Machine Learning

Lesson 8: Ensemble Learning

© Simplilearn. All rights reserved.

Concepts Covered

- ✓ Ensemble Learning
- ✓ Bagging and Boosting Algorithms
- ✓ Model Selection
- ✓ Cross-validation

©Simplilearn. All rights reserved.

Learning Objectives

By the end of this lesson, you will be able to:

- ✓ Explain ensemble learning
- ✓ Evaluate performance of boosting models

©Simplilearn. All rights reserved.

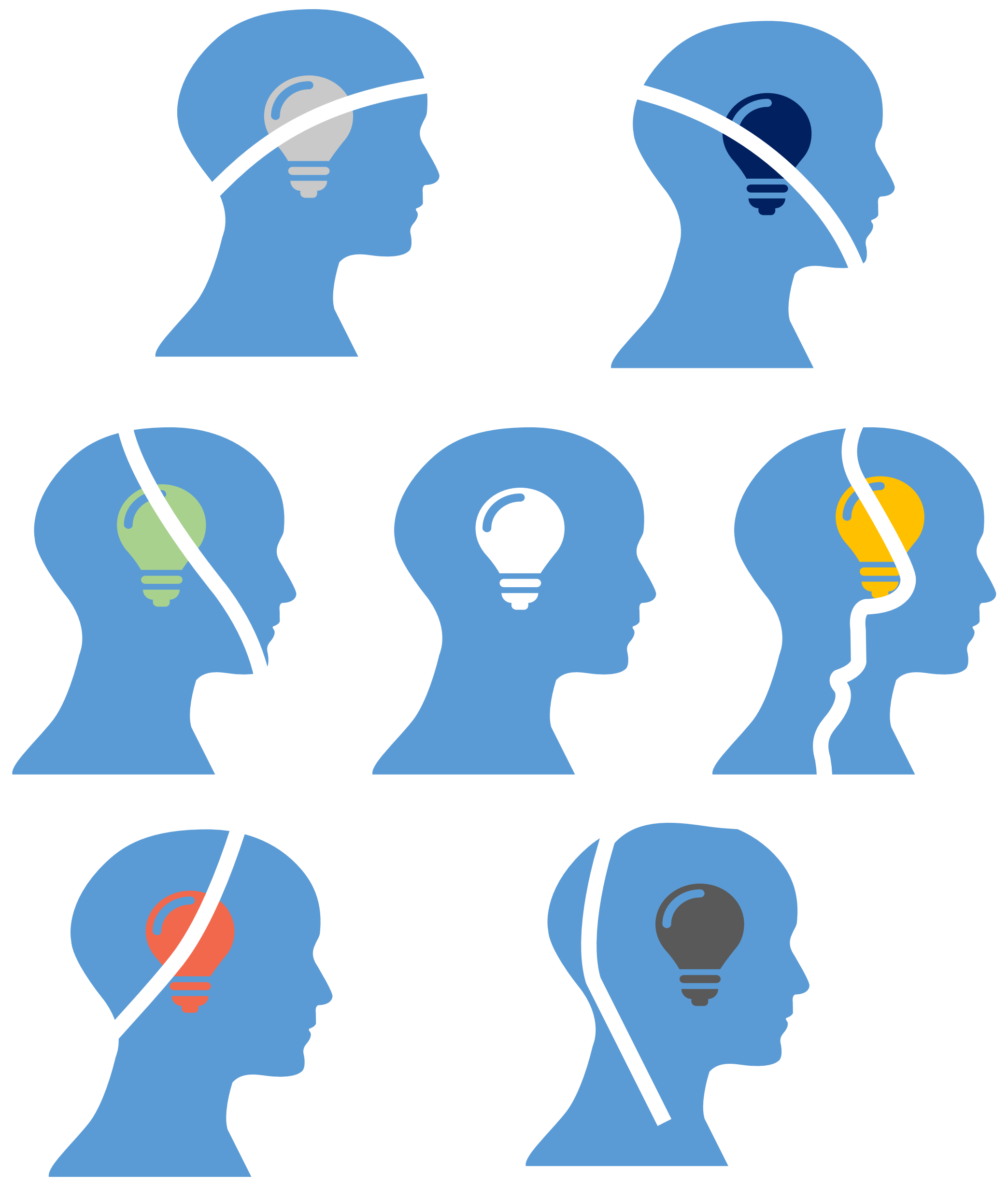
Ensemble Learning

Topic 1: Overview

©Simplilearn. All rights reserved.

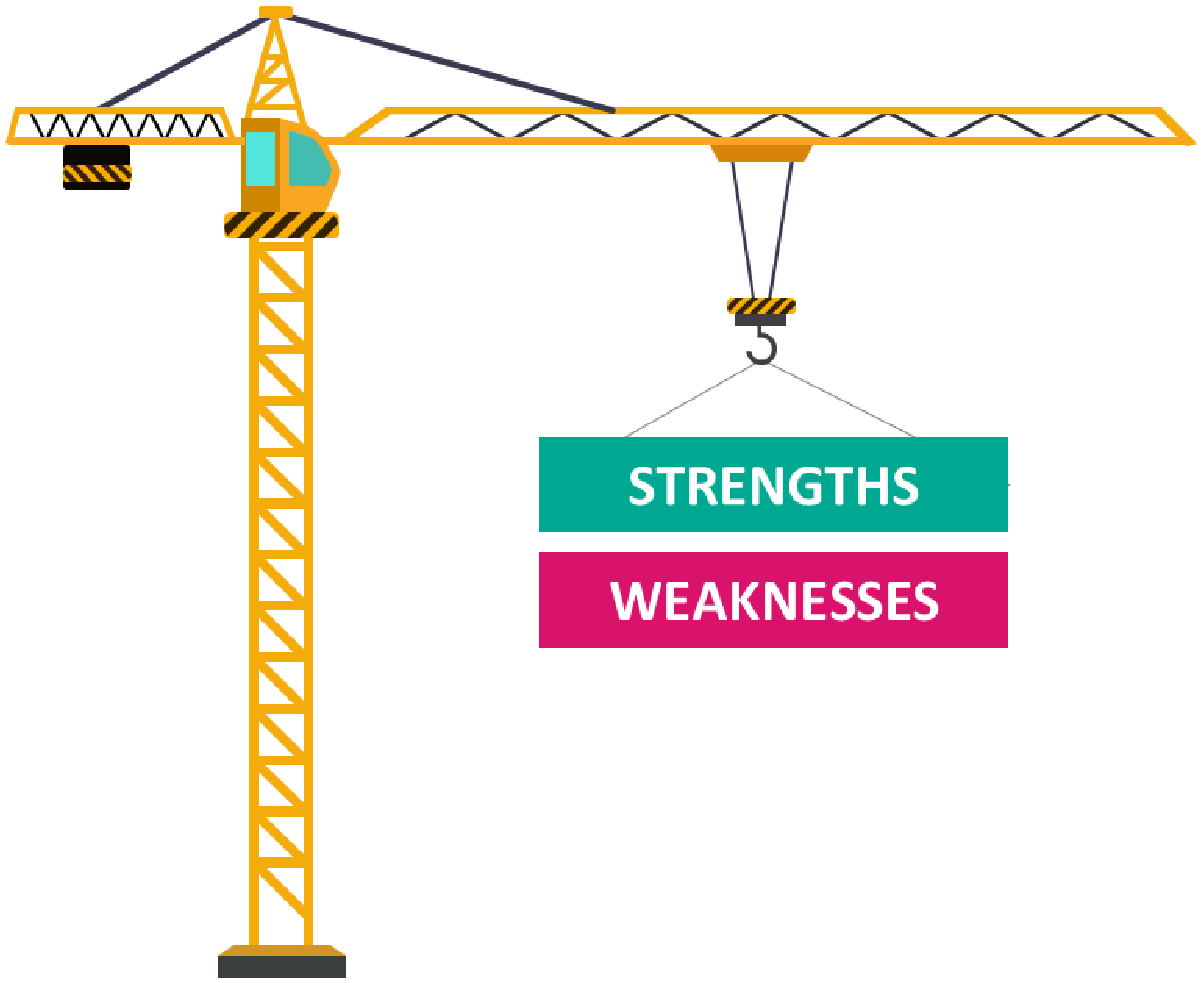
Definition

Ensemble techniques combine individual models together to improve the stability and predictive power of the model



©Simplilearn. All rights reserved.

Ideology



Certain models do well in modeling one aspect of the data, while others do well in modeling another

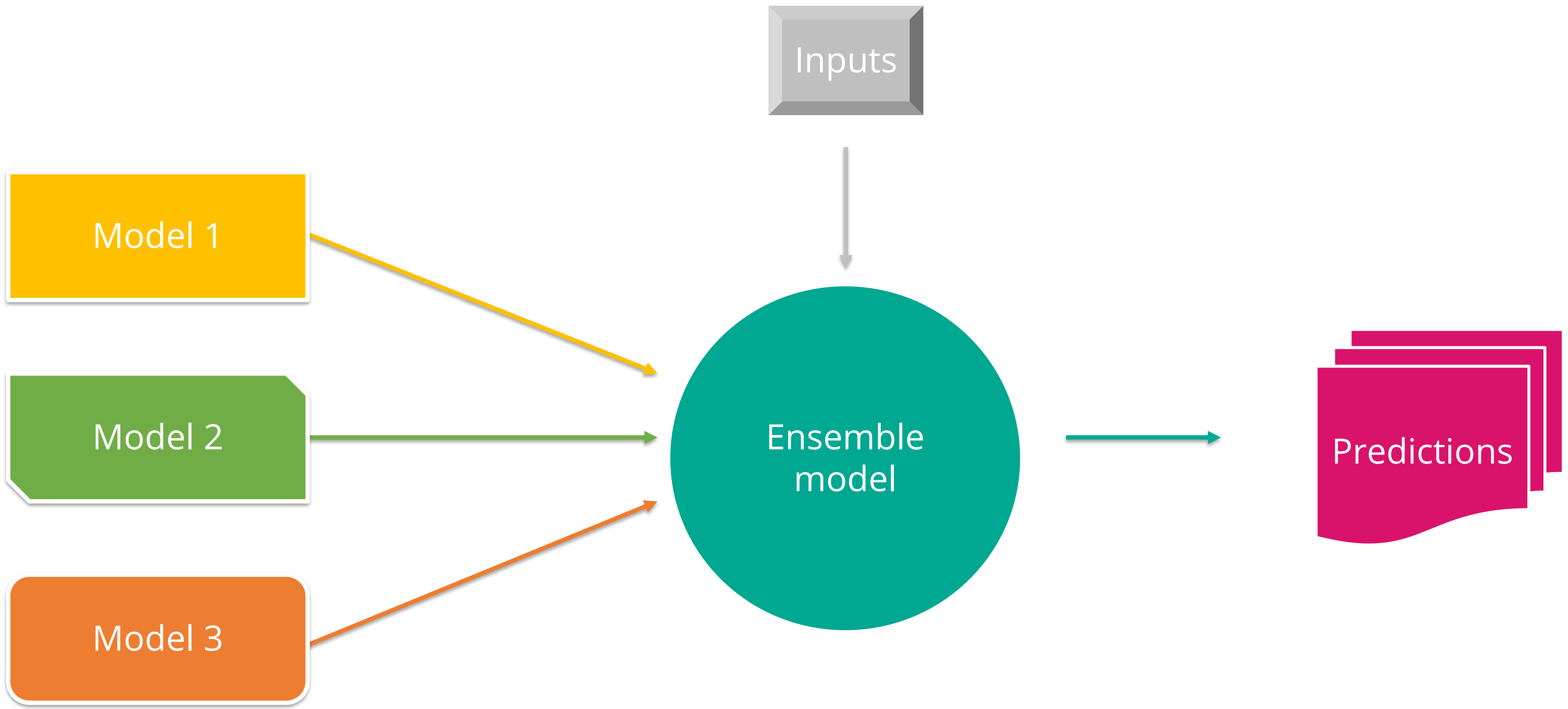
Instead of learning a single complex model, learn several simple models and combine their output to produce the final decision

In ensemble learning, other models strength performs offset on individual model variances and biases

Ensemble learning will provide a composite prediction where the final accuracy is better than the accuracy of individual models

©Simplilearn. All rights reserved.

Working



©Simplilearn. All rights reserved.

Significance

Robustness

Ensemble models incorporate the predictions from all the base learners

01

Accuracy

Ensemble models deliver accurate predictions and have improved performances

02

©Simplilearn. All rights reserved.

Ensemble Learning Methods

Techniques to create an Ensemble model

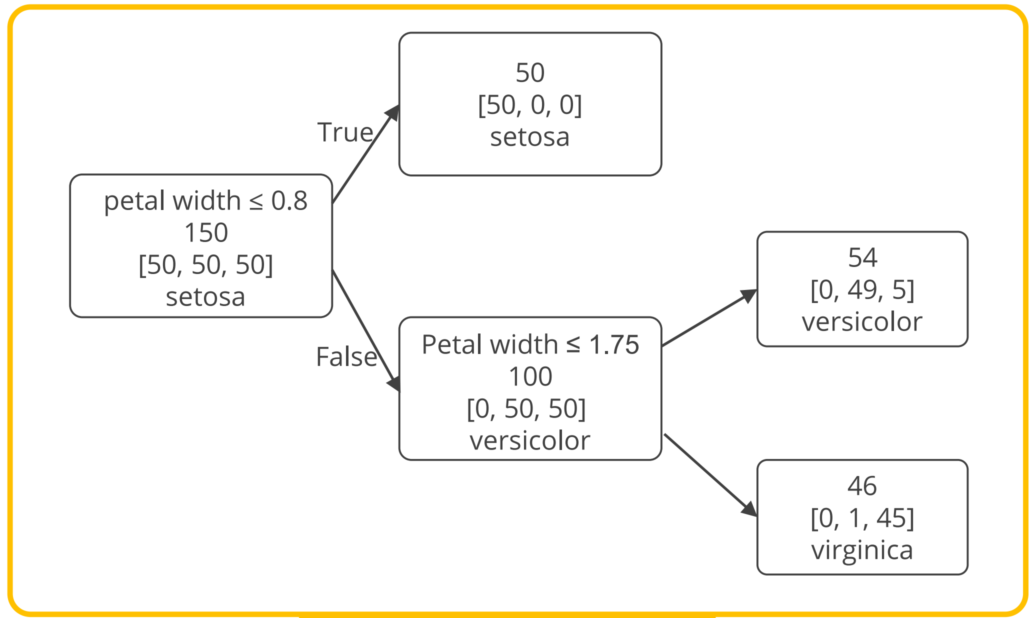
Combine all “weak” learners to form an ensemble

OR

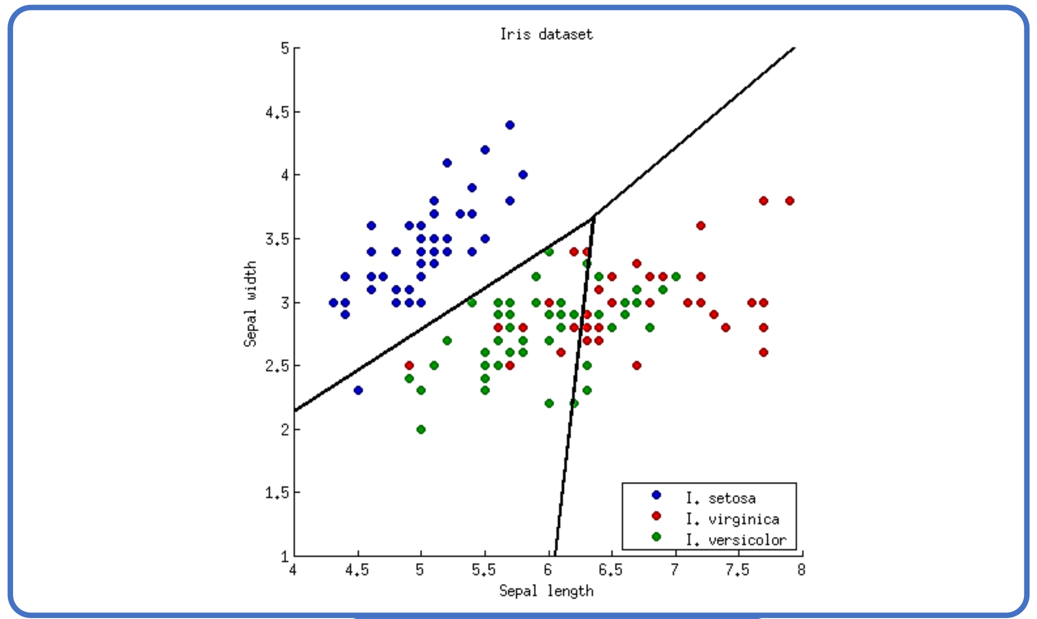
Create an ensemble of well-chosen strong and diverse models

©Simplilearn. All rights reserved.

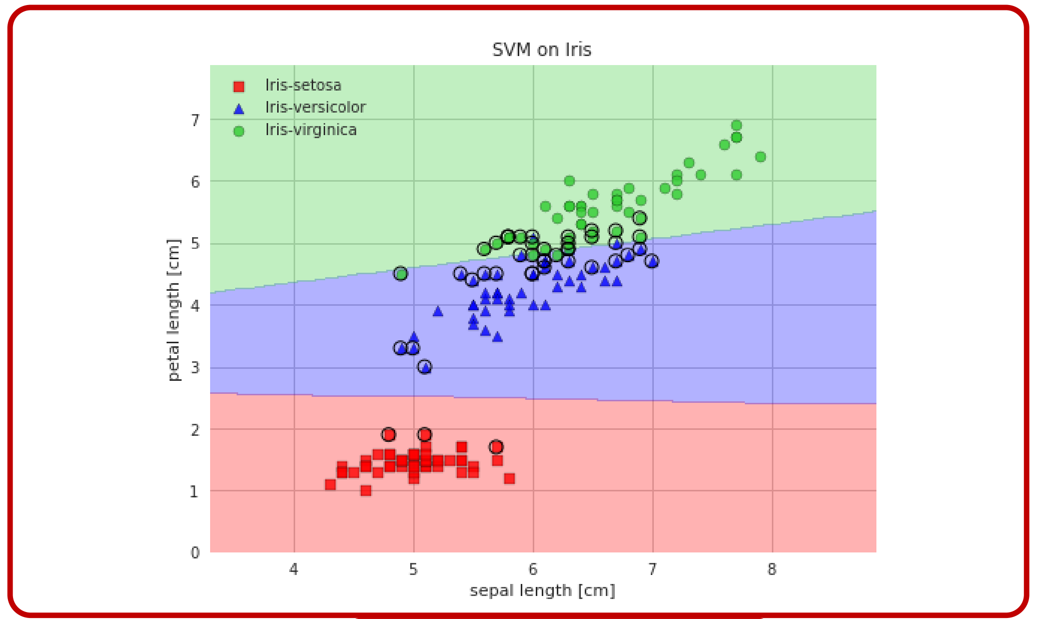
Averaging



Decision Tree



Logistic Regression



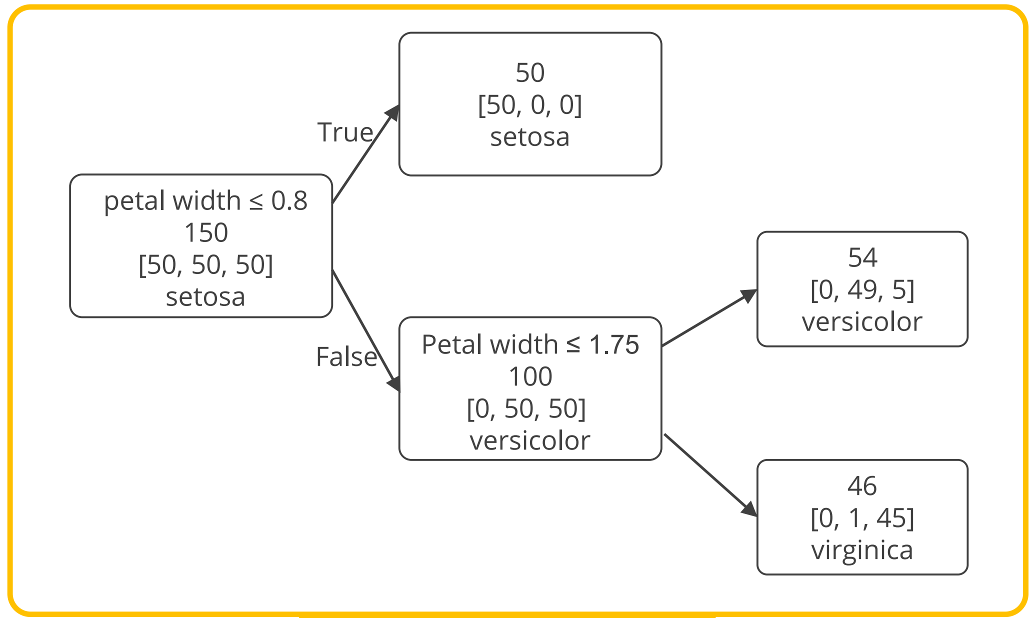
SVM

Equal weights are assigned to different models

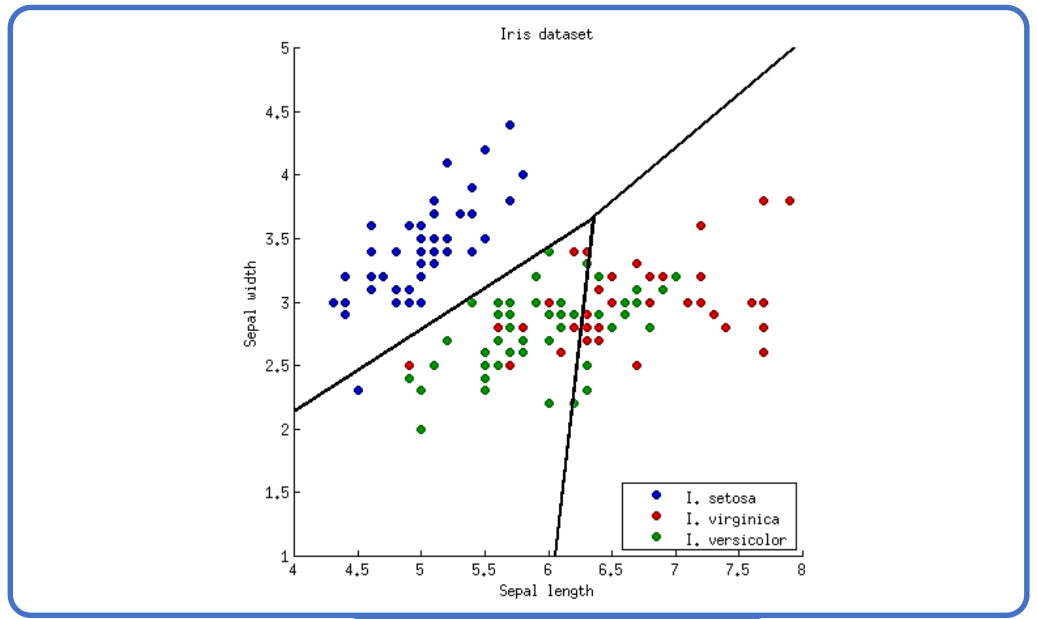
$$P = \frac{p1 + p2 + p3}{3}$$

©Simplilearn. All rights reserved.

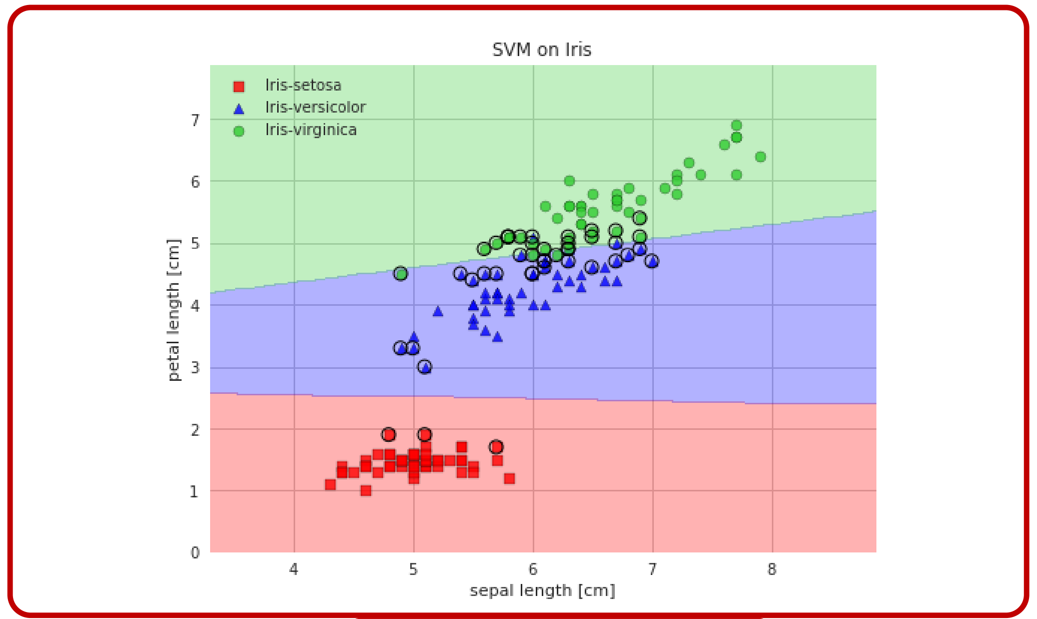
Weighted Averaging



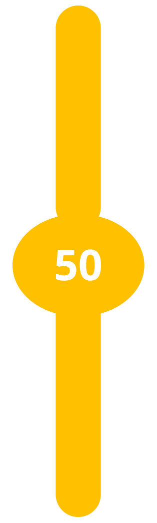
Decision Tree

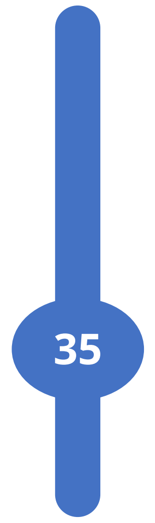


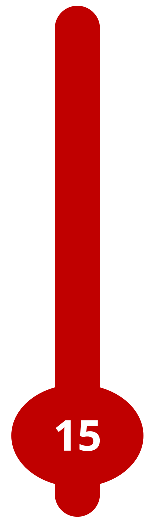
Logistic Regression



SVM







$w1 + w2 + w3 = 1$

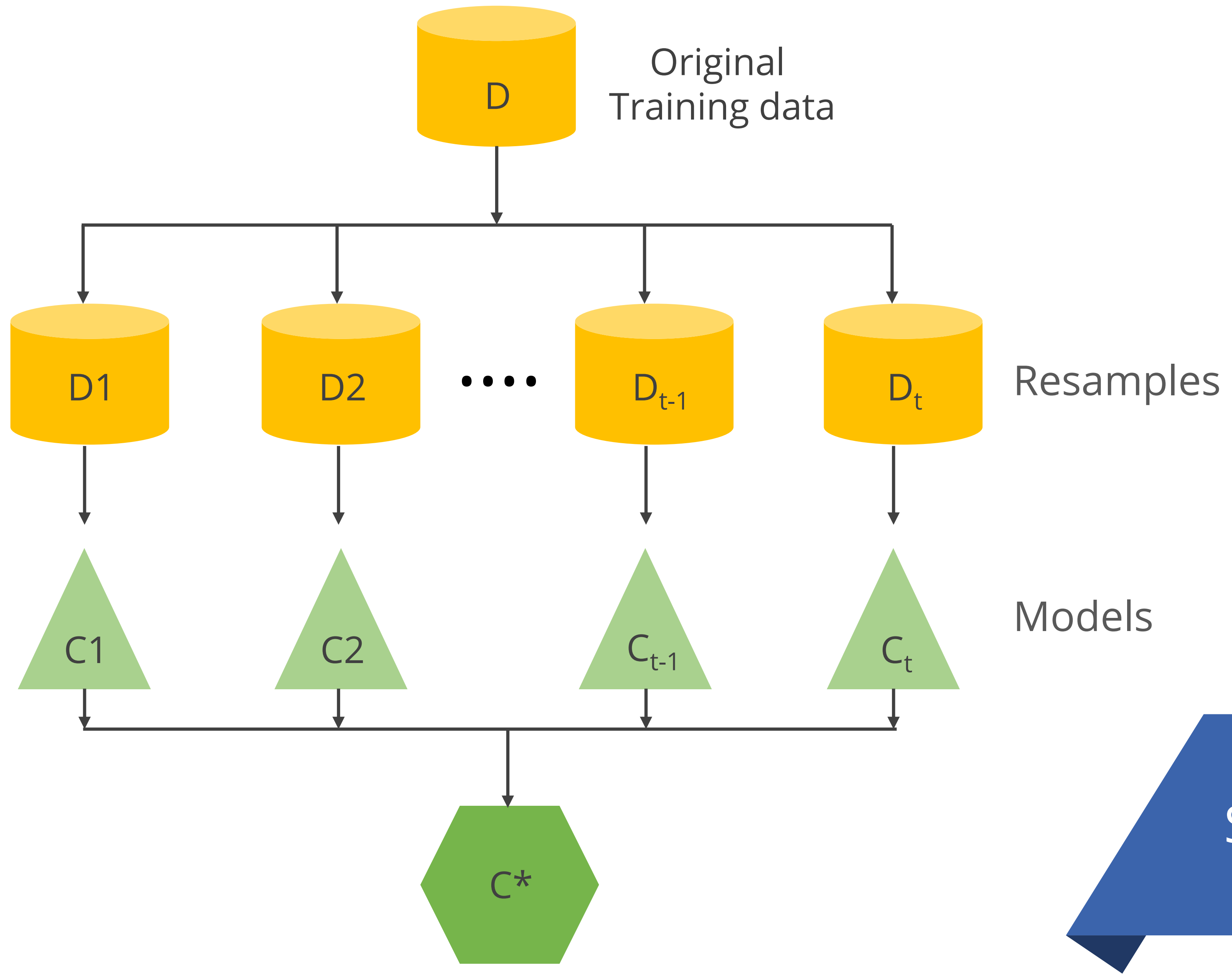
Each model is assigned a different weight

$P = w1 * p1 + w2 * p2 + w3 * p3$

©Simplilearn. All rights reserved.

Bagging

Bagging or bootstrap aggregation **reduces variance** of an estimate by taking mean of multiple estimates



STEP 1 Create randomly sampled datasets of the original training data (bootstrapping)

STEP 2 Build and fit several classifiers to each of these diverse copies

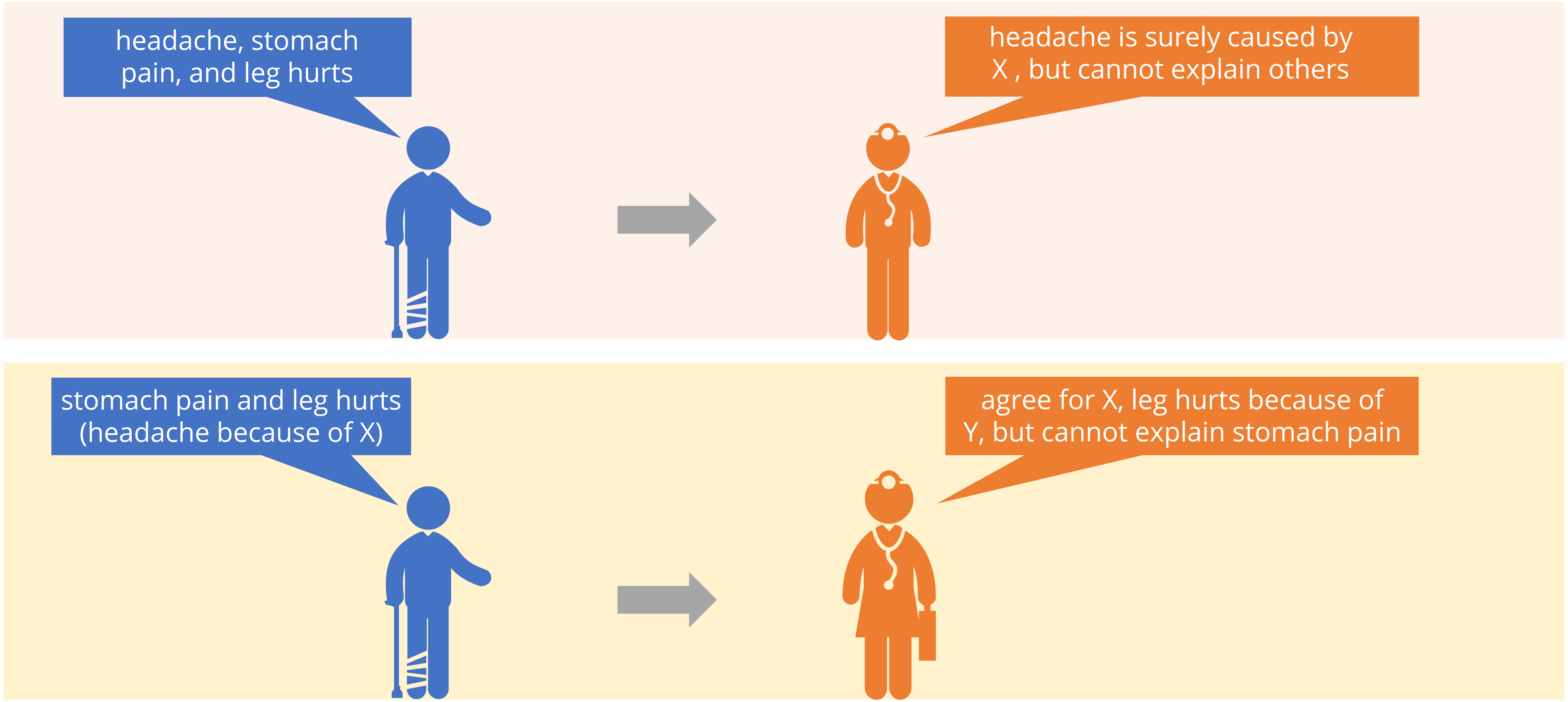
STEP 3 Take the average of all the predictions to make final overall prediction

$$f(x) = 1/M \sum_{m=1}^{M} f_m(x)$$

©Simplilearn. All rights reserved.

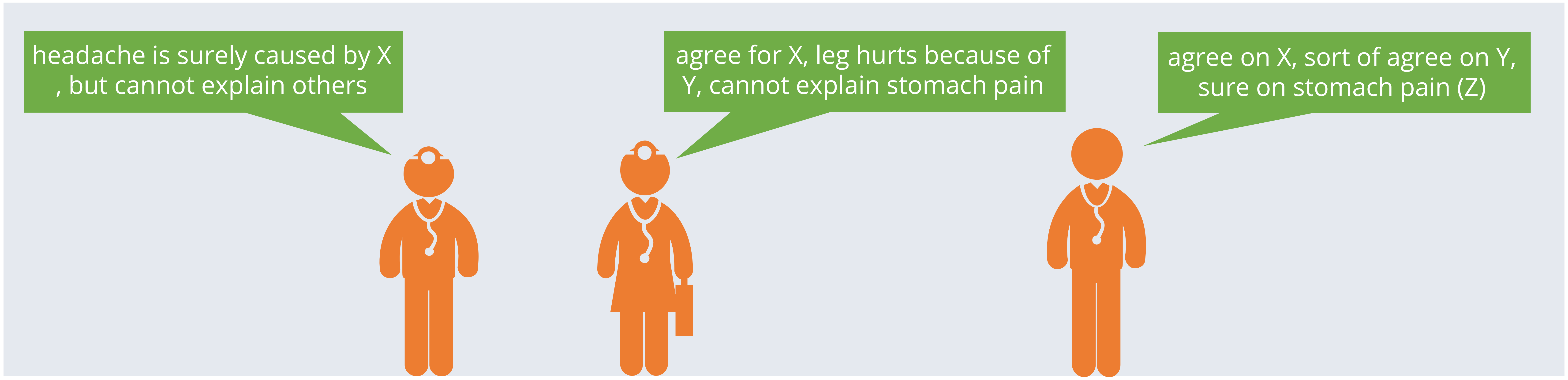
Boosting

Boosting **reduces bias** by training weak learners sequentially, each trying to correct its predecessor


headache, stomach pain, and leg hurts
headache is surely caused by X , but cannot explain others
stomach pain and leg hurts (headache because of X)
agree for X, leg hurts because of Y, but cannot explain stomach pain

©Simplilearn. All rights reserved.

Boosting (Contd.)


headache is surely caused by X , but cannot explain others
agree for X, leg hurts because of Y, cannot explain stomach pain
agree on X, sort of agree on Y, sure on stomach pain (Z)

Final Diagnosis

Result of the weighted opinions that the doctors (sequentially)

©Simplilearn. All rights reserved.

Boosting Algorithm

STEP: 01	STEP: 02	STEP: 03
Train a classifier H1 that best classifies the data with respect to accuracy	Identify the region where H1 produces errors, add weights to it and produce a H2 classifier	Exaggerate those samples for which H1 gives a different result from H2 and produces H3 classifier Repeat step 02 for a new classifier

©Simplilearn. All rights reserved.

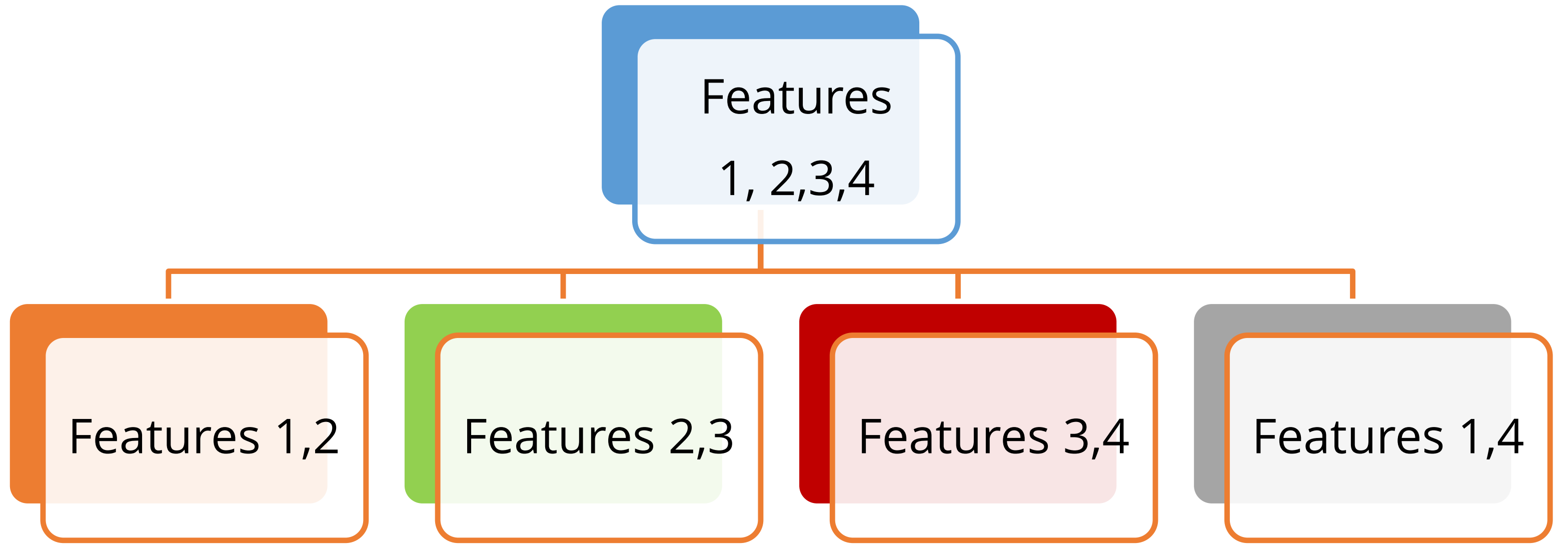
Ensemble Learning

Topic 2: Algorithms

©Simplilearn. All rights reserved.

Random Forests

Random forests are utilized to produce decorrelated decision trees



RF's create random subsets of the *features*

Smaller trees are built using these subsets creating tree diversity

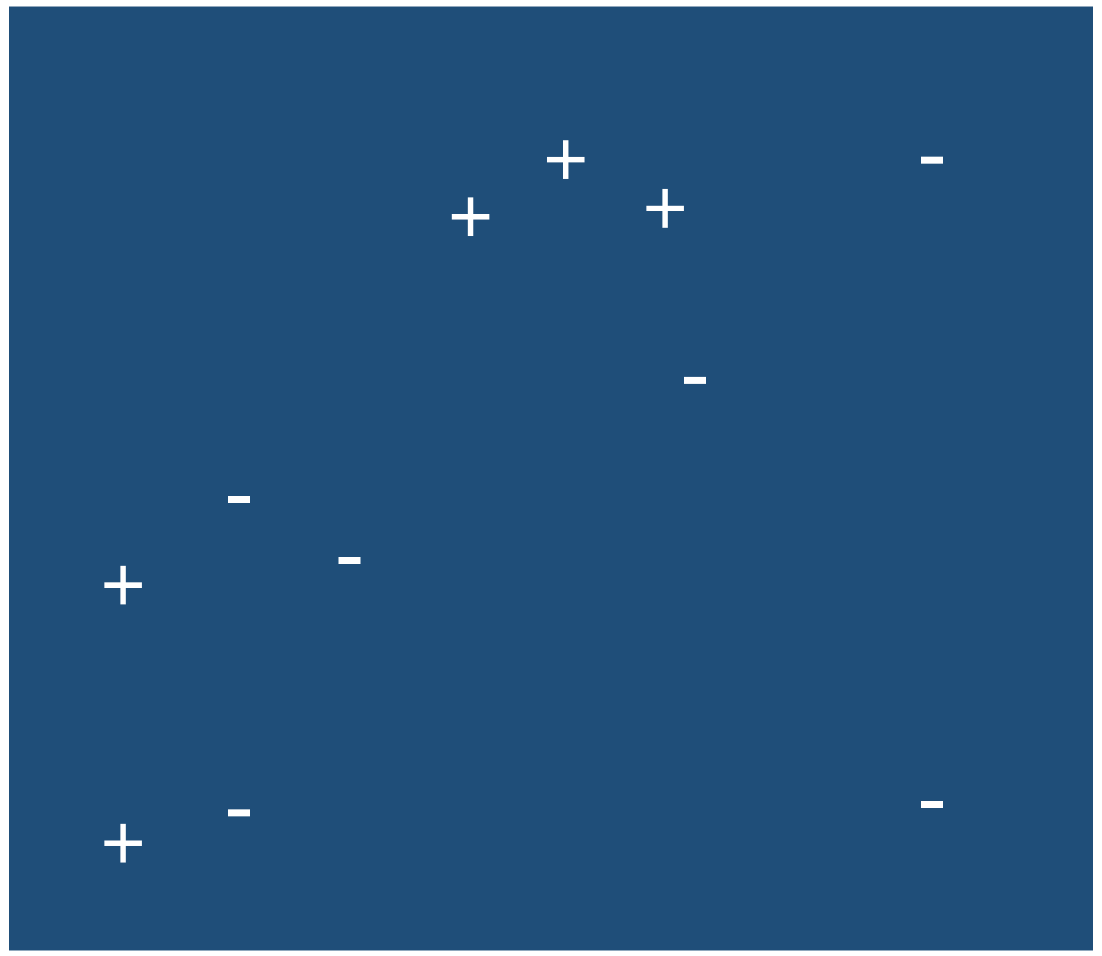
To overcome overfitting, diverse sets of decision trees are required

©Simplilearn. All rights reserved.

Adaboost

Consider a scenario, where there are '+' and '–'

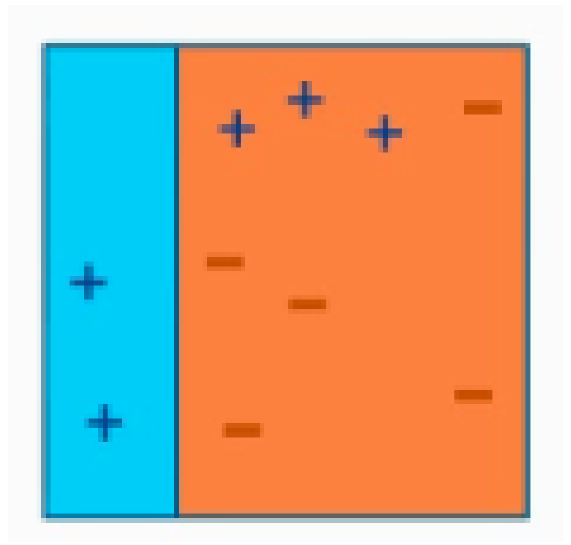
Objective : Classify '+' and '–'



©Simplilearn. All rights reserved.

Adaboost Working: Step 01

- Assign equal weights to each data point
- Apply a decision stump to classify them as + (plus) and – (minus)
- Decision stump (D1) has generated vertical plane at the left side to classify
- Apply higher weights to incorrectly predicted three + (plus) and add another decision stump

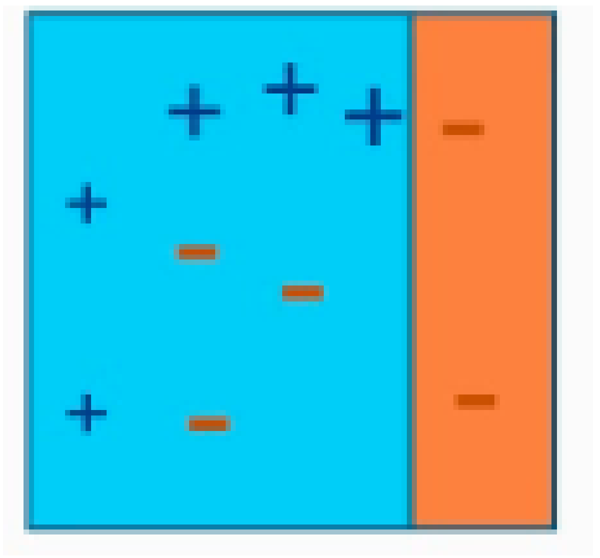


Iteration 01

©Simplilearn. All rights reserved.

Adaboost Working: Step 02

- Size of three incorrectly predicted + (plus) is made bigger as compared to rest of the data points
- The second decision stump (D2) will try to predict them correctly
- Now, vertical plane (D2) has classified three mis-classified + (plus) correctly
- D2 has also caused mis-classification errors to three – (minus)

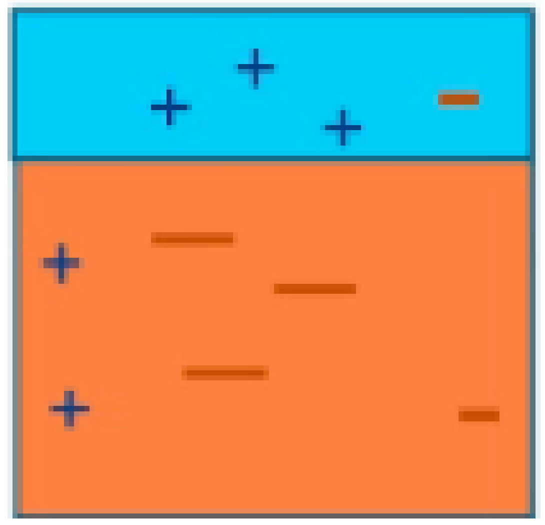


Iteration 02

©Simplilearn. All rights reserved.

Adaboost Working: Step 03

- D3 adds higher weights to three – (minus)
- Horizontal line is generated to classify + (plus) and – (minus) based on higher weight of mis-classified observation

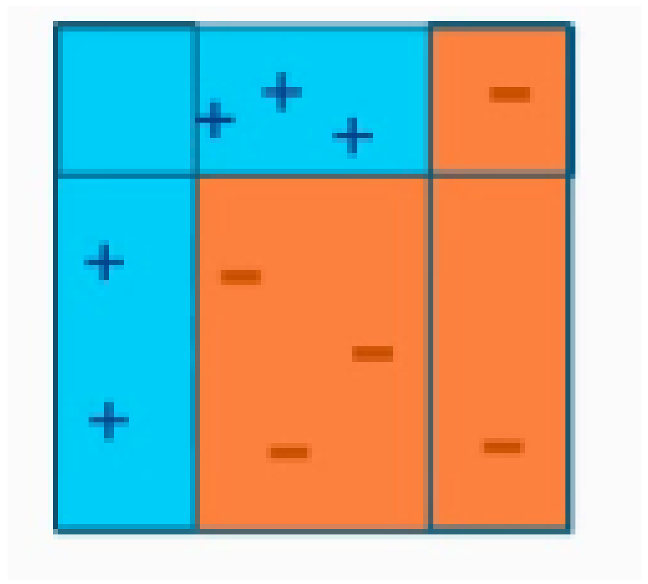


Iteration 03

©Simplilearn. All rights reserved.

Adaboost Working: Step 04

- D1, D2, and D3 are combined to form a strong prediction having complex rule as compared to individual weak learner



Final Classifier

©Simplilearn. All rights reserved.

Adaboost Algorithm

STEP 1

Initially each data point is weighted equally with weight

$Wi = 1/n$

where n is the number of samples

STEP 2

A classifier 'H1' is picked up that best classifies the data with minimal error rate

STEP 3

The weighing factor α is dependent on errors (ϵ_t) caused by the H1 classifier

$$\alpha^t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$$

STEP 4

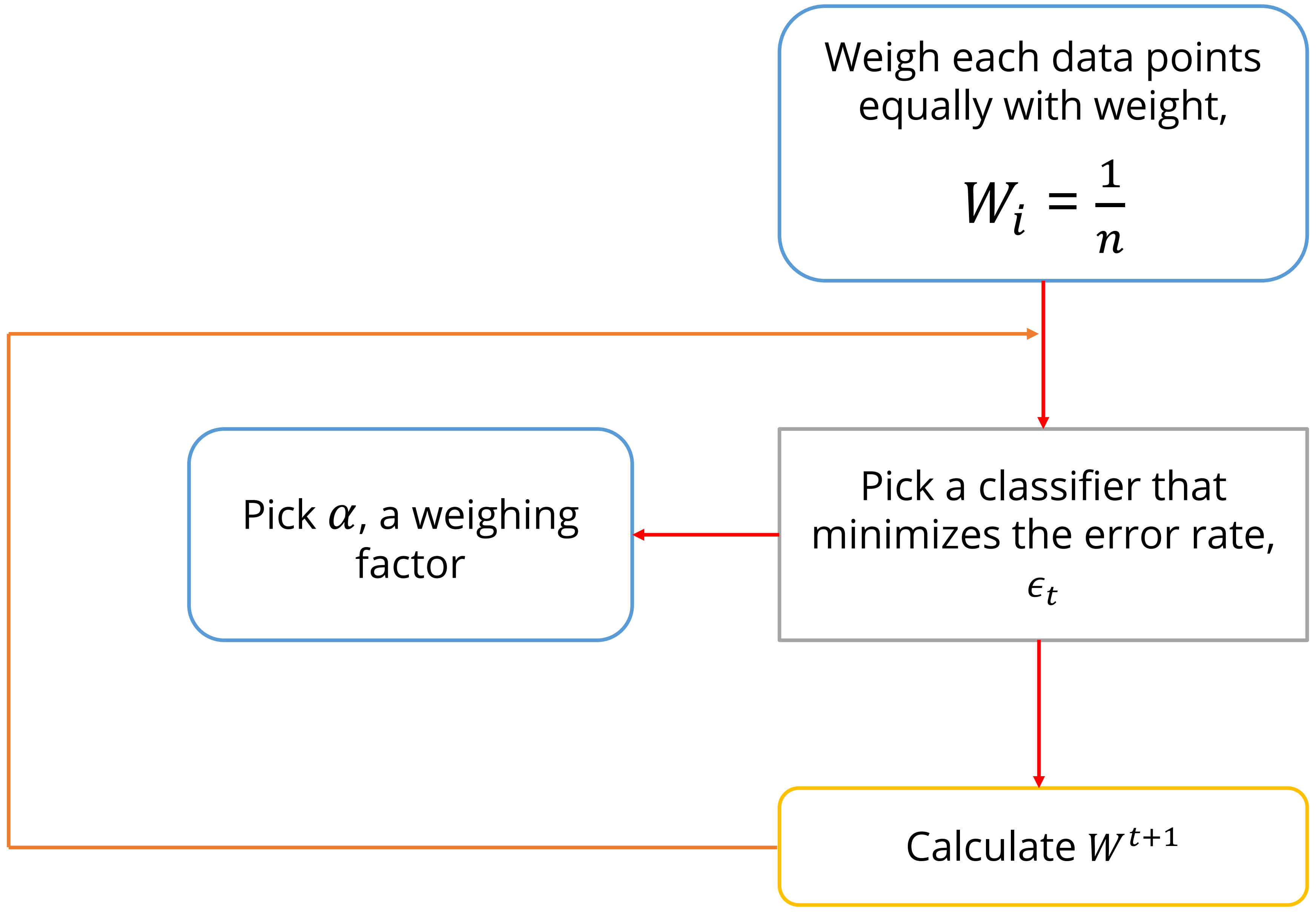
Weight after time t is given as :

$$\frac{w_i^{t+1}}{z} e^{-\alpha t.h1(x).y(x)}$$

where z is the normalizing factor, h1(x).y(x) is sign of the current output

©Simplilearn. All rights reserved.

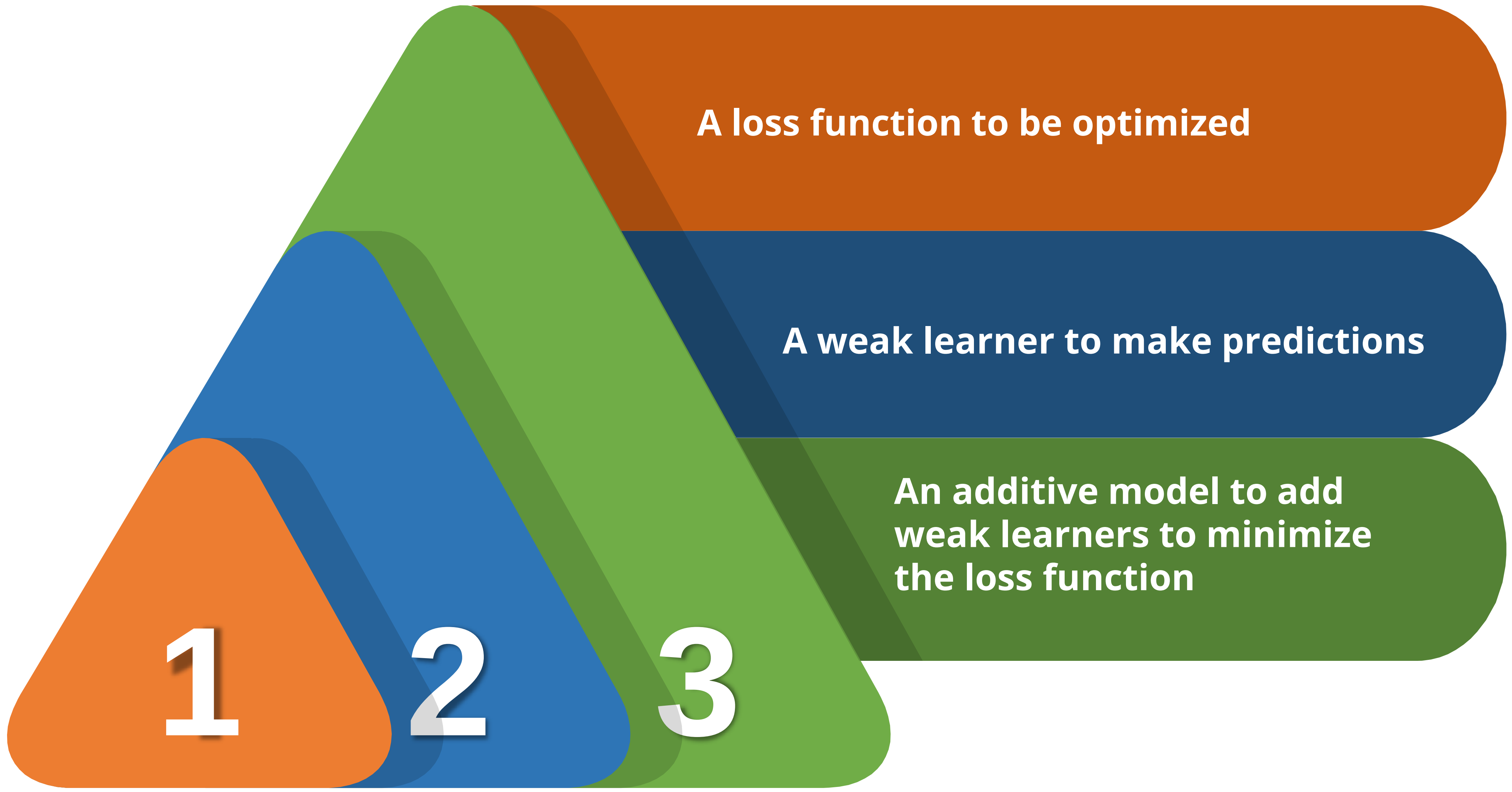
Adaboost Flowchart


Weigh each data points equally with weight, $W_i = \frac{1}{n}$
Pick a classifier that minimizes the error rate, ϵ_t
Pick α, a weighing factor
Calculate W^{t+1}

©Simplilearn. All rights reserved.

Gradient Boosting (GBM)

Gradient boosting involves three elements:



A loss function to be optimized

A weak learner to make predictions

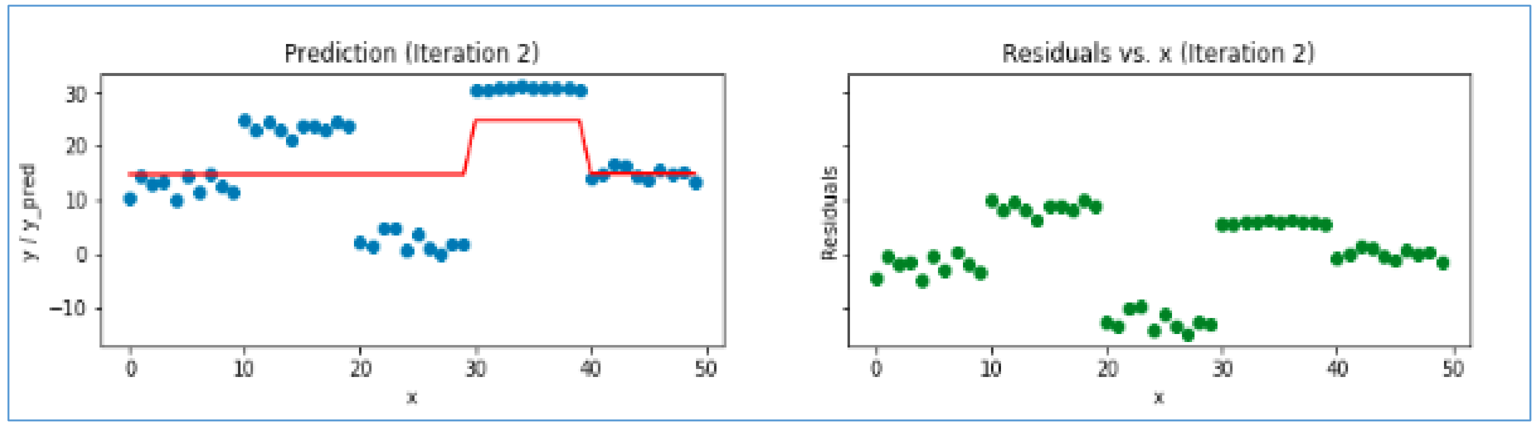
An additive model to add weak learners to minimize the loss function

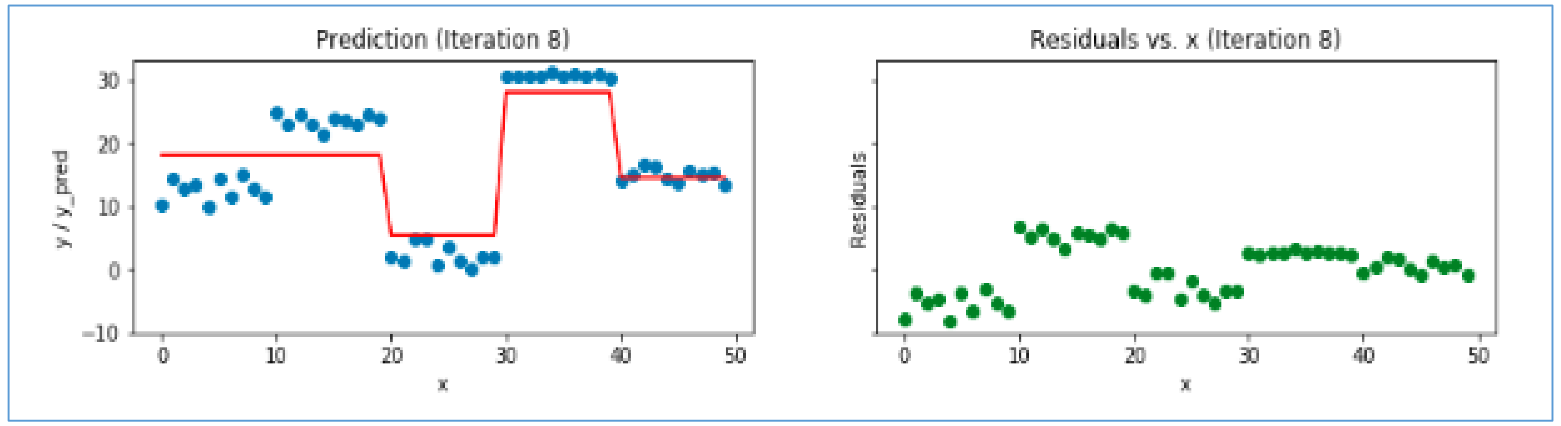
GBM minimizes the loss function (MSE) of a model by adding weak learners using a gradient descent procedure.

©Simplilearn. All rights reserved.

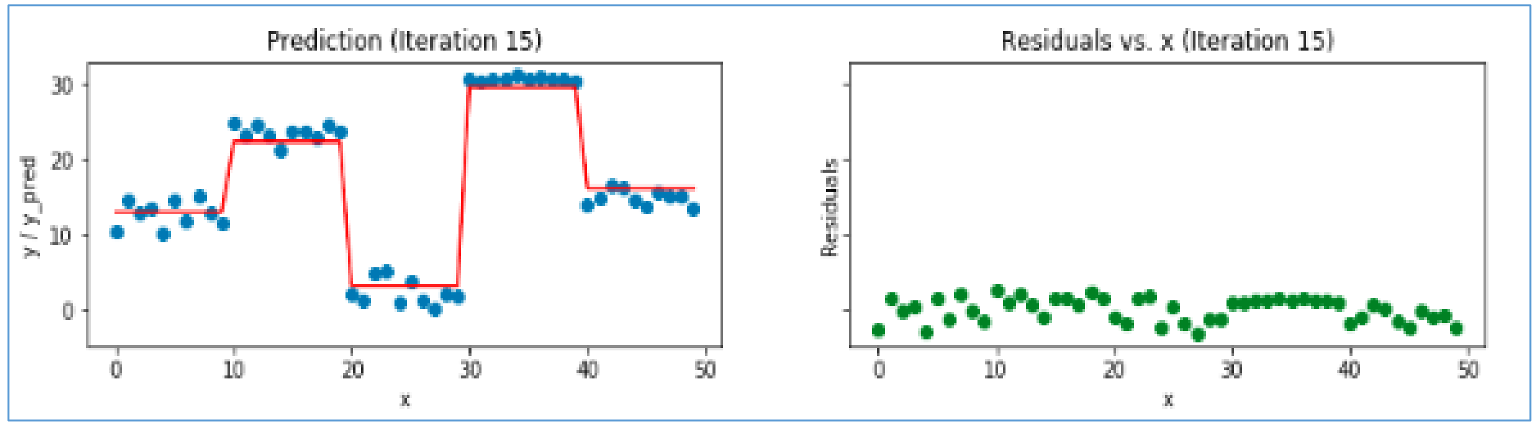
GBM Mechanism

01 GBM predicts the residuals or errors of prior models and then sums them to make the final prediction

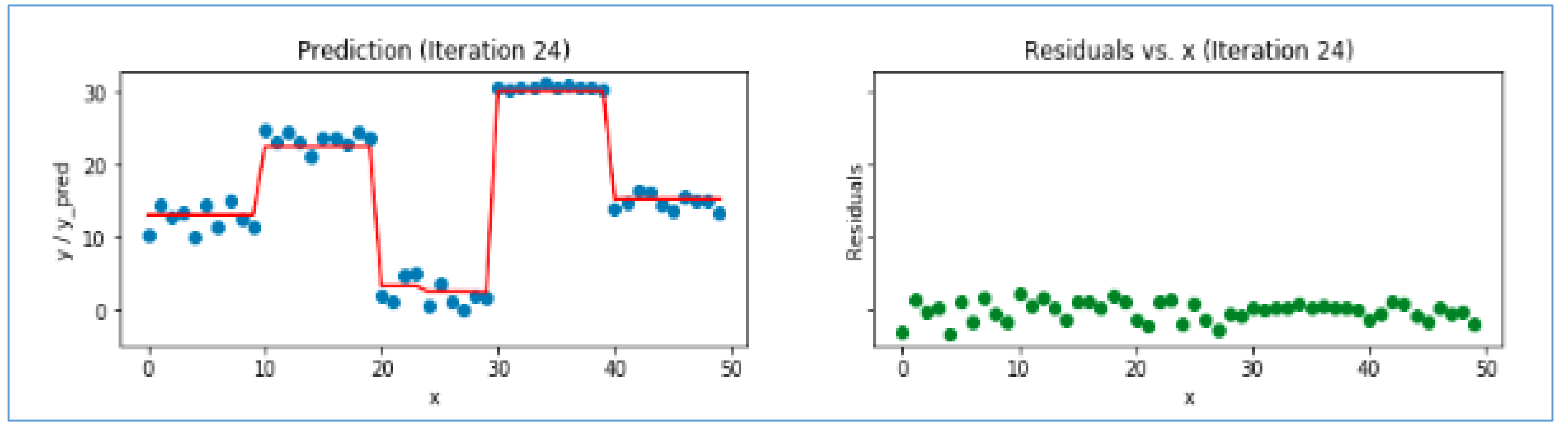




02 One weak learner is added at a time and existing weak learners in the model are left unchanged



03 GBM repetitively leverages the patterns in residuals and strengthens a model with weak predictions



04 Modeling is stopped when residuals do not have any pattern that can be modeled

©Simplilearn. All rights reserved.

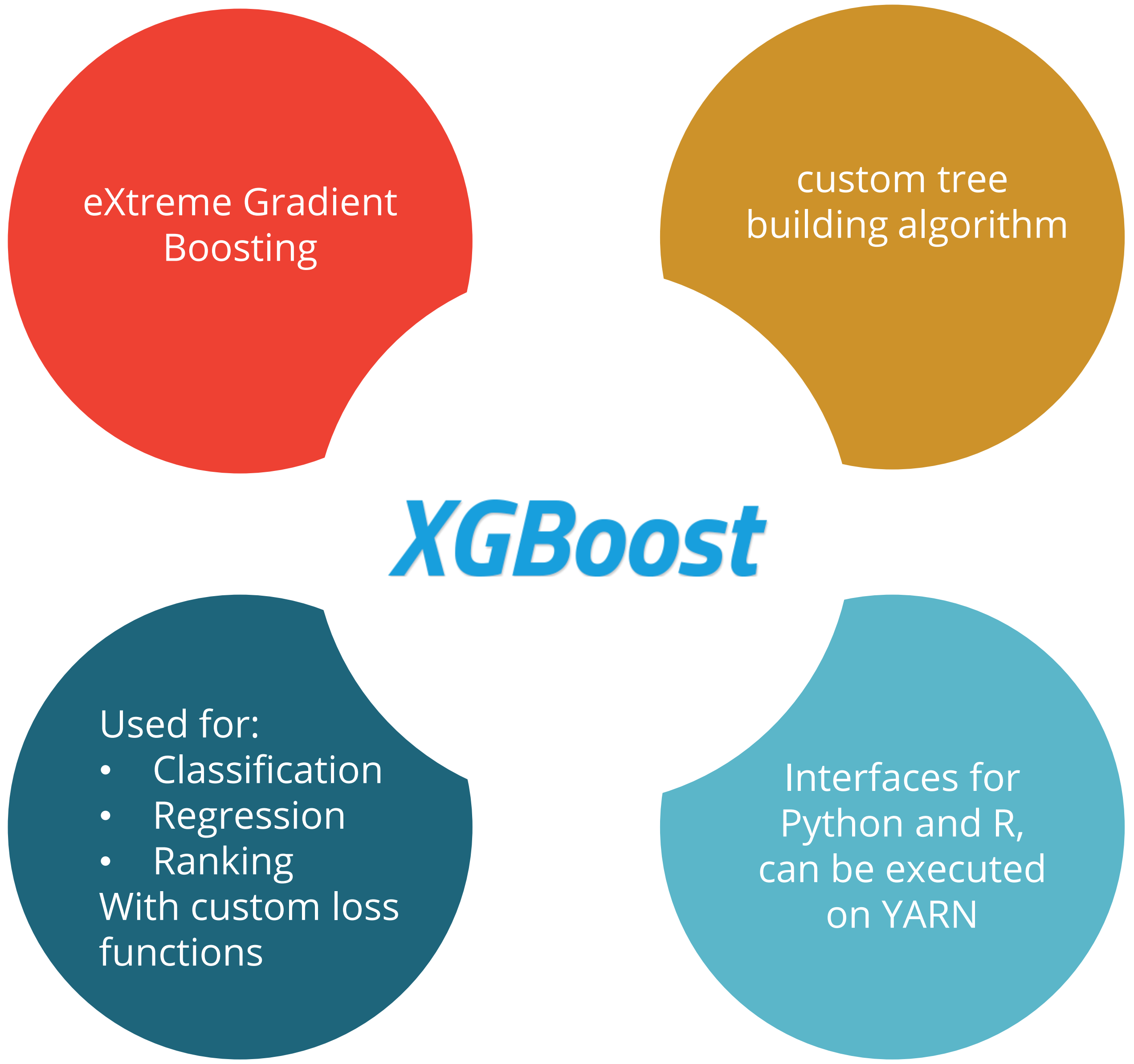
GBM Algorithm

Step 01	Fit a simple regression or classification model
Step 02	Calculate error residuals (actual value - predicted value)
Step 03	Fit a new model on error residuals as target variable with same input variables
Step 04	Add the predicted residuals to the previous predictions
Step 05	Fit another model on residuals that are remaining and repeat steps 2 and 5 until model is overfit or the sum of residuals becomes constant

©Simplilearn. All rights reserved.

XGBoost

e**X**treme **G**radient **B**oosting is a library for developing fast and high-performance gradient boosting tree models.


eXtreme Gradient Boosting
custom tree building algorithm
XGBoost
Used for:
• Classification
• Regression
• Ranking
With custom loss functions
Interfaces for Python and R, can be executed on YARN

XGBoost is extensively used in ML competitions as it is almost 10 times faster than other gradient boosting techniques

©Simplilearn. All rights reserved.

XGBoost Parameters

1

General Parameters

Number of threads

2

Booster Parameters

- Step size
- Regularization

3

Task Parameters

- Objective
- Evaluation metric

©Simplilearn. All rights reserved.

XGBoost Library Features

XGBoost library features tools are built for the sole purpose of model performance and computational speed.

01 SYSTEM

Parallelization
Tree construction using all CPU cores while training

Distributed Computing
Training very large models using a cluster of machines

Cache Optimization
Data structures make best use of hardware

02 ALGORITHM

Sparse Aware
Automatic handling of missing data values

Block Structure
Supports the parallelization of tree construction

Continued Training
To boost an already fitted model on new data

03 MODELS

Gradient Boosting
Gradient boosting machine algorithm including learning rate

Stochastic Gradient Boosting
Sub-sampling at the row, column and column per split levels

Regularized Gradient Boosting
With both L1 and L2 regularization

©Simplilearn. All rights reserved.

simplilearn

General Parameters

General parameters guide the overall functioning of XGBoost

- nthread
  - Number of parallel threads
  - If no value is entered, algorithm automatically detects the number of cores and runs on all the cores
- booster
  - gbtree: tree-based model
  - gblinear: linear function
- Silent [default =0]
  - if set to 1, no running messages will be printed. Hence, keep it ‘0’ as the messages might help in understanding the model

©Simplilearn. All rights reserved.

Booster Parameters

Booster parameters guide individual booster (Tree/Regression) at each step

Parameters for tree booster

- eta
  - Step size shrinkage is used in update to prevent overfitting
  - Range in [0,1], default 0.3
- gamma
  - Minimum loss reduction required to make a split
  - Range [0,∞], default 0
- max_depth
  - Maximum depth of a tree
  - Range [1, ∞], default 6
- min_child_weight
  - Minimum sum of instance weight needed in a child
  - Range [0, ∞], default 1

©Simplilearn. All rights reserved.

Booster Parameters (Contd.)

Parameters for tree booster

- max_delta_step
  - Maximum delta step allowed in each tree's weight estimation
  - Range in [0, ∞], default 0
- subsample
  - Subsample ratio of the training instance
  - Range [0,1], default 1
- Colsample_bytree
  - Subsample ratio of columns when constructing each tree
  - Range [0, 1], default 1

©Simplilearn. All rights reserved.

Booster Parameters (Contd.)

Parameters for Linear booster

- lambda
  - L2 regularization term on weights
  - default 0
- alpha
  - L1 regularization term on weights
  - default 0
- Lambda_bias
  - L2 regularization term on bias
  - default 0

©Simplilearn. All rights reserved.

Task Parameters

Task parameters guide optimization objective to be calculated at each step

1. Objectives [default = reg:linear]

- "binary:logistic": logistic regression for binary classification, output is probability not class
- "multi:softmax": multiclass classification using the softmax objective, need to specify num_class

2. Evaluation Metric

- "rmse"
- "logloss"
- "error"
- "auc"
- "merror"
- "mlogloss"

i https://xgboost.readthedocs.io/en/latest/parameter.html

©Simplilearn. All rights reserved.

Assisted Practice

Boosting

Duration: 15 mins.

Problem Statement: The *Pima Indians Diabetes* dataset has diagnostic measures like BMI, blood pressure of female patients of more than 21 years old.

Objective:

- Classify whether the person is diabetic with maximum accuracy

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

©Simplilearn. All rights reserved.

Unassisted Practice

Boosting

Duration: 20 mins.

Problem Statement: The Iris plant has 3 species : Iris Setosa, Iris Versicolour, Iris Virginica
One class is linearly separable from the other two whereas the latter are not linearly separable from each other.

Objective:

- Import the iris dataset using sklearn
- Build a classification model using AdaBoost and XGBoost
- Compare accuracy of both the models

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

©Simplilearn. All rights reserved.

Step 1: Data Import

Code

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import metrics

iris = datasets.load_iris()
X = iris.data
y = iris.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

©Simplilearn. All rights reserved.

Step 2: Classifier

Code

```
abc = AdaBoostClassifier(n_estimators=50, learning_rate=1)
model = abc.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9555555555555556
```

```
from sklearn import svm
from xgboost import XGBClassifier
clf = XGBClassifier()
clf.fit(X_train, y_train)
y_pred2 = clf.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred2))
```

```
Accuracy: 0.9777777777777777
```

©Simplilearn. All rights reserved.

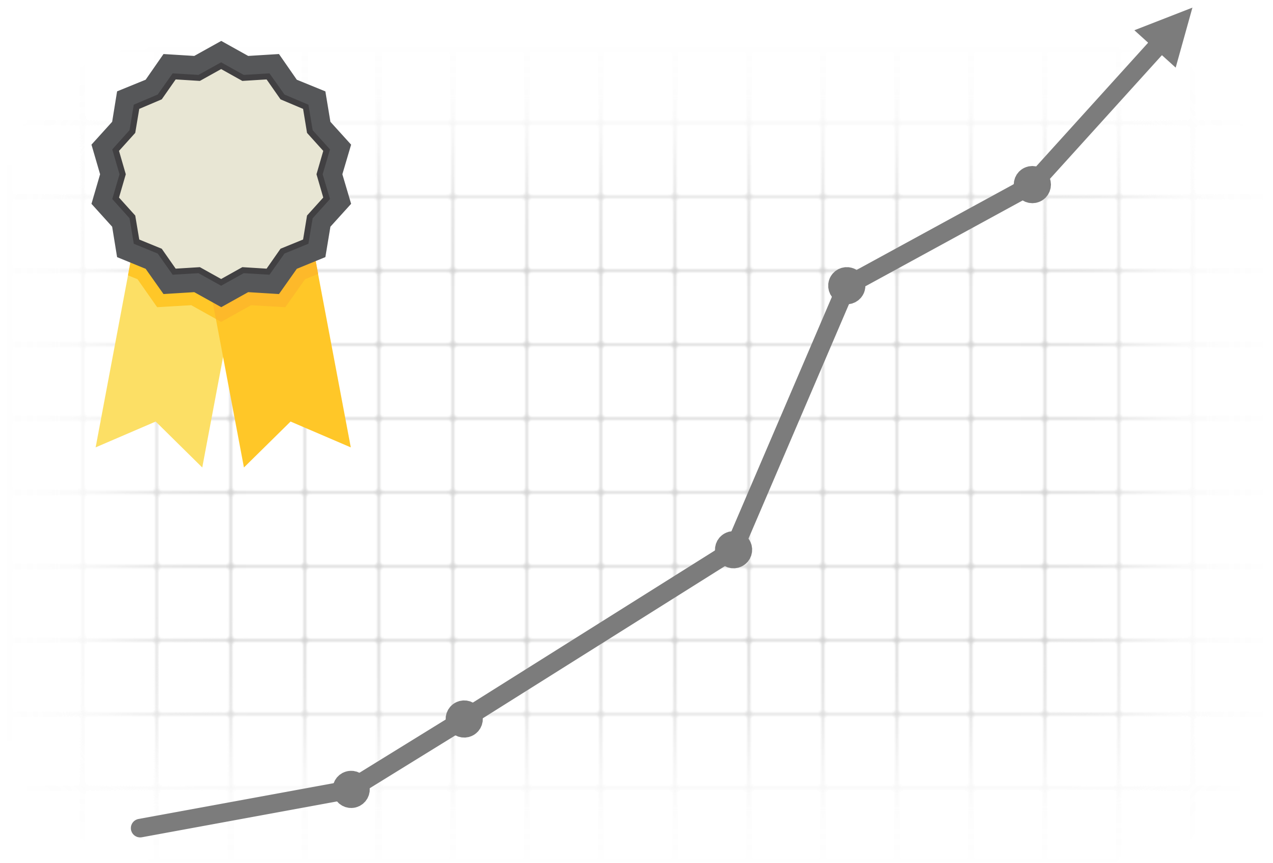
Ensemble Learning

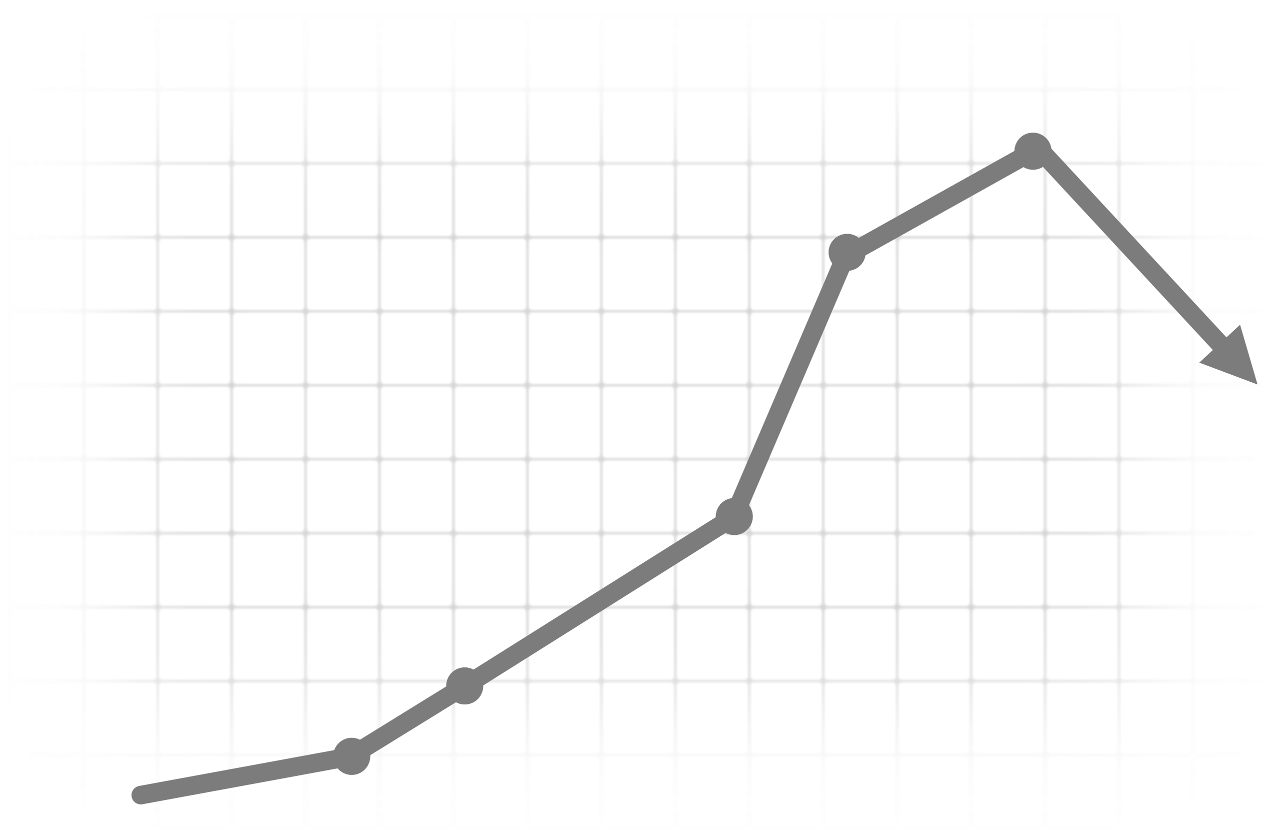
Topic 3: Model Selection

©Simplilearn. All rights reserved.

Model Evaluation

Models can be evaluated based on their measure of performance



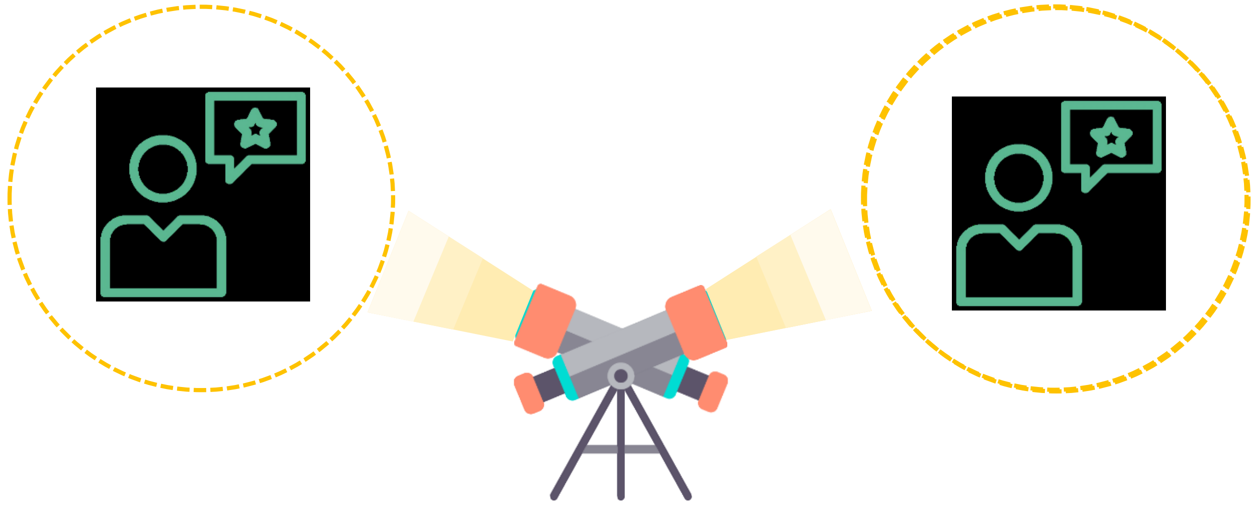


©Simplilearn. All rights reserved.

Assessing Model Performance

Train/Test Split

- Divide the training dataset
- Train on the first training set
- Test on the second set



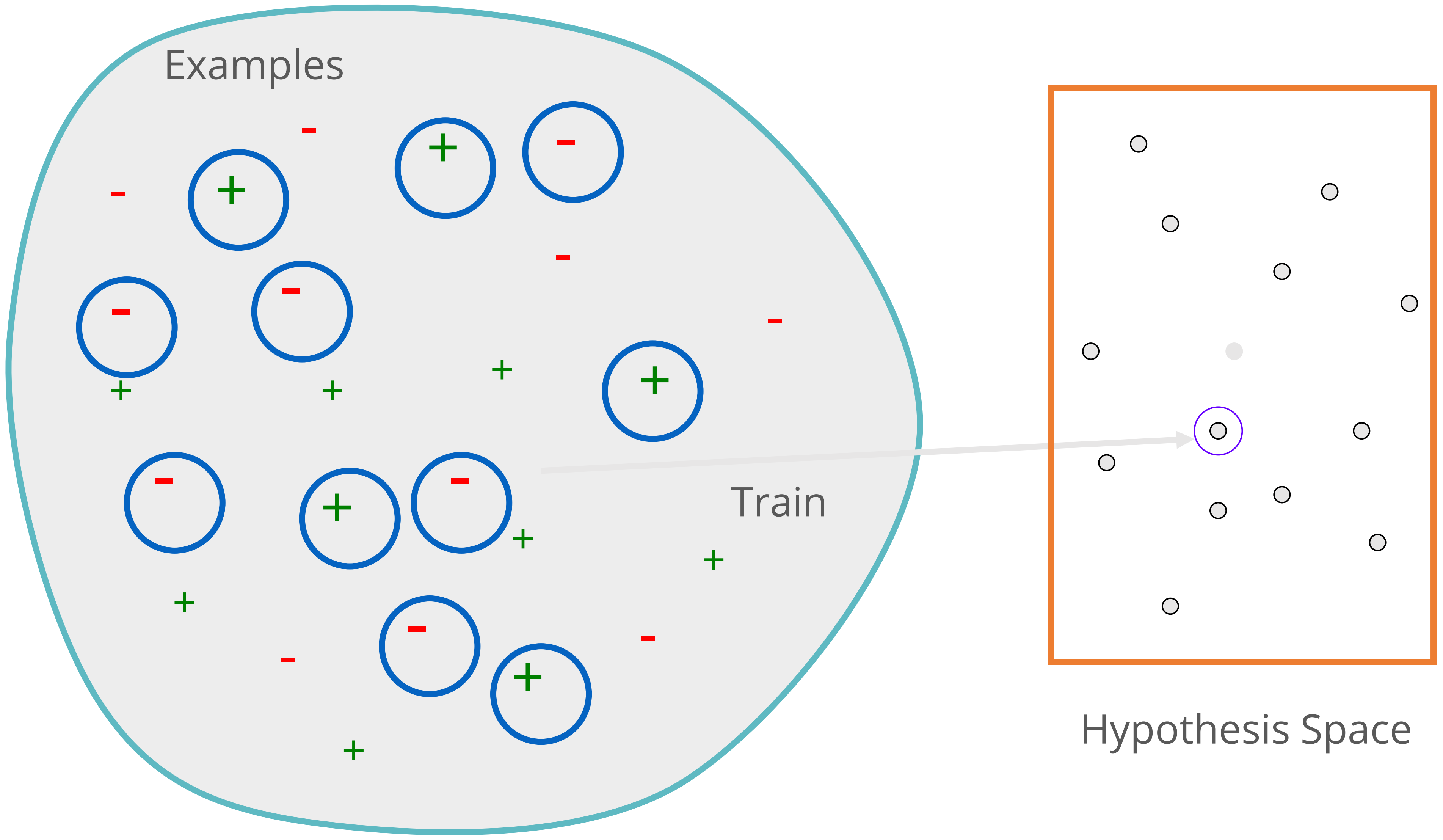
Cross Validation Split

- Sets of train/test splits created
- Accuracy for each set is checked
- Results are averaged

Techniques to assess model performance

©Simplilearn. All rights reserved.

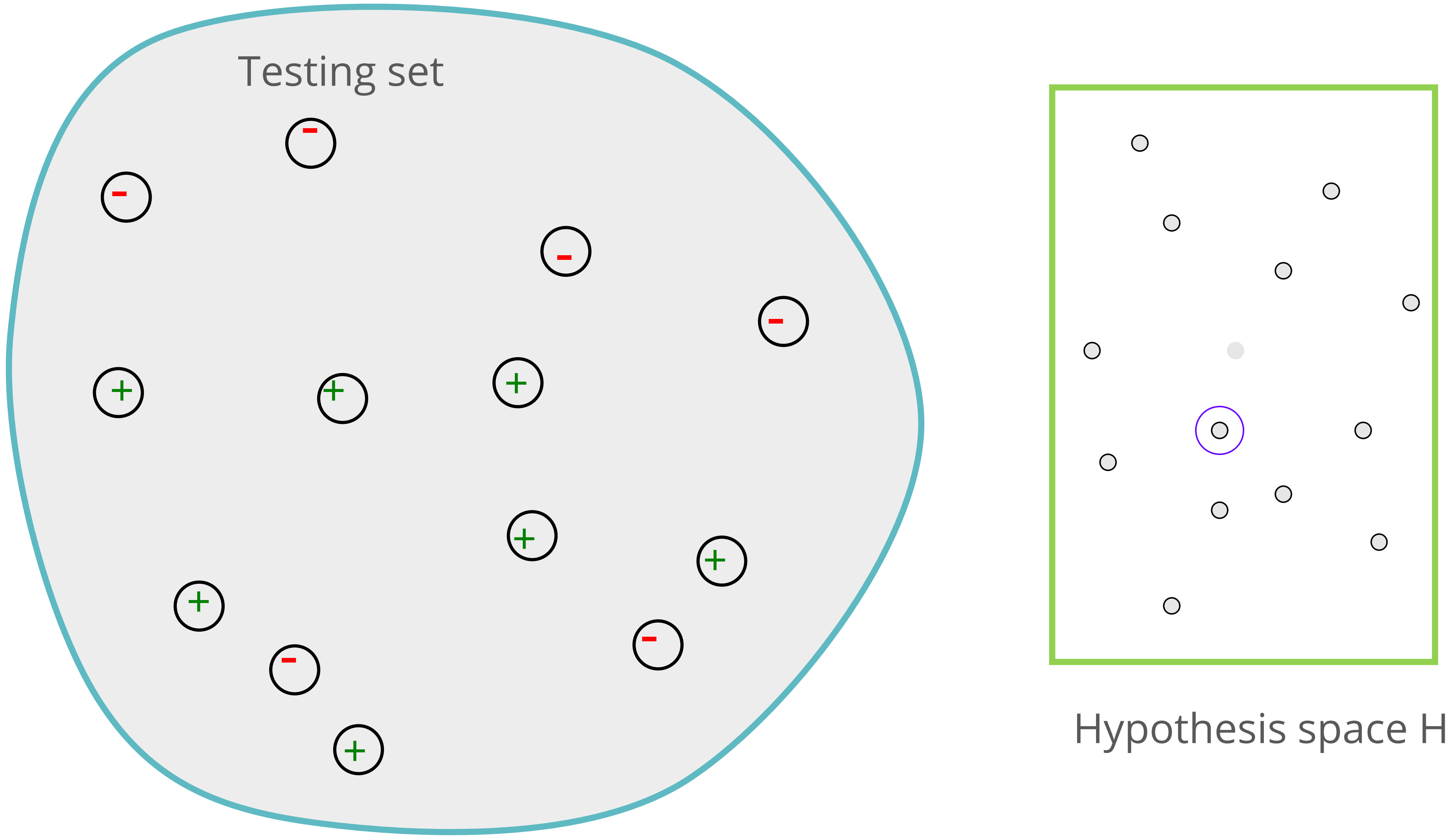
Train/Test Split



Creating the training dataset

©Simplilearn. All rights reserved.

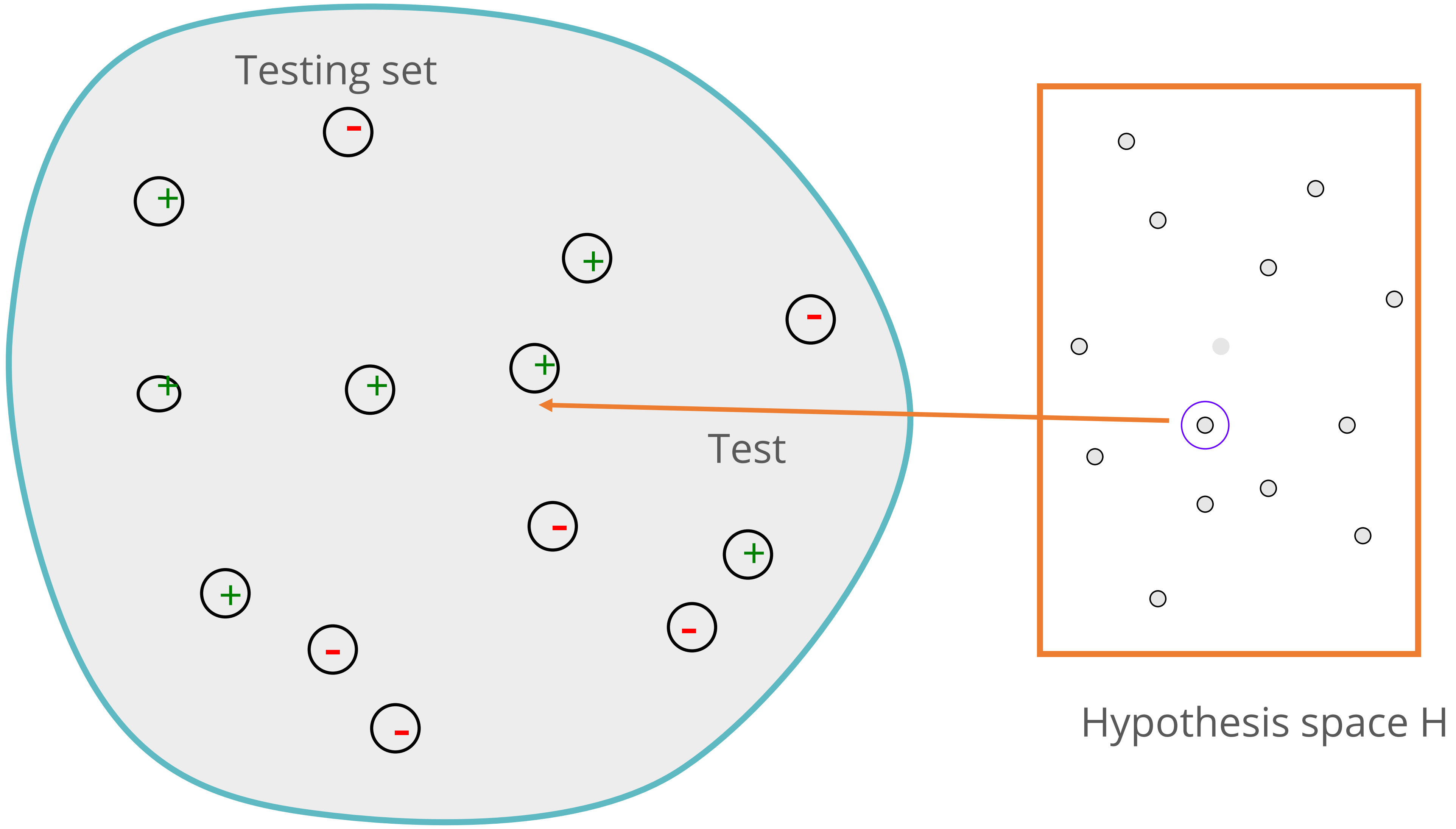
Train/Test Split (Contd.)



Creating the testing dataset

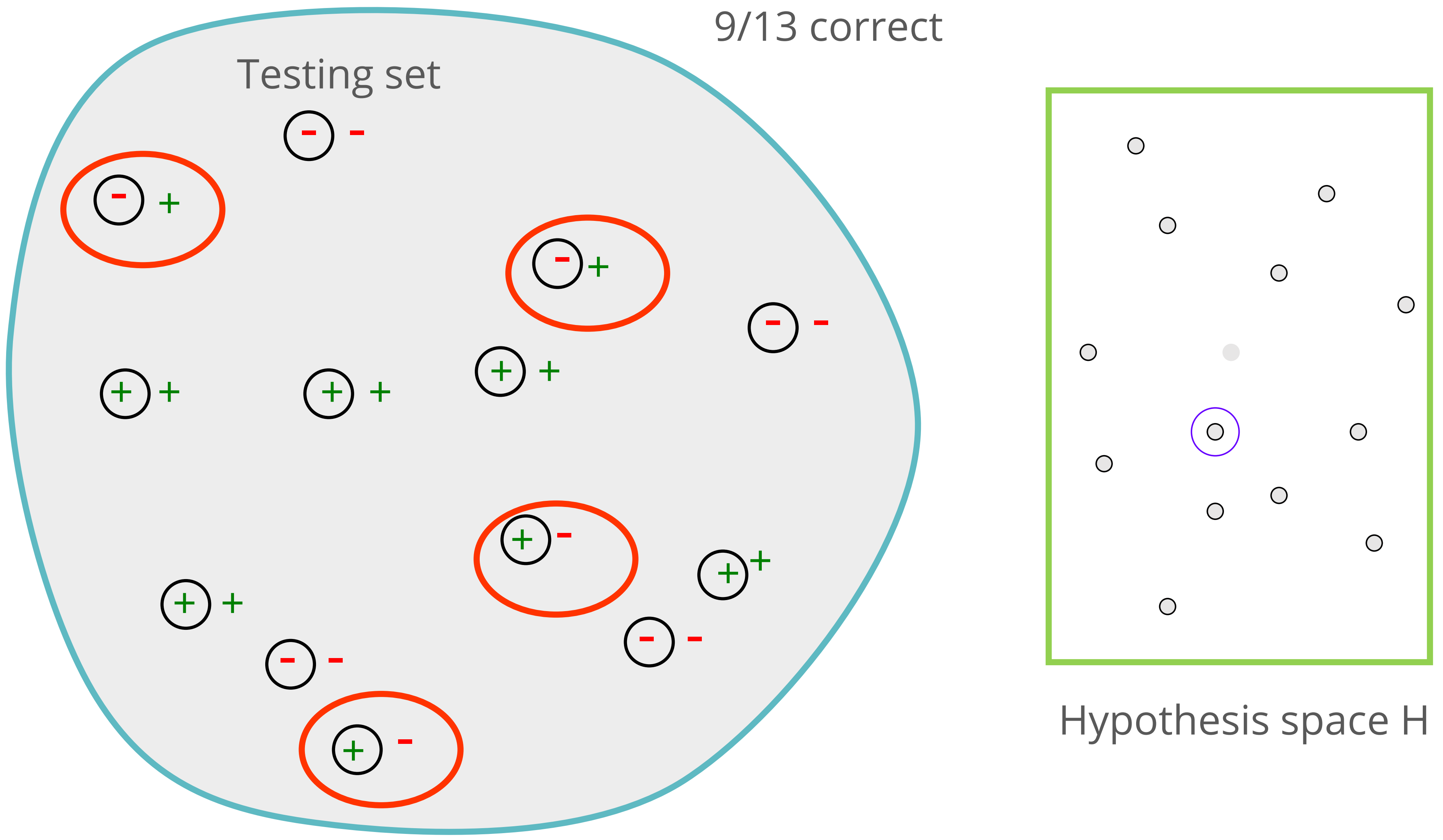
©Simplilearn. All rights reserved.

Train/Test Split (Contd.)



©Simplilearn. All rights reserved.

Train/Test Split (Contd.)



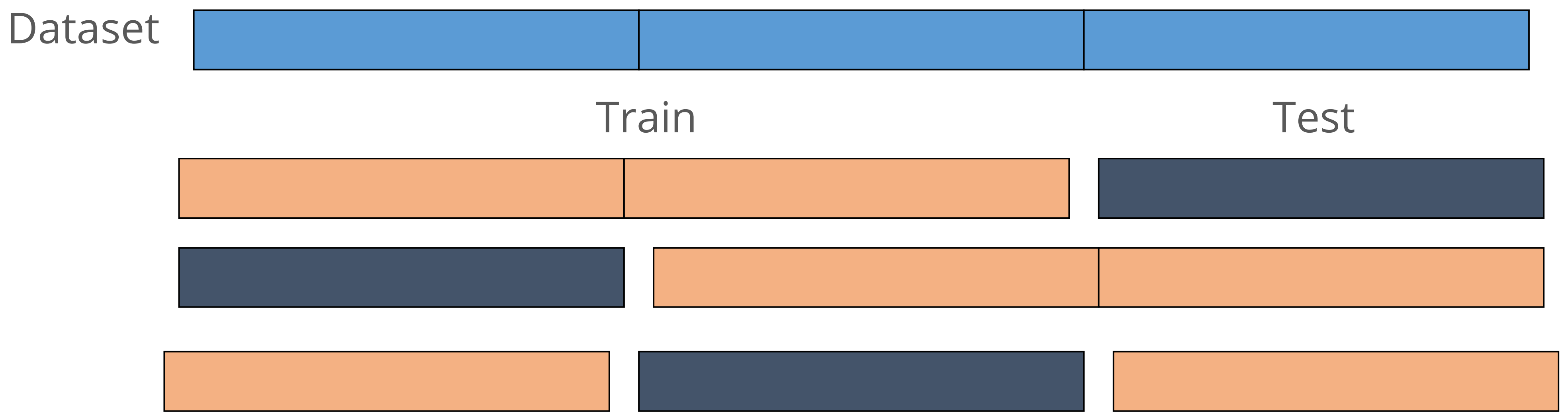
Verifying the results

©Simplilearn. All rights reserved.

Common Splitting Strategies

K-Fold Cross-Validation

Leave-one-out



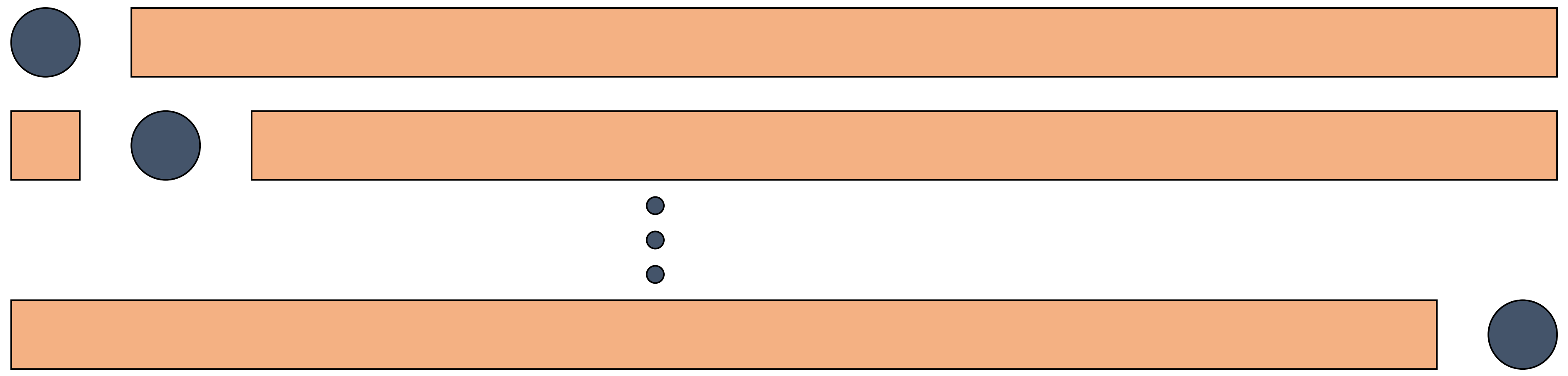
Identifying the Train/Test split ratio

©Simplilearn. All rights reserved.

Common Splitting Strategies (Contd.)

K-Fold Cross-Validation

Leave-one-out



©Simplilearn. All rights reserved.

Train/Test Split vs. Cross-Validation

Cross-validation

- More accurate estimate of out-of-sample accuracy
- More efficient use of data(every observation is used for both training and testing)

Train/Test Split

- Runs K-times faster than K-fold cross-validation
- Simpler to examine the detailed results of testing process

©Simplilearn. All rights reserved.

Assisted Practice

Cross-validation

Duration: 15 mins.

Problem Statement: Few learners have implemented random forest classifier on the Iris data but, better accuracy can be achieved using cross-validation sampling technique.

Objective:

- Generate the random forest using cross validation splitting technique.
- Determine the accuracy such that it is the average of all the resultant accuracies.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

©Simplilearn. All rights reserved.

Unassisted Practice

Cross-Validation

Duration: 20 mins.

Problem Statement: Mtcars, an automobile company in Chambersburg, United States has recorded the production of its cars within a dataset. In order to classify cars, the company has come up with two classification models (KNN and Logistic Regression).

Objective: Perform a model selection between the above two models using the sampling technique as 10-fold cross- validation.

Note: This practice is not graded. It is only intended for you to apply the knowledge you have gained to solve real-world problems.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

©Simplilearn. All rights reserved.

Defining a 10-Fold KNN Model

Define a 10-fold KNN model and calculate the average of each accuracy matrix obtained.

Code

```
from sklearn.cross_validation import cross_val_score
knn = KNeighborsClassifier(n_neighbors=4)
print(cross_val_score(knn, x, y, cv=10, scoring ='accuracy').mean())
```

The accuracy came out to be 56.66%.

©Simplilearn. All rights reserved.

Defining a 10-Fold Logistic Regression Model

Define a 10-fold Logistic Regression model and calculate the average of each accuracy matrix obtained.

Code

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
print (cross_val_score(logreg, x, y, cv=10, scoring = 'accuracy').mean())
```

The accuracy came out to be 28.33%.

Hence, you can infer that KNN model for the task is better as compared to Logistic Regression model.

©Simplilearn. All rights reserved.

Key Takeaways

Now, you are able to:

- ✓ Explain ensemble learning
- ✓ Evaluate performance of boosting models

©Simplilearn. All rights reserved.

? Knowledge Check

© Simplilearn. All rights reserved.

Knowledge Check

1

Which of the following is not an ensemble method?

a. Decision Tree

b. Random Forest

c. Adaboost

d. None of the above

© Simplilearn. All rights reserved.

Knowledge Check

1

Which of the following is not an ensemble method?

a. Decision Tree

b. Random Forest

c. Adaboost

d. None of the above

The correct answer is **a. Decision Tree**

In decision tree, single tree is built and no ensembling is required.

©Simplilearn. All rights reserved.

Knowledge Check

2

Some of the advantages of XGBoost include:

a. Parallelization

b. Handling missing values

c. Support for multiple GBM models

d. All of the above

© Simplilearn. All rights reserved.

Knowledge Check

2

Some of the advantages of XGBoost include:

a. Parallelization

b. Handling missing values

c. Support for multiple GBM models

d. All of the above

The correct answer is **d. All of the above**

XGBoost is scalable and has accurate implementation of gradient boosting machines.

It has proven to push the limits of computing power for boosted trees.

©Simplilearn. All rights reserved.

Lesson-End Project

Car Evaluation Database

Duration: 30 mins

Problem Statement: Used car market has significantly grown in recent times with clients ranging from used car dealers and buyers.You are provided with a car evaluation dataset that has features like price, doors, safety, and so on.
You are required to create a robust model that allows stakeholders to predict the condition of a used vehicle.

Objective:

- Predict the condition of a vehicle based on features
- Plot the most import features
- Train multiple classifiers and compare the accuracy
- Evaluate XGBoost model with K-fold cross-validation

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

©Simplilearn. All rights reserved.

Thank You

© Simplilearn. All rights reserved.