📘 Ordinal Encoding – Complete Notes

🔶 1. What are Categorical Variables?

Categorical variables are variables that contain **label values** rather than numeric values. These labels represent **categories** or **groups**.

Examples:

- Gender: "Male", "Female"
- City: "Lahore", "Karachi", "Islamabad"
- Education: "High School", "Bachelor", "Master"

These variables **cannot be used directly in ML models** because models require **numeric input**, not strings. So, we need to convert them using encoding techniques.

🔶 2. Types of Categorical Variables

There are **two main types** of categorical variables:

📍 a) Nominal Variables

- These have **no natural order or ranking** between categories.
- Examples:
    - "Red", "Blue", "Green" (Colors)
    - "Male", "Female" (Gender)
    - "Lahore", "Karachi" (City)

➡️ We use **One-Hot Encoding** for nominal data.

📍 b) Ordinal Variables

- These have a **meaningful order or ranking** between categories.
- But the **distance between them is not known**.
- Examples:
    - Education level: "High School" < "Bachelor" < "Master" < "PhD"
    - Customer Rating: "Bad" < "Average" < "Good" < "Excellent"

➡️ We use **Ordinal Encoding** for ordinal data.

🔶 **3. What is Ordinal Encoding?**

Ordinal Encoding is a technique to **convert ordinal categorical values into integers** based on **their order**.

Example:

["Low", "Medium", "High"] → [0, 1, 2]

So the order is preserved, and we can pass this numeric representation to ML models.

🔶 **4. When to Use Ordinal Encoding?**

✅ Use ordinal encoding **only when the categorical data has a clear order** or hierarchy.

❌ Do **not** use it for **nominal data**, because then numbers may give **false meaning** to models.

🔶 **5. How to Perform Ordinal Encoding in Python (sklearn)**

📦 **Import:**

```
from sklearn.preprocessing import OrdinalEncoder
import pandas as pd
```

📄 **Sample Data:**

```
data = pd.DataFrame({
   "Education": ["High School", "Bachelor", "Master", "PhD", "Bachelor"]
})
```

✅ **Encoding:**

```
encoder = OrdinalEncoder(categories=[["High School", "Bachelor", "Master", "PhD"]])
encoded = encoder.fit_transform(data[["Education"]])
print(encoded)
```

🧻 **Output:**

```
[[0.]
 [1.]
 [2.]
```

[3.]

[1.]]

🔶 6. Explanation of Code

- **OrdinalEncoder()**: Used to encode ordinal features as integer arrays.
- **categories parameter**: Manually defines the **order** of the categories.
- **fit_transform()**: Fits the encoder and transforms the data at once.

If you don't manually specify order, sklearn will sort alphabetically (⚠️ not good for ordinal data with custom order).

🔶 7. Using Ordinal Encoding with Multiple Columns

```
df = pd.DataFrame({
   "Education": ["High School", "Bachelor", "Master", "PhD"],
   "Experience": ["Junior", "Mid", "Senior", "Lead"]
})


encoder = OrdinalEncoder(categories=[
   ["High School", "Bachelor", "Master", "PhD"],  # custom order
   ["Junior", "Mid", "Senior", "Lead"]          # custom order
])


encoded = encoder.fit_transform(df)
print(encoded)
```

🔶 8. How is It Different from Label Encoding?

Feature	Label Encoding	Ordinal Encoding
Suitable for	Any categorical data	Only ordinal data (with natural order)
Order preserved?	No	Yes

Feature	Label Encoding	Ordinal Encoding
Manual order set?	❌ No	✅ Yes (you define category order)
Use Case	Mostly for target/label column	For ordinal input features

---

🔶 9. Pros and Cons

✅ **Pros:**

- Simple and fast
- Keeps the **ranking information**
- Useful for **tree-based algorithms**

❌ **Cons:**

- Can **mislead linear models** if the distance between categories is not meaningful.
    - E.g., model may assume that "PhD" = 3 is 3x more important than "High School" = 0, which may not be true.

🔶 10. When Not to Use Ordinal Encoding?

- When the categories **don't have a clear order** → Use **One-Hot Encoding**
- When **equal spacing is not guaranteed** → Use techniques like **target encoding** or **embedding** (in deep learning)

🔶 11. Alternative: Pandas .replace() Method

You can also perform ordinal encoding manually:

```
mapping = {"Low": 0, "Medium": 1, "High": 2}

df["Risk"] = df["Risk"].replace(mapping)
```

✅ Useful for quick and simple tasks
❌ Not reusable like sklearn's encoder

✅ Summary Chart

Category Type	Encoding Method	Order Exists?	Example
Nominal	One-Hot Encoding	❌	City, Gender, Country
Ordinal	Ordinal Encoding	✅	Education, Size, Satisfaction

📌 Bonus: Use with Pipelines

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

ordinal_cols = ["Education"]
ordinal_categories = [["High School", "Bachelor", "Master", "PhD"]]

ordinal_encoder = OrdinalEncoder(categories=ordinal_categories)

preprocessor = ColumnTransformer(
    transformers=[
        ("ord", ordinal_encoder, ordinal_cols)
    ],
    remainder='passthrough'
)

# Now plug `preprocessor` into any ML pipeline
```
