CHULA ENGINEERING
Foundation toward Innovation
COMPUTER

Machine Learning

Chula Big Data and IoT
Center of Excellence
(CUBIC)

Data Mining Group
Machine Intelligence and Knowledge Discovery Lab
Chulalongkorn University

Asst. Prof. Peerapon Vateekul, Ph.D.
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
Peerapon.v@chula.ac.th
www.cp.eng.chula.ac.th/~peerapon/

Outlines

- Introduction
- Supervised Learning
- Unsupervised Learning
- Extra Topic (Not Cover)
  - Association Rule Mining
  - Recommender System
- Demo
- Become Data Scientist and Build Your Team

Introduction

Retail's Adapt-Or-Die Moment: How Artificial Intelligence Is Reshaping Commerce

https://www.cbinsights.com/research/artificial-intelligence-reshaping-commerce/

July 18, 2019

Artificial Intelligence Retail

WHERE IS THIS DATA COMING FROM?

Start your free trial today

Email

SIGN UP

Traditional and new-school retailers alike are using AI and robotics to automate various parts of the retail chain, from manufacturing to last-mile delivery.

Retail is under pressure to crack the AI code.

After all, corporations in every industry are scrambling to adapt and integrate artificial intelligence into their products — and retail is no exception. Between Q1'13 and Q2'19, retail AI startups looking to address retailers' AI needs raised a total of $2.67B across 564 deals.

2019

January	February	March
GYMBOREE, IMS, SHOPKO, BEAUTY BRANDS	charlotte russe, Payless, FULLBEAUTY BRANDS, THINGS REMEMBERED	DIESEL

2018

January	February	March	April	May	June
KIKO, A'GACI	CHARLOTTE OLYMPIA, BON-TON	Remington, Southeastern Grocers, THE WALKING COMPANY, claire's	NINE WEST	ROCKPORT	

July	August	September	October	November	December
	FALLAS, GUMP'S, Brookstone, SAMUELS JEWELERS		sears, MATTRESS FIRM	DAVID'S BRIDAL	

2017

January	February	March	April	May	June
THE LIMITED	Wet Seal, BCBGMAXAZRIA	hhgregg, Vanity, RadioShack, GANDER MTN, Gordmans	Payless	rue21	GYMBOREE OUTLET, Papaya

July	August	September	October	November	December
TRUE RELIGION, alfred angelo	PERFUMANIA	VITAMIN WORLD, AEROSOLES, TOYS R US		STYLES	charming charlie

CBINSIGHTS

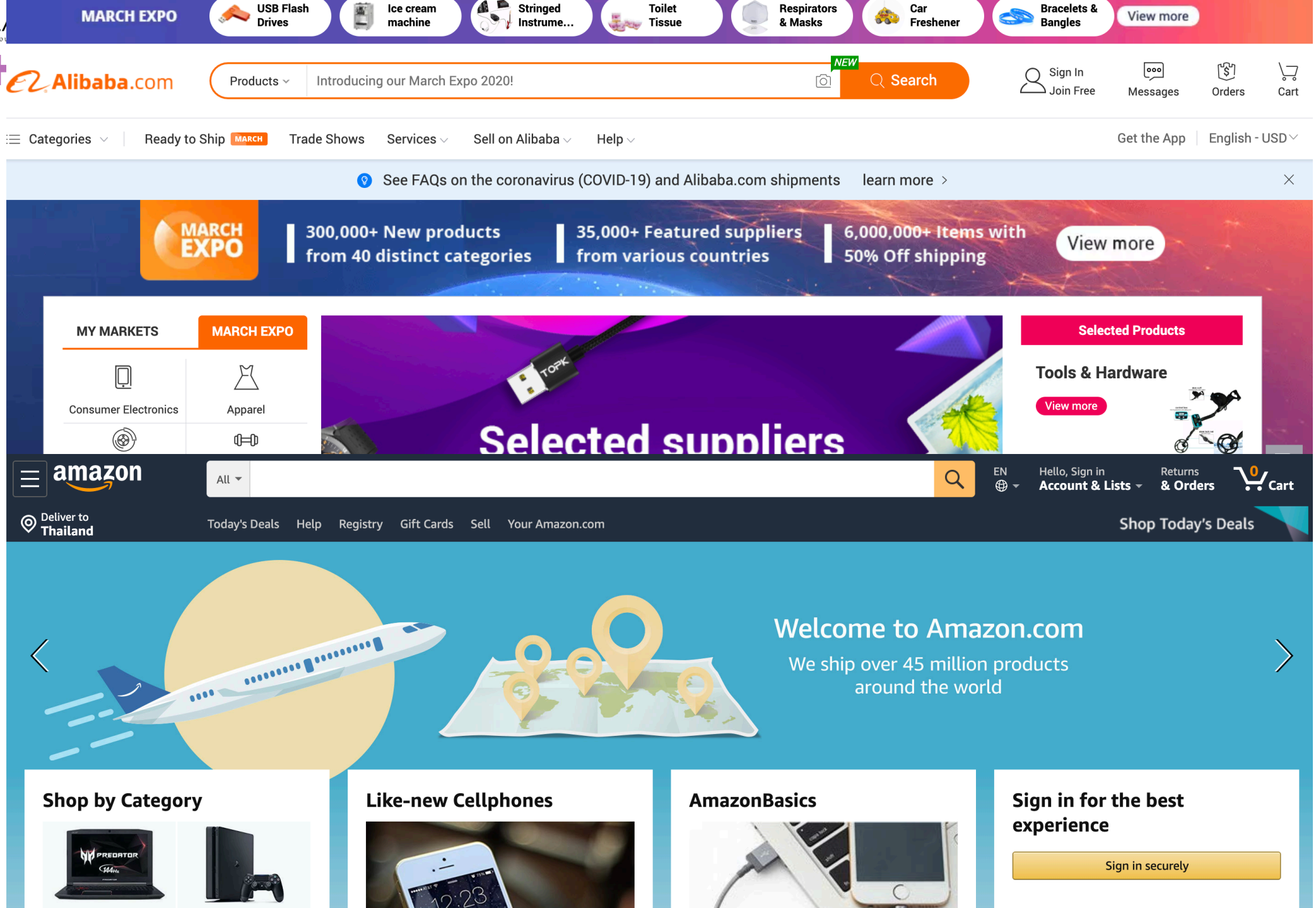
2019

January	February	March
GYMBOREE, IMS, SHOPKO, BEAUTY BRANDS	charlotte russe, Payless, FULLBEAUTY BRANDS, THINGS REMEMBERED	DIESEL

Bankruptcy!

Consumer channels: selling online vs. in-store

personalized experience & direct targeted marketing

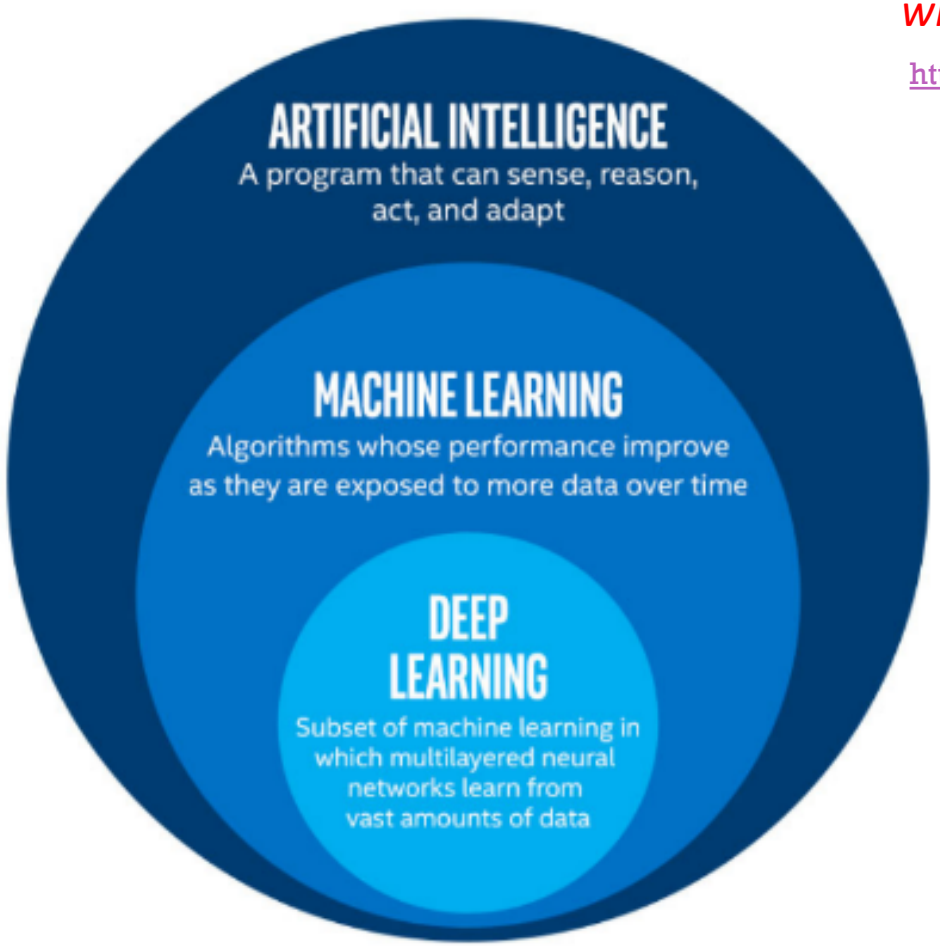

MARCH EXPO
USB Flash Drives
Ice cream machine
Stringed Instrume...
Toilet Tissue
Respirators & Masks
Car Freshener
Bracelets & Bangles
View more
Alibaba.com
Products
Introducing our March Expo 2020!
NEW
Search
Sign In
Join Free
Messages
Orders
Cart
Categories
Ready to Ship MARCH
Trade Shows
Services
Sell on Alibaba
Help
Get the App
English - USD
See FAQs on the coronavirus (COVID-19) and Alibaba.com shipments
learn more
MARCH EXPO
300,000+ New products from 40 distinct categories
35,000+ Featured suppliers from various countries
6,000,000+ Items with 50% Off shipping
View more
MY MARKETS
MARCH EXPO
Consumer Electronics
Apparel
Selected suppliers
Selected Products
Tools & Hardware
View more
amazon
All
EN
Hello, Sign in
Account & Lists
Returns
& Orders
Cart
Deliver to
Thailand
Today's Deals
Help
Registry
Gift Cards
Sell
Your Amazon.com
Shop Today's Deals
Welcome to Amazon.com
We ship over 45 million products around the world
Shop by Category
Like-new Cellphones
AmazonBasics
Sign in for the best experience
Sign in securely

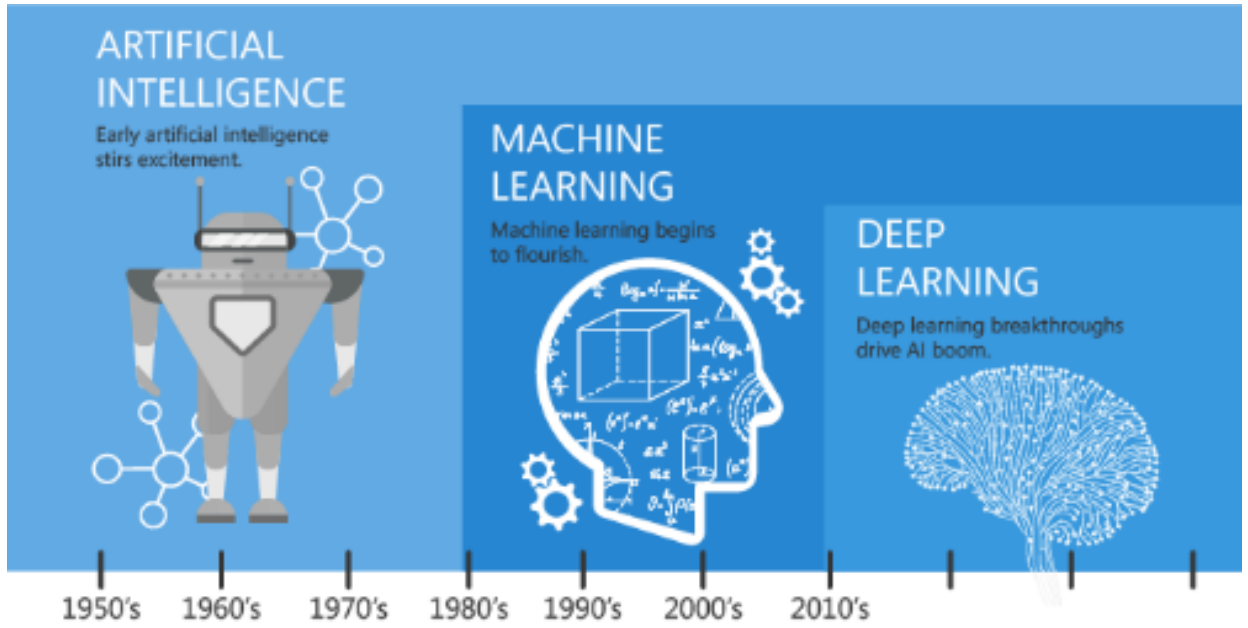
AI, Machine Learning, and Deep Learning

- Machine Learning (ML) is a subfield in AI focusing on making to learn by itself without human intervention.

"Machine learning is the *science of getting computers to act without being explicitly programmed*." — Stanford University

https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55

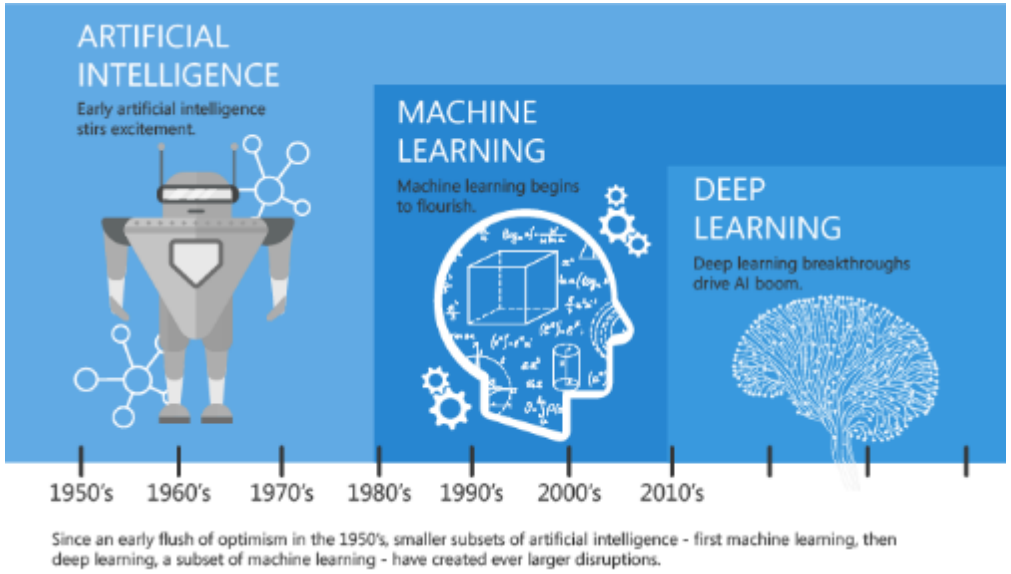


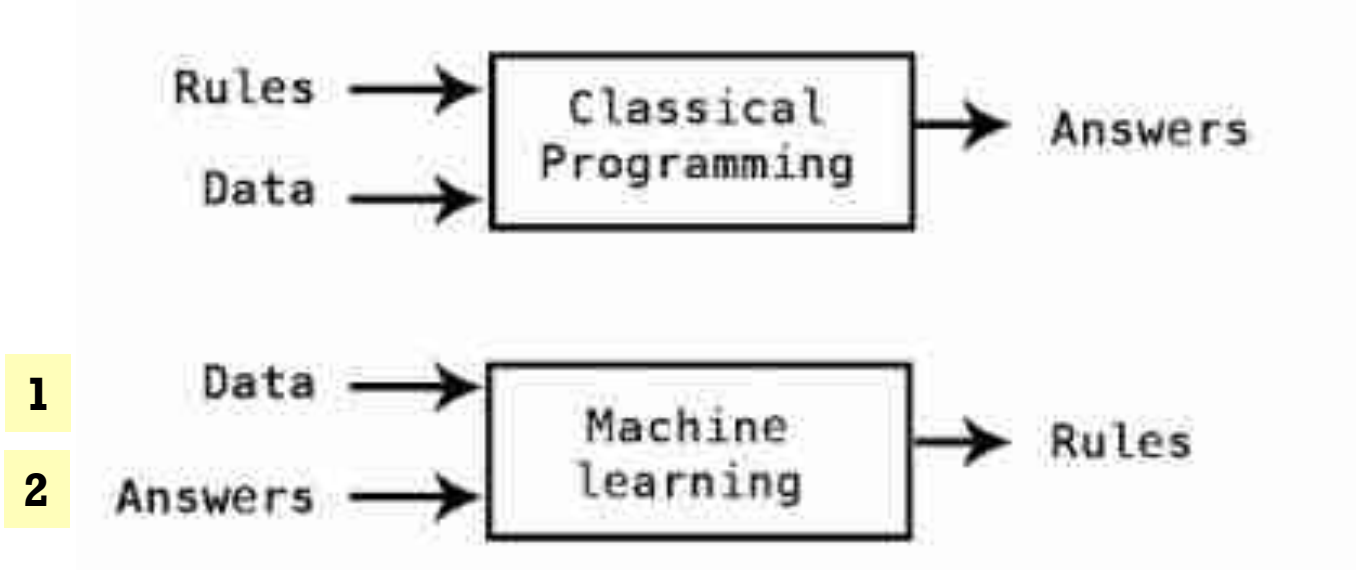


Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

AI = Automation

- 1) Rule-based AI (Symbolic AI)
- 2) Machine Learning





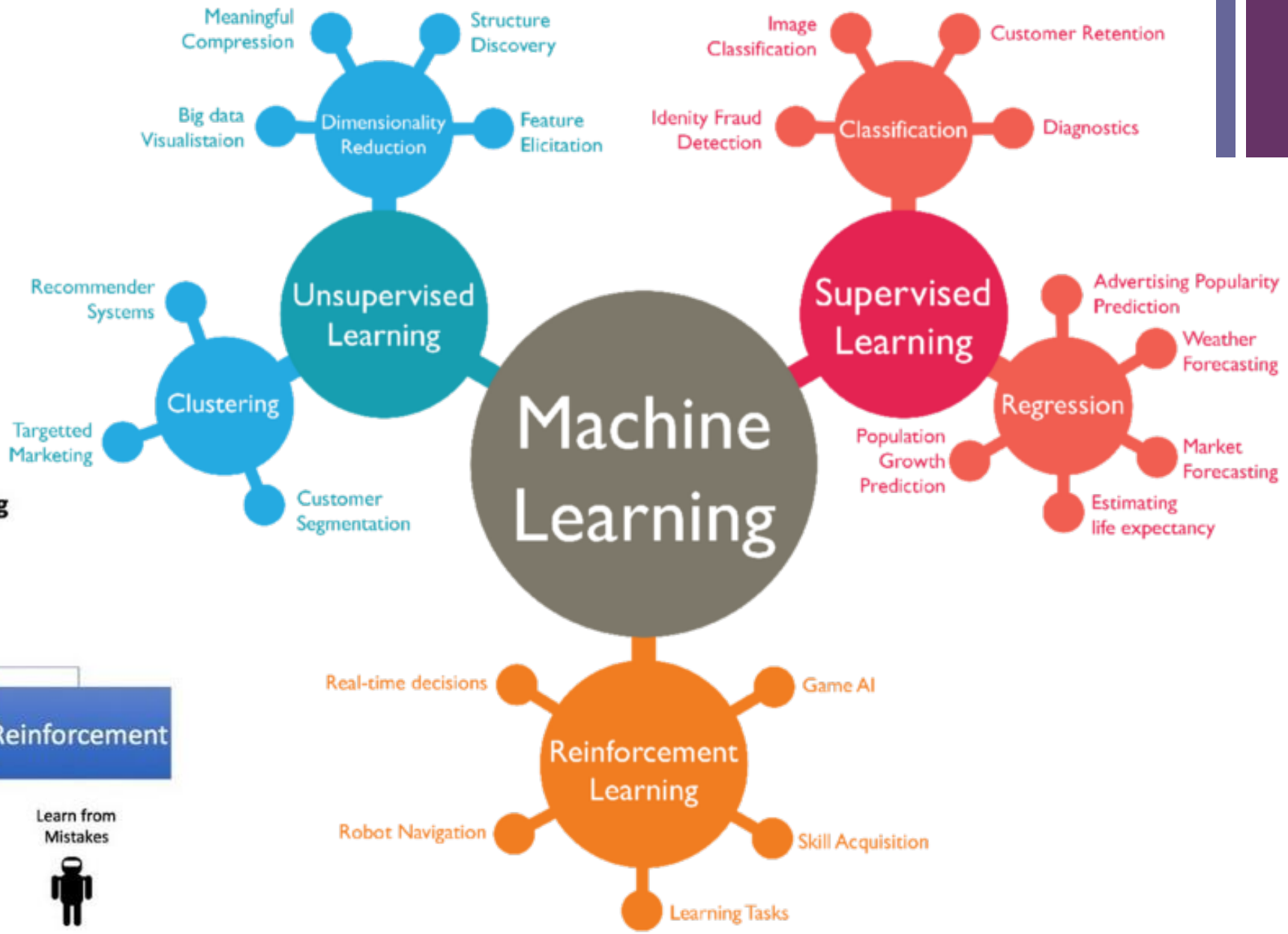
https://mc.ai/machine-learning-basics-artificial-intelligence-machine-learning-and-deep-learning/

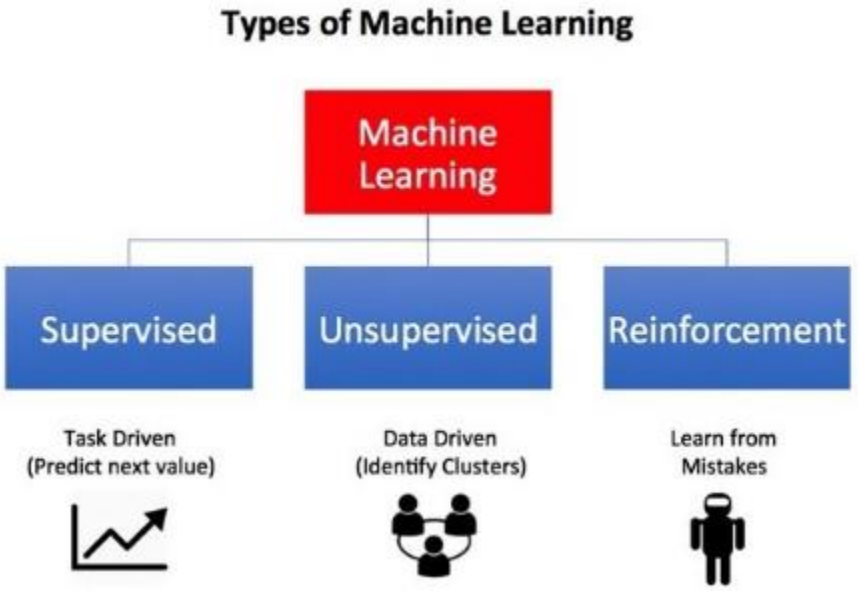
Sample of ML Application: Finding Waldo

PROTOTYPE v0.1
redpepper.land/innovation

rp

Machine Learning (cont.)





scikit-learn

Machine Learning in Python

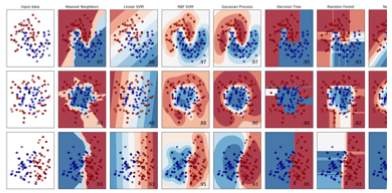
Getting Started | What's New in 0.22.2 | GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

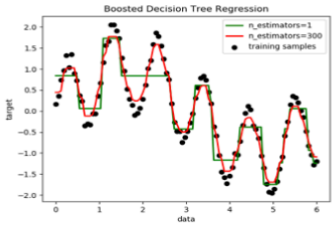


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

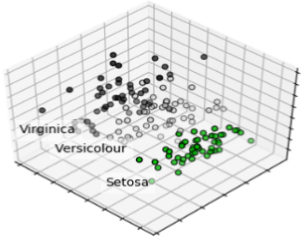
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...

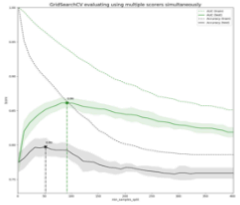


Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...

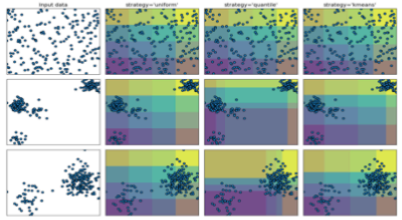


Examples

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



Examples

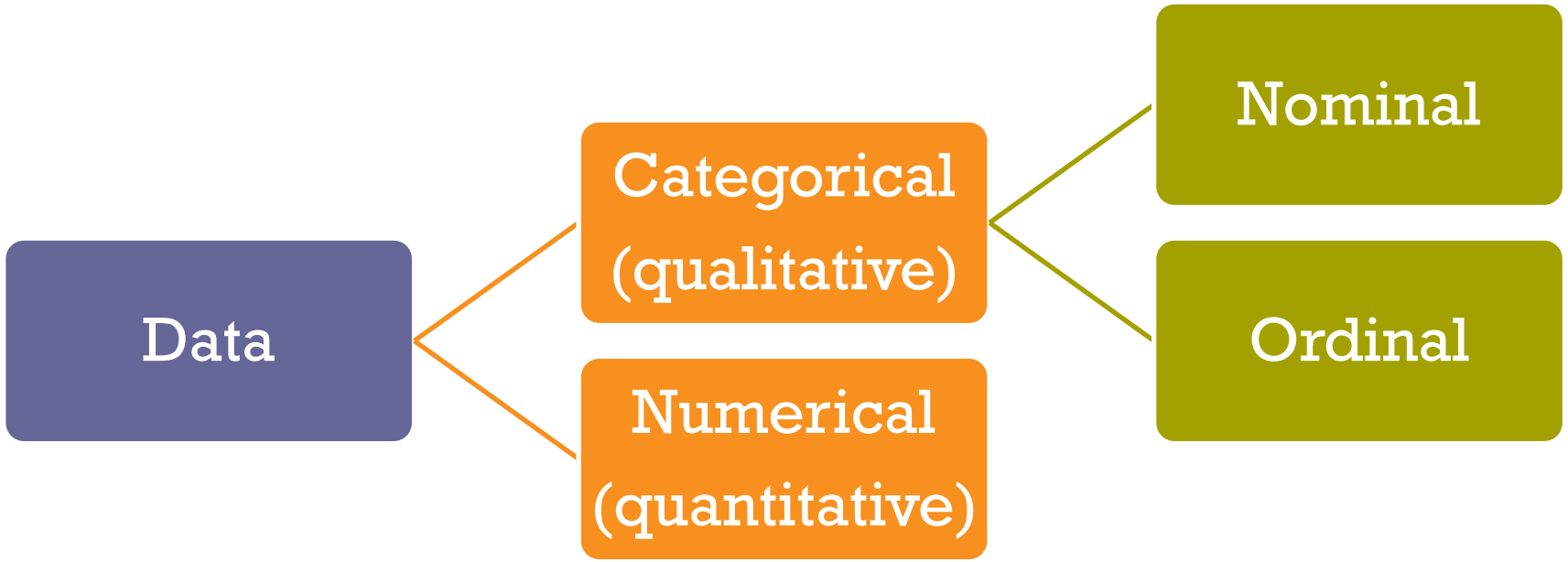
Terminology: Data table

inputs | target

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No

- Row
  - Example, instance, case, observation, subject
- Column
  - Feature, variable, attribute
- Input
  - Predictor, independent, explanatory variable
- Target
  - Output, outcome, response, dependent variable

Terminology: Kinds of data

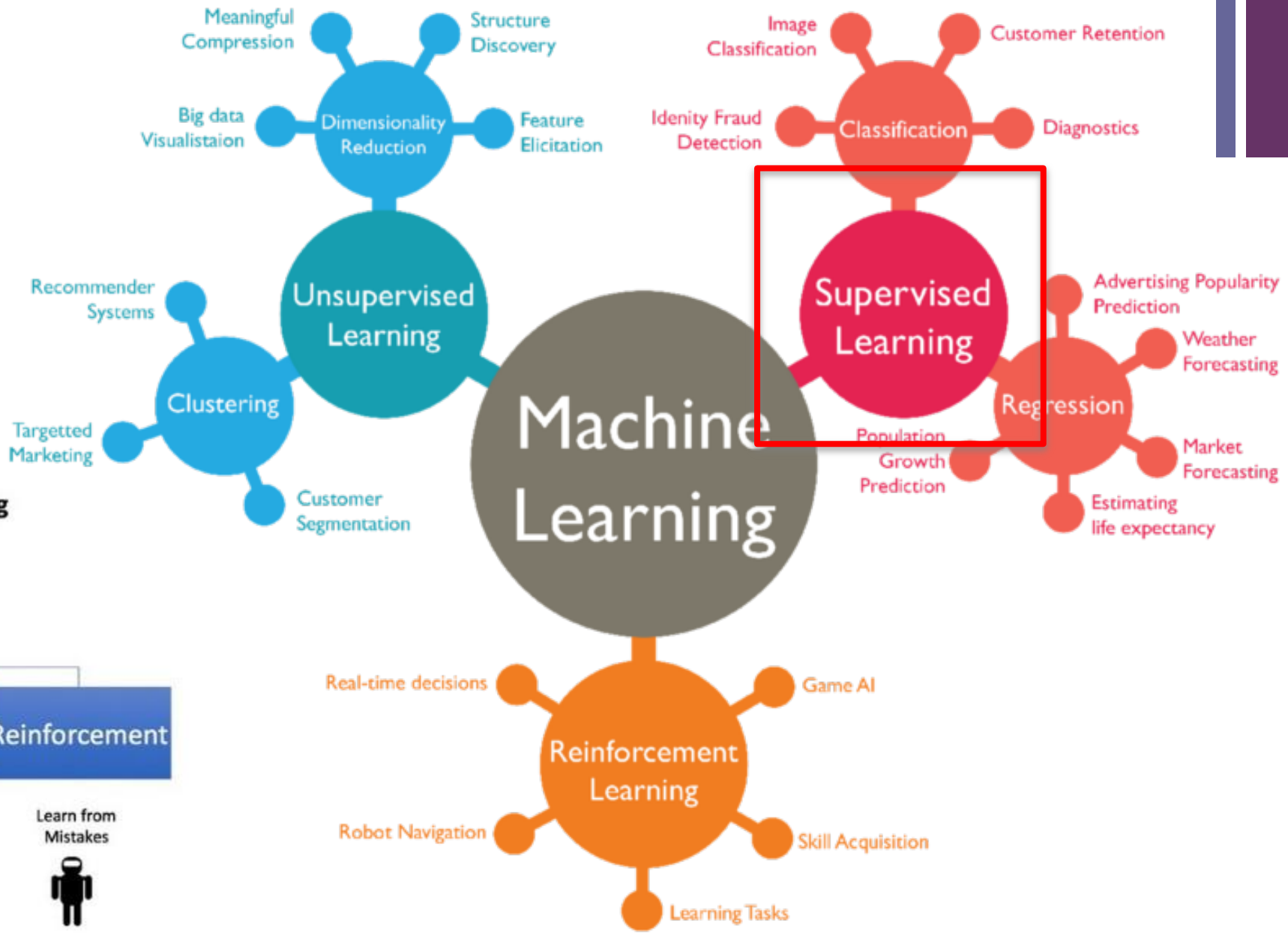


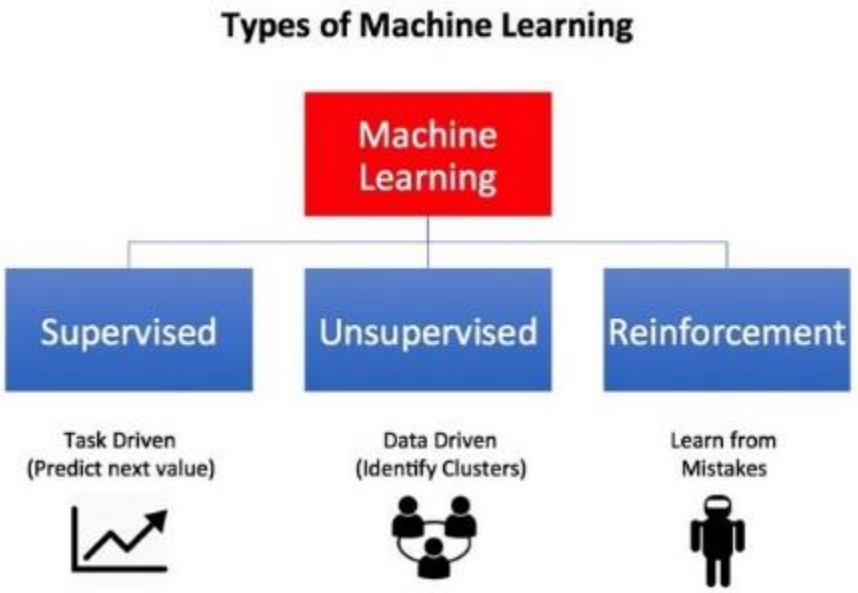
Categories cannot be compared

Categories with an implied order

Supervised Learning (Predictive Task)

Machine Learning (cont.)





Task1: Supervised learning

Handcrafted features

Training Data

inputs | target

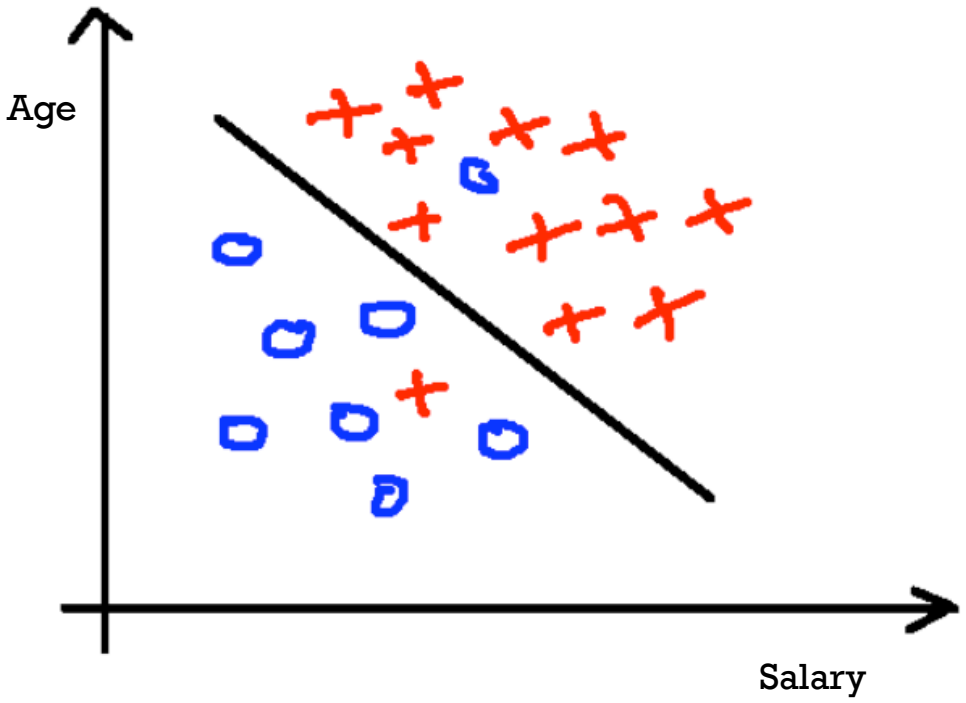
Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No

Testing Data

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	?

Application: Direct Target Marketing

Classification: predicting a category
Logistic Regression

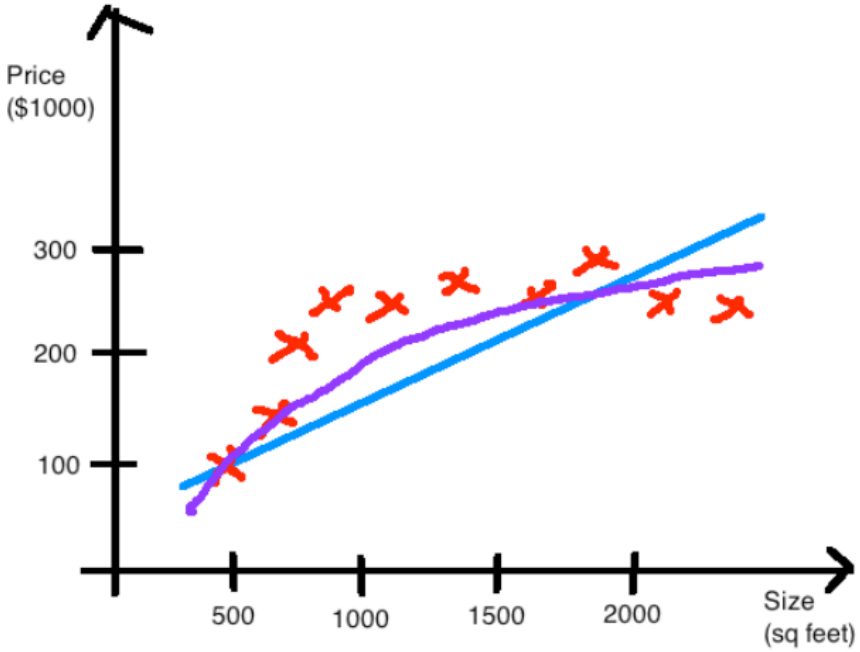


Predict targeted customers who tend to buy our product (yes/no)

- **Some techniques:**
  - Naïve Bayes
  - Decision Tree
  - Logistic Regression
  - Support Vector Machines
  - Neural Network
  - Ensembles
- **Sample Applications**
  - Database marketing
  - Fraud detection
  - Pattern detection
  - Churn customer detection

Regression: predict a continuous value

Linear Regression



Predict a sale price of each house

- **Some techniques:**
  - Linear Regression / GLM
  - Decision Trees
  - Support vector regression
  - Neural Network
  - Ensembles
- **Sample Applications**
  - Financial risk management
  - Revenue forecasting

Scikit-learn: Machine learning library in Python

- Provides many machine learning tools with a common Estimator interface
- Built in helpers for common ML tasks (e.g., metrics, preprocessing)
- Easily combine algorithms to make a complex pipeline
- Relies heavily on numpy and scipy, often used with pandas

How do you pronounce the project name?

sy-kit learn. sci stands for science!

Why scikit?

There are multiple scikits, which are scientific toolboxes built around SciPy. You can find a list at https://scikits.appspot.com/scikits. Apart from scikit-learn, another popular one is scikit-image.

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

http://scikit-learn.org/stable/index.html

CHULA ENGINEERING COMPUTER
Foundation toward Innovation

Estimator Interface

Decision Trees

We'll start just by training a single decision tree.

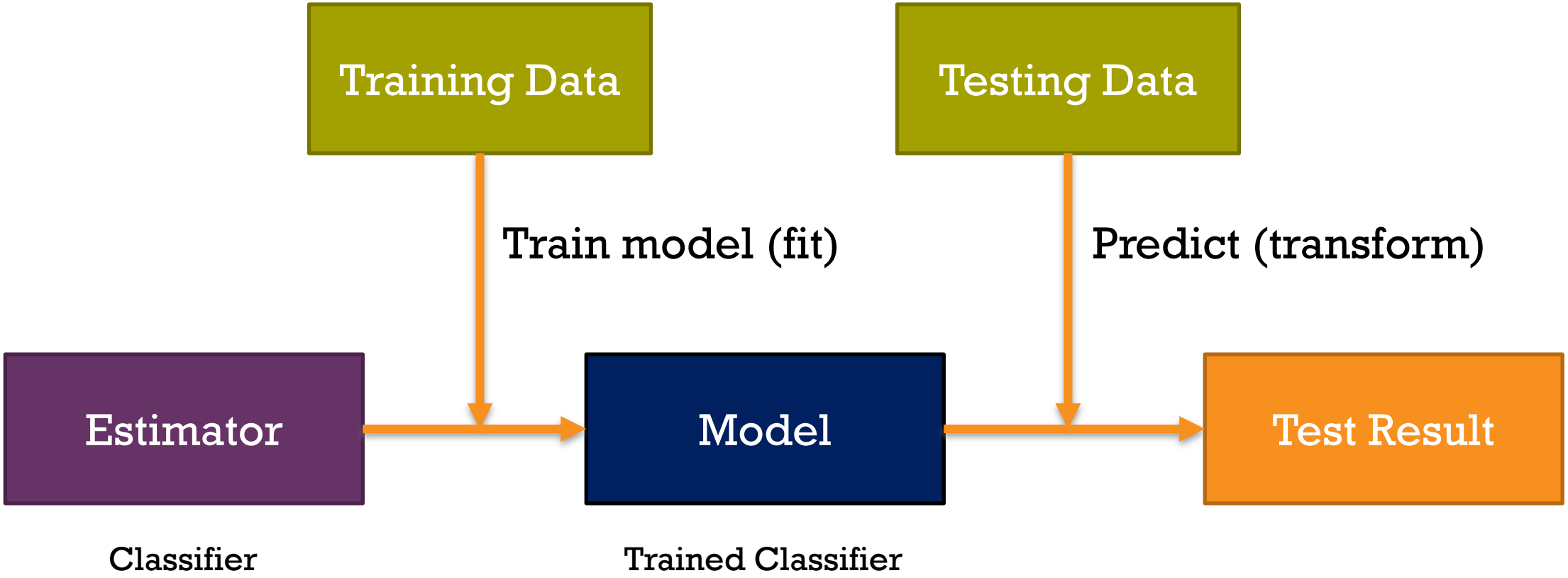
```
In [8]: from sklearn.tree import DecisionTreeClassifier

In [9]: dtree = DecisionTreeClassifier(min_samples_leaf=10, criterion='entropy')

In [10]: dtree.fit(X_train,y_train)

Out[10]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
             max_features=None, max_leaf_nodes=None,
             min_impurity_decrease=0.0, min_impurity_split=None,
             min_samples_leaf=10, min_samples_split=2,
             min_weight_fraction_leaf=0.0, presort=False, random_state=None,
             splitter='best')
```

Prediction and Evaluation

Let's evaluate our decision tree.

```
[11]: predictions = dtree.predict(X_test)

[12]: from sklearn.metrics import classification_report,confusion_matrix

[13]: print(classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
absent	0.85	0.85	0.85	20
present	0.40	0.40	0.40	5
avg / total	0.76	0.76	0.76	25



Training Phase

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

cancer = load_breast_cancer()     # Get some data
X_train, X_test, y_train, y_test = train_test_split(
        cancer.data, cancer.target,
        stratify=cancer.target, random_state=1337)

tree = DecisionTreeClassifier(random_state=7331)
tree.fit(X_train, y_train)  # Learn a Decision Function
```

Ref: C. Travis Johnson, Mines Linux Users Group on April 27, 2017

Testing Phase (Prediction)

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

```
>>> from sklearn.metrics import classification_report
>>> y_true = [0, 1, 2, 2, 2]
>>> y_pred = [0, 0, 2, 2, 1]
>>> target_names = ['class 0', 'class 1', 'class 2']
>>> print(classification_report(y_true, y_pred, target_names=target_names))
              precision    recall  f1-score   support

     class 0       0.50      1.00      0.67         1
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.67      0.80         3

   micro avg       0.60      0.60      0.60         5
   macro avg       0.50      0.56      0.49         5
weighted avg       0.70      0.60      0.61         5
```

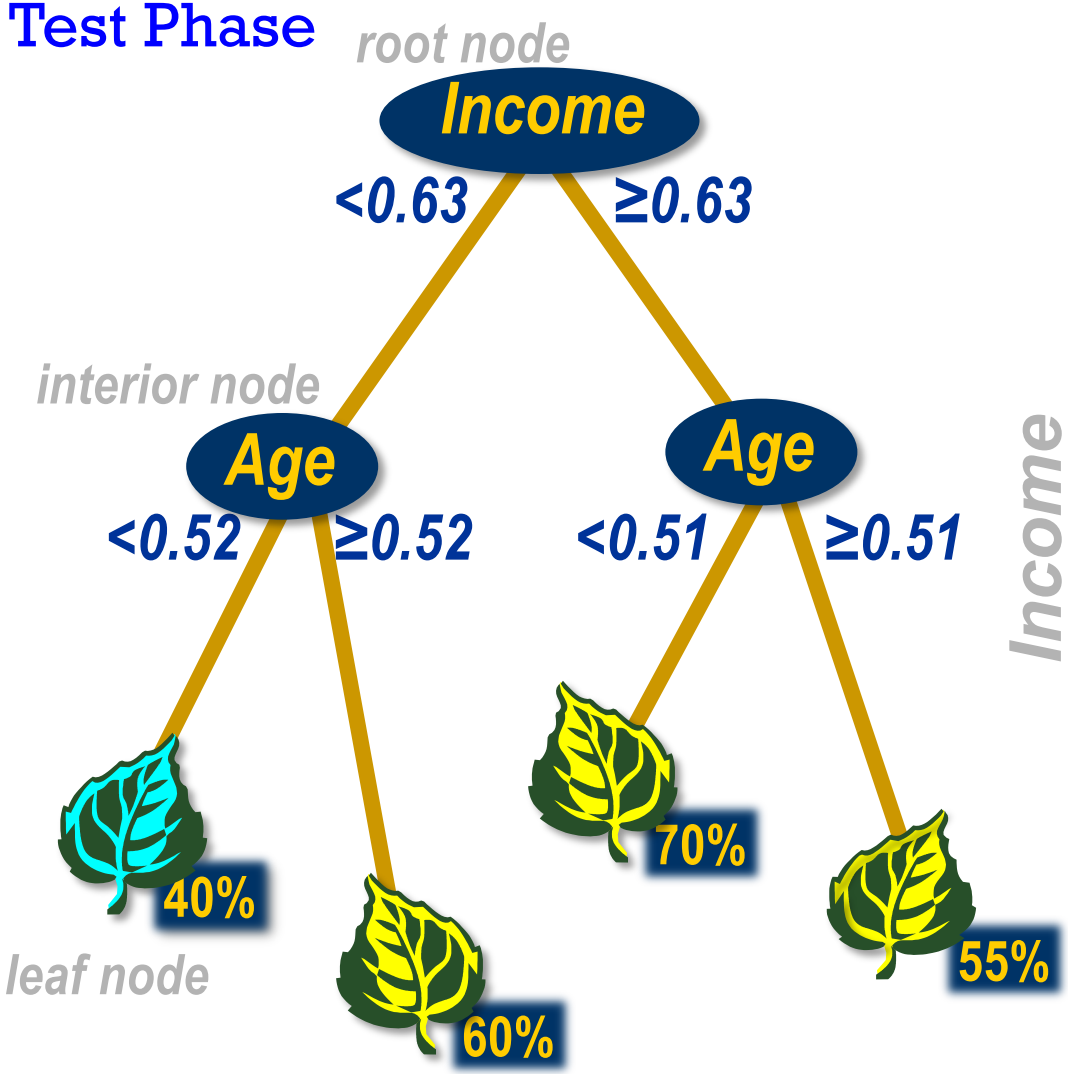
Prediction Algorithms

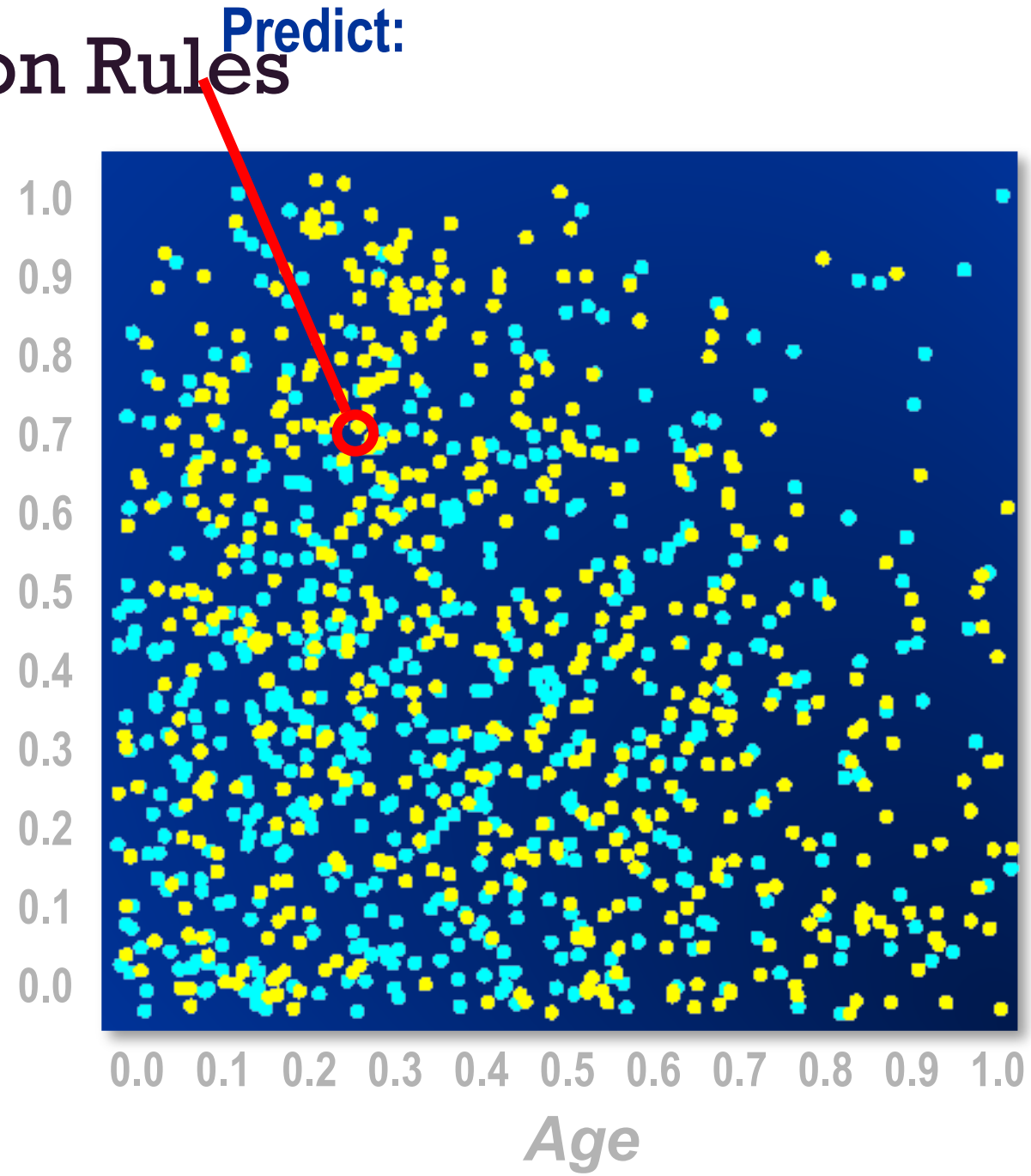
- Decision Tree
- (Logistic) Regression
- Neural Networks (NN)

- kNN
- Support Vector Machine
- Deep Learning

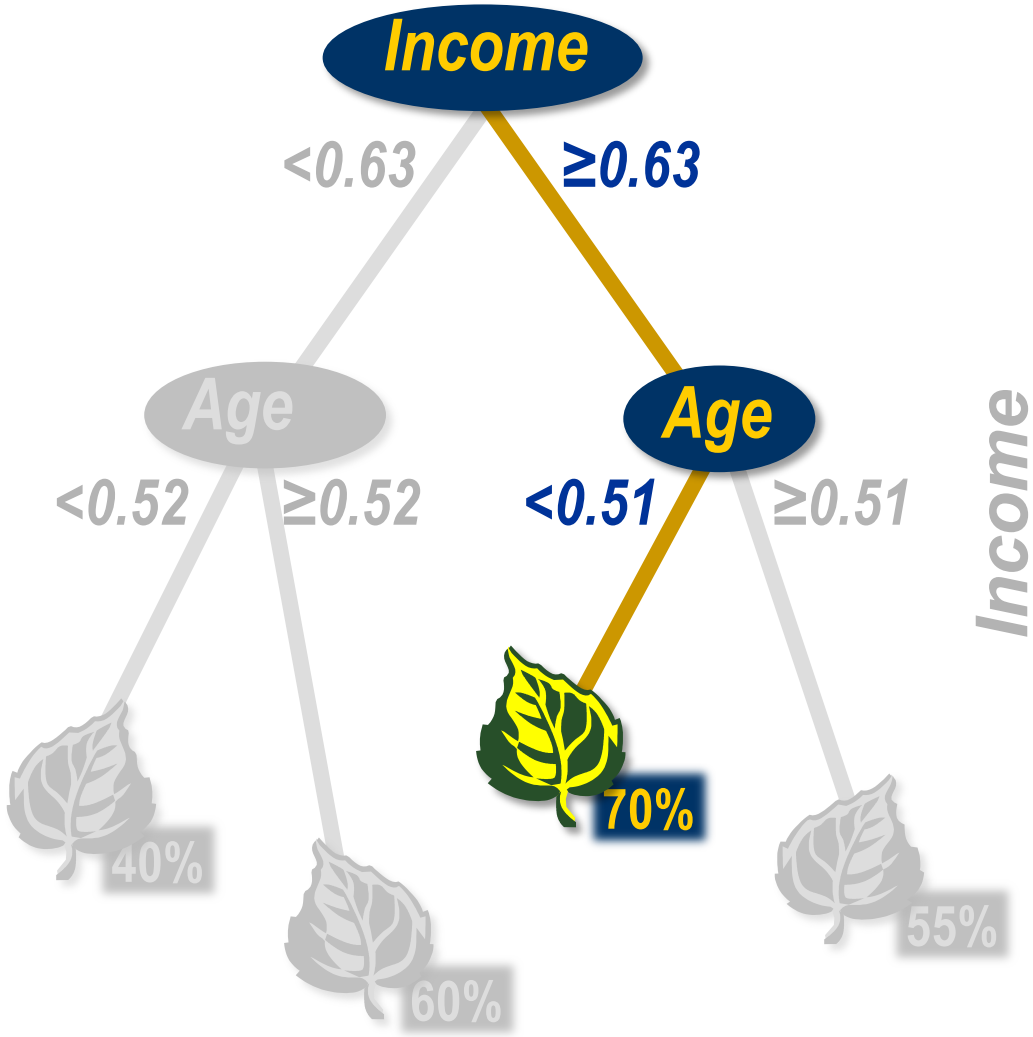
1) Decision Tree Prediction Rules

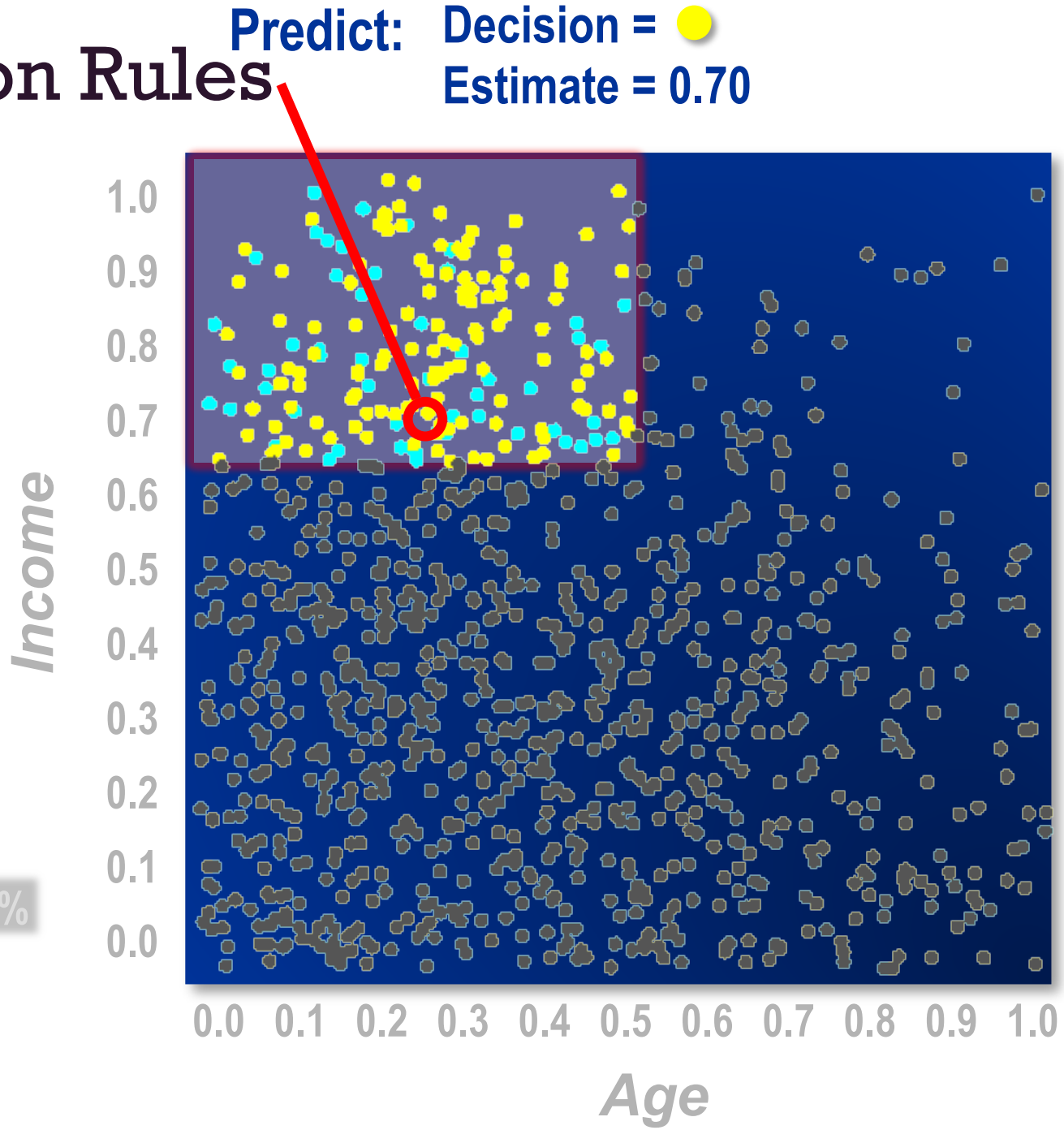
Test Phase





Decision Tree Prediction Rules





Types of Decision Tree

Algorithm	Splitting Measure
ID3	Entropy
C4.5	Gain Ratio
CART	Gini index
CHAID	Chi-squared test

Important Parameters in Decision Tree

- Minimum leaf size
- Minimum threshold
- Maximum depth
- Splitting measure
- Pruning adjustment

Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

sklearn.tree.DecisionTreeClassifier
Examples using sklearn.tree.DecisionTreeClassi

sklearn.tree.DecisionTreeClassifier

class sklearn.tree.**DecisionTreeClassifier**(*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort='deprecated', ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entropy"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

sklearn.tree.DecisionTreeRegressor
Examples using sklearn.tree.DecisionTreeRegres

sklearn.tree.DecisionTreeRegressor

class sklearn.tree.**DecisionTreeRegressor**(*criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, presort='deprecated', ccp_alpha=0.0*) [source]

A decision tree regressor.

Read more in the User Guide.

Parameters:

criterion : *{"mse", "friedman_mse", "mae"}, default="mse"*

The function to measure the quality of a split. Supported criteria are "mse" for the mean squared error, which is equal to variance reduction as feature selection criterion and minimizes the L2 loss using the mean of each terminal node, "friedman_mse", which uses mean squared error with Friedman's improvement score for potential splits, and "mae" for the mean absolute error, which minimizes the L1 loss using the median of each terminal node.

New in version 0.18: Mean Absolute Error (MAE) criterion.

splitter : *{"best", "random"}, default="best"*

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

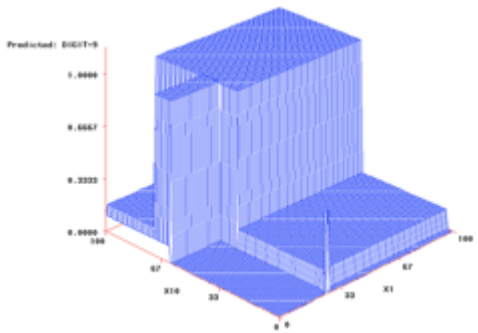
max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

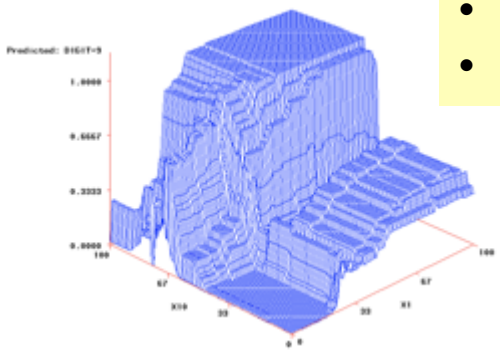
Random Forest Algorithm (RF)

- Forest algorithm samples the rows and the columns at each step.
- Forest takes bootstrap samples of the rows in training data (sampling with replacement)
- At each step, a set of variables (columns) is sampled.
- This increases variation among trees in the **ensemble** often leads to improved prediction accuracy.

Important Params:
- #Tree
- #Columns (variables)
- #Rows (examples)



Single Tree



Random Forest

Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier
3.2.4.3.1.1. Examples using sklearn.ensemble.RandomForestCl

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier

class `sklearn.ensemble.`**`RandomForestClassifier`**(*n_estimators=100, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

Read more in the User Guide.

Parameters:

n_estimators : *integer, optional (default=100)*

The number of trees in the forest.

> *Changed in version 0.22:* The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *string, optional (default="gini")*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

3.2.4.3.2. sklearn.ensemble.RandomForestRegressor
3.2.4.3.2.1. Examples using sklearn.ensemble.RandomForestRe

3.2.4.3.2. sklearn.ensemble.RandomForestRegressor

class `sklearn.ensemble.`**`RandomForestRegressor`**(*n_estimators=100, criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None*) ¶ [source]

A random forest regressor.

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

Read more in the User Guide.

Parameters:

n_estimators : *integer, optional (default=10)*

The number of trees in the forest.

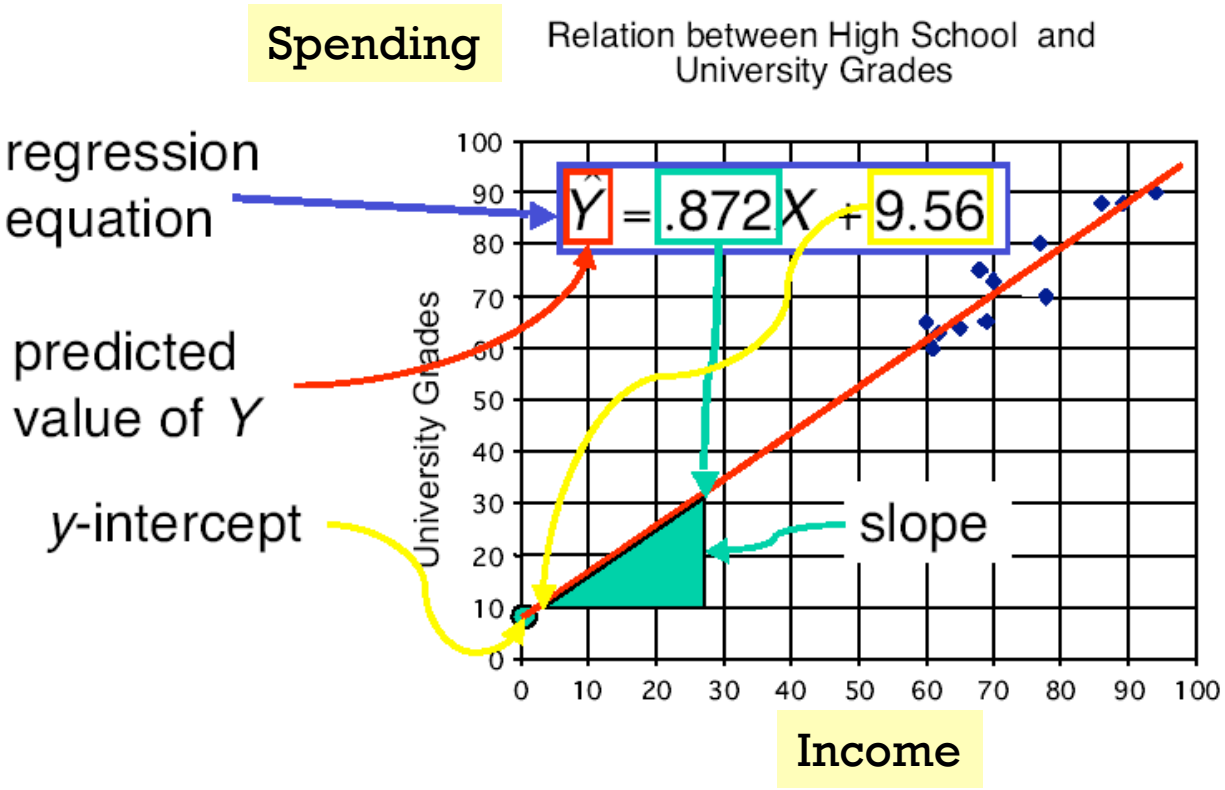
> *Changed in version 0.22:* The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *string, optional (default="mse")*

The function to measure the quality of a split. Supported criteria are "mse" for the mean squared error, which is equal to variance reduction as feature selection criterion, and "mae" for the mean absolute error.

New in version 0.18: Mean Absolute Error (MAE) criterion.

2) Regression



weight, coefficient

$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

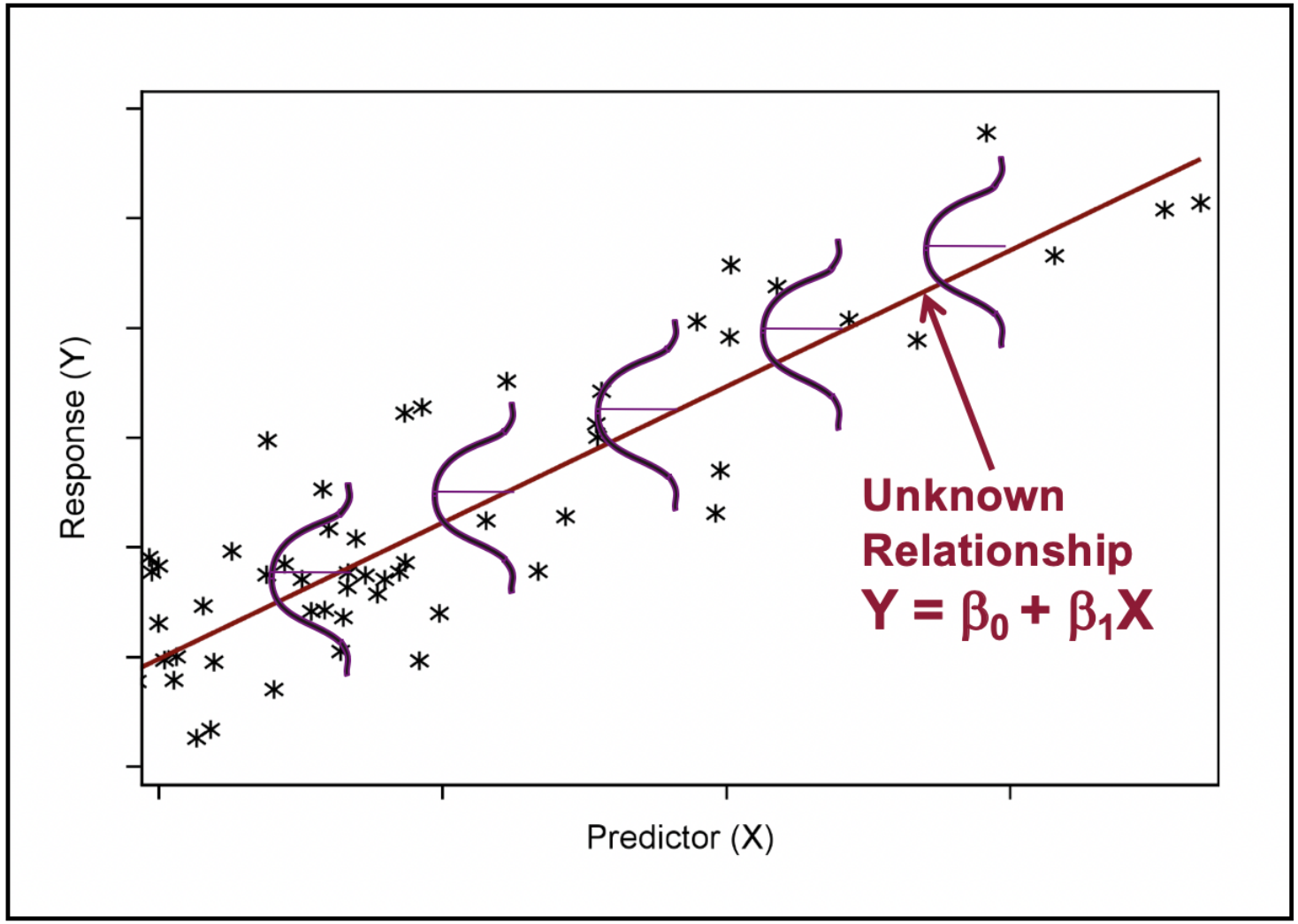
target intercept input

- The least square method aims to minimize the following term

$$\sum_{training\ data} (y_i - \hat{y}_i)^2$$

Linear Regression Assumption

Spending = 500 + 10*Income10K + 2*Age



- Linear relationship between (y, x_i). [Pearson correlation]
- Error is normal distributed. [remove outliers, log-transformation]
- Error has equal variance (homoscedasticity) [remove outliers, log-transformation]
- Errors are independent from each other. [design new data correction]

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

scikit learn Install User Guide API Examples More

Prev Up Next

scikit-learn 0.22.1
Other versions

Please cite us if you use the software.

sklearn.linear_model.LinearRegression
Examples using sklearn.linear_model.LinearRegr

sklearn.linear_model.LinearRegression

class sklearn.linear_model.**LinearRegression**(*fit_intercept=True, normalize=False, copy_X=True, n_jobs=None*) [source]

Ordinary least squares Linear Regression.

LinearRegression fits a linear model with coefficients w = (w1, ..., wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

Parameters:

fit_intercept : *bool, optional, default True*
Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).

normalize : *bool, optional, default False*
This parameter is ignored when `fit_intercept` is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm. If you wish to standardize, please use

se.

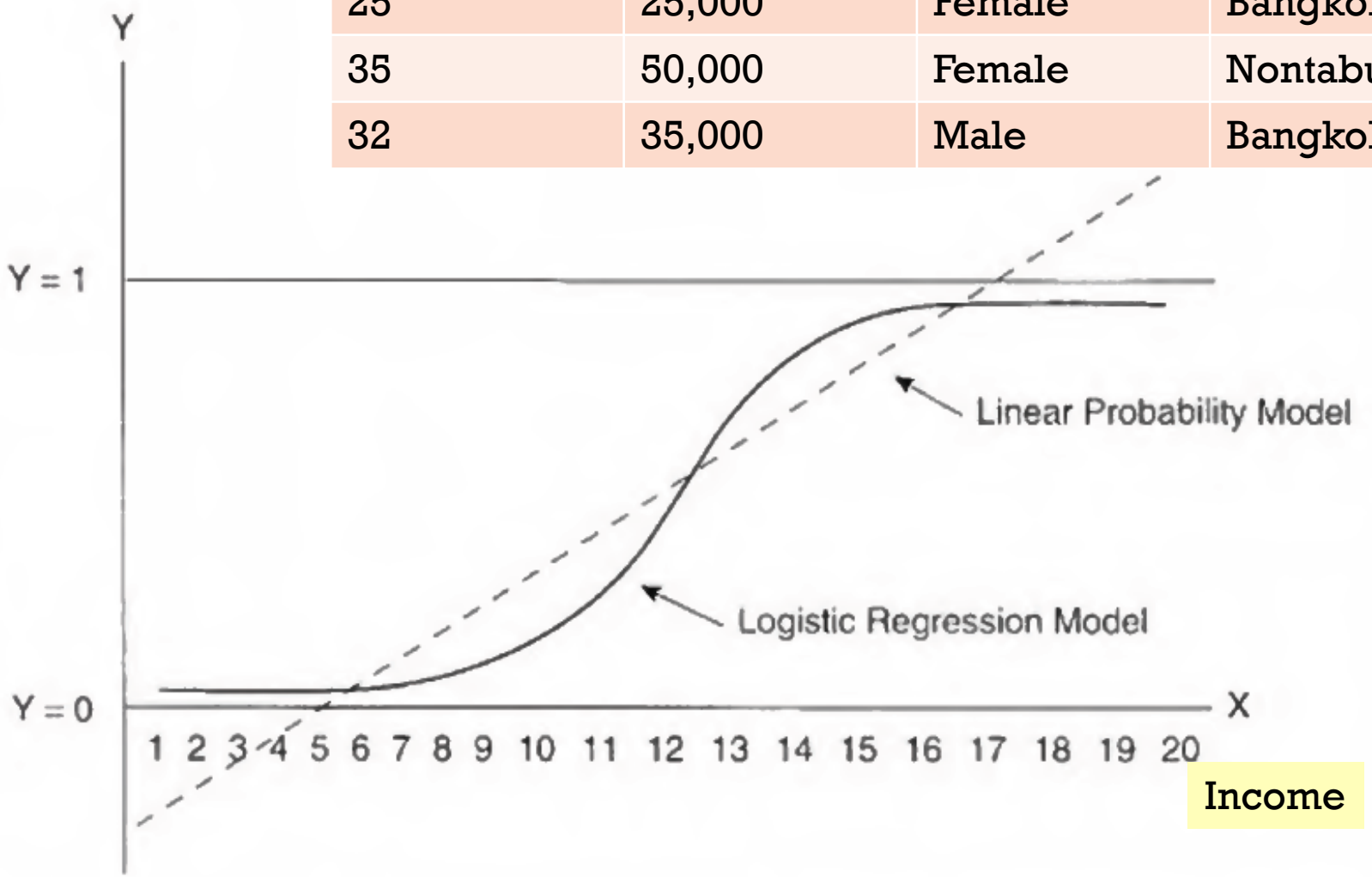
Methods

fit(self, X, y[, sample_weight])	Fit linear model.
get_params(self[, deep])	Get parameters for this estimator.
predict(self, X)	Predict using the linear model.
score(self, X, y[, sample_weight])	Return the coefficient of determination R^2 of the prediction.
set_params(self, **params)	Set the parameters of this estimator.

fficient
ocessors.

Classification Problem

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	Yes (1)
35	50,000	Female	Nontaburi	Yes (1)
32	35,000	Male	Bangkok	No (0)

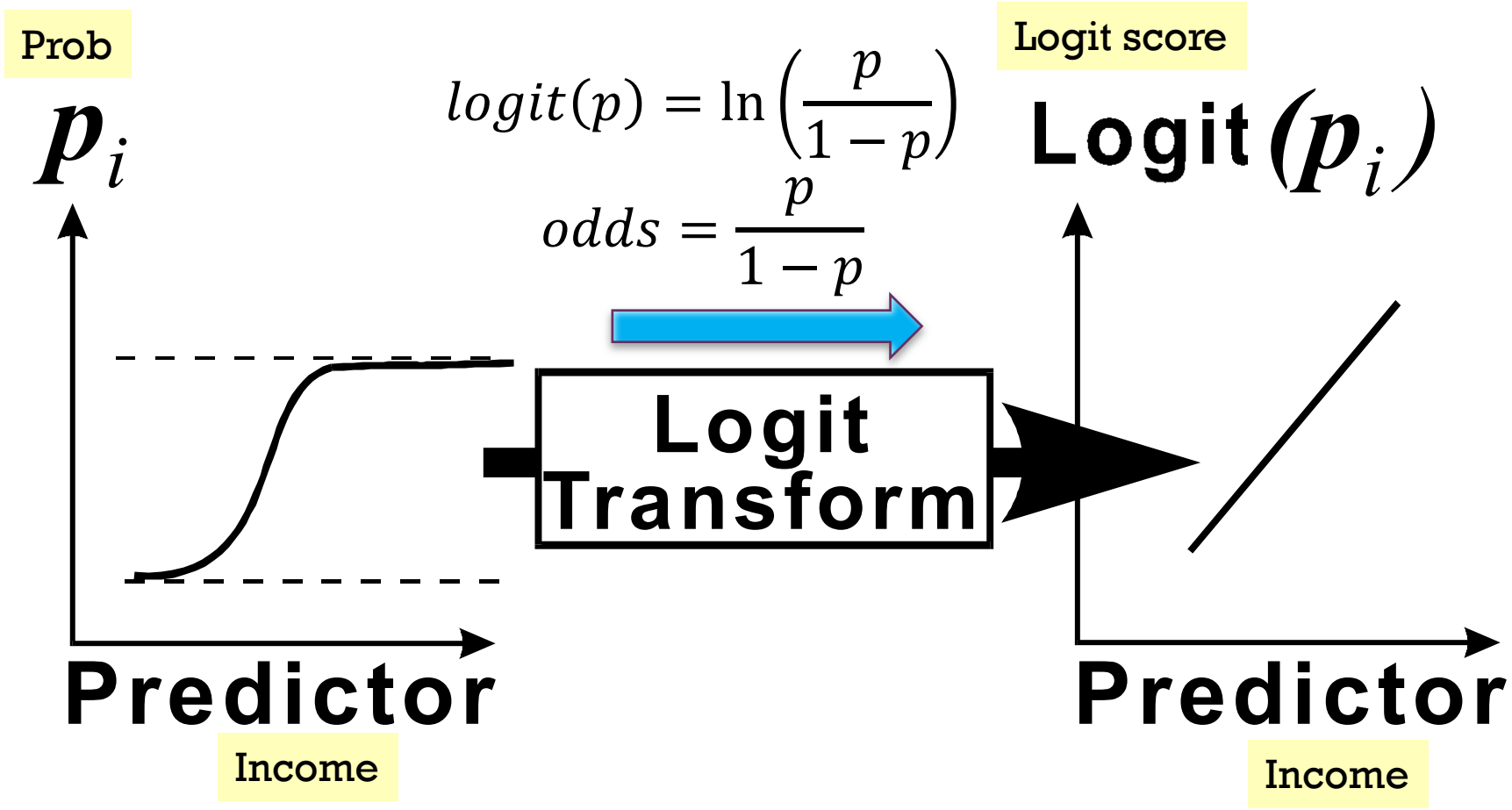


Logistic function
Sigmoid function

Logistic Regression (forward pass)
Logit link function: Non-linear to Linear

$$logit(p) = \ln\left(\frac{p}{1-p}\right)$$

$$odds = \frac{p}{1-p}$$



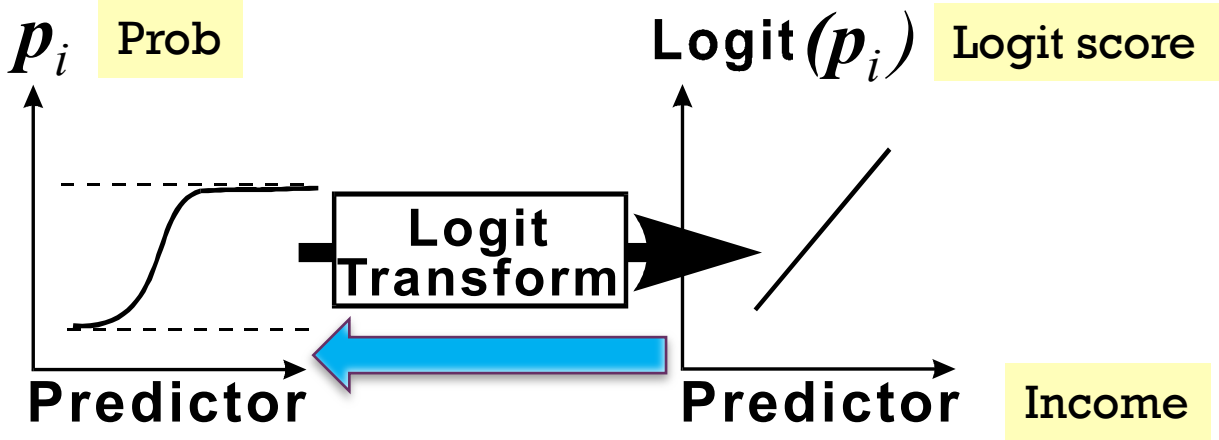
Logit Link Function (backward pass)

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2 = \text{logit}(\hat{p})$$

- Maximum likelihood estimates

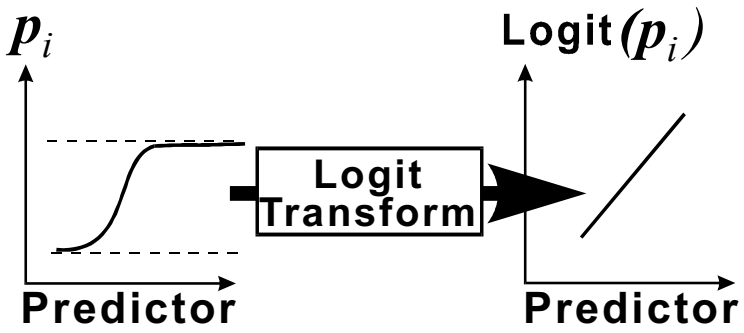
$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

To obtain prediction estimates, the logit equation is solved for $\hat{p}$.



…

Quiz



- What is the logistic regression prediction for the indicated point?

a. 0.243

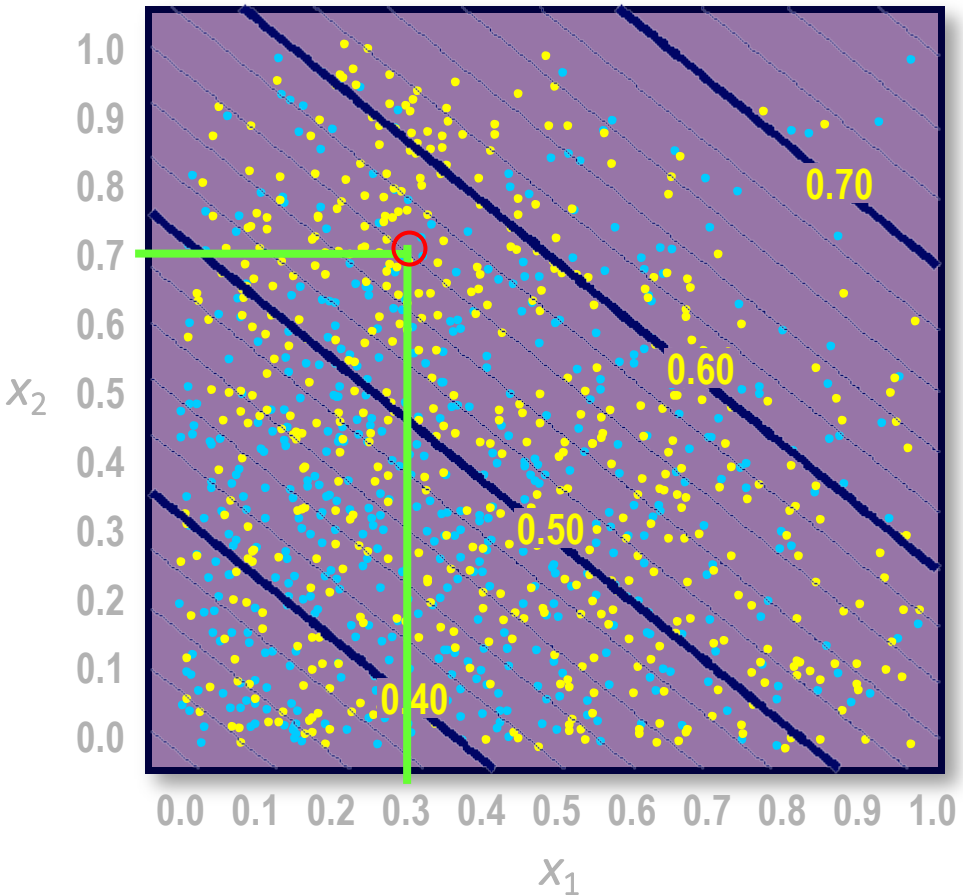
b. 0.56

c. yellow

d. It depends.

$$\text{logit}(\hat{p}) = -0.81 + 0.92\,x_1 + 1.11\,x_2$$

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$



Prev Up Next

scikit-learn 0.22.1
Other versions

Please cite us if you use the software.

sklearn.linear_model.LogisticRegression
Examples using sklearn.linear_model.LogisticRe

sklearn.linear_model.LogisticRegression

class sklearn.linear_model.**LogisticRegression**(*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) ¶ [source]

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using
regularization is applied by default. It can handle both c
64-bit floats for optimal performance; any other input form

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2
solver supports both L1 and L2 regularization, with a dual
supported by the 'saga' solver.

Read more in the User Guide.

Examples

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```

Parameters: **penalty : *{'l1', 'l2', 'elasticnet', 'none'}, default='l2'***

Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties. 'elasticnet' is only supported by the 'saga' solver. If 'none' (not supported by the liblinear solver), no regularization is applied.

New in version 0.19: l1 penalty with SAGA solver (allowing 'multinomial' + L1)

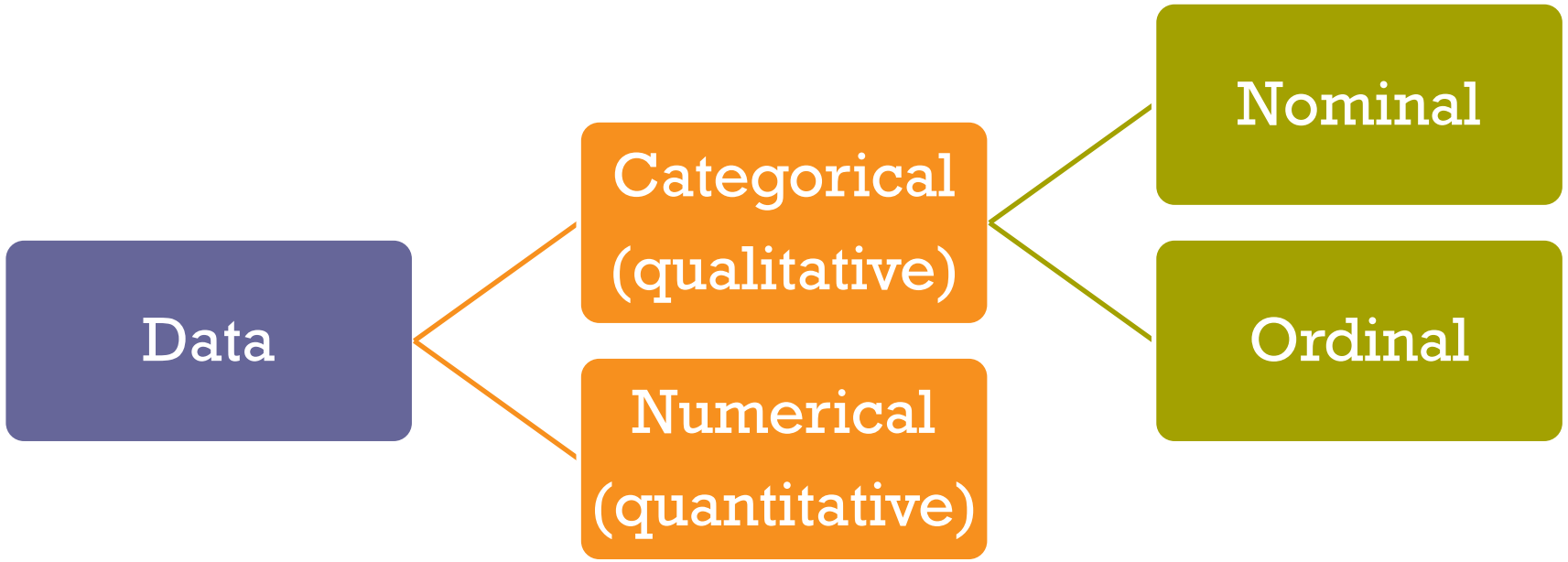
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Linear Regression Limitation

- Manage missing value
- Handle outliers (skewness)
- Handle categorical variables (dummy codes)
- Variable selection:
  - Forward, Backward, Stepwise (not exist in scikit-learn)
- Accounted for nonlinearities

Important Params:
- Require good data preparation
- Variable selection

Terminology: Kinds of data (recap)



Categories cannot be compared

Categories with an implied order

Ordinal: Recode

Grade	GradeNum
A	4.00
B+	3.50
B	3.00
C+	2.50
C	2.00
D+	1.50
D	1.00
F	0.00

Size	SizeNum
XL	5
L	4
M	3
S	2
XS	1

Categorical

- Dummy coding = (n-1) dummy codes

Branch	BranchNum	D_BKK	B_Patum	D_Non	D_BKK	B_Patum	
BKK	1	1	0	0	1	0	
Patumtani	2	0	1	0	0	1	
Nontaburi	3	0	0	1	0	0	reference

pandas.get_dummies

`pandas.get_dummies`(*data, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False, dtype=None*) → 'DataFrame' [source]

Convert categorical variable into dummy/indicator variables.

Parameters: **data** : *array-like, Series, or DataFrame*

Data of which to get dummy indicators.

prefix : *str, list of str, or dict of str, default None*

String to append DataFrame column names. Pass a list with length equal to the number of columns when calling get_dummies on a DataFrame. Alternatively, *prefix* can be a dictionary mapping column names to prefixes.

prefix_sep : *str, default '_'*

If appending prefix, separator/delimiter to use. Or pass a list or dictionary as with *prefix*.

dummy_na : *bool, default False*

Add a column to indicate NaNs, if False NaNs are ignored.

ke, default None

es in the DataFrame to be encoded. If *columns* is None then all the columns with *object* or *category* converted.

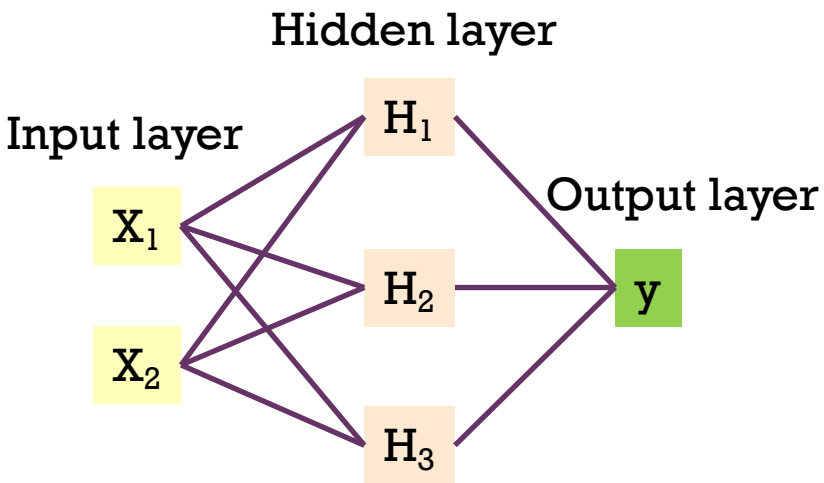
efault False

dummy-encoded columns should be backed by a `SparseArray` (True) or a regular NumPy array (False).

```
>>> pd.get_dummies(pd.Series(list('abcaa')), drop_first=True)
   b  c
0  0  0
1  1  0
2  0  1
3  0  0
4  0  0
```

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html

3) Neural Networks (universal approximator)

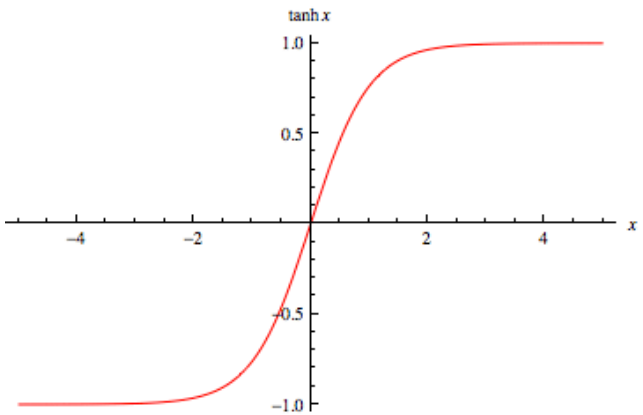


$$log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \widehat{w}_0 + \widehat{w}_1 H_1 + \widehat{w}_2 H_2 + \widehat{w}_3 H_3$$

$$H_1 = tanh(\widehat{w}_{10} + \widehat{w}_{11} x_1 + \widehat{w}_{12} x_2)$$

$$H_2 = tanh(\widehat{w}_{20} + \widehat{w}_{21} x_1 + \widehat{w}_{22} x_2)$$

$$H_3 = tanh(\widehat{w}_{30} + \widehat{w}_{31} x_1 + \widehat{w}_{32} x_2)$$



Stop when?
- Converge (no change in loss)
- Max epochs

Important Params:
- #hidden units, #hidden layers
- Learning rate, Momentum, decay
- Seed number, etc.

Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

sklearn.neural_network.MLPClassifier
Examples using sklearn.neural_network.MLPClass

sklearn.neural_network.MLPClassifier

class sklearn.neural_network.**MLPClassifier**(*hidden_layer_sizes=(100,), activation='relu', solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000*) ¶ [source]

Multi-layer Perceptron classifier.

This model optimizes the log-loss function using LBFGS or stochastic gradient descent.

New in version 0.18.

Parameters:

hidden_layer_sizes : *tuple, length = n_layers - 2, default=(100,)*
The ith element represents the number of neurons in the ith hidden layer.

activation : *{'identity', 'logistic', 'tanh', 'relu'}, default='relu'*
Activation function for the hidden layer.

- 'identity', no-op activation, useful to implement linear bottleneck, returns f(x) = x
- 'logistic', the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).
- 'tanh', the hyperbolic tan function, returns f(x) = tanh(x).
- 'relu', the rectified linear unit function, returns f(x) = max(0, x)

solver : *{'lbfgs', 'sgd', 'adam'}, default='adam'*
The solver for weight optimization.

Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

sklearn.neural_network.MLPRegressor
Examples using sklearn.neural_network.MLPRegre

sklearn.neural_network.MLPRegressor

class sklearn.neural_network.**MLPRegressor**(*hidden_layer_sizes=(100,), activation='relu', solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000*) ¶ [source]

Multi-layer Perceptron regressor.

This model optimizes the squared-loss using LBFGS or stochastic gradient descent.

New in version 0.18.

Parameters:

hidden_layer_sizes : *tuple, length = n_layers - 2, default=(100,)*
The ith element represents the number of neurons in the ith hidden layer.

activation : *{'identity', 'logistic', 'tanh', 'relu'}, default='relu'*
Activation function for the hidden layer.

- 'identity', no-op activation, useful to implement linear bottleneck, returns f(x) = x
- 'logistic', the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).
- 'tanh', the hyperbolic tan function, returns f(x) = tanh(x).
- 'relu', the rectified linear unit function, returns f(x) = max(0, x)

solver : *{'lbfgs', 'sgd', 'adam'}, default='adam'*
The solver for weight optimization.

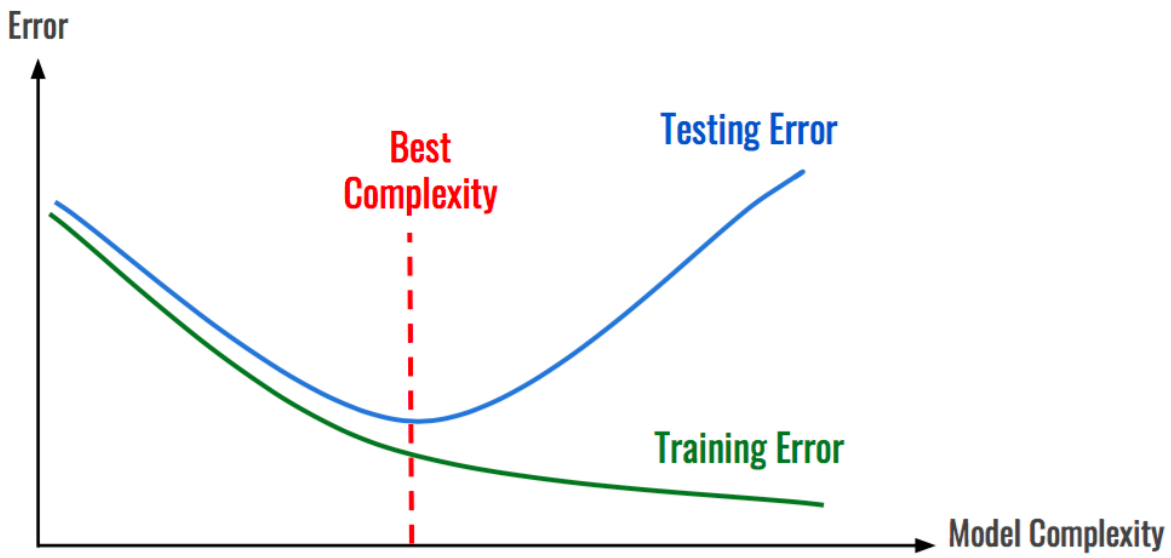
Evaluation (Train/Test Split)

Training Data

Age	Income	Purchase
25	25,000	Yes
35	50,000	Yes
32	35,000	No

Testing Data

Age	Income	Purchase
27	35,000	Yes
23	20,000	No
45	34,000	No



Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

sklearn.model_selection.train_test_split
Examples using sklearn.model_selection.train_t

sklearn.model_selection.train_test_split

`sklearn.model_selection.train_test_split`(**arrays*, ***options*) [source]

Split arrays or matrices into random train and test subsets

Quick utility that wraps input validation and `next(ShuffleSplit().split(X, y))` and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

Read more in the User Guide.

Parameters:

arrays : *sequence of indexables with same length / shape[0]
Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

test_size : *float, int or None, optional (default=None)*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. If `train_size` is also None, it will be set to 0.25.

train_size : *float, int, or None, (default=None)*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

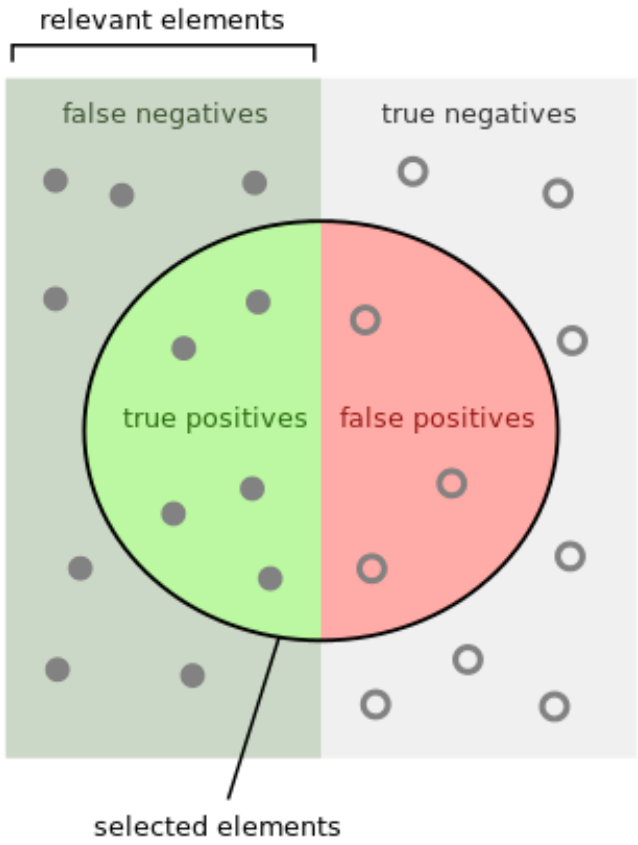
random_state : *int, RandomState instance or None, optional (default=None)*
If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by `np.random`.

Model Evaluation

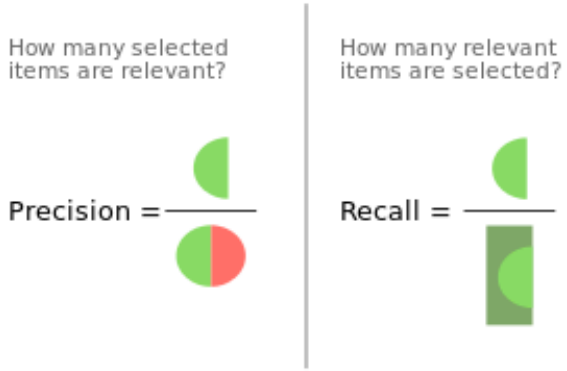
- Regression
  - Sum Squared Error (SSE)
  - Average Squared Error (ASE)
- Classification
  - **Accuracy**
  - Misclassification
  - Precision
  - Recall
  - F1
- Graph
  - ROC Curve
  - Area Under ROC (c-statistic)
  - Lift
  - Gain
  - Response

Precision, Recall, F1

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	



- Precision = correctly predict = TP / (TP + FP)
- Recall = coverage = TP / (TP + FN)
- F1 = (2*pre*rec) / (pre + rec)



Prev Up Next

scikit-learn 0.22.1
Other versions

Please cite us if you use the software.

sklearn.metrics.confusion_matrix
Examples using sklearn.metrics.confusion_matri

sklearn.metrics.confusion_matrix

sklearn.metrics.**confusion_matrix**(*y_true, y_pred, labels=None, sample_weight=None, normalize=None*) [source]

Compute confusion matrix to evaluate the accuracy of a classification.

By definition a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i and predicted to be in group j.

Thus in binary classification, the count of true negatives is $C_{0,0}$, false negatives is $C_{1,0}$, true positives is $C_{1,1}$ and false positives is $C_{0,1}$.

Read more in the User Guide.

Parameters:

y_true : *array-like of shape (n_samples,)*
Ground truth (correct) target values.

f shape (n_samples,)
s returned by a classifier.

shape (n_classes), default=None
x the matrix. This may be used to reorder or select a subset of labels. If `None` is given, those once in `y_true` or `y_pred` are used in sorted order.

y-like of shape (n_samples,), default=None

normalize : *{'true', 'pred', 'all'}, default=None*
Normalizes confusion matrix over the true (rows), predicted (columns) conditions or all the population. If None, confusion matrix will not be normalized.

```
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

Prev Up Next

scikit-learn 0.22.1
Other versions

Please cite us if you use the software.

sklearn.metrics.classification_report
Examples using sklearn.metrics.classification_

sklearn.metrics.classification_report

sklearn.metrics.**classification_report**(*y_true, y_pred, labels=None, target_names=None, sample_weight=None, digits=2, output_dict=False, zero_division='warn'*) [source]

Build a text report showing the main

Read more in the User Guide.

Parameters:

y_true : *1d array*
Ground truth (c

y_pred : *1d array*
Estimated targe

labels : *array, sh*
Optional list of

target_names : *l*
Optional displa

sample_weight :
Sample weight

digits : *int*
Number of digi
the returned va

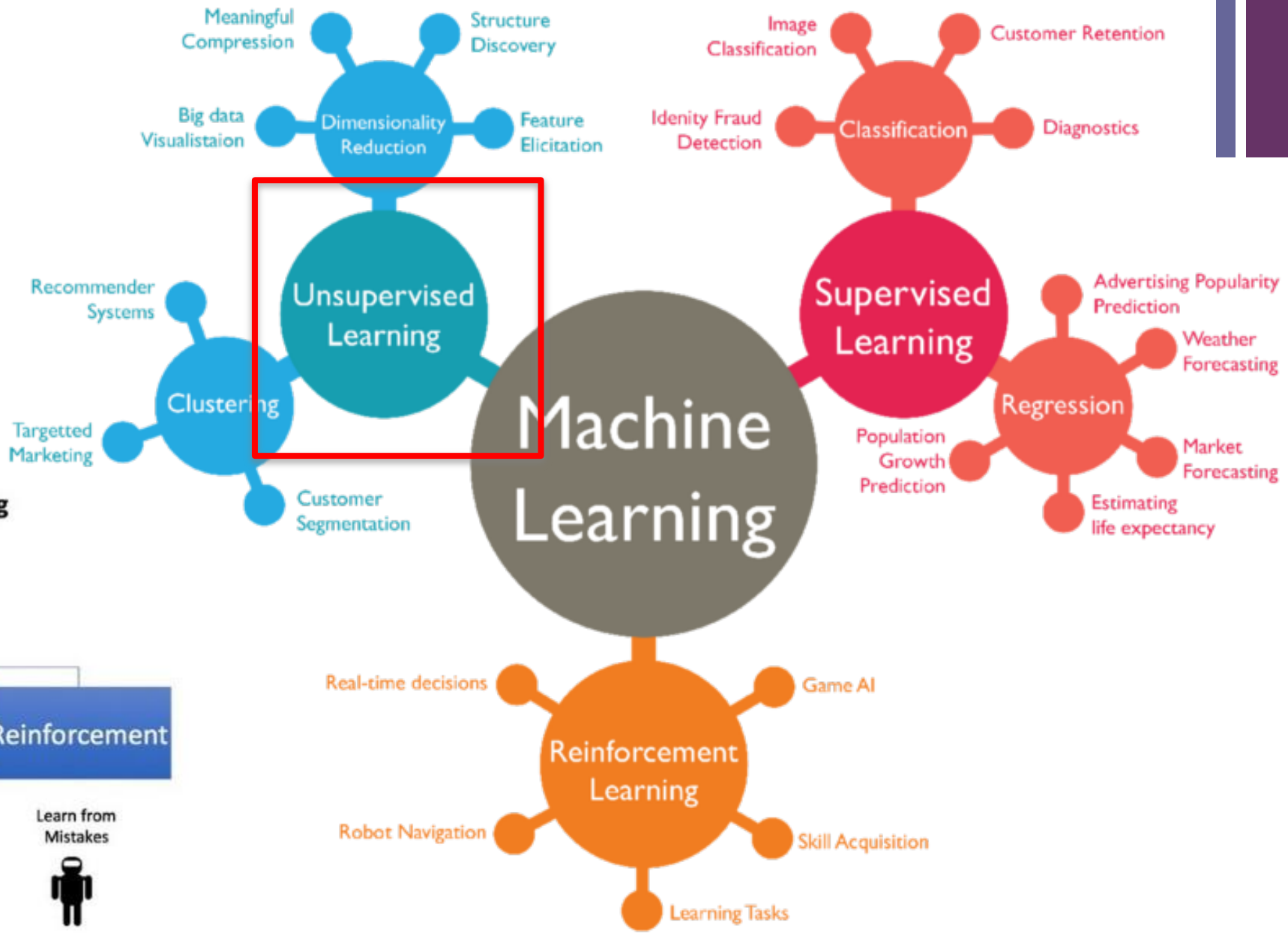
Examples

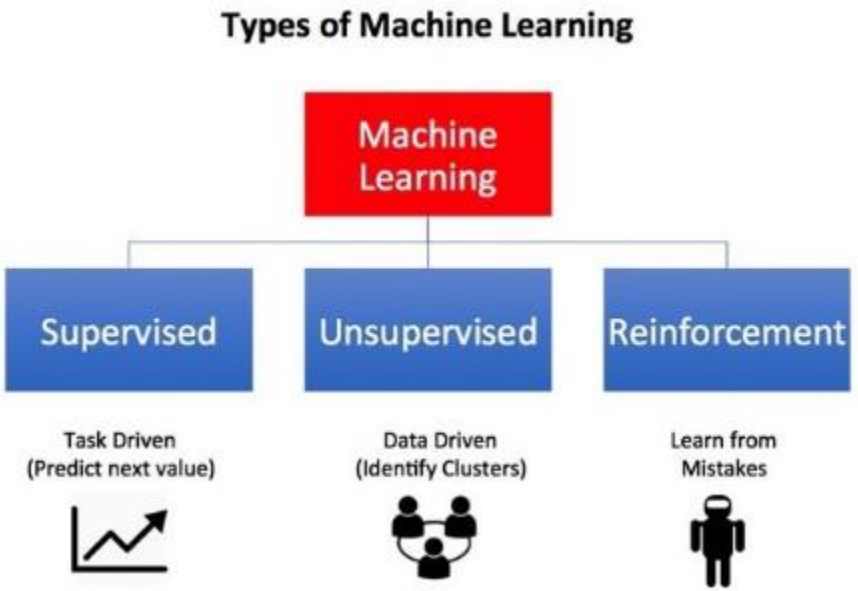
```
>>> from sklearn.metrics import classification_report
>>> y_true = [0, 1, 2, 2, 2]
>>> y_pred = [0, 0, 2, 2, 1]
>>> target_names = ['class 0', 'class 1', 'class 2']
>>> print(classification_report(y_true, y_pred, target_names=target_names))
              precision    recall  f1-score   support
<BLANKLINE>
     class 0       0.50      1.00      0.67         1
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.67      0.80         3
<BLANKLINE>
    accuracy                           0.60         5
   macro avg       0.50      0.56      0.49         5
weighted avg       0.70      0.60      0.61         5
<BLANKLINE>
>>> y_pred = [1, 1, 0]
>>> y_true = [1, 1, 1]
>>> print(classification_report(y_true, y_pred, labels=[1, 2, 3]))
              precision    recall  f1-score   support
<BLANKLINE>
           1       1.00      0.67      0.80         3
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
<BLANKLINE>
   micro avg       1.00      0.67      0.80         3
   macro avg       0.33      0.22      0.27         3
weighted avg       1.00      0.67      0.80         3
<BLANKLINE>
```

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report

Unsupervised Learning (Descriptive Task)

Machine Learning (cont.)





Task2: Unsupervised learning (descriptive task)

Training Data

inputs

target

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	[illegible]

Testing Data

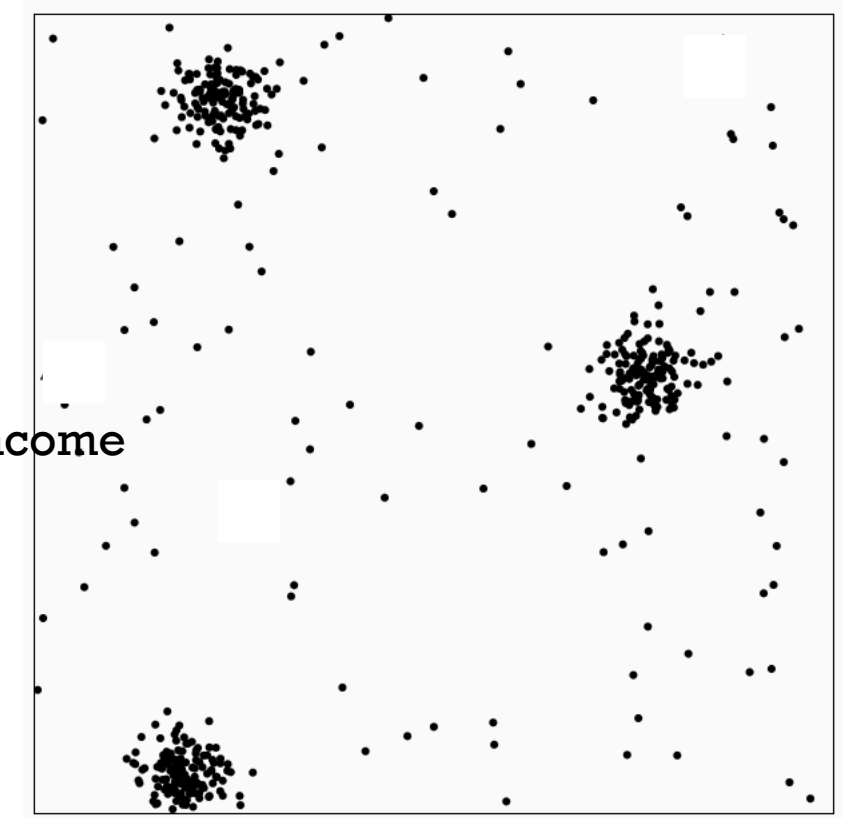
Age	Income	[illegible]	Province	Purchase
25	25,000	[illegible]	Bangkok	?

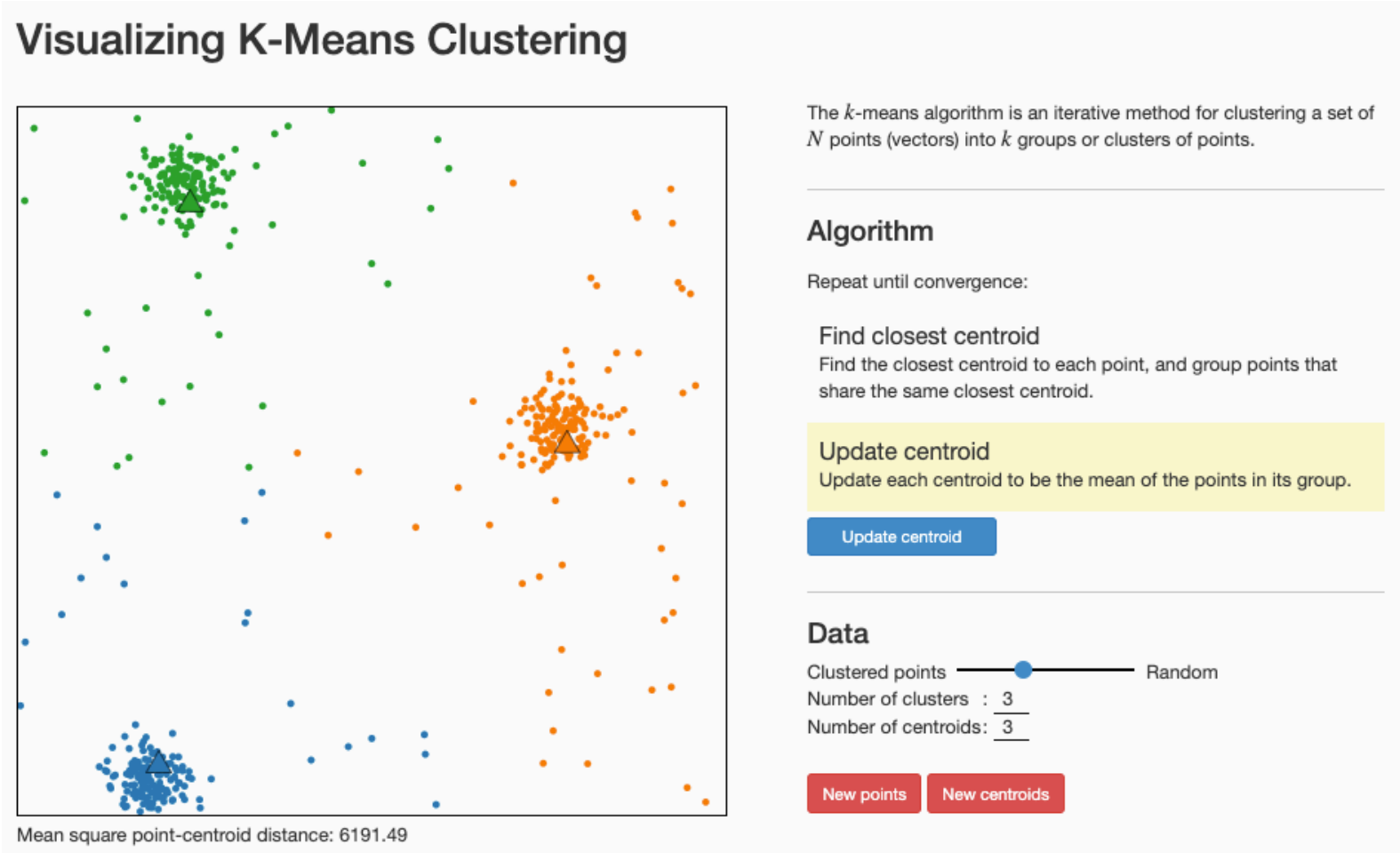
Clustering

- In our class, there are many participants. Should we teach them using the same method?
- May be not! Since they may have different learning behaviors and backgrounds.
- Inputs
  - Education field
  - Level of English communication
  - Level of computer skills
  - Age range
  - Gender

K-means Clustering

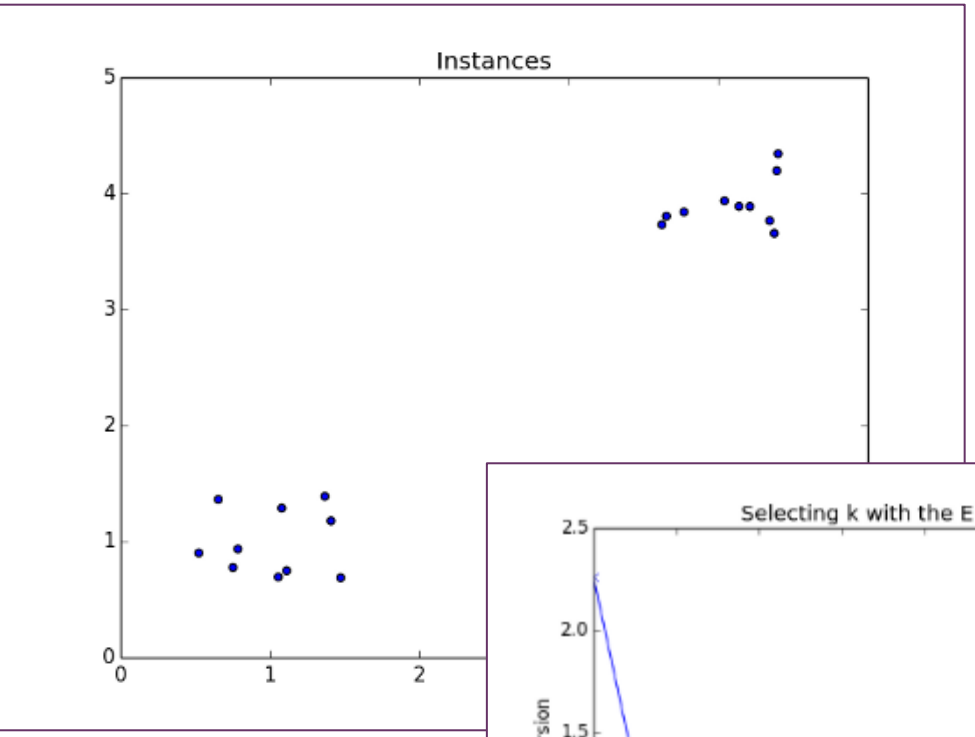
- http://web.stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

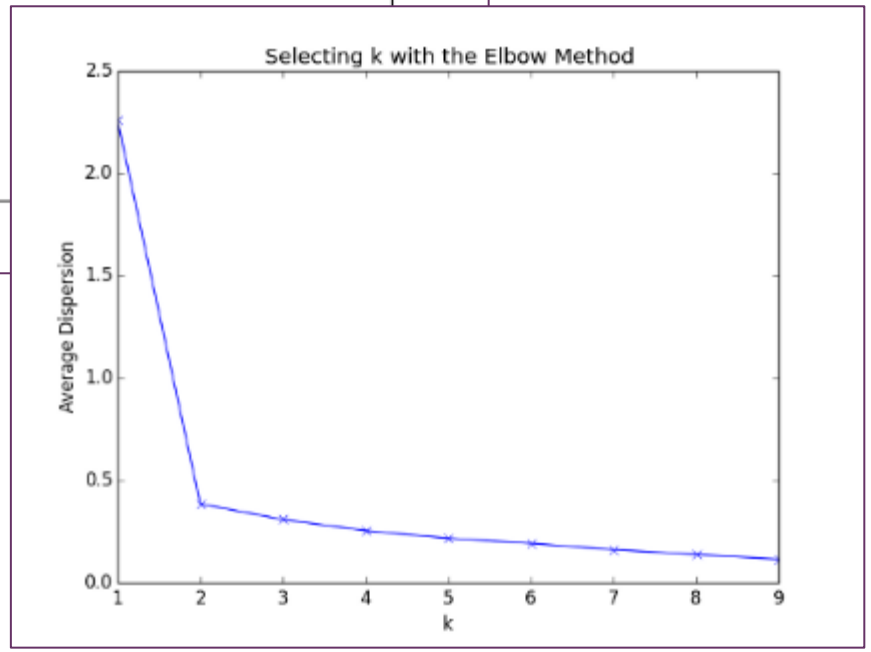


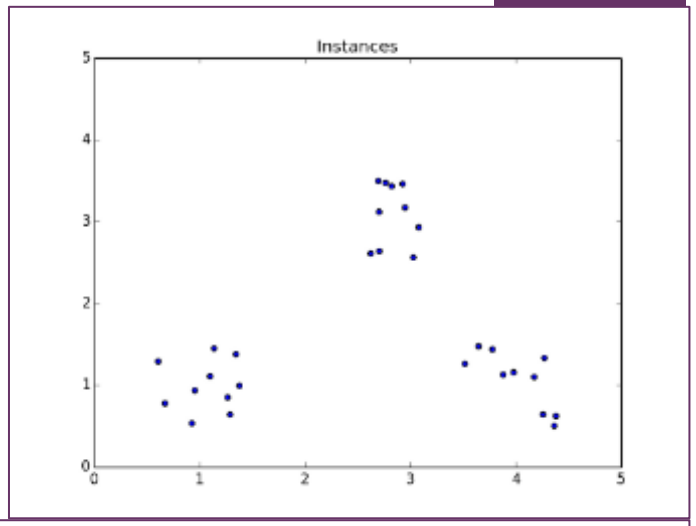


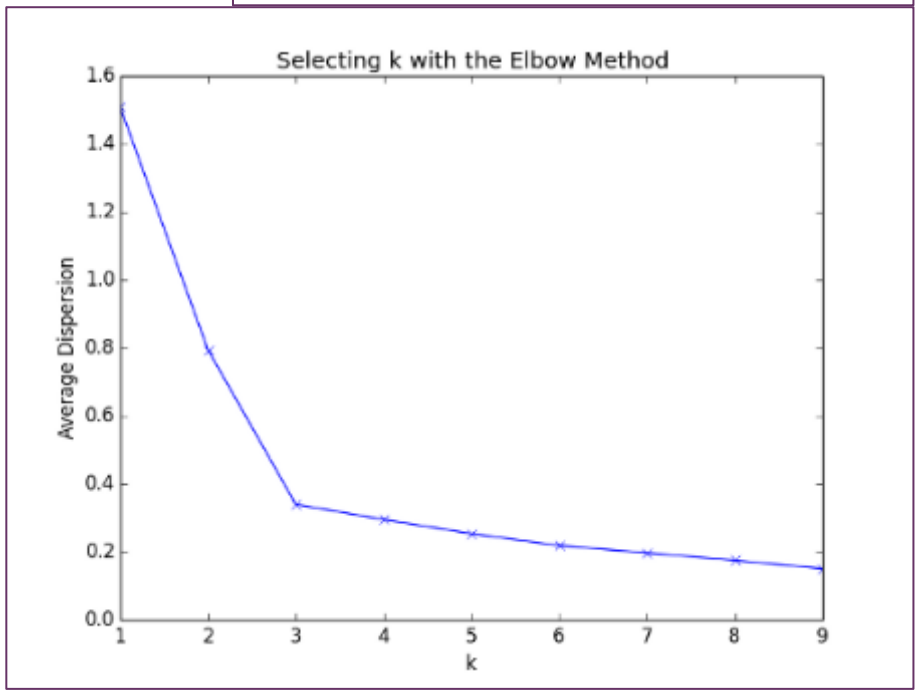
Determine the number of *k*: Elbow Method

https://www.packtpub.com/books/content/clustering-k-means









Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

sklearn.preprocessing.StandardScaler
Examples using sklearn.preprocessing.StandardS

sklearn.preprocessing.StandardScaler

class `sklearn.preprocessing.`**`StandardScaler`**(*copy=True, with_mean=True, with_std=True*) [source]

Standardize features by removing the mean and scaling to unit variance

The standard score of a sample `x` is calculated as:

z = (x - u) / s

where `u` is the mean of the training samples or zero if `with_mean=False`, and `s` is the standard deviation of the training samples or one if `with_std=False`.

Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using `transform`.

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).

For instance many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the L1 and L2 regularizers of linear models) assume that all features are centered around 0 and have variance in the same order. If a feature has a variance that is orders of magnitude larger that others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

This scaler can also be applied to sparse CSR or CSC matrices by passing `with_mean=False` to avoid breaking the sparsity structure of the data.

Toggle Menu

Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

sklearn.cluster.KMeans
Examples using sklearn.cluster.KMeans

sklearn.cluster.KMeans

class `sklearn.cluster.`**`KMeans`**(*n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=None, algorithm='auto'*) [source]

K-Means clustering.

Read more in the User Guide.

Parameters:

n_clusters : *int, default=8*
The number of clusters to form as well as the number of centroids to generate.

init : *{'k-means++', 'random'} or ndarray of shape (n_clusters, n_features), default='k-means++'*
Method for initialization, defaults to 'k-means++':

'k-means++' : selects initial cluster centers for k-mean clustering in a smart way to speed up convergence. See section Notes in k_init for more details.

'random': choose k observations (rows) at random from data for the initial centroids.

If an ndarray is passed, it should be of shape (n_clusters, n_features) and gives the initial centers.

n_init : *int, default=10*
Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.

max_iter : *int, default=300*

Toggle Menu

Extra Topics (Not Cover)

Market Basket Analysis (Association Rule Mining)

Rule	Support	Confidence
Apple => Donut	2/5	2/3
Coconut > Apple	2/5	2/4
Apple => Coconut	2/5	2/3
Banana & Coconut => Donut	1/5	1/3

Association Rule Mining (cont.)

- Wal-Mart customers who purchase Barbie dolls have a 60% likelihood of also purchasing one of three types of candy bars [Forbes, Sept 8, 1997]

- Strategies?
  1. Put them closer together in the store.
  2. Put them far apart in the store.
  3. Package candy bars with the dolls.
  4. Package Barbie + candy + poorly selling item.
  5. Raise the price on one, and lower it on the other.
  6. Offer Barbie accessories for proofs of purchase.
  7. Do not advertise candy and Barbie together.
  8. Offer candies in the shape of a Barbie doll.

Caution in Association Rule Mining

1. Basket size: per bill, customer, day
2. Item level: SKU, product category

apriori
association_rules
fpgrowth
fpmax

mlxtend version: 0.17.2

apriori

apriori(df, min_support=0.5, use_colnames=False, max_len=None, verbose=0, low_memory=False)

Get frequent itemsets from a one-hot DataFrame

Parameters

- `df` : pandas DataFrame

  pandas DataFrame the encoded format. Also supports DataFrames with sparse data; for more info, please see (https://pandas.pydata.org/pandas-docs/stable/ user_guide/sparse.html#sparse-data-structures)

  Please note that the old pandas SparseDataFrame format is no longer supported in mlxtend >= 0.17.2.

  The allowed values are either 0/1 or True/False. For example,

```
 Apple  Bananas   Beer  Chicken   Milk   Rice
0     True    False   True     True  False   True
1     True    False   True    False  False   True
2     True    False   True    False  False  False
```

apriori
association_rules
fpgrowth
fpmax

association_rules

association_rules(df, metric='confidence', min_threshold=0.8, support_only=False)

Generates a DataFrame of association rules including the metrics 'score', 'confidence', and 'lift'

Parameters

- `df` : pandas DataFrame

  pandas DataFrame of frequent itemsets with columns ['support', 'itemsets']

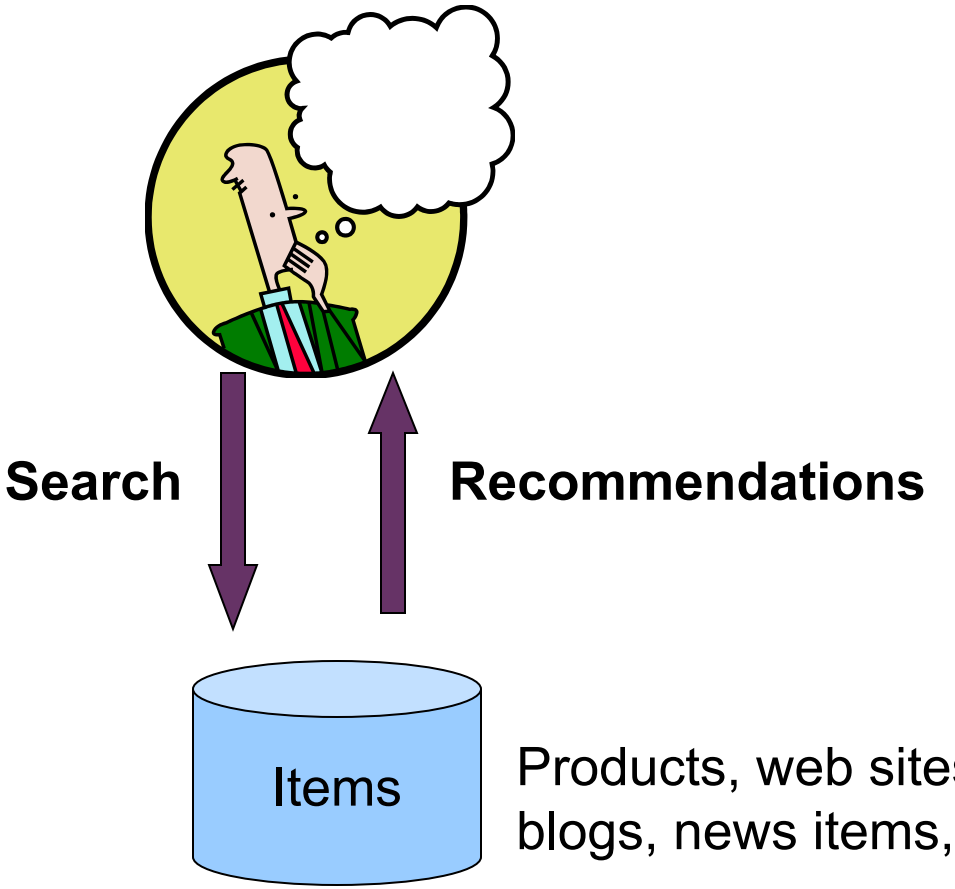
- `metric` : string (default: 'confidence')

  Metric to evaluate if a rule is of interest. **Automatically set to 'support' if `support_only=True`.** Otherwise, supported metrics are 'support', 'confidence', 'lift',

'leverage', and 'conviction' These metrics are computed as follows:

```
- support(A->C) = support(A+C) [aka 'support'], range: [0, 1]

- confidence(A->C) = support(A+C) / support(A), range: [0, 1]

- lift(A->C) = confidence(A->C) / support(C), range: [0, inf]

- leverage(A->C) = support(A->C) - support(A)*support(C),
range: [-1, 1]

- conviction = [1 - support(C)] / [1 - confidence(A->C)],
```

Recommendations



Products, web sites, blogs, news items, …

Examples:

amazon.com
PANDORA
StumbleUpon
del.icio.us
NETFLIX
movielens
helping you find the *right* movies
last·fm
the social music revolution
Google News
YouTube
XBOX LIVE

How to recommend to each user?

- 1) Content-based (k-NN)
- 2) Collaborative Filtering (CF)
- 3) Latent factor based (SVD, NN)

1	3	4			
	3	5			5
		4	5		5
		3			
1					

scipy.sparse.linalg.svds

scipy.sparse.linalg.**svds**(*A, k=6, ncv=None, tol=0, which='LM', v0=None, maxiter=None, return_singular_vectors=True, solver='arpack'*) [source]

Compute the largest or smallest k singular values/vectors for a sparse matrix. The order of the singular values is not guaranteed.

Parameters:

A : *{sparse matrix, LinearOperator}*

Array to compute the SVD on, of shape (M, N)

k : *int, optional*

Number of singular values and vectors to compute. Must be 1 <= k < min(A.shape).

ncv : *int, optional*

The number of Lanczos vectors generated ncv must be greater than k+1 and smaller than n; it is recommended that ncv > 2*k Default: `min(n, max(2*k + 1, 20))`

tol : *float, optional*

Tolerance for singular values. Zero (default) means machine precision.

which : *str, ['LM' | 'SM'], optional*

Which *k* singular values to find:

- 'LM' : largest singular values
- 'SM' : smallest singular values

New in version 0.12.0.

Previous topic

scipy.sparse.linalg.lobpcg

Next topic

scipy.sparse.linalg.splu

Quick search

search

Demo

Demo

- 1_Decision-Trees_Random-Forests.ipynb
- 2_Linear-Regression.ipynb
- 3_Logistic-Regression.ipynb
- 4_Neural-Network.ipynb
- 5_K-Means-Clustering.ipynb

- 6_Market_Basket_Intro.ipynb
- 7_Recommender-Systems.ipynb

Exercise

Target=y - has the client subscribed a term deposit? (binary: 'yes', 'no')

Bank Direct Target Marketing

M1: Simple Logistic Regression

1) data prep

2) Model

- Logistic Regression
- print model (coef, intercept)

3) Evaluation

- confusion matrix
- classification report

M2: Categorical

- dummy code
- recode - ordinal

M3: Model Selection

- backward selection

The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).

Data Set Information: The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

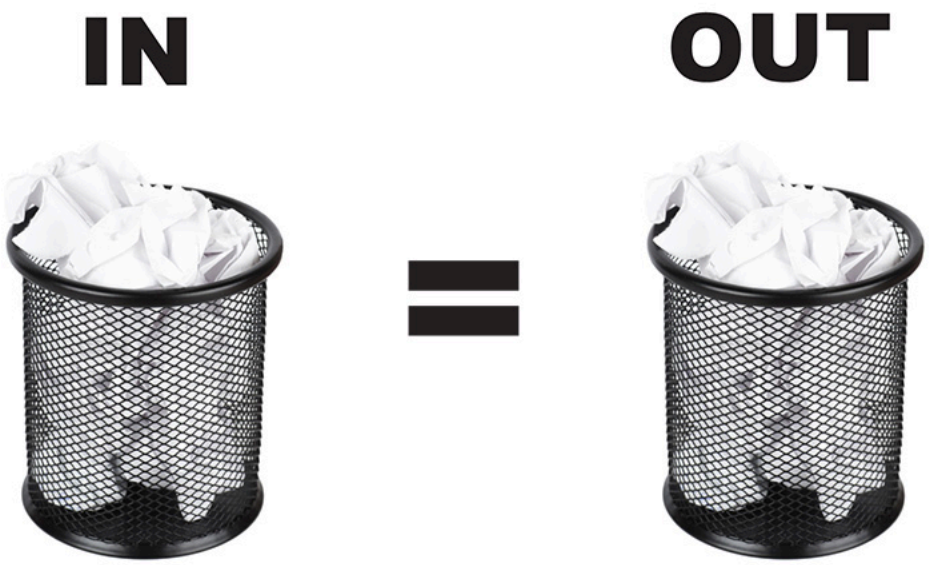
Attribute Information:

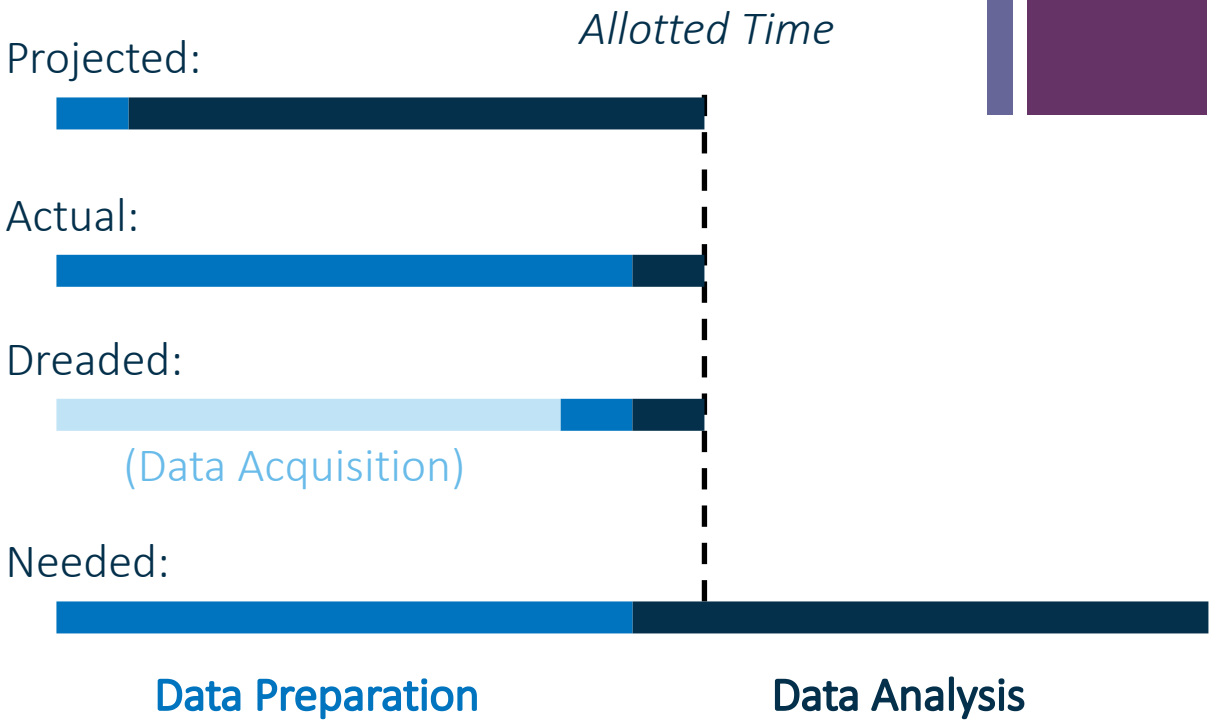
Bank client data:

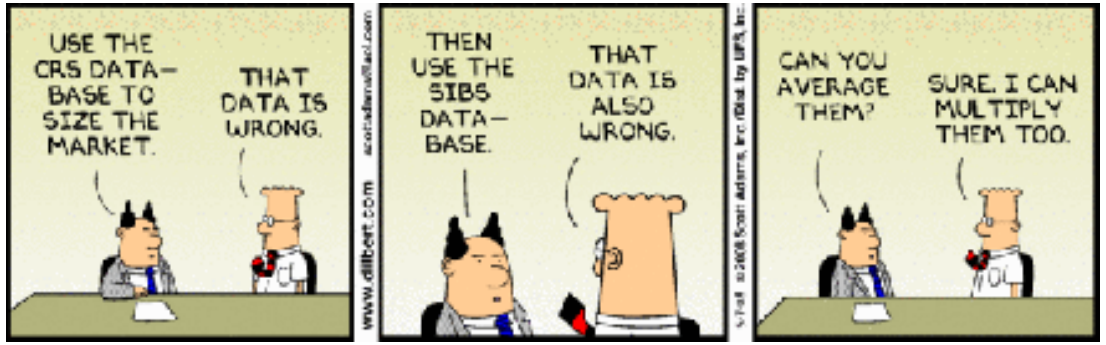
- Age (numeric)
- Job : type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
- Marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown' ; note: 'divorced' means divorced or widowed)

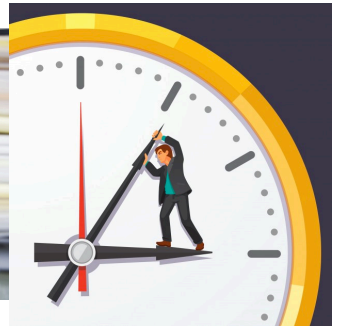
Become Data Scientist and Build Your Team

Data preparation is very important!









Practice is
everything.
Periander
quotefancy

UCI Machine Learning Repository

Center for Machine Learning and Intelligent Systems

View ALL Data Sets

Browse Through: **440** Data Sets

Table View List View

Default Task
- Classification (325)
- Regression (87)
- Clustering (77)
- Other (54)

Attribute Type
- Categorical (37)
- Numerical (284)
- Mixed (55)

Data Type
- Multivariate (335)
- Univariate (20)
- Sequential (45)
- Time-Series (84)
- Text (47)
- Domain-Theory (23)
- Other (21)

Area
- Life Sciences (101)
- Physical Sciences (47)
- CS / Engineering (156)
- Social Sciences (26)
- Business (27)
- Game (10)
- Other (69)

 # Attributes
- Less than 10 (101)
- 10 to 100 (200)
- Greater than 100 (82)

Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year
Abalone	Multivariate	Classification	Categorical, Integer, Real	4177	8	1995
Adult	Multivariate	Classification	Categorical, Integer	48842	14	1996
Annealing	Multivariate	Classification	Categorical, Integer, Real	798	38	
Anonymous Microsoft Web Data		Recommender-Systems	Categorical	37711	294	1998
Arrhythmia	Multivariate	Classification	Categorical, Integer, Real	452	279	1998
Artificial Characters	Multivariate	Classification	Categorical, Integer, Real	6000	7	1992
Audiology (Original)	Multivariate	Classification	Categorical	226		1987

kaggle | Search kaggle | Competitions | Datasets | Kernels | Discussion | Learn | ••• | Sign In

House Prices: Advanced Regression Techniques

Predict sales prices and practice feature engineering, RFs, and gradient boosting

4,527 teams · Ongoing

Overview | Data | Kernels | Discussion | Leaderboard | Rules

Overview

- Description
- Evaluation
- Frequently Asked Questions
- Tutorials

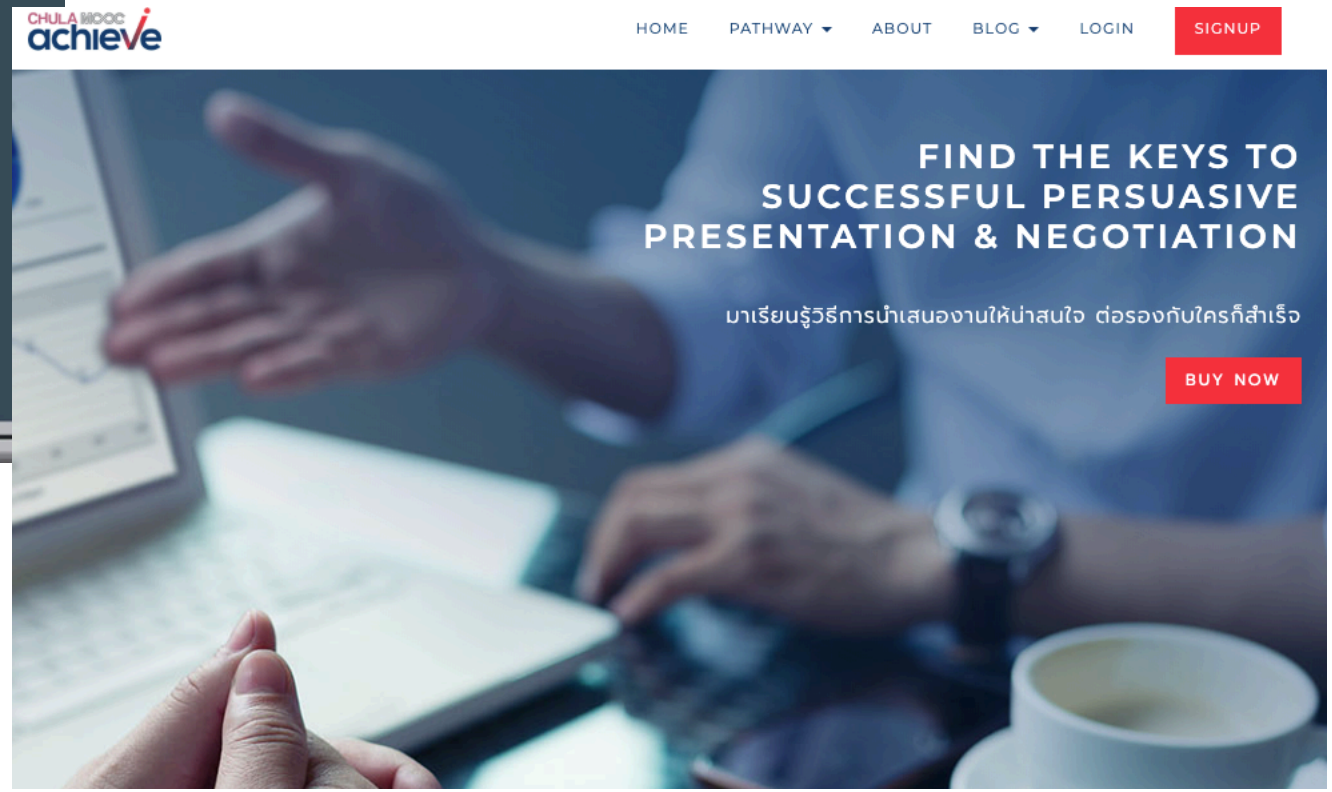
Start here if...

You have some experience with R or Python and machine learning basics. This is a perfect competition for data science students who have completed an online course in machine learning and are looking to expand their skill set before trying a featured competition.

Competition Description

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the

CHULA MOOC
การเรียนรู้แบบไร้ขีดจำกัด
มาเรียนรู้แบบไร้ขีดจำกัดกัน
Anyone can learn !
CHULA MOOC


CHULA MOOC achieve
HOME
PATHWAY
ABOUT
BLOG
LOGIN
SIGNUP
FIND THE KEYS TO SUCCESSFUL PERSUASIVE PRESENTATION & NEGOTIATION
มาเรียนรู้วิธีการนำเสนองานให้น่าสนใจ ต่อรองกับใครก็สำเร็จ
BUY NOW

K KEEP
E EDUCATING
Y YOURSELF

หลักสูตร 2 ปี ภาคในเวลา และนอกเวลาราชการ (เสาร์-อาทิตย์)

2110697
Big Data Tools

2110746
Big Data Analytics

M.Sc.
Big Data

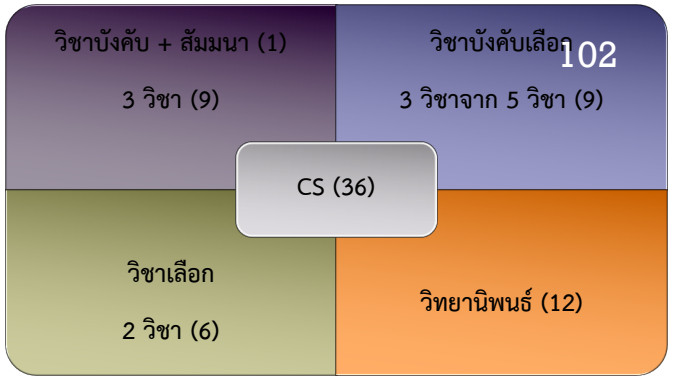
2110747
Social Network Analysis

2110636
Performance Analysis

2110637
Large Scale

Data Science Architecture
Machine Learning
Data Mining
Security
Cloud Technology

Research Lab



- AI/machine learning/data mining
- Deep Learning
- Big data analytics
- Text mining
- Bio informatics
- Smart cities
- Internet of things
- Robotics
- Computer graphics
- 3D printing
- Software engineering

Thank you
& any questions