Prof Andrew Ng’s Notes : <http://www.holehouse.org/mlclass/>

Difference between statistics, machine learning and data mining : <https://www.r-bloggers.com/whats-the-difference-between-machine-learning-statistics-and-data-mining/>

Classification:

* [Linear classifiers](https://en.wikipedia.org/wiki/Linear_classifier)
  + [Fisher's linear discriminant (Linear Discriminant Analysis)](https://en.wikipedia.org/wiki/Fisher%27s_linear_discriminant)
  + [Logistic regression](https://en.wikipedia.org/wiki/Logistic_regression)
  + [Naive Bayes classifier](https://en.wikipedia.org/wiki/Naive_Bayes_classifier)
  + [Perceptron](https://en.wikipedia.org/wiki/Perceptron)
* [Support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine)
  + [Least squares support vector machines](https://en.wikipedia.org/wiki/Least_squares_support_vector_machine)
* [Quadratic classifiers](https://en.wikipedia.org/wiki/Quadratic_classifier)
* [Kernel estimation](https://en.wikipedia.org/wiki/Variable_kernel_density_estimation#Use_for_statistical_classification)
  + [k-nearest neighbor](https://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm)
* [Boosting (meta-algorithm)](https://en.wikipedia.org/wiki/Boosting_(meta-algorithm))
* [Decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning)
  + [Random forests](https://en.wikipedia.org/wiki/Random_forest)
* [Neural networks](https://en.wikipedia.org/wiki/Artificial_neural_networks)
* [FMM Neural Networks](https://en.wikipedia.org/w/index.php?title=FMM_Neural_Networks&action=edit&redlink=1) [[10]](https://en.wikipedia.org/wiki/Statistical_classification#cite_note-10)
* [Learning vector quantization](https://en.wikipedia.org/wiki/Learning_vector_quantization)

Regression Analysis:

* + [Linear regression](https://en.wikipedia.org/wiki/Linear_regression)
  + [Simple linear regression](https://en.wikipedia.org/wiki/Simple_linear_regression)
  + [Logistic regression](https://en.wikipedia.org/wiki/Logistic_regression)
  + [Nonlinear regression](https://en.wikipedia.org/wiki/Nonlinear_regression)
  + [Nonparametric regression](https://en.wikipedia.org/wiki/Nonparametric_regression)
  + [Robust regression](https://en.wikipedia.org/wiki/Robust_regression)
  + [Stepwise regression](https://en.wikipedia.org/wiki/Stepwise_regression)

Supervised Learning:

Classification:

Regression:

Unsupervised Learning: there’s no label or target value given for the data

Clustering:

Density Estimation:

Dimensionality Reduction:

**How to choose the right algorithm?**

* What is your goal? If you are trying to predict something then supervised learning. If you are trying to predict numerical value then regression otherwise classification if you are trying to predict class/category. If you are not trying to predict anything then unsupervised learning, e.g Do you need to have some numerical estimate of how strong the fit is into each group? If you answer yes, then you probably should look into a density estimation algorithm
* Things to know about your data are these:
  + Are the features nominal or continuous?
  + Are there missing values in the features?
  + If there are missing values, why are there missing values?
  + Are there outliers in the data?
  + Are you looking for a needle in a haystack, something that happens very infrequently?

All of these features about your data can help you narrow the algorithm selection process.

* You’re going to have to try different algorithms and see how they perform. There are other machine learning techniques that you can use to improve the performance of a machine learning algorithm. The relative performance of two algorithms may change after you process the input data. **The point is that finding the best algorithm is an iterative process of trial and error.**

**Steps in developing a machine learning application:**

* **Collect Data:**
* **Prepare the input data:** Prepare the format of data and its features according to the programming language and algorithm you are using.
* **Analyze the input data: (**Exploratory data analysis) this step involves human involvement. If we are working with production system and we know that the data should look like and trust the source, we can skip this step.
* **Train the algorithm:** This is core machine learning step. In the case of unsupervised learning, there’s no training step because you don’t have a target value. Everything is used in the next step.
* **Test the algorithm:** In the case of supervised learning, you have some known values you can use to evaluate the algorithm. In unsupervised learning, you may have to use some other metrics to evaluate the success. In either case, if you’re not satisfied, you can go back to step 4 or step 1, change some things, and try testing again.
* **Use it:**  Here you make a real program to do some task, and once again you see if

all the previous steps worked as you expected. You might encounter some new

data and have to revisit steps 1–5.

**Why Python?**

* Executable pseudo code : The default install of Python already carries high-level data types like lists, tuples, dictionaries, sets, queues, and so on, which you don’t have to program in yourself.
* With Python, you can program in any style you’re familiar with: object-oriented, procedural, functional, and so on.
* Easy Text Processing: With Python it’s easy to process and manipulate text, which makes it ideal for processing non-numeric data. You can get by in Python with little to no regular expression usage. There are a number of libraries for using Python to access web pages, and the intuitive text manipulation makes it easy to extract data from HTML .
* Python is popular: Since python is open source, there are many libraries available which are very useful in machine learning.
* A number of scientific libraries such as SciPy and NumPy allow you to do vector and matrix operations. This makes the code even more readable and allows you to write code that looks like linear algebra. In addition, the scientific libraries SciPy and NumPy are compiled using lower-level languages (C and Fortran); this makes doing computations with these tools much faster.
* Plotting tool called Matplotlib
* Python compared to other languages :
  + Matlab: The problem with MATLAB is that to legally use it will cost you a few thousand dollars. There are third-party add-ons to MATLAB but nothing on the scale of an open source project.
  + Java and C : The problem with these languages is that it takes a lot of code to get simple things done.

\*\***Drawback of python :**  The only real drawback of Python is that it’s not as fast as Java or C. You can, however, call C-compiled programs from Python.

You can, however, call C-compiled programs from Python. This gives you the best of both worlds and allows you to incrementally develop a program. If you experiment with an idea in Python and decide it’s something you want to pursue in a production system, it will be easy to make that transition. If the program is built in a modular fashion, you could first get it up and running in Python and then to improve speed start building portions of the code in C. The Boost C++ library makes this easy to do. Other tools such as **Cython** and **PyPy** allow you write typed versions of Python with performance gains over regular Python.

**Classification :**

**K-nearest neighbour:**

**Pros:** High accuracy, insensitive to outliers, no assumptions about data

**Cons:** Computationally expensive, requires a lot of memory

**Works with:** Numeric values, nominal values

### **KNN for Density Estimation**

Although classification remains the primary application of KNN, we can use it to do density estimation also. Since KNN is non parametric, it can do estimation for arbitrary distributions. The idea is very similar to use of [**Parzen window**](http://en.wikipedia.org/wiki/Parzen_window) . Instead of using hypercube and kernel functions, here we do the estimation as follows – For estimating the density at a point x, place a hypercube centered at x and keep increasing its size till k neighbors are captured. Now estimate the density using the formula,

p(x) = \frac{k/n}{V}

Where n is the total number of V is the volume of the hypercube. Notice that the numerator is essentially a constant and the density is influenced by the volume. The intuition is this : Lets say density at x is very high. Now, we can find k points near x very quickly . These points are also very close to x (by definition of high density). This means the volume of hypercube is small and the resultant density is high. Lets say the density around x is very low. Then the volume of the hypercube needed to encompass k nearest neighbors is large and consequently, the ratio is low.

The volume performs a job similar to the bandwidth parameter in kernel density estimation. In fact , KNN is one of common methods to estimate the bandwidth (eg adaptive mean shift) .

**Decision trees**

**Pros:** Computationally cheap to use, easy for humans to understand learned results,

missing values OK, can deal with irrelevant features

**Cons:** Prone to overfitting

**Works with:** Numeric values, nominal values

**Use:** This can be used in any supervised learning task. Often, trees are used to

better understand the data.

To build a decision tree, you need to make a first decision on the dataset to dictate

which feature is used to split the data. To determine this, you try every feature and measure

which split will give you the best results. After that, you’ll split the dataset into subsets.

The subsets will then traverse down the branches of the first decision node. If the

data on the branches is the same class, then you’ve properly classified it and don’t need

to continue splitting it. If the data isn’t the same, then you need to repeat the splitting

process on this subset. The decision on how to split this subset is done the same way as

the original dataset, and you repeat this process until you’ve classified all the data.

We choose to split our dataset in a way that makes our unorganized data more organized.

There are multiple ways to do this, and each has its own advantages and disadvantages.

One way to organize this messiness is to measure the information. Using information theory, you can measure the information before and after the split. Information

theory is a branch of science that’s concerned with quantifying information.

The change in information before and after the split is known as the information

gain . When you know how to calculate the information gain, you can split your data

across every feature to see which split gives you the highest information gain. The split

with the highest information gain is your best option.

Before you can measure the best split and start splitting our data, you need to

know how to calculate the information gain. The measure of information of a set is

known as the Shannon entropy , or just entropy for short. Its name comes from the father

of information theory, Claude Shannon.

Entropy is defined as the expected value of the information. First, we need to

define information. If you’re classifying something that can take on multiple values,

the information for symbol xi is defined as

where p(xi) is the probability of choosing this class.

To calculate entropy, you need the expected value of all the information of all possible

values of our class. This is given by

where n is the number of classes.

Information gain is used by the [ID3](https://en.wikipedia.org/wiki/ID3_algorithm), [C4.5](https://en.wikipedia.org/wiki/C4.5_algorithm) and C5.0 tree-generation algorithms.

Although information gain is usually a good measure for deciding the [relevance](https://en.wikipedia.org/wiki/Relevance) of an attribute, it is not perfect. A notable problem occurs when information gain is applied to attributes that can take on a large number of distinct values. For example, suppose that one is building a decision tree for some data describing the customers of a business. Information gain is often used to decide which of the attributes are the most relevant, so they can be tested near the root of the tree. One of the input attributes might be the customer's credit card number. This attribute has a high mutual information, because it uniquely identifies each customer, but we do *not* want to include it in the decision tree: deciding how to treat a customer based on their credit card number is unlikely to generalize to customers we haven't seen before ([overfitting](https://en.wikipedia.org/wiki/Overfitting)).

Another common measure of disorder in a set is the **Gini impurity** (used in CART)

Used by the CART (classification and regression tree) algorithm, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

<https://en.wikipedia.org/wiki/Decision_tree_learning>

There are many specific decision-tree algorithms. Notable ones include:

* [ID3](https://en.wikipedia.org/wiki/ID3_algorithm) (Iterative Dichotomiser 3)
* [C4.5](https://en.wikipedia.org/wiki/C4.5_algorithm) (successor of ID3)
* [CART](https://en.wikipedia.org/wiki/Predictive_analytics#Classification_and_regression_trees) (Classification And Regression Tree)
* [CHAID](https://en.wikipedia.org/wiki/CHAID) (CHi-squared Automatic Interaction Detector). Performs multi-level splits when computing classification trees
* [MARS](https://en.wikipedia.org/wiki/Multivariate_adaptive_regression_splines): extends decision trees to handle numerical data better.

**Limitations:**

* The problem of learning an optimal decision tree is known to be [NP-complete](https://en.wikipedia.org/wiki/NP-complete) under several aspects of optimality and even for simple concepts.[[15]](https://en.wikipedia.org/wiki/Decision_tree_learning#cite_note-15)[[16]](https://en.wikipedia.org/wiki/Decision_tree_learning#cite_note-16) Consequently, practical decision-tree learning algorithms are based on heuristics such as the [greedy algorithm](https://en.wikipedia.org/wiki/Greedy_algorithm) where locally-optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally-optimal decision tree. To reduce the greedy effect of local-optimality some methods such as the dual information distance (DID) tree were proposed.[[17]](https://en.wikipedia.org/wiki/Decision_tree_learning#cite_note-17) [[1]](http://www.eng.tau.ac.il/~bengal/DID.pdf)
* Decision-tree learners can create over-complex trees that do not generalise well from the training data. (This is known as [overfitting](https://en.wikipedia.org/wiki/Overfitting).[[18]](https://en.wikipedia.org/wiki/Decision_tree_learning#cite_note-18)) Mechanisms such as [pruning](https://en.wikipedia.org/wiki/Pruning_(decision_trees)) are necessary to avoid this problem (with the exception of some algorithms such as the Conditional Inference approach, that does not require pruning [[12]](https://en.wikipedia.org/wiki/Decision_tree_learning#cite_note-Hothorn2006-12)[[13]](https://en.wikipedia.org/wiki/Decision_tree_learning#cite_note-Strobl2009-13)).
* For data including categorical variables with different numbers of levels, [information gain in decision trees](https://en.wikipedia.org/wiki/Information_gain_in_decision_trees) is biased in favor of those attributes with more levels.[[20]](https://en.wikipedia.org/wiki/Decision_tree_learning#cite_note-20) However, the issue of biased predictor selection is avoided by the Conditional Inference approach.[[12]](https://en.wikipedia.org/wiki/Decision_tree_learning#cite_note-Hothorn2006-12)

**Packt : Machine Learning With Python**

**By : Sebastian Raschka**

***Supervised Learning***

***Unsupervised Learning***

***Reinforcement Learning:*** The goal is to develop a system that improves its performance based on interactions with the environment. Through the interaction with the environment, an agent can then use reinforcement learning to learn a series of actions that maximizes this reward( like win or lose) via an exploratory trial-and-error approach or deliberative planning.

