University Of Canberra

WEEK9

Pattern Recognition and Machine Learning

Created by **Elane Peng**

Objectives

- Data Preprocessing
- Feature Extraction
- Visualisation

Example 1 – Tabular Data (Iris Dataset)

Step 1. Load dataset

```
from sklearn.datasets import load_iris
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sns

# ----------------------------
# 1. Load dataset
# ----------------------------
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = [iris.target_names[i] for i in iris.target]  # add class labels

print("First 5 rows of the Iris dataset:")
print(df.head())
```

Example 1 – Tabular Data (Iris Dataset)

Step 1. Load dataset

- Classic dataset: 150 iris flowers.
- Each flower has 4 numeric features: sepal length/width, petal length/width.
- 3 classes: Setosa, Versicolor, Virginica.

Purpose: understand what kind of data we are dealing with.

Example 1 – Tabular Data (Iris Dataset)

When you first see this dataset, what would you want to check?
Number of classes? Balance of samples?

- There are 3 classes: Setosa, Versicolor, Virginica.
- Each has 50 samples, so the dataset is balanced.
- Balanced classes are useful because otherwise one class might dominate the analysis.

Example 1 – Tabular Data (Iris Dataset)

Step 2. Compute statistical features

```
# ------------------------------
# 2. Compute statistical features
# ------------------------------
stats = df.describe().T[['mean', 'std']]
stats['var'] = df.var(numeric_only=True)
stats['skew'] = df.skew(numeric_only=True)
print("\nStatistical Summary:")
print(stats)
```

Example 1 – Tabular Data (Iris Dataset)

Step 2. Compute statistical features

- mean: central value of each feature.
- std & var: how much it varies.
- skew: whether the distribution is symmetric or skewed.
- Purpose: to see which features vary more. For example, petal length varies much more than sepal width.

Example 1 – Tabular Data (Iris Dataset)

Looking at this table, which feature shows the largest variation?

Why do you think petal features vary more than sepal features?

- Petal length and petal width have much higher variance than sepal length/width.
- That means petals differ more across species.
- Biological reason: sepals are mainly protective, so they don’t vary much; petals are key in species reproduction and attract pollinators, so their size differs strongly across species.

Example 1 – Tabular Data (Iris Dataset)

Step 3. Normalize features

```
# ----------------------------
# 3. Normalize the features
# ----------------------------
scaler = MinMaxScaler()
df_norm = scaler.fit_transform(df.iloc[:, :-1])  # only numeric features
df_norm = pd.DataFrame(df_norm, columns=iris.feature_names)

print("\nFirst 5 rows of normalized data:")
print(df_norm.head())
```

Example 1 – Tabular Data (Iris Dataset)

Step 3. Normalize features

- Different features have different scales (ranges).
- Min-Max normalization rescales everything into [0,1].
- Without it, features with large ranges dominate distance-based models (like KNN).
- Analogy: comparing height (in meters) with weight (in kilograms) — unfair unless you rescale.

Example 1 – Tabular Data (Iris Dataset)

If we used KNN without normalization, which feature would dominate? Why would that be a problem?

- Features with larger numeric ranges (petal length ~1–7 cm) would dominate.
- Smaller-ranged features (sepal width ~2–4 cm) would be almost ignored.
- Problem: model bias — it will treat petal length as the most important even before training, just because of scale.

Example 1 – Tabular Data (Iris Dataset)

Step 4. Scatter plot

```
# ----------------------------
# 4. Scatter plot of two features
# ----------------------------
plt.figure(figsize=(6,4))
plt.scatter(df_norm.iloc[:,0], df_norm.iloc[:,1], c=iris.target, cmap='viridis')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.title("Normalized Iris Data (2D scatter)")
plt.grid(alpha=0.5, linestyle='--')
plt.show()
```

Example 1 – Tabular Data (Iris Dataset)

Step 4. Scatter plot

- Plot two features (sepal length vs sepal width).
- Each point = a flower, color = species.
- You can already see some clustering.

Example 1 – Tabular Data (Iris Dataset)

Do these two features separate the classes well? If not, what other feature pairs might work better?

- Sepal length vs sepal width: Setosa is somewhat separable, but Versicolor and Virginica overlap heavily.
- Better pair: petal length vs petal width, which almost perfectly separates Setosa from the others and also improves separation between Versicolor and Virginica.

Example 1 – Tabular Data (Iris Dataset)

Step 5. Pair plot

- Pair plot shows all feature combinations at once.
- Diagonal = distribution histograms.
- Off-diagonal = scatter plots of feature pairs.
- You can visually identify which feature pairs are most discriminative (e.g., petal length vs petal width).

Example 2 – Text Data (Movie Reviews Example)

1. Define Text Corpus
   - We start with a mini dataset, each entry is a short sentence.
   - The sentences share overlapping terms like machine, learning, AI.
   - Purpose: to show how different feature extraction methods handle shared meaning.

Example 2 – Text Data (Movie Reviews Example)

2. Bag-of-Words (CountVectorizer)
   - Break each sentence into tokens (words).
   - Build a Document-Term Matrix:
     - Rows = documents (sentences).
     - Columns = words in the vocabulary.
     - Entries = counts of each word.
   - Example: the word machine occurs in multiple sentences, so the counts appear in those rows.
   - Limitation:
     - All words are treated equally.
     - No weighting, no context (so is and learning are seen as equally important).

Example 2 – Text Data (Movie Reviews Example)

3. TF-IDF (Term Frequency–Inverse Document Frequency)

- Improvement over Bag-of-Words: words are weighted by importance.
- TF = frequency of a word in a document.
- IDF = penalty for words that appear everywhere.
- Effect:
- Rare but meaningful words (e.g. learning) → high weight.
- Common words (e.g. is) → low weight.
- Output: a sparse matrix of fractional scores instead of raw counts.

Example 2 – Text Data (Movie Reviews Example)

4. Sentence Similarity (Cosine Similarity)

- Represent each sentence as a TF-IDF vector.
- Compute similarity as the cosine of the angle between two vectors:
  - 1 = identical (same direction).
  - 0 = no overlap (orthogonal).
- Example:
  - “Machine learning is fascinating” and “Machine learning and AI are related” → high similarity.
  - “Natural language processing is a key AI field” → less similar to the first two.

Example 2 – Text Data (Movie Reviews Example)

Summary

This example shows the basic workflow for text preprocessing:

1. Start with raw sentences.
2. Convert them into numerical form using Bag-of-Words.
3. Improve representation with TF-IDF weighting.
4. Use mathematical similarity (cosine similarity) to compare documents.

Example 2 – Text Data (Movie Reviews Example)

Summary

This example shows the basic workflow for text preprocessing:

1. Start with raw sentences.
2. Convert them into numerical form using Bag-of-Words.
3. Improve representation with TF-IDF weighting.
4. Use mathematical similarity (cosine similarity) to compare documents.

Example 3 – Image Data (Facial Image Processing)

1. Load Image
   - The example uses the astronaut image, a standard test image in image processing.
   - It is first converted from color (RGB) to grayscale:
     - Color images have 3 channels (red, green, blue), which makes computation heavier.
     - Grayscale keeps only intensity (brightness), simplifying calculations.
   - Purpose: For feature extraction, structure and shape are often more important than color.

Example 3 – Image Data (Facial Image Processing)

2. Edge Detection

Edges are places where intensity changes sharply — e.g., face outlines, eyes, nose boundaries. They are essential for detecting object shapes.

Sobel Operator

- Calculates gradients in the x and y directions.
- Large gradients indicate edges.
- Produces thicker, less precise edges.
- Use case: Quick detection of general object boundaries.

Example 3 – Image Data (Facial Image Processing)

Canny Edge Detector

- More advanced than Sobel, with multiple steps:
    a. Noise reduction.
    b. Gradient calculation.
    c. Non-maximum suppression (keep only thin edges).
    d. Edge tracking with thresholds.
- Produces thin, clean, accurate edges.
- Use case: Face recognition, license plate detection, medical images.

Example 3 – Image Data (Facial Image Processing)

3. Texture Features – Local Binary Pattern (LBP)

- Idea: For each pixel, compare its neighbors to the center pixel:
  - If a neighbor is brighter → assign 1.
  - If a neighbor is darker → assign 0.
- The 8 neighbors form a binary number (pattern).
- This number encodes the local texture of that region.

Example 3 – Image Data (Facial Image Processing)

Why useful?

- Smooth areas (e.g., skin, sky): neighbors look similar → uniform patterns.
- Rough areas (e.g., hair, beard, fabric): neighbors differ → more complex patterns.
- Applications:
  - Face recognition (skin textures and local features).
  - Material classification (smooth metal vs rough wood).
  - Scene classification (grassland, desert, forest).

Example 3 – Image Data (Facial Image Processing)

Why useful?

- Smooth areas (e.g., skin, sky): neighbors look similar → uniform patterns.
- Rough areas (e.g., hair, beard, fabric): neighbors differ → more complex patterns.
- Applications:
  - Face recognition (skin textures and local features).
  - Material classification (smooth metal vs rough wood).
  - Scene classification (grassland, desert, forest).

Example 3 – Image Data

Summary

- Grayscale: simplify computation, focus on structure.
- Edges (Sobel, Canny): detect boundaries; Sobel = coarse, Canny = precise.
- Texture (LBP): turn local brightness differences into binary codes, useful for surface detail.

Example 4 – Audio Data Processing

1. Load Audio
   - We use scipy.io.wavfile.read() to load a .wav file.
   - A .wav file is essentially an array of numbers — each number represents the sound wave’s amplitude at a specific time.
   - Audio is often recorded in stereo (two channels). If so, we take just one channel to simplify.

Example 4 – Audio Data Processing

2. Compute Features Manually

Zero Crossing Rate (ZCR)

- Definition: number of times the waveform crosses the zero line per unit length.
- Intuition:
  - High-frequency sounds (like hiss “sss”) → high ZCR.
  - Low-frequency sounds (like deep “aaa”) → low ZCR.
- Use case: speech/music classification, detecting noisy vs tonal sounds.

Example 4 – Audio Data Processing

Energy

- Definition: the average power of the signal.
- Intuition:
  - Louder sounds → higher energy.
  - Quiet or silent sections → low energy.
- Use case: speech detection (separating silence vs voiced segments).

Example 4 – Audio Data Processing

3. Visualization

Waveform

- Plot amplitude vs time.
- Shows how loudness changes.

Spectrogram

- Generated using scipy.signal.spectrogram.
- Shows frequency content over time.
- X-axis = time, Y-axis = frequency, Color = intensity (power).
- Interpretation: vowels appear as horizontal “formant bands,” consonants as bursts.

Example 4 – Audio Data Processing

Power Spectrum (via FFT)

- FFT = Fast Fourier Transform.
- Converts signal from time domain → frequency domain.
- Shows how much energy is present at each frequency.
- Useful to identify dominant frequencies (e.g., pitch of a note).

Example 4 – Audio Data Processing

- Audio features:
  - ZCR → frequency characteristics.
  - Energy → loudness.
  - Spectrogram/FFT → frequency structure.
  - Together, they allow analysis of sound beyond just raw waveforms.

Example 5 – Visualization & Dimensionality Reduction

Dimensionality reduction means taking high-dimensional data (lots of features) and projecting it into fewer dimensions, while trying to keep the “essence” of the data. Different methods emphasize different kinds of structure:

- PCA (Principal Component Analysis): Finds directions where the data varies the most, and keeps those. It’s like rotating the dataset to find the “widest view.” It doesn’t care about labels (unsupervised).
- LDA (Linear Discriminant Analysis): Uses class labels to find directions that best separate categories. It’s like finding camera angles that make two different groups stand apart most clearly. (supervised).
- t-SNE (t-Distributed Stochastic Neighbor Embedding): Focuses on local neighborhoods, making sure similar points in high-dim space stay close together in the low-dim space. It’s nonlinear, good for visualization, but less for prediction.