Data Science and ML Algorithms in scikit-learn

Objectives

In this chapter, participants will learn about some of the algorithms and common analytical methods used in Data Science and Machine Learning (ML), including:

- Terminology
- Dimensionality reduction
- k-Nearest Neighbors
- Decision Trees
- Support Vector Machines (SVMs)
- Naive Bayes Classifier
- Cluster Analysis with k-Means
- Regression Analysis
- Time-Series Analysis

1.1 In-Class Discussion

- What, if any, data science or machine learning initiatives have been undertaken in your organization?
- Could you share some of the successes / failures?
- What are some of the insights you would like to share with the class?

1.2 Types of Machine Learning

- There are three main types of machine learning (ML):
  - unsupervised learning
  - supervised learning, and
  - reinforcement learning
- We will be dealing only with the unsupervised and supervised learning types
- Just FYI: The goal of reinforcement learning is to instruct computer-based algorithms to select actions that maximize a domain-specific gain or minimize a cost (which, essentially, emulates the way humans learn)

1.3 Terminology: Features and Observations

- **A feature** is similar to a relational table's column (entity attribute, or property)
- **An observation** is like a table's row or record
- In Data Science, Machine Learning, and statistics, features are also referred to as variables
- Features are used in making predictions and are called predictors or independent variables
- What you predict may come in a variety of names: response / outcome / predicted variable / dependent variable, etc.
- In Machine Learning (ML) , observations are often referred to as examples
- Vector notation is widely used to represent observations; the vector's elements are features (x_1, x_2, x_3, … x_N below) ; generally, they are vectors themselves (so **X** below is, in fact, a matrix - an array (vector) of arrays):

$$\mathbf{X} \{x_1, x_2, x_3, \ldots x_N\}$$

Notes:

For more terminology used in ML, visit https://developers.google.com/machine-learning/crash-course/glossary

1.4 Terminology: Labels

- A label is a type of object or a value that we assign to an observation or what we are trying to predict
  - You can have labeled and unlabeled observations (examples); the former are mostly used in classification, the latter are found in clustering (unsupervised learning)
  - In classification, labeled examples are used to train the model, then the trained model is fed unlabeled observations (examples) to have the model infer (intelligently guess) the labels of the observations
- Label examples:
  - Software severity levels: Blocker, Critical, Major, Minor, UI cosmetic
  - Trading recommendations: Buy, Sell, Hold
  - E-mail categories: Spam, Non-spam

1.5 Terminology: Continuous and Categorical Features

- Features can be of two types:
  - continuous or
  - categorical
- Categorical features, in turn, are divided in nominal and ordinal

1.6 Continuous Features

- **Continuous** features represent something that can be physically or theoretically measured in numeric values, e.g. blood pressure, size of a tumor, speed, humidity, IQ scoring, etc.
  - Regression models work with continuous features for learning and predicting, e.g.
    - Given our past sales, what is the expected sales figure for the next month?

1.7 Categorical Features

- **Categorical** variables are discrete, enumerated types that can be *ordinal* or *nominal*, like hurricane category, security threat level, city regions, car types, etc.
  - The **nominal** and **ordinal** categories can be illustrated using playing cards:
    - Nominal categories are represented by suits: hearts, diamonds, spades, and clubs (generally, there is no ordering in suites and if one exists, it is game-specific)
    - Ordinal categories are (with some variations) represented by the ranks in each suite (Ace, 2, 3, 4, …., J, Q, K)

1.8 Common Distance Metrics

- A data point is a value at the intersection of a feature (column) and an instance of the observation instance (row)
- Data Science (including ML) uses the concept of a distance between data points as a measure of object similarity

- For continuous numeric variables, the Minkowski distance is used, which has this generic form:

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=0}^{n-1} |x_i - y_i|^p \right)^{1/p}$$

- Which has three special cases:
  - For p=1, the distance is known as the Manhattan distance (a.k.a the L1 norm)
  - For p=2, the distance is known as the Euclidean distance (a.k.a. the L2 norm)
  - When p → +infinity, the distance is known as the Chebyshev distance
- In text classification scenarios, the most commonly used distance metric is *Hamming* distance

1.9 The Euclidean Metric

- The most commonly used distance metric in ML for continuous numeric variables is the *Euclidean* distance
- In mathematics, the Euclidean distance or Euclidean metric is the "ordinary" distance between two points that one would measure with a ruler, and is given by the Pythagorean formula
- In Cartesian coordinates, if we have two points in Euclidean n-space: p and q, the distance from p to q (or from q to p) is given by the Pythagorean formula:

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

1.10 What is a Model

- A model is a formula, or an algorithm, or a prediction function that establishes a relationship between features (predictors) and labels (the output / predicted variable)
- The model is trained to predict (make inference) the labels or values of new observations (examples)
- There are two major life-cycle phases of a model:
  - Model training (fitting)
    - ✔ You train or learn your model on labeled observations (examples) fed to the model
  - Inference (predicting)
    - ✔ Here you use your trained model to calculate / predict the labels of unlabeled observations (examples)

1.11 Supervised vs Unsupervised Machine Learning

- In essence, unsupervised learning (UL) attempts to extract patterns without much human intervention; supervised learning (SL) tries to fit rules and equations
- SL defines a target variable that needs to be predicted / estimated by applying an SL algorithm using predictor (independent) variables (features)
  - Classification and regression are examples of SL algorithms
  - Uses labeled examples
- SL algorithms are built on top of mathematical formulas with predictive capacity
- UL is the opposite of SL
- UL does not have a concept of a target value that needs to be found or

estimated

- Rather, a UL algorithm, for example, can deal with a task of grouping (forming a cluster of) similar items together based on some automatically defined or discovered criteria of data elements' affinity (automatic classification technique)
  - Uses unlabeled examples

Notes:

Some classification systems are referred to as *expert systems* that are created in order to let computers take much of the technical drudgery out of data processing leaving humans with the authority, in most cases, to make the final decision.

1.12 Supervised Machine Learning Algorithms

- Some of the most popular supervised ML algorithms are:
  - Decision Trees
  - Random Forest
  - k-Nearest Neighbors (kNN)
  - Naive Bayes
  - Regression (linear simple, multiple, locally weighted, etc.)
  - Support Vector Machines (SVMs)

1.13 Unsupervised Machine Learning Algorithms

- Some of the most popular unsupervised ML algorithms are:
  - k-Means
  - Hierarchical clustering

- Gaussian mixture models
- Dimensionality reduction falls into the realm of unsupervised learning:
  - PCA, Isomap

1.14 Choose the Right Algorithm

- First, understand your data, identify your needs and the ultimate goal
- The rules below may help you get your direction but those are not written in stone
  - If you are trying to find a probability of an event or predict a value based on existing historical observations, look at the supervised learning (SL) algorithms. Otherwise refer to the unsupervised learning (UL)
  - If you are dealing with discrete (nominal) values like TRUE:FALSE, bad:good:excellent, etc., you need to go with classification algorithms of SL
  - If you are dealing with continuous numerical values, you need to go with regression algorithms of SL
  - If you want to let the machine categorize data into a number of groups, you need to go with clustering algorithms of UL

1.15 The scikit-learn Package

- ML in Python is supported through the scikit-learn package (http://scikit-learn.org), which is described in its documentation as:
- *"Simple and efficient tools for data mining and data analysis*
- *Accessible to everybody, and reusable in various contexts*
- *Built on NumPy, SciPy, and matplotlib*

- *Open source, commercially usable - BSD license*"
- The scikit-learn package supports well-established algorithms for classification, clustering, regression, dimensionality reduction, model selection, and data preprocessing
- The package does not support GPU interfaces and deep / reinforcement learning, which, for the most part, depends on GPU for acceleration

1.16 scikit-learn Estimators, Models, and Predictors

- **Note:** The scikit-learn package uses the terms model, estimator, and predictor in most cases interchangeably
- For the most part, scikit-learn works on NumPy arrays, SciPy sparse matrices, or pandas' DataFrame structure
  - pandas structures are converted to NumPy's ndarrays, where necessary
- Critical to the success of scikit-learn is its uniform API for the supported algorithms, which is based on the ***fit, predict, and transform*** operations

1.17 Model Evaluation

- Once you have your ML model built, you can (and should) evaluate the quality of your model (i.e. how well it can do predictions – its predictive capability)
- The common way is to use your estimator's *score()* method
  - The *score()* method is specific to the estimator you use and you need to refer to the estimator's documentation page
- Another way is to pass the *scoring* named parameter to some of the model's scoring methods
- Your model should have the ability to make accurate predictions (or

generalize) on new data (not seen during training)

Notes:

To get an idea of the diversity of model scoring algorithms, here is a dump of the *cross_val_score()* method from the *sklearn.model_selection* module:

```
['accuracy', 'adjusted_mutual_info_score', 'adjusted_rand_score',
'average_precision', 'completeness_score', 'explained_variance', 'f1', 'f1_macro',
'f1_micro', 'f1_samples', 'f1_weighted', 'fowlkes_mallows_score',
'homogeneity_score', 'mutual_info_score', 'neg_log_loss',
'neg_mean_absolute_error', 'neg_mean_squared_error', 'neg_mean_squared_log_error',
'neg_median_absolute_error', 'normalized_mutual_info_score', 'precision',
'precision_macro', 'precision_micro', 'precision_samples', 'precision_weighted',
'r2', 'recall', 'recall_macro', 'recall_micro', 'recall_samples','recall_weighted',
'roc_auc', 'v_measure_score
```

The R2 Score

One of the most popular scoring metric in statistics which is widely used in evaluating models is the coefficient of determination, depicted as R^2. It shows the proportion of the variance in the dependent variable that is predictable from the independent variable(s). R^2 is a normalized value between 0 and 1:

- 0 (or a value close to zero) indicates that there is no linear relationship (no correlation)
- 1 (or, more practically, a value close to one) indicates that your model is a good fit and can explain most of the data

For more information, visit https://en.wikipedia.org/wiki/Coefficient_of_determination

1.18 The Error Rate

- A common measure of a model's accuracy is the error rate which is the number of wrong predictions (e.g. classification of test observations) divided by the total number of tests
- In the ideal world (when you have a perfect training set and your test objects have strong affinity with some classes), your model makes predictions with no errors (the error rate is 0)
- On the other side of the spectrum, an error rate of 1.0 indicates a major

problem with the training set and/or ambiguous test objects

- Error tolerance levels depend on the type of the model

1.19 Feature Engineering

- The process of creating/transforming predictors from the raw data is called feature engineering or feature extraction
- Common operations here include: scaling, creating additional (synthetic) features based on the original data, dropping features that might correlate with the ones you have selected, etc.
  - ✔ Synthetic features are usually some ratios of two of more raw features

1.20 Scaling of the Features

- If some variables have significantly larger values than others, you may have skewed distances where smaller variables have no influence on the overall distance
- To avoid this situation, apply data normalization / scaling techniques (e.g. min-max or z-score transformations)

1.21 Feature Blending (Creating Synthetic Features)

- If variables X_1 and X_2 share variance (they are correlated), you may try to introduce a new feature that blends both variables using some form of relationship
  - ◇ There may be more than two variables involved
- You need to come up with the importance of each variable and assign their weights accordingly
- For example, you can create a new variable X_3 like so:

- $X_3 = w_2 * X_2 + w_1 * X_1$
- where $w_1 + w_2 = 1.0$ (you keep the weights scaled !)
  - **Note:** X_1 and X_2 should be normalized before being blended
- You may want to decide the $w_1 + w_2$ weights using the Singular Value Decomposition (SVD) technique [http://bit.ly/2rEu6QE]

Notes:

Correlation coefficient is often defined as "shared variance divided by combined [or hybrid] variance"

1.22 The one-hot Encoding Scheme

- String values (e.g. 'buy', 'sell', 'hold') that cannot be (easily) represented by numeric values can be encoded using the **one-hot** encoding scheme where only one element of the resulting feature vector can have a value of one (1), while other elements are set to zero (0)
- In the one-hot transformation, you, essentially, end up with as many new features as there are levels in that variable;
  - For example, a feature with these three levels: 'buy', 'sell', 'hold' will be transformed into three new features (that we may want to call *buy, sell,* and *hold*) holding this data; those features will be now represented as a sparse matrix:

```
buy sell  hold
1      0      0    #the mapping of 'buy'
0      0      1    #the mapping of 'hold'
1      0      0    #the mapping of 'buy' again
0      1      0    #the mapping of 'sell'
```

1.23 Bias-Variance (Underfitting vs Overfitting) Trade-off

- **Underfitting** is a property of your model which makes your model less accurate by virtue of being too generic, or **biased**
  - Such a model appears to be rather simple failing to account for some important regularities in the training data and that has low variance in predictions
- **Overfitting** is the opposite of underfitting – it makes your model too sensitive to information noise / variance in your training data
  - Usually, this property is exhibited in more complex data models which are trying to describe your training data as close as possible
- A good model strikes a good balance between bias and its overreaction to variance (a **bias-variance** balance or trade-off)
- The bias-variance trade off applies to classification and regression models (supervised learning)

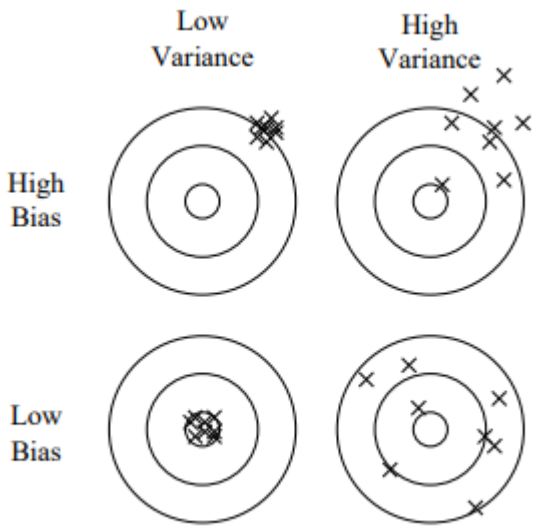
1.24 The Modeling Error Factors

- Generally, prediction errors of your model can be decomposed into three terms:

```
Error = Bias + Variance + Data_Noise
```

- To minimize errors, you will need to find the best compromise between *Bias* and *Variance*

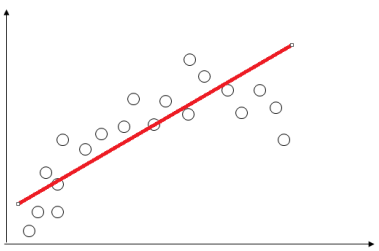
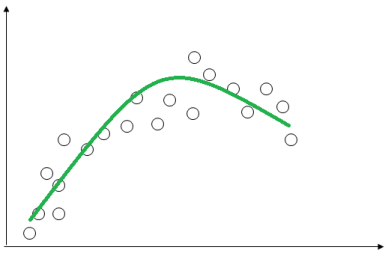
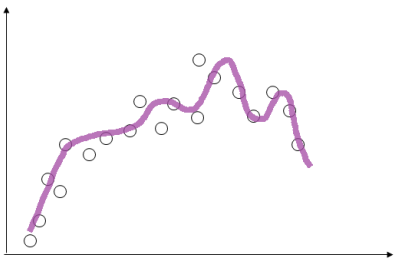
1.25 One Way to Visualize Bias and Variance



Bias and variance in dart-throwing.

Source: http://bit.ly/1c3QB48

1.26 Underfitting vs Overfitting Visualization

		
Underfitting (high bias)	Best fitting (the true relationship)	Overfitting (high variance)

1.27 Balancing Off the Bias-Variance Ratio

- The common techniques to balance off the bias-variance ratio is **dimensionality reduction** and **feature selection**
  - These techniques are commonly referred to as regularization
- These techniques can decrease variance by simplifying models (increasing bias)
  - Another way to decrease variance is getting larger training sets
- Machine Learning algorithms typically offer some hyperparameters to control bias and variance

1.28 Regularization in scikit-learn

- In ML, regularization is a process of reducing overfitting by reducing the variance of the estimates
  - Note that high variance may be caused by multicollinearity (correlation) between predictors
- Regularization achieves its goal by introducing penalty to the cost function that “shrinks” estimate coefficients (it is a controlled way to add some bias)
- In some scikit-learn algorithms (e.g. *LogisticRegression or LinearSVC*), the regularization strength (how aggressive it is) is controlled by a parameter named C which is an inverse of regularization strength meaning that smaller values of C specify stronger regularization
  - The C parameter carries a positive value with a default of 1.0.
- To make things more interesting, other algorithms (e.g. *Ridge*) introduce a parameter which corresponds to C^-1 (which is 1/C) and call it alpha; larger alpha values signify stronger (more aggressive) regularization

Notes:

Mathematically, the penalty is a summation of the predictors' coefficients; if the coefficients are squared, it is a case of L2 regularization (Ridge or Tikhonov's regularization); in case where an absolute value of the coefficients is taken, it is the Lasso regularization).

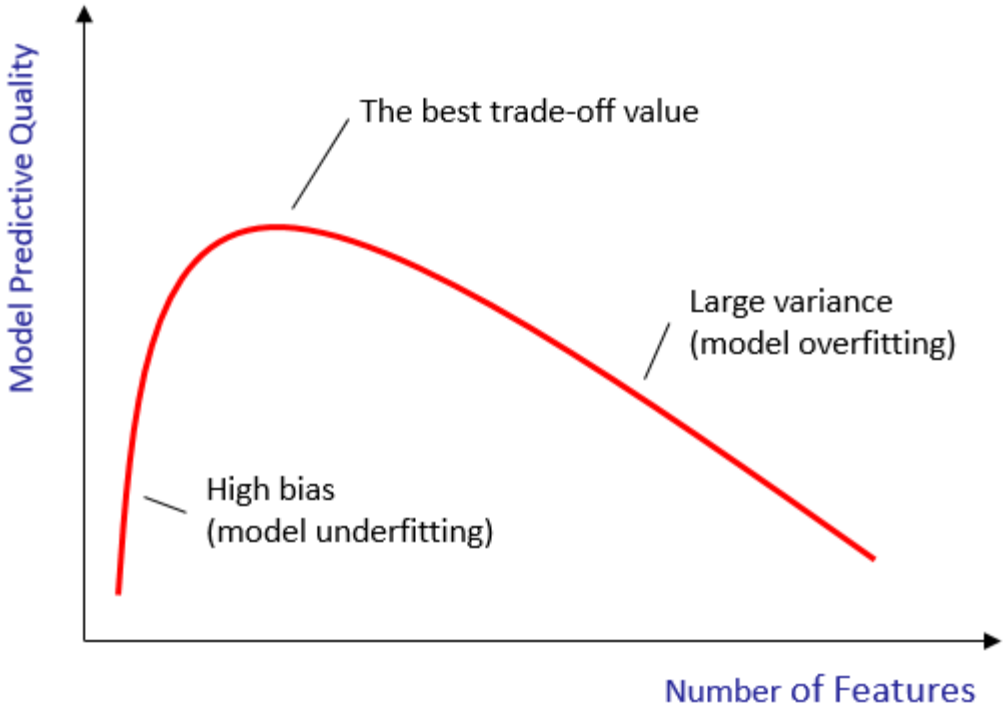
scikit-learn's Ridge regression improves on the ordinary linear regression models by introducing a penalty on the size of the regression coefficients as its *alpha* parameter e.g:

```
from sklearn import linear_model
regModel = linear_model.Ridge (alpha = .01)
regModel.fit(X, y)
…
```

To lean more about regularization support in scikit-learn, visit http://scikit-learn.org/stable/modules/linear_model.html

1.29 Regularization, Take Two

- In some situations, regularization means a processes of finding an optimum number of features that maximize the quality of your model by balancing off bias against variance



1.30 Dimensionality Reduction

- In ML and statistics, dimensionality reduction (or dimension reduction) is the process of transforming the original feature set into another one with fewer features
- Features may be dropped or combined using some inter-feature relationships
- Essentially, the process is trying to deal with “The curse of dimensionality” (see the slide*'s Notes*)
- Examples:
  - Compress a video stream by reducing the number of colors and/or pixels
  - Creating a digest (executive summary) of some textual material

Notes:

The Curse of Dimensionality

Generally, this expression attests to the fact that with high-dimensional data sets processing becomes exponentially more difficult; algorithms that work fine in low-dimensional spaces become intractable.

In other cases, algorithms that are implemented as matrix operations fail to work in situations where the number of features exceeds the number of observations.

Also, similarity-based reasoning (classification) would require massive amount of training data in high-dimensional problem domains not suitable for processing using traditional computing systems (you would probably require specialized cluster-based solutions like Apache Spark).

1.31 PCA and isomap

- Principal Component Analysis (**PCA**) is a fast dimensionality reduction algorithm that transforms (maps) data in the high-dimensional space into a space with fewer dimensions (features) called *principal components*
- The principal components (new features) are mathematically bound to the old features using some linear transformation formulas
- Essentially, PCA transforms possibly correlated features into a set of linearly uncorrelated variables (principal components are orthogonal, that is uncorrelated with each other)
- PCA is very fast but only supports linear transformations (it assumes that a linear relationship exists between features)
- **Kernel** (trick) **PCA** is a variation of PCA which allows PCA to be applied to non-linear problems
- **isomap** is a non-linear dimensionality reduction algorithm

Notes:

The principal components generated by PCA are ranked by the size of the variance of input data they explain with the one which accounts for the largest variance being referred to as the first PC or PC1.

Other PCs are ordered by the decreasing amount of variance explained. The higher the variance a PC accounts for, the more informative that PC is. Components with smaller variances can be dropped altogether without losing much of the informative value of your PCA model. This is a critical feature of PCA that can help you reduce the number of features (variables) used in your data model, effectively compressing your datasets, for which PCA is, sometimes, referred to as dimensionality reduction algorithm. PCA may also be effective in predictive modeling and outlier detection.

According to Wikipedia [https://en.wikipedia.org/wiki/Principal_component_analysis], "*PCA can be thought of as fitting an n-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipsoid is small, then the variance along that axis is also small, and by omitting that axis and its corresponding principal component from our representation of the dataset, we lose only a commensurately small amount of information.*"

1.32 The Advantages of Dimensionality Reduction

- Dimensionality reduction operations usually reduce the storage space, conserves networking bandwidth, and improves I/O times when using your model
  - Solutions running on cluster-based platforms like Hadoop can help even further
- Dimensionality reduction helps identify and handle multi-collinearity problems as some features in your training data sets may be strongly correlated and dropping some of them goes a long way to improving the performance of the ML models
- It may become easier to perform EDA when your data sets are reduced to two or three features

1.33 The LIBSVM format

- *LIBSVM* is a compact text format for encoding data
- A file in *LIBSVM* format is shaped as a matrix in which each line is a space-delimited record that represents a labeled sparse feature (attribute / property) vector

- In this format, each line takes the form <label> <feature-id>:<feature-value> <feature-id>:<feature-value> ....
- This format is especially suitable for sparse datasets
- For loading files in the *LIBSVM* format, scikit-learn offers the *load_svmlight_file()* method in the *sklearn.datasets* module
- Example:

```
featuresDS, labels =
    load_svmlight_file("/path/to/train_dataset")
```

Notes:

LIBSVM is a library (and the data format) for support vector machines [*http://www.csie.ntu.edu.tw/~cjlin/libsvm/faq.html*].

This resource [*http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/*] contains a great number of classification, regression, multi-label and string data sets stored in LIBSVM format.

1.34 Life-cycles of Machine Learning Development

- Collect data
- Understand the data
  - That will help to select the appropriate algorithm
- Prime (clean up) data
  - Remove obvious outliers (be careful with this step!); transform data into a usable machine processing format
- In SL, train the algorithm
  - in UL, there is no training phase
- Perform your task by applying the selected algorithm

Notes:

In statistics, an **outlier** is a term that describes an observation that is, according to some the established metric, way too distant from the rest of the data in the sample. In many cases, outliers are the result of measurement errors.

1.35 Data Split for Training and Test Data Sets

- You cannot use the same data used for training your model when you are testing it
- Sometimes (e.g. at the initial stages of your ML project, or such like), you may be constrained by data availability
- If you need to use the same data set for training and testing your ML models, you need to split the data so that you have one data set for training your model, and one for testing
- There are many variations of the splitting techniques, but the most common (and simple) one is to allocate about 70% of the initial data for training and 30% for testing

Notes:

Holdout data

Observations that are set aside for testing and not used during training are called "holdout" data. Holdout data is used to evaluate your model's predictive capability and its ability to generalize to data other than the data the model saw during the trained step.

1.36 Data Splitting in scikit-learn

- There are a number of considerations when performing a data split operation, including:
  - Observations (records) must be selected randomly to prevent record sequence bias

- Every labeled record (for classification models) should have a fair share in both training and testing data sets

- *scikit-learn* simplifies this operation through its specialized *model_selection* module which offers a number of classes and functions to support a variety of data splitting options, e.g.
  - The *train_test_split* function
  - The *ShuffleSplit* class

1.37 Hands-on Exercise

- The Data Splitting Lab

1.38 Classification (Supervised ML) Examples

- Identification of prospective borrowers who are likely to default on their loans (based on historical observations)
- Spam detection
- Image recognition (a smiling face, a type of a musical instrument)

1.39 Classifying with k-Nearest Neighbors

- k-Nearest Neighbors (*kNN*) algorithm is a method for classifying objects
- It is used for answering such questions as "Which class does the test data belong to?"
- The test objects are classified based on the proximity of their features to one or more (*k*) objects from the training set
  - The test object is assigned to a class that has features most common amongst its *k* nearest neighbors in the feature space

- *kNN* has the following characteristics
  - Computationally stable and insensitive to outliers
  - In some rich-feature data (data with many attributes) becomes CPU and memory bound (requires powerful machines with lots of memory)
  - A high value of k leads to high bias and low variance

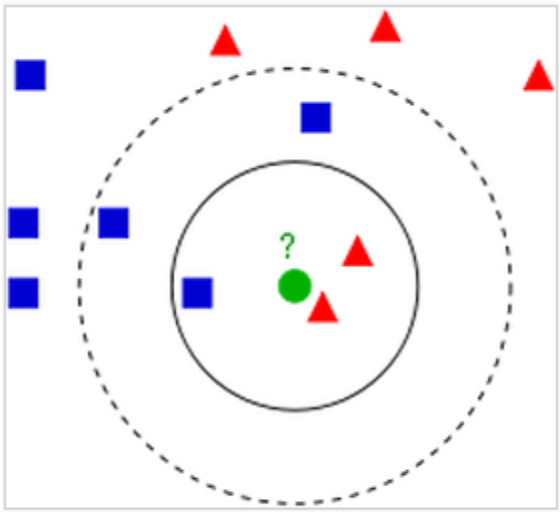
Notes:

The k parameter in the *kNN* algorithm is a small positive integer. With k = 1, the object is assigned to the class of objects based on its proximity to a single nearest neighbor.

1.40 k-Nearest Neighbors Algorithm

- The training dataset represents a matrix with rows as instances of a class and columns representing the attributes (features)
  - Training datasets must be loaded in computer memory (RAM) for the algorithm to work
  - Big training datasets require a large amount of RAM and may lead to active disk swapping situations that may significantly slow down data processing
- Each row in the training dataset must be tagged with a class label that will be used to classify the test object
- In the process of classification, the test object is classified by assigning the class label which is most frequent among the k training rows nearest to the test object in the feature space
  - The algorithm uses the Euclidean distance metric which is CPU-bound
- For high-dimensional datasets (number of features > 10), dimensionality reduction is usually performed prior to applying kNN

1.41 k-Nearest Neighbors Algorithm



Example of *k*-NN classification. The test sample (circle) should be classified either to the first class of blue squares or to the second class of red triangles. If *k* = 3 (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If *k* = 5 (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

Adapted from:
http://upload.wikimedia.org/wikipedia/commons/thumb/e/e7/KnnClassification.svg/220px-KnnClassification.svg.png

1.42 Hands-on Exercise

- The k-Nearest Neighbors Algorithm Lab

1.43 Regression Analysis

- The term regression has many specialized meanings and here we are referring to it as a set of modeling and analyzing techniques that help determine the relationship between a dependent variable and one or more independent variables
- Regression analysis is primarily used for the purpose of predicting and forecasting a dependent (response) variable based on the values of at least one independent (explanatory/predictor) variable
- Most often, the linear regression analysis leverages the least squares method to find a best-fitting straight line for sample data
- Predicting the response variable from a single or a set of predictor variables is also called “regressing” the response variable on the predictor variable(s)

Notes:

To "regress" is to go back, or revert to an earlier or more primitive state. The statistical term "regression" seems to have been first used by Francis Galton, Charles Darwin's cousin. … Galton noticed that the children of tall parents tended to themselves be tall, but not as tall as their parents. Galton called this "regression to mediocrity", but nowadays it is usually referred to as "regression to (or towards) the mean

Source: http://www.fallacyfiles.org/regressf.html

Major Underlying Assumptions for Regression Analysis

- The data sample is representative of the population (has enough data elements in it)
- The predictor variables are independent (if more than one is used)
- The error variance is consistent across observations (you expect same magnitude of an error in each observation instance)

1.44 Regression vs Correlation

- Correlation and regression are very similar
- The difference is that regression tries to offer a model which, when fitted, can predict one variable (usually named Y) from another (or others), usually named X
- Correlation simply describes the association of the variables without discriminating them over the predictor / predicted dichotomy
- Just as with correlation, a predictive relationship in regression does not mean that X does cause Y
  - Sometimes, Y is viewed as the hypothesized consequence of X, which is viewed as a hypothesized cause
    - For that you need further investigation
  - Regression analysis only requires that one variable is specified as a predictor of another

1.45 Regression vs Classification

- Both regression and classification aim at predicting a target value
- The difference between regression and classification is in that
  - the variable being predicted / forecasted in **regression** is continuous
  - the variable being predicted / forecasted in **classification** is discrete (a class)

1.46 Simple Linear Regression Model

- Uses a single numerical independent, predictor variable (X) to predict the numerical dependent, response variable (Y)

- **Note**: Multiple regression models use several explanatory variables (X_1, X_2, ...X_n) to predict a numerical dependent variable Y

- The linear regression model is expressed by a regression equation:

  ```
  Y = A * X + B
  ```

  - where
    - A - The slope for the population, a.k.a. the regression coefficient
    - B -The Y-intercept for the population (the predicted value of Y when the predictor variable is zero); also referred to as *bias*
  - Slope (positive or negative) represents the expected change in Y per unit change in X ($\Delta Y/\Delta X$)

- **Note**
  - In ML documentation, you can meet the following notation for simple (one variable) regression model:

    $$y = w_1 x_1 + w_0$$

    - Where w_1 is the weight of x_1 and w_0 is the bias (intercept)

Notes:

To find the slope of a straight line, take two points on the line, (x1,y1) and (x2,y2); the slope is equal to (y2 - y1)/(x2 - x1).

The Y-intercept of a line is the value of Y at the point where the line crosses the Y-axis.

1.47 Linear Regression Illustration

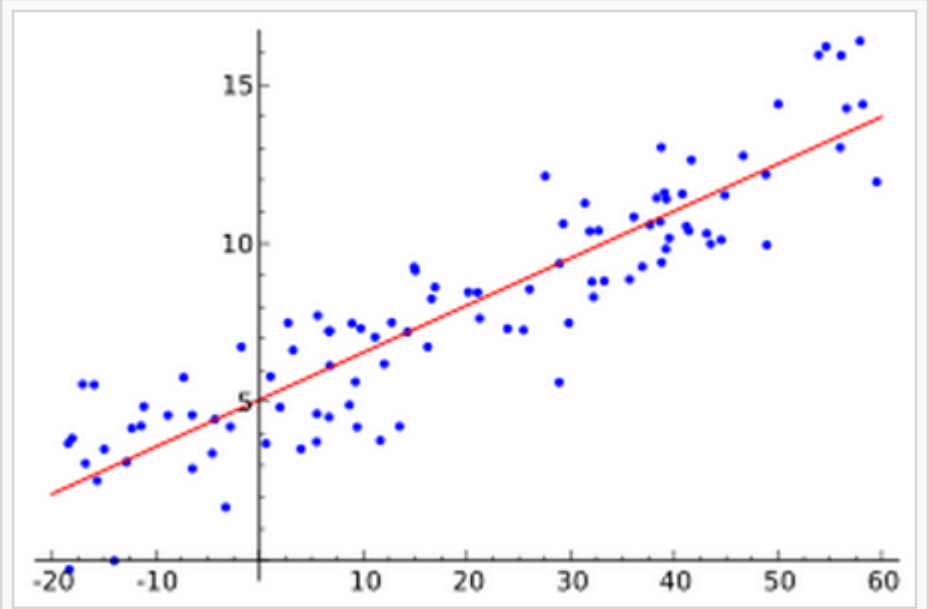


Illustration of linear regression on a data set.

Source: http://wikipedia.org

Notes:

Linear vs Non-Linear Regression

Linear regression does not mean that it fits data with a straight line.

For example, $\mathbf{Y_i = A_1 * sin(X_i) + A_0}$ is still treated as linear

It is about linear use of parameters (A1 and A0 in our case) that will be determined by regression algorithm are added up in the regression equation to produce the response variable.

Nonlinear regression models break the linearity of using parameters by using nonlinear models.

For example, $\mathbf{Y_i = e^{AiXi} + A_0}$ (notice that the A1 parameter is used non-linearly with respect of A0).

Nonlinear regression uses iterative approach to calculate the parameters with initial estimated values for each parameter.

1.48 Least-Squares Method (LSM)

- Simple linear regression analysis aims at finding the straight line that best fits the linear relationship between X and Y
- LSM is a mathematical technique to determine the values of A_1 and A_2 which minimize the sum of the squared differences between actual (observed) values of the dependent variable and the modeled value of it)

$$\sum_{i=1}^{n} (Y_i - F(X_i))^2 \rightarrow \min$$

  - where

    Y_i is the actual value of Y for observation of X_i

    $F(X_i)$ is the value of Y for observation of X_i as determined by the linear model

1.49 Gradient Descend Optimization

- The LSM method depends on the data matrix inversion computation which is not always possible based on matrix calculus
- The performance of LSM may also be constrained by the size of data
- ML algorithms prefer to use the gradient descend (GD) optimization techniques (and some boosting derivatives of it) to address all the LSM deficiencies
- One requirement for GD is to apply data scaling (e.g. z-transform) or data normalization (setting feature values to a specific range, e.g. [-1, +1])
- The scikit-learn module uses GD

1.50 Locally Weighted Linear Regression

- One problem with linear regression is that in some cases it underfits the

sample data which produces not accurate predictions

- Locally weighted linear regression (LWLR) technique is provide a better fitting model by introducing (adding) some bias in the estimators
- In LWLR, data points near the point of higher significance are given more weight (bias) before applying the usual least-squares regression

Notes:

An **estimator** is a statistic calculated from a sample that provides an estimate of a true value that is being observed, e.g. a mean, a standard deviation, etc.

1.51 Regression Models in Excel

- Microsoft Excel offers the *LINEST* built-in function to construct a linear regression model using the "least squares" method that best fits input data with a straight line
- When you have only one independent variable X, you can obtain the slope and the Y intercept values directly by using the following formulas:
  - Slope:

```
=INDEX(LINEST(known_y's,known_x's),1)
```

  - Y-intercept:

```
=INDEX(LINEST(known_y's,known_x's),2)
```

- **Note:** LINEST also support multiple variables

1.52 Multiple Regression Analysis

- Simple regression analysis is concerned with a single independent (explanatory/predictor) variable X that was used to predict the value of a dependent (response) variable Y
- The regression model was expressed as follows:

$$Y = A*X + B$$

- Where
  - Y is the the vector of the response values
  - A is a matrix of features
  - B is the bias
- In many practical cases, a better-fitting model can be built if more than one predictor variables are considered
- Multiple regression models are built with several (more than one) predictor variables which is represented in a multidimensional space as a hyperplane
- The multiple regression models are expressed as follows:

$$Y_i = A_1X_{1i} + A_2X_{2i} + A_3X_{3i} + \ldots + A_nX_{ni} + B$$

- where A_1, A_2 A_3, etc. are regression coefficients
  - ✔ A_1 - Slope for the population with variable X_1 holding other variables (X_2, X_3, …) constant
  - ✔ A_2 - Slope for the population with variable X_2 holding other variables (X_1, X_3, …) constant
  - ✔ …
  - ✔ B - The Y-intercept (a.k.a. bias) for the population (the predicted value of Y when the predictor variables are all zero)

Notes:

In many books on statistics, you can see the following notation for the linear regression formular:

$$y = \beta X + \alpha$$

1.53 Linear Logistic (Logit) Regression

- Like all regression algorithms, Logistic Regression (LR) is used in predictive

analytics; LR, however, despite its name, is used for classification rather than regression

- LR predicts binary outcome of the dependent (target) variable; in other words, the dependent variable is categorized into two classes: class **one** (which is mapped to True / 1) and class **zero** (mapped to False / 0)**,** e.g. success / failure; up / down; healthy / sick
- LR provides a probability estimate as output
  - It can, for example, be used in assessing eligibility of a client for a loan (a yes / no decision by the loan officer)
- As with other regression models, you have a set of explanatory variables (features) which can be discrete and/or continuous
- Unlike ordinary regression that calculates coefficients / weights that minimize the sum of squared errors, LR computes coefficients such that they maximize the likelihood of observing the sample values
- It is fast and scales well to train on massive data; it also great for low latency predictions

Notes:

For multi-class (>2) classification tasks, Logistic Regression (LR) can use either of the following two strategies:

One versus Rest (OvR) where the LR compares every class with other classes. The class with the highest probability is chosen as class one.

So, if you have three classes A, B, and C, LG will build these three models: A-(BC), B-(AC), and C-(AB). The one with the highest probability will be voted as class one.

One versus One (OvO) where the LR compares every class against each of the other classes. The class with the highest probability is chosen as class one.

So, if you have four classes A, B, C, and D, LG will build these six models (A-B and B-A model pairs are treated as equivalent):

A-B

A-C

A-D

B-C

B-D

C-D

1.54 Interpreting Linear Logistic Regression Results

- LR builds a linear model for a transformed target variable into probabilities
- For example, if your logit model distinguishes between spam (the **one** class) and non-spam (**zero**) and your logit model infers a value of 0.81 on a particular (test) email message to be spam (**one**), it means that, given your training set, the probability of that email to be spam is 81% and, conversely, the chances that that email to be non-spam is (1.0 - 0.81)*100 = 19%

1.55 Decision Trees

- Decision trees are decision support algorithms of supervised learning that use a tree-like graph to model decision points
- Outcome of each decision point is a probability-based event associated with some consequences (cost, time, etc.)
- Decision trees are widely used in decision analysis to help identify the strategy that most likely lead to reaching the established goal
  - For example, you may use it to build a model for segmenting respondents based on their answers to a survey
- In the most basic form, decision trees are represented by binary trees
- The depth of the tree is driven by training data variance causing overfitting
  - To prevent overfitting, trees are commonly pruned

- Decision tree are regarded as the most commonly used classification algorithms
- Computationally efficient for both training and testing phases
- Human-friendly for interpreting results
- The most popular decision tree algorithms are C4.5 and CART

Notes:

Trees vs kNN

The *kNN* algorithm does not concern itself with the underlying data structures which, among other things, may be a limiting factor in some decision-making cases.

Similar to *kNN*, decision tree algorithms are trying to accurately predict the class of the test data (to correctly classify the data).

Properties of Trees:

Can handle a combination of quantitative and qualitative predictors.
Easily ignore statistically insignificant (redundant) variables.
Robust against missing data.
Small trees are easy to interpret; while large trees are hard to interpret.

Prediction performance using large trees is often poor.

1.56 Decision Tree Terminology

- A Decision Tree has a root and branches with leafs
- Traversing from the root to a leaf is controlled by classification rules
- A branch represents an outcome of a test
- A leaf node represents a class label (category of objects)
- Decision trees use 3 types of nodes:
  - Decision nodes – normally shown as squares on diagrams

- Chance nodes – normally shown as circles on diagrams
- End nodes – normally shown as triangles on diagrams

1.57 Decision Tree Classification in Context of Information Theory

- Splitting sample datasets into branches pursues the goal of getting data more organized
- Getting organized is a subjective notion
- The term *Information gain* describes a quantifiable measure of making information more organized along the way of getting closer to the final conclusion / decision
- In Information Theory, information contained in a data set is measured in the context of the Shannon entropy concept (http://en.wikipedia.org/wiki/Entropy_(information_theory))

1.58 Information Entropy Defined

- In Information Theory, entropy is defined as a measure of uncertainty
  - Independent flips of a geometrically symmetric coin have an entropy of 1 bit per flip (head or tail)
  - A string of symbols, e.g. AAA..A consisting of the same symbol ('A') has an entropy rate of 0 as there is no uncertainty associated with the text – the next symbol is always (with a 100% guarantee) the same symbol then the previous one
  - The entropy of sequence AZAZAZ ..AZ is 1 bit per (next) character provided each letter is treated as independent

1.59 The Shannon Entropy Formula

- For a random variable X with n outcomes {x_1, x_2..., x_n}, the Shannon entropy, denoted by **H(X)**, is defined as

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$$

  - where

    $p(x_i)$ is the probability mass function of outcome x_i

    Note: In practice, 2 is used as the base of the logarithm (shown as ***b*** in the above formula)

- In practical terms, the uncertainty of an outcome (entropy) is the number of bits needed to specify the outcome
  - Two outcomes of a coin flip (head or tail) can be coded with one bit (0/1) as also supported by the $\log_2 2 = 1$ calculation
  - The toss of a die has 6 possible equal outcomes, which would require $\log_2 6$ or 2.58... bits, which is rounded up to 3
  - The higher the entropy, the more bits is required to represent the number of possible outcomes and more work is required to get information organized

1.60 The Simplified Decision Tree Algorithm

- The core problem classification algorithms have to tackle is making an informed decision as to which feature (attribute) is the best to split on
  - At each level where a split occurs, calculate the degree of information organization (measure the entropy) before and after split for all possible outcomes (features to split on)

- The split that yields the highest information gain (the larges reduction in entropy) is the best choice
- Repeat until classification is complete

- The above technique is called the recursive partitioning and it is widely used in data mining
- Where values are continuous rather than nominal, those need to be converted to some discrete ranges

1.61 Using Decision Trees

- Decision trees are supervised learning algorithms and they need training
- Training (machine learning) phase is conducted in the above process of growing (constructing) the tree
  - This is potentially a time consuming phase
- When the tree is built, it is best practice to save the generated classifier so that it can be re-used on subsequent test data without re-calculating information gains of splits, etc.
- Now you can start using it for classifying the test data
  - This is usually a fast phase
  - This step also helps you understand the data

1.62 Random Forests

- Random Forests (RF) algorithm is an aggregator of multiple decision trees in an ensemble
  - Multiple decision trees are built against the same training data set in parallel and they are later used to vote on decisions through majority rule

- Usually, RF uses a single decision tree implementation to build the forest (ensemble)

- RF can be used for both classification and regression
- The majority rule voting on a decision helps reduce the risk of overfitting
  - **Note**: Overfitting is too much local focus - not seeing a "bigger picture"
  - RF combines many "weak" (high bias) models in an ensemble that has lower bias than the individual models
- RF handles multi-class classification of both continuous features (data attributes) and categorical features (like enumerations, or discrete values)
- As an added benefit, RF does not require feature scaling (data normalization), and it is able to capture feature cross-dependencies

Notes:

Example of a Random Forests model dump for predicting the class label (0 or 1):

```
TreeEnsembleModel classifier with 3 trees:

  Tree 0:
    If (feature 0 <= 10.0)
     Predict: 0.0
    Else (feature 0 > 10.0)
     Predict: 1.0
  Tree 1:
    If (feature 0 <= 13.0)
     Predict: 0.0
    Else (feature 0 > 13.0)
     Predict: 1.0
  Tree 2:
    If (feature 0 <= 10.0)
     Predict: 0.0
    Else (feature 0 > 10.0)
     Predict: 1.0
```

1.63 Hands-On Exercise

- The Random Forest Algorithm Lab

1.64 Support Vector Machines (SVMs)

- A support vector machine (SVM) algorithm is a supervised ML classifier
- It is not fast during learning, but extremely fast in doing predictions which makes it a popular algorithm for real-time applications, like self-driving cars
- Stock (unmodified) SVMs are intended for the binary (two-class) classification where the decision boundary is represented by a hyperplane built by the algorithm
- Non-linear class boundaries are dealt with by kernels which perform mapping (transformation, called *kernel trick*) from one feature space to another feature space
- Kernels are simply functions a bunch of which is already available and which you can create yourself

Notes:

Classification with SVMs goes after finding a hyperplane (a straight line in 2D spaces), which is a decision boundary that separates classes. What's interesting is that the algorithm only cares about the samples that are the closest to the decision boundary (those samples are called support vectors) leaving the rest of samples in the data set without its attention This property of SVMs makes them quite scalable without sacrificing classification accuracy.

1.65 Naive Bayes Classifier (SL)

- Naive Bayes Classifier (*nBC*) is a probabilistic classifier that uses some basic (naive) assumptions about feature independence in the underlying data (variables)

- The *nBC* algorithm only requires a small amount of training data to estimate the sample parameters (means and variances) needed for classification
- The *nBC* algorithm is based on the Bayesian decision theory
- One of the practical implementation of *nBC* is in spam email filtering applications and message board postings classification (e.g. as abusive, etc.)

Notes:

We estimate the probability of a feature by dividing the number of times the feature is of a particular value by the total number of instances in the dataset.

1.66 Naive Bayesian Probabilistic Model in a Nutshell

- *nBC* makes a decision about test data instance classification based on the highest conditional probability of that instance belonging to a class
- The probabilistic model used by *nBC* is based on the Bayes' formula
- In the task of classifying test instances that has features w, x,y,z into two classes (say, A and B), you need to find two probabilities for each test instance:
  - $p1(w_i,x_i,y_i,z_i)$ – the probability of an instance belonging to class A
  - $p2(w_i,x_i,y_i,z_i)$ – the probability of the same instance belonging to class B
  - Then perform a simple comparison of the values

    If p1 > p2, then the instance is deemed by *nBC* as belonging to A

    Otherwise (p1 < p2), the decision is made that the instance belongs to class B

1.67 Bayes Formula

- Bayes' theorem yields the following formula for calculating the conditional probability:

$$p(C|F_1,\ldots,F_n) = \frac{p(C)\ p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}$$

  - where the dependent class variable C with a small number of outcomes or classes is conditional on several feature variables, F_1 through F_n
- For discussion of the Naive Bayes classifier, see http://en.wikipedia.org/wiki/Naive_Bayes_classifier

1.68 Classification of Documents with Naive Bayes

- Document classification is one of the important application of *nBC*
- Documents are made up of words each of which represents an independent feature
- Compact specialized vocabularies are used to minimize the number of words (features)
- *nBC* (naively) treats words as independent features even those some words are likely to be used together with other words (like in common phrases and idiomatic expressions)
- The training phase involves manual classification of documents
- The test documents are converted into word/token vectors that are compared with documents of various supported classes
- Tests that score highest in a class, are treated as belonging to that class

1.69 Unsupervised Learning Type: Clustering

- Clustering (grouping similar items together) is a type of unsupervised machine learning
- Groups of similar observations are referred to as clusters
- Clustering algorithms automatically define a set of criteria (features) to belong to a cluster
- The criteria of similarity or affinity are determined by cluster identification algorithms, such as k-Means, that take data sets as input and try grouping elements with similar features together and reporting on the grouping details
- The key difference between clustering and classification is that in classification you must know criteria for classification; this is not the case with clustering
- Other terms that refer to the clustering concept are:
  - Automatic classification, numerical taxonomy, and typological analysis

1.70 Clustering Examples

- Building groups of genes with related expression patterns [https://en.wikipedia.org/wiki/Gene_expression]
- Customer segmentation
- Grouping experiment outcomes
- Differentiating social network communities

1.71 k-Means Clustering (UL)

- One of the most widely used cluster algorithm in data mining
- Works by forming *k* (user-defined value) clusters for a given dataset

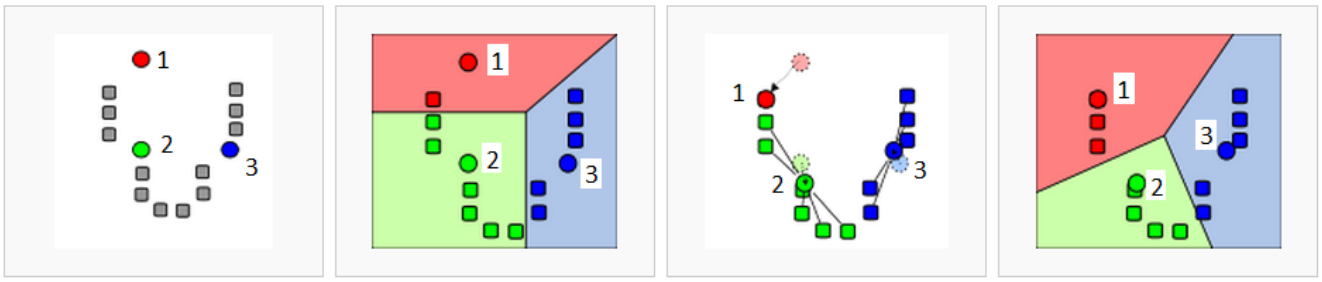
- The value of k is user-defined and presupposes understanding of the underlying data set by the user

- Each cluster is formed around the center point known as the centroid
- It uses a variety of distance metrics to classify items around the centroid
  - Most popular is the Euclidean distance
- Variables in datasets should have numeric values for distance calculation; nominal values should be mapped into numeric values as well
- The algorithm is computationally intensive
  - Efficient optimization techniques are used to improve the overall performance of *k-Means*

1.72 k-Means Clustering in a Nutshell

- The *k-Means* algorithm starts with randomly generating k centroids for future clusters
  - Each point in the dataset is assigned to a cluster based on the closest centroid using the lowest distance metric
  - For every cluster, the mean of the points in that cluster is calculated
  - Centroid is assigned to the mean
  - Repeat the process until there is no changes in cluster assignment of points in the dataset (points stop changing clusters)

1.73 k-Means Clustering in a Nutshell



1) *k* initial "means" (in this case *k*=3) are randomly generated within the data domain

2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3) The centroid of each of the *k* clusters becomes the new mean.

4) Steps 2 and 3 are repeated until convergence has been reached.

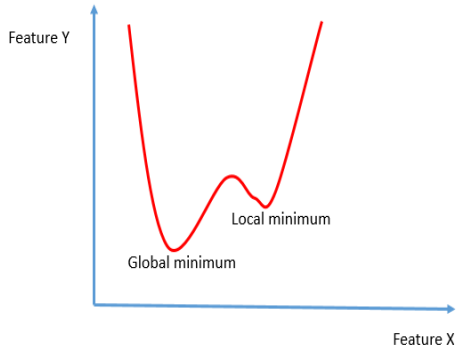
Adapted from

http://en.wikipedia.org/wiki/File:K_Means_Example_Step_2.svg

1.74 k-Means Characteristics

- k-Means is one of the fastest clustering algorithms available
- It always converges; however, it may converge in local minima
- However, ff you run the algorithm more than one time, each time with different initial centroid assignment (positioning), you may get different cluster centroids (clusters)
- To assess the best run outcome, use the within-cluster data points cohesion expressed by a sum of squared errors in a cluster (also referred to as within-cluster inertia) - the one with the minimum value of this factor is your best choice

1.75 Global vs Local Minimum Explained



1.76 Hands-On Exercise

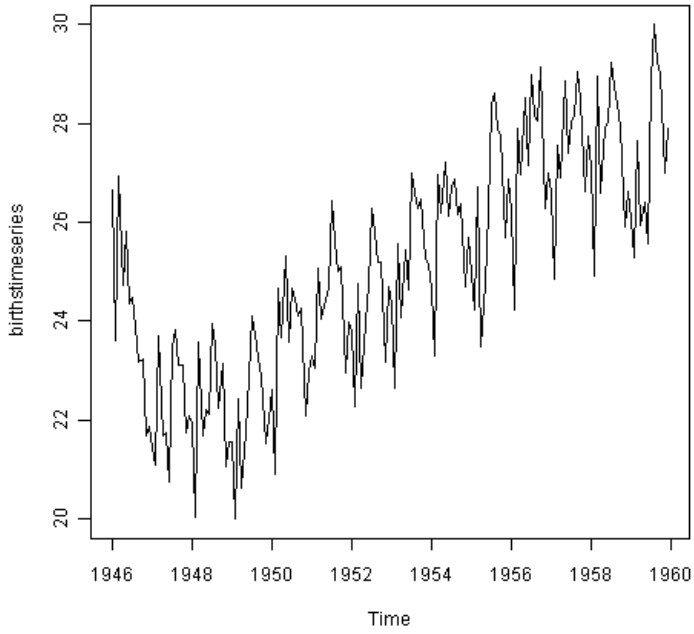
- The k-means Algorithm Lab

1.77 Time-Series Analysis

- The goal of most statistical models is to predict the value of the response variable based on a set of predictor variables
  - Observations are considered independent and their sequence of no effect on the response variable
- Time series analysis proceeds from the opposite direction: previous observations are indispensable in predicting future observations
- Time-series are used to analyze historical data in order to derive insights from underlying data as well as forecast / predict future values based on previously observed values
- An examples of time series are the daily closing values of the Dow Jones Industrial Average and NASDAQ Composite index

1.78 Decomposing Time-Series

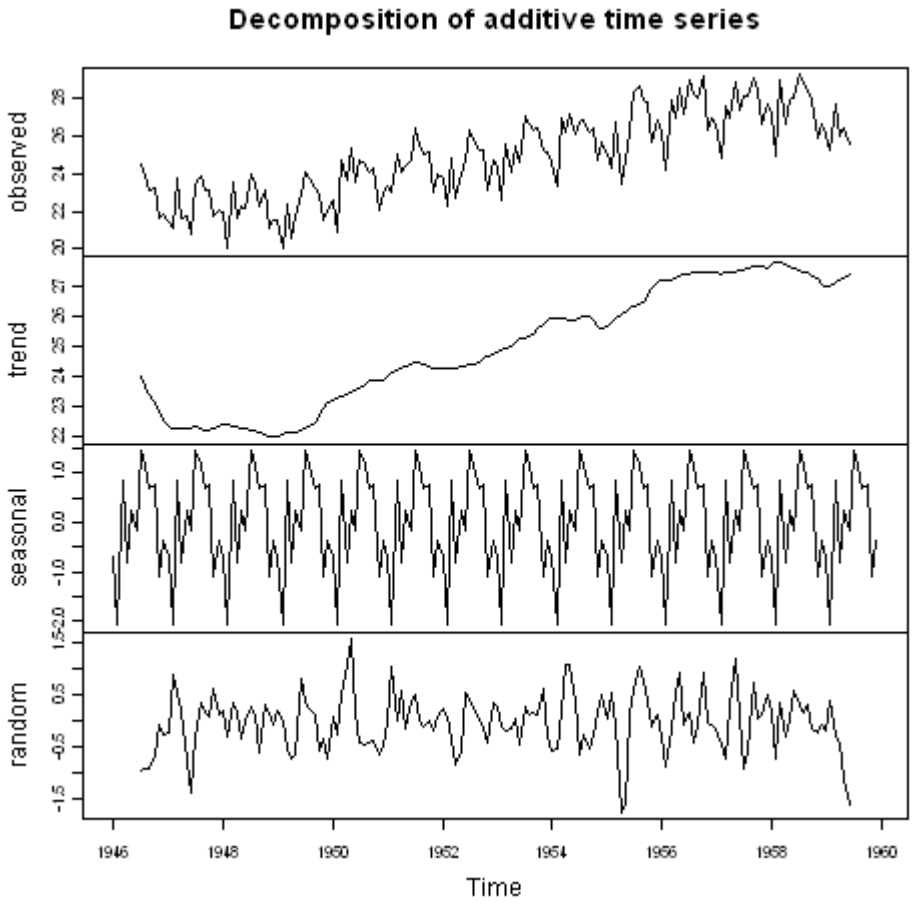
- Building time-series models usually involves decomposing a time series into a trend component (e.g. moving average), irregular (e.g. seasonal) component(s), and sometimes random components (information noise)



The time series of the number of births per month in New York City

Source: http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html

1.79 Decomposing Time-Series



Notes:

Additive time-series models are used when the random fluctuations in the data are roughly constant in size over time and you can isolate them from the data under analysis.

1.80 A Better Algorithm or More Data?

- If you model does not yield the expected level of accuracy, you have a choice between:

- Tuning the model using hyperparamers
  - ✔ Hyperparameter is a parameter whose value is set before model training
- Another (and, hopefully, better) learning algorithm
- More training data
  - ✔ You may want to add more features (feature engineering is important) but be aware of the Curse of Dimensionality

- Generally, ML practitioners have this rule of thumb:
  - A dumb algorithm with enough data to feed it beats a smart algorithm that is starved with small amount of data

1.81 Summary

- In this chapter, we reviewed terminology used in Data Science and ML, as well as a number of algorithms and common analytical methods used in Data Science, including:
  - Dimensionality reduction (isomap and PCA)
  - k-Nearest Neighbors
  - Decision Trees and Random Forest
  - Support Vector Machines (SVMs)
  - Naive Bayes Classifier
  - Cluster Analysis with k-Means
  - Regression Analysis
  - Time-Series Analysis