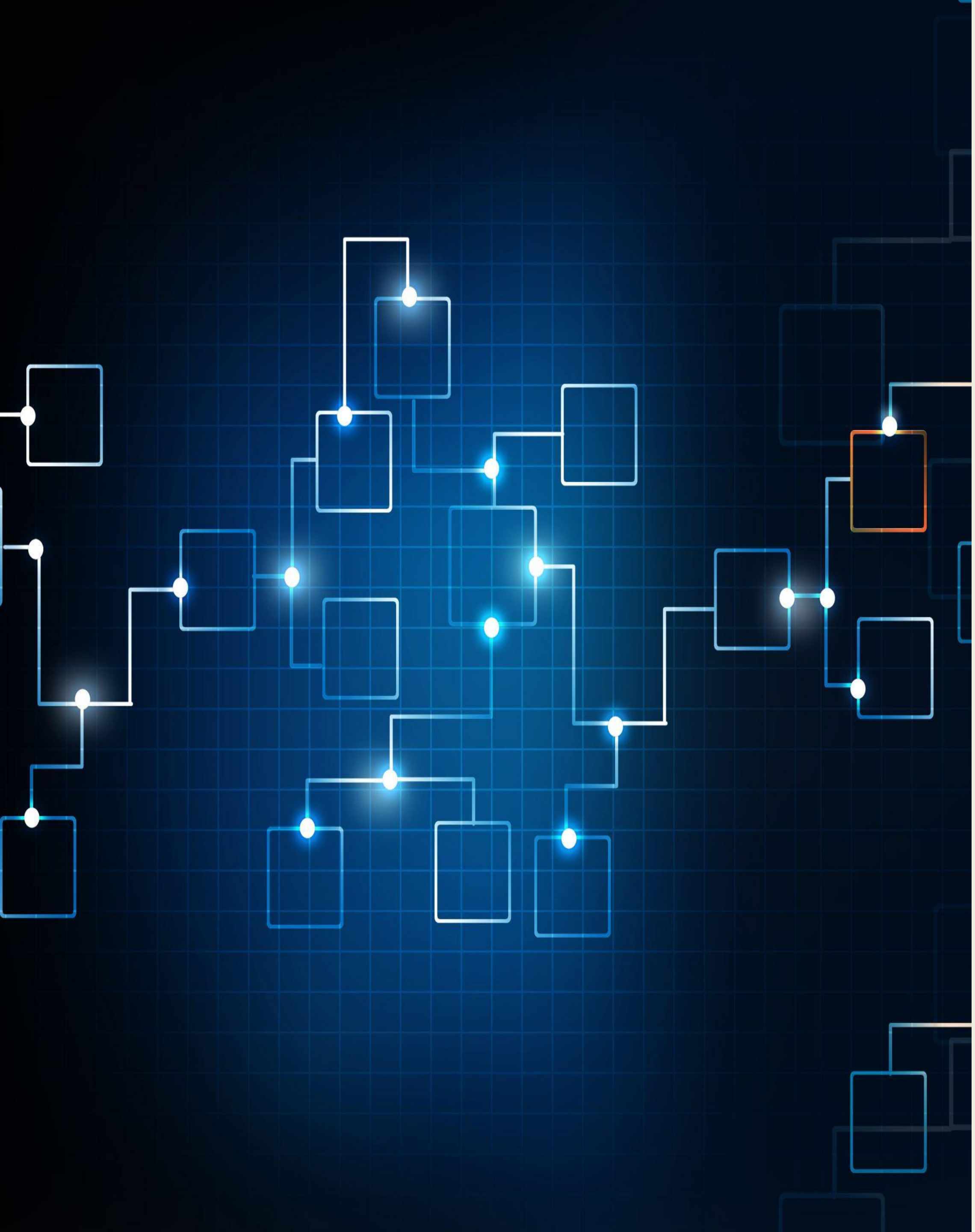


HIT391

MACHINE LEARNING: ADVANCEMENTS AND APPLICATIONS

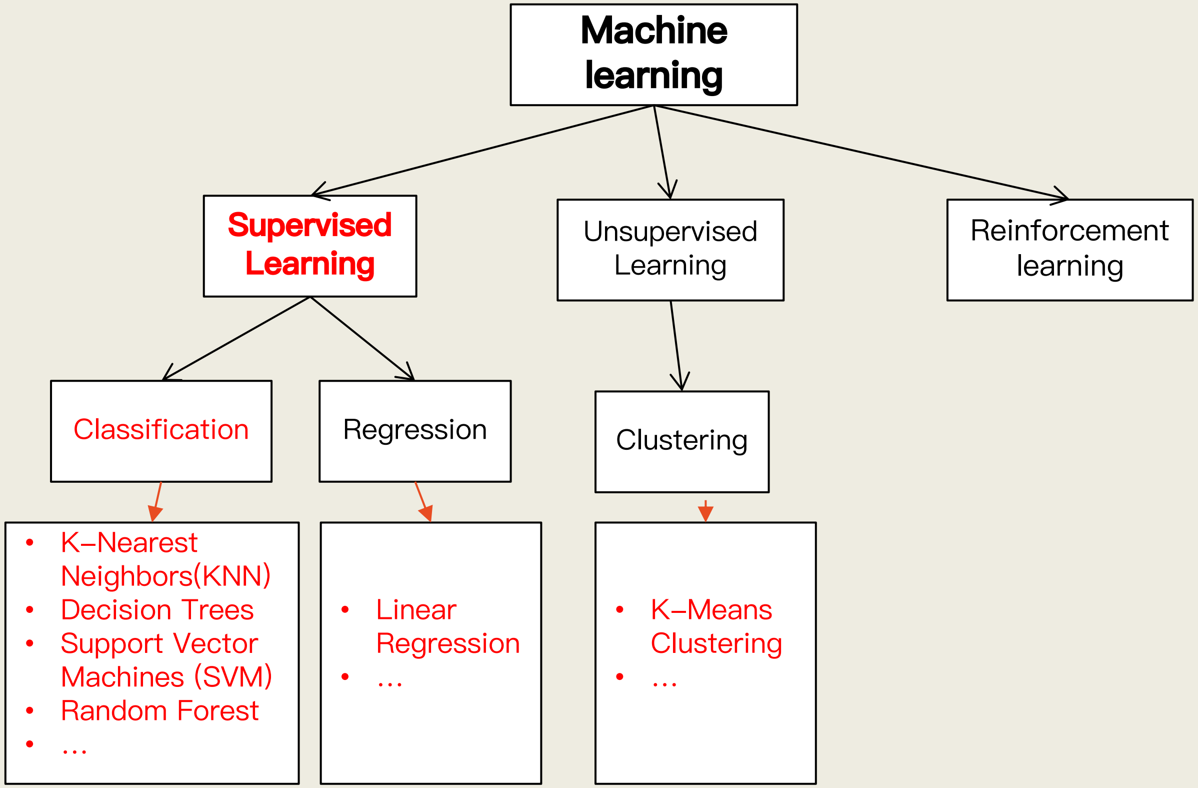
Week 4:
Supervised Learning – Classification

- Learning Outcomes
  - Supervised Learning
    - Classification Methods

      1. K-Nearest Neighbors (KNN)

      2. Decision Trees

Machine Learning



Supervised Learning

- In supervised learning, the algorithms are presented with a set of labeled data from which they learn a way of <u>predicting labels of unseen instances</u>.
- Classification vs Regression

	Attributes (Input)	Labels (Output)
Classification	Any types	Discrete Numeric / Categorical
Regression	Numeric	Continuous

Supervised Learning Procedure

- 1. Data Preprocessing
- 2. Data spliting – Training set / Testing set
- 3. Training a model (classification / regression)
- 4. Testing the performance of trained model.

Training phase

- ➢ Learn / Train a model (classifier) from the available data.
- Using ‘Training set’ (known labels)

Testing phase

prediction

- ➢ Testing how well the classifier Performs.
- Using ‘Testing set’ (un-known labels)

  - Predict the labels

Data Splitting – Creating Training/Testing Sets

- Methods:
  - Holdout (2/3rd training, 1/3rd testing)
  - Cross validation (n – fold), e.g., Leave-one-out
    - Divide into n parts
    - Train on (n-1), test on last
    - Repeat for different combinations
  - Bootstrapping
    - Select random samples to form the training set

Classification

- Training data, each case:
  - Set of attribute (feature) values - “independent variables”
  - Numerical / Categorical output value - “labels”
- Model is function from features to output
  - Use model to predict output value for new features
- Example 1
  - **Features**: age, gender, income, profession
  - **Label:** buyer, non-buyer

Example 2: Medical Diagnosis

- Objective: Predict a patient's disease based on various factors.
- Features (Input Data):
  - Age
  - Gender
  - Medical history
  - Symptom 1 (severity level)
  - Symptom 2 (severity level)
  - Test result 1
  - Test result 2
- Label (Target Output): Disease diagnosis

Example 3: Credit Card Fraud Detection

- Objective: Determine whether a credit card transaction is fraudulent.
- Features (Input Data):
  - User ID
  - Location of transaction
  - Purchased item
  - Price of the transaction
- Label (Target Output): Fraudulent (fraud) or legitimate (okay) transaction

Explanations

- In both cases, **features** (input variables) are used to predict **labels** (output categories).
- A machine learning model learns patterns **from past labeled data** to make **predictions on new cases**.
- **Feature Engineering**: Choosing the right features is crucial for accurate predictions.
  - Example: In medical diagnosis, symptom severity is an important predictor.
  - Example: In fraud detection, the location of a transaction can indicate suspicious activity.
- **Model Training & Prediction**: The model is trained on historical data where the disease or fraud status is known; After training, it can predict the outcome for new, unseen data.
- **Real-World Impact:**
  - Medical diagnosis: Helps doctors make better decisions and detect diseases early.
  - Fraud detection: Prevents financial losses by identifying suspicious transactions in real time.

Algorithms for Classification

- Although classification problems may seem similar to regression, their **non-numerical (categorical)** nature requires entirely different approaches. Instead of predicting **continuous values**, <u>**classification models assign inputs to discrete categories or classes.**</u>
- K-nearest neighbors
  - Decision trees
  - Naïve Bayes
  - Support Vector Machine

K–Nearest Neighbours(KNN)

- What is K-NN?
  - K-Nearest Neighbors (K-NN) is a classification algorithm that predicts a data point's category based on its K closest neighbors.
- Key idea: Similar data points are closer in feature space, while different data points are farther apart.
- Goal: Classify a new data point
  - We measure its distance to existing labeled points and assign it as **the majority class** among its **K nearest neighbors**.

Example

- We want to **classify** people based on certain features:

Feature	Person 1	Person 2
Gender	Male	Female
Profession	Teacher	Teacher
Age	47	43
Income	$25K	$28K
Postal Code	94305	94309

Goal: Compute distance(Person1, Person2) to determine **how similar** they are.

Distance is the inverse of **similarity** → Smaller distance = More similar

Computing Distance in K–NN

- Measuring Distance Between Data Points i_1 and i_2, from their feature values compute distance $d(i_1,i_2)$
- To classify a new person, we compute how "close" they are to existing people using distance functions.

1. Euclidean distance (for numeric data)

$$d_{num} = \sqrt{(Age_1 - Age_2)^2 + (Income_1 - Income_2)^2 + (PostalCode_1 - PostalCode_2)^2}$$

$$d_{num} = \sqrt{(47-43)^2 + (25000-28000)^2 + (94305-94309)^2}$$

$$= \sqrt{16 + 9000000 + 16} = \sqrt{9000032} \approx 3000.05$$

Distance Functions

2. Hamming distance (for categorical data)
    - If values match, distance = 0 (same category).
    - If values don't match, distance = 1 (different category).

Feature	Person 1	Person 2	Distance
Gender	Male	Female	1
Profession	Teacher	Teacher	0

3. Manhattan distance
4. Minkowski distance

- ❑ Combine Numerical/Categorical/other Distances

K–Nearest Neighbors (KNN)

- **Features** - gender, profession, age, income, postal-code

  label

  - person1 = (male, teacher, 47, $25K, 94305), buyer
  - person2 = (female, teacher, 43, $28K, 94309), non-buyer
- Remember training data has labels
- Objective:
  - To classify a new item i: In the labeled data find the K closest items to i, assign most frequent label
  - person3 = (female, doctor, 40, $40K, 95123), ?

  Label prediction

KNN Summary

To classify a new item i: find K closest items to i in the labeled data, assign it as the majority class of these K items

- No hidden complicated math!
- Once distance functions are defined, rest is easy
- Though not necessarily efficient
  - Real examples often have thousands of features
    - Medical diagnosis: symptoms (yes/no), test results
    - Email spam detection: words (frequency)
  - Database of labeled items might be enormous

Decision Tree

- A Decision Tree is a flowchart-like structure used for **classification** and **decision-making**. It splits data into branches based on feature values, *making it easy to interpret.*

- How Does a Decision Tree Work?

Step 1: Constructing the Decision Tree (Training Phase)

– The model analyzes training data and selects the best features to split on.

– Each split divides the data into smaller groups based on a decision rule.

– The process continues until the tree reaches a stopping condition (e.g., max depth or pure leaf nodes).

Step 2: Classifying New Data (Prediction Phase)

– A new data point follows the decision path in the tree.

– At each step, the model checks a feature value and moves to the next branch.

– The process continues until the data point reaches a leaf node, which determines the final classification.

Example: Should We Play Golf?

- Training data
  - <Outlook, Temperature, Humidity, Windy> -> Play Golf ?

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcoast	Mild	High	True	Yes
Overcoast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Problem Formulation

- What's the problem/task?
  - We want to predict whether a person should play golf based on weather conditions.
- The training data includes the following features:
  - Outlook, Temperature, Humidity, Windy
- Target Variable: Play Golf? (Yes/No)
  - **The goal** is to use past weather data to build a decision tree that predicts whether golf can be played on a given day.

Decision Tree – **Step 1**

- Step 1: Constructing the Decision Tree
  - A Decision Tree splits the data step by step, selecting the **most important feature** at each level.
  - Start with "Outlook" (Best Feature for the First Split)
    - If Overcast, always play golf (Yes).
    - If Sunny or Rainy, further conditions (like humidity and wind) determine the decision.

```
                  Outlook?
                 /   |   \
           Sunny  Overcast  Rainy
              /       |       \
      Humidity?        Yes   Windy?
        /   \                /   \
     High  Normal          No    Yes
      No      Yes          Yes     No
```

Tree Explanation

- If the outlook is overcast, always play .
- If the outlook is sunny, check humidity:
  - High humidity → No
  - Normal humidity → Yes
- If the outlook is rainy, check wind:
  - Not windy → Yes
  - Windy → No

Decision Tree – **Step 2**

- Step 2: Classifying New Data Using the Decision Tree
  - Now, we can **classify a new day** based on the tree:

    New Day 1: Sunny, Cool, Normal Humidity, Windy

    - Outlook = Sunny → Check Humidity
    - Humidity = Normal → Play Golf

    New Day 2: Rainy, Mild, High Humidity, Windy

    - Outlook = Rainy → Check Windy
    - Windy = Yes → Don’t Play Golf

Advantages

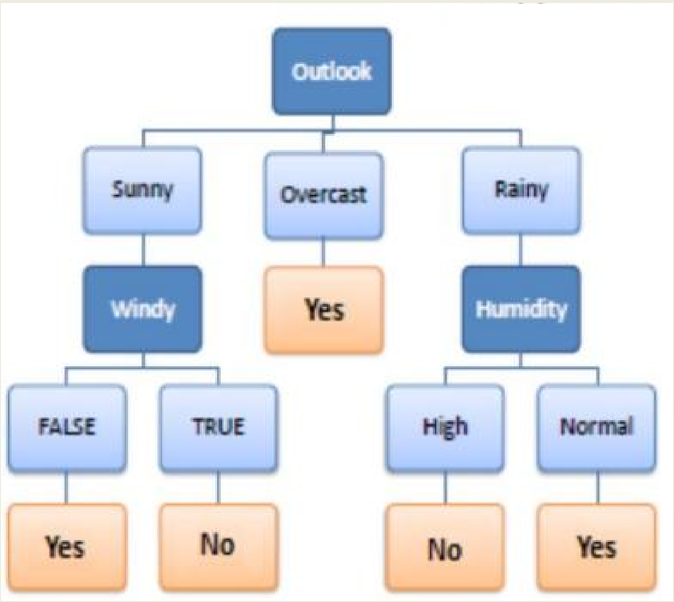
Easy to Interpret – Works like a set of "if-then" rules.

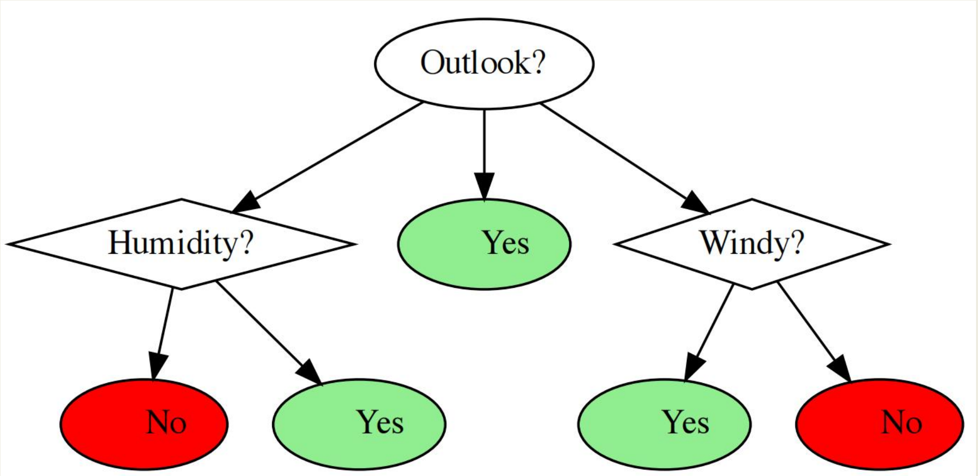
Can Handle Both Categorical & Numerical Data – Works for weather conditions, medical diagnosis, etc.

No Need for Complex Calculations – Unlike K-NN or SVM, no distance formulas are needed.

Visual Decision Tree

- Greedy Top-Down Learning of Decision Tree
  - Final decision tree





- New data item to classify: Navigate tree based on feature values

Challenges

- Primary challenge is building good decision trees from training data
  - Which features and feature values to use at each choice point
  - HUGE number of possible trees even with small number of features and value
- Common approach: “forest” of many trees, combine the results
  - Still impossible to consider all trees

Summary

- K-Nearest Neighbors (KNN)
  - Creating training / testing datasets
  - Different distance functions
- Decision tree
  - Greedy Top-down learning

References

- Web.stanford.edu/class/cs102