Data Preprocessing in Machine Learning using Scikit-learn

In this lecture, Professor Ashish discusses data preprocessing techniques in machine learning, focusing on Scikit-learn's capabilities. Real-world datasets often contain imperfections requiring preprocessing before model training.

Common Data Issues and Preprocessing Steps

Typical problems include:

- Missing values in features.
- Features on different scales.
- Categorical attributes needing numerical representation.
- Irrelevant or excessive features.
- Non-numerical data (images, text) requiring feature extraction.

Scikit-learn (`sklearn`) provides tools to address these issues. The `sklearn.preprocessing` module handles standardization, missing value imputation, and feature scaling. `sklearn.feature_extraction` offers feature extractors for various data types. Dimensionality reduction techniques, such as Principal Component Analysis (PCA) in `sklearn.decomposition`, are used to handle high-dimensionality. Feature expansion methods are also available (discussed later in the context of Support Vector Machines).

Handling Missing Values with Scikit-learn

Scikit-learn's `impute` module provides methods for handling missing values. `SimpleImputer` can replace missing values using strategies like:

- Mean
- Median
- Most frequent value
- A constant value

`KNNImputer` uses k-Nearest Neighbors to impute missing values based on the values of its nearest neighbors. For example, to impute a missing value in a feature, it averages the values of that feature among its k-nearest neighbors. The `MissingIndicator` class helps track the presence of missing values.

```
# Example using SimpleImputer
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
transformed_data = imputer.fit_transform(data)
```

> ***Note:*** *The output of `MissingIndicator` is a SciPy sparse matrix.*

Feature Transformation and Vectorization

`DictVectorizer` transforms dictionaries into feature matrices. `FeatureHasher` offers a high-speed, low-memory alternative using feature hashing, but sacrifices inspectability of the features.

Pipelines for Uniform Preprocessing

Scikit-learn's pipeline functionality allows chaining multiple transforms for uniform application across training, evaluation, and testing sets. This ensures consistency in preprocessing steps.

Prof. Ashish concludes by emphasizing the importance of consistent preprocessing across different datasets (train, test, etc.) and the utility of Scikit-learn's tools in efficiently and effectively handling data preprocessing challenges.