Python第三方库

outline

- 数值计算 numpy
- 数据处理分析 pandas
- 可视化 matplotlib/seaborn
- 机器学习 Sklearn / keras
- 交互 pygame
- 网络 Selenium etc…

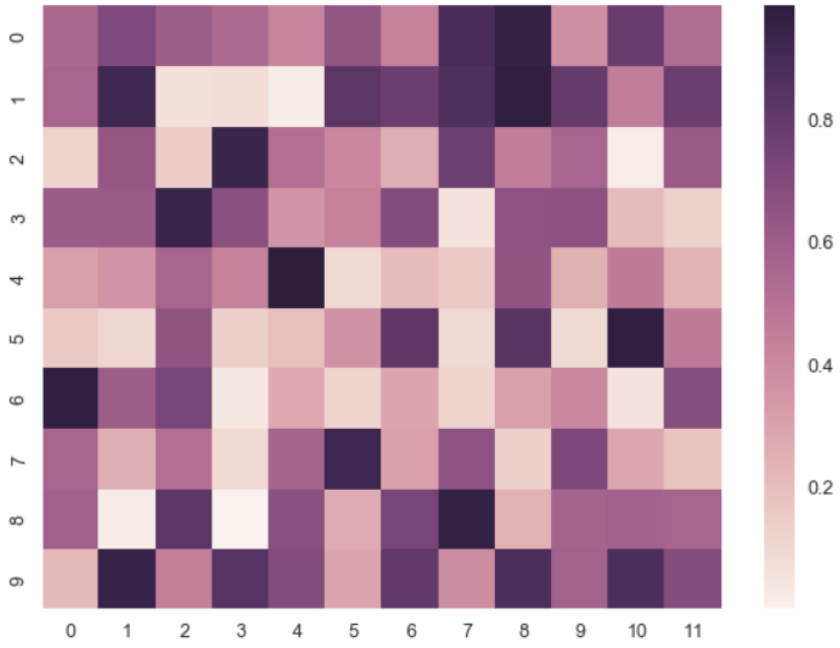
(今天讲红色字)

硬广告：python爬虫项目班开课了。。

seaborn

Plot a heatmap for a numpy array:

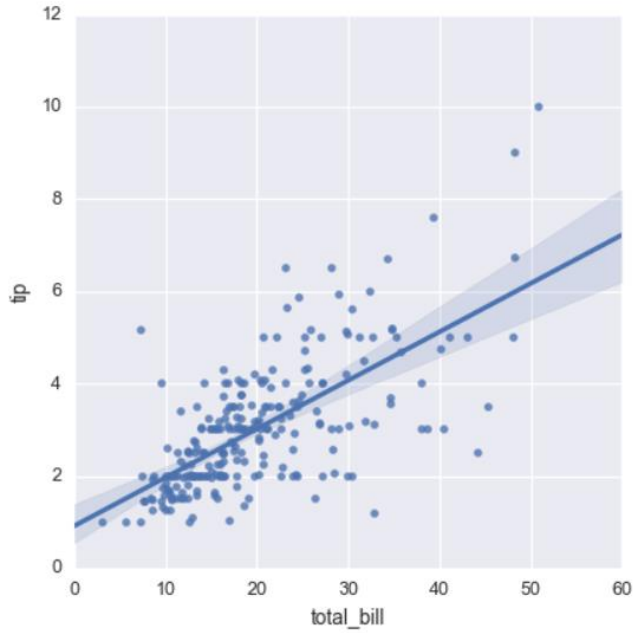
```
>>> import numpy as np; np.random.seed(0)
>>> import seaborn as sns; sns.set()
>>> uniform_data = np.random.rand(10, 12)
>>> ax = sns.heatmap(uniform_data)
```



julyedu.com
Julyedu.com

seaborn

```
>>> import seaborn as sns; sns.set(color_codes=True)
>>> tips = sns.load_dataset("tips")
>>> g = sns.lmplot(x="total_bill", y="tip", data=tips)
```



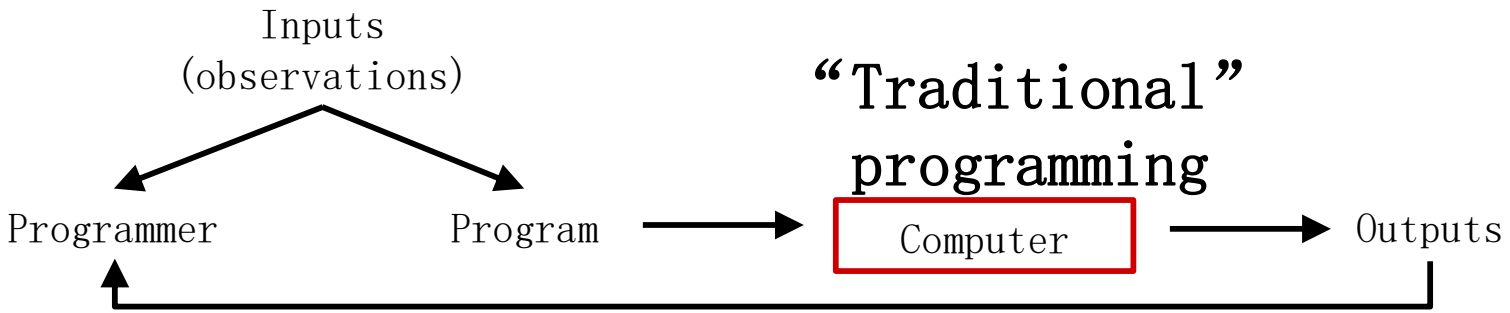
julyedu.com
Julyedu.com

第三方库的安装

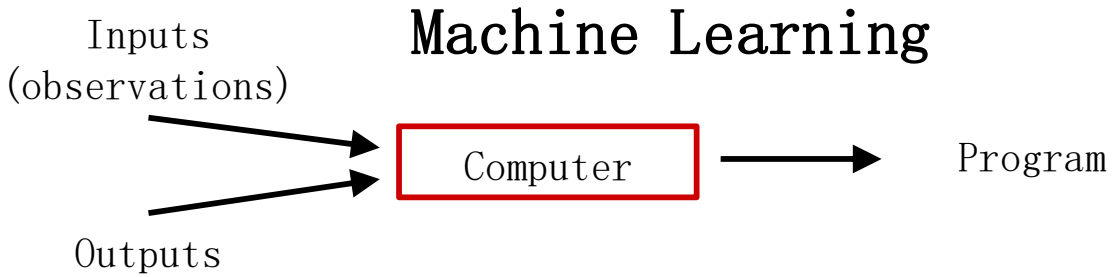
- Pip install …..
- Use anaconda, if possible

julyedu.com Julyedu.com

What is Machine Learning?



Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.
-- Arthur Samuel (1959)



Examples of Machine Learning

https://flic.kr/p/5BLW6G [CC BY 2.0]

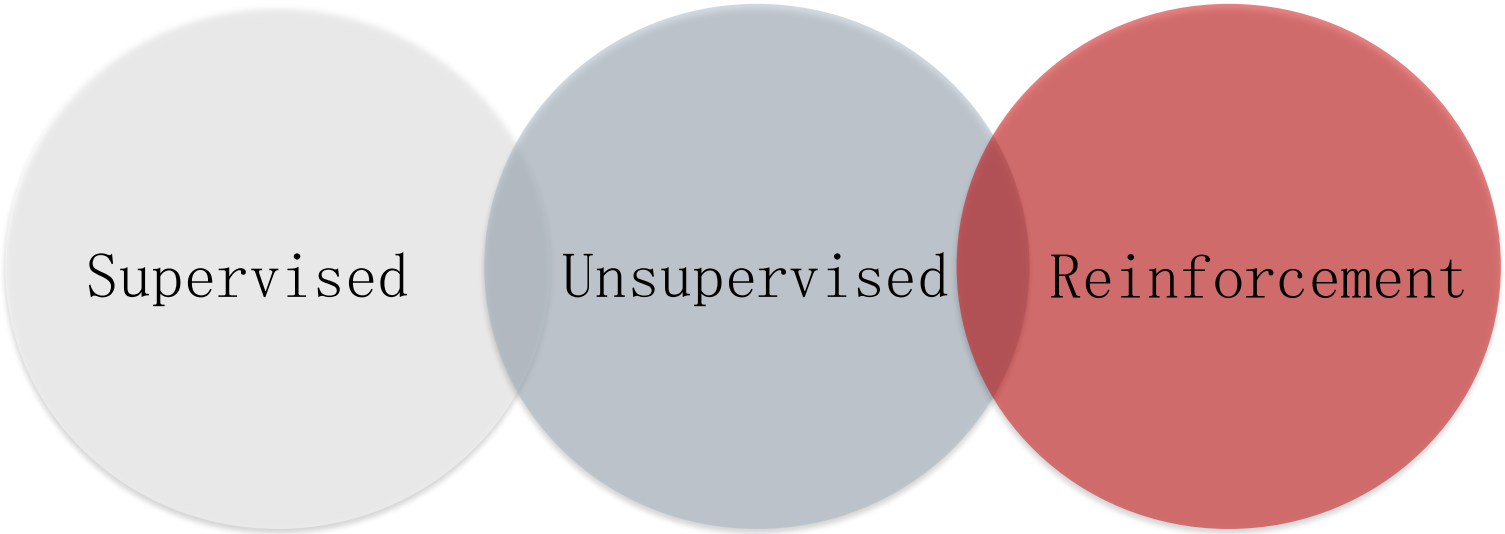
http://commons.wikimedia.org/wiki/File:American_book_company_1916._letter_envelope-2.JPG#filelinks [public domain]

http://commons.wikimedia.org/wiki/File:Netflix_logo.svg [public domain]

By Steve Jurvetson [CC BY 2.0]

And many, many more …

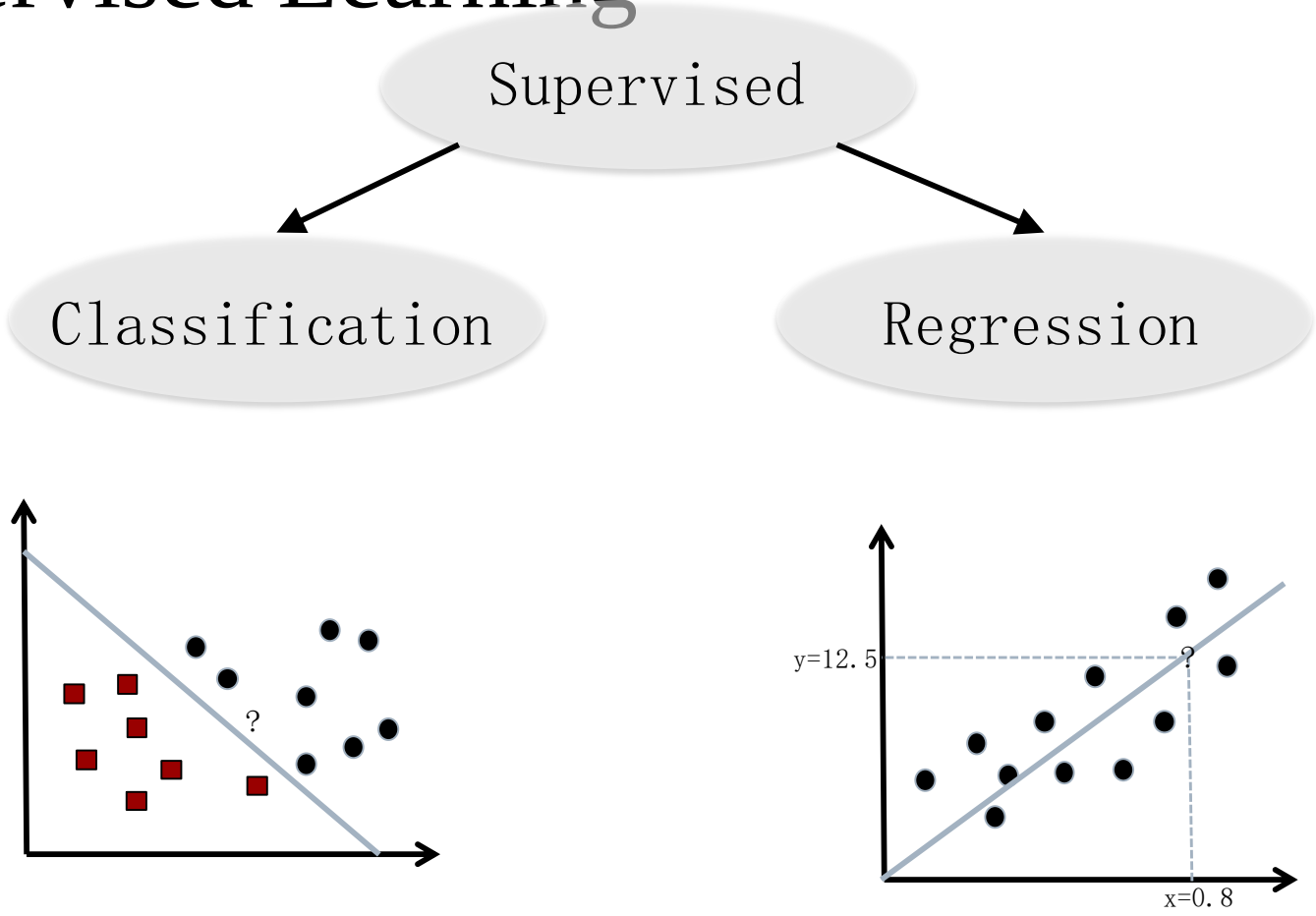
3 Types of Learning



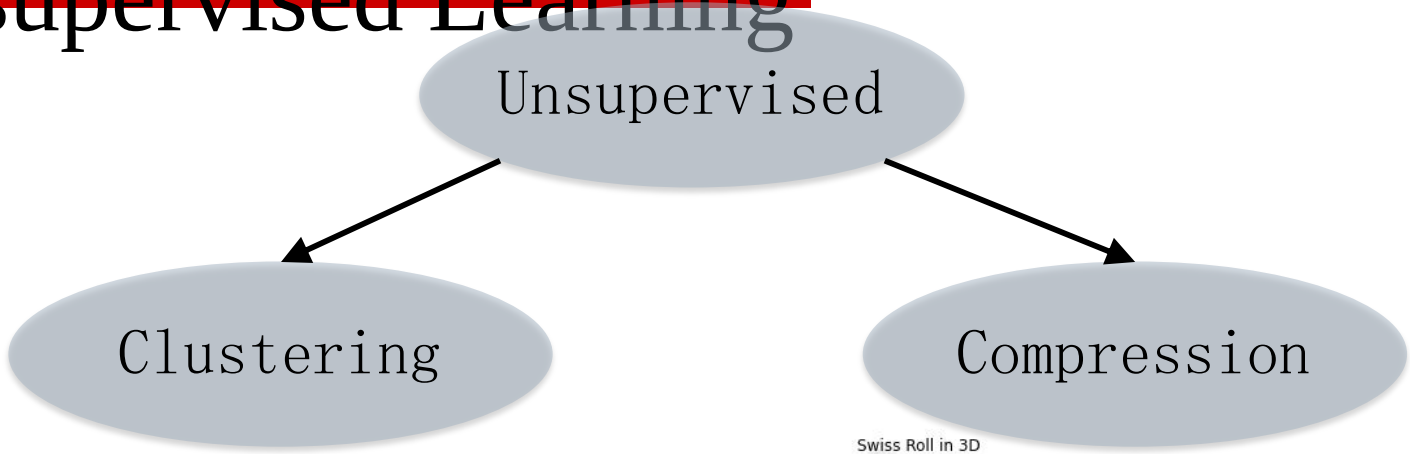
- Discover structure in unlabeled data
- E.g., Document clustering

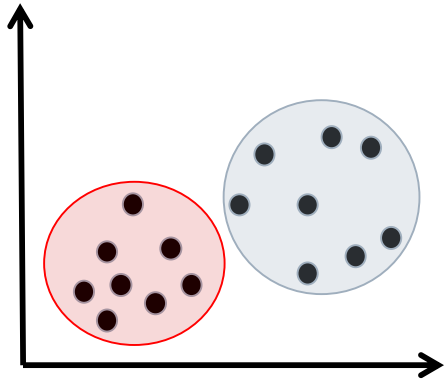
- Learning by "doing" with delayed reward
- E.g., Chess computer

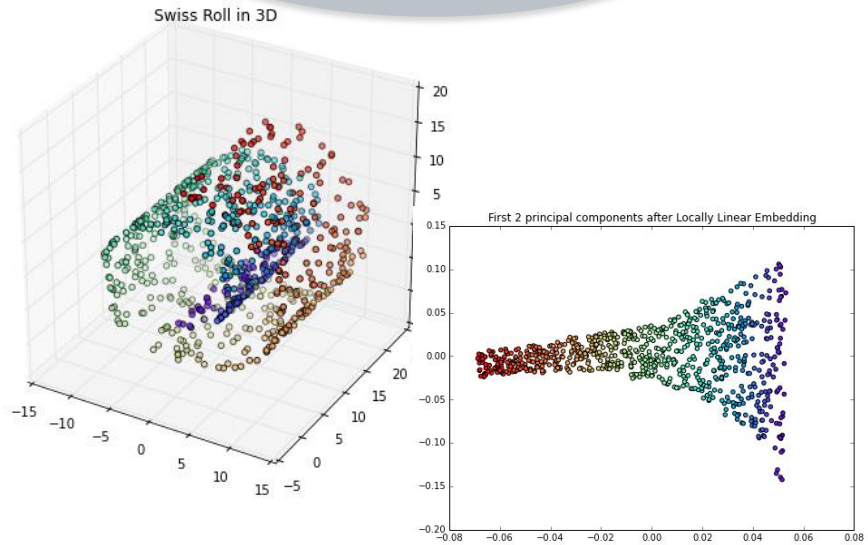
Supervised Learning

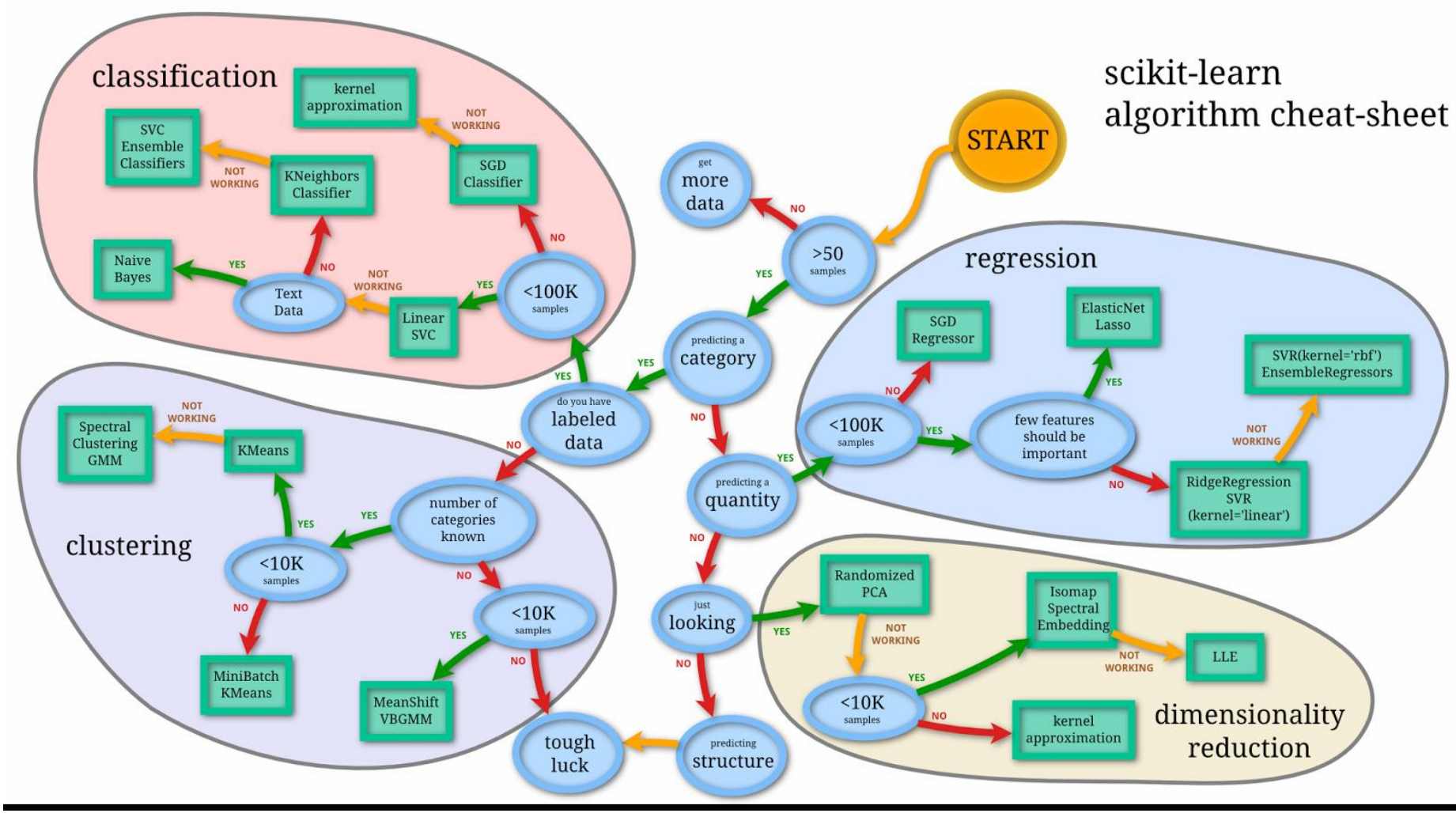


Unsupervised Learning








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

The simplest Sklearn workflow

```
train_x, train_y, test_x, test_y = getData()

model = somemodel()
model.fit(train_x,train_y)
predictions = model.predict(test_x)

score = score_function(test_y, predictions)
```

Flower Classification

Iris-Setosa

Iris-Setosa

Iris-Versicolor

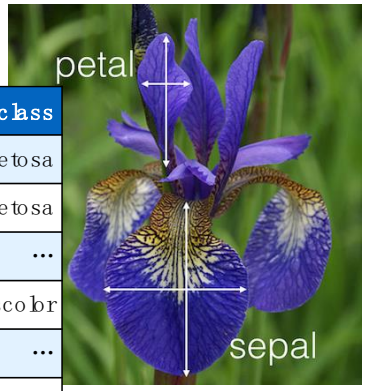
Data Representation

IRIS

https://archive.ics.uci.edu/ml/datasets/Iris

Instances (samples, observations)

	sepal_length	sepal_width	petal_length	petal_width	class
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
…	…	…	…	…	…
50	6.4	3.2	4.5	1.5	veriscolor
…	…	…	…	…	…
150	5.9	3.0	5.1	1.8	virginica



Features (attributes, dimensions)

Classes (targets)

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ x_{31} & x_{32} & \cdots & x_{3D} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}$$

$$\mathbf{y} = [y_1, y_2, y_3, \cdots y_N]$$

```
In [2]: from sklearn.datasets import load_iris
        iris = load_iris()
```

The resulting dataset is a `Bunch` object: you can see what's available using the method `keys()`:

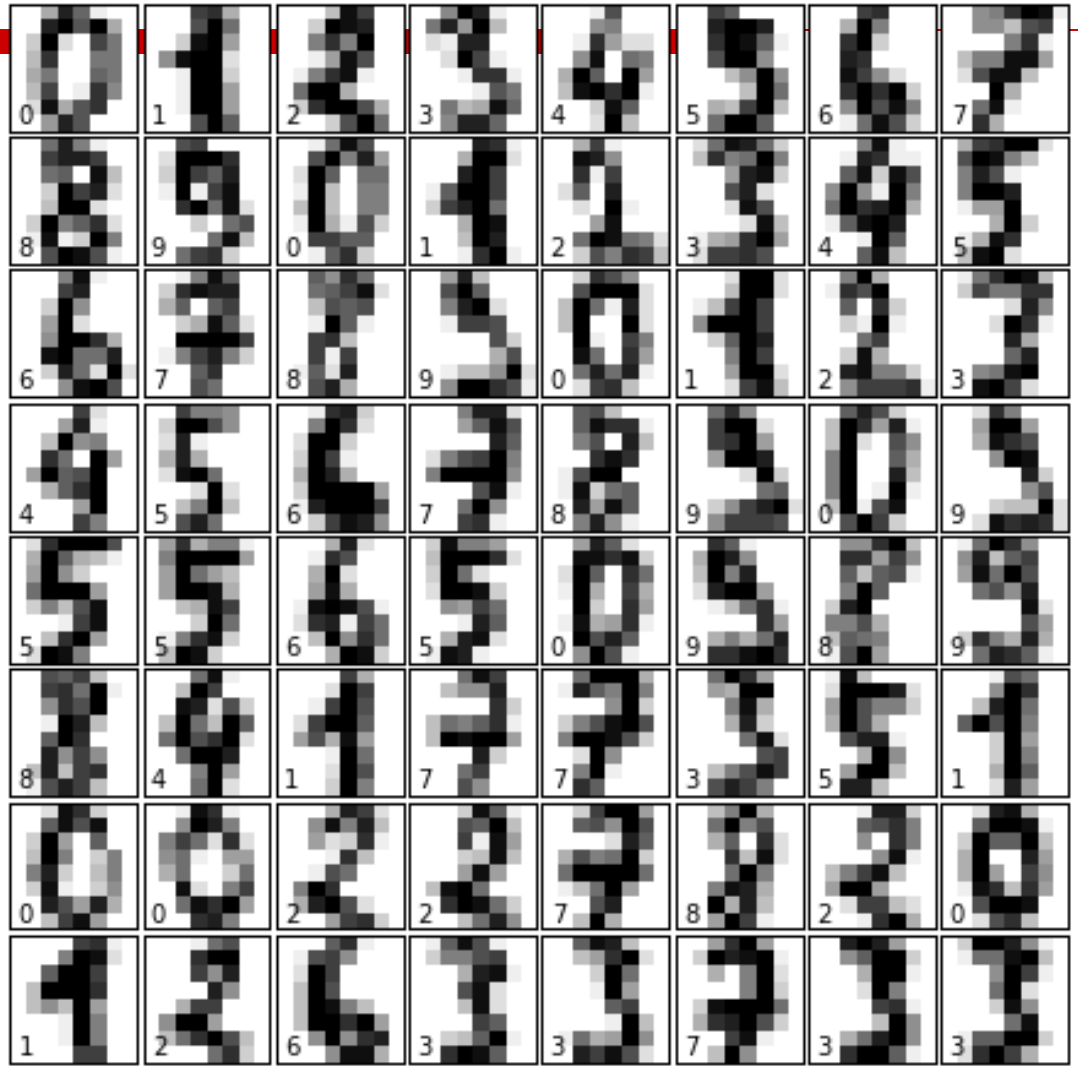
```
In [3]: iris.keys()
Out[3]: dict_keys(['target_names', 'data', 'feature_names', 'DESCR', 'target'])
```

Iris-Setosa

Iris-Setosa

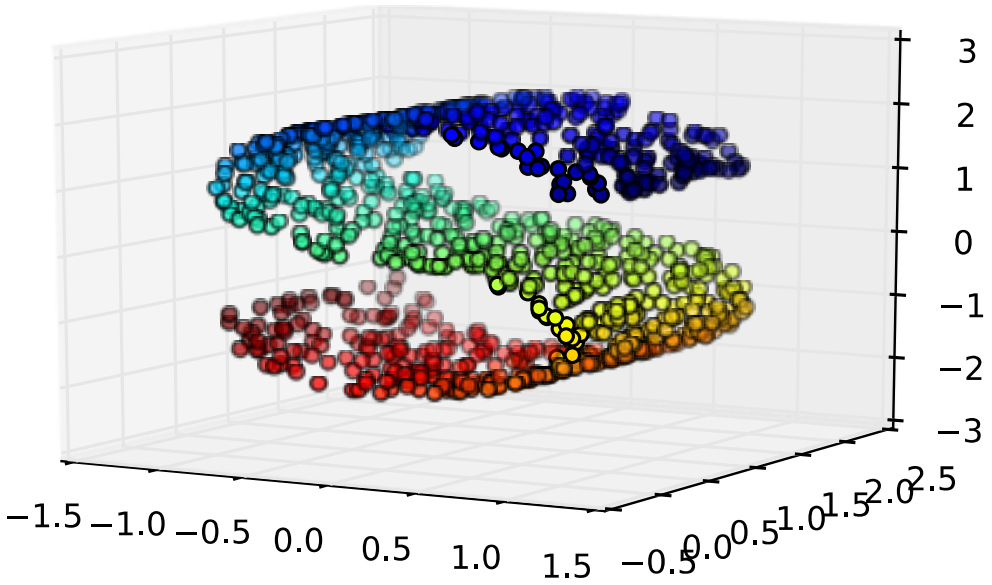
Iris-Versicolor

Digits

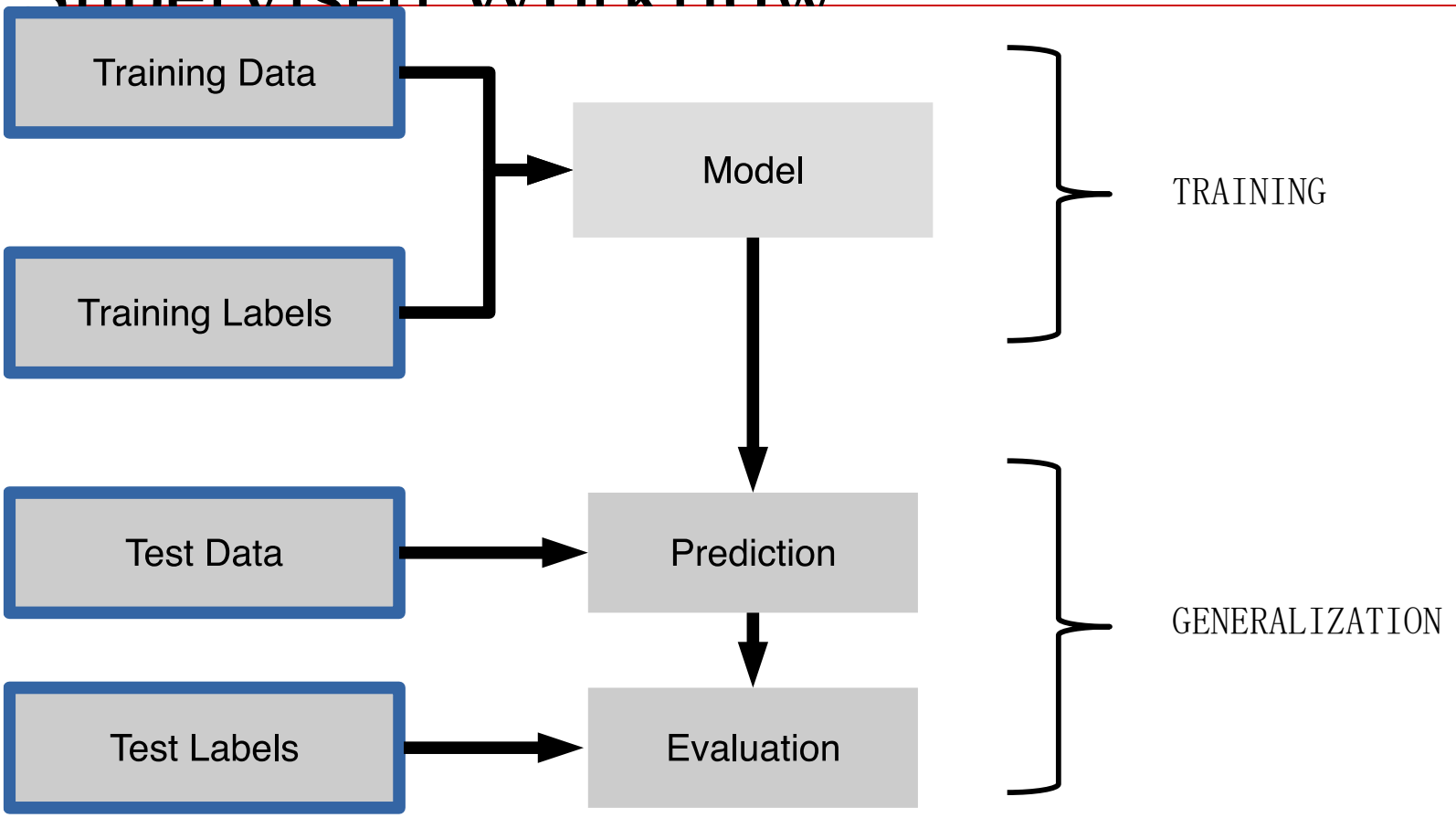

0 1 2 3 4 5 6 7
8 9 0 1 2 3 4 5
6 7 8 9 0 1 2 3
4 5 6 7 8 9 0 9
5 5 6 5 0 9 8 9
8 4 1 7 7 3 5 1
0 0 2 2 7 8 2 0
1 2 6 3 3 7 3 3

Generating Synthetic Data

```
from sklearn.datasets import make_...
```



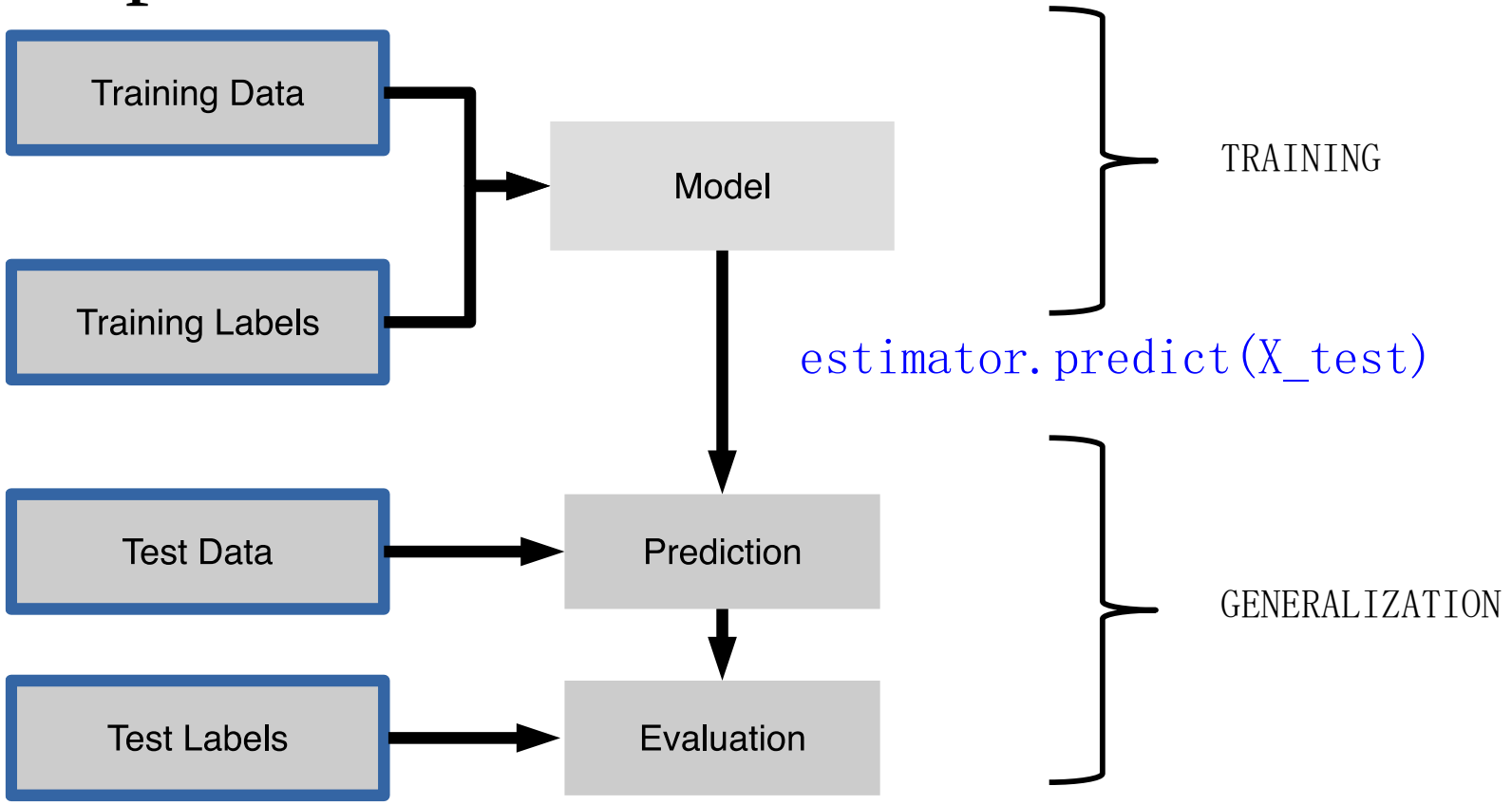
Supervised Workflow



- Fit model on all data after evaluation

Supervised Workflow

estimator.fit(X_train, y_train)

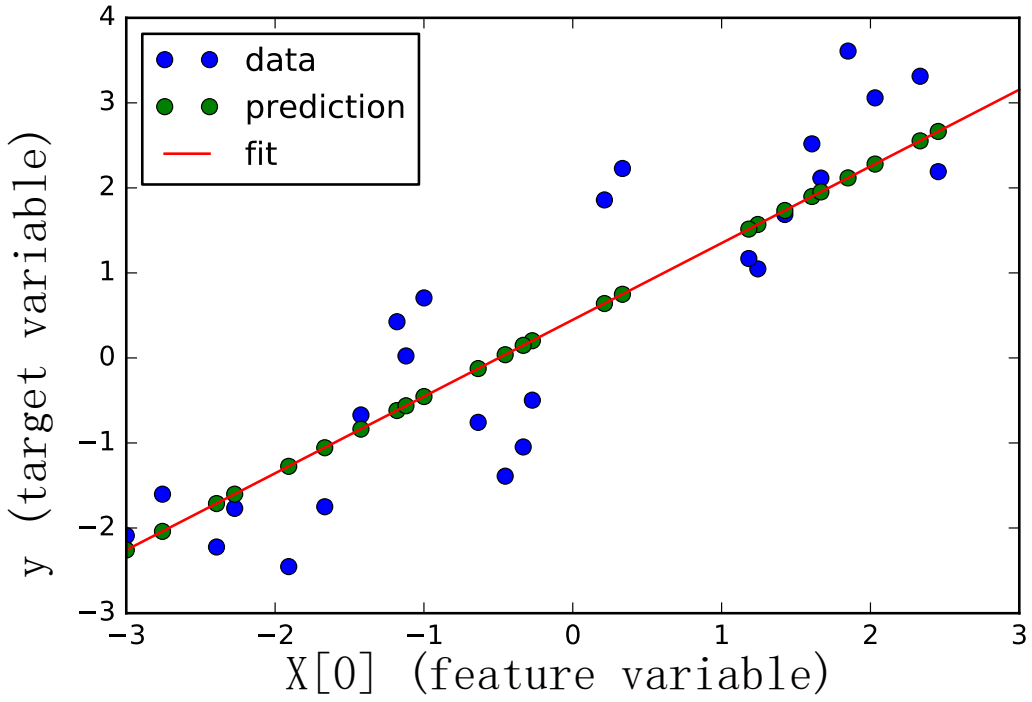


estimator.score(X_test, y_test)

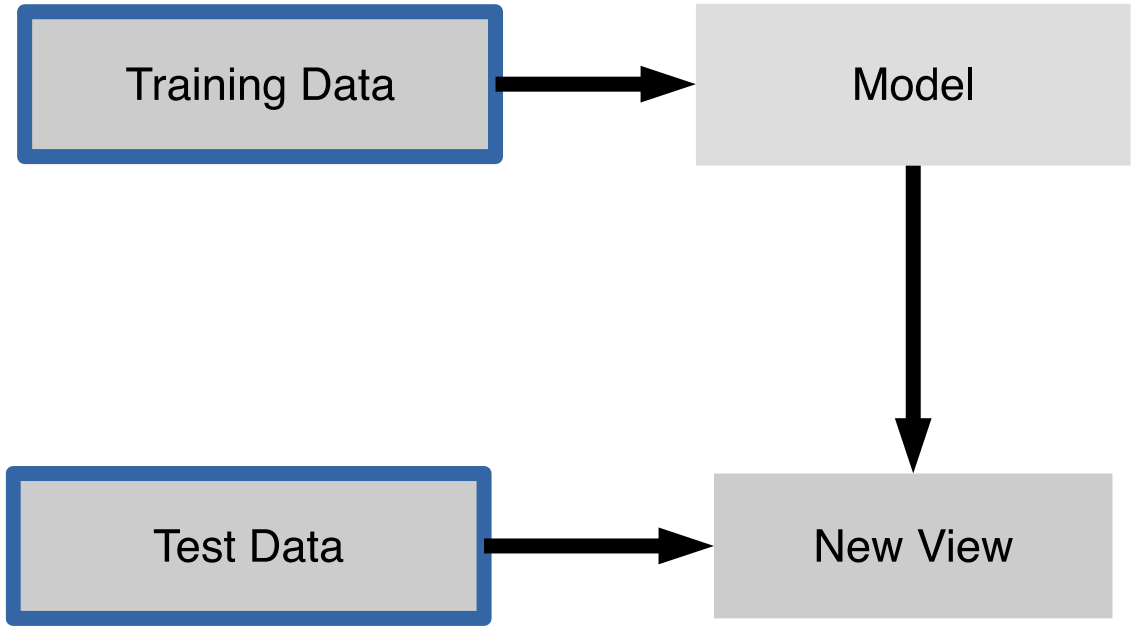
Julyedu.com

Linear Regression

y =coef_[0]*X[0] + intercept_



Unsupervised Transformers



① transformer.fit(X_train)

② X_train_transf = transformer.transform(X_train)

③ X_test_transf = transformer.transform(X_test)

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x}$$

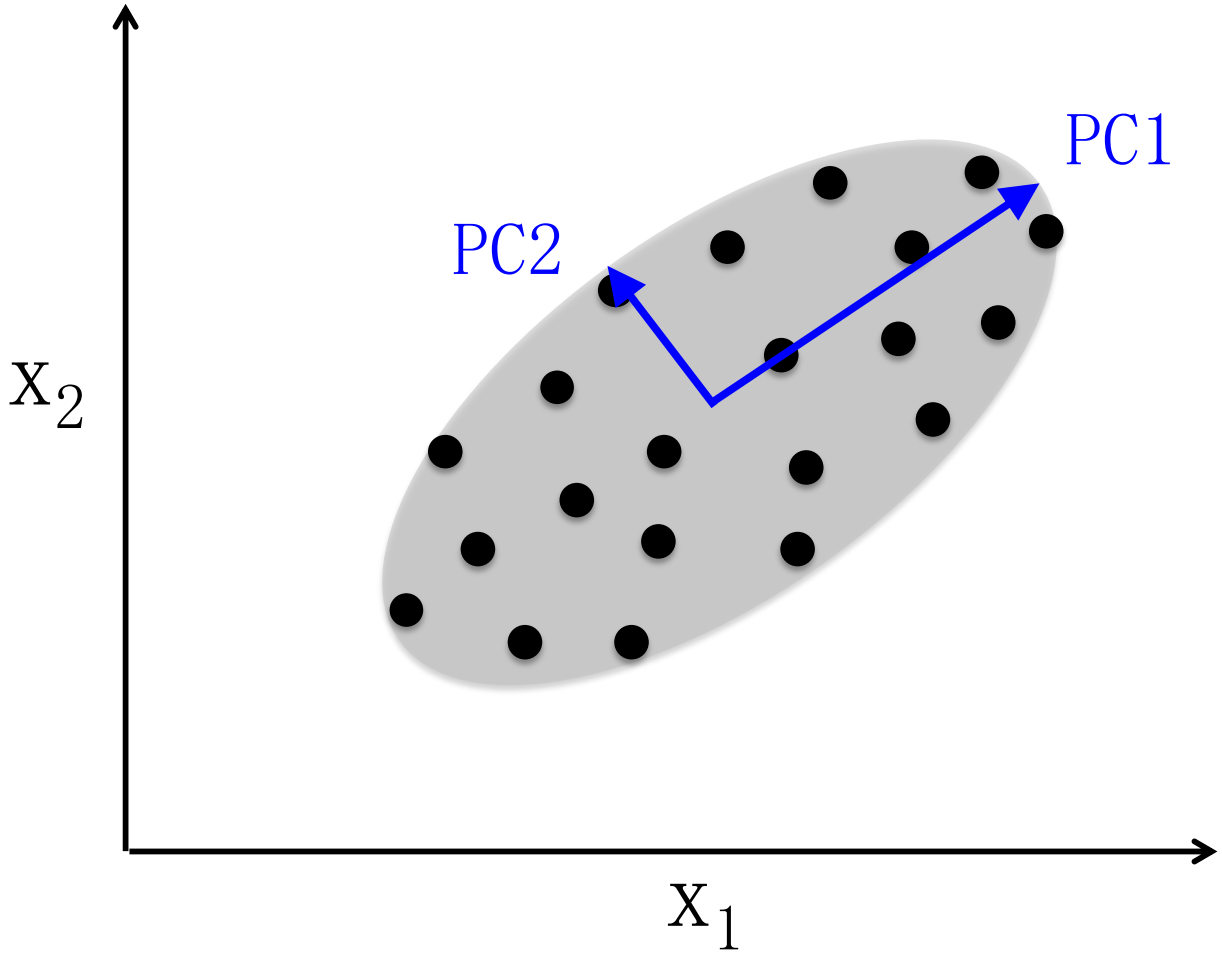
$$x_{norm}^{(i)} = \frac{x^{(i)} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}}$$

standardization

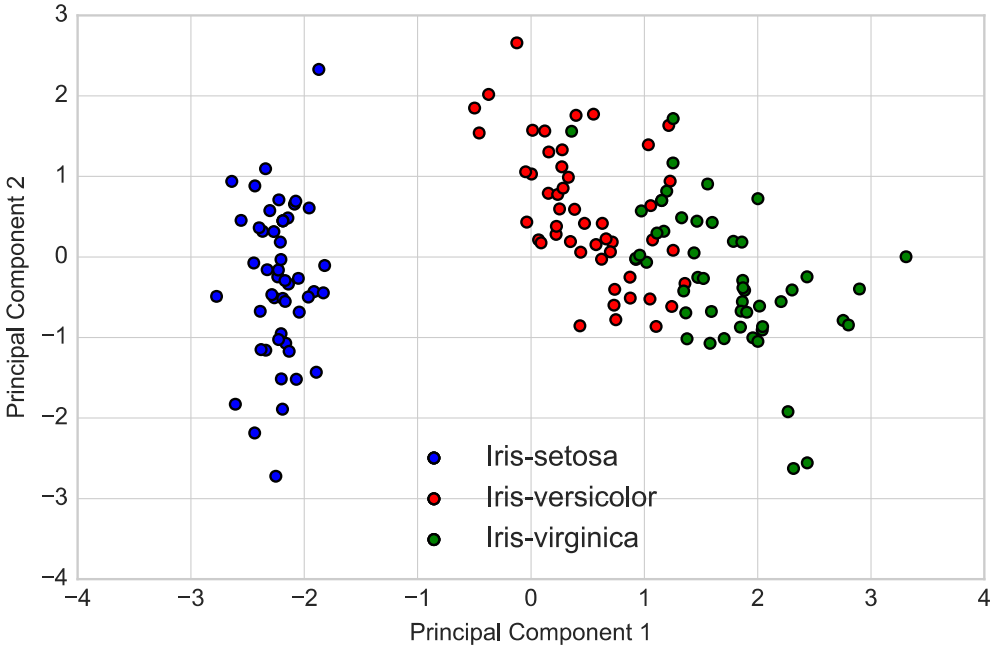
min-max scaling ("normalization")

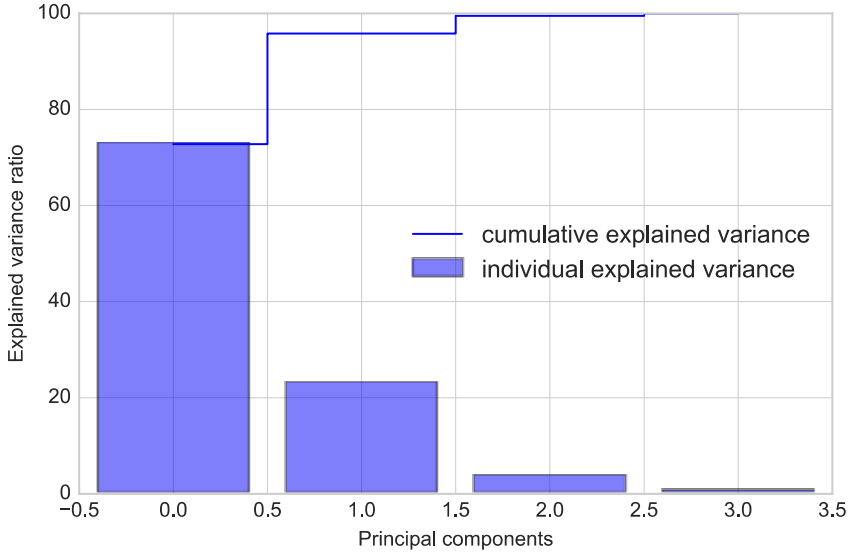
	input	standardized	normalized
0	0	-1.46385	0.0
1	1	-0.87831	0.2
2	2	-0.29277	0.4
3	3	0.29277	0.6
4	4	0.87831	0.8
5	5	1.46385	1.0

Principal Component Analysis



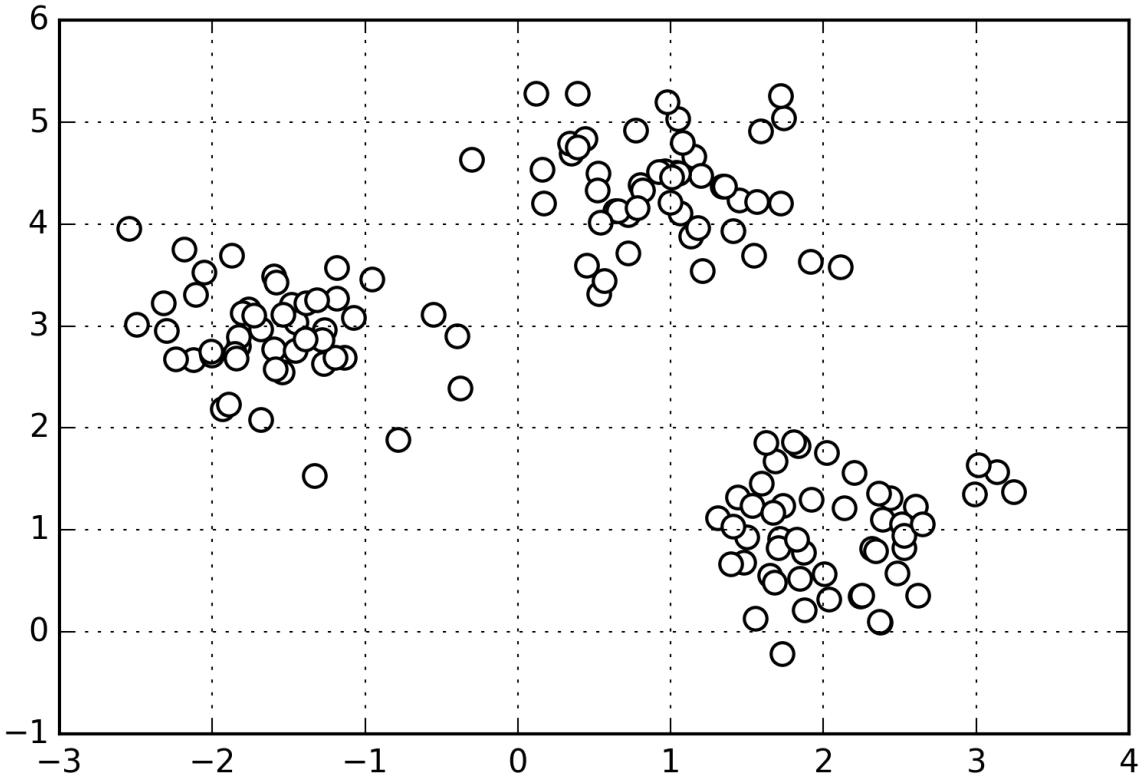
PCA for Dimensionality Reduction





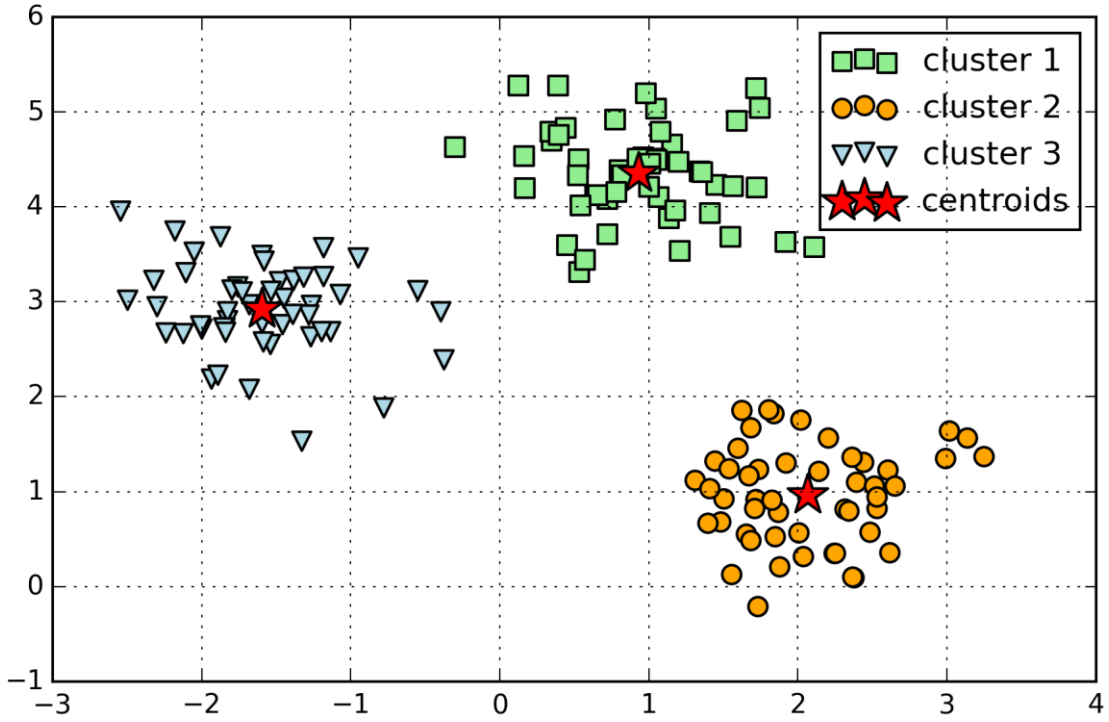
Julyedu.com

K-means Clustering



K-means Clustering

$$d(\boldsymbol{x},\boldsymbol{y})^2=\sum_{j=1}^{m}\left(x_j-y_j\right)^2=\left\|\boldsymbol{x}-\boldsymbol{y}\right\|_2^2$$



Scikit-learn API

`estimator.fit(X_train, [y_train])`	
`estimator.predict(X_test)`	**`estimator.transform(X_test)`**
Classification	Preprocessing
Regression	Dimensionality Reduction
Clustering	Feature Extraction
	Feature selection

Bag of Words

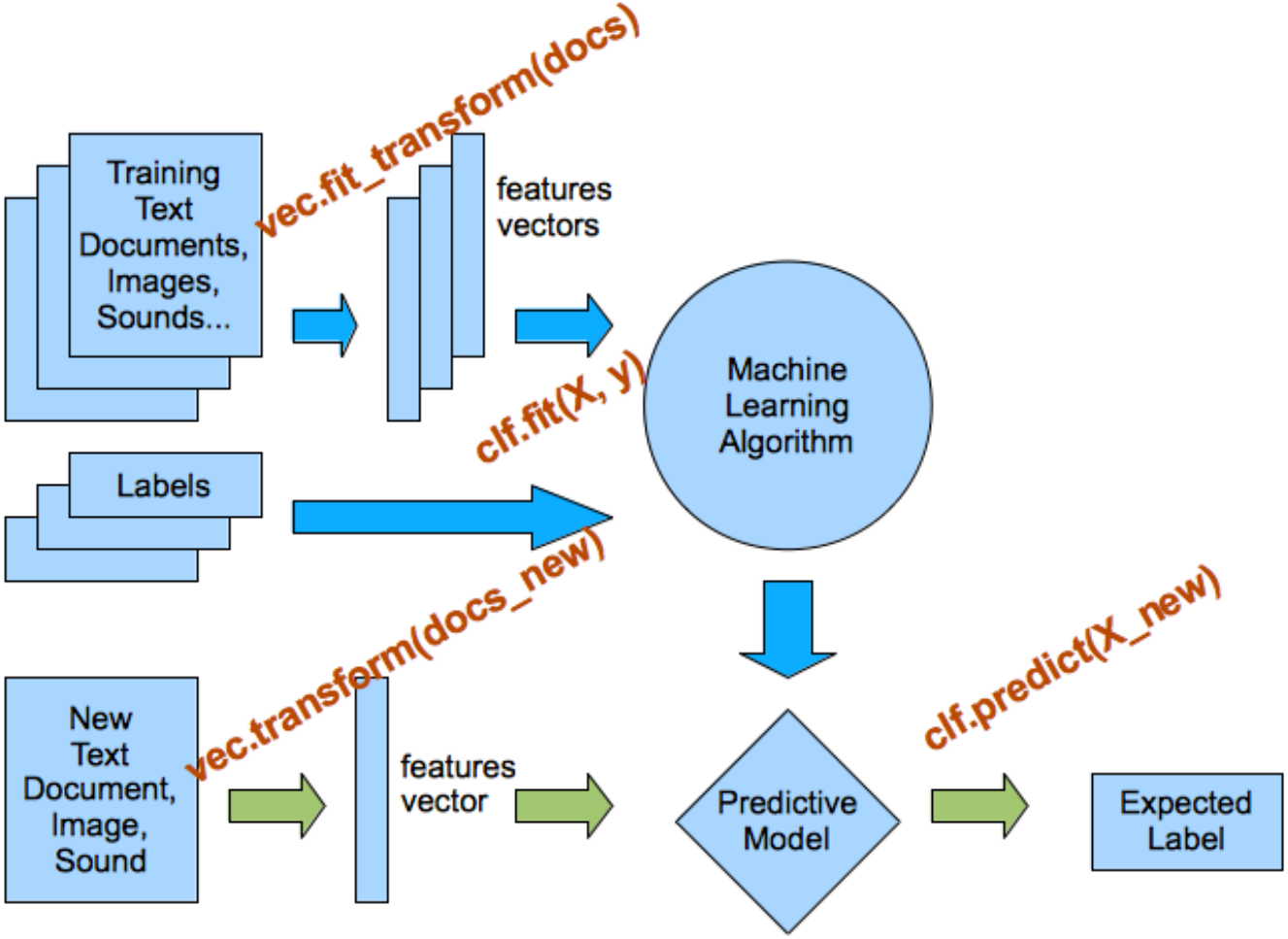
- D1: ”Each state has its own laws.”
- D2: ”Every country has its own culture.”

V ={each:1, state:1, has:2, its:2, own:2,
laws: 1, every: 1, country: 1, culture: 1}

	each	state	has	its	own	laws	every	country	culture
$\mathbf{x}_{D1}$	1	1	1	1	1	1	0	0	0
$\mathbf{x}_{D2}$	0	0	1	1	1	0	1	1	1
$\sum$	1	1	2	2	2	1	1	1	1

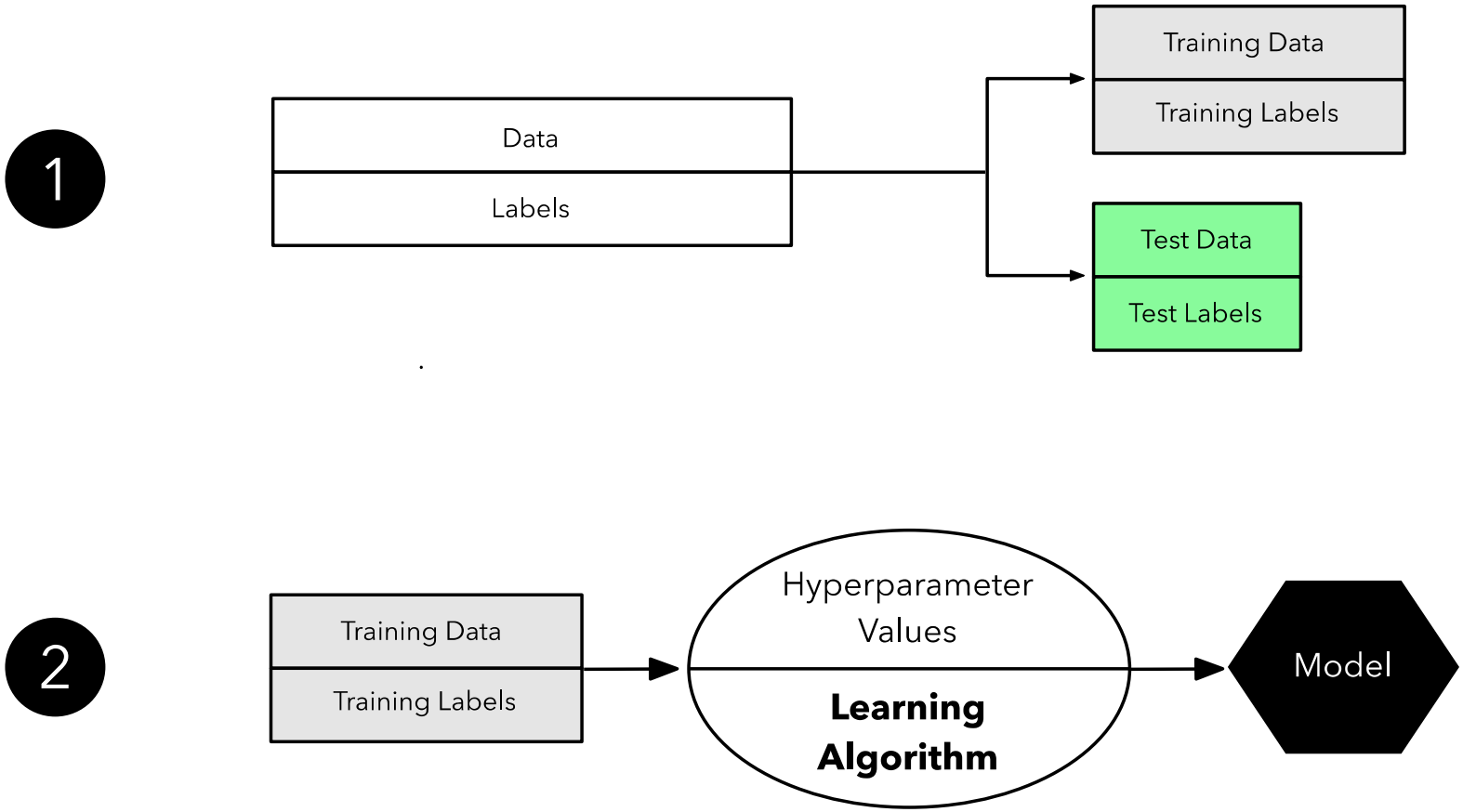
Julyedu.com

Preprocessing & Classification Overview



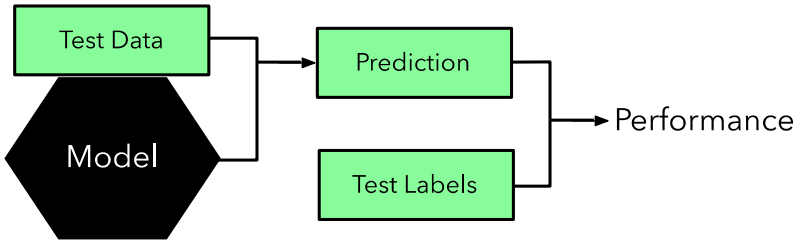
Julyedu.com

Holdout Evaluation I

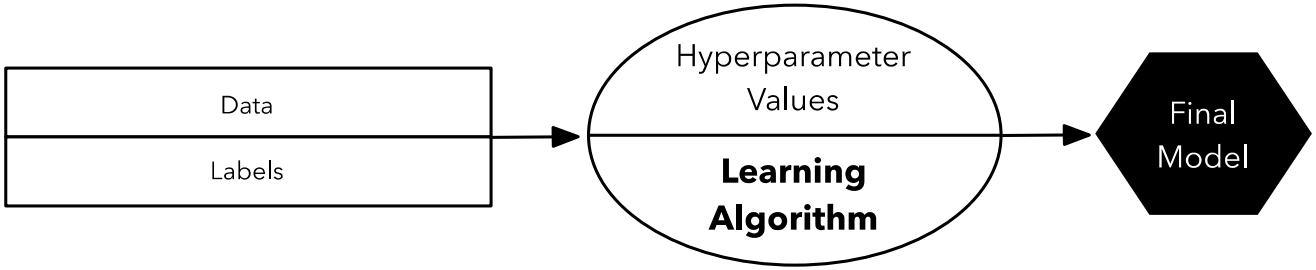


Holdout Evaluation II

3



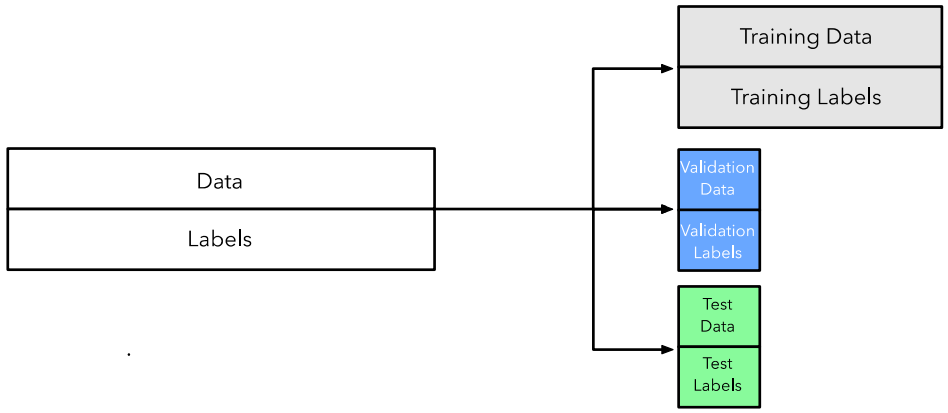
4



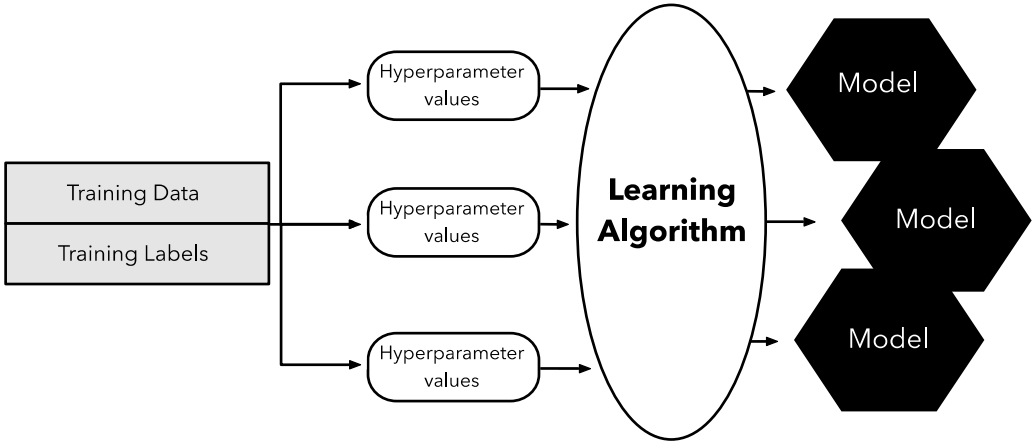
This work by Sebastian Raschka is licensed under a
Creative Commons Attribution 4.0 International License.

Holdout Validation I

1



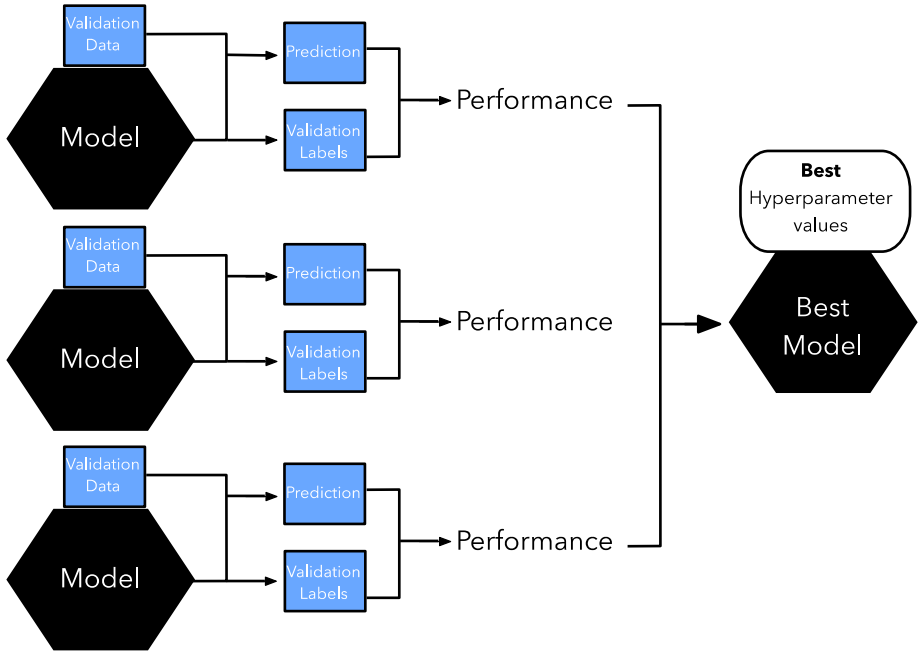
2



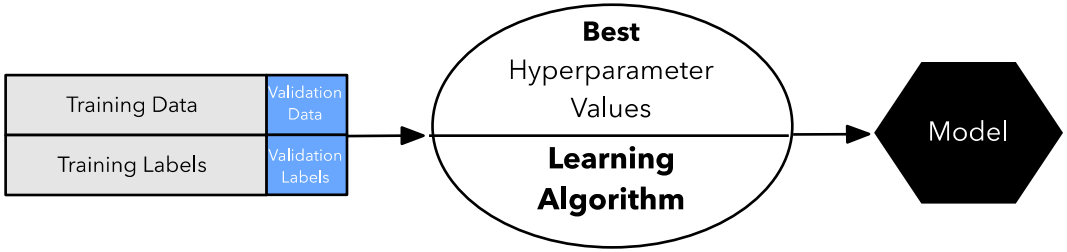
·edu. com

Holdout Validation II

3

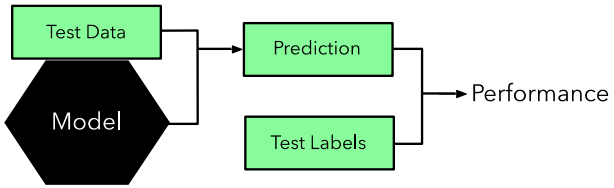


4

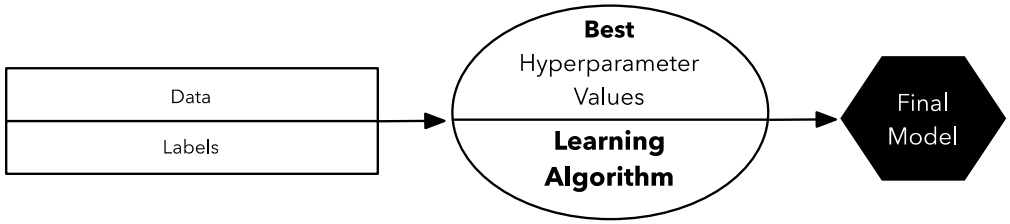


Holdout Validation III

5

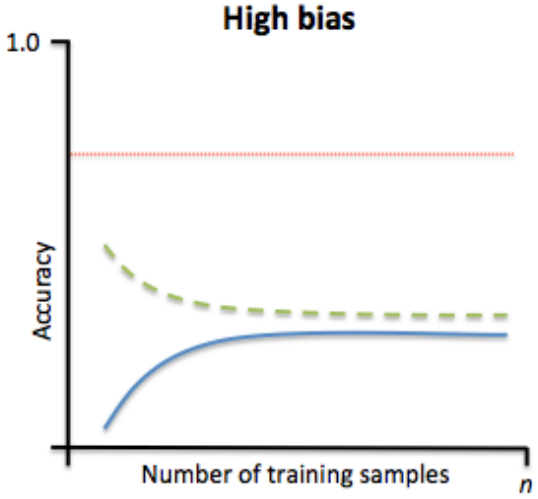


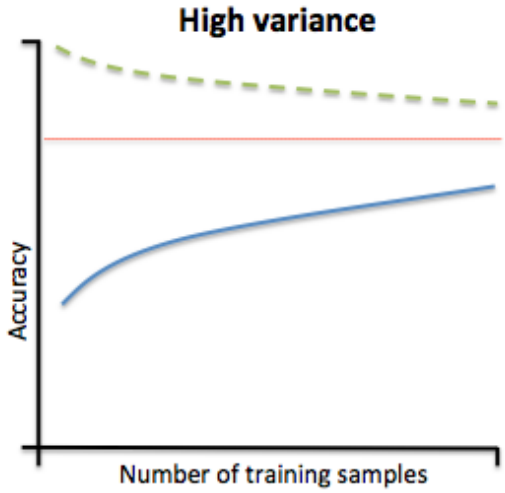
6

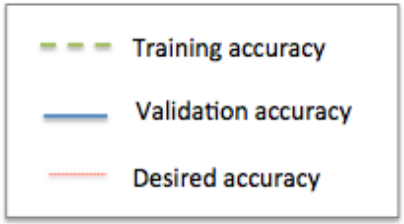


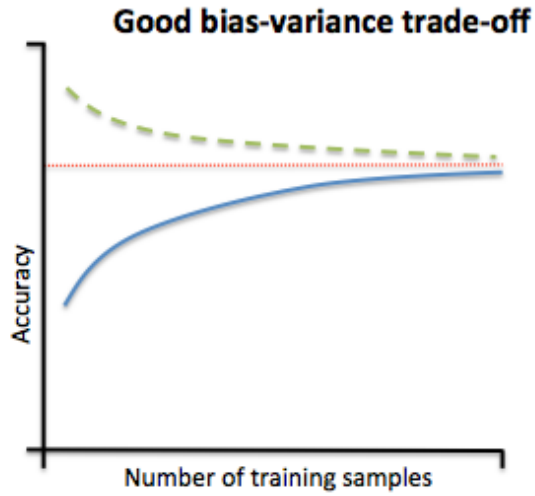
This work by Sebastian Raschka is licensed under a
Creative Commons Attribution 4.0 International License.

Learning Curves



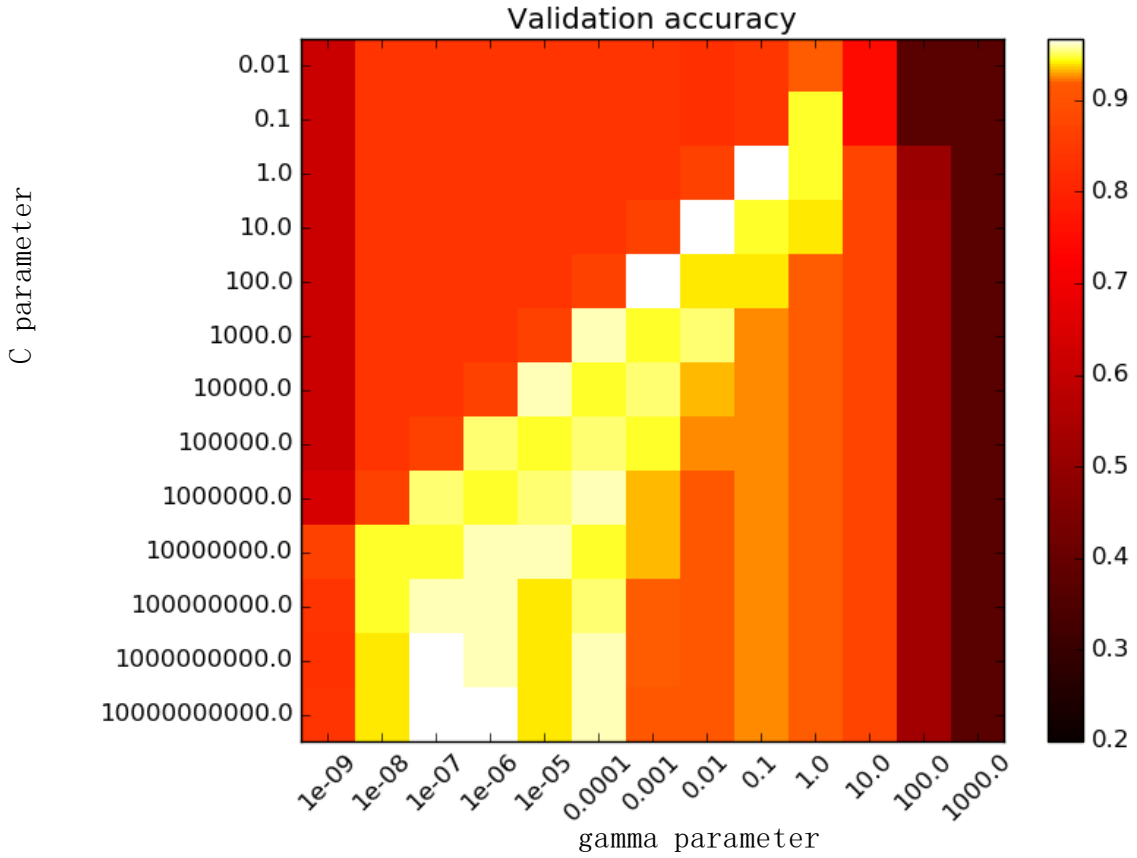




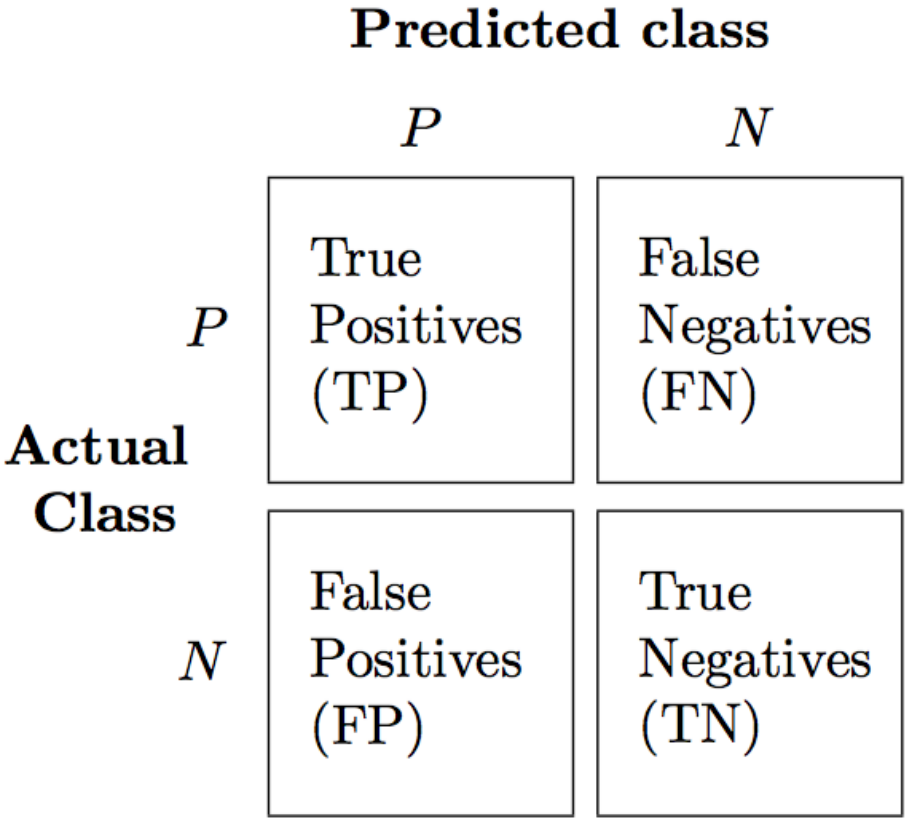


Julyedu.com

Grid Search

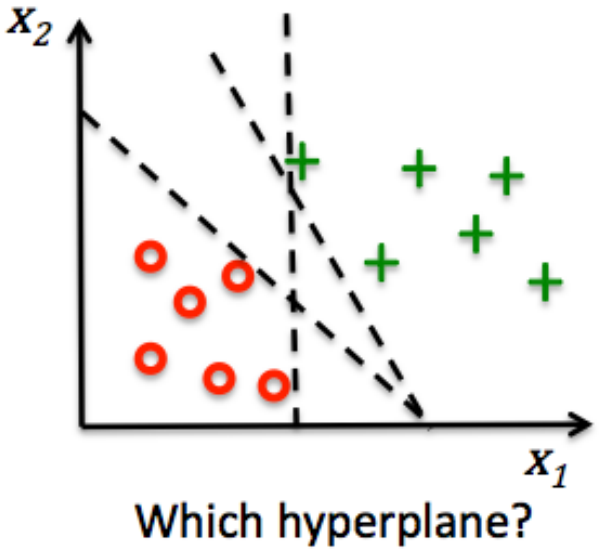

Validation accuracy
C parameter
0.01
0.1
1.0
10.0
100.0
1000.0
10000.0
100000.0
1000000.0
10000000.0
100000000.0
1000000000.0
10000000000.0
1e-09
1e-08
1e-07
1e-06
1e-05
0.0001
0.001
0.01
0.1
1.0
10.0
100.0
1000.0
gamma parameter
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

Confusion Matrix

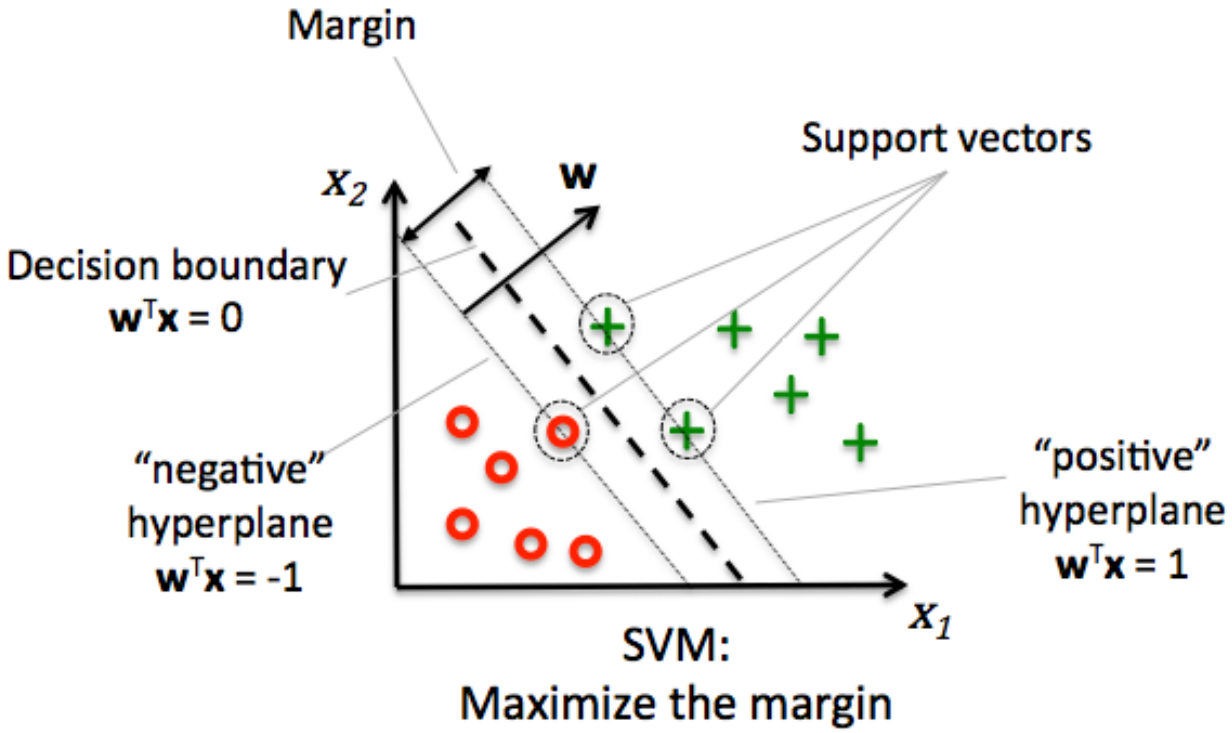


Julyedu.com

Support Vector Machines



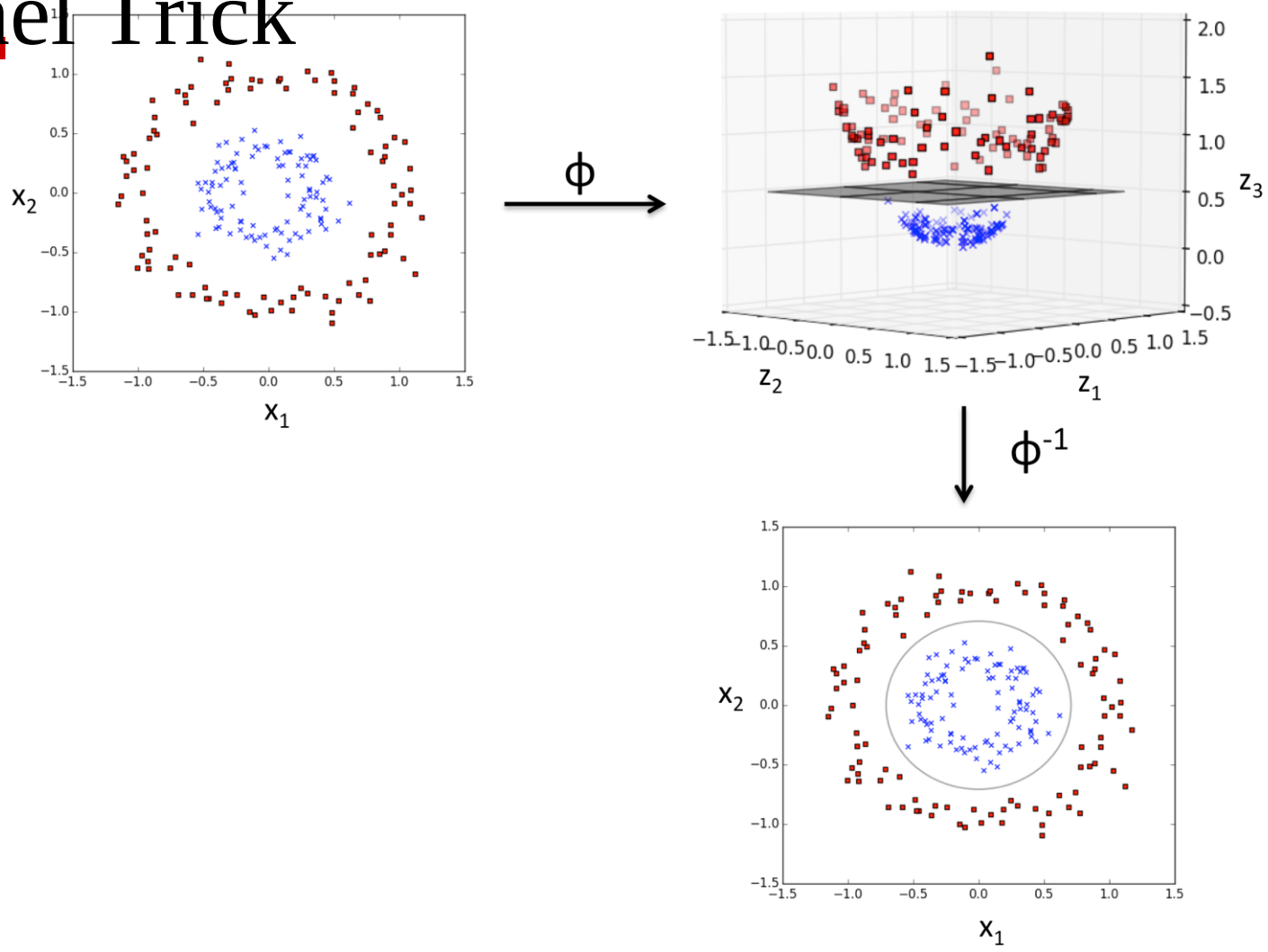
Which hyperplane?



SVM:
Maximize the margin

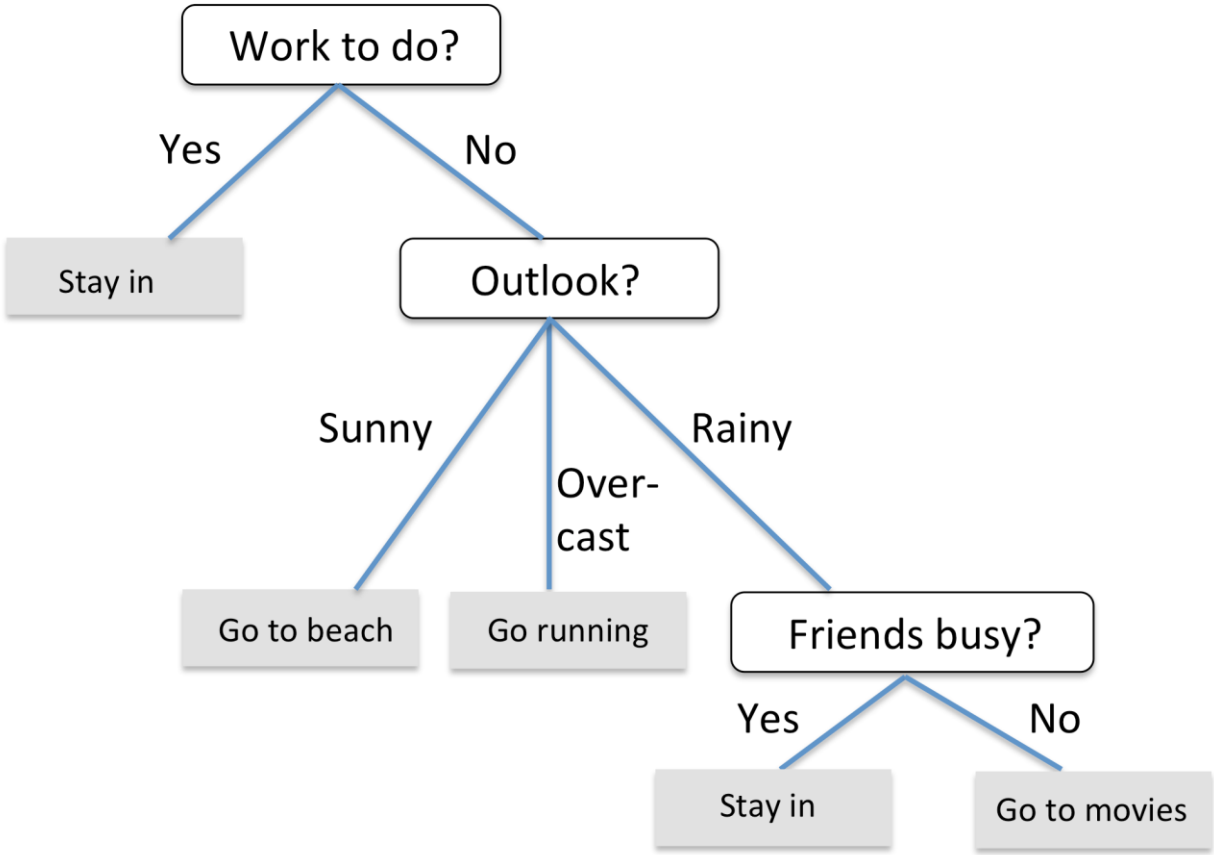
Julyedu.com

Kernel Trick

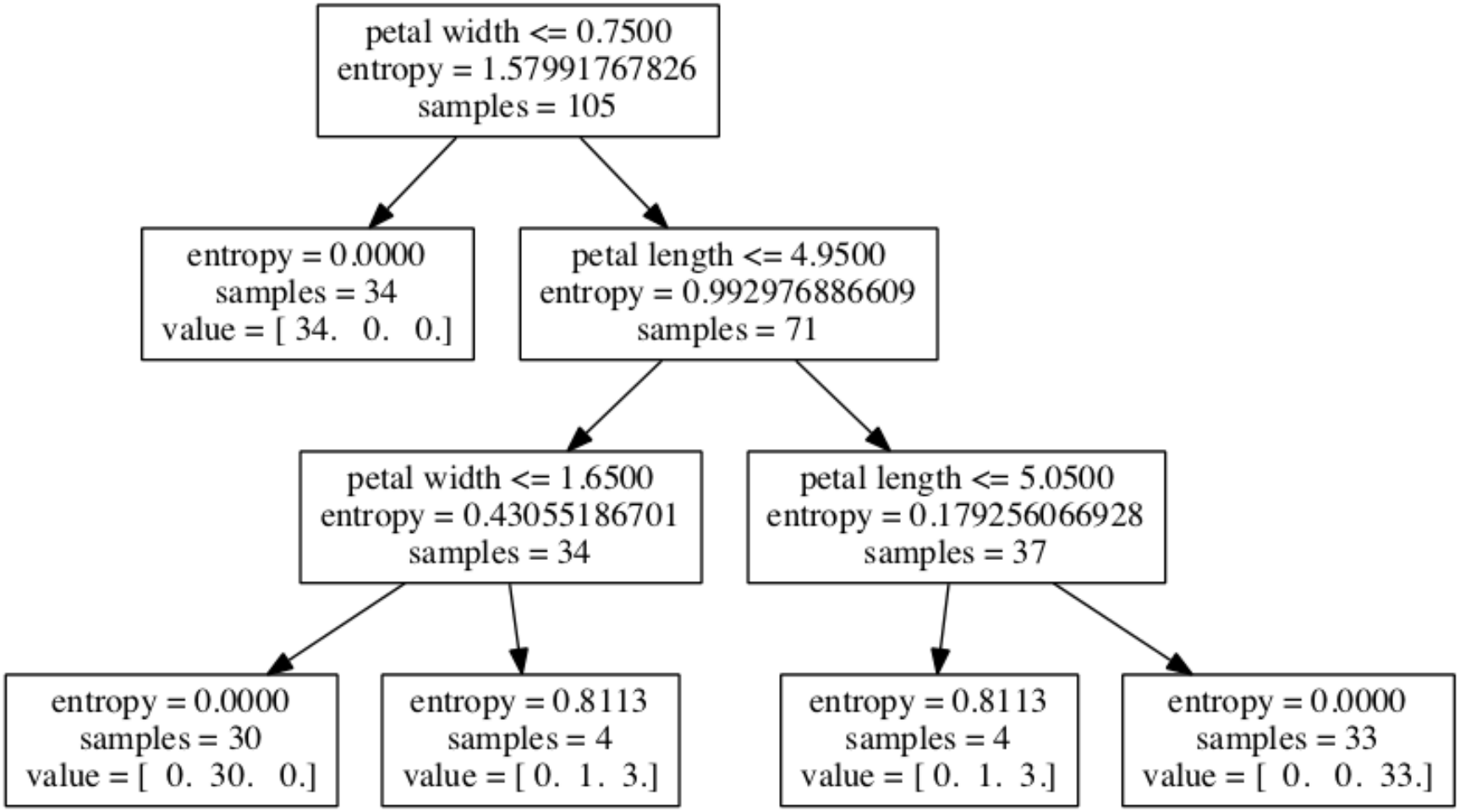


Julyedu.com

Decision Trees

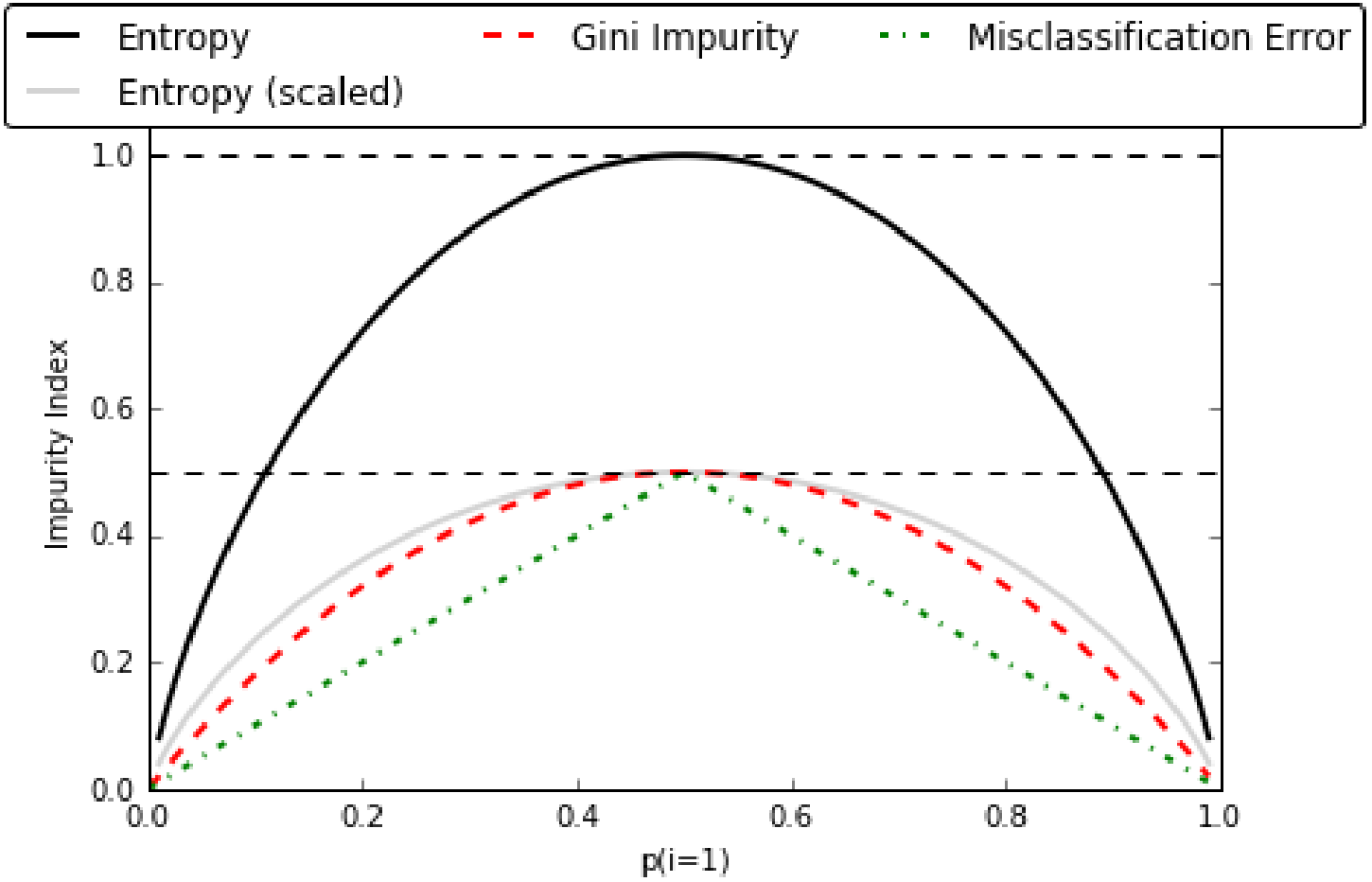


Classification w. Continuous Features



Julyedu.com

Impurity measures



Julyedu.com

Dimensionality Reduction

```python
# Create first network with Keras
from keras.models import Sequential
from keras.layers import Dense
import numpy
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
# load pima indians dataset
dataset = numpy.loadtxt("pima-indians-diabetes.csv", delimiter=",")
# split into input (X) and output (Y) variables
X = dataset[:,0:8]
Y = dataset[:,8]
# create model
model = Sequential()
model.add(Dense(12, input_dim=8, init='uniform', activation='relu'))
model.add(Dense(8, init='uniform', activation='relu'))
model.add(Dense(1, init='uniform', activation='sigmoid'))
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Fit the model
model.fit(X, Y, nb_epoch=150, batch_size=10)
# evaluate the model
scores = model.evaluate(X, Y)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
model = Sequential()
model.add(Convolution2D(nb_filters, kernel_size[0], kernel_size[1], border_mode='valid',input_shape=input_shape))
model.add(Activation('relu'))
model.add(Convolution2D(nb_filters, kernel_size[0], kernel_size[1]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=pool_size))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adadelta',metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=nb_epoch,verbose=1, validation_data=(X_test, Y_test))
score = model.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

```python
import smtplib
from email.mime.text import MIMEText


mailto_list=["julyedu@qq.com,julyedu@126.com"]
mail_host="smtp.126.com" #设置服务器
mail_user="my126account" #用户名
mail_pass="xiSm!x4" #口令
mail_postfix="126.com" #发件箱的后缀
def send_mail(to_list,sub,content): #to_list：收件人；sub：主题；content：邮件内容
    me="yourname"+"<"+mail_user+"@"+mail_postfix+">"
    msg = MIMEText(content,_subtype='html',_charset='gb2312') #创建一个实例，这里设置为html格式邮件
    msg['Subject'] = sub #设置主题
    msg['From'] = me
    msg['To'] = ";".join(to_list)
    s = smtplib.SMTP()
    s.connect(mail_host) #连接smtp服务器
    s.login(mail_user,mail_pass) #登陆服务器
    s.sendmail(me, to_list, msg.as_string()) #发送邮件
    s.close()
content = "test test 123"
send_mail(to_list,sub,content)
```

Selenium 模块实战

Demo时间。。

julyedu.com Julyedu.com