Hands on Supervised Learning Algorithm with Python, Scikit Learn [case: classification]

Instructor:
Heru Praptono
[heru.pra@cs.ui.ac.id]

Additional from samuel louvan's slide, with modification.
some from scikit learn: http://scikit-learn.org/stable/index.html

Agenda

- Supervised Learning Algorithm Refresher
- Classification with Scikit Learn
- Visualisation
  - Use Pandas (or Matplotlib)
- Get your hands dirty

Supervised Algorithm Concept Refresher

- Given a number of instances in dataset $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ where $\mathbf{x} \in \mathcal{R}^D$ and
  $y \in \mathcal{Z}$ (Classification), or
  $y \in \mathcal{R}$ ((General) Regression)
- **Learning**: Estimate the distribution of $p(\theta|\mathcal{D})$
- **Inference**: when there is new $\mathbf{x}$, that is $\mathbf{x}^*$, estimate the value of its corresponding y ; $p(y^*|\mathbf{x}^*, \theta)$

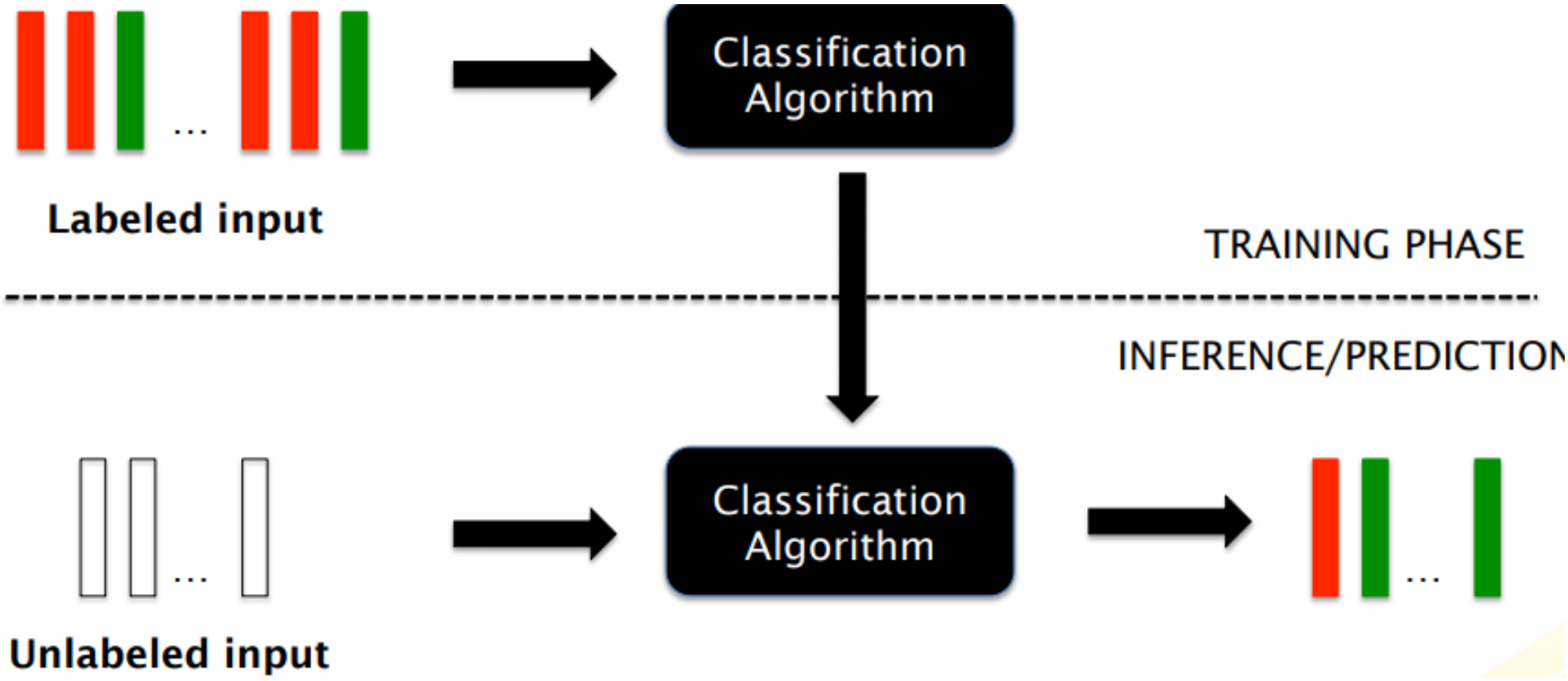
Supervised Algorithm Concept Refresher

- Given :
  - A representation of data (using features/attributes)
  - A fixed set of classes/labels or real values
  - A training set data with labels or real values
- Determine
  - A learning method/algorithm to learn a classifier/(general) regressor
  - The classifier should perform well for “unseen” data

Supervised Algorithm Concept Refresher

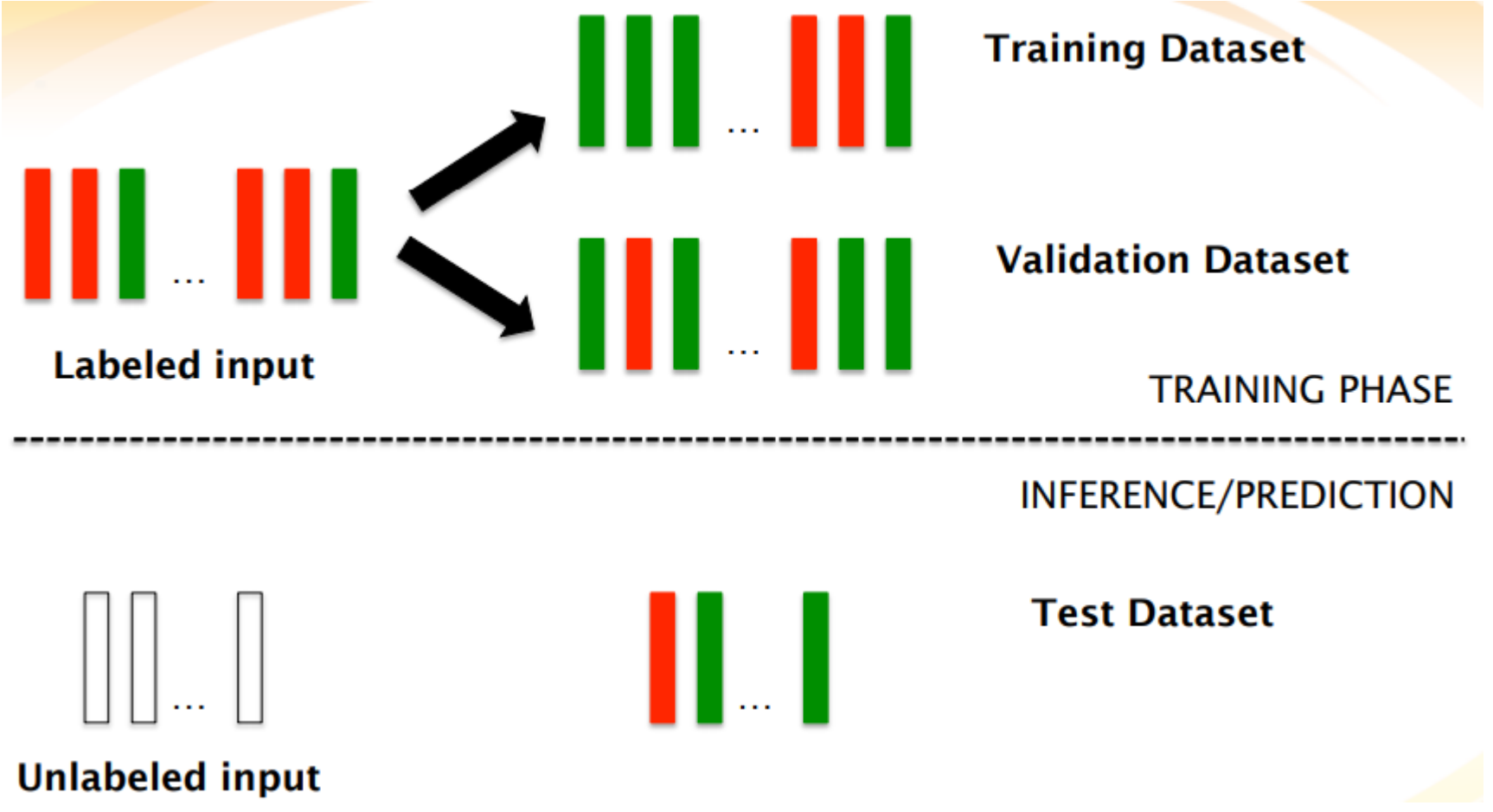
Human learn from past experiences, machines learn from past (data) instances

Supervised Algorithm Concept Refresher (case: classification)



Supervised Algorithm Concept Refresher (case: classification)

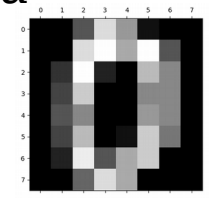
How is it implemented



Supervised Algorithm Concept Refresher (case: classification)

- Previously, we predict/classify digit (inference)

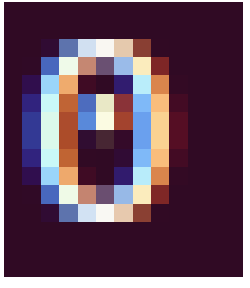
data



```
>>> n = digits.data[0]
>>> n
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

$$\mathbf{x} = \{x_1, x_2, x_3, ..., x_{64}\}$$

Model



Supervised Algorithm Concept Refresher (case: classification)

- Represents the data as features
- Build/train the model
- Evaluate how well the model performs

Let’s try to follow the [example] steps

Remember some classifier algorithms on Scikit-Learn

- Naive Bayes
- SVM
- Decision Tree
- .....and so on..

Some steps on experiment

- Step 0: Import Package Scikit-Learn **as needed/you wish**

```
from sklearn.datasets import load_svmlight_file

from sklearn import preprocessing

from sklearn.cross_validation import train_test_split

from sklearn.metrics import classification_report,
confusion_matrix

from sklearn.linear_model import LogisticRegression

import numpy as np
```

Some steps on experiment

- Step 1: Preprocess the data
  - Standardization
    - Zero mean & unit variance
  - Normalization
  - Binarization
  - Imputation of missing values
  - ..etc

Some steps on experiment

- Step 1: Preprocess the data (zero mean and unit variance)

```
>>> from sklearn import preprocessing
>>> import numpy as np
>>> X = np.array([[ 1., -1.,  2.],
...               [ 2.,  0.,  0.],
...               [ 0.,  1., -1.]])
>>> X_scaled = preprocessing.scale(X)

>>> X_scaled
array([[ 0.  ..., -1.22...,  1.33...],
       [ 1.22...,  0.  ..., -0.26...],
       [-1.22...,  1.22..., -1.06...]])
```

Some steps on experiment

- Step 1: Preprocess the data (scaling to **[0,1]**)

```
>>> X_train = np.array([[ 1., -1.,  2.],
...                     [ 2.,  0.,  0.],
...                     [ 0.,  1., -1.]])
...
>>> min_max_scaler = preprocessing.MinMaxScaler()
>>> X_train_minmax =
min_max_scaler.fit_transform(X_train)
>>> X_train_minmax
array([[ 0.5       ,  0.        ,  1.        ],
       [ 1.        ,  0.5       ,  0.33333333],
       [ 0.        ,  1.        ,  0.        ]])
```

Some steps on experiment

- Step 2: Split the data

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

seed = 7
test_size = 0.2
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=test_size, random_state=seed)
```

Some steps on experiment

- Step 3: Train and Evaluate (consist of learning, inference, with crossval)

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
import sklearn.metrics as metrics
from sklearn.model_selection import cross_val_predict


seed = 7
num_folds = 5
k_fold = KFold(n_splits=num_folds, random_state=seed)
model = LogisticRegression()
predicted = cross_val_predict(model, X_train, Y_train, cv=k_fold)
print ("Accuracy : %.3f "%(metrics.accuracy_score(Y_train, predicted) * 100))
```

Or print classification report

```
from sklearn.metrics import classification_report
report = classification_report(Y_train, predicted)
```

Some steps on experiment

- Step 4: Do learning from training dataset

```
model.fit(X_train, y_train)
```

Some steps on experiment

- Step 5 (optional) : if you wish to **save** the model

```
from pickle import dump
from pickle import load

file_name = "final_model.sav"
dump(model, open(file_name, "wb"))
```

Some steps on experiment

- Step 5 (optional) : then you load from the model saved

```
from pickle import dump
from pickle import load

loaded_model = load(open(file_name, "rb"))
```

Some steps on experiment

- Step 6: Inference/prediction (and get an evaluation)

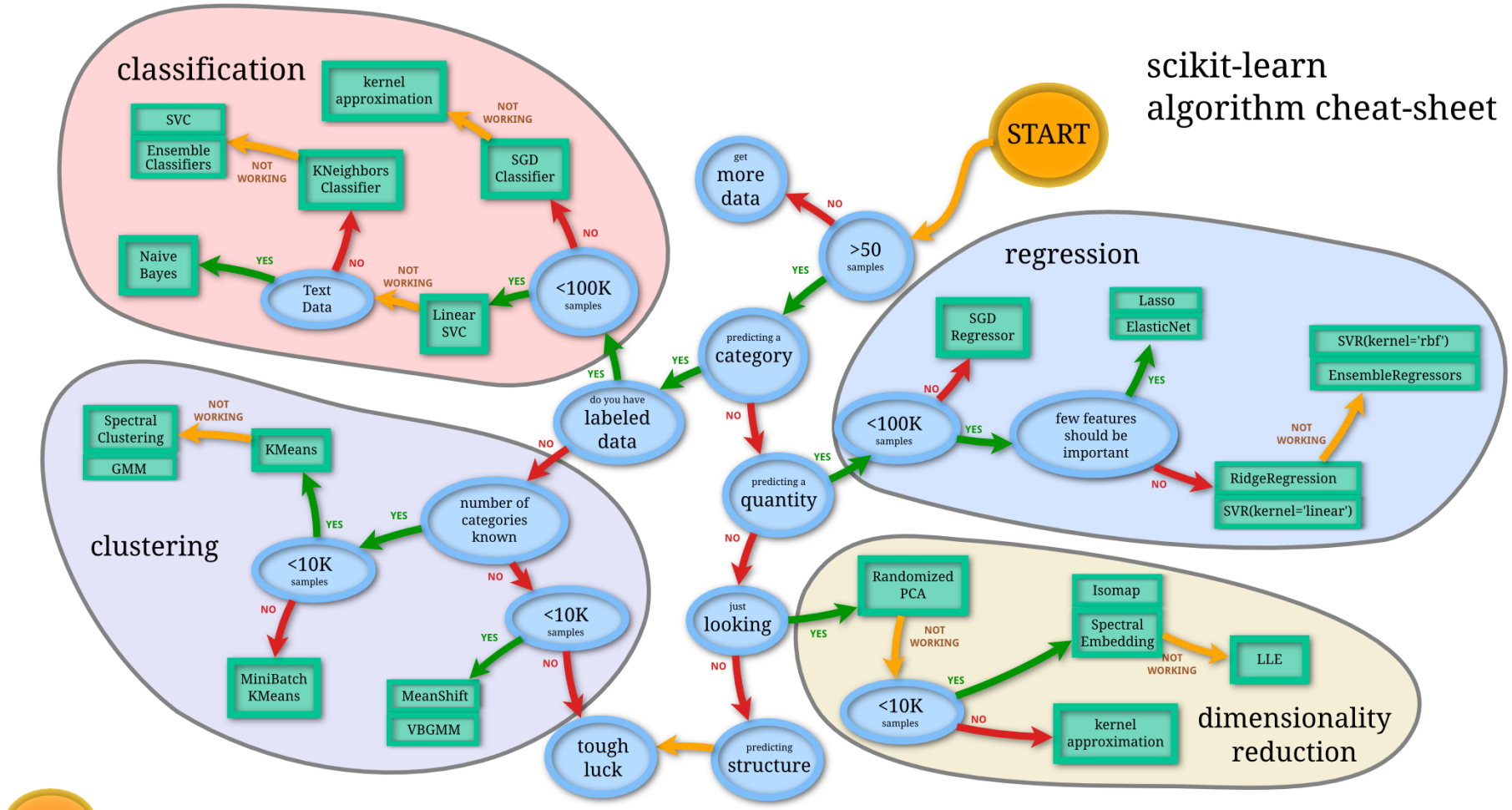
```
predicted = loaded_model.predict(X_test)
metrics.precision_score(Y_test, predicted)
```

Data Analysis toolkit

- Pandas

UNIVERSITAS INDONESIA
pusilkom ui

scikit-learn algorithm cheat-sheet



Back

scikit learn

It is now your turn.
Open the pdf document,
and explore on your own