ACM SIG AI - SEM 2 Recruitment Tasks

Task 0: Complete the following courses

To ensure you have the foundational knowledge needed for the upcoming tasks, please complete the following courses:

1. **Kaggle: Intro to Machine Learning** - Kaggle Course
2. **Kaggle: Supervised Machine Learning** - Kaggle Supervised Learning
3. **Kaggle: Unsupervised Learning** - Kaggle Unsupervised Learning
4. **Scikit-Learn Official Documentation** - Scikit-Learn Docs

Note: It's highly recommended that you complete these courses before proceeding with the tasks.

Task 1: Implement a Regression Model (Building from Scratch)

Objective: Select a dataset that requires substantial preprocessing, and implement a Linear Regression model from scratch. You are required to handle preprocessing techniques and build an end-to-end workflow. This task evaluates your understanding of regression, data preprocessing, and model evaluation.

1. **Dataset**: Choose a dataset that requires extensive preprocessing.
   - Example: House prices, medical data, etc.
   - You should apply techniques like handling missing values, feature scaling, encoding categorical variables, etc.
2. **Preprocessing**: Perform necessary preprocessing steps:
   - Handle missing data.
   - Normalize/standardize features if necessary.
   - Visualize data and check for correlations using scatter plots, etc.
3. **Model**: Implement Linear Regression from scratch (do not use any inbuilt libraries like scikit-learn for the model).
   - Use basic linear algebra to perform calculations.

   - Implement a simple gradient descent approach to optimize the model.
4. **Evaluation**:
   - Evaluate your model using **Mean Squared Error (MSE)** and **Root Mean Squared Error (RMSE)**.
   - Perform a comparison between **Linear Regression** and **Polynomial Regression**. Plot the performance differences and analyze them.

Task 2: Dealing with Images (Classification)

Objective: Work with image data to build a classification model. You will explore image preprocessing, feature extraction, and model evaluation using different classifiers.

1. **Dataset**: Choose a small-scale image classification dataset (e.g., MNIST, CIFAR-10, Fashion-MNIST).
   - Download one of the mentioned datasets from Kaggle or other repositories.
2. **Preprocessing**: Perform image preprocessing:
   - Normalize the pixel values (e.g., to the range [0, 1]).
   - Resize images to a fixed shape (if required).
   - Optionally, perform augmentation (e.g., rotations, flips).
3. **Model**: Select a classification algorithm:
   - Start with a basic classifier such as **Logistic Regression**, **SVM**, or **Decision Trees**.
   - Optionally, implement a **simple Neural Network** if you are familiar with it.
4. **Evaluation**:
   - Measure model performance using **accuracy**, **precision**, **recall**, and **F1-score**.
   - Generate and analyze a **confusion matrix**.
   - Visualize misclassified examples and correct ones.

Task 3: Text Analysis (Unsupervised Learning)

Objective: Work with text data to perform unsupervised learning through clustering. You will preprocess the text, extract features, and apply a clustering algorithm.

1. **Dataset**: Choose a small text dataset (e.g., product reviews, news headlines, or a collection of short articles).
   - A sample dataset could be found on Kaggle or any public text repository.
2. **Preprocessing**:
   - Convert all text to lowercase.
   - Remove punctuation and special characters.
   - Optionally, remove stop words and apply stemming/lemmatization.
3. **Model**:
   - Use **TF-IDF (Term Frequency-Inverse Document Frequency)** to convert the text into numerical format.
   - Apply **K-Means clustering** to group the text into 2 clusters.
4. **Interpretation**:
   - Identify key terms for each cluster.
   - Explain why these terms belong to the same group.
   - **Bonus**: Calculate cosine similarity to find the two most similar texts in the dataset.

Task 4: Dealing with PyTorch

Objective: Understand basic PyTorch operations like tensor manipulation, matrix operations, and automatic differentiation using autograd. Complete the following steps to gain hands-on experience.

1. **Matrix Multiplication and Autograd**:
   - Create two random tensors of shape **3x3**.
   - Perform matrix multiplication between the two tensors.
   - Use PyTorch’s **autograd** to compute the gradient of the result with respect to one of the input tensors.

    - **Explain**: What happens in each step and how autograd computes gradients.

2. **Broadcasting**:

    - Create a **3x1** tensor and a **1x3** tensor.
    - Use broadcasting to add the two tensors together.
    - Multiply the result by another **3x3** tensor.
    - **Explain**: What broadcasting is and how it works in this case.

3. **Reshaping and Slicing**:

    - Create a tensor of shape **(6, 4)** using random values and reshape it to shape **(3, 8)**.
    - After reshaping, extract a slice (e.g., the first two columns, all rows).
    - **Explain**: What reshaping does and how slicing helps to extract specific parts of the tensor.

Submission Guidelines:

1. **Code**: All code should be written in Python. Please use Jupyter notebooks or Python scripts to submit your work.
2. **Documentation**: Provide clear explanations for each step of your process, including how you approached preprocessing, model selection, and evaluation.
3. **Visualization**: Ensure that you include relevant data visualizations (graphs, plots, etc.) where necessary.
4. **GitHub**: All completed tasks must be uploaded to a GitHub repository. The repository should include:
    - Make it **Public** with repo name as **“***<yourname>***_ACM_SIGAI_Recr”**
    - A clear README file explaining how to run the code and the results.
    - Separate folders for each task (Task 1, Task 2, etc.), with respective files clearly named.

Please ensure that the notebook or code is well-commented and easy to understand.
