DASC 41103
Machine Learning
Project 1
Foundations of Classification Algorithms

Total Points: 200

Lesson Objective Alignment:

- Design and implement machine learning models using Python.
    - Develop machine learning algorithms for practical applications
    - Understand and implement perceptron and Adaline algorithms.
    - Apply logistic regression and support vector machines using scikit-learn.
- Evaluate model performance and decision boundaries.
- Work with real-world datasets and apply feature preprocessing.
- Communicate machine learning principles and methods to diverse audiences.

Project Objective: Demonstrate ability to apply simple machine learning classification algorithms in Python.

Description: For this project, students will work in **pairs (groups of 2)** to complete each part, provide requested deliverables, and answer any questions.

Data: Use the provided data, which comes from the **UCI Adult Income dataset** (also known as the Census Income dataset) to predict whether a person earns more than $50K/year.

- project_adult.csv
- project_validation_inputs.csv

You can learn more about the data variables by going to https://archive.ics.uci.edu/dataset/2/adult.

Parts:

1. Preprocess the dataset: **project_adult.csv** & **project_validation_inputs.csv**
    a. Handle missing values.
    b. Encode categorical features.
    c. Standardize numerical features.
2. Implement the **Perceptron** and **Adaline** algorithms
    a. Train Perceptron and Adline models (at least AdalineSGD).
    b. Plot the number of misclassifications (Perceptron) and MSE (Adaline) over epochs.
    c. Find the accuracy of your best models from both algorithms
    d. Use best performing models to predict outputs for **project_validation_inputs**.

    e. Use provided code to implement scikit-learn's Perceptron and Adaline algorithms. Find the accuracy on the validation data using your best models for both.
3. Implement Logistic Regression and SVM using scikit-learn
    a. Train Logistic Regression and SVM models using scikit-learn.
    b. Find the accuracy of your best models from both algorithms
    c. Use best performing models to predict outputs for **project_validation_inputs**.
    d. Select 2 features and visualize the decision boundaries.
4. Reflection and Conceptual Questions
    a. Why is feature scaling important for gradient-based algorithms?
    b. Explain the difference between batch gradient descent and stochastic gradient descent.
    c. Why does scikit-learn Perceptron and Adline algorithms outperform book code?
        i. **Research and develop an informed answer**.
    d. Compare the decision boundaries of logistic regression and SVM.
    e. What is the role of regularization in preventing overfitting?
    f. Vary the C values of the scikit-learn LogisticRegression and linear SVC models with [0.01, 1.0, 100.0]. **Discuss the impact**.

Deliverables:
Students should submit a video recording of their presentation, their slides, the predicted values of their model on the validation inputs, and a link to their open GitHub repository via blackboard.

- 15-minute presentation that at a high-level discussing what you did in parts 1 – 3 and the answers to questions in part 4 in more depth.
- 4 validation output files. This should be the predictive values your models made on project_validation_inputs. We will use these files to determine the accuracy based on the actual outputs. **This should be the scikit-learn implementations of each.** You should use the following names:
    - Group_#_Perceptron_PredictedOutputs.csv
    - Group_#_Adaline_PredictedOutputs.csv
    - Group_#_LogisticRegression_PredictedOutputs.csv
    - Group_#_SVM_PredictedOutputs.csv
- Linke to your group's open GitHub repository with code that is clean and well-commented.

Grading:
All team members will receive the same grade unless a team member requests otherwise. Grades will be based on the grading rubric below.

Project Grading Rubric

Group Number:	Points
1. **Presentation:**	
Project Discussion: **30 points**: A clear, concise, and high-level summary of the methodology, challenges, and results for Parts 1, 2, and 3. Demonstrates a strong understanding of the project's practical steps. **15 points**: Provides a partial summary of the project parts. Some key steps or findings are missing or unclear. **0 points**: The project overview is missing or lacks a coherent narrative.	30
Conceptual Depth: **30 points**: Offers a comprehensive and articulate explanation of the concepts from Part 4, demonstrating a deep understanding of algorithms and theory. **15 points**: Provides a basic but incomplete explanation of the concepts. Lacks depth or contains some inaccuracies. **0 points**: The conceptual answers are missing or incorrect.	30
Presentation Quality: **20 points**: The presentation is well-structured, professional, and within the 15-minute time limit. Visuals on the slides are clean and easy to follow. Audio and video quality are excellent. **10 points**: The presentation is somewhat disorganized or exceeds the time limit. Some visuals are cluttered, or the audio/video quality is poor. **0 points**: The presentation is incomprehensible, significantly exceeds the time limit, or was not submitted.	20
2. **Code and Repository**	
Code Functionality: **30 points**: All code runs without errors, producing the expected outputs and plots. The implementations of Perceptron and Adaline are correct. **15 points**: The code runs with minor errors or does not fully complete all tasks. **0 points**: The code fails to run or contains significant errors, making it unusable.	30
Code Quality and Comments: **30 points**: The code is clean, logically organized, and highly readable. It includes comprehensive and helpful comments explaining key functions, logical blocks, and complex steps. Variable names are descriptive. **15 points**: The code is functional but could be cleaner. Comments are sparse or do not adequately explain the logic. **0 points**: The code is unreadable, not commented, or disorganized.	30
GitHub Repository: **20 points**: The repository is public, contains all required code and a README file, and is easily accessible via the link provided. The commit history is logical. **10 points**: The repository is missing a README, is not public, or is difficult to navigate. **0 points**: The GitHub link is broken, or the repository is not submitted.	20

3. Model Performance and Deliverable	
Validation Output File Naming & Submission: **10 points**: All four .csv files are submitted to Blackboard with the correct and exact file names as specified in the assignment. **5 points**: Files are submitted but some have incorrect names or are missing. **0 points**: No files are submitted.	10
Model Accuracy on Validation Set: **30 points**: Your models achieve an acceptable level of accuracy on the provided validation set, demonstrating a robust implementation and an effective approach to feature preprocessing and model training. **15 points**: Your models achieve a moderate level of accuracy. **0 points**: Your models achieve a low level of accuracy, indicating fundamental errors in the implementation or training process.	30
Total	