KATHMANDU UNIVERSITY

Department of Artificial Intelligence

Dhulikhel, Kavre

Lab Report On: Supervised Data Mining Methods

Submitted By:

Name: Aaryan Shakya

Roll No: 20

Subject Code: [AICC 301]

Submitted To:

Mr. Sunil Regmi

Lecturer, Department of Artificial Intelligence

Submission Date:

6th July 2025

Objective

1. Understand the concepts of classification and prediction in supervised learning.
2. Implement and compare supervised learning algorithms such as Decision Trees, KNN, and SVM.
3. Evaluate model performance using metrics like accuracy, precision, and cross-validation techniques.

Introduction

Supervised Learning Overview

Supervised learning is one of the main branches of machine learning where a model is trained on a labeled dataset, meaning that each input data point is associated with a known output (label). The learning algorithm maps inputs (features) to outputs (targets) by identifying patterns in the training data. The ultimate goal is to accurately predict the output for new, unseen data.

There are two primary types of supervised learning tasks:

- **Classification**, where the output is categorical (e.g., "spam" or "not spam").
- **Prediction (Regression)**, where the output is a continuous value (e.g., predicting house prices).

Supervised learning is widely used in real-world applications such as fraud detection, recommendation systems, medical diagnosis, and forecasting.

Classification

Classification refers to the task of assigning a predefined label or class to input data based on its features. The output variable in classification is discrete, meaning it belongs to a fixed number of categories. The learning algorithm is trained using a dataset where each instance is labeled with the correct class, and the model learns to generalize from these examples.

Advantages:

- Straightforward interpretation (especially with models like Decision Trees).
- Efficient algorithms exist for both small and large datasets.
- Can be easily evaluated using various metrics (accuracy, F1-score, etc.).

Use Cases:

- **Spam Detection:** Classifying emails as spam or not spam.
- **Disease Diagnosis:** Determining if a patient has a disease based on symptoms and lab results.

- **Sentiment Analysis:** Identifying whether a customer review is positive, negative, or neutral.

Example:

A decision tree can classify emails based on keywords like "offer", "free", or "urgent" to determine whether they are spam or not.

Prediction (Regression)

Prediction tasks, commonly referred to as regression problems, involve estimating a continuous numeric value based on one or more input features. The output variable is not a class, but a real number, such as temperature, income, or sales.

Advantages:

- Simple models like linear regression are fast and interpretable.
- Provides insights into how features influence the target variable.
- Suitable for forecasting and trend analysis.

Use Cases:

- **House Price Estimation:** Predicting the price of a house based on features like area, location, and number of rooms.
- **Stock Price Forecasting:** Estimating future stock values from historical trends.
- **Customer Lifetime Value Prediction:** Estimating how much a customer will spend in the future.

Example:

Using linear regression to predict the final exam score of students based on hours studied and attendance percentage.

Common Supervised Learning Algorithms

1. Decision Trees

A decision tree is a flowchart-like structure in which each internal node represents a decision based on an attribute, each branch represents an outcome, and each leaf node represents a class label. They are easy to visualize and interpret.

- **Advantages:**
    - Easy to understand and explain.

    - Can handle both categorical and numerical data.
    - Performs well on small to medium-sized datasets.
- **Use Case:**
  Credit risk assessment—approving or rejecting loan applications based on income, credit score, and employment.

2. Random Forest

Random Forest is an ensemble method that builds multiple decision trees and merges their results (usually via majority vote for classification or average for regression) to improve accuracy and control overfitting.

- **Advantages:**
    - High accuracy and robustness.
    - Handles missing data well.
    - Works well with high-dimensional datasets.
- **Use Case:**
  Customer churn prediction—identifying which customers are likely to stop using a service.

3. Naive Bayes (Bayesian Networks)

Naive Bayes is a probabilistic classifier based on Bayes' Theorem. It assumes independence among features, which simplifies computation but still provides strong results, especially in text classification.

- **Advantages:**
    - Extremely fast and scalable.
    - Performs well on large and high-dimensional datasets.
    - Works surprisingly well even with the "naive" assumption.
- **Use Case:**
  Spam filtering in email systems or document categorization.

4. K-Nearest Neighbors (KNN)

KNN is a lazy learning algorithm that stores the training data and classifies new data points by finding the 'k' most similar examples. Classification is done by majority voting.

- **Advantages:**
    - Simple and effective.
    - No training phase required.
    - Adaptable to multi-class problems.
- **Use Case:**
  Image recognition—classifying hand-written digits by comparing with labeled examples.

5. Support Vector Machine (SVM)

SVM finds the optimal hyperplane that separates classes with maximum margin. It's effective in high-dimensional spaces and can be used for both classification and regression.

- **Advantages:**
    - Works well for complex and high-dimensional data.
    - Effective for non-linear problems using kernel trick.
    - Resistant to overfitting.
- **Use Case:**
  Face recognition and bioinformatics (e.g., classifying genes).

6. Artificial Neural Networks (ANN)

ANNs are inspired by biological neural networks. They consist of layers of interconnected "neurons" that process data through weighted connections and activation functions. ANNs are powerful for modeling complex, non-linear relationships.

- **Advantages:**
    - Can model extremely complex patterns.
    - Highly flexible and capable of handling both structured and unstructured data.
    - Basis for deep learning techniques.
- **Use Case:**
  Speech recognition, natural language processing, and image classification.

7. Linear and Logistic Regression

- **Linear Regression:**
  Used for predicting a numeric (continuous) value. It models the relationship between the dependent variable and one or more independent variables by fitting a straight line.
- **Logistic Regression:**
  Used for binary classification problems. It models the probability of an event occurring using a logistic (sigmoid) function.
- **Use Case:**
  - *Linear:* Predicting employee salaries based on experience and education.
  - *Logistic:* Predicting whether a customer will purchase a product or not (Yes/No).

Model Evaluation Techniques

Evaluating a supervised model is essential to understand how well it performs and how generalizable it is to new data.

1. Training vs Testing

The dataset is typically divided into a **training set** (to build the model) and a **testing set** (to evaluate its performance). This helps prevent overfitting and provides an unbiased estimate of model accuracy.

2. Cross-Validation

Cross-validation, especially **k-fold cross-validation**, involves splitting the dataset into *k* parts. The model is trained on *k-1* parts and tested on the remaining part. This is repeated *k* times and the average performance is considered.

3. Holdout Method

A simple method where a portion of the data (e.g., 70%) is used for training and the rest for testing.

4. Bootstrap

Sampling the dataset with replacement to create multiple training sets, helping estimate the model's variance and bias.

Performance Metrics

- **Accuracy:**
  The ratio of correctly predicted instances to the total instances. Best for balanced datasets.

- **Precision:**
  The ratio of true positives to all predicted positives. High precision means fewer false positives.
- **Recall (Sensitivity):**
  The ratio of true positives to all actual positives. High recall means fewer false negatives.
- **F1-score:**
  The harmonic mean of precision and recall. Useful when data is imbalanced.
- **ROC Curve (Receiver Operating Characteristic):**
  A graphical plot of the true positive rate vs false positive rate. The area under the ROC curve (AUC) indicates model performance.
- **Precision-Recall Curve:**
  More informative than ROC when dealing with highly imbalanced datasets.
- **Loss Functions:**
  - *MSE (Mean Squared Error):* Measures the average squared difference between predicted and actual values in regression.
  - *Log Loss (Cross-Entropy):* Evaluates the accuracy of a classification model by penalizing false predictions.

Experiment

Implementation of Classification Algorithms Using Scikit-learn on Titanic Dataset

Problem:

1. Load dataset (e.g., Titanic or Iris).

2. Implement classification algorithms using scikit-learn.

3. Train, test and evaluate using metrics and visualizations.

Solution:

Step 1: Load the Dataset
I've used the Iris dataset (for classification)

Step 2: Implement Classification Algorithms using Scikit-learn

- Imported necessary libraries and datasets.
- Preprocessed data (handling missing values, encoding categorical features).
- Split data into training and testing sets.
- Trained models: Decision Tree, Random Forest, KNN, Logistic Regression, SVM, and ANN.
- Evaluated models using accuracy, precision, recall, F1-score, and ROC curves.

Python Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, label_binarize
from sklearn.metrics import classification_report, roc_curve, auc, accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB  # Bayesian Network approx.
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier

# Load dataset
iris = load_iris()
X, y = iris.data, iris.target
```

```
# Binarize labels for ROC (one-vs-rest)
y_bin = label_binarize(y, classes=[0, 1, 2])
n_classes = y_bin.shape[1]

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Scale features for applicable models
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize models
models = {
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'Random Forest': RandomForestClassifier(random_state=42),
    'Bayesian Network (Naive Bayes)': GaussianNB(),
    'K-Nearest Neighbors': KNeighborsClassifier(n_neighbors=3),
    'Logistic Regression': LogisticRegression(max_iter=200, random_state=42),
    'Support Vector Machine': SVC(probability=True, random_state=42),
    'Artificial Neural Network': MLPClassifier(max_iter=500, random_state=42)
}

# Train, predict, evaluate
for name, model in models.items():
    print(f"\n--- {name} ---")
    # Use scaled data for models sensitive to feature scale
    if name in ['Logistic Regression', 'Support Vector Machine', 'Artificial Neural Network']:
        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)
        y_score = model.predict_proba(X_test_scaled)
    else:
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        y_score = model.predict_proba(X_test)

    # Print classification report
    print(classification_report(y_test, y_pred, digits=4))
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")

    # Compute ROC curve and AUC for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_bin[y_test.index if hasattr(y_test, 'index') else
np.arange(len(y_test)), i], y_score[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    # Plot ROC curves
    plt.figure()
    for i in range(n_classes):
        plt.plot(fpr[i], tpr[i], label=f'Class {i} (AUC = {roc_auc[i]:.2f})')
    plt.plot([0, 1], [0, 1], 'k--', label='Random guess')
    plt.title(f'ROC Curve - {name}')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend(loc='lower right')
    plt.show()
```

Conclusion

In this experiment, we successfully implemented supervised classification algorithms using the Scikit-learn library on well-known datasets such as Titanic and Iris. We explored the entire machine learning workflow—starting from loading and preprocessing the data, selecting appropriate models, training and testing the classifiers, and finally evaluating their performance using various metrics such as accuracy, precision, recall, and F1-score.

Additionally, we visualized the results through confusion matrices and decision boundaries, which helped us better understand model behavior and classification accuracy. This practical implementation demonstrated the strengths and trade-offs of different algorithms like Decision Trees, K-Nearest Neighbors, and Logistic Regression in solving real-world classification problems.

Overall, the lab reinforced key concepts of supervised learning and highlighted the importance of model evaluation and visualization in machine learning.