Homework 1: Prompting LLMs for Code Generation

Due Date: Sunday, October 12, 2025

1 Introduction

The purpose of this assignment is to familiarize you with the process of prompting a Large Language Model (LLM) for Python code generation. There are two goals that you want to achieve with this homework:

- Data Exploration
- Prompt Engineering

1.1 Data Exploration

In the next homework, we will build machine learning models with two datasets. Before that, let's explore two datasets. Understanding the dataset is essential before applying machine learning models. We will use the `pandas` library to load the dataset into a DataFrame, which is similar to a spreadsheet. Exploration often starts with examining the dataset structure (rows, columns, data types) and continues with investigating missing values, distributions, and correlations.

1.2 Prompt Engineering

Prompt engineering is the practice of designing prompts to generate consistent, reliable outputs from LLMs. Since different LLMs (e.g., Gemini, Claude, ChatGPT) may behave differently, prompts must be explicit, unambiguous, and structured to minimize the fluctuation of the behavior.

Key guidelines:

- **Be explicit.** For example, write "load `sample.csv` as a pandas DataFrame" rather than "load the sample dataset."
- **Reduce ambiguity.** For example, specify exact train/validation/test split percentages instead of vague instructions.
- **Use consistent terminology.** For example, use one term (e.g., "dataset") rather than alternating between dataset, file, or DataFrame.
- **Add structure.** Break tasks into ordered steps, separating goals, instructions, and output format.
- **Include additional instructions.** For example: "Only return Python code. Do not include "python." or "Return a dictionary in the format: `{'name': '...', 'ID': '...'}`."

Prompt engineering is especially important when designing prompts that must perform consistently multiple times.

2 Datasets

We will make use of two datasets:

- **California Housing Prices**[1] - a dataset for building a regression model to predicts the median house price within a block.
- **Dry Bean Dataset**[2] - a dataset for building a multi-class classification model that predicts the type of dry beans based on 16 features (12 dimensions and 4 shape forms).

Now, let's take a deeper dive into the individual datasets.

2.1 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. The dataset used is the "California Housing Prices" dataset which is available here. Here is a summary of the Kaggle description of the dataset:

> This dataset contains information about houses in California districts derived from the 1990 California census. While it may not be suitable for predicting current housing prices, it serves as a great introduction for building machine learning models due to its need for basic data cleaning, a clear and easily comprehensible list of features, and an optimal size that strikes a balance between being overly simplistic and excessively complex.

The dataset 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.

2.2 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 dataset used is the "Dry Bean Dataset" which is available here. The UCI dataset description of the dataset reads:

> Seven different types of dry beans were used in this research, taking into account the features such as form, shape, type, and structure by the market situation. A computer vision system was developed to distinguish seven different registered varieties of dry beans with similar features in order to obtain uniform seed classification. For the classification model, images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. Bean images obtained by computer vision system were subjected to segmentation and feature extraction stages, and a total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains.

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. $$Ex = \frac{A}{A_B} where A_B = AreaOfBoundingRectangle$$
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.

3 Tasks

3.1 Homework Overview

- The LLM used by TAs, is **gpt-4o-mini** (no token limit).
- There are total 2 questions. You will have **two 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 and dataset paths provided. Incorrect names will result in lost points.

3.2 Example Walkthrough

To illustrate the workflow, here are one simple, one complex example.

3.2.1 Sample 1

In this sample, generated code itself will be graded.

Question:

Write Python code that reads the csv file into a DataFrame. Then, store the shape of the DataFrame in a variable called `answer_1`.

Instructions:

- Use the pandas library.
- Dataset path: `/autograder/source/diabetes.csv`
- Submit prompt as: `hw1_sample1_prompt.txt`

Prompt (`hw1_sample1_prompt.txt`)**:**

```
You are an expert at writing a Python code.

Follow the instructions below to generate a Python code.

  1. Import pandas as pd.

  2. Load the csv file '/autograder/source/diabetes.csv' into a DataFrame called df.

  3. Store the shape of df in a variable called answer_1.

Do not include any explanations or comments, only return the Python code.

Do not include ```python.
```

Expected Output:

```
import pandas as pd
df = pd.read_csv('/autograder/source/diabetes.csv')
answer_1 = df.shape
```

3.2.2 Sample 2

In this sample, generated code's output will be graded.

Question:

Write Python code to build a binary classification model. Print a dictionary keys `train`, `valid`, and `test` and their accuracy values.

Instructions:

- Use pandas, numpy, scikit-learn library.
- Dataset path: `/autograder/source/diabetes.csv`
- Submit prompt as: `hw1_sample2_prompt.txt`

Prompt (`hw1_sample2_prompt.txt`)**:**

```
You are an expert at writing a Python code.

Follow the instructions below to generate a Python code.

  1. Use pandas, numpy, scikit-learn libraries.
```

2. Load the csv file '/autograder/source/diabetes.csv'as a pandas dataframe.
3. Set label = 'Diabetes_binary', and features as all columns except label.
4. Split the data into training and testing sets.
5. Split the training set into training and validation sets.
6. Train a Decision Tree model, with 'max_depth' = 8.
7. Get accuracy scores of a model on the training, validation, and testing sets.
8. Print a dictionary with keys train, valid, and test and their corresponding accuracy scores as values.

Do not include any explanations or comments, only return the Python code.

Do not include ```python.

Expected Output:

```
{'train': 0.7477, 'valid': 0.7400, 'test': 0.7414}
```

3.3 Question 1

Typical data exploration involves checking first few rows, column names, the number of rows and columns, and the number of unique values of each column. Let's explore the Dry Bean dataset.

Question:

Write Python code that reads the csv file into a DataFrame. Store the first 5 rows of the DataFrame in a variable called `answer_1`. Store a list of column names of the DataFrame in a variable called `answer_2`. Store the shape of the DataFrame in a variable called `answer_3`. Store the number of unique values of each column in a variable called `answer_4`.

Instructions:

- Use the pandas library.
- Dataset path: `/autograder/source/drybean.csv`
- Submit prompt as: `hw1_q1_prompt.txt`

3.4 Question 2

Unlike Dry Bean dataset, Housing dataset includes missing values, which can cause errors or bias in machine learning model training. Common missing value handling strategies include: (1) Dropping rows with missing values (2) Filling missing values with the median (3) Using K-Nearest Neighbor imputation.

Question: Identify a column with missing values. Find the median of that column. Using KNN imputation (neighbors=10), determine the imputed value at row 291. Print a Python list: [column name (string), median value (float), imputed value (float)].

Instructions:

- Use pandas, numpy, scikit-learn library.
- Dataset path: `/autograder/source/housing.csv`
- Submit prompt as: `hw1_q2_prompt.txt`

3.5 What to turn in

Submit your prompts in format (.txt):

- hw1_q1_prompt.txt
- hw1_q2_prompt.txt

References

[1] California Housing Prices. Kaggle. https://www.kaggle.com/datasets/camnugent/california-housing-prices/data.

[2] Dry Bean Dataset. UCI Machine Learning Repository, 2020. DOI: https://doi.org/10.24432/C50S4B.