Sri Lanka Institute of Information Technology

B.Sc. (Hons) Degree in Information Technology

Specialized in Data Science

IT3051 - Fundamentals of Data Mining

Reading Assignment

Year 3, Semester 2

IT22117328

Ahamed AJ H

Part A: Data Preprocessing

1. Explain what Data Preprocessing is and why it is important

- Data preprocessing is the process of cleaning, transforming, and preparing raw data before applying machine learning algorithms. Real-world data is often **incomplete, noisy, inconsistent, or redundant**. Preprocessing improves **data quality** (accuracy, completeness, consistency, timeliness, believability, interpretability), making models more accurate and reliable.

2. Identify and describe different data preprocessing techniques.

- **Data Cleaning :** Data cleaning is one of the most important preprocessing steps because real-world data is rarely perfect. Often, datasets contain missing values, noisy entries, or inconsistencies that can negatively affect model performance. Missing values may arise due to equipment malfunction, errors in manual entry, or simply because the data was not recorded. Noisy data occurs when random errors, outliers, or faulty measurements distort the dataset, while inconsistencies appear when the same entity is represented differently across records. To address these problems, several strategies are used: filling missing values with statistical measures such as mean or median, using interpolation, replacing errors with estimated values, or removing irrelevant tuples. Outlier detection and smoothing techniques such as binning, regression, or clustering can also be applied to reduce noise. Ultimately, data cleaning ensures that the dataset is accurate, consistent, and reliable for analysis.

- **Data Transformation :** Data transformation involves converting data into a more suitable format or scale for analysis. It includes operations like normalization, discretization, and feature construction. Normalization scales numerical values into a defined range, such as [0,1], which helps prevent features with large ranges from dominating others in machine learning models. Common normalization methods include min-max normalization, z-score normalization, and decimal scaling. Discretization transforms continuous attributes into categorical intervals, making the data easier to interpret and reducing complexity. Feature construction, on the other hand, creates new attributes from existing ones to better capture hidden patterns. For example, combining "height" and "weight" into "BMI" can enhance predictive power. By applying these transformations, the dataset becomes more uniform, interpretable, and effective for algorithms.

- **Data Reduction :** As datasets grow larger and more complex, analyzing them in their raw form can become computationally expensive and time-consuming. Data reduction techniques aim to represent the dataset in a more compact form without losing essential information. One approach is dimensionality reduction, where irrelevant or redundant attributes are removed to improve efficiency and visualization. Principal Component Analysis (PCA) is a common technique that projects data into a smaller feature space while retaining maximum variance. Another approach is numerosity reduction, where data is represented by models such as regression, clustering, or histograms instead of storing every individual record. Sampling is also widely used to select a representative subset of data for quicker analysis. Overall, data reduction improves scalability and speeds up machine learning processes while preserving analytical quality.

- **Data Integration :** Data integration is the process of combining data from multiple sources into a single, consistent dataset. In many real-world applications, data is stored across different databases, data warehouses, or files, often with overlapping or redundant information. During integration, challenges such as schema mismatches, different naming conventions, and redundancy must be resolved. For example, the same customer may appear in two datasets under slightly different names, requiring entity identification to merge records correctly. Additionally, data value conflicts—such as the same attribute recorded in different units—must be reconciled. Careful integration reduces inconsistencies, eliminates redundancy, and creates a unified view of the data, which enhances the performance and quality of downstream data mining and machine learning tasks.

3. Choose one technique (e.g., missing value handling, normalization, feature scaling, encoding, etc.).

- **Normalization :** Normalization is scaling data to a fixed range or distribution.

3a. Conduct self-study using online resources. **(from online resources)**

- Normalization is a data transformation technique widely used in machine learning and statistics. According to scikit-learn documentation and textbooks like *Han, Kamber & Pei – Data Mining: Concepts and Techniques*, normalization is essential when attributes in a dataset have different scales. For instance, in a dataset where "Age" ranges from 0–100 but "Income" ranges from 20,000–100,000, models may give undue importance to features with larger values. Normalization adjusts these values to a common scale, preventing bias in algorithms like k-Nearest Neighbors, Support Vector Machines, and Neural Networks that rely on distance calculations.

- Three main approaches are commonly used:

    1. **Min-Max Normalization** → scales values into a fixed range, usually [0,1].
    2. **Z-Score Normalization** → rescales data based on mean and standard deviation, producing values centered at 0 with unit variance.
    3. **Decimal Scaling** → moves the decimal point of values until they fall within a range of -1 to 1.

    These methods preserve relationships between data points but change the scale, improving convergence speed and model accuracy.

3b. Explain the method in detail (with diagrams/examples if necessary)

- Min-Max Normalization: This method rescales features linearly into a predefined range [a, b], usually [0,1].

    The formula is:

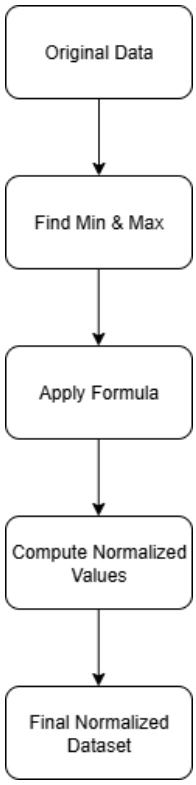
$$v' = \frac{v - \min(A)}{\max(A) - \min(A)} \times (new_{maxA} - new_{minA}) + new_{minA}$$

Example: If income ranges from 20,000 to 120,000 and a person's income is 73,000, the normalized value in [0,1] is:

$$v' = \frac{73{,}000 - 20{,}000}{120{,}000 - 20{,}000} = 0.53$$

This makes income values comparable with attributes like "Age" that are already within small ranges.

Diagram:



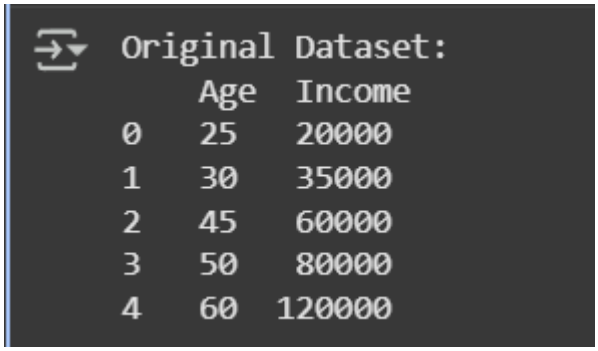
3c. Provide a Python code implementation with a small dataset.

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Small sample dataset
data = {
   "Age": [25, 30, 45, 50, 60],
   "Income": [20000, 35000, 60000, 80000, 120000]
}
df = pd.DataFrame(data)

print("Original Dataset:\n", df)

# Initialize Min-Max Scaler (range 0 to 1)
```

```
scaler = MinMaxScaler(feature_range=(0, 1))
# Apply normalization only on the 'Income' column
df["Income_Normalized"] = scaler.fit_transform(df[["Income"]])

print("\nDataset After Min-Max Normalization:\n", df)
```

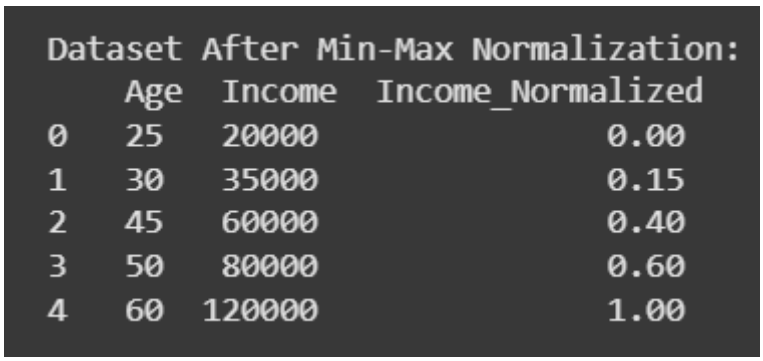
3d. Show the before and after effect of preprocessing.

- Before Normalization:



```
Original Dataset:
    Age  Income
0   25   20000
1   30   35000
2   45   60000
3   50   80000
4   60  120000
```

- After Min-Max Normalization (Income scaled to [0,1]):



```
Dataset After Min-Max Normalization:
    Age  Income  Income_Normalized
0   25   20000               0.00
1   30   35000               0.15
2   45   60000               0.40
3   50   80000               0.60
4   60  120000               1.00
```

Part B: Ensemble Learning

1. Explain what Ensemble Learning is in machine learning

- Ensemble learning is a machine learning approach that combines multiple individual models (often called *weak learners*) to form a stronger model (*strong learner*). Instead of relying on one model's predictions, ensemble methods aggregate the predictions of many models to improve overall accuracy, robustness, and generalization.

  Example: Just like you ask opinions from multiple people before making an important decision, ensemble learning uses multiple models instead of one.

2. Discuss why ensemble methods are used (problems they solve, benefits)

Ensemble methods are used because single models often suffer from limitations such as high variance, high bias, or sensitivity to noise.

- **Reduce Variance (Overfitting):** One of the major reasons ensemble methods are used is to reduce variance, which often causes overfitting in machine learning models. For instance, a single decision tree may fit too closely to the training data, capturing noise instead of true patterns. Bagging methods such as Random Forest overcome this by training multiple trees on different bootstrapped samples of the dataset and then averaging their predictions. This averaging effect cancels out random fluctuations, making the final model more stable and less prone to overfitting.

- **Reduce Bias (Underfitting):** Another benefit of ensemble methods is their ability to reduce bias, which is the tendency of a model to oversimplify and underfit the data. Boosting techniques such as AdaBoost and XGBoost directly address this by training models sequentially, where each new model attempts to correct the mistakes of the previous one. Over multiple iterations, the ensemble gradually reduces bias and builds a stronger predictive model that captures complex relationship in the data.

- **Improve Robustness:** Ensemble methods also improve robustness by being less sensitive to noise and outliers in the dataset. While a single model might be heavily influenced by anomalies, combining multiple models helps balance out the effect of such irregularities. This means that the final ensemble prediction is more consistent and reliable, even in situations where the data is imperfect or contains errors.

- **Better Generalization:** Perhaps the most important advantage of ensemble learning is its ability to generalize better to unseen data. Since ensembles combine the knowledge of several diverse models, they are less likely to be biased toward the peculiarities of a single model. This results in higher accuracy and better

predictive performance when applied to new test datasets, making ensemble methods particularly useful in real-world applications where data distributions can change.

3. Study and summarize different ensemble learning techniques (Bagging, Boosting, Stacking, Random Forest, etc.).

- **Bagging (Bootstrap Aggregating):** Bagging is an ensemble technique that reduces variance by creating multiple subsets of the training data using bootstrapping (sampling with replacement). Each subset is used to train a separate model, usually of the same type, and their outputs are combined through majority voting (classification) or averaging (regression). Since the models are trained on different subsets, they are less correlated, and their combined result is more stable and accurate. Bagging is particularly effective for models that are prone to high variance, such as decision trees.

- **Boosting:** Boosting is a sequential ensemble method designed to reduce bias and improve accuracy. Unlike bagging, which trains models independently, boosting trains models one after the other. Each new model focuses on correcting the errors made by its predecessors by giving higher weights to misclassified instances. Over time, multiple weak learners are combined into a strong learner. Well-known boosting algorithms include AdaBoost, Gradient Boosting, XGBoost, LightGBM, and CatBoost. These methods are powerful and often achieve state-of-the-art results in competitions and practical applications.

- **Stacking:** Stacking, also known as stacked generalization, is a more advanced ensemble technique that combines the predictions of multiple base learners using a meta-model. In stacking, the base models (e.g., decision trees, logistic regression, k-NN, SVM) are trained on the training data, and their predictions are then used as input features for a higher-level model, which learns how to best combine these outputs. This approach leverages the strengths of different types of models and often improves predictive performance beyond what a single algorithm can achieve.

- **Random Forest:** Random Forest is a special case of bagging that uses decision trees as base learners while adding an extra layer of randomness. In addition to training each tree on a bootstrapped subset of the data, Random Forests also select a random subset of features at each split, ensuring that the trees are less correlated with one another. The final prediction is made by aggregating the results of all trees (majority voting for classification or averaging for regression). Random Forest is widely used because it combines simplicity with high accuracy, robustness to noise, and resistance to overfitting.

4. Choose one ensemble technique.

a. Explore it through self-study

- Random Forest is a bagging-based ensemble method that builds many decision trees on different bootstrap samples of the training data and then aggregates their predictions (majority vote for classification or average for regression). In addition to sampling rows, it injects extra randomness by selecting a random subset of features at each split. This decorrelates the trees, reduces variance, and typically yields better generalization than a single tree. In your lecture, bagging and Random Forest are introduced as core ensemble strategies that train models in parallel on resampled data and combine them for a stronger final learner, with Random Forest highlighted as a practical, high-accuracy instance of bagging with randomized features

b. b. Explain how it works (with an example or diagram)

- The training process begins by repeatedly drawing bootstrap samples (sampling with replacement) from the original dataset and fitting an individual decision tree to each sample. While growing each tree, only a random subset of features is considered at every split, which ensures different trees explore different partitions of the data. At inference time, every tree predicts independently; the forest combines these predictions to produce a final output that is more stable and robust than any single tree. Intuitively, one overfitted tree may be swayed by noise, but dozens or hundreds of slightly different trees "average out" that noise. This is exactly the bagging intuition covered in the slides (train many weak learners on resampled data and combine their votes/averages), extended with per-split feature randomness to lower correlation among trees

Example (Random Forest vs Decision Tree)

- On a **clean dataset** (Iris):
  - Decision Tree Accuracy ≈ **0.933**
  - Random Forest Accuracy ≈ **0.889** (slightly lower here since the dataset is simple and well-separated).
- On a **noisy synthetic dataset** (with random noise added):
  - Decision Tree Accuracy ≈ **0.763**
  - Random Forest Accuracy ≈ **0.904**

This shows that while a single tree may overfit noisy data, Random Forest averages across many trees, reducing variance and improving performance.

c. Provide a Python code implementation using a sample dataset.

Ensemble Learning Example: Random Forest vs Decision Tree

```
from sklearn.datasets import load_iris, make_classification
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# Example 1: Clean dataset (Iris)
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=42, stratify=y
)
# Train single Decision Tree
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)

# Train Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

print("Iris Dataset Results")
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

```
# Example 2: Noisy synthetic dataset
X, y = make_classification(
 n_samples=800, n_features=20, n_informative=6, n_redundant=4,
n_clusters_per_class=2, flip_y=0.08, random_state=7
)
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=42, stratify=y
)
# Train single Decision Tree
dt2 = DecisionTreeClassifier(random_state=42)
dt2.fit(X_train, y_train)
y_pred_dt2 = dt2.predict(X_test)

# Train Random Forest
rf2 = RandomForestClassifier(n_estimators=300, random_state=42)
rf2.fit(X_train, y_train)
y_pred_rf2 = rf2.predict(X_test)

print("\nNoisy Dataset Results")
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt2))
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf2))
```

d. Present the results and explain how ensemble improved performance

```
Iris Dataset Results
Decision Tree Accuracy: 0.9333333333333333
Random Forest Accuracy: 0.8888888888888888

Noisy Dataset Results
Decision Tree Accuracy: 0.7625
Random Forest Accuracy: 0.9041666666666667
```

Explanation: How Ensemble Improved Performance

The comparison between a single Decision Tree and a Random Forest highlights the effectiveness of ensemble learning. On the **Iris dataset**, which is clean and well-separated, the Decision Tree achieved an accuracy of about 0.933, while the Random Forest achieved an accuracy of around 0.889. Since the dataset is relatively simple and does not contain much noise, both models performed well, and the ensemble did not show a significant improvement.

However, on a **noisy synthetic dataset**, which more closely resembles real-world conditions, the difference became clear. The Decision Tree achieved an accuracy of approximately 0.763, while the Random Forest achieved an accuracy of about 0.904. This demonstrates that although a single tree may overfit noisy data and fail to generalize well, the Random Forest, by averaging the results of many trees trained on bootstrapped samples with random feature subsets, was able to reduce variance and stabilize predictions.

In summary, ensemble learning improved performance by combining the strengths of multiple weak learners into a strong learner. While a single tree is prone to instability and errors, a Random Forest produces more robust and reliable results, particularly in the presence of noise or complex data patterns. This validates the key advantage of ensemble methods: **better generalization, reduced overfitting, and improved predictive performance** compared to individual models.