L04 Exercise: Machine Learning

Complete the following exercise problems and demo the results individually or as a team.
Total score: 120

You are strongly encouraged to utilize a Large Language Model (LLM) (e.g., ChatGPT, CoPilot, Gemini) to assist with this exercise. However, consider the LLM as a learning tool or assistant to help you understand the required subjects and programming techniques. Your goal is to acquire the necessary knowledge and programming techniques, grasp the problem-solving process, and develop the skills to complete this exercise independently. This approach will prepare you to tackle more complex problems in the future.

1. Supervised learning for linear regression [50 points]

(a) Write a Python program "**simple_reg.py**" that

- Loads a training dataset X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

  Y = [2.5, 4.1, 5.6, 7.2, 8.8, 10.3, 11.9, 13.5, 15.0, 16.8]

- Displays a scatter plot for the dataset using "**matplotlib**" package for data visualization.
- Finds the function from the dataset using "**scikit-learn**" package, write the function in the form of a polynomial equation $y = w_0 + w_1x_1 + \cdots + w_nx_n$, and display the predicted y value, $\hat{y}$ when $x = 100$.
- Similarly, loads a training dataset

  X = [-3, -2.5, -2, -1.5, -1, 0, 1, 1.5, 2, 2.5, 3]

  Y = [17.5, 12.9, 9.5, 7.2, 5.8, 5.5, 7.1, 9.7, 13.5, 18.4, 24.4]

  and find the function from the dataset and write the function in the form of a polynomial equation, and display the coefficients of the equation $[w_0, w_1,\ldots,w_n]$ and the predicted y value, $\hat{y}$ when $x = 0.5$.

  The process of finding a function from a dataset is called "**training**" and the entire process for analyzing a dataset and understanding the dataset by finding a function is called "**linear regression**" or "**regression analysis**". This is one of common supervised learning methods. The function learned from the training can be used to predict $\hat{y}$ for any given x value if the function is reasonably accurate.

- Briefly explains how well these two functions fit the datasets from visual inspection of the plots.
- Computes the sum of error (total error) using difference $\Delta = \sum_i (y_i - \hat{y}_i)$ where y_i is the actual value and $\hat{y}_i$ is the predicted value using the learned function and the sum of squared error using the Squared Error $SE = \sum_i (y_i - \hat{y}_i)^2$, the Mean Squared Error $\boldsymbol{MSE} = \frac{\sum_i (y_i - \hat{y}_i)^2}{N}$ where N is the number of examples, and the Rooted Mean Squared Error $\boldsymbol{RMSE} = \sqrt{\frac{\sum_i (y_i - \hat{y}_i)^2}{N}}$, compare the different error values, and explains which formula is better error measure.

(b) Write a Python program "**multiple_reg.py**" that:

- Loads the dataset from "Q2.csv" and find the function, write the polynomial equation, and display the coefficients $[w_0, w_1, \ldots, w_n]$ and the predicted y value, $\hat{y}$ for any randomly chosen x value.
- Computes, display the RMSE of the function, and explain if the function is acceptable in terms of accuracy of prediction.

(c) Write a Python program "**least_sq.py**" that:

- Computes the coefficients using the formula $\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ with the dataset loaded from "Q2.csv". Note that this equation requires matrix transpose, inverse, and multiplication. This is a solution of the optimization problem $argmin_w J(w) = \frac{1}{N}\sum_i (y_i - w^T x_i)^2$ as we wanted the minimum error between the actual value of y in the training dataset loaded from "Q2.csv" and predicted value of y as coefficient vector $\boldsymbol{W}$. Finding $\hat{\boldsymbol{w}}$ using this method is called "**the least square**" method.
- Writes the polynomial equation obtained from the least square method and computes the RMSE of the function, and compares it with the function obtained in (b), and explains which one is more accurate.

2. **Supervised learning for classification** [50 points]

(a) Write a Python program "**decision_tree.py**" that:

- Loads the "Iris flower" dataset directly from "**sklearn.datasets**" package.
- Splits 80% of the original dataset into training dataset and 20% into test dataset (20%).

- Train the decision tree algorithm to classify the data by creating a decision tree.

The "**Iris flower**" dataset (or Fisher's Iris dataset) is a multivariate dataset used for testing the performance of many supervised learning algorithms. Review the dataset at a dataset repository (https://archive.ics.uci.edu/dataset/53/iris/) and more information about the data set (https://en.wikipedia.org/wiki/Iris_flower_data_set/).

- Create a decision tree for the loan application data in the lecture slide and compare the tree from training with the one shown on the slide.

(b) Write a Python program "**knn_iris.py**" that classifies the "**Iris flower**" dataset using the K-nearest neighbor (K-NN) and compare the classification accuracy with the decision tree. Try to change any parameters to improve the accuracy and show the results of improvement.

(c) Write a Python program "**knn_digit.py**" that
Loads the "**digits**" dataset directly from "**sklearn.datasets**" package. This digit dataset is a collection of handwritten digits images (0-9). It consists of 1,797 grayscale images, each 8x8 pixels, where each pixel has an intensity value ranging from 0 (white) – 16 (black). The images are flattened into a 64-dimensional feature vector (8x8=64). The target value (labels, 0-9) represents the actual digit in the image.
Display the **first image** from the dataset using matplotlib.
Classifies the digit dataset using the K-NN and computes the % accuracy.
Modifies any K-NN parameters to improve the accuracy and show the results if an improvement is achieved.

(d) Write a Python program "**mlp_digit.py**" that

- Classifies the digit dataset using a Multilayer Perceptron (**MLP**), a type of Artificial Neural Networks (**ANNs**), computes the % accuracy, and compare the classification accuracy with the K-NN.
- Adjusts any MLP hyperparameters to enhance classification performance and shows the improved results. Some of the hyperparameters include hidden_layer_sizes, activation functions, learning_rate, max_iter, batch_size, momentum.

3. **Unsupervised learning for clustering** [20 points]

(a) Write a Python program "**kmeans.py**" that

- Clusters the "**Iris flower**" dataset using the **K-means** with K=3. Ignore the label column of the dataset for clustering since clustering is an unsupervised learning technique and does not require labels.
- Computes the **RMSE** for each cluster based on its centroids.
- Compares the clustering results from K-means with the classification results from "knn_iris.py", focusing on how well the clusters align with the actual classes in the dataset. This evaluation process, known as **cluster quality validation**, helps determine whether data points within each cluster are similar to those in the actual dataset classes.

What to submit

- A report file in word or PDF format that includes the name or names of the team members, and the % contribution of each member. If there is no agreement on individual contributions, briefly describe the tasks assigned and completed by each member. Different scores may be awarded based on individual contributions. If all members contributed equally, simply state "equal contribution." The report should also include descriptions of the answers (if questions are asked) and a portion of the screenshot demonstrating the functionality of each program.
- Each program file individually. DO NOT submit any zip file as Canvas cannot open them.
- Submit only one report file and the program file(s) for the entire team.

Grading criteria

- Does each program successfully perform the required tasks and generate the correct results?
- Does the report demonstrate that the team has a clear understanding of the relevant concepts, programming techniques, functional requirements? Does it show how the team applied this knowledge to successfully complete the tasks?
- Is the effort put into the project clearly reflected in both the report and the program files?