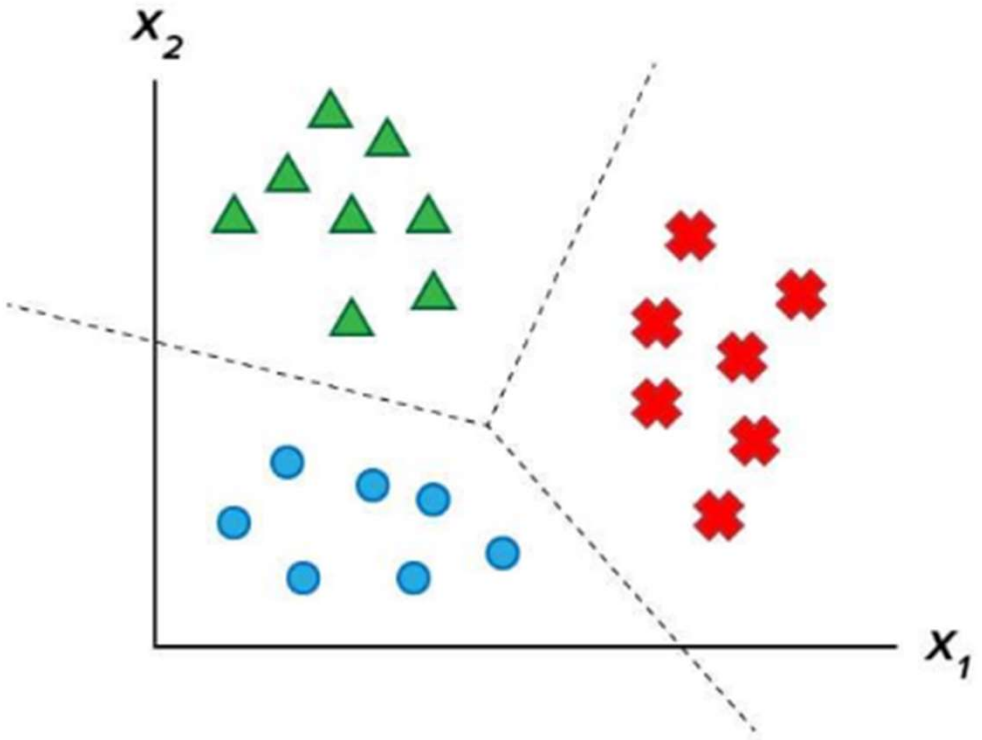


Chapter 9 Classification

CSS 341 Introduction to Data Science

Chukiat Worasucheep

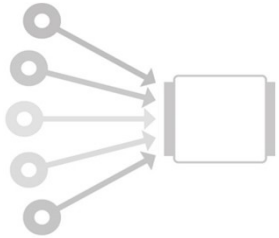
Important Notice

การเรียนการสอนหัวข้อนี้ผ่านทางสื่อออนไลน์ (VDO conference) และอาจมีการบันทึกภาพและเสียง ซึ่งมีไว้ใช้เพื่อประโยชน์ทางการศึกษาต่อไปในอนาคต.
หากท่านไม่ยินยอม ขอให้แจ้งให้ผู้สอนทราบภายในไม่เกินวันรุ่งขึ้น.

Learning objectives

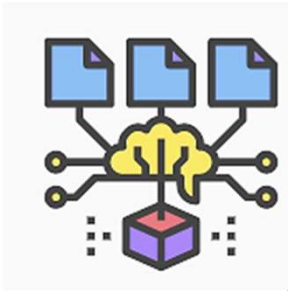
1. เข้าใจแนวคิดของปัญหาการจำแนกประเภท (Classification problem)
2. เข้าใจหลักการเบื้องต้นของขั้นตอนวิธี Logistics regression, Decision tree, Random Forest, และ Artificial neural networks สำหรับปัญหาการจำแนกประเภท
3. เข้าใจแนวทางและหัดใช้ Python ในการแก้ปัญหาการจำแนกประเภทด้วยขั้นตอนวิธีต่างๆ ดังกล่าวข้างต้น
4. เข้าใจแนวทางในการประเมินเทียบประสิทธิภาพของขั้นตอนวิธีจำแนกประเภท
5. วิเคราะห์ผลลัพธ์ที่ได้จากการแก้ปัญหาการจำแนกประเภทด้วย Python
6. เข้าใจปัญหาและสามารถจัดการข้อมูลไม่สมดุล (Imbalanced data)

Data science process



Data acquisition 02

Data Preparation 04



06 Data modeling

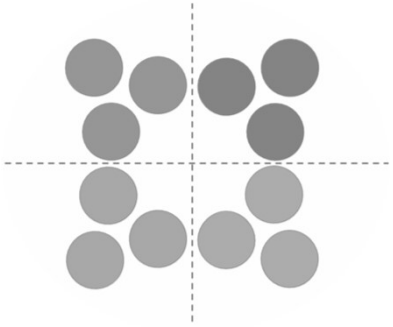
07 Deployment and maintenance

05 Data visualization

03 Data Exploration

01 Understand business and problems

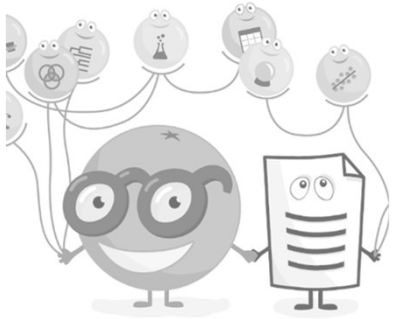
Major modeling techniques for data science



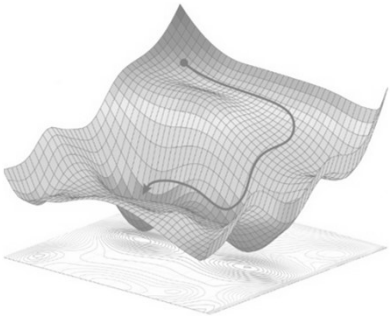
Clustering
การจัดกลุ่ม

Classification
การจำแนกประเภท

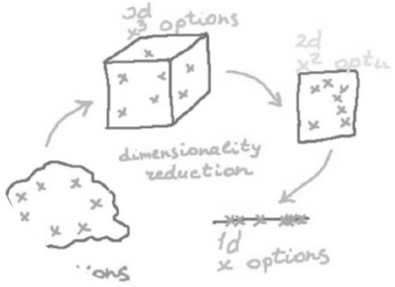
Regression
การวิเคราะห์ถดถอย



Association rule mining
การค้นหากฎความสัมพันธ์



Optimization
การหาค่าเหมาะที่สุด



Dimensionality reduction
การลดมิติข้อมูล

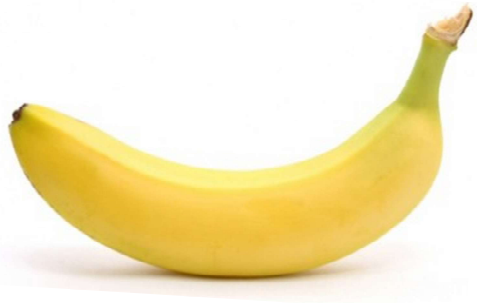
Outline

- Foundation of classification
- Logistics regression
- Decision Tree, Random Forest
- Artificial Neural Networks
- Generic classification modeling process
- Model validation methods
- มาตรวัดประสิทธิภาพของ classification algorithms
- Imbalanced data
- Feature selection
- Hyperparameter tuning

Classification problems ปัญหาการจำแนกประเภท

Classification
การจำแนกประเภท

- ในทาง machine learning, ปัญหาการจำแนกประเภท (Classification) หมายถึง ปัญหาการจำแนกข้อมูลต่าง ๆ ออกเป็นประเภท (class) ต่าง ๆ ตามคอลัมน์เป้าหมาย (label) ที่กำหนดไว้
- โดยใช้ คุณลักษณะจำเพาะเพื่อจำแนก (features) บางอย่างเป็นเกณฑ์ในการจำแนก



Features (or attributes)

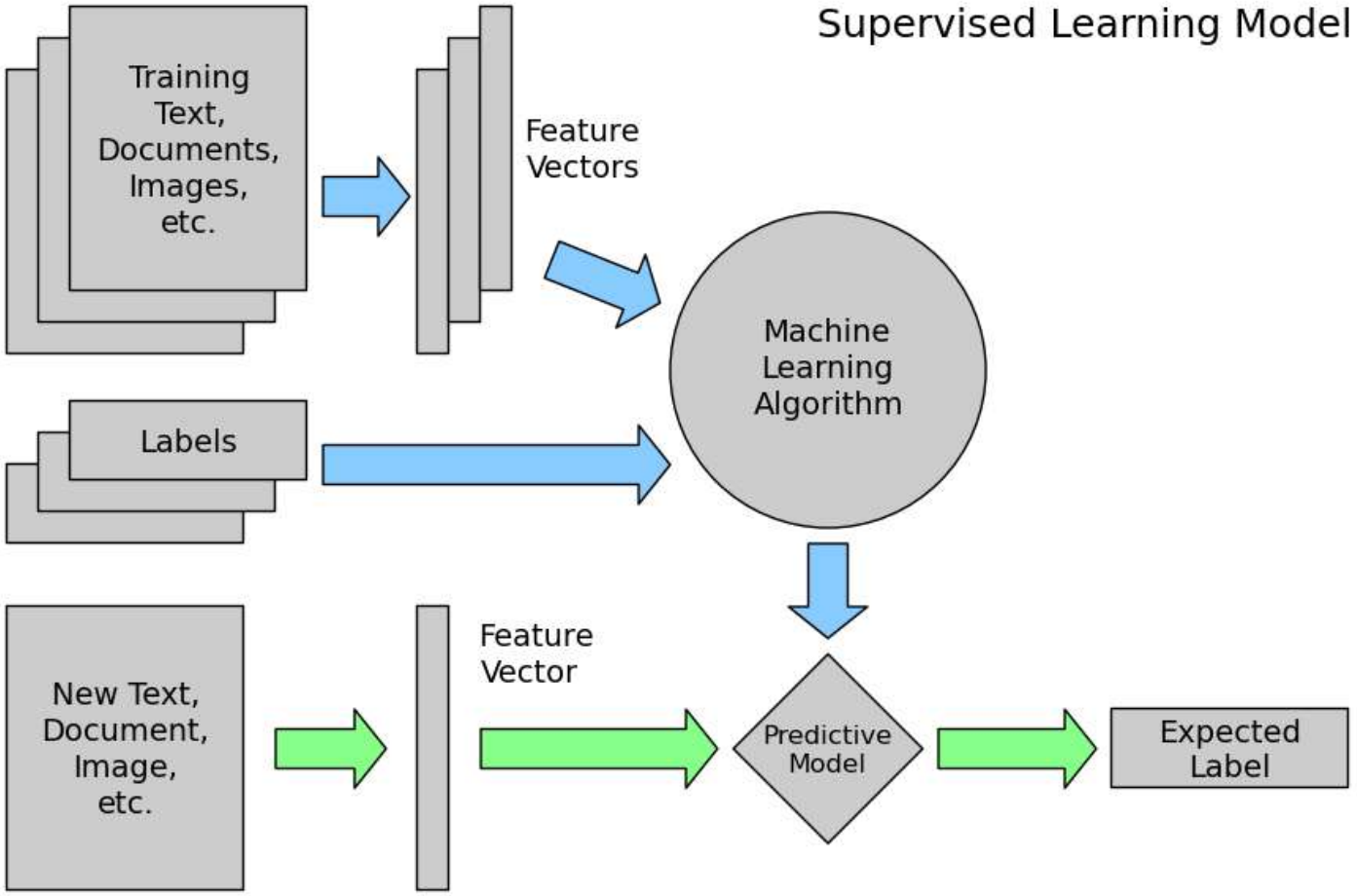
ขนาด	สี	รูปร่าง	ชนิด
3-4"	แดง	กลม	แอปเปิล
3-4"	ส้ม	กลม	ส้ม
5-7"	เหลือง	ยาว	กล้วย
3-4"	เขียว	กลม	แอปเปิล
...	...	...	...

Label

classes

Classification algorithms are supervised learning

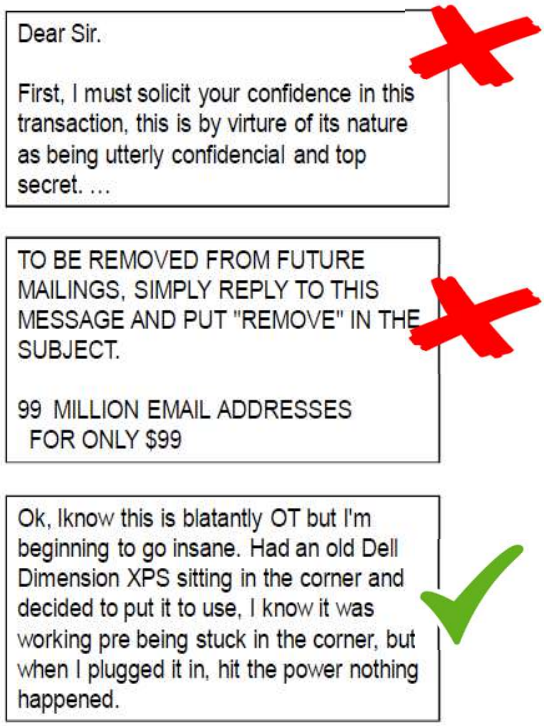
- Classification algorithms เป็นขั้นตอนวิธีประเภทมีผู้ฝึกสอน (supervised learning)
- หมายถึง ต้องมีการนำข้อมูลฝึกสอน (*training set*) ที่รู้คลาส *class* (หรือคำตอบ) ของคอลัมน์เป้าหมาย (*label*) แล้วมาฝึกสอน เพื่อจะสร้าง predictive model ขึ้น
- จากนั้นจะนำ model ที่ได้มานั้นไปใช้งานกับข้อมูลที่เราต้องการจำแนกประเภท
- เรียกกลุ่มข้อมูลที่นำมาทำสอบ (หรือใช้งาน) ว่า *test data(set)*



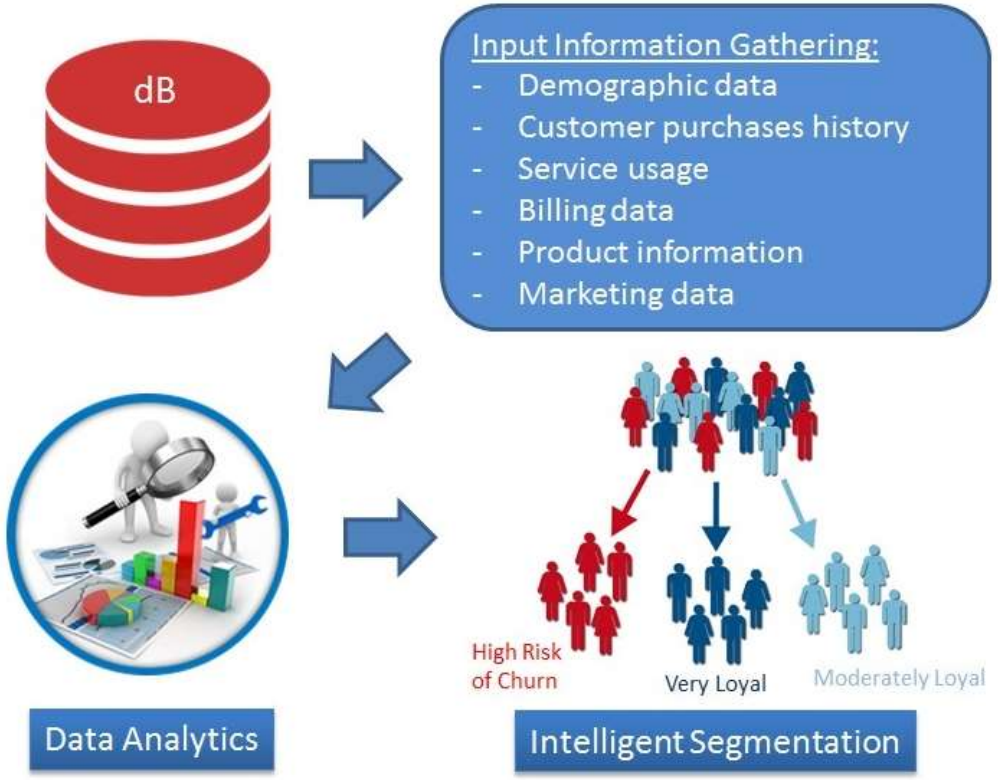
Source: http://bhagyeshvikani.blogspot.com/2015/10/what-is-supervised-learning.html

Common applications of classification problems

การจำแนกอีเมลขยะ
Spam mail checker



Churn modeling (การยกเลิกการใช้บริการ) หรือพนักงานลาออก ฯลฯ



Common applications of classification problems

การวิเคราะห์ความเสี่ยงของเครดิต (Credit risk analysis) e.g. automatic credit approval

- ลูกค้ารายบุคคล
  - ✓ อายุ
  - ✓ รายได้
  - ✓ อาชีพ
  - ✓ จำนวนบุตรที่ต้องอุปการะ
  - ✓ หนี้สินคงค้าง
  - ✓ ประวัติการผ่อนชำระหนี้
  - ✓ ฯลฯ

- ลูกค้านิติบุคคล
  - ✓ debt/equity ratio
  - ✓ debt/capital ratio
  - ✓ interest coverage ratio
  - ✓ gross profit margin
  - ✓ return on equity
  - ✓ working capital
  - ✓ firm size
  - ✓ collateral
  - ✓ etc.

Prediction of non-performing loans

Common applications of classification problems

Prediction of outcomes of bank telemarketing campaign

การจำแนกประเภทเอกสาร Document/Text classification

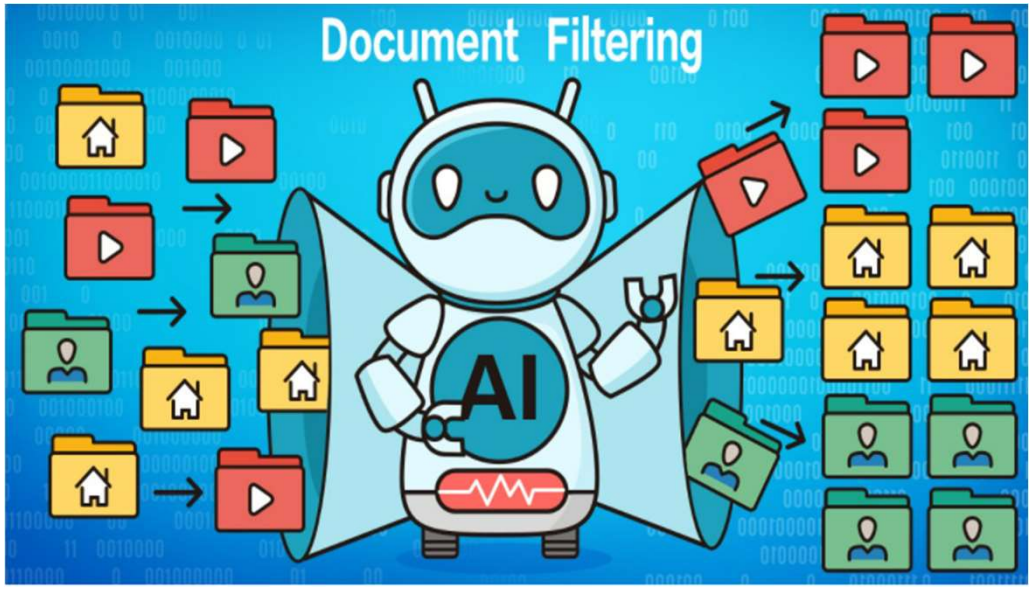
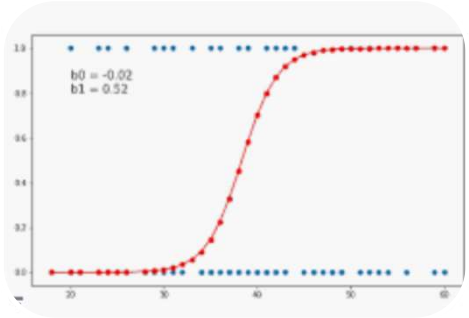
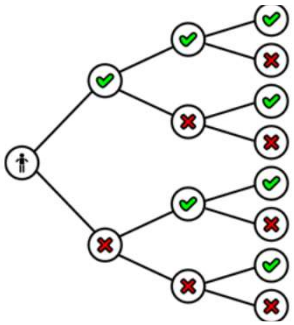


Image source: https://blog.goodaudience.com/document-filtering-finding-a-needle-in-a-haystack-9c00ac4ae1

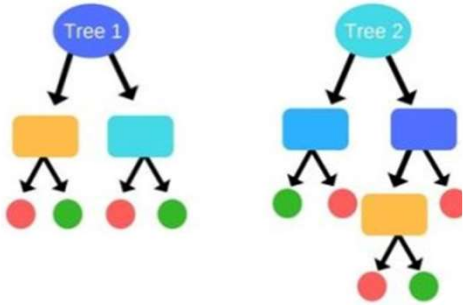
Commonly used classification algorithms



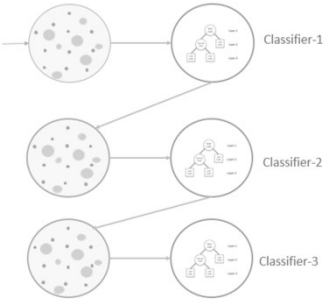
Logistic Regression (LR)



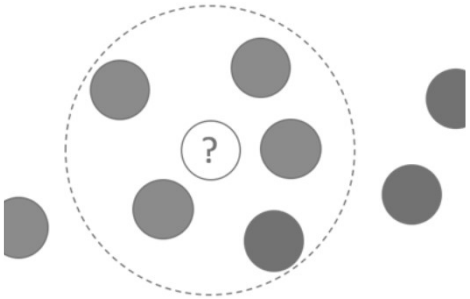
Decision Tree (DT)



Random Forest (RF)



XGBoost



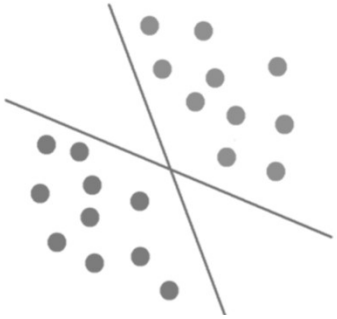
K-Nearest Neighbors (kNN)

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

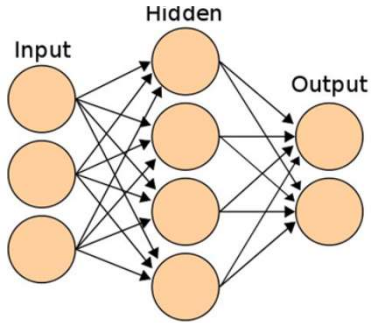
ıg Bayesian probability terminology, the above equation can be writ

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$

Naive Bayes (NB)



Support Vector Machine (SVM)



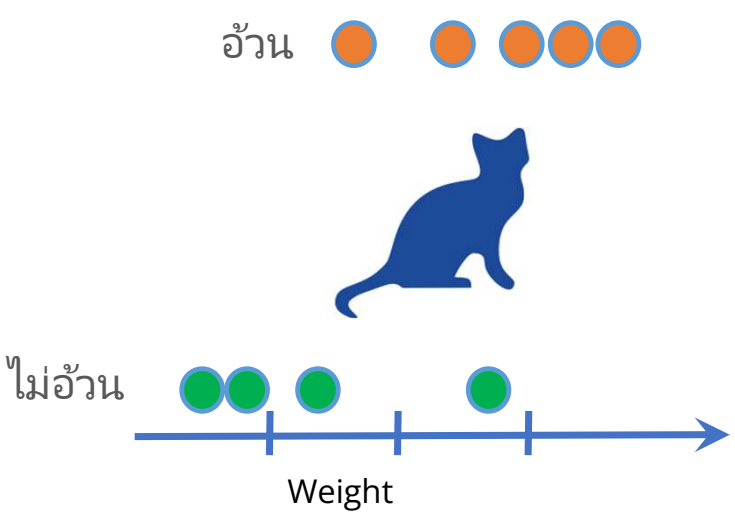
Artificial Neural Network (ANN)

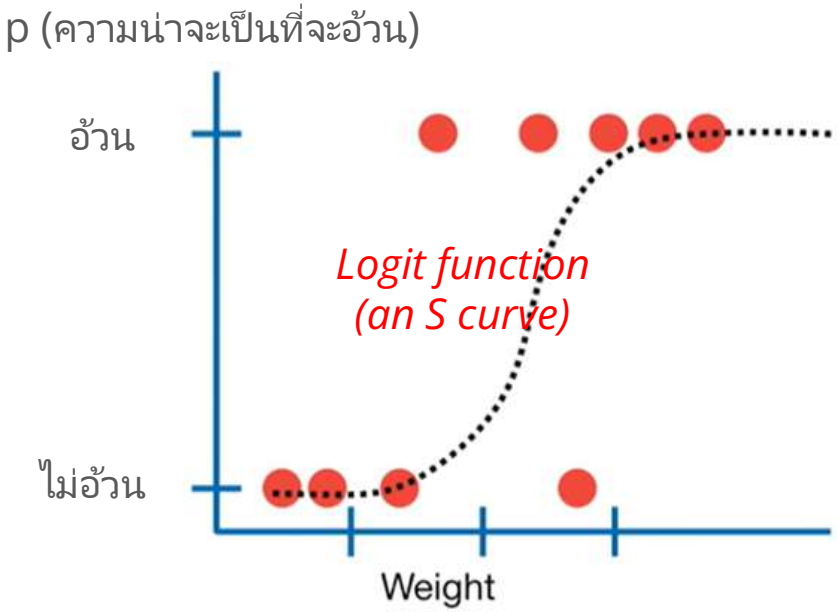
Outline

- Foundation of classification
- Logistics regression
- Decision Tree, Random Forest
- Artificial Neural Networks
- Generic classification modeling process
- Model validation methods
- มาตรวัดประสิทธิภาพของ classification algorithms
- Imbalanced data
- Feature selection
- Hyperparameter tuning

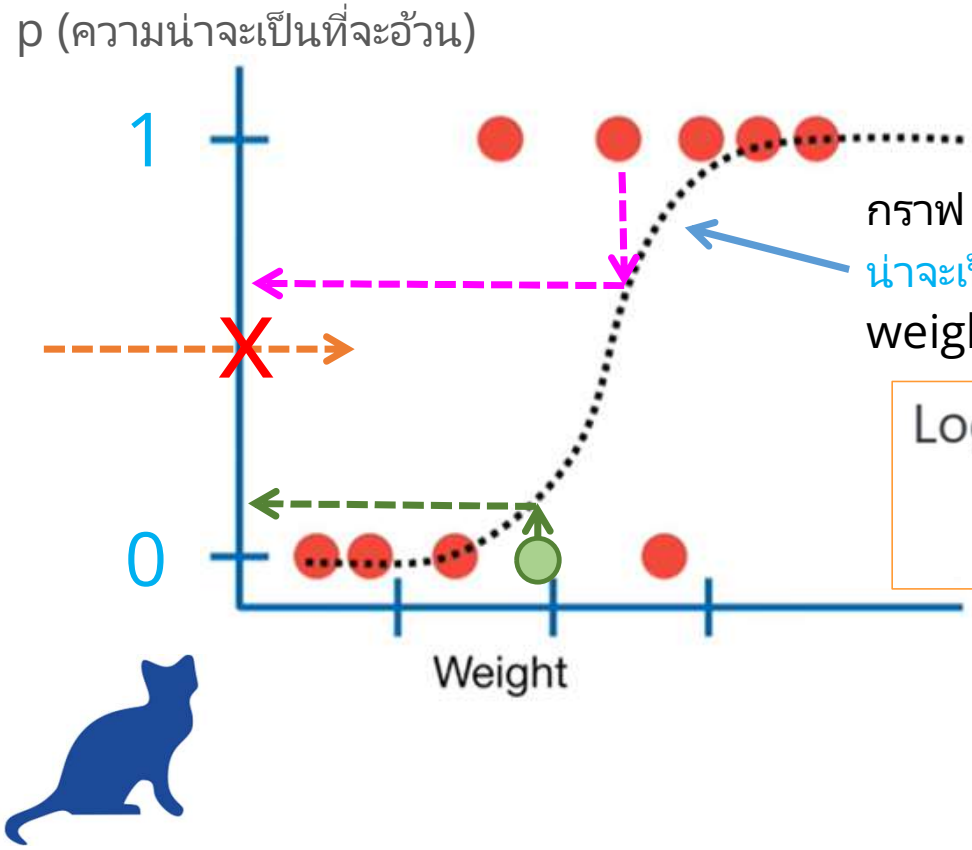
Logistic regression (LR)

- LR เป็นวิธีทางสถิติวิธีหนึ่งในการพยากรณ์*ความน่าจะเป็น (probability)* ของผลลัพธ์ที่ค่าเป็นข้อมูลชนิดไม่ต่อเนื่อง (discrete value) โดยกำหนดด้วยค่าของตัวแปรเข้า
- โมเดล LR ที่นิยมใช้มากในการพยากรณ์ค่า binary outcome (*binary classification*)
  - จริง/เท็จ, เลือก/ไม่เลือก, ออก/ไม่ออก ฯลฯ





Logistic regression (LR)



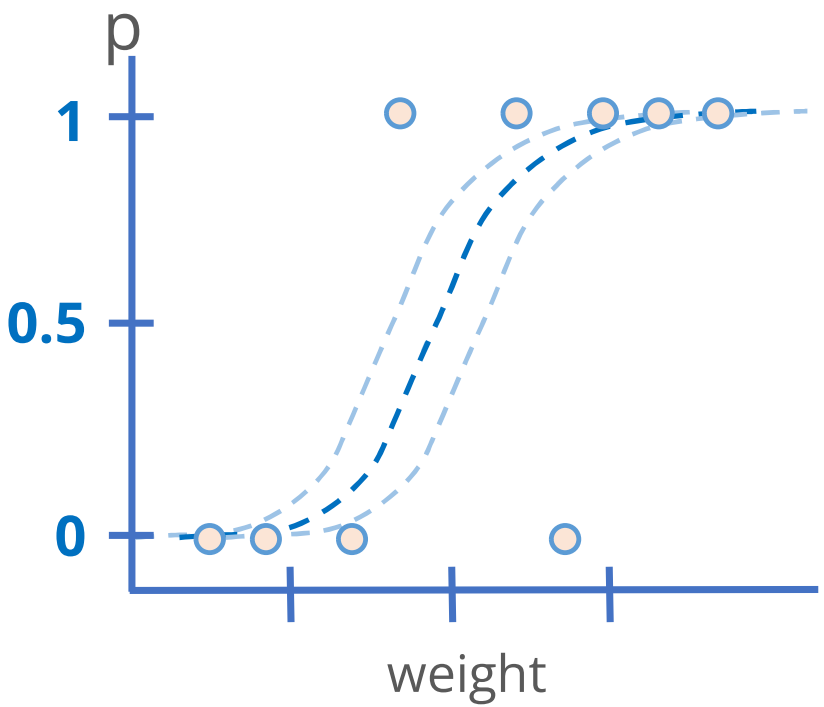
กราฟ S (logit function) นี้บอกความน่าจะเป็นที่แมวจะอ้วน (เมื่อพิจารณาจาก weight ของมัน)

Logit: $\ln[p/(1-p)] = a + BX$

$$p = \frac{e^{a+BX}}{1+e^{a+BX}}$$

How logistic regression works?

Concept of **Maximum likelihood estimation**



- เลือกเส้นความน่าจะเป็น S ที่กำหนดด้วยค่า weight และใช้สังเกตแมวแต่ละตัวเพื่อคำนวณค่า likelihood* ของแมวแต่ละตัว แล้วนำมารวมกัน (ด้วยการคูณกันทั้งหมด) เพื่อให้ได้ likelihood ของแมวทุกตัวจากเส้นนี้
- จากนั้น เลื่อนเส้นความน่าจะเป็นดังกล่าวไปทางขวาเรื่อย ๆ และคำนวณ likelihood นั้นทุกครั้งไป
- จนได้เส้น S ที่ได้ค่า *maximum* likelihood

* *Likelihood refers to finding the best distribution of the data given a particular value of some feature or some situation in the data.*

Advantages and disadvantages of logistic regression

Advantages

- ใช้งานง่าย ตีความเข้าใจได้ไม่ยาก
- ไม่ต้องการสมมติฐานเรื่อง distributions of classes in feature space.
- ประสิทธิภาพดีหากข้อมูลแบ่งแยกตัวแปรได้แบบเชิงเส้น (linearly separable)

Disadvantages

- ตัวแปรอิสระต้องมีค่า multicollinearity ระหว่างกันน้อยหรือไม่มี
- ประสิทธิภาพไม่ดีกับข้อมูลที่ non-linear problems เพราะพื้นผิวการแบ่งเป็นแบบเชิงเส้น (ตรง)
- โดยธรรมชาติแล้วใช้สำหรับ *binomial* classification แต่สามารถขยายให้ใช้กับ *multi-class* ได้ด้วยวิธีการ:
  - One-vs-Rest (OvR) technique หรือ
  - Multinomial LR

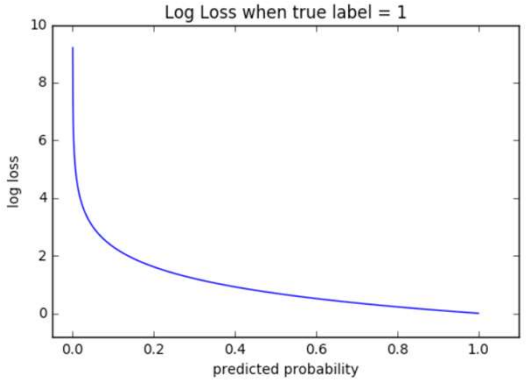
Methods for using LR for multi-class classifier

One-vs-Rest (OvR)

- Splits the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.
- R, G, B
  - Problem 1: R vs [G, B, Y]
  - Problem 2: G vs [R, B, Y]
  - Problem 3: B vs [R, G, Y]
- N problems
- Python: **LogisticRegression(multi_class='ovr')**

Multinomial LR

- change the loss function to cross-entropy loss.
- predict probability distribution to a multinomial probability distribution.
- Python: **LogisticRegression(multi_class=multinomial')**



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*)

Outline

- Foundation of classification
- Logistics regression
- Decision Tree, Random Forest
- Artificial Neural Networks
- Generic classification modeling process
- Model validation methods
- มาตรวัดประสิทธิภาพของ classification algorithms
- Imbalanced data
- Feature selection
- Hyperparameter tuning

Decision Tree (ต้นไม้ตัดสินใจ)

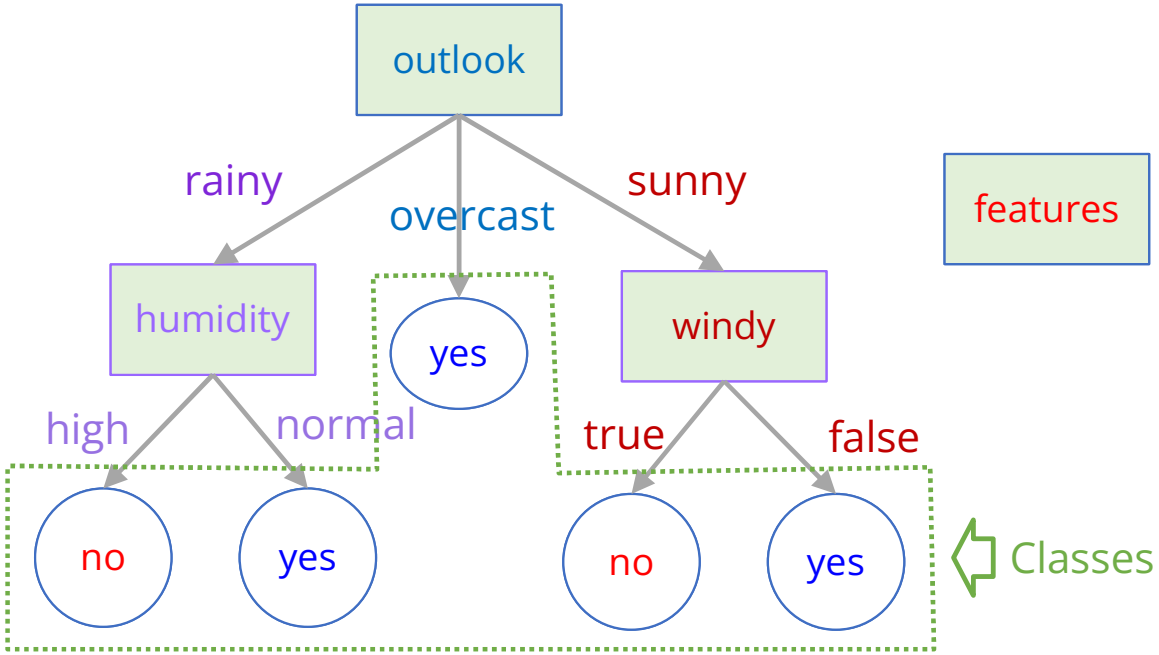
- เทคนิคพื้นฐานสำหรับแก้ปัญหา classification เลือก attribute มาใช้จำแนกประเภท (class) ต่างๆ ตามลำดับ

➢ ตัวอย่างข้อมูล Weather เก็บสภาพภูมิอากาศ 14 วัน เพื่อพิจารณาว่าจะมีการแข่งขันกีฬาได้หรือไม่

Attributes (หรือ Features)

Label

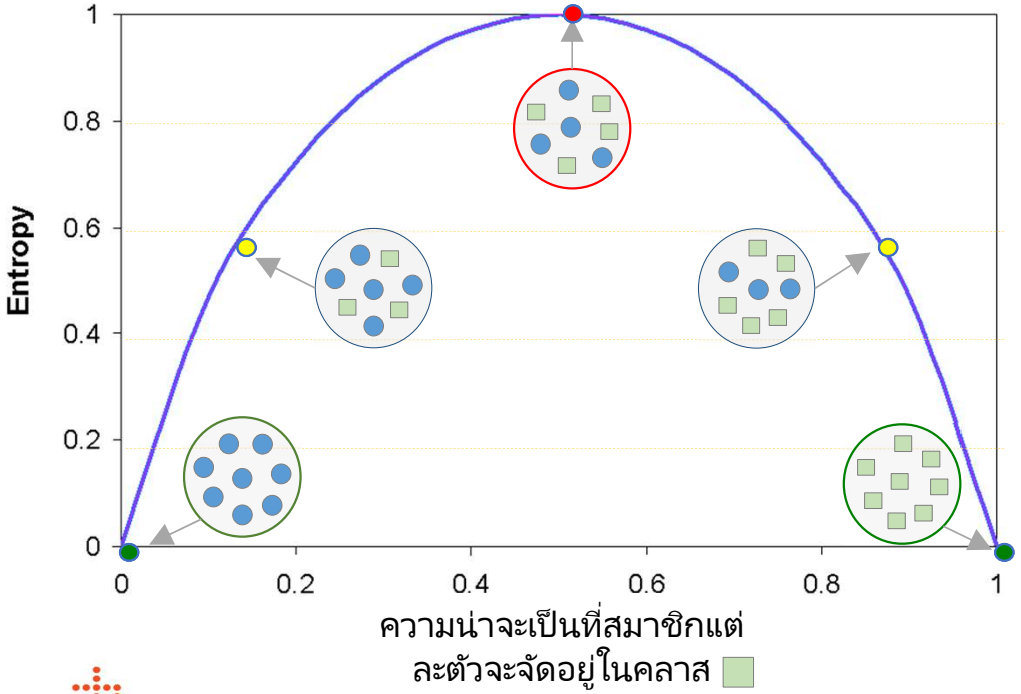
ID	Outlook	Temperature	Humidity	Windy	Play
1	rainy	mild	high	FALSE	yes
2	rainy	cool	normal	FALSE	yes
3	rainy	cool	normal	TRUE	no
4	rainy	mild	normal	FALSE	yes
5	rainy	mild	high	TRUE	no
6	overcast	hot	high	FALSE	yes
7	overcast	cool	normal	TRUE	yes
8	overcast	mild	high	TRUE	yes
9	overcast	hot	normal	FALSE	yes
10	sunny	hot	high	FALSE	no
11	sunny	hot	high	TRUE	no
12	sunny	mild	high	FALSE	no
13	sunny	cool	normal	FALSE	yes
14	sunny	mild	normal	TRUE	Yes

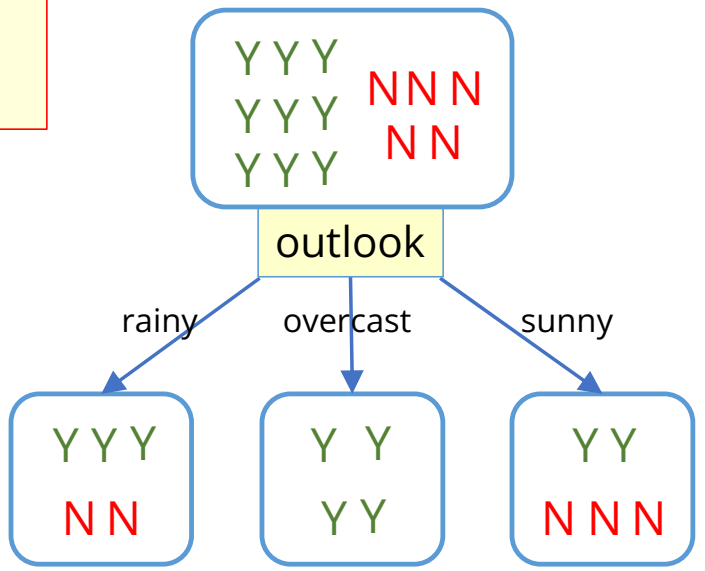


Foundation of Decision tree – *Entropy* and *Information Gain*

- *Entropy* (ความคละหรือความไร้ระเบียบ) is the measure of randomness or unpredictability in the dataset.

ถ้าในกลุ่มมีความคละน้อย Entropy จะต่ำ
ถ้าในกลุ่มมีความคละมาก Entropy จะสูง





E1
(high entropy)

After splitting

E2
(lower entropy)

Information gain
= E1 – E2

สรุปสั้น ๆ .... Decision tree algorithms....

จะพยายามเลือก features มาแบ่งแยก records ต่างๆ เพื่อให้ปลายทาง (leaf nodes) ที่ได้เป็นประเภท (class) ที่ . . .

- มีค่า entropy (ความคละหรือความไร้ระเบียบ) น้อยที่สุด
- นั่นคือมี information gain ที่สูงสุด
- หรือ gini index (จำแนกผิด) น้อยที่สุด

How the decision tree creates the tree?

- Well-known algorithms for decision trees:
  - ID3 (use information gain) (Quinn, 1986)
  - C4.5 (successor of ID3, use gain ratio) (Quinn, 1994)
  - CART (Classification And Regression Tree) (Breiman, 1994)
  - CHAID (Chi-square automatic interaction detection) performs multi-level splits when computing classification trees (Kass, 1980)
  - MARS (Multivariate Adaptive Regression Splines) (Friedman, 1991)

- Steps in ID3 algorithm
  - It begins with the original set S as the root node.
  - On each iteration of the algorithm, it iterates through the very *unused* attribute of the set S and calculates *entropy* and *information gain (IG)* of this attribute.
  - It then selects the attribute which has the smallest *entropy* or largest *information gain*.
  - The set S is then split by the selected attribute to produce a subset of the data.
  - The algorithm continues to recur on each subset, considering only attributes never selected before.

Source: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

Attribute Selection Measures

- Gini index ... วัดความน่าจะเป็นที่ data item หนึ่งๆ จะถูกจำแนกผิด
- Entropy ... วัดความคละข้อมูล (randomness) มีแนวโน้มจะสร้างคลาสขนาดเล็ก (มีสมาชิกน้อย)
- Information gain ... วัดการลดลงของ entropy
- *Gain Ratio* ... ปรับปรุง information gain โดยคำนึงถึงจำนวนข้อมูลและขนาดของแต่ละกิ่งในการเลือก attribute เพื่อให้ทนต่อ (ไม่โดนหลอกด้วย) ข้อมูลประเภทท unique values e.g. ID.
- Chi-square
- etc.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2 \qquad Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

Details about these are in either A.I. or Machine Learning courses.

Strengths and weaknesses of Decision Trees

Strengths

- ผลลัพธ์เข้าใจง่าย
- ทำงานได้เร็ว
- ไม่จำเป็นต้องเตรียมข้อมูลมากมาย
  - Not require normalization of data.
  - Not require scaling of data as well.
  - Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent.

Weaknesses

- ประสิทธิภาพไม่ค่อยดี
- มักพบปัญหา *overfit* ง่าย คือการที่ต้นไม้ที่ได้เหมาะกับข้อมูลที่ใช้สอนมากเกินไป แต่เอาไปใช้งานกับข้อมูลใหม่ที่ยังไม่เคยพบ (unseen data) ไม่ดี
- A small change in the data can cause a large change in the structure of the decision tree causing instability.

Timeline of tree-based classification algorithms

Decision Tree

A graphical representation of possible solutions of a decision based on certain conditions

1987 Quinlan

Bagging

1994 Breiman

Bagging

Bagging (Bootstrap aggregating) is a ensemble predictor that combines several decision trees through a majority voting mechanism

Random Forest

Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees.

2001 Breiman

Boosting

2001 Friedman

Gradient Boosting

Models are built sequentially and employs gradient descent algorithm to minimize the errors from previous models while increasing influence of high-performing models.

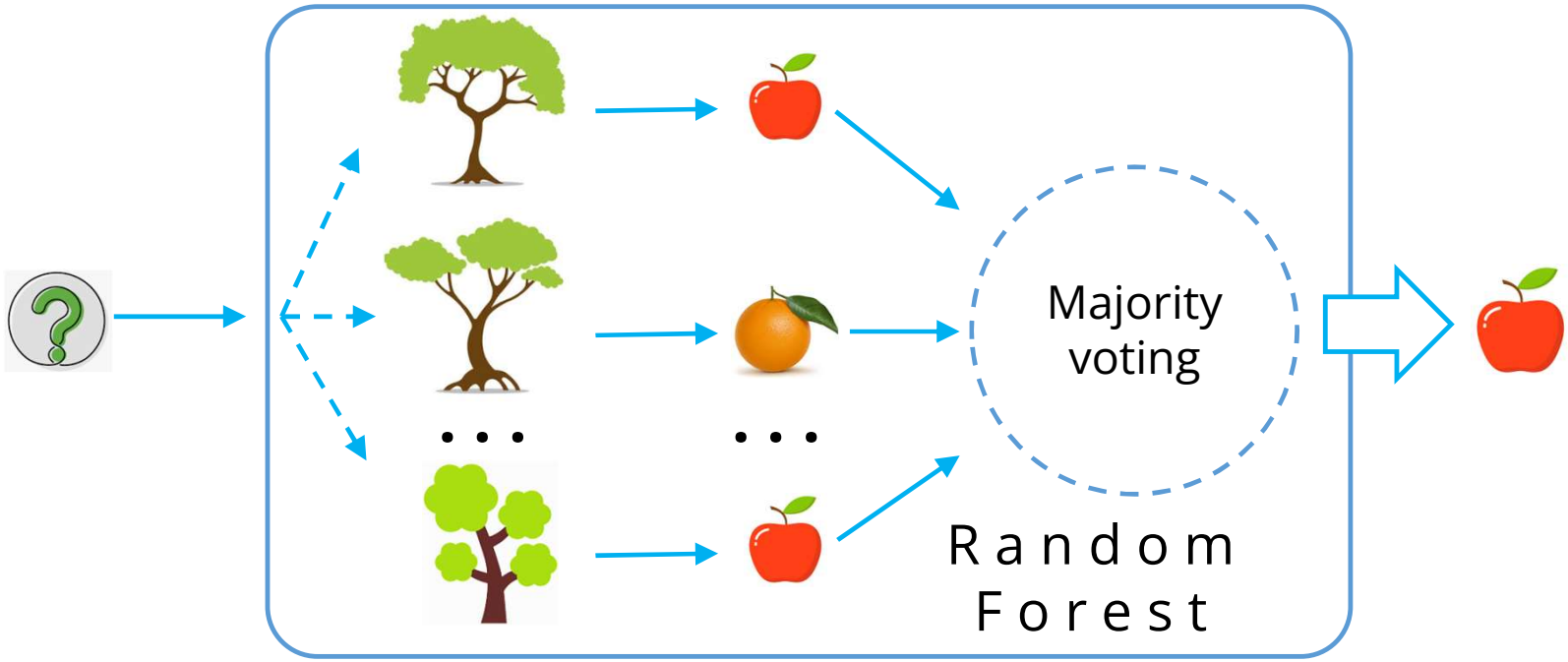
XGBoost

Optimized gradient boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias.

2016 Chen

Random Forest

- *Random forests* คือ เทคนิคการเรียนรู้ที่ใช้ต้นไม้ตัดสินใจจำนวนมากร่วมกัน (bagging) เพื่อทำ classification หรือ regression โดยที่จะสุ่มเลือกบางแถว (rows) และบาง attributes ไปสร้างต้นไม้หลาย ๆ ต้น แล้วหาข้อสรุปโดย...
  - classification จะใช้ majority vote (หรือ ฐานนิยม (mode))
  - regression จะใช้ค่าเฉลี่ย



Concept of Random Forest

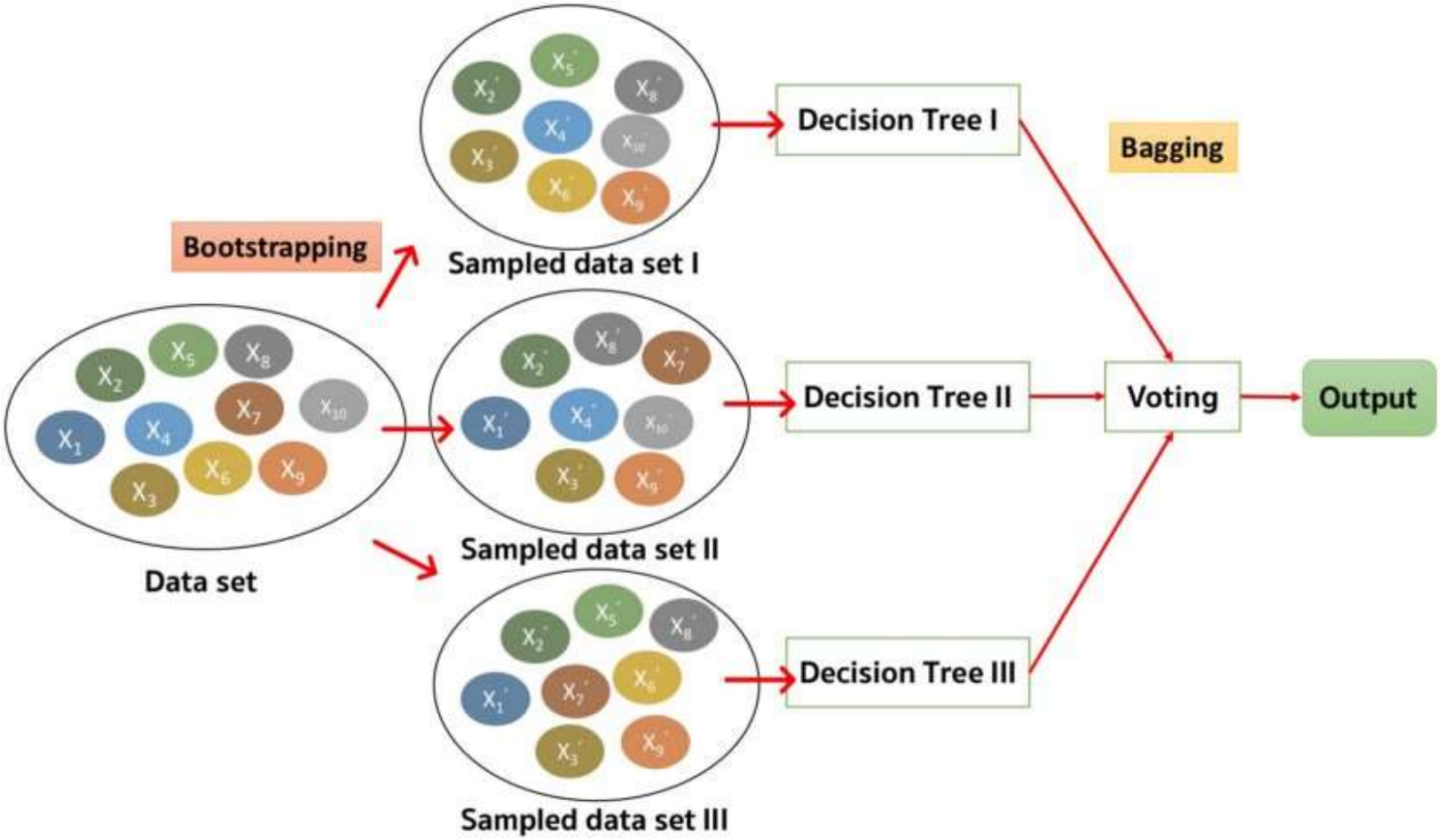
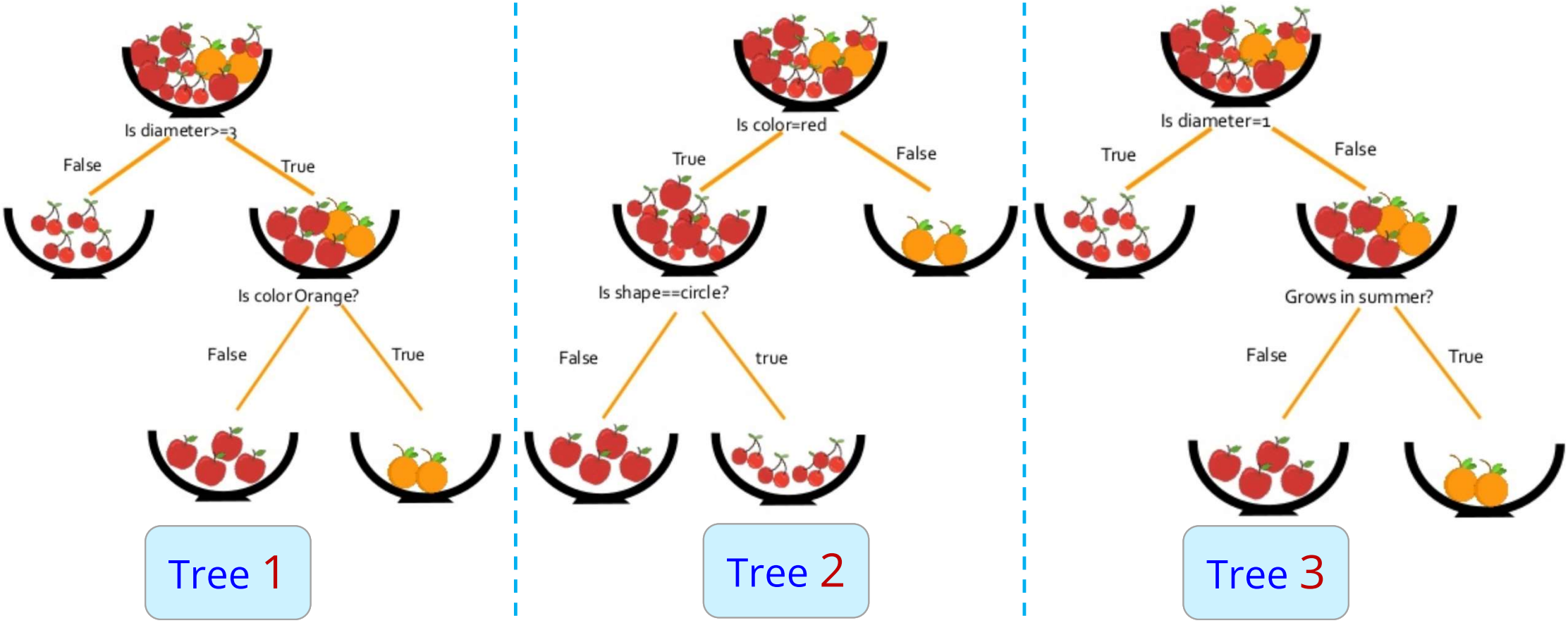
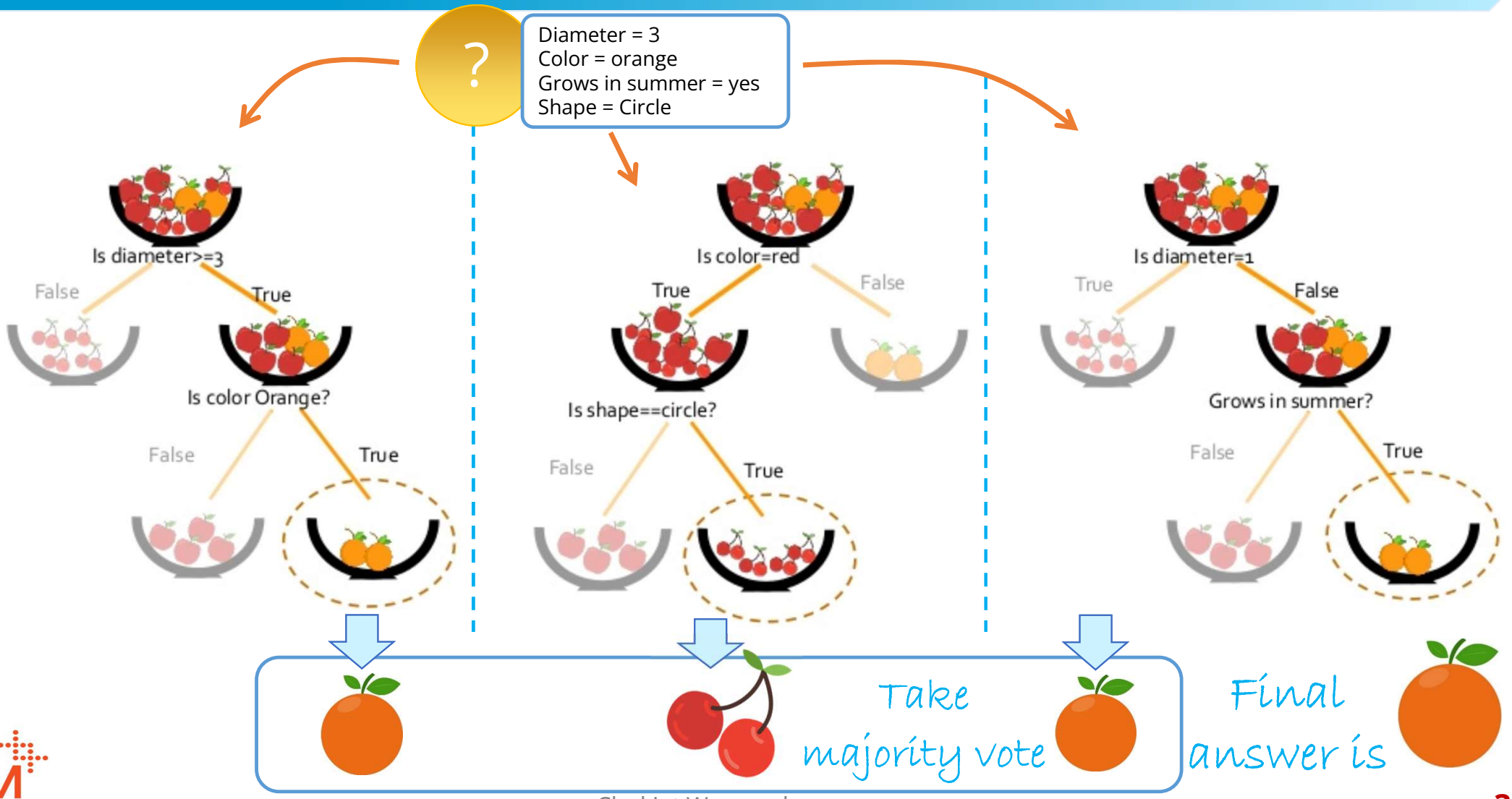


Image source: https://medium.com/@witchapongdaroontham/เจาะลึก-random-forest-part-2-of-รู้จัก-decision-tree-random-forest-และ-xgboost-79b9f41a1c1c

How Random Forest Works? – Example of classifying a fruit



Example: Now, we want to classify this unknown fruit



Random Forest vs Decision Tree

- Decision tree สร้างจากทุก features ของ training set ทั้งหมด
- แต่ random forest จะทำ bagging โดยสุ่มเลือกบางแถว (rows) และบาง features ไปสร้างต้นไม้จำนวนมาก แล้วหาข้อสรุปโดย...
  - classification จะใช้ majority vote (หรือ ฐานนิยม (mode))
  - regression จะใช้ค่าเฉลี่ย

Strengths of RF

- overfitting น้อยกว่า decision tree
- ประสิทธิภาพของ RF จะดีกว่า decision tree ในเกือบทุกกรณี

Weaknesses of RF

- ใช้เวลาทำงานนานกว่า และมีพารามิเตอร์ต้องเซตค่ามากกว่า decision tree
- Random Forest เป็นขั้นตอนวิธีแนว *black box* ไม่สามารถตีความผลของคลาสต่างๆ ที่ได้จากต้นไม้ได้อย่างที่เราได้จาก decision tree

Important Reference

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

sklearn.ensemble.RandomForestClassifier

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='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, 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 controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, 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 : *{"gini", "entropy", "log_loss"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation. Note: This parameter is tree-specific.

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.

XGBoost

- *eXtreme Gradient Boosting* technique uses an optimized gradient *boosting* algorithm through parallel processing, tree-pruning, handling missing values and regularization* to avoid overfitting*.

*regularization is a method of adding some penalty value into the model to reduce overfitting.

* overfitting occurs when the model fits very well with the training data and performs badly with unseen data.

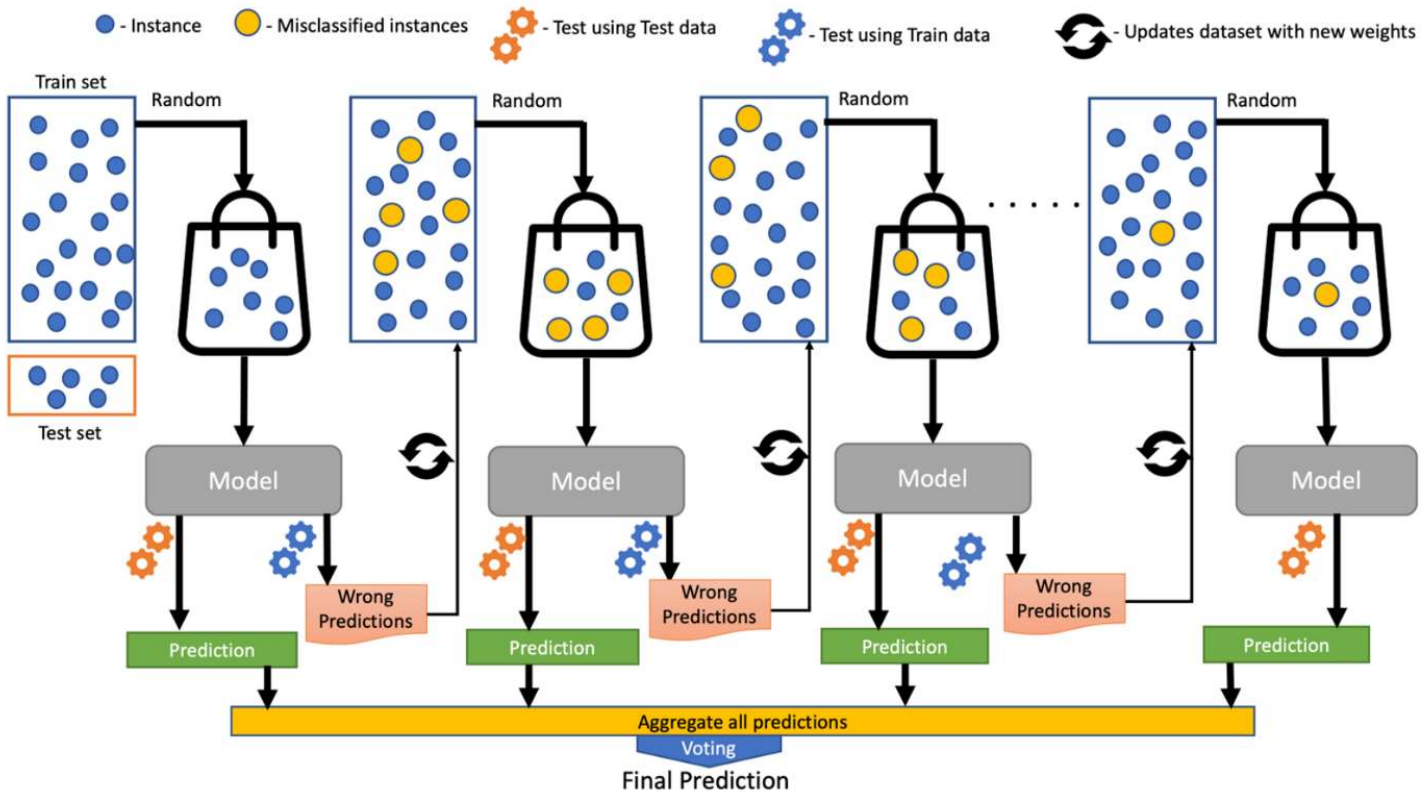


Image Source: https://medium.com/sfu-cspmp/xgboost-a-deep-dive-into-boosting-f06c9c41349

Strengths and weaknesses of XGBoost

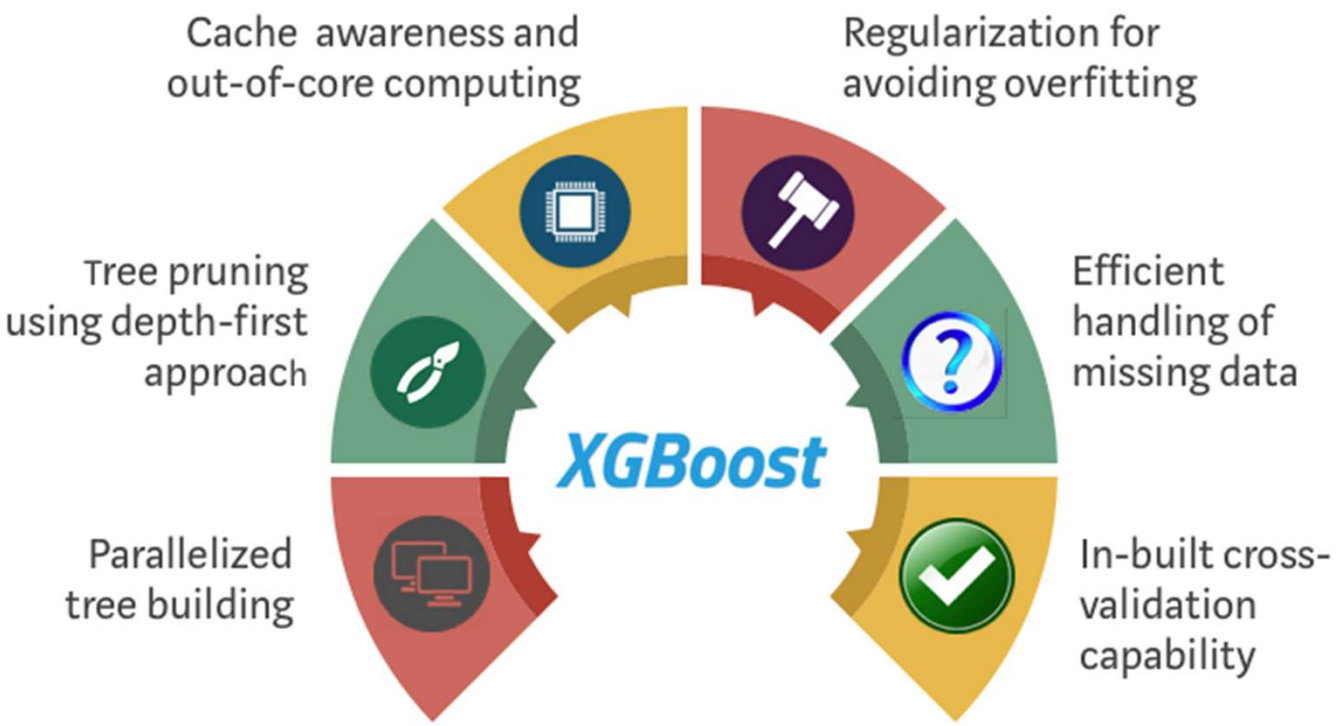


Image source: https://medium.com/time-to-work/xgboost-eafd3beb3fa6

Strengths

- Better performance

Weaknesses

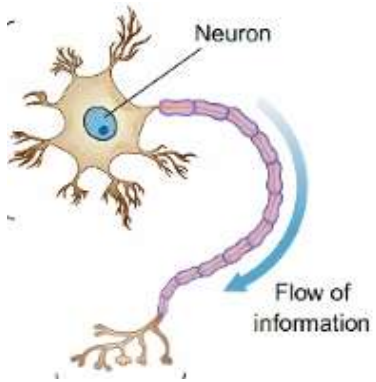
- Complex, difficult to understand.
- Lots of parameters to tune!

More about RandomForest and XGBoost

- sklearn.ensemble.RandomForestClassifier
- sklearn.ensemble.GradientBoostingClassifier.html
- Chen & Guestrin's "XGBoost: A Scalable Tree Boosting System" from https://arxiv.org/abs/1603.02754
- https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d
- https://medium.com/sfu-cspmp/xgboost-a-deep-dive-into-boosting-f06c9c41349

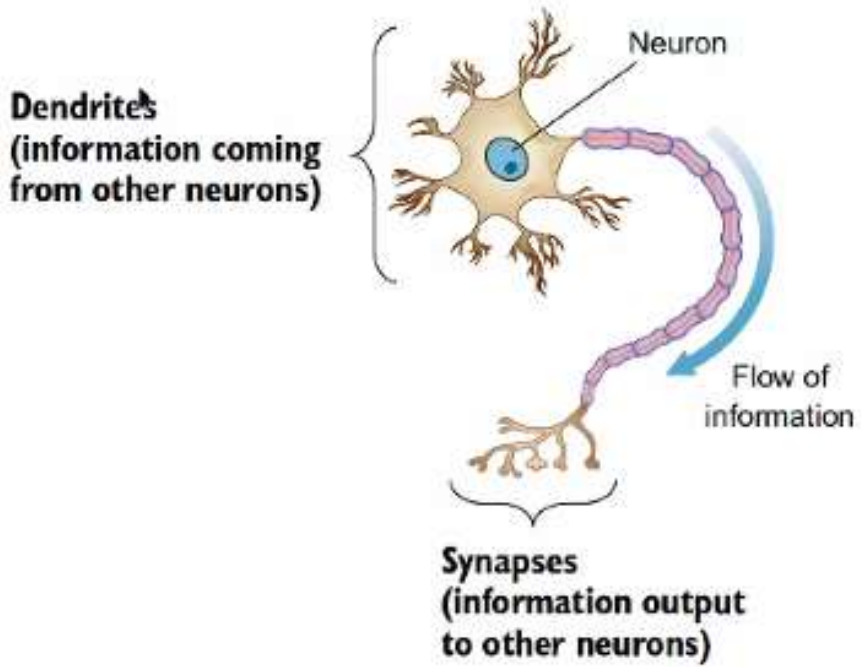
Outline

- Foundation of classification
- Logistics regression
- Decision Tree, Random Forest
- Artificial Neural Networks
- Generic classification modeling process
- Model validation methods
- มาตรวัดประสิทธิภาพของ classification algorithms
- Imbalanced data
- Feature selection
- Hyperparameter tuning

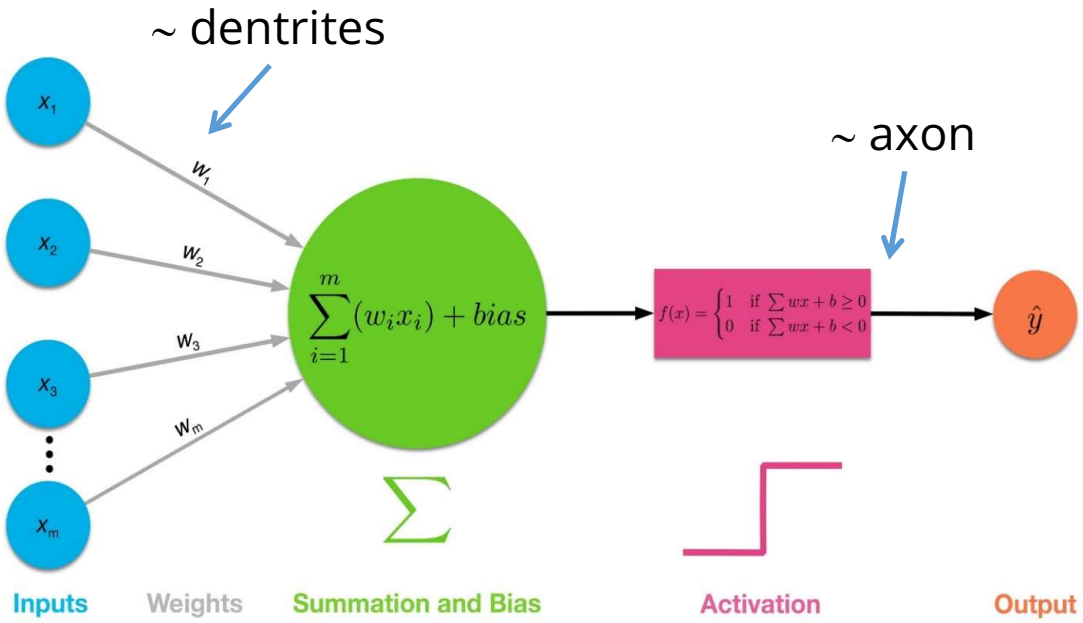


Artificial Neural Networks (ANN)

Mammal's *neuron*

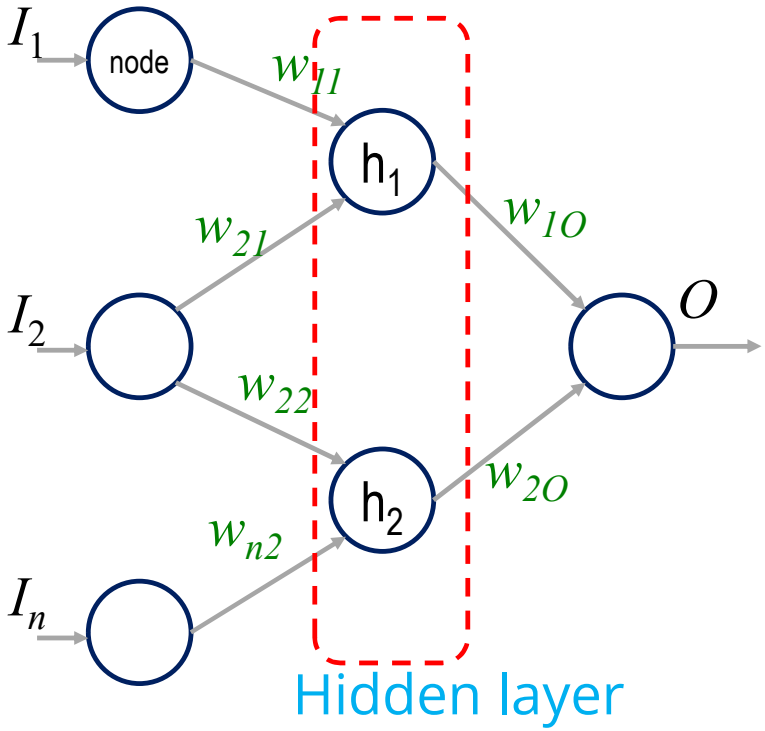


Artificial *neuron*



องค์ประกอบหลักของ ANN

1. Nodes หรือ neurons
2. Interconnections (links) forming architecture (หน้าถัดไป)
3. Activation function of each node
4. Weight of each link
5. Learning algorithms
   เช่น Back Propagation (BP)



- ข้อมูลเข้าต้องเป็นตัวเลข (*numerical* data)
- สามารถมีได้หลายชั้น (layers) แต่พื้นฐานนิยมมี 1 hidden layer
- การเชื่อมต่อพื้นฐานที่นิยมใช้ คือ feed-forward NN (FFNN)

Some Architectures of ANN

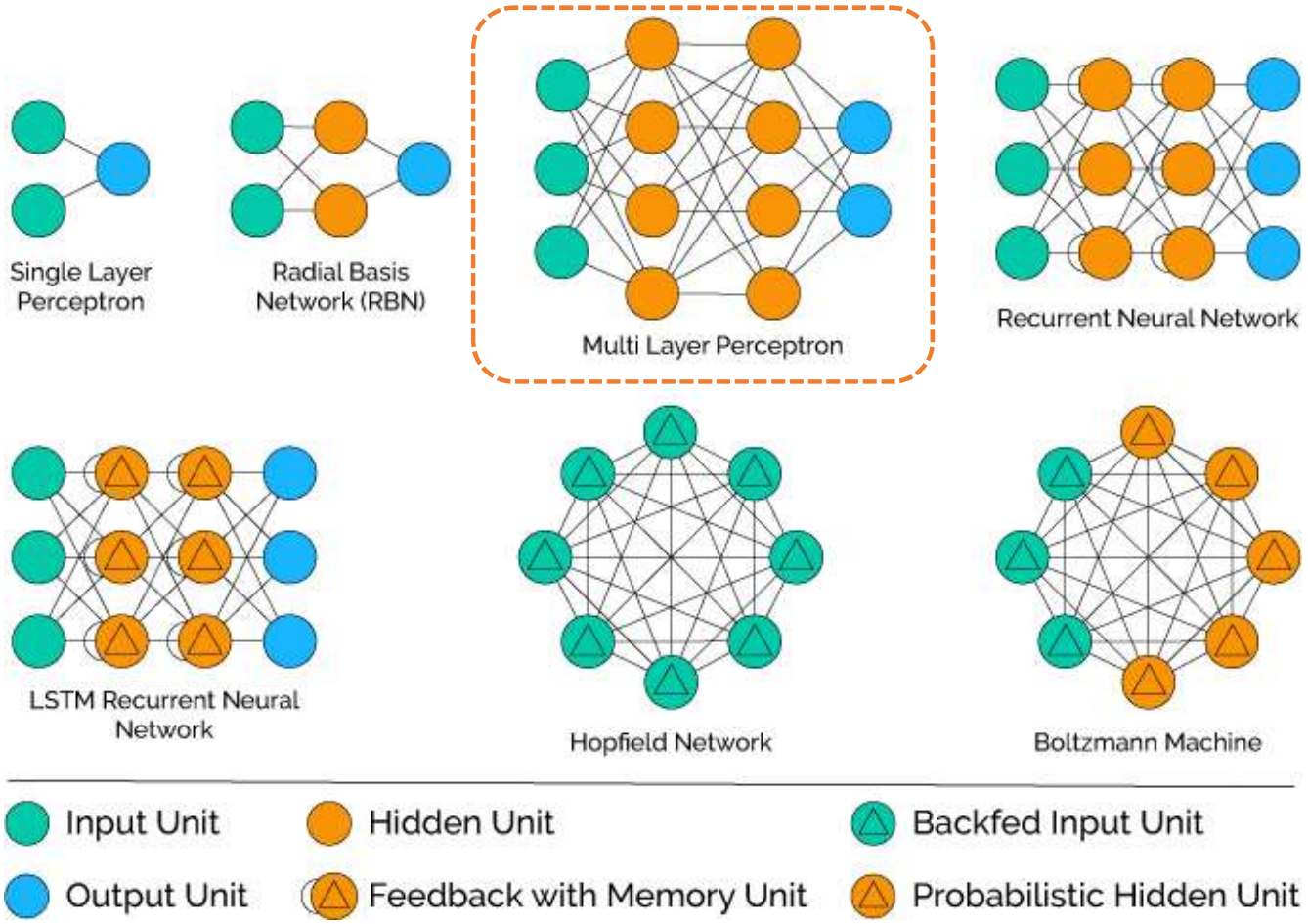
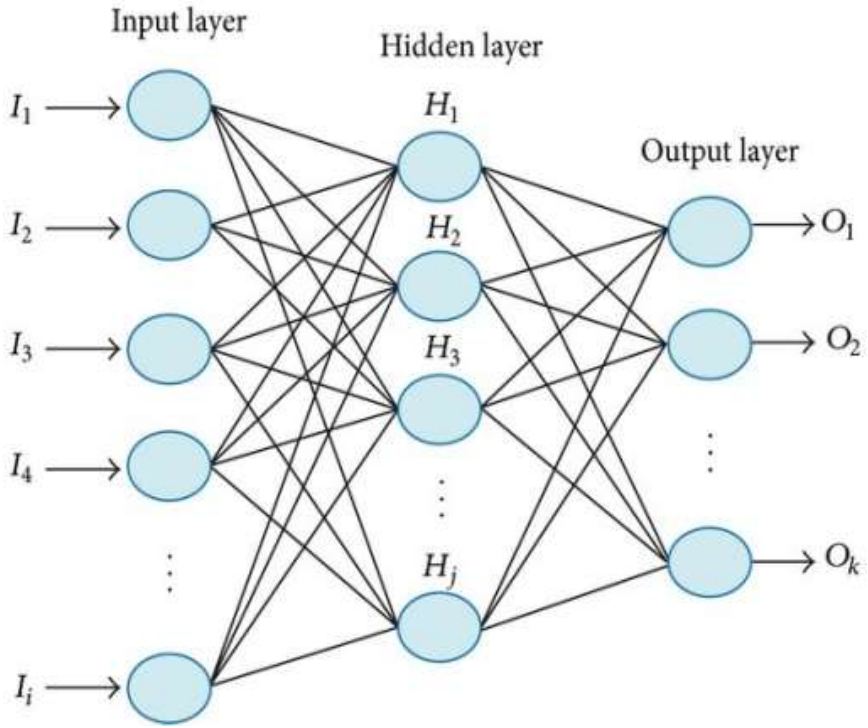
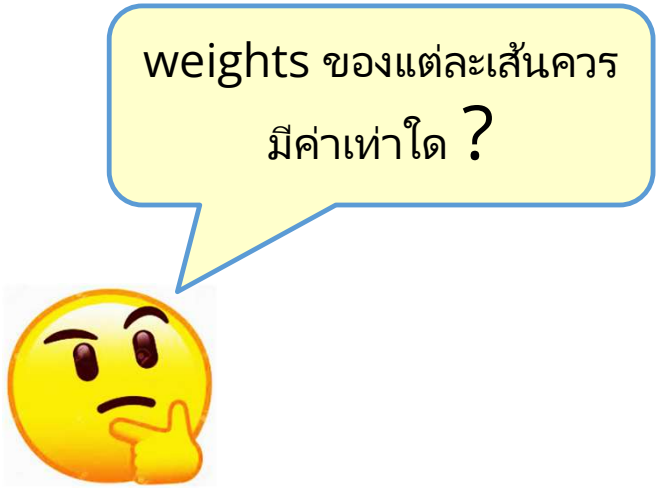


Image source: https://www.xenonstack.com/blog/artificial-neural-network-applications/

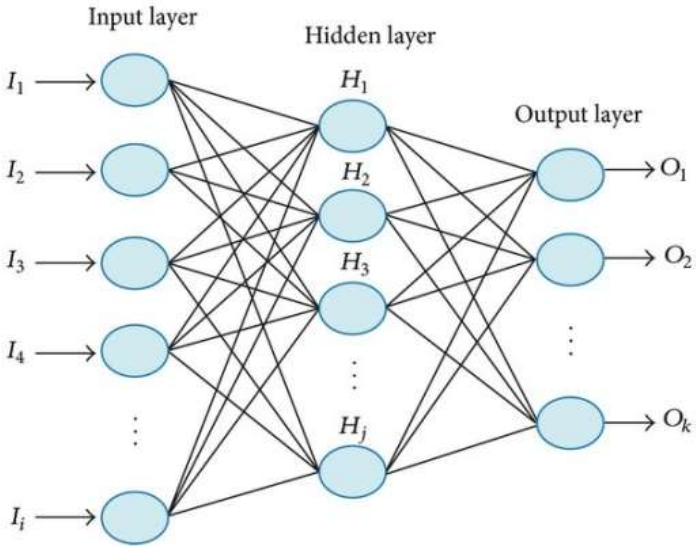
Python's Artificial Neural Networks (MLP)

- Python's sklearn library มี module *MLPClassifier* ชนิด feed-forward neural network (FFNN) หรือ multilayer perceptron (MLP) ที่มากับขั้นตอนการเรียนรู้แบบ backpropagation (BP) learning.
- sklearn's MLPClassifier กำหนดจำนวน hidden layer และจำนวน nodes ในแต่ละ hidden layer ได้ด้วย





Back propagation learning algorithm



- ขั้นตอนการเรียนรู้แบบ Back Propagation (BP) พยายามปรับค่าน้ำหนักของเส้นต่าง ๆ ระหว่าง nodes ใน ANN
- พารามิเตอร์สำคัญของ ANN ได้แก่
  - Training cycles or *max_iter* เป็นจำนวนรอบของการปรับค่าน้ำหนัก
    - default = 200 (sklearn lib)
  - *Learning rate* (อัตราการเรียนรู้) มีผลต่อความเร็วในการลู่เข้าสู่คำตอบ
    - default = 0.001 (sklearn lib)
  - *Momentum* ช่วยเร่งการปรับค่าน้ำหนักเพื่อให้เข้าสู่คำตอบได้เร็วขึ้น และป้องกันการแกว่งข้ามคำตอบไปมา
    - default = 0.9 (sklearn lib)
  - โครงสร้างของ hidden layers
    - จำนวนชั้นและจำนวน nodes ของแต่ละชั้น

ข้อดีข้อด้อยของ ANN

Advantages

- ประสิทธิภาพดี
- มีการพัฒนาปรับปรุงไปมาก

Disadvantages

- โมเดลดูซับซ้อนภายใต้คณิตศาสตร์ซ่อนอยู่
- ปรับค่าพารามิเตอร์ไม่ง่าย
- ใช้เวลานาน (หลายวินาที)
- Output ตีความยากมาก เป็น Blackbox algorithm

Important Reference

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Outline

- Foundation of classification
- Logistics regression
- Decision Tree, Random Forest
- Artificial Neural Networks
- Generic classification modeling process
- Model validation methods
- มาตรวัดประสิทธิภาพของ classification algorithms
- Imbalanced data
- Feature selection
- Hyperparameter tuning

General Steps for Classification Modeling with Scipy

1. Data preparation
2. Feature engineering (identify features and target variable)
3. Split data *training : testing* = 60 : 40 to 90 : 10 (start with 80:20 or 85:15)
4. Feature selection (RFE, PCA, SelectKBest, etc.)
5. Create classifier model e.g. RF, ANN, XGBoost, LR, etc.
6. Train the model using `.fit()`
7. Test the model using `.predict()`
8. Obtain metrices: Accuracy, F1-score and AUC, etc.
9. Analyze the results and then report

Ref: https://www.kaggle.com/code/prashant111/extensive-analysis-eda-fe-modelling/notebook

1. Data preparation

- Handling missing values (imputation)
- Handling outliers
- Discretization

2. Feature engineering

2.1 Identify target variable (label)

2.2 Variable generation/transformation (requires domain expertise!)

- Log transform
  - Reduce or remove skewed data
  - E.g. df[log_price] = np.log(df['Price'])
- Ratio/Delta (with time)
  - Ratio of two or more attributes
  - E.g. Price/earning (P/E) ratio or many other financial ratios
- Composite functions
  - Financial technical indicators
- Categorical encoding (if needed)
- Feature scaling

More: https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10

2A. Categorical Encoding

- Categorical Encoding คือการแปลงข้อมูล categorical (nominal) variable ให้กลายเป็นตัวเลข (also known as *dummy variables*) ที่ต้องทำเพราะขั้นตอนวิธีส่วนใหญ่วิเคราะห์เชิงตัวเลข
- มี 2 วิธี คือ

Country	Age	Salary
India	44	72000
US	34	65000
Japan	46	98000
US	35	45000
Japan	23	34000

1. *Label Encoding*

Country	Age	Salary
0	44	72000
2	34	65000
1	46	98000
2	35	45000
1	23	34000

✓ จำนวนตัวแปรเท่าเดิม ไม่เพิ่ม

✘ แต่ลำดับเลขที่ได้อาจเป็นปัญหา

More: https://datascience.stackexchange.com/questions/9443/when-to-use-one-hot-encoding-vs-labelencoder-vs-dictvectorizor

2. *One-Hot Encoding*

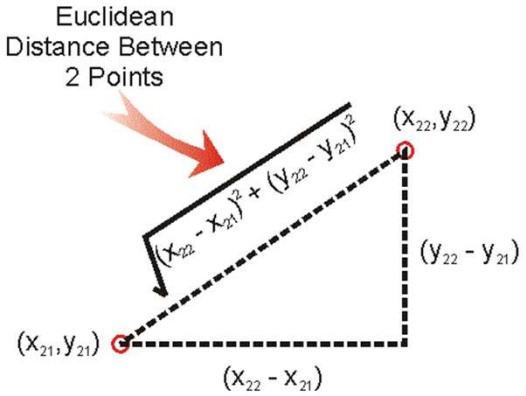
✘ จำนวนตัวแปรเพิ่มขึ้น

0	1	2	Age	Salary
1	0	0	44	72000
0	0	1	34	65000
0	1	0	46	98000
0	0	1	35	45000
0	1	0	23	34000

2B. Feature scaling

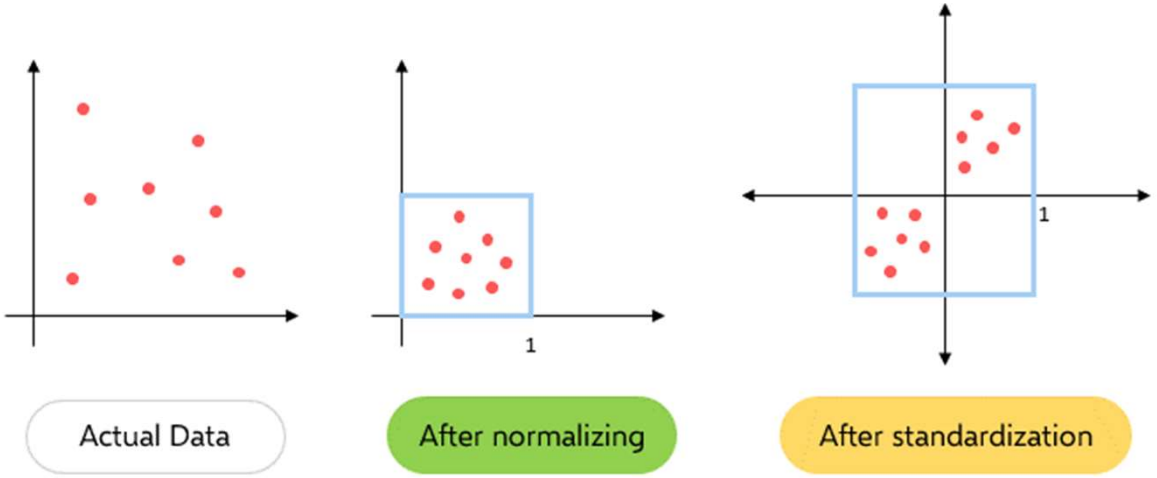
- โดยทั่วไป attribute ต่างๆ มีช่วงของค่าที่แตกต่างกัน เช่น อายุ กับ รายได้
- แต่ machine learning algorithms หลายตัว ใช้ระยะทาง (Euclidian distance) ในการคำนวณ
  - *Strongly recommended*: ANN, KNN
  - Maybe: Naive Bayes
  - Not needed: Tree-based e.g. DT, RF, XGBoost
- Biased if not scale the features

Image source: https://billstainton.com/tale-two-restaurants/big-and-little-sumo/



Common methods of feature scaling

- *Normalization* refers to the rescaling of the features to a range of [min, max] = [0, 1].
- *Standardization* is about transforming the feature values to fall around mean as 0 with standard deviation as 1.



Standardization

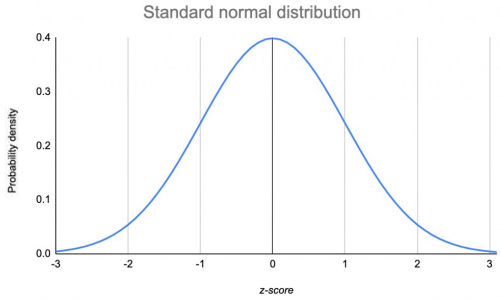
$$z = \frac{x - \mu}{\sigma}$$

with mean:

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

and standard deviation

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$



Normalization เช่น ไปที่ [0,1]

$$z = \frac{x \text{ - } \min(x)}{\max(x) \text{ - } \min(x)}$$

Sklearn's normalization (left) and standardization (right)

```
from sklearn.preprocessing import MinMaxScaler

# fit scaler on training data
norm = MinMaxScaler().fit(X_train['col'])

# transform training data column
X_train_norm = norm.transform(X_train['col'])

# transform testing data column
X_test_norm = norm.transform(X_test['col')
```

```
from sklearn.preprocessing import StandardScaler

# fit on training data column
scale = StandardScaler().fit(X_train['col'])

# transform the training data column
X_train_stand[i] = scale.transform(X_train['col'])

# transform the testing data column
X_test_stand[i] = scale.transform(X_test['col'])
```

Ref: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

Practice: first, install all necessary libraries

d:\>pip install numpy pandas matplotlib seaborn sklearn scipy imblearn

Practice: classification of Social_Network_Ads.csv

- Goal → predict if the customer would purchase the product after viewing the social network ads

```
# Read the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
dataset.describe().applymap('{:.2f}'.format)
```

	User ID	Age	EstimatedSalary	Purchased
count	400.00	400.00	400.00	400.00
mean	15691539.76	37.66	69742.50	0.36
std	71658.32	10.48	34096.96	0.48
min	15566689.00	18.00	15000.00	0.00
25%	15626763.75	29.75	43000.00	0.00
50%	15694341.50	37.00	70000.00	0.00
75%	15750363.00	46.00	88000.00	1.00
max	15815236.00	60.00	150000.00	1.00

```
dataset.head(10)
```

	Gender	Age	EstimatedSalary	Purchased
0	Male	19	19000	no
1	Male	35	20000	no
2	Female	26	43000	no
3	Female	27	57000	no
4	Male	19	76000	no
5	Male	27	58000	no
6	Female	27	84000	no
7	Female	32	150000	yes
8	Male	25	33000	no
9	Female	35	65000	no

Categorical encoding

```
# Encode data: transform each categorical variables into numerical

from sklearn.preprocessing import LabelEncoder

# Create instance of label encoder
le1 = LabelEncoder()

# Assign numerical values and replace in the same column
df['Gender'] = le1.fit_transform(df['Gender'])

# Do similarly for the label Purchased
le2 = LabelEncoder()
df['Purchased'] = le2.fit_transform(df['Purchased'])

df.head(15)
```

	Gender	Age	EstimatedSalary	Purchased
0	1	19	19000	0
1	1	35	20000	0
2	0	26	43000	0
3	0	27	57000	0
4	1	19	76000	0
5	1	27	58000	0
6	0	27	84000	0
7	0	32	150000	1
8	1	25	33000	0
9	0	35	65000	0
10	0	26	80000	0
11	0	26	52000	0
12	1	20	86000	0
13	1	32	18000	0
14	1	18	82000	0

Identify features, split data, and scale the features

```
# Define features X and target variable y
#
X = dataset.iloc[:, 0:3].values
y = dataset.iloc[:, 3].values

# Split the dataset into the Training set and Test set
#
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)

# Scale the features
#
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
X_train = sc.fit_transform(X_train)        # .fit() and then .transform()
X_test = sc.transform(X_test)
```

Perform classification with Decision tree

```
# Decision Tree Classifier
#
from sklearn.tree import DecisionTreeClassifier

# Create the DT and train it with training set
#
dt = DecisionTreeClassifier(random_state=101)
dt.fit(X_train, y_train)

# predict with the testing set and see results
#
dt_pred = dt.predict(X_test)  #Accuracy
dt_pred

# Look at the actual y_test
#
y_test
```

```
array([0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0])
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0])
```

Perform classification with Random Forest

```
from sklearn.ensemble import RandomForestClassifier

# Create RF and train it with Training set
rf = RandomForestClassifier(random_state=101)
rf.fit(X_train, y_train)

# Predict with the Test set and see the results
rf_pred = rf.predict(X_test)
rf_pred
```

Perform classification with Artificial Neural Network (MLP)

```
from sklearn.neural_network import MLPClassifier

# Create MLP and train it with Training set
nn = MLPClassifier(max_iter=1000, random_state=101)
nn.fit(X_train, y_train)

# Predict with the Test set and see the results
nn_pred = nn.predict(X_test)
nn_pred
```

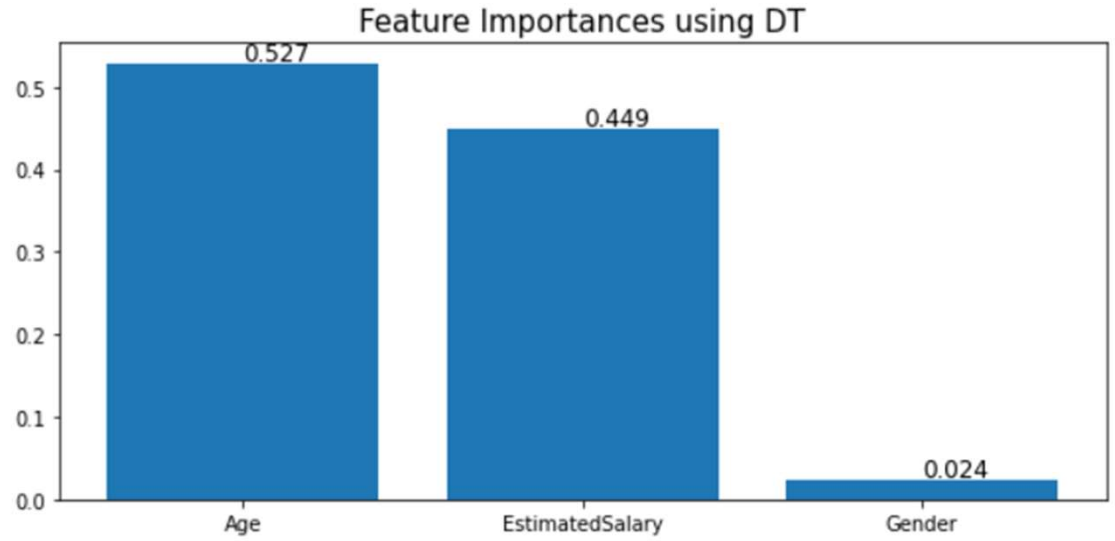
Feature importances

- Tree-based classification algorithms (e.g. decision tree, random forest) can report importance of each feature using model.feature_importances_

```
import matplotlib.pyplot as plt

def show_feature_importances(algo_name, column_names, importances):
    sorted_indices = np.argsort(importances)[::-1] # Sort the feature importance in descending order

    plt.rcParams["figure.figsize"] = (8, 4)
    title = 'Feature Importances using ' + algo_name
    plt.title(title, fontsize=15)
    plt.bar(range(X_train.shape[1]), importances[sorted_indi
    plt.xticks(range(X_train.shape[1]), column_names[sorted_i

    a = importances[sorted_indices]
    for index, data in enumerate(a):
        plt.text(x=index , y=data+.005 , s=f"{data:.3f}" , fo

    plt.tight_layout()
    print(column_names, importances)
    plt.show()
```

```
show_feature_importances('DT', df.columns.values, dt.feature_importances_)
```

```
['Gender' 'Age' 'EstimatedSalary' 'Purchased'] [0.02368942 0.52744149 0.44886909]
```



Parameters of some classification algorithms

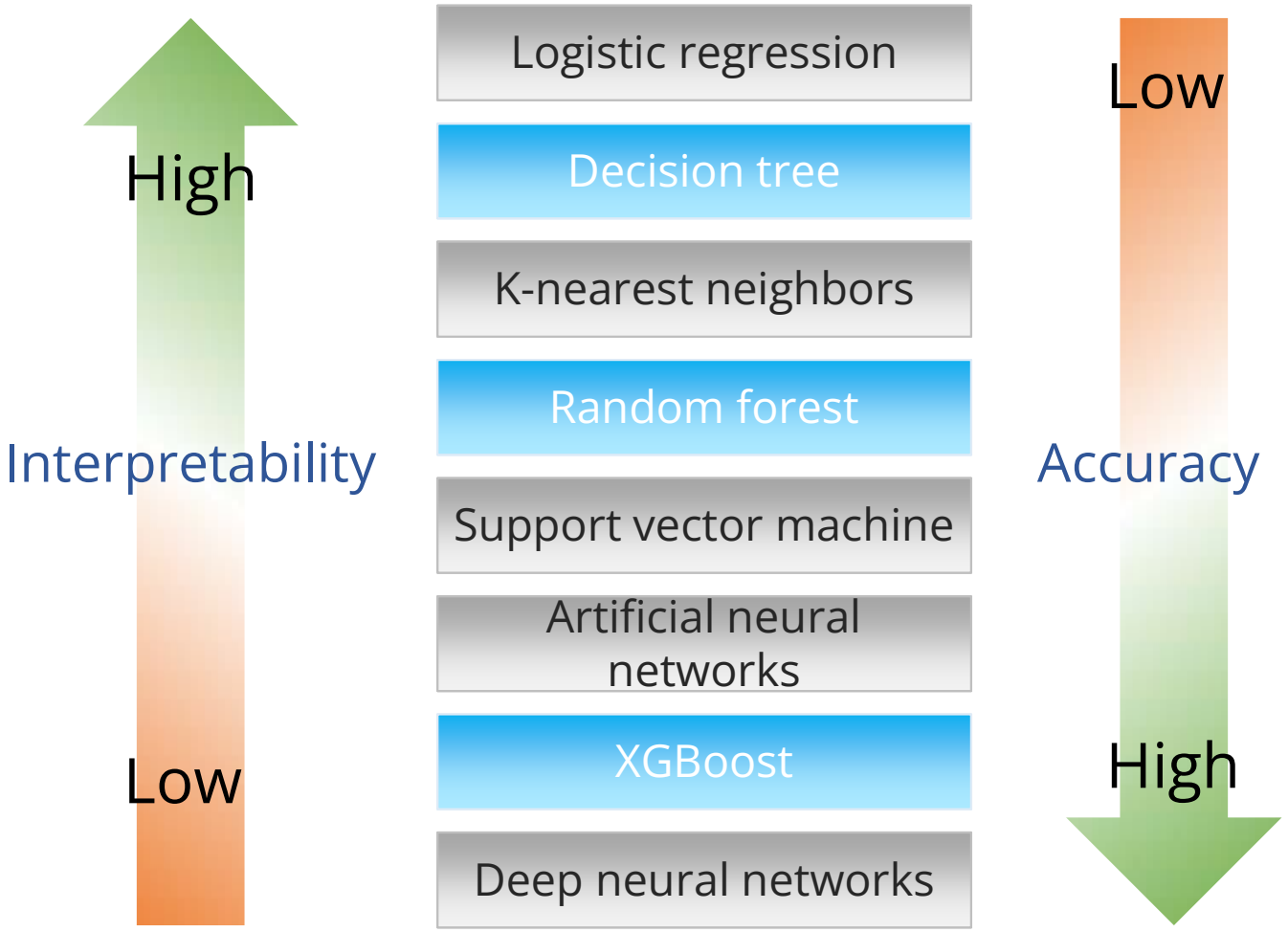
- Decision trees:
  - criterion, max_depth, min_samples_split, min_samples_leaf, max_features
  - More details are here.
- Random forest
  - n_estimators=100, max_depth=none, min_samples_split=2, max_features=auto, min_samples_leaf=1, bootstrap
  - More details are here.
- Artificial neural networks
  - hidden_layer_sizes, activation, solver='adam', max_iter=200, learning_rate='constant', learning_rate_init=0.001, momentum=0.9, etc.
  - More details are here.

Comparison of Classification Algorithms

Algorithm	Primary Domain	Predictors	Power	Raw Implementation	Interpretability	Normalization
Logistic regression	Binary	Numeric	Low	Easy	Good	No
K-NN	Multiclass or binary	Numeric	Medium	Easy	Good	Required
Naïve Bayes	Multiclass or binary	Category	Medium	Medium	Good	Required
Decision tree	Multiclass or binary	Numerical or Category	Medium	Medium	Good	No
Random Forest	Multiclass or binary	Numerical or Category	High	Difficult	Weak	No
SVM	Binary	Numerical or Category	High	Very Difficult	Medium	Yes
Artificial Neural network	Multiclass or binary	Numerical or Category	Very high	Very Difficult	Weak	Yes

Source: https://jamesmccaffrey.wordpress.com/2018/11/07/a-comparison-of-ten-machine-learning-classification-algorithms/

Interpretability vs Accuracy



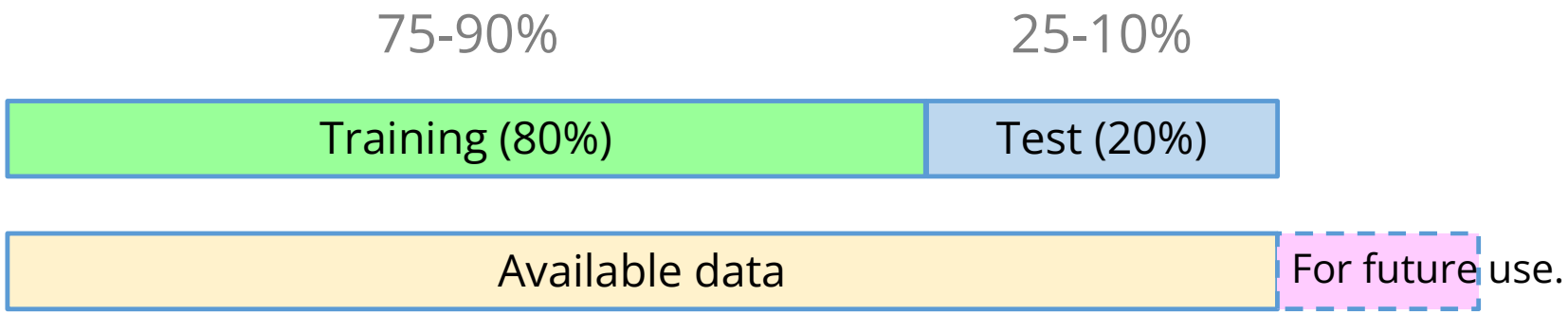
Outline

- Foundation of classification
- Logistics regression
- Decision Tree, Random Forest
- Artificial Neural Networks
- Generic classification modeling process
- Model validation methods
- มาตรวัดประสิทธิภาพของ classification algorithms
- Imbalanced data
- Feature selection
- Hyperparameter tuning

Common methods of model validation

1. Train-Test split (or holdout method)
2. K-Fold cross-validation
3. Stratified K-Fold cross validation
4. Time-series cross validation

Model validation: Train-Test split




```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=101)
```

Pros

- Easy to implement and understand

Cons

- Not whole data are not used for training
- Not suitable for imbalanced data

Model validation: K-Fold cross-validation

Most commonly used K = 10 or 5

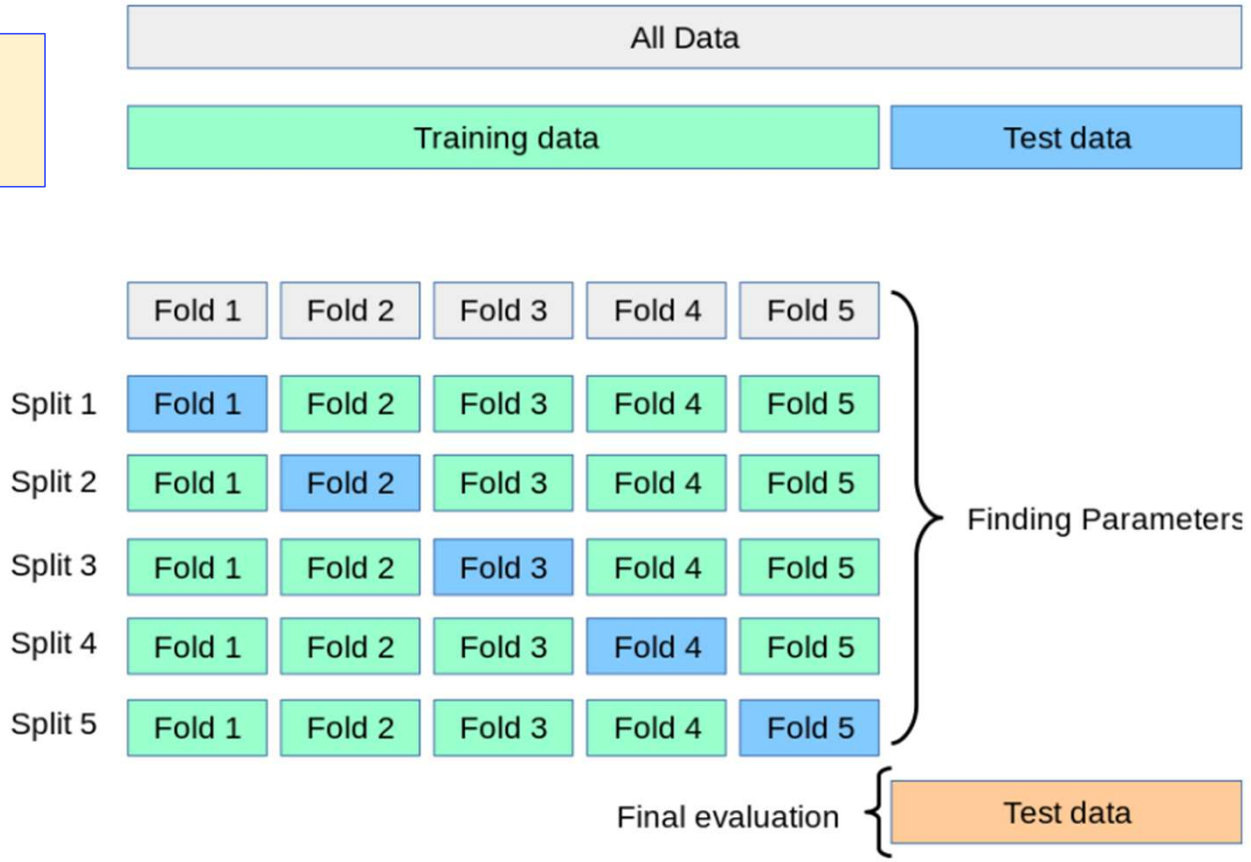


Image source: https://www.analyticsvidhya.com/blog/2021/11/top-7-cross-validation-techniques-with-python-code/

Model validation: K-Fold cross-validation

```
from sklearn.model_selection import KFold, cross_val_score

# prepare the cross-validation procedure
cv = KFold(n_splits=10, random_state=1, shuffle=True) # shuffle before splitting, use if data is sorted

# create model
rf = RandomForestClassifier(random_state=101)

# evaluate model
scores = cross_val_score(rf, X, y, scoring='accuracy', cv=cv, n_jobs=-1)

# report performance
print('Accuracy: %.3f (%.3f)' % (scores.mean(), scores.std()))
```

Pros

- Whole data are used for training

Cons

- Longer computing time
- Not suitable for imbalanced data
- Not suitable for time-series data

Model validation: Stratified K-Fold cross validation

- An improved version of K-Fold validation for imbalanced data so that each of the folds is a good representative of the whole dataset wrt different classes.

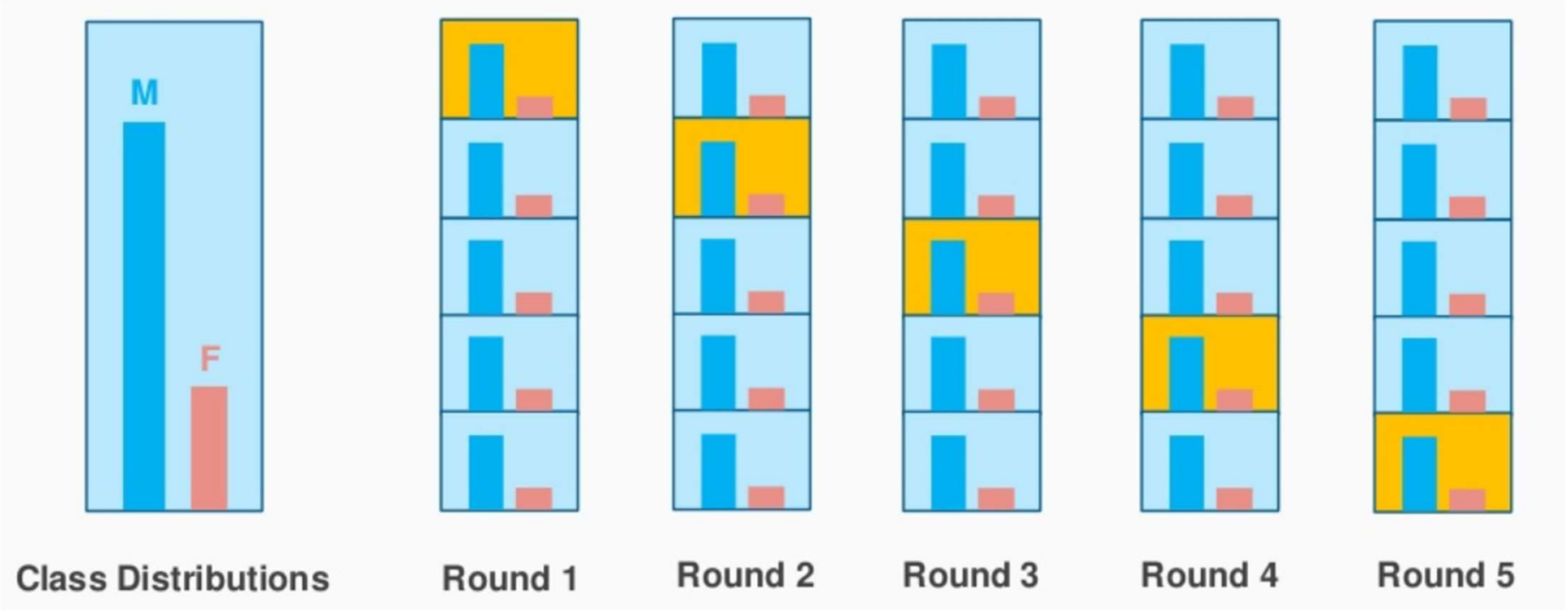


Image source: https://www.analyticsvidhya.com/blog/2021/05/importance-of-cross-validation-are-evaluation-metrics-enough/

Model validation: Stratified K-Fold cross-validation

```
from sklearn.model_selection import StratifiedKFold, cross_val_score

# prepare the cross-validation procedure
cv = StratifiedKFold (n_splits=10)

# create model
rf = RandomForestClassifier(random_state=101)

# evaluate model
scores = cross_val_score(rf, X, y, scoring='accuracy', cv=cv, n_jobs=-1)

# report performance
print('Accuracy: %.3f (%.3f)' % (scores.mean(), scores.std()))
```

Pros

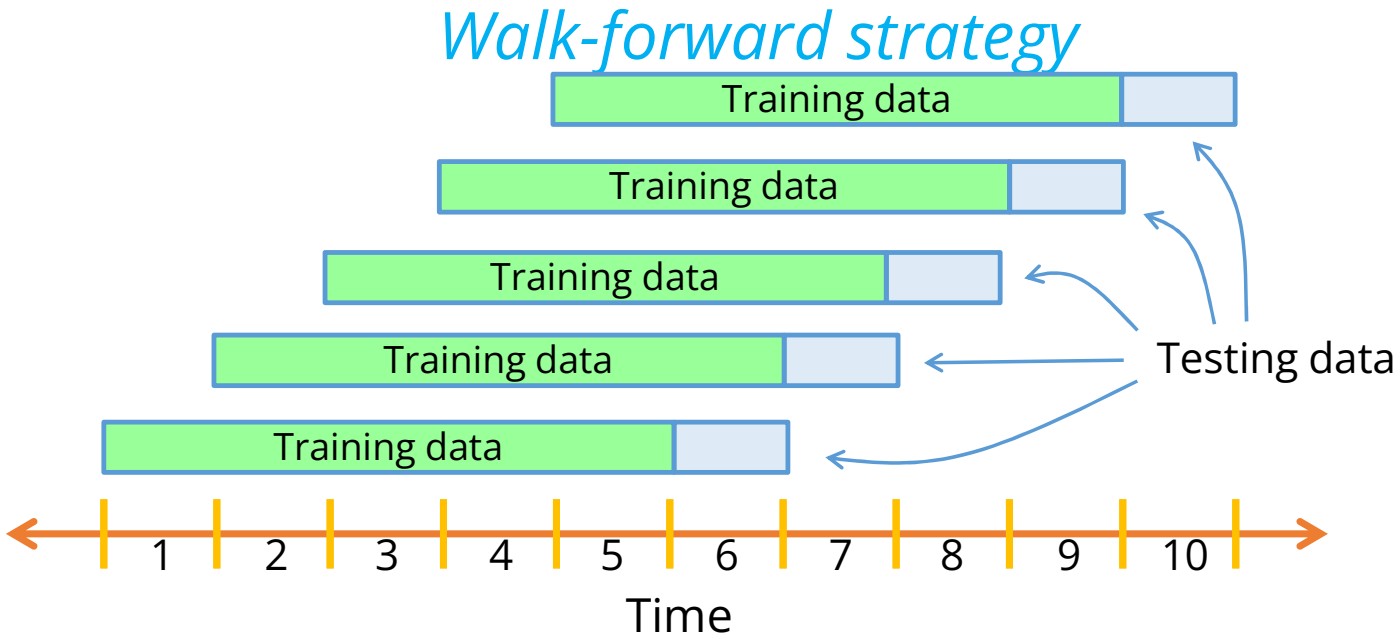
- Better for imbalanced data than KFold

Cons

- Longer computing time
- Not suitable for time-series data

Model validation: Time-series cross validation

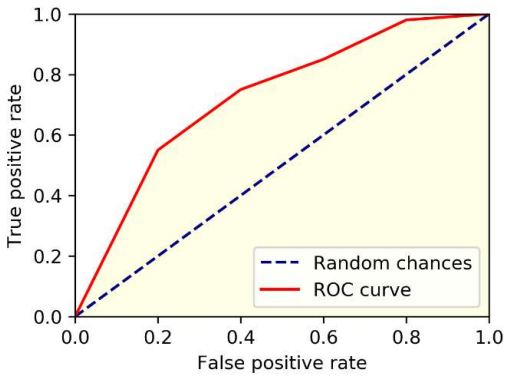
- Data points are at adjacent time periods has potential correlation between observations.
- The order of the data is very important.



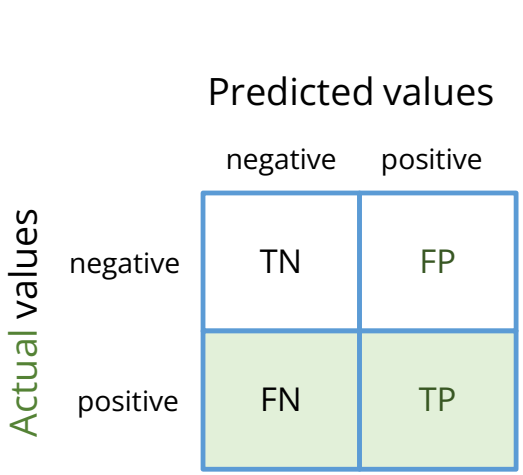
- Note that Python sklearn has TimeSeriesSplit() but it's not walk-forwarded.

Outline

- Foundation of classification
- Logistics regression
- Decision Tree, Random Forest
- Artificial Neural Networks
- Generic classification modeling process
- Model validation methods
- มาตรวัดประสิทธิภาพของ classification algorithms
- Imbalanced data
- Feature selection
- Hyperparameter tuning

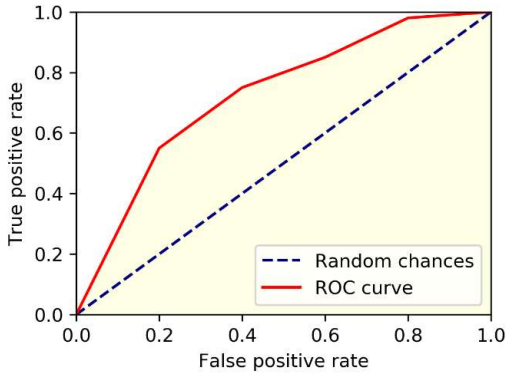


มาตรวัดประสิทธิภาพ (Performance metrices) ของ classification algorithm



Confusion matrix

- Accuracy
- Recall
- Precision
- F-score



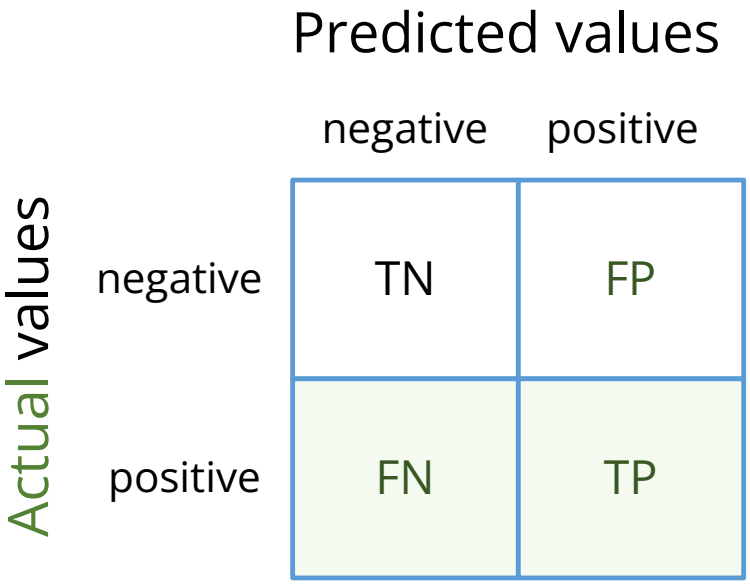
Receiver Operating Characteristic (ROC) curve

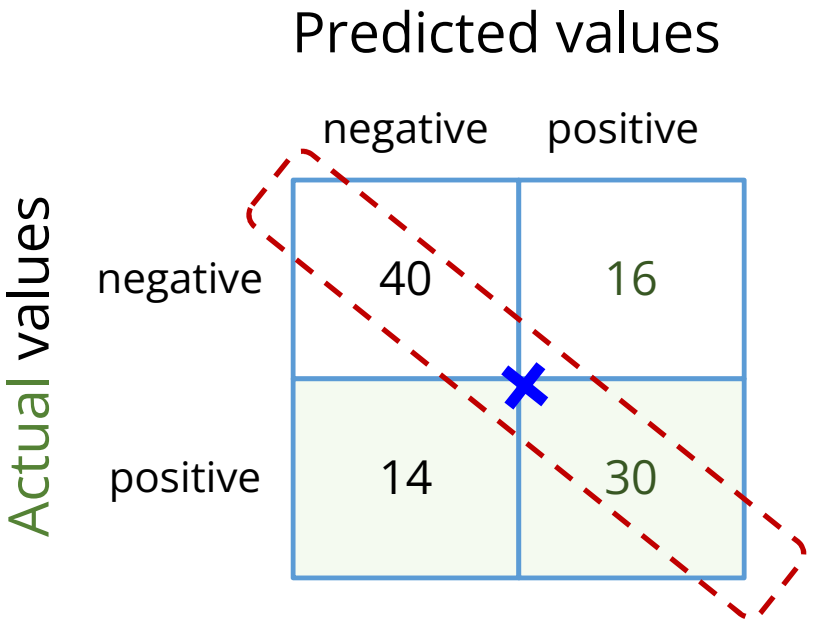
AUC (Area under ROC)

มาตรวัดประสิทธิภาพของ Classification algorithm

- ***Confusion matrix***
  - True negative (TN), False positive (FP)
  - False negative (FN), True positive (TP)

Accuracy = จำนวนที่ทำนายถูกรวมของทุกกลุ่ม / จำนวนทั้งหมด
= (TP + TN) / จำนวนทั้งหมด
= (40 + 30) / (40 + 16 + 14 + 30)
= 70%

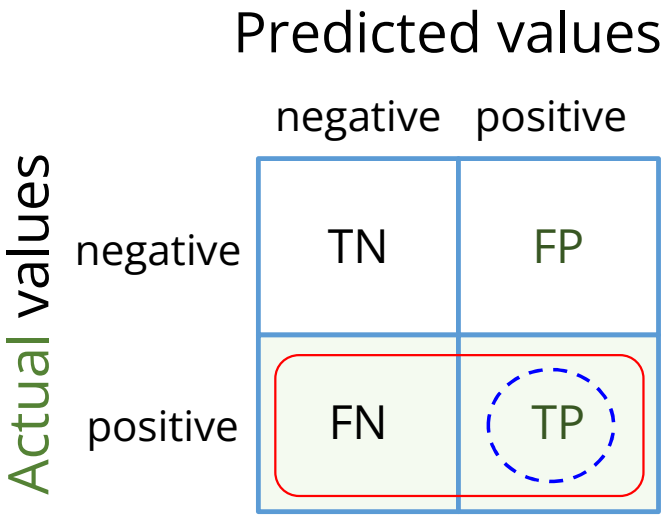




More: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

มาตรวัดประสิทธิภาพของ Classification algorithm

- ***Recall*** (or *True Positive Rate* or *Sensitivity*) หมายถึงความสามารถในการหา positive items ทั้งหมดใน dataset



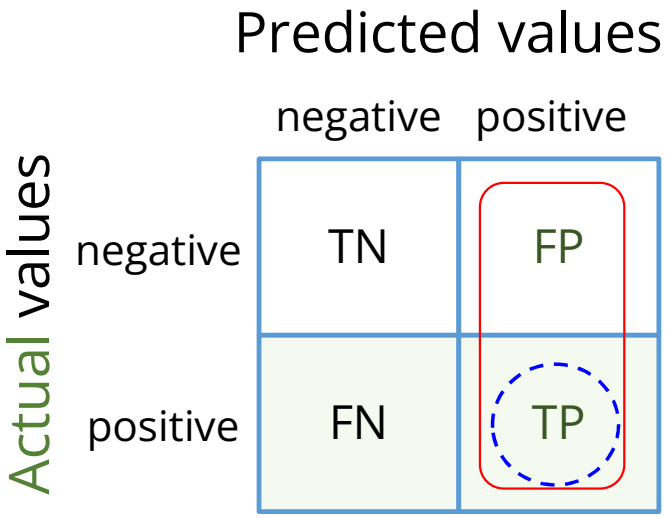
$$recall = \frac{TP}{TP + FN}$$

$$= \frac{\bigcirc}{\square}$$

- *Recall* มีความสำคัญมากในกรณี เช่น
  - ในการ scan ใบหน้าผู้โดยสารขึ้นเครื่องบิน การระบุผู้ก่อการร้ายได้ครบจากผู้ก่อการร้ายจริง ๆ ทั้งหมดที่ขึ้นเครื่อง
  - หรือในการหา Fraud ในธุรกรรมการเงิน สามารถหาธุรกรรมที่ฉ้อโกงทั้งหมดได้ครบหรือไม่

มาตรวัดประสิทธิภาพของ Classification algorithm

- ***Precision*** หมายถึง สัดส่วนของการ items ที่เป็น positive จริง ๆ จากทั้งหมดที่เราระบุว่า positive



$$precision = \frac{TP}{TP + FP}$$

$$= \frac{\bigcirc}{\square}$$

- ต.ย. *Precision* เช่น ในการ scan ผู้โดยสารขึ้นเครื่องบิน การระบุผู้ก่อการร้ายได้ถูกต้อง (คือเป็นผู้ก่อการร้ายจริง ๆ) ทั้งหมดจากคนที่ระบุ (หรือคาดว่า) เป็นผู้ก่อการร้าย
- หรือในการหา Fraud ในธุรกรรมการเงิน สามารถระบุธุรกรรมที่ฉ้อโกงได้ถูกต้อง คือฉ้อโกงจริง ๆ (ไม่ใช่ไประบุธุรกรรมปรกติว่ากลายเป็นฉ้อโกงไป)

F-score หรือ F-measure

- Recall เน้นที่ false-negative (FN) ส่วน Precision เน้นที่ false-positive (FP)
- F-score ให้ความสำคัญของทั้ง false positive และ false negative
- คำนวณโดยคิดค่าเฉลี่ยฮาร์มอนิก (harmonic mean) ของทั้ง precision และ recall

$$F = \frac{2}{\frac{1}{recall} + \frac{1}{precision}}$$

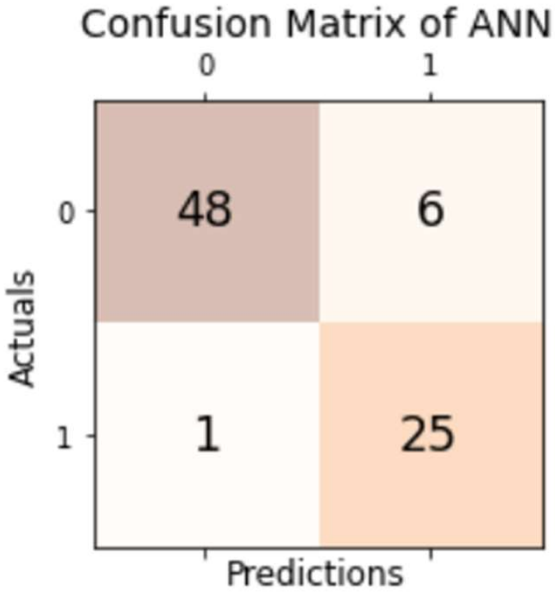
$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

ยิ่งมากยิ่งดี

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

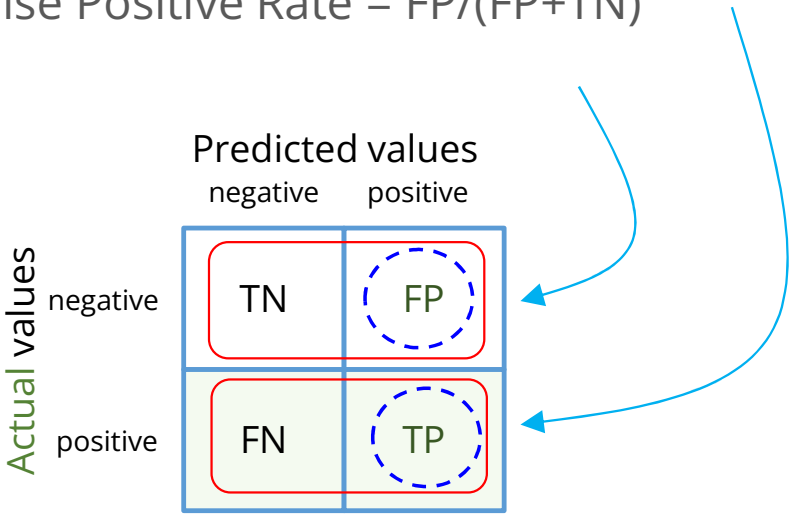
Visualize confusion matrix with matplotlib

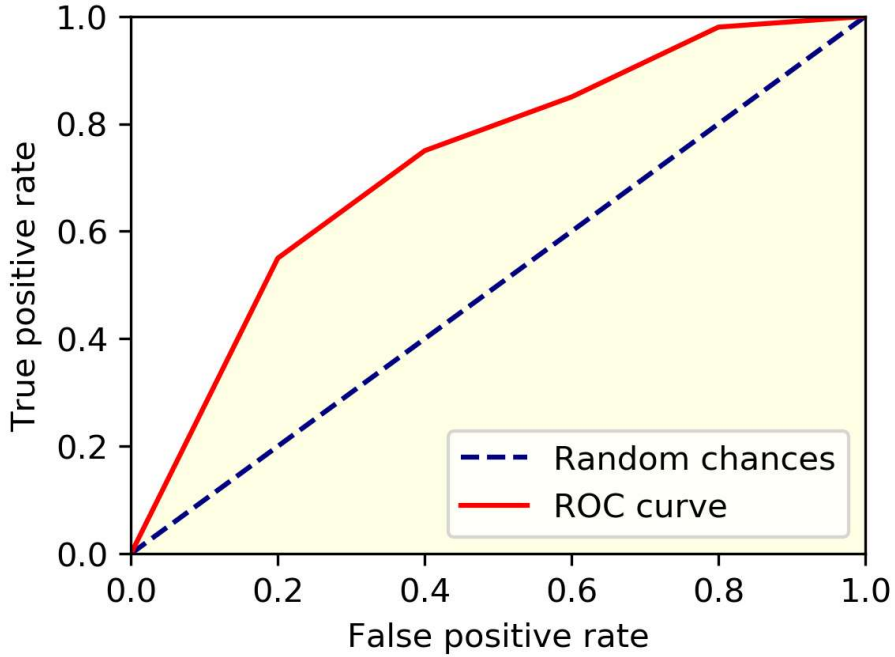
```
#
# Print the confusion matrix using Matplotlib
#
fig, ax = plt.subplots(figsize=(3, 3))
ax.matshow(cmat, cmap=plt.cm.Oranges, alpha=0.3)
for i in range(cmat.shape[0]):
    for j in range(cmat.shape[1]):
        ax.text(x=j, y=i,s=cmat[i, j], va='center', ha='center', size='xx-large')


plt.xlabel('Predictions', fontsize=12)
plt.ylabel('Actuals', fontsize=2)
plt.title('Confusion Matrix of ANN', fontsize=14)
plt.show()
```



ROC Curve and AUC

- An ***ROC curve*** (***R*eceiver *O*perating *C*haracteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:
  - True Positive Rate (Recall) = TP/(TP+FN)
  - False Positive Rate = FP/(FP+TN)

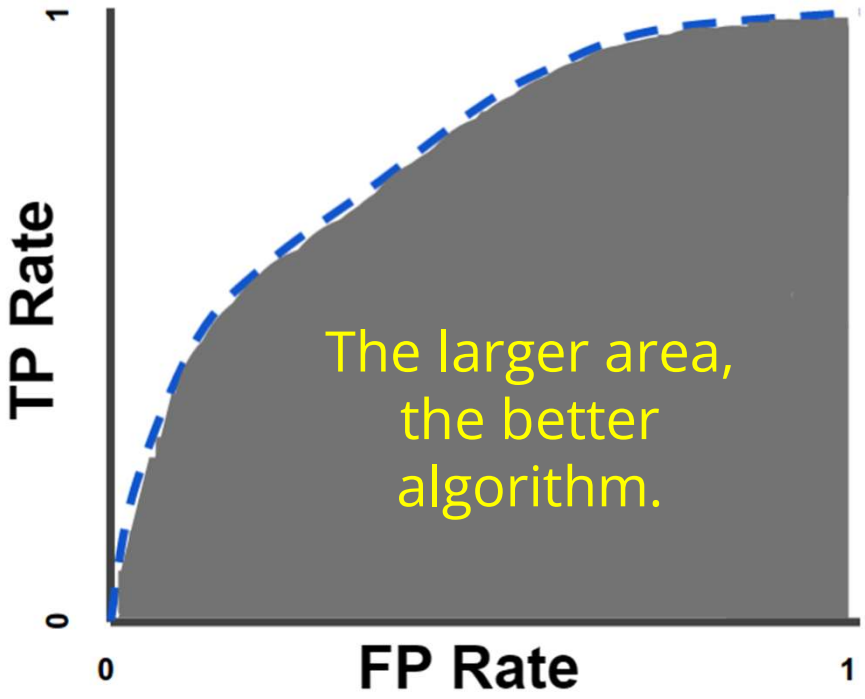


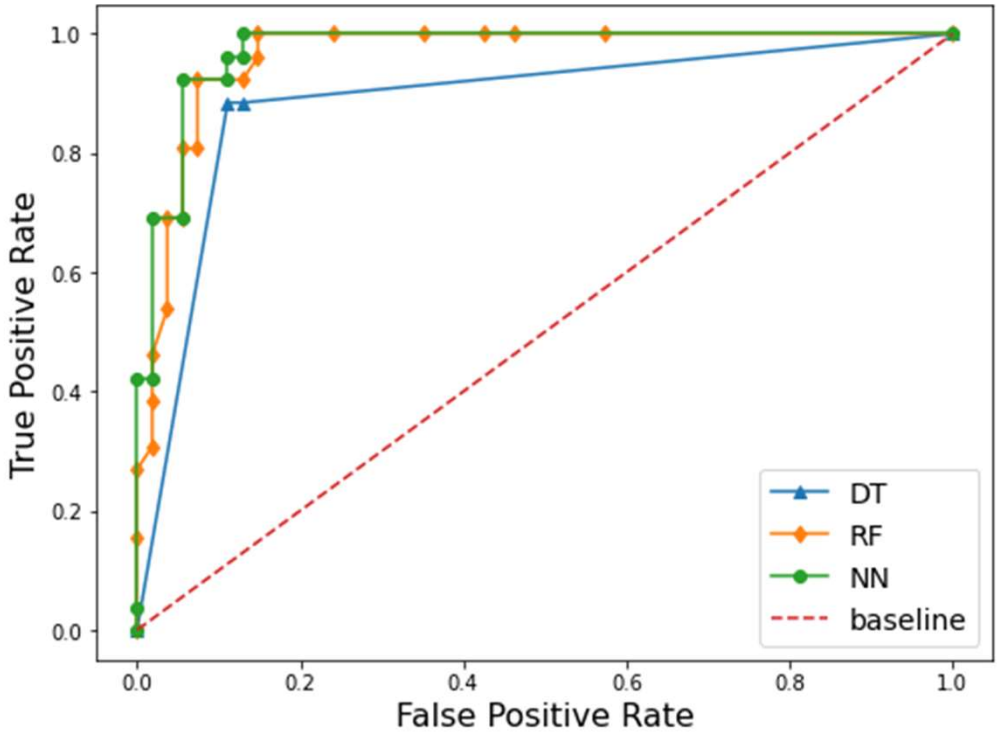


Ref: https://towardsdatascience.com/understanding-the-roc-curve-and-auc-dd4f9a192ecb

ROC Curve and AUC

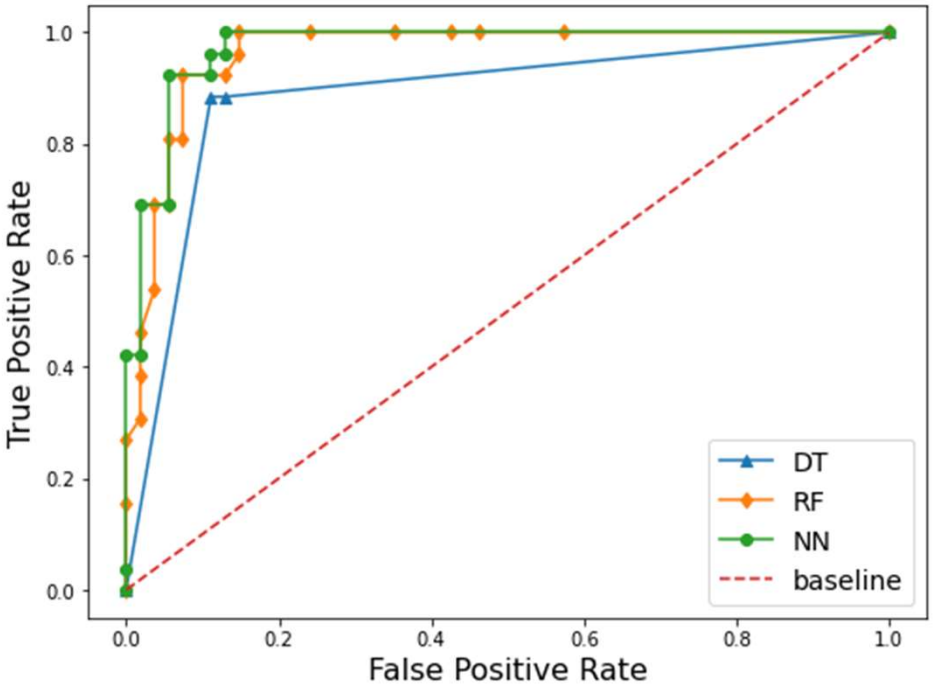
- ***AUC*** (Area under the ROC) Curve measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1).





Python's sklearn.metrics.*roc_curve()*

- Typically for *binary* class. For multiclass, see https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html



sklearn.metrics.*roc_auc_score()*

```
>>> import numpy as np
>>> from sklearn.metrics import roc_auc_score
>>> y_true = np.array([0, 0, 1, 1])
>>> y_scores = np.array([0.1, 0.4, 0.35, 0.8])
>>> roc_auc_score(y_true, y_scores)
0.75
```

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

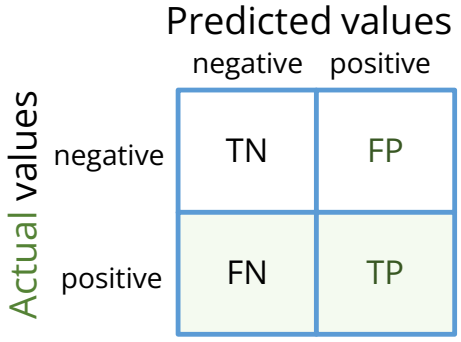
Summary of Metrics for Classification

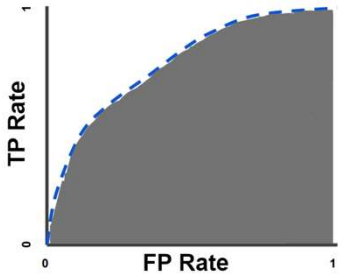
- Confusion Matrix
- Accuracy
- Recall
- Precision
- F1-Score
- ROC curve and AUC

- ส่วนใหญ่นิยมใช้ accuracy เพราะเข้าใจง่าย
- แต่หาก imbalance ใช้ AUC หรือ F1-score จะดีกว่า
- สรุป... ใช้ทั้ง accuracy และ AUC

แต่.....

- สนใจ recall หากต้องหากตรวจหาให้ครบ
- สนใจ precision หากไม่ต้องการตรวจจับเกิน
- ใช้ F1-score หากต้องการทั้ง recall + precision





Model complexity

How complex is a good model?

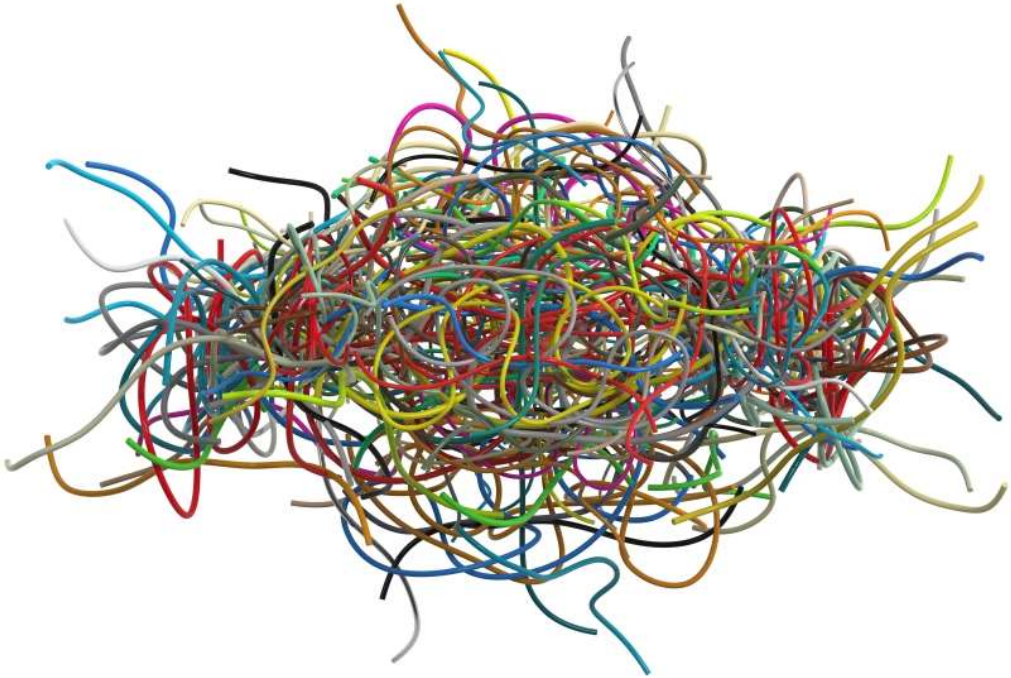
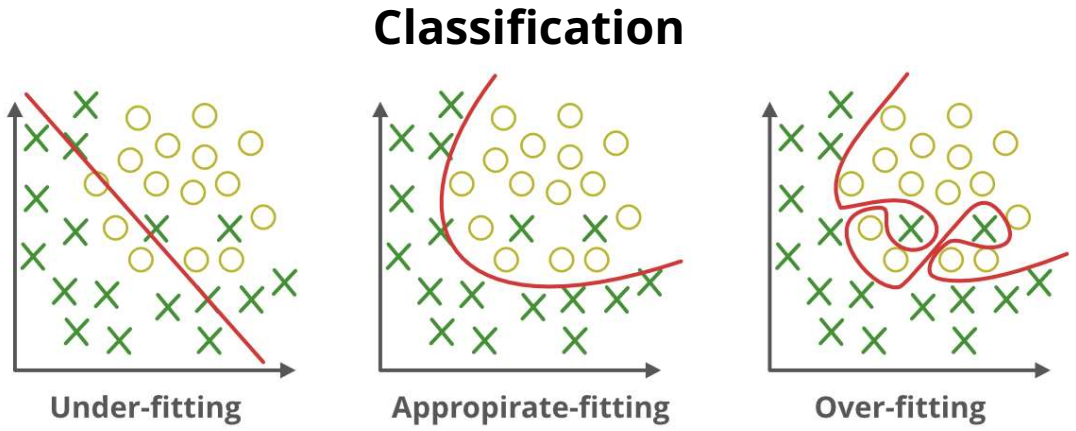


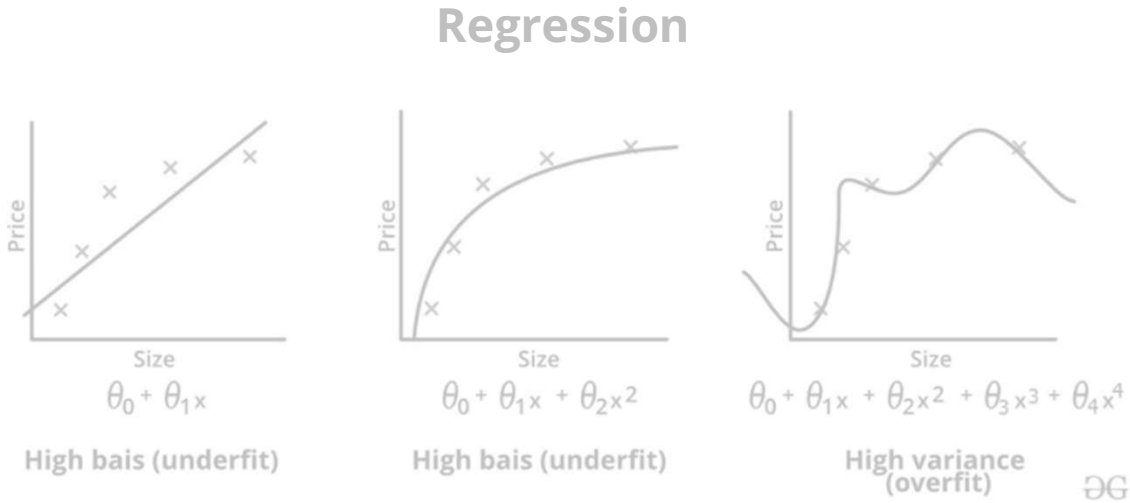
Image source: https://kingsley-books.com/insanity-by-way-of-complexity/

Underfitting and overfitting Bias and Variance

- *Underfitting* is the situation that the model fails to learn the pattern in the data and performs poorly on both training (and testing data).
  - *Too simple* models.
  - The model is *high bias.*

- *Overfitting* is the situation that model memorizes the training data, performs good on training data but fails to generalize to testing data.
  - *Too complex* models.
  - The model is *high variance*.

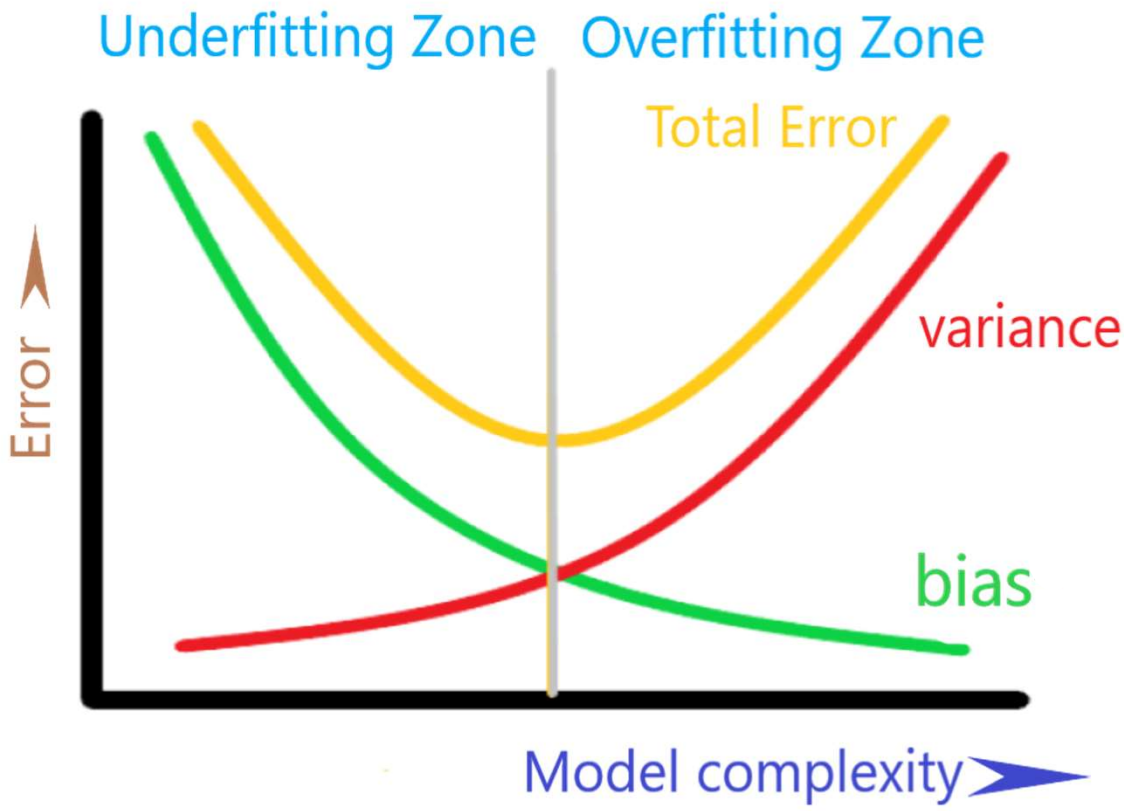




Source: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

Bias-Variance Trade-off: *Overfitting and Underfitting*

- If the bias is too high, model doesn't learn and fails to perform,
- whereas if the variance is too high, model fails to generalize to test dataset.
- *If complexity increases, bias decreases but variance increases and vice versa.*
- Optimal bias and variance are required for the best model performance and least error.

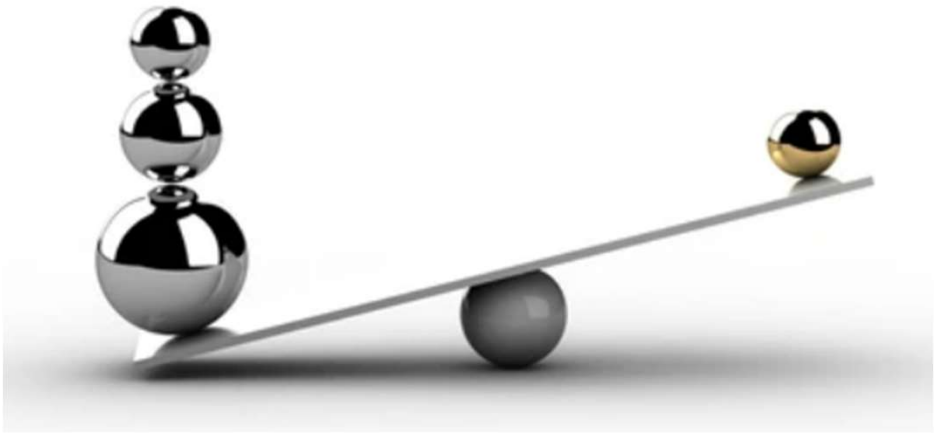


Outline

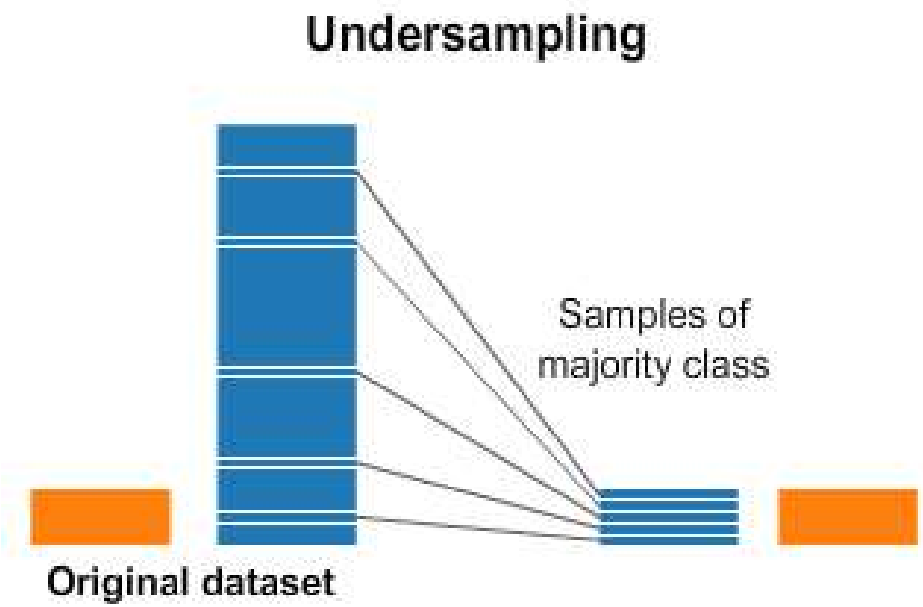
- Foundation of classification
- Logistics regression
- Decision Tree, Random Forest
- Artificial Neural Networks
- Generic classification modeling process
- Model validation methods
- มาตรวัดประสิทธิภาพของ classification algorithms
- Imbalanced data
- Feature selection
- Hyperparameter tuning

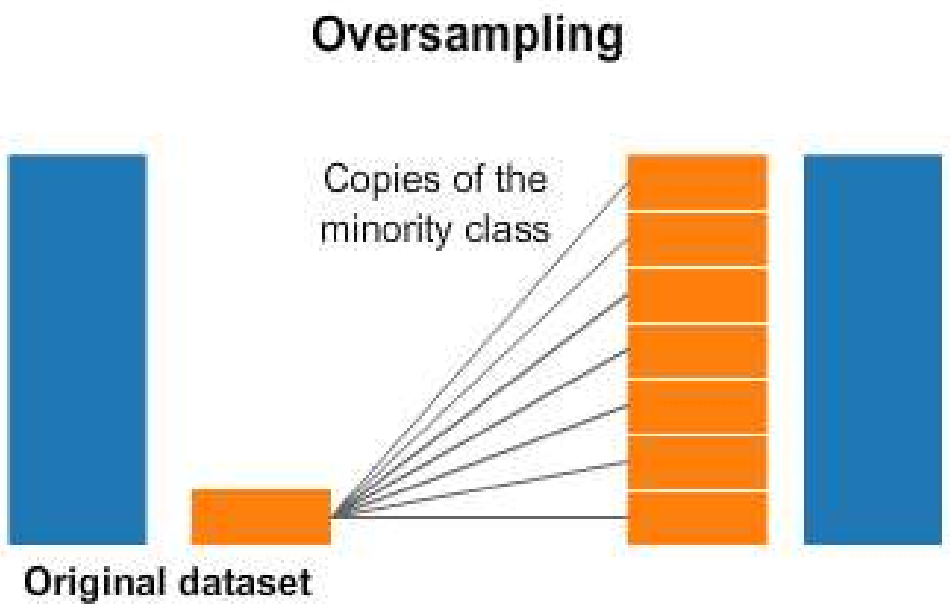
Imbalanced data

- *Imbalanced data* หมายถึงข้อมูลที่มีคลาสใดคลาสหนึ่งมีจำนวนหรือสัดส่วนมากกว่า (หรือน้อยกว่า) คลาสอื่น ๆ อย่างมาก ๆ เช่น 99:1 → 90:10
  - การโกงบัตรเครดิต (credit card fraudulent)
  - การตรวจหาเชื้อโรค (disease detection)
  - การหาสแปมเมล (spam email filtering)
  - Forecasting of ozone levels
  - จำนวนผู้ใช้ Internet ที่ click บน Ads เทียบกับไม่คลิก



การจัดการ Imbalanced data





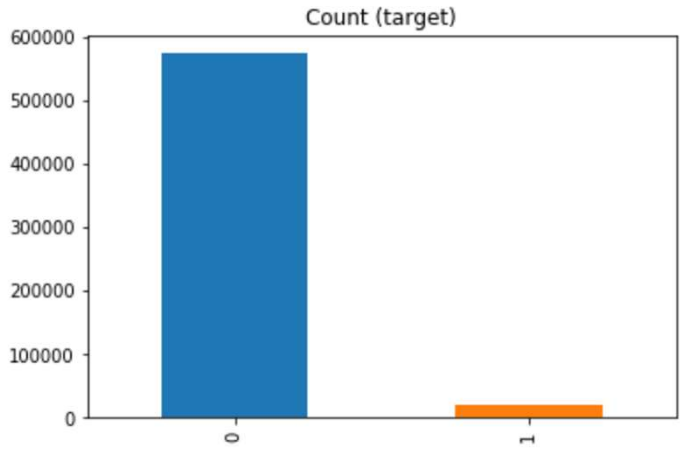
https://medium.com/analytics-vidhya/undersampling-and-oversampling-an-old-and-a-new-approach-4f984a0e8392

https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets

Python library for handling imbalanced data

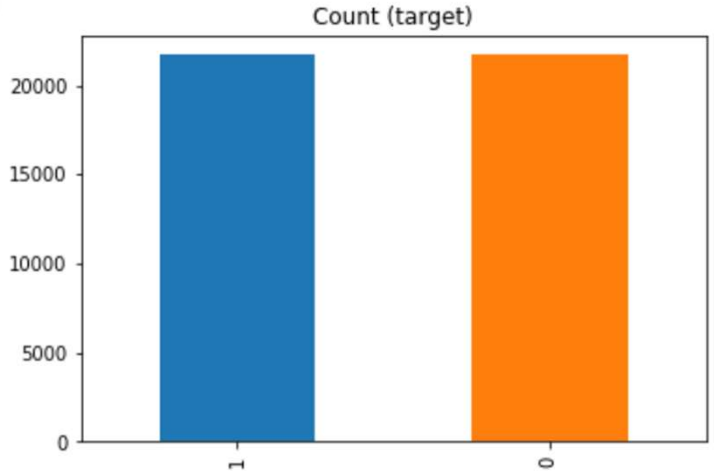
- https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets

- *Random Sampling*, an easy way.



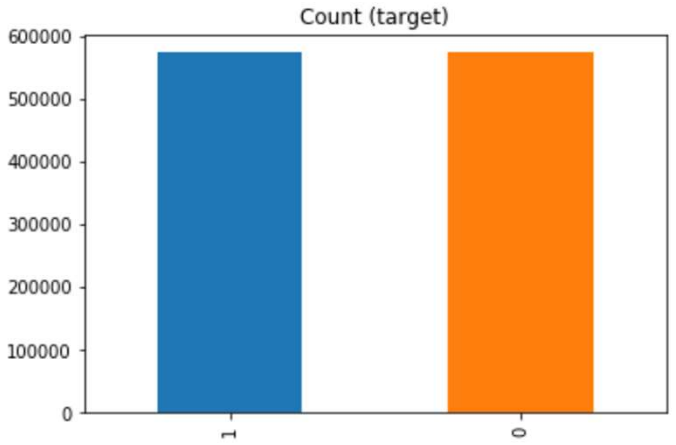
Random under-sampling

```
df_class_0_under = df_class_0.sample(count_class_1)
df_test_under = pd.concat([df_class_0_under, df_class_1], axis=0)
```



Random over-sampling

```
df_class_1_over = df_class_1.sample(count_class_0, replace=True)
df_test_over = pd.concat([df_class_0, df_class_1_over], axis=0)
```

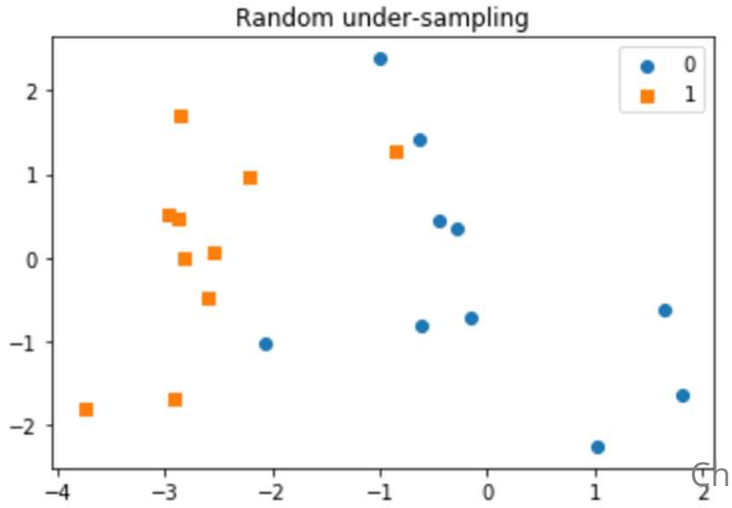


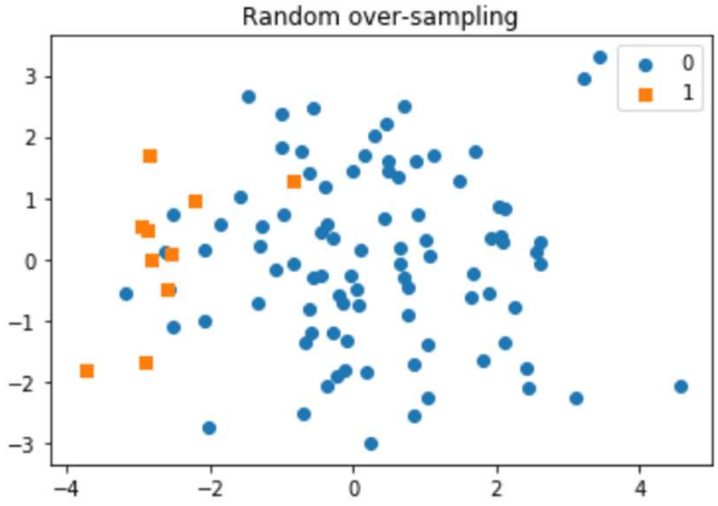
Python library imbalanced-learn

```
import imblearn
```

```
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(return_indices=True)
X_rus, y_rus, id_rus = rus.fit_sample(X, y)

print('Removed indexes:', id_rus)

plot_2d_space(X_rus, y_rus, 'Random under-sampling')
```

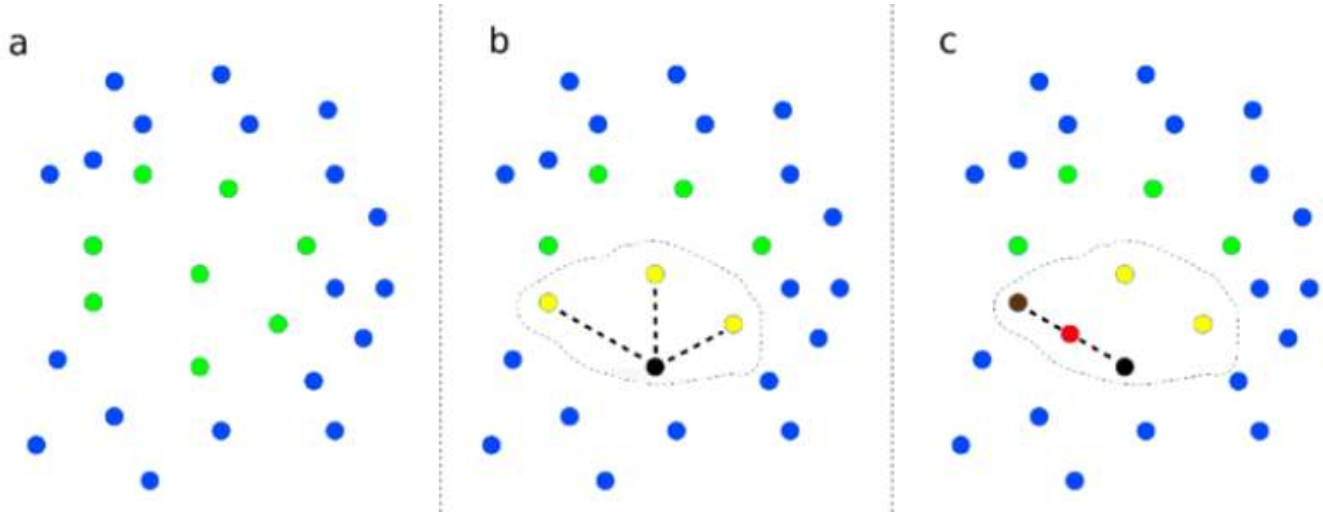
```
from imblearn.over_sampling import RandomOverSampler

ros = RandomOverSampler()
X_ros, y_ros = ros.fit_sample(X, y)

print(X_ros.shape[0] - X.shape[0], 'new random picked points')

plot_2d_space(X_ros, y_ros, 'Random over-sampling')
```





Imbalanced handling – *Over-Sampling*

- *SMOTE (Synthetic Minority Over-sampling Technique)*
- Chawla et. al., *Journal of Artificial Intelligence Research* (2002).
- Very popular since then.
- Create synthetic samples from the minor class instead of creating copies.



a: SMOTE starts from a set of positive (green points) and negative (blue points) examples; b: It then selects a positive example (black) and its *K-nearest neighbors* among the positives (yellow points, with k = 3), c: Finally, one of the k-nearest neighbors is randomly selected (brown point) and a new synthetic positive example is added, by randomly generating an example (red point) along the straight line that connects the black and brown points. The procedure depicted in b and c is repeated for all the positives, by adding each time a new synthetic example similar (in an Euclidean sense) to the other positive examples.

SMOTE Example – *SMOTE-Imbalance.ipynb*

SMOTE with Imbalance Data using imblearn Module

Date: 9 Mar 2022

Modified from https://www.kaggle.com/qianchao/smote-with-imbalance-data

[1]:
```
import pandas  as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

from imblearn.over_sampling import SMOTE
```

[2]:
```
data = pd.read_csv('creditcard.csv')
data.head(3)
```

[2]:

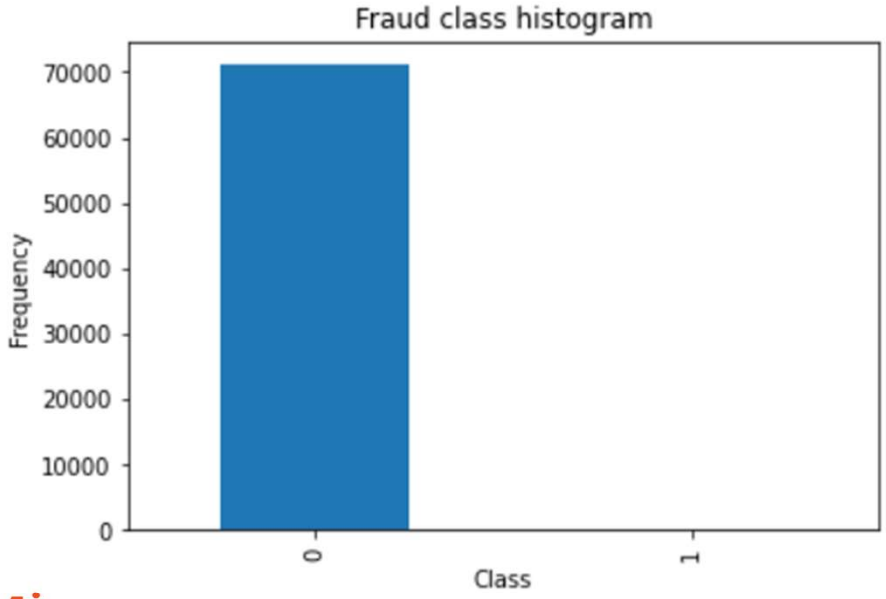
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V2
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.0669
1	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.1412
2	7	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	...	-0.073425	-0.268092	-0.204233	1.0115

3 rows × 31 columns

Credit card data are imbalanced

```
[3]: pd.value_counts(data['Class']).plot.bar()
     plt.title('Fraud class histogram')
     plt.xlabel('Class')
     plt.ylabel('Frequency')
     data['Class'].value_counts()
```

```
[3]: 0    71075
     1      127
     Name: Class, dtype: int64
```




```
X = data.iloc[:, data.columns != 'Class']
y = data.iloc[:, data.columns == 'Class']
print('Shape of X:', X.shape)
print('Shape of y:', y.shape)
```

```
Shape of X: (71202, 29)
Shape of y: (71202, 1)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
            test_size=0.3, random_state=0)
print("X_train: ", X_train.shape)
print("y_train: ", y_train.shape)
print("X_test : ", X_test.shape)
print("y_test : ", y_test.shape)
```

```
X_train:  (49841, 29)
y_train:  (49841, 1)
X_test :  (21361, 29)
y_test :  (21361, 1)
```

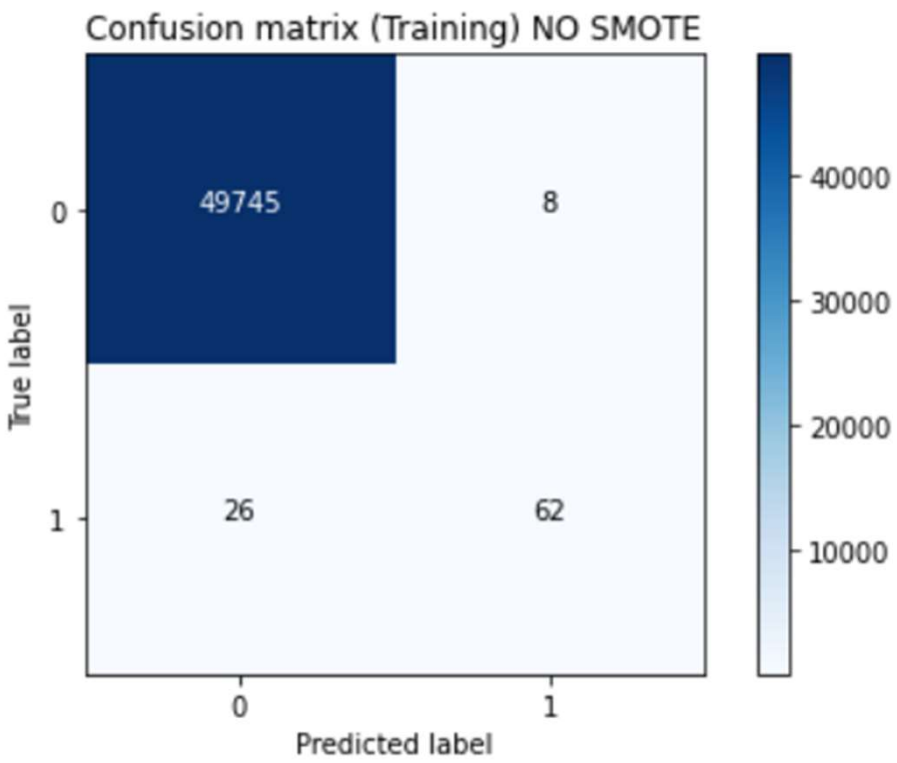
Output of running WITHOUT SMOTE

```
print("Let's first see running without SMOTE\n")


print('Shape of train_X:', X_train.shape)
print('Shape of train_y:', y_train.shape)
print(y_train.value_counts())
```

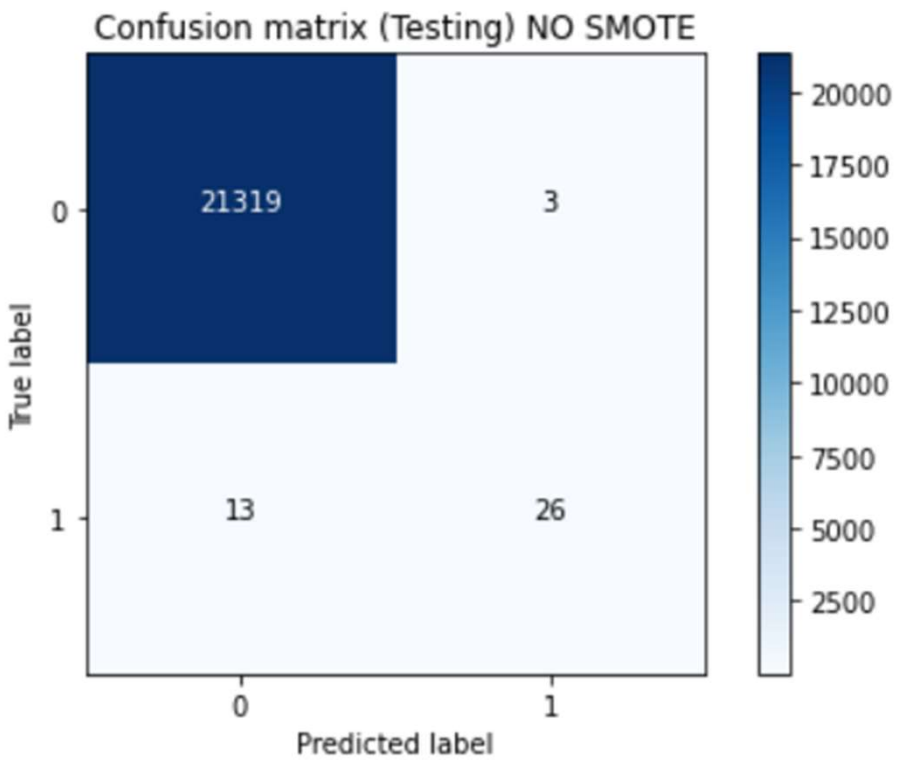
```
Let's first see running without SMOTE

Shape of train_X: (49841, 29)
Shape of train_y: (49841, 1)
Class
0        49753
1           88
dtype: int64
```

Output of running WITHOUT SMOTE

Recall score: 0.7045



Recall score: 0.6667



Running with SMOTE

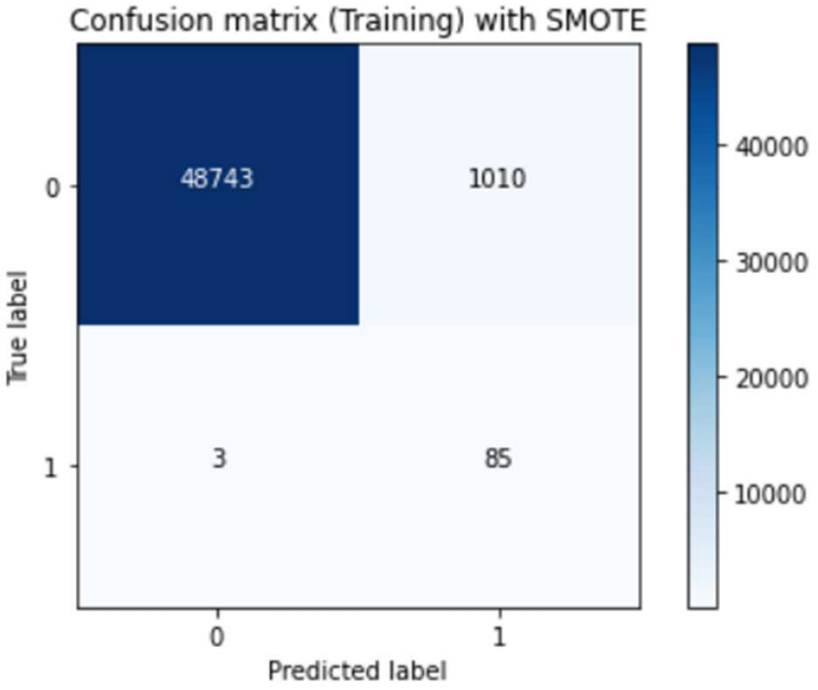
```
from imblearn.over_sampling import SMOTE

print("Now, let's run with SMOTE\n")
print('Before OverSampling: y_train', y_train.value_counts(), '\n')

sm = SMOTE(random_state=2)
X_train_res, y_train_res = sm.fit_sample(X_train, y_train.values.ravel())

print('After OverSampling:')
print('Shape of train_y:', y_train_res.shape)
print(type(y_train_res), np.bincount(y_train_res)) # like df.value_counts()
```

```
Now, let's run with SMOTE

Before OverSampling: y_train Class
0        49753
1           88
dtype: int64

After OverSampling:
Shape of train_y: (99506,)
<class 'numpy.ndarray'> [49753 49753]
```

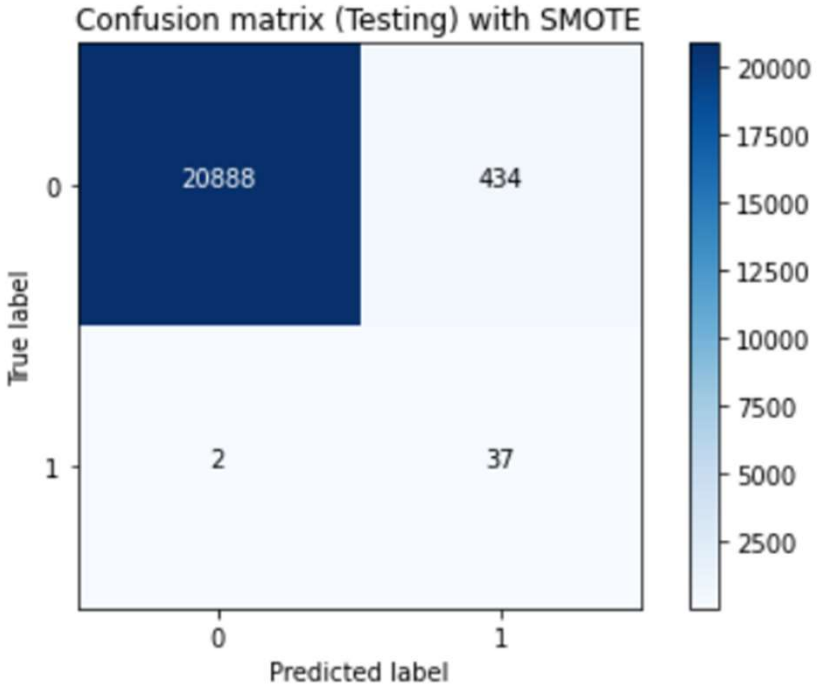
Ref: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

Output of running WITH SMOTE

Recall score: 0.9659

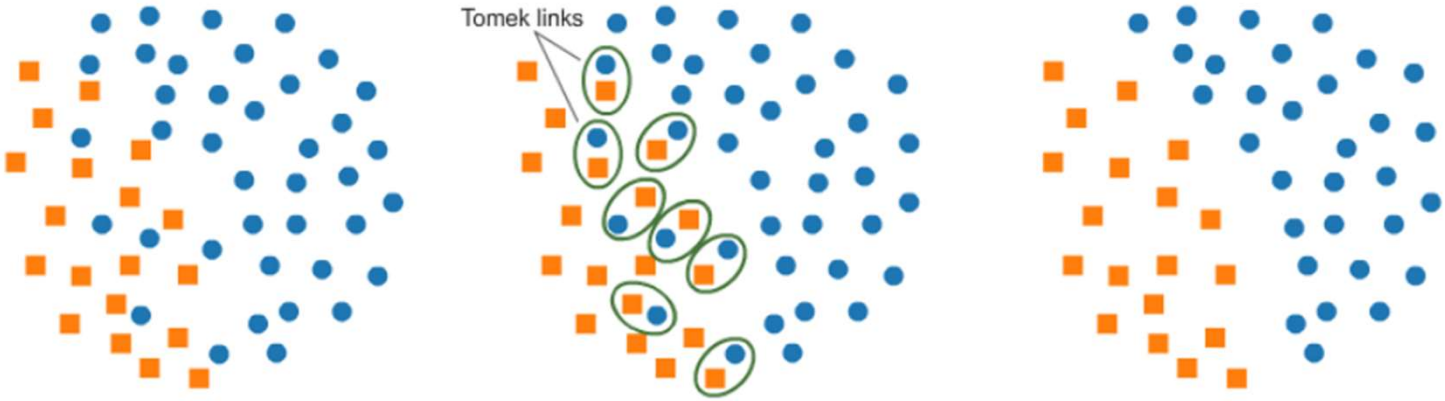


Recall score: 0.9487



Combined over-sampling with under-sampling

- *SMOTE-Tomek (SMOTE-TL)*
  - SMOTE then cleaning using *Tomek links*
  - The objects that form Tomek links are those from different classes having a distance between them smaller than the distance from them to any other object in the dataset.

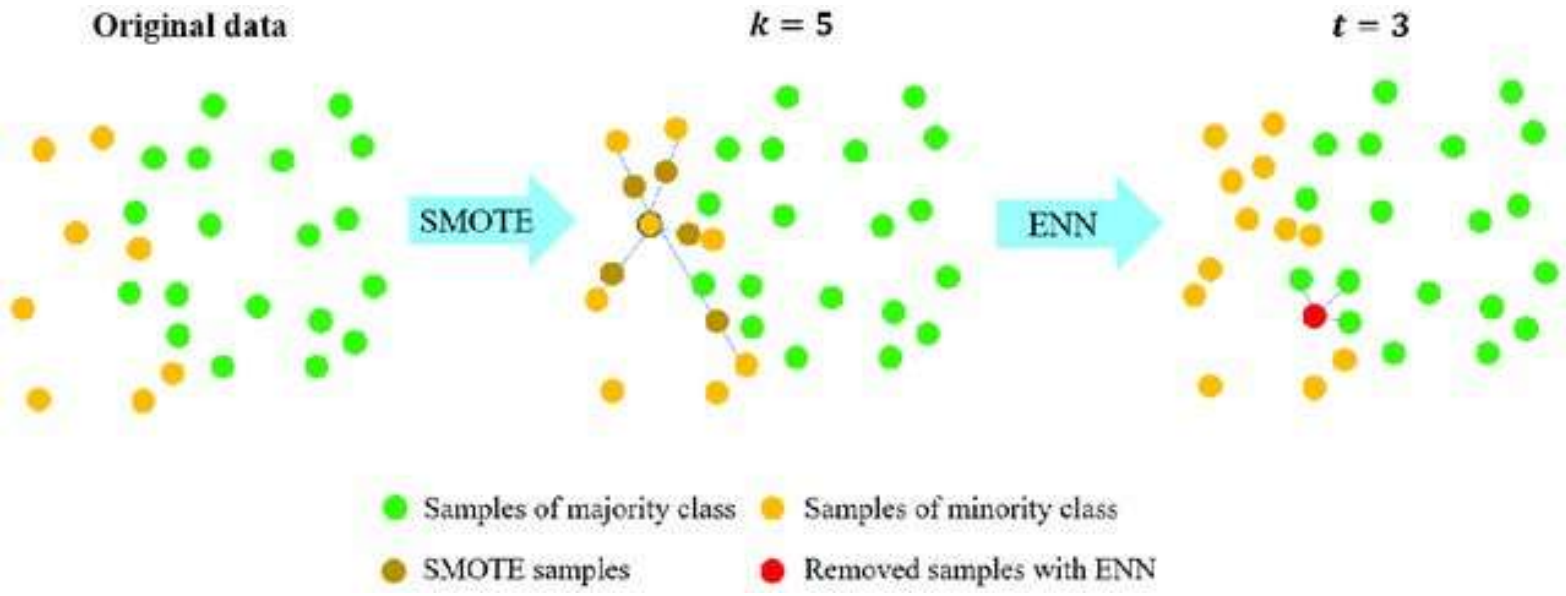



```
from imblearn.combine import SMOTETomek

smt = SMOTETomek(random_state=101)
```

G. Batista, *et al.* "A study of the behavior of several methods for balancing machine learning training data," ACM SIGKDD Explor News. 2004.

Combined over-sampling with under-sampling

- *SMOTE-ENN*
  - SMOTE with Wilson's Edited Nearest Neighbor (ENN) method.
  - Randomly selects instances and remove noise instances that there are no *k* instances in nearest neighbors.



G. Batista, *et al.* "A study of the behavior of several methods for balancing machine learning training data," ACM SIGKDD Explor News. 2004.

Outline

- Foundation of classification
- Logistics regression
- Decision Tree, Random Forest
- Artificial Neural Networks
- Generic classification modeling process
- Model validation methods
- มาตรวัดประสิทธิภาพของ classification algorithms
- Imbalanced data
- Feature selection
- Hyperparameter tuning

Feature Selection

- What is *Feature Selection*?
  - Finding a subset of *significant* and *relevant* features from all available features.

- *Benefits*
  - Training faster
  - Reducing model complexity and making it easier to interpret
  - Building a sensible model with better prediction power
  - Reducing overfitting by selecting the right set of features

Sample needs of feature selection

Columns

- # battery_power Total energy a battery can store in one time measured in mAh
- # blue Has bluetooth or not
- # clock_speed speed at which microprocessor executes instructions
- # dual_sim Has dual sim support or not
- # fc Front Camera mega pixels
- # four_g Has 4G or not
- # int_memory Internal Memory in Gigabytes
- # m_dep Mobile Depth in cm
- # mobile_wt Weight of mobile phone
- # n_cores Number of cores of processor
- # pc Primary Camera mega pixels
- # px_height Pixel Resolution Height
- # px_width Pixel Resolution Width
- # ram Random Access Memory in Mega Bytes
- # sc_h Screen Height of mobile in cm
- # sc_w Screen Width of mobile in cm
- # talk_time longest time that a single battery charge will last when you are
- # three_g Has 3G or not
- # touch_screen Has touch screen or not
- # wifi Has wifi or not
- # price_range This is the target variable with value of 0(low cost), 1(medium cost), 2(high cost) and 3(very high cost).

Mobile Phone Price

Feature selections using Python

- Filter approaches
  - Removing features with low variance
  - Univariate feature selection (with `SelectK` method)
  - *Correlation Matrix with Heatmap*
- Wrapper approaches
  - Backward elimination
  - Forward selection
  - Bi-directional elimination(Stepwise Selection)
  - *Recursive Feature Elimination (RFE)*
- Embedded approaches
  - Feature selection using `SelectFromModel`

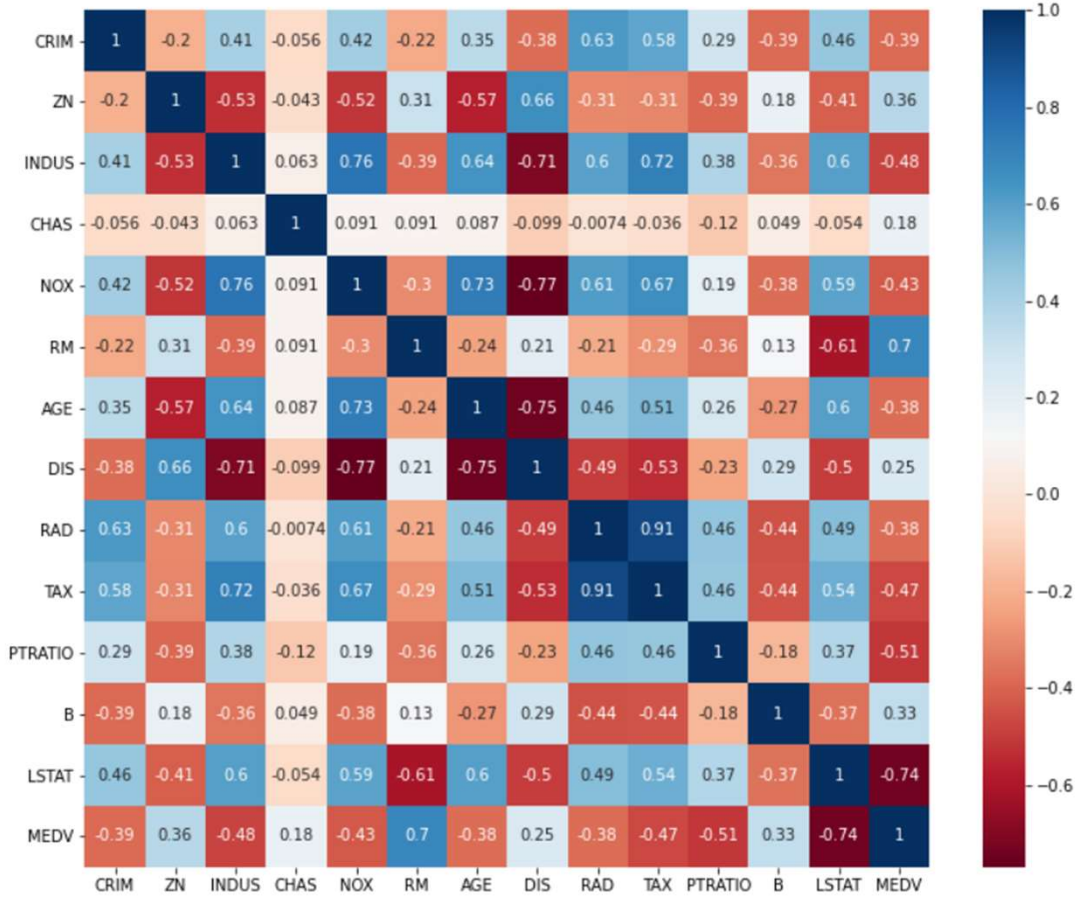
- https://towardsdatascience.com/feature-selection-with-pandas-e3690ad8504b

Feature selection – Dataset

- *Boston Dataset* (Ref: http://lib.stat.cmu.edu/datasets/boston)
  - The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978.
- Variables

1. CRIM per capita crime rate by town
2. ZN proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS proportion of non-retail business acres per town
4. CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX nitric oxides concentration (parts per 10 million)
6. RM average number of rooms per dwelling
7. AGE proportion of owner-occupied units built prior to 1940
8. DIS weighted distances to five Boston employment centers
9. RAD index of accessibility to radial highways
10. TAX full-value property-tax rate per USD10,000
11. PTRATIO pupil-teacher ratio by town
12. B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT % lower status of the population
14. MEDV Median value of owner-occupied homes in $1000's ← Target variable

Feature selection – Filter approach – Correlation Matrix with Heatmap



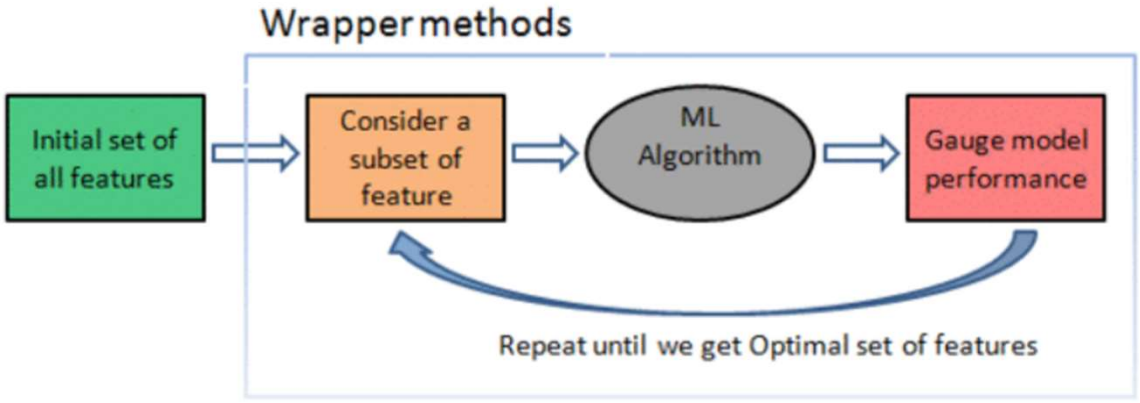

```
# Correlation with output variable
cor_target = abs(cor["MEDV"])
#Selecting highly correlated features with target
features = cor_target[cor_target>0.5]
features
```

```
RM          0.695360
PTRATIO     0.507787
LSTAT       0.737663
MEDV        1.000000
Name: MEDV, dtype: float64
```

Source: https://towardsdatascience.com/feature-selection-with-pandas-e3690ad8504b

Feature selection - Wrapper methods

- Take a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion
  - Regression: p-values or adjusted R-squared
  - Classification: accuracy, precision, recall, f1-score, etc.
  - Finally, it selects the combination of features that gives the optimal results for the specified machine learning algorithm.
- Most common methods are:
  - Forward selection
  - Backward elimination
  - Bi-directional elimination(Stepwise Selectior
  - Recursive Feature Elimination (RFE)



More: https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python/

Wrapper feature selection: Recursive Feature Elimination (RFE)

```
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE


model = LinearRegression()
#Initializing RFE model
rfe = RFE(model, n_features_to_select=7)
#Transforming data using RFE
X_rfe = rfe.fit_transform(X,y)
#Fitting the data to model
model.fit(X_rfe,y)
print(rfe.support_)
print(rfe.ranking_)
```

```
[False False False  True  True  True False  True  True False  True False True]
[2 4 3 1 1 1 7 1 1 5 1 6 1]
```

Summary of feature selection methods in Python

1. Filter method is less accurate. It is great while doing EDA, it can also be used for checking multi co-linearity in data.
2. Wrapper and Embedded methods give more accurate results but as they are computationally expensive, these method are suited when you not too many features (~20).

https://towardsdatascience.com/feature-selection-with-pandas-e3690ad8504b

Outline

- Foundation of classification
- Logistics regression
- Decision Tree, Random Forest
- Artificial Neural Networks
- Generic classification modeling process
- Model validation methods
- มาตรวัดประสิทธิภาพของ classification algorithms
- Imbalanced data
- Feature selection
- Hyperparameter tuning

Hyperparameter tuning

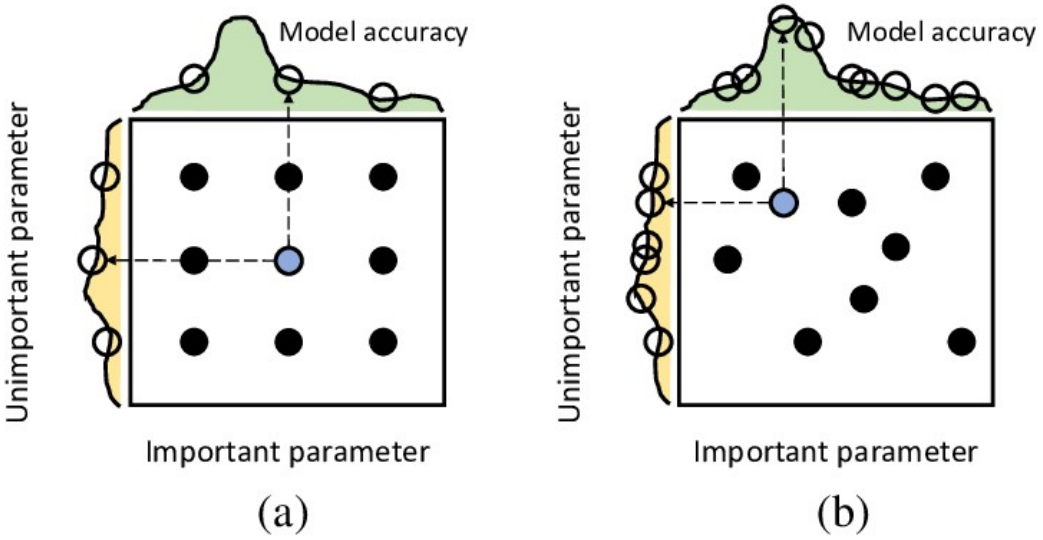
- Hyperparameter tuning (or hyperparameter optimization) is the process of determining the right combination of hyperparameters that maximizes the model performance.
- It works by running multiple trials in a single training process.
- Manual tuning is tedious: costly and time-consuming.
- Automated hyperparameter tuning:
  - uses already existing algorithms to automate the process.
  - First, specify a set of hyperparameters and their possible value ranges.
  - Then the algorithm runs those trials and fetches you the best set of hyperparameters that will give optimal results.

Hyperparameter tuning in Python

Model Parameter	Hyperparameter
Required for making prediction	Required for estimating the model parameters
The final parameters found after training will decide how the model will perform on unseen data.	The choice of hyperparameters decide how efficient the training is.
e.g. weights of ANN or coefficients in a linear regression	e.g. learning_rate in ANN or max_depth of Random Forest
Estimated by optimization algorithms (Gradient Descent, Adam)	Estimated with hyperparameters running

Approaches to hyperparameter tuning

1. Grid search
2. Random search
3. Bayesian Optimization
   - Work only to continuous hyperparameters, not categories.
   - Efficient for a few hyperparameters
4. Tree-structured Parzen estimators (TPE)
   - Do not model interactions between the hyper-parameters.



Sources:
1. Pilario *et al.* "A Kernel Design Approach to Improve Kernel Subspace Identification", IEEE Trans. on Industrial Electronics, 2020.
2. Bergstra J. *et al.*, Algorithms for Hyper-Parameter Optimization, NIPS'11, 2011.
3. https://medium.com/criteo-engineering/hyper-parameter-optimization-algorithms-2fe447525903

Commonly-used packages for hyperparameter tuning

- Scikit-learn
  - Random search, grid search
- Scikit-optimize
  - BayesSearchCV
- Optuna
- Hyperopt
  - Random Search
  - Tree of Parzen Estimators (TPE)
  - Adaptive TPE

Outline

- Foundation of classification
- Logistics regression
- Decision Tree, Random Forest
- Artificial Neural Networks
- Generic classification modeling process
- Model validation methods
- มาตรวัดประสิทธิภาพของ classification algorithms
- Imbalanced data
- Feature selection
- Hyperparameter tuning

References

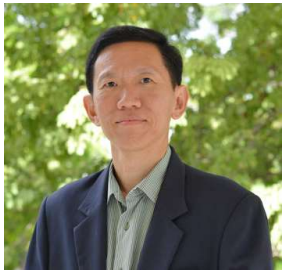
- Muneeb Asif, Predicting the success of bank telemarketing using various classification algorithms (2019).
- Derya Birant, Data Mining in Banking Sector Using Weighted Decision Jungle Method (2020).
- Decision Tree Classification in Python, https://www.datacamp.com/community/tutorials/decision-tree-classification-python
- One-Hot Encoding vs. Label Encoding, https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/

More information about imbalanced data

- https://github.com/pb111/Data-Preprocessing-Project-Imbalanced-Classes-Problem
- https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/
- G. Batista, et al. “A study of the behavior of several methods for balancing machine learning training data,” ACM SIGKDD Explor. News. 2004.
- https://towardsdatascience.com/imbalanced-classification-in-python-smote-tomek-links-method-6e48dfe69bbc
- https://towardsdatascience.com/imbalanced-classification-in-python-smote-enn-method-db5db06b8d50
- Ajinkya More, “Survey of resampling techniques for improving classification performance in unbalanced datasets,” 2016

About me

- Assoc. Prof. Chukiat Worasucheep
- Research Areas:
  - Computational Intelligence for financial and engineering applications
  - Data science for financial industry
  - Automatic trading systems
  - Financial prediction
- Teaching:
  - Data science
  - Computational Intelligence
  - Operating Systems
  - Exploring Computer Science



- Education:
  - Master of Science in Computer Science
    - Oregon State University, USA
  - Master of Business Administration (MBA)
    - Thammasat University
  - Bachelor of Engineering (Computer Engineering)
    - Chulalongkorn University