Introduction to Machine Learning with scikit.learn

West of Ireland Data Science

scikit.learn - Machine Learning with Python

What is PyData?

PyData : Conference Mission Statement

- 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.

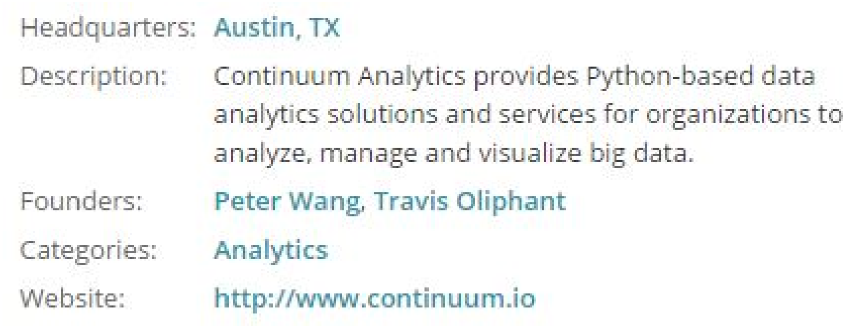
PyData : Conference Mission Statement

- 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

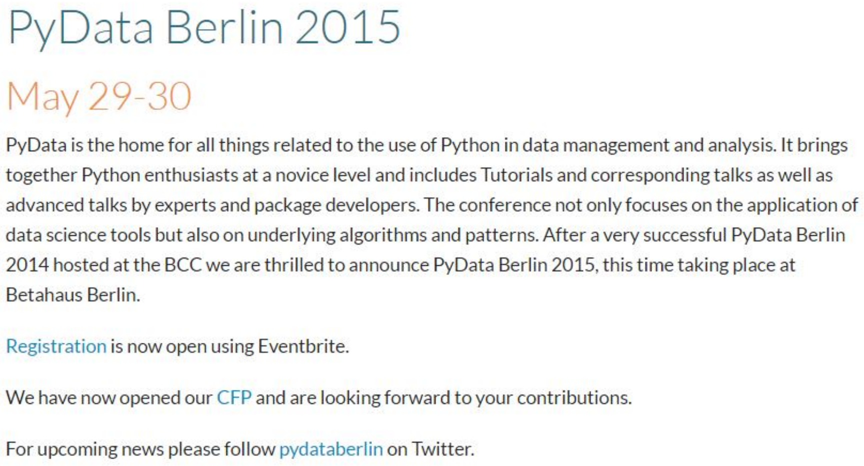
CONTINUUM
ANALYTICS



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

PyData Berlin Conference


PyData Berlin 2015

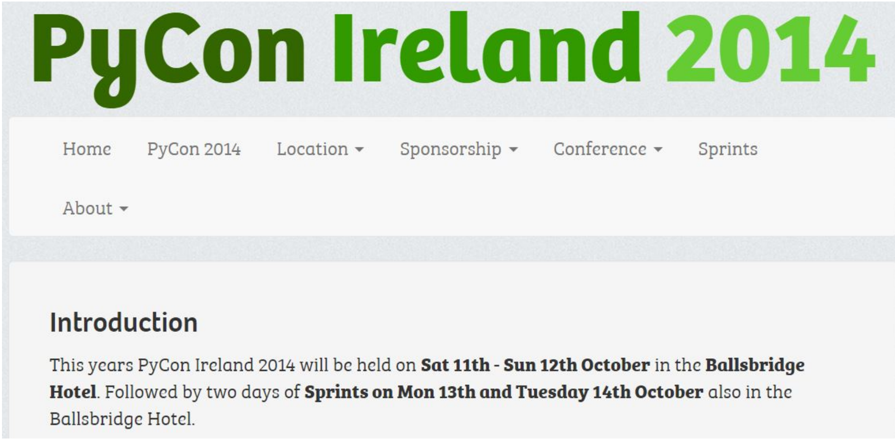
May 29-30

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.



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.

PyData London 2015

Home | Venue | About ▾ | Sponsors ▾ | CFP | Account ▾

PyData London 2015

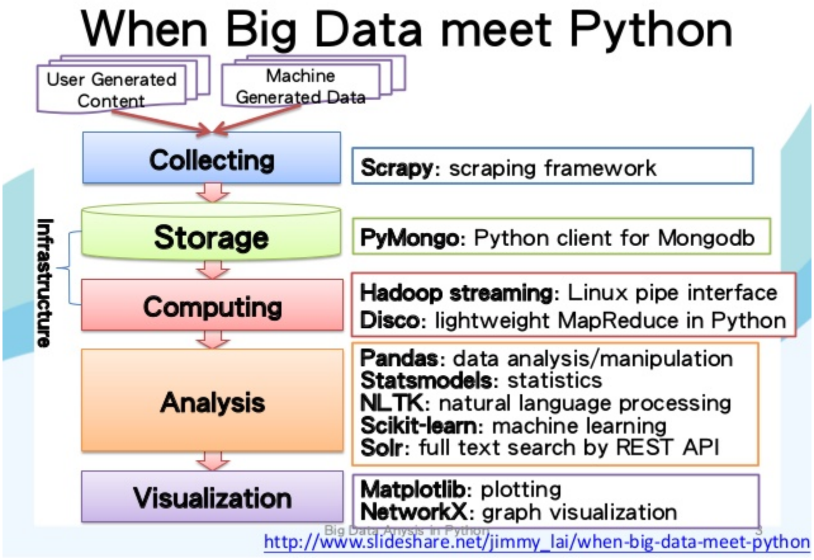
June 19-21

Hosted by Bloomberg

SPONSORS

CONTINUUM

Important Components of the Python Computing Environment



When Big Data meet Python

Scrapy: scraping framework

PyMongo: Python client for Mongodb

Hadoop streaming: Linux pipe interface
Disco: lightweight MapReduce in Python

Pandas: data analysis/manipulation
Statsmodels: statistics
NLTK: natural language processing
Scikit-learn: machine learning
Solr: full text search by REST API

Matplotlib: plotting
NetworkX: graph visualization

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

Continuum Analytics Anaconda

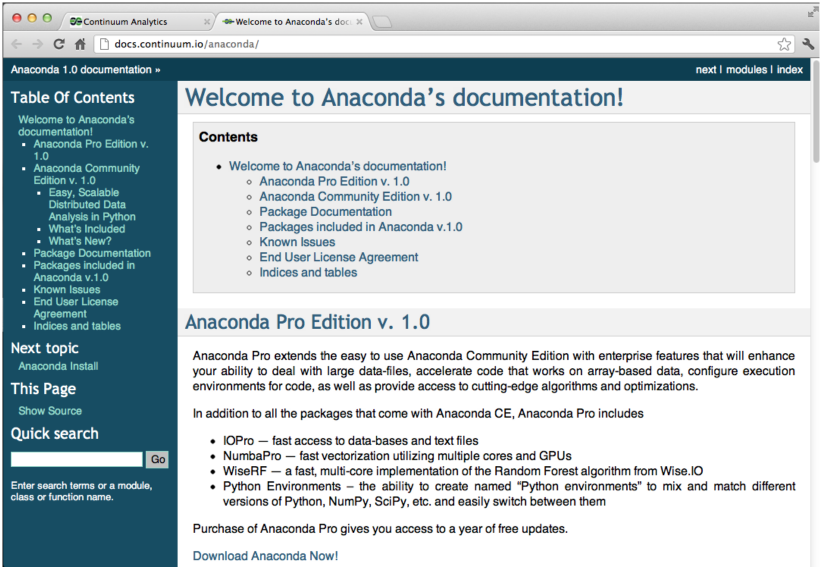

Continuum Analytics
Welcome to Anaconda's doc
docs.continuum.io/anaconda/
Anaconda 1.0 documentation »
next | modules | index

Table Of Contents

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

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!

Continuum Analytics Anaconda

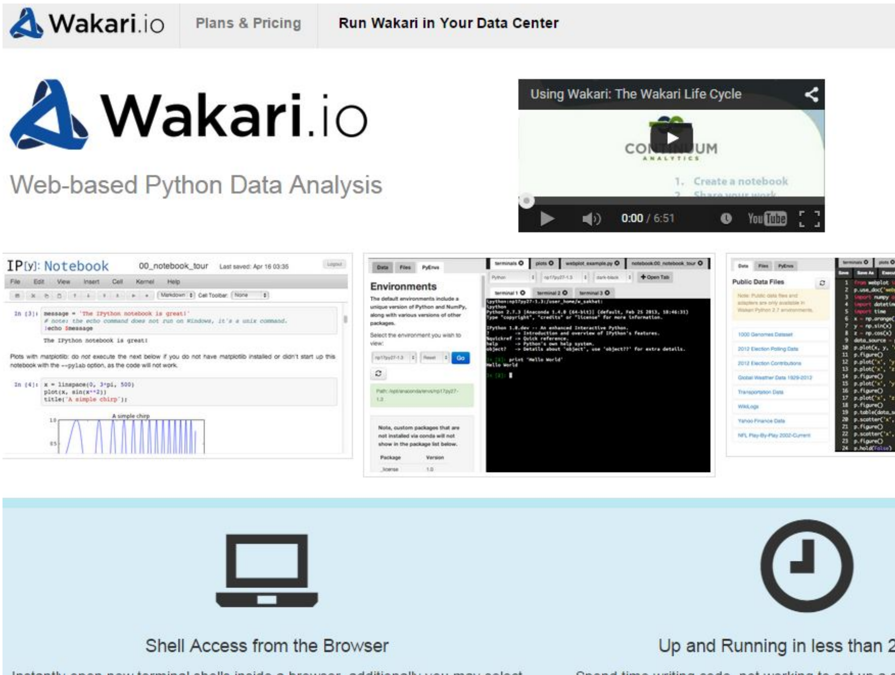
Anaconda

- 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.

Continuum Analytics Anaconda

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.



Wakari.io

Web-based Python Data Analysis

```
In [3]: message = 'The IPython notebook is great!'
        # note: the echo command does not run on Windows, it's a unix command.
        !echo $message

        The IPython notebook is great!
```

Plots with *matplotlib*: do *not* execute the next below if you do not have matplotlib installed or didn't start up this notebook with the `--pylab` option, as the code will not work.

```
In [4]: x = linspace(0, 3*pi, 500)
        plot(x, sin(x**2))
        title('A simple chirp');
```

Environments

The default environments include a unique version of Python and NumPy, along with various versions of other packages.

Select the environment you wish to view:

Path: /opt/anaconda/envs/np17py27-1.3

Note, custom packages that are not installed via conda will not show in the package list below.

Package	Version
_license	1.0

```
(python:np17py27-1.3:/user_home/w_sakhat)
%python
Python 2.7.3 |Anaconda 1.4.0 (64-bit)| (default, Feb 25 2013, 18:46:33)
Type "copyright", "credits" or "license" for more information.

IPython 1.0.dev -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: print 'Hello World'
Hello World

In [2]:
```

Public Data Files

Note: Public data files and adapters are only available in Wakari Python 2.7 environments.

- 1000 Genomes Dataset
- 2012 Election Polling Data
- 2012 Election Contributions
- Global Weather Data 1929-2012
- Transportation Data
- WikiLogs
- Yahoo Finance Data
- NFL Play-By-Play 2002-Current

```
from webplot
p.use_doc('
import numpy
import datetim
import time
x = np.arange(
y = np.sin(x)
z = np.cos(x)
data_source =
p.plot(x, y,
p.figure()
p.plot('x', 'y
p.plot('x', 'z
p.figure()
p.plot('x', 'y
p.figure()
p.plot('x', 'z
p.figure()
p.table(data_s
p.scatter('x',
p.figure()
p.scatter('x',
p.figure()
p.hold(False)
```

Shell Access from the Browser

Up and Running in less than 2

Wakari

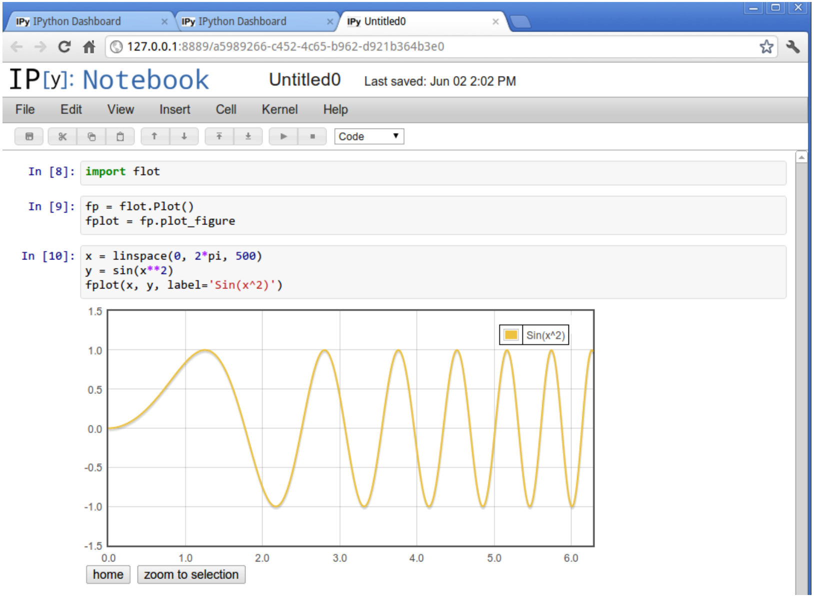
Wakari is a collaborative data analytics platform that includes tools to explore data, develop analytics scripts, collaborate with IPython notebooks, visualize, and share data analysis and findings. Save time and money by getting right down to business analyzing data with the fully-configured Wakari. 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

- 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



```
In [8]: import flot

In [9]: fp = flot.Plot()
        fplot = fp.plot_figure

In [10]: x = linspace(0, 2*pi, 500)
         y = sin(x**2)
         fplot(x, y, label='Sin(x^2)')
```

Ipython Notebook / Jupyter

Evolved from the IPython Project

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

IPython

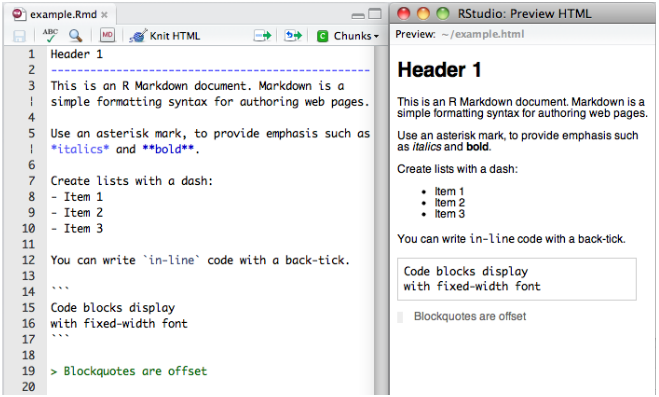
- Interactive Python shell
- Python kernel for Jupyter
- Interactive Parallel Python

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).



```
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
```

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

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.

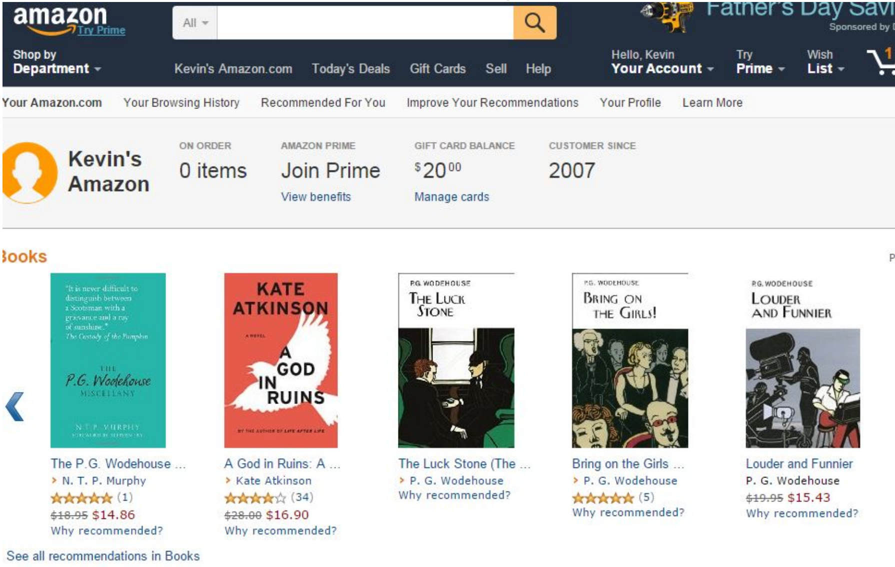
What is Machine Learning

- 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



amazon Try Prime

All

Father's Day Savi

Sponsored by

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

Hello, Kevin Your Account · Try Prime · Wish List · 1

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

Kevin's Amazon

ON ORDER	AMAZON PRIME	GIFT CARD BALANCE	CUSTOMER SINCE
0 items	Join Prime	$20.00	2007
	View benefits	Manage cards	

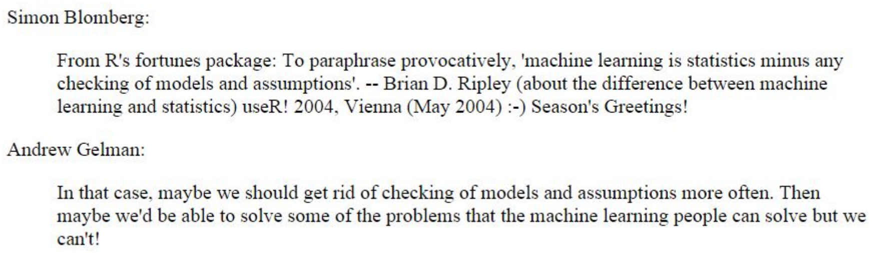
Books

Title	Author	Rating	Price	
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.


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

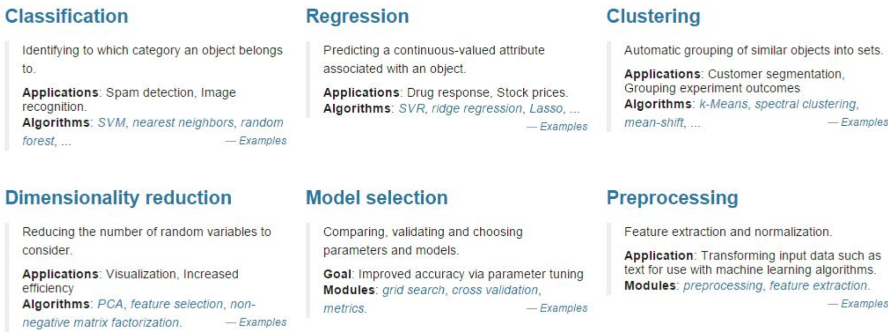
Data Science Retreat (Berlin)

> *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

- 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

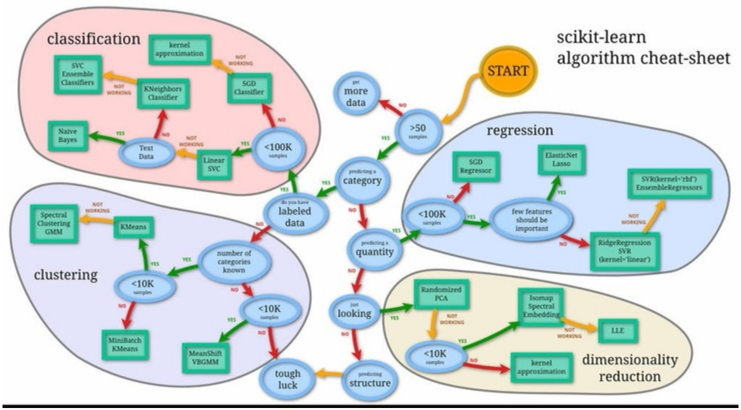
Comparing, validating and choosing parameters and models.

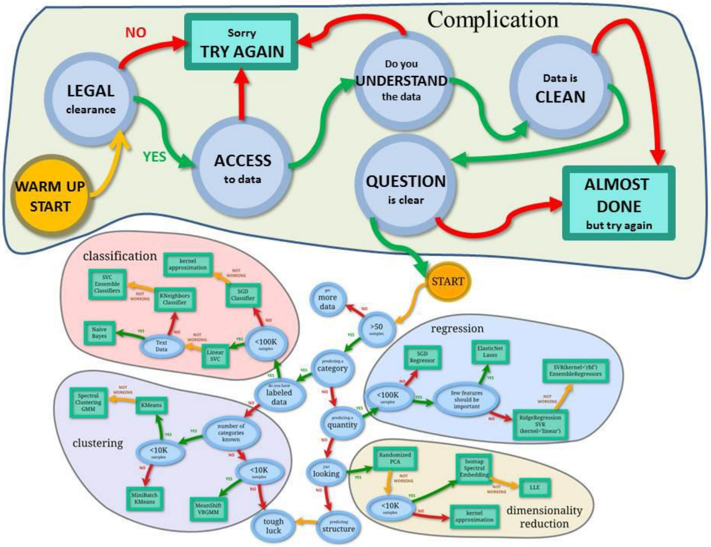
Goal: Improved accuracy via parameter tuning
Modules: *grid search, cross validation, metrics.* — *Examples*

Preprocessing

Feature extraction and normalization.

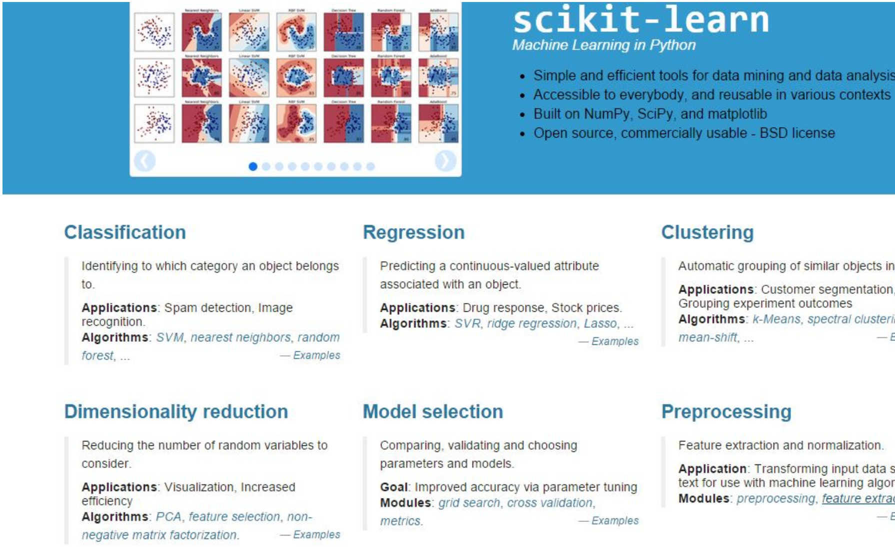
Application: Transforming input data such as text for use with machine learning algorithms.
Modules: *preprocessing, feature extraction.* — *Examples*


scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
<100K samples
SGD Classifier
kernel approximation
Linear SVC
Text Data
KNeighbors Classifier
SVC Ensemble Classifiers
Naive Bayes
clustering
number of categories known
<10K samples
KMeans
Spectral Clustering GMM
MiniBatch KMeans
<10K samples
MeanShift VBGMM
regression
<100K samples
SGD Regressor
few features should be important
ElasticNet Lasso
RidgeRegression SVR (kernel='linear')
SVR(kernel='rbf') EnsembleRegressors
dimensionality reduction
Randomized PCA
<10K samples
Isomap Spectral Embedding
LLE
kernel approximation
YES
NO
NOT WORKING



Complication
WARM UP START
LEGAL clearance
NO
YES
Sorry TRY AGAIN
ACCESS to data
Do you UNDERSTAND the data
Data is CLEAN
QUESTION is clear
ALMOST DONE but try again
START
get more data
>50 samples
NO
YES
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
<100K samples
SGD Classifier
kernel approximation
Linear SVC
Text Data
Naive Bayes
KNeighbors Classifier
SVC Ensemble Classifiers
NOT WORKING
clustering
number of categories known
<10K samples
KMeans
Spectral Clustering GMM
MiniBatch KMeans
MeanShift VBGMM
regression
SGD Regressor
few features should be important
ElasticNet Lasso
RidgeRegression SVR (kernel='linear')
SVR(kernel='rbf') EnsembleRegressors
dimensionality reduction
Randomized PCA
Isomap Spectral Embedding
LLE
kernel approximation

Machine Learning with Python



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 clusteri* *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 algor
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.

Classification with scikit-learn

- In this segment, we look into the problem of classification, a situation in which a response is a categorical variable.
- We will introduces a number of classification techniques, and convey their corresponding strengths and weaknesses by visually inspecting the decision boundaries for each model.
- Here we will use **scikit-learn**, an easy-to-use, general-purpose toolbox for machine learning in Python

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.

Unspervised Learning

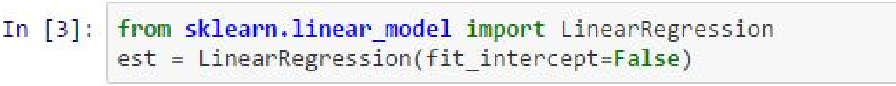
- Unsupervised machine learning, on the other hand, refers to the problem of finding interesting patterns or structure in the data; it comprises techniques such as clustering and dimensionality reduction.
- In addition to statistical learning techniques, scikit-learn provides utilities for common tasks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

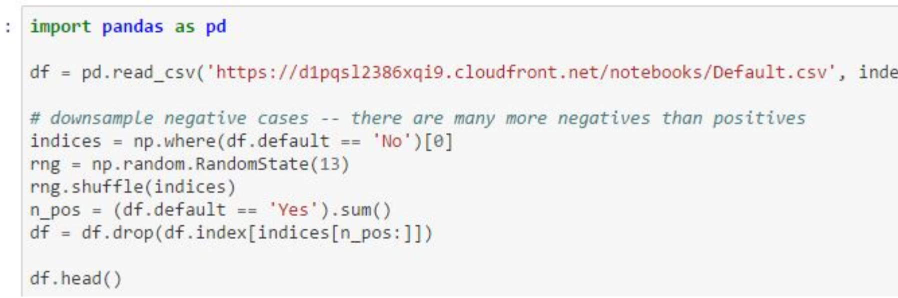
Binary Classsification Problems

- 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 its 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
- he dataset is provided by James et al., **Introduction to Statistical Learning**.

Credit Card Fraud



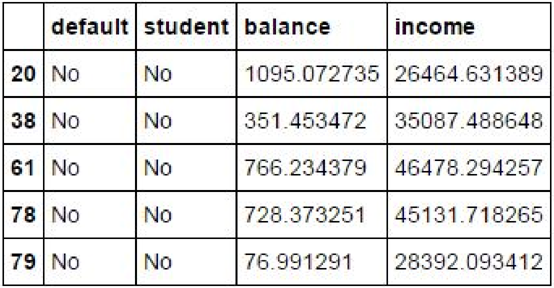
```
import pandas as pd

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

# 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()
```

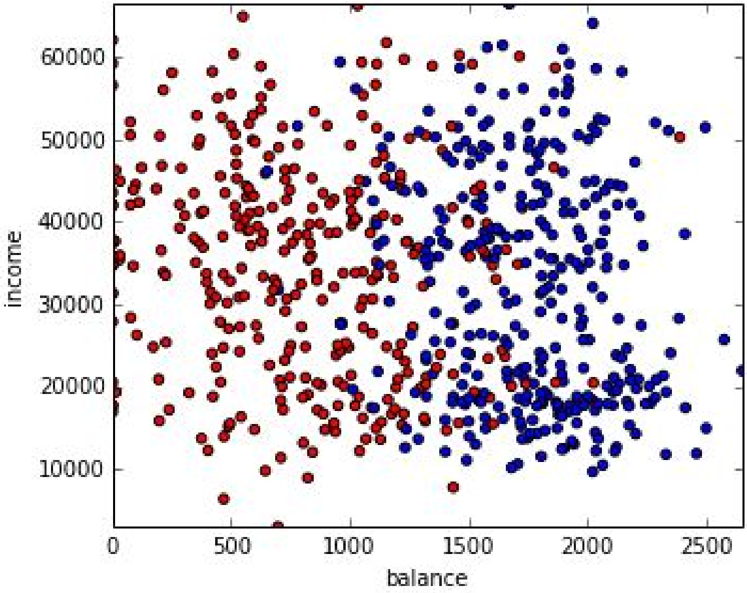
Credit Card Fraud

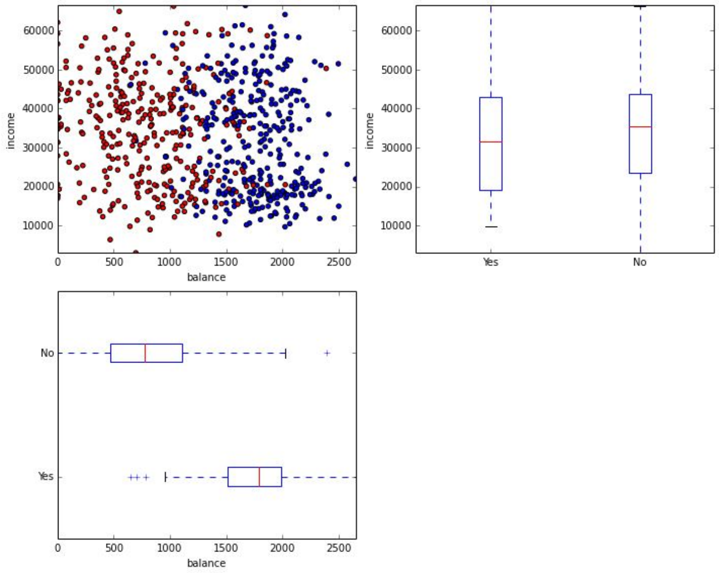


	default	student	balance	income
20	No	No	1095.072735	26464.631389
38	No	No	351.453472	35087.488648
61	No	No	766.234379	46478.294257
78	No	No	728.373251	45131.718265
79	No	No	76.991291	28392.093412

Credit Card Fraud

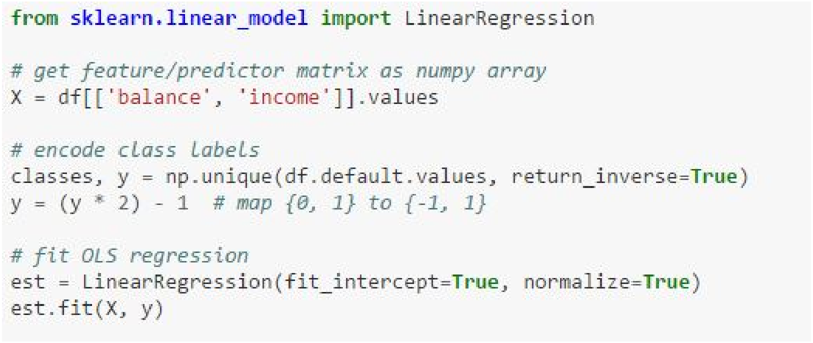
- On the left you can see a scatter plot where fraudulent cases are red dots and non-fraudulent cases are blue dots.
- A good separation seems to be a vertical line at around a balance of 1400 as indicated by the boxplots on the next slide.


income
balance
60000
50000
40000
30000
20000
10000
0
500
1000
1500
2000
2500


income
balance
Yes
No
0
500
1000
1500
2000
2500
10000
20000
30000
40000
50000
60000

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.

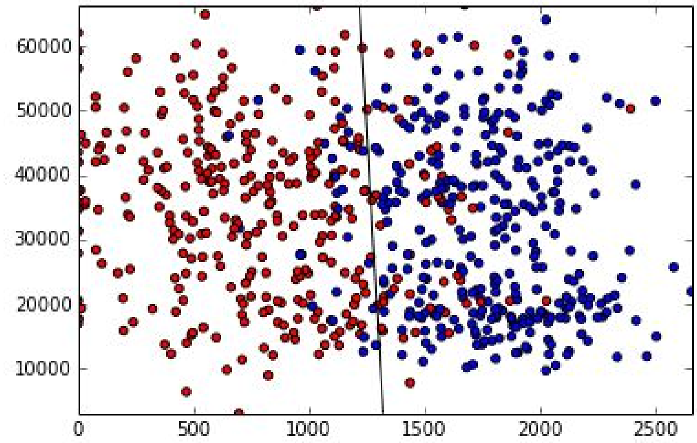


```python
from sklearn.linear_model import LinearRegression

# 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)
```



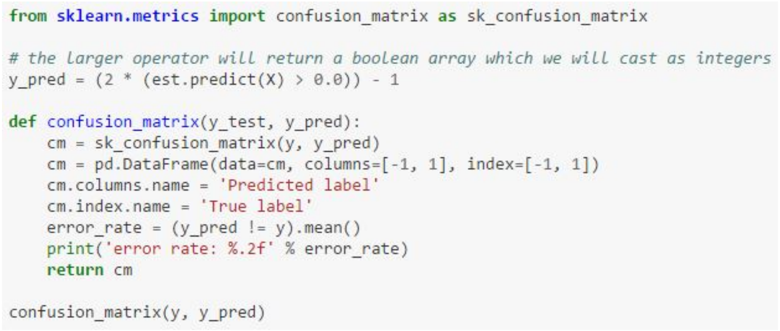
60000
50000
40000
30000
20000
10000
0
500
1000
1500
2000
2500

- 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



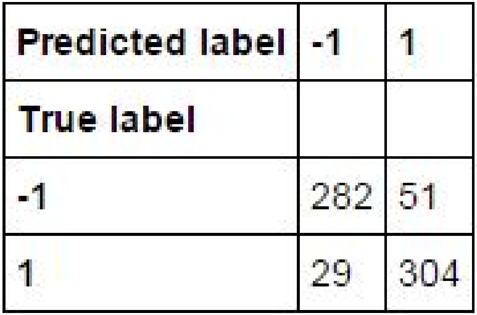
```python
from sklearn.metrics import confusion_matrix as sk_confusion_matrix

# 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**.

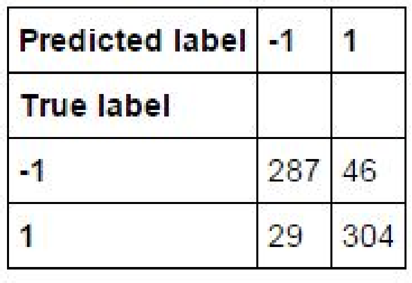
```python
from sklearn.cross_validation import train_test_split

# 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

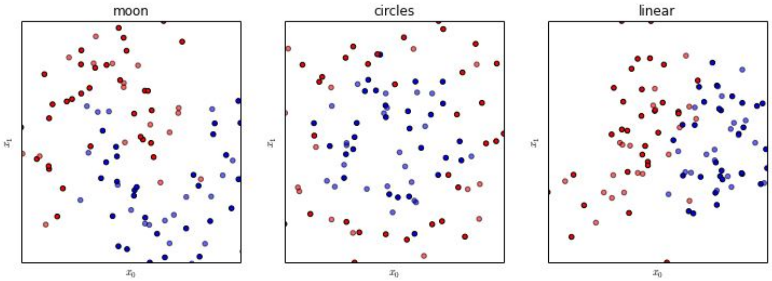
- Different classification techniques can often be compared using the type of decision surface they can learn.
- The decision surfaces describe for what values of the predictors the model changes its predictions and it can take several different shapes: piece-wise constant, linear, quadratic, vornoi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

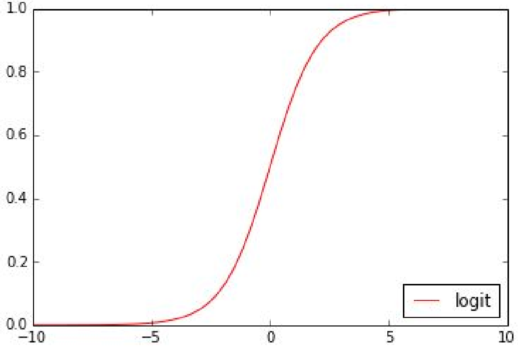
Method 1: Logistic Regression

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

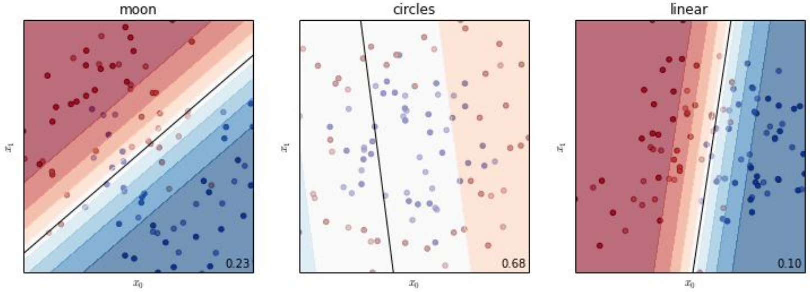
- Logistic regression is available in scikit-learn via the class `sklearn.linear_model.LogisticRegression`.
- Lets see how Logistic Regression does on our three toy datasets.

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.

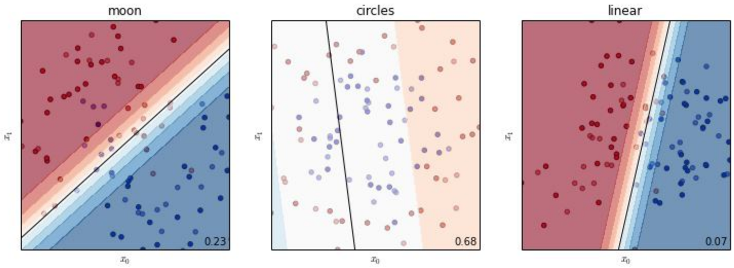
Method 2: Linear Discriminant Analysis

```
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

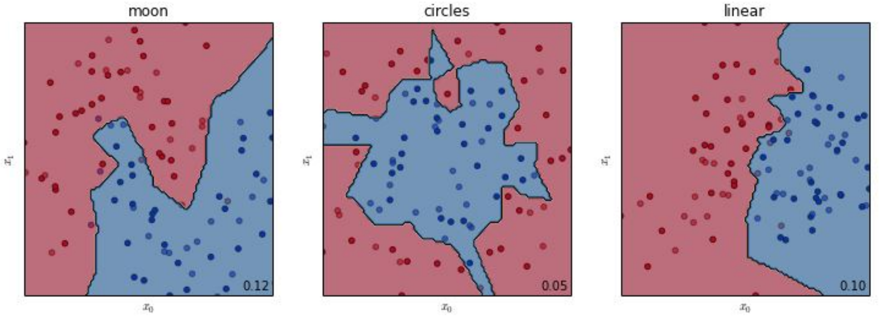
- Nearest Neighbor uses the notion of similarity to assign class labels; it is based on the smoothness assumption that points which are nearby in input space should have similar outputs.
- It does this by specifying a similarity (or distance) metric, and at prediction time it simply searches for the 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 tesselation** of the training data, that show you the regions of constant respones.

Method 3: Nearest Neighbor



Method 3: Nearest Neighbor

Nearest Neighbours

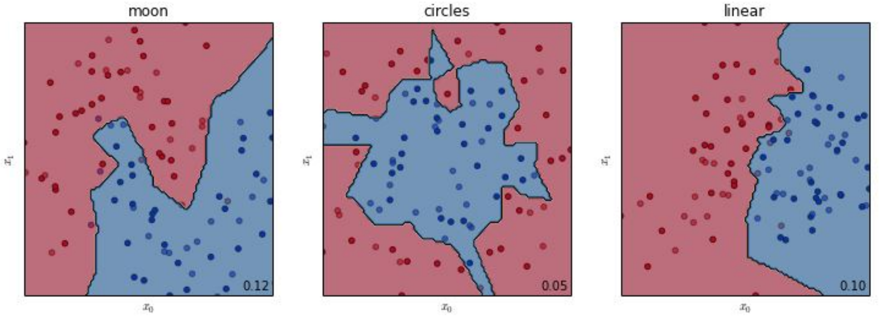
- Yet Nearest Neighbor differs fundamentally from the above models in that it is a so-called non-parametric technique: the number of parameters of the model can grow infinitely as the size of the training data grows.
- Furthermore, it can model non-linear decision boundaries, something that is important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

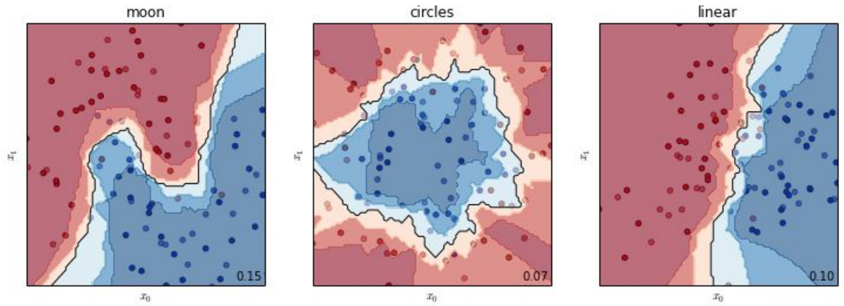


Method 3: Nearest Neighbor

- 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

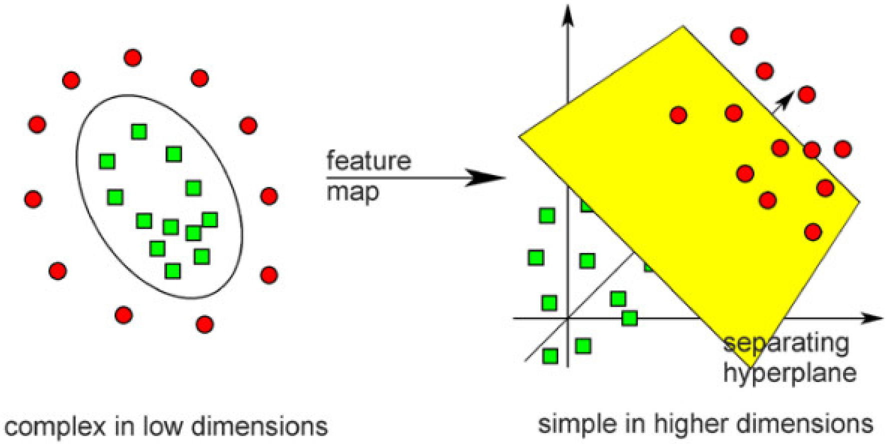
- For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of *If-Then-Else* decision rules.
- The deeper the tree, the more complex the decision rules and the fitter the model.

Decision Trees

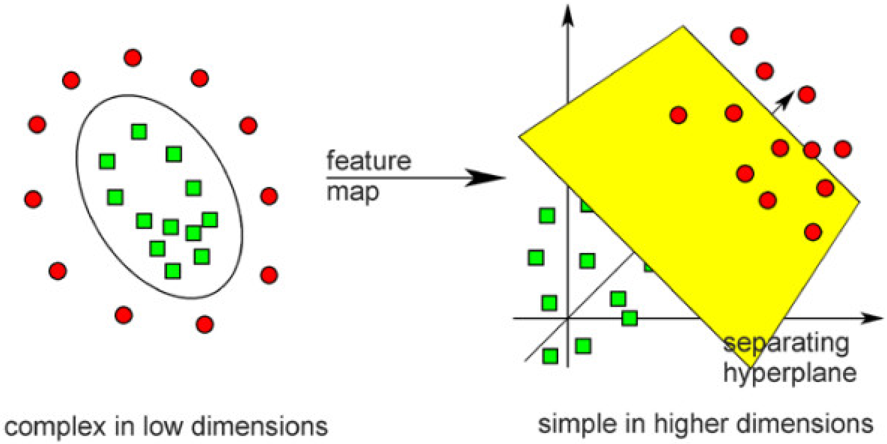
Using the Iris dataset, we can construct a tree as follows:

```
>>> 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



scikit.learn - Machine Learning with Python

```
import scikit.learn
```