Introduction to Project Session 2

Course: Machine Learning for Geo-Science
Session Title: Practical Data Processing and Visualization with SYNOP Weather Data
Instructor: Amir Baqerzadeh [amirbaqerzadeh_at_gmail.com]

Project Overview

In this session, you will work with **real-world meteorological data** extracted from SYNOP (Surface Synoptic Observations) messages. These messages are encoded in a special format called **FM-12**, used worldwide for weather reporting.

Your goal is to:

Decode SYNOP messages, extract meaningful weather parameters (like temperature), and prepare the data for further analysis or machine learning tasks.

This is a **real-world data wrangling task**, where you'll:

- Read messy, encoded data
- Clean and parse it
- Extract numeric values
- Handle missing or malformed data
- Visualize and summarize the results

Sub-Purposes

This project is broken into the following **sub-tasks**:

Sub-Purpose	Description
1. Load and Explore Data	Read multiple Excel files containing SYNOP messages and combine them into a single DataFrame.
2. Understand the Message Format	Learn how temperature is encoded in FM-12 messages (e.g., `10166` → 16.6°C).
3. Extract Temperature	Write a function to extract temperature from the encoded string using indexing and string slicing.
4. Handle Missing or Malformed Data	Use `if` conditions and `try/except` to skip or interpolate missing values.
5. Clean and Prepare the Dataset	Drop NaNs, reset indices, and prepare a clean time-series dataset.
6. Visualize the Results	Use `seaborn` or `matplotlib` to plot the distribution of temperature values.

Tools and Libraries

You will use the following **Python tools**:

Tool	Purpose
`pandas`	Reading Excel files, cleaning data, applying functions, handling NaNs
`numpy`	Numerical operations, handling missing values (`np.nan`)
`matplotlib` / `seaborn`	Visualizing temperature distributions
`glob`	Loading multiple Excel files from a folder
`str` methods	Parsing and slicing encoded strings
`apply()`	Applying custom functions to DataFrame columns

Skills You Should Be Comfortable With

To succeed in this session, you should be familiar with:

Skill	Why It Matters
String indexing and slicing	You’ll extract temperature from a fixed-position substring
Writing functions	You’ll write a function to parse temperature from a message
Handling missing data	Some messages are incomplete or malformed
Using `.apply()`	You’ll apply your parsing function to a whole column
Working with datetime	The `date` column is a string that needs to be parsed
Basic plotting	You’ll visualize the distribution of temperature
Debugging	You’ll need to test your function on sample rows

Final Output

By the end of this session, you will have:

- A **clean DataFrame** with a new `temp` column
- A **time-series** of temperature values
- A **histogram** showing the distribution of temperatures
- A **real-world example** of how to extract insights from encoded data

Tips for Students

- Start by **exploring** a few rows manually before writing your function.
- Use `.apply()` **only after** your function works on a single row.
- Use `.interpolate()` to fill missing values **only if** it makes sense.
- Always **visualize** your final data to check for anomalies.

Understanding the SYNOP Code Format

Each SYNOP message is a long string of encoded weather data.
Let's break down how **temperature** is hidden inside it.

Example:

We are interested in the **5-digit group** that starts with `1`. Look at figure 1.

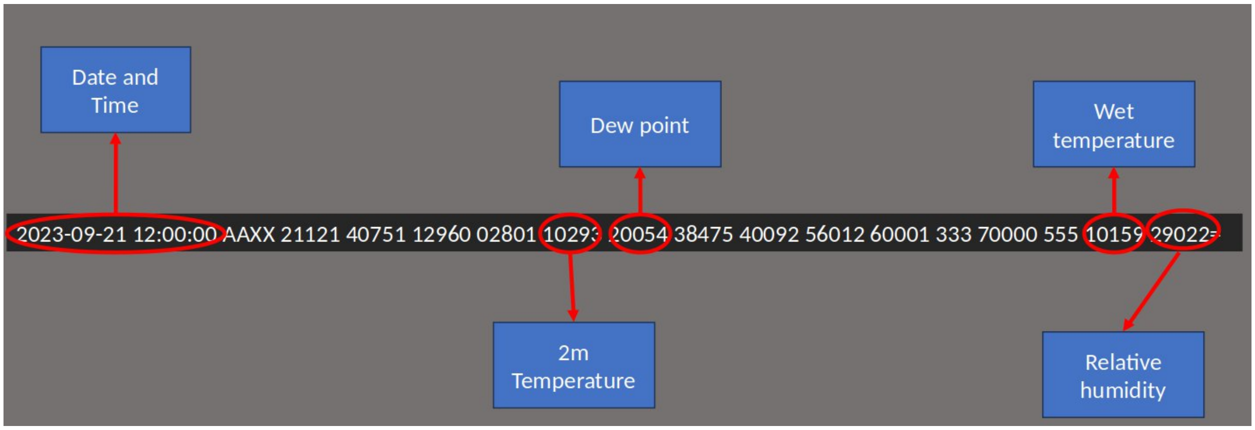
Date and Time
Dew point
Wet temperature
2023-09-21 12:00:00 AAXX 21121 40751 12960 02801 10293 20054 38475 40092 56012 60001 333 70000 555 10159 29022=
2m Temperature
Relative humidity

Figure 1. A Synope code and meaning of some its groups

What the Image Tells Us

(The screenshot is a **CSV-like preview** of the raw SYNOP messages you are working with.)

Column	Example	Meaning
Date and Time	`2023-09-21 12:00:00`	Observation time-stamp
SYNOP string	`AAXX 21121 40751 12960 02801 10293 20054 …`	**Full FM-12 encoded message**
Temperature	*(empty)*	**This is what we must extract**
Relative humidity	*(empty)*	Another parameter we could decode later

Focus on the Temperature Field

Inside the string you can see the group:

```
10293
```

Decode it with the rule we just learnt:

Digit	Value	Meaning
1	**1**	Temperature group identifier
0	**0**	Sign: **positive**
293	**29.3**	Temperature × 10 ⇒ **29.3 °C**

Therefore, for this row the extracted temperature is **+29.3 °C**

Your Function Should

1. Grab the **SYNOP string** column.
2. Search for the **5-digit group starting with ‘1’**.

3. Apply the **1 / 0 / 293 → +29.3 °C** rule.
4. Return a **float** and store it in the **Temperature** column.

Practice Exercise – Based on Figure 1

Use the **exact same SYNOP string** you saw in Figure 1:

```
AAXX 21121 40751 12960 02801 10293 20054 38475 40092 56012 60001
333 70000 555 10159 29022=
```

Your task is to **write Python code** that extracts the **three** parameters below **for every 8 standard synoptic hours**:

Table

Copy

Parameter	Encoded Group	Decode Rule	Example Value
Air temperature	`1xxxx`	`1 s ttt` → sign + `ttt/10`	`10293` → **+29.3 °C**
Dew-point temperature	`2xxxx`	`2 s ddd` → sign + `ddd/10`	`20054` → **+5.4 °C**
Relative humidity	next 2 digits **after** the dew-point group	`%` directly	`38` → **38 %**

Only Process These Synoptic Hours

SYNOP messages are reported **every 3 hours**.
Only process the rows whose **time ends in**:

```
00 03 06 09 12 15 18 21
```

(These are the **standard synoptic hours**; their structure is **guaranteed** to be complete.)

Deliverable

A **clean DataFrame** with three new columns:

- `Temp` – air temperature (°C)
- `Dewp` – dew-point temperature (°C)
- `RH` – relative humidity (%)

Only for synoptic hours 00, 03, 06, 09, 12, 15, 18, 21.