Machine Learning for APAT

2024. 11

Yongjin Jeong, KwangWoon University

Contents

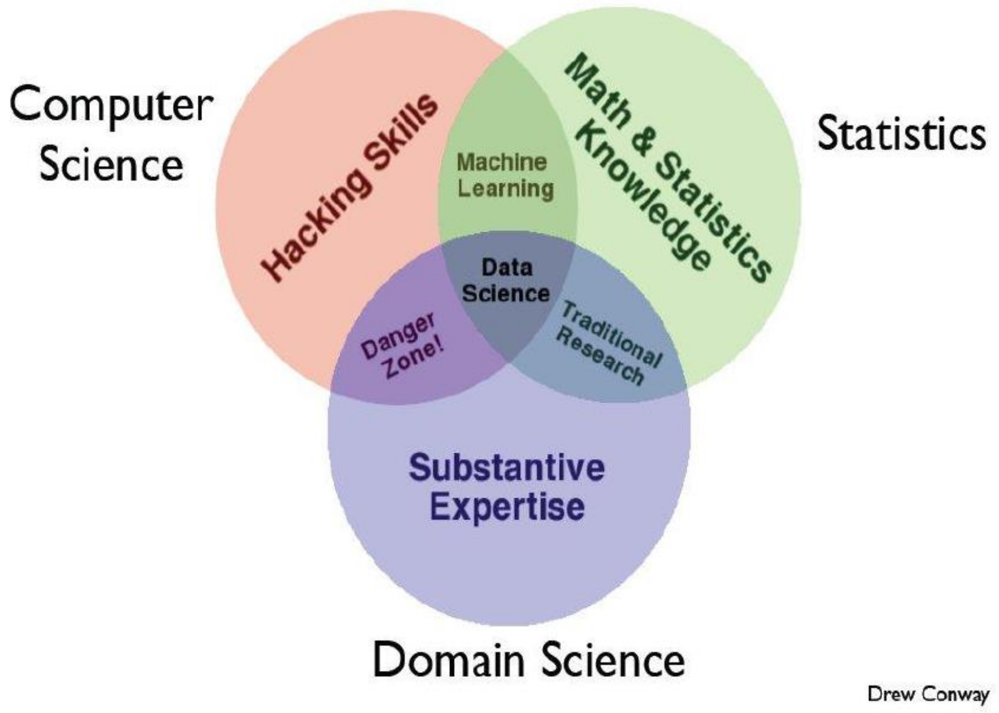
- Data science workflow
- Data preprocessing
- Machine learning models
- Linear Models
- Performance
- Overfitting
- Imbalance problem
- Sklearn library convention

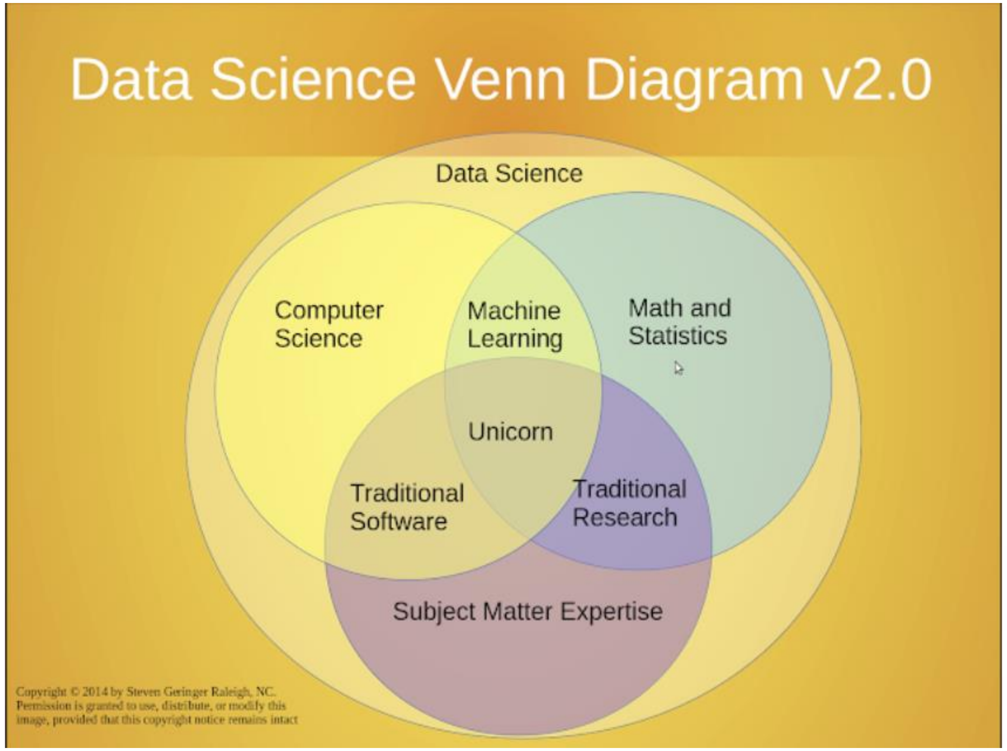
What is Data Science?

- **Definition (from Wikipedia)**
  - ✓ concept to unify statistics, data analysis, machine learning, domain knowledge and their related methods in order to understand and analyze actual phenomena with data
  - ✓ It uses techniques and theories drawn from many fields within the context of mathematics, statistics, computer science, domain knowledge, and information science
- **Components of Data Science**
  - ✓ Software Programming -> Data mining, Database
  - ✓ Statistics/mathematical modeling -> Machine Learning, Scientific Computing
  - ✓ Domain Knowledge -> Data driven business analytics
- **Main applications**
  - ✓ E-commerce, social media, IoT, biometrics, financial, management, health, pharmacy
  - ✓ Autonomous vehicles, smart energy, medical, ships, logistic, robots, etc.
  - ✓ Almost all areas

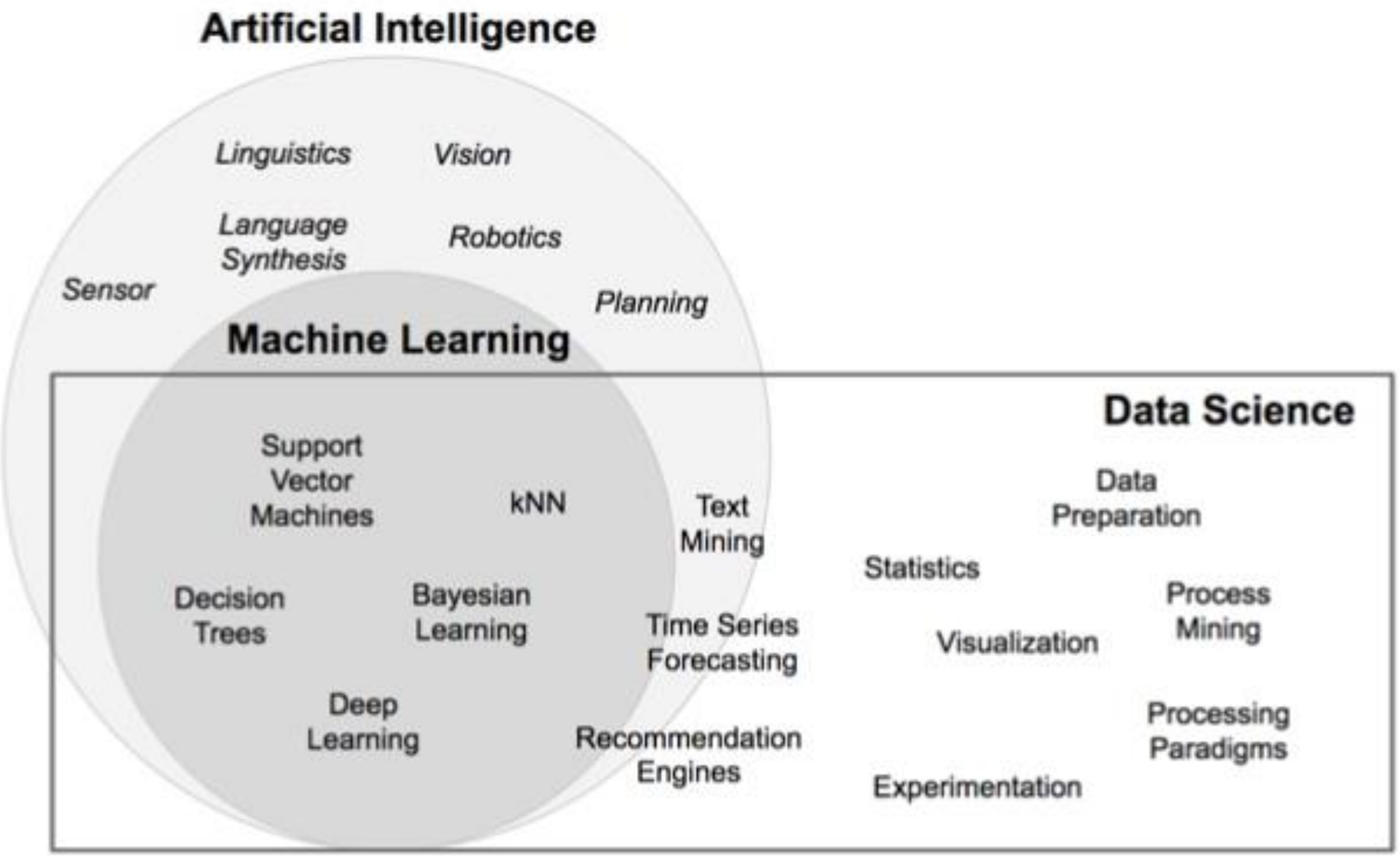
What is Data Science? – One definition

- **Venn Diagrams (Drew Conway 2010, Steven Geringer 2014)**





What is Data Science?



ref: Data Science: Concepts and Practice (by V.Kotu and B.Deshpande), 2022

Statistics vs. Data Science

	Statistics	Data science
Common	aim to extract knowledge from data - exploratory data analysis & Visualization - characterization & prediction - use sample data to make conclusions about either **future data (machine learning)** or **the population (statistics)**	
Main goal	- Inference and explanation - emphasis on understanding relationships and estimating population parameters	- prediction accuracy and complex datasets - focus on accurate predictions often without detailed understanding or interpretability
Language	estimating (inferencing) data point/observation independent variable dependent variable dummy variable	learning (training) & prediction example/instance/sample feature label or target one-hot encoding
Data	small or medium sized mostly structured more manual data collection (or surveys) In general, no web scraping or data processing	huge (big data) structured or unstructured more data collection/acquisition (from web and SNS)
Processing	query (past)	predict (future)
Tools	Mathematics prefer R SAS (statistics package)	programming (prefer Python) ML libraries (sklearn, tensorflow, etc.)

Data Science Work Flow

Problem Define → Data Collection, Acquisition → Analysis, Cleaning, Transform → Machine Learning (Train) → Performance Evaluation (Test) → Deployment

Problem Define

- Domain knowledge
- Business strategy

Data Collection, Acquisition

- CSV/Excel
- JSON
- HTML/XML
- SNS
- String(structured)
- Text(unstructured)
- Image, Voice
- Language
- Multi-modal

Analysis, Cleaning, Transform

- Visualization
- Missing values
- Invalid values
- Outliers
- Categorical encoding
- Scaling
- Transform
- Feature engineering

Machine Learning (Train)

- Supervised
- Unsupervised
- Loss (or Error)
- Bias and Variance
- Overfitting
- Regularization
- MLP/CNN/RNN
- Generative model
- Reinforcement learning
- Transformer

Performance Evaluation (Test)

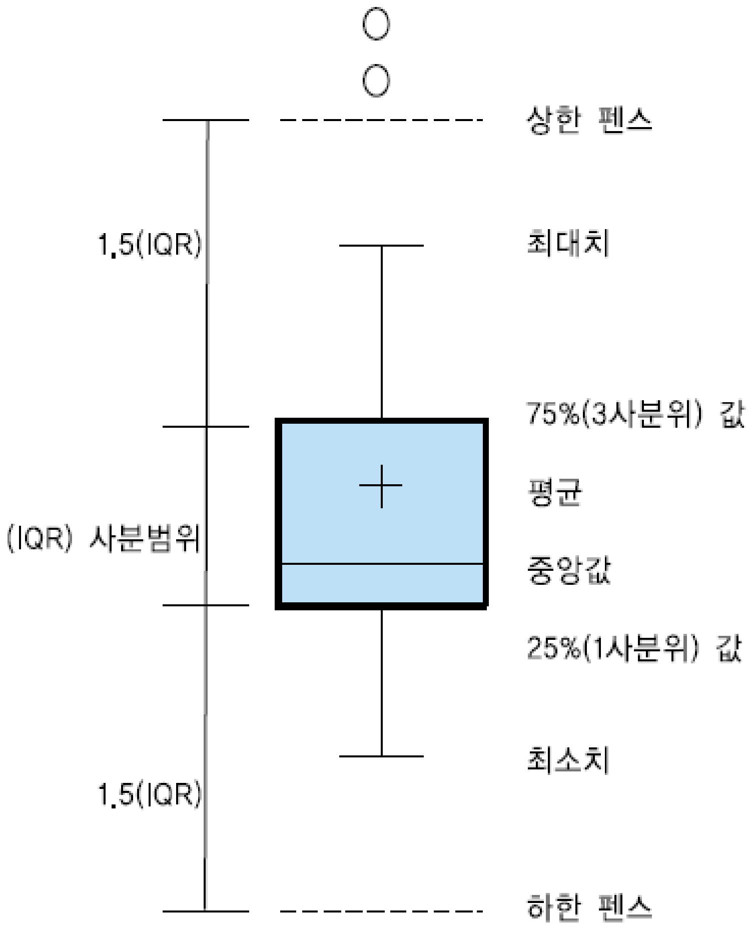
- R-square
- Accuracy
- Precision/recall
- F-1 score
- ROC/AUC
- mAP
- IoU

Deployment

- Server
- Mobile

Data Preprocessing - Scaling

- **Why scaling?**
  - What if two columns have different ranges for the values? Hard to compare.
- **Normalization (min-max scaling):**
  - $x = (x - x_{min}) / (x_{max} - x_{min})$
- **Standardization (standard scaling):**
  - $z = (x - \mu) / \sigma$
- **Robust scaling:**
  - $z = (x - \text{median}) / \text{IQR}$
  - use median and IQR (instead of μ and σ), robust to outliers
- **Which one to use?**
  - depends on your data



Categorical Encoding

- Label encoding
- Ordinal encoding
- One-hot encoding
- Binary encoding
- Target encoding
- many more…

State (Nominal Scale)
Maharashtra
Tamil Nadu
Delhi
Karnataka
Gujarat
Uttar Pradesh

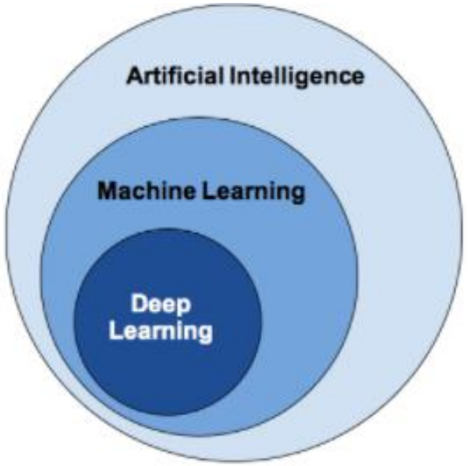
State (Label Encoding)
3
4
0
2
1
5

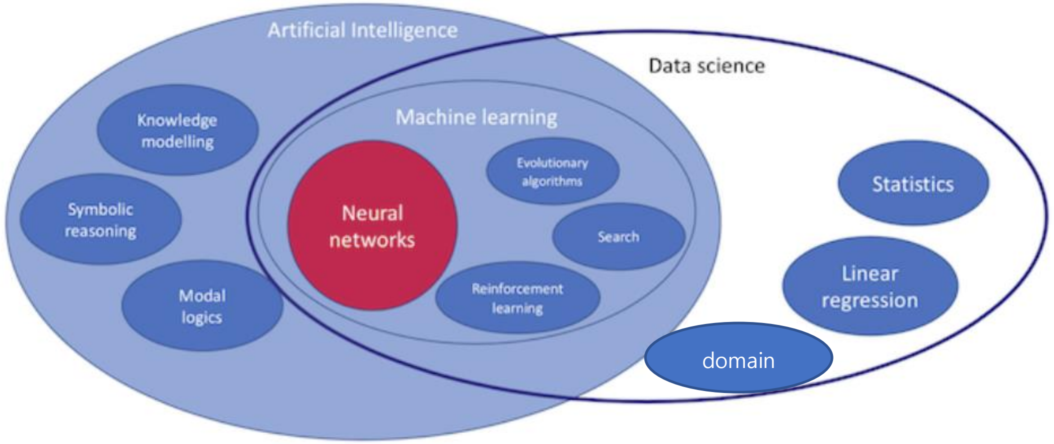
Original Encoding	Ordinal Encoding
Poor	1
Good	2
Very Good	3
Excellent	4

State	State_Maharashtra	State_Tamil Nadu	State_Delhi	State_Karnataka	State_Gujarat	State_Uttar Pradesh
Maharashtra	1	0	0	0	0	0
Tamil Nadu	0	1	0	0	0	0
Delhi	0	0	1	0	0	0
Karnataka	0	0	0	1	0	0
Gujarat	0	0	0	0	1	0
Uttar Pradesh	0	0	0	0	0	1

Machine Learning

- **What is ML**
  - the study of computer algorithms that improve automatically **through experience and by the use of data**. It is seen as a part of artificial intelligence. [wikipedia]
  - ML algorithms build a model based on sample data (training data) in order to make **predictions** or **decisions** without being explicitly programmed to do so.





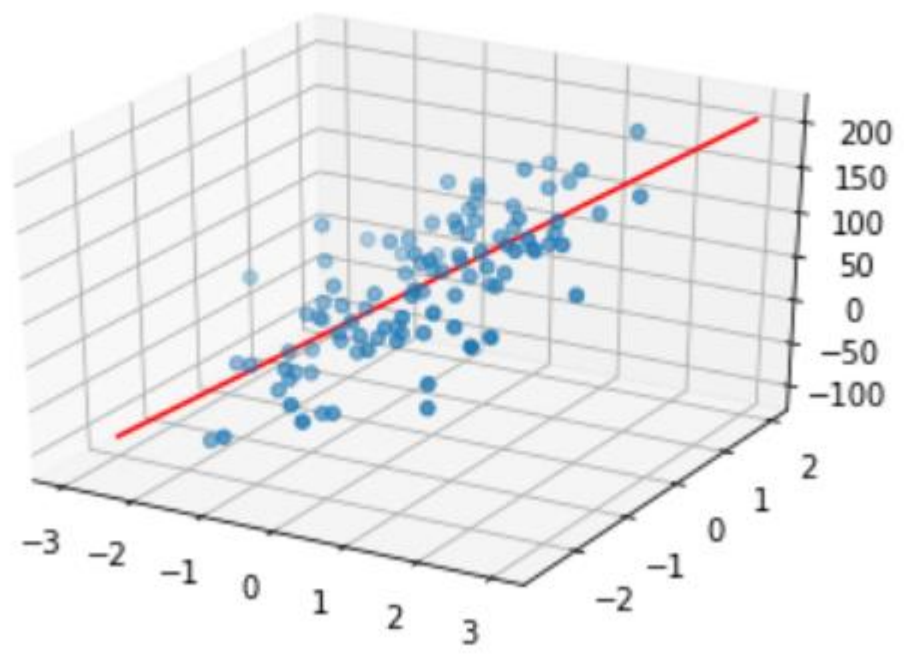
[ref] https://ictinstitute.nl/ai-machine-learning-and-neural-networks-explained/

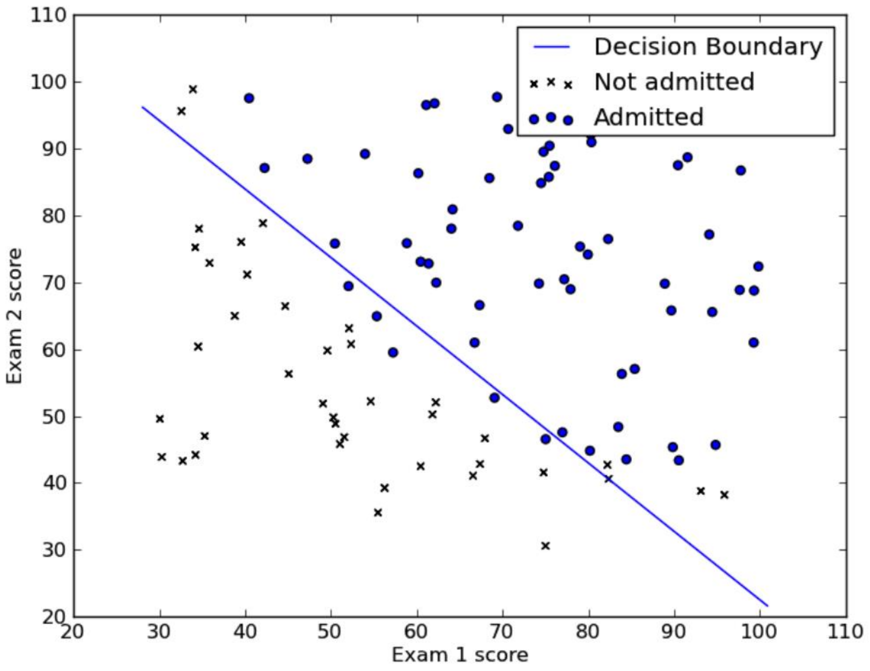
Machine Learning

- **Machine Learning approaches**
  - Supervised learning: inputs(features) and outputs(labels) are both given (by a teacher) to learn a general rule to map features to labels
    - **Regression** : labels are continuous values
    - **Classification**: labels are categorical values
  - Unsupervised learning: no labels are given, and to discover hidden patterns or features in data
    - **Dimension reduction**: PCA(Principal Component Analysis), tSNE, Autoencoder
    - **Clustering** (or grouping)
  - Semi-supervised learning: large unlabeled data with small labeled data
  - Reinforcement learning: interacts with the environment by producing actions and discovers errors or rewards (trial and error search, delayed reward)
    - Model-based RL (like control theory)
    - Model-free RL: Policy-iteration (Policy gradient, A3C) and value-iterations (Q-learning)

Supervised Learning

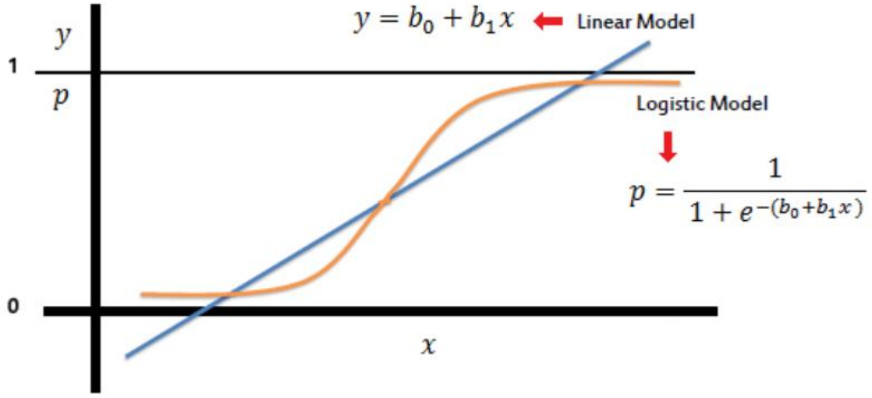
- **Linear Regression and Linear Classification**
  - (regression) $y = w1 \cdot x1 + w2 \cdot x2 + b$
  - (classification) $w1 \cdot x1 + w2 \cdot x2 + b = 0$

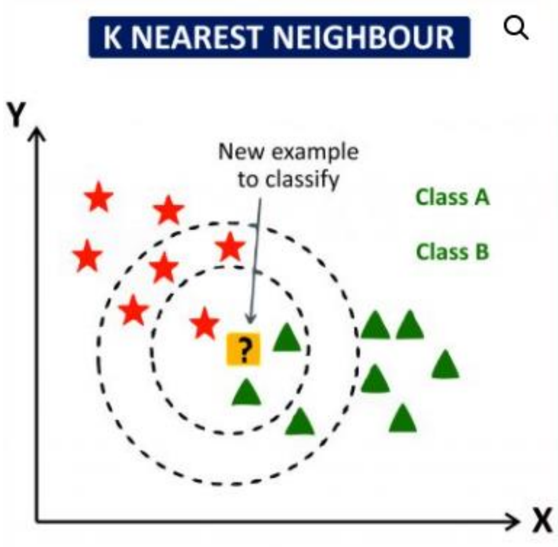


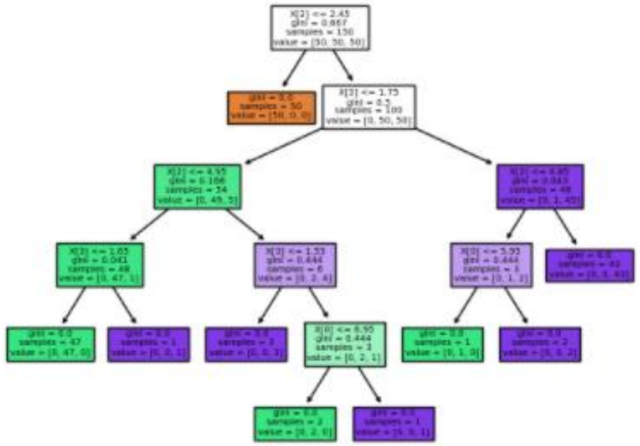


Supervised Learning

- **Other algorithms (linear and nonlinear)**
  - Logistic Regression Classifier
  - Knn (k-nearest neighbor)
  - Decision Tree
  - SVM (Support Vector Machine)

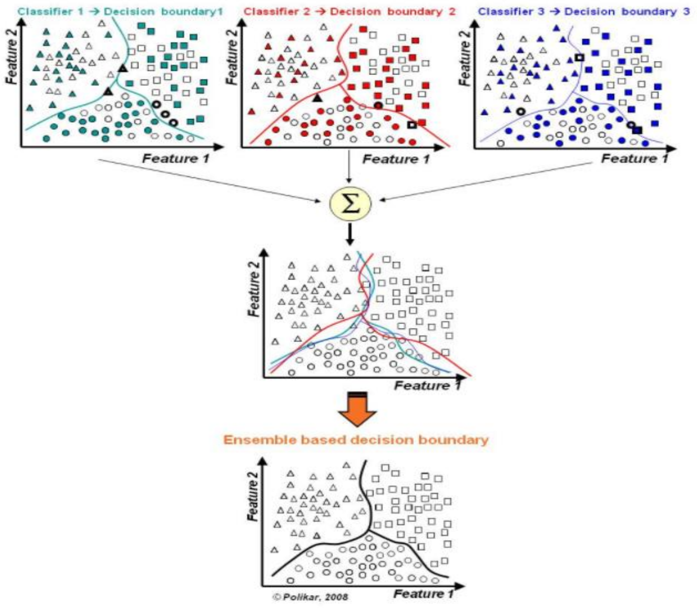




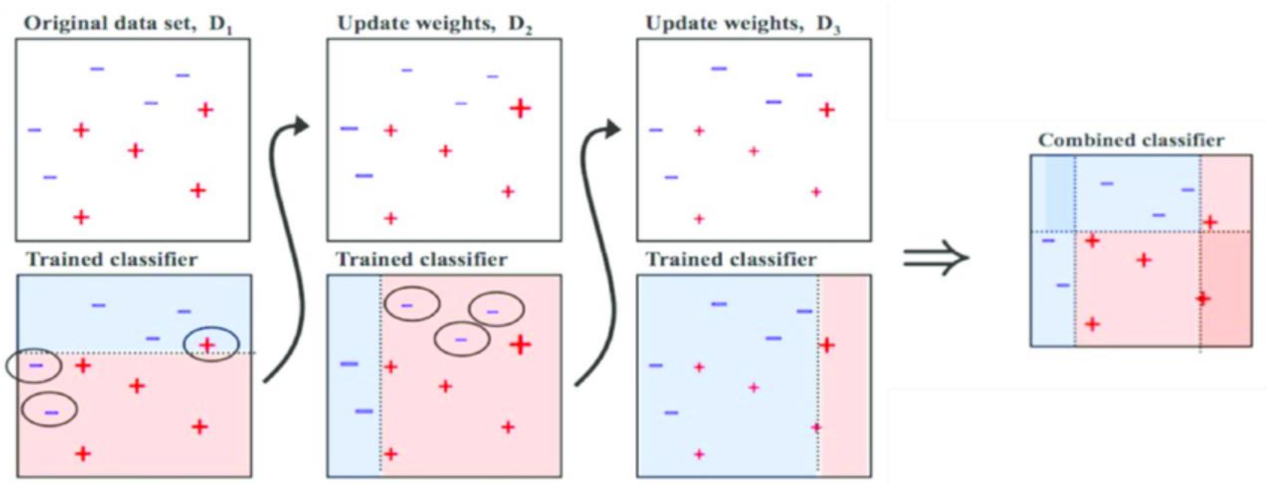


Supervised Learning

- **Ensemble method**
  - Combine many weak learners
  - Bagging: learns them independently in **parallel** and averages them (ex: Random Forest)
  - Boosting: learns them **sequentially** in an adaptive way and combines them (ex: Gradient Boost, Adaboost)



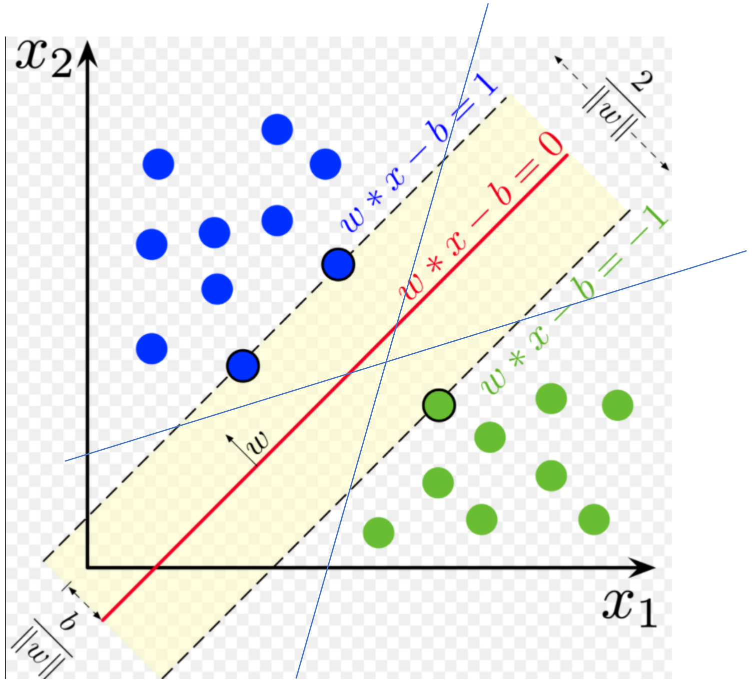
출처: https://swalloow.github.io/bagging-boosting

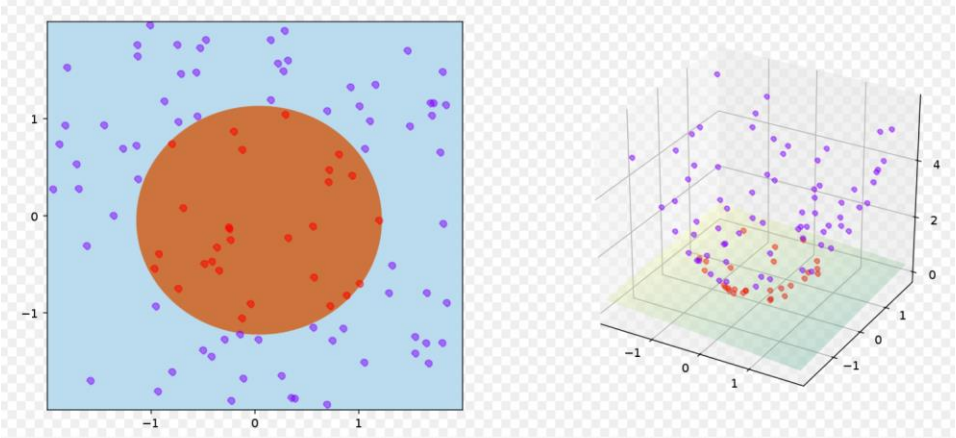


출처: Medium (Boosting and Bagging explained with examples)

Supervised Learning

- **SVM (Support Vector Machine)**
  - Finds a maximum-margin hyperplane
  - Linear SVM
  - Nonlinear SVM: use kernel (polynomial, sigmoid, rbf)

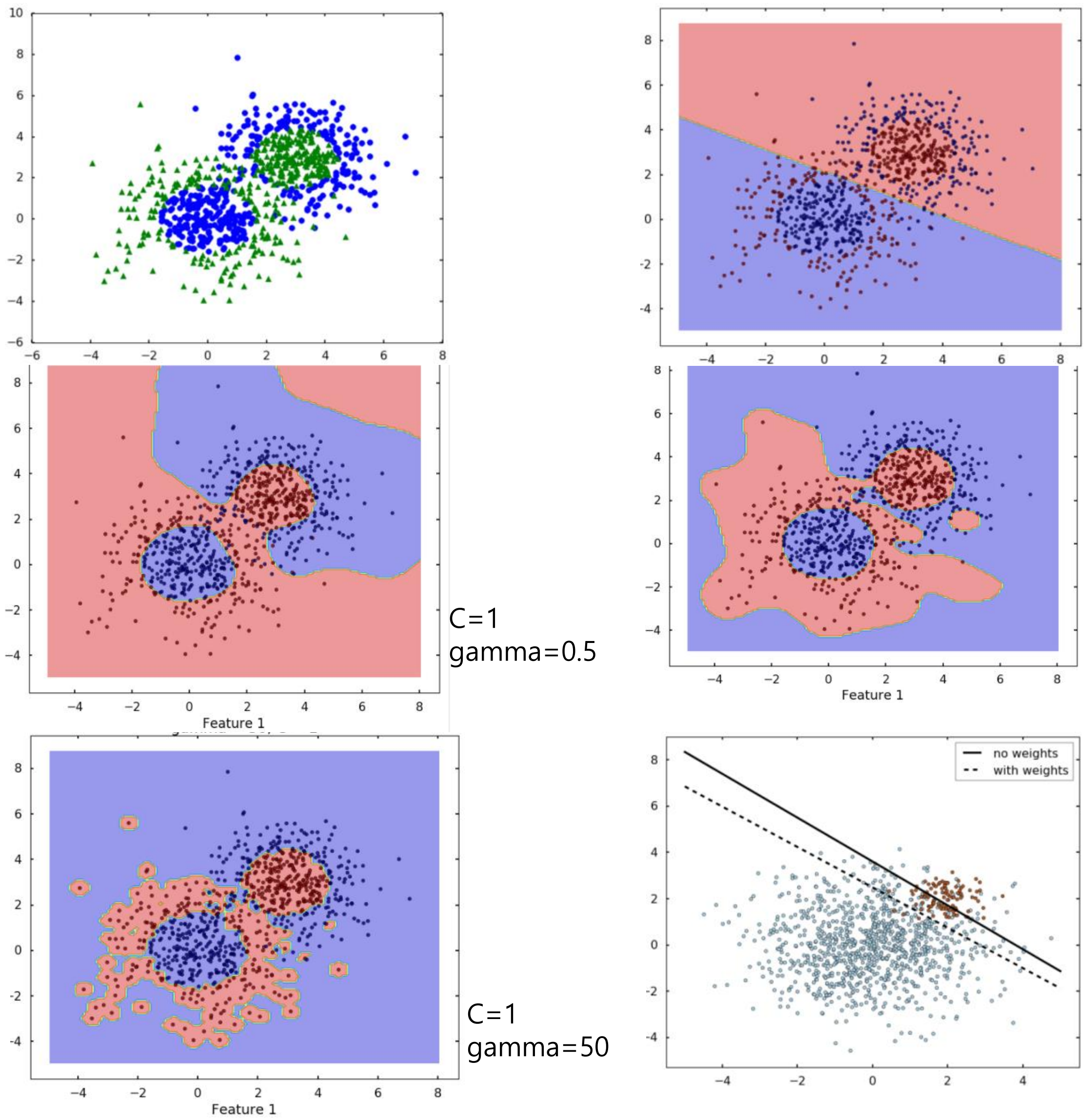




SVM with kernel given by φ((*a*, *b*)) = (*a*, *b*, $a^2 + b^2$)

Supervised Learning

- **SVM (continued)**
  - ‘**rbf**’ kernel (hyperparameters: C and gamma)
  - C: tradeoff between classification error and simplicity of the boundary
  - gamma: defines how far the influence of a single example reaches (high value is ‘close’)
  - class_weight: imbalance cases



linear

C=1
gamma=0.5

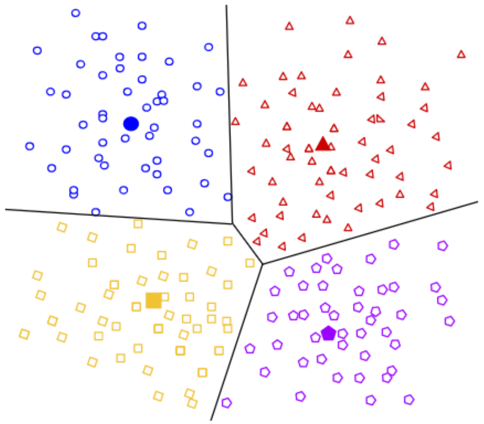
C=10000
gamma=0.5

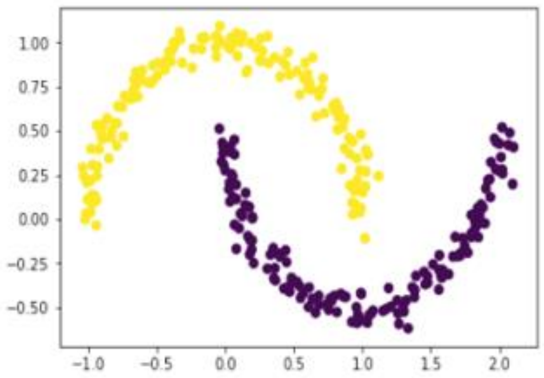
C=1
gamma=50

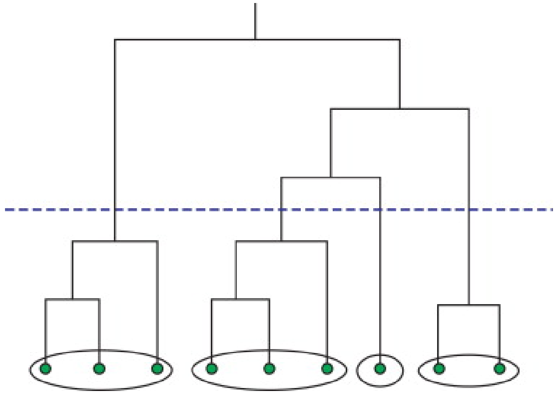
Linear with
class_weight=1:10

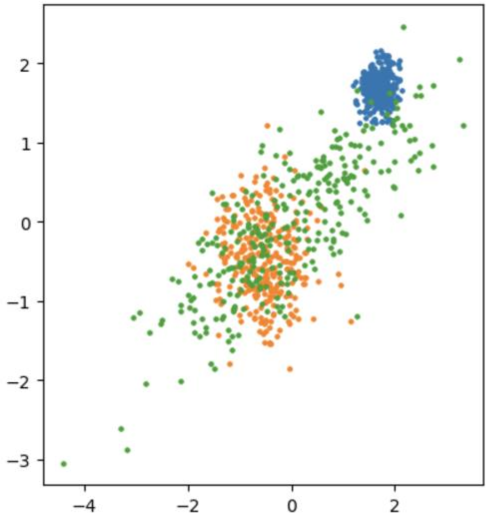
Unsupervised Learning

- **Clustering (or Grouping)**
  - Divide the data points into several clusters based on <u>similarity (유사도)</u>
  - Need scaling as a preprocessing step
  - Applications: detect hackers and criminal activity, identifying fake news, etc.
  - Centroid-based (K-means)
  - Density-based (DBSCAN)
  - Hierarchical (ex: dendrogram)
  - GMM (Gaussian Mixture Model)



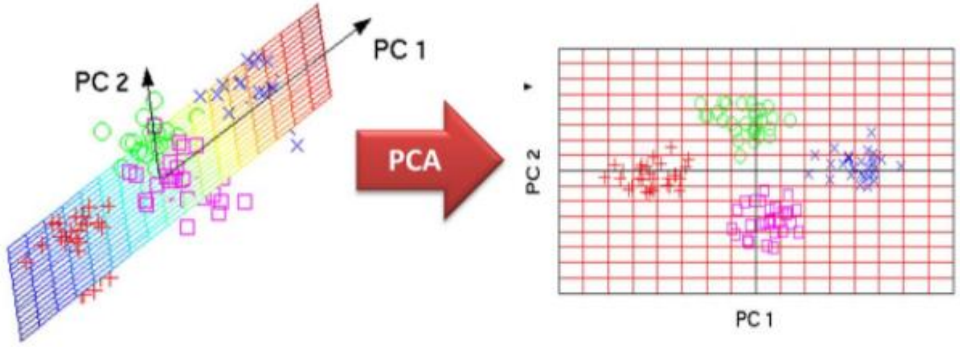






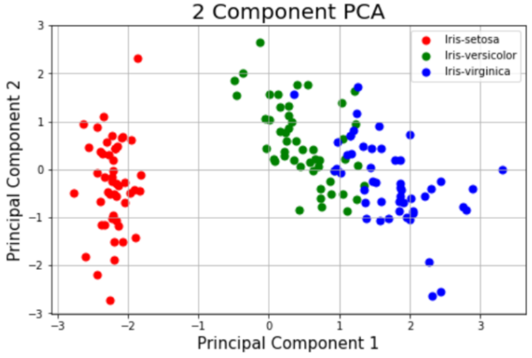
Unsupervised Learning

- **Dimension reduction by PCA**
  - Standard scaling
  - Calculate covariance (or correlation) matrix
  - Eigen-decomposition ($A = P \Lambda P^{-1}$)
  - Select k eigenvectors
  - pca_result = PCA(n_components=2).fit_transform(X_all)
  - Also, tSNE(), Autoencoder
  - SelectPercentile()



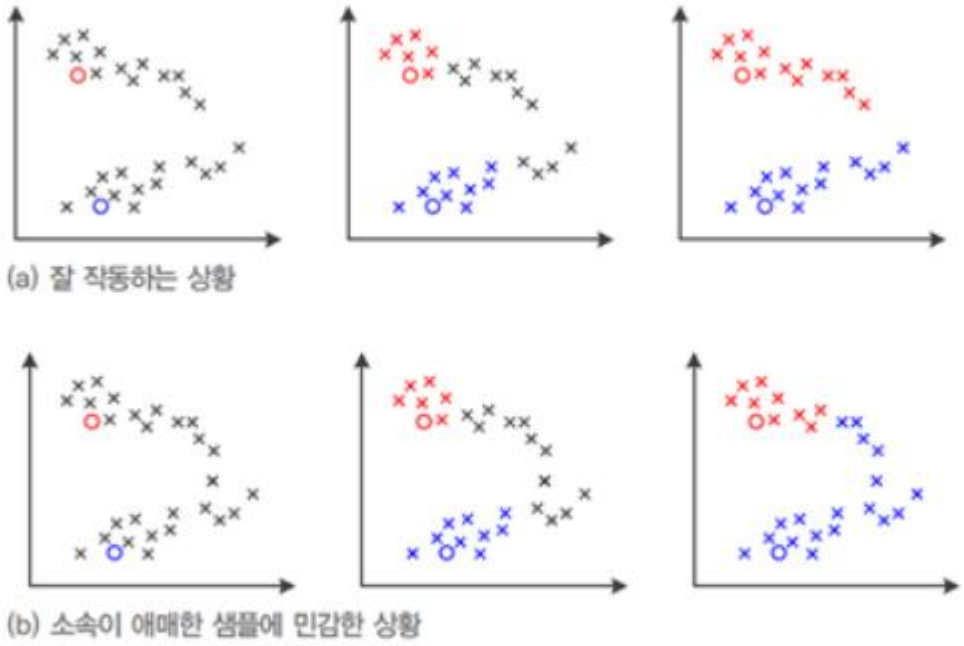
	sepal_len	sepal_wid	petal_len	petal_wid	class
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

	principal component 1	principal component 2	species
0	-2.264542	-0.505704	Iris-setosa
1	-2.086426	0.655405	Iris-setosa
2	-2.367950	0.318477	Iris-setosa
3	-2.304197	0.575368	Iris-setosa
4	-2.388777	-0.674767	Iris-setosa

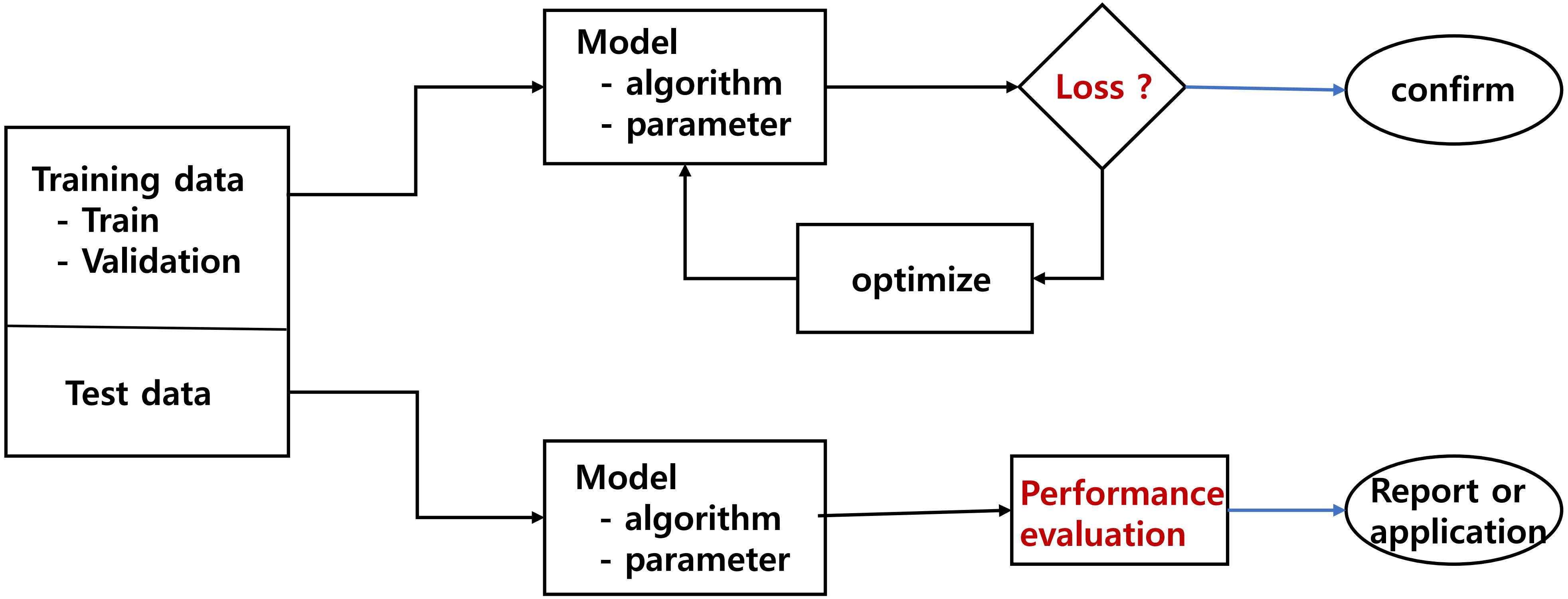


Semi-supervised Learning

- **Semi-supervised**
  - large unlabeled data with small labeled data
  - Will unlabeled data be helpful? Yes or No
  - Self-training:
    - Train with labeled data
    - Classify unlabeled data
    - Select samples with **high confidence** and include them in labeled data
    - Retrain the classifier
    - Repeat
  - Using GAN
  - Many others



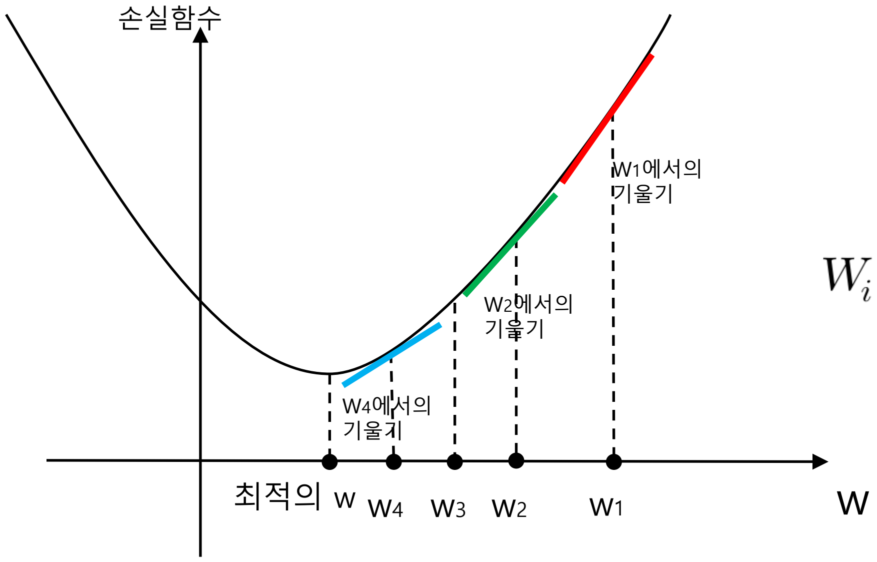
Machine Learning Model (supervised)



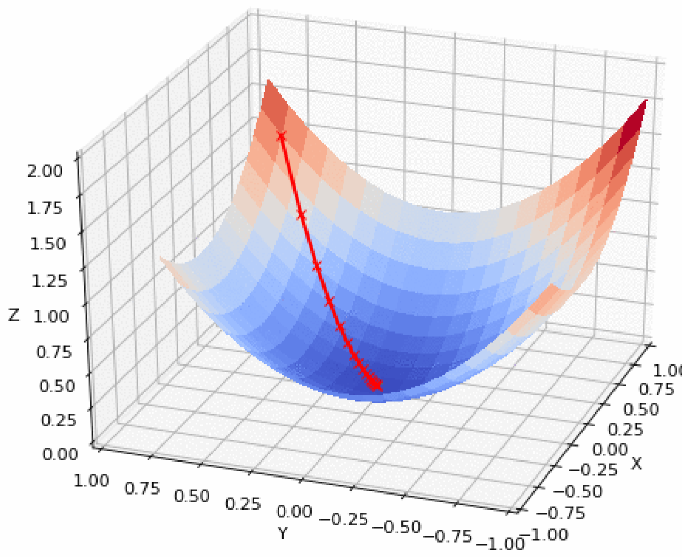
- (model) Parameter: estimated from the dataset (learned during training from the historical data sets)
- Hyper-parameter: external to the model (defined manually before the model training by trial-error)

Gradient Descent (GD) algorithm

- **Gradient Descent (경사하강법)**
  - General optimization algorithm
  - take repeated steps in the opposite direction of the **gradient** (or approximate **gradient**) of the function at the current point

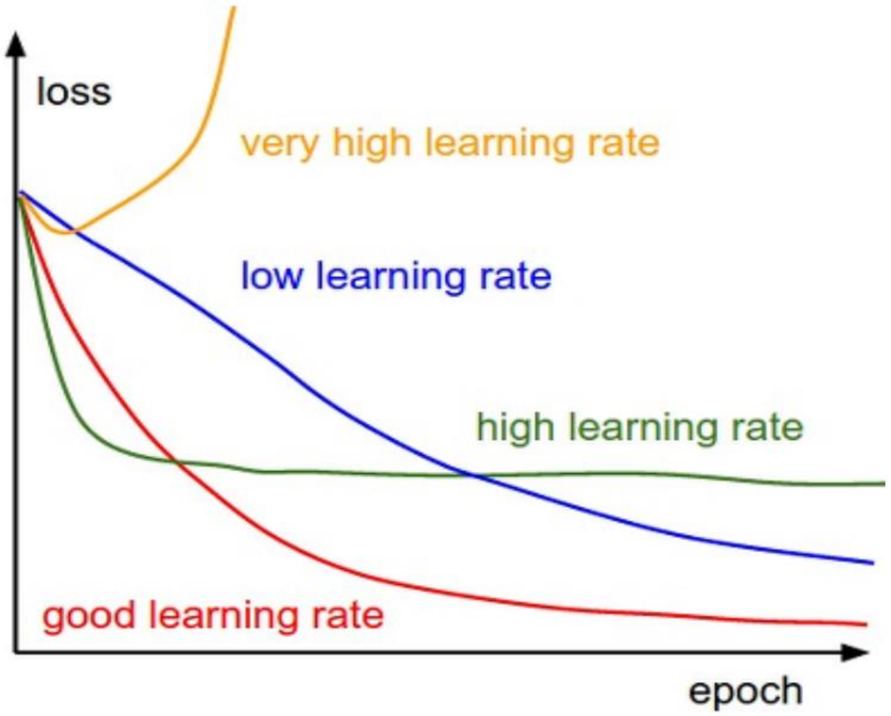


$$W_i = W_{i-1} - \eta Grad(i)$$



Gradient Descent (GD) algorithm

- **Learning rate: η (eta)**
  - low: takes time to converge, and may get stuck in an undesirable local minimum
  - high: may jump over minima
  - too high: may diverge
  - Need adaptive adjustment



Loss Function

- **What to reduce? (Loss or Error or Cost: 손실함수)**
  - Regression (회귀): MSE (Mean Square Error)

$$MSE = \sum_{k=1}^{N} (y - \hat{y})^2$$

  - Classification (분류): Cross Entropy (CE), Gini Coefficient

$$CE = \sum_{i} p_i \log\left(\frac{1}{p_i}\right) \qquad Gini = 1 - \sum_{k=1}^{m} p_k^2$$

  - Binary case:

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Linear Regression

- **Gradient Descent Algorithm**
  - **Loss** (or **cost**) function: MSE (mean square error)

$$\mathcal{L}(y, t) = \frac{1}{2}(y - t)^2$$

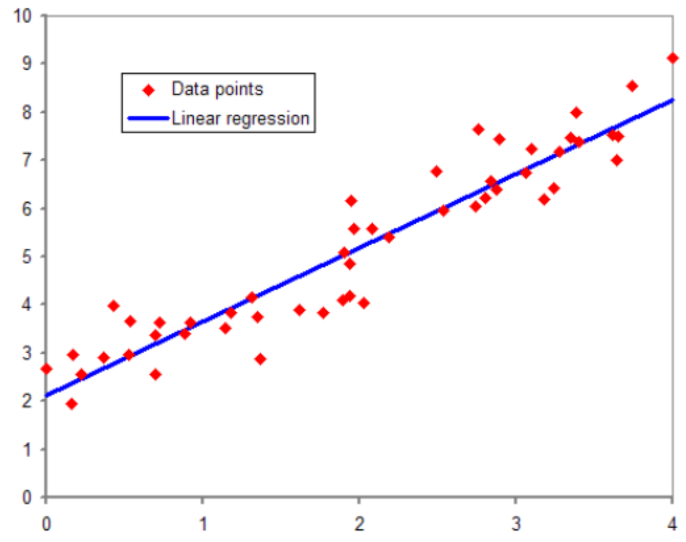
$$\begin{aligned} w_j &\leftarrow w_j - \alpha \frac{\partial \mathcal{J}}{\partial w_j} \\ &= w_j - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, x_j^{(i)} \end{aligned}$$

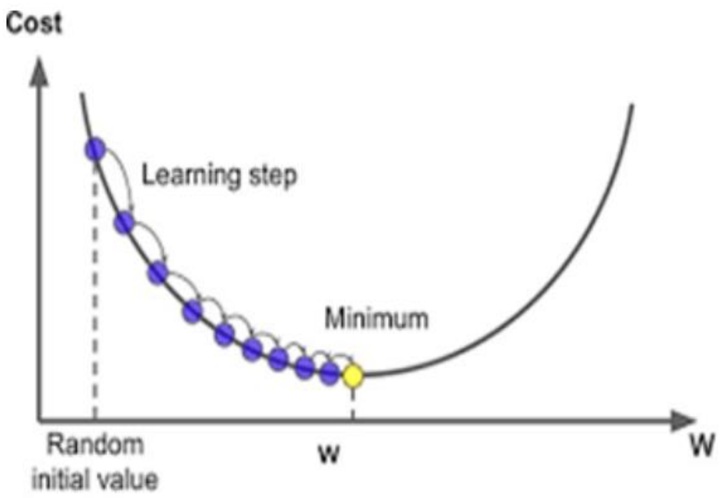
  - In vector form

$$\boxed{\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \alpha \frac{\partial \mathcal{J}}{\partial \mathbf{w}} \\ &= \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, \mathbf{x}^{(i)} \end{aligned}}$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial \mathcal{J}}{\partial w_1} \\ \vdots \\ \frac{\partial \mathcal{J}}{\partial w_D} \end{pmatrix}$$

$$y = wx + b$$

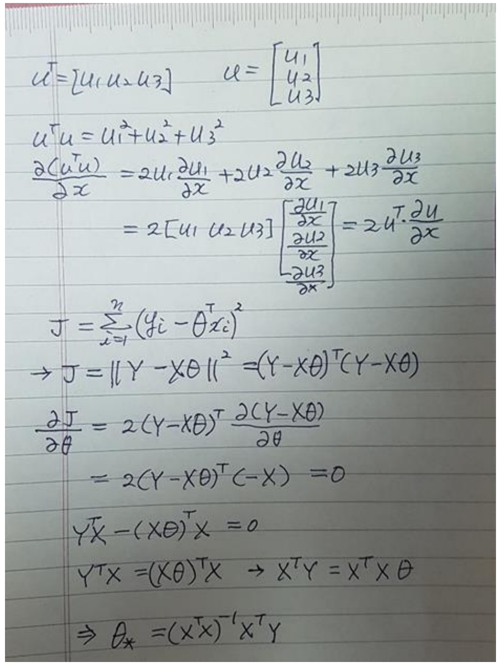


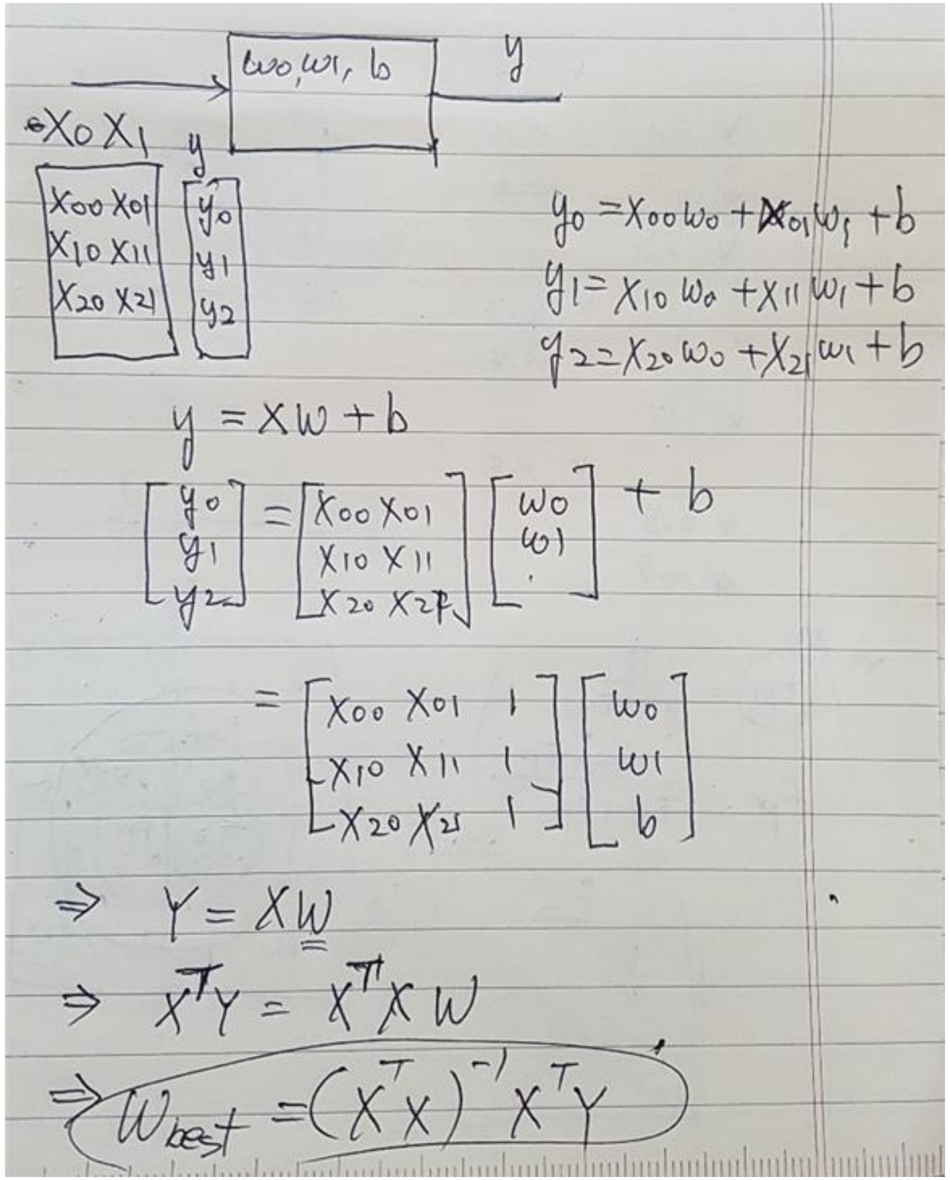


Linear Regression

- **Normal equation (OLS: Ordinary Least Squares)**
  - LinearRegression() in sklearn library
  - Don’t require learning rate (no iterative steps)
  - High computational complexity (inverse requires $O(n^3)$ complexity)

$$\hat{\theta} = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$





Linear Regression

- ## OLS regression analysis

```
sm.OLS(y_train, X_train_ols).fit().summary()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       SystemProduction   R-squared:                       0.641
Model:                            OLS   Adj. R-squared:                  0.641
Method:                 Least Squares   F-statistic:                     2084.
Date:                Tue, 26 Nov 2024   Prob (F-statistic):               0.00
Time:                        09:14:26   Log-Likelihood:                -57635.
No. Observations:                7008   AIC:                         1.153e+05
Df Residuals:                    7001   BIC:                         1.153e+05
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        697.0526     10.787     64.618      0.000     675.906     718.199
x1            19.7436     11.578      1.705      0.088      -2.953      42.440
x2          -214.2258     17.908    -11.963      0.000    -249.331    -179.121
x3           -66.7511     10.909     -6.119      0.000     -88.136     -45.366
x4          1196.0869     19.381     61.714      0.000    1158.094    1234.080
x5            86.6055     12.972      6.676      0.000      61.177     112.034
x6          -177.8948     15.225    -11.685      0.000    -207.739    -148.050
==============================================================================
Omnibus:                     1086.969   Durbin-Watson:                   2.017
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            19237.530
Skew:                           0.040   Prob(JB):                         0.00
Kurtosis:                      11.116   Cond. No.                         3.74
==============================================================================
```

결정계수 R^2 (R-Squared):
모델이 종속 변수의 변동을 얼마나 잘 설명하는지 나타냄.
- **R^2** = 0: 모델이 아무것도 설명하지 못함.
- **R^2** = 1: 모든 변동을 완벽히 설명.

F-통계량 (F-test)
- 회귀모델 전체의 유의성을 평가
- H0: “모든 회귀 계수가 0이다" (독립 변수들이 종속 변수에 영향을 미치지 않는다)
- P-value < 0.05 (reject H0 -> 모델이 유의미하다)

개별독립 변수의 유의성
- 각 독립 변수가 종속 변수에 유의미한 영향을 미치는지를 평가 (t-검정)
- H0: “해당 독립 변수의 회귀 계수 $\beta i=0$, 즉 종속 변수에 영향을 미치지 않는다”

Omnibus: 잔차(residuals)가 정규성을 따르는지를 검정 (잔차의 왜도와 첨도를 함께 분석)
- H0: “잔차가 정규분포를 따른다”
- prob(omnibus): p-value < 0.05 (reject H0, 정규성을 벗어남)

Linear Regression

- **(결과 해석)** F-test, t-test 는 모두 유의미하지만 잔차는 정규성을 벗어나고 있다.

  (1) **비선형성 존재:**
  - 독립 변수와 종속 변수의 관계가 선형이 아닌 경우, 회귀 모델은 오차를 정규적으로 처리하지 못한다.
  - 하지만, 모델은 여전히 전체 데이터의 일부 패턴을 학습하여 유의미한 결과를 낼 수 있다.

  (2) 이상치(Outliers):
  - 데이터셋에 이상치가 포함되면 잔차의 분포가 왜곡될 수 있다.
  - 이상치는 잔차의 정규성을 심각하게 훼손하지만, t-test나 F-test에는 상대적으로 덜 민감할 수 있다.

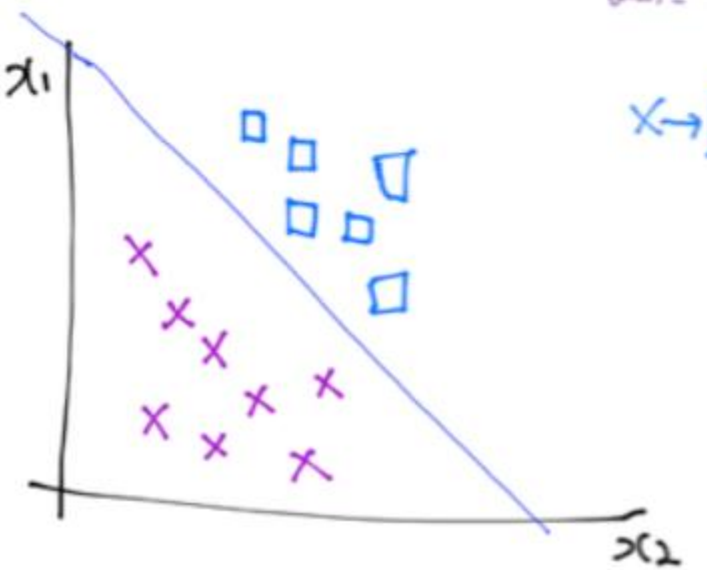
  (3) 데이터 변환의 필요성:
  - 독립 변수나 종속 변수가 특정 분포(예: 비정규 분포)를 가진다면, 데이터 변환(예: 로그 변환, Box-Cox 변환)이 필요할 수 있다.
  - 변환을 하지 않으면 잔차의 정규성이 깨질 가능성이 크다.

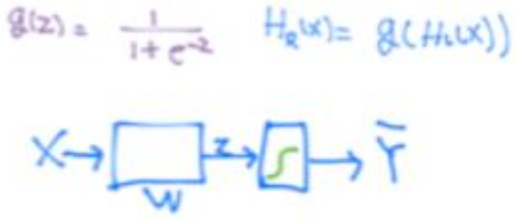
  (4) 다중공선성(Multicollinearity):
  - 독립 변수들 간 강한 상관관계가 있는 경우, 잔차가 정규성을 벗어나면서도 모델 자체는 유의미한 결과를 보여줄 수 있다.

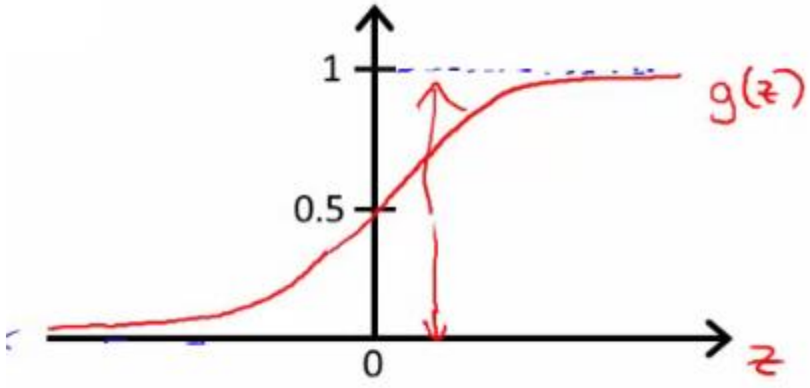
  (5) 해결방안:
  - 비선형 모델 사용, 이상치 제거, 데이터 변환 시도, 높은 상관관계 변수 제거, 등

Linear Classification

- **Logistic Regression Classifier (Binary)**





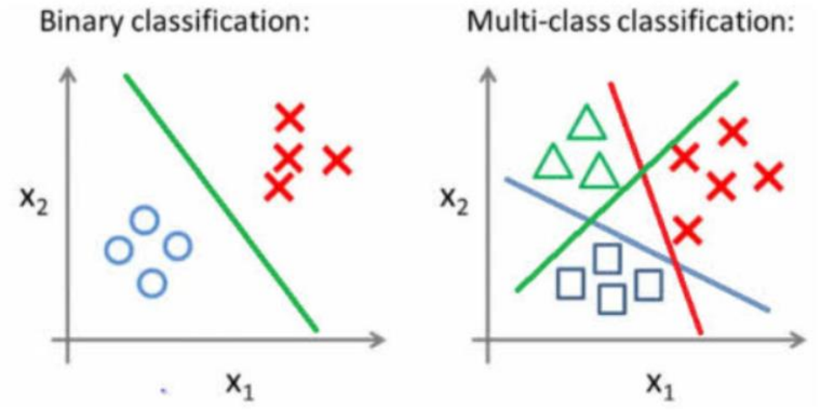


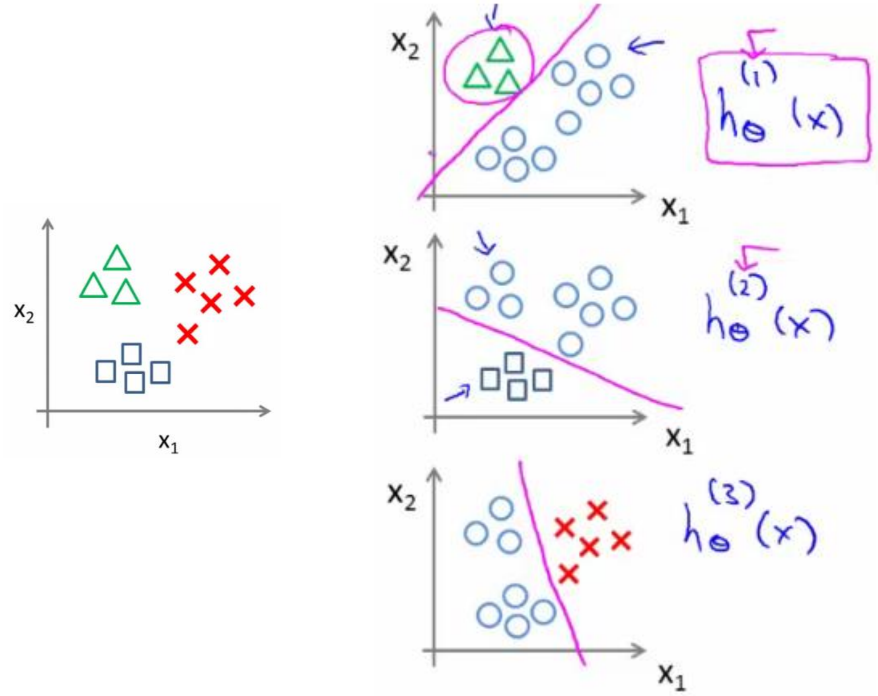
- Since this is a binary classification task we know y = 0 or 1
  - So the following must be true
    - $P(y=1|x ; \theta) + P(y=0|x ; \theta) = 1$
    - $P(y=0|x ; \theta) = 1 - P(y=1|x ; \theta)$

Ref: http://holehouse.org/mlclass/06_Logistic_Regression.html

Linear Classification

- **Binary classification vs. Multi-class classification**





- Multi-class classification (with *n*-classes)
  - One vs. All: *n* binary classifier models (preferred)
    but, prone to creating an imbalance
  - One vs. One: ${}_nC_2$ = $n*(n-1)$ binary classifier models

https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b

Performance Metrics - Regression

- **MSE and MAE**
  - MSE(Mean squared error)
  - MAE(Mean absolute error)
  - MAE 가 이상치(outlier)에 대해서는 MSE 보다 robust 하다고 알려짐.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad \mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

- **R Squared (R^2)**
  - 실제 출력 값에 대한 예측 출력 값 세트의 우수성 또는 적합성 표시
  - Numerator(MSE), denominator(variance)
  - What if all Yi is correctly predicted?
  - What if all Yi is predicted as the mean?

SS: sum of squares around mean, fit

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2} = \frac{SS(\text{mean}) - SS(\text{fit})}{SS(\text{mean})}$$

- R^2=0 means that the model predicts the expected value of *y* disregarding the input features.

Performance Metrics - classification

- **Classification – static, dynamic**
- **Static: confusion matrix (a.k.a. Error matrix)**
  - Tabular visualization of the model prediction vs. ground-truth labels
  - Row: the instances in a predicted class
  - Column: the instances in an actual class

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

  - True-positive (100), false-negative (5)
  - true-negative (50), false-positive (10)

Performance Metrics - classification

- **Accuracy**
  - Number of correct predictions divided by the total number of predictions
  - (ex) accuracy = (100+50)/165
- **Precision**
  - In some cases, accuracy is not a good indicator of your model performance, for example, when your class distribution is imbalanced.
  - Precision_yes = 100/110, Precision_noncat = 50/55
  - Says "**How reliable your prediction is…**" or "how much I can trust…"
- **Recall**
  - Recall_yes = 100/105
  - Recall_no = 50/60
  - Says "**how many actual class samples are correctly predicted**…"
- **F1-Score**
  - Combine the above two metrics (precision and recall) as harmonic mean:

  F1-score = 2*Precision*Recall / (Precision+Recall)

Performance Metrics - classification

- **Dynamic: ROC Curve (Receiver Operating Characteristic curve)**
  - Shows the performance of a binary classifier as function of its cut-off threshold
  - It essentially shows the true positive rate(tpr) against the false positive rate(fpr) for various threshold values.
  - **AUC** (Area Under Curve)
- **As an example,**
  - Suppose your model predicts 4 sample images with probabilities [0.45, 0.6, 0.7, 0.3].
  - Then, depending on the threshold, you will get different labels:

    cut_off=0.5: predicted_value=[0,1,1,0] (default threshold)

    cut_off=0.2: predicted_value=[1,1,1,1]

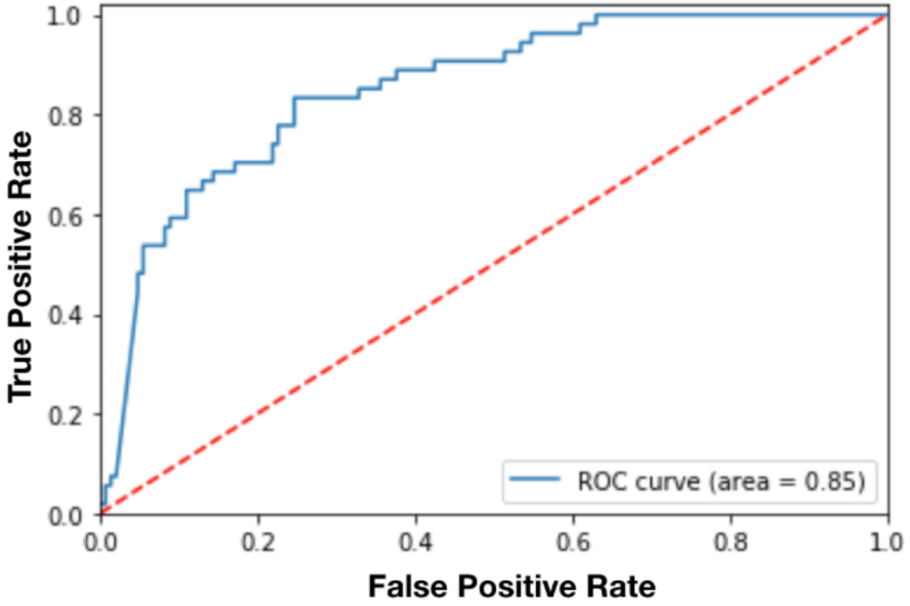
    cut_off=0.8: predicted_value=[0,0,0,0]

  - **Making different confusion matrix depending on the threshold**.

ROC/AUC (Example)

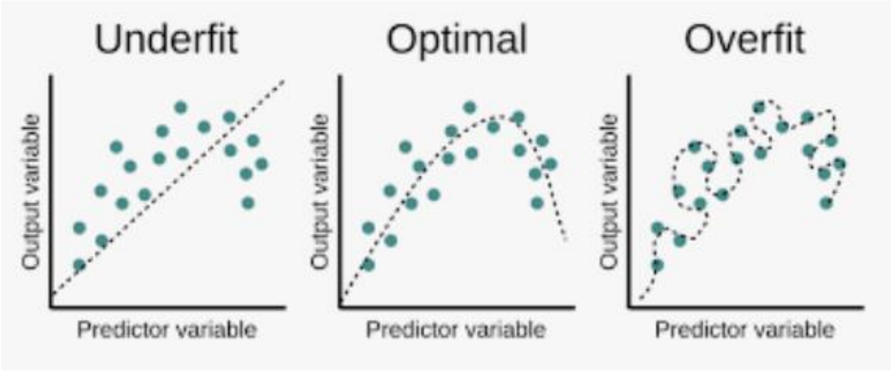
환자번호	성별	점수	순위	실제 값	TPr	FPr
7	F	0.98	1	N (0)	0	1/m
125	M	0.96	2	C (1)	1/n	
4	F	0.95	3	N		2/m
199	M	0.86	4	C	2/n	
2	F	0.84	5	N		3/m
200	M	0.82	6	C	3/n	
176	M	0.81	7	C	4/n	
73	M	0.80	8	N		4/m
82	M	0.79	9	C	5/n	
3	F	0.77	10	N		5/m
123	F	0.76	11	N		6/m
		...		C	6/n	
43	F	0.48	198	N	..	7/m
93	M	0.42	199	N	..	..
120	F	0.40	200	N	1	1

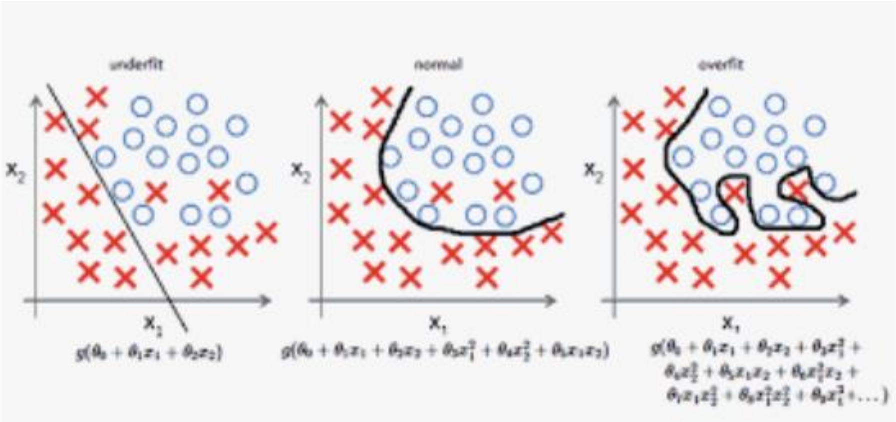
(*) score : probability of cancer
(*) depending on the threshold, you will get different confusion matrix



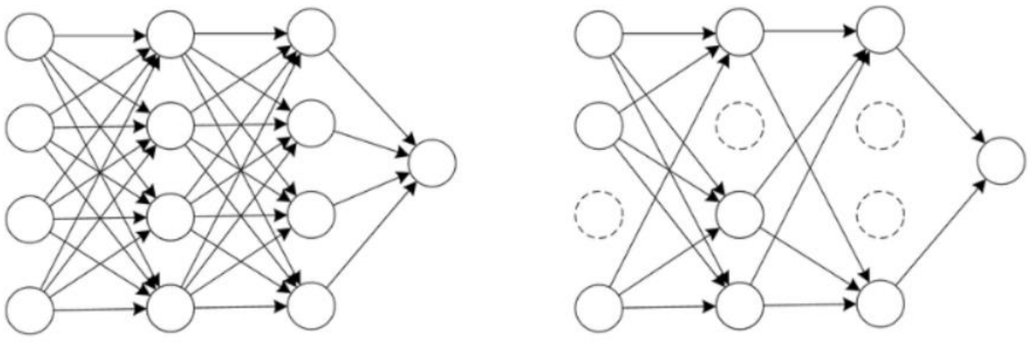
Overfitting and Underfitting

- Overfitting and Underfitting





- How to reduce overfitting?
  - More data
  - Data augmentation
  - Simplify the model
  - Feature selection (or reduction)
  - Regularization
  - Hyperparameter tuning
  - Early stopping
  - Dropouts

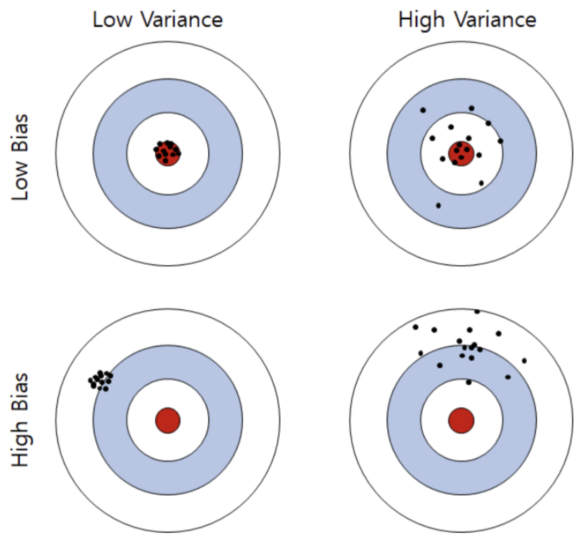


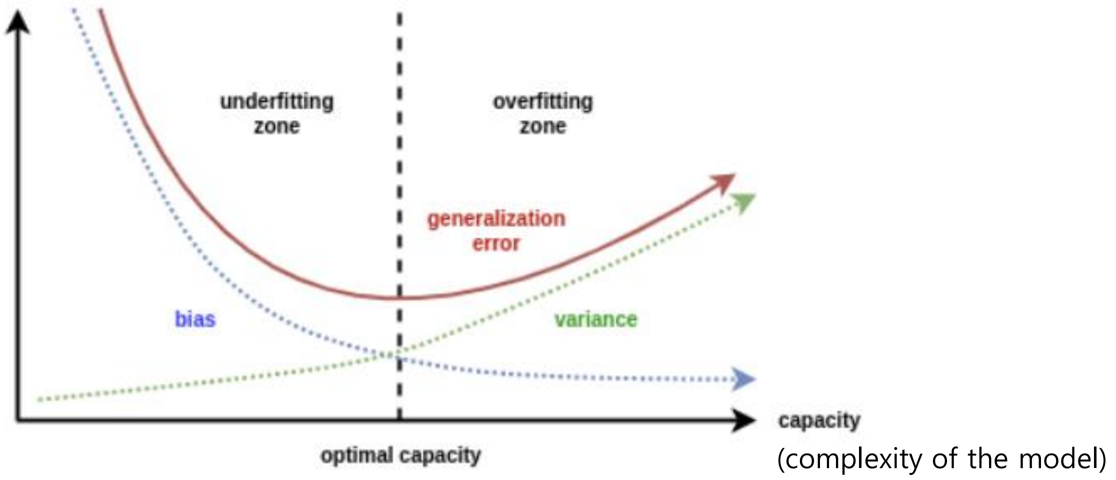
(a) Standard Neural Network

(b) Network after Dropout

Bias and Variance

- Bias-Variance tradeoff (편향과 분산)
  - Bias: difference between the average prediction of the model and the correct value (wrong model -> high bias -> underfitting)
  - Variance: variability of model prediction (noisy dataset -> high variance -> overfitting)
  - total error = Bias + Variance + noise

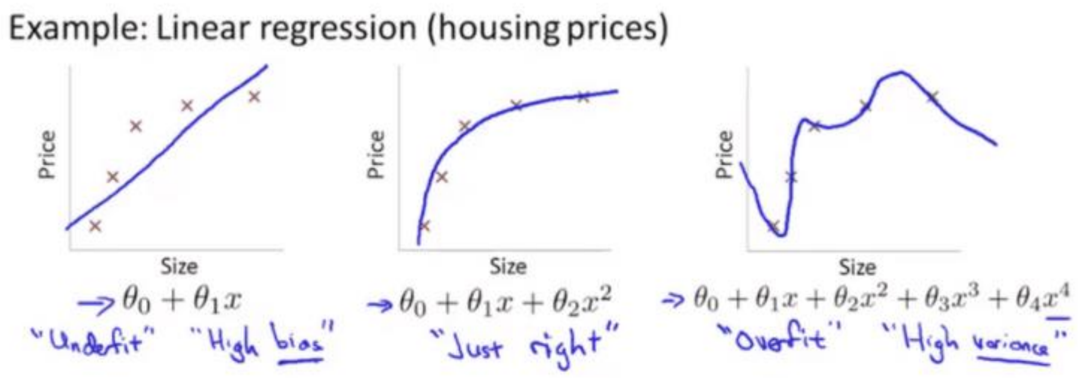




Bias-Variance tradeoff

Regularization

- **Regularization (규제화)**
  - Add information in objective function to reduce model complexity for reducing overfitting (regression and classification)
  - **Ridge** (L2): give more penalties on <u>large-valued coefficients</u>
  - **Lasso** (L1) (Least Absolute Shrinkage and Selection Operator): shrinks the <u>less important features' coefficient</u> to zero (good for **feature selection**)
  - **Elastic net**: use the both



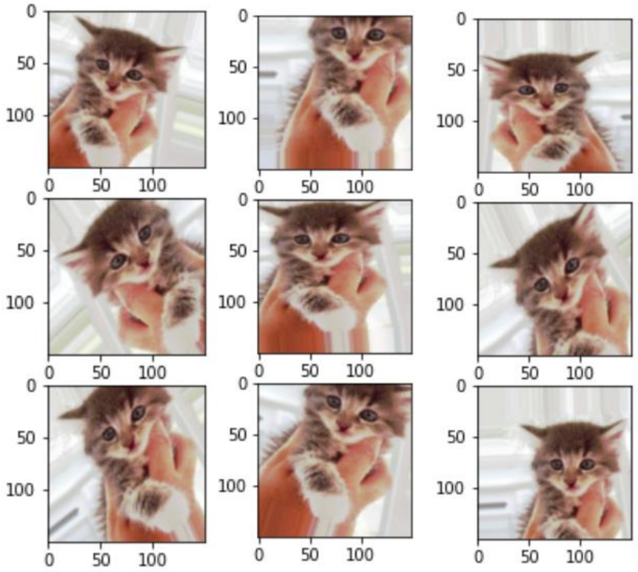
$$J(W) = MSE(W) + \alpha \frac{1}{2} \sum_{i=1}^{n} W_i^2$$

$$J(W) = MSE(W) + \alpha \sum_{i=1}^{n} |W_i|$$

$$J(\theta) = MSE(\theta) + \gamma\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-\gamma}{2} \alpha \sum_{i=1}^{n} \theta_i^2$$

Augmentation

- **Augmentation (증강 or 확장)**
  - Increase the amount of data by adding slightly modified copies of existing data or newly created synthetic data from existing data
  - Introducing new synthetic images: transformation, GAN, image synthesis
  - Data augmentation for speech recognition based on RNN



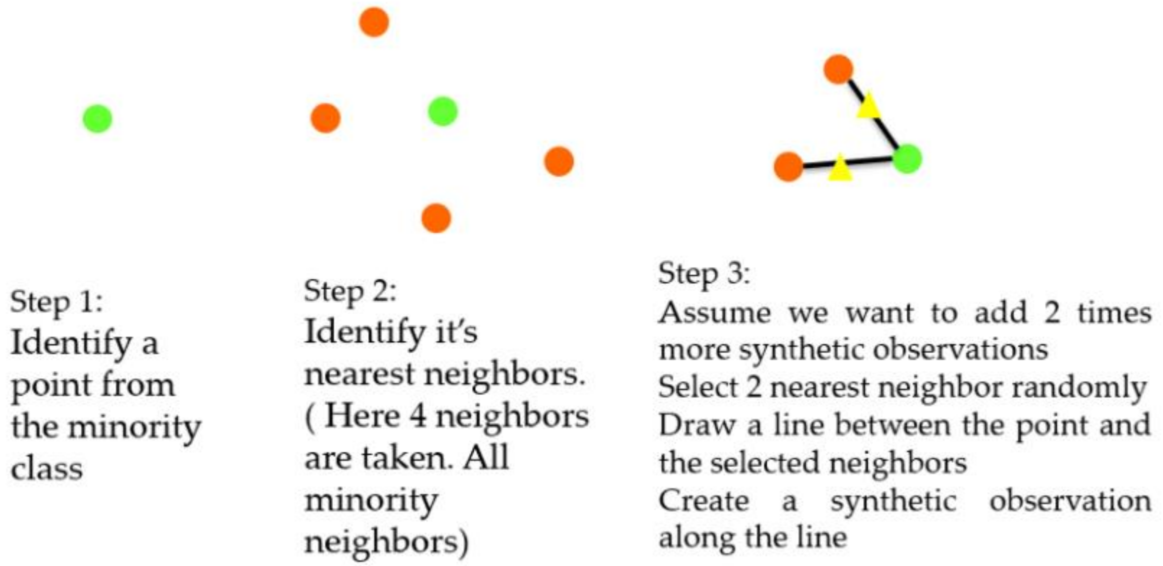
```
train_datagen = ImageDataGenerator(
    rescale= 1./255,
    rotation_range = 40,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip = True)
```

Imbalance Problem

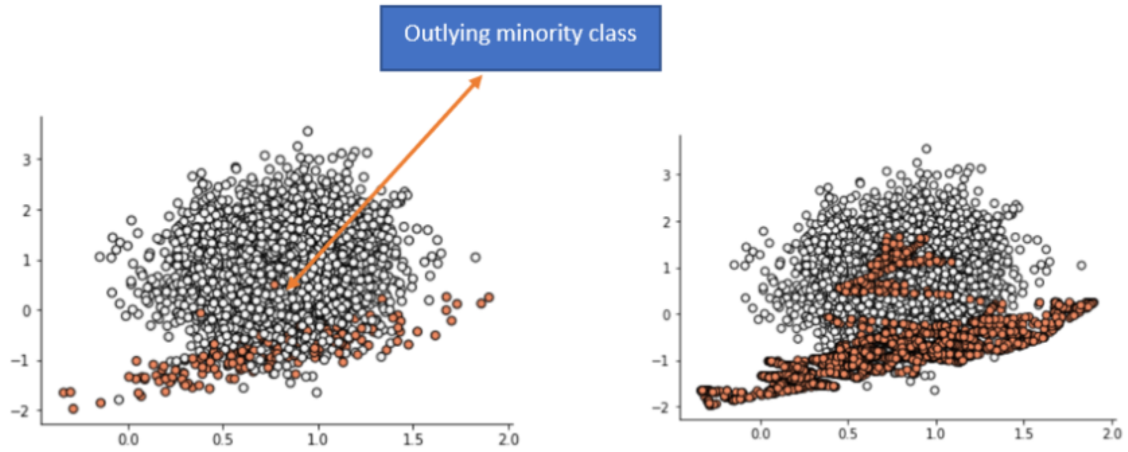
- **Class imbalance problem**
  - Class distributions are highly imbalanced in the training dataset
  - Tend to have low prediction accuracy for the infrequent (minority) class
  - Exists in many real word classification problems, such as fraud detection, spam detection, threat-object detection, anomaly detection, etc.
  - Causes: properties of the domain, biased sampling, measurement error
- **How to reduce the imbalance problem?**
  - Artificial resampling: over-sampling (replicating minority class), under-sampling of majority class
  - SMOTE(Synthetic Minority Oversampling TEchnique): make synthetic data points by finding the nearest neighbors to each minority sample
  - Augmentation or Use generative model for synthetic data
  - More weights on minority samples
  - Majority sample selection based on RL
  - Resampling is to be done only on Train dataset (not Test dataset !)
  - Still a hot research topic

Imbalance Problem

- **SMOTE (**Synthetic Minority Oversampling Technique**)**



SMOTE, Synthetic Minority Observation Generation Process (Source: Author)

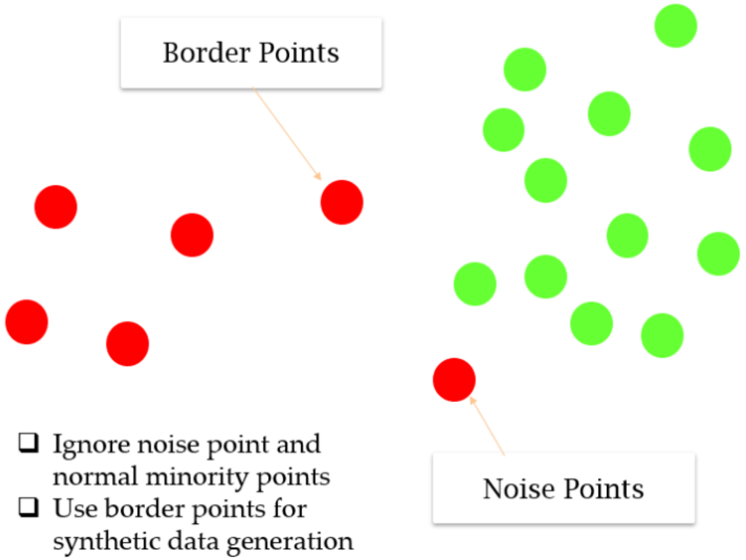


Left Side: Original Data Right Side: Data after SMOTE is Applied (Image Source: Author)

- If there are outlying minority classes and appear in the majority class, it creates a line bridge with the majority class. (problem!)

Imbalance Problem

- **Borderline SMOTE**
  - Classify minority classes as noise point if all neighbors are the majority class – ignored.
  - Classifies a few points as border points that have both majority and minority class.
  - Resample only from border points.
  - **End up giving more attention to extreme observations.**
- More variants
  - ADASYN: generating more synthetic samples in areas where classification is difficult, such as where the density of minority examples is low.
  - and more …



Border Line SMOTE : (Image Source Author)

Scikit-Learn design principle

- **Consistency**
  - **Estimators**: **estimate some parameters based on dataset**
    - `fit(X [,y])` method with some hyper-parameters
  - **Transformers: some estimators can transform a dataset**
    - `transform(X)` method
    - `fit_transform(X [,y])`: equivalent to calling `fit()` and `transform()`
  - **Predictors: making predictions**
    - `predict()` method : returns a dataset of corresponding predictions
    - `score()` method : measure the quality of the predictions
- **Inspection**
  - **Hyper-parameters** are accessed via instance variable (e.g. `imputer.strategy`)
  - Estimator's **learned parameters** are accessed via instance variable with an underscore suffix (e.g. `imputer.statistics_`)