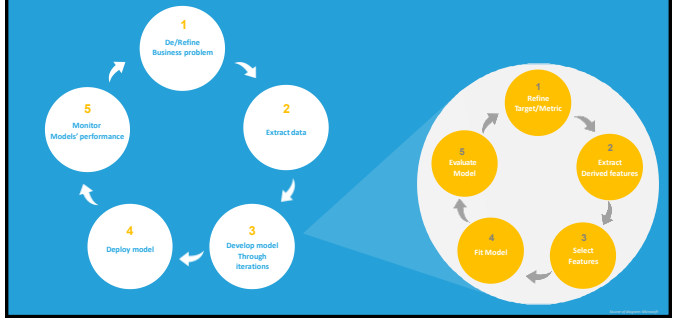
getting started with

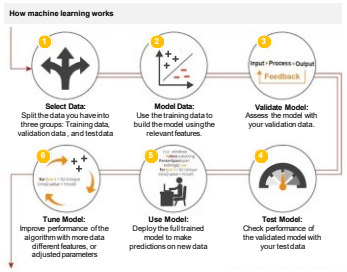
Machine Learning

:: Machines Learning uses historical data to make predictions

:: Steps to build a machine learning solution

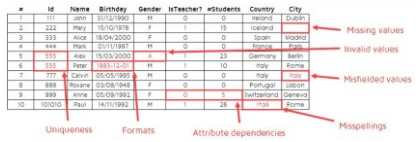


:: How machine learning works



:: Data in the real world is dirty

Data is rarely clean and often you can have data quality issues



The typical data quality issues that arise are:

- **Incomplete**: Data lacks attributes or containing missing values.
- **Noisy**: Data contains erroneous records or outliers.
- **Inconsistent**: Data contains conflicting records or discrepancies.

:: Data in the real world is dirty

What kind of issues affect the quality of data?

- **Invalid values**
  Some datasets have well-known values, e.g. gender must only have "F" (Female) and "M" (Male). In this case it's easy to detect wrong values.
- **Formats**
  The most common issue. It's possible to get values in different formats like a name written as "Name, Surname" or "Surname, Name".
- **Attribute dependencies**
  When the value of a feature depends on the value of another feature. For example, if we have some school data, the "number of students" is related to whether the person "is teacher?". If someone is not a teacher he/she can't have any students.
- **Uniqueness**
  It's possible to find repeated data in features that only allow unique values. For example, we can't have two products with the same identifier.
- **Missing values**
  Some features in the dataset may have blank or null values.
- **Misspellings**
  Incorrectly written values.
- **Misfielded values**
  When a feature contains the values of another.

:: Preprocessing data - Clean your data

Why to deal with missing values?

- Missing values in a dataset can be due to error or because observations that were not recorded
- When missing value are present, certain algorithms may not work or you may not have the desired result.
- Missing data affects some models more than others[1]
- Even for models that handle missing data, they can be sensitive to it (missing data for certain variables can result in poor predictions)[2]

Missing value is probably the most common problems in data mining/machine learning

:: Preprocessing data - Clean your data

How to deal with missing values?

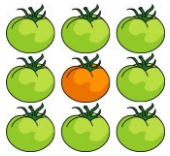
Typical missing value handling methods are:

- **Deletion**
  Remove records with missing values
- **Dummy substitution**
  Replace missing values with a dummy value: e.g, unknown for categorical or 0 for numerical values.
- **Mean substitution**
  If the missing data is numerical, replace the missing values with the mean.
- **Frequent substitution**
  If the missing data is categorical, replace the missing values with the most frequent item
- **Regression substitution**
  Use a regression method to replace missing values with regressed values.

:: Preprocessing data - Clean your data

What you should know about the outliers/anomalies

- Outliers may bring about problems by distorting the predictive model.
- What's an outlier is somewhat subjective.[1]
- Outliers can be very common in multidimensional data.[2]
- Some models are less sensitive (more robust) to outliers than others.[3]
- Outliers can be result of bad data collection, or they can legitimate extreme (or unusual) values.[4]
- Sometimes outliers are the interesting data points we want to model, and other times they just get in the way.[5]



An outlier is a data point that distinctly separate from the rest of the data.

:: Preprocessing data - Clean your data

Cause of outliers

Data from different classes

Data measurement and collection Errors

Natural variation

:: Preprocessing data - Clean your data

How to deal with outliers?

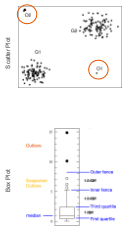
The choice of how to deal with an outlier should depend on the cause.

- **Keep outliers**
  Outliers should not necessarily be omitted from the analysis as they may be genuine observations in the data.

  In many applications, outliers provide crucial information. For example, in a credit card fraud detection app, they indicated purchases that fall outside a customer's usual buying patterns.[1]

- **Exclude outliers**
  There are two common approaches to exclude outliers[2]
  - **Trimming/Truncation:** Trimming discards the outliers
  - **Winsorising:** Winsorising replaces the outliers with the nearest "non-suspect" data

Popular plots for outlier detection:
Scatter Plot & **Box Plot**



:: Preprocessing data - Clean your data

Examples on how to deal with outliers

- **Example for Trimming**
  Eliminate the outliers "2" & "22"

- **Example for Winsorising**
  Assign outlier the next highest or lowest value found in the sample that is not an outlier. In this example, "10" & "14" are not outliers and used to replace the outliers "2" & "22".

- **Trimming or Winsorizing less than 5% of data points**
  It will not likely affect the hypothesis testing outcome.
- **Trimming or Winsorizing great than 5% of data points**
  Trimming or Winsorizing great than 5% may affect the outcome results.
  - Reduce the power of analysis
  - Makes samples less representative
  - May affect normalcy of data
  - Consider transforming data, choosing an alternate outcome variable or data analysis technique.

:: Preprocessing data - Data normalization

How to normalize data?

Data normalization re-scales numerical values to a specified range. Popular data normalization methods include:

- **Min-Max Normalization**:
  Linearly transform the data to a range, say between 0 and 1, where the min value is scaled to 0 and max value to 1.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- **Z-score Normalization** (or **Standardization**):
  Scale data based on mean and standard deviation: divide the difference between the data and the mean by the standard deviation. Feature standardization makes the values of each feature in the data have zero-mean (when subtracting the mean in the numerator) and unit-variance.

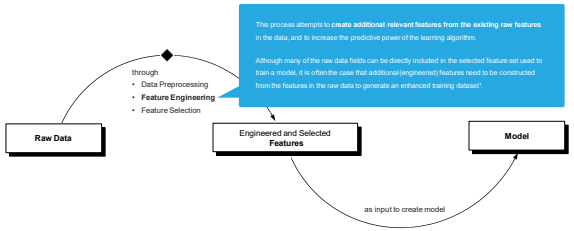
$$z = \frac{X - \mu}{\sigma} \quad \text{Z-Score} = \frac{\text{Value} - \text{mean}}{\text{Standard Deviation}}$$

- **Decimal scaling**:
  Scale the data by moving the decimal point of the attribute value.

Different scales

Different scales in a dataset may be problematic in some cases where certain machine algorithms require data to be in the same scale.

:: Feature Engineering can augment your data



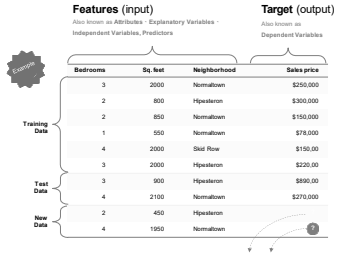
This process attempts to **create additional relevant features from the existing raw features** in the data, and to increase the predictive power of the learning algorithm.

Although many of the raw data fields can be directly included in the selected feature set used to train a model, it is often the case that additional (engineered) features need to be constructed from the features in the raw data to generate an enhanced training dataset[1].

Note: There is no formal definition of Feature Engineering. Someone include feature selection, even data preprocessing as a part of Feature Engineering.

- *In Google's definition, Feature engineering means transforming raw data into a feature vector*
- *In Microsoft's documentation feature engineering mainly focuses on feature construction.*

:: What's feature ?



This unknown value should be predicted by the model

a **feature** is an individual measurable property or characteristic of a phenomenon being observed[1].

In this example, there are a couple of features like bedrooms, house size, neighborhood which are used as input for the modeling. The target is the sales price of the house.

There are some known price for some houses (Training data & Test Data).

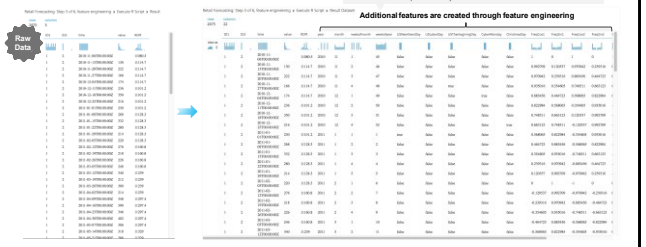
A model can be built to predict the the sales price of the houses which are still unknown.

In a nutshell, once we've trained our model with labeled examples, we use that model to predict the label on unlabeled examples.[2]

:: Feature Engineering - example

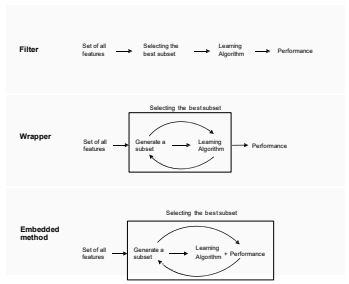
Example 2

- Date features: year, month, week of month, etc.
- Time features
- Season features
- Weekday-and-weekend features
- Holiday features: New Year, U.S. Labor Day, U.S. Thanksgiving, Cyber Monday, Christmas, etc.
- Fourier features to capture seasonality



The template provides two ID fields, which she associates with the store ID and product/SKU ID.

:: Summary: the modern approaches for Feature Selection



- Filter methods use statistical methods for evaluation of a subset of features while wrapper methods use cross validation.
- Filter methods are much faster compared to wrapper methods as they do not involve training the models. On the other hand, wrapper methods are computationally very expensive as well.
- Filter methods might fail to find the best subset of features in many occasions but wrapper methods can always provide the best subset of features.
- Using the subset of features from the wrapper methods make the model more prone to overfitting as compared to using subset of features from the filter methods.
- Embedded methods combine the qualities' of filter and wrapper methods. It's implemented by algorithms that have their own built-in feature selection methods.

:: Feature Engineering - example

The comparison of the performance results of the four models which are based on 4 training sets

Features	Mean Absolute Error	Root Mean Squared Error	Coefficient of Determination
Feature Set A (training set1) baseline: weather + holiday + weekday + weekend features for the predicted day	89.7	124.9	0.6
Feature Set A+B (training set2) baseline + previous 12 hours demand	51.7	88.3	0.8
Feature Set A+B+C (training set3) baseline + previous 12 hours demand + previous 12 days at the same hour	47.6	81.1	0.8
Feature Set A+B+C+D (training set4) baseline + previous 12 hours demand + previous 12 days at the same hour + previous 12 weeks at the same hour and the same day demand	48.3	82.1	0.8

Same algorithm but different performance

we used the Boosted Decision Tree Regression module, a commonly used nonlinear algorithm, to build the models.

To understand the performance of four models, see the comparison results in the following table.

- The best results were from the combination of features A+B+C and A+B+C+D.
- Feature set D does not provide additional improvement over A+B+C.

:: Feature Selection vs Dimension Reduction

Dimensionality reduction methods employ engineered features that can transform the original features and thus modify them.

Examples of dimensionality reduction methods include Principal Component Analysis, canonical correlation analysis, and Singular Value Decomposition. [1]

Although feature selection does seek to reduce the number of features in the dataset used to train the model, it is not usually referred to by the term "dimensionality reduction".

Feature selection methods extract a subset of original features in the data without changing them. [1]

Modelling

Table of contents

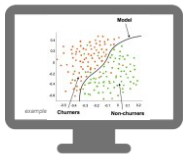
:: Train the model

The modelling is an iterative process of building model which is a combination of data structure, algorithm, and mathematics to captures the relationship between features and target within the dataset

:: What's a model ?

A model defines the relationship between features and label (i.e. target) [1]

Machine learning uses a model to capture the relationship between **feature vectors** and some **target variables** within a training data set.
A feature vector is a set of features or attributes that characterize a particular object, such as the number of bedrooms, bathrooms, and location of an apartment. The target is either a scalar value like rent price, or it's an integer classification such as "creditworthy" or "it's not cancer." [2]

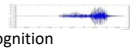


"All models are wrong, but some are useful."
– George Box

- *An machine learning model is a mathematical model that generates predictions by finding patterns in your data [3].*
- *At a high level, a model is a simplification of something more complex [4].*
- *A machine learning algorithm use data to automatically learn the rules. It simplifies the complexity of the data into relationships described by rules [5].*
- *Modelling is part art and part science [6].*

Machine Learning is to learn a project function for the task

- Speech Recognition

  $f($  $)=$ "How are you"
- Image Recognition

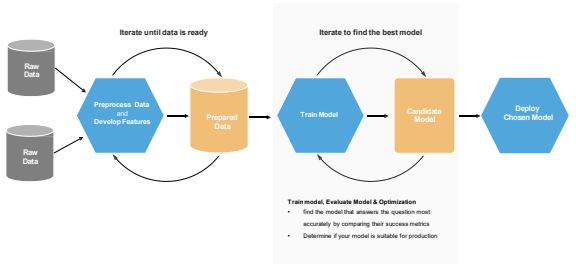
  $f($  $)=$ "Cat"
- Playing Go

  $f($  $)=$ "5-5" (next move)
- Dialogue System

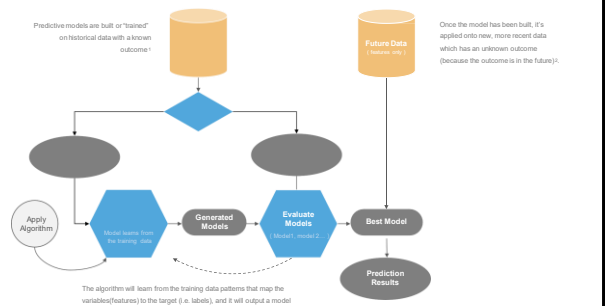
  $f($ "How are you?" $)=$ "I am fine."

  (what the user said) (system response)

:: The process of Machine Learning



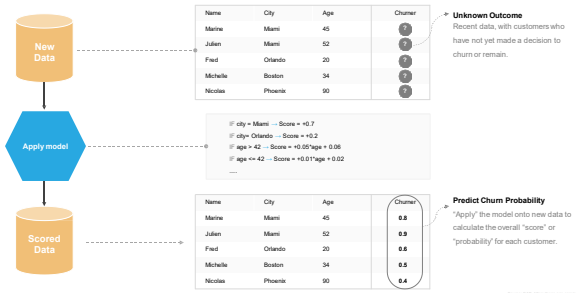
:: How to develop a model ?



:: Example of building the model (Learning phase)

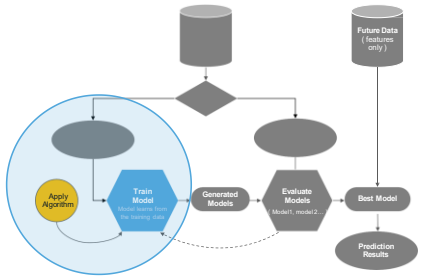
Predict if a customer is going to switch to another supplier

:: Example of using the model (Applying phase)



:: Train the model

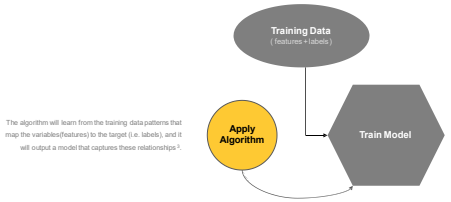
Let's take a closer look at the model training



:: Train a model using algorithms and training data

Machine Learning uses algorithms to iteratively learn from data

Algorithm is a self-contained set of rules used to solve problems through data processing, math, or automated reasoning. Machine learning algorithms use computational methods to "learn" information directly from data without relying on a predetermined equation as a model.



:: Let's take a look at an example for regression problem

Regression algorithm is a major types of machine learning algorithms

Let's take a look at a very simple regression algorithm now.

:: Let's take a look at an example for regression problem

Example: Predict house's price using linear regression

Suppose we have a dataset giving the living areas and prices of 21,613 houses from House Sales in King County, USA. Given data like this, we can learn to predict the prices of other houses in King County.

Living Area (Feet²)	Price ($)
1180	221,900
2570	538,000
770	180,000
1960	604,000
1680	510,000
5420	1,225,000
1715	257,500
1060	291,850
1780	229,500
1890	323,000
3560	662,500
1160	468,000
1430	310,000
1370	400,000
1810	530,000
...	...
x	y

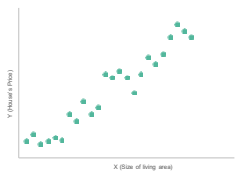
Data set

Picture of house. Designed by Freepik

:: Example: Train a model using Linear regression algorithm

Use linear regression algorithm to approximate the relationship between x and y

Take linear regression as example, the algorithm is trying to find a best-fit line to represent the relationship between the input feature x and target y.



Regression line (model) can be presented as :

The straight line can be seen in the plot, showing how linear regression attempts to draw a straight line that will best minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation.

:: Sample python code for building a very simple model

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model

# Load the diabetes dataset
diabetes = datasets.load_diabetes()


# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes.target[:-20]
diabetes_y_test = diabetes.target[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean squared error
print("Mean squared error: %.2f"
      % np.mean((regr.predict(diabetes_X_test) - diabetes_y_test) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regr.score(diabetes_X_test, diabetes_y_test))

# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')
plt.plot(diabetes_X_test, regr.predict(diabetes_X_test), color='blue',
         linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```

- A model is built based on Linear regression algorithm.
- The model is trained using the training dataset

Important:

Developing a model is a process of experimentation and incremental adjustment as much as it is of applying algorithms to a problem.

You should expect to spend a lot of time refining and modifying your model to get the best results.

It's very important that you establish a threshold of success for your model before you begin because otherwise you may not know when to stop refining.

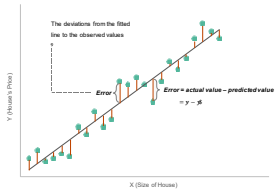
http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py

Source: https://cloud.google.com/ml-engine/docs/tensorflow/ml-solutions-overview

:: Usually there is deviation between the actual value and prediction

The deviations indicate how bad the model's prediction was on the training examples

Loss (i.e. error) is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater. [1]



Mean square error (MSE) is a commonly-used function to measure how large the loss is. It's called as **Loss function** or **Cost function**.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$\hat{y}_i$ is the prediction
y_i is the actual value

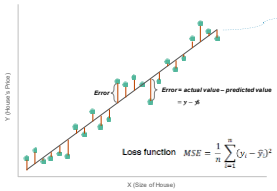
Mean square error (MSE) is the average squared loss per example over the whole dataset. We will elaborate the details of loss function later on.

The smaller the Mean square error, the better the fit of the line to the data.

:: Train the model to minimize the loss/error

Training the model is an iterative process of finding the "best" parameters to minimize the error

Training a model simply means learning (determining) good values for all the parameters of the model from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss. [1]



Model:

$y = \theta_0 + \theta_1 x$

parameter

The goal of training a model is to find a set of parameters that have low loss, on average, across all examples. [2]

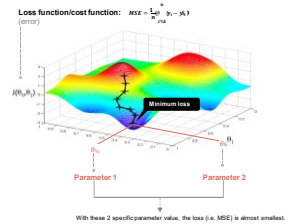
In this linear regression example, the goal of the training is to find estimated value for the parameters θ_0 and θ_1 which would provide the "best" fit for the data points within the training set.

If you don't understand the linear regression equation, don't worry about it. You can find the detailed explanation in *Part 2 – popular algorithms*

:: How does the model find the "best" parameters ?

Gradient Descent is one of the most common algorithms to find the good parameters

A Machine Learning model is trained by starting with an initial guess for the parameters (e.g. weights and bias in neural network) and iteratively adjusting those guesses until learning parameters with the lowest possible loss [1]



Usually, you iterate until overall loss stops changing or at least changes extremely slowly.

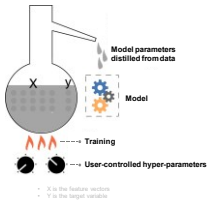
When that happens, we say that the model has **converged**. [3]

If you don't get it, don't worry. I will explain the details later in the *Part 2 – popular algorithms*

:: The model has parameters and hyper-parameters

Iteratively tuning the Hyperparameter so that the model can learn the "best" Parameters from data

The hyper-parameters are specified by the developer/data scientist while parameters are computed from the data via the algorithms.



- X is the feature vectors
- Y is the target variable

- 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 estimated or learned from data. They are often not set manually by the practitioner.
- Hyperparameters control how a machine learning algorithm fits the model to the data. Hyper-parameters are specified by the programmer, not computed from the training data, and are often used to tune a model to improve accuracy for a particular data set.

The examples of hyper-parameters:

- Number of layers, learning-rate in Neural network
- Number of trees in Random forest

The detailed explanation of the difference will be elaborated later.

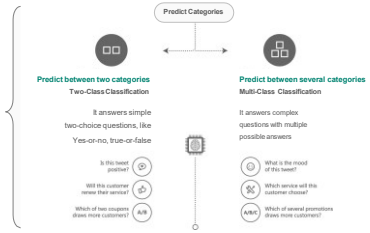
:: Let's take a look at another example for classification problem

Classification algorithm is another major types of machine learning algorithms

Just now we quickly go over the regression model example, now let's move to the example of classification.

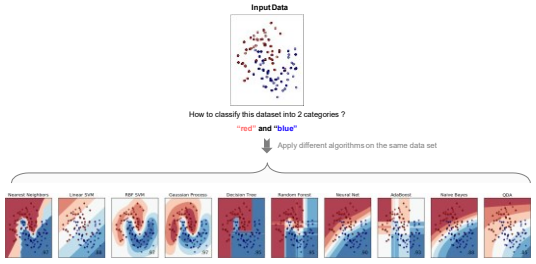
:: Let's take a look at an example for classification problem

Classification algorithm is another major types of machine learning algorithms



:: Example: how to classify the data points ?

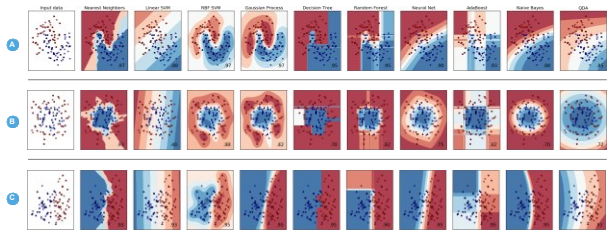
How to classify this dataset into 2 categories ?



:: Example: how to classify the data points ?

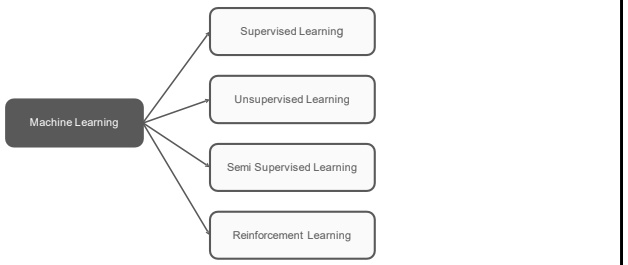
Use different algorithms to classify the data set.

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

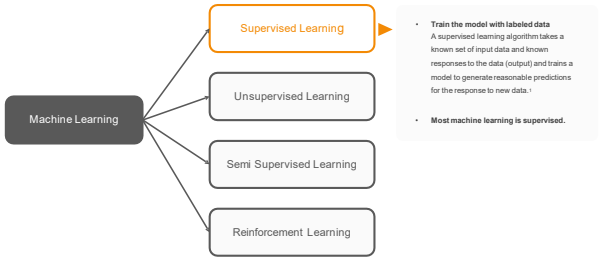


There is a wide range of algorithms …

:: The categories of machine learning algorithms

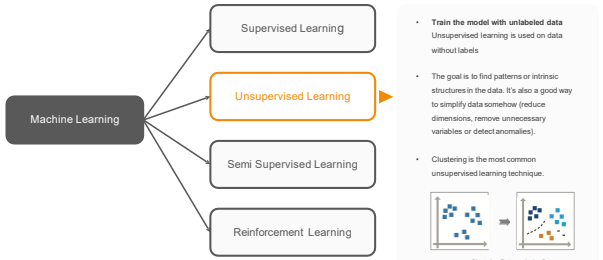


:: The categories of machine learning algorithms



- **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.**

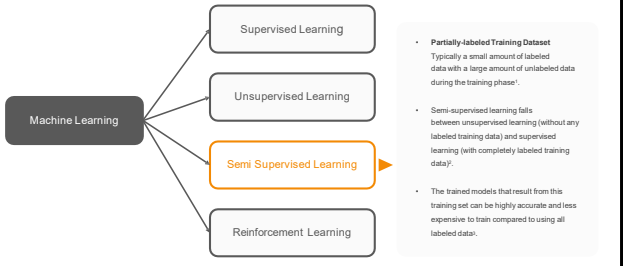
:: The categories of machine learning algorithms



- **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.

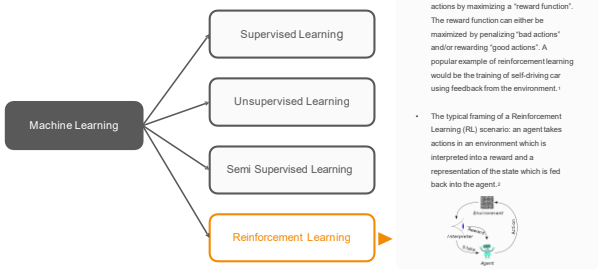
Clustering Patterns in the Data

:: 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[1].

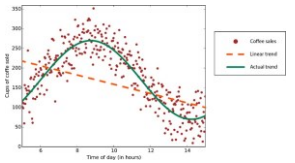
:: The categories of machine learning algorithms



- 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]

:: 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

:: Some of Machine Learning Algorithms

:: Which algorithm should I use ?

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

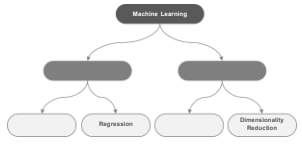
Choosing the right algorithm can seem overwhelming—there are dozens of supervised and unsupervised machine learning algorithms, and each takes a different approach to learning. There is no best method or one size fits all.

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.



:: Train the model

The modelling is an iterative process of building model which is a combination of data structure, algorithm, and mathematics to captures the relationship between features and target within the dataset

6 Model Evaluation

:: 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?

:: 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

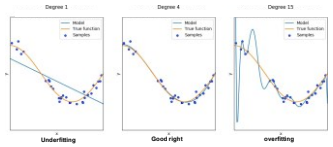
A model's ability to generalize to new data is crucial for the success of a model

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.

:: 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]

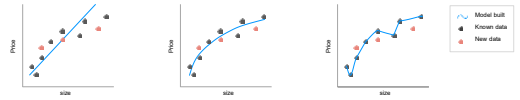


This example demonstrates the problems of underfitting and overfitting and how we can use linear regression with polynomial features to approximate nonlinear functions.

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.[3]

:: About the model fitting

Regression example

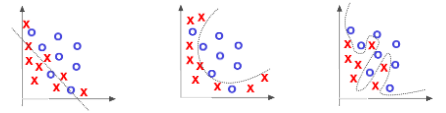


Underfitting model (high training error, high test error)

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)

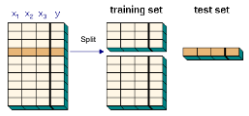
Overfitting Model (no training error, high test error)

:: 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

:: 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

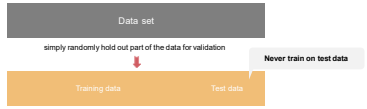
K-fold cross validation

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.

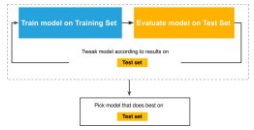
However, its evaluation can have a high variance. The evaluation may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made[2].

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]

:: 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*. [1]

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

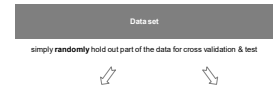
This approach will not be a fair estimate of how well my hypothesis generalizes

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]

:: 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..



For relatively small data set:

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)
http://www.youtube.com/watch?v=uxTBdC0DGs4

For big data set:

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.

:: Model validation strategy

Common validation strategies

Hold-out validation

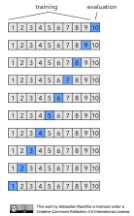
K-fold cross validation

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!

:: 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=_Fe56KmFiag

- 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/validtest 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

:: Evaluate the model's performance

Test Dataset

Evaluate Model

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.

:: Evaluate the model's performance

Test Dataset

Evaluate Model

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)

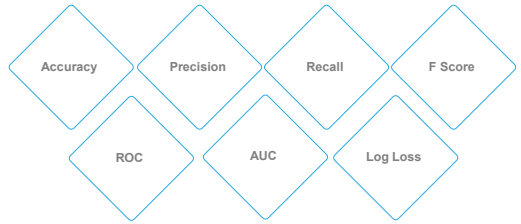
:: Model Evaluation Metrics

Different machine learning tasks have different performance metrics

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

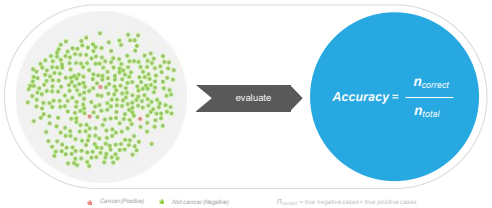
evaluation metrics explain the performance of a model

:: Metrics for evaluating classification model



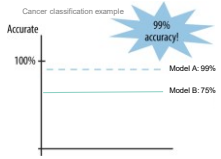
:: What's the accuracy?

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.

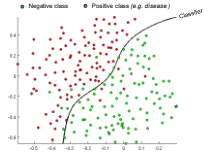
- Model A would achieve 99% accuracy by always predicting "no cancer".
- Model B has 75% accuracy. It correctly predict some cancer classes but also make some misclassification.

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[1].

:: Confusion matrix visually shows more details of the error

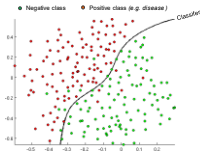


Binary classification example
In this example, the model correctly classifies many classes but also makes some mistakes:

- 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



Display the right and wrong predictions in the table

Confusion matrix

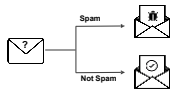
Actual Class \ Predicted Class	Positive (+)	Negative (-)
Positive (+)	TP True Positive	FN False Negative
Negative (-)	FP False Positive	TN True Negative

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)

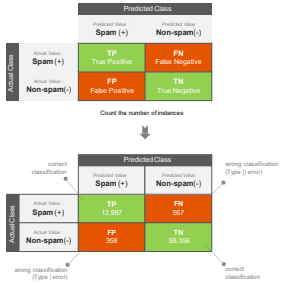
:: Case study: spam detection



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)



:: 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

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

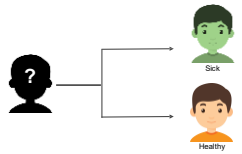
Recall : number of true positives is divided by the total number of elements that actually belong to the positive class

Example

Actual Class \ Predicted Class	Predicted as Spam (+)	Predicted as Non-spam(-)
Actual = Spam (+)	True Positive 2,344	False Negative 5,567
Actual = Non-spam(-)	False Positive 358	True Negative 18,336

- **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

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

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 predicted positive

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

Recall (true positive rate)

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 actual 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.

:: About the cost of prediction error

The model can make mistakes

Confusion Matrix

Actual Class	Predicted Class: Positive (+)	Predicted Class: Negative (-)
Actual Value : Positive (+)	TP True Positive	FN False Negative → Type II error
Actual Value : Negative (-)	FP False Positive → Type I error	TN True Negative

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.

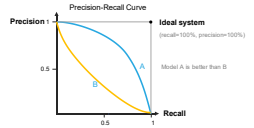
The cost of misclassifying normal email as spam is higher than the cost of failing to detect spam. False positive is much worse than false negative in this case.

:: 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]

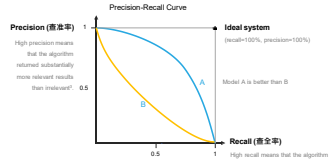
- **Within any one model, you can decide to emphasize either precision or recall.**
  Which one is more important depends on the business needs. In some contexts you mostly care about precision, and in other contexts you really care about recall.
- **You can influence precision and recall by changing 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].

:: 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 $h\theta(x) \geq 0.5$
- 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 $h\theta(x) \geq 0.7$
- 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 $h\theta(x) \geq 0.3$
- 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)

:: 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.

:: 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

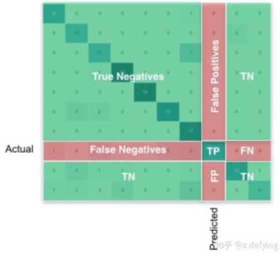
- The F_1 score is a good way to summarize the evaluation of performance in a single number. It combines precision and recall into one measure.
- Bigger F_1 score is better. The ideal F_1 score value is 1. The worst value is 0. In this example, Algorithm1 has highest F_1 score, so it's the best one.

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].

:: Metric: Multi-class Confusion Matrix



- TP: the sum of diagonal value
- FP for specific class: the sum of corresponding column minus the value of corresponding tp
- FN for specific class: the sum of corresponding row minus the value of corresponding tp
- TN for specific class: the sum of the matrix minus the sum of (tp + fp + fn) of the corresponding class

:: Metric: Confusion Matrix exercise （10 min）

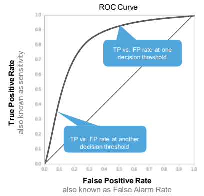
信用卡欺诈行为判断：一共有500条数据，其中正常行为400条，欺诈行为100条
模型A：预测正常行为为270条，整体准确率66%
模型B：预测正常行为400条，整体准确率80%

Q：请问应该采取哪个模型的结果？

:: 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.

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

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

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

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

:: 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].

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].

In this example
- Classifier A is better than B
- Classifier B is better than Random Guessing

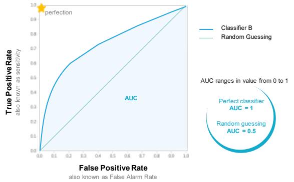
Main advantages of ROC:
- ROC Curves are insensitive to class distribution/unbalanced datasets.
- If the proportion of positive to negative instances changes, the ROC Curve will not change.

:: 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.

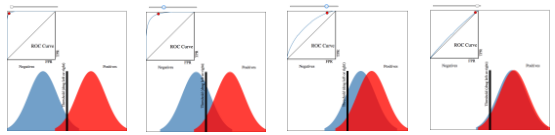
  On the other hand, unlike accuracy, precision, recall and F1 score, AUC does not tell us what performance to expect from the model for a given threshold setting, nor can it be used to determine the optimal value for threshold. In that regard it doesn't take away the need for the other statistical measures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

:: Good demo: ROC curve & AUC

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

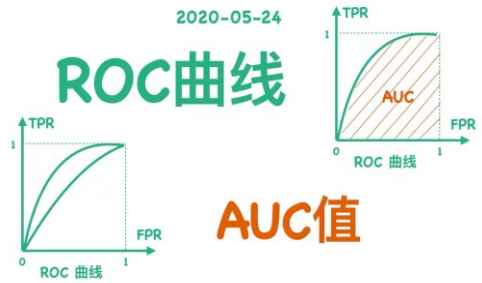


http://www.navan.name/roc

(created by Navan)

:: Good demo: Acc, Precision, Recall

:: Good demo: ROC curve & AUC



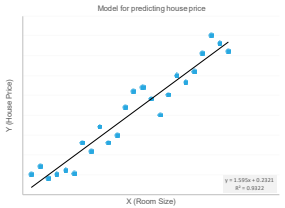
:: Metrics for evaluating regression model

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].

:: Linear regression errors example

Linear Regression example

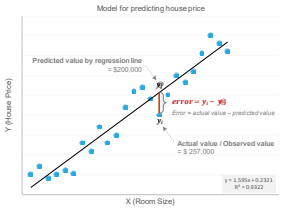
Suppose you want to create a simple model for predicting the house price just based on the room size. A linear regression model can be built for predicting the price.



:: 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

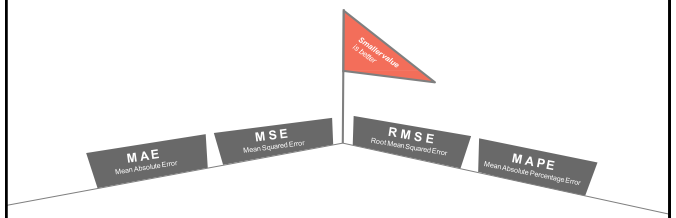
There are difference between the actual value and predicted value from the model. That's the prediction error.

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

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.*

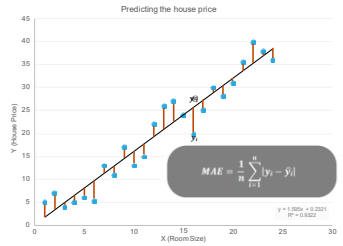
:: Common metrics for measuring the regression errors



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

:: Metric for regression model: MAE

MAE is the mean of the absolute value of the errors



Mean Absolute Error (MAE)

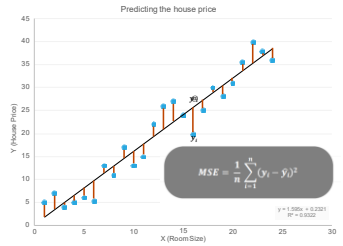
- MAE measures how close the predictions are to the actual outcomes.

  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 !

:: Metric for regression model: MSE

MSE is the mean of the squared value of the errors



Mean Square Error (MSE)

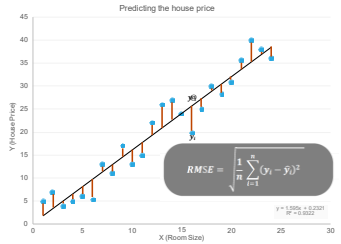
- MSE measures the average sum of squares of the difference between the actual value the predicted values for all data points. Because of the square, the negative values don't cancel positive values and it also amplifies the impact of the errors.

- 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 !

:: 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"

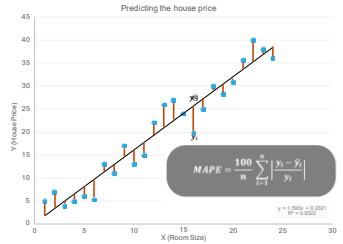
- RMSE may be the most common metric, but it has some problems. RMSE is highly affected by outlier values.

- RMSE more aggressively punishes big errors than small ones. This means the RMSE should be more useful when large errors are particularly undesirable[1].

- A smaller RMSE value is better !

:: 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)

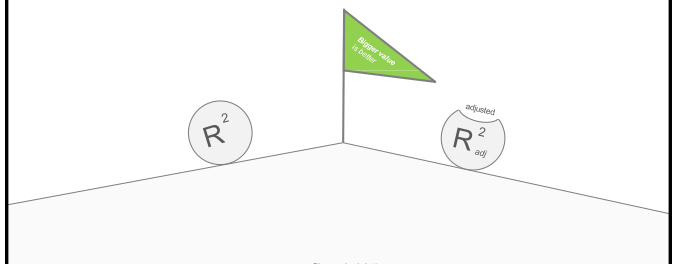
- MAPE functions best when there are no extremes to the data (including zeros).

- 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

:: Other metrics for evaluating the regression model



Bigger value is better

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

r

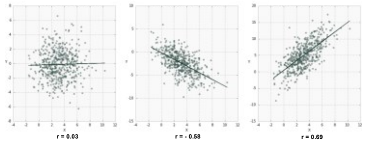
R^2

adjusted R^2_{adj}

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

r is also called as Pearson correlation coefficient

r



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*.

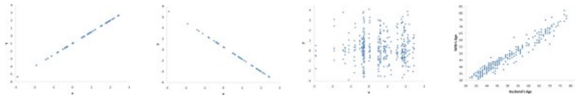
By examining the coefficient values , you can infer something about the strength of the relationship between the two variables, and whether they are positively correlated or negatively correlated.

- 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

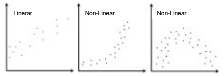
- A positive correlation indicates a relationship between x and y measures such that as values of x increase, values of y also increase.
- A negative correlation indicates the opposite—as values of x increase, values of y decrease

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 ?

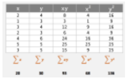
- **Step 1: Determine linearity**
  The first step in studying the relationship between two continuous variables is to draw a scatter plot of the variables to check for linearity. The correlation coefficient should not be calculated if the relationship is not linear[1].



- **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**

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

Example



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

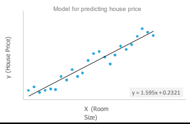
R^2 is a standard way of measuring how well the model fits the data

R^2

- **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 x 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. →



r^2 = 0.93

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

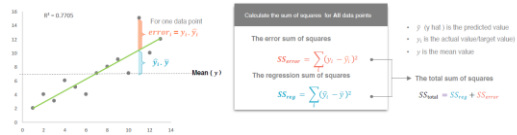
R^2 gives us some information about the **goodness of fit** of a model. In another words, *R^2* measures how well the regression line approximates the real data points.

***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



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 $\hat{y}_i$.

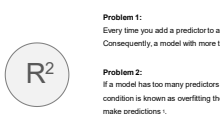
SS_{total} quantifies how much the data points, y_i vary around their mean $\bar{y}$.

R^2 is the regression sum of squares divided by the total sum of squares [1].

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

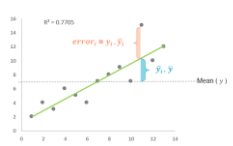
:: R^2 - coefficient of determination

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].



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

:: 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 the total number of explanatory variables in the model (not including the constant term)