UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Evaluation of Statistical and Machine Learning Models

An empirical approach

Presented by :
Aya WIFAK, Alicia DAHMANI, Aya EL FARFAR

Under the supervision of:
Philippe DE PERETTI

Academic Year 2025 – 2026

Contents

1 Introduction

Econometrics emerged in a context of data scarcity. Historically, the aim was to validate a specific economic theory using a limited number of explanatory variables on medium-sized samples. Today, with the rise of Big Data and increasing digitization, access to information is no longer a limitation and has even reversed the paradigm. We have moved from scarcity to data overload.

This structural change has fostered the emergence of statistical learning. Classical econometrics focuses on inference (understanding the causality between X and Y and testing its significance), while statistical learning prioritizes prediction. The objective is no longer to understand the current situation but to anticipate the future as accurately as possible.

In this environment, OLS estimation methods become inefficient. This is because the data becomes scattered and the risk of misleading correlations rises dramatically. In this situation of abundant variables, an initial intuition would be to include all possible predictors in the model, hoping to capture all the information, but in the field of statistics, "more" does not mean "better".

A model that is too complex captures the noise of the training data set rather than the signal, which degrades its predictive ability on a new sample: this is known as overfitting. Conversely, an overly simple model misses key patterns, resulting in poor predictive capabilities: this is known as underfitting.

This context raises a central question: *How can we automatically select relevant variables to build a model that is both interpretable and accurate in its predictions?* .

To address this challenge, two main learning approaches have emerged. Discrete methods (hard tresholding) such as Stepwise selection (Forward/Backward) were the first response. However, the introduction of penalization methods (soft tresholding) marked a turning point. Tibshirani's seminal paper (1996) introducing the LASSO revolutionized statistical learning, paving the way for continuous methods such as LARS (Efron et al., 2004) and ElasticNet (Zou Hastie, 2005). The choice between these methods depends on data structure and involves a fundamental trade-off: the bias-variance tradeoff.

This trade-off requires metrics for variable selection and model assessment. We will distinguish between, significance tests used in iterative procedures (t-test, partial F-test) goodness-of-fit measures (MSE, R^2, adjusted R^2), information criteria (AIC, BIC) and resampling methods (K-FOLD, LOO, CV). This will allow us to evaluate model generalization and the stability of the variable selection.

This thesis aims to compare these different approaches empirically using several simulated data scenarios: the null hypothesis of independence, multicollinearity, outliers, and struc-

tural breaks. We will conclude with an empirical application on the Diabetes dataset.

We will perform all of our simulations on SAS, generate our data using IML procedures and analyze it with the PROC GLMSELECT command, considering the various metrics mentioned above.

2 Theory

2.1 The bias-variance trade-off

To understand the variable selection mechanics, we need to formalize what we are trying to minimize. In a regression setting, we assume the relationship $Y = f(X) + \epsilon$, where ϵ represents random noise with zero mean and variance σ^2. The goal is to estimate $f(X)$ by a function $\hat{f}(X)$. The quality of this estimate, measured by the Mean Squared Error (MSE), depends on more than just the model's fit to the training data. Indeed, standard statistical theory demonstrates that the expected prediction error can be decomposed into three distinct terms:

$$\mathbb{E}[(Y - \hat{f}(X))^2] = (f(X) - \mathbb{E}[\hat{f}(X)])^2 + \text{Var}[\hat{f}(X)] + \sigma^2 \tag{1}$$

This equation represents the bias-variance decomposition.

- σ^2 : the irreducible error.
- $(f(X) - \mathbb{E}[\hat{f}(X)])^2$: the squared bias. It measures how closely the estimated function approximates the true function.
- $\text{Var}[\hat{f}(X)]$: the variance. It measures the sensitivity of the model to variations in the training set.

Minimizing both simultaneously is impossible: increasing model complexity reduces bias but increases variance, and vice-versa.

To illustrate this trade-off, we designed a data-generating process based on the function $f(x) = 2x^2 + 1$. The procedure is outlined in the pseudo-code below.

Algorithm 1: Monte Carlo Simulation: Bias-Variance Estimation

Inputs: $N = 100$, $M = 1000$, $\mathcal{X}_{eval}$ (grid), $\sigma = 0.5$

1. Define the true function $f(x) = 2x^2 + 1$ and compute the true target:

   $Y_{true} = f(\mathcal{X}_{eval})$.

2. For each iteration k from 1 to M:

   a. Generate $X \sim \mathcal{U}[-1, 1]$ and noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

   b. Compute the response vector: $Y = f(X) + \epsilon$.

   c. For each degree $d \in \{0, 2, 8\}$:

   - Construct the design matrix (Column of 1s for $d = 0$).
   - Estimate $\hat{\beta}_d$ via OLS and compute predictions $\hat{Y}_d$.
   - Store $\hat{Y}_d$ for iteration k.

3. For each degree $d \in \{0, 2, 8\}$:

   a. Expectation = Mean($\hat{Y}_d$).

   b. $Bias_d^2 = (\text{Expectation} - Y_{true})^2$.

   c. $Var_d = \text{Variance}(\hat{Y}_d)$.

4. Return the global Mean of $Bias^2$ and Var per model.

The analysis considers three types of estimators. The results, based on 1,000 simulations, are presented below:

Model	**Squared bias**	**Variance**
Degree 0 (Under)	0.2899	0.0062
Degree 2 (Good)	3.97E-6	0.0071
Degree 8 (Over)	0.372E-4	0.0210

Table 1: Monte Carlo simulation results (1,000 replications)

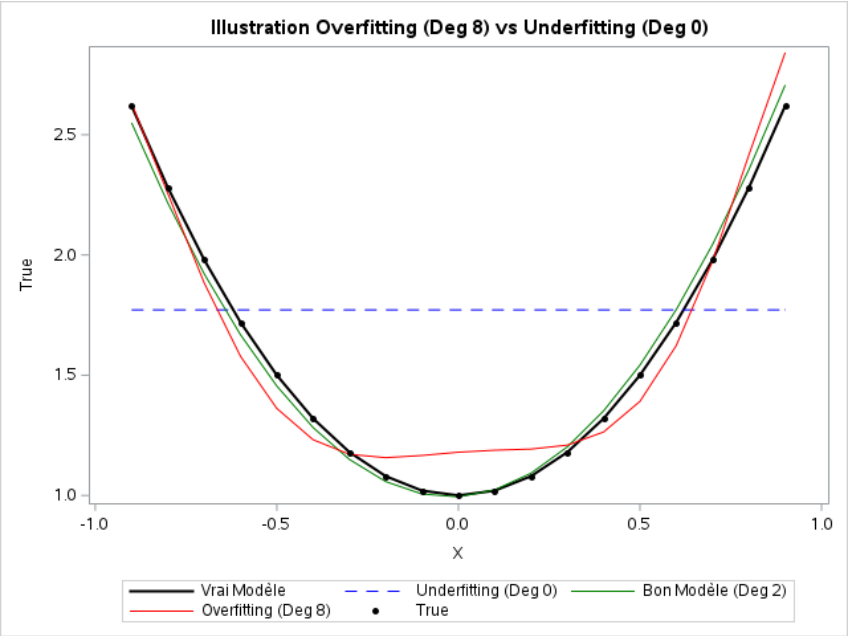


Figure 1: Illustration Overfitting (Deg 8) vs Underfitting (Deg 0)

- $f(x) = c$, represented by the blue curve (horizontal dotted line). This is a case of *underfitting*. We observe a very high squared bias (0.2899) but very low variance (0.0062). The model is stable but incorrect.
- $f(x) = ax^2 + bx + c$, represented by the green curve. This is the optimal model, achieving the best bias-variance tradeoff and minimizing the MSE.
- $f(x) = \sum_{i=0}^{8} \beta_i x^i$, represented by the red curve. This is a case of *overfitting*. Although the bias remains close to zero, the variance (0.0210) is about three times higher than that of the quadratic model, proving that it makes unstable predictions when exposed to new data sets.

This simulation highlights the need to control model complexity to ensure generalization. The optimal model is neither the simplest (which fails to capture the signal) nor the most complex (which fits the noise), but the one that strikes the ideal bias-variance tradeoff.

However, in real-world applications, the actual process of generating data $f(x)$ remains unknown. Consequently, the ideal number of variables or the required complexity cannot be determined. Variable selection methods have been developed to solve this optimization problem in the absence of such information. In the following sections, we detail the two major families of algorithms designed to navigate this tradeoff: iterative approaches rooted in Statistical Learning, and penalization techniques derived from Machine Learning.

2.2 Statistical Learning

2.2.1 Forward selection

Forward Selection, extensively reviewed by Hocking (1976), is a bottom-up, greedy strategy. It is particularly well-suited for high-dimensional settings ($p > N$) where fitting the full model is infeasible. The algorithm begins with the null model ($\mathcal{M}_0$) and sequentially adds the variable that most significantly improves the fit.

Algorithm 2: Forward Selection Algorithm

Input : Data (X, Y), Significance level α

Output: Selected subset $\mathcal{M}$

Initialization: $\mathcal{M} \leftarrow \emptyset$ (Null Model), Candidates $\mathcal{C} \leftarrow \{1, \ldots, p\}$.

while *variables remain in* $\mathcal{C}$ **do**

1. Identify the best candidate j^* that maximizes fit improvement:

$$j^* = \arg\min_{j \in \mathcal{C}} RSS(\mathcal{M} \cup \{j\})$$

2. Compute p-value associated with adding x_{j^*}.

if *p-value* $< \alpha$ **then**

Update model: $\mathcal{M} \leftarrow \mathcal{M} \cup \{j^*\}$;

Update candidates: $\mathcal{C} \leftarrow \mathcal{C} \setminus \{j^*\}$;

else

Stop (No significant improvement);

return $\mathcal{M}$

Despite its computational efficiency, its major drawback is monotonicity. Actually, once a variable enters the model, it cannot be removed, even if it becomes redundant following the addition of subsequent variables.

2.2.2 Backward selection

Backward Elimination, described by Draper and Smith (1966) as the standard procedure when computationally feasible, is a top-down approach. Unlike Forward Selection, it requires the full model to be estimable, imposing the constraint $N > p$.

Algorithm 3: Stepwise Selection Algorithm

Input : Data (X, Y), Thresholds α_{enter} and α_{remove} (usually $\alpha_{enter} \leq \alpha_{remove}$)

Output: Selected subset $\mathcal{M}$

Initialization: $\mathcal{M} \leftarrow \emptyset$ (Null Model), Candidates $\mathcal{C} \leftarrow \{1, \ldots, p\}$.

while *TRUE (Loop untill stability)* **do**

  1. Identify best candidate $j^* \in \mathcal{C}$ to enter.

  if $p\text{-}value_{j^*} < \alpha_{enter}$ **then**

    Add variable: $\mathcal{M} \leftarrow \mathcal{M} \cup \{j^*\}$;

    Remove from candidates: $\mathcal{C} \leftarrow \mathcal{C} \setminus \{j^*\}$;

  else

    Break (Cannot add any more variables);

  2. Identify worst variable $k^* \in \mathcal{M}$ to remove (highest p-value).

  if $p\text{-}value_{k^*} > \alpha_{remove}$ **then**

    Remove variable: $\mathcal{M} \leftarrow \mathcal{M} \setminus \{k^*\}$;

    Return to candidates: $\mathcal{C} \leftarrow \mathcal{C} \cup \{k^*\}$;

return $\mathcal{M}$

This method captures joint effects from the outset. This allows it to identify complex relationships that Forward Selection might miss due to its sequential nature.

2.2.3 Stepwise forward

Stepwise Selection, introduced by Efroymson (1960), is a hybrid strategy designed to overcome the unidirectional limitation of Forward Selection.

It operates as a forward procedure but integrates a backward elimination step after each inclusion. Specifically, if a previously selected variable loses significance due to the addition of a new predictor (capturing the variance more effectively), it is immediately dropped.

Algorithm 4: Stepwise Selection Algorithm

Input : Data (X, Y), Thresholds $\alpha_{enter} \leq \alpha_{remove}$

Output: Selected subset $\mathcal{M}$

Initialization: $\mathcal{M} \leftarrow \emptyset$ (Null Model), Candidates $\mathcal{C} \leftarrow \{1, \ldots, p\}$.

while *Model* $\mathcal{M}$ *changes* **do**

  1. Forward Step (Addition)

  Identify best candidate $j^* \in \mathcal{C}$ (min RSS or max partial F).

  if $p\text{-}value_{j^*} < \alpha_{enter}$ **then**

    Add variable: $\mathcal{M} \leftarrow \mathcal{M} \cup \{j^*\}$;

    Remove from candidates: $\mathcal{C} \leftarrow \mathcal{C} \setminus \{j^*\}$;

  else

    Break (No further addition possible);

  2. Backward Step (Pruning)

  Identify worst variable $k^* \in \mathcal{M}$ (max p-value).

  if $p\text{-}value_{k^*} > \alpha_{remove}$ **then**

    Remove variable: $\mathcal{M} \leftarrow \mathcal{M} \setminus \{k^*\}$;

    Return to candidates: $\mathcal{C} \leftarrow \mathcal{C} \cup \{k^*\}$;

return $\mathcal{M}$

This bidirectional procedure enables convergence towards a more parsimonious and stable model. The entry and exit criteria are often based on AIC (*Akaike Information Criterion*) or BIC (*Bayesian Information Criterion*), which we will define in more detail in the section dedicated to criteria.

2.3 Machine Learning

2.3.1 Ridge Regression

Although it is not part of the methods at the center of our analysis, it is relevant to explain what Ridge Regression is. Introduced by Hoerl and Kennard (1970), it is designed to address the limitations of Ordinary Least Squares (OLS) in the presence of multicollinearity. When predictors are highly correlated, the $X^T X$ matrix becomes singular (or nearly so), rendering the OLS estimator unstable and its variance extremely high.

To stabilize the estimation, Ridge Regression imposes a penalty on the sum of squared coefficients (L_2 norm). The minimization problem is written as:

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{2}$$

Where $\lambda \geq 0$ is a parameter controlling the degree of shrinkage. Despite the fact that Ridge Regression reduces estimator variance and improves prediction (at the cost of a slight bias), it presents a major drawback for interpretability: it does not perform variable selection. It shrinks coefficients towards zero without ever setting them exactly to zero. It was to address this limitation that LASSO was introduced.

2.3.2 Least absolute shrinkage and selection operator (LASSO)

LASSO, proposed by Robert Tibshirani (1996), retains the stability of Ridge regression while improving model interpretability. As the author points out in his seminal paper *Regression Shrinkage and Selection via the Lasso* in the Journal of the Royal Statistical Society: Series B (Methodological), the Lasso "...enjoys some of the favourable properties of both subset selection and ridge regression".

The fundamental difference lies in the nature of the penalty. While Ridge uses an L_2 norm, the LASSO uses an L_1 norm. This problem can be formulated as minimizing the squared error subject to a constraint t:

$$\min_{\beta_0,\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq t \tag{3}$$

For computational reasons, the equivalent Lagrangian form is often used:

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{4}$$

This modification radically changes the estimator's behavior. The geometric constraint imposed by the L_1 norm allows the solution to touch the constraint region exactly on the axes.

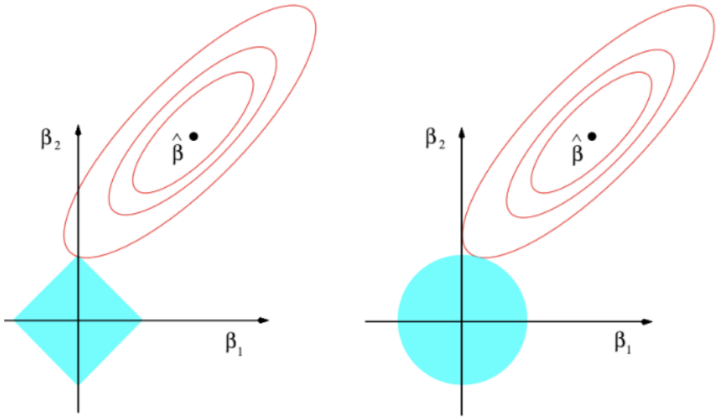


Figure 2: Geometric comparison of constraints: LASSO (left) vs Ridge (right). *Source: The Elements of Statistical Learning, Hastie, Tibshirani, Friedman*

As illustrated above, this geometry allows LASSO to strictly set certain coefficients to zero (those where the optimal solution lies on a vertex), thereby performing automatic and continuous variable selection.

The term λ acts as a sparsity parameter:

- If $\lambda = 0$, we recover the OLS solution (no penalty).
- If λ is very large, all coefficients are shrunk to zero (null model).
- In between, the Lasso selects a subset of variables.

However, the Lasso has limitations. In the presence of highly correlated variables, it tends to select only one arbitrarily. Furthermore, it saturates when the number of predictors p exceeds the number of observations n.

2.3.3 Elastic-Net Regression

Elastic-Net, introduced by Hui Zou and Trevor Hastie (2005), was designed to address the limitations of LASSO and Ridge discussed earlier. It is a hybrid method combining L_1 and L_2 penalties to leverage both sparsity and the grouping effect required to handle correlated variables. In their seminal paper *Regularization and variable selection via the elastic net* (Journal of the Royal Statistical Society: Series B), the authors liken the method to "a stretchable fishing net that retains 'all the big fish' ".

Mathematically, Elastic-Net minimizes the residual sum of squares subject to a dual regularization constraint. The objective function is defined as:

$$\hat{\beta}^{\text{enet}} = \underset{\beta}{\operatorname{argmin}} \left\{ \|y - X\beta\|^2 + \lambda \left(\alpha \sum_{j=1}^{p} \beta_j^2 + (1-\alpha) \sum_{j=1}^{p} |\beta_j| \right) \right\} \tag{5}$$

This formulation highlights the role of the hyperparameters:

- The mixing parameter α ($0 \leq \alpha \leq 1$) determines the trade-off between Ridge ($\alpha = 1$) and LASSO ($\alpha = 0$).
- The L_1 penalty induces sparsity by forcing coefficients to zero.
- The L_2 penalty encourages the grouping effect: unlike LASSO, Elastic-Net tends to assign similar coefficients to correlated variables, selecting or rejecting them as a group ("in or out together").

Geometrically, while Ridge forms a sphere and LASSO a diamond, Elastic-Net creates a constraint region with convex edges and sharp vertices. The vertices enable variable selection, while the edge curvature provides the stability characteristic of Ridge.

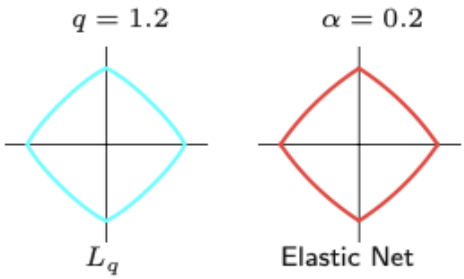


Figure 3: Geometry of constraints: LASSO (diamond), Ridge (circle), and Elastic-Net (intermediate). *Source: The Elements of Statistical Learning, Hastie, Tibshirani, Friedman*

Zou and Hastie note that direct minimization of the above equation, termed the "Naive Elastic Net", results in double shrinkage, which can degrade predictive performance. To correct this bias while preserving variable selection, the final coefficients are rescaled by the quadratic penalty factor:

$$\hat{\beta}^{\text{enet}} = (1 + \lambda_2)\hat{\beta}^{\text{naive}} \tag{6}$$

2.3.4 Least Angle Regression (LARS)

Introduced by Efron, Hastie, Johnstone, and Tibshirani (2004), LARS is presented as a democratic version of Forward Stepwise. Unlike classical methods that incorporate variables abruptly and fully, LARS allows them to enter the model progressively, and "only as much as they deserve".

The core idea is to evolve the coefficients continuously, ensuring that all variables in the active set remain equally correlated with the current residual.

The algorithm starts with a null model (all coefficients at zero) and builds the estimate step-by-step as described below :

Algorithm 5: LARS Algorithm (Intuitive Approach)

Centered/Scaled predictors X, Centered response y **Result:** Sequence of models (Regularization path)

Initialization: All coefficients $\beta = 0$, residual $r = y$. Active set $\mathcal{A} = \emptyset$.

while *variables remain outside* $\mathcal{A}$ **do**

  1. Selection: Identify variable x_j (not in $\mathcal{A}$) with the highest absolute correlation with residual r. Add to active set $\mathcal{A}$.

  2. Progression: Move coefficients of variables in $\mathcal{A}$ jointly towards their Least Squares solution (based on current $X_{\mathcal{A}}$).

  3. Step Size Condition: *Do not take the full step.* Stop exactly when the correlation of a non-active variable x_k "catches up" and becomes equal to that of the active variables.

  4. Update: Record new coefficients β and update residual r.

end

This procedure naturally generates a complete regularization path. To obtain the exact LASSO solution, a specific rule is added: if a coefficient hits zero during the progression step, the variable is dropped from the active set, and the algorithm proceeds without it.

2.4 Model Selection Criteria

As mentioned previously, relying solely on minimizing the Residual Sum of Squares (SSE) for optimal model selection will inevitably lead to retaining the most complex model. To bypass this overfitting problem, it is necessary to use metrics that penalize the addition of explanatory variables or that estimate error on unobserved data.

We decided to group these tools into four categories: goodness-of-fit measures, statistical significance tests, information criteria derived from likelihood, and finally predictive criteria based on cross-validation.

2.4.1 Goodness-of-Fit Measures

These indicators measure the proportion of variance of the target variable explained by the model.

The Coefficient of Determination (R^2) R^2 mechanically increases with the addition of variables, even if they are only noise.

$$R^2 = 1 - \frac{SSE}{SST} \tag{7}$$

where SSE is the Sum of Squared Errors and SST is the Total Sum of Squares.

Adjusted R-squared (R^2_{adj}) To correct the inflation of R^2, Theil (1961) introduced a version including a penalty related to degrees of freedom. It is defined by:

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1} \tag{8}$$

Unlike R^2, R^2_{adj} can decrease if the addition of a variable does not provide enough information to compensate for the loss of degrees of freedom. This is a first measure of parsimony.

2.4.2 Significance Tests

These tests are fundamental as they constitute the decision-making engine of classical iterative algorithms (Stepwise, Forward, Backward).

Student's t-test (t-test) It tests the individual significance of a coefficient ($H_0 : \beta_j = 0$). In a *Backward* procedure, it is this test that identifies the least significant variable to remove (the one with the highest p-value).

$$t_j = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)} \tag{9}$$

Partial Fisher's F-test (F-test) It allows comparing two nested models to determine whether the addition of a variable (or a group) significantly reduces the error. It is the main entry criterion in a *Forward* procedure.

$$F = \frac{(SSE_{reduced} - SSE_{full})/(p_{full} - p_{reduced})}{SSE_{full}/(n - p_{full} - 1)} \tag{10}$$

2.4.3 Information Criteria

These criteria rely on the maximization of the model's log-likelihood, denoted $\ln(L)$, penalized by a function of the number of parameters p and the sample size n.

Generally, the likelihood $L(\theta)$ measures the probability of observing the data $(y_1, \ldots, y_n)$ for a given set of parameters θ. Under the assumption of independence of observations, it is defined by the product of probability densities f:

$$L(\theta|y) = \prod_{i=1}^{n} f(y_i; \theta) \tag{11}$$

In the specific context of linear regression under the assumption of normality of errors, maximizing this likelihood is equivalent to minimizing the sum of squared errors.

Akaike Information Criterion (AIC) This criterion was proposed in Akaike's seminal paper (1973); it measures information loss and seeks to minimize:

$$AIC = -2\ln(L) + 2p \quad \approx \quad n\ln\left(\frac{SSE}{n}\right) + 2p \tag{12}$$

AIC prioritizes predictive efficiency but is asymptotically inconsistent: it tends to overparameterize the model when n is large, as the $2p$ penalty becomes negligible.

Corrected AIC Criterion (AICC) Hurvich and Tsai (1989) propose a correction term to compensate for AIC bias on small samples:

$$AICC = AIC + \frac{2p(p+1)}{n-p-1} \tag{13}$$

Schwarz Bayesian Criterion (BIC or SBC) BIC was introduced by Schwarz (1978); this criterion seeks to maximize the posterior probability of the model. Note that SAS software uses the acronym SBC (Schwarz Bayesian Criterion), but it is strictly the same metric as BIC.

$$BIC = -2\ln(L) + p\ln(n) \tag{14}$$

The fundamental difference lies in the penalty: $p\ln(n)$. Since $\ln(n) > 2$ as soon as $n \geq 8$, BIC penalizes complexity more severely than AIC. It therefore favors more parsimonious models and is consistent (it identifies the true model if $n \to \infty$).

2.4.4 Predictive Criteria and Cross-Validation

The true validation of a model lies in its ability to predict *out-of-sample* data.

Mallows' C_p Criterion C_p was introduced by Mallows (1973); it estimates the standardized Mean Squared Prediction Error (MSPE).

$$C_p = \frac{SSE_p}{\hat{\sigma}^2} - (n - 2p) \tag{15}$$

We look for a model where $C_p \approx p$. If $C_p \gg p$, the model suffers from significant bias (underfitting).

PRESS Statistic (Leave-One-Out) The PRESS statistic (Prediction Sum of Squares), formalized by Allen (1974), is a form of cross-validation where each observation is predicted by a model trained on all others.

$$PRESS = \sum_{i=1}^{n} (y_i - \hat{y}_{i,-i})^2 \tag{16}$$

A model with a low SSE but a high PRESS indicates overfitting: the model "memorizes" the data but does not generalize.

K-Fold Cross-Validation As PRESS is computationally expensive, the K-Fold method is often preferred; its theoretical properties were extensively studied by Stone (1974). The sample is divided into K blocks. The model is trained on $K-1$ blocks and tested on the remaining block. The operation is repeated K times.

$$CV_{(K)} = \frac{1}{K} \sum_{k=1}^{K} MSE_k \tag{17}$$

It is this robust metric (generally with $K = 5$ or 10) that we will use to calibrate the hyperparameters of our Machine Learning models.

3 Data Generating Process

In the context of our simulations, we have established 4 distinct scenarios. We will start with the Gaussian DGP, which is the benchmark, then we will consider the case of multicollinearity, outliers, and structural breaks. We will conclude with mixed scenarios to cover cases closer to reality.

For each scenario, we generated $MC = 1,000$ replications of a dataset containing $N = 200$ observations and $P = 50$ explanatory variables.

3.1 DGP 1 : Gaussian (Benchmark)

This first scenario serves as a baseline for comparison. It follows the classical assumptions of the linear model: independence of regressors and normality of the errors. The explanatory variables X are drawn from a standard normal distribution $\mathcal{N}(0, 1)$ and are mutually independent (the variance-covariance matrix is the identity I_{50}).

The model is written as:

$$Y = X\beta + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 0.1^2)$. The coefficient vector β is defined such that only the first 6 variables are significant:

$$\beta_{1-6} = (1, -0.35, 0.15, 0.27, 0.57, -0.14)'$$

The remaining coefficients (β_7 to β_{50}) are zero.

Algorithm 6: Data Generation Process: Gaussian Case

Initialization: $N \leftarrow 200$, $P \leftarrow 50$, $MC \leftarrow 1000$

Define $\beta \leftarrow (1, -0.35, 0.15, 0.27, 0.57, -0.14, 0, \ldots, 0)^T$

Define $\mu \leftarrow \mathbf{0}_P$ and $\Sigma \leftarrow I_P$

for $k \leftarrow 1$ **to** MC **do**

- Generate matrix $X \sim \mathcal{N}(\mu, \Sigma)$ of size $(N \times P)$
- Generate noise vector $\epsilon \sim \mathcal{N}(0, 0.1^2)$
- Compute $Y \leftarrow X\beta + \epsilon$
- Store dataset (Y, X) for iteration k

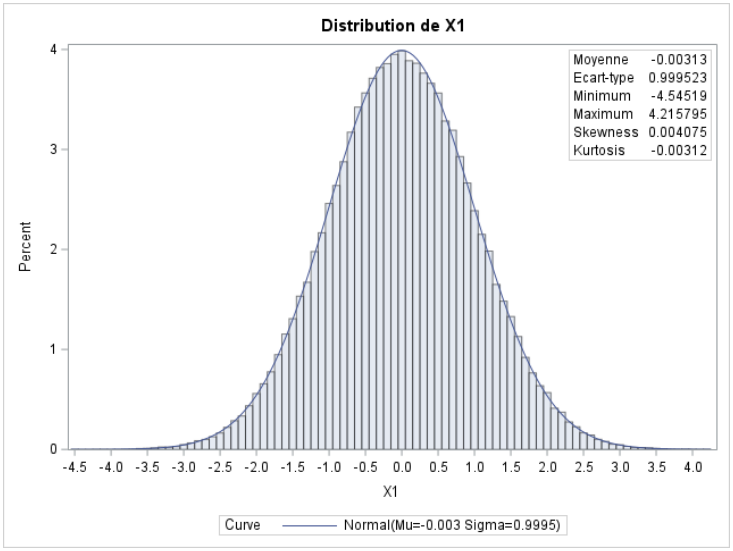


Figure 4: X_1 following a standard normal distribution.

3.2 DGP 2 : Multicollinearity

In this scenario, we introduce internal linear dependence among the active variables. Since real-world data is often correlated, we want to verify if the algorithms can still distinguish the true predictors in this context.

The correlation structure is applied internally to the first 6 variables via a Toeplitz matrix, while the remaining variables (noise) remain independent. The variance-covariance matrix Σ is defined in blocks:

- For X_1 to X_6, the internal correlations follow the structure $r = (1, 0.8, 0.75, 0.7, 0.65, 0.6)$.
- For X_7 to X_{50}, the matrix is the identity.

The vector β and the error term ϵ remain identical to the Benchmark.

Algorithm 7: Data Generation Process: Multicollinearity

Initialization: $N \leftarrow 200$, $P \leftarrow 50$, $MC \leftarrow 1000$

Define $\beta \leftarrow (1, -0.35, 0.15, 0.27, 0.57, -0.14, 0, \ldots, 0)^T$

Define $\mu \leftarrow \mathbf{0}_P$

Define correlation structure $r \leftarrow (1, 0.8, 0.75, 0.7, 0.65, 0.6)$

Construct Covariance Matrix Σ:

    Block 1..6: Toeplitz matrix based on r

    Block 7..50: Identity matrix I_{44} (Independence)

for $k \leftarrow 1$ **to** MC **do**

    Generate $X \sim \mathcal{N}(\mu, \Sigma)$

    Generate $\epsilon \sim \mathcal{N}(0, 0.1^2)$

    Compute $Y \leftarrow X\beta + \epsilon$

    Store dataset

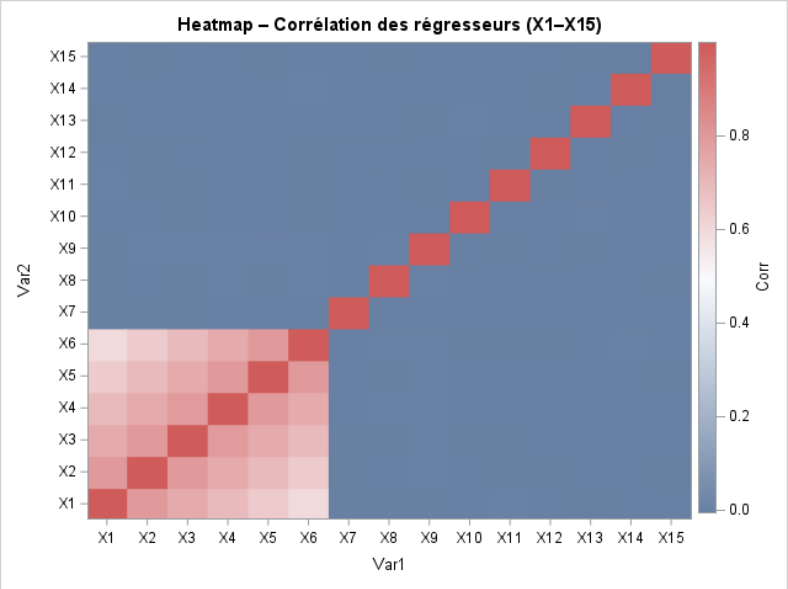


Figure 5: Heatmap showing strong correlations between the first 6 regressors (red block).

La procédure CORR

10 Variables :	X1 X2 X3 X4 X5 X6 X7 X8 X9 X10

Coefficients de corrélation de Pearson, N = 200000

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
X1	1.00000	0.80099	0.75005	0.70113	0.65182	0.60166	0.00110	0.00267	0.00218	0.00011
X2	0.80099	1.00000	0.80121	0.75032	0.70219	0.65233	0.00173	-0.00004	0.00337	-0.00093
X3	0.75005	0.80121	1.00000	0.79976	0.75157	0.70156	0.00152	0.00166	0.00494	0.00215
X4	0.70113	0.75032	0.79976	1.00000	0.80069	0.75011	0.00039	0.00419	0.00391	-0.00061
X5	0.65182	0.70219	0.75157	0.80069	1.00000	0.80203	0.00091	0.00270	0.00345	0.00082
X6	0.60166	0.65233	0.70156	0.75011	0.80203	1.00000	0.00037	0.00363	0.00424	0.00142
X7	0.00110	0.00173	0.00152	0.00039	0.00091	0.00037	1.00000	0.00064	-0.00111	-0.00085
X8	0.00267	-0.00004	0.00166	0.00419	0.00270	0.00363	0.00064	1.00000	0.00394	-0.00116
X9	0.00218	0.00337	0.00494	0.00391	0.00345	0.00424	-0.00111	0.00394	1.00000	-0.00356
X10	0.00011	-0.00093	0.00215	-0.00061	0.00082	0.00142	-0.00085	-0.00116	-0.00356	1.00000

Figure 6: Pearson correlation coefficients.

3.3 DGP 3 : Outliers

The goal here is to test the robustness of the methods when the data contains extreme values. Unlike the pure Gaussian case, the explanatory variables X do not follow a single standard normal distribution.

We applied contamination via a mixture of distributions directly to the X matrix. For each observation i and each variable j, the value is generated according to the following rule:

$$X_{i,j} \sim \begin{cases} \mathcal{N}(0,1) & \text{with probability } p = 0.9 \\ \mathcal{N}(4,1) & \text{with probability } p = 0.1 \end{cases}$$

This means that approximately 10% of the data are outliers drawn from a shifted distribution (mean of 4). The response variable Y is then computed classically via $Y = X\beta + \epsilon$.

Algorithm 8: Data Generation Process: Outliers (Contamination)

Initialization: $N \leftarrow 200$, $P \leftarrow 50$, $MC \leftarrow 1000$

Define $\beta \leftarrow (1, -0.35, 0.15, 0.27, 0.57, -0.14, 0, \ldots, 0)^T$

for $k \leftarrow 1$ **to** MC **do**

  Generate $X_{raw} \sim \mathcal{N}(0, I_P)$

  for $i \leftarrow 1$ **to** N **do**

    for $j \leftarrow 1$ **to** P **do**

      Draw $u \sim \mathcal{U}(0, 1)$

      if $u > 0.90$ **then**

        $X_{i,j} \leftarrow$ draw from $\mathcal{N}(4, 1)$

      else

        $X_{i,j} \leftarrow X_{raw}[i, j]$

  Generate $\epsilon \sim \mathcal{N}(0, 0.1^2)$

  Compute $Y \leftarrow X\beta + \epsilon$

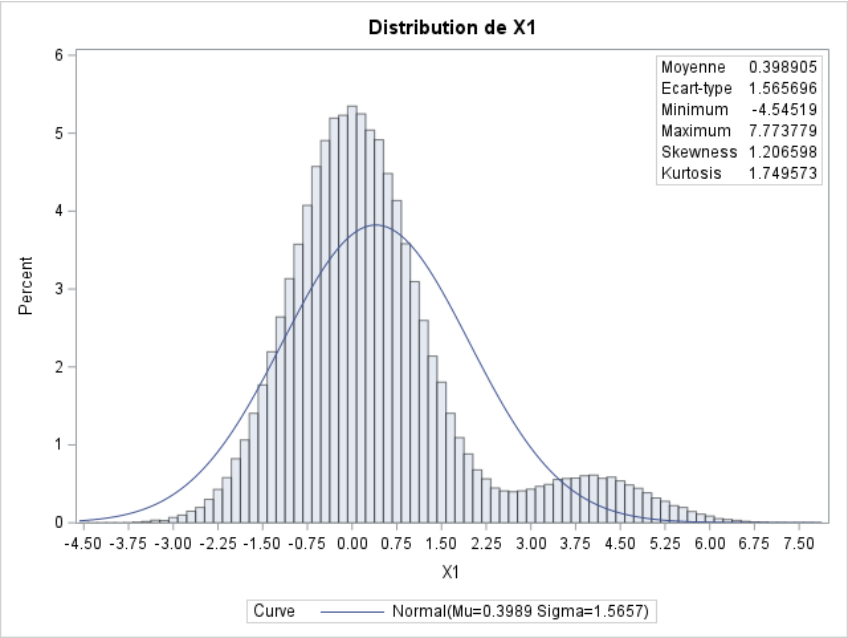


Figure 7: Distribution of X_1 with outliers.

3.4 DGP 4: Structural Break

This final scenario simulates a regime change within the data. Unlike the previous models where the relationship between X and Y is stable, here the coefficient vector β changes abruptly at a random time t_{break}.

The break point t_{break} is drawn uniformly between 10% and 70% of the sample.

- Before the break ($i \leq t_{break}$): The model relies on 6 active variables with the vector $\beta_1 = (1, -0.35, 0.15, 0.27, 0.57, -0.14)'$.
- After the break ($i > t_{break}$): The relationship changes, a 7th variable becomes active, and some signs are inverted or modified : $\beta_2 = (-1, 0.55, 0.6, -0.35, 0.15, 0.45, 0.1)'$.

The goal is to observe how the algorithms react to these sudden changes, whether the selection procedures favor the first regime, the second, or a mixture of both.

Algorithm 9: Data Generation Process: Structural Break

Initialization: $N \leftarrow 200$, $P \leftarrow 50$, $MC \leftarrow 1000$

Define β_1 (Pre-break coefficients with 0.15 at pos 3)
Define β_2 (Post-break coefficients)
for $k \leftarrow 1$ **to** MC **do**
    $t_{break} \leftarrow \text{RandInt}(0.1N, 0.7N)$
    Generate $X \sim \mathcal{N}(0, I_P)$ and $\epsilon \sim \mathcal{N}(0, 0.1^2)$
    Initialize vector Y of size N
    for $i \leftarrow 1$ **to** N **do**
        if $i \leq t_{break}$ **then**
            $Y_i \leftarrow X_i\beta_1 + \epsilon_i$
        else
            $Y_i \leftarrow X_i\beta_2 + \epsilon_i$
    Store dataset with break index t_{break}

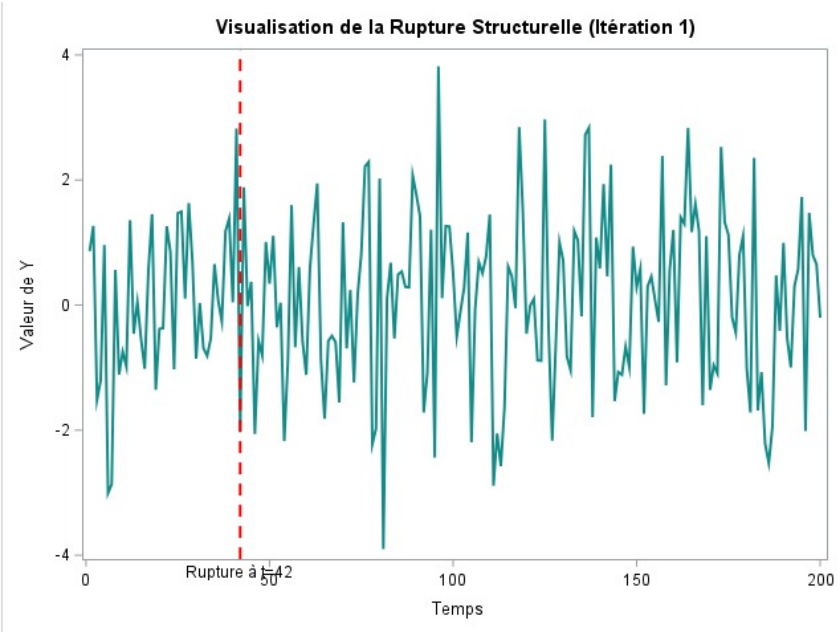


Figure 8: Visualization of the structural break in the time series Y.

3.5 DGP 5: Correlation and Outliers

This scenario combines multicollinearity and outliers. The problem is that if we just multiply the contaminated data by a standard correlation matrix, it tends to "normalize" the distributions and might destroy the outliers we created.

To fix this, we use the Iman-Conover method. It is a technique based on ranks: we reorder the values of our contaminated data in ordre to have the desired correlation structure (Toeplitz), but we keep the exact values (including the outliers) we generated at the start.

Algorithm 10: Mixed DGP: Correlation + Outliers

Initialization: $N \leftarrow 200$, $P \leftarrow 50$, Target Matrix C

for $k \leftarrow 1$ **to** MC **do**

- Generate X_{indep} with outliers (90% Normal, 10% Shifted)
- Compute ranks of X_{indep} columns
- Apply Cholesky decomposition of C to the scores
- Reorder X_{indep} columns to match the new rank order
- $X_{final} \leftarrow$ Combine modified columns (1-6) with the rest (7-50)
- Generate $\epsilon \sim \mathcal{N}(0, 0.1^2)$
- Compute $Y \leftarrow X_{final}\beta + \epsilon$

3.6 DGP 6: Structural Break and Outliers

This scenario tests if the methods can handle both unstable coefficients and bad data points. We first generate the predictors X with the same contamination as before (mixture of normals).

Then, we introduce the structural break. The response Y is simulated using β_1 for the first part of the sample (before the break) and β_2 for the second part (before the break). The structural break happens at a random time t_{break}.

Algorithm 11: Mixed DGP: Structural Break + Outliers

Initialization: $N \leftarrow 200$, $P \leftarrow 50$

Define β_1 (Pre-break) and β_2 (Post-break)

for $k \leftarrow 1$ **to** MC **do**

  Generate $X_{raw} \sim \mathcal{N}(0, I_P)$

  $X \leftarrow$ Apply Outliers module to X_{raw}

  $t_{break} \leftarrow \text{RandInt}(0.1N, 0.7N)$

  Generate $\epsilon \sim \mathcal{N}(0, 0.1^2)$

  for $i \leftarrow 1$ **to** N **do**

    if $i \leq t_{break}$ **then**

      $Y_i \leftarrow X_i\beta_1 + \epsilon_i$

    else

      $Y_i \leftarrow X_i\beta_2 + \epsilon_i$

3.7 DGP 7: Correlation and Structural Break

Here, we combine the structural break with multicollinearity. The idea is to see if the variable selection algorithms get confused when variables are correlated and their effect on Y changes over time.

We use the Iman-Conover function here as well to induce the Toeplitz correlation on the first 6 variables before generating the response variable with the regime change.

Algorithm 12: Mixed DGP: Correlation + Structural Break

Initialization: $N \leftarrow 200$, $P \leftarrow 50$, Target Matrix C

for $k \leftarrow 1$ **to** MC **do**

  Generate independent $X_{indep} \sim \mathcal{N}(0, I_P)$

  $X_{corr} \leftarrow \text{ImanConover}(X_{indep}[, 1:6], C)$

  $X_{final} \leftarrow$ Combine X_{corr} and remaining columns

  $t_{break} \leftarrow \text{RandInt}(0.1N, 0.7N)$

  Generate $\epsilon \sim \mathcal{N}(0, 0.1^2)$

  for $i \leftarrow 1$ **to** N **do**

    if $i \leq t_{break}$ **then**

      $Y_i \leftarrow X_{final}[i]\beta_1 + \epsilon_i$

    else

      $Y_i \leftarrow X_{final}[i]\beta_2 + \epsilon_i$

3.8 DGP 8: The Combined Scenario

This is the most complex scenario. It aggregates all the difficulties: the variables are correlated, they contain outliers, and the regression coefficients change during the period.

In the simulation, the order of operations is important:

1. We generate the outliers.
2. We reorder them to force correlation (Iman-Conover).
3. We generate Y with the structural break.

Algorithm 13: Complete Mixed DGP (Corr + Outliers + Break)

Initialization: $N \leftarrow 200$, $P \leftarrow 50$, Matrix C

for $k \leftarrow 1$ **to** MC **do**
- Generate $X_{indep} \sim \mathcal{N}(0, I_P)$
- $X_{out} \leftarrow$ Apply Outliers(X_{indep})
- $X_{final} \leftarrow$ ImanConover(X_{out}, C)
- $t_{break} \leftarrow$ RandInt$(0.1N, 0.7N)$
- Generate $\epsilon \sim \mathcal{N}(0, 0.1^2)$
- **for** $i \leftarrow 1$ **to** N **do**
  - **if** $i \leq t_{break}$ **then**
    - $Y_i \leftarrow X_{final}[i]\beta_1 + \epsilon_i$
  - **else**
    - $Y_i \leftarrow X_{final}[i]\beta_2 + \epsilon_i$

3.9 Metrics

To evaluate the performance of the variable selection algorithms, we compared, for each iteration k, the set of variables selected by the model (denoted $\hat{S}_k$) with the set of true active variables of the generating process (denoted S_{true}).

This comparison relies on the calculation of two quantities defined in our simulation code:

- **False Positives (FP):** We take all selected variables and remove those that are part of the true model. We are left with the noise.

$$FP = |\hat{S}_k \setminus S_{true}|$$

- **False Negatives (FN):** We take the true variables and remove those that were selected. We are left with those that were forgotten.

$$FN = |S_{true} \setminus \hat{S}_k|$$

Based on these two counts, each iteration result is classified into one of the following four categories:

- **Perfect Fitting:** The algorithm recovers exactly the variables of the true model, without addition or omission.

$$FP = 0 \quad \text{and} \quad FN = 0$$

- **Overfitting:** The model selects all relevant variables but also includes noise variables.

$$FP > 0 \quad \text{and} \quad FN = 0$$

- **Underfitting:** The model selects only a subset of the true variables and no false variable. This category includes the case where the selected model is empty or very parsimonious; it retained only correct but insufficient signals.

$$FP = 0 \quad \text{and} \quad FN > 0$$

- **Wrong Model:** The model is "mixed": it misses some true variables of the model while including incorrect variables (noise).

$$FP > 0 \quad \text{and} \quad FN > 0$$

The probabilities associated with each metric are estimated by calculating the frequency of appearance of each case over the $MC = 1000$ repetitions.

Since every iteration falls into exactly one of these four scenarios, the sum of the calculated probabilities is always equal to 1.

Algorithm 14: Performance Metrics Calculation

Input: Simulated dataset (Y, X) with MC iterations, Set of true variables S_{true}

Output: Estimated probabilities for the 4 metrics (Perfect, Over, Under, Wrong)

$N_{Perfect} \leftarrow 0$, $N_{Over} \leftarrow 0$, $N_{Under} \leftarrow 0$, $N_{Wrong} \leftarrow 0$
for $k \leftarrow 1$ **to** MC **do**
    Apply selection procedure on sample k
    Retrieve the set of selected variables $\hat{S}_k$
    Compute False Positives: $FP_k \leftarrow |\hat{S}_k \setminus S_{true}|$
    Compute False Negatives: $FN_k \leftarrow |S_{true} \setminus \hat{S}_k|$
    if $FP_k = 0$ ***and*** $FN_k = 0$ **then**
        $N_{Perfect} \leftarrow N_{Perfect} + 1$
    else if $FP_k > 0$ ***and*** $FN_k = 0$ **then**
        $N_{Over} \leftarrow N_{Over} + 1$
    else if $FP_k = 0$ ***and*** $FN_k > 0$ **then**
        $N_{Under} \leftarrow N_{Under} + 1$
    else
        $N_{Wrong} \leftarrow N_{Wrong} + 1$
$P(Perfect) \leftarrow N_{Perfect}/MC$
$P(Overfit) \leftarrow N_{Over}/MC$
$P(Underfit) \leftarrow N_{Under}/MC$
$P(Wrong) \leftarrow N_{Wrong}/MC$
return *Vector of 4 probabilities*

4 Simulation and analysis

4.1 DGP 1 : Gaussian (Benchmark)

4.1.1 Statistical Learning Methods

The Forward method exhibits overfitting when the decision is based on significance threshold (SL) rules or unpenalised predictive criteria. The inclusion of random variables is inevitable during the iteration of sequential tests, resulting in an overfitting rate close to 1. The predictive criterion "cross-validation" exacerbate this tendency when not limited by a complexity penalty, as they may select uninformative variables that only slightly improve out-of-sample prediction.

However, incorporating Schwarz's rule (SBC) as a stopping criterion helps to stabilize the selection process. This is because it encourages the use of parsimonious models that, regardless of the selection criterion, reduce the over-selection bias inherent in progressive methods.

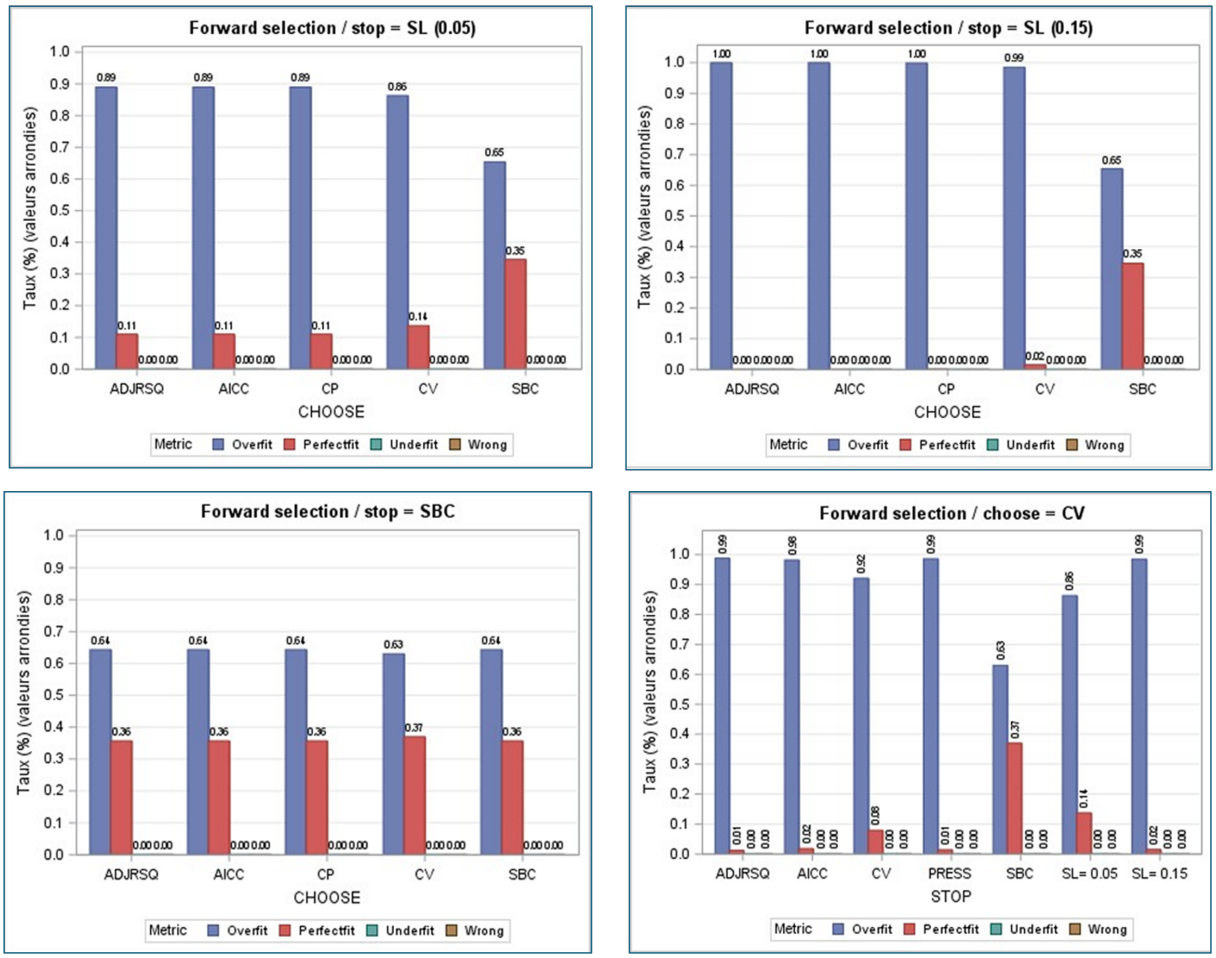


Figure 9: Forward selection for gaussian simulation

The backward procedure generally shows overfitting (close to 100%) for all selection criteria, because, given that this approach starts from an overfitted model, it does not sufficiently penalize dimensionality to remove noisy variables. Only the SBC-SBC combination leads to effective model contraction, mitigating overfitting and generating approximately 28.6% perfect fit.

The stepwise method stands out for its remarkable stability: the perfect fit rate is generally 38 to 40%, while overfitting is around 60%, regardless of criteria. This consistency contrasts with the forward method, which appeared sensitive to explicit penalties. This consistency is due to the add-drop selection algorithm, which offers the possibility of correcting selection errors : random variables can be eliminated in subsequent steps. Stepwise therefore functions as a tacit sanction of complexity.

In an orthogonal context, the stepwise approach therefore appears to be structurally more robust than the forward approach, whose effectiveness relies crucially on the stopping rule.

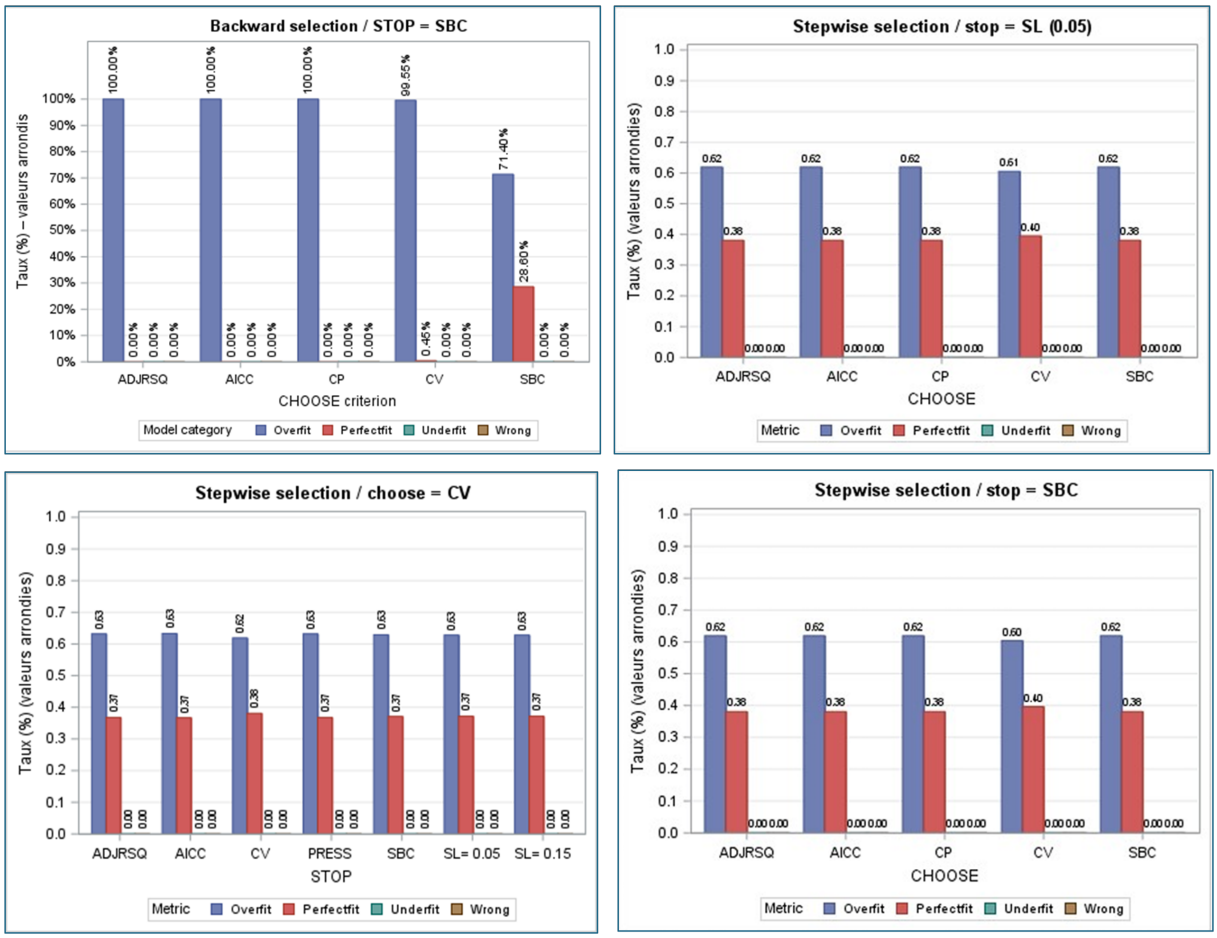
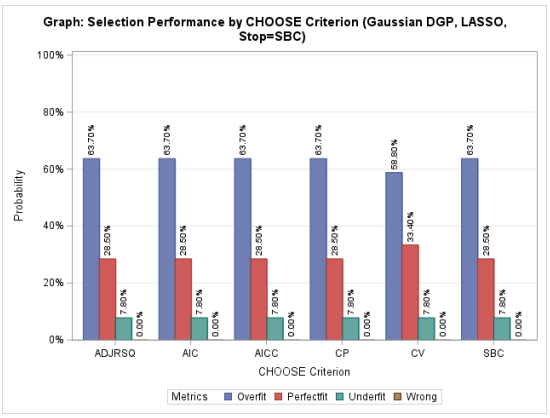


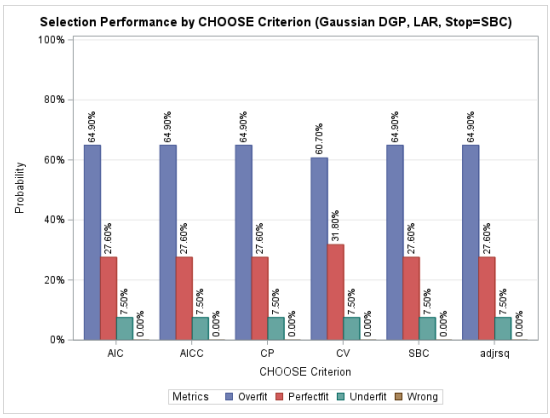
Figure 10: Backward/Stepwise selection for gaussian simulation

4.1.2 Machine Learning Methods

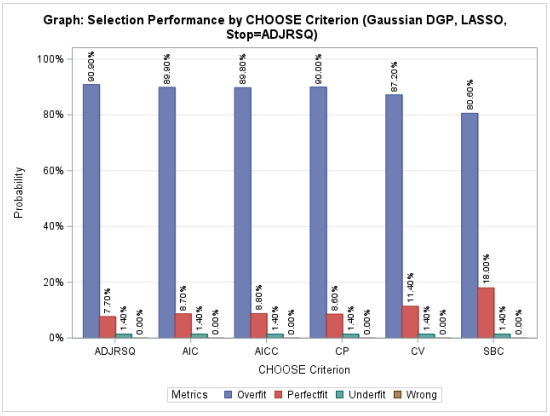
In the context of a Gaussian DGP, the evaluated methods (LASSO, LAR and Elastic Net) achieve similar perfect fitting scores. LASSO appears to perform slightly better than LAR, achieving a perfect fit rate of 33.40% (Stop=SBC and Choose=CV), followed by LAR, which achieves 31.80% (Stop=SBC and Choose=CV). This highlights that the Stop=SBC criterion is better suited to these two methods.



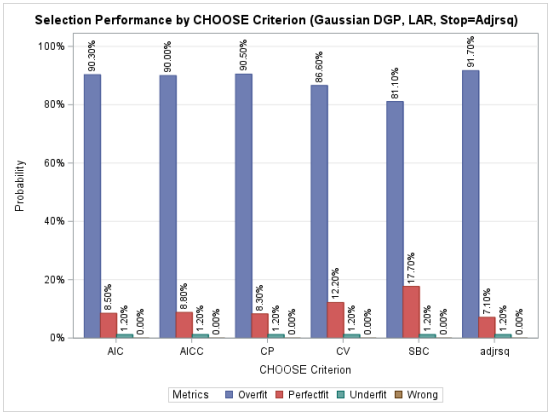
(a) LASSO (Stop = SBC)



(b) LAR (Stop = SBC)



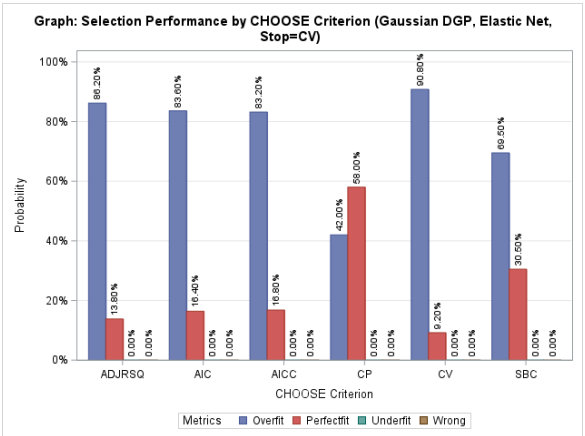
(c) LASSO (Stop = Adjusted R^2)



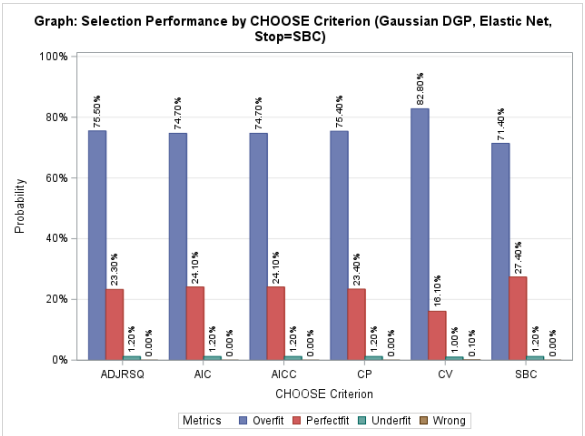
(d) LAR (Stop = Adjusted R^2)

Figure 11: LASSO and LAR in Gaussian DGP

For K-Fold Cross Validation (CV) as the stopping criterion, the Elastic Net achieves 58% with Choose = Cp and 30.50% with choose = SBC.



(a) Elastic Net (Stop = CV)



(b) Elastic Net (Stop = SBC)

Figure 12: Elastic Net (Gaussian DGP)

Furthermore, the probability of underfitting is almost zero for Gaussian DGP (<1%). But overfitting rates are very high. The simulations demonstrate that in the case of Stop = Adjrsq (R^2), the overfitting exceeds 90%. The adjusted R^2 does not sufficiently penalize the entry of spurious variables for all three methods. In contrast, Stop = SBC (BIC) is the most parsimonious criterion, reducing the probability of overfitting to around 60-65% and maximizing the perfect fit rates for LASSO and LAR. Stop=SBC combined with Choose=CV offers the best perfect fit rates. While Choose SBC does not produce the most optimal results.

4.2 DGP 2 : Multicollinearity

4.2.1 Statistical Learning Methods

The Forward method demonstrates performance almost identical to the one observed in Gaussian DGP. When the stopping rule is the Schwarz's criterion, the percentage of perfect fit remains between 33-36% and that of overfitting around 64-67%, with no apparent signs of underfitting. The correlation between explanatory variables therefore does not affect the selection process, it makes it more convenient to add noise variables that are correlated with the relevant variables, but does not change the SBC's ability to handle the complexity of the model. As in the benchmark, the unpenalized predictive criteria or significance thresholds remain excessively permissive, leading to considerable overfitting.

In the context of intern multicollinearity, the Stepwise approach performs almost identically to the process of generating independent Gaussian data, with approximately 35% perfect fit and 65% overfitting. The interdependence between variables does not interfere with the identification of the support. As the algorithm is bidirectional, the structure of the results is probably influenced by its dynamics rather than by the choice of selection criteria.

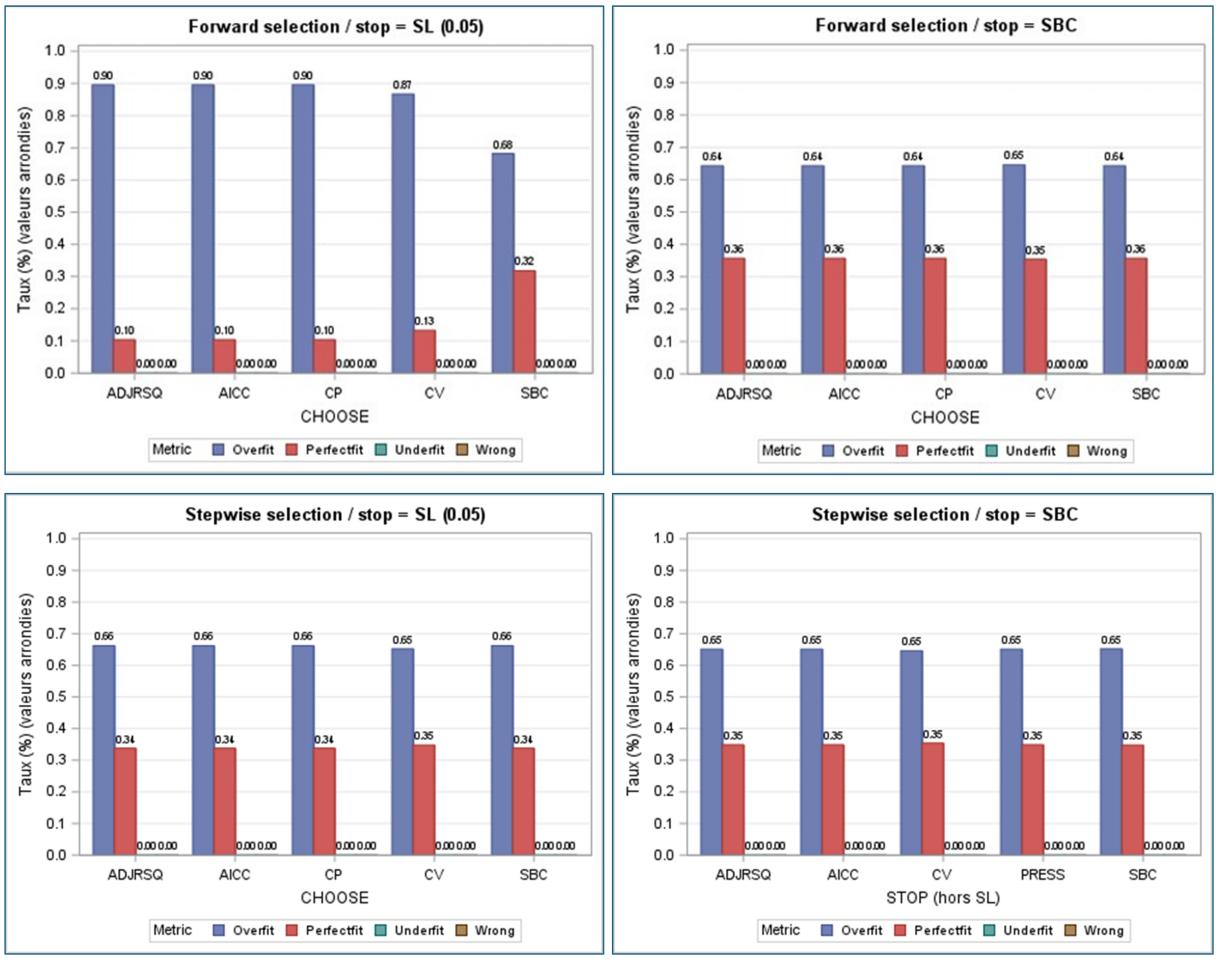
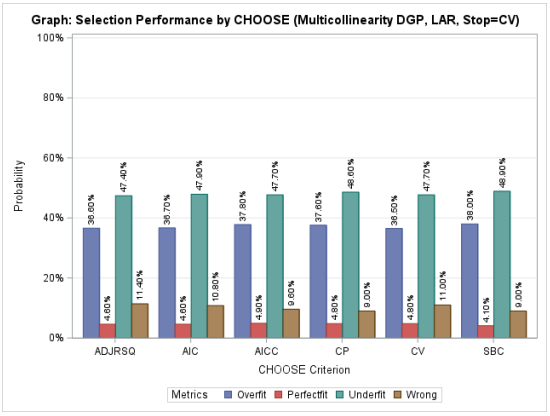


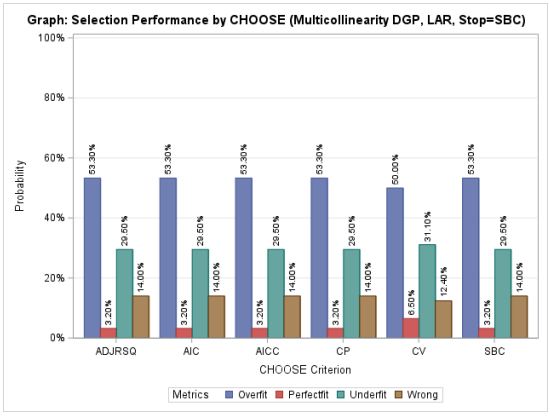
Figure 13: Forward/Stepwise selection for intern multicollinearity simulation

4.2.2 Machine Learning Methods

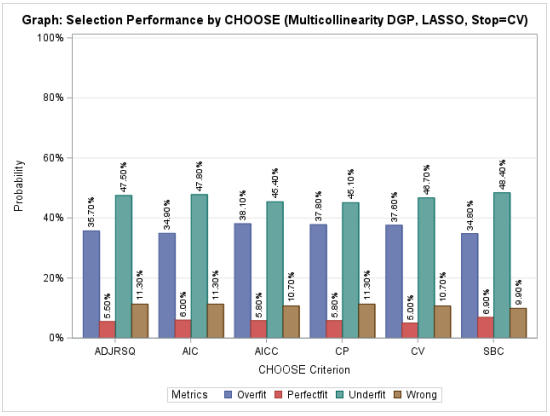
This section demonstrates the impact of internal correlation between the true variables of the model. Internal multicollinearity disrupts the identification of the signal for L1 type penalties. Consequently, the introduction of multicollinearity causes a drop of perfect fit rates for both LAR and LASSO, falling below 10%. Because LAR and LASSO tend to select only one variable from a highly correlated group. This leads to high underfitting rates. In addition, the STOP = CV (K-fold Cross Validation) criterion becomes inefficient for those two methods, resulting in underfitting rates of nearly 50%.



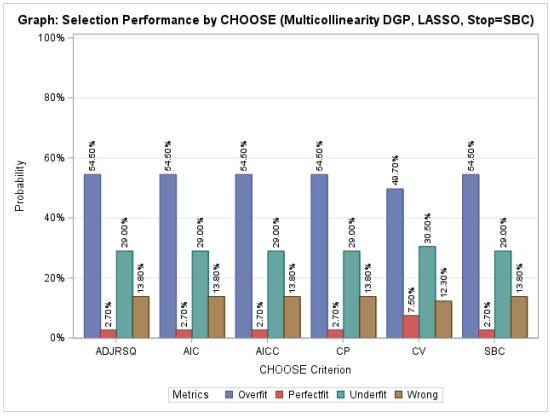
(a) LAR (Stop = CV)



(b) LAR (Stop = SBC)



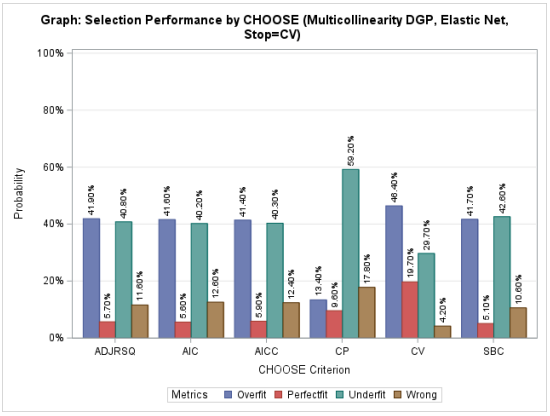
(c) LASSO (Stop = CV)



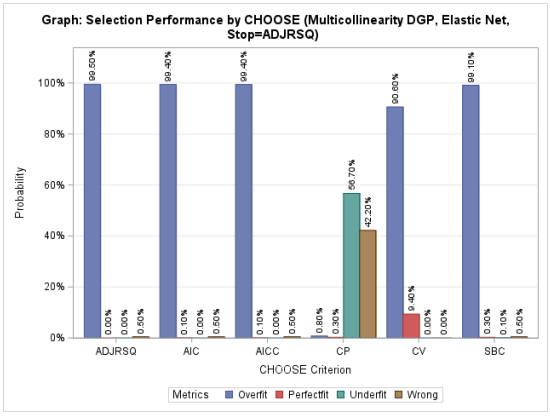
(d) LASSO (Stop = SBC)

Figure 14: LAR and LASSO (Multicollinearity DGP)

However, the Elastic Net remains efficient in this case, reaching 19,70% of perfect fit rates (Stop=CV Choose=CV), because unlike LASSO, the L2 type penalties of the Elastic Net forces the algorithm to include the entire group of correlated variables. Finally, it can observed that the Mallows Cp criterion when applied to the Elastic Net combined to a AICc or Adjusted R^2 stopping criterion, leads to important rates of underfit (58 %).



(a) Elastic Net (Stop = CV)



(b) Elastic Net (Stop = ADJRSQ)

Figure 15: Elastic Net (Multicollinearity DGP)

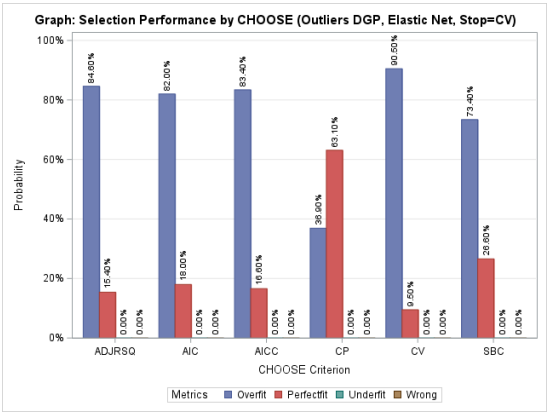
4.3 DGP 3 : Outliers

4.3.1 Statistical Learning Methods

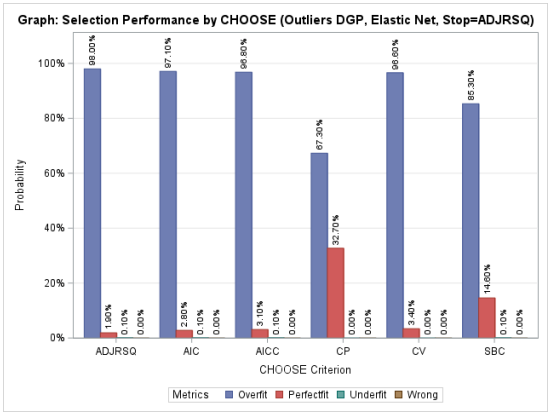
In the presence of extreme values, both Forward and Stepwise exhibit almost exactly the same behavior as in the Gaussian benchmark. Outliers increase the variability of the regressors but do not create structural confusion between relevant and irrelevant variables, so true predictors continue to be selected while noise variables remain prone to over-selection. Consequently, the balance between perfect fit and overfitting is unchanged and remains driven by the same mechanisms: permissive sequential tests, predictive criteria favoring richer models, and the dominant role of the SBC penalty in controlling model complexity.

4.3.2 Machine Learning Methods

The addition of outliers significantly impacts the selection dynamics: Elastic net outperforms other methods, reaching a perfect fit rate of 63,10% (Stop=CV / Choose=CP).



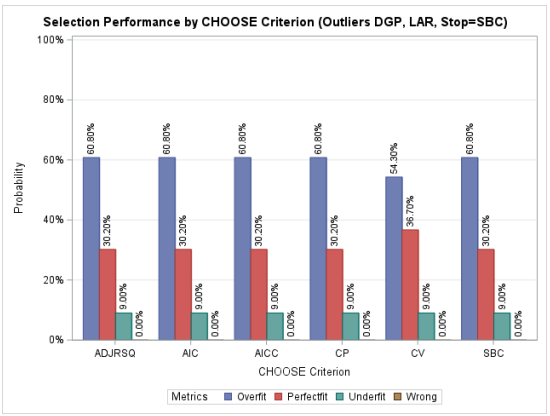
(a) Elastic Net (Stop = CV)



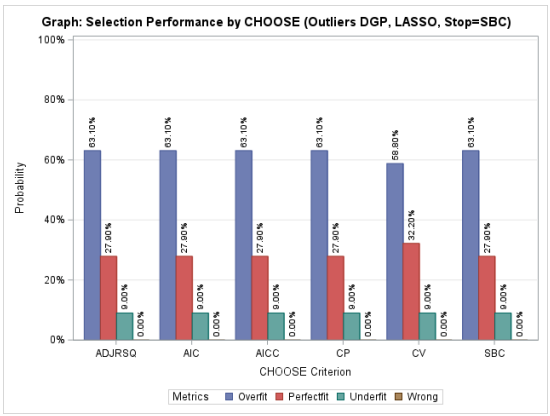
(b) Elastic Net (Stop = Adjusted R^2)

Figure 16: Elastic Net (Outliers DGP)

LAR and LASSO achieve perfect fit rates of 36.70% and 32.20% , respectively (Stop=SBC Choose=CV).



(a) LAR (Stop = SBC)



(b) LASSO (Stop = SBC)

Figure 17: LAR and LASSO (Outliers DGP)

Paradoxically, LAR and Elastic Net achieve higher perfect fit rates than in the Gaussian case because the noise forces the selection criteria to become more conservative. However, this leads to an increase in the underfitting which reaches 9% for LAR and LASSO (using Stop=SBC). Elastic net keeps a very low underfitting probability (0.10%).

4.4 DGP 4 : Structural Break

4.4.1 Statistical Learning Methods

Forward selection is almost entirely ineffective in cases of structural break. The percentage of perfect fits is always below 2%, while the proportion of wrong models usually exceeds 60% and frequently rises to 80–90%, regardless of the selection rule applied.

Criteria based on the p-value (SL) prove particularly ineffective. In fact, the instability of the coefficients from one model to another makes the entry tests uninformative, resulting in irregular selection choices. Severe penalty criteria such as SBC slightly reduce the wrong rate of the models, but this comes at the cost of significant underfitting, with the elimination of many active variables. Cross-validation-based selection are the most unstable, as interruption makes average predictive performance less representative. This instability is amplified by the sequential and irreversible nature of Forward selection.

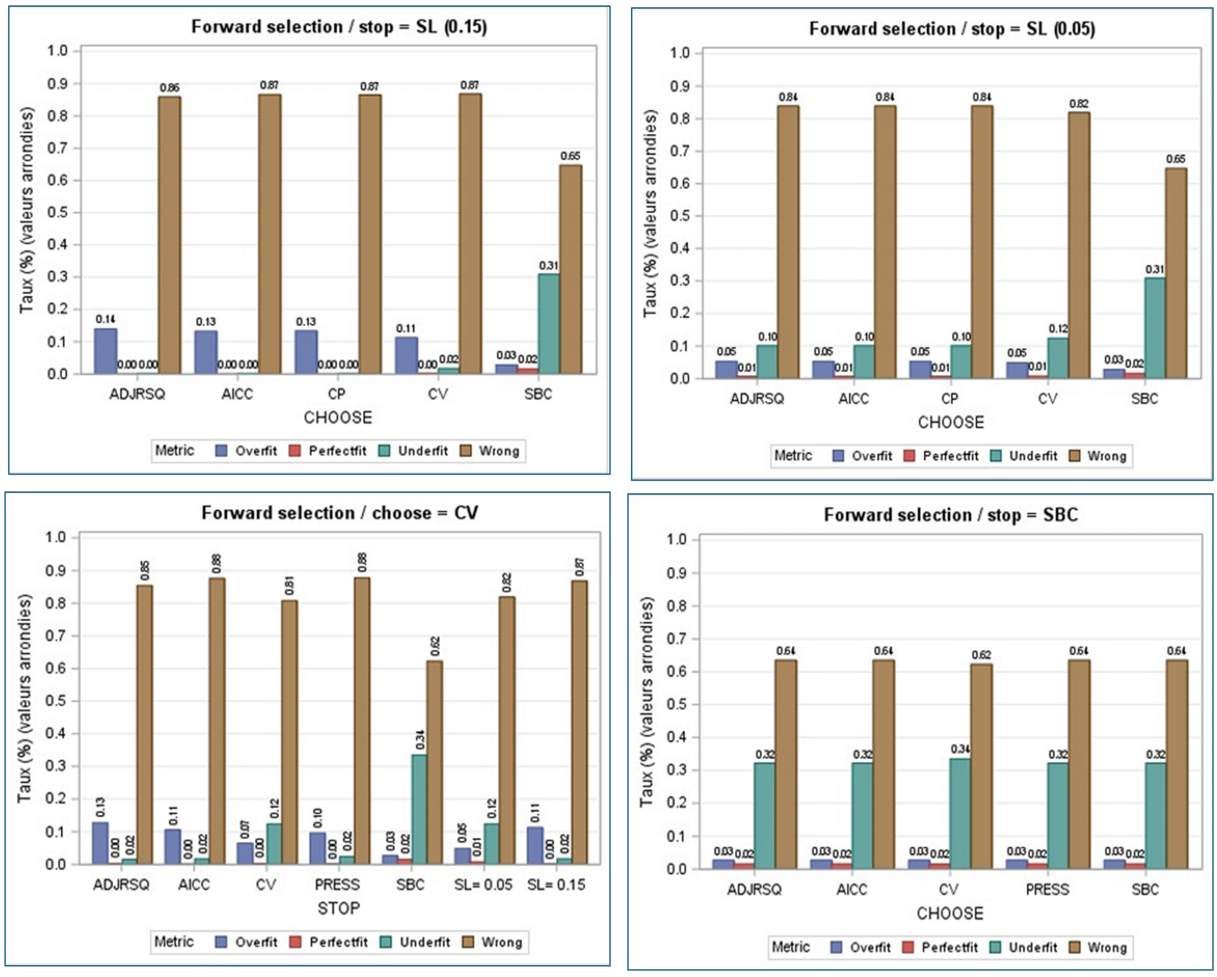


Figure 18: Forward selection for structural break simulation

The Backward method combines overfitting, underfitting, and wrong models. From the saturated model, it retains a large number of irrelevant variables, while incompletely removing those whose impact fluctuates depending on the regime. This results in a mixture of false positives and false negatives, leading to a particularly high proportion of erroneous models, even with the use of the SBC penalty.

Stepwise proves to be relatively more effective. Its add-drop mechanism offers the possibility of correcting some of the previous and subsequent errors, thereby reducing the proportion of incorrect models. However, it is unable to remedy the intrinsic instability

of the support. In fact, variables whose influence changes over time remain difficult to identify, and underfitting persists.

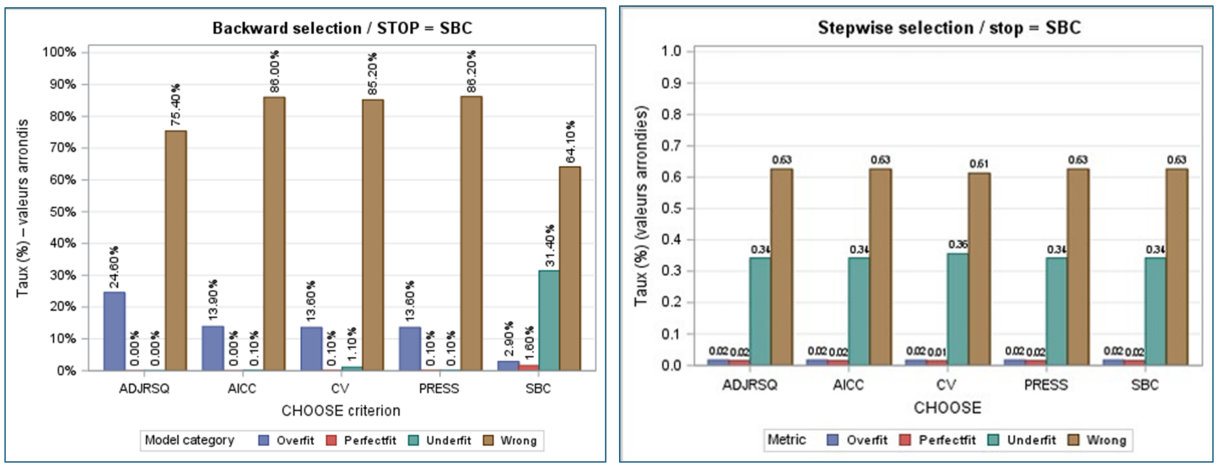
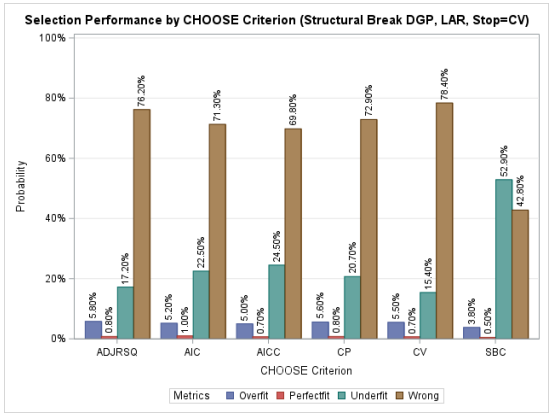


Figure 19: Backward/Stepwise selection for structural break simulation

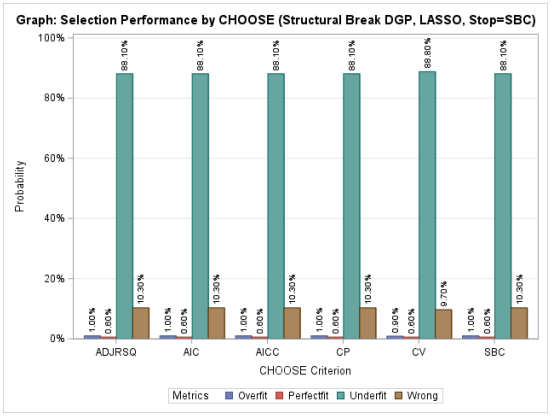
4.4.2 Machine Learning Methods

In this structural break DGP, the perfect fit rate drops to 0% for most evaluated methods. We mostly observe higher rates of a wrong classification (a combination of underfitting and overfitting) and significant underfitting. This is due to the shifting beta coefficients between the two regimes preventing the algorithms from distinguishing the real signal of the true variables from the noise, resulting in an incomplete or incorrect variable selection.

In the case of the LAR (stop = K-fold Cross Validation), the failure of the prediction criteria can clearly be observed. The probability of having a wrong model reaches 70-80 %. LASSO (stop= SBC) leads to high underfitting rates 88,10 % due to the strict penalties of SBC criterion, so the instability of the coefficients is translated into an absence of signal and most of the variables are not selected.



(a) LAR (Stop = CV)



(b) LASSO (Stop = SBC)

Figure 20: LAR and LASSO (Structural Break DGP)

Finally, the Elastic Net (Stop=CV) also shows a considerable probability of having the wrong model (81,10 %) proving that even the L2 type penalties are insufficient to reconcile the parameter shifts of the structural break.

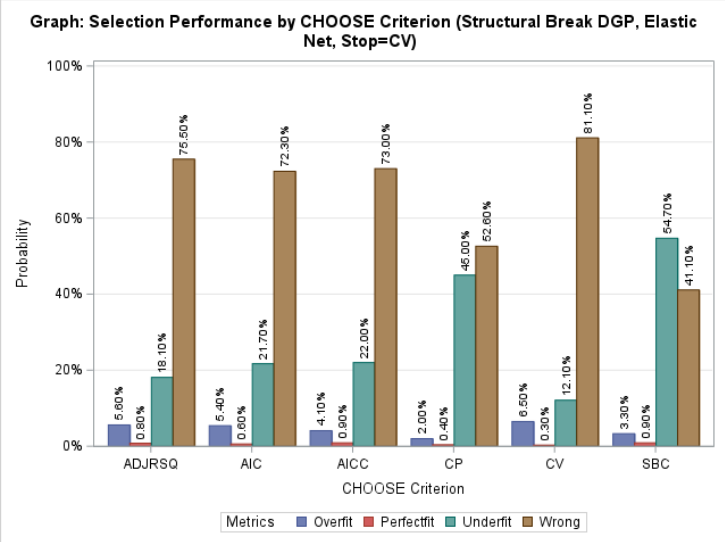


Figure 21: Elastic Net (Stop = CV)

4.5 DGP 5 : Mix Multicollinearity & Outliers

4.5.1 Statistical Learning Methods

The performance of the Forward and Stepwise methods remains constant regardless of the selection criteria and stopping rules applied. Indeed, the proportion of perfect matches is around 35-36%, while overfitting is close to 64-65%. There is no underfitting or incorrect models. This invariance indicates that different criteria empirically lead to the same final models, so it is likely that the structure of the data makes various models practically equivalent in terms of fit.

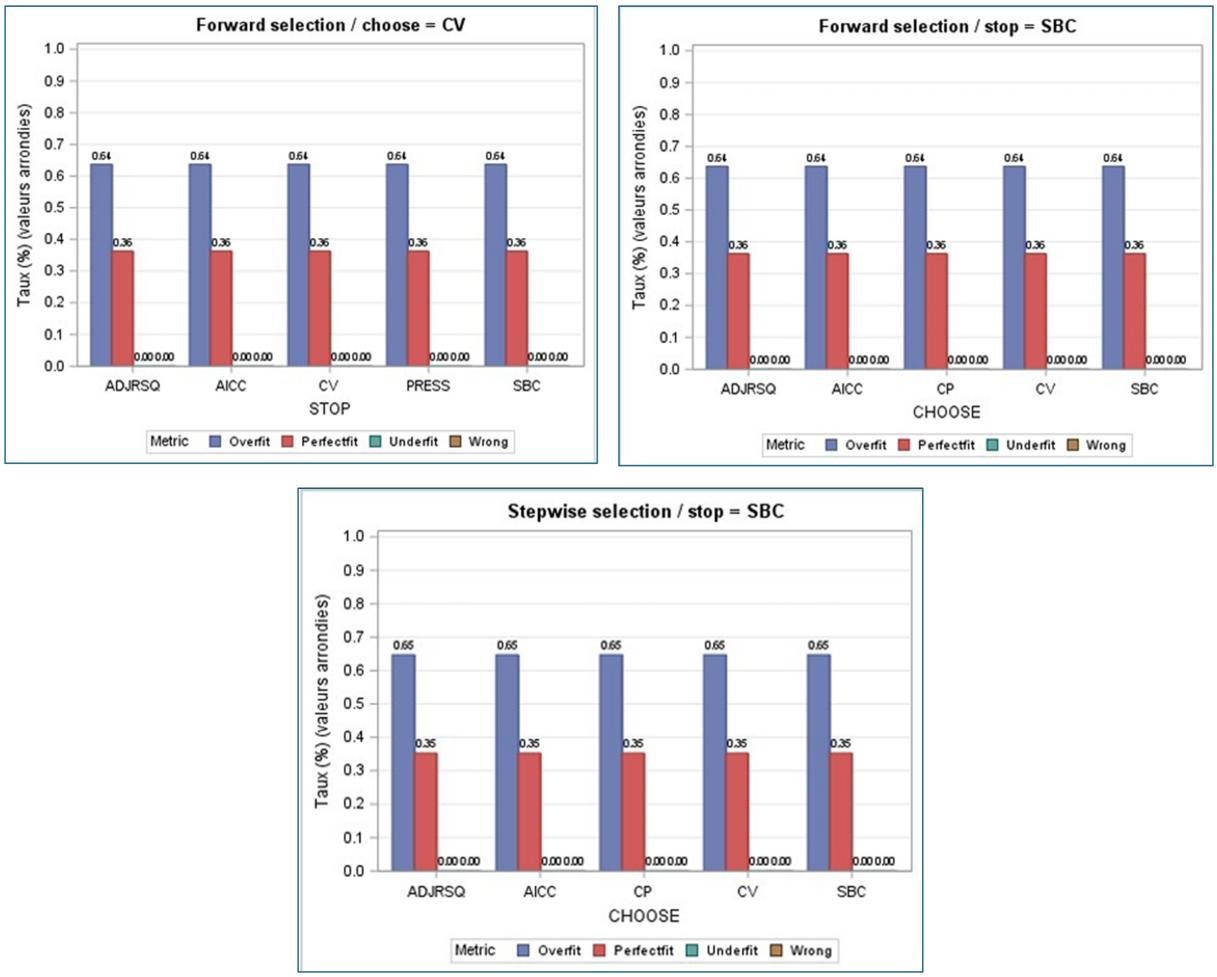
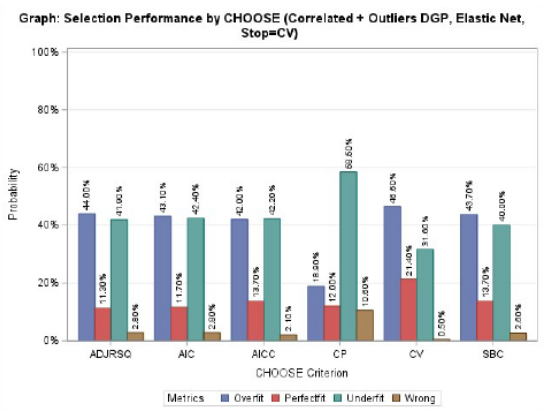


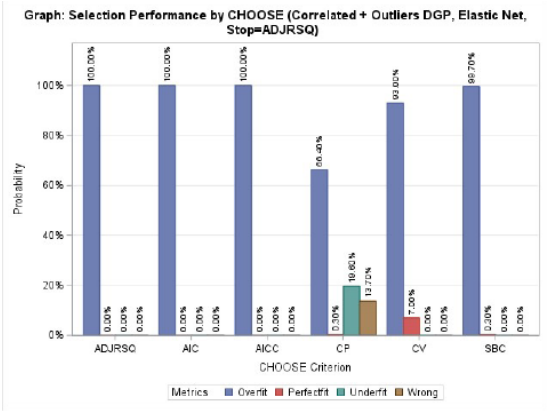
Figure 22: Forward/Stepwise selection for multicollinearity and outliers simulation

4.5.2 Machine Learning Methods

In this section, the Elastic Net with the Stop=CV Choose=CV configuration demonstrates a superior stability with the highest rates of perfect fit (24.4%) compared to the other two methods. This is due to the Ridge (L_2) which helps stabilize the coefficients given the correlation, unlike LASSO (L_1). The mallows' C_p criterion leads to severe underfitting (reaching 58.5% for the Elastic Net (Stop=CV)) which confirms that this criterion fails to manage the bias-variance tradeoff in this DGP.



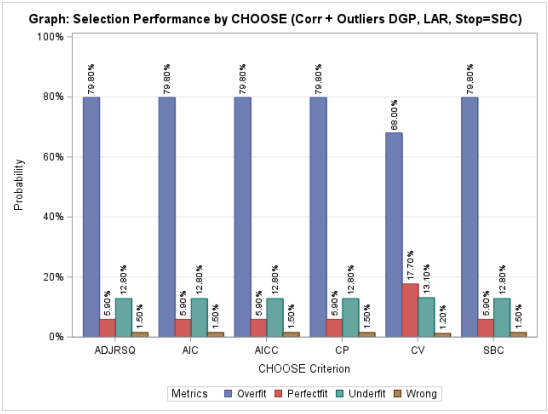
(a) Elastic Net (Stop = CV)



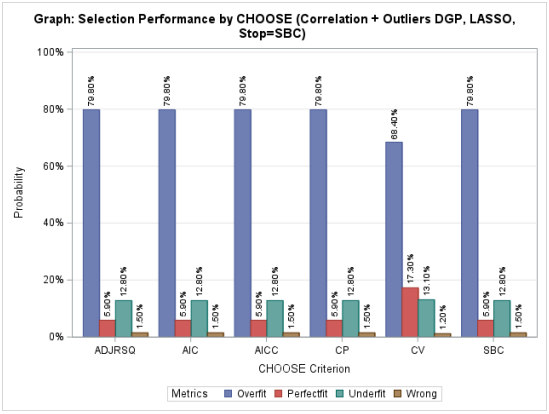
(b) Elastic Net (Stop = Adjusted R^2)

Figure 23: Elastic Net (Correlation + Outliers DGP)

In contrast, LAR and LASSO show perfect fit rates of 17.7% and 17.3% respectively for the Stop=SBC and Choose=CV configuration. the Adjusted R^2 and AICC as stopping rules are a total failure, the overfitting rates exceed 90%.



(a) LAR (Stop = SBC)



(b) LASSO (Stop=SBC)

Figure 24: LAR and LASSO (Correlation + Outliers DGP)

4.6 DGP 6 : Mix Structural Break & Outliers

4.6.1 Statistical Learning Methods

For this simulation, the Forward procedure is kind-of-neutralized. The rate of perfect matches drops to about 0.3%, while over 80% of the models are classified as wrong and over 12% as being underfitted. The fact that all combinations of criteria lead to exactly the same results indicates that the selection is totally controlled by the structure of the DGP and not by the decision rule.

Stepwise method shows a decrease in the wrong model rate but remains severely degraded. The perfect match rate remains low, around 1.5%, underfitting is considerable of 38%, and

most models are incorrect, around 59%. The mechanism is not capable of reconstructing a support that changes over time. As with Forward, the selection criteria seem to have no impact.

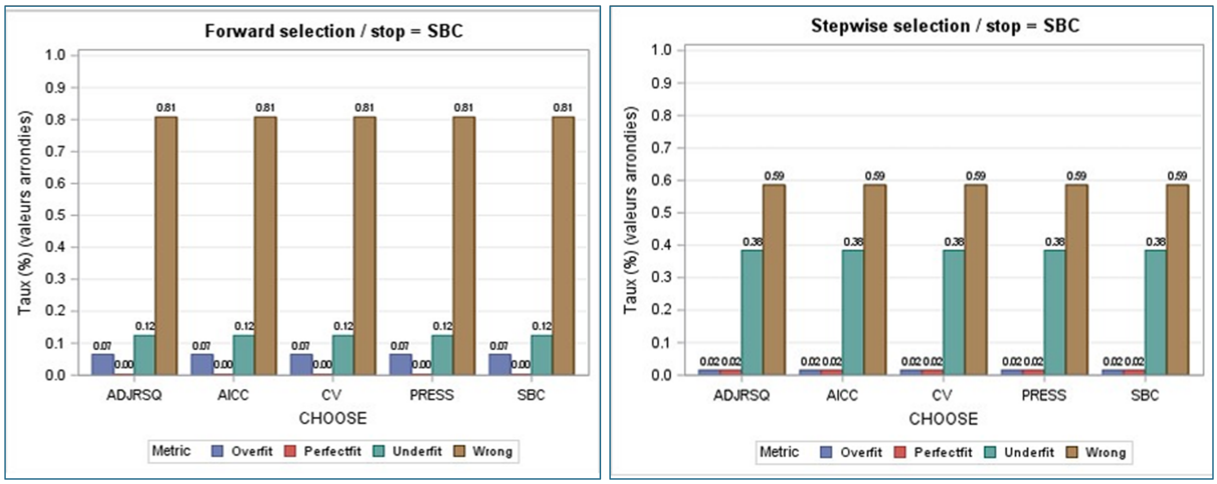


Figure 25: Forward/Stepwise selection for break and outliers simulation

4.6.2 Machine Learning Methods

In this scenario, the rate of perfect fit is near 0% for all three methods. The Elastic Net shows high rates of underfit using SBC as a stopping criterion leads to 91-97.6%. The severe penalty of the SBC combined with the high variance caused by outliers, forces the algorithm to reject most of the variables. Instead, using K-fold cross validation results in a high probability of selecting the wrong model.

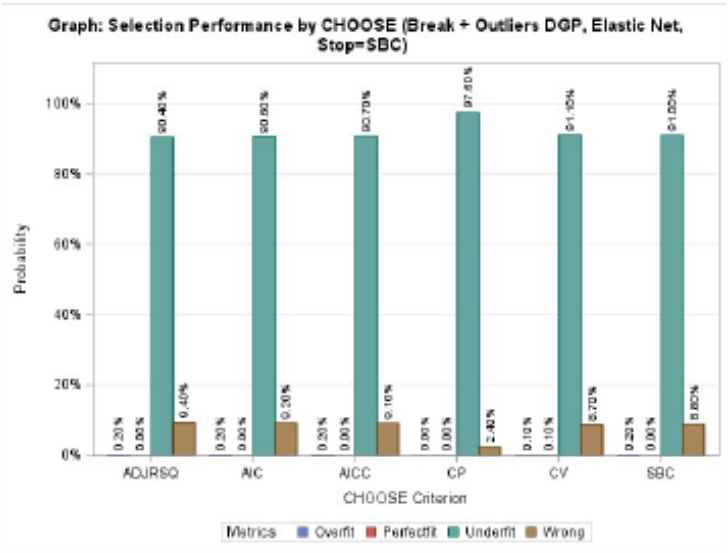
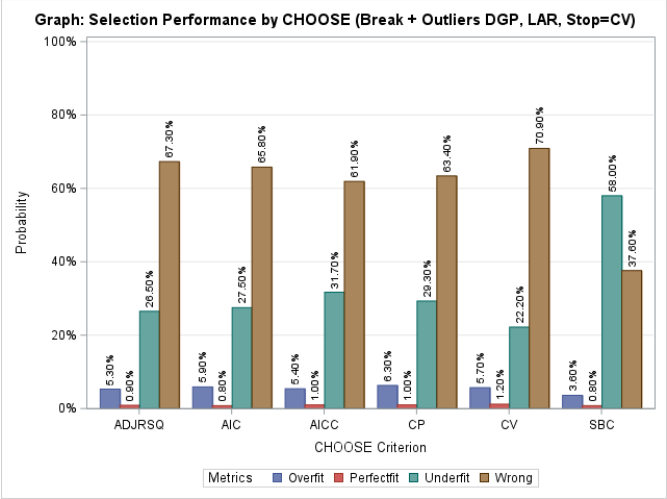


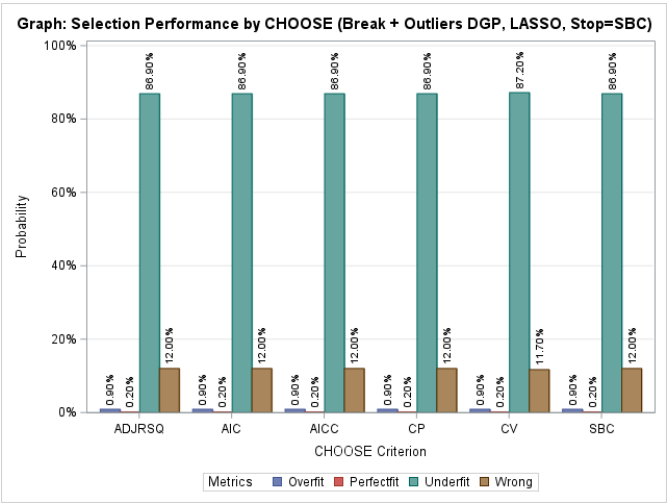
Figure 26: Elastic Net (Stop = SBC)

LAR produces models with significant probabilities of having both over and underfitting

(70.9%, with Stop=CV) because, the presence of outliers falsify the prediction error calculation, causing the LAR to include irrelevant variables while missing the true variables impacted by the break. LASSO shows a high probability of underfitting (88.1%) because the outliers hide the signal, so the LASSO no longer sees any significant variables.



(a) LAR (Stop = CV)



(b) LASSO (Stop = SBC)

Figure 27: LAR and LASSO (Break + Outliers DGP)

4.7 DGP 7 : Mix Multicollinearity & Structural Break

4.7.1 Statistical Learning Methods

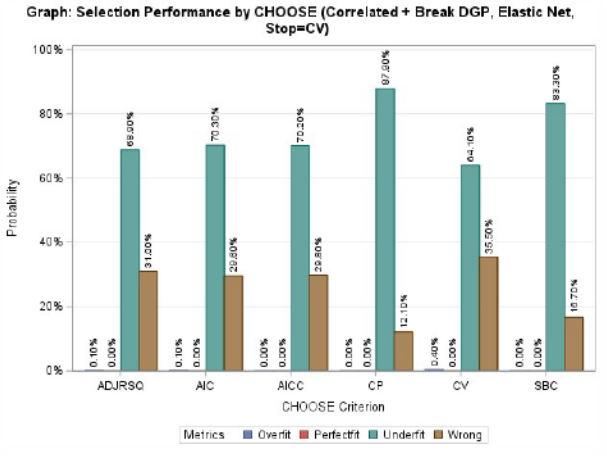
For a DGP combining structural discontinuity and internal correlation, the Forward and Stepwise methods fail to accurately identify the support. The selection is dominated by wrong and underfitted models, even with a strong penalty. The Stepwise method can slightly reduce errors compared to the Forward method, but the break remains the main factor in selection failure.

4.7.2 Machine Learning Methods

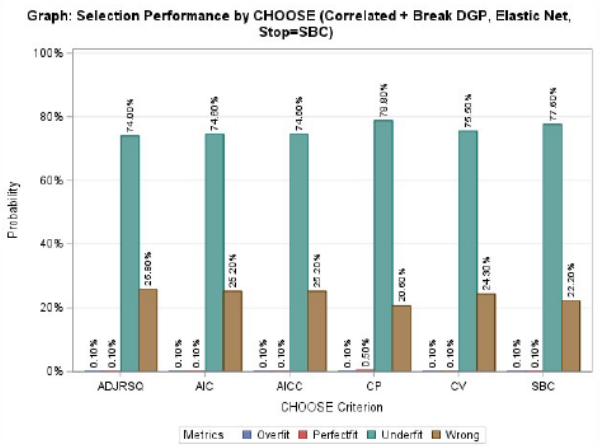
This DGP that combines multicollinearity and structural break highlights the limits of these selection methods. Most of the configurations for the three methods lead to a nearly 0% perfect fit rate.

In this scenario, the LASSO and LAR show similar results to those in the structural break DGP, struggling to reconcile the parameter changes.

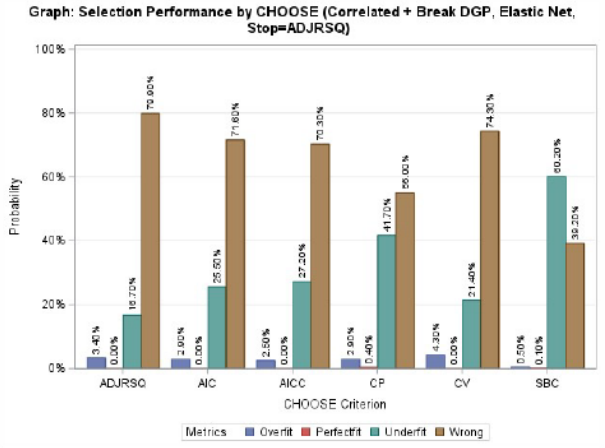
The Elastic Net, even using K-fold cross validation (CV) as a stopping criterion, leads to high underfitting (87.9 %), showing that it's inefficient when correlation and structural instability are combined. Using SBC as a stopping criterion leads to a high underfit rate (78%), because of its severe penalties. But it limits the probability of selecting the wrong fit (20-25%) Conversely, using the Adjusted R^2 leads to a wrong fit probability of nearly 80%, which proves that this criterion should not be used in cases involving both correlation and structural breaks; it fails to distinguish between true signal shifts and noise.



(a) Stop = CV



(b) Stop = SBC



(c) Stop = Adjusted R^2

Figure 28: Elastic Net Performance (Correlation + Structural Break)

5 Empirical Analysis

We now turn to an empirical evaluation. The goal of this section is to compare the behavior of classical statistical approaches against modern regularization techniques on real-world data.

In this empirical analysis, we will use the "Diabetes" dataset, it contains data on 442 diabetes patients ($n = 442$). The response variable (Y) is a quantitative measure of disease progression one year after baseline.

5.1 Exploratory Data Analysis (EDA)

Before moving on to the modelling stage, we carried out an exploratory analysis to understand the structure of the data. We identified two main issues that mirror the scenarios discussed in the previous section.

First, we detected multicollinearity. The matrix below reveals extremely strong relationships between some variables (for example, between S1 and S2 we have $r = 0.90$).

R										
	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6
AGE	1.00	0.17	0.19	0.34	0.26	0.22	-0.08	0.20	0.27	0.30
SEX	0.17	1.00	0.09	0.24	0.04	0.14	-0.38	0.33	0.15	0.21
BMI	0.19	0.09	1.00	0.40	0.25	0.26	-0.37	0.41	0.45	0.39
BP	0.34	0.24	0.40	1.00	0.24	0.19	-0.18	0.26	0.39	0.39
S1	0.26	0.04	0.25	0.24	1.00	0.90	0.05	0.54	0.52	0.33
S2	0.22	0.14	0.26	0.19	0.90	1.00	-0.20	0.66	0.32	0.29
S3	-0.08	-0.38	-0.37	-0.18	0.05	-0.20	1.00	-0.74	-0.40	-0.27
S4	0.20	0.33	0.41	0.26	0.54	0.66	-0.74	1.00	0.62	0.42
S5	0.27	0.15	0.45	0.39	0.52	0.32	-0.40	0.62	1.00	0.46
S6	0.30	0.21	0.39	0.39	0.33	0.29	-0.27	0.42	0.46	1.00

Figure 29: Correlation Matrix - Empirical Analysis

Second, after standardizing the variables to facilitate comparison, the boxplots show several extreme values, in particular for BMI and the S2 variable, which exhibit the most extreme outliers, while the S6 variable displays outliers in both directions.

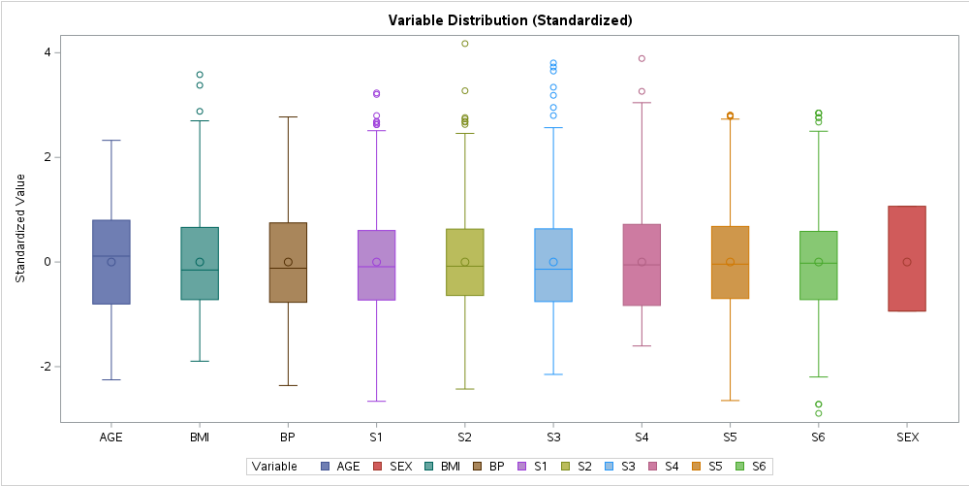


Figure 30: Boxplots - Empirical Analysis

Standardization is essential here so that the LASSO or Elastic Net penalties are applied fairly across all variables, regardless of their unit of measurement.

5.1.1 Choice of algorithms and criteria

We chose to restrict our empirical analysis to three of the methods analyzed previously. This decision aims to contrast three distinct philosophies given the characteristics of our dataset.

Stepwise represents the classical econometric approach. It will allow us to assess whether, with a sample of this size and a small number of variables, iterative methods based on significance tests remain effective compared with regularization approaches. For this method, we will use the AIC and SBC (BIC) information criteria.

LASSO, as the first regularization method to have been established, uses an L_1 norm which, in theory, yields a more readable and interpretable solution.

Elastic Net can be viewed as an extension of LASSO. The latter has weaknesses in the presence of multicollinearity. Since Elastic Net was specifically designed to address this issue via its "grouping effect", it will allow us to retain the information contained in correlated variables.

For the last two methods, we will rely on cross-validation (CV).

5.2 Discussion of results

We divided the dataset into training (70%) and validation (30%) sets. For the Stepwise procedures, the model selection relies on minimizing the Validation-ASE. For the regularization methods (LASSO and Elastic Net), we implemented a hybrid strategy: using

the Schwarz Bayesian Criterion (SBC) as a stopping rule for parcimony, combined with 10-fold Cross-Validation (CV) to validate the choosing step.

The results, summarized in Table 2, show that the choice of algorithm alters the model structure.As expected, the Stepwise-SBC approach is parsimonious ($SEX, BMI, BP, S3, S5$), whereas the AIC criterion includes $S1$ at step 6, slightly improving the ASE.

However, the comparison between LASSO and Elastic Net une the SBC clearly highlights their differences..

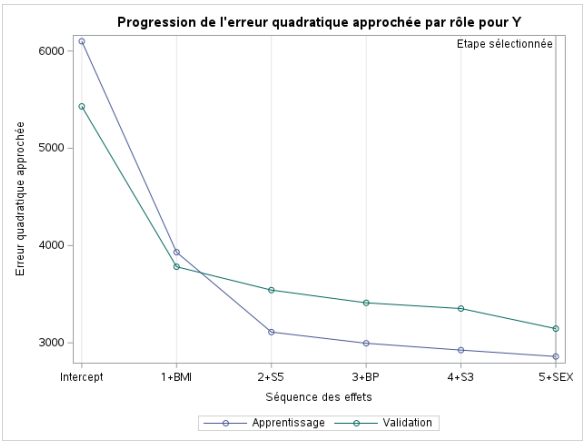
The LASSO algorithm stops early at step 4, retaining only $BMI, S5, BP$, and $S3$. With a Validation-ASE of 2975.80, it produces the simplest model but fails to capture all predictive information.

In comparison, Elastic Net continues the selection process up to step 6. It includes SEX and, most importantly, $S2$, a variable discarded by LASSO. This action reduces the validation ASE to 2816.91.

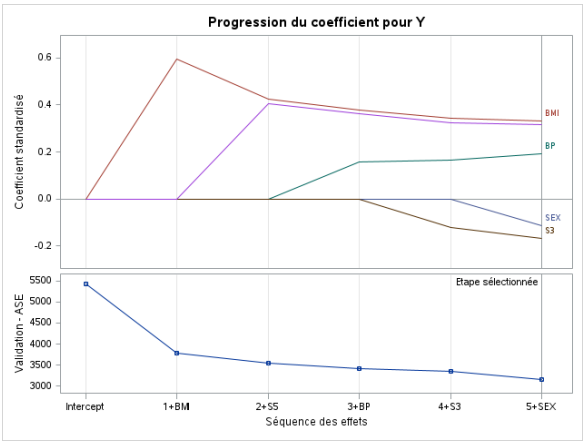
Method	Selected step	Selected variables	Validation-ASE
Stepwise (SBC/BIC)	5	$SEX, BMI, BP, S3, S5$	3146.28353
Stepwise (AIC)	6	$SEX, BMI, BP, S1, S3, S5$	3105.92138
LASSO (SBC + CV)	4	$BMI, BP, S3, S5$	2975.80052
Elastic Net (SBC + CV)	6	$SEX, BMI, BP, S2, S3, S5$	2816.91203

Table 2: Empirical comparison on the Diabetes dataset: selected models and predictive performance.

This illustrates the bias-variance trade-off described in Chapter 2. The ASE curves show that regularization finds a stable minimum before the error increases due to complexity.

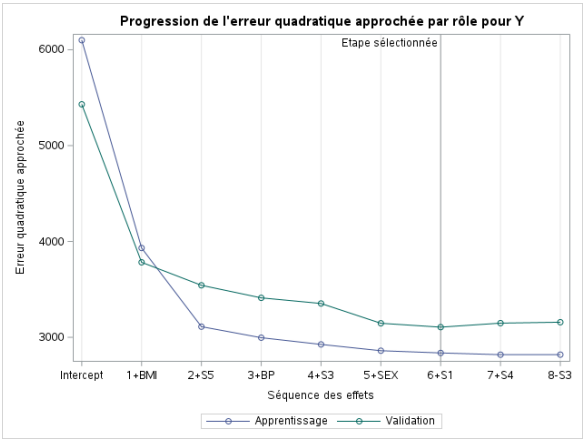


(a) Training vs validation ASE (SBC).

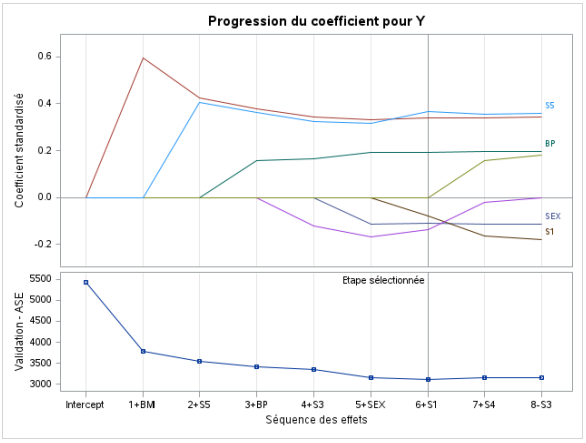


(b) Standardized coefficient path (SBC).

Figure 31: Stepwise selection with SBC/BIC



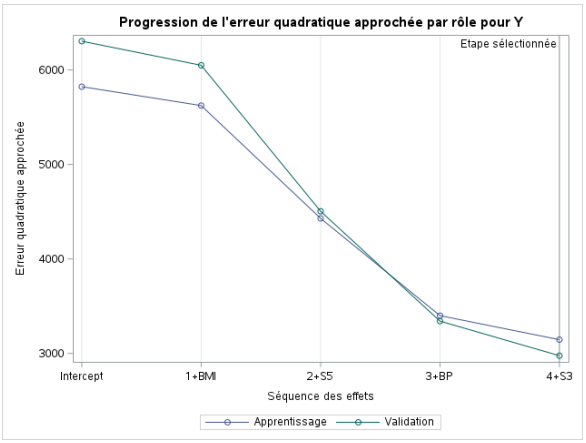
(a) Training vs validation ASE (AIC).



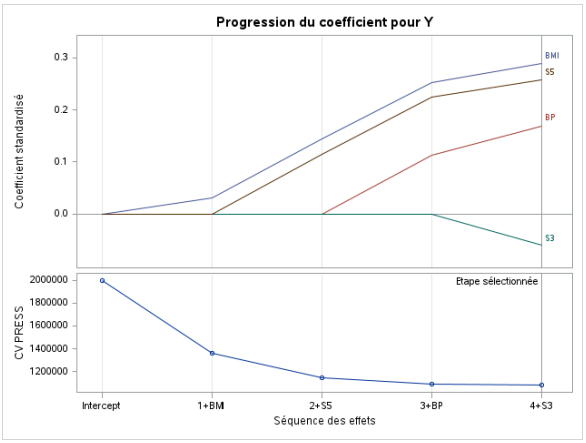
(b) Standardized coefficient path (AIC).

Figure 32: Stepwise selection with AIC

The coefficient paths reveal how each method handles multicollinearity. As shown in the exploratory analysis, $S2$ and $S3$ are correlated. LASSO (L_1) selects only $S3$ and removes the others in order to reduce variance. Elastic Net uses its L_2 penalty to create a "clustering effect," allowing $S2$ to remain alongside $S3$. Elastic Net's better performance proves that these variables contain complementary information that LASSO failed to capture.

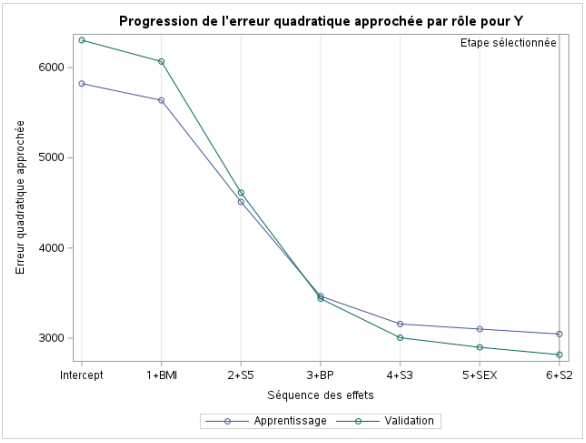


(a) Training vs validation ASE (LASSO path).

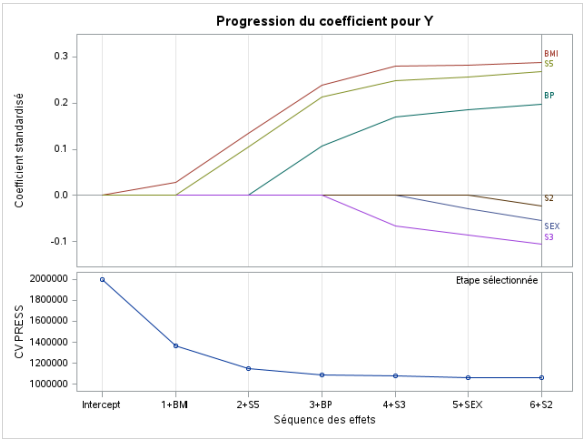


(b) Coefficient path and CV PRESS (LASSO).

Figure 33: LASSO (Stop=SBC, Choose=CV)



(a) Training vs validation ASE (Elastic Net path).



(b) Coefficient path and CV PRESS (Elastic Net).

Figure 34: Elastic Net (Stop=SBC, Choose=CV)

This empirical application confirms the limitations mentioned above. While LASSO produces the most interpretable model (maximum sparsity), it performs less well when the predictors are highly correlated. Elastic Net overcomes this difficulty by retaining a slightly more complex structure (6 variables versus 4), which allows it to offer the best compromise between the parsimony of Stepwise methods and the predictive efficiency of regularization.

6 Conclusion

The research shows that the Stepwise technique stands out for its ability to correct selection errors, far outperforming purely sequential methods. Using the Schwarz criterion (SBC) is key to preventing overfitting, as it keeps the model simple where other criteria fail. Nevertheless, these algorithms have a major issue in dealing with structural breaks. In fact, no approach can accurately define the model, highlighting that data consistency is crucial for effective selection.

The empirical evaluation conducted on the various DGP methods (LASSO, LAR, and Elastic Net) provides an overview of the performance of these methods. Firstly, we observe that Elastic Net is more robust than the other two machine learning methods. Secondly, with LASSO and LAR, perfect fit rates collapse as soon as multicollinearity is introduced, leading to underfitting, whereas Elastic Net, thanks to its penalty, allows more real variables to be selected for the model. Due to the addition of the new variable and the reversal of signs in our DGP, the structural break remains an insurmountable limitation. The work also demonstrates that the choice of stopping criterion is sometimes more decisive than the choice of algorithm itself.

In conclusion, by applying these algorithms to our "Diabetes" dataset, we were able to verify the results of our simulations on more realistic data. Our results demonstrated a real difference between iterative methods (such as Stepwise) and regularization methods (such as LASSO and ElasticNet). Although the latter two had the best ASE scores, we still noticed a tendency for LASSO to be too restrictive, which validates the theoretical work presented in section 2.

In general, our perfectfit rates were very low (or even zero) in most scenarios, highlighting persistent flaws in variable selection and robustness.

References

[1] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.

[2] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), 267–288.

[3] Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1), 55–67.

[4] Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2), 301–320.

[5] Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, 32(2), 407–499.

[6] Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory* (pp. 267–281).

[7] Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2), 461–464.

[8] Hurvich, C. M., & Tsai, C. L. (1989). Regression and time series model selection in small samples. *Biometrika*, 76(2), 297–307.

[9] Mallows, C. L. (1973). Some comments on Cp. *Technometrics*, 15(4), 661–675.

[10] Allen, D. M. (1974). The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16(1), 125–127.

[11] Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2), 111–147.

[12] Wang, R. (s.d.). *Bias-Variance Tradeoff and Ridge Regression*. Harvey Mudd College. Consulted on https://pages.hmc.edu/ruye/MachineLearning/lectures/ch7/node11.html

[13] SAS Institute Inc. (2018). *SAS/STAT 15.2 User's Guide: The GLMSELECT Procedure*. Cary, NC: SAS Institute Inc. Consulted on https://documentation.sas.com/doc/en/statug/15.2/statug_glmselect_syntax01.htm

[14] Wicklin, R. (2021, 14 june). Simulate data from a specified rank correlation with the Iman-Conover transformation. *The DO Loop (SAS Blogs)*. Consulted on https://blogs.sas.com/content/iml/2021/06/14/simulate-iman-conover-transformation.html

A SAS Codes

A.1 Bias/variance tradeoff simulation

```
proc iml;
call randseed(12345);

N = 100;
N_Sim = 1000;
sigma = 0.5;

X_eval = do(-0.9, 0.9, 0.1)‘;
Y_true_eval = 2*(X_eval##2) + 1;

Pred_M0 = j(nrow(X_eval), N_Sim, 0);
Pred_M2 = j(nrow(X_eval), N_Sim, 0);
Pred_M8 = j(nrow(X_eval), N_Sim, 0);

do rep = 1 to N_Sim;

    X = j(N, 1);
    call randgen(X, "Uniform");
    X = X*2 - 1;

    Eps = j(N, 1);
    call randgen(Eps, "Normal");
    Eps = Eps * sigma;

    Y = 2*(X##2) + 1 + Eps;

    X_d0 = j(N, 1, 1);

    X_d2 = X_d0 || X || (X##2);

    X_d8 = X_d2;
    do p = 3 to 8; X_d8 = X_d8 || (X##p); end;

    Beta0 = inv(X_d0‘ * X_d0) * X_d0‘ * Y;
    Beta2 = inv(X_d2‘ * X_d2) * X_d2‘ * Y;
    Beta8 = ginv(X_d8‘ * X_d8) * X_d8‘ * Y;

    X_eval_d0 = j(nrow(X_eval), 1, 1);
```

```
    X_eval_d2 = X_eval_d0 || X_eval || (X_eval##2);
    X_eval_d8 = X_eval_d2;
    do p = 3 to 8; X_eval_d8 = X_eval_d8 || (X_eval##p); end;

    Pred_M0[,rep] = X_eval_d0 * Beta0;
    Pred_M2[,rep] = X_eval_d2 * Beta2;
    Pred_M8[,rep] = X_eval_d8 * Beta8;

end;

Mean_Pred0 = Pred_M0[,:];
Mean_Pred2 = Pred_M2[,:];
Mean_Pred8 = Pred_M8[,:];

BiasSq_0 = (Mean_Pred0 - Y_true_eval)##2;
BiasSq_2 = (Mean_Pred2 - Y_true_eval)##2;
BiasSq_8 = (Mean_Pred8 - Y_true_eval)##2;

Var_0 = j(nrow(X_eval), 1, 0);
Var_2 = j(nrow(X_eval), 1, 0);
Var_8 = j(nrow(X_eval), 1, 0);

do i = 1 to nrow(X_eval);
   Var_0[i] = var(Pred_M0[i,]`);
   Var_2[i] = var(Pred_M2[i,]`);
   Var_8[i] = var(Pred_M8[i,]`);
end;

Resultats = (BiasSq_0[:] || Var_0[:]) //
            (BiasSq_2[:] || Var_2[:]) //
            (BiasSq_8[:] || Var_8[:]);

RowNames = {"Degre 0 (Under)", "Degre 2 (Bon)", "Degre 8 (Over)"};
ColNames = {"Biais_Carre", "Variance"};

print "--- COMPROMIS BIAIS VARIANCE (Degr s 0, 2, 8) ---";
print Resultats[rowname=RowNames colname=ColNames];

PlotData = X_eval || Y_true_eval || Pred_M0[,N_Sim] ||
   Pred_M2[,N_Sim] || Pred_M8[,N_Sim];
```

```
create PlotDS from PlotData[colname={"X" "True" "Deg0" "Deg2"
   "Deg8"}];
append from PlotData;
close PlotDS;

quit;

title "Illustration Overfitting (Deg 8) vs Underfitting (Deg 0)";
proc sgplot data=PlotDS;

    series x=X y=True / lineattrs=(color=black thickness=2)
       legendlabel="Vrai Mod le";

    series x=X y=Deg0 / lineattrs=(color=blue pattern=dash)
       legendlabel="Underfitting (Deg 0)";

    series x=X y=Deg2 / lineattrs=(color=green) legendlabel="Bon
       Mod le (Deg 2)";

    series x=X y=Deg8 / lineattrs=(color=red)
       legendlabel="Overfitting (Deg 8)";

    scatter x=X y=True / markerattrs=(symbol=circlefilled size=5
       color=black);
run;
```

Listing 1: SAS Code for bias/variance tradeoff

A.2 Monte Carlo Simulation (DGP)

```
proc datasets lib=work nolist;
    delete DGP_Gaussien Tableau_Recapitulatif ParEst Statistiques
       Recap_Ligne;
quit;

proc iml;

call randseed(12345);
n  = 200;
p  = 50;
MC = 1000;

```

```
Beta = j(p, 1, 0);
Beta[1:6] = {1, -0.35, 0.15, 0.27, 0.57, -0.14};

Mu       = j(1, p, 0);
Varcovar = I(p);

simulated_data = j(n*MC, 2+p, .);
a = 1;

do iteration = 1 to MC;
  X   = RandNormal(n, Mu, Varcovar);
  eps = normal(j(n, 1, 0))*0.1;
  Y   = X * Beta + eps;

  simulated_data[a:a+n-1, 1]     = iteration;
  simulated_data[a:a+n-1, 2]     = Y;
  simulated_data[a:a+n-1, 3:2+p] = X;
  a = a + n;
end;

cname = {"Iteration_ID" "Y"};
do i = 1 to p;
  cname = cname || cats("X", i);
end;

create DGP_Gaussien from simulated_data[colname=cname];
append from simulated_data;
close DGP_Gaussien;

quit;
```

Listing 2: SAS Code for Gaussian Simulation

```
proc iml;
call randseed(12345);

N  = 200;
P  = 50;
MC = 1000;

Beta = j(P, 1, 0);
Beta[1:6] = {1, -0.35, 0.15, 0.27, 0.57, -0.14};
```

```
10
11 Mu = j(1, P, 0);
12
13 r = {1, 0.8, 0.75, 0.7, 0.65, 0.6};
14 V1 = toeplitz(r);
15 Varcovar = I(P);
16 Varcovar[1:6, 1:6] = V1;
17
18 simulated_data = j(N*MC, 2+P, .);
19 a = 1;
20
21 do iteration = 1 to MC;
22   X = RandNormal(N, Mu, Varcovar);
23
24     eps = normal(j(N,1,0)) * 0.1;
25     Y   = X * Beta + eps;
26
27     simulated_data[a:a+N-1, 1]     = iteration;
28     simulated_data[a:a+N-1, 2]     = Y;
29     simulated_data[a:a+N-1, 3:2+P] = X;
30
31     a = a + N;
32 end;
33
34 cname = {"Iteration_ID" "Y"};
35 do i = 1 to P;
36     cname = cname || cats("X", i);
37 end;
38
39 create DGP_Multicolinearite from simulated_data[colname=cname];
40 append from simulated_data;
41 close DGP_Multicolinearite;
42
43 quit;
```

Listing 3: SAS Code for Multicollinearity Simulation

```
1
2 proc iml;
3 start Outliers(X_in);
4   X_out = X_in;
5   n_obs = nrow(X_in);
```

```
    p_var = ncol(X_in);


  do i = 1 to n_obs ;
    do j = 1 to p_var;
      u = uniform(0);
      if u > 0.90 then do;
        X_out[i,j] = 4 + normal(0);
      end;
    end;
  end;
  return(X_out);
finish;

store module=(Outliers);
load module=( Outliers);
call randseed(12345);

N = 200;
P = 50;
MC = 1000;

Mu = j(1, p, 0);
Varcovar = I(p);

Beta = j(P, 1, 0);
Beta[1:6] = {1, -0.35, 0.15, 0.27, 0.57, -0.14};

simulated_data = j(N*MC, 2+P, .);
a = 1;

do iteration = 1 to MC;

    X_0     = RandNormal(N, Mu, Varcovar);
    X_final = Outliers(X_0);

    eps = normal(j(N, 1, 0)) * 0.1;
    Y   = X_final * Beta + eps;

    simulated_data[a:a+N-1, 1]     = iteration;
    simulated_data[a:a+N-1, 2]     = Y;
```

```
    simulated_data[a:a+N-1, 3:2+P] = X_final;

    a = a + N;
end;

cname = {"Iteration_ID" "Y"};
do i = 1 to P;
    cname = cname || cats("X", i);
end;

create DGP_Outliers from simulated_data[colname=cname];
append from simulated_data;
close DGP_Outliers;

quit;
```

Listing 4: SAS code for the outlier DGP simulation

```

proc iml;
call randseed(12345);
n  = 200;
P  = 50;
MC = 1000;

Beta1 = j(P, 1, 0);
Beta1[1:6] = {1, -0.35, 0.15, 0.27, 0.57, -0.14};

Beta2 = j(P, 1, 0);
Beta2[1:7] = {-1, 0.55, 0.6, -0.35, 0.15, 0.45, 0.1};

Mu       = j(1, P, 0);
Varcovar = I(P);

simulated_data = j(n*MC, 2+P, .);
a = 1;

do iteration = 1 to MC;

    t_break = rand("Integer", floor(0.1*n), floor(0.7*n));

    X   = RandNormal(n, Mu, Varcovar);
```

```
    eps = normal(j(n, 1, 0)) * 0.1;
    Y   = j(n, 1, .);

    Y[1:t_break]           = X[1:t_break, ] * Beta1 +
       eps[1:t_break];
    Y[(t_break+1):n]       = X[(t_break+1):n, ] * Beta2 +
       eps[(t_break+1):n];

    simulated_data[a:a+n-1, 1]     = iteration;
    simulated_data[a:a+n-1, 2]     = Y;
    simulated_data[a:a+n-1, 3:2+P] = X;

    a = a + n;
end;

cname = {"Iteration_ID" "Y"};
do i = 1 to P;
    cname = cname || cats("X", i);
end;

create DGP_Rupture from simulated_data[colname=cname];
append from simulated_data;
close DGP_Rupture;
quit;
```

Listing 5: SAS code for the structural break DGP simulation

```

proc iml;

    load module=(Outliers);

    call randseed(12345);


    n  = 200;
    P  = 50;
    MC = 1000;

    Mu       = j(1, P, 0);
    Varcovar = I(P);

```

```
16
17      Beta1 = j(P, 1, 0);
18      Beta1[1:6] = {1, -0.35, 0.15, 0.27, 0.57, -0.14};
19
20      Beta2 = j(P, 1, 0);
21      Beta2[1:7] = {-1, 0.55, 0.6, -0.35, 0.15, 0.45, 0.1};
22
23      simulated_data = j(n*MC, 2+P, .);
24      a = 1;
25
26      do iteration = 1 to MC;
27
28          t_break = rand("Integer", floor(0.1*n), floor(0.7*n));
29
30          X0  = RandNormal(n, Mu, Varcovar);
31          X   = Outliers(X0);
32
33          eps = normal(j(n, 1, 0)) * 0.1;
34          Y   = j(n, 1, .);
35
36          Y[1:t_break]     = X[1:t_break, ] * Beta1 + eps[1:t_break];
37          Y[(t_break+1):n] = X[(t_break+1):n, ] * Beta2 +
               eps[(t_break+1):n];
38
39          simulated_data[a:a+n-1, 1]     = iteration;
40          simulated_data[a:a+n-1, 2]     = Y;
41          simulated_data[a:a+n-1, 3:2+P] = X;
42
43          a = a + n;
44      end;
45
46      cname = {"Iteration_ID" "Y"};
47      do i = 1 to P;
48          cname = cname || cats("X", i);
49      end;
50
51      create DGP_BreakOutliers from simulated_data[colname=cname];
52      append from simulated_data;
53      close DGP_BreakOutliers;
54
55  quit;
```

Listing 6: SAS code for the mix break/outliers DGP simulation

```
1
2
3  proc iml;
4  start ImanConover(X, C);
5  n_obs = nrow(X);
6  p_var = ncol(X);
7  S = J(n_obs, p_var);
8
9  do i = 1 to ncol(X);
10 ranks = ranktie(X[,i], "mean");
11 S[,i] = quantile("Normal", ranks/(n_obs+1));
12 end;
13
14 CS = corr(S);
15 Q = root(CS);
16 P = root(C);
17
18 T = solve(Q,P);
19 Y = S*T;
20
21 W = X;
22 do i = 1 to ncol(Y);
23 r_vec = rank(Y[,i]);
24 tmp = W[,i];
25 call sort(tmp);
26 W[,i] = tmp[r_vec];
27 end;
28
29 return( W );
30 finish;
31
32 store module=(ImanConover);
33
34 load module=(ImanConover Outliers);
35
36 call randseed(12345);
37
38 N   = 200;
39 P   = 50;
```

```
MC = 1000;

Mu       = j(1, P, 0);
Varcovar = I(P);

Beta = j(P, 1, 0);
Beta[1:6] = { 1, -0.35, 0.15, 0.27, 0.57, -0.14 };

rho = {1, 0.8, 0.75, 0.7, 0.65, 0.6};
C = toeplitz(rho);

simulated_data = j(N*MC, 2+P, .);
a = 1;

do iteration = 1 to MC;

  X_indep = RandNormal(N, Mu, Varcovar);

  X_outl = Outliers(X_indep);

  X1 = ImanConover(X_outl[, 1:6], C);
    X_final = X1 || X_outl[, 7:P];

  eps = normal(j(N, 1, 0)) * 0.1;
  Y   = X_final * Beta + eps;

  simulated_data[a:a+N-1, 1]     = iteration;
  simulated_data[a:a+N-1, 2]     = Y;
  simulated_data[a:a+N-1, 3:2+P] = X_final;

  a = a + N;
  end;

cname = {"Iteration_ID" "Y"} || ("X" + strip(char(1:P)));
create DGP_CorrOutliers from simulated_data[colname=cname];
append from simulated_data;
close DGP_CorrOutliers;

quit;
```

Listing 7: SAS code for the mix multicollinearity/outliers DGP simulation

```
1
2  proc iml;
3  start ImanConover(X, C);
4  n_obs = nrow(X);
5  p_var = ncol(X);
6  S = J(n_obs, p_var);
7
8  do i = 1 to ncol(X);
9  ranks = ranktie(X[,i], "mean");
10 S[,i] = quantile("Normal", ranks/(n_obs+1));
11 end;
12
13 CS = corr(S);
14 Q = root(CS);
15 P = root(C);
16 T = solve(Q,P);
17 Y = S*T;
18
19 W = X;
20 do i = 1 to ncol(Y);
21 r_vec = rank(Y[,i]);
22 tmp = W[,i];
23 call sort(tmp);
24 W[,i] = tmp[r_vec];
25 end;
26
27 return( W );
28 finish;
29
30 load module=(ImanConover);
31
32 call randseed(12345);
33
34 N  = 200;
35 P  = 50;
36 MC = 1000;
37
38 Mu       = j(1, P, 0);
39 Varcovar = I(P);
40
41 Beta1 = j(P, 1, 0);
```

```
Beta1[1:6] = {1, -0.35, 0.15, 0.27, 0.57, -0.14};

Beta2 = j(P, 1, 0);
Beta2[1:7] = {-1, 0.55, 0.6, -0.35, 0.15, 0.45, 0.1};
rho = {1, 0.8, 0.75, 0.7, 0.65, 0.6};
C = toeplitz(rho);

simulated_data = j(N*MC, 2+P, .);
a = 1;

do iteration = 1 to MC;

  t_break = rand("Integer", floor(0.1*n), floor(0.7*n));

  X_indep = RandNormal(N, Mu, Varcovar);
  X_corr = ImanConover(X_indep[, 1:6], C);
    X_final = X_corr || X_indep[, 7:P];

    eps = normal(j(N, 1, 0)) * 0.1;

  Y   = j(n, 1, .);

    Y[1:t_break]     = X_final[1:t_break, ] * Beta1 +
       eps[1:t_break];
  Y[(t_break+1):n] = X_final[(t_break+1):n, ] * Beta2 +
     eps[(t_break+1):n];

  simulated_data[a:a+N-1, 1]     = iteration;
  simulated_data[a:a+N-1, 2]     = Y;
  simulated_data[a:a+N-1, 3:2+P] = X_final;

  a = a + N;
  end;

cname = {"Iteration_ID" "Y"} || ("X" + strip(char(1:P)));
create DGP_CorrBreak from simulated_data[colname=cname];
append from simulated_data;
close DGP_CorrBreak;

quit;
```

Listing 8: SAS code for the mix multicollinearity/break DGP simulation

```
1
2  proc iml;
3  start ImanConover(X, C);
4  n_obs = nrow(X);
5  p_var = ncol(X);
6  S = J(n_obs, p_var);
7
8  do i = 1 to ncol(X);
9  ranks = ranktie(X[,i], "mean");
10 S[,i] = quantile("Normal", ranks/(n_obs+1));
11 end;
12
13 CS = corr(S);
14 Q = root(CS);
15 P = root(C);
16
17 T = solve(Q,P);
18 Y = S*T;
19
20 W = X;
21 do i = 1 to ncol(Y);
22 r_vec = rank(Y[,i]);
23 tmp = W[,i];
24 call sort(tmp);
25 W[,i] = tmp[r_vec];
26 end;
27
28 return( W );
29 finish;
30
31 store module=(ImanConover);
32
33 load module=(ImanConover Outliers);
34
35 call randseed(12345);
36
37 N  = 200;
38 P  = 50;
39 MC = 1000;
40
41 Mu       = j(1, P, 0);
```

```
Varcovar = I(P);

Beta1 = j(P, 1, 0);
Beta1[1:6] = {1, -0.35, 0.15, 0.27, 0.57, -0.14};

Beta2 = j(P, 1, 0);
Beta2[1:7] = {-1, 0.55, 0.6, -0.35, 0.15, 0.45, 0.1};

rho = {1, 0.8, 0.75, 0.7, 0.65, 0.6};
C = toeplitz(rho);

simulated_data = j(N*MC, 2+P, .);
a = 1;

do iteration = 1 to MC;

  t_break = rand("Integer", floor(0.1*n), floor(0.7*n));

  X_indep = RandNormal(N, Mu, Varcovar);
  X_outl = Outliers(X_indep);
  X1 = ImanConover(X_outl[, 1:6], C);
    X_final = X1 || X_outl[, 7:P];

  eps = normal(j(N, 1, 0)) * 0.1;

  Y   = j(n, 1, .);

  Y[1:t_break]     = X_final[1:t_break, ] * Beta1 + eps[1:t_break];
  Y[(t_break+1):n] = X_final[(t_break+1):n, ] * Beta2 +
     eps[(t_break+1):n];

  simulated_data[a:a+N-1, 1]     = iteration;
  simulated_data[a:a+N-1, 2]     = Y;
  simulated_data[a:a+N-1, 3:2+P] = X_final;

  a = a + N;
  end;

cname = {"Iteration_ID" "Y"} || ("X" + strip(char(1:P)));
create DGP_CorrOutliersBreak from simulated_data[colname=cname];
append from simulated_data;
```

```
close DGP_CorrOutliersBreak;

quit;
```

Listing 9: SAS code for the combined scenario simulation

A.3 Variable Selection with PROC GLMSELECT

```
%macro Metrique(data_in, methode, choose, stop, sle_val=0.15);

  ods exclude all;
  ods output ParameterEstimates = ParEst;

  proc glmselect data=&data_in. plots=none;
      by Iteration_ID;
      model Y = X1-X50 / selection=&methode. (choose=&choose.
         stop=&stop. sle=&sle_val.);
  run;

  ods exclude none;

  proc iml;
      vrai = {"X1" "X2" "X3" "X4" "X5" "X6"};

      use ParEst;
          read all var {"Iteration_ID"} into IDs;
          read all var {"Parameter"} into Choisi;
      close;

      uniqueIDs = unique(IDs);
      nSim = ncol(uniqueIDs);

      results = j(nSim, 3, .);

      do i = 1 to nSim;
          currentID = uniqueIDs[i];

          idx = loc(IDs = currentID & upcase(Choisi) ^=
             "INTERCEPT");

          if ncol(idx) > 0 then do;
              vars_selectionnees   = upcase(Choisi[idx]‘);
```

```
33              Faux_Positifs_list   = setdif(vars_selectionnees,
                   vrai);
34              Faux_Negatifs_list   = setdif(vrai,
                   vars_selectionnees);
35          end;
36          else do;
37              vars_selectionnees   = { };
38              Faux_Positifs_list   = { };
39              Faux_Negatifs_list   = vrai;
40          end;
41
42          results[i, 1] = currentID;
43          results[i, 2] = ncol(Faux_Positifs_list);
44          results[i, 3] = ncol(Faux_Negatifs_list);
45      end;
46
47      create Statistiques from results[colname={"Iteration_ID"
           "Faux_Positifs" "Faux_Negatifs"}];
48      append from results;
49      close Statistiques;
50  quit;
51
52  proc sql;
53      create table Recap_Ligne as
54      select
55          "&methode" as Methode length=20,
56          "&choose"  as Choose  length=20,
57          case
58              when "&stop" = "SL" then "SL=" || put(&sle_val, 5.2)
59              else "&stop"
60          end as Stop length=20,
61
62          sum(case when Faux_Positifs=0 and Faux_Negatifs=0 then 1
               else 0 end) / count(*) as Perfectfit
               format=percent8.2,
63          sum(case when Faux_Positifs>0 and Faux_Negatifs=0 then 1
               else 0 end) / count(*) as Overfit
               format=percent8.2,
64          sum(case when Faux_Positifs=0 and Faux_Negatifs>0 then 1
               else 0 end) / count(*) as Underfit
               format=percent8.2,
```

```
          sum(case when Faux_Positifs>0 and Faux_Negatifs>0 then 1
             else 0 end) / count(*) as Wrong      format=percent8.2
      from Statistiques;
  quit;

  proc append base=Tableau_Recapitulatif data=Recap_Ligne force;
  run;

%mend;
```

Listing 10: Metric Macro for the Gaussian, Multicollinearity, and Outliers DGPs

```

%macro Metrique(data_in, methode, choose, stop, sle_val=0.15);

  ods exclude all;
  ods output ParameterEstimates = ParEst;

  proc glmselect data=&data_in. plots=none;
      by Iteration_ID;
      model Y = X1-X50 / selection=&methode. (choose=&choose.
         stop=&stop. sle=&sle_val.);
  run;

  ods exclude none;

  proc iml;
      vrai = {"X1" "X2" "X3" "X4" "X5" "X6" "X7"};

      use ParEst;
          read all var {"Iteration_ID"} into IDs;
          read all var {"Parameter"} into Choisi;
      close;

      uniqueIDs = unique(IDs);
      nSim = ncol(uniqueIDs);

      results = j(nSim, 3, .);

      do i = 1 to nSim;
          currentID = uniqueIDs[i];

```

```
30          idx = loc(IDs = currentID & upcase(Choisi) ^=
               "INTERCEPT");
31
32          if ncol(idx) > 0 then do;
33              vars_selectionnees   = upcase(Choisi[idx]‘);
34              Faux_Positifs_list   = setdif(vars_selectionnees,
                   vrai);
35              Faux_Negatifs_list   = setdif(vrai,
                   vars_selectionnees);
36          end;
37          else do;
38              vars_selectionnees   = { };
39              Faux_Positifs_list   = { };
40              Faux_Negatifs_list   = vrai;
41          end;
42
43          results[i, 1] = currentID;
44          results[i, 2] = ncol(Faux_Positifs_list);
45          results[i, 3] = ncol(Faux_Negatifs_list);
46      end;
47
48      create Statistiques from results[colname={"Iteration_ID"
           "Faux_Positifs" "Faux_Negatifs"}];
49      append from results;
50      close Statistiques;
51  quit;
52
53  proc sql;
54      create table Recap_Ligne as
55      select
56          "&methode" as Methode length=20,
57          "&choose"  as Choose  length=20,
58          case
59              when "&stop" = "SL" then "SL=" || put(&sle_val, 5.2)
60              else "&stop"
61          end as Stop length=20,
62
63          sum(case when Faux_Positifs=0 and Faux_Negatifs=0 then 1
               else 0 end) / count(*) as Perfectfit
               format=percent8.2,
64          sum(case when Faux_Positifs>0 and Faux_Negatifs=0 then 1
```

```
            else 0 end) / count(*) as Overfit
            format=percent8.2,
        sum(case when Faux_Positifs=0 and Faux_Negatifs>0 then 1
            else 0 end) / count(*) as Underfit
            format=percent8.2,
        sum(case when Faux_Positifs>0 and Faux_Negatifs>0 then 1
            else 0 end) / count(*) as Wrong      format=percent8.2
    from Statistiques;
  quit;

  proc append base=Tableau_Recapitulatif data=Recap_Ligne force;
  run;

%mend;
```

Listing 11: Metric Macro for the structural break DGPs

A.4 Empirical case-study simulation

```
data WORK.DIABETES;
    infile "/home/u64142266/Modelisation Stoch/Analyse
       Empirique/Diabetes.txt"
            dlm='09'x
            firstobs=3
            missover;

    input AGE SEX BMI BP S1 S2 S3 S4 S5 S6 Y_char :$20.;

    Y = input(compress(Y_char, '\;'), best.);

    drop Y_char;
run;


title "1. Selection STEPWISE avec critere AIC";
proc glmselect data=WORK.DIABETES plots=(CriterionPanel ASEPlot
   CoefficientPanel) seed=12345;
    partition fraction(validate=0.3);

    model Y = AGE SEX BMI BP S1 S2 S3 S4 S5 S6 /
          selection=stepwise(select=AIC choose=validate)
          stats=all;
```

```
run;

title "2. Selection STEPWISE avec critere SBC";
proc glmselect data=WORK.DIABETES plots=(CriterionPanel ASEPlot
   CoefficientPanel) seed=12345;
    partition fraction(validate=0.3);

    model Y = AGE SEX BMI BP S1 S2 S3 S4 S5 S6 /
          selection=stepwise(select=SBC choose=validate)
          stats=all;
run;


title "3. Regression LASSO";
proc glmselect data=WORK.DIABETES plots=(CriterionPanel ASEPlot
   CoefficientPanel) seed=12345;
    partition fraction(validate=0.3);

    model Y = AGE SEX BMI BP S1 S2 S3 S4 S5 S6 /
          selection=lasso(stop=CV choose=CV)
          cvmethod=random(10);
run;

title "4. Regression ELASTIC NET";
proc glmselect data=WORK.DIABETES plots=(CriterionPanel ASEPlot
   CoefficientPanel) seed=12345;
    partition fraction(validate=0.3);

    model Y = AGE SEX BMI BP S1 S2 S3 S4 S5 S6 /
          selection=elasticnet(stop=CV choose=CV)
          cvmethod=random(10);
run;

title;
```

Listing 12: SAS code for Exploratory Data Analysis

```
data DIABETES;
    infile "/home/u64142266/Modelisation Stoch/Analyse
       Empirique/Diabetes.txt" dlm='09'x firstobs=3;
    input AGE SEX BMI BP S1 S2 S3 S4 S5 S6 Y;
run;
```

```
5
6
7  ods graphics on;
8
9  proc corr data=DIABETES;
10     var AGE -- S6;
11 run;
12
13
14 proc iml;
15     use DIABETES;
16     read all var {AGE SEX BMI BP S1 S2 S3 S4 S5 S6} into MatX;
17     close DIABETES;
18
19     R = corr(MatX);
20
21     noms = {AGE SEX BMI BP S1 S2 S3 S4 S5 S6};
22     print R[colname=noms rowname=noms format=5.2];
23 quit;
```

Listing 13: SAS code for Correlation Matrix

```
1  data DIABETES;
2      infile "/home/u64142266/Modelisation Stoch/Analyse
           Empirique/Diabetes.txt" dlm='09'x firstobs=3;
3      input AGE SEX BMI BP S1 S2 S3 S4 S5 S6 Y;
4  run;
5
6  proc stdize data=DIABETES out=DIABETES_STD;
7      var AGE -- S6;
8  run;
9
10 data DIABETES_PLOT;
11     set DIABETES_STD;
12     array vars[*] AGE -- S6;
13     do i = 1 to dim(vars);
14         Variable = vname(vars[i]);
15         Value = vars[i];
16         output;
17     end;
18     keep Variable Value;
19 run;
```

```
20
21 title "Variable Distribution (Standardized)";
22 proc sgplot data=DIABETES_PLOT;
23     vbox Value / category=Variable group=Variable;
24     xaxis display=(nolabel);
25     yaxis label="Standardized Value";
26 run;
27 title;
```

Listing 14: SAS code for Boxplots

```
1  data WORK.DIABETES;
2      infile "/home/u64142266/Modelisation Stoch/Analyse
          Empirique/Diabetes.txt"
3          dlm='09'x
4          firstobs=3
5          missover;
6
7      input AGE SEX BMI BP S1 S2 S3 S4 S5 S6 Y_char :$20.;
8
9      Y = input(compress(Y_char, '\;'), best.);
10
11     drop Y_char;
12 run;
13
14
15 title "1. Selection STEPWISE avec critere AIC";
16 proc glmselect data=WORK.DIABETES plots=(CriterionPanel ASEPlot
      CoefficientPanel) seed=12345;
17     partition fraction(validate=0.3);
18
19     model Y = AGE SEX BMI BP S1 S2 S3 S4 S5 S6 /
20           selection=stepwise(select=AIC choose=validate)
21           stats=all;
22 run;
23
24 title "2. Selection STEPWISE avec critere SBC";
25 proc glmselect data=WORK.DIABETES plots=(CriterionPanel ASEPlot
      CoefficientPanel) seed=12345;
26     partition fraction(validate=0.3);
27
28     model Y = AGE SEX BMI BP S1 S2 S3 S4 S5 S6 /
```

```
          selection=stepwise(select=SBC choose=validate)
          stats=all;
run;


title "3. Regression LASSO";
proc glmselect data=WORK.DIABETES plots=(CriterionPanel ASEPlot
   CoefficientPanel) seed=12345;
    partition fraction(validate=0.3);

    model Y = AGE SEX BMI BP S1 S2 S3 S4 S5 S6 /
          selection=lasso(stop=SBC choose=CV)
          cvmethod=random(10);
run;

title "4. Regression ELASTIC NET";
proc glmselect data=WORK.DIABETES plots=(CriterionPanel ASEPlot
   CoefficientPanel) seed=12345;
    partition fraction(validate=0.3);

    model Y = AGE SEX BMI BP S1 S2 S3 S4 S5 S6 /
          selection=elasticnet(stop=SBC choose=CV)
          cvmethod=random(10);
run;

title;
```

Listing 15: SAS code for Simulations