Ensemble methods

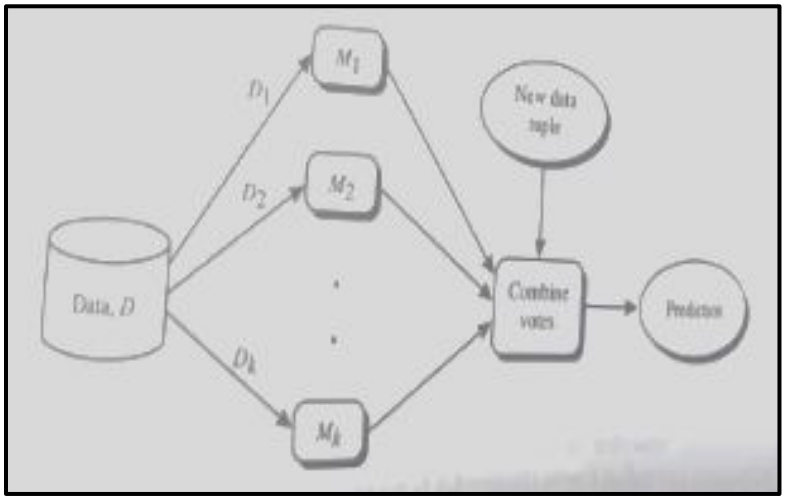
Ensemble methods

- **Ensemble methods** is a machine learning technique that combines several base models in order to produce one optimal predictive model.
- Machine learning models suffer bias and/or variance
- Bias is the difference between the predicted value and actual value by the model.
- (1)Bias is introduced when the model doesn’t consider the variation of data and creates a simple model. The simple model doesn’t follow the patterns of data, and hence the model gives errors in predicting training as well as testing data i.e. the model with high bias and high variance
- (2)When the model follows even random quirks of data, as pattern of data, then the model might do very well on training dataset i.e. it gives low bias, but it fails on test data and gives high variance.

Ensemble methods

An Ensemble combines a series of

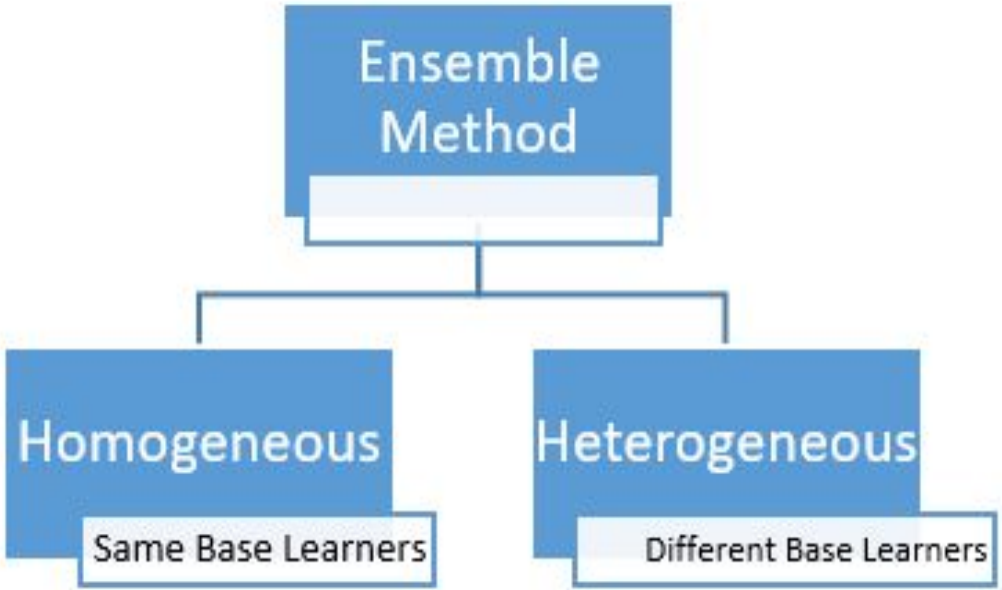
- K learned models or base classifiers, M_1,M_2…., M_k with the aim of creating an improved composite classification model, M*
- Given Dataset, D is used to create k training sets D_1,D_2…D_k where Di(1<=i<=k-1) is used to generate classifier Mi



Ensemble methods

- Therefore, to improve the accuracy (estimate) of the model, ensemble learning methods are developed.
- Ensemble is a machine learning concept, in which several models are trained using machine learning algorithms.
- It combines low performing classifiers (also called as weak learners or base learner) and combine individual model prediction for the final prediction.
- On the basis of type of base learners, ensemble methods can be categorized as **homogeneous** and **heterogeneous** ensemble methods.
- If base learners are same, then it is a homogeneous ensemble method. If base learners are different then it is a heterogeneous ensemble method.

Ensemble methods



Types of Ensemble Methods

Ensemble techniques are classified into three types:

1. ***Bagging***
2. ***Boosting***
3. ***Stacking***

Bagging is a way to decrease the variance in the prediction by generating additional data for training from dataset using combinations with repetitions to produce multi-sets of the original data.

Boosting is an iterative technique which adjusts the weight of an observation based on the last classification. If an observation was classified incorrectly, it tries to increase the weight of this observation. Boosting in general builds strong predictive models.

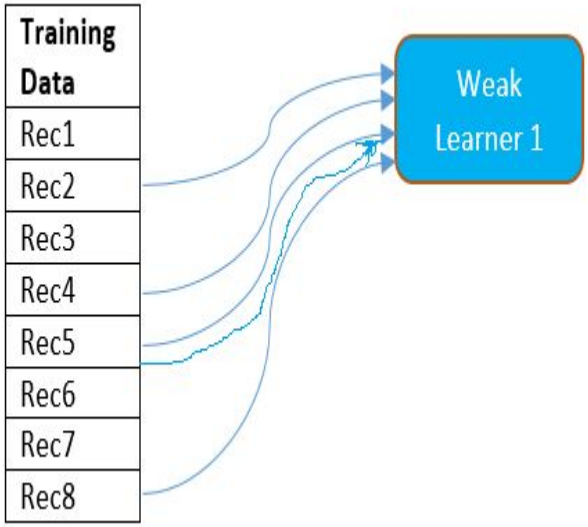
*1.Bagg*ing, or *B*ootstrap *Agg*regating.

- **BAGG**ing gets its name because it combines ***B***ootstrapping and ***Agg***regation to form one ensemble model.
- Given a sample of data, multiple bootstrapped subsamples are pulled.

*1.Bagg*ing, or *B*ootstrap *Aggr*egating

Concept:

- Consider a scenario where we are looking at the users' ratings for a product. Instead of approving one user's good/bad rating, we consider average rating given to the product.
- With average rating, we can be considerably sure of quality of the product. Bagging makes use of this principle.
- Instead of depending on one model, it runs the data through multiple models in parallel, and average them out as model's final output.
- Bagging is an acronym for **Bootstrapped Aggregation**.
- Bootstrapping means random selection of records with replacement from the training dataset. 'Random selection with replacement' can be explained as follows:



*1.Bagg*ing, or *B*ootstrap *Aggr*egating

1. Consider that there are 8 samples in the training dataset. Out of these 8 samples, every weak learner gets 5 samples as training data for the model. These 5 samples need not be unique, or non-repetitive.
2. The model (weak learner) is allowed to get a sample multiple times. For example, as shown in the figure, Rec5 is selected 2 times by the model. Therefore, weak learner1 gets Rec2, Rec5, Rec8, Rec5, Rec4 as training data.
3. All the samples are available for selection to next weak learners. Thus all 8 samples will be available for next weak learner and any sample can be selected multiple times by next weak learners.

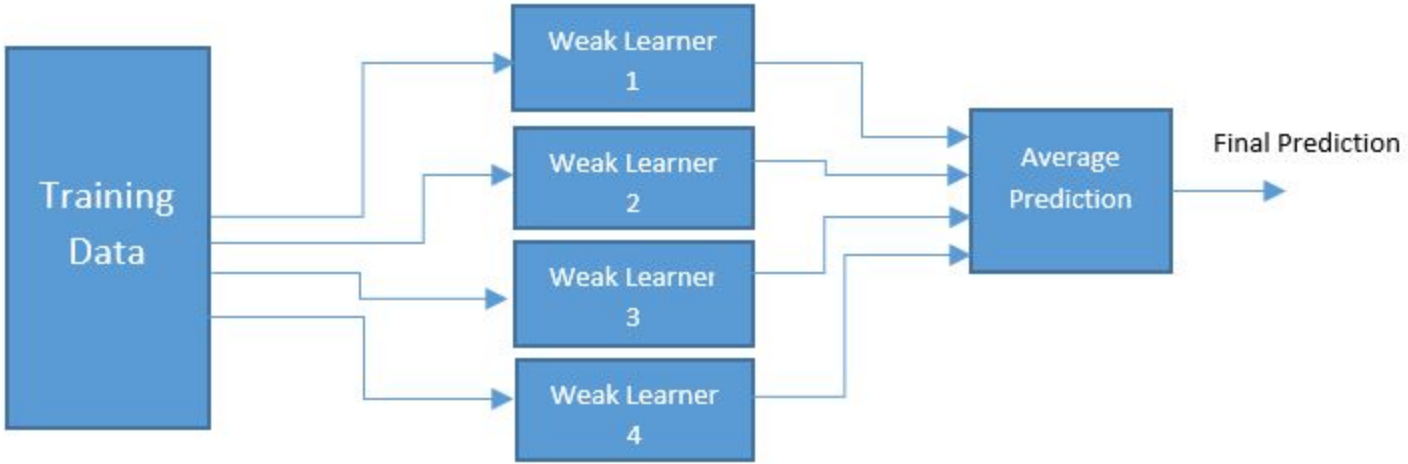
*1.Bagg*ing, or *B*ootstrap *Agg*regating

Bagging Steps:

- Suppose there are N observations and M features in training data set. A sample from training data set is taken randomly with replacement.
- A subset of M features are selected randomly and whichever feature gives the best split is used to split the node iteratively.
- The tree is grown to the largest.
- Above steps are repeated n times and prediction is given based on the aggregation of predictions from n number of trees.

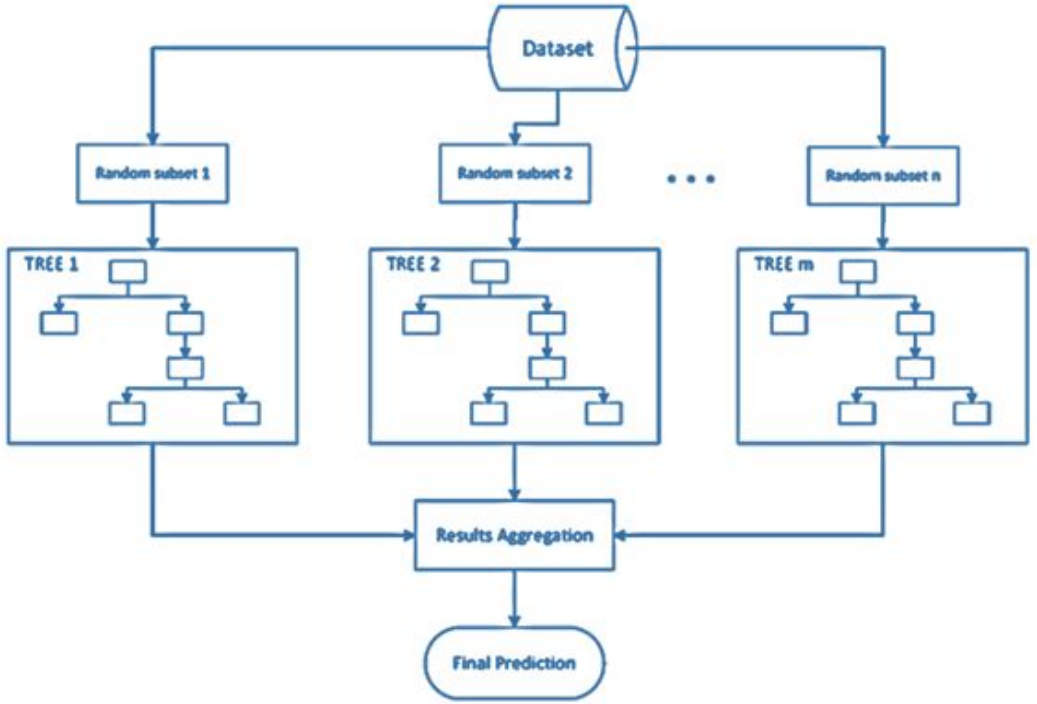
*1.Bagg*ing, or *B*ootstrap *Aggr*egating

Bagging is a **parallel method**, which means several weak learners learn the data pattern independently and simultaneously.



*1.BAGG*ing, or *B*ootstrap *AGG*regating

Given a Dataset, bootstrapped subsamples are pulled. A Decision Tree is formed on each bootstrapped sample. The results of each tree are aggregated to yield the strongest, most accurate predictor



*1.Bagg*ing, or *B*ootstrap *Aggr*egating

- The output of each weak learner is averaged to generate final output of the model.
- Since the weak learner's outputs are averaged, this mechanism helps to reduce variance or variability in the predictions. However, it does not help to reduce bias of the model.
- Since final prediction is an average of output of each weak learner, it means that each weak learner has equal say or weight in the final output.

To summarize:

1. Bagging is Bootstrapped Aggregation
2. It is Parallel method
3. Final output is calculated by averaging the outputs produced by individual weak learner
4. Each weak learner has equal say
5. Bagging reduces variance

*1.BAGG*ing, or *B*ootstrap *AGG*regating

Advantages:

- Reduces over-fitting of the model.
- Handles higher dimensionality data very well.
- Maintains accuracy for missing data.

Disadvantages:

- Since final prediction is based on the mean predictions from subset trees, it won't give precise values for the classification and regression model.

Boosting

We saw that in bagging every model is given equal preference, but if one model predicts data more correctly than the other, then higher weightage should be given to this model over the other.

Also, the model should attempt to reduce bias.

Boosting is used to create a collection of predictors. In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analysing data for errors. Consecutive trees (random sample) are fit and at every step, the goal is to improve the accuracy from the prior tree. When an input is misclassified by a hypothesis, its weight is increased so that next hypothesis is more likely to classify it correctly. This process converts weak learners into better performing model.

These concepts are applied in the second ensemble method that we are going to learn, that is Boosting.

Boosting

Concept:

1. To start with, boosting assigns equal weights to all data points as all points are equally important in the beginning. For example, if a training dataset has N samples, it assigns weight = 1/N to each sample.
2. The weak learner classifies the data. The weak classifier classifies some samples correctly, while making mistake in classifying others.
3. After classification, sample weights are changed. Weight of correctly classified sample is reduced, and weight of incorrectly classified sample is increased. Then the next weak classifier is run.
4. This process continues until model as a whole gives strong predictions.

Note: Adaboost is the ensemble learning method used in binary classification only.

Boosting

- In this, weights are also assigned to each training tuple.
- A series of k classifiers is iteratively learned
- After a classifier Mi is learned, the weights are updated to allow the subsequent classifier , Mi+1, to pay more attention to the training tuples that were misclassified by Mi.
- The final boosted classifier ,M*, combines the votes of each individual classifier, where the weights of each classifier s vote is a function of its accuracy.

Boosting

Boosting Steps:

- Draw a random subset of training samples d1 without replacement from the training set D to train a weak learner C1
- Draw second random training subset d2 without replacement from the training set and add 50 percent of the samples that were previously falsely classified/misclassified to train a weak learner C2
- Find the training samples d3 in the training set D on which C1 and C2 disagree to train a third weak learner C3
- Combine all the weak learners via majority voting.

Boosting : Adaboost

- Adaboost, short of Adaptive boosting
- Suppose we want to boost the accuracy of the learning method, given dataset D of d class labeled tuples , (X1,Y1), (X2, Y2)....where yi is the class label of tuple Xi.
- Initially, Adaboost assigns equal weight of 1/d to each training sample.
- Generating k classifiers for the ensemble requires k rounds of the algorithm.
- For round i, the tuples from D are sampled to form a training set, Di, of size d.
- Sampling with replacement is used - the same tuple may be selected more than once.
- Each tuples chance of selection is based on its weight.

Boosting : Adaboost

- A classifier model,Mi, is derived from the training tuples of Di
- Its error is then calculated using Di as the as a test set.
- The weights of he training tuples are then adjusted according to how they were classified.
- If a tuple was incorrectly classified, its weight is increased
- If a tuple was correctly classified, its weight is decreased
- A tuples weight reflects how easy or difficult it is to classify, higher the weight , the more it has been misclassified.
- These weights would be used to generate the training samples for the classifier for next round

Boosting : Adaboost

- Basic idea is when you build a classifier, the focus needs to be more on the misclassified tuples of the previous round.
- To compute the error rate of model Mi, the sum of the weights of each of the tuples in Di that Mi misclassified are taken .

$$error(M_i) = \sum_{j=1}^{d} w_j \times err(\mathbf{X}_j),$$

Where, err(Xj) is the misclassification error of tupl Xj.

If the tuple was misclassified, then err(Xj) is 1, else 0

If the performance of the classifier is so poor that its errornexceeds 0.5, it is abandoned.

Boosting : Adaboost

- Lower the classifiers error rate, more accurate it is.Therefore, higher its weight for voting should be.
- The weight of classifier M is vote is :-

$$log \frac{1 - error(M_i)}{error(M_i)}.$$

- For each class,c, we sum the weights of each classifier that assigned class c to X.
- The class with the highest sum is the winner and is returned as the class prediction for tuple X.

Boosting : Adaboost

Consider below training data for heart disease classification.

Blocked Arteries	Chest Pain	Weight	Heart Disease
Y	Y	200	Y
Y	N	185	Y
N	Y	200	Y
Y	Y	160	Y
Y	N	140	N
Y	N	130	N
N	Y	170	N
Y	Y	170	N

1. Initialize Weights To All Training Points

First step is to assign equal weights to all samples as all samples are equally important. Always, sum of weights of all samples equals 1. There are 8 samples, so each sample will get weight = 1/8 = 0.125

Since all samples are equally important to start with, all samples get equal weight: 1 / total number of samples = 1/8

Boosting : Adaboost

Blocked Arteries	Chest Pain	Weight	Heart Disease	Weights Assigned
Y	Y	200	Y	1/8
Y	N	185	Y	1/8
N	Y	200	Y	1/8
Y	Y	160	Y	1/8
Y	N	140	N	1/8
Y	N	130	N	1/8
N	Y	170	N	1/8
Y	Y	170	N	1/8

2. Create Stump

After assigning weight, next step is to create stumps.

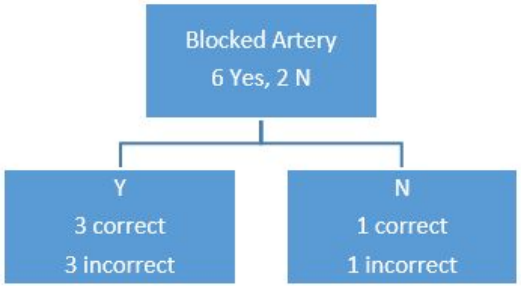
Stump is a decision tree with one node and two leaves.

Adaboost creates forest of decision stumps.

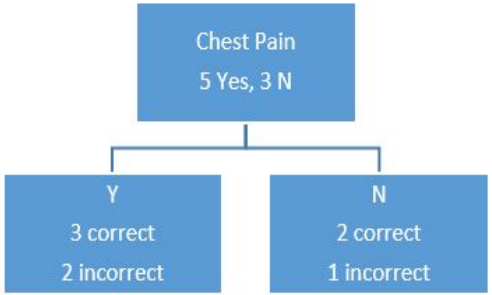
To create stump, only one attribute should be chosen. But, it is not randomly selected. The attribute that does the best job of classifying the sample is selected first.

So let's see how each attribute classifies the samples.

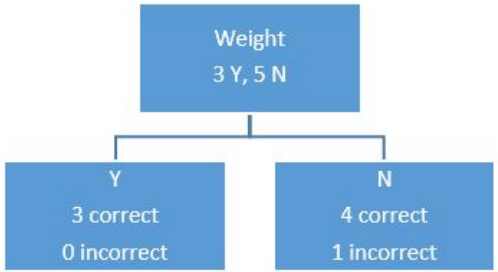
Blocked Artery:



Chest Pain



Weight:



Weight is continuous quantity, and there is a method to find threshold for continuous quantity in decision tree algorithm. According to that method, the threshold is 175. Based on this condition draw the decision stump.

From above classification, it can be seen that

- **'Blocked Arteries' as a stump made 4 errors.**
- **'Chest Pain' as a stump made 3 errors.**
- **'Weight' as a stump made 1 error.**

Since, the attribute 'Weight' made least number of errors, 'Weight' is the best stump to start with.

Boosting

Advantages:

- **Supports different loss function (we have used ‘binary:logistic’ for this example).**
- **Works well with interactions.**

Disadvantages:

- **Prone to over-fitting.**
- **Requires careful tuning of different hyper-parameters.**

Comparison : Boosting Vs Bagging

Boosting	**Bagging**
Focuses on misclassified tuples	There is little correlation among classifiers
Weights are assigned to each training tuple	It combine a series of K learned models or base classifiers.
Risks overfitting the resulting composite model to such data	Model is less susceptible to overfitting
Tends to achieve greater accuracy	Lower accuracy as compared to boosting
Eg: Weigh the opinion of doctors.	Eg: Opinion of doctors taken