How to use scikit-learn to solve machine learning problems

AutoML Hackathon

April 2015

Olivier Grisel

@ogrisel

Datageek, contributor to scikit-learn, works with Python / Java / Clojure / Pig, interested in Machine Learning, NLProc, {Big|Linked|Open} Data and braaains!

Paris, France · http://github.com/ogrisel

Outline

- Machine Learning refresher
- scikit-learn
- Demo: interactive predictive modeling on Census Data with IPython notebook / pandas / scikit-learn
- Combining models with Pipeline and parameter search

Predictive modeling ~= machine learning

- Make predictions of outcome on new data
- Extract the structure of historical data
- Statistical tools to summarize the training data into a executable predictive model
- Alternative to hard-coded rules written by experts

type (category)	# rooms (int)	surface (float m2)	public trans (boolean)
Apartment	3	50	TRUE
House	5	254	FALSE
Duplex	4	68	TRUE
Apartment	2	32	TRUE

type (category)	# rooms (int)	surface (float m2)	public trans (boolean)	sold (float k€)
Apartment	3	50	TRUE	450
House	5	254	FALSE	430
Duplex	4	68	TRUE	712
Apartment	2	32	TRUE	234

features

target

type (category)	# rooms (int)	surface (float m2)	public trans (boolean)	sold (float k€)
Apartment	3	50	TRUE	450
House	5	254	FALSE	430
Duplex	4	68	TRUE	712
Apartment	2	32	TRUE	234

samples (train)

features

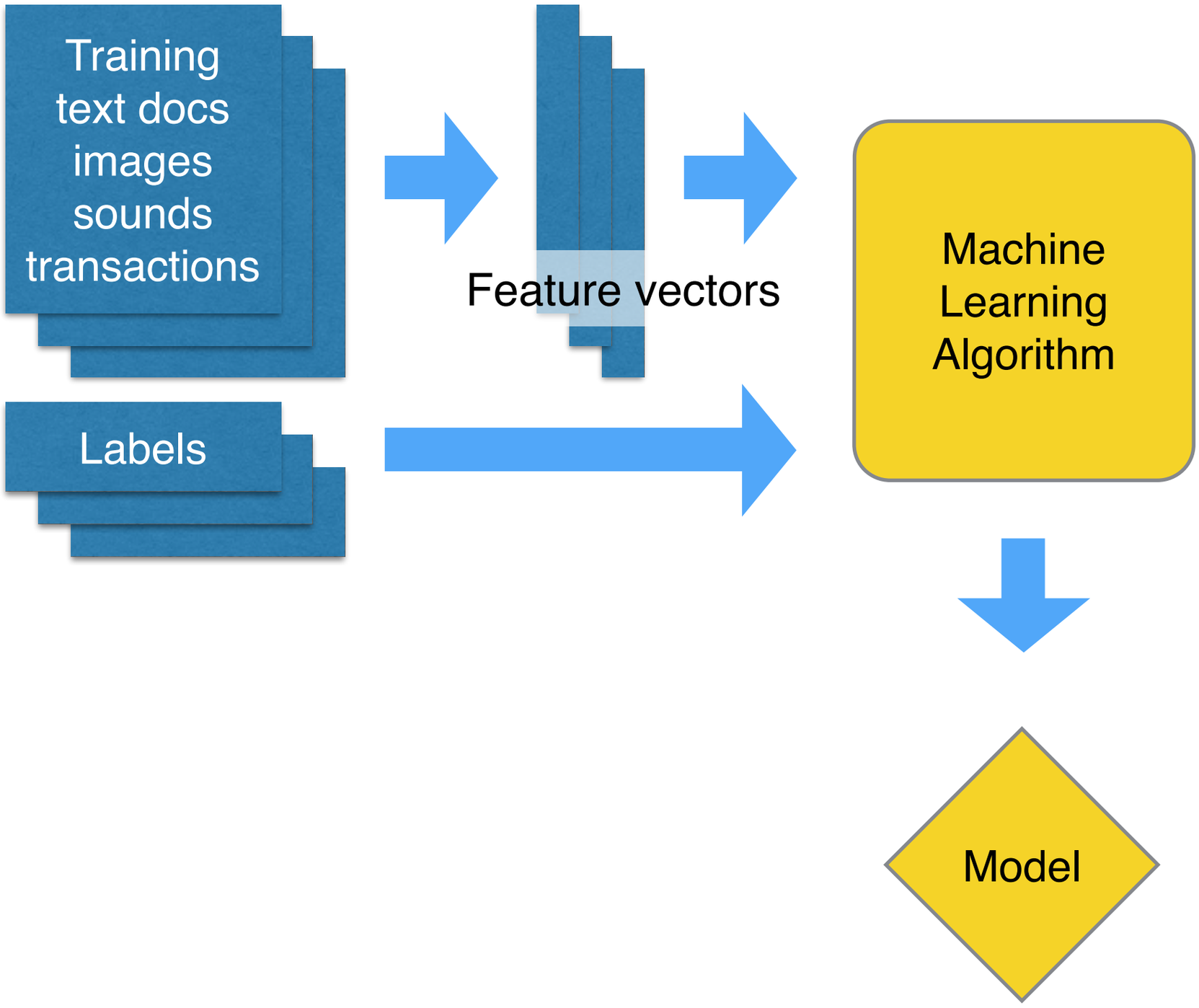
target

samples (train)

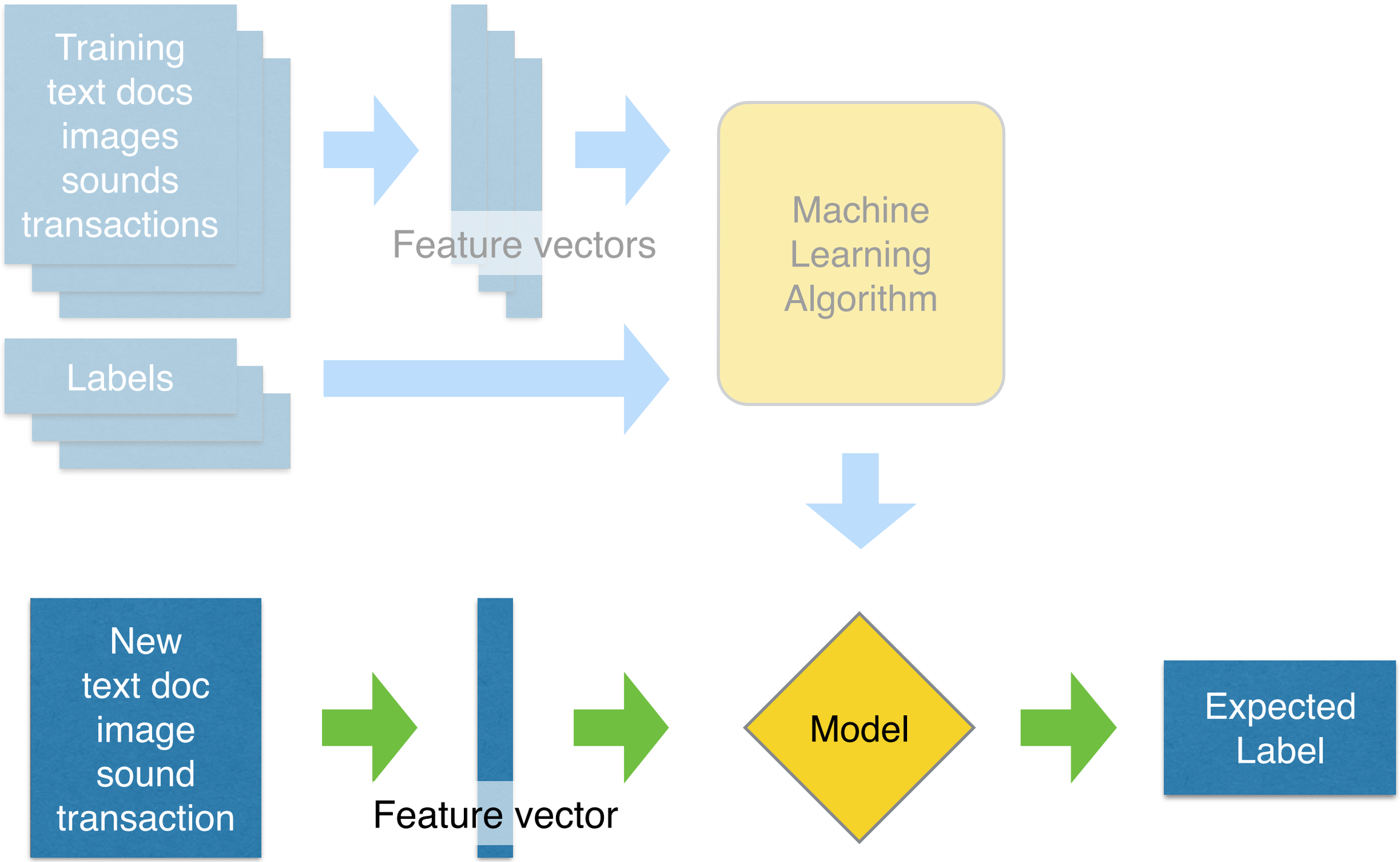
type (category)	# rooms (int)	surface (float m2)	public trans (boolean)	sold (float k€)
Apartment	3	50	TRUE	450
House	5	254	FALSE	430
Duplex	4	68	TRUE	712
Apartment	2	32	TRUE	234

samples (test)

Apartment	2	33	TRUE	?
House	4	210	TRUE	?



Predictive Modeling Data Flow



Predictive Modeling Data Flow

Predictive modeling in the wild

Virality and readers engagement

Fraud detection

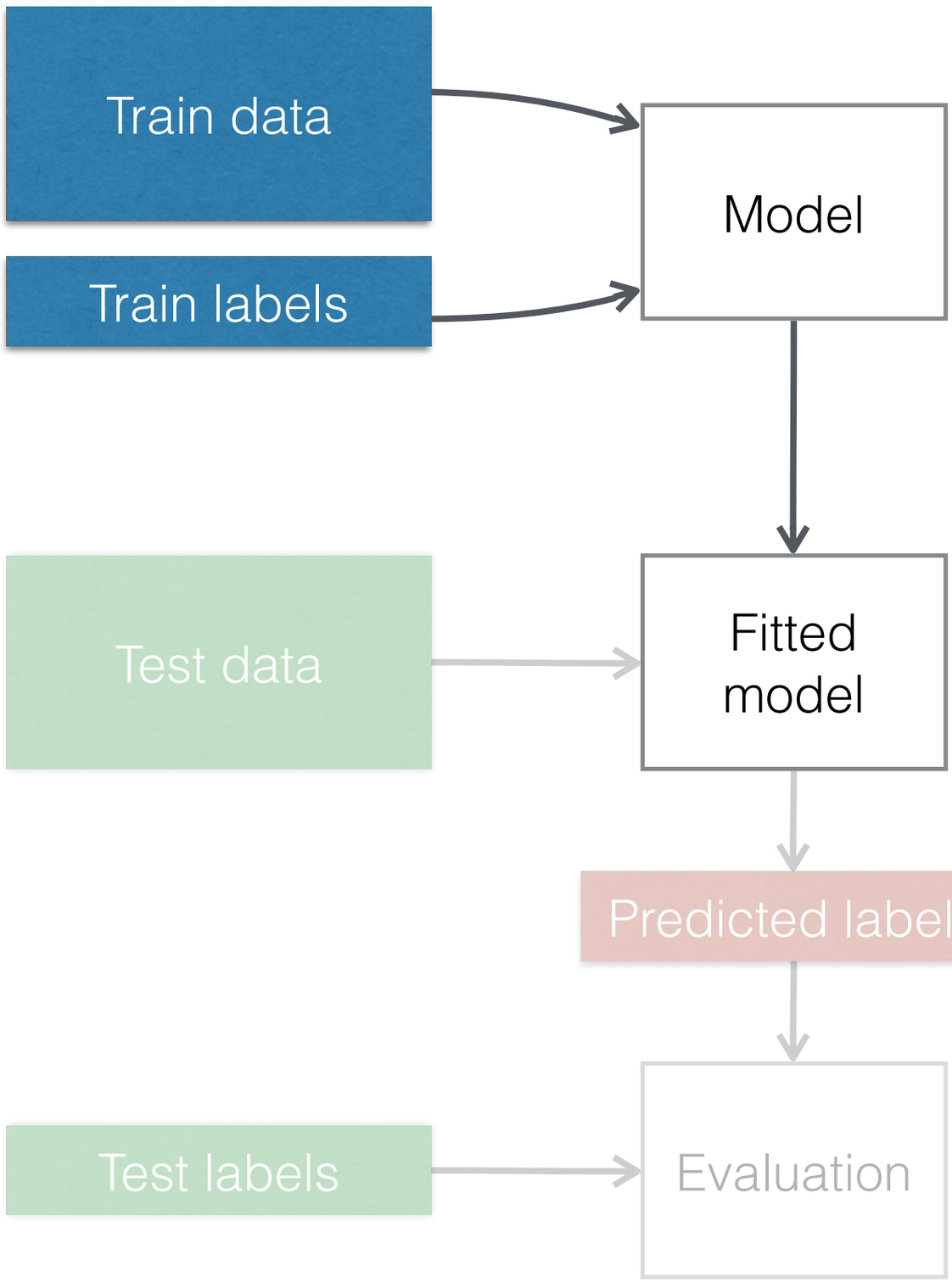
Personalized radios

Inventory forecasting & trends detection

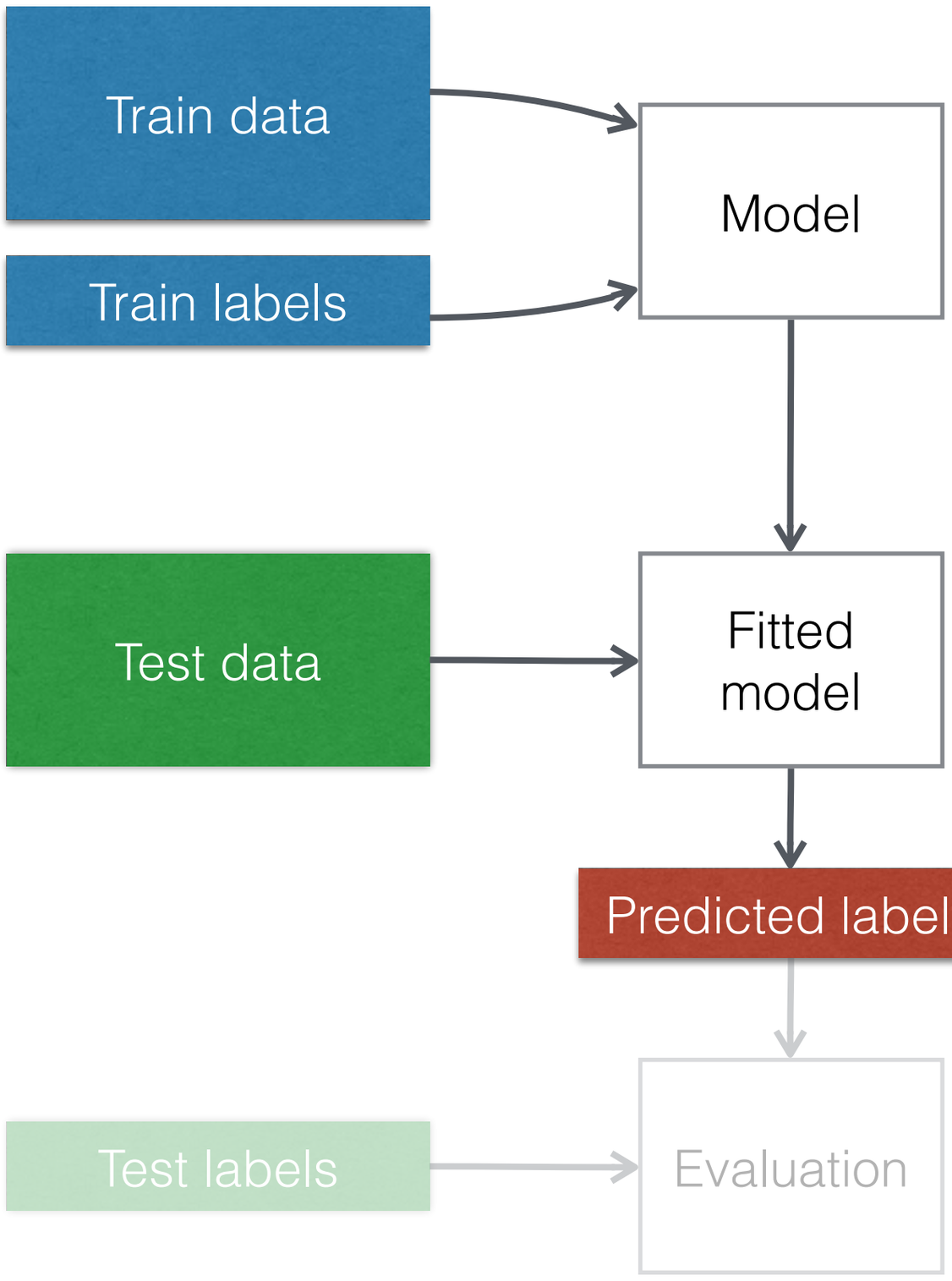
Predictive maintenance

Personality matching

- Library of Machine Learning algorithms
- Focus on established methods (e.g. ESL-II)
- Open Source (BSD)
- Simple **`fit`** / **`predict`** / **`transform`** API
- Python / NumPy / SciPy / Cython
- Model Assessment, Selection & Ensembles

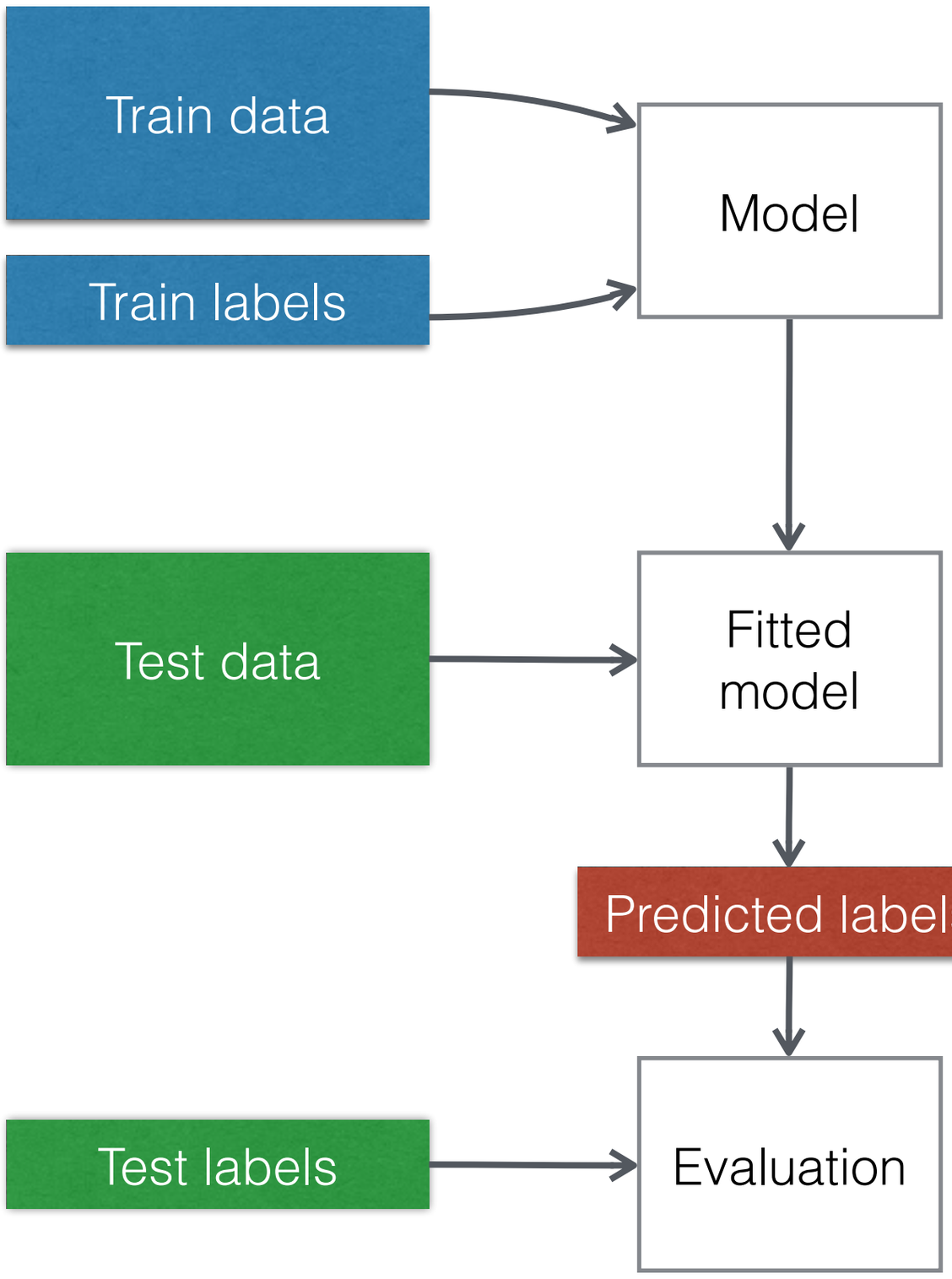



```
model = ModelClass(**hyperparams)
model.fit(X_train, y_train)
```




```
model = ModelClass(**hyperparams)
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
```




```
model = ModelClass(**hyperparams)
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
```

```
accuracy_score(y_test, y_pred)
```

Support Vector Machine

```
from sklearn.svm import SVC

model = SVC(kernel="rbf", C=1.0, gamma=1e-4)
model.fit(X_train, y_train)


y_predicted = model.predict(X_test)

from sklearn.metrics import f1_score
f1_score(y_test, y_predicted)
```

Linear Classifier

```
from sklearn.linear_model import SGDClassifier

model = SGDClassifier(alpha=1e-4,
                      penalty="elasticnet")
model.fit(X_train, y_train)


y_predicted = model.predict(X_test)

from sklearn.metrics import f1_score
f1_score(y_test, y_predicted)
```

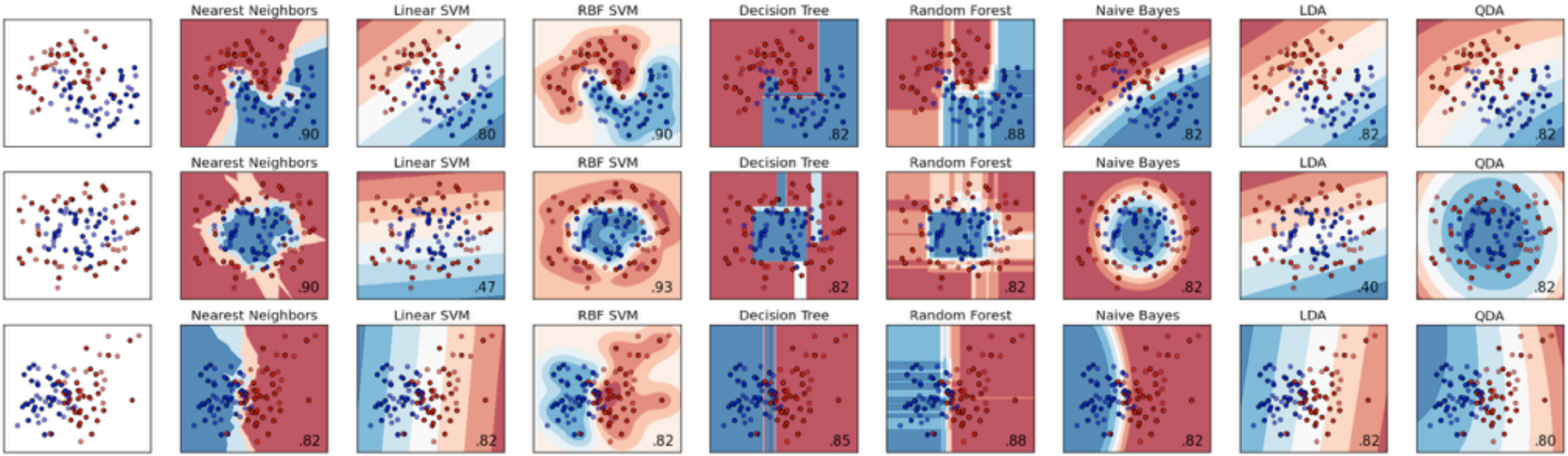
Random Forests

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=200)
model.fit(X_train, y_train)


y_predicted = model.predict(X_test)

from sklearn.metrics import f1_score
f1_score(y_test, y_predicted)
```


Nearest Neighbors
Linear SVM
RBF SVM
Decision Tree
Random Forest
Naive Bayes
LDA
QDA
.90
.80
.90
.82
.88
.82
.82
.82
Nearest Neighbors
Linear SVM
RBF SVM
Decision Tree
Random Forest
Naive Bayes
LDA
QDA
.90
.47
.93
.82
.82
.82
.40
.82
Nearest Neighbors
Linear SVM
RBF SVM
Decision Tree
Random Forest
Naive Bayes
LDA
QDA
.82
.82
.82
.85
.88
.82
.82
.80

scikit-learn: machine learning in Python — scikit-learn 0.14 documentation

scikit-learn: machine learning i...

scikit-learn.org/stable/

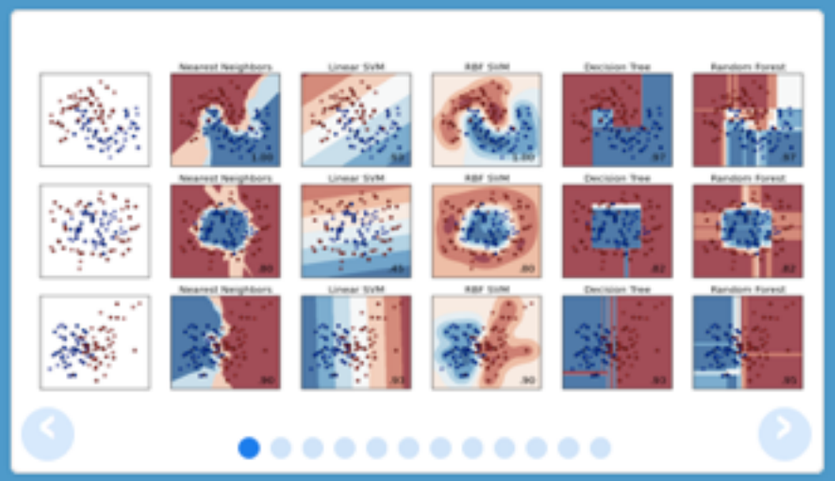
Google

scikit learn

Home | Installation | Documentation | Examples

Google™ Custom Search | Search

Fork me on GitHub



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 set of categories a new observation belong to.

Applications: Spam detection, Image recognition.
Algorithms: *SVM*, *nearest neighbors*, *random forest*, ... — *Examples*

Regression

Predicting a continuous value for a new example.

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*, *Isomap*, *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*

Demo time!

http://nbviewer.ipython.org/github/ogrisel/notebooks/blob/master/sklearn_demos/Income%20classification.ipynb

https://github.com/ogrisel/notebooks

Combining Models

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import RandomizedPCA
from sklearn.svm import SVC

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

pca = RandomizedPCA(n_components=10)
X_train_pca = pca.fit_transform(X_train_scaled)

svm = SVC(C=0.1, gamma=1e-3)
svm.fit(X_train_pca, y_train)
```

Pipeline

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import RandomizedPCA
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
```

```
pipeline = make_pipeline(
    StandardScaler(),
    RandomizedPCA(n_components=10),
    SVC(C=0.1, gamma=1e-3),
)

pipeline.fit(X_train, y_train)
```

Scoring manually stacked models

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

pca = RandomizedPCA(n_components=10)
X_train_pca = pca.fit_transform(X_train_scaled)

svm = SVC(C=0.1, gamma=1e-3)
svm.fit(X_train_pca, y_train)

X_test_scaled = scaler.transform(X_test)
X_test_pca = pca.transform(X_test_scaled)
y_pred = svm.predict(X_test_pca)
accuracy_score(y_test, y_pred)
```

Scoring a pipeline

```
pipeline = make_pipeline(
    RandomizedPCA(n_components=10),
    SVC(C=0.1, gamma=1e-3),
)
pipeline.fit(X_train, y_train)


y_pred = pipeline.predict(X_test)
accuracy_score(y_test, y_pred)
```

Parameter search

```
import numpy as np
from sklearn.grid_search import RandomizedSearchCV

params = {
    'randomizedpca__n_components': [5, 10, 20],
    'svc__C': np.logspace(-3, 3, 7),
    'svc__gamma': np.logspace(-6, 0, 7),
}
search = RandomizedSearchCV(pipeline, params,
                            n_iter=30, cv=5)
search.fit(X_train, y_train)
# search.best_params_, search.grid_scores_
```

Thank you!

- http://scikit-learn.org

- https://github.com/scikit-learn/scikit-learn

@ogrisel