Decreasing the number of variables is not always that beneficial

It is widely accepted that machine learning methods suffer from too many input variables (or features) being available. It was originally pointed out by R. E. Bellman (1961) that when more variables are added, it gets increasingly (well, exponentially) difficult to tabulate all the possible combinations, and to find representative instances of each. Modern methods do not really require tabulation of all the possibilities, thanks to their ability to generalize, but too many variables are nevertheless considered harmful.

Partly because of this, input variable selection (or feature selection) has been a hot research topic already for decades. It is tempting to assume that by picking an optimal subset from a large pool of candidate variables, more accurate predictors can be created. However, it is not usually the case that the variable selection algorithm could fight the curse of dimensionality any better than the lower-level machine learning method itself can.

Indeed, if one uses only the cross-validation estimates found during the search, it is quite easy to reach the hasty conclusion that accuracy has been improved:

The cross-validation scores are overfitted, because they result from a multiple-comparison procedure (see Jensen and Cohen, 2000).
Results from straightforward outer-loop cross-validation may still be overfitted

A simple solution to the aforementioned problem is that an outer loop of cross-validation is executed. Basically, it gives an (almost) unbiased accuracy estimate for each subset size (number of variables selected). If you have 100 candidate variables, you now have 100 accuracy estimates.

You can use these 100 estimates to pick the optimal subset size, but you must realize that after having done this, the estimate associated to the winner has again been overfitted (because each estimate is a random variable, so we have another multiple-comparison procedure). Therefore, the estimate of the winner should not be used in any report whatsoever that is concerned with the accuracy of the eventual predictor built using a subset of this optimal size.

What if we need an unbiased estimate of the accuracy that can be attained using a subset of the optimal size? This is a requirement that makes a lot of sense, because often getting a valid estimate of the eventual accuracy is highly important.

In order to get such an unbiased estimate, we could add another outer loop of cross-validation, or we can turn to the cross-indexing method described in Publication III (and later generalized in Publication IV). In cross-indexing, we use each fold either to choose the optimal subset size, or to assess the accuracy related to it. This way, the overfitting problem can be largely circumvented.

---

1-3. Perform steps 1–3 of Algorithm 1, including the substeps.
4. for \( k = 1 \) to \( K \), (The indexing loop.)
4.1 This time, compute for each \( i \) the mean of \( N \) estimates:
\[
\hat{x}_{i}^{[k]} = \frac{1}{N} \left( \sum_{j=\alpha}^{K} x_{i}^{(j)} + \sum_{j=\beta}^{k} x_{i}^{(j)} \right),
\]
where \( \alpha = K + k - N + 1 \), and \( \beta = \max(1, k - N + 1) \).
4.2 Use them to select the optimal model complexity:
\[
\tilde{d}_{K}^{[k]} = \arg \max_{d} \left( \hat{x}_{i}^{[k]} \right).
\]
4.3 Then, calculate the average performance at this level of complexity for the other \( K - N \) folds:
\[
\bar{\hat{x}}_{d}^{[k]} = \frac{1}{K-N} \left( \sum_{\ell=1}^{K-N} \hat{x}_{d}^{(\ell)} + \sum_{\ell=k+1}^{K} x_{d}^{(\ell)} \right),
\]
where \( \gamma = \min(K, K + k - N) \).
end:
5. When a performance estimate is needed for the model complexity \( d_{K}^{[k]} \), use \( \bar{\hat{x}}_{d}^{[k]} \).

**Algorithm 4:** Generalized \((N, K - N)\)-fold cross-indexing.
(When \( a > b, \sum_{a}^{b} \cdot = 0 \).)