Statistiques : Final TP

1) Write a python code that given a sample of datapoints X where $X_i \sim N(\mu', \sigma^2)$, a value μ and a significance level α implements a hypothesis test to decide whether $\mu = \mu'$. Print the z-statistic, the reject region, the p-value and the final decision of the test. Generate 3 different Gaussian samples to test it. Assume that the sample size is large enough, $n = 100$. What would you change if the sample would be small, say $n = 20$?
2) Load the dataset and split it into training and test sets. Use an 80-20 split ratio, and set the random seed to 0 for the split. Calculate the R-squared and Mean Squared Error (MSE) metrics.
3) Implement a variable selection method based on testing the null effect for each coefficient θ_j for the same dataset. This is a greedy forward selection algorithm for choosing variables believed to be important for prediction, i.e., it selects iteratively one relevant variable until a certain halting condition is met. The process, illustrated in Algorithm 1, is as follows :

Algorithm 1: FVS : Forward Variable Selection

Data: X, y
Result: A collection of variables
$\tilde{y} \leftarrow y$
$V \leftarrow$ all the columns in X `// the set of unused variables`
while *halting condition not reached* **do**
  for $i \in V$ **do**
    $x \leftarrow$ the i-th column of X
    $\hat{\theta}_0, \hat{\theta}_1 \leftarrow$ fit a linear model with the data $(x, \tilde{y})$
    Run a hypothesis test to check wether $\hat{\theta}_1 = 0$, compute the t-statistic and the p-value of the test
  $i \leftarrow$ the variable with the smallest p-value in the last for
  Remove i from the set of unused variables V
  $\tilde{y} \leftarrow \tilde{y} - \hat{y}$ `// remove the effect of` Xi `from the model by updating the residuals.`

   (a) In each iteration, perform a hypothesis test for the nullity of a variable. Output the variable selected in each iteration, along with the associated p-value.
   (b) Select the variables for which the test rejects the null hypothesis with a significance level of 0.1.
   (c) Conduct an Ordinary Least Squares (OLS) regression using only the selected variables and calculate the R-squared and MSE for both the training and test sets.

4) Run ridge regression using scikit-learn on the training set without cross-validation this time. Display two subplots at the end :
   (a) The first subplot should show the evolution of the coefficients for each different value of the penalty parameter.
   (b) The second subplot should display the evolution of the R-squared coefficient at each of the 30 iterations.

Since we are going to perform similar tasks for Lasso and Elastic Net, it is mandatory to write this code as an independent function that can be parameterized for each specific case. Store the error metrics on the train and the test sets.

5) Select the penalty parameter using RidgeCV with 10-fold cross-validation. Store the R2 and MSE results.
6) Run the code for Lasso as explained in Point 4. Run the code for 30 different values of the penalty parameter, which should be on a logarithmic scale between 1×10^{-3} and 1×10^{2}. Without further tuning the parameters, you will get a warning. Explain the reason for that warning and propose a solution for it.
7) Select the penalty parameter using LassoCV with 10-fold cross-validation. Store the R2 and MSE results.
8) Run the code for ElasticNet as explained in Point 4. Run the code for 30 different values of the penalty parameter, which should be on a logarithmic scale between 1×10^{-3} and 1×10^{2}. Without further tuning the parameters, you will get a warning. Explain the reason for that warning and propose a solution for it.
9) Select the penalty parameter using ElasticNetCV with 10-fold cross-validation. Store the R2 and MSE results.
10) Plot the test and training errors of all the methods and explain the 3 differences and similarities of the penalized models, Ridge, Lasso and ElasticNet.