# Project Description: Regression and Classification on Restaurant Orders Dataset

## Overview

In this assignment, you will revisit the pre-processed "Restaurant Orders" dataset from Homework #1 and apply regression techniques to uncover relationships between different variables (features) in the dataset. The assignment is divided into three main parts:  
   
1. Single-Feature Linear Regression  
2. Multiple Linear Regression (3 Features)  
3. Polynomial Regression  
4. Binary Classification  
5. Multiple-Classes Classification

## 1. Single-Feature Linear Regression

Objective:  
- Select one dependent variable (output) and one independent variable (feature) from the restaurant dataset.  
- Train a simple linear regression model to predict the output from the single chosen feature.  
  
Deliverables:  
- A plot with data points and the regression line.  
- A short write-up explaining your training procedure, final parameters, final loss, and observations.  
  
Tasks/Steps:  
1. Data Selection: Justify which single feature and output you chose.  
2. Implementation in PyTorch:  
 - Initialize model parameters.  
 - Implement forward pass and loss function (MSE).  
 - Apply an optimization algorithm (gradient descent).  
3. Visualization:  
 - Plot the data points (scatter plot).  
 - Plot the best-fit line learned by your model on the same figure.  
4. Analysis:  
 - Summarize the training process.  
 - Discuss any difficulties or anomalies observed when fitting the line.  
 - Interpret how well the linear model fits the data visually and numerically (final loss, etc.).

## 2. Multiple-Feature Linear Regression

Objective:  
- Select three features and one output from the restaurant dataset (may be the same output variable as in Part 1 or a different one).  
- Train a simple linear regression model to predict the output from the chosen features.  
  
Deliverables:  
- Similar to Single-Feature Linear Regression if achievable.  
- If the task is not achievable, provide an analysis explaining why.  
  
Tasks/Steps:  
1. Data Selection: Justify which three features and output you chose and provide a brief rationale.  
2. Implementation in PyTorch:  
 - Build a multi-feature linear model.  
 - Train and optimize using MSE and gradient descent.  
3. Results:  
 - Show the final loss (training error; MSE).  
 - If the model successfully converges, report the final set of learned weights and bias.  
 - If the model fails to converge or you encounter difficulties, analyze and explain potential reasons.  
4. Interpretation:  
 - Discuss whether the multi-feature regression model appears to be a better fit than a single-feature model.  
 - Reflect on any new challenges that arose when using multiple features.

## 3. Polynomial Regression

Objective:  
- Using the same three features and one output from Part 2, implement polynomial regression of at least three different polynomial degrees (e.g., degree=2, degree=4, degree=6).  
- Train a polynomial regression model to predict the output from the chosen features.  
  
Deliverables:  
- A summary table or short discussion comparing performance for each chosen polynomial degree.  
- Plots or numerical results illustrating how well each polynomial model fits.  
- A reflection on potential risks of higher-degree polynomials (e.g., overfitting).  
  
Tasks/Steps:  
1. Feature Transformation:  
 - Explain how you generated polynomial terms (e.g., manually expanding each feature or using a PyTorch mechanism for polynomial features).  
 - Decide how to handle interactions (only single-feature powers vs. cross-terms).  
2. Training & Model Comparison:  
 - Train a polynomial regression model for each degree (≥ 3 degrees).  
 - Compare the training losses across different degrees.  
3. Analysis:  
 - Discuss any overfitting or underfitting observed.  
 - Identify which polynomial degree produced the most favorable result based on loss or other metrics.  
 - Provide insights into runtime or complexity differences.

## 4. Binary Classification

Tasks/Steps:  
1. Choose a Binary Label:  
 - Construct a binary classification label from the dataset (e.g., "Satisfied" vs. "Unsatisfied" or another appropriate yes/no outcome).  
2. Implement Logistic Regression in PyTorch:  
 - Loss function: Binary Cross-Entropy (BCE).  
3. Data Splitting & Preprocessing:  
 - Clearly split data into training and testing (or validation) sets.  
4. Model Training & Evaluation:  
 - Train on the training set for a certain number of epochs or until convergence.  
5. Report & Visualization:  
 - Summarize final training loss, test performance metrics, and any interesting findings.  
 - (Optional) Provide a decision boundary plot (if feasible for a single or two-feature scenario) or a confusion matrix heatmap illustrating predictions vs. ground truth.

## 5. Multiple-Classes Classification

Tasks/Steps:  
1. Label Selection:  
 - Identify a multi-class label from your dataset (≥ 3 classes); if the dataset does not inherently have three or more distinct classes, derive one.  
2. Strategy:  
 - Implement One-vs-All (OvA) and One-vs-One (OvO) logistic regression.  
 - OvA: Train a separate logistic regression classifier for each class vs. “all others.”  
 - OvO: Train pairwise classifiers for each possible pair of classes.  
3. Implementation Details:  
 - Manually handle the multi-class scenario within PyTorch, rather than relying on built-in high-level methods.  
4. Training & Evaluation:  
 - Train each classifier on the training data.  
 - On the test set, produce predictions by combining the outputs of sub-classifiers (OvA and OvO logic).  
5. Analysis & Discussion:  
 - Compare OvA vs. OvO in terms of code complexity, training time, and performance.