Ensemble Learning

"We unfortunately live in a corporate world where group decision making is made to avoid failure rather than to achieve success."

Bill Cahan

Ensemble learning is a machine learning model which combines the predictions of several machine learning models to improve the prediction accuracy of using a single classifier. In other words, ensemble models have been developed to overcome the limitations of using a single-hypothesis machine learning model. The general principle of this model is to ensemble hypotheses of different models either by combining the collective predictions of different models into a single final model or by employing different models in a sequential fashion to increase the accuracy and robustness of learning. Ensemble models are useful to solve both classification and regression problems where they reduce bias and variance to boost the accuracy of models. They are used in many real-world applications, such as remote sensing, security, gene classification, image segmentation, video retrieval, healthcare, and so on.

Learning Objectives

- Understand the basics of ensemble learning, its advantages, and disadvantages
- Study the different ensembling techniques such as parallel ensemble models, sequential ensemble models and incremental ensemble models
- Gain knowledge about parallel ensemble models like Voting, Bootstrap resampling and Bagging and explore the popular bagging technique called Random Forest
- Learn about incremental ensemble models such as Stacking and Cascading
- Acquire knowledge about sequential ensemble models and explore the popular boosting technique called AdaBoost model

12.1 INTRODUCTION

Ensembling is a process of combining many weak learners to produce a strong learner that performs better than a single weak learner. The idea is to learn better with a set of models, instead of making one model to learn better. It is like getting opinions from many in order to decide and to protect oneself in decision-making.

A very simple way of ensembling is majority voting or taking average or weighted average of predictions of multiple machine learning models or base learners. 'Stacking' is another way of ensembling that uses the predictions of multiple models as 'features' to train a new model and then predict from that model. Another approach is to employ the same machine learning model multiple times with different datasets and combine the hypothesis generated from these models.

Bagging and Boosting are two popular models of ensemble learning. Bagging is a method of ensembling simultaneously many independent models and averaging the predictions obtained from the different models, which in fact, is more robust than the prediction made by a single classifier. Thus, bagging solves the overfitting problem which normally happens with a single classifier. On the other hand, boosting employs different classifiers in a sequential fashion where the prediction made by a classifier is enhanced by boosting the weights for the data instances and consecutively the predicted output is given as input to the next learning model. Thus, boosting increases the accuracy of predictions made by the previous learning model. Bagging minimizes the variance of the learning model, whereas boosting increases the accuracy of prediction.

Ensemble learning is an approach to minimize bias and variance and to build accurate models that avoid over-fitting and under-fitting of learning models. Even, cross-validation can be done to train on many datasets and average their predictions to ensemble different predictions from many models. Fine tuning a machine learning model will improve the accuracy of the model. For example, fine tuning the parameters of the model, which mean building a random forest classifier to tune the number of trees to build and the number of variables to choose for splitting, etc. Ensembling is to combine various fine tuned individual models that improve the performance of the model. On the other hand, there is always a tradeoff between the accuracy and complexity of the ensemble models.

To perform ensemble learning, different subsets of data are generated from the training dataset where each subset contains data instances that are chosen randomly with replacement. These subsets are given as input to different learning models simultaneously in bagging and sequentially in boosting to train the classifier. Thus, ensembling can be done in different ways such as using different machine learning models, different hypothesis, different hyperparameters, different population (data sets), different initial seed and different features, etc.

Some of the advantages and disadvantages of ensemble learning are listed below:

Advantages of Ensemble Learning

1. Minimizes bias and variance of a weak learner
2. Reduces overfitting and underfitting of the model
3. Increases accuracy of prediction
4. Handles higher dimensionality data efficiently

Disadvantages of Ensemble Learning

1. Does not give precise values for the classification and regression model
2. Error prone and sometimes leads to overfitting
3. Requires careful tuning of the number and type of features used

Let us study about all these approaches of ensembling in a detailed manner in the further sections.

12.1.1 Ensembling Techniques

Ensemble models are generally classified as 'Parallel' ensemble models and 'Sequential' ensemble models based on how the base learners are ensembled for training. In a parallel ensemble mode, the base learners learn simultaneously and independently, whereas in a sequential ensemble mode the base learners are used sequentially to learn. Ensembling models can also be classified as 'Homogenous' and 'Heterogenous' ensemble models, depending on the type of base learners used in each stage. If same type of base learner is used for ensembling, then such a model is known as a Homogenous ensemble model, while in case of a heterogenous ensemble model, different type of base learner is used for ensembling.

The parallel ensemble models include Voting, Bootstrap Resampling and Bagging. Random forest model is a popular ensemble model constructed based on the idea of bagging. Some of the sequential ensemble models are AdaBoost, Gradient Tree Boosting and XGBoost which are also called as boosting models. Stacking and Cascading belong to the family of incremental ensemble models. Parallel ensemble learners are time and space efficient, as compared to sequential ensemble learners.

12.2 PARALLEL ENSEMBLE MODELS

Parallel ensemble models provide a simple way of ensembling the predictions from multiple independent models and then applying averaging techniques such as normal average or weighted average or majority voting to estimate the final prediction.

Scan for illustrations of 'Types of Ensemble Models', 'Parallel Ensemble Model', 'Voting', 'Bootstrap Resampling', 'Working Principle of Bagging', Learning Mechanism through Stacking', and 'Sequential Ensemble Model'

12.2.1 Voting

It is a simple way of ensembling multiple learners. It is the process of combining multiple machine learning models or using the same model multiple times as base learners and taking average of their predictions in the case of regression problem and considering the maximum vote of their predictions in the case of classification problem.

12.2.2 Bootstrap Resampling

It is a process of creating many bootstrap samples S_i $(1 \leq i \leq M)$ using a method called Random Sampling with replacement from the training dataset. Each bootstrap sample contains 63.2% distinct tuples from the original training dataset and the remaining tuples are duplicate tuples drawn from the original dataset. If the training dataset T contains N tuples, M samples can be created, where each bootstrap sample S_i consists of N tuples by randomly choosing N tuples from the original training dataset with replacement.

12.2.3 Bagging

Bagging is otherwise called as Bootstrap Aggregation which combines both bootstrapping and aggregation (Breiman, 1996). It uses Bootstrap Resampling to create *M* bootstrap samples from the original training dataset *T*. It then trains *M* independent weak classifiers with each classifier containing a bootstrap sample S_i $(1 \le i \le M)$. Then, it combines the predictions made by *M* weak classifiers either by majority voting if the target variable is a categorical variable or by averaging if the target variable is a continuous variable. By combining or ensembling many classifiers predictions, the model reduces the error or decreases the variance, resulting in the prediction of a single weak classifier.

Algorithm 12.1: Bagging

Input: Training dataset *T* with *N* data instances.

Step 1: Create *M* bootstrap samples S_i $(1 \le i \le M)$ of *N* data instances each from the training dataset *T* using Random sampling with Replacement.

Step 2: Train *M* weak classifiers H_i $(1 \le i \le M)$ using the samples S_i $(1 \le i \le M)$.

Step 3: Combine predictions from all classifiers taking average $H_{ave} = \frac{1}{N}\sum_{i=1}^{M} H_i$.

12.2.4 Random Forest

This ensemble model is a variant of Bagging which uses decision trees for learning. *M* bootstrap samples, each of size *N* tuples, are created by random sampling with replacement from the training dataset of *N* tuples. These *M* samples are used to train *M* decision tree learners. Instead of training different models with the same features, Random Forest chooses a random set of features to construct decision trees. Moreover, the split feature chosen at every node for each decision tree is also different, thus exhibiting a level of differentiation among all trees in the ensemble model. The Random Forest ensemble model uses different hyperparameters that can be tuned prior to learning. Hence, all the decision trees have different leaves but are of the same depth. Then, it uses majority voting or averaging to finalize the prediction from all the decision trees. Thus, Random Forest is a collection of trees with each tree constructed with random samples, random set of features and random split features.

The model improves accuracy and reduces variance by employing more diversity among the different weak learners. Since the learning model is a decision tree with a chosen subset of features, the ensemble model also improves the efficiency with quick searching. Figure 12.1 illustrates the working principle of learning using the Random Forest algorithm.

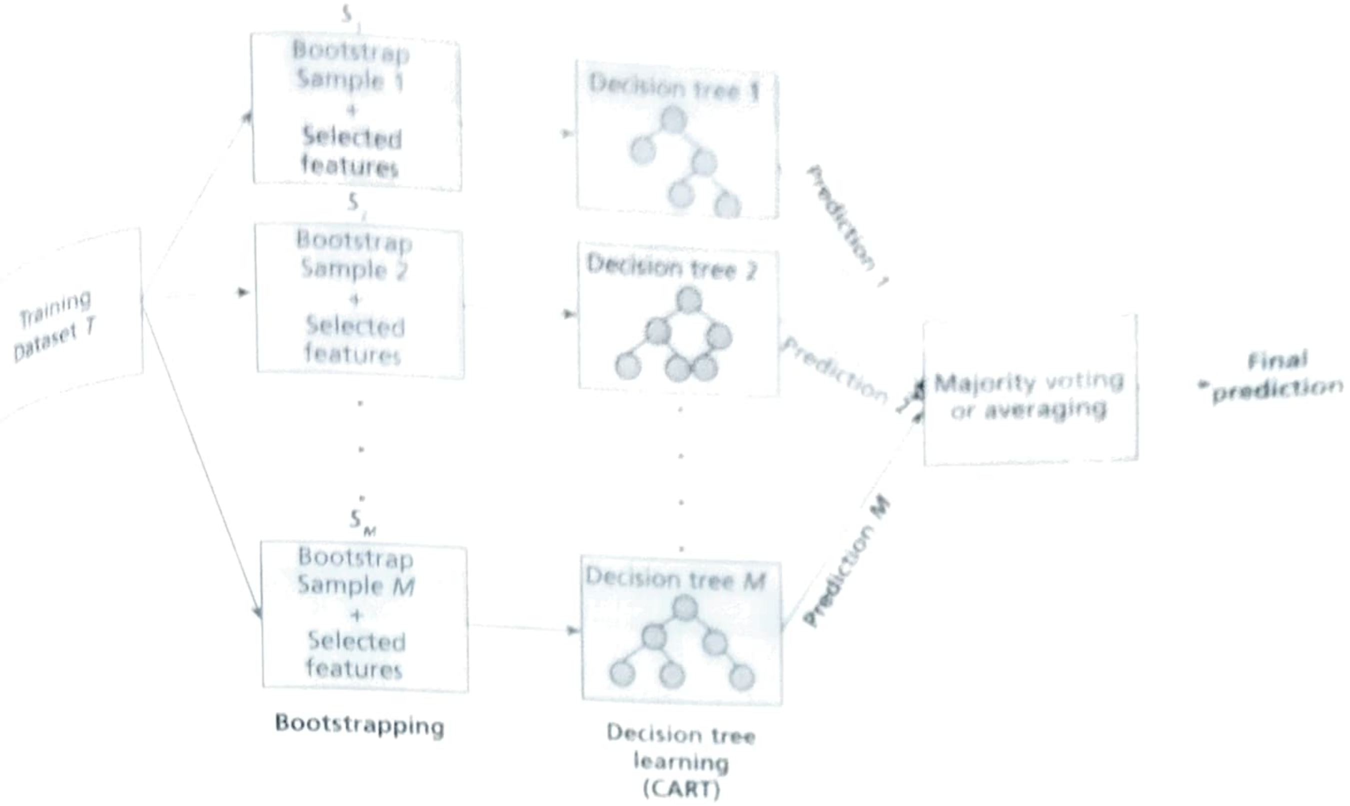
Training Dataset T
S_1
Bootstrap Sample 1 + Selected features
S_2
Bootstrap Sample 2 + Selected features
S_M
Bootstrap Sample M + Selected features
Bootstrapping
Decision tree 1
Decision tree 2
Decision tree M
Decision tree learning (CART)
Prediction 1
Prediction 2
Prediction M
Majority voting or averaging
Final prediction

Figure 12.1: Random Forest Model

Algorithm 12.2: Random Forest

```
Input: Training dataset $T$ with $N$ data instances.
Step 1: Create M bootstrap samples $S_i$ $(1 \leq i \leq M)$ of $N$ data instances each from the training dataset T using Random sampling with Replacement.
Step 2: Choose number of Decision trees to be constructed $DT_M$.
Step 3: Choose the number of features to be used in each tree. Normally, $\sqrt{D}$ randomly chosen features are taken.
Step 4: Choose the depth of the decision tree. All trees are to be same depth.
Step 5: Repeat $M$ times. For each tree:
    (a) Choose a bootstrap sample $S_i$ of $N$ data instances.
    (b) Compute the best split feature to be placed at each node.
    (c) Construct decision tree using CART. Do not prune the tree.
Step 6: Combine predictions from all classifiers taking Average $H_{ave} = \frac{1}{N}\sum_{i=1}^{M} H_i$ for classification problem or Majority voting in the case of regression problem.
```

Example 12.1: Consider a training dataset of 5 data instances shown in Table 12.1 which describes the performance of students and their skills obtained during study. Based on their performance and the skills obtained, they would get a Job Offer at the end of the course. Train using a Random Forest model and test this ensemble model.

Table 12.1: Training Dataset

S.No.	CGPA	Interactiveness	Communication Skills	Practical Knowledge	Job Offer
1.	≥9	Yes	Good	Good	Yes
2.	<9	No	Moderate	Good	Yes
3.	≥9	No	Moderate	Average	No
4.	<9	No	Moderate	Average	No
5.	≥9	Yes	Moderate	Good	Yes

Solution:

Step 1: Create 3 bootstrap samples of 5 data instances each from the training dataset T using random sampling with replacement as shown in Tables 12.2, 12.3 and 12.4, respectively.

Table 12.2: Sample S_1

S.No.	CGPA	Interactiveness	Communication Skills	Practical Knowledge	Job Offer
1.	≥9	Yes	Good	Good	Yes
2.	<9	No	Moderate	Good	Yes
3.	≥9	No	Moderate	Average	No
3.	≥9	No	Moderate	Average	No
5.	≥9	Yes	Moderate	Good	Yes

Table 12.3: Sample S_2

S.No.	CGPA	Interactiveness	Communication Skills	Practical Knowledge	Job Offer
2.	<9	No	Moderate	Good	Yes
3.	≥9	No	Moderate	Average	No
3.	≥9	No	Moderate	Average	No
5.	≥9	Yes	Moderate	Good	Yes
5.	≥9	Yes	Moderate	Good	Yes

Table 12.4: Sample S_3

S.No.	CGPA	Interactiveness	Communication Skills	Practical Knowledge	Job Offer
1.	≥9	Yes	Good	Good	Yes
1.	≥9	Yes	Good	Good	Yes
2.	<9	No	Moderate	Good	Yes
3.	≥9	No	Moderate	Average	No
3.	≥9	No	Moderate	Average	No

Step 2: Choose number of Decision trees to be constructed as 3.

Step 3: Choose the number of features to be used in each tree. Here, we will choose 2 features to construct every decision tree.

Step 4: Choose the depth of each decision tree to be 2.

Step 5: Repeat the following steps for 3 decision trees that are constructed:

Step 5 (a): Choose bootstrap sample S_1 of 5 data instances to construct Decision tree 1.

Step 5 (b): Compute the best split feature to be placed at each node. Here, we choose CGPA and Interactiveness.

Step 5 (c): Construct decision tree using CART shown in Figure 12.2.

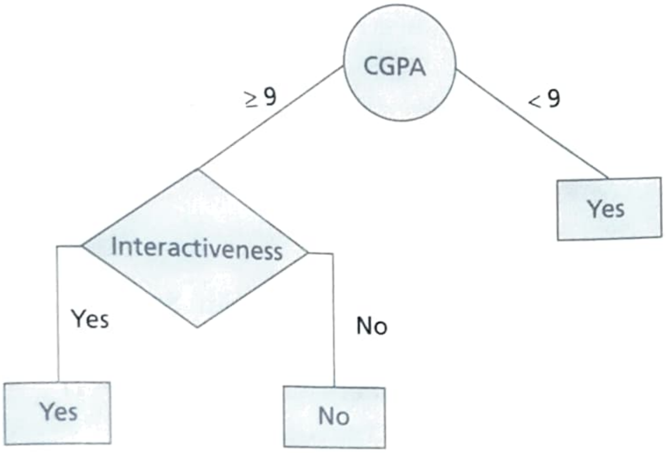

Figure 12.2: Decision Tree 1 with Sample S_1

Step 5 (a): Choose bootstrap sample S_2 of 5 data instances to construct Decision tree 2.

Step 5 (b): Compute the best split feature to be placed at each node. Here, we choose Interactiveness and Practical Knowledge.

Step 5 (c): Construct decision tree using CART shown in Figure 12.3.

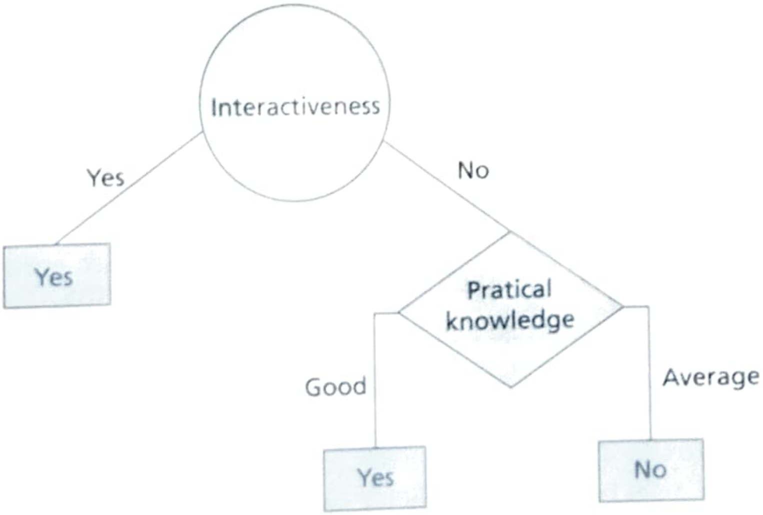

Figure 12.3: Decision Tree 2 with Sample S_2

Step 5 (a): Choose bootstrap sample S_3 of 5 data instances to construct Decision tree 3.

Step 5 (b): Compute the best split feature to be placed at each node. Here, we choose Communication Skills and Practical Knowledge.

Step 5 (c): Construct decision tree using CART shown in Figure 12.4.

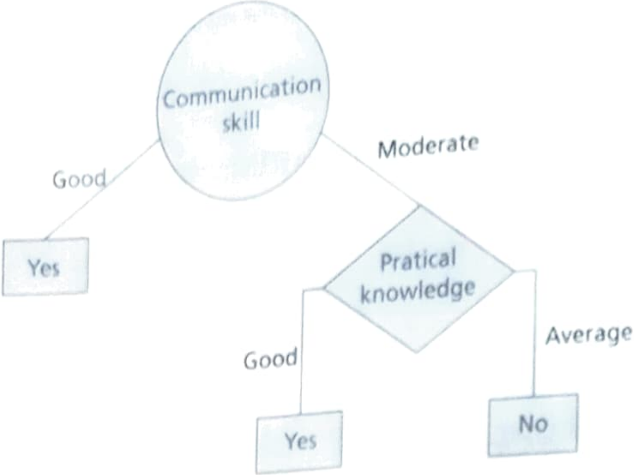

Figure 12.4: Decision Tree 3 with Sample S_3

Step 6: Combine predictions from all classifiers taking average $H_{avg} = \frac{1}{N}\sum_{i=1}^{M} H_i$ for Regression problem or majority voting in the case of classification problem.

Use the data instance 4 as the test instance is not used in the training. The data instance 4 is the Out of Bag instance used for testing.

Following are the predictions given by each of the decision tree constructed using the different samples:

Decision Tree 1 - Yes

Decision Tree 2 - No

Decision Tree 3 - No

It is a classification problem. So, using majority voting, the prediction Job Offer = No is the final prediction of this Random Forest model.

Advantages of the Random Forest Model

1. Can solve both classification problems and regression problems
2. Reduces overfitting and variance compared to a single decision tree
3. Provides more accuracy
4. Performs well even with smaller set of data

Disadvantages of the Random Forest Model

1. Increased complexity in construction and prediction
2. Uses more computational resources

12.3 INCREMENTAL ENSEMBLE MODELS

Incremental ensemble models use the predictions that are learnt from the previous learner as features to train a meta learner and make final predictions. Stacking and cascading models are popular incremental ensemble models which are discussed below.

12.3.1 Stacking

Stacking is a meta modelling technique introduced by Wolpert in the year 1992. Stacking model includes two types of learners called base-learners and a meta-learner. It is a process of ensembling multiple heterogeneous machine learning models or base learners to make predictions and then use these predictions as features to train the second level meta learner model whose prediction is considered as the final prediction. Base-learners use normal machine learning algorithms like Random Forests, SVM, Perceptron, etc., whereas the meta learner fits on the predictions of the base learner. The meta-learner can be meta-regressor or meta-classifier, which collects predictions from each base learner to estimate the final predictions, thereby improving the overall performance of the model. Stacking is making the second level of learner to estimate the biases and correct it. The stacking technique can be further extended to more than two levels which is called as multi-level stacking.

12.3.2 Cascading

A cascading ensemble model is a multistage learning technique with a sequence of classifiers ordered in terms of increasing complexity and specificity such that early classifiers are simple and general whereas later ones are more complex and specific. It moves to the next stage of learning only if the preceding learner's prediction is not confident. In cascaded models, different classifiers are coupled with their input/output function in a cascade, improving performance at each level. In a typical cascading ensemble model, the complexity of the model increases as more models are added to the cascade.

12.4 SEQUENTIAL ENSEMBLE MODELS

Boosting algorithms such as AdaBoost, Gradient Boosting and XGBoost are categorized as Sequential Ensemble models. The general principle of this model is to use multiple weak learners sequentially and the error exhibited by a weak learner is used to give importance to data instances which are further retrained by the next weak learner. Thus, the accuracy of prediction is improved in the ensemble model.

12.4.1 AdaBoost

AdaBoost, which is otherwise called as Adaptive Boosting, is a popular boosting algorithm and it is a non-linear classifier. It is a sequential ensemble model that trains multiple weak classifiers sequentially. It would train one weak classifier at a time, so that the errors made by poorly performing weak classifiers are reduced when they are trained sequentially.

AdaBoost is a greedy algorithm optimizing weights of data instances in the training dataset by adding a weak classifier at each step. Each weak classifier is trained with a random bootstrap sample drawn from the training dataset. Initially, weights are assigned to all data instances in the training dataset with equal probability. If there are 10 data instances in the training dataset, the probability of choosing a data instance to be trained by the first weak classifier would be 1/10. Hence, equal distribution of weights, that is, 1/10 is assigned to all data instances. After training by one weak classifier, the weights of all data instances are boosted based on the accuracy of classification done by it. The idea of boosting is to minimize the errors made by a weak classifier by increasing the weights of misclassified data instances and decreasing the weights of correctly classified data instances. The updated probabilities of data instances are reflected in the training dataset and these instances get trained with the next classifier.

Each weak classifier is also assigned a weight after training, based on the accuracy of its classification. If the classifier is more accurate, it is given more weight. A 50% accurate classifier is given a weight of 0. If the classifier has less than 50% accuracy, it gets a negative weight.

There are many weak classifiers such as Decision trees, Decision stumps, and Multilayer Perceptron. Linear classifiers such as Decision stumps are one-level decision trees which perform poorly with low accuracy. The depth of decision stumps is 1 and it has two leaves. The hypothesis of a Decision Stump model can be given as follows:

$$H_{Decision\text{-}Stump}(x) = \begin{cases} 1 & \text{if } x > \theta \\ 0 & \text{else} \end{cases} \tag{12.1}$$

The final classifier makes a linear combination of all the predictions made by the weak classifiers.

Figure 12.5 illustrates the method of learning using the AdaBoost algorithm.

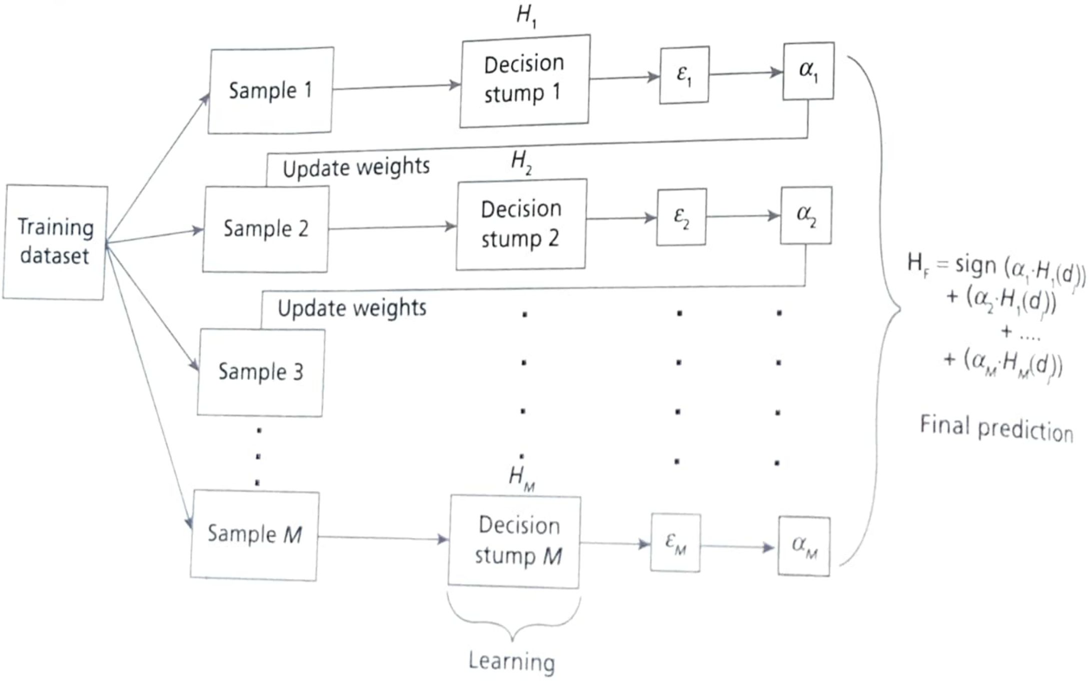

Figure 12.5: Working of AdaBoost Algorithm

Algorithm 12.3: AdaBoost

Input: Training dataset *T* with *N* data instances and *M* weak classifiers.

Step 1: Assign uniform weights 1/N to all the data instances d_j $(1 \leq j \leq N)$ in the training dataset *T*, i.e., $wt(d_1) = 1/N$, $wt(d_2) = 1/N$, ..., $wt(d_N) = 1/N$. Here, *wt* implies weight.

Step 2: Repeat for each weak classifier.

Step 2 (a): Train a weak classifier H_i with a random bootstrap sample from the training dataset *T*.

Step 2 (b): Compute the weighted error ε_i of H_i on current training dataset:

$$\varepsilon_i = \sum_{j=1}^{N} H_i(d_j)wt(d_j)$$

$H_i(d_j) = 0$ if prediction is correct with H_i

$H_i(d_j) = 1$ if prediction is wrong with H_i

Step 2 (c): Compute the weight of each weak classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

α_i is based on the error rate.

As the error rate increases, α_i grows exponentially negative.

Better classifiers get exponentially more weight.

If error rate is 50%, the classifier gets a weight of 0.

If error rate is more than 50%, the classifier gets a negative weight.

Step 2 (d): Calculate the normalizing factor Z_i. This is needed so that the sum of the weights of data instances adds up to 1.

Z_i = Total weight of correctly classified instances + Total weight of incorrectly classified instances and Here, *wt* implies weight.

$$Z_i = wt(\text{correctly classified instance}) \times \text{Number of correct classifications} \times e^{-\alpha_i}$$
$$+\, wt\left(\begin{matrix}\text{incorrectly}\\ \text{classified instance}\end{matrix}\right) \times \text{Number of incorrect classifications} \times e^{+\alpha_i}$$

Step 2 (e): Update the weight of all data instances:

$$wt(d_j)_{i+1}\ \text{correct-instance} = \frac{wt(d_j)_i\ \text{correct} - \text{instance} \times e^{-\alpha_i}}{Z_i}$$

$$wt(d_j)_{i+1}\ \text{incorrect-instance} = \frac{wt(d_j)_i\ \text{incorrect} - \text{instance} \times e^{+\alpha_i}}{Z_i}$$

where, $wt(d_j)_{i+1}$ correct-instance, is the updated weight of a correctly classified data instance to be trained by the next weak classifier.

$wt(d_j)_{i+1}$ incorrect-instance, is the updated weight of an incorrectly classified data instance to be trained by the next weak classifier.

(*Continued*)

Data instances that are wrongly classified will have their weights increased.

Data instances that are classified correctly will have their weights decreased.

Step 3: Compute the final predicted value for each data instance. It is the linear combination of weighted average predictions of all the classifiers multiplied by weight α_i computed for each classifier.

$$H_F(d_1) = \text{sign}\left(\sum_{i=1}^{M} \alpha_i \times H_i(d_j)\right)$$

Example 12.2: Consider a training dataset of six data instances as shown in Table 12.5.

Table 12.5: Training Dataset

CGPA	Interactiveness	Practical Knowledge	Communication Skills	Job Offer
≥9	Yes	Good	Good	Yes
<9	No	Good	Moderate	Yes
≥9	No	Average	Moderate	No
<9	No	Average	Good	No
≥9	Yes	Good	Moderate	Yes
≥9	Yes	Good	Moderate	Yes

Use 4 Decision Stumps for each of the 4 attributes.

The target attribute 'Job Offer' is a categorical attribute which predicts each data instance as 'Yes' or 'No'.

Number of data instances, N = 6.

Apply AdaBoost algorithm here.

Solution:

Step 1: Initial weight assigned to each item = 1/6.

Step 2: Iterate for each Weak classifier.

I. Decision Stump for CGPA

Step 2 (a): Train the Decision Stump H_{CGPA} with a random bootstrap sample from the training dataset *T*. Since there are only 6 data instances, use the full training dataset.

The first Decision stump classifies the instances based on the CGPA attribute as shown in Table 12.6. If CGPA ≥ 9, the data instance is predicted to have 'Job Offer' as 'Yes' else 'No'.

Table 12.6: Decision Stump Prediction using H_{CGPA}

CGPA	Predicted Job Offer	Actual Job Offer
≥9	Yes	Yes
<9	No	Yes
≥9	Yes	No
<9	No	No
≥9	Yes	Yes
≥9	Yes	Yes

Step 2 (b): Compute the weighted error ε_{CGPA} of H_{CGPA} on current training dataset T:

$$\varepsilon_i = \sum_{j=1}^{N} H_i(d_j)wt(d_j) \qquad H_i(d_j) = 0 \text{ if prediction is correct with } H_i$$

$$H_i(d_j) = 1 \text{ if prediction is wrong with } H_i$$

$$\varepsilon_{CGPA} = 2 \times 1/6 = 0.333$$

Step 2 (c): Compute the weight of each weak classifier:

$$\alpha_{CGPA} = \frac{1}{2}\ln\left(\frac{1-\varepsilon_{CGPA}}{\varepsilon_{CGPA}}\right)$$

$$\alpha_{CGPA} = \frac{1}{2}\ln\left(\frac{1-0.333}{0.333}\right)$$
$$= 0.347$$

Step 2 (d): Calculate the normalizing factor Z_{CGPA}.

$$Z_{CGPA} = wt(\text{correctly classified instance}) \times \text{Number of correct classifications} \times e^{-\alpha_{CGPA}}$$
$$+ wt\left(\begin{matrix}\text{incorrectly}\\ \text{classified instance}\end{matrix}\right) \times \text{Number of incorrect classifications} \times e^{+\alpha_{CGPA}}$$

$$Z_{CGPA} = 1/6 \times 4 \times e^{-0.347} + 1/6 \times 2 \times e^{0.347}$$
$$= 0.1178 \times 4 + 0.2358 \times 2$$
$$= 0.9428$$

Step 2 (e): Update the weight of all data instances:

$$wt(d_i)_{i+1} \text{ correct-instance} = \frac{wt(d_i)_{CGPA} \text{ correct} - \text{instance} \times e^{-\alpha_{CGPA}}}{Z_{CGPA}}$$

$$wt(d_i)_{i+1} \text{ incorrect-instance} = \frac{wt(d_i)_{CGPA} \text{ incorrect} - \text{instance} \times e^{\alpha_{CGPA}}}{Z_{CGPA}}$$

$$wt(d_i)_{i+1} \text{ correct-instance} = \frac{1/6 \times e^{-0.347}}{0.9428}$$
$$= \frac{0.1178}{0.9482} = 0.1249$$

$$wt(d_i)_{i+1} \text{ incorrect-instance} = \frac{1/6 \times e^{0.347}}{0.9428}$$
$$= \frac{0.2358}{0.9482} = 0.2501$$

Table 12.7 shows the modified weights for the data instances after learning with H_{CGPA}.

Table 12.7: Modified Weights with H_{CGPA}

CGPA	Predicted Job Offer	Actual Job Offer	Weights
≥9	Yes	Yes	0.1249
<9	No	Yes	0.2501
≥9	Yes	No	0.2501
<9	No	No	0.1249
≥9	Yes	Yes	0.1249
≥9	Yes	Yes	0.1249

II. Decision Stump for Interactiveness

Step 2 (a): Train the Decision Stump $H_{Interactiveness}$ with the sample obtained from the previous weak classifier H_{CGPA}.

The second Decision stump classifies the instances based on the Interactiveness attribute as shown in Table 12.8. If Interactiveness = 'Yes', the data instance is predicted to have 'Job Offer' as 'Yes', else if Interactiveness = 'No' the data instance is predicted to have 'Job Offer' as 'No'.

Table 12.8: Decision Stump Prediction using $H_{Interactiveness}$

Interactiveness	Predicted Job Offer	Actual Job Offer	Weights
Yes	Yes	Yes	0.1249
No	No	Yes	0.2501
No	No	No	0.2501
No	No	No	0.1249
Yes	Yes	Yes	0.1249
Yes	Yes	Yes	0.1249

Step 2 (b): Compute the weighted error $\varepsilon_{Interctiveness}$ of $H_{Interactiveness}$.

Only one instance is misclassified by this decision stump:

$$\varepsilon_{Interactiveness} = 1 \times 0.2501 = 0.2501$$

Step 2 (c): Compute the weight of each weak classifier:

$$\boxed{\alpha_{\text{Interactiveness}} = \frac{1}{2}\ln\left(\frac{1-\varepsilon_{Interactiveness}}{\varepsilon_{Interactiveness}}\right)}$$

$$\alpha_{Interactiveness} = \frac{1}{2}\ln\left(\frac{1-0.2501}{0.2501}\right)$$
$$= 0.5490$$

Step 2 (d): Calculate the normalizing factor $Z_{Interactiveness}$.

$$Z_{Interactiveness} = wt(\text{correctly classified instance}) \times \text{Number of correct classifications} \times e^{-\alpha_{Interactiveness}} + wt\left(\begin{array}{c}\text{incorrectly}\\ \text{classified instance}\end{array}\right) \times \text{Number of incorrect classifications} \times e^{+\alpha_{Interactiveness}}$$

$$
\begin{aligned}
Z_{Interactiveness} &= 0.1249 \times 4 \times e^{-0.5490} + 0.2501 \times 1 \times e^{0.5490} + 0.2501 \times 1 \times e^{-0.5490} \\
&= 0.2885 + 0.4331 + 0.1444 \\
&= 0.866
\end{aligned}
$$

Step 2 (e): Update the weight of all data instances:

$$wt(d_j)_{i+1} \text{ correct-instance} = \frac{wt(d_j)_{Interactiveness} \text{ correct} - \text{instance} \times e^{-\alpha_{Interactiveness}}}{Z_{Interactiveness}}$$

$$wt(d_j)_{i+1} \text{ incorrect-instance} = \frac{wt(d_j)_{Interactiveness} \text{ incorrect} - \text{instance} \times e^{\alpha_{Interactiveness}}}{Z_{Interactiveness}}$$

$$
\begin{aligned}
wt(d_j)_{i+1} \text{ correct-instance} &= \frac{0.1249 \times e^{-0.5490}}{0.866} \\
&= \frac{0.072}{0.866} = 0.0832
\end{aligned}
$$

$$
\begin{aligned}
wt(d_j)_{i+1} \text{ incorrect-instance} &= \frac{0.2501 \times e^{0.5490}}{0.866} \\
&= \frac{0.4331}{0.866} = 0.5001
\end{aligned}
$$

$$
\begin{aligned}
wt(d_j)_{i+1} \text{ correct-instance} &= \frac{0.2501 \times e^{-0.5490}}{0.866} \\
&= \frac{0.1444}{0.866} = 0.1667
\end{aligned}
$$

Table 12.9 shows the modified weights for the data instances after learning with $H_{Interactiveness}$.

Table 12.9: Modified Weights with $H_{Interactiveness}$

Interactiveness	Predicted Job Offer	Actual Job Offer	Updated Weights
Yes	Yes	Yes	0.0832
No	No	Yes	0.5001
No	No	No	0.1667
No	No	No	0.0832
Yes	Yes	Yes	0.0832
Yes	Yes	Yes	0.0832

III. **Decision Stump (Practical Knowledge)**

Step 2 (a): Train the Decision Stump $H_{Practical\ Knowledge}$ with the sample obtained from the previous weak classifier $H_{Interactiveness}$.

The third Decision stump classifies the instances based on the Practical Knowledge attribute as shown in Table 12.10. If Practical Knowledge = 'Good' the data instance is predicted to have 'Job Offer' as 'Yes', else if Practical Knowledge = 'Average' the data instance is predicted to have 'Job Offer' as 'No'.

Table 12.10: Decision Stump Prediction using $H_{Practical\ Knowledge}$

Practical Knowledge	Actual Job Offer	Predicted Job Offer	Weights
Good	Yes	Yes	0.0832
Good	Yes	Yes	0.5001
Average	No	No	0.1667
Average	No	No	0.0832
Good	Yes	Yes	0.0832
Good	Yes	Yes	0.0832

No instances are misclassified by this Decision Stump. So, don't change the weights of the data instances.

IV. Decision Stump (Communication Skills)

Step 2 (a): Train the Decision Stump $H_{Communication\ Skills}$ with the sample obtained from the previous weak classifier $H_{Interactiveness}$.

The fourth Decision stump classifies the instances based on the Communication Skills attribute as shown in Table 12.11. If Communication Skills = 'Good' the data instance is predicted to have 'Job Offer' as 'Yes', else if Communication Skills = 'Moderate' the data instance is predicted to have 'Job Offer' as 'No'.

Table 12.11: Decision Stump Prediction using $H_{Communication\ Skills}$

Communication Skills	Predicted Job Offer	Actual Job Offer	Weights
Good	Yes	Yes	0.0832
Moderate	No	Yes	0.5001
Moderate	No	No	0.1667
Good	Yes	No	0.0832
Moderate	No	Yes	0.0832
Moderate	No	Yes	0.0832

Step 2 (b): Compute the weighted error $\varepsilon_{Communication\ Skills}$ of $H_{Communication\ Skills}$.

two ~~Four~~ instances are misclassified by this decision stump:

$$\varepsilon_{Communication\ Skills} = 1 \times 0.5001 + 3 \times 0.0832 = 0.7497$$

Step 2 (c): Compute the weight of each weak classifier:

$$\alpha_{\text{Communication Skills}} = \frac{1}{2}\text{In}\left(\frac{1-\varepsilon_{Communication\ Skills}}{\varepsilon_{Communication\ Skills}}\right)$$

$$\alpha_{\text{Communication Skills}} = \frac{1}{2}\text{In}\left(\frac{1-0.7497}{0.7497}\right)$$

$$= -0.5485$$

Step 2 (d): Calculate the normalizing factor $Z_{\text{Communication Skills}}$.

$$Z_{Communication\ Skills} = wt(\text{correctly classified instance}) \times \text{Number of correct classifications} \times e^{-\alpha_{Communication\ Skills}}$$

$$+\, wt\begin{pmatrix}\text{incorrectly}\\ \text{classified instance}\end{pmatrix} \times \text{Number of incorrect classifications} \times e^{+\alpha_{Communication\ Skills}}$$

$$\text{mmunication Skills} = 0.0832 \times 1 \times e^{-(-0.5485)} + 0.1667 \times 1 \times \underline{e^{-(-0.5485)}} + 0.0832 \times 3 \times e^{+(-0.5485)} + \underline{0.5001} \times 1 \times e^{+(-0.5485)}$$

$$= 0.1440 + 0.2885 + 0.1442 + 0.2889$$

$$= 0.866$$

ep 2(e): Update the weight of all data instances:

$$wt(d_j)_{i+1} \text{ correct-instance} = \frac{wt(d_j)_{Communication\ Skills} \text{correct} - \text{instance} \times e^{-\alpha_{Communication\ Skills}}}{Z_{Communication\ Skills}}$$

$$wt(d_j)_{i+1} \text{ incorrect-instance} = \frac{wt(d_j)_{Communication\ Skills} \text{incorrect} - \text{instance} \times e^{+\alpha_{Communication\ Skills}}}{Z_{Communication\ Skills}}$$

$$wt(d_j)_{i+1} \text{ correct-instance} = \frac{0.0832 \times e^{-(-0.5485)}}{0.866}$$

$$= 0.1663$$

$$wt(d_j)_{i+1} \text{ correct-instance} = \frac{0.1667 \times e^{-(-0.5485)}}{0.866}$$

$$= 0.3331$$

$$wt(d_j)_{i+1} \text{ incorrect-instance} = \frac{0.5001 \times e^{+(-0.5485)}}{0.866}$$

$$= 0.3337$$

$$wt(d_j)_{i+1} \text{ incorrect-instance} = \frac{0.0832 \times e^{+(-0.5485)}}{0.866}$$

$$= 0.0555$$

Table 12.12 shows the modified weights for the data instances after learning with $H_{Communication\ Skills}$.

Table 12.12: Modified Weights with $H_{Communication\ Skills}$

Communication Skills	Predicted Job Offer	Actual Job Offer	Updated Weights
Good	Yes	Yes	0.1663
Moderate	No	Yes	0.3337
Moderate	No	No	0.3331
Good	Yes	No	0.0555 ~~0.3337~~
Moderate	No	Yes	0.0555
Moderate	No	Yes	0.0555

We can observe that the weights of misclassified instances are increased.

Step 3: Compute the final predicted value for each data instance:

$$H_F(d_j) = \text{sign}\left(\sum_{i=1}^{M} \alpha_i \times H_i(d_j)\right)$$

The sign function classifies instances as positive if greater than 0 and negative if less than 0. A value of 0 is considered neither positive nor negative.

The final prediction for data instance 1 is computed as shown below,

$$H_F(d_{j=1}) = \alpha_{CGPA} \times Yes + \alpha_{Interactiveness} \times Yes + \alpha_{Communication\ Skills} \times Yes$$

$$= 0.347 \times 1 + 0.5490 \times 1 + -0.5485 \times 1 = 0.3475$$

$$H_F(d_{j=1}) = Yes$$

This instance is classified with 'Job Offer = Yes' since the weighted average is positive.

Table 12.13 shows the final predicted value for all the data instances in the training dataset. It can be observed that the algorithm does not over-fit with the training dataset.

Table 12.13: Final Prediction

Instances	$\alpha_{CGPA} = 0.347$	$\alpha_{Interactiveness} = 0.5490$	$\alpha_{Communiction\ Skills} = -0.5485$	Weighted Average	Final Prediction
1.	Yes	Yes	Yes	0.3475	Yes
2.	No	No	No	0	No
3.	Yes	No	No	0.347	Yes
4.	No	No	Yes	–0.5485	No
5.	Yes	Yes	No	0.896	Yes
6.	Yes	Yes	No	0.896	Yes

Advantages of AdaBoost Algorithm

1. Fast
2. Simple to implement

Disadvantages of AdaBoost Algorithm

1. From empirical evidence and theoretical evidence, AdaBoost may overfit
2. Sensitive to noise data and outliers

Scan for information on *'Gradient Tree Boosting'* and *'XGBoost'*

Summary

1. Ensemble learning is a machine learning model which combines the predictions of several classifiers in order to improve the prediction accuracy of using a single classifier.
2. Bagging and Boosting are two popular models of ensemble learning.
3. A simple way of ensemble learning is majority voting or taking an average or weighted average of predictions of multiple machine learning models or base learners.
4. Ensembling machine learning is an approach to minimize bias and variance, and to build accurate models that avoid over-fitting and under-fitting of learning models.
5. Bagging is otherwise called as bootstrap aggregation, which combines both bootstrapping and aggregation.

The final prediction for data instance [illegible] is computed as shown below:

$H(d_{[illegible]}) = \alpha_{CGPA} \times \text{Yes} + \alpha_{[illegible]} \times \text{Yes} + \alpha_{[illegible]} \times \text{Yes}$

$= 0.347 \times 1 + 0.5490 \times 1 + (-0.5485) \times 1 = 0.3475$

$H(d_{[illegible]}) = \text{Yes}$

This instance is classified with Job Offer = Yes since the weighted average is positive.

Table 12.13 shows the final predicted value for all the data instances in the training dataset. [illegible] be observed that the algorithm does not over fit with the training dataset.

12.13: Final Prediction

nces	$\alpha_{CGPA} = 0.347$	$\alpha_{Interactiveness} = 0.5490$	$\alpha_{Communication\ Skills} = -0.5485$	Weighted Average	Final Prediction
	Yes	Yes	Yes	0.3475	Yes
	No	No	No	0	No
	Yes	No	No	0.347	Yes
	No	No	Yes	-0.5485	No
	Yes	Yes	No	0.896	Yes
	Yes	Yes	No	0.896	Yes

ges of AdaBoost Algorithm

- st
- mple to implement

tages of AdaBoost Algorithm

- m empirical evidence and theoretical evidence, AdaBoost may overfit
- sitive to noise data and outliers

mation on 'Gradient Tree Boosting' and 'XGBoost'

Summary

- le learning is a machine learning model which combines the predictions of several classifiers to improve the prediction accuracy of using a single classifier
- and Boosting are two popular models of ensemble learning
- way of ensemble learning is majority voting or taking an average or weighted average of s of multiple machine learning models or base learners

- The random forest ensemble model is [illegible] Bagging [illegible]. Boosting algorithms train multiple weak learners sequentially [illegible] learner is used to alter [illegible] learner [illegible]
- AdaBoost, called as adaptive boosting, is a popular boosting algorithm that trains multiple [illegible] classifiers sequentially [illegible] when they are trained sequentially
- AdaBoost is a greedy algorithm updating weights of data instances in the training dataset [illegible] adding a weak classifier [illegible]

Key Terms

- **Stacking** – A method of ensembling that [illegible] predictions of multiple [illegible] a new model and predict from the new model
- **Bagging** – A method of ensembling [illegible] independent [illegible] predictions obtained from different models
- **Boosting** – Employs different classifiers in a sequential fashion where the prediction [illegible] classifier is enhanced by boosting the weights for the data instances
- **Parallel Ensemble Model** – Enables the base learners to learn simultaneously
- **Sequential Ensemble Model** – Enables the base learners to learn sequentially
- **Voting** – A process of combining multiple machine learning [illegible] as base learners and taking average of their predictions [illegible] considering the maximum vote of their predictions [illegible]
- **Bootstrap Resampling** – A process of creating many bootstrap samples [illegible] called Random Sampling with replacement from the training dataset
- **Cascading Ensemble Model** – [illegible] different machine learning models and [illegible] prediction is not confident

Review Questions

1. What is meant by ensembling? [illegible]
2. What are the advantages and disadvantages of ensemble learning?
3. [illegible]
4. Compare bagging and stacking.
5. What is the principle [illegible] Random [illegible] model?
6. What is the general idea of boosting technique?
7. Compare AdaBoost and gradient tree boosting. [illegible]