

WOMEN WHO

**SSID: WWCode**

**Password: icecream12345**



# MACHINE LEARNING AND AI STUDY GROUP

- Download these datasets:
  - <http://bit.ly/2os5SHm>
  - <http://bit.ly/2phqFfT>
- Please answer the following form: <http://bit.ly/2qzdys6>
- Having fun with AI:
  - <https://quickdraw.withgoogle.com/>
  - <http://en.akinator.com/>
  - <https://aiexperiments.withgoogle.com/>





# WOMEN WHO CODE MANILA

## MACHINE LEARNING AND AI STUDY GROUP

Twitter: [@wwcodemanila](https://twitter.com/wwcodemanila)  
FB: [fb.com/wwcodemanila](https://fb.com/wwcodemanila)



**Isabelle Tingzon**

Research Fellow II

Philippine-California Advanced Research Institutes (PCARI)

Graduate student of Computer Science



# WOMEN WHO CODE<sup>®</sup>

## **New Member's Introduction**



**I am <name>**

<your current profession>

<why did you join this study group?>



# **Introduction to Machine Learning (Beginners)**



# Machine Learning



what society thinks I  
do



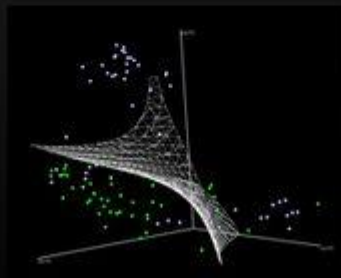
what my friends think  
I do



what my parents think  
I do

$$\begin{aligned} L_p &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^n \alpha_i \\ \alpha_i &\geq 0, \forall i \\ \mathbf{w} &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i, \sum_{i=1}^n \alpha_i y_i = 0 \\ \nabla \hat{g}(\theta_t) &= \frac{1}{n} \sum_{i=1}^n \nabla \ell(x_i, y_i; \theta_t) + \nabla r(\theta_t) \\ \theta_{t+1} &= \theta_t - \eta_t \nabla \ell(x_{i(t)}, y_{i(t)}; \theta_t) - \eta_t \cdot \nabla r(\theta_t) \\ \mathbb{E}_{i(t)}[\ell(x_{i(t)}, y_{i(t)}; \theta_t)] &= \frac{1}{n} \sum_{i=1}^n \ell(x_i, y_i; \theta_t). \end{aligned}$$

what other programmers  
think I do



what I think I do

```
>>> from scipy import svm
```

what I really do



# Machine Learning

A branch of AI that gives computers the ability to learn from examples and experience without being explicitly programmed.

# Write a program to classify images



Manually program rules:

```
def detect_colors(image):  
    # lots of code  
  
def detect_edges(image):  
    # lots of code  
  
def analyze_shapes(image):  
    # lots of code  
  
def guess_texture(image):  
    # lots of code  
  
def define_fruit():  
    # lots of code  
  
def handle_probability():  
    # lots of code
```

# Write a program to classify images



Manually program rules:

```
def detect_colors(image):  
    # lots of code  
  
def detect_edges(image):  
    # lots of code  
  
def detect_shape(image):  
    # lots of code  
  
def guess_breed(image):  
    # lots of code  
  
def handle_probability():  
    # lots of code
```

# Classifiers and Learning Algorithms

Learns the rules for us so that we don't need to explicitly program it!

A function that takes data as input and assigns a label as output



# Classifiers and Learning Algorithms

Learns the rules for us so that we don't need to explicitly program it!

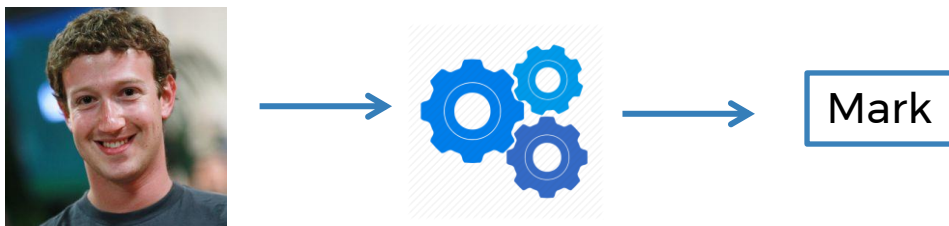
A function that takes data as input and assigns a label as output



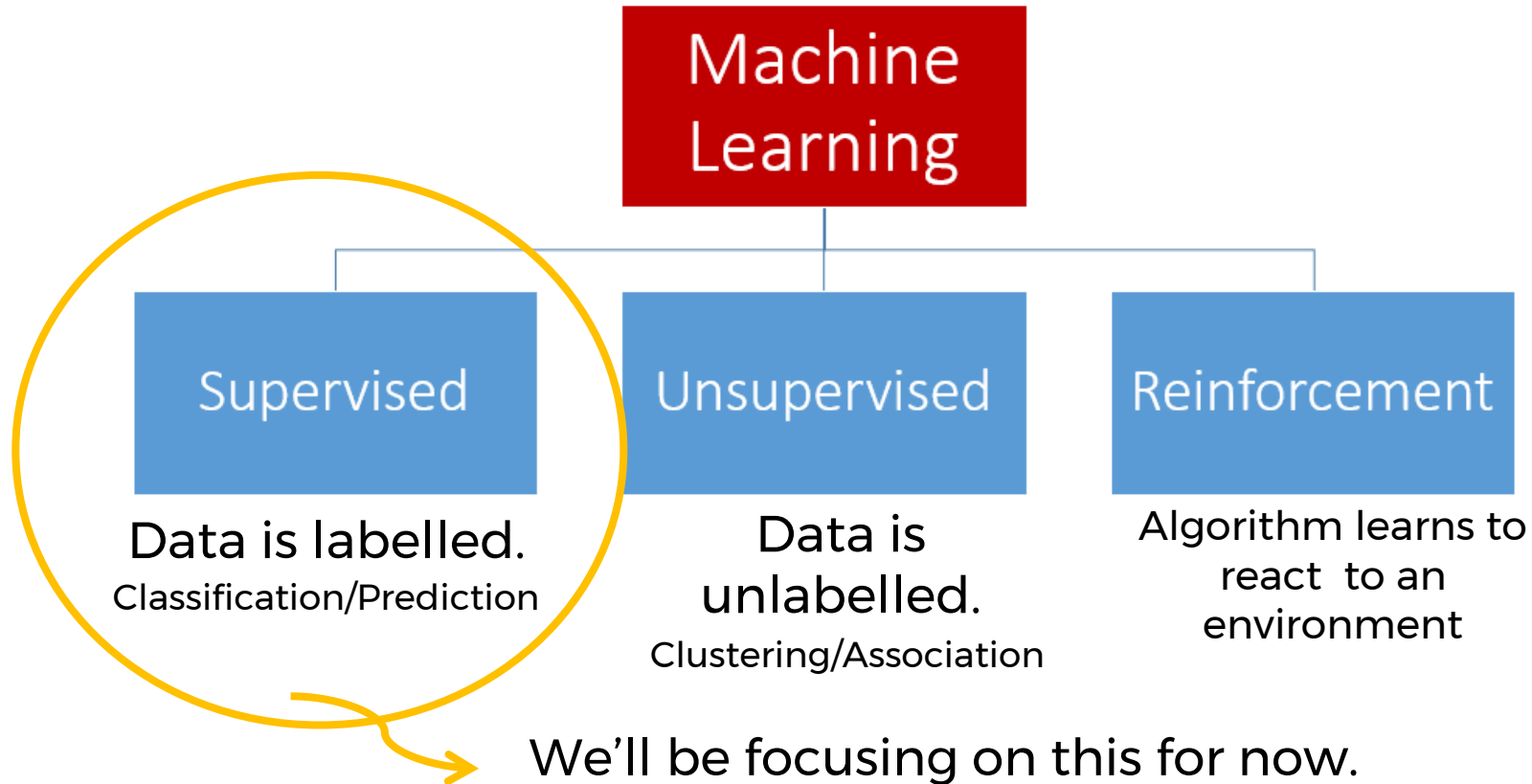
# Classifiers and Learning Algorithms

Learns the rules for us so that we don't need to explicitly program it!

A function that takes data as input and assigns a label as output



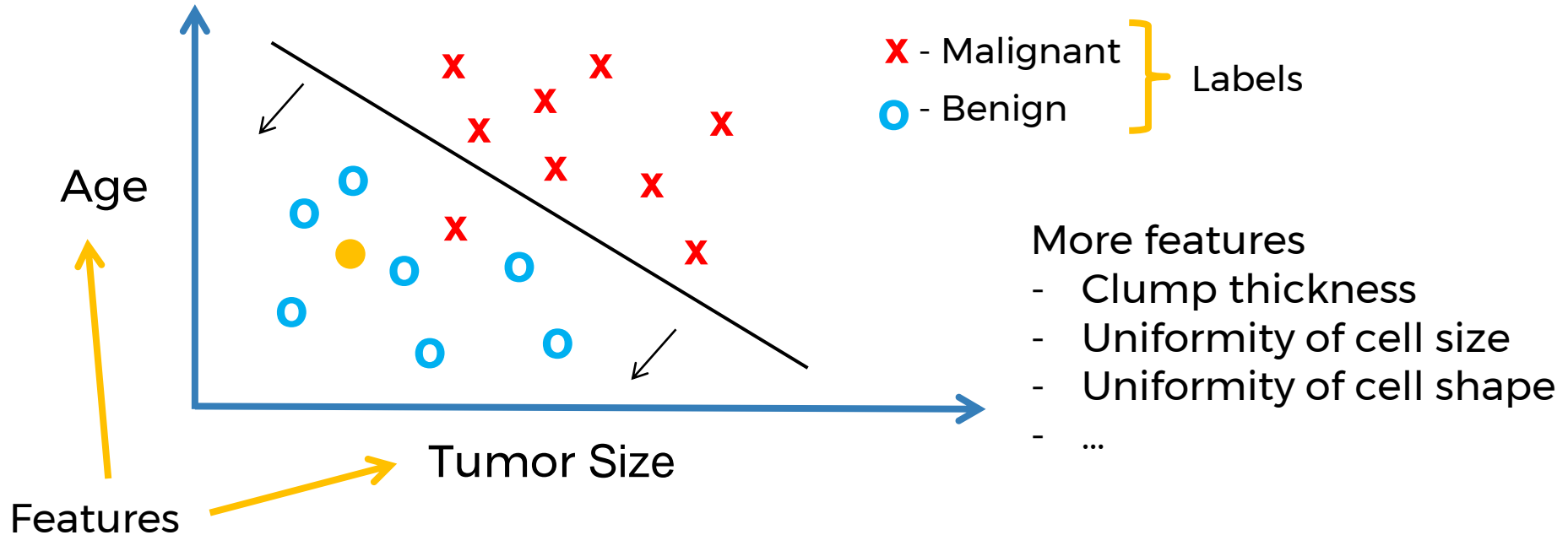
# Types of Machine Learning



# Supervised Learning

Labels are given.

Classification task: Breast Cancer (malignant or benign)

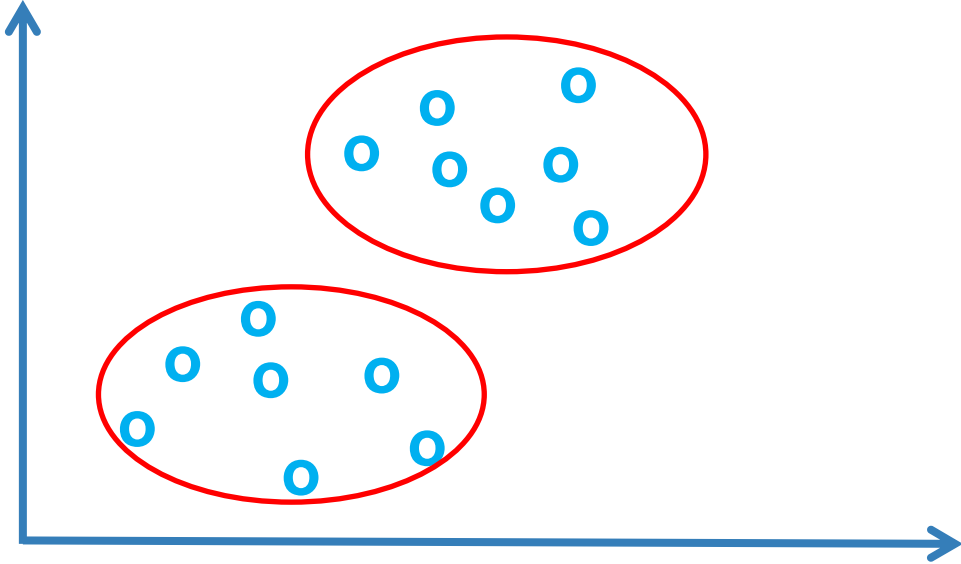




# Unsupervised Learning

Labels are **not** given.

Algorithm tries to find inherent structures within the data.



Examples:

- Market Segmentation (Cluster Customers)
- Social Network Analysis
- DNA Sequencing (group sequences into gene families)

# Applications of Machine Learning

- Database mining
  - Large datasets from growth of automation/web.
  - e.g., Web click data, medical records, biology, engineering
- Applications can't program by hand.
  - e.g., Autonomous helicopter, handwriting recognition, most of Natural Language Processing (NLP), Computer Vision.
- Self-customizing programs
  - E.g., Amazon, Netflix product recommendations

# Machines that can compose sonnets.

*When I in dreams behold thy fairest shade  
Whose shade in dreams doth wake the sleeping morn  
The daytime shadow of my love betray'd  
Lends hideous night to dreaming's faded form  
Were painted frowns to gild mere false rebuff  
Then should'st my heart be patient as the sands  
For nature's smile is ornament enough  
When thy gold lips unloose their drooping bands  
As clouds occlude the globe's enshrouded fears  
Which can by no astron'my be assail'd  
Thus, thyne appearance tears in atmospheres  
No fond perceptions nor no gaze unveils  
Disperse the clouds which banish light from thee  
For no tears be true, until we truly see*

# And Rap (DopeLearning)

The smell of success I want y'all to get a whiff of this  
Teaching us how to hate but does it in a way that we love it

Happy birthday baby

To sit with white folks in a meetin' pleasebaeleaveit

But if I pop the trunk its to hand you a rag

I had to jump in my car and call you a cab

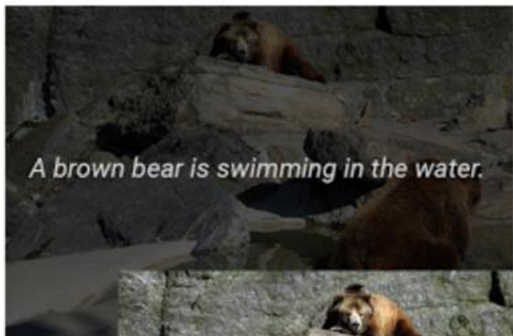
Least common denominator raise truth like Charlie Willy and Gramps

**Compose Music (Aiva)**

**Play Video Games.**

**Mi.Mu. Gloves  
(Gesture Recognition)**

# Google Image Captioning (94% accuracy)



A brown bear is swimming in the water.



Two brown bears sitting on top of rocks.



A train that is sitting on the tracks.



A blue and yellow train traveling down train tracks.

# Three Components of ML

1. Patterns exists
2. Cannot express the function mathematically
3. Data (lots of it!)

# The Machine Learning Process





# Dataset: Heart Disease

Class/  
Label/  
Target  
attribute

Features/ Attributes

Examples/instances

ID	Name	Age	Weight (kg)	Exercise	Smoker	Family History	Risk
1	Maria Lopez	24	55	Daily	False	False	Low
2	Jack Daniel	55	75	Rarely	True	True	High
3	Marie Clare	21	53	Weekly	False	False	Low
4	Alan Turin	31	60	Daily	True	True	High
5	John Newman	35	55	Weekly	False	False	Low
6	Lina Corazon	42	80	Rarely	False	True	High

Numerical

Nominal

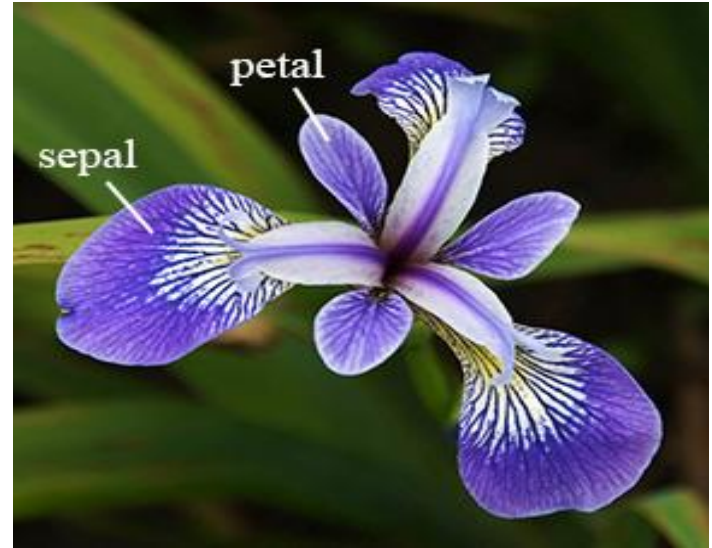
Binary

[illegible]

\_\_\_\_\_

# Exercise: Iris Plant Classification

- Download the dataset at: <http://bit.ly/2phqFfT>
- Each iris plant has 4 features/attributes:
  - Sepal Length
  - Sepal Width
  - Petal Length
  - Petal Width
- Classify iris plants into 3 species:
  - Iris Setosa
  - Iris Versicolour
  - Iris Virginica



# Let's get started...

## 1. Install Anaconda (Python)

Why Anaconda? It's prepackaged with many scientific libraries (scipy, numpy, scikit-learn, matplotlib)

## 2. Run a simple 'Hello World!'

- ▷ Create a Python file containing: `print("Hello World")`
- ▷ Run it in the Anaconda terminal.

### 3. Import libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas.tools.plotting import scatter_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
```

Data analysis, scientific  
computing tools

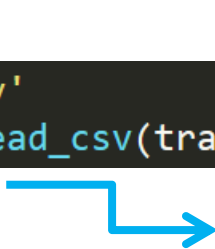
Data visualization

Machine learning  
algorithms

#### 4. Load data.

Specify that the first row of the dataset consists of the headers

```
train = 'iris.csv'  
dataframe = pd.read_csv(train, header=0)
```



Reads the data as a Dataframe

#### 5. It's always a good idea to inspect your data.

```
print(dataframe.shape)
```

Check dimensions of the Dataframe

```
print(dataframe.head(10))
```

Inspect the first 10 instances.

```
print(dataframe.groupby('Species').size())
```

Class distribution: the number of instances belonging to each class.

## 5. Extract attributes and class labels

```
attributes = dataframe.values[:, :4]  
labels = dataframe.values[:, 4]
```

→ Extract features/attributes of data instances

Index	0	1	2	3	4
[headers]	<b>Sepal Length</b>	<b>Sepal Width</b>	<b>Petal Length</b>	<b>Petal Width</b>	<b>Species</b>
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
...	...	...	...	...	...
149	5.9	3	5.1	1.8	Iris-virginica

## 5. Extract attributes and class labels

```
attributes = dataframe.values[:, :4]  
labels = dataframe.values[:, 4]
```

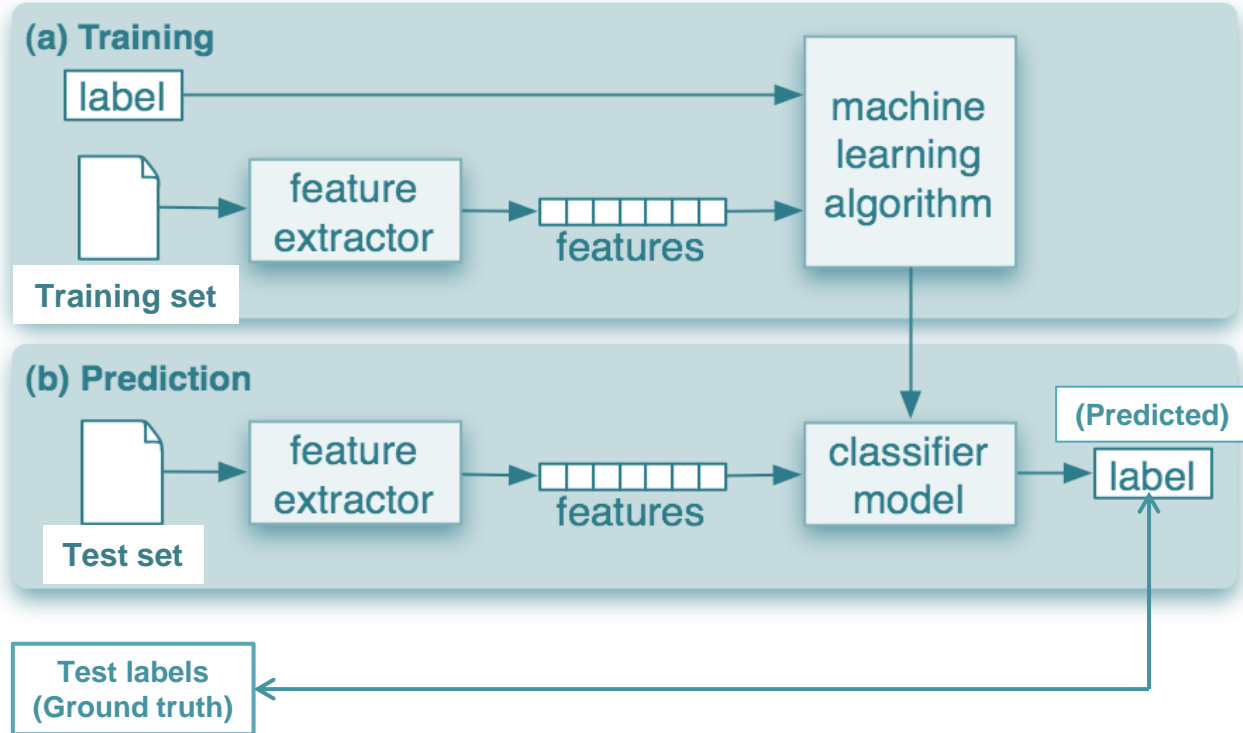


Extract classes/labels of instances

Index	0	1	2	3	4
[headers]	<b>Sepal Length</b>	<b>Sepal Width</b>	<b>Petal Length</b>	<b>Petal Width</b>	<b>Species</b>
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
...	...	...	...	...	...
151	5.9	3	5.1	1.8	Iris-virginica



# 2 Phases of Supervised Learning



6. We need to split the dataset into (random) training and test sets.

- ▷ Note that the size of the training set  $\gg$  test set
- ▷ Example: Training set is 80%, test set is 20%.
- ▷ Luckily, there's a built in function that does just that.

```
train_size = 0.8
seed = 7
train_attributes, test_attributes, train_labels, test_labels \
    = train_test_split(attributes, labels, train_size=train_size, random_state=seed)
```

## 7. **Training:** Building the classifier model.

- ▷ Choose a machine learning algorithm
  - ▷ Gaussian Naïve Bayes
  - ▷ Decision Trees
  - ▷ etc.
- ▷ Train the classifier model on the training set.

```
clf = GaussianNB()  
clf.fit(train_attributes, train_labels)
```

## 8. **Testing:** Evaluate the classifier model on the test set.

- ▷ Predict the class labels ('Species') of the test set.

```
# Predict labels for test set  
predicted_labels = clf.predict(test_attributes)
```

- ▷ You can compare the predicted labels with the true labels ("ground truth")

```
# Compare results  
results = pd.DataFrame({'Predicted label': predicted_labels, 'True label': test_labels})  
print(results)
```

## 9. **Evaluate accuracy.** How well did our classifier model perform?

```
# Evaluate results
accuracy = np.mean(predicted_labels == test_labels)
print("Gaussian Naive Bayes: " + str(accuracy))
```

$$\text{accuracy} = \frac{\text{number of correct predicted labels}}{\text{total number of test labels}}$$

The closer the accuracy is to 100%, the better.

Tip: Steps 8 and 9 can be done with a single line of code:

```
# Or, more concisely:
accuracy = clf.score(test_attributes, test_labels)
print("Gaussian Naive Bayes: " + str(accuracy))
```

# Great! 83% is good, but can we do better?

Try a different classifier. For example,

```
clf = SVC()
```

Or, you can store different classifiers into a dictionary.

```
classifiers = dict()
classifiers['Gaussian Naive Bayes'] = GaussianNB()
classifiers['Decision Tree Classifier'] = DecisionTreeClassifier(random_state=seed)
classifiers['Support Vector Machines'] = SVC()
```

Then iterate over this dictionary:

```
for clf_name, clf in classifiers.items():
    clf.fit(train_attributes, train_labels)
    accuracy = clf.score(test_attributes, test_labels)
    print(clf_name + ': ' + str(accuracy))
```

# Present your results! 😊

Try searching the net for other classifiers and apply them to the problem:

- Multi-layer Perceptron (MLP)
- Random Forests
- Adaboost
- Neural Networks
- k-Nearest Neighbour
- etc.

# Assignment: Computer Vision – Handwritten Digit Recognition

- Download the dataset: <http://bit.ly/2os5SHm>
- The data files mnist.csv contains grey-scale images of hand-drawn digits, from 0 through 9.
- Each image is 28 x 28 pixels, for a total of 784 pixels.
- Each pixel has a single pixel-value, indicating the lightness or darkness of that pixel, with higher numbers meaning darker.
  - Each pixel-value is an integer between 0 and 255, inclusive.





# Assignment: Computer Vision – Handwritten Digit Recognition

- **Features/attributes:** 784 pixels values
- **Classes:** Labels 0-9 (first column of the dataset)
- **Exercise:** Try using different classifiers to predict the labels of a test set generated using mnist.csv.
  - You can use our code from iris plant classification as a template

**Tip:** After loading the dataset, you can visualize an instance  $i \in [0, 42000]$  using the following code snippet:

```
i = 9
image_data = dataframe.iloc[:, 1:]
img = image_data.iloc[i].as_matrix()
img = img.reshape((28, 28))
plt.imshow(img, cmap='gray_r')
plt.title('Class label: ' + str(dataframe.iloc[i, 0]))
plt.show()
```

# Future sessions

- Linear Algebra Review: Vectors, Matrices, Vectorized computation
- Under the Hood: ML Algorithms (Logistic Regression, Decision Trees, Naïve Bayes, etc.)
- Implementing ML Algorithms from scratch (Spam Filter)
- **Note:** Try practicing on datasets available on Kaggle, UCI Machine Learning Repository, or on data you might have on hand
- Suggestions, comments?

# Survey Form

Please answer the following form:

<http://bit.ly/2qzdys6>

# References

<https://www.kaggle.com/c/digit-recognizer/data>

<http://machinelearningmastery.com/>

<https://www.coursera.org/learn/machine-learning>



# T.I.L.

**SHARE IT!**  
**In front!**

On Twitter: @wwcodemanila  
Or FB: fb.com/wwcodemanila

Don't forget to tag WWCodeManila so we can retweet or share it.

WOMEN WHO

**THANK YOU :)**

CODE®