Machine Learning Engineer Course
Day 15

- Ensemble Learning -

DIVE INTO CODE

Thursday June 17, 2021

DIOP Mouhamed

Agenda

1. Check-in
2. How to proceed
3. Quick Review
4. Ensemble Learning
5. Ensemble module of Scikit-learn
6. Assignment
7. Ensemble Learning – Sample Code
8. Check-out

Check-in

3 minutes Please post the following point to Zoom chat.

Q. What did you learn in the previous week?
(Anything is fine.)

How to proceed - Objective

Understand Ensemble Learning

1. Blending

A method of evaluation using estimates from multiple models

2. Bagging

A method of dividing the training data and evaluating the estimates of multiple decision tree models

3. Stacking

A hybrid method of blending and bagging. A method that uses multiple models and divides the training data for evaluation

Let’s learn the basics of Ensemble Learning

Quick Review (Scratch Clustering)

K-means method

Review the implementation steps.

① Assign random initial labels for k classes to the index of the number of samples.

② Create clusters by grouping data points for each label

③ Find the average value of the data points for each cluster and use it as the center of gravity for that cluster.

④ Calculate the distance from its center of gravity to the data points of all samples.

⑤ Assign that data point to the cluster of centers of gravity with the smallest distance from each data point.

⑥ Repeat steps ③ to ⑤.

⑦ Exit when the convergence conditions (value does not change, the defined number of iterations has been reached, etc.) are met.

⑧ Change the initial value, repeat ① to ⑦ n times, and select the one with the smallest SSE.

Target audience for this assignment

① Able to write code to train and estimate using scikit-learn's ensemble module.

② Went through the Decision Tree and Clustering assignments.

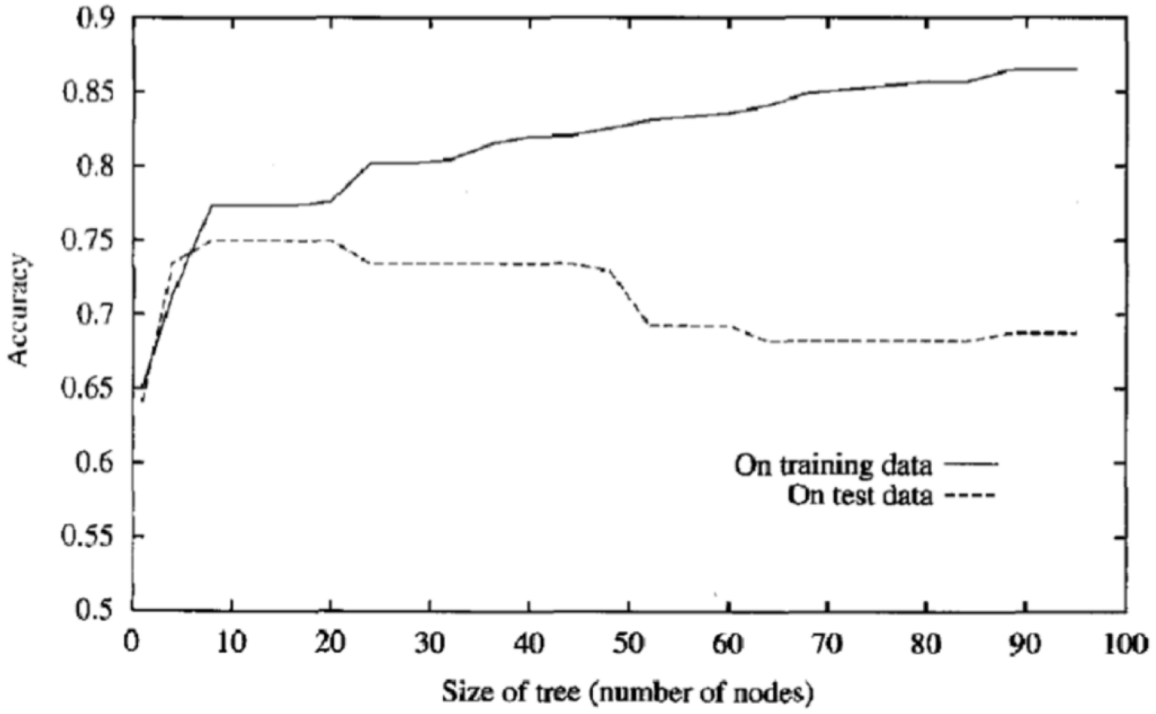
What is ensemble learning?

A method of combining multiple learning models to improve prediction accuracy.

The model at the bottom of the hierarchy is often called the base model (base / weak learner / genelizer) and the model at the top is often called the meta model (meta learner / stacker)

Decision Tree Problems

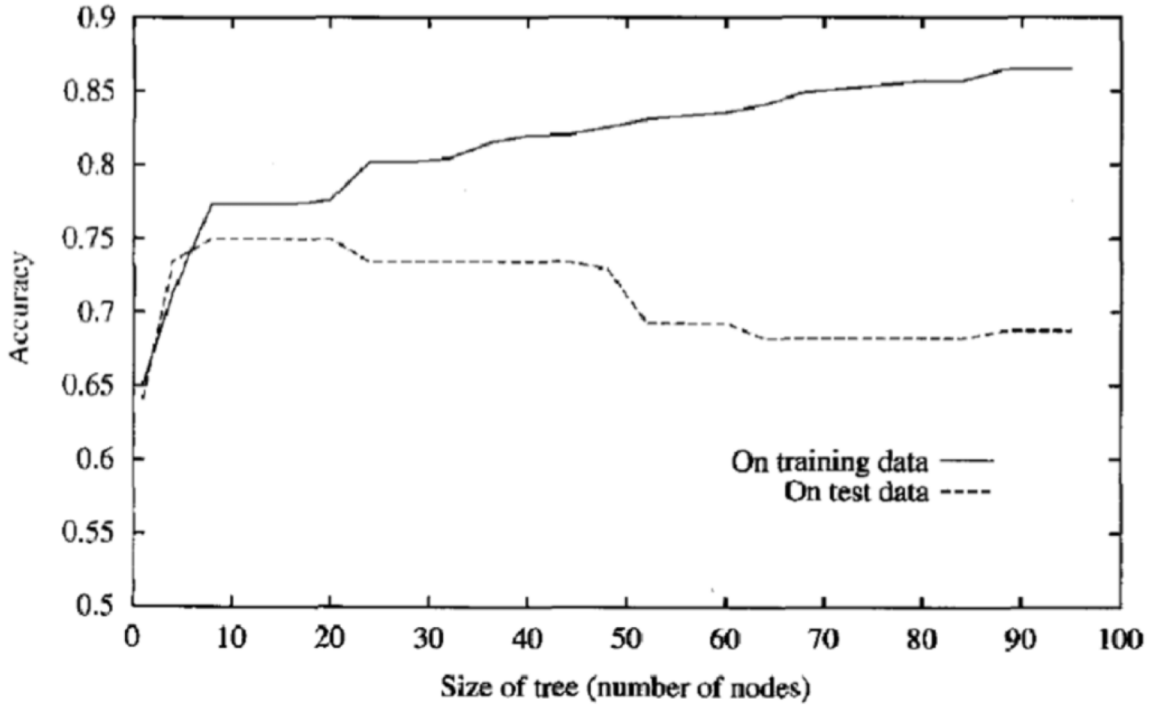
The problem with decision trees. Decision trees tend to have a high variance (variance of predictions).



If the train data was different, the resulting tree structure was very different, i.e., it tended to overfit the train data.

Decision Tree Problems

generalization error = model complexity (variance) + deviation between true function and model (bias)^2 + noise

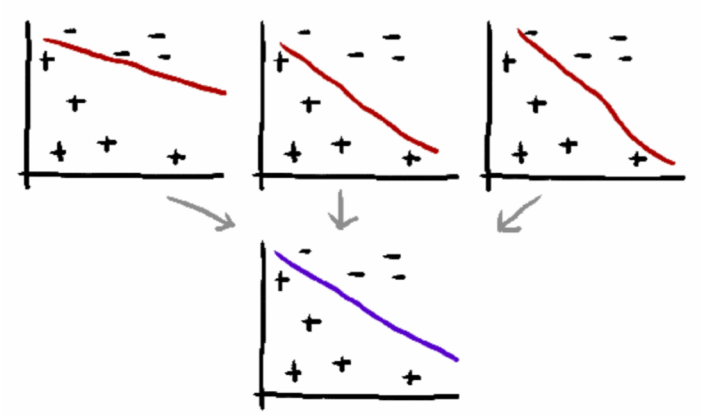


This is where bagging is proposed. It can reduce the variance, which tends to be high in a single decision tree.

Bagging

Bagging, applying the bootstrap method to high variance models to reduce variance.

https://www.kaggle.com/kashnitsky/topic-5-ensembles-part-1-bagging



Idea :.
Randomly resample the sample according to a discrete uniform distribution (bootstrap sampling), fit a decision tree to each subset, and obtain multiple decision tree results. Finally, do a majority vote (or average for regression).
Example : https://www.kaggle.com/mayankkestwal10/ensemble-learning-bagging-and-boosting

Assumptions : Proposed by Leo Brayman in 1994.
http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf

Bagging

Bagging & RandomForest

RandomForest is an improved version of Bagging, which adds feature sampling to the Bagging algorithm.

RandomForest

RandomForest, which lowers the correlation of the tree to reduce the variance of bagging.

Predicted value　　Random Forest Trees

$$f_{\mathrm{rf}}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T(x;\Theta_b)$$

Number of specimens

Correlation coefficient between variables

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Follow an independent homodistribution with variance σ2

Variance of the mean of B predictions

Number of specimens

Idea :.

Before each segmentation, randomly select m features from the features and use them as candidates for segmentation.

Case : 1.

Assumption : To begin with.

In Bagging, there is a high correlation between each decision tree created by the bootstrap method. On the other hand, RandomForest, which generates many decision trees with different features to be selected by branching, has a lower correlation between decision trees, so the first term of the variance formula becomes smaller and the variance (variance) decreases, resulting in a higher generalization performance than Bagging.

http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf

RandomForest

$$Var\left(\underbrace{\frac{1}{B}\sum_{i=1}^{B} T_i(c)}_{\bar{X}}\right) = \frac{1}{B^2}\sum_{i=1}^{B}\sum_{j=1}^{B} Cov(T_i(x), T_j(x))$$

$\bar{X}$: $\frac{1}{B}\sum_{i=1}^{B} X_i$, with $X_i = T_i(c)$

$\bar{X}$ → $Var(\bar{X})$

$$= \frac{1}{B^2}\sum_{i=1}^{B}\left(\sum_{j\neq i}^{B} Cov(T_i(x), T_j(x)) + Var(T_i(x))\right)$$

i=j

$$= \frac{1}{B^2}\sum_{i=1}^{B}\left((B-1)\sigma^2 \cdot \rho + \sigma^2\right)$$

$$= \frac{B(B-1)\rho\sigma^2 + B\sigma^2}{B^2}$$

$$= \frac{(B-1)\rho\sigma^2}{B} + \frac{\sigma^2}{B}$$

$$= \rho\sigma^2 - \frac{\rho\sigma^2}{B} + \frac{\sigma^2}{B}$$

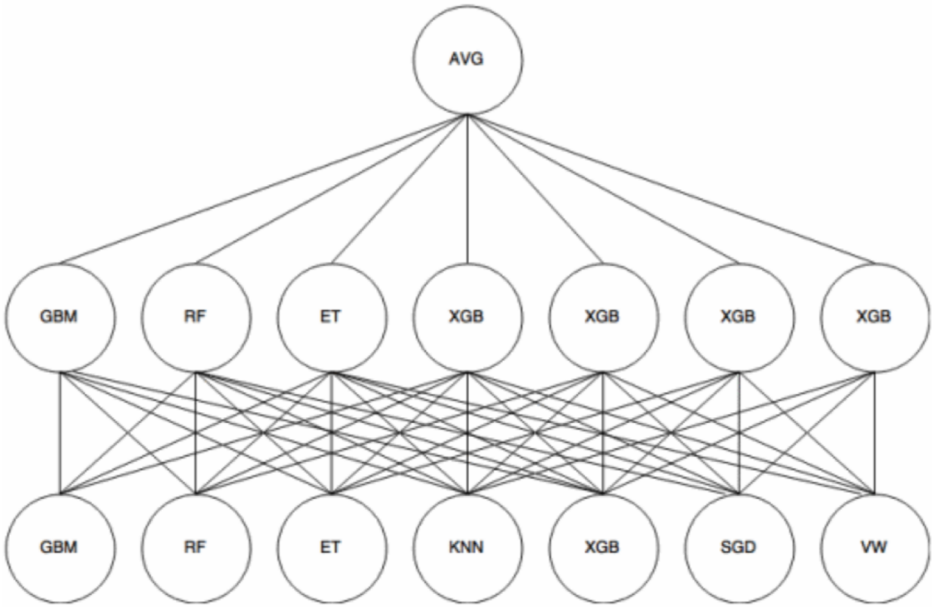
$$= \rho\sigma^2 + \sigma^2\frac{1-\rho}{B}$$

Decreaes, if ρ decreases, i.e., if m decreases

Decreases, if number of trees B increases (irrespective of ρ)

Blending & Stacking

Blending & Stacking, which unlike **Bagging/RandomForest**, combines different predictive models.

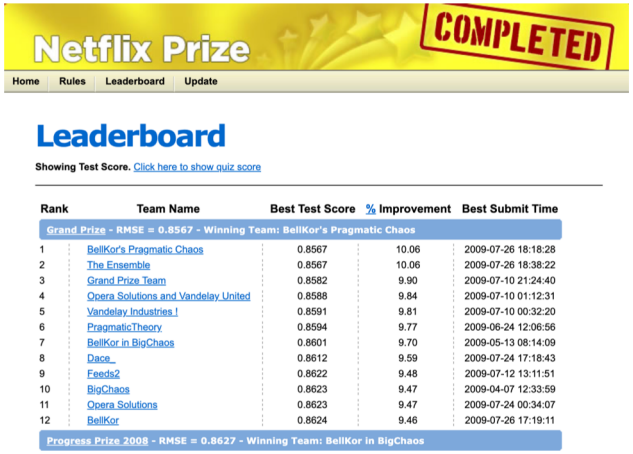


We want to get enough diversity by using different prediction models, different features, different parts of the training data, and different parameters.
Combination methods include majority rule, average, maximum, minimum....

Blending

Mix and match predictions in some way to improve them.

https://www.quora.com/What-is-blending-in-machine-learning
https://mlwave.com/kaggle-ensembling-guide/

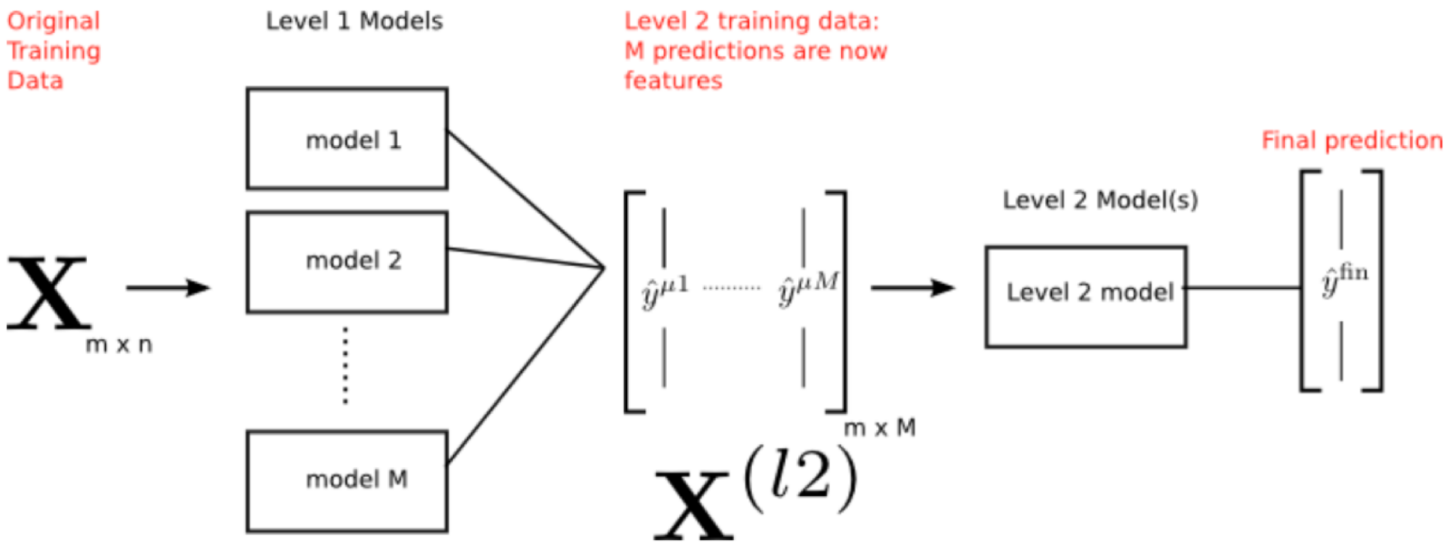


Idea : Use only 10% holdout data as input to the later model.
Or simply use the weighted average of each model. ...and so on, in multiple senses of the word.
https://www.datarobot.com/wiki/training-validation-holdout/
https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/discussion/44588
Example: https://github.com/ghmagazine/kagglebook/blob/master/ch01/ch01-01-titanic.py

Assumption: First used by the winner of the "Netflix Prize" (an algorithm contest) held by Netflix; often compared to Stacking, but simpler and less likely to leak information.
https://www.netflixprize.com/

Stacking

Stacking models to improve prediction



Idea : Use the output of the previous model as input for the later model.
Example : https://www.kaggle.com/yekenot/simple-stacker-lb-0-284

Assumption: Using train data to make predictions in a sample (validation) is equivalent to answering a question based on having seen similar questions beforehand, and there is a risk of overfitting. This problem can be avoided.

Sample code

How to solve problems
“Scratch Ensemble Learning”

[Problem 1] Scratch implementation of Blending
[Problem 2] Scratch implementation of Bagging
[Problem 3] Scratch implementation of Stacking

Ensemble module of scikit-learn

Let’s first have a look at the ensemble module of the scikit-learn library.

Scikit-learn’s Ensemble module

Sprint 8 – Ensemble Learning

Explanation about this Sprint is given but please try it on your own first.

Sprint 8 – Ensemble Learning

Please work on your own after class and submit your assignments on DIVER.

Sprint 8 – Ensemble Learning

A Sample Code of this Sprint is given but please try it on your own.

Sprint 8 – Ensemble Learning

Please work on your own after class and submit your assignments on DIVER.

ToDo by next class

Next class will be Zoom : Thursday June 24, 2021 19:30~20:30

🚨ToDo: Sprint Neural Network
https://diver.diveintocode.jp/curriculums/1876

Check-out

3 minutes Please post the following point to Zoom chat.

Q. What do you do to graduate on time

Bonus

```
# Model samples
model =     {"LinearRegression": LinearRegression(),
             "Ridge": Ridge(),
             "Lasso": Lasso(),
             "ElasticNet": ElasticNet(),
             "Polynomial_deg2": Pipeline([('poly', PolynomialFeatures(degree=2)),('linear', LinearRegression())]),
             "Polynomial_deg3": Pipeline([('poly', PolynomialFeatures(degree=3)),('linear', LinearRegression())]),
             "Polynomial_deg4": Pipeline([('poly', PolynomialFeatures(degree=4)),('linear', LinearRegression())]),
             "Polynomial_deg5": Pipeline([('poly', PolynomialFeatures(degree=5)),('linear', LinearRegression())]),
             "KNeighborsRegressor": KNeighborsRegressor(n_neighbors=3),
             "DecisionTreeRegressor": DecisionTreeRegressor(),
             "RandomForestRegressor": RandomForestRegressor(),
             "SVR": SVR(kernel='rbf', C=1e3, gamma=0.1, epsilon=0.1),
             "GaussianProcessRegressor": GaussianProcessRegressor(),
             "SGDRegressor": SGDRegressor(),
             "MLPRegressor": MLPRegressor(hidden_layer_sizes=(10,10), max_iter=100, early_stopping=True, n_iter_no_change=5),
             "ExtraTreesRegressor": ExtraTreesRegressor(n_estimators=100),
             "PLSRegression": PLSRegression(n_components=10),
             "PassiveAggressiveRegressor": PassiveAggressiveRegressor(max_iter=100, tol=1e-3),
             "TheilSenRegressor": TheilSenRegressor(random_state=0),
             "RANSACRegressor": RANSACRegressor(random_state=0),
             "HistGradientBoostingRegressor": HistGradientBoostingRegressor(),
             "AdaBoostRegressor": AdaBoostRegressor(random_state=0, n_estimators=100),
             "BaggingRegressor": BaggingRegressor(base_estimator=SVR(), n_estimators=10),
             "GradientBoostingRegressor": GradientBoostingRegressor(random_state=0),
             "VotingRegressor": VotingRegressor([('lr', LinearRegression()), ('rf', RandomForestRegressor(n_estimators=10))]),
             "StackingRegressor": StackingRegressor(estimators=[('lr', RidgeCV()), ('svr', LinearSVR())], final_estimator=RandomForestRegressor(n_estimators=
             "ARDRegression": ARDRegression(),
             "HuberRegressor": HuberRegressor(),
             }
```

Thank You For Your Attention