**Data transformations**

Data transformations in machine learning involve altering or preprocessing raw data into a form suitable for training and evaluating machine learning models. These transformations are crucial for improving model performance, ensuring data consistency, and addressing issues such as missing values, noise, or scaling differences.

Feature scaling

Standardiviation normalization

* Minimax scaller
* Robust scaller

**Standardivation is also called Z-score normalization**

**Xi=Xi-Xbar/sigma** for example we are taking age and salary age=12,18,17,19,20 age’

Here xbar is the mean

**1. Scaling and Normalization**

**a. Min-Max Scaling**

Transforms data to a fixed range, usually [0, 1].

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

# Sample Data

data = pd.DataFrame({'A': [10, 20, 30], 'B': [40, 50, 60]})

# Min-Max Scaling

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(data)

# Result

scaled\_df = pd.DataFrame(scaled\_data, columns=data.columns)

print(scaled\_df)

**b. Z-Score Standardization**

Standardizes data to have a mean of 0 and a standard deviation of 1.

from sklearn.preprocessing import StandardScaler

# Z-Score Standardization

scaler = StandardScaler()

z\_scaled\_data = scaler.fit\_transform(data)

# Result

z\_scaled\_df = pd.DataFrame(z\_scaled\_data, columns=data.columns)

print(z\_scaled\_df)

2. Encoding Categorical Variables

a. One-Hot Encoding

Converts categorical variables into a binary matrix.

# Sample Data

data = pd.DataFrame({'Category': ['A', 'B', 'A', 'C']})

# One-Hot Encoding

encoded\_data = pd.get\_dummies(data, columns=['Category'])

print(encoded\_data)

b. Label Encoding

Converts categorical variables into integer codes.

from sklearn.preprocessing import LabelEncoder

# Label Encoding

encoder = LabelEncoder()

data['Category\_Encoded'] = encoder.fit\_transform(data['Category'])

print(data)

**3. Handling Missing Data**

**a. Fill Missing Values**

Replace missing values with a specific value, mean, median, or mode.

# Sample Data

data = pd.DataFrame({'A': [1, None, 3], 'B': [4, 5, None]})

# Fill with Mean

data.fillna(data.mean(), inplace=True)

print(data)

**b. Drop Missing Values**

Remove rows or columns with missing data.

# Drop Missing Values

data.dropna(inplace=True)

print(data)

**4. Feature Transformation**

**a. Log Transformation**

Compresses large ranges of values.

import numpy as np

# Sample Data

data = pd.DataFrame({'A': [1, 10, 100]})

# Log Transformation

data['Log\_A'] = np.log(data['A'])

print(data)

**5. Feature Selection**

**a. Removing Low-Variance Features**

Keeps features with high variance.

from sklearn.feature\_selection import VarianceThreshold

# Sample Data

data = pd.DataFrame({'A': [1, 1, 1], 'B': [1, 2, 3]})

# Remove Low-Variance Features

selector = VarianceThreshold(threshold=0.1)

selected\_data = selector.fit\_transform(data)

# Result

selected\_df = pd.DataFrame(selected\_data, columns=['B'])

print(selected\_df)