Machine Learning I

Mohamed Hussien

Ensemble Learning

What Is Ensemble Learning?

Imagine you need to choose people to take their votes to help you to answer questions.

You have two options:

- Choosing some of your friends randomly

What Is Ensemble Learning?

Imagine you need to choose people to take their votes to help you to answer questions.

You have two options:

- Choosing some of your friends randomly

What Is Ensemble Learning?

Imagine you need to choose people to take their votes to help you to answer questions.

You have two options:

- Choosing some of your friends randomly

- Choosing experts in different fields to complete each other

What Is Ensemble Learning?

Imagine you need to choose people to take their votes to help you to answer questions.

You have two options:

- Choosing some of your friends randomly

bagging

- Choosing experts in different fields to complete each other

What Is Ensemble Learning?

Imagine you need to choose people to take their votes to help you to answer questions.

You have two options:

- Choosing some of your friends randomly

bagging

- Choosing experts in different fields to complete each other

boosting

What Is Ensemble Learning?

Imagine you need to choose people to take their votes to help you to answer questions.

You have two options:

- Choosing some of your friends randomly

bagging

ensemble learning

- Choosing experts in different fields to complete each other

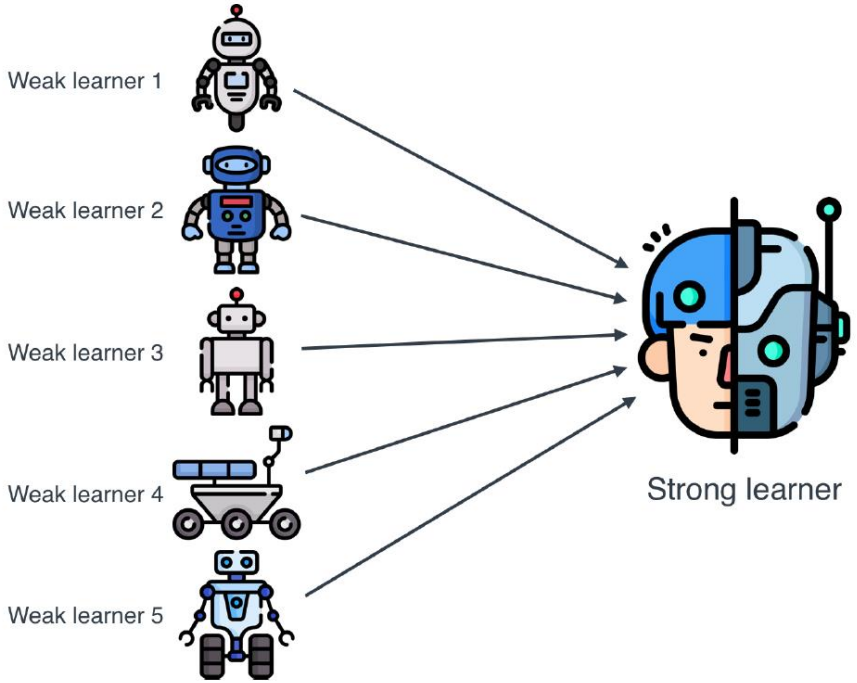
boosting

What Is Ensemble Learning?

Ensemble Learning:
Combine weak learners to form a strong one.

Learners could be any type of
machine learning model.

Ensemble Learning types:
- Bagging
- Boosting



Bagging

Bagging

A technique in which we build a strong learner based on a set of weak learners **trained on data splitted randomly** and making predictions by **voting**.

If these learners are **decision trees**, then our model is called **Random Forest**.

Random Forest Steps:

- Split data **randomly** into sets:
- Train each decision tree at one of these sets.
- Take decision based on **vote** between these models.

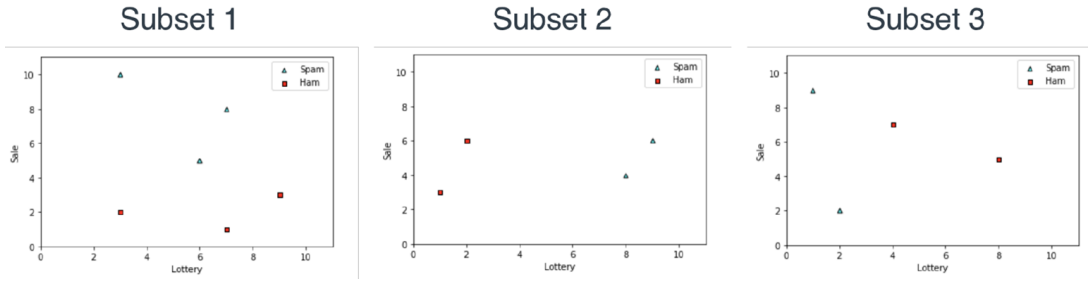
Bagging

A technique in which we build a strong learner based on a set of weak learners **trained on data splitted randomly** and making predictions by **voting**.

If these learners are **decision trees**, then our model is called **Random Forest**.

Random Forest Steps:

- Split data **randomly** into sets:
- Train each decision tree at one of these sets.
- Take decision based on **vote** between these models.



Bagging

A technique in which we build a strong learner based on a set of weak learners **trained on data splitted randomly** and making predictions by **voting**.

If these learners are **decision trees**, then our model is called **Random Forest**.

Random Forest Steps:

- Split data **randomly** into sets:
- Train each decision tree at one of these sets.
- Take decision based on **vote** between these models.

Boosting

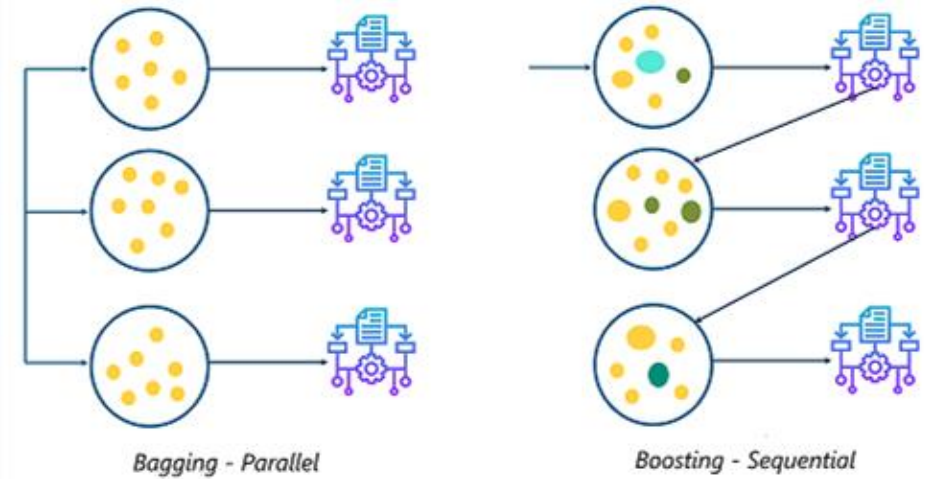
A technique in which we build a strong learner based on a set of weak learners **not randomly selected,but trained in a smarter way** and making predictions by **voting**. Ex: AdaBoost, XGBoost

Each model trained to **cover the weakness of the previous ones**.

Boosting

A technique in which we build a strong learner based on a set of weak learners **not randomly selected,but trained in a smarter way** and making predictions by **voting**. Ex: AdaBoost, XGBoost

Each model trained to **cover the weakness of the previous ones**.



AdaBoost(Adaptive Boosting)

AdaBoost(Adaptive Boosting)

Using very simple learners (1-depth decision tree) to build a strong learner

Steps:

- Train the first learner
- **Increase the weight** of the data samples that learner **predicts incorrectly**
- **Decrease the weight** of the data samples that learner **predicts correctly**
- Train another learner using **the modified dataset**
- Repeat steps 2,3,4 for all other learners
- Make the prediction using the weighted voting based on the performance of each learner

AdaBoost could be used for classification or regression. Our example is a classification problem.

AdaBoost [Big Picture]

AdaBoost [Big Picture]

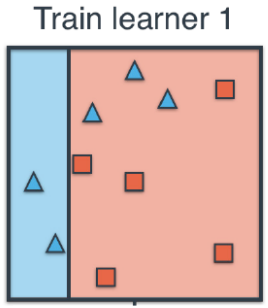
Building the learners

Our learners are very simple.
Ex: 1-Depth Decision Tree

AdaBoost [Big Picture]

Building the learners

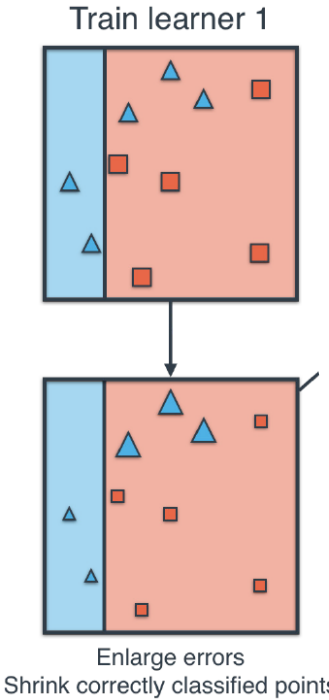
Our learners are very simple.
Ex: 1-Depth Decision Tree



AdaBoost [Big Picture]

Building the learners

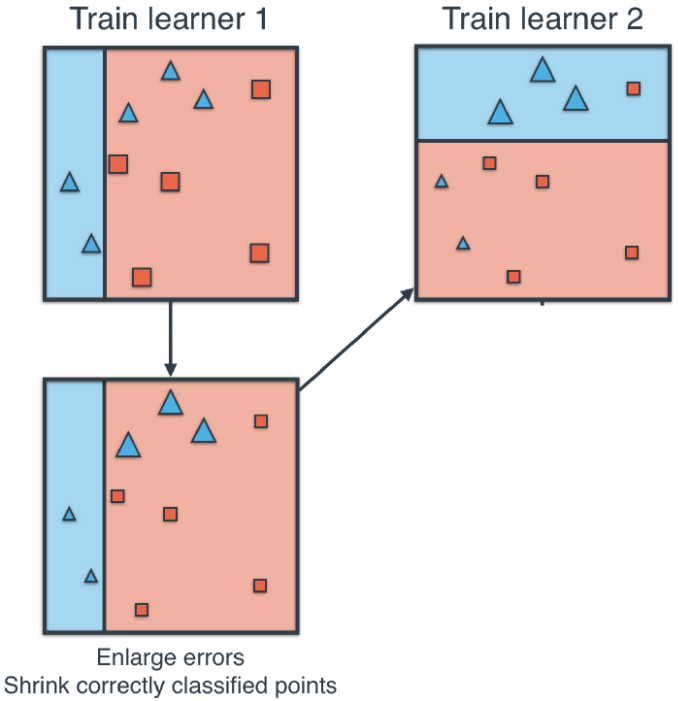
Our learners are very simple.
Ex: 1-Depth Decision Tree



AdaBoost [Big Picture]

Building the learners

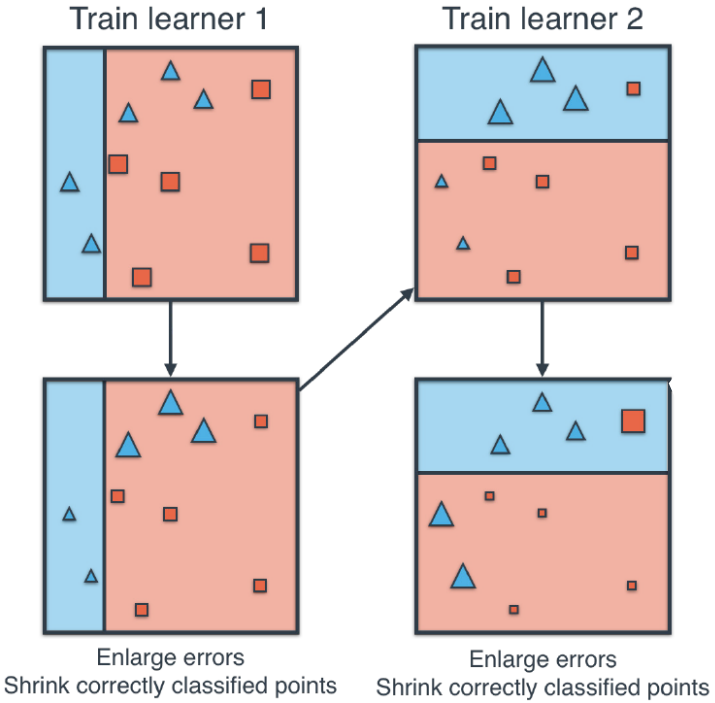
Our learners are very simple.
Ex: 1-Depth Decision Tree



AdaBoost [Big Picture]

Building the learners

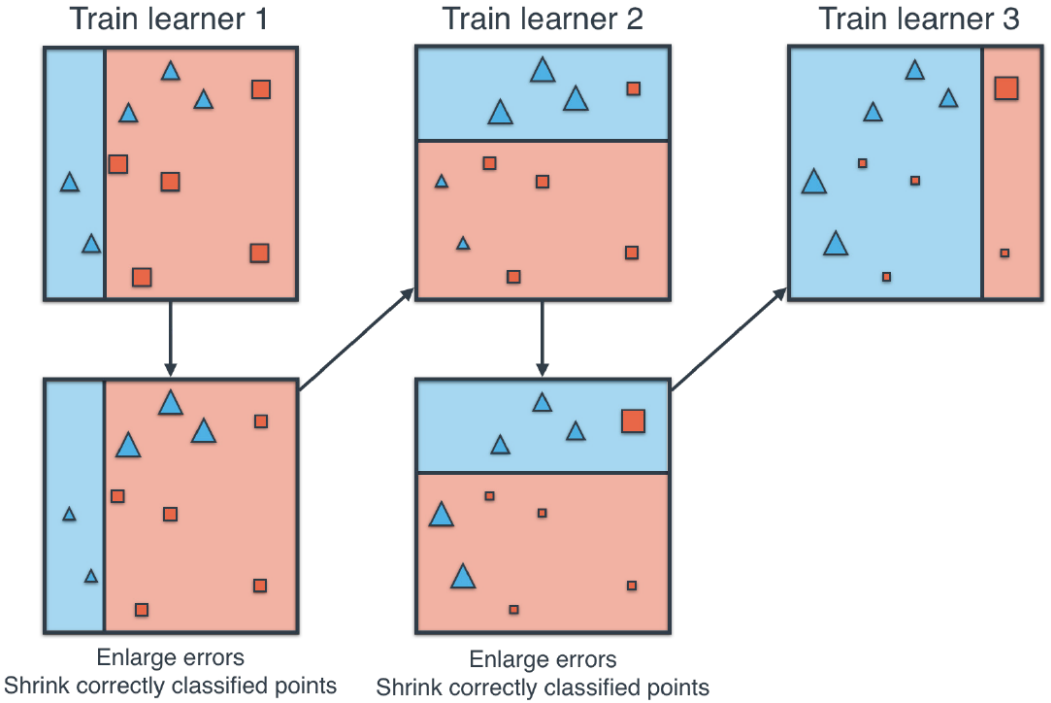
Our learners are very simple.
Ex: 1-Depth Decision Tree



AdaBoost [Big Picture]

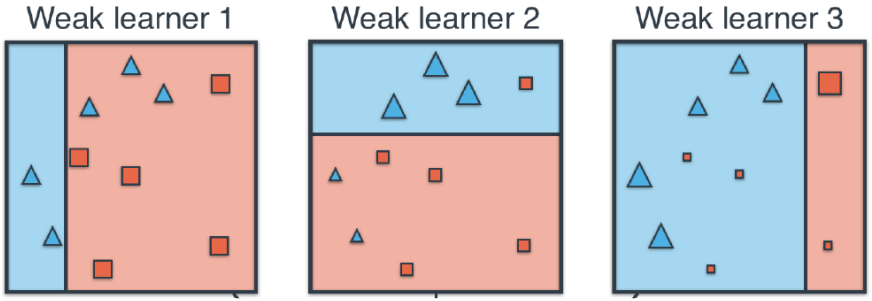
Building the learners

Our learners are very simple.
Ex: 1-Depth Decision Tree



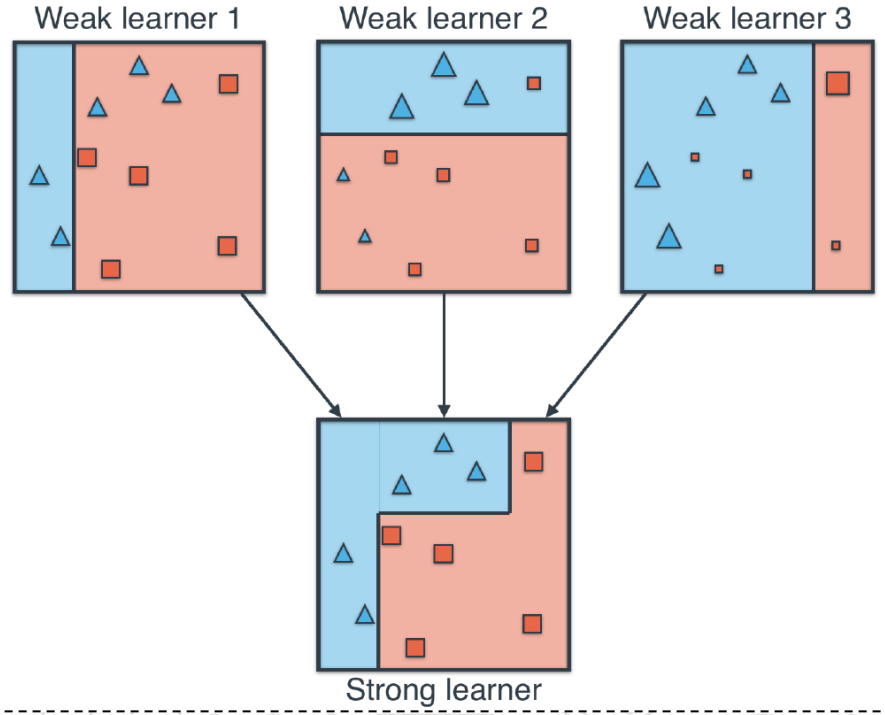
AdaBoost [Big Picture]

Prediction



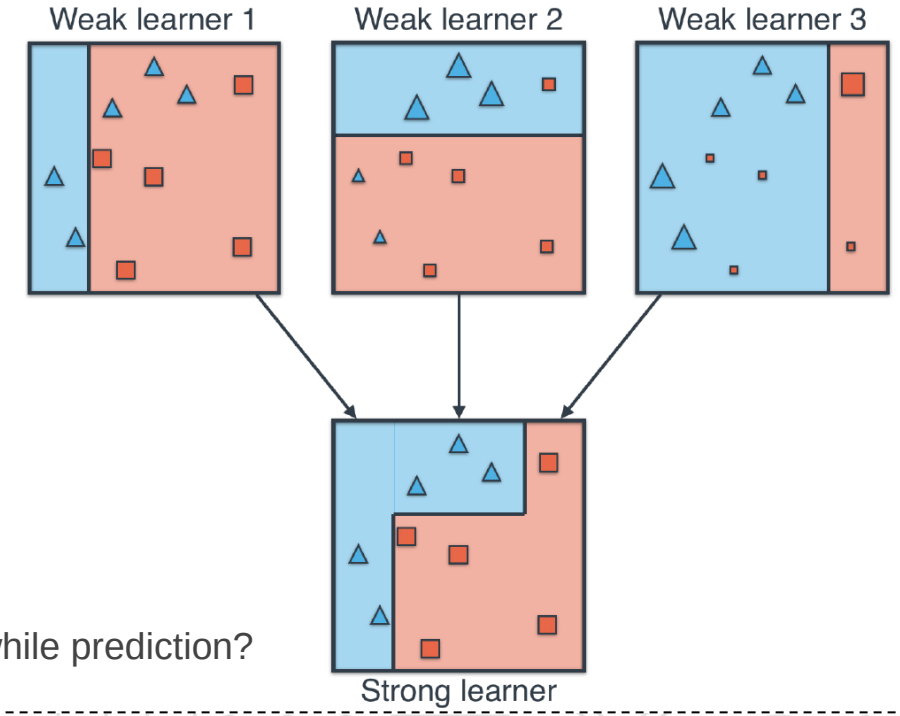
AdaBoost [Big Picture]

Prediction



AdaBoost [Big Picture]

Prediction



Need to answer two questions:

- How to change the points weights?
- How to choose the weight of each learner while prediction?

AdaBoost [In Details]

How to update the weights?

Let’s introduce the **Odd Ratio**

AdaBoost [In Details]

How to update the weights?

Let’s introduce the **Odd Ratio**

Odd Ratio =

AdaBoost [In Details]

How to update the weights?

Let's introduce the **Odd Ratio**

$$\text{Odd Ratio} = \frac{\text{Number of correctly classified points}}{\text{Number of incorrectly classified points}}$$

AdaBoost [In Details]

How to update the weights?

Let's introduce the **Odd Ratio**

$$\text{Odd Ratio} = \frac{\text{Number of correctly classified points}}{\text{Number of incorrectly classified points}} = \text{Rescaling Factor}$$

AdaBoost [In Details]

How to update the weights?

Let's introduce the **Odd Ratio**

$$\text{Odd Ratio} = \frac{\text{Number of correctly classified points}}{\text{Number of incorrectly classified points}} = \text{Rescaling Factor}$$

$$= \frac{\text{Accuracy}}{\text{1- Accuracy}}$$

AdaBoost [In Details]

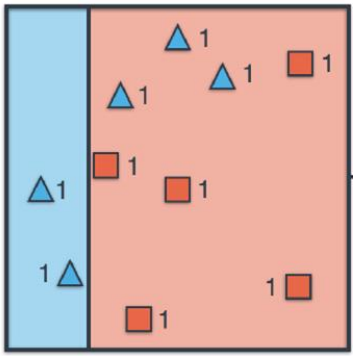
How to update the weights?

Let's introduce the **Odd Ratio**

$$\text{Odd Ratio} = \frac{\text{Number of correctly classified points}}{\text{Number of incorrectly classified points}} = \text{Rescaling Factor}$$

$$= \frac{\text{Accuracy}}{\text{1- Accuracy}}$$

ex: 1st learner



AdaBoost [In Details]

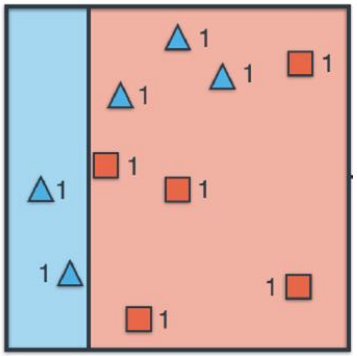
How to update the weights?

Let's introduce the **Odd Ratio**

$$\text{Odd Ratio} = \frac{\text{Number of correctly classified points}}{\text{Number of incorrectly classified points}} = \text{Rescaling Factor}$$

$$= \frac{\text{Accuracy}}{1 - \text{Accuracy}}$$

ex: 1st learner



Rescaling Factor = 7/3 = 2.33

AdaBoost [In Details]

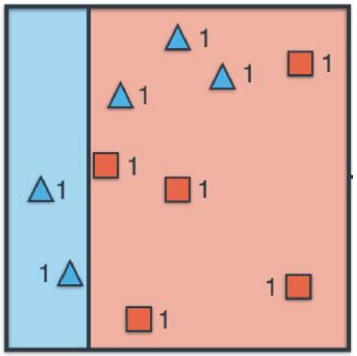
How to update the weights?

Let's introduce the **Odd Ratio**

$$\text{Odd Ratio} = \frac{\text{Number of correctly classified points}}{\text{Number of incorrectly classified points}} = \text{Rescaling Factor}$$

$$= \frac{\text{Accuracy}}{1 - \text{Accuracy}}$$

ex: 1st learner



Rescaling Factor = 7/3 = 2.33

Could rescaling factor be < 1?

AdaBoost [In Details]

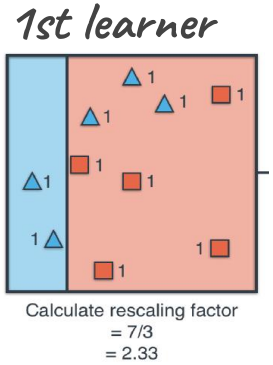
How to update the weights? [Steps]

- Train the learner
- Calculate the rescaling factor
- Multiply incorrected classified Points by the rescaling factor
- Normalize the weights

AdaBoost [In Details]

How to update the weights? [Steps]

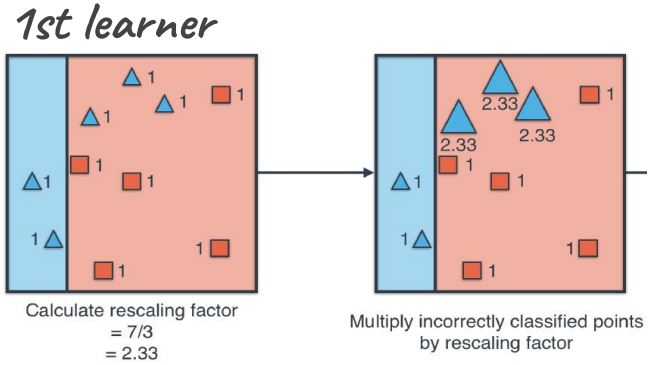
- Train the learner
- Calculate the rescaling factor
- Multiply incorrected classified Points by the rescaling factor
- Normalize the weights



AdaBoost [In Details]

How to update the weights? [Steps]

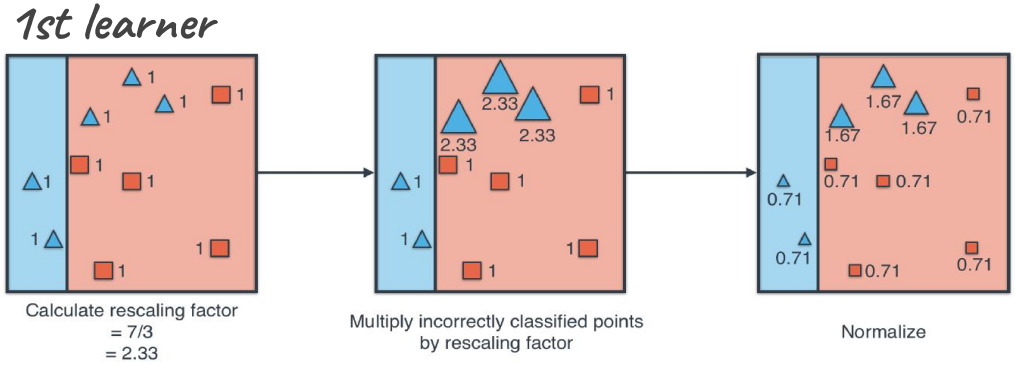
- Train the learner
- Calculate the rescaling factor
- Multiply incorrected classified Points by the rescaling factor
- Normalize the weights



AdaBoost [In Details]

How to update the weights? [Steps]

- Train the learner
- Calculate the rescaling factor
- Multiply incorrected classified Points by the rescaling factor
- Normalize the weights

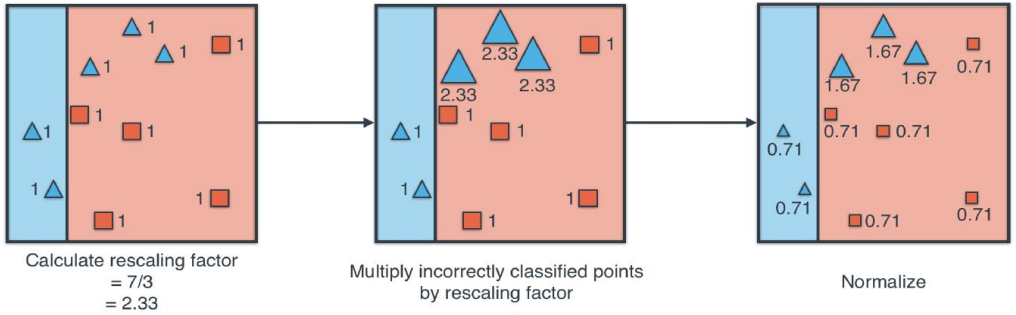


AdaBoost [In Details]

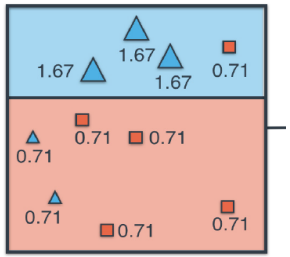
How to update the weights? [Steps]

- Train the learner
- Calculate the rescaling factor
- Multiply incorrected classified Points by the rescaling factor
- Normalize the weights

1st learner



2nd learner

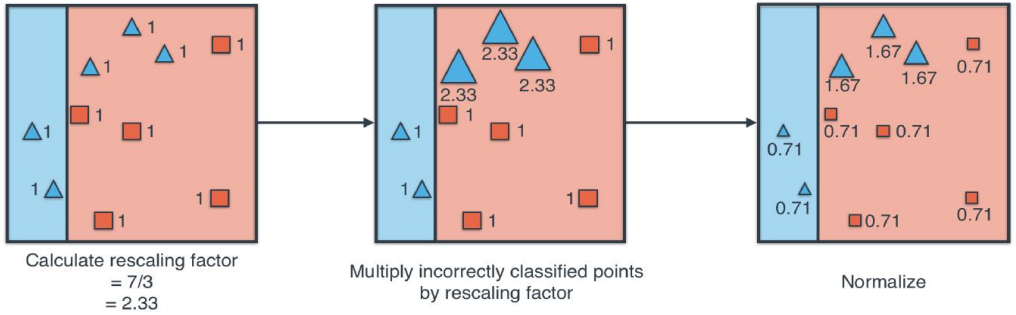


AdaBoost [In Details]

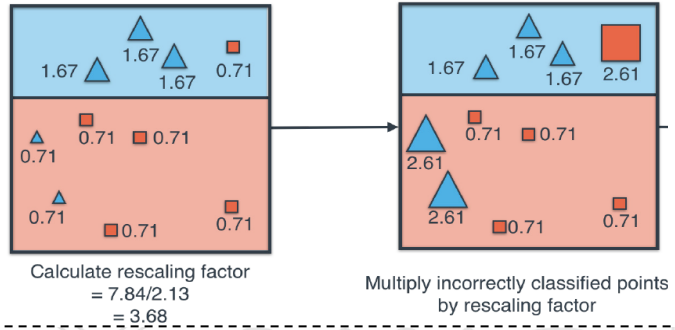
How to update the weights? [Steps]

- Train the learner
- Calculate the rescaling factor
- Multiply incorrected classified Points by the rescaling factor
- Normalize the weights

1st learner



2nd learner

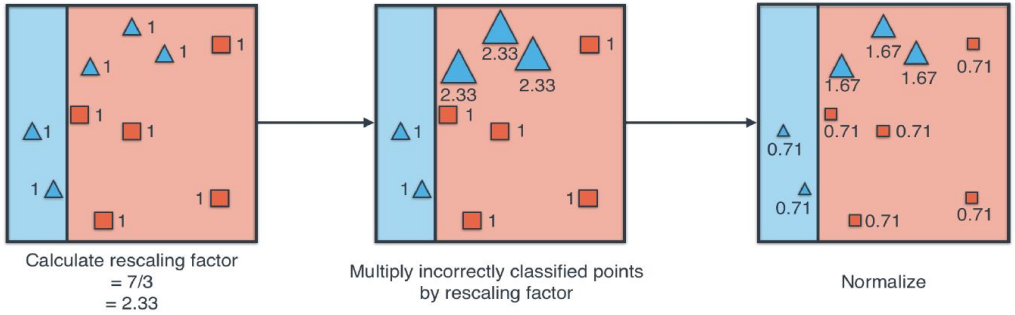


AdaBoost [In Details]

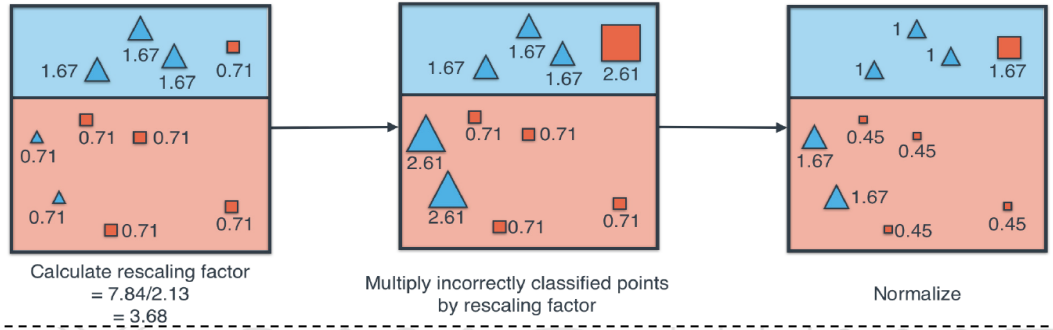
How to update the weights? [Steps]

- Train the learner
- Calculate the rescaling factor
- Multiply incorrected classified Points by the rescaling factor
- Normalize the weights

1st learner



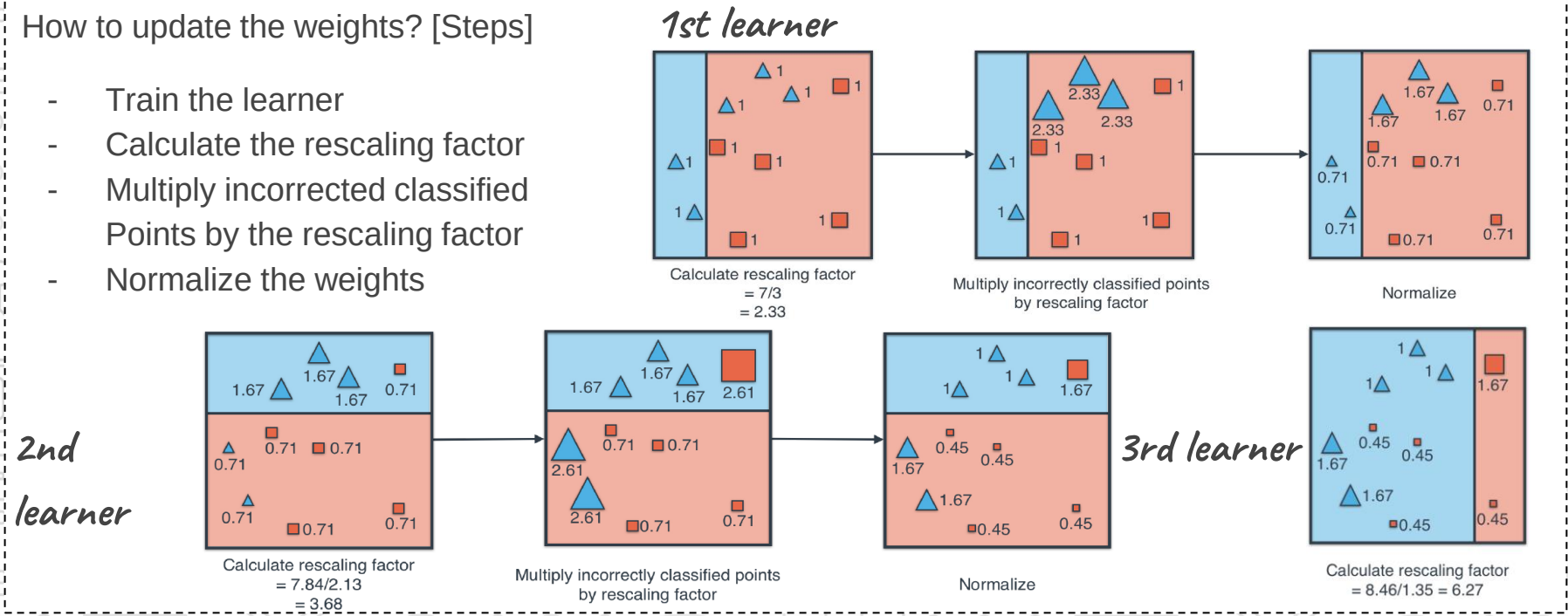
2nd learner



AdaBoost [In Details]

How to update the weights? [Steps]

- Train the learner
- Calculate the rescaling factor
- Multiply incorrected classified Points by the rescaling factor
- Normalize the weights



AdaBoost [In Details]

AdaBoost [In Details]

What does point weight mean?

We create a new dataset with the same number of elements by resampling from the old dataset according to points weights.

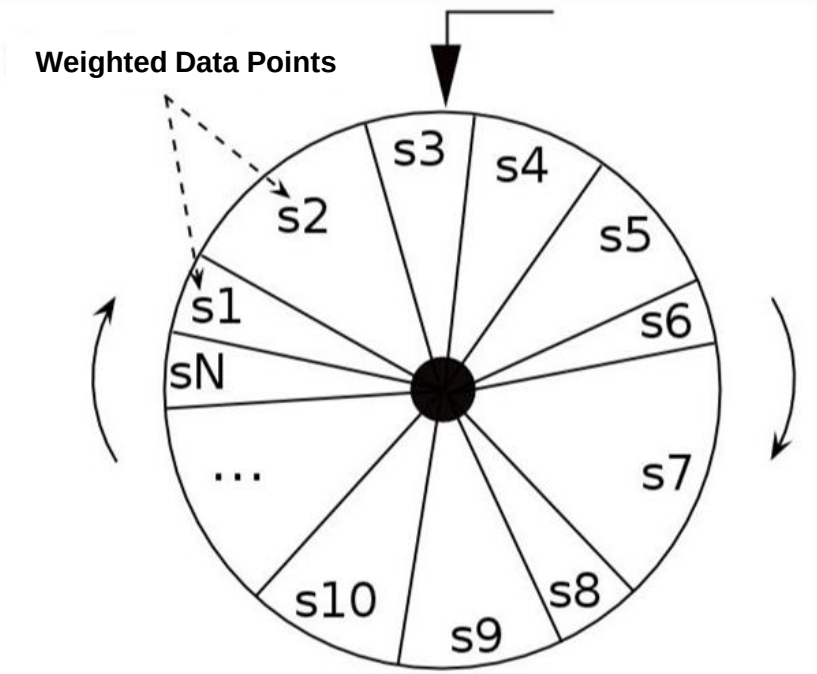
It’s like a roulette wheel as the following.

AdaBoost [In Details]

What does point weight mean?

We create a new dataset with the same number of elements by resampling from the old dataset according to points weights.

It’s like a roulette wheel as the following.



AdaBoost [In Details]

AdaBoost [In Details]

Weight of each learner while prediction?

We can use **odd ratio** of the learner as a weight for that learner

BUT, Which model is better accuracy?
99%, 70%, 50%, 30%, 1%

If we inverted the 1% accuracy model predictions.
It will equal to the 99% accuracy.

We need a function that
multiply the weight by (-) if accuracy<50 (odd ratio<1)
And multiply the weight by (+) if accuracy>50 (odd ratio>1)

So we use the **ln(odd ratio)** of the learner to be the weight of this learner

AdaBoost [In Details]

Weight of each learner while prediction?

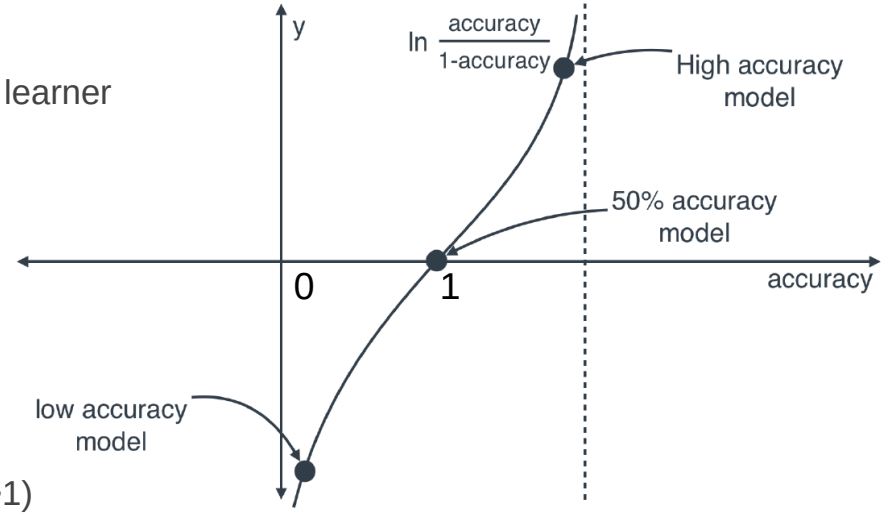
We can use **odd ratio** of the learner as a weight for that learner

BUT, Which model is better accuracy?
99%, 70%, 50%, 30%, 1%

If we inverted the 1% accuracy model predictions.
It will equal to the 99% accuracy.

We need a function that
multiply the weight by (-) if accuracy<50 (odd ratio<1)
And multiply the weight by (+) if accuracy>50 (odd ratio>1)

So we use the **ln(odd ratio)** of the learner to be the weight of this learner



AdaBoost [In Details]

Weight of each learner while prediction?

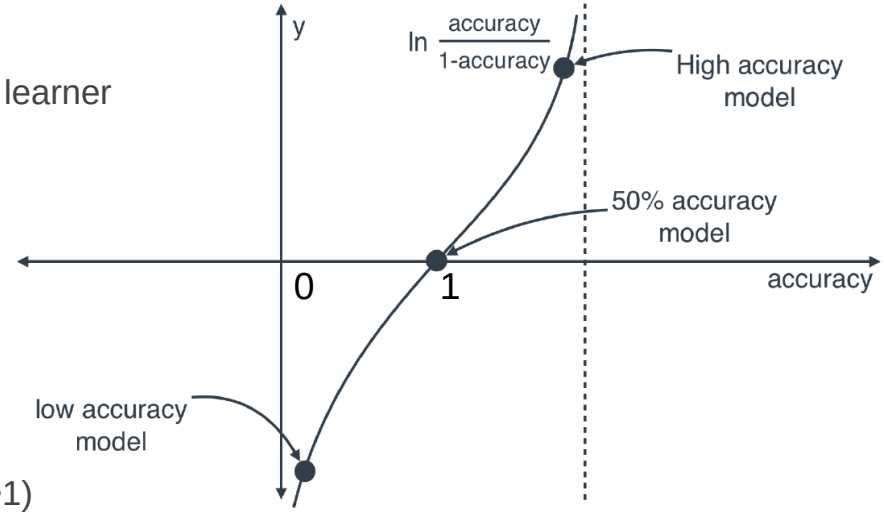
We can use **odd ratio** of the learner as a weight for that learner

BUT, Which model is better accuracy?
99%, 70%, 50%, 30%, 1%

If we inverted the 1% accuracy model predictions.
It will equal to the 99% accuracy.

We need a function that
multiply the weight by (-) if accuracy<50 (odd ratio<1)
And multiply the weight by (+) if accuracy>50 (odd ratio>1)

So we use the **ln(odd ratio)** of the learner to be the weight of this learner



Notice that for the low-accuracy models we **increased the weight of the true points** then we **inverted the prediction**

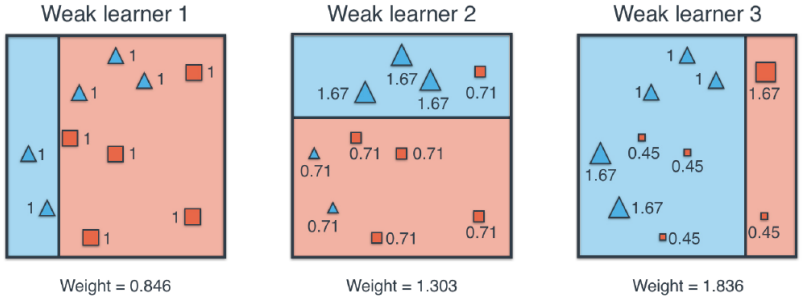
AdaBoost [In Details]

AdaBoost [In Details]

Weight of each learner while prediction?

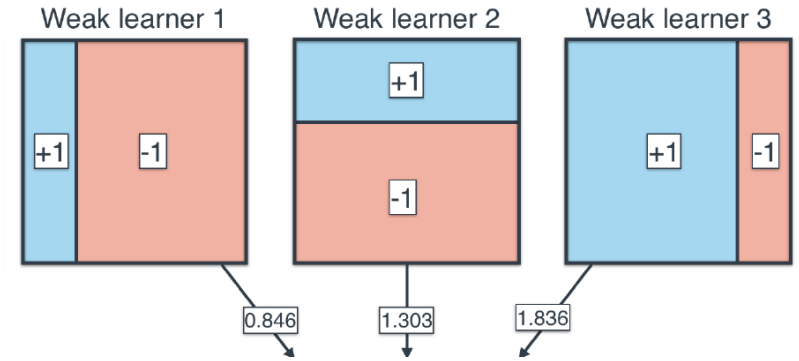
AdaBoost [In Details]

Weight of each learner while prediction?



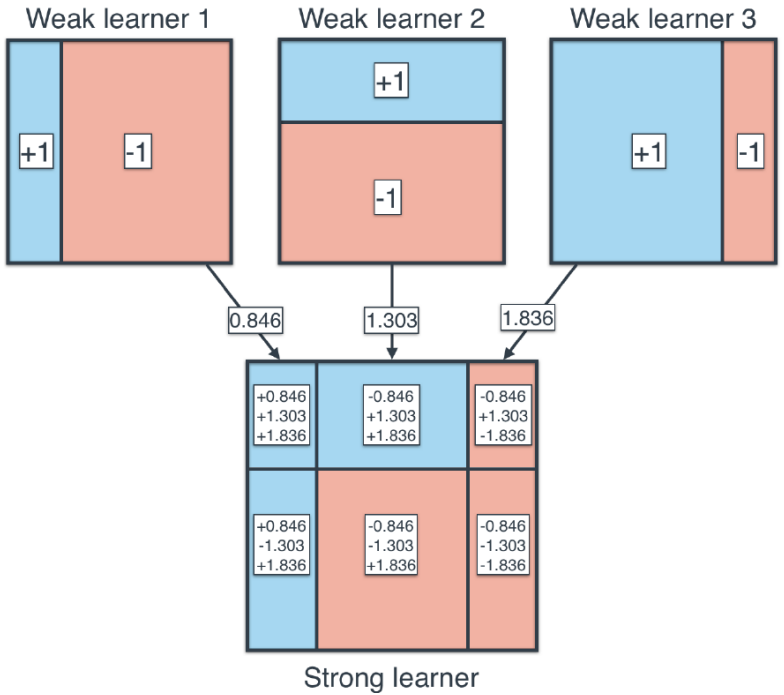
AdaBoost [In Details]

Weight of each learner while prediction?



AdaBoost [In Details]

Weight of each learner while prediction?



Coding AdaBoost

Use colab to open this github notebook:

“s7s/machine_learning_1/ensemble_methods/AdaBoost.ipynb”

XGBoost

XGBoost

AdaBoost consists of 1-depth weak learners. What if we build a similar algorithm with deeper trees?

XGBoost could be used for classification or regression. Our example is a regression problem.

Steps:

- Start with a naive learner that predicts 0.5 in all cases
- Calculate residual between predicted and label
- Build a new learner to predict the residual
- Repeat step 2,3 for other learners
- Combine all learners to get the prediction

XGBoost

AdaBoost consists of 1-depth weak learners. What if we build a similar algorithm with deeper trees?

XGBoost could be used for classification or regression. Our example is a regression problem.

Steps:

- Start with a naive learner that predicts 0.5 in all cases
- Calculate residual between predicted and label
- Build a new learner to predict the residual
- Repeat step 2,3 for other learners
- Combine all learners to get the prediction

Need to answer two questions:

- How to build the learner?
- How to combine the learners to make the prediction?

XGBoost

XGBoost

How to build the learner?

In XGBoost we use a similarity instead of Gini and Entropy

XGBoost

How to build the learner?

In XGBoost we use a similarity instead of Gini and Entropy

$$\text{Similarity} = \frac{\text{Square of sum of labels of the set}}{\text{Number of points of the set} + \lambda}$$

XGBoost

How to build the learner?

In XGBoost we use a similarity instead of Gini and Entropy

$$\text{Similarity} = \frac{\text{Square of sum of labels of the set}}{\text{Number of points of the set} + \lambda} \qquad \text{let } \lambda = 0$$

XGBoost

How to build the learner?

In XGBoost we use a similarity instead of Gini and Entropy

$$\text{Similarity} = \frac{\text{Square of sum of labels of the set}}{\text{Number of points of the set} + \lambda} \qquad \text{let } \lambda = 0$$

Example: If we have three sets with these labels {10, -10, 4}, {7, 7, 7}, {7}

XGBoost

How to build the learner?

In XGBoost we use a similarity instead of Gini and Entropy

$$Similarity = \frac{\text{Square of sum of labels of the set}}{\text{Number of points of the set} + \lambda} \qquad let\ \lambda = 0$$

Example: If we have three sets with these labels {10, -10, 4}, {7, 7, 7}, {7}

Set 1: $Similarity = \frac{(10-10+4)^2}{3} = \frac{16}{3}$.

XGBoost

How to build the learner?

In XGBoost we use a similarity instead of Gini and Entropy

$$Similarity = \frac{\text{Square of sum of labels of the set}}{\text{Number of points of the set} + \lambda} \qquad let\ \lambda = 0$$

Example: If we have three sets with these labels {10, -10, 4}, {7, 7, 7}, {7}

Set 1: $Similarity = \frac{(10-10+4)^2}{3} = \frac{16}{3}.$

Set 2: $Similarity = \frac{(7+7+7)^2}{3} = 147$

XGBoost

How to build the learner?

In XGBoost we use a similarity instead of Gini and Entropy

$$\text{Similarity} = \frac{\text{Square of sum of labels of the set}}{\text{Number of points of the set} + \lambda} \qquad \text{let } \lambda = 0$$

Example: If we have three sets with these labels {10, -10, 4}, {7, 7, 7}, {7}

Set 1: $Similarity = \frac{(10-10+4)^2}{3} = \frac{16}{3}$.

Set 2: $Similarity = \frac{(7+7+7)^2}{3} = 147$

Set 3: $Similarity = \frac{7^2}{1} = 49$

XGBoost

How to build the learner?

In XGBoost we use a similarity instead of Gini and Entropy

$$\text{Similarity} = \frac{\text{Square of sum of labels of the set}}{\text{Number of points of the set} + \lambda} \qquad \text{let } \lambda = 0$$

Example: If we have three sets with these labels {10, -10, 4}, {7, 7, 7}, {7}

Set 1: $Similarity = \frac{(10-10+4)^2}{3} = \frac{16}{3}$.

Set 2: $Similarity = \frac{(7+7+7)^2}{3} = 147$

Set 3: $Similarity = \frac{7^2}{1} = 49$

This is **not a perfect similarity metric** as it gives **high priority for high values**, but this is **good for our purpose** as we will see.

XGBoost

Age (feature)	Engagement (label)
10	7
20	5
30	6
40	0
50	1
60	0
70	4
80	3

XGBoost

Building the learners

Step1:
Start with a naive learner that predicts 0.5 in all cases

Age (feature)	Engagement (label)
10	7
20	5
30	6
40	0
50	1
60	0
70	4
80	3

XGBoost

Building the learners

Step1:
Start with a naive learner that predicts 0.5 in all cases

Age (feature)	Engagement (label)	Predicted label
10	7	0.5
20	5	0.5
30	6	0.5
40	0	0.5
50	1	0.5
60	0	0.5
70	4	0.5
80	3	0.5

XGBoost

Building the learners

Step1:
Start with a naive learner that predicts 0.5 in all cases

Step2:
Calculate residual between predicted and label

Age (feature)	Engagement (label)	Predicted label
10	7	0.5
20	5	0.5
30	6	0.5
40	0	0.5
50	1	0.5
60	0	0.5
70	4	0.5
80	3	0.5

XGBoost

Building the learners

Step1:
Start with a naive learner that predicts 0.5 in all cases

Step2:
Calculate residual between predicted and label

Age (feature)	Engagement (label)	Predicted label	Residual
10	7	0.5	6.5
20	5	0.5	4.5
30	6	0.5	5.5
40	0	0.5	-0.5
50	1	0.5	0.5
60	0	0.5	-0.5
70	4	0.5	3.5
80	3	0.5	2.5

XGBoost

Age (feature)	Residual
10	6.5
20	4.5
30	5.5
40	-0.5
50	0.5
60	-0.5
70	3.5
80	2.5

XGBoost

Building the learners

Step3:
Build a decision tree to predict the residual based on similarity gain

Base similarity:

Age (feature)	Residual
10	6.5
20	4.5
30	5.5
40	-0.5
50	0.5
60	-0.5
70	3.5
80	2.5

XGBoost

Building the learners

Step3:
Build a decision tree to predict the residual based on similarity gain

Base similarity:

Age (feature)	Residual
10	6.5
20	4.5
30	5.5
40	-0.5
50	0.5
60	-0.5
70	3.5
80	2.5

XGBoost

Building the learners

Step3:

Build a decision tree to predict the residual based on similarity gain

Base similarity: $\frac{(6.5 + 4.5 + 5.5 - 0.5 + 0.5 - 0.5 + 3.5 + 2.5)^2}{8} = 60.5$

Age (feature)	Residual
10	6.5
20	4.5
30	5.5
40	-0.5
50	0.5
60	-0.5
70	3.5
80	2.5

XGBoost

Building the learners

Step3:

Build a decision tree to predict the residual based on similarity gain

Base similarity: $\frac{(6.5 + 4.5 + 5.5 - 0.5 + 0.5 - 0.5 + 3.5 + 2.5)^2}{8} = 60.5$

Split at 15: Similarity Score = 76.57
Split at 25: Similarity Score = 80.67
Split at 35: Similarity Score = 96.8
Split at 45: Similarity Score = 73
Split at 55: Similarity Score = 64.53
Split at 65: Similarity Score = 60.67
Split at 75: Similarity Score = 60.57

Age (feature)	Residual
10	6.5
20	4.5
30	5.5
40	-0.5
50	0.5
60	-0.5
70	3.5
80	2.5

XGBoost

Building the learners

Step3:

Build a decision tree to predict the residual based on similarity gain

Base similarity: $\frac{(6.5 + 4.5 + 5.5 - 0.5 + 0.5 - 0.5 + 3.5 + 2.5)^2}{8} = 60.5$

Split at 15: Similarity Score = 76.57
Split at 25: Similarity Score = 80.67
Split at 35: Similarity Score = 96.8
Split at 45: Similarity Score = 73
Split at 55: Similarity Score = 64.53
Split at 65: Similarity Score = 60.67
Split at 75: Similarity Score = 60.57

Split at 15: Similarity Gain = 16.07
Split at 25: Similarity Gain = 20.16
Split at 35: Similarity Gain = 36.3
Split at 45: Similarity Gain = 12.5
Split at 55: Similarity Gain = 4.03
Split at 65: Similarity Gain = 0.17
Split at 75: Similarity Gain = 0.07

Age (feature)	Residual
10	6.5
20	4.5
30	5.5
40	-0.5
50	0.5
60	-0.5
70	3.5
80	2.5

XGBoost

Age (feature)	Residual
10	6.5
20	4.5
30	5.5
40	-0.5
50	0.5
60	-0.5
70	3.5
80	2.5

XGBoost

Building the learners

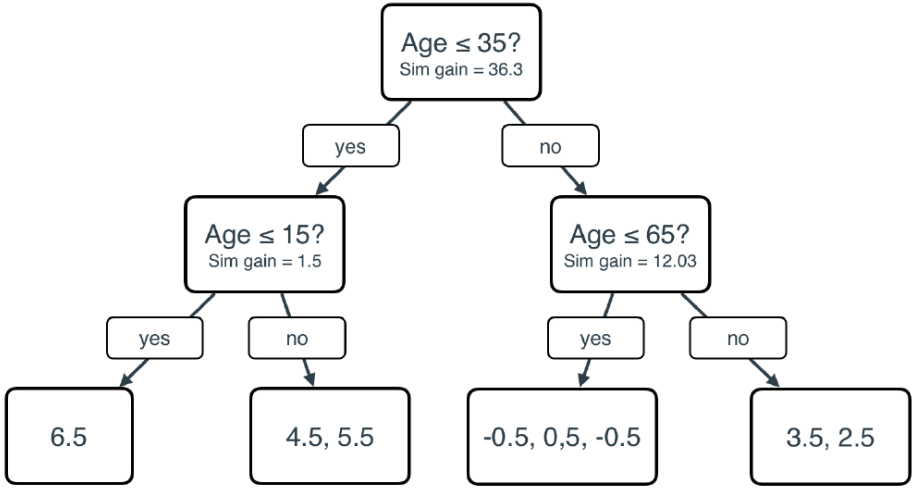
Step3:
Build a decision tree to predict the residual based on similarity gain

Age (feature)	Residual
10	6.5
20	4.5
30	5.5
40	-0.5
50	0.5
60	-0.5
70	3.5
80	2.5

XGBoost

Building the learners

Step3:
Build a decision tree to predict the residual based on similarity gain



Age (feature)	Residual
10	6.5
20	4.5
30	5.5
40	-0.5
50	0.5
60	-0.5
70	3.5
80	2.5

XGBoost

XGBoost

Hyperparameters:

Maximum Tree Depth: If increased, Overfitting increased.

Number Of Trees: Number of learners (estimators)

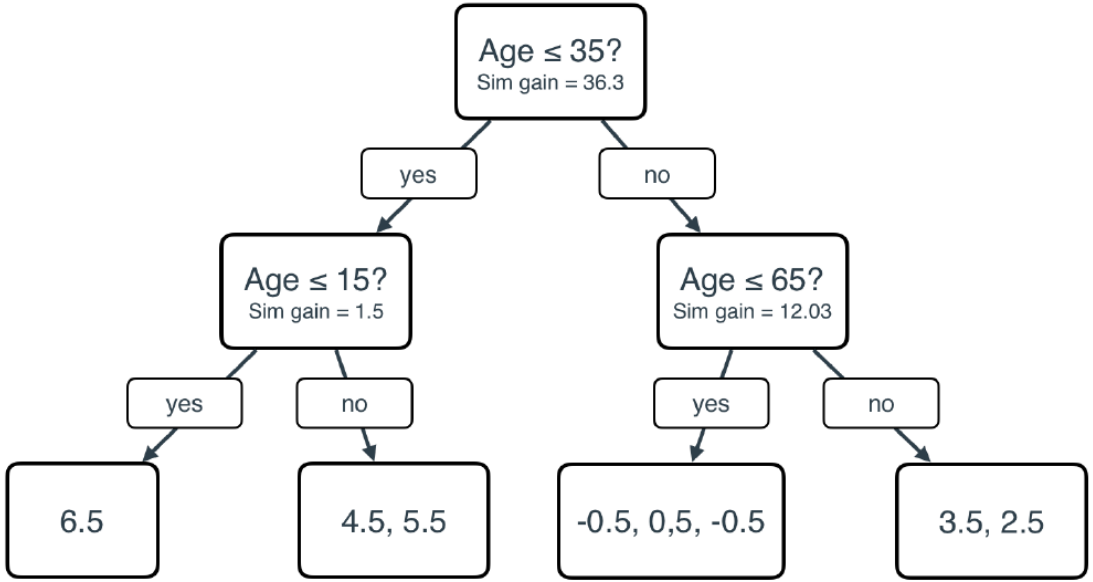
Minimum Number Of Elements Per Leaf: If increased, Overfitting decreased.

Minimum Similarity Gain (gamma γ): Minimum gain to split. If increased, Overfitting decreased.

Lambda λ: If increased, similarity gain decrease, overfitting decreased.

Learning Rate (eta η): The percentage of prediction that we take to not take the full step of decreasing.

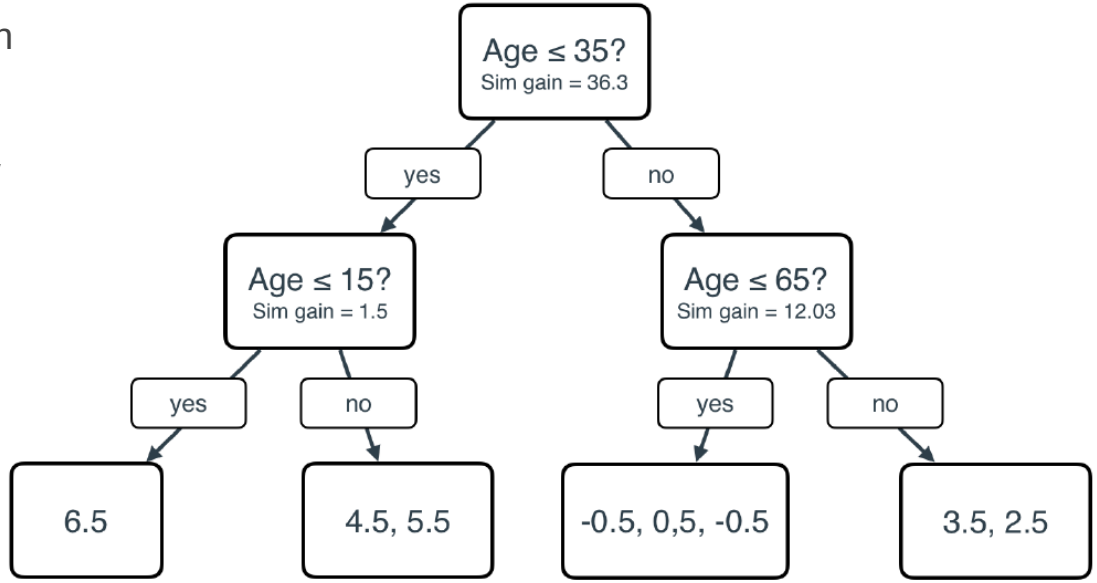
XGBoost


Age ≤ 35?
Sim gain = 36.3
yes
no
Age ≤ 15?
Sim gain = 1.5
Age ≤ 65?
Sim gain = 12.03
yes
no
yes
no
6.5
4.5, 5.5
-0.5, 0,5, -0.5
3.5, 2.5

XGBoost

Tree Pruning:
Stop splitting or remove leaves based on the above hyperparameters.

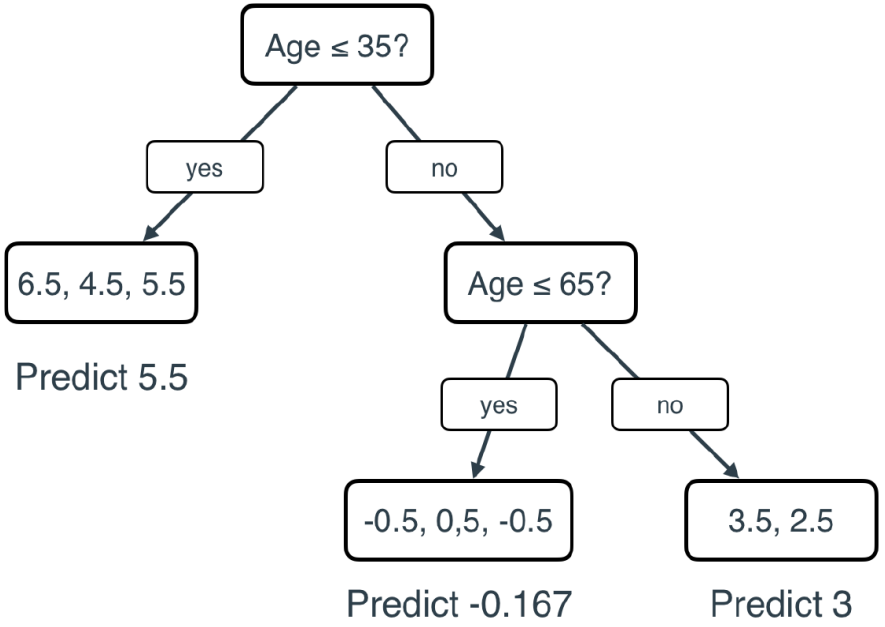
For example we set “Minimum Similarity Gain (gamma γ)” to 5



XGBoost

Tree Pruning:
Stop splitting or remove leaves based on the above hyperparameters.

For example we set “Minimum Similarity Gain (gamma γ)” to 5



XGBoost

XGBoost

Steps till now:

- Start with a naive learner that predicts 0.5 in all cases
- Calculate residual of learner 1
- Build learner 2 to fit the residual
- Multiply the predictions of learner 2 by learning rate (η), say 0.7
- Calculate the new predictions (learner 1 preds + learner 2 preds) Why?
- Calculate the new resuals

XGBoost

Steps till now:

- Start with a naive learner that predicts 0.5 in all cases
- Calculate residual of learner 1
- Build learner 2 to fit the residual
- Multiply the predictions of learner 2 by learning rate (η), say 0.7
- Calculate the new predictions (learner 1 preds + learner 2 preds) Why?
- Calculate the new resuals

Age (feature)	Engagement (label)
10	7
20	5
30	6
40	0
50	1
60	0
70	4
80	3

XGBoost

Steps till now:

- Start with a naive learner that predicts 0.5 in all cases
- Calculate residual of learner 1
- Build learner 2 to fit the residual
- Multiply the predictions of learner 2 by learning rate (η), say 0.7
- Calculate the new predictions (learner 1 preds + learner 2 preds) Why?
- Calculate the new resuals

Age (feature)	Engagement (label)	Prediction 1 (First tree)
10	7	0.5
20	5	0.5
30	6	0.5
40	0	0.5
50	1	0.5
60	0	0.5
70	4	0.5
80	3	0.5

XGBoost

Steps till now:

- Start with a naive learner that predicts 0.5 in all cases
- Calculate residual of learner 1
- Build learner 2 to fit the residual
- Multiply the predictions of learner 2 by learning rate (η), say 0.7
- Calculate the new predictions (learner 1 preds + learner 2 preds) Why?
- Calculate the new resuals

Age (feature)	Engagement (label)	Prediction 1 (First tree)	First residual
10	7	0.5	6.5
20	5	0.5	4.5
30	6	0.5	5.5
40	0	0.5	-0.5
50	1	0.5	0.5
60	0	0.5	-0.5
70	4	0.5	3.5
80	3	0.5	2.5

XGBoost

Steps till now:

- Start with a naive learner that predicts 0.5 in all cases
- Calculate residual of learner 1
- Build learner 2 to fit the residual
- Multiply the predictions of learner 2 by learning rate (η), say 0.7
- Calculate the new predictions (learner 1 preds + learner 2 preds) Why?
- Calculate the new resuals

Age (feature)	Engagement (label)	Prediction 1 (First tree)	First residual	Prediction 2 (Second tree)
10	7	0.5	6.5	5.5
20	5	0.5	4.5	5.5
30	6	0.5	5.5	5.5
40	0	0.5	-0.5	-0.167
50	1	0.5	0.5	-0.167
60	0	0.5	-0.5	-0.167
70	4	0.5	3.5	3
80	3	0.5	2.5	3

XGBoost

Steps till now:

- Start with a naive learner that predicts 0.5 in all cases
- Calculate residual of learner 1
- Build learner 2 to fit the residual
- Multiply the predictions of learner 2 by learning rate (η), say 0.7
- Calculate the new predictions (learner 1 preds + learner 2 preds) Why?
- Calculate the new resuals

Age (feature)	Engagement (label)	Prediction 1 (First tree)	First residual	Prediction 2 (Second tree)	Prediction 2 times learning rate
10	7	0.5	6.5	5.5	4.4
20	5	0.5	4.5	5.5	4.4
30	6	0.5	5.5	5.5	4.4
40	0	0.5	-0.5	-0.167	-0.134
50	1	0.5	0.5	-0.167	-0.134
60	0	0.5	-0.5	-0.167	-0.134
70	4	0.5	3.5	3	2.4
80	3	0.5	2.5	3	2.4

XGBoost

Steps till now:

- Start with a naive learner that predicts 0.5 in all cases
- Calculate residual of learner 1
- Build learner 2 to fit the residual
- Multiply the predictions of learner 2 by learning rate (η), say 0.7
- Calculate the new predictions (learner 1 preds + learner 2 preds) Why?
- Calculate the new resuals

Age (feature)	Engagement (label)	Prediction 1 (First tree)	First residual	Prediction 2 (Second tree)	Prediction 2 times learning rate	Total prediction (pred 1 + pred 2)
10	7	0.5	6.5	5.5	4.4	4.9
20	5	0.5	4.5	5.5	4.4	4.9
30	6	0.5	5.5	5.5	4.4	4.9
40	0	0.5	-0.5	-0.167	-0.134	0.366
50	1	0.5	0.5	-0.167	-0.134	0.366
60	0	0.5	-0.5	-0.167	-0.134	0.366
70	4	0.5	3.5	3	2.4	2.9
80	3	0.5	2.5	3	2.4	2.9

XGBoost

Steps till now:

- Start with a naive learner that predicts 0.5 in all cases
- Calculate residual of learner 1
- Build learner 2 to fit the residual
- Multiply the predictions of learner 2 by learning rate (η), say 0.7
- Calculate the new predictions (learner 1 preds + learner 2 preds) Why?
- Calculate the new resuals

Age (feature)	Engagement (label)	Prediction 1 (First tree)	First residual	Prediction 2 (Second tree)	Prediction 2 times learning rate	Total prediction (pred 1 + pred 2)	Second residual
10	7	0.5	6.5	5.5	4.4	4.9	2.1
20	5	0.5	4.5	5.5	4.4	4.9	0.1
30	6	0.5	5.5	5.5	4.4	4.9	1.1
40	0	0.5	-0.5	-0.167	-0.134	0.366	-0.366
50	1	0.5	0.5	-0.167	-0.134	0.366	0.636
60	0	0.5	-0.5	-0.167	-0.134	0.366	-0.366
70	4	0.5	3.5	3	2.4	2.9	1.1
80	3	0.5	2.5	3	2.4	2.9	0.1

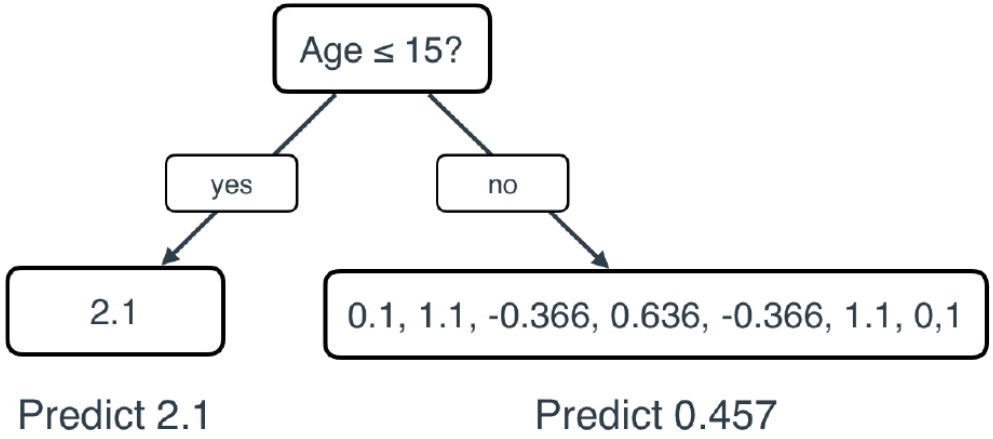
XGBoost

Time to build learner 3 with the same way

Age (feature)	Second residual
10	2.1
20	0.1
30	1.1
40	-0.366
50	0.636
60	-0.366
70	1.1
80	0.1

XGBoost

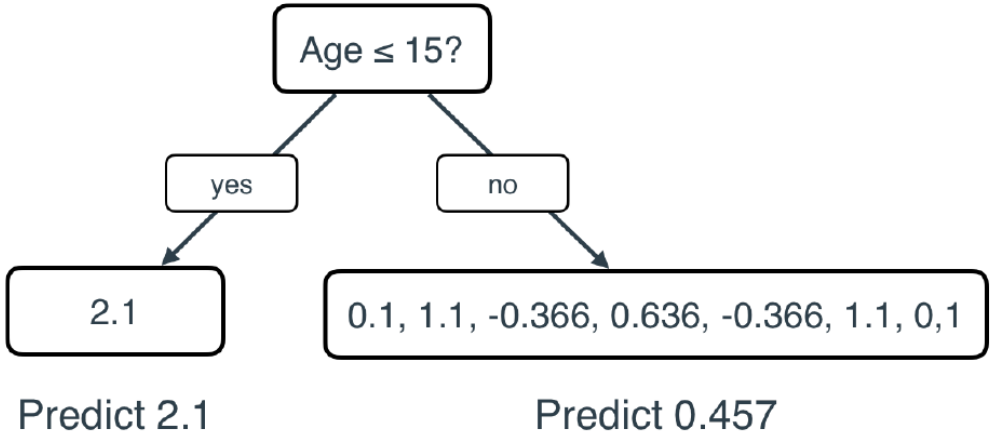
Time to build learner 3 with the same way



Age (feature)	Second residual
10	2.1
20	0.1
30	1.1
40	-0.366
50	0.636
60	-0.366
70	1.1
80	0.1

XGBoost

Time to build learner 3 with the same way



Age (feature)	Second residual
10	2.1
20	0.1
30	1.1
40	-0.366
50	0.636
60	-0.366
70	1.1
80	0.1

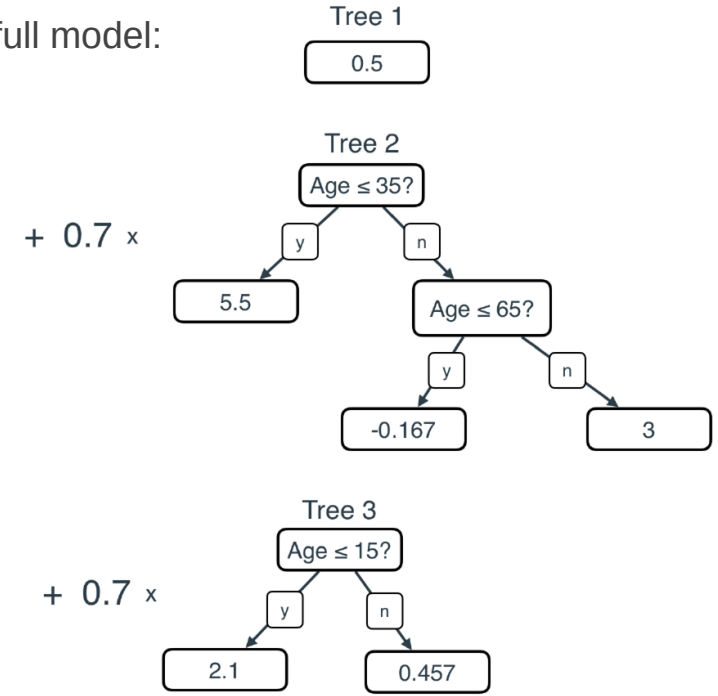
To get the predictions from learner 3, we multiply the output by the learning rate (0.7)

XGBoost

The full model:

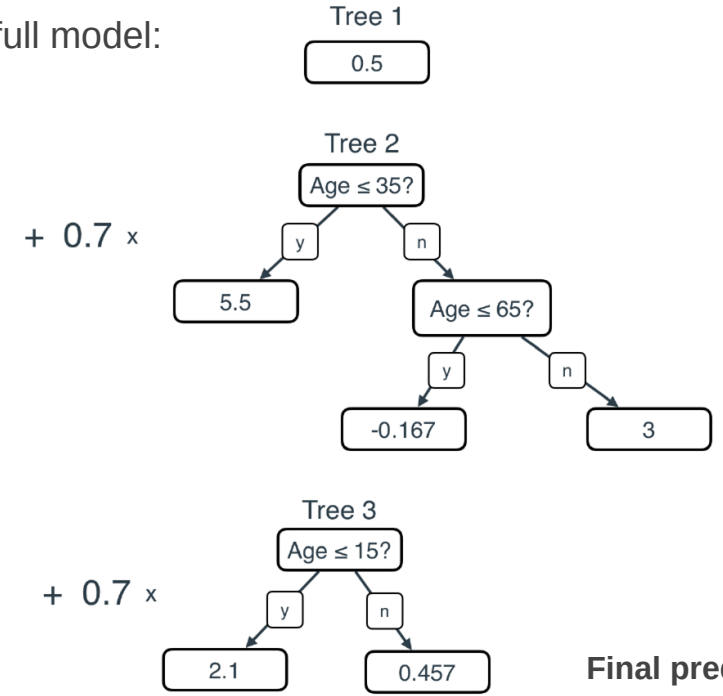
XGBoost

The full model:



XGBoost

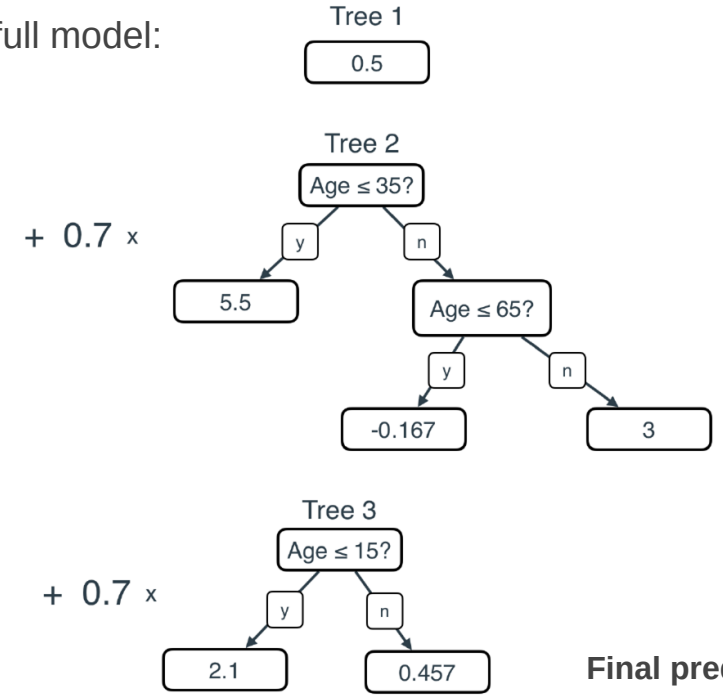
The full model:



Final prediction = Prediction 1 + 0.7 * Prediction 2 + 0.7 * Prediction 3

XGBoost

The full model:



Age (feature)	Engagement (label)	Prediction 1 (First tree)	Prediction 2 (Second tree)	Prediction 3 (Third tree)	Final prediction
10	7	0.5	5.5	2.1	5.82
20	5	0.5	5.5	0.457	4.67
30	6	0.5	5.5	0.457	4.67
40	0	0.5	-0.167	0.457	0.7
50	1	0.5	-0.167	0.457	0.7
60	0	0.5	-0.167	0.457	0.7
70	4	0.5	3	0.457	2.9
80	3	0.5	3	0.457	2.9

Final prediction = Prediction 1 + 0.7 * Prediction 2 + 0.7 * Prediction 3

Coding XGBoost

Use colab to open this github notebook:

“s7s/machine_learning_1/ensemble_methods/XGBoost.ipynb”