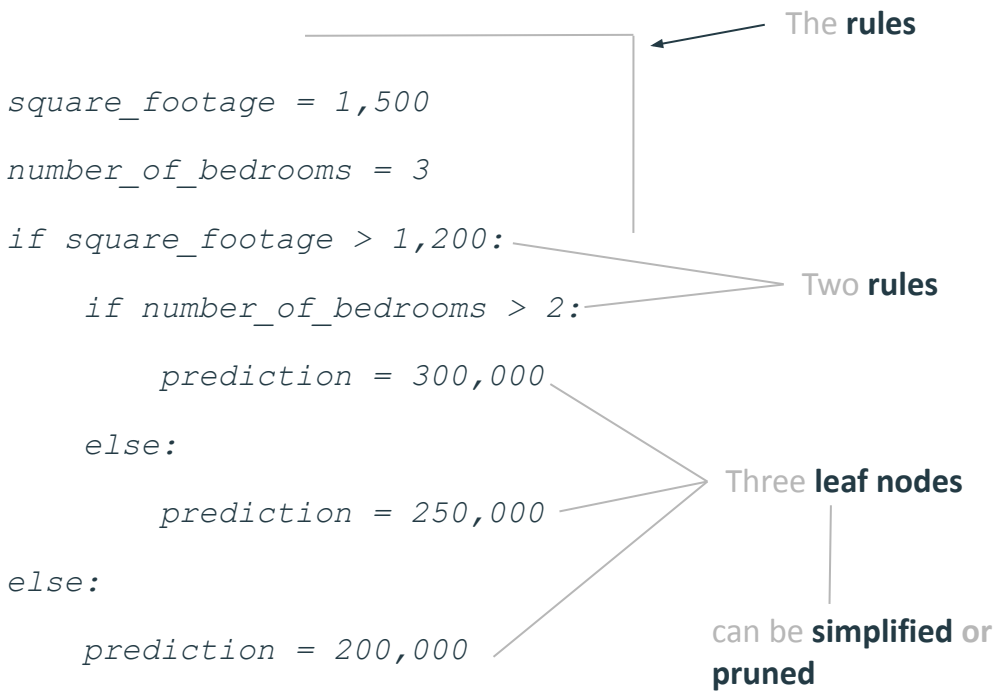
Tree Models

Authors: Tony Fraser, Shariq Mian, Seung-min Song

Contents

Introduction	An overview of terms, and pros and cons
Basic Regression Trees	Trees that model data by splitting it based on feature values to predict a continuous outcome.
Regression Model Trees	Advanced trees that combine the structure of decision trees with linear regression models at their leaves.
Rule-Based Models	Models that apply a set of if-then rules for decision making, often used in systems requiring interpretable logic.
Bagged Trees	A method that combines the results of multiple regression trees built on different subsets of the data to improve model accuracy.
Random Forests	An ensemble learning method that constructs a multitude of decision trees at training time to improve predictive performance.
Boosting	A technique that sequentially builds models, each one correcting the errors of its predecessor, to enhance predictive capability.
Cubist	A specific type of model that builds regression trees with linear regression models at the leaves, similar to regression model trees but with additional features like rule-based conditions.

Introduction to Tree Models



Pros:

- Easy to understand
- Handles sparse, skewed, continuous, categorical predictors
- Good with missing data

Cons:

- Slight data changes drastically change tree structure. (Instability)
- Struggles with complex patterns. (Trees define simple 'boxes' for prediction, but real life relationships are more more intricate)
- These favor predictors with a higher number of distinct values over more granular predictors. (Selection bias)

More Vocabulary

Root/Decision Node	This is the starting point of the tree where the first split is made.
Edge/Branch	The lines between the nodes, flowing from the split to/through the answer(s)
Parent/Child Nodes	The highest node of the edge is the parent, subsequent nodes are children.
Splitting Criteria	This describes the method or metric used to decide how to split at each node (e.g., Gini impurity, information gain).
Depth of the Tree	Refers to the number of levels in the tree.
Terminal Nodes / Leaves	Nodes that do not split further.
Pruning	Removing branches that have little to no importance. Reduces complexity of model to prevent overfitting.
Overfitting/Underfitting	When a model is too complex or too simple for the underlying data.
Feature Importance	How valuable each feature was in the construction of the tree.

Basic Regression Trees

CART (Classification and Regression Tree)

For a data set:

search every predictor, and find all distinct predictor values

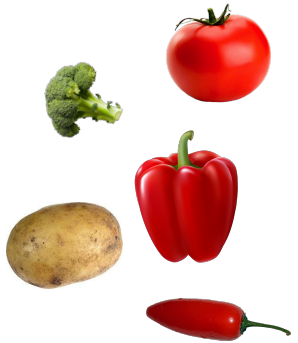
find constant split value that partitions data in two groups

- such that overall sums of squares is minimized

- and no further splits are possible

Group by predictors:
Find all options:

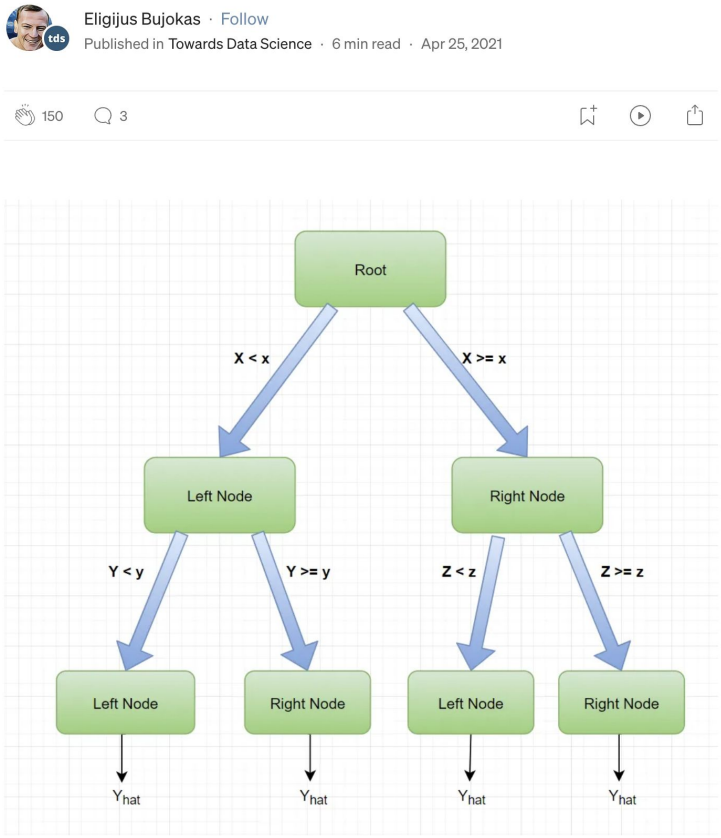
Size
Color
Type
Shape
Wrapper
Seeds



Predict price with smallest error

Rule Based Models

- rules are the "forks" in the tree
- path has only one possible route
- coverage is the number of samples affected by a rule
- pruning and smoothing



Graph of a regression tree; Schema by author

Regression Model Trees

Like basic trees because:

- split is found using exhaustive search over predictors

Unlike basic trees because:

- each terminal leaf path contains an LR (or more complex) model, instead of an average or constant
- splits are made to optimize within regions assuming leaf regression paths
- ends with smoothing, which often handles collinearity and reduction of variables
- and pruning, which deals with final nodes that don't contain enough data

Bagged Trees

short for **b**ootstrap **agg**regation

```
for i between 1 and M do:
   generate a bootstrap sample of original data
   train an unpruned tree model on this sample
end
 -- now predict y hat with all M models, and average
```

- an ensemble technique, bootstrapping plus another model

- reduces variance of prediction through aggregation

- bootstrapping adds stability to more unstable models (like regression trees)

- *out of bag* samples can provide predictive performance estimates

-works fast, lowers RMSE, and with few iterations of M

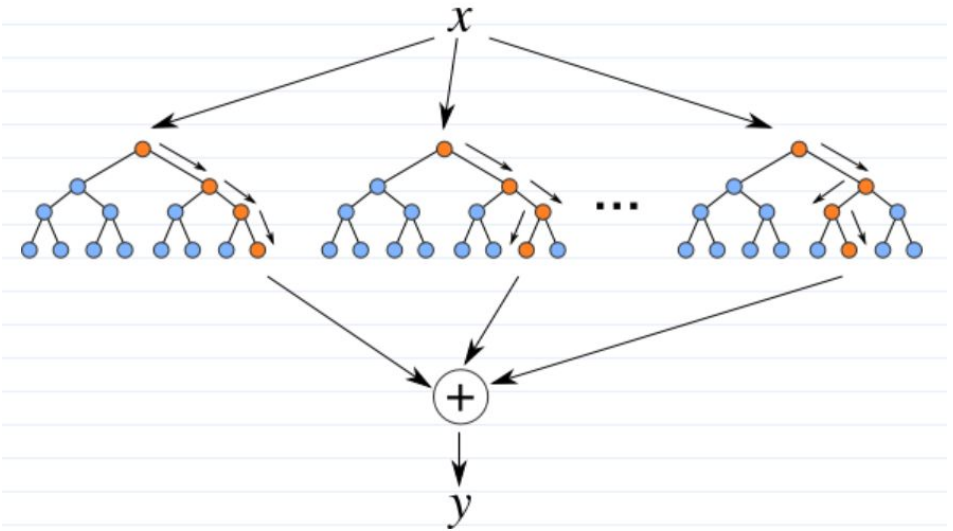
Random Forests

What is the ensemble method?

It is a method of making more accurate predictions by combining several weaker models instead of using a single strong model. The principle of decorrelated is adopted to ensure that each decision tree is mutually independent.

Decision tree algorithms are not commonly used for classification in practice due to their tendency to overfit. Instead, random forests are often preferred as they help mitigate this issue.

- Random extraction for each tree: All variables and observations used in learning each Decision Tree are randomly extracted.
- Since each tree is trained independently from each other, it becomes resistant to noise when making integrated decisions.



Variance Bias

Random Forests

Advantages of Random Forest: A powerful ensemble of decision trees applying the bagging/pasting method for optimization

- Can be used for both classification and regression problems
- Easy to handle missing values
- It is easy to process large amounts of data
- Solves the overfitting problem
- Can determine which features are most important.
- Strong against non-linear data

Hyper Parameters

ntree and mtry are two important hyperparameters that can be set in the Random Forest algorithm. Each of these determines the number of trees in the model and the number of features to consider for splitting.

ntree *(Number of Trees):* Typical defaults are 100 or 500. It is recommended that the actual optimal value be determined through cross-validation.

mtry*(Number of Features to Consider at Each Split):* Basically, 1/3 of the number of available features is used in regression problems, and the square root of the number of features is used as the default in classification problems.

E.g. rf_model <- randomForest(Species ~ ., data=iris, ntree=100, mtry=2, importance=TRUE)

Random Forests

A powerful ensemble of decision trees applying the bagging/pasting method for optimization.

- Classification Use: RandomForestClassifier
  - e.g. medical diagnosis, customer segmentation, anomaly detection, etc.
- Regression usage: RandomForestRegressor
  - e.g. predict housing prices, stock prices, etc.

The two models below are basically the same model.

```
rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16,
                                 n_jobs=-1, random_state=42)

bag_clf = BaggingClassifier(DecisionTreeClassifier(splitter="random",
                                                   max_leaf_nodes=16,
                                                   random_state=42),
                            n_estimators=500, max_samples=1.0,
                            bootstrap=True, random_state=42)
```

Boosting

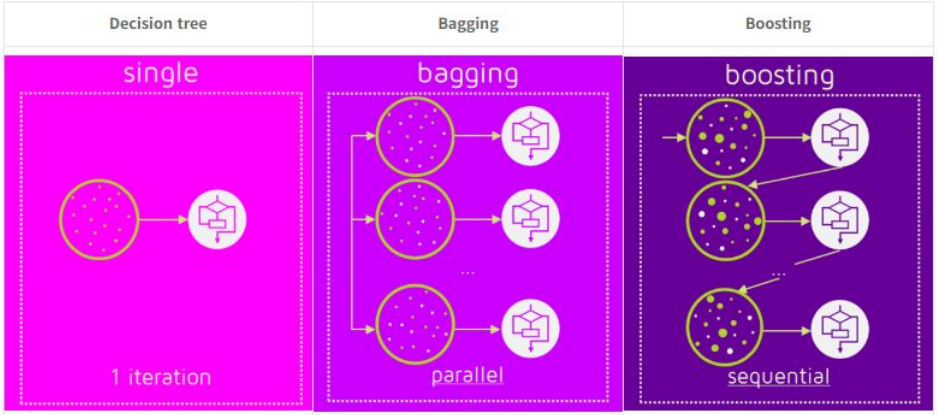
An ensemble technique that creates a high-performance learner by linearly connecting several low-performance learners. Representative algorithms are as follows.

- AdaBoost
- Gradient Boosting

Boosting is a technique where performance is gradually improved based on the results of the previous learner, reducing bias.

The main idea of boosting is to train a predictor with good performance by compensating for the weaknesses of a weak predictor.

Because it learns sequentially, it is less scalable than bagging/pasting.



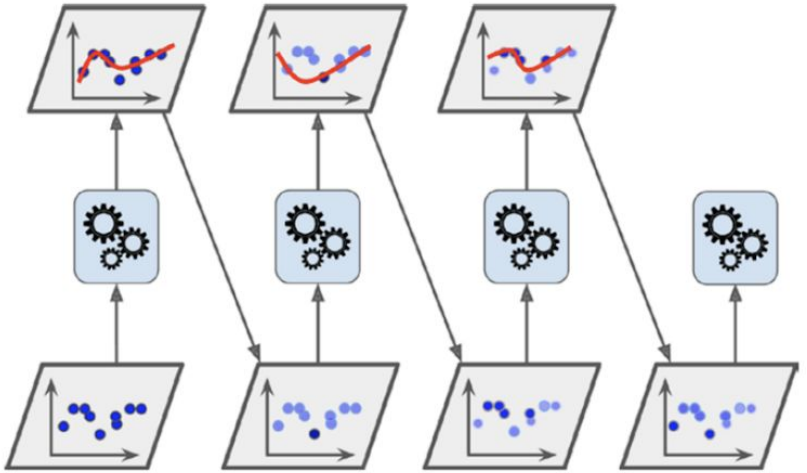
ADABoost

An ensemble technique that adds new predictors by adjusting incorrectly applied weights to produce better predictors. Create a new model by increasing the weight of samples that the previous model did not learn properly, that is, were underfitted.

New predictors are successively created to better adapt to samples that are difficult to learn from.

- Boosting has fewer errors than bagging. In other words, it performs well
- It is slow and has the potential for overfitting
- Low performance of individual decision trees -> boosting
- Overfitting is an issue -> bagging

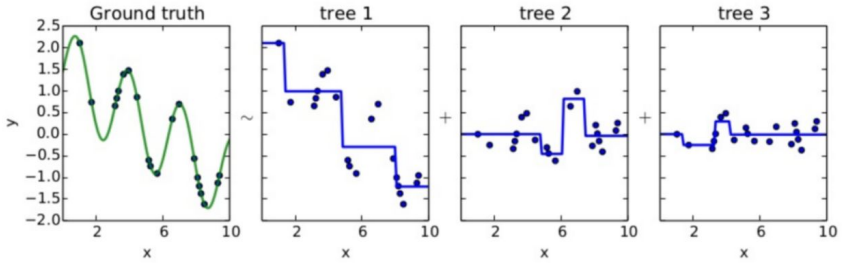
(!) Although it appears to be a powerful learning method, some point out that it is too powerful and gives more weight to misclassification than necessary.
Not only is it necessary to re-solve the problems got wrong, but also to look again at the problems got right.



Gradient Boosting

- Bundle of decision trees
- Unlike random forests, decision trees are created sequentially and errors in the previous tree are compensated for
- Creating a final model with good performance by connecting many shallow trees
- High prediction performance
- Algorithms that require a significant amount of calculation -> Requires efficient implementation of hardware
- Light GBM is often preferred for its speed and reduced susceptibility to overfitting.

Gradient boosting, like Adaboost, uses sequential learning to correct previous errors. However, rather than modifying the sample weight at each repetition like AdaBoost, a new predictor is trained on the residual error created by the previous predictor.



※ Residual is the difference between predicted and actual values.

Cubist

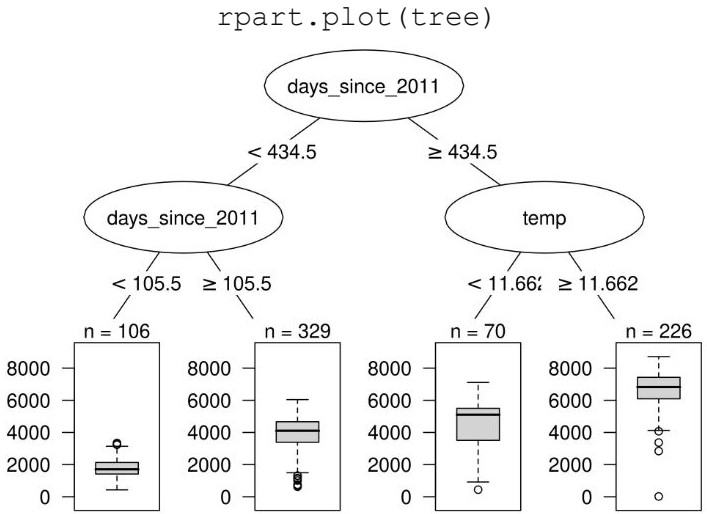
```
library(Cubist)
E.g. m.cubist <- cubist(x=wine_train[-12], y=wine_train$quality)
```

- Cubist is a rule-based model that combines several methodologies published in 1987, 1992, and 1993
- The model has evolved over time and became open source in 2011

Feature	Traditional Models	Cubist
Linear Model Smoothing	Potential for discontinuity between nodes	Smooth combination of linear models between parent and child nodes
Rule Generation	Independent linear regression on split data	Rules are combined into a continuous model with smoothing
Pruning	Complexity parameter adjustment or use of validation data	Use of adjusted error rate to prune without increasing error rate
Committees	Prediction made by a single model	Sequential generation of rule-based models affected by previous models
Adjustment from Training Set	Direct fit to training data	Ability to adjust model prediction using the nearest neighbors from the training set

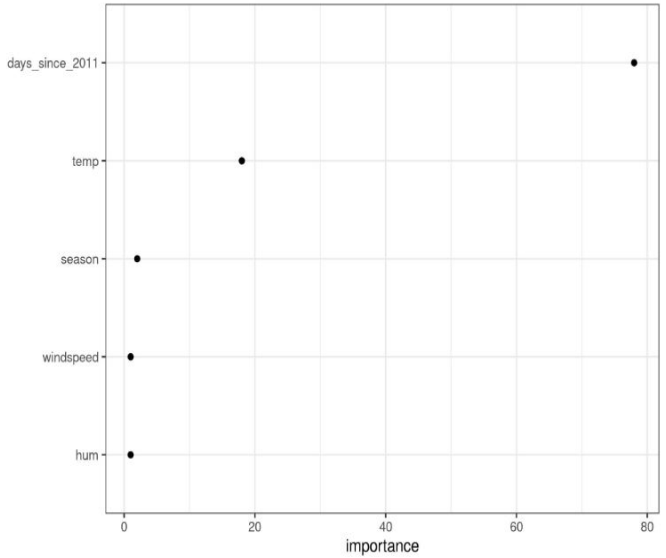
Decision Tree Evaluation

Tree



Feature Importance

Importance(tree_model)

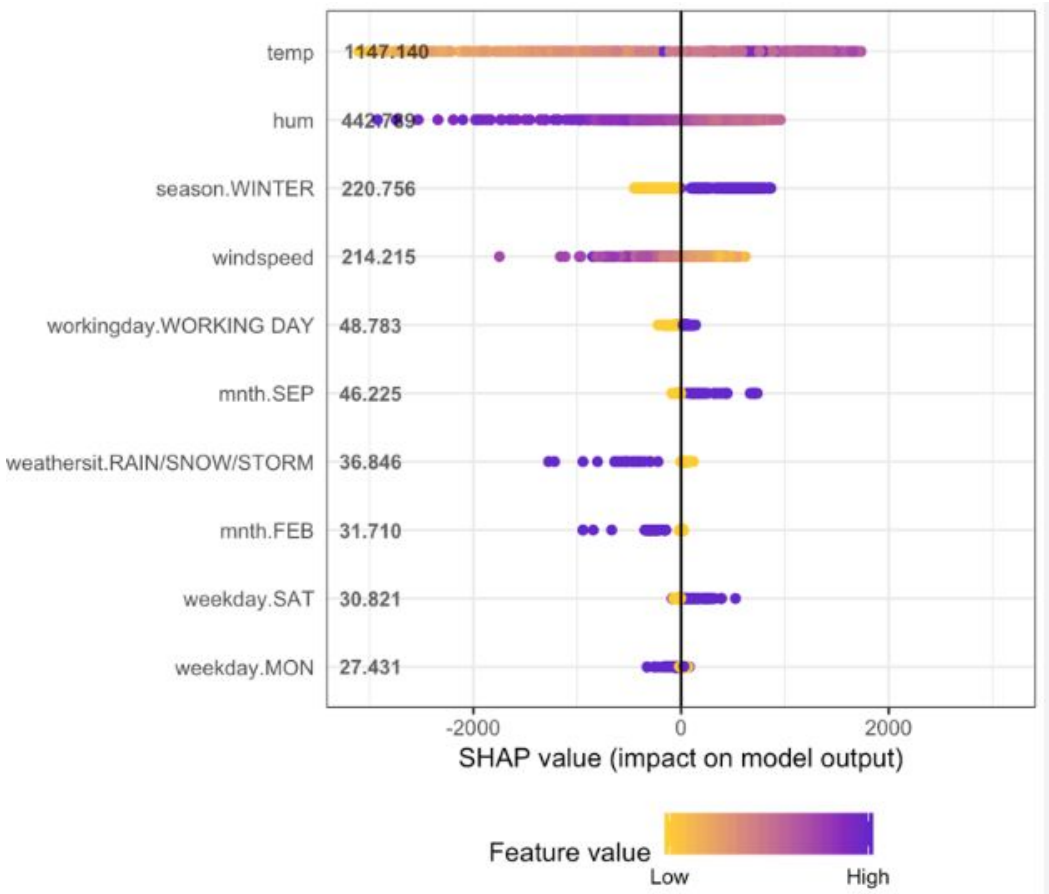


Feature Impact

```
library (shapper)

shap_values <- shap(xgb_model, X = train$data)

plot(shap_values)
```



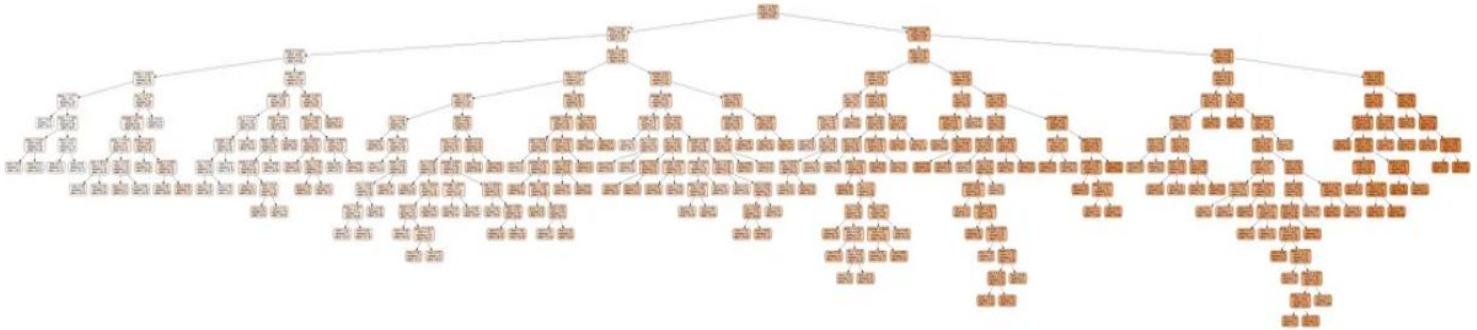
Hyperparameter Tuning for Tree-Based Regression Models:

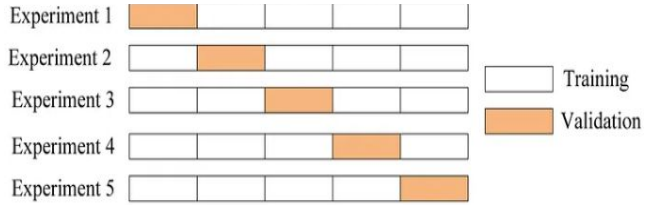
- Hyperparameters is like the settings for an algorithm that can be adjusted to optimize performance. Tuning hyperparameters is a crucial step in building the most accurate model. It is not usually necessary to tune every hyperparameter, but it is important to adjust certain ones so that you can improve your overall model.
    a. **Maximum Depth:** The maximum depth of the tree. A deeper tree can capture more complex relationships but is prone to overfitting.
    b. **Minimum Samples Split:** The minimum number of samples required to split an internal node. Increasing this parameter can prevent overfitting.
    c. **Minimum Samples Leaf:** The minimum number of samples required to be at a leaf node. Similar to min_samples_split, it helps control overfitting.
    d. **Maximum Features:** The number of features to consider when looking for the best split. Limiting the number of features can reduce model complexity.
    e. **mtry:** Num of features to be considered for each tree.
    f. **Sampsize:** the size of sample for each bootstrap sample
    g. **Criterion:** The function to measure the quality of a split. Common options include "mse" for mean squared error and "mae" for mean absolute error. ***“squared_error”, “friedman_mse”, “absolute_error”, “poisson”***

Overfitting

Overfitting occurs when a machine learning model learns the training data too well, capturing noise or random fluctuations in the data rather than the underlying patterns or relationships.

-Max Depth and Cross Validation can solve for overfitting.





5 Fold Cross Validation (Source)

Grid Search CV

- Grid Search: Exhaustively search through a specified grid of hyperparameters. It evaluates the model's performance for each combination of hyperparameters and selects the best one.

288 Combinations

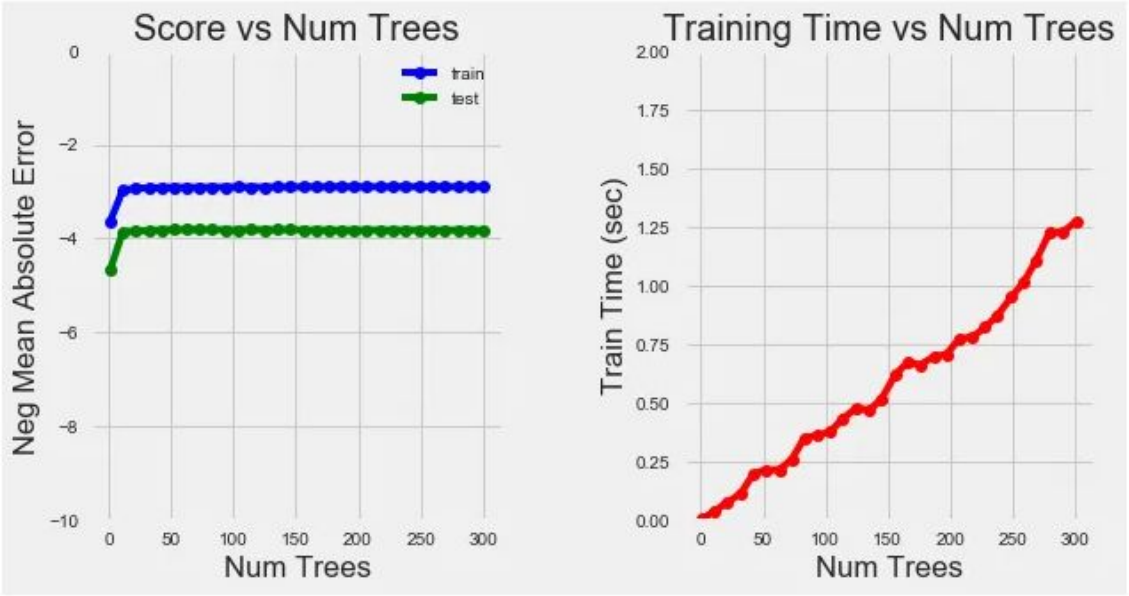
```
param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}
```

Best Grid Search

```
{'bootstrap': True,
 'max_depth': 80,
 'max_features': 3,
 'min_samples_leaf': 5,
 'min_samples_split': 12,
 'n_estimators': 100}
```

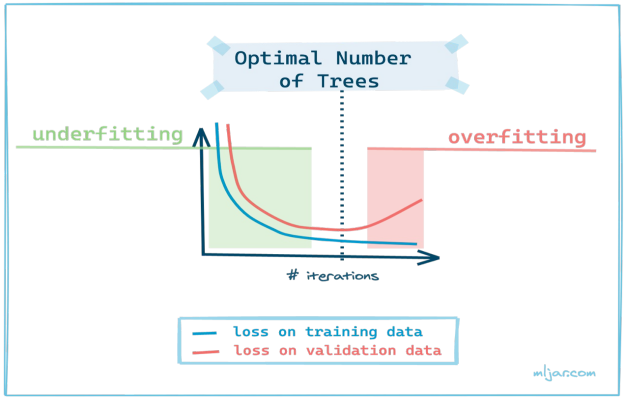
- **Random Search:** Randomly sample hyperparameters from specified distributions. It is computationally less expensive than grid search and can be more effective in high-dimensional hyperparameter spaces.

Tradeoff Between Optimization & Compute



XG Boost Hyper Parameter Tuning

Early Stopping-Using patience parameter



*https://towardsdatascience.com/a-guide-to-xgboost-hyperparameters-87980c7f44a9

Hyperparameter	Role	Range	Default
subsample	Subsample ratio of training instance (e.g.|0.5 means 50% data used prior to growing trees)	>0 – 1	1
colsample_bytree	Frac. of columns selected for each tree	>0 - 1	1
colsample_bylevel	Frac. of columns selected for training at each level	>0 - 1	1
gamma (alias: min_split_loss)	Regularization param for tree pruning. Specifies minimum loss reduction required to make a split (a node is split only if there's positive reduction in loss func)	0 – inf.	0
booster	Selection of a booster (gbtree: tree-based, gblinear: Ridge regression)	'gbtree', 'gblinear', 'dart'	'gbtree'
learning_rate (alias: eta)	Regularization param. Step size. Shrinks feature weights in each boosting step.	0-1	0.3 {0.01, 0.1, 0.2}
max_depth	Maximum tree depth	0 – inf.	6
monotone_constraints	Increasing constraints on predictors (e.g. non-linear, increasing likelihood of loan approval with higher credit score)	(1,0), (0, -1)	'( )'
n_estimators	Number of trees boosted (early_stopping_rounds can be activated)	1 – inf.	100
reg_alpha (alias: alpha)	L1 regularization (increasing its value makes model more conservative)		0
reg_lambda (alias: lambda)	L2 regularization (increasing its value makes model more conservative)		1
scale_pos_weight	Dealing with imbalance data	Tot neg. instance (no-fraud)/tot pos. instance (fraud)	1

Business Use Case

-For real time use cases often XG boost is used due to its Accuracy, Speed and Streaming capabilities and its ability to cache and parallelize the processing on different nodes.

-For Random Forest you often have to pre process/Grid Search CV the data to avoid overfitting while for XG boost you can use regularization L1,L2 to solve for overfitting while leeping ther real time capabilities.

Questions??