Chapter 7. Ensemble Learning and Random Forests

Nidhin Pattaniyil

Table of Contents

- Voting Classifier
- Bagging and Pasting
- Random Forests
- Boosting
- Stacking

Introduction

- Ensemble: group of predictors
- Ensemble Learning: aggregate predictions of a group of predictors
- Ensemble method: ensemble learning algorithm
- Ensemble Methods:
    - Bagging
    - Boosting
    - Stacking
- Work best when predictors are independent from one another as possible

Voting Classifiers

Voting Classifiers

- Hard Voting Classifier: predict the class that gets the most vote

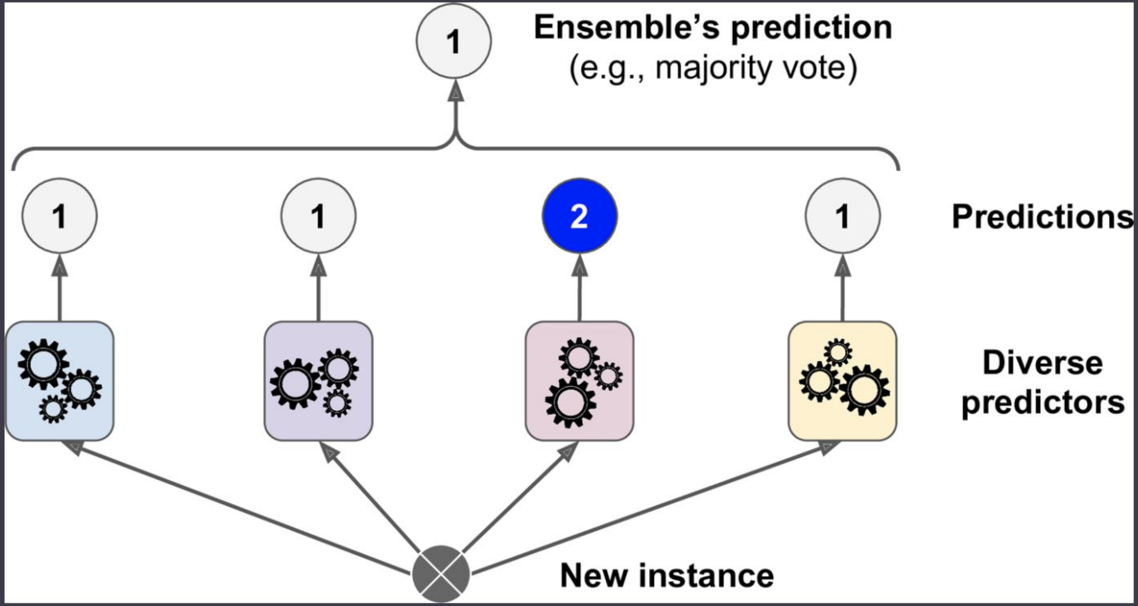
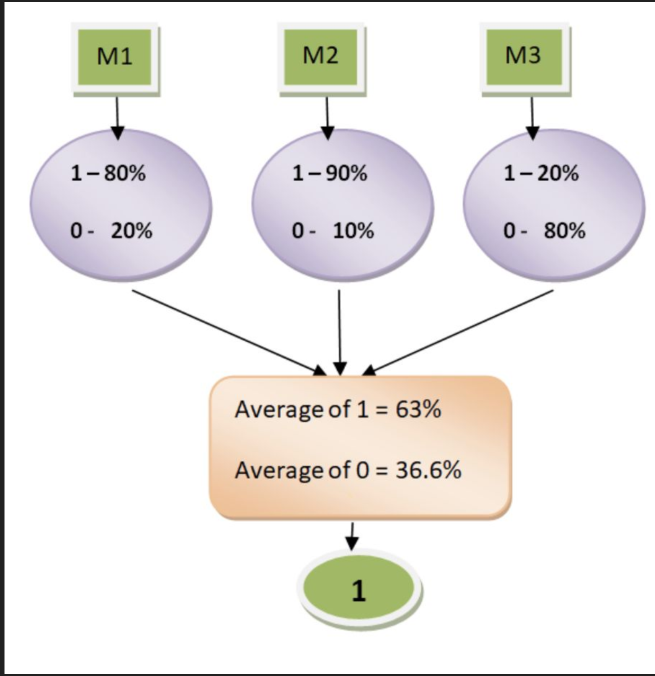


Figure 7-2. Hard voting classifier predictions

Voting Classifiers: Soft Voting

- clf1 -> [0.2, 0.8], clf2 -> [0.1, 0.9], clf3 -> [0.8, 0.2]
-
- With equal weights, the probabilities will get calculated as the following:
-
- Prob of Class 0 = 0.33*0.2 + 0.33*0.1 + 0.33*0.8 = 0.363
-
- Prob of Class 1 = 0.33*0.8 + 0.33*0.9 + 0.33*0.2 = 0.627
-
- The probability predicted by ensemble classifier will be [36.3%, 62.7%].



Logistic Regression: 0.864
RandomForestClassifier 0.896
SVC: 0.888

VotingClassifier 0.904

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression()
rnd_clf = RandomForestClassifier()
svm_clf = SVC()

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')
voting_clf.fit(X_train, y_train)
```

Bagging and Pasting

Bagging and Pasting

- Use same training algorithm but train on different random subsets of training set
- Two types:
    - Bagging: sampling with replacement;
    - Pasting: sampling without replacement
- Each individual predictor has a higher bias
- Ensemble has a similar bias but a lower variance than a single predictor trained on original training set

Bagging and Pasting

- Ensemble's prediction will likely generalize better than single Decision Tree

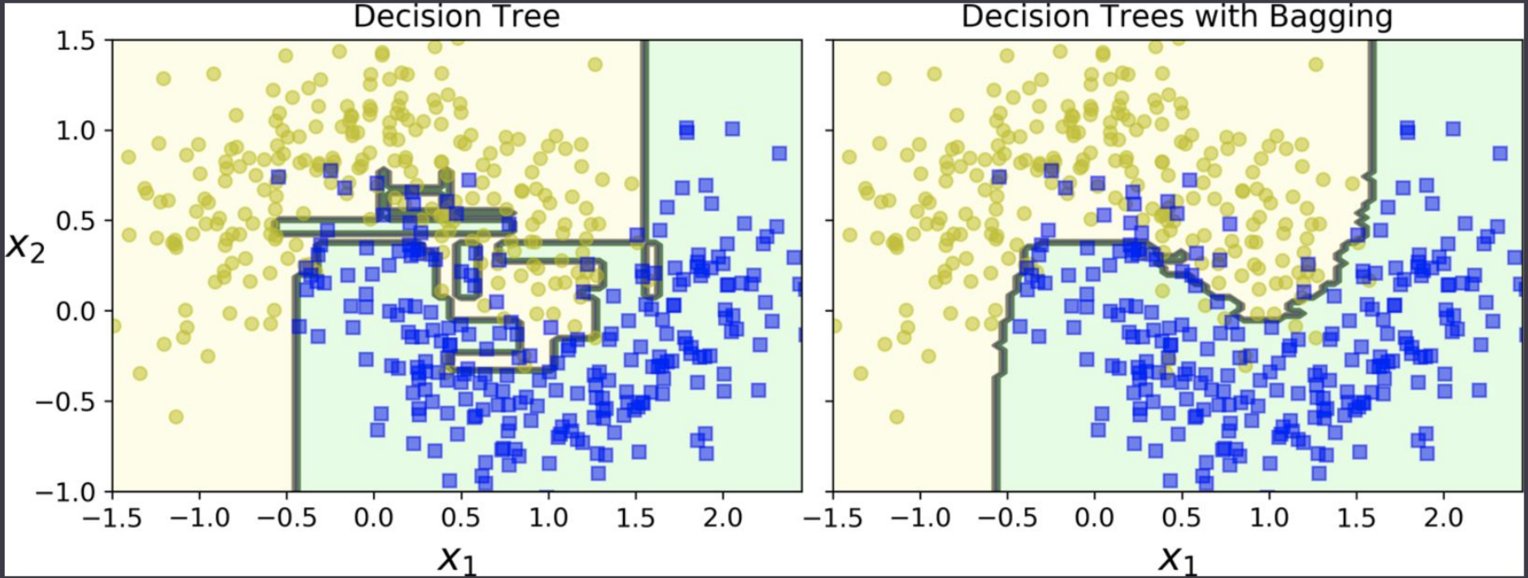
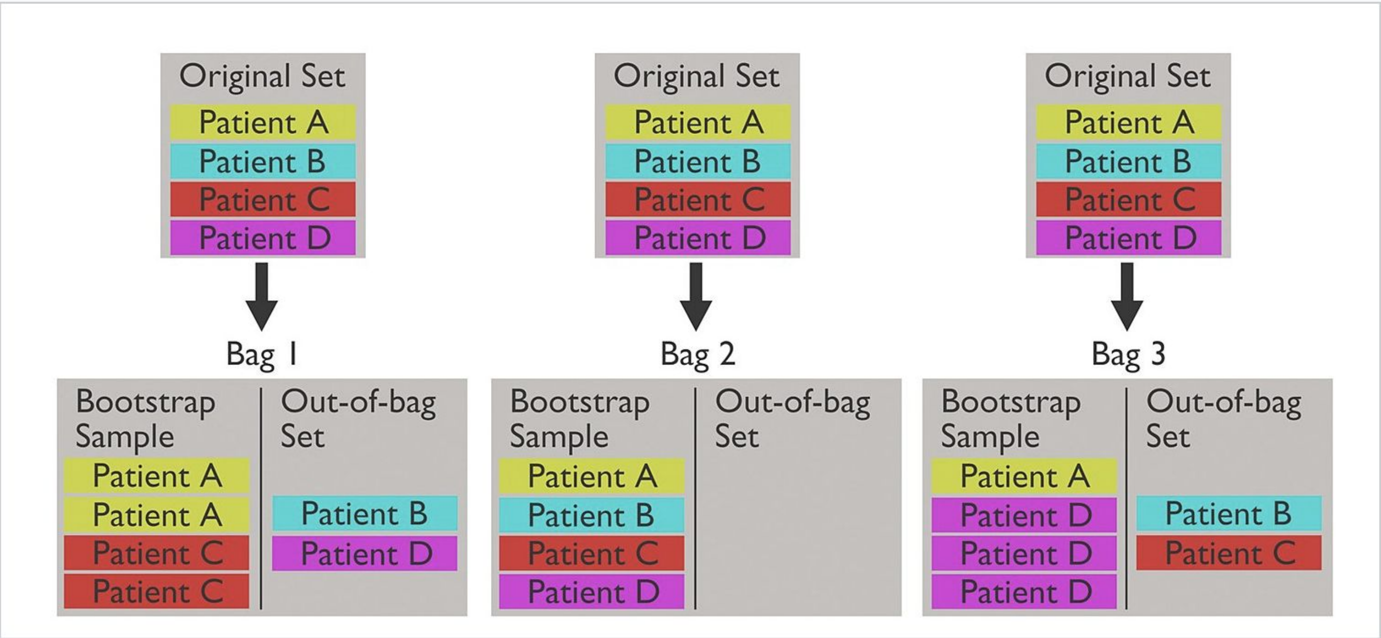


Figure 7-5. A single Decision Tree (left) versus a bagging ensemble of 500 trees (right)

Out-of-bag Dataset



Visualizing the bagging process. Sampling 4 patients from the original set with replacement and showing the out-of-bag sets. Only patients in the bootstrap sample would be used to train the model for that bag.

[Wikipedia: Out-of-bag error]

Random Patches and Random Subspaces

- Sample features (bootstrap_features and max_features)
- Sample records: (bootstrap and max_samples)
- Random Patches
  - Sampling both training instances and features
- Random Subspaces
  - Keeping all training instances but sampling features
- Sampling features results in even more predictor diversity, trading a bit more bias for a lower variance.

Random Forests

Random Forest

- Ensemble of Decision Trees trained via bagging
- If using a BaggingClassifier of DecisionTreeClassifier, you could just use RandomForestClassifier
- All the hyperparameters of DecisionTree and BaggingClassifier
- at each split , it only searches for the best feature among a random subset of features..
- leads to greater tree diversity thus higher bias, low variance
-

Extra-Trees

- Faster to train than RandomForest
- Extra Trees uses random thresholds instead of searching for best threshold

Feature Importance

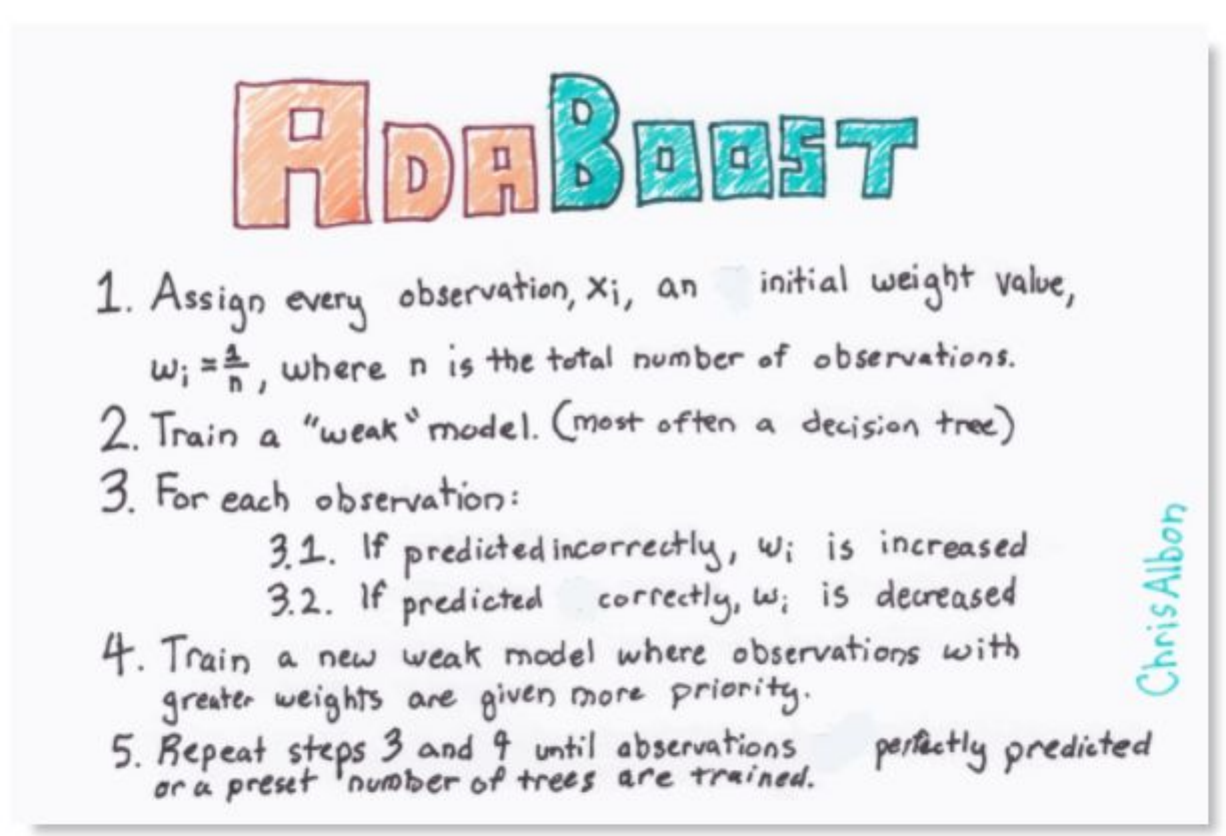
- For each feature we can collect how on average it decreases the impurity.
- The average over all trees in the forest is the measure of the feature importance.
- weighted average, where each node’s weight is equal to the number of training samples that are associated with it

Boosting

Boosting

- In Random Forest, all the trees can be independently trained
- Train predictors sequentially , each trying to correct its predecessors error
- Two popular boosting methods:
  - AdaBoost : increase misclassified instance weight at each iteration
  - Gradient Boosting: new predictor trained on residual errors of previous predictor

AdaBoost



AdaBoost
1. Assign every observation, x_i, an initial weight value, $w_i = \frac{1}{n}$, where n is the total number of observations.
2. Train a "weak" model. (most often a decision tree)
3. For each observation:
3.1. If predicted incorrectly, w_i is increased
3.2. If predicted correctly, w_i is decreased
4. Train a new weak model where observations with greater weights are given more priority.
5. Repeat steps 3 and 4 until observations perfectly predicted or a preset number of trees are trained.
ChrisAlbon

Gradient Boosting (step 0)

- Trying to predict income

ID	Age	City	Income
1	32	A	51000
2	30	B	78000
3	21	A	20000
4	27	B	44000
5	36	B	89000
6	25	A	37000
7	47	A	56000
8	54	B	92000

TARGET Income	PREDICTION	RESIDUAL
51000		
78000		
20000		
44000		
89000		
37000		
56000		
92000		

Reference:
https://www.analyticsvidhya.com/blog/2021/03/gradient-boosting-machine-for-data-scientists/

Gradient Boosting (step 1)

- Train model 1
- compute predictions

ID	Age	City	Income
1	32	A	51000
2	30	B	78000
3	21	A	20000
4	27	B	44000
5	36	B	89000
6	25	A	37000
7	47	A	56000
8	54	B	92000

TARGET	PREDICTION	RESIDUAL
Income	Predictions	
51000	53500	
78000	61000	
20000	28500	
44000	61000	
89000	90500	
37000	28500	
56000	53500	
92000	90500	

Reference: https://www.analyticsvidhya.com/blog/2021/03/gradient-boosting-machine-for-data-scientists/

Gradient Boosting (step 2)

- Using the predictions , compute residual
- Save model 1 predictions

ID	Age	City	Income	Model 1 Income
1	32	A	51000	53500
2	30	B	78000	61000
3	21	A	20000	28500
4	27	B	44000	61000
5	36	B	89000	90500
6	25	A	37000	28500
7	47	A	56000	53500
8	54	B	92000	90500

TARGET	PREDICTION	RESIDUAL
Income	Predictions	Error
51000	53500	-2500
78000	61000	17000
20000	28500	-8500
44000	61000	-17000
89000	90500	-1500
37000	28500	8500
56000	53500	2500
92000	90500	1500

Reference: https://www.analyticsvidhya.com/blog/2021/03/gradient-boosting-machine-for-data-scientists/

Gradient Boosting (step 3)

- Train a new model where the target is the error from model 1
- Save model 1 predictions
- Repeat for further models

ID	Age	City	Income	Model 1 Income	Model 2 Income
1	32	A	51000	53500	48000
2	30	B	78000	61000	69000
3	21	A	20000	28500	23000
4	27	B	44000	61000	56700
5	36	B	89000	90500	98500
6	25	A	37000	28500	36500
7	47	A	56000	53500	49200
8	54	B	92000	90500	86200

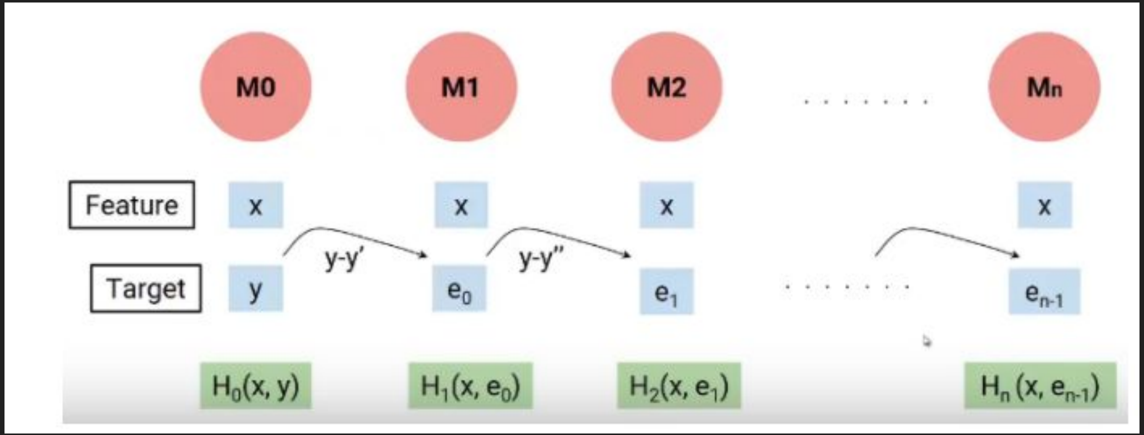
TARGET	PREDICTION	RESIDUAL
Error	Predictions	
-2500	-5500	
17000	8000	
-8500	-5500	
-17000	-4300	
-1500	8000	
8500	8000	
2500	-4300	
1500	-4300	

Model 2 Income = Model 1 Income + Predicted Errors

Reference: https://www.analyticsvidhya.com/blog/2021/03/gradient-boosting-machine-for-data-scientists/

Gradient Boosting

- Model 0: predicts the target
- Model 1 and above, target is the previous error



Reference: https://www.analyticsvidhya.com/blog/2021/03/gradient-boosting-machine-for-data-scientists/

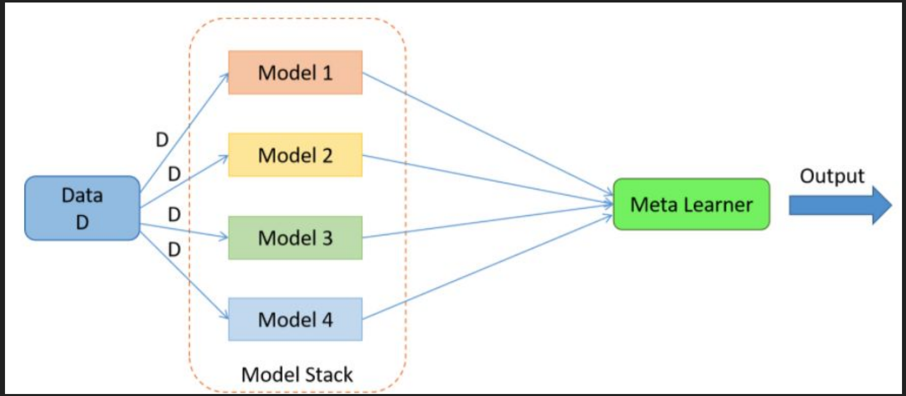
Gradient Boosting

- XGBoost, LightGBM, Catboost are other popular libraries
- Gradient Boosting also used for ranking

Stacking

Stacking

- Instead of using hard voting, train a model to perform the aggregating
- Training
    - Create a hold out dataset
    - Train classifiers on split 1
    - Get output from classifier on split 2 and use as training data
    - Blender is trained from first layers predictions



Summary

- Ensemble methods: Bagging / Boosting / Stacking
- Voting: Hard or Soft Voting
- Sample Training Data / Sample Features
- Random Forests: Bagging Tree Classifier ; feature importance, OOB score
- Boosting: AdaBoost / Gradient Boosting
- Stacking: model to perform aggregation