Instructions: Please read the following instructions thoroughly

- For the entire assignment, use `Python` for your analysis. Write your code in a Jupyter Notebook named as `[your-student-ID]_hw2.ipynb` (e.g., `2025-20000_hw2.ipynb`). The use of `R` is not allowed. You are allowed to use any libraries in `Python`.
- Type up your report and save as PDF named as `[your-student-ID]_hw2.pdf`. We do not allow the submission of a photo or a scanned copy of hand-written reports.
- Please upload the two files on eTL **without zipping**. Submissions via email are not allowed. The violation of the filename or submission instruction will result in the penalty of 5 points.
- You can discuss the assignment with your classmates but each student must write up his or her own solution and write their own code. Explicitly mention your classmate(s) you discussed with or reference you used (e.g., website, Github repo) if there is any. If we detect a copied code without reference, it will be treated as a serious violation of the student code of conduct.
- We will apply a grace period of late submissions with a delay of each hour increment being discounted by 5% after the deadline (i.e., 1-minute to 1-hour delay: 95% of the graded score, 1 to 2-hour delay: 90% of the graded score, 2 to 3-hour delay: 85%, so on). Hence, if you submit after 20 hours post-deadline, you will receive 0 points. No excuses for this policy, so please make sure to submit in time.

1. [**50 pts**] You will use the `CarPrice` dataset attached (`CarPrice.csv`). The response variable is `price` (the price of car); use the remaining variables as predictors. *Standardize* predictors for all regularized methods; do not penalize the intercept.

   (a) **[5 pts] Baseline Multiple Linear Regression.** Fit a multiple linear regression model to predict `price` using all available predictors.
   - State whether you included an intercept and whether you standardized predictors (explain briefly why standardization is important for ridge/lasso).
   - Report the estimated coefficients (including the intercept), which are statistically significant, and the training R^2.

   (b) **[15 pts] Ridge Regression (Closed-Form Implementation).** Implement your *own function* for ridge using the closed form

   $$\hat{\boldsymbol{\beta}}_{\text{ridge}}(\lambda) = \left(X^\top X + \lambda D\right)^{-1} X^\top y,$$

   where $D = \text{diag}(0, 1, \ldots, 1)$ so the intercept (first column of X) is *not* penalized. Use standardized predictors.
   - For λ on a logarithmic grid (e.g., $10^{-3}, 10^{-2}, \ldots, 10^{3}$), plot coefficient paths vs. $\log_{10} \lambda$.
   - Compare your implementation against `sklearn.linear_model.Ridge` with `fit_intercept=False` (since you already added the intercept column) and matching standardization. Provide a small table of coefficients and training R^2 for three representative λ values (e.g., $0.01, 1, 100$).

- Briefly comment on shrinkage patterns you observe.

(c) **[15 pts] Lasso Regression (with Optimization Function).** Lasso has no closed form. Implement it via generic optimization:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \frac{1}{2n}\|y - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}_{-0}\|_1,$$

where $\boldsymbol{\beta}_{-0}$ excludes the intercept (no penalty on intercept).

- Write the objective and provide its (sub)gradient for the squared-loss term; clearly state how you handle the non-differentiability of $\|\cdot\|_1$
- You can use an off-the-shelf optimizer only for the optimization routine (e.g., `scipy.optimize.minimize` with `method="SLSQP"` or `"L-BFGS-B"` on a smooth approximation). The loss and its gradient must be coded by *you*.
- Compare coefficients and training R^2 to `sklearn.linear_model.Lasso` under matched preprocessing and λ. Identify which coefficients are driven exactly to zero and briefly interpret why.

(d) **[15 pts] Model Selection, Validation, Reporting Test Performance.**

- First, perform a random 80%/20% train/test split (**report the random seed**).
- Then, using 5-fold cross-validation on the training data, choose the best λ for ridge and lasso (minimizing validation MSE).
- Fit each model including the plain linear regression model in part (a) on the training set with its chosen λ, and report test R^2 and MSE.

Model	$\lambda^\star$	Train R^2	Test R^2	Test MSE
Least Squares	·	·	·	·
Ridge	·	·	·	·
Lasso	·	·	·	·

Briefly discuss which regularizer performed better and provide a plausible explanation (bias–variance trade-off, sparsity, correlated predictors, etc.).

2. Use the `CarPrice` dataset again (`CarPrice.csv`) to predict the continuous response `price` from the remaining variables. Treat this as a *regression* task. *Do not* standardize features for tree-based models. Reuse the same random seed and the same 80%/20% train/test split you created in Part 1(d).

   (a) **[10 pts] Decision Tree (Off-the-Shelf).** Fit a regression tree using `sklearn.tree.DecisionTreeRegressor` with criterion `"squared_error"`.

   - Report the hyperparameters you used (e.g., `max_depth`, `min_samples_leaf`, `random_state`) and provide a brief rationale for any regularization you applied (depth/leaf control).
   - Evaluate on the held-out test set: report test R^2 and test RMSE. Also report training R^2.
   - Provide either (i) a plot of predicted vs. actual `price` on the test set (with a 45-degree reference line) or (ii) the top 5 splitting rules from the root downward (feature and threshold).

   (b) **[15 pts] Bagging (Implement from Scratch; Use Cross-Validation).** Implement *bagging* yourself using `DecisionTreeRegressor` as the base learner (unpruned unless you justify otherwise).

   - For B bootstrap models (e.g., $B \in \{1, 5, 10, 20, 50, 100\}$), draw with replacement from the *training* set, fit one tree per bootstrap sample, and form predictions by *averaging* predictions across the B trees.
   - Use **5-fold cross-validation on the training set** to select B (and any tree hyperparameters you choose to tune, such as `max_depth` or `min_samples_leaf`) by minimizing CV RMSE.
   - After selecting B, refit bagging on the full training set and report: training R^2, test R^2, and test RMSE.
   - Make a plot of *CV RMSE* versus B; comment on the trend (variance reduction, stabilization).

   (c) **[15 pts] Random Forest (Implement from Scratch; Use Cross-Validation).** Extend your bagging implementation to a *random forest* by injecting feature randomness at each split. You may leverage `DecisionTreeRegressor` with `max_features` set appropriately, but you must write the ensemble loop, bootstrapping, aggregation, and cross-validation yourself.

   - Train B trees (use the same B grid as above). At each split, restrict candidate features to a random subset of size m. Consider two settings, e.g., $m = \lfloor \sqrt{p} \rfloor$ and $m = \lfloor p/3 \rfloor$, where p is the number of predictors.
   - Use **5-fold cross-validation on the training set** to select B and m by minimizing CV RMSE.
   - After selection, refit the random forest on the full training set and report: training R^2, test R^2, and test RMSE. Compare to bagging and to a single tree.

- Compute **variable importance** via *permutation importance* on the test set (implement the permutation routine yourself): for each feature, measure the increase in test MSE when its column is randomly permuted. Present a bar plot of importances and briefly interpret the top 3 features.

(d) **[10 pts] Synthesis and Comparison.** Summarize and compare all methods on the same split. Present a table such as

Model	Train R^2	Test R^2	Test RMSE
Decision Tree	·	·	·
Bagging (best B)	·	·	·
RF (best B, m)	·	·	·

Briefly discuss: (i) which method generalizes best and why, (ii) how variance and bias evolve from a single tree to bagging to random forest, and (iii) how your permutation-importance results align with domain intuition about `price` drivers.