Regression: Initial Steps, LR and kNN

1. Load Data
- Ensure the data is loaded correctly.
- Examine the first few rows to understand its structure and contents.

2. Train / Test Split
- Decide on the proportion for your train-test split. A common choice is 80-20 or 70-30.
- After splitting, set the test set aside. Do not touch or peek into it until the best models are ready for evaluation (in some days).

3. Cross-validation
- Set up a cross-validation strategy (e.g., sklearn's `KFold`).
- Choose the evaluation metrics appropriate for your problem.

4. Baseline
- Establish a simple baseline model. This will give you an initial performance metric to beat.
- Perform cross-validation on this model and record the results.

5. Linear Regression
- Train a simple linear regression model, performing cross-validation and analyzing its performance. Compare it with the baseline.

6. K-Nearest Neighbors
- Choose a range for the number of neighbors, e.g., from 1 to 30.
  - Train and validate the kNN model across this range. Set other parameters as default, except `n_jobs`.
  - Plot the training and validation performance of the different number of neighbors.
  - What do you think is the best number of neighbors?
- Did you scale your data before applying kNN?
  - If yes: now check the performance of kNN without scaling.
  - If no: now check the performance of kNN with scaling.
  - Which option seems to be better?
- Investigate the impact of different scaling algorithms (e.g., MinMaxScaler, StandardScaler, RobustScaler) on kNN's performance.
- Examine how changing the `weights` parameter to `distance` affects performance. What happens? Why?
- Using the best hyperparameters derived from the above experiments, create a heatmap to visualize the performance of kNN with various combinations of `n_neighbors` (e.g., 1 to 10) and `p` values (e.g., 1 to 5). Evaluate whether the previously determined "best hyperparameters" remain optimal.

7. Final Comparison
- Plot a comparison of the performances of your baseline model, Linear Regression (LR), and the optimal kNN model.