Korea Advanced Institute of Science and Technology
Department of Electrical Engineering & Computer Science

EE331 Introduction to Machine Learning Fall 2025

Assignment Project

Issued: 11/24/2025 (Updated: 12/03/2025),
Due: 12/19/2025

Policy

Group study is encouraged; however, assignment that you hand-in must be of your own work. The assignment will take considerable amount of time so start early.

1 Introduction

In this project, you will design and analyze machine learning models that predict whether a patient is at risk of a heart attack, using a tabular dataset: the *Heart Attack Risk Prediction Dataset*. The dataset consists of 14,309 patient records and 26 attributes (columns), including demographics, clinical measurements (such as cholesterol, blood pressure, heart rate, diabetes), lifestyle factors (smoking, obesity, physical activity, sleep, sedentary time), and socio-economic/geographical indicators (income, country, continent, hemisphere).

You will:

- Propose and implement three different predictive systems under different constraints (best overall performance, memory-efficient, and non-neural).
- Perform error analysis on your best model to understand *why* it fails.
- Apply unsupervised learning (clustering) to visualize and interpret structure in the data.

The grading will be based on a combination of (1) correctness and completeness of your implementation, (2) model performance, and (3) the quality of your analysis and report.

2 Task

2.1 Dataset: Heart Attack Risk Prediction

Source and basic information

- **Name**: Heart Attack Risk Prediction Dataset
- **Source**: Provided via KLMS
- **Samples**: 14,309 patients

- **Columns**: 26 attributes
- **Label**: `Heart Attack Risk` (or `heartAttackRisk`)
  Value 1 = "At risk", Value 0 = "Not at risk"

The features include (but are not limited to) the following categories:

- **Demographic**: Age, Sex
- **Clinical**: Cholesterol, Blood Pressure, Heart Rate, Diabetes, Family History, BMI, Triglycerides
- **Lifestyle**: Smoking, Obesity, Alcohol Consumption, Exercise Hours Per Week, Physical Activity Days Per Week, Sedentary Hours Per Day, Sleep Hours Per Day, Stress Level
- **Socio-economic / Geographic**: Income, Country, Continent, Hemisphere

Download and file format

Download the dataset `heart_attack_dataset_updated.csv` from KLMS and upload it to your Google Drive. In the Colab skeleton[1], you will only need to set the correct path to the CSV file:

```
DATA_PATH = "/content/drive/MyDrive/EE331/heart_attack_dataset_updated.csv"
```

Preprocessing guidelines

You are expected to:

- **Inspect** the dataset: check basic statistics, class balance, and data types.
- **Handle identifiers**: drop purely identifying columns such as `Patient ID`.
- **Handle special features**: split `Blood Pressure` into `Systolic_BP` and `Diastolic_BP`.
- **Handle categorical variables**: encode them using some encoding strategy (e.g., one-hot encoding or label encoding).
- **Split** the data into train/test sets using an 80/20 ratio. You must **fix a random seed (42)** and use stratified splitting to maintain class balance.

2.2 Coding: Model / System Design

You must design and implement **three** systems. You may reuse code across systems, but each system should be clearly defined and evaluated.

(a) High-performance model

Design a model that aims to achieve the best possible predictive performance on this dataset.

[1]https://colab.research.google.com/drive/1JvZs9nVGkv4sEPbMTZNBnYMVg7Yvqrm3?usp=sharing

- You may use any supervised learning algorithm (including neural networks) that is covered in the course or reasonable extensions thereof.
- You must implement the model yourself (e.g., logistic regression, decision tree, neural network) without using scikit-learn model classes. You may reuse your own helper functions across systems.
- You should tune hyperparameters in a principled way (grid search, random search, cross-validation, etc.) and report what you tried.

You must report at least:

- Train, validation, and test accuracy
- At least one additional metric: precision, recall, F1-score, or ROC-AUC

(b) Memory-efficient model

Design a model (or system) that:

- Uses the **least amount of memory** (model size) among your considered models,
- While achieving at least a given minimum accuracy threshold on the validation set:

$$\textbf{Accuracy} \geq A_{\min} = 75\% \text{ on validation set}$$

You must:

- Explain how you **measure memory usage** (e.g., size in KB of the serialized model file).
- Compare at least two candidate models in terms of *both* accuracy and memory. You should compare **your own implementations** of different models (e.g., a small neural network vs. logistic regression) in terms of both accuracy and model size.
- Argue why your chosen model is a good compromise between performance and memory.

(c) Non-neural-network model

Design the best model you can **without using any neural networks**. Allowed model families include, for example:

- Linear models (e.g., logistic regression, linear SVM)
- Probabilistic models (e.g., Naive Bayes)
- Tree-based models (e.g., decision tree)
- k-nearest neighbors

This model can reuse preprocessing code from previous parts but should be clearly reported and evaluated separately. This model must be implemented without any neural network libraries and without any predefined scikit-learn models.

2.3 Error Analysis

Pick your **best** model (you can choose any of the three systems above) and thoroughly analyze its errors. Your error analysis should answer questions such as:

- Which features (or feature combinations) appear frequently among misclassified examples?
- Are there systematic differences between:
  (a) False positives (predicted "at risk" but actually not at risk) and
  (b) False negatives (predicted "not at risk" but actually at risk)?
- Are errors concentrated in specific ranges (e.g., certain age groups, income levels, countries, etc.)?
- Does the model behave differently for different subgroups? (e.g., by sex, continent, or lifestyle factors)

You are encouraged to use:

- Confusion matrices
- Distribution plots (histograms, box plots)
- Feature importance scores (if available for your model)
- Partial dependence plots or similar tools

Finally, propose at least **two concrete ideas** that could reduce errors (e.g., new features, different model type, better regularization, handling class imbalance).

2.4 Clustering-based Visualization

Apply **k-means clustering** to the dataset and visualize the results. You must implement the k-means clustering algorithm yourself (initialization, assignment step, centroid update step). You may use `numpy` for vectorized operations, but you may not use `sklearn.cluster.KMeans` or similar built-in implementations.

- Choose an appropriate subset or transformation of features (e.g., after scaling and encoding).
- Choose a reasonable number of clusters k and justify your choice (you may use the elbow method, silhouette scores, or domain intuition).
- Reduce the feature dimension to 2D or 3D for visualization using PCA.

Your analysis could address:

- What does each cluster roughly correspond to (e.g., "older, high-cholesterol, sedentary group")?
- How does the **heart-attack-risk label** (0/1) distribute across clusters?
- Do some clusters seem clearly high-risk vs. low-risk?

- How do your findings relate to your supervised models?

Include at least one clear 2D (or 3D) plot where:

- Points are colored by cluster index, and
- You optionally overlay label information (e.g., different marker shapes or colors by risk).

3 Report

Your report should be written in clear, concise English and should not exceed **10 pages** (excluding references). A suggested structure is:

1. **Introduction**
   - Brief description of the dataset and problem
   - Your overall goals and what you attempted
2. **Data Understanding and Preprocessing**
   - Summary of the dataset (class balance, key feature distributions)
   - Preprocessing steps: feature selection, encoding, scaling, handling missing values
3. **Model / System Design**
   - For each of the three systems:
     - i Model choice and rationale
     - ii Hyperparameters and tuning strategy
     - iii Final performance (train/val/test)
     - iv Model size (for the memory-efficient part)
4. **Error Analysis**
   - Analysis of misclassifications of your best model
   - Plots/tables illustrating systematic errors
   - Proposed improvements
5. **Clustering and Visualization**
   - Clustering setup (features used, value of k, dimensionality reduction method)
   - Visualizations and interpretation of clusters
   - Relationship between clusters and the heart-attack-risk label
6. **Discussion and Conclusion**
   - Summary of what you learned
   - Limitations of your approach
   - Potential future directions
7. **References**
   - Cite any external libraries, papers, or online resources you used.

4 Submission

You will submit the following files to KLMS:

(a) **Report** (`studentID_name_report.pdf`)
A PDF report following the structure described above.

(b) **Code Notebook** (`studentID_name_code.ipynb`)
A Colab notebook that:

- Runs from top to bottom without errors.
- Downloads/loads the dataset from your Google Drive.
- Performs preprocessing, training, evaluation, error analysis, and clustering.
- Clearly separates the three systems (high-performance, memory-efficient, non-neural).

Your notebook must be self-contained: we will run your notebook in a fresh Colab environment with access to the dataset file (provided via Drive).

5 Grading

The grading for this project will be:

- **40 points** – Report quality and clarity
  (a) Explanation of preprocessing and modeling choices
  (b) Depth of error analysis and clustering interpretation
  (c) Quality of figures/tables and overall writing
  (d) Creativity on problem solving
- **40 points** – Code correctness and completeness
  (a) Correct implementation of three systems
  (b) Proper use of train/validation/test splits
  (c) Reproducible, well-organized code with comments
  (d) If you received any help (from friends, online resources, ChatGPT, etc.), please briefly describe it.
- **20 points** – Model performance relative to peers
  (a) We will compare the test performance of your *best* model.
  (b) You will receive full points for strong performance and partial points for reasonable performance above a baseline threshold.

Exact scoring details may be adjusted slightly and will be announced on KLMS.

Implementation Rules

- You must implement the core machine learning algorithms **by yourself**, using only basic numerical and data libraries such as `numpy` and `pandas`.

- The use of high-level machine learning libraries that provide ready-made models is **not allowed**. In particular, you may **not** use:

  (a) `sklearn.linear_model.LogisticRegression`, `sklearn.cluster.KMeans`, etc.

  (b) other scikit-learn classifier/regressor/clustering model.

- You may use utility functions such as:

  (a) Data handling: `pandas`, `numpy`

  (b) Plotting: `matplotlib`, `seaborn`

  (c) (Optional, if you allow) Convenience utilities such as `sklearn.model_selection.train_test_split`. If you use such utilities, clearly state them in your report.

- If you use neural networks for the high-performance system, you may:

  (a) Either implement a simple neural network by yourself using `numpy`,

  (b) Or use a low-level deep learning framework such as PyTorch for basic layers and backprop, but you may not use pre-trained models.

You may freely reorganize or extend this structure, as long as your notebook remains clear, readable, and easy to run. The skeleton Colab will only import basic libraries such as `numpy`, `pandas`, `matplotlib`, and `seaborn`. All classifiers, regressors, and clustering algorithms must be implemented by you.

Important Notes

- **Dataset Update**: The dataset has been updated from the original Kaggle version to improve data quality and class balance. Please use only the dataset provided on KLMS (`heart_attack_dataset_updated.csv`).
- **Sample Size**: The updated dataset contains 14,309 samples (increased from 8,763) with almost perfect class balance.
- **Performance Expectations**: With the improved dataset, you should expect higher model performance compared to preliminary tests. Tree-based models and ensemble methods may perform particularly well on this dataset.
- **Task 2 Threshold**: The accuracy threshold for the memory-efficient model has been updated to 75% to reflect the improved data quality.