Machine Learning

Lesson 6

Ensemble Models

Lecturer: Dr. Msagha J Mbogholi, PhD

Flashback from Lesson 5

- The clustering method helps grouping valuable data into clusters and picks appropriate results based on different techniques.

- Clustering is used when data is not labelled; it is an unsupervised machine learning technique.

- Clustering methods can be largely grouped as being partition based, hierarchical based, density based, grid based or model based

.

- GMM can be used to find clusters in datasets where the clusters may not be clearly defined.

- Gaussian mixture models take into account the variance, mean and weights of the data

Content

- Ensemble Models
- Bagging
- Boosting

Part 1

Ensemble Models

1.1 Introduction

- In lesson 4 and 5 the concept of classifiers and clustering techniques were introduced.
- Traditionally, Applications using Machine Learning were built on a single learner, like a Naïve Bayes, Decision Tree, Support Vector Machine, or an Artificial Neural Network, training it and feeding it with data to perform a certain task through this data.
- Combinations of models are generally known as model ensembles. They are among the most powerful techniques in machine learning, often outperforming other methods . While the performance of an ensemble model outperforms a single algorithm in the majority of cases, the degree of model complexity and sophistication can pose as a potential drawback. (Theobald, 2021)
- Ensemble methods are general techniques in machine learning for combining several predictors to create a more accurate one. (Mohri et al., 2012)
- To make better predictions ensemble models are built to achieve better performance than any single model. An ensemble reduces the spread or dispersion of the predictions and model performance.
- Ensemble models combines multiple other models in the prediction process. Those models are referred to as base estimators. It is a solution to overcome the following technical challenges of building a single estimator:
  - High variance
  - Low accuracy
  - Features noise and bias

1.1 Introduction (cont'd)

- Ensemble models are generated using one of two techniques: using a single technique variantly which is known as homogeneous ensemble, or using different techniques which is also called heterogeneous ensemble. (Theobald, 2021)
- Examples of homogeneous ensemble include: bagging and boosting
- Example of heterogeneous ensemble include: bucket of models and stacking.
- Bucket of models training uses many different models using the same training data and then picking the one that performed best on the test data.
- Stacking involves running many models in parallel and then combining the results to produce an optimal model.
- Bagging and boosting are described in this lesson.

1.1 Introduction (cont'd)

- An ensemble design includes two main steps: model training and model combination. During the training process, each model is trained with the same training examples, but using different subsets of the input factors. Fig. 1 shows common ensemble architecture.

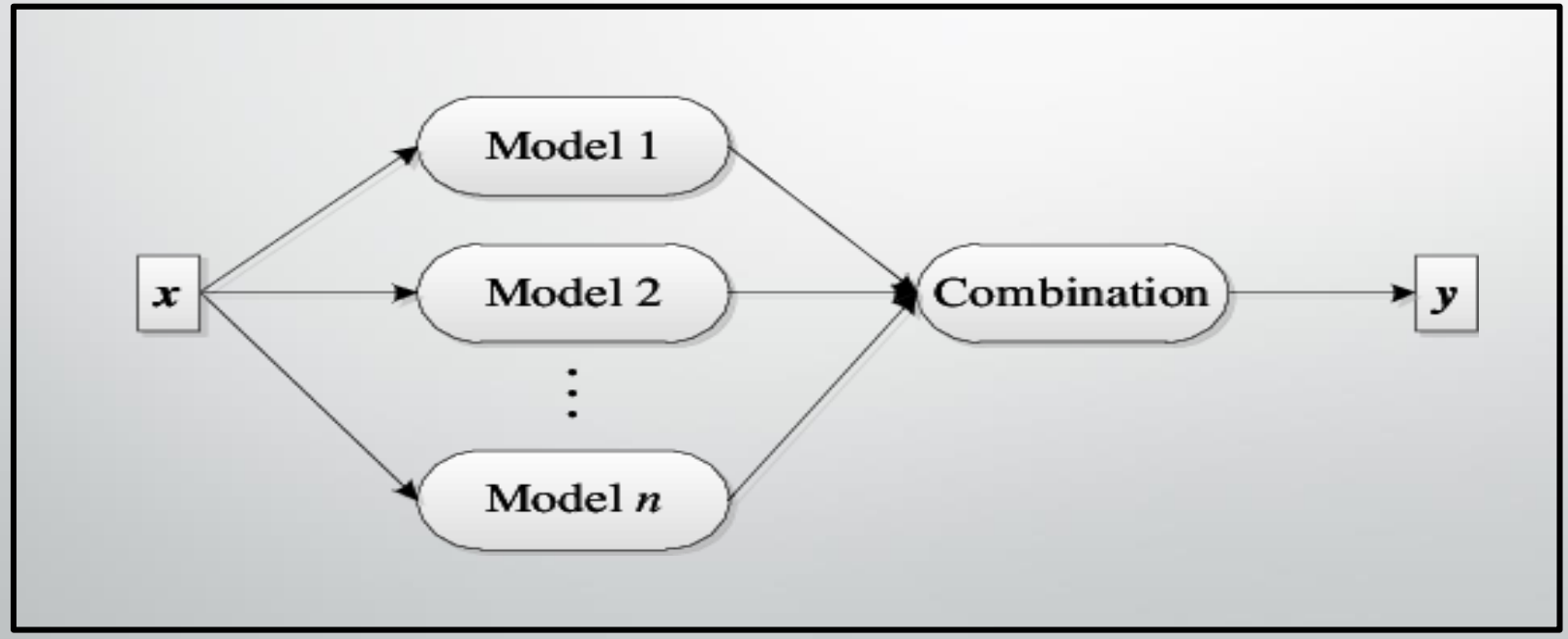


Fig 1. Petrakova et al. (2015)

1.2. Ensemble Techniques

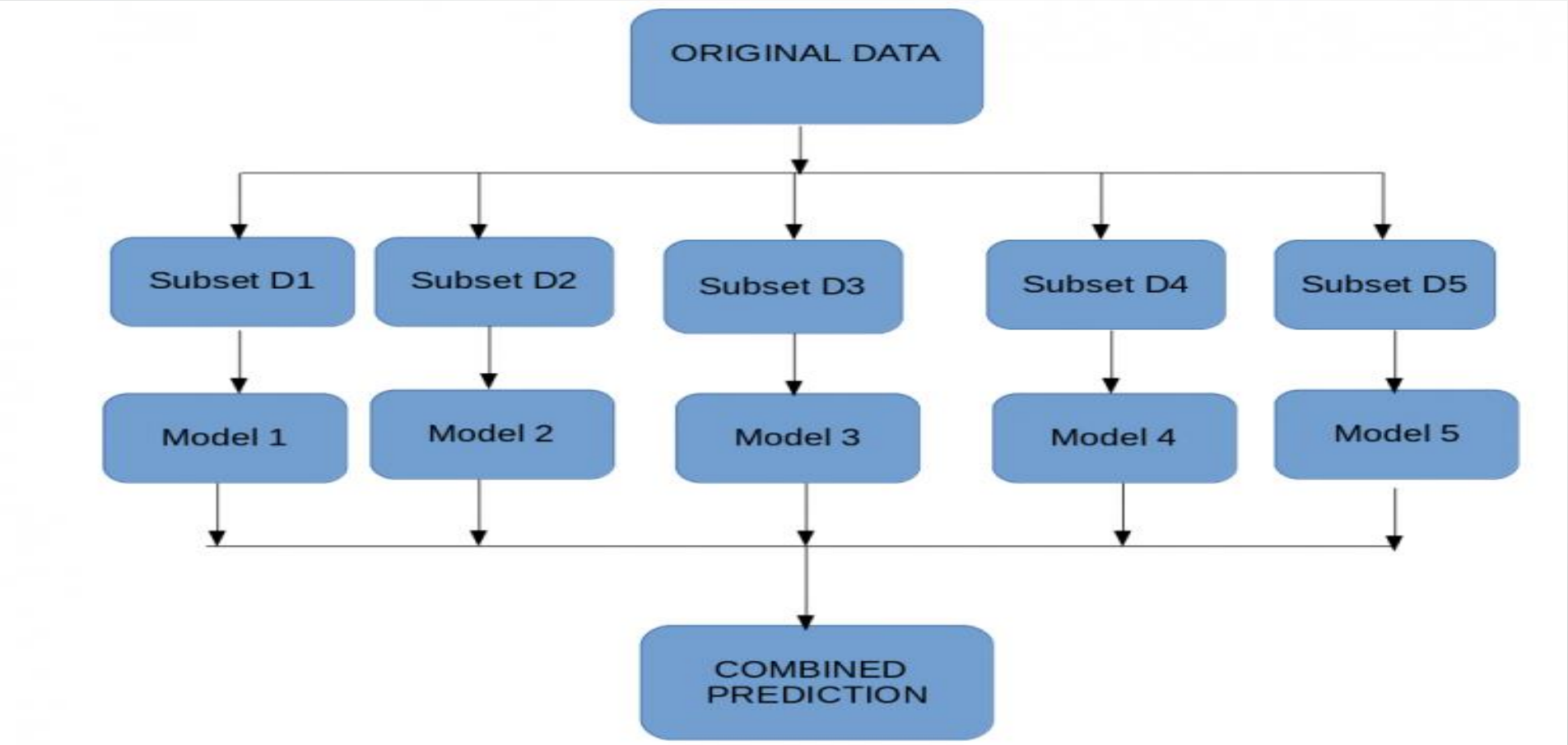
- The output of ensemble models can be implemented by using simple ensemble methods or advanced ensemble methods.
- Simple Ensemble Methods
- The output of each model can be determined by using one of the following statistical approaches
  - Mode: the final prediction is the one made by most models taken as votes
  - average./Mean : the average prediction made by all models is taken
  - Weighted Average: . different weights are assigned to all the models in order to make a prediction, where the assigned weight defines the relevance of each model.
- Advanced Ensemble Methods:
  - They are used to reduce the model error and maintaining its generalization. The way to implement such aggregation can be achieved using some techniques such as Bagging, Boosting ,Stacking etc.

1.2 Ensemble Techniques(Cont'd)

- **Bagging(Bootstrap aggregating)** : multiple models of same learning algorithm trained with subsets of datasets(bootstraps) randomly picked from the training datasets.
- A combination of these multiple models, reduces variance and noise, as the average prediction generated from different models is much more reliable and robust than a single model as shown in Figure 2
- In fig 2 the final prediction will be determined by combining the predictions of all the models.

.

1.2 Ensemble techniques (cont'd)

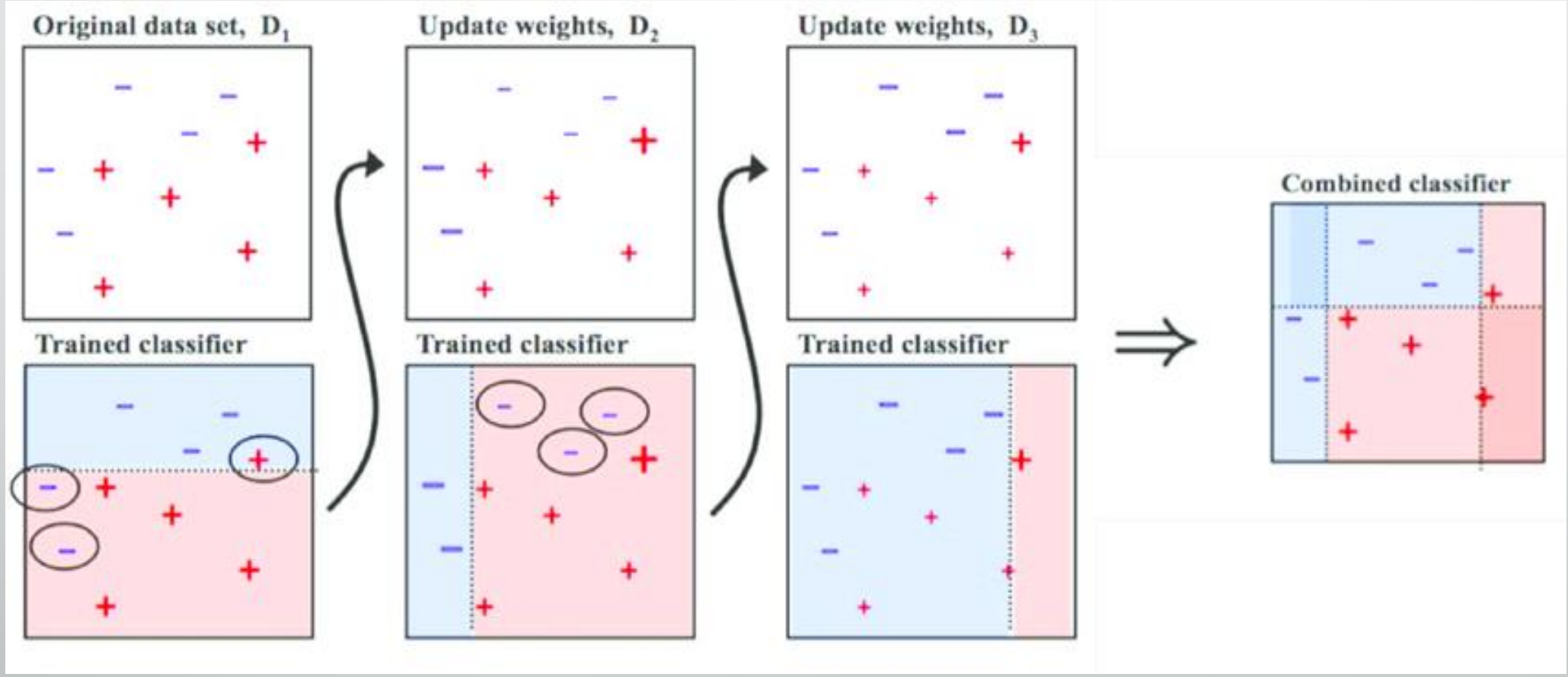


- Fig 2 Bagging example (analyticsvidhya.com)

1.2 Ensemble Techniques(Cont'd)

- **Boosting** this techniques attempts to build a strong classifier from the number of weak classifiers. Each model built tries to correct the errors present in the previous model.
- This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added. This is shown in Figure 2
- Boosting is a homogeneous ensemble and its main strength is to address errors and misclassified data; by so doing it produces a sequential model. Examples of algorithms that use this technique are gradient boost, Ada boost, Catboost and XGBM

1.2 Ensemble techniques (cont'd)



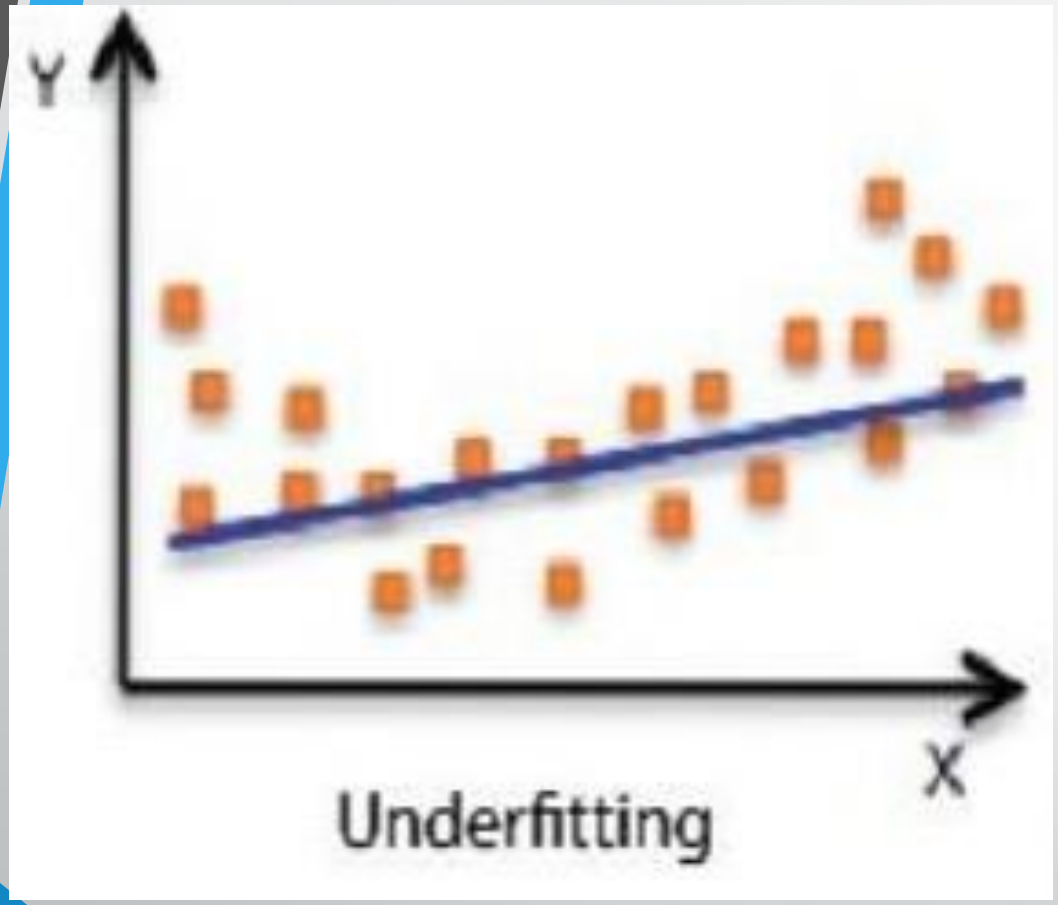
- Fig 2 Boosting example (microsoftbing.com)

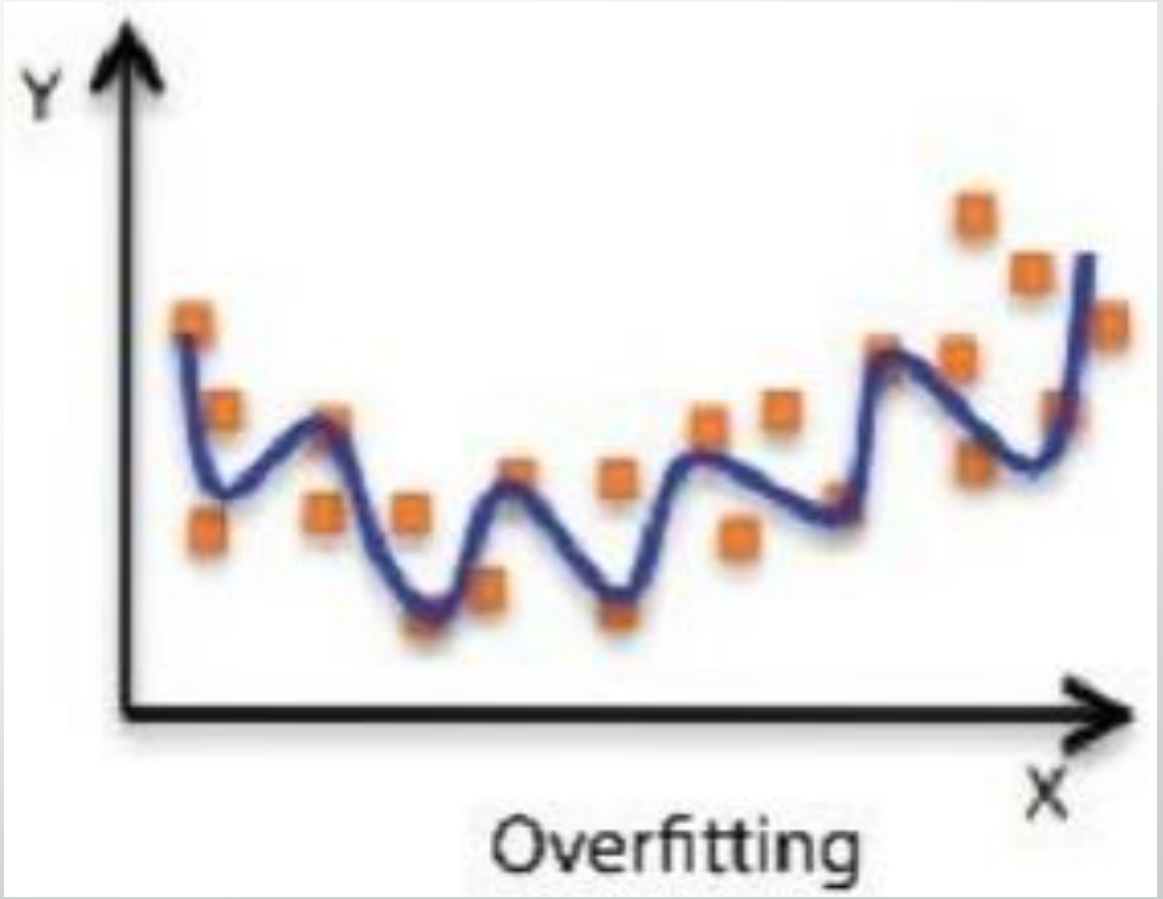
1.3 Prediction Errors

- Prediction errors are defined as the collection of bias, variance, and irreducible errors.
- High variance occurs when the model overfits the training data such that the training data error is zero. But the test error varies greatly based on the selection of the training dataset.
- Bias is a measurement of how accurately a model can capture a pattern in a training dataset. High Bias mostly occurs when a model underfits the training dataset and the train error is high.
- The two parameters that are of interest are bias and variance. The goal is to understand the two errors and how ensemble methods affect them.

1.4.1 Bias Error

- **Bias Error:** High bias refers to when a model shows high inclination towards an outcome of a problem it seeks to solve. It is highly biased towards the given problem. This leads to a difference between estimated and actual results. When the bias is high, the model is most likely not learning enough from the training data as shown in Figure 3 (a).
- It does not learn the key features therefore the resulting predictions are unreliable, and the generalization is poor. This is what is known as underfitting. (Ayuya, 2021)
- In this case, the model does only shallow learning and fails to capture important aspects of the training data.
- When applied to new data, the underfit model will also do poorly. But it will do poorly even on the training data, despite being exposed to it.

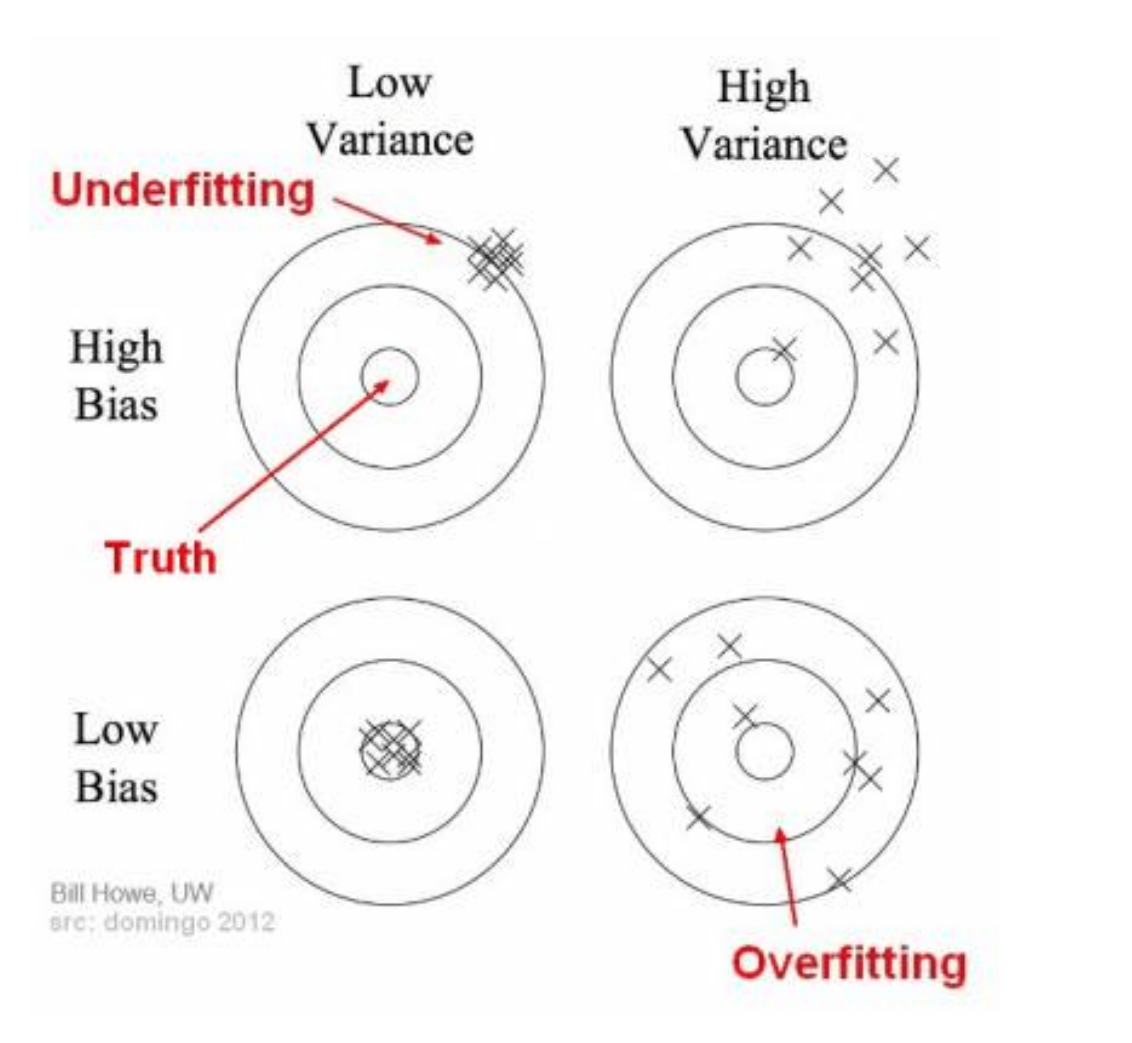




- Fig 3 (a) Underfitting and (b) Overfitting (Mbogho, 2019)

1.4.2 Variance Error

- Variance error: The sensitivity of models to slight fluctuations in the training data describes the variance. When a function fits a bit too close to a given number of data points, we say that the model is overfitting. High variance is an indication of overfitting. (Ayuya, 2021). Fig 4 examines the relationship between variance, bias, and fitting.
- An overfit model is one that has learnt the training data so well that it underperforms when applied to unseen data. The blue line in Figure 3(b) represents a model that has learnt the training data too well (i.e. has overfit the training data), and is therefore probably going to make wrong predictions for new data
- To deal with this, we can get more training data if the data is inadequate. We could also use a less complex model. We shall, however, look at how to reduce variance using bagging(in part 2 of this lesson).



- Fig 4 Relationship between variance, bias and fitting (Ayuya, 2021)

1.5 Bias variance trade-off

- It is desirable to achieve a low bias and variance to ensure accurate predictions. High bias and high variance hint at lower performance. This occurs dur to underfitting and overfitting
- There is a need to have an balanced fit model in order to have low variance and low bias
- This can be achieved by using the following methods:
  - Ensemble techniques such as Bagging and Boosting(discussed in part 2 of this lesson)
  - Cross validation such as k fold cross validation
  - Dimensionality Reduction such as principal component analysis(discussed in Lesson 10)

Part 2

Bagging

2.1 Introduction

- Bagging, short for 'bootstrap aggregating', is a simple but highly effective ensemble method that creates diverse models on different random samples of the original dataset(Flach, 2012).
- These samples are taken uniformly with replacement and are known as bootstrap samples. Because samples are taken with replacement the bootstrap sample will in general contain duplicates, and hence some of the original data points will be missing even if the bootstrap sample is of the same size as the original data set(Flach,2012).
- The differences between the bootstrap samples will create diversity among the models in the ensemble(Flach, 2012).
- The idea of bagging is based on making the training data available to an iterative process of learning. Each model learns the error produced by the previous model using a slightly different subset of the training dataset. Bagging reduces variance and minimizes overfitting.
- Bagging minimizes the overfitting of data, It improves the model's accuracy and deals with higher dimensional data efficiently.

1.2 Bootstrapping

- **Bootstrapping:** Bagging is based on a bootstrapping sampling technique. Bootstrapping creates multiple sets of the original training data with replacement. Replacement enables the duplication of sample instances in a set. Each subset has the same equal size and can be used to train models in parallel.

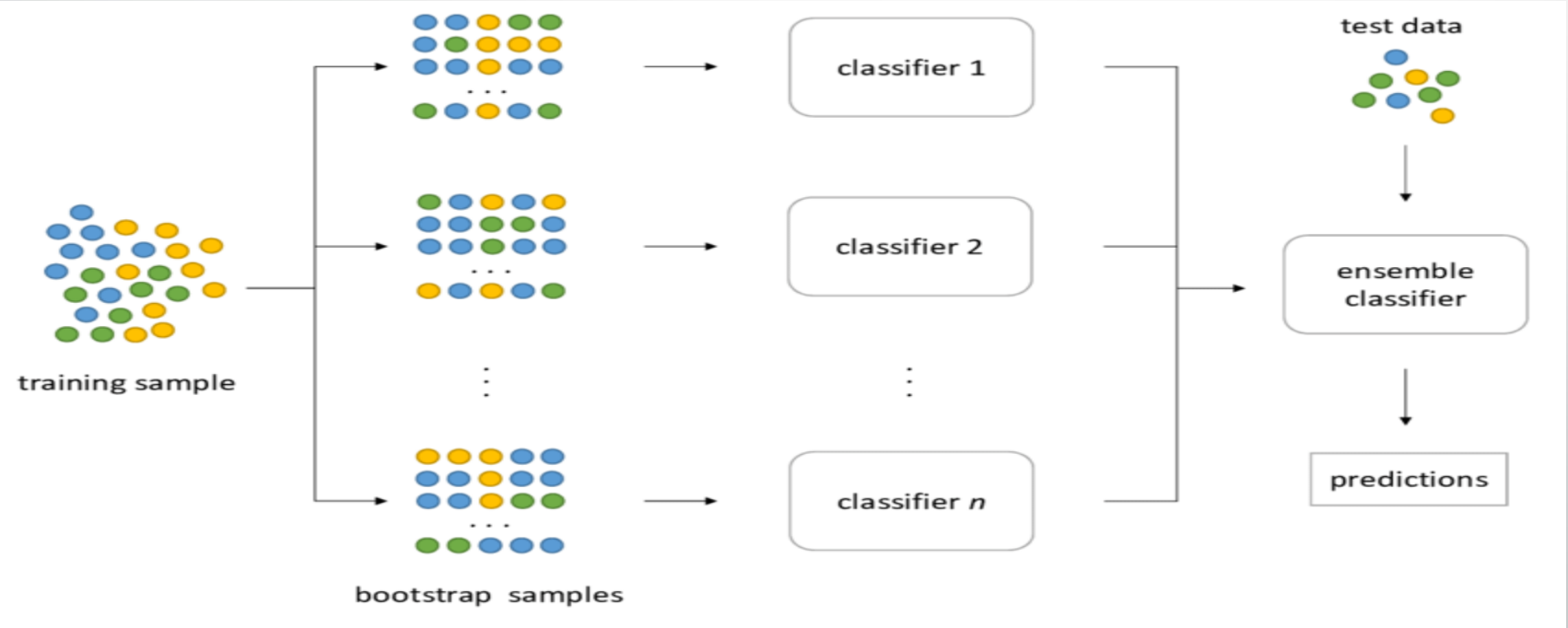


Fig 5. Bootstrapping (Mbaskills.in)

1.3 Steps to Perform Bagging

1. Consider there are n observations and m features in the training set. You need to select a random sample from the training dataset without replacement
2. A subset of m features is chosen randomly to create a model using sample observations
3. The feature offering the best split out of the lot is used to split the nodes
4. The tree is grown, so you have the best root nodes
5. The above steps are repeated n times. It aggregates the output of individual decision trees to give the best prediction

1.4 Bagging Algorithm

- Algorithm 1 gives the basic bagging algorithm, which returns the ensemble as a set of models. We can choose to combine the predictions from the different models by voting the class predicted by the majority of models wins or by averaging, which is more appropriate if the base classifiers output scores or probabilities.

Algorithm 1: Bagging (D,T,A) – train an ensemble of models from bootstrap samples.
Input: data set D; ensemble size T ; learning algorithm A .
Output: ensemble of models whose predictions are to be combined by voting
or averaging.
1 for t = 1 to T do
2 build a bootstrap sample Dt from D by sampling |D| data points with
replacement;
3 run A on Dt to produce a model Mt;
4 end
5 return {Mt |1 ≤ t ≤ T }

1.4 Bagging with Random Forest

- Bagging is particularly useful in combination with tree models, which are quite sensitive to variations in the training data. When applied to tree models, bagging is often combined with another idea: to build each tree from a different random subset of the features, a process also referred to as subspace sampling. This encourages the diversity in the ensemble even more, and has the additional advantage that the training time of each tree is reduced. The resulting ensemble method is called random forests, and the algorithm is given in Algorithm 2.

Algorithm 2: RandomForest(D,T,d) – train an ensemble of tree models from bootstrap samples and random subspaces.
Input : data set D; ensemble size T ; subspace dimension d.
Output : ensemble of tree models whose predictions are to be combined by voting or averaging.
1 **for** t = 1 **to** T do
2 build a bootstrap sample Dt from D by sampling |D| data points with replacement;
3 select d features at random and reduce dimensionality of Dt accordingly;
4 train a tree model Mt on Dt without pruning;
5 **end**
6 **return** {Mt |1 ≤ t ≤ T }

1.5 An example of Bagging

- N = {18,20,24,30,34,95,62,21,14,58,26,19} — Original sample with 12 elements
- Bootstrap sample A: {20, 34, 58, 24, 95,18}
- Bootstrap sample B: {62, 21, 19, 30, 14,26}
- Bootstrap sample C: {58, 24, 18, 24, 34,20}
- Once bootstrap samples are created, a model classifier is used for training or building a model and then selecting model based on popularity votes. In the classification model, a label with maximum votes will be assigned to the observations. The average value is used in case of a regression model.

1.6 Bagging Demonstration in Python Using IRIS Dataset

Figures 6,7 and 8 shows sample codes for Bagging using Random Forest

```
#Import libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

Load the dataset

```
dataset_train = pd.read_csv("Google_Stock_Price_Train.csv")
dataset_train.head()
```

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800

Split the dataset into training and testing

Fig 6. Bagging (Source : Simplelearn.com)

1.5 Bagging Demonstration in Python Using IRIS Dataset(Cont'd)

Split the dataset into training and testing

```
#Split data in training and testing set
X_fit, X_eval, y_fit, y_test= model_selection.train_test_split( X, Y, test_size=0.30, random_state=1 )
```

Creating sub samples to train models

```
#Create random sub samples to train multiple models
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
```

Define a decision tree

```
#Define a decision tree classifier
cart = DecisionTreeClassifier()
num_trees = 100
```

Classification model for bagging

```
#Create classification model for bagging
model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
```


Fig 7. Bagging (Source : Simplelearn.com)

2.3 Bagging Demonstration in Python Using IRIS Dataset(Cont'd)

Classification model for bagging

```
#Create classification model for bagging
model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
```

Train models and print their accuracy

```
#Train different models and print their accuracy
results = model_selection.cross_val_score(model, X_fit, y_fit,cv=kfold)
for i in range(len(results)):
    print("Model: "+str(i)+" Accuracy is: "+str(results[i]))
```

```
Model: 0 Accuracy is: 1.0
Model: 1 Accuracy is: 1.0
Model: 2 Accuracy is: 1.0
Model: 3 Accuracy is: 0.9090909090909091
Model: 4 Accuracy is: 1.0
Model: 5 Accuracy is: 1.0
Model: 6 Accuracy is: 0.9
Model: 7 Accuracy is: 1.0
Model: 8 Accuracy is: 1.0
Model: 9 Accuracy is: 0.7
```


Fig 8. Bagging (Source : Simplelearn.com)

Part 3

Boosting

3.1 Introduction

- Boosting, initially named ***Hypothesis Boosting***, consists on the idea of filtering or weighting the data that is used to train our team of weak learners, so that each new learner gives more weight or is only trained with observations that have been poorly classified by the previous learners.
- General method for improving the accuracy of any given learning algorithm. When a weak learner can be implemented efficiently, boosting provides a tool for aggregating such weak hypotheses to approximate gradually good predictors for larger, and harder to learn, classes(David, 2014).
- The boosting paradigm allows the learner to have smooth control over the Bias/variance tradeoff(David,2014).
- Works by creating a series of challenge datasets. even modest performance on these can be used to produce an overall high-accuracy predictor
- In practice there are actually many types of algorithms that are used for boosting, including:XGBoost, AdaBoost and Gradient Boost.

3.2 Bagging vs Boosting.

- In bagging the weak learners are trained in parallel using randomness, while in boosting the learners are trained sequentially, in order to be able to perform the task of data weighting/filtering
- In boosting the models can have different importance or weights (represented in the different sizes of the learners), while in bagging all learners have the same weight in the final decision.
- Also, in boosting, the data set is weighted (represented by the different sizes of the data points), so that observations that were incorrectly classified by classifier *n* are given more importance in the training of model *n + 1*, while in bagging the training samples are taken randomly from the whole population

(Source: https: //towardsdatascience.com/What is Boosting in Machine Learning?)

3.3 How is a Boosting Model Trained?

1. All the data samples start with the same weights. These samples are used to train an individual model (a Decision Tree lets say).
2. The prediction error for each sample is calculated, increasing the weights of those samples which have had a greater error, to make them more important for the training of following individual model.
3. Depending on how well this individual model did on its predictions, it gets assigned an importance/weight or amount of say. A model that outputs very good predictions will have a high amount of say in the final decision.
4. The weighted data is passed on to the posterior model, and 2) and 3) are repeated.
5. Number 4) is repeated until we have reached an certain number of models or until the error is bellow a certain threshold.

(Source: https: //towardsdatascience.com/What is Boosting in Machine Learning?)

3.4 Boosting Algorithm

Algorithm 11.3: Boosting($D, T, \mathscr{A}$) – train an ensemble of binary classifiers from reweighted training sets.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.
Output : weighted ensemble of models.

$w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
for $t = 1$ to T **do**
  run $\mathscr{A}$ on D with weights w_{ti} to produce a model M_t;
  calculate weighted error ϵ_t;
  if $\epsilon_t \geq 1/2$ **then**
    set $T \leftarrow t-1$ and break
  end
  $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
  $w_{(t+1)i} \leftarrow \frac{w_{ti}}{2\epsilon_t}$ for misclassified instances $x_i \in D$; // increase weight
  $w_{(t+1)j} \leftarrow \frac{w_{tj}}{2(1-\epsilon_t)}$ for correctly classified instances $x_j \in D$; // decrease weight
end
return $M(x) = \sum_{t=1}^{T} \alpha_t M_t(x)$

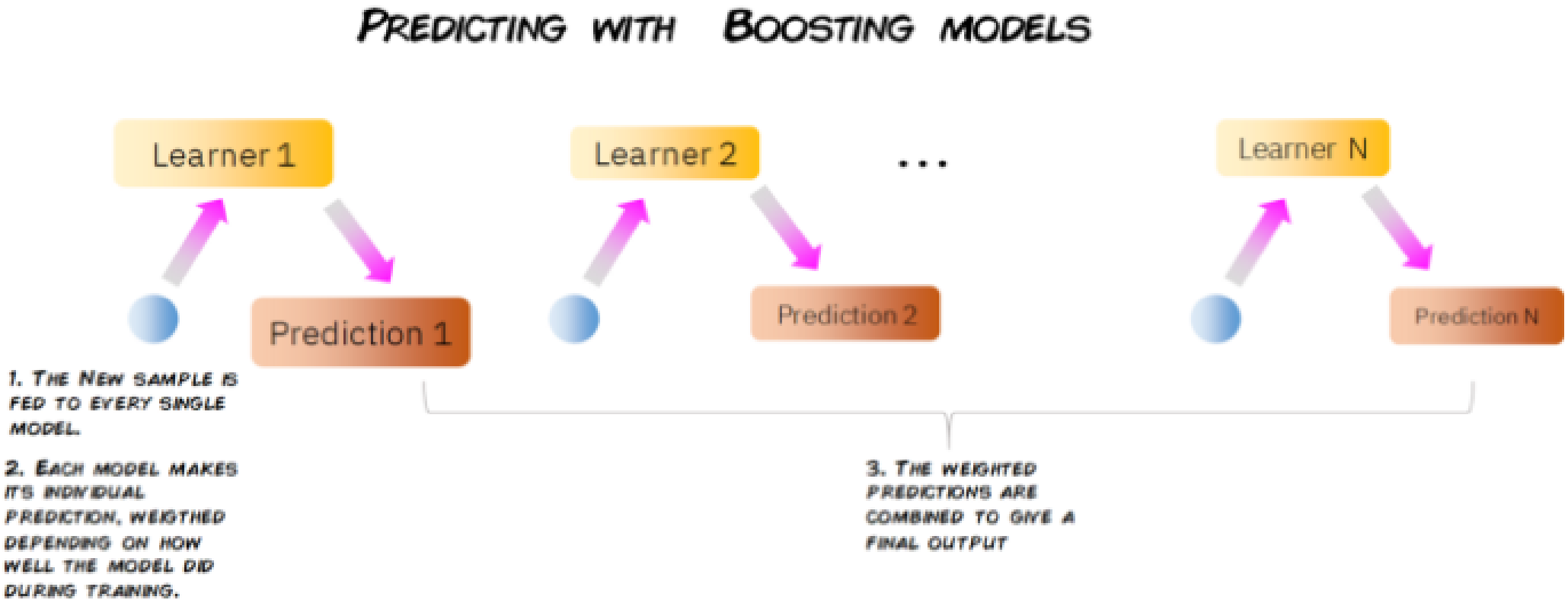
Fig 9. Boosting algorithm. (Flach, 2017)

3.5 Predicting with Boosting Models

- They way a boosting model makes predictions on new data is very simple. When we get a new observation with its features, it is passed through every one of the individual models, having each model make its own prediction.
- Then, taking into account the weight of each one of these models, all these predictions are scaled and combined, and a final global prediction is given.

(Source : towardsdatascience.com)

3.5 Predicting with boosting models



Predicting with boosting models

- Fig 10. Boosting predictions (towardscience.com)

3.6 Example of Boosting using Gradient Boost

- Following is a sample from a random dataset where we have to predict the car price based on various features. The target column is price and other features are independent features.

Row No.	Cylinder Number	Car Height	Engine Location	Price
1	Four	48.8	Front	12000
2	Six	48.8	Back	16500
3	Five	52.4	Back	15500
4	Four	54.3	Front	14000

Table 1. Dataset (Source : https://www.analyticsvidhya.com/)

3.7 Example (cont'd)

- **Step -1** The first step in gradient boosting is to build a base model to predict the observations in the training dataset. For simplicity we take an average of the target column and assume that to be the predicted value as shown below:

Row No.	Cylinder Number	Car Height	Engine Location	Price	Prediction 1
1	Four	48.8	Front	12000	14500
2	Six	48.8	Back	16500	14500
3	Five	52.4	Back	15500	14500
4	Four	54.3	Front	14000	14500

Table 2. Dataset with prediction (Source : https://www.analyticsvidhya.com/)

3.3 Example (cont'd)

- **Step-2** The next step is to calculate the pseudo residuals which are (observed value – predicted value)

Row No.	Cylinder Number	Car Height	Engine Location	Price	Prediction 1	Residual 1
1	Four	48.8	Front	12000	14500	-2500
2	Six	48.8	Back	16500	14500	2000
3	Five	52.4	Back	15500	14500	1000
4	Four	54.3	Front	14000	14500	-500

Table 3. Dataset with residual (Source : https://www.analyticsvidhya.com/)

3.3 Example (cont'd)

- **Step- 3** In this step we find the output values for each leaf of our decision tree. That means there might be a case where 1 leaf gets more than 1 residual, hence we need to find the final output of all the leaves. To find the output we can simply take the average of all the numbers in a leaf, doesn't matter if there is only 1 number or more than 1.
- Suppose this is our regressor tree:

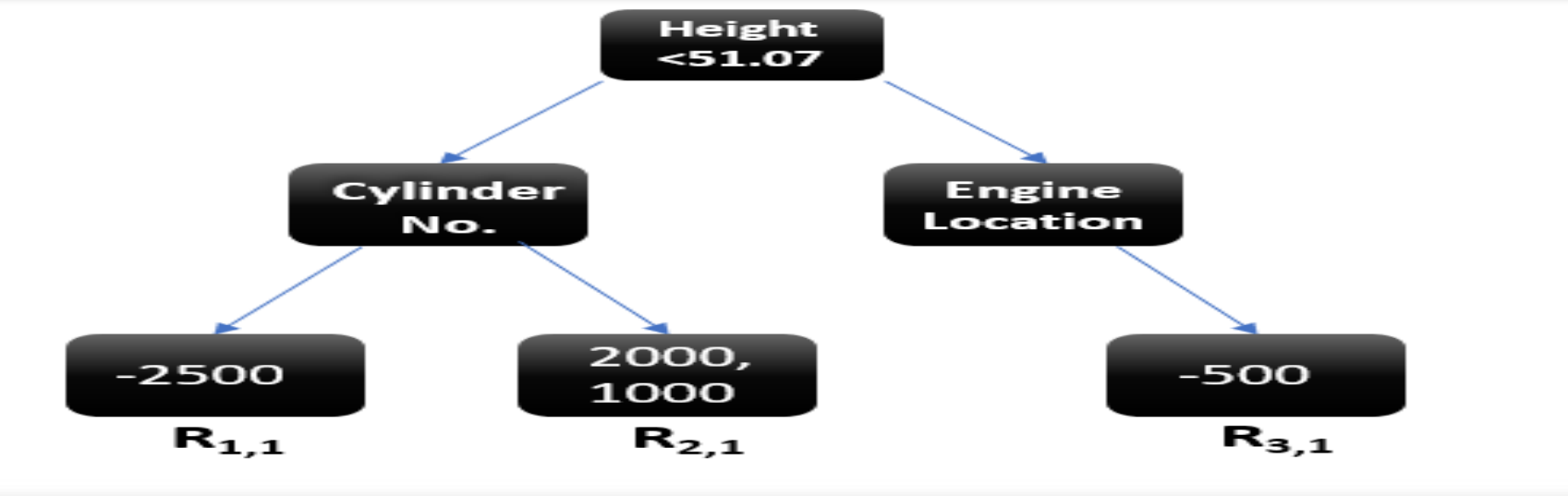


Fig 10. Regressor tree (Source : https://www.analyticsvidhya.com/)

3.3 Example (cont'd)

- We see 1st residual goes in R1,1 ,2nd and 3rd residuals go in R2,1 and 4th residual goes in R3,1 .
- Step 4 calculate the **average** of the residuals in the leaf $R_{2,1}$. Hence if we get any leaf with more than 1 residual, we can simply find the average of that leaf and that will be our final output.

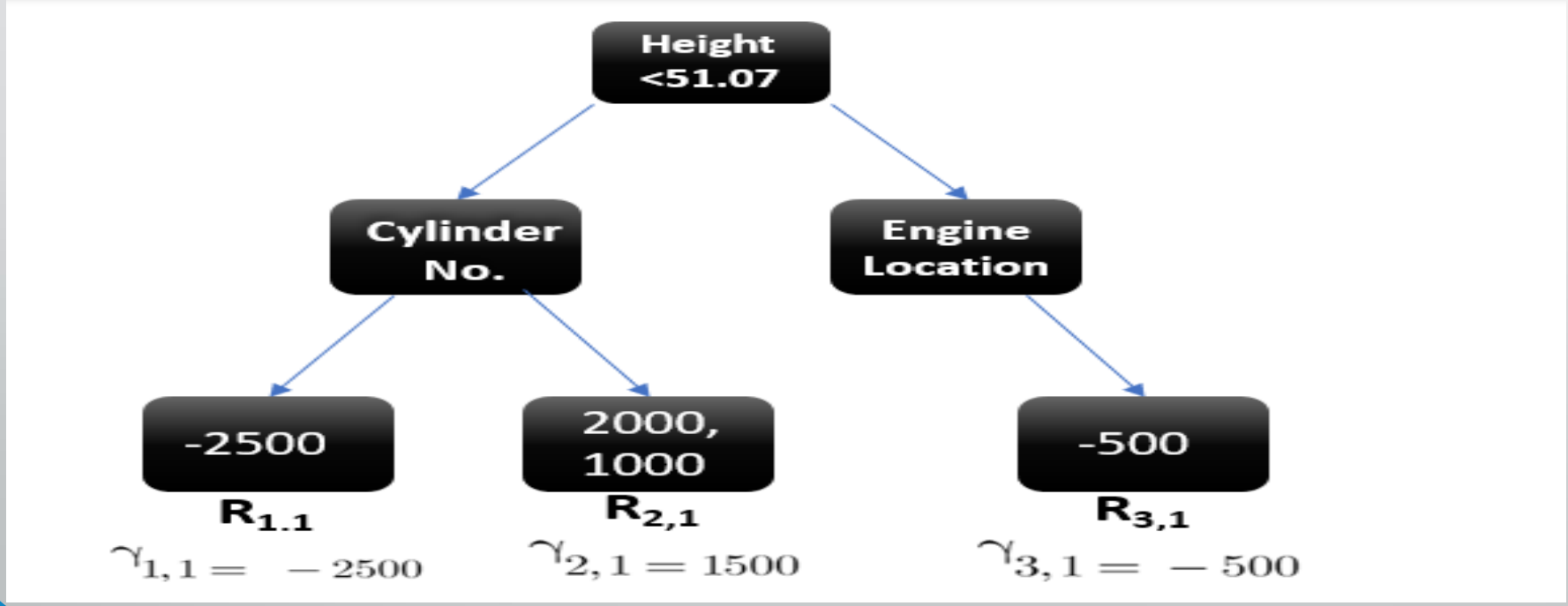


Fig 11. Averages calculations(Source : https://www.analyticsvidhya.com/)

3.3 Example (cont'd)

Step-5 This is finally the last step where we have to update the predictions of the previous model. It can be updated as:

Since we have just started building our model so our m=1. Now to make a new DT our new predictions will be:

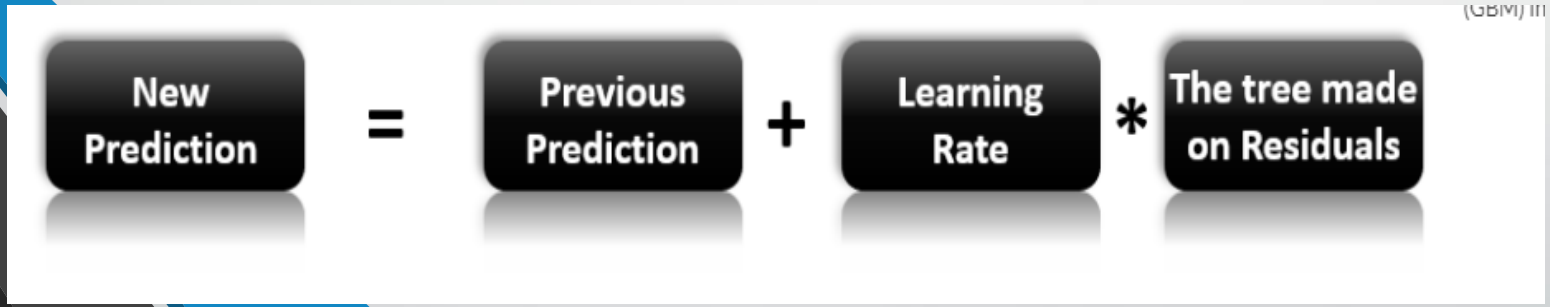


Fig 12. New predictions (Source : https://www.analyticsvidhya.com/)

3.3 Example (cont'd)

- ***Learning rate*** that is usually selected between **0-1**. It reduces the effect each tree has on the final prediction, and this improves accuracy in the long run. Let's take ***learning rate=0.1*** in this example.

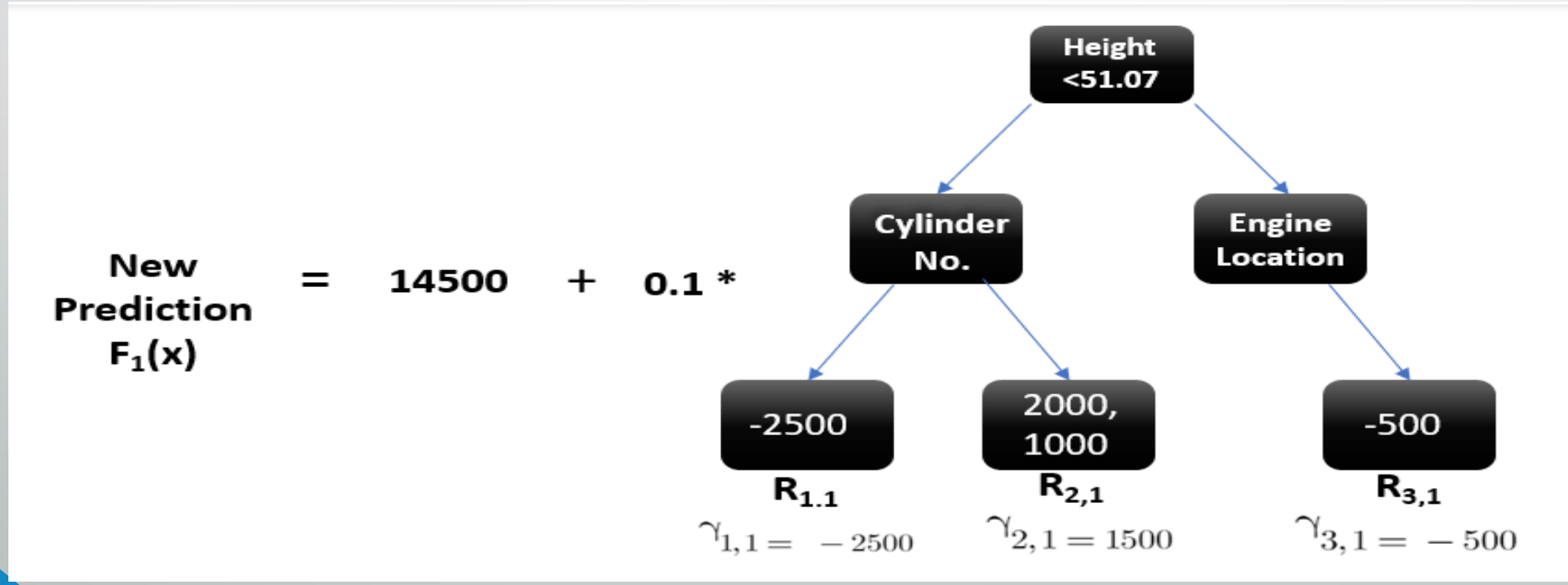


Fig 13. New prediction (Source : https://www.analyticsvidhya.com/)

3.3 Example (cont'd)

- Suppose we want to find a prediction of our first data point which has a car height of 48.8. This data point will go through this decision tree and the output it gets will be multiplied with the learning rate and then added to the previous prediction.
- Now let's say m=2 which means we have built 2 decision trees and now we want to have new predictions.
- This time we will add the previous prediction that is $\mathbf{F_1(x)}$ to the new DT made on residuals. We will iterate through these steps again and again till the ***loss is negligible.***

3.3 Example (cont'd)

- Let us take an example theoretically that will enable us to understand how this predicts for a new dataset:

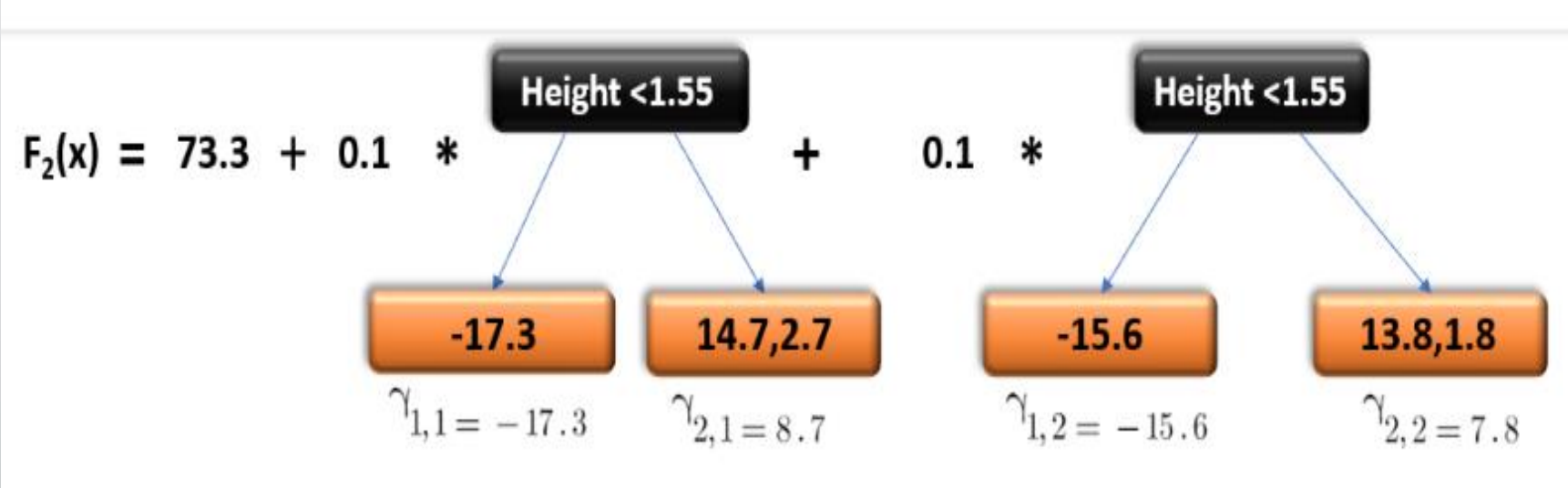


Fig 14. Example dataset (Source : https://www.analyticsvidhya.om/)

If a new data point says height = 1.40 comes, it'll go through all the trees and then will give the prediction. Here we have only 2 trees hence the datapoint will go through these 2 trees and the final output will be F2(x).

3.4 Bagging vs Boosting

Similarities

- Bagging and Boosting are ensemble methods focused on getting N learners from a single learner.
- Bagging and Boosting make random sampling and generate several training data sets
- Bagging and Boosting arrive upon the end decision by making an average of N learners or taking the voting rank done by most of them.
- Bagging and Boosting reduce variance and provide higher stability with minimizing errors.

3.4 Bagging vs Boosting(Cont'd)

- **Differences.**
- Bagging is a method of merging the same type of predictions. Boosting is a method of merging different types of predictions.
- Bagging decreases variance, not bias, and solves over-fitting issues in a model. Boosting decreases bias, not variance.
- In Bagging, each model receives an equal weight. In Boosting, models are weighed based on their performance.
- Models are built independently in Bagging. New models are affected by a previously built model's performance in Boosting.

Summary

- All ensemble methods have in common is that they construct several base models from adapted versions of the training data, on top of which some technique is employed to combine the predictions or scores from the base models into a single prediction of the ensemble.
- The key idea in boosting is to train diverse models by increasing the weight of previously misclassified examples. This helps to reduce the bias of otherwise stable learners such as linear classifiers or decision stumps.
- Bagging trains diverse models from samples of the training data, These techniques are particularly useful to reduce the variance of low-bias models such as tree models.

References

- Ayuya, C. (2021, January 20). *Ensemble learning on bias and variance*. Section. Retrieved April 20, 2022, from https://www.section.io/engineering-education/ensemble-bias-var/
- *Bagging*. MBASkillsIN. (n.d.). Retrieved April 20, 2022, from http://mbaskills.in/bagging/
- Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.
- Mohru, M. (2012). *Fondations of Machine Learning*. MIT Press.
- Data Analytics Skills using R programming(nd). From https://Mbaskills.IN/bagging
- Biswal, A. (2021). *Bagging in Machine Learning: Step to Perform And Its Advantages.Retreived from : https://simplelearn/tutorials/machine-learning-tutorial/*What is Bagging in Machine Learning And How to Perform Bagging.
- Flach, P. (2012). *Stacked Framework for Ensemble of Heterogeneous Classification Algorithms*. Cambridge University Press.
- Flach, P. A. (2017). *Machine learning: The art and science of algorithms that make sense of data*. Cambridge University Press.

References

- Petrakova, A. , Affenzeller, M., & Merkuryeva, G. (2015). Heterogeneous versus Homogeneous Machine Learning Ensembles. Information Technology and Management Science. 18. 10.1515/itms-2015-0021.
- Saini, A(2021). *Gradient Boosting Algorithm: A Complete Guide for Beginners. Retrived from : https: //Analyticsvoya.com/ Boosting Algorithm: A Complete Guide for Beginners (analyticsvidhya.com)*
- Theobald, O. (2021). *Machine learning for absolute beginners: A Plain english introduction*. nakładem autora.
- What is boosting in Machine Learning. Retrieved June, 8 ,2020. From https: //towardsdatascience.com/What is Boosting in Machine Learning?