PROBLEM SOLVING USING PATTERN RECOGNITION
DAY 1B

Dr Zhu Fangming

Institute of Systems Science

National University of Singapore

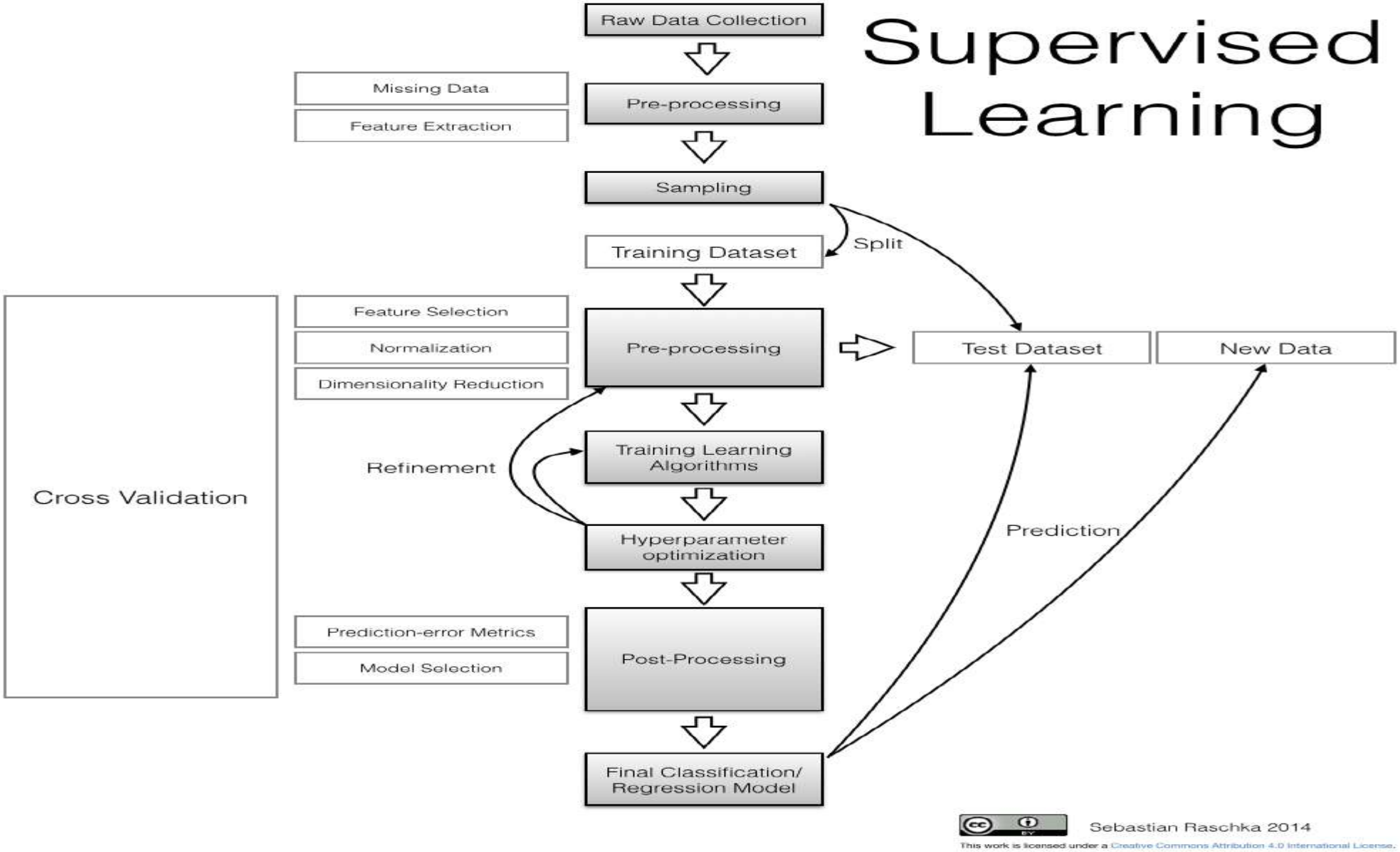
fangming@nus.edu.sg

1.2
HOW TO ANALYSE, MODEL AND SOLVE PATTERN RECOGNITION PROBLEMS

Topics

- Important steps in solving pattern recognition problems
- Important issues for pattern recognition: data pre-processing, feature selection, model evaluation.

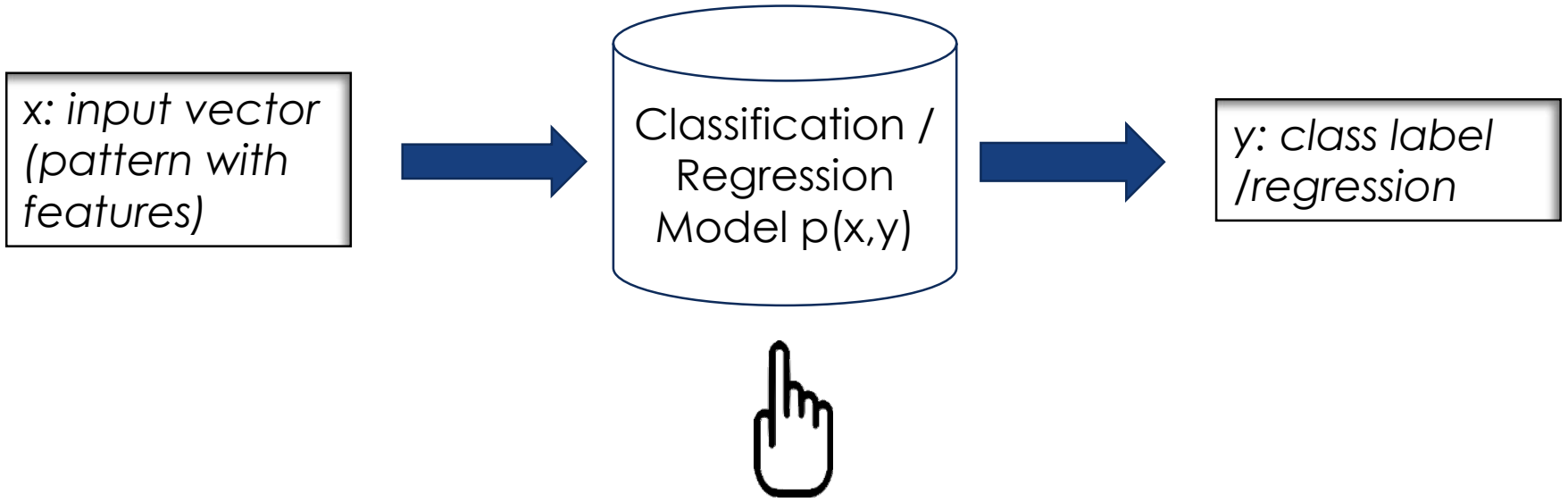
Pattern Recognition Process with Supervised Learning



Models, Features and Classes

- **A pattern is represented by a set of *d* features, or attributes, viewed as a *d*-dimensional *feature vector*.**

$$\boldsymbol{X} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots \boldsymbol{x_d})^{\mathsf{T}}$$



Data Pre-processing

- Data cleaning
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data integration
  - Integration of multiple databases
- Data transformation
  - Normalization and aggregation
- Data reduction
  - Dimensionality reduction - feature selection
  - Numerosity reduction – select/ sample records

Normalization

- Normalization & feature scaling techniques are important for many machine learning algorithms.
- Min-Max scaling

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \longrightarrow [0,1]$$

- Z-score (standardization)

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

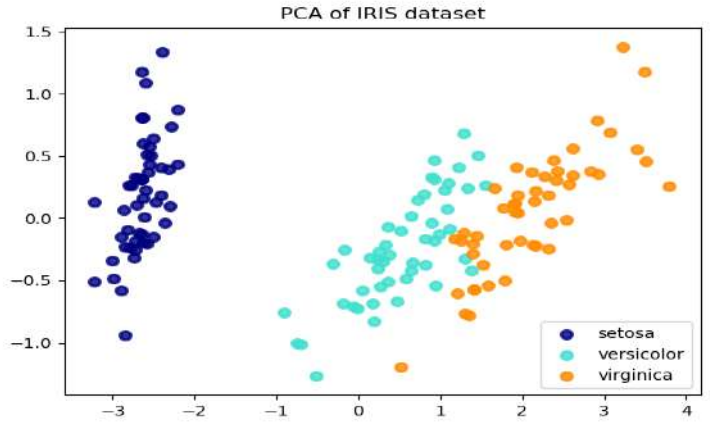
- Use the same parameters on the test dataset and new unseen data.

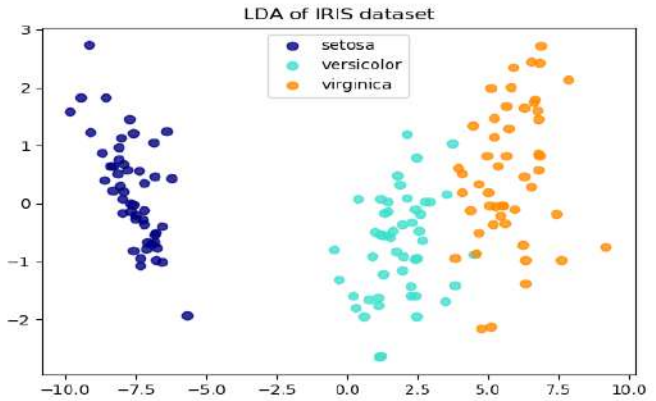
Feature Selection

- Curse of dimensionality
- Retain only "useful" (discriminatory) information and avoid overfitting.
- Reasons to reduce the number of features:
  - Computational complexity
  - Generalization properties

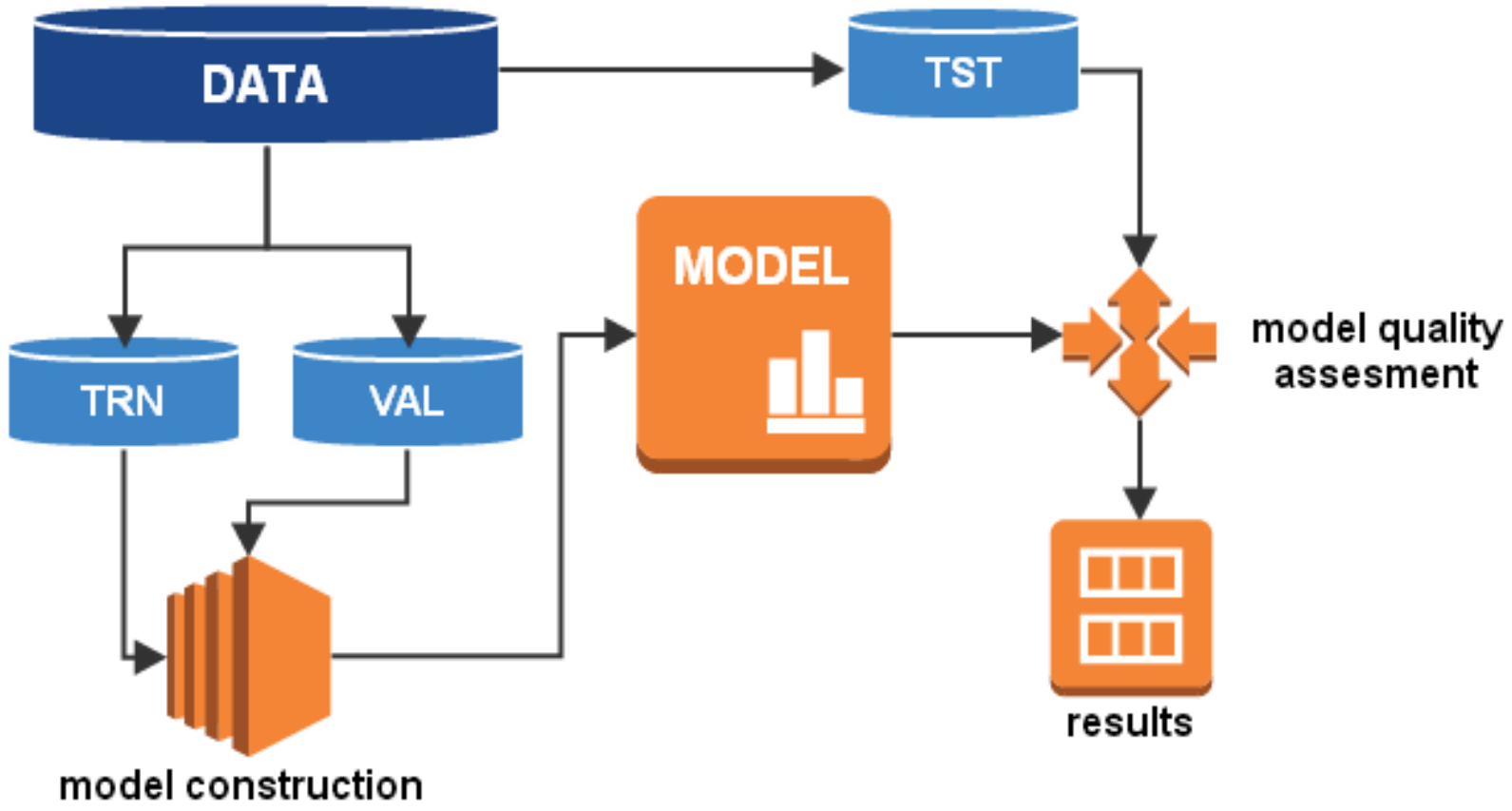
Dimensionality Reduction

- Principal Component Analyses (PCA) and Linear Discriminant Analysis (LDA) can be used.
- Linear Discriminant Analysis (LDA) tries to identify attributes that account for the most variance *between classes*.
- In particular, LDA, in contrast to PCA, is a supervised method, using known class labels.



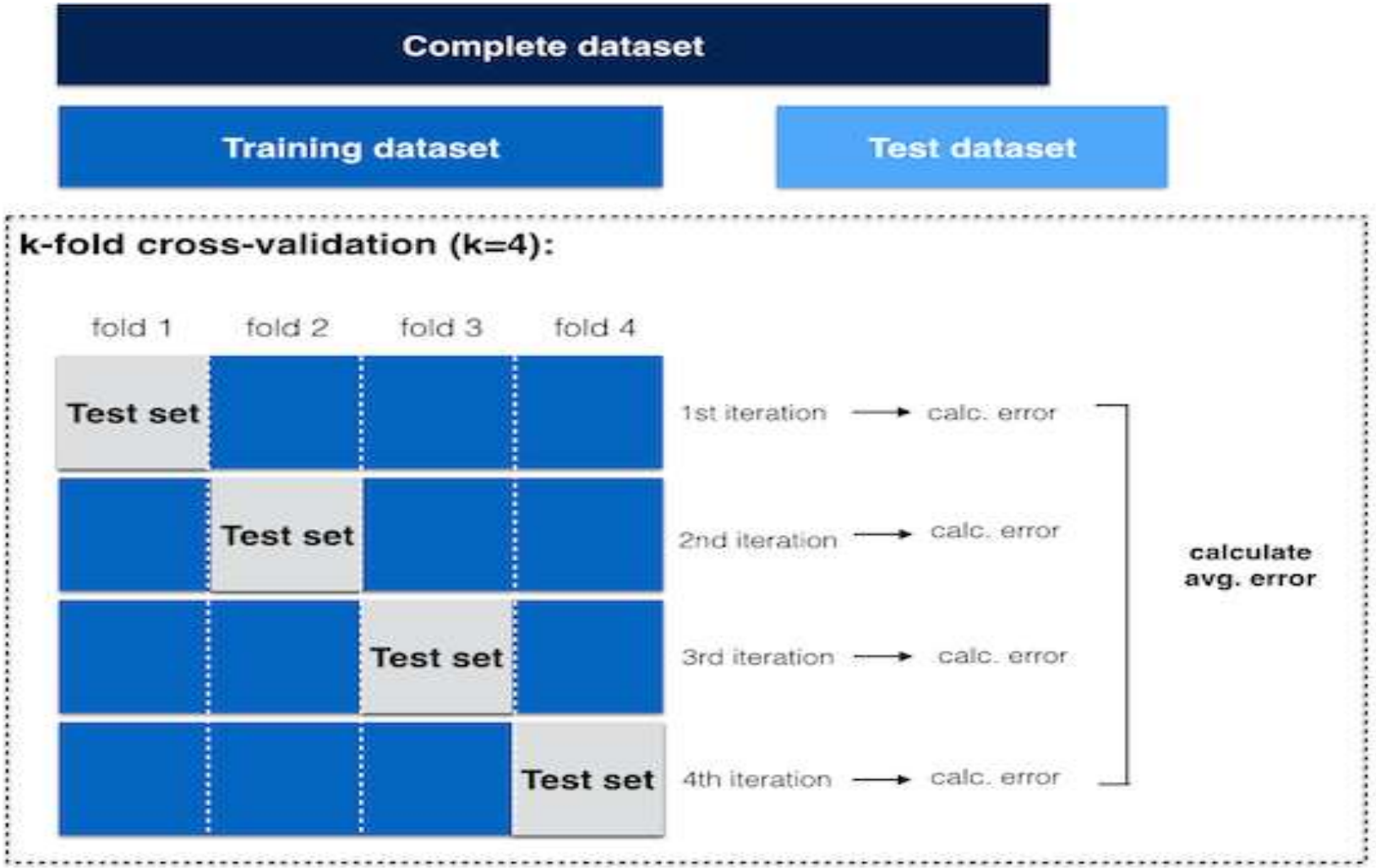


Data Partition and Preparation

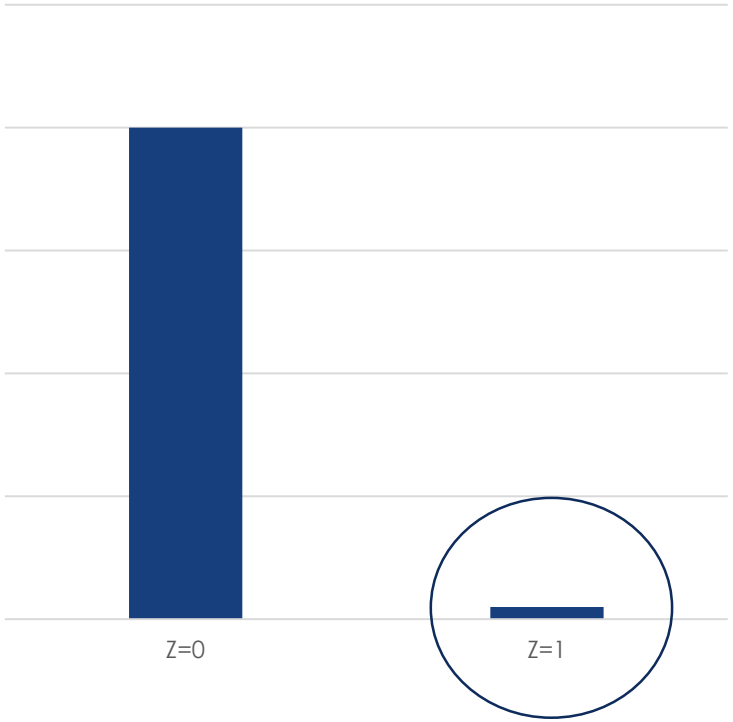


https://www.datasciencecentral.com

Cross Validation



Learning from Imbalanced Data



- Fraud detection
- Churn Modeling
- Anomaly detection

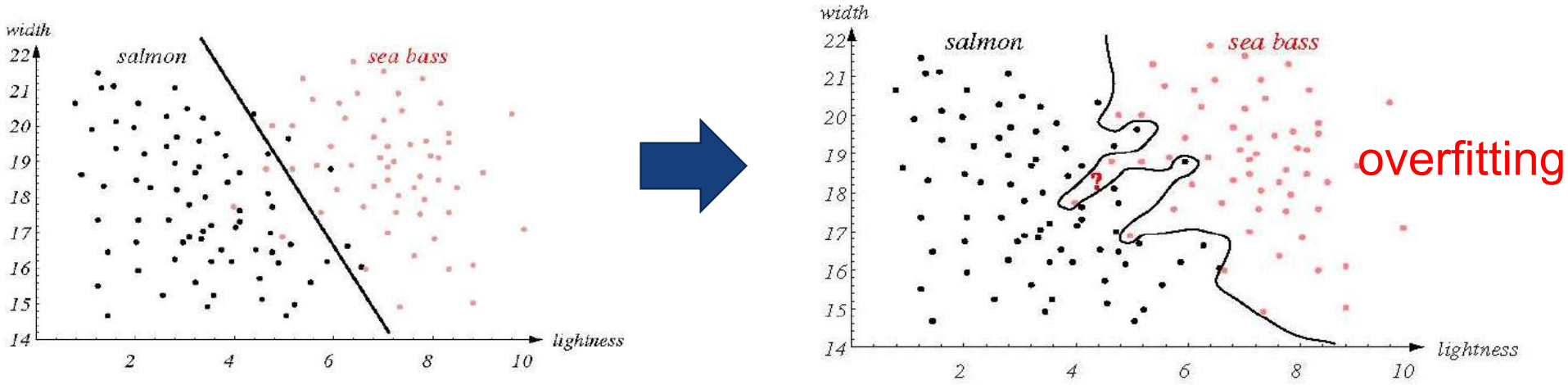
- Techniques for Learning from imbalanced data:
  - Data Augmentation
  - Custom Loss Function

Model Evaluation

- Error Measures
- Overtraining/overfitting
- Confusion Matrix
- ROC Charts
- Gains Chart/ Lift Chart

Overfitting

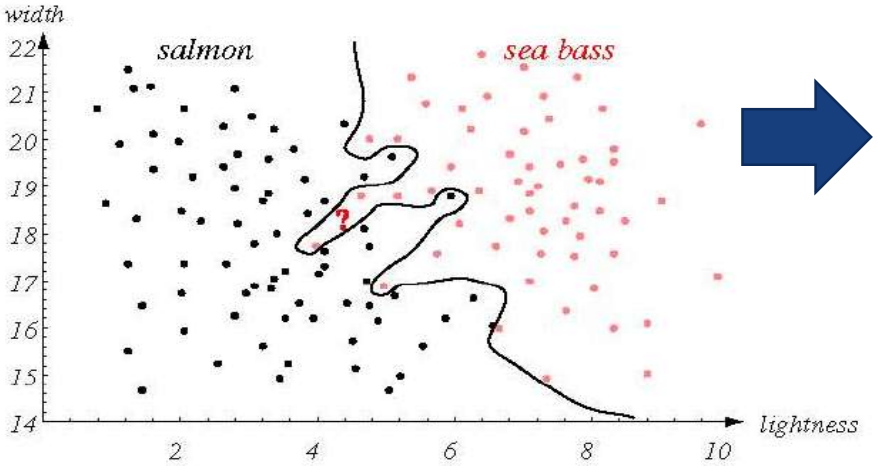
- We can get perfect classification performance on the training data by choosing a more complex model.
- Complex models are tuned to the particular training samples, rather than on the characteristics of the true model.



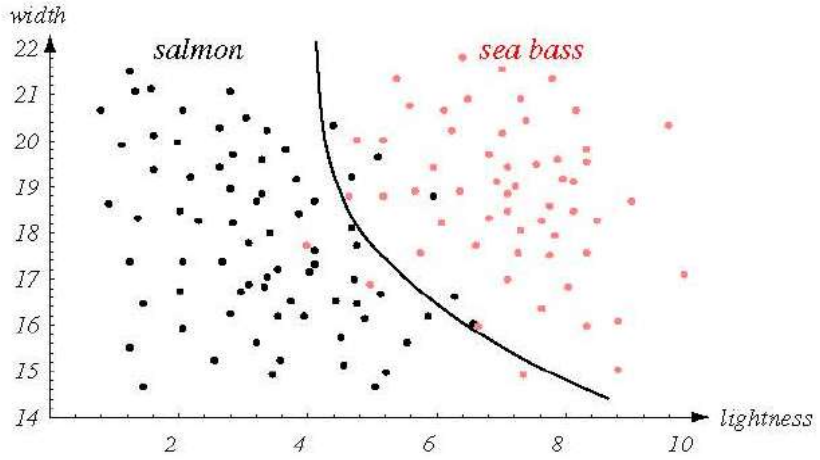
Generalization

- **Generalization is defined as the ability of a classifier to produce correct results on novel patterns.**
- **How can we improve generalization performance ?**
  - More training examples (i.e., better model estimates).
  - Simpler models usually yield better performance.

complex model



simpler model

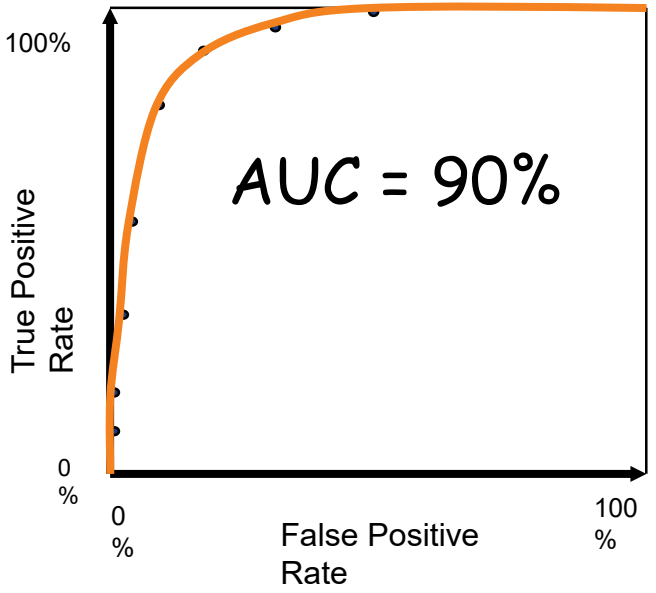


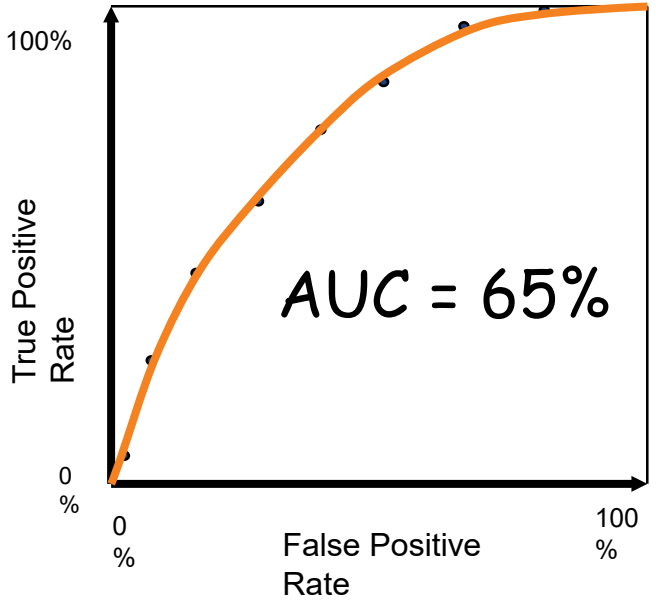
Confusion Matrix

Actual class\Predicted class	Predicted C_1	Predicted ¬ C_1
Actual C_1	**True Positives (TP)**	**False Negatives (FN)** **Type-II Error**
Actual¬ C_1	**False Positives (FP)** **Type-I Error**	**True Negatives (TN)**

Accuracy = (TP + TN)/All
Sensitivity = True Positive Rate = Recall= TP/(TP+FN)
Specificity = True Negative Rate = TN/(FP+TN)
Precision = TP/(TP+FP)
F1 score = 2*Precision* Recall/(Precision + Recall)

ROC (Receiver Operating Characteristic) Curve

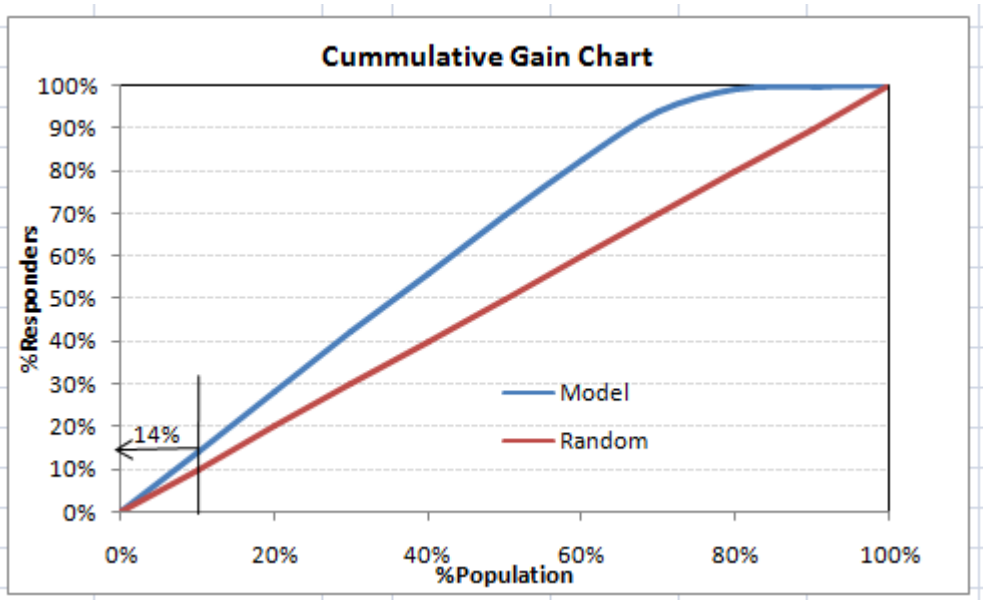


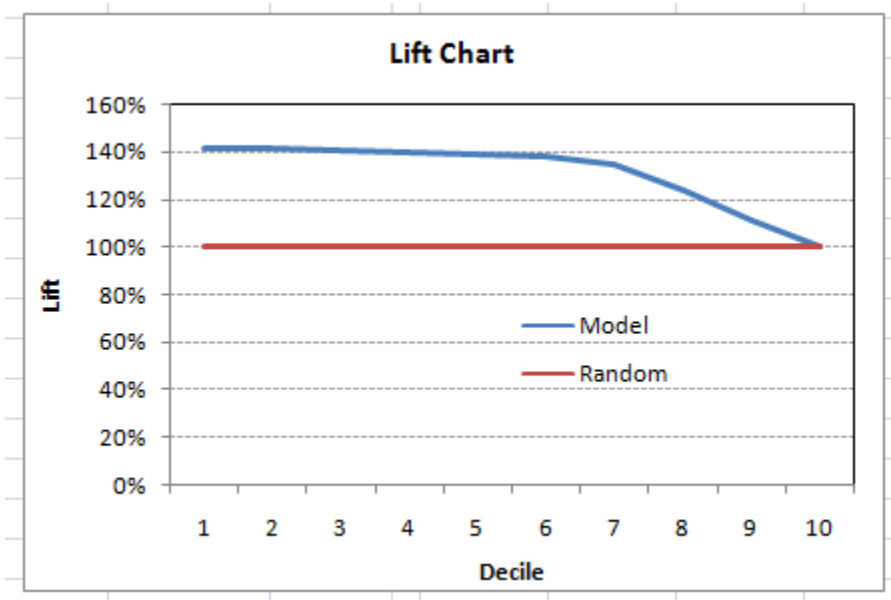


- AUC = Area Under Curve
- Overall measure of test performance
- Comparisons between two tests based on differences between (estimated) AUC

the higher the AUC, the better is the model.

Gain Chart and Lift Chart





Lift/Gain	Column Labels			%Rights	%Wrongs	%Population	Cum %Right	Cum %Pop	Lift @decile	Total Lift
Row Labels	0	1	Grand Total	0%	0%	0%	0%	0%		
1		543	543	14%	0%	10%	14%	10%	141%	141%
2	2	542	544	14%	0%	10%	28%	20%	141%	141%
3	7	537	544	14%	0%	10%	42%	30%	139%	141%
4	15	529	544	14%	1%	10%	56%	40%	137%	140%
5	20	524	544	14%	1%	10%	69%	50%	136%	139%
6	42	502	544	13%	3%	10%	83%	60%	130%	138%
7	104	440	544	11%	7%	10%	94%	70%	114%	134%
8	345	199	544	5%	22%	10%	99%	80%	52%	124%
9	515	29	544	1%	32%	10%	100%	90%	8%	111%
10	540	5	545	0%	34%	10%	100%	100%	1%	100%
Grand Total	1590	3850	5440							

Hyperparameter Tuning

- **Hyperparameter are parameters that are not directly learnt within estimators.**
- **For example, *C, kernel and gamma* for Support Vector machine. *Learning rate, dropout rate, batch size*, etc. for neural networks.**
- **Methods used to find out Hyperparameters**
  - *Manual Search*
  - *Grid Search*
  - *Random Search*
  - *Bayesian Optimization*
  - *Evolutionary Optimization*
  - …

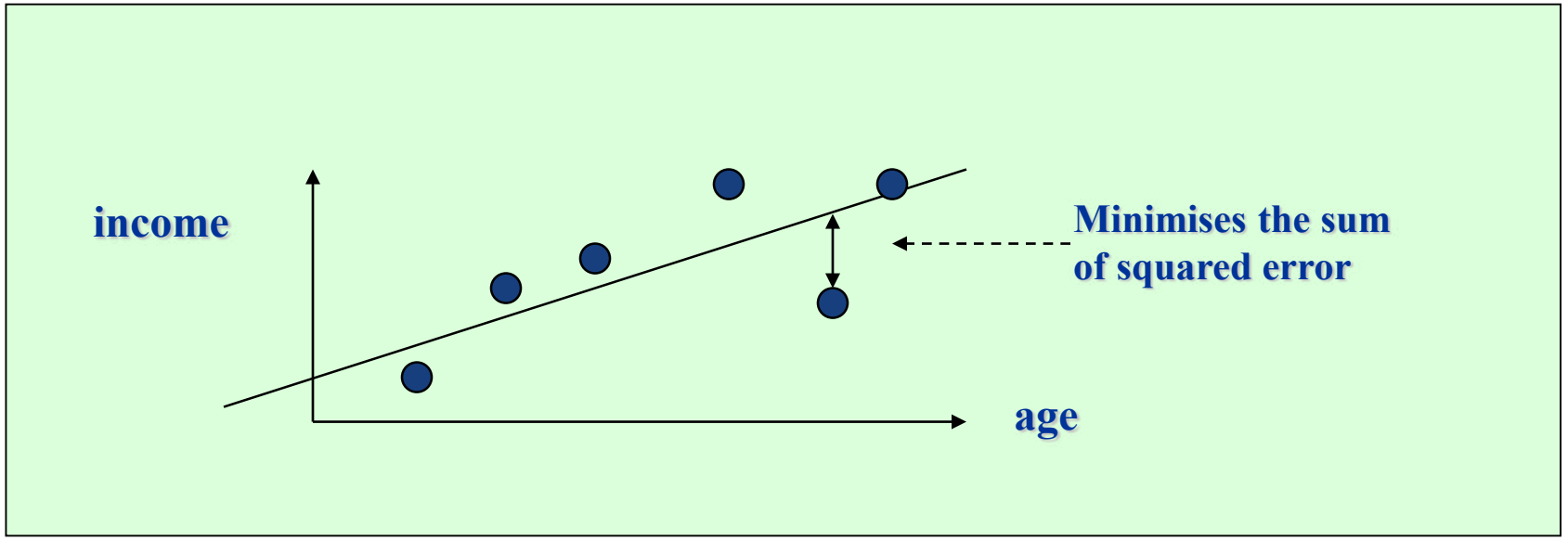
1.3
Solving Pattern Recognition Problems Using Supervised Learning Techniques (I)

Supervised Learning Techniques

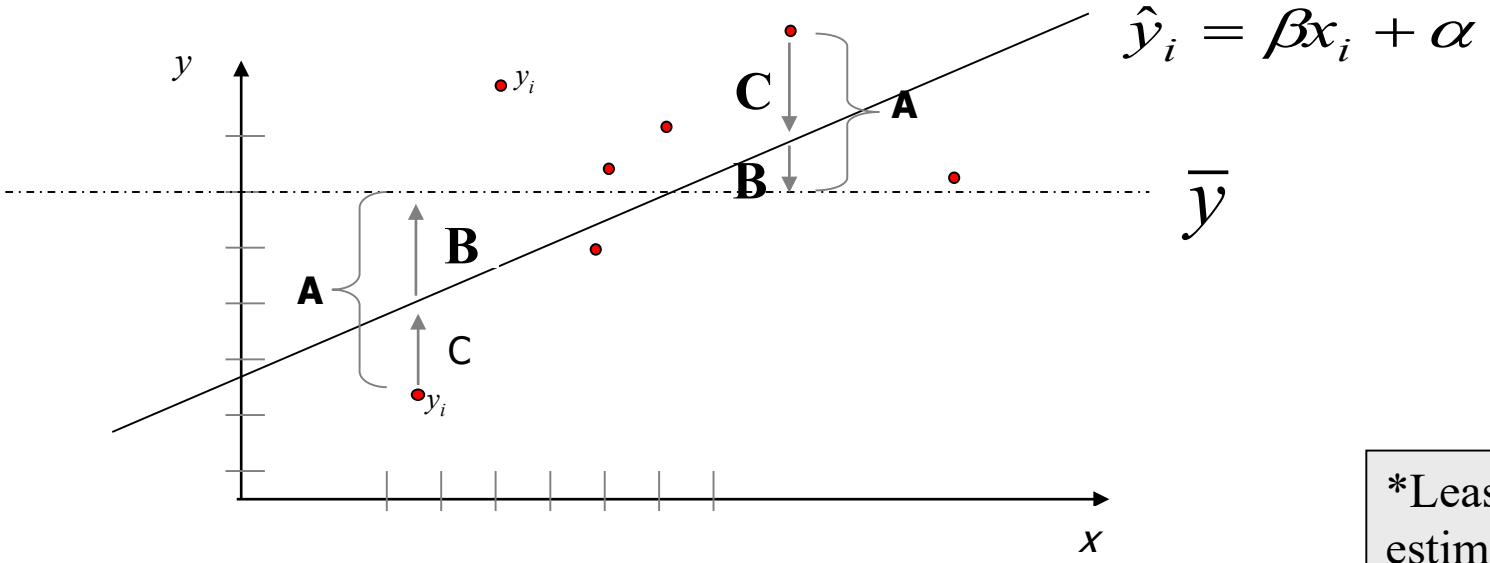
- Linear Regression & Logistic Regression
- Instance based Learning (K-NN)
- Naïve Bayes Classifiers
- Decision Trees
- Neural Networks
- SVM and Kernel Methods

Linear Regression

- Use for numeric targets
- Good if you know the target changes linearly
- Assumes the model: t = ax +by +cz + d etc.



Linear Regression



*Least squares estimation gave us the line (β) that minimized C^2

$$\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

A^2

SS_{total}

Total squared distance of observations from naïve mean of y

Total variation

B^2

SS_{reg}

Distance from regression line to naïve mean of y

Variability due to x (regression)

C^2

$SS_{residual}$

Variance around the regression line

Additional variability not explained by x—what least squares method aims to minimize

$R^2 = SS_{reg}/SS_{total}$

Logistic Regression

- Designed for classification problems
- Tries to estimate class probabilities directly

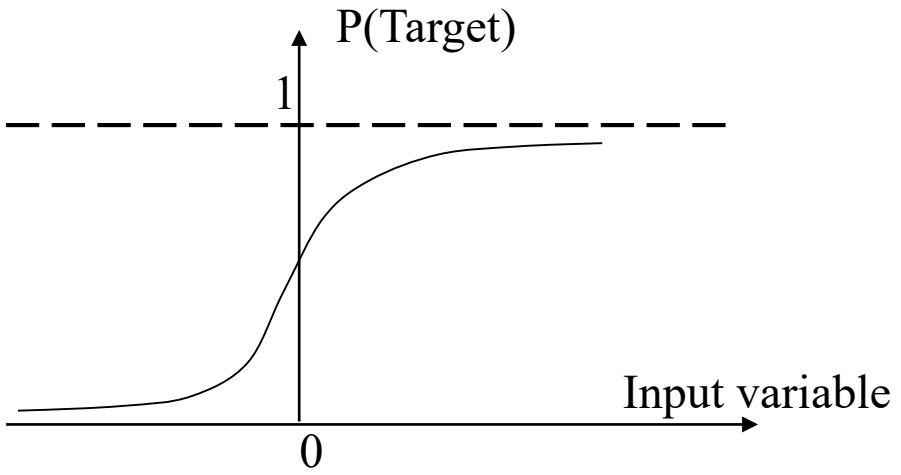
$$\ln\left(\frac{P}{1-P}\right) = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i$$

P= Class probability
P/(1-P) = odds
ln(p/(1-p) = logit (log odds)

Logistic Regression

- **Logistic Regression**

  For one input variable we can draw the logistic function as



Which is a good match for many T/F prediction situations

The transformation ln(p/1-p) turns this into a straight line (p = prob(target))

Multinomial Logistic Regression

- Generalizes logistic regression to multiclass problems.
- Predict the probabilities of the different possible outcomes

$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

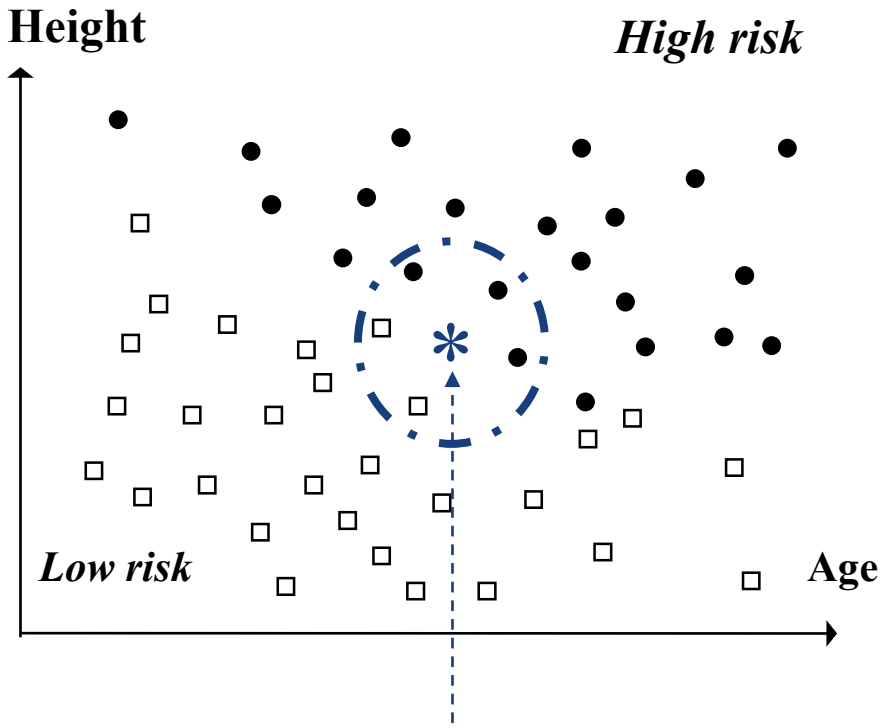
$$\Pr(Y_i = 2) = \frac{e^{\boldsymbol{\beta}_2 \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

$$\cdots\cdots$$

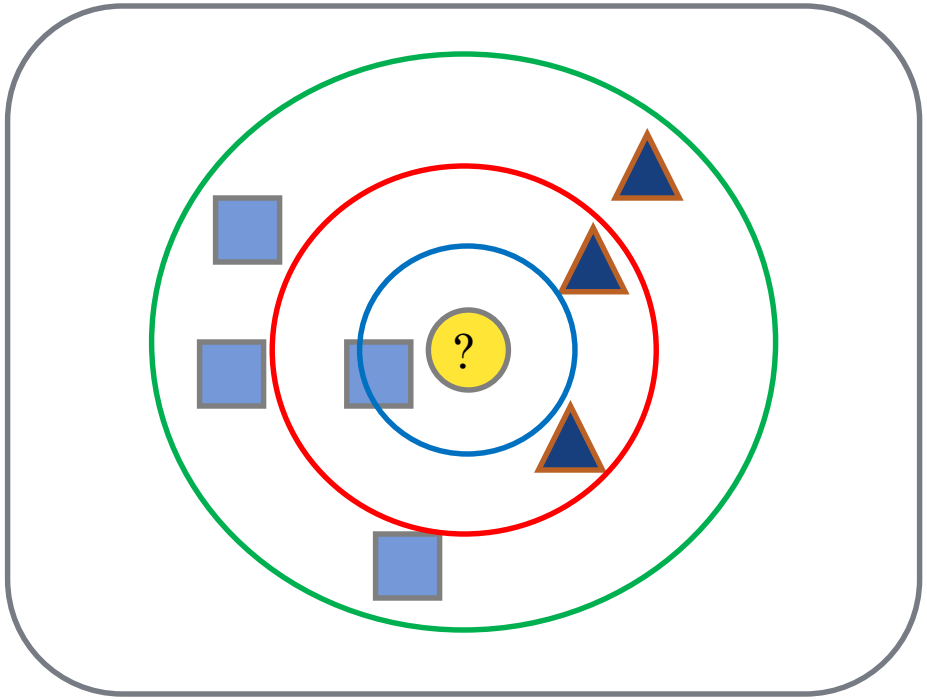
$$\Pr(Y_i = K) = \frac{e^{\boldsymbol{\beta}_K \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

K- Nearest Neighbour

- **Uses the "distances" between data items**
- **E.g. Assign a new pattern to the most represented class in the K nearest neighbours (e.g. K = 5)**
  - Non-linear decision surfaces
  - Can be computationally intensive
  - Distance measure is important



K- Nearest Neighbour



- Requires 3 things:
  - Feature Space(Training Data)
  - Distance metric
    - to compute distance between records
  - The value of *k*
    - the number of nearest neighbors to retrieve from which to get majority class

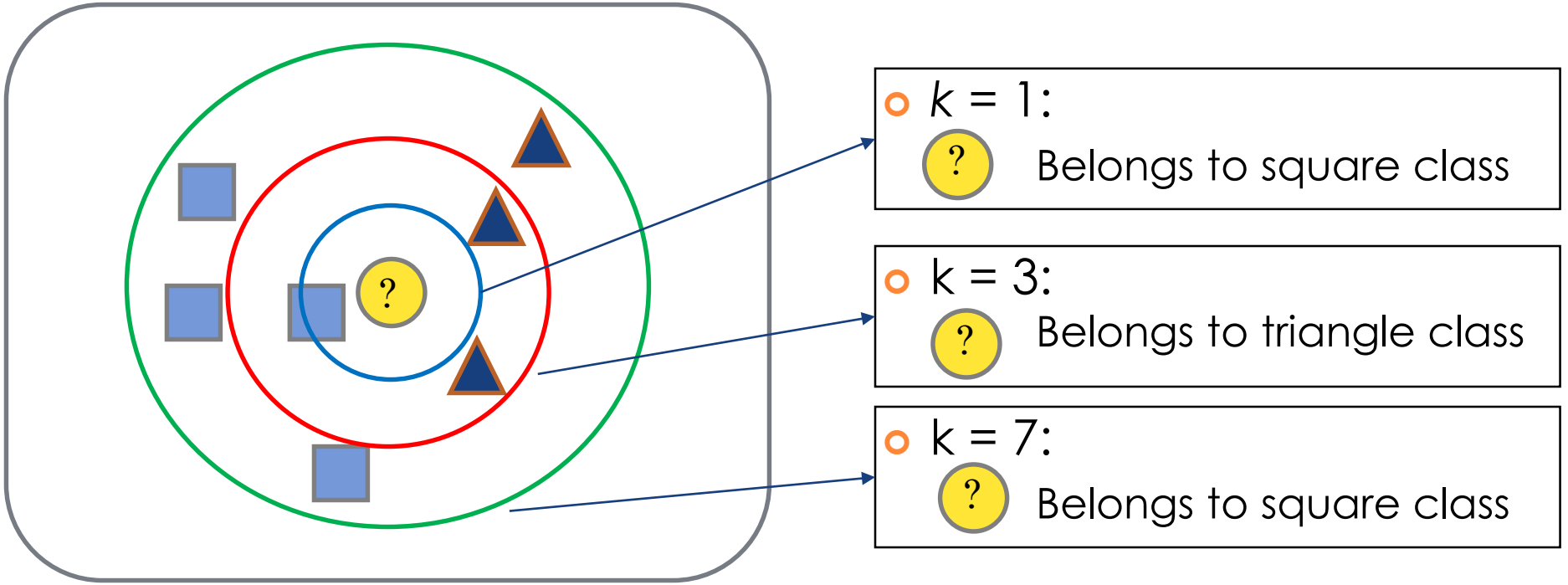
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record

K- Nearest Neighbour

- **Common Distance Metrics:**
  - Euclidean distance
  - Hamming distance
- **Determine the class from *k* nearest neighbor list**
  - Take the majority vote of class labels among the k-nearest neighbors
  - Weighted factor

ICDM: Top Ten Data Mining Algorithms, k nearest neighbor classification, December 2006

K- Nearest Neighbour



- If *k* is too small, sensitive to noise points
- If *k* is too large, neighborhood may include points from other classes
- Choose an odd value for *k*, to eliminate ties

(Source: ICDM: Top Ten Data Mining Algorithms, k nearest neighbor classification,2006)

K- Nearest Neighbour Advantages

- **Simple technique that is easily implemented**
- **Building model is inexpensive**
- **Extremely flexible classification scheme**
- **Well suited for**
  - Multi-modal classes (classes of multiple forms)
  - Records with multiple class labels
- **Nearest Neighbor classifiers are lazy learners**
- **Scaling issues**
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes.

Naive Bayes

- Naive Bayes is a probabilistic machine learning algorithm based on the Bayes Theorem.
- It is used in a wide variety of classification tasks.
- Typical applications include filtering spam, classifying documents, sentiment prediction, recommendation systems, etc.

Bayesian Classification

- **It performs *probabilistic prediction, i.e.,* predicts class membership probabilities, based on Bayes' Theorem.**

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})}$$

  - Let **X** be a data sample: class label is unknown
  - Let H be a *hypothesis* that X belongs to class Ci
  - Classification is to determine P(H | **X**), (i.e., *posteriori probability):* the probability that the hypothesis holds given the observed data sample **X**
  - P(H) (*prior probability*): the initial probability
  - P(**X**): probability that sample data is observed
  - P(**X** | H) (*likelihood*): the probability of observing the sample **X**, given that the hypothesis holds

Bayesian Classification

- Suppose X = (x_1, x_2, …, x_n) and *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|X)$
- Using Bayes' theorem, maximize $P(C_i|X)$ is equivalent to maximize

$$\frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

$$P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Naïve Bayes Classifier

- A naïve Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naïve) independence assumptions.

- It assumes that attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times ... \times P(x_n|C_i)$$

Naïve Bayes Classifier Example

- **Fruit Prediction Problem**: predict if a given fruit is a 'Banana' or 'Orange' or 'Other' based on three features: long (0/1), sweet (0/1) and yellow (0/1).

Training Data:

Fruit	Long (x1)	Sweet (x2)	Yellow (x3)
Orange	0	1	0
Banana	1	0	1
Banana	1	1	1
Other	1	1	0
…	…	…	…

Source: https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/

Naïve Bayes Classifier Example

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	Total
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

Let's say you are given a fruit that is: Long (1), Sweet (1) and Yellow(1), can you predict what fruit it is ?

Naïve Bayes Classifier Example

- **Step 1: Compute the 'Prior' probabilities for each of the class of fruits.**

  - P(C=Banana) = 500 / 1000 = 0.50
  - P(C=Orange) = 300 / 1000 = 0.30
  - P(C=Other) = 200 / 1000 = 0.20

Naïve Bayes Classifier Example

- **Step 2: Compute the probability of evidence that goes in the denominator. (Optional)**

  - P(x1=Long) = 500 / 1000 = 0.50
  - P(x2=Sweet) = 650 / 1000 = 0.65
  - P(x3=Yellow) = 800 / 1000 = 0.80

  - This is an optional step because the denominator is the same for all the classes and so will not affect the probabilities.

Naïve Bayes Classifier Example

- **Step 3: Compute the probability of likelihood of evidences that goes in the numerator.**

Probability of Likelihood for Banana:

- P(x1=Long | C=Banana) = 400 / 500 = 0.80
- P(x2=Sweet | C=Banana) = 350 / 500 = 0.70
- P(x3=Yellow | C=Banana) = 450 / 500 = 0.90

So, the overall probability of Likelihood of evidence for Banana = 0.8 * 0.7 * 0.9 = 0.504

Naïve Bayes Classifier Example

- **Step 4: Substitute all the values into the Naive Bayes formula to get the probability for "banana".**

  - P(C=Banana | X1=Long, X2=Sweet and X3=Yellow)=

$$\frac{P(Long \mid Banana) * P(Sweet \mid Banana) * P(Yellow \mid Banana) * P(Banana)}{P(Long) * P(Sweet) * P(Yellow)}$$

$$= \frac{0.8 * 0.7 * 0.9 * 0.5}{P(evidence)} = 0.252/p(evidence)$$

Naïve Bayes Classifier Example

- **Step 5: Repeat Step 3 and Step 4 to get the probability for "Orange" and "Other".**
  - P(C=Orange | X1=Long, X2=Sweet and X3=Yellow)=

$$\frac{P(Long \mid Orange) * P(Sweet \mid Orange) * P(Yellow \mid Orange) * P(Orange)}{P(Long) * P(Sweet) * P(Yellow)}$$

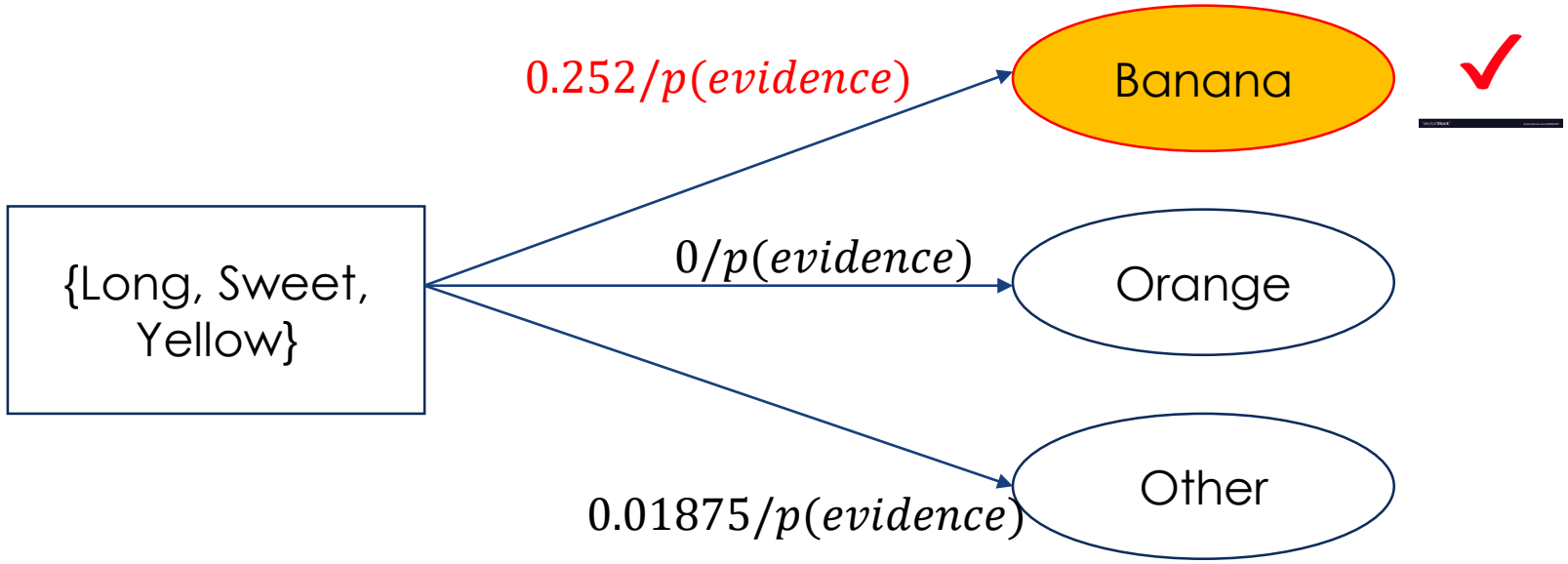
$$= \frac{\frac{0}{300} * \frac{150}{300} * \frac{300}{300} * \frac{300}{1000}}{P(evidence)} = 0/p(evidence)$$

  - P(C=Other | X1=Long, X2=Sweet and X3=Yellow)=

$$\frac{P(Long \mid Other) * P(Sweet \mid Other) * P(Yellow \mid Other) * P(Other)}{P(Long) * P(Sweet) * P(Yellow)} = \frac{\frac{100}{200} * \frac{150}{200} * \frac{50}{200} * \frac{200}{1000}}{P(evidence)} =$$

$$0.01875/p(evidence)$$

Naïve Bayes Classifier Example



- Answer: **Banana** – as it has the highest probability among the three classes.

Laplace Correction

- When having a model with many features, the entire probability will become zero because one of the feature's value was zero, as we have seen in the previous example.
- To avoid this, we increase the count of the variable with zero to a small value (usually 1) in the numerator, so that the overall probability doesn't become zero.

Pros and Cons of Naïve Bayes Classifier

- **Pros:**
  - Easy and fast to predict. Perform well in multi class prediction
  - When assumption of independence holds, a Naive Bayes classifier performs better than other models like logistic regression and needs less training data.
  - It perform well in case of categorical input variables compared to numerical variables. For numerical variable, normal distribution is assumed.
- **Cons:**
  - The assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

Modeling with Scikit-learn

```
import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.datasets import load_iris

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score,classification_report,
confusion_matrix


iris = load_iris()

X = iris.data

y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)


LR = LogisticRegression(random_state=0,solver='lbfgs',multi_class='multinomial')

LR.fit(X_train, y_train)
```

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X_train)

X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)

knn=KNeighborsClassifier(10)

knn.fit(X_train, y_train)


from sklearn.naive_bayes import GaussianNB

nb=GaussianNB()

nb.fit(X_train, y_train)


predictions = nb.predict(X_test)

print("Accuracy", accuracy_score(y_test, predictions))

print(confusion_matrix(y_test,predictions))

print(classification_report(y_test,predictions))
```

1.4
WORKSHOP 1
SOLVING PATTERN RECOGNITION PROBLEMS USING PYTHON AND SCIKIT-LEARN

Installation Instructions

- **Download and install latest Anaconda for Python 3.7: https://www.anaconda.com/download/**
- **Start Menu -> Anaconda Prompt:**
  - conda create -n psupr python=3.7
  - conda activate psupr
  - conda install numpy matplotlib jupyter pandas scikit-learn pydotplus graphviz
  - conda install -c conda-forge scikit-plot
  - Navigate to your working directory, eg. "d:\myfolder"
  - Run "jupyter notebook"
- Now you can open .ipynb files in the jupyter notebook within your browser
- If you need an IDE to edit and run a python program, install and run spyder in Anaconda Prompt.

Problem Description

Diabetes Prediction

- This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.
- The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *In Proceedings of the Symposium on Computer Applications and Medical Care* (pp. 261--265). IEEE Computer Society Press.

Workshop 1

- Open the iPython notebook provided for this workshop.
- As you go through the notebook, make sure you understand how each different model is built. (you can save notes as markdown in the notebook).
- Compare the performance of these models.
- Experiment with different parameter settings.
- You may try with your own datasets.
- Save your notebook with the cell output and upload it to LumiNUS.