UNIT 1: INTRODUCTION TO DATA PREPROCESSING

1.1 Data Objects and Attribute Types

Data Objects: A data object is a collection of attributes that describe an entity. For example, in a student database, each student is a data object, and their name, age, roll number, and GPA are the attributes.

Attribute Types:

1. **Nominal**: Categories without order (e.g., color: red, blue, green)
2. **Ordinal**: Categories with a meaningful order but unequal differences (e.g., satisfaction: low, medium, high).
3. **Interval**: Numerical data with equal differences, but no true zero (e.g., temperature in Celsius)
4. **Ratio**: Like interval, but with a true zero point (e.g., age, salary, height)

Importance: Understanding attribute types helps in selecting the right preprocessing and modeling techniques. Some models require numerical inputs, while others can work with categorical data.

1.2 Measuring Data Similarity and Dissimilarity

Similarity and Dissimilarity:

- **Similarity** measures how alike two data objects are. Higher similarity means they are more alike.
- **Dissimilarity (Distance)** is how different two objects are. Higher distance means more difference.

Distance Metrics:

1. **Euclidean Distance**: Straight-line distance between two points. Formula: $d = \sqrt{\sum (x_i - y_i)^2}$
2. **Manhattan Distance**: Sum of absolute differences. Formula: $d = \sum |x_i - y_i|$
3. **Minkowski Distance**: Generalization of Euclidean and Manhattan. Formula: $d = (\sum |x_i - y_i|^p)^{1/p}$
4. **Cosine Similarity**: Measures the cosine of the angle between two vectors. Formula: $\cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||}$
5. **Jaccard Similarity**: Used for set-based data. Formula: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$

Mixed-Type Attributes: For data with both numeric and categorical values, use:

- Gower's Distance
- Convert categories using encoding methods first

1.3 Data Preprocessing

Definition: Data preprocessing involves transforming raw data into a format suitable for analysis. It is crucial as raw data is often noisy, inconsistent, and incomplete.

Steps in Data Preprocessing:

1. **Data Cleaning:**
2. Handle missing values (mean/median imputation, drop rows)
3. Remove duplicates
4. Fix inconsistent formats
5. **Data Transformation:**
6. Encoding categorical variables (Label encoding, One-hot encoding)
7. Normalization and standardization
8. Binning or discretization
9. **Data Reduction:**
10. Dimensionality reduction (PCA)
11. Sampling

Normalization vs Standardization:

- **Normalization (Min-Max Scaling)**: Scales features to [0, 1] $x' = \frac{x - min(x)}{max(x) - min(x)}$
- **Standardization (Z-Score Normalization)**: Rescales data to have mean = 0 and std dev = 1 $z = \frac{x - \mu}{\sigma}$
- **Decimal Scaling**: Move decimal point based on max absolute value $x' = \frac{x}{10^j}$, where j is the smallest integer s.t. |x'| < 1

UNIT 2: INTRODUCTION TO MACHINE LEARNING

2.1 Origins of Machine Learning

Definition: Machine Learning (ML) is a field of artificial intelligence that focuses on developing algorithms that allow computers to learn from data without being explicitly programmed.

History:

- Originated from pattern recognition and computational learning theory.
- Evolved from statistics, data mining, and computer science.

- Gained popularity due to increase in data, computational power, and advancements in algorithms.

2.2 Basic Learning Process

Key Steps:

1. **Data Collection**: Gathering raw data from various sources
2. **Data Preprocessing**: Cleaning and transforming data
3. **Model Selection**: Choosing the right algorithm
4. **Training**: Fitting the model on training data
5. **Testing and Evaluation**: Measuring accuracy on unseen data
6. **Deployment**: Putting the model into real-world use

Overfitting: Model performs well on training data but poorly on testing data. Solution: cross-validation, regularization.

Underfitting: Model performs poorly on both training and testing data. Solution: use more complex models or add features.

2.3 Machine Learning in Practice

Applications:

- Spam detection
- Face recognition
- Credit scoring
- Recommender systems

Challenges:

- Poor data quality
- Data bias
- Model interpretability
- Overfitting/underfitting

Tools/Libraries:

- **Python Libraries**: Pandas, NumPy, Scikit-learn, TensorFlow, Keras

---

2.4 Types of Machine Learning Algorithms

1. **Supervised Learning:**

2. Labeled data

3. Tasks: Classification (e.g., spam detection), Regression (e.g., price prediction)

4. Algorithms: Linear Regression, Decision Tree, SVM, KNN

5. **Unsupervised Learning:**

6. No labels

7. Tasks: Clustering (e.g., customer segmentation), Association (e.g., market basket)

8. Algorithms: K-Means, DBSCAN, Apriori

9. **Semi-Supervised Learning:**

10. Small labeled + large unlabeled data

11. Useful when labeling is expensive

12. **Reinforcement Learning:**

13. Agent learns by interacting with environment

14. Uses rewards and penalties
15. Example: Game AI, robotics

LAB EXERCISES

1. Data Exploration

Pandas for Exploration:

```
import pandas as pd

# Load data
df = pd.read_csv('data.csv')

# Basic exploration
print(df.head())
print(df.info())
print(df.describe())
print(df.isnull().sum())
```

2. Preprocessing with Normalization

Handling Missing Values:

```
# Fill missing numeric values
df['salary'].fillna(df['salary'].median(), inplace=True)
```

```python
# Drop rows with missing values
df.dropna(inplace=True)
```

Normalization & Standardization:

```python
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Min-Max Normalization
scaler = MinMaxScaler()
df[['age', 'salary']] = scaler.fit_transform(df[['age', 'salary']])

# Standardization
std = StandardScaler()
df[['age']] = std.fit_transform(df[['age']])
```

Encoding Categorical Variables:

```python
# Label Encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['gender'] = le.fit_transform(df['gender'])

# One-Hot Encoding
df = pd.get_dummies(df, columns=['city'])
```

Distance Computation Example:

```python
from scipy.spatial.distance import euclidean, cityblock

# Euclidean Distance
e1 = euclidean([1, 2], [4, 6])

# Manhattan Distance
m1 = cityblock([1, 2], [4, 6])
```

End of Notes