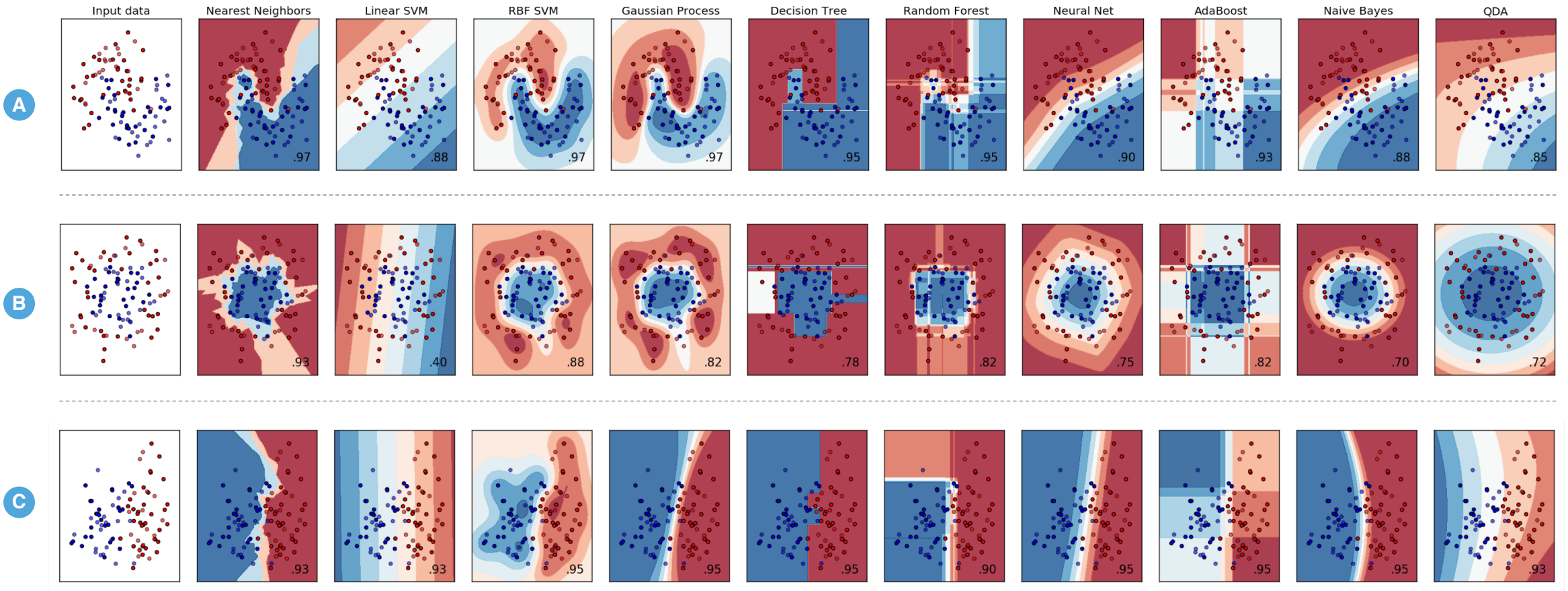
:: Example: how to classify the data points ?

Use different algorithms to classify the data set.

In this example, there are 3 different data sets- A, B, C. You can see how they're classified with different algorithms.



The plots show training points in solid colors and testing points semi-transparent. The lower right shows the classification accuracy on the test set.

Source: Classifier comparison http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#

There is a wide range of algorithms …

Regression

Ordinal Regression
Data in rank ordered categories

Poisson Regression
Predicts event counts

Fast forest quantile regression
Predicts a distribution

Linear Regression
Fast training, linear model

Bayesian Linear Regression
Linear model, small data sets

Neural Network Regression
Accurate, long training times

Decision Forest Regression
Accurate, fast training times

Boosted Decision Tree Regression
Accurate, fast training times, large memory footprint

Clustering

K-Means
Unsupervised learning

Anomaly Detection

PCA-Based Anomaly Detection
Fast training times

Two-Class Classification
Under 100 features, aggressive boundary

Two –Class Classification

Two-class SVM
Under 100 features, linear model

Two-class averaged perceptron
Fast training, linear model

Two-class Bayes point machine
Fast training, linear model

Two-class decision forest
Accurate, fast training

Two-class logistic regression
Fast training, linear model

Two-class boosted decision tree
Accurate, fast training, large memory footprint

Two-class decision jungle
Accurate, small memory footprint

Two-class locally deep SVM
Under 100 features

Two-class neural network
Accurate, long training times

Multiclass Classification

Multiclass logistic regression
Fast training times, linear model

Multiclass neural network
Accuracy, long training times

Multiclass decision forest
Accuracy, fast training times

Multiclass decision jungle
Accuracy, small memory footprint

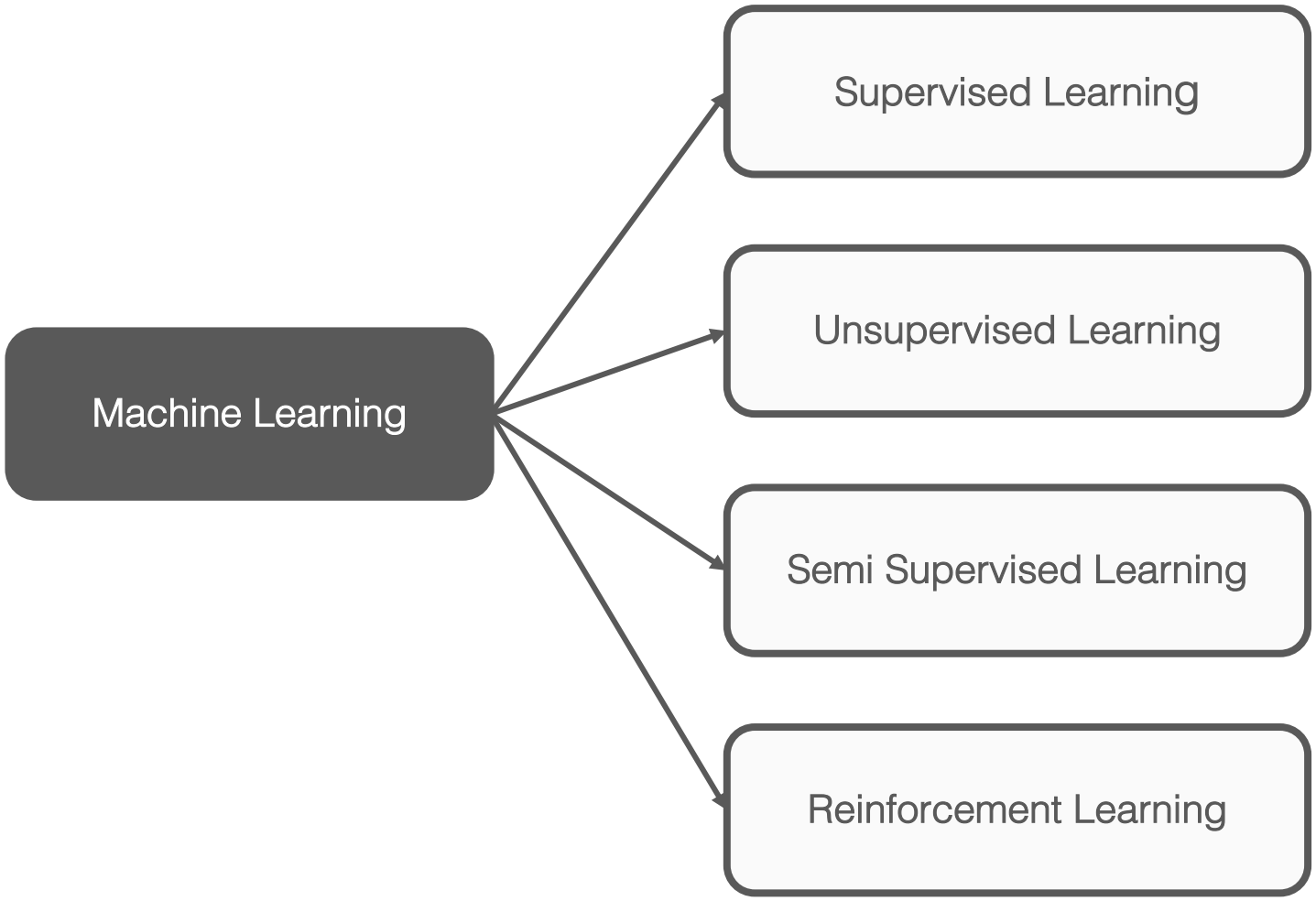
One-v-all multiclass
Depends on the two-class classifier

Looks Overwhelming?

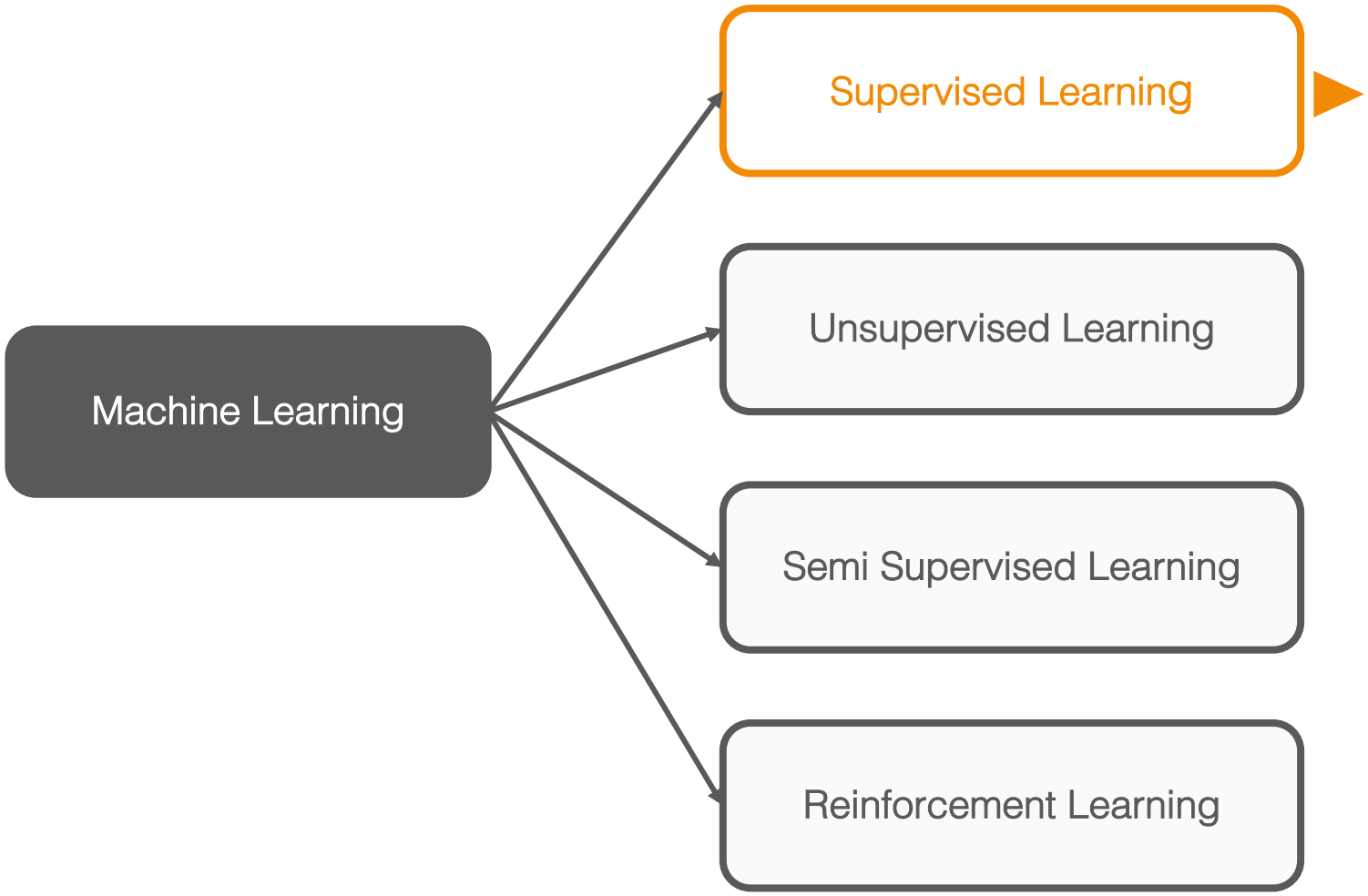
Don't worry! The details will be explained later on.

Source: http://download.microsoft.com/download/0/5/A/05AE6B94-E688-403E-90A5-6035DBE9EEC5/machine-learning-basics-infographic-with-algorithm-examples.pdf

:: The categories of machine learning algorithms



:: The categories of machine learning algorithms

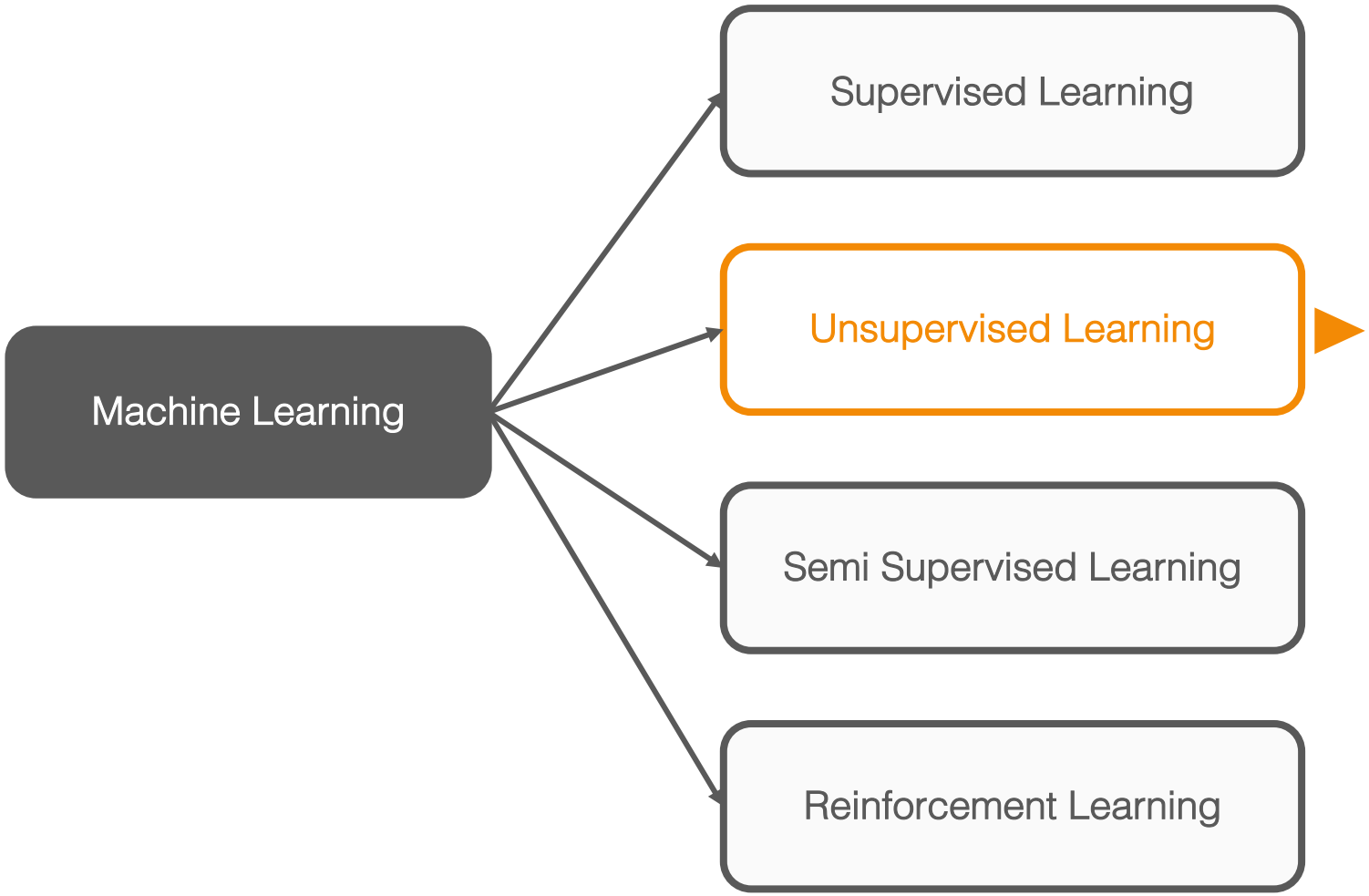


- **Train the model with labeled data**
  A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data.[1]

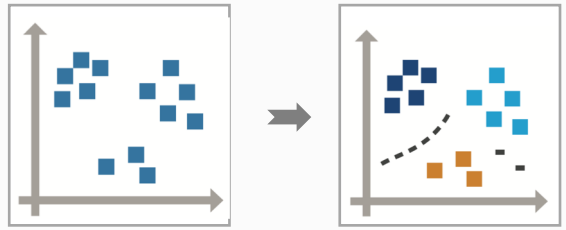
- **Most machine learning is supervised.**

1 Source: mathworks

:: The categories of machine learning algorithms



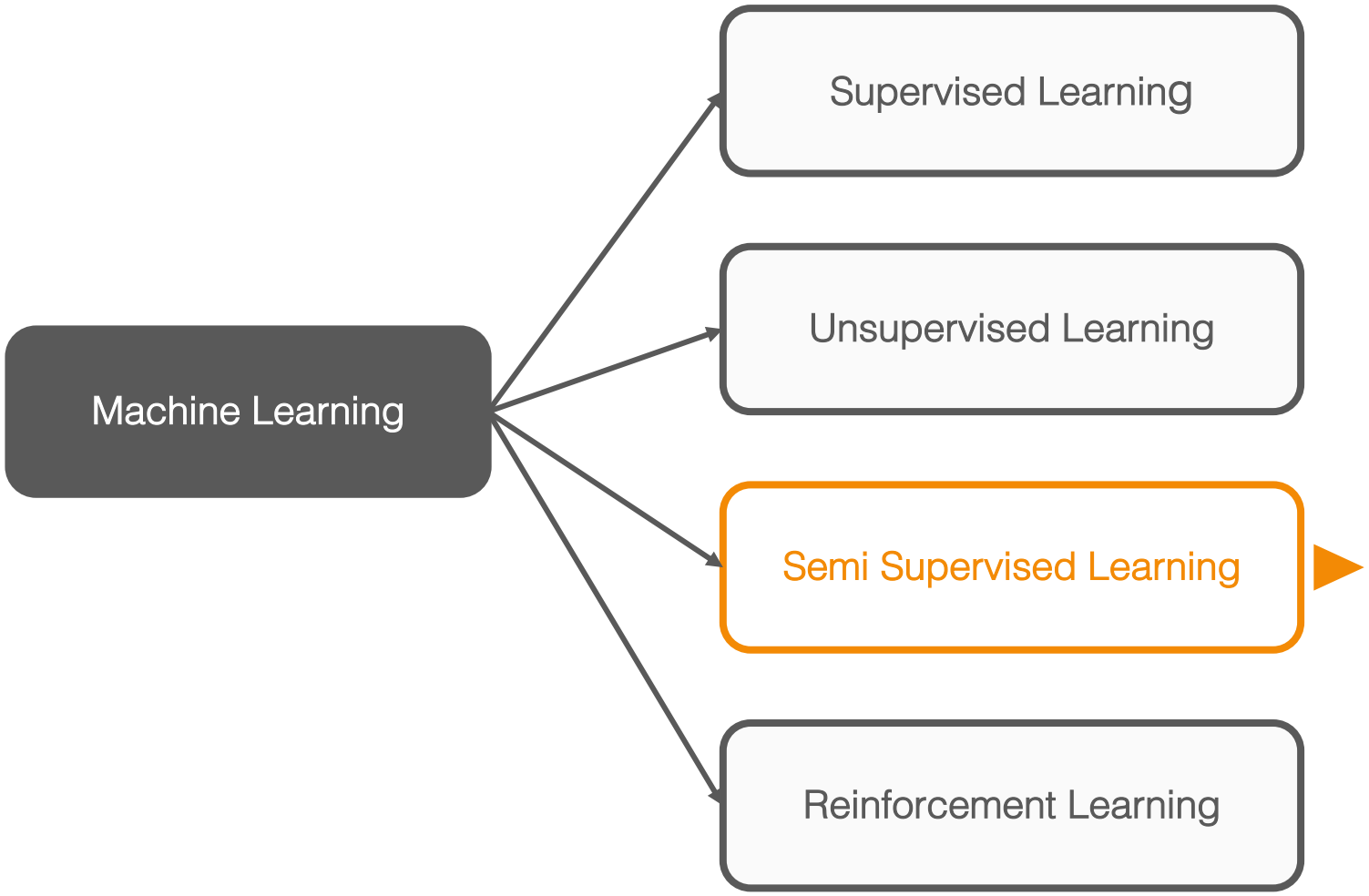
- **Train the model with unlabeled data**
  Unsupervised learning is used on data without labels
- The goal is to find patterns or intrinsic structures in the data. It's also a good way to simplify data somehow (reduce dimensions, remove unnecessary variables or detect anomalies).
- Clustering is the most common unsupervised learning technique.



Clustering Patterns in the Data

1 Source : Mathworks

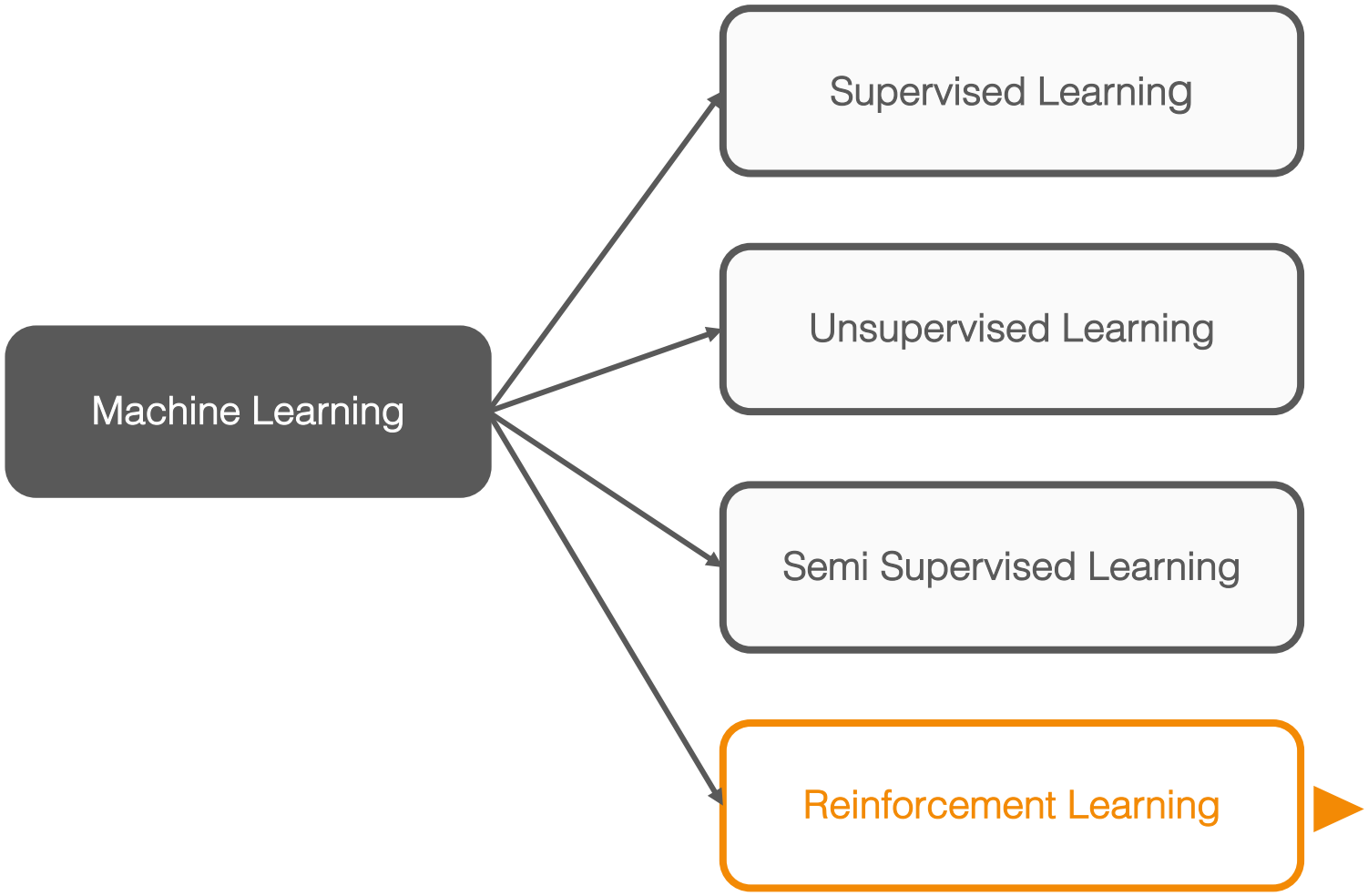
:: The categories of machine learning algorithms



- **Partially-labeled Training Dataset**
  Typically a small amount of labeled data with a large amount of unlabeled data during the training phase[1].
- Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data)[2].
- The trained models that result from this training set can be highly accurate and less expensive to train compared to using all labeled data[3].

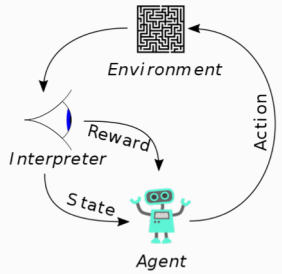
1 &2Source : https://en.wikipedia.org/wiki/Semi-supervised_learning
3. Source : https://hackernoon.com/ways-in-which-machines-learn-b1824464dd5f

:: The categories of machine learning algorithms



- The model is learned from a series of actions by maximizing a “reward function”. The reward function can either be maximized by penalizing “bad actions” and/or rewarding “good actions”. A popular example of reinforcement learning would be the training of self-driving car using feedback from the environment.[1]

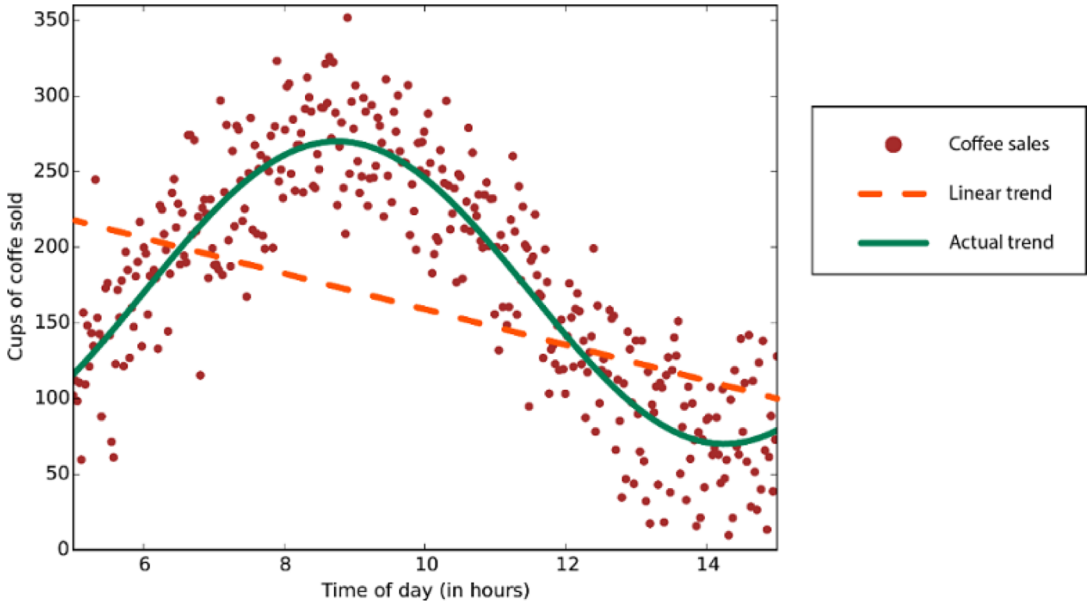
- The typical framing of a Reinforcement Learning (RL) scenario: an agent takes actions in an environment which is interpreted into a reward and a representation of the state which is fed back into the agent.[2]



1 source : https://github.com/rasbt/pattern_classification/blob/master/machine_learning/supervised_intro/introduction_to_supervised_machine_learning.md
2 source: https://en.wikipedia.org/wiki/Reinforcement_learning

:: You should choose the suitable algorithm

Unsuitable algorithm will result in low accuracy

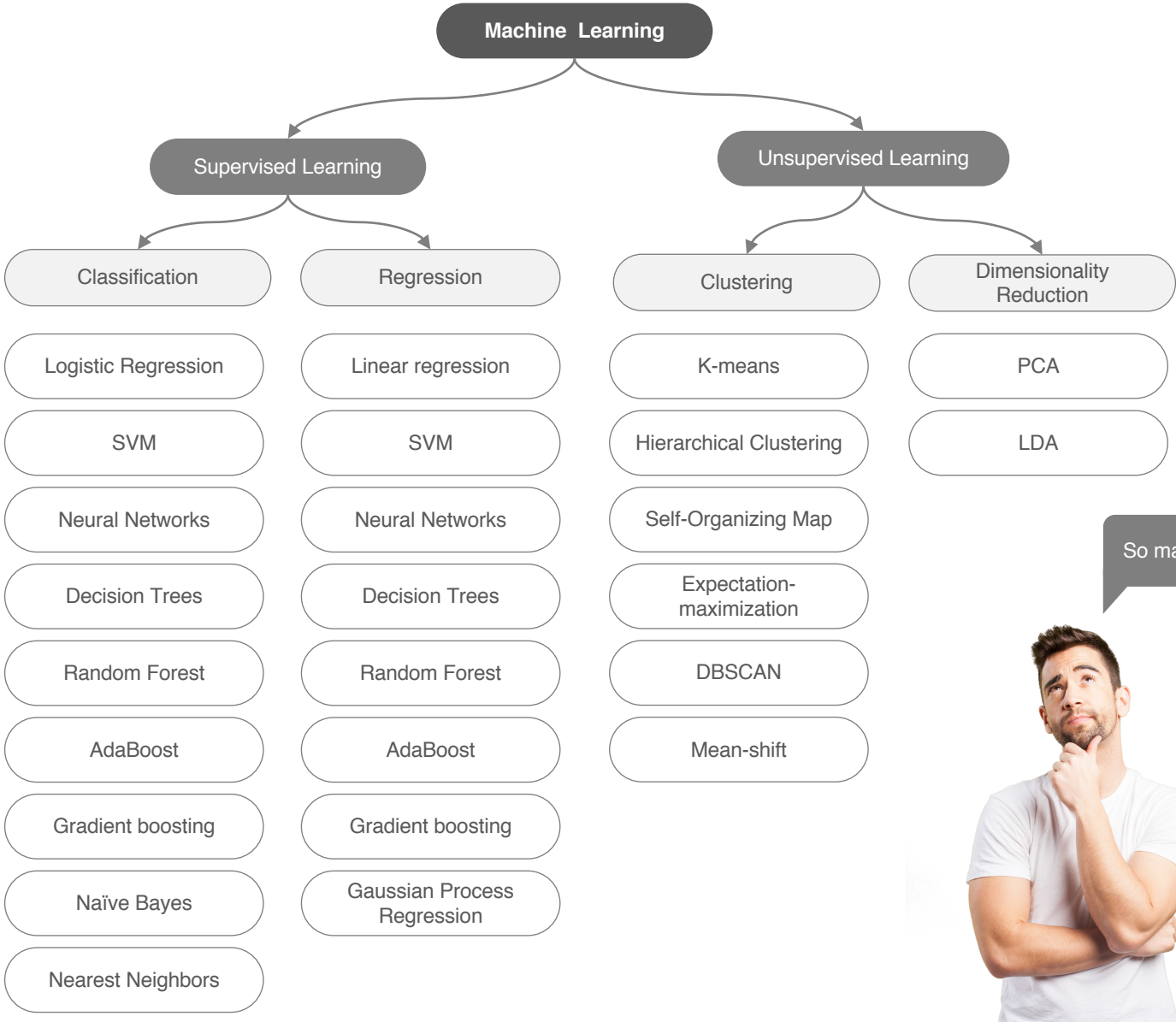


Data with a nonlinear trend

In this example, the data points are not linear . So using a linear regression method would generate much larger errors than necessary

Source: https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice

:: Some of Machine Learning Algorithms



:: Which algorithm should I use ?

Selecting a machine learning algorithm is a process of trial and error

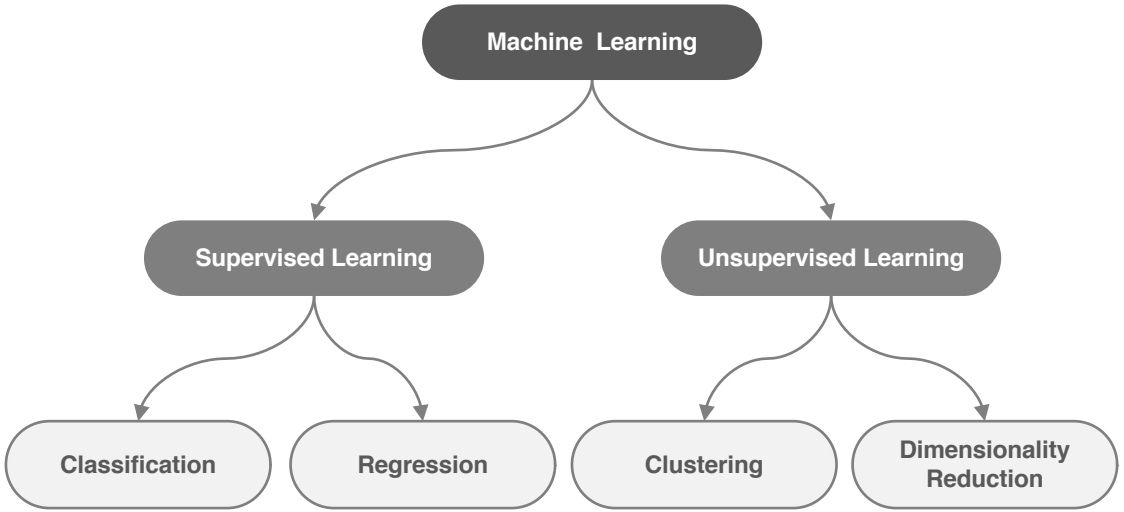
Choosing the right algorithm can seem overwhelming—there are dozens of supervised and unsupervised machine learning algorithms, and each takes a different approach to learning. There is no best method or one size fits all.

Finding the right algorithm is partly just trial and error—even highly experienced data scientists can't tell whether an algorithm will work without trying it out. [2]

How to select the right algorithm ?

The answer to the question varies depending on many factors, including: [1]

- The size, quality, and nature of data.
- The available computational time.
- The urgency of the task.
- What you want to do with the data.



1. source: Mathworks, Introducing Maching Learning
2. Source: SAS, https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/

:: Want to know the details of Machine learning Algorithm ?

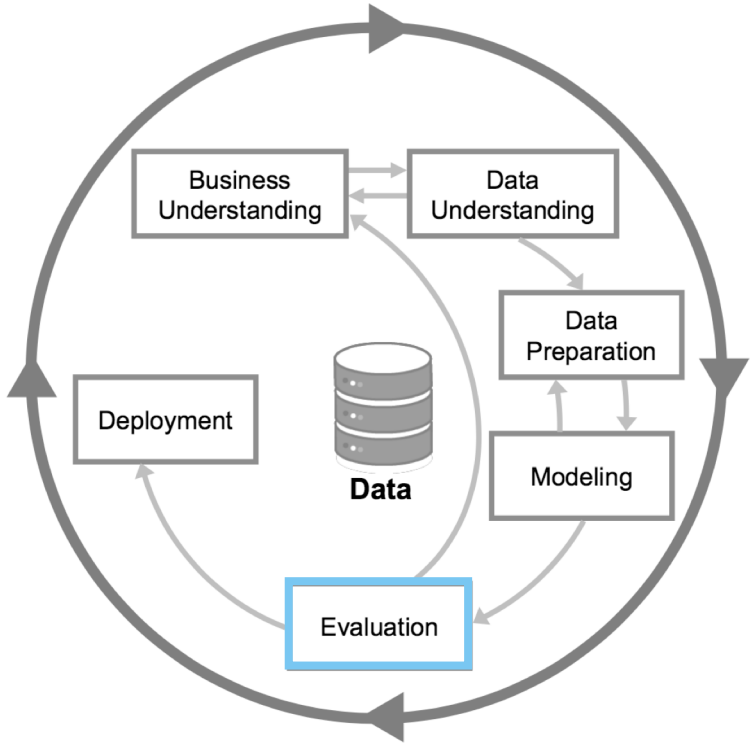
For the details of the most popular algorithms, please check it out later on

Popular Algorithm Chapter

But now let's take a look at how to evaluate the model first

06 | Model Evaluation

:: Evaluate the model



Evaluate and interpret results

- Evaluate Performance
- Optimize parameters
- Interpret results

Source: https://www.the-modeling-agency.com/crisp-dm.pdf

:: What's a good model ?

- **Accurate**

  Are we making good predictions ?

- **Interpretable**

  How easy is it to explain how the predictions are made ?

- **Fast**

  How long does it take to build a model and how long does the model take to make predictions?

- **Scalable**

  How much longer do we have to wait if we build/predict using a lot more data ?

Source: Microsoft, https://channel9.msdn.com/Events/OpenSourceTW/DevDays-Asia-2017/AI11

:: Does your model make good predictions in real situations?

Training Phase

Applying Phase

The model making perfect fitting during training phase might perform very poorly in practice.

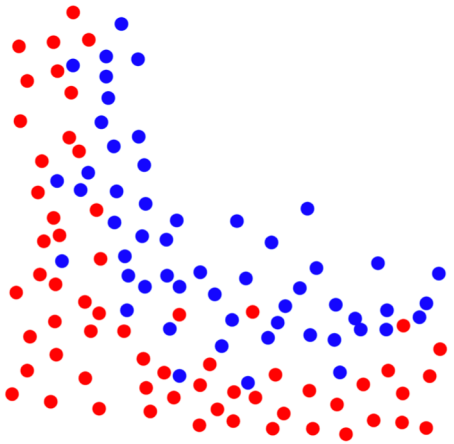
The prefect approximation in training phase might be a overfitting problem.

:: The model might not generalize well to unseen new data

A model's ability to generalize to new data is crucial for the success of a model

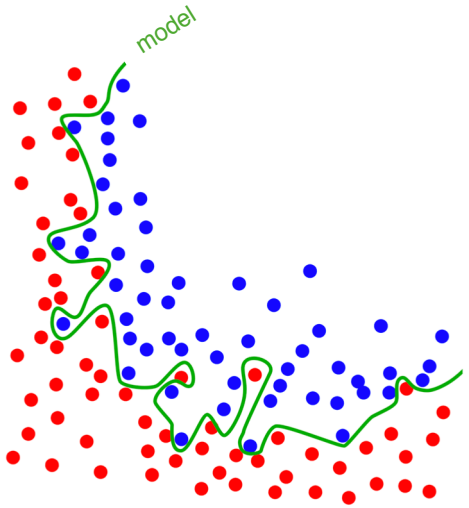
Generalization refers to a model's ability to perform well on new unseen data rather than only the training data. If the model is trained too well, it can fit perfectly the random fluctuations or noise in the training data but it will fail to predict accurately on new data.

Training Data



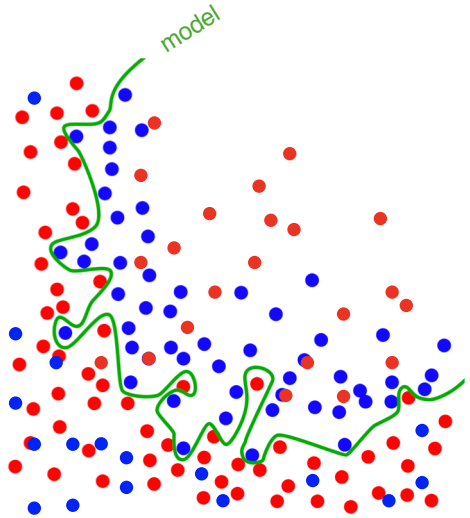
Train a model to classify this data set as "blue" and "red" category

Train a model



During training model learns perfectly the inherent patterns in data. However, If a model has been trained too well on training data, it will be unable to generalize.

Model makes bad prediction on <u>new data</u>

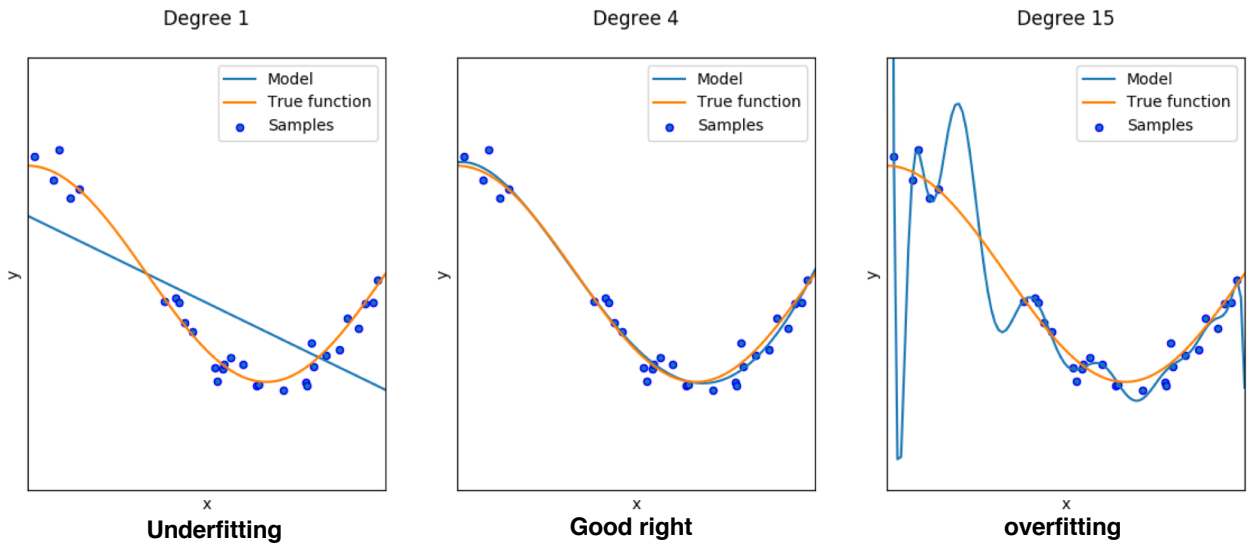


The "perfect" model performs badly on never-seen new data. This is called as "overfitting". If the model is overfitting then it will not generalize well.

:: About the model fitting

Underfitting vs. Overfitting

- When a model fails to capture important distinctions and patterns in the data, so it performs poorly even in training data, that is called **underfitting**. [1] Underfitting is often a result of an excessively simple model. [2]
- The **overfitting** model performs perfectly on training data but fail to predict well on new data. In other words, it has low/no training error but has high test error.
- Both overfitting and underfitting lead to poor predictions on new data sets [3]



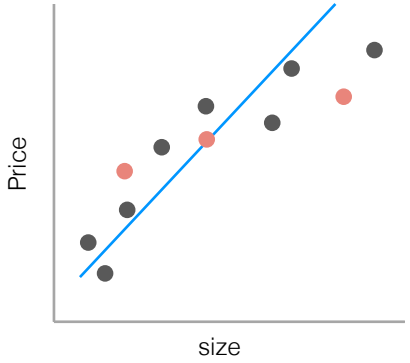
This example demonstrates the problems of underfitting and overfitting and how we can use linear regression with polynomial features to approximate nonlinear functions.

The models have polynomial features of different degrees. We can see that a linear function (polynomial with degree 1) is not sufficient to fit the training samples. This is called underfitting. A polynomial of degree 4 approximates the true function almost perfectly. However, for higher degrees the model will overfit the training data, i.e. it learns the noise of the training data.[4]

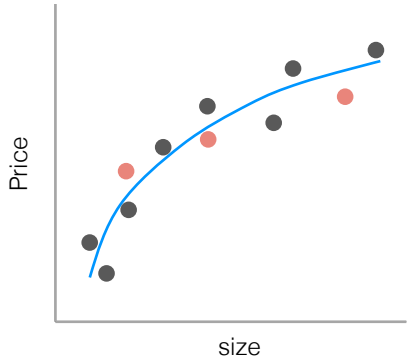
1 Sourcde: 1 Source: https://www.kaggle.com/dansbecker/underfitting-overfitting-and-model-optimization
2, 3 Source: https://chemicalstatistician.wordpress.com/2014/03/19/machine-learning-lesson-of-the-day-overfitting-and-underfitting/
4 Source: https://www.kaggle.com/dansbecker/underfitting-overfitting-and-model-optimization

:: About the model fitting

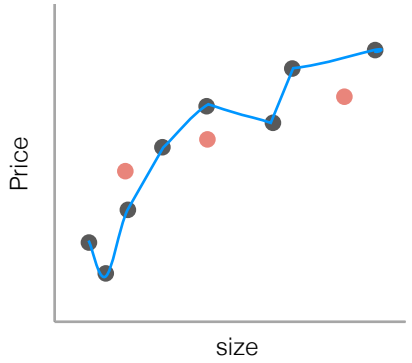
Regression example



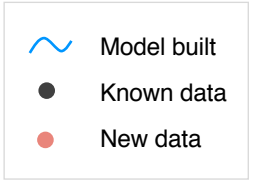
Underfitting model
(high training error, high test error)



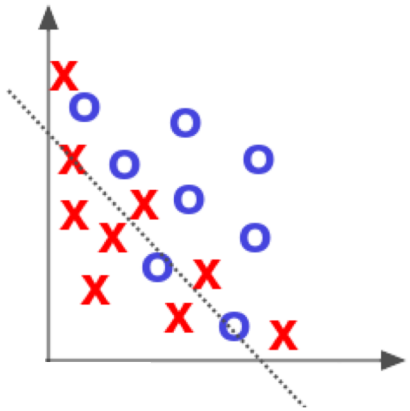
Good Fitting
(low training error, low test error)



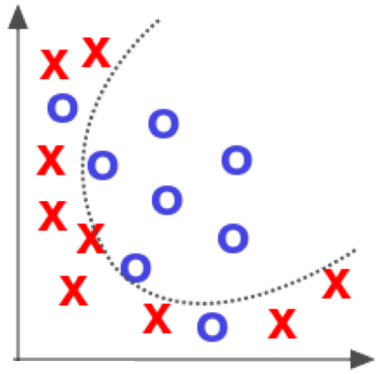
Overfitting Model
(no training error, high test error)



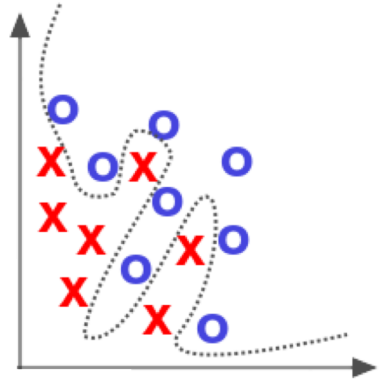
Classification example



Underfitting model
(high training error, high test error)



Good Fitting
(low training error, low test error)



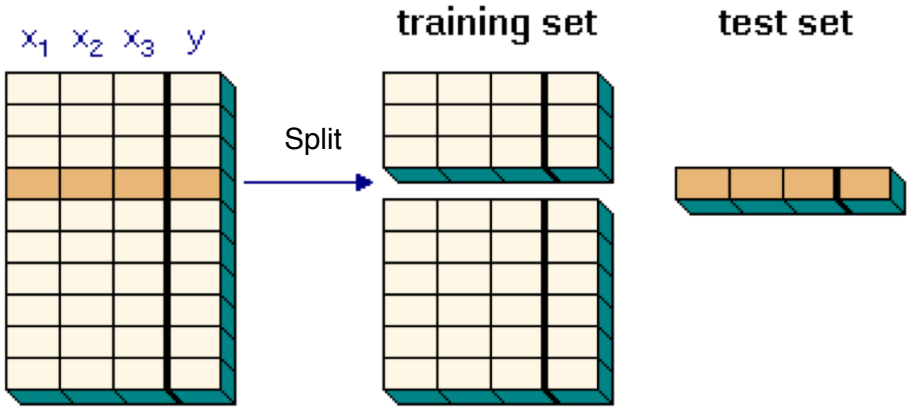
Overfitting Model
(no training error, high test error)

:: Model validation checks how well a model generalizes to new data

Reserve or split a portion of data to validate the model's performance

The fundamental goal of ML is to generalize beyond the data instances used to train models. We want to evaluate the model to estimate the quality of its pattern generalization for data the model has not been trained on[1].

The only way to know how well a model will generalize to new cases is to actually try it out on new cases. It is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a **test set.** You train your model using the training set, and you test it using the test set. The error rate on new cases is called the generalization error, and by evaluating your model on the test set, you get an estimation of this error. This value tells you how well your model will perform on instances it has never seen before.



split the data into training set and test set

1 source: http://docs.aws.amazon.com/machine-learning/latest/dg/splitting-types.html
2 source: Hands-On Machine Learning with Scikit-Learn & TensorFlow
Picture source: http://www.statistics4u.info/fundstat_eng/cc_cross_validation.html

:: Model validation strategy

Common validation strategies

Hold-out validation

K-fold
cross validation

Leave-one-out
cross validation

:: Model validation strategy

Common validation strategies

Hold-out validation

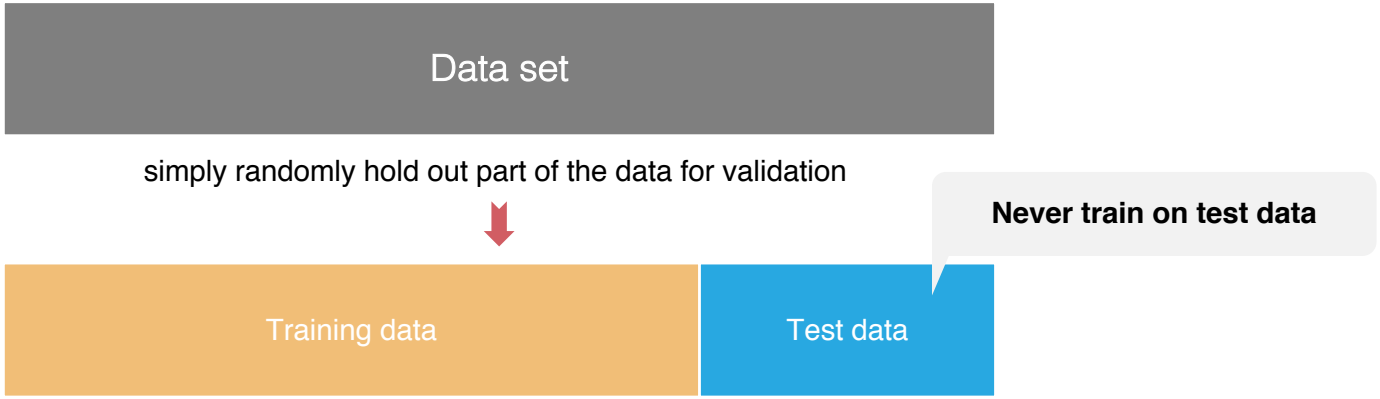
K-fold
cross validation

Leave-one-out
cross validation

:: Model validation strategy : Hold-out validation

Split the data into 2 parts - a training set , test set

The Hold-out validation is a type of simple validation.



The ML system evaluates predictive performance by comparing predictions on the evaluation data set/test data set with true values (known as ground truth) using a variety of metrics.

However, its evaluation can have a high variance. The evaluation may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made[2].

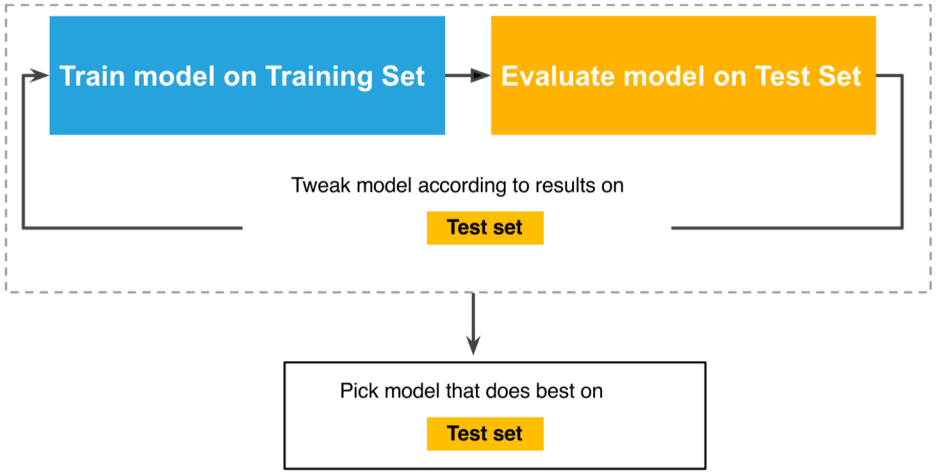
If you are seeing surprisingly good results on your evaluation metrics, it might be a sign that you are accidentally training on the test set. For example, high accuracy might indicate that test data has leaked into the training set.[3]

1 Source: https://www.youtube.com/watch?v=e0JcXMzhtdY&t=812s
2 Source: https://www.cs.cmu.edu/~schneide/tut5/node42.html
3 source: https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data

:: Model validation strategy : Hold-out validation

Using only two partitions may be insufficient when doing many rounds of hyperparameter tuning

The previous slide introduced partitioning a data set into a training set and a test set. This partitioning enabled you to train on one set of examples and then to test the model against a different set of examples. With two partitions, the workflow could look as follows: [1]



"Tweak model" means adjusting anything about the model you can dream up—from changing the learning rate, to adding or removing features, to designing a completely new model from scratch. At the end of this workflow, you pick the model that does best on the *test set*. [2]

However, this 2 partitions approach is not recommended although it's widely introduced in many books.

This approach will not be a fair estimate of how well my hypothesis generalizes

Because we fit the parameters using the test set and it gave us the best possible performance on the test set, that's likely to be an overly optimistic estimate of generalization error. So It's no longer fair to evaluate my hypothesis on this test set.[3]

1, 2, & image Source: https://developers.google.com/machine-learning/crash-course/validation/another-partition
3 source: Andrew Ng, Model Selection And Train Validation Test Sets, https://www.youtube.com/watch?v=uoTBdCODGvk

:: Model validation strategy : Hold-out validation - *continued*

Better approach: Split the data into a training, cross validation and test set

Instead of 2 partitions of data, split data set into 3 parts. Use your cross validation data to select the model and evaluate it on the test data. This approach is recommended by Andrew Ng..

simply **randomly** hold out part of the data for cross validation & test

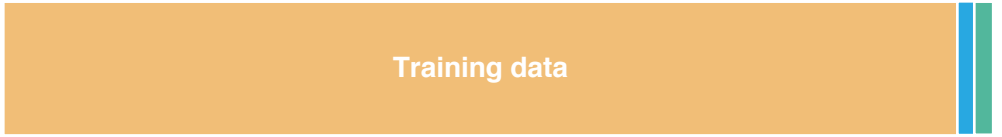
For relatively small data set:

Usually 60% 20% 20%

This traditional ratio is best practice for small data set . For example, it's good for the smaller data set . For example, 1000, or 10,000 samples.

Video: Model Selection And Train Validation Test Sets (12mins)
https://www.youtube.com/watch?v=uoTBdCODGvk

For big data set:



98% 1% 1%

The traditional splitting ratio (60%-20%-20%) is no longer recommended in big data era. For example, you might have a million samples in total. Then , you can use 98% of the data as training data, and 1% as cross validation data, 1% as test data.

:: Model validation strategy

Common validation strategies

Hold-out validation

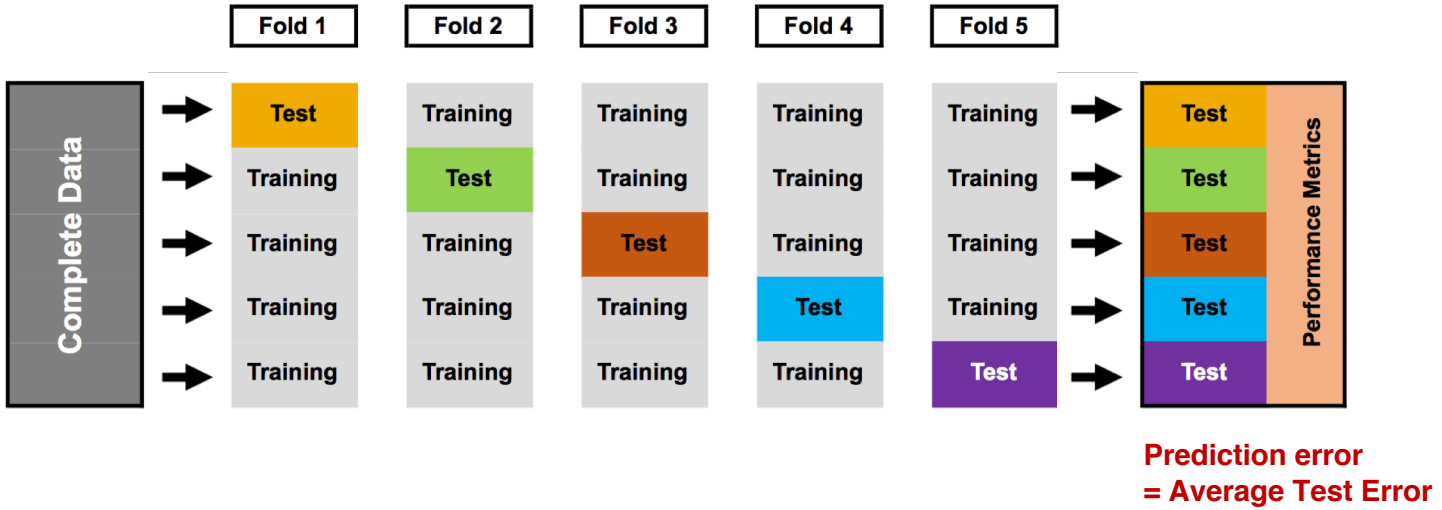
K-fold cross validation

Leave-one-out cross validation

:: Model validation strategy : k-fold cross validation

K-fold cross validation: the data set is divided into k equal size subsets.

The data set is divided into *k* subsets, and the Hold-out validation is repeated *k* times. Each time, one of the *k* subsets is used as the test set and the other *k-1* subsets are put together to form a training set. Then the average error across all *k* trials is computed.



The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set *k-1* times. The variance of the resulting estimate is reduced as *k* is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch *k* times, which means it takes *k* times as much computation to make an evaluation.

1 Source: https://www.youtube.com/watch?v=e0JcXMzhtdY&t=812s
2 & image Source: https://www.cs.cmu.edu/~schneide/tut5/node42.html

:: Model validation strategy

Common validation strategies

Hold-out validation

K-fold
cross validation

**Leave-one-out
cross validation**

:: Model validation strategy : Leave-one-out cross validation (LOOCV)

Leave-one-out cross validation: specific case of k-fold cross validation, where k = n

Leave-one-out cross validation is K-fold cross validation taken to its logical extreme, with K equal to N, the number of data points in the set. Thus, the learning algorithm is applied once for each instance, using all other instances as a training set and using the selected instance as a single-item test set.

- use one observation as the validation set and and the remaining observations as the training set
- This is repeated such that each observation in the sample is used once as the validation data.
- LOOCV is computationally intensive !

2 Source: https://www.cs.cmu.edu/~schneide/tut5/node42.html

:: Which validation strategy should we use ?

- Hold-out validation is simpler testing and computationally cheaper.
- Hold-out strategy is suitable for deep learning where the amount of data is huge.

However, the old way of splitting data like 80% vs 20% is no longer valid in deep learning case.

→ *Youtube video (5min): https://www.youtube.com/watch?v=_Fe5kKmFieg*

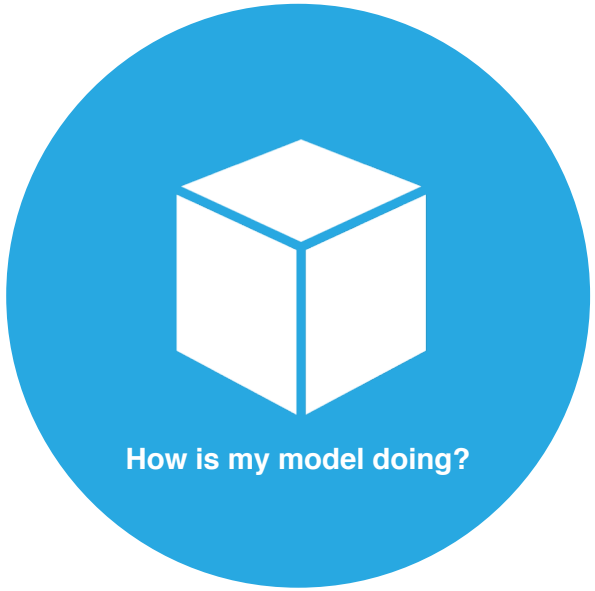
- Cross-validation is useful when the dataset is small.
- In small dataset cases not much training data can be held out to reliably estimate test performance
- 10-fold cross validation is common, but smaller values of k are often used when learning takes a lot of time [2].

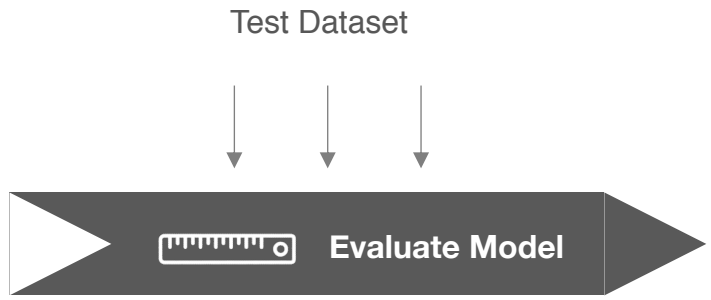
" We mostly have large datasets when it is not worth the trouble to do something like k-fold cross-validation. We just use a train/valid/test split. Cross-validation becomes useful when the dataset is tiny (like hundreds of examples), but then you can't typically learn a complex model." [1]

- Yoshua Bengio

1 Source: https://www.quora.com/Is-cross-validation-heavily-used-in-deep-learning-or-is-it-too-expensive-to-be-used
2 Source: http://pages.cs.wisc.edu/~dpage/cs760/evaluating.pdf

:: Evaluate the model's performance





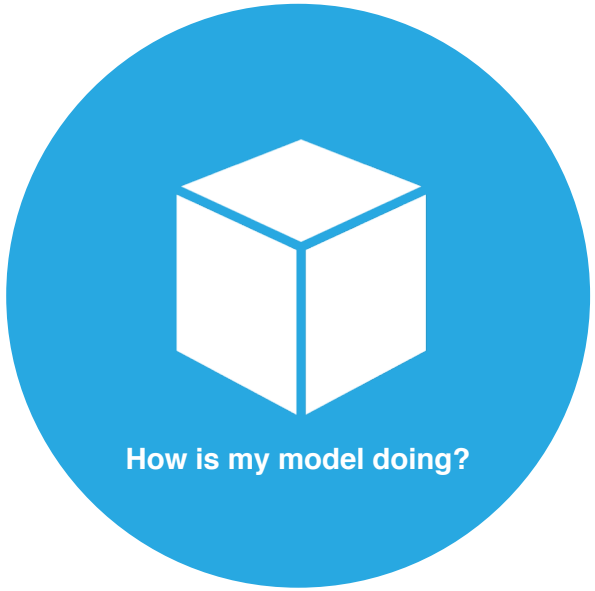
Model evaluation measures the quality of the machine learning model and determines how well our machine learning model will generalize to predict the target on new and future data.

Because future instances have unknown target values, you need to check the accuracy metric of the ML model on data for which you already know the target answer, and use this assessment as a proxy for predictive accuracy on future data [1].

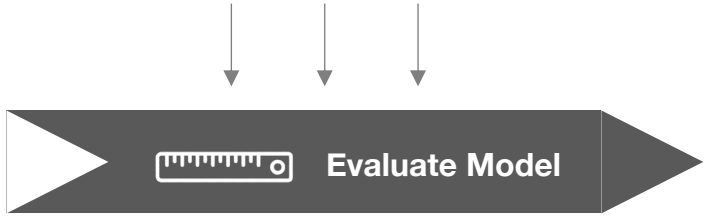
Evaluate your trained model by using validation/test dataset. You compare the results of your model's predictions to the target values in the evaluation data and use statistical techniques appropriate to your model to gauge your success.

1 Source: http://docs.aws.amazon.com/machine-learning/latest/dg/evaluating_models.html

:: Evaluate the model's performance



Test Dataset



We need criteria or metrics to evaluate the performance of model

- Accuracy
- Precision
- Recall
- F score
- ROC
- AUC
- Log Loss
- MAE (Mean absolute error)
- MSE (Mean Squared Error)
- RMSE (Root means squared error)
- MAPE (Mean absolute % error)
- R^2 (Coefficient of determination)

1 Source: http://docs.aws.amazon.com/machine-learning/latest/dg/evaluating_models.html

:: Model Evaluation Metrics

Different machine learning tasks have different performance metrics

Classification Model

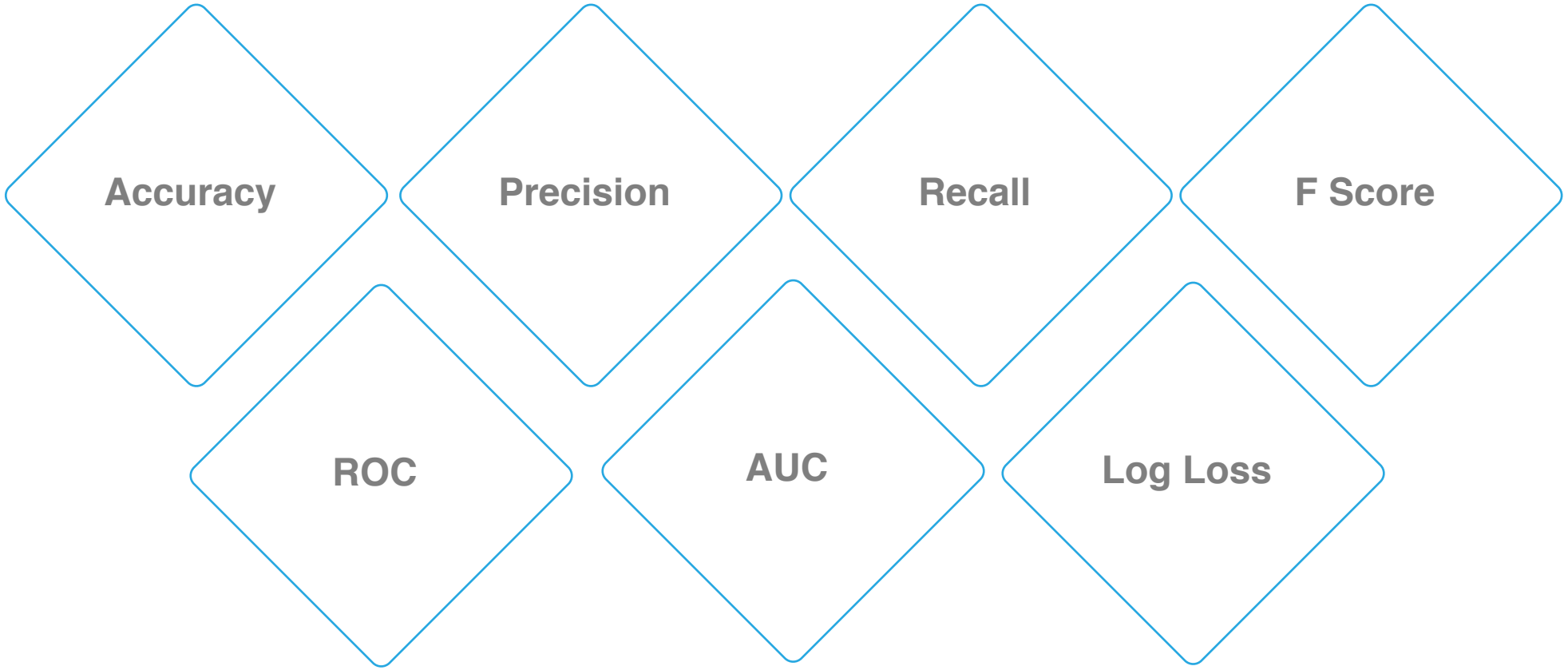
- Accuracy
- Precision
- Recall
- F score
- ROC
- AUC
- Log Loss

Regression Model

- MAE (Mean absolute error)
- MSE (Mean Squared Error)
- RMSE (Root means squared error)
- MAPE (Mean absolute % error)
- R^2 (Coefficient of determination)

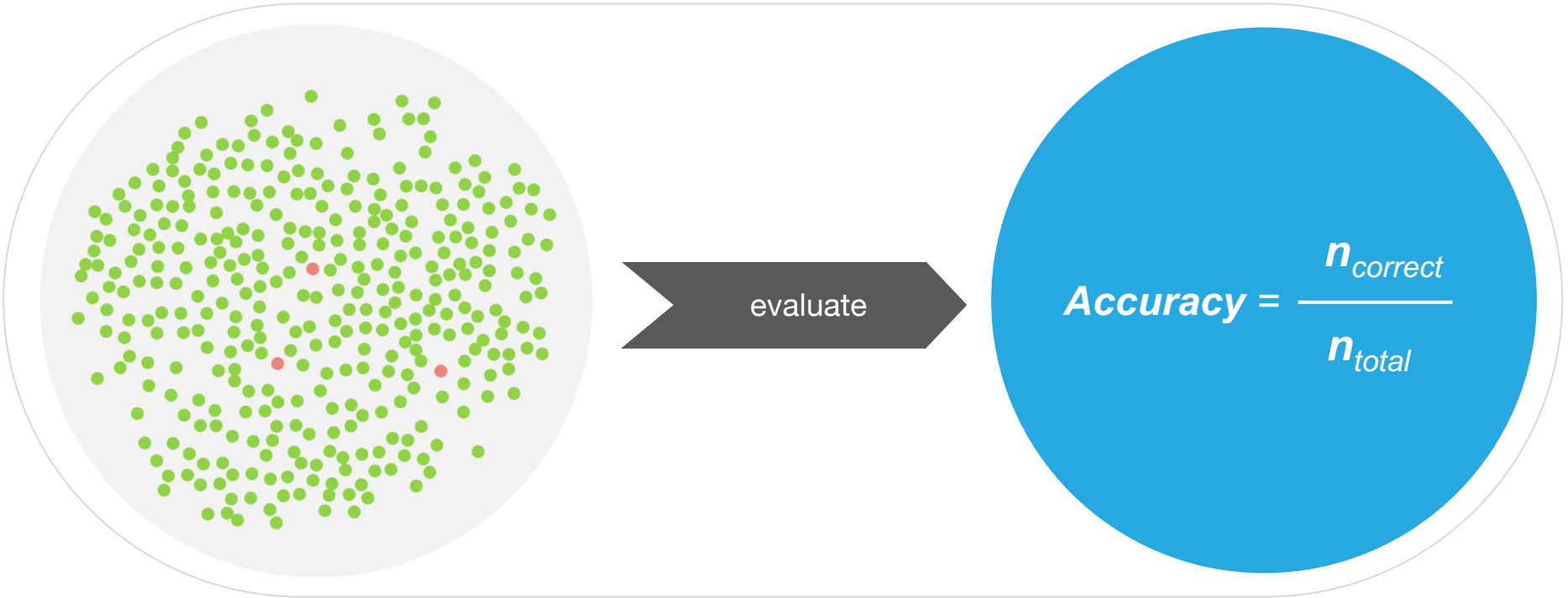
evaluation metrics explain the performance of a model

:: Metrics for evaluating classification model



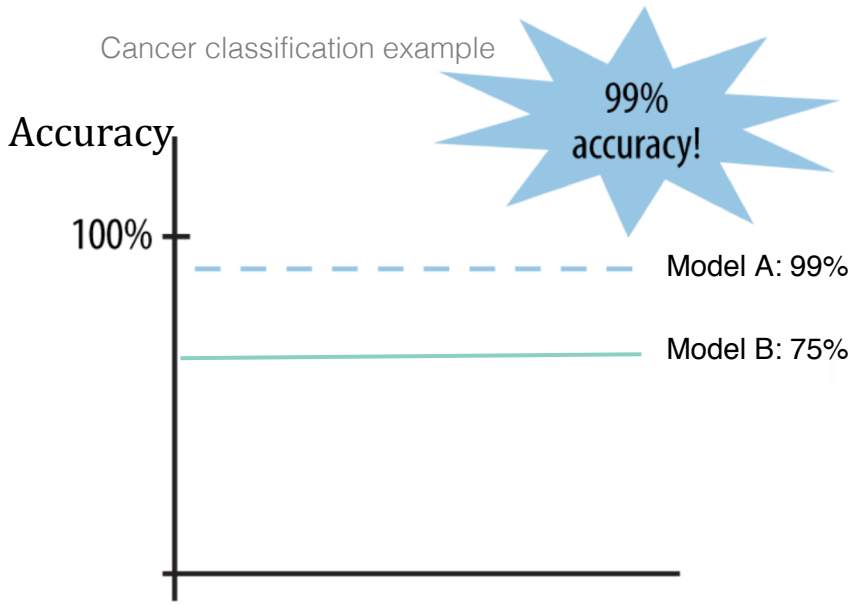
:: What's the accuracy?

Accuracy measures the ratio of correct predictions to the total number of cases evaluated.



:: Accuracy is not a good metric for Skewed Datasets

A predictive model may have high accuracy, but be useless [1]



Suppose the positive class is only a tiny portion of the observed data. For example, only 1% of patients has true cancer while other 99% of patients don't have any cancers. This is a typical case about data skew.

For such imbalanced classes, suppose you build two models.

- Model A would achieve 99% accuracy by always predicting "no cancer".
- Model B has 75% accuracy. It correctly predict some cancer classes but also make some misclassification.

Is Model A better than Model B ?

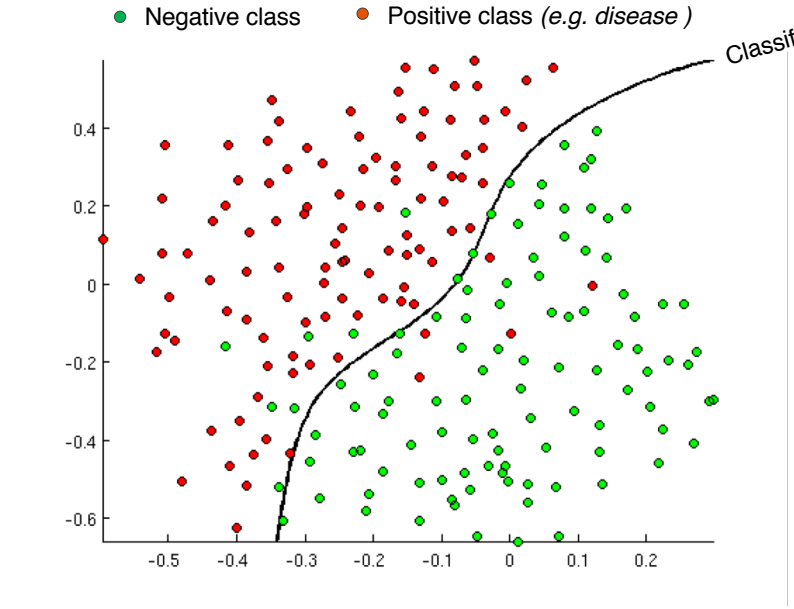
NO ! Model A always classifies incoming data as negative classes (i.e. no cancer) can achieve 99% accuracy but can't identify any cancer cases, therefore it is useless.

Using accuracy is only good for symmetric data sets where the class distribution is 50/50 and the cost of false positives and false negatives are roughly the same[3].

1 source: https://en.wikipedia.org/wiki/Accuracy_paradox

3 source: https://blogs.msdn.microsoft.com/andreasderuiter/2015/02/09/performance-measures-in-azure-ml-accuracy-precision-recall-and-f1-score/

:: Confusion matrix visually shows more details of the error



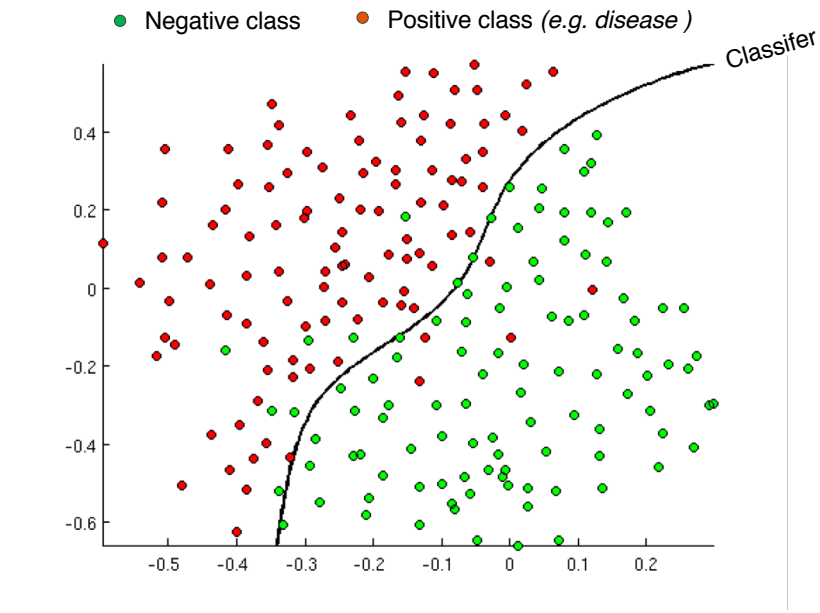
Binary classification example

In this example, the model correctly classifies many classes but also makes some mistakes:

- some positive classes are wrongly classified as negative values.
- some negative classes are wrongly classified as positive classes

How to summarize the the prediction result ?

:: Confusion matrix visually shows more details of the error



Binary classification example

In this example, the model correctly classifies many classes but also makes some mistakes:

- some positive classes are wrongly classified as negative values.
- some negative classes are wrongly classified as positive classes

Display the right and wrong predictions in the table

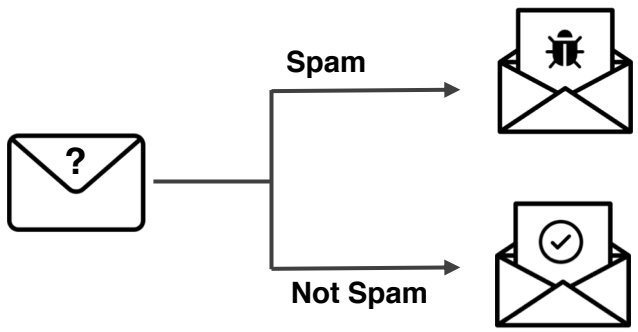
Confusion matrix		Predicted Class: Predicted Value : **Positive** (+)	Predicted Class: Predicted Value **Negative** (-)
Actual Class	Actual Value : **Positive** (+)	**TP** True Positive	**FN** False Negative
Actual Class	Actual Value : **Negative** (-)	**FP** False Positive	**TN** True Negative

Confusion Matrix

It is a technique to summarize the performance of classification model.
It displays the number of correct and incorrect predictions made by the model compared with the actual classifications in the test data. Main values of this matrix:

- **True Positive** - we predicted "+" and the true class is "+"
- **True Negative** - we predicted "-" and the true class is "-"
- **False Positive** - we predicted "+" and the true class is "-" (Type I error)
- **False Negative** - we predicted "-" and the true class is "+" (Type II error)

:: Case study: spam detection



Junk email detection is a binary classification problem. A model is trained to classify the email as “spam” or “not spam”.

Confusion matrix is used to compare the value predicted by the model with the actual value. In this example, there are :

- 12,987 TP (correct positive prediction)
- 98,356 TN (correct negative prediction)
- 358 FP (They are false positive. We predicted them as “spam”, but they are not spam.)
- 567 FN (we predict them as negative value (i.e. not spam), but they actually are spam)

		Predicted Class: Predicted Value : **Spam (+)**	Predicted Class: Predicted Value **Non-spam(-)**
Actual Class	Actual Value : **Spam (+)**	**TP** True Positive	**FN** False Negative
	Actual Value : **Non-spam(-)**	**FP** False Positive	**TN** True Negative

Count the number of instances

		Predicted Class: Predicted Value : **Spam (+)**	Predicted Class: Predicted Value **Non-spam(-)**
Actual Class	Actual Value : **Spam (+)**	**TP** 12,987	**FN** 567
	Actual Value : **Non-spam(-)**	**FP** 358	**TN** 98,356

correct classification

wrong classification (Type II error)

wrong classification (Type I error)

correct classification

:: Example for Accuracy, Precision, Recall

Accuracy

$$Accuracy = \frac{TP+TN}{TP+FN+FP+TN} = \frac{\text{Correct predictions}}{\text{Total data points}}$$

Precision

$$Precision = \frac{TP}{TP+FP} = \frac{\text{Correctly Predicted Positive}}{\text{All Predicted Positive}}$$

Precision : the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) is divided by the total number of elements predicted as belonging to the positive class

Recall

$$Recall = \frac{TP}{TP+FN} = \frac{\text{Correctly Predicted Positive}}{\text{All Real Positive}}$$

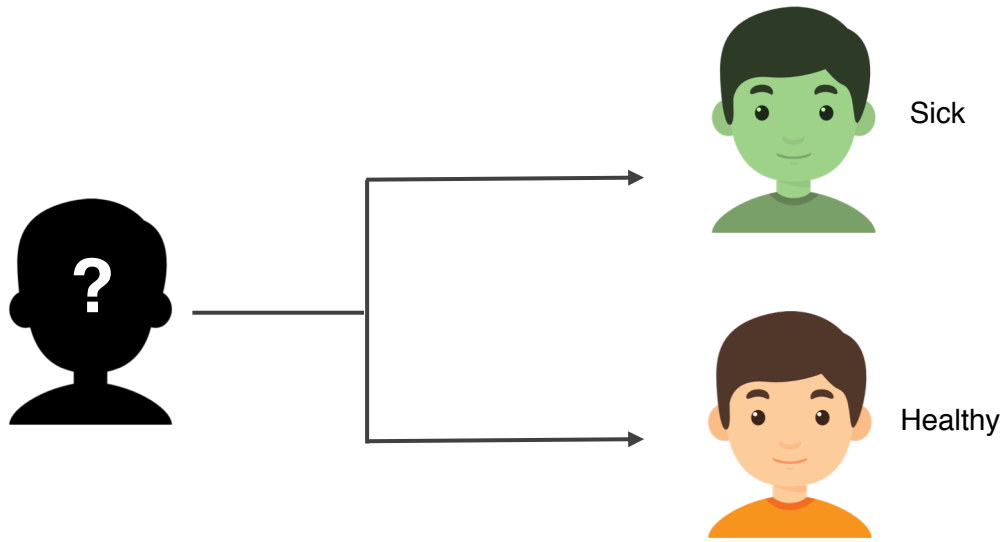
Recall : number of true positives is divided by the total number of elements that actually belong to the positive class

Example

		Predicted Class: Predicted as **Spam** (+)	Predicted Class: Predicted as **Non-spam**(-)
Actual Class	Actual = **Spam** (+)	**True Positive** 2,344	**False Negative** 5,567
	Actual = **Non-spam**(-)	**False Positive** 358	**True Negative** 18,356

- **Accuracy** = (TP+TN) / (TP+FN+FP+TN)
  Accuracy = (2344+18536)/(2344+5567+358+18536)= 77%
- **Precision** = True Positives / (True Positives + False Positives)
  Precision = 2344/(2344+358)= 86%
- **Recall** = True Positives / (True Positives + False Negatives)
  Recall = 2344/(2344+5567)= 29%

:: Case study: Medical diagnosis



A model is trained to diagnose the people as “sick” or “healthy”.

Confusion matrix is used to compare the value predicted by the model with the actual value. In this example, there are :

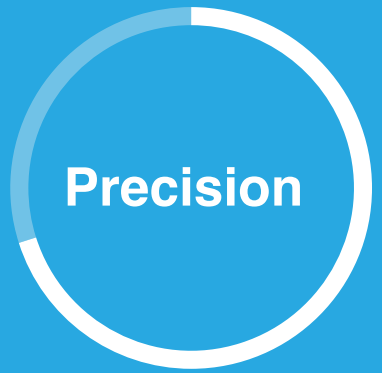
- 12,987 TP (correct positive prediction)
- 98,356 TN (correct negative prediction)
- 358 FP (False Positive)
- 567 TN (False Negative)

Example

		Predicted Class: Diagnosed as **Sick** (+)	Predicted Class: Diagnosed as **Healthy** (-)
Actual Class	Actual = **Sick** (+)	**True Positive** 967	**False Negative** 588
Actual Class	Actual = **Healthy** (-)	**False Positive** 1,958	**True Negative** 7,356

- **Accuracy** = (TP+TN) / (TP+FN+FP+FN)
  Accuracy = (967+7356)/(967+7356+1958+588) = 76.5%
- **Precision** = True Positives / (True Positives + False Positives)
  Precision = 967/(967+1958) = 33%
- **Recall** = True Positives / (True Positives + False Negatives)
  Recall = 967/(967+588) = 62%

:: Precision vs Recall



Precision attempts to answer the following question:

What proportion of positive identifications was actually correct? [1]

$$precision = \frac{\#\ True\ Positive}{\#\ True\ Positive + \#\ False\ Positive}$$

No. of <u>predicted</u> positive

Note that the denominator is the count of all data points predicted as positive, including positive observations which were in fact negative.

Recall attempts to answer the following question:

What proportion of actual positives was identified correctly? [2]

$$recall = \frac{\#\ True\ Positive}{\#\ True\ Positive + \#\ False\ Negative}$$

No. of <u>actual</u> positive

Note that the denominator is the count of all actual/real positive data points. They are all the real positive classes although some of them might be misclassified as negative.

1,2 Source: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

:: About the cost of prediction error

The model can make mistakes

Confusion Matrix		Predicted Class: Predicted Value : Positive (+)	Predicted Class: Predicted Value Negative (-)
Actual Class	Actual Value : Positive (+)	**TP** True Positive	**FN** False Negative
Actual Class	Actual Value : Negative (-)	**FP** False Positive	**TN** True Negative

Type II error

Type I error

The cost of wrong predictions

It is important to realize there are two types of errors – false positives and false negatives - which often have a different associated cost[1].

It will be great if both False Positive and False Negative are low at the same time. However, it's not ideal in real situation.

Which Is Worse, False Positive or False Negative?

It depends on the business cases.

Medical Diagnosis

Early diagnosis of malignant tumor is very important . Not finding malignant tumor is worse than misclassifying benign tumor as malignant one.

High false negative result means low probability of detecting of diseases and therefore the patient missed the opportunity for treatment.

Spam Detection

Higher false negative rate means there might be more junk mails. Although more junk emails are annoying, but it's better than missing the important email from your business partner.

The cost of misclassifying normal email as spam is higher than the cost of failing to detect spam. False positive is much worse than false negative in this case.

1Source: https://blogs.msdn.microsoft.com/andreasderuiter/2015/02/09/performance-measures-in-azure-ml-accuracy-precision-recall-and-f1-score/

Source of picture: designed by Freepik.com, Flaticon.com

:: Trade-off between precision and recall

Often, there is an inverse relationship between precision and recall

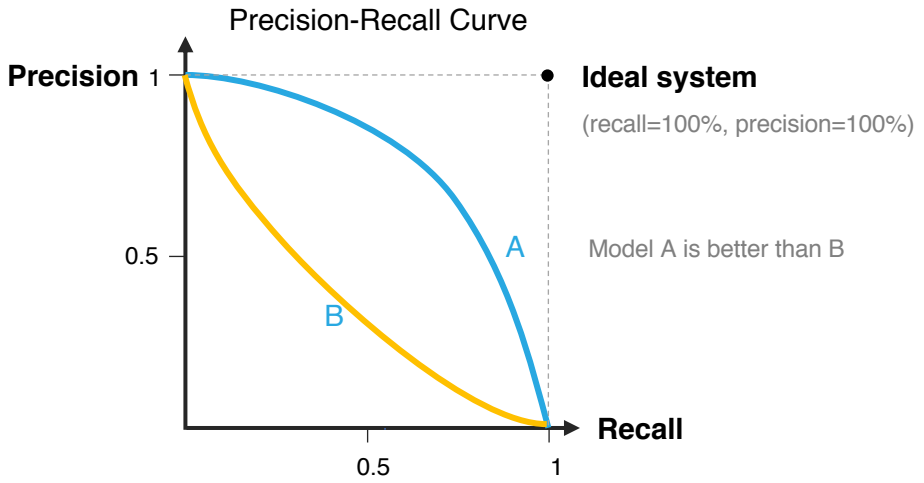
Increasing precision reduces recall, and vice versa. This is called the precision/recall tradeoff. [1]

- **Within any one model, you can decide to emphasize either precision or recall.**

  Which one is more important depends on the business needs. In some contexts you mostly care about precision, and in other contexts you really care about recall.

- **You can influence precision and recall by changing the threshold of the model.**

  When you select a lower threshold value/cut-off value, then the recall will increase. On the other hand the false positive fraction will also increase, and therefore the true negative fraction /specificity will decrease



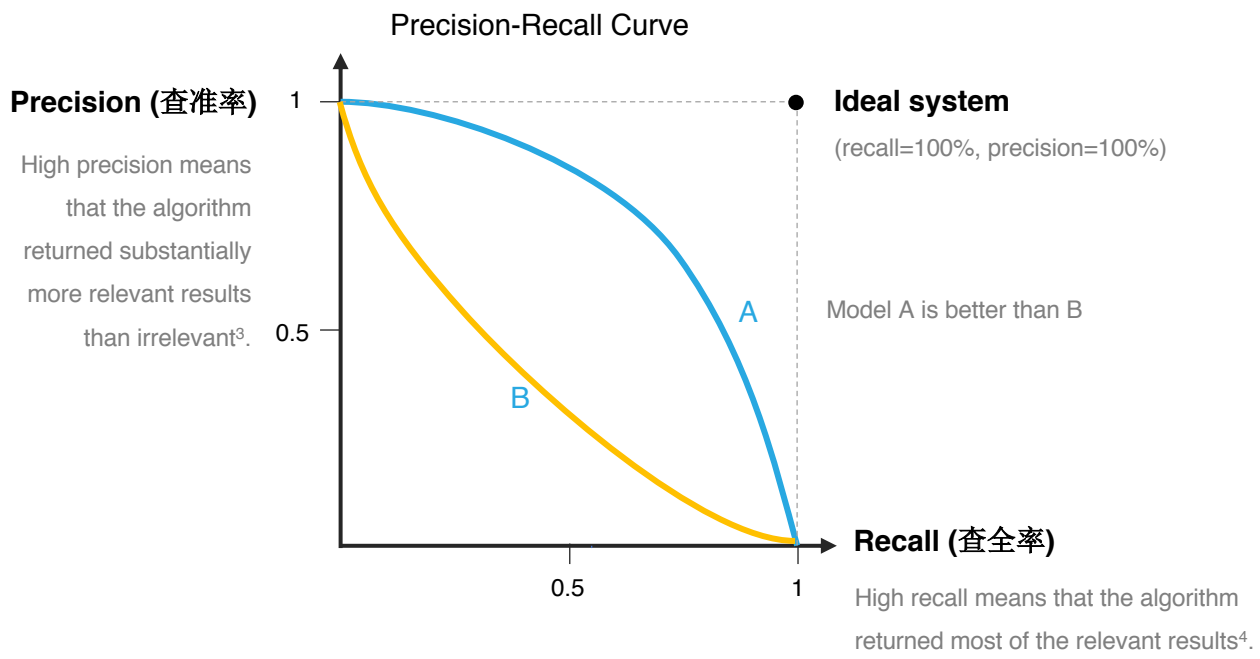
- High precision means that the algorithm returned substantially more relevant results than irrelevant [2].

- High recall means that the algorithm returned most of the relevant results [3].

1 Source: https://en.wikipedia.org/wiki/Precision_and_recall
2&3 source: https://channel9.msdn.com/Events/OpenSourceTW/DevDays-Asia-2017/AI11

:: Trade-off between precision and recall

Change the threshold of a classifier



Let's say we trained a Logistic Regression classifier

- we predict 1 if $h\theta(x) \geqslant 0.5$
- we predict 0 if $h\theta(x) < 0.5$

Case 1: when precision is more important

Suppose we want to predict y = 1 (i.e. people have cancer) only if we're very confident. We may change the threshold to 0.7

- we predict 1 if $h\theta(x) \geqslant 0.7$
- we predict 0 if $h\theta(x) < 0.7$

That leads to

- higher precision in this case (all for who we predicted y = 1 are more likely to actually have it).
- But lower recall (we'll miss more patients that actually have cancer, but we failed to spot them)

Case 2: when recall is more important

Suppose we want to avoid missing too many cases of y=1 (i.e. we want to avoid false negatives). We may change the threshold to 0.3

- we predict 1 if $h\theta(x) \geqslant 0.3$
- we predict 0 if $h\theta(x) < 0.3$

That leads to

- Higher recall (we'll correctly flag higher fraction of patients with cancer)
- Lower precision (and higher fraction will turn out to actually have no cancer)

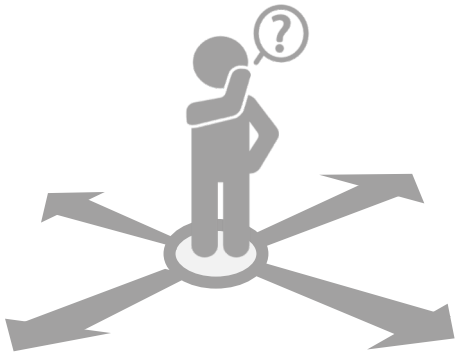
Source: http://mlwiki.org/index.php/Precision_and_Recall

:: Metric: F score

How to compare precision/recall numbers and decide which algorithm is better ?

	Precision(P)	Recall(R)	**Which is best?**
Algorithm 1	0.5	0.4	?
Algorithm 2	0.7	0.1	?
Algorithm 3	0.02	1	?

- By varying threshold parameter we will get different precision and recall
- Improving recall will lead to worse precision
- Improving precision will lead to worse recall
- How to pick the threshold/cut-off value?
- Is there a way to automatically choose the threshold for us?



Which algorithm is better?

- Now have two numbers and need to choose which one to prefer
- F_1 score helps to decide since it's just one number.

Source of table: https://www.youtube.com/watch?v=W5meQnGACGo
Source of text: http://mlwiki.org/index.php/Precision_and_Recall

:: Metric: F score

F score combines precision and recall into one measure

	Precision(P)	Recall(R)	**F_1 Score**
Algorithm 1	0.5	0.4	0.444
Algorithm 2	0.7	0.1	0.175
Algorithm 3	0.02	1	0.0392

- The F_1 score is a good way to summarize the evaluation of performance in a single number. It combines precision and recall into one measure.

- Bigger F_1 score is better. The ideal F_1 score value is 1. The worst value is 0. In this example, Algorithm1 has highest F_1 score, so it's the best one.

A combined measure: F score

$$F_1 = 2\frac{PR}{P+R} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- It is often convenient to combine precision and recall into a single metric called the F1 score, in particular if you need a simple way to compare two classifiers [2].

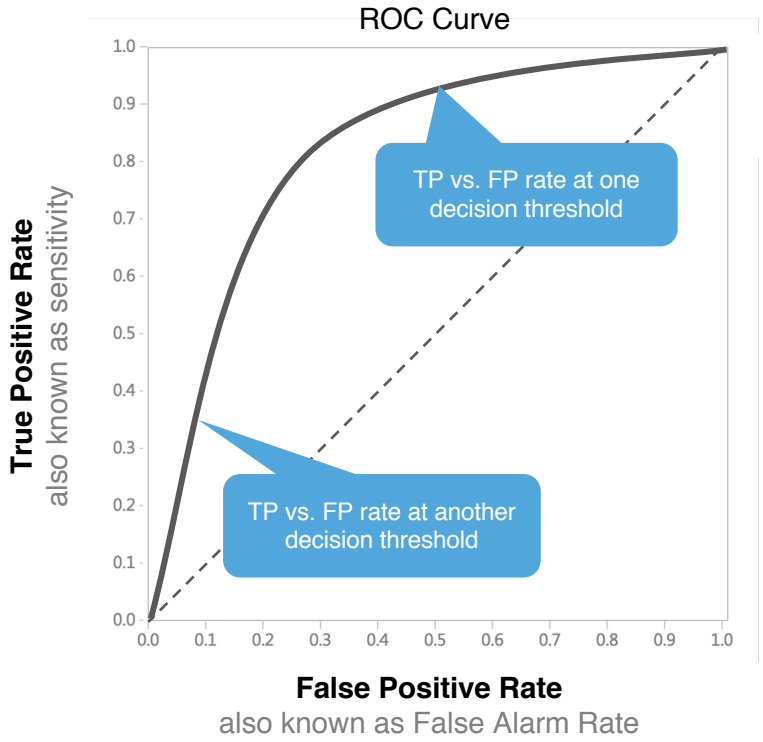
- The F_1 Score is the **weighted average** of Precision and Recall. Therefore, this score takes both false positives and false negatives into account[3] .

1&3 Source: https://blogs.msdn.microsoft.com/andreasderuiter/2015/02/09/performance-measures-in-azure-ml-accuracy-precision-recall-and-f1-score/
2 source: Aurélien Géron , Hands-On Machine Learning with Scikit-Learn & TensorFlow

:: Metric: ROC Curve

ROC curve plots True Positive Rate vs. False Positive Rate at different classification thresholds[1]

ROC Curves shows the tradeoff between true positive rate and false-positive rates of classification algorithms. In other words, ROC shows you how many correct positive classifications can be gained as you allow for more and more false positives[2].



Based on confusion matrix, we can calculate True Positive Rate and False Positive Rate.

- **True Positive Rate (Recall)**
  It is sometimes referred to as the hit rate - what percent of the actual positives does the classifier get right.

$$TPR = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Negative}$$

- **False Positive Rate**
  It is the ratio of negative instances that are incorrectly classified as positive

$$FPR = \frac{\#False\ Positive}{\#False\ Positive + \#True\ Negative}$$

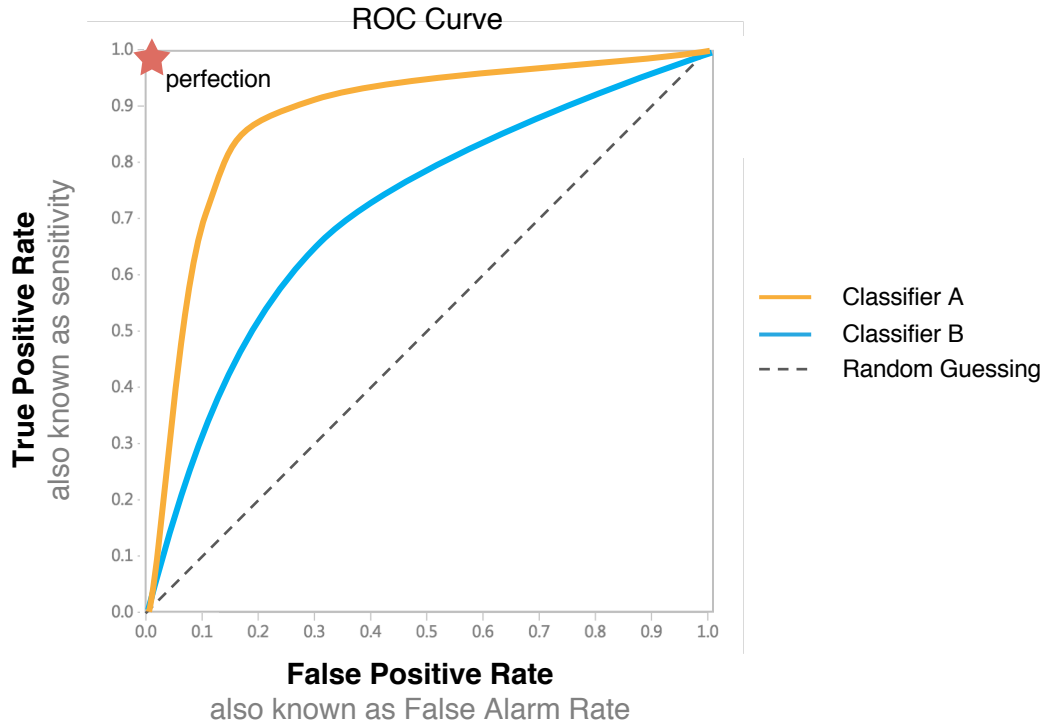
> **YouTube Video: How to plot the ROC curve ? (5mins)**
> https://youtu.be/a6TD9xLXZw0?t=25s

1 https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc
2 Source: Alice Zheng, Evaluating Machine Learning Models

:: Metric: ROC Curve

The closer this curve is to the upper left corner, the better the classifier's performance is

Curves that are close to the diagonal of the plot, result from classifiers that tend to make predictions that are close to random guessing[1]. The perfect classifier that makes no mistakes would hit a true positive rate of 100% immediately, without incurring any false positives—this almost never happens in practice[2].



In this example

- Classifier A is better than B
- Classifier B is better than Random Guessing

Main advantages of ROC:

- ROC Curves are insensitive to class distribution/unbalanced datasets.
- If the proportion of positive to negative instances changes, the ROC Curve will not change.

Note that no classifier should be in the lower right triangle of a ROC graph.
This represents performance that is worse than random guessing[3].

1 Source: https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-evaluate-model-performance
2&3 Source: Alice Zheng, Evaluating Machine Learning Models

:: Metric: AUC (area under the curve)

AUC is (arguably) the best way to summarize model's performance in a single number

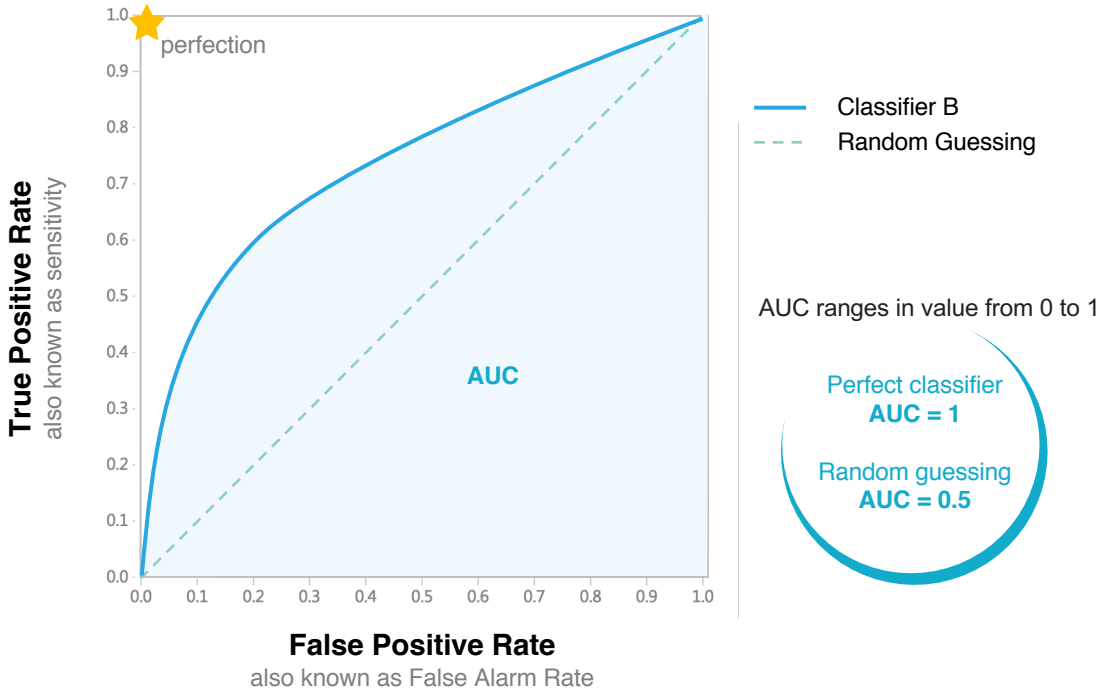
- **Compare multiple models by a single number**

  The ROC curve is not just a single number; it is a whole curve. It provides nuanced details about the behavior of the classifier, but it's hard to quickly compare many ROC curves to each other. In particular, if one were to employ some kind of automatic hyper-parameter tuning mechanism, the machine would need a quantifiable score instead of a plot that requires visual inspection. The AUC is one way to summarize the ROC curve into a single number, so that it can be compared easily and automatically[1].

- **Higher AUC will be the better**

  Using the AUC metric you can quickly compare multiple learning models. when comparing two models, the one with a higher AUC will be the better one regardless of the threshold setting. Compared to the statistical measures of accuracy, precision, recall and F1 score, AUC's independence of threshold makes it uniquely qualified for model selection.

  On the other hand, unlike accuracy, precision, recall and F1 score, AUC does not tell us what performance to expect from the model for a given threshold setting, nor can it be used to determine the optimal value for threshold. In that regard it doesn't take away the need for the other statistical measures[2].

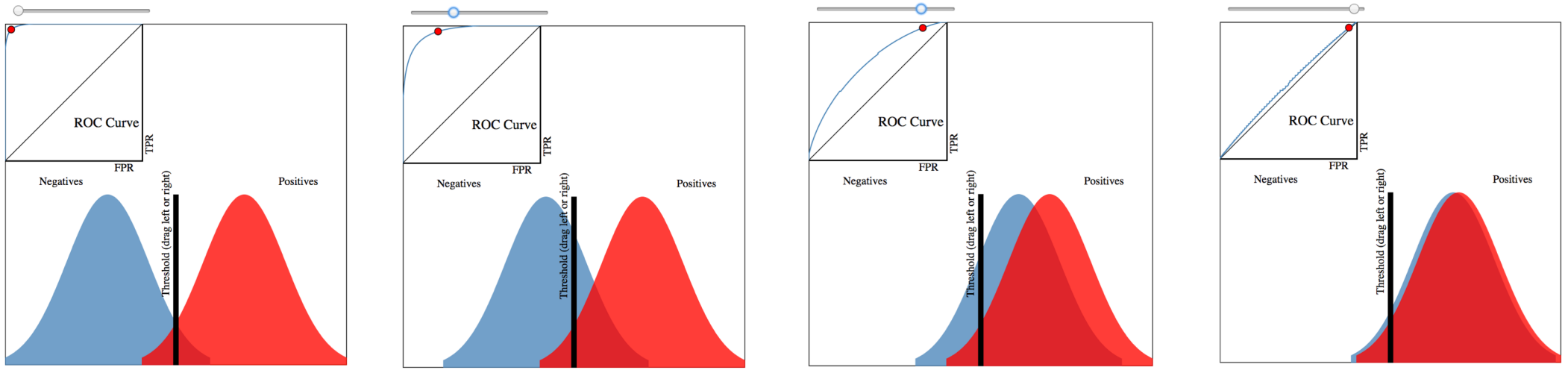


In practice, if you have a "perfect" classifier with an AUC of 1.0, you should be suspicious, as it likely indicates a bug in your model. For example, you may have overfit to your training data, or the label data may be replicated in one of your features.[3]

1 Source: Alice Zheng, Evaluating Machine Learning Models
2 Source: Micosoft: https://blogs.msdn.microsoft.com/andreasderuiter/2015/02/09/using-roc-plots-and-the-auc-measure-in-azure-ml/
3 https://developers.google.com/machine-learning/crash-course/classification/check-your-understanding-roc-and-auc

:: Good demo: ROC curve & AUC

An interactive visualization that may help you better understand ROC curves and AUC

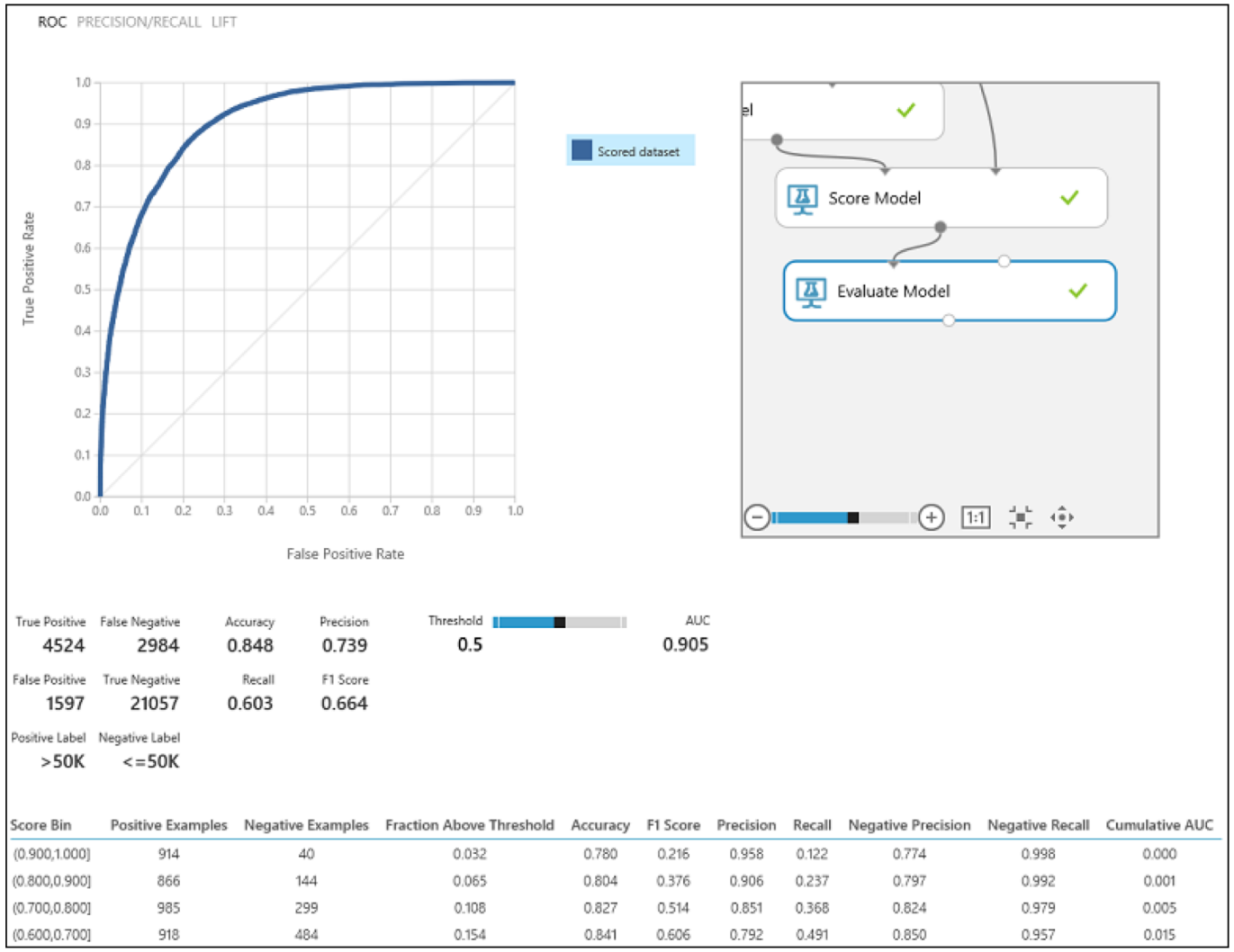


http://www.navan.name/roc

(created by Navan)

:: Sample: classification evaluation result

Binary Classification Evaluation Results



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
4524	2984	0.848	0.739	0.5	0.905

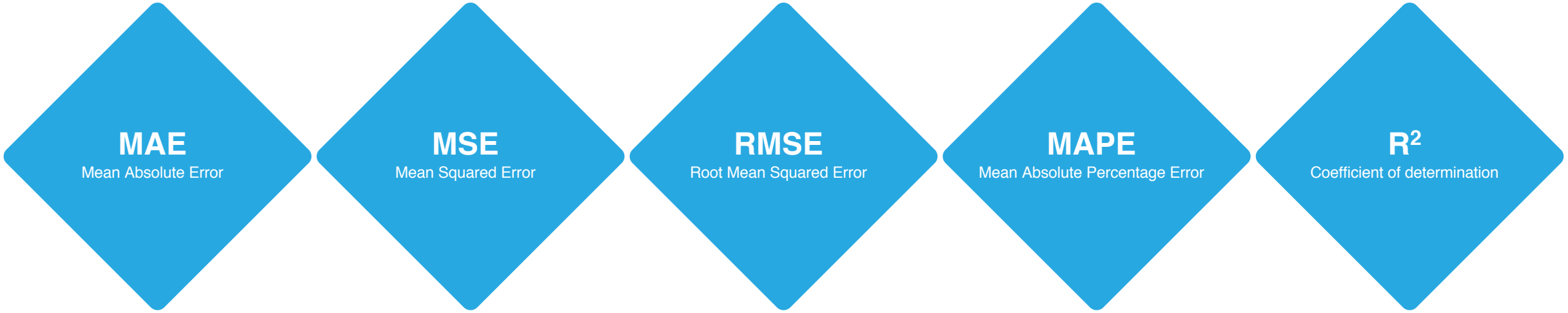
False Positive	True Negative	Recall	F1 Score
1597	21057	0.603	0.664

Positive Label	Negative Label
>50K	<=50K

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	914	40	0.032	0.780	0.216	0.958	0.122	0.774	0.998	0.000
(0.800,0.900]	866	144	0.065	0.804	0.376	0.906	0.237	0.797	0.992	0.001
(0.700,0.800]	985	299	0.108	0.827	0.514	0.851	0.368	0.824	0.979	0.005
(0.600,0.700]	918	484	0.154	0.841	0.606	0.792	0.491	0.850	0.957	0.015

Image source: Microsoft Azure

:: Metrics for evaluating regression model



The metrics returned for regression models are generally designed to estimate the amount of error. A model is considered to fit the data well if the difference between observed and predicted values is small. However, looking at the pattern of the residuals (the difference between any one predicted point and its corresponding actual value) can tell you a lot about potential bias in the model[1].

1 Source: Microsoft, https://msdn.microsoft.com/en-us/library/azure/dn905915.aspx#bkmk_regression

:: Linear regression errors example

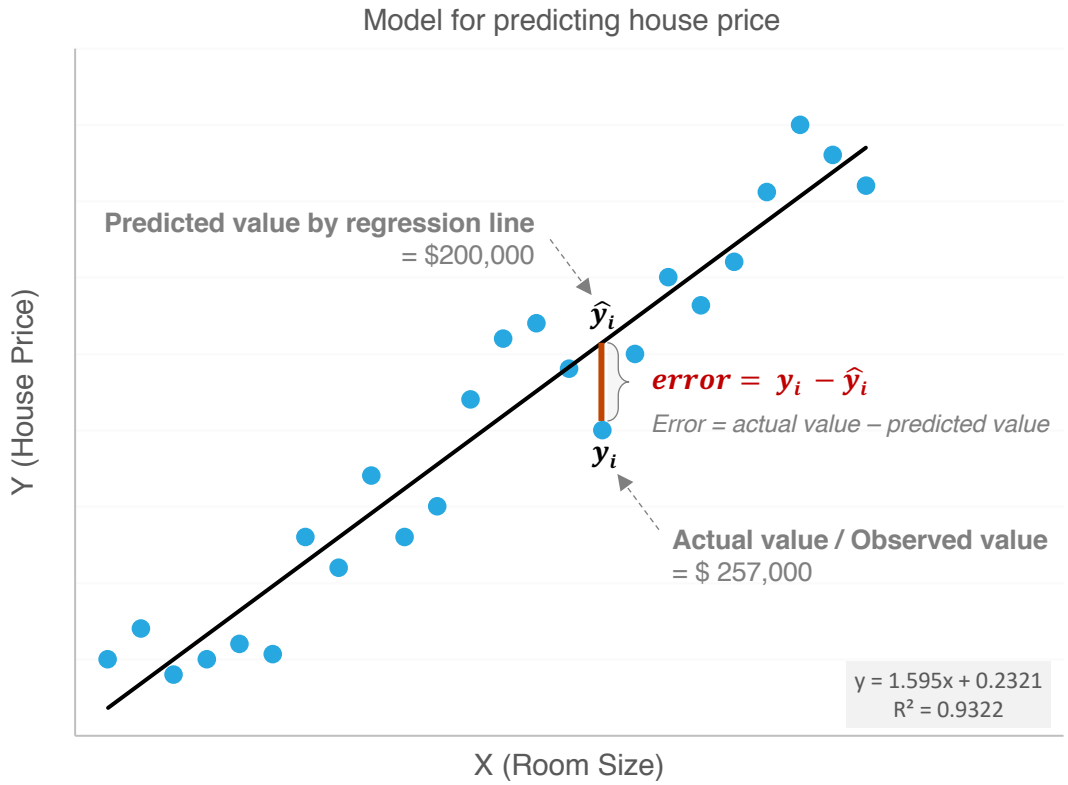
Linear Regression example

Suppose you want to create a simple model for predicting the house price just based on the room size. A linear regression model can be built for predicting the price.

:: Linear regression errors example

The prediction error

You can compare the observed value with the predicted value. Such error can be used to evaluate the model's performance.



Linear Regression Errors

There are difference between the actual value and predicted value from the model. That's the prediction error.

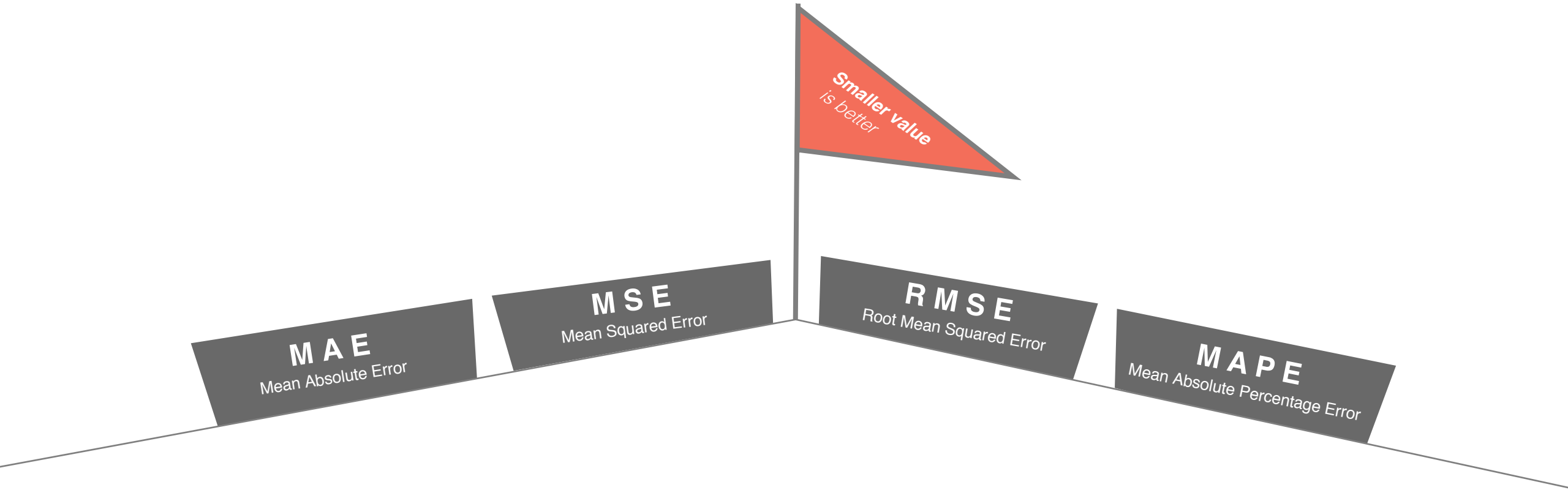
$$Error = y_i - \hat{y}_i$$

In this example, error for this data point:

Error = 200,000 – 257,000 = - 57,000

*To find the "best fit" line, the goal is to **minimize** the sum of the errors of all data points. However, the negative value can cancel out the positive value if only calculating the sum. So some techniques can be used to address this issue. See next slides for more details.*

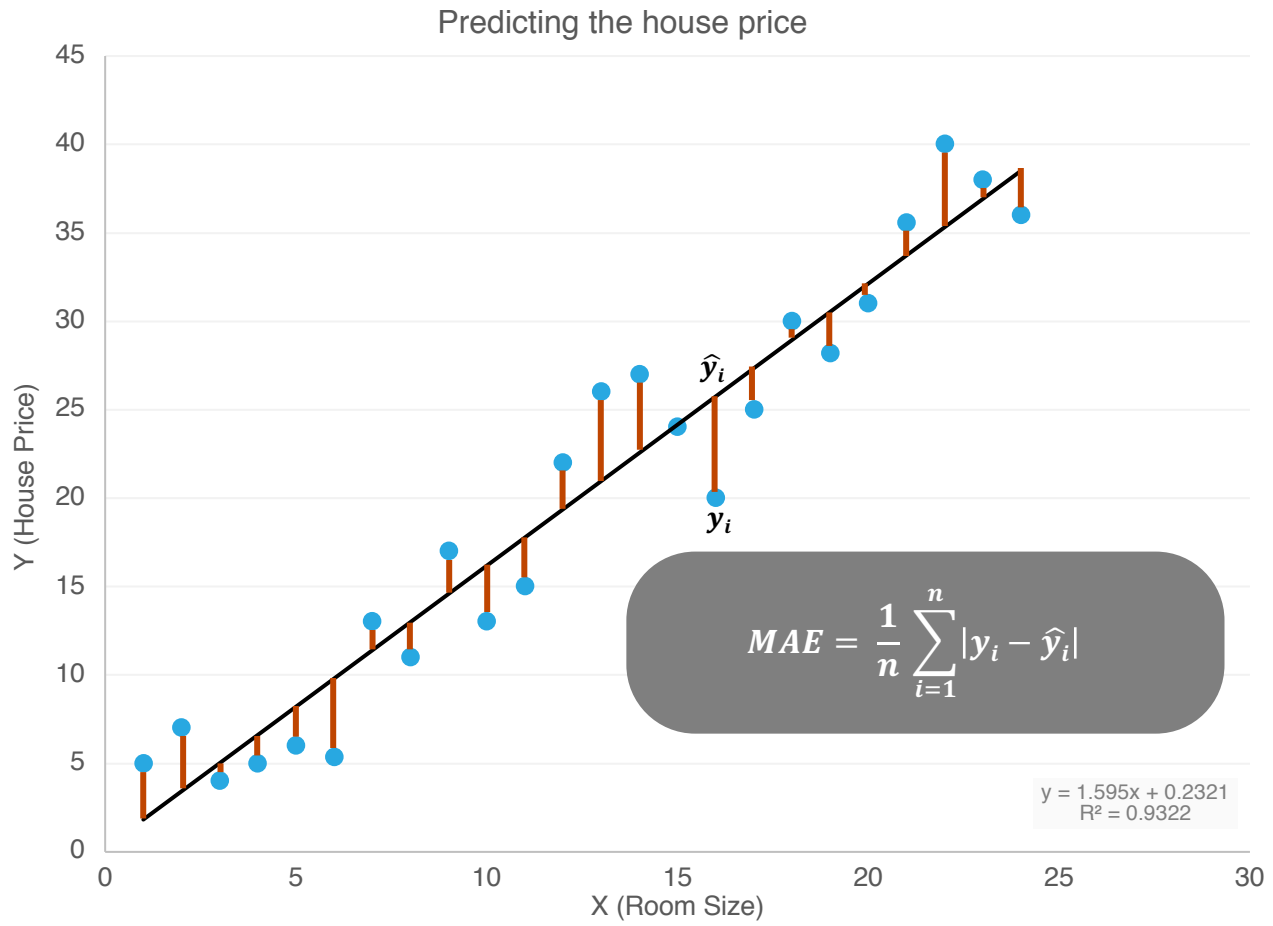
:: Common metrics for measuring the regression errors



Lower error values mean the model is more accurate in making predictions

:: Metric for regression model: MAE

MAE is the <u>mean</u> of the <u>absolute value</u> of the errors



Mean Absolute Error (MAE)

- MAE measures how close the predictions are to the actual outcomes.

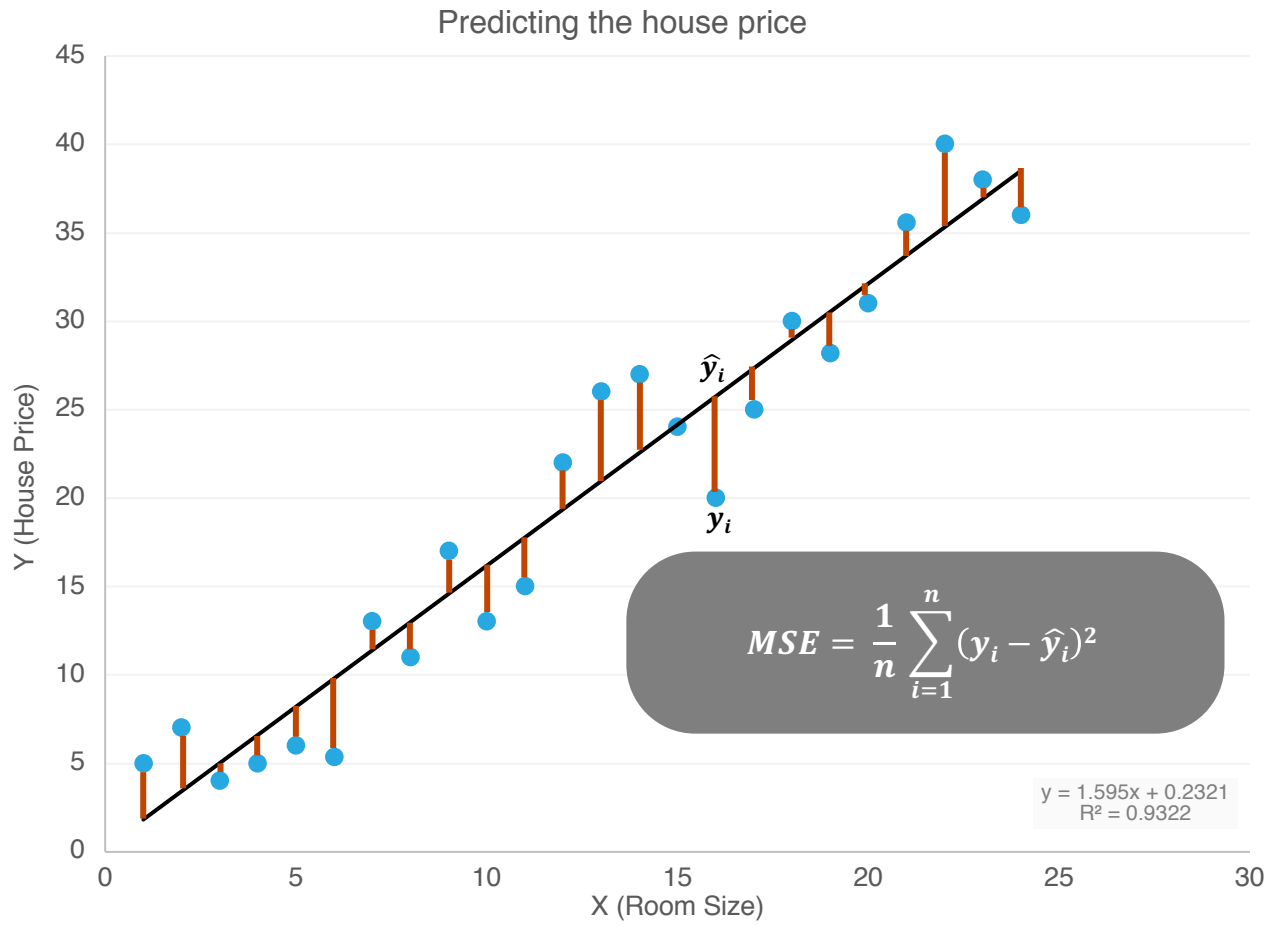
  It's the average sum of absolute difference between the actual value the predicted values for all data points. Because of the absolute value, the negative and positive errors don't cancel out each other

- The best regression is the one that minimizes MAE. Thus, a smaller score is better !

1 Source: http://www.ritchieng.com/machine-learning-evaluate-linear-regression-model/

:: Metric for regression model: MSE

MSE is the <u>mean</u> of the <u>squared value</u> of the errors



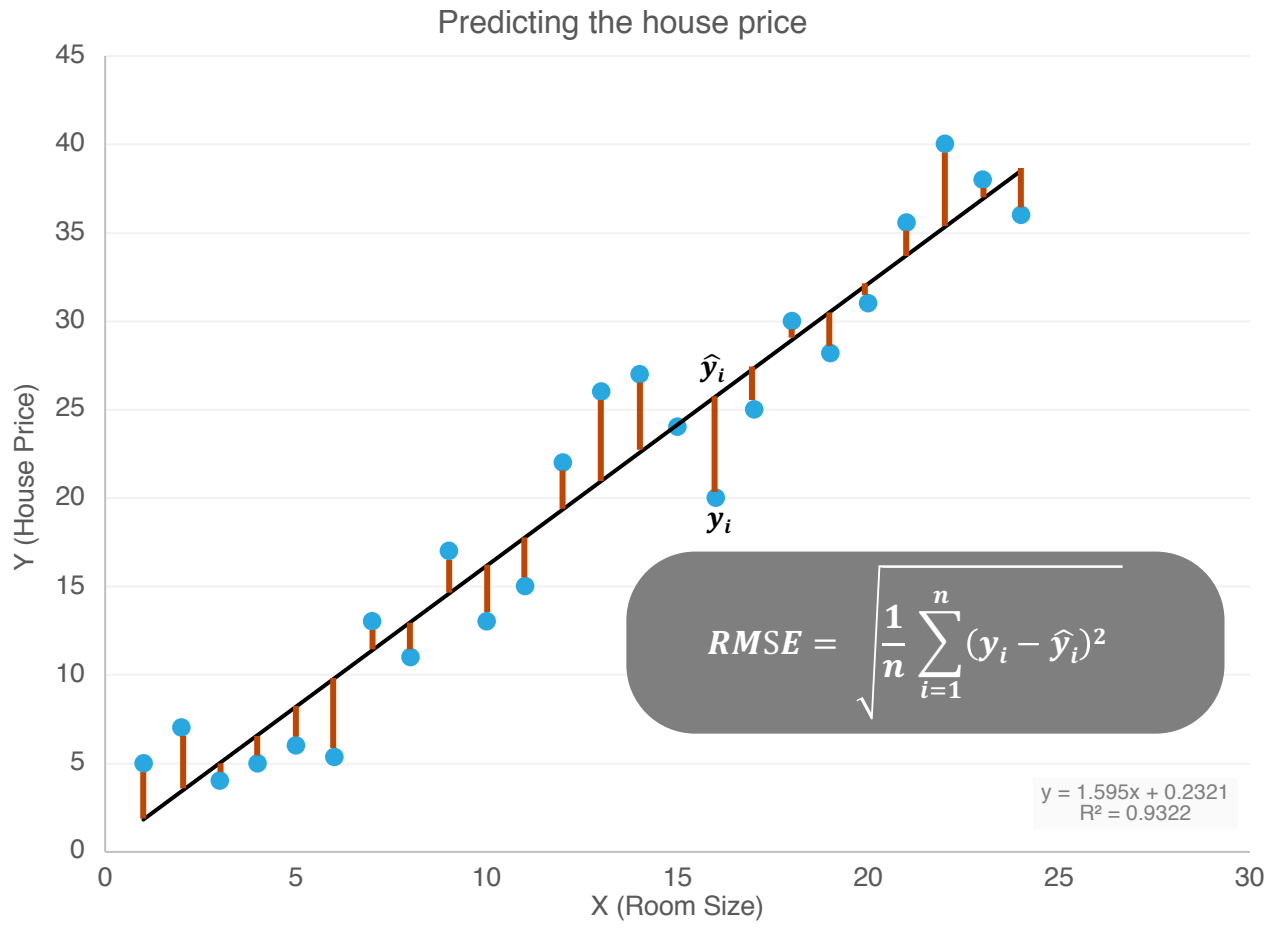
Mean Square Error (MSE)

- MSE measures the average sum of squares of the difference between the actual value the predicted values for all data points. Because of the square, the negative values don't cancel positive values and it also amplifies the impact of the errors.
- Because of the square, large errors have relatively greater influence on MSE than do the smaller error.
- The best regression is the one that minimizes MSE. Thus, a smaller score is better !

1 Source: http://www.ritchieng.com/machine-learning-evaluate-linear-regression-model/

:: Metric for regression model: RMSE

RMSE is the square root of the mean of the squared errors



Root Mean Square Error (RMSE)

- RMSE is the most popular evaluation metrics used in regression problems. It has same unit with "Y"
- RMSE may be the most common metric, but it has some problems. RMSE is highly affected by outlier values.
- RMSE more aggressively punishes big errors than small ones. This means the RMSE should be more useful when large errors are particularly undesirable[1].

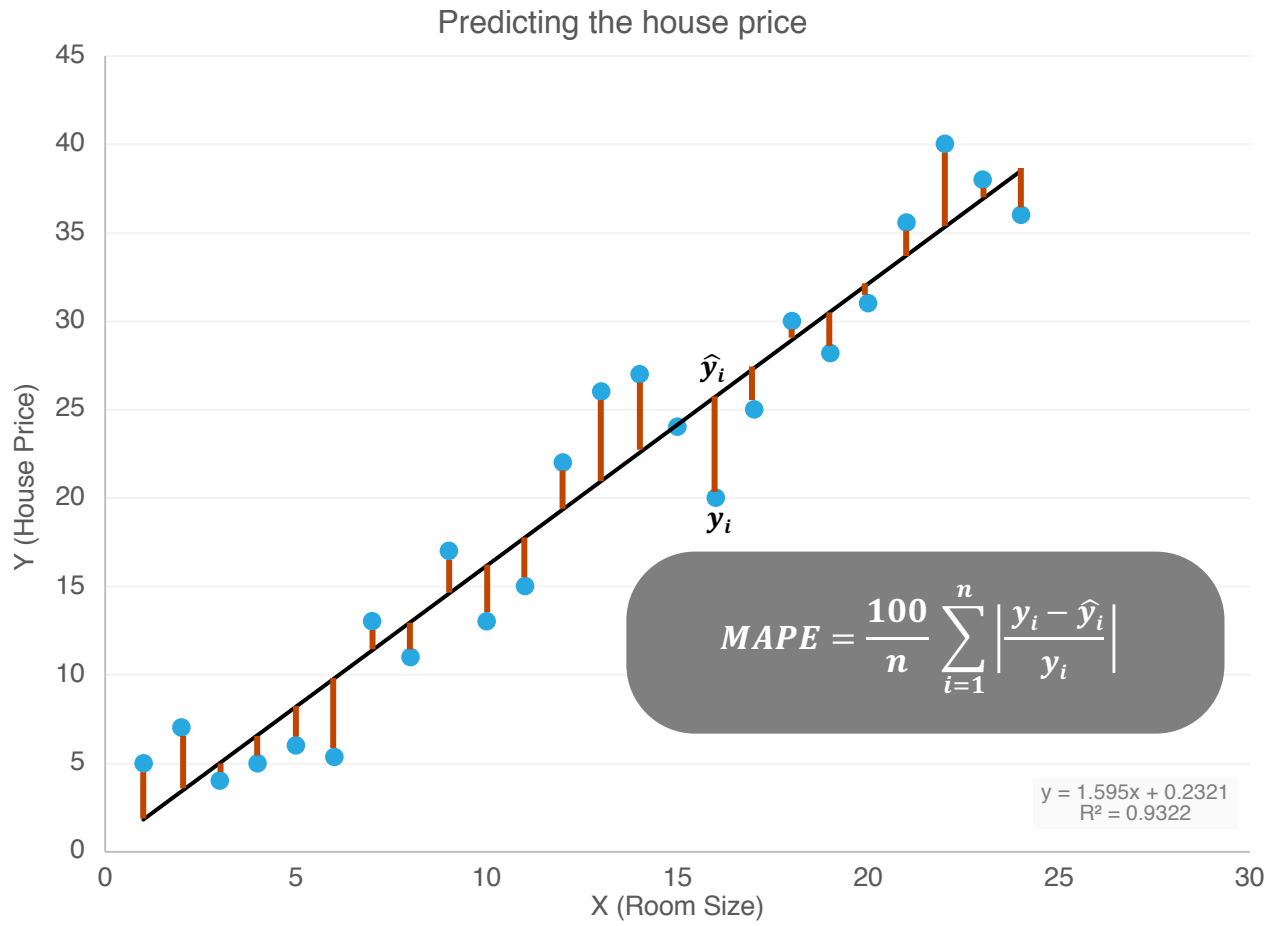
  The RMSE is more sensitive to outliers than the MAE. But when outliers are exponentially rare (like in a bell-shaped curve), the RMSE performs very well and is generally preferred [2].

- A smaller RMSE value is better !

1 Source: https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d
2 Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow ,Page 61

:: Metric for regression model: MAPE

The MAPE (Mean Absolute Percent Error) measures the size of the error in percentage terms



MAPE is one of the most popular measures for forecasting time series error.

Mean Absolute Percentage Error (MAPE)

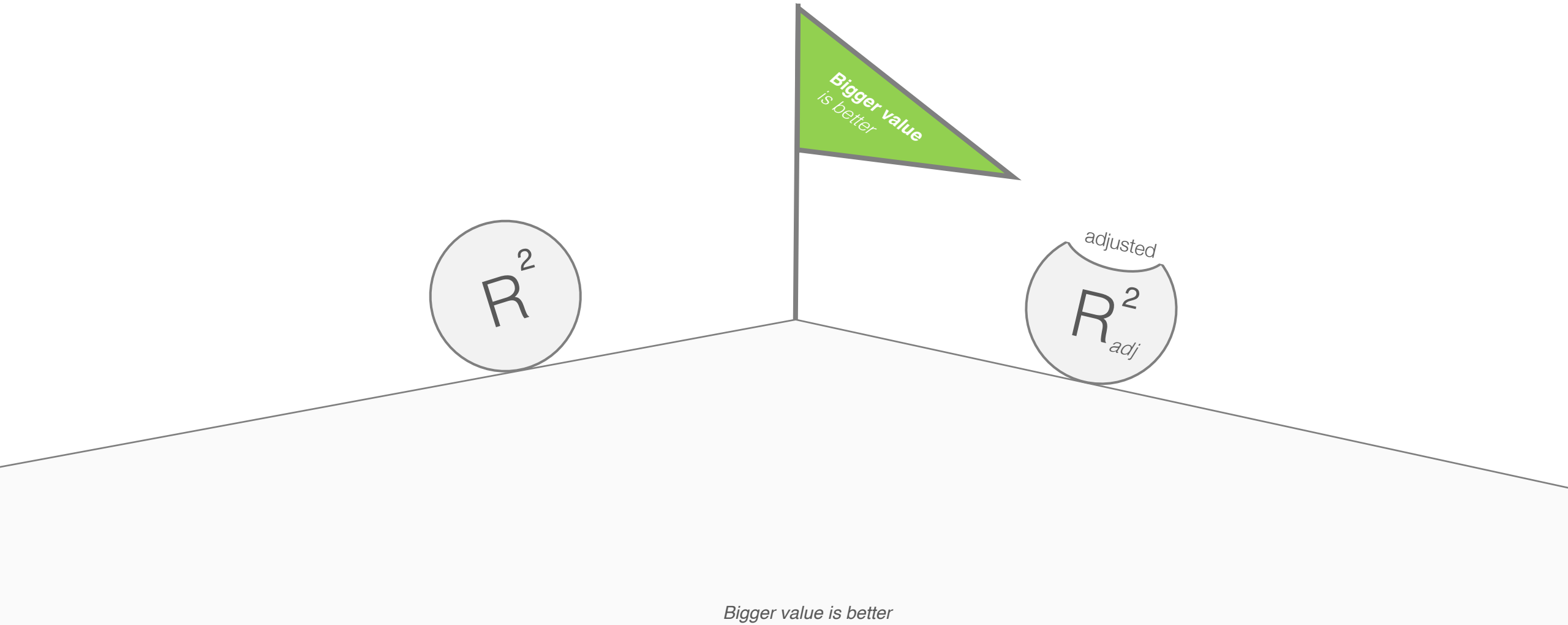
- MAPE functions best when there are no extremes to the data (including zeros)[1].
- In spite of the popularity in measuring the forecasting error, MAPE has major drawbacks in practical application and it should be used cautiously.

With zeros or near-zeros, MAPE can give a distorted picture of error. The error on a near-zero item can be infinitely high, causing a distortion to the overall error rate when it is averaged in. For forecasts of items that are near or at zero volume, Symmetric Mean Absolute Percent Error (SMAPE) is a better measure[2].

- A smaller MAPE value is better

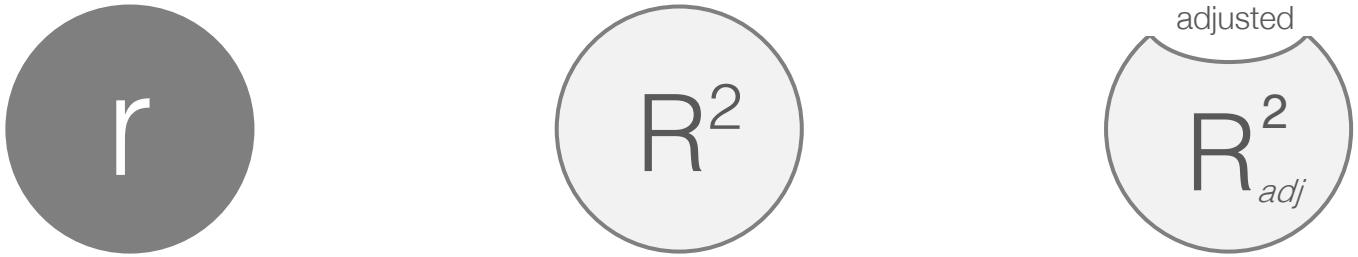
1&2 Source: http://www.vanguardsw.com/business-forecasting-101/mean-absolute-percent-error/-

:: Other metrics for evaluating the regression model



Bigger value is better

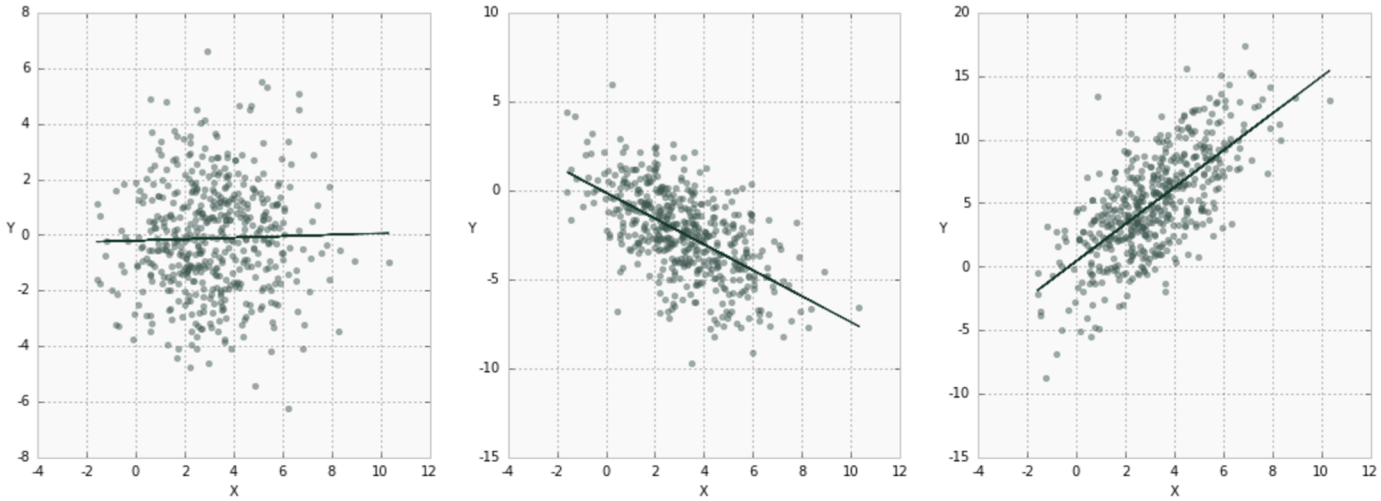
:: Let's learn the key concept one by one



:: At first let's take a look at *r*

r is also called as Pearson correlation coefficient

r



r = 0.03 **r = - 0.58** **r = 0.69**

Pearson correlation coefficient , also referred to as the Pearson's *r* , is is a measure of the linear correlation between two variables *X* and *Y.*

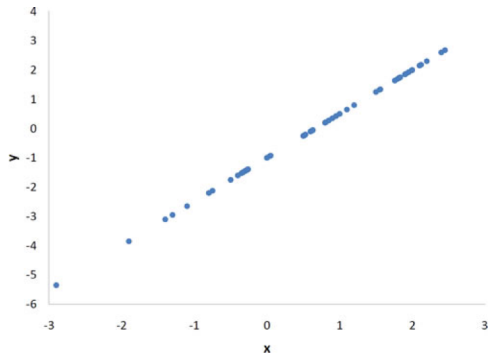
By examining the coefficient values , you can infer something about the strength of the relationship between the two variables, and whether they are positively correlated or negatively correlated.

- It has a value between +1 and −1.
- 0 is no linear correlation

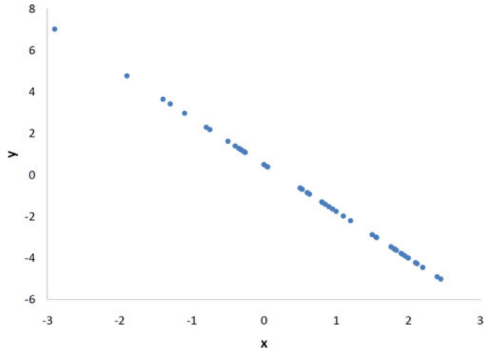
Source of picture: https://www.datascience.com/blog/introduction-to-correlation-learn-data-science-tutorials

:: At first let's take a look at r

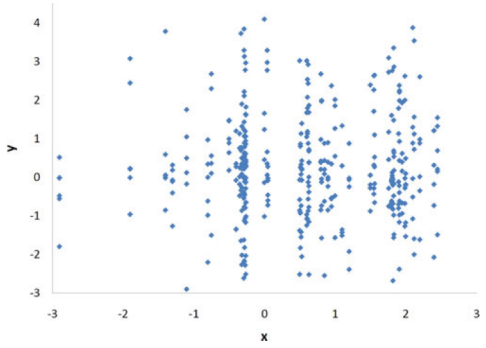
Pearson's r measures the strength of the liner relationship between two variables.



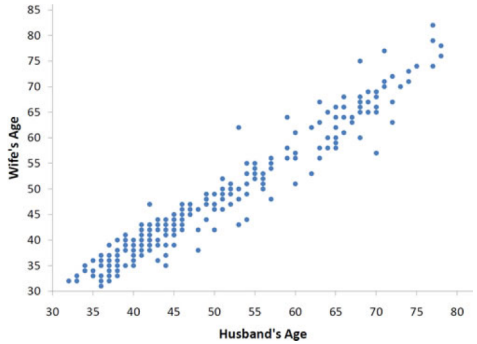
r = 1
A perfect positive linear relationship



r = -1
A perfect negative linear relationship



r = 0
There is no relationship between X and Y



r = 0.97
Strong positive relationship

- A positive correlation indicates a relationship between x and y measures such that as values of x increase, values of y also increase.
- A negative correlation indicates the opposite—as values of x increase, values of y decrease

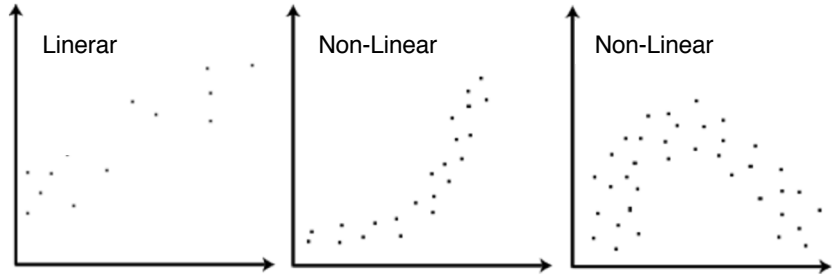
Source of picture: http://onlinestatbook.com/2/describing_bivariate_data/pearson.html

:: At first let's take a look at *r*

How to calculate Pearson's r ?

- **Step 1: Determine linearity**

  The first step in studying the relationship between two continuous variables is to draw a scatter plot of the variables to check for linearity. The correlation coefficient should not be calculated if the relationship is not linear[1].



- **Step 2: Clean data**

  You must remove or fill in missing values, remove or clip outliers, and ensure that the columns have the proper data type[2].

- **Step 3: Calculate the coefficient**

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2(y-\bar{y})^2}} \quad \textit{Or} \quad r = r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}$$

where:

- n is the number of samples
- x_i, y_i are the single samples indexed with i
- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (the sample mean); and analogously for $\bar{y}$

Example

x	y	xy	x^2	y^2
2	4	8	4	16
1	3	3	1	9
3	4	12	9	16
2	3	6	4	9
4	6	24	16	36
5	5	25	25	25
3	5	15	9	25
$\sum x$	$\sum y$	$\sum xy$	$\sum x^2$	$\sum y^2$
20	**30**	**93**	**68**	**136**

Video: Correlation Coefficient (9 minutes)
https://www.youtube.com/watch?v=atLZNGsTN6k

1 Source: http://learntech.uwe.ac.uk/da/Default.aspx?pageid=1442
2 Source: https://msdn.microsoft.com/en-us/library/azure/dn905819.aspx

:: R^2 - coefficient of determination

R^2 is a standard way of measuring how well the model fits the data

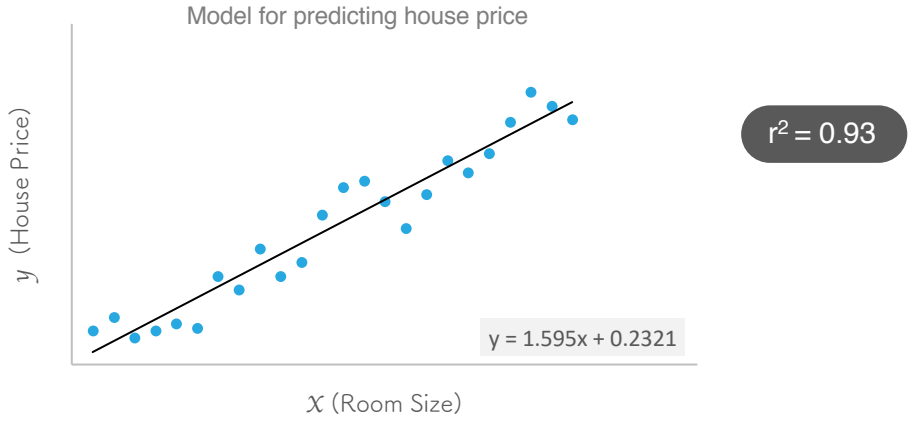
- **The coefficient of determination, denoted R2 or r2 and pronounced "R squared"**

  It is the square of the Pearson correlation coefficient *r* . So, for example, a Pearson correlation coefficient of 0.6 would result in a coefficient of determination of 0.36, (i.e., $r^2 = 0.6 \times 0.6 = 0.36$). [1]

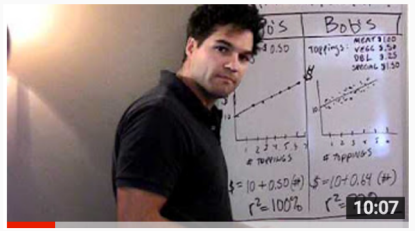
- **R^2 describes the proportion of the variance/variation in the dependent variable y explained by the regression model.**

  For example, r^2 =0.93 or 93% means that approximately 93% of the variation the dependent variable y can be explained by the variation in the variable x.

  Still confused ? Please watch the video for detailed explanation. →



Video : What does r squared tell us? What does it all mean



Youtube Video (10 min)

https://www.youtube.com/watch?v=IMjrEeeDB-Y

:: R^2 - coefficient of determination

In general, the higher the R-Squared, the better the model fits the data

R^2 gives us some information about the **goodness of fit** of a model. In another words, R^2 measures how well the regression line approximates the real data points.

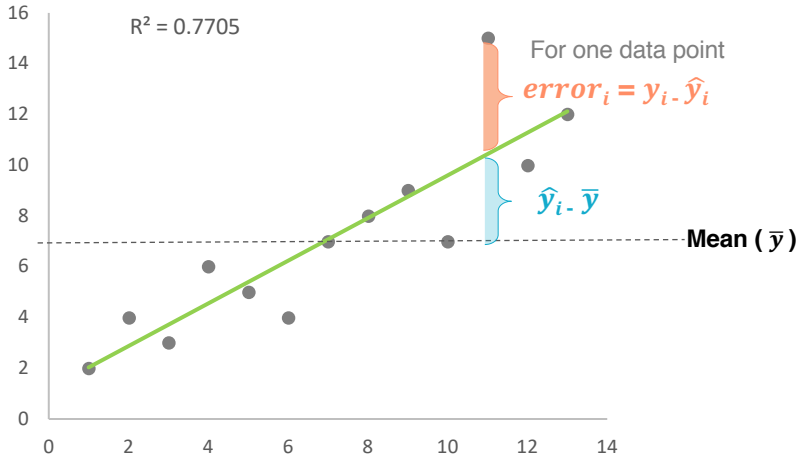
***R^2* is a value between 0 and 1.**

- 0 means the model is random (explains nothing);
- 1 means there is a perfect fit.
- However, caution should be used in interpreting R^2 values, as low values can be entirely normal and high values can be suspect[1].

1 Source: https://msdn.microsoft.com/en-us/library/azure/dn905915.aspx

:: R^2 - coefficient of determination

The formula of R^2



Calculate the sum of squares for **All** data points

The error sum of squares

$$SS_{error} = \sum_i (y_i - \hat{y}_i)^2$$

The regression sum of squares

$$SS_{reg} = \sum_i (\hat{y}_i - \bar{y})^2$$

- $\hat{y}$ (y hat) is the predicted value
- y_i is the actual value/target value)
- $\bar{y}$ is the mean value

The total sum of squares

$$SS_{\text{total}} = SS_{reg} + SS_{error}$$

SS_{reg} quantifies how far the estimated sloped regression line $\hat{y}_i$, is from the horizontal "no relationship line," the sample mean $\bar{y}$

SS_{error} quantifies how much the data points y_i vary around the estimated regression line $\hat{y}_i$,

SS_{total} quantifies how much the data points, y_i vary around their mean $\bar{y}$

> R^2 is the regression sum of squares divided by the total sum of squares [1].
>
> $$R^2 = \frac{SS_{reg}}{SS_{total}} = 1 - \frac{SS_{error}}{SS_{total}}$$

1 source : https://newonlinecourses.science.psu.edu/stat501/node/255/

:: R^2 - coefficient of determination

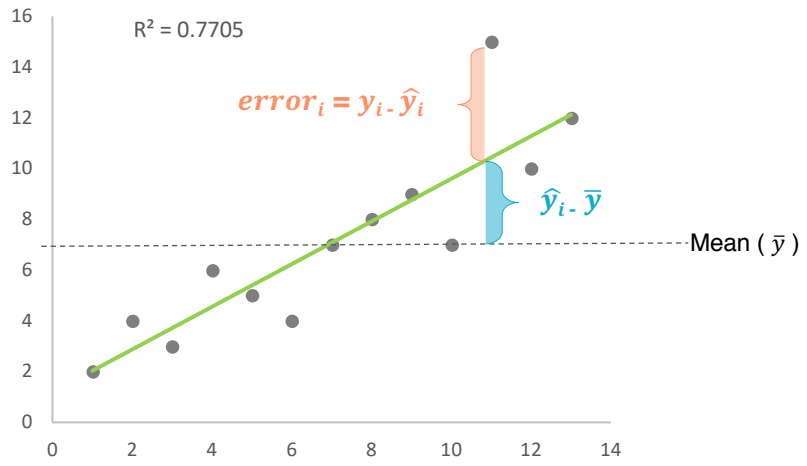
Some Problems with R^2

Problem 1:

Every time you add a predictor to a model, the R-squared increases, even if due to chance alone. It never decreases. Consequently, a model with more terms may appear to have a better fit simply because it has more terms.

Problem 2:

If a model has too many predictors and higher order polynomials, it begins to model the random noise in the data. This condition is known as overfitting the model and it produces misleadingly high R-squared values and a lessened ability to make predictions [1].



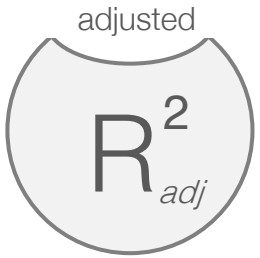
$$R^2 = \frac{SS_{reg}}{SS_{total}} = 1 - \frac{SS_{error}}{SS_{total}}$$

1 source: http://blog.minitab.com/blog/adventures-in-statistics-2/multiple-regession-analysis-use-adjusted-r-squared-and-predicted-r-squared-to-include-the-correct-number-of-variables

:: Adjusted R^2

Adjusted R^2 is a modified version of R^2

Adjusted R^2 is a corrected goodness-of-fit (model accuracy) measure for linear models[1]. Adjusted R-squared penalizes the model for the inclusion of nonsense variables. If you add more and more useless variables to a model, adjusted r-squared will decrease. If you add more useful variables, adjusted r-squared will increase[2].



R^2 tends to optimistically estimate the fit of the linear regression. Adjusted R^2 attempts to correct for this overestimation.

Adjusted R^2 is always less than or equal to R^2.

- A value of 1 indicates a model that perfectly predicts values in the target field.
- A value that is less than or equal to 0 indicates a model that has no predictive value.
- In the real world, adjusted R^2 lies between these values [3].

About which one to use...in the case of a linear regression with more than one variable: adjusted R-squared. For a single independent variable model, both R^2 and adjusted R^2 are interchangeable [4]

formula: $$R_{adj}{}^2 = 1 - \left[\frac{(1-R^2)(n-1)}{n-k-1}\right]$$

Where:

- n is the sample size (i.e. number of points in your data sample)
- k is s the total number of explanatory variables in the model (not including the constant term)

1 source: https://www.ibm.com/support/knowledgecenter/en/SSWLVY_1.0.0/com.ibm.spss.analyticcatalyst.help/analytic_catalyst/rsquared_adjusted.html
2Source: http://www.statisticshowto.com/adjusted-r2/
3 source: https://www.ibm.com/support/knowledgecenter/en/SSWLVY_1.0.0/com.ibm.spss.analyticcatalyst.help/analytic_catalyst/rsquared_adjusted.html
4 source: https://www.quora.com/What-is-the-difference-between-R-squared-and-Adjusted-R-squared

:: Regression evaluation results example

Linear Regression Evaluation Metrics

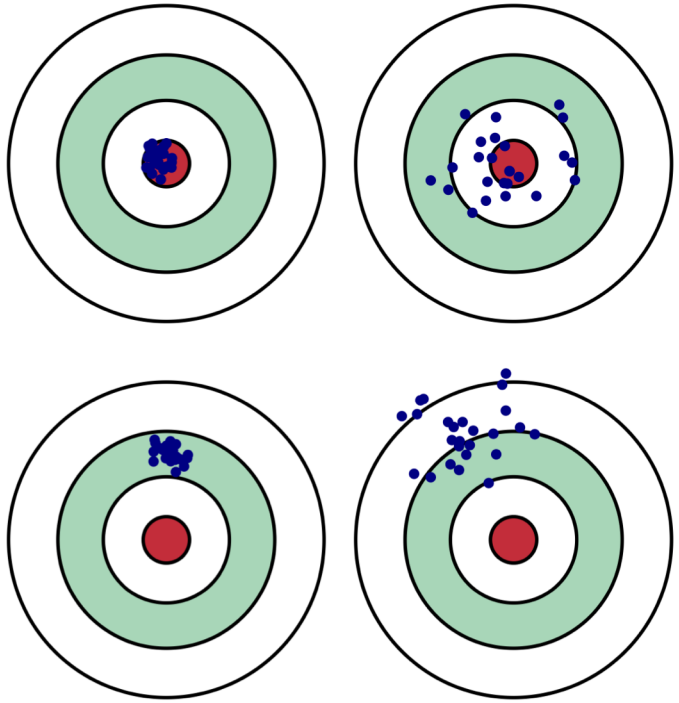
Metrics	
Mean Absolute Error	747.975254
Root Mean Squared Error	955.587783
Relative Absolute Error	0.163528
Relative Squared Error	0.026598
Coefficient of Determination	0.973402

:: Summary: Testing and Error Metrics

Youtube Video (44 min)
Machine Learning: Testing and Error Metrics

https://www.youtube.com/watch?v=aDW44NPhNw0

:: Cause of error

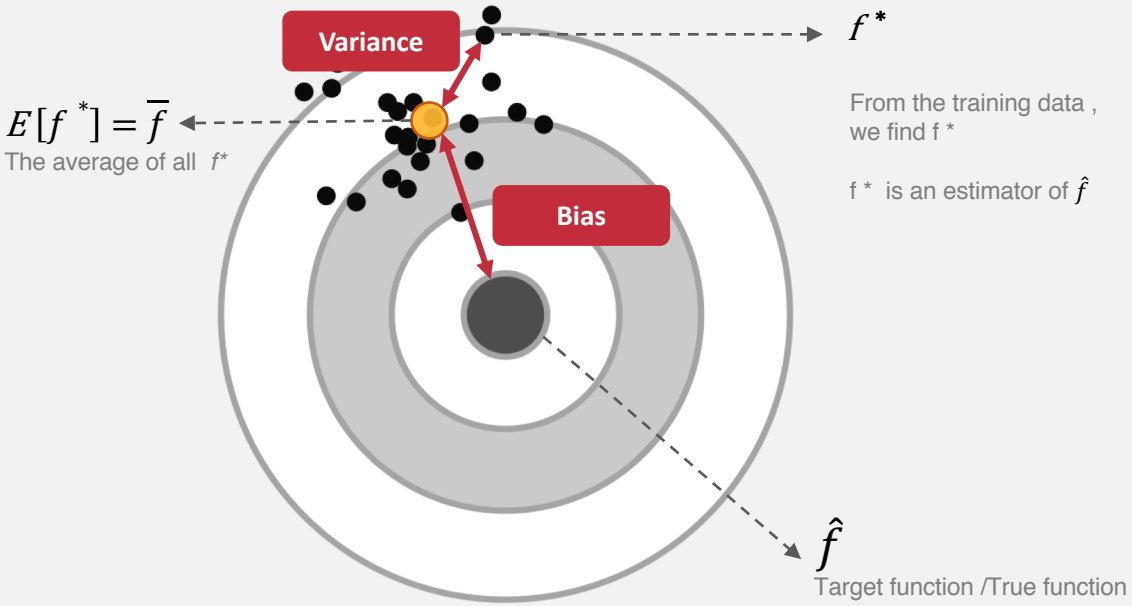


Where does the error come from?

Error mainly comes from Bias and Variance

Source of diagram: http://scott.fortmann-roe.com/docs/BiasVariance.html

:: Decompose the error



- **The variance** of the learning method represents how much the learning method will move around its mean; The variance is error from sensitivity to small fluctuations in the training set[3].

- **The Bias** of learning method represents the difference between the expected (or average) prediction of our model and the correct value which we are trying to predict.

Expected predicted error = **Bias²** + **Variance** + Irreducible error

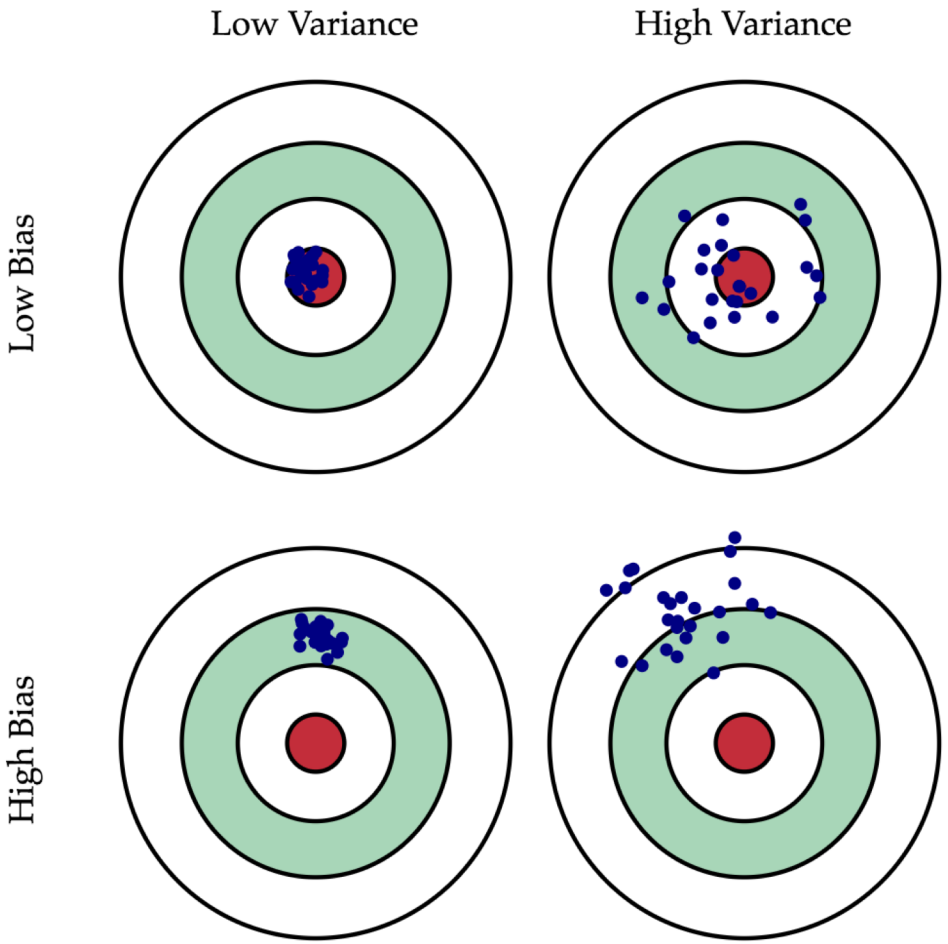
The prediction errors can be decomposed into two main subcomponents we care about:

- error due to "bias"
- error due to “variance” [1]

Irreducible error, is the noise term that cannot fundamentally be reduced by any model. Given the true model and infinite data to calibrate it, we should be able to reduce both the bias and variance terms to 0. However, in a world with imperfect models and finite data, there is a tradeoff between minimizing the bias and minimizing the variance[2].

1&2 Source: http://scott.fortmann-roe.com/docs/BiasVariance.html
3 Source: https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff
Source of idea for diagram: https://www.youtube.com/watch?v=D_S6y0Jm6dQ

:: Bias vs Variance



There are 4 cases representing combinations of both high and low bias and variance.

- Low Bias & Low Variance → Good case
- Low Bias & High Variance
- High Bias & Low Variance
- High Bias & High Variance → Bad case

Ideally, one wants to choose a model that both accurately captures the regularities in its training data, but also **generalize**s well to unseen data (i.e. new data). Unfortunately, it is typically impossible to do both simultaneously[1].

Model Selection

- There is usually a trade-off between Bias and Variance. So normally reducing one tends to increase the other
- Select a model that balances two kinds of error to minimize the total error

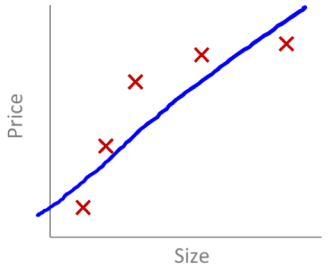
1 Source: https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff
Source of diagram: http://scott.fortmann-roe.com/docs/BiasVariance.html

:: Diagnosing Bias vs Variance

Is it a bias problem or variance problem ?

$\theta_0 + \theta_1 x$

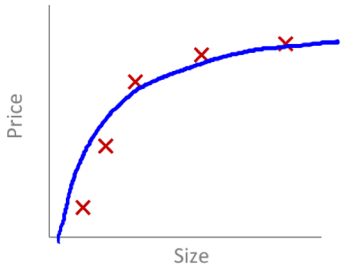
Degree of polynomial = 1



High bias (underfit)

$\theta_0 + \theta_1 x + \theta_2 x^2$

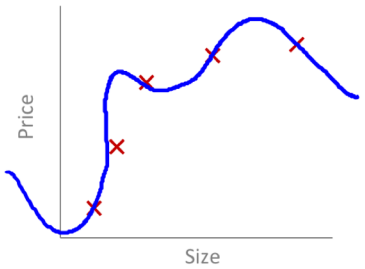
Degree of polynomial = 2



“Just right”

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

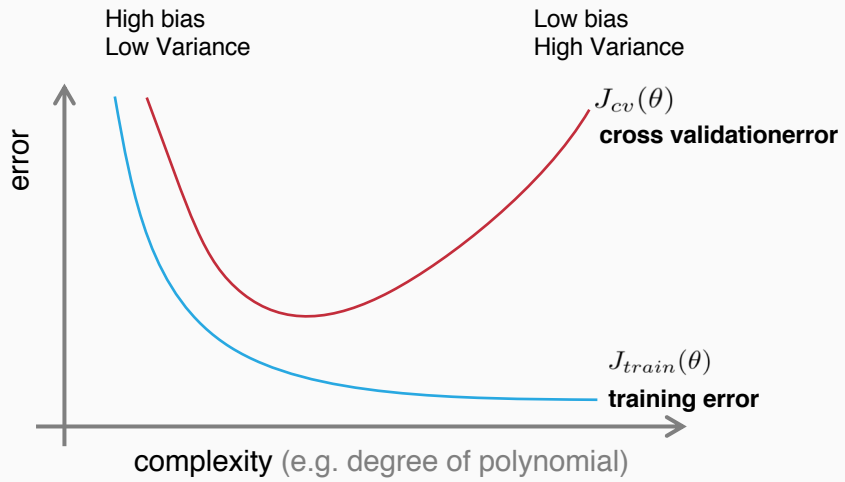
Degree of polynomial = 4



High variance (overfit)

Your model may be suffering from a bias problem or variance problem.

The diagnosis helps you figuring out how to improve the performance of model



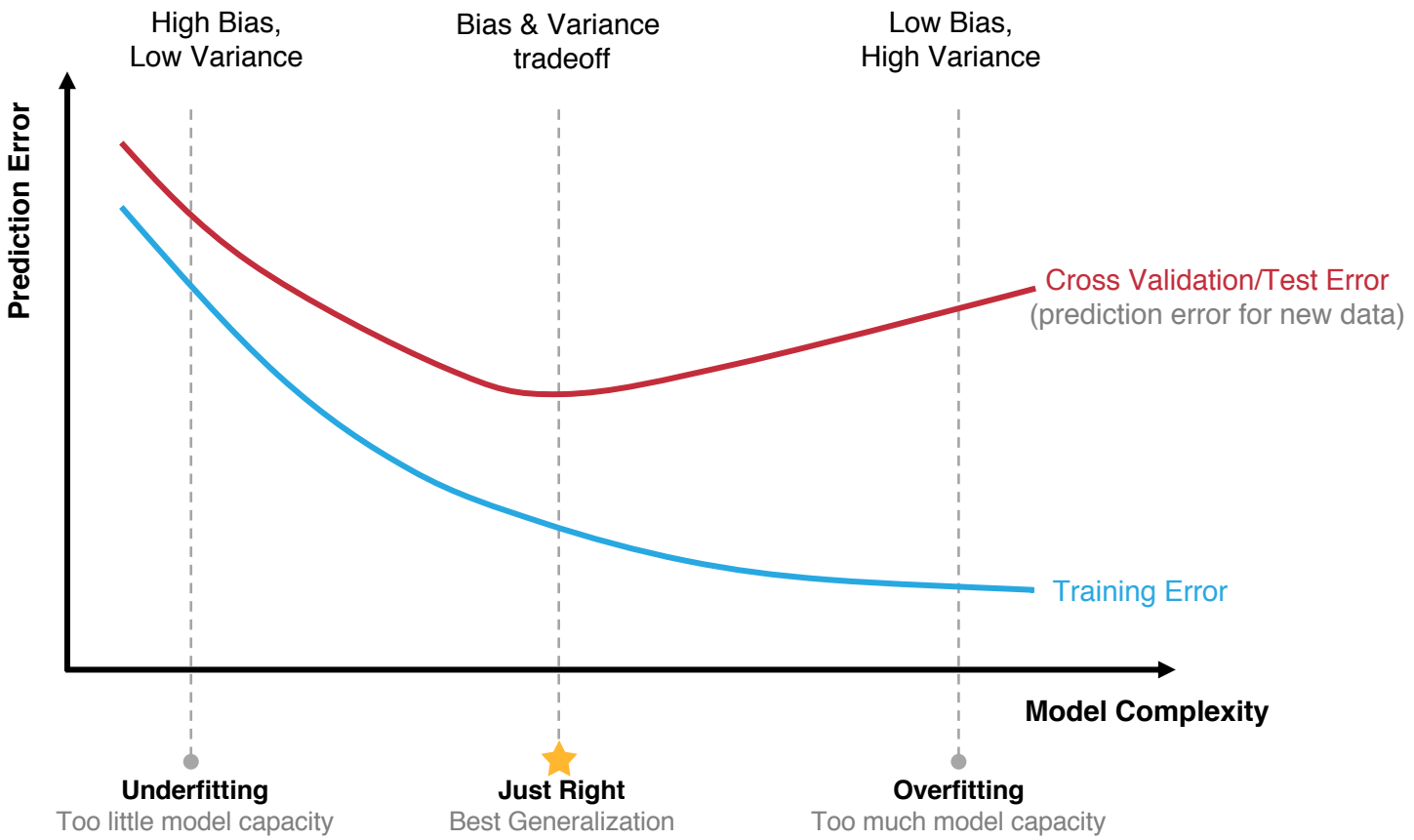
The training error will tend to **decrease** as we increase the degree of the polynomial.

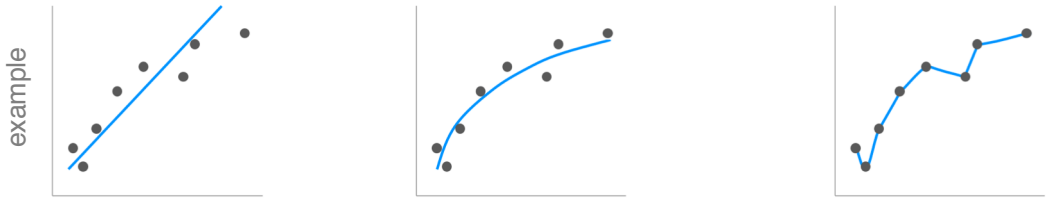
At the same time, the cross validation error will tend to **decrease** as we increase d up to a point, and then it will **increase** as d is increased, forming a convex curve.

Video: Diagnosing Bias vs. Variance (7min)
https://youtu.be/fDQkUN9yw44

Source: https://www.coursera.org/learn/machine-learning/lecture/yCAup/diagnosing-bias-vs-variance

:: Bias and Variance Trade-off





Underfitting = Bias problem

Overfitting = Variance problem

The simplest way to determine if your model is suffering more from bias or from variance is the following rule of thumb:

- If your model is performing really well on the training set, but much poorer on the hold-out set, then it's suffering from high variance[1].
- On the other hand if your model is performing poorly on both training and test data sets, it is suffering from high bias[2].

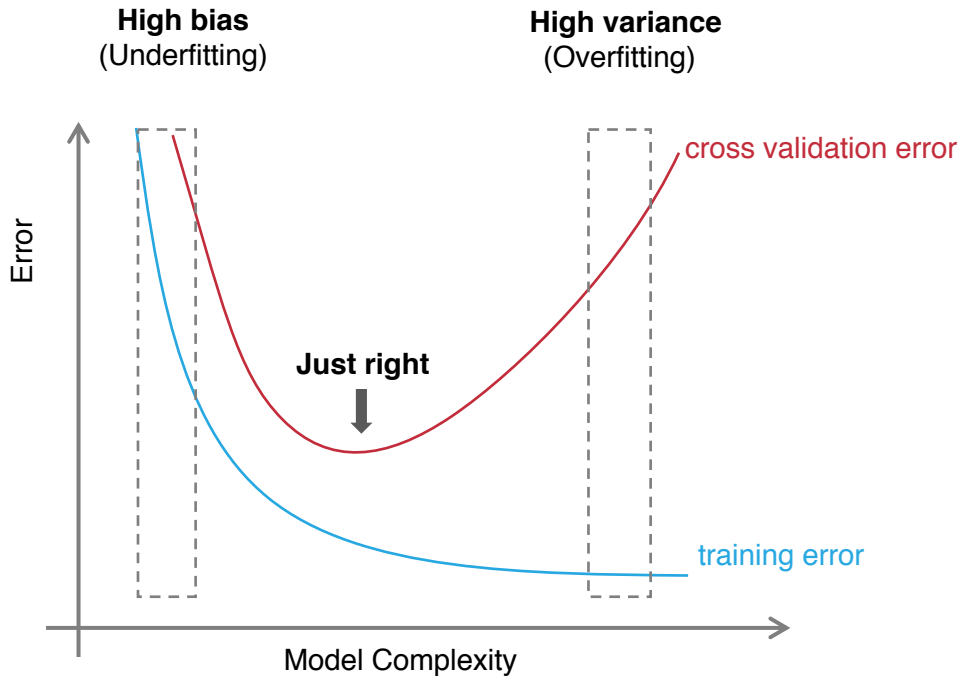
Source: https://medium.com/rants-on-machine-learning/7-ways-to-improve-your-predictive-models-753705eba3d6

:: What to do next ?

Diagnose if the model has Bias or Variance problem

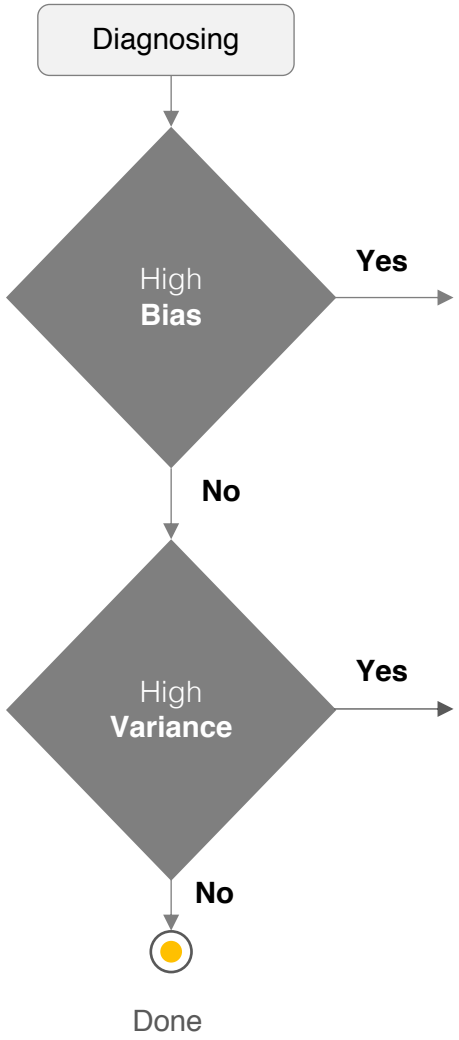
Use cross validation set to diagnose if you have a bias or variance problem.

- **A high bias problem** has the following characteristics [1]
  - High training error.
  - Validation error is similar in magnitude to the training error.
- **A high variance problem** on the other hand has the following characteristics [2]
  - Low training error
  - Very high validation error



1 &2 Source: http://www.learnopencv.com/bias-variance-tradeoff-in-machine-learning/

:: What to do next ?

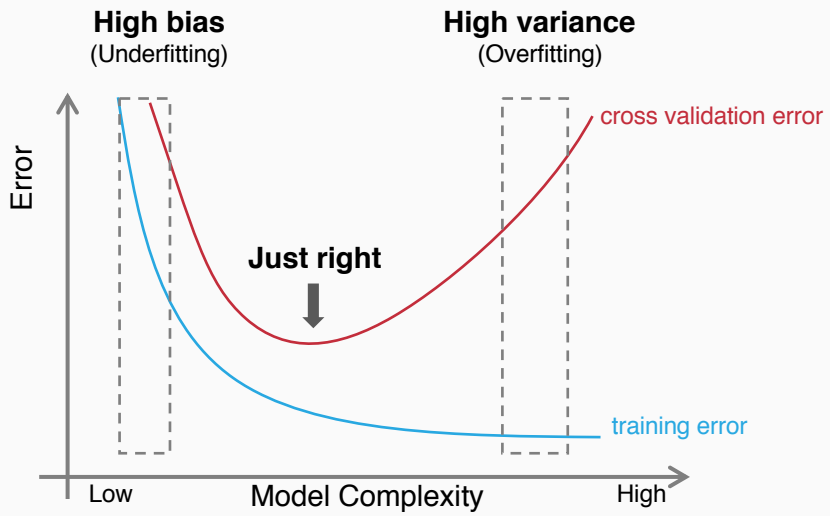


What to do ?

- Make the model more complex
- Get more features
- Tune hyperparameters
- Use an ensemble of low complexity algorithms (Boosting)

What to do ?

- Obtain More data
- Decrease number of features
- Use regularization
- Tune hyperparameters
- Change to less complex algorithm
- Use an ensemble of high complex algorithms (Bagging / Random Forrest)



:: What to do next ?

What to do with **high Bias**?

What to do with **high Variance**?

:: What to do next ?

What to do with **high Bias**?

What to do with **high Variance**?

Image: Designed by Asierromero / Freepik

:: What to do with high bias ?

How to address Underfitting (High Bias) ?

if your model can't even fit the training examples, then you have high Bias. It means the model is too simplistic (too rigid) and not powerful enough to capture salient pattern in data[1]. To address the underfitting, you can

- **Train a more complex model /different model**
  - Try out other model, or other model architectures
  - More complicated model. For example, bigger network in neural network
- **Add more features as input**
- **Use better optimization algorithm such as Adam**
- **Hyperparameters search**
- **Use ensemble learning – Boosting**
  please refer to AdaBoost algorithm for more details

However,

- more model complexity also means more computational overhead (e.g. a single decision tree vs a random forest of 100 trees) [1]
- More model complexity relies on more complex feature engineering[2].
- More complexity usually means the opposite of explained ability. As we choose more complex models, our models because harder to explain. Most complex models are very hard to understand[3].
- More complexity could increase the risks of overfitting[4].

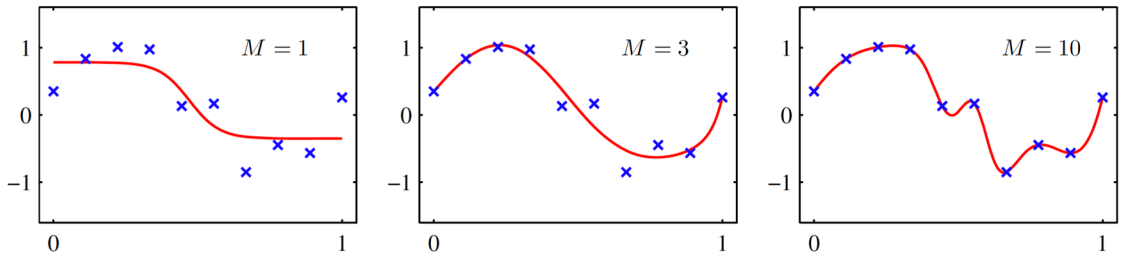
1, 2 &3 Source: https://channel9.msdn.com/Events/OpenSourceTW/DevDays-Asia-2017/AI12
4 source: https://www.youtube.com/watch?v=j9_yzC-x-js

:: What to do with high bias ?

Train a more complex model

You can add the complexity to the model.

- **To make the same algorithm more complex**. For example,
  - A regression model can have have more polynomial terms
  - A decision tree can have more depth
- **Change to a more complex algorithm/model**. For example,
  - Change the Regression algorithm to Neural Network
  - Change Decision Tree to Random Forest



Examples of two-layer networks trained on 10 data points. The graphs show the result of fitting networks having M = 1, 3 and 10 hidden units, respectively

Image : Christopher M. Bishop Pattern Recognition and Machine Learning

:: What to do next ?

What to do with **high Bias**?

What to do with **high Variance**?

Source of image: Designed by Asierromero / Freepik

:: What to do with high variance ?

How to fix High Variance issue (Overfitting) ?

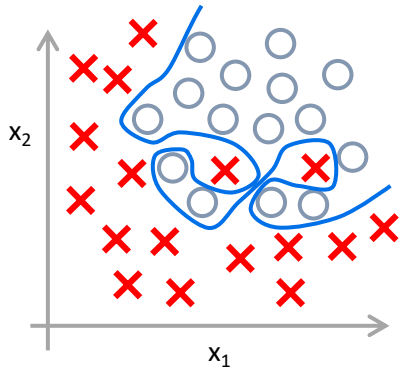
If your model can fit the training data, but large error on the testing data/cross validation dataset, the you probably have large variance. High Variance fails to generalize to new/unseen examples not in training set. It just fits “noise” in the training data.

Overfitting often occurs when a model is **too complex** or when there is **insufficient data**.

To address this issue, you could

- Use more data
- Use Regularization
- Reduce number of features
- Change to less complex model. For example,
  - Limit the number of hidden layers and number of units per layer in Neural network
- Hyper-parameters search
- Use ensemble learning – Bagging & Random Forest
  - please refer to Random Forest t algorithm for more details

Example for overfitting / high variance



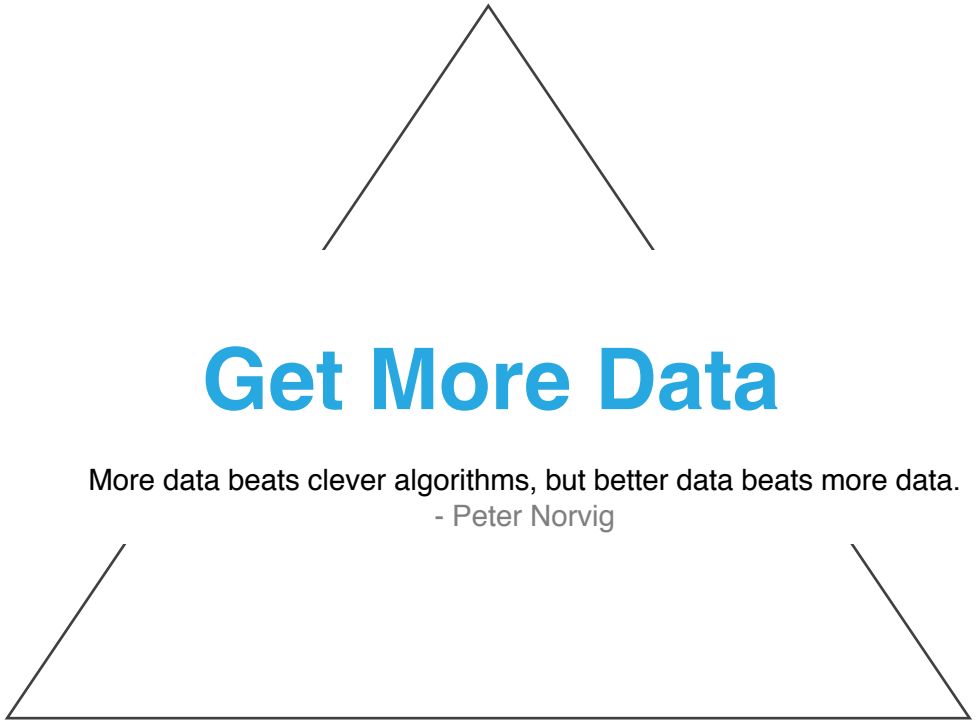
Logistic regression for classification

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$$

(g = sigmoid function)

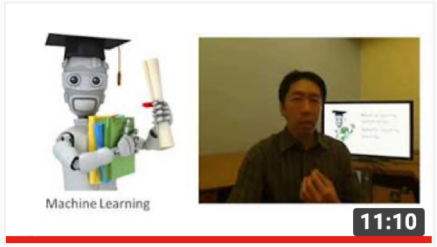
:: What to do with high variance ?

Get more training data



- Almost always the best bet if you have enough compute power to train on more data [1].
- Although more data is very effective, but not always practical. It's usually expensive and time-consuming to get more data.

Youtube Video



Data for Machine Learning
https://www.youtube.com/watch?v=5T77nG7YJhk

1 Source: https://www.youtube.com/watch?v=W0SP8FTmGW0

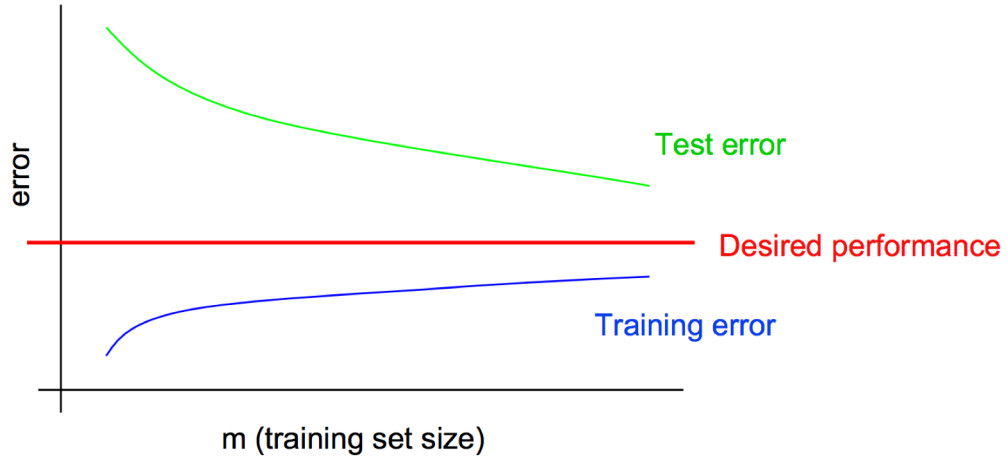
:: What to do with high variance ?

Learning curve is a diagnostic for bias and variance

Learning curve is a plot that compares the performance of a model on training and testing data over a varying number of training instances.

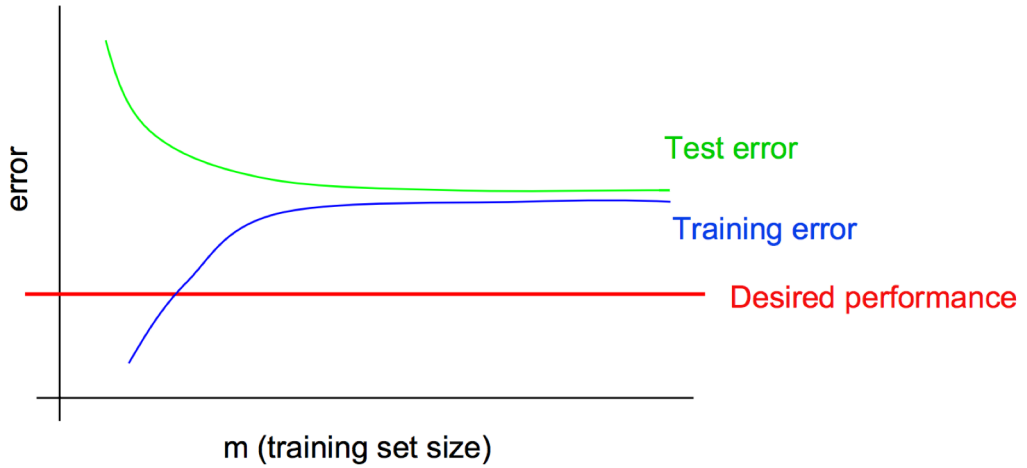
All else being equal, the generalization performance of data-driven modeling generally improves as more training data become available, up to a point[1]. “ learning curves is a tool that I actually use very often to try to diagnose if a physical learning algorithm may be suffering from bias, sort of variance problem or a bit of both. – Andrew Ng ”

Typical learning curve for high variance:



- Test error still decreasing as m increases. Suggests larger training set will help.
- Large gap between training and test error.

Typical learning curve for high bias:

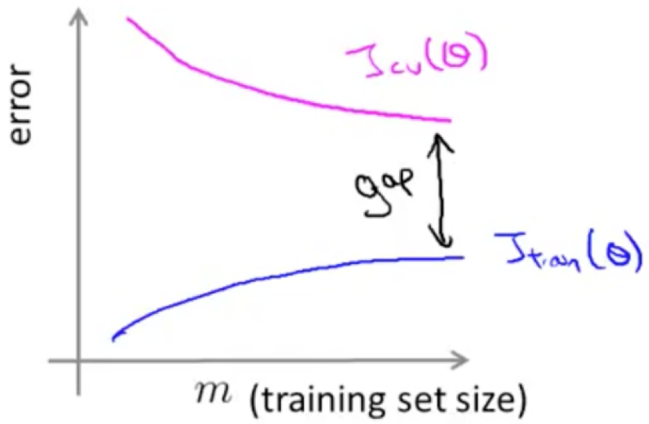


- Even training error is unacceptably high
- Poor generalization
- Small gap between training and test error.

1 , Source: Data Science for Business
2 Source of diagram: http://cs229.stanford.edu/materials/ML-advice.pdf

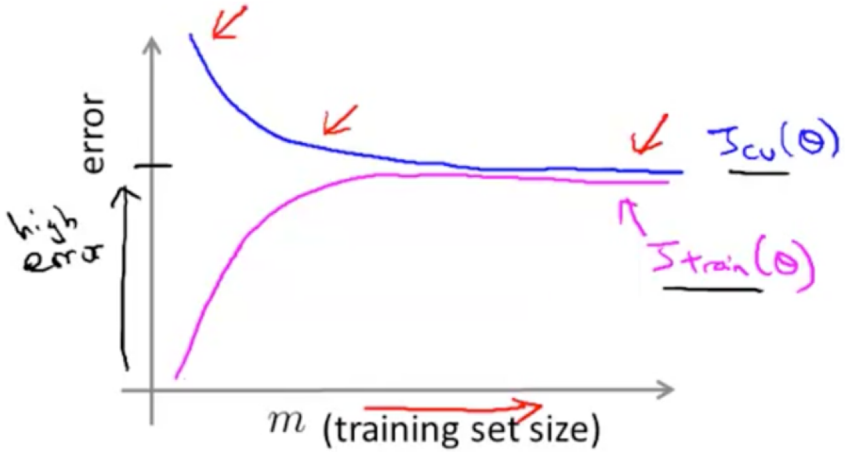
:: What to do with high variance ?

High Variance or High Bias ?



Bad Learning Curve: High Variance (overfitting)

- There is a large gap between the errors
- If a learning algorithm is suffering from high variance
    - getting more data is likely to help
    - Can simplify the model with less features



Bad Learning Curve: High Bias (underfitting)

- Both training error and cross-validation error are high.
- If learning algorithm is suffering from high bias, getting more examples will not help. No matter how much data we feed the model, the model cannot represent the underlying relationship and has high systematic errors

Source: https://www.youtube.com/watch?v=ISBGFY-gBug&t=1s

:: What to do with high variance ?

Ideal learning curve

- Model that generalizes to new data
- Testing and training learning curves converge at similar values
- Smaller the gap, the better our model generalizes

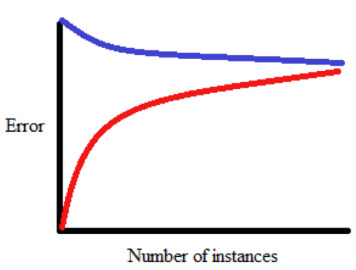


Image #1 (high bias)

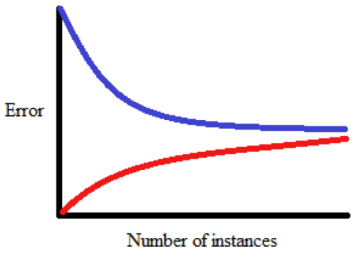


Image #2 (ideal)

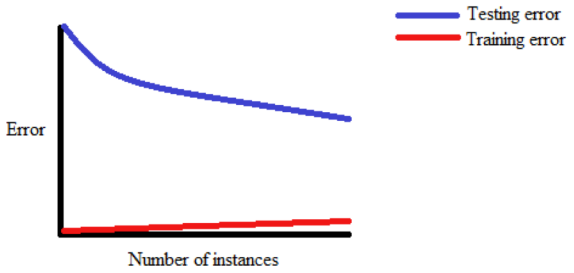


Image #3 (high variance)

Advice For Applying Machine Learning - Learning Curves

https://www.youtube.com/watch?v=ISBGFY-gBug&t=1s

Source: http://www.ritchieng.com/machinelearning-learning-curve/

:: What to do with high variance ?

Use regularization to prevent overfitting

In regression or classification, regularization refers to the process of adding additional terms to our cost function, often to introduce a preference for simpler models.

- Keep all the features, but reduce magnitude/value of parameters θ_j
- Work well when we have a lot of features, each of which contribute a bit to the predicting y

Regularized cost function (for linear regression)**:**

A regularization term is added to a loss function. It is typically chosen to impose a penalty on the complexity of predicting function.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

Regularization parameter :

λ (lambda) is a tuning parameter which controls the importance of the regularization term.

If lambda is set to an extremely large value, parameters $\theta_0, \theta_1, \ldots, \theta_n$ will be shrunken to very small value, even close to zero.

Source: https://www.youtube.com/watch?v=nmHNXsDPPFQ&list=PLZMSzUtKT3z-34luKo-g-t55cvYZBA7wJ

:: What to do with high variance ?

2 most popular regularization methods – L1 & L2 regularization

Regularization essentially keeps all features, but reduces (or penalizes) the effect of some features on the model's predicted values. A regularization term will be added to the cost function

L2 regularization

A type of regularization that penalizes weights in proportion to the sum of the squares of the weights.

L_2 regularization helps drive outlier weights (those with high positive or low negative values) **closer to 0** but not quite to 0.
L_2 regularization always improves generalization in linear models.[1]

$$L_2\ regrlarization\ term = \| \theta \|_2^2 = \sum_{i=1}^{n} \theta_i^2$$

Squares of weight

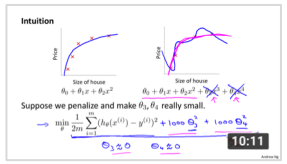
L1 regularization

A type of regularization that penalizes weights in proportion to the sum of the absolute values of the weights.

In models relying on sparse features, L_1 regularization helps drive the weights of irrelevant or barely relevant features **to exactly 0**, which removes those features from the model. [2]

$$L_1\ regrlarization\ term = \| \theta \|_1 = \sum_{i=1}^{n} |\theta_i|$$

Absolute values of the weight



YouTube Video: Add a penalty term to the loss function
https://www.youtube.com/watch?v=wkZUv5dIGJo

1&2 source: https://developers.google.com/machine-learning/crash-course/glossary#L2_regularization

:: What to do with high variance ?

L2 regularization

It uses L2-norm penalty (L2 is the sum of the square of the weights). Performing L_2 regularization encourages weight values toward 0 (but not exactly 0). Let's take Ridge regression as example. Ridge regression is a regularized version of Linear Regression, it uses L2 regularization.

Ridge regression includes all of the features in the model. As the value of Lambda increases, the coefficient shrinks. Though the coefficients can be very very small, they are NOT zero. In other words, the L2 regularization results in plenty of relatively small but nonzero ones. The cost function of Ridge regression with regularization term is as follow,

$$J(\theta) = MSE(\theta) + \lambda \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

Original cost function

sum of the squares of all the feature weights

$$L_2\, regrlarization\ term = \theta_1^2 + \theta_2^2 + \cdots + \theta_n^2$$

For example, a linear model with the following weights:

{ θ_1=0.2, θ_2=0.5, θ_3=5, θ_4=1, θ_5=0.25, θ_6=0.75 }

$$L_2\, regrlarization\ term = 0.2^2 + 0.5^2 + 5^2 + 1^2 + 0.25^2 + 0.75^2 = 26.915$$

The hyperparameter λ

It controls how much you want to regularize the model. Increasing the lambda value strengthens the regularization effect.

- If λ is very large, then all weights end up very close to zero and the result is a flat line going through the data's mean.
- If $\lambda = 0$, it removes regularization completely, and then Ridge Regression is just Linear Regression.

2 Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow
source of example : https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization

:: What to do with high variance ?

L2 regularization *- continued*

How to choose the Regularization parameter lambada value ?

Ridge Regression cost function

$$J(\theta) = MSE(\theta) + \lambda \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

sum of the squares of all the feature weights

$$L_2\, regrlarization\, term = \theta_1^2 + \theta_2^2 + \cdots + \theta_n^2$$

The hyperparameter λ

When choosing a lambda value, the goal is to strike the right balance between simplicity and training-data fit:

- If your lambda value is too high, your model will be simple, but you run the risk of *underfitting* your data. Your model won't learn enough about the training data to make useful predictions.[1]
- If your lambda value is too low, your model will be more complex, and you run the risk of *overfitting* your data. Your model will learn too much about the particularities of the training data, and won't be able to generalize to new data.[2]

The ideal value of lambda produces a model that generalizes well to new, previously unseen data. Unfortunately, that ideal value of lambda is data-dependent, so you'll need to do some tuning.[3]

1,2,3 : : https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization

:: What to do with high variance ?

L1 regularization

L1 regularization uses L1-norm penalty (L1 is just the absolute sum of the weights). Unlike L2, L1 will push certain weights to be exactly 0. An important characteristic of L1 regularization is that it tends to completely eliminate the weights of the least important features (i.e., set them to zero). In other words, L1 regularization automatically performs feature selection and outputs a *sparse model* (i.e., with few nonzero feature weights)[2].

Lasso regression is a regularized version of Linear Regression, it uses L1-norm penalty. The cost function of Lasso regression with regularization term is as follow,

$$J(\theta) = MSE(\theta) + \lambda \frac{1}{2} \sum_{i=1}^{n} |\theta_i|$$

sum of absolute value of all the feature weights

The hyperparameter λ

2 Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow

:: What to do with high variance ?

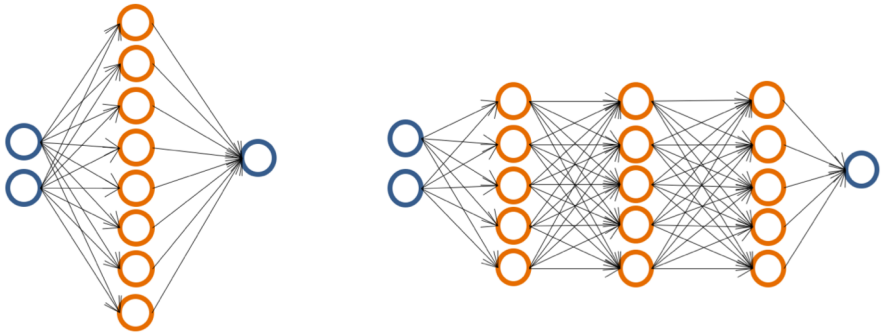
Use regularization to address overfitting in neural network

“Small” neural network

- fewer parameters
- computationally cheaper
- more prone to underfitting;

“Large” neural network

- more parameters
- computationally more expensive
- more prone to overfitting



One of effective ways to address overfitting is regularization

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

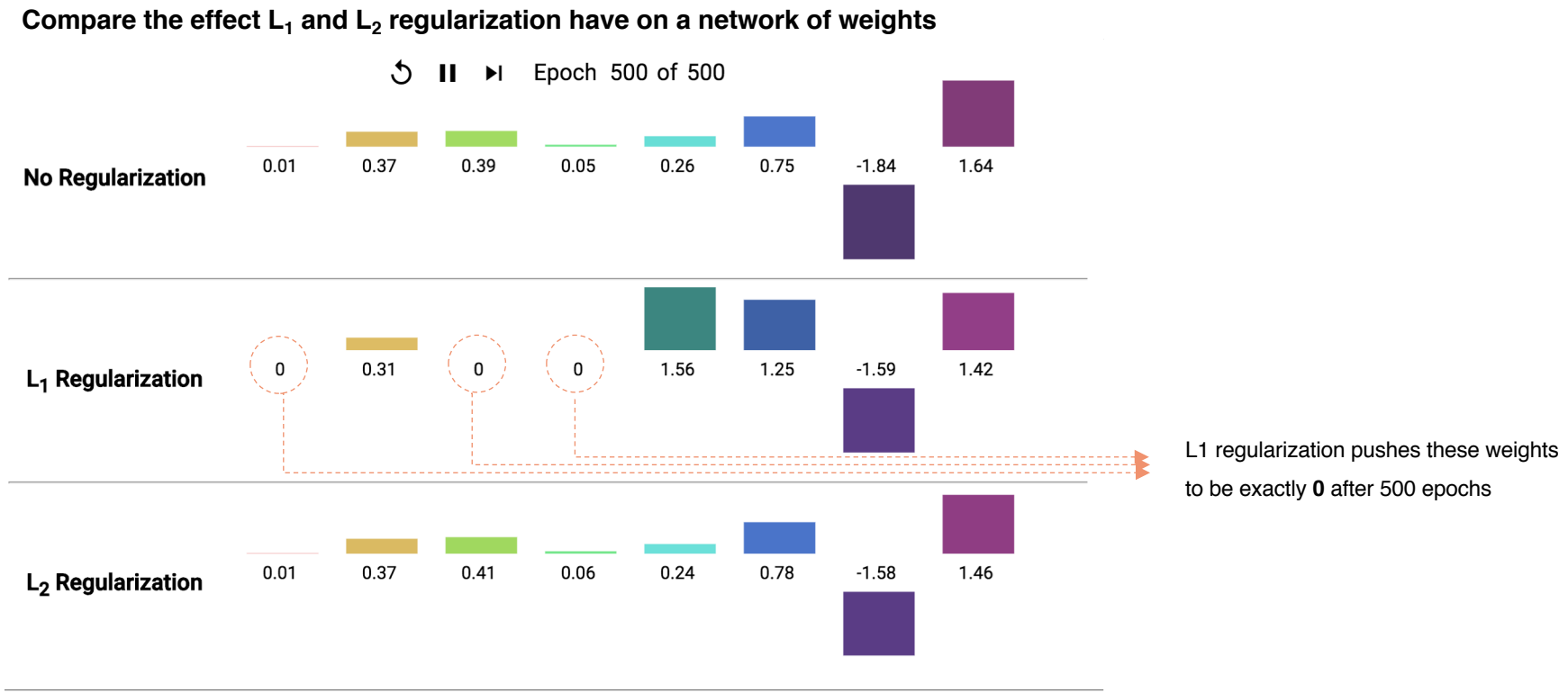
Regularization term

Source:Andrew Ng, https://www.youtube.com/watch?v=0twSSFZN9Mc
Source:Andrew Ng ,https://www.youtube.com/watch?v=yoYA1MFpYRg

:: What to do with high variance ?

Use regularization to address overfitting in neural network *- continued*

Let's have a look at an example of comparing L1 and L2 regularization. The following screenshot shows the final effect after 500 epochs.



You can view the animation from the link below
https://developers.google.com/machine-learning/crash-course/regularization-for-sparsity/l1-regularization

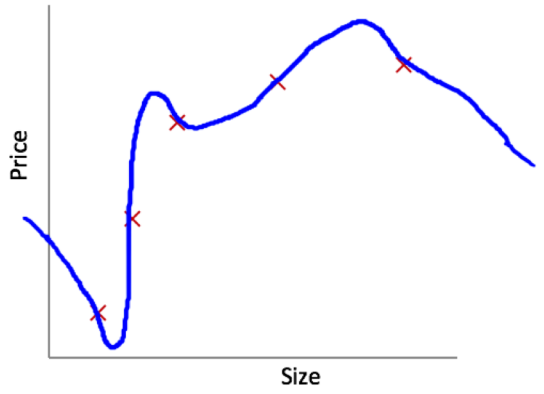
:: What to do with high variance ?

Reduce the number of features

In the following regression example for predicting house price, the overfitting is not unique issue to polynomial regression, but also a general issue if there are lots of features but with very little training data[1].

Overfitting because of higher order polynomials

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$



Overfitting with too many features : $x_1, x_2, \ldots, x_{100}$

$x_1 =$ size of house

$x_2 =$ no. of bedrooms

$x_3 =$ no. of floors

$x_4 =$ age of house

$x_5 =$ average income in neighborhood

$x_6 =$ kitchen size

$\vdots$

x_{100}

If we have too many features, the learned hypothesis may fit the training set very well but fail to generalize to new examples (predict prices on new examples)[5].

Use less features

One of main solutions for addressing overfitting is to reduce the number of features:

- Manually select which features to keep[2]
- Automatically select which features to keep or throw out[3]

The disadvantage is that by throwing away some of features is also throwing away some of information you have about the problem[4].

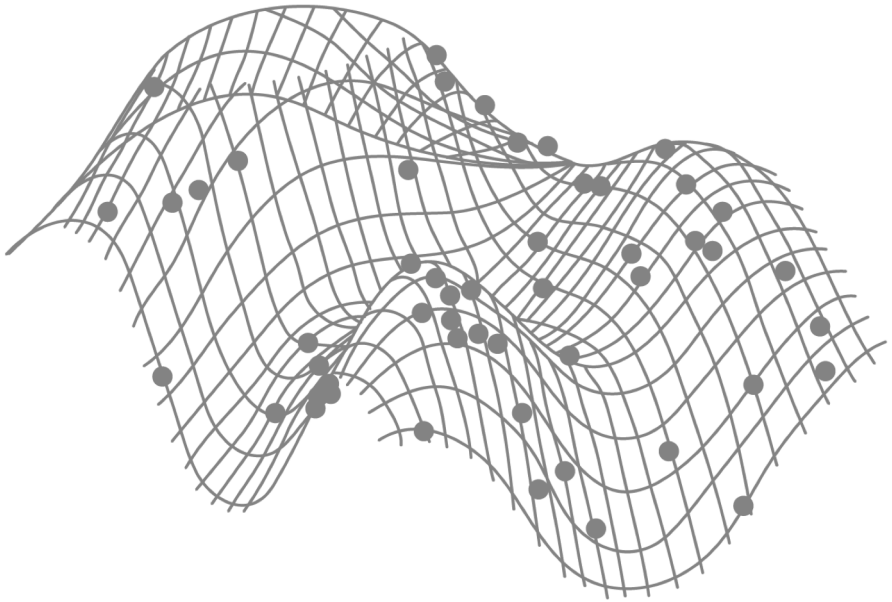
1 source: https://www.youtube.com/watch?v=F7Y4rhwtG-c&list=PLb6mIWj-UCxZTR_xOHyoysTluOjLIS2AV&index=66
2,3 ,4 &5 Source: Andrew Ng,https://www.youtube.com/watch?v=u73PU6Qwl1I

:: Tuning the Hyperparameter

Hyper-parameter tuning is an iterative process

You begin by setting parameters based on a “best guess” of the outcome. Your goal is to find the “best possible” values— those that yield the best model. As you adjust parameters and model performance begins to improve, you see which parameter settings are effective and which still require tuning. [1]

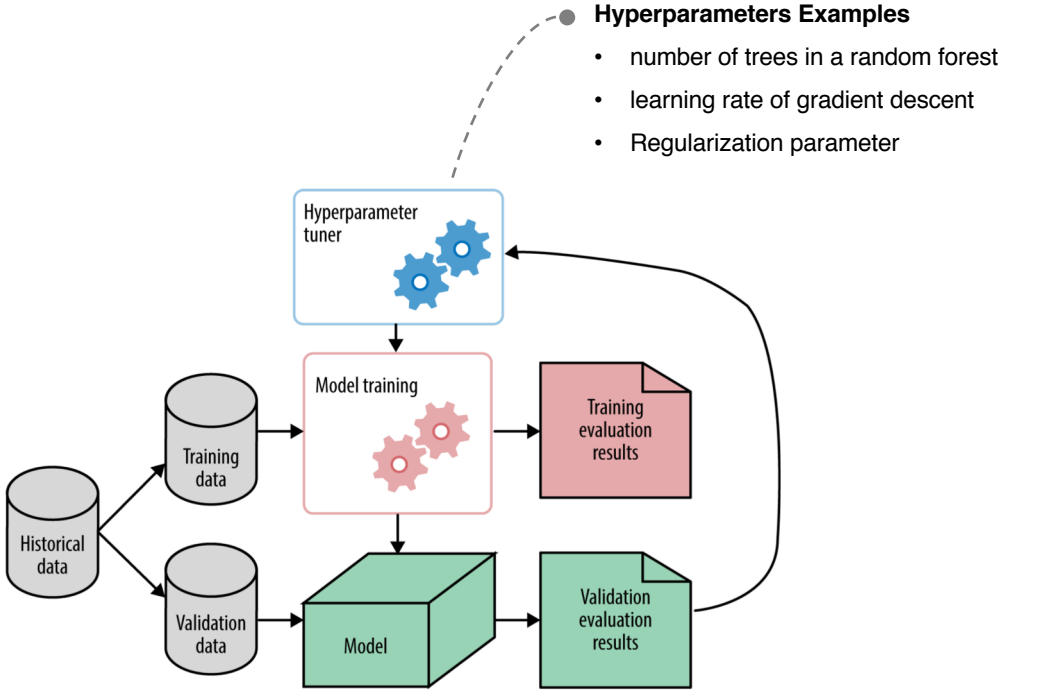
“ A simple algorithm with well-tuned parameters often produces a better model than an inadequately tuned complex algorithm. ” [2]



Source: mathworks, Applying Supervised Learning

:: Tuning the Hyperparameter - *continued*

Hyperparameters control how a machine learning algorithm fits the model to the data



The results from the cross validation are passed back to the hyperparameter tuner, which tweaks some knobs and trains the model again.

- **Hyperparameters control the model complexity**

  A regularization hyperparameter controls the *capacity* of the model, i.e., how flexible the model is, how many degrees of freedom it has in fitting the data. Proper control of model capacity can prevent overfitting, which happens when the model is too flexible, and the training process adapts too much to the training data, thereby losing predictive accuracy on new test data. So a proper setting of the hyperparameters is important[1].

- **Hyperparameters control the behavior of the training algorithm**

  Another type of hyperparameter comes from the training process itself. Training a machine learning model often involves optimizing a loss function (the training metric). A number of mathematical optimization techniques may be employed, some of them having parameters of their own. For instance, stochastic gradient descent optimization requires a learning rate or a learning schedule. Some optimization methods require a convergence threshold. Random forests and boosted decision trees require knowing the number of total trees. These also need to be set to reasonable values in order for the training process to find a good model[2].

1 ,2 & Diagram Source: Alice Zheng , Evaluating Machine Learning Models

:: Tuning the Hyperparameter - *continued*

Hyperparameters tuning is the process of looking for the most optimal hyperparameters for an algorithm

Hyperparameters are specified before the training algorithm starts and can't be optimized inside the training algorithm itself. The first choice of hyperparameters will not result in optimal model performance, so hyperparameters are tuned by adjusting parameters and taking another step of training the model – essentially an optimization procedure by itself.[1]

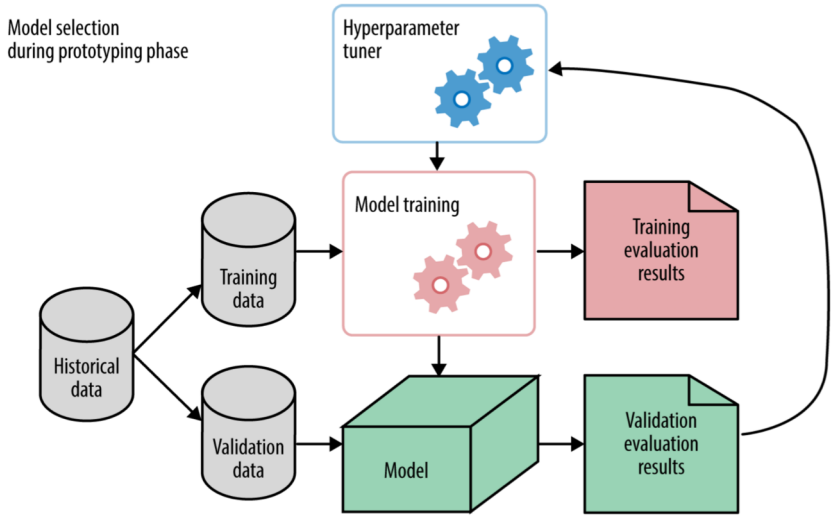
- **Hyperparameter tuning is a "meta" process that controls the training process**

  Hyperparameter tuning is the process of finding "best possible" hyperparameters that yield the best model; Hyperparameters are important because they directly control the behavior of the training algorithm and have a significant effect on the performance of the model being trained[2].

- **Hyperparameter optimization / tuning is an iterative process.**

  You can tune hyperparameters manually. You can explore many combinations of hyperparameter values until you find a great combination. But this would be very tedious work. Instead you can have better approaches by using

  - Grid search
  - Random search
  - Bayesian Optimization and so on.

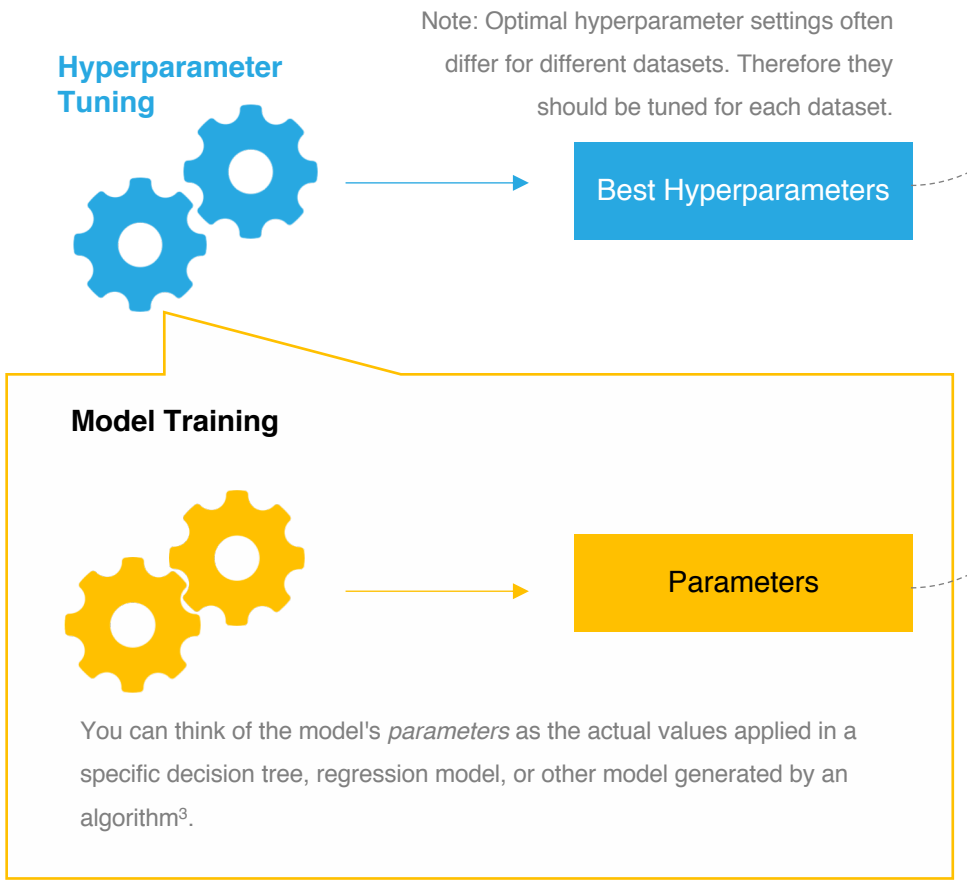


Source of Diagram: Alice Zheng , Evaluating Machine Learning Models

1,2 Source: https://arimo.com/data-science/2016/bayesian-optimization-hyperparameter-tuning/

:: Tuning the Hyperparameter - *continued*

What is the difference between Hyperparameter and Parameter?



Hyperparameters[1]

Hyperparameters are specified before the training algorithm starts and can't be optimized inside the training algorithm itself. They are external to the model.

- They are often used in processes to help estimate model parameters.
- They can not be estimated from data.
- They are often specified by the practitioner.
- They usually don't changes during a training job.

Model parameters[2]

Model's parameters are the variables that your chosen machine learning technique uses to adjust to your data. They are internal to the model.

- They are required by the model when making predictions.
- They are estimated or learned from data.
- They are often not set manually by the practitioner.
- The values are usually chosen by a learning algorithm, like the weights in linear regression or a neural network
- They changes during the a training job.

1 &2 Source: https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/
3Source: https://msdn.microsoft.com/en-us/library/dn905810.aspx

:: Tuning the Hyperparameter - *continued*

Example of Hyperparameters and Parameters

Hyperparameter examples

Neural networks

- Learning rate
- Number of layers
- Number of hidden units
- Type of unit
- Mini-batch size, etc

SVM

- C, Kernel and gamma

Lasso, Ridge Regression

- Regularization parameter

K-Means

- Number of clusters

Parameter examples

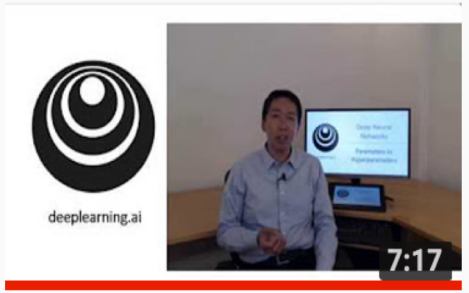
Neural networks

- Weight

Linear Regression

- $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}$ ...

Parameters vs Hyperparameters



Youtube Video (7 min)
https://www.youtube.com/watch?v=VTE2KlfoO3Q

:: Tuning the Hyperparameter *- continued*

The common hyperparameter optimization algorithms

- **Random Search is better solution than Grid Search**
- **Bayesian optimization looks promising, but it is not yet sufficiently mature or reliable.**

  In practice, Bayesian optimization has been shown to obtain better results in fewer experiments than grid search and random search[1]. However, " *Currently, we cannot unambiguously recommend Bayesian hyperparameter optimization as an established tool for achieving better deep learning results or for obtaining those results with less effort. Bayesian hyperparameter optimization sometimes performs comparably to human experts, sometimes better, but fails catastrophically on other problems. It may be worth trying to see if it works on a particular problem but is not yet sufficiently mature or reliable* "[5]

Grid Search	Random Search	Bayesian Optimization
A brute force search of every combination of hyperparameters[2]	Randomly sample and evaluate sets of hyperparameters specified by a probability distribution[3]	Use a statistical mode to help choose which set of hyperparameters to explore next based on the performance of past sets of hyperparameters[4]

1, 2 &3 Source :http://www.nersc.gov/users/data-analytics/data-analytics-2/deep-learning/hyperparameter-o/
1 Source: https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning)#Bayesian
5 source: Ian Goodfellow,et al, Deep Learning,

:: Tuning the Hyperparameter *- continued*

Grid search tries the exhaustive searches for all the possible hyperparameter combinations

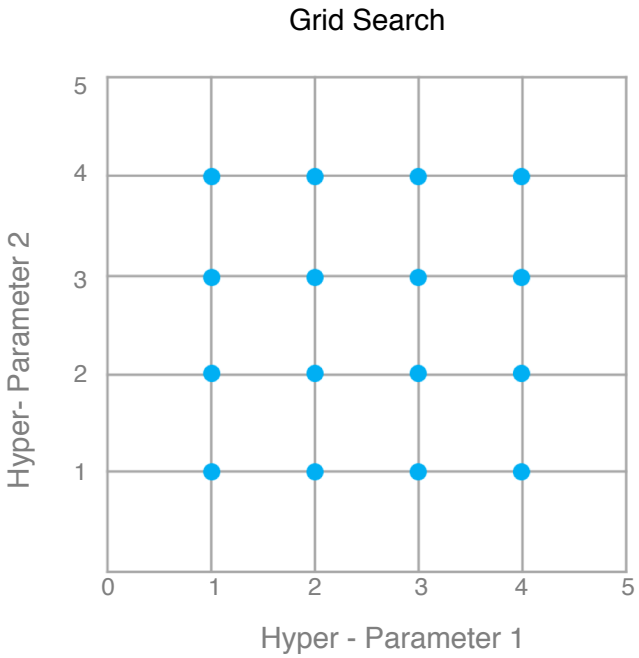
- **Form a grid of hyper-parameter values**

  In practice, practitioners specify the bounds and steps between values of the hyperparameters, so that it forms a grid of configurations. we try a set of configurations of hyperparameters and train the algorithm accordingly, choosing the hyperparameter configuration that gives the best performance

- **Grid search is a costly and time-consuming approach.**

  This method works ok when the number of hyperparameters was relatively small. It doesn't work in cases like neural networks.

  Imagine that we need to optimize 5 parameters. Let's assume, for simplicity, that we want to try 10 different values per each parameter. Therefore, we need to make 100,000 (105105) evaluations. Assuming that network trains 10 minutes on average we will have finished hyperparameter tuning in almost 2 years[2].



2 Source: http://neupy.com/2016/12/17/hyperparameter_optimization_for_neural_networks.html

:: Tuning the Hyperparameter *- continued*

Prefer random search to grid search especially when hyperparameter search space is large

- **Try random value, don't use a grid**

  Instead of trying all possible combinations we will just use randomly selected subset of the parameters.

  It has been found to be more effective in high-dimensional spaces than exhaustive search. This is because oftentimes, it turns out some hyperparameters do not significantly affect the loss[1].

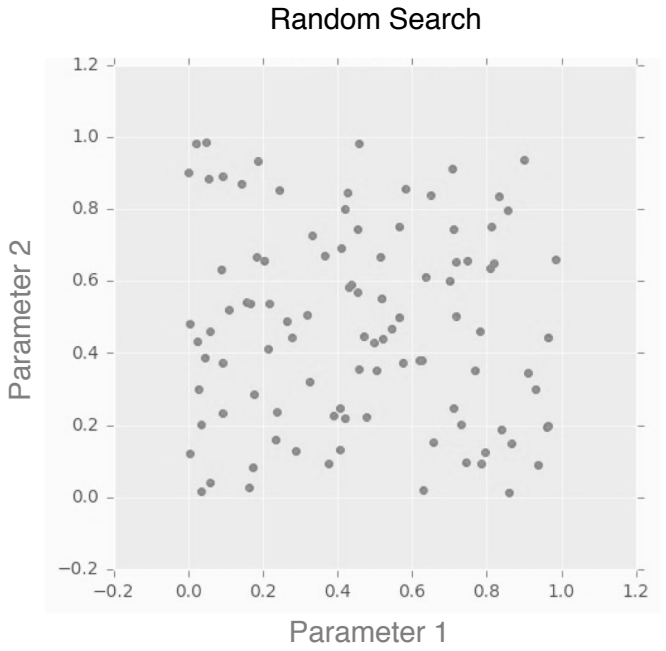
- **Use a coarse to fine sampling scheme**

  In practice, it can be helpful to first search in coarse ranges (e.g. 10 ** [-6, 1]), and then depending on where the best results are turning up, narrow the range. [2].

For more details, please check out on YouTube video:

Tuning process by Andrew Ng (7 mins)

https://www.youtube.com/watch?v=AXDByU3D1hA



It is very often the case that some of the hyperparameters matter much more than others. Performing random search rather than grid search allows you to much more precisely discover good values for the important ones.

1 Source: https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning) 2 Source: http://cs231n.github.io/neural-networks-3/

:: Tuning the Hyperparameter - *continued*

// Beginner can skip this page

Bayesian optimization approaches looks promising among the current optimization strategies

- **This process is sequential process.**

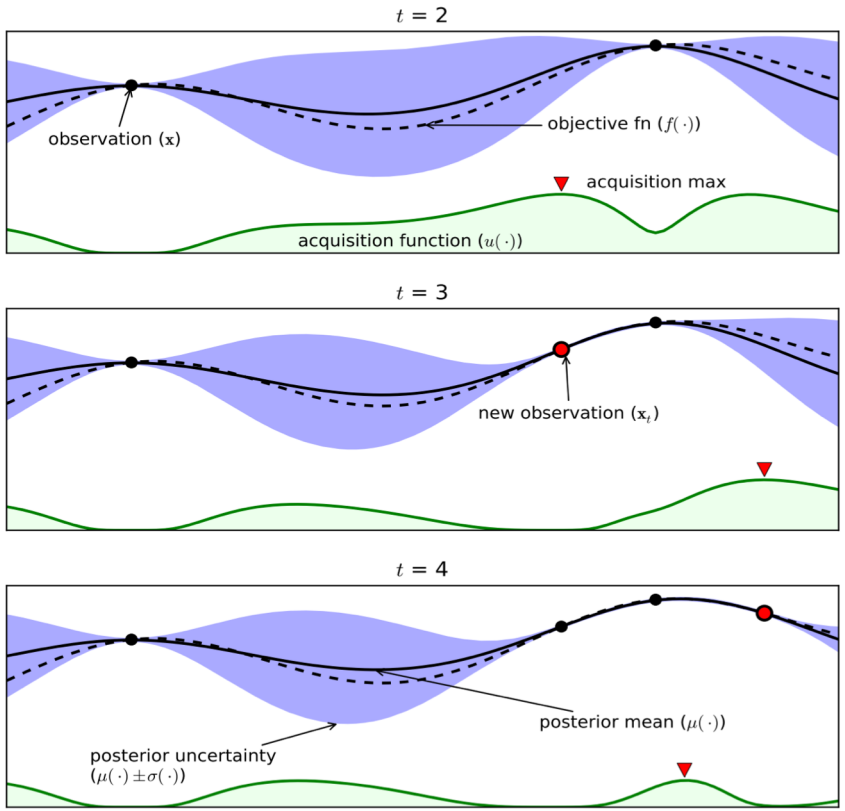
  The idea behind Bayesian optimization is to exploit previous experiments to build a model/approximator of the unknown objective function and to optimize this model to determine the next experiment.[1] This process attempts to minimize the number of steps required to find a combination of parameters that are close to the optimal combination.

  Bayesian optimization uses a surrogate function to approximate the true blackbox function that are extremely expensive to evaluate. Bayesian optimization uses a Gaussian Process to model the surrogate, and typically optimizes the Expected Improvement, which is the expected probability that new trials will improve upon the current best observation. Using Gaussian process, one can compute the Expected Improvement of any point in the search space. The one gives the highest expected improvement will be tried next[2].

- **How does it work ?**

  Bayesian optimization works by constructing a posterior distribution of functions (Gaussian Process) that best describes the function you want to optimize. As the number of observations grows, the posterior distribution improves, and the algorithm becomes more certain of which regions in parameter space are worth exploring and which are not.

  As you iterate over and over, the algorithm balances its needs of exploration and exploitation taking into account what it knows about the target function[3].



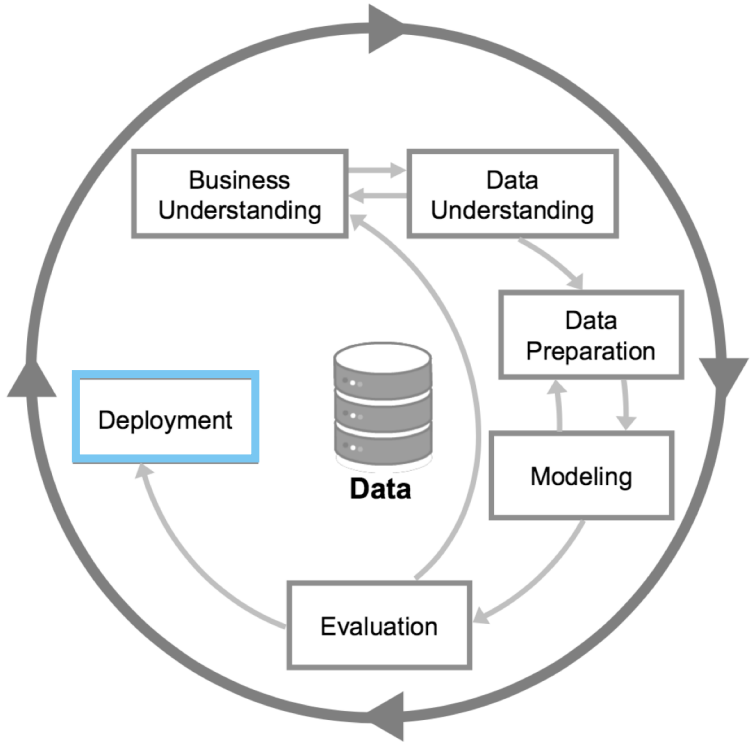
The figures show a Gaussian process (GP) approximation of the objective function over four iterations of sampled values of the objective function. The figure also shows the acquisition function in the lower shaded plots [4].

- Gaussian Process uses a set of previously evaluated parameters and resulting accuracy to make an assumption about unobserved parameters.
- Acquisition Function using this information suggest the next set of parameters.

1source: http://wp.doc.ic.ac.uk/sml/project/bayesian-optimization 2 Source: https://arimo.com/data-science/2016/bayesian-optimization-hyperparameter-tuning/ 3Source: https://github.com/fmfn/BayesianOptimization 4 Source: https://arxiv.org/pdf/1012.2599v1.pdf

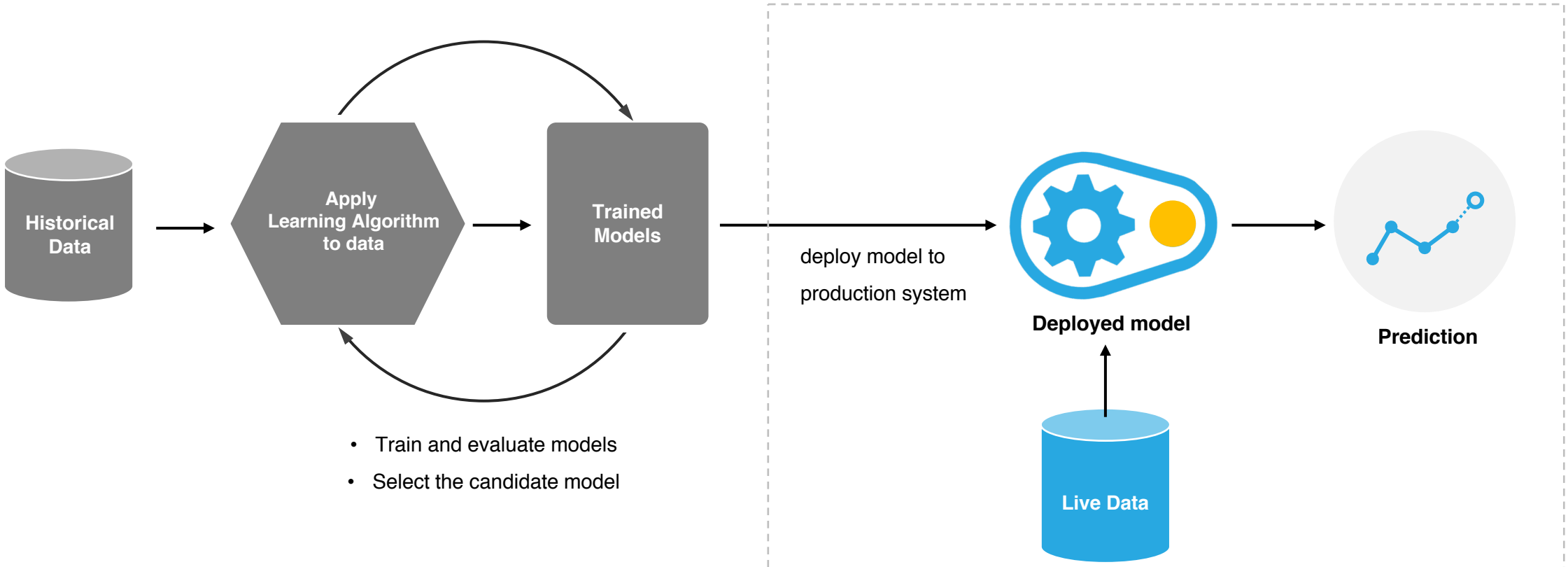
07 | Model Deployment

:: Deploy the model into production



Source: https://www.the-modeling-agency.com/crisp-dm.pdf

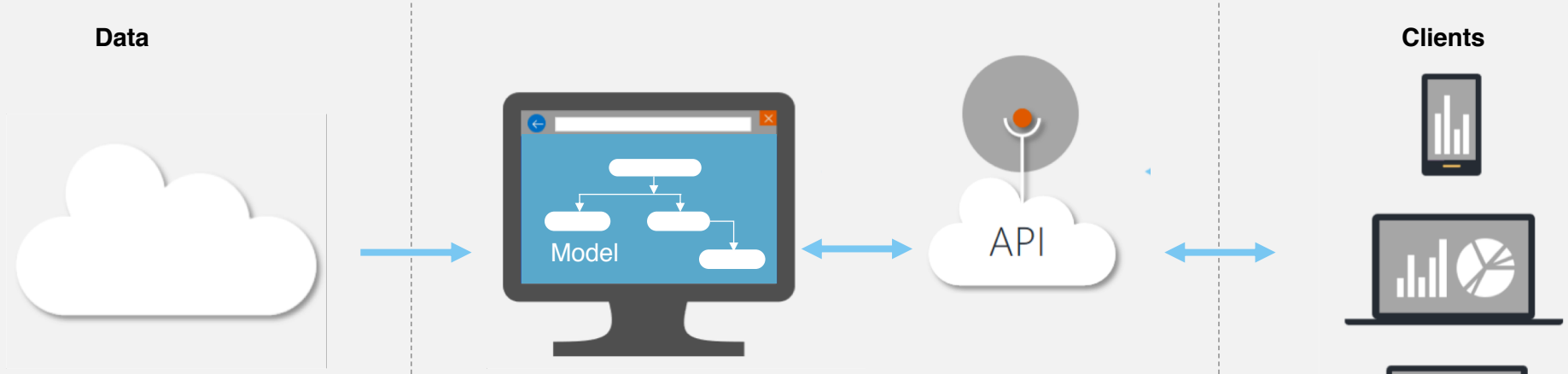
:: Deployment phase in machine learning workflow



- Train and evaluate models
- Select the candidate model

During training, you apply the model to known data to find the right settings to get the best results. When your results are good enough for the needs of your application, you should deploy the model to whatever system your application uses and test it.

:: Deploy the trained models as web service



Deploy model as Web Service [1]

You can deploy the model as a web service. Using the web service, users can send data to your model and the model will return its predictions.

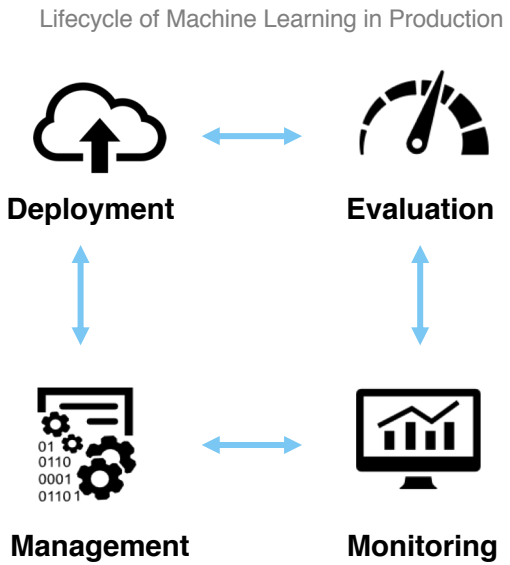
Once you deploy a Machine Learning predictive model as a Web service, you can use a REST API to send it data and get predictions. You can send the data in real-time or in batch mode.

Consume Web Service [2]

The Web service can then be consumed in web sites, business application, dashboards, and mobile apps.

Source of diagram: https://biz-excellence.com/2016/09/13/machine-learning-dt/
1&2 Source: https://docs.microsoft.com/en-us/azure/machine-learning/studio/publish-a-machine-learning-web-service

:: What happens after (initial) deployment ?



Evaluation

Measuring quality of deployed models

Monitoring

Track model quality over time ！ The models have been deployed into business environment must be regularly monitored for acceptable performance. Remember - Model performance will reduce over time.

Management

Choosing between deployed models.

Improving deployed models with feedback

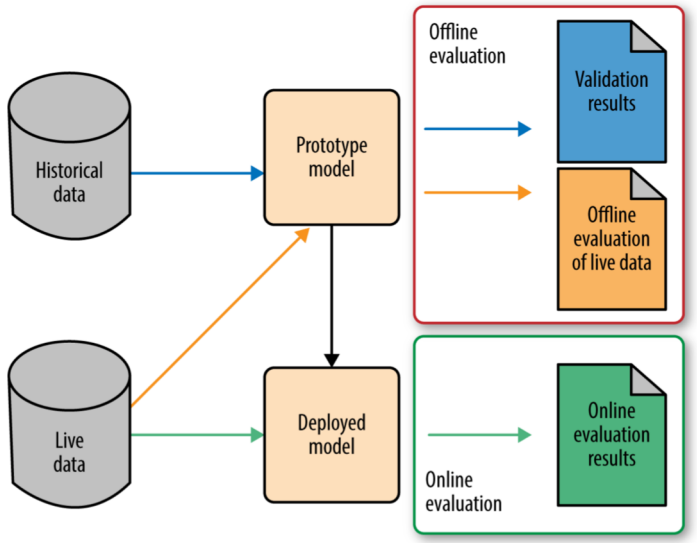
Source of diagram: Alice Zheng, Deploying and Managing Machine Learning Models

:: When/how to evaluate Machine Learning Model ?

Offline evaluation

- Offline evaluation measures offline metrics of the prototyped model on historical data (and sometimes on live data as well).
- Offline evaluation might use one of the metrics like accuracy , precision- recall, AUC, log-loss ,RMSE, etc.

Machine learning model development and evaluation workflow



Roughly speaking, the first phase involves *prototyping*, where we try out different models to find the best one (*model selection*). Once we are satisfied with a prototype model, we deploy it into production, where it will go through further testing on live data.

Source of the content and diagram: Alice Zheng, Evaluating Machine Learning Models

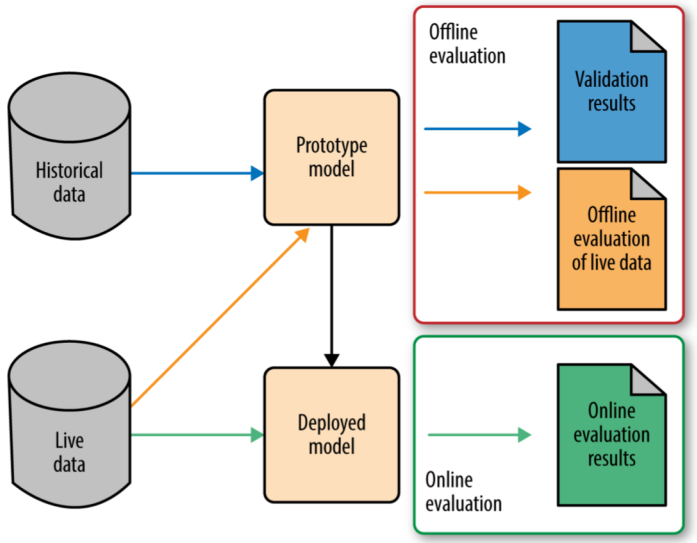
:: When/how to evaluate Machine Learning Model ?

Online evaluation

- **Online evaluation measures live metrics of the deployed model on live data**
- **Offline metric is not equal to business metric**

  Online and offline evaluations may measure very different metrics. Online evaluation might measure business metrics such as Customer Lifetime Value, Click Through Rate which may not be available on historical data but are closer to what your business really cares about.
- **The online phase has its own testing procedure**
  - The most commonly used form of online testing is A/B testing, which is based on statistical hypothesis testing.
  - Multiarmed bandits is a better alternative to A/B tests in some situations
- **Track both business and ML metrics to see if they correlate**

Machine learning model development and evaluation workflow



Roughly speaking, the first phase involves *prototyping*, where we try out different models to find the best one (*model selection*). Once we are satisfied with a prototype model, we deploy it into production, where it will go through further testing on live data.

Source of the content and diagram: Alice Zheng, Evaluating Machine Learning Models

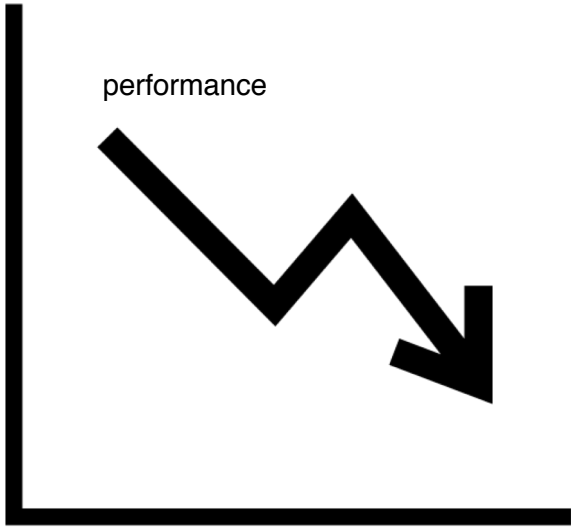
:: Update/Re-train models

Retrain the models which don't perform well over time

- **Track statistics of data over time**
- **Monitor both offline and online metrics**

  To catch distribution drift (namely, the distribution of data changes over time), it's a good idea to monitor the offline metric (used for evaluations during offline testing/prototyping) on live data, in addition to online testing.

  Monitoring for distribution drift is often done "offline" from the production environment. Hence we are grouping it into offline evaluation. If the offline metric changes significantly, then it is time to update the model by retraining on new data.
- **Update when offline metrics diverges from online metrics**



Why update ?

- Business environment changes over time
- Data changes over time.
- Model performance drops over time

Part 2

The well-known algorithms

:: Now let's study some popular algorithms

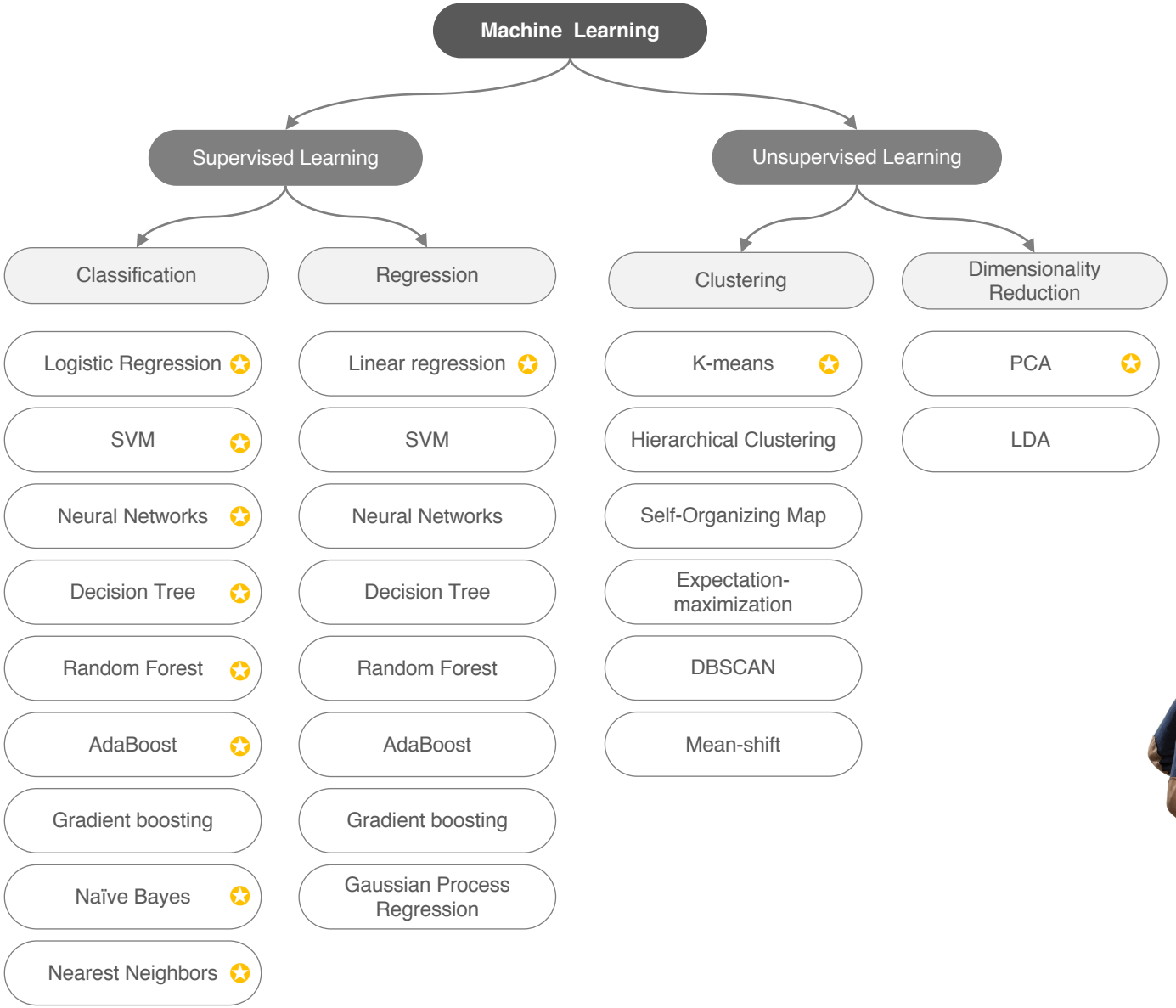


Image: Designed by Asierromero / Freepik