KATHMANDU UNIVERSITY

Department of Artificial Intelligence

Dhulikhel, Kavre

Lab Report On

Data Preparation

Submitted By:

Name: Aaryan Shakya

Roll No: 20

Subject Code: [AICC 301]

Submitted To:

Mr. Sunil Regmi

Lecturer, Department of Artificial Intelligence

6th July 2025

Objective

1. To clean raw data by handling missing values, noise, and inconsistencies.
2. To transform data through normalization, encoding, and aggregation techniques.
3. To reduce data size using methods like feature selection and dimensionality reduction.
4. To prepare a final dataset that is suitable and efficient for data mining tasks.

1. Introduction

1.1 Data Cleaning:

Data cleaning is the process of identifying and correcting errors or inconsistencies in raw data to improve its quality and reliability. Data quality problems arise due to misspellings during data entry, missing values, or any other invalid data. It ensures that the dataset is accurate, complete, and ready for analysis or mining.
Missing values are data points that are absent for a specific variable in a dataset. They can be represented in various ways, such as blank cells, null values, or special symbols like "NA" or "unknown." These missing data points pose a significant challenge in data analysis and can lead to inaccurate or biased results. Missing values can Reduce the sample size, introduce bias and can make it difficult to perform certain analyses.

1.1.1 Types of missing values:

1. Missing Completely at Random (MCAR): MCAR is a specific type of missing data in which the probability of a data point being missing is entirely random and independent of any other variable in the dataset. In simpler terms, whether a value is missing or not has nothing to do with the values of other variables or the characteristics of the data point itself.
Example: In a health survey, some participants accidentally skip a question about their age because of a printing error on some paper forms. The missing data has no relationship with age or any other variable.

2. Missing at Random (MAR): MAR is a type of missing data where the probability of a data point missing depends on the values of other variables in the dataset, but not on the missing variable itself. This means that the missingness mechanism is not entirely random, but it can be predicted based on the available information.
Example: In a study on income, younger respondents are less likely to report their income. Here, the missing income data depends on age (an observed variable), but not on the income itself.

3. Missing Not at Random (MNAR): MNAR is the most challenging type of missing data to deal with. It occurs when the probability of a data point being missing is related to the missing value itself. This means that the reason for the missing data is informative and directly associated with the variable that is missing.
Example: In a mental health survey, people with severe depression may choose not to answer

questions about their mental health condition. The reason for missing data is directly related to the severity of depression.

Methods to handle missing data include:

1. **Deletion Techniques:**
   - *Row Deletion:* Entire records (rows) with missing values are removed. This is suitable when the number of such rows is small and unlikely to affect the dataset's integrity.
   - *Column Deletion:* Entire columns with a large percentage of missing values are dropped if they are not crucial to analysis.
2. **Imputation Techniques:**
   - *Mean Imputation:* Missing values are replaced with the average of the non-missing values of that attribute.
   - *Median Imputation:* Uses the median instead of the mean, more robust to outliers.
   - *Mode Imputation:* For categorical attributes, the most frequent category is used to fill missing values.
3. **Advanced Imputation:**
   - *KNN Imputation:* Uses the k-nearest neighbors' values to predict the missing value based on similarity.
   - *Model-Based Imputation:* Builds a predictive model (e.g., linear regression) to estimate missing values using other correlated attributes.

1.1.2 Handling Noisy Data

Noisy data contains errors, outliers, or random variations. It can distort statistical summaries and lead to incorrect conclusions.

Methods to handle noisy data:

1. **Binning:**
   - *Equal-width binning:* Divides data into bins of equal size.
   - *Smoothing by mean/median/boundaries:* Replaces values in a bin with the average (mean), central (median), or boundary values.
2. **Regression:**
   - Fits a mathematical function (e.g., linear regression) to the data and uses the trend line to smooth out fluctuations or remove outliers.

3. **Clustering:**
    - Groups data into clusters. Noise or outliers can be identified as data points that do not belong to any cluster or belong to small clusters.

1.1.3 Handling Inconsistent Data

Inconsistent data includes contradictions, mislabeling, or values in different formats. This reduces data reliability and makes integration difficult.

Techniques to handle inconsistent data:

- **Standardization:**
    - Uniform representation of data values. Example: Converting all temperature values to Celsius, or standardizing date formats (e.g., MM/DD/YYYY).
- **Conflict Resolution:**
    - Resolving naming inconsistencies. For example, unifying "M" and "Male" into a single representation.
- **Domain Constraints and Business Rules:**
    - Apply rules based on domain knowledge. For example, a student's age cannot be negative; if such a value exists, it should be flagged and corrected.
- **Duplicate Removal:**
    - Detecting and merging duplicate records that may have slight variations in format.

1.2 Data Integration

Data integration is the process of combining data from different sources into a unified and coherent view. It ensures that data collected from multiple systems or formats can be analyzed together effectively.

Key techniques and concepts:

- **Schema Integration:** Aligning different data structures (schemas) from various sources to create a consistent format. Example: Merging customer tables from two databases with different attribute names like CustID and Customer_ID.
- **Entity Identification:** Matching and identifying the same real-world entities across datasets. Example: Recognizing that "Jon Smith" in one source is the same as "J. Smith" in another.

- **Data Value Conflicts:** Resolving inconsistencies such as different units or formats. Example: Converting weight from pounds to kilograms or dates from DD/MM/YYYY to MM/DD/YYYY.
- **Redundancy Elimination:** Removing duplicate data that appears in multiple places. This avoids double counting or inflated results.
- **Record Linkage:** Connecting records that refer to the same entity across different datasets using keys or matching algorithms.
- **Data Fusion:** Combining complementary information from different sources to produce more informative records.

1.3 Data Transformation

Data transformation involves converting data into suitable formats or structures for mining. It improves consistency, compatibility, and performance of data analysis.

Major transformation techniques:

- **Normalization:** Scales data to a specific range, usually [0, 1].
    - *Min-Max Normalization:* Rescales data using minimum and maximum values.
      Formula:
      $(X-\min)/(\max-\min)(X - \min) / (\max - \min)(X-\min)/(\max-\min)$
- **Standardization:** Centers data around the mean with unit variance.
    - *Z-score Standardization:*
      Formula:
      $(X-\mu)/\sigma(X - \mu) / \sigma(X-\mu)/\sigma$
      where μ is mean, σ is standard deviation.
- **Encoding:**
    - *Label Encoding:* Assigns unique integers to each category (e.g., Male = 0, Female = 1).
    - *One-Hot Encoding:* Creates binary columns for each category (e.g., Gender_Male, Gender_Female).
- **Smoothing and Aggregation:** Reduces noise or summarizes information by combining data (e.g., averaging daily sales to compute weekly sales).
- **Attribute Construction:** Creating new features from existing ones. Example: Creating BMI from weight and height.

1.4 Data Reduction

Data reduction techniques reduce the size of data while preserving its analytical value. This improves efficiency in storage, computation, and mining.

Common methods:

- **Feature Selection:** Selects the most relevant attributes, removing redundant or irrelevant ones.
    - Techniques: Filter (based on correlation), Wrapper (using model performance), Embedded (e.g., Lasso regression).
- **Principal Component Analysis (PCA):** Reduces dimensionality by transforming original features into a smaller set of uncorrelated components, while retaining most of the variance.
- **Binning:** Groups continuous values into intervals to reduce granularity.
    - *Equal Width Binning:* Intervals of equal range.
    - *Equal Frequency Binning:* Intervals with the same number of data points.
- **Clustering-Based Reduction:** Replaces a group of similar data points with a representative (like cluster centroids).
- **Sampling:** Selects a subset of data for analysis. Can be random, stratified, or systematic.

1.5 Discretization

Discretization is the process of converting continuous numerical attributes into categorical values. It simplifies data and makes it suitable for algorithms that require categorical inputs.

Common discretization techniques:

- **Equal Width Binning:** Divides the range of values into equal intervals. Simple but may not distribute data evenly.
- **Equal Frequency Binning:** Each bin has approximately the same number of data points. Handles skewed data better than equal width.
- **K-Means Clustering:** Uses unsupervised learning to find natural groupings and assign cluster labels as categories.
- **Decision Tree-Based Discretization:** Splits data based on entropy or Gini index. Creates bins that best separate the target classes.

- **Manual Discretization:** Based on domain knowledge. Example: Classifying age as "child", "adult", and "senior".

Discretization is especially useful in classification tasks and rule mining, where categorical data leads to better model interpretability.

1.6 Concept Hierarchy

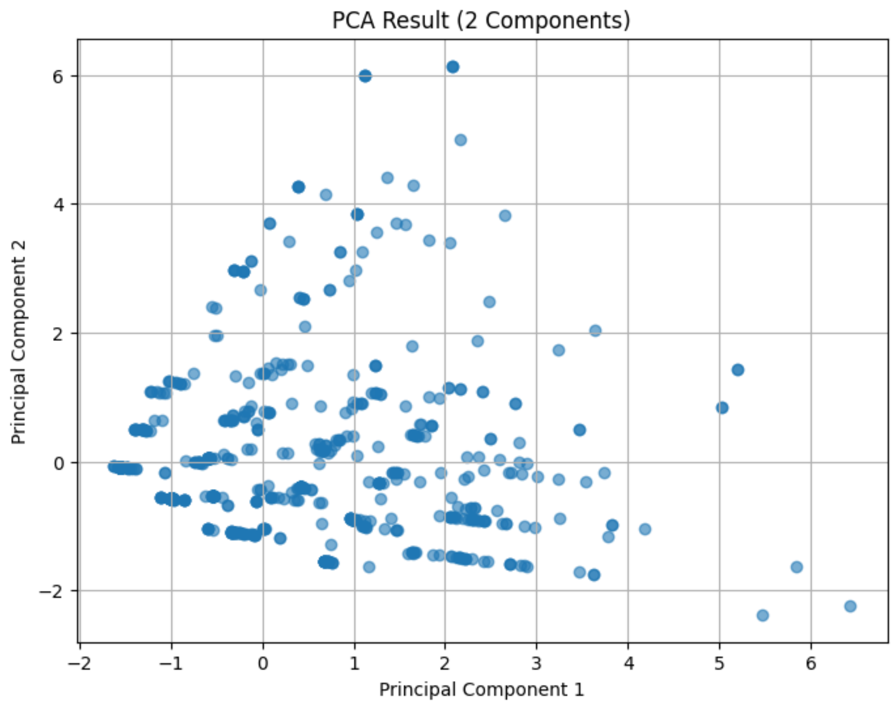
Concept hierarchy organizes data attributes into multiple levels of abstraction, from detailed values to broader categories. It helps simplify and generalize data, making it easier to analyze and discover patterns at different granularity levels. For example, grouping countries into regions or days into months. Concept hierarchies are useful in data preprocessing steps like data generalization and aggregation.

2. Experiment

Load a dataset with missing and noisy values and apply imputation, feature encoding, normalization or standardization, and PCA for dimensionality reduction.

```
# Check missing values
print(titanic.isnull().sum())
# Imputation Techniques for Missing Values
# Impute 'age' with median (numerical feature)
titanic['age']=titanic['age'].fillna(titanic['age'].median(), inplace=True)

# Impute 'embarked' with mode (categorical feature)
titanic['embarked']=titanic['embarked'].fillna(titanic['embarked'].mode()[0], inplace=True)

# Drop columns with too many missing values (e.g., 'deck')
titanic['deck']=titanic.drop(columns=['deck'], inplace=True)

# Feature encoding
# label encoding for binary categorical feature(sex)
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
titanic['sex'] = le.fit_transform(titanic['sex'])  # male = 1, female = 0

 #One-Hot Encoding for Multiclass Categorical Features (embarked, class, who)

titanic = pd.get_dummies(titanic, columns=['embarked', 'class', 'who'], drop_first=True)

# normalization
from sklearn.preprocessing import StandardScaler

# Select numeric features only
num_cols = titanic.select_dtypes(include=['float64', 'int64']).columns

# Apply standardization
scaler = StandardScaler()
titanic[num_cols] = scaler.fit_transform(titanic[num_cols])

from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA

# Drop non-numeric or identifier columns you don't want
df_pca = titanic.drop(columns=['alive', 'embark_town'], errors='ignore')
```

```
# Optional: select only numeric columns explicitly (this avoids non-numeric errors)
df_pca = df_pca.select_dtypes(include=['number'])

# Impute missing values with median
imputer = SimpleImputer(strategy='median')
df_pca_imputed = imputer.fit_transform(df_pca)

# Apply PCA to reduce to 2 components
pca = PCA(n_components=2)
pca_result = pca.fit_transform(df_pca_imputed)

# Convert to DataFrame
pca_df = pd.DataFrame(pca_result, columns=['PC1', 'PC2'])
print(pca_df.head())

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.scatter(pca_df['PC1'], pca_df['PC2'], alpha=0.6)
plt.title('PCA Result (2 Components)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)
plt.show()
```



3. Conclusion

In this lab, we successfully applied essential data preparation techniques to clean, transform, and reduce raw data for effective mining. Handling missing and noisy values through imputation and smoothing improved data quality, while feature encoding and normalization standardized the dataset for analysis. Finally, dimensionality reduction using PCA helped simplify the data by reducing its complexity without significant loss of information. These preprocessing steps are crucial for enhancing the accuracy and efficiency of subsequent data mining tasks.