Advanced Data Handling

It’s getting intense

3.1
What is **Data Encoding?**

An intuitive understanding of Encoding Data

- When we have a dataset ready and preprocessed we can start to feed it to the model
- However, it cannot be that direct
- We all know that computers only understand 1s and 0s
- Similarly, models can only understand **encoded data**

Types of Encoding

When it comes to the ways to encode data, the list does not end. We will cover two ways of encoding categorical data to demonstrate

- Label Encoding
- One Hot Encoding

3.2
Label **Encoding.**

What is Label Encoding

- Label Encoding allows us to convert categorical data into numerical representations
- Example:
  - A column that measures feature as high, medium, low
  - To preserve meaning of the ordinal values we encode as:
    - High = 3
    - Medium = 2
    - Low = 1

Limitations of Label Encoding

- The unique number assigned may lead to a priority issue during training the model
- A label with a high value could we wrongly perceived to have high priority than a label having a lower value
    - Example: If we have 3 countries Canada, America, Mexico and we label encode them as 0, 1, 2. It could be interpreted that Mexico has high priority even though that is not meanigful

3.3
One-Hot Encoding.

What is One-Hot Encoding

- “One-Hot” refers to the idea that each category of data is made into a series where the category that is true is set to 1 and the rest are 0
- This is best understood through an example

Original Data

Fruit	Price
apple	5
mango	10
apple	15
orange	20

One-Hot Encoded Data

apple	mango	orange	price
1	0	0	5
0	1	0	10
1	0	0	15
0	0	1	20

Advantages of One-Hot Encoding

- Avoids the problem of ordinality after encoding
- Allows the use of categorical variables in models that require numerical inputs

Disadvantages of One-Hot Encoding

- Increased dimensionality
  - Additional columns created for each category which makes the data complex and harder to train
- Sparse data
  - Most observations in the columns created will be 0

3.4
Feature Scaling.

What is Feature Scaling?

- Featuring scaling is transforming features to a similar scale
- This is needed for algorithms sensitive to scaling
- Examples of such algorithms are:
  - K-means clustering
  - Support Vector Machines
  - Neural Networks

Feature Scaling Method: Normalization

- Beneficial when we know the bounds of the data and the need values are between 0 and 1
- Implementation

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
```

Feature Scaling Method: Z-score normalization

- Beneficial when data has a Gaussian distribution.
- Implementation

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
```

Feature Scaling Method: Max Absolute Scaling

- Beneficial when absolute values are required.
- Implementation

```
from sklearn.preprocessing import MaxAbsScaler
scaler = MaxAbsScaler()
data_scaled = scaler.fit_transform(data)
```

Feature Scaling Method: L2 normalization

- Beneficial when data is a vector and we need to consider the cosine similarity
- Implementation

```
from sklearn.preprocessing import Normalizer
scaler = Normalizer()
data_scaled = scaler.fit_transform(data)
```

3.5
Data **Visualization** Techniques.

Why Visualize Data?

- Data often hides patterns that can be found if visualized correctly
- This is especially helpful if the data in question is novel
- Visualization also helps catch issues in data that might not seem apparent in any other way
- We can use a lot of libraries to visualize data. Best shown through code

END.