

Data Science with Python

(with `scikit.learn`)

Big Data Analytics Research Group
University of Limerick

Data Science with Python

- (i) Some Opening Comments
- (ii) Python Environment
- (iii) What is Machine Learning?
- (iv) Useful Packages
- (iv) `scikit.learn`

What is PyData?



PyData : Conference Mission

- ▶ PyData is a gathering of users and developers of data analysis tools in Python.
- ▶ The goals are to provide Python enthusiasts a place to share ideas and learn from each other about how best to apply our language and tools to ever-evolving challenges in the vast realm of data management, processing, analytics, and visualization.

PyData : Conference Mission

- ▶ We aim to be an accessible, community-driven conference, with tutorials for novices, advanced topical workshops for practitioners, and opportunities for package developers and users to meet in person.
- ▶ A major goal of the conference is to provide a venue for users across all the various domains of data analysis to share their experiences and their techniques, as well as highlight the triumphs and potential pitfalls of using Python for certain kinds of problems.



CONTINUUM

ANALYTICS



Headquarters: **Austin, TX**

Description: Continuum Analytics provides Python-based data analytics solutions and services for organizations to analyze, manage and visualize big data.

Founders: **Peter Wang, Travis Oliphant**

Categories: **Analytics**

Website: **<http://www.continuum.io>**



PyData Berlin Conference

[Register](#)[CFP](#)[Venue](#)[News](#)[Speakers](#)[Schedule](#)A wide-angle photograph of the Berlin skyline at sunset. The sky is a deep orange and yellow, with scattered clouds. The city's buildings are silhouetted against the bright horizon. On the right side of the image, the Berlin TV Tower (Fernsehturm Berlin) stands prominently, its spherical observation deck clearly visible. The text "Learn about Python Tools and Algorithms for Data Science" is overlaid in white on the left side of the image.

Learn about Python Tools and
Algorithms for Data Science

PyData Berlin Conference

PyData Berlin 2015

May 29-30

PyData is the home for all things related to the use of Python in data management and analysis. It brings together Python enthusiasts at a novice level and includes Tutorials and corresponding talks as well as advanced talks by experts and package developers. The conference not only focuses on the application of data science tools but also on underlying algorithms and patterns. After a very successful PyData Berlin 2014 hosted at the BCC we are thrilled to announce PyData Berlin 2015, this time taking place at Betahaus Berlin.

[Registration](#) is now open using Eventbrite.

We have now opened our [CFP](#) and are looking forward to your contributions.

For upcoming news please follow [pydataberlin](#) on Twitter.

PyData Berlin Conference

Dates for 2016 : May 20th and 21st.

Also planned

- PyData Madrid
- PyData Amsterdam
- PyData London
- PyData Paris

PyCon Ireland 2014

[Home](#) [PyCon 2014](#) [Location ▼](#) [Sponsorship ▼](#) [Conference ▼](#) [Sprints](#)

[About ▼](#)

Introduction

This years PyCon Ireland 2014 will be held on **Sat 11th - Sun 12th October** in the **Ballsbridge Hotel**. Followed by two days of **Sprints on Mon 13th and Tuesday 14th October** also in the Ballsbridge Hotel.

- ▶ Dates for 2016 event to be announced shortly.
- ▶ PyData track to be included again



PyData London 2015

June 19-21

Hosted by Bloomberg

SPONSORS



CONTINUUM

Data Science with Python



What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

(Python.org)

What is Python?

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

(Python.org)

History of Python

Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum in the Netherlands as a successor of a language called ABC. Guido remains Python's principal author, although it includes many contributions from others.



WORK

Python Displacing R As The Programming Language For Data Science

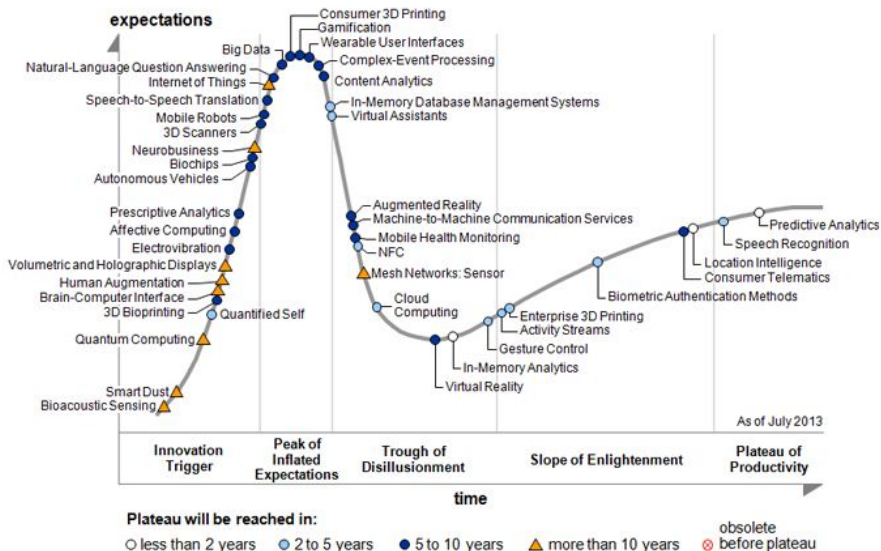
R remains popular with the PhDs of data science, but as data moves mainstream, Python is taking over.

MATT ASAY · NOV 25, 2013

10.8K
SHARES



Figure 1. Hype Cycle for Emerging Technologies, 2013




INFOWORLD TECH WATCH

By **Matt Assay** | [Follow](#)
About

Informed news analysis every weekday

In data science, the R language is swallowing Python



Credit: flickr/Torkild Retvedt

According to a new survey, Python's data science training wheels increasingly lead to the R language

MORE LIKE THIS


Why R? The pros and cons of the R language



Review: Tableau makes sophisticated analysis a snap



Open source lives! The R project is the real deal

on IDG Answers

How is data stored and accessed in the cloud?





Important Components of the Python Computing Environment

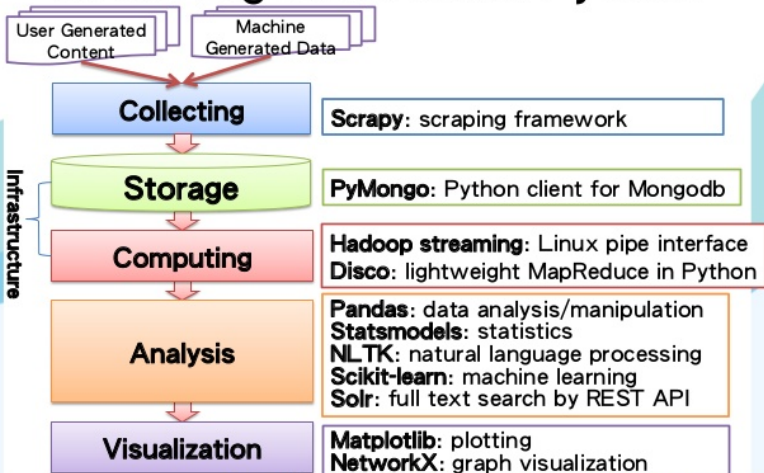


pandas

$$y_H = \beta x_H + \mu_H + e_H$$



When Big Data meet Python



http://www.slideshare.net/jimmy_lai/when-big-data-meet-python

Continuum Analytics Anaconda

Continuum Analytics

Welcome to Anaconda's doc

docs.continuum.io/anaconda/

Anaconda 1.0 documentation »

next | modules | index

Table Of Contents

Welcome to Anaconda's documentation!

- Anaconda Pro Edition v. 1.0
- Anaconda Community Edition v. 1.0
 - Easy, Scalable Distributed Data Analysis in Python
 - What's Included
 - What's New?
- Package Documentation
- Packages included in Anaconda v.1.0
- Known Issues
- End User License Agreement
- Indices and tables

Next topic

Anaconda Install

This Page

Show Source

Quick search

Go

Enter search terms or a module, class or function name.

Welcome to Anaconda's documentation!

Contents

- Welcome to Anaconda's documentation!
 - Anaconda Pro Edition v. 1.0
 - Anaconda Community Edition v. 1.0
 - Package Documentation
 - Packages included in Anaconda v.1.0
 - Known Issues
 - End User License Agreement
 - Indices and tables

Anaconda Pro Edition v. 1.0

Anaconda Pro extends the easy to use Anaconda Community Edition with enterprise features that will enhance your ability to deal with large data-files, accelerate code that works on array-based data, configure execution environments for code, as well as provide access to cutting-edge algorithms and optimizations.

In addition to all the packages that come with Anaconda CE, Anaconda Pro includes

- IOPro — fast access to data-bases and text files
- NumbaPro — fast vectorization utilizing multiple cores and GPUs
- WiseRF — a fast, multi-core implementation of the Random Forest algorithm from Wise.IO
- Python Environments — the ability to create named "Python environments" to mix and match different versions of Python, NumPy, SciPy, etc. and easily switch between them

Purchase of Anaconda Pro gives you access to a year of free updates.

Download Anaconda Now!

Anaconda

- ▶ Anaconda, a free product of Continuum Analytics (www.continuum.io), is a virtually complete scientific stack (i.e. distribution) for Python.
- ▶ It includes both the core Python interpreter and standard libraries as well as most modules required for data analysis.

Anaconda

- ▶ Anaconda is free to use and modules for accelerating the performance of linear algebra on Intel processors using the **Math Kernel Library** (MKL) are available (free to academic users and for a small cost to non-academic users).
- ▶ Continuum Analytics also provides other high-performance modules for reading large data files or using the GPU to further accelerate performance for an additional, modest charge.

Installing Anaconda

Most importantly, installation is extraordinarily easy on Windows, Linux and OS X. Anaconda is also simple to update to the latest version using

```
conda update conda  
conda update anaconda
```

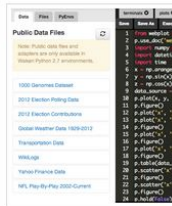
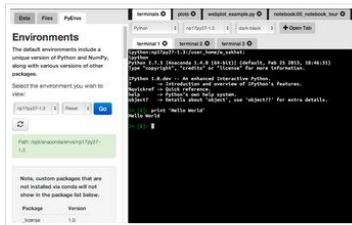
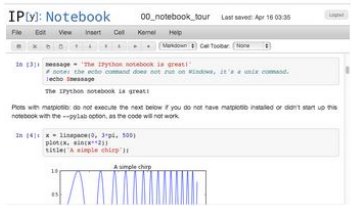


Plans & Pricing

Run Wakari in Your Data Center



Web-based Python Data Analysis



Shell Access from the Browser



Up and Running in less than 2

Wakari

Wakari is a collaborative data analytics platform that includes tools to explore data, develop analytics scripts, collaborate with IPython notebooks, visualize, and share data analysis and findings. Save time and money by getting right down to business analyzing data with the fully-configured Wakari. In the cloud or on your own servers, Wakari makes accessing your compute resources and data, reproducing your process, and sharing your results easy.

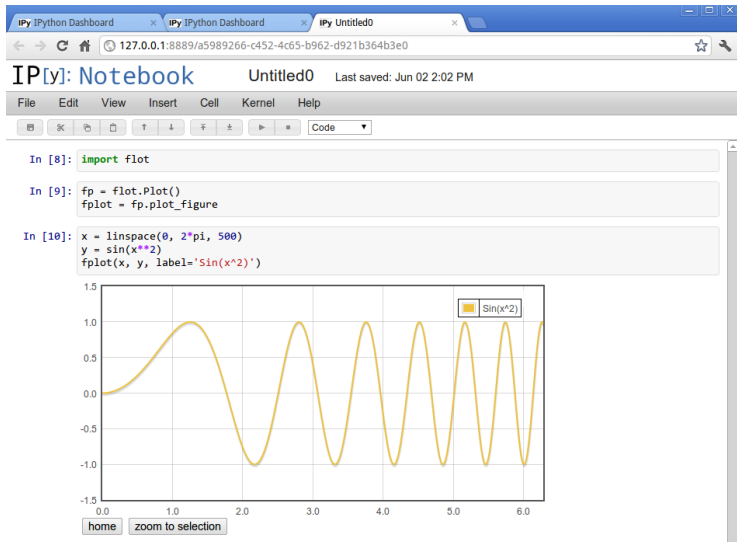
(Source: Continuum Analytics)

IPython and IPython Notebooks

IPython / Jupyter Notebooks

- ▶ IPython provides an interactive Python environment which enhances productivity when developing code or performing interactive data analysis.
- ▶ The IPython Notebook is a web-based interactive computational environment where you can combine code execution, text, mathematics, plots and rich media into a single document.

IPython Notebook



Ipython Notebook / Jupyter



Evolved from the IPython Project

The language-agnostic parts of IPython are getting a new home in Project Jupyter

IPython

- Interactive Python shell
- Python kernel for Jupyter
- Interactive Parallel Python

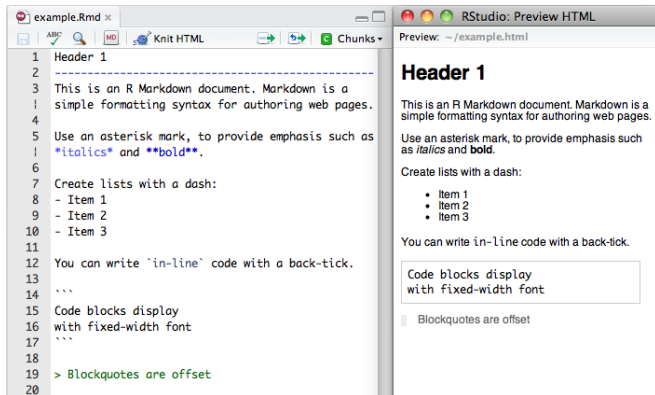
Jupyter

- Rich REPL Protocol
- Notebook (format, environment, conversion)
- [JupyterHub](#) (multi-user notebook server)
- [More...](#)

Markdown

Markdown is a text-to-HTML conversion tool for web writers.

Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML).



The screenshot displays the RStudio interface with a document titled 'example.Rmd' on the left and a 'Preview HTML' window on the right. The document content is as follows:

```
1 Header 1
2 -----
3 This is an R Markdown document. Markdown is a
4 simple formatting syntax for authoring web pages.
5 Use an asterisk mark, to provide emphasis such as
6 *italics* and **bold**.
7 Create lists with a dash:
8 - Item 1
9 - Item 2
10 - Item 3
11
12 You can write `in-line` code with a back-tick.
13
14 ```
15 Code blocks display
16 with fixed-width font
17 ```
18
19 > Blockquotes are offset
20
```

The 'Preview HTML' window shows the rendered output of this document:

Header 1

This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages.

Use an asterisk mark, to provide emphasis such as *italics* and **bold**.

Create lists with a dash:

- Item 1
- Item 2
- Item 3

You can write `in-line` code with a back-tick.

```
Code blocks display
with fixed-width font
```

Blockquotes are offset

Versions of Python

Two Main Versions of Python

- ▶ Version 2.7
- ▶ Version 3

Python Coding Conventions

There are a number of common practices which can be adopted to produce Python code which looks more like code found in other modules:

- ▶ Use 4 spaces to indent blocks avoid using tab, except when an editor automatically converts tabs to 4 spaces
- ▶ Avoid more than 4 levels of nesting, if possible
- ▶ Limit lines to 79 characters. The `\` symbol can be used to break long lines
- ▶ Use two blank lines to separate functions, and one to separate logical sections in a function.

Python Coding Conventions

- ▶ Use ASCII mode in text editors, not UTF-8
- ▶ One module per import line
- ▶ Avoid `from module import *` (for any module). Use either `from module import func1, func2` or `import module as shortname`.
- ▶ Follow the NumPy guidelines for documenting functions

Machine Learning with Python

What is Machine Learning

- ▶ Machine Learning is a discipline involving algorithms designed to find patterns in and make predictions about data.
- ▶ It is nearly ubiquitous in our world today, and used in everything from web searches to financial forecasts to studies of the nature of the Universe.


What is Machine Learning


- ▶ Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed.
- ▶ Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data.


What is Machine Learning


- ▶ The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns.
- ▶ However, instead of extracting data for human comprehension – as is the case in data mining applications – machine learning uses that data to improve the program's own understanding.
- ▶ Machine learning programs detect patterns in data and adjust program actions accordingly.




Recommender Engines

 Try Prime


All 

 Father's Day Savings
Sponsored by

Shop by Department  Kevin's Amazon.com Today's Deals Gift Cards Sell Help

Hello, Kevin **Your Account**  Try Prime  Wish List 

Your Amazon.com Your Browsing History Recommended For You Improve Your Recommendations Your Profile Learn More

**Kevin's Amazon**

ON ORDER
0 items

AMAZON PRIME
Join Prime
[View benefits](#)

GIFT CARD BALANCE
\$20⁰⁰
[Manage cards](#)

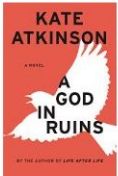
CUSTOMER SINCE
2007

books





The P.G. Wodehouse ...
N. T. P. Murphy
★★★★★ (1)
~~\$18.95~~ \$14.86
[Why recommended?](#)



A God in Ruins: A ...
Kate Atkinson
★★★★☆ (34)
~~\$28.00~~ \$16.90
[Why recommended?](#)



The Luck Stone (The ...
P. G. Wodehouse
[Why recommended?](#)



Bring on the Girls ...
P. G. Wodehouse
★★★★★ (5)
[Why recommended?](#)



Louder and Funnier
P. G. Wodehouse
~~\$19.95~~ \$15.43
[Why recommended?](#)

[See all recommendations in Books](#)

Facebook's Newsfeed

- ▶ For example, Facebook's News Feed changes according to the user's personal interactions with other users.
- ▶ If a user frequently tags a friend in photos, writes on his wall or "likes" his links, the News Feed will show more of that friend's activity in the user's News Feed due to presumed closeness.

Data Visualization with Python

Seaborn: statistical data visualization

Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.

For a brief introduction to the ideas behind the package, you can read the [introductory notes](#).

Much more detail can be found in the seaborn [tutorial](#). You can also browse the [example gallery](#) or [API reference](#) to see the kind of tools that are available.

To check out the code, report a bug, or contribute a new feature, please visit the [github repository](#). You can also get in touch on [twitter](#).

Documentation

Tutorial

- [An introduction to seaborn](#)
- [What's new in the package](#)

seaborn

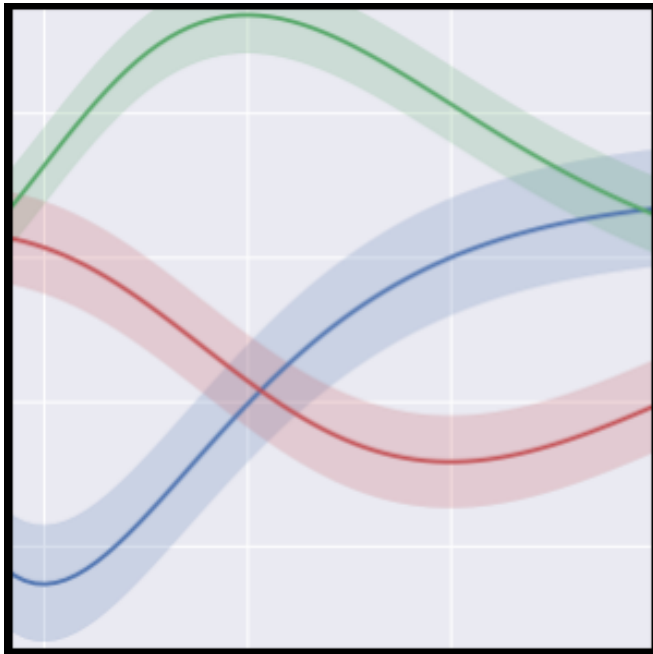
An introduction to seaborn

Seaborn is a library for making attractive and informative statistical graphics in Python. It is built on top of [matplotlib](#) and tightly integrated with the [PyData](#) stack, including support for [numpy](#) and [pandas](#) data structures and statistical routines from [scipy](#) and [statsmodels](#).

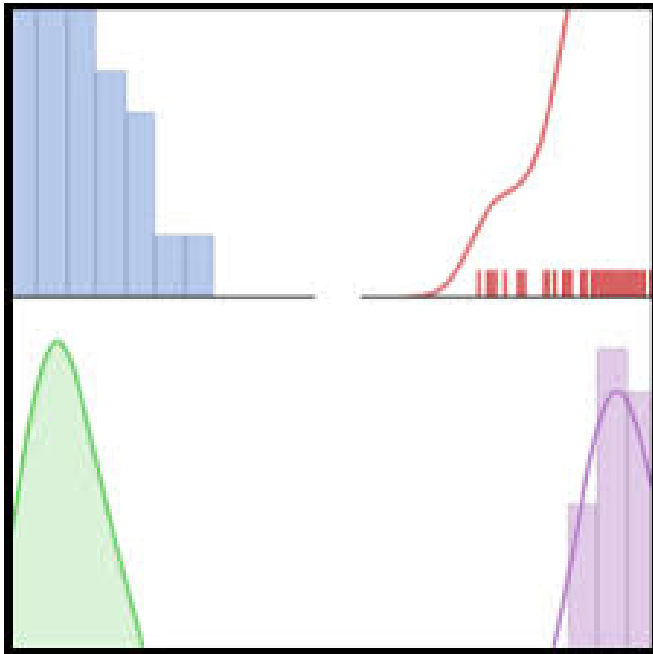
Some of the features that seaborn offers are

- Several [built-in themes](#) that improve on the default matplotlib aesthetics
- Tools for choosing [color palettes](#) to make beautiful plots that reveal patterns in your data
- Functions for visualizing [univariate](#) and [bivariate](#) distributions or for [comparing](#) them between subsets of data
- Tools that fit and visualize [linear regression](#) models for different kinds of [independent](#) and [dependent](#) variables
- Functions that visualize [matrices of data](#) and use clustering algorithms to [discover structure](#) in those matrices
- A function to plot [statistical timeseries](#) data with flexible estimation and [representation](#) of uncertainty around the estimate
- High-level abstractions for structuring [grids of plots](#) that let you easily build [complex](#) visualizations

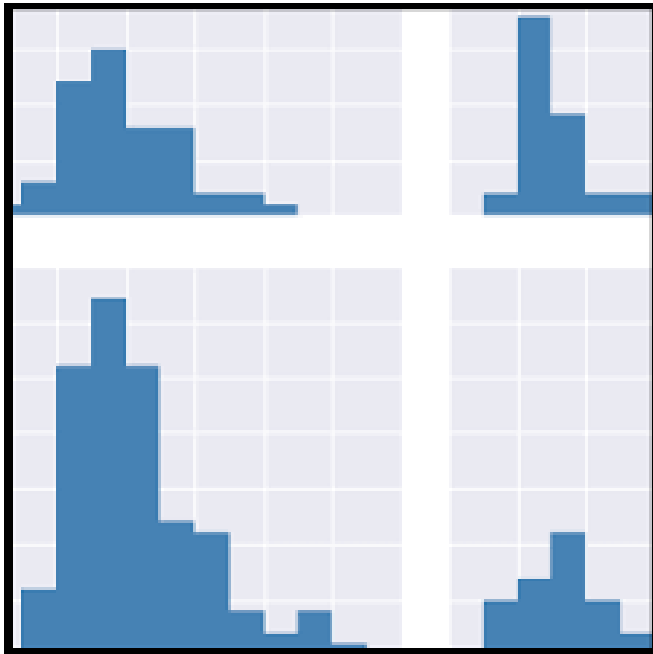
seaborn



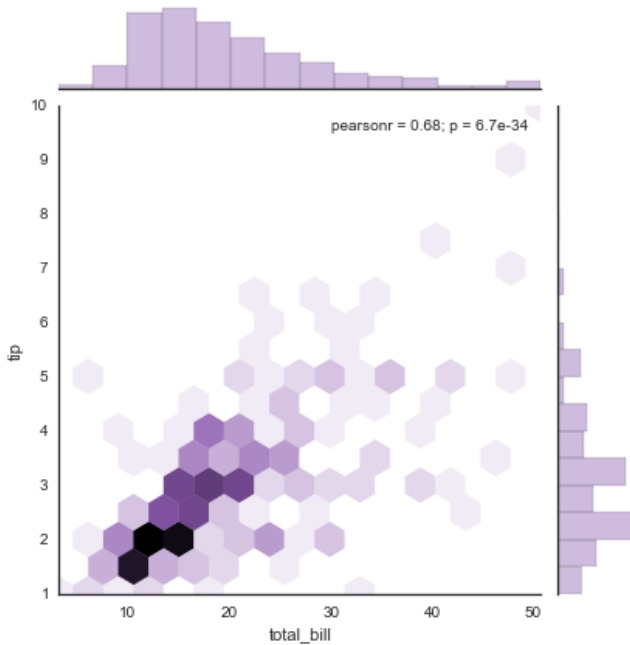
seaborn



seaborn



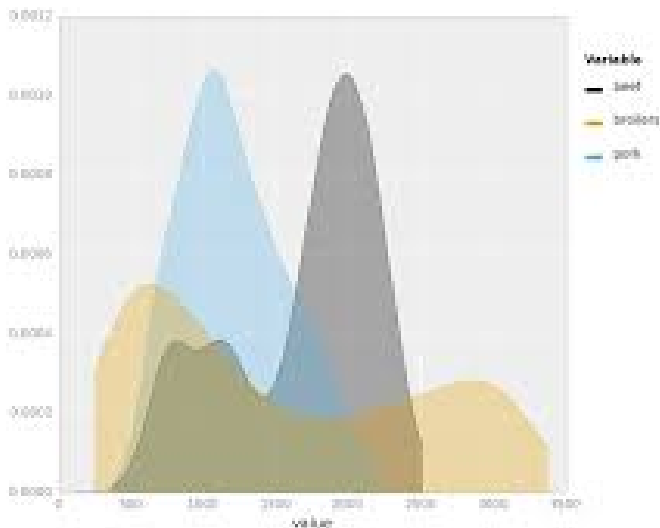
seaborn



Bokeh Data Visualization



Bokeh Data Visualization



Bokeh Data Visualization

Bokeh Data Visualization

- ▶ interactive graphics for the web
- ▶ designed for large data sets
- ▶ Designed for streaming data
- ▶ Native interface in python
- ▶ Fast javascript components
- ▶ DARPA funded
- ▶ v.01 relase imminent

Three Core Packages

1. numpy
2. pandas
3. scipy

pandas

$$y_{it} = \beta x_{it} + \mu_i + \epsilon_{it}$$



SciPy.org



Sponsored By
ENTHOUGHT

The numpy package

- ▶ The Python programming language was not initially designed for numerical computing, but attracted the attention of the scientific/engineering community early on.
- ▶ NumPy is an extension to the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

The numpy package

- ▶ The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers.
- ▶ In 2005, Travis Oliphant created NumPy by incorporating features of Numarray into Numeric with extensive modifications.

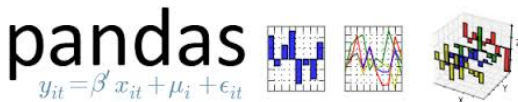
The numpy package

- ▶ NumPy is open source and has many contributors.
- ▶ **Website** <http://www.numpy.org/>

The numpy package

Useful Commands for simulation exercises

- ▶ `random.randint(a, b)` - Return a random integer N such that $a \leq N \leq b$.
- ▶ `random.choice(seq)` - return a random element from the non-empty sequence `seq`. If `seq` is empty, raises `IndexError`.
- ▶ `random.sample(population, k)` - Return a k length list of unique elements chosen from the population sequence. Used for random *sampling without replacement*.



- ▶ pandas is a high-performance module that provides a comprehensive set of structures for working with data.
- ▶ pandas excels at handling structured data, such as data sets containing many variables, working with missing values and merging across multiple data sets.

Data Wrangling with Pandas, NumPy, and IPython

Python for Data Analysis



O'REILLY®

Wes McKinney

- ▶ While extremely useful, pandas is not an essential component of the Python scientific stack unlike NumPy, SciPy or matplotlib, and so while pandas doesn't make data analysis possible in Python, it makes it much easier.
- ▶ pandas also provides high-performance, robust methods for importing from and exporting to a wide range of formats.

Data Structures

pandas provides a set of data structures which include Series, DataFrames and Panels.

- ▶ **Series** are 1-dimensional arrays.
- ▶ **DataFrames** are collections of Series and so are 2-dimensional,
- ▶ **Panels** are collections of DataFrames, and so are 3-dimensional.

- ▶ SciPy (pronounced Sigh Pie) is an open source Python library used by scientists, analysts, and engineers doing scientific computing and technical computing.
- ▶ SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

Simon Blomberg:

From R's fortunes package: To paraphrase provocatively, 'machine learning is statistics minus any checking of models and assumptions'. -- Brian D. Ripley (about the difference between machine learning and statistics) useR! 2004, Vienna (May 2004) :- Season's Greetings!

Andrew Gelman:

In that case, maybe we should get rid of checking of models and assumptions more often. Then maybe we'd be able to solve some of the problems that the machine learning people can solve but we can't!

Machine Learning is Statistics minus any checking of models or assumptions

The Data Science Profession

Data Science Retreat (Berlin)

MOOC have not decreased the barrier of entry to machine-learning.

Nowadays, you cannot be 'the guy who knows how to run (insert off-the-shelf-algo-here)'.

In dataland, that's the equivalent to being a code monkey. MOOCs and superb libraries (scikit-learn, R's ecosystem) made sure there is plenty of people who can throw say a random forest to a problem. In the modern world, this is not adding that much value.



- ▶ scikit-learn is an open source machine learning library for the Python programming language.
- ▶ scikit-learn features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting, k-means and DBSCAN.
- ▶ scikit-learn is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Sci-Kit Learn Site info

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: *SVM, nearest neighbors, random forest, ...* — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: *SVR, ridge regression, Lasso, ...* — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: *k-Means, spectral clustering, mean-shift, ...* — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: *PCA, feature selection, non-negative matrix factorization.* — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: *grid search, cross validation, metrics.* — Examples

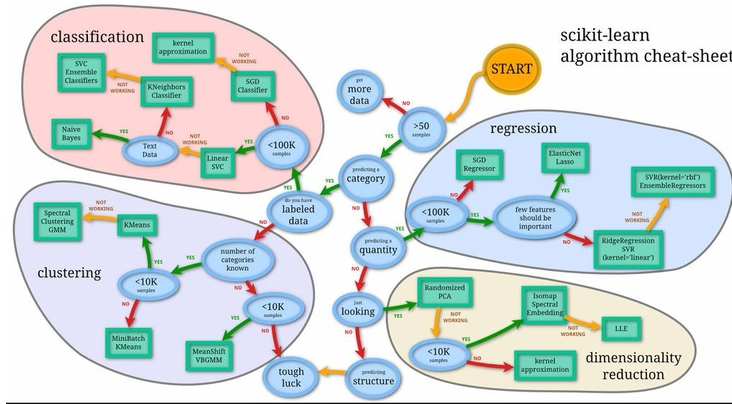
Preprocessing

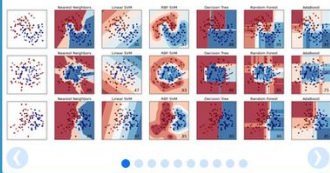
Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

Modules: *preprocessing, feature extraction.* — Examples

scikit-learn algorithm cheat-sheet





scikit-learn

Machine Learning in Python

- 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

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: *SVM, nearest neighbors, random forest, ...*

— Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: *SVR, ridge regression, Lasso, ...*

— Examples

Clustering

Automatic grouping of similar objects in

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: *k-Means, spectral clustering, mean-shift, ...*

— E

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: *PCA, feature selection, non-negative matrix factorization.*

— Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: *grid search, cross validation, metrics.*

— Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data s text for use with machine learning algo

Modules: *preprocessing, feature extra*

— E

Classification

- ▶ **Description:** Identifying to which category an object belongs to.
- ▶ **Applications:** Spam detection, Image recognition.
- ▶ **Algorithms:** SVM, nearest neighbors, random forest,

Regression

- ▶ **Description:** Predicting a continuous-valued attribute associated with an object.
- ▶ **Applications:** Drug response, Stock prices.
- ▶ **Algorithms:** SVR, ridge regression, Lasso,

Clustering

Automatic grouping of similar objects into sets. Applications: Customer segmentation, Grouping experiment outcomes Algorithms: k-Means, spectral clustering, mean-shift, ...

Dimensionality Reduction

- ▶ **Description:** Reducing the number of random variables to consider.
- ▶ **Applications:** Visualization, Increased efficiency
- ▶ **Algorithms:** PCA, feature selection, non-negative matrix factorization.

Model selection

- ▶ **Description:** Comparing, validating and choosing parameters and models.
- ▶ **Goal:** Improved accuracy via parameter tuning
- ▶ **Modules:** grid search, cross validation, metrics

Preprocessing

- ▶ **Description:** Feature extraction and normalization.
- ▶ **Application:** Transforming input data such as text for use with machine learning algorithms.
- ▶ **Modules:** preprocessing, feature extraction.

statsmodel

- ▶ statsmodels provides a large range of cross-sectional models aswell as some time-series models.
- ▶ statsmodels uses a model descriptive language (provided via the Python package patsy) to formulate the model when working with pandas DataFrames.
- ▶ Models supported include linear regression, generalized linear models, limited dependent variable models, ARMA and VAR models.





gensim

topic modelling for humans



Download

latest version from the Python Package Index



Direct install with:
`easy_install -U gensim`

Home

Tutorials

Install

Support

API

About

```
>>> from gensim import corpora, models, similarities
>>>
>>> # Load corpus iterator from a Matrix Market file on disk.
>>> corpus = corpora.MmCorpus('/path/to/corpus.mm')
>>>
>>> # Initialize Latent Semantic Indexing with 200 dimensions.
>>> lsi = models.LsiModel(corpus, num_topics=200)
>>>
>>> # Convert another corpus to the latent space and index it.
>>> index = similarities.MatrixSimilarity(lsi[another_corpus])
>>>
>>> # Compute similarity of a query vs. indexed documents
>>> sims = index[query]
```

Gensim is a FREE Python library



Scalable statistical semantics



Analyze plain-text documents for semantic structure



Retrieve semantically similar documents

The tutorials are organized as a series of examples that highlight various features of *gensim*. It is assumed that the reader is familiar with the [Python language](#), has [installed gensim](#) and read the [introduction](#).

The examples are divided into parts on:

- [Corpora and Vector Spaces](#)
 - [From Strings to Vectors](#)
 - [Corpus Streaming – One Document at a Time](#)
 - [Corpus Formats](#)
 - [Compatibility with NumPy and SciPy](#)
- [Topics and Transformations](#)
 - [Transformation interface](#)
 - [Available transformations](#)
- [Similarity Queries](#)
 - [Similarity interface](#)
 - [Where next?](#)
- [Experiments on the English Wikipedia](#)
 - [Preparing the corpus](#)
 - [Latent Semantic Analysis](#)
 - [Latent Dirichlet Allocation](#)
- [Distributed Computing](#)
 - [Why distributed computing?](#)
 - [Prerequisites](#)
 - [Core concepts](#)
 - [Available distributed algorithms](#)

Beautiful Soup

Beautiful Soup 4.2.0 documentation »

[index](#)

Table Of Contents

Beautiful Soup Documentation

- Getting help
- Quick Start
- Installing Beautiful Soup

- Problems after installation
- Installing a parser

Making the soup

Kinds of objects

- Tag
 - Name
 - Attributes
 - Multi-valued attributes
- NavigableString
- BeautifulSoup
- Comments and other special strings

Navigating the tree

- Going down
 - Navigating using tag names
 - .contents and .children
 - .descendants

Beautiful Soup Documentation

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

These instructions illustrate all major features of Beautiful Soup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

The examples in this documentation should work the same way in Python 2.7 and Python 3.2.

You might be looking for the documentation for [Beautiful Soup 3](#). If so, you should know that Beautiful Soup 3 is no longer being developed, and that Beautiful Soup 4 is recommended for all new projects. If you want to learn about the differences between Beautiful Soup 3 and Beautiful Soup 4, see [Porting code to BS4](#).

This documentation has been translated into other languages by Beautiful Soup users:

- [这篇文档当然还有中文版](#)



Beautiful Soup

Beautiful Soup

- ▶ Beautiful Soup is a Python library for pulling data out of HTML and XML files.
- ▶ It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree.
- ▶ It commonly saves programmers hours or days of work.

Classification with `scikit-learn`

- ▶ Here we look at problem of classification, a situation in which a response is a categorical variable.
- ▶ We will build upon the techniques that we previously discussed in the context of regression and show how they can be transferred to classification problems.
- ▶ Here we will introduce a number of classification techniques, and it will try to convey their corresponding strengths and weaknesses by visually inspecting the decision boundaries for each model.

Classification with `scikit.learn`

- ▶ We provide only a small amount of background on the concepts and techniques we cover, so if you'd like a more thorough explanation check out [Introduction to Statistical Learning](#) or sign up for the free online course run by the book's authors [here](#).

Scikit-learn

- ▶ Scikit-learn is a library that provides a variety of both supervised and unsupervised machine learning techniques.
- ▶ Supervised machine learning refers to the problem of inferring a function from labeled training data, and it comprises both regression and classification.

Unsupervised Learning

- ▶ Unsupervised machine learning, on the other hand, refers to the problem of finding interesting patterns or structure in the data; it comprises techniques such as clustering and dimensionality reduction.
- ▶ In addition to statistical learning techniques, scikit-learn provides utilities for common tasks such as model selection, feature extraction, and feature selection.

Estimators

- ▶ Scikit-learn provides an object-oriented interface centered around the concept of an Estimator.
- ▶ According to the scikit-learn tutorial *An estimator is any object that learns from data; it may be a classification, regression or clustering algorithm or a transformer that extracts/filters useful features from raw data.*

- ▶ The `Estimator.fit` method sets the state of the estimator based on the training data.
- ▶ Usually, the data is comprised of a two-dimensional numpy array `X` of shape `(n_samples, n_predictors)` that holds the so-called feature matrix and a one-dimensional numpy array `y` that holds the responses.
- ▶ Some estimators allow the user to control the fitting behavior.

- ▶ For example, the `sklearn.linear_model.LinearRegression` estimator allows the user to specify whether or not to fit an intercept term.
- ▶ This is done by setting the corresponding constructor arguments of the estimator object:

```
In [3]: from sklearn.linear_model import LinearRegression  
est = LinearRegression(fit_intercept=False)
```

```
from sklearn.linear_model import LinearRegression  
est = LinearRegression(fit_intercept=False)
```

- ▶ During the fitting process, the state of the estimator is stored in instance attributes that have a trailing underscore ('_').
- ▶ For example, the coefficients of a LinearRegression estimator are stored in the attribute `coef_`:

```
import numpy as np

# random training data
X = np.random.rand(10, 2)
y = np.random.randint(2, size=10)
est.fit(X, y)
est.coef_    # access coefficients

# Output : array([ 0.33176871,  0.34910639])
```

Estimators

- ▶ Estimators that can generate predictions provide a `Estimator.predict` method.
- ▶ In the case of regression, `Estimator.predict` will return the predicted regression values; it will return the corresponding class labels in the case of classification.
- ▶ Classifiers that can predict the probability of class membership have a method `Estimator.predict_proba` that returns a two-dimensional numpy array of shape `(n_samples, n_classes)` where the classes are lexicographically ordered.

Classification with Scikit-Learn

Understanding Classification

Although regression and classification appear to be very different they are in fact similar problems.

- ▶ In regression our predictions for the response are real-valued numbers
- ▶ on the other hand, in classification the response is a mutually exclusive class label
- ▶ Example *“Is the email spam?”* or *“Is the credit card transaction fraudulent?”*.

Binary Classification Problems

- ▶ If the number of classes is equal to two, then we call it a binary classification problem; if there are more than two classes, then we call it a multiclass classification problem.
- ▶ In the following we will assume binary classification because it's the more general case, and we can always represent a multiclass problem as a sequence of binary classification problems.

Credit Card Fraud

- ▶ We can also think of classification as a function estimation problem where the function that we want to estimate separates the two classes.
- ▶ This is illustrated in the example below where our goal is to predict whether or not a credit card transaction is fraudulent
- ▶ the dataset is provided by James et al.,

Introduction to Statistical Learning.

Credit Card Fraud

```
: import pandas as pd

df = pd.read_csv('https://d1pqsl2386xqi9.cloudfront.net/notebooks/Default.csv', index_col=0)

# downsample negative cases -- there are many more negatives than positives
indices = np.where(df.default == 'No')[0]
rng = np.random.RandomState(13)
rng.shuffle(indices)
n_pos = (df.default == 'Yes').sum()
df = df.drop(df.index[indices[n_pos:]])

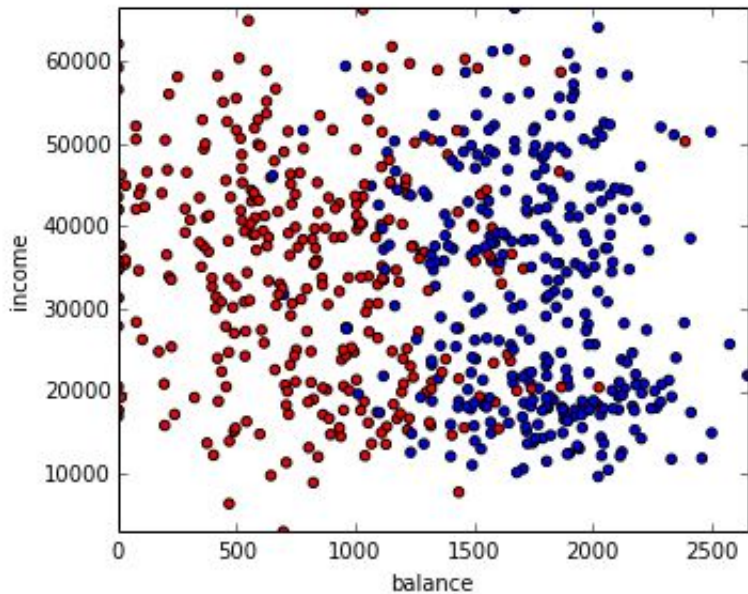
df.head()
```

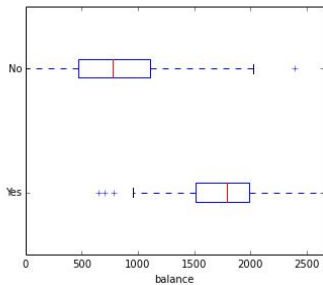
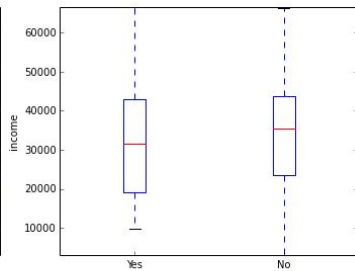
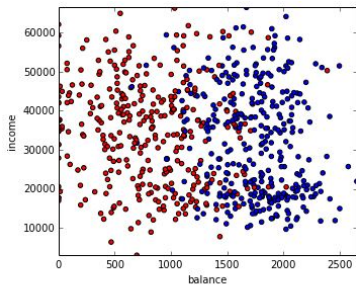
Credit Card Fraud

	default	student	balance	income
20	No	No	1095.072735	26464.631389
38	No	No	351.453472	35087.488648
61	No	No	766.234379	46478.294257
78	No	No	728.373251	45131.718265
79	No	No	76.991291	28392.093412

Credit Card Fraud

- ▶ On the left you can see a scatter plot where fraudulent cases are red dots and non-fraudulent cases are blue dots.
- ▶ A good separation seems to be a vertical line at around a balance of 1400 as indicated by the boxplots on the next slide.





Simple Approach - Linear Regression

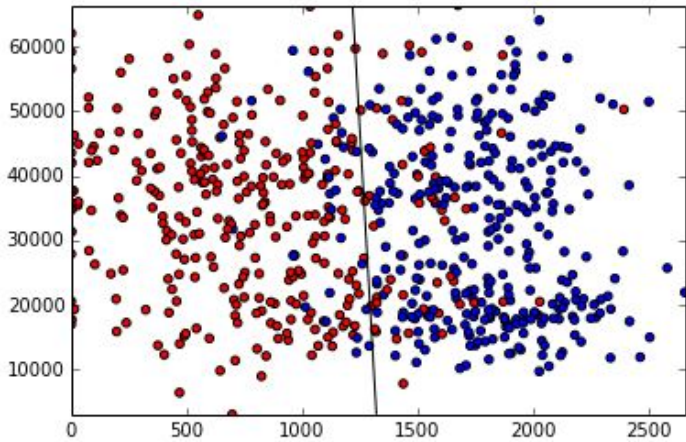
- ▶ A simple approach to binary classification is to simply encode default as a numeric variable with 'Yes' == 1 and 'No' == -1; fit an Ordinary Least Squares regression model and use this model to predict the response as 'Yes' if the regressed value is higher than 0.0 and 'No' otherwise.
- ▶ The points for which the regression model predicts 0.0 lie on the so-called decision surface since we are using a linear regression model, the decision surface is linear as well.

```
from sklearn.linear_model import LinearRegression

# get feature/predictor matrix as numpy array
X = df[['balance', 'income']].values

# encode class labels
classes, y = np.unique(df.default.values, return_inverse=True)
y = (y * 2) - 1 # map {0, 1} to {-1, 1}

# fit OLS regression
est = LinearRegression(fit_intercept=True, normalize=True)
est.fit(X, y)
```

- ▶ Points that lie on the left side of the decision boundary will be classified as negative;
- ▶ Points that lie on the right side, positive.

Confusion Matrix

- ▶ We can assess the performance of the model by looking at the confusion matrix a cross tabulation of the actual and the predicted class labels.
- ▶ The correct classifications are shown in the diagonal of the confusion matrix. The off-diagonal terms show you the **classification errors**.
- ▶ A condensed summary of the model performance is given by the **misclassification rate** determined simply by dividing the number of errors by the total number of cases.

Confusion Matrix

```
from sklearn.metrics import confusion_matrix as sk_confusion_matrix

# the larger operator will return a boolean array which we will cast as integers
y_pred = (2 * (est.predict(X) > 0.0)) - 1

def confusion_matrix(y_test, y_pred):
    cm = sk_confusion_matrix(y, y_pred)
    cm = pd.DataFrame(data=cm, columns=[-1, 1], index=[-1, 1])
    cm.columns.name = 'Predicted label'
    cm.index.name = 'True label'
    error_rate = (y_pred != y).mean()
    print('error rate: %.2f' % error_rate)
    return cm

confusion_matrix(y, y_pred)
```

Confusion Matrix

Predicted label	-1	1
True label		
-1	282	51
1	29	304

Cross Validation

- ▶ In this example we are assessing the model performance on the same data that we used to fit the model.
- ▶ This might be a biased estimate of the models performance, for a classifier that simply memorizes the training data has zero training error but would be totally useless to make predictions.
- ▶ It is much better to assess the model performance on a separate dataset called the test data.
- ▶ Scikit-learn provides a number of ways to compute such held-out estimates of the model performance.
- ▶ One way is to simply split the data into a **training set** and **testing set**.

```
from sklearn.cross_validation import train_test_split

# create 80%-20% train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# fit on training data
est = LinearRegression().fit(X_train, y_train)

# test on data that was not used for fitting
y_pred = (2 * (est.predict(X) > 0.0)) - 1

confusion_matrix(y_test, y_pred)
```

Predicted label	-1	1
True label		
-1	287	46
1	29	304

Classification Techniques

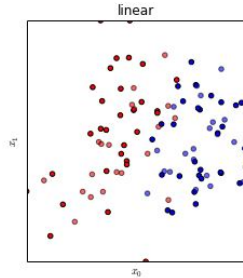
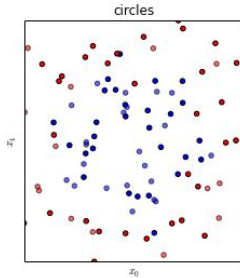
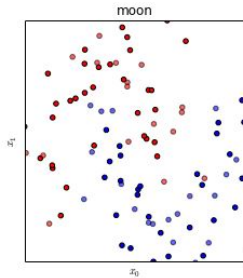
- ▶ Different classification techniques can often be compared using the type of decision surface they can learn.
- ▶ The decision surfaces describe for what values of the predictors the model changes its predictions and it can take several different shapes: piece-wise constant, linear, quadratic, voronoi tessellation, . . .

This next part will introduce three popular classification techniques:

- 1 Logistic Regression,
- 2 Discriminant Analysis,
- 3 Nearest Neighbor.

We will investigate what their strengths and weaknesses are by looking at the decision boundaries they can model. In the following we will use three synthetic datasets that we adopted from this scikit-learn example.

Synthetic Data Sets



Synthetic Data Sets

- ▶ The task in each of the above examples is to separate the red from the blue points.
- ▶ Testing data points are plotted in lighter color.
- ▶ The left example contains two intertwined moon sickles; the middle example is a circle of blues framed by a ring of reds; and the right example shows two linearly separable gaussian blobs.

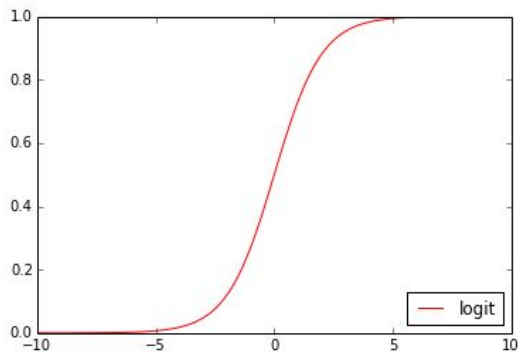
Method 1: Logistic Regression

Logistic Regression

- ▶ Logistic regression can be viewed as an extension of linear regression to classification problems.
- ▶ One of the limitations of linear regression is that it cannot provide class probability estimates.
- ▶ This is often useful, for example, when we want to inspect manually the most fraudulent cases.
- ▶ Basically, we would like to constrain the predictions of the model to the range $[0, 1]$ so that we can interpret them as probability estimates.
- ▶ In Logistic Regression, we use the logit function to clamp predictions from the range $[-\infty, \infty]$ to $[0, 1]$.

Method 1: Logistic Regression

Logistic Transformation



Method 1: Logistic Regression

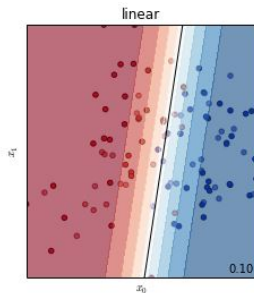
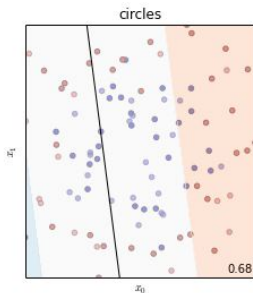
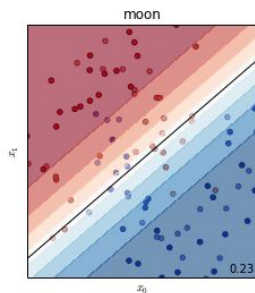
Logistic Regression

- ▶ Logistic regression is available in scikit-learn via the class `sklearn.linear_model.LogisticRegression`.
- ▶ Lets see how Logistic Regression does on our three toy datasets.

Method 1: Logistic Regression

```
from sklearn.linear_model import LogisticRegression  
  
est = LogisticRegression()  
plot_datasets(est)
```


Method 1: Logistic Regression



Method 1: Logistic Regression

Model Appraisal

- ▶ As we can see, a linear decision boundary is not a poor approximation for the moon datasets, although we fail to separate the two tips of the sickles in the center.
- ▶ The cicles dataset, on the other hand, is not well suited for a linear decision boundary.

Method 1: Logistic Regression

Model Appraisal

- ▶ The error rate of 0.68 is in fact worse than random guessing.
- ▶ For the linear dataset we picked in fact the correct model class the error rate of 10% is due to the noise component in our data.
- ▶ The gradient shows you the probability of class membership white shows you that the model is very uncertain about its prediction.

Method 2: Linear Discriminant Analysis

Linear Discriminant Analysis

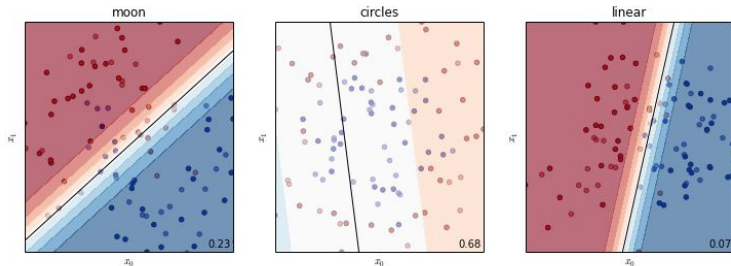
- ▶ Linear discriminant Analysis (LDA) is another popular technique which shares some similarities with Logistic Regression.
- ▶ LDA too finds linear boundary between the two classes where points on side are classified as one class and those on the other as classified as the other class.

Method 2: Linear Discriminant Analysis

```
from sklearn lda import LDA  
  
est = LDA()  
plot_datasets(est)
```

Method 2: Linear Discriminant Analysis

Model Appraisal



(Remark - almost same as logistic regression)

Method 2: Linear Discriminant Analysis

Linear Discriminant Analysis

- ▶ The major difference between LDA and Logistic Regression is the way both techniques picks the linear decision boundary.
- ▶ Linear Discriminant Analysis models the decision boundary by making distributional assumptions about the data generating process
- ▶ Logistic Regression models the probability of a sample being member of a class given its feature values.

Method 3: Nearest Neighbor

Nearest Neighbor

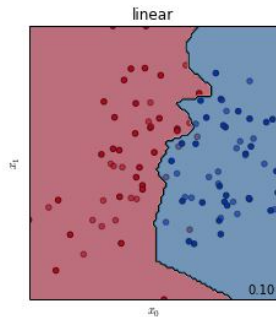
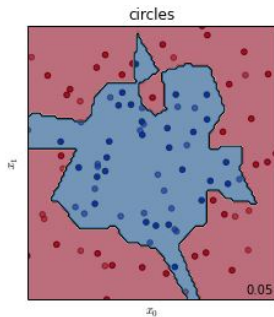
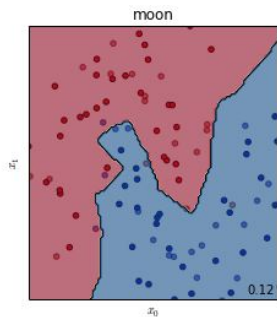
- ▶ Nearest Neighbor uses the notion of similarity to assign class labels; it is based on the smoothness assumption that points which are nearby in input space should have similar outputs.
- ▶ It does this by specifying a similarity (or distance) metric, and at prediction time it simply searches for the k most similar among the training examples to a given test example.

Method 3: Nearest Neighbor

Nearest Neighbor

- ▶ The prediction is then either a majority vote of those k training examples or a vote weighted by similarity.
- ▶ The parameter k specifies the smoothness of the decision surface.
- ▶ The decision surface of a k -nearest neighbor classifier can be illustrated by the **Voronoi tessellation** of the training data, that show you the regions of constant responses.

Method 3: Nearest Neighbor



Method 3: Nearest Neighbor

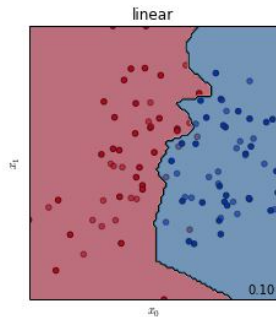
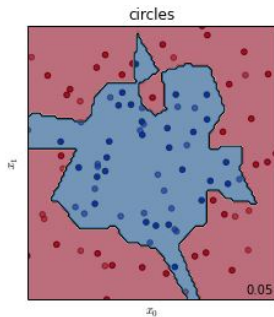
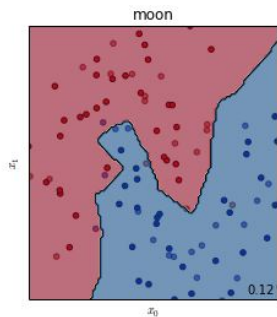
Nearest Neighbours

- ▶ Yet Nearest Neighbor differs fundamentally from the above models in that it is a so-called non-parametric technique: the number of parameters of the model can grow infinitely as the size of the training data grows.
- ▶ Furthermore, it can model non-linear decision boundaries, something that is important for the first two datasets: moons and circles.

Method 3: Nearest Neighbor

```
from sklearn.neighbors import KNeighborsClassifier  
  
est = KNeighborsClassifier(n_neighbors=1)  
plot_datasets(est)
```

Method 3: Nearest Neighbor

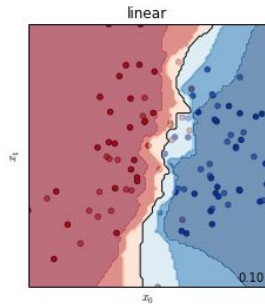
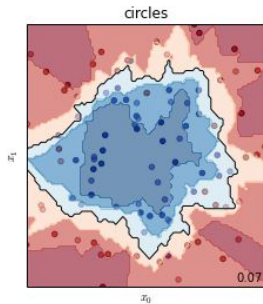
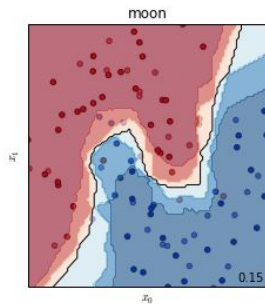


Method 3: Nearest Neighbor

- ▶ If we increase k we enforce the smoothness assumption.
- ▶ This can be seen by comparing the decision boundaries in the plots below where $k=5$ to those above where $k=1$.

```
est = KNeighborsClassifier(n_neighbors=5)
plot_datasets(est)
```

Method 3: Nearest Neighbor



Decision Trees

- ▶ Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression.
- ▶ The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

A decision tree is a simple representation for classifying examples. Decision tree learning is one of the most successful techniques for supervised classification learning[citation needed]. For this section, assume that all of the features have finite discrete domains, and there is a single target feature called the classification. Each element of the domain of the classification is called a class.

Decision Trees

- ▶ A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature.
- ▶ The arcs coming from a node labeled with a feature are labeled with each of the possible values of the feature.
- ▶ Each leaf of the tree is labeled with a class or a probability distribution over the classes.

Decision Trees

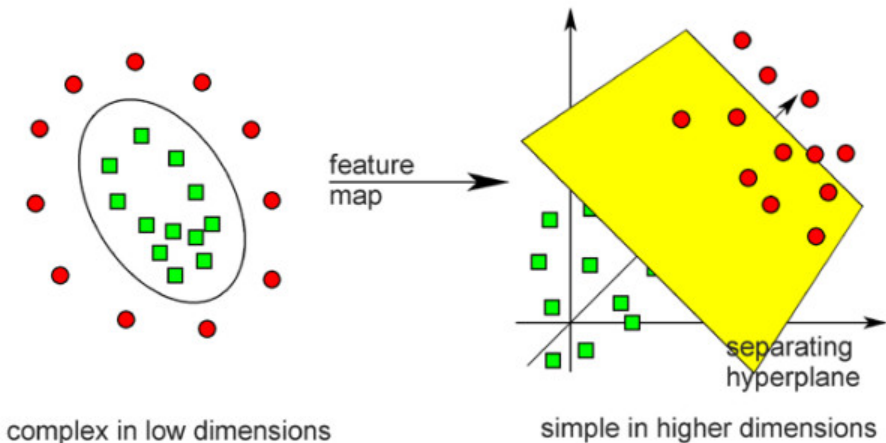
- ▶ For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of *If-Then-Else* decision rules.
- ▶ The deeper the tree, the more complex the decision rules and the fitter the model.

Decision Trees

Using the Iris dataset, we can construct a tree as follows:

```
>>> from sklearn.datasets import load_iris
>>> from sklearn import tree
>>> iris = load_iris()
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(iris.data, iris.target)
```

Separation may be easier in higher dimensions



Separation may be easier in higher dimensions

