UTSA
CS 6243, EE 4463, EE 5573
Spring 2025
Assignment: HW-3
Topic: Parametric and Nonparametric Regression

I. Assignment Instructions

- **Submission:** Submit your report in the designated drop box on CANVAS by the corresponding assignment deadline.
- **Format:** Reports must be typed and uploaded as a PDF. Handwritten reports will not be graded. Any requested code should be presented at the end of your report in an appendix titled "Code." All code must include appropriate comments. Display equations and figures must be numbered, labeled, and captioned accordingly.
- **References:** You may use any source (notes, books, online), but all sources must be cited in a References List at the end of your report.
- **Originality:** You are not allowed to outsource this assignment (or parts of it) to another intelligent entity (human or AI – this includes ChatGPT and similar tools). You can still browse the internet and other sources, find information and ideas, and use them to compile your own solutions. Except for explicitly cited content, by submitting your work you verify and commit that it is your own intellectual work.
- **Grading:** Your assignment will be graded based on three equally weighted factors: Correctness, Completeness, and Clarity. For each problem, each criterion will be rated as follows: 100% (Excellent), 90%, 70%, 50% (Fair Effort), 30%, 10%, or 0% (Missing Effort). For example, a correct final result (100%), with almost complete method (90%), and almost clear presentation (90%) would receive $100\% \times 90\% \times 90\% = 81\%$. If the total points for the problem are 18, this would earn you 14.6 points (rounded to the nearest first decimal).
- **Teams:** Work in your determined project/assignment teams (see CANVAS). Each team member must individually submit a copy of the same team report on CANVAS. Only team members who make a submission before the deadline will receive a grade. It is expected that all team members contribute equally to the assignment. In case of doubt, the instructor will assign individual grades based on individual examination during office hours. In case members submit distinct reports, each team member will be graded based on their own submission.
- **Figures:** All figures must have the correct numbers, descriptive captions, axis labels, and legends for every curve included. Color, marker, and line-style combination should be such that distinct curves are discernible even in gray-scale printing.
- **Task Terminology:**

  Present: Show only the final result in math.

  Derive: Show all the necessary mathematical steps leading to final result.

  Compute: Write the Python code and present it.

  Plot: Write Python code that computes and plots; present both the code and the plots.

  Discuss: Discuss in words and reason/justify in detail (no need for math or code).

II. Problems

Problem 1: Data Splitting and Standardization (10 points)

Load the California Housing Dataset using `sklearn.datasets.fetch_california_housing`. Extract the feature `MedInc` (median income, column 0) as $x \in \mathbb{R}^N$ and the target (median house value) as $y \in \mathbb{R}^N$, where $N = 20640$ is the number of samples. Fix the test set size to 5000 samples, leaving the remaining 15640 samples available for training purposes. Use only `MedInc` as x for Problems 1–3; Problems 4–5 may use all features as specified later.

Tasks:

1. **Compute:** Write a function `split_data(x, y, train_size)` in Python using NumPy to prepare training and test sets. Inside the function, shuffle the data indices using `numpy.random.permutation` with a fixed seed (e.g., 42) for reproducibility, take the first `train_size` samples as training data, and assign the next 5000 samples as test data. Ensure `train_size` $\leq$ 15640; raise a ValueError with an appropriate message if exceeded. Return `x_train`, `x_test`, `y_train`, `y_test` in that order. Note that for each `train_size`, only the first `train_size` samples are used for training and the subsequent 5000 for testing; any remaining samples (up to 15640 - `train_size` - 5000) are excluded.

2. **Compute:** Test your `split_data` function with `train_size` = 1000, 5000, 10000. For each case, print the sizes of the resulting sets (e.g., `len(x_train)`, `len(x_test)`, `len(y_train)`, `len(y_test)`). Verify correctness: for `train_size` = 1000, expect 1000 training samples and 5000 test samples, leaving 15640 - 1000 - 5000 = 10640 samples unused; similarly, for 5000, expect 5000 and 5000; for 10000, expect 10000 and 5000.

3. **Compute:** Write a function `standardize(v)` that standardizes a data vector $v \in \mathbb{R}^n$ (e.g., `x_train`, `y_train`) to zero mean and unit variance. Compute the estimated mean $\mu = \frac{1}{n}\sum_{i=1}^{n} v_i$ and standard deviation $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_i - \mu)^2}$ using NumPy, then transform each element as $v_i' = \frac{v_i - \mu}{\sigma}$. Apply this function to standardize `x_train` and `y_train` after splitting, using `train_size` = 5000. Standardization ensures features and targets are on a comparable scale, critical for regression models sensitive to magnitude and distance-based methods. For example, unstandardized `MedInc` ranges from approximately 0.5 to 15, while the target ranges from 0.15 to 5 (in $100,000s), potentially skewing model performance if left unscaled.

4. **Plot:** Using the split and standardized data from Task 3 with `train_size` = 5000, create a Fig. 0, with two scatter plots side-by-side (e.g., using `plt.subplot`). In the left plot, show unstandardized `x_train` (x-axis) vs. unstandardized `y_train` (y-axis), labeled “Unstandardized Data.” In the right plot, show standardized `x_train` (x-axis) vs. standardized `y_train` (y-axis), labeled “Standardized Data.” Use small markers (e.g., dots), add axis labels (“Median Income” and “Median House Value”), and include a title for each subplot. This visualization highlights how standardization centers and scales the data.

Problem 2: Radius-Based Regression (15 points)

Implement radius-based regression from scratch in Python using NumPy. For each test input $x_i \in x_{\text{test}}$, compute the prediction $\hat{y}_i$ as the mean of all y_j corresponding to training inputs $x_j \in x_{\text{train}}$ such that $|x_j - x_i| \leq C$, where $C > 0$ is a fixed radius (Euclidean distance). If no training inputs fall within the radius of x_i, then predict $\hat{y}_i$ using the overall mean of y_{train}. This fallback must be handled explicitly. Use only training data for all predictions. Use a fixed random seed (e.g., 42) for all data shuffling. Evaluate performance using testing MSE as defined in the Assignment Instructions.

Tasks:

1. **Compute:** Write a function `radius_predict(x_train, y_train, x_test, C)` that computes predictions for `x_test`.

2. **Compute:** Fix `train_size` = 5000. For $C = 0.001, 0.005, 0.01, 0.05, 0.1, 0.3, 0.5, 1, 2, 3, 4, 5$, compute testing MSE.

3. **Plot:** In Fig. 1, plot testing MSE (y-axis) vs. C (x-axis) for `train_size` = 5000 (blue solid line).
4. **Compute:** Fix $C = 0.1$. For `train_size` = $1000, 2000, \dots, 10000$, compute testing MSE. When varying the training size, ensure that the training sets are nested: the training set for a smaller size N must be a subset of the training set for a larger N. To achieve this:
   - Perform a single shuffle of the entire dataset indices using `numpy.random.permutation` with a fixed seed (e.g., 42).
   - For the largest training size (e.g., 10000), take the first 10000 indices for training and the next 5000 indices for testing (fixed test set).
   - For each training size N (e.g., 1000, 2000, ..., 10000), take the first N indices from the same shuffled order. For example, indices [0:1000] are used for `train_size` = 1000, [0:2000] for `train_size` = 2000, etc., with [10000:15000] as the fixed test set.
5. **Plot:** In Fig. 2, plot testing MSE (y-axis) vs. training size (x-axis) for $C = 0.1$ (blue solid line).
6. **Discuss:** Figs. 1–2. Explain testing MSE trends with C and training size, identifying regions of good and poor generalization. Additionally, note how frequently the fallback training set mean is used for each C in Fig. 1, and discuss its impact on performance.

Problem 3: K-Nearest Neighbors Regression (15 points)

Implement K-Nearest Neighbors (KNN) regression from scratch in Python using NumPy. Define the prediction for a test point as the mean of the target values of the K nearest training points (based on Euclidean distance); if K exceeds the training set size, use all available points. Use a fixed random seed (e.g., 42) for consistency. Evaluate performance using testing MSE as defined in the Assignment Instructions.

- For each test point $x_i \in x_{\text{test}}$, compute $\hat{y}_i$ as the mean of y_j for the K closest $x_j \in x_{\text{train}}$ based on $|x_j - x_i|$.
- Use only training data for predictions.

Tasks:

1. **Compute:** Write a function `knn_predict(x_train, y_train, x_test, K)` that computes predictions for `x_test`.
2. **Compute:** Fix `train_size` = 5000. For $K = 1, 3, 5, 10, 20, 50, 100, 500, 1000$, compute testing MSE.
3. **Plot:** In Fig. 3, plot testing MSE (y-axis) vs. K (x-axis) for `train_size` = 5000 (blue solid line).
4. **Compute:** Fix $K = 100$. For `train_size` = $1000, 2000, \dots, 10000$, compute testing MSE. Use nested training sets as described in Problem 2. For example, after shuffling with seed 42, indices [0:1000] are used for `train_size` = 1000, [0:2000] for `train_size` = 2000, etc., with [10000:15000] as the fixed test set.
5. **Plot:** In Fig. 4, plot testing MSE (y-axis) vs. training size (x-axis) for $K = 100$ (blue solid line).
6. **Discuss:** Figs. 3–4. Explain testing MSE trends with K and training size, identifying regions of good and poor generalization. Compare with radius-based regression.

Problem 4: Polynomial Regression with One Feature (20 points)

Implement polynomial regression (solve LS via SVD), applied to a single scalar input feature. For an input feature vector $x \in \mathbb{R}^n$ (e.g., `x_train` from Problem 1, where $x = [x_1, x_2, \dots, x_n]^T$), the model predicts $y = X_{\text{poly}}^T w$, where $w = [w_0, w_1, \dots, w_d]^T \in \mathbb{R}^{d+1}$ is the weight vector, $y = [y_1, y_2, \dots, y_n]^T \in \mathbb{R}^n$ is the target vector, and $X_{\text{poly}} \in \mathbb{R}^{(d+1)\times n}$ is the polynomial feature matrix, the i-th column of which is $[1, x_i, x_i^2, \dots, x_i^d]^T$, corresponding to sample $i = 1, \dots, n$. The loss function is:

$$L(w) = \|X_{\text{poly}}^T w - y\|_2^2$$

Feature Standardization: To ensure numerical stability, standardize the polynomial feature rows (except the bias row) in X_{poly}. For each row $k = 1, \ldots, d$, treat $X_{\text{poly}}[k,:] \in \mathbb{R}^n$ as a vector, compute its mean $\mu_k = \frac{1}{n}\sum_{j=1}^{n} X_{\text{poly}}[k,j]$ and standard deviation $\sigma_k = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(X_{\text{poly}}[k,j] - \mu_k)^2}$, then standardize as $X_{\text{poly}}[k,:] \leftarrow \frac{X_{\text{poly}}[k,:]-\mu_k}{\sigma_k}$. You may use the `standardize(v)` function from Problem 1, by applying it to each row vector $X_{\text{poly}}[k,:]$. When varying training size (Task 6), compute μ_k and σ_k separately on each training set and use those values to standardize both the training and test sets.

Tasks:

1. **Compute:** Write a function `poly_features(x, degree)` that takes a column vector $x \in \mathbb{R}^n$ (e.g., standardized `x_train` or `x_test` from Problem 1) and returns the polynomial feature matrix $X_{\text{poly}} \in \mathbb{R}^{(d+1)\times n}$ with columns $[1, x_i, x_i^2, \ldots, x_i^{\text{degree}}]^T$ for each sample $i = 1, \ldots, n$. Apply feature standardization as described above after constructing the matrix, using `standardize(v)` from Problem 1.

2. **Compute:** Write a function `solve_LS(X_poly, y)` that trains $w \in \mathbb{R}^{d+1}$ using an SVD-based solution.

3. **Plot:** Using `train_size` = 5000 from Problem 1, scatter plot the standardized training data `x_train` (x-axis) vs. `y_train` (y-axis) in Fig. 5. Use small blue dots to visualize the data's shape. Label axes as "Median Income" and "Median House Value" and title the plot "Training Data Shape".

4. **Compute:** Fix `train_size` = 5000. For degrees $d = 1, 2, 3, 4, 5, 10, 15, 20$, train the model on `x_train` and `y_train`, then compute training MSE (on `x_train`, `y_train`) and testing MSE (on `x_test`, `y_test`) using the standard MSE definition.

5. **Plot:** In Fig. 6, plot training (blue) and testing (red) MSE (y-axis) vs. degree (x-axis) for `train_size` = 5000.

6. **Compute:** Fix $d = 9$. For `train_size` = 1000, 3000, 5000, 7000, 10000, compute training and testing MSE. Use nested training sets as in Problem 2: shuffle once with seed 42, take the first N indices for training (forming `x_train`, `y_train`), and the next 5000 for testing (e.g., [0:1000] for 1000, [0:3000] for 3000, ..., [10000:15000] for the fixed test set). Compute polynomial feature statistics separately on each training set.

7. **Plot:** In Fig. 7, plot testing MSE (blue solid line) (y-axis) vs. training size (x-axis) for $d = 9$. Label axes. Compare the results with the testing MSE attained by KNN and radius-based non-parametric regression.

8. **Plot:** For $d = 9$ and `train_size` = 5000, scatter plot the training data `x_train` (x-axis) vs. `y_train` (y-axis) in Fig. 8 using small blue dots. On the same figure, overlay the trained model's predictions $X_{\text{poly}}^T w$ (y-axis) vs. `x_train` (x-axis) as a red line. Sort `x_train` and corresponding predictions for a smooth curve. Label axes as "Median Income" and "Median House Value", title the plot "Polynomial Fit ($d = 9$)", and include a legend.

9. **Discuss:** Figs. 5–8. For Fig. 5, describe the training data's shape (e.g., linear, curved, scattered). For Figs. 6-7, explain MSE trends with degree and training size, identifying overfitting and underfitting. For Fig. 8, assess how well $d = 9$ fits the training data shape.

Problem 5: Fish Dataset – Standardization and Correlation (5 points)

In this problem, we analyze a new dataset containing fish measurements. The input $x \in \mathbb{R}^{2\times n}$ includes two features per fish: width (x_1) and length (x_2). The output $y \in \mathbb{R}^n$ is the fish weight. Files are provided on Canvas:

- `FISHDTR.csv` (training): columns `width`, `length`, `weight`.
- `FISHDTE.csv` (testing): same format.

Define the train and testing dataset matrices: TRI (input matrix from FISHDTR; size $2 \times n_{\text{train}}$), TRO (output vector from FISHDTR; size $n_{\text{train}} \times 1$), TEI (input matrix from FISHDTE; size $2 \times n_{\text{test}}$), and TEO (output vector from FISHDTE; size $n_{\text{test}} \times 1$).

Tasks:

1. **Compute:** Standardize each row of TRI and vector TRO using `standardize(v)` from Problem 1. Standardize also TEI and TEO using the same means/stds found on the training data.
2. **Compute:** Compute correlation matrix $C \in \mathbb{R}^{3\times 3}$, where:
$$C(i,j) = |\rho(x_i, x_j)|, \quad C(i,3) = C(3,i) = |\rho(x_i, y)|,$$
and $\rho(a,b) = \frac{\sum_k (a_k-\bar{a})(b_k-\bar{b})}{\sqrt{\sum_k (a_k-\bar{a})^2}\sqrt{\sum_k (b_k-\bar{b})^2}}$ is the Pearson correlation coefficient. Present matrix C. Discuss which input is most correlated with the output.
3. **Plot:** In Fig. 9, plot TRO vs. TRI[0,:] (width). In Fig. 10, plot TRO vs. TRI[1,:] (length). Use small blue dots, axis labels, and titles.

Problem 6: Fish Dataset - Polynomial Regression with One Feature (15 points)

In this problem, apply polynomial regression to each input feature individually using the standardized training data (TRI, TRO). Consider the first feature (width, TRI[0, :]) and the second feature (length, TRI[1, :]) as scalar inputs. Use degrees $M = 0, 1, 2, 3, 4, 5$. Construct univariate polynomial feature matrices, apply SVD to solve for the weight vector, and evaluate model performance on both training and testing sets. Use the standardized TEI and TEO for testing. For each degree, use the same standardization approach as in Problem 4, applying `poly_features` and reusing the `standardize(v)` function as needed.
Tasks:

- **Plot:** Fit a polynomial regression model of degree $M = 3$ using the "width" feature (TRI[0, :] as input, TRO as output). Generate predicted values over a densely sampled input domain $x \in [1, 9]$ with a step size of 0.001 (use standardized version of this input domain). In Fig. 11, plot the predicted curve $f(x)$ and overlay the training data as a scatter plot (TRO vs. TRI[0, :]). Label axes, add a legend, and include an appropriate title.
- **Plot:** Repeat the above procedure for the "length" feature (TRI[1, :]) using degree $M = 3$, with input domain $x \in [1, 60]$ (standardized). In Fig. 12, plot the predicted curve and training data scatter plot (TRO vs. TRI[1, :]) with full labeling and title.
- **Plot:** For each feature separately, train univariate polynomial models with degrees $M = 1, 2, 3, 4, 5$, and compute the test MSE on TEI and TEO (using TRI[0,:] and TRI[1,:] independently). In Fig. 13, plot test MSE (y-axis) vs. degree M (x-axis), using blue for width-based models and red for length-based models. Add legend, labels, and title. Discuss which feature is more predictive, and relate this to the correlation matrix from Problem 5.

Problem 7: Fish Dataset - Polynomial Regression with Both Features (20 points)

Extend polynomial regression to use both input features (width and length) jointly. Construct bivariate polynomial feature matrices of total degree $M = 1, 2, 3$. For example, for $M = 2$, the model includes terms: $1, x_1, x_2, x_1^2, x_1x_2, x_2^2$. Let $x_1 = \text{TRI}[0, :]$, $x_2 = \text{TRI}[1, :]$, and $y = \text{TRO}$. Use the same standardization and regression framework as in Problem 6. Solve the least-squares regression problem using the SVD-based solution.

Tasks:

- **Present:** For each degree $M = 1, 2, 3, 4$, write the explicit expression for the bivariate polynomial model including all terms.

- **Compute:** Construct bivariate polynomial features for each degree $M = 1, 2, 3$, and train a model on TRI/TRO using the SVD solution. Evaluate the trained model on both training and testing data. Compute both train and test MSE for each M.
- **Plot:** In Fig. 14, plot MSE (y-axis) vs. polynomial degree M (x-axis), using red for training MSE and blue for testing MSE. Label axes, add a title, and include a legend. Compare bivariate test MSE with those of the best univariate models in Problem 6. Discuss model complexity, overfitting, and generalization.