SRM Institute of Science and Technology
Delhi – Meerut Road, Sikri Kalan, Ghaziabad, Uttar Pradesh – 201204
Department of Computer Applications
Circular – 2023-24
BCA DS 6th Sem

MACHINE LEARNING FOR ENTERPRISE (UDS23D03J)

Lab Manual

Lab 1: Machine Learning Approaches

Title: Exploration of Machine Learning Approaches

Aim: To introduce and explore fundamental concepts and different approaches in machine learning.

Procedure:

1. Research and study the main categories of machine learning: Supervised Learning, Unsupervised Learning, and Reinforcement Learning.
2. For each category, identify and describe at least two common algorithms or techniques (e.g., Linear Regression and Decision Trees for Supervised Learning).
3. Discuss the types of problems each category is suited for (e.g., classification, regression, clustering).
4. Provide a simple example scenario for each category.

Source Code: (This lab is primarily conceptual, so source code would be example code snippets for specific algorithms within the approaches.)

```
# Example: Linear Regression (Supervised Learning)
import numpy as np
from sklearn.linear_model import LinearRegression

# Sample data
X = np.array([[1], [2], [3], [4], [5]])  # Input feature
y = np.array([2, 4, 5, 4, 5])           # Output variable

# Create and train the model
model = LinearRegression()
model.fit(X, y)

# Predict a new value
x_new = np.array([[6]])
y_pred = model.predict(x_new)
print(f"Prediction for x = 6: {y_pred[0]:.2f}")
```

Input: N/A (Conceptual Lab) / For the example code: The input is the training data X and y, and the new data point x_new for prediction.

Expected Output: (For the example code)

```
Prediction for x = 6: 6.00
```

(Conceptual parts of the lab will have descriptive/textual output)

Lab 2: Python Code for Binary Class Classification

Title: Binary Class Classification with Python

Aim: To implement a binary class classification model using Python and a suitable library (e.g., scikit-learn).

Procedure:

1. Select a suitable dataset for binary classification (e.g., the Iris dataset for a specific pair of classes, or a synthetic dataset).
2. Choose a classification algorithm (e.g., Logistic Regression, Support Vector Machine).
3. Split the dataset into training and testing sets.
4. Implement the chosen algorithm using Python (e.g., with scikit-learn).
5. Train the model on the training data.
6. Evaluate the model's performance on the testing data (e.g., using accuracy, precision, recall, F1-score).

Source Code: (Python code using scikit-learn or similar)

Input: A binary classification dataset.

Expected Output: Classification results, including performance metrics.

Lab 3: Perform K-Means Clustering Algorithm

Title: K-Means Clustering Implementation

Aim: To implement and apply the K-Means clustering algorithm to a given dataset.

Procedure:

1. Select a dataset for clustering (e.g., a customer segmentation dataset, or a synthetic dataset).
2. Choose a value for K (the number of clusters).
3. Implement the K-Means algorithm in Python (e.g., using scikit-learn).
4. Visualize the data points and the resulting clusters.
5. Experiment with different values of K and observe the effect on the clustering results.

Source Code: (Python code using scikit-learn)

Input: A dataset for clustering.

Expected Output: Cluster assignments for each data point, visualization of the clusters.

Lab 4: Demonstrate Markov Decision Processes

Title: Markov Decision Processes (MDP) Demonstration

Aim: To understand and demonstrate the concept of Markov Decision Processes.

Procedure:

1. Define a simple environment with states, actions, rewards, and transition probabilities.
2. Represent the MDP mathematically (states, actions, transition matrix, reward function).
3. Implement a method to solve the MDP (e.g., Value Iteration, Policy Iteration).
4. Show how an agent can interact with the environment and make decisions based on the MDP.

Source Code: (Python code to define and solve the MDP)

Input: Definition of the MDP (states, actions, etc.).

Expected Output: Optimal policy, value function.

Lab 5: Steps Involved in Data Preprocessing, Feature Engineering Best Practices

Title: Data Preprocessing and Feature Engineering

Aim: To learn and apply data preprocessing techniques and feature engineering best practices.

Procedure:

1. Obtain a dataset with missing values and various data types.
2. Apply appropriate data cleaning techniques (e.g., handling missing values, removing outliers).
3. Perform data transformation (e.g., scaling, normalization).
4. Engineer new features from existing ones (e.g., polynomial features, interaction terms).
5. Discuss the importance of each step and best practices.

Source Code: (Python code using pandas and scikit-learn)

Input: A dataset requiring preprocessing.

Expected Output: Cleaned and transformed dataset with engineered features.

Lab 6: Decision Tree Regression

Title: Decision Tree Regression

Aim: To implement and apply Decision Tree Regression.

Procedure:

1. Select a regression dataset.
2. Split the dataset into training and testing sets.
3. Implement Decision Tree Regression.
4. Train the model.
5. Evaluate performance (e.g., MSE, R-squared).

Source Code: (Python)

Input: Regression dataset.

Expected Output: Regression predictions, performance metrics.

Lab 7: Gaussian Naïve Bayes

Title: Gaussian Naïve Bayes Classification

Aim: To implement and apply the Gaussian Naïve Bayes algorithm.

Procedure:

1. Select a classification dataset.
2. Split the data.
3. Implement Gaussian Naïve Bayes.
4. Train and evaluate.

Source Code: (Python)

Input: Classification dataset.

Expected Output: Classification predictions, metrics.

Lab 8: Demonstrate Clustering Problems, Collaborative Filtering

Title: Clustering and Collaborative Filtering

Aim: To demonstrate clustering and collaborative filtering techniques.

Procedure:

1. Perform Clustering (K-Means)
2. Implement Collaborative Filtering (User-User or Item-Item)

Source Code: (Python)

Input: Dataset for clustering and collaborative filtering.

Expected Output: Clusters, recommendations.

Lab 9: Reinforcement Learning

Title: Introduction to Reinforcement Learning

Aim: To implement a basic reinforcement learning algorithm.

Procedure:

1. Define a simple environment (e.g., a grid world).
2. Implement a reinforcement learning algorithm (e.g., Q-learning).
3. Train an agent to interact with the environment.

Source Code: (Python)

Input: Definition of the environment.

Expected Output: Trained agent, learned policy.

Lab 10: Application of Reinforcement Learning Real-World Example

Title: Reinforcement Learning in a Real-World Example

Aim: To explore a real-world application of reinforcement learning.

Procedure:

1. Choose a real-world application (e.g., game playing, robotics).
2. Research how reinforcement learning is applied in that domain.
3. Simulate a simplified version of the application.

Source Code: (Python)

Input: (Depends on the chosen application)

Expected Output: Simulation results, demonstration of the RL application.

Lab 11: Demonstrate Learning Agent

Title: Learning Agent Implementation

Aim: To implement a basic learning agent.

Procedure:

1. Define an environment for the agent.
2. Implement an agent that can perceive the environment and take actions.
3. Incorporate a learning mechanism (e.g., a simple update rule).

Source Code: (Python)

Input: Environment definition.

Expected Output: Agent behavior, learning progress.

Lab 12: Bagging and Boosting Algorithms

Title: Ensemble Methods: Bagging and Boosting

Aim: To implement and compare bagging and boosting algorithms.

Procedure:

1. Select a classification or regression dataset.
2. Implement Bagging (e.g., Random Forest) and Boosting (e.g., AdaBoost, Gradient Boosting).
3. Train and evaluate the models.
4. Compare the performance of bagging and boosting.

Source Code: (Python)

Input: Dataset for classification or regression.

Expected Output: Model predictions, performance comparison.

Lab 13: Demonstration of AutoML Classification

Title: Automated Machine Learning (AutoML) for Classification

Aim: To demonstrate the use of an AutoML tool for classification tasks.

Procedure:

1. Select a classification dataset.
2. Choose an AutoML tool (e.g., scikit-learn's Auto-sklearn, or a cloud-based AutoML service).
3. Use the AutoML tool to search for the best classification model.
4. Evaluate the performance of the best model found by AutoML.

Source Code: (Python, using an AutoML library)

Input: Classification dataset.

Expected Output: Best model found by AutoML, its performance.

Lab 14: Data Pipeline

Title: Data Pipeline Implementation

Aim: To build a data pipeline for a machine learning task.

Procedure:

1. Select a dataset and a machine learning task.
2. Design a pipeline that includes data loading, preprocessing, feature engineering, model training, and evaluation.
3. Implement the pipeline using a suitable library (e.g., scikit-learn's Pipeline).

Source Code: (Python)

Input: Dataset.

Expected Output: Results of the machine learning task, demonstration of the pipeline.

Lab 15: Data Visualization

Title: Data Visualization Techniques

Aim: To apply data visualization techniques to explore and present data.

Procedure:

1. Select a dataset.
2. Choose appropriate visualization techniques (e.g., scatter plots, histograms, box plots) to explore the data.
3. Use a visualization library (e.g., Matplotlib, Seaborn) to create the visualizations.
4. Visualize the data.

Source Code: (Python)

Input: Dataset.

Expected Output: Various data visualizations.