Case Study: Encoding the Data in Machine Learning

Objective

To understand how **different encoding methods** (Label Encoding, One-Hot Encoding, and Ordinal Encoding) are applied to transform categorical data into numerical form suitable for machine learning algorithms.

Dataset: Customer Purchase Data

Customer_ID	Gender	City	Education	Purchased
1	Male	Delhi	Graduate	Yes
2	Female	Mumbai	Post-Graduate	No
3	Female	Delhi	Undergraduate	Yes
4	Male	Chennai	Graduate	No
5	Female	Kolkata	Post-Graduate	Yes

Step 1: Understanding the Problem

Machine learning models cannot process **textual or categorical data** directly.
So before building a model, we must convert features like **Gender**, **City**, and **Education** into **numeric representations**.

Step 2: Import Required Libraries

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, OrdinalEncoder
```

Step 3: Create the DataFrame

```
# Create dataset
data = {
   'Gender': ['Male', 'Female', 'Female', 'Male', 'Female'],
   'City': ['Delhi', 'Mumbai', 'Delhi', 'Chennai', 'Kolkata'],
   'Education': ['Graduate', 'Post-Graduate', 'Undergraduate', 'Graduate', 'Post-Graduate'],
   'Purchased': ['Yes', 'No', 'Yes', 'No', 'Yes']
}
```

```
df = pd.DataFrame(data)
print(df)
```

Step 4: Method 1 – Label Encoding

Used for **ordinal or binary categorical data** (e.g., Gender, Purchased).

```
le = LabelEncoder()
df['Gender_Label'] = le.fit_transform(df['Gender'])
df['Purchased_Label'] = le.fit_transform(df['Purchased'])
print(df[['Gender', 'Gender_Label', 'Purchased', 'Purchased_Label']])
```

Result:

Gender	Gender_Label	Purchased	Purchased_Label
Male	1	Yes	1
Female	0	No	0
Female	0	Yes	1
Male	1	No	0
Female	0	Yes	1

Step 5: Method 2 – One-Hot Encoding

Used for **nominal categorical data** (e.g., City).

```
df_encoded = pd.get_dummies(df, columns=['City'])
print(df_encoded)
```

Result:

Gender	Education	Purchased	City_Chennai	City_Delhi	City_Kolkata	City_Mumbai
Male	Graduate	Yes	0	1	0	0
Female	Post-Graduate	No	0	0	0	1
Female	Undergraduate	Yes	0	1	0	0
Male	Graduate	No	1	0	0	0
Female	Post-Graduate	Yes	0	0	1	0

Each city becomes its own column with binary indicators.

Step 6: Method 3 – Ordinal Encoding

Used for **ordered categorical data** (e.g., Education level).

```
ord_enc = OrdinalEncoder(categories=[['Undergraduate', 'Graduate', 'Post-Graduate']])
df['Education_Encoded'] = ord_enc.fit_transform(df[['Education']])
print(df[['Education', 'Education_Encoded']])
```

Result:

Education	Education_Encoded
Graduate	1
Post-Graduate	2
Undergraduate	0
Graduate	1
Post-Graduate	2

Maintains the **order/priority** of education levels.

Step 7: Final Encoded Dataset

Gender_Label	City_Delhi	City_Mumbai	City_Kolkata	City_Chennai	Education_Encoded	Purchased_Label
1	1	0	0	0	1	1
0	0	1	0	0	2	0
0	1	0	0	0	0	1
1	0	0	0	1	1	0
0	0	0	1	0	2	1

Step 8: Observation

- **Label Encoding** simplifies binary features.
- **One-Hot Encoding** helps represent categories without implying order.
- **Ordinal Encoding** is ideal for ranked data.
- Together, they make categorical data ready for ML models like **Linear Regression, Decision Trees, and Logistic Regression**.