Homework 2: Prompting LLMs for Machine Learning

Due Date: Monday 4:00pm, October 27, 2025

Before we begin, after reviewing the previous homework submissions, it was found that there are meaningful differences between using ChatGPT and gpt-4o-mini. Therefore, for future assignments, it is recommended to use the OpenAI API. As mentioned in the session slides, a balance of $5 should be sufficient for the entire quarter.

1 Introduction

The purpose of this assignment is to familiarize you with the process of building machine learning models by prompting a Large Language Model (LLM). This process is divided into three main parts:

1. Data Exploration
2. Data Preprocessing
3. Model Building and Evaluation

Homework 1 was specifically about Data Exploration. In this homework, we will focus on Data Preprocessing, and Model Building and Evaluation. We will use the same two datasets.

1.1 Data Preprocessing

Goal: make the raw tables ready for a fair model comparison.

- **Split the data** into *train*, *validation*, and *test* sets. Keep the test set unseen until the end.
- **Build a preprocessing pipeline** so that the same steps happen the same way for train/validation/test.
- **Common steps:**
  - **Scaling/standardization** (for features whose size matters)
  - **Encoding categories** (turning text labels into numeric columns)
  - **Simple feature tweaks** (e.g., log transform for highly skewed counts)

How to:

- Put all transforms in a single `Pipeline`/`ColumnTransformer`. Fit them on the *training* split only; apply (transform) to validation/test.
- Keep it simple: only include steps that help your models learn from the data more fairly (not more complicated).

1.2 Model Building and Evaluation

Goal: try a few reasonable models, pick good settings, and evaluate the performance.

Minimal loop:

1. **Choose a model family** (e.g., decision tree, k-nearest neighbors, linear/logistic).

2. **Connect it to your preprocessing** with a single `Pipeline` so training and inference use identical steps.
3. **Tune simple settings** ("hyperparameters") using the *validation* split.
4. **Evaluate** on train/validation during tuning. When satisfied, check test metrics using the *test* split.

Good practice:

- Keep a fixed `random_state` for reproducibility.
- Do not look at or tune on the test set. Use it once at the end.

2 Background Concepts

2.1 Scikit-learn

Scikit-learn is one of the most widely used Python libraries for machine learning. It provides tools for preprocessing, training, evaluating, and tuning models. It includes implementations of algorithms for classification, regression, clustering, and dimensionality reduction. Its consistent API makes it simple to experiment with and compare models.

2.2 Classification Models

- **Decision Tree Classifier:** Splits data based on feature values until reaching predictions. Easy to interpret and can handle categorical and numerical data.
- **K-Nearest Neighbors (KNN) Classifier:** Classifies a sample by majority vote among its k nearest neighbors. Effective for small datasets but computationally expensive for large ones.
- **Logistic Regression Classifier:** A linear model that predicts class membership probabilities using the logistic (sigmoid) function. Despite its name, it is used for classification tasks.

2.3 Regression Models

- **Linear Regression Model:** Fits a straight line (or hyperplane) to minimize errors between predictions and actual values. Assumes a linear relationship between features and the target.
- **K-Neighbors Regression Model:** Predicts values by averaging the outputs of the k nearest neighbors. Flexible but sensitive to noise and computationally expensive on large datasets.

3 Datasets

We will make use of two datasets, same as homework 1.

3.1 California Housing Prices

This dataset is used to create a regression model that predicts the median house price within a block based on a number of features. It contains 20640 rows and 10 columns. The columns are described in Table 1.

Column Name	**Data Type**	**Description**
longitude	float	A measure of how far west a house is; a higher value is farther west
latitude	float	A measure of how far north a house is; a higher value is farther north
housing_median_age	float	Median age of houses within a block; a lower number is a newer building

total_rooms	float	Total number of rooms within a block
total_bedrooms	float	Total number of bedrooms within a block
population	float	Total number of people residing within a block
households	float	Total number of households, a group of people residing within a home unit, for a block
median_income	float	Median income for households within a block of houses (measured in tens of thousands of US Dollars)
median_house_value (target value)	float	Median house value for households within a block (measured in US Dollars)
ocean_proximity	string	Location of the house w.r.t. ocean/sea

Table 1: Columns in the California Housing Prices dataset.

3.2 Dry Bean Dataset

This dataset is used to create a multi-class classification model that predicts the type of dry beans based on 16 features (12 dimensions and 4 shape forms). The columns are described in Table 2.

Column Name	**Data Type**	**Denote**	**Description**
Area	integer	A	The area of a bean's zone and the number of pixels within its boundaries.
Perimeter	float	P	Bean circumference is defined as the length of its border.
MajorAxisLength	float	L	The distance between the ends of the longest line that can be drawn in a bean.
MinorAxisLength	float	l	The longest line that can be drawn in the bean while standing perpendicular to the main axis.
AspectRatio	float	K	Defines the relationship between MajorAxisLength and MinorAxisLength. $$K = \frac{L}{l}$$
Eccentricity	float	Ec	Eccentricity of the ellipse having the same moments as the region.
ConvexArea	integer	C	Number of pixels in the smallest convex polygon that can contain the area of a bean.
EquivDiameter	float	Ed	Equivalent diameter. The diameter of a circle having the same area as a bean seed area. $$d = \sqrt{\frac{4 * A}{\pi}}$$
Extent	float	Ex	The ratio of the pixels in the bounding box to the bean area.
Solidity	float	S	Also known as convexity. The ratio of the pixels in the convex shell to those found in beans. $$S = \frac{A}{C}$$

Roundness	float	R	Calculated with the following formula: $$R = \frac{4\pi A}{P^2}$$.
Compactness	float	CO	Measures the roundness of an object. $$CO = \frac{Ed}{L}$$
ShapeFactor1	float	SF1	Defines the relationship between MajorAxisLength and Area. $$SF1 = \frac{L}{A}$$
ShapeFactor2	float	SF2	Defines the relationship between MinorAxisLength and Area. $$SF2 = \frac{l}{A}$$
ShapeFactor3	float	SF3	Defines the relationship between Area and area of a circle having the same MajorAxisLength as the bean. $$SF3 = \frac{A}{(\frac{L}{2} * \frac{L}{2} * \pi)}$$
ShapeFactor4	float	SF4	Defines the relationship between Area and area of a circle having the same MinorAxisLength as the bean. $$SF4 = \frac{A}{(\frac{L}{2} * \frac{l}{2} * \pi)}$$
Class (target value)	string		Type of a bean. Value of class variable is one of: BARBUNYA, BOMBAY, CALI, DERMASON, HOROZ, SEKER, SIRA.

Table 2: Columns in the Dry Bean Dataset.

Table 3 contains some descriptions of each class, which might help us determine the features that are important for their classification.

Bean Type	**Description**
BARBUNYA	Beige-colored background with red stripes or variegated, speckled color, its seeds are large, physical shape is oval close to the round.
BOMBAY	It is white in color, its seeds are very big and its physical structure is oval and bulging.
CALI	It is white in color, its seeds are slightly plump and slightly larger than dry beans and in shape of kidney.
DERMASON	This type of dry beans, which are fuller flat, is white in color and one end is round and the other ends are round.

HOROZ	Dry beans of this type are long, cylindrical, white in color and generally medium in size.
SEKER	Large seeds, white in color, physical shape is round.
SIRA	Its seeds are small, white in color, physical structure is flat, one end is flat, and the other end is round.

Table 3: Description of each bean type.

4 Tasks

4.1 Homework Overview

- The designated LLM is **gpt-4o-mini** (no token limit).
- There are 2 questions in total. You will have **three submission attempts per question**. Additional attempts will not be graded, and the best score will be counted.
- Submit all prompts (`.txt` files) via **Gradescope**. Each question has a separate submission slot.
- File naming is strict. Use the exact file names, dataset paths, and dictionary keys provided.
- ‘‘‘`python` will be automatically removed by the autograder.

4.2 Question 1

Question: Using the Dry Bean dataset, train a multi-class classifier to predict `Class`. Split the data into training, validation, and test sets; apply the following preprocessing; then train and compare reasonable models (e.g., Decision Tree, KNN, Logistic Regression).

- Standardize: `AspectRatio`, `Eccentricity`, `Roundness`, `ShapeFactor1`, `ShapeFactor2`, `ShapeFactor3`.
- Apply a log transform to `Area`, then standardize it.

Use a single end-to-end `Pipeline`/`ColumnTransformer` so the same steps are applied to train/validation/test without leakage. Finally, print a Python dictionary with keys `train`, `valid`, and `test` whose values are the corresponding accuracies (float). Explore diverse models and hyperparameters, and select the best ones to fulfill the grading criteria.

Instructions:

- Use pandas, numpy, scikit-learn only.
- Dataset path: `/autograder/source/drybean.csv`
- Submit prompt as: `hw2_q1_prompt.txt`

Example Output:

```
{’train’: 0.9477, ’valid’: 0.8510, ’test’: 0.8414}
```

Grading:

- Training accuracy ≥ 0.9 – 1 point
- Validation accuracy ≥ 0.85 – 1 point
- Test accuracy ≥ 0.8 – 2 points

4.3 Question 2

Question: Using the California Housing dataset, train a regression model to predict `median_house_value`. Split the data into training, validation, and test sets; implement the preprocessing below; then train and compare reasonable models (e.g., Linear Regression, KNN Regression).

- **Custom transformer (nearest-anchor distance):** Create a transformer that, for each row's `(latitude, longitude)`, computes the Euclidean distances to the three anchors $(37.38, -122.21)$ — Bay Area; $(33.99, -118.50)$ — Los Angeles; $(32.82, -117.31)$ — San Diego, then outputs a *single* feature equal to the distance to the **nearest** anchor (i.e., the minimum of the three).

Apply the following preprocessing (use a single end-to-end `Pipeline/ColumnTransformer` fit on the training split only to avoid leakage), and set the column transformer's `remainder='drop'` so only the specified outputs are kept:

- `latitude`, `longitude`: custom transformer (nearest-anchor distance) $\rightarrow$ standardization
- `total_rooms`: KNN imputer $\rightarrow$ log transform $\rightarrow$ standardization
- `housing_median_age`, `median_income`: KNN imputer $\rightarrow$ standardization
- `ocean_proximity`: one-hot encoding

Evaluate using Mean Absolute Error (MAE) and print a Python dictionary with keys `train`, `valid`, and `test` whose values are the corresponding MAE scores (floats). Explore diverse models and hyperparameters, and select the best ones to fulfill the grading criteria.

Instructions:

- Use pandas, numpy, scikit-learn only.
- Dataset path: `/autograder/source/housing.csv`
- Submit prompt as: `hw2_q2_prompt.txt`

Example Output:

```
{'train': 45332.13, 'valid': 46753.60, 'test': 45379.38}
```

Grading:

- Training error $\leq$ 46,000 – 2 points
- Validation error $\leq$ 47,000 – 2 points
- Test error $\leq$ 48,000 – 2 points

4.4 What to turn in

Submit your prompts in format (.txt):

- hw2_q1_prompt.txt
- hw2_q2_prompt.txt