📘 One-Hot 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"

Machine learning models cannot work with text data directly, so we need to **convert categorical data into numeric format**.

🔶 2. Types of Categorical Variables

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

📍 a) Nominal Variables

- No natural order or ranking
- Examples:
  - Gender: "Male", "Female"
  - City: "Lahore", "Karachi", "Islamabad"

➡️ Use **One-Hot Encoding**

📍 b) Ordinal Variables

- Have meaningful order but unknown distance
- Examples:
  - Education: "High School" < "Bachelor" < "Master" < "PhD"

➡️ Use **Ordinal Encoding**

🔶 3. What is One-Hot Encoding?

One-Hot Encoding converts each category value into a new **binary column (0 or 1)**.

Example:
Categories: ["Red", "Blue", "Green"]

After One-Hot Encoding:

Red	Blue	Green
1	0	0
0	1	0
0	0	1

Each category is represented as a **new column** with a 1 indicating presence.

🔶 4. When to Use One-Hot Encoding?

✅ Use it when:

- The categorical variable is **nominal**
- There is **no natural ordering** in the values

❌ Do NOT use it for **ordinal data** (like Low < Medium < High)

🔶 5. How to Perform One-Hot Encoding in Python

🧪 Example 1: Using pandas

```
import pandas as pd


df = pd.DataFrame({
    "City": ["Lahore", "Karachi", "Islamabad", "Lahore"]
})


encoded = pd.get_dummies(df, columns=["City"])
print(encoded)
```

Output:

City_Islamabad	City_Karachi	City_Lahore
0	0	1
0	1	0
1	0	0
0	0	1

🧪 Example 2: Using sklearn

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

df = pd.DataFrame({
    "City": ["Lahore", "Karachi", "Islamabad"]
})

encoder = OneHotEncoder(sparse_output=False)
encoded = encoder.fit_transform(df[["City"]])
print(encoded)
```

Output:

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

Note: sparse_output=False gives dense NumPy array.

🔶 6. Drop First Column (to avoid multicollinearity)

To reduce redundancy:

```
pd.get_dummies(df, columns=["City"], drop_first=True)
```

OR

```
encoder = OneHotEncoder(drop="first", sparse_output=False)
```

This helps especially in **linear models** where multicollinearity is a problem.

🔶 7. Pros and Cons

✅ **Pros:**

- Simple and effective
- Preserves all category info
- Works well with tree-based models

❌ **Cons:**

- Increases dimensionality
- Not suitable for high-cardinality features (e.g. thousands of categories)

🔶 8. One-Hot Encoding vs Ordinal Encoding

Feature	**One-Hot Encoding**	**Ordinal Encoding**
Type of data	Nominal	Ordinal
Order considered?	❌ No	✅ Yes
Suitable for	Any model	Tree-based or ordinal
Output columns	Multiple	Single integer column

🔶 9. Use with ColumnTransformer in Pipelines

```
from sklearn.compose import ColumnTransformer


ohe = OneHotEncoder(drop="first", sparse_output=False)


preprocessor = ColumnTransformer(
    transformers=[
        ("cat", ohe, ["City"])
    ],
    remainder="passthrough"
)
```

✅ **Summary:**

- Use **One-Hot Encoding** for categorical data **without order**
- Can use pandas.get_dummies() or sklearn.OneHotEncoder
- Be cautious of **dimensionality explosion**
- Drop first column if needed to avoid **dummy variable trap**
