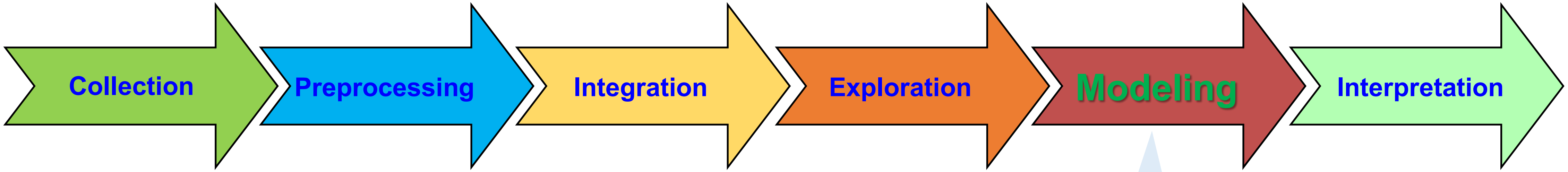
Data Science and Visualization (DSV, F23)

4. Classification (I)

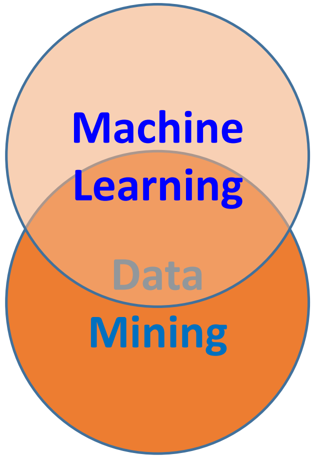
Hua Lu
https://luhua.ruc.dk; luhua@ruc.dk
PLIS, IMT, RUC

Next phase in this course



- **scikit-learn (sklearn)**: most prominent open source Python library for machine learning
  - https://scikit-learn.org/

- Classification
- Regression
- Clustering
- Association rules

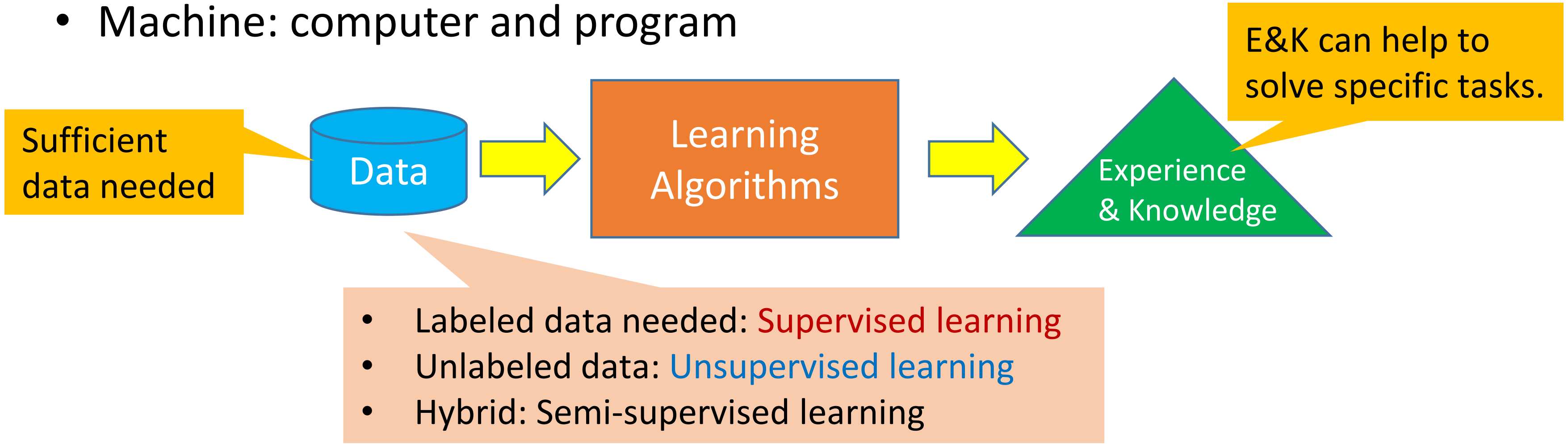


Agenda

- Introduction to machine learning
  - Supervised learning
  - Unsupervised learning
- Classification and model evaluation
- Typical classification models

What is Machine Learning (ML)?

- ML is about extracting knowledge from data. It involves statistics, artificial intelligence and computer science.
- Through means of computing, making use of experience to improve the performance of a target system.
    - Data -> Experience & Knowledge (often as models) -> Improved performance
    - Learning: Generating a E&K from the data
    - Machine: computer and program



Typical tasks for ML
- Regression
- Classification
- Clustering
- Association rules
- Dimensionality reduction

Supervised vs. Unsupervised Learning

- Supervised learning generalizes from *known examples* to automate decision-making processes.
  - **Classification**: Predict a discrete value from a *pre-defined* set of class labels
    - E.g., given a loan applicant, predict if she/he is a *good* or *bad* client. (*Approval* or *rejection*)
    - More examples: digital recognition from handwritings, fraud detection in banking, spam filtering.
  - **Regression**: Predict a continuous value from a continuous range
    - E.g., predict the price of a stock
- Unsupervised learning does *not* need any known examples. It works on input data directly. (*Future lectures*)
  - E.g., similarity based client grouping, outlier detection for website access patterns

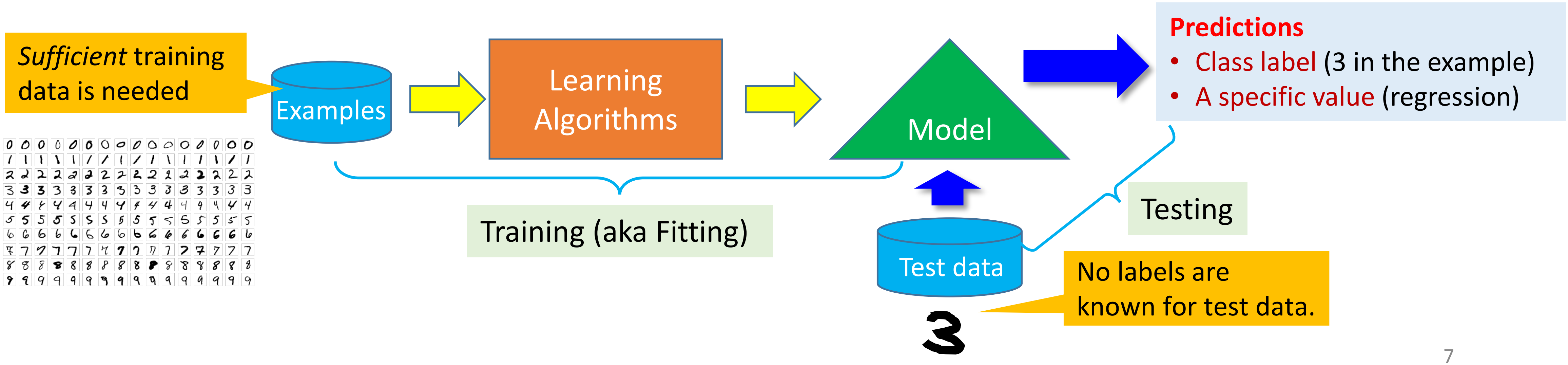
Supervised Learning: Training Data

- Known examples are called training data, aka labelled data.
- For classification, each data example consists of some *features* and a class label. Each class label should have at least one example. E.g.,
  - (*client's full record*, fraud *or* not) pairs for fraud detection
    - Features: original attributes or values derived from them
  - *Handwriting images* for 0, 1, …, 8, 9
    - Features: all pixels in an image
    - We know the number of each image
  - (*email*, spam *or* not) pairs for spam filtering
    - Features: words in an email.
- Feature selection and engineering
  - Which attributes to use?
  - How to derive features from attributes?

160 sample images from MNIST test dataset (wiki)

Supervised Learning Processes

- Data labeling: the process of identifying raw data (bank records, images, email, etc.) and adding a meaningful and informative label for each data sample
  - Often manual, tedious but necessary for machine learning
- Training: the process of using training data to generate a model.
  - We *fit* the model to the training data for which we know the labels or ‘Y’ values.
  - We hope the model can *generalize* to various unseen test data.
- Testing: Applying the model to new, unseen data (test data).



Generalization

- The goal of machine learning: Building from training data a model that can *generalize* to (unseen) test data.
- Overfitting: A model works well on the training data but generalizes poorly to unseen data.
  - Noises exist in the training/test data.
  - Training data is too little, failing to contain sufficient variance.
  - Too many features are used in training, while some of them don't exist in test data.
  - The model type used is too complex.
- Underfitting: A model even does not work well on the training data.
  - Too few features are used in training.
  - The model type used is too simple.

Example: Leaf Detection/Classification

- Training data (leaf samples)

- Test data (unseen)

Fix: Include more training samples.
E.g., those without sawtooth.

An *overfitting* model might say ”Not a leaf”.
- The training data samples all have sawtooth
- The model thinks a leaf must have sawtooth.

Fix: Use more features or a more complex model.

An *underfitting* model might say ”A leaf”.
- Only color is used as the feature.
- The model thinks everything green is a leaf.

Fix: Include more training samples.
E.g., those in other colors.

An *overfitting* model might say ”Not a leaf”.
- The training data samples are all green.
- The model thinks a leaf must be green.

Leaf images from https://www.dreamstime.com/

Agenda

- Introduction to machine learning
- Classification and model evaluation
  - Classification steps
  - Classification performance
- Typical classification models

Classification: Three Major Steps

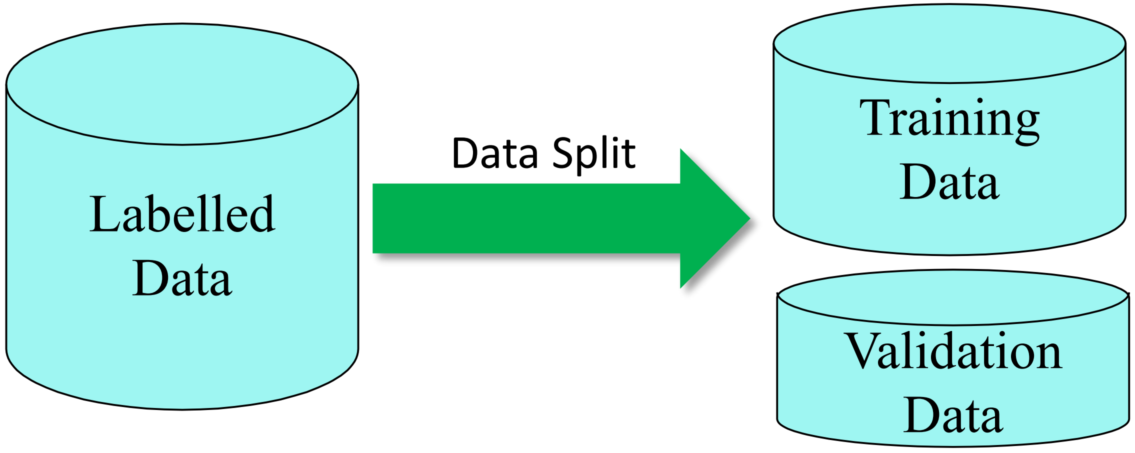
1. **Model construction**: describing a set of *predetermined* classes
   - Each tuple/sample is assumed to belong to a predefined class, as determined by the **class label** column
   - The set of tuples used for model construction is **training set**
   - A model is created using an algorithm on selected features.
     - Simply speaking, features are columns or generated based on columns.
     - Not all columns are used for creating a model.
     - **Feature selection** or **engineering** decides which features (columns) to be used.
   - The model is represented as classification rules, a decision tree, or mathematical formulae.
   - The model can predict the class label for a given (unseen) tuple
2. **Model validation**
3. **Model application/test**

Classification: Three Steps (cont.)

2. **Model validation**: to *evaluate* how good your model is for the given validation data set; to tune the parameters of a model (*parameter tuning*).
   - **Estimate accuracy** of the model using **validation set** (data set for validation)
     - The known label of test sample is compared with the classified result from the model
     - **Accuracy** rate is the percentage of test set samples that are correctly classified by the model
     - **Validation set** should be independent of training set (otherwise **overfitting**)
   - If the accuracy is *acceptable*, the model can be used to *classify new/unseen data* (model application/test)
   - Different models may be compared for selection of the best
   - *NB*: Sometimes validation is also called test (e.g., in sklearn)
3. **Model application/test**: for classifying future or *unseen* objects
   - For those objects, you don't know their classes!

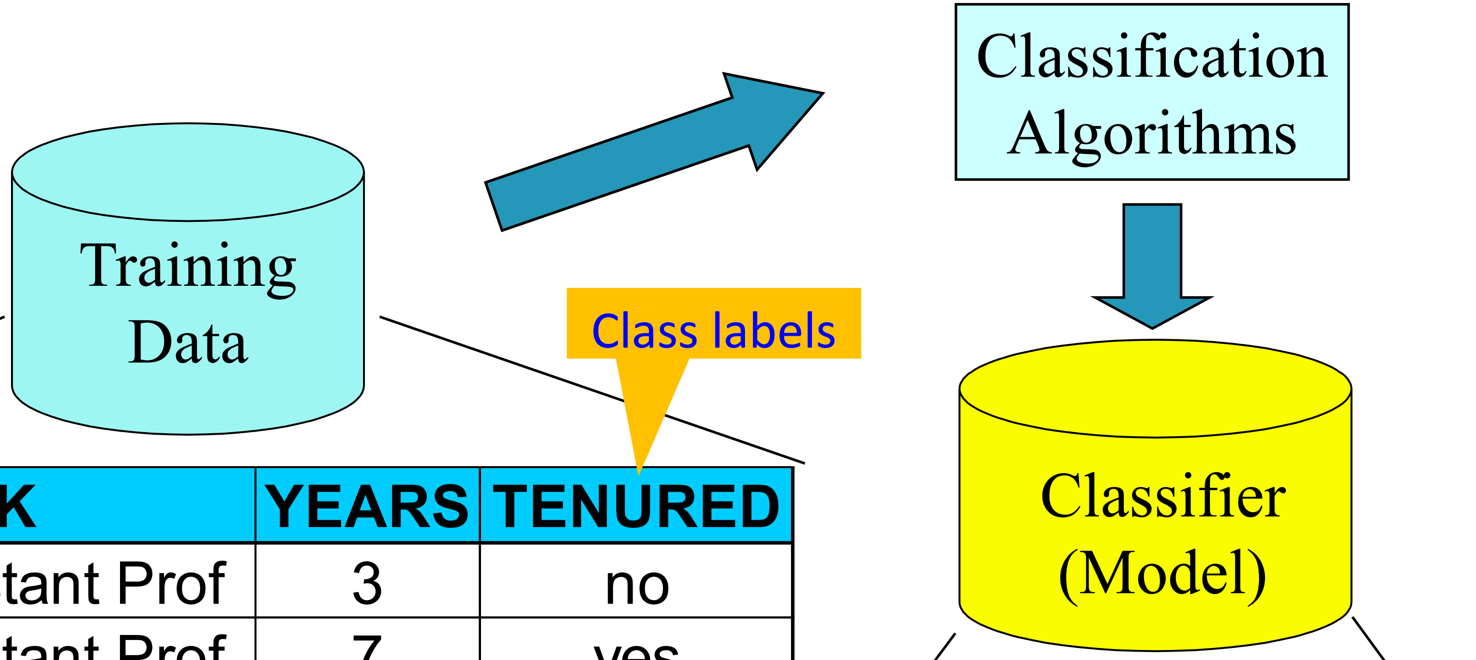
Split Labelled Data

- sklearn.model_selection.train_test_split
- X_train, X_test, y_train, y_test = **train_test_split**(X, y, test_size=0.30, random_state=42)
  - X_train: features of training data; y_train: class labels of training data
  - X_test: features of validation data; y_test: class labels of validation data
  - test_size: percentage of validation data
  - random_state: randomization of the split. A fixed number will enable reproducibility.
- Different ways of split can result in different models and performance
  - We will see more next week



Step 1: Model Construction

- Two classes
  - Tenured
  - Not tenured

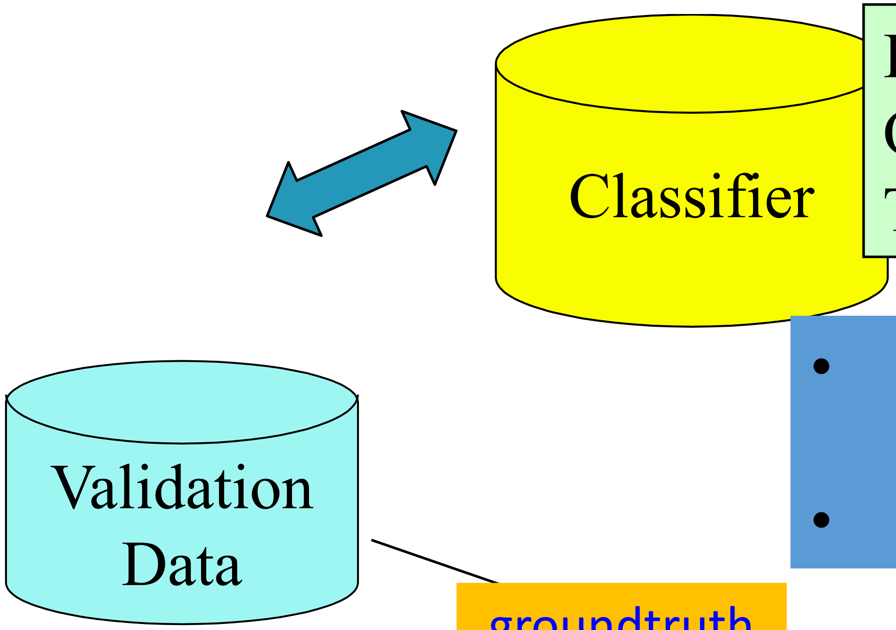


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = 'professor'
OR years > 6
THEN tenured = 'yes'

Step 2: Model Validation

- Two classes
  - Tenured
  - Not tenured



IF rank = 'professor'
OR years > 6
THEN tenured = 'yes'

- How good is the results on the validation data?
- Evaluate your model

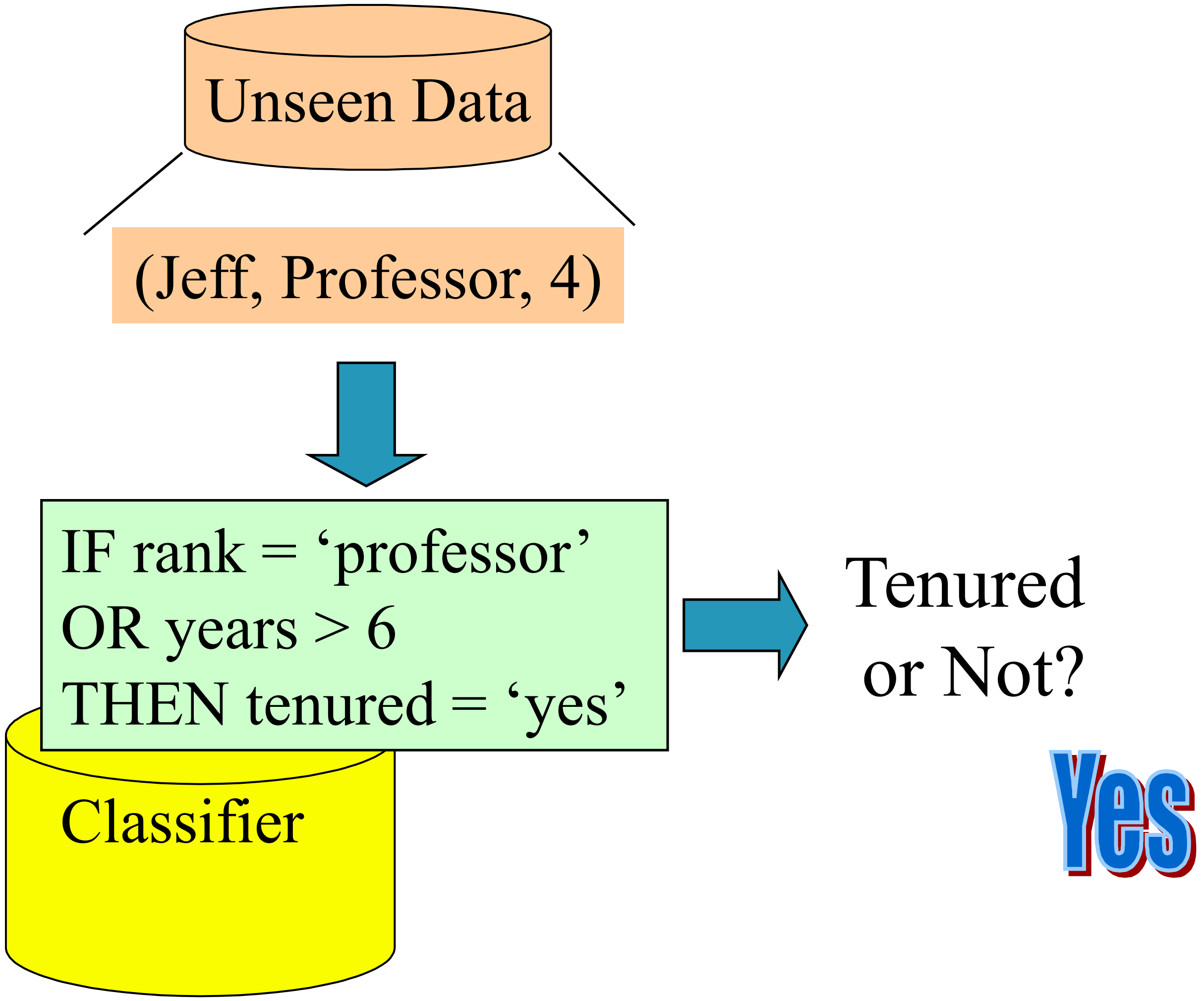
groundtruth

NAME	RANK	YEARS	TENURED	predicted
Tom	Assistant Prof	2	no	no
Merlisa	Associate Prof	7	no	yes
George	Professor	5	yes	yes
Joseph	Assistant Prof	7	yes	yes

Model accuracy is 75%

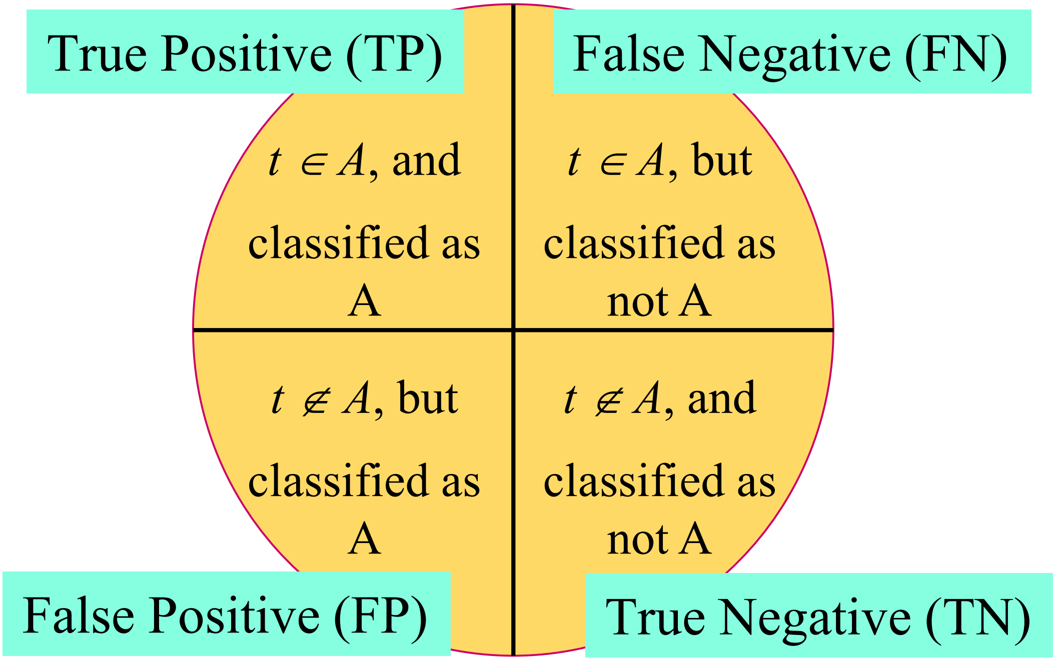
Step 3: Model Application/Test

- Two classes
  - Tenured
  - Not tenured



Classifier Performance

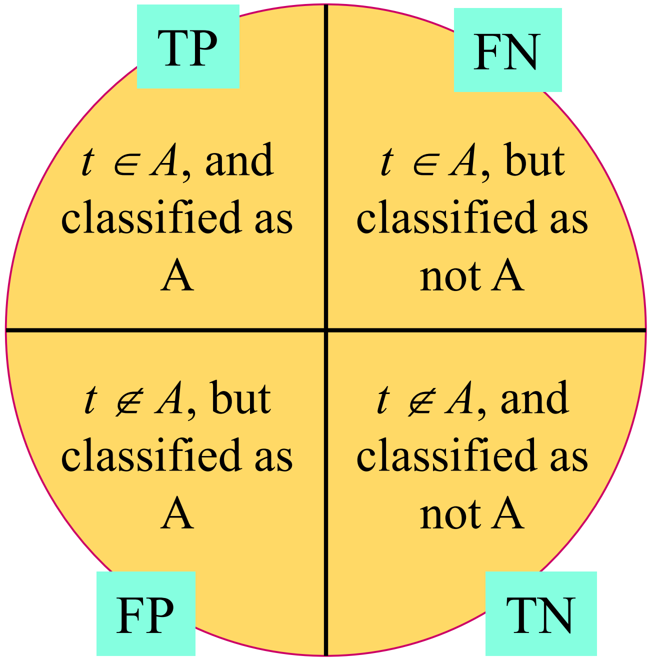
- Consider a binary classifier
  - Target class A: *positive*
  - Target class NOT A: *negative*
- Consider the *ground truth* and classification result. There are four cases.
  - Put it simply, ground truth is the ‘fact’ we know.



Performance Metrics (for One Class A)

- **Precision** (exactness)
  - How often the positive classification is correct.
  - TP / (TP+FP)
- **Recall** (completeness)
  - How many of the actual positive cases are classified as positive.
  - TP / (TP+FN)
- **Accuracy**
  - The fraction of the time when the classifier gives the correct classification.
  - (TP+TN) / (TP+FP+TN+FN)

This is easy to understand for *binary classification*.



Precision, Recall and F-measures

- Perfect score for Precision and Recall is 1
- Inverse relationship exists between Precision and Recall
- ***F* measure** (*F_1* or ***F*-score**): harmonic mean of Precision and Recall

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

- ***F_β*:** weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

What if more than two class labels?

Confusion Matrix

- Three pre-defined classes A, B, C
- Ground truth and classification result

Object	Ground Truth	Classification Result
object-1	*A*	A
object-2	*B*	A
object-3	*C*	C
object-4	*C*	C
object-5	*B*	B
object-6	*A*	B
object-7	*B*	B

- Confusion Matrix (N x N for N classes)
  - The cell M[*i*, *j*] counts the case that groundtruth *i* is classified as *j*

Classification result

FN for A

Ground truth

	A	B	C	Total
A	1	1	0	2
B	1	2	0	3
C	0	0	2	2
Total	2	3	2	7

FP for A

TP for one class, and TN for others

- **Accuracy** = (TP+TN) /All
  - In this example, Accuracy = (1+2+2)/7 = 71.4%

Precision/Recall in Confusion Matrix

- Calculate the precision p_i for each class C_i
  - Overall precision is the *average* of all p_i s
- Calculate the recall r_i for each class C_i
  - Overall recall is the *average* of all r_i s
- Example
  - p_A=30/60=1/2, r_A=30/100=3/10
  - p_B=60/120=1/2, r_B=60/100=3/5
  - p_C=80/120=2/3, r_C=80/100=4/5
  - Overall precision = 5/9,
    overall recall = 17/30

Confusion matrix

Classification result (columns); Ground truth (rows)

	A	B	C	Total
A	30	50	20	100
B	20	60	20	100
C	10	10	80	100
Total	60	120	120	300

Recall for A: r_A

Precision for A: p_A

Analyze Your Confusion Matrix

- Essentially, the more zeroes or smaller the numbers on all cells but the diagonal, the better a classifier is. So you may analyze your confusion matrix and tweak your features accordingly.
- Confusion matrix gives strong clues as to where your classifier is going wrong.
  - E.g., if for Class A you can see that the classifier incorrectly predicts Class B for majority of the mislabeled cases, it indicates the classifier is somehow confused between classes A and B.
  - One way to fix this is to add biasing features to improve classification of class A, e.g., more training data of A.

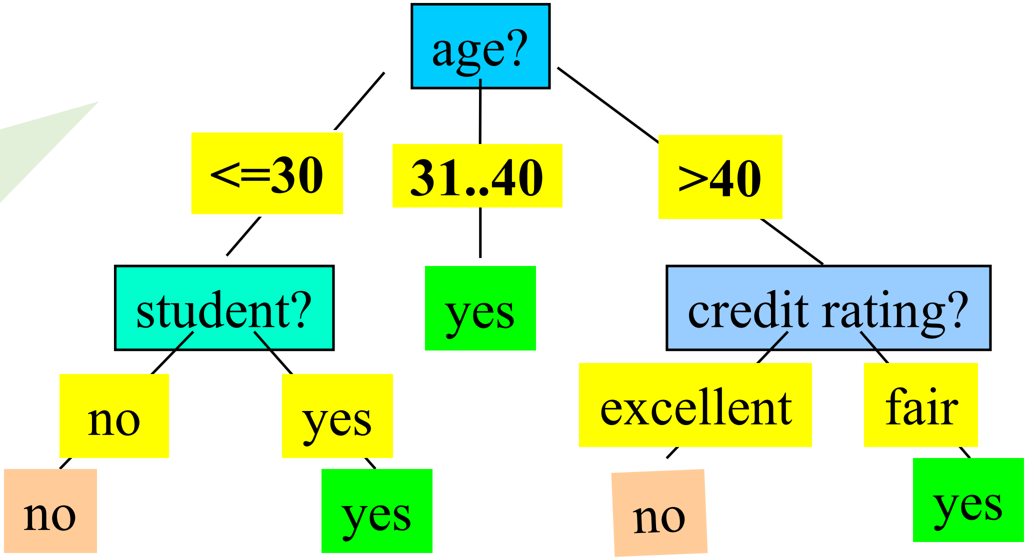
Agenda

- Introduction to machine learning
- Classification and model evaluation
- Typical classification models
  - Rule based classifier (the previous tenure example)
  - Decision tree
  - Random forest
  - K nearest neighbors (KNN, next week)

Classification Using Decision Trees

- The classification model (classifier) is organized as a tree for decision making.
  - It’s thus called a **decision tree**.
- Internal nodes are associated with an *attribute/column* and arcs with *values* for that attribute.
- A leaf node tells the predicted class label.

- Each person has attributes/columns:
  - (age, student [yes/no], credit rating)
  - p1(18, yes, fair)
  - p2(55, no, excellent)
- Two classe labels
  - Buy computer
  - Not buy computer



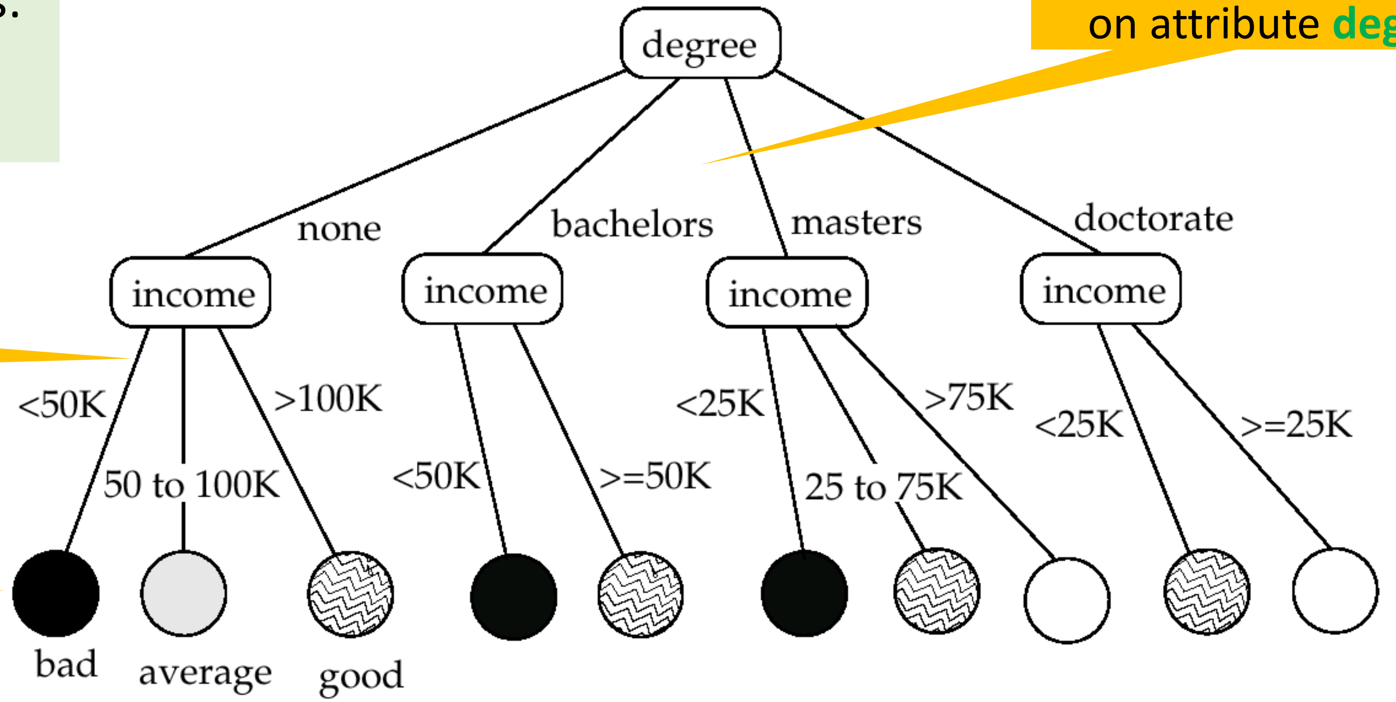
Another Decision Tree Example

- Each client has attributes/columns:
  - (degree, income, …)
  - E.g., c1(bachelor, 55K, …)

Partitioning (or splitting) on attribute **degree**

Partitioning on attribute **income**

A leaf node predicts a **class label**



- Four class labels (credit levels)

Example in Jupyter Notebook

- Diabetes dataset
  - 768 data objects of 9 columns/attributes
  - Available in Moodle
- Decision tree for classification (2 classes)
  - 1: Positive of diabetes
  - 0: Negative
- Lecture4_diabetes.ipynb

Construction/Optimization of Decision Trees

Advanced

- Input:
  - Training data: a set of data in which the classes are already known.
- Ouput:
  - A decision tree that can be used for classification.
- Basic idea:
  - Generating a decision tree *top-down* using the training data.
  - Each internal node of the tree partitions the training data into groups based on a **splitting attribute** and a **splitting condition** for the node
    - Measure of the quality of a split: **gini** index and **entropy**.
    - **Best** split vs. **Random** split (in sklearn)
  - In a leaf node, all the items at the node belong to the same class, or all attributes have been considered and no further partitioning is possible.

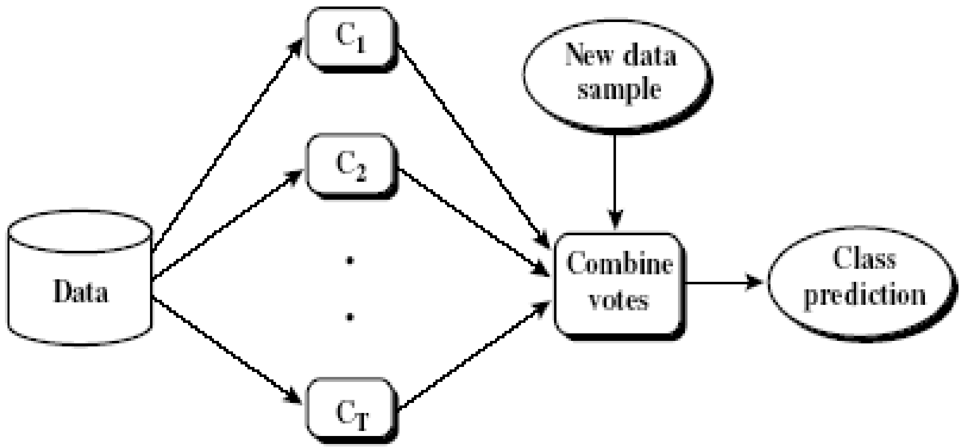
NB: You can specify how a DT is constructed.

Random Forest (of Decision Trees)

- Decision trees' main drawback: tendency to **overfit** the training data.
  - Overfitting: A model focuses so much on training data that it does not generalize well to unseen data in predication.
- A random forest
  - A collection of decision trees (DT)
  - Each DT is slightly different from the others
  - With many DTs, we can maintain good prediction and reduce overfitting by using all trees' majority vote as the classification result.

Random Forest, *cont.*

- Two ways of introducing randomness to the tree building
  - Randomly selecting the training data points
  - Randomly selecting the features in each tree node split
- Random forest is an example of ensemble learning
  - Use a combination of models to increase prediction accuracy
  - Combine a series of T learned models/classifiers, C_1, C_2, …, C_T, with the aim of creating an improved model C*
    - A simple combination: majority vote



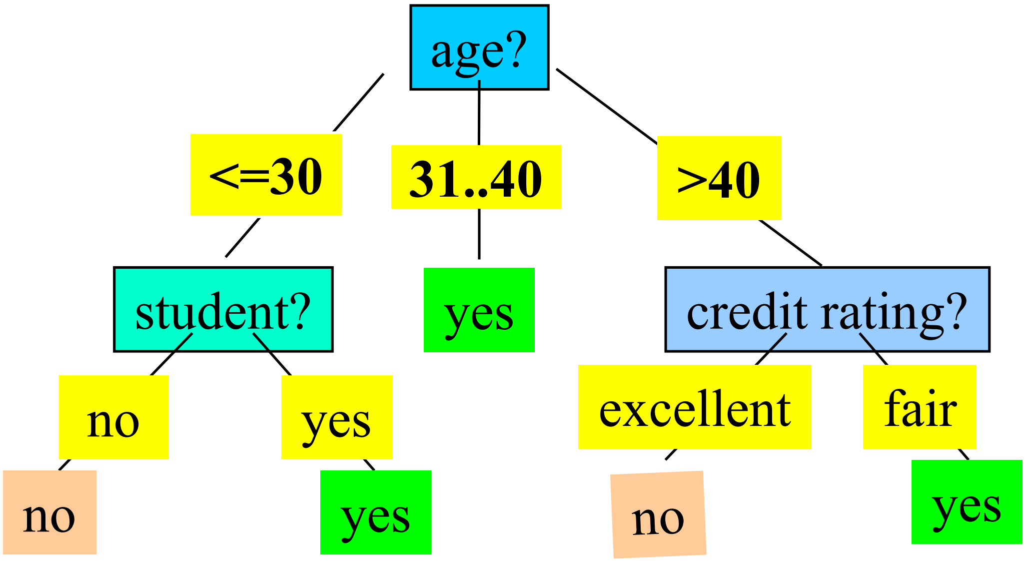
Continued Example in Jupyter Notebook

- Diabetes dataset
  - 768 data objects of 9 columns/attributes
  - Available in Moodle
- On the same data set, we can construct many different decision trees.
- We can also construct a random forest.

- Lecture4_diabetes.ipynb
  - Optimizing decision trees
  - Random forest

Comments on Decision Trees

- Given the same training data set, different decision trees may be constructed by different methods (with different parameters)
- A decision tree may be unbalanced
- At the same level, different nodes in a decision tree may split on different attributes.
- Decision trees belong to *eager learners*



Eager *vs* Lazy Learning

- **Eager learning** (model based methods): Given a set of training samples, constructs a classification model before receiving new (e.g., test) data to classify.
  - More time in training but less time in predicting/classification
  - E.g., we need to construct a decision tree before using it.
- **Lazy learning** (e.g., instance-based learning): Simply stores training data as instances (or only minor processing) and waits until a new instance must be classified
  - Less time in training but more time in predicting/classification
  - E.g., k nearest neighbors: Instances represented as points in a metric space (e.g., Euclidean space).

Summary

- Supervised learning
  - Model construction/training, validation, test
  - Labelled data splitting
- Classification performance evaluation
  - Precision, recall, accuracy
  - Confusion matrix
- Classification models
  - Decision tree
  - Random forest

References

- Mandatory reading
  - Muller and Guido: Introduction to Machine Learning with Python, O'Reilly, 2016
    - Chapter 1
      - A First Application: Classifying Iris Species
    - Chapter 2
      - Decision Trees, Ensembles of Decision Trees
  - You may refer to Lecture-4/Lecture4_iris_csv.ipynb in Moodle
- Further reading
  - Decision tree
    - **Tutorial**: https://www.datacamp.com/community/tutorials/decision-tree-classification-python
    - **Doc**: https://scikit-learn.org/stable/modules/enerated/sklearn.tree.DecisionTreeClassifier.html
  - Random forest
    - **Tutorial**: https://www.datacamp.com/community/tutorials/random-forests-classifier-python
    - **Doc**: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Exercises

Using the Titanic dataset (available in Moodle), do the following in Lecture4-Exercise_Titanic_template.ipynb (available in Moodle)

1. Obtain a reduced dataset *D* that only contains the following features
   - Survived, Pclass, Sex and Age
   - *NB*: Data preprocessing is needed, e.g., transform to numerals, imputing missing values (NA)
2. Split the dataset *D* into two parts: 80% for training (D_T) and 20% for validation/test D_V.
   - *NB*: You may use different ratios, do the subsequent steps, and see the effect
3. Build a decision tree for predicting if a passenger can survive. Use D_T to train the model, apply the model to D_V, and evaluate the classification accuracy.
   - Try to build a number of different trees using different parameters, see their accuracy
4. Build a random forest on the same training/test datasets, obtain its accuracy, and plot the important features for it.