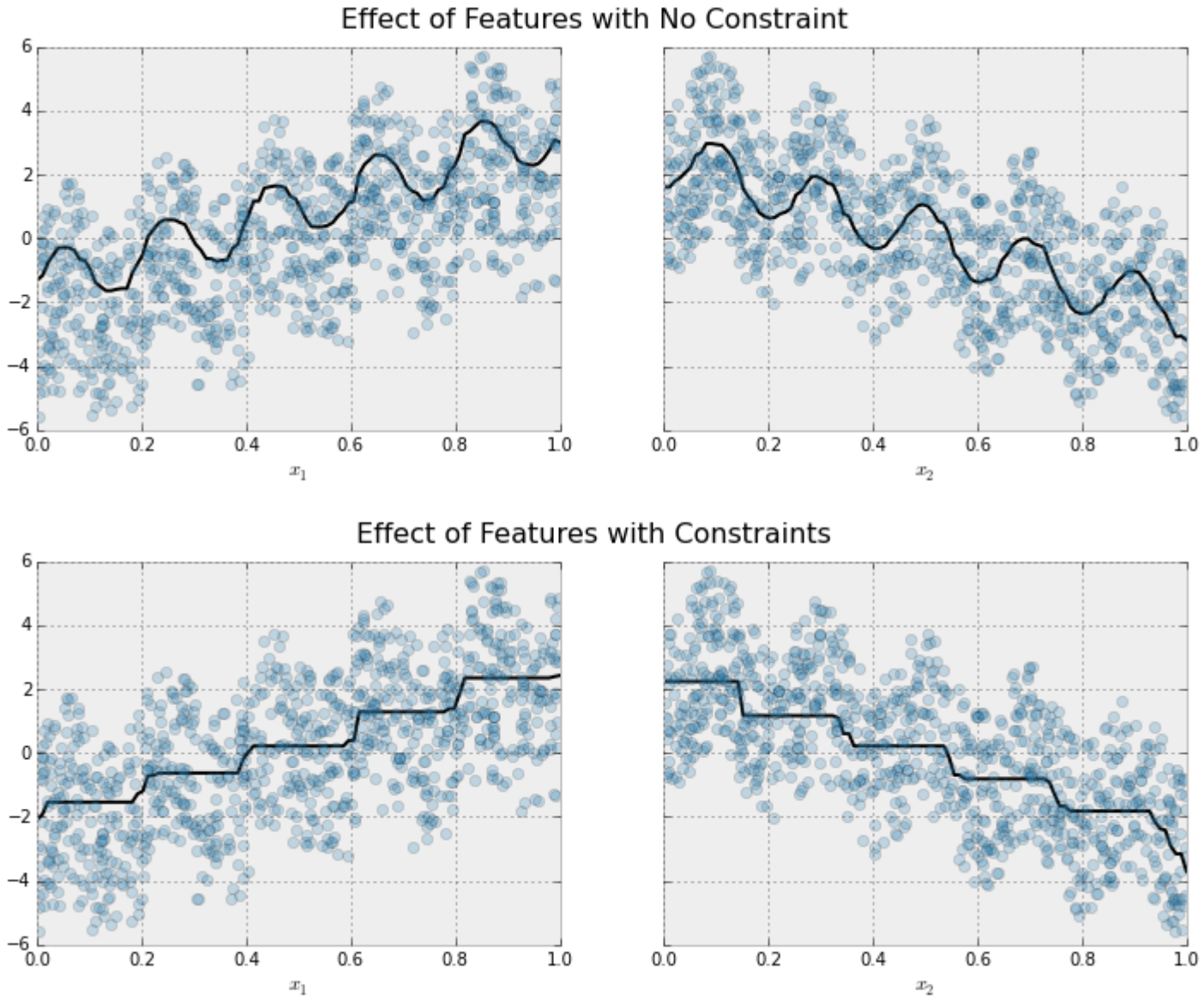
Extreme Gradient Boosting: XGboost

New capabilities



- Able to handle missing data
  - Pick which branch to assign missing value during learning
- Allow regularizations
- Allow monotonicity constraints
  - Reject branching decisions that violate the constraints

Monotonicity constraint



sklearn-style usage

***class* xgboost.XGBClassifier(*, *objective='binary:logistic'*, *use_label_encoder=True*, ***kwargs*)**

Bases: `xgboost.sklearn.XGBModel`, `object`

Implementation of the scikit-learn API for XGBoost classification.

Parameters

- **n_estimators** (*int*) – Number of boosting rounds.
- **use_label_encoder** (*bool*) – (Deprecated) Use the label encoder from scikit-learn to encode the labels. For new code, we recommend that you set this parameter to False.
- **max_depth** (*Optional[int]*) – Maximum tree depth for base learners.
- **learning_rate** (*Optional[float]*) – Boosting learning rate (xgb's "eta")
- **verbosity** (*Optional[int]*) – The degree of verbosity. Valid values are 0 (silent) - 3 (debug).
- **objective** (*Union[str, Callable[[numpy.ndarray, numpy.ndarray], Tuple[numpy.ndarray, numpy.ndarray]], NoneType]*) – Specify the learning task and the corresponding learning objective or a custom objective function to be used (see note below).
- **booster** (*Optional[str]*) – Specify which booster to use: gbtree, gblinear or dart.
- **tree_method** (*Optional[str]*) – Specify which tree method to use. Default to auto. If this parameter is set to default, XGBoost will choose the most conservative option available. It's recommended to study this option from the parameters document: https://xgboost.readthedocs.io/en/latest/treemethod.html.
- **n_jobs** (*Optional[int]*) – Number of parallel threads used to run xgboost. When used with other Scikit-Learn algorithms like grid search, you may choose which algorithm to parallelize and balance the threads. Creating thread contention will significantly slow down both algorithms.
- **gamma** (*Optional[float]*) – Minimum loss reduction required to make a further partition on a leaf node of the tree.

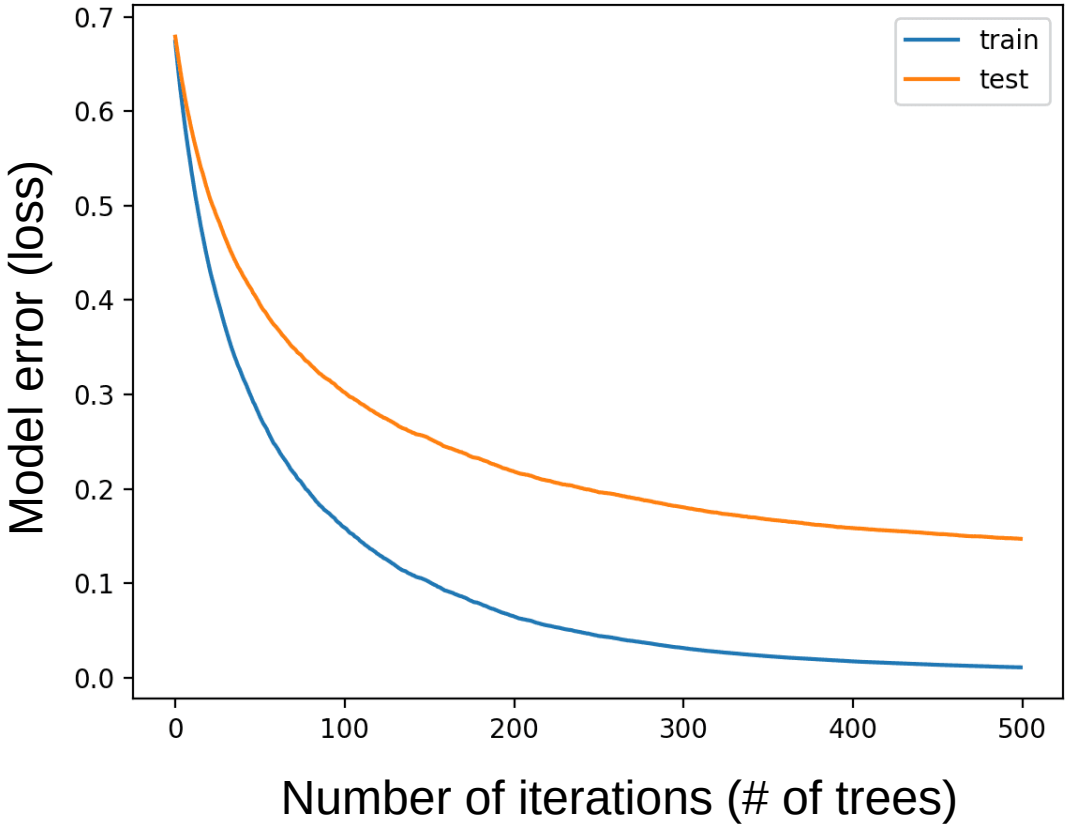
Tuning gradient boosting tree

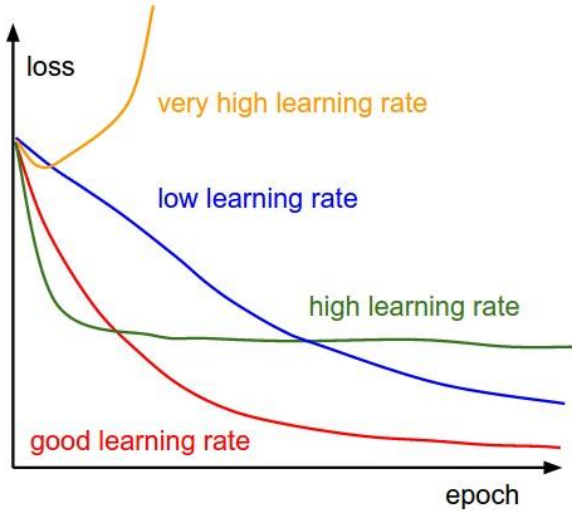
- Unlike random forest, GBT is prone to overfitting

- First line of defense: *max_depth, min_child_weight, gamma*
  - *min_child_weight* ~ *min_sample_split*, but defined with real number
  - *gamma* ~ *min_impurity_decrease*

- Second line of defense: add randomness
  - *subsample* = bootstrapping
  - *colsample_bytree* = feature bagging

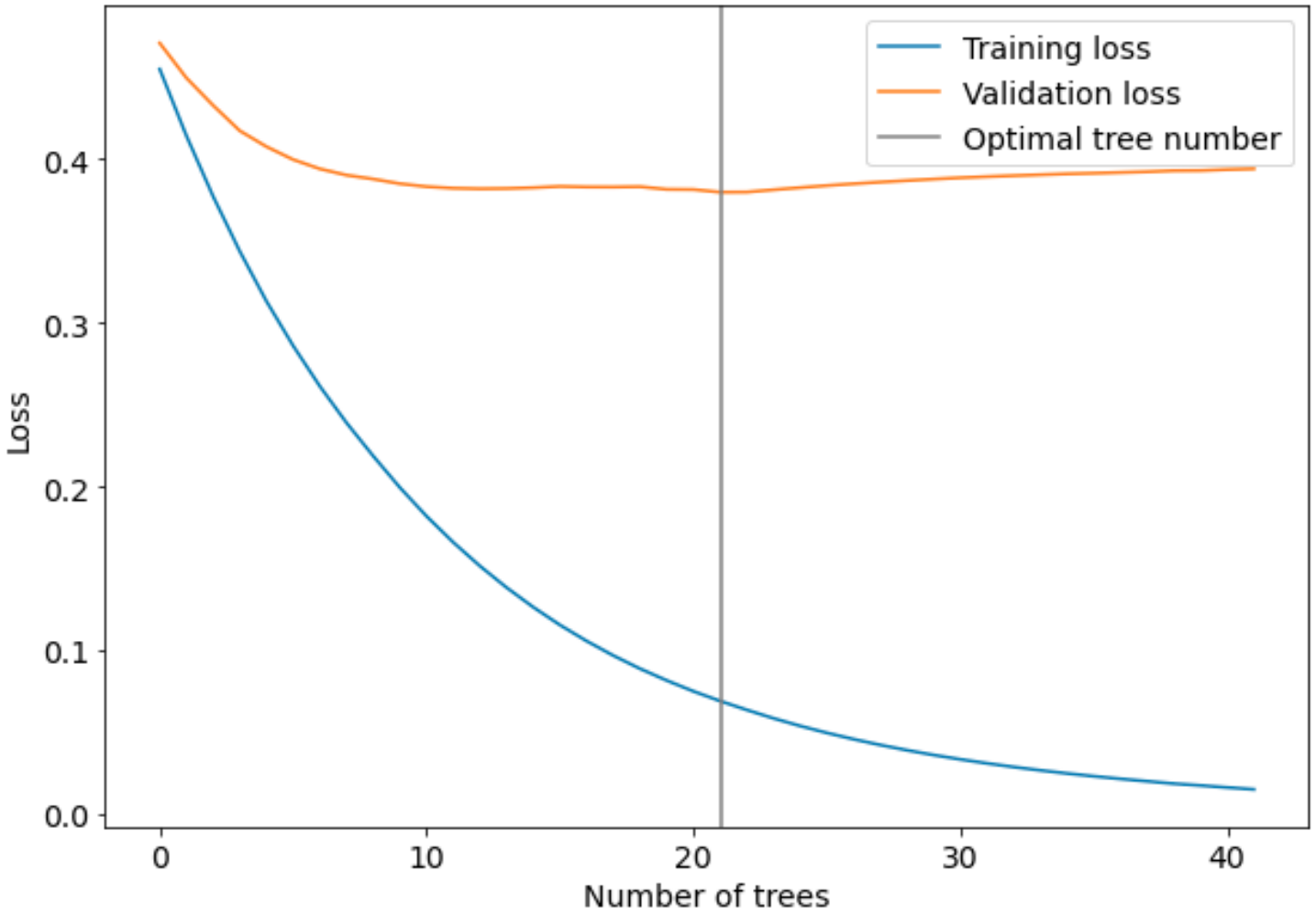
Learning rate





- *learning_rate* and *max_delta_step* cap the learning progress by limiting the impact of new trees

Early stopping



- *Early_stopping_tounds* stop the training once performance on validation set stop decreasing for a certain steps

Let’s do some coding