



## Assignment 2 Decision Trees and K-nn

### Problem 1 [Decision Trees using Scikit-learn][2 experiments]:

The objective is to build a model that can predict the appropriate medication/Drug type (target variable) for patients suffering from a specific illness based on their Age, Sex, Blood Pressure, and Cholesterol levels. [Drag.csv]

Assignment Tasks:

1. **Data Preprocessing:** Perform data preprocessing steps by handling missing data [count how many missing values occurs + **handle** empty cell by your own way] and encoding categorical variables.
2. **First experiment:** Training and Testing with Fixed Train-Test Split Ratio:
  - Divide the dataset into a training set and a testing set (30% of the samples).
  - Repeat this experiment five times with different random splits of the data into training and test sets.
  - Report the sizes and accuracies of the decision trees in each experiment, Compare the results of different models and select the one that achieves the highest overall performance.
3. **Second experiment:** Training and Testing with a Range of Train-Test Split Ratios:  
Consider training set sizes in the range of 30% to 70% (increments of 10%). Start with a training set size of 30% and increase it by 10% until you reach 70%.  
**For each training set size:**
  - Run the experiment with five different random seeds.
  - Calculate the mean, maximum, and minimum accuracy at each training set size.
  - Measure the mean, maximum, and minimum tree size.
  - Store the statistics in a report.
  - Create two plots: one showing accuracy against training set size and another showing the number of nodes in the final tree against training set size.

## Problem 2 [KNN]:

Use the diabetes.csv data to implement your own simple KNN classifier using python, (**Don't use any built-in functions**), divide your data into 70% for training and 30% for testing. [diabetes.csv]

1. Objective:
  - Perform multiple iterations of k (e.g., 5 iterations each different k value ex. K=2,3,4...) on the dataset.
  - Use Euclidean distance for computing distances between instances.
2. Data preprocessing:
  - Normalize each feature column separately for training and test objects using Log Transformation **or** Min-Max Scaling.
3. Break ties using Distance-Weighted Voting:
  - When there is a tie, consider the distances between the test instance and the tied classes' neighbors.
  - Assign higher weights to closer neighbors and use these weights to break the tie, reflecting the idea that closer neighbors might have a stronger influence on the classification decision.
4. Output:
  - **For each iteration**, output the value of k and the following summary information:
    - Number of correctly classified instances.
    - Total number of instances in the test set.
    - Accuracy.
  - At the end of all iterations, output the average accuracy across all iterations.

## Output Example:

k value: 3

Number of correctly classified instances: 418

Total number of instances: 658

Accuracy: 63%

## Min/Max Scaling:

Formula:  $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$

## Deliverables:

You are required to submit ONE zip file containing the following:

- Your code 2(.py) files for each problem.
- A report (.pdf) containing the team members' names and IDs, and the code with screenshots of the output of each part.

The zip file must follow this naming convention: ID1\_ID2\_ID3\_ID4\_ID5\_Group.

## Submission Remarks:

- The maximum number of students in a team is 5 and the minimum is 4.
- Team members must be from the same lab (or have the same TA).
- No late submission is allowed.
- A **penalty** will be imposed for violating any of the assignment rules.
- **Cheaters will get ZERO** and no excuses will be accepted.

## Grading Criteria:

Problem 1 [DT]:	[2]
Data preprocessing	0.5
Experiment with Fixed Training Set size	0.5
Experiment with a range of train_test split ratio	1
Problem 2 [Knn]:	[3]
Knn algorithm (+ handling tie case)	2
preprocessing+ Euclidean distance Calc.	0.5
Experiment with different k values and print summary info as required	0.5