Project 1: Linear Regression and Regularization

(First discussion: Sept 24; Last questions: Oct 8; Deadline: Oct 15; In charge: Urban Ulrych)

This project explores linear regression and regularization techniques in the context of predicting house prices. You will use the *Housing* dataset from Ames, Iowa (USA), which contains 1,460 observations of residential properties. The dataset includes one target variable, *SalePrice* (the sale price of the house in U.S. dollars), and 79 explanatory variables describing different aspects of the homes. Of these predictors, 35 are numerical and 44 are categorical.

The goal of the project is to build models that both explain and predict *SalePrice*, while comparing the strengths and limitations of different regression approaches. You can refer to the file `data_description.txt` for a detailed description of all variables.

This dataset is widely used in predictive modeling competitions and mimics real-world challenges such as skewed target distributions, missing values, multicollinearity among predictors, and many categorical features. Addressing these issues is part of the learning objective of the project.

1. To start, you need to import the dataset into Python in the correct format.

   a) Import the dataset `Housing.csv` in Python as a pandas DataFrame. Separate the explanatory features X (*Housing*) from the target variable y (*SalePrice*).

   b) Graphically determine whether the target variable *SalePrice* is approximately Gaussian. If not, suggest a suitable transformation to bring *SalePrice* closer to a Gaussian distribution and apply this transformation to the dataset. Why is it important to consider such potential transformations?

   c) Since the regression models have to be evaluated on a different dataset than the one used for training, split the data into two subsets: $(X, y)_{train}$ and $(X, y)_{test}$. Randomly assign 70% of the observations to the training set and the remaining 30% to the test set.

   d) Replace missing values in X using the training data statistics only:
      - For numerical features, replace missing values with the mean of the column (computed from the training set).
      - For categorical features, replace missing values with the most frequent category (computed from the training set).
      - Some categorical variables admit `'NA'` (or `'None'`) as a valid category, which should be treated as an actual level and not as missing.

      Next, standardize all numerical features by applying a z-score transform (subtract the training mean and divide by the training standard deviation), and apply the same transformation to the test data. Finally, use one-hot encoding for all categorical features (e.g., with `pd.get_dummies`). After encoding, ensure that the training and test sets have exactly the same columns.

      Hint: After one-hot encoding, some categories may appear in the training set but not in the test set (or vice versa). You can use the command

      ```
      X_test = X_test.reindex(columns=X_train.columns, fill_value=0)
      ```

      to align the columns of the test set with those of the training set, filling missing dummy columns with zeros.

2. This question focuses on building a linear regression model to predict the variable *SalePrice* using only the 35 numerical features.

   a) Restrict your analysis to the numerical predictors only (i.e., exclude all categorical features from X). Fit a linear regression model on the training dataset using the `sklearn` Python package. The regression should include an intercept term. Present a table with the regression coefficients for each feature. Compare the in-sample and out-of-sample Mean Squared Error (MSE) and R^2.

      If you transformed the target variable, you have to inverse-transform the predictions before computing MSE and R^2, so that these metrics are reported on the original *SalePrice* scale. In addition, also report the MSE and R^2 on the transformed scale, and comment on the differences between the two. What does each set of metrics tell you about model performance?

   b) The `sklearn` package does not provide standard errors for the estimated regression coefficients, which are essential tools to assess the statistical precision of an estimate.[1] Therefore, you will now use matrix algebra in Python with the `numpy` package to compute the standard errors of the estimated coefficients $\hat{\beta}$. All computations in this part should be performed using the training set only. Let $A \in \mathbb{R}^{m \times (d+1)}$ denote the design matrix (including a column of ones for the intercept term), and let $y \in \mathbb{R}^m$ denote the observed target values.

      (i) Compute the estimated coefficients $\hat{\beta}$ using

      $$\hat{\beta} = (A^\top A)^{-1} A^\top y\,.$$

      Note that $\hat{\beta}_0$ denotes the estimate of the intercept.
      Practical note: In code, do not form matrix inverses explicitly; instead use a numerically stable equivalent such as `np.linalg.solve(A.T @ A, A.T @ y)`.

      (ii) Compute the standard error of each coefficient $\hat{\beta}_j$, $j = 0, \ldots, d$, using

      $$\widehat{\sigma}^2 = \frac{1}{m-(d+1)} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2, \qquad \mathrm{Var}(\hat{\beta}) = \widehat{\sigma}^2 (A^\top A)^{-1},$$

      and

      $$\mathrm{SE}(\hat{\beta}_j) = \sqrt{\widehat{\sigma}^2 \cdot \left[(A^\top A)^{-1}\right]_{jj}}\,,$$

      where $[(A^\top A)^{-1}]_{jj}$ denotes the j-th diagonal element. Recall that $j = 0$ corresponds to the intercept.

      (iii) Compute the in-sample MSE and R^2 using matrix algebra.

      (iv) Replace $(A^\top A)^{-1} A^\top$ by the Moore–Penrose pseudoinverse A^+ (use `np.linalg.pinv`). Do $\hat{\beta}$, $\widehat{\sigma}^2$, and the standard errors change? Briefly explain when the results are identical and when they can differ.

      (v) Confirm your results using the OLS function from the `statsmodels` package. Report the coefficient table and standard errors, and check that they match your matrix-algebra results up to numerical rounding.

[1] Under the classical linear model with homoscedastic, normally distributed errors, the OLS estimator $\hat{\beta}$ is exactly normal, and each t-statistic $\hat{\beta}_j / \mathrm{SE}(\hat{\beta}_j)$ follows a t-distribution with $m-(d+1)$ degrees of freedom. This enables confidence intervals and significance tests.

3. In this question, you will implement regularization techniques and compare their performance to ordinary least squares. Work with the full *Housing* dataset prepared in Question 1, which already includes both numerical features (standardized) and categorical features (one-hot encoded dummies). Use the same training and test splits as before to ensure consistency across questions.

   a) Fit an OLS regression of the (potentially transformed) target variable *SalePrice* on all explanatory variables in the prepared *Housing* dataset. Report the in-sample and out-of-sample MSE and R^2. Compare these results to the numerical-only regression from Question 2.a). How do the in-sample and out-of-sample metrics change when including categorical features, and what does this reveal about the model's ability to generalize?

   b) Implement the Truncated Pseudoinverse, Ridge, Lasso, and Elastic Net regressions. Use 8-fold cross-validation on the training set to tune the hyperparameters of each regularization technique, using the MSE as the selection criterion. Compare their in-sample and out-of-sample performance (MSE and R^2) with the OLS results from Questions 2.a) and 3.a).

      All regressions should include an intercept term. The intercept must not be penalized during regularization. Why is it important that the intercept is not penalized in these models?

   c) During cross-validation, what are possible sources of *information leakage*? Briefly describe what leakage means in this context, and explain what steps you would take to avoid it if you were building a more complete data preprocessing pipeline (e.g., with imputation, scaling, or encoding). You do not need to implement these steps here, only to explain the idea.

   d) For the Lasso and Elastic Net regularization techniques, how many coefficients are non-zero ($\hat{\beta}_j \neq 0$)? Compare this number with the number of coefficients retained by the Ridge and Truncated Pseudoinverse models and provide an explanation.

   e) Based on your findings from Questions 2 and 3, which model would you recommend for predicting house prices? Justify your choice not only by comparing performance metrics, but also by discussing the nature of the problem (e.g., number of features, presence of categorical variables, potential collinearity, sparsity, nonlinearity). Explain how the strengths and limitations of the chosen method align with this problem structure.