# CHAPTER 8: Ensemble Boosting

## Introduction

In the previous chapters, we studied ensemble bagging techniques like the random forest, resampling techniques, and finally hyperparameter tuning, in this, we are going to look at boosting algorithms which are part of ensemble learning.

## What is Boosting?

Lots of analysts misinterpret the term ‘boosting’ used in machine learning. Let me provide an interesting explanation of this term. Boosting grants power to machine learning models to improve their accuracy of prediction. Boosting algorithms are one of the most widely used algorithms in machine learning competitions.

In this chapter, I will explain how boosting algorithm works in a very simple manner. Boosting is an ensemble approach (meaning it involves several trees) that starts from a weaker decision and keeps on building the models such that the final prediction is the weighted sum of all the weaker decision-makers. The weights are assigned based on the performance of an individual tree.

Diagram

Description automatically generated

Ensemble parameters are calculated in a **stagewise way** which means that while calculating the subsequent weight, the learning from the previous tree is considered as well. The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model.

**Weak Classifier – Why a tree?**

First, what is a weak classifier? **Weak classifier** - *slightly better* than random guessing.

Any algorithm could have been used as a base for the boosting technique, but the reason for choosing trees is:

Pro's

* computational scalability,
* handles missing values,
* robust to outliers,
* does not require feature scaling,
* can deal with irrelevant inputs,
* interpretable (if small),
* handles mixed predictors as well (quantitive and qualitative)

Con's

* inability to extract a linear combination of features
* high variance leading to a small computational power

And that’s where boosting comes into the picture. It minimizes the variance by taking into consideration the results from various trees.

In every machine learning model, the training objective is a sum of a loss function *L* and regularisation Ω:



## Ada Boost (Adaptive Boosting)

What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a low error is received.

Diagram, box and whisker chart

Description automatically generated

This diagram aptly explains Ada-boost. Let’s understand it closely.

Chart, calendar

Description automatically generated

*Box 1:* You can see that we have assigned equal weights to each data point and applied a decision stump to classify them as + (plus) or – (minus). The decision stump (D1) has generated a vertical line at the left side to classify the data points. We see that this vertical line has incorrectly predicted three + (plus) as – (minus). In such a case, we’ll assign higher weights to these three + (plus) and apply another decision stump. (Decision Stump is nothing but Decision trees with only one split.)

Icon

Description automatically generated with medium confidence

A picture containing text, clock

Description automatically generated

*Box 2:* Here, you can see that the size of three incorrectly predicted + (plus) is bigger as compared to the rest of the data points. In this case, the second decision stump (D2) will try to predict them correctly. Now, a vertical line (D2) at the right side of this box has classified three misclassified + (plus) correctly. But again, it has caused misclassification errors. This time with three -(minus). Again, we will assign a higher weight to three – (minus) and apply another decision stump.

A picture containing graphical user interface

Description automatically generated

*Box 3:* Here, three – (minus) are given higher weights. A decision stump (D3) is applied to predict these misclassified observations correctly. This time a horizontal line is generated to classify + (plus) and – (minus) based on the higher weight of misclassified observation.

A picture containing text, clock

Description automatically generated

*Box 4:* Here, we have combined D1, D2, and D3 to form a strong prediction having complex rules as compared to the individual weak learner. You can see that this algorithm has classified these observations quite well as compared to any individual weak learner.

## Gradient Boosting

Gradient boosting algorithm is one of the most powerful algorithms in the field of machine learning. As we know that the errors in machine learning algorithms are broadly classified into two categories i.e. Bias Error and Variance Error. As gradient boosting is one of the boosting algorithms it is used to minimize bias error of the model.

Unlike, the Adaboosting algorithm, the base estimator in the gradient boosting algorithm cannot be mentioned by us. The base estimator for the Gradient Boost algorithm is fixed and i.e. Decision Stump. Like, AdaBoost, we can tune the n\_estimator of the gradient boosting algorithm. However, if we do not mention the value of n\_estimator, the default value of n\_estimator for this algorithm is 100.

**Steps to build gradient boosting machine learning model.**

To simplify the understanding of the Gradient Boosting Machine, we have broken down the process into five simple steps.

**Step1**

The first step is to build a model and make predictions on the given data. Let’s go back to our data, for the first model the target will be the Income value given in the data. So, I have set the target as original values of Income.

Table

Description automatically generated

Now we will build the model using the features age and city with the target income. This trained model will be able to generate a set of predictions. Which are supposed as follows.

Table

Description automatically generated

Now I will store these predictions with my data. This is where I complete the first step.

A picture containing table

Description automatically generated

**Step2**

The next step is to use these predictions to get the error, which will be used further as a target. At the moment we have the Actual Income values and the predictions from model1. Using these columns, we will calculate the error by simply subtracting the actual income and the predictions of income. A shown below.

Table

Description automatically generated

As we mentioned previously the successive models focus on the error. So the errors here will be our new target. That covers up step two.

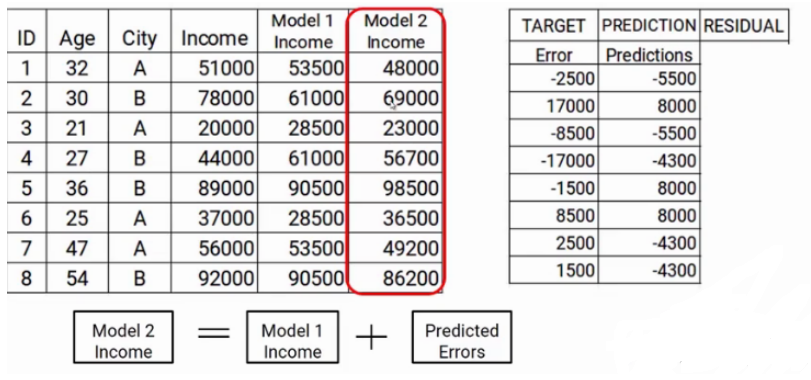
**Step3**

In the next step, we will build a model on these errors and make predictions. Here the idea is to determine, Is any hidden pattern in the error.

So using the error as target and the original features Age and City, we will generate new predictions. Note that the predictions, in this case, will be the error values, not the predicted income values, since our target is the error. Let’s say the model gives the following predictions

**Step4**

Now we must update the predictions of the model1. We will add the prediction from the above step and add that to the prediction from model1 and name it Model2 Income.



As you can see my new predictions are closer to my actual income values.

Finally, we will repeat steps 2 to 4, which means we will be calculating new errors and setting this new error as a target. We will repeat this process till the error becomes zero or we have reached the stopping criteria, which says the number of models we want to build. That’s the step-by-step process of building a gradient boosting model.

## XGBoost

Extreme Gradient Boosting or XGBoost is another popular boosting algorithm. In fact, XGBoost is simply an improvised version of the GBM algorithm! The working procedure of XGBoost is the same as GBM. The trees in XGBoost are built sequentially, trying to correct the errors of the previous trees.

But there are certain features that make XGBoost slightly better than GBM:

* One of the most important points is that XGBM implements parallel preprocessing (at the node level) which makes it faster than GBM
* XGBoost also includes a variety of regularization techniques that reduce overfitting and improve overall performance. You can select the regularization technique by setting the hyperparameters of the XGBoost algorithm
* Most existing tree-based algorithms can find the split points when the data points are of equal weights (using quantile sketch algorithm). However, they are not equipped to handle weighted data. XGBoost has a distributed weighted quantile sketch algorithm to effectively handle weighted data

Additionally, if you are using the XGBM algorithm, you don’t have to worry about imputing missing values in your dataset. **The XGBM model can handle the missing values on its own**. During the training process, the model learns whether missing values should be in the right or left node.

## Hands-on

**Problem Statement**

This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family Mushroom drawn from The Audubon Society Field Guide to North American Mushrooms (1981). Each species is identified as definitely edible, definitely poisonous. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets three, let it be'' for Poisonous Oak and Ivy.

Data:

**Step1**: Importing dependencies and data

First, we can load in the data as Pandas Data frame and look:

Graphical user interface, text, application

Description automatically generated

Let’s have a glance at the top 5 observations





Table

Description automatically generated

As seen from above, our dataset has 23 columns. Each column has a different feature of the mushrooms which are the column names of the dataset.

**Step2:** Exploratory Data Analysis

In the EDA step, I will visualize each feature by its class distribution. The "class" refers that the information of this mushroom is edible or poisonous.

**Mushroom Class**

A picture containing text

Description automatically generated

Chart, pie chart

Description automatically generated

As seen from the figure above, the red part represents the percentage of the poisonous mushrooms in the dataset. The green part represents the percentage of edible mushrooms. The percentages of these classes are pretty close to each other. From now on, I will represent the red color for the poisonous class and green for the edible class of the mushrooms.

**Cap Shapes**

Diagram

Description automatically generated

A picture containing text

Description automatically generated

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Cap Shapes. The x-axis represents the Types of the Cap Shapes of the Mushrooms and the y-axis represents the Number of the Mushrooms. According to this graph, the Convex Cap Shape is the most frequent. The Edible and Poisonous class mushrooms are pretty close to each other for the Convex and the Flat type. In contrast, for the following types which are Kobbed and Bell, the class types are pretty imbalanced distributed. The Sunken and the Conical types are completely edible.

**Cap Surfaces**

Diagram

Description automatically generated

Timeline

Description automatically generated with medium confidence

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Cap Surfaces. The x-axis represents the Types of Cap Surfaces of the Mushrooms and the y-axis represents the Number of the Mushrooms. As seen from the figure, the Scaly and Smooth types are high probability poisonous. In contrast, the Fibrous type is mostly edible.

**Cap Colours**

Text

Description automatically generated

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Cap Colors. The x-axis represents The Cap Colors of the Mushrooms and the y-axis represents the Number of the Mushrooms. According to this graph, the Edible and Poisonous class mushrooms are pretty close to each other for Brown, Gray, and Redcap colors. The White cap-colored mushrooms are highly edible. Purple and Green cape-colored mushrooms have no poisonous class.

**Bruises**

Text

Description automatically generated with low confidence

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Bruises. The x-axis represents that mushrooms by having Bruises and, the y-axis represents the Number of Mushrooms. As you can see from the graph above, Mushrooms without Bruises are high frequently poisonous. In contrast, the Mushrooms with the Bruises are highly edible.

**Odor**

Text

Description automatically generated with medium confidence

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Odors. The x-axis represents the Odor of the Mushrooms and, the y-axis represents the Number of Mushrooms. According to the graph above, mushrooms with no odor are highly safe for humans. But the mushrooms with the Foul, Fishy, and Spicy odor are high frequently poisonous. The mushrooms with the Anise and Almond odor have no poisonous class.

**Gill Attachments**

Shape, logo, company name, arrow

Description automatically generated

Text

Description automatically generated with medium confidence

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Gill Attachments. The x-axis represents the Types of the Gill Attachments and, the y-axis represents the Number of Mushrooms. The dataset has high frequently Free type gill attachments. According to this graph, the Free type gill attachments are approximately equally distributed. In contrast, the Attached type gill attachments are most frequently edible.

**Gill Spacing**

Diagram

Description automatically generated

Text

Description automatically generated

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Gill Spacing. The x-axis represents the Types of the Gill Spacing and, the y-axis represents the Number of Mushrooms. According to this figure, the Crowded type gill spacing is high frequently edible. In contrast, we cannot say the same thing for the Close-type gill spacing mushrooms. Their class distribution by gill spacing is pretty close to each other.

**Gill Color**

Text

Description automatically generated

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Gill Colors. The x-axis represents the Types of the Gill Colors and, the y-axis represents the Number of Mushrooms. As seen from the figure above, the Buff-colored gills are highly poisonous. In contrast, the White, Brown, Purple, Black, and Red gill colors are high frequently edible.

**Stalk Shape**

A picture containing text, clipart

Description automatically generated

Text

Description automatically generated

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Types of Stalk Shapes. The x-axis represents the Types of the Stalk Shapes and, the y-axis represents the Number of Mushrooms.

**Stalk Root**

Timeline

Description automatically generated with medium confidence

Chart, bar chart, waterfall chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Types of Stalk Roots. The x-axis represents the Types of Stalk Roots and, the y-axis represents that the Number of Mushrooms. As seen from the figure, most data points are Bulbous-type stalk root. The Equal, Club, and Rooted type stalk roots are highly edible but the Bulbous type stalk root is highly poisonous.

**Stalk Surface Above and Below Ring**

Text

Description automatically generated with medium confidence

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Types of Stalk Surfaces Above the Ring. The x-axis represents the Types of Stalk Surfaces Above the Ring and, the y-axis represents the Number of Mushrooms.

**Stalk Surface Below Ring**

A picture containing text

Description automatically generated

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Types of Stalk Surfaces Below the Ring. The x-axis represents the Types of Stalk Surfaces Below the Ring and, the y-axis represents that the Number of Mushrooms.

**Veil Types**

Diagram, engineering drawing

Description automatically generated

Text

Description automatically generated

Chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Veil Types. The x-axis represents the Veil Types and, the y-axis represents that the Number of Mushrooms. In this dataset, we have only Partial type Veils. As you can see, most of them are edible but the number of the poisonous class mushrooms are pretty close to the edible class of mushrooms.

**Veil Colors**

Timeline

Description automatically generated

Chart, waterfall chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Veil Colors. The x-axis represents the Veil Colors and, the y-axis represents the Number of Mushrooms. According to our dataset, most of the mushrooms have White colors. The class distribution of the White-colored Veils is approximately equal. The Brown, Orange, and Yellow Veil-colored mushrooms have no poisonous class.

**Number of Rings**

A picture containing timeline

Description automatically generated

Chart, treemap chart

Description automatically generated

**Ring Types**

A picture containing text, tool

Description automatically generated

Text

Description automatically generated with medium confidence

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Ring Types. The x-axis represents the Ring Types and, the y-axis represents the Number of Mushrooms. As seen from the graph, the mushrooms with the Pendant type are high frequently edible. In contrast, the mushrooms with the Large type have only the poisonous class.

**Spore Print Colours**

Text

Description automatically generated

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Spore Pring Colors. The x-axis represents the Spore Pring Colors and, the y-axis represents the Number of Mushrooms. As seen from the figure, the White and Chocolate spore print-colored mushrooms are high frequently poisonous. In contrast, the mushrooms with the Brown and Black spore print colors are high frequently edible.

**Populations**

A picture containing timeline

Description automatically generated

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Classes and Populations. The x-axis represents the Populations and, the y-axis represents the Number of Mushrooms. According to the graph above, the mushrooms populated as Several types are highly poisonous. In contrast, the mushrooms with Numerous, Abundant, and Clustered populations are pretty edible.

**Habitats**

Text

Description automatically generated with medium confidence

Chart, bar chart

Description automatically generated

The Figure above represents the Distribution of the Mushrooms by their Habitats and Populations. The x-axis represents the Habitats and, the y-axis represents the Number of Mushrooms.

**Step4: Data Pre-Processing**

In this step, I will check for the missing data points and then, encode the string data types. According to dataset exploration, all columns have string-type data.

A picture containing diagram

Description automatically generated

Text, table

Description automatically generated

There are no missing data points there were some data points for the "stalk-type" feature. But I filled the missing values with the most frequent one which is 'b'

**Step4: Encoding the data**

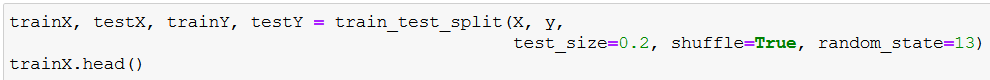
A picture containing text

Description automatically generated

A picture containing text, light, screenshot

Description automatically generated

**Step5**: **Train-Test Split**



A picture containing diagram

Description automatically generated

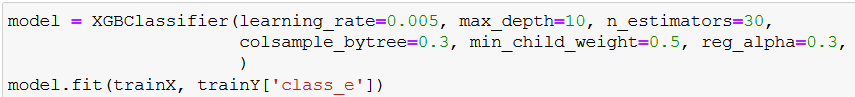
Text

Description automatically generated

Table

Description automatically generated with low confidence

**Step6: Model Building - XGBoost**



**Step7: Model Testing and Evaluation**

Graphical user interface, text, email

Description automatically generated

As seen from the results, the XGBClassifier performed with 100% accuracy.

**Step8: Model Building – Ada Boost**

Graphical user interface, text, application

Description automatically generated

**Step9: Model Testing and Evaluation**

Text

Description automatically generated

As seen from the results, even AdaBoost performed with 100% accuracy.

**Step10: Feature Importance**

Text

Description automatically generated

Table

Description automatically generated

**SUMMARY**

In this chapter, we studied what is a ensemble boosting, how the boosting algorithm works, and different algorithms like Ada Boosting, Gradient Boosting, and XGBoosting. We have also seen the implementation of XGBoosting the algorithm.

## Assessment

**Choose the appropriate option**

1. **Which of the following algorithm doesn’t use the learning rate as one of its hyperparameters?**
   1. Gradient Boosting
   2. Extra Trees
   3. AdaBoost
   4. Random Forest
2. **Which of the following algorithm are not an example of an ensemble learning algorithm?**
   1. Random Forest
   2. Decision Tree
   3. Gradient Boosting
   4. XGBoost
   5. All of the above
3. **Which of the following is the main reason for aggregating weak learners in boosting?**
4. To prevent overfitting.
5. To prevent underfitting
6. None of the above
7. **Which of the following are unique features of XGBoost?**
8. Regularization
9. Handling Missing Data
10. Parallel Learning using multiple cores
11. All of the above
12. **AdaBoost is decision trees with one level which means with Decision trees with only 1 split. These trees are called as \_\_\_\_\_\_\_\_\_\_\_\_\_**
    1. Decision Trees
    2. Extra Trees
    3. Decision Stumps
    4. None of these

**Fill in the spaces with appropriate answers**

1. Ada Boosting assigns higher weights to points that are \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. XGBoost has an in-built capability to \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. Gradient Boosting uses \_\_\_\_\_\_\_\_\_\_\_\_ loss function for regression and \_\_\_\_\_\_\_\_\_\_\_ for classification
4. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ conditions can be set to alleviate overfitting and also reduce runtime in Gradient Boosting.
5. XGBoost uses \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ parameter of decision tree to prune the tree which reduces runtime significantly.

**True or False**

1. Boosting is the method for improving the performance by aggregating the results of weak learners sequentially.
   1. True
   2. False
2. Ensembles will yield bad results when there is significant diversity among the models.
   1. True
   2. False
3. If you use an ensemble of different base models, it is necessary to tune the hyperparameters of all base models to improve the ensemble performance.
   1. True
   2. False
4. In contrast to bagging techniques like Random Forest, in which trees are grown to their maximum extent, boosting makes use of trees with fewer splits.
   1. True
   2. False
5. In boosting, each tree learns from its predecessors and updates the residual errors.
   1. True
   2. False

## Programming Assessment

Using the data in the below URL, Perform the following tasks

https://github.com/fenago/MLBook/blob/main/Chapter%208%20-%20Ensemble%20Boosting/Code/Dataset/pima-diabetes.csv

1. Import the data
2. Perform Data Cleaning
3. Perform EDA
4. Fit Gradient Boosting, Ada Boosting, and XG Boosting models
5. Evaluate the models.

## Assessment Solutions

**Choose the appropriate options**

1. D
2. B
3. A
4. D
5. C

**Fill in the spaces with appropriate answers**

1. Wrongly Classified
2. Imbalanced Class and Missing Values
3. Mean Average Error and Log Loss
4. Early Stopping
5. Maximum depth

**True or False**

1. True
2. False
3. False
4. True
5. True